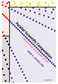
Sequential LU Decomposition
(PSC §2.1–2.2)



Solving a linear system of equations

Find x_0, x_1, x_2 such that

$$\begin{array}{rcrcrcr} x_0 & + & 4x_1 & + & 6x_2 & = & 16 \\ 2x_0 & + & 10x_1 & + & 17x_2 & = & 44 \\ 3x_0 & + & 16x_1 & + & 31x_2 & = & 78 \end{array}$$

In matrix language, solve

$$A\mathbf{x} = \mathbf{b},$$

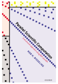
where

$$A = \begin{bmatrix} 1 & 4 & 6 \\ 2 & 10 & 17 \\ 3 & 16 & 31 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 16 \\ 44 \\ 78 \end{bmatrix}$$

Solving linear systems is important

Applications often have as their core a linear system solver.

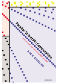
- Building bridges. Finite element models in engineering give rise to linear systems involving a stiffness matrix.
- Designing aircraft. Boundary element methods lead to huge dense linear systems of equations.
- Optimising oil refineries. Linear programming by interior point methods requires solving a sparse linear system (with many zero coefficients) at every step of the computation.



Lower and upper triangular matrices

$$A = \begin{bmatrix} 1 & 4 & 6 \\ 2 & 10 & 17 \\ 3 & 16 & 31 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 4 & 6 \\ 0 & 2 & 5 \\ 0 & 0 & 3 \end{bmatrix} = LU.$$

- L is unit lower triangular if $l_{ii} = 1$ for all i and $l_{ij} = 0$ for all $i < j$.
- U is upper triangular if $u_{ij} = 0$ for all $i > j$.
- LU decomposition is the factorisation of A into $A = LU$, with L unit lower triangular and U upper triangular.



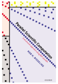
Triangular systems are easier to solve

Let $A = LU$. Then

$$A\mathbf{x} = \mathbf{b} \Longleftrightarrow L(U\mathbf{x}) = \mathbf{b} \Longleftrightarrow L\mathbf{y} = \mathbf{b} \text{ and } U\mathbf{x} = \mathbf{y}.$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 16 \\ 44 \\ 78 \end{bmatrix} \Longrightarrow \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 16 \\ 12 \\ 6 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 4 & 6 \\ 0 & 2 & 5 \\ 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 16 \\ 12 \\ 6 \end{bmatrix} \Longrightarrow \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}.$$



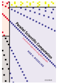
Deriving an algorithm for LU decomposition

Some simple algebra:

$$A = LU \iff a_{ij} = \sum_{r=0}^{n-1} l_{ir} u_{rj} \quad \text{for all } i, j.$$

Assume $i \leq j$. Then:

$$\begin{aligned}
a_{ij} &= \sum_{r=0}^{n-1} l_{ir} u_{rj} = \sum_{r=0}^{i} l_{ir} u_{rj} \quad (\text{because } l_{ir} = 0 \text{ for } r > i) \\
&= \sum_{r=0}^{i-1} l_{ir} u_{rj} + l_{ii} u_{ij} = \sum_{r=0}^{i-1} l_{ir} u_{rj} + u_{ij} \\
&\iff \\
u_{ij} &= a_{ij} - \sum_{r=0}^{i-1} l_{ir} u_{rj}.
\end{aligned}$$



Formulae for computing l_{ij} and u_{ij}

Aim: rewrite the linear system to express l_{ij} and u_{ij} in terms of a_{ij} and previously computed l_{ij} and u_{ij}.

We have obtained

$$u_{ij} = a_{ij} - \sum_{r=0}^{i-1} l_{ir} u_{rj} \quad \text{for } i \leq j.$$

Similarly,

$$l_{ij} = \frac{1}{u_{jj}} \left(a_{ij} - \sum_{r=0}^{j-1} l_{ir} u_{rj} \right) \quad \text{for } i > j.$$

Modifying the matrix A in stages

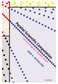
For $0 \leq k \leq n$, define the intermediate matrix $A^{(k)}$ of stage k:

$$a_{ij}^{(k)} = a_{ij} - \sum_{r=0}^{k-1} l_{ir} u_{rj}.$$

Note that $A^{(0)} = A$ and $A^{(n)} = 0$. In this notation,

$$\begin{aligned} u_{ij} &= a_{ij} - \sum_{r=0}^{i-1} l_{ir} u_{rj} \Longleftrightarrow u_{ij} = a_{ij}^{(i)} \\ l_{ij} &= \frac{1}{u_{jj}} \left(a_{ij} - \sum_{r=0}^{j-1} l_{ir} u_{rj} \right) \Longleftrightarrow l_{ij} = \frac{a_{ij}^{(j)}}{u_{jj}} \end{aligned}$$

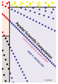
We retrieve values u_{ij} ($i \leq j$) in stage i and l_{ij} ($i > j$) in stage j.



