ROYAL INSTITUTE
OF TECHNOLOGY

Numerical Linear Algebra

Bärbel Janssen

Computational Technology Laboratory
KTH Royal Institute of Technology, Stockholm, Sweden

October 13, 2014

Overview

About myself

- Computational mathematical modeling with differential equations.
- High Performance distributed methods.
- Adaptive finite element methods.
- Focus on turbulent flow and fluid-structure interaction.
- Applied projects in aerodynamics, aero-acoustics, biomedicine and geophysics.

About this course

Schedule

Today: Introduction, discussion of prestudy results, condition, stability, direct methods.

Tuesday: Iterative methods.

Wednesday: Methods for eigenvalue problems.

Thursady: Multigrid methods.

Friday: Outlook to methods for nonlinear problems.

About this course

Schedule

Today: Introduction, discussion of prestudy results, condition, stability, direct methods.

Tuesday: Iterative methods.

Wednesday: Methods for eigenvalue problems.

Thursady: Multigrid methods.

Friday: Outlook to methods for nonlinear problems.

About this course

Schedule

Today: Introduction, discussion of prestudy results, condition, stability, direct methods.

Tuesday: Iterative methods.

Wednesday: Methods for eigenvalue problems.

Thursady: Multigrid methods.

Friday: Outlook to methods for nonlinear problems.

About this course

Schedule

Today: Introduction, discussion of prestudy results, condition, stability, direct methods.
Tuesday: Iterative methods.
Wednesday: Methods for eigenvalue problems.
Thursady: Multigrid methods.
Friday: Outlook to methods for nonlinear problems.

About this course

Schedule

Today: Introduction, discussion of prestudy results, condition, stability, direct methods.
Tuesday: Iterative methods.
Wednesday: Methods for eigenvalue problems.
Thursady: Multigrid methods.
Friday: Outlook to methods for nonlinear problems.

Overview

First exercise

The fibonacci numbers are defined as

$$\text{fib}(n) = \begin{cases} 0, & \text{for } n = 0, \\ 1, & \text{for } n = 1, \\ \text{fib}(n-1) + \text{fib}(n-2), & \text{for } n > 1. \end{cases}$$

Write a program in MATLAB to compute fib(25). Going through the following steps might help you if you are not yet so familiar with MATLAB.

1. Click on file $\rightarrow$ new $\rightarrow$ m-file. This opens a new m-file. In this file we can now write the instructions for the algorithm. But before we do that, name the file `fib.m` and open it in an editor.

The syntax should look like:

```
function result = fib(n)
  if(n == 0)
    result = ...
  elseif(n == 1)
    result = ...
  else
    result = ...
  end
```

Fill in the blanks ... and save the file. Now you can call the function in the directory in which you saved the file form the command line in MATLAB:

```
>> fib(25)
```

Second exercise

copy the following function in a m-file and save it with the name of the function.

```
function result = fun(x)
tmp = x + sin(x);
result = tmp.^2;
```

1. Which result is computed by this function?
2. Plot the function on the interval $I = [0, 1]$ at 10 equidistant points. This can be achieved in MATLAB by the instruction `v = 0 : 0.1 : 1;` and calling `plot (v, fun(v), 'r-');`
3. Functions can have more than one parameter for input and output, e.g.:
   `function[res1, res2] = fun2(x,A,c)`

Third exercise

Compute the value of the polynomial

$$p(x) = 0.01x^3 + 5.7x - 45$$

at the points $x = -3, -1, 1, 3, 5$. Which predefined functions within MATLAB can be used? Consider MATLAB help.

Forth exercise

Write a program in MATLAB to approximate the exponential function

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}, \quad x \in \mathbb{R}$$

by the Taylor sum

$$T_n(x) = \sum_{k=0}^{n} \frac{x^k}{k!}, \quad x \in \mathbb{R}, n \in \mathbb{N}.$$

Plot the relative error for $n \in [0, 20]$ and points $x \in \{10, 1, -1, -10\}$. What do you observe?

Overview

First exercise

Write a program in MATLAB to estimate the machine precision ε. It is by definition the smallest floating-point number with the property that

$$1 + \varepsilon > 1.$$

Second exercise

In a textbook from 1988, I found the following interesting exercise. Let us evaluate the expression

$$f(x) = \frac{(x+1)^2 - 1}{x} \tag{1}$$

at $x = .1, .01, .001, \ldots$ using a computer program on an IBM 370. The results are reported in table 1. Expanding the expression in eq. (1) we get

$$f(x) = \frac{(x+1)^2 - 1}{x} = \frac{x^2 + 2x + 1 - 1}{x} = x + 2.$$

x	Computed $f(x)$	Correct $f(x)$
10^{-1}	2.1000	2.1000
10^{-2}	2.0098	2.0100
10^{-3}	1.9999	2.0010
10^{-4}	1.9836	2.0001
10^{-5}	1.9073	2.0000
10^{-6}	1.9073	2.0000
10^{-7}	0.0000	2.0000

Table: Value of $f(x)$ computed.

Write a program in MATLAB to compute the values in eq. (1) and compare your results to the reported results. What do you abserve? How can you explain your observations?

Third exercise

Archimedes approximated π by calculating the perimeters of polygons inscribing and circumscribing a circle, starting with hexagons, and successively doubling the number of sides. Two forms of the recurrence formula for the circumscribed polygon are:

$$t_0 = \frac{1}{\sqrt{3}}, \quad t_{i+1} = \frac{\sqrt{t_i^2+1}-1}{t_i} \quad \text{first form,}$$

$$t_{i+1} = \frac{t_i}{\sqrt{t_i^2+1}+1} \quad \text{second form.}$$

The value of π is approximated by

$$\pi \approx 6 \cdot 2^i \cdot t_i.$$

Write a MATLAB program for the approximation of π to compare both forms. Hint: Use the instruction `format long` in MATLAB to show more digits.

Overview

First exercise

We are interested in solving the following problem

$$x = e^{-x}. \tag{2}$$

Which different ways to solve (2) come to your mind? What are the differences between those solution methods? What do they have in common?

Second exercise

Let us solve

$$x = e^{-x}.$$

numerically. We approximate the solution by a sequence $x_1, x_2, \ldots$ using the rule

$$x_{k+1} = e^{-x_k}. \tag{3}$$

Write a MATLAB program which implements (3). Think of a suitable start value for x_0. Print the values of k and x_k in a table. What do you observe? How accurate is the approximation? How accurate could it be?

Second exercise

The equation

$$x = e^{-x}.$$

can equivalently be written as

$$x = -\log x.$$

Formulate a similar approximation as (3). Write a program in MATLAB which implements your approximation. What is a good start value in this implementation? Print the values for k and x_k in a table again. Compare the result to the results you got before with iteration (3).

Third exercise

What is a good stopping criterion for the iterations you implemented? What is a good stopping criterion in general?

Overview

First exercise

The unit sphere with respect to a vector norm $\|\cdot\|$ on $\mathbb{R}^n$ is defined by

$$S := \left\{x \in \mathbb{R}^n : \|x\| = 1\right\}.$$

Sketch the spheres in $\mathbb{R}^2$ corresponding to the l_1-norm, the euclidian norm and the l_∞-norm:

$$\|x\|_1 = |x_1| + |x_2|, \quad \|x\|_2 = \sqrt{|x_1|^2 + |x_2|^2}, \quad \|x\|_\infty = \max\left\{|x_1|, |x_2|\right\}.$$

How do the unit spheres corresponding to the general l_p-norms look like?

Second exercise

Write a program in MATLAB which implements the Gram-Schmidt algorithm for orthonormalizing a set of linearly independent vectors.

1. Apply your program to the following vectors:

$$v_1 = \begin{pmatrix} 1 \\ -2 \\ 1 \\ 1 \\ -4 \end{pmatrix}, v_2 = \begin{pmatrix} -9 \\ 4 \\ 4 \\ 1 \\ 9 \end{pmatrix}, v_3 = \begin{pmatrix} 6 \\ 1 \\ -5 \\ -2 \\ 1 \end{pmatrix}, v_4 = \begin{pmatrix} -8 \\ 3 \\ 2 \\ 3 \\ 9 \end{pmatrix}, v_5 = \begin{pmatrix} -6 \\ 2 \\ 3 \\ 1 \\ 5 \end{pmatrix}$$

   Make sure they are linearly independent before you try to apply the Gram-Schmidt algorithm.

