Lecture Notes on Essential Numerical Analysis

***** DRAFT *****

*** UNDER CONSTRUCTION ***

please keep checking for new changes

Jan Mandel

August 27, 2003

Contents

Chapter 1

Introduction

These are notes for my graduate class "Numerical Analysis," which I have been teaching at the University of Colorado at Denver. This class serves graduate students in Mathematics as well as in Computer Science, with varying degrees of mathematical preparation; yet the class needs to be at the graduate mathematics level.

The objective of these notes is to present important parts of Numerical Analysis in a concise form that corresponds to what I actually presented in the classroom. Proofs are selected to provide insight and connections rather than to cover all material with mathematical rigor. A separate part of the book reviews prerequisites from Linear Algebra and Calculus , which grow naturally into a presentation of the elements of the theory of linear spaces, essential for work in modern Numerical Analysis. Only what is actually needed for the numerical analysis is presented.

Part I

Numerical Analysis

Chapter 2

Computer Arithmetic and Rounding Errors

2.1 Representation of integers

Bits and bytes Data in a computer are represented as binary digits (bits). A bit can attain the value of 0 or 1. The memory of contemporary computers is organized in bytes, which are groups of 8 bits. The memory is a linear sequence of bytes starting with byte 0. Every byte is identified by its address, which is an integer. Memory is measured in bytes and their multiples. One kilobyte (kB) is $2^{10} = 1024 \approx 10^3$ bytes, megabyte (MB) is $2^{20} = 1024^2 \approx 10^6$ bytes, and gigabyte (GB) is $2^{30} = 1024^3 \approx 10^9$ bytes.

Binary numbers and unsigned integers Integers are represented as binary numbers:

$$(a_0 a_1 \dots a_{n-1})_2 = 2^n a_0 + 2^{n-1} a + \dots + a_{n-1}, \tag{2.1}$$

Integers stored in n bits according to (2.1) attain values from 0 to

$$(\underbrace{11\dots1}_{n})_2 = 2^n - 1.$$

This is the *unsigned integer* type.

Signed integers Negative numbers are usually represented using the complementary storage format: the n bit binary number $x = (a_0 a_1 \dots a_{n-1})_2$ represents $x - 2^n$ if $a_0 = 1$. The advantage of the complementary storage format is that operations on such integers are simply operations modulo 2^n. This means that two integers different by an integer multiple of 2^n are considered equivalent. Algebraic operations in the complementary format are performed exactly as for unsigned integers and nonzero bits carried to the left beyond a_0 are discarded. This results in simple design and fast operation of the computer circuits. The maximal number that can be represented is $(011\dots1)_2 = 2^{n-1}-1$. The minimal number is represented by $(100\dots0)_2 - 2^n = 2^{n-1} - 2^n = -2^{n-1}$.

Integer overflow is the situation when the result of an integer operation falls outside of the integer range. In the past, integer overflow usually caused the program to abort with an error message. Today, integer overflow is usually ignored and the operation silently performed modulo 2^n. This is a common source of hard to find run-time errors. Sometimes it is possible to force the computer to watch for integer overflow by using compiler flags, but the resulting program may execute much more slowly.

The most common integer representation has 8 bytes = 32 bits. Signed 32 bit integers range from $-2^{31} \approx -2.1 \times 10^{10}$ to $2^{31} - 1 \approx 2.1 \times 10^{10}$. Because the amount of memory that 32 bit integers can address is only 2GB (signed) or 4GB (unsigned), the computer industry is moving towards the

use of 64 bit integers. Major computer vendors such as SUN and IBM changed their processor lines to 64 bits many years ago. Their CPUs use 64 bit integers to address memory and usually support 64 bit integer operations in hardware. The PC architecture is now in transition. Without 64 bit integers, a programmer has to pay special care even for the size of problems quite common now.

Example 1 *To compute the percentage of nonzero entries in a sparse matrix of order m by n with k nonzeros, a program uses the expression*

$$\mathtt{p} = (\mathtt{100.0} * \mathtt{k})/(\mathtt{m} * \mathtt{n}).$$

Suppose m and n are integers, integer arithmetic has 32 bits, and $m = n = 10^7$. Then $m \times n = 10^{14} > 2^{31} - 1$, therefore the result of the multiplication will be incorrect.

Exercise 2 *Rewrite the code segment in Example 1 to avoid the integer overflow.*

2.2 Representation of reals in IEEE arithmetics

We will cover IEEE arithmetics only, because of its prevalence in computer industry. The generic form of IEEE standard representation is

$$\begin{array}{ll} \text{value} & \text{bit string} \\ (-1)^s 1.a_1 \dots a_n 2^{b_1 \dots b_m - f} & \underbrace{s}_{(-1)^s} \underbrace{b_1 \dots b_m}_{2^{b_1 \dots b_m - f}} \underbrace{a_1 \dots a_n}_{1.a_1 \dots a_n} \end{array} \tag{2.2}$$

The bit s is the sign bit, $b_1 \dots b_m$ is the exponent, and $a_1 \dots a_n$ is the significant.The cases when all bits $b_1 \dots b_m$ are zero or all $b_1 \dots b_m$ are one are special.

precision	total bits	sign	exponent	significant	offset
single	32	1	$m = 8$	$n = 23$	$f = 127$
double	64	1	$m = 11$	$n = 52$	$f = 1023$

Here are some examples.

value	bit string
1	$\underbrace{0}_{(-1)^0}\ \underbrace{01111111111}_{2^{1023-1023}}\ \underbrace{0000000000000000000000000000000000000000000000000000}_{1}$
2	$\underbrace{0}_{(-1)^0}\ \underbrace{10000000000}_{2^{1024-1023}}\ \underbrace{0000000000000000000000000000000000000000000000000000}_{1}$
-1	$\underbrace{-1}_{(-1)^1}\ \underbrace{01111111111}_{2^{1023-1023}}\ \underbrace{0000000000000000000000000000000000000000000000000000}_{1}$
3/4	$\underbrace{0}_{(-1)^0}\ \underbrace{01111111110}_{2^{1022-1023}}\ \underbrace{1000000000000000000000000000000000000000000000000000}_{1\frac{1}{2}}$

The largest and the smallest positive numbers that can be represented in this format are

value	bit string
$\approx 2^{1024} \approx$`1.8e+308`	$\underbrace{0}_{(-1)^0}\ \underbrace{11111111110}_{2^{2046-1023}}\ \underbrace{1111111111111111111111111111111111111111111111111111}_{\frac{1-1/2^{53}}{1-1/2}}$
$2^{-1022} \approx$`2.2e-308`	$\underbrace{0}_{(-1)^0}\ \underbrace{00000000001}_{2^{1-1023}}\ \underbrace{0000000000000000000000000000000000000000000000000000}_{1}$

Some of the special values are

value	bit string
0	00000000000000000000000000000000000000000000000000000000000000000
-0	10000000000000000000000000000000000000000000000000000000000000000
Inf	$0\underbrace{11111111111}_{\text{all bits are 1}}00000000000000000000000000000000000000000000000000$
$-Inf$	$1\underbrace{11111111111}_{\text{all bits are 1}}00000000000000000000000000000000000000000000000000$
NaN	$0\underbrace{11111111111}_{\text{all bits are 1}}$ at least one bit nonzero

The values of 0 and -0 equal. the values Inf, $-Inf$ (infinity and minus infinity) result from operations that overflow the floating point range:

$$\begin{aligned}(\texttt{1.5e+308}) * (\texttt{1.5e+308}) &= \texttt{Inf}\\ \texttt{1/0} &= \texttt{Inf}\\ \texttt{-1/0} &= \texttt{-Inf}\end{aligned}$$

and the value NaN (not a number) is the result of operations with an invalid result, for example

$$\begin{aligned}0/0 &= NaN\\ Inf - Inf &= Nan\end{aligned}$$

Floating point numbers will all zeros in the exponent are subnormalized numbers:

value	bit string
$(-1)^s 0.a_1 \dots a_n 2^{-f+1}$	$\underbrace{s}_{(-1)^s}\underbrace{0\dots0}_{2^{-f}}\underbrace{a_1\dots a_n}_{0.a_1\dots a_n}$

for example

value	bit string
2^{-1023}	$\underbrace{0}_{(-1)^0}\underbrace{00000000000}_{2^{-1022}}\underbrace{10000000000000000000000000000000000000000000000000}_{(0.1)_2=1/2}$

Try the following computer experiment. Initialize $\varepsilon = 1$,and as long as the expression $1+\varepsilon > 1$ is true, set $\varepsilon = \varepsilon/2$. After a while, this process stops, and you obtain a number t such that, in the computer, $1+\varepsilon = 1$, while $1+2\varepsilon > 1$. Clearly, $1+2\varepsilon$ is different from 1 only in the last bit of the mantissa: if the floating point representation is

value	bit string
$(-1)^s 1.a_1 \dots a_n 2^{b_1\dots b_m - f}$	$\underbrace{s}_{(-1)^s}\underbrace{b_1\dots b_m}_{2^{b_1\dots b_m-f}}\underbrace{a_1\dots a_n}_{1.a_1\dots a_n}$

then

$$1+2\varepsilon = (1.0\dots01)_2 = 1+2^{-n}.$$

In double precision IEEE arithmetics,

value	bit string
$1+2^{-52}$	$\underbrace{0}_{(-1)^0}\underbrace{01111111111}_{2^{1023-1023}}\underbrace{0000000000000000000000000000000000000000000000000001}_{1}$

Denote by $\oplus$, $\ominus$, $\otimes$ and $\oslash$ operations on floating point numbers in a floating point arithmetic. We will discuss implementation of these operations later.

Definition 3 *In any computer arithmetics, the number* `eps` *is the smallest floating point number such that*

$$1 \oplus \texttt{eps} > 1$$

We have shown above that

$$\texttt{eps} = 2^{-n} \tag{2.3}$$

In in IEEE double precision,

$$\texttt{eps} = 2^{-52} \approx 2.2204e-016,$$

which is also the value of the Matlab built-in function `eps`.

2.3 Rounding

Numbers that have more digits than can be represented in the floating point format have to be rounded. We will consider only rouding to nearest. Then, for a real number x, the result of rounding is the floating point number $fl(x)$, defined as the floating point number nearest to x. In IEEE arithmetics, we break the tie by rounding to even result.

Here is how to find.the normalized floating point number $(-1)^s 1.a_1 \dots a_n 2^{b_1 \dots b_m - f}$, cf., (2.2), nearest to $x \neq 0$, with rounding to even.

Clearly, $s = 1$ if $x < 0$ and $s = 0$ if $x > 0$. Then find the exponent M so that

$$1 \leq \left| \frac{x}{2^M} \right| < 2$$

and set $(b_1 \dots b_m)_2 - f = M$. If such a binary number does not exist, x is outside of the range of the floating point arithmetics. Now the fractional part has the in general infinite binary representation

$$\left| \frac{x}{2^M} \right| = 1.a_1 a_2 \dots a_n a_{n+1} \dots \tag{2.4}$$

If $a_{n+1} = a_{n+1} = \dots = 0$, clearly $fl(x) = 0$. Otherwise, the floating point neighbors of the number x are

$$\begin{aligned} x_- &= 2^M (1.a_1 a_2 \dots a_n)_2 \\ x_+ &= 2^M \left((1.a_1 a_2 \dots a_n)_2 + 2^{-n} \right) \end{aligned}$$

The errors are

$$\begin{aligned} |x - x_-| &= 2^{M-n} (0.a_{n+1} a_{n+2} \dots)_2 \\ |x - x_+| &= 2^{M-n} (1 - (0.a_{n+1} a_{n+2} \dots)_2) \end{aligned}$$

If $a_{n+1} = 1$ and $a_{n+2} = a_{n+3} = \dots$, the errors are same and we choose $fl(x) = x_-$ if $a_n = 0$ and $fl(x) = x_+$ if $a_n = 1$, to make the last bit of the result zero (break tie by rounding to even). Otherwise, we choose $fl(x) = x_-$ if $a_{n+1} = 0$ and $fl(x) = x_+$ if $a_{n+1} = 1$. In any case,

$$|x - fl(x)| \leq 2^{M-n}/2 = 2^{M-(n+1)} \tag{2.5}$$

We get the following theorem on relative error of rounding.

Theorem 4 *If $|x|$ is within the range of the normalized floating point arithmetics (that is, between the smallest positive normalized floating point number and the largest floating point number), then*

$$\left| \frac{x - fl(x)}{x} \right| \leq \frac{\texttt{eps}}{2}$$

Proof. From (2.4), it follows that

$$|x| \geq 2^M.$$

Unsing (2.5), we get

$$\left|\frac{x - fl(x)}{x}\right| \leq \frac{2^{M-(n+1)}}{2^M} = 2^{-(n+1)} = \frac{\texttt{eps}}{2}$$

from (2.3). ■

Computer rounding satisfies for any real number x in the range of the normalized floating point arithmetic

$$fl(x) = x(1+\delta), \quad |\delta| \leq \epsilon, \tag{2.6}$$

where ϵ is the *rounding precision*. For IEEE double precision, $\epsilon = 2^{-53} \approx 1.1 \times 10^{-16}$.

For more about floating point computations and IEEE arithmetics in particular, see very nice and clear booklet [Ove01].

Review of Taylor and binomial theorems

See section 14.1.

2.4 Floating point operations and error analysis

We will assume that the result of floating point operations satisfies

$$x \oplus y = (x+y)(1+\delta), \quad |\delta| \leq \epsilon, \tag{2.7}$$

and similarly for the operations $\ominus$, $\otimes$ and $\oslash$. Here, ϵ is the rounding precision of the computer, see (2.6). The bound (2.7) is satisfied on virtually all modern computers.

In particular, (2.7) is satisfied when the result of a floating point operation is the exact result rounded to the precision of the computer, that is, for floating point numbers x and y,

$$x \oplus y = \mathrm{fl}(x+y) \tag{2.8}$$

and similarly for the other operations.

To implement floating point operation, the processor computes the result in *extended precision registers*, which is then rounded. In IEEE arithmetics, the extended precision is 80 bits. Typically, intermediate results are kept in extended precision registers and only the final result is rounded when stored in memory, so floating point operations are sometimes more accurate. This is usually beneficial but it may break some software that relies on strict conformance of the arithmetic with (2.8). Compiler flags can be used to require strict conformance with (2.7), but the program is then usually slower, because intermediate results must be written to memory or rounded.

Exercise 5 *Show that (2.8) implies that ϵ is the smallest positive floating point number such that $1 + 2\epsilon > 1$.*

The estimate (2.7) can be extended to a series of operations. For example:

Theorem 6 *Let $S = \sum_{i=0}^{n} x_i$ be calculated by floating point arithmetic as $S^* = (\ldots((x_0 \oplus x_1) \oplus x_2) \cdots \oplus x_n)$, and assume that $n\epsilon < 1$. Then*

$$\begin{aligned} |S^* - S| &\leq ((1+\epsilon)^n - 1) \sum_{i=0}^{n} |x_i| \\ &\leq (n\epsilon + (e-2)(n\epsilon)^2) \sum_{i=0}^{n} |x_i| \approx n\epsilon \sum_{i=0}^{n} |x_i|. \end{aligned}$$

Proof. From (2.7), we have, with $|\delta_i| \leq \epsilon$,

$$\begin{aligned}
x_0 \oplus x_1 &= (x_0 + x_1)(1+\delta_1) \\
(x_0 \oplus x_1) \oplus x_2 &= ((x_0 + x_1)(1+\delta_1) + x_2)(1+\delta_2) \\
&\vdots \\
(\ldots((x_0 \oplus x_1) \oplus x_2) \cdots \oplus x_n) &= (\ldots(x_0 + x_1)(1+\delta_1) + \cdots + x_n)(1+\delta_n) \\
&= x_0(1+\delta_1)\ldots(1+\delta_n) \\
&\quad + x_1(1+\delta_1)\ldots(1+\delta_n) \\
&\quad + x_2(1+\delta_2)\ldots(1+\delta_n) \\
&\vdots \\
&\quad + x_{n-1}(1+\delta_{n-1})(1+\delta_n) \\
&\quad + x_n(1+\delta_n)
\end{aligned}$$

Using the definition of S, it follows that

$$\begin{aligned}
S^* - S &= x_0((1+\delta_1)\ldots(1+\delta_n) - 1) \\
&\quad + x_1((1+\delta_1)\ldots(1+\delta_n) - 1) \\
&\quad + x_2((1+\delta_2)\ldots(1+\delta_n) - 1) \\
&\vdots \\
&\quad + x_n((1+\delta_n) - 1),
\end{aligned}$$

hence

$$|S^* - S| \leq \sum_{i=0}^{n} |x_i| \max\{(1+\epsilon)^n - 1, 1 - (1-\epsilon)^n\}. \tag{2.9}$$

From the binomial theorem,

$$(1+\epsilon)^n = 1 + n\epsilon + \frac{n(n-1)}{2!}\epsilon^2 + \cdots + \frac{n(n-1)(n-2)\ldots 1}{n!}\epsilon^n. \tag{2.10}$$

Using in (2.10) $-\epsilon$ in place of ϵ and adding the two equations, we obtain

$$(1+\epsilon)^n + (1-\epsilon)^n \geq 2$$

because only positive terms in ϵ remain, which implies that

$$1 - (1-\epsilon)^n \leq (1+\epsilon)^n - 1. \tag{2.11}$$

Using (2.10) again, we have

$$(1+\epsilon)^n - 1 = 1 + n\epsilon + R,$$

where

$$\begin{aligned}
0 \leq R &\leq \frac{(n\epsilon)^2}{2!} + \cdots + \frac{(n\epsilon)^n}{n!} \\
&\leq (n\epsilon)^2 \left(\frac{1}{2!} + \cdots + \frac{1}{n!}\right) \\
&< (n\epsilon)^2 (e-2).
\end{aligned}$$

This bound together with (2.9) and (2.11) concludes the proof. ∎

2.5 Backwards error analysis

Analysis of rounding errors gets quickly complicated, and it is not clear if the errors are acceptable or not. A large rounding error may mean badly organized computations, but it also may be the best we can do if a similar difference can be caused by changing the input data only by a factor of ϵ. Also, rounding errors are not the only errors in numerical computations; more often than not, the data themselves are uncertain.

The technique of backward error analysis is based on the idea that *the result of numerical computation on the data can interpreted as the result of exact computations on perturbed data.* For example, we can write (2.7) in the form

$$x \oplus y = (x+y)(1+\delta) = x(1+\delta) + y(1+\delta). \tag{2.12}$$

That is, the floating point sum of two floating point numbers x and y is understood as the *exact* sum of *perturbed* numbers $x(1+\delta)$ and $y(1+\delta)$.

Consider again the sum of $n+1$ numbers as in Theorem 6. From the analysis in the proof of Theorem 6, we have

$$S^* = (\dots((x_0 \oplus x_1) \oplus x_2) \cdots \oplus x_n) = x_0(1+\bar{\delta}_0) + x_1(1+\bar{\delta}_2) + \cdots + x_n(1+\bar{\delta}_n),$$

where

$$\begin{aligned}
1+\bar{\delta}_0 &= (1+\delta_1)\dots(1+\delta_n) \\
1+\bar{\delta}_1 &= (1+\delta_1)\dots(1+\delta_n) \\
1+\bar{\delta}_2 &= (1+\delta_2)\dots(1+\delta_n) \\
&\vdots \\
1+\bar{\delta}_{n-1} &= (1+\delta_{n-1})(1+\delta_n) \\
1+\bar{\delta}_n &= (1+\delta_n).
\end{aligned}$$

Using the same reasoning as in the proof, we have $|\bar{\delta}_i| \le (1+\epsilon)^n - 1 \approx n\epsilon$. We can conclude that the effect of rounding errors in adding $n+1$ number in floating point arithmetic is no worse than perturbing each input number by at most $n\epsilon$ and performing the summation exactly.

Derive similar bounds for $\prod_{i=0}^{n} x_i$. Let us start with the case $n = 1$:

$$\begin{aligned}
(x_0 \otimes x_1) &= x_0 x_1 (1+\delta) \\
&= x_0(x_1(1+\delta)) \\
&= (x_0(1+\delta_0))(x_1(1+\delta_1))
\end{aligned}$$

where $|\delta| \le \varepsilon, \delta_0 = \delta_1$, so from Taylor theorem,

$$(1+\delta_0)^2 = 1+\delta \Rightarrow \delta_0 = \sqrt{1+\delta} - 1 \approx \frac{\delta}{2} \Rightarrow \delta_0 \lesssim \frac{\varepsilon}{2}$$

In the general case, expressing $p = (\cdots(x_0 \otimes x_1) \otimes x_2 \cdots) \otimes x_n$ as exact operation on perturbed data,

$$\begin{aligned}
p &= (x_0 x_1 (1+\delta_1)) x_2 (1+\delta_2) \cdots) x_n (1+\delta_n) \\
&= x_0 x_1 x_2 \cdots x_n \underbrace{(1+\delta_1)\cdots(1+\delta_n)}_{=(1+\bar{\delta})^n} \\
&= (x_0(1+\bar{\delta}))(x_1(1+\bar{\delta}))\cdots(x_n(1+\bar{\delta}))
\end{aligned}$$

where

$$\begin{aligned}
1+\bar{\delta} &= \sqrt[n+1]{(1+\delta_1)(1+\delta_2)\cdots(1+\delta_n)}, \\
|\bar{\delta}| &\le \max\left\{ \sqrt[n+1]{(1+\epsilon)^n} - 1, 1 - \sqrt[n+1]{(1-\epsilon)^n} \right\} \approx \frac{n}{n+1}\varepsilon,
\end{aligned}$$

similarly as in the previous example.

Exercise 7 *Derive a similar bound on* $\sum_{i=0}^{n} x_i y_i$.

To use the backwards error analysis to bound the error of the result of the computation, we need to know what is the sensitivity ofthe result to a small change in the input data. This is the subject of the following chapter.

Chapter 3

Condition Numbers

We now estimate the effect of a perturbation of input data.

3.1 Loss of significance

Suppose we are given $\overline{x} > \overline{y} > 0$, known to relative accuracy δ. The numbers actually stored in the computer are

$$x = \overline{x}(1+\delta_1) \quad y = \overline{y}(1+\delta_2) \quad |\delta_1|, |\delta_2| \leq \delta$$

Then

$$\begin{aligned} x - y &= \overline{x}(1+\delta_1) - \overline{y}(1+\delta_2) \\ &= \overline{x} - \overline{y} + \overline{x}\delta_1 - \overline{y}\delta_2 \\ &= (\overline{x} - \overline{y})\left(1 + \frac{\overline{x}\delta_1 - \overline{y}\delta_2}{\overline{x} - \overline{y}}\right) \\ &= (\overline{x} - \overline{y})(1+h), \qquad |h| \leq \frac{2}{1 - \frac{\overline{y}}{\overline{x}}}\delta \end{aligned}$$

because using $\overline{x} > \overline{y} > 0$, we have

$$\left|\frac{\overline{x}\delta_1 - \overline{y}\delta_2}{\overline{x} - \overline{y}}\right| \leq \frac{2\overline{x}\delta}{\overline{x} - \overline{y}} = \frac{2}{1 - \frac{\overline{y}}{\overline{x}}}\delta$$

Example 8 *Let $y = \sqrt{1+x^2} - 1$, $x = 10^{-5}$. Suppose$\sqrt{1+x^2}$ is known with relative accuracy $|\delta| \leq \varepsilon \approx 2 \cdot 10^{-16}$. Then the relative accuracy of $\sqrt{1+x^2} - 1$ is bounded by $\frac{2 \cdot 10^{-16}}{\frac{1}{2}10^{-10}} = 4 \cdot 10^{-6}$. Indeed, $1 - \frac{\overline{y}}{\overline{x}}$ becomes*

$$1 - \frac{\sqrt{1+x^2}}{1} \approx 1 - \frac{1}{1 + \frac{1}{2}x^2} \approx 1 - \left(1 - \frac{1}{2}x^2\right) = \frac{1}{2}x^2$$

using Taylor expansion

$$\sqrt{1+x} = 1 + \frac{x}{2} + \frac{\frac{1}{2}(\frac{1}{2}-1)}{2}x^2 + \frac{\frac{1}{2}(\frac{1}{2}-1)(\frac{1}{2}-2)}{3!}x^3 \cdots$$

A better way to evaluate this expression is

$$\begin{aligned} \sqrt{1+x^2} - 1 &= \frac{(\sqrt{1+x^2}-1)(\sqrt{1+x^2}+1)}{\sqrt{1+x^2}-1} \\ &= \frac{x^2}{\sqrt{1+x^2}+1}. \end{aligned}$$

3.2 Condition number of evaluation of an expression

Condition number is the amplification factor of a small relative change of data. From the previous section, condition number of the operation $(x, y) \mapsto x - y$, where $x \approx y$, is

$$\kappa = \left| \frac{2}{1 - \frac{y}{x}} \right| = \frac{2|x|}{|x - y|}.$$

The condition number is usually denoted by κ.

For function evaluation $f(x)$, from Taylor's polynomial of order one,

$$f(x + \delta) \approx f(x) + f'(x)\delta$$

Then

$$\underbrace{\frac{f(x+\delta) - f(x)}{f(x)}}_{\text{relative change in } f(x)} \approx \frac{f'(x)\delta}{f(x)} = \frac{xf'(x)}{f(x)} \overbrace{\frac{\delta}{x}}^{\text{relative change in } x}$$

The condition numer is

$$\kappa = \left| \frac{xf'(x)}{f(x)} \right|.$$

Example 9 $f(x) = x^\alpha \quad x \to 0, \quad x > 0$:

$$\kappa = \left| \frac{xf'(x)}{f(x)} \right| = \left| \frac{x\alpha x^{\alpha-1}}{x^\alpha} \right| = \alpha$$

Example 10 $f(x) = \arcsin x, \quad x \to 1, x < 1$:

$$\kappa = \left| \frac{xf'(x)}{f(x)} \right| = \left| \frac{x\frac{1}{\sqrt{1-x^2}}}{\arcsin x} \right| = \left| \frac{x}{\sqrt{1-x^2}\arcsin x} \right| \to \infty \qquad x \to 1$$

Example 11 $f(x) = x - 1, \quad x \to 1$:

$$\kappa = \left| \frac{x \cdot 1}{x - 1} \right| \to \infty$$

Amazingly, even a simple operation like $x - 1$ is ill-conditioned in the case of loss of accuracy.

Review of linear spaces and norms

See Section 15.1.

3.3 Condition number of matrix-vector multiply

Consider the mapping $x \longmapsto Ax - y$, where

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} in\mathbb{R}^n, \quad A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \dots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \in \mathbb{R}^{m\times n}, \quad y = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ y_m \end{pmatrix} \in \mathbb{R}^m$$

Instead of absolute value, the size of a vector is measured by a norm $\|x\|$. Let $\|A\|$ be the corresponding matrix norm. If

$$x = \overline{x} + \rho, \qquad \overline{x} \in \mathbb{R}^n, \qquad \rho \in \mathbb{R}^n, \qquad \frac{\|\rho\|}{\|\overline{x}\|} \leq \delta$$

then the relative change in the result is

$$\frac{\|Ax - A\overline{x}\|}{\|A\overline{x} - y\|} = \frac{\|A(x-\overline{x})\|}{\|A\overline{x} - y\|} \leq \frac{\|A\|\,\|(x-\overline{x})\|}{\|A\overline{x} - y\|} = \underbrace{\|A\| \frac{\|\overline{x}\|}{\|A\overline{x} - y\|}}_{\text{condition number}} \frac{\|\rho\|}{\|x\|}.$$

and get the condition number of evaluation of Ax as

$$\kappa(A, x) = \|A\| \frac{\|\overline{x}\|}{\|A\overline{x} - y\|}$$

Exercise 12 *Consider multiplication by the matrix $A = (1, -1)$ and derive the bound from the section on loss of significance.*

In the important case when A is square and nonsingular and $y = 0$, we use the inequality

$$\|\overline{x}\| = \left\|A^{-1}A\overline{x}\right\| \leq \left\|A^{-1}\right\| \|A\overline{x}\|$$

to get the bound on relative accuracy of $A\overline{x}$ as

$$\frac{\|Ax - A\overline{x}\|}{\|A\overline{x}\|} \leq \frac{\|A\|\|x-\overline{x}\|}{\frac{1}{\|A^{-1}\|_\infty}\|\overline{x}\|} = \underbrace{\|A\|\|A^{-1}\|}_{\text{condition number of } A} \frac{\|x-\overline{x}\|}{\|\overline{x}\|} = \kappa(A)\delta,$$

where

$$\kappa(A) = \|A\|\|A^{-1}\|$$

is the *condition number of the matrix* A. Note that $\kappa(A) = \kappa(A^{-1})$ and that the value of the condition number depends on what the underlying norm is. Usually, condition number of a matrix is introduced when discussing the solution of the linear system $Ax = b$. We see that the condition number is in fact related to any matrix-vector multiplication, rather than just solution of a linear system.

3.4 Stability of zeros of function

Suppose we are solving an equation of the form $f(x) = 0$. Because of rouding errors and because f may not be known exactly, the solution we actually get is the solution of a perturbed problem, with f replace by a slightly different f_δ. Because f is a function, a proper type of perturbation is by another function: f is replaced by $f_\delta = f + \delta g$, where g is a function and δ is a small parameter. Write the solution of the perturbed equation as $x + h$, and derive an estimate for h :

$$f_\delta(x+h) = \underbrace{f(x+h)}_{\approx f(x)+hf'(x)} + \underbrace{\delta g(x+h)}_{\approx \delta(g(x)+hg'(x))} = 0$$

so

$$f(x) + hf'(x) + \delta(g(x) + hg'(x)) \approx 0,$$

which gives

$$h \approx \frac{-\delta g(x)}{f'(x) + \delta g'(x)} \approx \frac{-\delta g(x)}{f'(x)}$$

if $\delta \approx 0$.

Example 13 *(Wilkinson) Consider perturbation of the polynomial*

$$f(x) = a_0 + \cdots + a_n x^n$$

by a change in its coefficients; then the perturbation function $g(x)$ is also a polynomial. Specifically, consider the change in the root $x = 20$ of the polynomial

$$f(x) = (x-1)(x-2)\cdots(x-20) = a_{20}x^{20} + a_{19}x^{19} + \cdots + a_0$$

when the leading coefficient $a_{20} = 1$ is replaced by $1 + \delta$. Then $g(x) = x^{20}$, and we have using the formula for the derivative of a product,

$$\begin{aligned}
(y_1 \cdots y_n)' &= y_1' y_2 \cdots y_n + y_1 y_2 \cdots y_n + \cdots + y_1 y_2 \cdots y_n' \\
f(x) &= (x-1)(x-2)\cdots(x-20) \\
f'(20) &= (x-1)(x-2)\cdots(x-19)\underbrace{(x-20)'}_{=1} = 19! \\
g(20) &= 20^{20}
\end{aligned}$$

so the root 20 becomes $20 + h$, where

$$h \approx \frac{-\delta 20^{20}}{19!} \approx -\delta 8.6e8$$

In a computer with double precision, $\delta \approx 10^{-16}$, so $h \approx 10^{-7}$.

3.5 Unstable calculations

Consider numerical evaluation of the recursion

$$x_{n+1} = \frac{13}{3}x_n - \frac{4}{3}x_{n-1} \quad x_0 = 1 \quad x_1 = \frac{1}{3}$$

It is easy to verify that $x_n = \frac{1}{3^n}$. However a numerical experiment shows something quite different; eventually the numbers x_n become very large. To see why, find the general solution of the above difference equation: With the assumed form of solution λ^n

$$\lambda^{n+1} = \frac{13}{3}\lambda^n - \frac{4}{3}\lambda^{n-1}$$

we get the *characteristic equation*

$$\lambda^2 = \frac{13}{3}\lambda - \frac{4}{3}$$

with the roots

$$\lambda = \left\{ \begin{array}{l} \frac{1}{3} \\ 4 \end{array} \right.$$

giving the general solution

$$x_n = A\left(\frac{1}{3}\right)^n + B4^n$$

The coefficients A, B are determined from known x_0, x_1. Even if $B = 0$ for our initial conditions, a small second component is created because of rounding errors, and eventually takes over.

Chapter 4

Solution of Nonlinear Equations

4.1 Bisection Method

Given a continuous function $a < b$, $sign(f(a)) \neq sign(f(b))$, $f(a) \neq 0$, $f(b) \neq 0$,to solve $f(x) = 0$: :
$a_0 = a_1$, $b_0 = b_1$
for $i = 1 \ldots n$
$c_2 = \frac{a_{i-1}+b_{i-1}}{2}$
if $f(c_1) = 0$, stop
if $sign\, f(c_1) = sign\, f((a_{i-1}))$ then
$a_i = c_i$, $b_i = b_{i-1}$
else $a_i = a_{i-1}$, $b_i = c_{i-1}$
end

The solution is in the interval (a_i, b_i) and $b_i - a_i = \frac{b_{i-1}-a_{i-1}}{2}$, so

$$b_n - a_n = \frac{1}{2^n}(b_0 - a_0)$$

For accuracy ϵ, we need at most $\log_2 \frac{b_0-a_0}{\varepsilon}$ steps One evaluation of f per step is required.

4.2 Newton's Method

Given f, initial x, we want to find h so that $f(x + h) = 0$. From Taylor's theorem:

$$f(x+h) \approx \underbrace{f(x) + hf'(x) = 0}_{h=-\frac{f(x)}{f'(x)}}$$

Solving for h gives $h = -\frac{f(x)}{f'(x)}$. Repeating this step, we obtain the iterative method

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

Assume $f(r) = 0$ and denote $e_n = x_n - r$

$$\begin{aligned} \text{definition of } x_{n+1} &\implies f(r) = f(x_n) + (x_{n+1} - x_n)f'(x_n) \\ \text{Taylor expansion} &\implies f(r) = f(x_n) + (r - x_n)f'(x_n) + \frac{1}{2}(r - x_n)^2 f"(\xi_n) \end{aligned}$$

Subtracting, we get

$$(x_{n+1} - x_n - r + x_n)f'(x_n) = \frac{1}{2}(r - x_n)^2 f"(\xi_n)$$

hence

$$e_{n+1} = \frac{1}{2} e_n^2 \frac{f"(\xi_n)}{f'(x_n)}$$

If $x_n \to r, f'(r) \neq 0$, f'' continuous at r, then

$$\frac{e_{n+1}}{e_n^2} \to \frac{1}{2} \frac{f"(r)}{f'(r)}$$

that is, quadratic convergence. Also we can expect convergence of the iterations if x_n is close enough to r, because then

$$e_{n+1} = e_n \left(e_n \frac{1}{2} \frac{f''(\xi_n)}{f(x_n)} \right) \Longrightarrow |e_{n+1}| < |e_n| \, \delta_n, \qquad \delta_n = \left| e_n \frac{1}{2} \frac{f''(\xi_n)}{f'(x_n)} \right| < \frac{1}{2}$$

Formulating this carefully leads to the following theorem.

