Cholesky Decomposition

Cholesky decomposition is a special version of LU decomposition tailored to handle symmetric matrices more efficiently.

For a symmetric matrix A, by definition, $a_{ij} = a_{ji}$. LU decomposition is not efficient enough for symmetric matrices. The computational load can be halved using Cholesky decomposition.

Using the fact that A is symmetric, write

$$A = LL'$$

where L is a lower triangular matrix. That is,

$$L = \begin{bmatrix} l_{11} & 0 & 0 & \dots & 0 \\ l_{21} & l_{22} & 0\dots & 0 & \\ \vdots & & & & \\ l_{n1} & l_{n2} & n_{n3} & \dots & l_{nn} \end{bmatrix} \tag{1}$$

Note that the diagonal elements of L are not 1s as in the case of LU decomposition.

With Cholesky decomposition, the elements of L are evaluated as follows:

$$l_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2} \quad k = 1, 2, \dots, n \tag{2}$$

$$l_{ki} = \frac{1}{l_{ii}} \left(a_{ki} - \sum_{j}^{i-1} l_{ij} l_{kj} \right) \quad i = 1, 2, \dots, k-1 \tag{3}$$

where the first subscript is the row index and the second one is the column index.

Cholesky decomposition is evaluated column by column (starting from the first column) and, in each row, the elements are evaluated from top to bottom. That is, in each column the diagonal element is evaluated first using (2) (the elements above the diagonal are zero) and then the other elements in the same row are evaluated next using (3). This is carried out for each column starting from the first one.

Note that if the value within the square root in (2) is negative, Cholesky decomposition will fail. However, this will not happen for positive semidefinite matrices, which are encountered commonly in many engineering systems (e.g., circuits, covariance matrix). Thus, Cholesky decomposition is a good way to test for the positive semidefiniteness of symmetric matrices.

Pseudocode for Cholesky decomposition is given below:

```
for k = 1 : n
    % evaluate off-diagonal terms
    for i = 1 : k-1
        s = 0
        for j = 1 : i -1
            s = s + aij * akj
        end
        aki = (aki - s) / aii
    end

    % evaluate diagonal term
    s = 0
    for j = 1 : k-1
        s = s + (akj)^2
    end
    akk = (akk - s)^(0.5)

end
```

Steps for solving $Ax = b$, where A is symmetric, using Cholesky decomposition is given below.

1. Factorize A in to $A = LL'$.
2. Solve for x
   (a) Forward substitution:
       Solve for d using $Ld = b$
   (b) Back substitution:
       Solve for x using $L'x = d$

Iterative Methods for Solving a Set of Linear Equations

Gauss-Seidel iteration

In some cases, the straightforward LU or Cholesky decomposition may become inefficient. Then, the alternative is to use some iterative technique that starts with an initial guess and iterates through the solution until convergence. One such method is Gauss-Seidel iteration.

We'll illustrate the method using a set of three equations, which are given by:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \tag{4}$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2 \tag{5}$$

$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3 \tag{6}$$

From (4), we can solve for x_1 in terms of x_2 and x_3 as

$$x_1 = \frac{b_1 - a_{12}x_2 - a_{13}x_3}{a_{11}} \tag{7}$$

Similarly, x_2 and x_3 can be solved in terms of the others as

$$x_2 = \frac{b_2 - a_{21}x_1 - a_{23}x_3}{a_{22}} \tag{8}$$

$$x_3 = \frac{b_3 - a_{31}x_1 - a_{32}x_2}{a_{33}} \tag{9}$$

from (5) and (6), respectively.

Then, x_1, x_2 and x_3 can be solved for using the following steps:

1. Start with an initial guess for x_2 and x_3 (for example, $x_2 = x_3 = 0$).
2. Update x_1 using (7) based on the current values of x_2 and x_3.
3. Update x_2 using (8) based on the current value of x_3 and the value of x_1 found from Step 2.
4. Update x_3 using (9) based on the value of x_1 found from Step 2 and the value of x_2 found from Step 3.
5. Check for convergence and if not converged, go to Step 2.

