

MA5233 Computational Mathematics

Lecture 7: Sparse LU Factorisation

Simon Etter



2019/2020

Sparse LU Factorisation

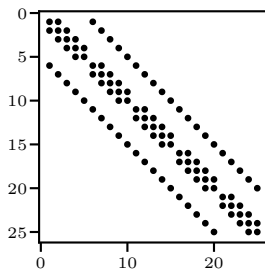
Recap from previous lectures

Poisson equation in 1D:

- ▶ Tridiagonal system of equations.
- ▶ LU factorisation can be performed in $\mathcal{O}(n)$ operations and memory.

Poisson equation in 2D:

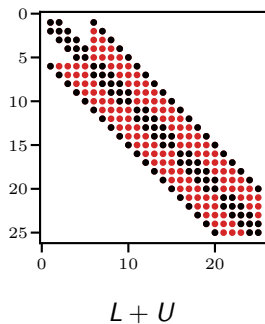
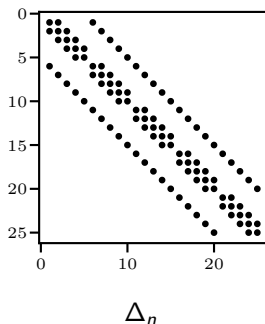
- ▶ More complicated sparsity pattern.
- ▶ Can LU factorisation still be done efficiently?



Sparse LU Factorisation

Bad news

- ▶ LU factors have more nonzero entries than original matrix.
- ▶ These extra entries in L , U are called *fill-in*.
- ▶ Fill-in significantly increases the memory consumption and workload of sparse LU factorisation.



Sparse LU Factorisation

Aim for this lecture

- ▶ Understand how fill-in arises.
- ▶ Find ways to reduce fill-in as much as possible.

Terminology

Let $A = \text{sparse}(i, j, v) = LU$ be a sparse matrix with coordinate-list vectors i, j, v . We introduce the following terms.

- ▶ *Structure of A* : the vectors i, j but not v .
- ▶ *Structurally nonzero fill-in entries*:
entries $L[i, j]$, $U[i, j]$ which are nonzero for some v .

In the following, all statements of the form $A[i, j] \neq 0$ are meant in the structural sense.

Sparse LU Factorisation

Example

Consider

$$A_1 := \begin{pmatrix} 1 & & 1 \\ & 1 & 1 \\ & & 1 \\ 1 & 1 & & 1 \end{pmatrix} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ 1 & 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ & 1 \\ & 1 \\ & & 1 \end{pmatrix} =: L_1 U_1,$$

$$A_2 := \begin{pmatrix} 1 & & 1 \\ & -1 & 1 \\ & & 1 \\ 1 & 1 & & 1 \end{pmatrix} = \begin{pmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ 1 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ & -1 \\ & 1 \\ & & 1 \end{pmatrix} =: L_2 U_2.$$

- ▶ A_1 and A_2 have the same *structure*.
- ▶ $L_2[4, 3] = 0$ is *structurally nonzero* since $L_1[4, 3] \neq 0$.

Sparse LU Factorisation

Graph of a sparse matrix $A \in \mathbb{K}^{n \times n}$

Graph $G(A) := (V(A), E(A))$ defined by

$$V(A) := \{1, \dots, n\}, \quad E(A) := \{j \rightarrow i \mid A[i, j] \neq 0\}.$$

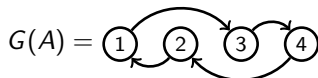
Note transpose in $E(A)$: entry $A[i, j]$ corresponds to edge $j \rightarrow i$.

Path in $G = (V, E)$

Ordered sequence $k_0, \dots, k_p \in V$ such that $k_{q-1} \rightarrow k_q \in E$ for all q .
Number of edges p is called the length of the path.

Example

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



$2 \rightarrow 1 \rightarrow 3$ is a path of length 2.

