Solving systems of linear equations

1 Introduction

We solve simultaneously the equations

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \quad b_2 \\ \qquad \vdots \qquad\qquad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nn}x_n = b_n \end{cases}$$

It is a system of linear equations.

$x_1, \_, x_n$ are the unknown quantities, the $(a_{ij})_{\substack{i=1..n \\ j=1..n}}$ are the coefficients, the $(b_i)_{i=1..n}$ are the constants.

Notations

with matrices

$$Ax = b \quad \text{or} \quad \underbrace{\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}}_{\substack{A \\ \text{coefficient matrix}}} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}}_{b}$$

for computational purpose:

$$[A|b] \quad \text{or} \quad \left[\begin{array}{ccc|c} a_{11} & \dots & a_{1n} & b_1 \\ \vdots & & & \vdots \\ a_{n1} & \dots & a_{nn} & b_n \end{array}\right]$$

Questions

- does a solution x exist? Is it unique?
- can we compute it automatically?
- Are there special ways to do it when A has many zeros?

existence and uniqueness of solution

If the rows (or columns) of A are linearly independent or equivalently if $\det A = |A| \neq 0$, the system has a unique solution.

Otherwise, we say that A is singular and the system can have no solution or a continuum of solutions depending on the constant vector b.

e.g. $\begin{cases} 2x+y=3 \\ 4x+2y=6 \end{cases}$ has infinitely many solutions, namely the line of equation $2x+y=3$, while

$\begin{cases} 2x+y=3 \\ 4x+2y=0 \end{cases}$ has no solution.

Conditioning

Problem may occur if A is "almost singular" i.e. when $|A|$ is small compared to it's coefficients, i.e. $|A| \ll \|A\|$ where $\|A\|$ is any norm of A (we say that the system or the matrix A are ill-conditioned.

e.g.

Euclidian norm: $\|A\|_e = \left(\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}^2 \right)^{1/2} = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}^2}$

Infinity norm $\|A\|_\infty = \max_{i=1..n} \sum_{j=1}^{n} |a_{ij}|$

matrix condition number $\quad cond(A) = \|A\| \, \|A^{-1}\|$

If $cond(A) \sim \|I\|$, the matrix is well conditioned. This number increases with the degree of ill conditioning of A.

But this number is difficult to compute.

When A is ill conditioned, small changes in the coefficients may result in large changes in the solution.

e.g. We solve $2x+y=3$, $2x+1.001y=0$.

$\Rightarrow 0.001y=-3 \Leftrightarrow y=-3000 \Rightarrow x=3003/2=1501.5$.

If we change the second equation as $2x+1.002y=0$, we find $y=-1500$ $x=751.5$, so that 0.1% change on the coefficient produces 100% change in the solution.

Note that $|A| = 2\times1.001-2=0.002$ is small compare to the coefficients $2, 1$, and the system is ill conditioned.

One cannot therefore trust computed solution of ill conditioned systems.

In general, the coefficient matrix A is defined permanently while b represents the input of a system and we need to be able to solve $Ax=b$ for any kind of b.

Two kinds of methods for solving systems: "direct" or "iterative".

In direct methods, we carry out some changes on the equations in order to simplify the system. This done through elementary operations which do not affect the solution but may change the determinant of the system:

- exchanging two equations (changes the sign of the determinant).

- multiplying an equation by a non zero constant (multiplies the determinant by this constant)
- multiplying an equation by a constant and then subtracting it from another equation (does not change the determinant).

In iterative methods or indirect methods, we view the solution of the system as the limit of a very large (infinite) number of steps. We stop the process according to the accuracy we want for the solution. These methods may be interesting for very large and sparse matrices.

② Gauss elimination method

We transform the system into a diagonal system of the form

$$Ux = b'$$

where U is an upper triangular matrix i.e. a matrix which has zeros below the main diagonal:

e.g.

$$U = \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$ in 3 dimensions.

We explain the method on an example:

$$\begin{cases} 4x_1 - 2x_2 + x_3 = 11 & (e1) \\ -2x_1 + 4x_2 - 2x_3 = -16 & (e2) \\ x_1 - 2x_2 + 4x_3 = 17 & (e3) \end{cases}$$

We proceed by using one of the elementary operation listed above.

