Sesión 5: Resolución de sistemas de ecuaciones lineales: métodos iterativos.

Una orden para que sólo aparezcan unos cuantos dígitos.

```
--> fpprintprec : 6 $
```

1 Método de Jacobi

Consideramos un ejemplo con una matriz de dimensión baja para controlar la corrección de la implementación.

```
--> A : matrix([10,2,1],[3,7,1],[1,3,6]) $
    b : matrix([3],[6],[9]) $
    n : 3 $
```

En esta implementación se hacen 20 iteraciones.
Declaramos x,y con ``float´´ para que los cálculos sean aproximados.
La orden ``print("Iteración ", m, ": ", transpose(y), ", dif =", dif)´´ es para controlar visualmente la convergencia.

```
--> x : float(zeromatrix(n,1)) $
    y : float(zeromatrix(n,1)) $

    for m:1 thru 20 do(
        for i:1 thru n do (
            y[i,1] : (b[i,1]-sum(A[i,j]*x[j,1],j,1,i-1)-sum(A[i,j]*x[j,1],j,i+1,n))/A[i,i]
        ),
        dif : sum(abs(x[i,1]-y[i,1]),i,1,n),
        print("Iteración " , m , ":  " , transpose(y), ", dif =" , dif),
        x : copymatrix(y)
    ) $
```

Cuando los vectores se indican por columna parece que no hace falta indicar el segundo subíndice.
Veamos este hecho en la siguiente modificación de la anterior implementación.

```
--> x : float(zeromatrix(n, 1)) $
    y : float(zeromatrix(n, 1)) $

    for m:1 thru 20 do(
        for i:1 thru n do (
            y[i] : (b[i]-sum(A[i,j]*x[j],j,1,i-1)-sum(A[i,j]*x[j],j,i+1,n))/A[i,i]
        ),
        dif : sum(abs(x[i,1]-y[i,1]),i,1,n),
        print("Iteración ",m,":  ",transpose(y), ", dif= " , dif),
        x : copymatrix(y)
    ) $
```

2 Ejercicio: Método de Gauss-Seidel

Implementa el método de Gauss-Seidel. Para controlar la corrección de la implementación utiliza el mismo sistema que el dado en el método de Jacobi.

3 Ejemplos para comparar ambos métodos

A continuación damos ejemplos de matrices para observar la diferencia entre los métodos de Jacobi y Gauss-Seidel. Es posible que tengas que modificar el valor de "fpprintprec" para observar cada caso.

La dimensión de las matrices dependerá del siguiente valor prefijado.

```
--> n : 8 $
```

Ejemplo 1: ambos convergen muy rápido.

```
--> A : genmatrix(lambda([i,j], float(sin(i)+cos(j))), n, n)+diagmatrix (n, 2*n) $
    b : A.makelist(i, i, 1, n) $
```

Ejemplo 2: ambos convergen despacio, si bien Gauss-Seidel es más rápido que Jacobi.

```
 -->  A : diagmatrix(n,2.0) $
      for i:1 thru n-1 do (
          A[i,i+1] : -1.0/2,
          A[i+1,i] : -1.0/2
      ) $
      b : A.makelist(i, i, 1, n) $
```

Ejemplo 3: ambos convergen muy despacio, si bien Gauss-Seidel es más rápido que Jacobi.

```
 -->  A : diagmatrix(n,6.5) $
      for i:1 thru n-1 do (
          A[i,i+1] : -1.0,
          A[i+1,i] : -1.0
      )$
      for i:1 thru n-3 do (
          A[i,i+3] : -2.0,
          A[i+3,i] : -2.0
      )$
      b : A.makelist(i, i, 1, n) $
```

Ejemplo 4: Jacobi converge despacio y Gauss-Seidel converge bastante más rápido.

```
 -->  A : diagmatrix(n,2.0) $
      for i:1 thru n-1 do (
          for j:i+1 thru n do (
              A[i,j] : 1.0/(i+j),
              A[j,i] : 1.0/(i+j)
          )
      )$
      b : A.makelist(i, i, 1, n) $
```

Ejemplo 5: Jacobi converge muy muy despacio, Gauss-Seidel converge bastante rápido.

```
-->  A : diagmatrix(n,1.0) $
     for i:1 thru n-1 do (
         for j:i+1 thru n do (
             A[i,j] : 1.0/(i+j),
             A[j,i] : 1.0/(i+j)
         )
     )$
     b : A.makelist(i, i, 1, n) $
```

Ejemplo 6: Jacobi no converge, Gauss-Seidel sí converge (aunque muy despacio).

```
-->  A : zeromatrix(n,n) $
     for i:1 thru n do (
         for j:1 thru n do (
             A[i,j] : float(n-abs(i-j))
         )
     )$
     b : A.makelist(i, i, 1, n) $
```

Ejemplo 7: Jacobi sí converge, Gauss-Seidel no converge.

```
-->  dd : zeromatrix(n,n) $
     for i:1 thru n-1 do (
         dd[i,i+1] : 1.0
     ) $
     pp : diagmatrix(n,1.0) $
     for i:1 thru n do (
         for j:1 thru n do (
             if integerp((i-j)/2)=true then pp[i,j] : float(abs(i-j))
         )
     ) $
     bj : pp.dd.invert(pp) $

     A : bj+diagmatrix(n,1.0) $
     b : A.makelist(i, i, 1, n) $
```

4 Ejercicio: Método de relajación

Implementa el método de relajación. Para controlar la corrección de la implementación utiliza el mismo sistema que el dado en el método de Jacobi.

5 Ejercicio: otros criterios de parada

Modifica las implementaciones realizadas para incluir un criterio de parada que dependa de la diferencia entre dos iteraciones sucesivas.

¿Podrías diseñar algún otro criterio de parada?

6 Ejercicio para pensar detenidamente

Implementa el método de Gauss-Siedel, con el primer criterio de parada del ejercicio 5, de manera que sí se utilice el vector x pero no el vector y. Observa que de esta forma reduciremos costes.