PRACTICE 9: Linear Systems I: Direct Methods

1 Linear Systems Introduction

We are interested in solving linear systems of the form:

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

where $A \in \mathbb{R}^{m\times n}$, $\mathbf{x} \in \mathbb{R}^n$, and $\mathbf{b} \in \mathbb{R}^m$.

- The system $A\mathbf{x} = \mathbf{b}$ has at least one solution if and only if $\mathbf{b} \in \mathrm{Col}(A)$.
- It has a unique solution if $\mathbf{b} \in \mathrm{Col}(A)$ and $\ker(A) = \{\mathbf{0}\}$.

  For square matrices ($A \in \mathbb{R}^{n\times n}$), uniqueness occurs if and only if $\mathrm{rank}(A) = n$ (i.e., A is invertible or regular matrix).

  In such cases, the solution is $\mathbf{x} = A^{-1}\mathbf{b}$.
- In the case of a purely overdetermined system, we still have the previuos case for the system $B\mathbf{x}_{\mathrm{ls}} = \mathbf{c}$, where:
$$\underbrace{A^T A}_{B}\mathbf{x}_{\mathrm{ls}} = \underbrace{A^T\mathbf{b}}_{\mathbf{c}}$$

However, in practice we want to avoid computing A^{-1} directly. Instead, we aim to find a solution numerically.

Types of Methods:

- Direct Methods
- Iterative Methods

2 Gauss-Jordan

Gauss-Jordan reduces the matrix to the identity:

$$[A|\mathbf{b}] \sim_{GJ} [I_n|\mathbf{s}],$$

thus $\mathbf{s} = A^{-1}\mathbf{b}$. This method can also be used to compute A^{-1} by:

$$[A|I_n] \sim [I_n|A^{-1}].$$

To solve the system $A\mathbf{x} = \mathbf{b}$, the augmented matrix is formed:

$$[A \mid \mathbf{b}] \sim_{GJ} [I_n \mid \mathbf{x}]$$

This method can also be used to compute the inverse of A by applying it to the augmented matrix:

$$[A \mid I_n] \sim [I_n \mid A^{-1}]$$

Algorithm 1 Gaussian Elimination with Partial Pivoting

Input: Square matrix $A \in \mathbb{R}^{n\times n}$, right-hand side $\mathbf{b} \in \mathbb{R}^n$
Goal: Solve $A\mathbf{x} = \mathbf{b}$ by reducing to reduced row echelon form (RREF)
Form the augmented matrix $[A \mid \mathbf{b}]$
for $p = 1$ to n **do**
  Find row $k \geq p$ such that $|a_{kp}|$ is maximal and swap row p with row k
  Normalize row p to make the pivot equal to 1
  for each row $i \neq p$ **do**
    Eliminate entry in column p: subtract a multiple of row p from row i
  end for
end for
Output: $[I_n \mid \mathbf{x}]$

Example: Solve the system

$$\begin{cases} x + y + z = 6 \\ 2y + 5z = -4 \\ 2x + 5y - z = 27 \end{cases}$$

Matrix form:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 5 \\ 2 & 5 & -1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 6 \\ -4 \\ 27 \end{bmatrix}$$

MATLAB Code

```
A = [1 1 1; 0 2 5; 2 5 -1];
b = [6; -4; 27];
Ab = [A b];
n = size(Ab, 1);

for i = 1:n
% Pivoting
[~, k] = max(abs(Ab(i:n, i)));
k = k + i - 1;
Ab([i k], :) = Ab([k i], :);

% Normalize pivot row
Ab(i, :) = Ab(i, :) / Ab(i, i);

% Eliminate all other rows
for j = 1:n
if j ~= i
Ab(j, :) = Ab(j, :) - Ab(j, i) * Ab(i, :);
end
end
end

x = Ab(:, end)
```

Solution:

$$\mathbf{x} = \begin{pmatrix} 5 \\ 3 \\ -2 \end{pmatrix}$$

3 Gauss Elimination

Gauss elimination transforms the matrix A into an upper triangular matrix U, such that:

$$A\mathbf{x} = \mathbf{b} \Rightarrow U\mathbf{x} = \mathbf{c},$$

where the system $U\mathbf{x} = \mathbf{c}$ is solved by back substitution.

