Set 2: Solving Linear Systems

Kyle A. Gallivan

Department of Mathematics

Florida State University

Foundations of Computational Math 1

Fall 2012

The Problem

- $A \in \mathbb{C}^{n\times n}$, $x \in \mathbb{C}^n$ and $b \in \mathbb{C}^n$
- Given A and b find x where

$$Ax = b$$

- n equations and n unknowns
- Many ways to interpret the problem , e.g., algebraic, analytic, geometric.
- Many ways to characterize the development of algorithms

Solving Linear Systems

Theorem 2.1. *Let $A \in \mathbb{C}^{n\times n}$, $x \in \mathbb{C}^n$ and $b \in \mathbb{C}^n$. A is* **nonsingular** *if and only if any of the following equivalent conditions are true*

- *The rank of A is n.*
- $\mathcal{N}(A) = \{0\}$
- *For $x \in \mathbb{C}^n$, $Ax = 0 \to x = 0$.*
- *The columns and rows of A are linearly independent.*
- *For any $b \in \mathbb{C}^n$, $Ax = b$ has a unique solution $x \in \mathbb{C}^n$.*
- *There is a (unique) matrix denoted A^{-1} such that $A^{-1}A = AA^{-1} = I$ where $I = [e_1, e_2, \cdots, e_n]$*

Solving Linear Systems

A unique solution x for

$$Ax = b$$

where A is an $n \times n$ matrix and x and b are n vectors is equivalent to A being nonsingular and then $x = A^{-1}b$.

- How do we compute this in practice?
- Is the solution accurate?
- Is the algorithm reliable?
- What do we mean by accurate and reliable?

Inverses and their Avoidance

- A nonsingular
- $x = A^{-1}b$
- Algorithm:
  - compute A^{-1}
  - compute $x = A^{-1}b$
- DISASTROUS in both complexity and numerical robustness
- Generally, we compute the effect of the inverse NOT the inverse itself.

Identity Element for Matrices

Simplest case: $A = I = [e_1, e_2, \cdots, e_n]$

$$\begin{aligned} Ax &= b \\ Ix &= b \\ x &= b \end{aligned}$$

Example 2.1. $n = 4$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \end{pmatrix} = \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{pmatrix}$$

Diagonal Matrix

$A \in \mathbb{C}^{n\times n}$ is a nonsingular diagonal matrix $\alpha_{ij} = 0$, for $i \neq j$ and $\alpha_{ii} \neq 0$, for $i = 1, \ldots, n$ due to structural orthogonality.

Example 2.2. $n = 4$

$$\begin{pmatrix} \alpha_{11} & 0 & 0 & 0 \\ 0 & \alpha_{22} & 0 & 0 \\ 0 & 0 & \alpha_{33} & 0 \\ 0 & 0 & 0 & \alpha_{44} \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \end{pmatrix} = \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{pmatrix}$$

Diagonal Matrix

This defines the following identities and nonsingularity guarantees they can be solved.:

$$
\begin{array}{lcl}
\alpha_{11}\xi_1 = \phi_1 & \rightarrow & \xi_1 = \alpha_{11}^{-1}\phi_1 \\
\alpha_{22}\xi_2 = \phi_2 & \rightarrow & \xi_2 = \alpha_{22}^{-1}\phi_2 \\
\alpha_{33}\xi_3 = \phi_3 & \rightarrow & \xi_3 = \alpha_{33}^{-1}\phi_3 \\
\alpha_{44}\xi_4 = \phi_4 & \rightarrow & \xi_4 = \alpha_{44}^{-1}\phi_4
\end{array}
$$

In other words, D^{-1} is trivially constructed and

$$x = D^{-1}b$$

This is one of the very few times we actually construct the inverse!

Unitary Matrices

Definition 2.1. A matrix in $\mathbb{C}^{m\times m}$ (or in $\mathbb{R}^{m\times m}$) with columns $Ae_i = a_i$ is said to be **unitary** (**orthogonal**) if

- $\|a_i\|_2 = 1$ for all $i = 1, \cdots, m$
- $a_i^H a_j = 0$ for $i \neq j$
- Equivalently, $AA^H = A^H A = I_m$

Example 2.3.

$$Q = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

Orthogonal Matrices

- A orthogonal $\rightarrow A^{-1} = A^T$
- Simple to check nearness to orthogonality
- Easy to solve systems $Ax = b \rightarrow x = A^T b$
- Extremely important role as
  - accurate computational primitive class
  - powerful analytical tool

Triangular System of Equations Example

For $n = 4$:

$$
\begin{aligned}
\lambda_{11}\xi_1 &= \phi_1 \\
\lambda_{21}\xi_1 + \lambda_{22}\xi_2 &= \phi_2 \\
\lambda_{31}\xi_1 + \lambda_{32}\xi_2 + \lambda_{33}\xi_3 &= \phi_3 \\
\lambda_{41}\xi_1 + \lambda_{42}\xi_2 + \lambda_{43}\xi_3 + \lambda_{44}\xi_4 &= \phi_4
\end{aligned}
$$

Note the flow of information.

