

Linear Systems: Matrices, eigen-values, ...



William Oquendo, woquendo@gmail.com

Outline

Linear Systems and Matrices

Solving linear systems $Ax = b$

Computing the determinant

Computing the inverse

Matrix factorizations

Eigen value and eigen vectors

Problems

Bibliography

Linear Systems and Matrices

Solving linear systems $Ax = b$

Computing the determinant

Computing the inverse

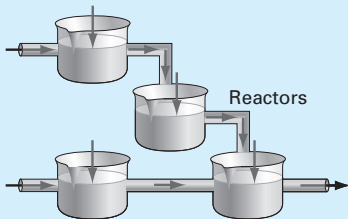
Matrix factorizations

Eigen value and eigen vectors

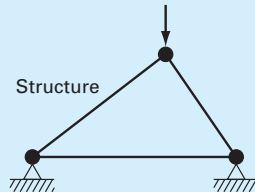
Problems

Bibliography

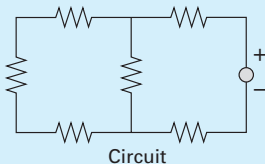
Example : Some applications [1]



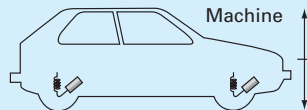
(a) Chemical engineering



(b) Civil engineering



(c) Electrical engineering



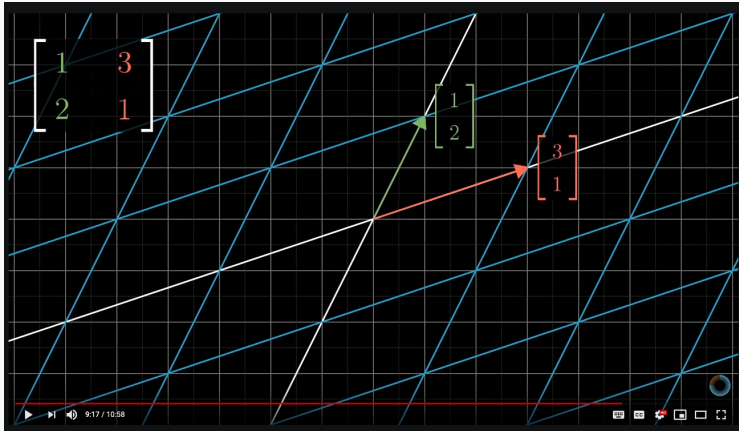
(d) Mechanical engineering

FIGURE 8.7

Engineering systems which, at steady state, can be modeled with linear algebraic equations.

Example : Some applications [1]

3Brown1Blue channel



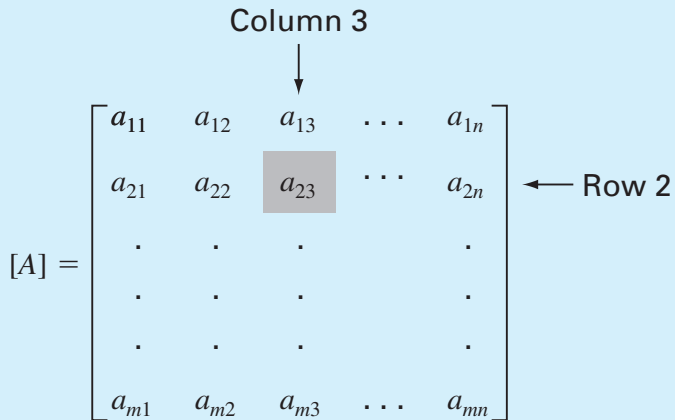
A matrix

Column 3

↓

$$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{bmatrix}$$

← Row 2

The diagram shows a matrix [A] with m rows and n columns. The elements are labeled a_{ij} where i is the row index and j is the column index. The first row contains a_{11}, a_{12}, a_{13}, ..., a_{1n}. The second row contains a_{21}, a_{22}, a_{23}, ..., a_{2n}. The third row contains a dot, a dot, a dot, and then a dot. The fourth row contains a dot, a dot, a dot, and then a dot. The fifth row contains a dot, a dot, a dot, and then a dot. The last row contains a_{m1}, a_{m2}, a_{m3}, ..., a_{mn}. An arrow points down to the third column, labeled 'Column 3'. Another arrow points left to the second row, labeled 'Row 2'. The element a_{23} is highlighted in a grey box.

