Linear equations 1

Linear algebra basics

Dr.ir. Ivo Roghair, Prof.dr.ir. Martin van Sint Annaland

Chemical Process Intensification group
Eindhoven University of Technology

Numerical Methods (6BER03), 2024-2025

Today's outline

- Introduction
- Matrix inversion
- Solving a linear system
- Towards larger systems
- Summary

Today's outline

- Introduction
- Matrix inversion
- Solving a linear system
- Towards larger systems
- Summary

Overview

Goals

- Different ways of looking at a system of linear equations
- Determination of the inverse, determinant and the rank of a matrix
- The existence of a solution to a set of linear equations

Different views of linear systems

- Separate equations:

$$\begin{aligned} x + y + z &= 4 \\ 2x + y + 3z &= 7 \\ 3x + y + 6z &= 5 \end{aligned}$$

Different views of linear systems

- Separate equations:

$$\begin{aligned} x + y + z &= 4 \\ 2x + y + 3z &= 7 \\ 3x + y + 6z &= 5 \end{aligned}$$

- Matrix mapping $Mx = b$:

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 3 \\ 3 & 1 & 6 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 4 \\ 7 \\ 5 \end{bmatrix}$$

Different views of linear systems

- Separate equations:

$$\begin{aligned} x + y + z &= 4 \\ 2x + y + 3z &= 7 \\ 3x + y + 6z &= 5 \end{aligned}$$

- Matrix mapping $Mx = b$:

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 3 \\ 3 & 1 & 6 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 4 \\ 7 \\ 5 \end{bmatrix}$$

- Linear combination:

$$x \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + y \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + z \begin{bmatrix} 1 \\ 3 \\ 6 \end{bmatrix} = \begin{bmatrix} 4 \\ 7 \\ 5 \end{bmatrix}$$

Different views of linear systems

- Separate equations:

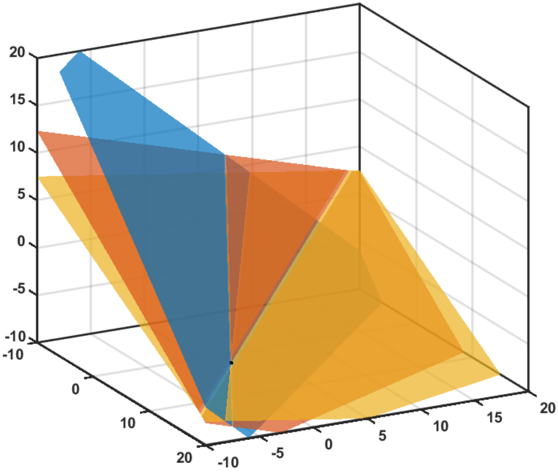
$$\begin{aligned} x+y+z&=4\\ 2x+y+3z&=7\\ 3x+y+6z&=5 \end{aligned}$$

- Matrix mapping $Mx=b$:

$$\begin{bmatrix}1&1&1\\2&1&3\\3&1&6\end{bmatrix}\begin{bmatrix}x\\y\\z\end{bmatrix}=\begin{bmatrix}4\\7\\5\end{bmatrix}$$

- Linear combination:

$$x\begin{bmatrix}1\\2\\3\end{bmatrix}+y\begin{bmatrix}1\\1\\1\end{bmatrix}+z\begin{bmatrix}1\\3\\6\end{bmatrix}=\begin{bmatrix}4\\7\\5\end{bmatrix}$$



Today's outline

- Introduction
- Matrix inversion
- Solving a linear system
- Towards larger systems
- Summary

Inverse of a matrix

- The inverse M^{-1} is defined such that:

$$MM^{-1} = I \quad \text{and} \quad M^{-1}M = I$$

- Use the inverse to solve a set of linear equations:

$$\begin{aligned} M\boldsymbol{x} &= \boldsymbol{b} \\ M^{-1}M\boldsymbol{x} &= M^{-1}\boldsymbol{b} \\ I\boldsymbol{x} &= M^{-1}\boldsymbol{b} \\ \boldsymbol{x} &= M^{-1}\boldsymbol{b} \end{aligned}$$

How to calculate the inverse?

