Práctica 3

3.1 Sistemas de ecuaciones lineales: Métodos iterativos

Estos métodos de resolución de sistemas de ecuaciones lineales proporcionan una aproximación de la solución. Si el sistema a estudiar es

$$AX = B,$$

y tiene por solución X^*, el objetivo es generar una sucesión X_n que converja[1] a X^*.

Para ello, se transforma el sistema en un problema del tipo

$$X = PX + C,$$

y partiendo de un valor inicial X^0, las aproximaciones sucesivas se obtienen de la forma

$$X^{n+1} = PX^n + C.$$

A partir de la matriz A, definimos las siguientes matrices, la diagonal[2]

$$D = \begin{pmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn} \end{pmatrix},$$

y las triangulares

$$A_L = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ -a_{21} & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -a_{n1} & -a_{n2} & -a_{n3} & \dots & 0 \end{pmatrix} \text{ y } A_U = \begin{pmatrix} 0 & -a_{12} & -a_{13} & \dots & -a_{1n} \\ 0 & 0 & -a_{23} & \dots & -a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix},$$

que verifican

$$A = D - A_L - A_U.$$

[1] La convergencia a la solución no está garantizada, ni es el objetivo de esta práctica. Los ejemplos que pongamos aquí estan convenientemente escogidos.

[2] Suponemos que A no tiene nulo ningún elemento diagonal.

3.1.1 Método de Jacobi

Sustituyendo en $AX = B$, la matriz A por $D - A_L - A_U$ queda

$$DX - (A_L + A_U)X = B,$$

es decir,

$$DX = (A_L + A_U)X + B$$

y multiplicando por D^{-1}

$$X = D^{-1}(A_L + A_U)X + D^{-1}B$$

con lo que tomando

$$P = D^{-1}(A_L + A_U)$$

y

$$C = D^{-1}B$$

ya tenemos el problema escrito en la forma deseada.

Las órdenes de *Maxima* que nos permiten obtener estas dos matrices, después de haber introducido A y B, son

```
-->   n:length(A)$
      load("diag")$
      D:makelist(A[i,i],i,1,n)$
      D:diag(D);
      AL:zeromatrix(n,n)$
      for i:1 thru n do for j:1 thru i-1 do AL[i,j]:-A[i,j]$
      AU:D-A-AL;
      p:invert(D).(AL+AU);
      c:invert(D).B;
```

Una vez calculadas P y C, para aplicar el método, se introduce un valor inicial X^0

```
-->   x:transpose([...,  ,...]);
```

y la siguiente aproximación será float(p.x+c); Un criterio de parada habitual, es fijar un valor para la diferencia entre dos aproximaciones sucesivas. Es decir, tomamos un valor *tol* y cuando $\| X^{k+1} - X^k \| < tol$ dejamos de hacer iteraciones, considerando que X^{k+1} es una aproximación aceptable de X^*. Además, para evitar que el número de iteraciones sea excesivo (si la convergencia es lenta) también suele fijarse un número máximo de iteraciones (`maxit`). Una posibilidad, para efectuar estos cálculos con *Maxima* es:

```
-->   tol: ....$     maxit: ....$
```

```
-->   k:1$
      y:float(p.x+c)$
      err:abs(transpose((y-x)))$
      err:makelist(err[1,i],i,1,n)$
      load("descriptive")$
      di:maxi(err)$

-->   for k:2 while k<=maxit  do
      (y:float(p.x+c),
      err:abs(transpose((y-x))),
      err:makelist(err[1,i],i,1,n),
      di:maxi(err), if di<tol then k:maxit,
      x:y,
      print(k,  transpose(x), di));
```

Ejercicio 1. Aplicar el método al sistema

$$\begin{pmatrix} 2 & -1 & 0 \\ 1 & 6 & -2 \\ 4 & 3 & -8 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2 \\ -4 \\ 5 \end{pmatrix}$$

tomando $tol = 10^{-5}$ y `maxit` $= 30$.

3.1.2 Método de Gauss–Seidel

Igual que antes, cambiando A por $D - A_L - A_U$, el sistema queda

$$(D - A_L)X - A_U X = B,$$

es decir, $(D - A_L)X = A_U X + B$ y multiplicando ahora por $(D - A_L)^{-1}$ se obtiene

$$P = (D - A_L)^{-1} A_U$$

y

$$C = (D - A_L)^{-1} B.$$

Ejercicio 2. Adaptando las órdenes del método de Jacobi, programar el método de Gauss–Seidel y aplicar el método programado al sistema

$$\begin{pmatrix} 10 & -1 & 2 & 0 \\ -1 & 11 & -1 & 3 \\ 2 & -1 & 10 & -1 \\ 0 & 3 & -1 & 8 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ t \end{pmatrix} = \begin{pmatrix} 6 \\ 25 \\ -11 \\ 15 \end{pmatrix}$$

tomando $tol = 10^{-6}$ y `maxit` $= 30$.

