2.6 Appendix

2.6.1 More examples for LU factorization

**Example LU using code** Consider the following linear system

\[
\begin{align*}
3x_1 + 4x_2 + 2x_3 &= 21 \\
10x_1 + 2x_2 + x_3 &= 53 \\
x_1 + x_2 + x_3 &= 7
\end{align*}
\]

or

\[
\begin{pmatrix}
3 & 4 & 2 \\
10 & 2 & 1 \\
1 & 1 & 1
\end{pmatrix} \times
\begin{pmatrix}
x_1 \\
x_2 \\
x_3
\end{pmatrix} =
\begin{pmatrix}
5 \\
15 \\
1
\end{pmatrix}
\]

(13)

The matrix in this example is non-singular, hence there exists a solution to the system. We will follow this example by the following Julia code. First we define the following linear system, and show its solution using Julia’s operator that solves it.

```julia
julia> A = [3.0 4 2; 10 2 1; 1 1 1]; b = [21.0; 53; 7];

julia> A
3x3 Array{Float64,2}:
3.0 4.0 2.0
10.0 2.0 1.0
1.0 1.0 1.0

julia> b
3-element Array{Float64,1}:
21.0
53.0
7.0

julia> A\b # solves the system
3-element Array{Float64,1}:
5.0
1.0
1.0
```

Now we multiply the first row by \( \frac{10}{3} \) and \( \frac{1}{3} \) (or by \( \frac{a_{11}}{a_{11}} \) and \( \frac{a_{11}}{a_{11}} \)), and subtract the answers from the second and third rows respectively.
2. Direct solution of linear systems

```julia


julia> A
3x3 Array{Float64,2}:
3.0 4.0 2.0
0.0 -11.3333 -5.66667
0.0 -0.333333 0.333333

julia> b
3-element Array{Float64,1}:
21.0
-17.0
0.0
```

Finally, we multiply the first row by \( \frac{a_{32}}{a_{22}} \) and subtract it from the third row:

```julia

julia> A
3x3 Array{Float64,2}:
3.0 4.0 2.0
0.0 -11.3333 -5.66667
0.0 0.0 0.5

julia> b
3-element Array{Float64,1}:
21.0
-17.0
0.5
```

Now we can solve the triangular system by solving the 1-variable equation for \( x_3 \), then for \( x_2 \) using \( x_3 \), and then for \( x_1 \) (this process of solving an upper triangular system from below upwards is called “backward substitution”). As expected, if we solve the triangular system we get the same answer as before.

```julia
julia> A\b
3-element Array{Float64,1}:
5.0
1.0
1.0
```

In the process above we manipulated the rows of A by a scalars \( p \), which where based on the diagonal values \( a_{ii} \) along the algorithm. Those diagonal values are called pivots. Interestingly, the process above is equivalent to “dividing” \( A \) by a lower triangular system:
2. Direct solution of linear systems

```julia
julia> A = [3.0 4 2; 10 2 1; 1 1 1];
julia> L = [1.0 0.0 0.0; p1 1.0 0.0; p2 p3 1.0]
3x3 Array{Float64,2}:
1.0 0.0 0.0
3.33333 1.0 0.0
0.333333 0.0294118 1.0
julia> L\A
3x3 Array{Float64,2}:
3.0 4.0 2.0
0.0 -11.3333 -5.66667
0.0 0.0 0.5
```

Example for an LU factorization of a singular matrix:

In this example we will see how both algorithms behave when a singular matrix $A$ is given to them. It is easy to show that if only the last few rows of $A$ are linearly dependent of the others, then we will just have a few zero lines at the bottom of $U$. However, if there is a linearly independent row after a linearly dependent row - then the LU without pivoting is expected to fail (have zero pivot). We will construct a $4 \times 4$ matrix from 3 random vectors.

Initialization: $A, U = \begin{bmatrix} 0.358 & 0.085 & 0.009 & 0.529 \\
0.057 & 0.481 & 0.328 & 0.748 \\
0.415 & 0.566 & 0.337 & 1.277 \\
0.369 & 0.108 & 0.555 & 0.062 \\
\end{bmatrix}$, $L = \begin{bmatrix} 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}$.

Note that the third row in this matrix is a sum of the first two, hence the matrix is singular. First, we will run a pivoting-less LU decomposition.

1st iter: $U = \begin{bmatrix} 0.359 & 0.085 & 0.009 & 0.529 \\
-6.9e-18 & 0.468 & 0.327 & 0.664 \\
0.0 & 0.468 & 0.327 & 0.664 \\
5.5e-17 & 0.020 & 0.546 & -0.483 \\
\end{bmatrix}$, $L = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\
0.16 & 1.0 & 0.0 & 0.0 \\
1.16 & 0.0 & 1.0 & 0.0 \\
1.03 & 0.0 & 0.0 & 1.0 \\
\end{bmatrix}$.