Theorem 14 *Let f'' be continuous at $r, f(r) = 0$, $f'(r) \neq 0$. Then there exists $\varepsilon > 0$ such that for any x_0, $|x_0 - r| < \varepsilon$, it holds that $\lim_{n\to\infty} x_n = r$, and $|x_{n+1} - r| \leq c|x_n - r|^2$ for some c.*
Proof. *Because f'' is continuous at r, it is bounded on same neighborhood of r. Because $|f'(r)| > 0$ and f' is continuous on a neighborhood of r, we have $|f\prime(x| > \frac{1}{2}|f(r)|$ on some neighborhood of r. Then the function δ, defined by*

$$\delta(\epsilon) = \frac{1}{2} \frac{\max\limits_{|r-x|\leq\varepsilon} |f''(\xi)|}{\min\limits_{|r-x|\leq\varepsilon} |f'(x)|},$$

is bounded on some neighborhood of zero. So we can pick $\varepsilon > 0$ so that

$$\underbrace{\varepsilon\delta(\varepsilon)}_{\text{continuous and equal to } 0 \text{ at } 0} < \frac{1}{2}$$

Then, if $\varepsilon_0 < \varepsilon$, we have by induction

$$|\varepsilon_{n+1}| \leq |\varepsilon_n| \frac{1}{2} \Rightarrow \lim_{n\to\infty} \varepsilon_n = 0$$

and

$$|\varepsilon_{n+1}| \leq |\varepsilon_n||\varepsilon_n| \, |\delta(\varepsilon_n)| \, ,$$

so the theorem holds with $c = \max_{|t-r|\leq\varepsilon} \delta(t)$. ■

Example 15 *Newton's method applied to compute square root: $f(x) = x^2 - a$ is $x_{n+1} = F(x_n)$, where $F(x) = x - \frac{f(x)}{f'(x)} = x - \frac{x^2-a}{2x} = \frac{1}{2}(x + \frac{a}{x})$. This is the method used to compute square root in the microcode of many current CPUs. A good x_0 is found by normalizing the argument to a fixed interval, such as $(1, 4)$ by a bit shift and table lookup.*

4.3 Newton's Method for systems

A system of n equations in n unknowns can be written as $f(x) = 0$, where

$$f = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}, \qquad x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

Replacing in Newton's method the derivative by the Jacobian matrix,

$$f' = \begin{bmatrix} \frac{df_1}{dx_1} & \cdots & \frac{df_1}{dx_n} \\ \vdots & \ddots & \vdots \\ \frac{df_m}{dx_1} & \cdots & \frac{df_m}{df_n} \end{bmatrix}$$

and its reciprocal by the matrix inverse, we obtain formally the Newton's method for systems,

$$f(x^{n+1}) = x^n - (f'(x^{n+1})^{-1} f(x^n).$$

Review of Cauchy sequences and Banach contraction theorem

See Sec. **??** and 15.2.

4.4 Estimate of contraction number from derivatives.

Theorem 16 *If $F : S \to S \subset \mathbb{R}^n$, S convex, $\|\cdot\|$ a norm on $\mathbb{R}^n$, and $\|F'(x)\| \le q \ \forall x \in S$, then for any $x, y \in S$, $\|F(x) - F(y)\| \le q\|x - y\|$.*

Define

$$g(t) = F(x + t(y - x))$$

Then

$$F(y) - F(x) = g(1) - g(0) = \int_0^1 g'(t)dt$$

and by the chain rule

$$\begin{aligned} g'(t) &= \sum_{j=1}^n \frac{\partial F_i}{\partial x_j} \frac{\partial (x + t(y-x))_j}{\partial t} = \sum_{j=1}^n \frac{\partial F_i}{\partial x_j}(x + t(y-x))(y_j - x_j) \\ &= F'(x + t(y-x))(y - x), \end{aligned}$$

where

$$F' = \begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_n}{\partial x_1} & \cdots & \frac{\partial F_n}{\partial x_n} \end{pmatrix}.$$

So,

$$F(y) - F(x) = g(1) - g(0) = \int_0^1 g'(t)dt = \int_0^1 F'(x + t(y-x))(y-x)dt.$$

Because for integral of a vector function $f(t) \in \mathbb{R}^n$, it holds that $\| \int_b^a f(t)dt\| \le \int_b^a \|f(t)dt\|$, we obtain

$$\begin{aligned} \|F(y) - F(x)\| &\le \int_0^1 \| \overbrace{F'(x + t(y-x))}^{\text{matrix}} \overbrace{(y-x)}^{\text{vector}} \| dt \\ &\le \int_0^1 \|F'(x + t(y-x))\| \|y - x\| dt \le q\|y - x\|. \end{aligned}$$

4.5 Application of Banach contraction theorem to systems of equations

We can now apply the the Banach contraction theorem to show that functional iterations converge if the norm of the iteration function is small enough.

Theorem 17 *Let $F : \tilde{S} \to \tilde{S} \subset \mathbb{R}^n$, $\tilde{S}$ closed, $\|\cdot\|$ a norm on $\mathbb{R}^n$, and $\|F'(x)\| \leq q \ \forall x \in S$, where $\tilde{S} \subset S$, S is convex, and $q < 1$. Then F has exactly one fixed point in $\tilde{S}$, $x = F(x)$,, for any $x^{(0)}$, the sequence defined by $x^{(n+1)} = F(x^{(n)})$ converges to x, and*

$$||x^{(0)} - x|| \leq \frac{q^n}{1-q}||x^{(1)} - x^{(0)}||.$$

4.6 Local convergence and Newton's method

The next theorem shows that if the norm of the derivative at the fixed point is less than one, the iterations will converge to the fixed point if started close enough to it. This is called *local convergence*.

Theorem 18 *Let $F : S \to S \subset \mathbb{R}^n$, S open, $x^* = F(x^*) \in S$, $\|\cdot\|$ a norm on $\mathbb{R}^n$, F' continuous at x^*, and $||F'(x^*)|| < 1$. Then, for $x^{(0)}$ close enough to x^*, the sequence defined by $x^{(n+1)} = F(x^{(n)})$ converges to x^*, and $||x^{(n)} - x^*|| \leq q^n||x^{(0)} - x^*||$.*
Proof. *Because F' is continuous at x^*, $||F'(x)|| \leq \frac{1+q}{2} < 1$ for x close enough to x^*, that is, $||x - x^*|| < \varepsilon$, for some $\varepsilon > 0$. Then, if $||x^{(0)} - x^*|| < \varepsilon$, then $||x^{(1)} - x^*|| \leq q||x^{(0)} - x^*||$, $||x^{(2)} - x^*|| \leq q||x^{(1)} - x^*|| \leq q^2||x^{(0)} - x^*||$, etc.* ■

In the 1D case, Newton's method is defined by the iterations $x_{n+1} = F(x_n)$, where

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

Note that

$$F'(x) = 1 - \frac{(f'(x))^2 - f(x)f'(x)}{(f'(x))^*} = f(x)f''(x),$$

so $F(x^*) = 0$, $F'(x^*) = 0$, hence the iterations converge faster than geometric sequence with any positive quotient. Recall that Newton's method is extended to more than 1D by analogy: $x_{n+1} = F(x_n)$, $F(x) = x - (f^{'}(x))^{-1}f(x)$.

Theorem 19 *If $f'(x^*)$ is nonsingular and all partial derivatives of order 2 of f are continuous at x^*, then the derivative of the iteration function F from the Newton's method is continuous at x^*, and it satisfies $F'(x^*) = 0$.*
Proof. *Denote the entries of the matrix $(f''(x))^{-1}$ by $(f^{'}(x))^{-1}_{ij}$. Then $F_i(x) = x_i - \sum_{k=1}^{n}(f''(x))^{-1}_{ik}f_k(x)$, and using the rule for derivative of product,*

$$F'_{ij}(x) = \frac{\partial}{\partial x_j}F_i(x) = \delta_{ij} - \left(\sum_{k=1}^{n} f_k(x)\frac{\partial}{\partial x_j}(f^{'}(x))^{-1}_{ik} + \sum_{k=1}^{n}(f^{'}(x))^{-1}_{ik}\frac{\partial}{\partial x_j}f_k(x)\right).$$

To show that the first sum is zero at $x = x^*$: From Cramer's rule, the entries of the inverse of a matrix is are rational functions of its coefficient with the numerator nonzero when the matrix is nonsingular; consequently, because $f'(x^*)$ is nonsingular, the entries of $(f^{'}(x))^{-1}_{ik}$ are continuously differentiable at $x = x^*$from the chain rule, hence $\frac{\partial}{\partial x_j}(f^{'}(x))^{-1}_{ik}$are continuous at $x = x^*$. Further, since $f(x^*) = 0$, it holds that $f_k(x^*)\frac{\partial}{\partial x_j}(f^{'}(x^*))^{-1}_{ik} = 0$.

To show that the second sum at $x = x^*$equals δ_{ij}, note that it equals to the product of the i-th row of the matrix $(f'(x))^{-1}$ with the j-th column of the matrix f', where $(f'(x))^{-1}f' = I$. ■

4.7 Functional iterations and orders of convergence

Consider iterations of the form $x_{k+1} = F(x_k)$ with $x^* = F(x^*)$ and $F'(x^*) \neq 0$, $|F'(x^*)| < 1$. From the mean value theorem,

$$x_{k+1} - x^* = F(x_k) - F(x^*) = F(\xi_k)(x_k - x^*)$$

which gives

$$\lim_{k\to\infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|} = \lim_{k\to\infty} |F('\xi_k)| = F'(x^*) \neq 0.$$

Because the error converges to zero about as a geometric sequence, the iterates are said *converge geometrically*, or with order 1. Recall that for Newton's method we have derived that the error in step n satisfies $\lim_{n\to\infty} \frac{e_{n+1}}{e_n^2} = C \neq 0$. Here we take another view. Consider the 1D case. Since Newton's method is given by $x_{k+1} = F(x_k)$ with $F(x) = x - (f'(x))^{-1} f(x)$, and $F'(x^*) = 0$, we have from Taylor's theorem

$$x_k - x^* = F(x_k) - F(x^*) = F(x^*) + F'(x^*)(x_k - x^*) + \frac{1}{2!}F''(\xi_k)(x_k - x^*)^2$$

giving again

$$\lim_{t\to\infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|^2} = \frac{1}{2}F''(x^*).$$

This is called convergence of order 2, or quadratic. In general, we have

Definition 20 *Sequence $a_n \to 0$ has convergence of order α if*

$$\lim_{k\to\infty} \frac{|a_{k+1}|}{|a_n|^\alpha} = C \in (0, +\infty)$$

Application of the Taylor theorem shows that functional iterations have always integer order of convergence, equal of the order of the first nonzero derivative at the fixed point:

Theorem 21 *Suppose $x_{n+1} = F(x_n) \to x^* = F(x^*)$, $F^{(m)}$ is continuous derivatives at x^*, and $F(x^*) = F'(x^*) = \cdots = F^{(m-1)}(x^*) = 0$. Then $x_n \to x^*$ with order m.*

For $m > 1$, if x_0 is sufficiently close to x^*, the iterations are in fact guaranteed to converge to x^* regardless of the value of $F^{(m)}(x^*)$, while for $m = 1$, it is required that $|F'(x^*)| < 1$.

Exercise 22 *Extend the results of this section to multiple dimensions, using a norm in place of absolute value.*

If the iterates are obtained by another method than functional iterations, the convergence order does not have to be an integer, as shown in the next section.

4.8 Secant method

The secant method is obtained by approximating the derivative in Newton's method by a divided difference:

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

It can be shown that for $e_n = x_n - x^*$, it holds that $e_{n+1} \approx C e_n e_{n-1}$. To obtain the order of convergence, assume $|e_{n+1}| \approx A|e_n|^\alpha$. From this, $e_{n-1} \approx A^{\frac{1}{\alpha}}|e_n|^{\frac{1}{\alpha}}$, and substituting in the previous equation yields $A|e_n|^\alpha \approx C|e_n|A^{\frac{1}{\alpha}}|e_n|^{\frac{1}{\alpha}}$. Comparing the powers of $|e_n|$ gives $\alpha = 1 + \frac{1}{\alpha}$, hence $\alpha = \frac{1+\sqrt{5}}{2}$.

The advantage of the secant method is that it does not require the evaluation of the derivative of f; only one evaluation of the value of f per iteration is required. However, the secant method cannot be easily generalized to multiple dimension like the Newton's method.

Chapter 5

Polynomials and Horner's rule

Review of polynomials

See Section 14.3.

5.1 Horner's rule as recursive evaluation of polynomial

Consider numerical evaluation of the a polynomial

$$p(z) = a_0 + az + \ldots \cdots + a_n z^n$$

The straightforward approach directly from the definition is to compute the powers z^k for all k, form the terms $a_k z^k$, and sum. This takes $O(n^2)$ operations.

A better approach is to form the powers z^k recursively and build the sum at the same time, which takes about $2n$ operations:

$p = 1$
$s = a_0$
$for\ k = 1 : n$
$p = p * z$
$s = s + a_k p$
end now $p(s) = s$

But a few quick experiments will convince you that the terms arizing in the sum (values of the variable s above) can be quite large, even if the result is not, and the numbers z^k can be out of scale. From what we have learned in the chapter about floating point arithmetic, this does not bode well for numerical accuracy.

A still better approach is to use the recursion

$$\begin{aligned} p(z) &= a_0 + z(a_1 + a_2 z + \ldots + a_n z^{n-1}) \\ &= a_0 + z(a_1 + z(a_2 + a_3 z + \ldots + a_n z^{n-2})) \\ &= a_0 + z(a_1 + z(a_2 + z(a_3 \ldots + z a_n)\ldots)) \end{aligned} \tag{5.1}$$

which is known as *Horner's scheme.* We write the recursion as follows:

to evaluate $p(z_0)$
$p(z) = a_0 + zb_0, \qquad b_0 = a_1 + a_2 z + \ldots + a_n z^{n-1}$
$b_0 = a_1 + zb_1, \qquad b_1 = a_2 + a_3 z + \ldots + a_n z^{n-2}$
$\vdots$
$b_{n-2} = a_{n-1} + zb_{n-1}, \qquad b_{n-1} = a_n$
This can be rewritten as the *Horner's algorithm:*

$b_n = 0$
for $k = n : -1 : 0$
$b_{k-1} = a_k + zb_k$
end
now $p(z) = b_{-1}$

Of course, if all we care about is the value $p(z)$, we can use a single scalar variable b. However we will see that the numbers b_k have a useful interpretation.

A few quick experiments will convince you that the numbers arizing the Horner's method are much smaller than in the naive evaluation, especially if z is close to a root, so Horner's scheme can be expected to be much more stable.

5.2 Horner's scheme as division of polynomials

Let z_0 be fixed and consider the division of polynomials in variable z,

$$q(z) = \frac{p(z) - p(z_0)}{z - z_0} \Leftrightarrow p(z) = (z - z_0)q(z) + p(z_0)$$

Denote $q(z) = b_0 + b_1 z + \ldots + b_{n-1}z^{n-1}$ and compare coefficient of the same z^k in $p(z)$ and $(z - z_0)q(z) + p(z_0)$:

$$\begin{aligned} p(z) &= a_0 + a_1 z + \ldots + a_n z^n \\ &= z(b_0 + b_1 z + \ldots + b_{n-1}z^{n-1}) - z_0(b_0 + \ldots + b_{n-1}z^{n-1}) + p(z_0) \\ &= \underbrace{p(z_0) + z_0 b_0}_{a_0} + \underbrace{(b_0 - z_0 b_1)z}_{a_1} + \ldots + \underbrace{(b_{n-2} - z_0 b_{n-1})z^{n-1}}_{a_{n-1}} + \underbrace{(b_{n-1})}_{a_n} z^n \end{aligned}$$

That is,
$b_{n-1} = a_n$
$b_{n-2} = a_{n-1} + z_0 b_{n-1}$
$b_0 = a_1 + z_0 b_1$
$b_{-1} = p(z_0) = a_0 + z_0 b_0$

which are the same numbers as come up in Horner's scheme. Hence, we have

Theorem 23 *The coefficients $b_{-1}, b_0, \ldots, b_{n_1}$ from Honer's scheme satisfy $b_{-1} = p(z_0)$, and $\frac{p(z)-p(z_0)}{z-z_0} = b_0 + b_1 z + \ldots + b_{n-1}z^{n-1}$.*

5.3 Horner's scheme for evaluation of derivatives

Given a polynomial $p(z)$ and number z_0, we have Taylor polynomial

$$p(z) = p(z_0) + p'(z_0)(z - z_0) + \frac{p''(z_0)}{2!}(z - z_0)^2 + \ldots + \frac{p^{(n)(z_0)}}{n!}(z - z_0)^n$$

where the remainder $R_n = \frac{p^{(n+1)}(\xi)}{n!}(z - z_0)^{n+1} \equiv 0$ because $p^{(n+1)} = 0$. Dividing by $(z - z_0)$, we get

$$\begin{aligned} \frac{p(z) - p(z_0)}{z - z_0} &= p'(z_0) + \frac{p''(z_0)}{2!}(z - z_0) + \ldots + \frac{p^{(n)}}{(z_0)}(z - z_0)^{n-1} \\ &= b_0 + b_1 z + \ldots + b_{n-1}z^{n-1} \end{aligned}$$

Taking the limit $z \to z_0$, we get

$$p'(z_0) = b_0 + b_1 z_0 + \ldots + b_{n-1} z_0^{n-1},$$

which can be evaluated again by Horner's scheme. Higher order derivatives are computed by applying this proces several times.

5.4 Finding roots of polynomials

The general procedure for finding roots of polynomial p of degree n is

1. *root localization*: find a reasonably good approximation of a root
2. *iterative solution:* using the approximation as a starting point, use a method such as Newton's method to find a root, say z_n
3. *deflation:* if $p(z_0) = 0$, then $p(z) = q(z)(z - z_n)$, where q is of degree $n - 1$, and apply the same procedure recursively for the deflated polynomial q

Eventually we obtain representation of the polynomial as product of root factors $p(z) = c(z - z_0) \ldots (z - z_{n-1})$.

For numerical stability, it is important to improve the roots of deflated polynomials by repeating the Newton's method on the original polynomial p before using the root in further deflation. Horner's scheme is useful in the iterative solution to evaluate a polynomial and its derivative or derivatives, and to divide the polynomial by the root factor in the deflation step. However, finding roots of a polynomial by a process involving evaluation of the polynomial suffers from the fact that it is inherently difficult to evaluate a polynomial accurately. With fast computers and more memory available, the indirect method described next is preferred.

Overview of determinants

See Section**??**.

5.5 Horner's rule and companion matrix

Consider polynomial $P(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} + x^n$. (Such polynomial with $a_n = 1$ is said to be in *canonical form.*)

Definition 24 *Companion matrix of* $p(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} + x^n$ *is*

$$A = \begin{pmatrix} 0 & 1 & & & \\ & 0 & \ddots & & \\ & & \ddots & 1 & \\ & & & 0 & 1 \\ -a_0 & -a_1 & \cdots & -a_{n-2} & -a_{n-1} \end{pmatrix}$$

Example 25 *The companion matrix of* $p(x) = 1 + 2x + 3x^2 + x^3$ *is*

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

.

Theorem 26 *Let A be companion matrix of $p(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} + x^n$. Then* $\det(\lambda I - A) = p(\lambda)$.

Proof. Expanding the characteristic polynomial by the first column, we have

$$\begin{aligned}
\det(\lambda I - A) &= \det \begin{pmatrix} \lambda & -1 & & & \\ & \lambda & \ddots & & \\ & & \ddots & -1 & \\ & & & \lambda & -1 \\ a_0 & a_1 & \cdots & a_{n-2} & \lambda + a_{n-1} \end{pmatrix} \\
&= \lambda \det \begin{pmatrix} \lambda & -1 & & & \\ & \lambda & \ddots & & \\ & & \ddots & -1 & \\ & & & \lambda & -1 \\ a_1 & a_2 & \cdots & a_{n-2} & \lambda + a_{n-1} \end{pmatrix} \\
&\quad +(-1)^{n+1} a_0 \det \begin{pmatrix} \lambda & -1 & & & \\ & \lambda & \ddots & & \\ & & \ddots & -1 & \\ & & & \lambda & -1 \end{pmatrix},
\end{aligned}$$

that is,

$$\det(\lambda I - A) = \lambda \det(\lambda I - A_1) + a_0,$$

where A_1 is the companion matrix of the polynomial $a_1 + a_2 x + \cdots + a_{n-1}x^{n-2} + x^{n-1}$. Continuing the same way, we get

$$\begin{aligned}
\det(\lambda I - A) &= a_0 + \lambda \left(a_1 + a_2 \lambda + \cdots + a_{n-1}\lambda^{n-2} + \lambda^{n-1} \right) \\
&= a_0 + \lambda \left(a_1 + \lambda \left(a_2 + \cdots + a_{n-1}\lambda^{n-3} + \lambda^{n-2} \right) \right) \\
&\quad \vdots \\
&= a_0 + \lambda \left(a_1 + \lambda \left(a_2 + \cdots + \lambda (a_{n-1} + \lambda) \ldots \right) \right)
\end{aligned}$$

which is the Horner's scheme for computing $p(\lambda)$. ■

Review of eigenvalues

See Section 16.4

5.6 Computing Roots of Polynomials as Eigenvalues

Theorem 27 *If x is a root of polynomial $p(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} + x^n$, then*

$$|\lambda| \leq \max \left\{ 1, \sum_{i=0}^{n-1} |a_i| \right\}$$

and

$$|\lambda| \leq \max \left\{ |a_0|, 1 + |a_1|, \ldots, 1 + |a_n| \right\}.$$

Proof. Let A be the companion matrix of p. Then $p(x) = 0$ implies

$$|x| \leq \|A\|_\infty = \max | \{a_0|, 1 + |a_1|, \ldots, 1 + |a_n|\}$$

and

$$|x| \leq \left\|A^T\right\|_\infty = \max\left\{1, \sum_{i=0}^{n-1} |a_i|\right\}$$

■

Numerically stable algorithms for computing eigenvalues and eigenvectors are known. So, it makes sense to replace computation of roots by computation of eigenvalues of the companion matrix [EM95]. This is how Matlab does it, for example. Type in Matlab `type root` to see the Matlab source.

Chapter 6

Direct Solution of Linear Systems

6.1 Triangular matrices

Definition 28 *Matrix $U = (u_{ij})$ is called* upper triangular *if $u_{ij} = 0$ for all $i > j$. Matrix $L = (l_{ij})$ is called* lower triangular *if $l_{ij} = 0$ for all $i < j$.*

Square upper and lower triangular matrices look like this:

$$U = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ \vdots & & & \vdots \\ 0 & & & \\ 0 & 0 & \cdots & u_{nn} \end{pmatrix}, \qquad L = \begin{pmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & & & \vdots \\ \vdots & & & \\ l_{n1} & 0 & \cdots & l_{nn} \end{pmatrix}.$$

In this chapter, we deal only with square triangular matrices. Solving a system with a triangular matrix is particularly simple. For upper triangular matrix, $Ux = y$ gives

$$\begin{aligned} u_{11}x_1 + \cdots\cdots + u_{1n}x_n &= y_1 \Rightarrow \qquad x_1 = \frac{1}{u_{11}}(y_1 - (u_{12}y_2 + \cdots + u_{1n}y_n) \\ &\vdots \\ u_{n-1,n-1}x_{n-1} + u_{n-1,n}x_n = y_{n-1} &\Rightarrow \qquad x_{n-1} = \frac{1}{u_{mn}}(y_{n-1} - u_{n-1,n}y_n) \\ u_{nn}x_n = y_n &\Rightarrow \qquad x_n = \frac{y_n}{u_{mn}} \end{aligned}$$

In general,

$$\begin{aligned} \sum_{j=1}^{n} u_{ij}x_j = y_i \quad &\Longleftrightarrow \quad u_{ii}x_i + \sum_{j=i+1}^{n} u_{ij}x_j = y_i \\ &\Longleftrightarrow \quad x_i = \frac{1}{u_{ii}}\left(y_i - \sum_{j=i+1}^{n} u_{ij}x_j\right), \qquad i = n, n-1, \ldots, 1 \end{aligned}$$

Algorithm 29 (Solution of upper triangular system)
for i=n:-1:1
$x_i = \frac{1}{u_{ii}}\left(y_i - \sum_{j=i+1}^{n} u_{ij}x_j\right)$
end

6.2 LU by Gaussian elimination

Algorithm 30 (Gauss elimination) *for i=1:n-1*
subtract row i from row $\imath + 1 \dots n$ *to force 0 under diagonal term i*
end

Gauss elimination can be interpreted as multiplication by transformation matrices. Let A be $n \times n$.

$$A = A_0 = \begin{pmatrix} a_{11} & c^T \\ b & A_{11} \end{pmatrix}$$

One step of Gauss elimination is the same as multiplication of A by the transformation matrix

$$E_1 = \begin{pmatrix} 1 & 0 \\ -\frac{b}{a_{11}} & A_I \end{pmatrix}$$

which gives

$$A_1 = E_1 A_0 = \begin{pmatrix} a_{11} & c \\ 0 & A_{11} - \frac{bc^T}{a_{11}} \end{pmatrix}$$

Note that b is size $n-1 \times 1$, c is size $1 \times n-1$, so $\frac{bc^T}{a_{11}}$size $n-1 \times n-1$, which is a matrix of rank one or zero. Proceeding this way further, we express Gauss elimination as multiplication by transformation matrices:

Let $A_0 = A$. For $i = 1, 2, \dots, n-1$, when we already have

$$A_i = E_{i-1} \cdots E_1 A = \begin{pmatrix} U_i & \cdot & \cdot \\ 0 & a_{ii} & c_i^T \\ 0 & b_i & A_{ii} \end{pmatrix}$$

define the transformation matrix

$$E_i = \begin{pmatrix} I & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{b_i c_i^T}{a_{ii}} & I \end{pmatrix}$$

and express the operation of adding to all row $i+1, \dots n$ a multiple of row i as matrix-matrix multiplication,

$$A_i = E_i A_{i-1} = \begin{pmatrix} U_i & \cdot & \cdot \\ 0 & a_{ii} & c_i^T \\ 0 & 0 & A_{ii} - \frac{b_i c_i^T}{a_{ii}} \end{pmatrix}$$

In the end, we get

$$\underbrace{E_{n-1} \cdots E_1}_{L^{-1}} A = U$$

where U is upper triangular. This allows solution of the linear system $Ax = b$ as follows:

$$y_0 = b, \qquad y_i = E_i y_{i-1}, i = 1, \dots, n-1, \qquad Ux = y_{n-1},$$

that is, multiplication by the matrix L^{-1} in factored form, followed by the solution of aan upper triangular system.

We will show that

$$L = E_1^{-1} \cdot \cdots \cdot E_{n-1}^{-1}$$

is lower triangular and give an explicit formula for L. First,

$$E^{-1} = \begin{pmatrix} I & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{b_i}{a_{ii}} & I \end{pmatrix}$$

because

$$\begin{pmatrix} I & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{b_i}{a_{ii}} & I \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{b_i}{a_{ii}} & I \end{pmatrix} = \begin{pmatrix} I & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & I \end{pmatrix}$$

This shows that L is lower triangular, because product of lower triangular matrices is lower triangular.

In fact the matrix L has the columns $\frac{b_i}{a_{ii}}$ under the diagonal. This follows easily by induction. Suppose that

$$E_{i+1}^{-1} \dots E_{n-1}^{-1} = \begin{pmatrix} I & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & L_{ii} \end{pmatrix}$$

and compute

$$E_i^{-1} E_{i+1}^{-1} \dots E_{n-1}^{-1} = \begin{pmatrix} I & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{b_i}{a_{ii}} & I \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & L_{ii} \end{pmatrix} =: \begin{pmatrix} I & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{b_i}{a_{ii}} & L_{ii} \end{pmatrix}.$$

In conclusion, if Gauss elimination can be performed (that it, division by zero does not occur), we obtain the decomposition

$$A = LU,$$

with L lower and U upper triangular, and the diagonal entries of L equal to one.

6.3 Outer product LU decomposition

While it is possible to compute LU decomposition from Gauss elimination, we can compute LU decomposition directly. Write again the matrix A in the block form

$$A = \begin{pmatrix} a_{11} & c^T \\ b & A_1 \end{pmatrix}$$

and suppose we can decompose matrices of order $n-1$. To get a decomposition of A, write the assumed form

$$\begin{pmatrix} a_{11} & c^T \\ b & A_1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 \\ l_1 & L_1 \end{pmatrix}}_{L} \underbrace{\begin{pmatrix} a_{11} & u_1^T \\ 0 & U_1 \end{pmatrix}}_{U}$$

and compare terms:

$$\begin{aligned} l_1 a_{11} = b \quad &\Rightarrow \quad l_1 = \frac{1}{a_{11}} b \\ u_1^T = c^T & \\ l_1 u_1^T + L_1 U_1 = A_{11} \quad &\Rightarrow \quad L_1 U_1 = A_{11} - l_1 u_1^T \end{aligned}$$

To compute LU decomposition of A, we use the following algorithm:

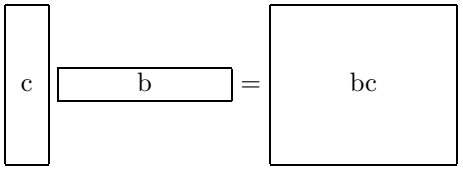
1. compute l_1, u_1^T, and $A_{11} - l_1 u_1^T$ as above

2. use the same algorithm to compute LU decomposition of the matrix $A_{11} - l_1 u_1^T$ of order one less

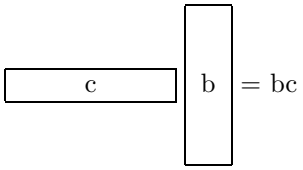
3. eventually, A_{11} will be a scalar, at which point we take $L_{11} = 1$, $U_{11} = A_{11}$.

Because the dominant operation is the rank one update $A_{11} - l_1 u_1^T$, where $l_1 u_1^T$ is the so-called outer product, this method is called the outer product algorithm.

outer product



inner product



6.4 BLAS numerical kernels

The dominant operation in outer LU decomposition is the rank one update. This is an example of a numerical kernel. Numerical kernels are frequently used operations that dominate linear algebra. Common numerical kernels are collected in the Basic Linear Algebra Subroutines (BLAS) library. BLAS are available as a free reference source code, optimized libraries by computer vendors and researchers, and recently as an adaptive optimized library ATLAS, which probes the computer during its installation and selects the best algorithms for the given hardware, compiler, and matrix size. The public codes and documentation can be obtained from `www.netlib.org`.

BLAS routines exist for various types of matrices (general, symmetric, hermitean,...). The first letter denotes the data type; we use D for double precision here, because this is the most common one.

Level 1 BLAS are vector operations and perform $O(n)$ operations $O(n)$ inputs. For example,

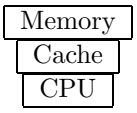
DAXPY	a times x plus y	$y = ax + b$
DDOT	dot product	$s = x^T y$
DSCAL	multiplication by scalar	$x = ax \quad i = 1, \ldots, n$
DCOPY	copy $y = x$	$y = x, \quad i = 1, \ldots, n$

Level 1 were the first BLAS developed. When vector computers came, such as CRAY, it became apparent that several vector operations should be done in the same loop to allow for more optimization and better use of the hardware. Thus Level 2 BLAS were developed, which are are matrix-vector operations that perform $O(n^2)$ operations on $O(n^2)$ inputs, for example

DGEMV	matrix times vector	$y = \alpha Ax + \beta y$
DGER	rank one update	$A = \alpha x y^T + A$
DTRSV	solve a triangular system	x computed from $Lx = b$ or $Ux = b$

As the increase of CPU speed has vastly overcame the memory speed, the use of hierarchical memory became more important. This includes today's PCs. In the simplest case, such computers

have a small but very fast cache memory and large but much slower main memory.



Data are moved between the cache and the memory by the computer by a combination of system hardware and software, completely transparent to the running code. Typically, the cache contains data that were most recently used, and a request for data that is not in the cache can incur penalty of hundreds or thousands of CPU cycles. If program operaters on matrices that fit completely in cache, the calculations are very fast, but getting the data in and out of the cache is slow. For this reason, matrix computations are structured around Level 3 BLAS, which implement operations involving $O(n^3)$ operations on $O(n^2)$ data, such as matrix-matrix multiplication.

DGEMM	matrix-matrix multiply	$C = \alpha AB + \beta C$
DSYRK	symmetric rank k update	$C = \alpha AA^T + \beta C$
DTSRM	solve a triangular system	$LX = \alpha B$ or $UX = \alpha B$

For matrices that do not fit in the cache, Level 3 BLAS split the matrices in blocks that do fit in the cache. In fact, optimized BLAS implementations use a multilevel block splitting to take advantage of the complete memory hierarchy, from CPU registers to higher level caches. The core of Level 3 BLAS is the matrix-matrix multiplication. There are even implementations of Level 3 BLAS that implement all other functions by calling a highly optimized DGEMM.

6.5 Block LU decomposition

We now present a version of LU decomposition that spends all $O(n^3)$ work in Level 3 operations. We look for the decomposition in the form

$$\begin{aligned} A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} &= \begin{bmatrix} L_{11} & 0 \\ L_{21} & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & \tilde{A} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & I \end{bmatrix} \begin{matrix} \} \\ \} \end{matrix} \begin{matrix} r \\ n-r \end{matrix} \\ &= \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix} \end{aligned}$$

Comparing the terms, we get

$L_{11}U_{11} = A_{11}$	LU decomposition
$L_{21}U_{11} = A_{21}$	back substitution with multiple right-hand sides
$L_{11}U_{12} = A_{12}$	forward substitution with multiple right-hand sides
$\tilde{A} = A_{22} - L_{21}U_{12}$	rank r update
$L_{22}U_{22} = \tilde{A}$	LU decomposition

If either of the blocks A_{11} of A_{22} is too large, the decomposition is used recursively. A common choce is $r << n$ small enough that matrices size $r \times r$ fit in the cache, decompose A_{11} directly, and apply the method recursively to $\tilde{A}$.

6.6 Existence of LU decomposition

Choosing $r = n - 1$, write the decomposition above as

$$A = \begin{bmatrix} A_{n-1} & b_n \\ c_n^T & a_{nn} \end{bmatrix} = \begin{bmatrix} L_{n-1} & 0 \\ l_n^T & l_{nn} \end{bmatrix} \begin{bmatrix} U_{n-1} & u_n \\ 0 & u_{nn} \end{bmatrix}$$

where

$$A_{n-1} = L_{n-1}U_{n-1}, \quad L_{n-1}u_n = b_n, \quad l_n^T U_{n-1} = c_n, \quad l_{nn}u_{nn} = a_{nn} - l_n^T u_n$$

Lemma 31 *The matrix A has LU decomposition with nonzero diagonal of L and U if and only if the block A_{11} has LU decomposition with nonzero diagonal, and $a_{22} - l_1^T u_1 \neq 0$.*

Theorem 32 *Matrix $A \in \mathbb{C}^{n\times n}$has LU decomposition with nonzero diagonal of L and U if and only if all principal minors of A are nonzero.*

Proof. The proof is by induction. For $n = 1$, the theorem is obviously true. Assume that the Theorem is true for n replaced by $n-1$, and prove that it is true for n.If A has LU decomposition with nonzero diagonal, then A_{n-1} does and $\det A \neq 0$. Because all principal minors of A are det A and the principal minors of A_{n-1}, it holds that all principal minors of A are nonzero. By induction assumption, A_{n-1} has decomposition $A_{n-1} = L_{n-1}U_{n-1}$ with nonzero diagonals. Since $\det A_{n-1} \neq 0$, it holds that A_{n-1} is nonsingular, and we can write

$$\begin{bmatrix} I & 0 \\ -c_n^T A_{n-1}^{-1} & I \end{bmatrix} \begin{bmatrix} A_{n-1} & b_n \\ c_n^T & a_{nn} \end{bmatrix} = \begin{bmatrix} A_{n-1} & \cdot \\ 0 & a_{nn} - c_n^T A_{n-1}^{-1} b_n \end{bmatrix},$$

where

$$a_{nn} - c_n^T A_{n-1}^{-1} b_n = a_{nn} - l_n^T U_{n-1} A_{n-1}^{-1} L_{n-1} u_n = a_{nn} - l_n^T u_n$$

which gives

$$detA = detA_{11}(a_{nn} - l_n^T u_n)$$

Since $\det A \neq 0$ and $\det A_{n-1} \neq 0$, we have $a_{nn} - l_n^T u_n \neq 0$, and there exist l_{nn}, $u_{nn} \neq 0$ such that $l_{nn}u_{nn} = a_{nn} - l_n^T u_n$. ■

LU decomposition can be used to compute the determinant of a matrix, since

$$A = LU \Rightarrow \det A = \det L \det U = \prod_i l_{ii} \prod_i u_{ii}$$

See the source of the Matlab function `det`. Different choices in the selection of l_{nn}, u_{nn} lead to different algorithms

$$\begin{array}{cll} & l_{nn} = 1,\ u_{nn} = a_{nn} - l_n^T u_n & \text{Doolittle decomposition} \\ : & l_{nn} = a_{nn} - l_n^T u_n, u_{nn} = 1 & \text{Crout decomposition} \\ & u_{nn} = l_{nn} = \sqrt{a_{nn} - l_n^T u_n} & \text{Cholesky decomposition} \end{array}$$

Overview of minors of positive definite matrices

See Sec. 16.6.

