Práctica 4

4.1 Diagonalización

El paquete `eigen` contiene funciones para el cálculo simbólico de valores y vectores propios. *Maxima* carga el paquete automáticamente si se hace una llamada a cualquiera de las dos funciones eigenvalues o eigenvectors que veremos a continuación. Si eigen.mac no está ya cargado, load ("eigen") lo carga.

Intoducimos la matriz

```
--> m:matrix([1,2,-2],[-2,5,-2],[-6,6,-3]);
```

la orden `charpoly(m,x)` proporciona el polinomio característico de la matriz `m` en la variable `x`

```
--> charpoly(m,x);
    expand(%);
```

con lo que los valores propios se obtienen con

```
--> solve(%,x);
```

y así, los valores propios de `m` son 3 y -3.
La orden

```
--> eigenvalues(m);
```

o `eivals(m)`, devuelve una lista con dos sublistas

$$[[\lambda_1, \ldots, \lambda_k], [m_a(\lambda_1), \ldots, m_a(\lambda_k)]]$$

La primera sublista la forman los distintos valores propios de la matriz `m` y la segunda sus correspondientes multiplicidades algebraicas. Con la matriz `m` de nuestro ejemplo, vemos que el valor propio 3 tiene multiplicidad algebraica 2 y el -3 es simple. Para calcular las raíces del polinomio característico, eigenvalues también llama a la función `solve` y no siempre podrá encontrar dichas raíces. En esos casos, podemos calcular el polinomio característico y obtener una aproximación de todas sus raíces utilizando el comando `allroots`.
La orden

```
-->     eigenvectors(m);
```

o `eivects(m)`, calcula los vectores propios de la matriz M. Devuelve una lista de listas,

$$[[[\lambda_1,\ldots,\lambda_k],[m_a(\lambda_1),\ldots,m_a(\lambda_k)]],[[v_1^{\lambda_1},\ldots,v_{m_g(\lambda_1)}^{\lambda_1}],\ldots,[[v_1^{\lambda_k},\ldots,v_{m_g(\lambda_k)}^{\lambda_k}]]]$$

la primera con el mismo resultado que la orden `eivals(m)` (es decir, valores propios y multiplicidades algebraicas) y el segundo elemento es una lista de listas de vectores propios, una por cada valor propio, pudiendo haber uno o más vectores propios en cada lista, según sea la multiplicidad geométrica del correspondiente valor. Estas sublistas contienen las bases de los subespacios fundamentales.
En nuestro caso, `m` tiene 3 valores propios (-3 simple y 3 doble). Un vector propio asociado al valor propio -3 es $a_1 = (1,1,3)$ por lo que se verifica $m.(1,1,3)^T = -3(1,1,3)^T$

```
-->     is(m.matrix([1],[1],[3])=-3*matrix([1],[1],[3]));
```

y dos vectores propios, linealmente independientes, asociados al valor propio 3 son $a_2 = (1,0,-1)$ y $a_3 = (0,1,1)$.

Ejercicio 1. Probar que $m.a_2 = 3a_2$ y que $m.a_3 = 3a_3$

Sabemos que si la matriz es de orden n, para que sea diagonalizable debe verificarse que la suma de las multiplicidades algebraicas sea n y que además, para cada valor propio, λ, sea $m_a(\lambda) = m_g(\lambda)$.
Una vez ejecutada la orden `eivects(m)`, si ejecutamos

```
-->     nondiagonalizable;
```

nos devuelve el valor true si la matriz no es diagonalizable o false cuando sí es diagonalizable (siempre tras la ejecución de eigenvectors.)

Cuando una matriz A es diagonalizable, existen matrices P regular y D diagonal, tales que $P^{-1}AP = D$. La matriz P tiene las columnas formadas por las coordenadas de los vectores propios, en nuestro caso, podemos tomar

```
-->     p:transpose(matrix([1,1,3],[1,0,-1],[0,1,1]));
```

la matriz diagonal asociada a P es

```
-->     invert(p).m.p;
```

que tiene en la diagonal principal los valores propios de `m` (el orden depende del orden en el que se han colocado los vectores en las columnas de P)

Para obtener directamente estas matrices vamos a cargar un paquete, **diagonalizacion.mac** donde se define la función

```
diagonalizacion(m);
```

que podemos utilizar cuando ya sabemos que m es diagonalizable. La respuesta es una lista con las dos matrices $[D,P]$.

Ejercicio 2. Estudiar si son diagonalizables las matrices

$$A = \begin{pmatrix} 4 & 0 & 0 \\ -2 & 5 & -2 \\ 4 & 0 & 4 \end{pmatrix}, \quad B = \begin{pmatrix} 4 & 0 & 0 \\ 4 & 0 & 4 \\ 1 & 0 & 4 \end{pmatrix} \text{ y } C = \begin{pmatrix} 1 & 4 & -12 \\ 0 & 3 & -9 \\ 0 & 1 & 3 \end{pmatrix}$$

Ejercicio 3. Con las matrices del ejercicio anterior, si son diagonalizables, hallar las matrices P y D y comprobar que $P^{-1} \dots P = D$

El paquete `diagonalizacion.mac` sigue el mismo proceso que realizaríamos a mano

4.2 Formas cuadráticas

4.2.1 Diagonalización ortogonal

Se necesita tener cargados los paquetes `eigen` y `diagonalizacion.mac`. Se introduce la matriz simétrica de la forma cuadrática.

```
--> a:matrix([3,1,1],[1,3,1],[1,1,3])$
```