Basic sequential LU decomposition algorithm

input: $A^{(0)}$: $n \times n$ matrix.
output: L : $n \times n$ unit lower triangular matrix,
U : $n \times n$ upper triangular matrix,
such that $LU = A^{(0)}$.

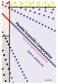
for $k := 0$ **to** $n-1$ **do**
    for $j := k$ **to** $n-1$ **do**
        $u_{kj} := a_{kj}^{(k)}$;



Basic sequential LU decomposition algorithm

input: $A^{(0)}$: $n \times n$ matrix.
output: L : $n \times n$ unit lower triangular matrix,
U : $n \times n$ upper triangular matrix,
such that $LU = A^{(0)}$.

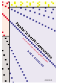
$$
\begin{array}{l}
\textbf{for } k := 0 \textbf{ to } n-1 \textbf{ do} \\
\quad \textbf{for } j := k \textbf{ to } n-1 \textbf{ do} \\
\quad\quad u_{kj} := a_{kj}^{(k)}; \\
\quad \textbf{for } i := k+1 \textbf{ to } n-1 \textbf{ do} \\
\quad\quad l_{ik} := a_{ik}^{(k)} / u_{kk};
\end{array}
$$



Basic sequential LU decomposition algorithm

input: $A^{(0)}$: $n \times n$ matrix.
output: L : $n \times n$ unit lower triangular matrix,
U : $n \times n$ upper triangular matrix,
such that $LU = A^{(0)}$.

for $k := 0$ **to** $n-1$ **do**
    for $j := k$ **to** $n-1$ **do**
        $u_{kj} := a_{kj}^{(k)}$;
    for $i := k+1$ **to** $n-1$ **do**
        $l_{ik} := a_{ik}^{(k)}/u_{kk}$;
    for $i := k+1$ **to** $n-1$ **do**
        for $j := k+1$ **to** $n-1$ **do**
            $a_{ij}^{(k+1)} := a_{ij}^{(k)} - l_{ik}u_{kj}$;

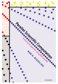


Loop invariant

- A loop invariant is a statement that remains true while a loop is being executed; usually it depends on a changing loop index.
- For LU decomposition, we state

$$\text{Invariant}(k): \quad a_{ij}^{(k)} = a_{ij} - \sum_{r=0}^{k-1} l_{ir} u_{rj} \text{ for all } i, j \geq k.$$

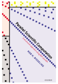
- Giving an invariant at the right place in an algorithm text helps in proving the correctness of the algorithm.
- You can use the `assert` facility in the C-language to check invariants (and other statements).



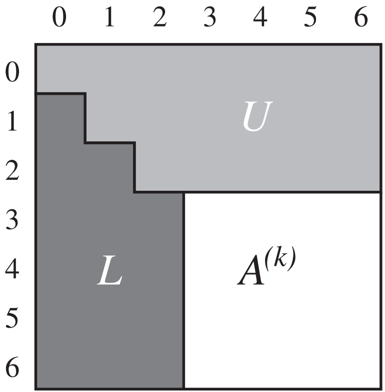
Basic algorithm with loop invariant

input: $A^{(0)}$: $n \times n$ matrix.
output: L : $n \times n$ unit lower triangular matrix,
U : $n \times n$ upper triangular matrix,
such that $LU = A^{(0)}$.

for $k := 0$ **to** $n-1$ **do**
  $\{$ Invariant(k) $\}$
  for $j := k$ **to** $n-1$ **do**
    $u_{kj} := a_{kj}^{(k)}$;
  for $i := k+1$ **to** $n-1$ **do**
    $l_{ik} := a_{ik}^{(k)} / u_{kk}$;
  for $i := k+1$ **to** $n-1$ **do**
    for $j := k+1$ **to** $n-1$ **do**
      $a_{ij}^{(k+1)} := a_{ij}^{(k)} - l_{ik} u_{kj}$;
  $\{$ Invariant$(k+1)$ $\}$



Storing L, U, $A^{(k)}$ in the space of A

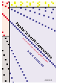


At the start of stage $k = 3$: rows 0, 1, 2 of U and columns 0, 1, 2 of L below the diagonal have already been computed.

