




# Analysis of Some Iterative Techniques for Systems of Linear Equations and Their Study of the Convergence Through the Number of Conditioning

Análisis de algunas técnicas iterativas para sistemas de ecuaciones lineales y su estudio de la convergencia a través del número de condicionamiento

F. Mesa  ; D. M. Devia-Narváez  ; G. Correa-Vélez   
 DOI: <https://doi.org/10.22517/23447214.24617>  
 Artículo de investigación científica y tecnológica

**Abstract**— At present, numerical analysis provides us with powerful tools to determine the solution of various problems whose mathematical model can be represented by a system of linear equations, these tools correspond to a number of direct and iterative methods, among which are Carl's method. Gustav Jakob Jacobi and the Doolittle and Crout method, which we analyze and compare in this document. To do this we will initially explore the concepts of conditioning the problem to determine how stable is the system from which the model was obtained, until we reach the decomposition of LU arrays proposed in the Doolittle and Crout method. As a result of the analysis and comparison in this document, depending on what is sought when solving a system of equations, either very large or small enough for our computer, we can choose an approximation that will bring a short-term result with an error. Due to the starting point as proposed in the Jacobi method, or it is possible to reach a direct result by implementing fewer iterations as proposed in the Doolittle and Crout method.

**Index Terms**— Coefficients, convergence, direct, iterative, conditioning number, numerical solution.

**Resumen**— En la actualidad el análisis numérico nos brinda poderosas herramientas para determinar la solución de diversos problemas cuyo modelo matemático puede ser representado por un sistema de ecuaciones lineales, estas herramientas corresponden a un sinnúmero de métodos directos e iterativos entre los que se encuentran el método de Carl Gustav Jakob Jacobi y el método de Doolittle y Crout los cuales analizamos y comparamos en este documento. Para ello exploraremos inicialmente los conceptos de condicionamiento del problema para determinar que tan estable es el sistema de donde se obtuvo el modelo, hasta llegar a la descomposición de matrices LU propuestas en el método de Doolittle y Crout. Como resultado del análisis y comparación en este documento dependiendo de lo que se busque al resolver un sistema de ecuaciones ya sea de tamaño muy grande o lo suficiente pequeño para nuestra computadora,

podemos optar por una aproximación que traerá un resultado a corto plazo con un error debido al punto de partida tal y como como se propone en el método del Jacobi o es posible llegar a un resultado directo implementando menor cantidad de iteraciones como se propone en el método de Doolittle y Crout.

**Palabras claves**— Coeficientes, convergencia, directo, iterativo, número de condicionamiento, solución numérica.

## I. INTRODUCTION

At present we are encountering numerous problems that can be solved using a numerical technique, and many of them are presented as a system of equations. There are various numerical techniques to solve this type, however, before solving any problem posed as systems of equations, the conditioning number of the matrix should be found, in order to determine how stable is the system from which the model was obtained [1] [2]. That is why before presenting some common methods and others not so much it is intended to initially find the conditioning number. To understand the concept of Conditioning Number, it is necessary to first consider the types of errors that can occur in measurements [3].

- Error in the measurement estimates.
- Error in the way the computer is stored.
- Error due to previous calculations.

Thus, the numerical analysis will seek to design an algorithm that is insensitive to such errors, an algorithm that produces a response with greater accuracy, and is now called the stable algorithm, so we need a good definition of stability.

This manuscript was sent on November 28, 2018 and accepted on November 23, 2020.

F. Mesa; D. M. Devia-Narvaez; G. Correa- Velez work as professors of Basic Faculty, Mathematics department. They all belong to the research group of GIMAE (femesa@utp.edu.co, dmdevian@utp.edu.co, gecove@utp.edu.co).



Knowing the conditioning number of the problem, we will use an iterative method to find the solution of the system that the model represents, comparing these results with a direct method of matrix decomposition [4].

## II. CONDITIONING OF A PROBLEM

Each problem can be represented as a function  $f(x)$  of a normed space of data and results [5], thus, it is possible to analyze the change in the image  $y$  from the modifications in the variable  $x$ , and how the function is affected as such.

It is said that for a value in the variable  $x$  a problem is well conditioned, as long as said value in  $x$  produces a relatively small modification in the function  $f(x)$ .

On the other hand, a problem will be badly conditioned if for small changes in the variable  $x$  there is a relatively large change in the function  $f(x)$ [6][7].

