Background: Linear systems

The Problem: A is an \(n \times n\) matrix, and \(b\) a vector of \(\mathbb{R}^n\). Find \(x\) such that:

\[ Ax = b \]

\(x\) is the unknown vector, \(b\) the right-hand side, and \(A\) is the coefficient matrix.

Example:

\[
\begin{align*}
2x_1 + 4x_2 + 4x_3 &= 6 \\
x_1 + 5x_2 + 6x_3 &= 4 \\
x_1 + 3x_2 + x_3 &= 8
\end{align*}
\]

or

\[
\begin{pmatrix}
2 & 4 & 4 \\
1 & 5 & 6 \\
1 & 3 & 1
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3
\end{pmatrix}
= \begin{pmatrix} 6 \\ 4 \\ 8 \end{pmatrix}
\]

A solution of above system?

Standard mathematical solution by Cramer’s rule:

\[ x_i = \frac{\det(A_i)}{\det(A)} \]

\(A_i\) = matrix obtained by replacing \(i\)-th column by \(b\).

Note: This formula is useless in practice beyond \(n = 3\) or \(n = 4\).

Three situations:

1. The matrix \(A\) is nonsingular. There is a unique solution given by \(x = A^{-1}b\).
2. The matrix \(A\) is singular and \(b \in \text{Ran}(A)\). There are infinitely many solutions.
3. The matrix \(A\) is singular and \(b \notin \text{Ran}(A)\). There are no solutions.
Triangular linear systems

Example:

\[
\begin{pmatrix}
2 & 4 & 4 \\
0 & 5 & -2 \\
0 & 0 & 2
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3
\end{pmatrix}
= 
\begin{pmatrix}
2 \\
1 \\
4
\end{pmatrix}
\]

- One equation can be trivially solved: the last one. \( x_3 = 2 \)
- \( x_3 \) is known we can now solve the 2nd equation:

\[5x_2 - 2x_3 = 1 \rightarrow 5x_2 - 2 \times 2 = 1 \rightarrow x_2 = 1\]

- Finally \( x_1 \) can be determined similarly:

\[2x_1 + 4x_2 + 4x_3 = 2 \rightarrow ... \rightarrow x_1 = -5\]

Backward error analysis for the triangular solve

The computed solution \( \hat{x} \) of the triangular system \( Ux = b \) computed by the previous algorithm satisfies:

\[(U + E)\hat{x} = b\]

with

\[|E| \leq n \ u \ |U| + O(u^2)\]

- Backward error analysis. Computed \( x \) solves a slightly perturbed system.
- Backward error not large in general. It is said that triangular solve is “backward stable”.

ALGORITHM : 1. Back-Substitution algorithm

For \( i = n : -1 : 1 \) do:

\[t := b_i\]

For \( j = i + 1 : n \) do

\[t := t - a_{ij} x_j\]

End

\[x_i = t/a_{ii}\]

End

- We must require that each \( a_{ii} \neq 0 \)
- Operation count?
- Round-off error (use previous results for \((\cdot, \cdot)\))?

Column version of back-substitution:

Back-Substitution algorithm. Column version

For \( j = n : -1 : 1 \) do:

\[x_j = b_j/a_{jj}\]

For \( i = 1 : j - 1 \) do

\[b_i := b_i - x_j * a_{ij}\]

End

End

Justify the above algorithm [Show that it does indeed compute the solution]

- See text for analogous algorithms for lower triangular systems.
Linear Systems of Equations: Gaussian Elimination

Back to arbitrary linear systems.

Principle of the method: Since triangular systems are easy to solve, we will transform a linear system into one that is triangular.

Main operation: combine rows so that zeros appear in the required locations to make the system triangular.

