Notes for Numerical Analysis
Math 4445
by
S. Adjerid
Virginia Polytechnic Institute
and State University

(A Rough Draft)

Contents

Chapter 1

Solving Linear Systems

1.1 Review

We start with a review of vectors and matrices and matrix-matrix and matrix-vector multiplication and other matrix and vector operations. **Vectors operations and norms:**

Let $\mathbf{x} = [x_1, x_2, \cdots, x_n]^t$ and $\mathbf{y} = [y_1, y_2, \cdots, y_n]^t$

Then

$\mathbf{x} + \mathbf{y} = [x_1 + y_1, x_2 + y_2, \cdots, x_n + y_n]$

$\lambda\mathbf{x} = [\lambda x_1, \lambda x_2, \cdots, \lambda x_n]^t$

```
>>x = [1,2,3,-5];
>>y = [3,5,2,1];
>> x+y
      [4 7 5 -4]
>> 2*x
   [2,4,6,-10]
```

Norms of vectors

$||\mathbf{x}||_1 = |x_1| + |x_2| + \cdots + |x_n|$

$||\mathbf{x}||_\infty = \max(|x_1|, |x_2|, \cdots, |x_n|)$

$||\mathbf{x}||_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$

$||\mathbf{x}||_p = (x_1^p + x_2^p + \cdots + x_n^p)^{1/p}$

Properties of norms

(i) $||\mathbf{x}|| \geq 0$

(ii) $||\mathbf{x}|| = 0 \;\Leftrightarrow\; \mathbf{x} = 0$

(iii) $||\mathbf{x} + \mathbf{y}|| \leq ||\mathbf{x}|| + ||\mathbf{y}||$

(iv) $||\lambda\mathbf{x}|| = |\lambda|||\mathbf{x}||$

Matrix operations and norms

Let us consider two matrices

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \vdots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix} \quad B = \begin{bmatrix} b_{11} & \cdots & b_{1m} \\ \vdots & \vdots & \vdots \\ b_{n1} & \cdots & b_{nm} \end{bmatrix}.$$

The matrices A and B have n rows and m columns. If $n = m$, A and B are square matrices, otherwise they are rectangular matrices.

We define the following matrix operations

$C = A + B$, $c_{ij} = a_{ij} + b_{ij}$
$C = \lambda A$, $c_{ij} = \lambda a_{ij}$

$C = AB$, $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$
$AB \neq BA$ in general
$C = A^t$, $c_{ij} = a_{ji}$
$A \neq A^t$ in general

Determinant of square matrices

for $n = 1$, $det(A) = a_{11}$
for $n = 2$, $det(A) = a_{11}a_{22} - a_{21}a_{12}$
for $n > 2$, $det(A) = \sum_{j=1}^{n} a_{ij}(-1)^{i+j} M_{ij}$
where M_{ij} is the determinant of the matrix obtained by deleting the i^{th} row and j^{th} column

$det(AB) = det(A)det(B)$

$det(A^t) = det(A)$
$det(\lambda A) = \lambda^n det(A)$
$det(A) = 0$ if and only if A is singular, i.e., A does not have an inverse matrix

A has an inverse A^{-1} if and only if $AA^{-1} = A^{-1}A = I$
where I is the $n \times n$ identity matrix.

Theorem 1.1.1. *If A is an $n \times n$ matrix, the following statements are equivalent*

- *A is nonsingular*
- *The inverse A^{-1} exists*
- *$Ax = 0$ has the unique solution $x = 0$*
- *$Ax = b$ has a unique solution*
- *$det(A) \neq 0$*
- *$\lambda = 0$ is not an eigenvalue for A*

Matrix norms:

Definition 1. *Given a vector norm* $||.||_p$, $p = 1, 2, \infty$, *we define the subordinate matrix norm as*

$$||A||_p = \max_{x \neq 0} \frac{||Ax||_p}{||x||_p}.$$

Theorem 1.1.2. *Let* **A** *be an* $n \times n$ *matrix then*

$$||A||_\infty = \max_{i=1,\cdots,n} \sum_{j=1}^{n} |a_{i,j}|$$

$$||A||_1 = \max_{j=1,\cdots,n} \sum_{i=1}^{n} |a_{i,j}|$$

$$||A||_2 = \sqrt{\rho(A^t A)} = \max i = 1, \cdots, n|\sigma_i|$$

Proof. let $Ax = y$, where $y_i = \sum_{i=1}^{n} a_{i,j} x_j$.

$|y_i| \leq (\sum_{i=1}^{n} |a_{i,j}|)||x||_\infty$

$$||y||_\infty = ||Ax||_\infty \leq (\max_{i=1,\cdots,n} \sum_{j=1}^{n} |a_{i,j}|)||x||_\infty$$

thus

$$||A||_\infty \leq \max_{i=1,\cdots,n} \sum_{j=1}^{n} |a_{i,j}|$$

Let k such that $|y_k| = ||y||_\infty$ and select a vector $\hat{x}$ such that $\hat{x}_j = sign(a_{k,j})$. Then

$$||A||_\infty \geq \frac{||A\hat{x}||_\infty}{||\hat{x}||_\infty} \geq \sum_{j=1}^{n} |a_{i,j}|.$$

The assertion (ii) can be established following the same line of reasoning. The l_2 norm is established by looking at

$$||A||_2 = \max_{x \neq 0} \sqrt{\frac{|(Ax, Ax)|}{|(x, x)|}}$$

Noting that $(Ax, Ax) = x^t A^t Ax$ where $A^t A$ is symmetric positive which can be factored as

$$A^t A = Q^t D Q, \quad Q^t Q = I, \quad D = diag(\sigma_1^2, \cdots, \sigma_n^2)$$

Thus

$$||A||_2 = \max_{x \neq 0} \sqrt{\frac{y^t D y}{y^t y}} \leq \max_{i=1,n} |\sigma_i| = |\sigma_k|$$

Selecting $\hat{x} = v_k$ the eigenvector associated with σ_k we show that $||A||_2 \geq |\sigma_k|$. This concludes the proof. □

1.2 Direct methods for solving linear systems

We start with an example

$$\begin{array}{lcl} 2x_1 + 3x_2 - 5x_3 & = & 1 \\ 2x_1 - x_2 + 6x_3 + x_4 & = & \text{-1} \\ x_1 + 5x_2 - 16x_3 - 8x_4 & = & 13 \\ x_1 + x_2 + x_4 & = & 13 \end{array}$$

The matrix formulation can be written as

$$\begin{bmatrix} 2 & 4 & -5 & 0 \\ 2 & -1 & 6 & 1 \\ 1 & 5 & -16 & -8 \\ 1 & 1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \quad = \quad \begin{bmatrix} 1 \\ -1 \\ 13 \\ 3 \end{bmatrix}$$

An upper-triangular system:

$$\begin{bmatrix} 3 & 1 & 1 & 0 \\ 0 & -2 & 0 & 1 \\ 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & 3 \end{bmatrix} \quad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \quad = \quad \begin{bmatrix} 4 \\ 0 \\ 4 \\ 8 \end{bmatrix}$$

Use the backward substitution to solve the system starting from the last equation

1. Equation 4, $4x_4 = 8$ yields $\boxed{x_4 = 2}$
2. Equation 3, $3x_3 + 2x_4 = 4$ with $x_4 = 2$ yields $\boxed{x_3 = 0}$
3. Equation 2, $-2x_2 + x_4 = 0$, yields $\boxed{x_2 = 1}$
4. Equation 1, $3x_1 + x_2 + x_3 = 4$ yields $\boxed{x_1 = 1}$.