We choose the norms of  $F^h$  and  $F^m$  and using its corresponding induced norm for  $f'(x)$  we obtain the following approximation (1):

$$\frac{\|f(x + dx) - f(x)\|}{\|dx\|} \approx f'(x) \quad (1)$$

We can observe the ratio between the absolute error and the absolute error of the data, which is known by the name of absolute conditioning number, so that, if said number has a very high value, small modifications in the variable  $x$  alter largely the solution.

From the relationship of the errors we can obtain of (2) and (3):

$$\frac{\|f(x + dx) - f(x)\|}{\|f(x)\|} \approx \frac{\|dx\|}{\|x\|} * \frac{\|f'(x)\| \|x\|}{\|f(x)\|} \quad (2)$$

$$\frac{\|f(x + dx) - f(x)\|}{\frac{\|dx\|}{\|x\|}} \approx \frac{\|f'(x)\| \|x\|}{\|f(x)\|} \quad (3)$$

So the new analysis in the relations takes the name of relative conditioning number or simply number of conditioning of the problem, which is the number which we will analyze, this number measures the sensitivity of the problem and is represented by the letter  $k(x)$ .

From where we can conclude that:

- If  $k(x)$  is a small number (close to 1), it raises a ratio of small relative errors, where the errors of the data produce small relative errors in the solution, and, therefore, the problem is well conditioned.
- If  $k(x)$  is a large number (much larger than 1), it raises a relative error ratio, where relatively small errors in the data produce a large relative error in the solution,

producing an ill-conditioned problem.

It is important to note that the implementation of a differentiable function is necessary, which is why another feature of condition number that is reduced from function  $f$  previously stated is required; this procedure, although of low complexity, requires large number of calculations [8].

However, when we refer to the conditioning number of a problem, we are talking about the relative conditioning number, so we can put it like this (4):

$$k(x) = \frac{\|x\|}{\|f(x)\|} k'(x) \quad (4)$$

The analysis of the conditioning number in problems and vectors is necessary to give a correct compression to said number in the matrix analysis, which is where we will focus, and on which the code is developed in MATLAB [9].

*A. Number of conditioning for the product of matrices and vectors.*

For the development of the conditioning number of a matrix we can consider the new matrix as a function  $f(x)$  given by  $f(x) = b = Ax$ , where  $A$  is the square matrix to be analyzed and  $x$  is a vector  $nx1$ .

Continuing with the previous approach, in (5) and (6) we proceed to find the derivative of the function  $f$ :

$$f(x) = Ax = \left[ \sum_{j=1}^n a_{ij} x_j \right]_{i=1}^m \quad (5)$$

$$f_i(x) = \sum_{j=1}^n a_{ij} x_j \quad (6)$$

Achieving the derivative of the function through of (7) and (8) :

$$\frac{\partial f_i}{\partial x_j} = a_{ij} \quad (7)$$

$$f'(x) = A \quad (8)$$

Replacing these values in the formula previously proposed for the condition number  $k(x)$  given in (9):

$$k = \frac{\|f'(x)\| \|x\|}{\|f(x)\|} = \frac{\|A\| \|x\|}{\|Ax\|} \quad (9)$$

Where the number  $k$ , condition number of the matrix takes a value greater than or equal to 1, and the same analysis is carried out as for a normal problem; so that:

- If  $k$  takes values close to 1, the matrix is well conditioned.



```

25.     a(i,i)=0;
26. end
27. % The matrix is formed without the
    diagonal
28. T=d^-1*(-a)%T will be the matrix a
    for the negative values and divided
    by the diagonal
29.
30. re=max(abs(eig(T))) % calculation
    of the spectral radius that lets me
    know if it converges or diverges
31. if re>1
32.     disp('Greater Spectral Radio than
    1')
33.     disp('the method does not
    converge')
34. return
35. end
36. C=d^-1*b %we should do the same
    with the vector of independent
    terms i=0;
37. err=tol+1;
38. s=[i,x(1),x(2),x(3),err]; %vector
    that allows me to graph the
    table disp('The first column will be
    the iteration and the last column
    will be the error between each
    value obtained')
39. while err>tol & i<iter
40.     xi=T*x+C;
41. err=norm(xi-x); %the difference
    between the value obtained and
    previous
42. % this step is done to know if you
    have reached an acceptable point
43. x=xi;
44. i=i+1;
45. s(i,1)=i;
46. for j=2:n+1
47.     s(i,j)=x(j-1);
48. end
49. s(i,j+1)=err;
50. end
51. fprintf('\nTABLE:\n');

```

`disp(s)%printing of the table`

### B. Doolittle and Crout method

This method consists of decomposing matrix  $A$  into a lower triangular matrix and another upper triangular matrix in such a way that when multiplied, the original matrix is obtained, as shown in (15).

$$A = L * U \quad (15)$$

Where  $A$  is the original matrix,  $L$  is the lower matrix and  $U$  is the upper matrix.