2. Check the orthogonality after you applied the algorithm for each pair of vectors:

$$(w_i, w_j) \overset{??}{=} 0 \quad \text{for } i \neq j, \quad i, j = 1, \ldots, 5,$$

   where w_i, $i = 1, \ldots, 5$ are the vectors after orthonormalizing.

Third exercise

For the matrices

$$A_1 = \begin{pmatrix} 2 & -1 & 2 \\ 1 & 2 & -2 \\ 2 & 2 & 2 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 5 & 5 & 0 \\ -1 & 5 & 4 \\ 2 & 3 & 8 \end{pmatrix},$$

compute

1. $\|A_1\|_1, \|A_2\|_1$,
2. $\|A_1\|_2, \|A_2\|_2$,
3. $\|A_1\|_\infty, \|A_2\|_\infty$.

Forth exercise

Given the matrix

$$A = \begin{pmatrix} 1 & 0.1 & -0.2 \\ 0 & 2 & 0.4 \\ -0.2 & 0 & 3 \end{pmatrix},$$

1. apply the Gerschgorin-Hadamard theorem to obtain the Gerschgorin circles,
2. draw a sketch of the circles you got in 1,
3. derive the eigenvalues of A.

Fifth exercise

Consider the linear system

$$\begin{aligned} .550x_1 + .423x_2 &= .127 \\ .484x_1 + .372x_2 &= .112, \end{aligned}$$

which corresponds to

$$Ax = b \quad \text{with} \quad A = \begin{pmatrix} .550 & .423 \\ .484 & .372 \end{pmatrix} \quad \text{and} \quad b = \begin{pmatrix} .127 \\ .112 \end{pmatrix}.$$

Suppose that you are given two candidate solutions,

$$\tilde{x} = \begin{pmatrix} 1.7 \\ -1.91 \end{pmatrix} \quad \text{and} \quad \bar{x} = \begin{pmatrix} 1.01 \\ -.99 \end{pmatrix}.$$

1. Decide depending on the residual $b - Ax$ which of the two candidates is a “better” solution.
2. Compute the exact solution.
3. Compute ther erros to the exact solution:

$$\|\tilde{x} - x\|_\infty, \quad \|\bar{x} - x\|_\infty.$$

Overview

First exercise

1. Implement a backward and a forward solve of lower left and upper right triangular systems in MATLAB.
2. Implement the Gaussian elimination in MATLAB. A description of the algorithm can be found in the lecture notes, see chapter 3.2. Use the implementation of the backward and forward solve.

Second exercise

Apply your algorithm to the following problem

$$\begin{aligned} 10^{-16}x_1 + x_2 &= 1, \\ x_1 + x_2 &= 2. \end{aligned}$$

Compare your result to the exact result. Where does the discrepancy come from?

Third exercise

Implement the pivoting strategy into your Gaussian elimination. Apply the pivoted algorithm to the system

$$\begin{aligned} 10^{-16}x_1 + x_2 &= 1, \\ x_1 + x_2 &= 2. \end{aligned}$$

Overview

Errors

Numerical methods require use of computers and therefore precision is limited. The methods we use have to be analyzed in view of the finite precision.

Absolute error

Let $\tilde{x}$ be an approximation to x. The absolute error e is defined as

$$e = |x - \tilde{x}|, \quad x \in \mathbb{R}, \quad e = \|x - \tilde{x}\|, \quad x \in \mathbb{R}^n.$$

The absolute error e might be big just because $|x|$ or $\|x\|$ is big. This gives rise to the following definition.

Relative error

For $x \neq 0$, we define the relative error between x and its approximation $\tilde{x}$ as

$$e_R = \frac{|x - \tilde{x}|}{|x|}, \quad x \in \mathbb{R}, \quad e_R = \frac{\|x - \tilde{x}\|}{\|x\|}, \quad x \in \mathbb{R}^n.$$

Errors may have different reasons. But one of them is the already mentioned finite precision. This is connected with the computer's represenatation of numbers.

Floating-point representation

In scientific computuations, floating-point numbers are typically used. The three digit floating-point representation of $\pi = 3.1415926\ldots$ is

$$+.314 \times 10^1.$$

Decimal floating-point number

The fraction .314 is called the mantissa, 10 is called the base, and 1 is called the exponent. Generally, n-digit decimal floating-point numbers have the form

$$\pm .d_1 d_2 \cdots d_n \times 10^p,$$

where the digits d_1 through d_n are integers between 0 and 9 and d_1 is never zero except in the case $d_1 = d_2 = \cdots = d_n = 0$.

Arithmetic operations an cancellation

Errors in finite arithmetic occur in

- representing numbers,
- performing arithmetic operations.

The result after an operation may be not representable even if the input numbers both were representable. The result needs to be rounded or truncated to be representable.
The result of a floating point operation satifies

$$\mathrm{fl}(a \,\mathrm{op}\, b) = (a \,\mathrm{op}\, b)(1+\delta)$$

where op is one of the basic operations $+, -, \cdot\, :$. Usually, properties of exact mathematic operations are not valid in arithmetic operations as

$$\mathrm{fl}(\mathrm{fl}(x+y)+z) \neq \mathrm{fl}(x+\mathrm{fl}(y+z)).$$

Arithmetic operations an cancellation

Errors in finite arithmetic occur in

- representing numbers,
- performing arithmetic operations.

The result after an operation may be not representable even if the input numbers both were representable. The result needs to be rounded or truncated to be representable.

The result of a floating point operation satifies

$$\mathrm{fl}(a \operatorname{op} b) = (a \operatorname{op} b)(1+\delta)$$

where op is one of the basic operations $+, -, \cdot :$. Usually, properties of exact mathematic operations are not valid in arithmetic operations as

$$\mathrm{fl}(\mathrm{fl}(x+y)+z) \neq \mathrm{fl}(x+\mathrm{fl}(y+z)).$$

Arithmetic operations an cancellation

Errors in finite arithmetic occur in

- representing numbers,
- performing arithmetic operations.

The result after an operation may be not representable even if the input numbers both were representable. The result needs to be rounded or truncated to be representable.
The result of a floating point operation satifies

$$\mathrm{fl}(a \,\mathrm{op}\, b) = (a \,\mathrm{op}\, b)(1+\delta)$$

where op is one of the basic operations $+, -, \cdot :$. Usually, properties of exact mathematic operations are not valid in arithmetic operations as

$$\mathrm{fl}(\mathrm{fl}(x+y)+z) \neq \mathrm{fl}(x+\mathrm{fl}(y+z)).$$

Example for loss of precision

Since the convergence radius of the exponential series is infinity which means it converges for all numbers $x \in \mathbb{R}$ the power-series expansion

$$\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{1}{2}x^2 + \frac{1}{3!}x^3 + \cdots \tag{4}$$

is a good candidate for computations.

However, for negative values of x, the convergence is very bad with a high relative error.

Reason

When the expansion eq. (4) is applied to any negative argument, the terms alternate in sign. Subtraction is required to compute the approximation. The large relative error is caused by rounding in combination with subtraction.

Example for loss of precision

Since the convergence radius of the exponential series is infinity which means it converges for all numbers $x \in \mathbb{R}$ the power-series expansion

$$\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{1}{2}x^2 + \frac{1}{3!}x^3 + \cdots \qquad (4)$$

is a good candidate for computations.
However, for negative values of x, the convergence is very bad with a high relative error.

Reason

When the expansion eq. (4) is applied to any negative argument, the terms alternate in sign. Subtraction is required to compute the approximation. The large relative error is caused by rounding in combination with subtraction.

Example for loss of precision

Since the convergence radius of the exponential series is infinity which means it converges for all numbers $x \in \mathbb{R}$ the power-series expansion

$$\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{1}{2}x^2 + \frac{1}{3!}x^3 + \cdots \tag{4}$$

is a good candidate for computations.
However, for negative values of x, the convergence is very bad with a high relative error.

Reason

When the expansion eq. (4) is applied to any negative argument, the terms alternate in sign. Subtraction is required to compute the approximation. The large relative error is caused by rounding in combination with subtraction.

Example for loss of precision

Since the convergence radius of the exponential series is infinity which means it converges for all numbers $x \in \mathbb{R}$ the power-series expansion

$$\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{1}{2}x^2 + \frac{1}{3!}x^3 + \cdots \tag{4}$$

is a good candidate for computations.
However, for negative values of x, the convergence is very bad with a high relative error.

