

EE731 Lecture Notes: Matrix Computations for Signal Processing

James P. Reilly©
Department of Electrical and Computer Engineering
McMaster University

November 7, 2005

Lecture 5

In this chapter we discuss *Gaussian elimination* from a "big block" perspective, which is significantly different from what is taught in undergraduate curricula. This treatment leads very nicely into the LU decomposition of a matrix, which is a valuable tool in the solution of systems of linear equations. It is shown that there is a one-to-one correspondence between the Gaussian elimination process and the LU decomposition. We then discuss a numerical error analysis of Gaussian elimination, and show that the technique is unstable without pivoting. We also discuss an iterative refinement technique that produces a computed solution to full single precision accuracy.

Then we extend the above treatment on Gaussian elimination to develop the Cholesky factor of a matrix. Error analysis shows that this decomposition is stable without pivoting. We discuss properties of the Cholesky factorization and look at several signal processing applications, particularly with regard to whitening sequences.

8 Gaussian Elimination

In this section, we discuss the concept of Gaussian elimination in some detail. But first, we present a very quick review, by example, of the elementary approach to Gaussian elimination. Given the system of equations

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

where $\mathbf{A} \in \mathfrak{R}^{3 \times 3}$ is nonsingular. The above system can be expanded into the form

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}.$$

To solve the system, we transform this system into the following upper triangular system by Gaussian elimination:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ & a'_{22} & a'_{23} \\ & & a''_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b'_2 \\ b''_3 \end{bmatrix} \rightarrow \mathbf{U}\mathbf{x} = \mathbf{b}' \quad (1)$$

using a sequence of elementary row operations, as follows:

$$\text{row } 2' := \text{row } 2 - \frac{a_{21}}{a_{11}} \text{row } 1,$$

$$\text{row } 3' := \text{row } 3 - \frac{a_{31}}{a_{11}} \text{row } 1,$$

and

$$\text{row } 3'' := \text{row } 3' - \frac{a'_{32}}{a'_{22}} \text{row } 2'.$$

The prime indicates the respective quantity has been changed. Each elementary operation preserves the original system of equations. Each operation is designed to place a zero in the appropriate place below the main diagonal of \mathbf{A} .

Once \mathbf{A} has been triangularized, the solution \mathbf{x} is obtained by applying *backward substitution* to the system $\mathbf{U}\mathbf{x} = \mathbf{b}'$. With this procedure, x_n is first determined from the last equation of (1). Then x_{n-1} may be determined

from the second-last row, etc. The algorithm may be summarized by the following schema:

```

for  $i$  =  $n, \dots, 1$ 
   $x_i$  :=  $b_i$ 
  for  $j$  =  $i + 1, \dots, n$ 
     $x_i$  :=  $x_i - u_{ij}x_j$ 
   $x_i$  :=  $\frac{x_i}{u_{ii}}$ 
end

```

8.1 What About the Accuracy of Back Substitution?

With operations on floating point numbers, we must be concerned about the accuracy of the result, since the floating point numbers themselves contain error. We want to know if it is possible that the small errors in the floating point representation of real numbers can lead to large errors in the computed result. In this vein, we can show¹ that the computed solution $\hat{\mathbf{x}}$ obtained by back substitution satisfies the expression

$$(\mathbf{U} + \mathbf{F})\hat{\mathbf{x}} = \mathbf{b}'$$

where $|\mathbf{F}| \leq nu|\mathbf{U}| + O(u^2)$, and u is machine epsilon. (Note the use of absolute value notation as discussed in the last lecture). The above equation says that $\hat{\mathbf{x}}$ is the *exact* solution to a slightly perturbed system. We see that all elements of \mathbf{F} are of $O(u)$; hence, we conclude that the error in the solution induced by the backward substitution process is of the same order as that due to floating point representation alone; hence, back substitution *is stable*. By a numerically stable algorithm, we mean one that produces relatively small errors in its output values for small errors in the input values. This error performance is in contrast to the ordinary form of Gaussian elimination, as we see later.

The total number of flops required for Gaussian elimination of a matrix $\mathbf{A} \in \mathfrak{R}^{n \times n}$ may be shown to be $O(\frac{2n^3}{3})$ (one “flop” is one floating point arithmetic operation; i.e., a floating point add, subtract, multiply, or divide).

¹Golub and Van Loan

It is easily shown that backward substitution requires $O(n^2)$ flops. Thus, the number of operations required to solve $\mathbf{Ax} = \mathbf{b}$ is dominated by the Gaussian elimination process.

8.2 The LU Decomposition

Suppose we can find a lower triangular matrix $\mathbf{L} \in \mathfrak{R}^{n \times n}$ with ones along the main diagonal, and an upper triangular matrix $\mathbf{U} \in \mathfrak{R}^{n \times n}$ such that:

$$\mathbf{A} = \mathbf{LU}.$$

This decomposition of \mathbf{A} is referred to as the *LU decomposition*. To solve the system $\mathbf{Ax} = \mathbf{b}$, or $\mathbf{LUx} = \mathbf{b}$ we define the variable \mathbf{z} as $\mathbf{z} = \mathbf{Ux}$ and then

$$\begin{aligned} \text{solve } \mathbf{Lz} &= \mathbf{b} \text{ for } \mathbf{z} \\ \text{and } \mathbf{Ux} &= \mathbf{z} \text{ for } \mathbf{x}. \end{aligned}$$

Since both systems are triangular, they are easy to solve. The first system requires only forward elimination; and the second only back-substitution. Forward elimination is the analogous process to backward substitution, but performed on a lower triangular system instead of an upper triangular one. Forward substitution requires an equal number of flops as back substitution and is just as stable. Thus, once the LU factorization is complete, the solution of the system is easy: the total number of flops required to solve $\mathbf{Ax} = \mathbf{b}$ is $2n^2$, once the *LU* factorization of \mathbf{A} is complete. The details of the computation of the LU factorization and the number of flops required is discussed later.

We are lead to several significant questions:

1. How does one perform the LU decomposition?
2. How much computational effort is required to perform the LU decomposition?
3. What is the relationship of LU decomposition, if any, to Gaussian elimination?

4. Is the LU decomposition process numerically stable?

The answer to these questions is provided in the following sections.

8.3 Gauss Transforms

The Gaussian elimination process may be described as a sequence of Gauss transformations $\mathbf{M}_1 \dots \mathbf{M}_{n-1} \in \mathfrak{R}^{n \times n}$ such that

$$\mathbf{M}_{n-1} \dots \mathbf{M}_2 \mathbf{M}_1 \mathbf{A} = \mathbf{U} \quad (2)$$

where \mathbf{U} is the $n \times n$ upper triangular matrix yielded by Gaussian elimination. The matrix \mathbf{M}_k introduces zeros below the main diagonal in the k th column of the version of \mathbf{A} which results after $k - 1$ previous transformations. Thus, after $n - 1$ such transformations, the resulting product is upper triangular and the Gaussian elimination procedure is complete. Now we look at the structure of \mathbf{M}_k in (2).

Suppose for $k < n$ we have already determined Gauss transformations $\mathbf{M}_1 \dots \mathbf{M}_{k-1}$ so that the resulting matrix \mathbf{A}^{k-1} has the form

$$\mathbf{A}^{k-1} = \mathbf{M}_{k-1} \dots \mathbf{M}_1 \mathbf{A} = \begin{bmatrix} \mathbf{A}_{11}^{(k-1)} & \mathbf{A}_{12}^{(k-1)} \\ \mathbf{0} & \mathbf{A}_{22}^{(k-1)} \end{bmatrix} \begin{matrix} k-1 \\ n-k+1 \end{matrix} \quad (3)$$

$k-1 \quad n-k+1$

where $\mathbf{A}_{11}^{(k-1)}$ is upper triangular.

The fact that $\mathbf{A}_{11}^{(k-1)}$ is upper triangular means that the decomposition of (3) has already progressed $(k - 1)$ stages, as indicated by the superscript $(k - 1)$. The next stage of Gaussian elimination proceeds one step to make the first column of $\mathbf{A}_{22}^{(k-1)}$ zero below the main diagonal. Lets see how this can be done by pre-multiplication of \mathbf{A}^{k-1} by a matrix \mathbf{M}_k .