- The inverse of an $N \times N$ matrix can be calculated using the co-factors of each element of the matrix:

$$M^{-1} = \frac{1}{\det\left|M\right|}\begin{bmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{bmatrix}^T$$

- $\det\left|M\right|$ is the *determinant* of matrix M.
- C_{ij} is the *co-factor* of the ij^{th} element in M.

Computing the co-factors

Consider the following example matrix: $M = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 3 \\ 3 & 1 & 6 \end{bmatrix}$

Computing the co-factors

Consider the following example matrix: $M = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 3 \\ 3 & 1 & 6 \end{bmatrix}$

A co-factor (e.g. C_{11}) is the determinant of the elements left over when you cover up the row and column of the element in question, multiplied by ± 1, depending on the position.

$$\begin{bmatrix} 1 & \times & \times \\ \times & 1 & 3 \\ \times & 1 & 6 \end{bmatrix}$$

Computing the co-factors

Consider the following example matrix: $M = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 3 \\ 3 & 1 & 6 \end{bmatrix}$

A co-factor (e.g. C_{11}) is the determinant of the elements left over when you cover up the row and column of the element in question, multiplied by ± 1, depending on the position.

$$\begin{bmatrix} 1 & \times & \times \\ \times & 1 & 3 \\ \times & 1 & 6 \end{bmatrix} \qquad \begin{bmatrix} + & - & + \\ - & + & - \\ + & - & + \end{bmatrix}$$

Computing the co-factors

Consider the following example matrix: $M = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 3 \\ 3 & 1 & 6 \end{bmatrix}$

A co-factor (e.g. C_{11}) is the determinant of the elements left over when you cover up the row and column of the element in question, multiplied by ± 1, depending on the position.

$$\begin{bmatrix} 1 & \times & \times \\ \times & 1 & 3 \\ \times & 1 & 6 \end{bmatrix} \qquad \begin{bmatrix} + & - & + \\ - & + & - \\ + & - & + \end{bmatrix} \qquad C_{11} = +1 \cdot \det \begin{vmatrix} 1 & 3 \\ 1 & 6 \end{vmatrix}$$

$$= 6 \times 1 - 3 \times 1 = 3$$

Computing the co-factors

Back to our example:

$$M^{-1} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 3 \\ 3 & 1 & 6 \end{bmatrix}^{-1} = \frac{1}{\det\left|M\right|} \begin{bmatrix} 3 & -3 & -1 \\ -5 & 3 & 2 \\ 2 & -1 & -1 \end{bmatrix}^{T}$$

Computing the co-factors

Back to our example:

$$M^{-1} = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 3 \\ 3 & 1 & 6 \end{bmatrix}^{-1} = \frac{1}{\det\left|M\right|} \begin{bmatrix} 3 & -3 & -1 \\ -5 & 3 & 2 \\ 2 & -1 & -1 \end{bmatrix}^{T}$$

- The determinant is very important
- If $\det\left|M\right| = 0$, the inverse does not exist (singular matrix)

Calculating the determinant

Compute the determinant by multiplication of each element on a row (or column) by its cofactor and adding the results:

$$\det\left|\begin{bmatrix} \boxed{1 \quad 1 \quad 1} \\ 2 \quad 1 \quad 3 \\ 3 \quad 1 \quad 6 \end{bmatrix}\right| = +\det\left|\begin{bmatrix} 1 & 3 \\ 1 & 6 \end{bmatrix}\right| - \det\left|\begin{bmatrix} 2 & 3 \\ 3 & 6 \end{bmatrix}\right| + \det\left|\begin{bmatrix} 2 & 1 \\ 3 & 1 \end{bmatrix}\right| = -1$$