A lower-triangular system

$$\begin{bmatrix} 5 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -2 & 3 & 0 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -5 \\ -4 \\ 1 \\ -4 \end{bmatrix}$$

Use forward substitution to solve the system

1. Equation 1, $5x - 1 = -5$ yields $\boxed{x_1 = -1}$
2. Equation 2, $2x_1 + x_2 = -4$ yields $\boxed{x - 2 = -1}$
3. Equation 3, $-x_1 + x_3 = 1$ yields $\boxed{x_3 = 0}$
4. Equation 4, $-2x_1 + 3x_2 - 4x_4 = -4$, yields $\boxed{x_4 = 0}$

Matlab program for backward substitution

```
function [x]=backward(U,b)
%Input: U an upper triangular nxn matrix
%        b a vector of length n
%Ouput: x a vector of length n solution to Ux = b
%
[n,m] = size(b);
x(n,:) = b(n,:)/U(n,n);
for i = n-1:-1:1
   x(i,:) = (b(i,:) - U(i,i+1:n)*x(i+1:n,:)')/U(i,i);
end
```

Matlab program for forward substitution

```
function [x]=forward(L,b)
%input: L a lower triangular n x n matrix
%       b a nxm matrix
%output: x a n x m matrix solution of Lx=b
%
[n,m] = size(b);
x(1,:) = b(1,:)/L(1,1);
for i = 2:n
   x(i,:) = (b(i,:) - L(i,1:i-1)*x(1:i-1,:))/L(i,i);
end
```

1.2.1 Naive Gaussian elimination

Gaussian elimination helps transform a general system $Ax = b$ into an equivalent, i.e., have the same solution as $Ax = b$, upper-triangular system $Ux = \tilde{b}$. using the operations

(i) $E_j \leftrightarrow E_j - \lambda_j E_i,\ \lambda_j \neq 0$
(ii)$E_j \leftrightarrow E_i$ (row interchanges)

General form

$$\begin{array}{rcl} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &=& b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &=& b_2 \\ \vdots & & \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &=& b_n \end{array}$$

into an upper triangular system

$$\begin{array}{rcccl} u_{11}x_1 + & u_{12}x_2 + \cdots + & u_{1n}x_n &=& \tilde{b}_1 \\ & u_{22}x_2 + \cdots + & u_{2n}x_n &=& \tilde{b}_2 \\ & \ddots & & & \\ & & u_{nn}x_n &=& \tilde{b}_n \end{array}$$

Let us start with an example:

$$\begin{bmatrix} 1 & 1 & 0 & 3 \\ 2 & 1 & -1 & 1 \\ 3 & -1 & -1 & 2 \\ -1 & 2 & 3 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ -3 \\ 4 \end{bmatrix}$$

Step 1 in Gaussian elimination:

$$\begin{array}{rcl} E_1 & & \\ E_2 - 2E_1 & \rightarrow & E_2 \\ E_3 - 3E_1 & \rightarrow & E_3 \\ E_4 + E_1 & \rightarrow & E_4 \end{array} \begin{bmatrix} 1 & 1 & 0 & 3 \\ 0 & -1 & -1 & -5 \\ 0 & -4 & -1 & -7 \\ 0 & 3 & 3 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 4 \\ -7 \\ -15 \\ 8 \end{bmatrix}$$

Step 2 in Gaussian elimination:

$$\begin{array}{rcl} E_1 & & \\ E_2 & & \\ E_3 - 4E_2 & \rightarrow & E_3 \\ E_4 + 3E_2 & \rightarrow & E_4 \end{array} \begin{bmatrix} 1 & 1 & 0 & 3 \\ 0 & -1 & -1 & -5 \\ 0 & 0 & 3 & 13 \\ 0 & 0 & 0 & -13 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 4 \\ -7 \\ 13 \\ -13 \end{bmatrix}$$

Step 3 : will make the $a_{43} = 0$ which is already the case. In practice we just do it to avoid checking for zero entries.

Solve the resulting system using backward substitution

1. Equation 4, $-13x_4 = -13$ yields $\boxed{x_4 = 1}$
2. Equation 3, $3x_3 + 13x_4 = 13$ yields $\boxed{x_3 = 0}$
3. Equation 2, $-x_2 - x_3 - 5x_4 = -7$ yields $\boxed{x_2 = 2}$
4. Equation 1, $x_1 + x_2 + 3x_4 = 4$ yields $\boxed{x_1 = -1}$.

Matlab program for Gaussian elimination with no pivoting

```
function [x]=gausselim(A,b)
%Input: A is a nxn matrix
%       B is a nxm matrix
```

```
%Output: x is nxm matrix solution of Ax=b
%
[n,m] = size(b);
for i = 1:n
for j = i+1:n
   c = A(j,i)/A(i,i);
   A(j,i:n) = A(j,i:n) - c*A(i,i:n);
   b(j,1:m) = b(j,1:m) - c*b(i,1:m);
end
end
x = backward(A,b);
```

Breakdown of Gaussian elimination:

Consider the following example:

$$\begin{bmatrix} 1 & -3 & 0 \\ 1 & -3 & 5 \\ 2 & 1 & -1 \end{bmatrix} \quad \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad = \quad \begin{bmatrix} -1 \\ 4 \\ 4 \end{bmatrix}$$

We write the system using an augmented matrix formulation and perform Gaussian elimination

$$\left[\begin{array}{ccc:c} 1 & -3 & 0 & -1 \\ 1 & -3 & 5 & 4 \\ 2 & 1 & -1 & 4 \end{array}\right]$$

Step 1:

$$\begin{array}{rcl} E_1 & & \\ E_2 - E1 & \rightarrow & E_2 \\ E_3 - 2E_1 & \rightarrow & E_3 \end{array} \quad \left[\begin{array}{ccc:c} 1 & -3 & 0 & -1 \\ 0 & 0 & 5 & 5 \\ 0 & 7 & -1 & 6 \end{array}\right]$$

Step 2: Cannot use E_2 to eliminate x_2 from equation E_3. $E_3 - \lambda E_2 \rightarrow E_3$ will not eliminate x_2 from E_3. Instead we interchange the rows $E_2 \leftrightarrow E_3$ to obtain

$$\begin{matrix} E_1 & & \\ E_3 & \rightarrow & E_2 \\ E_2 & \rightarrow & E_3 \end{matrix} \begin{bmatrix} 1 & -3 & 0 & \vdots & -1 \\ 0 & 7 & -1 & \vdots & 6 \\ 0 & 0 & 5 & \vdots & 5 \end{bmatrix}$$

Now we use the backward substitution algorithm to compute the solution

$\boxed{x_1 = 2}$, $\boxed{x_2 = 1}$, $\boxed{x_3 = 1}$

1.2.2 Stability of Gaussian elimination and pivoting strategies

Let us consider the example:

$$\begin{bmatrix} \epsilon & 1 & \vdots & 1+\epsilon \\ 2 & 3 & \vdots & 5 \end{bmatrix}$$

with exact solution $x_1 = 1,\ x_2 = 1$.

Set $\epsilon = 10^{-8}$ and apply Gaussian elimination with four significant digits in the decimal system to obtain

$$\begin{matrix} E_1 & & \\ E_2 - E_1/\epsilon & \rightarrow & E_2 \end{matrix} \begin{bmatrix} \epsilon & 1 & \vdots & 1+\epsilon \\ 0 & 3-210^8 & \vdots & 5-210^8 \end{bmatrix}$$

Rounding-off leads to the system

$$\begin{bmatrix} \epsilon & 1 & \vdots & 1 \\ 0 & -210^8 & \vdots & -210^8 \end{bmatrix}$$

Solving leads to the incorrect answer $x_1 = 0$ and $x_2 = 1$.

Now if we interchange E_1 and E_2 prior to applying Gaussian elimination we obtain

$$\begin{matrix} E_2 & \rightarrow & E_1 \\ E_1 & \rightarrow & E_2 \end{matrix} \begin{bmatrix} 2 & 3 & \vdots & 5 \\ \epsilon & 1 & \vdots & 1+\epsilon \end{bmatrix}$$

Gaussian elimination leads to the system

$$\begin{bmatrix} 2 & 3 & \vdots & 5 \\ 0 & 1-\epsilon/2 & \vdots & 1-5\epsilon/2 \end{bmatrix}$$

Rounding-off yields

$$\begin{bmatrix} 2 & 3 & \vdots & 5 \\ 0 & 1 & \vdots & 1 \end{bmatrix}$$

Finally, we obtain the correct solution $x_1 = 1$, $x_2 = 1$.

In the first system the problem was caused by the division by a small number. The row interchange strategy that uses the largest possible pivot will help reduce of effect of round-off errors. Next we will discuss several pivoting strategies.

Partial pivoting

We consider the general system in augmented form

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & \vdots & b_1 \\ a_{11} & a_{12} & \cdots & a_{1n} & \vdots & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & b_n \\ a_{11} & a_{12} & \cdots & a_{1n} & \vdots & b_1 \end{bmatrix}$$

Step1:

(i)Select smallest p such that $|a_{p,1}| = \max_{i=1}^{n} |a_{i,1}|$

(ii) Interchange rows p and 1

(iii) Perform Gaussian elimination $E_j - \lambda_j E_1 \rightarrow E_j$, $\lambda_j = a_{j1}/a_{11}$ $j = 2, \cdots n$.

