Numerical analysis (5/7): linear systems

University of Luxembourg

Philippe Marchner

Siemens Digital Industries Software, France

November 16th, 2023

Outline

Outline

Overview of the content

We already encountered linear system in different situations

- interpolation (global and splines),
- least-square data fit (normal equations),
- finite difference methods (tridiagonal system),

In general, almost all PDE problems (even non-linear ones) require to solve at some point a linear system of the form

$$Ax = b, \quad \text{of size } N \times N$$

We will study two kind of resolution methods

1. Direct methods,
2. Iterative methods,

The "best" method for solving A quickly and accurately is problem dependent.

Introduction

There are two families of methods

1. Direct methods : where we obtain the exact solution (up to round-off errors) in a finite number of operations
   - LU factorization, $A = LU$
   - QR factorization (see exercise), $A = QR$
   - Singular value decomposition (SVD), $A = U\Sigma V^*$
2. Iterative methods : they consist in building a sequence of vectors x^k converging towards the solution x. It is stopped after a finite number of iterations k chosen in such a way that x^k is close enough to x
   - Fixed point iteration: Jacobi, Gauss-Seidel, SOR
   - Krylov subspace: conjugate gradient, GMRES, ...

Introduction

Remarks

- we never use the Cramer's formulas

$$A^{-1} = \frac{\text{com}A^T}{\det A}$$

since they need to compute determinants which is a highly computationally expensive task (too much elementary operations)

- we never compute A^{-1} to solve $Ax = b$.
Numerically, this is in the other way : the computation of A^{-1}, and also $\det A$, are a by-product of the solution to the linear systems.

To compute A^{-1} we may solve

$$\{Ax_i = e_i,\ i = 1, ..., n\}$$

where e_i is i-th vector of the $\mathbb{R}^n$ canonical basis. The i-th column of A^{-1} is x_i.

Outline

Outline

Linear system that are easy to solve

A general idea is to try to transform the linear systems into an easier one

Diagonal matrix

all the elements are zero expect the ones on the main diagonal

$$A = \begin{bmatrix} a_1 & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_n \end{bmatrix}$$

If A is invertible then all the a_i are non zero and the solution of $Ax = b$ is trivial.
If $x = (x_1, ..., x_n)$ and $b = (b_1, ..., b_n)$, we have

$$x_i = \frac{b_i}{a_i},\ i = 1, ..., n$$

Upper (or lower) triangular matrices

All the elements below (or above) the main diagonal are zeros, i.e. $a_{ij} = 0$ for $i > j$ or $(i < j)$

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{nn} \end{bmatrix}$$

In this case, the linear system writes

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n = b_{n-1} \\ a_{nn}x_n = b_n \end{cases}$$

We always use a direct method for the solution, called backward (forward) elimination process

Backward substitution

we start by solving the last equation ; we substitute the result x_n in the previous equation, which provides x_{n-1}, and so on...

$$\begin{cases} x_n = b_n / a_{nn} \\ x_{n-1} = (b_{n-1} - a_{n-1,n} x_n) / a_{n-1,n-1} \\ \vdots \\ x_i = (b_i - a_{i,n} x_n - a_{i,n-1} x_{n-1} - \ldots - a_{i,i+1} x_{i+1}) / a_{ii} \\ \vdots \end{cases}$$

(the triangular matrix A is always assumed to be invertible so $a_{ii} \neq 0$ for all i)

Cost of the backward procedure

The computation of x_i requires: 1 division, $n - i$ multiplications, $n - i$ additions.
Therefore it needs a total of n divisions, $\frac{n(n-1)}{2}$ multiplications, $\frac{n(n-1)}{2}$ additions, which is about $\mathcal{O}(n^2)$ elementary operations.

Upper (or lower) triangular matrices

Remarks

- the same algorithm is used for lower triangular systems, which is called forward substitution,
- we will see that a cost of $\mathcal{O}(n^2)$ is cheap compared to the cost of more general direct methods,
- the idea is then to transform more complicated systems into such a triangular form

Outline

Gauss elimination

The Gauss method is a general method for solving a linear system

$$Ax = b$$

where A is an invertible matrix (not necessarily square). It needs 3 steps :

1. a procedure called "elimination process" which corresponds to find an invertible matrix M such that the matrix MA is upper triangular
2. Compute the vector Mb
3. Solve the easier linear system

   $$MAu = Mb$$

   where the matrix MA is upper triangular and can be solved by the backward elimination process.

Description of the first step

We will perform elimination thanks to row operations, namely we are allowed to

1. Multiplying a row by a value,
2. Adding one row to another row,
3. Exchanging two rows with each other.

Let $A = (a_{i,j}), \quad i \in [\![1, n]\!],\ j \in [\![1, n]\!]$

- at least one of the coefficients $a_{i,1},\ i \in [\![1, n]\!]$ of the first column A is not zero, otherwise the matrix would not be invertible
- we choose one of these nonzero coefficients, called the first pivot of the elimination.
- we exchange the row of the pivot with the first row which in a matrix notation is equivalent to left multiply A by a specific matrix.