Notation: use a Tableau:

\[
\begin{bmatrix}
2x_1 + 4x_2 + 4x_3 &=& 2 \\
x_1 + 3x_2 + x_3 &=& 1 \\
x_1 + 5x_2 + 6x_3 &=& -6
\end{bmatrix}
\]

Example:

Replace row2 by: row2 - \( \frac{1}{2} \times \) row1:

\[
\begin{bmatrix}
2 & 4 & 4 & 2 \\
1 & 3 & 1 & 1 \\
1 & 5 & 6 & -6
\end{bmatrix} \rightarrow \begin{bmatrix}
2 & 4 & 4 & 2 \\
0 & 1 & -1 & 0 \\
1 & 5 & 6 & -6
\end{bmatrix}
\]

This is equivalent to:

\[
\begin{bmatrix}
1 & 0 & 0 \\
-\frac{1}{2} & 1 & 0 \\
0 & -\frac{1}{2} & 0
\end{bmatrix} \times \begin{bmatrix}
2 & 4 & 4 & 2 \\
1 & 3 & 1 & 1 \\
1 & 5 & 6 & -6
\end{bmatrix} = \begin{bmatrix}
2 & 4 & 4 & 2 \\
0 & 1 & -1 & 0 \\
0 & 3 & 4 & -7
\end{bmatrix}
\]

The left-hand matrix is of the form

\[
M = I - ve_1^T
\]

with

\[
v = \begin{bmatrix}
\frac{1}{2} \\
0 \\
\end{bmatrix}
\]

Go back to original system. Step 1 must transform:

\[
\begin{bmatrix}
2 & 4 & 4 & 2 \\
0 & 1 & -1 & 0 \\
1 & 5 & 6 & -6
\end{bmatrix}
\]

row2 := row2 - \( \frac{1}{2} \) \times row1: row3 := row3 - \( \frac{1}{2} \) \times row1:

\[
\begin{bmatrix}
2 & 4 & 4 & 2 \\
0 & 1 & -1 & 0 \\
1 & 5 & 6 & -6
\end{bmatrix} \rightarrow \begin{bmatrix}
2 & 4 & 4 & 2 \\
0 & 1 & -1 & 0 \\
0 & 3 & 4 & -7
\end{bmatrix}
\]

Equivalent to

\[
\begin{bmatrix}
1 & 0 & 0 \\
-\frac{1}{2} & 1 & 0 \\
0 & -\frac{1}{2} & 0
\end{bmatrix} \times \begin{bmatrix}
2 & 4 & 4 & 2 \\
0 & 1 & -1 & 0 \\
0 & 3 & 4 & -7
\end{bmatrix} = \begin{bmatrix}
2 & 4 & 4 & 2 \\
0 & 1 & -1 & 0 \\
0 & 0 & 1 & -1
\end{bmatrix}
\]

New system \( A_1x = b_1 \). Step 2 must now transform:

\[
\begin{bmatrix}
2 & 4 & 4 & 2 \\
0 & 1 & -1 & 0 \\
0 & 3 & 4 & -7
\end{bmatrix} \rightarrow \begin{bmatrix}
2 & 4 & 4 & 2 \\
0 & 1 & -1 & 0 \\
0 & 0 & 1 & -1
\end{bmatrix}
\]

\[
[A, b] \rightarrow [M_1A, M_1b]; \quad M_1 = I - v^{(1)}e_1^T; \quad v^{(1)} = \begin{bmatrix}
0 \\
\frac{1}{2} \\
\end{bmatrix}
\]
\[ row_3 := row_3 - 3 \times row_2 : \rightarrow \begin{bmatrix} 2 & 4 & 4 & 2 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 7 & -7 \end{bmatrix} \]

Equivalent to
\[
\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -3 & 1 \end{bmatrix} \times \begin{bmatrix} 2 & 4 & 4 & 2 \\ 0 & 1 & -1 & 0 \\ 0 & 3 & 4 & -7 \end{bmatrix} = \begin{bmatrix} 2 & 4 & 4 & 2 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 7 & -7 \end{bmatrix}
\]

Second transformation is as follows:
\[
[A_1, b_1] \rightarrow [M_2A_1, M_2b_1] \quad M_2 = I - v^{(2)}e^T_2 \quad v^{(2)} = \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix}
\]

Triangular system

\[ \text{Solve.} \]

\[ ALGORITHM : 2. \quad \text{Gaussian Elimination} \]

1. For \( k = 1 : n - 1 \) Do:
2. For \( i = k + 1 : n \) Do:
3. \( \text{piv} := a_{ik}/a_{kk} \)
4. For \( j := k + 1 : n + 1 \) Do :
5. \( a_{ij} := a_{ij} - \text{piv} \times a_{kj} \)
6. End
7. End
8. End

Operation count:
\[
T = \sum_{k=1}^{n-1} \left[ \sum_{i=k+1}^{n} \sum_{j=k+1}^{n+1} 2 \right] = \sum_{k=1}^{n-1} \sum_{i=k+1}^{n} (2(n-k)+3) = \ldots
\]

Complete the above calculation. Order of the cost?

\[ \text{The LU factorization} \]

Now ignore the right-hand side from the transformations.

