Numerical Analysis 1
Lecture notes

Dmitriy Leykekhman

0 Introduction

This lecture notes are designed for the MATH 5510, which is the first graduate course in numerical analysis at University of Connecticut. Here I present the material which I consider important for students to see in their first numerical analysis course. The material and the style of these lecture notes are strongly influenced by the lecture notes of Prof. Matthias Heinkenschloss for CAMM 353 at Rice University. There are many other nice lecture notes that one can find freely online. Let me just mention four volumes **Numerical Analysis course (in German)** by Rolf Rannacher and the **lecture notes** by Doron Levy . There are plenty of other sources on the internet.

There are plenty of misconceptions among mathematicians what really numerical analysis is. In my opinion one of the best definitions was given by N. Trefethen in his essay *"What is numerical analysis?"*

Definition 0.0. Numerical analysis is the study of algorithms for the problem of continuous mathematics.

We strongly encourage to read this essay whoever is interested in the subject, it is only 5 pages long.

This lecture notes start with interpolation, which is not orthodox, but in my opinion it is an interesting topic that captures students attention and introduces important ideas, challenges and motivations for the other topics in numerical analysis.

The required background is rather light, good understanding of linear algebra and calculus would be sufficient. The students also encouraged to code as many algorithms as possible appearing in the notes. From time to time I adopt Matlab notation and syntax for columns or rows of matrices, loops and etc.

Dmitriy Leykekhman
Department of Mathematics, University of Connecticut, Storrs, CT 06269, USA. e-mail: dmitriy.leykekhman@uconn.edu

Contents

1 Floating point arithmetics

A very good introduction to floating point arithmetics is a book by Michael Overton [1].

In everyday life we use decimal representation of numbers. For example

$$1234.567$$

for us means

$$1*10^4+2*10^3+3*10^2+4*10^0+5*10^{-1}+6*10^{-2}+7*10^{-3}.$$

More generally

$$d_j\dots d_1d_0.d_{-1}\dots d_{-i}\dots$$

represents

$$\cdots d_j*10^j+\cdots+d_1*10^1+d_0*10^0+d_{-1}*10^{-1}+\cdots+d_{-i}*10^{-i}+\cdots.$$

Let $\beta\geq 2$ be an integer. For every $x\in$ there exist integers e and $d_i\in\{0,\dots,\beta-1\}$, $i=0,1,\dots$, such that

$$x=sign(x)\left(\sum_{i=0}^{\infty}d_i\beta^{-i}\right)\beta^e. \tag{1.1}$$

The representation is unique if one requires that $d_0>0$ when $x\neq 0$.

Example 1.1.

$$\begin{aligned}\frac{11}{2}&=5*10^0+5*10^{-1}=(5.5)_{10},\\ \frac{11}{2}&=1*2^2+0*2^1+1*2^0+1*2^{-1}\\ &=(1*2^0+0*2^{-1}+1*2^{-2}+1*2^{-3})*2^2=(1.011)_2*2^2.\end{aligned}$$

Of course every integer has finite representation, but very often even simple rational numbers have infinite representations

Example 1.2.

$$\begin{aligned}\frac{1}{3}&=(0.33333\dots)_{10}=(3.33333\dots)_{10}*10^1,\\ \frac{1}{3}&=(0.0101\dots)_2=(1.0101)_2*2^2.\end{aligned}$$

In a computer only a finite subset of all real numbers can be represented. These are the so–called **floating point numbers** and they are of the form

$$\bar{x} = (-1)^s \left(\sum_{i=0}^{m-1} d_i \beta^{-i} \right) \beta^e$$

with $d_i \in \{0, \ldots, \beta - 1\}$, $i = 0, 1, \ldots, m-1$, and $e \in \{e_{\min}, \ldots, e_{\max}\}$.

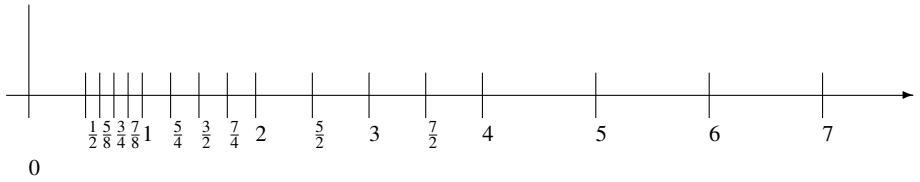
- β is called the **base**,
- $\sum_{i=0}^{m-1} d_i \beta^{-i}$ is the **significant** or **mantissa**, m is the **mantissa length**,
- e is the **exponent**, and $\{e_{\min}, \ldots, e_{\max}\}$ is the **exponent range**.
- If $\beta = 2$, then we say the floating point number system is a **binary system**. In this case the d_i's are called **bits**.
- If $\beta = 10$, then we say the floating point number system is a **decimal system**. In this case the d_i's are called **digits**.
- A floating point number $\bar{x} \neq 0$ is said to be **normalized** if $d_0 > 0$.

Example 1.3. Consider the floating point number system $\beta = 2, m = 3, e_{\min} = -1, e_{\max} = 2$.

The normalized floating point numbers $\bar{x} \neq 0$ are of the form

$$\bar{x} = \pm 1.d_1 d_2 \times 2^e$$

since the normalization condition implies that $d_0 \in \{1, \ldots, \beta - 1\} = \{1\}$. For the exponent we have choices $e = -1, 0, 1, 2$. Below is the we plot all the numbers in this example. Notice that spacing between numbers is increasing as we move away from 0.



Consider the floating point number system

$$\bar{x} = (-1)^s \left(\sum_{i=0}^{m-1} d_i \beta^{-i} \right) \beta^e$$

with $d_i \in \{0, \ldots, \beta - 1\}$, $i = 0, 1, \ldots, m-1$, and $e \in \{e_{\min}, \ldots, e_{\max}\}$.

- The mantissa satisfies
$$\sum_{i=0}^{m-1} d_i \beta^{-i} \leq \sum_{i=0}^{m-1} (\beta - 1)\beta^{-i} = \beta(1 - \beta^{-m}) < \beta.$$
- The mantissa of a normalized floating point number is always ≥ 1.
- The largest floating point number is
$$\bar{x}_{\max} = \left(\sum_{i=0}^{m-1} (\beta - 1)\beta^{-i} \right) \beta^{e_{\max}} = (1 - \beta^{-m})\beta^{e_{\max}+1}.$$
- The smallest positive normalized floating pt. number is $\bar{x}_{\min} = \beta^{e_{\min}}$.
- The distance between 1 and the next largest floating pt. number is β^{1-m}.

- Half this number, $\varepsilon_{\mathsf{mach}} = \frac{1}{2}\beta^{1-m}$, is called **machine precision** or **unit roundoff**.
- The spacing between the floating pt. numbers in $[1,\beta]$ is $\beta^{-(m-1)}$.
- The spacing between the floating pt. numbers in $[\beta^e, \beta\beta^e]$ is $\beta^{-(m-1)}\beta^e$.

Almost all modern computer implements the IEEE binary ($\beta = 2$) floating point standard. IEEE single precision floating point numbers are stored in 32 bits. IEEE double precision floating point numbers are stored in 64 bits.

Common Name	(Approximate) Equivalent Value	
	Single Precision	Double Precision
Unit roundoff	$2^{-24} \approx 6.e-8$	$2^{-53} \approx 1.1e-16$
Maximum normal number	$3.4e+38$	$1.7e+308$
Minimum positive normal number	$1.2e-38$	$2.3e-308$
Maximum subnormal number	$1.1e-38$	$2.2e-308$
Minimum positive subnormal number	$1.5e-45$	$5.0e-324$

1.1 Rounding

Given a real number x we define

$$\mathrm{fl}(x) = \text{ normalized floating point number closest to } x.$$

A floating point number $\bar{x}$ closest to x is obtained by rounding. If

$$x = sign(x)\left(\sum_{i=0}^{\infty} d_i\beta^{-i}\right)\beta^e,$$

then

$$\mathrm{fl}(x) = \begin{cases} sign(x)\left(\sum_{i=0}^{m-1} d_i\beta^{-i}\right)\beta^e, & \text{if } d_m < \frac{1}{2}\beta, \\ sign(x)\left(\sum_{i=0}^{m-1} d_i\beta^{-i} + \beta^{-(m-1)}\right)\beta^e, & \text{if } d_m \geq \frac{1}{2}\beta. \end{cases}$$

Example 1.4. Let $\beta = 10$, $m = 3$. Then

$$\begin{aligned} \mathrm{fl}(1.234 * 10^{-1}) &= 1.23 * 10^{-1}, \\ \mathrm{fl}(1.235 * 10^{-1}) &= 1.24 * 10^{-1}, \\ \mathrm{fl}(1.295 * 10^{-1}) &= 1.30 * 10^{-1}. \end{aligned}$$

Note, there may be two floating point numbers closest to x. $\mathrm{fl}(x)$ picks one of them. For example, let $\beta = 10$, $m = 3$. Then $1.235 - 1.24 = 0.005$, but also $1.235 - 1.23 = 0.005$.

Theorem 1.1. *If x is a number within the range of floating point numbers and $|x| \in [\beta^e, \beta^{e+1})$, then the* **absolute error** *between x and the floating point number $\mathrm{fl}(x)$ closest to x is given by*

$$|\mathrm{fl}(x) - x| \leq \frac{1}{2}\beta^{e(1-m)}$$

and, provided $x \neq 0$, the **relative error** *is given by*

$$\frac{|\operatorname{fl}(x)-x|}{|x|} \leq \frac{1}{2}\beta^{1-m}. \tag{1.2}$$

In other words $\operatorname{fl}(x)$ *is a floating point number closest to* $x = \left(\sum_{i=0}^{\infty} d_i \beta^{-i}\right)\beta^e$ *with* $d_0 > 0$.

Definition 1.1. The number

$$\varepsilon_{mach} := \frac{1}{2}\beta^{1-m}$$

is called **machine precision** or **unit roundoff**.

Proof. If $x = 0$, then $\operatorname{fl}(x) = x$ and the assertion follows immediately.

Consider $x > 0$. (The case $x < 0$ can be treated in the same manner.) Recall that the spacing between the floating point numbers

$$\bar{x} = \left(\sum_{i=0}^{m-1} d_i \beta^{-i}\right)\beta^e \in [\beta^e, \beta^{e+1})$$

is $\beta^{-(m-1)}\beta^e$. Hence if $x \in [\beta^e, \beta^{e+1})$, then the floating point number $\bar{x}$ closest to x satisfies $|\bar{x}-x| \leq \frac{1}{2}\beta^{-(m-1)}\beta^e$. Since $x \geq \beta^e$,

$$\frac{|\bar{x}-x|}{|x|} \leq \frac{1}{2}\beta^{-(m-1)}.$$

Example 1.5. Let $\beta = 10$, $m = 3$, thus $\varepsilon_{\mathsf{mach}} = 5 * 10^{-3}$.

$$\begin{aligned}
&|\operatorname{fl}(1.234 * 10^{-1}) - 1.234 * 10^{-1}| = 0.0004,\\
&\frac{|\operatorname{fl}(1.234 * 10^{-1}) - 1.234 * 10^{-1}|}{1.234 * 10^{-1}} = \frac{0.0004}{1.234 * 10^{-1}} \approx 3.2 * 10^{-3},\\
&|\operatorname{fl}(1.295 * 10^{-1}) - 1.295 * 10^{-1}| = 0.0005,\\
&\frac{|\operatorname{fl}(1.295 * 10^{-1}) - 1.295 * 10^{-1}|}{1.295 * 10^{-1}} = \frac{0.0005}{1.295 * 10^{-1}} \approx 3.9 * 10^{-3}.
\end{aligned}$$

1.2 Floating Point Arithmetic

Let $\square$ represent one of the elementary operations $+, -, *, /$. If $\bar{x}$ and $\bar{y}$ are floating point numbers, then $\bar{x}\square\bar{y}$ may not be a floating point number, for example: $\beta = 10$, $m = 4$: $1.234 + 2.751 * 10^{-1} = 1.5091$. What is the computed value for $\bar{x}\square\bar{y}$? In IEEE floating point arithmetic the result of the computation $\bar{x}\square\bar{y}$ is equal to the floating point number that is nearest to the exact result $\bar{x}\square\bar{y}$. Therefore we use $\operatorname{fl}(\bar{x}\square\bar{y})$ to denote the result of the computation $\bar{x}\square\bar{y}$ Model for the computation of $\bar{x}\square\bar{y}$, where $\square$ is one of the elementary operations $+, -, *, /$.

1. Given *floating point* numbers $\bar{x}$ and $\bar{y}$.
2. Compute $\bar{x}\square\bar{y}$ exactly.
3. Round the exact result $\bar{x}\square\bar{y}$ to the nearest floating point number and normalize the result.

In the above example: $1.234 + 2.751 * 10^{-1} = 1.5091$. Comp. result: 1.509 The actual implementation of the elementary operations is more sophisticated [1].

Given two numbers $\bar{x}, \bar{y}$ in *floating point format*, the computed result satisfies

$$\frac{|\,\mathrm{fl}(\bar{x}\Box\bar{y}) - (\bar{x}\Box\bar{y})|}{\bar{x}\Box\bar{y}} \leq \varepsilon_{\mathsf{mach}}.$$

Example 1.6. Consider the floating point system $\beta = 10$ and $m = 4$.

i. $\bar{x} = 2.552 * 10^3$ and $\bar{y} = 2.551 * 10^3$.
$\bar{x} - \bar{y} = 0.001 * 10^3 = 1.000 * 10^0$. In this case $\bar{x} - \bar{y}$ is a floating point number and nothing needs to done; no error occurs in the subtraction of $\bar{x}$, $\bar{y}$.

ii. $\bar{x} = 2.552 * 10^3$ and $\bar{y} = 2.551 * 10^2$.
$\bar{x} - \bar{y} = 2.2969 * 10^3$. This is not a floating point number. The floating point number nearest to $\bar{x} - \bar{y}$ is $\mathrm{fl}(\bar{x} - \bar{y}) = 2.297 * 10^3$.

$$\frac{|\,\mathrm{fl}(\bar{x} - \bar{y}) - (\bar{x} - \bar{y})|}{|\bar{x} - \bar{y}|} = \frac{|2.297 * 10^3 - 2.2969 * 10^3|}{2.2969 * 10^3} \approx 4.4 * 10^{-5} < \varepsilon_{\mathsf{mach}} = 5 * 10^{-4}.$$

For the previous result on the error between $\bar{x}\Box\bar{y}$ and the computed $\mathrm{fl}(\bar{x}\Box\bar{y})$ only holds if $\bar{x}, \bar{y}$ in **floating point format**. What happens when we operate with numbers that are **not in floating point format**?

Example 1.7. Consider the floating point system $\beta = 10$ and $m = 4$.
Subtract the numbers $x = 2.5515052 * 10^3$ and $y = 2.5514911 * 10^3$.

1. Compute the floating point numbers $\bar{x}$ and $\bar{y}$ nearest to x and y, respectively: $\bar{x} = 2.552 * 10^3$ and $\bar{y} = 2.551 * 10^3$.
2. Compute $\bar{x} - \bar{y}$ exactly: $\bar{x} - \bar{y} = 0.001 * 10^3$.
3. Round the exact result $\bar{x} - \bar{y}$ to the nearest floating point number: $\mathrm{fl}(0.001 * 10^3) = 0.001 * 10^3$. Normalize the number: $\mathrm{fl}(0.001 * 10^3) = 1.000$. The last digits are filled with (spurious) zeros.

The exact result is $2.5515052 * 10^3 - 2.5514911 * 10^3 = 1.410 * 10^{-2}$. The relative error between exact and computed solution is

$$\frac{|1.000 - 1.410 * 10^{-2}|}{1.410 * 10^{-2}} \approx 70 \gg \varepsilon_{\mathsf{mach}} = 5 * 10^{-4}.$$

Note that this large error is not due the computation of $\mathrm{fl}(\bar{x} - \bar{y})$. The large error is caused by the rounding of x and y at the beginning.

To analyze the analyze the error incurred by the subtraction of two numbers, the following representation is useful: For every $x \in$, there exists ε with $|\varepsilon| \leq \varepsilon_{\mathsf{mach}}$ such that

$$\mathrm{fl}(x) = x(1 + \varepsilon).$$

Note that if $x \neq 0$, then the previous identity is satisfied for $\varepsilon := (\mathrm{fl}(x) - x)/x$. The bound $|\varepsilon| \leq \varepsilon_{\mathsf{mach}}$ follows from (1.2).

For $x, y \in$ we have $\varepsilon_1, \varepsilon_2$ with $|\varepsilon_1|, |\varepsilon_2| \leq \varepsilon_{\mathsf{mach}}$ such that

$$\mathrm{fl}(x) = x(1 + \varepsilon_1), \quad \mathrm{fl}(y) = y(1 + \varepsilon_2).$$

Moreover $\mathrm{fl}(\mathrm{fl}(x) - \mathrm{fl}(y)) = (\mathrm{fl}(x) - \mathrm{fl}(y))(1 + \varepsilon_3)$, with $|\varepsilon_3| \leq \varepsilon_{\mathsf{mach}}$.

Thus,

$$\begin{aligned} \mathrm{fl}(\mathrm{fl}(x) - \mathrm{fl}(y)) &= (\mathrm{fl}(x) - \mathrm{fl}(y))(1 + \varepsilon_3) = [x(1 + \varepsilon_1) - y(1 + \varepsilon_2)](1 + \varepsilon_3) \\ &= (x - y)(1 + \varepsilon_3) + (x\varepsilon_1 - y\varepsilon_2)(1 + \varepsilon_3) \end{aligned}$$

and, if $x - y \neq 0$, then the relative error is given by

$$\frac{|\,\mathrm{fl}(\mathrm{fl}(x)-\mathrm{fl}(y))-(x-y)|}{|x-y|} = \left|\varepsilon_3 + \frac{x\varepsilon_1 - y\varepsilon_2}{x-y}(1+\varepsilon_3)\right| \tag{1.3}$$

If $\varepsilon_1\varepsilon_2 \neq 0$ and $x-y$ is small, the quantity on the rhs could be $\gg \varepsilon_{\mathsf{mach}}$.

Similar analysis can be carried out for $+,-,*,/$. Catastrophic cancelation can only occur with $+,-$. Catastrophic cancelation can only occur if one subtracts two numbers which are not both in floating point format and which have the same sign and are of approximately the same size, see (1.3), or if one adds two numbers which are not both in floating point format, which have opposite sign and their absolute values of approximately the same size.

Example 1.8. The roots of the quadratic equation

$$ax^2+bx+c=0$$

are given by

$$x_\pm = \left(-b \pm \sqrt{b^2-4ac}\right)/(2a).$$

When $a = 5*10^{-4}, b = 100$, and $c = 5*10^{-3}$ the computed (using single precision Fortran) first root is

$$x_+ = 0.$$

Cannot be exact, since $x=0$ is a solution of the quadratic equation if and only if $c=0$. Since $\mathrm{fl}(b^2-4ac) = \mathrm{fl}(b^2)$ for the data given above, we suffer from catastrophic cancellation.

A remedy is the following reformulation of the formula for x_+:

$$\frac{-b+\sqrt{b^2-4ac}}{2a} = \frac{1}{2a}\frac{\left(-b+\sqrt{b^2-4ac}\right)\left(-b-\sqrt{b^2-4ac}\right)}{-b-\sqrt{b^2-4ac}} = \frac{2c}{-b-\sqrt{b^2-4ac}}$$

Here the subtraction of two almost equal numbers is avoided and the computation using this formula gives $x_+ = -0.5E-04$.

A 'stable' (see later for a description of stability) formula for both roots

$$x_1 = \left(-b-\mathrm{sign}(b)\sqrt{b^2-4ac}\right)/(2a), \quad x_2 = c/(ax_1).$$

---

2 Interpolation

We start this section with general interpolation problem.

2.1 Interpolation problem:

Let $\Phi(x;a_0,\ldots,a_n)$ be a family of functions of variable x, which can be real or complex. Given $n+1$ pairs (x_i,f_i), $i=0,1,\ldots,n$, find parameters $a_0,a_1,\ldots,a_n$ such that

$$\Phi(x_i;a_0,\ldots,a_n) = f_i, \quad i=0,1,\ldots,n.$$

Here are some examples.

Example 2.1 (Polynomial interpolation).

$$\Phi(x;a_0,\dots,a_n) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n.$$

Example 2.2 (Rational interpolation).

$$\Phi(x;a_0,\dots,a_n;b_0,\dots,b_m) = \frac{a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n}{b_0 + b_1 x + \cdots + b_m x^m}.$$

Example 2.3 (Trigonometric interpolation).

$$\Phi(x;a_0,\dots,a_n) = a_0 + a_1 e^{ix} + a_2 e^{2ix} + \cdots + a_n e^{inx}.$$

Example 2.4 (Exponential interpolation).

$$\Phi(x;a_0,\dots,a_n;\lambda_0,\dots,\lambda_n) = a_0 e^{\lambda_0 x} + a_1 e^{\lambda_i x} + \cdots + a_n e^{\lambda_n x}.$$

There are many others, for example splines, which we will address later.

Definition 2.1. The interpolation problem is linear if $\Phi(x_i;a_0,\dots,a_n)$ depends linearly on $a_0, a_1, \dots, a_n$, i.e.

$$\Phi(x;a_0,\dots,a_n) = a_0\Phi_0(x) + a_1\Phi_1(x) + \cdots + a_n\Phi_n(x),$$

for some functions $\Phi_i(x)$, $i = 0,1,\dots,n$.

Easy to see that the interpolation problems in Example 2.1 and Example 2.3 are linear and in Example 2.2 and Example 2.4 are nonlinear.

2.2 Polynomial Interpolation in 1D

Polynomial interpolation is historically very important and well investigated problem, due to all nice properties of polynomials.

Problem 1 (Polynomial Interpolation). Given (x_i, f_i), $i = 0,1,\dots,n$ find

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

such that

$$p(x_i) = f_i \quad i = 0,1,\dots,n.$$

The main motivation for the approximation is to estimate the unknown values of $f(x)$.

Definition 2.2. If the point $\bar{x}$ lies inside the interval formed by the points $x_1,\dots,x_n$, we speak about interpolation; if the point $\bar{x}$ lies outside the interval formed by the points $x_1,\dots,x_n$, we speak about extrapolation.

First, we establish that the interpolation problem is well-posed.

Theorem 2.1 (Existence and Uniqueness). *Given $n+1$ distinct points x_i, i.e. $x_i \neq x_j$ for $i \neq j$ and arbitrary $n+1$ values $f_0, f_1, \dots, f_n$. There exists a unique polynomial $p_n(x)$ of degree n or less such that $p_n(x_i) = f_i$ for $i = 0,1,\dots,n$.*

Proof. First we will establish uniqueness.

Uniqueness.

Assume there are two such polynomials $p_n(x)$ and $\tilde{p}_n(x)$. Then $q(x) = p_n(x) - \tilde{p}_n(x)$ is a polynomial of degree at most n that has $n+1$ roots, namely x_i, $i = 0, 1, \ldots, n$. The only polynomial with such property is zero polynomial. Thus $q(x) \equiv 0$ and $\tilde{p}_n(x) = p_n(x)$.

Existence.

For $i = 0, 1, \ldots, n$ consider

$$L_i(x) = \frac{(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)} = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j}.$$

Then each $L_i(x)$ is a polynomial of degree n with the property

$$L_i(x_j) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j. \end{cases}$$

Thus,

$$p_n(x) = f_0 L_0(x) + f_1 L_1(x) + \cdots + f_n L_n(x)$$

is the desired polynomial.

Example 2.5.

x_i	0	1	3
f_i	1	3	2

Thus,

$$L_0(x) = \frac{(x-1)(x-3)}{(0-1)(0-3)}, \quad L_1(x) = \frac{x(x-3)}{(1-0)(1-3)}, \quad L_2(x) = \frac{x(x-1)}{(3-0)(3-1)}$$

and cross-multiplying, we compute

$$p(x) = 1 \cdot \frac{(x-1)(x-3)}{3} - 3 \cdot \frac{x(x-3)}{2} + 2 \cdot \frac{x(x-1)}{6} = \frac{-5x^2+17x+6}{6}.$$

From now on we assume that the points $x_1, \ldots, x_n$ are distinct. Given a basis $\psi_1(x), \ldots, \psi_n(x)$ of $\mathbb{P}_{n-1}$. Problem 1 we now can state as to find coefficients $a_1, \ldots, a_n$ for the polynomial

$$p(x) = a_1\psi_1(x) + a_2\psi_2(x) + \cdots + a_n\psi_n(x) \in \mathbb{P}_{n-1}$$

such that

$$p(x_i) = a_1\psi_1(x_i) + a_2\psi_2(x_i) + \cdots + a_n\psi_n(x_i) = f_i$$

for $i = 1, 2, \ldots, n$. Which is equivalent to the linear system

$$\begin{pmatrix} \psi_1(x_1) & \psi_2(x_1) & \ldots & \ldots & \psi_n(x_1) \\ \psi_1(x_2) & \psi_2(x_2) & \ldots & \ldots & \psi_n(x_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \psi_1(x_n) & \psi_2(x_n) & \ldots & \ldots & \psi_n(x_n) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix}. \tag{2.1}$$

In Theorem 2.1 we have established the existence and uniqueness of the solution for the linear system (2.1) with arbitrary right hand side. As a result the matrix in (2.1) is non-singular for any choice of basis. Thus mathematically any choice of basis would work, however computationally it makes a huge difference. Here are some natural choices.

2.2.1 Monomial Basis

We denote the monomial basis by

$$M_0 = 1, \quad M_1 = x, \quad M_2 = x^2, \quad \cdots \quad M_{n-1} = x^{n-1}.$$

The resulting matrix takes the form

$$\begin{pmatrix} M_0(x_1) & M_1(x_1) & \dots & \dots & M_{n-1}(x_1) \\ M_0(x_2) & M_1(x_2) & \dots & \dots & M_{n-1}(x_2) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ M_0(x_n) & M_1(x_n) & \dots & \dots & M_{n-1}(x_n) \end{pmatrix} = \begin{pmatrix} 1 & x_1 & \dots & \dots & x_1^{n-1} \\ 1 & x_2 & \dots & \dots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & \dots & \dots & x_n^{n-1} \end{pmatrix},$$

which is known as Vandermonde matrix. We can take several observation. First of all the matrix is full, so potentially it can be expensive to solve the linear system. Secondly, looking at the plot of the monomial basis we can observe that they are very similar near 0, meaning that if the interpolating points are near zero, the matrix can be close to a singular. Later we make it more precise when we study the condition number of a matrix. On the other hand once the coefficients $a_0, \dots, a_{n-1}$ are found out, the evaluation of the resulting polynomial at any point $\bar{x}$ is rather simple. We can make it even very efficient by noticing

$$\begin{aligned} p(x) &= a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-2} x^{n-2} + a_{n-1} x^{n-1} \\ &= a_0 + \left[a_1 + a_2 x + \cdots + a_{n-2} x^{n-3} + a_{n-1} x^{n-2}\right] x \\ &= a_0 + \left[a_1 + \left[a_2 + \cdots + a_{n-2} x^{n-4} + a_{n-1} x^{n-3}\right] x\right] x \\ &= a_0 + [a_1 + [a_2 + \cdots + [a_{n-2} + a_{n-1} x] \cdots] x] x. \end{aligned} \tag{2.2}$$

Using this nested form, we can write algorithm for evaluation, known as Horner's Scheme.

Algorithm 2.2 (Horner's Scheme for monomial basis)

Input: *The interpolation points $x_1, \dots, x_n$*
The coefficients $a_1, \dots, a_n$
The point (or points) $\bar{x}$ at which the polynomial to be evaluated

Output: *The value of the interpolating polynomial $p(\bar{x})$*

1. $p = a_n$
2. *for* $i = n-1 : -1 : 1$
3. $\quad p = p * x + a_i$
4. *end*

2.2.2 Lagrange Basis

Another obvious choice using Lagrange basis

$$L_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

Since

$$L_i(x_j) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j. \end{cases}$$

The resulting matrix is just an identity matrix and as a result $a_1 = f_1, \ldots, a_n = f_n$ and

$$p_n(x) = f_1 L_1(x) + f_2 L_2(x) + \cdots + f_n L_n(x).$$

However the evaluation of the above expression at some $\bar{x}$ is not cheap.