Row exchange can be seen as a matrix operation

Description of the first step - Row exchange

- indeed, exchanging the rows i and j of A is equivalent to multiply A by

$$T(i,j) = \begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 0 & & & 1 & & \\ & & & 1 & & & & \\ & & & & \ddots & & & \\ & & & & & 1 & & \\ & & 1 & & & 0 & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{pmatrix} \begin{matrix} \\ \\ \leftarrow i \\ \\ \\ \\ \leftarrow j \\ \\ \\ \end{matrix}$$

$$\begin{matrix} & & \uparrow & & \uparrow & & \\ & & i & & j & & \end{matrix}$$

Remark: $\det(T(i,j)) = -1$

Description of the first step - scaling and adding rows

- we then set
$$P = \begin{cases} I & \text{if } a_{1,1} \text{ is the pivot and } \det(P) = 1 \\ T(1,i) & \text{if } a_{i,1} \text{ is the pivot and } \det(P) = -1 \end{cases}$$
- the obtained matrix $PA = (\alpha_{i,j})$ is such that $\alpha_{1,1} \neq 0$
- by linear combinations of the first with the other rows of PA, we force the other elements of the first column of PA under the main diagonal to be zero, the first row remaining unchanged
- written with matrices, this amounts to left multiply PA by the matrix
$$E = \begin{pmatrix} 1 & & & & \\ -\frac{\alpha_{2,1}}{\alpha_{1,1}} & 1 & & 0 & \\ -\frac{\alpha_{3,1}}{\alpha_{1,1}} & & 1 & & \\ \vdots & 0 & & \ddots & \\ -\frac{\alpha_{n,1}}{\alpha_{1,1}} & & & & 1 \end{pmatrix}$$

Description of the 1st step

- hence the matrix $B = EPA$ writes

$$B = \begin{pmatrix} \alpha_{1,1} & \alpha_{1,2} & \cdots & \alpha_{1,n} \\ 0 & b_{2,2} & \cdots & b_{2,n} \\ \vdots & \vdots & & \vdots \\ 0 & b_{n,2} & \cdots & b_{n,n} \end{pmatrix}, \quad b_{ij} = \alpha_{ij} - \frac{\alpha_{i1}}{\alpha_{11}}\alpha_{1j}, \quad (i,j) \in [\![2,n]\!]$$

Remarks

- simplicity of the calculations
- $det(E) = 1$ so $det(B) = \pm det(A)$, and B is still invertible
- as a consequence, at least one of the $b_{i,2}$, $i \in [\![2,n]\!]$ is not zero
- the second step of the elimination process consists in doing the same operations but only on the sub-matrix $(b_{i,j})_{2 \leq i,j \leq n}$

Description of the kth step

- by setting

$$A = A_1 = (a_{i,j}) = (a^1_{i,j}), \quad P = P_1, \quad P_1 A_1 = (\alpha^1_{i,j})$$

$$E = E_1, \quad B = A_2 = E_1 P_1 A_1 = (a^2_{i,j})$$

one gets finally, at the $(k-1)$th step of the elimination process, a matrix

$$A_k = E_{k-1} P_{k-1} \dots E_2 P_2 E_1 P_1 A_1$$

which has the form

Description of the kth step

$$A_k = \begin{pmatrix} a^k_{1,1} & a^k_{1,2} & \cdots & \cdots & \cdots & a^k_{1,n} \\ 0 & a^k_{2,2} & \cdots & \cdots & \cdots & a^k_{2,n} \\ \vdots & \ddots & \ddots & & \vdots & \\ \vdots & & 0 & a^k_{k,k} & \cdots & a^k_{k,n} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & a^k_{n,k} & \cdots & a^k_{n,n} \end{pmatrix} = \begin{pmatrix} \alpha^1_{1,1} & \alpha^1_{1,2} & \cdots & \cdots & \cdots & \alpha^1_{1,n} \\ 0 & \alpha^2_{2,2} & \cdots & \cdots & \cdots & \alpha^2_{2,n} \\ \vdots & \ddots & \ddots & & \vdots & \\ \vdots & & 0 & a^k_{k,k} & \cdots & a^k_{k,n} \\ \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & a^k_{n,k} & \cdots & a^k_{n,n} \end{pmatrix}$$

Description of the kth step

- since $det(A_k) = \pm det(A)$, the matrix A_k is invertible and then at least one of the $a_{i,k}^k$, $i \in [\![k, n]\!]$ is not zero.
- we choose one of these nonzero elements as a pivot.
- we exchange the pivot row with the kth row of A_k which is equivalent to multiply A_k by a matrix P_k which is the identity or a transposition matrix
- then the element $\alpha_{k,k}^k$ of the matrix $P_k A_k = (\alpha_{i,j}^k)$ is nonzero
- the elimination corresponds to the left multiplication of the matrix $P_k A_k$ by the matrix E_k

