2. Linear algebra

1. Systems of linear equations (SLEs). Matrix operations
2. Solutions of SLEs by the Gauss elimination
3. Two-dimensional arrays and operations with matrices in MATLAB
4. Determinant and matrix inverse
5. Solution of SLEs in MATLAB
6. Summary

Text

A. Gilat, *MATLAB: An Introduction with Applications*, 4th ed., Wiley

Wikipedia: Matrix, SLE, etc.

2.1. Systems of linear equations (SLEs). Matrix operations

- Systems of linear equations (SLEs)
- Notion of a matrix
- Operations with matrices
- Special matrices
- Matrix form of a SLE

Reading assignment

Gilat, 3.1, 3.2, 3.4, and 3.5

2.1. Systems of linear equations (SLEs). Matrix operations

Systems of linear equations

The **system of linear equations (SLE)** of m equations with n unknowns is the system :

$$
\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + \cdots + a_{1j-1}x_{j-1} + a_{1j}x_j + a_{1j+1}x_{j+1} + \cdots + a_{1n-1}x_{n-1} + a_{1n}x_n = b_1 \\
&a_{21}x_1 + a_{22}x_2 + \cdots + a_{2j-1}x_{j-1} + a_{2j}x_j + a_{2j+1}x_{j+1} + \cdots + a_{2n-1}x_{n-1} + a_{2n}x_n = b_2 \\
&\ldots \qquad (2.1.1) \\
&a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{ij-1}x_{j-1} + a_{ij}x_j + a_{ij+1}x_{j+1} + \cdots + a_{in-1}x_{n-1} + a_{in}x_n = b_i \\
&\ldots \\
&a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mj-1}x_{j-1} + a_{mj}x_j + a_{mj+1}x_{j+1} + \cdots + a_{mn-1}x_{n-1} + a_{mn}x_n = b_m
\end{aligned}
$$

a_{ij}, **coefficients** of equations ($i = 1,..,m, j = 1,..,n$). m is the number of equations.

x_j, **unknowns** ($j = 1,..,n$). n is the number of unknowns.

b_j, **right-hand sides** (RHSs) of equations ($j = 1,..,m$).

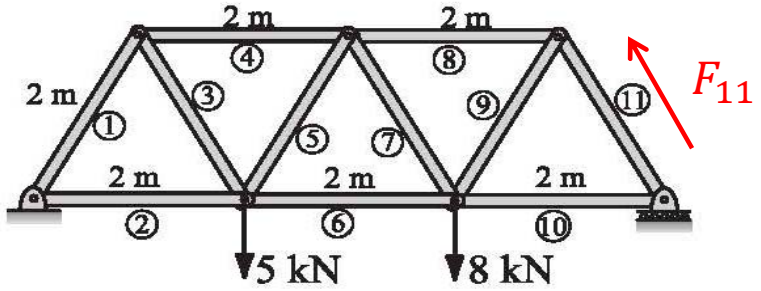
To solve a SLE means to find such x_j that turn every equation in the system into identity.

- *In general, a SLE may have/not have a solution.*
- *If a solution exits, it can be unique, or, alternatively, the system may have multiple solutions.*
- Solution of SLEs is a fundamental mathematical problem. Multiple problems in science and engineering reduce to the solution of SLEs.

2.1. Systems of linear equations (SLEs). Matrix operations

Example: SLE with respect to forces in a truss

A truss is a structure made of members joined at their ends. For the truss shown in the figure, the forces in the 11 members are determined by solving the following system of 11 equations.

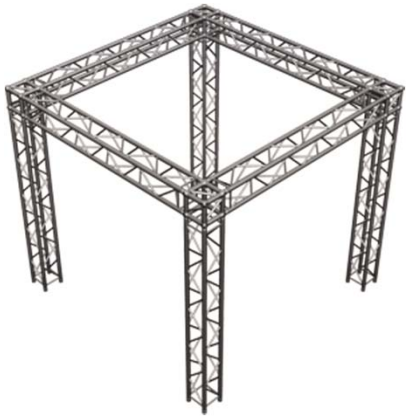


$$\frac{1}{2}F_1 + F_2 = 0,\ \frac{\sqrt{3}}{2}F_1 = -6$$

$$-\frac{1}{2}F_1 + \frac{1}{2}F_3 + F_4 = 0,\ -\frac{\sqrt{3}}{2}F_1 - \frac{\sqrt{3}}{2}F_3 = 0,\ -F_2 - \frac{1}{2}F_3 + \frac{1}{2}F_5 + F_6 = 0$$

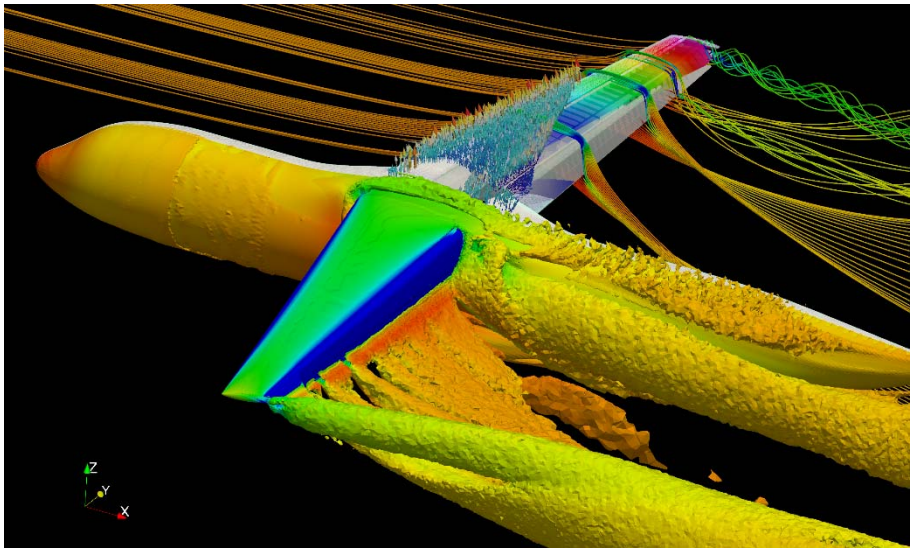
$$\frac{\sqrt{3}}{2}F_3 + \frac{\sqrt{3}}{2}F_5 = 5,\ -F_4 - \frac{1}{2}F_5 + \frac{1}{2}F_7 + F_8 = 0,\ -\frac{\sqrt{3}}{2}F_5 - \frac{\sqrt{3}}{2}F_7 = 0$$

