Lecture 03 - Solving Linear Systems

May 14, 2025

Outline

1. Solving Linear Systems
    1. Symmetric Positive Definite (SPD) Systems
        1. Constructing the Cholesky Factor
    2. Banded Systems
    3. General Sparse Matrices

Symmetric Positive Definite (SPD) Systems

Theorem 1

If A is SPD, then there exists unique lower triangular G, with strictly positive entries on its diagonal, such that

$$A = GG^T.$$

Symmetric Positive Definite (SPD) Systems

Proof: By results from Lecture 02, write $A = LDL^T$, for some lower triangular L and $D = diag(d_1, \ldots, d_n), d_i > 0$.
Define $D^{\frac{1}{2}}$, one co-ordinate (i), at a time, as follows:

- If the i^{th} diagonal entry of L is positive, then the i^{th} diagonal entry of $D^{\frac{1}{2}}$ is $\sqrt{d_i}$.
- Otherwise, if the i^{th} diagonal entry of L is negative, then the i^{th} diagonal entry of $D^{\frac{1}{2}}$ is $-\sqrt{d_i}$.

Let $G = LD^{\frac{1}{2}}$. Then G is lower triangular, with strictly positive entries on its diagonal.
Further, $GG^T = LD^{\frac{1}{2}}(LD^{\frac{1}{2}})^T = LD^{\frac{1}{2}}D^{\frac{1}{2}}L^T = LDL^T = A$.

Symmetric Positive Definite (SPD) Systems

Explanation for why the Cholesky Factorization is Unique:

- Let $A = GG^T = HH^T$, for some lower triangular H with strictly positive entries on its diagonal.
- First, note that $I = H^{-1}GG^TH^{-T}$:

$$\begin{aligned} HH^T &= A \\ &= GG^T, \text{ so that} \\ H^{-1}(GG^T)H^{-T} &= H^{-1}(HH^T)H^{-T} \\ &= (H^{-1}H)(H^TH^{-T}) \\ &= I \end{aligned}$$

Symmetric Positive Definite (SPD) Systems

- Then we have

$$\begin{aligned} I &= H^{-1}GG^TH^{-T} \\ &= H^{-1}G(H^{-1}G)^T, \text{ and therefore} \qquad (1) \\ H^{-1}G &= (H^{-1}G)^{-T}. \qquad (2) \end{aligned}$$

- The LHS of (2) is a product of lower triangular matrices, hence it is lower triangular.
- The RHS of (2) is upper triangular.
- Let $E = H^{-1}G$.
- Then since E is both upper and lower triangular, therefore E is diagonal. In particular $E^T = E$.
- Therefore (1) implies that $E^2 = I$, in other words E is diagonal, with ± 1 entries on its diagonal.
- Because $G = HE$, therefore the two factorizations can differ only by the signs of their columns.
- But since the diagonal entries of both G and H must be strictly positive, therefore they must be equal.

□

Symmetric Positive Definite (SPD) Systems

The factorization $A = GG^T$ is called the **Cholesky factorization**.
The matrix G is referred to as the **Cholesky factor**.

Q & A

1. Will we need to reproduce proofs on Quizzes / Assignments / Exams?
   A: Not on Quizzes / Assignments, since they will be open book. I may ask for short, new proofs, on Assignments. It is also possible that I will ask for proofs on Exams. My idea about Quizzes, so far, is to have you work out some examples of the results we are proving in class.

Symmetric Positive Definite (SPD) Systems

Constructing the Cholesky Factor We now discuss how to construct the Cholesky factor. Lecture 23 of Trefethen and Bau or Section 4.2.5 of Golub and Van Loan are good supplemental reads. First, view A as

$$A = \begin{bmatrix} \alpha & v^T \\ v & B \end{bmatrix},$$

where $\alpha = a_{11} \in \mathbb{R}$, $v = a_{2:n,1} \in \mathbb{R}^{n-1}$, and $B = a_{2:n,2:n} \in \mathbb{R}^{(n-1)\times(n-1)}$.

Symmetric Positive Definite (SPD) Systems

The colon notation follows from MATLAB.

- For a matrix A, $a_{i,:}$ and $a_{:,j}$ denote the i-th row and the j-th column, respectively.
- Moreover, $a_{i:j,p:q}$ denotes the submatrix with rows from i to j and columns from p to q.
- For example, $a_{2:n,1}$ corresponds to entries in the 1^{st} column of A, from the 2^{nd} to the n^{th} row.