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Calculating the determinant

Compute the determinant by multiplication of each element on a row (or column) by its cofactor and adding the results:

$$\det\left|\begin{bmatrix}1 & 1 & 1\\ 2 & 1 & 3\\ 3 & 1 & 6\end{bmatrix}\right| = +\det\left|\begin{bmatrix}1 & 3\\ 1 & 6\end{bmatrix}\right| - \det\left|\begin{bmatrix}2 & 3\\ 3 & 6\end{bmatrix}\right| + \det\left|\begin{bmatrix}2 & 1\\ 3 & 1\end{bmatrix}\right| = -1$$

$$\det\left|\begin{bmatrix}1 & 1 & 1\\ 2 & 1 & 3\\ 3 & 1 & 6\end{bmatrix}\right| = +\det\left|\begin{bmatrix}2 & 1\\ 3 & 1\end{bmatrix}\right| - 3\det\left|\begin{bmatrix}1 & 1\\ 3 & 1\end{bmatrix}\right| + 6\det\left|\begin{bmatrix}1 & 1\\ 2 & 1\end{bmatrix}\right| = -1$$

Today's outline

- Introduction
- Matrix inversion
- Solving a linear system
- Towards larger systems
- Summary

Solving a linear system

- Our example:

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 3 \\ 3 & 1 & 6 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 4 \\ 7 \\ 5 \end{bmatrix}$$

Solving a linear system

- Our example:

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 3 \\ 3 & 1 & 6 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 4 \\ 7 \\ 5 \end{bmatrix}$$

- The solution is:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = M^{-1} b = \frac{1}{-1} \begin{bmatrix} 3 & -5 & 2 \\ -3 & 3 & -1 \\ -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} 4 \\ 7 \\ 5 \end{bmatrix} = \frac{1}{-1} \begin{bmatrix} -13 \\ 4 \\ 5 \end{bmatrix} = \begin{bmatrix} 13 \\ -4 \\ -5 \end{bmatrix}$$

Solving a linear system

- Our example:

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & 3 \\ 3 & 1 & 6 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 4 \\ 7 \\ 5 \end{bmatrix}$$

- The solution is:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = M^{-1} b = \frac{1}{-1} \begin{bmatrix} 3 & -5 & 2 \\ -3 & 3 & -1 \\ -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} 4 \\ 7 \\ 5 \end{bmatrix} = \frac{1}{-1} \begin{bmatrix} -13 \\ 4 \\ 5 \end{bmatrix} = \begin{bmatrix} 13 \\ -4 \\ -5 \end{bmatrix}$$

- The inverse exists, because $\det \left| M \right| = -1$.

Solving a linear system in Python using the inverse

- Create the matrix:

```
>>> A = np.array([[1, 1, 1], [2, 1, 3], [3, 1, 6]])
```

Solving a linear system in Python using the inverse

- Create the matrix:

```
>>> A = np.array([[1, 1, 1], [2, 1, 3], [3, 1, 6]])
```

- Create solution vector:

```
>>> b = np.array([4, 7, 5])
```

Solving a linear system in Python using the inverse

- Create the matrix:

```
>>> A = np.array([[1, 1, 1], [2, 1, 3], [3, 1, 6]])
```

- Create solution vector:

```
>>> b = np.array([4, 7, 5])
```

- Get the matrix inverse:

```
>>> Ainv = np.linalg.inv(A)
```

Solving a linear system in Python using the inverse

- Create the matrix:

```
>>> A = np.array([[1, 1, 1], [2, 1, 3], [3, 1, 6]])
```

- Create solution vector:

```
>>> b = np.array([4, 7, 5])
```

- Get the matrix inverse:

```
>>> Ainv = np.linalg.inv(A)
```

- Compute the solution:

```
>>> x = np.dot(Ainv, b)
```

Solving a linear system in Python using the inverse

- Create the matrix:

```
>>> A = np.array([[1, 1, 1], [2, 1, 3], [3, 1, 6]])
```

- Create solution vector:

```
>>> b = np.array([4, 7, 5])
```

- Get the matrix inverse:

```
>>> Ainv = np.linalg.inv(A)
```

- Compute the solution:

```
>>> x = np.dot(Ainv, b)
```

- Python's internal direct solver:

```
>>> x = np.linalg.solve(A, b)
```

- These are black boxes! We are going over some methods later!