6.7 Cholesky decomposition

Choleski decomposition is usually applied to a symmetric matrix. Then, by induction, $c_n = b_n$ and $L_n = U_n^T$. Because a symmetric matrix is positive definite if and only if all its principal minors are positive, we have

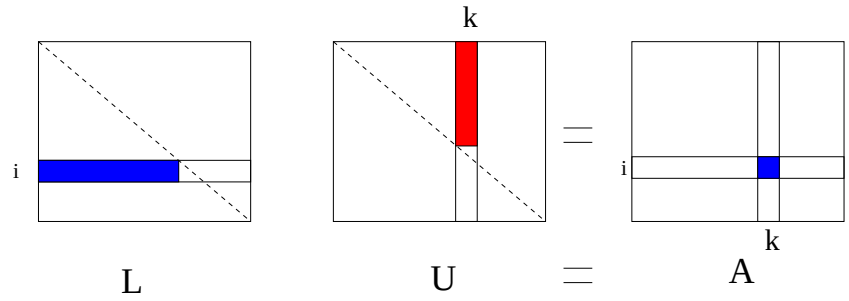
Theorem 33 *A matrix $A \in \mathbb{R}^{n\times n}$, $A = A^T$, has Choleski decomposition $A = LL^T$, with $L \in \mathbb{R}^{n\times n}$ lower triangular with nonzero diagonal, if and only if A is positive definite.*

In practice, computing Choleski decomposition is the best way to test if a symmetric matrix is positive definite. If the matrix is not positive definite, there will be at some point $a_{ii} - l_i^T u_i \leq 0$.

To find the smallest eigenvalue of $A = A^T$ is equivalent to finding the smallest t such that $A - tI$ is positive definite, a good (and stable) but expensive method is to employ an interval halving strategy.
t_1 $A - t_1 I$ is p. d.
t_2 $A - t_2 I$ is not p. d.
$\Rightarrow$ the smallest eigenvalue of A is in $(t_1, t_2]$

6.8 Inner product LU

Assume $A = LU$, where L is lower and U upper triangular and derive formulas for the entries of L and U.



Clearly, $\sum_{j=1}^{\min\{i,k\}} l_{ij}u_{jk} = a_{ik}$. For $i = k$, this becomes $a_{ii} = \sum_{j=1}^{i} l_{ij}u_{ji} = a_{ii}$, hence

$$a_{ii} = l_{ii}u_{ii} + \sum_{j=1}^{i-1} l_{ij}u_{ji}.$$

Variants:

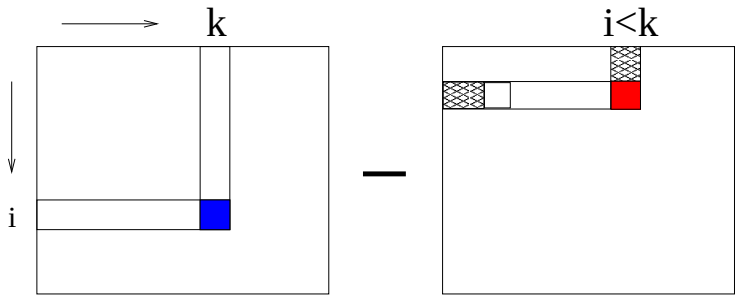
$l_{ii} = 1 \Rightarrow$ Doolittle's LU

$u_{ii} = 1 \Rightarrow$ Crout's LU,

$l_{ii} = u_{ii} \Rightarrow$ Cholesky's LU.

Derive formulas for Doolittle's LU:

$u_{ii} = 1$, $l_{ii} = a_{ii} - \sum_{j=1}^{i-1} l_{ij}u_{ji}$

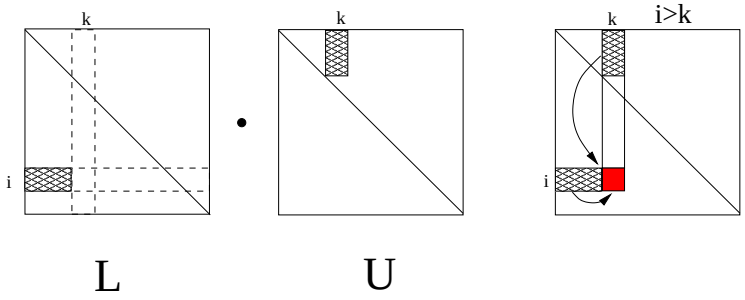


$i < k \implies a_{ik} = \sum_{j=1}^{i} l_{ij}u_{jk} \Rightarrow a_{ik} = l_{ii}u_{ik} + \sum_{j=1}^{i-1} l_{ij}u_{jk}$ so

$$u_{ik} = \frac{1}{l_{ii}}\left(a_{ik} - \sum_{j=1}^{i-1} l_{ij}u_{jk}\right)$$

$i > k \implies a_{ik} = \sum_{j=1}^{k} l_{ij}u_{jk} \Rightarrow a_{ik} = l_{ik}u_{kk} + \sum_{j=1}^{k-1} l_{ij}u_{jk}$ gives

$$l_{ik} = \frac{1}{u_{kk}}\left(a_{ik} - \sum_{j=1}^{k-1} l_{ij}u_{jk}\right)$$



Definition 34 *If A is $n \times n$ matrix and $a_{ik} = 0$ for $|i-k| > b$, then A that is called a band matrix and b is called its half-bandwidth.*

Theorem 35 *If $A = LU$, A is band matrix with half-bandwidth b, then L and U are also.*

6.9 Inner product Cholesky

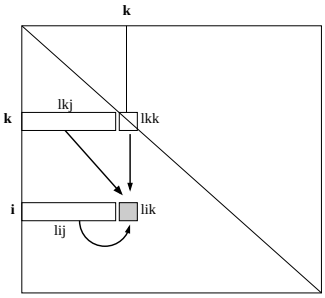
Let A s.p.d. We know $A = LL^T$ (exists Cholesky decomposition) Find L efficiently? Develop algorithm

$$A = (a_{ik}) \quad L = (l_{ij}) \quad l_{ij} = 0 \quad \text{if} j > i$$

$$LL^T = A \Rightarrow \sum_{j=1}^{n} l_{ij} l_{kj} = a_{ik} = \sum_{j=1}^{\min(i,k)} l_{ij} l_{kj} = a_{ik}$$

$$i = k \quad \Rightarrow \quad a_{ii} = l_{ii}^2 + \sum_{j=1}^{i-1} l_{ij}^2 \Rightarrow l_{ii} = \sqrt{a_{ii} - \sum_{j=1}^{i-1} l_{ij}^2}$$

$$i > k \quad \Rightarrow \quad a_{ik} = l_{ik} l_{kk} + \sum_{j=1}^{k-1} l_{ij} l_{kj} \Rightarrow l_{ik} = \frac{1}{l_{kk}} (a_{ik} - \sum_{j=1}^{k-1} l_{ij} l_{kj})$$



Algorithm 36 (Cholesky decomposition)
for k=1:n
$l_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} l_{jk}^2}$
for i=k+1:n
$l_{ik} = \frac{1}{l_{kk}} (a_{ik} - \sum_{j=1}^{k-1} l_{ji} l_{jk})$
end
end

Here is the algorithm in a form suitable for coding.

Algorithm 37 (Choleski algorithm)
input: matrix A only lower triangular part used
output: lower triangular of A replaced by entries of L
for k=1:n
for i=k:n
t = a(k,k)
for j=1:k-1
*t = t-a(j,i)*a(j,k)*
end
if i==k
if t ≤ 0, error, end
a(k,k)=sqrt(t)
else
a(i,k)=t/a(k,k)
end
end
end

6.10 Operations counts

For Cholesky:

$$\begin{aligned} \text{Operations} &\sim \sum_{k=1}^{n}\sum_{i=1}^{n-k-1} k-1 = \sum_{k=1}^{n}(n-k+1)(k-1) = \sum_{k=1}^{n} n(k-1)-(k-1)^2 \\ &= n\frac{n^2}{2} - \frac{n^3}{3} + \sigma(n^2) = \frac{n^3}{6} + O(n^2) \end{aligned}$$

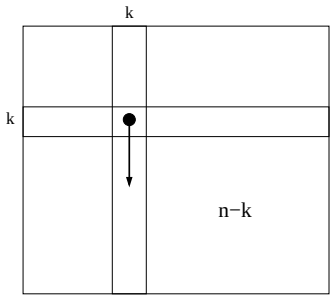
because

$$\sum_{k=1}^{n} k^m = \frac{n^{m+1}}{m+1} + O(n^m), n \to \infty$$

To illustrate this formula:

$$1 + \cdots + n = \frac{n(n+1)}{2} = \frac{n^2}{2} + \cdots$$

For Gauss elimination:



$$\begin{aligned} \text{Operations} &\sim \sum_{k=1}^{n}(n-k)^2 = \frac{(n-1)^3}{3} + O(n^2) \\ &= \frac{n^3}{3} + O(n^2) = 0^2 + 1^2 + \cdots + (n-1)^2 \end{aligned}$$

So Cholesky decomposition saves about half of the operations by taking advantage of the symmetry of the matrix.

6.11 Root-free Cholesky

To avoid square roots and generalize Cholesky decomposition for indefinite matrices,

$$A = LDL^T \quad a_{ik} = \sum_{j=1}^{k} l_{ij} d_{jj} l_{jk} \quad l_{ii} = 1$$

$$D = \begin{bmatrix} d_{11} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & d_n \end{bmatrix}$$

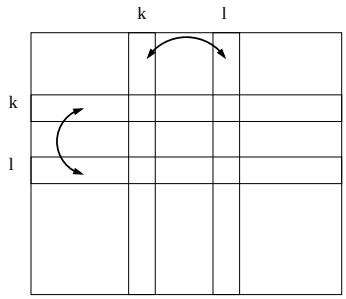
If all leading minors are nonzero we can use LU decomposition to get $A = L\tilde{U} = LD(\overbrace{D^{-1}\tilde{U}}^{U})$

$$A = A^T \Rightarrow^? A = LDL^T \quad A = LDU, A^T = U^T D L^T \Rightarrow^? L = U^T$$

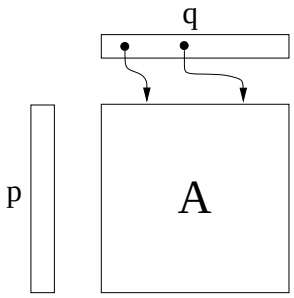
$$\begin{aligned} a_{ik} = \sum_{j=1}^{k} &= l_{ij} d_{jj} l_{jk}, \quad l_{ii} = 1 \\ \Rightarrow d_{kk} &= a_{kk} - \sum_{j=1}^{k-1} d_{jj} l_{ij} u_{jk} \\ \Rightarrow u_{jk} &= \cdots \\ & \quad l_{ij} = \cdots \end{aligned}$$

Algorithm 38 (LDLT decomposition, root-free Cholesky)
for $k = 1 : n$
for $j = 1 : k-1$,
$v_{j_k} = d_{jj} l_{kj}$
end
for $i = k : n$
$t = a_{ik} - \sum_{j=1}^{k-1} l_{ij} \underbrace{d_{jj} l_{kj}}_{V_j}$
if $k == i$
$d_{kk} = t \cdots$
else
$l_{ik} = \frac{1}{d_{kk}}(a_{ik} - t) \cdots$
end
end
end

6.12 Gauss elimination with pivoting



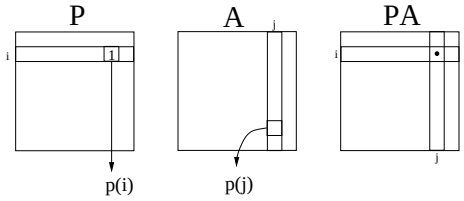
Row pivoting: exchange rows k and l, $l > k$
Scaled row pivoting: choose l so that $\frac{a_{lk}^{(k)}}{\sum_{k=1}^{n} |a_{lk}|} \to \max$
Column pivoting:
Full pivoting: choose $l, \tilde{l}$ $|a_{l\tilde{l}}| \to \max$ $l, \tilde{l} \geq k$
1. Initialize $p = [1:n], q = [1:n]$
2. In the rest, $a(p(i), q(i))$ stores $a_{ij}^{(k)}$ after swapping
exchange row $i, l : p(i) \leftrightarrow p(l)$
for$k = 1 : n$
search for $|a(p(i), q(j))| \to \max = a(p(i_0), q(j_0))$
$p(k) \leftrightarrow p(i_0)$ $q(k) \leftrightarrow q(j_0))$
for $i = k : n$,for $j = k + 1 : n$
$a(i, j) = a(i, j) - \frac{a(j,k) \times a(k,j)}{a(k,k)}$
end, end
end
$a(p(i), q(j))$



after LU or Gauss or Cholesky we get
$B = (b_{ij})$ $b_{ij} = a(p(i), q(j))$
$B = LU$

This is usually written as $B = PAQ^T$, where P and Q are permutation matrices: define P by $P_{ij} = 1 \iff p(i) = j$

Permutation matrix P so that $(a_{(p(i),j})_{ij} = PA$



Definition 39 *A vector p is a permutation vector if the map $i \to p(i)$, $\{1, \ldots, n\} \to \{1, \ldots, n\}$) is one-to-one(that is, $\forall j \in \{1, \ldots, n\} \exists!$ $i : p(i) = j$. For a permutation vector p, define the corresponding permutation matrix $P = (P_{ij})$ by $P_{i,p(i)} = 1$, other entries are zero.*

Remark 40 *A square matrix is a permutation matrix iff every column and every row have exactly one one and the rest of its entries are zero.*

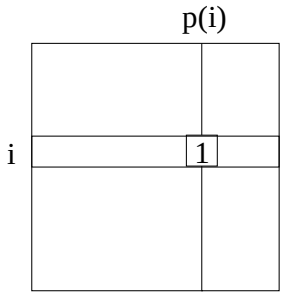
Example 41 $p = (1, 2, 2,)$ *there is no* $p(i) = 3$
and there are 2 *values of with* $p(i) = 2$
not a permutation
$p = (4, 2, 1, 3)$*: Permutation matrix*

Definition 42 *$P \in \mathbb{R}^{n\times n}$ is permutation matrix if $p_{ij} = 0$ or 1 & every row has exactly one 1 & every column has exactly one 1*

Lemma 43 *If $P \in \mathbb{R}^{n\times n}$, $p_{ij} = 0$ or 1, then P is permutation matrix $\Leftrightarrow PP^T = I$*

Proof. every row has exactly one 1 means every row is $\boxed{0 \quad \dots \quad 1 \quad 0 \quad \dots}$ every column has exactly one 1

$$\boxed{\begin{matrix}0\\ \vdots\\ 1\\ 0\\ \vdots\end{matrix}}$$



$(PP^T)_{ij} = P(i,:) * (P^T)(:,j) = P(i,:) * (P(j,:)) = \delta_{ij}$

■

Result of LU with pivoting is
$PA = LU \Rightarrow$ Row pivoting
$AQ^T = LU \Rightarrow$ Column pivoting
$PAQ^T = LU \Rightarrow$ Complete pivoting (both column & row)

To solve $Ax = b$:
get decomposition $PAQ^T = LU$
to solve $PAQ^T y = Pb$: solve $Lz = Pb$
solve $Uy = z$
compute $x = Q^T y$

Theorem 44 *If A is regular, the row pivoting method will succeed*

Proof. If Gauss elimination with row pivoting failed, we have in the reduced matrix the first column of all zeros, then $\det A = 0$. ■

6.13 Accuracy of Gauss Elimination

Theorem 45 *If A is regular, a_{ij} machine numbers, gauss elimination in machine arithmetic delivers $\tilde{L}\tilde{U} = A + E$, where $|e_{ij}| \leq 2n\epsilon \max_{i,j,k} |a_{ij}^{(k)}|$ where $a_{ij}^{(k)}$ are the intermediate values in Gauss elimination.*

$q = \frac{\max_{i,j,k} |a_{ij}^{(k)}|}{\max_{i,j} |a_{ij}|} = q$ is called the growth factor

Theorem 46 *$q \leq 2^{n-1}$ for row pivoting*

Better for full pivoting but still grows fast with n.

Remark 47 *For a random matrix q small on average. In practice q seen small but for large general matrices Gauss can be fishy, especially with very large matrices $n > 10^6$ as often seen today. Even for small matrices, if we need do be sure of the outcome, recommend QR instead. If Gauss elimination with pivoting is used, it is recommended to be followed by iterative refinement (later).*

Gauss elimination can be used without pivoting on
i) s.p.d. matrices
ii) Column diagonally dominant matrices
⇒ reduced matrices inherit their property.
⇒ pivoting not needed.
⇒ bound on growth factor

Definition 48 $\forall i$ *A is row diagonally dominant if* $\mid a_{ii} \mid > \sum_{j,j\neq i} \mid a_{ij} \mid$
$\forall j$ *A is column diagonally dominant if* $\mid a_{ii} \mid > \sum_{i,i\neq j} \mid a_{ij} \mid$

6.14 Sparse matrices

$A = (a_{ij})$ only "few" $a_{ij} \neq 0$.Define

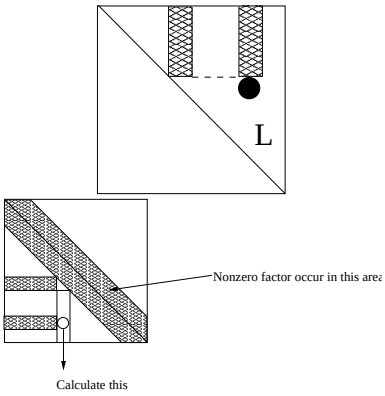
$$\text{fill} = \frac{\text{number of nonzero entries}}{\text{number of all entries}}$$

typically as $m, n \to \infty \Rightarrow$ fill $\to 0$

Example 49 $n = m = 1.5 \cdot 10^6$
nonzero $\sim 5 \cdot 10^7$
fill $= \frac{5 \cdot 10^7}{(1.5)^2 \cdots 10^1 2} \approx 10^{-5}$

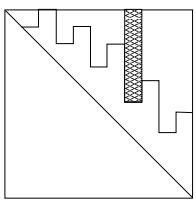
Example 50 *For a permutation matrix, fill is* $\frac{1}{n}$

Storing sparse matrices
easiest: in three arrays, i, j,a_{ij} (a_{ij}-format)
LL^T Decomposition on band matrices



We already know that LU decomposition preserves band structure
. In fact:

Theorem 51 LL^T *decomposition preserve skyline structure*



Chapter 7

Iterative methods for linear systems

Direct method: full A $n \times n$ $O(n^3)$ operations
Iterative method: involving $x \mapsto Ax$ requires only $O(mn^2)$ where m is number of iterations and only $O(mN)$,
where N is the number of nonzeros, if A is sparse.

Iterative methods share the main idea with Newton's method. Recall Newton's method:

$$x \leftarrow x - \alpha f(x) \quad \alpha = (f'(x))^{-1}$$

$$\underbrace{f(x) = 0}_{\text{equation}} \Leftrightarrow \underbrace{x = F(x)}_{\text{fixed point}}$$

Iterations: $x^{(k+1)} = F(x^{(k)})$

Consider system $Ax = b$

$$Ax = b \Leftrightarrow Ax - b = 0 \Leftrightarrow x = x - Q(Ax - b)$$

Lemma 52 *Let $A \in \mathbb{R}^{n \times n}$, $Q \in \mathbb{R}^{n \times n}$, regular, then $Ax = b \Leftrightarrow x = x - Q(Ax - b)$*

$$Ax = b \Rightarrow x = x + 0 = x - Q(Ax - b)$$
$$x = x - Q(Ax - b) \Rightarrow Q(Ax - b) = 0 \Rightarrow Ax = b$$

7.1 Stationary linear iterative method

Definition 53 *Stationary linear iterative method is $x^{(k+1)} = x^{(k)} - Q(Ax^{(k)} - b)$*

Definition 54 *Q is preconditioner*

Alternative view

$A = Q^{-1} + (A - Q^{-1})$: Splitting of A

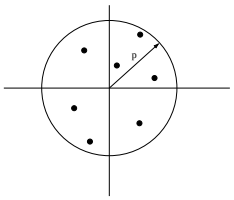
$$\begin{aligned} Ax = b &\Leftrightarrow Q^{-1}x + (A - Q^{-1})x = b \\ &\Leftrightarrow Q^{-1}x = -(A - Q^{-1})x = b \\ &\Leftrightarrow x = -Q(A - Q^{-1})x + Qb \\ &\Leftrightarrow x = x - Q(Ax - b) \end{aligned}$$

Convergence of iterations

$$x^{(k+1)} = F(x^{(k)}) \quad F(x) = x - Q(Ax - b) = (I - QA)x - Qb$$
$$F'(x) = 1 - Q(Ax - b)$$

Banach contraction theorem $\Rightarrow$ if $\|I - QA\| = q < 1$ in some norm induced by matrix norm. $\|\cdot\|$, then:
i)$\exists!$ $x^* : x^* = F(x^*)$ $(\Leftrightarrow Ax^* - b = 0)$
ii)$\forall x^{(\cdot)} \in \mathbb{R}^n$, $x^{(k+1)} = F(x^{(k)}) \to x^*$
iii) $\|x^{(k+1)} - x^*\| \leq \frac{q^k}{1-q}\|x^{(1)} - x^{(0)}\|$
iv) $\|x^{(k+1)} - x^*\| \leq q^k\|x^{(0)} - x^*\|$
(In Banach: $\|x^{(k+1)} - x^*\| = F(x^{(k)}) - F(x^*)\| = q\|x^{(k)} - x^*\|$)

Definition 55 *If $A \in \mathbb{C}^{n\times n}$ then $\rho(A)$=$\{\max \mid \lambda \mid : \lambda$ eigenvalue of $A\}$*



Theorem 56 *If $\|\cdot\|$ is a norm induced by a vector norm, then $\rho(A) \leq \|A\|$*

Proof.

$$\begin{aligned} Ax = \lambda x \qquad & x \neq 0 \\ \Rightarrow \qquad & \mid \lambda \mid \|x\| \leq \|A\|\|x\| \\ \Rightarrow \qquad & \mid \lambda \mid \leq \|A\| \end{aligned}$$

■

Theorem 57 (Householder) *If $A \in \mathbb{C}^{n\times n}$, then*

$$\rho(A) = \inf_{\|\cdot\|} \|A\|,$$

where inf *is taken over all norms induced by a vector norm.*

Remark 58 *$\rho(A) = inf\, \|A\|$ means*
(i)$\forall \|\cdot\|$ $\rho(A) \leq \|A\|$
(ii)$\forall \epsilon > 0$ $\exists \|\cdot\|$, $\|A\| < \rho(A) + \epsilon$

Proof of Householder's theorem. (i) done already

(ii) Shur's theorem from linear algebra: For any $A \in \mathbb{C}^{n\times n}$ there exist a unitary matrix U and upper triangular T such that $A = UTU^*$

Recall from linear algebra
$U^* = \overline{U}^T$ complex conjugate transpose $U \in \mathbb{C}^{n\times n}$ is unitary if $U^*U = I(\Leftrightarrow U^* = U^{-1})$
λ eigenvalue of T $\Leftrightarrow Tu = \lambda u \Leftrightarrow \underbrace{UTU^*}_{A} v = \lambda v$, $v = Uu$
$\rho(T) = \rho(A)$ Let $\epsilon > 0$
if we find norm $\|\cdot\|_\epsilon$ such that $\|T\|_\epsilon < \rho(T) + \epsilon$
$\|T\|_\epsilon = \max_{u\neq 0} \frac{\|Tu\|_\epsilon}{\|u\|_\epsilon} = \max_{u\neq 0} \frac{\|U^*AUu\|_\epsilon}{\|u\|_\epsilon} = \max_{v\neq 0} \frac{\|U^*Av\|_\epsilon}{\|U^*v\|_\epsilon} = \max \frac{\|Av\|_{\tilde{\epsilon}}}{\|v\|_{\tilde{\epsilon}}} = \|A\|_{\tilde{\epsilon}}$
$T = U^*AU$ $v = Uu$ $u = U^*v$

Need to find $\|\cdot\|_\epsilon$ so that $\|T\|_\epsilon < \rho(T) + \epsilon$ $\widetilde{t}_{ij} = \epsilon^{-i} t_{ij} \epsilon^j = t_{ij}\epsilon^{j-i}$
$t_{ij} = 0$ unless $j \le i$
$\left\|\widetilde{T}\right\|_\infty$

$$\left\|\widetilde{\mathbf{T}}\right\| = \begin{pmatrix} t_{11} & t_{12}\epsilon & t_{13}\epsilon^2 & & & \\ & t_{22} & \ddots & \ddots & & \\ & & \ddots & \ddots & & \\ & & & & t_{n-1,n-1} & t_{n-1,n}\epsilon \\ & & & & & t_{n,n} \end{pmatrix}$$

$\left\|\widetilde{T}\right\|_\infty \le \max_i\{\|t_{ii}\| + \epsilon \sum_{j=1}^n \|t_{ij}\|$
$\le \underbrace{\max_i \|t_{ii}\|}_{=\rho(A)} + \epsilon n \max_{ij} \|t_{ij}\|$

$\left\|\widetilde{T}_\epsilon\right\|_\infty = \max_{w\neq 0} \frac{\left\|\widetilde{T}_\epsilon v\right\|_\infty}{\|v\|_\infty} = \max_{wneq0} \frac{\left\|D_\epsilon^{-1} T D_\epsilon v\right\|_\infty}{\left\|D_\epsilon^{-1} w\right\|_\infty} = \max_{wneq0} \frac{\|Tw\|_\epsilon}{\|w\|_\epsilon}$
$\|w\|_\epsilon = \left\|D_\epsilon^{-1} w\right\|_\infty$, $\widetilde{T}_\epsilon = D_\epsilon^{-1} T D_\epsilon$, $v = D_\epsilon^{-1} w$, $w = D_\epsilon v$
$\|v\|_{\widetilde{\epsilon}} = \|U^* v\|_\epsilon$
$\|w\|_\epsilon = \left\|D_\epsilon^{-1} w\right\|_\infty$
$\|v\|_{\widetilde{\epsilon}} = \left\|D_\epsilon^{-1} v^* v\right\|_\infty$
Then

$$\begin{aligned} \|A\|_{\widetilde{\epsilon}} &= \max_v \frac{\|Ax\|_{\widetilde{\epsilon}}}{\|v\|_{\widetilde{\epsilon}}} \\ &= \max_{u\neq 0} \frac{\|Tu\|_\epsilon}{\|u\|_\epsilon} \\ &= \max_{w\neq 0} \frac{\left\|D_\epsilon^{-1} T w\right\|_\infty}{\left\|D_\epsilon^{-1} w\right\|_\infty} \\ &= \max_{w\neq 0} \frac{\left\|D_\epsilon^{-1} U^* A U D_\epsilon D_\epsilon^{-1} w\right\|_\infty}{\left\|D_\epsilon^{-1} U^* A U D_\epsilon\right\|_\infty} \\ &= \left\|D_\epsilon^{-1} U^* A U D_\epsilon\right\|_\infty \end{aligned}$$

■

7.1.1 Error Analysis of iterative methods

Assume A regular, denote x^* the exact solution $Ax^* = b$ $x^* \implies x^* - Q\underbrace{(Ax^* - b)}_{=0}$

substract
$x^{(k+1)} - x* = x^{(k)} - x^* - Q(A(x^{(k)} - x^*))$
denote $e^{(k)} = x^{(k)} - x^*$ =error in k-th iteration $e^{(k+1)} = (I - QA)e^{(k)}$ error transformation equation
denote $r^{(k)} = Ax^k - b$ = residual $r^k = Ae^{(k)}$
$r^k = A\underbrace{(x^{(k)} - x^*)} = Ax^{(k)}$

Theorem 59 *Iterations $x^{(k+1)} = Bx^{(k)} + f$ converge for any $x^{(0)}$ to $x^* = Bx^* + f \Leftrightarrow \rho(B) < 1$*

Proof.
(a) $\Leftarrow$ direction
$\rho(B) < 1 \Rightarrow \exists \|\cdot\| : \quad \|B\| < 1$

$x \longmapsto Bx + f$ is a contraction in $\| \cdot \|$ norm
$\Rightarrow \exists x^* : x^* = Bx^* + f, x^{(k)} \to x^*$ as $k \to \infty$ by Banach contraction theorem
(b) $\Rightarrow$ direction
Instead: $\rho(B) \geq 1 \Rightarrow$ either x^* doesn't exist or $\exists x^{(0)} : x^{(k)} \nrightarrow x^*$
Suppose $\rho(B) \geq 1$
1) if x^* doesn't exist, done
2) x^* exists $e^{(k)} = x(k) - x^*$

$$x^{(k+1)} = Bx^{(k)} + f$$

$$x^* = Bx^* + f$$

Subtract above second equation from the first equation

$$\underbrace{x(k+1) - x^*}_{e^{(k+1)}} = \underbrace{B(x^{(k)} - x^*)}_{Be(k)}$$

$\rho(B) \geq 1$
$\exists \lambda \quad | \lambda | \geq 1, \quad Bu = \lambda u \quad u \neq 0$
In the case, $f \in \mathbb{C}^n, \quad B \in \mathbb{C}^{n \times n}$
Pick $e^{(0)} = u$
$e^{(0)} = x^* + e^{(0)} = x^* + u$
$e^{(k)} = \lambda^k e^{(0)} \nrightarrow 0 \quad$ QED
In the case, $f \in \mathbb{R}^n$, B$\in \mathbb{R}^{n \times n}$
If $| \lambda | = \rho(B)$, $\lambda \in \mathbb{R} \Rightarrow$ Same as above
If $\lambda_{1,2} = \mathrm{Re}\lambda \pm i\mathrm{Im}\lambda \quad u_{1,2} = \mathrm{Re}u \pm i\mathrm{Im}u$

$$Bu_1 = \lambda_1 u_1 \quad Bu_2 = \lambda_2 u_2 \quad \lambda_1 = \bar{\lambda}_2 u_1 = \bar{u}_2$$

$$\begin{aligned} \text{Pick} \quad e^{(0)} &= u_1 + u_2 \quad (\text{if} Reu_1 = Reu_2 \neq 0, Re\lambda_1 = Re\lambda_2 \neq 0 \\ &= u_1 + \bar{u}_1 \neq 0 \\ e^{(k)} &= \lambda_1^k u_1 + \bar{\lambda}_1^k \bar{u}_1 = Re(\lambda_1^k u_1) \end{aligned}$$

Case eigenvalues of B purely imaginary if B has imaginary eigenvalues
$\Rightarrow B^2$ has real eigenvalues of λ^2
$| \lambda_1 | \geq 1 \quad \lambda_1^2 \in \mathbb{R} \quad B^2 u_1 = \lambda_1^2 u_1 \quad | \lambda_1^2 | \geq 1$

$$e^{(2)} = \lambda_1^2 e^{(0)} \quad e^{(0)} = u$$

$$\vdots$$

$$e^{(2k)} = \lambda_1^{2k} e^{(0)} \nrightarrow 0$$

■

$$x^{(k+1)} = Bx^{(k)} + f$$

Example: $x^{(k)}$ is density of neutron flow in a reactor in time t_k
f is external flux
B is $b_{ij} \geq 0$

$$\begin{aligned} \rho(B) &< 1\text{fixed point} \\ \rho(B) &= 1\text{steady reaction} \\ \rho(B) &> 1\text{Boom} \end{aligned}$$

Theorem 60 (Perron-Frobenius) *If $B \in \mathbb{R}^{n\times n}$, $b_{ij} \geq 0$ then here exists eigenvalue λ of B, $\lambda = \rho(B)$ and the associated eigenvector $u \geq 0$. If B is irreducible, (cannot be permuted into the form*

$$\begin{pmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{pmatrix}$$

by same permutation of rows and columns), then $\rho(B)$ is the only eigenvalue of B such that $\mid \lambda \mid = \rho(B)$, and eigenvector $u > 0$.