3.2 Espacios vectoriales

Antes de ejecutar las órdenes de esta sección, borraremos los valores previamente asignados en esta práctica.

```
--> kill(all)
```

Dados los vectores de $\mathbb{R}^3$

$$v_1 = (1,0,0),\ v_2 = (-1,1,0),\ v_3 = (0,-1,1),$$

$$u_1 = (1,0,-1),\ u_2 = (2,1,0),\ u_3 = (-1,1,1),$$

vamos a probar que $B_1 = (v_1, v_2, v_3)$ es base de $\mathbb{R}^3$. Introducimos los vectores

```
--> v1:[1,0,0]$ v2:[-1,1,0]$ v3:[0,-1,1]$
```

y comprobamos que la matriz

```
--> v:matrix(v1,v2,v3);
```

tiene rango 3

```
--> rank(v);
```

Ejercicio 3. Probar que $B_2 = (u_1, u_2, u_3)$ es base de $\mathbb{R}^3$.

Vamos a hallar, mediante la matriz de cambio de base `mc`, las coordenadas en B_2 del vector $\omega \in \mathbb{R}^3$ cuyas coordenadas en B_1 son $(2,1,3)$. Para eso, recordemos que `mc` tiene por columnas las coordenadas de los elementos de B_1 en B_2. Así, si la primera columna de `mc` es (a,b,c), estos escalares verifican

$$(1,0,0) = a(1,0,-1) + b(2,1,0) + c(-1,1,1)$$

es decir,

$$\left.\begin{array}{rcl} 1 & = & a + 2b - c \\ 0 & = & b + c \\ 0 & = & -a + c \end{array}\right\}$$

en el que la columna de términos independientes es el primer vector de B_1. Las otras dos columnas de `mc` se obtienen con los sistemas que tienen la misma matriz de coeficientes y términos independientes segundo y tercer vector de B_1, respectivamente. Por tanto, debemos resolver los sistemas

```
--> eq:a*[1,0,-1] + b*[2,1,0] + c*[-1,1,1];

--> col1:linsolve([eq[1]=v[1,1],eq[2]=v[1,2],eq[3]=v[1,3]],[a,b,c]);
    col2:linsolve([eq[1]=v[2,1],eq[2]=v[2,2],eq[3]=v[2,3]],[a,b,c]);
    col3:linsolve([eq[1]=v[3,1],eq[2]=v[3,2],eq[3]=v[3,3]],[a,b,c]);
```

de las soluciones obtenidas, eliminamos ahora las expresiones del tipo `a=`

```
-->     col1:makelist(rhs(col1[i]),i,1,length(col1));
        col2:makelist(rhs(col2[i]),i,1,length(col2));
        col3:makelist(rhs(col3[i]),i,1,length(col3));
```

y construimos la matriz que tiene esas columnas

```
-->     mc:transpose(matrix(col1,col2,col3));
```

Para calcular las coordenadas del vector $\omega = (2,1,3)_{B_1}$ en B_2

```
-->     mc.matrix([2],[1],[3]);
```

es decir, $\omega = (-7,2,-4)_{B_2}$.

Ejercicio 4. Hallar las coordenadas en B_1 del vector de $\mathbb{R}^3$ cuyas coordenadas en B_2 son $(3,4,5)$.

3.3 Aplicaciones lineales

Sea f la aplicación lineal de $\mathbb{R}^3$ en $\mathbb{R}^4$ cuya matriz coordenada respecto de las bases canónicas es

```
-->     a:matrix([1,2,3],[1,1,1],[0,1,2],[0,1,2]);
```

para pasar a la expresión coordenada de f

```
-->     f(x,y,z):=a.matrix([x],[y],[z])$
```

que podemos utilizar para obtener la imagen por f de los vectores de $\mathbb{R}^3$

```
-->     f(2,3,1);
```

Sea g la aplicación lineal de $\mathbb{R}^3$ en $\mathbb{R}^2$ dada por

$$g(x,y,z) = (x+2y, x+y+4z)$$

para obtener su matriz coordenada respecto de las bases canónicas, recordemos que la columna i-ésima de esta matriz está formada por las coordenadas, en la base canónica de $\mathbb{R}^2$, de la imagen por g del i-ésimo vector de la base canónica de $\mathbb{R}^3$. Para obtener la matriz, primero definimos g

```
-->     g(x,y,z):=[x+2*y,x+y+4*z];
```

y tenemos en cuenta cómo está formada esta matriz

```
-->     b:transpose(matrix(g(1,0,0),g(0,1,0),g(0,0,1)));
```