The method of Doolittle and Crout differ in a diagonal with ones (1s), that is, in the lower triangular diagonal it will be filled with ones (1s) and this will be the Doolittle method and if the superior triangular diagonal matrix has ones (1s), this will be the Crout method.

The difference of finding the respective  $l_{ij}$  and  $u_{ij}$  of each method does not very much, they are similar and will be explained later.

The representation of  $A = LU$  by the Doolittle method is  $l_{ii} = 1$

$$A = \begin{pmatrix} 1 & 0 & 0 & l_{21} & 1 & 0 & l_{31} & l_{32} & 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 & \cdots \\ \vdots & l_{i1} & l_{i2} & l_{i3} & \ddots & & & & & & & & & & & & & & \\ \vdots & \cdots & 1 & u_{12} & u_{13} & u_{13} & 0 & u_{22} & u_{23} & 0 & 0 & u_{33} & \cdots & u_{1j} & \cdots & u_{2j} & \cdots & u_{3j} \\ \vdots & 0 & 0 & 0 & \ddots & \cdots & u_{ii} & & & & & & & & & & & & \end{pmatrix}$$

Where  $A$  is

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{21} & a_{22} & a_{23} & \cdots & a_{1j} & \cdots & a_{ij} & \cdots & a_{ij} & \cdots \\ \vdots & a_{i1} & a_{i2} & a_{i3} & \ddots & \cdots & a_{ij} & \end{pmatrix}$$

To determine the coefficients of  $L$  and  $U$  as a function of those of  $A$ , it is enough to multiply the matrices and compare them with the coefficients of  $A$ , taking the first row of  $L$  and multiplying by the first column of  $U$  the result of each coefficient of the first row of  $A$ . If you multiply the entire row of  $L$  by each and every one of the columns of  $U$  remembering that the first position of  $L$  is 1

$$u_{1j} = a_{1j} \quad 1 \leq j \leq n.$$

Proceeding in the same way we multiply all the rows of  $L$  by the first column of  $U$ , we obtain

$$l_{i1}u_{11} = a_{i1} \rightarrow l_{i1} = \frac{a_{i1}}{u_{11}} = \frac{a_{i1}}{a_{11}}, \quad 2 \leq i \leq n$$

This is where the method to calculate the  $l_{i2}$  becomes interesting since it will be necessary to calculate first those of the first row, to operate in the same way with the second row of  $L$  for the column of  $U$  and to add both results and we will obtain. We multiply the second row of  $L$  by the columns of  $U$  with the above already calculated.

$$\begin{aligned} l_{j1}u_{j1} + u_{2j} = a_{2j} &\rightarrow u_{2j} = a_{2j} - l_{j1}u_{j1} \\ &= a_{2j} - \frac{a_{j1}}{a_{11}}a_{1j} \quad 2 \leq j \leq n \end{aligned}$$

These equations will be used to perform meticulous operations with MATLAB or by hand. For this next part, it is observed that initially the first two rows must be calculated and then the missing rows can be calculated as follows:

Multiplying the rows of  $L$  by the second  $U$  columns

$$l_{i1}u_{1i} + l_{i2}u_{2i} = a_{i2} \rightarrow l_{i1} \\ = \frac{1}{u_{2i}}(a_{i2} - l_{i1}u_{1i}) \quad 3 \leq i \leq n$$

And so, we can continue calculating the coefficients of U and L step by step, filling some rows at the same time and other columns in the same way, obtaining the following equation to calculate each coefficient:

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj}, \quad 1 \leq i \leq j \leq n \\ l_{ij} = \frac{1}{u_{jj}}(a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}) \quad 1 \leq j \leq i \leq n - 1$$

For a good handling of the two equations one should start first with row  $i$  of U and then pass for column  $j$  of L