2nd iter: $U = \begin{bmatrix} 0.359 & 0.085 & 0.009 & 0.529 \\
-6.9e-18 & 0.468 & 0.327 & 0.664 \\
0.0 & 0.0 & 0.0 & 0.0 \\
5.5e-17 & 0.0 & 0.532 & -0.512 \\
\end{bmatrix}$, $L = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\
0.160 & 1.0 & 0.0 & 0.0 \\
1.16 & 1.0 & 1.0 & 0.0 \\
1.03 & 0.044 & 0.0 & 1.0 \\
\end{bmatrix}$.

3rd iter: $U = \begin{bmatrix} 0.359 & 0.085 & 0.009 & 0.529 \\
-6.9e-18 & 0.468 & 0.327 & 0.664 \\
0.0 & 0.0 & 0.0 & 0.0 \\
5.5e-17 & 0.0 & NaN & NaN \\
\end{bmatrix}$, $L = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\
0.160 & 1.0 & 0.0 & 0.0 \\
1.16 & 1.0 & 1.0 & 0.0 \\
1.03 & 0.044 & Inf & 1.0 \\
\end{bmatrix}$.
The algorithm failed. It reached a zero row (and pivot), and then divided by zero. With pivoting, the LU factorization will have the following results:

1st: \[ U = \begin{bmatrix} 0.415 & 0.567 & 0.338 & 1.278 \\ 0.0 & 0.403 & 0.282 & 0.572 \\ 0.0 & -0.403 & -0.282 & -0.572 \\ 0.0 & -0.395 & 0.256 & -1.073 \end{bmatrix} \quad L = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.138 & 1.0 & 0.0 & 0.0 \\ 0.862 & 0.0 & 1.0 & 0.0 \\ 0.888 & 0.0 & 0.0 & 1.0 \end{bmatrix} \quad p = \begin{bmatrix} 3 \\ 2 \\ 1 \\ 4 \end{bmatrix}, \]

2nd: \[ U = \begin{bmatrix} 0.415 & 0.567 & 0.338 & 1.278 \\ 0.0 & 0.403 & 0.282 & 0.572 \\ 0.0 & 0.0 & 1.11e-16 & -0.512 \\ 0.0 & 0.0 & 0.532 & -0.512 \end{bmatrix} \quad L = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.862 & 1.0 & 0.0 & 0.0 \\ 0.138 & -1.0 & 1.0 & 0.0 \\ 0.888 & 0.979 & 0.0 & 1.0 \end{bmatrix} \quad p = \begin{bmatrix} 3 \\ 2 \\ 1 \\ 4 \end{bmatrix}, \]

3rd: \[ U = \begin{bmatrix} 0.415 & 0.567 & 0.338 & 1.278 \\ 0.0 & -0.403 & -0.282 & -0.572 \\ 0.0 & 0.0 & 0.532 & -0.512 \\ 0.0 & 0.0 & 0.0 & 1.11e-16 \end{bmatrix} \quad L = \begin{bmatrix} 1.0 & 0.0 & 0.0 & 0.0 \\ 0.862 & 1.0 & 0.0 & 0.0 \\ 0.138 & -1.0 & 1.0 & 0.0 \\ 0.888 & 0.979 & 1.0 & 0.0 \end{bmatrix} \quad p = \begin{bmatrix} 3 \\ 1 \\ 4 \\ 2 \end{bmatrix}. \]

### 2.6.2 The Cholesky factorization for HPD matrices

We mentioned before that there is an important class of matrices called Hermitian or symmetric positive definite matrices. For these matrices, we have a special LU decomposition that is easier to perform in practice.

**Theorem 4.** Let \( A \in \mathbb{C}^{n \times n} \) be a Hermitian positive definite then there exists a lower triangular matrix \( L \in \mathbb{C}^{n \times n} \) with a positive diagonal such that

\[ A = LL^*. \]

This decomposition is called the Cholesky factorization and it is unique.

**Remark 2.** If a matrix \( A \) is symmetric positive definite, then \( A \) can be decomposed as \( A = LL^T \), where \( L \in \mathbb{R}^{n \times n} \).

