1.1 Espacios vectoriales lineales

7/11/2019/

1 - 2.1.7 Grupos,campos y espacios vectoriales

En este módulo utilizaremos algunas de las herramientas disponibles para incorporar el algebra discreta. Comenzaremos con la introducción a los conjuntos. Podemos operar con conjuntos pero primero debemos definirlos. Existen varias posibilidades, como mostramos a continuación:

```
(%i1) A:{1,2,3,4,5,6,7,8,9};
(A)   {1,2,3,4,5,6,7,8,9}

(%i2) B:set(1,3,5,7,9);
(B)   {1,3,5,7,9}

(%i3) C:set(2,4,6,8);
(C)   {2,4,6,8}

(%i4) D:makeset(i/j, [i,j], [[1,3], [2,3], [3,3], [4,3]]);
(D)   {1/3,2/3,1,4/3}
```

Notemos que es igual definir los conjuntos con llaves, con la función set o makeset. Podemos preguntar si determinado elemento perteneces, o no, a un conjunto.

```
(%i5) elementp(7,A);
(%o5) true

(%i6) elementp(7,C);
(%o6) false
```

Operaciones elementales con conjuntos

```
(%i7) UBC:union(B,C);
(UBC) {1,2,3,4,5,6,7,8,9}
```

```
(%i8)  is (A=UBC);
(%o8)  true

(%i9)  Cv:intersection(B,C);
(Cv)   {}
```

Para Maxima el conjunto vacío es {}.

```
(%i10) setdifference(A,C);
(%o10) {1,3,5,7,9}
```

Esto es, el conjunto con los elementos del conjunto A que no pertenecen al conjunto C.
Si queremos el conjunto de todos los subconjuntos del conjunto {a, b, c} le podemos pedir al programa que nos lo muestre:

```
(%i11) powerset({a,b,c});
(%o11) {{},{a},{a,b},{a,b,c},{a,c},{b},{b,c},{c}}
```

El producto cartesiano de los conjuntos A y B es el conjunto conformado por los pares (a, b):

$$A \times B = \{(a, b)/a \in A \ , \ b \in B\}.$$

```
(%i12) cartesian_product(B,C);
(%o12) {[1,2],[1,4],[1,6],[1,8],[3,2],[3,4],[3,6],[3,8],[
       5,2],[5,4],[5,6],[5,8],[7,2],[7,4],[7,6],[7,8],[9,2],[9,
       4],[9,6],[9,8]}
```

Le podemos pedir al programa la suma de los pares del producto cartesiano anterior:

```
(%i13) makeset(a+b, [a,b], cartesian_product(B,C));
(%o13) {3,5,7,9,11,13,15,17}
```

Maxima trata a los conjuntos y a las listas como objetos de distinta naturaleza, lo que permite trabajar con conjuntos cuyos elementos puedan ser también conjuntos o listas, es decir, subconjuntos.

```
(%i14) lista:makelist(i,i,1,30);
(lista) [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,
       21,22,23,24,25,26,27,28,29,30]
```

La lista anterior la convertiremos en un conjunto, para este fin debemos utilizar el comando "setify"

```
(%i15) E:setify(lista);
(E)    {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,
       21,22,23,24,25,26,27,28,29,30}
```

De este último conjunto podemos construir el subconjunto conformado por los números primos:

```
(%i16) Primos:subset(E,primep);
(Primos) {2,3,5,7,11,13,17,19,23,29}
```

Con "cardinality" podemos saber cuantos elementos contiene un conjunto:

```
(%i17) cardinality(%);
(%o17) 10
```

La función de Maxima que nos permite calcular las tablas que utilizamos en la sección 2.1 es la función "mod". Veamos:

```
(%i18) Gm8:{1,3,5,7};
(Gm8)  {1,3,5,7}
```

Todos los productos posibles entre los elementos de este conjunto son:

```
(%i19) makeset(a.b, [a,b],cartesian_product(Gm8,Gm8));
(%o19) {1,3,5,7,9,15,21,25,35,49}
```

Los modulo 8 para algunos de los números anteriores son:

```
(%i22) mod(21,8);mod(35,8);mod(49,8);
(%o20) 5
(%o21) 3
(%o22) 1
```

Para generar el grupo G[mod5] escribimos:

```
(%i23) setify(makelist(mod(i,5),i,1,4));
(%o23) {1,2,3,4}
```

En la sección 2.1.1, definimos el orden de un elemento g ∈ G. Consideremos el siguiente conjunto de números enteros:

Z15 = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14}.

Se podría demostrar que el orden de g ∈ G es igual al número de elementos de < g > y lo dejaremos como ejercicio.
Vamos a calcular el orden de todos los elementos de Z15, sabiendo que el orden de cada uno de esos elementos divide a 15, que es el cardinal de Z15.