Triangular System of Equations Example

This corresponds to the system in matrix form:

$$\begin{pmatrix} \lambda_{11} & 0 & 0 & 0 \\ \lambda_{21} & \lambda_{22} & 0 & 0 \\ \lambda_{31} & \lambda_{32} & \lambda_{33} & 0 \\ \lambda_{41} & \lambda_{42} & \lambda_{43} & \lambda_{44} \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \end{pmatrix} = \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{pmatrix}$$

Triangular Matrices

Definition 2.2. Suppose $L \in \mathbb{R}^{n \times n}$ and let $\lambda_{ij} = e_i^T L e_j$ then

- L is a lower triangular matrix if $\lambda_{ij} = 0$ for all $i < j$.
- L is nonsingular if and only if $\lambda_{ii} \neq 0$ for $1 \leq i \leq n$.
- $Lx = f$ is a lower triangular system of equations.

Note. This class of matrix has an interesting and extensive algebraic structure that can be exploited to derive an unexpectedly large set of algorithms.

Algorithms from Equation Structure

Rewriting the equations gives the following identities:

$$\begin{aligned}
\lambda_{11}\xi_1 &= \phi_1 \\
\lambda_{22}\xi_2 &= \phi_2 - \lambda_{21}\xi_1 \\
\lambda_{33}\xi_3 &= \phi_3 - \lambda_{31}\xi_1 - \lambda_{32}\xi_2 \\
\lambda_{44}\xi_4 &= \phi_4 - \lambda_{41}\xi_1 - \lambda_{42}\xi_2 - \lambda_{43}\xi_3
\end{aligned}$$

- Form looks more “algorithmic”.
- Two loop-based sequential algorithms are easily deduced.
- Complexity to solve $Lx = f$ is $O(n^2)$ operations.
- The two algorithms differ in that one is oriented towards rows, and the other columns of L.

Data Structure Choice

- Choose data structures to store the mathematical objects L, x and f
- a two-dimensional array and two one-dimensional arrays.
- Mapping of mathematical objects to data structures

$$\begin{aligned} L(I,J) &= \lambda_{ij} \\ X(I) &= \xi_i \\ F(I) &= \phi_i \end{aligned}$$

Row-oriented Algorithm

$Row\_oriented:$

```
X(1) = F(1) / L(1,1)
do I = 2, N
        do J = 1, I - 1
                F(I) = F(I) - L(I,J) X(J)
        enddo
        X(I) = F(I) / L(I,I)
enddo
```

Column-oriented Algorithm

$Column\_oriented:$

```
do J = 1, N-1
        X(J) = F(J) / L(J,J)
        do I = J+1, N
                F(I) = F(I) - L(I,J) X(J)
        end do
end do
X(N) = F(N) / L(N,N)
```

Matrix Form Column Algorithm

Consider the problem of expressing the actions of the column algorithm in terms of matrix operations.

Let $n = 4$, assume $\lambda_{ii} = 1$. We identify the computations performed on each iteration of the algorithm and express them as a matrix operation. These can then be combined into a factorization/transformation matrix view of the algorithm.

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ \lambda_{21} & 1 & 0 & 0 \\ \lambda_{31} & \lambda_{32} & 1 & 0 \\ \lambda_{41} & \lambda_{42} & \lambda_{43} & 1 \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \end{pmatrix} = \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{pmatrix}$$

Matrix Form Column Algorithm

for iteration $j = 1 \;\; i = 2, \cdots, 4$ computes

$$\xi_1 \leftarrow \phi_1 \quad \text{and} \quad \begin{pmatrix} \phi_2' \\ \phi_3' \\ \phi_4' \end{pmatrix} \leftarrow \begin{pmatrix} \phi_2 \\ \phi_3 \\ \phi_4 \end{pmatrix} - \begin{pmatrix} \lambda_{21} \\ \lambda_{31} \\ \lambda_{41} \end{pmatrix} \xi_1$$

Matrix Form

$$\begin{pmatrix} \xi_1 \\ \phi_2' \\ \phi_3' \\ \phi_4' \end{pmatrix} \leftarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ -\lambda_{21} & 1 & 0 & 0 \\ -\lambda_{31} & 0 & 1 & 0 \\ -\lambda_{41} & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{pmatrix}$$

Matrix Form Column Algorithm

for iteration $j = 2 \;\; i = 3, 4$ computes

$$\xi_2 \leftarrow \phi_2' \quad \text{and} \quad \begin{pmatrix} \phi_3'' \\ \phi_4'' \end{pmatrix} \leftarrow \begin{pmatrix} \phi_3' \\ \phi_4' \end{pmatrix} - \begin{pmatrix} \lambda_{32} \\ \lambda_{42} \end{pmatrix} \xi_2$$

Matrix Form

$$\begin{pmatrix} \xi_1 \\ \xi_2 \\ \phi_3'' \\ \phi_4'' \end{pmatrix} \leftarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -\lambda_{32} & 1 & 0 \\ 0 & -\lambda_{42} & 0 & 1 \end{pmatrix} \begin{pmatrix} \xi_1 \\ \phi_2' \\ \phi_3' \\ \phi_4' \end{pmatrix}$$

Matrix Form Column Algorithm

for iteration $j = 3 \;\; i = 4$ computes

$$\xi_3 \leftarrow \phi_3'' \quad \text{and} \quad \left(\phi_4'''\right) \leftarrow \left(\phi_4'\right) - \left(\lambda_{43}\right) \xi_3$$

Matrix Form

$$\begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \phi_4''' \end{pmatrix} \leftarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\lambda_{43} & 1 \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \phi_3'' \\ \phi_4' \end{pmatrix}$$

Matrix Form Column Algorithm

Last statment computes

$$\xi_4 \leftarrow \phi_4'''$$

Matrix Form

$$\begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \end{pmatrix} \leftarrow \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \phi_4''' \end{pmatrix}$$

Matrix Form Column Algorithm

$$\begin{pmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \xi_4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\lambda_{43} & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -\lambda_{32} & 1 & 0 \\ 0 & -\lambda_{42} & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ -\lambda_{21} & 1 & 0 & 0 \\ -\lambda_{31} & 0 & 1 & 0 \\ -\lambda_{41} & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \\ \phi_4 \end{pmatrix}$$

- Column form algorithm is equivalent to applying L^{-1} in a factored form
- An implementation **never applies full matrix factors**.
- Exploiting nonzero structure yields implementation.

