

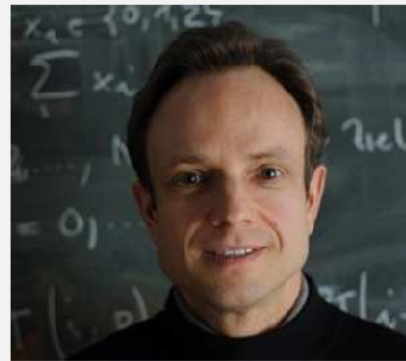
Course Video

Section 2.3–Section 2.5: Gaussian Elimination

Prof. R. Hiptmair, SAM, ETH Zurich

Date: September 18, 2020

(C) Seminar für Angewandte Mathematik, ETH Zürich

**Prerequisites.**

- Familiarity with Gaussian elimination as introduced in the linear algebra course.
- Ability to read C++ codes

Dependencies. Section 1.4

Duration: 30 minutes

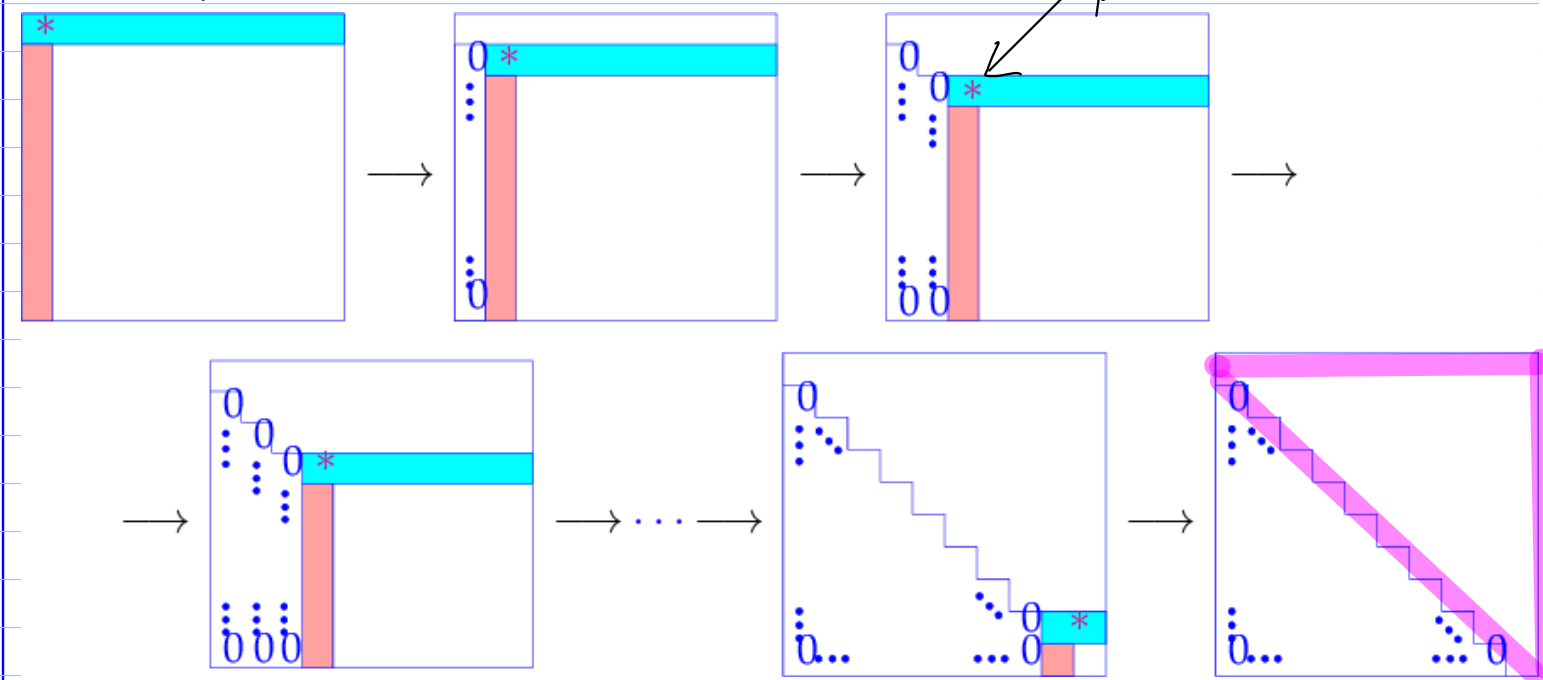
II. Direct Methods for Square Linear Systems of Equations