Probemos primero con el número 6 ∈ Z15.

```
(%i24) setify(makelist(mod(6·i,15),i,0,14));
(%o24) {0,3,6,9,12}
```

El orden de 6 ∈ Z15 es:

```
(%i25) length(%);
(%o25) 5
```

En la siguiente instrucción lo haremos para todos los elementos de Z15.

```
(%i26) makelist([j,length(setify(makelist(mod(j·i,15),i,0,14)))],j,0,14);
(%o26) [[0,1],[1,15],[2,15],[3,5],[4,15],[5,3],[6,5],[7,15
],[8,15],[9,5],[10,3],[11,15],[12,5],[13,15],[14,15]]
```

La salida no es más que una lista [x, y] con cada elemento x ∈ Z15 y su orden y. Por lo tanto, el conjunto de órdenes es: {1, 3, 5, 15}, todos divisores de 15, como estipula el teorema de Lagrange.

```
(%i27) kill(all)$
```

FIN

2 - 2.2.5 Espacios métricos, normados y conproducto interno

1- Espacios y subespacios vectoriales:

Sea el espacio vectorial V = K^n, definido en K = R y donde n es un entero positivo. Consideremos el caso n = 4. El producto de un elemento de K^4, digamos |x⟩ = (x1,x2,x3,x4) por un escalar α ∈ K resulta en otro elemento de K^4.
Primero introducimos los elementos como listas:

```
(%i3)  X:[x1,x2,x3,x4];Y:[y1,y2,y3,y4];Z:[z1,z2,z3,z4];
(X)    [x1,x2,x3,x4]
(Y)    [y1,y2,y3,y4]
(Z)    [z1,z2,z3,z4]

(%i4)  α·X=Y;
(%o4)  [x1 α,x2 α,x3 α,x4 α]=[y1,y2,y3,y4]
```

El resultado es un elemento del espacio vectorial K^4. La suma de |x⟩ = (x1, x2, x3, x4) y |y⟩ = (y1, y2, y3, y4) serä:

```
(%i5)  X+Y=Z;
(%o5)  [y1+x1,y2+x2,y3+x3,y4+x4]=[z1,z2,z3,z4]
```

con (z1, z2, z3, z4) ∈ K^4. Podemos ver rápidamente que el conjunto de vectores que tienen la forma (x1, x2, x3, 0) conforman un subespacio de K^4, ya que:

```
(%i6)  map(":",[x4,y4,z4],[0,0,0]);
(%o6)  [0,0,0]

(%i9)  X:[x1,x2,x3,x4];Y:[y1,y2,y3,y4];Z:[z1,z2,z3,z4];
(X)    [x1,x2,x3,0]
(Y)    [y1,y2,y3,0]
(Z)    [z1,z2,z3,0]

(%i10) α·X+β·Y=Z;
(%o10) [y1 β+x1 α,y2 β+x2 α,y3 β+x3 α,0]=[z1,z2,z3,0]
```

Para recobrar las variables x4, y4, z4 escribimos:

```
(%i11) kill(x4,y4,z4)$
```

```
(%i14) X:[x1,x2,x3,x4];Y:[y1,y2,y3,y4];Z:[z1,z2,z3,z4];
(X)    [x1,x2,x3,x4]
(Y)    [y1,y2,y3,y4]
(Z)    [z1,z2,z3,z4]
```

Para calcular el producto interno entre vectores es necesario utilizar la librería "eigen".

```
(%i15) load("eigen")$
```

```
(%i16) innerproduct(X,Y);
(%o16) x4 y4+x3 y3+x2 y2+x1 y1
```

Consideremos ahora V = K^n, definido en K = C, con n = 3. Por lo tanto, los vectores serán ahora de la siguiente forma: z = (x1 + iy1, x2 + iy2, x3 + iy3).

```
(%i18) Z1:[x1+%i·y1,x2+%i·y2,x3+%i·y3]; Z2:[u1+%i·v1,u2+%i·v2,u3+%i·v3];
(Z1)   [i y1+x1,i y2+x2,i y3+x3]
(Z2)   [i v1+u1,i v2+u2,i v3+u3]
```

Y los escalares de la forma α = a + ib.

```
(%i19) α:a+%i·b;
(α)    i b+a
```

El producto por el escalar α es:

```
(%i20) Z3: α·Z1;
(Z3)   [(i b+a) (i y1+x1),(i b+a) (i y2+x2),(i b+a) (i y3+x3)]
```

```
(%i21) map(rectform,Z3);
(%o21) [i (a y1+b x1)−b y1+a x1,i (a y2+b x2)−b y2+a x2,i (a y3+b x3)
       −b y3+a x3]
```

```
(%i23) map(realpart,Z3),factor; map(imagpart,Z3),factor;
(%o22) [−(b y1−a x1),−(b y2−a x2),−(b y3−a x3)]
(%o23) [a y1+b x1,a y2+b x2,a y3+b x3]
```

Calculemos ahora el producto interno:

Z1 ·Z2 =(x1 +iy1)*(u1 +iv1)+(x2 +iy2)*(u2 +iv2)+(x3 +iy3)*(u3 +iv3)

```
(%i24) Z4:innerproduct(Z1,Z2);
```

(Z4) $(v3-i\,u3)\,y3+(v2-i\,u2)\,y2+(v1-i\,u1)\,y1+(i\,v3+u3)\,x3+(i\,v2+u2)\,x2+(i\,v1+u1)\,x1$

```
(%i26) Re:map(realpart,Z4)$ Im:map(imagpart,Z4)$
```

```
(%i27) Re+%i·Im;
```

(%o27) $i\,(-u3\,y3-u2\,y2-u1\,y1+v3\,x3+v2\,x2+v1\,x1)+v3\,y3+v2\,y2+v1\,y1+u3\,x3+u2\,x2+u1\,x1$

```
(%i28) kill(all)$
```

2- Producto de polinomios

Consideremos el producto escalar entre elementos de un espacio vectorial de polinomios.