Reason

When the expansion eq. (4) is applied to any negative argument, the terms alternate in sign. Subtraction is required to compute the approximation. The large relative error is caused by rounding in combination with subtraction.

Conditioning

Condition

A problem is well conditioned if small changes in problem parameters produce small changes in the outcome.

Remark

The property to be well or ill conditioned is a property of the problem itself. It does not have to do anything with an algorithm applied to solve a certain problem. Especially, it is independent of computation and rounding errors.

Example

Let us consider the problem of determining the roots of the polynomial

$$(x-1)^4 = 0, \tag{5}$$

whose four routs are all exactly equal to 1. Now we suppose that the right-hand side is perturbed by 10^{-8} such that the perturbed problem of eq. (5) now reads

$$(x-1)^4 = 10^{-8}, \tag{6}$$

and it can be equivalently written as

$$\begin{aligned}&(x-1)^4 - 10^{-8} = \\ &\left(x - \frac{99}{100}\right)\left(x - \frac{101}{100}\right)\left(x - 1 + \frac{i}{100}\right)\left(x - 1 - \frac{i}{100}\right),\end{aligned}$$

where $i \in \mathbb{C}$ is the imaginary number such that $i^2 = -1$.

- That means one root has changed by 10^{-2} compared to the root of eq. (5).
- The change in the solution is six orders larger in comparison to the size of the perturbation.

So we say that this problem is ill conditioned.

Stability

Stability

An algorithm is stable if small changes in algorithm parameters have a small effect on the algorithm's output.

An unstable algorithm

Example

Consider the example of solving the equation

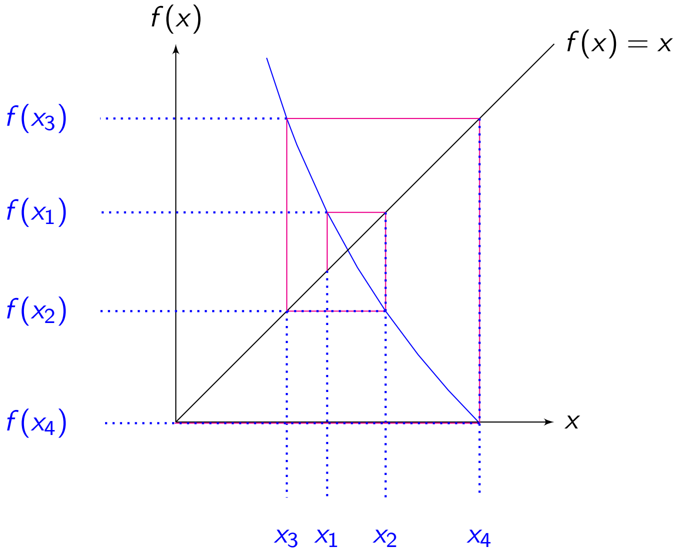
$$x = e^{-x} \quad \text{or equivalently} \quad x = -\ln(x).$$

The iteration procedure

$$x_{k+1} = -\ln(x_k) \tag{7}$$

produces approximations which diverge from the root except if you choose the solution x as the starting guess $x_1 = x$. This algorithm is unstable since each iteration amplifies the error.

Iteration cobweb for $f(x) = -\ln(x), x_1 = 0.5$



An stable algorithm

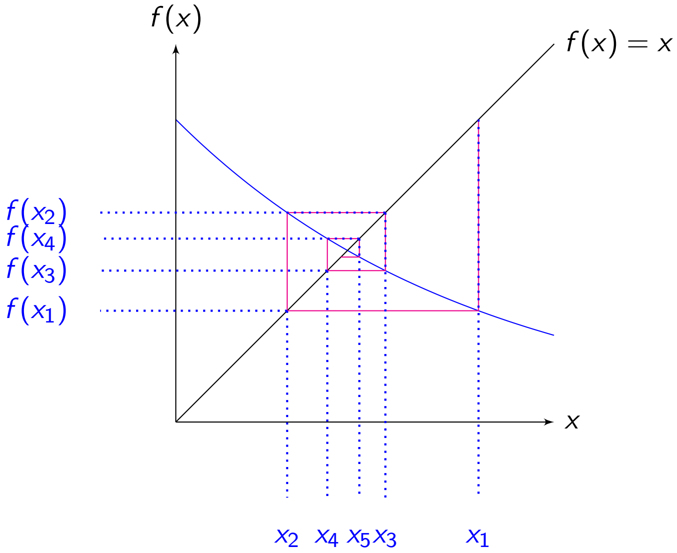
Example

As contrast to the unstable iteration $x_{k+1} = -\ln(x_k)$ let us look at a stable iteration which is given as

$$x_{k+1} = \exp(-x_k). \tag{8}$$

Beginning with the starting value $x_1 = 1$, this algorithm produces convergent iterates to the root.

Iteration cobweb for $f(x) = \exp(-x), x_1 = 1$



Overview

Good to know

Direct methods

- are known to be very robust.
- are well applicable to moderate sized problems.
- become infeasible for large systems due to their usually high storage requirements and high computational complexity.

Today, moderate size systems are of dimensions up to $n \approx 10^5 - 10^6$.

Triangular linear systems

We will consider to solve systems of the form

$$Ax = b$$

with a quadratic matrix A and vectors x and b of suitable sizes.

Systems which have a triangular form are particularly easy to solve. In the case of an upper triangular system, the matrix A has the coefficient matrix

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ & a_{22} & \dots & a_{2n} \\ & & \ddots & \vdots \\ & & & a_{nn} \end{pmatrix}$$

and the coresponding linear system looks like

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & = b_1 \\ & & a_{22}x_2 & + & \dots & + & a_{2n}x_n & = b_2 \\ & & & & & \ddots & \vdots & \vdots \\ & & & & & & a_{nn}x_n & = b_n \end{array}$$

Triangular linear systems

We will consider to solve systems of the form

$$Ax = b$$

with a quadratic matrix A and vectors x and b of suitable sizes. Systems which have a triangular form are particularly easy to solve. In the case of an upper triangular system, the matrix A has the coefficient matrix

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ & a_{22} & \dots & a_{2n} \\ & & \ddots & \vdots \\ & & & a_{nn} \end{pmatrix}$$

and the coresponding linear system looks like

$$\begin{array}{ccccccccl} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & = b_1 \\ & & a_{22}x_2 & + & \dots & + & a_{2n}x_n & = b_2 \\ & & & & & \ddots & \vdots & \vdots \\ & & & & & & a_{nn}x_n & = b_n \end{array}$$

Triangular linear systems

We will consider to solve systems of the form

$$Ax = b$$

with a quadratic matrix A and vectors x and b of suitable sizes. Systems which have a triangular form are particularly easy to solve. In the case of an upper triangular system, the matrix A has the coefficient matrix

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ & a_{22} & \dots & a_{2n} \\ & & \ddots & \vdots \\ & & & a_{nn} \end{pmatrix}$$

and the coresponding linear system looks like

$$\begin{array}{ccccccccl} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & = b_1 \\ & & a_{22}x_2 & + & \dots & + & a_{2n}x_n & = b_2 \\ & & & & & \ddots & & \vdots \\ & & & & & & a_{nn}x_n & = b_n \end{array}.$$

1. For $a_{nn} \neq 0$, we obtain $x_n = \frac{b_n}{a_{nn}}$ and plug this into the second last equation to compute x_{n-1}.
2. If we progress in that manner and in case $a_{jj} \neq 0, j = 1, \ldots, n$, we derive the solution as

$$x_n = \frac{b_n}{a_{nn}}, \quad x_j = \frac{1}{a_{jj}} \left(\sum_{k=j+1}^{n} a_{jk} x_k \right), \qquad j = n-1, \ldots 1.$$

Remark

The same operations to obtain a solution can be applied to a lower triangular system in opposite order.