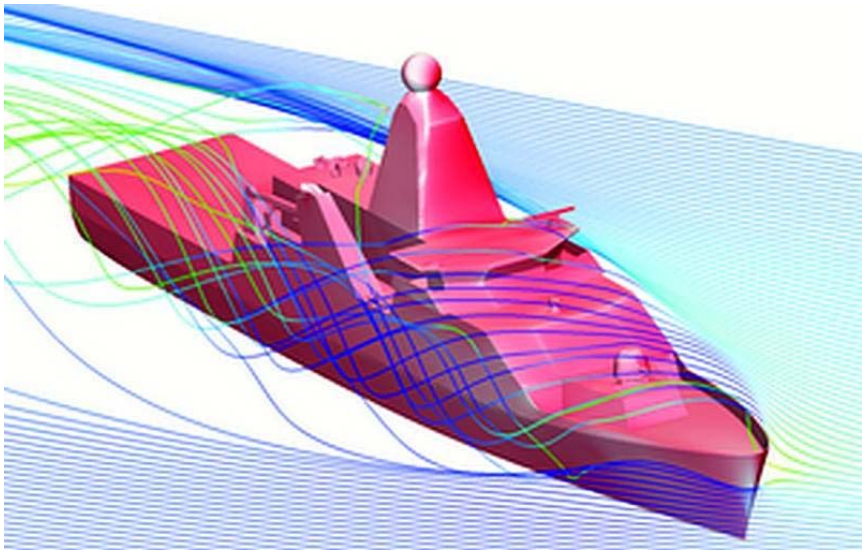
$$-F_6 - \frac{1}{2}F_7 + \frac{1}{2}F_9 + F_{10} = 0,\ \frac{\sqrt{3}}{2}F_7 + \frac{\sqrt{3}}{2}F_9 = 8,\ -F_8 - \frac{1}{2}F_9 + \frac{1}{2}F_{11} = 0$$

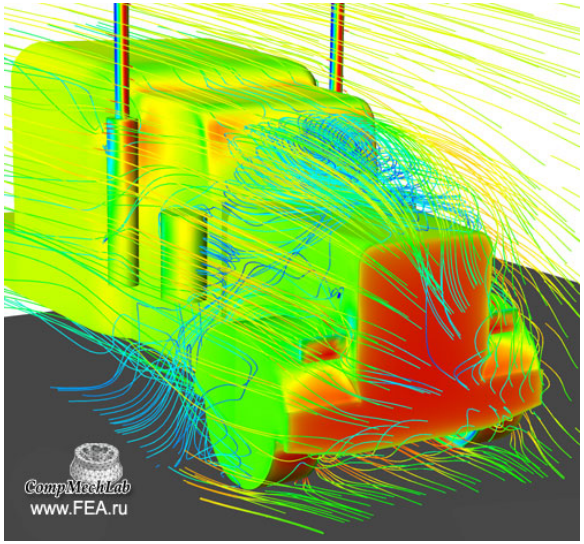


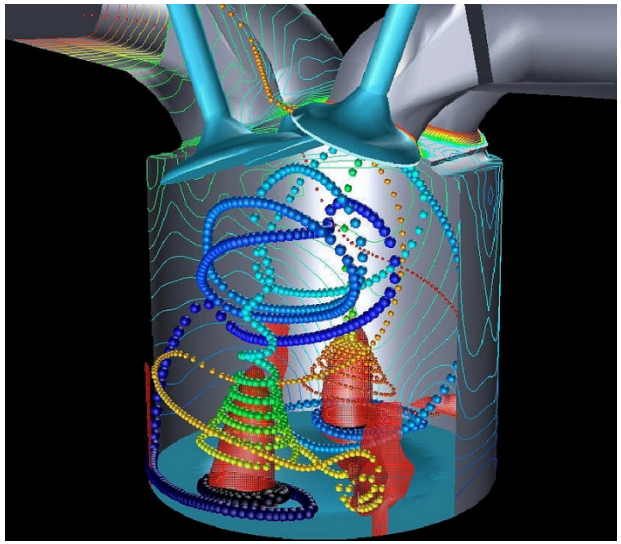
2.1. Systems of linear equations (SLEs). Matrix operations

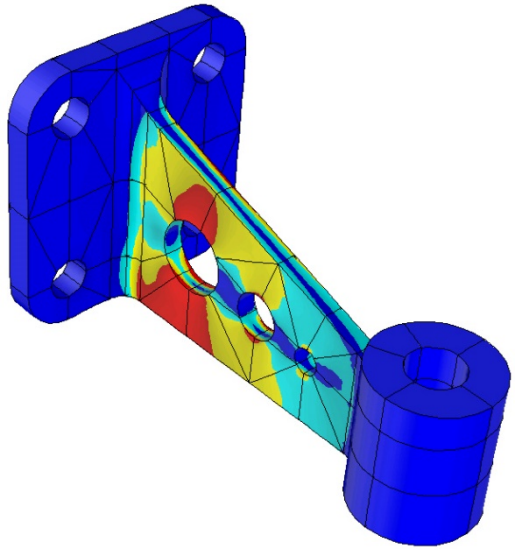
Applications of SLEs: Numerical solutions of partial differential equations: Computational heat transfer, computational fluid mechanics, computational solid mechanics, and many more...











2.1. Systems of linear equations (SLEs). Matrix operations

- The first step to solve a SLE is to re-formulate the problem in the matrix form.
- Matrix form is also important for computer-aided solution of SLEs, because all input data are prepared in the form of matrices (one- and two-dimensional arrays).

Notion of a matrix

In mathematics, the **matrix** is a (two-dimensional) table of values.

Mathematical notation:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n-1} & a_{1n} \\ a_{21} & a_{22} & & a_{2n-1} & a_{2n} \\ \vdots & & \ddots & \vdots & \\ a_{m-1,1} & a_{m-1,2} & & a_{m-1,n-1} & a_{m-1,n} \\ a_{m1} & a_{m2} & \cdots & a_{m,n-1} & a_{m,n} \end{bmatrix}$$

:Matrix of size $m \times n$

$a_{ij} = a_{i,j} = [\mathbf{A}]_{ij} = [\mathbf{A}]_{i,j}$:Element of matrix $\mathbf{A}$ in row i and column j

- Every entry in the table of the matrix is called the **element** of the matrix.
- Every element of a matrix is identified by the first **row index** i and second **column index** j.
- The **size** of the matrix is the number m of rows and number of columns n: $m \times n$.
- In MATLAB, two-dimensional arrays represent matrices.

2.1. Systems of linear equations (SLEs). Matrix operations

Operations with matrices

- **Scalar multiplication. Product** $\mathbf{C} = b\mathbf{A} = \mathbf{A}b$ of number b and matrix $\mathbf{A}$ of size $m \times n$ is the matrix $\mathbf{C}$ of size $m \times n$, where

$$c_{ij} = ba_{ij}$$

  ✓ Can be calculated for arbitrary $\mathbf{A}$ and b.

- **Matrix addition**: **Sum** $\mathbf{C} = \mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{C}$ of two matrixes $\mathbf{A}$ and B of size $m \times n$ is the matrix $\mathbf{C}$ of the same size $m \times n$, where

$$c_{ij} = a_{ij} + b_{ij}$$

  ✓ Can be calculated only if both matrices $\mathbf{A}$ and $\mathbf{B}$ have the same size $m \times n$.