**Proof.** Because \( A \) is HPD, it has a spectral decomposition \( A = U \Lambda U^* \), where \( U^* U = I \) and \( \Lambda = \text{diag}(\lambda_1, ..., \lambda_n) \) for \( \lambda_i > 0 \). Such a matrix \( A \) can be decomposed to

\[ A = U(\Lambda)^{\frac{1}{2}}(\Lambda)^{\frac{1}{2}}U^* = U(\Lambda)^{\frac{1}{2}}U^*U(\Lambda)^{\frac{1}{2}}U^* = A^{\frac{1}{2}}A^{\frac{1}{2}} = (A^*)^{\frac{1}{2}}A^{\frac{1}{2}} = (A^{\frac{1}{2}})^*A^{\frac{1}{2}}. \]
2. Direct solution of linear systems

It is clear that $A^{\frac{1}{2}}$ is also HPD. We will see later that any HPD matrix $B$ can be decomposed to $B = QR$ where $Q$ is a unitary matrix ($Q^*Q = I$) and $R$ is an upper triangular matrix with a positive diagonal, and they are unique if $B$ is non-singular. Therefore - let $A^{\frac{1}{2}} = QR$ for such $Q$ and $R$.

$$A = (A^{\frac{1}{2}})^*A^{\frac{1}{2}} = (QR)^*QR = R^*Q^*QR = R^*R.$$  

Now define $L = R^*$ and the Cholesky factorization is achieved.

The uniqueness of the Cholesky factorization is proved similarly to the proof of the uniqueness of pivoting-less LU factorization.

The Cholesky factorization is very important - in some sense $L$ is the square root of the matrix $A$ - and it exists only because $A$ is positive. The advantage of the Cholesky factorization over $LU$ decomposition is straightforward - it requires half of the memory and it does not require pivoting.

The algorithm for the Cholesky factorization is built on the equation for $a_{ij}$

$$a_{ij} = \sum_{k=1}^{n} l_{ik} \bar{l}_{kj} = \sum_{k=1}^{\min(i,j)} l_{ik} \bar{l}_{kj}. \quad (14)$$

In the first iteration we set $l_{11} = \sqrt{a_{11}}$, and then set the whole first column of $L$ according to Eq. (14) to be $l_{1j} = \frac{a_{1j}}{l_{11}}$. In the $i$-th row, we assume that all the entries that we’ve determined so far (columns 1, ..., $i-1$ of $L$) are given and set according to Eq. (14) for $a_{i,i}$

$$l_{ii} = \left( a_{ii} - \sum_{k=1}^{i-1} l_{ik} \bar{l}_{ki} \right)^{\frac{1}{2}}.$$  

Following that we set the $i$-th column of $L$ (set $l_{ij}$ for $j = i+1, ..., n$) as

$$l_{ij} = \frac{1}{l_{ii}} \left( a_{ij} - \sum_{k=1}^{i-1} l_{ik} \bar{l}_{kj} \right),$$

which is again because of Eq. (14)

Given an LU decomposition $A = LU$, one can extract the diagonal of $U$, and write
Algorithm: Cholesky
\[ A \in \mathbb{C}^{n \times n} \]
Initialize: \( L = 0^{n \times n} \).
for \( i = 1,\ldots,n \) do
\[
l_{ii} = \left( a_{ii} - \sum_{k=1}^{i-1} l_{ik} \bar{l}_{ki} \right)^{\frac{1}{2}}.
\]
for \( j = i + 1,\ldots,n \) do
\[
l_{ij} = \frac{1}{l_{ii}} \left( a_{ij} - \sum_{k=1}^{i-1} l_{ik} \bar{l}_{kj} \right),
\]
end
end

Algorithm 3: Cholesky decomposition

\[ U = D\hat{U}, \] where \( \hat{U} \) is an upper triangular system with ones on the diagonal and \( D \) is a diagonal matrix such that \( D_{ii} = U_{ii} \). We obtain

\[ A = LD\hat{U}. \]

This factorization is also called LDU factorization. To get a Cholesky factorization, we will denote \( \bar{L} = LD^\frac{1}{2} \), and \( \bar{U} = D^\frac{1}{2}\hat{U} \). We obtain \( A = \bar{L}\bar{U} \), where both matrices have identical entries on the diagonal. If \( A \) is Hermitian positive definite, then \( A = A^* \), and hence \( \bar{L}\bar{U} = \bar{U}^*\bar{L}^* \). From here, it is easy to show that \( \bar{L} = \bar{U}^* \), and we get the Cholesky factorization.

Computing Determinant. Similarly to the LU factorization, the Cholesky factorization can also be used to calculate the determinant of an Hermitian \( A \). If \( A = LL^* \), \( \det(L) = \Pi_i l_{ii} \), and \( \det(A) = \det(L)^2 \). Also, if the Cholesky factorization has only positive values on its diagonal - the matrix is HPD. If the factorization fails - the matrix is not positive definite.
2.6.3 Thomas Algorithm - solution of tridiagonal linear systems.