We present below the Doolittle and Crout Method through the code implemented in MATLAB.

```

1. clc
2. clear
3. fprintf('factoring LU
Doolittle\n\n\n');
4. % A=input(' Enter the matrix A =
\n');
5. % b=input('\n Enter the vector b,
correspond to the independent terms
b=\n');
6. A=[6 -1 2;4 -8 1;-3 4 10];
7. b=[21;5;48];
8. [n,m]=size(A); % it is necessary
that the matrix be squared and the
rows in n and the columns in m
9. if n==m %we assure that if the
matrix is square for k=1:n
10. L(k,k)=1; % the lower matrix
has ones on the diagonal
11. sum=0;
12. for p=1:k-1 % the
positions of the diagonal U
seran (A(k,k)-s) where s is the sum
and multiplication of the position
Lij*Uji
13. sum=sum+L(k,p)*u(p,k);
14. end
15. u(k,k)=(A(k,k)-
sum); % diagonal of U
16. % we started to assemble L
by columns
17. for i=k+1:n
18. sum=0;
19. for r=1:k-1
20. sum=suma+L(i,r)*u(r,
21. k);
22. end

```

```

23. L(i,k)=(A(i,k)-
sum)/u(k,k); % the positions of L
are the decomposition or subtraction
by gauss and the division of the
diagonal of U end
24.
25. for j=k+1:n
26. sum=0;
27. for s=1:k-1
28. sum=suma+L(k,s)*u(s,
j);
29. end
30. u(k,j)=(A(k,j)-suma); %
The matrix U is armed by rows for
ease end
31. end
32. % already decomposed to matrix A
in L and U respectively% it is
necessary to know that the
determinant of A that is the same as
LU is not
33. % 0 cero
34. mu=1; % for the determinant of U
35. mL=1;% this is the determinant
of L for i=1:n % we multiply
the elements of the diagonal of
U mu=mu*u(i,i);
36. end
37. product =mL*mu; % calculation of
the determinte
38. % L*b'= b is different from this
b by the decomposition and the
result is% keep in z if
producto~=0
39. for i=1:n % this for
will be to assemble vector b with
the operations that were done to
decompose A
40. sum=0;
41. for p=1:i-1
42. sum=sum+L(i,p)*z(p);
43. End
44. z(i)=(b(i)-sum)/L(i,i); %
obtaining the vector Z which will be
b as if it had been done with the
augmented matrix
45. end
46. % U*z= this is what is going
to be done
47. for i=n:-1:1
48. sum=0;
49. for p=(i+1):n
50. Sum
51. = sum+u(i,p)*x(p);
52. end
53. x(i)=(z(i)-sum)/u(i,i); %
here the results have already
been end
54. else

```

```

55.     fprintf('\n The determinant
        is zero and you may have infinite
        solutions or none \n')
56.     end
57. end
58.     fprintf('\n Matrix L:\n')
59.     disp(L)
60.     fprintf('\n Matrix U:\n')
61.     disp(u)
62.     fprintf('\n the vector Z:\n')
63.     disp(z)
64.     fprintf('\n\n The solution of X1 up
        Xn es:\n');
65. % a continuation of using a for
        statement, to show the user,
66. % the results in a more orderly
        manner.for i=1:n
67.     xi=x(1,i);
68.     fprintf('\nX%g=',i)
69.     disp(xi);

```

### C. Crout method

The decomposition by the Crout method is distinguished by having the diagonal of U, some; its diagonal is composed of ones, but its shape does not very much.

The matrix A is as follows:

- $u_{ii} = 1$

$$A = LU = \begin{pmatrix} l_{11} & 0 & 0 & l_{21} & l_{22} & 0 & l_{31} & l_{32} & l_{33} & \dots & 0 & \dots & 0 & \dots & 0 & \dots & \vdots \\ \vdots & l_{i1} & l_{i2} & l_{i3} & \ddots & & & & & & & & & & & & \\ \vdots & \dots & l_{nn} & (1 & u_{12} & u_{13} & 0 & 1 & u_{23} & 0 & 0 & 1 & \dots & u_{1j} & \dots & u_{2j} & \dots & u_{3j} & \vdots \\ \vdots & 0 & 0 & 0 & \ddots & \vdots & \dots & 1 & & & & & & & & & & & \end{pmatrix}$$

It can be shown that the diagonal of U is unitary and as it was done in the previous method, we are going to determine the coefficients of U and L that generate those of A For the first column of L

$$l_{i1} = a_{i1}, \quad 1 \leq i \leq n,$$

and the first row of U is also of the form:

$$l_{11}u_{1j} = a_{1j}, \quad \rightarrow u_{1j} = \frac{a_{1j}}{l_{11}} = \frac{a_{1j}}{a_{11}}, \quad 2 \leq j \leq n$$