7.1.2 Neumann Series

Now interpret iterations as multiplication by a matrix:

$$\begin{aligned} x^{(k+1)} &= Bx^{(k)} + f, \quad x^{(0)} \\ x^{(1)} &= f \\ x^{(2)} &= Bf + f = (B+I)f \\ x^{(3)} &= B(B+I)f + f = (B^2 + B + I)f \\ &\vdots \\ x^{(k)} &= (B^{k-1} + B^{k-2} + \cdots + I)f \\ x^{*} &= \sum_{k=0}^{\infty}(B^k f) \quad x^* = (1-B)^{-1}f \end{aligned}$$

(1 is not eigenvalue of B $\Rightarrow$ 0 is not eigenvalue of $I - B$)

Theorem 61 *If $\rho(B) < 1$ then $(1-B)^{-1} = \sum_{k=0}^{\infty} B^k$, that is, $\|(1-B)^{-1} - \sum_{k=0}^{\infty} B^k\| \to 0$ for any norm, as $n \to \infty$.*

Proof. Enough to prove this one specific norm, choose $\|\cdot\|$ so that $\|B\| < 1$

$$R_n = (1-B)^{-1} - \sum_{k=0}^{n} B^k$$

$$\begin{aligned} (1-B)R_n &= I - (I-B)\sum_{k=0}^{n} B^k \\ &= I - I - B - \cdots - B^n + I + B + \cdots + B^{n+1} \\ &= I + B^{n+1} \end{aligned}$$

$$\|B^{n+1}\| \leq \|B\|^{n+1} \to 0$$

$$(I-B)R_n = I + B^{n+1}$$

$$\begin{aligned} &\Rightarrow \quad R_n = (I-B)^{-1}B^{n+1} \\ &\Rightarrow \quad \|R_n\| \leq \|(I-B)^{-1}B^{n+1}\| \leq \|(I-B)^{-1}\|\|B\|^{n+1} \to 0 \quad n \to \infty \end{aligned}$$

■

7.2 Basic iterative methods

Recall the general form of a linear stationary method is

$$x^{(k+1)} = x^{(k)} - Q(Ax^{(k)} - b)$$

The error transformation matrix is

$$B = 1 - QA$$

Basic choices of Q:

1. $Q = A^{-1}$ $B = 0$ direct solver
2. $A = LU + E$ $Q = U^{-1}L^{-1}$ Iterative Refinement
3. $Q = \alpha I$ $B = I - \alpha A$ Richardson's method
4. $Q = \omega D^{-1}$ Jacobi (overrelaxed if $\omega > 1$,underrelaxed if $\omega < 1$)
5. $Q = (\frac{1}{\alpha}D - L)^{-1}$ where L is the strictly lower triangular part of A SOR, Gauss-Seidel if $\alpha = 1$

7.2.1 Iterative refinement

$A = LU + E$ $Q = U^{-1}L^{-1}$

E can be from rounding during LU decomposition, or from approximate decomposition on purpose: drop small nonzeros in the factors to preserve sparsity - known as incomplete LU (ILU) algorithms

Iterations:

$$\begin{aligned} Q &= U^{-1}L^{-1} \\ x^{(0)} &= 0 \\ x^{(1)} &= 0 - U^{-1}L^{-1}(A \cdot 0 - f) \\ &= U^{-1}L^{-1} \\ x^{(k+1)} &= x^{(k)} - U^{-1}L^{-1}(Ax^{(k)} - f) \quad k \geq 1 \end{aligned}$$

implementation:

$r^{(k)} = Ax^{(k)} - f$: residual

$Lu^{(k)}=r^{(k)}$

$Uv^{(k)}=u^{(k)}$

$x^{(k+1)}=x^{(k)} - \underbrace{v^{(k)}}_{\text{correction}}$

7.2.2 Richardson iteration

$Q = \omega I$ $x^{(k+1)} = x^{(k)} - \omega(Ax^{(k)} - b)$

$B = I - \omega A$

$\lambda_B = I - \omega\lambda_A$

$Re\lambda_A > 0 \Leftarrow$ for small ω, $|\lambda_B| < 1$

Theorem 62 *If $Re\lambda_A > 0$ for all $\lambda_A \in \sigma(A)$ then $\rho(I - \omega A) < 1$ for $w > 0$ small enough, $0 < \omega < \min_{\lambda_A \in \sigma(A)} \frac{2Re\lambda_A}{|\lambda_A|^2}$*

Proof. Suppose $\lambda_A = \operatorname{Re}\lambda_A + i \operatorname{Im}\lambda_A$
$\lambda_B = 1 - \omega \operatorname{Re}\lambda_A - i\omega \operatorname{Im}\lambda_A$
$f(w) = |\lambda_B|^2 = (1 - w \operatorname{Re}\lambda_A)^2 + (\omega \operatorname{Im}\lambda_A)^2$, $0 < \omega < \frac{2 \operatorname{Re}\lambda_A}{|\lambda_A|^2} \Rightarrow |\lambda_B| < 1$
$(I - w \operatorname{Re}\lambda_A)^2 + w^2 (\operatorname{Im}\lambda_A)^2$
$= I - 2w \operatorname{Re}\lambda_A + w^2 |\lambda_A|^2 = f(w)$
$f(0) = 1$
$f \to \min \Leftrightarrow f'(\omega) = 0 \Leftrightarrow -2 \operatorname{Re}\lambda_A + 2\omega |\lambda_A|^2 = 0$
$\Leftrightarrow \omega = \frac{\operatorname{Re}\lambda_A}{|\lambda_A|^2}$
$f\left(\frac{2 \operatorname{Re}\lambda_A}{|\lambda_A|^2}\right) = 1$ ■

Corollary 63 *If A is s.p.d then Richardson's interations converge $\Leftrightarrow 0 < \omega < \frac{2}{\rho(A)}$*

7.2.3 Jacobi method

$Q = \omega D^{-1}$, $D = \operatorname{diag} A$
$\omega = 1$ Jacobi
$\omega > 1$ overrelaxed
$\omega < 1$ underrelaxed
$x^{(k+1)} = x^{(k)} - \omega D^{-1}\left(Ax^{(k)} - b\right)$
iteration matrix $B = I - \omega D^{-1} A$
Practical implementation: step k $\quad r^{(k)} = Ax^{(k)} - b$, $x^{(k+1)} = x^{(k)} - \omega r^{(k)}$

Definition 64 *$A = \mathbb{C}^{n\times n}$ or $\mathbb{R}^{n\times n}$ is row diagonally dominant if*
$\forall_i \sum_{j=1, j\neq i}^{n} |a_{ij}| < |a_{ii}|$

Theorem 65 *If A is row diagonally dominant, then the Jacobi method with $\omega = 1$ converges.*
Proof. $B = I - D^{-1}A$
$b_{ij} = -\frac{a_{ij}}{a_{ii}}$ $i \neq j$
$b_{ii} = 1 - \frac{a_{ii}}{a_{ii}} = 0$

$$\begin{aligned} \sum_{j=1}^{n} |b_{ij}| &= \sum_{j=1, j\neq i}^{n} \frac{|a_{ij}|}{|a_{ii}|} \\ &\Longrightarrow \quad 1 > \max_i \frac{\sum_{j=1, j\neq i}^{n} |a_{ij}|}{|a_{ii}|} = \max_i \sum_{j=1}^{n} |b_{ij}| = \|B\|_\infty \end{aligned}$$

■

Jacobi for s.p.d matrices.
suppose A is s.p.d $\Rightarrow a_{ii} > 0$
$B = I - \omega D^{-1} A$

$$\mathbf{D} = \begin{pmatrix} a_{11} & & \\ & \ddots & \\ & & a_{nn} \end{pmatrix} \mathbf{D^{\alpha}} = \begin{pmatrix} a_{11}^{\alpha} & & \\ & \ddots & \\ & & a_{nn}^{\alpha} \end{pmatrix}$$

$D^{\frac{1}{2}} B D^{-\frac{1}{2}} = I - \omega \overbrace{D^{-\frac{1}{2}} A D^{-\frac{1}{2}}}^{s.p.d}$
$\Rightarrow \sigma(D^{\frac{1}{2}} B D^{-\frac{1}{2}} = \sigma(B)$
analysis of Richardson's method $\rho\left(D^{\frac{1}{2}} A D^{-\frac{1}{2}}\right) < 1 \Leftarrow 0 < \omega < \frac{2}{\rho(D^{-1}A)}$

Theorem 66 ***Corollary 67*** *If A is s.p.d, $D = \mathrm{diag}(A)$, then Jacobi method is converges iff $0 < \omega < \frac{2}{\rho(D^{-1}A)}$*

Another view on Jacobi as simultaneous iteration when $\omega = 1$:

$$\begin{array}{c} a_{11}x_1^{(k+1)} + a_{12}x_2^{(k)} + \cdots\cdots + a_{1n}x_n^{(k)} = b_1 \\ a_{21}x_1^{(k)} + a_{22}x_2^{(k+1)} + \cdots\cdots + a_{2n}x_n^{(k)} = b_2 \\ \vdots \\ a_{n1}x_1^{(k)} + a_{n2}x_2^{(k)} + \cdots\cdots + a_{nn}x_n^{(k+1)} = b_n \end{array}$$

$A = D - L - U$
$Dx^{(k+1)} + (A - D)\, x^{(k)} = b$
$x^{(k+1)} = x^{(k)} + D^{-1}\left(Ax^{(k)} - b\right)$
over and under relaxation: add more or less of the correction to $x^{(k)}$
$D\tilde{x} + (A - D)\, x^{(k)} = b$
$x^{(k+1)} = x^{(k)} + \omega\left(\tilde{x} - x^{(k)}\right)$

7.2.4 Gauss-Seidel

Use every values as soon as computed:

$$\begin{array}{c} a_{11}x_1^{(k+1)} + a_{12}x_2^{(k)} + \cdots\cdots + a_{1n}x_n^{(k)} = b_1 \\ a_{21}x_1^{(k+1)} + a_{22}x_2^{(k+1)} + \cdots\cdots + a_{2n}x_n^{(k)} = b_2 \\ \vdots \\ a_{n1}x_1^{(k+1)} + a_{n2}x_2^{(k+1)} + \cdots\cdots + a_{nn}x_n^{(k+1)} = b_n \end{array}$$

Another view: cyclical relaxation: in turn, from equation i compute x_i (that is, relax x_i while keeping all other unknows fixed)

$$\sum_{j=1}^{n} a_{ij}x_j = b_i$$

Algorithm 68 (Gauss-Seidel, successive relaxation method) *for $i = 1 : n$*
compute x_i from equation i:
$x_i = \frac{1}{b_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j\right)$
end

A major improvement on the Gauss-Seidel method was adding more correction (overrelaxation):

Algorithm 69 (Successive Overrelaxation Method, SOR) *for $i = 1 : n$*
$x_i = x_i - \omega\frac{1}{a_{ii}}\left(\sum_{j=1}^{n} a_{ij}x_j - b_i\right)$
end

Write SOR in the form of linear stationary iterative method:

$$x_i^{(k+1)} = x_i^{(k)} - \omega\frac{1}{a_{ii}}\left(\sum_{j=1}^{i-1} a_{ij}x_j^{(k-1)} + \sum_{j=i+1}^{n} a_{ij}x^{(k)} - b_i\right)$$

$$A = D - L - U$$

$$D=\begin{pmatrix} a_{11} & & & 0 \\ & a_{22} & & \\ & & \ddots & \\ 0 & & & a_{nn} \end{pmatrix} L=\begin{pmatrix} 0 & \cdots & \cdots & 0 \\ -a_{21} & 0 & & \vdots \\ \vdots & & \ddots & \vdots \\ -a_{n1} & \cdots & -a_{nn-1} & 0 \end{pmatrix} U=\begin{pmatrix} 0 & -a_{21} & \cdots & -a_{1n} \\ \vdots & 0 & \ddots & \vdots \\ \vdots & & \ddots & -a_{n-1n} \\ 0 & \cdots & 0 & 0 \end{pmatrix}$$

multiply by a_{ii}, terms involving $x_j^{(k+1)}$ to the left.

$$\omega\left(\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)}\right)+a_{ii}x_i^{(k+1)}=a_{ii}x_i^{(k)}\omega\left(\sum_{j=i+1}^{n} a_{ij}x_j^{(k)}-b+a_{ii}x_i^{(k)}\right)$$

$$\left(-\omega L+D\right)x^{(k+1)}=D_x^{(k)}\omega\left(-Ux^{(k)}-b+Dx^{(k)}\right)$$

$$-\omega\left(-L+\frac{1}{\omega}D\right)x^{(k+1)}=\frac{1}{\omega}\omega Dx^{(k)}-\omega\underbrace{\left(-Ux^{(k)}-b+Dx^{(k)}-Lx^{(k)}\right)}_{Ax^{(k)}-b}-\omega Lx^{(k)}$$

$$\left(\frac{1}{\omega}D-L\right)x^{(k+1)}=\left(\frac{1}{\omega}D-L\right)x^{(k)}-\left(Ax^{(k)}-b\right)$$

$$x^{(k+1)}=x^{(k)}-\left(\frac{1}{\omega}D-L\right)^{-1}\left(Ax^{(k)}-b\right)$$

$$B=I-QA\quad Q=\left(\frac{1}{\omega}D-L\right)^{-1}$$

7.3 Descent methods

Assume A is s.p.d. Given a directions d minimize

$$J\left(x\right)=\frac{1}{2}x^TAx-x^Tb$$

in this direction. Recall that

$$\nabla J\left(x\right)=Ax-b$$

so

$$\begin{aligned} J\left(x+td\right)\rightarrow\min_t \quad &\Leftrightarrow \quad \left\langle \nabla J\left(x+td\right),d\right\rangle =0 \\ &\Leftrightarrow \quad d^T\left(A\left(x+td\right)-b\right)=0 \\ &\Leftrightarrow \quad t=-\frac{\left(Ax-b\right)^Td}{D^TAd} \end{aligned}$$

Here, $\langle x,y\rangle=x^Ty$. Alternatively, let $f(t)=J\left(x+td\right)$ and apply the chain rule to obtain the same result. When we are given a sequence of directions, we have

Algorithm 70 (Descent method) *Given* $d^{(k)}$, $k=1,2,\ldots$
for $k=1,2,\ldots$
$r^{(k)}=b-Ax^{(k)}$
$t_k=\frac{\left\langle r^{(k)},d^{(k)}\right\rangle}{\left\langle d^{(k)},Ad^{(k)}\right\rangle}$
$x^{(k+1)}=x^{(k)}+t_kd^{(k)}$
end

When $d^{(k)}$ are chosen as the columns of identity matrix, we obtain one sweep of the Gauss-Seidel method. Hence, we have

Theorem 71 *If A is s.p.d. and $0 < \omega < 2$,then in the SOR method, $J(x)$ decreases in every iteration.*

From this, one can show that SOR converges for $0 < \omega < 2$.

When $d^{(k)} = \nabla J(x)$, the descent method becomes the *steepest descent method*. The steepest descent method is quite slow. One can do better by a smarter choice of directions.

7.3.1 Conjugate gradients

Inner product $\langle x, y\rangle_A = x^T Ay = \langle x, Ay\rangle = \langle Ax, y\rangle$
$\langle x, y\rangle_A = 0 \Leftrightarrow$ x,y are A-conjugate (A-orthogonal)

Algorithm 72 (Conjugate gradients - theoretical) *given $x^{(0)}$, b, A, M, ε*
$r^{(0)} = b - Ax^{(0)}$
$v^{(0)} = r^{(0)}$
for $k = 0, \cdots, M$
$t_k = \frac{\langle r^{(k)}, v^{(k)}\rangle}{\langle v^{(k)}, Av^{(k)}\rangle}$
$x^{(k+1)} = x^{(k)} + t_k v^{(k)}$
$r^{(k+1)} = r^{(k)} + t_k Av^{(k)}$
$s_k = -\frac{\langle r^{(k+1)}, Av^{(k)}\rangle}{\langle v^{(k)}, Av^{(k)}\rangle}$
$v^{(k+1)} = r^{(k)} + s_k v^{(k)}$
end
Here, $v^{(k+1)} = r^{(k)} + s_k v^{(k)}$ s_k determined from the condition that it is A-orthogonal to the previous search direction: $\langle v^{(k+1)}, v^{(k)}\rangle_A = 0 \Rightarrow \langle r^{(k+1)} + s_k v^{(k)}, Av^{(k)}\rangle = 0 \quad s_k = -\frac{\langle r^{(k+1)}, Av^{(k)}\rangle}{\langle v^{(k)}, Av^{(k)}\rangle}$
The miracle of Conjugate gradients is that then $v^{(k+1)}$ is orthogonal to all previous search directions.

Theorem 73

$$\left\langle v^{(k+1)}, Av^{(i)}\right\rangle = 0 \quad \forall i < k$$

$$\left\langle r^{(k)}, v^{(k)}\right\rangle = \left\langle r^{(k)}, r^{(k)}\right\rangle$$

$$\left\langle r^{(k+1)}, Av^{(k)}\right\rangle = \left\langle r^{(k+1)}, r^{(k+1)}\right\rangle$$

$$\frac{\langle r^{(k+1)}, r^{(k+1)}\rangle}{\langle v^{(k)}, Av^{(k)}\rangle} = \frac{\langle r^{(k+1)}, r^{(k+1)}\rangle}{\langle r^{(k)}, r^{(k)}\rangle}$$

The conjugate gradient algorithm as written is not numerically stable. The reason is essentially that in the inner products $\langle r^{(k)}, v^{(k)}\rangle$ and $\langle r^{(k+1)}, Av^{(k)}\rangle$, the components of the inner product will be large while the inner product itself will be small. The above theorem allows to rewrite CG to get rid of this instability.

Algorithm 74 (Conjugate gradients) *CG-version 1*

$r^{(0)} = b - Ax^{(0)}$
$v^{(0)} = r^{(0)}$
for k=1,$\cdots$,M
$t_k = \frac{\langle r^{(k)}, r^{(k)}\rangle}{\langle v^{(k)}, Av^{(k)}\rangle}$

$x^{(k+1)} = x^{(k)} + tv^{(k)}$
$r^{(k+1)} = r^{(k)} + tAv^{(k)}$
$S_k = \frac{\langle r^{(k+1)}, r^{(k+1)} \rangle}{\langle r^{(k)}, r^{(k)} \rangle}$
$v^{(k+1)} = r^{(k+1)} + S_k v^{(k)}$
end

Algorithm 75 (Conjugate gradient code) $r = b - Ax$
$v = r$
$n = \langle r, r \rangle$ *for k=1,*$\cdots$*,M*
$z = Av$
$t = \frac{n}{\langle v, z \rangle}$
$x = x + tv$
$r = r - tz$
$n_{old} = n$
$n = \langle r, r \rangle$
$s = \frac{n}{n_{old}}$
$v = r + sv$
end

Op	Count
Ax	1
dot	2
axpy	3

Storage
r,b (in/out)
v,z (Scratch)

7.3.2 Preconditioned CG (PCG)

general linear stationary iterative method:

Richardson	*Preconditioned*
$x \leftarrow x - \omega(Ax - b)$	$x \leftarrow x - Q(Ax - b)$
A	$QA \approx I$
A	$S^T AS \quad Q = SS^T$

to combine CG with preconditioning: run CG on $S^T AS$ instead of A substitution $x = S\bar{x} \quad \hat{x} = S^T r$ run CG on $\hat{x}, \hat{A}$ after some algebra,

Algorithm 76 (Preconditioned Conjugate Gradients, PCG) *given x, b, A, M,* ε
$r = b - Ax$
$z = Qr$
$v = z$
$c = \langle z, r \rangle$ *for k=1,*$\cdots$*,M*
$z = Av$
$t = \frac{c}{\langle v, z \rangle}$
$x = x + tv$
$r = x - tz$
$z = Qr$
$d = \langle z, r \rangle$
$(z < \varepsilon \langle r, r \rangle < \varepsilon)$ *stop* *end* $v = z + (d/c)v$
$c = d$
end

Op	Count
Qx	1
Ax	1
dot	2
axpy	3

Storage
x,b (in/out)
v,z (Scratch)

7.3.3 Properties of CG

(i) if $A \in \mathbb{R}^{n\times n}$, CG finds exact solution in n steps at most (in exact arithmetic)
(ii) $e^{(k)} = x^* - x^{(k)}$

$$\begin{aligned} \|e^{(k)}\|_A &= \min_{p\in\mathcal{P}_k, p(0)=1} \|P(A)e^{(0)}\|_A \\ &\leq \min_{p\in\mathcal{P}_k, p(0)=1} \max_{\lambda\in\sigma(A)} \mid p(\lambda) \mid \|e^{(0)}\|_A \end{aligned}$$

because $\|p(A)\| = \rho(p(A)) \quad \rho(p(A)) = \max_{\lambda\in\sigma(A)} \mid p(\lambda) \mid$
For PCG, replace $\sigma(A)$ by $\sigma(QA)$
by choosing $p \in \mathcal{P}_k$

$$\|e^{(k)}\|_A \leq 2\left(\frac{k-1}{k+1}\right)^n \|e^{(0)}\|_A$$

$$k = \frac{\lambda\max(QA)}{\lambda\min(QA)}$$

Assumption: A and Q are s.p.d.
If A not s.p.d. methods similar to CG exist but they are more expensive and do not converge so well

Chapter 8

Interpolation and approximation

8.1 Lagrange interpolation

Problem: find a polynomial from values at given points
2 points → line, linear function $p \in \mathcal{P}_1$

8.1.1 Existence and uniqueness

$\mathcal{P}_n = \{\text{set all polynomials degree} \leq n\}$

Theorem 77 *given $x_0, \ldots, x_n \in \mathbb{R}$, distinct, $y_j, \ldots, y_n \in \mathbb{R}$ there exists exactly one $p \in \mathcal{P}_n$ so that $p(x_i) = y_i, i = 0, \ldots, n$*

Proof. Uniqueness
$p, q \in \mathcal{P}_n$
$p(x_i) = q(x_i) = y_i, i = 0, \ldots, n$
$\Rightarrow p - q \in \mathcal{P}_n$
$(p-q)(x_i) = 0, i = 0, \ldots, n \Rightarrow p - q$ has n+1 roots
$\Rightarrow p - q = 0 \Rightarrow p = q$
Existence by construction(Newton's construction)
$n = 0 \quad p_0(x) = C_0,\ C_0 = y_0$
$n = 1 \quad p_1(x) = p_0(x) + C_1(x - x_0),\ p_1(x_1) = y_1 \Rightarrow C_1 = \frac{y_1 - p_0(x_1)}{x_1 - x_0}$
$p_1(x_0) = \underbrace{p_0(x_0)}_{y_0} + C_1(x_0 - x_0) = 0$

$\vdots$

have $p_{n-1}(x_i) = y_i,\ i = 0, \ldots, n-1\ p_{n-1} \in \mathcal{P}_{n-1}$
$p_n(x) = p_{n-1}(x) + C_n(x - x_0)(x - x_1) \ldots\ldots (x - x_{n-1})$
$C_n = \frac{y_n - p_{n-1}(x_n)}{(x_n - x_i)(x_n - x_1)\ldots(x_n - x_{n-1})}$
■

Remark 78 *generalization*
1. Regression (least-squares fit)
find $p \in \mathcal{P}_m$, $m \leq n$ to minimize $\sum_{i=0}^{n} |y_i - p(x_i)|^2$
2. $f \in C^1$, $x_i - x_{i+1} \approx 0$, $y_i = f(x_i)$ we get condition $p'(x_i) = f'(x_i)$, which is known as Hermite interpolation (next section)

Interpolation polynomial in Newton's form
$p_n(x) = C_0 + C_1(x - x_0) + C_2(x - x_0)(x - x_1) + \cdots + C_n(x - x_0) + \cdots + (x - x_n)$
evaluated by Horner's scheme

$$p_n(x) = C_0 + (x - x_0)\left[\underbrace{C_1 + C_2(x - x_1) + \cdots}_{\text{apply the same process}}\right]$$
$p_n(x) = t_0$, $C_1 + C_2(x - x_1) + \cdots = t_1$
$t_0 = C_0 + (x - x_0)\, t_1$
$t_1 = C_1 + (x - x_1)\, t_2$
$\vdots$
$t_n = C_n(x - x_n)$

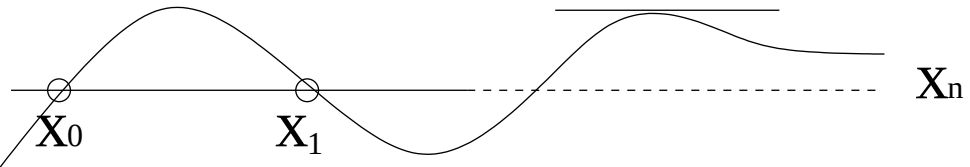
Algorithm 79 (Horner's scheme for Lagrange Interpolation polynomial) *To evaluate the Lagrange interpolation polynomial*
for $i = n - 1 : -1 : 0$
$t_i = C_i + (x - x_i)\, t_{i+1}$
end
then $p_n(x) = t_0$

Remark 80 *The algorithm used in practice is actually different, based on divided differences, cf., e.g. [KC02]*

The next theorem gives an error estimate.

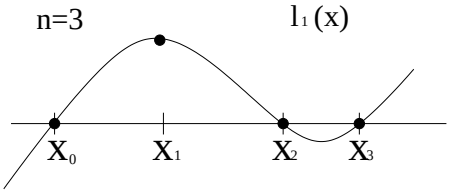
Theorem 81 $f \in C^{n+1}(a,b)$, $p \in \mathcal{P}_n$,$p(x_i)$, $i = 0, \ldots, n$, x_i, *distinct* $x \in (a,b)$, *then* $f(x) - p(x) = \frac{1}{(n+1)!} f^{(n+1)}(x - x_0) \ldots (x - x_n)$
$\xi \in$ *interval spanned by* $\{x, x_0 \ldots, x_n\} = I$

Proof. $x = x_i \Rightarrow$error is zero. assume $x \neq x_i$, $\forall_i$
denote $w(t) = (t - x_0) \ldots (t - x_n)$
$\Phi(t) = f(t) - p(t) - \lambda w(t)$
λ determined from $\Phi(x) = 0$
$\lambda = \frac{f(x) - p(x)}{w(x)}$, $w(x) \neq 0$
$\Phi(x) = \Phi(x_0) = \cdots\cdots = \Phi(x_n) = 0$ Φ has n+1 zeros.
■



Rolle's Theorem
Φ' has n zeros in I
Φ''has n-1 zeros in I
$\vdots$
$\Phi^{(n+1)}$ 1 zero in I
$\Rightarrow \exists \xi \in I$, $\Phi^{(n+1)} = 0$
$\Phi^{(n+1)}(\xi) = f^{(n+1)}(\xi) - p^{(n+1)}(\xi) - \lambda w^{(n+1)}(\xi)$
$w(t) = t^{n+1} + \ldots\ldots$
$w^{(n+1)}(t) = \underbrace{\left(t^{n+1}\right)^{(n+1)}}_{(n+1)n(n-1)\ldots+1=(n+1)!} + 0$
$0 = f^{(n+1)}(\xi) - \lambda(n+1)!$
$\lambda = \frac{f(x) - p(x)}{w(x)} = \frac{f^{(n+1)}(\xi)}{(n+1)!}$

8.1.2 Legendre form of the Lagrange interpolation polynomial



given $x_0, \ldots\ldots, x_n$, distinct
define $l_i \in \mathcal{P}_n$, $l_i(x_j) = \delta_{ij}$

Theorem 82 $p_n(x) = \sum_{i=0}^{n} y_i l_i(x_i)$

Proof. 1. $l_i \in \sigma_n \Rightarrow p_n \in \sigma_n$
2. $p_n(x_i) = \sum_{i=0}^{n} y_i \underbrace{l_i(x_j)}_{\delta_{ij}} = y_j$ ■

One can also try to determine the coefficients of the polynomial $p(x) = a_0 + \ldots\ldots + a_n x^n$ directly from the equations

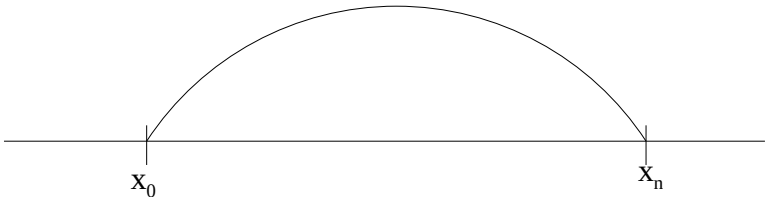
$$a_0 + a_i x_i + \ldots\ldots + a_n x_i^n = y_i, \qquad i = 0, \ldots\ldots, n$$

for unknowns $a_0, \ldots, a_n$, resulting in a system with Vandermont matrix, which is ill-conditioned for larger n.

8.1.3 Worst-case error for Lagrange interpolation

$$p \in \mathcal{P}_n$$
$$p(x_i) = f(x_i), \quad i = 0, \cdots, n$$
$$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \underbrace{(x - x_0) \cdots (x - x_n)}_{\text{build from Sin Cos}}$$

$x_0 = a, \cdots, x_n = b \quad x_k - x_{k-1} = h$
$x \in [a, b] \quad |(x - x_0) \cdots (x - x_n)| \le \frac{h}{2}\frac{h}{2} 2h 3h \cdots nh = h^{n+1}\frac{(n+1)!}{4}$
Consider $x \in [x_0, x_1]$
$|(x - x_0)(x - x_1)| \le \frac{h}{2}\frac{h}{2}$



$x \in [x_0, x_1]$
$|x - x_2| \le |x_0 - x_2| = 2h$
$|x - x_3| \le |x_0 - x_3| = 3h$
$\vdots$
$|x - x_n| \le |x_0 - x_n| = nh$

The resulting estimate is

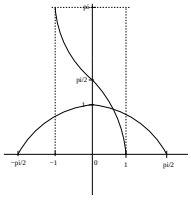
$$|f(x)-p(x)| \leq \max_{\xi} \left| \frac{f^{(n+1)}(\xi)}{(n+1)!} \right| \left(\frac{b-a}{n} \right)^{n+1} \frac{(n+1)!}{4}$$

We can do better by a smart choice of the nodes.

8.1.4 Chebyshev polynomials

$$T_n(x) = \cos(n \arccos x) \quad -1 \leq x \leq 1$$

$T_0(x) = 1$
$T_1(x) = \cos(\arccos x) = x$
$\cos a + \cos b = 2\cos\frac{a+b}{2}\cos\frac{a-b}{2}$
$\cos(n-1)t + \cos(n+1)t = 2\cos nt \cos t$
$\cos[(n-1)\arccos x] + \cos[(n+1)\arccos x] = 2\cos(n\arccos x)\cos(\arccos t)$
$T_{n-1}(x) + T_{n+1}(x) = 2T_n(x)x$
$T_0(x) = 1$
$T_1(x) = x$
$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$



Properties
(i) $T_n(x) \in \mathcal{P}_n, \quad n \leq 1$
(ii) $T_n(x) = 2^{n-1}x^n + \cdots \quad n \geq 1$
(iii) $x \in [-1,1] \Rightarrow |T_n(x)| \leq 1$
(iv) $T_n(1) = 1$
because x=1 $\quad \cos 0 = 1 \quad \arccos 1 = 0 \quad T_n(1) = 1$
(v) $T_n(-1) = (-1)^n$
because x=-1 $\quad \cos\pi = -1 \quad \arccos(-1) = \pi$
$\cos(n\arccos(-1)) = \cos n\pi = (-1)^n$
(vi) $T_n(x) = 0 \Leftrightarrow x = \cos\frac{(2k+1)}{2n} \quad k = 0, \cdots, n-1$
(vii) $|T_n(x)| = 1 \Leftrightarrow x = \cos\frac{k\pi}{n}, \quad k = 0, \cdots, n$
Roots

$$\begin{aligned} \cos(n\arccos x) = 0 \quad &\Leftrightarrow \quad n\arccos x = \frac{\pi}{2} + k\pi \\ &\Rightarrow \quad \arccos x = \frac{(2k+1)\pi}{2n} \in [0,\pi] \quad k = 0, \cdots, n-1 \\ &\Rightarrow \quad x = \cos\frac{(2k+1)\pi}{2n}, \quad k = 0, \cdots, n-1 \end{aligned}$$

Theorem 83 *(Optimality of Chebyshev polynomials)*
Let $p \in \mathcal{P}_n$, $p(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_0$ *then*

$$\max_{x\in[-1,1]} |p(x)| \geq \frac{1}{2^{n-1}} = \max_{x\in[-1,1]} \left| \frac{1}{2^{n-1}} T_n(x) \right|, \frac{1}{2^{n-1}} T_n(x) \text{ has leading coefficient}$$

or: $\frac{1}{2^{n-1}}T_n(x)$ *is the polynomial of degree n with leading coefficient 1 that minimizes maximum absolute value on* $[-1,1]$

Proof. Assume $p(x) = x^n + a_{n-1}x^{n-1} + \cdots \quad \max_{x\in[-1,1]} |p(x)| < 1$

$$q(x) = \underbrace{p(x) - 2^{1-n}T_n(x)}_{=0 \text{ between each min and max of } 2^{1-n}T_n(x)} \in \mathcal{P}_{n-1}$$

$\Rightarrow$ has n roots $\Rightarrow$ contradiction ■

8.1.5 Chebyshev interpolation

Let $f \in C^{n+1}(-1,1)$, define x_k by $T_{n+1}(x_k) = 0, \quad k = 0, \cdots, n$

$$x_k = \cos\frac{(2k+1)\pi}{2n+2}, \quad k = 0, \cdots, n$$

Let $x \in (-1,1)$ then $f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{2^n(n+1)!}\xi \in (-1,1)$
where, $p_n(x)$ is lagrange interpolation polynomial on $x_0, \cdots, x_n$

Proof.

$$f(x) - p_n(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\underbrace{(x-x_0)\cdots(x-x_n)}_{2^{-n}T_{n+1}(x)}$$

■

8.1.6 Caveats

There is no guarantee that Lagrange interpoloation will converge. For $n \to \infty$ we still may have

$$\lim_{n\to\infty} \max_{-1\le x\le 1} |f(x) - p_n(x)| \to +\infty \tag{8.1}$$

In fact, for any family of interplolation meshes there exists a continuous function such that (8.1) holds.

8.2 Hermite interpolation

Given f find p so that $p \in \mathcal{P}_{2n+1}$

$$p(x_i) = f(x_i)$$
$$p'(x_i) = f'(x_i)$$

where, $i = 0, \cdots, n$ (2n+2 parameters)
From Lagrange interpolation recall the basis polynomials

$$l_i \in \mathcal{P}_n, l_i(x_j) = \delta_{ij}$$

It can be shown that if

$$\begin{aligned} A_j(x) &= [1 - 2(x-x_j)l'(x)]\, l_j^2(x) \\ B_j(x) &= (x-x_j)\, l_j^2(x) \end{aligned}$$

then

$$\begin{aligned} A_i(x_i) &= \delta_{ij}, \quad A_i'(x_i) = 0 \\ B_i(x_i) &= 0, \quad B_i'(x_i) = \delta_{ij} \end{aligned}$$

This suggest construction of Hermite interpolation polynomial as

$$h_n = \sum_{i=0}^{n} f(x_i) A_i(x) + f'(x_i) B_i(x)$$

Theorem 84 *Hermite interpolation polynomial exists and is unique.*

Proof. Existence follows from the construction above. To prove uniqueness:
Suppose $p, q \in \mathcal{P}_{2n+1}$, $p(x_i) = q(x_i)$, $p'(x_i) = q'(x_i)$, $i = 0, \cdots, n$, and let $\Phi = p - q$.Then

$$\Phi \in \mathcal{P}_{2n+1},\ \Phi(x_i) = \Phi'(x_i) \quad i = 0, \cdots, n$$

and we need to show that $\Phi = 0$.Φ has $n+1$ zeros. $x_0, \cdots, x_n$
Rolle's theorem gives $\Phi' = 0$ at n points between $x_0, \cdots, x_n$
$\Rightarrow \Phi' = 0$ at $x_0, \cdots, x_n \Rightarrow n+1$ points
$\Rightarrow \Phi' = 0$ at total $2n+1$ nodes $\Phi' \in \mathcal{P}_{2n}$
$\Rightarrow \Phi' = 0$
$\Rightarrow \Phi =$ constant
$\Rightarrow \Phi = 0$
■

Theorem 85 *Let $f \in C^{2n+2}[a,b]$,and $x, x_0, \ldots, x_n \in [a,b]$. Then there exists $\xi \in (a,b)$ such that*

$$f(x) - h_n(x) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!}(x - x_0)^2 \cdots (x - x_n)^2$$

Proof. Similar to Lagrange interpolation ■

8.3 Splines, or piecewise polynomial interpolation

8.3.1 Piecewise linear interpolation

$$a = x_0 < x_1 < \cdots < x_n = b \quad x_{k+1} - x_k = h$$

Let $f \in C[a,b]$ and use Lagrange interpolation on each $[x_k, x_{k+1}]$ separately. Denote the resulting piecewise linear function by $I_1 f$ We have

$$x \in [x_k, x_{k+1}] \Rightarrow f(x) - I_1 f(x) = \frac{f''(\xi)}{2!} \underbrace{(x - x_k)(x - x_{k+1})}_{\leq \frac{h^2}{4}}$$

Theorem 86 *If $f \in C^2[a,b]$ then*

$$\|f - I, f\|_{C[a,b]} \leq \frac{h^2}{8} \|f''\|_{C[a,b]}$$

Because of that we say that linear inter2^{nd} order accurate or h^2 - accurate.
Similarly one can define piecewise quadratic on $[x_{2k}, x_{2k+2}]$ separately.

8.3.2 Piecewise cubic Hermite interpolation

Again let

$$a = x_0 < x_1 < \cdots < x_n = b \quad x_{k+1} - x_k = h$$

use Hermite interpolation on each $[x_k, x_{k+1}]$ separately get $H_3 f$, we have

$$\|f - H_3 f\| \Rightarrow f(x) - H_3 f(x) = \frac{f^{(4)}(\xi)}{4!} \underbrace{(x - x_k)^2 (x - x_{k+1})^2}_{\frac{h^4}{16}}$$

Theorem 87 *If $f \in C^4[a,b]$ then*

$$\|f - H_3 f\|_{c[a,b]} \leq \frac{h^4}{384} \left\| f^{(4)} \right\|_{C[a,b]}$$

8.3.3 Natural cubic spline

Again, let

$$a = x_0 < x_1 < \cdots < x_n = b$$

Definition 88 *Natural cubic spline is defined using prescribed values y_i on nodes x_i, $i = 1, \ldots, n$ as the solution of the problem of finding a function $f \in C[a,b]$ and $f \in C^2[x_k, x_{k+1}]$ for each k, such that*

$$f(x_i) = y_i \quad i = 0, \cdots, n, \qquad \int_a^b |f''(x)|^2 \, dx \to \min$$

Turns out the solution satisfies $f \in \mathcal{P}_3$ on each $[x_k, x_{k+1}]$, $f \in C^2[a,b]$ (that is, the first and the second derivatives are continuous across the nodes), and $f''(x_0) = f''(x_n) = 0$. In mechanics, an elastic rod that is fixed the nodes assumes the shape of a natural cubic spline. Such shapes have been in use in shipbuilding.

Chapter 9

Numerical differentiation

9.1 Taylor theorem

See section 14.1.

9.2 One-sided differences

From the definition of derivative: $f'(x) \approx \frac{f(x+h)-f(x)}{h}$ for a small h. Error estimate follows from Taylor theorem:

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2!}f''(\xi) \\ \frac{f(x+h)-f(x)}{h} &= f'(x) + \frac{h}{2}f''(\xi) \end{aligned}$$

9.3 Central differences

From Taylor theorem

$$\begin{aligned} f(x+h) &= f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f^{(3)}(\zeta) \\ f(x-h) &= f(x) - hf'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f^{(3)}(\xi) \end{aligned}$$

Subtracting we get

$$f(x+h) - f(x-h) = 2hf'(x) + \frac{2h^3}{3!}f^{(3)}(\eta)$$

which gives

$$\frac{f(x+h)-f(x-h)}{2h} = f'(x) + \frac{h^2}{6}f^{(3)}(\eta)$$

The error is bounded by $f^{(3)}$; hence formula is exact if $f \in P_2$.

