Numerical Analysis
Theory

Christian Rossi

Academic Year 2023-2024

Abstract

The topics of the course are:

- Floating-point arithmetic: different sources of the computational error; absolute vs relative errors; the floating point representation of real numbers; the round-off unit; the machine epsilon; floating-point operations; over- and under-flow; numerical cancellation.
- Numerical approximation of nonlinear equations: the bisection and the Newton methods; the fixed-point iteration; convergence analysis (global and local results); order of convergence; stopping criteria and corresponding reliability; generalization to the system of nonlinear equations (hints).
- Numerical approximation of systems of linear equations: direct methods (Gaussian elimination method; LU and Cholesky factorizations; pivoting; sparse systems: Thomas algorithm for tridiagonal systems); iterative methods (the stationary and the dynamic Richardson scheme; Jacobi, Gauss-Seidel, gradient, conjugate gradient methods (hints); choice of the preconditioner; stopping criteria and corresponding reliability); accuracy and stability of the approximation; the condition number of a matrix; over- and under-determined systems: the singular value decomposition (hints).
- Numerical approximation of functions and data: Polynomial interpolation (Lagrange form); piecewise interpolation; cubic interpolating splines; least-squares approximation of clouds of data.
- Numerical approximation of derivatives: finite difference schemes of the first and second order; the undetermined coefficient method.
- Numerical approximation of definite integrals: simple and composite formulas; midpoint, trapezoidal, Cavalieri-Simpson quadrature rules; Gaussian formulas; degree of exactness and order of accuracy of a quadrature rule.
- Numerical approximation of ODEs: the Cauchy problem; one-step methods (forward and backward Euler and Crank-Nicolson schemes); consistency, stability, and convergence (hints).

Contents

CHAPTER 1

Introduction

1.1 Numerical analysis and errors

Numerical analysis is the field of mathematics dealing with methods to find the solutions of certain mathematical problems with an electronic calculator. It is the intersection between math and computer science.

On the other hand, scientific computing also deals with the model formalization and so it needs also engineering knowledge.

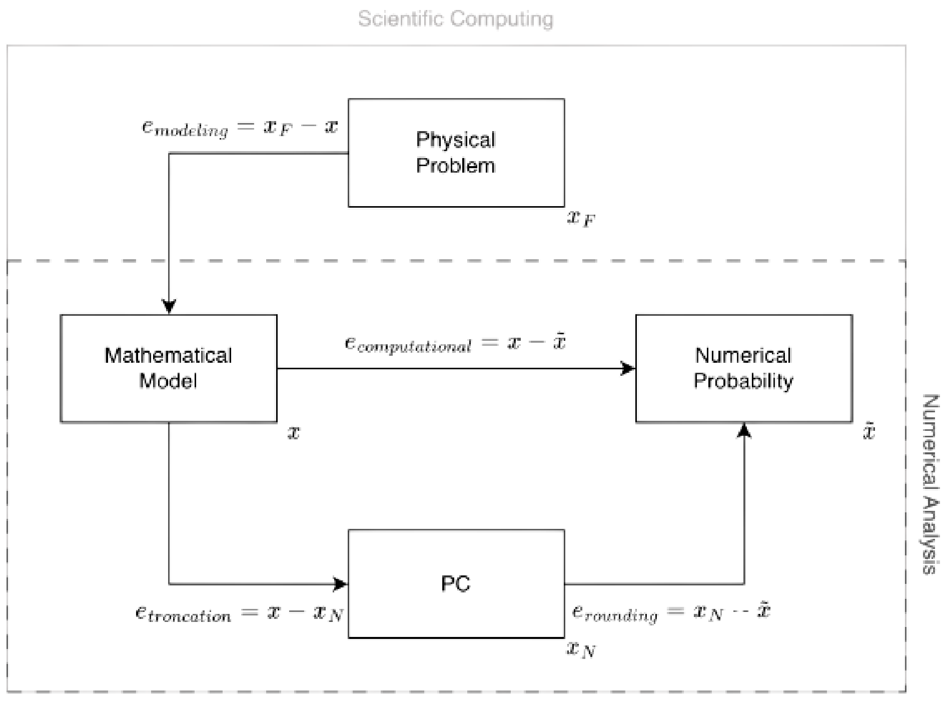


Figure 1.1: Difference between numerical analysis and scientific computing

As it is possible to see from the diagram every step of the computation have to deal with errors. The possible types of errors are:

- Absolute: $|x - \tilde{x}|$

- Relative: $\dfrac{|x-\tilde{x}|}{|x|}$, where $x \neq 0$

The relative error is more precise because it compares the error with the measure quantity.

Example : Let us consider $x = 100$ and $\tilde{x} = 100.1$. The errors in this case are:

$$e_{abs} = |x - \tilde{x}| = |100 - 100.1| = 0.1$$

$$e_{rel} = \frac{|x-\tilde{x}|}{|x|} = \frac{|100-100.1|}{|100|} = 0.001$$

Let us consider $x = 0.2$ and $\tilde{x} = 0.1$. The errors in this case are:

$$e_{abs} = |x - \tilde{x}| = |0.2 - 0.1| = 0.1$$

$$e_{rel} = \frac{|x-\tilde{x}|}{|x|} = \frac{|0.2-0.1|}{|0.2|} = 0.5$$

The result are that the measures have the same absolute error (10%), but the relative error is much grater in the second example (50% vs 0.1%). This result proves that the relative error is the most precise.

1.2 Floating point

A calculator can only handle a finite quantity of numbers and compute a finite number of operations. For those reason the set of the real numbers $\mathbb{R}$ is indeed represented by a finite set of machine numbers $\mathbb{F} = \{-\tilde{a}_{min}, \dots, \tilde{a}_{max}\}$ called floating points numbers. The function used to find the corresponding value in $\mathbb{F}$ to a number in $\mathbb{R}$ is $fl(x)$ that does an operation called truncation and rounding.

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

$$fl(x) = (-1)^s(0.a_1a_2\dots a_t)\beta^e$$

where:

- $\beta \geq 2$ is the basics, an integer that determines the numeric system.
- $m = (0.a_1a_2\dots a_t)$ is the mantissa.
- $e \in \mathbb{Z}$ is the exponent such that $L < e < U$, with $L < 0$ and $U > 0$.
- $s = \{0, 1\}$ is the sign.

In the definition of the numbers in the mantissa set we have to set the constraint $a_1 \neq 0$ to ensure the uniqueness of the representation. In this case we say that the number is normalized.

The set of floating points has the following characteristic values are:

- Machine epsilon, that is the distance between one and the smallest floating point number greater than one, and it is equal to:

$$\epsilon_M = \beta^{1-t}$$

- Round-off error, that is the relative error that is committed when substituting $x \in \mathbb{R}-\{0\}$ with his corresponding $fl(x) \in \mathbb{F}$. It is limited by:

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

  where $x \neq 0$.

