

## Pivoting

**Reading:** GV96 Section 3.4, Stew98 Chapter 3: 1.3

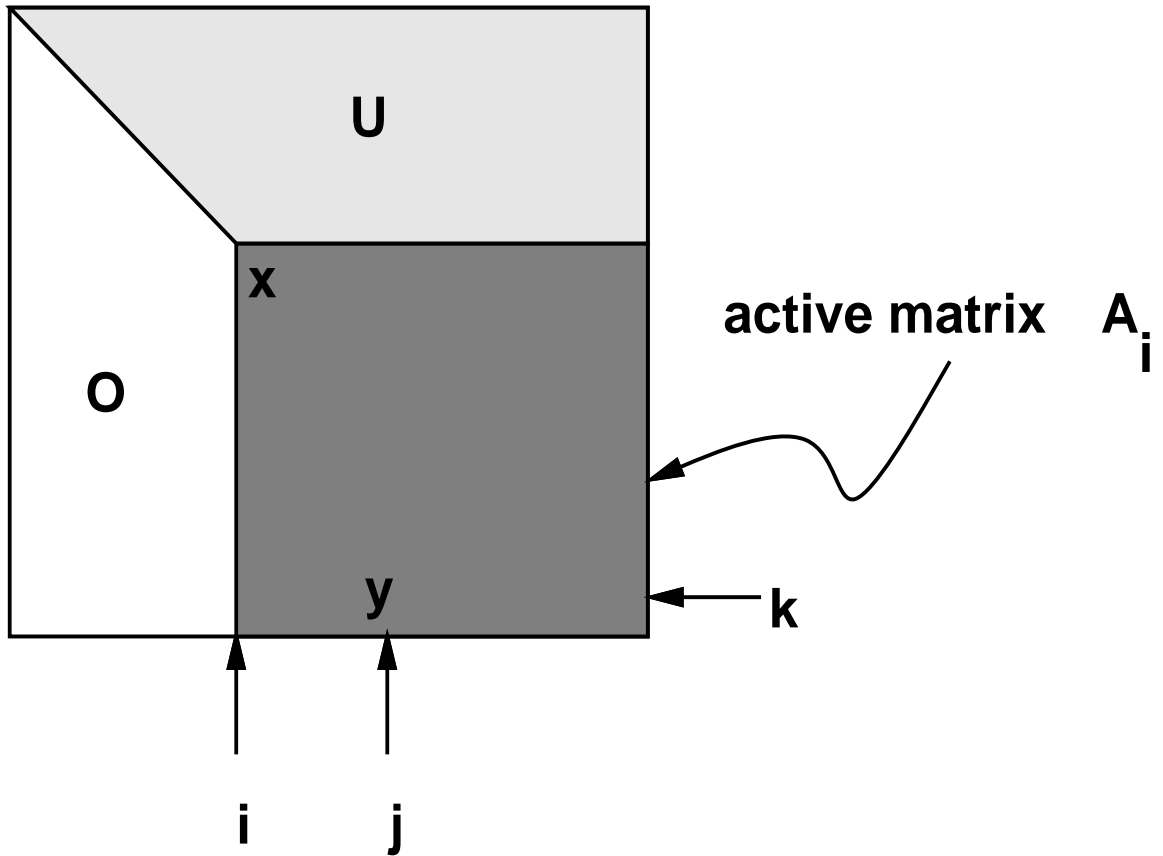
In the previous discussions we have assumed that the  $LU$  factorization of  $A$  existed and the various versions could compute it in a stable manner. The fact that  $A$  is nonsingular is not enough to justify this assumption.

Example: The first step fails for the following nonsingular matrix.

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$$

If, however, the rows are interchanged (or the columns) we can proceed.

The interchanging of rows and/or columns in order to find a nonzero pivot element is called *pivoting*. It is needed to guarantee existence and stability of the factorization.