9.4 Richardson extrapolation

Chapter 10

Numerical quadrature

Many authors call numerical computation of integrals quadrature, and reserve the word integration for solving ordinary differential equations. The standard reference for numerical quadrature is [DR84].

10.1 Interpolatory quadrature

10.2 Composite rules

Orthogonal polynomials

See Section 15.7.

10.3 Gaussian quadrature

Example 89 *We develop the 2 point Gaussian integration formula*

$$\int_0^1 |\log(x)| f(x)dx \approx a_0 f(x_0) + a_1 f(x_1)$$

First we find orthogonal polynomials in the inner product $\langle u, v\rangle = \int_0^1 |\log(x)| u(x)v(x)dx$. *We will find useful the formulas*

$$\begin{aligned}\int_0^1 |\log(x)| x^k dx &= \left[|\log(x)| \frac{x^{k+1}}{k+1}\right]_0^1 - \int_0^1 \frac{1}{x}\left(-\frac{x^{k+1}}{k+1}\right)dx \\ &= \int_0^1 \frac{x^k}{k+1}dx = \frac{1}{(k+1)^2}.\end{aligned}$$

Gram-Schmidt orthogonalization with $u_k = x^k$ *gives*

$$\begin{aligned}v_0 &= u_0 = 1 \\ v_1 &= u_1 - \frac{\langle v_0, u_1\rangle}{\langle v_0, v_0\rangle}v_0 = x - \frac{1}{4} \\ v_2 &= u_2 - \frac{\langle v_0, u_2\rangle}{\langle v_0, v_0\rangle}v_0 - \frac{\langle v_1, u_2\rangle}{\langle v_1, v_1\rangle}v_1 = x^2 - \frac{1}{9} - \frac{5}{7}\left(x - \frac{1}{4}\right) \\ &= x^2 - \frac{5}{7}x + \frac{17}{252}\end{aligned}$$

Using Matlab, the roots of v_2 are

$$x_0 = 0.60227690811874, \quad x_1 = 0.11200880616698$$

The requirement that the quadratures is exact for $u(x) = 1$ and $u(x) = x$ gives

$$\begin{aligned} a_0 + a_1 &= \int_0^1 |\log(x)|\, dx = 1 \\ a_0 x_0 + a_1 x_1 &= \int_0^1 |\log(x)|\, x dx = \frac{1}{4} \end{aligned}$$

and solving this system of equations by Matlab (use the stored vaues of the roots, do not type the numbers in!) gives

$$a_0 = 0.28146068096962, a_1 = 0.71853931903038$$

To verify that the quadrature rule is exact also for $u(x) = x^2$ and x^3, compute

$$\begin{aligned} a_0 x_0^2 + a_1 x_1^2 - \int_0^1 |\log(x)|\, x^2 dx &= 0 \\ a_0 x_0^3 + a_1 x_1^3 - \int_0^1 |\log(x)|\, x^3 dx &= 0 \end{aligned}$$

10.4 Convergence theory for continuous functions

Bernoulli polynomials and Euler-Maclaurin Formula

See section **??**

10.5 Romberg quadrature

From the Euler-Maclauring formula (**??**) with $2m$ in the place of n and $x = x_{i-1} + th$, we have for the trapezoidal rule

$$\begin{aligned} \int_{x_{i-1}}^{x_i} f(t)dt &= \frac{h}{2}\left(f(x_{i-1}) + f(x_i)\right) + \sum_{\substack{k=3 \\ k \text{ odd}}}^{2m} \frac{b_k h^k}{k!}\left(f^{(k-1)}(x_i) - f^{(k-1)}(x_{i-1})\right) + \\ & \frac{(-1)^{2m} h^{2m}}{(2m)!} \int_{x_{i-1}}^{x_i} B_{2m}\left(\frac{x - x_i}{h}\right) f^{(2m)}(x)dx. \end{aligned}$$

Adding over all subintervals and noting that the sum telescopes, we obtain an asymptotic expansion of the error of the composite trapezoidal rule for $f \in C^{2m}[a, b]$:

$$\int_a^b f(t)dt = I_h + C_2 h^2 + C_4 h^4 + \ldots + C_{2m-2} h^{2m-2} + h^{2m} R_{2m}$$

where

$$\begin{aligned} I_h &= h\left(\frac{f(x_0)}{2} + \sum_{i=1}^{n-1} f(x_i) + \frac{f(x_n)}{2}\right), \\ C_{2k} &= \frac{b_{2k}}{(2k)!}(f^{(2k-1)}(b) - f^{(2k-1)}(a)), \\ |R_{2m}| &\leq \left\| f^{(2m)} \right\|_{C[a,b]} \frac{(b-a)}{(2m)!} \left\| B_{2m} \right\|_{C[0,1]}. \end{aligned}$$

The specific form of the terms in the expansion is not important. We use the fact that

- C_{2k} does not depend on h
- R_{2m} depends only on b, a, m, and $f^{(2m)}$

With the asymptotic expansion, we can use Richardson's extrapolation Define

$$R(n,0) = I_h, \quad h = \frac{b-a}{2^n}.$$

and then

$$R(n,k) = \frac{4^k R(n,k-1) - R(n-1,k-1)}{4^k - 1}.$$

By combining the expansions we then get

$$\begin{aligned} I &= R(n,1) + C_4' h^4 + \ldots + C_{2m-2}' h^{2m-2} + h^{2m} R_{2m}' \\ I &= R(n,2) + C_6'' h^6 + \ldots + C_{2m-2}'' h^{2m-2} + h^{2m} R_{2m}'' \\ &\vdots \end{aligned}$$

Lemma 90 *$R(n,0)$ is the trapezoidal rule. $R(n,1)$ is the Simpson's rule. $R(n,k)$, $k \geq 3$, are quadrature rules of order h^{2k+2}, and not composite Newton-Cotes formulas.*

Lemma 91 *The error of Romberg quadrature is bounded by $(b-a)C(n,k)\left\|f^{(2n)}\right\|_{C[a,b]}$, where $C(n,k)$ are constants, so $R(n,k)$ is exact for polynomials of order $2k+1$.*

Lemma 92 *The weights of Romberg quadrature are positive.*

Proof. See, for example [Kre98, Theorem 9.30]. ■

Theorem 93 *Romberg quadrature converges for any continuous function:*

$$\forall k : \lim_{n\to\infty} R(n,k) = I$$
$$\lim_{k\to\infty} R(k,k) = I$$

Romberg quadrature provides an aposteriori error estimate: because $R(n,k-1)$ is (hopefully) much more accurate than $R(n,k-1)$, the quadrature error can be estimated as

$$|I - R(n,k-1)| \approx |R(n-1,k-1) - R(n,k-1)|$$

If we then take the extrapolated value $R(n,k)$ as the answer, we have the estimate

$$|I - R(n,k)| \lesssim |R(n-1,k-1) - R(n,k-1)|.$$

The value of the estimate is that it uses only quantities known from the computation. Warning: obviously, this estimate may fail for some functions - for a given a, b, n, k, it is easy to devise a function with wild behavior aside from the quadrature nodes, which will fool any estimate based on nodal values.

To implement Romberg quadrature efficiently, we need to avoid repeated evalulation of f : the trapezoidal rule

$$R(n,0 = h\left(\frac{f(x_0)}{2} + \sum_{i=1}^{2^n-1} f(x_i) + \frac{f(x_{2^n})}{2}\right), \quad x_i = a + ih, \quad h = \frac{b-a}{2^n},$$

contains values of f already computed for $R(n-1,0)$,which form the part of the sum for even i.This gives

$$\begin{aligned} h = b - a, &\qquad R(0,0) = \frac{h}{2}(f(a) + f(b)) \\ h = \frac{h}{2}, \quad M = 2^n, &\qquad R(n,0) = \frac{R(n-1,0)}{2} + h \sum_{\substack{i=1 \\ i \text{ odd}}}^{M-1} f(a+ih). \end{aligned}$$

Also, we put the extrapolation formula in a numerically more favorable form as adding a small update

$$R(n,k) = R(n,k-1) + \frac{1}{4^k - 1}(R(n,k-1) - R(n-1,k-1)).$$

10.6 Adaptive quadrature

Developing the idea of aposteriori estimate leads to adaptive quadrature. The goal of adaptive quadrature is to find the integral of the function to a given tolerance with minimum number of evaluations. We describe algorithm based on Simpson's rule, similar to the method used in the `quad` function in MATLAB. See [GG00] for more details. The error of Simpson's rule has the asymptotic expansion, from the beginning of the second column of the Romberg table, or directly from the Euler-Maclaurin formula. With

$$h = \frac{b-a}{4},\ x_j = a + hj$$

we have

$$\begin{aligned}
\int_a^b f(t)dt &= \overbrace{\frac{2h}{6}\left(f(x_0) + 4f\left(x_2\right) + f(x_4)\right)}^{Q_1} + C\left(2h\right)^4 + O(h^6), \\
\int_a^b f(t)dt &= \underbrace{\frac{h}{6}\left(f(x_0) + 2f(x_1) + 4f\left(x_2\right) + 2f(x_3) + f(x_4)\right)}_{Q_2} + Ch^4 + O(h^6)
\end{aligned}$$

Eliminating the h^4 terms gives the extrapolated value

$$Q = Q_2 + \frac{1}{15}(Q_2 - Q_1)$$

and we have the aposteriori estimate

$$\left| \int_a^b f(t)dt - Q \right| \lesssim \left| Q_2 - Q_1 \right|,$$

cf., (**??**).

The adaptive quadrature algorithm now proceeds as follows:

1. Given a, b, f, and tolerance τ, compute Q_1, Q_2, Q.

2. If $|Q_2 - Q_1| \leq \tau$, then Q is the answer, exit.

3. Otherwise let $c = \frac{a+b}{2}$, and apply the same algorighm twice, with $(a,b) := (a,c)$, $\tau = \tau/2$, and with $(a,b) := (c,a)$, $\tau = \tau/2$.

In a practical implementation, the values of f, once computed, are passed as function arguments rather than computed again. Note that here, the tolerance imposed on each subinterval is proportional to the subinterval length. so the sum of the error estimates will not be larger than the given tolerance. The quadrature in MATLAB does not decrease the tolerance, however. Type `edit quad` in MATLAB to see the source code. In general, it is not possible to design adaptive quadrature to deliver a result with the specified tolerance for every function.

10.7 Integration of periodic functions

Consider a periodic function $f \in C^m(\mathbb{R})$ with period M, $f(t+L) = f(t)$, and let

$$h = \frac{L}{n}, \quad x_i = ih.$$

The Euler-Maclaurin formula in one of the subintervals becomes with the substitution $x = x_{i-1} + th$

$$\begin{aligned}\int_{x_{i-1}}^{x_i} f(x)dx &= \frac{h}{2}\left(f(x_{i-1}) + f(x_i)\right) + \sum_{k=3}^{m} \frac{(-1)^k b_k h^{k-1}}{k!}\left(f^{(k-1)}(x_i) - f^{(k-1)}(x_{i-1})\right) \\ &\quad + \frac{(-1)^m h^m}{m!}\int_{x_{i-1}}^{x_i} B_m\left(\frac{x-x_i}{h}\right) f^{(m)}(x)dx\end{aligned}$$

By summation and noting that the terms with $f^{(k-1)}$ telescope, we get the rectangle rule

$$\begin{aligned}\int_0^L f(t)dt &= h\sum_{i=1}^{n} f(x_i) + R, \\ |R| &\le \frac{h^m}{m!}\sum_{i=1}^{n}\int_{x_{i-1}}^{x_i} B_m\left(\frac{x-x_i}{h}\right) f^{(m)}(x)dx \\ &\le C_m h^m \left\|f^{(m)}\right\|_{C[a,b]}, \quad C_m = \frac{L\left\|B_m\right\|_{C[a,b]}}{m!}.\end{aligned}$$

So, for periodic functions, the simplest rectangle rule has arbitrarily high order of convergence if f is smooth enough.

10.8 Improper integrals and special integration formulas

The easiest is to transform the integral before numerical computation:

$$\int_0^1 x^{-1/2}e^x dx = \left[2x^{1/2}e^x\right]_0^1 - 2\int_0^1 x^{1/2}e^x dx$$

The latter integral has continuous integrand, though singularity in derivative. Repeated integration by parts can be used to get smoother integrand.

We can put the singularity into a weight function and use Gaussian integration on $n+1$ points, which computes $\int_a^b w(x)f(x)dx$ exactly if $f \in \mathcal{P}_{2n+1}$, cf., Example 89.

Integral on infinite interval can be treated by a substitution: for example, an increasing function $w : (0,1) \to (1,+\infty)$, such as

$$w(t) = \frac{1}{(1-t)^p}, \quad p > 0,$$

can be used to convert an improper integral into an integral on a bounded interval

$$\int_0^{+\infty} f(x)dx = \int_0^1 \underbrace{w'(t)f(w(t))}_{g(t)}dt$$

Integration of a function on a bounded interval can be converted into integration of a periodic functions by a substitution as follows. Let w:$[0,2\pi] \mapsto [0,1]$,and

$$w^{(j)}(0) = w^{(j)}(2\pi) = 0, \quad j = 1,\dots,p$$

then

$$\int_0^1 f(x)dx = \int_0^{2\pi} \underbrace{w'(t)f(w(t))}_{g(t)}\, dt$$

where g and some of its derivatives vanish at 0 and 2π.We then can extend g to a periodic function on $\mathbb{R}$ and apply the rectangle rule to the transformed integral, which gives a quadrature rule for the original integral. One examples of functions w is

$$w_p(t) = \frac{t^p}{t^p + (2\pi - t)^p}.$$

See [Kre98, DR84] for more.

Sometimes an integrand can be decomposed into $f = f_1 + f_2$, where f_1 can be integrate analytically, and f_2 is smooth and amenable to being integrated numerically; for example,

$$\int_0^1 \underbrace{x^{-1/2}e^x}_{f} dx = \int_0^1 \underbrace{x^{-1/2}}_{f_1} dx + \int_0^1 \underbrace{x^{-1/2}\left(e^x - 1\right)}_{f_2} dx$$

For $x \approx 0$, one should evaluate the latter integrand using Taylor expansion, $e^x - 1 = x + \frac{x^2}{2} + \ldots$, to avoid loss of significance due to subtracting two close numbers (Sec. 3.1).

10.9 Multidimensional quadrature

Quadrature rules can be nested: for example, by applying the 2 point Gauss rule first to the outer integral, then to the inner integral,

$$\begin{aligned}
\int_{-1}^1 \int_{-1}^1 f(x,y)dxdy &= \int_{-1}^1 \left(\int_{-1}^1 f(x,y)dx \right) dy \\
&\approx \int_{-1}^1 f\left(x, \frac{-1}{\sqrt{3}}\right) dx + \int_{-1}^1 f\left(x, \frac{1}{\sqrt{3}}\right) dx \\
&\approx f\left(\frac{-1}{\sqrt{3}}, \frac{-1}{\sqrt{3}}\right) + f\left(\frac{-1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}, \frac{-1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right),
\end{aligned}$$

Clearly, this *product rule* is exact for all functions f that are polynomials of order at most 3 in each variable:

$$f(x,y) = \sum_{0 \le i,j \le 3} a_{ij} x^i y^j$$

Integrals on a more general domain K can be computed by substitution: if the domain K is an image of $\hat{K} = (-1,1) \times (-1,1)$:

$$\Phi : \hat{K} \to K, \quad \Phi(u,v) = \begin{bmatrix} \Phi_x(u,v) \\ \Phi_y(u,v) \end{bmatrix}, \quad \Phi \text{ is one-to-one,}$$

$$\iint_K f(x_1,x_2)dxdy = \iint_{\hat{K}} \underbrace{f(\Phi(u,v))\left|J_\Phi(u,v)\right|}_{g(u,v)} dudv$$

where $J_{\Phi(u,v)}$ is the Jacobian,

$$J_\Phi(u,v) = \det \begin{bmatrix} \frac{\partial}{\partial u}\Phi_x(u,v) & \frac{\partial}{\partial v}\Phi_x(u,v) \\ \frac{\partial}{\partial u}\Phi_y(u,v) & \frac{\partial}{\partial v}\Phi_y(u,v) \end{bmatrix}.$$

The numerical quadrature rule is applied to the gransformed function g on the *reference domain* $\hat{K}$.

On a triangle T with vertices A, B, C,one can show that the formula

$$\iint_T f(x,y)dxdy \approx \frac{\text{area}(T)}{3}\left(f\left(\frac{A+B}{2}\right)+f\left(\frac{A+C}{2}\right)+f\left(\frac{B+C}{2}\right)\right)$$

is exact for all quadratic polynomials

$$f(x,y)=\sum_{0\le i+j\le 2} a_{ij}x^i y^j.$$

Similar quadrature rules and their error estimates exist in higher dimension than 2. They have been recently in use in in the Finite Element method for solving partial differential equations. Similary as in the Finite Element method, one can compute the integral over a more general region by splitting the region into a union of non-overlapping rectangles or triangles, and adding up the integrals. For more on the quadrature rules used in the Finite Element method, see [Hug00].

Integrals in high dimensions are not tractable by such tensor product schemes: for example, m point composite Simpson's rule in $\mathbb{R}^k$ has accuracy $O(m^4)$ but require evaluation of f on $n=m^k$ points. Instead, such integrals are computed by methods such as Monte-Carlo, evaluating the function at random points as follows.

Let X be a random variable with uniform distribution on a bounded domain Ω in $\mathbb{R}^k$. To integrate a function f on Ω, we consider the relation of the integral and the expected value of the random variable $f(X)$:

$$I=\int_\Omega f(x)dx = E(f(X))\,\text{area}(\Omega).$$

If we now generate random points (with uniform distribution) $X_i \in \Omega$, the sums

$$I_n=\frac{\text{area}(\Omega)}{n}\sum_{i=1}^{n} f(X_i)$$

will approximate the integral I in the sense that the random variable $\sqrt{n}(I_n - I)$ has normal distribution $N(0,\sigma^2)$, in the limit for large n, where

$$\sigma^2=\int_\Omega f^2dx - I^2.$$

For example, this means that the probability of $|I_n - I| > 3\frac{\sigma}{\sqrt{n}}$is less than 1%.Practically, this means that the Monte Carlo integration converges as $1/\sqrt{n}$. If the the area of Ω is not known, the function f can be extended by zero to a function on a larger domain of a simple shape, such as the product of intervals. For more on Monte Carlo methods and quadrature in high dimension, see [**?**].

Chapter 11

Numerical solution of ordinary differential equations

11.1 Initial value problem for first order systems

We consider initial value problems for first order systems of differential equations,

$$u'(t) = f(t, u(t)), \quad u(t_0) = u_0.$$

The dependent variable $u(t)$ is a vector function in $\mathbb{R}^n$ of the scalar variable t,

$$u(t) = \begin{bmatrix} u_1(t) \\ \vdots \\ u_n(t) \end{bmatrix}.$$

The function f is a function of $n+1$ variables, written as $(t, x) \in \mathbb{R} \times \mathbb{R}^n$.A solution of the initial value problem in an interval $I \ni t_0$ is a function u that has derivative u' in I, satisfies the differential equation $u'(t) = f(t, u(t))$ in I, and satisfies the inital condition $u(t_0) = u_0$.

An initial value problem of arbitrary order p,

$$v^{(p)} = F(t, v, v', \ldots, v^{(p-1)}), \quad v(t_0) = v_0^0, \ldots, v^{(n-1)}(t_0) = v_0^{n-1},$$

can be written as (11.1), by introducing additional variables for the derivatives. Let

$$u_1 = v,\ u_2 = v',\ u_3 = v'', \ldots, u_n = v^{(n-1)};$$

then (11.1) becomes

$$\frac{d}{dt}\begin{bmatrix} u_1' \\ u_2' \\ \vdots \\ u_{n-1}' \\ u_n' \end{bmatrix} = \begin{bmatrix} u_2 \\ u_3 \\ \vdots \\ u_n \\ F(t, u_1, \ldots, u_n) \end{bmatrix}, \quad \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_{n-1} \\ u_n \end{bmatrix}(t_0) = \begin{bmatrix} v_0^0 \\ v_0^1 \\ \vdots \\ v_0^{n-2} \\ v_0^{n-1} \end{bmatrix},$$

which is of the form (11.1). This reduction of a high order initial value problem to a first order system is useful for studying properties of the initial value problem, such as existence and uniqueness of the solution, as well as for solving the problem numerically - most software for initial value problems is written for first order systems, and, in principle, one would need to develop only methods for first order systems. However, special methods for the second order systems $u'' = f(t, u, u')$ are also used.

Review of Banach contraction theorem and estimate of contraction number from derivatives

See Sec. 15.2 and 4.4.

11.2 Existence and uniqueness of solution, Lipschitz condition

Here, we denote vectors in $\mathbb{R}^{n+1}$ as (t, x), with $t\epsilon\mathbb{R}$ and $x\epsilon\mathbb{R}^n$. For two points $C, D \in \mathbb{R}^n$, inequalities are understood componentwise

$$C \leq D \Longleftrightarrow \forall i : C_i \leq D_i,$$

etc., and denote intervals by

$$\begin{aligned} [C, D] &= \{x \in \mathbb{R}^n : C \leq x \leq D\} \\ (C, D) &= \{x \in \mathbb{R}^n : C < x < D\} \end{aligned}$$

Theorem 94 *Let $f : R = [c, d] \times [C, D] \subset \mathbb{R}^{n+1} \to \mathbb{R}^n$ be continuous on the region R and satisfy the* Lipschitz condition *in the second variable,*

$$\forall t \in (c, d), x, y \in [C, D] : \|f(t, x) - f(t, y)\|_\infty \leq L \|x - y\|_\infty$$

with some constant L, and let $(t_0, u_0) \in (c, d) \times (C, D)$. Then there exists a constant $a > 0$ such that the initial value problem

$$u' = f(t, u), \quad u(t_0) = u_0,$$

has unique solution on $(t_0 - a, t_0 + a)$.

Proof. From the fundamental theorem of calculus, the initial value problem (**??**) is equivalent to the integral equation

$$u(t) = u_0 + \int_{t_0}^{t} f(\tau, u(\tau))d\tau.$$

This allows us to rewrite the initial value problem as a fixed point problem $u = Au$, where the operator A is defined by

$$A : u \mapsto v, \quad v(t) = u_0 + \int_{t_0}^{t} f(\tau, u(\tau))d\tau.$$

We need to find a Banach space V and a closed set $B \subset V$ such that A maps B into itself and is a contraction on B. Let u, v be continuous on $[c, d]$ and such that

$$u(\tau), v(\tau) \in [C, D] \quad \forall \tau \in [c, d].$$

From the Lipschitz condition (**??**), if ,

$$\begin{aligned} \|(Au - Av)(t)\|_\infty &= \left\| \int_{t_0}^{t} f(\tau, u(\tau)) - f(\tau, v(\tau))d\tau \right\|_\infty \\ &\leq \int_{t_0}^{t} \|f(\tau, u(\tau)) - f(\tau, v(\tau))\|_\infty d\tau \\ &\leq aL \max_{c \leq \tau \leq d} \|u(\tau) - v(\tau)\|_\infty . \end{aligned}$$

So, we choose the space $V = (C[t_0 - a, t_0 + a])^n$,which is the space of all $\mathbb{R}^n$ valued continuous functions on $[t_0 - a, t_0 + a]$, equipped with the norm

$$\|u\|_V = \max_{c \leq \tau \leq d} \|u(\tau)\|_\infty ,$$

which gives the contraction property

$$\|Au - Av\|_V \le c\,\|u - v\|_V\,, \quad c = aL < 1,$$

if a is small enough. For the values of Au, we have the estimate

$$\begin{aligned} \|(Au)(t) - u_0\|_\infty &= \left\| \int_{t_0}^{t} f(\tau, u(\tau)) d\tau \right\|_\infty \\ &\le aM, \quad M = \max_{(t,x)\in[c,d]\times[C,D]} \|f(\tau, x)\|_\infty\,. \end{aligned}$$

(Because the function $\|f(\tau, x)\|_\infty$ is continuous on the closed and bounded set $[c, d] \times [C, D]$, it attains maximum.) This means that $(Au)(t) \in [u_0 - aM, u_0 + aM]$. So, if we choose $a > 0$ small enough so that also

$$[u_0 - aM, u_0 + aM] \subset [C, D],$$

we have

$$\forall u \in B : Au \in B,$$

where

$$B = \{u \in V : \forall t \in [t_0 - a, t_0 + a] : u_0 - aM \le u(t) \le u_0 + aM\}\,.$$

If u^k is a sequence of functions from B and $\left\|u^k - u\right\|_V \to 0$, $k \to \infty$, then, by passing to the limit for each t and each coordinate, we have $u \in B$, hence B is closed. Hence, from the Banach contraction theorem, the equation $u = Au$ has a unique solution in B. ■

Remark 95 *Because a continuous function attains its maximum on a closed and bounded set, the Lipschitz condition will be satisfied as soon as all partial derivatives $\frac{\partial f(t,y)}{\partial y_i}$ are continuous on $[c, d] \times [C, D]$. However,*

Example 96 *Function $f(t, x) = x^2$ does not satisfy the Lipschitz condition (**??**) in any rectangle $[c, d] \times \mathbb{R}$; in general, the region R needs to be bounded. However,*

Remark 97 *Once the solution is known to exist and be unique in the interval $[t_0 - a,\, t_0 + a]$, it can be continued further by picking the point $t_0 - a$ or $t_0 + a$ to play the role of t_0. Because the value of a is determined only from the conditions*

$$aL < 1, \quad [u_0 - Ma, u_0 + Ma] \subset R,$$

the solution can be continued until it eventually reaches the boundary of R.This can be done for a general region R, not just a rectangle.

The importance of the Lipschitz condition is that it determines the sensitivity of the solution to the initial condition. It gives a bound on how far apart can two solutions with close initial values get

Theorem 98 *Let u, v satisfy*

$$u' = f(t, u), \quad v' = f(t, v)$$

in an interval (c, d), and f satisfy the Lipschiz condition

$$\|f(t, x) - f(t, y)\|_\infty \le L\,\|x - y\|_\infty$$

Then for any t_0, $t \in (c, d)$,

$$\|u(t) - v(t)\|_\infty \le \|u(t_0) - v(t_0)\|_\infty\, e^{|t-t_0|L}$$

Proof. Let $t > t_0$, and $k > 0$ be an integer, and let $z = u - v$. We then have from the differential equation and the Lipschitz condition

$$\|z'(\tau)\|_\infty \leq L \|z(\tau)\|_\infty ,$$

for all $\tau \in (t_0, t_1)$. Define

$$\begin{aligned} t_i &= t_0 + ih, \quad h = \frac{t - t_0}{k}, \\ m_i &= \max_{t_{i-1} \leq \tau \leq t_i} \|z(\tau)\|_\infty , \quad m_0 = \|z(t_0)\|_\infty . \end{aligned}$$

Using the mean value theorem in each coordinate separately, we have for $\tau \in (t_{i-1}, t_i]$,

$$z_j(\tau) = z_j(t_{i-1}) + z_j'(\xi_j)(\tau - t_{i-1}), \quad \xi_j \in (t_{i-1}, \tau),$$

which, taking the maximum over all j and all $\tau \in [t_{i-1}, t_i]$, gives

$$m_i \leq m_{i-1} + hLm_i,$$

so

$$\|z(t)\|_\infty \leq m_k \leq \frac{m_0}{\left(1 - \frac{t-t_0}{k} L\right)^k}.$$

Taking the limit for $k \to \infty$ and using $\left(1 - \frac{a}{k}\right)^k \to e^{-a}$, we get

$$\|z(t)\|_\infty \leq \|z(t_0)\|_\infty \, e^{(t-t_0)L}.$$

The case $t < t_0$ is similar, or one can use the substitution $t - t_0 \to t + t_0$. ■

Obviously, we can expect trouble when solving initial values problem with large Lipschitz constant L.

11.3 Euler method and its variants

Let

$$t_k = t_0 + kh, \quad h > 0.$$

We are looking for approximate values of the solution of the inital value problem at points t_k,

$$u_k \approx u(t_k).$$

Replacing in

$$u(t_{k+1}) = u(t_k) + \int_{t_k}^{t_{k+1}} f(\tau, u(\tau)) d\tau$$

the integral by

$$\int_{t_k}^{t_{k+1}} f(\tau, u(\tau)) d\tau \approx h f(t_k, u(t_k))$$

and further replacing the values of u by the approximate values u_k,we get the *Euler method*

$$u_{k+1} = u_k + h f(t_k, u_k).$$

The Euler method can be also thought of as replacing the derivative by one-sided forward difference,

$$\frac{u_{k+1} - u_k}{h} = f(t_k, u_k).$$

Euler method approximates the solution curve by a broken numerical solution line made of tangents: in Fig. 11.1 left, you can see that the slope of the line in $[t_k, t_{k+1}]$.is given by the derivative of the exact solution passing through the point (t_k, u_k).

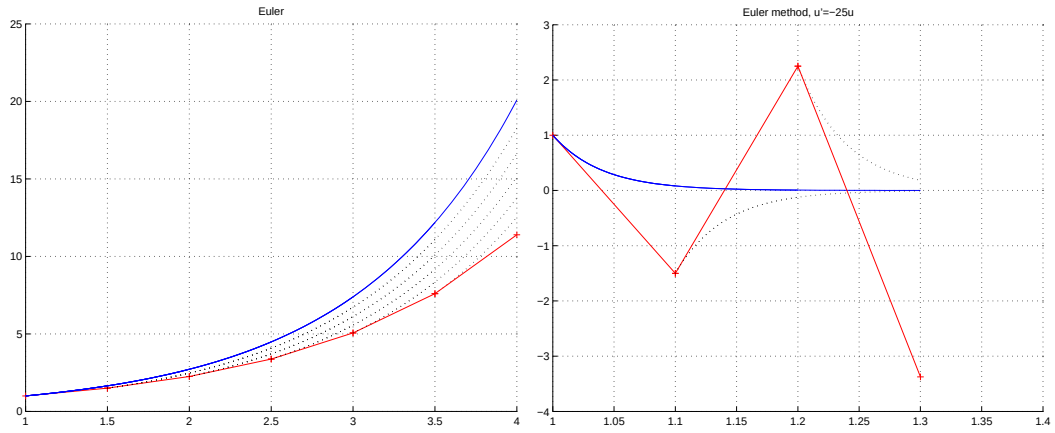


Figure 11.1: Euler method for $u' = u$, $u(1) = 1$, and $u' = -25u$, $u(1) = 1$

The initial value problem

$$u' = au, \quad u(t_0) = u_0$$

with $a < 0$ has the solution $u(t) = u_0 e^{at} \to 0$ as $t \to \infty$. So, it is desirable that the approximate solutions have the same property: $u_k \to 0$ as $k \to \infty$.The Euler method fails for large negative a (cf., Fig. 11.1 right). The reason is that for this model problem,

$$u_{k+1} = u_k + hau_k = u_k(1 + ah),$$

so

$$u_k = u_0(1 + ah)^k \to 0, \ k \to \infty \iff |1 - ah| < 1.$$

In this case, we are forced to use small $h < 1/a$ even if we are approximating a very smooth solution, essentially a constrant for large t.Another example of this kind the equation

$$u' = a(u(t) - \sin t), \quad u(t_0) = u_0,$$

with large negative a, which, for large t, has smooth solution $u(t) \approx \sin t$, and yet one has to use small h dictated by the condition $|1 - ah| < 1$.

More generally, consider the n-dimensional problem

$$u' = Au, \quad u(t_0) = u_0.$$

For simplicity, assume that A is diagonalizable; then there is a basis of $\mathbb{R}^n$ consisting of eigenvectors of A :

$$AU_j = \lambda_j U_j$$

and the solution of the initial value problem is found by decomposing the initial value in the basis of the eigenvectors, solving for each component separately, and adding the results:

$$u_0 = \sum_{j=1}^{n} c_j U_j, \quad u = \sum_{j=1}^{n} c_j U_j e^{\lambda_j (t - t_0)}.$$

But, in general, the eigenvalues of A may be complex, so we are forced to consider complex eigenvectors, and the complex exponential:

$$e^z = e^{\operatorname{Re} z}(\cos \operatorname{Im} z + i \sin \operatorname{Im} z).$$

Clearly, $u \to 0$ as $t \to \infty \iff \operatorname{Re} \lambda_j < 0$ for all $j = 1, \ldots, n$. Ordinary differential equations where some of the eigenvalues of A, or of the matrix $\frac{\partial f}{\partial x}$of the derivatives of f in the nonlinear case, have a large negative real part, are called *stiff*.

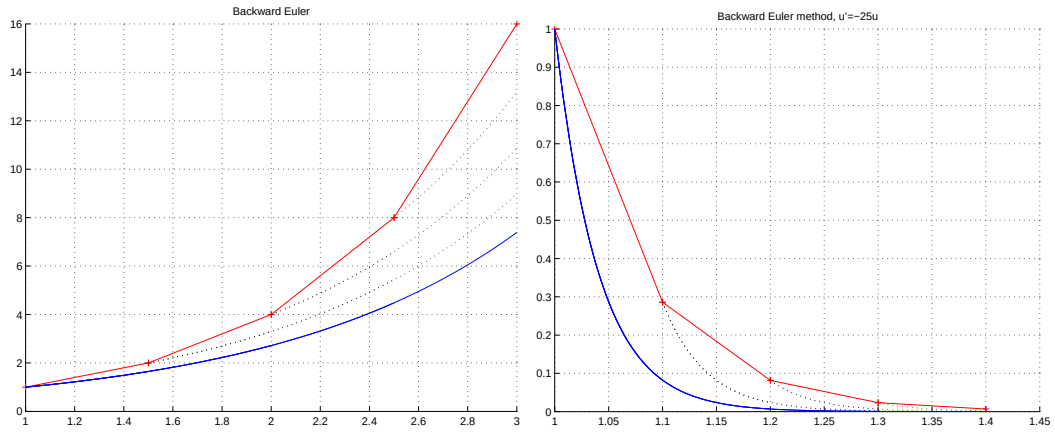


Figure 11.2: Backward Euler method for $u' = u$, $u(1) = 1$, and $u' = -25u$, $u(1) = 1$

Definition 99 *The region of stability of a method for initial value problems is the set*

$$R = \left\{ ha \in \mathbb{C} : \operatorname{Re} ha < 0, \quad \begin{array}{c} \textit{the numerical solution } u_k \textit{ of } u' = au,\ u_0 = 1, \\ \textit{satisfies } u_k \to 0 \textit{ as } k \to \infty \end{array} \right\}$$

The region of stability of the Euler method is the circle

$$R = \{ha \in \mathbb{C} : |1 - ha| < 1\}.$$

The *Backward Euler method* is obtained by using the approximation of the integral

$$\int_{t_k}^{t_{k+1}} f(\tau, u(\tau))d\tau \approx hf(t_{k+1}, u(t_{k+1})),$$

which gives

$$u_{k+1} = u_k + hf(t_{k+1}, u_{k+1}).$$

See Fig. 11.2. This is an *implicit method* - we have to solve for the value of u_{k+1} at every time step, for example, by Newton's method. In contrast, methods like the Euler method, which have u_{k+1} only on the left-hand side, are *explicit methods*. For the problem $u' = au$, we have

$$\begin{aligned} u_{k+1} &= u_k + hau_{k+1} \\ u_k &= \frac{u_0}{(1-ha)^k} \to 0 \iff |1 - ha| > 1 \end{aligned}$$

The last condition is satisfied when ah is outside of the circle in $\mathbb{C}$ with center at 1 and radius 1, so the region of stability of the Backward Euler method is

$$R = \{ah : \operatorname{Re} ah < 0\}.$$

Methods with this region is stability are called absolutely stable - there are no restrictions on the step size.

