# Proyecto Final

## Catonga Tecla Daniel Isaí     2BV1

### 1. Codificar un algoritmo de resolución de un sistema de ecuaciones Ax=b.

- **Eliminación gaussiana.**

Primer ejemplo 4x4.

```
disp("1. ");
```

```
1.
```

```
disp("Eliminación Gaussiana");
```

```
Eliminación Gaussiana
```

```
disp("Matriz A");
```

```
Matriz A
```

```
A = [3 6 -2 9;
    -5 4 5 -6;
    -3 8 2 -3;
    -4 10 3 9]
```

```
A = 4×4
     3     6    -2     9
    -5     4     5    -6
    -3     8     2    -3
    -4    10     3     9
```

```
disp("vector b");
```

```
vector b
```

```
b = [6;
    5;
    3;
    9]
```

```
b = 4×1
     6
     5
     3
     9
```

```
disp("La matriz escalonada es");
```

```
La matriz escalonada es
```

```
disp("[A|b]");
```

```
[A|b]
```

```
[A,b] = e_gaussiana(A,b);
[A,b]
```

```
ans = 4×5
    3.0000    6.0000   -2.0000    9.0000    6.0000
         0   14.0000    1.6667    9.0000   15.0000
         0         0   -1.6667   -3.0000   -6.0000
         0         0         0   12.6857    4.2286
```

```
disp("Hacemos sustitución hacía atrás:")
```

```
Hacemos sustitución hacía atrás:
```

```
x = backups(A,b)
```

```
x = 4×1
    2.0000
    0.5000
    3.0000
    0.3333
```

Otro ejemplo de solución por eliminación Gaussiana 3x3.

```
disp("Otro ejemplo de Eliminación Gaussiana");
```

```
Otro ejemplo de Eliminación Gaussiana
```

```
disp("Matriz A");
```

```
Matriz A
```

```
A = [1 2 7;
     1 2 5;
     3 4 8]
```

```
A = 3×3
     1     2     7
     1     2     5
     3     4     8
```

```
disp("Vector b");
```

```
Vector b
```

```
b = [4;
     3;
    -1]
```

```
b = 3×1
     4
     3
    -1
```

```
[A,b] = e_gaussiana(A,b);
disp("La matriz escalonada es");
```

```
La matriz escalonada es
```

```
disp("[A|b]");
```

```
[A|b]
```

```
[A,b]
```

```
ans = 3×4
     1     2     7     4
     0    -2   -13   -13
     0     0    -2    -1
```

```
disp("Hacemos sustitución hacía atrás:")
```

```
Hacemos sustitución hacía atrás:
```

```
x = backups(A,b)
```

```
x = 3×1
   -6.0000
    3.2500
    0.5000
```

- **Eliminación por Gauss-Jordan.**

Ejemplo 5x5:

```
disp("Gauss-Jordan");
```

```
Gauss-Jordan
```

```
disp("Matriz A");
```

```
Matriz A
```

```
A = [1 -2 2 -3;
     3 4 -1 1;
     2 -3 2 -1;
     1 1 -3 -2]
```

```
A = 4×4
     1    -2     2    -3
     3     4    -1     1
     2    -3     2    -1
     1     1    -3    -2
```

```
disp("Vector b");
```

```
Vector b
```

```
b = [15;-6;17;-7]
```

```
b = 4×1
    15
    -6
    17
    -7
```

```
[A,x] = gauss_jordan(A,b);
```

```
disp("La solución para el sistema");
```

```
La solución para el sistema
```

```
disp("Ax=b es con X igual a");
```

```
Ax=b es con X igual a
```

```
x
```

```
x = 4×1
    2.0000
   -2.0000
    3.0000
   -1.0000
```

```
disp("[A|b]");
```

```
[A|b]
```

```
[A,x]
```

```
ans = 4×5
    1.0000         0         0   -0.0000    2.0000
         0    1.0000         0         0   -2.0000
         0         0    1.0000         0    3.0000
         0         0         0    1.0000   -1.0000
```

Otro ejemplo 4x4:

```
disp("Eliminación Gaussiana");
```

```
Eliminación Gaussiana
```

```
disp("Matriz A");
```

```
Matriz A
```

```
A = [3 6 -2 9;
    -5 4 5 -6;
    -3 8 2 -3;
    -4 10 3 9]
```

```
A = 4×4
     3     6    -2     9
    -5     4     5    -6
    -3     8     2    -3
    -4    10     3     9
```

```
disp("vector b");
```

```
vector b
```

```
b = [6;
    5;
    3;
    9]
```

```
b = 4×1
     6
     5
     3
     9
```

```
[A,x] = gauss_jordan(A,b);
disp("La solución para el sistema");
```

```
La solución para el sistema
```

```
disp("Ax=b es con X igual a");
```

```
Ax=b es con X igual a
```

```
x
```

```
x = 4×1
    2.0000
    0.5000
    3.0000
    0.3333
```

```
disp("[A|b]");
```

```
[A|b]
```

```
[A,x]
```

```
ans = 4×5
    1.0000         0         0         0    2.0000
         0    1.0000         0         0    0.5000
         0         0    1.0000         0    3.0000
         0         0         0    1.0000    0.3333
```

- Matriz Inversa de orden n.

Ejemplo:

```
disp("Inversa de una matriz");
```

```
Inversa de una matriz
```

```
disp("Matriz A");
```

```
Matriz A
```

```
disp("Matriz A");
```

```
Matriz A
```

```
A = [26 -1 13 -1 12;
     10 -32 -2 2 -10;
     15 -1 31 -5 5;
     -15 -11 -15 30 -12;
     1 -12 0 -14 -25]
```

```
A = 5×5
    26    -1    13    -1    12
    10   -32    -2     2   -10
    15    -1    31    -5     5
   -15   -11   -15    30   -12
     1   -12     0   -14   -25
```

```
disp("Vector b");
```

```
Vector b
```

```
b = [-1530;
    902;
   -1452;
   588;
   1610]
```

```
b = 5×1
       -1530
         902
       -1452
         588
        1610
```

```
disp("La inversa de A es")
```

```
La inversa de A es
```

```
A_in = inversa_m(A)
```

```
A_in = 5×5
    0.0624   -0.0151   -0.0215    0.0117    0.0261
    0.0297   -0.0457   -0.0087    0.0139    0.0241
   -0.0219   -0.0010    0.0437    0.0049   -0.0037
    0.0216   -0.0133    0.0076    0.0361   -0.0002
   -0.0238    0.0288   -0.0009   -0.0264   -0.0505
```

```
disp("Entonces x = A'*b");
```

```
Entonces x = A'*b
```

```
disp("Entonces la solución es:")
```

```
Entonces la solución es:
```

```
x = A_in*b
```

```
x = 5×1
  -29.0000
  -27.0000
  -34.0000
  -35.0000
  -33.0000
```

