Numerical Linear Algebra

Decompositions, numerical aspects

Gerard Sleijpen and Martin van Gijzen

September 27, 2017

Program Lecture 2

- LU-decomposition
  - Basic algorithm
  - Cost
  - Stability
  - Pivoting
- Cholesky decomposition
- Sparse matrices and reorderings

Gaussian elimination

Consider the system

$$\mathbf{Ax} = \begin{bmatrix} 2 & 1 & 1 \\ 4 & 1 & 0 \\ -2 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 7 \end{bmatrix}$$

Then we can reduce this system to upper triangular form

1. Subtract $2\times$ equation one from equation two
2. Subtract $-1\times$ equation one from equation three
3. Subtract $-3\times$ the second equation from the third

Gaussian elimination (2)

The resulting equivalent system is

$$\mathbf{U}\mathbf{x} = \begin{bmatrix} 2 & 1 & 1 \\ 0 & -1 & -2 \\ 0 & 0 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ -4 \\ -4 \end{bmatrix}$$

This system can be solved by **back substitution**.

If we have a different right-hand side, do we have to do the same operations? Or can we save what we did?

Gaussian elimination (3)

The first reduction step was ‘Subtract $2\times$ equation one from equation two’. The so-called **elementary matrix** $\mathbf{E}_{2,1}$ that performs this operations is

$$\mathbf{E}_{2,1} \equiv \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \mathbf{I} - 2\,\mathbf{e}_2\,\mathbf{e}_1^*.$$

Here, $\mathbf{e}_1 \equiv (1,0,0)^T$, ...

Multiplying $\mathbf{A}$ with this matrix yields

$$\mathbf{E}_{2,1}\mathbf{A} = \begin{bmatrix} 2 & 1 & 1 \\ 0 & -1 & -2 \\ -2 & 2 & 1 \end{bmatrix}.$$

Gaussian elimination (4)

The second reduction step“Subtract $-1\times$ equation one from equation three’ is equivalent with multiplying with the matrix

$$\mathbf{E}_{3,1} \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \mathbf{I} + \mathbf{e}_3\,\mathbf{e}_1^*$$

and ’Subtract $-3\times$ the second equation from the third’ is equivalent with multiplying with the matrix

$$\mathbf{E}_{3,2} \equiv \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 3 & 1 \end{bmatrix} = \mathbf{I} + 3\,\mathbf{e}_3\,\mathbf{e}_2^*.$$

Gaussian elimination (5)

Hence, we can write

$$\mathbf{E}_{3,2}\,\mathbf{E}_{3,1}\,\mathbf{E}_{2,1}\,\mathbf{A} = \mathbf{U}$$

Notice that the matrix $\mathbf{E}_{3,2}\,\mathbf{E}_{3,1}\,\mathbf{E}_{2,1}$ is a product of lower triangular matrices and therefore lower triangular.
The above equation can also be written as

$$\mathbf{A} = \mathbf{E}_{2,1}^{-1}\,\mathbf{E}_{3,1}^{-1}\,\mathbf{E}_{3,2}^{-1}\,\mathbf{U}$$

Since the inverse of a lower triangular matrix is lower triangular, the matrix $\mathbf{E}_{2,1}^{-1}\,\mathbf{E}_{3,1}^{-1}\,\mathbf{E}_{3,2}^{-1}$ must be lower triangular:

$$\mathbf{L} \equiv \mathbf{E}_{2,1}^{-1}\,\mathbf{E}_{3,1}^{-1}\,\mathbf{E}_{3,2}^{-1}, \qquad \mathbf{A} = \mathbf{L}\mathbf{U}$$

Gaussian elimination (6)

It is easy to check that

$$\mathbf{E}_{2,1}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{E}_{3,1}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \mathbf{E}_{3,2}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{bmatrix},$$

and that

$$\mathbf{L} = \mathbf{E}_{2,1}^{-1}\,\mathbf{E}_{3,1}^{-1}\,\mathbf{E}_{3,2}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & -3 & 1 \end{bmatrix}$$

Gaussian elimination (6)