2.3. Gaussian Elimination (GE)

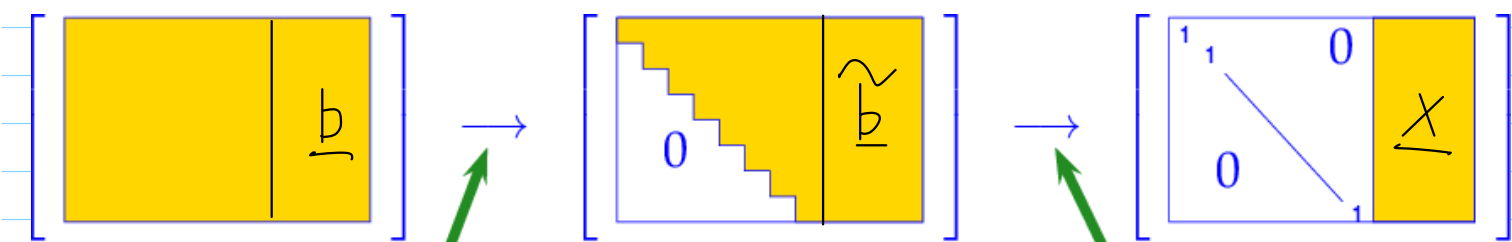
2.3.1. Basic Algorithm ($\rightarrow$ LA)

Successive row ops. of LSE $A\mathbf{x} = \mathbf{b}$ to convert it to triangular form

For A :



②

For $\underline{A}\underline{x} = \underline{b}$, $A \in \mathbb{R}^{n,n}$ 

forward elimination

back substitution
(row trfs.)

§ 2.3.1.15

 $O(n^3)$ $O(n^2)$

Rem 2.3.1.14: Block GE

LSE: $\underline{A}\underline{x} = \underline{b}$, $A \in \mathbb{R}^{n,n}$, $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ with $A_{11} \in \mathbb{R}^{k,k}$

Assume: A_{11} is regularRecall: blockwise $M \times M$:

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}. \quad (1.3.1.14)$$

Now: GE like for 2×2 LSE
[except for commutativity]

$$\begin{bmatrix} A_{11} & A_{12} & | & b_1 \\ A_{21} & A_{22} & | & b_2 \end{bmatrix} \xrightarrow[\text{B.S.}]{\text{F.E.}} \begin{bmatrix} A_{11} & A_{12} & | & b_1 \\ 0 & A_{22} - A_{21}A_{11}^{-1}A_{12} & | & b_2 - A_{21}A_{11}^{-1}b_1 \end{bmatrix} \xrightarrow{\text{B.S.}} \begin{bmatrix} I & 0 & | & A_{11}^{-1}(b_1 - A_{12}S^{-1}b_2) \\ 0 & I & | & S^{-1}b_2 \end{bmatrix};$$

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^n$$

$$S := A_{22} - A_{21}A_{11}^{-1}A_{12} : \text{Schur complement}$$

Rem 2.3.2.19: Block LU-dec.