Step 2:

(i) Select p such that $|a_{p,2}| = \max_{j=2}^{n} |a_{j2}|$, where $a_{i,j}$ are the updated entries.

(ii) Interchange rows p and 2.

(iii) Perform elimination $E_j - \lambda_j E_2 \rightarrow E_j$, $\lambda_j = a_{j2}/a_{22}$ $j = 3, 4, \cdots n$.

Step i:

(i) Select p such that $|a_{p,i}| = \max_{j=i}^{n} |a_{ji}|$
(ii) Interchange rows p and i.
(iii) Perform elimination $E_j - \lambda_j E_i \rightarrow E_j$, $\lambda_j = a_{ji}/a_{ii}$ $j = i+1, \cdots n$.

Let us apply Gaussian with partial pivoting to the following system

$$\left[\begin{array}{cccc:c} 2 & 1 & 0 & \vdots & 3 \\ 1 & -1 & 4 & \vdots & -4 \\ 3 & -1 & -2 & \vdots & 4 \end{array}\right]$$

Step 1:

since $|a_{31}| > |a_{11}|, |a_{21}|$, interchange rows three and one to obtain

$$\left[\begin{array}{cccc:c} 3 & -1 & -2 & \vdots & 4 \\ 1 & -1 & 4 & \vdots & -4 \\ 2 & 1 & 0 & \vdots & 3 \end{array}\right]$$

Applying Gaussian elimination leads to

$$\begin{array}{ccc} E_1 & \rightarrow & E_1 \\ E_2 - E1/3 & \rightarrow & E_2 \\ E_3 - 2E1/3 & \rightarrow & E_3 \end{array} \quad \left[\begin{array}{cccc:c} 3 & -1 & -2 & \vdots & 4 \\ 0 & -2/3 & 14/3 & \vdots & -16/3 \\ 0 & 5/3 & 4/3 & \vdots & 1/3 \end{array}\right]$$

Step 2:

Since $|a_{32}| > |a_{22}|$, interchange rows two and three to obtain

$$\left[\begin{array}{cccc:c} 3 & -1 & -2 & \vdots & 4 \\ 0 & 5/3 & 4/3 & \vdots & 1/3 \\ 0 & -2/3 & 14/3 & \vdots & -16/3 \end{array}\right]$$

Using Gaussian elimination to obtain

$$\begin{array}{rcl} E_1 & \rightarrow & E_1 \\ E_2 & \rightarrow & E_2 \\ E_3 + 2E_2/5 & \rightarrow & E_3 \end{array} \left[\begin{array}{cccc} 3 & -1 & -2 & 4 \\ 0 & 5/3 & 4/3 & 1/3 \\ 0 & 0 & 26/5 & -26/5 \end{array}\right]$$

Now apply the backward substitution algorithm to compute the solution

$x_1 = 1$, $x_2 = 1$, $x_3 = -1$ and $IP = [3, 1, 2]$.

Remark: Partial pivoting can be fooled by a wrong scaling of the equations.

For instance, if we multiply the first equation in the system (**??**) by $10^4/\epsilon$ we obtain a new system

$$\left[\begin{array}{ccc} 10^4 & 10^4/\epsilon & (1+\epsilon)10^4/\epsilon \\ 2 & 3 & 5 \end{array}\right]$$

Partial pivoting results in using the first equation to perform Gaussian elimination. This will lead again to incorrect results for small values of ϵ.

$$\left[\begin{array}{ccc} 10^4 & 10^4/\epsilon & (1+\epsilon)10^4/\epsilon \\ 0 & 3-2/\epsilon & 5-2(1+\epsilon)/\epsilon \end{array}\right]$$

We may remedy this problem using scaled-column pivoting where we

(i) compute the weights $s_i = \max_{j=1}^{n} |a_{i,j}|$ from the original matrix
At i^{th} Gaussian step we (ii) compute the smallest integer p such that $|a_{p,i}|/s_i = \max_{j=i}^{n} |a_{j,i}|/s_j$
(iii) interchange rows p and i
(iii) Apply Gaussian elimination

Example:

$$\begin{bmatrix} 2 & 1 & 0 & :3 \\ 1 & -1 & 4 & :-4 \\ 3 & -1 & -2 & :4 \end{bmatrix}$$

Computes the scales $s_1 = 2$, $s_2 = 4$ and $s_3 = 3$.
Step 1: $max(2/2, 1/4, 3/3) = 1$, thus, $p = 1$ no row interchanges are required
Gaussian elimination:

$$\begin{bmatrix} 2 & 1 & 0 & :3 \\ 0 & -3/2 & 4 & :-11/2 \\ 0 & -5/2 & -2 & :-1/2 \end{bmatrix}$$

Step 2:
$max((3/2)/s_2, (5/2)/s_3)$ leads to $p = 3$, interchange E_2 and E_3

$$\begin{bmatrix} 2 & 1 & 0 & :3 \\ 0 & -5/2 & -2 & :-1/2 \\ 0 & -3/2 & 4 & :-11/2 \end{bmatrix}$$

Gaussian Elimination:

$$\begin{bmatrix} 2 & 1 & 0 & :3 \\ 0 & -5/2 & -2 & :-1/2 \\ 0 & 0 & 26/5 & :-26/5 \end{bmatrix}$$

Applying the backward substitution we get the solution is: $x_3 = -1$, $x_2 = 1$ and $x_1 = 1$.

Number of operations for Gaussian elimination solving $Ax = b$

$$\begin{bmatrix} a_{1,1} & \cdots & a_{1,n} & \vdots & a_{1,n+1} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ a_{n,1} & \cdots & a_{n,n} & \vdots & a_{n,n+1} \end{bmatrix}$$

Number of operations in step i:

$*/: \ (n-i) + (n-i) * (n-i+1) = (n-i) * (n-i+2)$
$\pm: \ (n-i) * (n-i+1)$

In order to compute the total number of operations we will need the following identities:

$$\sum_{j=1}^{m} 1 = m$$

$$\sum_{j=1}^{m} j = m(m+1)/2$$

$$\sum_{j=1}^{m} j^2 = m*(m+1)*(2m+1)/6 \text{ (the proof is obtained using induction)}$$

Therefore the total number of $*$ and $/$ is

$$\sum_{i=1}^{n-1}(n-i)(n-i+2) = (n^2+2n)\sum_{i=1}^{n-1} 1 \; -2(n+1)\sum_{i=1}^{n-1} i \; + \; +\sum_{i=1}^{n-1} i^2$$

$$= (2n^3+3n^2-5n)/6 \approx n^3/3, \text{ for } n >> 1.$$

The total number of $+$ and $-$

$$\sum_{i=1}^{n-1}(n-i)(n-i+1) = (n^3-n)/3 \approx n^3/3, \; n >> 1$$

The number of operation for backward substitution is as follows

$$*/ : \; \sum_{j=1}^{n} j = (n^2+n)/2$$
$$\pm : \; \sum_{j=1}^{n-1} j = (n^2-n)/2$$

The total solution cost:

$$*/ : \; n^3/3 + n^2 - n/3$$
$$+- : \; n^3/3 + n^2/2 - 5n/6$$

Additional cost for partial pivoting:

$\sum_{j=1}^{n-1} j = (n-1)n/2$

Remarks:

(i) Cost of a comparison is comparable to cost of an addition
(ii) Cost of a multiplication and division is greater that that of an addition(not true in matlab)
(iii) Gaussian elimination is the most expensive part of the computation
(iv) Backward substitution cost $\approx n^2/2$ $*$ / and $\approx n^2/2$ $\pm$.

