الجمهورية الجزائرية الديمقراطية الشعبية

PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINIETRY OF HIGHER EDUCATIQN AND SCIENTIFIC RESEARCH

UNIVERSITY 20 AUGUST 1955-SKIKDA

FACULTY OF SCIENCES
DEPARTMENT OF MATHEMATICS

Course in

Numerical methods

For Engineers and scientists

Dr. SELMANI WISSAME

Skikda University

Academic year: **2023 / 2024**

Semester: 4

UEF 2.2.2

Material 1: Mathematics 5 **(**VHS: 45H00, Course: 1h30, TD: 1h30)

Objectives of education:

Familiarization of numerical methods and his applications in different domains.

Recommended background knowledge

Mathematics 1, Mathematics 2 , informatics 1 and 2

Content of the material:

Chapter 1: ROOTS OF NON LINEAR EQUATIONS $f(x) = 0$ **3 weeks**

Introduction about errors, introduction about resolution of nonlinear equations, bisection method, Newton Raphson method, fixed-point method.

Chapter 2: INTERPOLATION **2 weeks**

Lagrange polynomial, Newton polynomial.

Chapter 3: LEAST SQUARE APPROXIMATION **2 weeks**

Quadratic approximation, orthogonal systems and orthogonal polynomials, trigonometric approximation.

Chapter 4: NUMERICAL INTEGRATION **2 weeks**

Trapezoidal method, Simpson method , quadratic formula.

Chapter 5: NUMERICAL METHODS FOR ORDINARY DIFFERENTIAL EQUATIONS **2 weeks**

Euler's method, Euler's method developed (Taylor series method), Runge Kutta method

Chapter 6: SYSTEM OF LINEAR EQUATIONS-DIRECT METHODS **2 weeks**

Gaussian elimination, LU factorization, Cholesky factorization, TDMA algorithm.

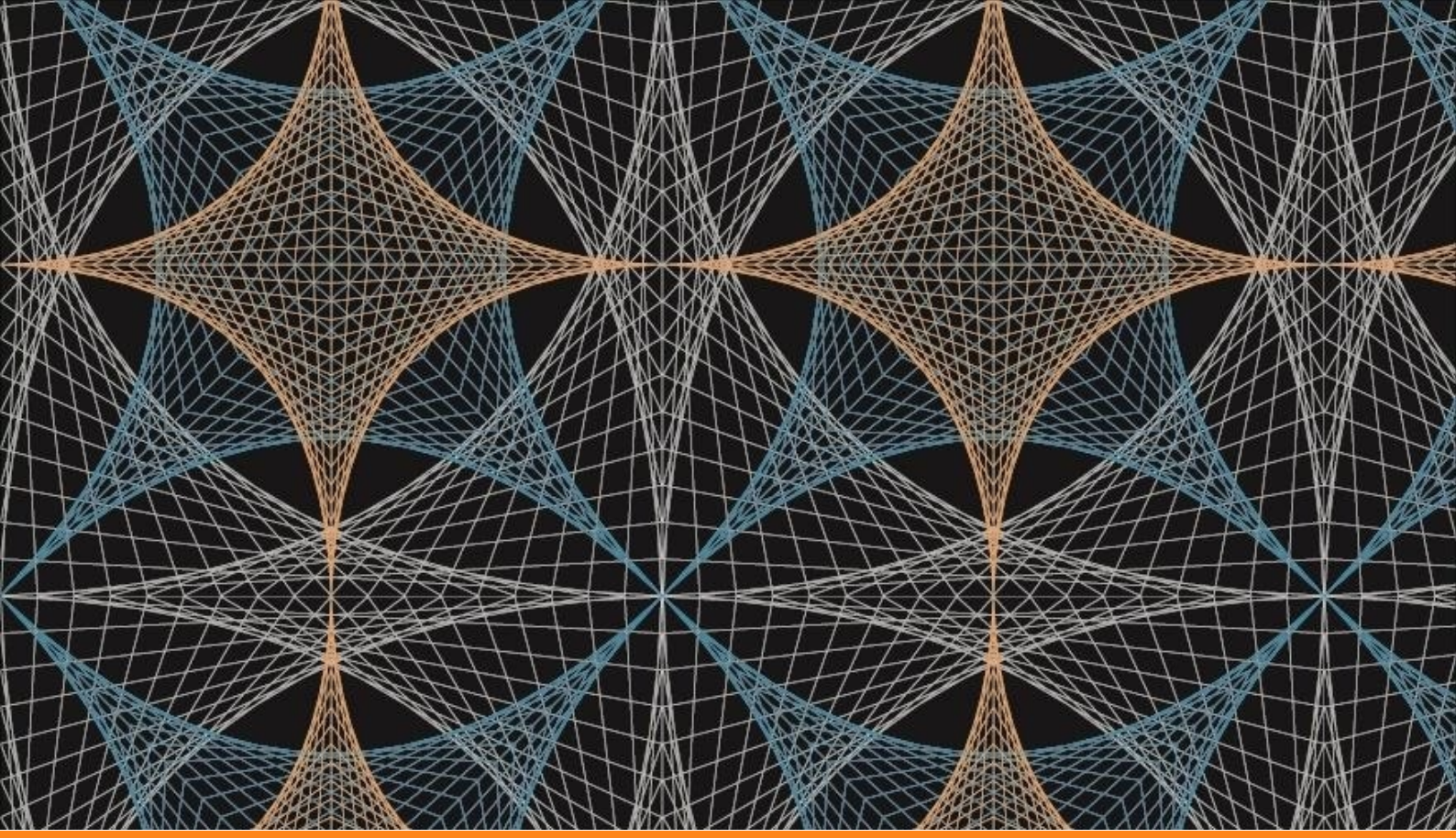
Chapter 7: ITERATIVE METHODS **2 weeks**

Jacobi method, Gauss Seidel method, the use of relaxation.

Evaluation method: Continuous control: 40 %; Final exam: 60 %.

Bibliographical references:

(Depending on the availability of documentation at the establishment level, Websites...etc.)



Numerical Methods for Engineers

Numerical Analysis

Author: SELMANI Wissame

Institute: Department of Mathematics, Skikda University

Date: 2024

Version: Version 1

Contents

The course on numerical methods in mathematics for engineers is a fundamental aspect of engineering education, designed to equip students with tools to solve complex mathematical problems using computational techniques. This course bridges the gap between theoretical concepts and real-world applications by teaching algorithms and strategies to approximate solutions for mathematical models that are otherwise challenging or impossible to solve analytically.

Throughout the course, students delve into numerical techniques such as root finding, interpolation, numerical integration, differential equations, and matrix computations. They learn how to apply these methods using programming languages like Maple , MATLAB, Python, or others, gaining hands-on experience in implementing algorithms and analyzing the accuracy and efficiency of their solutions.

Understanding numerical methods is crucial for engineers as it enables them to simulate, model, and solve problems encountered in various engineering disciplines. It empowers them to tackle real-world challenges where analytical solutions may be impractical or unavailable, allowing for accurate predictions, design optimizations, and informed decision-making in engineering projects.

This course aims to expose the different digital methods for level technicians L2 university. This document contains six (07) chapters and the objectives collectively aim to equip engineering students with a robust foundation in numerical methods, preparing them to tackle intricate engineering problems using computational tools and mathematical techniques.

Course objectives

- ❏ *To familiarize students with fundamental numerical algorithms used for solving mathematical problems encountered in engineering disciplines.*
- ❏ *To enable students to apply numerical techniques to approximate solutions for equations, systems of equations, optimization problems, and differential equations commonly found in engineering scenarios.*
- ❏ *To teach students how to assess and manage errors arising from numerical approximations, ensuring they understand the limitations and accuracy of their solutions.*
- ❏ *To equip students with the ability to identify engineering problems that can be addressed using numerical methods, and to select and apply the most appropriate numerical technique to solve them efficiently and accurately.*
- ❏ *To demonstrate the relevance of numerical methods in engineering by showcasing their applications in various fields such as mechanical engineering, civil engineering, electrical engineering, and more.*
- ❏ *To encourage critical thinking and decision-making by evaluating the suitability*

and limitations of different numerical methods for specific engineering problems.

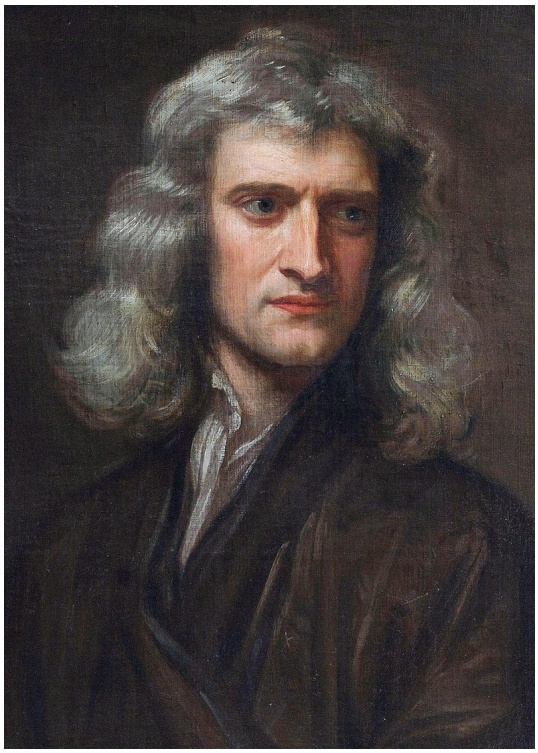
- *To enable students to effectively communicate their findings and solutions, both in written reports and presentations, fostering the ability to convey complex mathematical concepts to non-mathematical audiences within an engineering context.*

Chapter 1 Roots of non linear equations $f(x) = 0$

Note *[Newton]*

Sir Isaac Newton (25 December 1642 - 20 March 1726/27)was an English polymath active as a mathematician, physicist, astronomer, alchemist, theologian, and author who was described in his time as a natural philosopher. He was a key figure in the Scientific Revolution and the Enlightenment that followed. His pioneering book Philosophie Naturalis Principia Mathematica (Mathematical Principles of Natural Philosophy), first published in 1687, consolidated many previous results and established classical mechanics. Newton also made seminal contributions to optics, and shares credit with German mathematician Gottfried Wilhelm Leibniz for developing infinitesimal calculus, though he developed calculus years before Leibniz.

(Source : Wikipédia)



1.1 Introduction

In this chapter we shall discuss one of the oldest approximation problems which consists of finding the roots of an equation. It is also one of the most commonly occurring problems in applied mathematics. The root-finding problem consists of the following: given a continuous function f, find the values of x that satisfy the equation

$$f(x) = 0 \tag{1.1}$$

The solutions of this equation are called the zeros of f or the roots of the equation. In general, it is impossible to solve equation (1.1) exactly. Therefore, to obtain an approximation of the solution we need to use some numerical methods.

The methods we'll talk about in this chapter are iterative and contains basically of two categories: one in which the convergence is assured and the other in which the convergence depends on the initial guess.

Chapter objectives

- *To find the root of a continuous function within a specified interval by repeatedly narrowing down the interval where the root is located by bisection method.*
- *To iteratively approximate the roots of a real-valued function by using tangent lines and rapidly converging towards the actual root by Newton method.*
- *To find the fixed point(s) of a given function, which are the solutions to the equation $f(x) = x$, by iteratively generating a sequence that approaches the fixed point by fixed point method.*

1.2 Round off errors

There are two commonly used ways of representing a given real number x by a floating-point machine number, denoted by $fl(x)$, rounding and chopping. Consider a positive real number x in the normalized decimal form

$$x = (0.b_1b_2...b_kb_{k+1}...)_1010^e.$$

We say that the number x is chopped to k digits when all digits following the $k-th$ digits are discarded; that is, the digits $b_{k+1}b_{k+2}...$ are chopped off to obtain

$$fl(x) = (0.b_1b_2...b_k)_1010^e$$

Conversely, x is rounded to k digits when $fl(x)$ is obtained by choosing $fl(x)$ nearest to x; that is, adding one to b_k if $b_{k+1} \geq 5$ and chop off all but the first k digits if $b_{k+1} < 5$.

Definition 1.1

If $x \neq 0$, the absolute error is given by $|x - fl(x)|$ and the relative error is given by $\frac{|x-fl(x)|}{|x|}$.

Definition 1.2

If x^* is an approximation to x, then we say that x^* approximates x to k significant digits if k is the largest nonnegative integer for which the relative error

$$\left|\frac{x - x^*}{x} < 510^{-k}\right|$$

1.3 Truncation error

A round-off error arises in terminating the decimal expansion and rounding. In contrast, the truncation error terminates a process. So, the truncation error generally refers to the error involved in terminating an infinite sequence or series after a finite number of terms, that is,

replacing noncomputable problems with computable ones. To make the idea precise, consider the familiar infinite Taylor series

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - ....$$

which can be used to approximate the cosine of an angle x in radians.
Let us find, for instance, the value of $\cos(0.5)$ using three terms of the series. We have

$$\cos(0.5) \approx 0.8776041667 = x^*$$

Since $|\cos(0.5) - x^*|/|\cos(0.5)| = 2.461910^{-5} < 510^{-5}$, x^* approximates $\cos(0.5)$ to five significant digits.

1.4 The bisection method

The bisection method is the simplest to numerically execute and is almost always works. Convergence is slow, which is the biggest drawback. Other quicker methods may be used if the bisection method causes a computer program to operate too slowly. otherwise this method is a good choice.(see Figure1)
We want to build a sequence $x_0, x_1, x_2, ...$ that converges to the root $x = c$ that solves $f(x) = 0$.Using the Intermediate Value Theorem, we choose x_0 and x_1 such that $x_0 < c < x_1$. We say that x_0 and x_1 bracket the root. With $f(c) = 0$, we want $f(x_0)$ and $f(x_1)$ to be of opposite sign, so that $f(x_0)f(x_1) < 0$. We then assign x_2 to be the midpoint of x_0 and x_1, that is $x_2 = (x_0 + x_1)/2$. The sign of $f(x_2)$ can then be determined. The value of x_3 is then chosen as either the midpoint of x_0 and x_2 or as the midpoint of x_2 and x_1, depending on whether x_0 and x_2 bracket the root, or x_2 and x_1 bracket the root. The root, therefore, stays bracketed at all times. The process of halving the new interval continues until the root is located as accurately as desired, that is

$$|x_{n+1} - x_n| < \epsilon$$

where x_n and x_{n+1} are the endpoints of the n-th interval $[x_n, x_{n+1}]$ and ϵ is a specified tolerance value.
Example1.1 The function $f(x) = x^3 - x^2 - 1$ has exactly one zero in $[1, 2]$. Use the bisection algorithm to approximate the zero of f to within 10^{-4}.

Since $f(1) = -1 < 0$ and $f(2) = 3 > 0$, Starting with $a_0 = 1$ and $b_0 = 2$, we compute $c_0 = \frac{1+2}{2} = 1.5$ and $f(c_0) = 0.125$

Since $f(1.5)f(2) > 0$, the function changes sign on $[a_0, c_0] = [1, 1.5]$. To continue, we set $a_1 = a_0$ and $b_1 = c_0$; so

$c_1 = \frac{1+1.5}{2} = 1.25$ and $f(c_1) = -0.609375$.

Again $f(1.25)f(1.5) < 0$ so the function changes sign on $[c_1, b_1] = [1.25, 1.5]$. Next we set $a_2 = c_1$ and $b_2 = b_1$. Continuing in this manner leads to the values in Table , which converge to $r = 1.465454$.

1.5 Newton's method

Among the iterative methods for solving equations, Newton's method is one of the most widly. The method uses a tangent line to the curve (see Figure2).

We can use a Taylor series or a grafically to deduce Newton's Method. We want to create a sequence $x_0, x_1, x_2, ...$ that converges to the root $x = c$, consider the x_{n+1} element of this sequence, and let the Taylor series expand $f(x_{n+1})$ about x_n. There are

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

to find x_{n+1} we eliminate the higher-order terms in the Taylor series and assume that $f(x_{n+1}) = 0$. So, we obtain

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n \geq 0 \tag{1.2}$$

Newton's method begin with an initial guess x_0 and provided $f'(x_n)$ is not zero.

Example1.2 Use Newton's method to compute a root of $x^3 - x^2 - 1 = 0$ to an accuracy of 10^{-4}. Use $x_0 = 1$.

The derivative of f is $f'(x) = 3x^2 - 2x$. Using $x_0 = 1$ gives $f(1) = -1$ and $f'(1) = 1$ and so the first Newton iterate is $x_1 = 2$,$f(2) = 3$, $f'(2) = 8$.

The next iteration is $x_2 = 1.625$

, Continuing in this manner leads to the values in Table, which converge to $r = 1.465571$

1.6 Fixed point method

The idea of of the fixed point iteration methods is to formulate an equation to an equivalent fixed point problem :

$$f(x) = 0 \Leftrightarrow x = g(x)$$

and then to use iteration: with an initial guess x_0 chosen,(see Figure3) compute a sequence

$$x_{n+1} = g(x_n), \quad n \geq 0$$

in the hope that $x_n \to c$ when $n \to \infty$

Theorem 1.1. Fixed Point Theorem

If g is continuous on $[a,b]$ and $g(x) \in [a,b]$ for all $x \in [a,b]$, hence g has at least a fixed point in $[a,b]$ Consider, that $g'(x)$ is continuous on $[a,b]$ and K a positive constant exists where $0 < K < 1$.

$$|g'(x)| \leq K < 1 \quad forall \quad x \in [a,b] \tag{1.3}$$

Then there is a unique fixed point c of g in $[a,b]$. Also, the iterates

$$x_{n+1} = g(x_n), \quad n \geq 0$$

will converge to c for any chose of x_0 in $[a,b]$

1.7 order of convergence

1.7.1 bisection method

Let c be the root and x_n be the nth approximation to the root. Define the error as

$$\epsilon_n = c - x_n$$

for large n we have the approximation

$$|\epsilon_{n+1}| = \alpha |\epsilon_n|^k$$

with α a positive constant, then we say the root is of order k. The order of convergence of bisection is one: the error is reduced by approximately a factor of 2 with each iteration so that

$$|\epsilon_{n+1}| = \frac{1}{2}|\epsilon_n|$$

Now, we determine the Newton's Method and fixed point Method's order of convergence.