Memory-efficient sequential LU decomposition

input: A : $n \times n$ matrix, $A = A^{(0)}$.
output: A : $n \times n$ matrix, $A = L - I_n + U$, with
L : $n \times n$ unit lower triangular matrix,
U : $n \times n$ upper triangular matrix,
I_n : $n \times n$ identity matrix,
such that $LU = A^{(0)}$.

for $k := 0$ **to** $n - 1$ **do**
  for $i := k + 1$ **to** $n - 1$ **do**
    $a_{ik} := a_{ik} / a_{kk}$;
  for $i := k + 1$ **to** $n - 1$ **do**
    for $j := k + 1$ **to** $n - 1$ **do**
      $a_{ij} := a_{ij} - a_{ik} a_{kj}$;

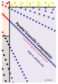


Transformations of A by LU decomposition

$$A = \begin{bmatrix} 1 & 4 & 6 \\ 2 & 10 & 17 \\ 3 & 16 & 31 \end{bmatrix} \xrightarrow{(0)} \begin{bmatrix} 1 & 4 & 6 \\ 2 & 2 & 5 \\ 3 & 4 & 13 \end{bmatrix} \xrightarrow{(1)} \begin{bmatrix} 1 & 4 & 6 \\ 2 & 2 & 5 \\ 3 & 2 & 3 \end{bmatrix}.$$

Hence,

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & 4 & 6 \\ 0 & 2 & 5 \\ 0 & 0 & 3 \end{bmatrix}.$$



Row permutations needed

LU decomposition breaks down immediately in stage 0 for

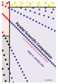
$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix},$$

because we try to divide by 0.

- A solution is to permute the rows suitably.
- Thus, we compute a permuted LU decomposition,

$$PA = LU.$$

- Here, P is a permutation matrix, obtained by permuting the rows of I_n.
- Output of LU decomposition of A: L, U, P.



Permutations and permutation matrices

Let $\sigma : \{0, \dots, n-1\} \rightarrow \{0, \dots, n-1\}$ be a permutation.
We define the permutation matrix P_σ corresponding to σ by

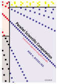
$$(P_\sigma)_{ij} = \begin{cases} 1 & \text{if } i = \sigma(j) \\ 0 & \text{otherwise.} \end{cases}$$

Thus, column j of P_σ is 1 in row $\sigma(j)$, and 0 everywhere else.

Relation between σ and P_σ

Let $\sigma(0) = 1$, $\sigma(1) = 2$, and $\sigma(2) = 0$. Then

$$P_\sigma = \begin{bmatrix} \cdot & \cdot & 1 \\ 1 & \cdot & \cdot \\ \cdot & 1 & \cdot \end{bmatrix}.$$

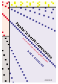


Property of P_σ

Let $\sigma : \{0, \ldots, n-1\} \to \{0, \ldots, n-1\}$ be a permutation.
Let $\mathbf{x}$ be a vector of length n. Then

$$(P_\sigma \mathbf{x})_i = \sum_{j=0}^{n-1} (P_\sigma)_{ij} x_j = x_{\sigma^{-1}(i)},$$

because only the term with $\sigma(j) = i$ is nonzero, i.e., the term $j = \sigma^{-1}(i)$.



Lemma 2.5 Properties of P_σ

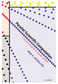
Let $\sigma : \{0, \ldots, n-1\} \to \{0, \ldots, n-1\}$ be a permutation.
Let $\mathbf{x}$ be a vector of length n and A an $n \times n$ matrix. Then

$$(P_\sigma \mathbf{x})_i = x_{\sigma^{-1}(i)}, \quad \text{for } 0 \leq i < n,$$

$$(P_\sigma A)_{ij} = a_{\sigma^{-1}(i),j}, \quad \text{for } 0 \leq i, j < n,$$

$$(P_\sigma A P_\sigma^T)_{ij} = a_{\sigma^{-1}(i),\sigma^{-1}(j)}, \quad \text{for } 0 \leq i, j < n.$$

Proofs: similar to before.



Lemma 2.6 Matrices isomorphic to permutations

Let $\sigma, \tau : \{0, \ldots, n-1\} \to \{0, \ldots, n-1\}$ be permutations. Then

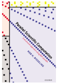
$$P_\tau P_\sigma = P_{\tau\sigma} \text{ and } (P_\sigma)^{-1} = P_{\sigma^{-1}}.$$

Here, $\tau\sigma$ denotes σ followed by τ.

Proof first part:

$$(P_\tau P_\sigma)_{ij} = \sum_{k=0}^{n-1} (P_\tau)_{ik}(P_\sigma)_{kj} = (P_\sigma)_{\tau^{-1}(i),j}$$

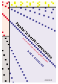
because only one term $k = \tau^{-1}(i)$ is nonzero. By the definition of P_σ, the result is 1 if $\tau^{-1}(i) = \sigma(j)$, i.e., $i = \tau(\sigma(j)) = (\tau\sigma)(j)$, and 0 otherwise. This is the same as for $(P_{\tau\sigma})_{ij}$. $\square$



LU decomposition with row permutations

input: A : $n \times n$ matrix, $A = A^{(0)}$.
output: A : $n \times n$ matrix, $A = L - I_n + U$, with
L : $n \times n$ unit lower triangular matrix,
U : $n \times n$ upper triangular matrix,
π : permutation vector of length n.