- The biggest and the smallest numbers in the set are found with the formula:

$$x_{min} = \beta^{L-1}$$

$$x_{max} = \beta^U(1-\beta^{-t})$$

Example : In MATLAB the floating point set is defined with the following variables:

$$(\beta = 2, t = 53, L = -1021, U = 1024)$$

With the command *eps* we can find the machine epsilon, that in MATLAB case is:

$$\epsilon_M = 2.22 \cdot 10^{-16}$$

With the command *realmin* and *realmax* we can find the smallest and the largest numbers representable that are equal to:

$$x_{min} = 2.225073858507201 \cdot 10^{-308}$$

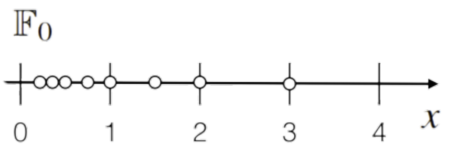
$$x_{max} = 1.797693134862316 \cdot 10^{308}$$

Since not all the numbers in the $\mathbb{R}$ set are also in the $\mathbb{F}$ set, in the second one there is no continuity. It is possible to demonstrate that while we are increasing the values of the numbers we are also increasing the distance between two consecutive numbers in $\mathbb{F}$.

Example : Let us consider the floating number set $\mathbb{F}(2, 2, -1, 2)$. The characteristic values of this set are:

- $\epsilon_M = \beta^{1-t} = 0.5$.
- $x_{min} = \beta^{L-1} = 0.25$.
- $x_{max} = \beta^U(1-\beta^t) = 3$.
- $\#\mathbb{F} = 2\beta^{t-1}(\beta-1)(U-L+1)+1 = 16$.

The exponent can have the values $-1, 0, 1$ and 2. The mantissa will be like $(a_1a_2)_\beta$ because $t = 2$. The possible positive values are reported in the figure.



e	-1	0	1	2
$m = (10)_2 = 2$	$\frac{1}{4}$	$\frac{1}{2}$	1	2
$m = (11)_2 = 3$	$\frac{3}{8}$	$\frac{3}{4}$	$\frac{3}{2}$	3

The other important aspect is that the passage between the two sets causes the loss of two important properties such as associativity end the neutral number for the sum.

CHAPTER 2

Nonlinear equations

2.1 Introduction

To solve a nonlinear equation $f(x)$ we have to find $\alpha \in \mathbb{R}$ that is a zero of f such that $f(\alpha) = 0$.

Definition

The point α is said to be a *zero with multiplicity* m if:

- $f(\alpha) = f^{'}(\alpha) = \cdots = f^{(m-1)}(\alpha) = 0$.
- $f^{(m)}(\alpha) \neq 0$.

If m is odd the function changes sign across the zero. It does not change the sign if n is even.

2.2 Iterative methods

The set of all the polynomials of degree n is denoted by the symbol $\mathbb{P}_n$ that contains all the polynomial that have a grade less or equal to n.

Theorem

There is no solution in radicals to general polynomial equations of degree five or higher with arbitrary coefficients.

So, to solve polynomials with a degree higher than four we need to use the iterative methods. The general idea of those methods is the following:

1. Select an arbitrary initial value $x^{(0)}$ called initial guess, that is a hypothetical value for α.
2. Use the selected value as an input for a black-box function.
3. Use the output of the black-box function as the new $x^{(0)}$ and return to point one.

After some iterations (that depends on the chosen method) we will have a set of values $\{x^{(n)}\}$ convergent such that:

$$\lim_{n\to\infty} = \alpha$$

and the error related to the value found for α is equal to:

$$\lim_{n\to\infty} e^n = 0$$

That Rightarrow that the error can be also written as:

$$e^n = \alpha - x^{(n)}$$

All the methods we are going to see constructs a sequence $x^{(1)}, x^{(2)}, \dots, x^{(n)}$ of numbers that hopefully converges to α:

$$\lim_{k\to+\infty} \left|x^{(k)} - \alpha\right| = 0$$

Definition (*order of convergence*)

An iterative method for the approximation of the zero α of the function $f(x)$ is convergent with order q if and only if for $k > k_0$:

$$\left|x^{(k)} - \alpha\right| \leq c\left|x^{(k+1)} - \alpha\right|^q$$

We can have two cases:

- If $q = 1$ it is called linear convergence and there are the constraint $0 < c < 1$.
- If $q > 1$, c can be any positive number grater than zero.

2.3 Stopping conditions

The iterative methods need a stopping criterion. It can be on four possible conditions:

- On the error, stops if $\left|x^{(k)} - \alpha\right| \leq \epsilon_e$.
- On the residual, stops if $\left|f\left(x^{(k)}\right)\right| \leq \epsilon_r$.
- On the step length, stops if $\left|x^{(k)} - x^{(k-1)}\right| \leq \epsilon_s$.
- On the max numbers of iterations, stops if $k \leq k_{max}$.

If none of the first three stopping criterions are satisfied, it means that there are no convergence.

2.4 Bisection method

Theorem

Let $f(x)$ be a continuous function on the interval $I = (a, b)$, that is $f \in C^0([a, b])$. If $f(a)f(b) < 0$, then there exists at least one zero $\alpha \in I$ of $f(x)$.

Let us assume that exist a unique zero, and let us call it α. The strategy of the bisection method is to have the given interval and select a sub-interval where f features a sign change. Following such procedure it is guaranteed that every interval selected this way will contain α. The sequence $\{x^{(k)}\}$ of the midpoints of these sub-intervals will inevitably tend to α since the length of the sub-intervals tends to zero as k tends to infinity. We have that:

$$\left|x^{(k)} - \alpha\right| \leq \frac{1}{2}\left|b^{(k)} - a^{(k)}\right|$$

So, since the first part of the equation is similar to the condition on the error, we have that the stopping criterion will be:

$$\left|b^{(k)} - a^{(k)}\right| \leq 2\epsilon_e$$

Now that we have the tolerance ϵ we can find that:

$$k_{min} = \left\lceil \log_2\left(\frac{|b-a|}{\epsilon}\right) - 1 \right\rceil$$

The inputs for the algorithm are: a function $f \in C(\mathbb{R})$ and an interval $[a, b]$ such that $f(a)f(b) \leq 0$.

Algorithm 1 Algorithm for the bisection method

```
1: for $k = 0, 1, \ldots, n$ do
2:     $x^{(k)} = \frac{a+b}{2}$
3:     if $\left|b^{(k)} - a^{(k)}\right| \leq 2\epsilon_e$ then
4:         return $x^{(k)}$
5:     else if $f(x^{(k)})f(a) < 0$ then
6:         $b \leftarrow x^{(k)}$
7:     else
8:         $a \leftarrow x^{(k)}$
9:     end if
10: end for
```

The pros are:

- I can control the maximal error.
- Convergence is guaranteed.
- Use only evaluation of f

The cons are:

- Work only if f changes sign.
- Convergence is slow.

2.5 Newton method

The sign of the given function f at the endpoints of the sub-intervals is the only information exploited by the bisection method. A more efficient method can be constructed by exploiting the values attained by f and its derivative. If f is differentiable we have that:

$$y(x) = f(x^{(k)}) + f^{'}(x^{(k)})(x - x^{(k)})$$

provides the equation of the tangent to the curve $(x, f(x))$ at the point $x^{(k)}$. If we pretend that $x^{(k+1)}$ is such that $f(x^{(k+1)}) = 0$, we obtain:

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f^{'}(x^{(k)})} \qquad k \geq 0$$

provided $f^{'}(x^{(k)}) \neq 0$. This method is known as Newton's method and corresponds to computing the zero of f locally replacing f by its tangent line.

Obviously, this method converges in a single step when f is linear.