Matrix Form Column Algorithm

This is due to the following computation-free factorization of L:

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \lambda_{21} & 1 & 0 & 0 \\ \lambda_{31} & 0 & 1 & 0 \\ \lambda_{41} & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \lambda_{32} & 1 & 0 \\ 0 & \lambda_{42} & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \lambda_{43} & 1 \end{pmatrix}$$

Each factor is easily inverted via negation.

This generalizes to any n for unit lower triangular and can be adapted to any nonsingular lower triangular.

Triangular Systems

- many algorithms possible
- sequential standards are row/column versions
- algebraic characterization
- analysis of structure in basic operations yields efficient computation in operations and space
- $O(n^2)$ operations and space

Gaussian Elimination

Problem 2.1. Find the unique x such that

$$Ax = b$$

where $A \in \mathbb{R}^{n \times n}$, $x, b \in \mathbb{R}^n$, and A^{-1} exists.

Elimination

- Elimination is the process of removing variables from equations, i.e., setting their coefficients to 0.
- A system of n equations and n unknowns with the same unique solution x must be maintained by this process.
- Basic Facts:
  - Any linear combination of the equations given is also a true equation.
  - Linear independence must be maintained to preserve the solution x.
  - many elimination “strategies” are possible.

Example

$$5\xi_1 + \xi_2 + 3\xi_3 = 9 \tag{1}$$

$$10\xi_1 + 5\xi_2 + 12\xi_3 = 27 \tag{2}$$

$$5\xi_1 + 10\xi_2 + 23\xi_3 = 38 \tag{3}$$

Combine to generate new equations and replace while keeping independence:

- Equation (2) minus 2 times Equation (1) $\rightarrow$ Equation (2) .
- Equation (3) minus Equation (1) $\rightarrow$ Equation (3) .

Example

$$5\xi_1 + \xi_2 + 3\xi_3 = 9 \tag{4}$$

$$0\xi_1 + 3\xi_2 + 6\xi_3 = 9 \tag{5}$$

$$0\xi_1 + 9\xi_2 + 20\xi_3 = 29 \tag{6}$$

Combine and replace:

- Equation (4) minus $1/3$ times Equation (5) $\rightarrow$ Equation (4) .
- Equation (6) minus 3 times Equation (5) $\rightarrow$ Equation (6) .

Example

$$5\xi_1 + 0\xi_2 + \xi_3 = 6 \tag{7}$$

$$0\xi_1 + 3\xi_2 + 6\xi_3 = 9 \tag{8}$$

$$0\xi_1 + 0\xi_2 + 2\xi_3 = 2 \tag{9}$$

Combine and replace:

- Equation (7) minus $1/2$ times Equation (9) $\rightarrow$ Equation (7) .
- Equation (8) minus 3 times Equation (9) $\rightarrow$ Equation (8) .

Example

$$5\xi_1 + 0\xi_2 + 0\xi_3 = 5 \tag{10}$$

$$0\xi_1 + 3\xi_2 + 0\xi_3 = 3 \tag{11}$$

$$0\xi_1 + 0\xi_2 + 2\xi_3 = 2 \tag{12}$$

- Each equation defines one variable.
- $\xi_1 = \xi_2 = \xi_3 = 1$
- Diagonal system of equations.
- This is called Gauss-Jordan elimination.

Matrix Form

$$\begin{pmatrix} 5 & 1 & 3 \\ 0 & 3 & 6 \\ 0 & 9 & 20 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 5 & 1 & 3 \\ 10 & 5 & 12 \\ 5 & 10 & 23 \end{pmatrix}$$

$$\begin{pmatrix} 5 & 0 & 1 \\ 0 & 3 & 6 \\ 0 & 0 & 2 \end{pmatrix} = \begin{pmatrix} 1 & -1/3 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{pmatrix} \begin{pmatrix} 5 & 1 & 3 \\ 0 & 3 & 6 \\ 0 & 9 & 20 \end{pmatrix}$$

$$\begin{pmatrix} 5 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1/2 \\ 0 & 1 & -3 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 5 & 0 & 1 \\ 0 & 3 & 6 \\ 0 & 0 & 2 \end{pmatrix}$$

Matrix Form

Apply matrices to b also.

$$\begin{pmatrix}5\\3\\2\end{pmatrix} = \begin{pmatrix}1 & 0 & -1/2\\0 & 1 & -3\\0 & 0 & 1\end{pmatrix} \begin{pmatrix}1 & -1/3 & 0\\0 & 1 & 0\\0 & -3 & 1\end{pmatrix} \begin{pmatrix}1 & 0 & 0\\-2 & 1 & 0\\-1 & 0 & 1\end{pmatrix} \begin{pmatrix}9\\27\\38\end{pmatrix}$$

$$\begin{pmatrix}5 & 0 & 0\\0 & 3 & 0\\0 & 0 & 2\end{pmatrix} \begin{pmatrix}1\\1\\1\end{pmatrix} = \begin{pmatrix}5\\3\\2\end{pmatrix}$$

Note we have an expression for the inverse of the original matrix!

