Tarea 7 Métodos Numéricos

Carlos Giovanny Encinia González

September 19 2021

Resumen

Al inicio se hablará de dos métodos para solucionar sistemas de ecuaciones lineales, que están en forma matricial. El primer método que se menciona es el método de Gauss con pivoteo total, el segundo método es el de fatorización LU. Cada uno de estos métodos serán puestos a prueba utilizando dos archivos que contienen matrices de distintos tamaños.

Por último se presentará un método de factorización llamado Cholesky modificado, el cual consiste en factorizar una matriz simétrica definida positiva, los factores de este método resultan en una matriz triangular inferior con unos en su diagonal, una matriz diagonal, y una matriz triangular superior con unos en su diagonal. Este método de factorización será puesto a prueba mediante una función generadora de matrices la cuál produce una matriz tridiagonal.

Desarrollo

Gauss Pivote Total

El método de Gauss con pivote total, es un método para la solución de sistemas de ecuaciones lineales, que trata de hacer ceros los elementos debajo de cada elemento de la diagonal de la matriz, una vez realizado esto, debe de quedar una matriz triangular superior de la cuál podemos aplicar algún método para resolver el sistema de ecuaciones modificado. Además de esto cada elemento de la diagonal, al que llamaremos pivote necesita ser aquel elemento que sea mayor dentro de lo que llamaremos submatrices. En seguida se da una explicación más detallada del método.

Supongamos que tenemos la siguiente ecuación matricial:

$$Ax = b \tag{1}$$

Si A es una matriz de mxm entonces los vectores x y b tienen dimensiones de m.

$$x = (x_1, x_2, x_3, \ldots, x_m)^T \tag{2}$$

$$b = (b_1, b_2, b_3, \ldots, b_m)^T \tag{3}$$

Sea A_a la siguiente matriz aumentada.

$$A_a = \left(\begin{array}{ccccc|c} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,m} & b_1 \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,m} & b_2 \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,m} & b_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,m} & b_m \end{array}\right) \tag{4}$$

Ahora tenemos que buscar el elemento con mayor valor absoluto dentro de la matriz A, al encontrarlo tendremos que hacer la permutaciones necesarias para que ahora este en la posición del elemento $a_{1,1}$

Supongamos que el elemento mayor es el $a_{m,3}$ entonces las permutaciones harían que nuestro sistema quede de la siguiente manera.

Primero cambiamos las filas correspondientes

$$F_1 \rightleftarrows F_m$$

$$A_a = \left(\begin{array}{ccccc|c} a_{m,1} & a_{m,2} & a_{m,3} & \cdots & a_{m,m} & b_m \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,m} & b_2 \\ a_{3,1} & a_{3,2} & a_{3,3} & \cdots & a_{3,m} & b_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,m} & b_1 \end{array}\right) \tag{5}$$

Como nuestro elemento a permutar ahora es el $a_{m,3}$ ahora tenemos que permutar las columnas de tal manera que $a_{m,3}$ se encuentre en la posición del primer elemento de la matriz original. Y nos queda que.

$$C_1 \rightleftarrows C_3$$

$$A_a = \left(\begin{array}{ccccc|c} a_{m,3} & a_{m,2} & a_{m,1} & \cdots & a_{m,m} & b_m \\ a_{2,3} & a_{2,2} & a_{2,1} & \cdots & a_{2,m} & b_2 \\ a_{3,3} & a_{3,2} & a_{3,1} & \cdots & a_{3,m} & b_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{1,3} & a_{1,2} & a_{1,1} & \cdots & a_{1,m} & b_1 \end{array}\right) \tag{6}$$

Como permutamos las columnas esto provoca un efecto en el vector x, es necesario que también se permute para no alterar el sistema de ecuaciones.

si

$$C_1 \rightleftarrows C_3$$

entonces

$$Fx_1 \rightleftarrows Fx_3$$

Donde Fx hace referencia a las filas del vector x

En general si

$$C_a \rightleftarrows C_b$$

entonces

$$Fx_a \rightleftarrows Fx_b$$

Entonces para nuestro caso el vector x quedaría como:

$$x = (x_3, x_2, x_1, \ldots, x_m)^T \tag{7}$$

Ahora procedemos a tratar de hacer ceros debajo del pivote que acabamos de encontrar, para esto debemos de encontrar un factor que multiplicado a la fila del pivote haga cero a las siguientes filas, claro que este factor cambiara dependiendo la fila a la que se volverá cero los elementos debajo de nuestro pivote.