**y is element of largest magnitude**

**and is in position (k,j).**

**interchange columns i and j  
and rows i and k to interchange  
old pivot element x with preferred  
pivot y.**

## Complete Pivoting

Before eliminating the subdiagonal elements in the  $i$ -th column of the transformed matrix (the first column of the active matrix) find the element with the largest magnitude and permute it via row and column interchanges to the  $(i, i)$  diagonal element position.

- Unconditionally stable
- Requires examining the entire active portion of the matrix.
- This typically has been viewed as too expensive since it was usually implemented as an extra pass through the matrix. This can be mitigated if the rank-1 primitive is extended
- The set of candidate pivot elements is the entire active matrix, therefore the immediate update forms **must** be used.

Row interchanges can be represented by an elementary permutation matrix.

**Definition:** An elementary permutation matrix  $P$  which interchanges rows  $i$  and  $j$  of the matrix to which it is applied by premultiplication, i.e.,  $PA$ , is the identity matrix  $I$  with rows  $i$  and  $j$  interchanged.

**EXAMPLE:**

To interchange rows 1 and 3 of a matrix  $A$  of order 3 compute  $PA$  where

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

- column interchanges is done by postmultiplication by an identity with two columns interchanged, call it  $Q$ .
- $P = P^{-1}$  and  $Q = Q^{-1}$  follow immediately from the definition.
- In general,  $P$  or  $Q$  can be specified by two integers, however, in the LU factorization one of the integers is always  $i$  if the permutation is applied on the  $i$ -th step of the algorithm.
- $Px$  and  $Qx$  simply interchange two components of the vector, the implied  $i$ -th position and the store index  $k_i$ .
- The product of a series of permutations can be stored as a vector with  $n$  entries that are initialized to  $j$  in the  $j$ -th component and to which all of the permutations are applied. The product can then be easily applied to a vector by simple indirect array addressing.

We have

$$U = (M_{n-1}^{-1}P_{n-1} \cdots M_1^{-1}P_1)A(Q_1 \cdots Q_{n-1})$$

$$U = TAQ$$

$$T = M_{n-1}^{-1}P_{n-1} \cdots M_1^{-1}P_1$$

$$Q = Q_1 \cdots Q_{n-1}$$

To solve  $Ax = b$  given  $T$ ,  $Q$ , and  $U$  we have

$$\begin{aligned} Ax &= b \\ TA(QQ^{-1})x &= Tb \\ (TAQ)(Q^{-1}x) &= (Tb) \\ U\tilde{x} &= \tilde{b} \end{aligned}$$

This yields the algorithm:

- Compute  $\tilde{b} = Tb = M_{n-1}^{-1}P_{n-1} \cdots M_1^{-1}P_1b$  via a modified forward substitution ( $T$  and  $T^{-1}$  are not lower triangular)
- Solve  $U\tilde{x} = \tilde{b}$  via backward substitution (upper triangular solve)
- Compute  $x = Q\tilde{x} = Q_1 \cdots Q_{n-1}\tilde{x}$  by applying permutations.

## Partial Pivoting

Rather than searching the entire active matrix search only its first column for the element of maximum magnitude. Permute the chosen element to the  $(i, i)$  diagonal element position via a row interchange. (A similar strategy can be defined by searching the first column of the active matrix.)

- less stable than complete pivoting and may be unstable but, in practice, satisfactory stability is achieved.
- only requires the production of the first column (row) of the active matrix so delayed updates can be used.
- only requires row (column) interchanges.

We have

$$U = (M_{n-1}^{-1}P_{n-1} \cdots M_1^{-1}P_1)A$$

$$U = TA$$

$$T = M_{n-1}^{-1}P_{n-1} \cdots M_1^{-1}P_1$$

To solve  $Ax = b$  given  $T$  and  $U$  we have

$$Ax = b$$

$$TAx = Tb$$

$$Ux = \tilde{b}$$

This yields the algorithm:

- Compute  $\tilde{b} = Tb = M_{n-1}^{-1}P_{n-1} \cdots M_1^{-1}P_1b$  via a modified forward substitution ( $T$  and  $T^{-1}$  are not lower triangular)
- Solve  $Ux = \tilde{b}$  via backward substitution (upper triangular solve)

**Have we lost the LU factorization since  $T$  is not lower triangular?**

**LEMMA:**

If  $A^{-1}$  exists then there exists a product of elementary row permutation matrices  $P = P_{n-1} \dots P_1$ , a unit lower triangular matrix  $L$ , and an upper triangular matrix  $U$  such that

$$PA = LU.$$

In other words, partial pivoting is equivalent to computing the  $LU$  factorization of a row permuted version of the matrix  $A$ .