Gaussian Elimination

- We do not have to transform the matrix into a diagonal form.
- We know how to solve triangular systems.
- Repeat the idea with fewer eliminations.

Matrix Form

$$\begin{pmatrix} 5 & 1 & 3 \\ 0 & 3 & 6 \\ 0 & 9 & 20 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 5 & 1 & 3 \\ 10 & 5 & 12 \\ 5 & 10 & 23 \end{pmatrix}$$

$$\begin{pmatrix} 5 & 1 & 3 \\ 0 & 3 & 6 \\ 0 & 0 & 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{pmatrix} \begin{pmatrix} 5 & 1 & 3 \\ 0 & 3 & 6 \\ 0 & 9 & 20 \end{pmatrix}$$

Matrix Form

Apply matrices to b also.

$$\begin{pmatrix} 9 \\ 9 \\ 2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 9 \\ 27 \\ 38 \end{pmatrix}$$

$$\begin{pmatrix} 5 & 1 & 3 \\ 0 & 3 & 6 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 9 \\ 9 \\ 2 \end{pmatrix}$$

Gaussian Elimination

- Gaussian elimination transforms the problem to one we know how to solve.
- equation-based interpretation
- transformation-based derivation
- factorization-based interpretation

$$A = LU$$

where L is a unit lower triangular matrix and U is an upper triangular matrix.

Note. We will assume for now that the factorization exists and the algorithm does not fail.

Transformation-based Point of View

- A is a nonsingular $\rightarrow$ its columns are a basis for $\mathbb{R}^n$.
- Problems involving bases that are the columns of lower or upper triangular systems are easy to solve.
- Change the basis of the problem. T must be nonsingular.

$$
\begin{aligned}
Ax &= b \\
T(Ax) &= Tb \\
(TA)x &= c \\
Ux &= c
\end{aligned}
$$

- x contains the coordinates of b relative to the basis given by A
- x contains the coordinates of c relative to the basis given by U.

Examples of Gauss Transforms

$$C_1 v = y$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ -\frac{1}{3} & 1 & 0 & 0 \\ -\frac{4}{3} & 0 & 1 & 0 \\ -\frac{5}{3} & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ 1 \\ 4 \\ 5 \end{pmatrix} = \begin{pmatrix} 3 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$$C_2 v = y$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -\frac{1}{3} & 1 & 0 \\ 0 & -\frac{4}{3} & 0 & 1 \end{pmatrix} \begin{pmatrix} 10 \\ 3 \\ 1 \\ 4 \end{pmatrix} = \begin{pmatrix} 10 \\ 3 \\ 0 \\ 0 \end{pmatrix}$$

Application of a Gauss Transform to a Matrix

Let $n = 4$

$$M_1^{-1} A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -\lambda_{21} & 1 & 0 & 0 \\ -\lambda_{31} & 0 & 1 & 0 \\ -\lambda_{41} & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} & \alpha_{24} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} & \alpha_{34} \\ \alpha_{41} & \alpha_{42} & \alpha_{43} & \alpha_{44} \end{pmatrix}$$

If $\lambda_j^{(1)} = \alpha_{j1}/\alpha_{11}$ where $\alpha_{11} \neq 0$ then

$$M_1^{-1} A = A^{(1)} = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} & \alpha_{14} \\ 0 & \tilde{\alpha}_{22} & \tilde{\alpha}_{23} & \tilde{\alpha}_{24} \\ 0 & \tilde{\alpha}_{32} & \tilde{\alpha}_{33} & \tilde{\alpha}_{34} \\ 0 & \tilde{\alpha}_{42} & \tilde{\alpha}_{43} & \tilde{\alpha}_{44} \end{pmatrix}$$

Update Structure

After applying the Gauss transform defined by Ae_1 to A:

- The first row of A remains the same and becomes the first row of U.
- Elements in rows 2 to n in the first column are 0
- All other elements are updated.
- Essentially, this is a rank-1 update of the lower right submatrix of order $n - 1$.
- Note the important difference in notation with the text book.

Preservation of Structure

Example 2.4. Let $n = 4$ then $M_2^{-1}M_1^{-1}A$ has the structure

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -\lambda_{21} & 1 & 0 \\ 0 & -\lambda_{31} & 0 & 1 \end{pmatrix} \begin{pmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \\ 0 & x & x & x \end{pmatrix} = \begin{pmatrix} x & x & x & x \\ 0 & x & x & x \\ 0 & 0 & x' & x' \\ 0 & 0 & x' & x' \end{pmatrix}$$

Note. Applying M_2^{-1} to $M_1^{-1}A$ **does not destroy the 0 elements introduced in the first column**.

Full Transformation

The process can be repeated to eliminate the nonzeros in subsequent columns without destroying the zeros introduced by all previous transformations.

After $n-1$ such steps we have

$$M_{n-1}^{-1} \cdots M_2^{-1} M_1^{-1} A = U = \begin{pmatrix} \mu_{11} & \mu_{12} & \cdots & \mu_{1n} \\ 0 & \mu_{22} & \cdots & \mu_{2n} \\ \vdots & & & \vdots \\ 0 & \cdots & 0 & \mu_{nn} \end{pmatrix}$$

$M_{n-1}^{-1} \cdots M_2^{-1} M_1^{-1} b = c$ can be performed at the same time or whenever b is supplied.

$Ux = c$ is easily solved.