2.2.3 Newton's Basis

We denote the Newton's basis by

$$N_0 = 1, \quad N_1 = x - x_1, \quad N_2 = (x - x_1)(x - x_2), \quad \cdots \quad , N_{n-1} = \prod_{j=1}^{n-1} (x - x_j).$$

The resulting matrix takes the form

$$\begin{pmatrix} N_0(x_1) & N_1(x_1) & N_2(x_1) & \ldots & \ldots & N_{n-1}(x_1) \\ N_0(x_2) & N_1(x_2) & N_2(x_2) & \ldots & \ldots & N_{n-1}(x_2) \\ N_0(x_3) & N_1(x_3) & N_2(x_3) & \ldots & \ldots & N_{n-1}(x_3) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ N_0(x_n) & N_1(x_n) & N_2(x_n) & \ldots & \ldots & N_{n-1}(x_n) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \ldots & \ldots & 0 \\ 1 & x_2 - x_1 & 0 & \ldots & \ldots & 0 \\ 1 & x_3 - x_1 & (x_3 - x_1)(x_3 - x_2) & \ldots & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n - x_1 & (x_n - x_1)(x_n - x_2) & \ldots & \ldots & \prod_{j=1}^{n-1}(x_n - x_j) \end{pmatrix},$$

which is a lower triangular matrix and can be solved explicitly by forward substituion. Thus,

$$\begin{aligned} a_1 &= f_1 \\ a_2 &= \frac{f_2 - a_1}{x_2 - x_1} \\ a_3 &= \frac{f_3 - a_1 - a_2(x_2 - x_1)}{(x_3 - x_1)(x_3 - x_2)} \\ &\vdots \\ a_n &= \frac{f_n - \sum_{i=1}^{n-1} a_i \prod_{j=1}^{i-1}(x_n - x_j)}{\prod_{j=1}^{n-1}(x_n - x_j)}. \end{aligned}$$

Similarly, to the Monomial basis, once the coefficients $a_0, \ldots, a_{n-1}$ are found, the evaluation of the resulting polynomial at any point $\bar{x}$ can be done via Horner's Scheme as well, since

$$\begin{aligned} p(x) &= a_0 + a_1(x-1) + a_2(x-x_1)(x-x_2) + \cdots + a_{n-2}\prod_{j=1}^{n-2}(x-x_j) + a_{n-1}\prod_{j=1}^{n-1}(x_n-x_j) \\ &= a_0 + \left[a_1 + a_2 x + \cdots + a_{n-2}\prod_{j=2}^{n-2}(x-x_j) + a_{n-1}\prod_{j=2}^{n-1}(x-x_j)\right](x-x_1) \\ &= a_0 + \left[a_1 + \left[a_2 + \cdots + a_{n-2}\prod_{j=3}^{n-2}(x-x_j) + a_{n-1}\prod_{j=3}^{n-2}(x-x_j)\right](x-x_2)\right](x-x_1) \\ &= a_0 + [a_1 + [a_2 + \cdots + [a_{n-2} + a_{n-1}(x-x_{n-1})]\cdots](x-x_2)](x-x_1). \end{aligned} \tag{2.3}$$

Using this nested form, we can write algorithm for evaluation, known as Horner's Scheme.

Algorithm 2.3 (Horner's Scheme for Newton's basis)

Input: *The interpolation points $x_1,\ldots,x_n$*
The coefficients $a_1,\ldots,a_n$
The point (or points) $\bar{x}$ at which the polynomial to be evaluated

Output: *The value of the interpolating polynomial $p(\bar{x})$*

1. $p = a_n$
2. *for* $i = n-1:-1:1$
3. $\quad p = p*(x-x_i)+a_i$
4. *end*

2.3 Divided differences

For this section we need new notation.

Definition 2.3. We denote by $P_{n-1}(f \mid x_1,\ldots,x_n)(x)$ a polynomial fo degree $n-1$ that interpolated a function f at points $x_1,\ldots,x_n$.

The main idea of divided differences is based on elementary fact that for any two given points (x_1,f_1) and (x_2,f_2), with $x_1 \neq x_2$ there is unique straight line that passes though them, namely

$$l(x) = f_1 + \frac{f_2-f_1}{x_2-x_1}(x-x_1).$$

Using the new notation from Definition 2.3, we can rewrite it as

$$P_1(f \mid x_1,x_2)(x) = P_0(f \mid x_1)(x) + \frac{x-x_1}{x_2-x_1}\Big(P_0(f \mid x_2)(x) - P_0(f \mid x_1)(x)\Big),$$

since $P_0(f \mid x_i)(x)$ is just a constant function that passes through x_i, which is just f_i. Surprisingly above expression can be generalized to polynomials of arbitrary degree.

Theorem 2.4. *Given $P_{n-2}(f \mid x_1,\ldots,x_{n-1})(x)$ and $P_{n-2}(f \mid x_2,\ldots,x_n)(x)$, we can obtain $P_{n-1}(f \mid x_1,\ldots,x_n)(x)$ from*

$$P_{n-1}(f \mid x_1,\ldots,x_n)(x) = P_{n-2}(f \mid x_1,\ldots,x_{n-1})(x) + \frac{x-x_1}{x_n-x_1}\Big(P_{n-2}(f \mid x_2,\ldots,x_n)(x) - P_{n-2}(f \mid x_1,\ldots,x_{n-1})(x)\Big).$$

Proof. First of all we notice that on the right (and as a result on the left) is a polynomial of degree $n-1$. Thus, the only thing we need to check that it indeed interpolates the function f at $x_1,\dots,x_n$.

For $x = x_1$, since $P_{n-2}(f \mid x_1,\dots,x_{n-1})(x_1) = f_1$, we obtain

$$P_{n-1}(f \mid x_1,\dots,x_n)(x_1) = P_{n-2}(f \mid x_1,\dots,x_{n-1})(x_1) + 0 = f_1.$$

For $x = x_n$, since $P_{n-2}(f \mid x_2,\dots,x_{n-1})(x_n) = f_n$, we obtain

$$P_{n-1}(f \mid x_1,\dots,x_n)(x_1) = P_{n-2}(f \mid x_1,\dots,x_{n-1})(x_n) + \frac{x_n - x_1}{x_n - x_1}\Big(f_n - P_{n-2}(f \mid x_1,\dots,x_{n-1})(x_n)\Big) = f_n.$$

For $x = x_i$, $1 < i < n$, we notice that $P_{n-2}(f \mid x_2,\dots,x_n)(x_i) - P_{n-2}(f \mid x_1,\dots,x_{n-1})(x_i) = f_i - f_i = 0$ and as a result

$$P_{n-1}(f \mid x_1,\dots,x_n)(x_i) = P_{n-2}(f \mid x_1,\dots,x_{n-1})(x_i) + 0 = f_i.$$

The above result gives us a recursive way to construct the interpolating polynomial

$$\begin{array}{lllll}
P_0(f \mid x_1) & \searrow & & & \\
P_0(f \mid x_2) & \to & P_1(f \mid x_1,x_2) & & \\
\vdots & \vdots & \vdots & \ddots & \\
\vdots & \vdots & \vdots & \searrow & \\
P_0(f \mid x_{n-1}) & \to P_1(f \mid x_{n-2},x_{n-1}) & \dots \ \dots & \to P_{n-2}(f \mid x_1,\dots,x_{n-1}) & \searrow \\
P_0(f \mid x_n) & \to \ P_1(f \mid x_{n-1},x_n) & \dots \ \dots & \to \ P_{n-2}(f \mid x_2,\dots,x_n) & \to P_{n-1}(f \mid x_1,\dots,x_n).
\end{array}$$

2.3.1 Divided differences for Newton's basis

In this section we will see how efficient the method of divided differences can be for Newton's basis. For Newton's basis the interpolating polynomial has the form

$$P_{n-1}(f \mid x_1,\dots,x_n)(x) = \sum_{i=1}^{n} a_i \prod_{j=1}^{i-1}(x - x_j) = a_n x^{n-1} + \cdots. \tag{2.4}$$

From the above we can observe that the leading coefficient a_n in the interpolating polynomial is the same as in the leading coefficient for the polynomial in Newton's basis.

Definition 2.4. The leading coefficient a_k of the polynomial $P_{k-1}(f \mid x_1,\dots,x_k)(x)$ is called the $(k-1)$ divided difference and is denoted by $f[x_1,\dots,x_k]$.

Using this definition we can write the $P_{n-1}(f \mid x_1,\dots,x_n)(x)$ in Newton's basis as

$$P_{n-1}(f \mid x_1,\dots,x_n)(x) = \sum_{i=1}^{n} f[x_1,\dots,x_i] \prod_{j=1}^{i-1}(x - x_j). \tag{2.5}$$

From Theorem 2.4 it follows that

$$P_{k-j-1}(f \mid x_j,\dots,x_k)(x) = P_{k-j-2}(f \mid x_j,\dots,x_{k-1})(x) + \frac{x - x_j}{x_k - x_j}\Big(P_{k-j-2}(f \mid x_{j+1},\dots,x_k)(x) - P_{k-j-2}(f \mid x_j,\dots,x_{k-1})(x)\Big). \tag{2.6}$$

Using (2.5) we have

$$P_{k-j-1}(f \mid x_j,\ldots,x_k)(x) = \sum_{i=j}^{k} f[x_j,\ldots,x_i] \prod_{m=j}^{i-1}(x-x_m)$$
$$P_{k-j-2}(f \mid x_j,\ldots,x_{k-1})(x) = \sum_{i=j}^{k-1} f[x_j,\ldots,x_i] \prod_{m=j}^{i-1}(x-x_m)$$
$$P_{k-j-2}(f \mid x_{j+1},\ldots,x_k)(x) = \sum_{i=j+1}^{k} f[x_j,\ldots,x_i] \prod_{m=j}^{i-1}(x-x_m).$$

Plugging the above expressions into (2.6), we obtain

$$\begin{aligned}\sum_{i=j}^{k} f[x_j,\ldots,x_i] \prod_{m=j}^{i-1}(x-x_m) = &\sum_{i=j}^{k-1} f[x_j,\ldots,x_i] \prod_{m=j}^{i-1}(x-x_m) \\ &+ \frac{x-x_j}{x_k-x_j}\left(\sum_{i=j+1}^{k} f[x_j,\ldots,x_i] \prod_{m=j}^{i-1}(x-x_m) - \sum_{i=j}^{k-1} f[x_j,\ldots,x_i] \prod_{m=j}^{i-1}(x-x_m)\right).\end{aligned} \tag{2.7}$$

Since the leading coefficients of the polynomial on the left and on the right hand sides must be equal, we obtain the following formula

$$f[x_j,\ldots,x_k] = \frac{f[x_{j+1,}\ldots,x_k] - f[x_j,\ldots,x_{k-1}]}{x_k - x_j} \tag{2.8}$$

Using it we obtain a recursive way to compute coefficients

$$\begin{array}{ccccccc}
f[x_1] & \searrow & & & & & \\
f[x_2] & \to & f[x_1,x_2] & & & & \\
\vdots & \vdots & \vdots & & \ddots & & \\
\vdots & \vdots & \vdots & & \searrow & & \\
f[x_{n-1}] & \to & f[x_{n-2},x_{n-1}] & \cdots\ \cdots & \to & f[x_1,\ldots,x_{n-1}] & \searrow \\
f[x_n] & \to & f[x_{n-1},x_n] & \cdots\ \cdots & \to & f[x_2,\ldots,x_n] & \to f[x_1,\ldots,x_n].
\end{array}$$

Algorithm 2.5 (Newton's Interpolating polynomials)

Input: *The interpolation points* $x_1,\ldots,x_n$
The values $f_1,\ldots,f_n$
Output: *The coefficients* $a_1,\cdots,a_n$ *of the Newton's interpolating polynomial (returned as* $a_{11},\ldots,a_{nn}$*)*

1. *for* $i = 1:n$
2. $\quad a_{i1} = f_i$
3. *end*
4. *for* $j = 2:n$
5. $\quad$ *for* $i = j:n$
6. $\quad\quad a_{ij} = \frac{a_{i,j-1}-a_{i-1,j-1}}{x_i - x_{i-j+1}}$
7. $\quad$ *end*
8. *end*

One can modify the algorithm to save some storage by overwriting the entries of the coefficients that are needed anymore

---

Algorithm 2.6 (Modified Newton's Interpolating polynomials)

Input: *The interpolation points* $x_1,\ldots,x_n$
The values $f_1,\ldots,f_n$
Output: *The coefficients* $a_1,\cdots,a_n$ *of the Newton's interpolating polynomials (returned as* $a_1,\ldots,a_n$*)*

1. *for* $i = 1:n$
2. $\quad a_i = f_i$
3. *end*
4. *for* $j = 2:n$
5. $\quad$ *for* $i = j:n$
6. $\quad\quad a_i = \frac{a_i - a_{i-1}}{x_i - x_{i-j+1}}$
7. $\quad$ *end*
8. *end*

Example 2.6.

x_i	0	1	-1	2	-2
f_i	-5	-3	-15	39	-9

Using (2.8), we obtain

x_i	f_i				
0	-5				
1	-3	2			
-1	-15	6	-4		
2	39	18	12	8	
-2	-9	12	6	2	3

Hence the polynomial is

$$P_4(f \mid x_1,\ldots,x_5)(x) = -5 + 2x - 4x(x-1) + 8x(x-1)(x+1) + 3x(x-1)(x+1)(x-2).$$

We can notice that the table actually contains much more information. For example, it contains the coefficients of the polynomial interpolating $(-1,-15)$, $(2,39)$, $(-2,-9)$, which is

$$P_2(f \mid x_3,x_4,x_5)(x) = -15 + 18(x+1) + 6(x+1)(x-2)$$

or the coefficients of the polynomial interpolating $(1,-3)$, $(-1,-15)$, $(2,39)$, which is

$$P_2(f \mid x_2,x_3,x_4)(x) = -3 + 6(x-1) + 12(x-1)(x+1).$$

Once the coefficients of the Newton's polynomial are computed we can use Horner's Scheme (Algorithm 2.3) to evaluate it at given points.

2.4 Neville Algorithm

The recursive formula in Theorem 2.4 can be used to compute the value of the interpolating polynomial at some point $\bar{x}$ without computing coefficients a_i. Since

$$P_{k-j-1}(f \mid x_j,\ldots,x_k)(\bar{x}) = P_{k-j-2}(f \mid x_j,\ldots,x_{k-1})(\bar{x}) + \frac{\bar{x}-x_j}{x_k-x_j}\left(P_{k-j-2}(f \mid x_{j+1},\ldots,x_k)(\bar{x}) - P_{k-j-2}(f \mid x_j,\ldots,x_{k-1})(\bar{x})\right)$$

and naturally

$$P_0(f \mid x_i)(\bar{x}) = f_i \quad i = 1,2,\ldots,n.$$

The other values can be computed recursively, we obtain Neville scheme:

$$\begin{array}{lllll}
P_0(f \mid x_1)(\bar{x}) & \searrow & & & \\
P_0(f \mid x_2)(\bar{x}) & \rightarrow & P_1(f \mid x_1,x_2)(\bar{x}) & & \\
\vdots & \vdots & \vdots & \ddots & \\
\vdots & \vdots & \vdots & \searrow & \\
P_0(f \mid x_{n-1})(\bar{x}) & \rightarrow & P_1(f \mid x_{n-2},x_{n-1})(\bar{x}) & \ldots\ \ldots \rightarrow & P_{n-2}(f \mid x_1,\ldots,x_{n-1})(\bar{x}) \searrow \\
P_0(f \mid x_n)(\bar{x}) & \rightarrow & P_1(f \mid x_{n-1},x_n)(\bar{x}) & \ldots\ \ldots \rightarrow & P_{n-2}(f \mid x_2,\ldots,x_n)(\bar{x}) \rightarrow P_{n-1}(f \mid x_1,\ldots,x_n)(\bar{x}).
\end{array}$$

Algorithm 2.7 (Modified Newton's Interpolating polynomials)

Input: *The interpolation points $x_1,\ldots,x_n$*
The values $f_1,\ldots,f_n$
The point $\bar{x}$ at which the interpolating polynomial to be evaluated.
Output: *The coefficients value of the interpolating polynomial p*

1. *for* $i = 1:n$
2. $\quad p_i = f_i$
3. *end*
4. *for* $j = 2:n$
5. $\quad$ *for* $i = n:-1:j$
6. $\quad\quad p_i = p_{i-1} + \frac{\bar{x}-x_{i-j+1}}{x_i-x_{i-j+1}} * (p_i - p_{i-1})$
7. $\quad$ *end*
8. *end*
9. $p = p_n$

Example 2.7.

Assume we want to evaluate the polynomial interpolating

x_i	0	1	-1	2	-2
f_i	-5	-3	-15	39	-9

at $\bar{x} = 3$. Using Neville algorithm, we obtain

x_i	f_i					
0	-5					
1	-3	1				
-1	-15	9	-23			
2	39	57	105	169		
-2	-9	51	81	121	241	

Thus, $P_4(f \mid x_1,x_2,x_3,x_4,x_5)(3) = 241$.

2.5 Approximation properties of interpolating polynomials

In the previous section we show how to compute approximating polynomials and some evaluation techniques. In this section we try to answer a question, how close the approximation polynomial to the function f, namely we want an estimate for

$$\sup_{x\in[a,b]} |f(x) - P(f|x_1,\ldots,x_n)(x)| \tag{2.9}$$

on some interval $[a,b]$.

The idea behind the main result of this section is the Rolle's theorem.

Theorem 2.8 (Rolle's Theorem). *If $g \in C^1[a,b]$ and $g(a) = g(b) = 0$, then there exists $c \in (a,b)$ such that $g'(c) = 0$.*

Using it we can show the following result.

Theorem 2.9. *Let $x_1,\ldots,x_n$ be distinct points and $f \in C^n[a,b]$. Then for each $\bar{x} \in [a,b]$ there exists $\xi(\bar{x})$ in the interval generated by $x_1,\ldots,x_n,\bar{x}$ such that*

$$f(\bar{x}) - P(f \mid x_1,\ldots,x_n)(\bar{x})| = \frac{1}{n!}\omega(\bar{x})f^{(n)}(\xi(\bar{x})),$$

where $\omega(x) = \prod_{j=1}^{n}(x - x_j)$.

Proof. If $x = x_i$ for some $i = 1,\ldots,n$, then the result naturally holds. Assume $x \neq x_i$ for any $i = 1,\ldots,n$. Consider a function

$$\psi(x) = f(x) - P(f \mid x_1,\ldots,x_n)(x) - c\omega(x),$$

where the constant c is taken to be

$$c = \frac{f(\bar{x}) - P(f \mid x_1,\ldots,x_n)(\bar{x})}{\omega(\bar{x})}.$$

With this choice of the constant c, the function $\psi(x)$ has at least $n+1$ roots, namely at $x_1,\ldots,x_n$ and $\bar{x}$. Thus, from the Rolle's Theorem there exist n points, call them $x_i^{(1)}$, $i = 1,\ldots,n$, such that

$$\psi'(x_i^{(1)}) = 0, \quad i = 1,\ldots,n.$$

Again, from the Rolle's Theorem there exist $n-1$ points, call them $x_i^{(2)}$, $i = 1,\ldots,n-1$, such that

$$\psi''(x_i^{(2)}) = 0, \quad i = 1,\ldots,n-1.$$

Continue this process, there exists a point $x_1^{(n)}$ such that

$$\psi^{(n)}(x_1^{(n)}) = 0.$$

From the definition of $\psi(x)$, we have

$$\frac{d^n}{dx^n}\psi(x) = \frac{d^n}{dx^n}f(x) - \frac{d^n}{dx^n}P(f \mid x_1,\ldots,x_n)(x) - c\frac{d^n}{dx^n}\omega(x).$$

Since $P(f \mid x_1,\ldots,x_n)(x)$ is a polynomial of degree $n-1$, we have

$$\frac{d^n}{dx^n}P(f \mid x_1,\ldots,x_n)(x) = 0$$

and since $\omega(x)$ is a polynomial of degree n with leading coefficient 1, we have

$$\frac{d^n}{dx^n}\omega(x) = n!.$$

As a result

$$\psi^{(n)}(x_1^{(n)}) = f^{(n)}(x_1^{(n)}) - cn! = 0 \qquad \Rightarrow \qquad c = \frac{f^{(n)}(x_1^{(n)})}{n!},$$

which shows the theorem with $\xi(\bar{x}) = x_1^{(n)}$.

As a corollary, we immediately obtain

Corollary 2.1.

$$\max_{x\in[a,b]} |f(x) - P(f \mid x_1,\ldots,x_n)(x)| \le \frac{1}{n!}\max_{x\in[a,b]}|f^{(n)}(\xi(x))| \max_{x\in[a,b]}\left|\prod_{j=1}^{n}(x-x_j)\right|.$$

Thus we observe that the error is bounded by three terms. First term $\frac{1}{n!}$ decreases rather fast, the second term $\max_{x\in[a,b]}|f^{(n)}(\xi(x))$ depends on the (unknown) function f, the last term $\max_{x\in[a,b]}\left|\prod_{j=1}^{n}(x-x_j)\right|$ looks rather mysterious. Of course we can estimate it roughly as

$$\max_{x\in[a,b]}\left|\prod_{j=1}^{n}(x-x_j)\right| \le (b-a)^n,$$

and if we can control n-th derivative of f by M^n, then from Corollary 2.1, we obtain

$$\max_{x\in[a,b]} |f(x) - P(f \mid x_1,\ldots,x_n)(x)| \le \frac{M^n(b-a)^n}{n!} \to 0 \quad \text{as} \quad n\to\infty.$$

Example 2.8. Consider $f(x) = \sin(3x)$ on $[0,\pi]$. Then $|f^{(n)}(x)| \le 3^n$ an we obtain

$$\max_{x\in[a,b]} |\sin(3x) - P(f \mid x_1,\ldots,x_n)(x)| \le \frac{3^n\pi^n}{n!}.$$

Although, we know that $\lim_{n\to\infty}\frac{3^n\pi^n}{n!} = 0$ we need many points before $\frac{3^n\pi^n}{n!}$ is small. Thus for $n=20$, $\frac{3^n\pi^n}{n!} \approx 12.5689$, for $n=30$, $\frac{3^n\pi^n}{n!} \le 6.3*10^{-4}$. Thus even for this simple example we need to deal with high order polynomials. Of course we used rather cruel estimate for $\max_{x\in[a,b]}|\omega(x)|$, and indeed if we have more information about the locations of $x_1,\ldots,x_n$ we can obtain better estimates.

2.6 Equidistant points

It is very natural to select n equidistant points on an interval (a,b)

$$x_i = a+(i-1)h \quad \text{with} \quad h=\frac{b-a}{n-1}, \quad i=1,\dots,n.$$

For $n=2$, we have $x_1=a$ and $x_2=b$, $h=b-a$ and

$$\omega(x)=(x-a)(x-b)=(x-x_1)(x-x_2).$$

Since it is a parabola the maximum value $|\omega(x)|$ attains at $\frac{x_1+x_2}{2}$ and as a result

$$|\omega(x)|=|(x-x_1)(x-x_2)|=\frac{(x_2-x_1)^2}{4}=\frac{(b-a)^2}{4}=\frac{h^2}{4}. \tag{2.10}$$

Thus, from Corollary 2.1 we obtain that in this case

$$\max_{x\in[a,b]}|f(x)-P_1(f\mid x_1,x_2)(x)|\le\frac{(b-a)^2}{8}\max_{x\in[a,b]}|f''(x)|\le\frac{h^2}{8}\max_{x\in[a,b]}|f''(x)|. \tag{2.11}$$

For $n\ge 3$ we can obtain the following result.

Lemma 2.1. *For equidistant points*

$$x_i = a+(i-1)h \quad \textit{with} \quad h=\frac{b-a}{n-1}, \quad i=1,\dots,n,$$

and arbitrary $x\in[a,b]$ the following estimate holds

$$|\omega(x)|=\left|\prod_{i=1}^{n}(x-x_i)\right|\le\frac{h^n}{4}(n-1)!.$$

Proof. Let $x\in[x_j,x_{j+1}]$ for some $1\le j\le n-1$. From (2.10) it follows that

$$|(x-x_j)(x-x_{j+1})|\le\frac{h^2}{4}.$$

We have

$$\begin{aligned}\left|\prod_{i=1}^{n}(x-x_i)\right| &= \left|\prod_{i=1}^{j-1}(x-x_i)\right|\cdot|(x-x_j)(x-x_{j+1})|\cdot\left|\prod_{i=j+2}^{n}(x-x_i)\right| \\ &\le \left|\prod_{i=1}^{j-1}(x-x_i)\right|\cdot\frac{h^2}{4}\cdot\left|\prod_{i=j+2}^{n}(x-x_i)\right| \\ &\le \frac{h^2}{4}\left|\prod_{i=1}^{j-1}(x_{j+1}-x_i)\right|\cdot\left|\prod_{i=j+2}^{n}(x_j-x_i)\right|.\end{aligned}$$

Since $x_i=a+(i-1)h$, we have $|x_j-x_i|=|j-i|h$ and as a result

$$\left|\prod_{i=1}^{n}(x-x_i)\right| \le \frac{h^2}{4}\prod_{i=1}^{j-1}(j+1-i)h\cdot\prod_{i=j+2}^{n}(i-j)h$$
$$\le \frac{h^n}{4}\prod_{i=1}^{j-1}(j+1-i)\cdot\prod_{i=j+2}^{n}(i-j)$$
$$\le \frac{h^n}{4}j!(n-j)!$$

and since $j!(n-j)! \le (n-1)!$ for $1 \le j \le n-1$, we obtain the result.

Using the above result we obtain.

Theorem 2.10. *Let*

$$x_i = a+(i-1)h \quad with \quad h=\frac{b-a}{n-1}, \quad i=1,\ldots,n.$$

If $f \in C^n[a,b]$, then

$$\max_{x\in[a,b]}|f(x)-P(f\mid x_1,\ldots,x_n)(x)| \le \frac{h^n}{4n}\max_{x\in[a,b]}|f^{(n)}(\xi(x))|.$$

We revisit Example 2.8, where the above theorem gives sharper estimate.

Example 2.9. Consider $f(x)=\sin(3x)$ on $[0,\pi]$. Then $|f^{(n)}(x)| \le 3^n$ and for equidistant approximation with $h=\pi/(n-1)$ from Theorem 2.10 we obtain

$$\max_{x\in[a,b]}|\sin(3x)-P(f\mid x_1,\ldots,x_n)(x)| \le \frac{3^n}{4n}\left(\frac{\pi}{n-1}\right)^n.$$

The above estimate is much sharper than the one we used in Example 2.8. Thus for $n=20$, $\frac{3^n}{4n}\left(\frac{\pi}{n-1}\right)^n \approx 1.0169e-08$ and for $n=30$, $\frac{3^n}{4n}\left(\frac{\pi}{n-1}\right)^n \approx 1.8924e-17$.

2.7 Chebyshev points

A natural question: is there a choice for the interpolation notes that minimizes $max_{x\in[a,b]}|\omega(x)|$, i.e. what is the solution to the following min-max problem

$$\min_{x_1,\ldots,x_n}\max_{x\in[a,b]}\left|\prod_{j=1}^{n}(x-x_j)\right|. \tag{2.12}$$

The solution $x_1^*,\ldots,x_n^*$ to (2.12) are called the Chebyshev points and are given by the formula

$$x_i^* = \frac{a+b}{2}+\frac{a+b}{2}\cos\left(\frac{(2i-1)\pi}{2n}\right), \quad i=1,\ldots,n.$$

In addition, one can show

$$\left|\prod_{j=1}^{n}(x-x_j^*)\right| \le \frac{2^{1-2n}}{n!}(b-a)^n.$$

Example 2.10. Consider $f(x)=\frac{1}{1+x^2}$ TO ADD

2.8 Hermite interpolation

2.9 Spline interpolation

For approximation of a function f on an interval $[a,b]$, instead of choosing high order interpolating polynomial one can partition the interval into small pieces and on each small piece use small or moderate order polynomials for an approximation. In addition, one may choose various ways to connect pieces together, resulting in global smoothness properties. The advantage of such approach is that no high order of smoothness of f is required. We will consider two popular choices, linear (continuous) splines and cubic (C^2) splines.

2.9.1 Linear Splines

Let $x_1,\dots,x_n$ be such that

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

Our goal to approximate f by piecewise linear polynomials, i.e. on each subinterval $[x_i,x_{i+1}]$ we seek a linear function

$$S_i(x) = a_i + b_i(x - x_i), \quad i = 1,\dots,n-1.$$

Thus on each subinterval we have 2 unknowns. Since there are $n-1$ subintervals we have $2n-2$ unknowns in total. We want our spline function $S(x)$ to have the following properties:

1. Interpolation at nodes, i.e. $S(x_i) = f(x_i) := f_i$ for $i = 1,2,\dots,n$
2. Continuity $S_{i+1}(x_{i+1}) = S_i(x_{i+1})$ for $i = 2,\dots,n-1$.

Thus in total we have $n+n-2 = 2n-2$ conditions, which matches the total number of unknowns. Easy to see that the conditions above uniquely determine $S(x)$ and on each subinterval $[x_i,x_{i+1}]$ the coefficients a_i and b_i in $S_i(x) = a_i + b_i(x-x_i)$ for $i = 1,\dots,n-1$ are given by

$$a_i = f_i \quad b_i = \frac{f_{i+1} - f_i}{x_{i+1} - x_i}.$$

Below we provide algorithms for computing the coefficients and evaluation.

Algorithm 2.11 (Computing coefficients for linear spline)

Input: *The interpolation nodes $x_1,\dots,x_n$*
The function values $f_1,\dots,f_n$

Output: *The coefficients $a_1,\dots,a_n$ and $b_1,\dots,b_n$ of the linear spline*

1. *for* $i = 1 : n-1$
2. $a_i = f_i$
3. $b_i = (f_{i+1} - f_i)/(x_{i+1} - x_i)$
4. *end*

Algorithm 2.12 (Linear spline evaluation)

Input: *The interpolation nodes $x_1,\dots,x_n$*

The coefficients $a_1,\ldots,a_n$ and $b_1,\ldots,b_n$
The point $\bar{x}$ in $[x_1,x_n]$ at which the linear spline to be evaluated.

Output: *The value $S = S(\bar{x})$ of the linear spline*

1. *for* $i = 1 : n-1$
2. *if* $\bar{x} \le x_{i+1}$
3. $S = a_i + b_i * (\bar{x} - x_i)$
4. *end*
5. *end*

From (2.11), we obtain the following convergence property of the linear spline

Theorem 2.13. *Let $S(x)$ be the linear spline interpolating f at $x_1,\ldots,x_n$. If $f \in C^2[a,b]$ then*

$$\max_{x\in[a,b]} |f(x) - S(x)| \le \frac{h^2}{8} \max_{x\in[a,b]} |f''(x)|,$$

where

$$h = \max_{i=1,\ldots,n-1} h_i = \max_{i=1,\ldots,n-1} (x_{i+1} - x_i).$$

2.9.2 Cubic Splines

Again, let $x_1,\ldots,x_n$ be such that

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

Our goal to approximate f by piecewise cubic polynomials, i.e. on each subinterval $[x_i,x_{i+1}]$ we seek a cubic function

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad i = 1,\ldots,n-1.$$

Thus on each subinterval we have 4 unknowns. Since there are $n-1$ subintervals we have $4n-4$ unknowns in total. We want our spline function $S(x)$ to be smooth, we may asked for the following properties:

1. Interpolation at nodes, i.e. $S(x_i) = f(x_i) := f_i$ for $i = 1,2,\ldots,n$
2. Continuity $S_{i+1}(x_{i+1}) = S_i(x_{i+1})$ for $i = 2,\ldots,n-1$.
3. Continuity of the first derivatives $S'_{i+1}(x_{i+1}) = S'_i(x_{i+1})$ for $i = 2,\ldots,n-1$.
4. Continuity of the second derivatives $S''_{i+1}(x_{i+1}) = S''_i(x_{i+1})$ for $i = 2,\ldots,n-1$.