1.2.3 Solving systems with multiple right-hand sides

Consider m systems with same matrix and multiple right-hand sides. If all the right-hand sides are available at once we may use an augmented matrix formulation and apply Gaussian elimination at once and solve m upper-triangular systems as described below

$$\left[\begin{array}{ccc:ccc} a_{1,1} & \cdots & a_{1,n} & \vdots & b_{1,1} & \cdots & b_{1,m} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,n} & \vdots & b_{n,1} & \cdots & b_{n,m} \end{array}\right]$$

We illustrate the procedure with partial pivoting on the example

$$\left[\begin{array}{cccc:ccc} 1 & 1 & 0 & 3 & \vdots & 4 & 0 \\ 2 & 1 & -1 & 1 & \vdots & 1 & 1 \\ 3 & -1 & -1 & 2 & \vdots & -3 & 0 \\ -1 & 2 & 3 & -1 & \vdots & 4 & 0 \end{array}\right]$$

Step 1 : interchange rows 1 and 3 and apply Gaussian elimination

$$\left[\begin{array}{cccc:ccc} 3 & -1 & -1 & 2 & \vdots & -3 & 0 \\ 0 & 5/3 & -1/3 & -1/3 & \vdots & 3 & 1 \\ 0 & 4/3 & 1/3 & 7/3 & \vdots & 5 & 0 \\ 0 & 5/3 & 8/3 & -1/3 & \vdots & 3 & 0 \end{array}\right]$$

Step 2: Apply Gaussian elimination with no row interchanges

$$\left[\begin{array}{cccc:ccc} 3 & -1 & -1 & 2 & \vdots & -3 & 0 \\ 0 & 5/3 & -1/3 & -1/3 & \vdots & 3 & 1 \\ 0 & 0 & 3 & 0 & \vdots & 0 & -1 \\ 0 & 0 & 3/5 & 13/5 & \vdots & 13/5 & -4/5 \end{array}\right]$$

Step 3: no row interchanges

$$\left[\begin{array}{cccc:ccc} 3 & -1 & -1 & 2 & \vdots & -3 & 0 \\ 0 & 5/3 & -1/3 & -1/3 & \vdots & 3 & 1 \\ 0 & 0 & 3 & 0 & \vdots & 0 & -1 \\ 0 & 0 & 0 & 13/5 & \vdots & 13/5 & -3/5 \end{array}\right]$$

Using backward substitution we obtain

$X1 = [-1, 2, 0, 1]$ and $X2 = [8/39, 19/39, -1/3, -3/13]$

Inverse of matrix A

To compute the inverse of a matrix A we need to solve the n systems $AX = I$, where I is the $n \times n$ identity matrix.

Now let us consider the augmented matrix $[A : I]$ for the following example

$$\left[\begin{array}{ccc:ccc} 2 & 1 & 0 & \vdots & 1 & 0 & 0 \\ 1 & -1 & 4 & \vdots & 0 & 1 & 0 \\ 3 & -1 & -2 & \vdots & 0 & 0 & 1 \end{array}\right]$$

Step 1: interchange row 1 and 3 and apply Gaussian elimination

$$\left[\begin{array}{ccc|ccc} 3 & -1 & -2 & 0 & 0 & 1 \\ 0 & -2/3 & 14/3 & 0 & 1 & -1/3 \\ 0 & 5/3 & 4/3 & 1 & 0 & -2/3 \end{array}\right]$$

Step 2: interchange row 2 and 3

$$\left[\begin{array}{ccc|ccc} 3 & -1 & -2 & 0 & 0 & 1 \\ 0 & 5/3 & 4/3 & 1 & 0 & -2/3 \\ 0 & 0 & 26/5 & 2/5 & 1 & -3/5 \end{array}\right]$$

We use backward substitution to solve the system and obtain

$$A^{-1} = \begin{bmatrix} 3/13 & 1/13 & 2/13 \\ 7/13 & -2/13 & -4/13 \\ 1/13 & 5/26 & -3/26 \end{bmatrix}$$

Computational Cost:

Gaussian elimination and backward substitution:

$*:\quad 4n^3/3 - n/3$

$+-\quad 4n^3/3 - 3n^2/2 + n/6$

1.3 Matrix factorization

In this section will study several matrix factorization techniques where a matrix is write a product of two simpler matrices. The most popular is the LU factoization where L and U are lower and upper triangular matrices, respectively. Our goal in this section is to factor a matrix A into a product of simpler matrices such as lower triangular L and upper triangular U matrices $A = LU$ or $PA = LU$ where P is permutation matrix obtained by interchanging the rows of the identity.

1.3.1 LU Factorization

Theorem 1.3.1. *If Gaussian elimination can applied to a matrix A without row interchanges, then there exist a lower triangular matrix L and an upper triangular matrix U such that $A = LU$.*

Proof. We define Gaussian transformation corresponding to k^{th} step of Gaussian elimination by

$$M_k = I - \tau^{(k)} e_k^t$$

where $\tau^{(k)} = (\tau_1^{(k)}, \cdots, \tau_i^{(k)}, \cdots, \tau_n^{(k)})^t$

with $\tau_j^{(k)} = 0$, $j = 1, \cdots, k$ and $\tau_j^{(k)} = a_{j,k}^{(k-1)} / a_{k,k}^{(k-1)}$, $j = k+1, \cdots, n$

and e_k is the k^{th} canonical vector in $\mathbf{R^n}$. The Gauss transformation matrix is

$$M_k = \begin{bmatrix} 1 & 0 & 0 & & 0 & 0 & \cdots & 0 \\ 0 & \ddots & 0 & & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & 0 & 1 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & 0 & -\frac{a_{k+1,k}^{(k-1)}}{a_{kk}^{(k-1)}} & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & 0 & \vdots & 0 & 1 & \cdots & 0 \\ 0 & \cdots & 0 & -\frac{a_{n,k}^{(k-1)}}{a_{kk}^{(k-1)}} & 0 & \cdots & 0 & 1 \end{bmatrix}$$

Setting $A^{(0)} = A$

Applying the k^{th} step of Gaussian elimination as

$$A^{(k)} = M_k A^{(k-1)}, k = 1, 2, ..., n-1$$

To obtain

$$U = M_{n-1} \cdots M_2 M_1 A$$

Which can be written as

$$A = M_1^{-1} M_2^{-1} \cdots M_{n-1}^{(-1)} U$$

We note that

$$M_k^{-1} = I + \tau^{(k)} e_k^t$$

one can verify that

$$M_k * M_k^{-1} = I - \tau^{(k)} e_k^t \tau^{(k)} e_k^t$$

using the fact that $e_k^t \tau^{(k)} = 0$ establishes $M_k M_k^{-1} = I$.

A straight forward computation shows that

$$M_{k-1}^{-1} M_k^{-1} = I + \tau^{(k)} e_k^t + \tau^{(k-1)} e_{k-1}^t$$

Thus

$$L = I + \sum_{k=1}^{n-1} \tau^{(k)} e_k^t$$

which contains the elimination coefficients used in the Gaussian elimination.

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 \\ a_{21}/a_{11} & 1 & 0 & 0 & \cdots & 0 \\ a_{31}/a_{11} & a_{32}^1/a_{22}^{(1)} & 1 & 0 & \cdots & 0 \\ \vdots & & & & & \vdots \\ a_{n1}/a_{11} & a_{n2}^{(1)}/a_{22}^{(1)} & a_{n3}^{(2)}/a_{33}^{(2)} & a_{n4}^{(3)}/a_{44}^{(3)} & \cdots & 1 \end{bmatrix}$$

In order to prove uniqueness of $A = LU$ factorization, we assume that $A = L_1 U_1$ and $A = L_2 U_2$ and write

$$L_1 U_1 = L_2 U_2$$

this leads to

$$L_2^{-1} L_1 = U_2 U_1^{-1}$$

Since $L_2^{-1} L_1$ is a lower triangular matrix and $U_2 U_1^{-1}$ is an upper triangular matrix we conclude that the two matrices are diagonal matrices. The diagonal elements of $L_2^{-1} L_1$ are ones, thus

$$L_2^{-1} L_1 = U_2 U_1^{-1} = I$$

Thus, $L_1 = L_2$ and $U_1 = U_2$. □

Next, let us illustrate the procedure by the example:

$$\begin{bmatrix} 2 & 1 & 2 & 0 \\ 4 & 1 & 6 & 2 \\ -6 & -1 & -7 & -3 \\ 8 & 3 & 16 & 10 \end{bmatrix}$$

Step 1: no row interchanging

$$\begin{bmatrix} 2 & 1 & 2 & 0 \\ 0 & -1 & 2 & 2 \\ 0 & 2 & -1 & -3 \\ 0 & -1 & 8 & 10 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 \end{bmatrix}$$

Step 2:

$$\begin{bmatrix} 2 & 1 & 2 & 0 \\ 0 & -1 & 2 & 2 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 6 & 8 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ -3 & -2 & 0 & 0 \\ 4 & 1 & 0 & 0 \end{bmatrix}$$

Step 3:

$$U = \begin{bmatrix} 2 & 1 & 2 & 0 \\ 0 & -1 & 2 & 2 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 6 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ -3 & -2 & 0 & 0 \\ 4 & 1 & 2 & 0 \end{bmatrix}$$

Thus,

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ -3 & -2 & 1 & 0 \\ 4 & 1 & 2 & 1 \end{bmatrix}$$

The reader may check that $LU = A$.