**2. Resolver usando el programa de eliminación Gaussiana el siguiente problema. Considere el siguiente sistema de ecuaciones:**

$$\begin{bmatrix} 0.217 & 0.732 & 0.414 \\ 0.508 & 0.809 & 0.376 \\ 0.795 & 0.886 & 0.338 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 0.741 \\ 0.613 \\ 0.485 \end{bmatrix}$$

**a) Usar las funciones para resolver el sistema**

Resolviendo por Eliminación Gaussiana.

```
disp("2. ");
```

```
2.
```

```
disp("a)")
```

```
a)
```

```
disp("Matriz A");
```

```
Matriz A
```

```
A=[0.217 0.732 0.414;
   0.508 0.809 0.376;
   0.795 0.886 0.338]
```

```
A = 3×3
    0.2170    0.7320    0.4140
    0.5080    0.8090    0.3760
    0.7950    0.8860    0.3380
```

```
disp("Vector b");
```

```
Vector b
```

```
b = [0.741;
     0.613;
     0.485]
```

```
b = 3×1
    0.7410
    0.6130
    0.4850
```

```
[A,b] = e_gaussiana(A,b);
disp("La matriz escalonada es");
```

```
La matriz escalonada es
```

```
disp("[A|b]");
```

```
[A|b]
```

```
[A,b]
```

```
ans = 3×4
    0.2170    0.7320    0.4140    0.7410
         0   -0.9046   -0.5932   -1.1217
         0         0   -0.0012   -0.0031
```

```
disp("Hacemos sustitución hacía atrás:")
```

```
Hacemos sustitución hacía atrás:
```

```
x = backups(A,b)
```

```
x = 3×1
    0.0000
   -0.4160
    2.5254
```

Resolviendo por Gauss-Jordan.

```
A=[0.217 0.732 0.414;
   0.508 0.809 0.376;
   0.795 0.886 0.338];
b = [0.741;
     0.613;
     0.485];
[A,x] = gauss_jordan(A,b);
disp("La solución para el sistema");
```

```
La solución para el sistema
```

```
disp("Ax=b es con el vector X igual a");
```

```
Ax=b es con el vector X igual a
```

```
x
```

```
x = 3×1
    0.0000
   -0.4160
    2.5254
```

```
disp("La matriz se vería se la siguiente forma")
```

```
La matriz se vería se la siguiente forma
```

```
[A,x]
```

```
ans = 3×4
    1.0000         0         0    0.0000
         0    1.0000         0   -0.4160
         0         0    1.0000    2.5254
```

Resolviendo por Inversa

```
A=[0.217 0.732 0.414;
   0.508 0.809 0.376;
   0.795 0.886 0.338];
b = [0.741;
     0.613;
     0.485];
disp("La inversa es")
```

```
La inversa es
```

```
A_in = inversa_m(A)
```

```
A_in = 3×3
10^3 ×
   -0.2500    0.5000   -0.2500
    0.5328   -1.0712    0.5391
   -0.8086    1.6320   -0.8221
```

```
disp("Entonces x = A'*b");
```

```
Entonces x = A'*b
```

```
disp("Entonces la solución es:")
```

```
Entonces la solución es:
```

```
x = A_in*b
```

```
x = 3×1
    0.0000
   -0.4160
    2.5254
```

**b) Aumente las entradas de b por 0.005, 0,010, 0.015, 0.02, 0.025, 0.03, 0.035, . . . , 1 y resolver nuevamente el sistema.**

```
%Numero de iteraciones para crear una matriz en columnas
col = numel(0.005:0.005:1);
%Numero de filas
rows = size(A,1);
%creamos la matriz
matrix_soluciones = zeros(rows,col);
for i = 1:col
    b = b + 0.005;
    [~,x] = gauss_jordan(A,b);
    matrix_soluciones(:,i) = x;
end

disp("B) La matriz de soluciones es:");
```

```
B) La matriz de soluciones es:
```

```
matrix_soluciones
```

```
matrix_soluciones = 3×200
   -0.0000   -0.0000   -0.0000   -0.0000    0.0000   -0.0000   -0.0000   -0.0000 ···
   -0.4128   -0.4097   -0.4065   -0.4033   -0.4001   -0.3969   -0.3937   -0.3906
    2.5319    2.5383    2.5448    2.5512    2.5577    2.5641    2.5706    2.5770
```

## 3. El propósito de esta parte del proyecto es programar una función que

**calcula el determinante de una matriz A. Sin el desarrollo por cofactores.**

**a) Modifique la función lutx de tal forma que regrese 4 objetos.**

```
disp("3. ");
```

```
3.
```

```
disp("A) Aplicamos Lutx a A");
```

```
A) Aplicamos Lutx a A
```

```
disp("Matriz A");
```

```
Matriz A
```

```
A = [4 2 1 5;
    8 3 1 -4;
    2 4 3 3;
    1 2 4 1]
```

```
A = 4×4
     4     2     1     5
     8     3     1    -4
     2     4     3     3
     1     2     4     1
```

```
disp("Al aplicarle lutx, tenemos:")
```

```
Al aplicarle lutx, tenemos:
```

```
[L,U,p,sig] = lutx(A);
L
```

```
L = 4×4
    1.0000         0         0         0
    0.2500    1.0000         0         0
    0.1250    0.5000    1.0000         0
    0.5000    0.1538    0.0308    1.0000
```

```
U
```

```
U = 4×4
    8.0000    3.0000    1.0000   -4.0000
         0    3.2500    2.7500    4.0000
         0         0    2.5000   -0.5000
         0         0         0    6.4000
```

```
p
```

```
p = 4×1
     2
     3
     4
     1
```

```
sig
```

```
sig = -1
```

```
```

**b) Escriba una función mydet(A) que use su función modificada lutx para calcular el determinante de A. El producto de la diagonal se la matriz puede ser calculada con prod(diag(U)).**

```
disp("B)El determinante de la matriz A");
```

```
B)El determinante de la matriz A
```

```
disp("Sea A");
```

```
Sea A
```

```
A
```

```
A = 4×4
     4     2     1     5
     8     3     1    -4
     2     4     3     3
     1     2     4     1
```

```
disp("El determinante es");
```

```
El determinante es
```

```
det_ = mydet(A)
```

```
det_ = -416
```