It is easy to check that $(\mathbf{I} - \alpha\,\mathbf{e}_i\,\mathbf{e}_j^*)^{-1} = \mathbf{I} + \alpha\,\mathbf{e}_i\,\mathbf{e}_j^* \qquad (i \neq j)$

$$\mathbf{E}_{2,1}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{E}_{3,1}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \mathbf{E}_{3,2}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{bmatrix},$$

and that

$$\mathbf{L} = \mathbf{E}_{2,1}^{-1}\,\mathbf{E}_{3,1}^{-1}\,\mathbf{E}_{3,2}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ -1 & -3 & 1 \end{bmatrix}$$

$$(\mathbf{I} + \alpha\,\mathbf{e}_i\,\mathbf{e}_j^*)(\mathbf{I} + \beta\,\mathbf{e}_k\,\mathbf{e}_\ell^*) = (\mathbf{I} + \alpha\,\mathbf{e}_i\,\mathbf{e}_j^* + \beta\,\mathbf{e}_k\,\mathbf{e}_\ell^*) \qquad (j \neq k).$$

Gaussian elimination (7)

Clearly $\mathbf{L}$ is lower triangular, with 1's on the main diagonal. The special thing is that *the entries below the diagonal are exactly the multipliers* 2, -1, -3 *used in the elimination steps*.

The example shows that an "**LU-decomposition**" can be made by

- Reducing $\mathbf{A}$ to upper triangular form by elementary row operations, this gives $\mathbf{U}$,
- Storing all the multipliers in a lower triangular matrix $\mathbf{L}$ that has ones on the main diagonal.

Gaussian elimination algorithm

Given $\mathbf{A} = (a_{ij}) \in \mathbb{C}^{n \times n}$, the following algorithm computes the **LU-factorisation**: $\mathbf{A} = \mathbf{LU}$.

for $k = 1, \ldots, n-1$ do
  for $i = k+1, \ldots, n$ do
    $\eta = a_{ik} / a_{kk}$
    $a_{ik} = \eta$
    for $j = k+1, \ldots, n$
      $a_{ij} \leftarrow a_{ij} - \eta\, a_{kj}$
    end for
  end for
end for

The element a_{kk} is called the kth **pivot**.

Some remarks (1)

- The algorithm overwrites the matrix $\mathbf{A}$ with the matrix $\mathbf{U}$ (upper triangular part) and the matrix $\mathbf{L}$ (strictly lower triangular part). The main diagonal of $\mathbf{L}$ is not stored.
- The numerical stability of the algorithm depends on the size of the pivots. If a pivot is zero the algorithm breaks down. If a pivot is close to zero large numerical errors may occur.
- The number of floating point operations to compute the LU-decomposition is

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

Some remarks (2)

- A more 'symmetrical' variant of the LU-decomposition is the **LDU-decomposition**. Here $\mathbf{D}$ is a diagonal that scales the main diagonal elements of $\mathbf{U}$ to one.
- A system of the form $\mathbf{Ax} = \mathbf{LUx} = \mathbf{b}$ can be solved by a **forward substitution**: solve $\mathbf{Ly} = \mathbf{b}$ for $\mathbf{y}$ $(\mathbf{y} \equiv \mathbf{Ux})$, and a **back substitution**: solve $\mathbf{Ux} = \mathbf{y}$ for $\mathbf{x}$.
  Once the LU-decomposition of $\mathbf{A}$ is known, any system with $\mathbf{A}$ can simply be solved by forward and back substitution with the LU-factors.
- Both the back and forward substitution (algorithms given in the next two slides) require n^2 flops.

Forward substitution

Given an $n \times n$ non-singular lower triangular matrix $\mathbf{L} = (\ell_{ij})$ and $\mathbf{b} = (b_i) \in \mathbb{C}^n$, the following algorithm finds $\mathbf{y} = (y_i) \in \mathbb{C}^n$ such that $\mathbf{L}\mathbf{y} = \mathbf{b}$.

Forward substitution algorithm

for $i = 1, \ldots, n$ do
    $y_i = b_i$
    for $j = 1, \ldots, i-1$ do
        $y_i \leftarrow y_i - \ell_{ij}\, y_j$
    end for
    $y_i \leftarrow y_i / \ell_{ii}$
end for

Backward substitution

Given an $n \times n$ non-singular upper triangular matrix $\mathbf{U} = (u_{ij})$ and $\mathbf{y} = (y_i) \in \mathbb{C}^n$, the following algorithm finds $\mathbf{x} = (x_i) \in \mathbb{C}^n$ such that $\mathbf{U}\mathbf{x} = \mathbf{y}$.