1.7.2 Newton's method

Definition 1.3

Let $x_0, x_1, x_2, \ldots$ be a sequence that converges to a number c, and $e_n = c - x_n$ k is called the order of convergence of the sequence and α the asymptotic error constant, if there exists a number k and a positive constant α such that

$$\lim_{n \to \infty} \frac{|e_{n+1}|}{|e_n|^k} = \alpha$$

In the following section, we calcul the rate of convergence to simple roots for the Newton method.

From (1.2), we have

$$e_{n+1} = c - x_{n+1} = c - x_n + \frac{f(x_n)}{f'(x_n)} \tag{1.4}$$

if $f \in \mathcal{C}^2(\mathbb{R})$, the Taylor series expansion of f about x_n gives

$$f(c) = f(x_n) + (c - x_n)f'(x_n) + \frac{(c - x_n)^2}{2}f''(\xi_n) \tag{1.5}$$

where ξ_n is located between x_n and c Combining equations (1.4) and (1.5), and since $f(c) = 0$ and $f'(c) \neq 0$ we have

$$e_{n+1} = -(x_n - c)^2 \frac{f''(\xi_n)}{2f'(x_n)} \tag{1.6}$$

$$= -{e_n}^2 \frac{f''(\xi_n)}{2f'(x_n)} \tag{1.7}$$

Therefore, if c is a simple root, that is $f'(c) \neq 0$, (1.7) shows that Newton's method converges quadratically. In other words, for a simple root, Newton's method will usually converge with the number of accurate decimal places approximately doubling in one Newton iteration.

1.8 Conclusion

The conclusion about numerical methods like Newton's method, bisection method, and fixed point method is that they are effective approaches for approximating the roots of non-linear equations:

- Newton's method is a powerful iterative method for finding roots. It starts with an initial guess and uses the derivative of the equation to repeatedly update the guess, converging to the root with every iteration. It can converge rapidly, especially when the initial guess is close to the root and the equation is well-behaved. However, it may fail to converge or converge to a different root if the initial guess is far from the true root or the equation has multiple roots or singularities.
- The bisection method is a simple but reliable numerical method. It involves bracketing the root by identifying two points where the function changes sign. The method repeatedly bisects the interval and selects the subinterval where the function changes sign, converging to the root. The bisection method guarantees convergence, but it may converge slowly compared to other methods. It is more robust in finding the root compared to Newton's method, as it does not rely on an initial guess or the derivative of the equation.
- The fixed point method is another iterative technique for finding roots. It involves transforming the equation into an equivalent form where the root is the fixed point of a function. Then, an initial guess is iteratively updated using a defined iteration formula until convergence is reached. The fixed point method is often used when the original equation is difficult to solve directly. However, it may not always converge or converge to the desired root if the iteration formula is not properly chosen or if the equation has multiple fixed points.

In conclusion, Newton's method, the bisection method, and the fixed point method are valuable tools for approximating the roots of non-linear equations. The choice of method depends on factors such as the equation's properties, available initial guess or interval, desired accuracy, and the computational resources available. It is important to understand the strengths and limitations of each method and apply the most appropriate one for a given problem.

1.9 Annex

1.9.1 MATLAB program

1.9.1.1 bisection method

```
    while 1
clear all;
clc;
```

msg='Enter lower Guess :','Enter higher Guess :';
dtext='','';
title='Input For The Function y=cos(x)';
userinput=inputdlg(msg,title,1,dtext);
if(size(userinput,2) == 0) , disp('good bye'),break;
end
xl=str2double(userinput1);
xu=str2double(userinput2);
iter=0;
while 1
xr = (xl+xu) / 2;
fxl = cos(xl);
fxr = cos(xr);
if(abs(fxr) < 0.0001), break;
elseif((fxl * fxr) < 0)
xu = xr;
else
xl = xr;
end;
end;
iter = iter+1;
end;

tmpstr=sprintf('Result is:
mbox=msgbox(tmpstr,'Result');
uiwait(mbox);
end;

1.9.1.2 Newton method

function NewtonRaphson()

clear all;
clc;

while 1

msg='Your Function:','Enter Initial Guess :','ERROR', 'Number of Iteration: ';
dtext='$x^2 - 9$','2','0.00001','';

```
title='Input For Any Function';
userinput=inputdlg(msg,title,3,dtext);
if(size(userinput,2) == 0) , disp('good bye'),break;
end;

    syms x y d x0 e;

    y = userinput1;
d = diff(y,'x');
x0 = str2double(userinput2);

    error = str2double(userinput3);
iter = str2double(userinput4);
i = 0;
success = 0;
while i < iter
fprintf('————————

    fxi = subs(y,x,x0);
fprintf('fxi ====

    dfxi = subs(d,x,x0);
fprintf('dfxi ====

    if abs(dfxi) < 0.001
success = 0;
m = sprintf('Chose another guess nIts derivative at this point is closed to 0');
msg = msgbox(m,'Error');
uiwait(msg);
end;
a = x0 - (fxi/dfxi);
fprintf('a ====

    e = ((a - x0)/a) * 100;
fprintf('e ====

    if abs(e) < error
success = 1;
```

```
break;
end;

    x0 = a;

    i = i+1;

    end;

    if success
fprintf('The root is :
tmpstr=sprintf('Result is:
mbox=msgbox(tmpstr,'Result');
uiwait(mbox);
else
fprintf('Not Found n');
end;

    end;
```

1.9.1.3 Secant method

```
    function SecantMethod()

    clear all;
clc;
while 1

    msg = 'Enter Your Function : ', 'Enter First Guess: ', 'Enter Second Guess: ', 'Error: ',
'How many Iteration: ';
title = 'Input For Any Function';
dtext = 'x^2-25','2', '4', '0.00001', '50';
userinput = inputdlg(msg, title, 3, dtext);

    if (size(userinput,2) == 0), disp('Good Bye'), break;
end;

    syms x y x0 x1 fxi fx0 ;
```

```
        y = userinput1;
x0 = str2double(userinput2);
x1 = str2double(userinput3);
error = str2double(userinput4);
iter = str2double(userinput5);

        older = x0;
old = x1;

        folder = subs(y,x,older);

        i =0;
success = 0;

        while i < iter

        fold = subs(y,x,old);
dx = fold * ( old - older ) / ( fold - folder );
new = old - dx;

        fprintf ( ’  t  t

        if ( abs(dx) < error )
success = 1;
break;
else
older = old;
old = new;
folder = fold;
end

        i = i+1;

        end;

        if success
fprintf(’The root is :
tmpstr=sprintf(’Result is:
```

```
mbox=msgbox(tmpstr,'Result');
uiwait(mbox);
else
fprintf('Not Found n');
end;

    end;
```

1.10 Exercises

1. Verify that the function $f(x) = x^2 sinx + 2x - 3$ has exactly one root in $[0, 2]$. Find this root by using the bisection method with an error of no more than 10^{-5}.
2. Find a root of $f(x) = x^3 + 2x - 3$ in the range $0 \leq x \leq \frac{7}{5}$ using the bisection method.
3. The function $f(x) = x^2 - 2.6x - 2.31$ has one root in the interval $[3, 4]$. How many steps of the bisection method are needed to locate the root with an error of at most 10^{-5}.
4. The function $f(x) = x^5 - 3x^3 + 2.5x - 0.6$ has exactly one root in $[1, 2]$. Demonstrate it graphically and determine the root with an error of at most 10^{-5} ?
5. Find a root of the equation $e^{2x} - 7x = 0$.
6. Sketch the graph of $f(x) = \tan x + \tanh x = 0$ and find an interval in which its smallest positive root lies. Determine the root correct to two decimal digits using the bisection method.
7. Consider the function
$$f(x) = x^4 - 5x^3 + \frac{22}{3}x^2 - \frac{116}{27}x + \frac{8}{9}.$$
(a) Check that $f(x)$ has a root α_1 between 0 and 1 and another root α_2 between 1 and 4.
(b) Compute both roots using the bisection method.
8. The function $f(x) = x^4 - 8.6x^3 - 35.51x^2 + 464.4x - 998.46$ has a simple root in the interval $[6, 8]$ and a double root in the interval $[4, 5]$. Use the bisection method to find both roots.
9. Find to four decimal places a root of $0.1x^2 - x \ln x = 0$ between 1 and 2.
10. Use the bisection method to find the root of the equation $x + \cos x = 0$ correct to two decimal places.
11. If the bisection method is applied on the interval from $a = 14$ to $b = 16$, how many iterations will be required to guarantee that a root is located to the maximum accuracy of the IEEE double-precision standard?
12. Let n be the number of iterations of the bisection method needed to ensure that a root α is bracketed in an interval $[a, b]$ of length less than . Express n in terms of , a, and b.
13. Consider the function
$$f(x) = (x - 2)^2 - \ln x$$
on the interval $[1, 2]$.
(a) Prove that there is exactly one root of this equation in this interval.
(b) Use the bisection method to approximate a root to 6 digits accuracy.
(c) How many iterates of the Bisection method are needed to find an approximation to the root of $f(x) = 0$ in the interval to within an accuracy of 10^{-4}?
14. Find the points where the hyperbola $y = \frac{1}{x}$, $x > 0$ intersect the curve $y = cosx$.

15. Use the fixed point iteration to find an approximation to $\sqrt{2}$ with 10^{-3} accuracy.
16. The quadratic equation $x^2 - 2x - 3 = 0$ has two roots. Consider the following rearrangements to approximate the roots using the fixed point iteration:
    (a) $x = \sqrt{2x + 3}$
    (b) $x = 3/(x - 2)$,
    (c) $x = (x^2 - 3)/2$
    starting from $x_0 = 4$.
17. Solve the following equations using the fixed point iteration method:
    (a) $x = \sin x + x + 1$, in $[3.5, 5]$,
    (b) $x = \sqrt{x^2 + 1} - x + 1$, in $[0, 3]$,
    (c) $x = \ln x2 + x - 2$, in $[-4, -2]$.
18. For each of the following functions, locate an interval on which fixed point iteration will converge.
    (a) $x = 0.2 \sin x + 1$
    (b) $x = 1 - \frac{x^2}{4}$.
19. Find the solution of the following equations using the fixed point iteration:
    (a) $x = x^5 - 0.25$ starting from $x_0 = 0$,
    (b) $x = 2 \sin x$ starting from $x_0 = 2$,
    (c) $x = \sqrt{3x + 1}$ starting from $x_0 = 2$,
    (d) $x = \frac{2 - e^x + x^2}{3}$ starting from $x_0 = 1$
20. Let $g(x) = \frac{x+4}{2}$.
    (a) Show that $\alpha = 4$ is a fixed point of $g(x)$,
    (b) Let $x_0 = 5$, show that $|\alpha - x_n| = \frac{|\alpha - x_0|}{2n}$ for $n = 1, 2$,...
21. Use the bisection method to the zero of the function

$$f(x) = t^3 + t - 9$$

    correct to one decimal place.
    Give the definition of the fixed point of the iteration

$$s_{n+1} = s_n + \lambda(s_n^3 + s_n - 9);$$

    find an approximate range of values of the constant λ such that the iteration converges to the fixed point near 2.
22. Consider the fixed point iteration scheme: $x_{n+1} = 1 + e^{-x_n}$. Show that this scheme converges for any $x_0 \in [1, 2]$. How many iterations does the theory predict it will take to achieve 10^{-6} accuracy?
23. Consider the fixed point iteration $p_{n+1} = g(p_n)$ when the function $g(x) = 2(x - 1)^{1/2}$ for $x \geq 0$ is used. Plot the function and the line $y = x$ on the same plot and determine how

many fixed points exist. Iterate with $p_0 = 1.5$ and with $p_0 = 2.5$. Explain the result in the light of the fixed point theorem.

24. Approximate to within 10^{-6} the root of the equation $e^-2x - 7x = 0$ in $[\frac{1}{9}, \frac{2}{3}]$ by the secant method.

25. Solve the equation $x^3 - 4x^2 + 2x - 8 = 0$ with an accuracy of 10^{-4} by using the secant method with $x_0 = 3$, $x_1 = 1$.

26. Use the secant method to approximate the solution of the equation $x = x^2 - e^{-x}$ to within 10^{-5} with $x_0 = -1$ and $x_1 = 1$.

27. Use the secant method to approximate the solution of the equation $x = -e^x \sin x - 5$ to within 10^{-10} in the interval $[3, 3.5]$.

28. Given the following equations:

    (a) $x^4 - x - 10 = 0$

    (b) $x - e^{-x} = 0$.

    Determine the initial approximations. Use these to find the roots correct to four decimal places using the secant method.

29. Use the secant method to compute the next two iterates x_2 and x_3 for the zeros of the following functions.

    (a) $f(x) = x^2 - 3x - 1$ with $x_0 = 3.0$, $x_1 = 3.1$,

    (b) $f(x) = x^3 - x - 1$ with $x_0 = 1.2$, $x_1 = 1.3$,

    (c) $f(x) = x^3 - 2x + 1$ with $x_0 = 0.6$, $x_1 = 0.5$.

Chapter 2 Interpolation

Note *[Lagrange]*

Joseph-Louis Lagrange (25 January 1736 - 10 April 1813), was an Italian mathematician, physicist and astronomer, later naturalized French. He made significant contributions to the fields of analysis, number theory, and both classical and celestial mechanics.

Lagrange was one of the creators of the calculus of variations, deriving the Euler-Lagrange equations for extrema of functionals. He extended the method to include possible constraints, arriving at the method of Lagrange multipliers. Lagrange invented the method of solving differential equations known as variation of parameters, applied differential calculus to the theory of probabilities and worked on solutions for algebraic equations. He proved that every natural number is a sum of four squares. His treatise Theorie des fonctions analytiques laid some of the foundations of group theory, anticipating Galois. In calculus, Lagrange developed a novel approach to interpolation and Taylor's theorem. He studied the three-body problem for the Earth, Sun and Moon (1764) and the movement of Jupiter's satellites (1766), and in 1772 found the special-case solutions to this problem that yield what are now known as Lagrangian points. Lagrange is best known for transforming Newtonian mechanics into a branch of analysis, Lagrangian mechanics. He presented the mechanical "principles" as simple results of the variational calculus.

(Source : Wikipédia)

2.1 Introduction

Consider the following problem: Given the values of a known function $y = f(x)$ at a sequence of points $x_0, x_1, ..., x_n$ such that

$$x_0 < x_1 < x_2 < ... < x_n$$

Our objective is to find a polynomial curve that passes through the given points $(x_i, y_i),\ \ i = 0, 1, ..., n$ Therefore, we must find a polynomial $p(x)$ such that

$$p(x_i) = y_i,\ \ i = 0, 1, ..., n \tag{2.1}$$

The problem called interpolation and the points x_i are called the nodes.

In this chapter, two methods will be developed; Newton's method and Lagrange method.

Chapter objectives

- *To approximate data points with a polynomial function that passes through these points by Lagrange polynomial interpolation.*
- *To simplify numerical analysis and computations by approximating complex functions with simpler polynomial representations.*
- *To efficiently interpolate and approximate data points using divided difference coefficients by Newton polynomial interpolation.*
- *simplify the calculations involved in interpolation by using finite differences to generate the polynomial.*

2.2 polynomial interpolation theory

Given the data points (x_i, y_i) where the points are supposed to be distinct, we want to study the problem of finding a polynomial

$$p(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + ... + a_nx^n. \tag{2.2}$$

that interpolates the given data.

Then we can immediately form $n + 1$ linear equations for the $n + 1$ unknown coefficients $a_0, a_1, ..., a_n$ using the $n + 1$ known points and applying condition (2.1):

$$\begin{aligned} a_0 + a_1x_0 + a_2{x_0}^2 + a_3{x_0}^3 + ... + a_nx_0^n &= y_0 \\ a_0 + a_1x_1 + a_2{x_1}^2 + a_3{x_1}^3 + ... + a_nx_1^n &= y_1 \\ \vdots &= \vdots \\ a_0 + a_1x_n + a_2{x_n}^2 + a_3{x_n}^3 + ... + a_nx_n^n &= y_n. \end{aligned} \tag{2.3}$$

The system of equations in matrix form is

$$\begin{pmatrix} x_0^n & x_0^{n-1} & \dots & x_0 & 1 \\ x_1^n & x_1^{n-1} & \dots & x_1 & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_n^n & x_n^{n-1} & \dots & x_n & 1 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

We can whrite

$$Ma = Y \tag{2.4}$$

where

$$M = [x_i^j], \quad i, j = 0, 1, ..., n$$

$$a = [a_0, a_1, ..., a_n]^T, \; Y = [y_0, y_1, ..., y_n]^T \tag{2.5}$$

The matrix M is called the Vandermonde matrix.

Theorem 2.1

Let $n+1$ distinct points $x_0, x_1, ..., x_n$ and $n+1$ arbitrary real values $y_0, y_1, ..., y_n$, there is a unique polynomial $p_n(x)$ of degree $\leq n$ that interpolates the points (x_i, y_i)

♡

Proof The determinant of the matrix M in (2.5) is

$$\det(M) = \prod_{0 \leq j < i \leq n} (x_i - x_j)$$

and $\det(M) \neq 0$ since the points x_i are supposed to be distinct. Thus, there is a unique solution of the system (2.3) , so, there is a unique interpolating polynomial of degree $\leq n$.

2.3 Lagrange polynomial

We will see another way to solve the interpolation problem (2.3) by introducing particular polynoms called Lagrange polynoms.