Symmetric Positive Definite (SPD) Systems

Cholesky factorization can intuitively be thought of as applying Gaussian elimination in a "symmetric way". The goal of Gaussian elimination was to zero-out the column below by subtracting multiples of the current row. The "work-in-progress" LU-factorization of A after the first step of Gaussian elimination can be viewed as

$$A = \begin{bmatrix} 1 & 0 \\ \frac{v}{\alpha} & I \end{bmatrix} \begin{bmatrix} \alpha & v^T \\ 0 & B - \frac{vv^T}{\alpha} \end{bmatrix},$$

where the first column of L becomes

$$\begin{bmatrix} 1 \\ \frac{v}{\alpha} \end{bmatrix}.$$

Symmetric Positive Definite (SPD) Systems

Reminder/Explanation about LU-factorization:

1. To eliminate the entries of v below α in column 1, the multipliers are $-\frac{v}{\alpha}$.
2. Thus, per our LU-factorization procedure, we record the negatives of these multipliers, namely $\frac{v}{\alpha}$, in column 1 of our L matrix.
3. Why the $B - \frac{vv^T}{\alpha}$ block is correct:
   1. As above, the multipliers $-\frac{v}{\alpha}$ are required by the entries below the pivot in column 1.
   2. Because of symmetry, the entries that get applied to the lower rows are v^T.

   This explains where the term $-\frac{vv^T}{\alpha}$ comes from.
4. The provided multiplication creates a $\frac{vv^T}{\alpha}$ term, which cancels with $-\frac{vv^T}{\alpha}$, to leave the required B in the bottom right of the product matrix.

Symmetric Positive Definite (SPD) Systems

However, applying just Gaussian elimination to get an LU factorization does not take advantage of symmetry. Cholesky factorization therefore aims to zero out the corresponding row also to remain symmetric. The first stage of Cholesky factorization is

$$A = \begin{bmatrix} \sqrt{\alpha} & 0 \\ \frac{v}{\sqrt{\alpha}} & I \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & B - \frac{vv^T}{\alpha} \end{bmatrix} \begin{bmatrix} \sqrt{\alpha} & \frac{v^T}{\sqrt{\alpha}} \\ 0 & I \end{bmatrix},$$

which gives the first column of G as

$$\begin{bmatrix} \sqrt{\alpha} \\ \frac{v}{\sqrt{\alpha}} \end{bmatrix}.$$

Symmetric Positive Definite (SPD) Systems

Brief Explanation of the Change From LU to Cholesky

1. It is an exercise to verify that this "work-in-progress" factorization of A is correct.
   1. The job of the LH factor is to restore the entries below the pivot.
   2. The job of the RH factor is to restore the entries to the right of the pivot.
2. In LU, we produce the L-factor on the left, and the U factor by modifying the original A, on the right.
3. In Cholesky, we will produce by the end,

$$G \underbrace{\qquad\qquad}_{\text{where } A \text{ was, originally}} G^T$$

Symmetric Positive Definite (SPD) Systems

This first step is derived by considering that the final form must be

$$A = GG^T = \begin{bmatrix} g_{11} & 0 \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} g_{11} & G_{21}^T \\ 0 & G_{22}^T \end{bmatrix} = \begin{bmatrix} g_{11}^2 & g_{11} G_{21}^T \\ g_{11} G_{21} & G_{21} G_{21}^T + G_{22} G_{22}^T \end{bmatrix}.$$

Therefore,

$$A = \begin{bmatrix} \alpha & v^T \\ v & B \end{bmatrix} = \begin{bmatrix} g_{11}^2 & g_{11} G_{21}^T \\ g_{11} G_{21} & G_{22} G_{22}^T + G_{21} G_{21}^T \end{bmatrix},$$

Symmetric Positive Definite (SPD) Systems

implies

$$\begin{aligned} \alpha &= (g_{11})^2 \\ \Rightarrow g_{11} &= \sqrt{\alpha}, \\ v &= g_{11} G_{21} \\ \Rightarrow G_{21} &= \frac{v}{g_{11}} \\ &= \frac{v}{\sqrt{\alpha}}. \end{aligned}$$

This provides the ingredients (namely α and v) needed to compute the matrix to be processed at the next step (namely $B - \frac{vv^T}{\alpha}$).

