

□ **Sesión 4: Resolución de sistemas de ecuaciones lineales: métodos directos.**

□ ***1 Recordatorio sobre matrices, vectores y producto de matrices y vectores***

```
□ --> AA : matrix([1, 2, 1], [2, 5, 0], [-3, 2, 6]);  
    cc1 : matrix([4, -2, 1]);  
    cc2 : matrix([4], [-2], [1]);  
  
□ --> AA.cc1;  
  
□ --> AA.cc2;  
  
□ --> cc1.AA;  
  
□ --> cc2.AA;
```

□ ***2 Método de Gauss***

```
□ --> cc : matrix([4, -2, 1]) $
```

□ ***2.1 Gauss evitando ceros***

```
□ La expresión ``A[i,k] # 0`` sólo tiene sentido si operamos con valores exactos. Por tanto, usaremos  
  ``abs(A[i,k]) > tol`` siendo tol una tolerancia predefinida.
```

```
--> A : copymatrix(AA) $
c : copymatrix(cc) $
b : A.c $
n : length(A) $
tol : 10^(-15) $
print("Sistema original:") $
print(A, ".x=", b) $
print("Método de Gauss:") $
for k:1 while k < n do (
  filabuena : 0,
  for i:k while i <= n do (
    if abs(A[i,k]) > tol then (
      filabuena : i,
      i : n+1
    )
  ),
  if filabuena = 0 then return(print("ERROR: Matriz singular")),
  if filabuena > k then (
    print("cero en fila ", k, "; cambio con fila ", filabuena),
    tmp : A[k],
    A[k] : A[filabuena],
    A[filabuena] : tmp,
    tmp : b[k],
    b[k] : b[filabuena],
    b[filabuena] : tmp
  ),
  for i:k+1 while i<=n do (
    m : -A[i,k]/A[k,k],
    A[i] : A[i] + m*A[k],
    b[i] : b[i] + m*b[k]
  ),
  print(A, ".x=", b)
) $
```

```
--> x: genmatrix(lambda([i,j], 0), n, 1) $  
/* Otra opción: x: zeromatrix (n, 1) */  
  
for i:n thru 1 step -1 do  
x[i] : (b[i] - sum(A[i,j]*x[j],j,i+1,n))/A[i,i] $  
  
print("x=",x) $
```

Comprobemos que la solución obtenida es la correcta.

```
--> A.x - b;
```

2.2 Gauss con pivote parcial

EJERCICIO 1: Modifica el algoritmo anterior para realizar una estrategia de pivote parcial.

EJERCICIO 2: Resuelve, usando el Método de Gauss con estrategia de pivote parcial que has implementado en el ejercicio 1, el sistema de ecuaciones lineales $A.X=b$, donde A y b se describen a continuación.

A es una matriz 8×8 , cuyas entradas son:

- en la diagonal principal $a[i,i] = 1/i$,
- por encima de la diagonal $a[i,j] = j/i$,
- por debajo de la diagonal $a[i,j] = i/j$.

El vector de términos independientes tiene coordenadas $b[i] = \cos(i)$.

2.3 Gauss con pivote parcial escalado

EJERCICIO 3: Modifica el algoritmo anterior para realizar una estrategia de pivote parcial escalado y úsalo para resolver el mismo sistema del ejercicio 2.

3 Descomposición LU

□ 3.1 Algoritmo de factorización de Doolittle

```

--> A : copymatrix(AA) $
    c : copymatrix(cc) $
    b : A.c $
    [n,n2] : matrix_size (A) $

    L : ident(n) $
    U : zeromatrix(n,n) $

    for k:1 thru n do (
        U[k,k] : A[k,k]-sum(L[k,r]*U[r,k],r,1,k-1),
        for j:k+1 thru n do (
            U[k,j] : A[k,j]-sum(L[k,r]*U[r,j],r,1,k-1),
            L[j,k] : (A[j,k]-sum(L[j,r]*U[r,k],r,1,k-1))/U[k,k]
        )
    ) $

    LD : copymatrix(L) $
    UD : copymatrix(U) $

    print(A, "=", LD, UD, "=", LD.UD) $

```

□ 3.2 Resolución del sistema dada la descomposición LU

✎ EJERCICIO 4: Implementa un algoritmo que permita resolver el sistema lineal $A.x=b$ a partir de la factorización LU (de la matriz A) calculada anteriormente.

□ 3.3 Algoritmo de factorización de Crout

✎ EJERCICIO 5: Implementa un algoritmo que permita calcular la factorización LU de Crout de la matriz A y resuelve el sistema lineal $A.x=b$.