Métodos Numéricos
Soluciones y Factorización de Sistemas de Ecuaciones
Tarea 2

August 26, 2020

Erika Rivadeneira Pérez
Matemáticas Aplicadas - CIMAT
erika.rivadeneira@cimat.mx

1 Resumen

En el presente reporte se exponen distintos métodos de solución para sistemas de ecuaciones. Se consideran los casos cuando el sistema de ecuaciones es diagonal o triangular (inferior y superior). Además, se menciona el método de eliminación de Gauss con y sin pivoteo total del sistema. Se realiza una comparación de la efectividad del método al implementarlo usando pivoteo y sin usarlo. Por otro lado, se presenta un método de factorización LU denominado el método de Crout. Todos los métodos mencionados se implementan en distintas matrices y se comparan resultados.

2 Metodología

Existen algunos métodos de resolución de sistemas de ecuaciones lineales con el mismo número de ecuaciones que de incógnitas. En concreto, se expone la resolución de sistemas de ecuaciones de la forma

$$\begin{cases} E_1 & : & a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + ... + a_{1n}x_n = b_1, \\ E_2 & : & a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + ... + a_{2n}x_n = b_2, \\ E_3 & : & a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + ... + a_{3n}x_n = b_3, \\ \vdots & & \\ E_n & : & a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + ... + a_{nn}x_n = b_n \end{cases} \tag{1}$$

siendo a_{ij} y b_i, para $1 \leq i, j \leq n$, números reales dados, y siendo $x_j \in R, 1 \leq j \leq n$, las n incógnitas. El sistema (1) es por tanto un sistema lineal nn, es decir, un sistema de n ecuaciones lineales con n incógnitas. A los a_{ij} se les denominan los coeficientes del sistema (1).
La formulación del sistema (1) se puede hacer de forma más compacta. Para ello, se introduce la notación vectorial, denotando en particular a los elementos de R^n como vectores columna, es decir, como matrices $n1$. Denotemos

$$x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{pmatrix}, b = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{pmatrix} \tag{2}$$

y sea A la matriz cuadrada nn de término general a_{ij} , es decir,

$$A = (a_{ij})_{1 \leq i,j \leq n},$$

denominada matriz de coeficientes del sistema (1). [3] De esta forma, el sistema lineal (1) se escribe

$$Ax = b \tag{3}$$

Existen casos en el que los sistemas de ecuaciones tienen formas determinadas y esto conlleva a que tengan un método de solución específicamente para este tipo de sistemas. Estos casos especiales son cuando las matrices de coeficientes están en forma diagonal o triangular (inferior o superior).

2.1 Matriz Diagonal

Es la denominación de la matriz cuadrada de orden n cuyos elementos, excepto al menos uno de la diagonal principal, son cero. Es decir, tiene la forma

$$A = \begin{pmatrix} a_{11} & 0 & 0 & ... & 0 \\ 0 & a_{22} & 0 & ... & 0 \\ 0 & 0 & a_{33} & ... & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & ... & a_{nn} \end{pmatrix}$$

Para obtener la solución en este sistema de ecuaciones hay que tener en cuenta que la matriz A sea no-singular, es decir, $det(A) \neq 0$ o $\prod_{i=1}^{n} d_i \neq 0$, siendo a_{ii} los elementos de la diagonal de A. Si este es el caso, se puede conseguir la solución de este sistema al dividir cada término independiente b_i para cada elemento de la diagonal a_{ii}, es decir,

$$x_i = \frac{b_i}{a_i i}$$

2.2 Matrices Triangulares

Los sistemas triangulares tienen la forma

$$Lx = b \text{ ó } Ux = b$$

en donde, L denota a la matriz triangular inferior y U a la matriz triangular superior.

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

Considere el sistema triangular inferior no singular $3x3$

$$\begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

Dado que la matriz no es singular, sus entradas diagonales $l_{ii}, i = 1, 2, 3$ no desaparecen, por lo que podemos resolver secuencialmente los valores desconocidos $x_i, i = 1, 2, 3$ de la siguiente manera:

$$x_1 = \frac{b_1}{l_1 1}$$
$$x_2 = \frac{(b_2 - l_{21}x_1)}{l_2 2}$$
$$x_3 = \frac{(b_3 - l_{31}x_1 - l_{32}x_2)}{l_3 3}$$