The Newton method in general doe not converge for all possible choices of $x^{(0)}$, but only for those values of $x^{(0)}$ which are sufficiently close to α. Since we don't know the value of α a possible initial value $x^{(0)}$ can be obtained by resorting to a few iterations of the bisection method or through an investigation of the graph of f.

If $f \in C^2(\mathbb{R})$, $f^{'}(\alpha) \neq 0$, and $x^{(0)}$ is taken sufficiently near α the Newton method converges quadratically. In the case of zeros with multiplicity m larger than one Newton method converges linearly. To avoid this degradation it is possible to use the modified Newton method:

$$x^{(k+1)} = x^{(k)} - m\frac{f(x^{(k)})}{f^{'}(x^{(k)})} \qquad k \geq 0$$

provided $f^{'}(x^{(k)}) \neq 0$.

Another modification to this method is the quasi-Newton method that uses a derivative by finite difference:

$$f^{'}(x^{(k)}) \simeq \frac{f(x^{(k)} + h) - f(x^{(k)})}{h}$$

The inputs for the algorithm are: a function $f \in C^1(\mathbb{R})$ and an initial guess $x^{(0)} \in \mathbb{R}$.

Algorithm 2 Algorithm for the basic Newton method

```
1: for $k = 0, 1, \ldots, n$ do
2:     $x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f^{'}(x^{(k)})}$
3:     if $k > k_{max} \vee \left|x^{(k)} - x^{(k-1)}\right| \leq \epsilon_s \vee \left|f\left(x^{(k+1)}\right)\right| \leq \epsilon_r$ then
4:         return $x^{(k+1)}$
5:     end if
6: end for
```

The pros are:

- Fast convergence.
- Works also for zeros with even multiplicity.

The cons are:

- Require computation of the derivative.
- Choice of the right $x^{(0)}$.

2.6 Secant method

For the computation of the zeros of a function f whose derivative is not available in analytical form, the Newton method cannot be applied. However, we should be able to compute the function f at any arbitrary point, and we could replace the exact value $f'(x^{(k)})$ with an incremental ratio based on previously computed values of f. The secant method exploits this strategy and converges super-linearly ($q = 1.6$).

The inputs for the algorithm are: two initial guesses $x^{(0)} \in \mathbb{R}$ and $x^{(1)} \in \mathbb{R}$.

Algorithm 3 Algorithm for the secant method

```
1: for $k = 0, 1, \dots, n$ do
2:     $x^{(k+1)} = x^{(k)} - f(x^{(k)}) \dfrac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})}$
3:     if $k > k_{max} \vee \left|x^{(k)} - x^{(k-1)}\right| \le \epsilon_s \vee \left|f\left(x^{(k+1)}\right)\right| \le \epsilon_r$ then
4:         return $x^{(k+1)}$
5:     end if
6: end for
```

2.7 Fixed point method

Given a function $\phi : [a,b] \to \mathbb{R}$, find $\alpha \in [a,b]$ such that $\alpha = \phi(\alpha)$. If such an α exists it will be called a fixed point of ϕ, and it could be computed by the following algorithm:

$$x^{(k+1)} = \phi(x^{(k)}) \qquad k \ge 0$$

where $x^{(0)}$ is an initial guess. This algorithm is called fixed point iteration and ϕ is said to be the iteration function.

Theorem

1. Let us suppose that $\phi(x)$ is continuous in $[a,b]$ and such that $\phi(x) \in [a,b]$ for every $x \in [a,b]$; then, there exists at least a fixed point $\alpha \in [a,b]$.

2. Moreover, if

$$\exists L < 1 \text{such that} \left|\phi(x_1) - \phi(x_2)\right| \le L \left|x_1 - x_2\right| \qquad \forall x_1, x_2 \in [a,b]$$

then there exists a unique fixed point $\alpha \in [a,b]$ of ϕ and the sequence converges to α, for any choice of initial guess $x^{(0)} \in [a,b]$.

Proof of the first proposition: The function $g(x) = \phi(x) - x$ is continuous in $[a,b]$ and, thanks to assumption made on the range of ϕ, it holds $g(a) = \phi(a) - a \ge 0$ and $g(b) = \phi(b) - b \ge 0$. By applying the theorem of zeros of continuous functions, we can conclude that g has at least one zero in $[a,b]$. ■

Proof of the second proposition: Indeed, should two different fixed points α_1 and α_2 exist, then:

$$|\alpha_1 - \alpha_2| = |\phi(\alpha_1) - \phi(\alpha_2)| \le L\,|\alpha_1 - \alpha_2| < |\alpha_1 - \alpha_2|$$

which cannot be. We prove now that the sequence $x^{(k)}$ converges to the unique fixed point α when $k \to \infty$, for any choice of initial guess $x^{(0)} \in [a,b]$. It holds:

$$\frac{\left|x^{(k)} - \alpha\right|}{\left|x^{(0)} - \alpha\right|} \leq L^k$$

Passing to the limit as $k \to \infty$, we obtain $\lim_{k\to\infty} \left|x^{(k)} - \alpha\right| = 0$, which is the desired result. ■

In practice it is often very difficult to choose a priori an interval $[a,b]$ for which the assumptions of previous proposition are fulfilled. In such cases the following local convergence result will be useful.

Theorem

Let α be a fixed point of a function ϕ which is continuous and continuously differentiable in a suitable neighborhood $\mathcal{J}$ of α. If $\left|\phi^{'}(\alpha)\right| < 1$, then there exists $\delta > 0$ for which $\{x^{(k)}\}$ converges to α, for every $x^{(0)}$ such that $\left|x^{(0)} - \alpha\right| < \delta$. Moreover, it holds:

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

For the hypothesis of the Ostrowski theorem we have that:

$$\exists \delta | \left|\phi^{'}(\alpha)\right| < 1 \qquad \forall x \, |x - \alpha| < \delta$$

We will call this interval I_δ. If I choose a point x inside this interval, I will have that $\phi(x) \in I_\delta$.
Proof: Let us take $|\phi(x) - \alpha|$, where it is satisfied the relation $|x - \alpha| < \delta$. We have

$$|\phi(x) - \alpha| = |\phi(x) - \phi(\alpha)| \leq \left|\phi^{'}(\xi)(x - \alpha)\right|$$

So, we have that ξ is between x and $alpha$, and so it holds $\xi \in I_\delta$. At the meantime we have by hypothesis that:

$$\left|\phi^{'}(\xi)\right| < 1$$

Now we can write

$$\left|\phi^{'}(\xi)(x - \alpha)\right| = \left|\phi^{'}(\xi)\right| |x - \alpha| \leq |x - \alpha| < \delta$$

In the end we found that:

$$|\phi(x) - \alpha| < \delta$$

So we have proved that $\phi(x) \in I_\delta$. ■

Proof Ostrowski: Thanks to the Lagrange theorem, for any $k \geq 0$, there exists a point ξ_k between $x^{(k)}$ and α such that $\left|x^{(x)} - \alpha\right| = \left|\phi(x^{(x)}) - \phi(\alpha)\right| = \left|\phi^{'}(\xi_k)(x^{(x)} - \alpha)\right|$. We found that ξ_k is between $x^{(k)}$ and $x^{(k+1)}$. So if $x^{(k)} \in I_\delta$, for the previous proof I have that also $x^{(k+1)} \in I_\delta$ and $\xi_k \in I_\delta$. And so we have that:

$$\frac{\left|x^{(k+1)} - \alpha\right|}{\left|x^{(k)} - \alpha\right|} = \left|\phi^{'}(\xi_k)\right|$$

But since $\xi_k \to \alpha$, we have the convergence formula. ■

The fixed point iteration converge at least linearly. When $\left|\phi^{'}(\alpha)\right| > 1$, if $x^{(k)}$ is sufficiently close to α, such that $\left|\phi^{'}(x^{(k)})\right| > 1$, then $\left|\alpha - x^{(k+1)}\right| > \left|\alpha - x^{(k)}\right|$, and the sequence cannot converge to the fixed point. On the contrary, when $\left|\phi^{'}(\alpha)\right| = 1$, non conclusion can be drawn since either convergence or divergence could take place, depending on the properties of the iteration function $\phi^{'}(x)$.