- A symmetric matrix :

$$[A] = \begin{bmatrix} 5 & 1 & 2 \\ 1 & 3 & 7 \\ 2 & 7 & 8 \end{bmatrix}$$

Special cases

- A symmetric matrix :

$$[A] = \begin{bmatrix} 5 & 1 & 2 \\ 1 & 3 & 7 \\ 2 & 7 & 8 \end{bmatrix}$$

- A diagonal matrix :

$$[A] = \begin{bmatrix} a_{11} & & \\ & a_{22} & \\ & & a_{33} \end{bmatrix}$$

Special cases

- A symmetric matrix :

$$[A] = \begin{bmatrix} 5 & 1 & 2 \\ 1 & 3 & 7 \\ 2 & 7 & 8 \end{bmatrix}$$

- A diagonal matrix :

$$[A] = \begin{bmatrix} a_{11} & & \\ & a_{22} & \\ & & a_{33} \end{bmatrix}$$

- Identity matrix :

$$[I] = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}$$

Special cases

- A symmetric matrix :

$$[A] = \begin{bmatrix} 5 & 1 & 2 \\ 1 & 3 & 7 \\ 2 & 7 & 8 \end{bmatrix}$$

- A diagonal matrix :

$$[A] = \begin{bmatrix} a_{11} & & \\ & a_{22} & \\ & & a_{33} \end{bmatrix}$$

- Identity matrix :

$$[I] = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}$$

- Upper triangular :

$$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ & a_{22} & a_{23} \\ & & a_{33} \end{bmatrix}$$

Special cases

- A symmetric matrix :

$$[A] = \begin{bmatrix} 5 & 1 & 2 \\ 1 & 3 & 7 \\ 2 & 7 & 8 \end{bmatrix}$$

- A diagonal matrix :

$$[A] = \begin{bmatrix} a_{11} & & \\ & a_{22} & \\ & & a_{33} \end{bmatrix}$$

- Identity matrix :

$$[I] = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}$$

- Upper triangular :

$$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ & a_{22} & a_{23} \\ & & a_{33} \end{bmatrix}$$

- Lower triangular :

$$[A] = \begin{bmatrix} a_{11} & & \\ a_{21} & a_{22} & \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Special cases

- A symmetric matrix :

$$[A] = \begin{bmatrix} 5 & 1 & 2 \\ 1 & 3 & 7 \\ 2 & 7 & 8 \end{bmatrix}$$

- A diagonal matrix :

$$[A] = \begin{bmatrix} a_{11} & & \\ & a_{22} & \\ & & a_{33} \end{bmatrix}$$

- Identity matrix :

$$[I] = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}$$

- Upper triangular :

$$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ & a_{22} & a_{23} \\ & & a_{33} \end{bmatrix}$$

- Lower triangular :

$$[A] = \begin{bmatrix} a_{11} & & \\ a_{21} & a_{22} & \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

- Banded :

$$[A] = \begin{bmatrix} a_{11} & a_{12} & & \\ a_{21} & a_{22} & a_{23} & \\ & a_{32} & a_{33} & a_{34} \\ & & a_{43} & a_{44} \end{bmatrix}$$

Matrix definition in python

You can use either numpy/scipy or sympy.

See [numpy docs](#) .

```
import numpy as np
a = np.matrix('1 2; 3 4')
print(a)
a = np.matrix([[1, 2], [3, 4]])
print(a)
a = np.array([1, 2, 3,
↪ 4]).reshape(2,2)
print(a)
```

```
[[1 2]
 [3 4]]
[[1 2]
 [3 4]]
[[1 2]
```

See [sympy docs](#)

```
from sympy.matrices import Matrix,
↪ eye, zeros, ones, diag
M = Matrix([[1,0,4], [0,0,0]])
print(M)
print(eye(4))
print(M[0, 2])
```

```
Matrix([[1, 0, 4], [0, 0, 0]])
Matrix([[1, 0, 0, 0], [0, 1, 0, 0],
4
```

```
Matrix([[1, 0, 4], [0, 0, 0]])
Matrix([[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]])
4
```

Matrix operations in python

Example in python

```
import numpy as np
a = np.matrix('1 2; 3 4')
b = np.matrix([[5, -1], [-3, 24]])
c = a+b # sum
print(c)
c = a*b # Multiplication
print(c)
c = a/b # multiply by the inverse of b
print(c)
print(c.max())
print(c.min())
d = np.array([[5, -1], [-3, 24]])
print("d:\n",d)
```

Output

```
[[ 6  1]
 [ 0 28]]
[[-1 47]
 [ 3 93]]
[[ 0.2          -2.          ]
 [-1.          0.16666667]]
0.2
-2.0
d:
[[ 5 -1]
 [-3 24]]
```

You can watch : [https:](https://www.youtube.com/watch?v=XkY2DOUCWMU)