  ✓ We can define **subtraction**: **Difference** of two matrices is $\mathbf{C} = \mathbf{A} - \mathbf{B} = \mathbf{A} + (-1)\mathbf{B}$.

- **Matrix transpose**: **Transpose** $\mathbf{B} = \mathbf{A}^T$ of matrix $\mathbf{A}$ of size $m \times n$ is the matrix $\mathbf{B}$ of size $n \times m$, where

$$b_{ij} = a_{ji}$$

Examples:

$$b\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix} = \begin{bmatrix} ba_{11} & ba_{12} \\ ba_{21} & ba_{22} \\ ba_{31} & ba_{32} \end{bmatrix},$$

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \end{bmatrix}, \qquad \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}$$

2.1. Systems of linear equations (SLEs). Matrix operations

➢ **Matrix multiplication**: **Product** $\mathbf{C} = \mathbf{AB}$ of two matrices $\mathbf{A}$ and $\mathbf{B}$ of sizes $m \times n$ and $p \times q$ is the matrix $\mathbf{C}$ of size $m \times q$, where

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj} \qquad (2.1.2)$$

✓ Can be calculated only if $n = p$, i.e. if the number of columns in $\mathbf{A}$ is equal to the number of rows in $\mathbf{B}$.

✓ $\mathbf{AB} \neq \mathbf{BA}$ ($\mathbf{AB}$ or $\mathbf{BA}$ may not exist: $\mathbf{A}$: $n \times n$, $\mathbf{B}$: $1 \times n$, $\mathbf{BA}$ exists, $\mathbf{AB}$ does not exist).

Examples:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}, \qquad c_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31}$$

$$\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} = \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix}, \qquad d_{11} = b_{11}a_{11} + b_{12}a_{21}$$

Properties: Can be proved by direct evaluation of the left- and right-hand sides (LHSs and RHSs)

$$c(\mathbf{A} + \mathbf{B}) = c\mathbf{A} + c\mathbf{B}, \qquad \mathbf{C}(\mathbf{A} + \mathbf{B}) = \mathbf{CA} + \mathbf{CB}, \qquad a\mathbf{BC} = (a\mathbf{B})\mathbf{C} = a(\mathbf{BC})$$

$$(\mathbf{A} + \mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T, \qquad (a\mathbf{B})^T = a\mathbf{B}^T, \qquad (\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T$$

Special matrices

Let $\mathbf{A}$ be a matrix of size $m \times n$.

$\mathbf{A}$ is the **row vector** if the number of rows is equal to 1 ($m = 1$): $\mathbf{A} = [a_1 \quad \dots \quad a_n]$, $a_i = a_{1i}$.

$\mathbf{A}$ is the **column vector** if the number of columns is equal to 1 ($n = 1$): $\mathbf{A} = [a_1 \quad \dots \quad a_n]^T$, $a_i = a_{i1}$. Transpose of a column vector is a row vector.

The **zero matrix** $\mathbf{0}$ is the matrix with all zero elements.

$\mathbf{A}$ is the **square matrix** if the number of rows is equal to the number of columns ($n = m$).

The **main diagonal** of the square matrix is the diagonal where $i = j$.

Square matrix $\mathbf{A}$ is called **symmetric** if $a_{ij} = a_{ji}$. Consequence: $\mathbf{A}^T = \mathbf{A}$.

Square matrix $\mathbf{I}$ is called the **identity matrix** if $[\mathbf{I}]_{ii} = 1$, $[\mathbf{I}]_{ij} = 0$ for $i \neq j$.

Properties:

$$\mathbf{0} + \mathbf{A} = \mathbf{A}, \qquad \mathbf{0}\,\mathbf{A} = \mathbf{0}, \qquad \mathbf{I}\mathbf{A} = \mathbf{A}\mathbf{I} = \mathbf{A}\ (\mathbf{A} \text{ is the square matrix})$$

Examples:

$$[a\ b\ c], \quad \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \quad \begin{bmatrix} a & d & e \\ d & b & f \\ e & f & c \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Row Column Symmetric Identity Zero

2.1. Systems of linear equations (SLEs). Matrix operations

Matrix form of a SLE

Every SLE can be written in the matrix notation:

$$\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}, \qquad \mathbf{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} \tag{2.1.3}$$

$\mathbf{A}$, **Matrix of coefficients** of the SLE (of size $m \times n$).

$\mathbf{X}$, **Column vector of n unknowns**.

$\mathbf{B}$, **Column vector** of m **RHSs**.

The matrix form of the SLE given by Eq. (2.3.1):

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

or simply

$$\mathbf{AX} = \mathbf{B} \tag{2.1.4}$$

SLE with square matrix of coefficients

- If the number of equations m is equal to the number of unknowns n, then the matrix of coefficients $\mathbf{A}$ is a square matrix.
- This is the most important case for applications.

2.1. Systems of linear equations (SLEs). Matrix operations

Example: Re-write a SLE in the matrix form

$$\begin{aligned} -2x_3 + 2x_1 + 4x_2 &= 6 \\ 3x_1 + +3x_3 + 3x_2 &= 3 \\ 2x_1 - 2x_2 + 8x_3 &= 4 \end{aligned}$$

First step is to rewrite the system in the form a single matrix equation: two column vectors in the LHS and RHS and sort unknowns in the ascending order:

$$\begin{bmatrix} 2x_1 + 4x_2 - 2x_3 \\ 3x_1 + 3x_2 + 3x_3 \\ 2x_1 - 2x_2 + 8x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 3 \\ 4 \end{bmatrix}$$

Second step is to re-write the LHS in the forma of a product of the matrix of coefficients $\mathbf{A}$ and the column vector of unknowns $\mathbf{X}$:

$$\begin{bmatrix} 2 & 4 & -2 \\ 3 & 3 & 3 \\ 2 & -2 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 3 \\ 4 \end{bmatrix}$$

This is the matrix form of a SLE.

2.2. Solution of SLEs by the Gauss elimination

- Equivalent SLEs
- Forward sweep
- Back substitution

2.2. Solution of SLEs by the Gauss elimination

Equivalent SLEs

The Gauss elimination is the approach that reduces a SLE with the square matrix of coefficients and $\det \mathbf{A} \neq 0$ to an equivalent SLE with the **upper triangular matrix**.

Two SLEs are called **equivalent** if they have the same solution.