Definition 2.1

Given a set of $n+1$ nodes $\{x_0, x_1, ..., x_n\}$, which must all be distinct, the Lagrange basis for polynomials of degree $\leq n$ for those nodes is the set of polynomials $\{L_0(x), L_1(x), L_2(x), ..., L_n(x)\}$ defined by

$$L_i(x) = \frac{(x-x_0)(x-x_1)...(x-x_{i-1})(x-x_{i+1})...(x-x_n)}{(x_i-x_0)(x_i-x_1)...(x_i-x_{i-1})(x_i-x_{i+1})...(x_i-x_n)} = \prod_{0 \leq j \leq n, j \neq i} \frac{(x-x_j)}{(x_i-x_j)} \tag{2.6}$$

which take values $L_i(x_j) = 0$ if $i \neq j$ and $L_i(x_j) = 1$ if $i = j$, for $i = 0, 1, ..., n$ and $x \in \mathbb{R}$

♣

The Lagrange interpolation polynomial p at the points (x_i, y_i) for $i = 0, ..., n$ is given by

$$p(x) = \sum_{i=0}^{n} y_i L_i(x) \tag{2.7}$$

2.4 Newton polynomial

The Newton polynomial is somewhat more clever than the Vandermonde polynomial because it results in a system of linear equations that is lower triangular, and therefore can be solved by forward substitution.

First, let's determine the form of the first-degree Newton interpolating polynomial $(n = 1)$,

$$p_1(x) = a_0 + a_1(x - x_0),$$

Let define the following notation

$$f[x_0, x_1, ..., x_n] = \frac{f[x_1, ..., x_n] - f[x_0, ..., x_{n-1}]}{x_n - x_0} \tag{2.8}$$

and $y_n = f(x_i) = f[x_i], \quad i = 0, ..., n.$

we can write

$$p_1(x) = f[x_0] + f[x_0, x_1](x - x_0), \quad or \quad p_1(x) = p_0(x) + f[x_0, x_1](x - x_0) \tag{2.9}$$

Definition 2.2

The first divided difference of $f(x)$ at points x_i and x_{i+1} where $i = \overline{0, n}$, is given by:

$$[x_i, x_{i+1}] = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \tag{2.10}$$

The second divided difference of $f(x)$ at points x_i and x_{i+1} is given by:

$$[x_i, x_{i+1}, x_{i+2}] = \frac{[x_{i+1}, x_{i+2}] - [x_i, x_{i+1}]}{x_{i+2} - x_i} \tag{2.11}$$

In general, the divided difference of order k of $f(x)$ at points x_i and x_{i+1} is;

$$[x_i, x_{i+1}, ..., x_{i+k}] = \frac{[x_{i+1}, ..., x_{i+k}] - [x_i, ..., x_{i+k-1}]}{x_{i+k} - x_i} \tag{2.12}$$

♣

The interpolating polynomial is written in the form

$$p_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + ... + a_n(x - x_0)...(x - x_1) \tag{2.13}$$

where

$$a_0 = f(x_0) = f[x_0] \quad (2.14)$$
$$a_1 = [x_0, x_1] \quad (2.15)$$
$$a_2 = [x_0, x_1, x_2] \quad (2.16)$$
$$... = ... \quad (2.17)$$
$$a_n = [x_0, x_1, ..., x_n] \quad (2.18)$$

Example2.1 Let a function define on $[0, 3]$ by

x_i	0	2	4	6
f_i	-6	0	6	60

We want to find a polynomial that passes through these points.

Step 1: Calculate the divided differences:

x_i	$f(x_i)$	$1^{st}DD$	$2^{nd}DD$	$3^{rd}DD$
0	-6			
		3		
2	0		0	
		3		1
4	6			
			6	
		27		
6	60			

Step 2: Construct the Newton polynomial:

The Newton polynomial is given by:

$$P_3(x) = f(x_0)+[x_0, x_1](x-x_0)+[x_0, x_1, x_2](x-x_0)(x-x_1)+[x_0, x_1, x_2, x_3](x-x_0)(x-x_1)(x-x_2)$$

Substituting the calculated values:

$$\begin{aligned} P_3(x) &= -6 + 3(x-0) + 0(x-0)(x-2) + 1(x-0)(x-2)(x-4) \\ &= x^3 - 6x^2 + 11x - 6 \end{aligned}$$

So, the Newton polynomial that passes through the given data points is:

$$P_3(x) = x^3 - 6x^2 + 11x - 6$$

2.4.1 The error of the interpolating polynomial

Theorem 2.2

We suppose f a function, where $f \in \mathcal{C}^{n+1}[a,b]$ and $[a,b]$ is an interval containing $(n+1)$-distinct points. For any $x \in [a,b]$, there exist a real $\xi \in [a,b]$ for which

$$E = f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi) \Pi_{i=0}^{n}(x - x_i)$$

where $|f(x) - p_n(x)| \leq \frac{\theta_{n+1}}{(n+1)!} |\Pi_{i=0}^{n}(x - x_i)|$ *and* $\theta_{n+1} = \max_{x \in [a,b]} |f^{(n+1)}(x)|$.

♡

2.5 Conclusion

Polynomial interpolation, Lagrange, and Newton methods are both useful techniques for approximating functions using polynomial curves.

Polynomial interpolation is the process of finding a polynomial curve that passes through a given set of data points. This allows us to estimate the value of the function at points between the given data points.

Lagrange interpolation is a specific method of polynomial interpolation that uses the Lagrange basis polynomials. These polynomials are constructed in such a way that each one is zero at all data points except for one, where it has a value of one. The final polynomial curve is then formed by adding these Lagrange basis polynomials, each multiplied by the corresponding function value at the data point.

Newton interpolation is another method of polynomial interpolation that uses divided differences. Divided differences are a scheme for efficiently calculating the coefficients of the polynomial curve. The final polynomial curve is formed by summing these divided differences, each multiplied by an appropriate product of terms involving x values and their differences.

Both methods have their advantages and disadvantages. The Lagrange method is generally easier to understand and implement since it involves constructing the Lagrange basis polynomials directly. However, it can be computationally expensive when dealing with large datasets. On the other hand, the Newton method can be more efficient computationally due to the use of divided differences, but it may require additional calculations to update the polynomial when new data points are added.

In conclusion, both Lagrange and Newton methods are valuable tools for polynomial interpolation. The choice between them depends on the specific circumstances and requirements of the problem at hand.

2.6 Annex

2.6.1 MATLAB program

2.6.1.1 Interpolation

```
    function interpollution()
clear all;
clc;
while 1
n=2;
temp = inputdlg('function : ');
fun = temp1;
temp = inputdlg('value of x0 : ');
x0 = str2double(temp1);
temp = inputdlg('value of x1 :');
x1= str2double(temp1);
temp = inputdlg('value of x : ');
x = str2double(temp1);
ans = log(x0)+((log(x1)-log(x0))/(x1 - x0))*(x - x0);
disp('ANS : ');
disp(ans);
end
end


    function r = f(fun,x)
r = 25;
end
```

2.7 Exercises

1. Determine a polynomial of degree ≤ 3 that interpolates the data

x	2.4	3	3.6	4
y	0.7	8.5	11	31.1

2. Find a polynomial of order 2 that interpolates the table

x	0.4	0.6	0.8
y	-0.75	-0.82	-0.65

3. Use the table

x	1	1.2	1.4	1.6	1.8	2
f(x)	1	0.918168	0.887263	0.893515	0.931338	1

together with linear interpolation to construct an approximation to $f(1.51)$.

4. Use the table

x	1	1.2	1.4	1.6	1.8	2
f(x)	1	0.9181	0.8872	0.8935	0.9313	1

together with linear interpolation to construct an approximation to $f(1.44)$.

5. You are given the polynomial $p(x) = (x - 350)^4$. Suppose we want to interpolate this polynomial by another polynomial at n points equally distributed on the interval $[350, 351]$.

   (a) Under what condition on n do we expect the polynomial interpolant to coincide with the given polynomial p?

   (b) Expand p and find its five coefficients. Now, interpolate this polynomial at five and then at ten uniformly spaced points in $[350, 351]$ using the Vandermonde approach, and find the coefficient of the interpolating polynomial. What do you observe? What do you conclude from this observation?

6. The following tabulated function is a polynomial. Find the degree of the polynomial and the coefficient of the highest power of x.

7. Determine the polynomial of degree ≤ 3 that interpolates the function $f(x) = 2x^2 - x + 2$ at $x = 0, 1, 2, 3$. What is the degree of the polynomial?

8. Show that Newton's interpolating polynomial of degree one passing through the points (x_0, y_0) and (x_1, y_1) may be written in the form

$$p_1(x) = \frac{1}{x_1 - x_0} \begin{vmatrix} y_0 & x_0 - x \\ y_1 & x_1 - x \end{vmatrix}$$

9. Determine the polynomial of degree ≤ 5, using Newton's divided-differences, that inter-

polates the table

x	1	2	3	4	5	6
f(x)	14.5	19.5	30.5	53.5	94.5	159.5

Use the resulting polynomial to estimate the value of $f(4.5)$. Compare with the exact value, $f(4.5) = 71.375$

10. Construct Newton's interpolating polynomial for the data given by the table

x	-3	-2	-1	0	1
f(x)	-41	-17	-5	1	3

11. To investigate the relationship between yield of potatoes, y, and level of fertilizer application, x, an experimenter divided a field into 5 plots of equal size and applied differing amounts of fertilizer to each. The data recorded for each plot are given by the table (in pounds).

x	1	2	3	4	5
f(x)	22	23	25	30	28

(a) Find an interpolating polynomial for this table.

(b) According to the interpolating polynomial, approximately how many pounds would you expect from a plot to which 2.5 pounds of fertilizer had been applied?

12. Estimate $\log(2)$ using linear interpolation. Interpolate between $\log(1)$ and $\log(6) = 1.791759$. The true value of $\log(2)$ is 0.6931472.

13. Let $f(x) = \ln(1 + x)$, $x_0 = 1$, and $x_1 = 1.1$. Use linear interpolation to calculate an approximation value for $f(1.04)$.

14. Construct a quadratic polynomial that interpolates $f(x) = \log_2 x$ at the nodes $x_0 = 1$, $x_1 = 2$, $x_2 = 4$.

15. Consider the problem of constructing the cubic polynomial $p_3(x)$ that interpolates as follows:

$p_3(a) = f(a), p_3(b) = f(b), p_3(c) = f(c)$, and $p_3'(c) = f'(c)$

(a) Let $p_2(x)$ be the quadratic that interpolates f in the usual way at the nodes $x = a$, $x = b$, and $x = c$. Find a cubic polynomial $q(x)$ such that

$$q_3(x) = p_2(x) + q(x)$$

also interpolates f in the usual way at the same nodes.

(b) Now find the particular choice of q that implies that

$$q_3'(c) = f'(c)$$

16. Construct a divided-difference table for the interpolation problem $f(1) = 2$, $f(2) = 4$, $f'(2) = 0$. Write down Newton's interpolating polynomial.

17. Given the polynomial $p(x) = x^9$, try to interpolate p at 21 equidistant points on the interval $[0, 2]$. Find that interpolant, and explain how you find it.

18. Use the Newton divided-difference polynomial to approximate $f(3)$ from the following table:

x	0	1	2	3	4	5
f(x)	1	14	15	5	6	10

19. Consider a table of natural logarithm values, for the interval $[\frac{1}{2}, 1]$. How many entries do we have to have for linear interpolation between entries to be accurate to within 10^{-3}?

20. Derive the Lagrange polynomial that interpolates the data in the following table.

x	0	2	4	6
f(x)	1	-1	3	4

21. Let $p(x)$ be the quadratic polynomial that interpolates $f(x) = x^3$ at $x = 1, 2$, and 3.
    (a) Write down the Lagrange and Newton formulas for p.
    (b) Bound the relative error $e(x) = \frac{|f(x)-p(x)|}{|f(x)|}$ on the interval $1 < x < 3$
    (c) Bound the relative error $e(x) = \frac{|f(x)-p(x)|}{|f(x)|}$ on the interval $1.9 < x < 2.1$. Why is your bound so much smaller or larger than the bound in (b)?
    (d) State the general formula that gives the error on an interval $[a, b]$, in the degree-n polynomial p, which interpolates a smooth function f at $n + 1$ points t, in $[a, b]$. Separate the error into a product of three factors and explain why two of them are inevitable.

22. Let $f(x) = 2x^2e^x + 1$. Construct a Lagrange polynomial of degree two or less using $x_0 = 0$, $x_1 = 0.5$, and $x_2 = 1$. Approximate $f(0.8)$.

Chapter 3 Least square approximation

3.1 Introduction

In approximation theory, two general types of problems arise in fitting tabular data. The first one consists of finding an approximating function that traverses through every point in the table. The other problem consists of finding the "best" function that can be utilized to represent the data but does not precisely pass through every data point. This category of problems is called curve fitting.

We consider here only the simplest case of the same experimental error for all the data points. Let the data to be fitted be given by (x_i, y_i), with $i = 1$ to n.

Chapter objectives

- *To minimize the residuals, which are the differences between the observed data points and the values predicted by the linear model. By minimizing the sum of the squared residuals, you aim to find the best-fitting line that closely matches the data.*
- *To estimate the parameters of the linear model that result in the minimum sum of squared residuals. These estimated parameters can provide insights into the relationship between the variables and can be used for further analysis.*
- *Least squares problems have well-defined mathematical solutions, making it computationally efficient to find the optimal parameters of the model.*

3.2 Quadratic spline

In many cases, linear piecewise polynomials are unsatisfactory when being used to interpolate the values of a function, which deviate considerably from a linear function. In such cases, piecewise polynomials of higher degree are more suitable to use to approximate the function. In this section, we shall discuss the simplest type of differentiable, piecewise polynomial functions, known as quadratic splines. As before, consider the subdivision

$$a = x_1 < x_2 < ... < x_n = b$$

where $x_1, ..., x_n$ are given. For piecewise linear interpolation, we choose two points $(x_i, f(x_i))$ and $(x_{i+1}, f(x_{i+1}))$ in the subinterval $[x_i, x_{i+1}]$ and draw a line through those two points to interpolate the data. This approach is easily extended to construct the quadratic splines. Instead of choosing two points, we choose three points in the subinterval $[x_i, x_{i+1}]$ and pass a second-degree polynomial through these points as shown in Figure 6.2. We shall show

that there is only one such polynomial.

To construct a quadratic spline $Q(x)$, we first define a quadratic function in each subinterval $[xi, xi + 1]$ by

$$q_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2$$

where ai, bi, and ci are constants to be determined.

3.3 Fitting a straight line

Let $(x_i, y_i),\ i = 0, 1, 2, ..., n$ be the n sets of observations and let the related relation by $y = \alpha x + \beta$. Now we have to select α and β so that the straight line is the best fit to the data. The residual at $x = x_i$ or the distance from the data point (x_i, y_i) and the fitting curve is

$$\begin{aligned} d_i &= y_i - y(x_i),\ i = 1, 2, .., n \\ &= y_i - (\alpha x_i + \beta) \end{aligned}$$

By the principle of least squares, it fit minimizes the sum of the squares of the d_i's, we define

$$\mu(\alpha, \beta) = \sum_{i=1}^{n} {d_i}^2 \tag{3.1}$$

$$= \sum_{i=1}^{n} (y_i - (\alpha x_i + \beta))^2 \tag{3.2}$$

μ is the minimum.

Here $\mu(\alpha, \beta)$ is considered to be a function of two variables. This method of choosing α and β is commonly used in engineering, economics, and many other sciences. We know from calculus that the minimum of (3.1) will occur when

$$\frac{\partial \mu(\alpha, \beta)}{\partial \alpha} = 0, \quad \frac{\partial \mu(\alpha, \beta)}{\partial \beta} = 0$$

Taking the partial derivatives, we have

$$\begin{aligned} \frac{\partial \mu(\alpha, \beta)}{\partial \alpha} &= \sum_{i=1}^{n} 2(-x_i)(y_i - (\alpha x_i + \beta)) = 0 \\ \frac{\partial \mu(\alpha, \beta)}{\partial \beta} &= \sum_{i=1}^{n} 2(-1)(y_i - (\alpha x_i + \beta)) = 0 \end{aligned}$$

Since x_i, y_i are known, these equations form a system of two linear equations in the two unknowns α and β, which is evident when rewritten in the form

$$\alpha \sum_{i=1}^{n} {x_i}^2 + \beta \sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i, \tag{3.3}$$

$$\alpha \sum_{i=1}^{n} x_i + \beta n = \sum_{i=1}^{n} y_i \tag{3.4}$$

The second term, βn in the second equation comes from the fact that $\sum_{i=1}^{n} \beta = \beta n$. The equations (3.3) are called normal equations.

The solution of this linear system is

$$\alpha = \frac{n \sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n \sum_{i=1}^{n} {x_i}^2 - (\sum_{i=1}^{n} x_i)^2}$$

$$\beta = \frac{\sum_{i=1}^{n} {x_i}^2 \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i y_i \sum_{i=1}^{n} x_i}{n \sum_{i=1}^{n} {x_i}^2 - (\sum_{i=1}^{n} x_i)^2}$$

Example3.1 By the method of least squares find the straight line to the data given below:

x	5	10	15	20	25
y	16	19	23	26	30

Let the straight line be $y = \alpha x + \beta$

The normal equations are

$$\alpha \sum_{i} x_i + 5\beta = \sum_{i} y_i \tag{3.5}$$

$$\alpha \sum_{i} {x_i}^2 + \beta \sum_{i} x_i = \sum_{i} x_i y_i \tag{3.6}$$

To calculate $\sum_i x_i, \sum_i {x_i}^2, \sum_i y_i, \sum_i x_i y_i$ we form below the table

x	y	x^2	xy
5	16	25	80
10	19	100	190
15	23	225	345
20	26	400	520
25	30	625	750
75	114	1375	1885

The normal equations are

$$75\alpha + 5\beta = 114 \tag{3.7}$$

$$1375\alpha + 75\beta = 1885 \tag{3.8}$$

Eliminate β, multiply (3.7) by 15

$$1125\alpha + 75\beta = 1710 \tag{3.9}$$

We put (3.8)-(3.9) gives, $250\alpha = 175$ or $\alpha = 0.7$, hence $\beta = 12.3$ As a consequence, the best fitting line is $y = 0.7x + 12.3$

Let $X = \frac{x - x_{mind}}{h} = \frac{x-15}{5}$, $Y = \frac{y - y_{mind}}{h} = \frac{y-23}{5}$.