Applications

Solving $Ax = b$ is equivalent to solving $LUx = b$ by setting $Ux = z$ and

(i) first solve $Lz = b$

(ii) then solve $Ux = z$

A general algorithm for LU factorization

Doolittle's factorization where $l_{i,i} = 1,\ i = 1, \cdots, n$

$u_{1,j} = a_{1,j},\ j = 1, \cdots, n$
for $k = 1 \cdots n-1$
$u_{k,k} = (a_{k,k} - \sum_{j=1}^{k-1} l_{k,j}u_{j,k})/l_{k,k}$
$l_{j,k} = (a_{j,k} - \sum_{i=1}^{k-1} l_{j,i}u_{i,k})/u_{k,k},\ j = k+1, \cdots n$
$u_{k,j} = (a_{k,j} - \sum_{i=1}^{k-1} l_{k,i}u_{i,j})/l_{k,k}, j = k+1, \cdots n$

Crout's factorization

Set $u_{i,i} = 1$, $i = 1, \cdots n$
for $k = 1, \cdots n$
$l_{k,k} = (a_{k,k} - \sum_{j=1}^{k-1} l_{k,j}u_{j,k})/u_{k,k}$
$l_{j,k} = (a_{j,k} - \sum_{i=1}^{k-1} l_{j,i}u_{i,k})/u_{k,k},\ j = k+1, \cdots n$
$u_{k,j} = (a_{k,j} - \sum_{i=1}^{k-1} l_{k,i}u_{i,j})/l_{k,k},\ j = k+1, \cdots n$

Program for LU factorization with no row interchanges

```
function [L,U]=mylu(A)
%function computes the lu factorization of A with no pivoting
%input: A
%output:  L a lower triangular matrix
%         U an upper triangular matrix
%
[n,m] = size(A);
U(1,:) = A(1,:);
d = zeros(1,n);
L = diag(d+1,0);
for i = 1:n-1
L(i+1:n,i) = A(i+1:n,i)/U(i,i);
for j = i+1:n
   A(j,i:n) = A(j,i:n) - L(j,i)*U(i,i:n);
```

```
end
U(i+1,i+1:n) = A(i+1,i+1:n);
end
```

Definition: The leading principal minors of a square matrix A are

$$A_k = \begin{bmatrix} a_{11} & \cdots & a_{1k} \\ \vdots & \vdots & \vdots \\ a_{k1} & \cdots & a_{kk} \end{bmatrix}, \ k = 1, 2, \cdots n.$$

For instance,

$$A_1 = a_{11}, A_2 = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

Theorem 1.3.2. *Let A be an n by n matrix such that all leading principal minors are invertible, i.e., $det(A_k) \neq 0, \ k = 1, 2, \cdots, n$ then there exists a unique factorization $A = LU$, where $l_{ii} = 1$.*

Proof. We use induction to prove the theorem: Since $a_{11} \neq 0$ one can apply the first step in Gaussian elimination and obtain

$$A_2 = L_2 U_2 = \begin{bmatrix} 1 & 0 \\ a_{21}/a_{11} & 1 \end{bmatrix} \begin{bmatrix} a_{11} & a_{22} \\ 0 & -det(A_2)/a_{11} \end{bmatrix}$$

Assume $A_i = L_i U_i, \ i = 1, 2, \cdots, k-1$

Next we show that $A_k = L_k U_k$

$$\begin{bmatrix} A_{k-1} & c \\ d^t & a_{kk} \end{bmatrix} = \begin{bmatrix} L_{k-1} & 0 \\ \tilde{l}^t & 1 \end{bmatrix} \begin{bmatrix} U_{k-1} & \tilde{u} \\ 0 & u_{kk} \end{bmatrix}$$

where $c = (a_{1k}, \cdots, a_{k-1,k})^t$, $d = (a_{k1}, \cdots, a_{k,k-1})$, $\tilde{l} = (l_{k1}, \cdots, l_{k,k-1})$ and $\tilde{u} = (u_{1k}, \cdots, u_{k-1k})^t$.

Since A_{k-1} is nonsingular, L_{k-1} and U_{k-1} are also nonsingular. Using the LU factorization algorithm we find the next column in U_k, $u_{sk}, \ s = 1, \cdots, k$ by solving the system

$$\sum_{s=1}^{k-1} l_{is}u_{sk} = a_{ik},\ 1 \le i \le k-1.$$

Since L_{k-1} is nonsingular we have a unique solution.

Since U_{k-1} is nonsingular, the k^{th} row of L_k can be computed by solving the system:

$$\sum_{s=1}^{k-1} l_{ks}u_{sj} = a_{kj},\ 1 \le j \le k-1$$

Finally,

$$a_{kk} = \sum_{s=1}^{k} l_{sk}u_{sk} = \sum_{s=1}^{k-1} l_{sk}u_{sk} + u_{kk}$$

Thus,

$$u_{kk} = a_{kk} - \sum_{s=1}^{k-1} l_{sk}u_{sk},$$

When $k = n$ we obtain $A = LU$.

To prove uniqueness, we assume there are two factorizations $A = L_1U_1 = L_2U_2$. This can be written as

$$L_1^{-1}L_2 = U_1U_2^{-1}.$$

Since $L_1^{-1}L_2$ is lower triangular and $U_1U_2^{-1}$ is upper triangular, both matrices are diagonal. Noting that $diag(L_1^{-1}L_2) = I$ leads to $L_1 = L_2$ and $U_1 = U_2$.

□

Definition 2. *A matrix A is strictly diagonally dominant (SDD) if and only if*

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|, \quad i = 1, 2, \cdots, n.$$

Theorem 1.3.3. *Let A be a strictly diagonally dominant matrix then all leading principal minors are nonsingular. Thus A has a unique Doolittle factorization $A = LU$.*

Proof. Using Gershgorin Theorem from Chapter 4 on eigenvalues we show that all disks

$$D_i = \{\lambda \text{ such that } |\lambda - a_{ii}| \leq r_i = \sum_{j=1, j\neq i}^{n} |a_{ij}|\},\ i = 1, \cdots, n,$$

do not contain the origin. Thus, $\lambda = 0$ is not an eigenvalue □

Theorem 1.3.4. *If A^t is strictly diagonally dominant matrix, then, the Gaussian elimination with partial pivoting will not perform any row interchanges and leads to an $A = LU$ factorization.*

Proof. Let

$$A = \begin{bmatrix} a_{11} & w^t \\ v & C \end{bmatrix}$$

where $v = (a_{21}, \cdots, a_{n1})^t$ and $w = (a_{12}, \cdots, a_{1n})^t$ are two $n-1$ vectors and C is an $(n-1) \times (n-1)$ matrix.

The first step of Gaussian elimination yields:

$$L = \begin{bmatrix} 1 & 0 \\ \frac{v}{a_{11}} & C - \frac{vw^t}{a_{11}} \end{bmatrix}$$

We need to show that if A^t is SDD then the transpose of

$$B = C - \frac{vw^t}{a_{11}}$$

is also SDD. Now, let us write

$$B^t = C^t - \frac{wv^t}{a_{11}}, \quad b_{ij} = c_{ij} - \frac{v_i w_j}{a_{11}}$$

$$\sum_{i=1, i\neq j}^{n-1} |b_{ij}| = \sum_{i=1, i\neq j} |c_{ij} - \frac{v_i w_j}{a_{11}}|$$

$$\leq \sum_{i=1,i\neq j} |c_{ij}| + \frac{|w_j|}{|a_{11}|} \sum_{i=1,i\neq j} |v_i|$$

$$< (|c_{jj}| - |w_j| + \frac{|w_j|}{|a_{11}|}(|a_{11}| - |v_j|) = |c_{jj}| - \frac{|w_j||v_j|}{|a_{11}|} < |b_{jj}|$$

We have used the fact that A^t is SDD which leads to

$$\sum_{i=1,i\neq j}^{n-1} |v_i| < |a_{11}| - |v_j|$$

and

$$\sum_{i=1,i\neq j}^{n-1} |c_{ij}| < |c_{jj}| - |w_j|$$

which can be derived easily from

$$A^t = \begin{bmatrix} a_{11} & v^t \\ wc^t & \end{bmatrix}.$$

This completes the proof that B^t is strictly diagonally dominant. □

Theorem 1.3.5. *If A is SDD then the Gaussian elimination preserves the SDD property of the matrix. Furthermore, Gaussian elimination with scaled column partial pivoting, with scales recomputed at each step, will not require row interchanges. Thus, A can be factored as $A = LU$.*