By averaging the Euler method and the Backward Euler method we obtain the *Trapezoidal method* (or *Implicit Euler method, Crank-Nicholson method*)

$$u_{k+1} = u_k + \frac{h}{2}\left(f(t_k, u_k) + f(t_{k+1}, u_{k+1})\right),$$

which is also obtained by approximating the integral by the more accurate trapezoidal rule,

$$\int_{t_k}^{t_{k+1}} f(\tau, u(\tau))d\tau \approx \frac{h}{2}\left(f(t_k, u(t_k) + f(t_{k+1}, u(t_{k+1}))\right)$$

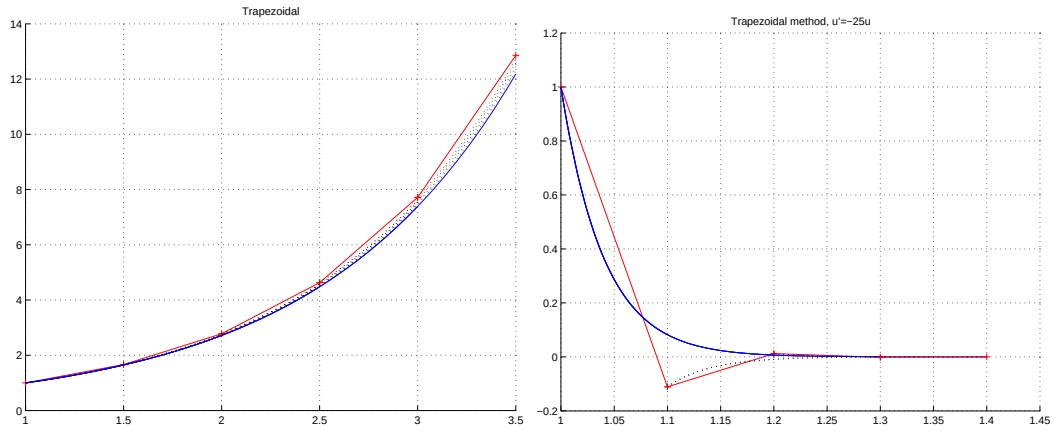


Figure 11.3: Trapezoidal method for $u' = u$, $u(1) = 1$, and $u' = -25u$, $u(1) = 1$

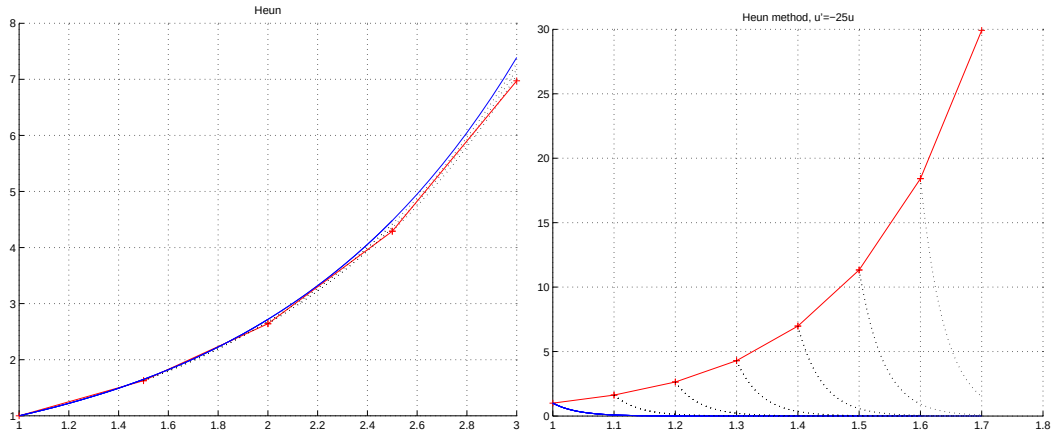


Figure 11.4: Heun method for $u' = u$, $u(1) = 1$, and $u' = -25u$, $u(1) = 1$

and replacing again $u(t_j)$ by u_j. See Fig. 11.3. To obtain the region of stability, we have for the problem $u' = au$,

$$\begin{aligned}
u_{k+1} &= u_k + \frac{h}{2}\left(au_k + au_{k+1}\right) \\
u_k &= \left(\frac{1+\frac{ah}{2}}{1-\frac{ah}{2}}\right)^k u_k \to 0, \quad k \to \infty \\
&\iff \left|1+\frac{ah}{2}\right|^2 < \left|1-\frac{ah}{2}\right|^2 \\
&\iff \left(1+\frac{\operatorname{Re} ah}{2}\right)^2 + \left(\frac{\operatorname{Im} ah}{2}\right)^2 < \left(1-\frac{\operatorname{Re} ah}{2}\right)^2 + \left(\frac{\operatorname{Im} ah}{2}\right)^2 \\
&\iff \operatorname{Re} ah < -\operatorname{Re} ah \iff \operatorname{Re} ah < 0
\end{aligned}$$

so the Trapezoidal method is absolutely stable,

$$R = \{ah : \operatorname{Re} ah < 0\}.$$

To solve the equation (or systems of equations in the n-dimensional case) in the Trapezoidal

method for u_{k+1}, we can use fixed point iterations on the equation

$$u_{k+1} = F(u_{k+1}), \quad F(x) = u_k + \frac{h}{2}\left(f(t_k, u_k) + f(t_{k+1}, x)\right).$$

Assuming that f satisfies the Lipschitz condition

$$\|f(t,x) - f(t,y)\|_\infty \leq L \|x - y\|_\infty,$$

we have

$$\|F(x) - F(y)\|_\infty \leq \frac{Lh}{2} \|x - y\|_\infty$$

so F will be a contraction and the iterations $v \longleftarrow F(v)$ will converge when $Lh < 2$. This will be satisfied for sufficiently small h, and, the smaller h, the faster the iteration. A good starting point for the iterations is the result of one step of the Euler method. Using one fixed-point iteration (which we will show later is sufficient to retain the order of the Trapezoidal method) gives the *Heun method* (or *Improved Euler method*)

$$\begin{aligned} u^p_{k+1} &= u_k + hf(t_k, u_k), \\ u_{k+1} &= u_k + \frac{h}{2}\left(f(t_k, u_k) + f(t_{k+1}, u^p_{k+1})\right). \end{aligned}$$

See Fig. 11.4. This method is an example of a *predictor-corrector method* − first an explicit rule of a lower order of accuracy is used in the prediction step, which gives an initial approximation for iterations on a more accurate implicit rule. Here, the Euler method is the predictor and the Trapezoidal rule is the corrector.

Exercise 100 *Show that the Heun method is not absolutely stable.*

To implement the Trapezoidal rule in the nonlinear case, one needs to use a better iteration, such as the Newton's method, for the corrector step. This requires the user to give also the matrix of the derivatives $\frac{\partial f}{\partial x}$. Again, the Euler method is used as the predictor. Such methods are most reliable and widely used for stiff problems. Alternatively, the derivatives can be estimated by finite differences, which of course creates new questions, such as how to determine the proper stepsize for the differences.

11.4 Local error analysis

Local error analysis The error of the numerical scheme in one step is defined as follows.

Definition 101 *Let u_k be numerical approximation of the solution of the IVP*

$$u' = f(t,u), \quad u(t_0) = u_0$$

at t_k, and for each k, let $\tilde{u}_k$ denote the solution of the IVP

$$\tilde{u}'_k = f(t, \tilde{u}_k), \quad \tilde{u}_k(t_k) = u_k$$

The local truncation error *is defined by*

$$\tau_k = u_{k+1} - \tilde{u}_k(t_{k+1}).$$

A method is of order p if $\tau_k = O(h^{p+1})$, where $h = t_{k+1} - t_k$.

The reason for the exponent $p+1$ is that error from multiple steps accumulate, and a solution over a fixed interval will involve $O(1/h)$ steps, hence the error of the solution over the whole interval will be $O(h^p)$. See Sec. 11.6 below.

The methods studied so far were one point methods:

Definition 102 One-point method *for the solution of IVP is a method of the form* $u_{k+1} = F(t_k, u_k)$.

To estimate the local truncation error and determine the order of the method, we expand both $\tilde{u}_k$ and u_{k+1}in the powers of h and compare the terms. We assume that all needed derivatives of f and of the solutions $\tilde{u}_k$ are bounded in the region of interest.

For the Euler method, we have

$$\begin{aligned} u_{k+1} &= u_k + hf(t_k, u_k), \\ \tilde{u}_k(t_{k+1}) &= \underbrace{\tilde{u}_k(t_k)}_{u_k} + h\,\underbrace{\tilde{u}_k'(t_k)}_{f(t_k,u_k)} + O(h^2), \end{aligned}$$

which gives $\tau_k = O(h^2)$; hence the Euler method is of order 1.

For the Backward Euler method, we need to use zero order Taylor expansion of f,

$$f(t_{k+1}, u_{k+1}) = f(t_k, u_k) + O(h),$$

(because $t_{k+1} - t_k = h$ and $u_{k+1} - u_k = hf(t_k, u_k) = O(h)$), which gives

$$\begin{aligned} u_{k+1} &= u_k + hf(t_{k+1}, u_{k+1}) = u_k + h\left(f(t_k, u_k) + O(h)\right) \\ &= u_k + hf(t_k, u_k) + O(h^2), \\ \tilde{u}_k(t_{k+1}) &= \underbrace{\tilde{u}_k(t_k)}_{u_k} + h\,\underbrace{\tilde{u}_k'(t_k)}_{f(t_k,u_k)} + O(h^2), \end{aligned}$$

and comparing the terms we get again $\tau_k = O(h^2)$; hence the Backward Euler method is also of order 1.

For the Trapezoidal method, we use the Taylor expansion of f of order 1, for increments $\zeta = O(h)$, $\xi = O(h)$,

$$f(t_k + \zeta, u_k + \xi) = f(t_k, u_k) + f_t\left(t_k, u_k\right)\zeta + f_x\left(t_k, u_k\right)\xi + O(h^2),$$

where $\xi \in \mathbb{R}^n$ and f_x is the matrix of the partial derivatives of f with respect to the last n arguments,

$$f_x = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}.$$

Then

$$u_{k+1} = u_k + \frac{h}{2}\left(f(t_k, u_k) + f(t_{k+1}, u_{k+1})\right).$$

To simplify notation, write just f for (t_k, u_k), and similarly f_t and f_x ; then

$$\begin{aligned} u_{k+1} - u_k &= \frac{h}{2}\left(f + f + hf_t + f_x(u_{k+1} - u_k) + O(h^2)\right) \\ u_{k+1} - u_k &= \left(I - \frac{h}{2}f_x\right)^{-1}\left(hf + \frac{h^2}{2}f_t + O(h^2)\right). \end{aligned}$$

From the Neumann series (Sec. 7.1.2), the inverse matrix has the expansion,

$$\left(I - \frac{h}{2}f_x\right)^{-1} = I + \frac{h}{2}f_x + O(h^2),$$

which gives

$$\begin{aligned} u_{k+1} &= u_k + \left(I + \frac{h}{2}f_x + O(h^2)\right)\left(hf + \frac{h^2}{2}f_t + O(h^2)\right) \\ &= u_k + hf + \frac{h}{2}^2\left(f_t + f_x f\right) + O(h^3) \end{aligned}$$

To get the Taylor expansion of $\tilde{u}_k(t_{k+1})$, use first that $\tilde{u}_k'(t) = f(t, \tilde{u}_k(t))$ and the chain rule to get

$$\tilde{u}_k''(t_k) = \frac{d}{dt}(f(t, \tilde{u}_k(t))\Big|_{t=t_k} = f_t + f_x \tilde{u}_k'(t_k) = f_t + f_x f,$$

hence

$$\begin{aligned} \tilde{u}_k(t_{k+1}) &= \tilde{u}_k(t_k) + h\tilde{u}_k'(t_k) + \frac{h^2}{2}\tilde{u}_k'' + O(h^3) \\ &= u_k + hf + \frac{h}{2}^2 (f_t + f_x f) + O(h^3). \end{aligned}$$

Consequently, $\tau_k = O(h^3)$, and so the Trapezoidal method is of order 2.

To analyze the local truncation error of the Heun method, it is useful to note that the method consists of *nested evaluations of* f and consider the more general form

$$u_{k+1} = u_k + a_1 hf(t_k, u_k) + a_2 hf(t_k + c_2 h, f(u_k + d_{21} hf(t_k, u_k)).$$

The Heun method is obtained for

$$a_1 = a_2 = \frac{1}{2},\ c_2 = d_{21} = 1..$$

Again, drop the arguments (t_k, u_k) when appropriate. Compute Taylor expansion of u_{k+1}using the Taylor expansion of f of order 1 and the chain rule:

$$\begin{aligned} u_{k+1} &= u_k + a_1 hf + a_2 hf(t_k + c_2 h, f(u_k + d_{21} hf(t_k, u_k)) + O(h^3) \\ &= u_k + a_1 hf + a_2 h\left(f + f_t c_2 h + f_x d_{21} hf + O(h^2)\right) + O(h^3) \\ &= u_k + (a_1 + a_2)\, hf + a_2 c_2 h^2 f_t + a_2 d_{21} h^2 f_x f + O(h^3). \end{aligned}$$

Comparing with the Taylor expansion (**??**),

$$\tilde{u}_k(t_{k+1}) = u_k + hf + \frac{h^2}{2}(f_t + f_x f) + O(h^3),$$

we see that the method will be of order 2 if the coefficients satisfy the equations

$$a_1 + a_2 = 1, \quad a_2 c_2 = \frac{1}{2}, \quad a_2 d_{21} = \frac{1}{2}.$$

Indeed, the coefficients of the Heun method satisfy these equations. Another solution is

$$a_1 = 0, \quad a_2 = 1, \quad c_2 = d_{21} = \frac{1}{2},$$

which gives the *Improved Euler method*

$$u_{k+1} = u_k + hf\left(t_k + \frac{1}{2}h, u_k + \frac{h}{2}f(t_k, u_k)\right),$$

which is of order 2. Note that this method can be understood as doing one step of Euler method with half the time step, and then using the result for central difference approximation of the derivative, which is of order 2.

11.5 Runge-Kutta methods

The Heun method and the Improved Euler method use nested evaluation of f to match the first three terms the Taylor series for $\tilde{u}_k(t_{k+1})$. They are special cases Runge-Kutta methods, which are

of the general form

$$\begin{aligned} u_{k+1} &= u_k + \sum_{i=1}^{m} a_i F_i \\ F_i &= hf\left(t + c_i h, x + \sum_{j=1}^{i-1} d_{ij} F_j\right), \quad i = 1, \ldots, m \end{aligned}$$

The original, and still widely used, Runge-Kutta method of order 4 has $m = 4$ evaluations, and the coefficients

$$a = \begin{bmatrix} \frac{1}{6} \\ \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{6} \end{bmatrix}, \quad c = \begin{bmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \\ 1 \end{bmatrix}, \quad d = \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The coefficients of Runge-Kutta methods are solutions of system of nonlinear algebraic equations arizing from matching the Taylor expansion of the solution. Solutions giving the order $p = m$ do not always exist; in general, the order that can be achieved is less than the number of evaluations:

Number of evaluations m	1	2	3	4	5	6	7	8
Maximum order p	1	2	3	4	4	5	6	6

Runge-Kutta method of order 5 and 6 evaluations of f is given by

$$a = \begin{bmatrix} \frac{16}{135} \\ 0 \\ \frac{6656}{12825} \\ \frac{28561}{56430} \\ -\frac{9}{50} \\ \frac{2}{55} \end{bmatrix}, \quad c = \begin{bmatrix} 0 \\ \frac{1}{4} \\ \frac{3}{8} \\ \frac{12}{13} \\ 1 \\ \frac{1}{2} \end{bmatrix}, \quad d = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{32} & \frac{9}{32} & 0 & 0 & 0 & 0 \\ \frac{1932}{2197} & -\frac{7200}{2197} & \frac{7296}{2197} & 0 & 0 & 0 \\ \frac{439}{216} & -8 & \frac{3680}{513} & -\frac{845}{4104} & 0 & 0 \\ -\frac{8}{27} & 2 & -\frac{3544}{2565} & \frac{1859}{4104} & -\frac{11}{40} & 0 \end{bmatrix}.$$

11.6 Global error analysis of one-point methods

Global error Recall from Definition 101 that the local truncation error at t_k is defined by

$$\tau_k = u_{k+1} - \tilde{u}_k(t_{k+1}).$$

where $\tilde{u}_k' = f(t, \tilde{u}_k)$, $\tilde{u}_k(t_k) = u_k$.Using the estimate of how far apart two solutions of differential equation can grow (Theorem 98),

$$\|u(t) - v(t)\|_\infty \leq \|u(t_0) - v(t_0)\|_\infty e^{|t-t_0|L}$$

where L is the Lipschitz constant, we have for the difference of the numerical and the exact solution

$$\begin{aligned} \|u_n - u(t_n)\|_\infty &= \|\tau_{n-1}\|_\infty + e^{hL} \|\tau_{n-2}\|_\infty + \ldots + e^{hL(n-1)} \|\tau_0\|_\infty \\ &\leq \frac{e^{hLn} - 1}{e^{hL} - 1} \max\left\{\|\tau_0\|_\infty, \ldots, \|\tau_{n-1}\|_\infty\right\}. \end{aligned}$$

Theorem 103 *Suppose the Lipschitz condition holds with constrant L, and $\|\tau_k\|_\infty \leq \tau$ for all k. Then*

$$\|u_n - u(t_n)\|_\infty \leq \frac{e^{L(t_n - t_0)} - 1}{e^{hL} - 1}\tau \approx \frac{e^{L(t_n - t_0)} - 1}{hL}\tau.$$

Note that the global error has one less power of h than the local error, and that global error can grow exponentially with the distance from t_0.

Long term behavior All error analysis are only for t in a finite time interval and linearization of the differential equation. For large t, we want the method reproduce asymptotic behavior of the solution. We have seen one such example in stiff problems, where the solution of the model problem satisfies $u(t) \to 0$ as $t \to \infty$,and we have studied which numerical methods exhibit the same behavior. More generally, if the solution has a limit as $t \to \infty$,we would expect the numerical solution converge to have the same limit. For nonlinear systems where there is no limit of the solution as the $t \to \infty$; the question is which more complicated attributes of the behavior of the solution for large t, such as oscillations, cycles and atractors, are reproduced by numerical methods [SH96].

11.7 Multi-point methods

11.7.1 Leapfrog method

Replacing in Euler's method the one-sided difference by the more accurate central difference

$$\frac{u(t+h) - u_{n-1}(t-h)}{2h} = u'(t) + O(h^2)$$

gives the *leapfrog method*

$$u_{n+1} = u_{n-1} + 2hf_n$$

Looks great, without any extra cost (still one evaluation of f per step) we get a method of order two. But numerical examples for the equation

$$u' = -u,\ u(0) = 1$$

λshow that after a while, the iterates start oscillating and the method blows up. Replacing u_0 and u_1 by values of exact solution and decreasing h delays the blowup but eventually the method fails anyway. Why? For the model problem the leapfrog method becomes

$$u_{n+1} = u_{n-1} - 2hu_n$$

To find solutions of this difference equation use assumed form of solution

$$u_n = \lambda^n, \lambda \neq 0$$

substitute in the difference equation to get

$$\lambda^{n+1} = \lambda^{n-1} - 2h\lambda^n$$

and dividing by λ^n get the characteristic equation

$$\lambda^2 + 2\lambda h - 1 = 0$$

which has roots

$$\lambda_{1,2} = \frac{-2h \pm \sqrt{4h^2 + 4}}{2} = \pm 1 - h + O(h^2)$$

Clearly, for arbitrary constants A_1, A_2 the difference solution is satisfied by

$$u_n = A_1\lambda_1^n + A_2\lambda_2^n$$

and $|\lambda_1| > 1$.Even if because of the choice of u_0 and u_1, one may have $A_1 = 0$, rounding errors will excite the component λ_1^n,which will eventually dominate.

11.7.2 Second difference method

Linear difference equations

See Section 15.9.

11.7.3 Analysis of multi-step methods

We consider methods of the general form

$$a_m u_{n+1} + a_{m-1} u_n + \ldots + a_0 u_{n-m+1} = h(b_m f_{n+1} + \ldots + b_0 f_{n-m+1}) \tag{11.1}$$

The model problem $u' = 0$ gives the following concept of stability. In this case, the method reduced to the difference equation (15.3). Because the difference equation needs to admit the constant solution $u_n = 1$, $\lambda = 1$ is always a root of the characteristic equation

$$a_m \lambda^m + \ldots + \lambda a_1 + a_0 = 0.$$

If there is a root with $|\lambda| > 1$, or a multiple root with $|\lambda| > 1$, then the difference equation has a solution that is not bounded. Such methods are obviously not suitable; the unbounded component of the solution will quickly dominate.

Definition 104 *Multi-step method is strongly stable if $\lambda = 1$ is a simple root of the characteristic equation, and all other roots satisfy $|\lambda| < 1$.The method is weakly stable if all roots of the characteristic equation satisfy $|\lambda| \leq 1$.*

For the leapfrog method, we have characteristic equation

$$\lambda^2 - 1 = 0,$$

with roots $\lambda = \pm 1$so the method is not strongly stable, but it is weakly stable. As we have seen with the leapfrog method, one can expect that one problem with weakly stable methods will be that a perturbation of the characteristic equation, caused by right-hand side ofthe form cu,will push the roots of the characteristic equation outside of the unit circle. Also, a multiple root λ with $|\lambda| = 1$ will cause unbounded components of the solution, but these grow not as fast as in the case $|\lambda| > 1$.

Suppose u is an exact solution of the differential equation; substituting into the formula for the multi-point method we get the residual, called the local truncation error

$$R(u) = (a_m u(t_{n+1}) - h b_m u'(t_{n+1})) + \ldots + (a_0 u_{n+1} - h b_0 u'(t_{n+1}))\,.$$

The idea is same as before: the local truncation error is by how much the discrete equations are not satified for the values of an exact solution.

Definition 105 *A multi-step method is of order k if $R(u) = O(h^{k+1})$*

From the Taylor expansion, we have immediately

Theorem 106 *A multipoint method is of order k if and only if $R(u) = 0$ for all $u \in P_k$.*

One can prove

Theorem 107 *If a multi-step method is of order k and it is stable, then the global error on a fixed interval $[t_0, t_0 + T]$ is $O(h^k)$.*

11.7.4 Adams-Bashforth and Adams-Moulton methods

The Adams-Bashforth method of order m is obtained from

$$u(t_{n+1}) = u(t_n) + \int_{t_n}^{t_{n+1}} f(t, u(t))dt,$$

replacing $f(t, u(t))$ by the unique polynomial of order m,defined by the values at $t_n, \ldots, t_{n-m+1}$.It is a special case of multi-step method, of the form

$$u_{n+1} = u_n + h(b_{m-1} f_n + \ldots + b_0 f_{n-m+1}).$$

The Adams-Moulton method of order $m+1$ is obtained by replacing $f(t,u(t))$ by the unique polynomial of order $m+1$,defined by the values at $t_{n+1},\ldots,t_{n-m+1}$,and it is of the form

$$u_{n+1} = u_n + h(c_m f_{n+1} + \ldots + c_0 f_{n-m+1}).$$

The Adams-Bashforth method is explicit and Adams-Moulton is implicit. The coefficients of the method can be obtained by integrating the interpolation polynomial, or, more easily, by the method of undetermined coefficients, requring that $R(u) = 0$ for all polynomials of degree up to m, resp. $m+1$.

A practical algorithm consists of

1. Runge-Kutta method of order $m+1$ to obtain the starting value
2. Adams-Bashforth method as the predictor
3. Adams-Moulton method as the corrector.

As for one-step methods, the order of the method is preserved when the nonlinear equation in the corrector is solved by just one step of the fixed-point iteration. For stiff problems, one has to again solve the corrector equation by other methods, such as Newton's method. An aposteriori estimate of error is obtained from the difference between the results of the predictor and the corrector. Adaptive variang varies the step size to satisfy a given tolerance without unnnecessarily many function evaluations.

11.8 Adaptive methods

Ruge-Kutta_Fehlberg... aposteriori estimate, change of step... comes later.

It is very useful to peruse the source code of ODE solvers in Matlab (at least ode45 and ode112), and to read the help. For more on on numerical solution of ODEs in Matlab, see [SR97].

Chapter 12

Boundary Value Problems

12.1 Shooting methods

12.2 Finite Difference method in 1D

Consider the model problem

$$-u'' = f \text{ in } (a,b),\ u(a) = u(b) = 0 \tag{12.1}$$

Recall approximation of the derivative by finite differences:

$$u''(x) = \frac{u(x+h) - 2u(x) + u(x-h)}{h^2} - \frac{h^2}{12}u^{(4)}(\xi),\ \xi \in (x-h, x+h) \tag{12.2}$$

Define the mesh

$$\begin{aligned} a &= x_0 < x_1 < \ldots < x_{n+1} = b \\ x_i &= a + hi,\ h = \frac{b-a}{n+1} \end{aligned}$$

We seek approximation to the solution u at the nodes x_i,

$$u_i \approx u(x_i)$$

Applying the difference formula on each node, replacing $u(x_i)$ by u_i and writing $f_i = f(x_i)$, we get the finite difference approximation to the boundary value problem (12.1):

$$\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} = f_i,\ i = 1, \ldots, n,\ u_0 = u_{n+1} = 0 \tag{12.3}$$

This is a system of n linear algebraic equations for the n uknowns $u_1, \ldots, u_n$..The boundary values u_0 and u_n are given so they are not considered unknowns; they are included in the difference formula for convenience, to avoid special cases at the boundary. Let

$$A = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & \ddots & & \\ & & & 2 & -1 \\ & & & -1 & 2 \end{bmatrix},\ F = \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix},\ U = \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix},\ U^* = \begin{bmatrix} u(x_1) \\ \vdots \\ u(x_n) \end{bmatrix}$$

We can then write discrete system in matrix form,

$$\frac{1}{h^2}AU = F \tag{12.4}$$

while the vector U^* of the values of the exact solution satisfies

$$\frac{1}{h^2}AU^* = F + R \tag{12.5}$$

where R is the local truncation error. Just as in the methods of ODEs, the local truncation error is obtained as the residual when the values of the exact solution are substituted in the discrete equations. From (12.2),

$$\|R\|_\infty \leq \frac{h^2}{12} \max_{[a,b]} \left| u^{(4)} \right|.$$

But, what does this equation say about the accuracy of the method? Really, nothing until we have answers to the following questions.

1. Does a solution to the continous problem (12.1) exist and is it unique?
2. Does solution to the discrete problem (12.3) exist and is it unique?
3. How big can $U - U^*$ be?

Existence and uniqueness of the solution u is known from the theory of differential equations. For the discrete solution U to exist and be unique we need the existence of the inverse A^{-1}. Then, subtracting (12.5) and (12.4), we get

$$\begin{aligned} \frac{1}{h^2}A\left(U^* - U\right) &= R \\ U^* - U &= h^2 A^{-1} R \\ \|U^* - U\|_\infty &\leq h^2 \left\|A^{-1}\right\|_\infty \|R\|_\infty \end{aligned} \tag{12.6}$$

So, to we answer the last two questions, we need to prove that A^{-1} exists and estimate its norm. First, we calculate the eigenvalues of A. Because $(\sin kt)'' = -k^2 \sin kt$, k integer, by substitution $x = a + t(b-a)$ we obtain eigenvalues $(b-a)^2k^2$ of the boundary value problem:

$$-u'' = (b-a)^2 k^2 u,\ u(a) = u(b) = 0.$$

(By the way, u describes the shape of a vibrating string, and the eigenvalue determines the frequency.) So, we expect the eigenvectors of the discrete problem have a similar form; indeed, if we define

$$v_{kj} = \sin j\frac{k\pi}{n+1},\ V_k = \begin{bmatrix} v_{k1} \\ \vdots \\ v_{kn} \end{bmatrix}$$

(note that $v_0 = v_{n+1} = 0$), we have $AV_k = \lambda_k V_k$, or

$$-v_{k,j-1} + 2v_{k,j} - v_{k,j+1} = \lambda_k v_{k,j},$$

with

$$\lambda_k = 2\left(1 - \cos\frac{k\pi}{n+1}\right)$$

from the trigonometric identity $\sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$, as follows:

$$\begin{aligned} & - \overbrace{\sin\left(j\frac{k\pi}{n+1} - \frac{k}{n+1}\right)}^{\sin j\frac{k\pi}{n+1}\cos\frac{k\pi}{n+1} - \cos j\frac{k\pi}{n+1}\sin\frac{k\pi}{n+1}} + 2\sin\left(j\frac{k\pi}{n+1}\right) - \overbrace{\sin\left(j\frac{k\pi}{n+1} - \frac{k\pi}{n+1}\right)}^{\sin j\frac{k\pi}{n+1}\cos\frac{k\pi}{n+1} + \cos j\frac{k\pi}{n+1}\sin\frac{k\pi}{n+1}} \\ = & \left(2 - 2\cos\frac{k\pi}{n+1}\right)\sin\left(j\frac{k\pi}{n+1}\right). \end{aligned}$$

Because we get n distinct eigenvalues for $k = 01, \ldots, n$,(the values $k = 0$ and $n+1$ do not give an eigenvalue, because $V_k = 0$), and A is of order n, these are all eigenvalues of A.In particular, all eigevalues $\lambda_k > 0$, so A^{-1} exists; in fact, A is symmetric positive definite. The smallest and the largest eigenvalue and the condition number of A are, from the Taylor expansion of the cos function,

$$\begin{aligned}
\lambda_{\min} &= 2\left(1 - \cos\frac{\pi}{n+1}\right) = 2\left(1 - \left(1 - \frac{1}{2}\frac{\pi^2}{(n+1)^2} + \ldots\right)\right) \\
&\approx \frac{\pi^2}{(n+1)^2} \\
\lambda_{\max} &= 2\left(1 - \cos\frac{n\pi}{n+1}\right) \approx 4 \\
\kappa &= \frac{\lambda_{\max}}{\lambda_{\min}} \approx \frac{4(n+1)^2}{\pi^2}
\end{aligned}$$

To estimate the $\left\|A^{-1}\right\|_\infty$, we need first

Lemma 108 *The entries of $A^{-1}$are nonnegative, $A^{-1} \geq 0$.*

Proof. We have $A = 2(I - B)$, where

$$B = \begin{bmatrix} 0 & 1/2 & & \\ 1/2 & \ddots & & \\ & & \ddots & 1/2 \\ & & 1/2 & 0 \end{bmatrix}$$

and the eigenvalues of A and B are related by

$$\lambda_A = 2(1 - \lambda_B)$$

so

$$0 < \lambda_A < 4 \implies |\lambda_B| < 1.$$

So the spectral radius $\rho(B) < 1$, and we have the Neumann expansion

$$A^{-1} = \frac{1}{2}(I - B)^{-1} = \frac{1}{2}\sum_{k=0}^{\infty} B^k \geq 0.$$

(In fact, the entries of A^{-1} are positive.) ■

Theorem 109 *The discrezation error satisfies*

$$\|U^* - U\|_\infty \leq \frac{(b-a)^2 h^2}{96} \|f''\|_\infty$$

Proof. Let

$$C = (c_{ij}) = h^2 A^{-1}.$$

Since $c_{ij} \geq 0$,

$$\|C\|_\infty = \max_i \sum_{j=1}^{n} |c_{ij}| = \max_i \sum_{j=1}^{n} c_{ij} = \|Ce\|_\infty$$

where e is the vector of all ones. To compute Ce, consider the boundary value problem

$$-v'' = 1 \text{ in } (a,b),\ v(a) = v(b) = 0$$

with the solution

$$v(x) = \frac{(x-a)(x-b)}{2}.$$

Since $v^{(4)} = 0$,the local truncation error is zero, and we so with the vector of values $V = (v(x_i))$,

$$\frac{1}{h^2} AV = e \implies Ce = V.$$

which gives

$$h^2 \left\| A^{-1} \right\|_\infty = \|Ce\|_\infty \leq \max_{a \leq x \leq b} \frac{(x-a)(x-b)}{2} = \frac{(b-a)^2}{8}.$$

Together with the estimate of the global error from the local truncation error (12.6), we get

$$\|U^* - U\|_\infty \leq \left\| h^2 A^{-1} \right\|_\infty \|R\|_\infty \leq \frac{(b-a)^2}{8} \frac{h^2}{12} \left\| u^{(4)} \right\|_\infty$$

and it is left to use that $u^{(4)} = -f''$ ■

The important point of the analysis are

1. the local truncation error is bounded by const $h^2 \left\| u^{(4)} \right\|_\infty$
2. $\left\| u^{(4)} \right\|_\infty$ is bounded from problem data
3. A^{-1} is nonnegative.

In some generalizations (like equations of the form $-(\alpha u')' = f$), with smooth coefficient function $\alpha(x)$ these properties still hold and an essentially similar analysis goes through. However, the 4th derivaties are typically not bounded in more than one dimension, and A^{-1} is not nonegative for many important equations and for systems, such as elasticity. More advanced analysis that overcomes these drawbacks is studied in numerical solution of partial differential equations and in the Finite Element method.

12.3 Finite Difference method in 2D

Inner product, norm, and completeness

See Sections 15.5 and 15.8.

12.4 Galerkin method and finite elements in 1D

12.4.1 Transformation of a model boundary problem to variational form

Consider the model boundary value problem

$$-u'' = f \text{ in } (a,b),\ u(a) = u(b) = 0$$

and assume it has a solution u Multiplying the equation by v and integrating we get

$$-\int_a^b u'' v dx = \int_a^b f v dx$$

By integration by parts

$$\int_a^b u'' v dx = [uv']_{x=a}^{x=b} - \int_a^b u' v' dx = \int_a^b u' v' dx \tag{12.7}$$

using the boundary conditions $u(a) = u(b) = 0$, so

$$\int_a^b u'v'dx = \int_a^b fvdx. \tag{12.8}$$

Not that this equation requires only one derivative of u, rather than two as the original boundary value problem. We can get from (12.8) a statement equivalent to the boundary value problem in the following sense.

Theorem 110 *Let $f \in C[a,b]$. Then u is a solution of the boundary value problem*

$$u \in C^2[a,b], -u'' = f \text{ in } (a,b),\ u(a) = u(b) = 0 \tag{12.9}$$

if and only if u satisfies

$$u \in V, \quad \int_a^b u'v'dx = \int_a^b fvdx \forall v \in V, \tag{12.10}$$

where

$$V = \left\{u \in C^2[a,b] : u(a) = u(b) = 0\right\}.$$

Proof. We have already shown that the boundary value problem (**??**) implies the variational form (12.10). On the other hand, assume that u satisfies the variational form (12.10). Then, using the integration by parts (12.7) again,

$$\int_a^b \left(-u'' - f\right) vdx = 0 \forall v \in V.$$

We only need to show that this statement implies $-u'' - f = 0$. Suppose that $\left(-u'' - f\right)(c) > 0$ for some $c \in (a,b)$. Because $-u'' - f$ is continuous, we have $-u'' - f > 0$ on the interval $(c-\varepsilon, c+\varepsilon)$ for some $\varepsilon > 0$.One can easily construct a function such that $v = 1$ on a smaller interval $(c-\varepsilon/2, c+\varepsilon/2)$ and $v = 0$ outside of the interval $(c-\varepsilon, c+\varepsilon)$ and $v \in C^2[a,b]$. Then $v \in V$, and $\int_a^b \left(-u'' - f\right) vdx > 0$, a contradiction. The case $\left(-u'' - f\right)(c) < 0$ is similar. ■

Finally, note that the bilinear form $\int_a^b u'v'dx$ is positive definite:

$$\int_a^b u'v'dx > 0 \forall v \in V, v \neq 0. \tag{12.11}$$

Indeed.

$$\int_a^b u'u'dx = \int_a^b |u'|^2 dx \geq 0,$$

and

$$\int_a^b |u'|^2 dx = 0 \implies u' = 0 \implies u = \text{const} \implies u = 0,$$

using the boundary conditions $u(a) = u(b) = 0$.