Para cada elemento debajo de nuestro pivote el factor debe ser aquel que al sumar con el elemento debajo se haga cero, y esto únicamente se cumple cuando.

$$factor_{i,j} = \frac{a_{i,j}}{pivote_{j,j}} \tag{8}$$

Entonces tenemos que

$$F_2 = F_2 - factor_{2,1} * F(1)$$

$$F_2 = F_2 - \frac{a_{2,1}}{pivote_{1,1}} * F(1)$$

Al hacer esto nos queda que:

$$A_a = \left(\begin{array}{rrrcr|c} a_{m,3} & a_{m,2} & a_{m,1} & \cdots & a_{m,m} & b_m \\ 0 & a'_{2,2} & a'_{2,1} & \cdots & a'_{2,m} & b'_2 \\ a_{3,1} & a_{3,2} & a_{3,1} & \cdots & a_{3,m} & b_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ am,1 & a_{1,2} & a_{1,1} & \cdots & a_{1,m} & b_1 \end{array}\right) \tag{9}$$

Observamos que los valores de las demás columnas de la misma fila también sufren cambios, al igual que los valores correspondientes en fila al vector b. Observamos que los nuevos valores los representamos como $a'_{i,j}$

Aplicando esto hasta la ultima fila m tenemos que

$$A_a = \left(\begin{array}{rrrcr|c} a_{m,3} & a_{m,2} & a_{m,1} & \cdots & a_{m,m} & b_m \\ 0 & a'_{2,2} & a'_{2,1} & \cdots & a'_{2,m} & b'_2 \\ 0 & a'_{3,2} & a'_{3,1} & \cdots & a'_{3,m} & b'_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & a'_{1,2} & a'_{1,1} & \cdots & a'_{1,m} & b'_1 \end{array}\right) \tag{10}$$

Una vez llagados a esto, debemos de volver a buscar el elemento con valor más grande dentro de la matriz, pero esta vez sin contar a la fila y columna en la que se encuentra el pivote anterior es decir, a la matriz:

$$subA_a = \left(\begin{array}{rrcr|c} a'_{2,2} & a'_{2,1} & \cdots & a'_{2,m} & b'_2 \\ a'_{3,2} & a'_{3,1} & \cdots & a'_{3,m} & b'_3 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a'_{1,2} & a'_{1,1} & \cdots & a'_{1,m} & b'_1 \end{array}\right) \tag{11}$$

Aquí aplicamos de nuevo el procedimiento para encontrar el mayor absoluto, después este es permutado y se toman en cuenta los cambios necesarios en el vector x, después se utiliza el factor para hacer 0 los elementos debajo del nuevo pivote. Al finalizar nos debe de quedar una matriz triangular superior, con está calculamos la solución al sistema de ecuaciones asociado a ella.

Al implementar el algoritmo lo que podemos hacer para considerar los cambios en x, es ir guardando los cambios que se realizan al permutar columnas y después utilizarlos para permutar las soluciones.

También, si necesitamos calcular el determinante del sistema de ecuaciones es necesario tener en cuenta las permutaciones que se realizan, si se permutan fila y columna para cambiar al nuevo pivote entonces no se altera el sistema al igual si no se realiza ninguna permutación, pero si se hace una sola permutación entonces el determinante debe se multiplicado por -1, esto sucede cada vez que sucede alguna permutación que cumpla lo anterior.

En seguida se muestra el pseudocódigo del método.

Algorithm 1: Método Gauss por pivoteo total

```
1  solve_gauss_total (matrix, rows, cols, x, y);
   Input  : matriz matrix, el número de filas de la matriz m, arreglo de
            tamaño igual a size donde se almacenará el resultado x,
            arreglo que contiene las variables dependientes y, variable que
            se puede usar para determinar el valor del determinante signo
   Output: x, signo
2  m ← rows;
3  matrix ← matriz_aumentada(matrix, y);
4  cambios_x ← array(0 : m);
5  for i:n do
6  |  i_c ← i;
7  |  j_c ← i;
8  |  (i_c, j_c, maximo) = encuentra_maximo(matrix);
9  |  if maximo=0 then
10 |  |  print("No hay solución para el sistema");
11 |  |  return 0
12 |  end
13 |  if i ≠ i_c and i = j_c or i = i_c and i ≠ j_c then
14 |  |  signo ← signo * (-1)
15 |  end
16 |  if  i ≠ i_c then
17 |  |  intercambias_filas(i, i_c);
18 |  end
19 |  if i ≠ j_c then
20 |  |  intercambia_columnas(j, j_c);
21 |  |  intercambia_x(i, j_c);
22 |  end
23 |  for l=i+1:m do
24 |  |  factor ← matrix_{l,i}/matrix_{i,i};
25 |  |  for j=i:m+1 do
26 |  |  |  matrix_{l,j} ← matrix_{l,j} − factor * matrix_{i,j};
27 |  |  end
28 |  end
29 end
30 x ← solve_superior(matrix, m, x);
31 x ← regresa_posicion_original(x, cambios_x);
32 return x, signo;
```

Solución mediante factorización LU

Este tipo de método propone primero factorizar una matriz A en dos matrices triangulares, una será triangular inferior y la otra triangular superior. Una vez hecho esto se procede a solucionar los sistemas triangulares para obtener una respuesta. En seguida describiremos a detalle el procedimiento.