Now if we generalize it as before we can get the following expressions

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj} \quad 1 \leq j \leq i \leq n$$

$$u_{ij} = \frac{1}{u_{ii}} \left( a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj} \right), \quad 2 \leq j \leq i \leq n$$

We present below the Crout Method through the code implemented in MATLAB.

```
1. clc
```

```

2. clear
3. fprintf('          facto
ring LU Crout\n\n\n');
4. A=input('Enter matrix A = \n');
5. b=input('\n Enter the vector b,
corresponding to the independent
terms b=\n');
6.
7. % A=[6 -1 2;4 -8 1;-3 4 10];
8. % b=[21;5;48];
9. [n,m]=size(A); % it is necessary
that the matrix be squared and the
rows in n and the columns in
mC=[A,b]; % the matrix will be
increased
10. % the matrix C, represents the
shape of the augmented matrix [Ab]
11.
12. fprintf(' \n Matrix C, which
corresponds to the augmented matrix
[Ab] is = \n');
13. disp(C)
14.
15. if n==m
16.     for k=1:n
17.         u(k,k)=1; % the lower
matrix has ones on the
diagonal         sum=0;
18.         for p=1:k-1 %las
posiciones de la diagonal U
seran (A(k,k)-s) where s is the sum
and multiplication of the position
Lij*Uji
19.             suma=suma+L(k,p)*u(p,k)
;
20.         end
21.         L(k,k)=(A(k,k)-
suma); %diagonal de L
22.
23.         % here we start to assemble
L by columns
24.         for i=k+1:n
25.             sum=0;
26.             for r=1:k-1
27.                 sum=sum+L(i,r)*u(r,
k);
28.             end
29.             L(i,k)=(A(i,k)-suma); %
the positions of L are the
decomposition or subtraction by
gauss and the division of the
diagonal of U
30.         end
31.
32.         for j=k+1:n
33.             sum=0;
34.             for s=1:k-1
35.                 sum=sum+L(k,s)*u(s,
j);

```

```

36.         end
37.         u(k,j)=(A(k,j)-
    suma)/L(k,k); % The matrix U is
    armed by rows for ease
38.         end
39.     end
40.     % already decomposed to
    matrix A in L and U respectively
41. % it is necessary to know that the
    determinant of A that is the same
    as LU is not
42. % 0 cero
43.     mu=1; % for the determinant of
    U
44.     mL=1; % this is the determinant
    of L
45.     for i=1:n
46.         mL=mL*L(i,i);
47.     end
48. product=mL*mu; % we multiply the
    elements of the diagonal of U
49. % L*b'= b is different from
    this b by the decomposition and the
    result is
50. % will keep in z
51.
52. if product~=0
53. for i=1:n
54. sum=0;
55. for p=1:i-1
56. sum=sum+L(i,p)*z(p);
57. end
58. z(i)=(b(i)-sum)/L(i,i); % obtaining
    the vector Z
59. end
60.
61.
62. for i=n:-1:1
63. suma=0;
64. for p=(i+1):n
65. sum = sum+u(i,p)*x(p);
66. end
67. x(i)=(z(i)-suma)/u(i,i); % here the
    results have already been end
68. else
69. fprintf('\n The determinant is
    equal to zero, therefore the system
    has infinite or no solution \n')
70. end

71. fprintf('\n Matrix L:\n')
72. disp(L)
73. fprintf('\n Matrix U:\n')
74. disp(u)
B   fprintf('\n the vector Z:\n')
75. disp(z)
76. fprintf('\n\n The solution of X1 up
    Xn is:\n');

```

```

77. % then use a for statement, to show
    the user,
78. % the results in a more orderly
    manner
79. for i=1:n
80. xi=x(1,i);
81. fprintf('\nX%g=',i)
82. disp(xi);
83. end

```

#### IV. ANALYSIS AND RESULTS

In this section the calculation of the conditioning number will be made to a very basic model, simply in order to observe how this value is obtained. We will also expose a very simple linear system which can represent a problem of some kind, which will help us to apply the methods set out in the previous section, and thus carry out the corresponding analysis and comparison between them.