Example

$$\begin{array}{rcrcrcrr} x_1 & + & x_2 & + & 2x_3 & = & & 3 \\ & & x_2 & - & 3x_3 & = & - & 4\,. \\ & & & & -19x_3 & = & - & 19 \end{array}$$

In the first step we obtain $x_3 = 1$. After inserting it in the second equation we derive

$$\begin{array}{rcrcrcrr} x_1 & + & x_2 & + & 2 & = & & 3 \\ & & x_2 & - & 3 & = & - & 4 \\ & & & & x_3 & = & & 1 \end{array}$$

and find that $x_2 = -1$. In the last step we solve the system and get

$$\begin{array}{rrrcrr} x_1 & & & = & & 2 \\ & x_2 & & = & - & 1. \\ & & x_3 & = & & 1 \end{array}$$

Example

$$\begin{array}{rcrcrcrr} x_1 & + & x_2 & + & 2x_3 & = & & 3 \\ & & x_2 & - & 3x_3 & = & - & 4\,. \\ & & & & -19x_3 & = & - & 19 \end{array}$$

In the first step we obtain $x_3 = 1$. After inserting it in the second equation we derive

$$\begin{array}{rcrcrcrr} x_1 & + & x_2 & + & 2 & = & & 3 \\ & & x_2 & - & 3 & = & - & 4 \\ & & & & x_3 & = & & 1 \end{array}$$

and find that $x_2 = -1$. In the last step we solve the system and get

$$\begin{array}{rrrcrr} x_1 & & & = & & 2 \\ & x_2 & & = & - & 1. \\ & & x_3 & = & & 1 \end{array}$$

Example

$$\begin{array}{rcrcrcrr} x_1 & + & x_2 & + & 2x_3 & = & & 3 \\ & & x_2 & - & 3x_3 & = & - & 4\,. \\ & & & & -19x_3 & = & - & 19 \end{array}$$

In the first step we obtain $x_3 = 1$. After inserting it in the second equation we derive

$$\begin{array}{rcrcrcrr} x_1 & + & x_2 & + & 2 & = & & 3 \\ & & x_2 & - & 3 & = & - & 4 \\ & & & & x_3 & = & & 1 \end{array}$$

and find that $x_2 = -1$. In the last step we solve the system and get

$$\begin{array}{rrrcrr} x_1 & & & = & & 2 \\ & x_2 & & = & - & 1. \\ & & x_3 & = & & 1 \end{array}$$

Gaussian elimination

The elimination method of Gauss

1. transforms the system $Ax = b$ in several elimination steps into an upper triangular form,
2. and then applies the backward solve procedure just described.

Only elimination steps that do not change the solution are allowed. These are the following:

- permutation of two rows of the matrix,
- permutation of two colums of the matrix, inlcuding according renumbering of the unkowns x_i,
- addition of a scalar multiple of a row to another row of the matrix.

Gaussian elimination

The elimination method of Gauss

1. transforms the system $Ax = b$ in several elimination steps into an upper triangular form,
2. and then applies the backward solve procedure just described.

Only elimination steps that do not change the solution are allowed. These are the following:

- permutation of two rows of the matrix,
- permutation of two colums of the matrix, inlcuding according renumbering of the unkowns x_i,
- addition of a scalar multiple of a row to another row of the matrix.

Algorithm for Gaussian elimination

Since we assumed the matrix to be regular, there exists an element $a_{r1}^{(0)} \neq 0, \quad 1 \leq r \leq n$.

First step

Permute the first and the rth row. We call the result $\left(\tilde{A}^{(0)}, \tilde{b}^{(0)}\right)$.

Each remaining row subtracts $q_{j1} = \tilde{a}_{j1}^{(0)} / \tilde{a}_{11}^{(0)}$ times the first row.

Algorithm for Gaussian elimination

Since we assumed the matrix to be regular, there exists an element $a_{r1}^{(0)} \neq 0, \quad 1 \leq r \leq n$.

First step

Permute the first and the rth row. We call the result $\left(\tilde{A}^{(0)}, \tilde{b}^{(0)}\right)$.

Each remaining row subtracts $q_{j1} = \tilde{a}_{j1}^{(0)} / \tilde{a}_{11}^{(0)}$ times the first row.

We arrive at

$$\left(A^{(1)},\ b^{(1)}\right) = \left(\begin{array}{cccc|c} \tilde{a}_{11}^{(0)} & \tilde{a}_{12}^{(0)} & \cdots & \tilde{a}_{1n}^{(0)} & \tilde{b}_1^{(0)} \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} & b_2^{(1)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} & b_n^{(1)} \end{array}\right),$$

where the new entries are given as

$$a_{j1}^{(1)} = \tilde{a}_{ji}^{(0)} - q_{j1}\tilde{a}_{1i}^{(0)}, \quad \tilde{b}_j^{(0)} - q_{j1}\tilde{b}_1^{(0)}, \quad 2 \leq i, j \leq n.$$

Next step

The submatrix $(a_{jk}^{(1)})$ is regular and we can repeat the same steps which we applied to the matrix $A^{(0)}$

After $n-1$ of these elimination steps we arrive at the desired upper triangular form

$$\left(A^{(n-1)},\ b^{(n-1)}\right) = \left(\begin{array}{ccccc|c} \tilde{a}_{11}^{(0)} & \tilde{a}_{12}^{(0)} & \tilde{a}_{13}^{(0)} & \cdots & \tilde{a}_{1n}^{(0)} & \tilde{b}_{1}^{(0)} \\ 0 & \tilde{a}_{22}^{(1)} & \tilde{a}_{23}^{(1)} & \cdots & \tilde{a}_{2n}^{(1)} & \tilde{b}_{2}^{(1)} \\ 0 & 0 & \tilde{a}_{33}^{(2)} & \cdots & \tilde{a}_{3n}^{(2)} & \tilde{b}_{3}^{(2)} \\ \vdots & \vdots & & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & & a_{nn}^{(n-1)} & b_{n}^{(n-1)} \end{array}\right).$$

Since all the permitted actions are linear manipulations they can be described in terms of matrices:

$$\left(\tilde{A}^{(0)}, \tilde{b}^{(0)}\right) = P_1 \left(A^{(0)}, b^{(0)}\right), \quad \left(A^{(1)}, b^{(1)}\right) = G_1 \left(\tilde{A}^{(0)}, \tilde{b}^{(0)}\right),$$

where P_1 is a permutation matrix and G_1 is a Frobenius matrix given by

$$P_1 = \begin{array}{cc} \begin{array}{cccccccc} 1 & & & & r & & & \end{array} & \\ \begin{pmatrix} 0 & & \cdots & & 1 & & & \\ & 1 & & & & & & \\ \vdots & & \ddots & & \vdots & & & \\ & & & 1 & & & & \\ 1 & & \cdots & & 0 & & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{pmatrix} & \begin{array}{c} 1 \\ \\ \\ \\ r \\ \\ \\ \\ \end{array} \end{array}, \quad G_1 = \begin{pmatrix} 1 & & & \\ -q_{21} & 1 & & \\ \vdots & & \ddots & \\ -q_{n1} & & & 1 \end{pmatrix}$$

Both matrices P_1 and G_1 are regular with determinants $\det(P_1) = \det(G_1) = 1$ and there holds

$$P_1^{-1} = P_1, \quad G_1^{-1} = \begin{pmatrix} 1 & & & \\ q_{21} & 1 & & \\ \vdots & & \ddots & \\ q_{n1} & & & 1 \end{pmatrix}.$$

Since there holds

$$Ax = b \Leftrightarrow A^{(1)}x = G_1 P_1 A x = G_1 P_1 b = b^{(1)},$$

the systems $Ax = b$ and $A^{(1)}x = b^{(1)}$ have the same solution.

Both matrices P_1 and G_1 are regular with determinants $\det(P_1) = \det(G_1) = 1$ and there holds

$$P_1^{-1} = P_1, \quad G_1^{-1} = \begin{pmatrix} 1 & & & \\ q_{21} & 1 & & \\ \vdots & & \ddots & \\ q_{n1} & & & 1 \end{pmatrix}.$$

Since there holds

$$Ax = b \Leftrightarrow A^{(1)}x = G_1 P_1 A x = G_1 P_1 b = b^{(1)},$$

the systems $Ax = b$ and $A^{(1)}x = b^{(1)}$ have the same solution.