We use equation $(e1)$ as a pivot. We change the lines below the pivot line by substracting to them the pivot line multiplied by a well chosen constant.

$$\begin{cases} 4x_1 - 2x_2 + x_3 = 11 & (e1) \text{ unchanged.} \\ 3x_2 - \frac{3}{2}x_3 = -\frac{21}{2} & (e2) - \left(-\frac{1}{2}\right)(e1) = (e'_2) \\ -\frac{3}{2}x_2 + \frac{15}{4}x_3 = \frac{57}{4} & (e3) - \left(\frac{1}{4}\right)(e1) = (e'_3) \end{cases}$$

Then we proceed with the second line as a pivot.

$$\begin{cases} 4x_1 - 2x_2 + x_3 = 11 & (e1) \\ 3x_2 - \frac{3}{2}x_3 = -\frac{21}{2} & (e'_2) \\ 3x_3 = 9 & (e'_3) - \left(-\frac{1}{2}\right)(e'_2) \end{cases}$$

Once we have such system, it is very easy to solve "backward"

$x_3 = 3$, $x_2 = -2$, $x_1 = 1$ ("back substitution")

And we get for free $|A| = 4 \times 3 \times 3 = 36$.

Algorithm.

We are modifying the coefficients of the system as we are proceeding.

Elimination

for $k = 1$ to $n-1$ (pivot row)

  for $i = k+1$ to n

    $\lambda = a_{ik}/a_{kk}$ provided that $a_{kk} \neq 0$

    for $j = k+1$ to n

      $a_{ij} \leftarrow a_{ij} - \lambda a_{kj}$

    $b_i \leftarrow b_i - \lambda b_k$

Rq: we do not compute the new a_{ik} which is zero and is not used in the substitution step.

Substitution

$x_n = b_n / a_{nn}$

for $k = n-1$ to 1 step -1

$$x_k = \Big(b_k - \sum_{j=k+1}^{n} a_{kj}\, x_j\Big)/a_{kk}$$

Operation count : we count multiplications and division.

for $k = 1$ to $n-1$, for $i = k+1$ to n, we have

$$1 + 1 + n - (k+1) + 1 = n - k + 2 \text{ such operations}$$

i.e. $\displaystyle\sum_{k=1}^{n-1} (n-k)(n-k+2)$ operations

We change variable : $\displaystyle\sum_{j=1}^{n-1} j(j+2) \sim \sum_{j=1}^{n-1} j^2 = \frac{(n-1)n(2n-1)}{6}$

(for the elimination phase) $\sim n^3/3$.

Programming First we change indices "from 1 to n" to "from 0 to n−1".

- for k = 0 to n−2 ⟷ for k in range(0, n−1):
  for i = k+1 to n−1 ⟷ for i in range(k+1, n):
  replace the loop on j by

```
a[i, k+1:n] = a[i, k+1:n] - λ * a[k, k+1:n]
```

- we can re-use b for substitution. Moreover python can deal with empty arrays. Finally we substitute the sum by a dot product.

```
for k in range(n-1, -1, -1):
    b[k] = (b[k] - np.dot(a[k, k+1:n], b[k+1:n]))/a[k,k]
```

We can define a function gaussElimin(a, b) that we put in a module that we call M1_linalg.py

③ LU decomposition method

Theorem Any square matrix can be decomposed as product of a lower triangular matrix and an upper triangular matrix. $A = LU$ (*) (not unique)

If we know L and U, solving $Ax = b$ consists in solving $Ly = b$ with a forward substitution procedure and then, knowing y, by solving $Ux = y$ with another substitution procedure.

Finding L and U in (*) is known as LU decomposition or factorization.

Since the decomposition is not unique, there are several ways of decomposing depending on the constraints we give for that decomposition.

Doolittle's decomposition: $l_{ii} = 1, i = 1..n$.
Crout's decomposition: $U_{ii} = 1, i = 1..n$
Choleski's decomposition $L = U^T$

Doolittle's decomposition method.