##### A. Calculation of the conditioning number

Suppose we have the model matrix is given by:

$$A = [2 \ 3 \ 1 \ 6 \ 4 \ 2 \ 1 \ 1 \ 1]$$

Then :

$$A^{-1} = [-0,3333 \ 0,3333 \ -0,3333 \ 0,6666 \ -0,1666 \\ -0,3333 \ -0,3333 \ -0,1666 \ 1,6667]$$

$$\text{norm}(A) = [2 \ 3 \ 1 \ 6 \ 4 \ 2 \ 1 \ 1 \ 1]$$

- Sum Column 1= 2+6+1= 9
- Sum Column 2= 3+4+1= 8
- Sum Column 3= 1+2+1= 4

So, we can conclude that the norm 1 of the matrix A, such that, norm (A, 1) is equal to 9.

$$\text{norm}(A^{-1}) = [-0,3333 \ 0,3333 \ -0,3333 \ 0,6666 \\ -0,1666 \ -0,3333 \ -0,3333 \\ -0,1666 \ 1,6667]$$

- Sum Column 1= |-0,333|+0,666+|-0,3333|= 1,333
- Sum Column 2= 0,333+0,1666+0,166= 0,666
- Sum Column 3= 0,333 + 0,333+ 1,667= 2,333

Then we can conclude that norm 1 of matrix A is equal to 9; that is, norm (A, 1) = 9. So, the norm 1 of the inverse of the matrix will be: 2,3333. Thus, the condition number of the matrix is

$$\text{cond}(A, 1) = \frac{\|A^{-1}\|_1}{\|A\|_1} = 9 * 2,333 = 20,99996$$

##### B. Jacobi method application

In the next exercise we will apply the Jacobi Method. Let's start from the following system of equations:

$$6x_1 - x_2 + 2x_3 = 21, \\ 4x_1 - 8x_2 + x_3 = 5$$

$$-3x_1 + 4x_2 + 10x_3 = 48$$

Which we can write in matrix form as:

$$[6 \ -1 \ 2 \ 4 \ -8 \ 1 \ -3 \ 4 \ 10][x_1 \ x_2 \ x_3] = [21 \ 5 \ 48]$$

Calculating the conditioning number as in literal A, we obtain the value  $k = 3.1132$ . Let us now find the numerical solution of this problem, understanding that in an iterative method it is necessary to give at the beginning an approximation of what the result is believed to be or if it is the case, the point  $(x_1, x_2, \dots, x_n) = (0, 0, \dots, 0)$ . Now, knowing the starting point, we will proceed to solve one of the unknowns for each equation, in the form:

$$\begin{aligned} x_1 &= \frac{1}{6}(x_2 - 2x_3 + 21) = \frac{1}{6}x_2 - \frac{1}{3}x_3 + \frac{21}{6} \\ x_2 &= \frac{1}{-8}(-4x_1 - x_3 + 5) = \frac{1}{2}x_1 + \frac{1}{8}x_3 - \frac{5}{8} \\ x_3 &= \frac{1}{10}(3x_1 - 4x_2 + 48) = \frac{3}{10}x_1 - \frac{2}{5}x_2 + \frac{48}{10} \end{aligned}$$

It is called  $[\frac{21}{6} \ \frac{5}{8} \ \frac{48}{10}]$  as the fixed vector called C

And,

$$T = [0 \ \frac{1}{6}x_2 \ \frac{2x_3}{6} \ \frac{4x_1}{8} \ 0 \ \frac{x_3}{8} \ \frac{3x_1}{10} \ \frac{4x_2}{10} \ 0]$$

Identifying who is each variable we will represent the equation in its successive form.

$$x_{i+1} = T * x_i + C \quad i = 1, 2, 3 \dots n$$

Following the example in matrix form would be the first approach as follows:

$$[x_1 \ x_2 \ x_3] = [0 \ \frac{1}{6} \ \frac{2}{6} \ \frac{4}{8} \ 0 \ \frac{1}{8} \ \frac{3}{10} \ \frac{4}{10} \ 0][0 \ 0 \ 0] + [\frac{21}{6} \ \frac{5}{8} \ \frac{48}{10}]$$

$$x_1 = 3.5 \quad x_2 = -0.625 \quad x_3 = 4.8$$

The next iteration would look like this:

$$\begin{aligned} [x_1 \ x_2 \ x_3] &= [0 \ \frac{1}{6} \ \frac{2}{6} \ \frac{4}{8} \ 0 \ \frac{1}{8} \ \frac{3}{10} \ \frac{4}{10} \ 0][3.5 \ -0.625 \ 4.8] \\ &\quad + [\frac{21}{6} \ \frac{5}{8} \ \frac{48}{10}] \end{aligned}$$

$$x_1 = 1.7958 \quad x_2 = 1.725 \quad x_3 = 6.1$$

### C. Doolittle and Crout method application

In the following exercise we will apply the Doolittle and Crout method to the same system of equations solved by the Jacobi method in literal B:

Let's start from the following system of equations already written in matrix form:

$$[6 \ -1 \ 2 \ 4 \ -8 \ 1 \ -3 \ 4 \ 10][x_1 \ x_2 \ x_3] = [21 \ 5 \ 48]$$

Where:

$$A = [6 \ -1 \ 2 \ 4 \ -8 \ 1 \ -3 \ 4 \ 10]$$

Now we proceed to do  $A = LU$

$$L = (1 \ 0 \ 0 \ l_{21} \ 1 \ 0 \ l_{31} \ l_{32} \ 1) \quad U = (6 \ -1 \ 2 \ 0 \ u_{22} \ u_{23} \ 0 \ 0 \ u_{33})$$

- $l_{21} = \frac{a_{21}}{a_{11}} = \frac{4}{6} \approx 0.6667$
- $l_{31} = \frac{a_{31}}{a_{11}} = \frac{-3}{6} = -0.5$
- $u_{22} = a_{22} - \sum_{k=1}^1 l_{2k}u_{k2} = a_{22} - (l_{21}u_{12}) = -8 - (\frac{4}{6} * -1) = -\frac{22}{3} \approx -7.3333$
- $u_{23} = a_{23} - \sum_{k=1}^1 l_{2k}u_{k3} = a_{23} - (l_{21}u_{13}) = 1 - (\frac{4}{6} * 2) = -\frac{1}{3} \approx -0.3333$
- $l_{32} = \frac{1}{u_{22}}(a_{32} - \sum_{k=1}^1 l_{3k}u_{k2}) = \frac{1}{u_{22}}(a_{32} - (l_{31}u_{12})) = \frac{1}{-7.3333}(4 - (-0.5 * -1)) = -\frac{21}{44} \approx -0.4773$
- $u_{33} = a_{33} - \sum_{k=1}^2 l_{3k}u_{k3} = a_{33} - (l_{31}u_{13} + l_{32}u_{23}) = 10 - (-\frac{1}{2} * 2 + (-\frac{21}{44} * -\frac{1}{3})) = \frac{477}{44} \approx 10.8409$

Already having the coefficients of the matrices is proceeded to replace in each

$$L = (1 \ 0 \ 0 \ \frac{4}{6} \ 1 \ 0 \ -\frac{1}{2} \ -\frac{21}{44} \ 1) \quad U = (6 \ -1 \ 2 \ 0 \ -\frac{22}{3} \ -\frac{1}{3} \ 0 \ 0 \ \frac{477}{44})$$

With the matrix already decomposed in L and U we can proceed to calculate the value of the unknowns with the following method or formula

$$Lz = b, \quad Ux = z$$

As the systems are staggered, it can be solved by substitution forward or backward according to the case

$$Lz = (1 \ 0 \ 0 \ \frac{4}{6} \ 1 \ 0 \ -\frac{1}{2} \ -\frac{21}{44} \ 1)(z_1 \ z_2 \ z_3) = (21 \ 5 \ 48)$$

Where:

- $z_1 = 21$
- $z_2 = 5 - \frac{4}{6}z_1 = -9$



$$\bullet \quad z_3 = 48 + \frac{21}{44}z_2 + \frac{1}{2}z_1 = \frac{2385}{44} \approx 54.2045$$

Now with the values of  $z$  we can calculate the value of the unknowns in the following way:

$$\begin{aligned} Ux &= \left( 6 \quad -1 \quad 2 \quad 0 \quad -\frac{22}{3} \quad -\frac{1}{3} \quad 0 \quad 0 \quad \frac{477}{44} \right) (x_1 \quad x_2 \quad x_3) \\ &= \left( 21 \quad -9 \quad \frac{2385}{44} \right) \end{aligned}$$

Where:

$$\begin{aligned} \bullet \quad x_3 &= \frac{2385}{44} * \frac{44}{477} = 5 \\ \bullet \quad x_2 &= -\frac{3}{22} \left( -9 + \frac{1}{3}x_3 \right) = 1 \\ \bullet \quad x_1 &= \frac{1}{6} \left( 21 + x_2 - 2z_3 \right) = 2 \end{aligned}$$

The method is direct and computationally has a lower cost than other methods of elimination.