Observation: Gaussian elimination is equivalent to \( n - 1 \) successive Gaussian transformations, i.e., multiplications with matrices of the form \( M_k = I - v^{(k)}e^T_k \), where the first \( k \) components of \( v^{(k)} \) equal zero.

Set \( A_0 \equiv A \)

\[ A \rightarrow M_1A_0 = A_1 \rightarrow M_2A_1 = A_2 \rightarrow M_3A_2 = A_3 \cdots \rightarrow M_{n-1}A_{n-2} = A_{n-1} \equiv U \]

Last \( A_k \equiv U \) is an upper triangular matrix.
At each step we have: \( A_k = M_{k+1}^{-1}A_{k+1} \). Therefore:

\[
A_0 = M_1^{-1} A_1 \\
= M_1^{-1} M_2^{-1} A_2 \\
= M_1^{-1} M_2^{-1} M_3^{-1} A_3 \\
= \ldots \\
= M_1^{-1} M_2^{-1} M_3^{-1} \cdots M_{n-1}^{-1} A_{n-1}
\]

\( L = M_1^{-1} M_2^{-1} M_3^{-1} \cdots M_{n-1}^{-1} \)

Consider only the first 2 matrices in this product.

Note: \( L \) is lower triangular, \( A_{n-1} \) is upper triangular.

LU decomposition: \( A = LU \)

A matrix \( A \) has an LU decomposition if

\[
\det(A(1:k,1:k)) \neq 0 \quad \text{for} \quad k = 1, \ldots, n - 1.
\]

In this case, the determinant of \( A \) satisfies:

\[
\det A = \det(U) = \prod_{i=1}^{n} u_{ii}
\]

If, in addition, \( A \) is nonsingular, then the LU factorization is unique.

Practical use: Show how to use the LU factorization to solve linear systems with the same matrix \( A \) and different \( b \)'s.

LU factorization of the matrix \( A = \begin{pmatrix} 2 & 4 & 4 \\ 1 & 5 & 6 \\ 1 & 3 & 1 \end{pmatrix} \) ?

Determinant of \( A \)?

True or false: “Computing the LU factorization of matrix \( A \) involves more arithmetic operations than solving a linear system \( Ax = b \) by Gaussian elimination”.
Gauss-Jordan Elimination

Principle of the method: We will now transform the system into one that is even easier to solve than triangular systems, namely a diagonal system. The method is very similar to Gaussian Elimination. It is just a bit more expensive.

Back to original system. Step 1 must transform:

\[
\begin{bmatrix}
2 & 4 & 4 & 2 \\
1 & 3 & 1 & 1 \\
1 & 5 & 6 & -6
\end{bmatrix}
\]

into:

\[
\begin{bmatrix}
x & x & x & x \\
x & 0 & x & x \\
x & 0 & x & x
\end{bmatrix}
\]

Step 2:

\[
\begin{bmatrix}
2 & 4 & 4 & 2 \\
0 & 1 & -1 & 0 \\
0 & 3 & 4 & -7
\end{bmatrix}
\]

into:

\[
\begin{bmatrix}
x & 0 & 0 & x \\
0 & x & 0 & x \\
0 & 0 & x & x
\end{bmatrix}
\]

There is now a third step:

To transform:

\[
\begin{bmatrix}
2 & 0 & 8 & 2 \\
0 & 1 & -1 & 0 \\
0 & 3 & 4 & -7
\end{bmatrix}
\]

into:

\[
\begin{bmatrix}
x & 0 & 0 & x \\
0 & 1 & 0 & 1 \\
0 & 0 & 7 & -7
\end{bmatrix}
\]

Solution: \(x_3 = -1; x_2 = -1; x_1 = 5\)

ALGORITHM: 3. Gauss-Jordan elimination

1. For \(k = 1 : n\) Do:
2. For \(i = 1 : n\) and if \(i \neq k\) Do:
3. \(\text{piv} := \frac{a_{ik}}{a_{kk}}\)
4. For \(j := k + 1 : n + 1\) Do:
5. \(a_{ij} := a_{ij} - \text{piv} \times a_{kj}\)
6. \(\text{End}\)
7. \(\text{End}\)