Symmetric Positive Definite (SPD) Systems

The Cholesky factorization algorithm then works recursively on the lower block $B - \frac{vv^T}{\alpha}$ since it is also SPD. To see that $B - \frac{vv^T}{\alpha}$ is SPD consider multiplying by the full rank matrix

$$X = \begin{bmatrix} 1 & -\frac{v^T}{\alpha} \\ 0 & I \end{bmatrix}.$$

We have that

$$X^T A X = \begin{bmatrix} \alpha & 0 \\ 0 & B - \frac{vv^T}{\alpha} \end{bmatrix},$$

hence by results from Lecture 02, the principal submatrix $B - \frac{vv^T}{\alpha}$ is PD. It is also symmetric since $X^T A X = (X^T A X)^T$ and A is symmetric.

Symmetric Positive Definite (SPD) Systems

So we can Cholesky factor $B - \frac{vv^T}{\alpha}$ as $B - \frac{vv^T}{\alpha} = G_1 G_1^T$. The recursion continues eliminating one row/column at a time. The Cholesky factor of A itself will have the form

$$G = \begin{bmatrix} \sqrt{\alpha} & 0 \\ \frac{v}{\sqrt{\alpha}} & G_1 \end{bmatrix}.$$

Symmetric Positive Definite (SPD) Systems

Cholesky Example:

- So that we will know what our goal is, we first choose a G. Then we compute our starting point, namely the matrix A, via $A = GG^T$.
- With the help of a student, we selected

$$
\begin{aligned}
G &= \begin{bmatrix} 2 & 0 & 0 \\ 3 & 3 & 0 \\ -4 & -6 & 5 \end{bmatrix}, \text{ so that} \\
A &= GG^T \\
&= \begin{bmatrix} 2 & 0 & 0 \\ 3 & 3 & 0 \\ -4 & -6 & 5 \end{bmatrix} \begin{bmatrix} 2 & 3 & -4 \\ 0 & 3 & -6 \\ 0 & 0 & 5 \end{bmatrix} \\
&= \begin{bmatrix} 4 & 6 & -8 \\ 6 & 18 & -30 \\ -8 & -30 & 77 \end{bmatrix}.
\end{aligned}
$$

Symmetric Positive Definite (SPD) Systems

- A is clearly symmetric.
- To make absolutely certain that A has a Cholesky factorization, we verify that A is also positive definite. It is an exercise to verify that row reducing A yields $\begin{bmatrix} 4 & 6 & -8 \\ 0 & 9 & -18 \\ 0 & 0 & 25 \end{bmatrix}$. From this matrix, we can see that the pivots of A are $4, 9$ and 25. They are all positive, and hence A is positive definite.

Symmetric Positive Definite (SPD) Systems

We carry out the algorithm to compute the Cholesky factor G.

1. Writing $\begin{bmatrix} 4 & 6 & -8 \\ 6 & 18 & -30 \\ -8 & -30 & 77 \end{bmatrix} = \begin{bmatrix} \alpha & v^T \\ v & B \end{bmatrix}$, we have

$$\begin{aligned} \alpha &= 4 \\ v &= \begin{bmatrix} 6 \\ -8 \end{bmatrix} \\ B &= \begin{bmatrix} 18 & -30 \\ -30 & 77 \end{bmatrix}. \end{aligned}$$

Symmetric Positive Definite (SPD) Systems

and therefore we obtain

$$
\begin{aligned}
g_{11} &= \sqrt{\alpha} \\
&= \sqrt{4} \\
&= 2 \\
G_{21} &= \frac{v}{g_{11}} \\
&= \frac{\begin{bmatrix} 6 \\ -8 \end{bmatrix}}{2} \\
&= \begin{bmatrix} 3 \\ -4 \end{bmatrix}, \text{ so that}
\end{aligned}
$$