Proof. Apply the first Gaussian elimination step to obtain

$$A^{(1)} = \begin{bmatrix} a_{11} & a_{12} & \cdots a_{1n} \\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} \end{bmatrix}$$

Then the matrix $A^{(1)}$ is SDD, i.e., we want to show that for $i = 2, \cdots n$,

$$|a_{ii}^{(1)}| > \sum_{j=2,j\neq i}^{n} |a_{ij}^{(1)}|$$

This means that

$$|a_{ii} - \frac{a_{i1}}{a_{11}}a_{1i}| > \sum_{j=2, j\neq i}^{n} |a_{ij} - \frac{a_{i1}}{a_{11}}a_{1j}|$$

since A is SDD we have

$$\sum_{j=2} \frac{|a_{1j}|}{|a_{11}|} < 1$$

multiplying by $|a_{i1}|$ we obtain

$$\sum_{j=2}^{n} \frac{|a_{i1}a_{ij}|}{|a_{11}|} \leq |a_{i1}|$$

From the diagonal dominance of A we have

$$|a_{i1}| < |a_{ii}| - \sum_{j=2, j\neq i}^{n} |a_{ij}|$$

This leads to

$$\sum_{j=2}^{n} \frac{|a_{i1}a_{1j}|}{|a_{11}|} \leq |a_{ii}| - \sum_{j=2, j\neq i}^{n} |a_{ij}|$$

which is equivalent to :

$$\sum_{j=2, j\neq i}^{n} \frac{|a_{i1}a_{1j}|}{|a_{11}|} + \sum_{j=2, j\neq i}^{n} |a_{ij}|$$

$$< |a_{ii}| - \frac{|a_{i1}a_{1i}|}{|a_{11}|} < |a_{ii} - \frac{a_{i1}a_{1i}}{a_{11}}|$$

Finally using

$$\sum_{j=2, j\neq i}^{n} |a_{ij} - \frac{a_{i1}a_{1j}}{a_{11}}| \leq \sum_{j=2, j\neq i}^{n} \frac{|a_{i1}a_{1j}|}{|a_{11}|} + \sum_{j=2, j\neq i}^{n} |a_{ij}|$$

we complete the proof that $A^{(1)}$ is SDD . By induction $A^{(k)}$ is SDD.

To show that no row interchanges are required we note that in step 1, $|a_{11}/s_1| = 1$, $|a_{k1}/s_k| < 1$, for $k = 2, \cdots, n$. □

Corollary 1. *Every SDD matrix can be factored as $A = LU$.*

Proof. Use the fact that all the pivots are nonzero by SDD property. □

1.3.2 LU factorization with row interchanges

Theorem 1.3.6. *If A is a non singular matrix then there exist a lower triangular matrix L, an upper triangular matrix U and a permutation matrix P (obtained by interchanging rows of the identity) such that $LU = PA$.*

We illustrate the procedure on an example using partial pivoting

$$U = \begin{bmatrix} 0 & 2 & 3 \\ 1 & 1 & 0 \\ 2 & 1 & 3 \end{bmatrix} \quad L = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Step 1: interchange row 1 and 3 and apply Gaussian elimination to obtain

$$U = \begin{bmatrix} 2 & 1 & 3 \\ 0 & 1/2 & -3/2 \\ 0 & 2 & 3 \end{bmatrix} \quad L = \begin{bmatrix} 0 & 0 & 0 \\ 1/2 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Step 2: interchange rows 2 and 3 and apply Gaussian elimination

$$U = \begin{bmatrix} 2 & 1 & 3 \\ 0 & 2 & 3 \\ 0 & 0 & -9/4 \end{bmatrix} \quad \tilde{L} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1/2 & 1/4 & 0 \end{bmatrix} \quad P = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

$$L = \tilde{L} + I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1/2 & 1/4 & 1 \end{bmatrix}$$

where $LU = PA$. In order to solve $Ax = b$ we multiply the system by P to obtain $PAx = Pb$ which leads to $LUx = Pb$.

We solve the system by solving
(i) $Lz = Pb$
(ii) $Ux = z$

For instance, we solve $Ax = [5, 0, 2]' = b$ by computing

(i) $Pb = [2, 5, 0]$
(ii) $Lz = Pb$ yields $z = [2, 5, -9/4]'$
(iii) $Ux = z$ leads to $x = [-1, 1, 1]'$.

Matlab program for Lu factorization with partial pivoting

```
function [L,U,P,mperm]=mylupiv(A)
%PA=LU factorization with partial pivoting
%input: matrix A
%output:  L lower triangular matrix
%         U upper triangular
%         P permutation matrix
%         mperm number of row interchanges
%
[n,m] = size(A);
P = eye(n);
U(1,:) = A(1,:);
d = ones(1,n);
L = zeros(n);
mperm = 0;
for i = 1:n-1
[mx,p]=max(abs(A(i:n,i)));
p = p-1+i;
if (p ~= i)
  mperm = mperm + 1;
% rows interchanges in A
  tmp=A(p,i:n);
  A(p,i:n)=A(i,i:n);
  A(i,i:n)=tmp;
% rows interchanges in L
  tmp= L(i,1:i-1);
  L(i,1:i-1)=L(p,1:i-1);
  L(p,1:i-1) = tmp;
% rows interchanges in P
  tmp=P(p,:);
  P(p,:)=P(i,:);
  P(i,:)=tmp;
end
U(i,i:n) = A(i,i:n);
%
% compute l factors
```

```
L(i+1:n,i) = A(i+1:n,i)/U(i,i);
%Gaussian elimination
for j = i+1:n
   A(j,i:n) = A(j,i:n) - L(j,i)*U(i,i:n);
end
end
U(n,n) = A(n,n);
L = L+diag(d);
```

I. **Determinant of A**

The method of co-factors requires $n!$ operations, for instance, for $n = 100$, we need $10^{158} operations$. Thus, on a machine with 10^{12} flops it will take 10^{138} years to compute the determinant of A. On the other-hand, using LU factorization it will take less than a second to compute the determinant as

$det(A) = det(L) * det(U) = (\Pi_{i=1}^{n} l_{i,i})(\Pi_{i=1}^{n} u_{i,i}) = \Pi_{i=1}^{n} u_{i,i}$

In general when $PA = LU$

$det(A) = (-1)^m(\Pi_{i=1}^{n} l_{i,i})(\Pi_{i=1}^{n} u_{i,i}) = (-1)^m(\Pi_{i=1}^{n} u_{i,i})$, m is the number of row interchanges during Gaussian elimination.

II. **Solving** $Ax = b$.

The LU factorization is useful for solving $Ax = b$ if all right-hand sides are not known at the beginning of the computation.

1.3.3 LU factorization of special matrices

Symmetric positive matrices

Definition A matrix A is a symmetric positive definite (SPD) matrix if and only if

- (i) $A^t = A$
- (ii) $x^t Ax > 0, \ \forall \ x \neq 0$

Theorem 1.3.7. *If A is symmetric matrix then*

- *(i) A has only real eigenvalues*
- *(ii) the associated eigenvectors form an orthogonal basis.*

Proof. Consult book on linear algebra by Johnson, Riess and Arnold. □

Theorem 1.3.8. *A symmetric matrix A is positive definite matrix if and only if $det(A_k) > 0$ for $k = 1, 2, \cdots n$, where*

$$A_k = \begin{bmatrix} a_{1,1} & \cdots & a_{1,k} \\ \vdots & \ddots & \vdots \\ a_{k,1} & \cdots & a_{k,k} \end{bmatrix}$$

Proof. Use $A_k = Q^t \Lambda Q$, $det(A_k) = \Pi_{i=1}^k \lambda_i^{(k)}$ and the fact that $x^t A_k x > 0$, $\forall x \neq 0,\ x \in \mathbf{R^k}$ □

Theorem 1.3.9. *If A is a symmetric positive matrix, then there exist a lower triangular matrix $\tilde{L}$ such that*

$$A = \tilde{L} * \tilde{L}^t$$

where $l_{ii} > 0,\ i = 1, \cdots n$

Proof. Using vectors of the form $x = (x_1, x_2, \cdots, x_k, 0, \cdots, 0)^t$ we show that Since A_k are also SPD, thus they are nonsingular. From the LU factorization there exist a unique factorization

$$A = LU = U^t L^t = A^t$$

which can be written as

$$U(L^t)^{-1} = L^{-1} U^t$$

Since $U(L^t)^{-1}$ is an upper triangular matrix while $L^{-1}U^t$ is a lower triangular matrix, both matrices are diagonal matrices and equal to D.