Exercise: performance of inverse computation

Create a script that generates matrices with random elements of various sizes $N \times N$ (e.g. values of $N \in \{10, 20, 50, 100, 200, \ldots, 5000, 10000\}$). Compute the inverse of each matrix, and use `import time` and `time.time()` to see the computing time for each inversion. Plot the time as a function of the matrix size N.

Exercise: performance of inverse computation

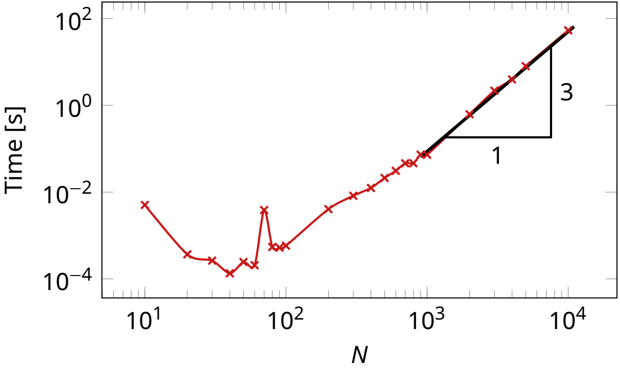
Create a script that generates matrices with random elements of various sizes $N \times N$ (e.g. values of $N \in \{10, 20, 50, 100, 200, \ldots, 5000, 10000\}$). Compute the inverse of each matrix, and use `import time` and `time.time()` to see the computing time for each inversion. Plot the time as a function of the matrix size N.

```
import numpy as np
import matplotlib.pyplot as plt
import time

# Generate random matrices of various sizes 's'.
# Invert the matrices and store the time required
# for the inversion. Plot the times vs 's'
s = np.array([10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000])
t_inv = []
for n in s:
    print(f'Working on size {n}')
    A = np.random.rand(n, n)
    start_time = time.time()
    Ainv = np.linalg.inv(A)
    t_inv.append(time.time() - start_time)

plt.loglog(s, t_inv)
plt.xlabel('N')
plt.ylabel('Time [s]')
plt.show()
```

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Exercise: sample results

Each computer produces slightly different results because of background tasks, different matrices, etc. This is especially noticable for small systems.



The time increases by 3 orders of magnitude, for every magnitude in N. The *computational complexity* of matrix inversion scales with $\mathcal{O}(N^3)$!

Today's outline

- Introduction
- Matrix inversion
- Solving a linear system
- Towards larger systems
- Summary

Towards larger systems

Computation of determinants and inverses of large matrices in this way is too difficult (slow), so we need other methods to solve large linear systems!

Towards larger systems

- Determinant of upper triangular matrix:

$$\det\left|M_{\text{tri}}\right| = \prod_{i=1}^{n} a_{ii} \qquad M = \begin{bmatrix} 5 & 3 & 2 \\ 0 & 9 & 1 \\ 0 & 0 & 1 \end{bmatrix} \Rightarrow \det\left|M\right| = 5 \times 9 \times 1 = 45$$

- Matrix multiplication:

$$\det\left|AM\right| = \det\left|A\right| \times \det\left|M\right|$$

- When A is an identity matrix ($\det\left|A\right| = 1$):

$$\det\left|AM\right| = \det\left|A\right| \times \det\left|M\right| = 1 \times \det\left|M\right|$$

- With rules like this, we can use row-operations so that we can compute the determinant more cheaply.