Elimination matrix at the kth step

$$E_k = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & 0 & 1 & & & \vdots \\ \vdots & & -\frac{\alpha^k_{k+1,k}}{\alpha^k_{k,k}} & \ddots & & \vdots \\ \vdots & & \vdots & & \ddots & 0 \\ 0 & 0 & -\frac{\alpha^k_{n,k}}{\alpha^k_{k,k}} & 0 & & 1 \end{pmatrix}$$

- after the $(n-1)$-th step, the matrix

$$A_n = E_{n-1}P_{n-1}A_{n-1} = E_{n-1}P_{n-1}\dots E_1P_1A$$

  is then upper triangular
- we have found an invertible matrix

$$M = E_{n-1}P_{n-1}A_{n-1} = E_{n-1}P_{n-1}$$

  such that MA is upper triangular

Theorem

Let A a an invertible or singular matrix. There exists at least an invertible matrix M such that MA is upper triangular

Illustration of the algorithm

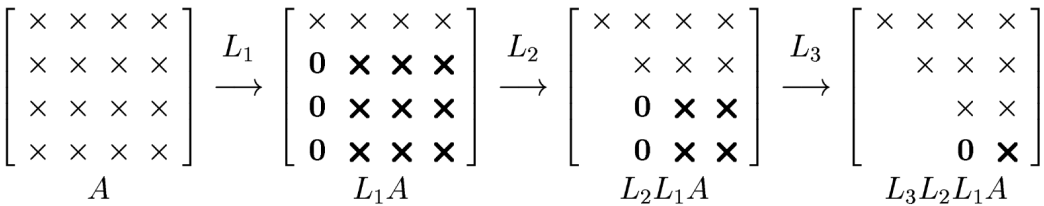

Figure: Illustration of the Gaussian elimination algorithm. From *Trefeten and Bau, Numerical Linear Algebra III*(1997)

Outline

Example

Resolution of a 3×3 linear system

$$\begin{pmatrix} 5 & 2 & 1 \\ 5 & -6 & 2 \\ -4 & 2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 12 \\ -1 \\ 3 \end{pmatrix}$$

here

$$A = A_1 = \begin{pmatrix} 5 & 2 & 1 \\ 5 & -6 & 2 \\ -4 & 2 & 1 \end{pmatrix} \quad \text{and} \quad b = b_1 = \begin{pmatrix} 12 \\ -1 \\ 3 \end{pmatrix}$$

- step 1 :

$$P_1 = I, \quad E_1 = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ \frac{4}{5} & 0 & 1 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 5 & 2 & 1 \\ 0 & -8 & 1 \\ 0 & \frac{18}{5} & \frac{9}{5} \end{pmatrix}, \quad b_2 = \begin{pmatrix} 12 \\ -13 \\ \frac{63}{5} \end{pmatrix}$$

Example

Resolution of a 3×3 linear system

- step 2 :

$$P_2 = I, \quad E_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{9}{20} & 1 \end{pmatrix}, \quad A_3 = \begin{pmatrix} 5 & 2 & 1 \\ 0 & -8 & 1 \\ 0 & 0 & \frac{9}{4} \end{pmatrix}, \quad b_3 = \begin{pmatrix} 12 \\ -13 \\ \frac{27}{4} \end{pmatrix}$$

- backward process :

$$x_3 = 3, \quad x_2 = 2, \quad x_1 = 1$$

Remark:

- we can compute M

$$M = E_2 E_1 = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ \frac{7}{20} & \frac{9}{20} & 1 \end{pmatrix}$$

- but the expression of M is not useful!

Choice of the pivot

- at the beginning of the kth step of the elimination process, theoretically one can choose any coefficient $a_{i,k}^k$ which is nonzero ($i \in [\![k, n]\!]$)
- in particular, if $a_{k,k}^k$ is nonzero we can choose it as the pivot (and then $P_k = I$)
- however, this way of proceeding can lead to round-off errors and an incorrect solution

Example

Let us consider the system

$$\begin{pmatrix} 10^{-4} & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

The exact solution is

$$x_1 = \frac{10000}{9999} \simeq 1, \quad x_2 = \frac{9998}{9999} \simeq 1$$

- let us assume that the computations are realized up to 3 digits
- first case:

$$\begin{array}{cc|c} 10^{-4} & 1 & 1 \\ 1 & 1 & 2 \end{array} \qquad \begin{array}{cc|c} 10^{-4} & 1 & 1 \\ 0 & -9.99\ 10^3 & -9.99\ 10^3 \end{array}$$

$$x_2 = 1 \quad \text{and} \quad x_1 = 0!$$

Example

Let us consider the system

$$\begin{pmatrix} 10^{-4} & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

- second case:

$$\begin{array}{cc|c|cc|c} 1 & 1 & 2 & 1 & 1 & 2 \\ 10^{-4} & 1 & 1 & 0 & 9.99\ 10^{-1} & 9.99\ 10^{-1} \end{array}$$

$$x_2 = 1 \quad \text{and} \quad x_1 = 1$$

- the rule is to rather use the pivot which corresponds to the coefficient with largest amplitude
- The partial pivot strategy : we determine the element $a_{i,k}^k$ (or one of the elements $a_{i,k}^k$) such that $k \leq i \leq n$ and

$$\left| a_{i,k}^k \right| = \max_{k \leq j \leq n} \left| a_{j,k}^k \right| .$$

Cost of Gauss method

- To get from A_k to A_{k+1} we need

$$n-k \text{ divisions, } (n-k)(n-k+1) \text{ additions and multiplications}$$

- leading a total of

$$\frac{n(n-1)}{2} \text{ divisions, } \frac{n^3-n}{3} \text{ multiplications and additions}$$

or

$$\frac{4n^3+3n^2-7n}{6} \text{elementary operations}$$

- we need to add the n^2 elementary operations for the forward substitution.
- When n is large, the terms scaling in n^2 and n small compared to n^3 and the global number of operations is around

$$\frac{2n^3}{3}$$

Remarks

- a cost of $\mathcal{O}(n^3)$ is huge for large linear systems !
- the numerical stability of Gaussian elimination, even with partial pivoting, is not obvious. A proper *backward analysis* is required: the key relies in the amplification of the entries $a_{k,k}^k$ during the elimination process, called the growth factor

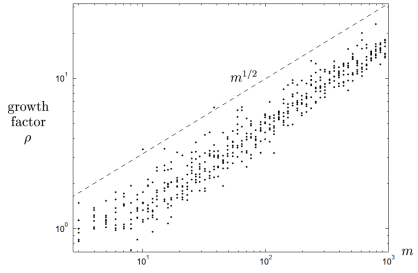

Figure 22.1. *Growth factors for Gaussian elimination with partial pivoting applied to* 496 *random matrices (independent, normally distributed entries) of various dimensions. The typical size of* ρ *is of order* $m^{1/2}$*, much less than the maximal possible value* 2^{m-1}.

Figure: From *Trefeten and Bau, Numerical Linear Algebra III*(1997)

Remarks

- What if we have many right hand side vectors, or we don't know b right away ?
- Note that determining transformation M such that $MA = U$ does not depend on b

Outline

LU factorization

- Let us suppose for a while that we do not use a pivot strategy, so if $a^k_{k,k}$ is non-zero we use it as the pivot
- if we can do so we have

$$P_1 = P_2 = \cdots = P_{n-1} = I \quad \text{and} \quad M = E_{n-1}E_{n-2}\dots E_1$$

- M being the product of lower triangular matrices it is lower triangular, and its inverse is also lower triangular

$$L = M^{-1}$$

LU factorization

- the matrix A can then be written

$$A = LU$$

where L is lower triangular and U is upper triangular such as

$$L = (E_{n-1}E_{n-2}\dots E_1)^{-1}, \quad U = (E_{n-1}E_{n-2}\dots E_1)A$$

Thanks to the first part, we have all the framework to express L and U explicitely !

LU factorisation

- we already know how to compute U since at the n-th step

$$U = A_n = \begin{pmatrix} a_{1,1}^n & \cdots & \cdots & a_{1,n}^n \\ 0 & a_{2,2}^n & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & a_{n,n}^n \end{pmatrix}$$

- the matrix L is obtained from the matrices E_k !

LU factorization

$$E_k = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & 0 & 1 & & & \vdots \\ \vdots & & -l_{k+1,k} & \ddots & & \vdots \\ \vdots & & \vdots & & \ddots & 0 \\ 0 & 0 & -l_{n,k} & 0 & & 1 \end{pmatrix} \quad \text{with} \quad l_{i,k} = \frac{a_{i,k}^k}{a_{k,k}^k}$$

It is easy to see

$$E_k^{-1} = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & \ddots & & & \vdots \\ \vdots & 0 & 1 & & & \vdots \\ \vdots & & l_{k+1,k} & \ddots & & \vdots \\ \vdots & & \vdots & & \ddots & 0 \\ 0 & 0 & l_{n,k} & 0 & & 1 \end{pmatrix}$$

LU factorization

then we luckily have

$$L = E_1^{-1}E_2^{-1}\dots E_{n-1}^{-1} = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ l_{2,1} & 1 & \ddots & & & & & \vdots \\ l_{3,1} & l_{3,2} & 1 & \ddots & & & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & & \ddots & 1 & \ddots & & \vdots \\ \vdots & & & & l_{k+1,k} & \ddots & \ddots & \vdots \\ \vdots & & & & \vdots & \ddots & \ddots & 0 \\ l_{n,1} & \cdots & \cdots & \cdots & l_{n,k} & \cdots & l_{n,n-1} & 1 \end{pmatrix}$$