$$U(L^t)^{(-1)} = D$$

This leads to $U = DL^t$ Now we can write:

$$A = LDL^t$$

Since A is symmetric positive definite matrix, $d_{ii} > 0$.

$$0 < x^t Ax = y^t Dy, \text{ where } y = x^t L.$$

Thus, we can write

$$D = \sqrt{D}\sqrt{D}$$

where

$$\sqrt{D} = diag(\sqrt{d_{11}}, \cdots, \sqrt{d_{nn}})$$

Finally, we define $\tilde{L} = L\sqrt{D}$. □

Cholesky Factorization algorithm

for $i = 1, 2, \cdots n$

$$l_{i,i} = \sqrt{a_{i,i} - \sum_{k=1}^{i-1} l_{i,k}^2}$$

$$l_{j,i} = (a_{j,i} - \sum_{k=1}^{i-1} l_{j,k} l_{i,k})/l_{i,i}, \ j = i+1, \cdots n$$

Applications

In order to solve $Ax = LL^t x = b$ we solve
(i) $Lz = b$
(ii) $L^t x = z$

Remarks

(i) Cholesky factorization costs $\approx n^3/6$ operations, i.e., half the cost of the regular LU factorization.

(ii) It is stable with respect to round-off errors.

(iii) Square root is expensive use $A = LDL^t$, $l_{i,i} = 1$.

Let us consider an example

```
>[L,p] = chol(A)
```

$$A = \begin{bmatrix} 4 & -2 & 8 \\ -2 & 2 & 1 \\ 8 & 1 & 141 \end{bmatrix}$$

One may check that

$det(A_1) = 4 > 0$
$det(A_2) = 8 - 4 = 4 > 0$
$det(A) = 400 > 0$

and $A' = A$. Then A is a positive definite matrix with

$$L = \begin{bmatrix} 2 & 0 & 0 \\ -1 & 1 & 0 \\ 4 & 5 & 10 \end{bmatrix}$$

Matlab program for Cholesky factorization

```
function U = mychol(A)
%function computes Cholesky factorization
%input: matrix A positive definite
%
%output: U an upper matrix such as U'U=A
%
[n,m]=size(A);
L(:,1) = A(:,1)/sqrt(A(1,1));
for i=2:n
nz = max([i-1,0]);
nz
    L(i,i) =sqrt(A(i,i) - norm(L(i,1:nz),2)^2);
    for j=i+1:n
       L(j,i) = (A(j,i) - L(j,1:i-1)*L(i,1:i-1)')/L(i,i);
    end
end
U = L';
```

Note: One can also factor A as $A = LDM^t$, where L and M are lower triangular matrices such that $l_{ii} = m_{ii} = 1,\ i = 1, \cdots, n$ and D is a diagonal matrix.

Tridiagonal matrices

We consider the tridiagonal matrix where $a_{i,j} = 0$ if $|i - j| > 1$.

$$\begin{bmatrix} a_{11} & a_{1,2} & 0 & \cdots & 0 \\ a_{21} & a_{22} & a_{23} & \ddots & 0 \\ 0 & a_{32} & a33 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & a_{n-1,n} \\ 0 & \cdots & 0 & a_{n,n-1} & a_{nn} \end{bmatrix}$$

Gaussian elimination can be applied to factor this matrix as $A = LU$ very efficiently using only $O(n)$ operations where

$$L = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \ddots & 0 \\ 0 & l_{32} & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots & 0 \\ 0 & \cdots & 0 & l_{n,n-1} & 1 \end{bmatrix} \quad U = \begin{bmatrix} u_{11} & u_{1,2} & 0 & \cdots & 0 \\ 0 & u_{22} & u_{23} & \ddots & 0 \\ 0 & 0 & u33 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots & u_{n-1,n} \\ 0 & \cdots & 0 & 0 & u_{nn} \end{bmatrix}$$

Algorithm:

$u_{11} = a_{11},\ u_{12} = a_{12}$

$l_{21} = a_{21}/a_{11}$

$fori = 2, 3, \cdots n$
$\quad u_{i,i} = a_{ii} - l_{i,i-1} * u_{i-1,i},\ u_{i,i+1} = a_{i,i+1}$
$\quad l_{i+1,i} = a_{i+1,i}/u_{i,i}$

$u_{n,n} = a_{n,n} - l_{n,n-1} * u_{n-1,n}$

Banded matrices

Matlab program for banded matrices

```
function [L,U]=mylu(A,nl,nu)
%Lu factorization with no pivoting
%input: A matrix
%       nl number of lower diagonals
%       nu number of upper diagonals
%output: L lower triangular matrix
%        U upper triangular matrix
%
[n,m] = size(A);
U = zeros(n);
U(1,1:min(n,1+nu)) = A(1,1:min(1+nu,n));
d = zeros(1,n);
L = diag(d+1,0);
for i = 1:n-1
L(i+1:min(i+nl,n),i) = A(i+1:min(i+nl,n),i)/U(i,i);
for j = i+1:min(i+nl,n)
 A(j,i:min(i+nu,n)) = A(j,i:min(i+nu,n)) - L(j,i)*U(i,i:min(i+nu,n));
end
U(i+1,i+1:min(i+1+nu,n)) = A(i+1,i+1:min(i+1+nu,n));
end
```

1.3.4 Conditioning and stability

Conditioning

If we start with an error in the data for instance in A or b what is the error in x, where $Ax = b$.

Let the exact problem be $Ax = b$.

Let an approximate problem be $A\tilde{x} = b + \tilde{b}$ and subtract the approximate problem from the exact problem to obtain

$$A(x - \tilde{x}) = \tilde{b}$$

which can be written as

$$(x - \tilde{x}) = A^{-1}\tilde{b}$$

Taking the norm of both sides we obtain

$$||(x - \tilde{x})|| = ||A^{-1}\tilde{b}||$$

The relative error leads to

$$\frac{||(x - \tilde{x})||}{||x||} = \frac{||b||||A^{-1}\tilde{b}||}{||x||||b||} \leq \frac{||b||||A^{-1}||||\tilde{b}||}{||x||||b||}$$

Using

$$||A^{-1}\tilde{b}|| \leq ||A^{-1}||||\tilde{b}||.$$

and

$$||b|| = ||Ax|| \leq ||A||||x||$$

leads to

$$\boxed{\frac{||(x - \tilde{x})||}{||x||} = ||A^{-1}||A||\frac{||\tilde{b}||}{|||b||} = \kappa(A)\frac{||\tilde{b}||}{||b||}},$$

where $\kappa(A) = ||A^{-1}||A||$ is the condition number.

Noting that $\tilde{b} = b - A\tilde{x} = r$ we can write

$$\boxed{\frac{||(x - \tilde{x})||}{||x||} \leq \kappa(A)\frac{||r||}{|||b||}}$$

which gives a relationship between the residual and the relative error in the solution.

Remarks

1. $\kappa(A) \geq 1$

2. if $\kappa(A) >> 1$ the system is ill-conditioned

3. if $\kappa(A) = O(1)$ the system is well conditioned

5. Small errors in b for ill-conditioned systems result in large errors in x

Example: Consider the following matrix

$$A = \begin{bmatrix} 10^{-6} & 1 & 1 \\ -10^{-10} & 15 & -5 \\ 0 & 11 & 2 \end{bmatrix}$$

$||A||_\infty = 20 + 10^{-10},\ ||A^{-1}||_\infty = 1.34\ 10^6$ leads to

$\kappa(A) = ||A||_\infty ||A^{-1}||_\infty = 2.68\ 10^7$

For instance if $||\tilde{b}||_\infty / ||b||_\infty = 10^{-7}$, then $\frac{||x-\tilde{x}||_\infty}{||x||_\infty} \approx 2.68$.