Define

$$\mathbf{M}_k = \mathbf{I} - \boldsymbol{\alpha}^{(k)} \mathbf{e}_k^T \quad (4)$$

where

\mathbf{I} is the $n \times n$ identity matrix

\mathbf{e}_k is the k^{th} column of \mathbf{I}

i.e., $\mathbf{e}_k^T = (0, \dots, 0, 1, 0, \dots, 0)$

↑ k^{th} position

and

$$\boldsymbol{\alpha}^{(k)} \triangleq (0, \dots, 0, l_{k+1,k}, \dots, l_{n,k})^T, \quad (5)$$

where

$$l_{ik} = \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}, \quad i = k+1, \dots, n. \quad (6)$$

The element $a_{kk}^{(k-1)}$ plays a strategically significant role in the Gaussian elimination process. As such, it is referred to as the *pivot element*. We discuss the role of this element in more detail, later.

By evaluating (4), we see that \mathbf{M}_k has the following structure:

$$\mathbf{M}_k = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \mathbf{0} \\ & & -l_{k+1,k} & 1 & & \\ \mathbf{0} & & \vdots & & \ddots & \\ & & -l_{n,k} & & & 1 \end{bmatrix} \begin{array}{l} \leftarrow k^{\text{th}} \text{ row} \\ \\ \\ \uparrow \\ k^{\text{th}} \text{ column} \end{array} \quad (7)$$

We assume the pivot element $a_{kk}^{(k-1)} \neq 0$. Because the upper $k \times k$ block of $\mathbf{M}_k = \mathbf{I}_{k \times k}$, the first k rows of $\mathbf{M}_k \mathbf{A}^{(k-1)}$ are identical to those of $\mathbf{A}^{(k-1)}$. Also, this premultiplication leaves the lower-left zero block of $\mathbf{A}^{(k-1)}$ intact. Thus, the premultiplication by \mathbf{M}_k only affects the block $\mathbf{A}_{22}^{(k-1)}$ depicted in (3). Because the l_{ik} values from (6) are precisely the multipliers required by the Gaussian elimination procedure to place zeros in desired positions of $\mathbf{A}_{22}^{(k-1)}$, the premultiplication of $\mathbf{A}^{(k-1)}$ by \mathbf{M}_k performs exactly the same elementary operations on $\mathbf{A}_{22}^{(k-1)}$ as would be performed by

elementary Gaussian elimination; i.e.,

$$\text{row}_i \leftarrow \text{row}_i - \frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} \text{row}_k, \quad \begin{cases} k = 1, \dots, n-1, \\ i = k+1, \dots, n. \end{cases}$$

Thus, $\mathbf{A}_{22}^{(k-1)}$ is replaced by zeros below the main diagonal in the first column, as desired. After $n-1$ stages of Gauss transforms, the Gaussian elimination process is complete.

8.4 Recovery of the LU factors from Gaussian Elimination

We now discuss the relationship between Gaussian elimination and the LU decomposition. Specifically, we investigate how to determine the \mathbf{L} and \mathbf{U} factors of \mathbf{A} in an efficient manner, from the Gaussian elimination process. We note the Gaussian elimination process produces

$$\mathbf{M}_{n-1} \dots \mathbf{M}_1 \mathbf{A} = \mathbf{U} \quad (8)$$

where \mathbf{U} is the upper triangular matrix resulting from the Gaussian elimination process. Each \mathbf{M}_i is lower triangular, and it is easily verified that the product of lower triangular matrices is also lower triangular. Therefore, we define a lower-triangular matrix \mathbf{L}^{-1} as

$$\mathbf{M}_{n-1} \dots \mathbf{M}_1 = \mathbf{L}^{-1} \quad (9)$$

From (8), we then have $\mathbf{L}^{-1} \mathbf{A} = \mathbf{U}$. But since the inverse of a lower triangular is also lower triangular, then

$$\mathbf{A} = \mathbf{L}\mathbf{U} \quad (10)$$

which is the product of lower and upper triangular factors as desired. We have therefore completed the relationship between \mathbf{LU} decomposition and Gaussian elimination. \mathbf{U} is simply the upper triangular matrix resulting from Gaussian elimination, and \mathbf{L} is the inverse of the product of the \mathbf{M}_i 's.

8.5 Efficient recovery of \mathbf{L}

We now show that \mathbf{L} can be recovered from the \mathbf{M}_k 's in a very efficient manner without having to perform any explicit computation. The reason is

that the \mathbf{M}_k 's have a very simple structure which can be exploited to our advantage. We note from (9) that

$$\mathbf{L} = \mathbf{M}_1^{-1} \dots \mathbf{M}_{n-1}^{-1}. \quad (11)$$

Therefore, we formulate \mathbf{L} efficiently in two steps: we first examine the relationship between \mathbf{M}_k^{-1} and \mathbf{M}_k , $k = 1, \dots, (n-1)$, and then investigate the structure of $\prod_{k=1}^{n-1} \mathbf{M}_k^{-1}$.

8.5.1 The structure of \mathbf{M}_k^{-1}

We note that

$$\mathbf{M}_k \mathbf{A}^{(k-1)} = \mathbf{A}^{(k)} \quad (12)$$

The matrix $\mathbf{A}^{(k)}$ is formed from $\mathbf{A}^{(k-1)}$ by implicitly performing a *subtraction operation* as indicated by the structure of \mathbf{M}_k :

$$\mathbf{M}_k = \mathbf{I} - \boldsymbol{\alpha}^{(k)} \mathbf{e}_k^T. \quad (13)$$

The matrix \mathbf{M}_k^{-1} must operate on $\mathbf{A}^{(k)}$ to restore $\mathbf{A}^{(k-1)}$. Do you suppose this could be achieved by implicitly performing an *addition operation* on $\mathbf{A}^{(k)}$? This insight is in fact correct. Consider \mathbf{M}_k^{-1} of the following form:

$$\mathbf{M}_k^{-1} = \mathbf{I} + \boldsymbol{\alpha}^{(k)} \mathbf{e}_k^T. \quad (14)$$

We may prove this form is indeed the desired inverse, as follows. Using the definition of \mathbf{M}_k^{-1} from (14), we have

$$\begin{aligned} \mathbf{M}_k^{-1} \mathbf{M}_k &= (\mathbf{I} + \boldsymbol{\alpha}^{(k)} \mathbf{e}_k^T)(\mathbf{I} - \boldsymbol{\alpha}^{(k)} \mathbf{e}_k^T) \\ &= \mathbf{I} - \boldsymbol{\alpha}^{(k)} \mathbf{e}_k^T + \boldsymbol{\alpha}^{(k)} \mathbf{e}_k^T - \boldsymbol{\alpha}^{(k)} \underbrace{\mathbf{e}_k^T \boldsymbol{\alpha}^{(k)}}_0 \mathbf{e}_k^T \\ &= \mathbf{I}. \end{aligned} \quad (15)$$

From (5), $\boldsymbol{\alpha}^{(k)}$ has non-zero elements only for those indices which are greater than k , (i.e., below the main diagonal position). The only nonzero element of \mathbf{e}_k^T is in the k^{th} position. Therefore, $\mathbf{e}_k^T \boldsymbol{\alpha}^{(k)} = 0$ as indicated. Thus \mathbf{M}_k^{-1} is given by (14). We therefore see, that by looking at the structure of \mathbf{M}_k carefully, we can perform the inversion operation simply by inverting a set of signs!