Thus in total we have $n + 3(n-2) = 4n-6$ conditions, which does not match the number of unknowns, we are two short. The popular choices are:

- Natural boundary $S''_1(x_1) = 0$ and $S''_{n-1}(x_n) = 0$
- Clamped boundary $S'_1(x_1) = f'(x_1)$ and $S'_{n-1}(x_n) = f'(x_n)$
- Periodic spline $S_1(x_1) = S_{n-1}(x_n)$, $S'_1(x_1) = S'_{n-1}(x_n)$, and $S''_1(x_1) = S''_{n-1}(x_n)$.

It is not still obvious how to compute the cubic spline from these conditions. This is what we will address next. First, we introduce a notation

$$h_i = x_{i+1} - x_i, \quad i = 1,\ldots,n-1.$$

Differentiating the expression

$$S_i(x) = a_i + b_i(x-x_i) + c_i(x-x_i)^2 + d_i(x-x_i)^3, \quad i = 1,\dots,n-1, \tag{2.13}$$

we obtain

$$S_i'(x) = b_i + 2c_i(x-x_i) + 3d_i(x-x_i)^2, \quad i = 1,\dots,n-1 \tag{2.14}$$
$$S_i''(x) = 2c_i + 6d_i(x-x_i), \quad i = 1,\dots,n-1. \tag{2.15}$$

From 2.13 we immediately find

$$a_i = S_i(x_i) = f_i \quad i = 1,\dots,n-1.$$

Since all a_i are known, the goal is to express the other coefficients, i.e. b_i, c_i, and d_i in terms of a_i. From the continuity at nodes we also have

$$a_{i+1} = S_{i+1}(x_{i+1}) = S_i(x_{i+1}) = a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 \quad i = 1,\dots,n-2.$$

Put $a_n = f(x_n)$. Then for $i = n-1$ we also have

$$a_n = S_{n-1}(x_n) = a_{n-1} + b_{n-1}h_{n-1} + c_{n-1}h_{n-1}^2 + d_{n-1}h_{n-1}^3.$$

Thus,

$$a_{i+1} = a_i + b_i h_i + c_i h_i^2 + d_i h_i^3, \quad i = 1,\dots,n-1. \tag{2.16}$$

Similarly, from the continuity of the derivatives we obtain

$$b_{i+1} = S_{i+1}'(x_{i+1}) = S_i'(x_{i+1}) = b_i + 2c_i h_i + 3d_i h_i^2 \quad i = 1,\dots,n-2.$$

Setting $b_n = S_{n-1}'(x_n)$. Thus, we also have

$$b_n = S_{n-1}'(x_n) = b_{n-1} + 2c_{n-1}h_{n-1} + 3d_{n-1}h_{n-1}^2$$

Summarizing,

$$b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2, \quad i = 1,\dots,n-1. \tag{2.17}$$

From the continuity of the second derivatives we obtain

$$2c_{i+1} = S_{i+1}''(x_{i+1}) = S_i''(x_{i+1}) = 2c_i + 6d_i h_i, \quad i = 1,\dots,n-2.$$

Again, setting $c_n = \frac{1}{2}S_{n-1}''(x_n)$. Thus, we also have

$$2c_n = S_{n-1}''(x_n) = b_{n-1} + 2c_{n-1}h_{n-1} + 3d_{n-1}h_{n-1}^2.$$

Summarizing,

$$2c_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2, \quad i = 1,\dots,n-1. \tag{2.18}$$

From (2.18), we find

$$d_i = \frac{c_{i+1} - c_i}{3h_i}, \quad i = 1,\dots,n-1. \tag{2.19}$$

Inserting it into (2.16) and (2.17), we obtain

$$a_{i+1} = a_i + b_i h_i + \frac{h_i^2}{3}(2c_{i+1} + c_i), \quad i = 1,\dots,n-1 \tag{2.20}$$

and

$$b_{i+1} = b_i + h_i(c_{i+1} + c_i), \quad i = 1,\dots,n-1. \tag{2.21}$$

Solving (2.20) for b_i we find

$$b_i = \frac{1}{h_i}(a_{i+1} - a_i) - \frac{h_i}{3}(2c_{i+1} + c_i), \quad i = 1, \ldots, n-1. \tag{2.22}$$

Replacing i with $i-1$ we have

$$b_i = b_{i-1} + h_{i-1}(c_i + c_{i-1}), \quad i = 2, \ldots, n. \tag{2.23}$$

and

$$b_{i-1} = \frac{1}{h_{i-1}}(a_i - a_{i-1}) - \frac{h_{i-1}}{3}(2c_i + c_{i-1}), \quad i = 2, \ldots, n. \tag{2.24}$$

Inserting (2.22) and (2.24) into (2.23), we obtain

$$\frac{1}{h_i}(a_{i+1} - a_i) - \frac{h_i}{3}(2c_{i+1} + c_i) = \frac{1}{h_{i-1}}(a_i - a_{i-1}) - \frac{h_{i-1}}{3}(2c_i + c_{i-1}) + h_i(c_{i+1} + c_i).$$

Moving all $c_i's$ to the left and $a_i's$ to the right and rearranging the terms, we obtain

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} = \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1}) \quad i = 2, \ldots, n-1. \tag{2.25}$$

Which is equivalent to $n-2$ equations with n unknowns. We need additional conditions (like natural or periodic boundary) to close the system.

Alternatively, we could introduce variables $M_i = S''(x_i)$, $i = 1, \ldots, n$, often called moments, and set equations for them. Of course, $M_i = 2c_i$, however the point of view is slightly different. Thus, since $S_i(x)$ is a cubic on $[x_i, x_{i+1}]$, $S_i''(x)$ is linear on $[x_i, x_{i+1}]$ and in terms of moments has the form

$$S_i''(x) = M_{i+1}\frac{x - x_i}{h_i} + M_i\frac{x_{i+1} - x}{h_i}, \quad i = 1, \ldots, n-1.$$

Integrating, we obtain

$$S_i'(x) = M_{i+1}\frac{(x - x_i)^2}{2h_i} - M_i\frac{(x_{i+1} - x)^2}{2h_i} + b_i, \quad i = 1, \ldots, n-1.$$

and integrating once again

$$S_i(x) = M_{i+1}\frac{(x - x_i)^3}{6h_i} + M_i\frac{(x_{i+1} - x)^3}{6h_i} + b_i(x - x_i) + a_i, \quad i = 1, \ldots, n-1.$$

And now one can work with this form to derive a linear system for M_i.

2.9.3 Natural Cubic Splines

From the conditions $S_1''(x_1) = 0$ and $S_{n-1}''(x_n) = 0$ we have $c_1 = 0$ and $c_n = 0$. Thus the equations (2.25) are equivalent to the system

$$A\mathbf{c} = \mathbf{g},$$

where A is $(n-2) \times (n-2)$ matrix given by

$$A = \begin{pmatrix} 2(h_1+h_2) & h_2 & & & & \\ h_2 & 2(h_2+h_3) & & & & \\ & \ddots & \ddots & \ddots & & \\ & & h_{n-4} & 2(h_{n-4}+h_{n-3}) & h_{n-3} & \\ & & & h_{n-3} & 2(h_{n-3}+h_{n-2}) \end{pmatrix} \tag{2.26}$$

and $(n-2)$-vectors $\mathbf{c}$ and $\mathbf{g}$ are given by

$$\mathbf{c} = \begin{pmatrix} c_2 \\ c_3 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{pmatrix}, \quad \mathbf{g} = \begin{pmatrix} \frac{3}{h_2}(a_3-a_2) - \frac{3}{h_1}(a_2-a_1) \\ \frac{3}{h_3}(a_4-a_3) - \frac{3}{h_2}(a_3-a_2) \\ \vdots \\ \frac{3}{h_{n-2}}(a_{n-1}-a_{n-2}) - \frac{3}{h_{n-3}}(a_{n-2}-a_{n-3}) \\ \frac{3}{h_{n-1}}(a_n-a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1}-a_{n-2}) \end{pmatrix}. \tag{2.27}$$

Notice that for equidistant nodes $x_1,\ldots,x_n$ we have $h = h_i$ and the matrix A and the vector $\mathbf{g}$ take the form

$$A = h\begin{pmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 1 & 4 \end{pmatrix}, \quad \mathbf{g} = \frac{3}{h}\begin{pmatrix} a_3-2a_2+a_1 \\ a_4-2a_3+a_2 \\ \vdots \\ a_{n-1}-2a_{n-2}+a_{n-3} \\ a_n-2a_{n-1}+a_{n-2} \end{pmatrix} = \frac{3}{h}\begin{pmatrix} f(x_3)-2f(x_2)+f(x_1) \\ f(x_4)-2f(x_3)+f(x_2) \\ \vdots \\ f(x_{n-1})-2f(x_{n-2})+f(x_{n-3}) \\ f(x_n)-2f(x_{n-1})+f(x_{n-2}) \end{pmatrix}. \tag{2.28}$$

Using Taylor it is easy to see that

$$\frac{f(x_i-h)-2f(x_i)+f(x_i+h)}{h^2} \approx f''(x_i).$$

2.9.4 Computation and Convergence of the natural cubic splines

In this section we will derive error estimates for

$$\max_{x\in[a,b]} |S^{(k)}(x) - f^{(k)}(x)|, \quad \text{for } k = 0,1,2,3.$$

The key to the analysis will be Lemma 2.2. Matrix A posses very nice properties. Obviously, it is symmetric and diagonally dominant, so it is non-singular. Moreover, we have

Lemma 2.2. *Given a linear system $A\mathbf{z} = \mathbf{w}$, where A is the matrix given in* (2.26) *and $\mathbf{w} = (w_1,\ldots,w_{n-2})^T$ is arbitrary. Then for $\mathbf{z} = (z_1,\ldots,z_{n-2})^T$, we have*

$$\max_{1\le i\le n-2} |z_i| \le \max_{1\le i\le n-2} \frac{|w_i|}{h_i+h_{i+1}}.$$

Proof. Let $\max_{1\le i\le n-2}|z_i| = |z_r|$ for some $2 \le r \le n-3$. Then looking at the r-th row of $A\mathbf{z} = \mathbf{w}$, we have

$$h_r z_{r-1} + 2(h_r+h_{r+1})z_r + h_{r+1}z_{r+1} = w_r.$$

Using it together with the triangle inequality we obtain

$$\begin{aligned}\max_i \frac{|w_i|}{h_i+h_{i+1}} \geq \frac{|w_r|}{h_r+h_{r+1}} &= \frac{|h_r z_{r-1}+2(h_r+h_{r+1})z_r+h_{r+1}z_{r+1}|}{h_r+h_{r+1}} \\ &\geq 2|z_r| - \frac{h_r}{h_r+h_{r+1}}|z_{r-1}| - \frac{h_{r+1}}{h_r+h_{r+1}}|z_{r+1}| \\ &\geq 2|z_r| - \frac{h_r}{h_r+h_{r+1}}|z_r| - \frac{h_{r+1}}{h_r+h_{r+1}}|z_r| = |z_r|,\end{aligned}$$

where in the last step we used that $|z_r|$ is maximal. The cases $r = 1$ and $r = n-2$ are left to the reader.

From the above lemma we immediately obtain

Lemma 2.3. *The matrix A given in* (2.26) *is non-singular.*

Proof. Assume it is. Then there exists a vector $\mathbf{z} \neq \mathbf{0}$ such that $A\mathbf{z} = \mathbf{0}$. But then we get a contradiction since from the above lemma $\max_i |z_i| = 0$.

Hence from the system of linear equations $A\mathbf{c} = \mathbf{g}$ with A, $\mathbf{c}$, and $\mathbf{g}$ given by (2.26) and (2.27), we can obtain $\mathbf{c}$, and then using (2.22), we can obtain b_i and from (2.19), we can compute d_i. We summarize it in the following algorithms.

Algorithm 2.14 (Computing coefficients for natural cubic spline)

Input: *The interpolation nodes $x_1, \ldots, x_n$*
The function values $f_1, \ldots, f_n$

Output: *The coefficients $a_1, \ldots, a_{n-1}$, $b_1, \ldots, b_{n-1}$, $c_1, \ldots, c_{n-1}$, and $d_1, \ldots, d_{n-1}$ of the natural cubic spline*

for $i = 1 : n$
  $a_i = f_i$
end
for $i = 1 : n-1$
  $h_i = x_{i+1} - x_i$
end
Generate matrix $A \in \mathbb{R}^{(n-2)\times(n-2)}$ *and the vector* $\mathbf{g} \in \mathbb{R}^{n-2}$ *given in* (2.26) *and* (2.27)
Compute $c_2, \ldots, c_{n-1}$ *by solving the system* $A\mathbf{c} = \mathbf{g}$
Set $c_1 = 0$ *and* $c_n = 0$
for $i = 1 : n-1$
  $b_i = \frac{1}{h_i} * (a_{i+1} - a_i) - \frac{h_i}{3} * (2c_i + c_{i+1})$
  $d_i = \frac{1}{3h_i} * (c_{i+1} - c_i)$
end

Once the coefficients of the cubic spline are computed, given a point $\bar{x}$ we can use the following algorithm to evaluate $S(\bar{x})$.

Algorithm 2.15 (Natural cubic spline evaluation)

Input: *The interpolation nodes $x_1, \ldots, x_n$*
The coefficients $a_1, \ldots, a_{n-1}$, $b_1, \ldots, b_{n-1}$, $c_1, \ldots, c_{n-1}$, $d_1, \ldots, d_{n-1}$
The point $\bar{x}$ in $[x_1, x_n]$ at which the cubic spline to be evaluated.

Output: *The value $S = S(\bar{x})$ of the cubic spline*

1. *for* $i = 1 : n-1$
2. *if* $\bar{x} \le x_{i+1}$
3. $S = a_i + b_i * (\bar{x} - x_i) + c_i * (\bar{x} - x_i)^2 + d_i * (\bar{x} - x_i)^3$
4. *end*
5. *end*

Now we address the question of convergence. The key result we will use is Lemma 2.2. Define $\mathbf{M} = (M_1, \ldots, M_n)^T$ where $M_i = S_i''(x_i)$ and $\mathbf{F} = (F_1, \ldots, F_n)^T$ where $F_i = f''(x_i)$, $i = 1, \ldots, n$. Define $\mathbf{r} = (r_1, \ldots, r_n)^T$ by

$$\mathbf{r} = A(\mathbf{M} - \mathbf{F}).$$

Since $\mathbf{M} = 2\mathbf{c}$ and using that $A\mathbf{c} = \mathbf{g}$, we have

$$\mathbf{r} = A(\mathbf{M} - \mathbf{F}) = A\mathbf{M} - A\mathbf{F} = 2A\mathbf{c} - A\mathbf{F} = 2\mathbf{g} - A\mathbf{F}. \tag{2.29}$$

On the other hand by Lemma 2.2

$$\max_i |M_i - F_i| \le \max_i \frac{|r_i|}{h_i + h_{i+1}}.$$

From (2.29) and definition of matrix A and vector $\mathbf{g}$ we have

$$\begin{aligned} r_i &= 2g_i - (A\mathbf{F})_i \\ &= \frac{6}{h_i}(f(x_{i+1}) - f(x_i)) - \frac{6}{h_{i-1}}(f(x_i) - f(x_{i-1})) - \left(h_{i-1}f''(x_{i-1}) + 2(h_{i-1} + h_i)f''(x_i) + h_i f''(x_{i+1})\right). \end{aligned}$$

Since $x_{i+1} = x_i + h_i$ and $x_{i-1} = x_i - h_{i-1}$, using Taylor expansion (for f sufficiently smooth), we obtain

$$f(x_{i+1}) = f(x_i) + h_i f'(x_i) + \frac{h_i^2}{2} f''(x_i) + \frac{h_i^3}{6} f'''(x_i) + O(h_i^4)$$

and

$$f(x_{i-1}) = f(x_i) - h_{i-1} f'(x_i) + \frac{h_{i-1}^2}{2} f''(x_i) - \frac{h_{i-1}^3}{6} f'''(x_i) + O(h_{i-1}^4).$$

As a result

$$\frac{6}{h_i}(f(x_{i+1}) - f(x_i)) - \frac{6}{h_{i-1}}(f(x_i) - f(x_{i-1})) = 3h_i f''(x_i) + h_i^2 f'''(x_i) + O(h_i^3) + 3h_{i-1} f''(x_i) - h_{i-1}^2 f'''(x_i) + O(h_{i-1}^3). \tag{2.30}$$

Similarly,

$$f''(x_{i+1}) = f''(x_i) + h_i f'''(x_i) + O(h_i^2)$$

and

$$f''(x_{i-1}) = f''(x_i) - h_{i-1} f'''(x_i) + O(h_{i-1}^2).$$

As a result

$$h_{i-1}f''(x_{i-1}) + 2(h_{i-1} + h_i)f''(x_i) + h_i f''(x_{i+1}) = 3(h_i + h_{i-1})f''(x_i) - h_{i-1}^2 f'''(x_i) + h_i^2 f'''(x_i) + O(h_{i-1}^3) + O(h_i^3)$$

Subtracting it from (2.30), we obtain

$$r_i = O(h_i^3 + h_{i-1}^3),$$

and thus

$$\max_i |M_i - F_i| \leq \max_i \frac{|r_i|}{h_i + h_{i+1}} \leq \frac{O(h_i^3 + h_{i-1}^3)}{h_i + h_{i+1}} = O(h_i^2 + h_{i-1}^2) = O(h^2). \tag{2.31}$$

Thus we established that at the nodes

$$\max_i |S''(x_i) - f''(x_i)| \leq Ch^2, \tag{2.32}$$

for some constant independent of h provided $f \in C^4[a,b]$. The estimate (2.31) is key result for error estimates.

Theorem 2.16. *Let $f \in C^4[a,b]$ and $\max_i(h/h_i) \leq K$ for some $K > 0$. Then there exists a constant C independent of h such that*

$$\max_{x\in[a,b]} |S^{(k)}(x) - f^{(k)}(x)| \leq Ch^{4-k}, \quad k = 0,1,2,3.$$

Proof. **Case k=3.**

First we treat the case $k = 3$. Let $x \in [x_j, x_{j+1}]$ for some $1 \leq j \leq n-1$. Notice that since $S_j(x)$ is cubic, $S_j'''(x)$ is constant on $[x_j, x_{j+1}]$ and can be expressed as

$$S_j'''(x) = \frac{M_{j+1} - M_j}{h_j},$$

where $M_j = S_j''(x_j)$. Then

$$S_j'''(x) - f'''(x) = \frac{M_{j+1} - M_j}{h_j} - f'''(x) = \left(\frac{M_{j+1} - f''(x_{j+1})}{h_j}\right) + \left(\frac{f''(x_j) - M_j}{h_j}\right) + \left(\frac{f''(x_{j+1}) - f''(x_j)}{h_j} - f'''(x)\right)$$

From (2.32) and using that $h/h_j \leq K$ we have

$$\left(\frac{M_{j+1} - f''(x_{j+1})}{h_j}\right) + \left(\frac{f''(x_j) - M_j}{h_j}\right) \leq \frac{Ch^2}{h_j} \leq CKh.$$

Using Taylor expansion

$$\frac{f''(x_{j+1}) - f''(x_j)}{h_j} - f'''(x) = f'''(x_j) - f'''(x) + O(h) = O(h).$$

Thus we have established

$$\max_{x\in[x_j,x_{j+1}]} |S_j'''(x) - f'''(x)| \leq Ch, \tag{2.33}$$

which gives us the Theorem for $k = 3$.

Case k=2. To show the result in this case we use the result fro $k = 2$. Thus for $x \in [x_j, x_{j+1}]$ by the Fundamental Theorem of Calculus

$$S_j''(x) - f''(x) = S_j''(x_j) - f''(x_j) + \int_{x_j}^{x} \left(S_j'''(t) - f'''(t)\right) dt \leq Ch^2 + |x - x_j|Ch \leq Ch^2, \tag{2.34}$$

where we used (2.33) and (2.32).

Case k=1. Since $f(x_i) - S(x_i) = 0$ for $i = 1,\ldots,n$, by the Rolle's Theorem there exist $\xi_j \in [x_j, x_{j+1}]$. Then for $x \in [x_j, x_{j+1}]$ by the Fundamental Theorem of Calculus

$$S_j'(x) - f'(x) = \int_{\xi_j}^{x} \left(S_j''(t) - f''(t)\right) dt \leq |x - \xi_j|Ch^2 \leq Ch^3, \tag{2.35}$$

where we used (2.34).

Case k=0. Since $f(x_i) - S(x_i) = 0$ again for $x \in [x_j, x_{j+1}]$ by the Fundamental Theorem of Calculus

$$S_j(x) - f(x) = \int_{x_j}^{x} \left(S_j'(t) - f'(t)\right) dt \leq |x_j - x| Ch^3 \leq Ch^4, \tag{2.36}$$

where we used (2.36).

3 Linear systems

Linear algebra is a wonderful subject. One of the wonderful aspects of linear algebra is a variety of ways one can look at the same problem. Sometimes a hard problem what appears from one perspective can turn out to be trivial from another. To illustrate my point, let's look at some $m \times n$ matrix $A \in \mathbb{R}^{m\times n}$

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}.$$

Looking at this matrix we can see different things. First of all we see $m \times n$ entries, or in other word an element of $\mathbb{R}^{m\times n}$. We also can see m rows $A(i,:) \in \mathbb{R}^n$ for $i = 1,2,\dots,m$, or n columns $A(:,j) \in \mathbb{R}^m$ for $j = 1,2,\dots,n$. More sophisticated, one can see a map $A : \mathbb{R}^n \to \mathbb{R}^m$ or that matrix A consists of four submatrices $A_{11} \in \mathbb{R}^{m_1\times n_1}$, $A_{12} \in \mathbb{R}^{m_1\times n_2}$, $A_{21} \in \mathbb{R}^{m_2\times n_1}$, $A_{22} \in \mathbb{R}^{m_2\times n_2}$,

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad \text{with} \quad n = m_1 + m_2 \quad \text{and} \quad n = n_1 + n_2.$$

3.1 Matrix-Vector multiplication

Let $A \in \mathbb{R}^{m\times n}$ and $x \in \mathbb{R}^n$. The i-th component of the matrix-vector product $y = Ax$ is defined by

$$y_i = \sum_{j=1}^{n} a_{ij} x_j \tag{3.1}$$

i.e., y_i is the dot product (inner product) of the i-th row of A with the vector x.

$$\begin{pmatrix} \vdots \\ \hline y_i \\ \hline \vdots \end{pmatrix} = \begin{pmatrix} \vdots & \vdots & & \vdots \\ \hline a_{i1} & a_{i2} & \cdots & a_{in} \\ \hline \vdots & \vdots & & \vdots \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Another useful point of view is to look at entire vector $y = Ax$,

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = x_1 \begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{pmatrix} + x_2 \begin{pmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{pmatrix} + \cdots + x_n \begin{pmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{pmatrix}. \tag{3.2}$$

Thus, y is a linear combination of the columns of matrix A.

3.2 Matrix-Matrix multiplication

If $A \in \mathbb{R}^{m\times p}$ and $B \in \mathbb{R}^{p\times n}$, then $AB = C \in \mathbb{R}^{m\times n}$ defined by

$$c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj},$$

i.e. the ij-th element of the product matrix is the dot product between i-th row of A and j-th column of B.

$$\begin{pmatrix} \\ c_{ij} \\ \\ \end{pmatrix} = \left(\begin{array}{c} \\ \hline a_{i1} \cdots a_{ip} \\ \hline \\ \end{array}\right) \left(\begin{array}{c|} & b_{1j} & \\ & \vdots & \\ & b_{pj} & \end{array}\right).$$

Another useful point of view is to look at j-th column of C

$$\begin{pmatrix} c_{1j} \\ \vdots \\ c_{ij} \\ \vdots \\ c_{mj} \end{pmatrix} = b_{1j} \begin{pmatrix} a_{11} \\ \vdots \\ a_{i1} \\ \vdots \\ a_{m1} \end{pmatrix} + b_{2j} \begin{pmatrix} a_{12} \\ \vdots \\ a_{i2} \\ \vdots \\ a_{m2} \end{pmatrix} + \cdots + b_{nj} \begin{pmatrix} a_{1n} \\ \vdots \\ a_{in} \\ \vdots \\ a_{mn} \end{pmatrix}.$$

Thus, j-th column of C is a linear combination of the columns of matrix A. Sometimes it is useful to consider matrices partitioned into blocks. For example,

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

with $m_1 + m_2 = m$, $p_1 + p_2 = p$, and $n_1 + n_2 = n$. This time $C = AB$ can be expressed as

$$C = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{21} + A_{22}B_{21} & A_{12}B_{12} + A_{22}B_{22} \end{pmatrix}.$$

Example 3.1.
Let

$$A = \left(\begin{array}{cc|c} 1 & 2 & 3 \\ 4 & 5 & 6 \\ \hline 7 & 8 & 9 \end{array}\right) \quad \text{and} \quad B = \left(\begin{array}{c|c} 1 & 2 \\ 3 & 4 \\ \hline 5 & 6 \end{array}\right),$$

then $C = AB$ can be computed as

$$C = \left(\begin{array}{c|c} \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \end{pmatrix} + \begin{pmatrix} 3 \\ 6 \end{pmatrix} 5 & \begin{pmatrix} 1 & 2 \\ 4 & 5 \end{pmatrix} \begin{pmatrix} 2 \\ 4 \end{pmatrix} + \begin{pmatrix} 3 \\ 6 \end{pmatrix} 6 \\ \hline \begin{pmatrix} 7 & 8 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \end{pmatrix} + 9 \cdot 5 & \begin{pmatrix} 7 & 8 \end{pmatrix} \begin{pmatrix} 2 \\ 4 \end{pmatrix} + 9 \cdot 6 \end{array}\right) = \left(\begin{array}{cc|c} 22 & 28 \\ 49 & 64 \\ \hline 76 & 100 \end{array}\right).$$

This idea is key for the asymptotically faster matrix-matrix multiplication of celebrated **Strassen Algorithm** [2].

3.3 Existence of Uniqueness of solution of linear systems

Let $A \in \mathbb{R}^{m\times n}$ and $x \in \mathbb{R}^n$. Then from (3.2), Ax is a linear combination of the columns of A, i.e.

$$Ax = x_1 \begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{m1} \end{pmatrix} + x_2 \begin{pmatrix} a_{12} \\ a_{22} \\ \vdots \\ a_{m2} \end{pmatrix} + \cdots + x_n \begin{pmatrix} a_{1n} \\ a_{2n} \\ \vdots \\ a_{mn} \end{pmatrix}$$

Hence $Ax = b$ has a solution if $b \in \mathbb{R}^m$ can be written as a linear combination of the columns of A.

Solvability:
$Ax = b$ is solvable for every $b \in \mathbb{R}^m$ iff the columns of A span R^m (necessary $n \geq m$).
Uniqueness:
If $Ax = b$ has a solution, then the solution is unique iff the columns of A are linearly independent (necessary $n \leq m$).
Existence and Uniqueness:
For any $b \in \mathbb{R}^m$, the system $Ax = b$ has a unique solution iff $n = m$ and the columns of A are linearly independent.

3.4 Transpose of a Matrix

If $A \in \mathbb{R}^{m\times n}$ then $A^T \in \mathbb{R}^{n\times m}$ is obtained by reflecting the elements with respect to main diagonal. Thus, if $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times k}$, then

$$(AB)^T = B^T A^T.$$

More generally,

$$(A_1 A_2 \ldots A_j)^T = A_j^T \ldots A_2^T A_1^T.$$

If $A \in \mathbb{R}^{n\times n}$ is invertible, then A^T is invertible and

$$(A^T)^{-1} = (A^{-1})^T.$$

We will write A^{-T}.

3.5 Solution of Triangular Systems

Definition 3.1 (Lower triangular). A matrix $L \in \mathbb{R}^{n\times n}$ is called *lower triangular matrix* if all matrix entries above the diagonal are equal to zero, i.e., if

$$l_{ij} = 0 \quad \text{for} \quad j > i.$$

Definition 3.2 (Upper triangular). A matrix $U \in \mathbb{R}^{n\times n}$ is called *upper triangular matrix* if all matrix entries below the diagonal are equal to zero, i.e., if

$$u_{ij} = 0 \quad \text{for} \quad i > j.$$

A Linear system with lower (upper) triangular matrix can be solved by forward substitution (backward substitution).

