Práctica 2: Resolución de sistemas de ecuaciones lineales
MNI, Curso 20/21

1 Métodos directos

Aunque no es objeto de la asignatura, puede ser útil reparar en que el comando linsolve discute y, en su caso, resuelve simultáneamente un sistema de ecuaciones lineales:

```
(%i1) linsolve([2·x+y-3·z=0, x+y+z=1,x-y+9·z=-2], [x,y,z]);
```

```
(%i2) linsolve([2·x+y-3·z=0, x+y+z=1,x+y+z=11], [x,y,z]);
```

```
(%i3) linsolve([2·x+y-3·z-7·w-t=1, x+y+z+w+t=1,2.5·x+
         1.5·y-2.5·z-7.5·w=0.5], [x,y,z,w,t]);
```

Puedes recurrir al menú "Ecuaciones/Resolver sistema lineal" para generar fácilmente la orden anterior.

Con las herramientas de que dispones, puedes programar el método de Gauss y, de hecho, aparece propuesto como el segundo ejercicio.

Con relación a los métodos de factorización LU como Doolittle y Crout, Maxima cuenta con el comando lu_factor, que aplicado a una matriz regular a da como salida su factorización LU tipo Doolittle. El primer argumento de la salida da una matriz s que, de forma compacta, no es más que la factorización LU de a: L es la matriz triangular inferior con 1's en la diagonal principal y con la parte inferior de s, y U es la matriz triangular superior con la diagonal principal de s y la parte superior de s. El segundo argumento indica la eventual permutación de filas.

```
(%i1) a:matrix([2,2,1,2],[4,6,1,3],[6,12,1,3],[2,4,-1,-1]);
```

```
(%i2) lu_factor(a);
```

Es decir, la matriz a admite la factorización tipo Doolittle

```
(%i3) l:matrix([1, 0, 0, 0],[2, 1,0,0],[3,3,1,0],[1,1,-1,1]);
```

```
(%i4) u:matrix([2, 2, 1, 2],[0, 2, -1,-1],[0,0,1,0],[0,0,0,-2]);
```

Por su parte, la sentencia lu_backsub(lu_factor) devuelve la solución del sistema ax=b a partir de la factorización LU de a que acabamos de obtener:

```
(%i5) b:transpose([1,-2,3,-4]);
```

```
(%i6) lu_backsub(lu_factor(a),b);
```

```
(%i7) ;
```

2 Metodos iterativos

Los métodos iterativos de resolución de sistemas de ecuaciones lineales pueden implementarse mediante el uso de sentencias trabajadas en la práctica anterior, y así se propone en los ejercicios.

```
(%i1) ;
```

3 Ejercicios

1.- Programa la resolución de un sistema triangular superior compatible determinado. Aplícalo al sistema de matriz de coeficientes

```
(%i2) matrix([0.34,-1.99,2/7,0],[0,1.1,2.3,-3.57],[0,0,3.2,33],
      [0,0,0,66.72]);
```

y vector de términos independientes

```
(%i3) [1,34,78,-9.42].;
```

2.- Programa el método de Gauss y úsalo para resolver el sistema con matriz de coeficientes

```
(%i4) matrix([0.24,1.1,3/2,3.45],[-1.2,1,3.5,6.7],[33.1,1,2,-3/8],[4,17,71,-4/81]);
```

y vector de términos independientes

```
(%i5) [1,2,4,-21/785].;
```

3.- Programa el método de Crout y aplícalo para encontrar la solución del sistema con matriz de coeficientes y vector de términos independientes, respectivamente

```
matrix([3,6,9],[1,4,11],[0,4,19]);
```

y

```
[1/2,-2/3,-3/4].;
```

4.- Programa los métodos de Jacobi y Gauss-Seidel y aplícalos, partiendo de la iteración inicial

```
[1,-1.34,1.456];
```

y realizando 15 iteraciones, para obtener una aproximación de la solución del sistema con matriz de coeficientes

```
matrix([3,-2,0.25],[2,9,-5],[2,3,-6]);
```

y vector de términos independientes

```
[1.1,2.2,3.3].;
```