Back substitution algorithm

for $i = n, \ldots, 1$ do
    $x_i = y_i$
    for $j = i + 1, \ldots, n$ do
        $x_i \leftarrow x_i - u_{ij}\, x_j$
    end for
    $x_i \leftarrow x_i / u_{ii}$.
end for

Round off errors, a bound

Recall

Convention. For $\mathbf{A} = (a_{ij})$ and $\mathbf{B} = (b_{ij})$ in $\mathbb{C}^{m\times n}$,

- $|\mathbf{A}| \equiv (|a_{ij}|)$
- $\mathbf{B} \leq \mathbf{A}$ means $b_{ij}, a_{ij} \in \mathbb{R}$, $b_{ij} \leq a_{ij}$ (all i, j)

Round off errors, a bound

Theorem. Let $\widehat{\mathbf{L}}$ and $\widehat{\mathbf{U}}$ be the computed LU-factors of the $n \times n$ floating point matrix $\mathbf{A}$. If $\widehat{\mathbf{y}}$ is the computed solution of $\widehat{\mathbf{L}}\mathbf{y} = \mathbf{b}$ and $\widehat{\mathbf{x}}$ is the computed solution of $\widehat{\mathbf{U}}\mathbf{x} = \widehat{\mathbf{y}},$ then

$$(\mathbf{A} + \Delta_A)\widehat{\mathbf{x}} = \mathbf{b} \quad \text{with} \quad |\Delta_A| \leq 3\,n\,\mathbf{u}\,|\widehat{\mathbf{L}}|\,|\widehat{\mathbf{U}}| + \mathcal{O}(\mathbf{u}^2).$$

Proof. see Golub and van Loan p.107 for a slightly weaker result

or Exer. 2.16, in our exercise set.

Round off errors, discussion

$$(\mathbf{A} + \Delta_A)\widehat{\mathbf{x}} = \mathbf{b} \quad \text{with} \quad |\Delta_A| \leq 3\,n\,\mathbf{u}\,|\widehat{\mathbf{L}}|\,|\widehat{\mathbf{U}}|.$$

- Except for the factor $3n$, the bound is sharp.
- A bound on Δ_A better than $|\Delta_A| \leq n\,\mathbf{u}\,|\mathbf{A}|$ is not to be expected: because (if $\mathbf{A}$ is dense) the multiplication $\mathbf{Ax}$ in floating point arithmetic already results in a vector equal to $(\mathbf{A} + \Delta_A)\mathbf{x}$ with Δ_A sharply satisfying $|\Delta_A| \leq n\,\mathbf{u}\,|\mathbf{A}|$.
- Except for the modest constant 3, the factor

$$\frac{\| \, |\widehat{\mathbf{L}}|\,|\widehat{\mathbf{U}}| \, \|}{\| \, |\mathbf{A}| \, \|}$$

appears to characterize the **stability of Gaussian elimination**.

Round off errors, discussion (2)

The term $|\widehat{\mathbf{L}}||\widehat{\mathbf{U}}|$ can be much larger than $|\mathbf{A}|$ if a small pivot is encountered during the elimination process:

Example.

$$\mathbf{A} = \begin{bmatrix} \epsilon & 1 \\ 1 & 0 \end{bmatrix}$$

Assignment. Compute the LU factors of $\mathbf{A}$ and bound $|\mathbf{L}|\,|\mathbf{U}|$.

Note. Small pivots do not necessarily indicate an ill-conditioned problem.

Conclusion. Gaussian elimination (without further precautions) can give arbitrarily poor results, even for well-conditioned problems. The method may be unstable, depending on $\mathbf{A}$.

Permutations

The problem in the previous example can of course be solved by interchanging the rows (or columns). This is a row permutation.

Let $\mathbf{e}_i$ denote the ith column of the identity matrix $\mathbf{I}$.

A row permuted version of $\mathbf{A}$ is given by $\mathbf{PA}$, where

the jth row of the permutation matrix $\mathbf{P}$ equals $\mathbf{e}_{\pi(j)}^T$

and π permutes $(1, 2, \ldots, n)$.

The matrix $\mathbf{AP}^T$ is a column permuted version of $\mathbf{A}$.

Permutations

The problem in the previous example can of course be solved by interchanging the rows (or columns). This is a row permutation.

Let $\mathbf{e}_i$ denote the ith column of the identity matrix $\mathbf{I}$.

A row permuted version of $\mathbf{A}$ is given by $\mathbf{PA}$, where

the jth row of the permutation matrix $\mathbf{P}$ equals $\mathbf{e}_{\pi(j)}^T$

and π permutes $(1, 2, \ldots, n)$.