Vamos a encontrar la distancia y el ángulo entre los vectores

|x1⟩ = x(x − 1) y |x2⟩ = x

en dos intervalos diferentes: [0, 1] y [−1, 1].

Debemos introducir los objetos a multiplicar

```
(%i2) P1:x·(x-1); P2:x;
```

(P1) $(x-1)\,x$

(P2) x

Ahora calculamos las distancias entre los vectores para ambos intervalos.

Haremos gala de algunas posibilidades que ofrece el programa para escribir las expresiones.

```
(%i3) sqrt('integrate(((P1-P2)^2),x,-1,1))=sqrt(integrate(((P1-P2)^2),x,-1,1));
```

(%o3) $\sqrt{\int_{-1}^{1} ((x-1)\,x-x)^2\,dx} = \frac{\sqrt{46}}{\sqrt{15}}$

(%i4) `sqrt('integrate(((P1-P2)^2),x,0,1))=sqrt(integrate(((P1-P2)^2),x,0,1));`

(%o4) $$\sqrt{\int_0^1 ((x-1)\,x-x)^2\,dx} = \frac{2^{3/2}}{\sqrt{15}}$$

El ángulo entre los polinomios definidos en el intervalo [-1, 1] es:

(%i5) `'integrate((P1·P2),x,-1,1)/(sqrt('integrate((P1·P1),x,-1,1))·sqrt('integrate((P2·P2),x,-1,1`

(%o5) $$\frac{\int_{-1}^{1} (x-1)\,x^2\,dx}{\sqrt{\int_{-1}^{1} x^2\,dx}\,\sqrt{\int_{-1}^{1} (x-1)^2\,x^2\,dx}}$$

(%i6) `ev(%,integrate);`

(%o6) $$-\frac{\sqrt{15}}{2^{3/2}\,\sqrt{3}}$$

(%i7) `acos(%),numer;`

(%o7) 2.482534617763384

Y ahora, el ángulo entre los polinomios definidos en el intervalo [0, 1]:

(%i8) `'integrate((P1·P2),x,0,1)/(sqrt('integrate((P1·P1),x,0,1))·sqrt('integrate((P2·P2),x,0,1)));`

(%o8) $$\frac{\int_{0}^{1} (x-1)\,x^2\,dx}{\sqrt{\int_{0}^{1} x^2\,dx}\,\sqrt{\int_{0}^{1} (x-1)^2\,x^2\,dx}}$$

(%i9) `ev(%,integrate);`

(%o9) $$-\frac{\sqrt{30}}{4\sqrt{3}}$$

```
(%i10) acos(%),numer;
(%o10) 2.482534617763385

(%i11) kill(all)$
```

FIN

3 - 2.3.7 Variedades lineales

1.- Independencia Lineal

En 2.3.1 vimos que si en la ecuación |0⟩ = C1 |v1⟩+C2 |v2⟩+C3 |v3⟩···+Cn |vn⟩ con todos los Ci = 0, entonces se dirá que el conjunto de vectores son linealmente independientes. Para el primer ejemplo de esa sección se obtuvo el siguiente sistema de ecuaciones:

$$\begin{aligned} C1 + 2C2 - C3 &= 0 \\ 3C1 + \quad C3 &= 0 \\ -C1 + C2 \quad &= 0 \\ 2C1 \;\; 3C2 \quad &= 0 \end{aligned}$$

que procedemos a resolver usando el comando "linsolve":

```
(%i1) linsolve([C1+2·C2-C3=0, 3·C1+C3=0, -C1+C2=0, 2·C1+3·C2=0], [C1,C2,C3,C4]);
solve: dependent equations eliminated: (4)
(%o1) [C1=0,C2=0,C3=0,C4=%r1]
```

2.- Bases para espacios vectoriales

En este ejercicio aprenderemos a calcular una base a partir de un conjunto de vectores perteneciente a un determinado espacio vectorial. Por ejemplo, si en R^5 tenemos el siguiente conjunto de vectores:

|v1⟩ = (1,2,3,4,5), |v2⟩ = (0,-1,1,2,3), |v3⟩ = (3,2,1,0,-1), |v4⟩ = (-4,-3,-2,-1,0).

```
(%i5) v1:[1,2,3,4,5]; v2:[0,-1,1,2,3]; v3:[3,2,1,0,-1]; v4:[-4,-3,-2,-1,0];
(v1) [1,2,3,4,5]
(v2) [0,-1,1,2,3]
(v3) [3,2,1,0,-1]
(v4) [-4,-3,-2,-1,0]
```