In order to understand how to operate, we start from the decomposition in the case when the dimension is $n = 3$: we assume

$$L = \begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix} \quad U = \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{pmatrix}$$

and therefore

$$A = \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ L_{21}U_{11} & L_{21}U_{12} + U_{22} & L_{21}U_{13} + U_{23} \\ L_{31}U_{11} & L_{31}U_{12} + L_{32}U_{22} & L_{31}U_{13} + L_{32}U_{23} + U_{33} \end{pmatrix};$$

We now proceed with Gauss elimination process.

$$\begin{array}{c|ccc} \text{Pivot} & U_{11} & U_{12} & U_{13} \\ L_{21} & 0 & U_{22} & U_{23} \\ L_{31} & 0 & L_{32}U_{22} & L_{32}U_{23} + U_{33} \end{array}$$

$$\begin{array}{c|ccc} & U_{11} & U_{12} & U_{13} \\ & 0 & U_{22} & U_{23} \\ L_{32} & 0 & 0 & U_{33} \end{array}$$

Therefore:

- In LU decomposition, U is obtained as the resulting matrix from Gauss elimination procedure
- The values of L are given as the pivots in the procedure.

We can use the matrix itself to store the pivot.
Once we have L and U we proceed by substitution.

Cf exercise 2 on sheet 1.

Choleski's decomposition method

This decomposition is not always possible.
Indeed, if $A = LL^T$ then A is symmetric since $A^T = (LL^T)^T = (L^T)^T L^T = LL^T$.
Moreover A is positive definite. Indeed, A is diagonalizable, we denote λ any eigen value of A and u any associated eigen vector. Then

$Au = \lambda$ can be written $LL^T u = \lambda u$. Therefore

$$\left.\begin{array}{l} \langle LL^T u, u\rangle = \langle L^T u, L^T u\rangle = \|L^T u\|^2 \geq 0 \\ \text{and } \langle LL^T u, u\rangle = \langle \lambda u, u\rangle = \lambda \|u\|^2 \end{array}\right\} \Rightarrow \lambda \geq 0$$

(since $u \neq 0$

Finally $\lambda > 0$ since A is invertible.

Theorem If A is symmetric positive definite then there exists $L \in \mathcal{M}_n(\mathbb{R})$ such that $LL^T = A$.

As before, we try to guess the algorithm. We assume $A = LL^T$ with

$$L = \begin{pmatrix} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \quad L^T = \begin{pmatrix} L_{11} & L_{21} & L_{31} \\ 0 & L_{22} & L_{32} \\ 0 & 0 & L_{33} \end{pmatrix}$$

$$\Rightarrow A = \begin{pmatrix} L_{11}^2 & L_{11}L_{21} & L_{11}L_{31} \\ L_{11}L_{21} & L_{21}^2 + L_{22}^2 & L_{21}L_{31} + L_{22}L_{32} \\ L_{11}L_{31} & L_{31}L_{21} + L_{32}L_{22} & L_{31}^2 + L_{32}^2 + L_{33}^2 \end{pmatrix}$$

$L_{11} = \sqrt{A_{11}}$ $\quad L_{21} = A_{12}/L_{11}$ $\quad L_{31} = A_{31}/L_{11}$

$L_{22} = (A_{22} - L_{21}^2)^{1/2}$ $\quad L_{32} = (A_{32} - L_{21}L_{31})/L_{22}$

Finally $L_{33} = (A_{33} - L_{31}^2 - L_{32}^2)^{1/2}$

This procedure can be generalized. We can write an algorithm which number of long operations is about $n^3/6$ (instead of $n^3/3$). This algorithm is detailed in the book by Kiusalaas.

Cf exercise 3 on sheet 1.

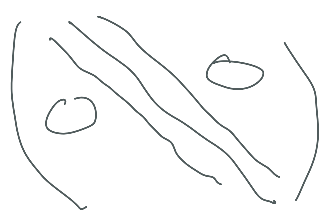
Remark: Cholesky's decomposition is also possible when A is not invertible. In this case, L is a $n \times m$ matrix where m is the rank of A.

Other decompositions:

- Crout's decomposition is the same as Doolittle's except that it is the diagonal of U and not of L which has ones on it.
- Gauss Jordan procedure: it is the same as Gauss elimination but you complete the procedure so that the resulting matrix is diagonal. It is not interesting because you need $n^3/2$ operations to compute it (you compute the inverse of A up to a final division on each row).

4 Symmetric and banded coefficient matrices.

Many engineering problems lead to matrices which have a lot of zeros or are "sparsely populated" or "sparse".
Some of them have their nonzero terms clustered around the diagonal, they are called banded matrix, or p-diagonal matrix where p is the maximum of non zero terms in one row or column symmetically placed around the diagonal. For instance, a tri-diagonal matrix has the form



where the only non zero terms are on the lines.