Symmetric Positive Definite (SPD) Systems

$$
\begin{aligned}
B - \frac{vv^T}{\alpha} &= \begin{bmatrix} 18 & -30 \\ -30 & 77 \end{bmatrix} - \frac{\begin{bmatrix} 6 \\ -8 \end{bmatrix} \begin{bmatrix} 6 & -8 \end{bmatrix}}{4} \\
&= \begin{bmatrix} 18 & -30 \\ -30 & 77 \end{bmatrix} - \frac{\begin{bmatrix} 36 & -48 \\ -48 & 64 \end{bmatrix}}{4} \\
&= \begin{bmatrix} 18 & -30 \\ -30 & 77 \end{bmatrix} - \begin{bmatrix} 9 & -12 \\ -12 & 16 \end{bmatrix} \\
&= \begin{bmatrix} 9 & -18 \\ -18 & 61 \end{bmatrix}
\end{aligned}
$$

Symmetric Positive Definite (SPD) Systems

2. Writing $\begin{bmatrix} 9 & -18 \\ -18 & 61 \end{bmatrix} = \begin{bmatrix} \alpha & v^T \\ v & B \end{bmatrix}$, we have

$$\begin{aligned} \alpha &= 9 \\ v &= \begin{bmatrix} -18 \end{bmatrix} \\ B &= \begin{bmatrix} 61 \end{bmatrix}. \end{aligned}$$

Symmetric Positive Definite (SPD) Systems

and therefore we obtain

$$
\begin{aligned}
g_{11} &= \sqrt{\alpha} \\
&= \sqrt{9} \\
&= 3 \\
G_{21} &= \frac{v}{g_{11}} \\
&= \frac{\left[-18\right]}{3} \\
&= \left[-6\right], \text{ so that} \\
B - \frac{vv^T}{\alpha} &= \left[61\right] - \frac{\left[-18\right]\left[-18\right]}{9} \\
&= \left[61\right] - \left[36\right] \\
&= \left[25\right]
\end{aligned}
$$

Symmetric Positive Definite (SPD) Systems

3. Writing $\begin{bmatrix} 25 \end{bmatrix} = \begin{bmatrix} \alpha & v^T \\ v & B \end{bmatrix}$, we have

$$\alpha = 25$$

and therefore we obtain

$$\begin{aligned} g_{11} &= \sqrt{\alpha} \\ &= \sqrt{25} \\ &= 5, \end{aligned}$$

at which point, we are finished.

We have now recovered all the diagonal and subdiagonal entries from the original choice of G.

Symmetric Positive Definite (SPD) Systems

Cost of Cholesky Factorization Algorithm 1 gives pseudocode for the in-place Cholesky factorization. This implementation exploits symmetry by working only on sub-diagonal entries. In the end, the lower triangle is the Cholesky factor G.

Symmetric Positive Definite (SPD) Systems

Algorithm 1 : Cholesky Factorization

for $k = 1, \ldots, n$ ▷ iterate down rows
  $a_{kk} = \sqrt{a_{kk}}$ ▷ factor diagonal element $(\sqrt{\alpha})$
  for $i = k + 1, \ldots, n$ ▷ go over rows
    $a_{ik} = a_{ik}/a_{kk}$ ▷ update current column entries below diagonal $(v/\sqrt{\alpha})$
  end for
  for $j = k + 1, \ldots, n$ ▷ go over columns
    for $i = j, \ldots, n$ ▷ go over rows
      $a_{ij} = a_{ij} - a_{ik} * a_{jk}$ ▷ update lower right block $B - vv^T/\alpha$
    end for
  end for
end for

Symmetric Positive Definite (SPD) Systems

Note that Algorithm 1 assumes that the diagonal entries a_{kk} are non-zero. Problems arise when these entries are zero or close to zero. We will address this issue through **pivoting**, when we discuss the stability of factorizations.

The cost of Cholesky factorization can be estimated by considering just the inner most loop. For that loop there is one subtraction and one multiplication. So the FLOP count is

$$\sum_{k=1}^{n} \sum_{j=k+1}^{n} \sum_{i=j}^{n} 2 = \frac{n^3}{3} + O\left(n^2\right),$$

which is calculated as displayed in the Lecture Notes. As promised Cholesky factorization is half that of LU factorization, which had a cost of $\frac{2n^3}{3} + O\left(n^2\right)$ FLOPs.

Symmetric Positive Definite (SPD) Systems

Detailed Computation: See Lecture Notes.