[//www.youtube.com/watch?v=XkY2DOUCWMU](https://www.youtube.com/watch?v=XkY2DOUCWMU)

Linear Systems and Matrices

Solving linear systems $Ax = b$

Computing the determinant

Computing the inverse

Matrix factorizations

Eigen value and eigen vectors

Problems

Bibliography

Example

Solve the following linear system (See `numpy.linalg`)

$$\begin{bmatrix} 150 & -100 & 0 \\ -100 & 150 & -50 \\ 0 & -50 & 50 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 588.6 \\ 686.7 \\ 784.8 \end{Bmatrix}$$

Example

Solve the following linear system (See [numpy linalg](#))

$$\begin{bmatrix} 150 & -100 & 0 \\ -100 & 150 & -50 \\ 0 & -50 & 50 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 588.6 \\ 686.7 \\ 784.8 \end{Bmatrix}$$

```
import numpy as np

A = np.array([[150, -100, 0], [-100, 150, -50], [0, -50, 50]])
b = np.array([588.6, 686.7, 784.8])
x = np.linalg.solve(A, b) # magic
print(x)
# confirm
print(A.dot(x) - b)
```

Example

Solve the following linear system (See `numpy.linalg`)

$$\begin{bmatrix} 150 & -100 & 0 \\ -100 & 150 & -50 \\ 0 & -50 & 50 \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 588.6 \\ 686.7 \\ 784.8 \end{Bmatrix}$$

```
import numpy as np

A = np.array([[150, -100, 0], [-100, 150, -50], [0, -50, 50]])
b = np.array([588.6, 686.7, 784.8])
x = np.linalg.solve(A, b) # magic
print(x)
# confirm
print(A.dot(x) - b)
```

```
[41.202 55.917 71.613]
```

```
[-5.68434189e-13 -2.27373675e-13  2.27373675e-13]
```

Exercise 1

8.3 Write the following set of equations in matrix form:

$$50 = 5x_3 - 6x_2$$

$$2x_2 + 7x_3 + 30 = 0$$

$$x_1 - 7x_3 = 50 - 3x_2 + 5x_1$$

Solve the system.

Exercise 2

Solve the following system

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 10 & -10 & 0 & -15 & -5 \\ 5 & -10 & 0 & -20 & 0 & 0 \end{bmatrix} \begin{Bmatrix} i_{12} \\ i_{52} \\ i_{32} \\ i_{65} \\ i_{54} \\ i_{43} \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 200 \end{Bmatrix}$$

Can you measure the time spent?

Exercise 3

Solve this system:

$$\frac{-2.3x_1}{5} + x_2 = 1.1$$

$$-0.5x_1 + x_2 = 1$$

Plot the system of equations and check whether this solution is or not special.

Exercise 4: Simulating temperature

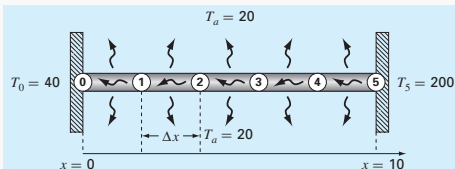


FIGURE 9.7

A noninsulated uniform rod positioned between two walls of constant but different temperature. The finite-difference representation employs four interior nodes.

$$\begin{bmatrix} 2.04 & -1 & 0 & 0 \\ -1 & 2.04 & -1 & 0 \\ 0 & -1 & 2.04 & -1 \\ 0 & 0 & -1 & 2.04 \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{Bmatrix} = \begin{Bmatrix} 40.8 \\ 0.8 \\ 0.8 \\ 200.8 \end{Bmatrix}$$

Linear Systems and Matrices

Solving linear systems $Ax = b$

Computing the determinant

Computing the inverse

Matrix factorizations

Eigen value and eigen vectors

Problems

Bibliography

See the [docs](#) for `determinant`

See the [docs](#) for determinant

```
from scipy import linalg
import numpy as np
A = np.array([[1,2,3], [4,5,6], [7,8,9]])
print(linalg.det(A))
A = np.array([[0,2,3], [4,5,6], [7,8,9]])
print(linalg.det(A))
```

Using scipy

See the [docs](#) for determinant

```
from scipy import linalg
import numpy as np
A = np.array([[1,2,3], [4,5,6], [7,8,9]])
print(linalg.det(A))
A = np.array([[0,2,3], [4,5,6], [7,8,9]])
print(linalg.det(A))
```

0.0

3.0000000000000001

You can watch : <https://www.youtube.com/watch?v=Ip3X9L0h2dk>

9.4 Given the system of equations

$$2x_2 + 5x_3 = 1$$

$$2x_1 + x_2 + x_3 = 1$$

$$3x_1 + x_2 = 2$$

(a) Compute the determinant.