Supongamos que tenemos la siguiente ecuación matricial:

$$Ax = b \tag{12}$$

Si A es una matriz de mxm entonces los vectores x y b tienen dimensiones de m.

$$x = (x_1, x_2, x_3, \ldots, x_m)^T \tag{13}$$

$$b = (b_1, b_2, b_3, \ldots, b_m)^T \tag{14}$$

Ahora si queremos descomponer la matriz A debe de quedarnos de la siguiente manera.

$$LUx = b \tag{15}$$

De aquí podemos hacer dos ecuaciones matriciales las cuales nos quedaría como:

$$Ux = y \tag{16}$$

$$Ly = b \tag{17}$$

Esto muestra que tendríamos que resolver primero un sistema triangular inferior y de esta manera encontrar los valores del vector y, y después sustituir en la ecuación (16) para poder obtener los valores del vector x, esta última ecuación corresponde a solucionar un sistema de ecuaciones cuyos factores corresponden a una matriz triangular superior.

Ahora la siguiente pregunta es cómo poder encontrar los factores de la matriz A. Para poder encontrar las matrices triangulares utilizaremos el método de factorización LU de Crount el cual consiste a realizar lo que sigue.

Sea

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,m} \end{pmatrix} \tag{18}$$

Entonces podemos factorizar de tal manera que:

$$A = \begin{pmatrix} l_{1,1} & 0 & \cdots & 0 \\ l_{2,1} & l_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{m,1} & l_{m,2} & \cdots & l_{m,m} \end{pmatrix} \begin{pmatrix} 1 & u_{1,2} & \cdots & u_{1,m} \\ 0 & 1 & \cdots & u_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{19}$$

Ahora procederemos a realizar la multiplicación de matrices y el resultado lo igualaremos a su correspondiente valor en la matriz A

Multiplicando primero la primera fila de L por la primera columna de U tenemos que:

$$l_{1,1} = a_{1,1} \tag{20}$$

Multiplicando la segunda fila de L por la primera columna de U tenemos que

$$l_{2,1} = a_{2,1} \tag{21}$$

Si seguimos de esta manera multiplicando cada fila de la matriz L por la primera columna de la matriz U, podemos encontrar la siguiente ecuación.

$$\boxed{l_{i,1} = a_{i,1}} \tag{22}$$

Ahora si multiplicamos la primera fila de L por la segunda columna de U tenemos que

$$l_{1,1}u_{1,2} = a_{1,2} \tag{23}$$

Si multiplicamos la primera fila de L por la tercera columna de U tenemos que

$$l_{1,1}u_{1,3} = a_{1,3} \tag{24}$$

Si seguimos haciendo esto para la primera columna de L y para cada columna de U podemos obtener que:

$$\boxed{u_{1,j} = \frac{a_{1,j}}{l_{1,1}}} \tag{25}$$

Si multiplicamos la segunda fila de L por la segunda columna de U tenemos que:

$$l_{2,1}u_{1,2} + l_{2,2} = a_{2,2} \tag{26}$$

Despejando $l_{2,2}$ tenemos que:

$$l_{2,2} = a_{2,2} - l_{2,1}u_{1,2} \tag{27}$$

Si multiplicamos la tercera fila de L con la segunda columna de U tenemos que:

$$l_{3,1}u_{1,2} + l_{3,2} = a_{3,2} \tag{28}$$

Despejando $l_{3,2}$ tenemos que:

$$l_{3,2} = a_{3,2} - l_{3,1}u_{1,2} \tag{29}$$

Si multiplicamos la segunda fila de L por la tercera columna de U tenemos que:

$$l_{2,1}u_{1,3} + l_{2,2}u_{2,3} = a_{2,3} \tag{30}$$

Despejando $u_{2,3}$ tenemos que:

$$u_{2,3} = \frac{a_{2,3} - l_{2,1}u_{1,3}}{l_{2,2}} \tag{31}$$

Ahora multiplicando la tercera fila de L por la tercera columna de U tenemos que:

$$l_{3,1}u_{1,3} + l_{3,2}u_{2,3} + l_{3,3} = a_{3,3} \tag{32}$$

Despejando $l_{3,3}$ tenemos que:

$$l_{3,3} = a_{3,3} - l_{3,1}u_{1,3} - l_{3,2}u_{2,3} \tag{33}$$

Observando los resultados anteriores podemos generalizar y obtener las siguientes ecuaciones:

$$\boxed{l_{i,j} = a_{i,j} - \sum_{k=1}^{j-1} l_{i,k}u_{k,j} \quad , i \geq j} \tag{34}$$

$$\boxed{u_{i,j} = \frac{a_{i,j} - \sum_{k=1}^{i-1} l_{i,k}u_{k,j}}{l_{i,i}} \quad , i < j} \tag{35}$$

Ahora usando la ecuación (22), (25), (34) y (35) podemos escribir el algoritmo para factorizar y resolver el sistema de ecuaciones.