Now, if we assume the approximate problem to be

$$(A + \tilde{A})\tilde{x} = b + \tilde{b},$$

then we can prove that

$$\boxed{\frac{||(x-\tilde{x})||}{||x||} = \kappa(A)\left(\frac{||\tilde{A}||}{||A||} + \frac{||\tilde{b}||}{|||b||}\right) + O(||\tilde{A}||^2)}.$$

If $\tilde{A}$ is small enough, we can prove the result by writing

$$\tilde{x} = (A + \tilde{A})^{-1}(b + \tilde{b}) = [A(I + A^{-1}\tilde{A})]^{-1}(b + \tilde{b})$$

$$\tilde{x} = (I - A^{-1}\tilde{A} + O(\tilde{A}^2))(A^{-1}b + A^{-1}\tilde{b})$$

Replacing $x = A^{-1}b$ we have

$$\tilde{x} = x - A^{-1}\tilde{A}x + A^{-1}\tilde{b} + O(\tilde{A}62)$$

Now, passing x to the left, taking the norm and applying the triangle inequality we can write

$$\frac{||\tilde{x} - x||}{||x||} \leq \frac{||A^{-1}||||\tilde{b}||}{||x||} + ||A^{-1}||||\tilde{A}||$$

which can be written using $||b|| \leq ||A||||x||$ as

$$\frac{||\tilde{x} - x||}{||x||} \leq \kappa(A)(\frac{||\tilde{b}||}{||b||} + \frac{||\tilde{A}||}{||A||}).$$

Example in matlab

```
>A = [ 2+10^(-13) 1;2 1];
>b = [ 3 + 10^(-13); 3];
>A\b

  0.9977777777777
  1.004444444444
```

Iterative refinement Algorithm

This algorithm is useful for ill-conditioned systems ($\kappa(A) >> 1$).

1. Perform $LU = PA$ with t digits
2. Solve $LUx^0 = Pb$
3. Compute $r^0 = b - Ax^0$ using 2t digits (double precision)
4. Solve $LUx_c = Pr^0$
5. update x^0 as $x^0 = x^0 + xc$
6. if $||x_c|| < \epsilon$ stop

   else go to 3

Remarks:

(i) If machine zero is $Eps = 10^{-d}$ and $\kappa_\infty(A) = 10^q$ and using double precision to compute $b - Ax$

x^0 has approximately $min(d, k * (d - q))$ correct digits

where k is the iteration number in the refinement algorithm

Thus for instance

when $k = 1$, x has $min(d, d - q)$ correct digits

when $k = 2$, x has $min(d, 2(d - q))$ correct digits

(ii) each iteration costs $O(n^2)$ operations not including the LU factorization.

Example:

Using the decimal system with $t = 3$ significant digits we solve

$$\begin{bmatrix} 0.986 & 0.579 \\ 0.409 & 0.237 \end{bmatrix} \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad = \quad \begin{bmatrix} 0.235 \\ 0.107 \end{bmatrix}$$

The Exact solution is $(2, 3)$ and $\kappa_\infty(A) = 52.73$.

Using the LU factorization we obtain

$$X^0 = \begin{bmatrix} 2.11 \\ -3.17 \end{bmatrix} \quad X^1 = \begin{bmatrix} 1.99 \\ -2.99 \end{bmatrix} \quad X^1 = \begin{bmatrix} 2.00 \\ -3.00 \end{bmatrix}$$

Stability

Let us consider the LU factorization of the matrix

$$A = \begin{bmatrix} \epsilon & 1 \\ 1 & 1 \end{bmatrix}, \text{ where } |\epsilon| << 1$$

$$A^{-1} = \frac{1}{1-\epsilon} \begin{bmatrix} -1 & +1 \\ 1 & -\epsilon \end{bmatrix},$$

Condition number of A in the ∞ norm is

$$\kappa(A) = ||A||_\infty ||A^{-1}||_\infty$$

$$||A||_\infty = 2$$

$$||A^{-1}||_\infty = 2/(1-\epsilon)$$

Thus $\kappa(A) = 4/(1-\epsilon) \approx 4$ when $|\epsilon| << 1$. The matrix A is well conditioned.

Now let us apply Gaussian elimination to perform $A = LU$ factorization to find that

$$A = \begin{bmatrix} \epsilon & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1/\epsilon & 1 \end{bmatrix} \begin{bmatrix} \epsilon & 1 \\ 0 & 1 - 1/\epsilon \end{bmatrix}$$

In order to eliminate the effect of conditioning of the problem, we assume that the input A and b are exact. When $|\epsilon| << 1$, round-off error cause a $O(1)$ relative error in the solution x. We explain that by performing a backward error analysis that starts with

$$A = LU \rightarrow \tilde{A} = \tilde{L}\tilde{U}$$

In order to perform an error analysis we consider

$$\phi(A) = LU \text{ and } \tilde{\phi}(A) = \tilde{L}\tilde{U}$$

If $\epsilon = 2^{-m}$, with m small enough we have

$$\tilde{L} = \begin{bmatrix} 1 & 0 \\ 1/\epsilon & 1 \end{bmatrix} \quad \tilde{U} = \begin{bmatrix} \epsilon & 1 \\ 0 & -1/\epsilon \end{bmatrix}$$

Thus

$$L = \tilde{L} \text{ and } \tilde{U} = U + \begin{bmatrix} 0 & 1 \\ 0 & 1 \end{bmatrix}$$

$$\frac{||U - \tilde{U}||}{||U||} \approx \epsilon$$

Now we have

$$A + \tilde{A} = \tilde{L}\tilde{U}$$

where

$$\tilde{A} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

Finally,

$$\frac{||\phi(A) - \tilde{\phi}A||}{||\phi(A)||} = \frac{||\tilde{A}||}{||A||} = O(1).$$

Therefore, using the previous sensitivity result we show that Gaussian elimination for this problem is unstable.

Apply the LU factorization with partial pivoting we obtain

$$\phi(A) = PA = \begin{bmatrix} 1 & 1 \\ \epsilon & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 0 \\ \epsilon & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1-\epsilon \end{bmatrix}$$

If ϵ is small enough we have

$$\tilde{\phi}(A) = PA + \tilde{A} =$$

$$\begin{bmatrix} 1 & 0 \\ \epsilon & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \phi(PA + \tilde{A})$$

where

$$\tilde{A} = \begin{bmatrix} 0 & 0 \\ 0 & \epsilon \end{bmatrix}.$$

Finally, we have

$$\frac{||\phi(A) - \tilde{\phi}(A)||}{||\phi(A)||} = \frac{||\tilde{A}||}{||A||} = O(\epsilon).$$

Thus Gaussian elimination with partial pivoting is backward stable for this problem.

Analysis of round-off error analysis: section 4.8 of the text book by Kincaid and Cheney.

Theorem 1.3.10. *Let A be an $n \times n$ matrix with all entries being machine numbers. Then Gaussian elimination with row pivoting produces $\tilde{L}$ and $\tilde{U}$ such that*

$$A + \tilde{A} = \tilde{L}\tilde{U}$$

where

$$|\tilde{a}_{ij}| \leq 2nEps \max_{i,j,k} |a_{ij}^{(k)}|$$

Theorem 1.3.11. *Let L be a lower triangular matrix and b a vector such that all entries are machine numbers. If $\tilde{y}$ is the the computed solution of $Ly = b$, then it is the exact solution of $(L + \tilde{L})\tilde{y} = b$ where $\tilde{l}_{ij} \leq 6(n+1)Epsl_{ij}/5$*

Theorem 1.3.12. *Let U be an upper triangular matrix and b a vector such that all entries are machine numbers. If $\tilde{y}$ is the the computed solution of $Uy = b$, then it is the exact solution of $(U + \tilde{U})\tilde{y} = b$ where $\tilde{u}_{ij} \leq 6(n+1)Epsu_{ij}/5$*

Theorem 1.3.13. *Let A be $n \times n$ matrix and b a vector such that all entries are machine numbers and $nEps < 1/3$. If $\tilde{x}$ is the the computed solution*

of $Ax = b$ using Gaussian elimination with row interchanges, then it is the exact solution of

$$(A + \tilde{A})\tilde{y} = b$$

where

$$\tilde{a}_{ij} \leq 10n^2 Eps\rho, \quad \rho = \max_{i,j,k} |a_{ij}^{(k)}|.$$

Furthermore, Wilkinson showed that

$$\frac{||x - \tilde{x}||}{||x||} < 4n^2 g_n(A)\kappa_\infty(A)Eps$$

where

$$g_n(A) = \frac{\max\limits_{ijk} |a_{ij}^{(k)}|}{\max\limits_{ij} |a_{ij}|}.$$

The largest bound is $g_n(A) \leq 2^{n-1}$ which is reached for the matrix A such that

$$a_{ij} = \begin{cases} -1 & i > j \\ 1 & i = j \text{ or } j = n \\ 0 & otherwise \end{cases}$$

where $a_{nn}^{(n-1)} = 2^{n-1}$. When $n = 64$ with double precison all accuracy is lost. Thus there is stability problem of the Gaussian elimination for large n. Read discussion in section 4.8 of Kincaid and Cheney.