Let the line in the new variable by $Y = AX + B$

x	y	X	X^2	Y	XY
5	16	-2	4	-1.4	2.8
10	19	-1	1	-0.8	0.8
15	23	0	0	0	0
20	26	1	1	0.6	0.6
25	30	2	4	1.4	2.8
		0	10	-0.2	7

The normal equations

$$A\sum X + 5B = \sum Y \tag{3.10}$$

$$A\sum X^2 + B\sum X = \sum XY \tag{3.11}$$

Therefore,

$$-5B = -0.2 \tag{3.12}$$

$$B = -0.04 \tag{3.13}$$

and

$$10A = 7 \tag{3.14}$$

$$A = 0.7 \tag{3.15}$$

The equations $Y = 0.7X - 0.04$ that's mean

$$\frac{y - 23}{5} = 0.7(\frac{x - 15}{5}) - 0.04 \tag{3.16}$$

$$y - 23 = 0.7x - 10.5 - 0.2 \tag{3.17}$$

$$y = 0.7x + 33.3 \tag{3.18}$$

Which is the same equation as seen before.

3.4 Mean-square approximation of a function with orthogonal systems

An approximation of a function $f(t)$ by a function $\phi(t)$, where the error measure $\mu(f;\phi)$ is defined by the formula

$$\mu_\sigma(f;\phi) = \int_a^b [f(t) - \phi(t)]^2 d\sigma(t),$$

where $\sigma(t)$ is a non-decreasing function on $[a,b]$ different from a constant.

Let

$$u_1(t), u_2(t) \dots$$

be an [[Orthonormal system|orthonormal system]] of functions on $[a,b]$ relative to the distribution $d\sigma(t)$. In the case of a mean-square approximation of the function $f(t)$ by linear combinations $\sum_{k=} 1^n \lambda_k u_k(t)$, the minimal error for every $n = 1, 2 \dots$ is given by the sums

$$\sum_{k=} 1^n c_k(f) u_k(t),$$

where $c_k(f)$ are the [[Fourier coefficients|Fourier coefficients]] of the function $f(t)$ with respect to the system (*); hence, the best method of approximation is linear.

3.5 Least squares approximation with polynomial-orthogonal polynomials

The least-squares data fitting method is not just applicable to linear functions, such as $f(x) = \alpha x + \beta$. In fact, in many cases data from experimental results are not linear, so we need to consider some other guess functions.

Consider the general fitting function

$$y(x) = \sum_{k=0}^{m} a_k f_k(x)$$

Suppose that for the data in the table below, the guess function $f_k(x)$ is polynomial, we can whrite $f_k(x) = x^k$. Typically, the number of functions f_k is less than the number of data points that is $m < n$. Then, we have to determine the coefficients a_k for $k = 0, ..., m$ in accordance with the least-squares principle.

x	x_1	x_2	...	x_n
y	y_1	y_2	...	y_n

$a_0, a_1, a_2, ..., a_n$ minimize

$$\mu(a_0, a_1, ..., a_m) = \sum_{i=1}^{n}[y(x_i) - y_i]^2 \tag{3.19}$$

$$= \sum_{i=1}^{n}\left[\sum_{k=0}^{m}(a_k {x_i}^k) - y_i\right]^2 \tag{3.20}$$

We know, μ is minimum if

$$\frac{\partial}{\partial a_i}\mu(a_0, ..., a_m) = 0, \quad i = 0, ..., m. \tag{3.21}$$

Then

$$\begin{aligned}
\frac{\partial \mu}{\partial a_0} &= \sum_{i=1}^{n} 2\left[\sum_{k=0}^{m}(a_k {x_i}^k) - y_i\right] = 0 \\
\frac{\partial \mu}{\partial a_1} &= \sum_{i=1}^{n} 2\left[\sum_{k=0}^{m}(a_k {x_i}^k) - y_i\right](x_i) = 0 \\
\vdots &= \vdots \\
\frac{\partial \mu}{\partial a_m} &= \sum_{i=1}^{n} 2\left[\sum_{k=0}^{m}(a_k {x_i}^k) - y_i\right]({x_i}^m) = 0
\end{aligned}$$

This gives the $(m+1)$ normal equations for the $(m+1)$ unknowns $a_0, a_1, ..., a_m$

$$\begin{aligned}
a_0 n + a_1 \sum x_i + ... + a_m \sum {x_i}^m &= \sum y_i \\
a_0 \sum x_i + a_1 \sum x_i^2 + ... + a_m \sum {x_i}^{m+1} &= \sum y_i x_i \\
a_0 \sum x_i^2 + a_1 \sum x_i^3 + ... + a_m \sum {x_i}^{m+2} &= \sum y_i x_i^2 \\
\vdots &= \vdots \\
a_0 \sum x_i^m + a_1 \sum x_i^{m+1} + ... + a_m \sum {x_i}^{2m} &= \sum y_i x_i^m
\end{aligned}$$

3.6 Orthogonal Trigonometric Polynomials

The set of $\phi_0, ..., \phi_{2n-1}$ is orthogonal on $[-\pi, \pi]$ the weight function $w(x) = 1$.

$$\begin{aligned}
\phi_0(x) &= 1/2, \\
\phi_k(x) &= \cos(kx), \quad for \;\; k = 1, 2, ..., n, \\
\phi_{n+k}(x) &= \sin(kx), \quad for \;\; k = 1, 2, ..., n-1.
\end{aligned}$$

3.7 Conclusion

The least square approximation method is a useful technique in various fields, including mathematics, statistics, and data analysis. It aims to find the best fit line or curve that minimizes the sum of the squared differences between the observed data points and the estimated values.

In conclusion, the least square approximation offers several benefits:

- It is relatively easy to calculate and understand, making it widely applicable.
- It provides a well-defined and unique solution, making it reliable for practical applications.
- It is robust to outliers in the data, as it focuses on minimizing overall differences instead of individual points.
- It can handle both linear and nonlinear regression problems, allowing for flexible modeling.

However, it is important to note some limitations:

- Least square approximation assumes that there is a linear relationship between the variables being modeled, which may not always be the case.
- It assumes that the errors in the data follow a normal distribution with constant variance, which may not hold in real-world scenarios.
- It can be sensitive to influential points, which can significantly impact the estimated regression parameters.

In summary, while the least square approximation method is a powerful tool for estimating relationships between variables, it is crucial to consider its assumptions and potential limitations when applying it in practice.

3.8 Annex

3.8.1 MATLAB program

3.8.1.1 Quadratic fitting

```
    t= [0:8];
y= [40.12 66.78 80.17 86.71 80.77 66.78 44.41 10.51 -32.60];

    pc = polyfit (t, y,2);
plot (t, y,'k+');
hold on;
plot (t, polyval (pc, t),'r-');

    txt = sprintf ('Best fit line y=
legend ('Data points',txt);

    root = roots(pc)

    height = polyval(pc,2.5)

    d = polyder(pc)
time = roots(d)
max height = polyval(pc,time)
```

3.9 Exercises

1. Find the linear function that best fits the data

x	1.2	2.2	3.2	4.2	5.2
y	2.7	5.8	9.1	10.1	11

by using the method of least-squares and compute the error.

2. The following table lists the temperatures of a room recorded during the time interval $[1, 7]$. Find the best linear least squares that approximate the table.

1	2	3	4	5	6	7	8
13	15	10	15	20	14	13	10

Use your result to predict the temperature of the room at $8:00$.

3. Find the best least-squares line through the points $(-8,-9),(-3,-4),(-1,-2)$, and $(12,11)$.

4. Find the best least-squares line through the points $(0,2),(0,8),(1,-1)$, and $(3,11)$.

5. Given the data,

x	1	1.2	1.5	1.8	2.1	2.4	2.8
y	5.5	7	10.3	16	80.5	100	102

(a) Find the least-squares polynomial of degree two that best fits the data and compute the error

(b) Find the least-squares polynomial of degree three that best fits the data and compute the error.

6. Find the polynomial of degree 3 that fits the function $f(x) = \sin(\pi x)$ at the points $x_1 = -1$, $x_2 = -0.5$, $x_3 = 0$, $x_4 = 0.5$, and $x_5 = 1$ in the least-squares sense.

7. Given a table of data points (x_i, y_i) for $1 \leq i \leq n$, find the normal equations for the following guess functions

(a) $f(x) = a + b \ln x$,

(b) $f(x) = \frac{1}{a+x}$

(c) $f(x) = ax^b$

8. Determine the least-squares approximation of the type $f(x) = ax^2 + bx + c$ to the function 2^x at the points $x_i = 0, 1, 2, 3, 4$.

9. Find the least-squares exponential that best fits the following data

x	0.0	1.5	2.5	3.5	4.5
y	2.0	3.6	5.4	8.1	12.0

10. Given a table of data points (x_i, y_i) for $1 \leq i \leq n$, find the normal equations for the following guess functions

(a) $f(x) = a + b \ln x$

(b) $f(x) = \frac{1}{a+x}$

(c) $f(x) = ax^b$

11. Find the normal equations for fitting a curve of the form

$$y = a \sin bx$$

12. Determine the least-squares approximation of the type $f(x) = (a - c)e^{-bx} + c$, for the data in the following table:

x	0	1	2	5	10	15	30	45	60
f(x)	210	198	187	155	121	103	77	71	70

13. Determine the least-squares approximation of the type $f(x) = ax^2 + bx + c$ to the function 2^x at the points $x_i = 0, 1, 2, 3, 4$.

14. Suppose the quadratic function

$$f(x) = x^2 + 3x + 7$$

is used to generate the data,

x	0	1	2	3
f(x)	7	11	17	25

Find the least-squares polynomials of degree two and three that best fit these data. Compute these polynomials at the four values of x and compare your results with the exact ones.

15. Given the following data, fit a second degree polynomial to the data using the least-squares criterion:

x	1.0	2.3	3.8	4.9	6.1	7.2
f(x)	0.1	1.5	7.4	15.2	26.2	27.9

Chapter 4 Numerical integration

In this chapter, we discuss the issue of estimating the definite integral

$$I = \int_a^b f(x)dx \tag{4.1}$$

with $[a, b]$ finite.

In order to approximate I, the idea is one must first replace $f(x)$ with an interpolating polynomial $p(x)$, and then integrate the polynomial by calculating the area of the region bounded by $p(x)$, the lines $x = a$, $x = b$, and the x-axis. Numerical quadrature is the name of this procedure. Other process will be developed in the next sections.

Chapter objectives

- *To approximate the value of a definite integral, which represents the accumulated area under a curve. This is useful in various scientific, engineering, and mathematical applications where exact analytical solutions are hard to come by.*
- *To implement and understand, making it a good starting point for learning about numerical integration by the Trapezoidal rule.*
- *Provide a more accurate approximation of the integral compared to simpler methods like the Trapezoidal rule and simpson's rule, especially for functions that exhibit curvature.*

4.1 Trapezoidal rule

The trapezoidal rule is one of the simplest ways for determining the area under a curve. We suppose that the function $f(x)$ is known at the $(n + 1)$ points designated as $x_0, x_1, ..., x_n$, with the endpoints given by $x_0 = a$ and $x_n = b$. Define

$$f_i = f(x_i),\ h_i = x_{i+1} - x_i$$

We begin by dividing the interval $[a, b]$ into n subintervals, each with size h, where $h = \frac{b-a}{n}$ and $x_i = a + ih$ for $i = 0, 1, ..., n$

The area of the i-th trapezoid is therefore

$$S_i = \frac{h}{2}[f(x_{i-1}) - f(x_i)] \tag{4.2}$$

From (4.2), the total area S_n is determined by

$$\begin{aligned} S_n &= S_1 + S_2 + ... + S_{n-1} + S_n \\ &= \frac{h}{2}[f(x_0) + f(x_1)] + \frac{h}{2}[f(x_1) + f(x_2)] + ... + \frac{h}{2}[f(x_{n-2}) + f(x_{n-1})] + \frac{h}{2}[f(x_{n-1}) + f(x_n)]. \end{aligned}$$

Simplify the terms, we obtain the formula

$$S_n = \frac{h}{2}[f(x_0) + f(x_n)] + h\sum_{i=1}^{n-1} f(x_i) \tag{4.3}$$

Equation (4.3) is called the composite trapezoidal rule for n subintervals.

Example4.1

Use the composite trapezoidal rule with $n = 2$ to approximate

$$\int_0^3 x^2 e^x dx$$

we use the formula (4.3) with $n = 2$, we obtain

$$\begin{aligned} \int_0^3 x^2 e^x dx &\approx \frac{3}{4}[f(0) + f(3)] + \frac{3}{2}f(1.5) && (4.4) \\ &\approx 150.70307 && (4.5) \end{aligned}$$

4.2 Simpson's rule

Here, we take into account the combined Simpson's rule for points with equal spacing. For intervals of $2h$, we apply Simpson's rule, starting at a and finishing at b:

$$\int_a^b f(x)dx = \frac{h}{3}(f_0 + 4f_1 + f_2) + \frac{h}{3}(f_2 + 4f_3 + f_4) + ... + \frac{h}{3}(f_{n-2} + 4f_{n-1} + f_n). \tag{4.6}$$

Combining terms, we have

$$\int_a^b f(x)dx = \frac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + ... + 4f_{n-1} + f_n). \tag{4.7}$$

The first and last terms have a multiple of one; the even indexed terms have a multiple of 2; the odd indexed terms have a multiple of 4; and the entire sum is multiplied by $h/3$.

Example4.2 Use Simpson's rule to find an approximation to

$$\int_0^3 x^2 e^x dx$$

Using (4.3), we have

$$\begin{aligned} \int_0^3 x^2 e^x dx &\approx \frac{1}{2}[f(0) + 4f(1.5) + f(3)] \\ &\approx 110.55252 \end{aligned}$$

4.3 Numerical quadrature formulas

For a function of one independent variable, the basic idea of a quadrature rule is to replace the definite integral by a sum of the integrand evaluated at certain points (called quadrature points) multiplied by a number (called quadrature weights).

All the quadrature formulas developed so far in the preceding sections were of the form

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i) \tag{4.8}$$

where w_i are the weights to be given to n functional values $f(x_i)$ and $x_1, x_2, ..., x_n$ are the nodes selected to be equally spaced. There are two groups of integration methods that are commonly used: the Newton-Cotes formulas that employ equally spaced nodes, and the Gaussian quadrature formulas that employ unequally spaced nodes. The Gaussian quadrature formula to be developed in this section has the same form as (4.8), but the nodes to be used are not equally spaced. The reason for choosing unequally spaced nodes is to minimize the error obtained in performing the approximation (4.8) by properly choosing the points x_i.

In the composite trapezoidal rule, the approximation of the area under the curve between $x = a$ and $x = b$ is obtained by choosing points x_1 and x_2 at the ends of the interval $[a, b]$. Using the Gaussian quadrature, on the other hand, we choose x_1 and x_2 inside the interval (a, b) so that the area in the trapezoid equals the area under the curve. This is the basic idea of Gaussian rules.

4.4 Conclusion

Numerical integration methods, such as Simpson's rule and the trapezoidal rule, are powerful tools for approximating the definite integral of a function. These methods are particularly useful when the function cannot be integrated analytically or when a large number of data points need to be processed.

Both Simpson's rule and the trapezoidal rule divide the interval of integration into smaller segments and approximate the area under the curve by summing the areas of these segments. However, there are some differences between the two methods.

Simpson's rule is more accurate than the trapezoidal rule as it utilizes quadratic polynomials for approximation. It divides the interval into an even number of segments and fits a parabola to each pair of adjacent segments. This method provides a more accurate estimation of the integral, especially for functions with higher-order variations.

On the other hand, the trapezoidal rule approximates the area under the curve by connecting the data points with straight lines. It divides the interval into a series of trapezoids and sums their areas to estimate the integral. While the trapezoidal rule is simpler to implement, it generally yields less accurate results compared to Simpson's rule for the same number of segments.

In conclusion, both Simpson's rule and the trapezoidal rule are valuable numerical integration methods. Simpson's rule is preferred when higher accuracy is required, especially for functions with complex variations. On the other hand, the trapezoidal rule is simpler to use and can still provide reasonably accurate results, particularly for functions with relatively smooth variations. The choice between the two methods depends on the specific requirements of the problem at hand.

4.5 Annex

4.5.1 MATLAB program

4.5.1.1 Tripizoidal method

```
clear all;

fx = [400 -900 675 -200 25 0.2];

a = 0;
b = 0.8;
h = b - a;
trap = h * (polyval(fx, a) + polyval(fx, b))/2;
disp(trap);
```

4.6 Exercises

1. Use the composite trapezoidal rule with $n = 1$ to compute the integral

$$\int_1^3 (2x + 1)dx$$

2. Approximate the integral

$$I = \int_1^5 (x^4 - 3x^3 + 2x^2 - 3)dx$$

using the composite trapezoidal rule with $n = 30$. Compare with the exact value $\frac{175}{12}$

3. Eliminate the singularity at $x = 1$ of the integral

$$I = \int_0^1 \frac{2}{\sqrt{1 - x^2}} dx$$

by using the transformation $x = \sin u$ and then use the composite trapezoidal rule to approximate I

4. Use the composite trapezoidal rule to approximate the integral

$$I = \int_{0.4}^{0.6} e^{2x} \sin 3x dx$$

and find a bound for the error if possible.