En seguida se muestra el pseudocódigo. El pseudocódigo esta creado para agregar los valores de las matrices triangulares en el espacio de la matriz original.

Algorithm 2: Algoritmo que resuelve una ecuación matricial mediante factorización LU

```
1  solve_lu (matrix, rows, cols, x, y);
   Input  : matriz matrix, el número de filas de la matriz rows, el
            numero de columnas cols, arreglo de tamaño igual a size
            donde se almacenará el resultado x, arreglo que contiene las
            variables dependientes y
   Output: x
2  i ← 0;
3  for (i : m) do
4      j ← 1;
5      for j : m do
6          if i=0 then
7              matrix_{i,j} = matrix_{i,j}/matrix_{0,0}
8          else
9              if i >= j then
10                 matrix_{i,j} = matrix_{i,j} − sumatoria(matrix, m, i, j);
11                 if i=j and matrix_{i,j} = 0 then
12                     print("Hay un cero en la diagonal, solucion diverge)
                       return 0;
13                 end
14             end
15             if i < j then
16                 matrix_{i,j} =
                     (matrix_{i,j} − sumatoria(matrix, m, i, j))/matrix_{i,i}
17             end
18         end
19     end
20 end
21 y_p ← solveTriangularInferior(matrix, m, y);
22 x ← solveTriangularSuperior(matrix, m, y_p);
23 return x;
```

Algorithm 3: Algoritmo que realiza la sumatoria para resolver factorizacion LU

```
1  sumatoria (matrix, m, i, j);
   Input  : matriz matrix, el número de filas de la matriz m, el numero
            de columnas m, i es in índice que hace referencia a la i-ésima
            fila, j hace referencia a la j-ésima columna
   Output: sum
2  sum ← 0;
3  k ← 0;
4  if i < j then
5  |  condition ← i
6  else
7  |  condition ← j
8  end
9  while k ≤ condition − 1 do
10 |  if k = i then
11 |  |  sum ← sum + matrix_{i,k}
12 |  else
13 |  |  sum ← sum + matrix_{i,k} ∗ matrix_{k,j};
14 |  end
15 |  k ← k + 1;
16 end
17 return sum;
```

Factorización LU método de Doolittle

Lo que hace distinta la descomposición Doolittle de Crount, es únicamente es la matriz triangular que contiene los 1 en la diagonal. Para el caso de Doolittle los 1 los contiene la matriz triangular inferior L, en seguida se muestra el desarrollo del método y las ecuaciones a las que se llega.

Figure 1: Doolittle

Sea $A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}$

Supongamos que A se puede factorizar tal que

$A = LU$ donde LU son una matriz triangular inferior y superior respectivamente.

tal que

$$\begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ l_{21} & 1 & 0 & \cdots & 0 \\ l_{31} & l_{32} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} \\ 0 & 0 & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}$$

Entonces podemos ir desarrollando, mediante la definición de multiplicación de matrices.

Denotando com M_{ij} la multiplicación de la fila i de la matriz L y j la columna j de la matriz U

$M_{11} = u_{11} = a_{11}$ (1)