In order to transform a SLE to an equivalent SLE with the upper triangular matrix, we will use three rules:

1. **Multiplying** any equation of the SLE by a non-zero constant does not change the solution.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &= b_1 \\ a_{21}x_1 + a_{22}x_2 &= b_2 \end{aligned} \quad \Rightarrow \quad \begin{aligned} ca_{11}x_1 + ca_{12}x_2 &= cb_1 \\ a_{21}x_1 + a_{22}x_2 &= b_2 \end{aligned}$$

: Equivalent SLE if $c \neq 0$

2. **Subtraction** of one equation multiplied by a constant from another equation does not change the solution.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &= b_1 \\ a_{21}x_1 + a_{22}x_2 &= b_2 \end{aligned} \quad \Rightarrow \quad \begin{aligned} a_{11}x_1 + a_{12}x_2 &= b_1 \\ (a_{21} - ca_{11})x_1 + (a_{22} - ca_{12})x_2 &= (b_2 - cb_1) \end{aligned}$$

: Equivalent SLE

3. **Swapping** any pair of equations (changing the order of equations) does not change the solution.

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 &= b_1 \\ a_{21}x_1 + a_{22}x_2 &= b_2 \end{aligned} \quad \Rightarrow \quad \begin{aligned} a_{21}x_1 + a_{22}x_2 &= b_2 \\ a_{11}x_1 + a_{12}x_2 &= b_1 \end{aligned}$$

: Equivalent SLE

2.2. Solution of SLEs by the Gauss elimination

Gauss eliminations includes two major steps:

- **Forward sweep**
- **Back substitution**

Let's consider the Gauss elimination based on the example of a SLE with matrix 3 x 3**.**

$$\begin{aligned} 2x_1 + 4x_2 - 2x_3 &= 6 \\ 3x_1 + 3x_2 + 3x_3 &= 3 \\ 2x_1 - 2x_2 + 8x_3 &= 4 \end{aligned} \qquad \begin{bmatrix} 2 & 4 & -2 \\ 3 & 3 & 3 \\ 2 & -2 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 3 \\ 4 \end{bmatrix}$$

Forward sweep

Let $i = 1$ be the index of an equation in the SLE

1. Take equation i and divide it by a_{ii}

$$\begin{aligned} x_1 + 2x_2 - x_3 &= 3 \\ 3x_1 + 3x_2 + 3x_3 &= 3 \\ 2x_1 - 2x_2 + 8x_3 &= 4 \end{aligned} \qquad \begin{bmatrix} 1 & 2 & -1 \\ 3 & 3 & 3 \\ 2 & -2 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \\ 4 \end{bmatrix}$$

2. Subtract equation i multiplied by a_{ji} from all equations below (equations with $j > i$):

$a_{21} = 3$
$a_{31} = 2$

$$\begin{aligned} x_1 + 2x_2 - x_3 &= 3 \\ -3x_2 + 6x_3 &= -6 \\ -6x_2 + 10x_3 &= -2 \end{aligned} \qquad \begin{bmatrix} 1 & 2 & -1 \\ 0 & -3 & 6 \\ 0 & -6 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ -6 \\ -2 \end{bmatrix}$$

2.2. Solution of SLEs by the Gauss elimination

3. Go to the next equation ($i = i + 1$) and repeat ## 1-2 until $i = n$.

$$\begin{aligned} x_1 + 2x_2 - x_3 &= 3 \\ x_2 - 2x_3 &= 2 \\ -6x_2 + 10x_3 &= -2 \end{aligned}$$

$$\begin{bmatrix} 1 & 2 & -1 \\ 0 & 1 & -2 \\ 0 & -6 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ -2 \end{bmatrix} \qquad i = 2$$

$$\begin{aligned} x_1 + 2x_2 - x_3 &= 3 \\ x_2 - 2x_3 &= 2 \\ -2x_3 &= 10 \end{aligned}$$

$$\begin{bmatrix} 1 & 2 & -1 \\ 0 & 1 & -2 \\ 0 & 0 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 10 \end{bmatrix}$$

$$\begin{aligned} x_1 + 2x_2 - x_3 &= 3 \\ x_2 - 2x_3 &= 2 \\ x_3 &= -5 \end{aligned}$$

$$\begin{bmatrix} 1 & 2 & -1 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ -5 \end{bmatrix} \qquad i = 3$$

An equivalent SLE **with upper triangular matrix** of coefficients (all elements below the main diagonal are zero)

In this example, swapping of equations is not necessary. In general, swapping is during the forward sweep if a coefficient on the main diagonal become equal to zero.

2.2. Solution of SLEs by the Gauss elimination

Back substitution

1. Take the last equation ($i = n$) and find x_i

$$x_1 + 2x_2 - x_3 = 3$$
$$x_2 - 2x_3 = 2 \qquad i = 3$$
$$x_3 = -5$$

2. Go to the previous equation ($i = i - 1$) and repeat #1 until $i = 1$

$$x_1 + 2x_2 - x_3 = 3$$
$$x_2 = 2 + 2x_3 = -8 \qquad i = 2$$
$$x_3 = -5$$

$$x_1 = 3 - 2x_2 + x_3 = 14$$
$$x_2 = 2 + 2x_3 = 12 \qquad i = 1$$
$$x_3 = 5$$

- Both forward sweep and back substitution can be coded for SLEs with square matrices of arbitrary dimension n.
- Examples of MATLAB codes implementing the Gauss elimination are posted on the Blackboard Learn. See MATLAB Codes/Chapter_2/Section_2_02.

2.3. Two-dimensional arrays and operations with matrices in MATLAB

- Two dimensional arrays in MATLAB
- Creation of two-dimensional arrays
- Column and row vectors. Transpose operation
- Matrix operations in MATLAB
- Accessing blocks of elements in one- and two-dimensional arrays

Reading assignment

Gilat, 3.1, 3.2, 3.4, and 3.5

MATLAB two-dimensional arrays

Dimension of the array is the number of indexes of its elements; **Size** is the number of elements.

- One-dimensional (1D) array T_i of size m is also called the **vector**.
- Two-dimensional (2D) array $T_{i,j}$ of size m x n is also called the **matrix** or **table**.
- MATLAB 2D arrays can be introduced and used similarly to 1D arrays.

Example: **Matrix with 5 x 60 elements**:

Row i

T(1,1)	T(1,2)	....	T(1,j-1)	T(1,j)	T(1,j+1)	...	T(1,59)	T(1,60)
				...				
T(i,1)	T(i,2)	....	T(i,j-1)	T(i,j)	T(i,j+1)	...	T(i,59)	T(i,60)
				...				
T(5,1)	T(5,2)	....	T(5,j-1)	T(5,j)	T(5,j+1)	...	T(5,59)	T(5,60)

Column j

T = **Name** of the array consisting of 5 x 60 scalar values

T(i,j) = **Element** of array T = scalar variable

i = First integer index of element T(i,j) of array T = **Row index**

j = Second integer index of element T(i,j) of array T = **Column index**

m x n = 5 x 60 is the **Size** of the array. m is the number of row. n is the number of columns.