8.5.2 Structure of $\mathbf{L} = \prod_k \mathbf{M}_k^{-1}$

From (9) we have

$$\begin{aligned} \mathbf{L} &= (\mathbf{M}_{n-1}, \dots, \mathbf{M}_1)^{-1} \\ &= \mathbf{M}_1^{-1}, \dots, \mathbf{M}_{n-1}^{-1} \\ &= \prod_{i=1}^{n-1} (\mathbf{I} + \boldsymbol{\alpha}^{(i)} \mathbf{e}_i^T) \end{aligned} \quad (16)$$

where the last line follows from (14). Eq. (16) may be expressed as

$$\mathbf{L} = \mathbf{I} + \sum_{k=1}^{n-1} \boldsymbol{\alpha}^{(k)} \mathbf{e}_k^T + \text{cross-products of the form } \boldsymbol{\alpha}^{(i)} \mathbf{e}_i^T \cdot \boldsymbol{\alpha}^{(j)} \mathbf{e}_j^T \quad (17)$$

Using similar reasoning to that used in (15), it may be shown that the cross-product terms in (17) are all zero. Therefore

$$\mathbf{L} = \mathbf{I} + \sum_{i=1}^{n-1} \boldsymbol{\alpha}^{(i)} \mathbf{e}_i^T. \quad (18)$$

Each term $\boldsymbol{\alpha}^{(k)} \mathbf{e}_k^T$ in (18) is a square matrix of zeros except below the main diagonal of the k^{th} column. Thus the addition operation in (18) in effect inserts the elements of $\boldsymbol{\alpha}^{(k)}$ in the k^{th} column below the main diagonal of \mathbf{L} , for $k = 1, \dots, n-1$. The addition of \mathbf{I} in (18) puts 1's on the main diagonal to complete the formulation of \mathbf{L} . We therefore note that \mathbf{L} is lower triangular with ones along the main diagonal. This form of matrix is called *unit lower triangular*.

As an example, we note from (6) that \mathbf{L} has the following structure, for $n = 4$:

$$\mathbf{L} = \begin{bmatrix} 1 & & & \\ \frac{a_{21}^{(0)}}{a_{11}^{(0)}} & 1 & & \\ \frac{a_{31}^{(0)}}{a_{11}^{(0)}} & \frac{a_{31}^{(1)}}{a_{22}^{(1)}} & 1 & \\ \frac{a_{41}^{(0)}}{a_{11}^{(0)}} & \frac{a_{42}^{(1)}}{a_{22}^{(1)}} & \frac{a_{43}^{(2)}}{a_{33}^{(2)}} & 1 \end{bmatrix}$$

Thus, given the sequence of Gauss transformations $\mathbf{M}_1 \dots \mathbf{M}_{n-1}$, we can form the factor \mathbf{L} without any explicit computations. The inverses are accomplished simply by inverting a set of signs, and the multiplication is performed by placing the nonzero elements of the $\alpha^{(k)}$'s into their respective positions in \mathbf{L} . With this simple formulation of \mathbf{L} , and the matrix \mathbf{U} given by (2), the relationship between Gaussian elimination and LU decomposition is complete.

8.5.3 Discussion and examples

Notes:

1. Note that in performing the sequence of Gauss transformations, we are performing *exactly the same arithmetic operations* as with elementary Gaussian elimination.
2. LU decomposition is a “high-level” description of Gaussian elimination. Matrix-level descriptions highlight connections between algorithms that may appear quite different at the scalar level.
3. The Gaussian elimination process requires $O(\frac{2n^3}{3})$ flops. This is the lowest number of any triangularization technique for square matrices with no specific structure.
4. \mathbf{L} is a unit lower triangular matrix. Since the determinant of a triangular matrix is the product of its diagonal elements, $\det(\mathbf{L}) = 1$. But since $\det(\mathbf{A}) = \det(\mathbf{U}) \cdot \det(\mathbf{L})$,

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

This gives us a faster way of computing a determinant.

Example 1:

Let

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 \\ 2 & -2 & 1 \\ -2 & -1 & 5 \end{bmatrix}$$

We will apply Gauss transforms to effect the LU decomposition of \mathbf{A} .

By inspection,

$$\mathbf{M}_1 = \begin{bmatrix} 1 & & & \\ -1 & 1 & & \\ & 1 & 0 & 1 \end{bmatrix} = \mathbf{I} - \alpha_1 \mathbf{e}_1^T$$

$$\mathbf{M}_1 \mathbf{A} = \begin{bmatrix} 2 & -1 & 0 \\ 0 & -1 & 1 \\ 0 & -2 & 5 \end{bmatrix} = \mathbf{A}^{(2)}$$

Thus,

$$\mathbf{M}_2 = \begin{bmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & -2 & 1 & \end{bmatrix}$$

and

$$\mathbf{M}_2 \mathbf{A}^{(2)} = \begin{bmatrix} 2 & -1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & 3 \end{bmatrix} = \mathbf{U}.$$

What is $\mathbf{L} = \mathbf{M}^{-1}$?

$$\mathbf{L} = \mathbf{M}_1^{-1} \mathbf{M}_2^{-1} = \mathbf{I} + \sum_{i=1}^2 \alpha^{(i)} \mathbf{e}_i^T$$

Thus,

$$\mathbf{L} = \begin{bmatrix} 1 & & & \\ \left(\begin{array}{c} 1 \\ -1 \end{array} \right) & 1 & & \\ \uparrow & \uparrow & & \\ \alpha^{(1)} & \alpha^{(2)} & & \end{bmatrix}$$

Note that \mathbf{LU} does in fact = \mathbf{A} , and that $\det(\mathbf{A}) = \prod u_{ii} = -6$.

8.6 Numerical properties of Gaussian Elimination

A significant amount of difficult analysis² leads to this rather simple conclusion:

Let $\hat{\mathbf{L}}\hat{\mathbf{U}}$ be the computed LU decomposition of $\mathbf{A} \in \mathfrak{R}^{n \times n}$. Then $\hat{\mathbf{y}}$ is the computed solution to $\hat{\mathbf{L}}\hat{\mathbf{y}} = \mathbf{b}$, and $\hat{\mathbf{m}}\hat{\mathbf{x}}$ the computed solution to $\hat{\mathbf{U}}\hat{\mathbf{x}} = \hat{\mathbf{y}}$. Then,

$$(\mathbf{A} + \mathbf{E})\hat{\mathbf{x}} = \mathbf{b},$$

where

$$|\mathbf{E}| \leq nu [3|\mathbf{A}| + 5|\mathbf{L}||\mathbf{U}|] + O(u^2). \quad (19)$$

This analysis shows that $\hat{\mathbf{x}}$ exactly satisfies a perturbed system. The question is whether the perturbation $|\mathbf{E}|$ is always small. If $|\mathbf{E}|$ is of the order induced by floating point representation alone, we may conclude that Gaussian elimination yields a solution which is as accurate as possible in the face of floating point error. (This is the error described in Sect. 6 in conjunction with condition number.) But further inspection reveals that (19) does not allow such an optimistic outlook. It may happen during the course of the Gaussian elimination procedure that the term $|\mathbf{L}||\mathbf{U}|$ may become large, if small pivot elements are encountered. To illustrate, we consider an example:

Example 2:

Here we show a case where a small pivot can create large $|\mathbf{L}|$ and $|\mathbf{U}|$, resulting in a grossly inaccurate result. Consider the following system of equations, where we work with base 10 arithmetic with $t = 3$ digits, with *chopping* :

$$\begin{array}{l} \text{small} \\ \text{pivot} \end{array} \rightarrow \begin{bmatrix} .001 & 1.00 \\ 1.00 & 2.00 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.00 \\ 3.00 \end{bmatrix}$$

²Golub and Van Loan, Sect. 3.3.2, 2nd. Ed.

Computing the LU decomposition with 3 digits, we get:

$$\hat{\mathbf{L}} = \begin{bmatrix} 1 & 0 \\ 1000 & 1 \end{bmatrix} \quad \hat{\mathbf{U}} = \begin{bmatrix} .001 & 1 \\ 0 & -1000 \end{bmatrix}$$

Note the presence of large elements in \mathbf{L} and \mathbf{U} due to large pivots. By multiplying the factors together we have

$$\hat{\mathbf{L}}\hat{\mathbf{U}} = \begin{bmatrix} .001 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} .001 & 1 \\ 1 & 2 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & -2 \end{bmatrix} \quad (20)$$

$$= \underset{\text{(true)}}{\mathbf{A}} + \underset{\text{(error)}}{\mathbf{H}} \quad (21)$$

Thus, the computed solution is in gross error. The calculated solution (using 3-digit arithmetic) is $\hat{\mathbf{x}} = (0, 1)^T$, whereas the true solution is $\mathbf{x} = (1.002, 0.998)^T$ (to 3 digits).