Upper-triangular Linear Systems:

Given the system

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n-1} & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n-1} & a_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{n-1\,n-1} & a_{n-1\,n} \\ 0 & 0 & \cdots & 0 & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix}$$

it is easy to obtain the solution as

$$x_n = \frac{b_n}{a_{nn}} \ ; \ x_k = \frac{1}{a_{kk}} \left(b_k - \sum_{j=k+1}^{n} a_{kj} x_j \right), \ k = n-1, \ n-2, \ \ldots, \ 1$$

MATLAB Code: Backward Sustitution

```
function x=back_substitution(A,b)
%     Function x=back_substitution(A,b)
%     Solve the system Ax=b, where A is an nxn upper-triangular
%  nonsingular matrix
% INPUT ARGUMENTS:
%   A ........ Coefficient matrix
%   b ........ Vector of constants terms
% OUTPUT ARGUMENT:
%   x ........ Numerical approximation for the solution

x=[]; [m n]=size(A);
if m~=n, disp('ERROR: the matrix is NOT a square matrix'), return, end
if m~=length(b), disp('ERROR: dimensions do NOT agree'), return, end
if isequal(A,triu(A))==0
disp('ERROR: the matrix is NOT an upper-triangular matrix')
return
end
if min(abs(diag(A)))==0
disp('ERROR: the matrix is singular')
return
end
x=zeros(n,1); x(n)=b(n)/A(n,n);
for k=n-1:-1:1
x(k)=(b(k)-A(k,k+1:n)*x(k+1:n))/A(k,k);
end
```

Lower-triangular Linear Systems:

Given the system

$$\begin{pmatrix} a_{11} & 0 & \cdots & 0 & 0 \\ a_{21} & a_{22} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n-1\,1} & a_{n-1\,2} & \cdots & a_{n-1\,n-1} & 0 \\ a_{n1} & a_{n2} & \cdots & a_{n\,n-1} & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix}$$

it is easy to obtain the solution as

$$x_1 = \frac{b_1}{a_{11}} \; ; \; x_k = \frac{1}{a_{kk}} \left(b_k - \sum_{j=1}^{k-1} a_{kj} x_j \right), \; k = 2, \; 3, \; \ldots, \; n$$

MATLAB Code: Forward Sustitution

```
function x=forward_substitution(A,b)
%     Function x=forward_substitution(A,b)
%     Solve the system Ax=b, where A is an nxn lower-triangular
% nonsingular matrix
% INPUT ARGUMENTS:
%   A ........ Coefficient matrix
%   b ........ Vector of constants terms
% OUTPUT ARGUMENT:
%   x ........ Numerical approximation for the solution
x=[]; [m n]=size(A);
if m~=n, disp('ERROR: the matrix is NOT a square matrix'), return, end
if m~=length(b), disp('ERROR: dimensions do NOT agree'), return, end
if isequal(A,tril(A))==0
disp('ERROR: the matrix is NOT a lower-triangular matrix'), return
end
if min(abs(diag(A)))==0, disp('ERROR: singular matrix'), return, end
x=zeros(n,1); x(1)=b(1)/A(1,1);
for k=2:n
x(k)=(b(k)-A(k,1:k-1)*x(1:k-1))/A(k,k);
end
```

Gaussian elimination transforms a linear system by eliminating the entries below the main diagonal, using the diagonal elements as *pivots*.

If a pivot element in row p and column p is zero, it cannot be used for elimination. In that case, a row $k > p$ is found where the entry in column p is nonzero, and rows p and k are swapped. This is called **pivoting**.

Since computers use finite-precision arithmetic, rounding errors accumulate with each operation. A pivoting strategy helps minimize those errors by choosing the best pivot available.

Partial Pivoting Strategy:

To improve numerical stability, partial pivoting selects the pivot as the largest (in absolute value) entry in column p, from rows p to n:

$$|a_{kp}| = \max\{|a_{pp}|, |a_{p+1,p}|, \ldots, |a_{np}|\}$$

Then, row p is swapped with row k. This ensures that the pivot is as large as possible, reducing the effect of rounding errors during elimination.