Proposition

Assume that all hypothesis of Ostrowski's theorem are satisfied. In addition, assume that ϕ is twice continuously differentiable and that $\phi^{'}(\alpha) = 0$ and $\phi^{''}(\alpha) \neq 0$. Then, the fixed point iteration converge with order two and:

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

Given a simple zero such that $f(\alpha) = 0$ we can use the fixed point method to derive the Newton method:

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

If we derive the previous function we obtain:

$$\phi^{'}_N(x) = \frac{f(x)f^{''}(x)}{\left[f'(x)\right]^2}$$

And we can say that:

$$\phi^{'}_N(\alpha) = \frac{f(\alpha)f^{''}(\alpha)}{\left[f'(\alpha)\right]^2} = 0$$

Proof: We can note that the denominator is not null (the zero is simple, so the first derivative is not null). We assumed that $f(\alpha) = 0$, and we can say nothing about $f^{''}(\alpha)$. But since we have

$$\phi^{'}_N(\alpha) = \frac{f(\alpha)f^{''}(\alpha)}{\left[f'(\alpha)\right]^2} = \frac{0 \cdot n}{n} = \frac{0}{n} = 0$$

We have that the function evaluated in α is zero. ■

The stopping criterion for the fixed point iteration method is:

$$\left|x^{(k+1)} - x^{(k)}\right| \leq \epsilon_s$$

Proof not null derivative: We have that the error in a certain iteration is:

$$\left|x^{(k+1)} - \alpha\right| = \left|\phi(x^{(k)}) - \phi(\alpha)\right| \leq \left|\phi^{'}(\xi_k)\right| \left|x^{(k)} - \alpha\right|$$

Then I add and subtract $x^{(k+1)}$:

$$\left|\phi^{'}(\xi_k)\right| \left|x^{(k)} - \alpha + x^{(k+1)} - x^{(k+1)}\right|$$

Applying the triangular inequality we obtain:

$$\left|\phi^{'}(\xi_k)\right| \left|x^{(k)} - \alpha + x^{(k+1)} - x^{(k+1)}\right| \leq \left|\phi^{'}(\xi_k)\right| \left|x^{(k+1)} - \alpha\right| \left|x^{(k)} - x^{(k+1)}\right|$$

From where we can derive:

$$\left(1-\left|\phi^{'}(\xi_k)\right|\right)\left|x^{k+1}-\alpha\right| \leq \left|\phi^{'}(\xi_k)\right|\left|x^{(k+1)}-x^{(k)}\right|$$

That is:

$$\left|x^{(k+1)}-\alpha\right| \leq \frac{\phi^{'}(\xi_k)}{1-|\phi^{'}(\xi_k)|}\left|x^{(k+1)}-x^{(k)}\right|$$

■

Proof null derivative : We have that:

$$\left|x^{(k+1)}-\alpha\right| \leq \left|x^{(k+1)}-\alpha+x^{(k)}-x^{(k)}\right| \leq \left|x^{(k+1)}-x^{(k)}\right|\left|\phi x^{(k+1)}-\phi(\alpha)\right|$$

That is less or equal than:

$$\left|x^{(k+1)}-x^{(k)}\right|+\left|\phi^{'}(\xi_k)\right|\left|x^{(k+1)}-\alpha\right|$$

But for hypothesis we have that $\phi^{'}(\alpha)=0$ we obtain:

$$\left|x^{(k+1)}-\alpha\right| \leq \left|x^{(k+1)}-x^{(k)}\right|$$

■

2.8 Aitken method

We have that:

$$x^{(k+1)}-\alpha = \phi(x^{(k+1)})-\phi(\alpha) = \lambda_k\left(x^{(k+1)}-\alpha\right)$$

And we know that λ_k is the first derivative of ϕ in a certain point ξ. If I know λ_k I have that:

$$\alpha = \frac{\phi(x^{(k)})-\lambda_k x^k}{1-\lambda_k}$$

We also have that the quantity

$$A_k = \frac{\phi(\phi(x^{(k)}))-\phi(x^{(k)})}{\phi(x^{(k)})-x^{(k)}}$$

is a good approximation of λ_k, so I can replace λ_k with A_k. By substituting, we obtain the formula used in the Aitken method.

Algorithm 4 Algorithm for the Aitken method

```
1: for $k = 0, 1, \ldots, n$ do
2:     $x^{(k+1)} = x^{(k)} - \frac{\left[\phi(x^{(k)})-x^{(k)}\right]^2}{\phi(\phi(x^{(k)}))-2\phi(x^{(k)})+x^{(k)}}$
3:     if stopping criterion is satisfied then
4:         return $x^{(k+1)}$
5:     end if
6: end for
```

Theorem

Let $\phi(x) = x - f(x)$ and $f(\alpha)$ be a simple zero. Let $\phi^{(k)}$ converge to the first order to α, then ϕ_A converge to the second order.
If $\phi(x)$ converges with order p and $f(x)$ is still a simple zero, then ϕ_A converges with order $2p - 1$.

Sometimes ϕ_A converges even if ϕ does not.

2.9 Systems of nonlinear equations

2.9.1 Newton method

Let us consider a system of nonlinear equations of the form

$$\begin{cases} f_1(x_1, x_2, \dots, x_n) = 0 \\ f_2(x_1, x_2, \dots, x_n) = 0 \\ \vdots \\ f_n(x_1, x_2, \dots, x_n) = 0 \end{cases}$$

where $f_1, \dots, f_n$ are nonlinear functions. Setting $\boldsymbol{f} = (f_1, \dots, f_n)^T$ and $\boldsymbol{x} = (x_1, \dots, x_n)^T \in \mathbb{R}^d$, the system can be written in a compact way as:

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{0}$$

In order to extend Newton's method to the case of a system, we replace the first derivative of the scalar function f with the Jacobian matrix $J_{\boldsymbol{f}}$ of the vectorial function $\boldsymbol{f}$ whose components are:

$$(\boldsymbol{J}_{\boldsymbol{f}})_{ij} = \frac{\partial f_1}{\partial x_1} \qquad i, j = 1, \dots, n$$

The input of the algorithm is the initial guess $\boldsymbol{x}^{(0)}$.