5. Given the table

x	0	0.1	0.2	0.3
f(x)	2.72	3.00	3.32	3.67

use the composite trapezoidal rule to find an approximation to the integral

$$\int_0^{0.3} f(x)dx$$

6. Find an expression for the error of the following integral rule

$$\int_a^{a+h} f(x)dx \approx hf(a + h) - \frac{1}{2}h^2 f'(a)$$

7. Using the smallest step size possible, approximate

$$\int_2^6 \left(\frac{1}{1 + x}\right)$$

using the composite trapezoidal rule using only the values of $f(x)$ at $x = 2, 3, ..., 6$.

8. Given

x	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
f(x)	440	0	-162	-160	-84	0	50	48	0	-64	-90	0

(a) Approximate $\int_0^6 f(x)dx$ using the composite trapezoidal rule with $h = 2$.

(b) Approximate $\int_0^6 f(x)dx$ using the composite trapezoidal rule with $h = 6$.

(c) Find the Lagrange polynomial through $f(-5)$, $f(-2)$, $f(3)$, and $f(4)$.

(d) Find the linear least squares through $f(-5)$, $f(-2)$, $f(3)$, and $f(4)$.

9. Let $f(x) = e^{-}x^2$ and consider the integral

$$I = \int_0^1 f(x)dx$$

(a) Use the composite trapezoidal rule with $h = 0.25$ to approximate I.

(b) Calculate the bound on the absolute error for the Trapezoidal rule.

10. Find an upper bound for the error incurred in estimating

$$I = \int_0^\pi x \sin x dx$$

with the composite trapezoidal rule with $n = 10$.

11. Let $f(x) = \sqrt{1 + x^3}$

(a) How large must n be in the trapezoidal rule approximation of $\int_0^1 f(x)dx$ to insure that the absolute error is less than 10^{-3}?

(b) Estimate the integral using the trapezoidal rule approximation with the value of n obtained in part (a).

12. Use Simpson's composite rule to approximate the integral

$$I = \int_0^3 x^2 cos(x^2 - 1)dx$$

with an accuracy of 10^{-3}

13. Estimate the error in the approximation of the integral

$$\int_2^3 \frac{2}{x}dx$$

by Simpson's composite rule with $n = 6$.

14. Let f be defined by

$$f(x) = \begin{cases} x^2 - x + 1, & 0 \leq x \leq 1, \\ 2x - 1, & 1 \leq x \leq 2 \end{cases} \tag{4.9}$$

(a) Determine whether f is continuous on $[0, 2]$.

(b) Approximate the integral

$$I = \int_0^2 f(x)dx \; with \; n = 2$$

(i) using the composite trapezoidal rule on the interval $[0, 2]$,

(ii) using Simpson's composite rule first over $[0, 1]$ and then the composite trapezoidal rule over $[1, 2]$,

(iii) using Simpson's composite rule over $[0, 2]$.

Chapter 5 Numerical methods for ordinary differential equations

Note *[Euler]*

Leonhard Euler (15 April 1707 - 18 September 1783), was a Swiss mathematician, physicist, astronomer, geographer, logician, and engineer who founded the studies of graph theory and topology and made pioneering and influential discoveries in many other branches of mathematics such as analytic number theory, complex analysis, and infinitesimal calculus. the Euler method (also called the forward Euler method) is a first-order numerical procedure for solving ordinary differential equations (ODEs) with a given initial value. It is the most basic explicit method for numerical integration of ordinary differential equations and is the simplest Runge-Kutta method. The Euler method is named after Leonhard Euler, who first proposed it in his book Institutionum calculi integralis (published 1768-1770).

(Source : Wikipédia)

5.1 Introduction

The study and resolution of ordinary differential equations (ODEs) constitute a fundamental field of mathematics and numerous applied sciences. ODEs are used to describe changes and evolutions of quantities that depend on an independent variable, often time. They take the form of equations involving derivatives of one or more unknown functions with respect to the independent variable.

ODEs are encountered in various domains, such as physics, engineering, biology, economics, and more. They allow us to model dynamic phenomena, such as population growth, capacitor discharge, particle motion under a force, or the changing concentration of a chemical substance in a reaction.

The resolution of ODEs involves finding one or more functions that satisfy the given differential equation while also potentially adhering to initial conditions or boundary conditions. Depending on the complexity of the ODE, resolution can be achieved through different methods: analytical methods (finding an exact solution), numerical methods (approximating the solution using computational algorithms), or a combination of both.

Chapter objectives

- ❏ *To obtain a numerical approximation of the solution by discretizing the differential equation and employing algorithms such as Euler's method, Runge-Kutta methods, or finite difference methods.*
- ❏ *objectif of Euler's method is to achieve a first-order accurate approximation of the solution. This means that the error between the true solution and the computed solution using Euler's method is proportional to the step size employed in the method.*
- ❏ *Euler's method is particularly useful for solving initial value problems where the initial conditions are known, and the goal is to find the solution at subsequent time points.*
- ❏ *Runge-Kutta methods offer the possibility of using higher-order approximations, which results in more accurate solutions compared to simpler methods like Euler's method.*

In this chapter we shall be concerned with numerical methods for solving problems of the form

$$\frac{du}{dt} = f(t,u), \quad a \leq t \leq b \tag{5.1}$$

$$u(a) = u_0 \tag{5.2}$$

Such problems are called initial-value problems (IVPs). This problems can be resolved using a

variety of effective techniques and methods, we shall present here two methods.

5.2 Numerical methods: initial value problem

We begin with the simple Euler method, then discuss the more sophisticated Runge-Kutta methods.

5.3 Euler's mathod

Euler's method is the most basic numerical approach for ordinary differential equations. The basic idea behind Euler's method is to approximate the solution of a differential equation by breaking it down into small steps and using a linear approximation for each step. This method is particularly useful when it's difficult or impossible to find a closed-form solution for the differential equation.

Suppose we want to approximate the solution of the initial-value problem

$$\frac{du}{dt} = f(t,u), \quad u(a) = u_0 \tag{5.3}$$

on the interval $[a,b]$.

To begin the presentation of Euler's method, let us divide the interval $[a,b]$ into N equal subintervals and define the mesh points

$$t_i = a + ih, \;\; i = 0,1,...,N$$

With $h = \frac{b-a}{N}$. h is the step size. Starting with the initial condition u_0, a numerical approch to solve the problem will produce u_i for $i = 0,1,...,N$, to approximate the solution $u(t)$ at t_i for $i = 0,1,...,N$.

To derive Euler's method, we consider the Taylor expansion series of $u_{i+1} = u(t_{i+1})$ we get

$$\begin{aligned} u(t_{i+1}) &= u(t_i + h) \\ &= u(t_i) + hy'(t_i) + O(ht^2) \\ u(t_i + h) &\approx u(t_i) + hf(t_i, u_i) \end{aligned}$$

Then, we can write

$$u_{i+1} = u_i + hf(t_i, u_i), \;\; i = 0,1,...,N-1. \tag{5.4}$$

Equation (5.4)

We say that the Euler method steps forward in time using a time-step h, starting from the initial value $u_0 = u(0)$. The local error of the Euler Method is $O(ht^2)$. It is therefore customary to call the Euler Method a first-order method.

Example5.1 Solve the initial-value problem

$$\frac{dy}{dt} = 2t - y, \;\; y(0) = -1$$

With $n = 10$ to get the value of y at $t = 1$. Compare with the values of the exact solution $y(t) = e^{-t} + 2t - 2$

Let $h = 1/10 = 0.1$ and $f(x, y) = 2t - y$, we have

$$\begin{aligned} y(0.1) \approx y_1 &= y_0 + 0.1f(0, -1) \\ &= -1 + 0.1[2(0) - (-1)] \\ &= -0.9 \end{aligned}$$

At $t = 1.0$ the Euler value is $y_{10} = 0.348678$ and the error is 0.0192.

5.4 Higher-order Taylor series methods

Using two terms from the series, Euler's method was derived from the Taylor Series. That's why we derive an approximate solution $u(t)$ to our initial-value problem using four terms in the Taylor series, we obtain

$$u(t_{i+1}) = u(t_i) + hu'(t_i) + ... + \frac{h^3}{3!}u^{(3)}(t_i) + \frac{h^4}{4!}u^{t_i}(O_i). \tag{5.5}$$

with $t_i < O_i < t_{i+1}$

Neglating the rest, we get the difference equation

$$u_{i+1} = u_i + hf(t_i, u_i) + \frac{h^2}{2}f'(t_i, u_i) + \frac{h^3}{6}f''(t_i, u_i) \tag{5.6}$$

for $i = 0, 1, ..., N - 1$

we use the chain rule, we obtain

$$u' = f(t, u) \tag{5.7}$$

$$u'' = f' = f_t + f_u u' \tag{5.8}$$

$$u''' = f'' = f_{tt} + 2f_{tu}f + f_{uu}f^2 + f_t f_u + f_u^2 f \tag{5.9}$$

Substitute the three equations (5.7) in the equation (5.6) we obtain what we called third-order Taylor's method formula.

the practical difficulty of this method is that the various derivatives of u are complicated and, in some cases, impossible to find. Even though we have reduced the truncation error by adding

more terms in the series, the method is generally impractical.

Example5.2 Use the third-order Taylor's method for the initial-value problem

$$\frac{dy}{dt} = 2t - y, \;\; y(0) = -1$$

with $n = 10$ to approximate $y(1)$

$$\begin{aligned} f'(t,y) &= 2 - y' = 2(1-t) + y \\ f''(t,y) &= -2 + y' = 2(t-1) - y \\ y_{i+1} &= y_i + h\phi_3(t_i, y_i) \end{aligned}$$

Such as

$$\begin{aligned} \phi_3(t_i, y_i) &= 2t_i - y_i + \frac{h}{2}[2(1-t_i) + y_i] + \frac{h^2}{6}[2(t_i - 1) - y_i] \\ &= 2t_i - y_i + (\frac{h}{2} - \frac{h^2}{6})[2(1-t_i) + y_i] \end{aligned}$$

For $i = 0, 1, ..., n-1$.

Since $n = 10, \;\; h = \frac{1-0}{10} = 0.1$ then we get

$y_{i+1} = 0.9048333y_i + 0.1903333t_i + 0.0096667$

with $t_i = 0.1i$ for $i = 0, 1, ..., 9$.

5.5 Second-order Runge-Kutta methods

We now derive all second-order Runge-Kutta methods. Higher-order methods can be similarly derived, but require substantially more algebra.

Consider the first-order differentiel equation $\dot{u} = f(u,t)$ with $u(0) = u_0$. A general second-order Runge-Kutta method may be written in the form

$$k_1 = hf(t_i, u_i) \tag{5.10}$$

$$k_2 = hf(t_i + \alpha h, u_i + \beta k_1) \tag{5.11}$$

$$u_{i+1} = u_i + ak_1 + bk_2 \tag{5.12}$$

with α, β, a and b constants that define the particular second-order Runge-Kutta method. These constants are to be constrained by setting the local error of the second-order Runge-Kutta method to be $O(ht^2)$.

To derive these constants, we will make Eqn.(5.12) agree with the Taylor series expansion of $u(t)$ about t_i. We have

$$\begin{aligned} u(t_{i+1}) &= u(t_i) + hu'(t_i) + \frac{h^2}{2}u''(t_i) + ... \\ &= u(t_i) + hf(t_i, u(t_i)) + \frac{h^2}{2}f'(t_i, u(t_i)) + ... \end{aligned}$$

Since $f'(t_i, u(t_i)) = f_t + f_u f$, then

$$u(t_{i+1}) = u(t_i) + hf + \frac{h^2}{2}(f_t + f_u f) + O(h^3) \tag{5.13}$$

where all the functions in equation (5.13) are calculated at the point $(t_i, u(t_i))$. We now expand $f(t_i + \alpha h, u_i + \beta k_1)$ in the Taylor's series for a function of two variables

$$f(t_i + \alpha h, u_i + \beta k_1) = f(t_i, y_i) + \alpha h f_t + \beta k_1 f_u + O(h^2) \tag{5.14}$$

From equations (5.12) and (5.14) imply

$$u(t_{i+1}) = u(t_i) + h(a+b)f + bh^2(\alpha f_t + \beta f f_u) + O(h^3) \tag{5.15}$$

All functions in (5.14) and (5.15) are evaluated at $(t_i, u(t_i))$ and by comparing (5.13) and (5.15), we obtain

$$\begin{cases} a + b = 1 \\ \alpha = \beta = \frac{1}{2b}. \end{cases} \tag{5.16}$$

solution is $a = b = \frac{1}{2}$ and $\alpha = \beta = 1$

This leads to the Runge-Kutta method of order 2, sometimes known as the modified Euler method

$$u_{i+1} = u_i + \frac{h}{2}[f(t_i, u_i) + f(t_i + h, u_i + hf(t_i, u_i))], \quad i = 0, 1, ..., n-1. \tag{5.17}$$

or

$$u_{i+1} = u_i + \frac{h}{2}(k_1 + k_2) \tag{5.18}$$

where

$k_1 = f(t_i, u_i)$ and $k_2 = f(t_i + h, u_i + hk_1)$

Other choices for the parameters are $\alpha = \beta = 1/2$, $a = 0$, $b = 1$. This leads to a formula known as the midpoint method given by

$$u_{i+1} = u_i + hf\left(t_i + \frac{h}{2}, u_i + \frac{h}{2}f(t_i, u_i)\right), \quad i = 0, 1, ..., n-1. \tag{5.19}$$

Example5.3

Find $y(0.2)$ for $y' = \frac{x-y}{2}$, $x_0 = 0$,$y_0 = 1$, with step length 0.1 using Runge-Kutta 2 method

Given $y' = \frac{x-y}{2}$, $y(0) = 1$, $h = 0.1$, $y(0.2) =?$

$$\begin{aligned} k_1 &= hf(x_0, y_0) = (0.1)f(0, 1) = (0.1)(-0.5) = -0.05 \\ k_2 &= hf(x_0 + h, y_0 + k_1) = (0.1)f(0.1, 0.95) = (0.1)(-0.425) = -0.0425 \\ y_1 &= y_0 + \frac{k_1 + k_2}{2} = 1 - 0.04625 = 0.95375 \\ y(0.1) &= 0.95375 \end{aligned}$$

Again taking (x_1, y_1) in place of (x_0, y_0) and repeat the process

$$\begin{aligned} k_1 &= hf(x_1, y_1) = (0.1)f(0.1, 0.95375) = (0.1)(-0.42688) = -0.04269 \\ k_2 &= hf(x_1 + h, y_1 + k_1) = (0.1)f(0.2, 0.91106) = (0.1)(-0.35553) = -0.03555 \\ y_2 &= y_1 + \frac{k1 + k2}{2} = 0.95375 - 0.03912 = 0.91463 \\ y(0.2) &= 0.91463 \end{aligned}$$

5.6 Conclusion

Numerical methods for Ordinary Differential Equations (ODEs) play a crucial role in solving differential equations when an analytical solution is not feasible or readily available. Three commonly used numerical methods for ODEs are Euler's method, Runge-Kutta methods, and the Taylor series method. Let's analyze each method and draw a conclusion about their effectiveness.

- Euler's Method: Euler's method is a simple and straightforward numerical method. It approximates the solution of an ODE by using tangent lines. While it is easy to implement and computationally efficient, it is less accurate compared to more advanced methods. Euler's method suffers from a global truncation error that increases with larger step sizes, leading to noticeable errors in long-term predictions. Therefore, it is most suitable for simple problems or as an initial approximation for other methods.
- Runge-Kutta Methods: Runge-Kutta methods, specifically the fourth-order Runge-Kutta (RK4) method, are widely used and more accurate than Euler's method. RK4 employs weighted averages of slopes at different intermediate points to approximate the solution. It mitigates the truncation error by using higher-order terms, resulting in improved accuracy. Runge-Kutta methods are reliable and have become the go-to choice for many ODE problems. However, they can still be computationally demanding for very large or complex systems.
- Taylor Series Method: The Taylor series method constructs a local approximation for the solution by expanding it as a polynomial around a given point. This method provides accurate results when a high number of terms in the series are considered. However, the computational complexity and requirement of high-order derivatives make it less commonly used in practice.

In conclusion, each numerical method for ODEs has its advantages and limitations. Euler's method is simple and efficient but less accurate, while Runge-Kutta methods offer a good balance between accuracy and computational complexity. The Taylor series method can provide high accuracy but is computationally demanding. The choice of method depends on the specific problem at hand, balancing accuracy requirements with available computational resources.

5.7 Exercises

1. Given the initial-value problem

$$y' = -3y\sin(t),\ y(0) = \frac{1}{2}$$

use Euler's method with $N = 10$ to approximate the solution. Compare with the values of the exact solution $y = \frac{1}{2}e^{3(\cos t - 1)}$.

2. Use Euler's method with $h = 0.1$ to approximate the solution of the IVP

$$y' = y - \frac{y}{t},\ 1 \le t \le 2\ y(1) = \frac{1}{2}$$

Use the data points generated by Euler's method to find the best function that fits this data in the least squares sense. Use the resulting function to approximate the following values of y:

(a) $y(1.02)$

(b) $y(1.67)$

(c) $y(1.98)$

Compare with the values of the exact solution $y = \frac{e^{t-1}}{t-2}$.

3. Use one step of Taylor's method of order 2 with $h = 0.1$ to calculate an approximate value for $y(0.1)$ of the following initial-value problems:

(a) $y' = -2ty^2,\ \ 0 \le t \le 1,\ \ y(0) = 1,$

(b) $y' = 3(t-1)^2,\ \ 0 \le t \le 1,\ \ y(0) = 1,$

4. Consider the IVP

$$y' = t + y + ty,\ y(0) = 1.$$

Use Taylor series method with terms through t^3 to approximate $y(0.1)$ and $y(0.5)$

5. Use Taylor series method of order 4 to approximate the solution of the IVP

$$y' = 4\cos t - ty,\ y(0) = 1$$

over the interval $[0, 3]$ with $h = 0.2$.