2.3. Two-dimensional arrays and operations with matrices in MATLAB

Creation of two-dimensional arrays

- When we create a 2D array we indicate its values row by row (we say "last index varies first")
- *Semicolon* **;** divides values corresponding to different rows
- **Explicit definition of every element of the array with square brackets []**

  Example: x = [0.1 2*pi 2^3 ; -3 2 17]

$$\mathbf{x} = \begin{bmatrix} 0.1 & 2\pi & 8 \\ -3 & 2 & 17 \end{bmatrix}$$

- **Create an array with equal spacing between neighbor points using square brackets []**

  x = [m1:n1; m2:n2] : Creates array x with 2 rows. Number of columns and initial values in every row is given by mi:ni. Number of columns is every row should be the same.

- **Create a specific array with build-in functions zeros, ones, and eye**

  x = zeros (m, n) : Creates array x of m rows and n columns, where x(i,j) = 0

  x = ones (m, n) : Creates array x of m rows and n columns, where x(i,j) = 1

  x = eye (m) : Creates array x of m rows and m columns, where x(i,i) = 1, while x(i,j)=0 if i≠j.

- **Create an array based on another array**

  y = sin (x) : Creates array y, which has the same number of elements as the array x and y(i,j) = sin (x(i,j)).

Column and row vectors. Transpose operation

- Every 1D array can be considered as 2D array where the number of either rows or columns is equal to 1
- **Row vector** = array where the number of rows is equal to 1

x = [1 2 3] $\quad\quad$ $\mathbf{x} = [1 \quad 2 \quad 3]$

- **Column vector** = array where the number of columns is equal to 1

y = [1 ; 2 ; 3] $\quad\quad$ $\mathbf{y} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

- It is not possible to calculate **z = x + y** because **x** and **y** have different sizes (dimensions)
- We can transform row vector to column and vice versa with the **transpose operation '**

z = **x** + **y**'

- All vectorized arythmetics can be used in calculations with 2D arrays

Problem 2.3.1: Create a matrix A of size 3 x 3, where a(i,i)=7, a(i,j)=-1 if i≠ *j*

Solution: A = (2.0 * eye (3)).^3 - 1.0

Matrix operations in MATLAB

- **Scalar multiplication**: a = 2 ; b = [1 2 ; 3 4] ; c = a * b ;
- **Addition (subtraction)**: d = [-1 2 ; -3 4]; f = d + b ;
- **Transpose**: g = [1 2 3 ; 3 4 5] ; h = g' ;
- **Matrix multiplication**: w = b * g ;
- **Matrix power** (*only for square matrices*): p = b^3 is equivalent to p = b * b * b.

- All these matrix operations are performed based on the rules for matrix operations formulated in linear algebra, See slides 7-8.

- Function [m n] = **size** (A) returns the numbers of rows m and columns n of matrix A.

- There are a lot of other build-in functions for manipulating matrices: See Gilat 3.5.

1.4. MATLAB arrays

Accessing blocks of elements in one- and two-dimensional arrays

- Any **block** of array elements can be used in order to form a new array (subarray).
- To access a continuous block of elements at one time MATLAB provides **colon notation**: x(2:5) is the block of four elements x(2), x(3), x(4), and x(5).
- **x(m:q:n)** says "form a block starting with element m, counting by q (i=i+q), and stopping at n." So we get the block of x(m), x(m+q), x(m+2q), ..., x(n).
- **x ([6 2 5])** says "form a block of three elements x(6), x(2), and x(5).
- A block of elements can be used as new array: y = x ([6 2 5]).

Z(1)	Z(2)	Z(3)
0.0	-1.6	-0.4

x(1)	x(2)	x(3)	x(4)	x(5)	x(6)	x(7)	x(8)	x(9)	x(10)	x(11)
-2	-1.6	-1.2	-0.8	-0.4	0.0	0.4	0.8	1.2	1.6	2.0

Y

Problem 1.4.1: Create two array containing elements 5, 6, 7 and 6, 2, 5 of an original array X.

File Problem_1_4_1.m

```
x = [ -2 : 0.4 : 2 ]
Y = x(5:7)
Z = x ( [ 6 2 5 ] )
```

1.4. MATLAB arrays

- In 2D array, any rectangular block of elements can be addres by specifying the range of row and column indices using colon ':'. Examples:

  x(:,n) Refers to the block of elements in all the rows of column n.

  x(m:n,:) Refers to the block of elements in all the columns between row m and row n.

Problem 1.4.2: For 3 x 3 matrix A with elements $a_{ij} = 3(i-1)+j$, create sub-arrays X containing the first row, Y containing the second column and 2 x 2 matrix Z containing submatrix $\mathbf{A}_{11}$.

File Problme_1_4_2.m

```
A = [ 1 : 9 ];
A = reshape ( A, 3, 3 )' % See description of reshape function in Gilat's book
X = A(1,:)
Y = A(:,2)
Z = A(2:3,2:3)
```

Y

X

1	2	3
4	5	6
7	8	9

1	2	3
4	5	6
7	8	9

Z

2.4. Determinant and matrix inverse

- Determinant
- Matrix inverse
- Cramer's rule
- Solution of a SLE with a square matrix of coefficients

Reading assignment

http://en.wikipedia.org/wiki/Determinant

http://en.wikipedia.org/wiki/Cofactor_(linear_algebra)

http://en.wikipedia.org/wiki/Cramer's_rule

Gilat, 3.5

Determinant

The **determinant** $\det \mathbf{A}$ of a square matrix $\mathbf{A}$ is a uniquely defined numerical value that is calculated based on the elements of the matrix with special rules.

Notation:

$$\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}, \qquad \det \mathbf{A} = \det(\mathbf{A}) = |\mathbf{A}| = \begin{vmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{vmatrix}$$

1. For every element a_{ij} of matrix $\mathbf{A}$ of size $n \times n$, let's introduce a square **submatrix** $\mathbf{A}_{ij}$ of size $(n-1) \times (n-1)$, which is obtained from $\mathbf{A}$ by removing all elements in row i and column j. The (**first**) **minor** of $\mathbf{A}$ corresponding to a_{ij} is $M_{ij} = \det \mathbf{A}_{ij}$.
2. For every element a_{ij} of matrix $\mathbf{A}$, let's introduce the **signature** s_{ij} that is equal to

$$s_{ij} = (-1)^{i+j} = \begin{cases} 1 & \text{if the number of "steps" from } a_{11} \text{ to } a_{ij} \text{ along rows and columns is even} \\ -1 & \text{otherwise (number of "steps" is odd)} \end{cases}$$

3. The **cofactor** of a_{ij} is $C_{ij} = s_{ij}|\mathbf{A}_{ij}|$.

Example:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \quad \mathbf{A}_{12} = \begin{bmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{bmatrix}, \quad \mathbf{S} = [s_{ij}] = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix}, \quad C_{12} = s_{12}|\mathbf{A}_{12}|$$

2.4. Determinant and matrix inverse

There are a lot of ways to introduce the rules that determine the values of the determinant. We will consider not the best, but simplest **iterative rule** of introducing the determinant. It include only two steps:

1. Determinant of the square matrix of size 1 x 1 is its element: $\det[a] = a$.

2. The determinant of the square matrix of arbitrary size can be calculated as:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$\det \mathbf{A} = |\mathbf{A}| = a_{11}C_{11} + a_{12}C_{12} + \cdots + a_{1n}C_{1n} = \sum_{j=1}^{n} a_{1j}C_{1j} \qquad (2.4.1)$$

- Eq. (2.4.1) reduces calculation of the determinant of a matrix of size $n \times n$ to determinants of matrices of smaller size $(n-1) \times (n-1)$.
- Now we can apply the same rule for $C_{1j} = s_{1j}|\mathbf{A}_{1j}|$ and reduces $|\mathbf{A}_{1j}|$ to determinants of matrices of size $(n-2) \times (n-2)$ etc. until we get matrixes of size 1 x 1 , i.e. just numbers.