Another special case of LU factorization is when a matrix $A$ is tridiagonal.

$$Ax = L U x = f \Rightarrow U x = g$$

Let $A$ be a tridiagonal matrix of order $n$ given by:

$$A = \begin{pmatrix}
    a_1 & c_1 & 0 & \cdots & 0 \\
    b_2 & a_2 & c_2 & \cdots & 0 \\
    & \ddots & \ddots & \ddots & \vdots \\
    0 & \cdots & b_{n-1} & a_n & 0 \\
    0 & \cdots & 0 & b_n & a_n
\end{pmatrix}$$

Then $A$ can be factored as:

$$A = LU = \begin{pmatrix}
    1 & \beta_2 & \beta_3 & \cdots & \beta_{n-1} \\
    \alpha_2 & 1 & \beta_3 & \cdots & \beta_{n-1} \\
    \alpha_3 & \alpha_2 & 1 & \cdots & \beta_{n-1} \\
    \vdots & \vdots & \vdots & \ddots & \vdots \\
    \alpha_n & \alpha_{n-1} & \alpha_{n-2} & \cdots & 1
\end{pmatrix}$$

$$x_n = \frac{g_n}{\alpha_n}, x_i = \frac{g_i - c_i x_{i+1}}{\alpha_i}, i = n - 1, \ldots, 1$$

For the given matrix $A$:

$$\begin{pmatrix}
    2 & 1 & 0 & 0 & 0 \\
    1 & 4 & 1 & 0 & 0 \\
    0 & 1 & 4 & 1 & 0 \\
    0 & 0 & 1 & 4 & 1 \\
    0 & 0 & 0 & 1 & 2
\end{pmatrix} \begin{pmatrix}
    k_0 \\
    k_1 \\
    k_2 \\
    k_3 \\
    k_4
\end{pmatrix} = \begin{pmatrix}
    2.7 \\
    3.82 \\
    0 \\
    -3.82 \\
    -2.7
\end{pmatrix}$$

The solution is:

$$x_n = \frac{g_n}{\alpha_n}, x_i = \frac{g_i - c_i x_{i+1}}{\alpha_i}, i = n - 1, \ldots, 1$$

For the given matrix $A$:

$$\begin{pmatrix}
    2 & 1 & 0 & 0 & 0 \\
    1 & 4 & 1 & 0 & 0 \\
    0 & 1 & 4 & 1 & 0 \\
    0 & 0 & 1 & 4 & 1 \\
    0 & 0 & 0 & 1 & 2
\end{pmatrix} \begin{pmatrix}
    k_0 \\
    k_1 \\
    k_2 \\
    k_3 \\
    k_4
\end{pmatrix} = \begin{pmatrix}
    2.7 \\
    3.82 \\
    0 \\
    -3.82 \\
    -2.7
\end{pmatrix}$$

$$\beta_2 = 0.5, \alpha_2 = 4 - 0.5 = 3.5$$

$$\beta_3 = 1/3.5, \alpha_3 = 4 - 1/3.5 \approx 3.714$$

$$\beta_4 = 1/3.714, \alpha_4 = 4 - 1/3.714 \approx 3.731$$

$$\beta_5 = 1/3.731, \alpha_5 = 2 - 1/3.731 \approx 1.732$$

$$g_1 = 2.70$$

$$g_2 = 3.82 - 0.5 \cdot 2.7 \approx 2.47$$

$$g_3 = 0 - 2.47 \cdot 3.5 \approx -0.706$$

$$g_4 = -3.82 + 0.706/3.714 \approx -3.629$$

$$g_5 = -2.7 + 3.63/3.731 \approx -1.727$$
2. Direct solution of linear systems

The Thomas algorithm is basically equivalent to a pivoting-less LU decomposition, and therefore the algorithm is not guaranteed to succeed or be stable. If the algorithm fails it means that the matrix may be singular. It does succeed when the matrix is Hermitian positive definite (then it can be LU or Cholesky decomposed without pivoting), or if the matrix is strictly diagonally dominant.  

\[
\begin{align*}
k_4 &= -1.727/1.732 \approx -0.997 \\
k_3 &= -3.629 + 0.997 \div 3.731 \approx -0.705 \\
k_2 &= -0.705 + 0.705 \div 3.714 = 0 \\
k_1 &= 2.47 - 0 \div 3.5 \approx 0.705 \\
k_0 &= 2.7 - 0.705 \div 2 \approx 0.997
\end{align*}
\]

A matrix \( A \) is strictly diagonally dominant if for every row \( i \) \(|a_{ii}| > \sum_j |a_{ij}|\) or for every column \( j \) \(|a_{jj}| > \sum_i |a_{ij}|\).