Para obtener los valores y vectores propios, utilizamos el paquete para diagonalizar

```
--> diagonalizacion(a)$
```

que nos permite conocer la matriz `p =` $\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & -1 & -1 \end{pmatrix}$ formada por los vectores propios.
Ahora debemos modificar las columnas de esta matriz, para obtener una ortogonal. Para ello, vamos a utilizar la función de *Maxima* `gramschmidt( )` Esta función, cuando actúa sobre una matriz, no altera el valor de la misma. Opera con las filas de la matriz (a nosotros nos interesa operar con las columnas de `p` por lo que utilizaremos la traspuesta) y el valor devuelto es una lista de listas,

```
--> y:gramschmidt(transpose(p));
```

$$[[1,1,1],[1,0,-1],[-\frac{1}{2},1,-\frac{1}{2}]]$$

estos elementos son ortogonales pero no unitarios, es decir, verifican las condiciones

$$\forall i,j = 1,\dots,3,\ i \neq j,\ \sum_{k=1}^{3} y_{ik}y_{jk} = 0$$

pero no las condiciones

$$\forall i = 1,\dots,3,\ \sum_{k=1}^{n} y_{ik}^2 = 1$$

para este último paso, definimos el módulo de un vector

```
--> modulo(v):= if listp(v) then sqrt(apply("+",v^2))
                            else error(v, "no es un vector")$
```

y lo aplicamos a cada uno de los vectores de y

```
--> y[1]:y[1]/modulo(y[1])$
    y[2]:y[2]/modulo(y[2])$
    y[3]:y[3]/modulo(y[3])$
    y;
```

$$[[\frac{1}{\sqrt{3}},\frac{1}{\sqrt{3}},\frac{1}{\sqrt{3}}],[\frac{1}{\sqrt{2}},0,-\frac{1}{\sqrt{2}}],[-\frac{1}{\sqrt{2}\sqrt{3}},\frac{\sqrt{2}}{\sqrt{3}},-\frac{1}{\sqrt{2}\sqrt{3}}]]$$

estos tres vectores ya verifican las condiciones necesarias. Para que estas listas formen las columnas de la matriz escribimos

```
--> b:matrix(y[1],y[2],y[3]);
    p:transpose(b);
```

y comprobamos que P^TAP es diagonal

```
--> transpose(p).a.p;
```

```
--> radcan(%);
```

Ejercicio 4. Hallar matrices P ortogonal y D diagonal tales que $P^TAP = D$ con

$$A = \begin{pmatrix} 1 & 3 & 4 \\ 3 & 1 & 0 \\ 4 & 0 & 1 \end{pmatrix}$$

4.2.2 Diagonalización completando cuadrados

Consideramos la matriz

```
--> a:matrix([1,-2,3],[-2,4,1],[3,1,5])$
```

cuya forma cuadrática asociada es

```
--> x:makelist(x[i],i,1,3)$
    f:expand(matrix(x).a.transpose(matrix(x)));
```

donde vemos que aparece el término x_1^2 (con coeficiente 1) y el coeficiente que acompaña a x_1 es

```
--> v2:coeff(f,x[1],1);
```

por lo que el cambio de variable que vamos a hacer es

```
--> f1:x[1]+v2/2;
```

y la forma cuadrática `f` se puede escribir en la forma $\texttt{f1}^2 + \texttt{nf}$ con

```
-->    nf:expand(f - f1^2);
```

en la que queda el cuadrado de x_3 acompañado del coeficiente -4

```
-->    v1:coeff(nf,x[3],2);
       s:signum(v1);
```

por lo que en la nueva variable, el coeficiente de x_3 será

```
-->    m:sqrt(s*v1);
```

para completar el cuadrado correspondiente a esta variable, repetimos el mismo proceso que con la anterior

```
-->    v2:coeff(nf,x[3],1);
       f2:m*x[3]+s*v2/(2*m);
       nf:expand(nf -s*f2^2);
```

donde ya queda únicamente el término con x_2^2 Así, haciendo ahora

```
-->    v3:coeff(nf,x[2],2);
       s:signum(%);
       m:sqrt(s*v3);
```

```
-->    f3:m*x[2];
       nf:expand(nf -s*f3^2);
```

se obtiene f como suma de 3 cuadrados

```
-->    expand(f-(f1^2-f2^2+f3^2));
```

Las nuevas variables son

$$\left.\begin{array}{rcl} f1 & = & 3x_3 - 2x_2 + x_1 \\ f2 & = & 2x_3 - \frac{7}{2}x_2 \\ f3 & = & \frac{7}{2}x_2 \end{array}\right\}$$

es decir,

$$\begin{pmatrix} f1 \\ f2 \\ f3 \end{pmatrix} = \begin{pmatrix} 1 & -2 & 3 \\ 0 & -\frac{7}{2} & 2 \\ 0 & \frac{7}{2} & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

por lo que, definiendo la matriz p

```
-->    matrix([1,-2,3],[0,- 7/2,2],[0, 7/2,0]);
       p:invert(%);
```

se verifica que

```
-->    transpose(p).a.p;
```

es diagonal.

Para obtener este cálculo directamente vamos a cargar el paquete, **forcuad.mac** en el que se define la función

```
completcuadr(a);
```

cuya respuesta es una lista del tipo `[v,s,p]` en la que `v` nos da las nuevas variables, `s` los elementos de la matriz diagonal y `p` la matriz anterior.

```
--> [v,s,p]:completcuadr(a);
    transpose(p).a.p;
```

Ejercicio 5. Diagonalizar, completando cuadrados, las matrices

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 1 & 1 & 2 & 0 \\ 1 & 0 & 1 & 0 & 2 \\ 1 & 1 & 0 & 2 & 2 \\ 2 & 0 & 2 & 0 & 1 \\ 0 & 2 & 2 & 1 & 0 \end{pmatrix}$$