Theorem: LU factorisation

Let $A = (a_{i,j})$ be a square matrix of size n such as all the submatrices

$$\Delta_k = \begin{pmatrix} a_{1,1} & \cdots & a_{1,k} \\ \vdots & & \vdots \\ a_{k,1} & \cdots & a_{k,k} \end{pmatrix} \quad k \in [\![1, n]\!]$$

are invertible. Then there is a unique lower triangular matrix $L = (l_{i,j})$ with $l_{i,i} = 1$ for all $i \in [\![1, n]\!]$ and an upper triangular matrix U such as

$$A = LU$$

Remark: if we allow row permutations (a pivoting startegy), we have that any square matrix admits a PLU decomposition such that

$$PA = LU$$

Remark 2: this is not obvious, because we may apply a permutation at each kth-step !

LU factorization pseudo-code

LU pseudo-code without pivoting

Initialize L to an identity matrix of dimension $n \times n$ and $U = A$

For $i = 1 \cdots n$
    For $j = (i + 1), \cdots n$
        $\ell_{ji} = u_{ji} / u_{ii}$
        $U_j \rightarrow U_j - \ell_{ji} U_i$
return L, U

LU factorization pseudo-code with pivoting

LU pseudo-code with simple pivoting

Initialize L to an identity matrix of dimension $n \times n$ and $U = A$

For $i = 1 \cdots n$
  Let $k = i$
  While $u_{ii} = 0$
    Swap row U_i with row U_{k+1}
    Swap row P_i with row P_{k+1}
    $k = k + 1$
  For $j = (i + 1), \cdots n$
    $\ell_{ji} = u_{ji} / u_{ii}$
    $U_j \rightarrow U_j - \ell_{ji} U_i$
return L, U

Remark: the partial pivoting strategy consists in looking for the max $u_{k,i}$ value over all remaining rows at the k-th step, and then swapping rows

Outline

Solving a linear system with LU factorization

- we would like to solve

$$Ax = b \iff LUx = b$$

- Once we have L and U, we first solve

$$Ly = b$$

with backward substitution (we have $y = Ux$)

- then we solve

$$Ux = y$$

with forward substitution

Factorization cost

- LU factorisation has the same cost as Gaussian elimination, that is $\frac{2n^3}{3}$ elementary operations
- we need to add forward-backward substitution which adds n^2 elementary operations
- Once the LU factorization is done, we can treat multiple right-hand sides b
- For positive-definite matrices, one can write $A = LL^T$, which reduces the cost by a factor 2. The associated method is called Cholesky factorization (see exercise).
- When the matrix contains many zeros, the computational cost can be highly reduced. We talk about sparse matrices

A few words on sparse matrices

The discretization of many engineering problems requires to solve linear systems with only a few non-zero entries (nz)

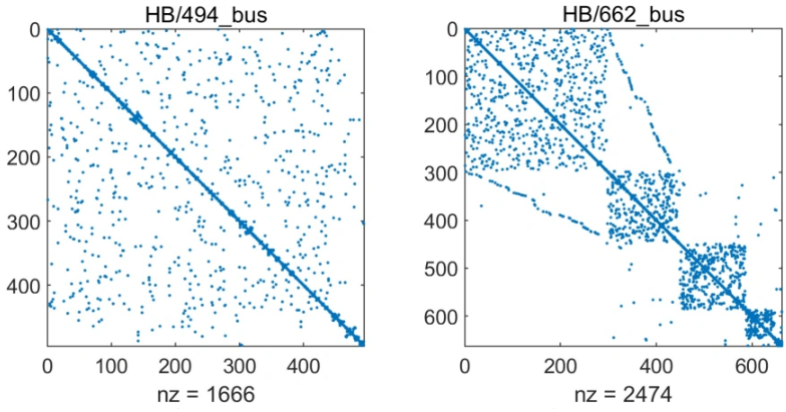

Figure: Example of sparsity patterns for two symmetric positive definite matrices. Nonzero elements are indicated by dots. From Nick Higham blog

A few words on sparse matrices

Bandwidth of a matrix

The bandwidth of a matrix A is the smallest non-negative integer m such that

$$a_{ij} = 0, \quad \text{for } |i - j| > m$$

The LU decomposition does not increase the bandwidth !
The cost of LU factorization reduces to $\mathcal{O}(m^2 N)$

Example

Gaussian elimination for a tridiagonal matrix (Thomas algorithm) - $\mathcal{O}(N)$ operations !

Moreover *reordering* strategies can be highly beneficial and highlight a sparsity pattern.

Special libraries are designed for sparse matrices and are routinely used in the industry: `MUMPS`, `pardiso`, `superLU`, `UMFPACK`, ...