Este algoritmo se puede extender a los sistemas nn y se denomina sustitución directa o hacia adelante. En el caso de un sistema $Lx = b$, donde L es una matriz triangular inferior no singular de orden $n, (n \geq 2)$, el método toma la forma

$$x_1 = \frac{b_1}{l_{11}}, \tag{4}$$

$$x_i = \frac{b_i - \sum_{j=1}^{i-1} l_{ij}x_j}{l_{ii}}, \quad i = 2, ..., n \tag{5}$$

Se pueden sacar conclusiones similares para un sistema lineal $Ux = b$, donde U es una matriz triangular superior no singular de orden $n, (n \geq 2)$ [1]. En este caso, el algoritmo se llama sustitución hacia atrás y en el caso general se puede escribir como

$$x_n = \frac{b_n}{u_{nn}}, \tag{6}$$

$$x_i = \frac{b_i - \sum_{j=i+1}^{n} u_{ij}x_j}{u_{ii}}, \quad i = n - 1, ..., 1 \tag{7}$$

2.3 Método de Eliminación de Gauss

El método de eliminación gaussiana tiene como objetivo reducir el sistema $Ax = b$ a un sistema equivalente, de la forma $Ux = \hat{b}$, donde U es una matriz triangular superior y $\hat{b}$ es un vector del lado derecho actualizado. Este último sistema puede resolverse mediante el método de sustitución hacia atrás. Denotemos el sistema original por $A^{(1)}x = b^{(1)}$. Durante el procedimiento de reducción, básicamente se emplea la propiedad que establece que reemplazar una de las ecuaciones por la diferencia entre esta ecuación y otro multiplicador por una constante no nula esto da como resultado un sistema equivalente (es decir, uno con la misma solución).
Por lo tanto, considere una matriz no singular $A \in R^{nn}$, y suponga que la entrada diagonal a_{11} no desaparece. Introduciendo los *multiplicadores*

$$m_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}, \quad i = 2, 3, ...n$$

donde $a^{(1)}$ denota los elementos de $A^{(1)}$, es posible eliminar la incógnita x_1 de las filas distintas de la primera simplemente restando de la fila i, con $i = 2, ..., n$, la primera fila multiplicada por m_{i1} y haciendo lo mismo en el lado derecho. Si ahora definimos

$$a_{ij}^{(2)} = a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)}, \quad i, j = 2, ..., n,$$
$$b_i^{(2)} = b_i^{(1)} - m_{i1}b_1^{(1)}, \quad i = 2, ..., n,$$

donde $b_i^{(1)}$ denota los componentes de $b^{(1)}$, se obtiene un nuevo sistema de la forma

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & ... & a_{11n}^{(1)} \\ 0 & a_{22}^{(2)} & ... & a_{2n}^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & ... & a_{nn}^{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(2)} \end{bmatrix}$$

asumiendo que $a_{ii}^{(i)} \neq o$ para $i = 1, ..., k-1$. Es claro que para $k = n$ se obtiene un sistema triangular superior $A^{(n)}x = b^{(n)}$

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & ... & ... & a_{11n}^{(1)} \\ 0 & a_{22}^{(2)} & ... & ... & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & & \vdots \\ 0 & & & \ddots & \vdots \\ 0 & & & & a_{nn}^{(n)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ \vdots \\ b_n^{(n)} \end{bmatrix}$$

De acuerdo con las notaciones que se han introducido previamente, se denota por U la matriz triangular superior $A^{(n)}$ [2].Las entradas $a_{kk}^{(k)}$ se llaman pivotes y no deben ser nulas para $k = 1, ..., n-1$..

Para resaltar las fórmulas que transforman el $k-$ésimo sistema en uno $k+1$-ésimo, para $k = 1, ..., n-1$ se asume que $a_{kk}^{(k)} \neq 0$ y se define el multiplicador

$$m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}, \quad i = k+1, ..., n. \tag{8}$$

Así

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}, \quad i, j = k+1, ..., n \tag{9}$$

$$b_{i}^{(k+1)} = b_{i}^{(k)} - m_{ik} b_{k}^{(k)}, \quad i = k+1, ..., n \tag{10}$$

Además, se puede realizar un *pivoteo total* en el proceso de eliminación gaussiana, este picoteo constituye una estrategia para evitar en la medida de lo posible los errores que se propagan en los métodos de eliminacion Gaussina simple y con pivoteo parcial. Este método consiste en escoger el pivote como el valor absoluto del numero mas grande de toda la matriz, siendo en ocasiones necesario un cambio no solo en las filas si no también en las columnas, por lo que se debe tener en cuenta que la solución del sistema cambia de acuerdo a como se varíen las columnas para hacer que el mayor numero de la matriz quede en la posición que se requiere.