The banded structure of a coefficient matrix can be exploited to save storage and computation time, as we now explain it on Doolittle's decomposition for a tridiagonal matrix.

LU decomposition of a tri-diagonal matrix.

storage: instead of n^2 coefficients, we have $n+2(n-1)$ $= 3n-2$ non zero coefficients. Assume

$$A = \begin{pmatrix} d_1 & e_1 & 0 & \cdots & 0 \\ c_2 & d_2 & e_2 & \ddots & \vdots \\ & \ddots & \ddots & \ddots & e_{n-1} \\ 0 & \cdots & 0 & c_n & d_n \end{pmatrix}$$

for $k = 2$ to n do
$\lambda = c_k / d_{k-1}$
$d_k \leftarrow d_k - \lambda e_{k-1}$
$c_k = \lambda$

And therefore the number of op. is $2n$.

Rq The matrix L and U remain tri-diagonal.

Other methods. When A is symmetric, A can be decomposed as $A = LU = LDL^T$ where D is a diagonal matrix. We have $U = DL^T$ and D, L^T can be easily recovered from U. We can use this together with the banded coefficients condition in order to construct a very efficient algorithm (see Kiusalaas).

5 Pivoting

There can be cases in which the procedure we saw do not work, namely when pivot is 0 and therefore we cannot compute 1.

Example

Imagine we have the following system

$$\begin{cases} \quad - x_2 + x_3 = 0 \\ -x_1 + 2x_2 - x_3 = 0 \\ 2x_1 - x_2 \qquad = 1 \end{cases}$$

We cannot even start the elimination procedure because the pivot coefficient is 0. A way to get rid off this problem is to invert lines: since the matrix is invertible there is at least 1 non zero coefficient in the first column.

Row reordering or row pivoting is also required when the pivot element is very small by comparison with the other terms.

Indeed if we solve the system

$$[A|b] = \left(\begin{array}{ccc|c} \varepsilon & -1 & 1 & 0 \\ -1 & 2 & -1 & 0 \\ 2 & -1 & 0 & 1 \end{array}\right)$$

by our procedure gaussElimin, then, with $\varepsilon = 1.E-15$ we get the solution

$x_1 \approx 1.1102 \quad x_2 = 1 \quad x_3 = 1$

(the solution with $\varepsilon = 0$ is $(1, 1, 1)$)

Diagonal dominance A $n \times n$ matrix is said to be diagonally dominant if on all raw i we have:

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|$$

example The matrix

$$A = \begin{pmatrix} -2 & 4 & 1 \\ 1 & -1 & 3 \\ 4 & -2 & 1 \end{pmatrix}$$

is not diagonally dominant (there is a pb at each row, but a problem at one row would be sufficient to loose the property).

However if we reorder the equations, we can end with a system whose matrix is

$$A' = \begin{pmatrix} 4 & -2 & 1 \\ -2 & 4 & 1 \\ 1 & -1 & 3 \end{pmatrix}$$

which is diagonally dominant

theorem: If the matrix A is diagonally dominant then pivoting is not necessary.

Therefore we need a procedure which transform A to a matrix A' as close as possible to a diagonally dominant matrix.

A simple procedure to avoid problems could be the following adapted Gauss elimination procedure:

$c = (1, \ldots, n) = (c_1, \ldots, c_n)$

for each $k = 1$ to n

$p = \operatorname{argmax}(|a_{kk}|, |a_{kk+1}|, \ldots, |a_{kn}|)$

if $p \neq k$

exchange column k and p

keep trace of the change by setting

$c_k, c_p \longleftarrow c_p, c_k$ simultaneously

At the end, we have a system whose variables $x_1, \ldots, x_n$ have been exchanged. In order to get the solution, whe have to re-order the solution vector:

$c_1, c_2, \ldots c_n$ contain the indices of the components of the solution which are stored in $b_1, b_2, \ldots b_n$ respectively, i.e. $b_k = x_{c_k}$ for all k.

When not necessary, it is better not to pivot because pivoting has drawbacks since it increases the number of computations and may even destroy the structure of the system (banded or symmetric). Most of the time, engineering problems end up well posed in that respect.