Con los vectores dados construimos la siguiente matriz

(%i6)
```
M:matrix(v1,v2,v3,v4);
```

(M)
$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 0 & -1 & 1 & 2 & 3 \\ 3 & 2 & 1 & 0 & -1 \\ -4 & -3 & -2 & -1 & 0 \end{pmatrix}$$

Como veremos más adelante, el Rango de una matriz indica el número máximo de vectores fila o columna linealmente independientes

(%i7)
```
rank(M);
```

(%o7) 3

Podemos aplicar el método de eliminación gaussiana a la matriz M para obtener una nueva matriz escalonada. El cálculo además se hace normalizando el primer elemento no nulo de cada fila. Esto se logra con el comando "echelon".

(%i8)
```
echelon(M);
```

(%o8)
$$\begin{pmatrix} 1 & \frac{2}{3} & \frac{1}{3} & 0 & -\frac{1}{3} \\ 0 & 1 & -1 & -2 & -3 \\ 0 & 0 & 1 & \frac{5}{3} & \frac{7}{3} \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Por lo tanto, cada fila de la matriz anterior conformará un conjunto de vectores linealmente independiente. Los podemos aislar de la matriz con el comando "row(M,i)", que devolverá la i-ésima fila de la matriz M.

(%i9)
```
e1:row(%o8,1);
```

(e1)
$$\begin{pmatrix} 1 & \frac{2}{3} & \frac{1}{3} & 0 & -\frac{1}{3} \end{pmatrix}$$

(%i10)
```
e2:row(%o8,2);
```

(e2)
$$\begin{pmatrix} 0 & 1 & -1 & -2 & -3 \end{pmatrix}$$

(%i11)
```
e3:row(%o8,3);
```

(e3)
$$\begin{pmatrix} 0 & 0 & 1 & \frac{5}{3} & \frac{7}{3} \end{pmatrix}$$

Note que usamos %o8 para usar la salida producida en (%i8)
Es trivial verificar que el conjunto: {e1, e2, e3} es linealmente independientes.

(%i12)
```
solve(α·[1,2/3,1/3,0,-1/3]+β·[0,1,-1,-2,-3]+γ·[0,0,0,5/3,7/3],[α,β,γ]);
```
solve: dependent equations eliminated: (3 4)
(%o12) *[[α=0,β=0,γ=0]]*

Consideremos los vectores a = (1, 3) y b = (−1, 1) ¿Serán linealmente independientes?

(%i13)
```
solve(α·[1,3]+β·[-1,1],[α,β]);
```
(%o13) *[[α=0,β=0]]*

Los vectores a = (1, 2, 3) y b = (4, 8, 12) ¿Serán linealmente independientes?

(%i14)
```
solve(α·[1,2,3]+β·[4,8,12],[α,β]);
```
solve: dependent equations eliminated: (2 3)
(%o14) *[[α=-4 %r2,β=%r2]]*

Aquí α = −4β.
Sea ahora {ei} = {(1,1,1),(1,2,1),(0,0,2)} una base para R^3. Vamos a calcular las componentes del vector a = (3, 2, 1) respecto de esa base.
Primero verifiquemos si efectivamente es una base:

(%i15)
```
solve(α·[1,1,1]+β·[1,2,1]+γ·[0,0,2],[α,β,γ]);
```
(%o15) *[[α=0,β=0,γ=0]]*

(%i16)
```
solve([3,2,1]-ax·[1,1,1]-ay·[1,2,1]-az·[0,0,2],[ax,ay,az]);
```
(%o16) *[[ax=4,ay=-1,az=-1]]*

Por lo tanto, a = 4e1 − e2 − e3.
En la base canónica las componentes del vector a serán:

(%i17)
```
solve([3,2,1]-ax·[1,0,0]-ay·[0,1,0]-az·[0,0,1],[ax,ay,az]);
```
(%o17) *[[ax=3,ay=2,az=1]]*

Es decir, a = 3i + 2j + k.

(%i18)
```
kill(all)$
```

3.- Ortogonalización con Maxima

Anteriormente hicimos los cálculos para hallar una base ortogonal a partir de un conjunto de vectores linealmente independientes.
|v1>= (1,3,-1,2), |v2>= (2,0,1,3), |v3>= (-1,1,0,0)

La función "gramschmidt(x)" es la que ejecuta el algoritmo de ortogonalización de Gram-Schmidt sobre el objeto x. Si x es una matriz el algoritmo actúa sobre sus filas. Antes de usar la función es necesario cargar la librería "eigen"

```
(%i1) load("eigen")$
```

```
(%i2) v:matrix ([-1, 1, 0, 0], [2, 0, 1,3], [1,3, -1,2]);
```

$$(v)\quad \begin{pmatrix} -1 & 1 & 0 & 0 \\ 2 & 0 & 1 & 3 \\ 1 & 3 & -1 & 2 \end{pmatrix}$$