## 4. Generar la matriz asociada al siguiente sistema y la matriz extendida [A|b] con ayuda de un ciclo

**a) Usa la función lutx y las funciones de sustitución hacia atrás backsubs para resolver el sistema con n = 100**

```
disp("4. ");
```

```
4.
```

```
%definimos n
n = 100;
%hacemos nuestro vector n de tipo columna o traspuesto por '
b = (1:n)';

%hacemos una matriz de ceros para la velocidad del programa

matriz = zeros(n,n);

%hacemos nuestra matriz donde van a ver dos casos donde en la primera fila
%no se va poder hacer el de i-1 cuando i = 1 - primera fila. Y con la
%última fila tampoco se va a poder hacer cuando i+i cuando i = 100 ya que
%estamos superando los limites
for i = 1:n
    if i > 1
```

```
        matriz(i,i-1) = -1;
    end
    matriz(i,i) = 2;
    if i < n
        matriz(i,i+1) = -1;
    end
end
disp("A) La matriz es de la forma: ")
```

```
A) La matriz es de la forma:
```

```
matriz
```

```
matriz = 100×100
     2    -1     0     0     0     0     0     0     0     0     0     0     0 ···
    -1     2    -1     0     0     0     0     0     0     0     0     0     0
     0    -1     2    -1     0     0     0     0     0     0     0     0     0
     0     0    -1     2    -1     0     0     0     0     0     0     0     0
     0     0     0    -1     2    -1     0     0     0     0     0     0     0
     0     0     0     0    -1     2    -1     0     0     0     0     0     0
     0     0     0     0     0    -1     2    -1     0     0     0     0     0
     0     0     0     0     0     0    -1     2    -1     0     0     0     0
     0     0     0     0     0     0     0    -1     2    -1     0     0     0
     0     0     0     0     0     0     0     0    -1     2    -1     0     0
     ⋮
```

```
disp("Y el vector b:")
```

```
Y el vector b:
```

```
b
```

```
b = 100×1
     1
     2
     3
     4
     5
     6
     7
     8
     9
    10
     ⋮
```

```
[L,U,p,~] = lutx(matriz);
disp("Despues de aplicar lutx tenemos que U es:")
```

```
Despues de aplicar lutx tenemos que U es:
```

```
U
```

```
U = 100×100
    2.0000   -1.0000         0         0         0         0         0         0 ···
         0    1.5000   -1.0000         0         0         0         0         0
         0         0    1.3333   -1.0000         0         0         0         0
         0         0         0    1.2500   -1.0000         0         0         0
         0         0         0         0    1.2000   -1.0000         0         0
```

```
         0         0         0         0         0    1.1667   -1.0000         0
         0         0         0         0         0         0    1.1429   -1.0000
         0         0         0         0         0         0         0    1.1250
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
     ⋮
```

```
disp("aplicamos sustitución hacía atrás y tenemos que el vector solución es:");
```

```
aplicamos sustitución hacía atrás y tenemos que el vector solución es:
```

```
x = backups(U,b);
x
```

```
x = 100×1
10^3 ×
    0.0958
    0.1906
    0.2839
    0.3755
    0.4654
    0.5535
    0.6398
    0.7242
    0.8067
    0.8873
     ⋮
```

**b) Obtener la factorización P LU. (Usa el vector p para obtener la matriz P).**

```
disp("B) Factorización PA = LU -> A = P'*L*U");
```

```
B) Factorización PA = LU -> A = P'*L*U
```

```
disp("Matriz A");
```

```
Matriz A
```

```
matriz
```

```
matriz = 100×100
     2    -1     0     0     0     0     0     0     0     0     0     0     0 ···
    -1     2    -1     0     0     0     0     0     0     0     0     0     0
     0    -1     2    -1     0     0     0     0     0     0     0     0     0
     0     0    -1     2    -1     0     0     0     0     0     0     0     0
     0     0     0    -1     2    -1     0     0     0     0     0     0     0
     0     0     0     0    -1     2    -1     0     0     0     0     0     0
     0     0     0     0     0    -1     2    -1     0     0     0     0     0
     0     0     0     0     0     0    -1     2    -1     0     0     0     0
     0     0     0     0     0     0     0    -1     2    -1     0     0     0
     0     0     0     0     0     0     0     0    -1     2    -1     0     0
     ⋮
```

```
disp("Aplicando lutx a nuestra Matriz A tenemos:")
```

Aplicando lutx a nuestra Matriz A tenemos:

```
L
```

```
L = 100×100
    1.0000         0         0         0         0         0         0         0 ···
   -0.5000    1.0000         0         0         0         0         0         0
         0   -0.6667    1.0000         0         0         0         0         0
         0         0   -0.7500    1.0000         0         0         0         0
         0         0         0   -0.8000    1.0000         0         0         0
         0         0         0         0   -0.8333    1.0000         0         0
         0         0         0         0         0   -0.8571    1.0000         0
         0         0         0         0         0         0   -0.8750    1.0000
         0         0         0         0         0         0         0   -0.8889
         0         0         0         0         0         0         0         0
     ⋮
```

```
U
```

```
U = 100×100
    2.0000   -1.0000         0         0         0         0         0         0 ···
         0    1.5000   -1.0000         0         0         0         0         0
         0         0    1.3333   -1.0000         0         0         0         0
         0         0         0    1.2500   -1.0000         0         0         0
         0         0         0         0    1.2000   -1.0000         0         0
         0         0         0         0         0    1.1667   -1.0000         0
         0         0         0         0         0         0    1.1429   -1.0000
         0         0         0         0         0         0         0    1.1250
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
     ⋮
```

```
p
```

```
p = 100×1
     1
     2
     3
     4
     5
     6
     7
     8
     9
    10
     ⋮
```

```
disp("Convertimos nuestro vector de permutación en una matriz y queda como:")
```

Convertimos nuestro vector de permutación en una matriz y queda como:

```
n = size(matriz,1);
P = eye(n,n);
P(:,:) = P(p',:);
P
```

```
P = 100×100
     1     0     0     0     0     0     0     0     0     0     0     0     0 ···
```

```
     0     1     0     0     0     0     0     0     0     0     0     0     0
     0     0     1     0     0     0     0     0     0     0     0     0     0
     0     0     0     1     0     0     0     0     0     0     0     0     0
     0     0     0     0     1     0     0     0     0     0     0     0     0
     0     0     0     0     0     1     0     0     0     0     0     0     0
     0     0     0     0     0     0     1     0     0     0     0     0     0
     0     0     0     0     0     0     0     1     0     0     0     0     0
     0     0     0     0     0     0     0     0     1     0     0     0     0
     0     0     0     0     0     0     0     0     0     1     0     0     0
     ⋮
```

```
disp("Entonces A = P*L*U ya que P es la identidad por lo tanto la inversa de P es la misma P");
```

```
Entonces A = P*L*U ya que P es la identidad por lo tanto la inversa de P es la misma P
```

```
M_A = P*L*U
```

```
M_A = 100×100
     2    -1     0     0     0     0     0     0     0     0     0     0     0 ···
    -1     2    -1     0     0     0     0     0     0     0     0     0     0
     0    -1     2    -1     0     0     0     0     0     0     0     0     0
     0     0    -1     2    -1     0     0     0     0     0     0     0     0
     0     0     0    -1     2    -1     0     0     0     0     0     0     0
     0     0     0     0    -1     2    -1     0     0     0     0     0     0
     0     0     0     0     0    -1     2    -1     0     0     0     0     0
     0     0     0     0     0     0    -1     2    -1     0     0     0     0
     0     0     0     0     0     0     0    -1     2    -1     0     0     0
     0     0     0     0     0     0     0     0    -1     2    -1     0     0
     ⋮
```

```
disp("Entonces tenemos la matriz");
```

```
Entonces tenemos la matriz
```