3.3.1 Ker f

Dada la aplicación lineal de E_n en F_m cuya matriz coordenada respecto de B_E y B_F es $A \in M_{m\times n}(\mathbb{R})$, la orden

```
--> nullity(a);
```

nos dice la dimensión de $Kerf$ y, si es no nula,

```
--> ker:nullspace(a);
```

nos facilita una base del subespacio. Esta base estará formada por los vectores de E_n cuyas coordenadas en B_E nos ha proporcionado la sentencia anterior. El resultado de esa sentencia es del tipo

$$\text{span}\left(\left[\begin{array}{c} -1 \\ 2 \\ 3 \end{array}\right]\right)$$

que, en este caso, nos diría que una base está formada por el vector de coordenadas $(-1, 2, 3)$ en B_E. Si queremos las columnas que nos determinan los vectores de la base podemos utilizar la orden

```
--> args(ker);
```

que es una lista de matrices columna, con lo que cada uno de los elementos de esta lista nos dará una matriz columna con las coordenadas de los vectores mencionados

```
--> args(ker)[1];
```

Ejercicio 5. Sea g la aplicación lineal introducida anteriormente. Hallar una base de $Kerg$ y decir su dimensión.

3.3.2 Im f

Del mismo modo que antes, pero para el subespacio Imf,

```
--> l:length(args(columnspace(a)));
```

nos proporciona la dimensión

```
--> columnspace(a); args(columnspace(a));
```

Ejercicio 6. Hallar una base de Imf, con f la aplicación lineal anterior y decir su dimensión. Lo mismo para la aplicación lineal g.

3.3.3 Cambio de base

Borrar los valores anteriormente asignados a las matrices a y b

```
-->     kill(a,b);
```

Sea $f \in \mathscr{L}_{I\!K}(E_n, F_m)$ la aplicación lineal cuya matriz coordenada respecto de las bases B_E y B_F es $A \in M_{m\times n}(I\!K)$. Si tomamos dos nuevas bases, B_E^* y B_F^* y las expresiones del cambio de base son $X = PX^*$ y $Y^* = QY$ entonces, la matriz coordenada de f respecto de B_E^* y B_F^* es $B = QAP$.

Ejercicio 7. Sea f la aplicación lineal de $I\!R^2$ en $I\!R^3$ cuya matriz coordenada respecto de

$$B_{R^2} = ((1,0),(0,1)) \text{ y } B_{R^3} = ((1,2,4),(3,4,6),(3,8,9))$$

es

$$\begin{pmatrix} 7 & -21/2 \\ 3 & -9/2 \\ -4 & 6 \end{pmatrix}$$

Probar que la matriz coordenada respecto de

$$B^*_{R^2} = ((1,1),(3,1)) \text{ y } B^*_{R^3} = ((1,1,1),(0,1,1),(0,0,1))$$

es

$$\begin{pmatrix} -2 & 6 \\ 5 & -15 \\ -8 & 24 \end{pmatrix}$$

Solución:
cambio de base en $I\!R^2$

```
-->     aa:matrix([7,-21/2],[3,-9/2],[-4,6])$

        v1:matrix([1,0],[0,1])$ v2:matrix([1,1],[3,1])$

        eq:a*v1[1] + b*v1[2]$

        col1:linsolve([eq[1]=v2[1,1],eq[2]=v2[1,2]],[a,b])$
        col2:linsolve([eq[1]=v2[2,1],eq[2]=v2[2,2]],[a,b])$
        col1:makelist(rhs(col1[i]),i,1,length(col1))$
        col2:makelist(rhs(col2[i]),i,1,length(col2))$
        mcE:transpose(matrix(col1,col2))$
```

cambio de base en $I\!R^3$

```
-->    u1:matrix([1,2,4],[3,4,6],[3,8,9])$
       u2:matrix([1,1,1],[0,1,1],[0,0,1])$

       eq:a*u2[1] + b*u2[2] + c*u2[3]$

       col1:linsolve([eq[1]=u1[1,1],eq[2]=u1[1,2],eq[3]=u1[1,3]],[a,b,c])$
       col2:linsolve([eq[1]=u1[2,1],eq[2]=u1[2,2],eq[3]=u1[2,3]],[a,b,c])$
       col3:linsolve([eq[1]=u1[3,1],eq[2]=u1[3,2],eq[3]=u1[3,3]],[a,b,c])$
       col1:makelist(rhs(col1[i]),i,1,length(col1))$
       col2:makelist(rhs(col2[i]),i,1,length(col2))$
       col3:makelist(rhs(col3[i]),i,1,length(col3))$
       mcF:transpose(matrix(col1,col2,col3))$
```

nueva matriz coordenada

```
-->    mcF.aa.mcE;
```