Algorithm 2 Gaussian Elimination with Partial Pivoting

Given a system $A\mathbf{x} = \mathbf{b}$ with $A \in \mathbb{R}^{n\times n}$
for $p = 1$ to $n-1$ **do**
  if $a_{pp} = 0$ **or** $|a_{pp}|$ is not the largest in column p **then**
    Find row $k \geq p$ such that $|a_{kp}| = \max\{|a_{ip}|$ for $i = p, \ldots, n\}$
    Swap rows p and k in both A and $\mathbf{b}$ ▷ Partial Pivoting
  end if
  for $i = p+1$ to n **do**
    Compute multiplier $m = a_{ip}/a_{pp}$
    Update row i: $a_{ij} \leftarrow a_{ij} - m \cdot a_{pj}$ for $j = p, \ldots, n$
    Update right-hand side: $b_i \leftarrow b_i - m \cdot b_p$
  end for
end for
Solve the upper triangular system $U\mathbf{x} = \mathbf{c}$ by back substitution

MATLAB Code

```
function x = gauss_pivoting(A, b)
% Solve Ax = b using Gaussian elimination with partial pivoting

[m, n] = size(A);
if m ~= n, error('Matrix must be square'); end
if length(b) ~= n, error('Dimension mismatch'); end

for k = 1:n
    [~, j] = max(abs(A(k:n, k))); j = j + k - 1;
    if A(j, k) == 0
        error('Matrix is singular');
    end
    if j ~= k
        A([k j], :) = A([j k], :);
        b([k j]) = b([j k]);
    end
    for i = k+1:n
        factor = A(i, k) / A(k, k);
        A(i, k:n) = A(i, k:n) - factor * A(k, k:n);
        b(i) = b(i) - factor * b(k);
    end
end

x = back_substitution(A, b);
```

Example 3.1 *Given the system* $\begin{cases} 10^{-15}x + y = 1 \\ x + y = 2 \end{cases}$,

a) Denote $\mathbf{s}$ *and* $\mathbf{s_1}$ *the solutions obtained with* `gauss` *and* `gauss_pivoting`*, respectively.*

b) Find the relative error (in percentage) made by approximating $\mathbf{s_1}$ *with* $\mathbf{s}$ *and the absolute* *errors made by approximating every component of* $\mathbf{s}$ *with the corresponding component of* $\mathbf{s_1}$.

4 LU Factorization

LU factorization expresses a square matrix $A \in \mathbb{R}^{n \times n}$ as the product of two triangular matrices:

$$A = LU$$

- L: lower triangular matrix (typically with ones on the diagonal)
- U: upper triangular matrix

This decomposition simplifies the solution of a linear system $A\mathbf{x} = \mathbf{b}$. Once we have $A = LU$, we solve the system in two steps:

$$LU\mathbf{x} = \mathbf{b} \Rightarrow \begin{cases} L\mathbf{y} = \mathbf{b} & \text{(Forward substitution)} \\ \\ U\mathbf{x} = \mathbf{y} & \text{(Back substitution)} \end{cases}$$

Pivoting

Not all matrices can be factorized as $A = LU$ without rearranging rows. To improve numerical stability, we use *partial pivoting*:

$$PA = LU$$

where P is a permutation matrix that reorders the rows of A.

Example 4.1 *Solve the linear system*

$$\begin{cases} 2x - y - 2z = -2 \\ -4x + 6y + 3z = 9 \\ -4x - 2y + 8z = -5 \end{cases}$$

using the triangular factorization LU.

MATLAB Example

LU Factorization in MATLAB

```
A = [2 -1 -2; -4 6 3; -4 -2 8];
[L, U, P] = lu(A);

% Solve A*x = b
b = [-2; 9; -5];
y = L \ (P * b);   % Forward substitution
x = U \ y;         % Back substitution
```

Example 4.2 *Solve the linear system*

$$\begin{cases} x_2 + x_3 = 5 \\ x_1 + x_3 = 4 \\ x_1 + x_2 = 3 \end{cases}$$

using the triangular factorization LU*.*

Solution: We introduce the coefficient matrix and the vector of constant terms and find L, U y P using the command `lu`

so, if we have the initial linear system $A\,\mathbf{x} = \mathbf{b}$, we multiply it by P and we obtain

$$PA\,\mathbf{x} = P\,\mathbf{b} \implies LU\,\mathbf{x} = \mathbf{b}_1, \text{ with } \mathbf{b}_1 = P\mathbf{b}$$

and, doing $U\,\mathbf{x} = \mathbf{y}$, we reduce the problem to solve two triangular linear systems $L\,\mathbf{y} = \mathbf{b}_1$, which is a lower-triangular linear system, and then $U\,\mathbf{x} = \mathbf{y}$, which is an upper-triangular linear system.