Outline

Outline

Introduction

Given an invertible matrix A inversible, we would like to determine the solution to the linear system

$$Au = b$$

by an iterative method. We focus the analysis on square matrices.
we would like to build a sequence of vectors $(u_k)_{k\geq 0}$ such that

1. $$\begin{cases} u_{k+1} = Bu_k + c, & \forall\ k \geq 0 \\ u_0 \text{ given} \end{cases}$$

   where the matrix B and the vector c are defined from A and b
2. the sequence $(u_k)_{k\geq 0}$ converges and the limit u is the solution to the linear system.

Remark: we solve the system only computing matrix-vector products at each iteration

Some basic notions

Convergence of an iterative method

These statements are equivalent :

- the iterative method converges, that is $(u_k)_{k\geq 0} \to u$ for any initial u_0
- the spectral radius is less than 1

$$\rho(B) < 1$$

- for at least one matrix norm $\|.\|$

$$\|B\| < 1$$

Remark: the iterative method is a fixed-point approach for

$$\begin{array}{rcl} f : \mathbb{C}^n & \to & \mathbb{C}^n \\ v & \to & Bv + c \end{array}$$

which is a contraction when $\|B\| < 1$

A few notes on matrix norms and spectral radius

Let $|.|$ be a vector norm. The induced matrix norm of a $m \times n$ matrix writes

$$\|A\| = \max_{x \neq 0} \frac{|Ax|}{|x|} = \max_{|\mathbf{x}|=1} |Ax|.$$

We have consistency properties

$$\|AB\| \leq \|A\|\|B\|, \quad |Ax| \leq \|A\||x|$$

in addition to the usual norm properties.

Remark: there are also non-induced matrix norms (example Fröbenius norm)
Remark 2: for any induced norm $\rho(A) \leq \|A\|$

Some induced matrix norms

- 2-norm

$$\|A\|_2 = \sqrt{\rho(A^T A)}$$

with $\rho(B) = \max\{|\lambda_i|, \lambda_i \text{ is an eigenvalue of } B\}$

- ∞-norm

$$\|A\|_\infty = \max_{1 \le i \le m} \sum_{j=1}^{n} |a_{ij}|$$

- 1-norm

$$\|A\|_1 = \max_{1 \le j \le n} \sum_{i=1}^{m} |a_{ij}|$$

Remark: the 2-norm is harder to compute in practice

Speed of convergence

Let $\|.\|$ be a matrix norm and u such that

$$u = Bu + c$$

If we consider the iterative method

$$u_{k+1} = Bu_k + c, \quad \forall\ k \geq 0$$

we have

$$|u_k - u| \leq \|B^k\||u_0 - u|$$

The convergence speed of u_k towards u depends on the convergence speed of B^k towards 0, so we need $\|B\| \leq 1$. Moreover, we have (Gelfand's formula)

$$\lim_{k\to\infty} \|B^k\|^{1/k} = \rho(B).$$

Hence the spectral radius $\rho(B)$ tells us about the speed of convergence of the iterative method.

Speed of convergence - intuition

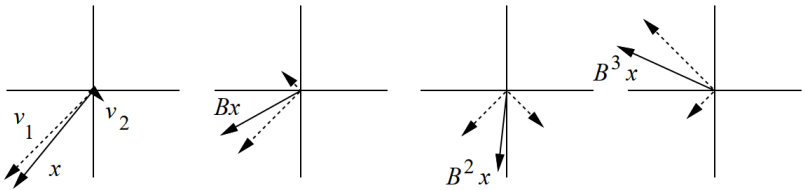

Figure 11: The vector x (solid arrow) can be expressed as a linear combination of eigenvectors (dashed arrows), whose associated eigenvalues are $\lambda_1 = 0.7$ and $\lambda_2 = -2$. The effect of repeatedly applying B to x is best understood by examining the effect of B on each eigenvector. When B is repeatedly applied, one eigenvector converges to zero while the other diverges; hence, $B^i x$ also diverges.

Figure: From Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain.

Another matrix decomposition

Given a linear system

$$Au = b,$$

we suppose that A is invertible and that it can be written under the form

$$A = M - N$$

where M and N are two matrices with M invertible.

We have

$$Au = b \iff Mu = Nu + b \iff u = M^{-1}Nu + M^{-1}b$$

If we set $B = M^{-1}N$ and $c = M^{-1}b$, we obtain $u = Bu + c$ and the iterative method reads

$$u_{k+1} = Bu_k + c = M^{-1}Nu_k + M^{-1}b$$

Another matrix decomposition

Remarks

- we do not compute M^{-1} ! it is enough to solve at each iteration

$$Mu_{k+1} = Nu_k + b$$

- to this end we need to choose M such that the system is easy to solve

Towards a decomposition of A

Let

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & \cdots & \cdots & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ddots & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & -F & \vdots \\ \vdots & & \ddots & D & \ddots & & \vdots \\ \vdots & -E & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & & \ddots & a_{n-1,n-1} & a_{n-1,n} \\ a_{n,1} & \cdots & \cdots & \cdots & \cdots & a_{n,n-1} & a_{n,n} \end{pmatrix}$$