Notemos que el vector |v3> es el que hemos puesto en la primera fila de la matriz. Ahora procedemos al cálculo:

```
(%i3) e:gramschmidt(v);
```

$$(e)\quad [[-1,1,0,0],[1,1,1,3],[\frac{5}{2},\frac{5}{2},-\frac{7}{2},-\frac{1}{2}]]$$

El resultado viene en forma de una lista: [e[1], e[2], e[3]]. Simplificando resulta:

```
(%i4) e:expand(%);
```

$$(e)\quad [[-1,1,0,0],[1,1,1,3],[\frac{5}{4},\frac{5}{4},-\frac{7}{4},-\frac{1}{4}]]$$

Podemos verificar que son ortogonales:

```
(%i5) map(innerproduct, [e[1], e[2], e[3]], [e[2], e[3], e[1]]);
(%o5) [0,0,0]
```

Normalizamos ahora cada uno de los vectores

(%i6)
```
e[1]/sqrt(innerproduct(e[1],e[1]));
```
(%o6) $$\left[-\frac{1}{\sqrt{2}},\frac{1}{\sqrt{2}},0,0\right]$$

(%i7)
```
e[2]/sqrt(innerproduct(e[2],e[2]));
```
(%o7) $$\left[\frac{1}{2\sqrt{3}},\frac{1}{2\sqrt{3}},\frac{1}{2\sqrt{3}},\frac{\sqrt{3}}{2}\right]$$

(%i8)
```
 e[3]/sqrt(innerproduct(e[3],e[3]));
```
(%o8) $$\left[\frac{1}{2},\frac{1}{2},-\frac{7}{10},-\frac{1}{10}\right]$$

La función "gramschmidt(x,F)" nos permite definir un producto interno diferente a innerproduct. Veamos como se hace para el otro ejemplo donde el conjunto de vectores linealmente independientes estaba dado por: {1,t,t^2,t^3,... ,t^n}.

(%i9)
```
producto(f,g):=integrate(f·g, t, a, b);
```
(%o9) producto$(f,g):=\int_a^b f\,g\,dt$

(%i10)
```
e:gramschmidt ([1, t, t^2,t^3,t^4], producto), a= -1, b=1;
```
(e) $$\left[1,t,\frac{3\,t^2-1}{3},\frac{t\,(5\,t^2-3)}{5},\frac{35\,t^4-30\,t^2+3}{35}\right]$$

(%i11)
```
e:expand(e);
```
(e) $$\left[1,t,t^2-\frac{1}{3},t^3-\frac{3\,t}{5},t^4-\frac{6\,t^2}{7}+\frac{3}{35}\right]$$

Verifiquemos si son ortogonales:

(%i12)
```
map(producto, [e[1], e[2], e[3]], [e[2], e[3], e[1]]), a= -1, b=1;
```
(%o12) $[0,0,0]$

Para normalizar hacemos lo siguiente:

(%i14)
```
a: -1$ b:1$
```

(%i15)
```
e[1]/sqrt(producto(e[1], e[1]));
```
(%o15) $\frac{1}{\sqrt{2}}$

(%i16)
```
e[2]/sqrt(producto(e[2], e[2]));
```
(%o16) $\frac{\sqrt{3}\,t}{\sqrt{2}}$

(%i17)
```
e[3]/sqrt(producto(e[3], e[3])),expand;
```
(%o17) $\frac{3\sqrt{5}\,t^2}{2^{3/2}}-\frac{\sqrt{5}}{2^{3/2}}$

(%i18)
```
e[4]/sqrt(producto(e[4], e[4])),expand;
```
(%o18) $\frac{5\sqrt{7}\,t^3}{2^{3/2}}-\frac{3\sqrt{7}\,t}{2^{3/2}}$

(%i19)
```
kill(all)$
```

FIN

4 - 2.4.6 Aproximación de funciones

1.- Series de Fourier

Existe una librería llamada "fourie" en Maxima que contiene instrucciones para el cálculo simbólico de series de Fourier de una función f(x) en el intervalo [-l,l]: "fourier (f,x,l)".
También encontraremos un conjunto de comandos en el paquete para calcular los coeficientes y para manipular las expresiones resultantes.

(%i1)
```
load(fourie)$
```

En el ejemplo anterior aproximamos la función: f(x) = x^2 .
Veamos como se trabaja con el programa para calcular la serie de Fourier. Los resultados aparecerán en la forma de listas temporales y entre ellas los coeficientes. Las listas temporales serán indicadas con la notación (%t).

(%i2)
```
f:x^2;
```

(f) x^2

(%i3)
```
fourier(f,x,%pi);
```

(%t3) $a_0=\frac{\pi^2}{3}$

(%t4) $a_n=\frac{2\left(\frac{\pi^2\sin(\pi n)}{n}-\frac{2\sin(\pi n)}{n^3}+\frac{2\pi\cos(\pi n)}{n^2}\right)}{\pi}$

(%t5) $b_n=0$

(%o5) *[%t3,%t4,%t5]*

Lo anterior se puede simplificar con el comando "foursimp":

(%i6)
```
foursimp(%);
```

(%t6) $a_0=\frac{\pi^2}{3}$

(%t7) $a_n=\frac{4(-1)^n}{n^2}$

(%t8) $b_n=0$

(%o8) *[%t6,%t7,%t8]*

Podemos evaluar la lista de los coeficiente hasta el término k. Aquí lo haremos hasta k = 4 y el resultado lo lo asignaremos a la variable F. Por otro lado, usaremos (%o8), la última salida, como entrada para siguiente comando.