LU Factorization

We have A and U but where is L ?

- We have

$$
\begin{aligned}
U &= M_{n-1}^{-1} \cdots M_2^{-1} M_1^{-1} A \\
A &= M_1 M_2 \cdots M_{n-1} U
\end{aligned}
$$

  where M_i is an elementary lower triangular matrix given by changing the sign of the elements below the diagonal in M_i^{-1}.

- $M_1 \cdots M_{n-1}$ is a unit lower triangular matrix.

- No further computation needed to determine L from the M_i

Elements of L

The nonzeros below the diagonal in the i-th column are given by the nonzero elements of l_i for $i = 1, \ldots, n-1$, i.e.,

$$L = \begin{pmatrix} 1 & & & \\ \lambda_{2,1} & 1 & & \\ \vdots & & \ddots & \\ \lambda_{n,1} & \cdots & \lambda_{n,n-1} & 1 \end{pmatrix}$$

Partitioning-based Pseudo-code

The i-th stage of the algorithm works on an $n-i \times n-i$ matrix that is updated on the previous step.

Algebraically this can be expressed as:

Let $A_0 = A$ and let $r_i, c_i \in \mathbb{R}^{n-i-1}$, $B_i \in \mathbb{R}^{n-i-1 \times n-i-1}$ and

$$A_i = \begin{bmatrix} \alpha_{11}^{(i)} & r_i^T \\ c_i & B_i \end{bmatrix}$$

We then have

$$\begin{array}{ll} \text{do} & I = 1, n-1 \\ & v_i = (1/\alpha_{11}^{(i-1)})c_{i-1} \\ & A_i = B_{i-1} - v_i r_{i-1}^T \\ \text{enddo} & \end{array}$$

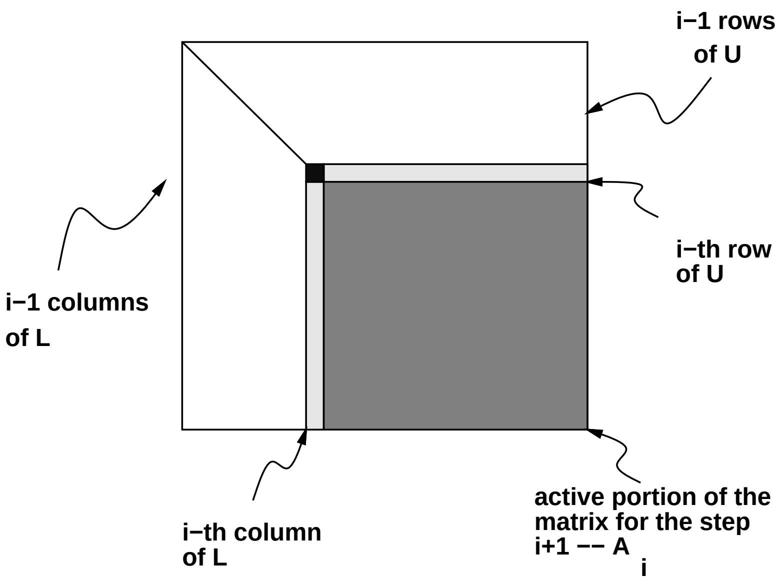
Computational Comments

- Algebraically, each step requires $M_i^{-1} A^{(i-1)}$, i.e., $n \times n$ matrix multiplication
- This form of the algorithm is a series of $n-1$ stages with $A = A_0$ the i-th stage of which is a BLAS1 primitive and a BLAS2 primitive:
  - Scale the nonzeros below the diagonal of the first column of A_{i-1} to produce l_i.
  - Perform a rank-1 update of order $n-i$ to produce A_i.
- Note the invariant structure of the product $M_i^{-1} A^{(i-1)}$ is exploited to reduce the size of the primitives on each step.
- storage complexity remains $O(n^2)$ since there is one λ_{ij} for each 0 produced in A.

Update/production Characterization

- Stage i produces column i of L
- Stage i produces row i of U
- Stage i updates the remaining *active* part of A of dimension $n - i \times n - i$.

Immediate Update Form of LU



i −1 columns of L and i−1 rows of U are not touched.

i−th column of L is computed and used along with the i−th row of U

A_i computed and the active portion

of the matrix updated.

Example In-place Elimination Data Structure

Distinguish between arrays (data structure – array) and matrices (mathematical objects – $A, \; L, \; U$)

Initialize : $\text{array}(I, J) = \alpha_{ij}$

$$A = \begin{pmatrix} 5 & 1 & 3 & 1 \\ 10 & 5 & 12 & 3 \\ 5 & 10 & 23 & 5 \\ 15 & 6 & 19 & 7 \end{pmatrix}$$

$$\text{array} = \begin{bmatrix} 5 & 1 & 3 & 1 \\ 10 & 5 & 12 & 3 \\ 5 & 10 & 23 & 5 \\ 15 & 6 & 19 & 7 \end{bmatrix}$$

Example In-place Elimination Data Structure

Step 1:

$$\begin{pmatrix} 5 & 1 & 3 & 1 \\ 0 & 3 & 6 & 1 \\ 0 & 9 & 20 & 4 \\ 0 & 3 & 10 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -2 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -3 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 5 & 1 & 3 & 1 \\ 10 & 5 & 12 & 3 \\ 5 & 10 & 23 & 5 \\ 15 & 6 & 19 & 7 \end{pmatrix}$$

$$\text{array} = \begin{bmatrix} 5 & 1 & 3 & 1 \\ 10 & 5 & 12 & 3 \\ 5 & 10 & 23 & 5 \\ 15 & 6 & 19 & 7 \end{bmatrix} \Rightarrow \text{array} = \begin{bmatrix} 5 & 1 & 3 & 1 \\ 2 & 3 & 6 & 1 \\ 1 & 9 & 20 & 4 \\ 3 & 3 & 10 & 4 \end{bmatrix}$$