To solve  $Ax = b$  given  $P$ ,  $L$  and  $U$  we have

$$\begin{aligned} Ax &= b \\ PAx &= Pb \\ LUx &= \tilde{b} \\ Ly &= \tilde{b} \end{aligned}$$

This yields the algorithm:

- Compute  $\tilde{b} = Pb$
- Solve  $Ly = \tilde{b}$  via forward substitution (lower triangular solve)
- Solve  $Ux = y$  via backward substitution (upper triangular solve)

**How do we get  $L$  and  $U$  in practice?**

**LEMMA:**

$$\begin{aligned}L &= \tilde{M}_1 \tilde{M}_2 \cdots \tilde{M}_{n-1} \\ \tilde{M}_i^{-1} &= P_{n-1} \cdots P_{i+1} M_i^{-1} P_{i+1} \cdots P_{n-1}\end{aligned}$$

and  $M_i^{-1}$  is the Gauss transform from the  $i$ -th step of the factorization.

To see that  $\tilde{M}_i^{-1}$  is still an elementary unit lower triangular matrix (and therefore  $L$  is unit lower triangular) note that if  $j > i$

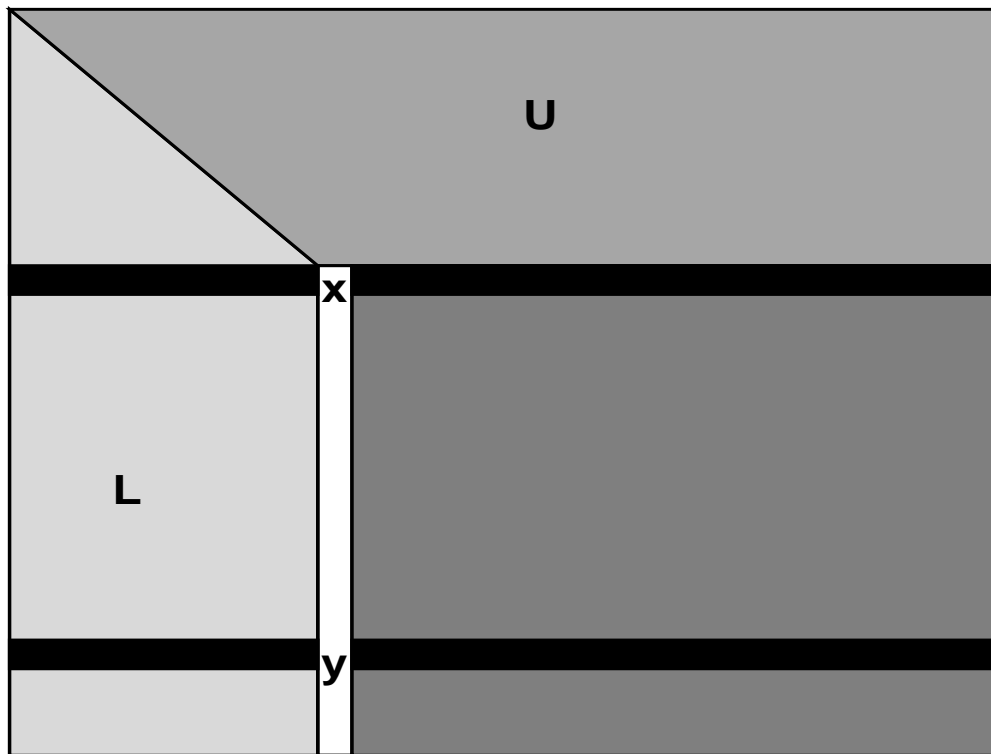
$$\begin{aligned} P_j \hat{M}_i^{-1} P_j &= P_j (I - l_i e_i^T) P_j \\ &= I - P_j l_i e_i^T P_j \\ &= I - \tilde{l}_i e_i^T \end{aligned}$$

where  $\hat{M}_i$  is an elementary unit triangular matrix and  $\tilde{l}_i = P_j l_i$ .