Operation count:

\[
T = \sum_{k=1}^{n} \sum_{i=1}^{n-1} \left[ 1 + \sum_{j=k+1}^{n+1} 2 \right] = \sum_{k=1}^{n-1} \sum_{i=1}^{n-1} (2(n - k) + 3) = \cdots
\]

Complete the above calculation. Order of the cost? How does it compare with Gaussian Elimination?
function x = gaussj (A, b)
%---------------------------------------------------
% function x = gaussj (A, b)
% solves A x = b by Gauss-Jordan elimination
%---------------------------------------------------
n = size(A,1) ;
A = [A,b];
for k=1:n
    for i=1:n
        if (i ~= k)
            piv = A(i,k) / A(k,k) ;
            A(i,k+1:n+1) = A(i,k+1:n+1) - piv*A(k,k+1:n+1);
        end
    end
end
x = A(:,n+1) ./ diag(A) ;

Gaussian Elimination: Partial Pivoting
Consider again Gaussian Elimination for the linear system
\[
\begin{align*}
2x_1 + 2x_2 + 4x_3 &= 2 \\
x_1 + x_2 + x_3 &= 1 \\
x_1 + 4x_2 + 6x_3 &= -5
\end{align*}
\]
Or:
\[
\begin{bmatrix}
2 & 2 & 4 & 2 \\
1 & 1 & 1 & 1 \\
1 & 4 & 6 & -5
\end{bmatrix}
\]
row_2 := row_2 - \frac{1}{2} \times row_1; 
row_3 := row_3 - \frac{1}{2} \times row_1:
\[
\begin{bmatrix}
2 & 2 & 4 & 2 \\
0 & 0 & -1 & 0 \\
1 & 4 & 6 & -5
\end{bmatrix}
\]
\[
\begin{bmatrix}
2 & 2 & 4 & 2 \\
0 & 0 & -1 & 0 \\
0 & 3 & 4 & -6
\end{bmatrix}
\]
\[\Rightarrow\]

Pivot \(a_{22}\) is zero. Solution:
permute rows 2 and 3:
\[
\begin{bmatrix}
2 & 2 & 4 & 2 \\
0 & 3 & 4 & -6 \\
0 & 0 & -1 & 0
\end{bmatrix}
\]

Gaussian Elimination with Partial Pivoting
function x = gaussp (A, b)
%---------------------------------------------------
% function x = gaussp (A, b)
% solves A x = b by Gaussian elimination with
% partial pivoting/
%---------------------------------------------------
n = size(A,1) ;
A = [A,b];
for k=1:n-1
    [t, ip] = max(abs(A(k:n,k)));
    ip = ip+k-1 ;
    temp = A(k,k:n+1) ;
    A(k,k:n+1) = A(ip,k:n+1);
    A(ip,k:n+1) = temp;
    for i=k+1:n
        piv = A(i,k) / A(k,k) ;
        A(i,k+1:n+1) = A(i,k+1:n+1) - piv*A(k,k+1:n+1);
    end
end
x = backsolv(A,A(:,n+1));

Partial Pivoting
- General situation:
  Always permute row \(k\) with row \(l\) such that
  \[
  |a_{lk}| = \max_{i=k,...,n} |a_{ik}|
  \]
- More 'stable' algorithm.
Pivoting and permutation matrices

A permutation matrix is a matrix obtained from the identity matrix by permuting its rows.

For example, for the permutation \( \pi = \{3, 1, 4, 2\} \), we obtain

\[
P = \begin{pmatrix}
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0
\end{pmatrix}
\]

Important observation: the matrix \( PA \) is obtained from \( A \) by permuting its rows with the permutation \( \pi \):\( (PA)_{i,:} = A_{\pi(i,:),} \).

