Numerical Computing, Spring 2019
Michael Overton

The Cholesky Factorization

As explained in A&G, p. 115, one way to compute the Cholesky factorization of a symmetric, positive definite matrix is to compute $A = LU$ without pivoting, set $D = \text{diag}(U) = \text{diag}(u_{11}, \ldots, u_{nn})$, and $	ilde{U} = D^{-1}U$, which is unit upper triangular. So then $A = LD\tilde{U}$ and, since $A = A^T$, also

$$A = \tilde{U}^TDL^T = \tilde{U}^T(DL^T),$$

which is an LU factorization of $A$ with $\tilde{U}^T$ unit lower triangular and $DL^T$ upper triangular. Since the LU factorization is unique, this means $\tilde{U} = L^T$, so

$A = LD_{1/2}L_{1/2}^T$ — this is the Cholesky factorization of $A$.

However, this is not an efficient way to compute $G$. Instead, we do the following. This is explained on p. 115–116 for the case $n = 2$, but the general case is not really explained, so here are some more details.

Suppose that $A$ is an $n \times n$ symmetric, positive definite matrix, and that $a_{11} = 1$. Then we can write

$$A = \begin{pmatrix} 1 & w^T \\ w & B \end{pmatrix}$$

where $w$ is a column vector of length $n - 1$ and $B$ is an $(n-1) \times (n-1)$ symmetric, positive definite matrix. Let us carry out the first step of Gauss elimination (LU factorization), subtracting $w_j$ times the first row from the $j$th row, for $j = 2, \ldots, n$. This gives:

$$\begin{pmatrix} 1 & 0 \\ -w & I \end{pmatrix} \begin{pmatrix} 1 & w^T \\ w & B \end{pmatrix} = \begin{pmatrix} 1 & w^T \\ 0 & B - ww^T \end{pmatrix}.$$

This eliminates the first column below the diagonal. Gauss elimination would then go on to use the second row to eliminate the second column
below the diagonal, but to exploit symmetry, Cholesky goes on to apply the same transformation from the right, subtracting \( w_j \) times the first column from the \( j \)th column, for \( j = 2,\ldots,n \). This gives:

\[
\begin{pmatrix}
1 & 0 \\
-w & I
\end{pmatrix}
\begin{pmatrix}
1 & w^T \\
w & B
\end{pmatrix}
\begin{pmatrix}
1 & -w^T \\
0 & I
\end{pmatrix}
= \begin{pmatrix}
1 & w^T \\
0 & B - ww^T
\end{pmatrix}
\begin{pmatrix}
1 & -w^T \\
0 & I
\end{pmatrix}
= \begin{pmatrix}
1 & 0 \\
0 & B - ww^T
\end{pmatrix}.
\]

or equivalently

\[
A = \begin{pmatrix}
1 & w^T \\
w & B
\end{pmatrix}
= \begin{pmatrix}
1 & 0 \\
0 & I
\end{pmatrix}
\begin{pmatrix}
1 & w^T \\
0 & B - ww^T
\end{pmatrix}
\begin{pmatrix}
1 & 0 \\
0 & I
\end{pmatrix}.
\]

This is the end of the first step of the Cholesky factorization. But now we can continue inductively — if we know that \( B - ww^T \) is positive definite. But we know this is true because of equation (1): just multiply the left and right-hand sides by \( v^T \) from the left and \( v \) from the right, since the original matrix \( A \) is positive definite.

But what if \( a_{11} \neq 1 \)? In that case, write

\[
A = \begin{pmatrix}
\alpha & w^T \\
w & B
\end{pmatrix}
\]

where we know \( \alpha > 0 \) since \( A \) is positive definite. Then it turns out we can write

\[
A = \begin{pmatrix}
\frac{\sqrt{\alpha}}{\alpha w} & 0 \\
\frac{\sqrt{\alpha}}{\alpha^2 w} & I
\end{pmatrix}
\begin{pmatrix}
1 & 0 \\
0 & B - \frac{1}{\alpha}ww^T
\end{pmatrix}
\begin{pmatrix}
\sqrt{\alpha} & \frac{1}{\sqrt{\alpha}}w^T \\
1 & I
\end{pmatrix}.
\]