Banded Systems

Banded matrices have nonzero entries only in "bands" adjacent to the main diagonal. Some example banded matrices are (empty entries are zero)

$$\begin{bmatrix} 9 & 5 & & \\ & 3 & 8 & \\ & & 2 & 7 \\ & & & 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 5 & 6 & & & & \\ 7 & 7 & 1 & & & \\ 4 & 2 & 9 & 2 & & \\ & 7 & 5 & 3 & 8 & \\ & & 3 & 8 & 2 & 7 \\ & & & 4 & 7 & 1 \end{bmatrix}. \tag{3}$$

Banded Systems

Definition 1.1

The matrix $A = [a_{ij}]$ *has*

1. **upper bandwidth**, q, *if* $a_{ij} = 0$ *for* $j > i + q$, *and*
2. **lower bandwidth**, p, *if* $a_{ij} = 0$ *for* $i > j + p$.

The general form of a banded matrix given in Definition 1.1 is

$$\begin{bmatrix} \times & \cdots & \times & & & & \\ \vdots & \ddots & & \ddots & & & \\ \times & & & & & & \\ & \ddots & & \ddots & & \ddots & \\ & & & & & & \times \\ & & & \ddots & & \ddots & \vdots \\ & & & & \times & \cdots & \times \end{bmatrix}.$$

Banded Systems

In the examples in (3) the first matrix has $q = 1, p = 0$ and the second has $q = 1, p = 2$. Exercise: how might you store these banded matrices efficiently?

Q & A

1. Can there be a piece, inside the band (i.e. parallel to the band), with all 0 entries?
   A: Yes! Definition 3 says nothing about what happens within the band. In the most extreme case, the zero matrix trivially satisfies the definition of a banded matrix!
2. Do upper lower triangular matrices satisfy Definition 3?
   A: Yes!
   1. An upper triangular matrix satisfies $p = 0$.
   2. A lower triangular matrix satisfies $q = 0$.

Banded Systems

Factoring Banded Matrices If A is banded then so are the factorizations LU, LDM^T, GG^T.

Theorem 1

Let $A = LU$. If A has upper bandwidth q and lower bandwidth p, then U has upper bandwidth q and L has lower bandwidth p.

Banded Systems

$$A = \begin{bmatrix} \times & \cdots & \times & & & \\ \vdots & \ddots & & \ddots & & \\ \times & & & & & \\ & \ddots & & \ddots & & \ddots \\ & & & & & & \times \\ & & & \ddots & & \ddots & \vdots \\ & & & & \times & \cdots & \times \end{bmatrix}$$

$$= \begin{bmatrix} \times & & & & & \\ \vdots & \ddots & & & & \\ \times & & & & & \\ & \ddots & \ddots & & & \\ & & \ddots & \ddots & & \\ & & & \times & \cdots & \times \end{bmatrix} \begin{bmatrix} \times & \cdots & \times & & & \\ & \ddots & & \ddots & & \\ & & & & & \\ & & \ddots & & \ddots & \\ & & & \ddots & & \times \\ & & & & \ddots & \vdots \\ & & & & & \times \end{bmatrix}.$$

L U

Banded Systems

A special case of banded matrices is **tridiagonal** matrices, which have $p = q = 1$. One can show that the flop count for LU factorization of tridiagonal matrices is $O(n)$. Table 1 gives other examples of important matrices we will see in this course.

Matrix Type	Lower Bandwidth p	Upper Bandwidth q
Diagonal	0	0
Upper Triangular	0	$n-1$
Lower Triangular	$m-1$	0
Tridiagonal	1	1
Upper Bidiagonal	0	1
Lower Bidiagonal	1	0
Upper Hessenberg	1	$n-1$
Lower Hessenberg	$m-1$	1

Table: Common types of matrices $A \in \mathbb{R}^{m \times n}$ in this course and their bandwidths.