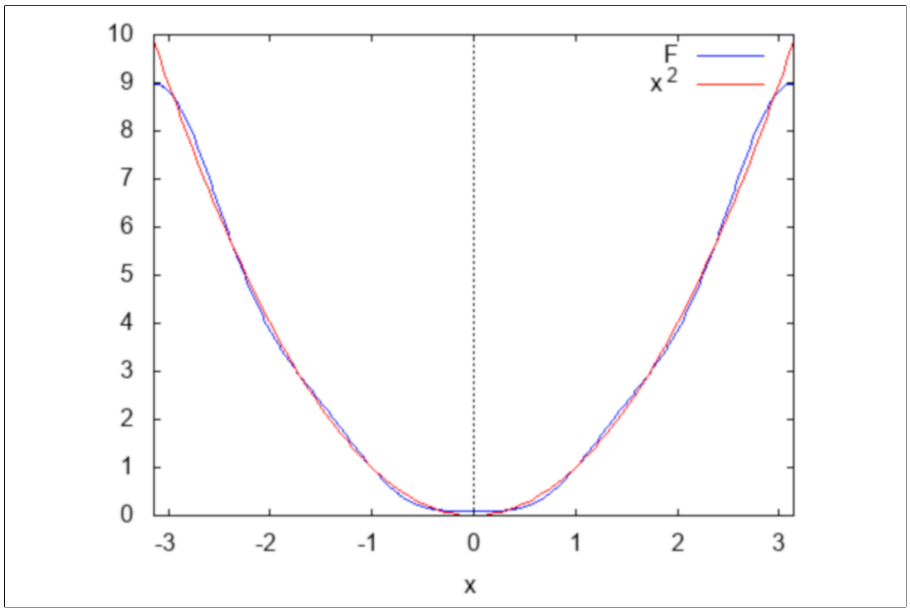
(%i9)
```
F:fourexpand(%o8,x,%pi,4);
```

(F) $\frac{\cos(4x)}{4}-\frac{4\cos(3x)}{9}+\cos(2x)-4\cos(x)+\frac{\pi^2}{3}$

Construiremos ahora en un mismo gráfico la función original y los primeros 5 términos de la serie, de esta manera podremos comparar el resultado de la aproximación. Las opciones para realizar los diferentes gráficos en Maxima se pueden consultar en el manual de programa.

(%i10)
```
wxplot2d([F,f], [x,-%pi,%pi],[legend, "F", "x^2"])$
```

(%t10)



Veamos que sucede si escribimos:

(%i11) `totalfourier(f,x,%pi);`

(%t11) $$a_0=\frac{\pi^2}{3}$$

(%t12) $$a_n=\frac{2\left(\frac{\pi^2\sin(\pi n)}{n}-\frac{2\sin(\pi n)}{n^3}+\frac{2\pi\cos(\pi n)}{n^2}\right)}{\pi}$$

(%t13) $$b_n=0$$

(%t14) $$a_0=\frac{\pi^2}{3}$$

(%t15) $$a_n=\frac{4(-1)^n}{n^2}$$

(%t16) $$b_n=0$$

(%o16) $$4\left(\sum_{n=1}^{\infty}\frac{(-1)^n\cos(n x)}{n^2}\right)+\frac{\pi^2}{3}$$

En este caso fueron aplicados de manera simultánea los comandos "fourier" y "foursimp" para finalmente presentar la serie en forma de una sumatoria.

(%i17) `kill(all)$`

2.- Mínimos Cuadrados

Maxima puede estimar los parámetros que mejor se ajusten a una función f=(x,y) para un conjunto de datos, utilizando el método de mínimos cuadrados. El programa buscará primero una solución exacta, si no la encuentra buscará una aproximada. El resultado lo presentará como una lista de ecuaciones. La función a utilizar será "lsquares".
Vamos a considerar los ejemplos estudiados con anterioridad:

1. En el primer ejemplo los datos eran los siguientes:

$$(x,y) = (1,2),(3,2),(4,5),(6,6)$$

Es necesario hacer uso de la libreríaa "lsquares" y los los datos deben introducirse en forma de matriz

```
(%i1) load(lsquares)$
```

```
(%i2) datos: matrix([1,2],[3,2],[4,5],[6,6])$
```

Por conveniencia, para más adelante hacer un gráfico, convertimos la matrix datos en una lista. Esto es sencillo si utilizamos el comando "args":

```
(%i3) datosL: args(datos);
(datosL) [[1,2],[3,2],[4,5],[6,6]]
```