Example In-place Elimination Data Structure

Step 2:

$$\begin{pmatrix} 5 & 1 & 3 & 1 \\ 0 & 3 & 6 & 1 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 4 & 3 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -3 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 5 & 1 & 3 & 1 \\ 0 & 3 & 6 & 1 \\ 0 & 9 & 20 & 4 \\ 0 & 3 & 10 & 4 \end{pmatrix}$$

$$\text{array} = \begin{bmatrix} 5 & 1 & 3 & 1 \\ 2 & 3 & 6 & 1 \\ 1 & 9 & 20 & 4 \\ 3 & 3 & 10 & 4 \end{bmatrix} \Rightarrow \text{array} = \begin{bmatrix} 5 & 1 & 3 & 1 \\ 2 & 3 & 6 & 1 \\ 1 & 3 & 2 & 1 \\ 3 & 1 & 4 & 3 \end{bmatrix}$$

Example In-place Elimination Data Structure

Step 3:

$$\begin{pmatrix} 5 & 1 & 3 & 1 \\ 0 & 3 & 6 & 1 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -2 & 1 \end{pmatrix} \begin{pmatrix} 5 & 1 & 3 & 1 \\ 0 & 3 & 6 & 1 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 4 & 3 \end{pmatrix}$$

$$\text{array} = \begin{bmatrix} 5 & 1 & 3 & 1 \\ 2 & 3 & 6 & 1 \\ 1 & 3 & 2 & 1 \\ 3 & 1 & 4 & 3 \end{bmatrix} \Rightarrow \text{array} = \begin{bmatrix} 5 & 1 & 3 & 1 \\ 2 & 3 & 6 & 1 \\ 1 & 3 & 2 & 1 \\ 3 & 1 & 2 & 1 \end{bmatrix}$$

Example In-place Elimination Data Structure

$$A = LU$$

$$\begin{pmatrix} 5 & 1 & 3 & 1 \\ 10 & 5 & 12 & 3 \\ 5 & 10 & 23 & 5 \\ 15 & 6 & 19 & 7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 1 & 3 & 1 & 0 \\ 3 & 1 & 2 & 1 \end{pmatrix} \begin{pmatrix} 5 & 1 & 3 & 1 \\ 0 & 3 & 6 & 1 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Complexity

The algorithm and its complexity is:

1. calculate L and U, $\Omega \approx \frac{2}{3}n^3$
2. solve $Ly = b$, $\Omega \approx n^2$
3. solve $Ux = y$, $\Omega \approx n^2$

Storage:

- A and b input requires $n^2 + n$ space
- L and U and x output requires $n^2 + O(n)$ space
- L and U can overwrite A giving $n^2 + O(n)$ space

Failure of LU

Note. A is nonsingular **does not** imply $A = LU$.

Example 2.5. The first step fails for the following nonsingular matrix.

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

Pivoting

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

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

Definition 2.3. 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.

Pivoting

Definition 2.4. 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.

- There are many pivoting strategies.
- Some depend on the assumptions made about the matrix.
- The main factor in defining them is the set of candidate pivots.
- On each step of the factorization, the entire set of candidate pivots must be computed and examined.

Row Permutations

Definition 2.5. An elementary permutation matrix P that interchanges rows i and j of the matrix to which it is applied by premultiplication is the identity matrix I with rows i and j interchanged.

Example 2.6. To interchange rows 1 and 3 of a matrix A of order 3 compute PA where

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{pmatrix} = \begin{pmatrix} \alpha_{31} & \alpha_{32} & \alpha_{33} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{11} & \alpha_{12} & \alpha_{13} \end{pmatrix}$$

Column Permutations

Definition 2.6. An elementary permutation matrix Q that interchanges columns i and j of the matrix to which it is applied by postmultiplication is the identity matrix I with columns i and j interchanged.

Example 2.7. To interchange columns 1 and 3 of a matrix A of order 3 compute AQ where

$$\begin{pmatrix} \alpha_{11} & \alpha_{12} & \alpha_{13} \\ \alpha_{21} & \alpha_{22} & \alpha_{23} \\ \alpha_{31} & \alpha_{32} & \alpha_{33} \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} \alpha_{13} & \alpha_{12} & \alpha_{11} \\ \alpha_{23} & \alpha_{22} & \alpha_{21} \\ \alpha_{33} & \alpha_{32} & \alpha_{31} \end{pmatrix}$$

Elementary Permutations

- $P = P^{-1}$ and $Q = Q^{-1}$ follow immediately from the definition of elementary permutations.
- 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 elementary 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.

Partial Pivoting

Definition 2.7. 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.

Partial Pivoting Algebraic Form

$$
\begin{aligned}
U &= (M_{n-1}^{-1}(P_{n-1} \cdots (M_1^{-1}(P_1 A)) \cdots)) \\
U &= (M_{n-1}^{-1} P_{n-1} \cdots M_1^{-1} P_1) A \\
U &= TA \\
T &= M_{n-1}^{-1} P_{n-1} \cdots M_1^{-1} P_1
\end{aligned}
$$

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

$$
\begin{aligned}
Ax &= b \\
TAx &= Tb \\
Ux &= \tilde{b}
\end{aligned}
$$

LU Factorization?