Algorithm 5 Algorithm for the Newton method for systems

```
1: for $k = 0, 1, \dots, n$ do
2:     solve $\boldsymbol{J}_{\boldsymbol{f}}(\boldsymbol{x}^{(k)})\boldsymbol{\delta x}^{(k)} = -\boldsymbol{F}(\boldsymbol{x}^{(k)})$
3:     $\boldsymbol{x}^{(k+1)} \leftarrow \boldsymbol{x}^{(k)} + \boldsymbol{\delta x}^{(k)}$
4:     if $\left\|\boldsymbol{\delta x}^{(k+1)} \leq \epsilon\right\| \vee \left\|\boldsymbol{F}(\boldsymbol{x}^{(k+1)}) \leq \epsilon_r\right\|$ then
5:         return $x^{(k+1)}$
6:     end if
7: end for
```

The computation of the Jacobian matrix $\boldsymbol{J}_{\boldsymbol{f}}$ can be expensive if the number of equations d is relatively large. To simplify the computation of this matrix it is possible to use the approximation with finite difference, that is:

$$(\boldsymbol{J}_{\boldsymbol{f}})_{ij} \approx \frac{f_i(x_1^k, \dots, x_{j-1}^k, x_{j+h}^k, x_{j+1}^k, \dots, x_d^k) - f(\boldsymbol{x}_k)}{h}$$

But also this method is costly. There are other method used to approximate the Jacobian that are less expensive that are called quasi-Newton methods. The most important quasi-Newton method used for this purpose is called Broyden.

2.9.2 Broyden method

The secant method can be adapted to the solution of systems of nonlinear equations still featuring super-linear rate of convergence. The idea consists in replacing the Jacobian matrix $\boldsymbol{J_f}(\boldsymbol{x}^{(k)})$ (for $k \geq 0$) of Newton's method with suitable matrices B_k, recursively defined starting from a convenient matrix B_0, representing a suitable approximation of $\boldsymbol{J_f}(\boldsymbol{x}^{(0)})$.

The starting idea is to replace $\boldsymbol{I}_k$ with a matrix $\boldsymbol{B}_k$ that satisfies the equation:

$$\boldsymbol{B}_k\left(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\right) = \boldsymbol{F}(\boldsymbol{x}_k) - \boldsymbol{F}(\boldsymbol{x}_{k-1})(S)$$

Since there are infinite matrices that satisfies the previous equation, the idea of Broyden was to choose $\boldsymbol{B}_k$ so that it satisfies (S) and minimizes:

$$\min \|\boldsymbol{B}_k - \boldsymbol{B}_{k-1}\|_{\boldsymbol{F}}^2$$

The input of the algorithm is the initial guess $\boldsymbol{x}^{(0)} \in \mathbb{R}^n$ and a given $\boldsymbol{B}_0 \in \mathbb{R}^{n\times n}$ (we usually set $\boldsymbol{B}_0 = \boldsymbol{I}$).

Algorithm 6 Algorithm for the Broyden method for systems

```
1: for $k = 0, 1, \dots, n$ do
2:     solve $\boldsymbol{B}_k \boldsymbol{\delta x}^{(k)} = -\boldsymbol{f}(\boldsymbol{x}^{(k)})$
3:     $\boldsymbol{x}^{(k+1)} \leftarrow \boldsymbol{x}^{(k)} + \boldsymbol{\delta x}^{(k)}$
4:     $\boldsymbol{\delta f}^{(k)} \leftarrow \boldsymbol{f}(\boldsymbol{x}^{(k+1)}) + \boldsymbol{f}(\boldsymbol{x}^{(k)})$
5:     $B_{k+1} = B_k + \frac{\left(\boldsymbol{\delta f}^{(k)} - B_k \boldsymbol{\delta x}^{(k)}\right)}{\|\delta \boldsymbol{x}^{(k)}\|^2} \boldsymbol{\delta x}^{(k)^T}$
6:     if $\left\|\boldsymbol{\delta x}^{(k+1)} \leq \epsilon\right\| \vee \left\|\boldsymbol{F}(\boldsymbol{x}^{(k+1)}) \leq \epsilon_r\right\|$ then
7:         return $B_{k+1}$
8:     end if
9: end for
```

We do not require the sequence $\{\boldsymbol{B}_k\}$ to converge to the Jacobian matrix $\boldsymbol{J_f}(\alpha)$. Rather, it can be proved that:

$$\lim_{k\to\infty} \frac{\|\left(\boldsymbol{B}_k - \boldsymbol{J_f}(\boldsymbol{\alpha})\right)\left(\boldsymbol{x}^{(k)} - \boldsymbol{\alpha}\right)\|}{\|\boldsymbol{x}^{(k)} - \boldsymbol{\alpha}\|} = 0$$

This property guarantees that $\boldsymbol{B}_k$ is a convenient approximation of $\boldsymbol{J_f}(\alpha)$ along the direction error $\boldsymbol{x}_{(k)} - \boldsymbol{\alpha}$. The convergence rate of the Broyden method is $q \simeq 1.6$, so it is super-linear.

Since the Newton method for systems of nonlinear equation uses the inverse of the Jacobian to compute the solution, it is better to compute directly the inverse instead of the normal matrix. The matrix found with the Broyden formula (the one added to the $\boldsymbol{B}_k$ matrix) is always of rank one, so it is possible to find the inverse with the Sherman-Morrison formula:

$$\left(\boldsymbol{A} + \boldsymbol{u}\boldsymbol{w}^T\right) = \boldsymbol{A}^{-1} - \frac{\boldsymbol{A}\boldsymbol{u}\boldsymbol{w}^T\boldsymbol{A}^{-1}}{1 + \boldsymbol{w}^T\boldsymbol{A}\boldsymbol{u}}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is an invertible square matrix, and $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{R}^n$ are column vectors. Now it is possible to compute the approximation of the Jacobian with the Broyden method:

$$\boldsymbol{B}_{k+1}^{-1} = \boldsymbol{B}_k^{-1} + \frac{\boldsymbol{\delta x}^{(k)} - \boldsymbol{B}_k^{-1}\boldsymbol{\delta f}_k}{\boldsymbol{\delta x}^{(k)}\boldsymbol{B}_k^{-1}\boldsymbol{\delta f}_k}\left(\boldsymbol{\delta x}^{(k)}\right)\boldsymbol{B}_k^{-1}$$

2.9.3 Bad Broyden method

The bad version of the Broyden method compute the difference between two steps with the secant condition: $\boldsymbol{x}_k - \boldsymbol{x}_{k+1} = \boldsymbol{B}_k^{-1}\left[F(\boldsymbol{x}_k) - F(\boldsymbol{x}_{k-1})\right]$, and then construct the approximation of the Jacobian directly imposing the secant condition and minimizing:

$$\min \left\|\boldsymbol{B}_k^{-1} - \boldsymbol{B}_{k-1}^{-1}\right\|_{\boldsymbol{F}}^2$$

With these formulas, the Broyden method becomes:

$$\boldsymbol{B}_{k+1}^{-1} = \boldsymbol{B}_k^{-1} + \frac{\boldsymbol{\delta x}^{(k)} - \boldsymbol{B}_k^{-1}\boldsymbol{\delta f}^{(k)}}{\left\|\boldsymbol{\delta f}^{(k)}\right\|^2}\left[\boldsymbol{\delta f}^{(k)}\right]^T$$

CHAPTER 3

Linear systems

3.1 Introduction

The systems of nonlinear equations can be expressed as:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

where $\boldsymbol{A}$ is a non-singular square matrix of dimension $n \times n$ whose elements a_{ij} are either real or complex, while $\boldsymbol{x}$ and $\boldsymbol{b}$ are column vectors of dimension n: $\boldsymbol{x}$ represents the unknown solution while $\boldsymbol{b}$ is a given vector. Component-wise, it can be written as:

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

Definition

The matrix $\boldsymbol{A}$ is *non-singular* if and only if:

$$\boldsymbol{A}\boldsymbol{v} = \boldsymbol{0} \leftrightarrow \boldsymbol{v} = \boldsymbol{0}$$

In other words, the kernel of $\boldsymbol{A}$ contains only the vector $\boldsymbol{0}$ or, equivalently:

$$\dim\left(\ker\left(\boldsymbol{A}\right)\right) = \boldsymbol{0}$$

We also need that $\text{rank}(\boldsymbol{A}) = n$ or $\det(\boldsymbol{A}) \neq 0$.