With

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \underbrace{\begin{bmatrix} I & 0 \\ A_{21}A_{11}^{-1} & I \end{bmatrix}}_{\text{block LU-factorization}} \begin{bmatrix} A_{11} & A_{12} \\ 0 & S \end{bmatrix}, \quad \text{with Schur complement} \quad \boxed{S := A_{22} - A_{21}A_{11}^{-1}A_{12}}. \quad (2.3.2.20)$$

③ 2.3.2. LU-Decomposition

[equivalent to GE $\Rightarrow \text{cost}(LU) = O(n^3)$]

$$\begin{bmatrix} \text{A} \end{bmatrix} = \begin{bmatrix} \text{L} \end{bmatrix} \cdot \begin{bmatrix} \text{U} \end{bmatrix}$$

(possibly permuted)
normalized lower triangular
upper triangular

Definition 2.3.2.3. LU-decomposition/LU-factorization

Given a square matrix $\mathbf{A} \in \mathbb{K}^{n,n}$, an *upper triangular matrix* $\mathbf{U} \in \mathbb{K}^{n,n}$ and a *normalized lower triangular matrix* ($\rightarrow$ Def. 1.1.2.3) form an **LU-decomposition/LU-factorization** of $\mathbf{A}$, if $\mathbf{A} = \mathbf{LU}$.

§ 2.3.2.15 : Solving LSE via LM

$$\text{LSA: } \mathbf{Ax} = \underline{b} \Leftrightarrow \mathbf{L}(\mathbf{Ux}) = \underline{b} \Rightarrow \begin{matrix} \mathbf{Lz} = \underline{b} \\ \mathbf{Ux} = \underline{z} \end{matrix}$$

- $\mathbf{Ax} = \mathbf{b}$:
- ① LU-decomposition $\mathbf{A} = \mathbf{LU}$, #elementary operations $= \frac{1}{3}n(n-1)(n+1) \rightarrow O(n^3)$
 - ② forward substitution, solve $\mathbf{Lz} = \mathbf{b}$, #elementary operations $= \frac{1}{2}n(n-1)$
 - ③ backward substitution, solve $\mathbf{Ux} = \mathbf{z}$, #elementary operations $= \frac{1}{2}n(n+1)$
- $\} \rightarrow O(n^3)$

$\hookrightarrow$ Three-stage splitting of GE

setup phase
(factorization)
Cost: $O(n^3)$

+

elimination phase
(forward/backward substitution)
Cost: $O(n^2)$

2.5.0.4. GE/LU-dec. in Eigen

$\mathbf{A} \in \mathbb{R}^{n,n} \leftrightarrow$ (square) matrix object

LSE multiple r.h.s. : $\mathbf{AX} = \mathbf{B} \in \mathbb{R}^{n,l}$
 $\hookrightarrow$ contained in columns of $\mathbf{B}$

linear algebra	EIGEN
$\mathbf{X} = \mathbf{A}^{-1}\mathbf{B} = [\mathbf{A}^{-1}(\mathbf{B})_{:,1}, \dots, \mathbf{A}^{-1}(\mathbf{B})_{:,l}]$	$\mathbf{X} = \mathbf{A}.\text{lu}().\text{solve}(\mathbf{B})$
	$\downarrow$ ① $O(n^3)$ $\downarrow$ ② ③ $O(n^2l)$
Cost:	

Never contemplate implementing a general solver for linear systems of equations!

If possible, use algorithms from numerical libraries! ($\rightarrow$ Exp. 2.3.1.7)

④ Rem 2.5.0, 10 : Many sequential solves of LSE

C++ code 2.5.0.11: Wasteful approach!
→ [GITLAB](#)

```

2 // Setting:  $N \gg 1$ ,
3 // large matrix  $A \in \mathbb{K}^{n,n}$ 
4 for(int j = 0; j < N; ++j){
5     x = A.lu().solve(b);
6     b = some_function(x);
7 }

```

computational effort $O(Nn^3)$

C++ code 2.5.0.12: Smart approach!
→ [GITLAB](#)

```

2 // Setting:  $N \gg 1$ ,
3 // large matrix  $A \in \mathbb{K}^{n,n}$ 
4 auto A_lu_dec = A.lu();
5 for(int j = 0; j < N; ++j){
6     x = A_lu_dec.solve(b);
7     b = some_function(x);
8 }

```

computational effort $O(n^3 + Nn^2)$

setup phase

(factorization)

Cost: $O(n^3)$

+

elimination phase

(forward/backward substitution)

Cost: $O(n^2)$

→ A rationale for LU

5

7