Supondremos entonces que los puntos se ajustan a un polinomio lineal del tipo:

y = ax. El parámetro a se calcula con la función "lsquares estimates". Es importante prestar bastante atención a la sintaxis del siguiente comando.

```
(%i4) param: lsquares_estimates(datos,[x,y],y=a·x,[a]), numer;
(param) [a=1.032258064516129]
```

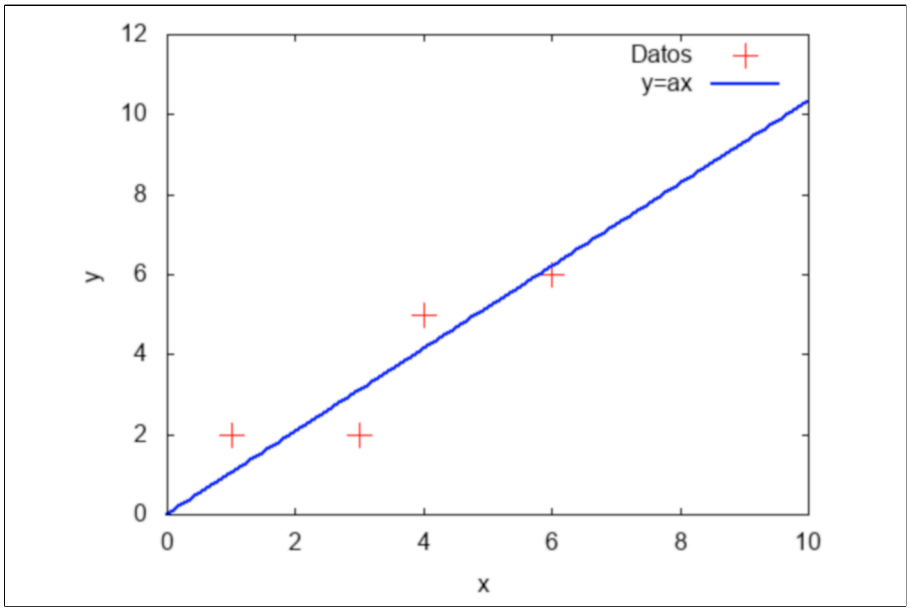
Este será entonces el valor del parámetro a de la ecuación de la recta y = ax que pasa por el origen. Notemos también que le hemos asignado el valor del parámetro a a la variable param. Lo que haremos ahora es escribir la ecuación de dicha recta. Podemos hacer uso de la instrucción "ev" que nos permite evaluar una expresión.

```
(%i5) y:ev(a·x,first(param));
(y) 1.032258064516129 x
```

Procederemos ahora a graficar los datos experimentales vs el ajuste por mínimos cuadrados en un mismo gráfico. Recordemos que el conjunto de puntos lo tenemos en la forma de una lista, que hemos denominada más arriba como datosL. Mientras que al ajuste que hemos calculado, es decir, la recta: 1,032258064516129x le hemos asignado la variable denominada y. Es recomendable consultar el manual del programa, en la parte que tiene que ver con gráficos, plot2d, plot3d, para identificar la sintaxis que aparecerá en la siguiente instrucción.

(%i6)
```
wxplot2d([[discrete,datosL], y], [x,0,10],[style, [points,5,2], [lines,2,1]],[point_type, pl
```

(%t6)



Nota: Se deja como ejercicio repetir éste mismo cálculo pero usando un ajuste para los datos de la forma: y = ax + b.

(%i7)
```
kill(all)$
```

2. Consideremos el conjunto de datos:
|x1⟩ = (1, 2, 1, 1) , |x2⟩ = (2, 1, 1, −1) , |y⟩ = (15, 12, 10, 0).
Vamos a suponer que ajustan de la manera siguiente: |y⟩ = a |x1⟩ + b |x2⟩.

(%i1)
```
datos2: matrix([1,2,15],[2,1,12],[1,1,10],[1,-1,0])$
```

Cambiemos ligeramente la notación por: z = ax + by y calculemos los parámetros a y b.

(%i2)
```
param: lsquares_estimates(datos2,[x,y,z], z=a·x+b·y,[a,b]), numer;
```

(param) $[[a=4.090909090909091, b=5.090909090909091]]$

3. Para el tercer ejemplo se consideraron los siguientes datos:
{(0,1),(1,3),(2,7),(3,15)} ⇔ y=ax^2 +bx+c.

Haremos con Maxima el cálculo directo usando un ajuste cuadrático para los datos suministrados.