Linear Systems and Matrices

Solving linear systems $Ax = b$

Computing the determinant

Computing the inverse

Matrix factorizations

Eigen value and eigen vectors

Problems

Bibliography

Using scipy

See [inverse with scipy](#)

```
from scipy import linalg
import numpy as np
A = np.array([[1., 2.], [3., 4.]])
B = linalg.inv(A)
print(B)
# verify
print(A.dot(B))
```

```
[[ -2.   1. ]
 [ 1.5 -0.5]]
[[1.0000000e+00  0.0000000e+00]
 [8.8817842e-16  1.0000000e+00]]
```

You can watch: <https://www.youtube.com/watch?v=uQhTuRlWMxw>

Condition number

The number $\kappa = ||A|| ||A^{-1}||$ is called the condition number of a matrix. Ideally it is 1. If κ is much larger than one, the matrix is ill-conditioned and the solution might have a lot of error.

Compute the condition number of the following matrix:

$$A = \begin{bmatrix} 1.001 & 0.001 \\ 0.000 & 0.999 \end{bmatrix} \quad (1)$$

Linear Systems and Matrices

Solving linear systems $Ax = b$

Computing the determinant

Computing the inverse

Matrix factorizations

Eigen value and eigen vectors

Problems

Bibliography

You can specifically solve with LU factorization. See [docs](#) .

```
from scipy.linalg import lu_factor, lu_solve
import numpy as np
A = np.array([[2, 5, 8, 7], [5, 2, 2, 8], [7, 5, 6, 6], [5, 4, 4, 8]])
b = np.array([1, 1, 1, 1])
lu, piv = lu_factor(A)
x = lu_solve((lu, piv), b)
print(x)
print(A.dot(x) - b)
```

```
[ 0.05154639 -0.08247423  0.08247423  0.09278351]
[-1.11022302e-16  0.00000000e+00 -1.11022302e-16  0.00000000e+00]
```

Cholesky

Or you can use the Cholesky factorization. See [Cholesky docs](#) . The matrix must be positive definite.

```
from scipy.linalg import cho_factor, cho_solve
import numpy as np
A = np.array([[9, 3, 1, 5], [3, 7, 5, 1], [1, 5, 9, 2], [5, 1, 2, 6]])
b = np.array([1, 1, 1, 1])
c, low = cho_factor(A)
x = cho_solve((c, low), b)
print(x)
print(A.dot(x) - b)
```

```
[-0.01749271  0.11953353  0.01166181  0.1574344 ]
[0.00000000e+00  2.22044605e-16  0.00000000e+00  0.00000000e+00]
```


Linear Systems and Matrices

Solving linear systems $Ax = b$

Computing the determinant

Computing the inverse

Matrix factorizations

Eigen value and eigen vectors

Problems

Bibliography

Definition

The eigen-values and eigen-vectors of a matrix satisfy the equation

$$Ax = \lambda x$$

The eigen-vectors form a basis where the matrix can be diagonalized. In general, computing the eigen vectors and eigenvalues is hard, and they can also be complex.

You can watch: <https://www.youtube.com/watch?v=PFDu9oVAE-g>

Implementation in Python

See [docs for scipy](#)

```
import numpy as np
from scipy import linalg
A = np.array([[0., -1.], [1., 0.]])
#A = np.array([[1, 0.], [0., 2.]])
#A = np.array([[2, 5, 8, 7], [5, 2, 2, 8], [7, 5, 6, 6], [5, 4, 4, 8]])
#A = np.array([[2, 5, 8, 7], [5, 2, 2, 8], [7, 5, 6, 6], [5, 4, 4, 8]])
sol = linalg.eig(A)
print("Eigen-values: ", sol[0])
print("Eigen-vectors:\n", sol[1])
# verify
print("Verification: ", A.dot(sol[1][:, 0]) - sol[0][0]*sol[1][:, 0])
```

Eigen-values: [0.+1.j 0.-1.j]

Eigen-vectors:

```
[[0.70710678+0.j          0.70710678-0.j          ]
 [0.          -0.70710678j 0.          +0.70710678j]]
```

Verification: [0.+0.j 0.+0.j]

Exercise 1 [2]

- ^a**11.** Consider $A = \begin{bmatrix} 5 & 4 & 1 & 1 \\ 4 & 5 & 1 & 1 \\ 1 & 1 & 4 & 2 \\ 1 & 1 & 2 & 4 \end{bmatrix}$. Find the eigenvalues and accompanying eigenvectors of this matrix, from Gregory and Karney [1969], without using software. *Hint:* The answers can be integers.