Recall that \mathbf{M}_k has the form

$$\mathbf{M}_k = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \mathbf{0} \\ \mathbf{0} & & -l_{k+1,k} & 1 & \\ & & \vdots & & \ddots \\ & & -l_{n,k} & & 1 \end{bmatrix} \begin{array}{l} \leftarrow k^{\text{th}} \text{ row} \\ \\ \\ \uparrow \\ k^{\text{th}} \text{ column} \end{array}$$

If any pivot $a_{kk}^{(k-1)}$ is small in magnitude, then the k th column of \mathbf{M}_k is large in magnitude. Because \mathbf{M}_k premultiplies $\mathbf{A}^{(k-1)}$, large elements in \mathbf{M}_k will result in large elements in the block $\mathbf{A}_{22}^{(k)}$ of (3). The result is that both \mathbf{U} and \mathbf{L} will have large elements, as k varies over its range from $1, \dots, n-1$. Hence, $|\mathbf{E}|$ in (19) is “large”, resulting in an inaccurate solution.

The fact that large $|\mathbf{L}|$ and $|\mathbf{U}|$ lead to an unstable solution can also be explained in a different way as follows. Consider two different LU decompositions on the same matrix \mathbf{A} :

1. $\mathbf{A} = \mathbf{L}\mathbf{U}$ (large pivots)

2. $\mathbf{A} = \mathbf{\Lambda}\mathbf{R}$ (small pivots)

Two different LU decompositions on the same matrix can exist, because it is possible to interchange rows and columns of the $\mathbf{A}_{22}^{(k-1)}$ block to place elements with either large or small magnitude as desired into the pivot position. This process is discussed in more detail later. Generally, the elements l_{ij} and u_{ij} of \mathbf{L} and \mathbf{U} respectively are small, whereas the elements λ_{ij} and r_{ij} of $\mathbf{\Lambda}$ and \mathbf{R} are large in magnitude. Consider the (i, j) th element a_{ij} of \mathbf{A} computed according to the two different decompositions. We have

$$a_{ij} = \mathbf{l}_i^T \mathbf{u}_j \quad \begin{cases} \mathbf{l}_i^T = i\text{th row of } \mathbf{L} \\ \mathbf{u}_j = j\text{th column of } \mathbf{U} \end{cases} \quad (22)$$

and

$$a_{ij} = \lambda_i^T \mathbf{r}_j \quad \left\{ \begin{array}{l} \text{likewise.} \end{array} \right. \quad (23)$$

Let the pivots in the second case be small enough so that

$$|\lambda_{ij}| \text{ and } |r_{ij}| \gg |a_{ij}|, \quad (i, j) \in [1, \dots, n]. \quad (24)$$

Eq. (23) can be written in the form

$$a_{ij} = P + N \quad (25)$$

where P , (N) is the sum of all terms in (23) which are positive (negative). But (24) implies that both $|P|, |N| \gg |a_{ij}|$. Thus, in (25), two nearly equal numbers are being subtracted, which leads to *catastrophic cancellation*, and ensuing numerical instability.

We note however, that the P and N terms corresponding to (22) do not satisfy (24), and as a result, little or no catastrophic cancellation arises from the computation of (25). In this case then, the resulting system is stable.

Thus, for stability, *large pivots* are required. Otherwise, even well-conditioned systems can have large error in the solution, when computed using Gaussian elimination.

Example 3

Here we consider the same situation as in example 2, except we have interchanged rows 1 and 2 to place a large element in the pivot position. The modified system is

$$\begin{bmatrix} 1.00 & 2.00 \\ .001 & 1.00 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3.00 \\ 1.00 \end{bmatrix}$$

Now computing the LU decomposition with 3 digits, we get:

$$\hat{\mathbf{L}} = \begin{bmatrix} 1 & 0 \\ .001 & 1 \end{bmatrix} \quad \hat{\mathbf{U}} = \begin{bmatrix} 1.00 & 2.00 \\ 0 & 0.998 \end{bmatrix}$$

Multiplying the factors together we have

$$\hat{\mathbf{L}}\hat{\mathbf{U}} = \begin{bmatrix} 1.00 & 2.00 \\ .001 & 1.00 \end{bmatrix}$$

In this case, we see the solution is exact to 3 decimal places.

8.7 Gaussian Elimination with Pivoting

The discussion above indicates that maximum numerical stability of the Gaussian elimination process occurs when the rows and columns of $\mathbf{A}_{22}^{(k-1)}$ are interchanged in such a way so that the element with largest magnitude is placed in the upper-left corner (pivot position). The following discusses the algebraic structure of this operation.

Consider permutation matrices $\mathbf{P}, \mathbf{\Pi} \in \mathfrak{R}^{n \times n}$, where \mathbf{P} is the identity matrix with permuted rows, and $\mathbf{\Pi}$ is the identity matrix with permuted columns. If \mathbf{P} is equivalent to \mathbf{I} with rows i and j interchanged, then premultiplication of \mathbf{A} by \mathbf{P} interchanges rows i and j of \mathbf{A} . Likewise, $\mathbf{A}\mathbf{\Pi}$ interchanges columns i and j .

To see how Gaussian elimination works with pivoting, suppose we have determined \mathbf{P}_i 's and $\mathbf{\Pi}_i$'s from previous stages so that at the $(k-1)$ th

stage:

$$\begin{aligned}
\mathbf{A}^{(k-1)} &= (\mathbf{M}_{k-1} \mathbf{P}_{k-1} \dots \mathbf{M}_1 \mathbf{P}_1) \mathbf{A} (\mathbf{\Pi}_1 \dots \mathbf{\Pi}_{k-1}) \\
&= \begin{bmatrix} \mathbf{A}_{11}^{(k-1)} & \mathbf{A}_{12}^{(k-1)} \\ \mathbf{0} & \mathbf{A}_{22}^{(k-1)} \end{bmatrix} \begin{matrix} k-1 \\ n-k+1 \end{matrix} \tag{26} \\
&\quad \begin{matrix} k-1 & n-k+1 \end{matrix}
\end{aligned}$$

which is analogous to the case without pivoting. $\mathbf{A}_{11}^{(k-1)}$ is upper triangular, and

$$\mathbf{A}_{22}^{(k-1)} = \begin{bmatrix} a_{kk}^{(k-1)} & \dots & a_{kn}^{(k-1)} \\ \vdots & \ddots & \vdots \\ a_{nk}^{(k-1)} & \dots & a_{nn}^{(k-1)} \end{bmatrix}$$

as before. In order to annihilate the first column below the first element in a stable way, we choose $\tilde{\mathbf{P}}_k$ and $\tilde{\mathbf{\Pi}}_k$ so that the leading element of

$$\tilde{\mathbf{P}}_k \mathbf{A}_{22}^{(k-1)} \tilde{\mathbf{\Pi}}_k$$

is the largest in absolute value of all elements of $\mathbf{A}_{22}^{(k-1)}$. Then, we find \mathbf{M}_k just like in the case without pivoting so that

$$\begin{aligned}
\mathbf{A}^{(k)} &= \mathbf{M}_k (\mathbf{P}_k \mathbf{A}^{(k-1)} \mathbf{\Pi}_k) \\
&= \begin{bmatrix} \mathbf{A}_{11}^{(k)} & \mathbf{A}_{12}^{(k)} \\ \mathbf{0} & \mathbf{A}_{22}^{(k)} \end{bmatrix} \begin{matrix} k \\ n-k \end{matrix} \\
&\quad \begin{matrix} k & n-k \end{matrix}
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{P}_k &= \text{diag}(\mathbf{I}_{n-k}, \tilde{\mathbf{P}}_k) \\
\mathbf{\Pi}_k &= \text{diag}(\mathbf{I}_{n-k}, \tilde{\mathbf{\Pi}}_k).
\end{aligned}$$

This explains how Gaussian elimination works with pivoting. In order to complete this analysis, we must consider how to recover \mathbf{L} . This is accomplished in the following way.³

³The assistance of Nima Ahmadvand is gratefully acknowledged.