Convergence can be tested using the relative error between successive iterations. That is,

$$\epsilon_i = \left| \frac{x_i^{\text{new}} - x_i^{\text{old}}}{x_i^{\text{old}}} \right| \times 100\% < \epsilon_t \quad i = 1, 2, \ldots, n \tag{10}$$

where ϵ_t is a suitable threshold such as, for example, 1%. Gauss-Seidel iteration can be terminated if each x_i satisfies the convergence criterion.

As with any iterative technique, the question is whether Gauss-Seidel will always converge. It can be shown that Gauss-Seidel will converge if

$$|a_{ii}| < \sum_{j=1, j\neq i} n|a_{ij}| \tag{11}$$

That is, the magnitude of the diagonal element in each row must be greater than the sum of the magnitude of the other terms in the same row. The system must be diagonally dominant.

Note that the above condition is sufficient, but not necessary. That is, while diagonally dominant systems will always converge, the converse is not always true.

Convergence depends on the order of evaluating x_i. Then, if the original system is not diagonally dominant, rows have to be rearranged.

Example: Using Gauss-Seidel iteration, solve

$$\begin{aligned} 11x_1 + 13x_2 &= 286 \\ 11x_1 - 9x_2 &= 99 \end{aligned}$$

This is not diagonally dominant. Rearrange the above equations as

$$\begin{aligned} 11x_1 - 9x_2 &= 99 \\ 11x_1 + 13x_2 &= 286 \end{aligned}$$

which is diagonally dominant.

See Figure 1 which illustrates an example where the order of execution matters for convergence (the evaluation in sub-figure a converges; but, if the evaluation order is reversed as in sub-figure b, the solution diverges).

Resources:

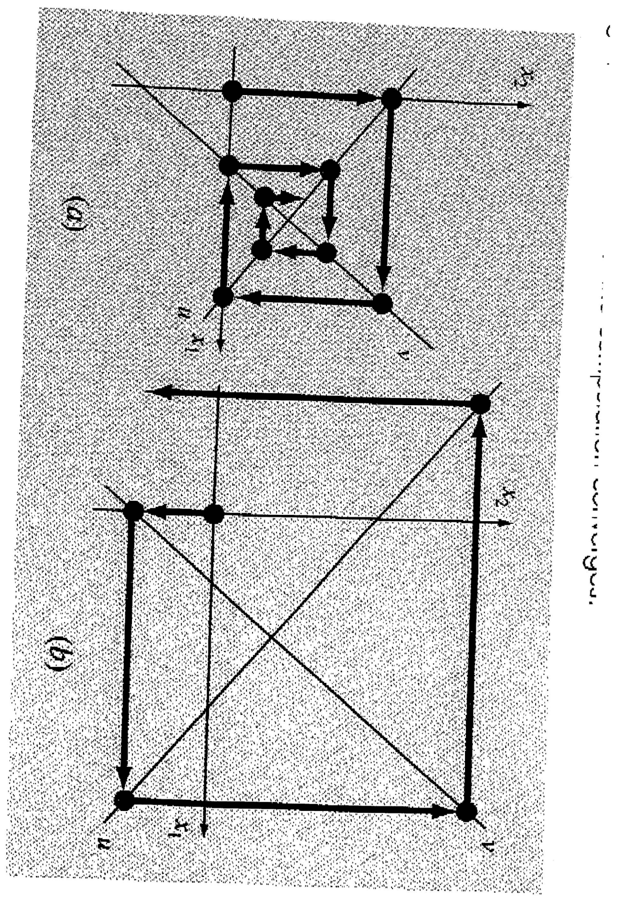

Figure 1: Convergence and divergence of Gauss-Seidel method depending on the order of evaluation

1. Steven Chapra and Raymond Canale, Numerical Methods for Engineers, Fourth Edition, McGraw-Hill, 2002 (see Sections 11.1-11.2).

2. `http://ikpe1101.ikp.kfa-juelich.de/briefbook_data_analysis/node33.html`

3. `http://ada.home.cern.ch/rkb/AN16pp/node156.html`