$M_{21} = l_{21}u_{11} = a_{21}$; pero sabemos el resultado de u_{11}

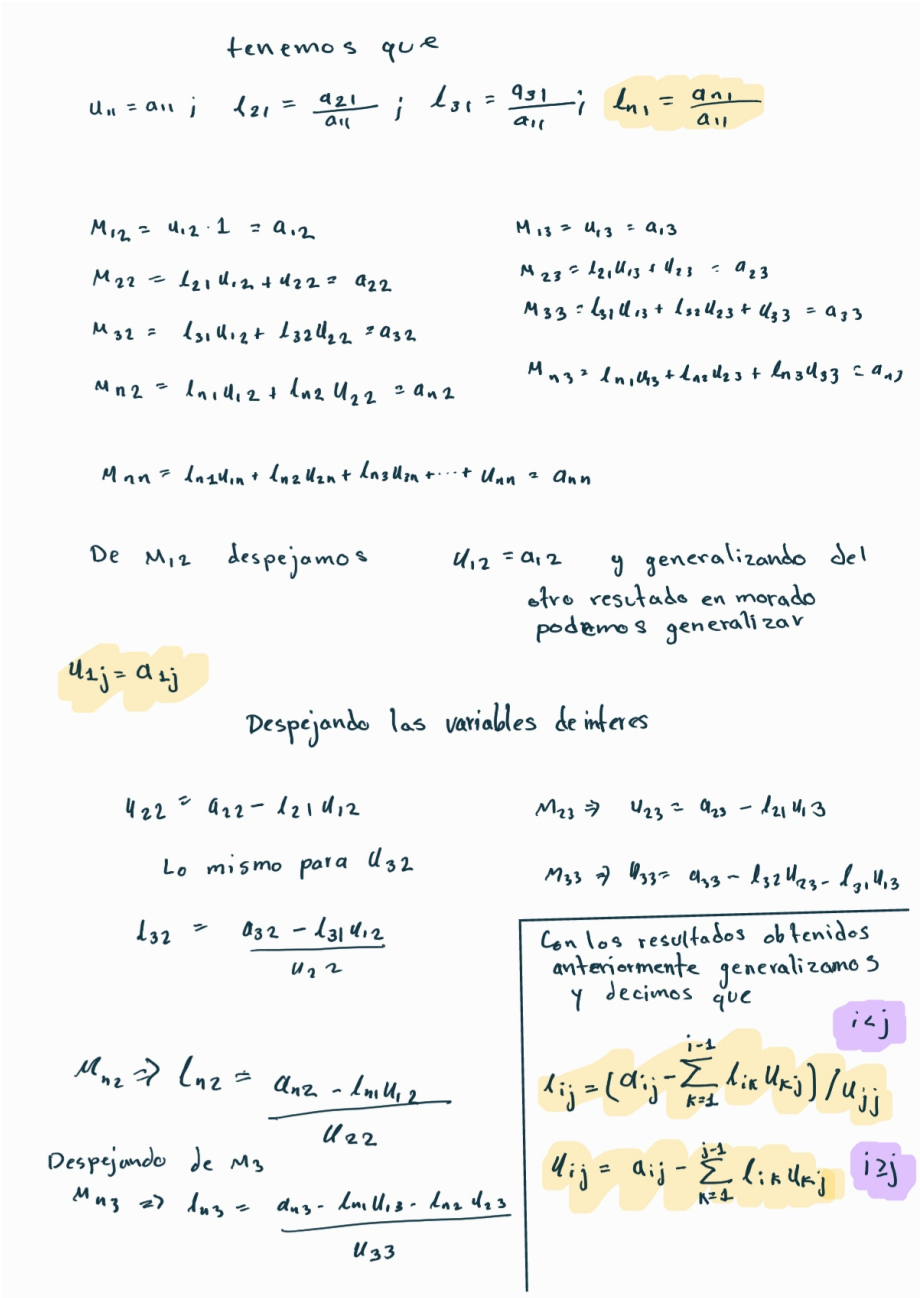
$l_{21}a_{11} = a_{21}$ (2)

$M_{31} = l_{31}u_{11} = l_{31}a_{11} = a_{31}$ (3)

$M_{n1} = l_{n1}u_{11} = l_{n1}a_{21}$ (4)

Ahora despejando los terminos l_{i1} de las 3 ecs.

Figure 2: Doolittle



Factorización Cholesky

Se realizara la factorización de Cholesky modificada la cuál consiste en obtener tres matrices, una triangular inferior, una diagonal, y una triangular superior, las matrices diagonales contienen 1 en sus diagonales.

Para hacer el desarrollo primero definamos una matriz simétrica definida positiva, que la matriz sea positiva quiere decir que sus eigenvalores son positivos, esto debe ser así para que el método converja.

Sea

$$A = \begin{pmatrix} a_{1,1} & a_{2,1} & \cdots & a_{m,1} \\ a_{2,1} & a_{2,2} & \cdots & a_{m,2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,m} \end{pmatrix} \tag{36}$$

Una matriz simétrica definida positiva, entonces se puede factorizar de tal forma que.

$$A = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ l_{2,1} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{m,1} & l_{m,2} & \cdots & 1 \end{pmatrix} \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_m \end{pmatrix} \begin{pmatrix} 1 & l_{2,1} & \cdots & l_{m,1} \\ 0 & 1 & \cdots & l_{m,2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{37}$$

Si multiplicamos la matriz triangular inferior por la matriz diagonal obtenemos lo siguiente:

$$A = \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ l_{2,1}d_1 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{m,1}d_1 & l_{m,2}d_2 & \cdots & d_m \end{pmatrix} \begin{pmatrix} 1 & l_{2,1} & \cdots & l_{m,1} \\ 0 & 1 & \cdots & l_{m,2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{38}$$