After the k th stage with pivoting we have

$$\mathbf{A}^{(k)} = \mathbf{M}_k \mathbf{P}_k \mathbf{M}_{k-1} \mathbf{P}_{k-1} \dots \mathbf{M}_1 \mathbf{P}_1 \mathbf{A} \mathbf{\Pi}_1 \mathbf{\Pi}_2 \dots \mathbf{\Pi}_k \quad (27)$$

We can rewrite this, noting that $\mathbf{P}_i^2 = \mathbf{I}$ as

$$\begin{aligned} \mathbf{A}^{(k)} &= \mathbf{M}_k \mathbf{P}_k \mathbf{M}_{k-1} (\mathbf{P}_k \cdot \mathbf{P}_k) \mathbf{P}_{k-1} \mathbf{M}_{k-2} (\mathbf{P}_{k-1} \mathbf{P}_k \cdot \mathbf{P}_k \mathbf{P}_{k-1}) \\ &\quad \cdot \mathbf{P}_{k-2} \mathbf{M}_{k-3} \dots \mathbf{P}_2 \mathbf{M}_1 \\ &\quad \cdot (\mathbf{P}_2 \mathbf{P}_3 \dots \mathbf{P}_k \cdot \mathbf{P}_k \dots \mathbf{P}_3 \mathbf{P}_2) \\ &\quad \cdot \mathbf{P}_1 \mathbf{A} (\mathbf{\Pi}_1 \dots \mathbf{\Pi}_k) \end{aligned} \quad (28)$$

The above can be re-grouped as

$$\begin{aligned} \mathbf{A}_{(k)} &= (\mathbf{M}_k) (\mathbf{P}_k \mathbf{M}_{k-1} \mathbf{P}_k) (\mathbf{P}_k \mathbf{P}_{k-1} \mathbf{M}_{k-2} \mathbf{P}_{k-1} \mathbf{P}_k) \\ &\quad \dots (\mathbf{P}_k \mathbf{P}_{k-1} \dots \mathbf{P}_2 \mathbf{M}_1 \mathbf{P}_2 \dots \mathbf{P}_{k-1} \mathbf{P}_k) \\ &\quad \cdot \mathbf{P}_k \mathbf{P}_{k-1} \dots \mathbf{P}_2 \mathbf{P}_1 \mathbf{A} (\mathbf{\Pi}_1 \dots \mathbf{\Pi}_k). \end{aligned} \quad (29)$$

Now let us define

$$\mathbf{M}'_i = \mathbf{P}_k \mathbf{P}_{k-1} \dots \mathbf{P}_{i+1} \mathbf{M}_i \mathbf{P}_{i+1} \dots \mathbf{P}_{k-1} \mathbf{P}_k. \quad (30)$$

Then after $n - 1$ stages we have

$$\mathbf{A}_{(n-1)} = \underbrace{\mathbf{M}'_{n-1} \mathbf{M}'_{n-2} \dots \mathbf{M}'_1}_{\mathbf{L}^{-1}} \mathbf{P} \mathbf{A} \mathbf{\Pi} = \mathbf{U} \quad (31)$$

where

$$\mathbf{P} = \mathbf{P}_{n-1} \dots \mathbf{P}_1 \quad (32)$$

and

$$\mathbf{\Pi} = \mathbf{\Pi}_1 \dots \mathbf{\Pi}_{n-1}. \quad (33)$$

From (31) we are left with

$$\mathbf{L} \mathbf{U} = \mathbf{P} \mathbf{A} \mathbf{\Pi}. \quad (34)$$

We therefore see that the product of the \mathbf{LU} factors as defined above for the row-column pivoting case yields a row-column permuted version of \mathbf{A} .

It only remains to show that \mathbf{L}^{-1} defined by (31) is indeed unit lower triangular as it should be. It is sufficient to show that \mathbf{M}'_i in (30) is unit lower triangular. We note that the permutation matrices associated with

M'_i have indices greater than i . This means that the P 's interchange rows of M_i which have indices greater than i . Suppose P_{i+1} interchanges rows $l, m > i$. Then, by simply interchanging the l th and m th rows and columns of M_i , it is easy to verify that the quantity $P_{i+1}M_iP_{i+1}$ is the same as M_i , but with elements l_{kl} and l_{km} interchanged. Thus, $P_{i+1}M_iP_{i+1}$ is unit lower triangular. By extension, so are all M'_i in (30); hence L^{-1} is lower triangular, and so is L .

Finally, to solve the pivotted system, we have

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{35}$$

or

$$PLU\Pi\mathbf{x} = \mathbf{b}. \tag{36}$$

Let us define $\mathbf{y} = U\Pi\mathbf{x}$ and $\mathbf{z} = \Pi\mathbf{x}$. Then (36) can be solved using the following sequence:

$$\begin{aligned} L\mathbf{y} &= P\mathbf{b} \\ U\mathbf{z} &= \mathbf{y} \\ \mathbf{x} &= \Pi\mathbf{z} \end{aligned}$$

The above process is a reflection of the fact that corresponding elements of \mathbf{b} must be be interchanged if any pair of rows of \mathbf{A} are permuted by P . Likewise, elements of \mathbf{x} must be be interchanged if any pair of columns of \mathbf{A} are permuted by Π .

8.8 Partial Pivoting

Full pivoting as we described is stable, yet expensive since arithmetic comparisons are almost as costly as flops, and many comparisons are required to complete the Gaussian elimination process with pivoting.

The number of comparisons can be drastically reduced if only row permutations take place. That is, the element in the first column of $\mathbf{A}_{22}^{(k-1)}$ which is largest in magnitude is permuted into the pivot position. The result, which is known as *partial pivoting*, is almost as stable. The algebraic description of the partial pivoting process is almost the same as that for full pivoting, except the Π -matrices disappear.

8.9 Heuristics of Gaussian Elimination

Because *pivoted* Gaussian elimination is stable, results of previous sections lead us to the conclusion that the computed solution $\hat{\mathbf{x}}$ to $\mathbf{Ax} = \mathbf{b}$ satisfies

$$(\mathbf{A} + \mathbf{E})\hat{\mathbf{x}} = \mathbf{b} \quad (37)$$

where

$$\frac{\|\mathbf{E}\|_\infty}{\|\mathbf{A}\|_\infty} \leq O(\beta^{-t}) \quad (38)$$

where $\beta =$ machine base, usually 2, and t is the number of base- β digits, and “ $O(\cdot)$ ” is “order” notation, which indicates the corresponding expression is a loose bound, to the nearest “order of magnitude”. Eq. (38) says that the computed solution $\hat{\mathbf{x}}$ is the exact solution to a nearby system, which is perturbed by $O(u)$ from the true system. This is the characteristic of any stable numerical procedure. Furthermore, from our discussion on *condition number*, the relative error in the solution can be expressed as $\kappa(\mathbf{A}) \times$ relative error in \mathbf{A} . Using (38) as the relative error in \mathbf{A} , we have

$$\frac{\|\hat{\mathbf{x}} - \mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty} \leq O(\beta^{-t})\kappa_\infty(\mathbf{A}), \quad (39)$$

Now suppose $\kappa_\infty(\mathbf{A}) \cong \beta^q$. Then from (38) and (39), we can say the following about the residual error $\mathbf{r} = \mathbf{b} - \mathbf{A}\hat{\mathbf{x}}$, and the computed solution $\hat{\mathbf{x}}$:

$$\begin{aligned} \|\mathbf{r}\|_\infty &= \|\mathbf{b} - \mathbf{A}\hat{\mathbf{x}}\|_\infty \\ &= \|(\mathbf{A}\hat{\mathbf{x}} + \mathbf{E}\hat{\mathbf{x}}) - \mathbf{A}\hat{\mathbf{x}}\|_\infty \\ &= \|\mathbf{E}\hat{\mathbf{x}}\|_\infty \\ &\leq \|\mathbf{E}\|_\infty \|\hat{\mathbf{x}}\|_\infty \\ &\leq O(\beta^{-t}) \|\mathbf{A}\|_\infty \|\hat{\mathbf{x}}\|_\infty \end{aligned} \quad (40)$$

and

$$\frac{\|\hat{\mathbf{x}} - \mathbf{x}\|_\infty}{\|\mathbf{x}\|_\infty} \simeq O(\beta^{q-t}). \quad (41)$$

Eq's (40) and (41) lead to the following **important heuristics**:

1. Gaussian elimination (with pivoting) produces a solution $\hat{\mathbf{x}}$ with relatively small residuals $\mathbf{r} = \mathbf{Ax} - \mathbf{b}$. Thus, regardless of how poor the condition number is, the residuals are usually small.
2. Gaussian elimination produces a solution $\hat{\mathbf{x}}$ that has about $t \log_{10} \beta - \log_{10} [\kappa_{\infty}(\mathbf{A})]$ correct decimal digits. Since a floating point number has at best $t \log_{10} \beta$ decimal digits (from the discussion on floating point representations), and if $\kappa_{\infty}(\mathbf{A}) = \beta^q$, then $\log_{10} [\kappa_{\infty}(\mathbf{A})]$ decimal digits are lost in the computed solution.