After $n-1$ elimination steps, we arrive at

$$(A,b) \to \left(A^{(1)}, b^{(1)}\right) \to \cdots \to \left(A^{(n-1)}, b^{(n-1)}\right) =: (R, c)\,,$$

where

$$\left(A^{(i)}, b^{(i)}\right) = G_i P_i \left(A^{(i-1)}, b^{(i-1)}\right), \quad \left(A^{(0)}, b^{(0)}\right) := (A, b)\,,$$

with permutation matrics P_i and Frobenius matrics G_i of the form

$$P_i = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array}
\begin{array}{cccccccccccc}
 & & & i & & & & r & & & & \\
\left(\begin{matrix} 1 & & & & & & & & & & \\ & \ddots & & & & & & & & & \\ & & 1 & & & & & & & & \\ & & & 0 & & \cdots & & 1 & & & \\ & & & & 1 & & & & & & \\ & & & \vdots & & \ddots & & \vdots & & & \\ & & & & & & 1 & & & & \\ & & & 1 & & \cdots & & 0 & & & \\ & & & & & & & & 1 & & \\ & & & & & & & & & \ddots & \\ & & & & & & & & & & 1 \end{matrix}\right) \end{array}
\begin{array}{c} \\ \\ \\ \\ i \\ \\ \\ \\ r \\ \\ \\ \\ \end{array}$$

$$G_i = \begin{array}{cc} \begin{array}{ccccccc} & & & i & & & \end{array} & \\ \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & & 1 & & & \\ & & & -q_{i+1,i} & 1 & & \\ & & & \vdots & & \ddots & \\ & & & -q_{ni} & & & 1 \end{pmatrix} & \begin{array}{c} \\ \\ i \\ \\ \\ \\ \end{array} \end{array}$$

The end result

$$(R, c) = G_{n-1} P_{n-1} \cdots G_1 P_1 \, (A, b)$$

is an upper triangular system which has the same solution as the original system.

An example for why pivoting is necessary

We solve the system

$$\begin{aligned} .001x_1 + x_2 &= 1, \\ x_1 + x_2 &= 2. \end{aligned}$$

If we assume to use exact arithmetic, we obtain the upper triangular system

$$\begin{aligned} .001x_1 + x_2 &= 1, \\ -999x_2 &= -998, \end{aligned}$$

which has the solution

$$\begin{aligned} x_2 &= \frac{998}{999} & &\approx .999, \\ x_1 &= \frac{1 - x_2}{.001} = \frac{1 - {}^{998}/_{999}}{.001} = \frac{1000}{999} & &\approx 1.001. \end{aligned}$$

An example for why pivoting is necessary

We solve the system

$$\begin{aligned} .001x_1 + x_2 &= 1, \\ x_1 + x_2 &= 2. \end{aligned}$$

If we assume to use exact arithmetic, we obtain the upper triangular system

$$\begin{aligned} .001x_1 + x_2 &= 1, \\ -999x_2 &= -998, \end{aligned}$$

which has the solution

$$\begin{aligned} x_2 &= \frac{998}{999} && \approx .999, \\ x_1 &= \frac{1 - x_2}{.001} = \frac{1 - {}^{998}/_{999}}{.001} = \frac{1000}{999} && \approx 1.001. \end{aligned}$$

An example for why pivoting is necessary

We solve the system

$$\begin{aligned} .001x_1 + x_2 &= 1, \\ x_1 + x_2 &= 2. \end{aligned}$$

If we assume to use exact arithmetic, we obtain the upper triangular system

$$\begin{aligned} .001x_1 + x_2 &= 1, \\ -999x_2 &= -998, \end{aligned}$$

which has the solution

$$\begin{aligned} x_2 &= \frac{998}{999} && \approx .999, \\ x_1 &= \frac{1 - x_2}{.001} = \frac{1 - {}^{998}/_{999}}{.001} = \frac{1000}{999} && \approx 1.001. \end{aligned}$$

However, if we assume to use two-digit decimal floating-point arithmetic with rounding, we would derive the following upper triangular system

$$\begin{aligned} .001x_1 + x_2 &= 1, \\ (1-1000)x_2 &= 2-1000. \end{aligned}$$

Evaluating the first difference, the computer produces

$$1-1000 \approx \mathrm{fl}(1-1000) = \mathrm{fl}(-999) = \mathrm{fl}(-.999\times 10^3) = -1.0\times 10^3.$$

For the second difference, the computer derives

$$2-1000 \approx \mathrm{fl}(2-1000) = \mathrm{fl}(-998) = \mathrm{fl}(-.998\times 10^3) = -1.0\times 10^3.$$

However, if we assume to use two-digit decimal floating-point arithmetic with rounding, we would derive the following upper triangular system

$$\begin{aligned} .001x_1 + x_2 &= 1, \\ (1 - 1000)x_2 &= 2 - 1000. \end{aligned}$$

Evaluating the first difference, the computer produces

$$1 - 1000 \approx \mathrm{fl}(1 - 1000) = \mathrm{fl}(-999) = \mathrm{fl}(-.999 \times 10^3) = -1.0 \times 10^3.$$

For the second difference, the computer derives

$$2 - 1000 \approx \mathrm{fl}(2 - 1000) = \mathrm{fl}(-998) = \mathrm{fl}(-.998 \times 10^3) = -1.0 \times 10^3.$$

However, if we assume to use two-digit decimal floating-point arithmetic with rounding, we would derive the following upper triangular system

$$\begin{aligned} .001x_1 + x_2 &= 1, \\ (1-1000)x_2 &= 2-1000. \end{aligned}$$

Evaluating the first difference, the computer produces

$$1-1000 \approx \mathrm{fl}(1-1000) = \mathrm{fl}(-999) = \mathrm{fl}(-.999\times 10^3) = -1.0\times 10^3.$$

For the second difference, the computer derives

$$2-1000 \approx \mathrm{fl}(2-1000) = \mathrm{fl}(-998) = \mathrm{fl}(-.998\times 10^3) = -1.0\times 10^3.$$

The upper triangular system that results from two-digit decimal floating-point arithmetic is

$$\begin{aligned} .001x_1 + x_2 &= 1, \\ -1000x_2 &= -1000. \end{aligned}$$

By backward substitution, the solution is given as

$$\begin{aligned} x_2 &= \frac{-1000}{-1000} = 1, \\ x_1 &= \frac{1 - x_2}{.001} = 0. \end{aligned}$$

The error in x_1 is 100% since the computed x_1 is zero while the correct solution for x_1 is close to 1.
As you observe, the upper triangular systems in

$$\begin{aligned} .001x_1 + x_2 &= 1, \\ (1-1000)x_2 &= 2-1000. \end{aligned}$$

and

$$\begin{aligned} .001x_1 + x_2 &= 1, \\ -1000x_2 &= -1000. \end{aligned}$$

do not differ a lot from each other.
The crucial step is the backward substitution where we compute $\frac{1-x_2}{.001}$.
Here, cancellation happens due to the fact that x_2 is close to one.

The error in x_1 is 100% since the computed x_1 is zero while the correct solution for x_1 is close to 1.
As you observe, the upper triangular systems in

$$\begin{aligned} .001x_1 + x_2 &= 1, \\ (1 - 1000)x_2 &= 2 - 1000. \end{aligned}$$

and

$$\begin{aligned} .001x_1 + x_2 &= 1, \\ -1000x_2 &= -1000. \end{aligned}$$

do not differ a lot from each other.
The crucial step is the backward substitution where we compute $\frac{1-x_2}{.001}$.
Here, cancellation happens due to the fact that x_2 is close to one.

The error in x_1 is 100% since the computed x_1 is zero while the correct solution for x_1 is close to 1.
As you observe, the upper triangular systems in

$$\begin{aligned} .001x_1 + x_2 &= 1, \\ (1-1000)x_2 &= 2-1000. \end{aligned}$$

and

$$\begin{aligned} .001x_1 + x_2 &= 1, \\ -1000x_2 &= -1000. \end{aligned}$$

do not differ a lot from each other.
The crucial step is the backward substitution where we compute $\frac{1-x_2}{.001}$.
Here, cancellation happens due to the fact that x_2 is close to one.

Avoid this dilemma

Change the two equations to

$$\begin{aligned} x_1 + x_2 &= 2, \\ .001x_1 + x_2 &= 1. \end{aligned}$$

Applying two-digit decimal floating-point arithmetic to this system yields the upper triangular system

$$\begin{aligned} x_1 + x_2 &= 2, \\ x_2 &= 1. \end{aligned}$$

With backward substitution the solution $x_2 = x_1 = 1$ is derived. This is in much better agreement with the exact solution.

This process is called pivoting.

Avoid this dilemma

Change the two equations to

$$\begin{aligned} x_1 + x_2 &= 2, \\ .001x_1 + x_2 &= 1. \end{aligned}$$

Applying two-digit decimal floating-point arithmetic to this system yields the upper triangular system

$$\begin{aligned} x_1 + x_2 &= 2, \\ x_2 &= 1. \end{aligned}$$

With backward substitution the solution $x_2 = x_1 = 1$ is derived. This is in much better agreement with the exact solution.