The solution of this system can be computed with the Cramer rule, that is:

$$x_i = \frac{\det \boldsymbol{A}_i}{\det \boldsymbol{A}} \qquad i = 1, \ldots, n$$

where $\boldsymbol{A}_i$ is the matrix obtained from $\boldsymbol{A}$ by replacing the i-th column by $\boldsymbol{b}$. The time complexity of Cramer rule operations is of the order of $3(n+1)!$, that is too complex for most cases. The two alternative possibilities are:

- Direct methods, that yield the solution of the system in a finite number of steps
- Iterative methods, that requires (in principle) infinite number of steps.

3.2 Direct methods

3.2.1 LU factorization method

Let $\boldsymbol{A} \in \mathbb{R}^{n\times n}$. Assume that there exists two suitable matrices *boldsymbolL* and *boldsymbolU*, lower triangular and upper triangular, such that $\boldsymbol{A} = boldsymbolLU$. We call this an LU factorization of $\boldsymbol{A}$. If $\boldsymbol{A}$ is non-singular, so are both $\boldsymbol{L}$ and $\boldsymbol{U}$, and thus their diagonal elements are non-null. In such a case, solving $\boldsymbol{Ax} = \boldsymbol{b}$ leads to the solution of the two triangular systems:

$$\boldsymbol{Ly} = \boldsymbol{b} \qquad \boldsymbol{Ux} = \boldsymbol{y}$$

The first equation can be solved with the forwarding substitution algorithm with a complexity of $O(n^2)$, that is:

Algorithm 7 Forward substitution algorithm

```
$y_1 \leftarrow \frac{b_1}{l_{11}}$
$y_n \leftarrow \frac{1}{l_{11}}\left(b_i - \sum_{j=1}^{i-1} l_{ij}y_j\right) \qquad i = 2,\dots,n$
```

The second equation can be solved with the backward substitution algorithm with a complexity of $O(n^2)$, that is:

Algorithm 8 Backward substitution algorithm

```
$x_n \leftarrow \frac{y_n}{u_{nn}}$
$x_i \leftarrow \frac{1}{u_{ii}}\left(y_i - \sum_{j=i+1}^{n} u_{ij}x_j\right) \qquad i = n-1,\dots,1$
```

The elements of $\boldsymbol{L}$ and $\boldsymbol{U}$ satisfy the system of nonlinear equations.

$$\sum_{r=1}^{\min(i,j)} l_{ir}u_{rj} = a_{ij} \qquad i,j = 1,\dots,n$$

This system is under-determined since there are n^2 equations and $n^2 + n$ unknowns. Consequently, the LU factorization cannot be unique. By forcing the n diagonal elements of $\boldsymbol{L}$ to be equal to 1, the previous system turns into a determined one. This determined system can be solved by the Gauss algorithm.

Algorithm 9 Gauss algorithm

```
1: for $k = 1,\dots,n-1$ do
2:     for $i = k+1,\dots,n$ do
3:         $l_{ik} \leftarrow \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$
4:         for $j = k+1,\dots,n$ do
5:             $a_{ij}^{(k+1)} \leftarrow a_{ij}^{(k)} - l_{ik}a_{kj}^{(k)}$
6:         end for
7:     end for
8: end for
```

The elements of $a_{kk}^{(k)}$ must all be different from zero and are called pivot elements. At the end of this procedure the elements of the upper triangular matrix $\boldsymbol{U}$ are given by u_{ij}, whereas those of $\boldsymbol{L}$ are given by the coefficients l_{ij} generated by this algorithm. Determining the elements of $\boldsymbol{L}$ and $\boldsymbol{U}$ requires $\frac{2}{3}n^3$ operations.

Proposition

For a given matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$, its $\boldsymbol{LU}$ factorization exists and is unique if and only if the principal sub-matrices A_i of A of order $i = 1, \dots, n-1$ are non-singular.

The hypothesis of this proposition are fulfilled by these special classes:

- Strictly dominant matrices.
- Real symmetric and positive definite matrices.
- Complex definite positive matrices.

The solution of a linear system with the LU factorization technique has a complexity $O(\frac{2}{3}n^3)$.

Example : Given the following system:

$$\begin{cases} x_1 + 3x_2 = b_1 \\ 2x_1 + 2x_2 + 2x_3 = b_2 \\ 3x_1 + 6x_2 + 4x_3 = b_3 \end{cases}$$

Find the lower triangular and the upper triangular matrices using the Gauss algorithm. The initial matrix is:

$$\boldsymbol{A}^{(0)} = \begin{bmatrix} 1 & 0 & 3 \\ 2 & 2 & 2 \\ 3 & 6 & 4 \end{bmatrix}$$

For the first iteration we compute the values for the lower matrices using the first column:

$$l_{21} = \frac{a_{21}}{a_{11}} = \frac{2}{1} = 2$$

$$l_{31} = \frac{a_{31}}{a_{11}} = \frac{3}{1} = 3$$

So, it is now possible to compute the new second and third rows with these formulas:

$$\boldsymbol{r}_2 \leftarrow \boldsymbol{r}_2 - l_{21}\boldsymbol{r}_1$$

$$\boldsymbol{r}_3 \leftarrow \boldsymbol{r}_3 - l_{31}\boldsymbol{r}_1$$

The new matrix now becomes:

$$\boldsymbol{A}^{(1)} = \begin{bmatrix} 1 & 0 & 3 \\ 2-2 & 2-0 & 2-6 \\ 3-3 & 6-0 & 4-9 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 2 & -4 \\ 0 & 6 & -5 \end{bmatrix}$$

For the first iteration we compute the values for the lower matrices using the first column (always from $\boldsymbol{A}^{(0)}$):

$$l_{32} = \frac{a_{32}}{a_{22}} = \frac{6}{2} = 3$$

So, it is now possible to compute the new third row with this formula:

$$\boldsymbol{r}_3 \leftarrow \boldsymbol{r}_3 - l_{32}\boldsymbol{r}_2$$

The new matrix now becomes:

$$\boldsymbol{A}^{(2)} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 2 & -4 \\ 0-0 & 6-6 & -5+12 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 2 & -4 \\ 0 & 0 & 7 \end{bmatrix}$$

Now we have found that the lower triangular matrix is:

$$\boldsymbol{L} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 3 & 1 \end{bmatrix}$$

and that the upper triangular matrix is:

$$\boldsymbol{U} = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 2 & -4 \\ 0 & 0 & 7 \end{bmatrix}$$

3.2.2 Cholesky factorization

If $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is symmetric positive definite, it is moreover possible to construct a special factorization:

$$\boldsymbol{A} = \boldsymbol{R}^T\boldsymbol{R}$$

where $\boldsymbol{R}$ is an upper triangular matrix with positive diagonal elements. This is the so-called Cholesky factorization and requires about $\frac{1}{3}n^3$ operations. Further, let's note that, due to symmetry, only the upper part of $\boldsymbol{A}$ is stored, and $\boldsymbol{R}$ can be stored in the same area.