Linear Systems and Matrices

Solving linear systems $Ax = b$

Computing the determinant

Computing the inverse

Matrix factorizations

Eigen value and eigen vectors

Problems

Bibliography

Problem 1

Create a random matrix, with random elements between $[-1, 1]$, and make a histogram for the largest eigenvalue.

Problem 2 [2]

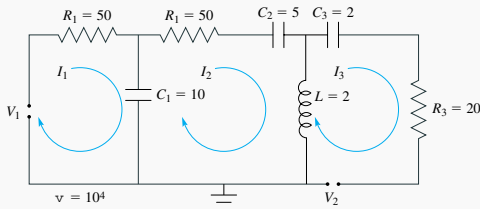
7. (Continuation) A common electrical engineering problem is to calculate currents in an electric circuit. For example, the circuit shown in the figure with R_i (ohms), C_i (microfarads), L (millihenries), and ω (hertz) leads to the system

$$\begin{cases} (50 - 10i)I_1 + (50)I_2 + (50)I_3 = V_1 \\ (10i)I_1 + (10 - 10i)I_2 + (10 - 20i)I_3 = 0 \\ - (30i)I_2 + (20 - 50i)I_3 = -V_2 \end{cases}$$

Select V_1 to be 100 millivolts, and solve two cases:

- ^aa. The two voltages are in phase; that is, $V_2 = V_1$.
^ab. The second voltage is a quarter of a cycle ahead of the first; that is, $V_2 = iV_1$.

Use the complex arithmetic version of *Naive_Gauss*, and in each case, solve the system for the amplitude (in milliamperes) and the phase (in degrees) for each current I_k . *Hint:* When $I_k = \text{Re}(I_k) + i \text{Im}(I_k)$, the amplitude is $|I_k|$, and the phase is $(180^\circ/\pi) \arctan[\text{Im}(I_k)/\text{Re}(I_k)]$. Draw a diagram to show why this is so.



Problem 3 [2]

- ^a4. The **Hilbert matrix** of order n is defined by $a_{ij} = (i + j - 1)^{-1}$ for $1 \leq i, j \leq n$. It is often used for test purposes because of its ill-conditioned nature. Define $b_i = \sum_{j=1}^n a_{ij}$. Then the solution of the system of equations $\sum_{j=1}^n a_{ij}x_j = b_i$ for $1 \leq i \leq n$ is $\mathbf{x} = [1, 1, \dots, 1]^T$. Verify this. Select some values of n in the range $2 \leq n \leq 15$, solve the system of equations for \mathbf{x} using procedures *Gauss* and *Solve*, and see whether the result is as predicted. Do the case $n = 2$ by hand to see what difficulties occur in the computer.

Thank you

Linear Systems and Matrices

Solving linear systems $Ax = b$

Computing the determinant

Computing the inverse

Matrix factorizations

Eigen value and eigen vectors

Problems

Bibliography



Steven C Chapra.

Applied Numerical Methods with MATLAB for Engineers and Scientists.

McGraw-Hill, 2012.



E Ward Cheney and David R Kincaid.

Numerical mathematics and computing.

Cengage Learning, 2012.



Yahya Esmail Osais.

Computer Simulation: A Foundational Approach Using Python.

Chapman and Hall/CRC, 2017.



Ivan Savov.

No bullshit guide to math and physics.

Minireference Co., 2014.



Simon Sirca and Martin Horvat.

Computational Methods for Physicists.

Springer Berlin Heidelberg, 2012.



Cyrille Rossant.

Learning IPython for Interactive Computing and Data Visualization.

Packt Publishing Ltd, 2015.



Gaël Varoquaux, Emmanuelle Gouillart, Olav Vahtras, Valentin Haenel, Nicolas P Rougier, Ralf Gommers, Fabian Pedregosa, Zbigniew Jędrzejewski-Szmek, Pauli Virtanen, Christophe Combelles, and others.

Scipy Lecture Notes.

2015.



Jaan Kiusalaas.

Numerical Methods in Engineering with Python 3.

Cambridge university press, 2013.



Philip Nelson Jesse M. Kinder.

A Student's Guide to Python for Physical Modeling.

Princeton University Press, 2015.



Robert Johansson.

Numerical Python. A Practical Techniques Approach for Industry.

Apress, 2015.



John V. Guttag.

Introduction to Computation and Programming Using Python, Revised & Expanded.

The MIT Press, revised and expanded edition edition, 2013.