2.4. Determinant and matrix inverse

Example: Determinant of matrix 2 x 2

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

Example: Determinant of matrix 3 x 3.

$$\begin{vmatrix} a & b & c \\ d & e & f \\ g & h & i \end{vmatrix} = a\begin{vmatrix} e & f \\ h & i \end{vmatrix} - b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix} = a(ei - fh) - b(di - gf) + c(dh - eg)$$

Basic properties of the determinant

- $\det \mathbf{A}^T = \det \mathbf{A}$.
- $\det \mathbf{0} = 0$.
- $\det(\mathbf{AB}) = \det \mathbf{A} \det \mathbf{B}$ if $\mathbf{A}$ and $\mathbf{B}$ are square matrices of the same size.

Determinant in MATLAB

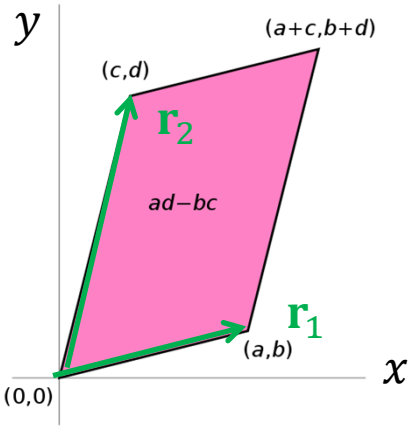
- Function **det** (A) returns the determinant of the square matrix A.

2.4. Determinant and matrix inverse

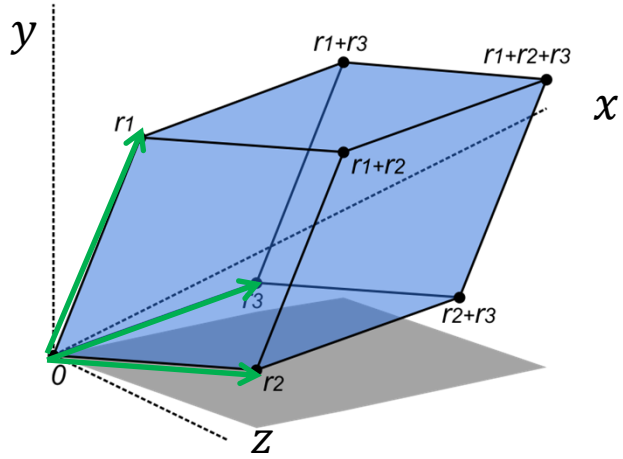
- Calculation of $\det \mathbf{A}$ based on our definition for large n is extremely lengthy procedure. It requires $N \sim n!$ individual arithmetic operations: $n = 10$, $N \sim 3.6 \times 10^6$; $n = 100$, $N \sim 9.3 \times 10^{157}$.

- For practical calculations of determinants at large n, the **decomposition methods** are used. They require $N \sim n^3$ of arithmetic operations: $n = 10, N \sim 10^3$; $n = 100, N \sim 10^6$.

Geometrical meaning of determinants of 2 x 2 and 3 x 3 matrices



$$\mathbf{r}_1 = (x_1, y_1)$$
$$\mathbf{r}_2 = (x_2, y_2)$$



$$\mathbf{r}_1 = (x_1, y_1, z_1)$$
$$\mathbf{r}_2 = (x_2, y_2, z_2)$$
$$\mathbf{r}_3 = (x_3, y_3, z_3)$$

$\det \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \end{bmatrix}$ = signed (±) area of parallelogram

$\det \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix}$ = signed (±) volume of parallelepiped

2.4. Determinant and matrix inverse

Matrix inverse

Let's consider a square matrix $\mathbf{A}$ of size $n \times n$. The matrix $\mathbf{A}^{-1}$ of size $n \times n$ is called the **matrix inverse** to $\mathbf{A}$ or the **inverse matrix** if ($\mathbf{I}$ is the identity matrix, see slide 9)

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{A}\mathbf{A}^{-1} = \mathbf{I} \tag{2.4.2}$$

Let's apply the properties of the determinant ($\det(\mathbf{AB}) = (\det \mathbf{A})\,(\det \mathbf{B})$, slide 25):

$$\det\left(\mathbf{A}^{-1}\mathbf{A}\right) = \det \mathbf{I}$$

$$(\det \mathbf{A}^{-1})(\det \mathbf{A}) = 1$$

- The matrix $\mathbf{A}$ is **invertible** (has the inverse) only if $\det \mathbf{A} \neq 0$.
- The matrix $\mathbf{A}$ is called **singular** or **degenerate** if $\det \mathbf{A} = 0$.
- If the matrix $\mathbf{A}$ is invertible, then

$$\det \mathbf{A}^{-1} = \frac{1}{\det \mathbf{A}} \tag{2.4.3}$$

- There are a lot of paths to practically find the inverse matrix, e.g., **Cramer's rule**.

Matrix inverse in MATLAB

- Function **inv** (A) returns the inverse of the square matrix A. **inv** (A) = A^-1.

2.4. Determinant and matrix inverse

Cramer's rule

Cramer's rule says that the inverse $\mathbf{A}^{-1}$ of any invertible square matrix $\mathbf{A}$ can be found in the form:

$$\mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}, \quad \det \mathbf{A} \neq 0, \quad \text{then} \quad \boxed{\mathbf{A}^{-1} = \frac{1}{\det \mathbf{A}} \begin{bmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{n1} & \cdots & C_{nn} \end{bmatrix}^T} \quad (2.4.4)$$

where C_{ij} are cofactors of elements of matrix $\mathbf{A}$.