12.4.2 Abstract variational form and the energy functional

We write the variational problem (12.10) in the abstract form

$$u \in V : a(v,u) = F(v) \quad \forall v \in V, \tag{12.12}$$

where $a(\cdot,\cdot)$ is the bilinear form on V

$$a(v,u) = \int_a^b u'v'dx \tag{12.13}$$

and F is the linear functional on V, defined by

$$F(v) = \int_a^b fv dx. \tag{12.14}$$

The reason (12.12) is called a variational form is the following. Recall that in our model problem, the bilinear form is symmetric,

$$a(u, v) = a(v, u),$$

and positive definite:

$$a(u, u) > 0 \forall u \in V, u \neq 0.$$

Consider the quadratic functional on V, defined by

$$J(u) = \frac{1}{2}a(u, u) - F(u).$$

Just like in Calculus, if the functional J attains a minimum, then the derivative of J in the direction v is zero. In the classical Variational Calculus, the directional derivative is called the *variation* of J with respect to the function v, and the equation $\frac{\partial J}{\partial v} = 0$ is called the *Euler equation.of the minimization problem* for J. Using the linearity of a in each argument, the linearity of F, and the symmetry of a, we get

$$\begin{aligned} 0 &= \frac{\partial J}{\partial v}(u) = \lim_{t\to 0} \frac{J(u+tv) - J(u)}{t} \\ &= \lim_{t\to 0} \frac{\left(\frac{1}{2}a(u+tv, u+tv) - F(u+tv)\right) - \left(\frac{1}{2}a(u,u) - F(u)\right)}{t} \\ &= \lim_{t\to 0} \frac{\frac{1}{2}t^2 a(v,v) + t\frac{1}{2}a(u,v) + t\frac{1}{2}a(v,u) - tF(v)}{t} \\ &= a(v,u) - F(v). \end{aligned}$$

So, the variational equation (12.12) is the first order condition of an extremum of J. Because a is positive definite, we have more.

Theorem 111 *The variational problem (12.12) is equivalent to minimizing $J(u)$, and there is at most one minimum.*

Proof. If $J(u)$ is the minimum of J, then

$$\frac{\partial J}{\partial v}(u) = 0, \quad \forall v \in V,$$

so u is a solution of the variational equation (12.12).

Conversely, suppose that u satisfies (12.12). We will show that $J(u)$ is indeed the unique minimum of J. Let $v \in V$. Then, using the linearity of a and F and rearranging terms,

$$\begin{aligned} J(u+v, u+v) &= \frac{1}{2}a(u+v, u+v) - F(u+v) \\ &= \frac{1}{2}a(v,v) + \underbrace{a(v,u) - F(v)}_{=0} + \frac{1}{2}a(u,u) - F(u) \\ &= \frac{1}{2}a(v,v) + J(u) \geq J(u) \end{aligned}$$

with equality if and only if $v = 0$. ■

12.4.3 Discretization by linear elements in 1D

The integrals in the bilinear form (12.13) and the linear functional (12.14) make sense for more general functions, for example continuous and piecewise differentiable ones. A discretization of (12.10) is obtained by choosing nodes

$$a = x_0 < x_1 < \ldots < x_{n+1} = b, \quad x_k = a + kh, \quad h = \frac{b-a}{n+1}$$

and replacing the space V by the space of continuous piecewise linear functions

$$V_h = \left\{ u \in C[a,b] : u|_{(x_{k-1},x_k)} \in \mathcal{P}_1 \right\}$$

to get the discrete problem

$$u_h \in V_h, \quad a\left(v_h, u_h\right) = F(v_h) \quad \forall v_h \in V_h.$$

Of course, according to Theorem 111, the discrete problem is equivalent to

$$J(u_h) \to \min, \quad u_h \in V_h.$$

The functions φ_k, $k = 1, \ldots, N$, defined by

$$\varphi_k \in V_h, \quad \varphi_k(x_j) = \delta_{kj},$$

form a basis of V_h. This follows from the face that any function $u_h \in V_h$, it holds that

$$u_h = \sum_{k=1}^{n} u_h(x_k)\varphi_k$$

Therefore, we will write the coefficients in the expansion as $u_k = u_h(x_k)$. Expanding also v_h in a similar way, we have

$$u_h = \sum_{k=1}^{n} u_k\varphi_k, \quad v_h = \sum_{k=1}^{n} v_k\varphi_k,$$

and

$$\begin{aligned} a(v_h, u_h) &= a\left(\sum_{j=1}^{n} v_j\varphi_j, \sum_{k=1}^{n} u_k\varphi_k\right) = \sum_{j=1}^{n}, \sum_{k=1}^{n} v_j u_k a(\varphi_j, \varphi_k) \\ &= \mathbf{v}^T A\mathbf{u}, \\ F(v_h) &= F\left(\sum_{j=1}^{n} v_j\varphi_j\right) = \sum_{j=1}^{n} v_j F(\varphi_j) \\ &= \mathbf{v}^T\mathbf{f}, \end{aligned}$$

where

$$A = \begin{bmatrix} a(\varphi_1,\varphi_1) & \ldots & a(\varphi_1,\varphi_n) \\ \vdots & \ddots & \vdots \\ a(\varphi_n,\varphi_1) & \ldots & a(\varphi_n,\varphi_n) \end{bmatrix}, \mathbf{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix}, \mathbf{v} = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}, \mathbf{f} = \begin{bmatrix} F(\varphi_1) \\ \vdots \\ F(\varphi_n) \end{bmatrix}.$$

For the model problem here, we get by a simple computation

$$A = \frac{1}{h}\begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & \ddots & & \\ & & & 2 & -1 \\ & & & -1 & 2 \end{bmatrix}.$$

If the integrals in the right-hand side vector $\mathbf{f}$ are approximated by the midpoint formula, we have

$$\mathbf{f} \approx \widetilde{\mathbf{f}} = h \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix}.$$

So, for this model problem, the resulting system of equations $A\mathbf{u} = \widetilde{\mathbf{f}}$ is exactly same as obtained in the finite difference method.

12.5 Galerkin method and finite elements for a model problem in 2D

12.6 Existence of solution and error estimates

12.6.1 Sobolev spaces

The problems to be solved are

$$u \in V : a(u,v) = F(v) \forall v \in V \text{ (continuous problem)} \tag{12.15}$$
$$u_h \in V : a(u_h, v_h) = F(v_h) \forall v_h \in V_h \text{ (discrete problem)} \tag{12.16}$$

In the discrete case (12.15), we have established the existence of the solution of the variational problem by reducing it to a system of linear algebraic equations with a positive definite (hence, nonsingular) matrix. But, in general, the variational problem (12.15) equivalently, the minimization problem $J(u) \to \min$, $v \in V$, does not necessarily have a solution. One way to see this is to realize that there is no obvious reason why a minimizing sequence

$$u^{(k)} \in V, \quad J(u^{(k)}) \to \inf_{w \in V} J(w)$$

should have a limit, which would serve as the solution. If V consists of twice differentiable functions, the functions $u^{(k)}$might converge to a function that is outside of V, for example, has a kink. If we modify the definition of V to contain all continuous piecewice differentiable functions, it is again possible to find a sequence of functions that converges to a function outside of V, such as with infinitely many kinks, and so on. To obtain a satisfactory theory, one needs the space V to be complete. To be able to speak about completeness, one needs to define a norm first. Since $a(\cdot,\cdot)$ is symmetric and positive definite, it can serve as inner product. However, the form $a(\cdot,\cdot)$ involves only first derivatives, and it is important to control also the values of the function. This is provided by the following theorem.

Theorem 112 (Poincaré inequality) *Let Ω be a bounded domain in $\mathbb{R}^N$ (in the case $N = 1$, a bounded open interval). Then there exists a constant $C = C(\Omega)$ such that*

$$\forall u \in C^1(\overline{\Omega}),\ v = 0 \text{ on } \partial\Omega : \int_\Omega |u|^2 \, dx \le C(\Omega) \int_\Omega \nabla u \cdot \nabla u dx.$$

Though the form $a(\cdot,\cdot)$ is often used as inner product, we find it usually more convenient to formulate the theory in terms of the inner product

$$(u,v) = \int_\Omega uv dx + \int_\Omega \nabla u \cdot \nabla v dx.$$

To add the missing limit to the space, we define V as the *completion* of the space $\left\{u \in \forall u \in C^1(\overline{\Omega}),\ v = 0 \text{ on } \partial\Omega\right\}$.This space is called the Sobolev space $H_0^1(\Omega)$. The letter H signals it is a Hilbert space, the superscript 1 stands for the order of the derivatives involved, and the subscript 0 stands for the zero boundary

conditions. The elements of this space are classes of sequences with the same limit; one can show that they can be identified with functions on Ω.The form $a(\cdot,\cdot)$ and the functional $F(\cdot)$ are extended to the space V by continuity:

$$\begin{aligned} & \|u_n - u\| \to 0, \|v_n - v\| \to 0 \\ \Longrightarrow \quad & a(u,v) = \lim_{n\to\infty} a(u_n, v_n), F(u) = \lim_{n\to\infty} F(u_n). \end{aligned}$$

. Note that this setting also generalizes the concept the derivative and integral; the full development of this theory requires the use of *Lebesgue integral* (taught in Real Analysis), and of *generalized derivatives and distributions* (taught in advanced Functional Analysis and in Mathematical Theory of Finite Elements).

So, we have constructed V as a Hilbert space with inner product $(.,.)$, $V_h \subset V$ is a finite dimensional subspace, $a(.,.)$ a bilinear form on V, and $F(.)$ a linear functional on V.We also have the inequalities

$$\forall u \in V : a(u,u) \geq \alpha \|u\|^2, \quad \alpha > 0 \text{ (V-ellipticity)} \tag{12.17}$$

$$\forall u, v \in V : |a(u,v)| \leq M \|u\| \|v\| \text{ (boundedness of } a) \tag{12.18}$$

$$\forall u, v \in V : |F(v)| \leq C \|v\| \text{ (boundedness of } F) \tag{12.19}$$

In our model problem, $\alpha = C(\Omega)$ is the constant from Poincaré inequality, $M = 1$ (from Cauchy inequality for a), and, from Cauchy inequality, with

$$C = \sqrt{\int_\Omega |f|^2}.$$

12.6.2 Existence of solution

We assume that V is a Hilbert space with inner product $(.,.)$, $V_h \subset V$ is a finite dimensional subspace, $a(.,.)$ a bilinear form on V, and $F(.)$ a linear functional on V, and (12.17-12.19) hold. The questions are

1. does the problem (12.15) have a unique solution u?
2. does the problem (12.16) have a unique solution u_h?
3. find a bound on the discretization error $u - u_h$

Theorem 113 (Lax-Milgram) *The problem (12.15) has a unique solution u.*

Proof. From (12.18), the mapping $v \mapsto a(u,v)$ (with a fixed u) is a bounded linear functional. From the Riesz' representation theorem (Theorem **??**), there is $Au \in V$ such that

$$(Au, v) = a(u,v) \forall v \in V$$

and the mapping $u \mapsto Au$ is linear. Also from the Riesz theorem, there is unique $g \in V$ such that

$$(g, v) = F(v) \forall v \in V$$

The problem (12.15) is then equivalent to $Au = t$, which, for an arbitrary $t \neq 0$, is equivalent to the fixed point problem

$$u = u - t(Au - g).$$

Note that using Lemma 172, for a fixed w,

$$\|Aw\| = \sup_{z\neq 0} \frac{|\langle z, Aw\rangle|}{\|z\|} = \sup_{z\neq 0} \frac{|a(z,w)|}{\|z\|} \leq M \|w\|.$$

We want to show that for a suitable t, the mapping $\mathcal{A}u \mapsto u - t(Au - g)$ is a contraction. We have

$$\|\mathcal{A}u - \mathcal{A}v\| = \|(I - tA)(u - v)\|$$

and writing $w = u - v$,

$$\begin{aligned}\|(I - tA)w\|^2 &= \langle w - tAw, w - tAw\rangle \\ &= \underbrace{\langle w, w\rangle}_{=\|w\|^2} - t\underbrace{\langle Aw, w\rangle}_{\geq\alpha\|w\|^2} - t\underbrace{\langle w, Aw\rangle}_{\geq\alpha\|w\|^2} + t^2 \underbrace{\langle Aw, Aw\rangle}_{=\|Aw\|^2\leq M^2\|w\|^2} \\ &\leq \underbrace{\left(1 - 2\alpha t + t^2M^2\right)}_{f(t)} \|w\|^2 .\end{aligned}$$

To apply the Banach contraction principle (Theorem 156), it is enough to note that V is complete and choose t so that $f(t) < 1$.This is clearly possible for $t > 0$ sufficiently small, because $f(0) = 0$ and $f'(0) < 0$. Or one can choose t so that $f'(t) = 0$, $t = \frac{M^2}{\alpha}$. ■

Corollary 114 *The discrete problem (12.16) has a unique solution.*

Proof. Apply the Lax-Milgram theorem in space V_h, which is finite dimensional and thus complete. ■

12.6.3 An error estimate

Here is a basic error estimate in the norm of the space V.

Theorem 115 (Céa's Lemma)

$$\forall v_h \in V_h : \|u - u_h\| \leq \frac{M}{\alpha} \|u - v_h\|$$

Proof. Let $w_h \in V_h$. Because $V_h \subset V$, we have from (12.15)

$$a(u, w_h) = F(w_h)$$

and recalling (12.16),

$$a(u_h, w_h) = F(w_h).$$

Subtracting these two equations gives

$$a(u - u_h, w_h) = 0 \forall w_h \in V_h.$$

Now for any $v_h \in V_h$,

$$\begin{aligned}\alpha \|u - u_h\|^2 &\leq a(u - u_h, u - u_h) \\ &= a(u - u_h, u - u_h + \underbrace{(u_h - v_h)}_{\in V_h}) \\ &= a(u - u_h, u - v_h) \\ &\leq M \|u - u_h\| \|u - v_h\|\end{aligned}$$

and it remains to divide the inequality by $\|u - u_h\|$. ■

Céa's Lemma shows that the discretization error is at most constant times the error of the best possible approximation. One can take v_h to be a suitable kind of interpolation of u. For problems on a mesh with step h, one can show for the model problem that the best approximation error is $O(h)$.A drawback of this variational analysis is that the norm in V may not be the error norm of interest and the error is $O(h)$ rather than the expected $O(h^2)$. This is caused by the special form of the norm on V.Much of the mathematical theory of finite elements is concerned with further

development of these estimates to other norms, and in particular obtaining errors of order $O(h^2)$ in the maximal difference between u and u_h (generalizing the concept of "maximal" in a suitable way).

The thin book by Johnson [Joh87] is very nice transparent introduction. For a modern, self-contained, and relatively accessible treatment including the development of the needed functional spaces and elliptic partical differential equations, see Brenner and Scott [BS02]. A standard reference from a more engineering point of view is the monograph by Hughes [Hug00]. The book [CL91] contains an updated version of the classical original monograph by Ciarlet as well as advanced chapters on estimates in various norms.

Chapter 13

Computing eigenvalues and eigenvectors

Review of eigenvalues and eigenvectors

See section **??**.

13.1 Power method

Let $A \in \mathbb{C}^n$ and $Av_j = \lambda_j v_j$, $v_j \neq 0$. For simplicity, suppose that A is diagonalizable, that is, the eigenvectors form a basis of $\mathbb{C}^n$, and the eigenvalue largest in magnitude is simple. Order the eigenvalues so that

$$|\lambda_1| > |\lambda_2| \geq \ldots |\lambda_n| .$$

Let $x_0 \in \mathbb{C}^n$ and consider the iterates $x_{k+1} = Ax_k$. An easy computer experiment shows that the ratios of the entries of x_{k+1} and x_k converge to an eigenvalue of A, at least much of the time, and the speed of convergence varies. After some iterations, x_k is too large or too small, and overflow or underflow occurs. A better version of the algorithm is as follows.

Algorithm 116 (Power method for eigenvalues) *Given x_0, $r \in \mathbb{C}^n$, let $w_0 = x_0$, and do for $k = 1, 2, \ldots$:*

$$\begin{aligned} z_k &= Aw_k \\ \Lambda_k &= w_k^* z_k \\ w_{k+1} &= \frac{z_k}{\|z_k\|} \end{aligned}$$

The last step, normalization, prevents overflow or underflow.

To see what is going on, expand x_0 in the basis of the eigenvectors:

$$x_0 = \sum_{i=1}^{n} c_i v_i$$

and assume $c_1 \neq 0$. Then

$$\begin{aligned} x_k &= A^k x_0 = \sum_{i=1}^{n} c_i A^k v_i \\ &= \sum_{i=1}^{n} c_i \lambda_i^k v_i \\ &= \lambda_1^k \left(c_1 v_1 + \sum_{i=2}^{n} c_i \left(\frac{\lambda_i}{\lambda_1} \right)^k v \right) \\ &= \lambda_1^k \left(c_1 v_1 + O\left(\left| \frac{\lambda_i}{\lambda_1} \right|^k \right) \right) \end{aligned} \tag{13.1}$$

and

$$w_k = \frac{x_k}{\|x_k\|}.$$

Theorem 117 *If A is diagonalizable, the eigenvalues of A satisfy*

$$|\lambda_1| > |\lambda_2| \geq \ldots \geq |\lambda_n|,$$

the first coefficient in the expansion of x_0 in the basis of eigenvector is $c_1 \neq 0$, then

$$\begin{aligned} \Lambda_k - \lambda_1 &= O\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \\ \left\| w_k - \left(\frac{\lambda_1}{|\lambda_1|} \right)^k v_1 \right\| &= O\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \end{aligned}$$

Proof. The proof follows immediately from the expansion (13.1) using $\|z_k\|^2 = z_k^* z_k$ and $w_k^* w_k = 1$. ■

The convergence of the vector w_k to the eigenvector is called convergence in direction.

To speed up the convergence we can apply the method to the shifted matrix $\mu I - A$,with eigenvalues $\mu - \lambda_i$. The power method will then converge to the eigenvalue farthest away from μ. The shift and inverse method consists of applying the power method to the matrix $B = (\mu I - A)^{-1}$. From the spectral mapping theorem, the eigenvalues of A and B are related by

$$\lambda_B = \frac{1}{\mu - \lambda_A}, \quad \lambda_A = \mu - \frac{1}{\lambda_B}.$$

The shift and invert method converges to the eigenvalue λ_j closest to the shift μ with the error $O\left(\left| \frac{\lambda_j - \mu}{\lambda_k - \mu} \right|^k \right)$,where $|\lambda_j|$ is the the eigenvalue second closest to μ This method is well suited for computing eigenvectors and improving accuracy of eigenvalues obtained by other methods. When the shift μ is updated in every iteration from the best available estimate, we get the inverse method.

Algorithm 118 (Inverse iteration for eigenvalues) *Let $x_0 \in \mathbb{C}^n$ and $\lambda_0 \in \mathbb{C}$, w_0, $w_0^* w_0 = 1$,and compute*

$$\begin{aligned} (\Lambda_k I - A)\, z_k &= w_k \\ \sigma_k &= w_k^* z_k \\ \Lambda_{k+1} &= \Lambda_k - \frac{1}{\sigma_k} \\ w_{k+1} &= \frac{z_k}{\|z_k\|} \end{aligned}$$

It can be shown that the inverse iteration method converges with order 3. In practice, two or three iterations are sufficient.

Unitary matrices

See Section 16.7.

13.2 Jacobi method for eigenvalues of symmetric matrices

13.3 QR decomposition by Givens rotations

Definition 119 *Givens rotation is the matrix*

$$G = \begin{bmatrix} \bar{c} & -\bar{s} \\ s & c \end{bmatrix}, \quad |c|^2 + |s|^2 = 1.$$

Proposition 120 *A Givens rotation is unitary:*

$$G^*G = \begin{bmatrix} \bar{c} & -\bar{s} \\ s & c \end{bmatrix}^* \begin{bmatrix} \bar{c} & -\bar{s} \\ s & c \end{bmatrix} = I$$

Proposition 121 *For any vector in $\mathbb{C}^2$, we can find a Givens rotation such that multiplication by the rotation anihilates the second entry of the vector:*

$$\begin{bmatrix} \bar{c} & -\bar{s} \\ s & c \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \dots \\ sx + cy \end{bmatrix} = \begin{bmatrix} \dots \\ 0 \end{bmatrix}$$

by choosing

$$s = \frac{-y}{\sqrt{|x|^2 + |y|^2}}, \quad c = \frac{x}{\sqrt{|x|^2 + |y|^2}} \tag{13.2}$$

Remark 122 *The notation is motivated by the fact that in the real case, we can write $c = \cos\varphi$, $s = \sin\varphi$, like in the Jacobi method.*

Similarly as in the Jacobi method, we embed Givens rotation in the row j and column k of the identity matrix to define

$$U = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & \bar{c} & \dots & -\bar{s} & & \\ & & \vdots & \ddots & \vdots & & \\ & & s & \dots & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix} \begin{matrix} \\ \\ i \\ \\ j \\ \\ \ddots \end{matrix}$$

Multiplication of $A = (a_{jk}) \in C^{m \times n}$ by U modifies only rows i and j; for $B = (b_{jk}) = UA$,

$$b_{ik} = ca_{ik} - sa_{jk}, \quad b_{jk} = sa_{ik} + ca_{jk},$$

To anihilate any given entry b_{kj} of row j, choose the rotation from (13.2) with

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{ik} \\ a_{jk} \end{bmatrix}$$

We denote this rotation by $U(i, j, k, A)$.

We can use rotations to anihilate the entries in the lower triangle of A are anihilated in turn and get the QR decomposition. To anihilate entry at the position (i, j), we use Givens rotation of the rows i and j. First the entries of the first column are eliminated under the diagonal, then the second column, and so on.

Algorithm 123 (QR decomposition by Givens rotations) $R = A$
$Q = I$
for $i = 1 : m$
  for $j = i + 1 : \min(m, n)$
    $k = j$
    $R = U(i, j, k)R$
    $Q = QU(i, j, k)^*$
  end
end

Proposition 124 *Algorithm 123 computes a decomposition $A = QR$, where Q is unitary and R is upper triangular.*

Proof. In each step of the algorithm, the relation $A = QR$ is preserved because $U(i, j, k)$ is unitary. Because of the choice of the sequence of i and j, once an entry of R is set to zero, it will remain zero, and in the end, R is upper triangular. Finally, Q is product of unitary matrices and so it is unitary. ■

In a practical implementation, instead of forming the matrix Q explicitly, it is possible to record the coefficients of the rotations to perform the mutiplication of Q or Q^* and a vector by applying all rotations in turn.

It is easy to see that if the columns of $A \in \mathbb{C}^{m\times n}$ are linearly independent, then the first n columns of Q are the same vectors (up to scalar multiples) as in the Gram-Schmidt orthogonalization of the columns of A.This follows from the fact that, due to R being upper triangular, the span of the first k columns of A is the same as the span of the first k columns of Q, for all k.

13.4 Reduction to Hessenberg form by Givens rotations

Modern methods for computing eigenvalues proceed by transforming the matrix to a similar one by unitary transformations. The first step is to transform the matrix into a similar one with many zeros. We can force zeros in a part of column k of matrix H by rotating rows $i = k + 1$ and $j > i$ (that is, multiplying by $U(i, j, k, H)$ from the left) to anihilate the entries position (j, k).Each such transformation affects in the column k only the rows $k+1$ and j; hence, once a zero is forced, it will remain. Also, the matching multiplications by the conjugate transpose of the rotations from the left only affect columns $i, j > k$ (the symbol x is an affected position, 0 means forced zero):

$$U(i,j,k,H)HU(i,j,k,H)^* = \begin{bmatrix} & & x & & x & & \\ & & x & & x & & \\ x & x & x & x & x & x & x \\ & & x & & x & & \\ x & 0 & x & x & x & x & x \\ & & x & & x & & \\ & & x & & x & & \end{bmatrix} \begin{matrix} \\ \\ k \\ i \\ \\ j \\ \\ \end{matrix}$$

Processing each column in turn, we get the following algorithm.

Algorithm 125 *Given matrix $A \in \mathbb{C}^{n\times n}$, do:*
$H = A$
for $k = 1 : n - 2$
  for $j = k + 2 : n$
    $i = k + 1$
    $H = U(i, j, k, H)HU(i, j, k, H)^*$
  end
end

Proposition 126 *Algorithm 125 returns*

$$H = QAQ^*,$$

where Q is a unitary matrix, and $H = (h_{jk})$ is in upper Hessenberg form, $h_{jk} = 0$ for $j \geq k + 2$:

$$H = \begin{bmatrix} h_{11} & h_{12} & \dots & \dots & h_{1n} \\ h_{21} & h_{22} & & & \vdots \\ 0 & h_{32} & & & \vdots \\ \vdots & \vdots & \ddots & & \vdots \\ 0 & 0 & & h_{n,n-1} & h_{nn} \end{bmatrix}$$

If A is Hermitean, $A^* = A$, then H is tridiagonal.

In the symmetric (or Hermitean, for complex numbers) case, efficient algorithms for finding eigenvalues of tridiagonal matrices are known. In the general case, the method of the following section is used.

13.5 QR algorithm for eigenvalues

Algorithm 127 *Let A be given. Set $A_1 = A$ and for $k = 1, 2, ...$compute the QR decomposition*

$$A_k = Q_k R_k$$

and set

$$A_{k+1} = R_k Q_k.$$

The matrices A_k are unitarily similar:

$$Q_k^* A_k Q = \underbrace{Q_k^* Q_k}_{I} R_k Q = A_{k+1}$$

A quick computer experiment will show that A_k converges to a triangular matrix or a block triangular matrix, where (in the real case) diagonal blocks of order two have have eigenvalues that are conjugate pairs of eigenvalues of A. For faster convergence, the algorithm is modified by using judiciously chosen shifts

$$A_k - \mu_k = Q_k R_k.$$

The analysis of the QR algorithm and formulation the shifed version are outside of the scope of these notes; see, for example,[GVL96].

The reason why the reduction to Hessenberg form is done first is that then all matrices R_k and Q_k are in Hessenberg form, which results in savings because the zeros are not computed.

For more details on eigenvalue algorithms, see Matlab help for the functions *qr*, *eig*, and *hess*, and LAPACK User's guide and source code, available from NETLIB at `http://www.netlib.org/lapack`

Part II

Topics from Linear Algebra and Analysis

The material here is not meant to be covered at once, but rather selected sections are to be presented right before they are needed, as indicated by the references in the main text. Some of the material are prerequisites that should be familiar from Linear Algebra and Calculus, some will be new. *Note: Exercises marked by (*) require some knowledge of Mathematical Analysis (also known as Advanced Calculus). Students not familiar with analysis will have to take these statements for granted.*

Chapter 14

Algebra and Calculus

14.1 Taylor and binomial theorems

Theorem 128 *If f has derivatives of order $n+1$ in the interval (a,b) spanned by the points x, $x+h$, and $f^{(n)}$ is continous in $[a,b]$, then*

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \cdots + \frac{h^n}{n!}f^{(n)}(x) + \frac{h^{n+1}}{(n+1)!}f^{(n+1)}(\xi)$$

for some $\xi \in I$.

Applying this theorem to the function $f(x) = (1+x)^a$ at $x = 0$, we have

$$\frac{d^k}{dx^k}(1+x)^a = a(a-1)\cdots(a-k+1)(1+x)^{a-k},$$

so

$$(1+x)^a = 1 + ax + \binom{a}{2}x^2 + \ldots + \binom{a}{n}x^n + R_n, \quad R_n = \binom{a}{n+1}x^{n+1}(1+\xi)^{a-(n+1)}, \quad -x < \xi < x,$$

where

$$\binom{a}{k} = \frac{a}{1}\frac{a-1}{2}\cdots\frac{a-k+1}{k}.$$

Note that if a is a positive integer, $\binom{a}{k} = 0$ for $k > a$ so we get a finite sum, known as the binomial theorem from algebra; otherwise, we obtain the infinite series

$$(1+x)^a = \sum_{n=0}^{\infty}\binom{a}{n}x^n, \qquad |x| < 1;$$

because it can be shown that the remainder $R_n \to 0$ as $n \to \infty$ if $|x| < 1$. For $a = -1$, this becomes the familiar power series from calculus,

$$\frac{1}{1+x} = \sum_{n=0}^{\infty}(-1)^n x^n, \qquad |x| < 1.$$

An important special case of the Taylor theorem is obtained for $n = 0$:

Theorem 129 (Mean Value) *If f is continous on interval $[a,b]$ and has derivative on (a,b), then there exists $\xi \in (a,b)$ such that*

$$f(b) - f(a) = f'(\xi)(b-a).$$

14.2 Directional derivatives and Taylor theorem in multiple dimensions

The derivative of function $f : \mathbb{R}^n \to \mathbb{R}$ at the point x in the direction of vector h is denoted as and defined by

$$D_h f(x) = \frac{\partial f}{\partial h}(x) =^{def} \lim_{t\to 0} \frac{f(x+th) - f(x)}{t} = \left[\frac{\partial f(x+th)}{\partial t}\right]_{t=0}$$

From the Taylor expansion of the function $g(t) = f(x+th)$ at $t = 0$ we have the Taylor expansion of f at x. In the $2D$ case, using the chain rule and the binomial theorem, this becomes

$$f(x+h) = \sum_{k=0}^{m} \frac{1}{k!}\left(h_1\frac{\partial}{\partial x_1} + h_2\frac{\partial}{\partial x_2}\right)^k f(x) + \frac{1}{(m+1)!}\left(h_1\frac{\partial}{\partial x_1} + h_2\frac{\partial}{\partial x_2}\right)^{m+1} f(x+\xi h), \quad 0 < \xi < 1.$$

Another important case is the total differential, obtained as Taylor expansion of order 1 in the n-dimensional case:

$$f(x+h) = f(x) + \sum_{j=1}^{n} h_j \frac{\partial f(x)}{\partial xj} + O(\|h\|^2).$$

14.3 Polynomials

Theorem 130 *(Fundamental theorem of algebra) Every polynomial of degree $n > 0$ has at least one root $\lambda \in \mathbb{C}$.*

When polynominal p of degree n has root λ_n, we can divide it by the root factor $(\lambda - \lambda_n)$, and get

Corollary 131 *If p is polynomial of degree n $p(\lambda) = a_0 + \cdots + a_n x^n, a_n \neq 0$, then $p(\lambda) = a_n(\lambda - \lambda_1)\cdots(\lambda \cdots \lambda_1)$, where $\lambda_1, \ldots \lambda \in \mathbb{C}$ are its roots.*

14.4 Bernoulli polynomials and Euler-Maclaurin Formula

Definition 132 *The polynomials defined by*

$$\begin{aligned} B_0(t) &= 1 \\ B_n(t) &= n\int B_{n-1}(t)dt, \quad \int_0^1 B_n(t)dt = 0, \quad n \geq 1, \end{aligned}$$

are called Bernoulli polynomials. *The numbers $b_n = B_n(0)$ are called* Bernoulli numbers.

Clearly, B_n is a polynomial of degree n with leading coefficient one, and $B_1(t) = t - \frac{1}{2}$.

Lemma 133 *It holds that $B_n(0) = B_n(1), \quad n \geq 2$*

Proof. $0 = \int_0^1 B_n(t)dt = (n+1)\left(B_{n+1}(1) - B_{n+1}(0)\right), \quad n \geq 1.$ ■

Lemma 134 *$b_n = 0$ for $n \geq 3$ odd*
Proof. *We first need symmetry of the Bernoulli polynomials,*

$$B_n(t) = (-1)^n B_n(1-t), \quad n \geq 0.$$

This is clearly true for $n = 0$. Assume the equation holds for $n-1$; then by integration, $B_n(t) = (-1)^n B_n(1-t) + c$. Using $\int_0^1 B_n(t)dt = 0$, we have $c = 0$. The Lemma follows by substituting $t = 0$.
■

These polynomials are useful in repeated integration by parts:

$$\begin{aligned}
\int_0^1 f(t)dt = \int_0^1 B_0(t)f(t)dt = \quad & [B_1(t)f(t)]_0^1 - \int_0^1 B_1(t)f'(t)dt \\
\int_0^1 B_1(t)f'(t)dt = \quad & \frac{1}{2}\left[B_2(t)f'(t)\right]_0^1 - \frac{1}{2}\int_0^1 B_2(t)f''(t)dt \\
& \vdots \\
\int_0^1 B_{n-1}(t)f^{(n-1)}(t)dt = \quad & \frac{1}{n}\left[B_n(t)f^{(n-1)}(t)\right]_0^1 - \frac{1}{n}\int_0^1 B_n(t)f^{(n)}(t)dt,
\end{aligned}$$

which gives

Theorem 135 (Euler-Maclaurin Formula) *Let $f \in C^n[a,b]$, $n \geq 2$ Then*

$$\begin{aligned}
\int_0^1 f(t)dt \quad = \quad & \frac{1}{2}\left(f(a)+f(b)\right) + \sum_{k=2}^{n} \frac{(-1)^{k-1}b_k}{k!}\left(f^{(k-1)}(1) - f^{(k-1)}(0)\right) \\
& + \frac{(-1)^n}{n!}\int_0^1 B_n(t)f^{(n)}(t)dt.
\end{aligned}$$

Only the terms with odd k may be nonzero.

See [AS92, KC02, **?**] for more details and a refined statement of the formula. Warning: in some of the literature, the scaling of the Bernoulli polynomials differs.

Chapter 15

Linear spaces

A linear space is a set of elements, called vectors, on which are defined the operations of addition of vectors and multiplication of scalars. See Linear Algebra for details. The important thing for us is that the operations obey the same laws as operations on vectors in Euclidean space. The scalars can be either real or complex. Linear spaces will be denoted by letters $U, V, \ldots$ Some examples of particular linear spaces of importance follow.

Example 136 *$\mathbb{R}^n$ is the n-dimensional Euclidean space known from linear algebra and calculus. $\mathbb{C}^n$ is the n-dimensional Euclidean space with complex numbers as scalars.*

Example 137 *$C[a,b]$ is the space of all continuous functions on a closed (and bounded) interval $[a,b]$, with operations on functions defined componentwise:*

$$(f+g)(x) = f(x) + g(x).$$

Example 138 *$\mathcal{P}_n$ is the linear space of all polynomial of degree at most n.The space of all polynomials (of any degree) is denoted by$\mathcal{P} = \bigcup_{n=0}^{\infty} \mathcal{P}$.*

Definition 139 *A set of vectors S in a linear space V is linearly independent if for any $u_1, \ldots, u_n \in S$, the only way a linear combination $\sum_{j=1}^{n} a_j u_j$, (a_j are scalars), equals to zero is that $a_1 = \ldots = a_n = 0$.*

A linear combination has always *finitely many terms* - without the concept of a limit, we cannot define an infinite sum. But a linearly independent set can be infinite:

Example 140 *The set $\{1, x, x^2, \ldots, x^n, \ldots\}$ is linearly independent.*

Definition 141 *The span of set S a linear space V is the set of all linear combinations of elements of S :*

$$\operatorname{span} S = \left\{ u \,\middle|\, u = \sum_{j=1}^{n} a_j u_j \text{ for some } n, \text{ scalars } a_j, \text{ and } u_j \in S \right\}$$

Example 142 $\mathcal{P} = \operatorname{span}\left\{1, x, x^2, \ldots, x^n, \ldots\right\}$.

Definition 143 *A set B in a linear space is a* basis *of V if B is linearly independent and $V = \operatorname{span} B$.A linear space is called finite dimensional if if has a finite basis.*

Theorem 144 *If $\{u_1, \ldots, u_n\}$ is a basis of a linear space V, then every element $u \in V$ can be written as*

$$u = \sum_{j=1}^{n} a_j u_j$$

where the scalars a_j are determined uniquely.

Example 145 *The set of all vectors $e^i \in \mathbb{R}^n$, defined as all zeros except one at position i, is a basis of $\mathbb{R}^n$. This basis is called the* canonical basis.

15.1 Normed linear spaces

Example 146 ***Definition 147*** *A* norm *on a linear space V is a function*

$$\|\cdot\| : v \to \|v\| \in \mathbb{R}$$

such that

Definition 148 *1.* $\|v\| = 0$, $\|v\| = 0 \Leftrightarrow v = 0$

2. α *scalar* $\Rightarrow \|\alpha v\| = |\alpha| \, \|v\|$

3. (triangle equality) $\|u + v\| \leq \|u\| + \|v\|$

A linear space equipped with a norm is called a *normed linear space.*

Definition 149 *A sequence of vectors $\{u_n\} \subset V$, V a normed linear space,* converges *to $v \in V$ (written as "$u_n \to u$ in V" or "$lim_{n\to\infty} u_n = u$ in V") if* $\lim_{n\to\infty} ||u_n - u|| = 0$.