T is not lower triangular so where is the L?

Lemma 2.2. *If A^{-1} exists then there exists a product of elementary row permutation matrices $P = P_{n-1} \ldots P_1$, 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.

LU Factorization with Partial Pivoting

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 2.3.

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

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

Constructive Proof

It is simple, given the definition of $\tilde{M}_i$ and P, to verify $PA = LU$.

It is also possible to give a constructive proof for the form of $\tilde{M}_i$.

The construction creates P, then $\tilde{M}_1^{-1}$, $\tilde{M}_2^{-1}, \ldots, \tilde{M}_{n-1}^{-1}$, by injecting I, in terms of the P_i, $n-2$ times and using associativity.

It can be seen from an example with $n = 5$. Recall,

- $P_i = P_i^{-1}$.
- $P_j \ldots P_{i+1} M_i^{-1} P_{i+1} \ldots P_j$ has elementary lower triangular structure for any $j > i$.

Constructive Proof

$$M_4^{-1}P_4M_3^{-1}P_3M_2^{-1}P_2M_1^{-1}P_1A = U$$

$$M_4^{-1}P_4M_3^{-1}P_3M_2^{-1}P_2M_1^{-1}(P_2P_3P_4P_4P_3P_2)P_1A = U$$

$$M_4^{-1}P_4M_3^{-1}P_3M_2^{-1}P_2M_1^{-1}P_2P_3P_4(P_4P_3P_2P_1)A = U$$

$$M_4^{-1}P_4M_3^{-1}P_3M_2^{-1}P_2M_1^{-1}P_2P_3P_4PA = U$$

$$M_4^{-1}P_4M_3^{-1}P_3M_2^{-1}(P_3P_4P_4P_3)P_2M_1^{-1}P_2P_3P_4PA = U$$

$$M_4^{-1}P_4M_3^{-1}P_3M_2^{-1}P_3P_4(P_4P_3P_2M_1^{-1}P_2P_3P_4)PA = U$$

$$M_4^{-1}P_4M_3^{-1}P_3M_2^{-1}P_3P_4\tilde{M}_1^{-1}PA = U$$

$$M_4^{-1}P_4M_3^{-1}(P_4P_4)P_3M_2^{-1}P_3P_4\tilde{M}_1^{-1}PA = U$$

$$M_4^{-1}(P_4M_3^{-1}P_4)(P_4P_3M_2^{-1}P_3P_4)\tilde{M}_1^{-1}PA = U$$

$$\tilde{M}_4^{-1}\tilde{M}_3^{-1}\tilde{M}_2^{-1}\tilde{M}_1^{-1}PA = U$$

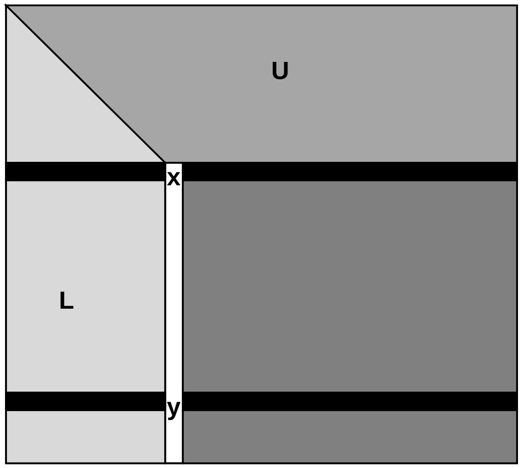
Algorithmic Consequence

$$\tilde{M}_i^{-1} = P_{n-1} \ldots P_{i+1} M_i^{-1} P_{i+1}^{-1} \ldots P_{n-1}^{-1}$$

- It follows that for BLAS2-based algorithms we must 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

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.

Example

$$A = \begin{pmatrix} 3 & 17 & 10 \\ 2 & 4 & -2 \\ 6 & 18 & -12 \end{pmatrix}$$

$$P_1 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad M_1^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ -1/3 & 1 & 0 \\ -1/2 & 0 & 1 \end{pmatrix}$$

$$M_1^{-1} P_1 A = \begin{pmatrix} 6 & 18 & -12 \\ 0 & -2 & 2 \\ 0 & 8 & 16 \end{pmatrix}$$

Example

$$M_1^{-1} P_1 A = \begin{pmatrix} 6 & 18 & -12 \\ 0 & -2 & 2 \\ 0 & 8 & 16 \end{pmatrix}$$

$$P_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad M_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1/4 & 1 \end{pmatrix}$$

$$M_2^{-1} P_2 M_1^{-1} P_1 A = \begin{pmatrix} 6 & 18 & -12 \\ 0 & 8 & 16 \\ 0 & 0 & 6 \end{pmatrix} = U$$

Example

$$P = P_2 P_1 = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\tilde{M}_1 = P_2 M_1 P_2 = \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 1/3 & 0 & 1 \end{pmatrix}, \quad \tilde{M}_2 = M_2$$

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 1/3 & -1/4 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 6 & 18 & -12 \\ 0 & 8 & 16 \\ 0 & 0 & 6 \end{pmatrix}$$

Example

$$LU = \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 1/3 & -1/4 & 1 \end{pmatrix} \begin{pmatrix} 6 & 18 & -12 \\ 0 & 8 & 16 \\ 0 & 0 & 6 \end{pmatrix} = \begin{pmatrix} 6 & 18 & -12 \\ 3 & 17 & 10 \\ 2 & 4 & -2 \end{pmatrix} = PA$$