Note. If $\texttt{Pi} \equiv [\pi(1), \ldots, \pi(n)]$, then, in MATLAB, `A(Pi,:)` equals $\mathbf{PA}$, while `A(:,Pi)` equals $\mathbf{AP}^T$.

Exchanging rows (and/or columns) for selecting (large) pivots is called **pivoting**.

Partial pivoting

A simple strategy to avoid small pivots is **partial pivoting**:

- Determine the in absolute value largest element below the pivot.
- Interchange the corresponding row with the pivot row.
- Eliminate all non-zeros elements below the pivot.

Practical remarks. Interchange the rows in the 'active' part of the matrix as well as the corresponding rows in the already constructed 'L-part'.
Define `Pi=1:n` at the start of the LU-decompostion process and interchange `Pi(i)` and `Pi(k)` whenever row i and row k are interchanges. Then `Pi` defines the permutation $\mathbf{P}$.

Partial pivoting (2)

Partial pivoting leads to the decomposition

$$\mathbf{PA} = \mathbf{LU}.$$

The permutation matrix $\mathbf{P}$ corresponds to all the row exchanges.

With partial pivoting, no multiplier is > 1 in absolute value:

$$\|\mathbf{L}\|_M \leq 1, \quad \text{where} \quad \|\mathbf{A}\|_M \equiv \max_{i,j} |a_{ij}|.$$

Partial pivoting (2)

Partial pivoting leads to the decomposition

$$\mathbf{PA} = \mathbf{LU}.$$

The permutation matrix $\mathbf{P}$ corresponds to all the row exchanges.

With partial pivoting, no multiplier is > 1 in absolute value:

$$\|\mathbf{L}\|_M \leq 1, \quad \text{where} \quad \|\mathbf{A}\|_M \equiv \max_{i,j} |a_{ij}|.$$

In particular, with $\rho_{part}(\mathbf{A}) \equiv \|\mathbf{U}\|_M / \|\mathbf{A}\|_M$,
we then have $\| \, |\widehat{\mathbf{L}}| \, |\widehat{\mathbf{U}}| \, \|_\infty \leq n^2 \, \rho_{part}(\mathbf{A}) \, \|\mathbf{A}\|_\infty$.

n^2 can be repaced by the square of the bandwidth of $\mathbf{A}$.

Note that n^3 can be large for high-dimensional systems.

Partial pivoting (2)

Partial pivoting leads to the decomposition

$$\mathbf{PA} = \mathbf{LU}.$$

The permutation matrix $\mathbf{P}$ corresponds to all the row exchanges.

With partial pivoting, no multiplier is > 1 in absolute value:

$$\|\mathbf{L}\|_M \leq 1, \quad \text{where} \quad \|\mathbf{A}\|_M \equiv \max_{i,j} |a_{ij}|.$$

In particular, with $\rho_{part}(\mathbf{A}) \equiv \|\mathbf{U}\|_M / \|\mathbf{A}\|_M$,
we then have $\| \, |\widehat{\mathbf{L}}| \, |\widehat{\mathbf{U}}| \, \|_\infty \leq n^2 \, \rho_{part}(\mathbf{A}) \, \|\mathbf{A}\|_\infty$.

Theorem. We have the sharp bound $\rho_{part}(\mathbf{A}) \leq 2^{n-1}$.

See, Exer. 2.19.

Partial pivoting (2)

Partial pivoting leads to the decomposition

$$\mathbf{PA} = \mathbf{LU}.$$

The permutation matrix $\mathbf{P}$ corresponds to all the row exchanges.

With partial pivoting, no multiplier is > 1 in absolute value:

$$\|\mathbf{L}\|_M \leq 1, \quad \text{where} \quad \|\mathbf{A}\|_M \equiv \max_{i,j} |a_{ij}|.$$

In particular, with $\rho_{part}(\mathbf{A}) \equiv \|\mathbf{U}\|_M / \|\mathbf{A}\|_M$,
we then have $\| \, |\widehat{\mathbf{L}}| \, |\widehat{\mathbf{U}}| \, \|_\infty \leq n^2 \, \rho_{part}(\mathbf{A}) \, \|\mathbf{A}\|_\infty$.

Theorem. We have the sharp bound $\rho_{part}(\mathbf{A}) \leq 2^{n-1}$.

Wilkinson's miracle [1963]. In practise, $\rho(\mathbf{A}) \leq 16$.