Algorithm 3.1 (Solution of Upper Triangular Systems (Row-Oriented Version))

Input: *Upper triangular matrix* $U \in \mathbb{R}^{n\times n}$
right hand side vector $b \in \mathbb{R}^n$
Output: *Solution* $x \in \mathbb{R}^n$ *of* $Ux = b$
Mathematically,

$$x_i = \left(b_i - \sum_{j=i+1}^{n} u_{ij}x_j \right) / u_{ii}, \quad \textit{if} \quad u_{ii} \neq 0.$$

Matlab code:

```
if all(diag(u)) == 0
        disp('the matrix is singular')
else
    b(n) = b(n)/U(n,n);
    for i = n-1:-1:1
        b(i)= (b(i) - U(i,i+1:n)*b(i+1:n))/U(i,i);
    end
end
```

We can also put column oriented version

Algorithm 3.2 (Solution of Upper Triangular Systems (Column-Oriented Version))

Input: *Upper triangular matrix* $U \in \mathbb{R}^{n\times n}$
right hand side vector $b \in \mathbb{R}^n$
Output: *Solution* $x \in \mathbb{R}^n$ *of* $Ux = b$
Mathematically,

$$x_i = \left(b_i - \sum_{j=i+1}^{n} u_{ij}x_j \right) / u_{ii}, \quad \textit{if} \quad u_{ii} \neq 0.$$

MATLAB code that overwrites b with the solution to $Ux = b$.

```
if all(diag(u)) == 0
        disp('the matrix is singular')
else

    for  j = n:-1:2
           b(j) = b(j)/U(j,j) ;
           b(1:j-1) = b(1:j-1) - b(j)*U(1:j-1,j);
    end

    b(1) = b(1)/U(1,1);
end
```

3.6 Gaussian Elimination.

Gaussian elimination for the solution of a linear system transforms the system $Ax = b$ into an equivalent system with upper triangular matrix. This is done by applying three types of transformations to the augmented matrix $(A|b)$.

- Type 1: Replace an equation with the sum of the same equation and a multiple of another equation;
- Type 2: Interchange two equations; and
- Type 3: Multiply an equation by a nonzero number.

Once the augmented matrix $(A|b)$ is transformed into $(U|y)$, where U is an upper triangular matrix, we can use the techniques discussed previously to solve this transformed system $Ux = y$.

We need to modify Gaussian elimination for two reasons:

- improve numerical stability (change how we perform pivoting)
- make it more versatile (leads to LU-decomposition)

Definition 3.3 (Partial pivoting). In step i, find row j with $j > i$ such that $|a_{ji}| \geq |a_{ki}|$ for all $k > i$ and exchange rows i and j. Such numbers a_{ji} we call pivots.

Partial pivoting is not relevant in exact arithmetic. Without partial pivoting with floating point arithmetic the method can be unstable.

Using the formula

$$\sum_{j=1}^{n} j^2 = \frac{n(n+1)(2n+1)}{6},$$

we can calculate that for large n the number of flops in the Gaussian elimination with partial pivoting approximately equal to $2n^3/3$.

3.7 LU decomposition

The Gaussian elimination operates on the augmented matrix $(A \mid b)$ and performs two types of operations to transform $(A \mid b)$ into $(U \mid y)$, where U is upper triangular. However, the right hand side does not influence how the augmented matrix $(A \mid b)$ is transformed into an upper triangular matrix $(U \mid y)$. This transformation depends only on the matrix A. Thus, if we keep a record of how A is transformed into an upper triangular matrix U, then we can apply the same transformation to any right hand side, without re-applying the same transformations to A.

What operations are performed? In step k of the Gaussian elimination with partial pivoting we perform two operations:

- Interchange a row $i_0 > k$ with row k.
- Add a multiple $-l_{ik}$ times row k to row i for $i = k+1, \ldots, n$.

How do we record this?

- Introduce an integer array $ipivt$. Set
$$ipivt(k) = i_0$$
when the k-th step the rows k and i_0 are interchanged.
- Store the $-l_{ik}$ in the (i,k)-th position of the array that originally held A. (Remember that we add a multiple $-l_{ik}$ times row k to row i to eliminate the entry (i,k). Hence this storage can be reused.)

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & \cdots & a_{1,n-1} & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & \cdots & a_{2,n-1} & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & \cdots & a_{3,n-1} & a_{3n} \\ \vdots & \vdots & \vdots & & & \vdots & \vdots \\ a_{n-1,1} & a_{n-1,2} & a_{n-1,3} & \cdots & \cdots & a_{n-1,n-1} & a_{n-1,n} \\ a_{n1} & a_{n2} & a_{n3} & \cdots & \cdots & a_{n,n-1} & a_{nn} \end{pmatrix}$$

$\downarrow$

Step 1

$\downarrow$

$$\left(\begin{array}{c|cccc|c} a_{11} & a_{12} & a_{13} & \cdots & \cdots & a_{1,n-1} & a_{1n} \\ \hline -l_{21} & a_{22} & a_{23} & \cdots & \cdots & a_{2,n-1} & a_{2n} \\ -l_{31} & a_{32} & a_{33} & \cdots & \cdots & a_{3,n-1} & a_{3n} \\ \vdots & \vdots & \vdots & & & \vdots & \vdots \\ -l_{n-1,1} & a_{n-1,2} & a_{n-1,3} & \cdots & \cdots & a_{n-1,n-1} & a_{n-1,n} \\ -l_{n1} & a_{n2} & a_{n3} & \cdots & \cdots & a_{n,n-1} & a_{nn} \end{array}\right)$$

$\downarrow$

Step 2

$\downarrow$

$$\left(\begin{array}{c|c|ccc|c} a_{11} & a_{12} & a_{13} & \cdots & \cdots & a_{1,n-1} & a_{1n} \\ \hline -l_{21} & a_{22} & a_{23} & \cdots & \cdots & a_{2,n-1} & a_{2n} \\ \hline -l_{31} & -l_{32} & a_{33} & \cdots & \cdots & a_{3,n-1} & a_{3n} \\ \vdots & \vdots & \vdots & & & \vdots & \vdots \\ -l_{n-1,1} & -l_{n-1,2} & a_{n-1,3} & \cdots & \cdots & a_{n-1,n-1} & a_{n-1,n} \\ -l_{n1} & -l_{n2} & a_{n3} & \cdots & \cdots & a_{n,n-1} & a_{nn} \end{array}\right)$$

$\downarrow$

Step n-1

$\downarrow$

$$\left(\begin{array}{c|c|c|ccc|c} a_{11} & a_{12} & a_{13} & \cdots & \cdots & a_{1,n-1} & a_{1n} \\ \hline -l_{21} & a_{22} & a_{23} & \cdots & \cdots & a_{2,n-1} & a_{2n} \\ \hline -l_{31} & -l_{32} & a_{33} & \cdots & \cdots & a_{3,n-1} & a_{3n} \\ \hline \vdots & \vdots & \vdots & \ddots & & \vdots & \vdots \\ \vdots & \vdots & \vdots & & \ddots & a_{n-1,n-1} & a_{n-1,n} \\ -l_{n1} & -l_{n2} & -l_{n3} & \cdots & \cdots & -l_{n,n-1} & a_{nn} \end{array}\right)$$

Row interchange in step k can be expressed by multiplying with

$$P_k = \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & 0 & & 1 & & \\ & & & & 1 & & & \\ & & & & & \ddots & & \\ & & & & & & 1 & \\ & & & 1 & & & & \\ & & & & & & & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ \leftarrow k \\ \\ \\ \\ \leftarrow ipivt(k) \\ \\ \end{matrix}$$

$$\begin{matrix} & & \uparrow & & \uparrow & & \\ & & k & & ipivt(k) & & \end{matrix}$$

from the left. P_k is a permutation matrix.

Easy to see that P_k satisfies $P_k = P_k^T$ and $P_k^2 = Id$ and as a result $P_k^{-1} = P_k$. Furthermore, $P_k A$ interchanges rows k and $ipivt(k)$ of A, but AP_k interchanges columns k and $ipivt(k)$ of A.

Adding $-l_{i,k}$ times row k to row i for $i = k+1,\ldots,n$ is equivalent to multiplying from the left with

$$M_k = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -l_{k+1,k} & & \\ & & \vdots & \ddots & \\ & & -l_{n,k} & & 1 \end{pmatrix}$$

Observe that matrix M_k is lower triangular and is called a Gauss transformation. Easy to see that M_k is invertible and

$$M_k^{-1} = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & l_{k+1,k} & & \\ & & \vdots & \ddots & \\ & & l_{n,k} & & 1 \end{pmatrix}$$

Furthermore, product of two lower triangular matrices results in a lower triangular matrix, i.e. $M_k M_j$ is again lower triangular

The transformation of a matrix A into an upper triangular matrix U can be expressed as a sequence of matrix-matrix multiplications

$$M_{n-1}P_{n-1}\ldots M_1 P_1 A = U.$$

Above we have observed that P_k and M_k are invertible. Hence, if we have to solve $Ax = b$ and if

$$M_{n-1}P_{n-1}\ldots M_1 P_1 A = U,$$

then we apply the matrices to the right hand side,

$$M_{n-1}P_{n-1}\dots M_1P_1b = y$$

and solve

$$Ux = y.$$

We observe that for $j > k$

$$P_jM_k = \tilde{M}_kP_j, \tag{3.3}$$

where $\tilde{M}_k$ is obtained from M_k by interchanging the entries $-l_{j,k}$ and $-l_{ipivt(j),k}$. $\tilde{M}_k$ has the same structure as M_k and we can easily compute $\tilde{M}_k^{-1}$. Using (3.3) we can move all P_j's to the right of the $\tilde{M}_k$'s

$$U = M_{n-1}P_{n-1}\dots M_1P_1A = M_{n-1}\tilde{M}_{n-2}\dots\tilde{M}_1P_{n-1}\dots P_1A.$$

Thus $P = P_{n-1}\dots P_1$ is the permutation matrix and $L = \tilde{M}_1^{-1}\dots\tilde{M}_{n-2}^{-1}M_{n-1}^{-1}$ is a lower triangular matrix with diagonal elements equal to 1. Matlab's function $[L,U,P] = lu(A)$ computes matrices P,L,U such that $PA = LU$.

Example 3.2. Consider the matrix

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

After step $k = 1$ the vector *ipivt* and the matrix A are given by

$$ipivt = (2,\ ,\ ) \qquad \left(\begin{array}{c|cc} 1 & 2 & -1 \\ \hline -\frac{1}{2} & \frac{3}{2} & -1 \\ \frac{1}{2} & \frac{5}{2} & 2 \end{array}\right)$$

After step $k = 2$ the vector *ipivt* and the matrix A are given by

$$ipivt = (2,3,\ ) \qquad \left(\begin{array}{c|c|c} 1 & 2 & -1 \\ \hline -\frac{1}{2} & \frac{5}{2} & 2 \\ \hline \frac{1}{2} & -\frac{3}{5} & -\frac{11}{5} \end{array}\right)$$

If we want to solve the linear system $Ax = b$ with

$$b = \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix}$$

We have to apply the same transformation to b

$$\begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix} \xrightarrow{\text{step 1}} \begin{pmatrix} 2 \\ -1 \\ 2 \end{pmatrix} \xrightarrow{\text{step 2}} \begin{pmatrix} 2 \\ 2 \\ -11/5 \end{pmatrix}$$

and then solve

$$\begin{pmatrix} 1 & 2 & -1 \\ 0 & \frac{5}{2} & 2 \\ 0 & 0 & -\frac{11}{5} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ -11/5 \end{pmatrix}$$

by back substitution to obtain $x_1 = 1, x_2 = 0, x_3 = 1$.

Using matrices P_i and M_i, same steps we can write as

Step 1:

$$P_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad P_1 A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 2 & -1 \\ -1 & 2 & 2 \end{pmatrix}$$

$$M_1 = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ \frac{1}{2} & 0 & 1 \end{pmatrix}, \quad M_1 P_1 A = \begin{pmatrix} 2 & 1 & 0 \\ 0 & \frac{3}{2} & -1 \\ 0 & \frac{5}{2} & 2 \end{pmatrix}$$

Step 2:

$$P_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad P_2 M_1 P_1 A = \begin{pmatrix} 2 & 1 & 0 \\ 0 & \frac{5}{2} & 2 \\ -1 & \frac{3}{2} & -1 \end{pmatrix}$$

$$M_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{3}{5} & 1 \end{pmatrix}, \quad M_2 P_2 M_1 P_1 A = \begin{pmatrix} 2 & 1 & 0 \\ 0 & \frac{5}{2} & 2 \\ 0 & 0 & -\frac{11}{5} \end{pmatrix}$$

For $b = (0,2,1)^T$ to solve $Ax = b$, we compute

$$M_2 P_2 M_1 P_1 b = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{3}{5} & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ -\frac{11}{5} \end{pmatrix}$$

Solving

$$\begin{pmatrix} 2 & 1 & 0 \\ 0 & \frac{5}{2} & 2 \\ 0 & 0 & -\frac{11}{5} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ -\frac{11}{5} \end{pmatrix}$$

we obtain $x = (1,0,1)^T$.

To obtain LU decomposition, we have

$$U = \begin{pmatrix} 2 & 1 & 0 \\ 0 & \frac{5}{2} & 2 \\ 0 & 0 & -\frac{11}{5} \end{pmatrix} = M_2 P_2 M_1 P_1 A.$$

Then using

$$P_2 M_1 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ \frac{1}{2} & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ -\frac{1}{2} & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} = \tilde{M}_1 P_2$$

we have

$$U = M_2 \tilde{M}_1 P_2 P_1 A.$$

Calling

$$L = \tilde{M}_1^{-1} M_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ \frac{1}{2} & \frac{3}{5} & 1 \end{pmatrix} \quad \text{and} \quad P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

we have $PA = LU$, i.e.

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 2 & -1 \\ 2 & 1 & 0 \\ -1 & 2 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{2} & 1 & 0 \\ \frac{1}{2} & \frac{3}{5} & 1 \end{pmatrix} \begin{pmatrix} 2 & 1 & 0 \\ 0 & \frac{5}{2} & 2 \\ 0 & 0 & -\frac{11}{5} \end{pmatrix}.$$

3.8 Applications of LU decomposition

LU-decomposition is very useful one and using it we can compute many quantities rather efficiently.

3.8.1 Solving Linear System

To solve the linear system $Ax = b$ using LU decomposition $PA = LU$, we need to solve two triangular systems, $Ly = Pb$ and $Ux = y$. It especially beneficial if we need to solve $Ax = bn$ for many right hand sides, as for computing A^{-1}.

3.8.2 Finding Inverse A^{-1}

Computing A^{-1} is rarely requited. Usually we need to compute $A^{-1}b$, which means we need to solve a linear system $Ax = b$. In the rare occasion when the explicit form of A^{-1} is needed, we can use LU-decomposition to find it by using $O(n^3)$ operations. Recall that A^{-1} is a unique matrix X such that

$$AX = I,$$

where I is the identity matrix. Denote the columns of the matrix X be x_i, $i = 1,\ldots,n$. Thus in column notation the above equation is equivalent to

$$[Ax_1, Ax_2, \ldots, Ax_n] = [e_1, e_2, \ldots, e_n].$$

In other words we need to solve n equations

$$Ax_i = e_i, \quad i = 1,\ldots,n.$$

In section 3.8.1 above we explained, how we can solve it using LU decomposition. Notice, LU decomposition takes $O(\frac{2n^3}{3})$ operations solving triangular systems $O(n^2)$ operations, and since we need to solve n of them it take $O(n^3)$ operations to compute A^{-1} explicitly.

3.8.3 Computing detetminates

Recall that if $A, B, C \in \mathbb{R}^{n\times n}$ and $C = AB$, then

$$det(C) = det(A)det(B).$$

Assume we have LU decomposition of A, i.e. $PA = LU$. Then

$$det(P)det(A) = det(L)det(U).$$

Since P is a permutation matrix $det(P) = \pm 1$. More, precisely, 1 if the permutation is even, and -1 if it is odd. Since L and U are triangular, their determinates are just product of diagonal elements. Thus, $det(L) = 1$ and $det(U) = \prod_{i=1}^{n} u_{ii}$. As a result

$$det(A) = \pm \prod_{i=1}^{n} u_{ii}.$$

3.9 Symmetric Positive definite matrices. Cholesky decomposition.

Definition 3.4. $A \in \mathbb{R}^{n\times n}$ is called symmetric if $A = A^T$.

Definition 3.5. $A \in \mathbb{R}^{n\times n}$ is called symmetric positive definite if $A = A^T$ and $v^T Av > 0$ for all $v \in \mathbb{R}^n$, $v \neq 0$.

If $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite, then the LU decomposition can be computed in a stable way without permutation, i.e.,

$$A = LU$$

and more efficiently.

First we notice that if L is a unit lower triangular matrix and U is an upper triangular matrix such that $A = LU$, then L and U are unique, i.e., if there $\tilde{L}$ is a unit lower triangular matrix and $\tilde{U}$ is an upper triangular matrix such that $A = \tilde{L}\tilde{U}$, then $L = \tilde{L}$ and $U = \tilde{U}$. Furthermore, if $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite and $A = LU$, then the diagonal entries of U are positive and we can write it as $U = D\tilde{U}$, where $\tilde{U}$ is unit upper triangular and

$$D = diag(u_{11}, \ldots, u_{nn}) \quad \text{with} \quad u_{ii} > 0, \quad i = 1, \ldots, n.$$

Thus,

$$A = LU = LD\tilde{U}.$$

On the other hand

$$A^T = (LD\tilde{U})^T = \tilde{U}^T DL^T.$$

Using that $A = A^T$ and LU decomposition is unique

$$A = LU = \tilde{U}^T DL^T = \tilde{U}^T (DL^T) = \text{(lower unit triangular)} \times \text{(upper triangular)}.$$

Thus,

$$L = \tilde{U}^T \quad \text{and} \quad U = DL^T.$$

We showed that if A is a symmetric positive definite matrix, then

$$A = LDL^T.$$

Recall that

$$D = diag(u_{11}, \ldots, u_{nn})$$

has positive diagonal entries. So we can define

$$D^{1/2} = diag(\sqrt{u_{11}}, \ldots \sqrt{u_{nn}}).$$

Define $R := D^{1/2}L^T$, then

$$A = LDL^T = LD^{1/2}D^{1/2}L^T = R^T R: \qquad \text{Cholesky-decomposition}$$

Matlab's function $[R] = chol(A)$ computes the matrix R such that $A = R^T R$. By clever implementation, one can show that the total number of basic operations is of order $n^3/3$, comparing to $2n^3/3$ for LU decomposition.

3.10 Tridiagonal matrices

Definition 3.6. $A \in \mathbb{R}^{n\times n}$ is called m- banded if $a_{ij} = 0$ for $|i - j| > m$.

Definition 3.7. $A \in \mathbb{R}^{n\times n}$ is called tridiagonal if it is 1- banded.

Let

$$A = \begin{pmatrix} d_1 & e_1 & & & \\ c_1 & d_2 & e_2 & & \\ & \ddots & \ddots & \ddots & \\ & & c_{n-2} & d_{n-1} & e_{n-1} \\ & & & c_{n-1} & d_n \end{pmatrix}$$

Since there are only about $3n$ entries in matrix A, a natural question, how can we compute the LU-decomposition of A efficiently? Let's look at LU decomposition of A

$$\underbrace{\begin{pmatrix} 1 & & & & \\ -\frac{c_1}{d_1} & 1 & & & \\ 0 & 0 & \ddots & & \\ & & \ddots & 1 & \\ & & & 0 & 1 \end{pmatrix}}_{=M_1}, \quad \underbrace{\begin{pmatrix} d_1 & e_1 & & & \\ & \tilde{d}_2 & e_2 & & \\ & c_2 & d_3 & e_3 & \\ & & \ddots & \ddots & \ddots \\ & & & c_{n-2} & d_{n-1} & e_{n-1} \\ & & & & c_{n-1} & d_n \end{pmatrix}}_{=M_1A, \quad \text{where} \quad \tilde{d}_2 = d_2 - \frac{c_1}{d_1}e_1}$$

$$\underbrace{\begin{pmatrix} 1 & & & & \\ 0 & 1 & & & \\ & -\frac{c_2}{d_2} & 1 & & \\ & 0 & 0 & \ddots & \\ & & & \ddots & 1 \\ & & & & 0 & 1 \end{pmatrix}}_{=M_2}, \quad \underbrace{\begin{pmatrix} d_1 & e_1 & & & \\ & \tilde{d}_2 & e_2 & & \\ & & \tilde{d}_3 & e_3 & \\ & & c_3 & d_4 & e_4 \\ & & & \ddots & \ddots & \ddots \\ & & & & c_{n-2} & d_{n-1} & e_{n-1} \\ & & & & & c_{n-1} & d_n \end{pmatrix}}_{=M_2M_1A, \quad \text{where} \quad \tilde{d}_3 = d_3 - \frac{c_2}{\tilde{d}_2}e_2}$$

After $n-1$ steps we obtain

$$A = \begin{pmatrix} 1 & & & & \\ \frac{c_1}{d_1} & 1 & & & \\ & \frac{c_2}{d_2} & 1 & & \\ & & \ddots & \ddots & \\ & & & \frac{c_{n-1}}{d_{n-1}} & 1 \end{pmatrix} \begin{pmatrix} d_1 & e_1 & & \\ & \tilde{d}_2 & e_2 & \\ & & \ddots & \ddots \\ & & & \tilde{d}_{n-1} & e_{n-1} \\ & & & & \tilde{d}_n \end{pmatrix},$$

where $\tilde{d_{i+1}} = d_{i+1} - \frac{c_i}{\tilde{d}_i}e_i$, for $i = 1 : n-1$.

Algorithm 3.3 *Input: Vectors c,d,e that form tridiagonal matrix A*
Output: vectors c,d that form LU-decomposition of A

1. *for* $k = 1 : n-1$
2. *If* $d_k = 0$, *stop*
3. $c_k = c_k/d_k$
4. $d_{k+1} = d_{k+1} - c_k e_k$
5. *end*

This algorithm requires about $3n$ flops.

Given three arrays of length n, $c = [c_1;\ldots;c_n]$, $d = [d_1;\ldots;d_n]$; $e = [e_1;\ldots;e_n]$.

The Matlab command

$$A = spdiags([c,d,e],-1:1,n,n);$$

generates a sparse form of the matrix

$$A = \begin{pmatrix} d_1 & e_2 & & & \\ c_1 & d_2 & e_3 & & \\ & \ddots & \ddots & \ddots & \\ & & c_{n-2} & d_{n-1} & e_n \\ & & & c_{n-1} & d_n \end{pmatrix}$$

3.11 Error analysis of Linear systems

3.11.1 Vector Norms

We remind that a norm $\|\cdot\|$ on a vector space V over $\mathbb{R}$ is a function $\|\cdot\| : V \to \mathbb{R}^+$ that satisfies the following properties

1. **(Positivity)** $\|\mathbf{x}\| > 0$ for any non-zero $\mathbf{x} \in V$.
2. **(Scalability)** $\|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\|$, for any $\alpha \in \mathbb{R}$.
3. **(Triangle inequalit)** $\|\mathbf{x}+\mathbf{y}\| \le \|\mathbf{x}\| + \|\mathbf{y}\|$ for any $x, y \in V$

As a consequence of the triangle inequality, we have the for following inequality.

$$\|\mathbf{x}+\mathbf{y}\| \ge \big|\|\mathbf{x}\| - \|\mathbf{y}\|\big| \quad \text{for all} \quad \mathbf{x},\mathbf{y} \in \mathbb{R}^n.$$

The Euclidian or 2-norm on $\mathbb{R}^n$ is given by

$$\|\mathbf{x}\|_2 = \left(\sum_{i=1}^{n} x_i^2\right)^{1/2}, \quad \text{2-norm}$$

More generally for any $p \in [1,\infty)$ we can define p-norm

$$\|\mathbf{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}.$$

The infinity norm is also often is used

$$\|\mathbf{x}\|_\infty = \max_{i=1,\ldots,n} |x_i|.$$

Example 3.3. Let $\mathbf{x} = (1,-2,3,-4)^T$. Then

$$\|\mathbf{x}\|_1 = 1+2+3+4 = 10,$$
$$\|\mathbf{x}\|_2 = \sqrt{1+4+9+16} = \sqrt{30} \approx 5.48,$$
$$\|\mathbf{x}\|_\infty = \max\{1,2,3,4\} = 4.$$

The unit ball with respect to p-norm is defined by

$$\{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_p \le 1\}.$$

TO ADD PICTURE
The following inequalities hold:

$$\|\mathbf{x}\|_\infty \le \|\mathbf{x}\|_2 \le \|\mathbf{x}\|_1.$$

Theorem 3.4. *All vector norms on $\mathbb{R}^n$ are equivalent, i.e. for every two vector norms $\|\cdot\|_a$ and $\|\cdot\|_b$ on $\mathbb{R}^n$ there exist constants c_{ab}, C_{ab} (depending on the vector norms $\|\cdot\|_a$ and $\|\cdot\|_b$, but not on x) such that*

$$c_{ab}\|\mathbf{x}\|_b \le \|\mathbf{x}\|_a \le C_{ab}\|\mathbf{x}\|_b \quad \forall \mathbf{x} \in \mathbb{R}^n.$$

Proof. It is sufficient to show any norm $\|\cdot\|$ is equivalent to infinity norm $\|\cdot\|_\infty$.

Let $\mathbf{e}_1, \dots, \mathbf{e}_n$ be the standard basis for $\mathbb{R}^n$. Then any $x \in \mathbb{R}^n$ can be written as

$$\mathbf{x} = x_1\mathbf{e}_1 + \cdots + x_n\mathbf{e}_n.$$

It follows that

$$\|\mathbf{x}\| \le \|\mathbf{x}\|_\infty \sum_{j=1}^{n} \|\mathbf{e}_j\| \le \gamma\|\mathbf{x}\|_\infty, \quad \text{with} \quad \gamma = \sum_{j=1}^{n} \|\mathbf{e}_j\|.$$

This shows that any norm $\|\cdot\|$ on $\mathbb{R}^n$ is continuous with respect to $\|\cdot\|_\infty$ norm.

Consider the set

$$S = \{\mathbf{y} \in \mathbb{R}^n : \ \|\mathbf{y}\|_\infty = 1\}.$$

Thus the set S is bounded closed set in $\mathbb{R}^n$ and as a result compact. On a compact set a function $\|\cdot\|$ attains its maximum and minimum values. So there exist $vecy_0$ and $\mathbf{y}_1$ such that

$$0 < \|\mathbf{y}_0\| \le \|\mathbf{y}\| \le \|\mathbf{y}_1\| < \infty, \quad \forall \mathbf{y} \in S.$$

For any $\mathbf{x} \neq 0$ consider $\mathbf{y} = \frac{\mathbf{x}}{\|\mathbf{x}\|_\infty} \in S$. Then

$$\|\mathbf{y}_0\| \le \frac{\mathbf{x}}{\|\mathbf{x}\|_\infty} \le \|\mathbf{y}_1\|,$$

which shows

$$m\|\mathbf{y}\|_\infty \le \|\mathbf{y}\|_\infty \le M\|\mathbf{y}\|_\infty,$$

with $m = \|\mathbf{y}_0\|$ and $M = \|\mathbf{y}_1\|$.

Remark 3.1. Although all vector norm are equivalent, they are not equivalent with respect to the dimensions n. For example, for $\mathbf{1} = (1, \dots, 1)^T$ we have

$$\|\mathbf{1}\|_\infty = 1, \quad \|\mathbf{1}\|_2 = \sqrt{n}, \quad \|\mathbf{1}\|_1 = n.$$

In particular, for any $\mathbf{x} \in \mathbb{R}^n$ we have the inequalities

$$\frac{1}{\sqrt{n}}\|\mathbf{x}\|_1 \le \|\mathbf{x}\|_2 \le \|\mathbf{x}\|_1$$
$$\|\mathbf{x}\|_\infty \le \|\mathbf{x}\|_2 \le \sqrt{n}\|\mathbf{x}\|_\infty$$
$$\|\mathbf{x}\|_\infty \le \|\mathbf{x}\|_1 \le n\|\mathbf{x}\|_\infty.$$

3.11.2 Matrix norm

We can view a matrix $A \in \mathbb{R}^{m\times n}$ as a vector in $\mathbb{R}^{mn}$, by stacking the columns of the matrix into a long vector. Appling the 2-vector norm to this vectors of length mn, we obtain the **Frobenius norm**,

$$\|A\|_F = \left(\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}^2\right)^{1/2}.$$

In many situations the Frobenius norm is not convenient. One of the reasons is that for the identity matrix $I \in \mathbb{R}^{n\times n}$, $\|I\|_F = \sqrt{n}$.

Another approach is to view a matrix $A \in \mathbb{R}^{m\times n}$ as a linear mapping, which maps a vector $\mathbf{x} \in \mathbb{R}^n$ into a vector $A\mathbf{x} \in \mathbb{R}^m$

$$A:\ \mathbb{R}^n \to \mathbb{R}^m$$
$$\mathbf{x} \to A\mathbf{x}.$$

To define the size of this linear mapping, we compare the size of the image $A\mathbf{x} \in \mathbb{R}^m$ with the size of $\mathbf{x}$. This leads us to look at

$$\sup_{\mathbf{x}\neq\mathbf{0}} \frac{\|A\mathbf{x}\|}{\|\mathbf{x}\|}$$

Here $A\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{x} \in \mathbb{R}^n$ are vectors and $\|\cdot\|$ are vector norms (in $\mathbb{R}^m$ and $\mathbb{R}^n$).

Definition 3.8 (Matrix p-norm).

$$\|A\|_p = \max_{\mathbf{x}\neq 0} \frac{\|A\mathbf{x}\|_p}{\|\mathbf{x}\|_p}, \quad 1 \le p \le \infty. \tag{3.4}$$

The following holds,

$$\sup_{\mathbf{x}\neq\mathbf{0}} \frac{\|A\mathbf{x}\|_p}{\|\mathbf{x}\|_p}\mathbf{x}\|_p = \max_{\mathbf{x}\neq\mathbf{0}} \frac{\|A\mathbf{x}\|_p}{\|\mathbf{x}\|_p} = \max_{\|\mathbf{x}\|_p=1} \|A\mathbf{x}\|_p.$$

With the above definition, now for the identity matrix I,

$$\|I\|_p = \max_{\mathbf{x}\neq\mathbf{0}} \frac{\|I\mathbf{x}\|_p}{\|\mathbf{x}\|_p} = 1.$$

Two important inequalities.