Note that we do not define the concept of convergence if there is no norm. In the case of the n-dimensional Euclidean space, one can prove that convergence in any norm is equivalent to convergence of the coordinates of the vectors.

Example 150 *$V = \mathbb{C}^n$, $\|v\| = \sqrt{\sum_{i=1}^n |v_i|^2}$ is the* Euclidean norm *known from calculus and linear algebra.*

Example 151 *$V = \mathbb{R}^n$ or $\mathbb{C}^n$, $\|v\|_p = (\sum_{i=1}^n |v_i|^p)^{1/p}$, $1 \leq p < +\infty$, is the so-called p-*norm*. The common important cases are the* 1*-norm,*

$$\|v\|_1 = \sum_{i=1}^n |v_i|,$$

and the 2*-norm, which is the Euclidean norm.*

Example 152 *$V = \mathbb{R}^n$ or $\mathbb{C}^n$, $\|v\|_\infty = \max_i |v_i|$. This is the* maximum norm.

Exercise 153 *Show that for any v, $\lim_{p\to\infty} \|v\|_p = \|v\|_\infty$.*

It is a simple exercise in Calculus to show that $\lim_{p\to+\infty} \|v\|_p = \|v\|_\infty$. In some applications, this allows to replace $\|v\|_\infty$ by $\|v\|_p$ with a (not very) large p. This is useful in optimization, because $\|v\|_p^p$ is a differentiable function of p, while $\|v\|_\infty$ is not differentiable.

Example 154 *$V = C[a,b]$, $\|v\|_{C[a,b]} = \max_{a\leq x\leq b} |v(x)|$. This is the space of all continuous functions on the interval $[a,b]$, equipped with the maximum norm for functions. Other common notations for this norm are $\|v\|_{\infty,a,b}$ or $\|v\|_\infty$, if a and b are clear from the context. Note that the norm is defined for any $v \in C[a,b]$, because the function $|v|$ is continuous on the closed interval $[a,b]$, and so it attains a maximum.*

Exercise 155 *Verify the axioms of norm to show that this is indeed a norm.*

15.2 Banach contraction theorem

Theorem 156 *(Banach) Let $S \subset V$, V complete normed linear space, S closed, and a mapping $T : S \to S$ be such that for some $C < 1$,*

$$\forall u, v \in S \quad \|Tu - Tv\| \leq C\|u - v\|.$$

Then (i) There exists exactly one fixed point u^* of T in S, and (ii) for any u_0, the sequence $u_{n+1} = Tu_n \to u^*$ as $n \to \infty$.

Proof. We show there exists at most one fixed point. Assume $x^* = Tx_8 \in S$ and $x^{**} = Tx^{**} \in S$. Then

$$\begin{aligned} x^* - x^{**} &= Tx^* - Tx^{**} \\ \|x^* - x^{**}\| &= \|Tx^* - Tx^{**}\| \leq q\|x^* - x^{**}\| \\ &\Rightarrow \underbrace{\|x^* - x^{**}\|}_{=0}\underbrace{(1-q)}_{>0} \leq 0 \\ &\Longrightarrow ||x^* - x^{**}|| = 0 \Longrightarrow x^* = x^{**} \end{aligned}$$

■

Proof. We need to prove that there is at least one fixed point. Let $u_0 \in S$, $u_{n+1} = Tu_n$. We show that the sequence
Show $\{a_n\}$ is Cauchy: From $u_1 = Tu_0$, $u_2 = Tu_1 \quad u_3 = Tu_2 \ldots$, we have

$$\begin{aligned} \|u_1 - u_2\| = \|Tu_0 - Tu_1\| &\leq q\|u_0 - u_1\| \\ \|u_3 - u_2\| = \|Tu_3 - Tu_2\| &\leq q\|u_1 - u_2\| \\ \vdots &\leq \quad \vdots \\ \|u_{n+1} - u_n\| &\leq q^n\|u_1 - u_0\| \end{aligned}$$

■

Proof.

$$\begin{aligned} \|u_m - u_n\| &\leq \|u_{n+1} - u_n\| + \|u_{n+2} - u_{n=1}\| + \ldots + \|u_{m-1} - u_m\| \\ &\leq (q^n + q^{n+1} + \ldots + q^m)\|u_1 - u_0\| \\ &= q^n(\underbrace{1 + q + \ldots + q^{m-n}}_{\leq \frac{1}{1-q}})\|u_1 - u_0\| \end{aligned}$$

hence,

$$\|u_m - u_n\| \leq \frac{q^n}{1-q}\|u_1 - u_0\|.$$

Given $\varepsilon > 0$, we choose N so that $\frac{q^n}{1-q}\|u_1 - u_0\| < \varepsilon$. Then

$$m > n \geq N \Rightarrow \|u_m - u_n\| \leq \frac{q^n}{1-q}\|u_1 - u_0\| < \varepsilon,$$

which shows that $\{u_m\}$ is Cauchy. Because the sequence $\{u_m\}$ is Cauchy, it has a limit $u \in V$.

Now we need to show that $u \in S$. and $Tu = u$. Since S is closed, $u_n \subset S \Rightarrow u \in S$. Finally,

$$\begin{aligned} \|Tu - u\| &= \|Tu - Tu_n + u_{n+1} - u\| \\ &\leq \|Tu - Tu_n\| + \|u_{n+1} - u\| \\ &\leq q\underbrace{\|u - u_n\|}_{\to 0} + \underbrace{\|u_{n+1} - u\|}_{\to 0} \qquad (15.1) \\ &\Rightarrow \|Tu - u\| = 0 \Rightarrow Tu = u. \qquad (15.2) \end{aligned}$$

■

15.3 Linear Operators

Definition 157 *Let U, V be linear spaces. A mapping $A : U \to V$ is a linear mapping (or a linear operator) if $A(au + bv) = aAu + bAv$, $\forall a, b$ scalars, $u, v \in U$.*

For linear operators, we customarily write Au for $A(u)$ if there is no danger of confusion, because of the relation of linear operators to matrix-vector multiplication.

Example 158 *Let $U = \mathbb{R}^m$, $V = \mathbb{R}^n$, $B \in \mathbb{R}^{m\times n}$. Then $A : u \in \mathbb{R}^m \to Bu \in \mathbb{R}^n$ is a linear operator.*

If there is no danger of confusion, the matrix and the operator are denoted by the same symbol.

Definition 159 *For linear operators from U to V, the operation of linear space are defined by*

$$(A + B)(u) = A(u) + B(u).$$

Definition 160 *Let $A : U \to V$ be a linear operator, $\|\cdot\|_U$, $\|\cdot\|_V$ norms on U and U respectively. Then $\|A\|_{U\to V}$ is defined as*

$$\|A\|_{U\to V} = \inf_{u\in V, u\neq 0} \frac{\|Au\|_V}{\|u\|_U}.$$

Remark 161 *Equivalently, $\|A\|_{U\to V}$ is the least number L such that $\forall u \in U$, $\|Au\|_V \leq L\|u\|_U$.*

Exercise 162 *(*) If $A : U \to V$ and $B : U \to V$ are linear operators,*

$$\|A + B\|_{U\to V} \leq \|A\|_{U\to V} + \|B\|_{U\to V}.$$

Definition 163 *If $\|A\|_{U\to V} < \infty$, the operator A is called* bounded *operator (from U to V).*

Remark 164 *(*) If the space U (at least) is finite dimensional, one can show that all linear operators are bounded.*

Theorem 165 *Let $U = \mathbb{R}^m$, $V = \mathbb{R}^n$, $A \in \mathbb{R}^{n\times m}$, $\|\cdot\| = \|\cdot\|_\infty$. Then*

$$||A||_{U\to V} = \max_{i=1,\ldots,n} \sum_{j=1}^{m} |a_{ij}|.$$

Proof. We need to show two things
1. $L = \max_i \sum_j |a_{ij}|$implies that $\|Au\|_\infty \leq L\|u\|_\infty$
2. no smaller L has this property.

ad 1. Estimate

$$|(A_u)_i| = |\sum_j a_{ij}u_j| \leq \sum_j |a_{ij}||u_j| \leq (\sum_j |a_{ij}|) \max_j(u_j)$$

to conclude that

$$\|Au\|_\infty = \max_i |(Au)_i| \leq (\max_i \sum_j |a_{ij}|)\|u\|_\infty$$

ad 2. To find u such that $\|Au\|_\infty = L\|u_\infty\|$, let $|u_j| = 1$ pick the sign of u_i to make all $a_{ij}u_j \geq 0$, $u_j = -sign\, a_{ij}$. Then pick i_0 so that $\sum_j |a_{i_0 j}|$ is maximal, and get

$$\|Au\|_\infty = \max_i \sum_j |a_i u_j| \geq \sum_j |a_{i_0}u_j| = \sum_j |a_{i_0 j}|\|u\|_\infty.$$

■

15.4 Linear functionals

Definition 166 *Linear operators that map a normed linear space into scalars ($\mathbb{R}$ or $\mathbb{C}$), are called linear functionals. When they are bounded as linear operators following Definition 163, they are called* bounded linear functionals. *The space of all bounded linear functionals on V is denoted by V'. The norm of a linear functional $F : V \to \mathbb{R}$ is defined as the norm of a linear operator and denoted by the subscript V' :*

$$\|F\|_{V'} = \sup_{u \in V, u \neq 0} \frac{|F(u)|}{\|u\|_V}.$$

Example 167 *Let $V = C[a,b]$, $I(u) = \int_a^b u(x)dx$ Then*

$$|I(u)| = \left| \int_a^b u(x)dx \right| \leq (b-a) \max_{a \leq x \leq b} |u(x)|$$

with equality attained for constant function $u = 1$, so

$$\|I\|_{C[a,b]'} = b - a.$$

Example 168 *Let $V = C[a,b]$, $x_0, \ldots, x_n \in [a,b]$, $I_n(u) = \sum_{i=1}^n A_i u(x_i)$. Then*

$$|I_n(u)| = \left| \sum_{i=0}^n A_i u(x_i) \right| \leq \sum_{i=0}^n |A_i| \, |u(x_i)| \leq \sum_{i=0}^n |A_i| \max_{a \leq x \leq b} |u(x)| \,,$$

with equality attained when $u(x_i) = 1$ for $A_i \geq 0$ and $u(x_i) = -1$ for $A_i < 0$. Hence,

$$\|I_n\|_{C[a,b]'} = \sum_{i=0}^n |A_i| \,.$$

15.5 Inner product spaces

Definition 169 *An inner product space is a linear space V with a scalar function $\langle u, v \rangle$, defined for all $u, v \in V$, such that*

1. $\forall u, v, w \in V$, a, b *scalar* : $\langle w, au + bv \rangle = a \langle w, u \rangle + b \langle w, v \rangle$
2. $\langle u, v \rangle = \overline{\langle v, u \rangle}$
3. $\langle u, u \rangle \geq 0$, $\langle u, u \rangle = 0 \implies u = 0$

The complex conjugate in 2. has no effect in the real case. The condition 3. implies, in particular, that $\langle u, u \rangle$ is real.

Lemma 170 *(Cauchy-Schwarz inequality)*

$$|\langle u, v \rangle|^2 \leq \langle u, u \rangle \langle v, v \rangle$$

Lemma 171 *Let $\langle u, u \rangle$ be an inner product on V. Then $\|u\| = \sqrt{\langle u, u \rangle}$ defines a norm on V.*

Cauchy inequality can be then written as

$$|\langle u, v \rangle| \leq \|u\| \, \|v\|$$

The next lemma shows that Cauchy inequality is sharp.

Lemma 172

$$\|u\| = \sup_{v \in V} \frac{|\langle u, v\rangle|}{\|v\|}$$

Example 173 *Let $V = \mathbb{R}^n$, $\langle u, v\rangle = u^T v$. This is the familiar n-dimensional Euclidean space.*

Example 174 *Let $V = \mathbb{C}^n$, $\langle u, v\rangle = u^* v$. This is the n-dimensional complex Euclidean space.*

Example 175 *Let $V = \mathcal{P}_n$ (the space of all polynomials of degree at most n), and $\langle u, v\rangle = \int_a^b u(x)v(x)dx$, where $-\infty < a < b < +\infty$*

Definition 176 *If $\langle u, v\rangle = 0$, we say that the vectors u and v are* orthogonal *and write $u \perp v$.*

15.6 Gram-Schmidt orthogonalization

Definition 177 *A set of vectors $\{v_0, \ldots, v_n\}$ is* orthogonal *if $\langle v_i, v_j\rangle = 0$ if $i \neq j$, and $v_i \neq 0$, for all $i, j = 0, \ldots, n$.*

From any set of vectors $\{u_0, \ldots, u_n\}$, we can attempt to build an orthogonal set by taking each vector u_i in turn and making it orthogonal to all previous $u_0, \ldots, u_{i-1}$,by adding a correction in their span:

$$\begin{aligned} v_0 &= u_0 \\ v_1 &= u_1 - a_{10}v_0,\ a_{10} \text{ determined from } v_1 \perp v_0 \\ v_2 &= u_2 - a_{20}v_0 - a_{21}v_1,\ a_{20}, a_{21} \text{ determined from } v_2 \perp v_0,\ v_2 \perp v_1 \text{ using } v_1 \perp v_0 \end{aligned}$$

In general,

$$v_j = u_j - \sum_{i=0}^{n-1} a_{ji} v_i,$$

where the coefficient of the linear combinations are found from the conditions that $v_j \perp v_i$ for all $i = 0, \ldots, j-1$, using that the set $\{v_0, \ldots, v_{j-1}\}$ is orthogonal:

$$\begin{aligned} v_j = u_j - \sum_{i=0}^{n-1} a_{ji} v_i \quad &\Longrightarrow \quad 0 = \langle v_k, v_j\rangle = \langle v_k, u_j\rangle - \sum_{i=0}^{n-1} a_{ji} \langle v_k, v_i\rangle \\ &\Longrightarrow \quad a_{ji} = \frac{\langle v_i, u_j\rangle}{\langle v_i, v_i\rangle} \end{aligned}$$

because all products $\langle v_k, v_i\rangle$ are zero except possibly $\langle v_i, v_i\rangle$. When $\langle v_i, v_i\rangle = 0$,then $v_i = 0$,and so u_i is a linear combination of $\{v_0, \ldots, v_i\}$, hence of $\{u_0, \ldots, u_i\}$. We thus have

Theorem 178 (Gram-Schmitd orthogonalization) *Let $\{u_0, \ldots, u_n\}$ be a linearly independent set in an inner product space.Define*

$$v_j = u_j - \sum_{i=0}^{n-1} \frac{\langle v_i, u_j\rangle}{\langle v_i, v_i\rangle} v_i, \quad j = 0, \ldots, n$$

Then division by zero cannot occur, and $\{v_0, \ldots, v_n\}$ is an orthogonal set.

Gram-Schmidt orthogonalization by the formulas (178) is a *theoretical tool only*, because the algorithm is numerically unstable. To find the Gram-Schmidt orthogonalization of a set of vectors in $\mathbb{R}^n$ or $\mathbb{C}^n$,always use the QR algorigthm (Sec. **??**).

15.7 Orthogonal polynomials

15.8 Cauchy sequences and completeness

Definition 179 *A sequence $\{u_k\} \subset U$, U a normed linear space, is called Cauchy if* $\forall \varepsilon > 0 \exists N \forall k, l > N : \|a_k - a_l\| < \varepsilon$

Definition 180 *A normed linear space is called complete if every Cauchy sequence has a limit. A complete normed linear space is shortly called* Banach space.

Example 181 *$\mathbb{R}^n$, $C[a,b]$, $\mathbb{C}^n$ are Banach spaces.*

Definition 182 *A set $S \subset U$ is closed if $\{a_n\} \subset S$, $\|a_n - a\| \to 0$, implies $a \in S$.*

Theorem 183 *For every normed linear space U there exists a complete normed linear space V such that $U \subset V$, and*

$$\begin{array}{rcl} \forall u & \in & U : \|u\|_V = \|u\|_U \\ \forall v & \in & V \exists \{u_n\} \subset U : \lim_{n\to\infty} \|v - u_n\|_V \ ) \end{array}$$

In other words, V is a completion of U if

1. the norm on V is an extension of the norm on U
2. every element of V is the limit of a sequence from U.

Theorem 184 *Every finite dimensional normed linear space is complete.*

Example 185 *$\mathbb{R}$, as a linear space of dimension one, with the norm defined as the absolute value, is complete.*

Definition 186 *An inner product space that is complete is called* Hilbert space.

Theorem 187 *Every bounded linear operator from U to a complete normed linear space, $A : U \to W$, can be extended to a bounded linear operator $B : V \to W$ by*

$$Bu = \lim_{n\to\infty} Bu_n \text{ if } \lim_{n\to\infty} u_n = u, u_n \in U.$$

This extension is unique and does not depend on the selection of the sequence $u_n \to v$.Similarly, a bounded bilinear form $a(u,v)$ on U can be extended by the limit to a form on V,

$$a(u,v) = \lim_{n\to\infty} a(u_n, v_n), \lim_{n\to\infty} u_n = u, \lim_{n\to\infty} v_n = v, u_n, v_n \in U,$$

and this extension is unique. In particular, if U is an inner product space, the inner product can be extended to the completion V or U in a unique way, and V is a Hilbert space.

Theorem 188 (Riesz representation theorem) *For every bounded linear functional f on a Hilbert space V there exists a unique $g \in V$ such that*

$$\forall v \in V : f(v) = \langle g, v \rangle .$$

The mapping $f \mapsto g$ is linear.

That is, every bounded linear functional on a Hilbert space can be written as an inner product with some vector. In the finite dimensional case, this a known simple fact; in the general case, it is proved using completeness.

Let $V = \mathbb{R}^n$ with the Euclidean inner product, and F be a linear functional on V.Consider the basis $\{e_1, \ldots, e_n\}$ of V.Then for any $v = (v_1, \ldots, v_n)^T \in V$,

$$\begin{array}{rcl} F(v) & = & F(v_1 e_1 + \ldots + v_n e_n) \\ & = & v_1 F(e_1) + \ldots v_n F(e_n) \\ & = & \langle g, v \rangle , \quad g = (F(e_1), \ldots, F(e_n)). \end{array}$$

15.9 Linear difference equations

To find all solution of the difference equation of order m

$$a_m u_{n+1} + a_{m-1} u_n + \ldots + a_0 u_{n-m+1} = 0 \tag{15.3}$$

assume the form of solution $u_n = \lambda^n$.Substituting get the characteristic equation

$$a_m \lambda^m + \ldots + \lambda a_1 + a_0 = 0$$

The characteristic equation has roots (in general complex, and not necessarily distinct) $\lambda_1, \ldots, \lambda_m$. If the roots are distinct, the solutions of the difference equation are linear combinations of the powers λ_k^n,

$$u_n = A_1 \lambda_1^n + \ldots + A_m \lambda_m^n$$

In case of multiple root λ_k, the identical powers λ_k^n are replaced by $n\lambda_k^n, n(n-1)\lambda_k^n, \ldots$(the number of additional function is multiplicity of λ_k minus one, so the total number of distinct functions is preserved).

In other words, the m sequences λ_k^n, $n\lambda_k^n$, $n(n-1)\lambda_k^n \ldots$(for each root of the characteristic equation, the number of terms equals to its multiplicity), form a basis of the space of all sequences that satisfy the linear difference equation (15.3).

Chapter 16

Matrices

16.1 Determinants

Definition 189 *For a matrix $A \in \mathbb{R}^{n\times n}$ or $\mathbb{C}^{n\times n}$, the determinant $\det A$ is defined by induction:*

$$
\begin{array}{lll}
n=1 & : & \det A = |A| = \det(a_{11}) = a_{11} \\
n=2 & : & \det A = \det \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = a_{11}a_{22} - a_{12}a_{21} \\
\text{any } n & : & \forall i \;\; \det A = \displaystyle\sum_{j=1}^{n} (-1)^{i+j} a_{ij} \det A_{ij} \quad \textit{expansion by row } i \\
\text{any } n & : & \forall j \;\; \det A = \displaystyle\sum_{i=1}^{n} (-1)^{i+j} a_{ij} \det A_{ij} \quad \textit{expansion by column } j
\end{array}
$$

Here A_{ij} is the submatrix of the matrix A obtained by deleting the row i and column j.

Of course, for this definition to make sense, one has to prove that the value of the determinant does not depend on the choice of the row and column chosen for the expansion. This and alternative equivalent definitions can be found in any good textbook on linear algebra.

Example 190 *Let* $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$. *Expanding by column one,*

$$
\begin{array}{rcl}
\det A & = & a_{11} \det A_{11} - a_{21} \det A_{21} + a_{31} \det A_{31} \\
& = & 1 \det \begin{pmatrix} 5 & 6 \\ 8 & 9 \end{pmatrix} - 4 \det \begin{pmatrix} 2 & 3 \\ 8 & 9 \end{pmatrix} + 7 \det \begin{pmatrix} 2 & 3 \\ 5 & 6 \end{pmatrix}
\end{array}
$$

It helps to remember that the signs $(-1)^{i+j}$ in front of the subdeterminants form a checkerboard pattern:

$$
\begin{vmatrix}
+ & - & + & - & \dots \\
- & + & - & + & \dots \\
+ & - & + & - & \dots \\
\vdots & \vdots & \vdots & \vdots & \vdots
\end{vmatrix}
$$

16.2 Basic properties of determinants

Theorem 191
$A, B \in \mathbb{C}^{n\times n} \Rightarrow \det(AB) = \det A \det B$

Lemma 192 $A \in \mathbb{C}^{n\times n} \Rightarrow \det(A) = \det(A^T)$

Theorem 193 *The determinant of a triangular matrix equals to the product of the diagonal terms:*

$$\det \begin{pmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ \vdots & & \ddots & \\ a_{nn} & a_{nn} & \dots & a_{nn} \end{pmatrix} = a_1 \dots a_n$$

Proof. The proof is by repeated expansion over the first row. ■

In particular, the determinant of a diagonal matrix equals to the product of the diagonal terms.:

$$\det \begin{pmatrix} a_1 & & \\ & \ddots & \\ & & a_n \end{pmatrix} = a_1 \dots a_n$$

For block matrices, we have

Theorem 194 *The determinant of a block triangular matrix equals to the product of the determinants of the diagonal terms:*

$$\det \begin{pmatrix} A_{11} & & & \\ A_{21} & A_{22} & & \\ \vdots & & \ddots & \\ A_{nn} & A_{nn} & \dots & A_{nn} \end{pmatrix} = \det A_{11} \dots \det A_{nn}$$

Theorem 195 $\det A$ *is a polynomial of order* n *in the entries of* A*, that is, of the* n^2 *variables* a_{ij} *.*
Proof. *By induction over the order of the matrix and expansion.* ■

16.3 Regular matrices

Theorem 196 ***Definition 197*** *The* rank *of an* m *by* n *matrix* A *is the number of linearly independent colums. It is denoted by* $\mathrm{r}ank(A)$*.*

Theorem 198 $\mathrm{r}ank(A) = \mathrm{r}ank(A^T)$ *, that is, the number of linearly independent columns and the number of linearly independent rows is the same.*

Definition 199 *If* $A \in \mathbb{C}^{n\times n}$*, then the* inverse *of* A *is a matrix* B *such that* $AB = I$ *. We write* $A^{-1} = B$*.*

Definition 200 $A \in \mathbb{C}^{n\times n}$ *is called* regular *if* A^{-1} *exists,* singular *if not.*

Theorem 201 $AB = I \Leftrightarrow BA = I$*, and* A^{-1} *is unique.*

Theorem 202 *Matrix* $A \in \mathbb{C}^{n\times n}$ *is regular*

$$\begin{aligned} &\Leftrightarrow && \forall b \in \mathbb{C}^n : \exists x \in \mathbb{C}^n \quad Ax = b \\ &\Leftrightarrow && \forall b \in \mathbb{C}^n \quad x, y \in \mathbb{C}^n : (Ax = b \& Ay = b) \Rightarrow x = y \\ &\Leftrightarrow && \det A \neq 0 \\ &\Longleftrightarrow && \mathrm{r}ank(A) = n \end{aligned} \tag{16.1}$$

16.4 Eigenvalues and eigenvectors

Definition 203 *Let $A \in \mathbb{C}^{n\times n}$. The polynomial* $\det(\lambda I - A)$ *is* characteristic polynomial *of A*

Definition 204 *The trace of $A \in \mathbb{R}^{n\times n}$ or $\mathbb{C}^{n\times n}$ is the sum of its diagonal elements,* $\operatorname{tr} A = \sum_{i=1}^{n} a_{ii}$.

Theorem 205 $\det(\lambda I - A)$ *is polynomial in λ of degree n,*

$$\det(\lambda I - A) = a_n\lambda^n + a_{n-1}\lambda^{n-1} + \ldots + a_0$$

with

$$a_n = 1, \quad a_{n-1} = -\operatorname{tr} A, \quad a_0 = (-1)^n \det A$$

Proof. First $a_0 = \det(0I - A) = \det(-A) = (-1)^n \det A$. The rest of the proof is by induction. If $n = 1$, $\det(\lambda I - A) = \lambda - a_{11}$. Suppose we know that the theorem holds for matrices of order $n-1$. Then, for A of order n, expand by the first column and let the subscripts $_{ij}$ mean the submatrix obtained by omitting row i and column j :

$$\begin{aligned}
\det(\lambda I - A) &= (\lambda - a_{11})\det(\lambda I - A_{11}) + \sum_{i=2}^{n}(-1)^{i+1}a_{i1}\det(\lambda I - A)_{i1} \\
&= (\lambda - a_{11}) \underbrace{\left(\lambda^{n-1} - \lambda^{n-2}\operatorname{tr} A_{11} + \ldots\right)}_{=\det(\lambda I - A_{11}) \text{ from induction assumption}} + q(\lambda), \quad \deg(q) \le n-2 \\
&= \lambda^n - \lambda^{n-1}(a_{11} + \operatorname{tr} A_{11}) + \ldots
\end{aligned}$$

and $\operatorname{tr} A = a_{11} + \operatorname{tr} A_{11}$. ■

Definition 206 *If $A \in \Re^{n\times n}$ or $\mathcal{C}^{n\times n}$, then $\lambda \in \mathcal{C}$ is an eigenvalue and v is an eigenvector of A if $v \neq 0$ and $Av = \lambda v$.*

Lemma 207 *λ is eigenvalue of $A \Leftrightarrow (\lambda I - A)$ is singular $\Leftrightarrow det(\lambda I - A) = 0$*

Proof. λ is eigenvalue of $A \Leftrightarrow \exists v = \mathbb{C}^n, v \neq 0, (\lambda I - A)v = 0$ ■

Lemma 208 $\det(\lambda I - A) = (\lambda - \lambda_1)\ldots(\lambda - \lambda_n)$*, where $\lambda_1 \cdots \lambda_n$ are eigenvalues of A.*

Lemma 209 *Let λ be eigenvalue of $A \in \Re^{n\times n}$ on $\mathcal{C}^{n\times n}$ and $\|\cdot\|$ be a norm on $\Re^n (on \mathcal{C}^n)$, and $\|\cdot\|$ also denote the associated matrix norm. Then $\mid \lambda \mid \le \| A \|$.*

Proof. $Av = \lambda v, v \neq 0 \Rightarrow \mid \lambda \mid \| v \| = \| \lambda v \| = \| Av \| \le \| A \| \| v \| \Rightarrow \mid \lambda \mid \le \| A \|$ ■

Definition 210 *If square matrices A and B are related by*

$$B = M^{-1}AM$$

for some matrix M, then A and B are similar.

Replacing M by M^{-1} shows that if A and B are similar, then B and A are also similar. Also, if A and B are similar and B and C are similar, then $A$and C are similar.

Lemma 211 *Similar matrices have the same characteristic polynomial and the same eigenvalues. In addition, if v is an eigenvector of B for eigenvalue λ, then Mv is an eigenvector of B for the same eigenvalue λ.*

Proof.

$$\begin{aligned}
\det(\lambda I - M^{-1}AM) &= \det(M^{-1}(\lambda I - A)M) = \frac{1}{\det M}\det(\lambda I - A)\det M = \det(\lambda I - A) \\
Bv &= \lambda v \implies M^{-1}AMv = \lambda v \implies AMv = \lambda Mv
\end{aligned}$$

■

Proposition 212 *The trace of a matrix equals to the sum of its eigenvalues. In particular, the trace does not change under a similarity transformation. The determinant equals to* $(-1)^n$ *times the product of the eigenvalues.*

Proof. Multiply out the root factors of the characteristic polynomial,

$$\det(\lambda I - A) = (\lambda - \lambda_1)\dots(\lambda - \lambda_1) = \lambda^n - \lambda^{n-1}(\lambda_1 + \dots \lambda_n) + \dots + (-1)^n \lambda_1 \dots \lambda_n,$$

and compare with the coefficients of the polynomial from Theorem 205. ■

16.5 Symmetric positive definite matrices

Definition 213 *If* $A = A^T$*, A is called* symmetric*.*

Definition 214 $A \in \Re^{n\times n}$*,* $A^T = A$*. Then A is* positive definite *if* $x^t Ax > 0\ \forall x \in \Re^n, x \neq 0$*.*

Definition 215 *For* $A \in \mathbb{C}^{n\times n}, A^* = (\overline{A}^T) = (\overline{a_{ij}})$ *is the* conjugate transpose *of A.*

Definition 216 *If* $A = A^*$*, A is* Hermitian*.*

Example 217 *I is positive definite, because* $x \neq 0 \quad x^T Ix = x^T x =\| x \|^2 > 0$

Example 218 *If A is a regular matrix, then* $A^T A$ *is positive definite. Indeed, for* $x \neq 0, 0 <\| Ax \|^2 = (Ax)^T(Ax) = x^T A^T Ax$

Theorem 219 *If A is symmetric positive definite and* λ *is an eigenvalue of A, then* $\lambda > 0$*.*

Proof. $x \neq 0, Ax = \lambda x \Rightarrow x^T Ax = \lambda x^t x \Rightarrow \lambda = \frac{x^T Ax}{x^T x} > 0.$ ■

Definition 220 $R(x) = \frac{x^T Ax}{x^T x}$ *is called the* Rayleigh quotient*.*

The Rayleigh quotient plays an important role in numerical computation of eigenvalues of symmetric matrices.

Exercise 221 *If A is symmetric, then* $\nabla R(x) = 0$ *if and only if x is eigenvector and* $R(x)$ *an eigenvalue of A.*

Exercise 222 *If A is an* $n \times n$ *matrix and* λ *an eigenvalue of A, then* $c + \lambda.$ *is an eigenvalue of* $cI + A$*.*

Theorem 223 *(Polynomial mapping) If A is an* $n \times n$ *matrix and p a polynomial, then all eigenvalues of* $p(A)$ *are exactly* $p(\lambda)$ *, where* λ *are all eigenvalues of A.*

Theorem 224 *If* $A \in \mathbb{C}^{n\times n}$ *is symmetric, then there exists an orthonormal basis of* $\mathbb{C}^{n\times n}$*, which consists of eigenvectors of A, and all eigenvalues of A are real.*

If $u_1, \dots, u_n$ is the orthonormal basis of eigenvectors, we have $A[u_1, \dots, u_n] = [\lambda_1 u_1, \dots, \lambda_n u_n]$ and $u_i^T u_j = \delta_{ij}$, which is the same as

$$AU = U\Lambda, \qquad U = [u_1, \dots, u_n], \qquad \Lambda = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}, \qquad U^T U = I.$$

16.6 Principal minors of symmetric, positive definite matrices

Lemma 225 *If A is s.p.d., then every principal submatrix A is s.p.d.*

Proof. Let A_{11}be a principal submatrix of A :

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \qquad x = \begin{pmatrix} x_1 \\ 0 \end{pmatrix}$$

Then

$$x_1 \neq 0 \implies 0 \leq x^T A x = \begin{pmatrix} x_1 \\ 0 \end{pmatrix}^T \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ 0 \end{pmatrix} = x_1^T A_{11} x_1$$

$\Rightarrow A_{11}$ is positive definite
■

Lemma 226 *If A is s.p.d., then* $\det A > 0$

Proof. Define $f(t) = \det((1-t)I + tA)$
For $t \in [0,1]$, $(1-t)I + tA$ is s.p.d. because

$$x \neq 0 \implies x^T((1-t)I + tA)x = (1-t)x^T x + t x^T A x > 0.$$

So $f(t) \neq 0$ for $t \in [0,1]$, $f(0) = 1$, and $f(1) = \det A$. Because f is continuous, $f(1) > 0$. ■

Corollary 227 *A is s.p.d. $\Rightarrow$ all principal minors of A are positive.*

16.7 Unitary matrices

Definition 228 *A matrix $U \in \mathbb{C}^{m \times n}$ is called unitary if $U^*U = I$.*

Remark 229 *The definition is written in the complex case, but of couse all holds in the real case and the traspose ^{T} instead of the complex conjugate transpose *.*

Proposition 230 *A matrix is unitary if and only if its columns are orthogonal.*

Proposition 231 *Transformation by a unitary matrix preserves the Euclidean norm: If $U^*U = I$*

$$\|Ux\|^2 = (Ux)^* (Ux) = x^* U^* U x = x^* x = \|x\|^2,$$

so the spectral norm of a unitary matrix is one.

Proposition 232 *The product of two unitary matrices is unitary: If $U^*U = I$ and $V^*V = I$, then*

$$(UV)^*(UV) = V^* \underbrace{U^*U}_{I} V = V^*V = I$$

Proposition 233 *All eigenvalues of a square unitary matrix have absolute value one:*

$$Ux = \lambda x \implies \underbrace{\|Ux\|}_{=\|x\|} = |\lambda| \, \|x\| \implies |\lambda| = 1$$

Bibliography

[AS92] Milton Abramowitz and Irene A. Stegun, editors. *Handbook of mathematical functions with formulas, graphs, and mathematical tables*. Dover Publications Inc., New York, 1992. Reprint of the 1972 edition.

[BS02] Susanne C. Brenner and L. Ridgway Scott. *The mathematical theory of finite element methods*, volume 15 of *Texts in Applied Mathematics*. Springer-Verlag, New York, second edition, 2002.

[CL91] P. G. Ciarlet and J.-L. Lions, editors. *Handbook of numerical analysis. Vol. II*. Handbook of Numerical Analysis, II. North-Holland, Amsterdam, 1991. Finite element methods. Part 1.

[DR84] Philip J. Davis and Philip Rabinowitz. *Methods of numerical integration*. Computer Science and Applied Mathematics. Academic Press Inc., Orlando, FL, second edition, 1984.

[EM95] Alan Edelman and H. Murakami. Polynomial roots from companion matrix eigenvalues. *Math. Comp.*, 64(210):763–776, 1995.

[GG00] Walter Gander and Walter Gautschi. Adaptive quadrature—revisited. *BIT*, 40(1):84–101, 2000.

[GVL96] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.

[Hug00] Thomas J. R. Hughes. *The Finite Element Method*. Dover, 2000. Linear Static and Dynamic Finite Element Analysis. Reprint 1987 Prentice Hall edition.

[Joh87] Claes Johnson. *Numerical solution of partial differential equations by the finite element method*. Cambridge University Press, Cambridge, 1987.

[KC02] David Kincaid and Ward Cheney. *Numerical analysis: Mathematics of scientific computing*. Brooks/Cole Publishing Co., Pacific Grove, CA, third edition, 2002.

[Kre98] Rainer Kress. *Numerical analysis*, volume 181 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1998.

[Ove01] Michael L. Overton. *Numerical computing with IEEE floating point arithmetic*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2001.

[SH96] A. M. Stuart and A. R. Humphries. *Dynamical systems and numerical analysis*, volume 2 of *Cambridge Monographs on Applied and Computational Mathematics*. Cambridge University Press, Cambridge, 1996.

[SR97] Lawrence F. Shampine and Mark W. Reichelt. The MATLAB ODE suite. *SIAM J. Sci. Comput.*, 18(1):1–22, 1997. Dedicated to C. William Gear on the occasion of his 60th birthday.