Otro ejemplo.

```
disp("Otro ejemplo de Factorización PA = LU -> A = P'*L*U");
```

```
Otro ejemplo de Factorización PA = LU -> A = P'*L*U
```

```
disp("Matriz A");
```

```
Matriz A
```

```
A
```

```
A = 4×4
     4     2     1     5
     8     3     1    -4
     2     4     3     3
     1     2     4     1
```

```
disp("Aplicando lutx a nuestra Matriz A tenemos:")
```

```
Aplicando lutx a nuestra Matriz A tenemos:
```

```
[L,U,p,~] = lutx(A);
L
```

```
L = 4×4
    1.0000         0         0         0
    0.2500    1.0000         0         0
    0.1250    0.5000    1.0000         0
    0.5000    0.1538    0.0308    1.0000
```

```
U
```

```
U = 4×4
    8.0000    3.0000    1.0000   -4.0000
         0    3.2500    2.7500    4.0000
         0         0    2.5000   -0.5000
         0         0         0    6.4000
```

```
p
```

```
p = 4×1
     2
     3
     4
     1
```

```
disp("Convertimos nuestro vector de permutación en una matriz y queda como:")
```

```
Convertimos nuestro vector de permutación en una matriz y queda como:
```

```
n = size(A,1);
P = eye(n,n);
P(:,:) = P(p',:);
P
```

```
P = 4×4
     0     1     0     0
     0     0     1     0
     0     0     0     1
     1     0     0     0
```

```
%Calculando la inversa para
disp("La inversa de P es:")
```

```
La inversa de P es:
```

```
P_ = inversa_m(P)
```

```
P_ = 4×4
     0     0     0     1
     1     0     0     0
     0     1     0     0
     0     0     1     0
```

```
disp("Entonces A = P_*L*U");
```

```
Entonces A = P_*L*U
```

```
A = P_*L*U
```

```
A = 4×4
     4     2     1     5
     8     3     1    -4
     2     4     3     3
     1     2     4     1
```

```
disp("Entonces tenemos la matriz A");
```

```
Entonces tenemos la matriz A
```

## Parte 2.

### 5 Investigar a los algoritmos de factorización

• Investigar el método de Dolittle para obtener la factorización LU. Incluir un pseudocódigo y un diagrama de flujo del método de Dolittle.

## Método de Doolittle.

Método para resolver sistemas lineales de ecuaciones algebraicas es el método de descomposición. En la descomposición o factorización LU, la matriz [L] tiene números 1 en su diagonal principal. Al método de Crout también se le denomina formalmente método de Doolittle.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

Este método genera [U] y [L] recorriendo las columnas y las filas de la matriz de forma alternada y se expresa mediante las siguientes fórmulas:

$$u_{1j} = a_{1j}, \quad j = 1, 2, \ldots, n.$$
$$l_{i1} = \frac{a_{i1}}{u_{11}}, \quad i = 2, \ldots, n.$$
$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}, \quad i = 2, \ldots, n \text{ y } j = i, i+1, \ldots, n.$$
$$l_{kj} = \frac{a_{kj} - \sum_{i=1}^{j-1} l_{ki} u_{ij}}{u_{jj}}, \quad j = 2, \ldots, n-1 \text{ y } k = j+1, \ldots, n. \qquad (4)$$

el método presenta ventajas para su implementación computacional sobre el método LU.

Supongamos que luego de la FEA(Fase de eliminación hacía adelante) del método Gauss simple se tiene una SEL(Sistema de ecuaciones lineales) con la forma:

$$U\vec{x} = \vec{d}, \qquad (1)$$

Es decir, que U es una matriz triangular superior y que a partir de (1) puede resolverse el SEL a tráves de la FSA.

Si se obtinen las fórmulas para determinar los elementos U y $\vec{d}$ es posible resolver el sistema original, $A\vec{x} = \vec{b}$.

Supongamos que L es una matriz triangular inferior con $L_{ii} = 1$ para toda $1 \le i \le n$. De (1) tenemos que:

$$U\vec{x} - \vec{d} = \vec{0}$$
$$L\left(U\vec{x} - \vec{d}\right) = \vec{0}$$
$$(LU)\,\vec{x} = L\vec{d}.$$

$(LU)\vec{x} = L\vec{d}$ es igual con $A\vec{x} = \vec{b}$ si y sólo si.