This follows from  $j > i$  since the two rows exchanged in the pivoting process that define  $P_j$  have indices larger than  $i$  and therefore row  $i$  of  $P_j = e_i^T$ . and  $\tilde{l}_i$  is just  $l_i$  with two components interchanged and with the same nonzero sparsity structure.

It follows that for BLAS2-based algorithms we need to apply the permutation to all previous  $M_i^{-1}$  and the rows of the active matrix. This can be implemented as interchanging the *entire* row  $i$  with the *entire* row  $j$ .

## Partial pivoting in a BLAS2-based LU



**Interchange the all elements in the old and new pivot rows.**

**The elements interchanged in the  $L$  portion updates the  $M_i$**

**The interchange in the active portion updates  $A$ .**

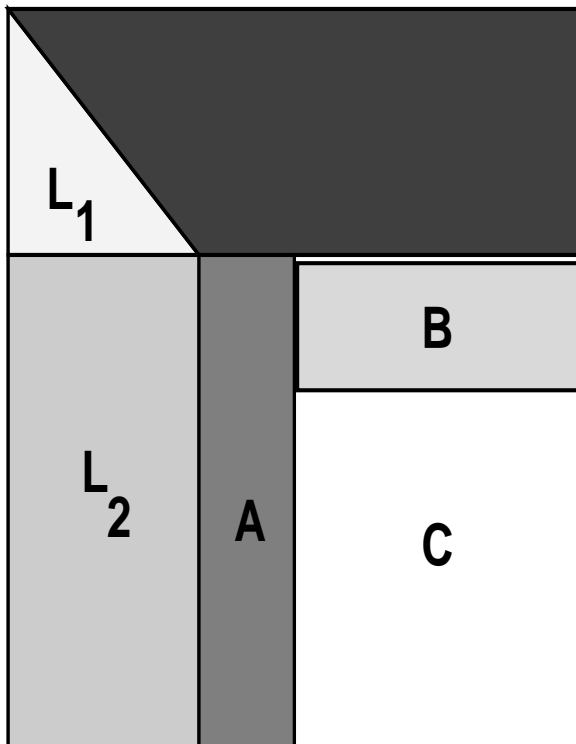
**At the end of the factorization, the LU stored in the array will be that of  $PA$ .**

For BLAS3-based algorithms we have a similar procedure. If  $k = n/\omega$  then

$$M_{k-1}^{-1}P_{k-1} \dots M_1^{-1}P_1A = U$$

where  $M_i^{-1}$  is the rank- $\omega$  update on the  $i$ -th block elimination step and  $P_i$  is the accumulated permutation matrix from the  $LU$  factorization of the rectangular matrix defined by the first  $\omega$  columns of the active matrix. This accumulation is done as described above for the BLAS2-based methods.

## Basic step of BLAS3-based LU with partial pivoting



1. compute  $LU = A$  w/ pivoting to get  $P_i$
2. apply  $P_i$  to  $\begin{bmatrix} B \\ C \end{bmatrix}$
3. apply  $P_i$  to  $L_2$
4. update  $B$  using  $L$  from step (1)
5. update  $C$  via a rank- $w$  update

Note that, due to pivoting, the BLAS2-based factorization of the first  $\omega$  columns of the active matrix is no longer decomposable into the  $LU$  factorization of an  $\omega \times \omega$  matrix followed by a triangular solve with multiple right-hand side vectors.

This portion of the algorithm therefore can become a bottleneck. A two level version would use a block  $LU$  factorization with block size  $\theta < \omega$  to mitigate this bottleneck.