2.4 Factorización de Matrices-Método de Crout

En el método de Crout la matriz A es factorizada como $A = LU$ en donde la matriz L es una matriz triangular inferior y U una matriz triangular superior con diagonal unitaria. Viéndolo matricialmete,

$$A = LU \tag{11}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & ... & a_{1n} \\ a_{21} & a_{22} & a_{23} & ... & a_{2n} \\ a_{31} & a_{32} & a_{33} & ... & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n13} & ... & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 & ... & 0 \\ l_{21} & l_{22} & 0 & ... & 0 \\ l_{31} & l_{32} & l_{33} & ... & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & ... & l_{nn} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} & ... & u_{1n} \\ 0 & 1 & u_{23} & ... & u_{2n} \\ 0 & 0 & 1 & ... & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & ... & 1 \end{bmatrix} \tag{12}$$

El método de Crout es un procedimiento del tipo recursivo, esto significa el desarrollo de un conjunto de pasos sucesivos en donde el trabajo a realizar en cada paso resulta similar o del mismo tipo pero basado en resultados obtenidos en pasos anteriores. Estos pasos consisten en la descomposición sucesiva de los menores principales de la matriz de coeficientes A. Para verlo más claro,

resolvamos (12) por menores para conseguir cada elemento de la matriz L y U.

$$
\begin{aligned}
&l_{11} = a_{11} \\
&l_{21} = a_{21} \\
&l_{21}u_{12} = a_{12} \rightarrow u_{12} = \frac{a_{12}}{l_{21}} \\
&l_{21}u_{12} + l_{22} = a_{22} \rightarrow l_{22} = a_{22} - l_{21}u_{12} \\
&l_{31} = a_{32} \\
&l_{31}u_{12} + l_{32} = a_{32} \rightarrow l_{32} = a_{32} - l_{31}u_{12} \\
&l_{11}u_{13} = a_{13} \rightarrow u_{13} = \frac{a_{13}}{l_{11}} \\
&l_{21}u_{13} + l_{22}u_{23} = a_{23} \rightarrow u_{23} = \frac{a_{23} - l_{21}u_{13}}{l_{22}} \\
&l_{31}u_{13} + l_{32}u_{23} + l_{33} = a_{33} \rightarrow l_{33} = a_{33} - l_{31}u_{13} - l_{32}u_{23} \\
&\vdots
\end{aligned}
$$

Por lo tanto, los elementos de las matrices L y U son

$$
\begin{aligned}
l_{ij} &= a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj} \\
u_{ij} &= \frac{a_{ij} - \sum_{k=1}^{j-2} l_{ik}u_{kj}}{l_{ii}} \\
l_{ii} &= a_{ii} - \sum_{k=1}^{i-1} l_{ik}u_{ki}
\end{aligned}
$$

3 Desarrollo

3.1 Implementación de los Métodos de Sustitución

- Sustitución en matrices diagonales: Para resolver este tipo de matrices se requiere como datos de entrada la matriz diagoanal A y el vector independiente b, aquí n es la dimensión de A, ya que es nxn.

Algorithm 1: Pseudocódigo de solución de sistemas de ecuaciones para matrices diagonales

```
Result: x=(x1,x2,...,xn)
x=(0,0,...,0);/* creamos vector de dimension n para guardar soluciones   */
for i=1:n do
|   x[i]=b[i]/A[ii]
end
```

- Matriz Triangular Inferior - Sustitución hacia adelante:
  Para resolver un sistema de ecuaciones en la cual la matriz de A tiene la forma de una matriz triangular inferior se realiza el siguiente procedimiento, con datos de entrada A y b, el vector de términos independientes:

Algorithm 2: Pseudocódigo sustitución hacia adelante

```
Result: x=(x1,x2,...,xn)
x=(0,0,...,0);/* creamos vector de dimension n para guardar soluciones   */
for i = 1:n do
    sumj = 0; /* inicialiamos una variable para guardar sumas          */
     for j = 1:i do
     |  sumj += A[i,j]*x[j]
    end
    x[i]=(b[i]-sumj)/A[i,i]
end
```

- Matriz Triangular Superior - Sustitución hacia atrás:
  Para este caso, los argumentos de entrada son la matriz triangular superior A con dimensión nxn y el vector b.