Theorem 3.5. *For any $A \in \mathbb{R}^{m\times n}$, $B \in \mathbb{R}^{n\times k}$ and $\mathbf{x} \in \mathbb{R}^n$, the following inequalities hold.*

$$\|A\mathbf{x}\|_p \le \|A\|_p\|\mathbf{x}\|_p \quad \textit{(compatibility of matrix and vector norm)}$$

and

$$\|AB\|_p \le \|A\|_p\|B\|_p \quad \textit{(submultiplicativity of matrix norms)}$$

Proof. The first statement follows directly from the definition of the p matrix norm. The second statement follows that for $B \neq 0$

$$\|AB\|_p = \sup_{\mathbf{x}\neq\mathbf{0}} \frac{\|(AB)\mathbf{x}\|_p}{\|\mathbf{x}\|_p} = \sup_{\mathbf{x}\neq\mathbf{0}} \frac{\|A(B\mathbf{x})\|_p \|B\mathbf{x}\|_p}{\|B\mathbf{x}\|_p \|\mathbf{x}\|_p} = \sup_{\mathbf{y}\neq\mathbf{0}} \frac{\|A\mathbf{y}\|_p}{\|\mathbf{y}\|_p} \sup_{\mathbf{x}\neq\mathbf{0}} \frac{\|B\mathbf{x}\|_p}{\|\mathbf{x}\|_p} = \|A\|_p \|B\|_p.$$

The case $B = 0$ holds trivially.

For the most commonly used matrix-norms (3.4) with $p = 1$, $p = 2$, or $p = \infty$, there exist rather simple representations.

Theorem 3.6. *Let $\|\cdot\|_p$ be the matrix norm defined in* (3.4)*, then*

$$\|A\|_1 = \max_{1\leq j\leq n} \sum_{i=1}^{m} |a_{ij}| \quad \textit{(maximum column norm)};$$

$$\|A\|_\infty = \max_{1\leq i\leq m} \sum_{j=1}^{n} |a_{ij}| \quad \textit{(maximum row norm)};$$

$$\|A\|_2 = \sqrt{\lambda_{max}(A^T A)} \quad \textit{(spectral norm)},$$

where $\lambda_{max}(A^T A)$ is the largest eigenvalue of $A^T A$.

Proof. We will show the result for the 1-norm. The rest we will leave as an exercise. We have

$$\begin{aligned}
\|A\mathbf{x}\|_1 &= \sum_{i=1}^{m} \left| \sum_{j=1}^{n} a_{ij} x_j \right| \leq \sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}||x_j| \\
&= \sum_{j=1}^{n} |x_j| \sum_{i=1}^{m} |a_{ij}| \leq \sum_{j=1}^{n} |x_j| \max_{1\leq k\leq n} \sum_{i=1}^{m} |a_{ik}| \\
&\leq \|\mathbf{x}\|_1 \max_{1\leq k\leq n} \sum_{i=1}^{m} |a_{ik}|,
\end{aligned}$$

and as a result

$$\|A\|_1 \leq \max_{1\leq k\leq n} \sum_{i=1}^{m} |a_{ik}|.$$

To the equality, it is sufficient to construct $\mathbf{x}_0$ with $\|\mathbf{x}_0\|_1 = 1$ such that

$$\|A\mathbf{x}_0\|_1 = \max_{1\leq k\leq n} \sum_{i=1}^{m} |a_{ik}|.$$

Let j_0 be such that

$$|a_{ij_0}| = \max_{1\leq k\leq n} \sum_{i=1}^{m} |a_{ik}|.$$

Then,

$$\|A\mathbf{e}_{j0}\|_1 = \sum_{i=1}^{m} |a_{ij_0}| = \max_{1\leq k\leq n} \sum_{i=1}^{m} |a_{ik}|.$$

That gives us the result.

From the above theorem, we immediately obtain the following results.

Corollary 3.1. *For any matrix $A \in \mathbb{R}^{m\times n}$, we have*

$$\|A\|_1 = \|A^T\|_\infty$$
$$\|A\|_2 = \|A^T\|_2.$$

In the case of symmetric matrices we can show sharper results.

Corollary 3.2. *For any symmetric matrix $A \in \mathbb{R}^{n\times n}$, we have*

$$\|A\|_1 = \|A\|_\infty = \max_{1\le j\le n} \sum_{i=1}^{n} |a_{ij}|$$
$$\|A\|_2 = \max_{1\le i\le n} |\lambda_i(A)|,$$

where $\lambda_i(A)$ denotes the i-th eigenvalue of A.

3.11.3 Error analysis

The linear systems

$$A\mathbf{x} = \mathbf{b}, \tag{3.5}$$

where $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$, usually come from some applications, where we do not know the exact values of A and $\mathbf{b}$. Instead we are often face with the perturbed system

$$(A + \Delta A)\tilde{\mathbf{x}} = \mathbf{b} + \Delta\mathbf{b}, \tag{3.6}$$

where $\Delta A \in \mathbb{R}^{n\times n}$ and $\Delta\mathbf{b} \in \mathbb{R}^n$ represent the perturbations in A and $\mathbf{b}$, respectively. The main question we are faced, what is the error $\Delta\mathbf{x} = \tilde{\mathbf{x}} - \mathbf{x}$ between the solution $\mathbf{x}$ of the exact linear system (3.5) and the solution ex perturbed linear system (3.10). For the simplicity of the representation, let's us first consider the case when A is exact.

Theorem 3.7. *Consider the perturbed system* (3.10) *with $\Delta A = 0$. Then the relative error*

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \le \|A\|\|A^{-1}\|\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|},$$

where $\|\cdot\|$ is any p-norm.

Proof. Using a representation

$$\tilde{\mathbf{x}} = \mathbf{x} + \Delta\mathbf{x},$$

from

$$A(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{b} + \Delta\mathbf{b},$$

since $A\mathbf{x} = \mathbf{b}$, we get

$$A\Delta\mathbf{x} = \Delta\mathbf{b}, \quad \text{or} \quad \Delta\mathbf{x} = A^{-1}\Delta\mathbf{b}.$$

Taking norms, we obtain

$$\|\Delta\mathbf{x}\| = \|A^{-1}\Delta\mathbf{b}\| \le \|A^{-1}\|\|\Delta\mathbf{b}\|. \tag{3.7}$$

Since $A\mathbf{x} = \mathbf{b}$,

$$\|\mathbf{b}\| = \|A\mathbf{x}\| \le \|A\|\|\mathbf{x}\| \quad \Rightarrow \quad \frac{1}{\|\mathbf{x}\|} \le \|A\|\frac{1}{\|\mathbf{b}\|}. \tag{3.8}$$

Combining (3.7) and (3.8) we get

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \le \|A\|\|A^{-1}\|\frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|}. \tag{3.9}$$

Definition 3.9. The (p-) condition number $\kappa_p(A)$ of a matrix A (with respect to inversion) is defined by

$$\kappa_p(A) = \|A\|_p\|A^{-1}\|_p.$$

Set $\kappa_p(A) = \infty$ is A is not invertible.

Notice that $\kappa_p(A) \geq 1$, since

$$1 = \|I\|_p = \|AA^{-1}\|_p \leq \|A\|_p \|A^{-1}\|_p = \kappa_p(A).$$

Definition 3.10. If $\kappa_p(A)$ is small, we say that the linear system is **well conditioned.**
Otherwise, we say that the linear system is **ill conditioned.**

To obtain similar result for the fully perturbed system we need the following auxiliary result.

Lemma 3.1. *Let $B \in \mathbb{R}^{n\times n}$ be arbitrary with $\|B\| < 1$, where $\|\cdot\|$ denotes any p matrix norm. Then $I+B$ is invertible and*

$$\|(I+B)^{-1}\| \leq \frac{1}{1-\|B\|}.$$

Proof. Since by the triangle inequality and the assumption of the lemma, for any $\mathbf{x} \neq \mathbf{0}$,

$$\|(I+B)\mathbf{x}\| = \|\mathbf{x}+B\mathbf{x}\| \geq \|\mathbf{x}\| - \|B\mathbf{x}\| = \|\mathbf{x}\|(1-\|B\|) > 0,$$

it shows that $(I+B)$ is invertible. Denote $C = (I+B)^{-1}$. Then

$$1 = \|I\| = \|(I+B)C\| = \|C+BC\| \geq \|C\| - \|CB\| \geq \|C\| - \|C\|\|B\| = \|C\|(1-\|B\|),$$

which give us the lemma.

Theorem 3.8. *Let*

$$(A+\Delta A)(\Delta\mathbf{x}+\mathbf{x}) = \mathbf{b}+\Delta\mathbf{b} \tag{3.10}$$

be the perturbed system, where $\Delta A \in \mathbb{R}^{n\times n}$ and $\Delta\mathbf{b} \in \mathbb{R}^n$ represent the perturbations in A and $\mathbf{b}$, respectively. If $\|A^{-1}\|_p\|\Delta A\|_p < 1$, then

$$\frac{\|\Delta\mathbf{x}\|_p}{\|\mathbf{x}\|_p} \leq \frac{\kappa_p(A)}{1-\kappa_p(A)\frac{\|\Delta A\|_p}{\|A\|_p}}\left(\frac{\|\Delta A\|_p}{\|A\|_p} + \frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|}\right). \tag{3.11}$$

Proof. First of all notice that since $\|A^{-1}\|_p\|\Delta A\|_p < 1$, we have

$$(A+\Delta A) = A(I+A^{-1}\Delta A).$$

Thus using the previous lemma, $(A+\Delta A)$ is a product of two invertible matrices A and $(I+A^{-1}\Delta A)$, hence invertible. Since

$$\Delta\mathbf{x} = \tilde{\mathbf{x}} - \mathbf{x},$$

we have

$$(A+\Delta A)\Delta\mathbf{x} = (A+\Delta A)\tilde{\mathbf{x}} - (A+\Delta A)\mathbf{x} = (\mathbf{b}+\Delta\mathbf{b}) - (\mathbf{b}+\Delta A\mathbf{x}) = \Delta\mathbf{b} - \Delta A\mathbf{x}.$$

Hence

$$\|\Delta\mathbf{x}\| \leq \|(A+\Delta A)^{-1}(\Delta\mathbf{b}-\Delta A\mathbf{x})\| \leq \|(A+\Delta A)^{-1}\|\left(\|\Delta\mathbf{b}\| + \|\Delta A\mathbf{x}\|\right).$$

Using the Lemma 3.1 with $B = A^{-1}\Delta A$, we have

$$\|(A+\Delta A)^{-1}\| = \|(A(I+A^{-1}\Delta A))^{-1}\| = \|((I+A^{-1}\Delta A)^{-1}A^{-1}\| \leq \frac{\|A^{-1}\|}{1-\|A^{-1}\|\|\Delta A\|}.$$

As a result

$$\|\Delta \mathbf{x}\| \leq \frac{\|A^{-1}\|}{1-\|A^{-1}\|\|\Delta A\|}(\|\Delta \mathbf{b}\| + \|\Delta A\|\|\mathbf{x}\|).$$

Using that $\|\mathbf{b}\| \leq \|A\mathbf{x}\| \leq \|A\|\|\mathbf{x}\|$, we finally obtain

$$\begin{aligned}\frac{\|\Delta \mathbf{x}\|}{\|\mathbf{x}\|} &\leq \frac{\|A^{-1}\|}{1-\|A^{-1}\|\|\Delta A\|}\left(\frac{\|\Delta \mathbf{b}\|}{\|\mathbf{x}\|} + \frac{\|\Delta A\|\|\mathbf{x}\|}{\|\mathbf{x}\|}\right)\\ &\leq \frac{\|A^{-1}\|\|A\|}{1-\|A^{-1}\|\|\Delta A\|}\left(\frac{\|\Delta \mathbf{b}\|}{\|\mathbf{b}\|} + \frac{\|\Delta A\|}{\|A\|}\right)\\ &\leq \frac{\kappa(A)}{1-\kappa(A)\frac{\|\Delta A\|}{\|A\|}}\left(\frac{\|\Delta \mathbf{b}\|}{\|\mathbf{b}\|} + \frac{\|\Delta A\|}{\|A\|}\right).\end{aligned}$$

If we solve the linear system in m-digit floating point arithmetic, then, as rule of thumb, we may approximate the the input errors due to rounding by

$$\frac{\|\Delta A\|}{\|A\|} \approx 0.5 * 10^{-m+1}, \quad \frac{\|\Delta \mathbf{x}\|}{\|\mathbf{x}\|} \approx 0.5 * 10^{-m+1}$$

If the condition number of A is $\kappa(A) = 10^{\alpha}$, then

$$\frac{\|\Delta \mathbf{x}\|}{\|\mathbf{x}\|} \leq \frac{10^{\alpha}}{1-10^{\alpha-m+1}}(0.5 * 10^{-m} + 0.5 * 10^{-m}) \approx 10^{\alpha-m}.$$

Provided $10^{\alpha-m+1} < 1$.

Rule of thumb: If the linear system is solved in m-digit floating point arithmetic and if the condition number of A is of the order 10^{α}, then only $m-\alpha-1$ digits in the solution can be trusted.

4 Numerical Integration

Our problem in this section is to compute

$$\int_a^b f(x)\, dx.$$

Even if $f(x)$ can be expressed in terms of elementary functions, the antiderivative of $f(x)$ may not have this property. For example: e^{-x^2}, $\sin(x^2)$, $\frac{\sin x}{x}$, etc. Hence, all exact techniques of integration taught in Calculus courses are more like exceptions than the rules. As a general rule, one must rely on numerical integration.

Our goal is to approximate the integral of a function f by a weighted sum of function values:

$$\int_a^b f(x)\, dx \approx \sum_{i=1}^{n} w_i f(x_i). \tag{4.1}$$

Definition 4.1. • In the above formula $x_i \in [a,b]$ are called the **nodes** of the integration formula and w_i are called the **weights** of the integration formula.
- When we approximate $\int_a^b f(x)\, dx$ by $\sum_{i=1}^n w_i f(x_i)$ we speak of **numerical integration** or **numerical quadrature**.
- $\sum_{i=1}^n w_i f(x_i)$ is called a quadrature formula.

Notice, that by introducing a new variable

$$x = a + \frac{b-a}{\beta-\alpha}(z-\alpha)$$

and by the change of variable formula

$$\int_a^b f(x)\,dx = \frac{b-a}{\beta-\alpha}\int_\alpha^\beta f\left(a+\frac{b-a}{\beta-\alpha}(z-\alpha)\right)\,dz,$$

if we have computed weights $\widehat{w}_i$ and nodes $\widehat{z}_i$ for the numerical integration on an interval $[\alpha,\beta]$, then we can use the above identity to approximate the integral of f over any interval $[a,b]$ (assuming, of course, that this integral exists) by

$$\begin{aligned}\int_a^b f(x)\,dx &= \frac{b-a}{\beta-\alpha}\int_\alpha^\beta f\left(a+\frac{b-a}{\beta-\alpha}(z-\alpha)\right)\,dz \\ &\approx \frac{b-a}{\beta-\alpha}\sum_{i=1}^n \widehat{w}_i f\left(a+\frac{b-a}{\beta-\alpha}(\widehat{z}_i-\alpha)\right).\end{aligned}$$

That is, the weights w_i and nodes x_i for the numerical integration on the interval $[a,b]$ are

$$w_i = \frac{b-a}{\beta-\alpha}\widehat{w}_i, \quad x_i = a+\frac{b-a}{\beta-\alpha}(\widehat{z}_i-\alpha).$$

This means it is sufficient to compute weights and nodes for the numerical integration on a certain interval like $[0,1]$ or $[-1,1]$, often called the reference intervals.

Before we discuss several quadrature methods, we list some properties of the integral which are important for the development of quadrature rules. First, we note that

$$\int_a^b 1\,dx = b-a.$$

Therefore we require

$$\sum_{i=1}^n w_i = b-a,$$

Otherwise, our quadrature formula could not even evaluate the integral of a constant function exactly.

Another useful property of the integral is

$$f(x)\geq 0 \quad \Longrightarrow \int_a^b f(x)\,dx \geq 0.$$

If

$$w_i \geq 0, \quad i=1,\ldots,n,$$

then

$$\sum_{i=1}^n w_i f(x_i) \geq 0,$$

for all functions $f(x)\geq 0$.

We also desire our numerical quadrature to be efficient. Efficiency often depends upon the number of function evaluations.

Basic idea: if $p(x)$ is some function such that

$$p(x)\approx f(x),$$

then

$$\int_a^b p(x)\,dx \approx \int_a^b f(x)\,dx$$

Thus we need a function $p(x)$ which close to $f(x)$ and easy to integrate.

The natural choice is the interpolating polynomials and their properties that we developed in Section 2.

Chose nodes $x_1,\ldots,x_n$ in the interval $[a,b]$ and compute the polynomial $P(f|x_1,\ldots,x_n)$ of degree less or equal to n interpolating f at $x_1,\ldots,x_n$. If we use the approximation

$$f(x) \approx P(f|x_1,\ldots,x_n)(x),$$

then we obtain an approximation for the integral:

$$\int_a^b f(x)\,dx \approx \int_a^b P(f|x_1,\ldots,x_n)(x)\,dx \quad (1). \tag{4.2}$$

These types of quadrature formulas are called interpolatory quadrature formulas.

From the form of the quadrature formula (4.1), it is useful to represent the interpolation polynomial using the Lagrange basis,

$$P(f|x_1,\ldots,x_n)(x) = \sum_{i=1}^{n} f(x_i) \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j}.$$

If we substitute this representation of the interpolation polynomial into (4.2), then we obtain

$$\int_a^b f(x)\,dx \approx \int_a^b P(f|x_1,\ldots,x_n)(x)\,dx = \int_a^b \sum_{i=1}^{n} f(x_i) \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j}\,dx = \sum_{i=1}^{n} f(x_i) \int_a^b \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j}\,dx.$$

This leads to the quadrature formula

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

where

$$w_i = \int_a^b \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j}\,dx.$$

Example 4.1 (Midpoint Rule). The simplest quadrature formula can be constructed using $n=1$ and $x_1 = \frac{a+b}{2}$. Since

$$\prod_{\substack{j=1 \\ j\neq i}}^{1} \frac{x-x_j}{x_i-x_j} = 1$$

we obtain the **midpoint rule:**

$$\int_a^b f(x)\,dx \approx (b-a) f\left(\frac{a+b}{2}\right). \tag{4.3}$$

Example 4.2 (Trapezoidal Rule). The next quadrature formula is constructed using $n=2$ and $x_1=a$, $x_2=b$. It holds

$$\int_a^b \frac{x-a}{b-a}dx = \frac{b-a}{2}, \quad \int_a^b \frac{x-b}{a-b}dx = \frac{b-a}{2}.$$

This yields the **Trapezoidal rule**:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{2}(f(a)+f(b)). \tag{4.4}$$

Example 4.3 (Simpson's rule).

The next quadrature formula is constructed using $n=3$ and $x_1=a$, $x_2=\frac{b+a}{2}$, $x_3=b$. Then

$$\begin{aligned}
&\int_a^b \frac{x-\frac{b+a}{2}}{a-\frac{b+a}{2}}\frac{x-b}{a-b}dx = \frac{b-a}{6},\\
&\int_a^b \frac{x-a}{\frac{b+a}{2}-a}\frac{x-b}{\frac{b+a}{2}-b}dx = 4\frac{b-a}{6},\\
&\int_a^b \frac{x-\frac{b+a}{2}}{b-\frac{b+a}{2}}\frac{x-b}{b-a}dx = \frac{b-a}{6}.
\end{aligned}$$

This yields the **Simpson rule**:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{6}\left(f(a)+4f\big(\frac{b+a}{2}\big)+f(b)\right). \tag{4.5}$$

4.1 Newton Cotes Quadrature Formula

If we have equidistant points

$$x_i = a+(i-1)h, \quad i=1,\dots,n, \quad h=\frac{b-a}{n-1},$$

then the resulting interpolatory quadrature formula is called a **closed Newton-Cotes quadrature formula** (a and b are nodes). In this case we can use the substitution $x=a+sh$, to compute

$$w_i = \int_a^b \prod_{\substack{j=1\\ j\neq i}}^{n}\frac{x-x_j}{x_i-x_j}dx = \frac{b-a}{n-1}\int_0^{n-1}\prod_{\substack{j=1\\ j\neq i}}^{n}\frac{s-j+1}{i-j}\,ds.$$

If we have equidistant points

$$x_i = a+ih, \quad i=1,\dots,n,$$

where

$$h=\frac{b-a}{n+1}, \quad x_1=a+h, \quad x_n=b-h,$$

then the resulting interpolatory quadrature formula is called an **open Newton-Cotes quadrature formula** (a and b are not nodes). Again, we can use the substitution $x=a+sh$, to compute

$$w_i = \int_a^b \prod_{\substack{j=1\\ j\neq i}}^{n}\frac{x-x_j}{x_i-x_j}dx = (b-a)\frac{1}{n+1}\int_0^{n+1}\prod_{\substack{j=1\\ j\neq i}}^{n}\frac{s-j}{i-j}\,ds.$$

4.1.1 Convergence of the numerical quadrature

Since the interpolation polynomial is uniquely determined, the interpolating polynomial for a polynomial p_{n-1} of degree less or equal to $n-1$ is the polynomial itself:

$$P(p_{n-1}|x_1,\ldots,x_n)(x) = p_{n-1}(x).$$

This implies that

$$\int_a^b p_{n-1}(x)dx = \int_a^b P(p_{n-1}|x_1,\ldots,x_n)(x)dx = \sum_{i=1}^n w_i p_{n-1}(x_i)$$

for all polynomials p_{n-1} of degree less or equal to $n-1$. If

$$\int_a^b p_{n-1}(x)dx = \sum_{i=0}^n w_i p_{n-1}(x_i)$$

for all polynomials p_{n-1} of degree less or equal to n we say that the integration method is exact of degree $n-1$.

In Theorem 2.9, we have established that for any $\bar{x} \in (a,b)$

$$f(\bar{x}) - P(f \mid x_1,\ldots,x_n)(\bar{x})| = \frac{1}{n!}\omega(\bar{x}) f^{(n)}(\xi(\bar{x})),$$

where $\omega(x) = \prod_{j=1}^n (x-x_j)$. Combining the above estimates, and integrating for the Newton-Cotes methods we obtain

$$\int_a^b f(x)dx - \sum_{i=1}^n w_i f(x_i) = \frac{1}{n!}\int_a^b f^{(n)}(\xi(x)) \prod_{j=1}^n (x-x_j)dx. \tag{4.6}$$

Taking the absolute values we immediately obtain

$$\left|\int_a^b f(x)dx - \sum_{i=1}^n w_i f(x_i)\right| \le \frac{1}{n!}\max_{a\le x\le b}|f^{(n)}(x)| \int_a^b \left|\prod_{j=1}^n (x-x_j)\right| dx. \tag{4.7}$$

However, using the weighted mean value theorem the for integrals, namely

Theorem 4.1 (Weighted Mean-Value Theorem for Integrals). *Suppose f is continuous on $[a,b]$ and g is integrable on $[a,b]$ and does not change sign. Then there exists $c \in (a,b)$ such that*

$$\int_a^b f(x)g(x)dx = f(c)\int_a^b g(x)dx.$$

we often can obtain sharper estimates.

Example 4.4 (Convergence for Trapezoidal Rule). If $f \in C^2(a,b)$, using Theorem 4.1 for the Trapezoidal rule ($n=2$, $h=b-a$) we obtain

$$\begin{aligned}\left|\int_a^b f(x)dx - \frac{b-a}{2}(f(a)+f(b))\right| &= \left|\frac{1}{2}\int_a^b f''(x)(x-a)(x-b)dx\right| \\ &= \left|\frac{f''(c)}{2}\int_a^b (x-a)(x-b)dx\right| \\ &= \left|\frac{f''(c)}{2}\frac{(b-a)^3}{6}\right| \\ &\le \frac{h^3}{12}\max_{a\le x\le b}|f''(x)|.\end{aligned}$$

Example 4.5 (Convergence for Midpoint Rule).

If $f \in C^1(a,b)$, using (4.6) for the Midpoint rule ($n=1$, $h=b-a$) we obtain

$$\left|\int_a^b f(x)dx-(b-a)f\left(\frac{a+b}{2}\right)\right|=\left|\frac{1}{2}\int_a^b f'(\xi(x))\left(x-\frac{a+b}{2}\right)dx\right|.$$

Since $\left(x-\frac{a+b}{2}\right)$ changes sign on (a,b) we can not use Theorem 4.1 and it seems the best we can do is just use (4.7) to obtain

$$\left|\int_a^b f(x)dx-(b-a)f\left(\frac{a+b}{2}\right)\right|\leq\frac{1}{2}\max_{a\leq x\leq b}|f'(x)|\int_a^b\left|\left(x-\frac{a+b}{2}\right)\right|dx=\frac{h^2}{8}\max_{a\leq x\leq b}|f'(x)|.$$

However, looking at the form of the error, we can notice that

$$\int_a^b\left(x-\frac{a+b}{2}\right)dx=0,$$

thus we can subtract any constant multiple of it from the function $f(x)$ without changing the value. Thus

$$\int_a^b f(x)dx-(b-a)f\left(\frac{a+b}{2}\right)=\int_a^b\left[f(x)-f'\left(\frac{a+b}{2}\right)\left(x-\frac{a+b}{2}\right)\right]dx-(b-a)f\left(\frac{a+b}{2}\right).$$

Now using that $f\left(\frac{a+b}{2}\right)$ is a constant we obtain

$$\int_a^b f(x)dx-(b-a)f\left(\frac{a+b}{2}\right)=\int_a^b\left[f(x)-f\left(\frac{a+b}{2}\right)-f'\left(\frac{a+b}{2}\right)\left(x-\frac{a+b}{2}\right)\right]dx.$$

Using the Taylor series expansion around $x_1=\frac{a+b}{2}$ provided $f\in C^2(a,b)$, we obtain

$$f(x)-f\left(\frac{a+b}{2}\right)-f'\left(\frac{a+b}{2}\right)\left(x-\frac{a+b}{2}\right)=\frac{1}{2}f''(\xi(x))\left(x-\frac{a+b}{2}\right)^2.$$

Now $\left(x-\frac{a+b}{2}\right)^2$ does not change the sign on (a,b) and we can use Theorem 4.1 to obtain

$$\int_a^b f(x)dx-(b-a)f\left(\frac{a+b}{2}\right)=\frac{1}{2}f''(c)\int_a^b\left(x-\frac{a+b}{2}\right)^2dx\leq\frac{(b-a)^3}{24}\max_{a\leq x\leq b}|f''(x)|=\frac{h^3}{24}\max_{a\leq x\leq b}|f''(x)|.$$

It is interesting to observe that the constant in the Trapezoidal method is twice as large as for the Midpoint Rule.

Example 4.6 (Convergence for Simpson's Rule). If $f\in C^3(a,b)$, using (4.6) for the Simpson's rule ($n=3$, $x_1=a$, $x_2=\frac{a+b}{2}$, $x_3=b$, $h=\frac{b-a}{2}$) we obtain

$$\left|\int_a^b f(x)dx-\frac{h}{3}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right)\right|=\left|\frac{1}{6}\int_a^b f'''(\xi(x))(x-a)\left(x-\frac{a+b}{2}\right)(x-b)dx\right|.$$

Similarly to the Midpoint Rule example, since $(x-a)\left(x-\frac{a+b}{2}\right)(x-b)$ changes sign on (a,b) we can not use Theorem 4.1 and it seems the best we can do is just use (4.7) to obtain

$$\begin{aligned}\left|\int_a^b f(x)dx-\frac{h}{3}\left(f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right)\right|&\leq\frac{1}{6}\max_{a\leq x\leq b}|f'''(x)|\int_a^b\left|(x-a)\left(x-\frac{a+b}{2}\right)(x-b)\right|dx\\&\leq Ch^3\max_{a\leq x\leq b}|f'''(x)|.\end{aligned}$$

However, exactly as in the Midpoint Rule example, looking at the form of the error, we can notice that

$$\int_a^b (x-a)\left(x-\frac{a+b}{2}\right)(x-b)dx = 0,$$

thus we can subtract any constant multiple of it from the function $f(x)$ without changing the value. In other words we can replace the interpolating polynomial $P(f|x_1,x_3,x_2)(x)$ that gives rise to the Simpson's Rule with

$$p(x) = P(f|x_1,x_3,x_2)(x) + c(x-x_1)(x-x_2)(x-x_3) \tag{4.8}$$

without changing the value of the numerical quadrature. Recalling the divided difference formula

$$f[x_j,\ldots,x_k] = \frac{f[x_{j+1},\ldots,x_k] - f[x_j,\ldots,x_{k-1}]}{x_k - x_j}. \tag{4.9}$$

We can extend the formula 6.4 to have equal nodes, i.e. $x_i = x_j$ for some $i \neq j$, with the convention that

$$f[x_i,x_i] = f'(x_i),$$

which is consistent with the definition of derivative, since

$$\lim_{\varepsilon\to 0} f[x_i,x_i+\varepsilon] = \lim_{\varepsilon\to 0}\frac{f(x_i+\varepsilon)-f(x_i)}{\varepsilon} = f'(x_i).$$

The constant c in (4.10), we take to be $f[x_1,x_3,x_2,x_2]$, and using the Newton basis, the interpolating polynomial takes the form

$$p(x) = f(x_1) + f[x_1,x_3](x-x_1) + f[x_1,x_3,x_2](x-x_1)(x-x_3) + f[x_1,x_3,x_2,x_2](x-x_1)(x-x_2)(x-x_3). \tag{4.10}$$

Carefully looking at the proof of Theorem 2.9, we can show

Proposition 1. *Let $f \in C^4[a,b]$. Then for each $\bar{x} \in [a,b]$ there exists $\xi(\bar{x}) \in (a,b)$ such that*

$$f(\bar{x}) - f(x_1) - f[x_1,x_3](\bar{x}-x_1) - f[x_1,x_3,x_2](\bar{x}-x_1)(\bar{x}-x_3) - f[x_1,x_3,x_2,x_2](\bar{x}-x_1)(\bar{x}-x_2)(\bar{x}-x_3) = \frac{1}{4!}f^{(4)}(\xi(\bar{x}))\omega(\bar{x}),$$

where $\omega(x) = (x-x_1)(x-x_2)^2(x-x_3)$.