## V. CONCLUSIONS.

We can see that depending on the need to solve a system of equations either very large or small enough for our computer, we can opt for an approximation that will bring a short-term result with an error due to the starting point as proposed in the Jacobi method it is possible to arrive at a direct result by implementing a smaller number of iterations as proposed in the Doolittle and Crout method.

In necessary the use of non-singular square matrices, because the process studied to find the condition number requires the inverse of the matrix.

A non-singular matrix can be close to the set of singular matrices; and be well conditioned. If the elements of A are very small in absolute value.

On the other hand, if the inverse of the matrix has large values in their absolute value, it is unlikely that this matrix is well conditioned.

The condition number of a matrix gives us an idea of whether its columns are linearly independent or not.

The best conditioned matrix is the unitary matrix, since all its singular values are equal to 1, producing a condition number  $k(A) = 1$ .

## REFERENCES

- [1] J. Nicholas, "Accuracy and Stability of Numerical Algorithms", SIAM, Philadelphia, 1996.
- [2] G.W. Stewart, "Matrix Algorithms. Volume 1, Basic Decompositions", SIAM, Philadelphia, 1998.
- [3] R. Burden y J. Faires, "Numerical Analysis. Ninth ed.", Brooks-Cole Cengage learning, 2011.
- [4] O. Axelsson, "Iterative Solution Methods", Cambridge University Press, New York, 1994.
- [5] F. Mesa, P. P. Cárdenas, C. A. Rodríguez, "Comparison of the Conjugate Gradient Methods of Liu-Storey and Dai-Yuan", *Contemporary Engineering Sciences*, vol. 10, no. 35, pp. 1719-1726, 2017. DOI: 10.12988/ces.2017.711189

- [6] W. Briggs, "A Multigrid Tutorial", Society for Industrial and Applied Mathematics, Estados Unidos, 1987.
- [7] A. Quarteroni, R. Sacco y F. Saleri, "Numerical Mathematics. Second Ed." Texts in applied mathematics. Springer, 2007.
- [8] Y. Skiba, "Métodos y esquemas numéricos: Un análisis computacional", Dirección General de Publicaciones y Fomento Editorial. México, 2005.
- [9] J. H. Mathews and K.D. Fink, "Métodos numéricos con Matlab", Pearson, Madrid, España, 2000.
- [10] H. Wolfgang, "Iterative Solution of Large Sparse Systems of Equations", Springer-Verlag, Estados Unidos, 1994.
- [11] J. M. Ledanois, "Métodos numéricos aplicados en ingeniería", McGraw-Hill, Venezuela, 2000.
- [12] S. C. Chapra and R.P Canale, "Métodos numéricos para ingenieros", McGraw-Hill, México, 2015.



**Fernando Mesa**, university professor of the Mathematics Department of the Technological University of Pereira, Colombia and director of the applied mathematics and education research group. Master of Science (Msc) in physical instrumentation (2007). Member of the plasma laboratory of the National University of Colombia, located in Manizales, Colombia. Member of the group of non-linear differential equations "GEDNOL" at the Technological University of Pereira, Colombia. Director of the research group in Applied Mathematics and Education "GIMAE" at the Technological University of Pereira, Colombia.

ORCID: <https://orcid.org/0000-0002-3418-5555>

**Diana Marcela Devia Narváez**. Associated teacher of mathematics and physics at Universidad Tecnológica de Pereira, Colombia. PhD in engineering (2012). Master of Science (Msc) in the faculty of Physics - Ciencia (2010). Member of plasma laboratory from Universidad Nacional de Colombia located in Manizales, Colombia. Member of nonlinear differential equations group "GEDNOL" at Universidad Tecnológica de Pereira, Colombia. Area of expertise: material processing through assisted plasma techniques, structural characterization mechanical of materials, simulation and modeling of material's physical properties. ORCID: <https://orcid.org/0000-0002-0447-4663>



**German Correa Vélez**, university lecturer of the department of Mathematics at the Universidad Tecnológica de Pereira, Colombia. Master of Science (Msc) in mathematics (2008). Member of nonlinear differential equations group "GEDNOL" at Universidad Tecnológica de Pereira, Colombia. Member of the research group in Applied Mathematics and Education "GIMAE" at the Universidad Tecnológica de Pereira.

ORCID: <https://orcid.org/0000-0002-5244-3095>