Multiplicando la primera fila por la primera columna obtenemos que

$$\boxed{d_1 = a_{1,1}} \tag{39}$$

Ahora multiplicando la primera fila de la primera matriz por la segunda columna de la matriz triangular superior tenemos que:

$$d_1 l_{2,1} = a_{2,1} \tag{40}$$

Despejando $l_{2,1}$ tenemos que

$$l_{2,1} = \frac{a_{2,1}}{d_1} \tag{41}$$

Si multiplicamos ;a primera fila de la primera matriz por alguna de las columnas de la matriz triangular superior observamos que se cumple que:

$$\boxed{l_{i,1} = \frac{a_{i,1}}{d_1}} \tag{42}$$

Si multiplicamos la fila 2 de la primera matriz por la columna 2 de la segunda matriz tenemos que;

$$a_{2,2} = l_{2,1}^2 d_1 + d_2 \tag{43}$$

Despejando d_2 tenemos que:

$$d_2 = a_{2,2} - l_{2,1}^2 d_1 \tag{44}$$

Si multiplicamos la tercera fila de la primera matriz por la columna 2 de la segunda matriz tenemos que:

$$a_{3,2} = l_{2,1} l_{3,1} d_1 + l_{3,2} d_2 \tag{45}$$

Despejando $l_{3,2}$ de la ecuación anterior tenemos que:

$$l_{3,2} = \frac{a_{3,2} - l_{2,1} l_{3,1} d_1}{d_2} \tag{46}$$

Multiplicando la fila 3 de la primera matriz y la columna 3 de la segunda matriz tenemos que:

$$a_{3,3} = l_{3,1}^2 d_1 + l_{3,2}^2 d_2 + d_3 \tag{47}$$

Despejando d_3 tenemos que:

$$d_3 = a_{3,3} - l_{3,1}^2 d_1 - l_{3,2}^2 d_2 \tag{48}$$

Con base a los resultados anteriores podemos generalizar y encontrar un patrón que nos lleva a las siguientes ecuaciones:

$$\boxed{l_{i,j} = \frac{a_{i,j} - \sum_{k=1}^{j-1} l_{i,k} l_{j,k} d_k}{d_j}} \tag{49}$$

$$\boxed{d_i = a_{i,i} - \sum_{k=1}^{i-1} l_{i,k}^2 d_k} \tag{50}$$

Teniendo estas ecuaciones, ahora mostraremos el pseudocódigo.

Algorithm 4: Algoritmo que realiza factorizacion Cholesky modificada

```
1  cholesky (matrix, rows);
   Input  : matriz matrix, el número de filas y columnas rows
   Output: matrix
2  cholesky_mat←matrix;
3  k ← 1;
4  for k:rows do
5  |  cholesky_mat_{k,0} ← cholesky_mat_{k,0}/cholesky_mat_{0,0}
6  end
7  i ← 1;
8  while i < row do
9  |  j ← 1;
10 |  while j ≤ i do
11 |  |  value ←cholesky_value(cholesky_mat, rows, i, j);
12 |  |  cholesky_mat_{i,j} ← value;
13 |  |  j ← j + 1;
14 |  end
15 |  i ← i + 1;
16 end
17 return matrix;
```

Algorithm 5: Algoritmo que calcula los elementos de las matrices triangulares

```
1 cholesky_value (cholesky_mat, m, i, j);
  Input  : matriz cholesky_mat, el número de filas y columnas m,indice
           de la i-esima fila i, indice de la j-esima columnaj
  Output: value
2 cholesky_mat←matrix;
3 if i == j then
4 |  value ← cholesky_mat_{i,j} -sum_cholesky(cholesky_mat, m, i, j)
5 else
6 |  value ←(cholesky_mat_{i,j} - sum_cholesky(cholesky_mat, m, i, j))
  |   /cholesky_mat_{j,j}
7 end
8 return value;
```

Algorithm 6: Algoritmo que calcula sumatoria

```
1 cholesky_value (cholesky_mat, m, i, j);
  Input  : matriz cholesky_mat, el número de filas y columnas m,indice
           de la i-esima fila i, índice de la j-esima columnaj
  Output: suma
2 suma ← 0; k ← 0;
3 if i = j then
4 |  for k:i - 1 do
5 |  |  suma ← suma +cholesky_mat^2_{i,k} *cholesky_mat_{k,k}
6 |  end
7 else
8 |  for k:j - 1 do
9 |  |  suma ←suma+ cholesky_mat_{i,k} * cholesky_mat_{j,k} *
  |  |   cholesky_mat_{k,k}
10|  end
11 end
12 return suma;
```