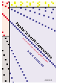
for $i := 0$ **to** $n-1$ **do** $\pi_i := i$;
for $k := 0$ **to** $n-1$ **do**
  $r := \mathrm{argmax}(|a_{ik}| : k \leq i < n)$;
  swap(π_k, π_r);
  for $j := 0$ **to** $n-1$ **do**
    swap(a_{kj}, a_{rj});



LU decomposition with row permutations

input: A : $n \times n$ matrix, $A = A^{(0)}$.
output: A : $n \times n$ matrix, $A = L - I_n + U$, with
L : $n \times n$ unit lower triangular matrix,
U : $n \times n$ upper triangular matrix,
π : permutation vector of length n.

for $i := 0$ **to** $n - 1$ **do** $\pi_i := i$;
for $k := 0$ **to** $n - 1$ **do**
$\quad r := \mathrm{argmax}(|a_{ik}| : k \leq i < n)$;
$\quad \mathrm{swap}(\pi_k, \pi_r)$;
$\quad$ **for** $j := 0$ **to** $n - 1$ **do**
$\qquad \mathrm{swap}(a_{kj}, a_{rj})$;
$\quad$ **for** $i := k + 1$ **to** $n - 1$ **do**
$\qquad a_{ik} := a_{ik} / a_{kk}$;
$\quad$ **for** $i := k + 1$ **to** $n - 1$ **do**
$\qquad$ **for** $j := k + 1$ **to** $n - 1$ **do**
$\qquad\quad a_{ij} := a_{ij} - a_{ik} a_{kj}$;

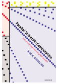


Partial row pivoting

- The pivot element in stage k is the largest element a_{rk} in column k. Everything revolves around it. It is farthest from 0 and division by a_{rk} is most stable.
- The pivot row r is thus determined by

$$|a_{rk}| = \max(|a_{ik}| : k \leq i < n).$$

- r is the argument (or index) of the maximum.
- Full pivoting would take the largest pivot from the whole submatrix $A(k : n-1, k : n-1)$. This gives the best stability, but is more costly. In practice, partial pivoting suffices.



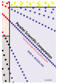
The meaning of π

- The algorithm permutes the matrix by a permutation matrix P_σ. We obtain the LU decomposition $P_\sigma A = LU$.
- The same matrix is applied to the initial vector $\mathbf{e} = (0, 1, 2, \ldots, n-1)^T$. We obtain $\pi = P_\sigma \mathbf{e}$.
- Therefore, by Lemma 2.5,

$$\pi(i) = (P_\sigma \mathbf{e})_i = e_{\sigma^{-1}(i)} = \sigma^{-1}(i).$$

- Thus, $\pi = \sigma^{-1}$ and hence

$$P_{\pi^{-1}} A = LU.$$



Sequential time complexity

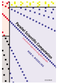
Lemma 2.7:

$$\sum_{k=0}^{n} k = \frac{n(n+1)}{2}, \quad \sum_{k=0}^{n} k^2 = \frac{n(n+1)(2n+1)}{6}.$$

Proof: By induction on n.

The number of flops of the LU decomposition algorithm is

$$\begin{aligned} T_{\mathrm{seq}} &= \sum_{k=0}^{n-1}(2(n-k-1)^2 + n - k - 1) = \sum_{k=0}^{n-1}(2k^2 + k) \\ &= \frac{(n-1)n(2n-1)}{3} + \frac{(n-1)n}{2} \\ &= (n-1)n\left(\frac{2n}{3} + \frac{1}{6}\right) = \frac{2n^3}{3} - \frac{n^2}{2} - \frac{n}{6}. \end{aligned}$$



Summary

- Solving a linear system $A\mathbf{x} = \mathbf{b}$ can best be done by:
  - finding an LU decomposition $PA = LU$;
  - permuting $\mathbf{b}$ into $P\mathbf{b}$;
  - solving the triangular systems $L\mathbf{y} = P\mathbf{b}$ and $U\mathbf{x} = \mathbf{y}$.
- The LU decomposition costs about $2n^3/3$ flops and each triangular system solve about n^2 flops.
- It is always difficult to keep permutations and their inverses apart. In theoretical analysis, it is sometimes easier to work with permutation matrices than with the corresponding permutations.
- We defined the matrix P_σ; its jth column is 1 in row $\sigma(j)$, and 0 everywhere else.
- An important connection between a permutation σ and the matrix P_σ is given by $(P_\sigma \mathbf{x})_i = x_{\sigma^{-1}(i)}$.