$$A = D - E - F$$

D diagonal matrix, E lower triangular matrix, F upper triangular matrix

We suppose $a_{i,i} \neq 0$ for all $i \in [\![1, n]\!]$

Outline

Jacobi iterative method

The simplest decomposition is

$$M = D, \quad N = E + F$$

$$Au = b \Longleftrightarrow Du = (E + F)u + b \Longleftrightarrow u = D^{-1}(E + F)u + D^{-1}b$$

we obtain the Jacobi method

$$Du_{k+1} = (E + F)u_k + b \Longleftrightarrow u_{k+1} = D^{-1}(E + F)u_k + D^{-1}b$$

The Jacobi iteration matrix is

$$J = D^{-1}(E + F)$$

Jacobi algorithm

$$
\left[
\begin{array}{l}
u^0 \text{ given} \\
\text{For } k = 0 \text{ to } \ldots \text{ (stopping criterion)} \\
\quad \left[
\begin{array}{l}
\text{For } i = 1 \text{ to } n \\
u_i^{k+1} := \left(- \sum_{p \neq i} a_{ip} u_p^k + b_i \right) / a_{ii}.
\end{array}
\right.
\end{array}
\right.
$$

Remarks

- we need to keep in memory all the u_p^k to obtain u_i^{k+1}
- we can reduce the memory usage by replacing the computed u_p^k by u_p^{k+1} in the sum

Outline

Gauss-Seidel algorithm

This gives the Gauss-Seidel algorithm

$$
\left[\begin{array}{l}
u^0 \text{ given} \\
\text{For } k = 0 \text{ to } \ldots \text{ (stopping criterion)} \\
\left[\begin{array}{l}
\text{For } i = 1 \text{ to } n \\
u_i^{k+1} := \left(- \sum_{p<i} a_{ip} u_p^{k+1} - \sum_{p>i} a_{ip} u_p^k + b_i \right) / a_{ii}.
\end{array}\right.
\end{array}\right.
$$

Gauss-Seidel iterative method

In its matrix form, we have

$$Du_{k+1} = Eu_{k+1} + Fu_k + b$$

which we rewrite as

$$(D - E)u_{k+1} = Fu_k + b \iff u_{k+1} = (D - E)^{-1}Fu_k + (D - E)^{-1}b$$

here we have

$$M = D - E, \quad N = F$$

(M is invertible since $a_{i,i} \neq 0$ for all $i \in [\![1, n]\!]$)

The iteration Gauss-Seidel matrix $\mathcal{L}_1$ is

$$\mathcal{L}_1 = (D - E)^{-1}F$$

Outline

Relaxation method

Remarks

- If the Gauss-Seidel method converges, we can add a real parameter $\omega \neq 0$ such that

$$A = M - N = \left(\frac{D}{\omega} - E\right) - \left(\frac{1-\omega}{\omega}D + F\right)$$

- if the method converges $\rho(\mathcal{L}_1) < 1$, and by continuity of the spectral radius the iterative method must converge when ω is close to 1
- a similar approach could be used for any iterative method

The point-wise update is at the k-th iteration is

$$u_i^{k+1} = (1-\omega)u_i^k + \frac{\omega}{a_{ii}}\left(b_i - \sum_{j<i} a_{ij}u_j^{k+1} - \sum_{j>i} a_{ij}u_j^k\right), \quad i = 1, 2, \ldots, n.$$

Relaxation method

The associated iterative method is

$$\left(\frac{D}{\omega} - E\right) u_{k+1} = \left(\frac{1-\omega}{\omega} D + F\right) u_k + b$$

The iterative matrix is called relaxation matrix $\mathcal{L}_\omega$

$$\mathcal{L}_\omega = \left(\frac{D}{\omega} - E\right)^{-1} \left(\frac{1-\omega}{\omega} D + F\right)$$

Remark

The relaxation method consists in finding

- an interval I ($0 \notin I$) such that $\omega \in I \Rightarrow \rho(\mathcal{L}_\omega) < 1$
- an optimal relaxation parameter $\omega_0 \in I$ such that

$$\rho(\mathcal{L}_{\omega_0}) = \min_{\omega \in I} \rho(\mathcal{L}_\omega)$$

Outline

Convergence of Jacobi, Gauss-Seidel and relaxation methods

Example

- For

$$A_1 = \begin{pmatrix} 1 & 2 & -2 \\ 1 & 1 & 1 \\ 2 & 2 & 1 \end{pmatrix}.$$

  we show that $\rho(J) < 1 < \rho(\mathcal{L}_1)$.

- For

$$A_2 = \begin{pmatrix} 2 & -1 & 1 \\ 2 & 2 & 2 \\ -1 & -1 & 2 \end{pmatrix}.$$

  we show that $\rho(\mathcal{L}_1) < 1 < \rho(J)$.