Let’s write this as

\[
A = G_1A_1G_1^T.
\]

Now we can apply the same idea to the \((n-1) \times (n-1)\) bottom right submatrix of \( A_1 \), giving \( A_1 = G_2A_2G_2^T \), then to the \((n-2) \times (n-2)\) bottom right submatrix of \( A_2 \), giving \( A_2 = G_3A_3G_3^T \), and so on. In the last step we just have to take the square root of the final bottom right entry. This finally gives

\[
A = G_1G_2 \ldots G_nG_n^T \ldots G_2^TG_1^T
\]

or, multiplying out these triangular matrices,

\[
A = GG^T,
\]

where \( G \) is lower triangular: the Cholesky factorization. (In MATLAB’s \texttt{chol}, the Cholesky factorization is denoted \( A = R^TR \), where \( R = G^T \) is upper triangular.)
There is actually a more straightforward way to derive the Cholesky algorithm as follows. We will illustrate it in the case \( n = 3 \). Let us write down the equation \( GG^T = A \), as

\[
\begin{pmatrix}
g_{11} & 0 & 0 
g_{21} & g_{22} & 0 
g_{31} & g_{32} & g_{33}
\end{pmatrix}
\begin{pmatrix}
g_{11} & g_{21} & g_{31} 
g_{21} & g_{22} & g_{32} 
g_{31} & g_{32} & g_{33}
\end{pmatrix} =
\begin{pmatrix}
a_{11} & a_{21} & a_{31} 
a_{21} & a_{22} & a_{32} 
a_{31} & a_{32} & a_{33}
\end{pmatrix}
\]

(recalling that \( A \) is symmetric). Now let’s determine the \( g_{ij} \) one at a time, by looking at the product of the \( i,j \) entry of this equation, for all \( i \geq j \), as follows:

- \((i = 1, j = 1)\): \( g_{11}^2 = a_{11} \) so \( g_{11} = \sqrt{a_{11}} \) if \( a_{11} > 0 \); otherwise \( A \) is not positive definite so stop

- \((i = 2, j = 1)\): \( g_{21}g_{11} = a_{21} \) so \( g_{21} = a_{21}/g_{11} \)

- \((i = 3, j = 1)\): \( g_{31}g_{11} = a_{31} \) so \( g_{31} = a_{31}/g_{11} \)

- \((i = 2, j = 2)\): \( g_{21}^2 + g_{22}^2 = a_{22} \) so \( g_{22} = \sqrt{a_{22} - g_{21}^2} \) if \( a_{22} - g_{21}^2 > 0 \); otherwise \( A \) is not positive definite so stop

- \((i = 3, j = 2)\): \( g_{31}g_{21} + g_{32}g_{22} = a_{32} \) so \( g_{32} = (a_{32} - g_{31}g_{21})/g_{22} \)

- \((i = 3, j = 3)\): \( g_{31}^2 + g_{32}^2 + g_{33}^2 = a_{33} \) so \( g_{33} = \sqrt{a_{33} - g_{31}^2 - g_{32}^2} \) if \( a_{33} - g_{31}^2 - g_{32}^2 > 0 \); otherwise, \( A \) is not positive definite.

We know that the diagonal of \( G \) has to be positive if \( A \) is positive definite. Note that we only divide by the \( g_{kk} \) which we know are positive. See my code chol3.m for the implementation. A good exercise would be to write down the algorithm for the \( n \times n \) case with the appropriate nested loops.

Note that this algorithm agrees with the top of p. 116, A&G where they derive the case \( n = 2 \). However, their algorithm written in the middle of the page is very difficult to understand because (a) it does the operations in a different order from the derivation here and (b) it overwrites the lower triangle of \( A \) with the Cholesky factor \( G \).

A very important property of the Cholesky factorization algorithm is that it is stable without any need for pivoting as long as \( A \) is positive definite. Note, however, that it fails if \( A \) is only positive semidefinite, e.g. \( A = 0 \). It is possible to use a Cholesky algorithm with pivoting to handle the positive semidefinite case, but this is not commonly used. Also, like the LU decomposition, Cholesky can be implemented with pivoting to reduce fill-in for sparse matrices.