6. Consider the IVP

$$y' = t + y,\ \ y(0) = 1$$

which has the analytic solution $y(t) = 2e^t - t - 1$. Use Taylor's series of order 4 to estimate $y(0.5)$ with $h = 0.1$. Compute the error.

7. Use the Taylor series method with terms through t^4 to approximate the solution of the IVP

$$y' = ty^{\frac{1}{3}},\ \ y(1) = 1$$

in the interval $[1, 5]$ with $h = 0.5$.

8. Solve the IVP

$$y' = \cos t - \sin y + t^2, \;\; y(-1) = 3$$

by using both the first and second-order Taylor series methods to estimate $y(-0.8)$.

9. Use the Runge-Kutta method of order 4 with $h = 0.1$ to approximate the solution of IVP

$$y' = y - \frac{y}{t}, \;\; 1 \leq t \leq 2, \;\; y(1) = \frac{1}{2}$$

Use the data points generated by the Runge-Kutta method to find the best function that fits this data in the least squares sense. Use the resulting function to approximate the following values of y:

(a) $y(1.02)$

(b) $y(1.67)$

(c) $y(1.98)$

Compare with the values of the exact solution $y = \frac{e^{t-1}}{2t}$.

10. Given the IVP

$$y' = e^{t^2} - \frac{y}{t}, \;\; 1 \leq t \leq 2, \;\; y(1) = \frac{e}{2}$$

use the Runge-Kutta method of order 2 with $N = 10$ to approximate the solution and compare with the values of the exact solution $y = \frac{1}{2t}e^{t^2}$.

11. Use the answers generated in Exercise below and the linear-least squares method to approximate y at

(a) $t = 1.25$

(b) $t = 1.65$

Compare your results with the actual values of y.

12. Use the Runge-Kutta method of order 2 to approximate the solution of the following IVPs in the interval indicated:

(a) $y' = -e^t y, \;\; [0, 1], \;\; y(0) = 3, \;\; N = 10$

(b) $y' = -4t^3 y, \;\; [1, 2], \;\; y(1) = 1, \;\; N = 10$

(c) $y' = -2y, \;\; [0, 1], \;\; y(0) = 4, \;\; N = 10$

(d) $y' = -\cos(t)y, \; [0, 1.2], \;\; y(0) = 2, \;\; N = 12$

13. Repeat the Exercise using the Runge-Kutta method of order 4.

14. When $f(t, y)$ depends only on t, i.e., $f(t, y) = f(t)$, show that the Runge Kutta method of order 4 reduces to Simpson's rule

$$\int_{t_n}^{t_{n+1}+h} f(t)dt \approx \frac{h}{6}\left[f(t_n) + 4f(t_n + \frac{h}{2}) + f(t_n + h)\right]$$

Chapter 6 System of linear equations-Direct methods

Note *[Cholesky]*

André-Louis Cholesky (15 October 1875, in Montguyon - 31 August 1918, in Bagneux), was a French military officer, geodesist, and mathematician. Cholesky was born in Montguyon, France. His paternal family was descendant from the Cholewski family who emigrated from Poland during the Great Emigration. He attended the Lycée in Bordeaux and entered the École Polytechnique, where Camille Jordan and Henri Becquerel taught. He worked in geodesy and cartography, and was involved in the surveying of Crete and North Africa before World War I. He is primarily remembered for the development of a form of matrix decomposition known as the Cholesky decomposition which he used in his surveying work.

(Source : Wikipédia)

6.1 Introduction

the system of linear equations is the set of two or more linear equations involving the same variables. Here, linear equations can be defined as the equations of the first order, i.e., the highest power of the variable is 1. Linear equations can have one variable, two variables, or three variables. Thus, we can write linear equations with n number of variables. In this chapter we will define the different methods of solving these systems of linear equations and solved examples.

Chapter objectives

- *Direct methods aim to provide the exact solution to a system of linear equations, without any approximation. This is particularly important when high precision is required in scientific and engineering applications.*
- *For systems with a unique solution, the Gauss method's objective is to transform the system into a triangular form, revealing the values of the variables through back-substitution. This confirms that the system has a consistent and unique solution.*
- *LU decomposition is a form of matrix factorization, which is a fundamental technique in linear algebra and numerical analysis. Matrix factorizations are crucial for understanding the structure and properties of matrices, as well as for various computational algorithms.*

Consider the system of n linear equations and n unknowns, given by

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n &= b_2 \\ \vdots\vdots\vdots &= \vdots \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n &= b_n \end{aligned} \tag{6.1}$$

for the unknowns $x_1, x_2, ..., x_n$, given the coefficients $a_{ij}, i, j = 1, 2, ..., n$ and the constants $b_i,\ i = 1, 2, ..., n$.

We can write this system as the matrix equation

$$Ax = b$$

with

$$A = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & ... & a_{nn} \end{pmatrix}, x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

6.2 Gaussian Elimination

Consider the system (7.1) in matrix form

$$Ax = b.$$

Let us denote the original system by $A^{(1)}x = b^{(1)}$. That is,

$$A = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & ... & a_{1n}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & ... & a_{2n}^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & ... & a_{nn}^{(1)} \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, b = \begin{bmatrix} {b_1}^{(1)} \\ {b_2}^{(1)} \\ \vdots \\ {b_n}^{(1)} \end{bmatrix} \tag{6.2}$$

The Gaussian elimination consists of reducing the system (6.2) to an equivalent system $A'x = d'$, in which A' is an upper triangular matrix. This new system can be easily solved by back substitution.

Algorithm:

Step 1: Assume $a_{11}^{(1)} \neq 0$. Define the row multipliers by

$$l_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}.$$

Multiply the first row by m_{i1} and subtract from the ith row $(i = 2, ..., n)$ to get

$$\begin{aligned} a_{ij}^{(2)} &= a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)}, \ j = 2, 3, ..., n \\ b_i^{(2)} &= b_i^{(1)} - m_{i1}b_1^{(1)} \end{aligned}$$

Here, the first rows of A and b are left unchanged, and the entries of the first column of A below $a_{11}^{(1)}$ are set to zeros.

The result of the transformed system is

$$A = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & ... & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & ... & a_{2n}^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & ... & a_{nn}^{(2)} \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, b = \begin{bmatrix} {b_1}^{(1)} \\ {b_2}^{(2)} \\ \vdots \\ {b_n}^{(2)} \end{bmatrix} \tag{6.3}$$

We continue in this way. At the kth step we have

Step k: Assume $a_{kk}^{(k)} \neq 0$. Define the row multipliers by

$$m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$$

Multiply the kth row by m_{ik} and subtract from the ith row $(i = k+1, ..., n)$ to get

$$\begin{aligned} a_{ij}^{(k+1)} &= a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}, \quad j = k+1, ..., n. \\ b_i^{(k+1)} &= b_i^{(k)} - m_{ik} b_k^{(k)} \end{aligned}$$

At this step, the entries of column k below the diagonal element are set to zeros, and the rows 1 through k are left undisturbed. The result of the transformed system is

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & ... & a_{1k}^{(1)} & a_{1,k+1}^{(1)} & ... & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & ... & a_{2k}^{(2)} & a_{2,k+1}^{(2)} & ... & a_{2n}^{(2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & ... & a_{kk}^{(k)} & a_{k,k+1}^{(k)} & ... & a_{kn}^{(k)} \\ 0 & 0 & ... & 0 & a_{k+1,k+1}^{(k+1)} & ... & a_{k+1,n}^{(k+1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & ... & 0 & a_{n,k+1}^{(k+1)} & ... & a_{nn}^{(k+1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \\ x_{k+1} \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_k^{(k)} \\ b_{k+1}^{(k+1)} \\ \vdots \\ b_n^{(k+1)} \end{bmatrix} \tag{6.4}$$

At $k = n-1$, we obtain the final triangular system

$$\begin{aligned} a_{11}^{(1)} x_1 + a_{12}^{(1)} x_2 + \ ... \ + a_{1n}^{(1)} x_n &= b_1^{(1)} \\ a_{22}^{(2)} x_2 + \ ... \ + a_{2n}^{(2)} x_n &= b_2^{(2)} \\ ... &= ... \\ a_{n-1,n-1}^{(n-1)} x_{n-1} + a_{n-1,n}^{(n-1)} x_n &= b_{n-1}^{(n-1)} \\ a_{nn}^{(n)} x_n &= b_n^{(n)} \end{aligned}$$

Using back substitution, we obtain the following solution of the system

$$\begin{aligned} x_n &= \frac{b_n^{(n)}}{a_n n^{(n)}} \\ x_{n-1} &= \frac{b_{n-1}^{(n-1)} - a_{n-1,n}^{(n-1)} x_n}{a_{n-1,n-1}^{(n-1)}} \\ x_i &= \frac{b_i^{(i)} - (a_{i,i+1}^{(i)} x_{i+1} + ... + a_{in}^{(i)} x_n)}{a_{ii}^{(i)}} \\ &= \frac{b_i^{(i)} - \sum_{j=i+1}^{n} a_{ij}^{(i)} x_j}{a_{ii}^{(i)}} \end{aligned}$$

Remark In the Gaussian elimination algorithm described above, we used the equations in their

natural order and we assumed at each step that the pivot element $a_{kk}^{(k)} \neq 0$ So the algorithm fails if the pivot element becomes zero during the elimination process. In order to avoid an accidental zero pivot, we use what is called Gaussian elimination with scaled partial pivoting.

Example6.1 Solve the following system of linear equations by Gaussian elimination method:

$$\begin{aligned} -3x_1 + 2x_2 - x_3 &= -1 \\ 6x_1 - 6x_2 + 7x_3 &= -7 \\ 3x_1 - 4x_2 + 4x_3 &= -6 \end{aligned}$$

To perform Gaussian elimination, we form an Augmented Matrix by combining the matrix A with the column vector b:

$$\begin{pmatrix} -3 & 2 & -1 & -1 \\ 6 & -6 & 7 & -7 \\ 3 & -4 & 4 & -6 \end{pmatrix}$$

Row reduction is then performed on this matrix. Allowed operations are

1. multiply any row by a constant,
2. add multiple of one row to another row
3. interchange the order of any rows. The goal is to convert the original matrix into an upper-triangular matrix.

We start with the first row of the matrix and work our way down as follows. First we multiply the first row by 2 and add it to the second row, and add the first row to the third row

$$\begin{pmatrix} -3 & 2 & -1 & -1 \\ 0 & -2 & 5 & -9 \\ 0 & -2 & 3 & -7 \end{pmatrix}$$

We then go to the second row.We multiply this row by -1 and add it to the third row

$$\begin{pmatrix} -3 & 2 & -1 & -1 \\ 0 & -2 & 5 & -9 \\ 0 & 0 & -2 & 2 \end{pmatrix}$$

The resulting equations can be determined from the matrix and are given by

$$\begin{aligned} -3x_1 + 2x_2 - x_3 &= -1 \\ -2x_2 + 5x_3 &= -9 \\ -2x_3 &= 2 \end{aligned}$$

These equations can be solved by backward substitution, starting from the last equation and working backwards. We have

$$\begin{aligned} -2x_3 &= 2 \\ -2x_2 = -9 - 5x_3 &= -4 \\ -3x_1 &= -1 - 2x_2 + x_3 = -6 \end{aligned}$$

implies that $x_3 = -1, x_2 = 2, x_1 = 2$. We can whrite

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ -1 \end{pmatrix}$$

6.3 LU decomposition

Consider the system of equations

$$Ax = b$$

The LU decomposition consists of transforming the coefficient matrix A into the product of two matrices, L and U, where L is a lower triangular matrix and U is an upper triangular matrix having 1's on its diagonal.
Once L and U are found, the solution of the system $Ax = b$ can be carried out by writing

$$LUx = b$$

and setting

$$Ux = y \tag{6.5}$$

So that

$$Ly = b \tag{6.6}$$

Equations (6.5) and (6.6) are two triangular systems which can be easily solved by first using the forward substitution in (6.6) to get y, and then with y known, we use the back substitution in (6.5) to get x. Two types of factorizations will now be presented, the first one uses Crout's and Cholesky's methods and the second one uses the Gaussian elimination method.

6.3.1 Crout's and Cholesky's methods

We shall illustrate the method of finding L and U in the case of a $4 * 4$ matrix:
We wish to find L, having nonzero diagonal entries, and U such that

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

Multiplying the rows of L by the first column of U, one gets

$$l_{i1} = a_{i1}, \;\; i = 1, 2, 3, 4.$$

Hence, the first column of L is given by the first column of A. Next, multiply the columns of U by the first row of L to get

$$l_{11}u_{1i} = a_{1i}, \;\; i = 2, 3, 4.$$

Thus,

$$u_{1i} = \frac{a_{1i}}{l_{11}}, \;\; i = 2, 3, 4.$$

which give the first row of U.

We continue in this way by getting alternatively a column of L and a row of U. The result is

$$\begin{aligned} l_{i2} &= a_{i2} - l_{i1}u_{12}, \;\; i = 2, 3, 4. \\ u_{2i} &= \frac{a_{2i} - l_{21}u_{1i}}{l_{22}}, \;\; i = 3, 4. \\ l_{i3} &= a_{i3} - l_{i1}u_{13} - l_{i2}u_{23}, \;\; i = 3, 4. \\ u_{34} &= \frac{a_{34} - l_{31}u_{14} - l_{32}u_{24}}{l_{33}}, \\ l_{44} &= a_{44} - l_{41}u_{14} - l_{42}u_{24} - l_{43}u_{34} \end{aligned}$$

In algorithmic form, the factorization may be presented as follows for an $n * n$ matrix:

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}, \;\; j \leq i, \; i = 1, 2, ..., n. \tag{6.7}$$

$$u_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}}{l_{ii}}, \;\; i \leq j, \; j = 2, 3, ..., n. \tag{6.8}$$

Note that this algorithm can be applied if the diagonal elements l_{ii}, for each $i = 1, ..., n$, of L, are nonzero.

The LU factorization that we have just described, requiring the diagonal elements of U to be one, is known as Crout's method. If instead the diagonal of L is required to be one, the factorization is called Doolittle's method.

6.3.2 Gaussian elimination method LU

We shall now illustrate a method of constructing L and U using Gaussian elimination. We process exactly as Gaussian elimination except that we keep a record of the elementary row operation performed at each step. We concentrate here on the factorization without pivoting. So, we assume that the naive Gaussian elimination can be successfully performed to solve the linear system $Ax = b$. Also, we are not concerned with the coefficient vector b. That is, we do not need to form an augmented matrix. Looking back at the elimination process, we see that the row multipliers for naive Gaussian elimination at k-th step are defined by

$$l_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$$

provided that $a_{kk}^{(k)} \neq 0$. It is easily verified that when the matrix

$$N_1 = \begin{bmatrix} 1 & & & & \\ -l_{21} & 1 & & 0 & \\ -l_{31} & 0 & 1 & & \\ \vdots & \vdots & & \ddots & \\ -l_{n1} & 0 & \cdots & 0 & 1 \end{bmatrix}$$

is multiplied by the matrix A on the left, the result is

$$N_1 A = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix}$$

which has the same effect of multiplying row 1 of A by the row multiplier l_{i1} and subtracting the result from row i of A for $i = 2, ..., n$. If $a_{kk}^{(k-1)} \neq 0$, the $(k-1)$ step is formed by

$$N_{k-1}N_{k-2}...N_1 A = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1k}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2k}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & 0 & \cdots & \vdots & & \vdots \\ 0 & 0 & \cdots & a_{kk}^{(k)} & & a_{kn}^{(k)} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & a_{nk}^{(k)} & & a_{nn}^{(k)} \end{bmatrix}$$

Where

$$N_{k-1} = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & 0 \\ & & -m_{k,k-1} & \ddots & \\ & & \vdots & & \\ & & -m_{n,k-1} & & 1 \end{bmatrix}$$

After $(n-1)$ steps, the elimination process without pivoting results in

$$N_{n-1}N_{n-2}...N_1A = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{nn}^{(n)} \end{bmatrix} = U \tag{6.9}$$

which is an upper triangular matrix. we multiply (6.9) on the left by N_{n-1}^{-1}, then N_{n-2}^{-1}, ... to have

$$A = N_1^{-1}N_2^{-1}\cdots N_{n-1}^{-1}U$$

It's easy to verify that

$$N_{k-1}^{-1} = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & 0 \\ & & m_{k,k-1} & \ddots & \\ & & \vdots & & \\ & & m_{n,k-1} & & 1 \end{bmatrix}$$

Hence,

$$N_1^{-1}N_2^{-1}\cdots N_{n-1}^{-1} = \begin{bmatrix} 1 & & & & \\ m_{21} & 1 & & 0 & \\ m_{31} & m_{32} & \ddots & & \\ \vdots & \vdots & \ddots & 1 & \\ m_{n1} & m_{n2} & \cdots & m_{n,n-1} & 1 \end{bmatrix} = L \tag{6.10}$$

which is a lower triangular matrix with all ones in its diagonal. Finally, we see that

$$LU = N_1^{-1}N_2^{-1}\cdots N_{n-1}^{-1}N_{n-1}N_{n-2}...N_1A = A$$

where U is given by (6.9) and L is given by (6.10).