Convergence of Jacobi, Gauss-Seidel and relaxation methods

Theorem: necessary condition for the convergence of the relaxation method

for all $\omega \neq 0$, we have

$$\rho(\mathcal{L}_\omega) \geq |\omega - 1|$$

thus if $\omega \notin]0, 2[$ the relaxation method does not converge.

Theorem: sufficient convergence condition for the relaxation method

If A is a symmetric positive-definite matrix, the relaxation method converges for all $\omega \in]0, 2[$. In particular the Gauss-Seidel method converges.

Remark: the theorem holds for hermitian matrices.

Convergence of Jacobi, Gauss-Seidel and relaxation methods

Theorem

If A is strictly diagonal dominant

$$\forall i = 1, ..., N \qquad |a_{ii}| > \sum_{j \neq i} |a_{ij}|$$

then A is invertible and the Gauss-Seidel and Jacobi methods converge.

Convergence of Jacobi, Gauss-Seidel and relaxation methods

Remarks:

- For many systems coming from discretization of PDEs, the Gauss-Seidel method converges faster than the Jacobi method
- For a large family of such systems, we can show that there exists a optimal parameter $\omega^\star$ for which the relaxation method is way more efficient: the number of required iterations for a given precision drops when $\omega \approx \omega^\star$

Convergence of Jacobi, Gauss-Seidel and relaxation methods

Let us note

$$L = D^{-1}E, \quad U = D^{-1}F$$

hence

$$\begin{aligned} A &= D(I - L - U), \quad J = L + U \\ L_\omega &= (I - \omega L)^{-1}((1-\omega)I + \omega U). \end{aligned}$$

Definition

we say that A if of type (V), if for $\alpha \neq 0$, the eigenvalues of $J(\alpha) = \alpha L + \frac{1}{\alpha}U$ are independent of α.

Remarks

- tridiagonal matrices are of type (V)
- many matrices form PDE discretization are of type (V)

Theorem

Let A be a matrix of type (V). Then
(i) $\rho(\mathcal{L}_1) = \rho(J)^2$
so Gauss-Seidel method converges if and only if Jacobi mathod converges, and it converges twice as fast.
(ii) If moreover, the eigenalues of J are real and $\rho(J) < 1$, then $\omega^* = \frac{2}{1+\sqrt{1-\rho(J)^2}}$

The graph of $\rho(\mathcal{L}_\omega)$ has the form $(\mu = \rho(J))$:

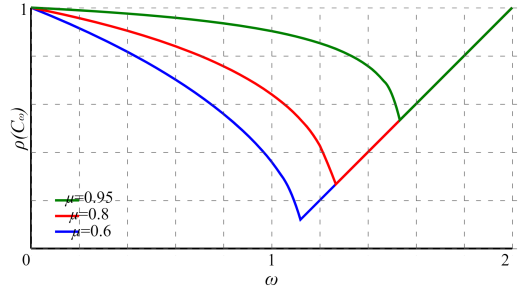

Figure: From *Wikipedia*

Summary of the contents

We have seen two families of methods for solving linear systems

Direct methods :

1. The modern form of Gaussian Elimination is called LU decomposition
2. A pivoting strategy is required for numerical stability
3. Cholesky decomposition is used for positive-definite matrices
4. Special algorithms are used for sparse matrices to reduce the computational cost

Iterative methods :

1. We have seen three types of fixed-point algorithms (Jacobi, Gauss-Seidel, SOR)
2. The convergence depends on the spectral radius of the matrix
3. They are advocated for very large systems, where direct methods become too costly

A few words about conditioning

Conditioning has a crucial role when solving linear systems. For a given induced matrix norm, it is defined as

$$\kappa(A) = \|A\|\|A^{-1}\|$$

Forward error analysis

Let A be invertible and u and $u + \hat{u}$ be the solutions of the linear systems

$$Au = b, \quad A(u + \hat{u}) = b + \hat{b}$$

If $b \neq 0$ then the inequality

$$\frac{|\hat{u} - u|}{|u|} \leq \kappa(A) \frac{|\hat{b} - b|}{|b|}$$

holds and it is optimal.

A few words about conditioning

Backward error analysis

Let A be invertible and u and $u + \tilde{u}$ be the solutions of the linear systems

$$Au = b, \quad (A + \tilde{A})(u + \tilde{u}) = b$$

If $b \neq 0$ then the inequality

$$\frac{|\tilde{u}|}{|u + \tilde{u}|} \leq \kappa(A) \frac{\|\tilde{A}\|}{\|A\|}$$

holds and it is optimal

If A is symmetric positive definite with eigenvalues $\lambda_1 \leq \cdots \leq \lambda_n$

$$\kappa_2(A) = \frac{\lambda_n}{\lambda_1}$$

For a more general matrix we use the singular values of A to define $\kappa_2(A)$.
In practice, only the order of magnitude of $\kappa(A)$ matters.