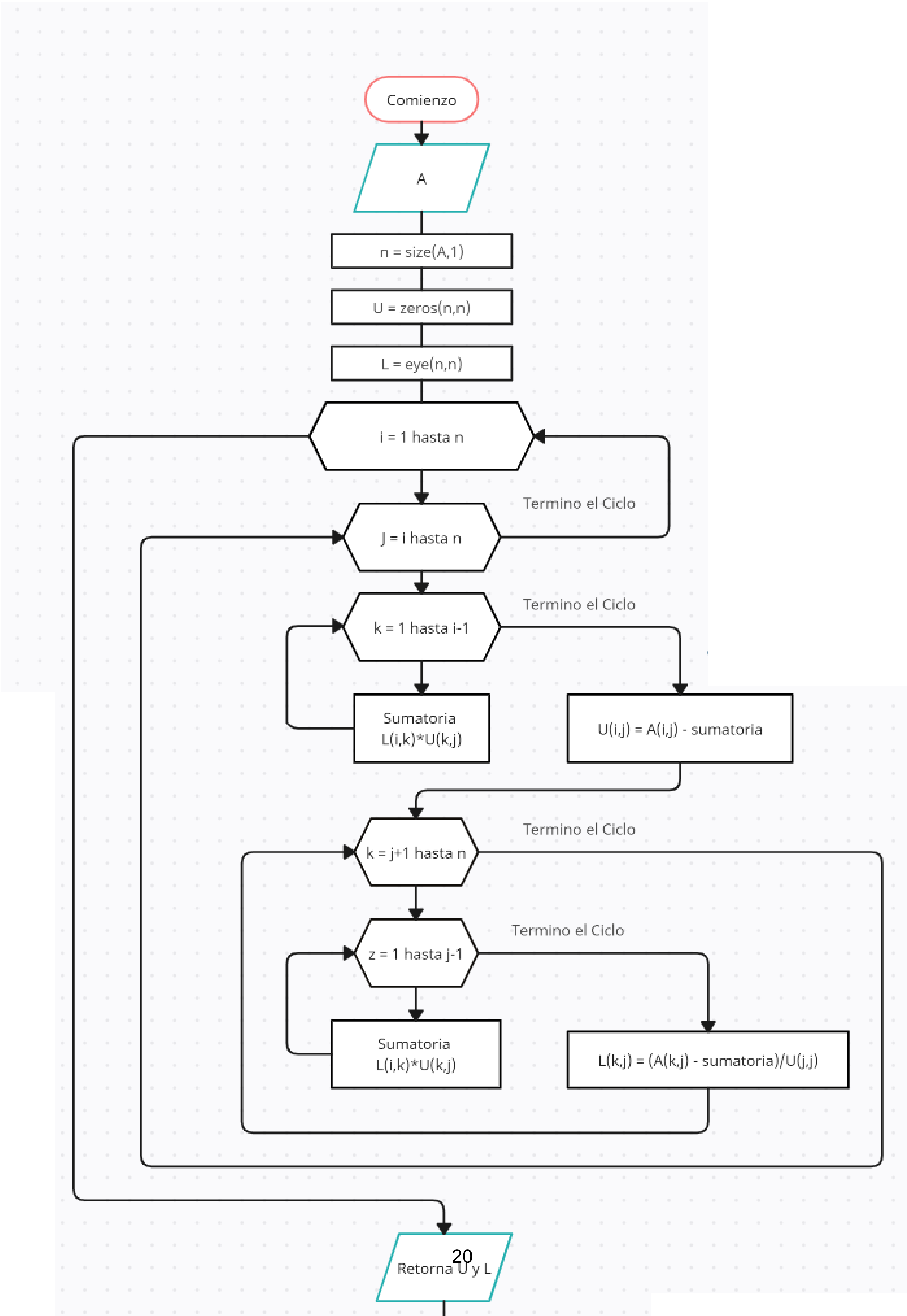
$$LU = A$$
$$\text{y } L\vec{d} = \vec{b}.$$

**Pseudocódigo.**

1. Declarar la matriz A
2. n = tamaño(A) solo filas, se asume que es cuadra
3. creamos U = ceros (n,n)
4. creamos L = identidad (n,n)
5. Para i = 1 hasta n
6. Para j = 1 hasta n
7. iniciamos un contador con cero c = 0

8. Para k =   hasta i-1
9. aplicamos c = c + L(i,k)*U(k,j)
10. fin
11. 
12. aplicamos U(i,j) = A(i,j) - c
13. 
14. Para k = j+1 hasta n
15. reiniciamos el contador c = 0
16. para z = 1 hasta j-1
17. aplicamos c = c  + L(k,z)*U(z,j)
18. fin
19. aplicamos L(k,j) = (A(k,j)-contador)/U(j,j)
20. fin
21. fin
22. fin
23. Se retorna U y L.

**Diagrama de Flujo.**


Comienzo
A
n = size(A,1)
U = zeros(n,n)
L = eye(n,n)
i = 1 hasta n
Termino el Ciclo
J = i hasta n
Termino el Ciclo
k = 1 hasta i-1
Sumatoria
L(i,k)*U(k,j)
U(i,j) = A(i,j) - sumatoria
Termino el Ciclo
k = j+1 hasta n
Termino el Ciclo
z = 1 hasta j-1
Sumatoria
L(i,k)*U(k,j)
L(k,j) = (A(k,j) - sumatoria)/U(j,j)
Retorna U y L

**Programa. Punto adicional.**

```
disp("Sea la matriz A");
```

```
Sea la matriz A
```

```
A = [2 1 5;
     4 4 -4;
     1 3 1]
```

```
A = 3×3
     2     1     5
     4     4    -4
     1     3     1
```

```
disp("Y el vector b");
```

```
Y el vector b
```

```
b = [5;0;6]
```

```
b = 3×1
     5
     0
     6
```

```
disp("Sacamos la factorización LU por el método Doolittle");
```

```
Sacamos la factorización LU por el método Doolittle
```

```
[L,U] = Doolittle_extra(A)
```

```
L = 3×3
    1.0000         0         0
    2.0000    1.0000         0
    0.5000    1.2500    1.0000
U = 3×3
     2     1     5
     0     2   -14
     0     0    16
```

```
disp("Entonces obtenemos el vector de terminos independientes");
```

```
Entonces obtenemos el vector de terminos independientes
```

```
disp("Tal que Ld=b");
```

```
Tal que Ld=b
```

```
d = forward(L,b)
```

```
d = 3×1
     5
   -10
    16
```

```
disp("Con esto ahora podemos resolver Ux=d");
```

```
Con esto ahora podemos resolver Ux=d
```

```
x = backups(U,d)
```

```
x = 3×1
    -1
     2
     1
```

```
disp("Entonces x es nuestra solución al sistema");
```

```
Entonces x es nuestra solución al sistema
```

```
disp("[A,b]");
```

```
[A,b]
```

```
[A,b]
```

```
ans = 3×4
     2     1     5     5
     4     4    -4     0
     1     3     1     6
```

• **Investigar la factorización de Choleski para factorizar una matriz simétrica.**

## Método de Cholesky

Se puede comprobar que, si A es una matriz simétrica y definida positiva, admite una descomposición en la forma LU con U = L traspuesta , es decir, A = L L traspuesta

En este caso, el algoritmo de cálculo de los elemento de L es.

$$\begin{cases} l_{11} = \sqrt{a_{11}}, \\ l_{i1} = \dfrac{a_{i1}}{l_{11}}, \quad i \geq 2, \\ l_{kk} = \sqrt{a_{kk} - \displaystyle\sum_{r=1}^{k-1} l_{kr}^2}, \quad k \geq 2, \\ l_{ik} = \dfrac{a_{ik} - \displaystyle\sum_{r=1}^{k-1} l_{ir} l_{kr}}{l_{kk}}, \quad i > k. \end{cases}$$

Por ejemplo, la matriz

$$A = \begin{pmatrix} 1 & -1 & 1 \\ -1 & 5 & -5 \\ 1 & -5 & 6 \end{pmatrix}$$

Es simétrica, obviamente, y sus menores principales son:

$$|1| = 1 > 0,$$

$$\begin{vmatrix} 1 & -1 \\ -1 & 5 \end{vmatrix} = 4 > 0,$$

$$|A| = 4 > 0,$$

por lo que A es definida positiva. Por tanto, admite una descomposición del tipo L L traspuesta mediante el método de Cholesky descrito. Aplicando el método se tiene que:

$$l_{11} = \sqrt{1} = 1,$$

$$l_{21} = \frac{-1}{1} = -1,$$

$$l_{31} = \frac{1}{1} = 1,$$

$$l_{22} = \sqrt{5 - (-1)^2} = 2,$$

$$l_{32} = \frac{-5 - 1.(-1)}{2} = -2,$$

$$l_{33} = \sqrt{6 - 1 - (-2)^2} = 1,$$

y por tanto

$$L = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 2 & 0 \\ 1 & -2 & 1 \end{pmatrix}.$$