Sparse LU Factorisation

Path theorem for matrix powers

$$A^p[i, j] \neq 0 \iff \exists \text{ path } j \rightarrow i \text{ of length } p.$$

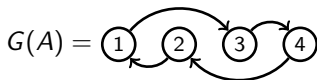
Proof.

$$A^p[i, j] = \sum_{k_{p-1}} \dots \sum_{k_1} A[i, k_{p-1}] \dots A[k_a, k_{a-1}] \dots A[k_1, j].$$

Each term is nonzero iff $j \rightarrow k_1 \rightarrow \dots \rightarrow k_{p-1} \rightarrow i$ is a path in $G(A)$.

Example (continued)

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



$A^2[4, 1] \neq 0$ because $1 \rightarrow 3 \rightarrow 4$ is a path of length 2 in $G(A)$.

$A^2[2, 1] = 0$ because $1 \rightarrow 3 \rightarrow 4 \rightarrow 2$ is a path of length 3 in $G(A)$.

Sparse LU Factorisation

Path theorem for matrix inverses

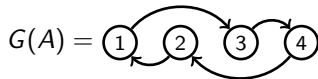
$$A^{-1}[i,j] \neq 0 \iff \exists \text{ path } j \rightarrow i.$$

Proof.

- ▶ $A^{-1} = p(A)$ for polynomial $p(x)$ interpolating $\frac{1}{x}$ on eigenvalues of A .
- ▶ Hence entry (i,j) of $A^{-1} = \sum_{p=0}^{n-1} c_p A^p$ is nonzero if there is a path $j \rightarrow i$ of arbitrary length.

Example (continued)

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



$A^{-1}[2,1] \neq 0$ because $1 \rightarrow 3 \rightarrow 4 \rightarrow 2$ is a path in $G(A)$.

Sparse LU Factorisation

Corollaries of path theorem

- ▶ If $G(A)$ is connected (there exists a path between any pair of vertices), then A^{-1} is dense.
- ▶ If $G(A)$ is disconnected, i.e. $A = \begin{pmatrix} A_{11} & 0 \\ 0 & A_{22} \end{pmatrix}$, then $A^{-1} = \begin{pmatrix} A_{11}^{-1} & 0 \\ 0 & A_{22}^{-1} \end{pmatrix}$.
- ▶ Inverse of upper/lower triangular matrix is upper/lower triangular.

Sparse LU Factorisation

Fill path

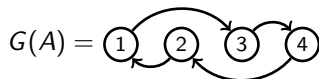
Path $i \rightarrow k_1 \rightarrow \dots \rightarrow k_p \rightarrow j$ in $G(A)$ such that $k_1, \dots, k_p < \min\{i, j\}$.

Fill Path Theorem

$$(L + U)[i, j] \neq 0 \iff \exists \text{ fill path } j \rightarrow i.$$

Example (continued)

$$L + U = \begin{pmatrix} 1 & \bullet & & \\ & 2 & & \bullet \\ \bullet & \bullet & 3 & \\ & \bullet & \bullet & 4 \end{pmatrix}$$



$L[3, 2] \neq 0$ because $2 \rightarrow 1 \rightarrow 3$ is a fill path in $G(A)$.

$L[4, 1] = 0$ because $1 \rightarrow 3 \rightarrow 4$ is not a fill path in $G(A)$.

Sparse LU Factorisation

Lemma

Let $i, j \in \{1, \dots, n\}$ and set $\ell := \{1, \dots, \min\{i, j\} - 1\}$. Then,

$$\begin{aligned}U[i, j] &= A[i, j] - A[i, \ell] A[\ell, \ell]^{-1} A[\ell, j] && \text{for } i \leq j, \\U[j, j] L[i, j] &= A[i, j] - A[i, \ell] A[\ell, \ell]^{-1} A[\ell, j] && \text{for } i \geq j.\end{aligned}$$