According to Cramer's rule, we need to perform three steps in order to find $\mathbf{A}^{-1}$:

1. To calculate cofactors C_{ij} for all elements of $\mathbf{A}$ and form the matrix

$$\mathbf{C} = \begin{bmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{n1} & \cdots & C_{nn} \end{bmatrix}$$

2. To find the transpose of $\mathbf{C}$:

$$\mathbf{C}^T = \begin{bmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{n1} & \cdots & C_{nn} \end{bmatrix}^T = \begin{bmatrix} C_{11} & \cdots & C_{n1} \\ \vdots & \ddots & \vdots \\ C_{1n} & \cdots & C_{nn} \end{bmatrix}$$

3. To calculate determinant $\det \mathbf{A}$ and divide $\mathbf{C}^T$ by $\det \mathbf{A}$:

$$\mathbf{A}^{-1} = \frac{1}{\det \mathbf{A}} \mathbf{C}^T$$

2.4. Determinant and matrix inverse

Example: Matrix 2 x 2

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \quad \det \mathbf{A} \neq 0$$

1.

$$\mathbf{C} = \begin{bmatrix} a_{22} & -a_{21} \\ -a_{12} & a_{11} \end{bmatrix}$$

2.

$$\mathbf{C}^T = \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}$$

3.

$$\det \mathbf{A} = a_{11}a_{22} - a_{12}a_{21}$$

$$\mathbf{A}^{-1} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}$$

- Finding the matrix inverse with Cramer's rule is a lengthy operation. It requires calculation of n^2 of determinants of size $(n-1) \times (n-1)$. If the determinants are calculated according to the expansion formula, Eq. (2.2.1), then the total number of arithmetic operations $N \sim n^2 \times (n-1)! = n \times (n!)$.
- In practical calculations, the **decomposition methods** are used to calculate minors, and the **time complexity** of Cramer's rule drops down to $N \sim n^4$ and even to $N \sim n^3$.

Solution of a SLE with a square matrix of coefficients

Consider a SLE with square matrix of coefficients

$$\mathbf{AX} = \mathbf{B}, \qquad \mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}, \qquad \mathbf{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

Theorem:

If $n = m$, then a unique solution of the SLE exists if $\det \mathbf{A} \neq 0$.

Proof:

If $\det \mathbf{A} \neq 0$, then the square matrix has the inverse $\mathbf{A}^{-1}$, such that $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$.

Let's then multiple the SLE from the left by $\mathbf{A}^{-1}$:

$$\mathbf{A}^{-1}\mathbf{AX} = \mathbf{A}^{-1}\mathbf{B} \quad \Rightarrow \quad \boxed{\mathbf{X} = \mathbf{A}^{-1}\mathbf{B}} \qquad (2.3.4)$$

We prove that al least one solution exists. It can be also proved that this solution is unique.

- Eq. (2.3.4) shows us how to find the solution of a SLE with the square matrix of coefficients: First we need to find the inverse of $\mathbf{A}$ and then to multiply it by the RHS vector $\mathbf{B}$.
- A SLE is called **homogeneous** if $\mathbf{B} = \mathbf{0}$. The theorem says that the homogeneous SLE with $\det \mathbf{A} \neq 0$ has only the **trivial solution** $\mathbf{X} = \mathbf{0}$.
- If $\det \mathbf{A} = 0$ then the SLE can have multiple solutions or may not have them at all.

2.4. Determinant and matrix inverse

Example: $\det \mathbf{A} = 0$ means that in one or a few pairs of individual equations in the SLE the left hand sides are "proportional" to each other (this is not an accurate statement):

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ -2 & -4 \end{bmatrix}, \qquad \det \mathbf{A} = 0$$

$$\begin{bmatrix} 1 & 2 \\ -2 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \begin{matrix} x_1 = -2x_2 \\ x_1 = -2x_2 \end{matrix}$$ ⇒ The SLE has multiple solutions

$$\begin{bmatrix} 1 & 2 \\ -2 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \Rightarrow \begin{matrix} x_1 = 1 - 2x_2 \\ x_1 = -0.5 - 2x_2 \end{matrix}$$ ⇒ The SLE has no solutions

2.5. Solution of SLEs in the MATLAB

- Five ways to solve a SLE in MATLAB
- Calculation of forces in the statically determined systems

Reading assignment

Gilat, 3.3

2.5. Solution of SLEs in MATLAB

Five ways to solve SLEs in MATLAB

Consider a SLE with square matrix of coefficients

$$\mathbf{AX} = \mathbf{B}, \qquad \mathbf{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}, \qquad \mathbf{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

Assume we coded $\mathbf{A}$ as a 2D array A, $\mathbf{B}$ as a column vector B. We can solve the SLE in MATLAB

1. Via the inverse matrix

X = **inv** (A) * B = A^-1 * B

2. Via the **left matrix division**

X = A \ B

3. Via the **right matrix division**

X = (B' / A')'

4. Via the function **mldivide**

X = **mldivide** (A, B)

5. Via the build-in MATLAB function **linsolve**

X = **linsolve** (A, B)

A \ B =**inv**(A)*B=A^-1*B
if B is a column vector and A is a square matrix.

B / A = B***inv**(A)=B*A^-1
if B is a row vector and A is a square matrix.

2.5. Solution of SLEs in MATLAB

Problem 2.5.1: Solve a SLE

$$\begin{aligned} 2x_1 + 4x_2 - 2x_3 &= 6 \\ 3x_1 + 3x_2 + 3x_3 &= 3 \\ 2x_1 - 2x_2 + 8x_3 &= 4 \end{aligned} \qquad \begin{bmatrix} 2 & 4 & -2 \\ 3 & 3 & 3 \\ 2 & -2 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 6 \\ 3 \\ 4 \end{bmatrix}$$

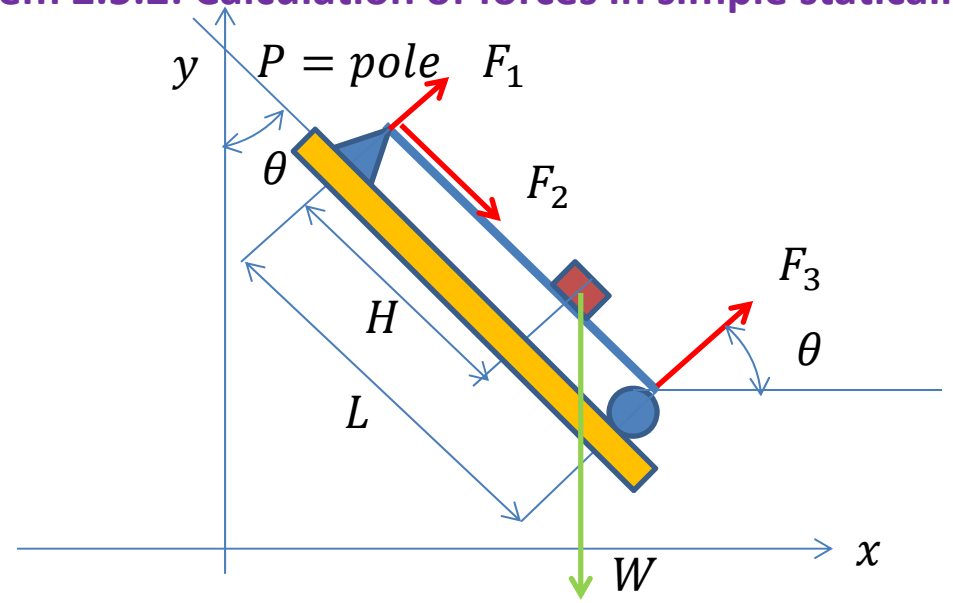
Solution:

```
A = [ 2, 4, -2 ; 3, 3, 3 ; 2, -2, 8 ];
B = [ 6, 3, 4 ]';
X = inv ( A ) * B % or X = A^-1 * B % or X = A \ B % or X = ( B' / A' )' or X = linsolve ( A, B )
```