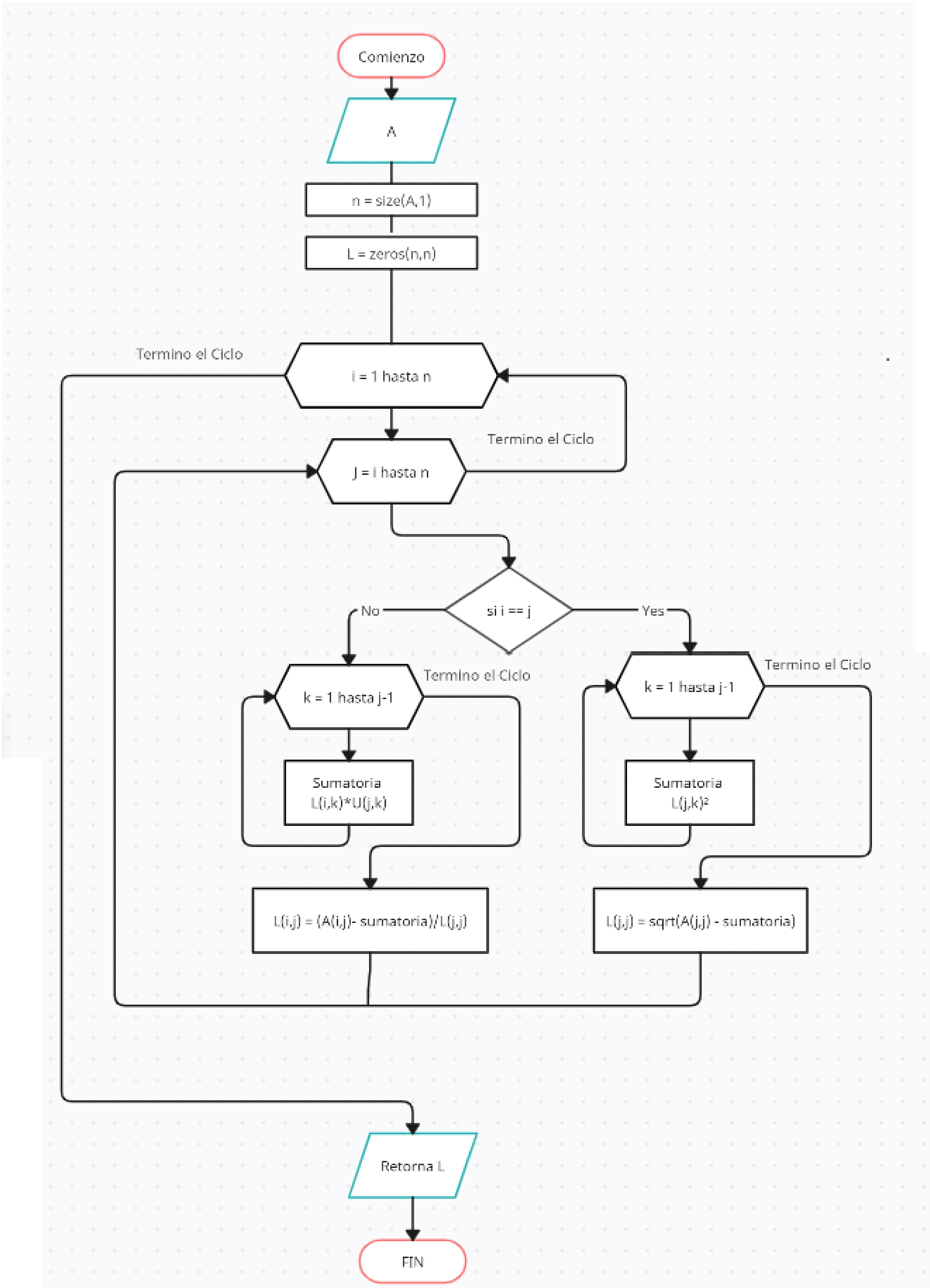
Comprobémoslo:

$$LL^T = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 2 & 0 \\ 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} 1 & -1 & 1 \\ 0 & 2 & -2 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 1 \\ -1 & 5 & -5 \\ 1 & -5 & 6 \end{pmatrix} = A.$$

**Pseudocódigo.**

1. n = tamaño de A
2. Crear L = ceros ( n, n)
3. Para i = 1 hasta n
4. Para j = 1 hasta i
5. si i == j
6. calculamos la suma de los cuadrados
7. contador = 0
8. Para k = 1 hasta j-1
9. suma = suma + L(j,k)^2
10. fin
11. calculamos L(j,j) = sqrt(A(j,j)-suma)
12. sino
13. calculamos la suma de los productos anteriores
14. contador = 0
15. Para k = 1 hasta j-1
16. suma = suma + L(i,k)*L(j,j)
17. fin
18. calculamos L(i,j) = (A(i,j)-suma)/L(j,i)
19. fin
20. fin
21. fin
22. retornamos L

**Diagrama de Flujo.**


Comienzo
A
n = size(A,1)
L = zeros(n,n)
Termino el Ciclo
i = 1 hasta n
Termino el Ciclo
J = i hasta n
si i == j
No
Yes
k = 1 hasta j-1
Termino el Ciclo
k = 1 hasta j-1
Termino el Ciclo
Sumatoria
L(i,k)*U(j,k)
Sumatoria
L(j,k)²
L(i,j) = (A(i,j)- sumatoria)/L(j,j)
L(j,j) = sqrt(A(j,j) - sumatoria)
Retorna L
FIN

## 6 Manipulación de imágenes

Aplicando una transformación lineal, generar la figura en amarillo a partir de la poligonal en color azul.

Sea la nuestra función que genera los vectores de la casa azul:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix}$$

```
disp("Manipulación de imagenes");
```

```
Manipulación de imagenes
```

```
%creamos nuestros vectores
v1 = [6;
      3]
```

```
v1 = 2×1
     6
     3
```

```
v2 = [6;
      1]
```

```
v2 = 2×1
     6
     1
```

```
v3 = [8;
      1]
```

```
v3 = 2×1
     8
     1
```

```
v4 = [8;
      3]
```

```
v4 = 2×1
     8
     3
```

```
%para el triangulo
v5 = [7;
      4]
```

```
v5 = 2×1
     7
     4
```

```


%graficar(v1,v2,v3,v4,v5);
```

**Figura 1**

La figura tiene una rotación de 45° hacía la izquierda por lo tanto:

Necesitamos una matriz asociada que nos rote la figura, en este caso, no es necesario hacer la traslación ya que todo se hace desde el origen.