The elements of $\boldsymbol{R}$ can be computed with the following algorithm:

Algorithm 10 Cholesky factorization algorithm

```
$r_{jj} \leftarrow \sqrt{a_{jj} - \sum_{k=1}^{j-1} r_{kj}^2}$
$r_{ij} \leftarrow \frac{1}{r_{ii}}\left(a_{ij} - \sum_{k=1}^{i-1} r_{ki}r_{kj}\right) \qquad i = 1, \dots, j-1$
```

In some cases it is possible to modify the matrix $\boldsymbol{A}$ to apply this factorization by using an appropriate matrix $\boldsymbol{P}$:

$$\boldsymbol{P}^T\boldsymbol{A}\boldsymbol{P} = \boldsymbol{R}^T\boldsymbol{R}$$

3.2.3 The pivoting technique

The pivoting technique allows to achieve the LU factorization for every non-singular matrix. A suitable row permutation of the original matrix $\boldsymbol{A}$ would make the entire factorization procedure feasible even if the hypothesis of the proposition are not satisfied, under the sole condition that $\det(\boldsymbol{A}) \neq 0$. The decision on which row to permute can be made at every step k at which a null diagonal element $a_{kk}^{(k)}$ is generated.

Since a row permutation entails changing the pivot element, this technique is given the name of pivoting by row. The factorization generated in this way returns the original matrix up to a row permutation. Precisely we obtain:

$$\boldsymbol{PA} = \boldsymbol{LU}$$

where $\boldsymbol{P}$ is a suitable permutation matrix that is initially set equal to the identity matrix, then whenever in the course of the procedure two rows of $\boldsymbol{A}$ are permuted, the same permutation must be performed on the corresponding rows of $\boldsymbol{P}$. At last, we should solve the following systems:

$$\boldsymbol{Ly} = \boldsymbol{Pb} \quad \boldsymbol{Ux} = \boldsymbol{y}$$

Other than null pivots, also small elements $a_{kk}^{(k)}$ are troublesome. In this case, possible round-off errors affecting the coefficients $a_{kj}^{(k)}$ will be severely amplified. It is therefore recommended to carry out the pivoting at every step of the factorization procedure, by searching among all virtual pivot elements $a_{ik}^{(k)}$, with $i = k, \dots, n$, the one with maximum modulus.

Algorithm 11 Gauss algorithm with pivoting

```
1: for $k = 1, \dots, n-1$ do
2:     find $\overline{r}$ such that $\left|a_{\overline{r}k}^{(k)}\right| = \max_{r=k,\dots,n} \left|a_{rk}^{(k)}\right|$
3:     exchange row $k$ with row $\overline{r}$ in both $\boldsymbol{A}$ and $\boldsymbol{P}$
4:     for $i = k+1, \dots, n$ do
5:         $l_{ik} \leftarrow \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$
6:         for $j = k+1, \dots, n$ do
7:             $a_{ij}^{(k+1)} \leftarrow a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}$
8:         end for
9:     end for
10: end for
```

3.2.4 Linear system solution accuracy

Due to round-off errors, when a linear system $\boldsymbol{Ax} = \boldsymbol{b}$ is solved numerically, one is indeed looking for the exact solution $\widetilde{\boldsymbol{x}}$ of a perturbed system:

$$(\boldsymbol{A} + \boldsymbol{\delta A})\,\widetilde{\boldsymbol{x}} = \boldsymbol{b} + \boldsymbol{\delta b}$$

where $\boldsymbol{\delta A}$ and $\boldsymbol{\delta b}$ are respectively a matrix and a vector which depend on the specific numerical method which is being used. We start by considering the case $\boldsymbol{\delta A} = \boldsymbol{0}$ and $\boldsymbol{\delta b} \neq \boldsymbol{0}$ which is simpler than the most general case. Let $\boldsymbol{\delta A} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{\delta b} \in \mathbb{R}^n$ with $\|\boldsymbol{\delta A}\|_2 \leq \varepsilon$ and $\|\boldsymbol{\delta b}\|_2 \leq \varepsilon$, with $\varepsilon > 0$. By comparing the previous equation with the general formula for system we obtain:

$$\boldsymbol{A}\,(\boldsymbol{x} - \widetilde{\boldsymbol{x}}) = \boldsymbol{\delta b} \Rightarrow \|\boldsymbol{x} - \widetilde{\boldsymbol{x}}\|_2 = \|\boldsymbol{A}^{-1} - \boldsymbol{\delta b}\|_2 \leq \|\boldsymbol{A}^{-1}\|_2 \|\boldsymbol{\delta b}\|_2$$

And also that:

$$\|\boldsymbol{Ax}\|_2 = \|\boldsymbol{b}\|_2 \Rightarrow \|\boldsymbol{A}\|_2 \|\boldsymbol{x}\|_2 \geq \|\boldsymbol{Ax}\|_2 = \|\boldsymbol{b}\|_2$$

By dividing the two equations we obtain:

$$\frac{\|\widetilde{\boldsymbol{x}} - \boldsymbol{x}\|_2}{\|\boldsymbol{x}\|_2} \leq \frac{\|\boldsymbol{A}\|_2 \left\|\boldsymbol{A}^{-1}\right\|_2 \|\boldsymbol{\delta b}\|_2}{\|\boldsymbol{b}\|_2} = k_2(A) \frac{\|\boldsymbol{\delta b}\|_2}{\|\boldsymbol{b}\|_2}$$

Since the matrix is symmetric positive definite we have that:

$$k_2(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)}$$

If the value of the constant $k_2(A)$ is too large (order of 1000) we have to correct the result by using the residual correction algorithm.

Algorithm 12 Residual correction algorithm

```
1: solve $\boldsymbol{Ax} = \boldsymbol{b}$
2: $\boldsymbol{r} \leftarrow \boldsymbol{b} - \boldsymbol{Ax}$
3: solve $\boldsymbol{Ad} = \boldsymbol{r}$
4: $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{d}$
```

3.3 Direct methods

A general technique to devise an iterative method is based on a splitting of the matrix $\boldsymbol{A}$ in:

$$\boldsymbol{A} = \boldsymbol{P} - (\boldsymbol{P} - \boldsymbol{A})$$

where $\boldsymbol{P}$ is a suitable non-singular matrix called preconditioner of $\boldsymbol{A}$.