- **mldivide** is completely equivalent to the left matrix division: **mldivide** (A, B) = A \ B.
- **mldivide** analyzes the type of the matrix of coefficients and employs different solvers to handle different kinds of coefficient matrices.
- **linsolve** has an optional parameter opts: **linsolve** (A, B, opts). This parameter allows one to specify the type of the matrix (upper triangular, etc.) and to speed up calculations for a specific type of matrices. See http://www.mathworks.com/help/matlab/ref/linsolve.html.
- **mldivide** and **linsolve** can be used for systems with non-square matrices or for system with square matrix with zero determinants. Then they can return some "generalized" solutions.
- **It is a good practice to check that**
  - $\det(A) \neq 0$ **and SLE has a unique solution.**
  - **X is an actual solution, i.e., calculate A * X - B and see that it is the zero vector**.

2.5. Solution of SLEs in MATLAB

Problem 2.5.2: Calculation of forces in simple statically determined (isostatic) system



Equations of equilibrium

$$\sum_i \mathbf{F}_i = 0$$

$$\sum_i \mathbf{M}_{(\boldsymbol{P})i} = 0$$

- Choose coordinates
- Choose direction of forces
- Choose pole to calculate torque

$$\sum F_{xi} = 0 \qquad F_1 \cos\theta + F_2 \sin\theta + F_3 \cos\theta = 0$$

$$\sum F_{yi} = 0 \qquad F_1 \sin\theta - F_2 \cos\theta + F_3 \sin\theta - W = 0$$

$$\sum M_{(P)zi} = 0 \qquad F_3 L - WH \sin\theta = 0$$

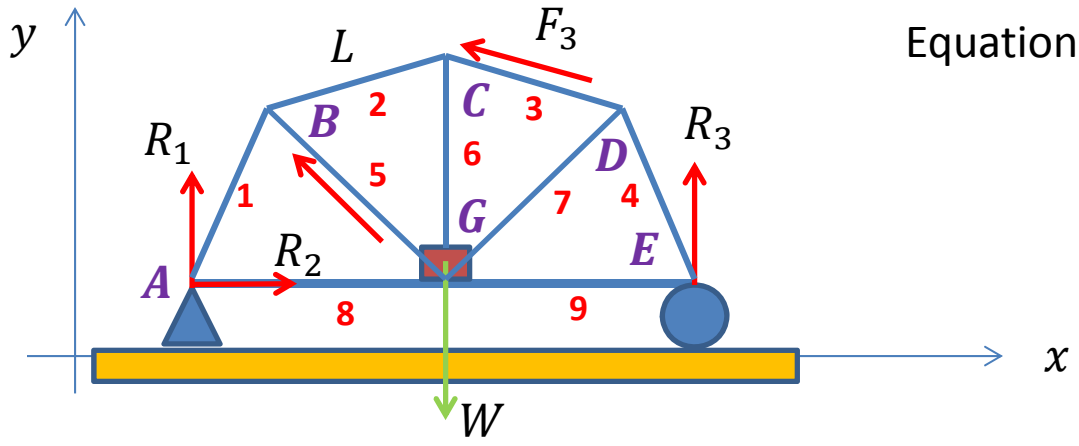
$$\begin{bmatrix} \cos\theta & \sin\theta & \cos\theta \\ \sin\theta & -\cos\theta & \sin\theta \\ 0 & 0 & L \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix} = \begin{bmatrix} 0 \\ W \\ WH\sin\theta \end{bmatrix}$$

2.5. Solution of SLEs in MATLAB

```
Th = 45.0; L = 1.0; H = 0.75; W = 100.0;
A = [ cosd ( Th ) sind ( Th ) cosd ( Th ) ; sind ( th ) ( - cosd ( Th ) ) sind ( Th ) ; 0 0 L ];
B = [ 0 ; W; W * H * sind ( Th ) ];
F = A \ B   % or
F = A^-1 * B % or
F = linsolve ( A, B )
A * F - B % This is a check. If solution of the SLE is correct, this vector is zero
```

2.5. Solution of SLEs in MATLAB

Problem 2.5.3: Calculation of forces in a isostatic truss



Equations of equilibrium for the **whole truss**

$$\sum_i \mathbf{F}_i = 0$$

$$\sum_i \mathbf{M}_i = 0$$

- We need to find totally 12 forces $R_1, R_2, R_3, F_1, \ldots, F_9$.
- Three equation of equilibrium for the whole truss has the same form as for the simple system in problem 2.5.2

$$\sum F_{xi} = 0, \qquad \sum F_{yi} = 0, \qquad \sum M_{zi} = 0$$

- Additional equation can be obtained from **equilibrium of every joint** $A, B, \ldots, G$.

  For instance, for joint G:

  x-component of force: $-F_8 - F_5 \cos 45^\circ + F_7 \cos 45^\circ + F_9 = 0$

  y-component of force: $F_5 \sin 45^\circ + F_6 + F_7 \sin 45^\circ - W = 0$
- **Total number of equations should be equal to the total number of unknowns (forces).**

2.6. Summary

For the exam we must know how

- To make basic matrix operations (scalar multiplication, addition, matrix multiplication, and matrix transpose) by hand and in the MATLAB.
- To define arbitrary and special matrixes (column vector, row vector, square, zero, identity, symmetric) in the MATLAB.
- To form block of elements of one- and two-dimensional arrays.
- To use function **size** in order to determine the size of two-dimensional arrays.
- To perform operations with matrices by hand and in MATLAB.
- To calculated the determinants of matrices 2 x 2 and 3 x 3 by hand and determinants of arbitrary square matrices with the **det** MATLAB function.
- To calculated the inverse of matrices 2 x 2 and 3 x 3 by hand with Cramer's rule and inverse of arbitrary square matrices with the **inv** MATLAB function.
- To write a system of linear equations (SLE) in the matrix form.
- To solve SLE by hand using the Gauss elimination.
- To solve SLE in the MATLAB by utilizing left/right matrix division, calculation of the matrix inverse, and functions **linsolve** and **mldivide**.
- To formulate a SLE defining forces in a statically determined truss and to solve this SLE in MATLAB.