This process is called pivoting.

Avoid this dilemma

Change the two equations to

$$\begin{aligned} x_1 + x_2 &= 2, \\ .001x_1 + x_2 &= 1. \end{aligned}$$

Applying two-digit decimal floating-point arithmetic to this system yields the upper triangular system

$$\begin{aligned} x_1 + x_2 &= 2, \\ x_2 &= 1. \end{aligned}$$

With backward substitution the solution $x_2 = x_1 = 1$ is derived. This is in much better agreement with the exact solution.

This process is called pivoting.

Pivoting

Pivot element

The element $a_{r1} = \tilde{a}_{11}^{(0)}$ in the elimination process is called pivot element and the whole substep of its determination is called pivot search. For reasons of numerical stability, usually the chioce

$$|a_{r1}| = \max_{j=1,\dots,n} |a_{j1}|$$

is made.

The whole process inluding permuation of rows is called column pivoting. If the elements of the matrix A are of very different size, total pivoting is advisable. This consists in the choice

$$|a_{rs}| = \max_{k,j=1,\dots,n} |a_{jk}|$$

and subsequent permutation of the 1st row with the rth row and the 1st column with the sth column.

Pivoting

Pivot element

The element $a_{r1} = \tilde{a}_{11}^{(0)}$ in the elimination process is called pivot element and the whole substep of its determination is called pivot search. For reasons of numerical stability, usually the chioce

$$|a_{r1}| = \max_{j=1,\dots,n} |a_{j1}|$$

is made.

The whole process inluding permuation of rows is called column pivoting. If the elements of the matrix A are of very different size, total pivoting is advisable. This consists in the choice

$$|a_{rs}| = \max_{k,j=1,\dots,n} |a_{jk}|$$

and subsequent permutation of the 1st row with the rth row and the 1st column with the sth column.

Remark

According to the column permutation also the unknowns x_k have to be renumbered. Total pivoting is costly so that column pivoting is usually preferred.

LR factorization

The matrices

$$L = \begin{pmatrix} 1 & & 0 \\ l_{21} & 1 & \\ \vdots & \ddots & \ddots \\ l_{n1} & \cdots & l_{n,n-1} 1 \end{pmatrix}, \quad R = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ & r_{22} & \cdots & r_{2n} \\ & & \ddots & \vdots \\ 0 & & & r_{nn} \end{pmatrix}$$

are factors in the (multiplicative) decomposition of the matrix PA,

$$PA = LR, \quad P \coloneqq P_{n-1} \cdots P_1.$$

If there exists such a decomposition with $P = I$, then it is uniquely determined.

Once an LR decomposition is computed, the solution of the linear system $Ax = b$ can be achieved by successively solving two triangular systems

$$Ly = Pb, \quad Rx = y$$

by forward and backward substitution, respectively.

If there exists such a decomposition with $P = I$, then it is uniquely determined.
Once an LR decomposition is computed, the solution of the linear system $Ax = b$ can be achieved by successively solving two triangular systems

$$Ly = Pb, \quad Rx = y$$

by forward and backward substitution, respectively.

Conditioning of Gaussian elimination

For any (regular) matrix A there exists an LR decomposition like $PA = LR$. We derive

$$R = L^{-1}PA, \quad R^{-1} = (PA)^{-1}L$$

for R and its inverse R^{-1}. Due to column pivoting, all the elements of L and L^{-1} are less or equal one and there holds

$$\mathrm{cond}_\infty(L) = \|L\|_\infty \left\|L^{-1}\right\|_\infty \leq n^2.$$

Therefore,

$$\begin{aligned}\mathrm{cond}_\infty(R) &= \|R\|_\infty \left\|R^{-1}\right\|_\infty = \left\|L^{-1}PA\right\|_\infty \left\|(PA)^{-1}L\right\|_\infty \\ &\leq \left\|L^{-1}\right\|_\infty \|PA\|_\infty \left\|(PA)^{-1}\right\|_\infty \|L\|_\infty \leq n^2 \,\mathrm{cond}_\infty(PA).\end{aligned}$$

Conditioning of Gaussian elimination

For any (regular) matrix A there exists an LR decomposition like $PA = LR$. We derive

$$R = L^{-1}PA, \quad R^{-1} = (PA)^{-1}L$$

for R and its inverse R^{-1}. Due to column pivoting, all the elements of L and L^{-1} are less or equal one and there holds

$$\operatorname{cond}_\infty(L) = \|L\|_\infty \left\|L^{-1}\right\|_\infty \leq n^2.$$

Therefore,

$$\begin{aligned}\operatorname{cond}_\infty(R) &= \|R\|_\infty \left\|R^{-1}\right\|_\infty = \left\|L^{-1}PA\right\|_\infty \left\|(PA)^{-1}L\right\|_\infty \\ &\leq \left\|L^{-1}\right\|_\infty \|PA\|_\infty \left\|(PA)^{-1}\right\|_\infty \|L\|_\infty \leq n^2 \operatorname{cond}_\infty(PA).\end{aligned}$$

Perturbation of the right-hand side

Applying the perturmation theorem, we obtain the estimate for the solution of the equation $LRx = Pb$ (considering only perturbation of the right-hand side b, $\delta A = 0$)

$$\frac{\|\delta x\|_\infty}{\|x\|\infty} \leq \mathrm{cond}_\infty(L)\,\mathrm{cond}_\infty(R)\frac{\|\delta Pb\|_\infty}{\|Pb\|\infty} \leq n^4\,\mathrm{cond}_\infty(PA)\frac{\|\delta Pb\|_\infty}{\|Pb\|\infty}.$$

Hence, the conditioning of the original system $Ax = b$ by the LR decomposition is amplified by n^4 in the worst case. However, this is an extremely pessimistic estimate which can be significantly improve.

Perturbation of the right-hand side

Applying the perturmation theorem, we obtain the estimate for the solution of the equation $LRx = Pb$ (considering only perturbation of the right-hand side b, $\delta A = 0$)

$$\frac{\|\delta x\|_\infty}{\|x\|\,\infty} \le \operatorname{cond}_\infty(L)\operatorname{cond}_\infty(R)\frac{\|\delta Pb\|_\infty}{\|Pb\|\,\infty} \le n^4 \operatorname{cond}_\infty(PA)\frac{\|\delta Pb\|_\infty}{\|Pb\|\,\infty}.$$

Hence, the conditioning of the original system $Ax = b$ by the LR decomposition is amplified by n^4 in the worst case. However, this is an extremely pessimistic estimate which can be significantly improve.

Symmetric matrices

For symmetric matrices $A = A^T$ we would like that the symmetry is also reflected in the LR factorization. It should be only half the work to achive this.

Let us assume that there exists a decomposition such that $A = LR$, where L is the a unit lower triangular matrix and R is a upper triangular matrix. From the symmerty of A it follows that

$$LR = A = A^T = (LR)^T = (LD\tilde{R})^T = \tilde{R}^T D L^T,$$

where

$$\tilde{R} = \begin{pmatrix} 1 & r_{12}/r_{11} & \cdots & r_{1n}/r_{11} \\ & \ddots & \ddots & \vdots \\ & & 1 & r_{n-1,n}/r_{n-1,n-1} \\ 0 & & & 1 \end{pmatrix}, \quad D = \begin{pmatrix} r_{11} & & 0 \\ & \ddots & \\ 0 & & r_{nn} \end{pmatrix}.$$

The uniqueness of the LR decomposition implies that $L = \tilde{R}^T$ and $R = DL^T$ such that A may be written as

$$A = LDL^T.$$

Symmetric matrices

For symmetric matrices $A = A^T$ we would like that the symmetry is also reflected in the LR factorization. It should be only half the work to achive this.

Let us assume that there exists a decomposition such that $A = LR$, where L is the a unit lower triangular matrix and R is a upper triangular matrix. From the symmerty of A it follows that

$$LR = A = A^T = (LR)^T = (LD\tilde{R})^T = \tilde{R}^T D L^T,$$

where

$$\tilde{R} = \begin{pmatrix} 1 & r_{12}/r_{11} & \cdots & r_{1n}/r_{11} \\ & \ddots & \ddots & \vdots \\ & & 1 & r_{n-1,n}/r_{n-1,n-1} \\ 0 & & & 1 \end{pmatrix}, \quad D = \begin{pmatrix} r_{11} & & 0 \\ & \ddots & \\ 0 & & r_{nn} \end{pmatrix}.$$

The uniqueness of the LR decomposition implies that $L = \tilde{R}^T$ and $R = DL^T$ such that A may be written as

$$A = LDL^T.$$

Symmetric matrices

For symmetric matrices $A = A^T$ we would like that the symmetry is also reflected in the LR factorization. It should be only half the work to achive this.