5 Symmetric Matrices

A matrix A is symmetric if:

$$A = A^T \quad \Rightarrow \quad a_{ij} = a_{ji} \quad \forall i, j = 1, \ldots, n$$

Properties

1. $a_{ij} = a_{ji}$
2. All symmetric matrices admit diagonalization:
$$A = QDQ^T \quad \text{with } \lambda_i \in \mathbb{R}$$
and the eigenvalues are real.
3. The eigenvectors form an orthonormal basis:
$$Q = \begin{bmatrix} \vec{v}_1 & \vec{v}_2 & \ldots & \vec{v}_n \end{bmatrix} \quad \text{with } A = QDQ^T$$

Spectral Decomposition

If A is symmetric and full-rank matrix, then D is an **invertible** matrix (since all the eigenvalues are different from zero: $\lambda_k \neq 0, \forall k = 1, ..., n$)

$$QDQ^T\mathbf{x} = \mathbf{b} \Rightarrow DQ^T\mathbf{x} = Q^T\mathbf{b} \Rightarrow \mathbf{x} = QD^{-1}Q^T\mathbf{b}$$

$$\boxed{\mathbf{x} = QD^{-1}Q^T\mathbf{b}}$$

6 Symmetric and Positive Definite Matrices

Let $A \in \mathbb{R}^{n\times n}$ be a real square matrix. We say that A is:

- **Symmetric** if $A^T = A$,
- **Positive definite** if $\mathbf{x}^T A\mathbf{x} > 0$ for all $\mathbf{x} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$

Characterizations

1. $A = A^T$
2. For all $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{x}^T A\mathbf{x} > 0$
3. All eigenvalues $\lambda_k \in \mathbb{R}^+$

Spectral Decomposition and Cholesky Descomposition

If A is symmetric and positive definite, then it admits a spectral decomposition:

$$A = QDQ^T = QD^{1/2}D^{1/2}Q^T = LL^T$$

where:

- Q is orthogonal: $Q^T = Q^{-1}$
- D is diagonal with real positive entries
- $L = QD^{1/2}$ is lower triangular
- $L^T = D^{1/2}Q^T$

7 Cholesky Decomposition

Cholesky decomposition is a special case of LU decomposition for **symmetric and positive definite** matrices.

In this case, A can be decomposed as:

$$A = LL^T,$$

where L is a lower triangular matrix.

To solve $A\mathbf{x} = \mathbf{b}$:

$$\begin{aligned} L\mathbf{c} &= \mathbf{b}, \\ L^T\mathbf{x} &= \mathbf{c}. \end{aligned}$$

In **MATLAB**, we can find the matrix L executing `chol(A)`.

Example 7.1 *Solve the linear system*

$$\begin{cases} x_1 + x_2 + x_3 = 0 \\ x_1 + 2x_2 + 2x_3 = -1 \\ x_1 + 2x_2 + 6x_3 = 1 \end{cases}$$

using the Cholesky factorization and making sure it is feasible in advance.

Solution:

We check that the matrix is symmetric

We verify if it is positive-definite by calculating all the leading principal minors of A and checking if they are positive:

```
for i=1:3
D(i)=det(A(1:i,1:i));
end
```

Note: If A is big, it would be better to do:

```
min(eig(A))
```

$$A = L^t \cdot L \quad A\,\mathbf{x} = \mathbf{b} \quad \leftrightarrows \quad L^t \cdot L\,\mathbf{x} = \mathbf{b}$$

denoting $L\,\mathbf{x} = \mathbf{y}$, we reduce the problem to two triangular linear systems: a lower-triangular linear system $L^t\,\mathbf{y} = \mathbf{b}$ and an upper-triangular linear system $L\,\mathbf{x} = \mathbf{y}$, therefore:

```
L=chol(A)
y=forward_substitution(L.',b)
x=back_substitution(L,y)
```

8 The command \

In MATLAB, the solution of the linear system $A\,\mathbf{x} = \mathbf{b}$ can be obtained simply by running `A\b`. Moreover, with this command, MatLab solves the linear system using the best algorithm for each matrix. For example, if the matrix is upper-triangular or lower-triangular, then MatLab uses back substitution or forward substitution, respectively; if the matrix is positive-definite then MatLab uses Cholesky factorization, etc.

Example 8.1 *Solve the previous linear system using the command* \ *.*