8.10 Iterative Improvement

It is possible to improve the number of significant digits in the solution $\hat{\mathbf{x}}$ given by Heuristic 2, provided the conditioning is not too bad, by *iterative refinement*. Specifically, suppose the system $\mathbf{Ax} = \mathbf{b}$ has been solved via $\mathbf{PA} = \mathbf{LU}$ to give $\hat{\mathbf{x}}$. The residual \mathbf{r} is then

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\hat{\mathbf{x}}. \quad (42)$$

One may well ask, “What is the vector \mathbf{z} for which $\mathbf{Az} = \mathbf{r}$ ”. Then \mathbf{z} is the error in \mathbf{x} which corresponds to the residual \mathbf{r} . It is evident that \mathbf{z} may then be obtained through

$$\begin{aligned} \mathbf{Ly} &= \mathbf{Pr} \\ \mathbf{Uz} &= \mathbf{y}. \end{aligned}$$

Then, an *improved* \mathbf{x} , \mathbf{x}_{new} , may be given by

$$\mathbf{x}_{\text{new}} = \hat{\mathbf{x}} + \mathbf{z}$$

In the ideal case, the quantity \mathbf{z} would be precisely the error in $\hat{\mathbf{x}}$, and \mathbf{x}_{new} would be exactly the correct solution. However, because we are working in finite precision, there is one flaw in the above argument. That is, the two terms on the right-hand side of (42) are nearly equal; hence, \mathbf{r} is subject

to significant catastrophic cancellation, and \mathbf{r} given by (42) has very few correct digits.

However, the following *iterative* scheme works well, even in finite precision arithmetic, provided $\kappa(\mathbf{A}) \simeq < \beta^q$:

$$\begin{aligned} \mathbf{x} &:= 0 && \text{(single precision)} \\ \mathbf{P}\mathbf{A} &= \mathbf{L}\mathbf{U} && \text{(single precision)} \end{aligned}$$

$$\begin{aligned} &\text{Repeat} \\ \mathbf{r} &:= \mathbf{b} - \mathbf{A}\mathbf{x} && \text{(double precision)} \\ \text{solve } \mathbf{L}\mathbf{y} &= \mathbf{P}\mathbf{r} \text{ for } \mathbf{y} && \text{(single precision)} \\ \text{solve } \mathbf{U}\mathbf{z} &= \mathbf{y} \text{ for } \mathbf{z} && \text{(single precision)} \\ \mathbf{x} &:= \mathbf{x} + \mathbf{z} && \text{(single precision)} \end{aligned}$$

Note that this algorithm is essentially the same as the one described, except the computation of \mathbf{r} is done in *double precision*, to yield a single precision result for \mathbf{x} . The above algorithm leads to the following 3rd heuristic:

* **Heuristic 3:**

With t -digit base- β arithmetic and $\kappa_\infty(\mathbf{A}) \simeq \beta^q$, then after k passes through the loop, \mathbf{x} will have approximately $\min[t, k(t - q)]$ correct base- β digits.

Thus we see that by iterative refinement, we can produce a solution to full single precision accuracy, provided $\kappa(\mathbf{A})$ is not too large.

9 The Cholesky Decomposition

reference: *Golub and van Loan Sections 4.1 and 4.2*

We now consider several modifications to the LU decomposition, which ultimately lead up to the Cholesky decomposition. These modifications are 1) the LDM decomposition, 2) the LDL decomposition on symmetric matrices, and 3) the LDL decomposition on positive definite symmetric matrices.

The Cholesky decomposition is relevant only for square symmetric positive-definite matrices and an is important concent in signal processing. Several examples of the use of the Cholesky decomposition are provided at the end of the section.

9.1 The LDM Factorization

If no zero pivots are encountered during the Gaussian elimination process, then there exist *unit* lower triangular matrices \mathbf{L} and \mathbf{M} and a diagonal matrix \mathbf{D} such that

$$\mathbf{A} = \mathbf{LDM}^T \quad (43)$$

Justification:

Since $\mathbf{A} = \mathbf{LU}$ exists, let $\mathbf{U} = \mathbf{DM}^T$ be upper triangular, where $d_i = u_{ii}$: hence, $\mathbf{A} = \mathbf{LDM}^T$ which was to be shown. Each row of \mathbf{M}^T is the corresponding row of \mathbf{U} divided by its diagonal element.

We then solve the system $\mathbf{Ax} = \mathbf{b}$ which is equivalent to $\mathbf{LDM}^T\mathbf{x} = \mathbf{b}$ in three steps:

1. let $\mathbf{y} = \mathbf{DM}^T\mathbf{x}$, and solve $\mathbf{Ly} = \mathbf{b}$ (n^2 flops)
2. let $\mathbf{z} = \mathbf{M}^T\mathbf{x}$, and solve $\mathbf{Dz} = \mathbf{y}$ (n flops)
3. solve $\mathbf{M}^T\mathbf{x} = \mathbf{z}$ (n^2 flops)

9.1.1 Error Analysis:

The computed solution $\hat{\mathbf{x}}$ to $\mathbf{Ax} = \mathbf{b}$ satisfies:

$$(\mathbf{A} + \mathbf{E})\hat{\mathbf{x}} = \mathbf{b}$$

where

$$|\mathbf{E}| \leq nu \left[3|\mathbf{A}| + 5|\hat{\mathbf{L}}| |\hat{\mathbf{D}}| |\hat{\mathbf{M}}| \right] + O(u^2) \quad (44)$$

Thus, pivoting is required for this case, to prevent growth in $|\hat{\mathbf{L}}|, |\hat{\mathbf{D}}|$ or $|\hat{\mathbf{M}}|$. This result is analogous to ordinary Gaussian elimination.

9.2 The LDL Decomposition for Symmetric Matrices

For a *symmetric* non-singular matrix $\mathbf{A} \in \mathfrak{R}^{n \times n}$, the factors \mathbf{L} and \mathbf{M} are identical.

Proof:

Let $\mathbf{A} = \mathbf{LDM}^T$. The matrix $\mathbf{M}^{-1}\mathbf{A}\mathbf{M}^{-T} = \mathbf{M}^{-1}\mathbf{LD}$ is symmetric (from left-hand side), and lower triangular (from RHS). Hence, they are *both* diagonal.

But \mathbf{D} is nonsingular, so $\mathbf{M}^{-1}\mathbf{L}$ is also diagonal. The matrices \mathbf{M} and \mathbf{L} are both unit lower triangular (ULT). It can be easily shown that the inverse of a ULT matrix is also ULT, and furthermore, the product of ULT's is ULT. Therefore \mathbf{M}^{-1} is ULT, and so is $\mathbf{M}^{-1}\mathbf{L}$. Thus, $\mathbf{M}^{-1}\mathbf{L} = \mathbf{I}$; $\mathbf{M} = \mathbf{L}$. \square

This means that for a symmetric matrix \mathbf{A} , the LU factorization requires only $\frac{n^3}{3}$ flops, instead of $\frac{2}{3}n^3$ as for the general case. This is because only the lower factor need be computed.