Let us assume that there exists a decomposition such that $A = LR$, where L is the a unit lower triangular matrix and R is a upper triangular matrix. From the symmerty of A it follows that

$$LR = A = A^T = (LR)^T = (LD\tilde{R})^T = \tilde{R}^T D L^T,$$

where

$$\tilde{R} = \begin{pmatrix} 1 & r_{12}/r_{11} & \cdots & r_{1n}/r_{11} \\ & \ddots & \ddots & \vdots \\ & & 1 & r_{n-1,n}/r_{n-1,n-1} \\ 0 & & & 1 \end{pmatrix}, \quad D = \begin{pmatrix} r_{11} & & 0 \\ & \ddots & \\ 0 & & r_{nn} \end{pmatrix}.$$

The uniqueness of the LR decomposition implies that $L = \tilde{R}^T$ and $R = DL^T$ such that A may be written as

$$A = LDL^T.$$

Cholesky decomposition

Symmetric positive definite matrices allow for a Cholesky decomposition

$$A = LDL^T = \tilde{L}\tilde{L}^T$$

with the matrix $\tilde{L} := LD^{1/2}$. For computing the Cholesky decomposition it suffices to compute the matrices D and L. This reduces the required work to half the work for the LR decomposition.

Cholesky decomposition – algorithm

The algorithm starts from the relation $A = \tilde{L}\tilde{L}^T$ which is

$$\begin{pmatrix} \tilde{l}_{11} & & 0 \\ \vdots & \ddots & \\ \tilde{l}_{n1} & \cdots & \tilde{l}_{nn} \end{pmatrix} \begin{pmatrix} \tilde{l}_{11} & \cdots & \tilde{l}_{n1} \\ & \ddots & \vdots \\ 0 & & \tilde{l}_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & \cdots & a_{nn} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix}$$

and yields equations to determine the first column of $\tilde{L}$:

$$\tilde{l}_{11}^2 = a_{11}, \quad \tilde{l}_{21}\tilde{l}_{11} = a_{21}, \quad \ldots, \quad \tilde{l}_{n1}\tilde{l}_{11} = a_{n1},$$

from which

$$\tilde{l}_{11} = \sqrt{a_{11}}, \quad \text{for} \quad j = 2, \ldots, n: \quad \tilde{l}_{21} = \frac{a_{21}}{\tilde{l}_{11}} = \frac{a_{21}}{\sqrt{a_{11}}}.$$

is obtained.

Let now for some $i \in \{2, \ldots, n\}$ the elements $\tilde{l}_{jk},\ k = 1, \ldots, i-1,\ j = k, \ldots, n$ be already computed. Then, via

$$\begin{aligned} \tilde{l}_{i1}^2 + \tilde{l}_{i2}^2 + \ldots \tilde{l}_{ii}^2 &= a_{ii}, \quad \tilde{l}_{ii} > 0, \\ \tilde{l}_{j1}\tilde{l}_{i1} + \tilde{l}_{j2}\tilde{l}_{i2} + \ldots + \tilde{l}_{ji}\tilde{l}_{ii} &= a_{ji}, \end{aligned}$$

the next elements $\tilde{l}_{ii}$ and $\tilde{l}_{ji},\ j = i+1, \ldots, n$ can be obtained as

$$\begin{aligned} \tilde{l}_{ii} &= \sqrt{a_{ii} - \tilde{l}_{i1}^2 - \tilde{l}_{i2}^2 - \ldots - \tilde{l}_{i,i-1}^2}, \\ \tilde{l}_{ji} &= \tilde{l}_{ii}^{-1} \left\{ a_{ji} - \tilde{l}_{j1}\tilde{l}_{i1} - \tilde{l}_{j2}\tilde{l}_{i2} - \ldots - \tilde{l}_{j,i-1}\tilde{l}_{i,i-1} \right\}, \end{aligned}$$

Orthogonal decomposition

Let $A \in \mathbb{R}^{m \times n}$ be a not necessarily quadratic matrix and $b \in \mathbb{R}^m$ a right-hand side vector. We consider the system

$$Ax = b$$

for $x \in \mathbb{R}^n$. We seek a vector $\tilde{x} \in \mathbb{R}^n$ with minimal defect norm $\|b - Ax\|$. A solution is obtained by solving the normal equation

$$A^T Ax = A^T b.$$

QR decomposition

Let $A \in \mathbb{R}^{m\times n}$ be a rectangular matrix with $m \geq n$ and rank$(A) = n$. Then there exists a uniquely determined othonormal matrix $Q \in \mathbb{R}^{m\times n}$ with the property

$$Q^T Q = I$$

and a uniquely determined upper triangular matrix $R \in \mathbb{R}^{n\times n}$ with diagonal $r_{ii} > 0,\ i = 1\ldots, n$, such that

$$A = QR.$$

Householder transformation

For any normalized vector $v \in \mathbb{R}^m, \|v\|_2 = 1$, the matrix

$$S = I - 2vv^T \in \mathbb{R}^{m \times m}$$

is called Householder transformation and the vector v is called Householder vector. Householder transformations are symmetric $S = S^T$ and orthonormal $S^T S = SS^T = I$.

Geometric interpretation

For the geometric interpretation of the Householder transformation S, let us consider an arbitrary normed vector $v \in \mathbb{R}^2, \|v\|_2 = 1$. The two vectors $\left\{v, v^\perp\right\}$ form a basis of $\mathbb{R}^2$, where $v^T v^\perp = 0$.

For an arbitrary vector $u = \alpha v + \beta v^\perp \in \mathbb{R}^2$, there holds

$$\begin{aligned} Su &= \left(I - 2vv^T\right)\left(\alpha v + \beta v^\perp\right) \\ &= \alpha v + \beta v^\perp - 2\alpha(v \underbrace{v^T)v}_{=1} - 2\beta(v \underbrace{v^T)v^\perp}_{=0} \\ &= -\alpha v + \beta v^\perp. \end{aligned}$$

Geometric interpretation

For the geometric interpretation of the Householder transformation S, let us consider an arbitrary normed vector $v \in \mathbb{R}^2, \|v\|_2 = 1$. The two vectors $\left\{v, v^\perp\right\}$ form a basis of $\mathbb{R}^2$, where $v^T v^\perp = 0$. For an arbitrary vector $u = \alpha v + \beta v^\perp \in \mathbb{R}^2$, there holds

$$
\begin{aligned}
Su &= \left(I - 2vv^T\right)\left(\alpha v + \beta v^\perp\right) \\
&= \alpha v + \beta v^\perp - 2\alpha(v \underbrace{v^T)v}_{=1} - 2\beta(v \underbrace{v^T)v^\perp}_{=0} \\
&= -\alpha v + \beta v^\perp.
\end{aligned}
$$

Householder algorithm

Starting from a matrix $A \in \mathbb{R}^{m \times n}$ the Householder algorithm generates a sequence of matrices

$$A := A^{(0)} \to \cdots \to A^{(i)} \to \cdots \to A^{(n)} = \tilde{R},$$

where $A^{(i)}$ has the form

$$A^{(i)} = \begin{array}{c c} \begin{array}{cccccc} & & & i & & \end{array} & \\ \left(\begin{array}{cccccc} * & \cdots & & & \cdots & * \\ & \ddots & & & & \vdots \\ & & * & & \cdots & * \\ & & & * & \cdots & * \\ & 0 & & \vdots & \ddots & \vdots \\ & & & * & \cdots & * \end{array}\right) & \begin{array}{c} \\ \\ \\ i \\ \\ \\ \end{array} \end{array}$$

In the ith step the Householder transformation $S_i \in \mathbb{K}^{m\times m}$ is determined such that

$$S_i A^{(i-1)} = A^{(i)}.$$

After n steps the result is

$$\tilde{R} = A^{(n)} = S_n S_{n-1} \cdots s_q A = \tilde{Q}^T A,$$

where $\tilde{Q} \in \mathbb{R}^{m\times m}$ as product of unitary matrices is also unitary and $\tilde{R} \in \mathbb{R}^{m\times m}$ has the form

$$\tilde{R} = \left.\begin{pmatrix} \left.\begin{matrix} r_{11} & \cdots & r_{1n} \\ & \ddots & \vdots \\ 0 & & r_{nn} \end{matrix}\right\} n \\ \hline \begin{matrix} 0 & \cdots & 0 \end{matrix} \end{pmatrix}\right\} m \ .$$