Partial pivoting (2)

Partial pivoting leads to the decomposition

$$\mathbf{PA} = \mathbf{LU}.$$

The permutation matrix $\mathbf{P}$ corresponds to all the row exchanges.

With partial pivoting, no multiplier is > 1 in absolute value:

$$\|\mathbf{L}\|_M \leq 1, \quad \text{where} \quad \|\mathbf{A}\|_M \equiv \max_{i,j} |a_{ij}|.$$

In particular, with $\rho_{part}(\mathbf{A}) \equiv \|\mathbf{U}\|_M / \|\mathbf{A}\|_M$,
we then have $\| \, |\widehat{\mathbf{L}}| \, |\widehat{\mathbf{U}}| \, \|_\infty \leq n^2 \, \rho_{part}(\mathbf{A}) \, \|\mathbf{A}\|_\infty$.

Theorem. We have the sharp bound $\rho_{part}(\mathbf{A}) \leq 2^{n-1}$.

Note. Computing $\| \, |\widehat{\mathbf{L}}| \, |\widehat{\mathbf{U}}| \, \|_\infty = \| \, |\widehat{\mathbf{L}}| \, (|\widehat{\mathbf{U}}|\mathbf{1}) \, \|_\infty$ costs n^2 flop.

Partial pivoting (3)

According to Wilkinson's miracle, LU-decomposition with partial pivoting is usually sufficiently stable (a modest factor [n^2?] less stable than the system $\mathbf{Ax} = \mathbf{b}$).

The stability can be estimated from $\rho(\mathbf{A})$ (or $\| \, |\widehat{\mathbf{L}}| \, |\widehat{\mathbf{U}}| \, \|_\infty / \|\mathbf{A}\|_\infty$).

In the extremely rare (according to Wilkinson) cases where this quantity is large, one can switch to a more stable approach:

- **Iterative refinement**
  (using the same "expensive" LU-factors).
- **Complete pivoting** Th. $\rho_{comp}(\mathbf{A}) \leq c\sqrt{n}\, n^{\frac{1}{4}\ln n}$
  (where it was conjectured that $\rho_{comp}(\mathbf{A}) \lesssim n$).
- Form a **QR-decomposition** (which is $2\times$ as expensive as an LU-decomp., but stable. Here $\mathbf{Q}$ is unitary, $\mathbf{R}$ upper Δ).

Row scaling

The stability of the *system* $\mathbf{A}\mathbf{x} = \mathbf{b}$ can often be improved by scaling the rows (**row equilibration**):

find a diagonal matrix $\mathbf{D}$ such that the condition number of the matrix $\mathbf{D}^{-1}\mathbf{A}$ is (much) smaller than that of the matrix $\mathbf{A}$ and solve the scaled system

$$\widetilde{\mathbf{A}}\mathbf{x} = \widetilde{\mathbf{b}} \quad \text{with} \quad \widetilde{\mathbf{A}} \equiv \mathbf{D}^{-1}\mathbf{A} \quad \text{and} \quad \widetilde{\mathbf{b}} \equiv \mathbf{D}^{-1}\mathbf{b}$$

using LU-decomposition with partial pivoting.

Example. $\mathbf{D} = \text{diag}(\|\mathbf{e}_1^*\mathbf{A}\|_\infty, \ldots, \|\mathbf{e}_n^*\mathbf{A}\|_\infty)$

Symmetric positive definite systems

In many applications the matrix $\mathbf{A} = (a_{ij}) \in \mathbb{C}^{n \times n}$, used in the linear system $\mathbf{Ax} = \mathbf{b}$, is **positive definite**, that is,

$$\mathbf{x}^* \mathbf{A} \mathbf{x} > 0 \qquad (\mathbf{x} \in \mathbb{C}^n,\ \mathbf{x} \neq \mathbf{0}).$$

Note. If $\mathbf{A}$ is positive definite, then $a_{ii} > 0$ for all i, and $\mathbf{A}$ is **Hermitian**, i.e., $\mathbf{A} = \mathbf{A}^*$.

For this type of matrices, memory and CPU time can be saved: since $\mathbf{A}$ is Hermitian only the elements a_{ij} with $i \leq j$ have to be stored in memory.

Moreover, an 'efficient' LU-like decomposition exists.