Example:
To obtain \( \pi = \{3, 1, 4, 2\} \) from \( \pi = \{1, 2, 3, 4\} \), we need to swap \( \pi(2) \leftrightarrow \pi(3) \), then \( \pi(3) \leftrightarrow \pi(4) \), and finally \( \pi(1) \leftrightarrow \pi(2) \). Hence:

\[
P = \begin{pmatrix}
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0
\end{pmatrix} = E_{1,2} \times E_{3,4} \times E_{2,3}
\]

Notes: (1) \( E_{ij} = E_{ji} \) and (2) \( M_{j}^{-1} \times E_{k+1} = E_{k+1} \times \tilde{M}_{j}^{-1} \) for \( k \geq j \), where \( \tilde{M}_{j} \) has a permuted Gauss vector:

\[
(I + v^{(j)}e_{j}^{T})E_{k+1} = E_{k+1}(I + E_{k+1}v^{(j)}e_{j}^{T})
\]

\[
\equiv E_{k+1}(I + \tilde{v}^{(j)}e_{j}^{T})
\]

\[
\equiv E_{k+1}\tilde{M}_{j}
\]

Here we have used the fact that above row \( k+1 \), the permutation matrix \( E_{k+1} \) looks just like an identity matrix.

What is the matrix \( PA \) when

\[
P = \begin{pmatrix}
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0
\end{pmatrix}
\]

\[
A = \begin{pmatrix}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 0 & -1 & 2 \\
-3 & 4 & -5 & 6
\end{pmatrix}
\]

At each step of G.E. with partial pivoting:

\[
M_{k+1}E_{k+1}A_{k} = A_{k+1}
\]

where \( E_{k+1} \) encodes a swap of row \( k+1 \) with row \( l > k+1 \).

Notes: (1) \( E_{i}^{-1} = E_{i} \) and (2) \( M_{j}^{-1} \times E_{k+1} = E_{k+1} \times \tilde{M}_{j}^{-1} \) for \( k \geq j \), where \( \tilde{M}_{j} \) has a permuted Gauss vector:

\[
(I + v^{(j)}e_{j}^{T})E_{k+1} = E_{k+1}(I + E_{k+1}v^{(j)}e_{j}^{T})
\]

\[
\equiv E_{k+1}(I + \tilde{v}^{(j)}e_{j}^{T})
\]

\[
\equiv E_{k+1}\tilde{M}_{j}
\]

Here we have used the fact that above row \( k+1 \), the permutation matrix \( E_{k+1} \) looks just like an identity matrix.
Result:

\[ A_0 = E_1 M^{-1}_1 A_1 \]
\[ = E_1 M^{-1}_1 E_2 M^{-1}_2 A_2 = E_1 E_2 \tilde{M}^{-1}_1 M^{-1}_2 A_2 \]
\[ = E_1 E_2 \tilde{M}^{-1}_1 M^{-1}_2 E_3 M^{-1}_3 A_3 \]
\[ = E_1 E_2 E_3 \tilde{M}^{-1}_1 M^{-1}_2 M^{-1}_3 A_3 \]
\[ = \ldots \]
\[ = E_1 \cdots E_{n-1} \times \tilde{M}^{-1}_1 M^{-1}_2 M^{-1}_3 \cdots \tilde{M}^{-1}_{n-1} \times A_{n-1} \]

In the end

\[ PA = LU \text{ with } P = E_{n-1} \cdots E_1 \]

**Error Analysis**

If no zero pivots are encountered during Gaussian elimination (no pivoting) then the computed factors \( \hat{L} \) and \( \hat{U} \) satisfy

\[ \hat{L} \hat{U} = A + H \]

with

\[ |H| \leq 3(n-1) \times u \left( |A| + |\hat{L}| |\hat{U}| \right) + O(u^2) \]

Solution \( \hat{x} \) computed via \( \hat{L} \hat{y} = b \) and \( \hat{U} \hat{x} = \hat{y} \) is s.t.

\[ (A + E) \hat{x} = b \text{ with } |E| \leq n u (3|A| + 5 |\hat{L}| |\hat{U}|) + O(u^2) \]

“Backward” error estimate.

|\( \hat{L} \) and |\( \hat{U} \) are not known in advance – they can be large.

What if partial pivoting is used?

Permutations introduce no errors. Equivalent to standard LU factorization on matrix \( PA \).

|\( \hat{L} \) is small since \( l_{ij} \leq 1 \). Therefore, only \( U \) is “uncertain”

In practice partial pivoting is “stable” – i.e., it is highly unlikely to have a very large \( U \).

Read Lecture 22 of Text (especially last 3 subsections) about stability of Gaussian Elimination with partial pivoting.