Algorithm 3: Pseudocódigo sustitución hacia atrás

```
Result: x=(x1,x2,...,xn)
x=(0,0,...,0);/* creamos vector de dimension n para guardar soluciones   */
for i in (n,1,-1) do
    sumj = 0; /* inicialiamos una variable para guardar sumas          */
     for j = i+1:n do
     |  sumj += A[i,j]*x[j]
    end
    x[i] = (b[i]-sumj)/A[i,i]
end
```

3.2 Implementación Eliminación Gaussiana

Para resolver el sistema de ecuaciones 1, tomando en cuenta que E_i son las ecuaciones del sistema y que los datos de entrada son la matriz A y b, se tienen las siguientes instrucciones para resolver un sistema de ecuaciones con eliminación Gaussiana, el cual incluye un pseudocódigo del método con pivoteo.

Algorithm 4: Pseudocódigo eliminación gaussiana

```
Result: x=(x1,x2,...,xn)
for i in (1:n-1) do
    b[i]= b[i]/A[i,i];
    A[i]=A[i]/A[i,i];
    for j = i+1:n do
        b[j]=b[j]-A[j,i]*b[i];
        A[j]=A[j]-A[j,i]*A[i];
    end
    eliminación hacia atras comienza()*
end
```

Algorithm 5: Pseudocódigo eliminación Gaussiana con pivoteo

```
Result: x=(x1,x2,...,xn)
x=(0,0,...,0);/* creamos vector de dimension n para guardar soluciones   */
for i = 1:n-1 do
    sumj = 0; /* inicialiamos una variable para guardar sumas          */
     for j = 1:i do
        Sea p el entero más pequeño con i≤ p ≤ n pi ≠ 0
        if p no se encuentra then
            return "No existe solución única"
        end
        if p!=i then
            (Ep )⟷ (Ei)end
        end
        for j=i+1:n do
            Set m[j,i]=a[j,i]/a[i,i];
            (Ej - A[j,i]*Ei) → (Ej)
        end
    end
    if a[n,n]=0 then
        "No existe solución única"
    end
     Set x[n]=a[n,n+1]/a[n,n] /* Empieza la sustitución hacia atrás    */
     Return (x1,...,xn)
```

3.3 Implementación Factorización LU con el método de Crout

Para la implementación de este método se requieren de argumentos de entrada las matrices A y b. Este algoritmo asegura arreglar la matriz L cuando se tiene uno o varios ceros en la diagonal.

Algorithm 6: Pseudocódigo factorización LU-Método de Crout

```
Result: L,U
L inicializamos matriz nxn de ceros U inicializamos matriz nxn de ceros
 for j=1:n do
    U[j,j]=1;
    for i=j:n do
        Ltemp = A[i,j];
        for k=1:j do
            Ltemp -= L[i,k]*U[k,j]
        end
        L[i,j] = Ltemp
    end
    for i=j+1:n do
        Utemp = A[j,i]
        for k=1:j do
            Utemp -= L[j,k]*U[k,i]
        end
        if int(L[j,j]) == 0 then
            L[j,j] = e-40/* Arreglo L cuando un elemento de la diagonal es
              cero                                                    */
        end
         U[j,i] = Utemp/L[j,j]
    end
end
```

4 Resultados

Para probar cada método descrito en la metodología se usaron distintas matrices las cuales se especifican a continuación:

4.1 Matriz Diagonal

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

OUTPUT:

```
La solución a este problema es:

x0 = 1.0
x1 = 1.0
x2 = 1.0
x3 = 1.0
Error:  0.0
```

4.2 Matriz triangular Inferior - Sustitución hacia adelante

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 \\ 4 & 5 & 6 & 0 \\ 7 & 8 & 9 & 10 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

OUTPUT:

```
La solución a este problema es:

x0 = 1.0
x1 = 0.0
x2 = -0.16666666666666666
x3 = -0.15
Error:  0.0
```

4.3 Matriz triangular Superior - Sustitución hacia atrás

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 0 & 5 & 6 & 7 \\ 0 & 0 & 8 & 9 \\ 0 & 0 & 0 & 10 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

OUTPUT:

```
La solución a este problema es:

x0 = -0.23499999999999988
x1 = -0.07000000000000002
x2 = -0.07500000000000001
x3 = 0.4
Error:  0.0
```

4.4 Eliminación de Gauss

$$A = \begin{bmatrix} 2.402822 & 4.425232 & 1.929374 & 1.370355 \\ 1.201411 & 2.212616 & 0.964687 & 0.685178 \\ 1.119958 & 0.964687 & 2.053172 & 0.566574 \\ 0.742142 & 0.685178 & 0.566574 & 1.696828 \end{bmatrix}, \quad b = \begin{bmatrix} 0.060000 \\ 0.542716 \\ 0.857204 \\ 0.761270 \end{bmatrix}$$

Sin pivoteo:

OUTPUT:

```
La solución a este problema es:

x0 = -0.4267613461825314
x1 = -0.021699160223549157
```

```
x2 = 0.2936989898500972
x3 = 0.448642997404569
Error:  0.7342014983541092
```

Con pivoteo:

OUTPUT:

```
La solución a este problema es

[-0.42676139 -0.001245    0.32350706  0.34062339]
Error: 0.5930381986534372
```

Adicionalmente, se utilizó una matriz con dimensión $100x100$ la cual dio un error igual a 2.4767092624687086 usando eliminación gaussiana sin pivoteo total, en cambio usanda el mismo método con pivoteo se obtuvo un error de 1.4458665778525779

4.5 Factorización LU con el Método de Crout

Para implementar este método se ultilizó nuevamente la matriz de dimensión $100x100$ cuyo resultado solamente se muestra en el código debido a la gran dimensión de la misma. También se probó este método nuevamente con la matriz

$$A = \begin{bmatrix} 2.402822 & 4.425232 & 1.929374 & 1.370355 \\ 1.201411 & 2.212616 & 0.964687 & 0.685178 \\ 1.119958 & 0.964687 & 2.053172 & 0.566574 \\ 0.742142 & 0.685178 & 0.566574 & 1.696828 \end{bmatrix}$$

OUTPUT:

```
   L= [[ 2.40282200e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]
 [ 1.20141100e+00 -3.72817182e+01  0.00000000e+00  0.00000000e+00]
 [ 1.11995800e+00 -1.09791854e+00  1.15388864e+00  0.00000000e+00]
 [ 7.42142000e-01 -6.81610937e-01 -2.93375903e-02  1.27174209e+00]]

U= [[ 1.00000000e+00  1.84168116e+00  8.02961684e-01  5.70310660e-01]
 [ 0.00000000e+00  1.00000000e+00 -2.97792881e-18 -1.34113991e-08]
 [ 0.00000000e+00  0.00000000e+00  1.00000000e+00 -6.25276982e-02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]

 >> np.dot(Crout_small[0],Crout_small[1])
Out[49]: L*U
array([[  2.402822  ,   4.425232  ,   1.929374  ,   1.370355  ],
       [  1.201411  , -35.06910217,   0.964687  ,   0.685178  ],
       [  1.119958  ,   0.964687  ,   2.053172  ,   0.566574  ],
       [  0.742142  ,   0.685178  ,   0.566574  ,   1.696828  ]])
```

5 Discusión

Se puede observar que para los casos de matrices diagonales y triangulares nos da una solución a los sistemas dados con error nulo. Esto ayuda a que el método de eliminación gaussiana tenga menos error ya que utiliza la sustitución hacia atrás, la cual hemos visto que es efectiva. Por otro lado, al usar el método de eliminación en matrices más complejas se puede notar que da soluciones con un error significativo, el cual disminuye al usar el mismo método con pivoteo. Al considerar matrices con mayor dimensión y complejidad este error puede aumentar.
Por otro lado, al implementar el método de Crout vemos que efectivamente se obtienen matrices triangulares L y U aunque al multiplicarlas para comprobar su exactitud vemos que nos retorna una matriz muy parecida a A ya que un elemento de esta matriz no es igual a la original. Este error puede deberse a una implementación incorrecta del método o a la complejidad de la matriz.

6 Conclusiones

No existe dificultad en resolver sistemas de ecuaciones diagonales o triangulares. Además, el método de eliminación de Gauss es muy costosa computacionalmente ya que destruye los perfiles originales de las matrices, también provoca inestabilidad numérica si no se utiliza pivoteo, aunque el pivoteo también es muy costoso ya que consume mucha memoria al iterar sobre las matrices. Finalmente, se concluye que el método de factorización de Crout facilita encontrar la solución a los sistemas de ecuaciones al generar matrices triangulares L y U.

References

[1] A. Quarteroni, P. Quarteroni, F. Riccardo Sacco, R. Sacco, and F. Saleri, Numerical Mathematics, ser. Texts in Applied Mathematics. Springer, 2007. [Online]. Available: https://books.google.com.ec/books?id=Y7grAAAAYAAJMathematician105Final Grade Project

[2] A. Quarteroni, P. Quarteroni, F. Riccardo Sacco, R. Sacco, and F. Saleri,NumericalMathematics, ser. Texts in Applied Mathematics.Springer, 2007. [Online].Available: https://books.google.com.ec/books?id=Y7grAAAAYAAJMathematician105Final Grade Project

[3] R.L. Burden and J.D. Faires, Análisis Numérico, Grupo Editorial Iberoamérica, México 1985.