Banded Systems

Cost of Banded LU Factorization

Algorithm 2 : Banded LU Factorization

for $k = 1, \ldots, n-1$ ▷ iterate over rows
  for $i = k+1, \ldots, \min(k+p, n)$ ▷ go over rows within band
    $a_{ik} = a_{ik}/a_{kk}$ ▷ determine multiplicative factors
  end for
  for $i = k+1, \ldots, \min(k+p, n)$ ▷ go over rows within band
    for $j = k+1, \ldots, \min(k+q, n)$ ▷ go over columns within band
      $a_{ij} = a_{ij} - a_{ik} * a_{kj}$ ▷ subtract scaled row data only in non-zero bands
    end for
  end for
end for

Banded Systems

- Algorithm 35 gives the banded version of LU factorization.
- We assume diagonal entries $a_{kk} \neq 0$ for now in Algorithm 35.
- Banded LU factorization is most beneficial when there are many zeros in A.
- In other words, when the upper/lower bandwidths (q/p) are small relative to the size of the matrix (n).
- If $n \gg p$ and $n \gg q$, then banded LU is $\sim 2npq$ flops.
- This is much faster than naïve implementation of LU factorization $\sim \frac{2n^3}{3}$ flops that would operate on zero entries.
- For example, with $n = 300, p = 2, q = 2$, basic LU takes $\sim 18,000,000$ flops but banded LU takes only ~ 2400 flops.

Banded Systems

Exercises:

1. Verify flop count of banded LU factorization (leading order term).
2. Work out **efficient** algorithms for banded forward/backward triangular solves. That is, modify the standard forward/backward solve algorithms to avoid touching entries you **know** are always zero (based on bandwidth).

General Sparse Matrices

- Patterns other than just simple bands are also common.
- General **sparse matrices** (i.e. matrices having few non-zero entries) contain mostly zero entries, but non-zeros can occur on more than the diagonal bands.
- For many problems the (max) number of non-zeros per row is constant, i.e., the total number of non-zeros is $O(n)$.
- So we still only want to store the non-zero entries.
- Various storage formats (data structures) exist for sparse matrices.
- We will not be coding our own in this course, but it is useful to be aware of them.
- The simplest approach is to have a vector of one (i, j, value) triplet per non-zero, but this is inefficient.

General Sparse Matrices

A more common storage structure is the Compressed Row Storage (CRS) (or Compressed Sparse Row (CSR)):

- array of non-zero entries ("val") with length = number of non-zeros (nnz),
- array of column indices ("colInd") with length = nnz, and
- array of indices where each row starts ("rowPtr") with length = number of rows.

General Sparse Matrices

For example, consider the CSR structure for the following matrix

$$\begin{bmatrix} 2 & & 5 & \\ & 3 & & \\ & 6 & -3 & \\ & & 10 & 2 \end{bmatrix} \Rightarrow$$

$$\text{val} = [2, 5, 3, 6, -3, 10, 2],$$

$$\text{colInd} = [1, 3, 2, 2, 3, 3, 4],$$

$$\text{rowPtr} = [1, 3, 4, 6].$$

(We could include 0 for an empty row.)

Remark: We have **not** made a rigourous definition of a sparse matrix.

General Sparse Matrices

Factorization

- For LU factorization the main cost is the row subtraction step:

$$a_{ij} = a_{ij} - a_{ik} a_{kj} / a_{kk}.$$

- Since most entries are zero, our algorithms should skip operating on them.
- However, an **important point** to realize is that even if A is sparse, its factorization **may not** be!
- This happens because row subtractions can turn zeros into non-zero entries, which is referred to as "fill-in".
- A classic example is the "arrow matrix", which has fully dense (triangular) L and U factors

$$A = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & & & \\ \times & & \times & & \\ \times & & & \times & \\ \times & & & & \times \end{bmatrix}$$

General Sparse Matrices

Exercises:

1. Can you see why the L and U are dense (in corresponding triangles)?
2. Therefore, what is the storage cost of the factors and what is the complexity of the factorization?

General Sparse Matrices

The key intuition for general sparse matrices is that we must reorder the system of equations. A matrix **reordering** permutes rows/columns to yield a matrix whose LU factorization suffers no fill-in (i.e., no new non-zeros).

General Sparse Matrices

With the arrow matrix for example we can solve the same system, but reorder the equations so that

$$
\begin{aligned}
A &= \begin{bmatrix} \times & & & & \times \\ & \times & & & \times \\ & & \times & & \times \\ & & & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} \\
&= \begin{bmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ \times & \times & \times & \times & 1 \end{bmatrix} \begin{bmatrix} \times & & & & \times \\ & \times & & & \times \\ & & \times & & \times \\ & & & \times & \times \\ & & & & \times \end{bmatrix} \\
&= LU.
\end{aligned}
$$

Matrix reorderings will be discussed in more detail later.