9.3 For Positive Definite Systems:

9.3.1 Error Analysis on Positive-Definite Matrices

Let $\mathbf{A} \in \mathfrak{R}^{n \times n}$ be positive definite, with $\mathbf{A} = \mathbf{LDM}^T$. Define the symmetric part \mathbf{T} and the asymmetric part \mathbf{S} of \mathbf{A} respectively as:

$$\mathbf{T} = \frac{\mathbf{A} + \mathbf{A}^T}{2}, \quad \mathbf{S} = \frac{\mathbf{A} - \mathbf{A}^T}{2}$$

It is shown by *Golub and van Loan* that

$$\left\| |\mathbf{L}| |\mathbf{D}| |\mathbf{M}^T| \right\|_F \leq n \left[\|\mathbf{T}\|_2 + \left\| \mathbf{S}\mathbf{T}^{-1}\mathbf{S} \right\|_2 \right]. \quad (45)$$

Let $\hat{\mathbf{L}}, \hat{\mathbf{D}}, \hat{\mathbf{M}}^T$ be the computed factors. We assume:

$$\left\| |\hat{\mathbf{L}}| |\hat{\mathbf{D}}| |\hat{\mathbf{M}}|^T \right\|_F \leq c \left\| |\mathbf{L}| |\mathbf{D}| |\mathbf{M}^T| \right\|_F \quad (46)$$

where c is a constant of modest size.

We can now use (45) and (46) in (44), to get the result that the computed solution $\hat{\mathbf{x}}$ satisfies

$$(\mathbf{A} + \mathbf{E})\hat{\mathbf{x}} = \mathbf{b}$$

where

$$\|\mathbf{E}\|_F \leq u \left\{ 3n\|\mathbf{A}\|_F + 5cn^2 \left[\|\mathbf{T}\|_2 + \left\| \mathbf{S}\mathbf{T}^{-1}\mathbf{S} \right\|_2 \right] \right\} + O(u^2) \quad (47)$$

Eq. (47) is an important result. For a symmetric matrix, $\left\| \mathbf{S}\mathbf{T}^{-1}\mathbf{S} \right\|_2$ is zero and the \mathbf{E} matrix for the bound (47) for symmetric, positive-definite \mathbf{A} is on the order of the error introduced by floating-point representation alone. Also, since it is independent of the factors \mathbf{L} , \mathbf{D} or \mathbf{M} , the bound (47) is stable *without pivoting*.

Putting the discussion for symmetric and positive-definite matrices together, we have the following:

9.4 Cholesky Decomposition:

For $\mathbf{A} \in \Re^{n \times n}$ symmetric and positive definite, there exists a lower triangular matrix $\mathbf{G} \in \Re^{n \times n}$ with positive diagonal entries, such that $\mathbf{A} = \mathbf{G}\mathbf{G}^T$.

Proof:

Consider \mathbf{A} which is positive definite and symmetric. Note that covariance matrices fall into this class. Therefore, $\mathbf{x}^T \mathbf{A} \mathbf{x} > 0, \mathbf{0} \neq \mathbf{x} \in \Re^{n \times n}$, and hence $\mathbf{x}^T \mathbf{L} \mathbf{D} \mathbf{L}^T \mathbf{x} > 0$. If \mathbf{A} is positive definite, then \mathbf{L} is full rank; let $\mathbf{y} \triangleq \mathbf{L}^T \mathbf{x}$. Then, $\mathbf{y}^T \mathbf{D} \mathbf{y} > 0$, *if and only if* all elements of \mathbf{D} are positive. Therefore, if \mathbf{A} is positive definite, then $d_{ii} > 0, i = 1 \dots, n$.

Because \mathbf{A} is symmetric, then $\mathbf{A} = \mathbf{L} \mathbf{D} \mathbf{L}^T$. Because the d_{ii} are positive, then $\mathbf{G} = \mathbf{L} \cdot \text{diag}(\sqrt{d_{11}}, \dots, \sqrt{d_{nn}})$. Then $\mathbf{G}\mathbf{G}^T = \mathbf{A}$ as desired.

□

As discussed earlier, this decomposition is stable *without pivoting*.

Therefore, in solving the system $\mathbf{Ax} = \mathbf{b}$, where \mathbf{A} is symmetric and positive definite (e.g., for the case where \mathbf{A} is a sample covariance matrix), the Cholesky decomposition requires only half the flops for the LU decomposition phase of the computation, and does not require pivoting. Both these factors significantly reduce the execution time of the algorithm.

9.4.1 Computation of the Cholesky Decomposition

An algorithm for computing the Cholesky decomposition is developed simply by direct comparison: e.g. in the 3×3 case we have:

$$\begin{bmatrix} g_{11} & & \\ g_{21} & g_{22} & \\ g_{31} & g_{32} & g_{33} \end{bmatrix} = \begin{bmatrix} g_{11} & g_{21} & g_{31} \\ & g_{22} & g_{32} \\ & & g_{33} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}$$

symmetric,
positive definite.

By following a proper order, each element of \mathbf{G} may be determined in sequence, simply by comparing a particular element a_{ij} of \mathbf{A} with the inner product $\mathbf{g}_i^T \mathbf{g}_j$, where \mathbf{g}_i^T is taken to be the i th row of \mathbf{G} . First, we may determine g_{11} by comparison with a_{11} . Then, all remaining elements of the first column of \mathbf{G} may be determined once g_{11} is known. Then, g_{22} can be determined, and the process repeats. For example,

$$g_{11}^2 = a_{11} \rightarrow g_{11} = \sqrt{a_{11}}$$

Also,

$$g_{i1} = \frac{a_{i1}}{g_{11}} \quad i = 2, \dots, n.$$

Thus, all elements in first column of \mathbf{G} can be solved. Now, consider second column. First, we solve g_{22} :

$$g_{21}^2 + g_{22}^2 = a_{22}$$

Thus,

$$g_{22} = (a_{22} - g_{21}^2)^{\frac{1}{2}}$$

where the term in the round brackets is positive if \mathbf{A} is positive definite. Once g_{22} is determined, all remaining elements in the second column may be found by comparison with corresponding element in the second column of \mathbf{A} . The third and remaining columns are solved in a similar way. If the process works its way in turn through columns $1, \dots, n$, each element in \mathbf{G} is found by solving a single equation in one unknown. Determining each diagonal element involves finding a square root of a particular quantity. This quantity is always positive if \mathbf{A} is positive definite.

If $\mathbf{A} = \mathbf{G}\mathbf{G}^T$, then to solve $\mathbf{A}\mathbf{x} = \mathbf{b}$ we solve

$$\mathbf{G}\mathbf{z} = \mathbf{b} \text{ for } \mathbf{z}$$

then

$$\mathbf{G}^T \mathbf{x} = \mathbf{z} \text{ for } \mathbf{x}$$

9.4.2 Discussion on the Cholesky Decomposition

1. If the positive square root is always taken in the computation of the Cholesky factorization, then the Cholesky factorization is unique.
2. The Cholesky decomposition $\mathbf{A} = \mathbf{G}\mathbf{G}^T$ is a matrix analog of a scalar square-root operation. Note however that this square root is not unique. The matrix $\mathbf{G}\mathbf{Q}$, where \mathbf{Q} is any orthonormal matrix of dimension $n \times k$, where $k \geq n$ is also a square root factor. Another square root matrix of \mathbf{A} is given as $\mathbf{V} \cdot \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)^{1/2}$, where the \mathbf{v}_i and λ_i are the eigenvectors and eigenvalues of \mathbf{A} , respectively. The uniqueness of the Cholesky factor is a result of it being lower triangular with positive diagonal elements.
3. Suppose we have a random vector process $\mathbf{x}_i \in \mathbb{R}^n$, $i = 1, \dots, m$, $m > n$. We can form a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ as

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_m^T \end{bmatrix} = \begin{bmatrix} \text{---} \\ \text{---} \\ \vdots \\ \text{---} \end{bmatrix} \begin{array}{l} \text{each row} \\ \text{is a} \\ \text{sample } x_i \end{array}$$

Then an estimate of the covariance matrix \mathbf{R} is

$$\hat{\mathbf{R}} = \mathbf{X}^T \mathbf{X}.$$

where the normalizing $1/n$ factor has been ignored. Later in this course, we learn about the QR decomposition of a matrix. We will learn that any matrix \mathbf{X} can be factored as follows:

$$\mathbf{X} = \underset{\substack{\uparrow \\ m \times m \text{ orthonormal}}}{\mathbf{Q}} \mathbf{U} \quad \mathbf{U} = \begin{bmatrix} \mathbf{U}_o \\ \mathbf{0} \end{bmatrix} \begin{matrix} n \\ m-n \\ n \end{matrix}$$

where \mathbf{U}_o is upper triangular. Then, \mathbf{U}_o^T is the Cholesky factor of $\hat{\mathbf{R}}$ since

$$\hat{\mathbf{R}} = \mathbf{X}^T \mathbf{X} = \mathbf{U}^T \mathbf{Q}^T \mathbf{Q} \mathbf{U} = \mathbf{U}^T \mathbf{U} = \mathbf{U}_o^T \mathbf{U}_o = \mathbf{G} \mathbf{G}^T$$

Since \mathbf{U}_o is upper triangular, and the factorization is unique to within a sign change on the diagonal elements, then $\mathbf{G} = \mathbf{U}_o^T$. This means that the matrix \mathbf{U}_o which results from the QR decomposition of \mathbf{X} is the transpose of the Cholesky factor of $\hat{\mathbf{R}}$.