Example Data In-place Pivoting

$$array = \begin{bmatrix} 3 & 17 & 10 \\ 2 & 4 & -2 \\ 6 & 18 & -12 \end{bmatrix} \xrightarrow{\text{permute matrix}} array = \begin{bmatrix} 6 & 18 & -12 \\ 2 & 4 & -2 \\ 3 & 17 & 10 \end{bmatrix}$$

$$array = \begin{bmatrix} 6 & 18 & -12 \\ 2 & 4 & -2 \\ 3 & 17 & 10 \end{bmatrix} \xrightarrow{\text{multipliers}} array = \begin{bmatrix} 6 & 18 & -12 \\ 1/3 & 4 & -2 \\ 1/2 & 17 & 10 \end{bmatrix}$$

$$array = \begin{bmatrix} 6 & 18 & -12 \\ 1/3 & 4 & -2 \\ 1/2 & 17 & 10 \end{bmatrix} \xrightarrow{\text{eliminate}} array = \begin{bmatrix} 6 & 18 & -12 \\ 1/3 & -2 & 2 \\ 1/2 & 8 & 16 \end{bmatrix}$$

Example Data In-place Pivoting

$$array = \begin{bmatrix} 6 & 18 & -12 \\ 1/3 & -2 & 2 \\ 1/2 & 8 & 16 \end{bmatrix} \xrightarrow[\text{permute } M_1]{\text{permute matrix}} array = \begin{bmatrix} 6 & 18 & -12 \\ 1/2 & 8 & 16 \\ 1/3 & -2 & 2 \end{bmatrix}$$

$$array = \begin{bmatrix} 6 & 18 & -12 \\ 1/2 & 8 & 16 \\ 1/3 & -2 & 2 \end{bmatrix} \xrightarrow{\text{multipliers}} array = \begin{bmatrix} 6 & 18 & -12 \\ 1/2 & 8 & 16 \\ 1/3 & -1/4 & 2 \end{bmatrix}$$

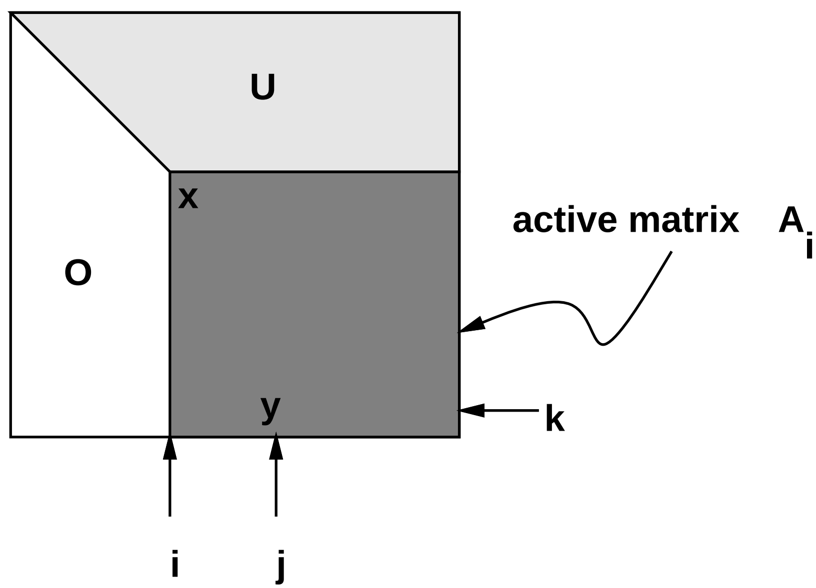
$$array = \begin{bmatrix} 6 & 18 & -12 \\ 1/2 & 8 & 16 \\ 1/3 & -1/4 & 2 \end{bmatrix} \xrightarrow{\text{eliminate}} array = \begin{bmatrix} 6 & 18 & -12 \\ 1/2 & 8 & 16 \\ 1/3 & -1/4 & 6 \end{bmatrix}$$

Complete Pivoting

Definition 2.8. Complete pivoting: Before eliminating the subdiagonal elements in the i-th column of the transformed matrix (the first column of the current 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
- The set of candidate pivot elements is the entire active matrix, therefore, the immediate update forms **must** be used.
- 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.

Complete Pivoting



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.

General Pivoting Algebraic Form

Given any choice of pivot element, we have

$$
\begin{aligned}
U =& (M_{n-1}^{-1}(P_{n-1}(\cdots(M_1^{-1}(P_1 A Q_1))\cdots))Q_{n-1})) \\
U =& (M_{n-1}^{-1}P_{n-1}\cdots M_1^{-1}P_1)A(Q_1\cdots Q_{n-1}) \\
U =& TAQ \\
T =& M_{n-1}^{-1}P_{n-1}\cdots M_1^{-1}P_1 \\
Q =& Q_1\cdots Q_{n-1}
\end{aligned}
$$

Transformation-based Solution

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

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

Complete Pivoting Factorization

Lemma 2.4. *If A^{-1} exists then there exists a product of elementary row permutation matrices $P = P_{n-1} \ldots P_1$, a product of elementary column permutation matrices $Q = Q_1 \ldots Q_{n-1}$, a unit lower triangular matrix L, and an upper triangular matrix U such that*

$$PAQ = LU.$$

L is defined in the same way as was done for partial pivoting.