Solutions of linear systems

Rank of a matrix: the number of linearly independent columns (columns that can not be expressed as a linear combination of the other columns) of a matrix.

$$M = \begin{bmatrix} 5 & 3 & 2 \\ 0 & 9 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

- 3 independent columns
- In Python:

```
>>> numpy.linalg.matrix_rank(M)
```

$$M = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

- col 2 = 2× col 1
- col 4 = col 3 − col 1
- 2 independent columns: rank = 2

Solutions of linear systems

The solution of a system of linear equations may or may not exist, and it may or may not be unique. Existence of solutions can be determined by comparing the rank of the Matrix M with the rank of the augmented matrix M_a:

```
>>> numpy.linalg.matrix_rank(A)
>>> numpy.linalg.matrix_rank(np.column_stack((A,b))) # Concatenated matrices
```

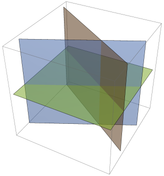
Our system: $Mx = b$

$$M = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \Rightarrow M_a = \begin{bmatrix} M_{11} & M_{12} & M_{13} & b_1 \\ M_{21} & M_{22} & M_{23} & b_2 \\ M_{31} & M_{32} & M_{33} & b_3 \end{bmatrix}$$

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Existence of solutions for linear systems

For a matrix M of size $n \times n$, and augmented matrix M_a:

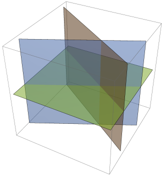


- $\text{Rank}(M) = n$:
  Unique solution

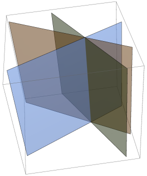
Existence of solutions for linear systems

For a matrix M of size $n \times n$, and augmented matrix M_a:

- $\text{Rank}(M) = n$:
  Unique solution



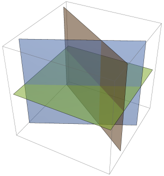
- $\text{Rank}(M) = \text{Rank}(M_a) < n$:
  Infinite number of solutions



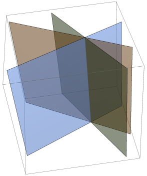
Existence of solutions for linear systems

For a matrix M of size $n \times n$, and augmented matrix M_a:

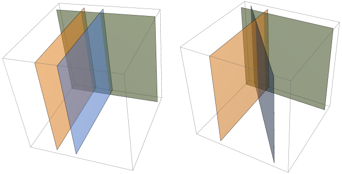
- $\text{Rank}(M) = n$:
  Unique solution



- $\text{Rank}(M) = \text{Rank}(M_a) < n$:
  Infinite number of solutions



- $\text{Rank}(M) < n$, $\text{Rank}(M) < \text{Rank}(M_a)$:
  No solutions



Two examples

$$M = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 3 & 1 \\ 0 & 0 & 2 \end{bmatrix} \quad b = \begin{bmatrix} 17 \\ 11 \\ 4 \end{bmatrix} \Rightarrow M_a = \begin{bmatrix} 1 & 1 & 2 & 17 \\ 0 & 3 & 1 & 11 \\ 0 & 0 & 2 & 4 \end{bmatrix}$$

$\text{rank}(M) = 3 = n \Rightarrow$ Unique solution

Two examples

$$M = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 3 & 1 \\ 0 & 0 & 2 \end{bmatrix} \quad b = \begin{bmatrix} 17 \\ 11 \\ 4 \end{bmatrix} \Rightarrow M_a = \begin{bmatrix} 1 & 1 & 2 & 17 \\ 0 & 3 & 1 & 11 \\ 0 & 0 & 2 & 4 \end{bmatrix}$$

$\text{rank}(M) = 3 = n \Rightarrow$ Unique solution

$$M = \begin{bmatrix} 1 & 1 & 2 \\ 0 & 3 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad b = \begin{bmatrix} 17 \\ 11 \\ 0 \end{bmatrix} \Rightarrow M_a = \begin{bmatrix} 1 & 1 & 2 & 17 \\ 0 & 3 & 1 & 11 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$\text{rank}(M) = \text{rank}(M_a) = 2 < n \Rightarrow$ Infinite number of solutions

Today's outline

- Introduction
- Matrix inversion
- Solving a linear system
- Towards larger systems
- Summary

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Summary

- Linear equations can be written as matrices
- Using the inverse, the solution can be determined
  - Inverse via cofactors
  - Inverse and solution in Python
- Introduced the concept of computational complexity: matrix inversion scales with N^3
- A solution depends on the rank of a matrix