Proof. The proof is almost identical to the proof of Theorem 2.9. If $x = x_i$ for some $i = 1,\ldots,n$, then the result naturally holds. Assume $x \neq x_i$ for any $i = 1,\ldots,n$. Consider a function

$$\psi(x) = f(x) - f(x_1) - f[x_1,x_3](\bar{x}-x_1) - f[x_1,x_3,x_2](\bar{x}-x_1)(\bar{x}-x_3) - f[x_1,x_3,x_2,x_2](\bar{x}-x_1)(\bar{x}-x_2)(\bar{x}-x_3) - c\omega(x),$$

where the constant c is taken to be

$$c = \frac{f(\bar{x}) - f(x_1) - f[x_1,x_3](\bar{x}-x_1) - f[x_1,x_3,x_2](\bar{x}-x_1)(\bar{x}-x_3) - f[x_1,x_3,x_2,x_2](\bar{x}-x_1)(\bar{x}-x_2)(\bar{x}-x_3)}{\omega(\bar{x})}.$$

With this choice of the constant c, the function $\psi(x)$ has at least 4 roots, namely at x_1,x_2,x_3 and $\bar{x}$. Thus, from the Rolle's Theorem there exist 3 points, call them $x_i^{(1)}$, $i = 1,2,3$, such that

$$\psi'(x_i^{(1)}) = 0, \quad i = 1,2,3.$$

In addition, by construction (it is also easy to check directly),

$$\psi'(x_2) = 0.$$

Since $x_i^{(1)} \neq x_2$ for all $i = 1,2,3$, it follows that $\psi'(x)$ has at least 4 roots as well, namely at x_1, x_2, x_3 and $\bar{x}$. Thus, from the Rolle's Theorem there exist 3 points, call them $x_i^{(2)}$, $i = 1,2,3$, such that

$$\psi''(x_i^{(1)}) = 0, \quad i = 1,2,3.$$

The of the proof is identical to the proof of Theorem 2.9. Continue this process, there exists a point $x_1^{(4)}$ such that

$$\psi^{(4)}(x_1^{(n)}) = 0.$$

From the definition of $\psi(x)$, we have

$$\frac{d^4}{dx^4}\psi(x) = \frac{d^n}{dx^4}f(x) - 0 - c\frac{d^4}{dx^4}\omega(x).$$

Since $\omega(x)$ is a polynomial of degree 4 with leading coefficient 1, we have

$$\frac{d^n}{dx^n}\omega(x) = n!.$$

As a result

$$\psi^{(4)}(x_1^{(4)}) = f^{(4)}(x_1^{(4)}) - c4! = 0 \qquad \Rightarrow \qquad c = \frac{f^{(4)}(x_1^{(4)})}{4!},$$

which shows the proposition with $\xi(\bar{x}) = x_1^{(4)}$.

Now we can continue with the Example 4.6. Since $\omega(x) = (x-x_1)(x-x_2)^2(x-x_3)$ does not change the sign on the interval (a,b), if $f \in C^4(a,b)$ we can use Theorem 4.1 to obtain

$$\int_a^b f(x)dx - \frac{h}{3}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right) = \frac{1}{4!}f^{(4)}(c)\int_a^b (x-x_1)(x-x_2)^2(x-x_3)dx \leq \frac{h^5}{90}\max_{a\leq x\leq b}|f^{(4)}(x)|,$$

with $h = \frac{b-a}{2}$.

The method discussed in the above example, can be generalized to any method for which $\int_a^b \omega(x)dx = 0$. In particular we can obtain the following result.

Theorem 4.2 (Exactness of Newton-Cotes Formulas). *Let $a \leq x_1 < \cdots < x_n \leq b$ be given and let w_i be the nodes and weights of a Newton-Cotes formula. If n is odd, then the quadrature formula is exact for polynomials of degree n. If n is even, then the quadrature formula is exact for polynomials of degree $n-1$.*

The weights and nodes for the most popular closed Newton-Cotes formulas are summarized in the table below.

n	h	$\widehat{w}_i$	formula	error	name
2	$b-a$	$\frac{1}{2}, \frac{1}{2}$	$\frac{(b-a)}{2}(f(a)+f(b))$	$h^3\frac{1}{12}f^{(2)}(\xi)$	Trapezoidal rule
3	$\frac{b-a}{2}$	$\frac{1}{6}, \frac{4}{6}, \frac{1}{6}$	$\frac{(b-a)}{6}(f(a)+4f(\frac{a+b}{2})+f(b))$	$h^5\frac{1}{90}f^{(4)}(\xi)$	Simpsons rule
4	$\frac{b-a}{3}$	$\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}$	$\frac{(b-a)}{8}(f(a)+3f(\frac{a+2b}{3})+3f(\frac{2a+b}{3})+f(b))$	$h^5\frac{3}{80}f^{(4)}(\xi)$	3/8-rule
5	$\frac{b-a}{4}$	$\frac{7}{90}, \frac{32}{90}, \frac{12}{90}, \frac{32}{90}, \frac{7}{90}$	$\frac{(b-a)}{90}(7f(x_1)+32f(x_2)+12f(x_3)+32f(x_4)+7f(x_5))$	$h^7\frac{8}{945}f^{(6)}(\xi)$	Boole's rule

In the above table

$$w_i = \widehat{w}_i(b-a), \quad x_i = a+(i-1)h, \quad i = 1,\ldots,n, \quad h = \frac{b-a}{n-1}.$$

The weights and nodes for the most popular open Newton-Cotes formulas are summarized in the table below.

n	h	$\widehat{w}_i$	formula	error	name
1	$\frac{b-a}{2}$	1	$(b-a)f(\frac{a+b}{2})$	$h^3\frac{1}{3}f^{(2)}(\xi)$	Midpoint rule
3	$\frac{b-a}{3}$	$\frac{1}{2},\frac{1}{2}$	$\frac{(b-a)}{2}(f(x_1)+f(x_2))$	$h^3\frac{1}{4}f^{(2)}(\xi)$	Trapezoid method
4	$\frac{b-a}{4}$	$\frac{2}{3},-\frac{1}{3},\frac{2}{3}$	$\frac{(b-a)}{3}(2f(x_1)-f(x_2)+2f(x_3))$	$h^5\frac{28}{90}f^{(4)}(\xi)$	Milne's rule
5	$\frac{b-a}{5}$	$\frac{11}{24},\frac{1}{24},\frac{1}{24},\frac{11}{24}$	$\frac{(b-a)}{24}(11f(x_1)+f(x_2)+f(x_3)+11f(x_4))$	$h^5\frac{95}{144}f^{(4)}(\xi)$	

In the above table

$$w_i = \widehat{w}_i(b-a), \quad x_i = a+ih, \quad i = 1,\ldots,n, \quad h = \frac{b-a}{n+1}.$$

4.2 Composite quadrature

Let $a = x_1 < x_2 \cdots < x_{m+1} = b$ be a partition of $[a,b]$ Then

$$\int_a^b f(x)dx = \sum_{j=1}^{m}\int_{x_j}^{x_{j+1}} f(x)dx.$$

Now we can approximate $\int_a^b f(x)dx$ by approximating each integral $\int_{x_j}^{x_{j+1}} f(x)dx$ by a (low degree) quadrature formula,

$$\int_{x_j}^{x_{j+1}} f(x)dx \approx \sum_{i=1}^{n} w_{ji}f(x_{ji})$$

and

$$\int_a^b f(x)dx = \sum_{i=1}^{n}\int_{x_j}^{x_{j+1}} f(x)dx \approx \sum_{j=1}^{m}\sum_{i=0}^{m} w_{ji}f(x_{ji}).$$

Example 4.7 (Composite Midpoint Rule).

$$\int_a^b f(x)dx \approx \sum_{j=1}^{m}(x_{j+1}-x_j)f\left(\frac{x_{j+1}+x_j}{2}\right).$$

Example 4.8 (Composite Trapezoidal Rule).

$$\int_a^b f(x)dx \approx \sum_{j=1}^{m}\frac{x_{j+1}-x_j}{2}\left(f(x_{j+1})+f(x_j)\right).$$

The function values $f(x_2),\ldots,f(x_m)$ appear twice in the summation. This can be utilized in the implementation of the composite Trapezoidal rule:

$$\int_a^b f(x)dx \approx \frac{x_2-x_1}{2}f(x_1)+\sum_{j=1}^{m}\left(\frac{x_j-x_{j-1}}{2}+\frac{x_{j+1}-x_j}{2}\right)f(x_j)+\frac{x_{m+1}-x_m}{2}f(x_{m+1}).$$

Example 4.9 (Composite Simpsons Rule).

$$\int_a^b f(x)dx \approx \sum_{j=1}^{m}\frac{x_{j+1}-x_j}{6}\left(f(x_j)+4f\big(\frac{x_{j+1}+x_j}{2}\big)+f(x_{j+1})\right).$$

Notice that the function values $f(x_2),\ldots,f(x_m)$ appear twice in the summation. This has to be utilized in the implementation of the composite Simpson rule.

4.3 Gauss Quadrature

The idea of the Gauss Quadrature is to choose nodes $x_1,\ldots,x_n$ and the weights $w_1,\ldots,w_n$ such that the formula

$$\int_a^b p(x)\,dx \approx \sum_{i=1}^{n} w_i p(x_i). \tag{4.11}$$

is exact for a polynomial of maximum degree. For example, if we require the formula (4.11) to be exact for N monomial basis, we obtain a **nonlinear system** of $N+1$ with $2n$ unknowns. Thus we can expect a formula to be exact for polynomials of degree $2n-1$.

Example 4.10 (Case $n=2$). Let's determine the weights w_1 and w_2 and the nodes x_1 and x_2 such that

$$w_1p(x_1)+w_2p(x_2)=\int_{-1}^{1}p(x)\,dx$$

holds for polynomials $p(x)$ of degree 3 or less. This seems possible since we have 4 parameters to choose w_1,w_2,x_1,x_2 and exactly 4 numbers are needed in order to define uniquely a polynomial of degree 3. Forcing formula to be exact for 1, x, x^2, and x^3, leads to

$$\begin{aligned}
w_1+w_2 &= \int_{-1}^{1}1\,dx = 2\\
w_1x_1+w_2x_2 &= \int_{-1}^{1}x\,dx = 0\\
w_1x_1^2+w_2x_2^2 &= \int_{-1}^{1}x^2\,dx = \frac{2}{3}\\
w_1x_1^3+w_2x_2^3 &= \int_{-1}^{1}x^3\,dx = 0
\end{aligned}$$

a nonlinear system of 4 equations with 4 unknowns. We can easily solve this system analytically to obtain

$$w_1=w_2=1,\quad x_1=-\frac{1}{\sqrt{3}},\ x_2=\frac{1}{\sqrt{3}}.$$

Actually we will consider more general problem. We want to choose nodes $x_1,\ldots,x_n$ and the weights $w_1,\ldots,w_n$ such that the formula

$$\int_a^b \omega(x)p(x)\,dx \approx \sum_{i=0}^{n} w_i p(x_i). \tag{4.12}$$

is exact for a polynomial of maximum degree, where $omega(x)$ is some positive function on (a,b). For example, 1, $\frac{1}{1+x^2}$, e^{-x^2} and etc.

It is seems reasonable to expect the quadrature formula (4.12) to be exact for polynomials of degree $2n-1$ or less, but not more than that as the following results shows.

Lemma 4.1. *There is no choice of nodes $x_1,\ldots,x_n$ and weights $w_1,\ldots,w_n$ such that*

$$\int_a^b \omega(x)p_N(x)\,dx \approx \sum_{i=1}^{n} w_i p_N(x_i).$$

for all polynomials p_N of degree less or equal to N if $N > 2n-1$.

Proof. Assume the formula exact for the polynomials of degree at least $2n$. Consider a function

$$g(x) = (x-x_1)^2\ldots(x-x_n)^2.$$

Easy to see that g is a polynomial of degree $2n$ and $g > 0$ on (a,b). Thus we obtain

$$0 < \int_a^b \omega(x)g(x)dx = \sum_{i=1}^{n} w_i g(x_i) = 0$$

a contradiction.

Lemma 4.2. *If* (4.12) *is exact for all polynomials of degree $2n-1$ or less, then the polynomials $p_0^*(x), p_1^*(x),\ldots,p_n^*(x)$ given by their roots, i.e.*

$$p_j^*(x) = \prod_{i=1}^{j}(x-x_i), \quad j = 1,2,\ldots,n$$

satisfy

$$\int_a^b \omega(x)p_j^*(x)p_i^*(x)dx = 0, \quad for \quad i \neq j.$$

Proof. If $i \neq j$, then $p_j^*(x)p_i^*(x)$ is a polynomial of degree at most $2n-1$, and since (4.12) is exact for $p_j^*(x)p_i^*(x)$, we obtain

$$\int_a^b \omega(x)p_j^*(x)p_i^*(x)dx = \sum_{k=1}^{n} w_k p_j^*(x_k)p_j^*(x_k) = 0.$$

Definition 4.2. For two integrable functions $f(x)$ and $g(x)$, and a given weight function $\omega(x) \geq 0$, we define a (weighted) inner-product by

$$(f,g) = \int_a^b \omega(x)f(x)g(x)dx.$$

One can easily check the above definition indeed satisfies all the conditions for the inner-product. Once we have a notion of inner-product for functions we can define an orthogonality.

Definition 4.3. We say that two functions $f(x)$ and $g(x)$ are orthogonal if

$$(f,g) = 0.$$

Thus the Lemma 4.2 says that if (4.12) is exact for all polynomials of degree $2n-1$ or less, then the nodes $x_1,\ldots,x_n$ are the roots of the orthogonal polynomials. Usually the roots of a polynomial can be repeated, complex, or lie outside of (a,b), this is not the case for the orthogonal polynomials.

Lemma 4.3. *The roots of the orthogonal polynomials in the sense of the Definition 4.3, are real, simple in lie inside the interval (a,b).*

Proof. Given an orthogonal polynomial $p_n(x)$ of degree n on (a,b), let $x_1,\dots,x_r$ be the points, where $p_n(x)$ changes it sign. Consider a function $g(x) = (x-x_1)\dots(x-x_r)$. Thus, $p_n(x)g(x) \geq 0$ on (a,b). If $r < n$, then $p_n(x)g(x)$ is a polynomial of degree less or equal than $2n-1$ and

$$0 < \int_a^b \omega(x)p_n(x)g(x)dx = \sum_{i=1}^{n} w_i p_n(x_i)g(x_i) = 0$$

a contradiction. Thus $r = n$ and it means that $p_n(x)$ changes sign n times and from this fact the conclusion of the lemma follows.

The next result can be think of as Gram-Schmidt orthogonalization process for polynomials.

Lemma 4.4. *We can construct orthogonal polynomials $p_j^*(x)$, $j = 1,\dots,n$ such that*

$$(p_j^*, p_k^*) = 0 \quad \textit{for} \quad j \neq k.$$

In the addition, the polynomials $p_j^*(x)$, $j = 1,\dots,n$ satisfy the three-term recursion

$$p_j^*(x) = (x-\delta_j)p_{j-1}^*(x) - \gamma_j^2 p_{j-2}^*(x), \quad j \geq 1,$$

where $p_{-1}^* = 0$, $p_0^* = 1$ and

$$\delta_j = \frac{(xp_{j-1}^*, p_{j-1}^*)}{(p_{j-1}^*, p_{j-1}^*)}, \quad \gamma_j^2 = \frac{(p_{j-1}^*, p_{j-1}^*)}{(p_{j-2}^*, p_{j-2}^*)}, \quad j \geq 1, \quad \gamma_1 = 0. \tag{4.13}$$

Proof. The proof is by induction. Suppose we constructed such polynomials $p_j^*(x)$ for $j \leq m$ and established that they are unique. Next, we want to construct $p_{m+1}^*(x)$ and that $(p_{m+1}^*, p_j) = 0$ for $j \leq m$ and satisfies (4.13)*.*

Any polynomial of degree $m+1$ with leading coefficient 1, we can write uniquely as

$$p_{m+1}(x) = (x-\delta_{m+1})p_m^*(x) + c_{m-1}p_{m-1}^*(x) + \dots + c_1 p_1^*(x). \tag{4.14}$$

Since $(p_j^*, p_k^*) = 0$ for any $k, j \leq m$ and $j \neq k$, we have an equation

$$0 = (p_{m+1}, p_m^*) = ((x-\delta_{m+1})p_m^*, p_m^*) + c_{m-1}(p_{m-1}^*, p_m^*) + \dots + c_0(p_0^*, p_m^*) = (xp_m^*, p_m^*) - \delta_{m+1}(p_m^*, p_m^*).$$

Since $(p_m^*, p_m^*) > 0$, it has a solution

$$\delta_{m+1} = \frac{(xp_m^*, p_m^*)}{(p_m^*, p_m^*)}.$$

We also require that $(p_{m+1}, p_j^*) = 0$ for all $j \leq m-1$. Using (4.14) *again, we have equations in c_j*

$$0 = (p_{m+1}, p_j^*) = ((x-\delta_{m+1})p_m^*, p_j^*) + c_{m-1}(p_{m-1}^*, p_j^*) + \dots + c_j(p_j^*, p_j^*) \dots + c_0(p_0^*, p_j^*) = (p_m^*, xp_j^*) + c_j(p_j^*, p_j^*). \tag{4.15}$$

By induction for $j \leq m-1$

$$p_{j+1}^*(x) = (x-\delta_j)p_j^*(x) - \gamma_j^2 p_{j-1}^*(x),$$

hence

$$xp_j^*(x) = p_{j+1}^*(x) + \delta_j p_j^*(x) + \gamma_j^2 p_{j-1}^*(x) \quad \textit{for} \quad j \leq m-1.$$

Plugging it into (4.15) *and using that $j \leq m-1$, we obtain*

$$\begin{aligned} 0 = (p_{m+1}, p_j^*) &= (p_m^*, xp_j^*) + c_j(p_j^*, p_j^*) \\ &= (p_m^*, p_{j+1}^*) + \delta_j(p_m^*, p_j^*) + \gamma_j^2(p_m^*, p_{j-1}^*) + c_j(p_j^*, p_j^*) \\ &= (p_m^*, p_{j+1}^*) + c_j(p_j^*, p_j^*). \end{aligned}$$

The above equations have solutions $c_j = 0$ *for* $j < m-1$ *and* $c_{m-1} = -\frac{(p_m^*, p_m^*)}{(p_{m-1}^*, p_{m-1}^*)}$. *Since* $\frac{(p_m^*, p_m^*)}{(p_{m-1}^*, p_{m-1}^*)} > 0$ *we set*

$$\gamma^2 := c_{m-1} = \frac{(p_m^*, p_m^*)}{(p_{m-1}^*, p_{m-1}^*)}.$$

The above lemma allows us to generate the orthogonal polynomials recursively. Later, we will see that we can compute the real roots of any polynomial very efficiently. We also have an alternative way to compute the nodes x_i.

Theorem 4.3. *The root* x_i, $i = 1, \ldots, n$ *are the eigenvalues of the tridiagonal matrix*

$$J_n = \begin{pmatrix} \delta_1 & \gamma_2 & & & \\ \gamma_2 & \delta_1 & \gamma_3 & & \\ & \cdot & \cdot & \cdot & \\ & & \cdot & \cdot & \gamma_n \\ & & & \gamma_n & \delta_n \end{pmatrix}$$

Proof. The proof follows easily be showing that the characteristic polynomials

$$p_n(x) = det(J_n - xI)$$

satisfy the three-term recursion

$$p_n(x) = (x - \delta_j)p_{n-1}(x) - \gamma_j^2 p_{n-2}(x), \quad n \geq 1,$$

which can be done by induction, for example.

We still need to obtain the corresponding weights. That is what we will address now.

Lemma 4.5. *Let* $p_0^*(x), \ldots, p_{n-1}^*(x)$ *be orthogonal polynomials and* $x_1, \ldots, x_n$ *be any distinct points. Then the matrix*

$$P = \begin{pmatrix} p_0^*(x_1) & p_0^*(x_2) & \ldots & \ldots & p_0^*(x_n) \\ p_1^*(x_1) & p_1^*(x_2) & \ldots & \ldots & p_1^*(x_n) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ p_{n-1}^*(x_1) & p_{n-1}^*(x_2) & \ldots & \ldots & p_{n-1}^*(x_n) \end{pmatrix}$$

is non-singular.

Proof. We will proof the result by contradiction. Assume the matrix P is singular. Then there exists a vector $\mathbf{z} \in \mathbb{R}^n$ such that $\mathbf{z}^T P = \mathbf{0}^T$. Hence

$$\sum_{j=0}^{n-1} z_j p_j^*(x_i) = 0 \quad \text{for} \quad i = 1, \ldots n.$$

Notice that

$$q(x) = \sum_{j=0}^{n-1} z_j p_j^*(x)$$

is a polynomial of degree $n-1$ that has n roots, namely at $x_1, \ldots, x_n$. Hence $q(x) \equiv 0$. Let k be the largest index such that $z_k \neq 0$. Then

$$p_k^*(x) = -\frac{1}{z_k}\sum_{i=0}^{k-1} z_i p_i^*(x),$$

which is a contradiction, since on the left we have a polynomial of degree $k-1$ and a polynomial of degree less than $k-1$ on the right.

Theorem 4.4. *Let $x_1,\dots,x_n$ be the roots of the p_n^* and let $\mathbf{w} = (w_1,\dots,w_n)^T$ be the solution of the system*

$$P\mathbf{w} = \mathbf{g}, \tag{4.16}$$

where P is the matrix defined in Lemma 4.5 and the $\mathbf{g} = (g_0,\dots,g_{n-1})^T$ is given by

$$g_i = \begin{cases} (p_0^*, p_0^*) & i = 0 \\ 0 & i = 1,\dots,n-1. \end{cases}$$

Then $w_i > 0$, $i = 1,\dots,n$ and

$$\int_a^b \omega(x)p(x)\,dx = \sum_{i=1}^{n} w_i p(x_i). \tag{4.17}$$

for all $p \in \mathbb{P}_{2n-1}$

Conversely, if w_i and x_i, $i = 1,\dots,n$, are such that (4.17) *holds for all $p \in \mathbb{P}_{2n-1}$, then x_i the roots of $p_{n+1}^*(x)$ and w_i satisfy* (4.16)*.*

Proof. Since the roots of the orthogonal polynomials are simple, real, and inside (a,b) by Lemma 4.5 the matrix P is non-singular. Hence the system (4.16) has a unique solution.

Consider any $p(x) \in \mathbb{P}_{2n-1}$. We can write

$$p(x) = p_n^*(x)q(x) + r(x), \tag{4.18}$$

where $q \in \mathbb{P}_{n-1}$ and $r \in \mathbb{P}_{n-1}$. Since $\{p_0^*,\dots,p_{n-1}^*\}$ form a basis for $\mathbb{P}_{n-1}$, we have

$$q(x) = \sum_{k=0}^{n-1} \alpha_k p_k^*(x) \quad \text{and} \quad r(x) = \sum_{k=0}^{n-1} \beta_k p_k^*(x).$$

Then,

$$\int_a^b \omega(x)p(x)dx = \int_a^b \omega(x)(p_n^*(x)q(x)+r(x))dx = (p_n^*, q) + (r,1) = \sum_{k=0}^{n-1} \alpha_k (p_n^*, p_k^*) + \sum_{k=0}^{n-1} \beta_k (p_k^*, p_0^*) = \beta_0 (p_0^*, p_0^*).$$

On the other hand using that x_i are the roots of p_n^*,

$$\sum_{i=1}^{n} w_i p(x_i) = \sum_{i=1}^{n} w_i (p_n^*(x_i)q(x_i) + r(x_i)) = \sum_{i=1}^{n} w_i r(x_i) = \sum_{k=0}^{n-1} \beta_k \left(\sum_{i=1}^{n} w_i p_k^*(x_i) \right) = \beta_0 (p_0^*, p_0^*),$$

by using (4.16). Hence we have (4.17).

To show $w_j > 0$ for $j = 1,\dots,n$, we consider $b_j(x) = (x-x_1)^2 \dots (x-x_{j-1})^2 (x-x_{j+1})^2 \dots (x-x_n)^2 \in \mathbb{P}_{2n-2}$. Using that $b_j(x) > 0$ on (a,b) and (4.17), we have

$$0 < \int_a^b \omega(x) b_j(x) dx = \sum_{i=1}^{n} w_i b_j(x_i) = w_j (x_j - x_1)^2 \dots (x_j - x_{j-1})^2 (x_j - x_{j+1})^2 \dots (x_j - x_n)^2.$$

From above, the positivity of w_j follows.

To show the converse, notice that the nodes $x_1,\dots,x_n$ are distinct (otherwise we could write the formula (4.17) with less than n nodes which would contradict Lemma 4.1), hence the matrix P is non-singular.

Applying formula (4.17) with $p(x)=p_k^*(x)$, $k=0,\dots,n-1$, we have

$$\sum_{i=1}^{n} w_i p_k^*(x_i) = \int_a^b \omega(x) p_n^*(x)dx = (p_0^*,p_k^*) = \begin{cases} (p_0^*,p_0^*) & k=0 \\ 0 & k=1,\dots,n-1. \end{cases},$$

Hence the weights w_i satisfy (4.16).

Applying formula (4.17) with $p(x)=p_k^*(x)p_n^*(x)$, $k=0,\dots,n-1$, we have

$$0=(p_k^*,p_n^*)=\sum_{i=1}^{n} w_i p_k^*(x_i)p_n^*(x_i).$$

In other words the vector

$$\mathbf{c}=(p_n^*(x_1),\dots,p_n^*(x_n))^T,$$

solves the homogeneous system

$$P\mathbf{c}=0.$$

Since P is non-singular, we have $\mathbf{c}=\mathbf{0}$, i.e. x_i are the roots of $p_n^*(x)$.

4.4 Error analysis

5 Linear Least Squares

The most common and simplest problem in linear least squares is the linear regression problem

Example 5.1 (Linear regression). Given m measurements

$$(x_i,y_i), \quad i=1,\dots,m,$$

find a linear function

$$y(x)=ax+b$$

that best fits these data, i.e.,

$$y_i \approx ax_i+b \quad i=1,\dots,m.$$

In other words, we want two numbers a and b such that

$$\sum_{i=1}^{m}(ax_i+b-y_i)^2$$

is minimized. To write the problem in matrix form, we let

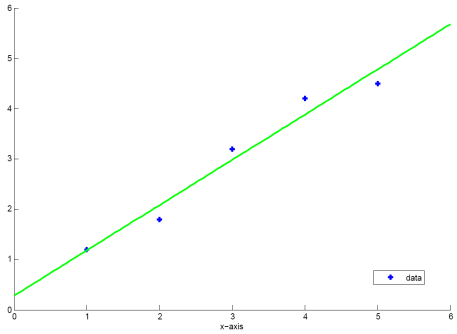
$$A=\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_m & 1 \end{pmatrix} \in \mathbb{R}^{m\times 2}, \quad \mathbf{b}=\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \in \mathbb{R}^m$$

then the ith residual

$$r_i = ax_i + b - y_i$$

is the i-th component of $A\mathbf{z} - b$, where $\mathbf{z} = [a\ b]^T$. Thus we want to minimize $\|\mathbf{r}\|_2^2$ which leads to

$$\min_{\mathbf{z}\in\mathbb{R}^2} \|A\mathbf{z} - \mathbf{b}\|_2^2$$



Loosely speaking, the linear least squares problem says: Given $A \in \mathbb{R}^{m\times n}$, find $\mathbf{x} \in \mathbb{R}^n$ such that $A\mathbf{x} \approx \mathbf{b}$.

Of course, if $m = n$ and A is invertible, then we can solve $A\mathbf{x} = \mathbf{b}$. Otherwise, we may not have a solution of $A\mathbf{x} = \mathbf{b}$ or we may have infinitely many of them.

We are interested in vectors $\mathbf{x}$ that minimize the norm of squares of the residual $A\mathbf{x} - \mathbf{b}$, i.e., which solve

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|_2^2$$

The Linear Least Squares (LLS) problem:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|_2^2.$$

Notice that

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|_2^2, \quad \min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|_2, \quad \frac{1}{2}\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|_2^2$$

are all equivalent in the sense that if $\mathbf{x}$ solves one of them it also solves the others.

Instead of finding $\mathbf{x}$ that minimizes the norm of squares of the residual $A\mathbf{x} - \mathbf{b}$, we could also try to find $\mathbf{x}$ that minimizes the p-norm of the residual

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|_p$$

This can be done, but is more complicated and will not be covered.

There are many examples of such problems.

Example 5.2 (Best Polynomial Fitting). Given m measurements

$$(x_i, y_i), \quad i = 1, \ldots, m,$$

find a polynomial function

$$y(x) = a_n x^n + \cdots a_1 x + a_0$$

that best fits these data, i.e.,

$$y_i \approx a_n x_i^n + \cdots a_1 x_i + a_0 \quad i = 1, \ldots, m.$$

We want two numbers a and b such that

$$\sum_{i=1}^{m} (a_n x_i^n + \cdots a_1 x_i + a_0 - y_i)^2$$

is minimized.