Example6.2

Let the system of equations

$$\begin{aligned} -3x_1 + 2x_2 - x_3 &= -1 \\ 6x_1 - 6x_2 + 7x_3 &= -7 \\ 3x_1 - 4x_2 + 4x_3 &= -6 \end{aligned}$$

The process of Gaussian Elimination also results in the factoring of the matrix A to $A = LU$, where L is a lower triangular matrix and U is an upper triangular matrix. We have

$$N_1 A = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} -3 & 2 & -1 \\ 6 & -6 & 7 \\ 3 & -4 & 4 \end{pmatrix} = \begin{pmatrix} -3 & 2 & -1 \\ 0 & -2 & 5 \\ 0 & -2 & 3 \end{pmatrix}$$

Note that the matrix N_1 performs row elimination on the first column. Two times the first row is added to the second row and one times the first row is added to the third row. The entries of the column of N_1 come from $2 = -(6/-3)$ and $1 = -(3/-3)$ as required for row elimination. The number -3 is called the pivot.

Then

$$N_2(N_1 A) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} -3 & 2 & -1 \\ 0 & -2 & 5 \\ 0 & -2 & 3 \end{pmatrix} = \begin{pmatrix} -3 & 2 & -1 \\ 0 & -2 & 5 \\ 0 & 0 & -2 \end{pmatrix}$$

Here, N_2 multiplies the second row by $-1 = -(-2/-2)$ and adds it to the third row. The pivot is -2.

We now have

$$N_2 N_1 A = U$$

$$A = N_1^{-1} N_2^{-1} U$$

The inverse matrices are easy to find. The matrix N_1 multiples the first row by 2 and adds it to the second row, and multiplies the first row by 1 and adds it to the third row. To invert these operations, we need to multiply the first row by -2 and add it to the second row, and multiply the first row by -1 and add it to the third row. To check, with

$$N_1 N_1^{-1} = I$$

where

$$N_1^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

Similarly

$$N_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

consequently,

$$L = N_1^{-1} N_2^{-1}$$

then by multiplication of two matrix, we obtain

$$L = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 1 & 1 \end{pmatrix}$$

which is lower triangular. The off-diagonal elements of N_1^{-1} and N_2^{-1} are simply combined to form L. Our LU decomposition is therefore

$$LU = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 1 & 1 \end{pmatrix} \begin{pmatrix} -3 & 2 & -1 \\ 0 & -2 & 5 \\ 0 & 0 & -2 \end{pmatrix}$$

We need to solve $Ax = b$ then we can write

$$(LU)x = L(Ux) = b$$

we let two systems

$$Ux = y$$

and

$$Ly = b$$

Firt solve

$$Ly = b$$

we have

$$\begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 1 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} -1 \\ -7 \\ -6 \end{pmatrix}$$

Using forward substitution

$$y = \begin{pmatrix} -1 \\ -9 \\ 2 \end{pmatrix}$$

We now solve $Ux = y$, that is

$$\begin{pmatrix} -3 & 2 & -1 \\ 0 & -2 & 5 \\ 0 & 0 & -2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -1 \\ -9 \\ 2 \end{pmatrix}$$

Finaly the solution is

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ -1 \end{pmatrix}$$

6.4 Cholesky method

In the case when the $n * n$ matrix A is symmetric $(A = A^T)$, and positive definite, that is $x^T Ax > 0$ for all nonzero n-vectors x

then it is possible to carry out a factorization without any need for pivoting or scaling. This factorization is known as Cholesky's method, and A can be factored in the form

$$A = LL^T$$

where L is a lower triangular matrix. The construction of L is similar to the one used for Crout's method. Multiplying L by L^T and setting the result equal to A gives

$$l_{ii} = \left[a_{ii} - \sum_{k=1}^{i-1} (l_{ik})^2 \right]^{1/2}, \ i = 1, 2, ..., n. \tag{6.11}$$

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} l_{ik} l_{jk}}{l_{jj}}, \ i = j+1, j+2, ..., n. \ j = 1, 2, ..., n \tag{6.12}$$

Example6.3 Factor the following matrix using Cholesky's method.

$$A = \begin{bmatrix} 16 & 4 & 4 \\ 4 & 26 & 6 \\ 4 & 6 & 11 \end{bmatrix} \tag{6.13}$$

From (6.11) we have

$$\begin{aligned} l_{11} &= \sqrt{16} = 4 \\ l_{21} &= 4/4 = 1,\ l_{22} = 5 \\ l_{31} &= 1,\ l_{32} = 1,\ l_{33} = 3 \end{aligned}$$

Thus,

$$L = \begin{bmatrix} 4 & 0 & 0 \\ 1 & 5 & 0 \\ 1 & 1 & 3 \end{bmatrix} \tag{6.14}$$

and

$$U = L^T \begin{bmatrix} 4 & 1 & 1 \\ 0 & 5 & 1 \\ 0 & 0 & 3 \end{bmatrix} \tag{6.15}$$

The LU method is particularly useful when it is necessary to solve a whole series of systems

$$Ax = b_1,\ Ax = b_2,\ ..., Ax = b_n$$

each of which has the same square coefficient matrix A.

6.5 The Tri-Diagonal Matrix Algorithm (TDMA)

The Tri-Diagonal Matrix Algorithm (TDMA) or Thomas Algorithm is a simplified form of Gaussian elimination that can be used to solve tri-diagonal systems of equations.

Advantages of the TDMA:

Less calculations and less storage than Gaussian Elimination Cost per unknown is independent of the number of unknowns (good scaling w.r.t. iterative methods) Disadvantages of the TDMA:

Round off error is still significant May not be suitable for non-linear problems unless equations can be linearised using the Jacobian Not stable in general, but it is stable if the matrix is diagonally dominant or symmetric positive definite Usually direct calculation using TDMA is not used, but instead iterative solvers are used such as:

Jacobi

Gauss-Seidel

SOR

Conjugate Gradient

Multi-grid

Parallel Multi-grid

6.6 Conclusion

Direct methods to solve linear equation systems, such as LU decomposition, Gaussian elimination, and Cholesky method, offer efficient and reliable solutions. Let's explore each method's characteristics and draw a conclusion about their effectiveness.

- LU Decomposition: LU decomposition factorizes a matrix into lower triangular (L) and upper triangular (U) matrices. By decomposing the coefficient matrix once, LU decomposition allows for efficient solving of multiple linear equation systems with different right-hand sides. The method is computationally efficient and provides a straightforward solution to the system. However, it may require pivoting to handle ill-conditioned matrices, and the decomposition process itself has a computational cost.
- Gaussian Elimination: Gaussian elimination is a widely used method for solving linear equation systems. It involves row operations to transform the coefficient matrix into an upper triangular form, making the system easy to solve through backward substitution. Gaussian elimination is straightforward to implement and works well for general systems. However, it may encounter difficulties when dealing with ill-conditioned matrices or systems with many equations or variables.
- Cholesky Method: The Cholesky method is specifically designed to solve symmetric positive definite matrices. It factors the matrix into the product of a lower triangular matrix and its transpose. This allows for efficient solving of the system using forward and backward substitutions. The Cholesky method is highly efficient and can take advantage of the symmetry and positive definiteness of the matrix, reducing computational effort. However, it is limited to symmetric positive definite systems.

In conclusion, each direct method for solving linear equation systems has its advantages and limitations. LU decomposition is versatile and efficient, but may require pivoting. Gaussian elimination works well for general systems but can encounter difficulties with ill-conditioned matrices. The Cholesky method is highly efficient for symmetric positive definite systems but cannot be applied to more general cases. The choice of method depends on the properties of the system being solved, balancing efficiency and accuracy requirements.

6.7 Annex

6.7.1 MATLAB program

6.7.1.1 Gaussian elimination

```
function GaussElimination()
clear all;
clc;

a = [3 2;
-1 2];

b = [18 2 ];
n = 2;

for k = 1:n-1
for i = k+1:n
factor = a(i,k)/a(k,k);
for j = k:n
a(i,j) = a(i,j) - factor*a(k,j);
end;
b(i) = b(i) - factor*b(k);
end;
end;

fprintf('After Eliminating ... a:  n');
disp(a);
fprintf('————————— ... b:  n');
disp(b);

x(n) = b(n) /a(n,n);
disp(x(n));
fprintf('————————————————————  n');

for i=n-1:-1:1
sum = b(i);
for j = i+1:n
sum = sum - a(i,j)*x(j);
```

```
end;
x(i) = sum/a(i,i);
end;

    disp(x);
```

6.7.1.2 LU method

```
    clear all;
clc;

    A = [3 -0.1 -0.2;0.1 7 -0.3;0.3 -0.2 10];
B = [7.8 -19.3 71.4];

    n = 3;

    for i = 2:n

    sum = B(i);

    for j = 1:i-1
sum = sum - A(i,j)*B(j);
end;

    B(i) = sum;
end;

    X(n) = B(n)/A(n,n);

    for i = n-1:-1:1

    sum = 0;
for j = i+1:n
sum = sum + A(i,j)*X(j);
end;

    X(i) = (B(i) -sum)/A(i,i);
end;
```

6.7.1.3 TDMA algorithm

```
function h = Thomas(ld,md,ud,a)

N = length(md) ;
w = zeros(N, 1) ; g = zeros(N, 1) ;
w(1) = ud(1)/md(1) ; g(1) = a(1)/md(1) ;

if isrow(ud)
ud = ud' ;
end
if isrow(ld)
ld = ld' ;
end
ud = [ud; 0] ; ld = [0; ld] ;

for i=2:N
w(i) = ud(i)/(md(i)-ld(i)*w(i-1)) ;
g(i) = (a(i)-ld(i)*g(i-1))/(md(i)-ld(i)*w(i-1)) ;
end

h = zeros(N, 1) ;
h(N) = g(N) ;
for i=N-1:-1:1
h(i) = -w(i)*h(i+1)+g(i) ;
end
end
```

6.8 Exercises

1. Consider the system

$$\begin{pmatrix} 1 & 1 & 4 & 7 \\ 1 & -1 & -2 & -3 \\ 0 & 2 & 2 & -8 \\ 5 & 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 4 \\ 1 \\ 2 \\ 1 \end{pmatrix}$$

   Attempt to solve the system by Gaussian elimination. Explain what happens

2. Solve the following systems using Gaussian elimination with partial scaled pivoting:

   (a)

$$\begin{aligned} 6x + 4y + 13z &= -23 \\ 2x + y - z &= 4 \\ -3x + 6y - z &= 8 \end{aligned}$$

   (b) $\begin{pmatrix} -2 & -3 & 1 & 2 \\ 7 & 6 & 0 & -3 \\ 0 & 3 & 1 & 5 \\ 2 & -2 & 6 & 6 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 2 \\ -4 \\ 1 \\ 8 \end{pmatrix}$

3. If the Gaussian algorithm with scaled partial pivoting is used on the following system, what is the scaled vector? What is the second pivot row?

$$\begin{pmatrix} 2 & 4 & 29 \\ -1 & 5 & 4 \\ 1 & -1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 15 \\ -3 \\ 1 \end{pmatrix}$$

4. Consider the system

$$\begin{pmatrix} 3 & 3 & 7 \\ 1 & 1 & -1 \\ -2 & -2 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 15 \\ -3 \\ 1 \end{pmatrix}$$

   Attempt to solve the system by Gaussian elimination with scaled partial pivoting. Explain what happens

5. Find the scale vector and the third pivot row if Gaussian elimination with scaled partial pivoting is used on the matrix

$$\begin{bmatrix} -2 & 4 & 5 & 1 \\ -1 & 2 & 9 & 6 \\ 6 & 2 & 4 & -7 \\ 1 & 1 & 0 & 5 \end{bmatrix}$$

6. Use Gaussian elimination with scaled partial pivoting and two-digit rounding arithmetic to solve the linear system

$$\begin{aligned} 0.0001x_1 + 66x_2 &= 65 \\ 2.0x_1 - 28x_2 &= 33 \end{aligned}$$

7. Consider the linear system $Ax = b$, where

$$\begin{pmatrix} 1 & 1+\epsilon \\ 1-\epsilon & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1-\epsilon \end{pmatrix} = \begin{pmatrix} 2-\epsilon^2 \\ 2-2\epsilon \end{pmatrix}$$

The inverse A^{-1} of A is given by

$$A^{-1} = \epsilon^{-2} \begin{bmatrix} 1 & -1-\epsilon \\ -1+\epsilon & 1 \end{bmatrix}$$

Compute for $\epsilon = 10^{-1}, ..., 10^{-6}$ the corresponding solution x of the above system by

(a) using the Gaussian elimination,

(b) direct multiplication $A^{-1}b$

8. Factor the matrix

$$A = \begin{bmatrix} 2 & -1 & 2 \\ 2 & -3 & 3 \\ 6 & -1 & 8 \end{bmatrix}$$

to the LU decomposition and use this decomposition of A to solve $Ax = b$, where $b = [-2, -5, 0]^T$

9. The matrix A in the system of the linear equation

$$Ax = b$$

has the LU decomposition

$$A = LU$$

with

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}, \; U = L^T, \; b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Determine x.

10. Use Cholesky method to solve the linear system $Ax = b$, where

$$A = \begin{bmatrix} 2 & -1 & 2 \\ 2 & -3 & 3 \\ 6 & -1 & 8 \end{bmatrix}, \; b = \begin{bmatrix} 1 \\ 0 \\ 3 \end{bmatrix}$$

Chapter 7 Iterative methods

Note *[Gauss]*

Carl Friedrich Gauss (30 April 1777 - 23 February 1855), was a German mathematician, geodesist, and physicist who made significant contributions to many fields in mathematics and science. He has been referred to as the "Prince of Mathematicians".

In numerical linear algebra, the Gauss-Seidel method, also known as the Liebmann method or the method of successive displacement, is an iterative method used to solve a system of linear equations. It is named after the German mathematicians Carl Friedrich Gauss and Philipp Ludwig von Seidel, and is similar to the Jacobi method. Though it can be applied to any matrix with non-zero elements on the diagonals, convergence is only guaranteed if the matrix is either strictly diagonally dominant, or symmetric and positive definite. It was only mentioned in a private letter from Gauss to his student Gerling in 1823. A publication was not delivered before 1874 by Seidel.

(Source : Wikipédia)

7.1 Introduction

The convergence of iterative methods is achieved only under certain conditions which we will specify. Therefore, great caution should be taken. Moreover, iterative methods, when they converge, become really advantageous only for very large linear systems. Thus, for sparse matrices, iterative methods are more attractive than direct methods.

An iterative scheme for linear systems consists of converting the system $Ax = b$ to the form

$$x = C + Tx$$

After an initial guess, $x^{(0)}$ is selected, the sequence of approximation solution vectors is generated by computing

$$x^{(k+1)} = C + Tx^{(k)}$$

for each $k = 0, 1, 2, 3...$

7.2 Jacobi iterative method

We shall illustrate the method for a $3 * 3$ linear system

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= b_3. \end{aligned}$$

We suppose that the diagonal terms a_{11}, a_{22}, a_{33} are all nonzero.
We begin our iterative scheme by solving each equation for one of the variables, choosing, when possible, to solve for the variable with the largest coefficient

$$\begin{aligned} x_1 &= l_{12}x_2 + l_{13}x_3 + c_1 \\ x_2 &= l_{21}x_1 + l_{23}x_3 + c_2 \\ x_3 &= l_{31}x_1 + l_{32}x_2 + c_3 \end{aligned}$$

Where $l_{ij} = -\frac{a_{ij}}{a_{ii}}$, $c_i = \frac{b_i}{a_{ii}}$, $i = 1, 2, 3.$

Let $x_1^{(0)}$,$x_2^{(0)}$,$x_3^{(0)}$ be an initial approximation of the solution. The $(n+1)$st approximation is obtained from the approximation by writing

$$\begin{aligned} x_1^{(n+1)} &= l_{12}x_2^{(n)} + l_{13}x_3^{(n)} + c_1 \\ x_2^{(n+1)} &= l_{21}x_1^{(n)} + l_{23}x_3^{(n)} + c_2 \\ x_3^{(n+1)} &= l_{31}x_1^{(n)} + l_{32}x_2^{(n)} + c_3 \end{aligned}$$

For $n = 0, 1, 2, ...$

In algorithmic form, the Jacobi iterative method may be presented as follows for an $n * n$ linear system;

Consider the system of n linear equations and n unknowns, given by

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n &= b_2 \\ \vdots\vdots\vdots &= \vdots \\ a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n &= b_n \end{aligned} \tag{7.1}$$

for the unknowns $x_1, x_2, ..., x_n$, given the coefficients $a_{ij}, i, j = 1, 2, ..., n$ and the constants $b_i,\ i = 1, 2, ..., n$.

Consider 7.1 and solve for x_i in the i-th equation to obtain, provided that $a_{ii} \neq 0$,

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

and generate $x_i^{(k)}$ for $k \geq 1$ by

$$x_i^{(k)} = \frac{-\sum_{j=1, j\neq i}^{n} \left(a_{ij}x_j^{(k-1)} + b_i \right)}{a_{ii}},\ i = 1, 2, ..., n. \tag{7.2}$$

The iterative process is terminated when a convergence criterion is satisfied. One commonly used stopping criterion, known as the relative change criteria, is to iterate until

$$\frac{|x^{(k)} - x^{(k-1)}|}{|x^{(k)}|},\ x^{(k)} = \left(x_1^{(k)}, ..., x_n^{(k)} \right)^T$$

is less than a prescribed tolerance $\epsilon > 0$. Contrary to Newton's method for finding the roots of an equation, the convergence or divergence of the iterative process in the Jacobi method does not depend on the initial guess, but depends only on the character of the matrices themselves. However, a good first guess in case of convergence will make for a relatively small number of iterations. This is also true for the Gauss-Seidel method that will be presented in the next section.