Therefore we have the representation

$$A = S_1^T \cdots S_n^T \tilde{R} = \tilde{Q}\tilde{R}.$$

In the ith step the Householder transformation $S_i \in \mathbb{K}^{m\times m}$ is determined such that

$$S_i A^{(i-1)} = A^{(i)}.$$

After n steps the result is

$$\tilde{R} = A^{(n)} = S_n S_{n-1} \cdots s_q A = \tilde{Q}^T A,$$

where $\tilde{Q} \in \mathbb{R}^{m\times m}$ as product of unitary matrices is also unitary and $\tilde{R} \in \mathbb{R}^{m\times m}$ has the form

$$\tilde{R} = \left(\begin{array}{ccc} r_{11} & \cdots & r_{1n} \\ & \ddots & \vdots \\ 0 & & r_{nn} \\ \hline 0 & \cdots & 0 \end{array}\right) \left.\begin{array}{l} \left.\begin{array}{c} \\ \\ \\ \end{array}\right\} n \\ \\ \end{array}\right\} m \; .$$

Therefore we have the representation

$$A = S_1^T \cdots S_n^T \tilde{R} = \tilde{Q}\tilde{R}.$$

In the ith step the Householder transformation $S_i \in \mathbb{K}^{m \times m}$ is determined such that

$$S_i A^{(i-1)} = A^{(i)}.$$

After n steps the result is

$$\tilde{R} = A^{(n)} = S_n S_{n-1} \cdots s_q A = \tilde{Q}^T A,$$

where $\tilde{Q} \in \mathbb{R}^{m \times m}$ as product of unitary matrices is also unitary and $\tilde{R} \in \mathbb{R}^{m \times m}$ has the form

$$\tilde{R} = \left(\begin{array}{ccc} r_{11} & \cdots & r_{1n} \\ & \ddots & \vdots \\ 0 & & r_{nn} \\ \hline 0 & \cdots & 0 \end{array} \right) \begin{array}{l} \left.\vphantom{\begin{array}{c} r \\ \ddots \\ r \end{array}}\right\} n \\ \vphantom{0} \end{array} \left.\vphantom{\begin{array}{c} r \\ \ddots \\ r \\ 0 \end{array}}\right\} m \ .$$

Therefore we have the representation

$$A = S_1^T \cdots S_n^T \tilde{R} = \tilde{Q}\tilde{R}.$$

Obtain QR decomposition

We obtain the desired QR decomposition of A by removing the last $m-n$ columns in $\tilde{Q}$ and the last $m-n$ rows in $\tilde{R}$:

$$A = \left(\begin{array}{c|c} \underbrace{\quad Q \quad}_{n} & \underbrace{\quad * \quad}_{m-n} \end{array} \right) \cdot \left(\begin{array}{c} R \\ \hline 0 \end{array} \right) \begin{array}{l} \left.\vphantom{R}\right\} n \\ \left.\vphantom{0}\right\} m-n \end{array} = QR.$$

Step 1

As a first step of this process, we construct a Householder matrix S_1 which transforms a_1 (the first column of A) into a multiple of e_1, the first coordinate vector.
This results in a vector which has just a first component. Depending on the sign of a_{11} we choose one of the axes $\operatorname{span}\{a_1 + \|a_1\| e_1\}$ or $\operatorname{span}\{a_1 - \|a_1\| e_1\}$ in order to reduce round-off errors. In case $a_{11} \geq 0$ we choose

$$v_1 = \frac{a_1 + \|a_1\|_2 e_1}{\left\|a_1 + \|a_1\|_2 e_1\right\|_2}, \quad v_1^{\perp} = \frac{a_1 - \|a_1\|_2 e_1}{\left\|a_1 - \|a_1\|_2 e_1\right\|_2}.$$

Step 1

As a first step of this process, we construct a Householder matrix S_1 which transforms a_1 (the first column of A) into a multiple of e_1, the first coordinate vector.
This results in a vector which has just a first component.
Depending on the sign of a_{11} we choose one of the axes $\operatorname{span}\{a_1 + \|a_1\| \, e_1\}$ or $\operatorname{span}\{a_1 - \|a_1\| \, e_1\}$ in order to reduce round-off errors. In case $a_{11} \geq 0$ we choose

$$v_1 = \frac{a_1 + \|a_1\|_2 \, e_1}{\left\| a_1 + \|a_1\|_2 \, e_1 \right\|_2}, \quad v_1^{\perp} = \frac{a_1 - \|a_1\|_2 \, e_1}{\left\| a_1 - \|a_1\|_2 \, e_1 \right\|_2}.$$

Then the matrix $A^{(1)} = (I - 2v_1 v_1^T)A$ has the column vectors

$$
\begin{aligned}
a_1^{(1)} &= (I - 2v_1 v_1^T)a_1 = -\|a_1\|_2\, e_1, \\
a_k^{(1)} &= (I - 2v_1 v_1^T)a_k = a_k - 2(a_k, v_1)v_1, \;\; k = 2, \dots, n.
\end{aligned}
$$

Remark

In contrast to the first step of the Gaussian elimination, the first row of the resulting matrix is changed.

Step i

Let the transformed matrix $A^{(i-1)}$ be already computed. The Householder transformation is given as

$$S_i = I - 2v_i v_i^T = \left(\begin{array}{c|c} I & 0 \\ \hline 0 & I - 2\tilde{v}_i\tilde{v}^T \end{array}\right), \quad v_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \tilde{v}_i \end{pmatrix} \begin{array}{l} \left.\begin{array}{c} \\ \\ \\ \end{array}\right\} i-1 \\ \\ \end{array} \Bigg\} m.$$

$\underbrace{}_{i-1}$

The application of the (orthonormal) matrix S_i to $A^{(i-1)}$ leaves the first $i-1$ rows and columns of $A^{(i-1)}$ unchanged. For the construction of v_i, we use the considerations of step 1 for the submatrix

$$\tilde{A}^{(i-1)} = \begin{pmatrix} \tilde{a}_{ii}^{(i-1)} & \cdots & \tilde{a}_{in}^{(i-1)} \\ \vdots & \ddots & \vdots \\ \tilde{a}_{mi}^{(i-1)} & \cdots & \tilde{a}_{mn}^{(i-1)} \end{pmatrix} = \left(\tilde{a}_1^{(i-1)}, \ldots, \tilde{a}_n^{(i-1)}\right).$$

Thus, it follows that

$$\tilde{v}_i = \frac{\tilde{a}_{ii}^{(i-1)} - \left\| \tilde{a}_{ii}^{(i-1)} \right\|_2 \tilde{e}_i}{\left\| \tilde{a}_{ii}^{(i-1)} - \left\| \tilde{a}_{ii}^{(i-1)} \right\|_2 \tilde{e}_i \right\|_2}, \quad \tilde{v}_i = \frac{\tilde{a}_{ii}^{(i-1)} + \left\| \tilde{a}_{ii}^{(i-1)} \right\|_2 \tilde{e}_i}{\left\| \tilde{a}_{ii}^{(i-1)} + \left\| \tilde{a}_{ii}^{(i-1)} \right\|_2 \tilde{e}_i \right\|_2},$$

and the matrix $A^{(i)}$ has the column vectors

$$\begin{aligned}
a_k^{(i)} &= a_k^{(i-1)}, \quad k = 1, \dots, i-1, \\
a_i^{(i)} &= \left(a_{1i}^{(i-1)}, \dots, a_{i-1,i}^{(i-1)}, \left\| \tilde{a}_i^{(i-1)} \right\|_2, 0, \dots, 0 \right)^T, \\
a_k^{(i)} &= a_k^{(i-1)} - 2 \left(\tilde{a}_k^{(i-1)}, \tilde{v}_i \right) v_i, \quad k = i+1, \dots, n.
\end{aligned}$$

Remark

For a quadratic matrix $A \in \mathbb{R}^{n \times n}$ the costs for a QR decomposition are about double the costs for a LR decomposition.

Programming

Implement the Householder algortihm to orthogonalize the vectors from prestudy 4.2.