Write in matrix form. Let

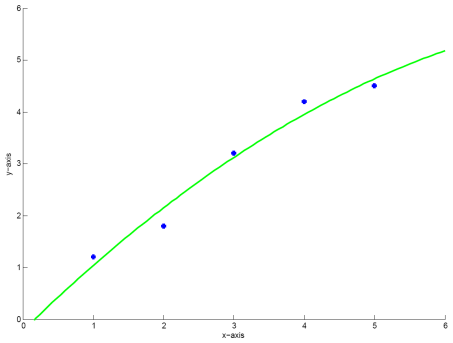
$$A = \begin{pmatrix} x_1^n & \dots & x_1 & 1 \\ x_2^n & \dots & x_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_m^n & \dots & x_m & 1 \end{pmatrix} \in \mathbb{R}^{m\times(n+1)}, \quad \mathbf{b} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \in \mathbb{R}^m$$

then the ith residual

$$r_i = a_n x_i^n + \cdots a_1 x_i + a_0 - y_i$$

is the ith component of $Az-b$, where $z = [a_n, \ \dots, \ a_1, \ a_0]^T$. Thus we want to minimize $\|r\|_2^2$ which leads again to

$$\min_{z\in\mathbb{R}^n} \|Az-b\|_2^2$$



Sometimes one has to modify the problem in order to state it as a LLS problem

Example 5.3 (Best Circle Fitting). Find a best fit circle through points $(x_1,y_1),(x_2,y_2),\dots,(x_m,y_m)$. Equation for the circle around (c_1,c_2) with radius r is

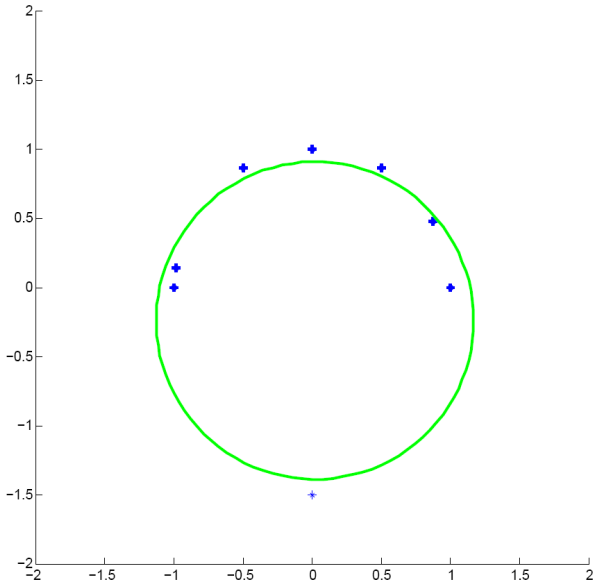
$$(x-c_1)^2+(y-c_2)^2 = r^2.$$

It is not a LLS problem, due to quadratic terms. However, rewrite the equation for the circle in the form

$$2xc_1+2yc_2+(r^2-c_1^2-c_2^2) = x^2+y^2$$

and setting $c_3 = r^2-c_1^2-c_2^2$, then we can compute the center (c_1,c_2) and the radius $r = \sqrt{c3+c_1^2+c_2^2}$ of the circle that best fits the data points by solving the least squares problem

$$\min_{[c_1,c_2,c_3]^T\in\mathbb{R}^3} \left\| \begin{pmatrix} 2x_1 & 2y_1 & 1 \\ 2x_2 & 2y_2 & 1 \\ \vdots & \vdots & \vdots \\ 2x_m & 2y_m & 1 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} - \begin{pmatrix} x_1^2+y_1^2 \\ x_2^2+y_2^2 \\ \vdots \\ x_m^2+y_m^2 \end{pmatrix} \right\|_2^2$$



5.1 Normal Equation

Suppose x_* satisfies

$$\|Ax_* - b\|_2^2 = \min_{x\in\mathbb{R}^n} \|Ax - b\|_2^2 \quad \text{(LLS)}$$

For any vector $z \in \mathbb{R}^n$

$$\begin{aligned}
&\|Ax_* - b\|_2^2 \le \|A(x_* + z) - b\|_2^2 \\
&= (A(x_* + z) - b)^T (A(x_* + z) - b) \\
&= x_*^T A^T A x_* - 2x_*^T A^T b + b^T b + 2z^T A^T A x_* - 2z^T A^T b + z^T A^T z \\
&= \|Ax_* - b\|_2^2 + 2z^T (A^T A x_* - A^T b) + \|Az\|_2^2.
\end{aligned}$$

Of course $\|Az\|_2^2 \ge 0$, but

$$2z^T (A^T A x_* - A^T b)$$

could be negative for some z if $A^T A x_* - A^T b \neq 0$. In fact setting

$$z = -\alpha(A^T A x_* - A^T b)$$

for some $\alpha \in \mathbb{R}$. For such $z \in \mathbb{R}^n$, we get

$$2z^T (A^T A x_* - A^T b) + \|Az\|_2^2 = -2\alpha \|A^T A x_* - A^T b\|_2^2 + \alpha^2 \|A(A^T A x_* - A^T b)\|_2^2 < 0$$

for

$$0 < \alpha < \frac{\|A^T A x_* - A^T b\|_2^2}{\|A(A^T A x_* - A^T b)\|_2^2}.$$

Thus, if x_* solves (LLS) then x_* must satisfy

$$A^T A x_* - A^T b = 0 \quad \textbf{normal equation}. \tag{5.1}$$

On the other hand if x_* satisfies

$$A^T A x_* - A^T b = 0,$$

then for any x

$$\begin{aligned} \|Ax-b\|_2^2 &= \|Ax_* + A(x-x_*) - b\|_2^2 \\ &= \|Ax_* - b\|_2^2 + 2(x-x_*)^T (A^T A x_* - A^T b) + \|A(x-x_*)\|_2^2 \\ &= \|Ax_* - b\|_2^2 + \|A(x-x_*)\|_2^2 \\ &\geq \|Ax_* - b\|_2^2 \end{aligned}$$

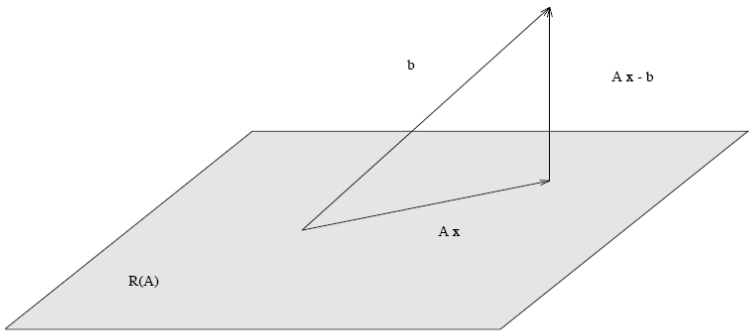
i.e. x_* solves (LLS).

Theorem 5.1. *The linear least square problem*

$$\min_{x \in \mathbb{R}^n} \|Ax-b\|_2^2 \quad \textit{(LLS)}$$

always has a solution. A vector x_* solves (LLS) iff x_* solves the normal equation

$$A^T A x = A^T b.$$



Note: If the matrix $A \in \mathbb{R}^{m \times n}$, $m \geq n$, has rank n, then $A^T A$ is symmetric positive definite and satisfies

$$v^T A^T A v = \|Av\|_2^2 > 0, \quad \forall v \in \mathbb{R}^n,\ v \neq 0.$$

If $A \in \mathbb{R}^{m \times n}$, $m \geq n$, has full rank n, then we can use the Cholesky-decomposition to solve the normal equation (and, hence, the linear least squares problem) as follows

1. Compute $A^T A$ and $A^T b$.
2. Compute the Cholesky-decomposition $A^T A = R^T R$.
3. Solve $R^T y = A^T b$ (forward solve),
   solve $Rx = y$ (backward solve) .

The computation of A^TA and A^Tb requires roughly mn^2 and $2mn$ flops. Roughly $\frac{1}{3}n^3$ flops are required to compute the Cholesky-decomposition. The solution of $R^Ty = A^Tb$ and of $Rx = y$ requires approximately $2n^2$ flops.

Computing the normal equations requires us to calculate terms of the form $\sum_{k=1}^{m} a_{ki}a_{kj}$. The computed matrix A^TA may not be positive definite, because of floating point arithmetic.

```
t = 10.^(0:-1:-10)';
A = [ ones(size(t)) t t.^2 t.^3 t.^4 t.^5];
B = A'*A;
[R,iflag] = chol( B );
if( iflag ~= 0 )
disp([' Cholesky decomposition returned with iflag = ', ...
int2str(iflag)])
end
```

In exact arithmetic $B = A^TA$ is symmetric positive definite, but the Cholesky-Decomposition detects that $a_{jj} - \sum_{k=1}^{j-1} r_{jk}^2 < 0$ in step $j = 6$.

```
>> Cholesky decomposition returned with iflag = 6
```

The use of the Cholesky decomposition is problematic if the condition number of A^TA is large. In the example, $\kappa_2(A^TA) \approx 4.7 * 10^{16}$.

5.2 Solving Linear Least Square problem using QR-decomposition

Definition 5.1. A matrix $Q \in \mathbb{R}^{m\times n}$ is called orthogonal if $Q^TQ = I_n$, i.e., if its columns are orthogonal and have 2-norm one.

The orthogonal matrices have the following important properties

1. If $Q \in \mathbb{R}^{n\times n}$ is orthogonal, then $Q^TQ = I$ implies that $Q^{-1} = Q^T$.
2. If $Q \in \mathbb{R}^{n\times n}$ is an orthogonal matrix, then Q^T is an orthogonal matrix.
3. If $Q_1, Q_2 \in \mathbb{R}^{n\times n}$ are orthogonal matrices, then Q_1Q_2 is an orthogonal matrix.
4. If $Q \in \mathbb{R}^{n\times n}$ is an orthogonal matrix, then

$$(Qx)^T(Qy) = x^Ty \quad x, y \in \mathbb{R}^n$$

   i.e. the angle between Qx and Qy is equal to the angle between x and y
5. As a result

$$\|Qx\|_2 = \|x\|_2$$

   i.e. orthogonal matrices preserve the 2-norm.

The last property is the key for solving LSS.

Example 5.4. In two dimensions a rotation matrix

$$Q = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

is orthogonal matrix.

5.2.1 QR-decompostion

Let $m \geq n$. For each $A \in \mathbb{R}^{m\times n}$ there exists a permutation matrix $P \in \mathbb{R}^{mn\times n}$, an orthogonal matrix $Q \in \mathbb{R}^{m\times m}$, and an upper triangular matrix $R \in \mathbb{R}^{n\times n}$ such that

$$AP = Q\begin{pmatrix} R \\ 0 \end{pmatrix} \begin{matrix} \} \, n \\ \} \, m-n \end{matrix} \quad \text{QR-decomposition.}$$

The QR decomposition of A can be computed using the Matlab command $[Q,R,P] = qr(A)$.

We will not go into the details of how Q,P,R are computed. If you interested check Chapter 5 of the book Gene Golub and Charles Van Loan, *Matrix Computations*

5.2.2 Case 1: $Rank(A) = n$

Assume that $A \in \mathbb{R}^{m\times n}$, has full rank n. (Rank deficient case will be considered later.)

Let

$$AP = Q\begin{pmatrix} R \\ 0 \end{pmatrix} \begin{matrix} \} \, n \\ \} \, m-n \end{matrix} \quad \Leftrightarrow \quad Q^T AP = \begin{pmatrix} R \\ 0 \end{pmatrix} \begin{matrix} \} \, n \\ \} \, m-n \end{matrix}$$

where $R \in \mathbb{R}^{n\times n}$ is upper triangular matrix. Since A has full rank n the matrix R also has rank n and, therefore, is nonsingular. Moreover, since Q is orthogonal it obeys $QQ^T = I$. Hence

$$\|Q^T y\|_2 = \|y\|_2 \quad \forall y \in \mathbb{R}^m.$$

In addition, the permutation matrix satisfies $PP^T = I$. Using these properties of Q we get

$$\begin{aligned} \|Ax-b\|_2^2 &= \|Q^T(Ax-b)\|_2^2 \\ &= \|Q^T(APP^T x-b)\|_2^2 \\ &= \|(Q^T AP)P^T x - Q^T b\|_2^2 \\ &= \|\begin{pmatrix} R \\ 0 \end{pmatrix} P^T x - Q^T b\|_2^2. \end{aligned}$$

Partitioning $Q^T b$ as

$$Q^T b = \begin{pmatrix} c \\ d \end{pmatrix} \begin{matrix} \} \, n \\ \} \, m-n \end{matrix}$$

and putting $y = P^T x$ we get

$$\begin{aligned} \|Ax-b\|_2^2 = \left\| \begin{pmatrix} R \\ 0 \end{pmatrix} y - \begin{pmatrix} c \\ d \end{pmatrix} \right\|_2^2 &= \left\| \begin{pmatrix} Ry-c \\ -d \end{pmatrix} \right\|_2^2 \\ &= \|Ry-c\|_2^2 + \|d\|_2^2. \end{aligned}$$

Thus,

$$\min_x \|Ax-b\|_2^2 \quad \Leftrightarrow \quad \min_y \|Ry-c\|_2^2 + \|d\|_2^2$$

and the solution is $y = R^{-1}c$.

Thus,

$$\min_x \|Ax-b\|_2^2 \quad \Leftrightarrow \quad \min_y \|Ry-c\|_2^2 + \|d\|_2^2$$

and the solution is $y = R^{-1}c$.

Recall

$$y = P^T x, \quad PP^T = I, \quad \Rightarrow \quad x = Py.$$

Hence the solution is $x = Py = PR^{-1}c$.

In Summary: To solve a Linear Least Squares Problem using the QR-Decomposition with matrix $A \in \mathbb{R}^{m\times n}$, of rank n and $b \in \mathbb{R}^m$:

1. Compute an orthogonal matrix $Q \in \mathbb{R}^{m\times m}$, an upper triangular matrix $R \in \mathbb{R}^{n\times n}$, and a permutation matrix $P \in \mathbb{R}^{n\times n}$ such that
$$Q^T AP = \begin{pmatrix} R \\ 0 \end{pmatrix}.$$
2. Compute
$$Q^T b = \begin{pmatrix} c \\ d \end{pmatrix}.$$
3. Solve
$$Ry = c.$$
4. Set
$$x = Py.$$

The MATLAB Implementation is very simple.

```
[m,n] = size(A);
[Q,R,P] = qr(A);
c = Q'*b;
y = R(1:n,1:n) \ c(1:n);
x = P*y;
```

If you type

$$x = A\backslash b;$$

in Matlab, then Matlab computes the solution of the linear least squares problem

$$\min_x \|Ax - b\|_2^2$$

using the QR decomposition as described above.

5.2.3 Case 2: $Rank(A) < n$

The Rank Deficient Case: Assume that $A \in \mathbb{R}^{m\times n}$, $m \geq n$ has rank $r < n$. (The case $m < n$ can be handled analogously.)

Suppose that

$$AP = QR,$$

where $Q \in \mathbb{R}^{m\times m}$ is orthogonal, $P \in \mathbb{R}^{n\times n}$ is a permutation matrix, and $R \in \mathbb{R}^{n\times n}$ is an upper triangular matrix of the form

$$R = \begin{pmatrix} R_1 & R_2 \\ 0 & 0 \end{pmatrix}$$

with nonsingular upper triangle $R_1 \in \mathbb{R}^{r\times r}$ and $R_2 \in \mathbb{R}^{r\times(n-r)}$.

We can write

$$\|Ax - b\|_2^2 = \|Q^T(APP^T x - b)\|_2^2 = \left\| \begin{pmatrix} R_1 R_2 \\ 0 \end{pmatrix} P^T x - Q^T b \right\|_2^2.$$

Partition $Q^T b$ as

$$Q^T b = \begin{pmatrix} c_1 \\ c_2 \\ d \end{pmatrix} \begin{matrix} \} \, r \\ \} \, n-r \\ \} \, m-n \end{matrix}$$

and put $y = P^T x$.

Partition

$$y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \begin{matrix} \} \, r \\ \} \, n-r \end{matrix}$$

This give us

$$\|Ax-b\|_2^2 = \left\| \begin{pmatrix} R_1 y_1 + R_2 y_2 - c_1 \\ c_2 \\ d \end{pmatrix} \right\|_2^2 = \|R_1 y_1 + R_2 y_2 - c_1\|_2^2 + \|c_2\|_2^2 + \|d\|_2^2.$$

Linear least squares problem $\min_x \|Ax-b\|_2^2$ is equivalent to

$$\|R_1 y_1 + R_2 y_2 - c_1\|_2^2 + \|c_2\|_2^2 + \|d\|_2^2,$$

where $R_1 \in \mathbb{R}^{r \times r}$ is nonsingular. the solution is

$$y_1 = R_1^{-1}(c_1 - R_2 y_2)$$

for any $y_2 \in \mathbb{R}^{n-r}$.

Since $y = P^T x$ and $P^T P = I$,

$$x = Py = P \begin{pmatrix} R_1^{-1}(c_1 - R_2 y_2) \\ y_2 \end{pmatrix}$$

We have infinitely many solutions since y_2 is arbitrary. Which one to choose?

If we use Matlab $x = A \backslash b$, then Matlab computes the one with $y_2 = 0$

$$x = P \begin{pmatrix} R_1^{-1} c_1 \\ 0 \end{pmatrix}.$$

MATLAB Implementation is:

```
[m,n] = size(A);
[Q,R,P] = qr(A);
c = Q'*b;
% Determine rank of A.
% The diagonal entries of R satisfy
%|R(1,1)| >= |R(2,2)| >= |R(3,3)| >= ..
% Find the smallest integer r such that
%|R(r+1,r+1)| < max(size(A))*eps*|R(1,1)|
tol = max(size(A))*eps*abs(R(1,1));
r = 1;
while ( abs(R(r+1,r+1)) >= tol & r < n ); r = r+1; end
y1 = R(1:r,1:r) \ c(1:r);
y2 = zeros(n-r,1);
x = P*[y1;y2];
```

where we used

to determinate of the effective rank of $A \in \mathbb{R}^{n \times n}$ using the QR decomposition

$$AP = QR,$$

where the diagonal entries of R satisfy $|R_{11}| \geq |R_{22}| \geq \ldots$. The effective rank r of $A \in \mathbb{R}^{n\times n}$ is the smallest integer r such that

$$|R_{r+1,r+1}| < \varepsilon \max\{m,n\}|R_{11}|$$

```
tol = max(size(A))*eps*abs(R(1,1));
r = 0;
while ( abs(R(r+1,r+1)) >= tol & r < n )
r = r+1;
end
```

All solutions of

$$\min_x \|Ax-b\|_2^2$$

are given by

$$x = Py = P\begin{pmatrix} R_1^{-1}(c_1 - R_2 y_2) \\ y_2 \end{pmatrix}$$

where $y_2 \in \mathbb{R}^{n-r}$ is arbitrary.

Minimum norm solution:

Of all solutions, pick the one with the smallest 2-norm. This leads to

$$\min_{y_2} \left\| P\begin{pmatrix} R_1^{-1}(c_1 - R_2 y_2) \\ y_2 \end{pmatrix} \right\|_2^2$$

Since permutation matrix P is orthogonal

$$\left\| P\begin{pmatrix} R_1^{-1}(c_1 - R_2 y_2) \\ y_2 \end{pmatrix} \right\|_2^2 = \left\| \begin{matrix} R_1^{-1}(c_1 - R_2 y_2) \\ y_2 \end{matrix} \right\|_2^2 = \left\| \begin{matrix} R_1^{-1}(c_1 - R_2 y_2) \\ -y_2 \end{matrix} \right\|_2^2 = \left\| \begin{pmatrix} R_1^{-1}R_2 \\ I \end{pmatrix} y_2 - \begin{pmatrix} R_1^{-1}c_1 \\ 0 \end{pmatrix} \right\|_2^2$$

which is another linear least squares problem with unknown y_2. This problem is $n \times (n-r)$ and it has full rank. It can be solved using the techniques discussed earlier.

MATLAB Implementation:

```
[m,n] = size(A);
[Q,R,P] = qr(A);
c = Q'*b;
% Determine rank of A (as before).
tol = max(size(A))*eps*abs(R(1,1));
r = 1;
while ( abs(R(r+1,r+1)) >= tol & r < n ); r = r+1; end
% Solve least squares problem to get y2
S = [ R(1:r,1:r) \ R(1:r,r+1:n);
eye(n-r) ];
t = [ R(1:r,1:r) \ c(1:r);
zeros(n-r,1) ];
y2 = S \ t; % solve least squares problem using backslash
% Compute x
y1 = R(1:r,1:r) \ ( c(1:r) - R(1:r,r+1:n) * y2 );
x = P*[y1;y2];
```

5.3 Solving Linear Least Square problem using SVD-decomposition

For any matrix $A \in \mathbb{R}^{m\times n}$ there exist orthogonal matrices $U \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$ and a 'diagonal' matrix $\Sigma \in \mathbb{R}^{m\times n}$, i.e.,

$$\Sigma = \begin{pmatrix} \sigma_1 & & & 0 \dots 0 \\ & \ddots & & \\ & & \sigma_r & \\ & & 0 & \\ & & & \ddots \\ & & & 0 \dots 0 \end{pmatrix} \quad \text{for} \quad m \le n$$

and

$$\Sigma = \begin{pmatrix} \sigma_1 & & & & \\ & \ddots & & & \\ & & \sigma_r & & \\ & & & 0 & \\ & & & & \ddots \\ 0 & & & & 0 \\ \vdots & & & & \vdots \\ 0 & & & & 0 \end{pmatrix} \quad \text{for} \quad m \ge n$$

with diagonal entries

$$\sigma_1 \ge \cdots \ge \sigma_r > \sigma_{r+1} = \cdots = \sigma_{\min\{m,n\}} = 0$$

such that $A = U\Sigma V^T$.

Definition 5.2. The decomposition

$$A = U\Sigma V^T$$

is called **S**ingular Value Decomposition (SVD). It is very important decomposition of a matrix and tells us a lot about its structure.

It can be computed using the Matlab command *svd*.

Definition 5.3. The diagonal entries σ_i of Σ are called the singular values of A. The columns of U are called *left singular vectors* and the columns of V are called *r*ight singular vectors.

Using the orthogonality of V we can write it in the form

$$AV = U\Sigma$$

We can interpret it as follows: there exists a special orthonormal set of vectors (i.e. the columns of V), that is mapped by the matrix A into an orthonormal set of vectors (i.e. the columns of U).

Given the SVD-Decomposition of A,

$$A = U\Sigma V^T$$

with

$$\sigma_1 \ge \cdots \ge \sigma_r > \sigma_{r+1} = \cdots = \sigma_{\min\{m,n\}} = 0$$

one may conclude the following:

- $rank(A) = r$,

- $R(A) = R([u_1, \ldots, u_r])$,
- $N(A) = R([v_{r+1}, \ldots, v_n])$,
- $R(A^T) = R([v_1, \ldots, v_r])$,
- $N(A^T) = R([u_{r+1}, \ldots, u_m])$.

Moreover if we denote

$$U_r = [u_1, \ldots, u_r], \quad \Sigma_r = diag(\sigma_1, \ldots, \sigma_r), \quad V_r = [v_1, \ldots, v_r],$$

then we have

$$A = U_r \Sigma_r V_r^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

This is called the *dyadic decomposition* of A, decomposes the matrix A of rank r into sum of r matrices of rank 1.

The 2-norm and the Frobenius norm of A can be easily computed from the SVD decomposition

$$\|A\|_2 = \sup_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \sigma_1$$
$$\|A\|_F = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij}^2 = \sqrt{\sigma_1^2 + \cdots + \sigma_p^2}, \quad p = \min\{m, n\}.$$

From the SVD decomposition of A it also follows that

$$A^T A = V \Sigma^T \Sigma V^T \quad \text{and} \quad AA^T = U \Sigma \Sigma^T U^T.$$

Thus, σ_i^2, $i = 1, \ldots, p$ are the eigenvalues of symmetric matrices $A^T A$ and AA^T and v_i and u_i are the corresponding eigenvectors.

Theorem 5.2. *Let the SVD of $A \in \mathbb{R}^{m \times n}$ be given by*

$$A = U_r \Sigma_r V_r^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

with $r = rank(A)$. If $k < r$

$$A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T,$$

then

$$\min_{rank(D)=k} \|A - D\|_2 = \|A - A_k\|_2 = \sigma_{k+1},$$

and

$$\min_{rank(D)=k} \|A - D\|_F = \|A - A_k\|_F = \sqrt{\sum_{k+1}^{p} \sigma_i^2}, \quad p = \min\{m, n\}.$$

Consider the LLS

$$\min_x \|Ax - b\|_2^2$$

and let $A = U \Sigma V^T$ be the SVD of $A \in \mathbb{R}^{m \times n}$. Using the orthogonality of U and V we have

$$\|Ax - b\|_2^2 = \|U^T (AVV^T x - b)\|_2^2 = \|\Sigma \underbrace{V^T x}_{=z} - U^T b)\|_2^2$$
$$= \sum_{i=1}^{r} (\sigma_i z_i - u_i^T b)^2 + \sum_{i=r+1}^{m} (u_i^T b)^2.$$

Thus,

$$\min_x \|Ax-b\|_2^2 = \sum_{i=1}^{r}(\sigma_i z_i - u_i^T b)^2 + \sum_{i=r+1}^{m}(u_i^T b)^2.$$

The solution is given

$$\begin{aligned} z_i &= \frac{u_i^T b}{\sigma_i}, \quad i = 1,\ldots,r, \\ z_i &= \text{ arbitrary}, \quad i = r+1,\ldots,n. \end{aligned}$$

As a result

$$\min_x \|Ax-b\|_2^2 = \sum_{i=r+1}^{m}(u_i^T b)^2.$$

Recall that $z = V^T x$. Since V is orthogonal, we find that

$$\|x\|_2 = \|VV^T x\|_2 = \|V^T x\|_2 = \|z\|_2.$$

All solutions of the linear least squares problem are given by $z = V^T x$ with

$$\begin{aligned} z_i &= \frac{u_i^T b}{\sigma_i}, \quad i = 1,\ldots,r, \\ z_i &= \text{ arbitrary}, \quad i = r+1,\ldots,n. \end{aligned}$$

The minimum norm solution of the linear least squares problem is given by

$$x_\dagger = V z_\dagger,$$

where $z_\dagger \in \mathbb{R}^n$ is the vector with entries

$$\begin{aligned} z_i^\dagger &= \frac{u_i^T b}{\sigma_i}, \quad i = 1,\ldots,r, \\ z_i^\dagger &= 0, \quad i = r+1,\ldots,n. \end{aligned}$$

The minimum norm solution is

$$x_\dagger = \sum_{i=1}^{r} \frac{u_i^T b}{\sigma_i} v_i$$

MATLAB code:

```
% compute the SVD:
[U,S,V] = svd(A);
   s    = diag(S);
% determine the effective rank r of A using singular values
r = 1;
while( r < size(A,2) & s(r+1) >= max(size(A))*eps*s(1) )
    r = r+1;
end
d = U'*b;
x = V* ( [d(1:r)./s(1:r); zeros(n-r,1) ] );
```

Suppose that the data b are

$$b = b_{ex} + \delta b,$$

where δb represents the measurement error. The minimum norm solution of $\min \|Ax-(b_{ex}+\delta b)\|_2^2$ is

$$x_\dagger = \sum_{i=1}^{r} \frac{u_i^T b}{\sigma_i} v_i = \sum_{i=1}^{r} \left(\frac{u_i^T b}{\sigma_i} + \frac{u_i^T \delta b}{\sigma_i} \right) v_i.$$

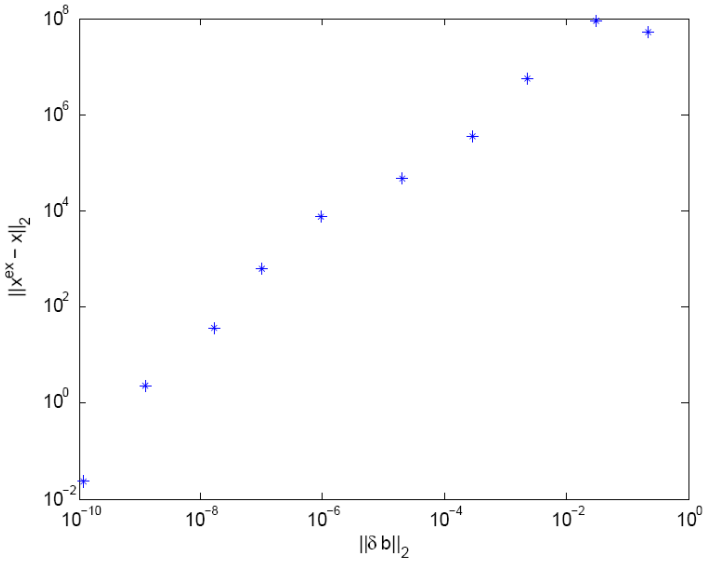
If a singular value σ_i is small, then $\frac{u_i^T(\delta b)}{\sigma_i}$ could be large, even if $u_i^T(\delta b)$ is small. This shows that errors δb in the data can be magnified by small singular values σ_i.

```
% Compute A
t = 10.^(0:-1:-10)';
A = [ ones(size(t)) t t.^2 t.^3 t.^4 t.^5];
% compute SVD of A
[U,S,V] = svd(A); sigma = diag(S);
% compute exact data
xex = ones(6,1); bex = A*xex;
for i = 1:10
    % data perturbation
    deltab = 10^(-i)*(0.5-rand(size(bex))).*bex;
    b = bex+deltab;
    % solution of perturbed linear least squares problem
    w = U'*b;
    x = V * (w(1:6) ./ sigma);
    errx(i+1) = norm(x - xex); errb(i+1) = norm(deltab);
end
loglog(errb,errx,'*');
ylabel('||x^{ex} - x||_2'); xlabel('||\delta b||_2')
```

The singular values of A in the above Matlab example are:

$$\begin{array}{ll} \sigma_1 \approx 3.4 & \sigma_4 \approx 7.2 * 10^{-4} \\ \sigma_2 \approx 2.1 & \sigma_5 \approx 6.6 * 10^{-7} \\ \sigma_3 \approx 8.2 * 10^{-2} & \sigma_6 \approx 5.5 * 10^{-11} \end{array}$$

The error $\|x_{ex} - x\|_2$ for different values of $\|\delta b\|_2$ (loglog-scale):



We see that small perturbations δb in the measurements can lead to large errors in the solution x of the linear least squares problem if the singular values of A are small.