Example7.1

Solve the following system using the Jacobi iterative method. Use $\epsilon = 10^{-3}$ and $x^{(0)} = 0$ as the starting vector.

$$\begin{aligned} 7x_1 - 2x_2 + x_3 &= 17 \\ x_1 - 9x_2 + 3x_3 - x_4 &= 13 \\ 2x_1 + 10x_3 + x_4 &= 15 \\ x_1 - x_2 + x_3 + 6x_4 &= 10 \end{aligned}$$

These equations can be rearranged to give

$$\begin{aligned} x_1 &= (17 + 2x_2 - x_3)/7 \\ x_2 &= (-13 + x_1 + 3x_3 - x_4)/9 \\ x_3 &= (15 - 2x_1 - x_4)/10 \\ x_4 &= (10 - x_1 + x_2 - x_3)/6 \end{aligned}$$

which provide the following Jacobi iterative process:

$$\begin{aligned} x_1^{(k+1)} &= (17 + 2x_2^{(k)} - x_3^{(k)})/7 \\ x_2^{(k+1)} &= (-13 + x_1^{(k)} + 3x_3^{(k)} - x_4^{(k)})/9 \\ x_3^{(k+1)} &= (15 - 2x_1^{(k)} - x_4^{(k)})/10 \\ x_4^{(k+1)} &= (10 - x_1^{(k)} + x_2^{(k)} - x_3^{(k)})/6 \end{aligned}$$

Substitute $x^{(0)} = (0, 0, 0, 0)$ into the right-hand side of each of these equations to get

$$\begin{aligned} x_1^{(1)} &= (17 + 2x_2^{(0)} - x_3^{(0)})/7 \\ x_2^{(1)} &= (-13 + x_1^{(0)} + 3x_3^{(0)} - x_4^{(0)})/9 \\ x_3^{(1)} &= (15 - 2x_1^{(0)} - x_4^{(0)})/10 \\ x_4^{(1)} &= (10 - x_1^{(0)} + x_2^{(0)} - x_3^{(0)})/6 \end{aligned}$$

Then

$$\begin{aligned} x_1^{(1)} &= 2.428571429 \\ x_2^{(1)} &= -1.444444444 \\ x_3^{(1)} &= 1.5 \\ x_4^{(1)} &= 1.666666667 \end{aligned}$$

and so $x^{(1)} = (2.428571429, -1.444444444, 1.5, 1.666666667)^T$

7.3 Gauss iterative method

The algorithm for Gauss-Seidel is almost the same as for Jacobi, except that each x-value is improved using the most recent approximations to the values of the other variables. In this case, the $(n+1)$st approximation is obtained from the nth approximation for a $3*3$ system by writing

$$\begin{aligned} x_1^{(n+1)} &= l_{12}x_2^{(n)} + l_{13}x_3^{(n)} + c_1 \\ x_2^{(n+1)} &= l_{21}x_1^{(n+1)} + l_{23}x_3^{(n)} + c_2 \\ x_3^{(n+1)} &= l_{31}x_1^{(n+1)} + l_{32}x_2^{(n+1)} + c_3 \end{aligned}$$

In algorithmic form, Gauss-Seidel may be presented as follows: $x_i^{(k)}$ is generated for $k \geq 1$ by

$$x_i^{(k)} = \frac{-\sum_{j=1}^{i-1}(a_{ij}x_j^{(k)}) - \sum_{j=i+1}^{n}(a_{ij}x_j^{(k-1)}) + b_i}{a_{ii}} \tag{7.3}$$

for $i = 1, 2, ..., n$

The comments following the Jacobi algorithm regarding stopping criteria and starting vectors also apply to the Gauss-Seidel algorithm. Because the new values can be immediately stored in the location that held the old values, the storage requirements for x with the Gauss-Seidel method is half what it would be with the Jacobi method and the rate of convergence is more rapid.

Example7.2

Solve the following system using the Gauss-Seidel iterative method. Use $\epsilon = 10^{-3}$ and $x^{(0)} = 0$ as the starting vector.

$$\begin{aligned} 7x_1 - 2x_2 + x_3 &= 17 \\ x_1 - 9x_2 + 3x_3 - x_4 &= 13 \\ 2x_1 + 10x_3 + x_4 &= 15 \\ x_1 - x_2 + x_3 + 6x_4 &= 10 \end{aligned}$$

From example 7.1, we have

$$\begin{aligned} x_1 &= (17 + 2x_2 - x_3)/7 \\ x_2 &= (-13 + x_1 + 3x_3 - x_4)/9 \\ x_3 &= (15 - 2x_1 - x_4)/10 \\ x_4 &= (10 - x_1 + x_2 - x_3)/6 \end{aligned}$$

which provide the following Gauss-Seidel iterative process:

$$
\begin{aligned}
x_1^{(k+1)} &= (17 + 2x_2^{(k)} - x_3^{(k)})/7 \\
x_2^{(k+1)} &= (-13 + x_1^{(k+1)} + 3x_3^{(k)} - x_4^{(k)})/9 \\
x_3^{(k+1)} &= (15 - 2x_1^{(k+1)} - x_4^{(k)})/10 \\
x_4^{(k+1)} &= (10 - x_1^{(k+1)} + x_2^{(k+1)} - x_3^{(k+1)})/6
\end{aligned}
$$

Substitute $x^{(0)} = (0, 0, 0, 0)$ into the right-hand side of each of these equations to get

$$
\begin{aligned}
x_1^{(1)} &= (17 + 2x_2^{(0)} - x_3^{(0)})/7 \\
x_2^{(1)} &= (-13 + 2.428571429 + 3(0) - 0)/9 \\
x_3^{(1)} &= (15 - 2(2.428571429) - 0)/10 \\
x_4^{(1)} &= (10 - 2.428571429 - 1.1746031746 - 1.0142857143)/6
\end{aligned}
$$

Then we obtain

$$
\begin{aligned}
x_1^{(1)} &= 2.428571429 \\
x_2^{(1)} &= -1.1746031746 \\
x_3^{(1)} &= 1.0142857143 \\
x_4^{(1)} &= 0.8970899472
\end{aligned}
$$

and so $x^{(1)} = (2.428571429, -1.1746031746, 1.0142857143, 0.8970899472)^T$.

7.4 Relaxation method

A method for the iterative solution of a system of linear algebraic equations $Ax = b$, the elementary step of which consists of varying only one component of the vector of unknowns, the number of variable components being chosen in a specific cyclic order. The relaxation method is most often used for solving systems with a positive-definite matrix A.

If one component of the vector of unknowns x^k is varied such that for the new approximation x^{k+1} the quadratic form $(A(x^{k+1} - x), x^{k+1} - x)$ is minimized, then the relaxation method is called a complete relaxation method. If, however, after one elementary step the value of the quadratic form is only reduced and not minimized, the relaxation method is called an incomplete relaxation method.

The best investigated method is that of successive upper relaxation, where the matrix A possesses the so-called property (A) and is ordered accordingly. A matrix A is called a matrix possessing property (A) if there is a permutation matrix P such that the matrix PAP^T has the form

$$
\begin{Vmatrix}
D_1 & H \\
K & D_2
\end{Vmatrix},
$$

where D_1 and D_2 are square diagonal matrices.

The iteration scheme of the relaxation method is as follows:

$$(D+\omega L)x^{k+1} = ((1-\omega)D - \omega U)x^k + \omega b \quad k = 0, 1, \ldots,$$

where ω is the relaxation parameter, D is the diagonal, L is the lower-triangular and U is the upper-triangular matrix in the decomposition $A = D + L + U$. If $\omega > 1$, then the method is called an upper relaxation method (over-relaxation), and if $\omega \leq 1$, a lower relaxation method. The parameter ω is chosen from the condition of minimization of the [[Spectral radius|spectral radius]] of the matrix S of transfer from iteration to iteration:

$$S = (D+\omega L)^{-1}((1-\omega)D - \omega U).$$

If A is a symmetric matrix with positive diagonal elements and λ_i are the roots of the determinant equation $\det(L+\lambda D+U) = 0$, then the optimum value of the parameter ω is given by the formula

$$\omega = \omega_0 = \frac{2}{1+\sqrt{1-\lambda_0^2}},$$

where $\lambda_0^2 = \max \lambda_i^2$. For $\omega = \omega_0$ the spectral radius of S is equal to

$$\omega_0 - 1 = \frac{1-\sqrt{1-\lambda_0^2}}{1+\sqrt{1-\lambda_0^2}} < 1.$$

Cases are examined where some λ_i are complex. Block relation methods have been developed.

7.5 Convergence

The matrix formulation of the Jacobi and Gauss-Seidel iterative methods can be obtained by splitting the matrix A into the sum

$$A = D + L + U$$

Where D is the diagonal of A, L the lower triangular part of A, and U the upper triangular part of A. That is,

$$D = \begin{bmatrix} a_{11} & 0 & \ldots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & a_{nn} \end{bmatrix}, L = \begin{bmatrix} 0 & \ldots & & 0 \\ a_{21} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \\ a_{n1} & \ldots & a_{n,n-1} & 0 \end{bmatrix}, U = \begin{bmatrix} 0 & a_{12} & \ldots & a_{1n} \\ \vdots & \ddots & \ddots & \vdots \\ & & \ddots & a_{n-1,n} \\ 0 & \ldots & & 0 \end{bmatrix}.$$

Thus, the system 7.1 can be written as

$$(D + L + U)x = b$$

The Jacobi method in matrix form is

$$Dx^{(k)} = -(L + U)x^{(k-1)} + b$$

and the Gauss-Seidel method in matrix form is

$$(D + L)x^{(k)} = -Ux^{(k-1)} + b$$

Before stating the theorem on the convergence of the Jacobi and Gauss-Seidel methods, we make the following definition:

Definition 7.1

An $n * n$ matrix A is strictly diagonally dominant if

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|, \ \ for \ \ i = 1, 2, ..., n.$$

♣

We now give a sufficient condition for Jacobi and Gauss-Seidel to convergence.

Theorem 7.1. Jacobi and Gauss-Seidel convergence theorem

If A is strictly diagonally dominant, then the Jacobi and Gauss-Seidel methods converge for any choice of the starting vector $x^{(0)}$.

♡

Proof The proof can be found in advanced texts on numerical analysis.

Example7.3 Consider the system of equation

$$\begin{bmatrix} 3 & 1 & 1 \\ -2 & 4 & 0 \\ -1 & 2 & -6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ 2 \end{bmatrix}$$

The coefficient matrix of the system is strictly diagonally dominant since

$$\begin{aligned} |a_{11}| &= |3| = 3 > |1| + |1| = 2 \\ |a_{22}| &= |4| = 4 > |-2| + |0| = 2 \\ |a_{33}| &= |-6| = 6 > |-1| + |2| = 3 \end{aligned}$$

Hence, if the Jacobi or Gauss-Seidel method is used to solve the system of equations, then it will converge for any choice of the starting vector $x^{(0)}$.

7.6 Conclusion

Iterative methods, such as the Jacobi method and the Gauss-Seidel method, offer approaches to solve linear equation systems by approximating the solution iteratively. Let's analyze each method and draw a conclusion about their effectiveness.

- Jacobi Method: The Jacobi method is a relatively simple iterative technique that solves a linear equation system by updating each variable based on the previous iteration. It requires the transformation of the system into an equivalent form with the diagonal elements isolated. While it may take longer to converge compared to more advanced iterative methods, the Jacobi method has the advantage of being straightforward to implement and has relatively low computational complexity. However, it may converge slowly or fail to converge for ill-conditioned or non-diagonally dominant systems.
- Gauss-Seidel Method: The Gauss-Seidel method is an extension of the Jacobi method, where the updated values for each variable are used immediately within the current iteration. This method exhibits improved convergence properties as it utilizes the most recent approximations throughout the iteration. The Gauss-Seidel method is generally faster in terms of convergence compared to the Jacobi method, particularly for systems that are diagonally dominant. However, it may still struggle with slow convergence for certain scenarios, and its convergence behavior is highly dependent on the characteristics of the system.

In conclusion, both the Jacobi and Gauss-Seidel methods offer iterative approaches to solving linear equation systems. The Jacobi method is simple to implement but may have slower convergence and potentially fail for non-diagonally dominant systems. The Gauss-Seidel method improves upon the Jacobi method by utilizing more recent approximations, leading to faster convergence. However, even the Gauss-Seidel method may experience slow convergence in some cases. The choice of method depends on the specific properties of the system being solved, including diagonal dominance and condition number, as well as the desired trade-off between accuracy and computational efficiency.

7.7 Annex

7.7.1 MATLAB program

7.7.1.1 Gauss Seidel method

```
clear all;
clc;

A = [3 -0.1 -0.2;
0.1 7 -0.3;
0.3 -0.2 10];

B = [7.85 -19.3 71.4];
X = [0 0 0];
n = 3;

lambda = 1.1;

for i = 1: n;

dummy = A(i,i);

for j = 1:n
A(i,j) = A(i,j)/dummy;
end;
B(i) = B(i) / dummy;

end;

for i = 1:n
sum = B(i);

for j = 1:n
if i  = j
sum = sum - A(i,j)*X(j);
end;
end;
```

```
        X(i) = sum;
end;

        iter = 1;

        while iter < 250

        sentinel = 1;

        for i = 1:n
old = X(i);
sum = B(i);

        for j = 1:n
if i =j
sum = sum - A(i,j)*X(j);
end;
end;
X(i) = lambda*sum +(1.0-lambda)*old;

        if sentinel == 1 and X(i) ~= 0
ea = abs((X(i)-old)/X(i))*100;
if ea > 0.0005
sentinel = 0;
end;
end;
end;

        iter = iter + 1;

        if sentinel == 1 || (iter >= 150)
break;
end;

        end;
```

7.8 Exercises

1. The solution of the system

$$\begin{aligned} x_1 + 3x_2 &= 3 \\ 4x_1 + x_2 &= 1 \end{aligned}$$

is $x_1 = 0$ and $x_2 = 1$. Apply the Jacobi and Gauss-Seidel methods to this rearrangement, starting with a vector close to the solution. Which method diverges most rapidly?

2. Consider the linear system

$$\begin{pmatrix} 4 & 0 & 1 \\ 1 & 4 & 1 \\ 1 & 0 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \\ -10 \end{pmatrix}$$

(a) Use naive Gaussian elimination to solve the system

(b) Find the matrix that needs to be analyzed to determine whether Jacobi's iteration method will converge for this problem.

(c) Perform one iteration of Gauss-Seidel, with starting guess $[12 - 3]^T$.

3. Consider the linear system

$$\begin{aligned} x_1 + 4x_2 &= -15 \\ 5x_1 + x_2 &= 1 \end{aligned}$$

Apply the Jacobi method to this arrangement, beginning with a vector close to the solution $x = [1.01, -4.01]$ and observe divergence. Now interchange the equations to solve the system again and observe convergence

4. Consider the linear system

$$\begin{pmatrix} 4 & 1 \\ 2 & 5 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$$

(a) Set up the Jacobi iteration with initial guess $x = 3$, $y = 11$ and perform two iteration of Jacobi's method.

(b) Set up the Gauss-Seidel iteration with initial guess $x = 3$, $y = 11$ and perform two iteration of the Gauss-Seidel method.

(c) Explain why both methods should converge for this case.

5. Solve the following systems using the Gauss-Seidel iterative method

(a)

$$\begin{aligned} x - y + 2z - w &= -1 \\ 2x + y - 2z - 2w &= -2 \\ -x + 2y - 4z + w &= 1 \\ 3x - 3w &= -3 \end{aligned}$$

(b)

$$\begin{aligned} 5x - y + 3z &= 3 \\ 4x + 7y - 2z &= 2 \\ 6x - 3y + 9z &= 9 \end{aligned}$$

6. Consider the system of equations

$$\begin{pmatrix} 4 & 0 & 1 \\ 1 & 4 & 1 \\ 1 & 0 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 5 \\ 3 \\ -10 \end{pmatrix}$$

(a) Find the matrix that needs to be analyzed to determine whether Jacobi's iteration method will converge for this problem

(b) Perform one iteration of Gauss-Seidel, with starting guess $[12 - 3]^T$.

7. Given the linear system

$$\begin{bmatrix} 3 & -5 & 47 & 20 \\ 11 & 16 & 17 & 10 \\ 56 & 22 & 11 & -18 \\ 17 & 66 & -12 & 7 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} 18 \\ 26 \\ 34 \\ 82 \end{bmatrix}$$

Reorder the equation of the system and use the Jacobi iteration to solve it.

8. The following system has an approximate solution $x_1 = 3.072$, $x_2 = -5.497$, $x3 = -2.211$, and $x_4 = 4.579$

$$\begin{pmatrix} 1.20 & 0.45 & 0.35 & 0.45 \\ 0.89 & 2.59 & -0.33 & -0.22 \\ 0.71 & 0.78 & 4.01 & -0.88 \\ 0.11 & 0.55 & 0.66 & 3.39 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 2.5 \\ -11.781 \\ -15.002 \\ 11.378 \end{pmatrix}$$

Use both the Gauss-seidel and Jacobi methods to approximate the solution of the system.

9. Use Jacobi's method and Gauss-Seidel to solve $Ax = b$ such that

$$A = \begin{bmatrix} -4 & 2 & 0 & . & . & . & 0 \\ 2 & -4 & 2 & 0 & . & . & 0 \\ 0 & 2 & -4 & 2 & 0 & . & 0 \\ 0 & 0 & 2 & -4 & 2 & 0 & 0 \\ 0 & . & 0 & . & . & . & 0 \\ 0 & . & . & 0 & 2 & -4 & 2 \\ 0 & . & . & . & 0 & 2 & -4 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ . \\ . \\ . \\ 1 \end{bmatrix}$$

10. Use Jacobi's method and Gauss-Seidel to solve the linear system

$$\begin{aligned} x - y + z &= 3 \\ 2x + y - 5z &= 2 \\ x + 3y + 9z &= 7 \end{aligned}$$

Bibliography

[1] Abdelwahab Kharab, Ronald B. Guenther *An Introduction to Numerical Methods A MATLAB Approach*, by Taylor and Francis Group, LLC, (2009)

[2] Nadjib Boussetila *Cours analyse numérique* , 2ème année CPST, (2014)

[3] Jeffrey R. Chasnov *Numerical methods for engineers*, The Hong Kong University of Science and Technology, (2012)