3.3.1 Jacobi method

If the diagonal entries of $\boldsymbol{A}$ are nonzero, we can set $\boldsymbol{P} = \boldsymbol{D}$, where $\boldsymbol{D}$ is the diagonal matrix containing the diagonal entries of $\boldsymbol{A}$. So we have that:

$$\boldsymbol{D}\boldsymbol{x}^{(k+1)} = \boldsymbol{b} - (\boldsymbol{A} - \boldsymbol{D})\,\boldsymbol{x}^{(k)} \qquad k \geq 0$$

or component-wise:

Algorithm 13 Jabobi method algorithm

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

The stopping criterions of the Jacobi method are:

$$\frac{\left\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\right\|_2}{\left\|\boldsymbol{x}^{(k)}\right\|_2} \leq \varepsilon \quad \vee \quad \frac{\left\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(k+1)}\right\|_2}{\left\|\boldsymbol{b}\right\|_2} \leq \varepsilon_r$$

Proposition

If the matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is strictly diagonally dominant by row, then the Jacobi method converges.

3.3.2 Gauss-Seidel method

The Gauss-Seidel method computes the result iteratively with this formula:

$$(\boldsymbol{D}-\boldsymbol{E})\,\boldsymbol{x}^{(k+1)}=\boldsymbol{b}+\boldsymbol{F}\boldsymbol{x}^{(k)}$$

where $-\boldsymbol{F}$ is the upper triangular part of the matrix, diagonal excluded, $-\boldsymbol{E}$ is the lower triangular part of the matrix, diagonal excluded, and $\boldsymbol{D}$ is the diagonal matrix with the diagonal entries of the matrix $\boldsymbol{A}$.

Algorithm 14 Gauss-Seidel algorithm

```
1: $x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right) \qquad i=1,\dots,n$
```

Proposition

Let $\boldsymbol{A}\in\mathbb{R}^{n\times n}$ be a tridiagonal non-singular matrix whose diagonal elements are all non-null. Then the Jacobi method and the Gauss-Seidel method are either both divergent or both convergent. In the latter case, the Gauss-Seidel method is faster than Jacobi's.

The case in which $\boldsymbol{P}=\boldsymbol{D}-\boldsymbol{E}$ is called Gauss-Seidel method, while the case in which $\boldsymbol{P}=\boldsymbol{D}-\boldsymbol{F}$ is called inverted Gauss-Seidel method.

3.3.3 Richardson method

The general form of the Richardson method is:

$$\boldsymbol{P}(\boldsymbol{x}^{(k+1)}-\boldsymbol{x}^{(k)})=\alpha_k\boldsymbol{r}^{(k)} \qquad k\geq 0$$

We call stationary the case when $\alpha_k=\alpha$ for any $k\geq 0$, dynamic the case in which α_k may change along the iterations. The case $\boldsymbol{P}\neq\boldsymbol{I}$ is called preconditioned Richardson method. In this case the formula become:

$$\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}+\alpha_k\boldsymbol{r}^{(k)}$$

Proposition

Let $\boldsymbol{A}\in\mathbb{R}^{n\times n}$. For any non-singular matrix $\boldsymbol{P}\in\mathbb{R}^{n\times n}$ the stationary Richardson method converges if and only if:

$$|\lambda_i|<\frac{2}{\alpha}\mathrm{Re}\,[\lambda_i] \qquad \forall i=1,\dots,n$$

where λ_i are the eigenvalues of $(\boldsymbol{P}^{-1}\boldsymbol{A})$. If the latter are all real, it converges if and only if:

$$0<\alpha\lambda_i<2 \qquad \forall i=1,\dots,n$$

If both $\boldsymbol{A}$ and $\boldsymbol{P}$ are symmetric positive definite matrices, the stationary Richardson method converges for any possible choice of $\boldsymbol{x}^{(0)}$ if and only if:

$$0<\alpha<\frac{2}{\lambda_{max}}$$

where $\lambda_{max}>0$ is the maximum eigenvalue of $(\boldsymbol{P}^{-1}\boldsymbol{A})$. Moreover, the spectral radius $\rho(B_\alpha)$ of the iteration matrix $B_\alpha=I-\alpha P^{-1}A$ is minimized for $\alpha=\alpha_{opt}$, where:

$$\alpha_{opt}=\frac{2}{\lambda_{min}+\lambda_{max}}$$

λ_{min} being the smallest eigenvalue of $P^{-1}A$. If I choose α_{opt}, then:

$$\left\|e^{(k+1)}\right\|_A = \left\|x^{(k+1)} - x^{(k)}\right\|_A \leq \frac{k_2(P^{-1}A) - 1}{k_2(P^{-1}A) + 1} \left\|e^{(k)}\right\|_A$$

CHAPTER 4

Definitions

4.1 Matrix

Definition

A matrix is *diagonally dominant by row* if:

$$|a_{ii}| \geq \sum_{j=1, j\neq i} |a_{ij}| \qquad i = 1, \dots, n$$

A matrix is *strictly diagonally dominant by row* if:

$$|a_{ii}| > \sum_{j=1, j\neq i} |a_{ij}| \qquad i = 1, \dots, n$$

A matrix is *diagonally dominant by column* if:

$$|a_{ii}| \geq \sum_{j=1, j\neq i} |a_{ji}| \qquad i = 1, \dots, n$$

A matrix is *strictly diagonally dominant by column* if:

$$|a_{ii}| > \sum_{j=1, j\neq i} |a_{ji}| \qquad i = 1, \dots, n$$

A matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is *positive definite* if

$$\forall \boldsymbol{x} \in \mathbb{R}^n, \qquad \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} > 0$$

A matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is *semi positive definite* if

$$\forall \boldsymbol{x} \in \mathbb{R}^n, \qquad \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} \geq 0$$

4.2 Norm of vectors

Definition

The *p-norm* of a vector $\boldsymbol{x}$ is defined as:

$$\|\boldsymbol{x}\|_p = \sqrt[p]{\sum |x_i|^p}$$

The properties for the vector's norm are:

1. $\|\boldsymbol{x}+\boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$
2. $\|\alpha\boldsymbol{x}\| \leq |\alpha| \, \|\boldsymbol{x}\|$

The most used vector's norms are:

- $\|\boldsymbol{x}\|_1 = \sum |x_i|$
- $\|\boldsymbol{x}\|_2 = \|\boldsymbol{x}\| = \sqrt{\sum |x_i|^2}$
- $\|\boldsymbol{x}\|_\infty = \max(x_i)$

4.3 Norm of matrices

Definition

The *p-norm* of a matrix $\boldsymbol{A}$ is defined as:

$$\|\boldsymbol{A}\|_p = \max_{x \neq 0} \left(\frac{\|\boldsymbol{A}_x\|_p}{\|x\|_p} \right) = \max_{\|x\|_p = 1} \left(\|\boldsymbol{A}_x\|_p \right)$$

The most used matrix's norms are:

- $\|\boldsymbol{A}\|_1 = \max_i \sum_{i=1}^n |a_{ij}|$
- $\|\boldsymbol{A}\|_2 = \|\boldsymbol{A}\| = \sqrt{\sum \lambda_{max}(\boldsymbol{A}^T\boldsymbol{A})}$
- $\|\boldsymbol{y}\|_A = \sqrt{y^T A y}$
- $\|\boldsymbol{A}\|_\infty = \max_j \sum_{j=1}^n |a_{ij}|$

Proposition

If the matrix $\boldsymbol{A}$ is symmetric positive definite we have that:

$$\|\boldsymbol{A}\|_2 = \lambda_{max}$$