5.4 Regularized Linear Least Squares

If $\sigma_1/\sigma_r \gg 1$, then it might be useful to consider the **regularized linear least squares problem** (Tikhonov regularization)

$$\min_{x\in\mathbb{R}^n} \frac{1}{2}\|Ax-b\|_2^2 + \frac{\lambda}{2}\|x\|_2^2. \tag{5.2}$$

Here $\lambda > 0$ is the *regularization parameter*.

The regularization parameter $\lambda > 0$ is not known *a-priori* and has to be determined based on the problem data. Observe that

$$\min_x \frac{1}{2}\|Ax-b\|_2^2 + \frac{\lambda}{2}\|x\|_2^2 = \min_x \left\| \begin{pmatrix} A \\ \sqrt{\lambda}I \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2^2.$$

For $\lambda > 0$, the matrix

$$\begin{pmatrix} A \\ \sqrt{\lambda}I \end{pmatrix} \in \mathbb{R}^{(m+n)\times n}$$

is always of full rank n. Hence, for $\lambda > 0$, the regularized linear least squares problem (5.2) has a unique solution. The normal equation corresponding to (5.2) are given by

$$\begin{pmatrix} A \\ \sqrt{\lambda}I \end{pmatrix}^T \begin{pmatrix} A \\ \sqrt{\lambda}I \end{pmatrix} x = (A^TA + \lambda I)x = A^Tb = \begin{pmatrix} A \\ \sqrt{\lambda}I \end{pmatrix}^T \begin{pmatrix} b \\ 0 \end{pmatrix}^T.$$

Using the SVD Decomposition of $A = U\Sigma V^T$, where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal matrices and $\Sigma \in \mathbb{R}^{m\times n}$ is a 'diagonal' matrix with diagonal entries

$$\sigma_1 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_{\min\{m,n\}} = 0.$$

the normal to (5.2)

$$(A^TA + \lambda I)x_\lambda = A^Tb,$$

can be written as

$$(V\Sigma^T \underbrace{U^TU}_{=I}\Sigma V^T + \lambda \underbrace{I}_{=VV^T})x_\lambda = V\Sigma^TU^Tb.$$

Rearranging the terms, we obtain

$$V(\Sigma^T\Sigma + \lambda I)V^Tx_\lambda = V\Sigma^TU^Tb,$$

multiplying both sides by V^T from the left and setting $z = V^Tx_\lambda$ we get

$$(\Sigma^T\Sigma + \lambda I)z = \Sigma^TU^Tb.$$

Thus, the normal equation

$$(A^TA + \lambda I)x_\lambda = A^Tb,$$

is equivalent to

$$\underbrace{(\Sigma^T\Sigma + \lambda I)}_{\text{diagonal}}z = \Sigma^TU^Tb,$$

where $z = V^T x_\lambda$ and has the solution

$$z_i = \begin{cases} \frac{\sigma_i(u_i^T b)}{\sigma_i^2+\lambda}, & i = 1,\ldots,r, \\ 0, & i = r+1,\ldots,n. \end{cases}$$

Since $x_\lambda = Vz = \sum_{i=1}^n z_i v_i$, the solution of the regularized linear least squares problem (5.2) is given by

$$x_\lambda = \sum_{i=1}^r \frac{\sigma_i(u_i^T b)}{\sigma_i^2+\lambda} v_i.$$

Note that

$$\lim_{\lambda\to 0} x_\lambda = \lim_{\lambda\to 0} \sum_{i=1}^r \frac{\sigma_i(u_i^T b)}{\sigma_i^2+\lambda} v_i = \sum_{i=1}^r \frac{u_i^T b}{\sigma_i} v_i = x_\dagger$$

i.e., the solution of the regularized Least Square problem (5.2) converges to the minimum norm solution of the original Least Square problem as λ goes to zero. The representation

$$x_\lambda = \sum_{i=1}^r \frac{\sigma_i(u_i^T b)}{\sigma_i^2+\lambda} v_i$$

of the solution of the regularized LLS also reveals the regularizing property of adding the term $\frac{\lambda}{2}\|x\|_2^2$ to the (ordinary) least squares functional. We have that

$$\frac{\sigma_i(u_i^T b)}{\sigma_i^2+\lambda} \approx \begin{cases} 0, & \text{if } 0 \approx \sigma_i \ll \lambda \\ \frac{u_i^T b}{\sigma_i}, & \text{if } \sigma_i \gg \lambda. \end{cases}$$

Hence, adding $\frac{\lambda}{2}\|x\|_2^2$ to the original least squares functional acts as a filter. Contributions from singular values which are large relative to the regularization parameter λ are left (almost) unchanged whereas contributions from small singular values are (almost) eliminated. It raises an important question:

How to choose λ?

Suppose that the data are $b = b_{ex} + \delta b$. We want to compute the minimum norm solution of the original Least Squares problem with unperturbed data b_{ex}

$$x_{ex} = \sum_{i=1}^r \frac{u_i^T b}{\sigma_i} v_i,$$

but we can only compute with $b = b_{ex} + \delta b$, we don't know b_{ex}. The solution of the regularized least squares problem is

$$x_\lambda = \sum_{i=1}^r \left(\frac{\sigma_i(u_i^T b_{ex})}{\sigma_i^2+\lambda} + \frac{\sigma_i(u_i^T \delta b)}{\sigma_i^2+\lambda} \right) v_i.$$

We observed that

$$\sum_{i=1}^r \frac{\sigma_i(u_i^T b_{ex})}{\sigma_i^2+\lambda} \to x_{ex} \quad \text{as} \quad \lambda \to 0.$$

On the other hand

$$\frac{\sigma_i(u_i^T \delta b)}{\sigma_i^2+\lambda} \approx \begin{cases} 0, & \text{if } 0 \approx \sigma_i \ll \lambda \\ \frac{u_i^T \delta b}{\sigma_i}, & \text{if } \sigma_i \gg \lambda, \end{cases}$$

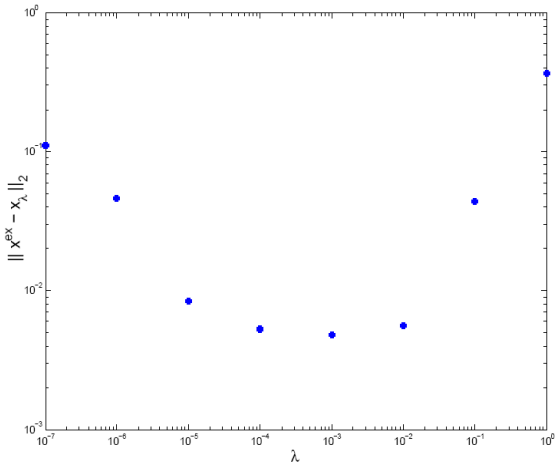
which suggests to choose λ sufficiently large to ensure that errors δb in the data are not magnified by small singular values.

Example 5.5 (Vandermonde matrix).

```
% Compute A
```

```
t = 10.^(0:-1:-10)';
A = [ ones(size(t)) t t.^2 t.^3 t.^4 t.^5];

% compute exact data
xex = ones(6,1); bex = A*xex;

% data perturbation of 0.1%
deltab = 0.001*(0.5-rand(size(bex))).*bex;
b      = bex+deltab;

% compute SVD of A
[U,S,V] = svd(A); sigma = diag(S);

for i = 0:7 % solve regularized LLS for different lambda
    lambda(i+1) = 10^(-i)
    xlambda = V * (sigma.*(U'*b) ./ (sigma.^2 + lambda(i+1)))
    err(i+1) = norm(xlambda - xex);
end
loglog(lambda,err,'*'); ylabel('||x^{ex} - x_{\lambda}||_2'); xlabel('\lambda');
```

The error $\|x_{ex} - x_\lambda\|_2$ for different values of λ (loglog-scale):



For this example $\lambda \approx 10^{-3}$, seems to be a good choice for the regularization parameter λ. However, we could only create this figure with the knowledge of the desired solution x_{ex}.

So the question is, how can we determine a $\lambda \geq 0$ so that $\|x_{ex} - x_\lambda\|_2$ is small without knowledge of x_{ex}. One approach is the **Morozov discrepancy principle**.

Suppose $b = b_{ex} + \delta b$. We do not know the perturbation δb, but we assume that we know its size $\|\delta b\|$. Suppose the unknown desired solution x_{ex} satisfies $Ax_{ex} = b_{ex}$. Hence,

$$\|Ax_{ex} - b\| = \|Ax_{ex} - b_{ex} - \delta b\| = \|\delta b\|.$$

Since the exact solution satisfies $\|Ax_{ex} - b\| = \|\delta b\|$ we want to find a regularization parameter $\lambda \geq 0$ such that the solution x_λ of the regularized least squares problem satisfies

$$\|Ax_\lambda - b\| = \|\delta b\|$$

This is **Morozov's discrepancy principle.**

Morozov's discrepancy principle: Find $\lambda \geq 0$ such that

$$\|Ax_\lambda - b\| = \|\delta b\|$$

To compute $\|Ax_\lambda - b\|$ for given $\lambda \geq 0$ we need to solve a regularized linear least squares problem

$$\min_x \frac{1}{2}\|Ax-b\|_2^2 + \frac{\lambda}{2}\|x\|_2^2 = \min_x \left\| \begin{pmatrix} A \\ \sqrt{\lambda} I \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2^2$$

to get x_λ and then we have to compute $\|Ax_\lambda - b\|$.

Let $f(\lambda) = \|Ax_\lambda - b\| - \|\delta b\|$. Finding $\lambda \geq 0$ such that

$$f(\lambda) = 0$$

is a **root finding problem**. We will discuss in the next section how to solve such problems. In this case f maps a scalar λ into a scalar

$$f(\lambda) = \|Ax_\lambda - b\| - \|\delta b\|,$$

but the evaluation of f requires the solution of a regularized Least Square problems and can be rather expensive, so one has to watch out for a number of function evaluations.

6 Numerical Solution of Nonlinear Equations in $\mathbb{R}^1$

Goal: Given a function $f : \mathbb{R} \to \mathbb{R}$ we want to find x_* such that $f(x_*) = 0$

Definition 6.1. A point x_* with $f(x_*) = 0$ is called a **root** of f or **zero** of f.

6.1 Bisection Method

The bisection method is based on the following version of **Intermediate Value Theorem**: If $f : \mathbb{R} \to \mathbb{R}$ is a continuous function and $a, b \in \mathbb{R}$, $a < b$, and $f(a)f(b) < 0$, then there exists $x \in [a,b]$, such that $f(x) = 0$.

Thus, given $[a_k, b_k]$ with $f(a_k)f(b_k) < 0$ (i.e., $f(a_k)$ and $f(b_k)$ have opposite signs, compute the interval midpoint $c_k = \frac{1}{2}(a_k + b_k)$ and evaluates $f(c_k)$. If $f(c_k)$ and $f(a_k)$ have opposite sign, then $[a_k, c_k]$ contains a root of f. Sets $a_{k+1} = a_k$ and $b_{k+1} = c_k$. Otherwise $f(c_k)$ and $f(b_k)$ must the opposite sign. In this case, $[c_k, b_k]$ contains a root of f and we sets $a_{k+1} = c_k$ and $b_{k+1} = b_k$.

The algorithm is rather straightforward.

```
Input: Initial values a(0); b(0) such that f(a(0))f(b(0)) < 0
and a tolerance tolx
Output: approximation of a root of f

for k = 0, 1,... do
    if b(k)-a(k) < tolx,
        return c(k) = (a(k) + b(k))/2 as an approximate root of f and stop
    end
```

```
    Compute c(k) = (a(k) + b(k))/2 and f(c(k)).
    if f(c(k)) = 0,   x = c(k);    end

    if f(a(k))f(c(k)) < 0 , then
             a(k+1) = a(k);
             b(k+1) = c(k);
    else
            a(k+1) = c(k);
            b(k+1) = b(k);
    end
 end
```

6.1.1 Convergence of the Bisection Method

In each step the interval $[a_k, b_k]$ is halved. Hence

$$|a_{k+1} - b_{k+1}| = \frac{1}{2}|a_k - b_k| = \frac{1}{2^2}|a_{k-1} - b_{k-1}| = \cdots = \frac{1}{2^{k+1}}|a_0 - b_0|.$$

There is a root of f such that the midpoints satisfy

$$|x_* - c_k| \leq 2^{-k+1}|b_0 - a_0|.$$

In particular, $\lim_{k\to\infty} c_k = x_*$. By $\lfloor z \rfloor$ we denote the largest integer less or equal to z. After

$$k = \lfloor \log_2 \left(\frac{|b_0 - a_0|}{tol_x} \right) \rfloor - 1$$

iterations we are guaranteed to have an approximation c_k of a root x_* of f that satisfies $|x_* - c_k| \leq tol_x$.

The following table gives "pros" and "cons" of the bisection method.

"pros"	"cons"
The method is very simple	Hard to generalize to higher dimensions
The Bisection method requires only function values. In fact, it only requires the sign of function values (that means as long as the sign is correct, the function values can be inaccurate).	The Bisection method only requires the sign of function values. In general, it will not find the root of the simple affine linear function $f(x) = ax + b$ in a finite number of iterations.
The method is very robust. The Bisection method converges for any $[a_0, b_0]$ that contains a root of f (the Bisection method converges globally).	The local convergence behavior of the Bisection method is rather slow (the error only reduced by a factor 2, no matter how close we are to the solution.

6.2 Regula Falsi

One of the problem with the bisection method, is that the midpoint may have nothing to do with the real root. One of the possible improvements of bisection method, is to use the function values at the end points (if available) more efficiently.

Thus, given an initial interval $[a_0, b_0]$ such that $f(a_0)f(b_0) < 0$ (i.e., $[a_0, b_0]$ contains a root of f), Regula Falsi constructs an affine linear function $m(x) = \alpha x + \beta$ such that $m(a_0) = f(a_0)$ and $m(b_0) = f(b_0)$. In the notation of

section 2, $m(x) = P(f|a_0, b_0)(x)$ i.e. m interpolates f at a_0 and at b_0. $m(x) = P(f|a_0, b_0)(x)$ is given by

$$m(x) = f(a_0) + (f(b_0) - f(a_0))\frac{x - a_0}{b_0 - a_0}$$

Instead of taking c_0 as a mid point, we choose c_0 such that $m(c_0) = 0$. Solving for c_0 we obtain

$$c_0 = a_0 - \frac{b_0 - a_0}{f(b_0) - f(a_0)} f(a_0).$$

Thus it use c_0 as an approximation of the root of f. This root satisfies $c_0 \in (a_0, b_0)$. If $f(c_0)$ and $f(a_0)$ have opposite signs, then we set $a_1 = a_0$ and $b_1 = c_0$. Otherwise $f(c_0)$ and $f(b_0)$ must have opposite signs and we sets $a_1 = c_0$ and $b_1 = b_0$. Then we proceed similarly to the bisection method. This give us an algorithm

```
Input: Initial values a(0); b(0) such that f(a(0))*f(b(0)) < 0,
a maximum number of iterations maxit, a tolerance tolf and a tolerance tolx

Output: approximation x of the root

1 For k = 0,1,...,maxit do
2 If b(k) - a(k) < tolx, then return x = c(k) and stop
3 Compute c(k) = a(k) - f(a(k))(b(k) - a(k))(f(b(k)) - f(a(k)))
    and f(c(k)).
4 If |f(c(k))| < tolf , then return x = c(k) and stop
5 If f(a(k))f(c(k)) < 0 , then
6 a(k+1) = a(k); b(k+1) = c(k),
7 else
8 a(k+1) = c(k); b(k+1) = b(k).
9 End
10 End
```

However the following simple numerical example shows that computationally we often can not see any advantage of Regula Falsi over Bisection Rule.

Example 6.1. Take $f(x) = x^3 - 2x - 5$ on interval $[0, 4]$, with tolerance $tol = 10^{-5}$.

6.3 Newton's Method

The idea of the Newton's method is similar Regula Falsi, to approximate the $f(x)$ with a linear function, but this time for a given an approximation x_0 of a root x_* of f, we select

$$m(x) = f(x_0) + f'(x_0)(x - x_0).$$

Comparing to the Taylor approximation of f around x_0

$$f(x_*) = f(x_0 + (x_* - x_0)) \approx f(x_0) + f'(x_0)(x_* - x_0).$$

We see that we use the tangent of f at x_0, as a model for f. The root

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

of the tangent model is used as an approximation of the root of f. Write the previous identity as

$$s_0 = -f(x_0)/f'(x_0) \quad \text{step (correction)}$$
$$x_1 = x_0 + s_0.$$

```
Input: Initial values x(0), tolerance tol,
maximum number of iterations maxit

Output: approximation of the root

1 For k = 0:maxit do
2 Compute s(k) = -f(x(k))/f'(x(k)).
3 Compute x(k+1) = x(k) + s(k).
4 Check for truncation
5 End
```

6.3.1 Convergence of Sequences.

Let $\{x_k\}$ be a sequence of real numbers.

Definition 6.2 (Convergence rates).

1. The sequence is called **linearly convergent** if there exists $c \in (0,1)$ and $\hat{k} \in \mathbb{N}$ such that
$$|x_{k+1} - x_*| \leq c|x_k - x_*| \quad \text{for all} \quad k \geq \hat{k}.$$
2. The sequence is called **superlinearly convergent** if there exists a sequence $\{c_k\}$ with $c_k > 0$ and $\lim_{k\to\infty} c_k = 0$ such that
$$|x_{k+1} - x_*| \leq c_k|x_k - x_*|$$
or, equivalently, if
$$\lim_{k\to\infty} \frac{|x_{k+1} - x_*|}{|x_k - x_*|} = 0.$$
3. The sequence is called **quadratically convergent** to x_* if $\lim_{k\to\infty} x_k = x_*$ and if there exists $c > 0$ and $\hat{k} \in \mathbb{N}$ such that
$$|x_{k+1} - x_*| \leq c|x_k - x_*|^2 \quad \text{for all} \quad k \geq \hat{k}.$$

Thus using the above definition, we can see that bisection method is linearly convergence with $c = 1/2$. On the other hand as we will see, the Newton's method is quadratically convergent under some conditions.

6.3.2 Convergence of Newton's Method.

Theorem 6.1. *Let $D \in \mathbb{R}$ be an open interval and let $f : D \to \mathbb{R}$ be differentiable on D Furthermore, let $f'(x)$ be Lipschitz continuous with Lipschitz constant L. If $x_* \in D$ is a root and if $f'(x_*) \neq 0$, then there exists an $\varepsilon > 0$ such that Newton's method with starting point x_0 with $|x_0 - x_*| < 0$ generates iterates x_k which converge to x_*,*

$$\lim_{k\to\infty} x_k = x_*,$$

and which obey

$$|x_{k+1}-x_*| \le \frac{L}{|f'(x_*)|}|x_k-x_*|^2 \quad \textit{for all} \quad k \in \mathbb{N}.$$

Before proving the above theorem, we give "pros" and "cons" of the Newton's method.

"pros"	"cons"
The method is very simple	Requires derivatives of a function
Fast convergence.	Only local convergence and requires good initial guess
Can be easily generalized to $\mathbb{R}^n$	
Can be easily modified	

We remind the following definition.

Definition 6.3 (Lipschitz continuity). The function $f : [a,b] \to \mathbb{R}$ is said to be **Lipschitz continuous** if there exists $L > 0$ such that

$$|f(x)-f(y)| \le L|x-y|, \quad \forall x,y \in [a,b].$$

The constant L is called the **Lipschitz constant**.

For a given $x,y \in [a,b]$, consider a function

$$\phi(t) = f(y+t(x-y)).$$

We can check that $\phi(0) = f(y)$ and $\phi(1) = f(x)$. Thus by the Fundamental Theorem of Calculus and the chain rule

$$f(x)-f(y) = \phi(1)-\phi(0) = \int_0^1 \phi'(t)\,dt = \int_0^1 f'(y+t(x-y))\,dt(x-y). \tag{6.1}$$

Taking $y = x_0$, we have

$$\begin{aligned} f(x)-f(x_0)-f'(x_0)(x-x_0) &= \int_0^1 f'(y+t(x-y))(x-x_0)\,dt - f'(x_0)(x-x_0) \\ &= \int_0^1 \left[f'(x_0+t(x-x_0))-f'(x_0)\right](x-x_0)\,dt. \end{aligned}$$

Thus if $f'(x)$ is Lipschitz with Lipschitz constant L

$$|f(x)-f(x_0)-f'(x_0)(x-x_0)| \le \int_0^1 \left|f'(x_0+t(x-x_0))-f'(x_0)\right| dt|x-x_0| \le \frac{L}{2}|x-x_0|^2. \tag{6.2}$$

Using the above inequality we observe that for the Newton's first step

$$x_1-x_* = x_0-\frac{f(x_0)}{f'(x_0)}-x_* = \frac{1}{f'(x_0)}\left(f(x_*)-f(x_0)-f'(x_0)(x_0-x_*)\right) \le \frac{L}{2|f'(x_0)|}|x_*-x_0|^2.$$

This strongly supports our previous claim that the Newton's method convergences quadratically, we only need to establish that $f'(x_k)$ stays away from zero, provided $f'(x_*) \neq 0$ and x_0 is sufficiently close to x_*.

6.4 Secant method

Recall

$$f'(x) = \lim_{h\to 0}\frac{f(x+h)-f(x)}{h}.$$

Thus if the derivatives of the function are not available, in the Newton's method we can replace $f'(x_k)$ by the finite difference

$$\frac{f(x_k+h_k)-f(x_k)}{h_k}$$

for some $h_k \neq 0$. It is natural to use $h_k = x_{k-1} - x_k$, then

$$\frac{f(x_k+h_k)-f(x_k)}{h_k} = \frac{f(x_{k-1})-f(x_k)}{x_{k-1}-x_k}.$$

This gives us the **Secant method**

$$x_{k+1} = x_k - f(x_k)\frac{x_{k-1}-x_k}{f(x_{k-1})-f(x_k)}. \tag{6.3}$$

```
Input: Initial values x_0 and x_1, tolerance tol,
maximum number of iterations maxit

Output: approximation of the root

1 For k = 1:maxit do
2 Compute s_k = -f(x_k)*(x_{k-1}-x_k))/(f(x_{k-1})-f(x_k)).
3 Compute x(k+1) = x(k) + s(k).
4 Check for truncation
5 End
```

We will see that the Secant method has very interesting properties. Recalling the divided difference formula (2.8),

$$f[x_j,\dots,x_k] = \frac{f[x_{j+1},\dots,x_k]-f[x_j,\dots,x_{k-1}]}{x_k-x_j} \tag{6.4}$$

we can rewrite (6.1) as

$$f[x,y] = \int_0^1 f'(y+t(x-y))\,dt. \tag{6.5}$$

We will also require the following estimate.

Lemma 6.1. *Let $f : D \to \mathbb{R}$ be differentiable on D and let furthermore $f'(x)$ be Lipschitz continuous with Lipschitz constant L. Then for any distinct $x,y,z \in D$*

$$f[x,y,z] \leq \frac{L}{2}.$$

Proof. From (6.4) and (6.5), and using that f' is Lipschitz continuous, we obtain

$$\begin{aligned} f[x,y,z] = \frac{f[y,z]-f[x,y]}{z-x} = \frac{f[z,y]-f[x,y]}{z-x} &= \frac{1}{z-x}\left(\int_0^1 \left[f'(y+t(z-y))-f'(y+t(x-y))\right]dt\right) \\ &\leq \frac{L}{|z-x|}\int_0^1 t\,dt|z-x| = \frac{L}{2}. \end{aligned}$$

Now we turn back to the secant method (6.3). Using the divided difference notation, we can rewrite it as

$$x_{k+1} = x_k - \frac{f(x_k)}{f[x_{k-1},x_k]}. \tag{6.6}$$

Using that x_* is a root, i.e. $f(x_*) = 0$, we have

$$\begin{aligned} x_{k+1}-x_* &= x_k - x_* - \frac{f(x_k)}{f[x_{k-1},x_k]} \\ &= (x_k-x_*)\left(1-\frac{f(x_k)}{(x_k-x_*)f[x_{k-1},x_k]}\right) \\ &= \frac{x_k-x_*}{f[x_{k-1},x_k]}\left(f[x_{k-1},x_k]-\frac{f(x_k)-f(x_*)}{(x_k-x_*)}\right) \\ &= \frac{x_k-x_*}{f[x_{k-1},x_k]}\left(f[x_{k-1},x_k]-f[x_k,x_*]\right) \\ &= \frac{(x_k-x_*)(x_{k-1}-x_*)}{f[x_{k-1},x_k]}\frac{f[x_{k-1},x_k]-f[x_k,x_*]}{x_{k-1}-x_*} \\ &= (x_k-x_*)(x_{k-1}-x_*)\frac{f[x_{k-1},x_k,x_*]}{f[x_{k-1},x_k]}. \end{aligned} \tag{6.7}$$

Now similarly to the Theorem 6.1, we show

Theorem 6.2. *Let $D \in \mathbb{R}$ be an open interval and let $f : D \to \mathbb{R}$ be differentiable on D Furthermore, let $f'(x)$ be Lipschitz continuous with Lipschitz constant L. If $x_* \in D$ is a root. In addition, assume*

$$\min_{a\le x\le b} |f'(x)| = m > 0.$$

Let ε be such that

$$q := \frac{L}{2m}\varepsilon < 1.$$

Then the Secant method with starting points $x_0, x_1 \in B_\varepsilon(x_*)$, generates iterates $x_k \in B_\varepsilon(x_*)$ which converge to x_*,

$$\lim_{k\to\infty} x_k = x_*,$$

and which obey

$$|x_{k+1}-x_*| \le q^{\lambda_k} \quad \textit{for all} \quad k \in \mathbb{N},$$

where λ_k are the elements of the Fibonnachi sequence

$$\lambda_{k+1} = \lambda_k + \lambda_{k-1},$$

i.e.

$$\lambda_k = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^k - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^k \sim \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^k, \quad \textit{as} \quad k\to\infty.$$

Proof. Using the Mean Value Theorem and the assumption of the theorem be obtain

$$f[x_{k-1},x_k] = |f'(\xi_k)| \ge m, \quad \text{for some} \quad \xi_k \in (x_{k-1},x_k).$$

Together with Lemma 6.1, from (6.7), we obtain

$$|x_{k+1}-x_*| \le \frac{L}{2m}|x_k-x_*||x_{k-1}-x_*|. \tag{6.8}$$

Thus if $x_{k-1}, x_k \in B_\varepsilon(x_*)$, then from the above estimate, and since $\frac{L\varepsilon}{2m} < 1$, we see that

$$|x_{k+1}-x_*| \le \frac{L}{2m}\varepsilon\varepsilon < \varepsilon,$$

i.e. $x_{k+1} \in B_\varepsilon(x_*)$ and as a result $\{x_k\} \in B_\varepsilon(x_*)$ if $x_0, x_1 \in B_\varepsilon(x_*)$.

Set $e_k = \frac{L}{2m}|x_k - x_*|$. Then (6.8), we can rewrite as

$$e_{k+1} \le e_k e_{k-1}, \quad k = 1, 2, \ldots.$$

Since $e_0 \le q$ and $e_1 \le q$, we have $e_k \le q^{\lambda_k}$ for $k = 1, 2, \ldots$. Since $q < 1$ and $\lambda_k \to \infty$ and $k \to \infty$, we have $e_k \to 0$ as $k \to \infty$. This concludes the proof of the theorem.

We can also obtain a posteriori error estimate.

Corollary 6.1. *Under the assumptions of Theorem 6.2, we have*

$$|x_k - x_*| \le \frac{1}{m}|f(x_k)| \le \frac{L}{2m}|x_k - x_{k-1}||x_k - x_{k-2}|.$$

Proof. Using the Mean Value Theorem and low bound for the derivative of the function we obtain

$$|x_k - x_*| \le \frac{1}{m}|f(x_k) - f(x_*)| = \frac{1}{m}|f(x_k)| = \frac{1}{m}\left|f(x_{k-1}) + (x_k - x_{k-1})\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}\right|.$$

From definition of the Secant method

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

we have

$$f(x_{k-1}) = -(x_k - x_{k-1})\frac{f(x_{k-1}) - f(x_{k-2})}{x_{k-1} - x_{k-2}}.$$

Combining it with the above estimate and using Lemma 6.1, we obtain

$$\begin{aligned}
|x_k - x_*| &\le \frac{|x_k - x_{k-1}|}{m}\left|\frac{f(x_{k-1})}{x_k - x_{k-1}} + \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}\right| \\
&= \frac{|x_k - x_{k-1}|}{m}\left|\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} - \frac{f(x_{k-1}) - f(x_{k-2})}{x_{k-1} - x_{k-2}}\right| \\
&= \frac{|x_k - x_{k-1}||x_k - x_{k-2}|}{m}|f[x_{k-2}, x_{k-1}, x_k]| \\
&\le \frac{L}{2m}|x_k - x_{k-1}||x_k - x_{k-2}|.
\end{aligned}$$

References

1. Michael L. Overton. *Numerical computing with IEEE floating point arithmetic*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001. Including one theorem, one rule of thumb, and one hundred and one exercises.
2. Volker Strassen. Gaussian elimination is not optimal. *Numer. Math.*, 13:354–356, 1969.