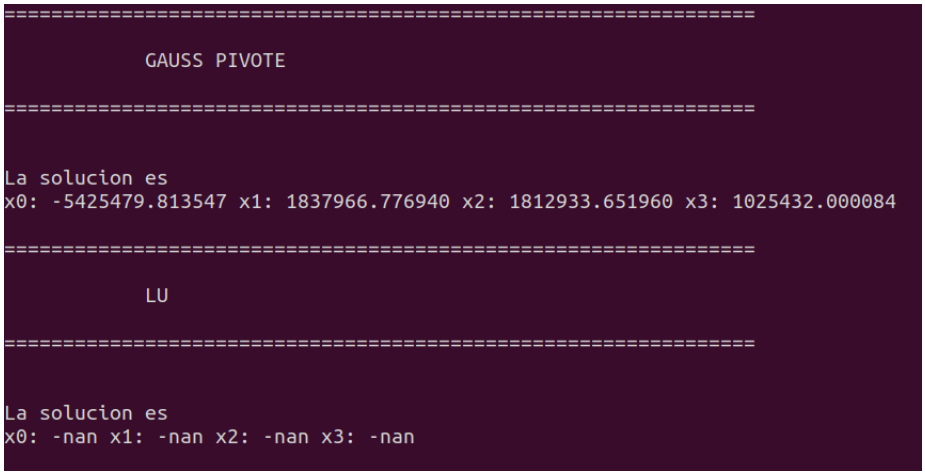
Resultados

Gauss y LU

SMALL

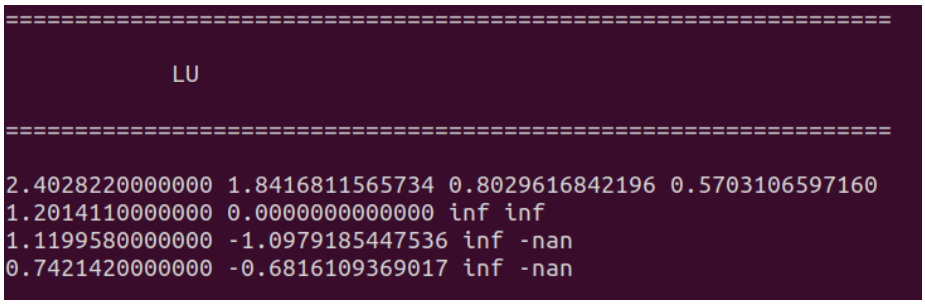
En seguida se muestran los resultados obtenidos al utilizar los métodos de Gauss y LU

Figure 3: Resultado



Como podemos apreciar al calcular y el resultado con el método de factorización LU nos arroja una indeterminación. Ahora mostraremos la matriz en la que se alojan los factores LU.

Figure 4: Error



Como podemos ver se hace un cero en la diagonal. Específicamente $a_{2,2} = l_{2,1}u_{2,1}$, al utilizar la ecuación (34) observamos que el nuevo valor $l_{2,2}$ es cero. Al tener este valor los valores de ui, j quedan indefinidos ya que se necesita de un valor de l distinto de cero ya que este participa como denominador.

LARGE

Como podemos apreciar ambos métodos nos dan como resultados las mismas soluciones, pero hay que tener en cuenta que el método de Gauss pivote es más lento, ya que realiza más operaciones.

