Tarea 7 - Métodos numéricos

Giovanni Gamaliel López Padilla

Índice

1. Introducción

La factorización de una matriz es usada para reducir la complejidad de la solución a un sistema de ecuaciones lineales. La factorización de una matriz da como resultado la multiplicación de dos matrices, una del tipo triangular inferior y la otra del tipo triangular superior. Esto es:

$$A = LU \tag{1}$$

donde L es la matriz triangular inferior y U matriz triangular superior.

2. Métodos

2.1. Factorización LU - Crout

Si A es una matriz de tamaño n×n, entonces su factorización en matrices L y U deben seguir la forma de las ecuaciones 2 y 3.

$$L = \begin{pmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{pmatrix} \tag{2}$$

$$U = \begin{pmatrix} 1 & u_{12} & \cdots & u_{1n} \\ 0 & 1 & \cdots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{3}$$

Para calcular los elementos de cada matriz se tiene el conjunto de ecuaciones 4.

$$\begin{cases} l_{ij} = a_{ij} - \sum\limits_{k=1}^{j-1} l_{ik}u_{kj} \\ u_{ij} = \frac{a_{ij} - \sum\limits_{k=1}^{i-1} l_{ik}u_{kj}}{l_{ii}} \end{cases} \tag{4}$$

El algoritmo planteado para la solución de este problema se encuentran en las carpetas Problema_1 y Problema_2, en el archivo LU_decomposition.h en la función obtain_LU_crout.
Al tener los elementos determiandos de las matrices L y U, es más sencillo realizar la solución a un sistema de ecuacuiones, ya que, Al tratarse de matrices triangulares sus algoritmos son más faciles de implementar. Suponiendo que se tiene el sistema $Ax = B$, entonces:

$$\begin{aligned} Ax &= B \\ LUx &= B \end{aligned}$$

nombrando

$$Ux = y \tag{5}$$

, entonces:

$$Ly = B \tag{6}$$

donde las ecuaciones 5 y 6 son dos sistemas de ecuaciones lineales del tipo triangular.
El algoritmo planteado para realizar la factorización LU con la version de Crount es la siguiente:

```
    // input: matriz
    // output: L,U
    for (int i = 0; i < n; i++)
    {
        for (int j = i; j < n; j++)
        {
            sum_ij = 0;
            for (int k = 0; k <= i - 1; k++)
            {
                sum_ij += l_ik * u_kj;
            }
            l_ij = matrix_ij - sum_ij;
        }
        for (int j = i + 1; j < n; j++)
        {
            sum_ij = 0;
            for (int k = 0; k < j - 1; k++)
            {
                sum_ij += l_ik * u_kj;
            }
            u_ij = (matrix_ij - sum_ij) / l_ii;
        }
    }
```

Al termino de este de implementaron las funciones solve_triangular_inferior_matrix y solve_triangular_superior_matrix.

2.2. Factorización Cholesky

Sea A, una matriz simetica y definida positiva puede ser factorizada en la forma LU con $U = L^T$, esto es descrito en la ecuacuión 7.

$$A = LL^T \tag{7}$$

El algoritmo emplea el conjunto de ecuaciones .

$$\begin{cases} l_{ii} = \sqrt{a_{ii} - \sum\limits_{k=1}^{i-1} l_{ik}^2} \\ l_{ij} = \frac{a_{ij} - \sum\limits_{k=1}^{j-1} l_{ik} l_{kj}}{l_{jj}} \end{cases} \tag{8}$$

El algoritmo de factorización de Cholesky se encuntra en la carpeta Problema_3 en el archivo Cholesky_decomposition.h en la función obtain_Cholesky. La función es la siguiente:

```
// inputs: matrix, n
// output
for (int i = 0; i < n; i++)
{
    sum = 0;
    for (int j = 0; j < i; j++)
    {
        sum += l_ik * l_ik;
    }
    sqrt_number = matrix_ii - sum;
    vadidate_positive_matrix(sqrt_number);
    validate_l_ii(l_ii);
    for (int j = i; j < n; j++)
    {
        sum = 0;
        for (int k = 0; k < i; k++)
        {
            sum += l_ik * l_jk;
        }
        l_ij = (matrix_ij - sum) / l_ii;
    }
}
fill_L_transpose(l, lt);
```

2.3. Factorización Doolittle

El método de factorización se basa en la eliminación Gaussiana. A partir de una matriz de tamaño n×n se obtiene una matriz triangular superior realizando operaciones entre columnas y renglones de la misma matriz. Entonces, la factorización de Doolittle se concentra en encontrar la matriz triangular inferior tal que se cumple la ecuación 9.

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{21} & 1 & \cdots & 0 \\ l_{31} & l_{32} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdot & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ 0 & 0 & \cdots & u_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdot & u_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ a_{31} & a_{32} & \cdots & a_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdot & a_{nn} \end{pmatrix} \tag{9}$$

Realizando la multiplicación de matrices se obtiene que:

$$\begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ l_{21} & l_{21}u_{12} + u_{22} & \cdots & l_{21}u_{1n} + u_{2n} \\ l_{31} & l_{31}u_{12} + l_{32}u_{22} & \cdots & l_{31}u_{1n} + l_{3n}u_{22} + u_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n1}u_{12} + l_{n2} + u_{22} & \cdots & l_{n1}u_{1n} + l_{n2}u_{2n} + \cdots + u_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ a_{31} & a_{32} & \cdots & a_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdot & a_{nn} \end{pmatrix} \tag{10}$$

Observando la diagonal de la ecuación 10, se puede obtener la ecuación 11.

$$u_{ii} = a_{ii} - \sum_{j=1}^{i} l_{ij} u_{ji} \tag{11}$$

Los elementos en los queda se obtiene un elemento de U sin realizar una multiplicación con un termino de L se obtiene la ecuación 12.

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{ki} \tag{12}$$

De los elementos donde un elemento de L esta multiplicando a un termino de la diagonal de U, se obtiene la ecuación 13.

$$l_{ij} = \frac{a_{ij} - \sum\limits_{k=1}^{j-1} l_{ik} u_{kj}}{u_{jj}} \tag{13}$$

3. Resultados

4. Conclusiones

5. Compilación y ejecución de los programas