The Cholesky decomposition

Let $\mathbf{A} = (a_{ij})$ be positive definite. Then the following **algorithm** computes the **Cholesky decomposition** $\mathbf{A} = \mathbf{C}\mathbf{C}^*$ of $\mathbf{A}$, i.e., $\mathbf{C} = (c_{ij}) \in \mathbb{C}^{n \times n}$ is lower triangular and $c_{ii} > 0$ all i.

for $k = 1, 2, \ldots, n$ do

$$a_{kk} \leftarrow (a_{kk} - \sum_{p=1}^{k-1} a_{kp}^2)^{1/2}$$

for $i = k+1, \ldots, n$ do

$$a_{ik} \leftarrow (a_{ik} - \sum_{p=1}^{k-1} a_{ip}\, a_{kp})/a_{kk}$$

end for

end for

The entry a_{ij} is overwritten by c_{ij} $\quad (i \geq j)$.

The Cholesky decomposition (2)

The Chol. dec. for p.d. matrices has many favourable properties:

- It exists without pivoting.
- $\mathbf{C}$ is real if $\mathbf{A}$ is real.
- The number of flops for the algorithm is $\frac{1}{3}n^3$; both memory and operations are half that of the LU-decomposition for general matrices.
- As for the LU-decomposition, we have that

$$(\mathbf{A} + \Delta_A)\widehat{\mathbf{x}} = \mathbf{b} \quad \text{with} \quad |\Delta_A| \leq 3\,n\,\mathbf{u}\,|\widehat{\mathbf{C}}|\,|\widehat{\mathbf{C}}^*|.$$

  The inequality $\| \, |\mathbf{C}| \, |\mathbf{C}^*| \, \|_2 \leq n\|\mathbf{C}\|_2^2 = n\|\mathbf{A}\|_2$ shows that **the Cholesky decomposition is stable**: pivoting is not necessary (and is even harmful: it would destroy symmetry).

Banded systems

In many applications the matrix is **banded**. This is the case whenever the equations can be ordered so that each unknown x_i appears in only a few equations in a 'neighbourhood' of the ith equation.

The matrix $\mathbf{A}$ has **upper bandwidth** q, where $q \geq 0$ is the smallest number such that $a_{ij} = 0$ whenever $j > i + q$, and **lower bandwidth** p, where $p \geq 0$ is the smallest number such that $a_{ij} = 0$ whenever $i > j + p$.

Typical examples are obtained after finite element or finite difference discretizations.

Banded systems (2)

- Substantial reduction of both work and memory can be realised for these systems, since there will only be **fill in** (non zeros in the factors at locations (i, j) with $a_{ij} = 0$) inside the band. In particular, $\mathbf{L}$ and $\mathbf{U}$ inherit the lower and upper bandwidth of $\mathbf{A}$.

This is easily checked by writing down some elimination steps for a banded system of equations.

- The LU-decomposition can now be obtained using $2npq$ flops if $n \gg p$ and $n \gg q$.

- However, the band structure is to a large extend destroyed if partial pivoting is applied.

General sparse systems (1)

Large matrices with general sparsity patterns arise for example in unstructured finite element calculation.

In order to limit the amount of **fill in**, the matrix is usually reordered before it is decomposed.

General sparse systems (2)

Therefore, solving a sparse system $\mathbf{Ax} = \mathbf{b}$ with a direct method normally consists of three phases:

- An analysis phase: a symbolic decomposition is made during which a suitable ordering is determined.
- A decomposition phase: the permuted matrix is decomposed.
- A solution phase: the solutions for one or more right-hand sides are determined using back and forward substitution, and back permutation to the original ordering.

Ordering algorithms

Quite sophisticated ordering algorithms exist to minimise the fill in. They are based on graph theory.

Some general principles that are used are:

- Try to minimise the bandwidth in every elimination step. An example is the **Reverse Cuthill-McKee** algorithm.
- Select rows with the fewest non-zeros. An example is the **minimum degree ordering**.

MATLAB backslash operator (`x=A\b;`) for sparse matrices is of the above type and is very effective (especially for problems from discretized 2-dimensional PDEs and as long the L- and U-factors are not explicitly required).

Concluding remarks

- Direct solution methods are the preferred choice for dense systems.
- Also for sparse methods they are widely used, in particular for positive definite banded systems.
- For very large sparse systems iterative methods are usually preferred. However, most of the time a combination of the two is used. Iterative methods can be used to improve the solution computed with the direct method, or an approximate factorisation is used to accelerate the convergence of an iterative method. These issues will be discussed in the next lectures.

Further reading: Golub and Van Loan, pp. 87-160