Proof. Block LU factorisation with $\bar{r} := \{\min\{i, j\}, \dots, n\}$:

$$\begin{pmatrix} A[\ell, \ell] & A[\ell, \bar{r}] \\ A[\bar{r}, \ell] & A[\bar{r}, \bar{r}] \end{pmatrix} = \begin{pmatrix} I & \\ A[\bar{r}, \ell] A[\ell, \ell]^{-1} & I \end{pmatrix} \begin{pmatrix} A[\ell, \ell] & A[\ell, \bar{r}] \\ A[\bar{r}, \bar{r}] - A[\bar{r}, \ell] A[\ell, \ell]^{-1} A[\ell, \bar{r}] \end{pmatrix}.$$

$$\text{Let } L_1 U_1 = A[\ell, \ell], \quad L_2 U_2 = A[\bar{r}, \bar{r}] - A[\bar{r}, \ell] A[\ell, \ell]^{-1} A[\ell, \bar{r}].$$

Full factorisation is then given by

$$\begin{pmatrix} A[\ell, \ell] & A[\ell, \bar{r}] \\ A[\bar{r}, \ell] & A[\bar{r}, \bar{r}] \end{pmatrix} = \begin{pmatrix} L_1 & \\ A[\bar{r}, \ell] A[\ell, \ell]^{-1} L_1 & L_2 \end{pmatrix} \begin{pmatrix} U_1 & L_1^{-1} A[\ell, \bar{r}] \\ & U_2 \end{pmatrix}.$$

Claim follows by noting that $L[i, j] = L_2[i, j]$ and $U[i, j] = U_2[i, j]$ have the given form.

Sparse LU Factorisation

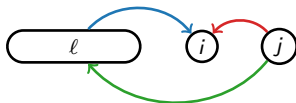
Fill Path Theorem (repeated from previous slide)

$$(L + U)[i, j] \neq 0 \iff \exists \text{ fill path } j \rightarrow i.$$

Proof. Follows immediately from

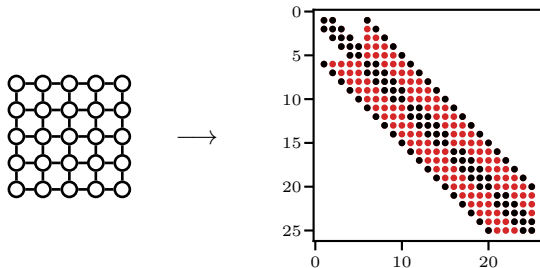
$$U[i, j] = A[i, j] - A[i, \ell] A[\ell, \ell]^{-1} A[\ell, j] \quad \text{for } i \leq j,$$

$$U[j, j] L[i, j] = A[i, j] - A[i, \ell] A[\ell, \ell]^{-1} A[\ell, j] \quad \text{for } i \geq j.$$



Sparse LU Factorisation

Corollary for 2d Laplacian ($n \times n$ grid, $N := n^2$ degrees of freedom)



Memory consumption:

- ▶ $\mathcal{O}(n)$ fill-in per column.
- ▶ Hence, $\mathcal{O}(n^3) = \mathcal{O}(N^{3/2})$ fill-in overall.

Floating-point operations (FLOPs)

- ▶ $\mathcal{O}(n)$ subdiagonal entries to eliminate per column.
- ▶ Each elimination takes $\mathcal{O}(n)$ FLOPs.
- ▶ Hence, $\mathcal{O}(n^4) = \mathcal{O}(N^2)$ FLOPs overall.

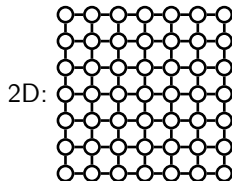
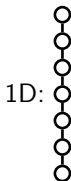
Sparse LU Factorisation

Observations

Amount of fill-in depends on

- ▶ physical dimension of problem, and
- ▶ ordering of rows and columns.

Can we permute the matrix to reduce fill-in?