Transformación lineal para la rotación.

$T\begin{bmatrix}x\\y\end{bmatrix} = \begin{bmatrix}\text{xcos}\theta - \text{ysen}\theta\\ \text{xsen}\theta + \text{ycos}\theta\end{bmatrix}$ donde el alguno varía dependiendo la rotación.

La matriz asociada es:

$T\begin{bmatrix}x\\y\end{bmatrix} = \begin{bmatrix}\cos\theta & -\text{sen}\theta\\ \text{sen}\theta & \cos\theta\end{bmatrix}\begin{bmatrix}x\\y\end{bmatrix}$ donde nuestro angulo será de 45

```
m1_asociada =  [cosd(45) -sind(45);
               sind(45) cosd(45)];

%sacamos los vectores
%Transformamos nuestros vectores con la matriz asociada a nuestra
%transformación lineal
u1 = m1_asociada*v1;
u2 = transformacion_l(m1_asociada,v2);
u3 = transformacion_l(m1_asociada,v3);
u4 = transformacion_l(m1_asociada,v4);
u5 = transformacion_l(m1_asociada,v5);
disp("Figura 1");
```

```
Figura 1
```

```
graficar(v1,v2,v3,v4,v5,u1,u2,u3,u4,u5);
title('Figura 1');
```

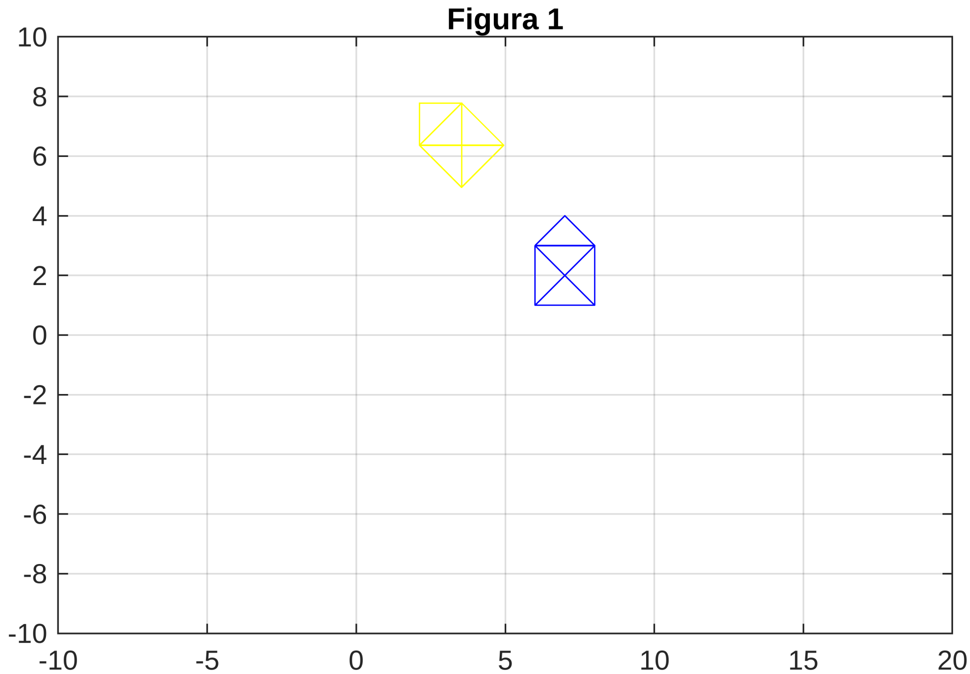



**Figura 2**

```
m_asociada =  [cosd(180) -sind(180);
               sind(180) cosd(180)];

%sacamos los vectores
%Transformamos nuestros vectores con la matriz asociada a nuestra
%transformación lineal
u1 = transformacion_l(m_asociada,v1);
u2 = transformacion_l(m_asociada,v2);
u3 = transformacion_l(m_asociada,v3);
u4 = transformacion_l(m_asociada,v4);
u5 = transformacion_l(m_asociada,v5);
disp("Figura 2");
```

```
Figura 2
```

```
graficar(v1,v2,v3,v4,v5,u1,u2,u3,u4,u5);
title('Figura 2');
```

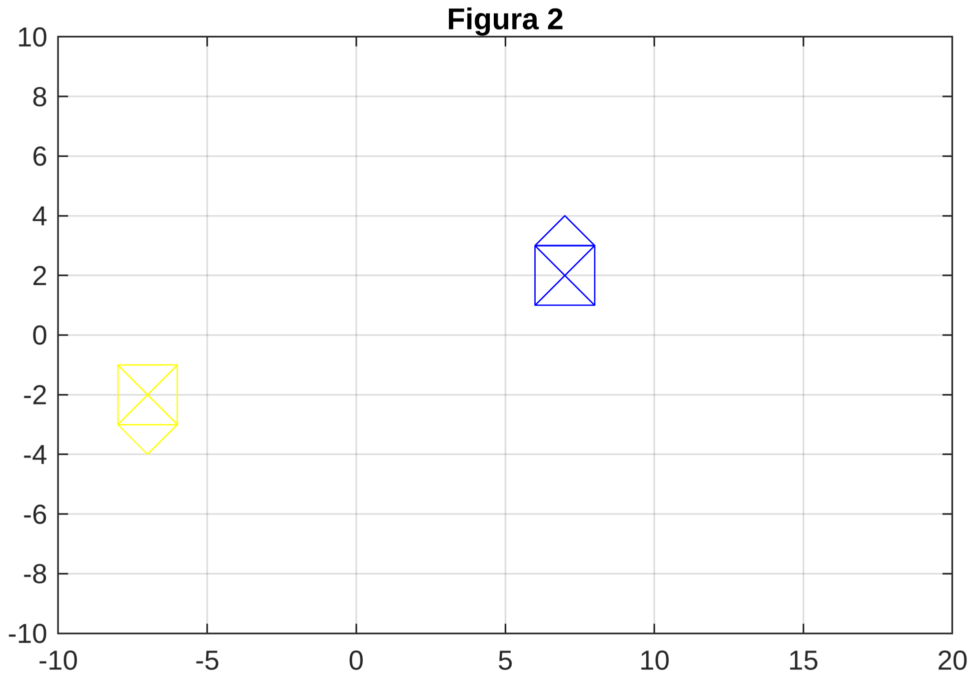



**Figura 3**

```
m_asociada =  [cosd(270) -sind(270);
               sind(270) cosd(270)];

%sacamos los vectores
%Transformamos nuestros vectores con la matriz asociada a nuestra
%transformación lineal
u1 = transformacion_l(m_asociada,v1);
u2 = transformacion_l(m_asociada,v2);
u3 = transformacion_l(m_asociada,v3);
u4 = transformacion_l(m_asociada,v4);
u5 = transformacion_l(m_asociada,v5);
disp("Figura 3");
```

```
Figura 3
```

```
graficar(v1,v2,v3,v4,v5,u1,u2,u3,u4,u5);
title('Figura 3');
```