9.5 Applications and Examples of the Cholesky Decomposition

9.5.1 Generating vector processes with desired covariance

We may use the Cholesky decomposition to generate a random vector process $\mathbf{x} \in \mathfrak{R}^n$ with a desired covariance matrix $\mathbf{\Sigma} \in \mathfrak{R}^{n \times n}$. Since $\mathbf{\Sigma}$ must be symmetric and positive definite, let

$$\mathbf{\Sigma} = \mathbf{G} \mathbf{G}^T$$

be the Cholesky factorization of $\mathbf{\Sigma}$. Let $\mathbf{w} \in \mathfrak{R}^n$ be a random vector with uncorrelated elements such that $E(\mathbf{w} \mathbf{w}^T) = \mathbf{I}$. Such \mathbf{w} 's are easily generated by random number generators on the computer.

Then, define \mathbf{x} as:

$$\mathbf{x} = \mathbf{G} \mathbf{w}$$

The vector process \mathbf{x} has the desired covariance matrix because

$$\begin{aligned} E(\mathbf{x}\mathbf{x}^T) &= E(\mathbf{G}\mathbf{w}\mathbf{w}^T\mathbf{G}^T) \\ &= \mathbf{G}E(\mathbf{w}\mathbf{w}^T)\mathbf{G}^T \\ &= \mathbf{G}\mathbf{G}^T \\ &= \mathbf{\Sigma}. \end{aligned}$$

This procedure is particularly useful for computer simulations when it is desired to create a random vector process with a specified covariance matrix.

9.5.2 Whitening a Process

This example is essentially the inverse of the one just discussed. Suppose we have a stationary vector process $\mathbf{x}_i \in \mathfrak{R}^n, i = 1, 2, \dots$. This process could be the signals received from the elements of an array of n sensors, it could be sets of n sequential samples of any time-varying signal, or sets of data in a tapped-delay line equalizer of length n , at time instants t_1, t_2, \dots , etc.

Let the process \mathbf{x} consist of a signal part \mathbf{s}_i and a noise part $\boldsymbol{\nu}_i$:

$$\mathbf{x}_i = \mathbf{s}_i + \boldsymbol{\nu}_i, \quad i = 1, 2, 3, \dots \quad (48)$$

where we assume the covariance of the noise $E(\boldsymbol{\nu}\boldsymbol{\nu}^T) \triangleq \mathbf{\Sigma}$ is not diagonal. The noise is thus correlated or coloured. This discussion requires that $\mathbf{\Sigma}$ is known or can be estimated.

Let \mathbf{G} be the Cholesky factorization of $\mathbf{\Sigma}$ such that $\mathbf{G}\mathbf{G}^T = \mathbf{\Sigma}$. Premultiply both sides of (48) with \mathbf{G}^{-1} :

$$\mathbf{G}^{-1}\mathbf{x}_i = \mathbf{G}^{-1}\mathbf{s}_i + \mathbf{G}^{-1}\boldsymbol{\nu}_i \quad (49)$$

The noise component is now $\mathbf{G}^{-1}\boldsymbol{\nu}_i$. The corresponding noise covariance matrix is

$$\begin{aligned} E(\mathbf{G}^{-1}\boldsymbol{\nu}_i\boldsymbol{\nu}_i^T\mathbf{G}^{-T}) &= \mathbf{G}^{-1}E(\boldsymbol{\nu}\boldsymbol{\nu}^T)\mathbf{G}^{-T} \\ &= \mathbf{G}^{-1}\mathbf{\Sigma}\mathbf{G}^{-T} \\ &= \mathbf{G}^{-1}\mathbf{G}\mathbf{G}^T\mathbf{G}^{-T} \\ &= \mathbf{I} \end{aligned} \quad (50)$$

Thus, by premultiplying the original signal \mathbf{x} by the inverse Cholesky factor of the noise, the resulting noise is *white*. Whitening sequences by pre-multiplying by an inverse square-root of the covariance matrix (as opposed to some other factor of $\mathbf{\Sigma}^{-1}$) is an important concept in signal processing. The inverse Cholesky factor is most commonly applied in these situations because it is stable and easy to compute.

Since the received signal $\mathbf{x} = \mathbf{s} + \boldsymbol{\nu}$, the joint probability density function $p(\mathbf{x}|\mathbf{s})$ of the received signal vector \mathbf{x} , given the noiseless signal \mathbf{s} , in the presence of Gaussian noise samples $\boldsymbol{\nu}$ with covariance matrix $\mathbf{\Sigma}$, is simply the *pdf* of the noise itself, and is given by the multi-dimensional Gaussian probability density function discussed in Sect. 5.1:

$$p(\mathbf{x}|\mathbf{s}) = (2\pi)^{\frac{n}{2}} |\mathbf{\Sigma}|^{\frac{1}{2}} \exp \left[-\frac{1}{2} (\mathbf{x} - \mathbf{s})^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mathbf{s}) \right] \quad (51)$$

In contrast, suppose we transform the vector $\mathbf{x} - \mathbf{s}$ by pre-multiplication with the inverse Cholesky factor \mathbf{G}^{-1} of $\mathbf{\Sigma}^{-1}$, to form $\mathbf{y} = \mathbf{G}^{-1}(\mathbf{x} - \mathbf{s})$. This transformation whitens the noise. From the discussion above, we see that the covariance matrix of the variable \mathbf{y} is the identity. Therefore,

$$\begin{aligned} p(\mathbf{y}) &= \frac{1}{(2\pi)^{\frac{n}{2}}} \exp \left[-\frac{1}{2} \mathbf{y}^T \mathbf{y} \right] \\ &= \frac{1}{(2\pi)^{\frac{n}{2}}} \exp \left[-y_1^2/2 \right] \dots \exp \left[-y_n^2/2 \right] \\ &= p(y_1)p(y_2) \dots p(y_n) \end{aligned} \quad (52)$$

Thus, in the case (52) where we have whitened the noise, any processing on the signal involving detection or estimation may be done by processing each whitened variable y_i independently of the rest, since these variables are shown to be independent. In contrast, if we wish to process the original signal \mathbf{x} in coloured noise, we must jointly process the entire vector \mathbf{x} , due to the fact there are dependencies amongst the elements introduced through the quadratic form in the exponent of (51). The whitening process thus significantly simplifies processing on the signal when the noise is coloured.

As a further example of the use of the Cholesky decomposition, we consider the multi-variate Gaussian *pdf* of a zero-mean random vector with covariance $\mathbf{\Sigma}$. The exponent of the distribution is then $\mathbf{x}^T \mathbf{\Sigma}^{-1} \mathbf{x} = \mathbf{x}^T \mathbf{G}^{-T} \mathbf{G}^{-1} \mathbf{x}$ where \mathbf{G} is the Cholesky factor of $\mathbf{\Sigma}$. If we let $\mathbf{w} = \mathbf{G}^{-1} \mathbf{x}$, then the exponent

of the distribution becomes $\mathbf{w}^T \mathbf{w}$. But from (50), we see that the covariance of \mathbf{w} is \mathbf{I} ; i.e., \mathbf{w} is white. Thus, we see that the matrix Σ^{-1} in the exponent of the Gaussian *pdf* has the role of transforming the original random variable \mathbf{x} into another random variable \mathbf{w} whose elements are uncorrelated with unit variance. Thus, Σ^{-1} whitens and normalizes the original variables \mathbf{x} .