Sparse LU Factorisation

Algorithm 1 Nested dissection ordering

- 1: Partition the vertices into three sets V_1 , V_2 , V_{sep} such that every path from V_1 to V_2 visits at least one vertex in V_{sep} .
 - 2: Arrange the vertices in the order V_1 , V_2 , V_{sep} , where V_1 and V_2 are ordered recursively according to the nested dissection algorithm.
-

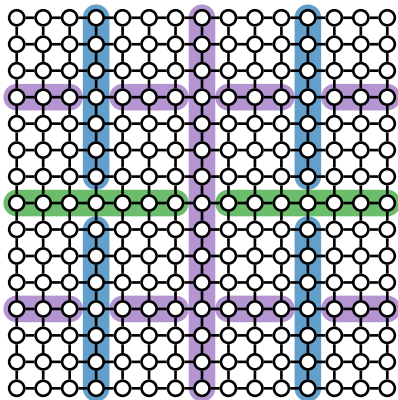
Consequences:

- ▶ No fill-in between V_1 and V_2 !
- ▶ Some fill-in between V_1 / V_2 and V_{sep} (red blocks below).



Sparse LU Factorisation

Separators for 2d mesh



Sparse LU Factorisation

Complexity of LU factorisation with nested dissection ordering

Observations:

- ▶ Diagonal block associated with V_{sep} is dense.
- ▶ Factorising this block costs $\mathcal{O}(|V_{\text{sep}}|^3)$.

The following can be shown for 2D and 3D meshes (see references):

- ▶ The overall runtime of LU factorisation with nested dissection ordering is dominated by factorisation of largest separator.
- ▶ Nested dissection is asymptotically optimal: no other ordering can outperform nested dissection in the big- \mathcal{O} sense.

Sparse LU Factorisation

Complexity of LU factorisation

	Runtime	Memory
$d = 1$	$\mathcal{O}(N)$	$\mathcal{O}(N)$
$d = 2$	$\mathcal{O}(N^{3/2})$	$\mathcal{O}(N \log(N))$
$d = 3$	$\mathcal{O}(N^2)$	$\mathcal{O}(N^{4/3})$

Runtime entries follow immediately from the observation

$$d = 1 \quad \implies \quad |V_{\text{sep}}| = \mathcal{O}(1),$$

$$d = 2 \quad \implies \quad |V_{\text{sep}}| = \mathcal{O}(n) = \mathcal{O}(N^{1/2}),$$

$$d = 3 \quad \implies \quad |V_{\text{sep}}| = \mathcal{O}(n^2) = \mathcal{O}(N^{2/3}).$$

Sparse LU Factorisation

Approximate Minimum Degree (AMD) ordering

- ▶ Finding good separators can be challenging in practice.
- ▶ AMD is another commonly used ordering which is often easier to compute but equally effective.

Sparse algorithms in Julia

Most functions (e.g. `+`, `*`, `\`, `lu()`) are overloaded to automatically exploit sparsity.

Sparse LU Factorisation

Fill-in and pivoting

Recall: LU factorisation of general invertible matrix A requires pivoting to ensure numerical stability.

Bad news: pivoting may undo the effect of fill-in-reducing orderings.

Good news: some classes of matrices provably do not need pivoting.

Matrices which do not require pivoting

- ▶ Column-wise diagonally dominant matrices: $A[j,j] \geq \sum_{i \neq j} |A[i,j]|$.
- ▶ Symmetric positive semi-definite matrices: $A = A^T$ and $v^T A v \geq 0$.

See Homework Sheet 4.

Sparse LU Factorisation

References and further reading

- ▶ T. A. Davis. *Direct Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics (2006),
doi:10.1137/1.9780898718881
- ▶ I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices*. Oxford University Press (2017),
doi:10.1093/acprof:oso/9780198508380.001.0001
- ▶ Fill-in on 2D and 3D meshes:

<https://sites.cs.ucsb.edu/~gilbert/cs219/cs219Spr2013/Notes/fill.pdf>