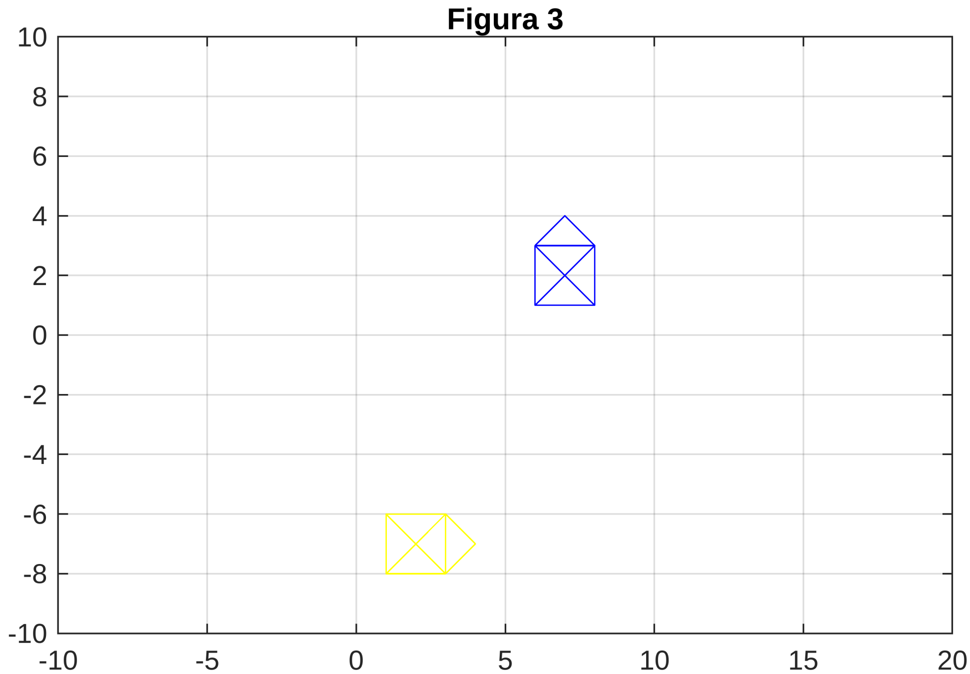



**Figura 4**

**La figura tiene una rotación de 45° hacía la derecha por lo tanto:**

$T\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\mathrm{sen}\theta \\ \mathrm{sen}\theta & \cos\theta \end{bmatrix}$ donde nuestro angulo será de -45

```
m_asociada =  [cosd(-45) -sind(-45);
               sind(-45) cosd(-45)];

%sacamos los vectores
%Transformamos nuestros vectores con la matriz asociada a nuestra
%transformación lineal
u1 = transformacion_l(m_asociada,v1);
u2 = transformacion_l(m_asociada,v2);
u3 = transformacion_l(m_asociada,v3);
u4 = transformacion_l(m_asociada,v4);
u5 = transformacion_l(m_asociada,v5);
disp("Figura 4");
```

```
Figura 4
```

```
graficar(v1,v2,v3,v4,v5,u1,u2,u3,u4,u5);
title('Figura 4');
```

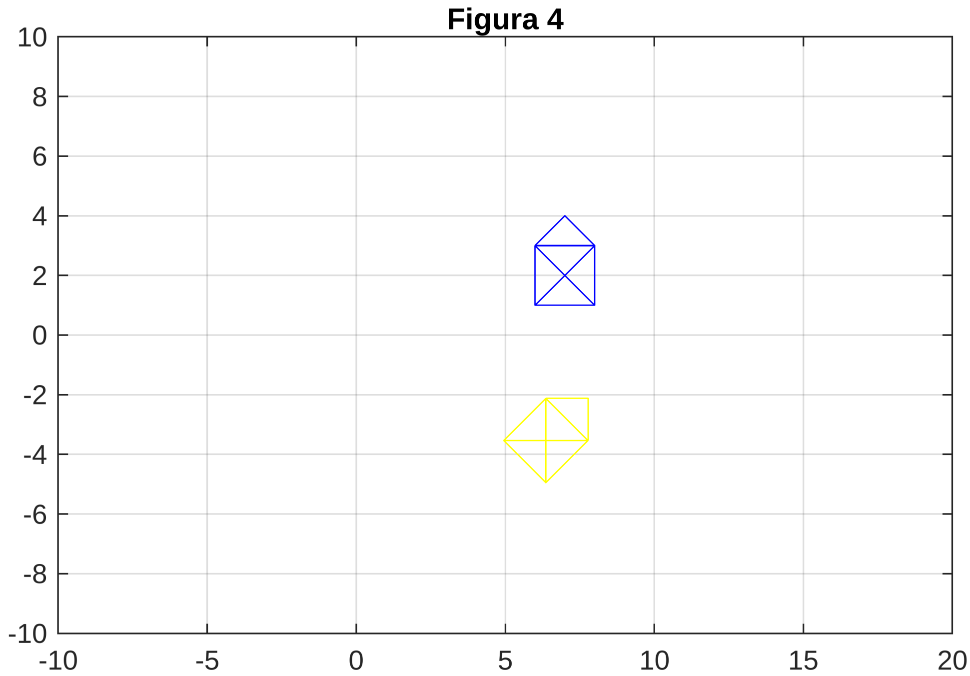



**Figura 5**

Creamos nuestra transformación tal que X se vuelve 2X y Y se vuelve 3Y

por lo tanto la transformación lineal es:

$$T\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2x \\ 3y \end{bmatrix}$$

entonces

$$T(V_n) = u_n$$

La matriz asociada es

$$\begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} V_n = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} u_x \\ u_y \end{bmatrix}$$

```
%Creamos la matriz
m_asociada =  [2 0;
               0 3];

%sacamos los vectores con nuestra matriz asociada a nuestra transformación
%lineal
u1 = transformacion_l(m_asociada,v1);
u2 = transformacion_l(m_asociada,v2);
u3 = transformacion_l(m_asociada,v3);
```

```
u4 = transformacion_l(m_asociada,v4);
u5 = transformacion_l(m_asociada,v5);
disp("Figura 5");
```

```
Figura 5
```

```
graficar(v1,v2,v3,v4,v5,u1,u2,u3,u4,u5);
title('Figura 5');
ax = gca;
chart = ax.Children(5);
datatip(chart,8,1);
datatip(chart,6,3);
chart = ax.Children(2);
datatip(chart,16,3);
datatip(chart,12,9);
```