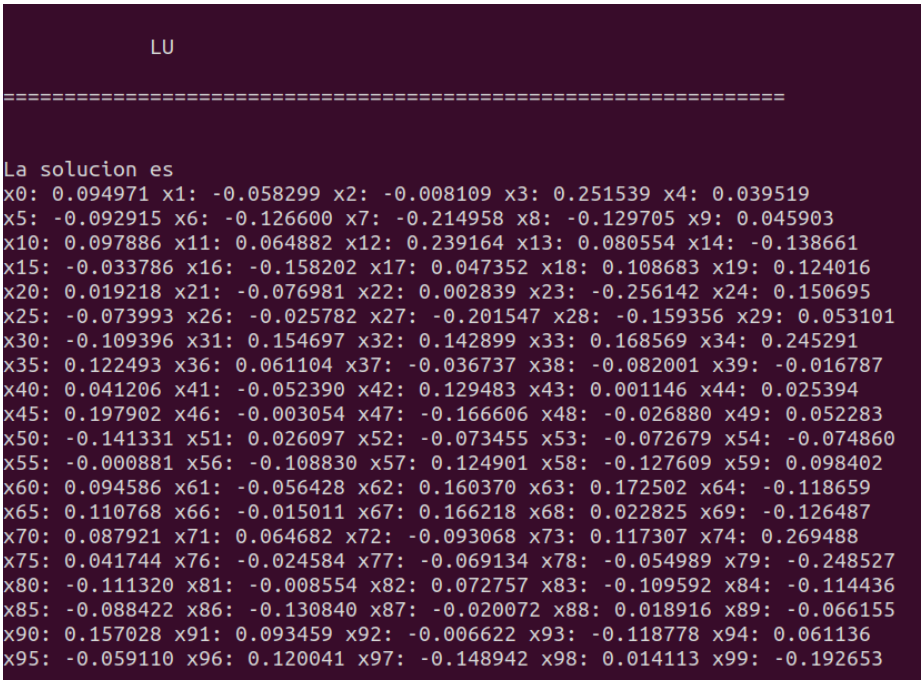
Figure 5: Gauss Pivote

```
            GAUSS PIVOTE

================================================================


La solucion es
x0: 0.094971 x1: -0.058299 x2: -0.008109 x3: 0.251539 x4: 0.039519
x5: -0.092915 x6: -0.126600 x7: -0.214958 x8: -0.129705 x9: 0.045903
x10: 0.097886 x11: 0.064882 x12: 0.239164 x13: 0.080554 x14: -0.138661
x15: -0.033786 x16: -0.158202 x17: 0.047352 x18: 0.108683 x19: 0.124016
x20: 0.019218 x21: -0.076981 x22: 0.002839 x23: -0.256142 x24: 0.150695
x25: -0.073993 x26: -0.025782 x27: -0.201547 x28: -0.159356 x29: 0.053101
x30: -0.109396 x31: 0.154697 x32: 0.142899 x33: 0.168569 x34: 0.245291
x35: 0.122493 x36: 0.061104 x37: -0.036737 x38: -0.082001 x39: -0.016787
x40: 0.041206 x41: -0.052390 x42: 0.129483 x43: 0.001146 x44: 0.025394
x45: 0.197902 x46: -0.003054 x47: -0.166606 x48: -0.026880 x49: 0.052283
x50: -0.141331 x51: 0.026097 x52: -0.073455 x53: -0.072679 x54: -0.074860
x55: -0.000881 x56: -0.108830 x57: 0.124901 x58: -0.127609 x59: 0.098402
x60: 0.094586 x61: -0.056428 x62: 0.160370 x63: 0.172502 x64: -0.118659
x65: 0.110768 x66: -0.015011 x67: 0.166218 x68: 0.022825 x69: -0.126487
x70: 0.087921 x71: 0.064682 x72: -0.093068 x73: 0.117307 x74: 0.269488
x75: 0.041744 x76: -0.024584 x77: -0.069134 x78: -0.054989 x79: -0.248527
x80: -0.111320 x81: -0.008554 x82: 0.072757 x83: -0.109592 x84: -0.114436
x85: -0.088422 x86: -0.130840 x87: -0.020072 x88: 0.018916 x89: -0.066155
x90: 0.157028 x91: 0.093459 x92: -0.006622 x93: -0.118778 x94: 0.061136
x95: -0.059110 x96: 0.120041 x97: -0.148942 x98: 0.014113 x99: -0.192653
```

Figure 6: LU

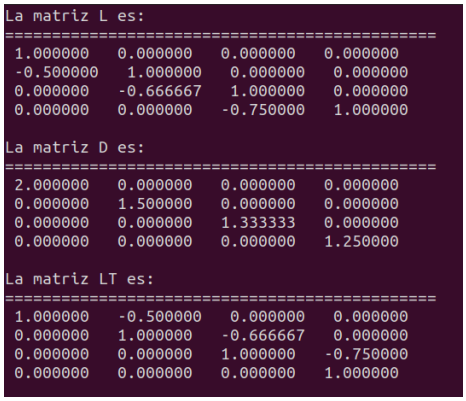


Factorizacion Cholesky

Para la factorización de cholesky se utilizó la siguiente función generadora de matriz, para una matriz de 4x4, 50x50 y 100x100.

Únicamente se mostraran los resultados de la matriz de 4x4, los demás resultados los pueden encontrar dentro de la carpeta "resultados_cholesky" que se encuentran en el archivo comprimido.

Figure 7: Cholesky 4x4



Conclusión

Aunque el método de factorización LU es más rápido que el método de Gauss por pivoteo total con el método de factorización para el caso de la matriz SMALL no se pudo encontrar una solución, aquí se encuentra la importancia de saber implementar diferentes métodos para encontrar soluciones a ecuaciones matriciales. Para la matriz LARGE ambos métodos dieron la misma solución, en lo particular pienso que si tuviésemos que utilizar alguno de los dos métodos para solucionar un sistema, primero usaría el método de factorización, ya que es más rápido, si este llega a que hay un cero en la diagonal, trataría de usar el de Gauss pivote total.

Para el método de factorización de Cholesky, prefiero el método modificado, que es el que se implementó en esta ocasión. Es un método que es relativamente rápido y que se puede aplicar también para poder solucionar sistema de ecuaciones lineales. Algo importante a mencionar es que el código hecho en C toma en cuenta dos casos para la obtención de la matriz que se factorizará, en el primer caso suponemos que la matriz se lee desde algún archivo de texto y en el segundo caso se da como input una función generadora de alguna matriz, qué es el caso que se utiliza para este documento.

No existe algo así como el mejor método para solucionar un sistema de ecuaciones lineales, pero es importante conocer los alcances y limitaciones de casa método ya sea este el más sencillo de implementar, hasta el más complejo.