(%i3)
```
datos3: matrix([0,1],[1,3],[2,7],[3,15])$
```

(%i4)
```
datosL3: args(datos3)$
```

(%i5)
```
param: lsquares_estimates(datos3,[x,y], y=a·x^2+b·x+c,[a,b,c]), numer;
```

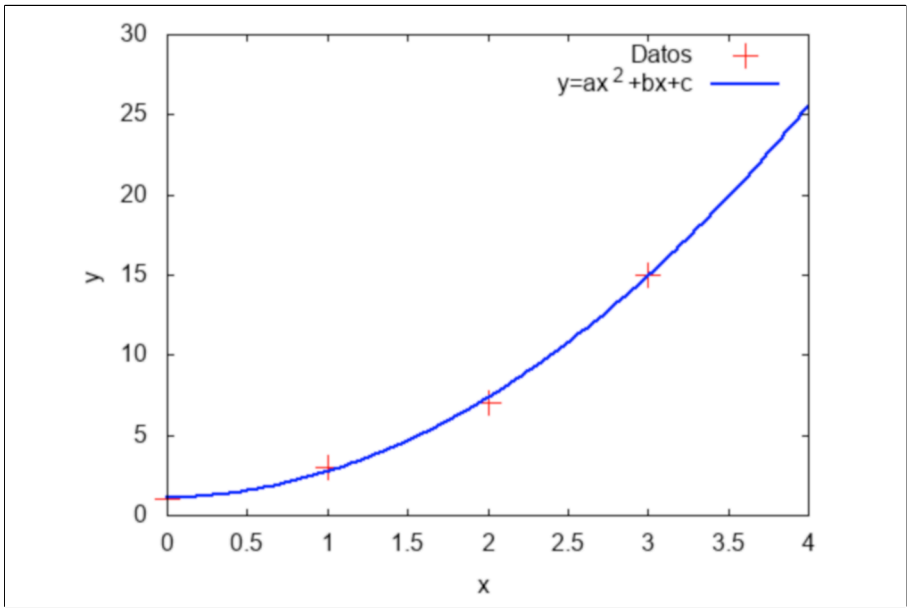
(param) $[[a=1.5, b=0.1, c=1.1]]$

(%i6)
```
y2:ev(a·x^2+b·x+c,first(param))$
```

Como hicimos anteriormente, graficamos los datos y el ajuste cuadrático en una misma figura.

(%i7)
```
wxplot2d([[discrete,datosL3], y2], [x,0,4],[style, [points,5,2], [lines,2,1]],[point_type, p
```

(%t7)



(%i8)
```
kill(all)$
```

3.- Polinomios ortogonales

Maxima contiene la librería "orthopoly" que nos permite acceder a la evaluación simbólica y numérica de los diferentes tipos de polinomios ortogonales: Chebyshev, Laguerre, Hermite, Jacobi, Legendre...

(%i1)
```
load (orthopoly)$
```

Por ejemplo, para obtener los primeros 6 polinomios de Legendre escribimos los siguientes comandos:

(%i2)
```
[legendre_p(0,x),legendre_p(1,x),legendre_p(2,x),legendre_p(3,x),legendre_p(4,x),legendre_p(
```

Simplificamos con "ratsimp":

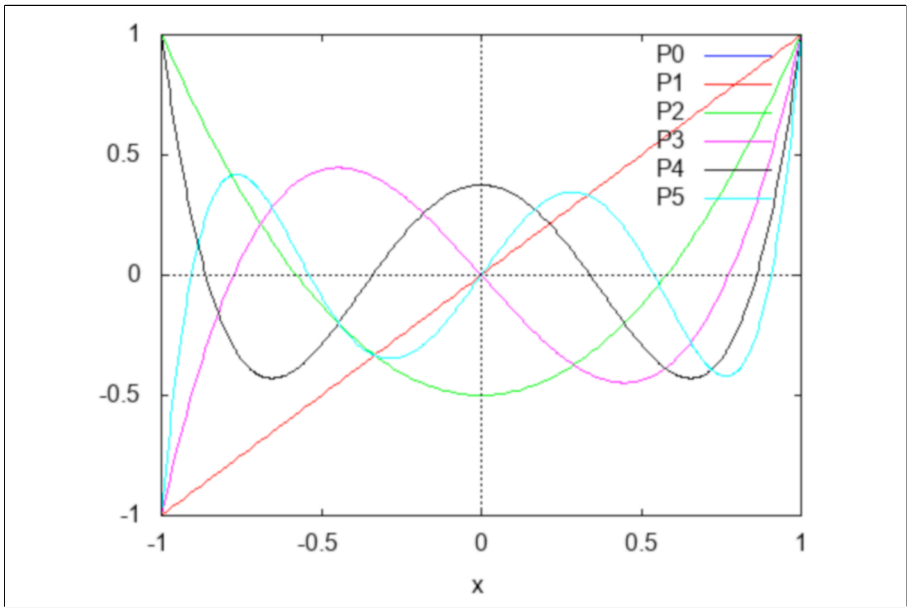
(%i3)
```
ratsimp (%);
```

(%o3)
$$\left[1,x,\frac{3x^2-1}{2},\frac{5x^3-3x}{2},\frac{35x^4-30x^2+3}{8},\frac{63x^5-70x^3+15x}{8}\right]$$

Los diferentes polinomios de Legendre se pueden visualizar de la manera siguiente:

(%i4)
```
wxplot2d(%,[x,-1,1],[legend, "P0", "P1","P2", "P3","P4", "P5" ])$
```

(%t4)



Ahora bien, con los datos de la figura 2.3 se planteó un sistema de ecuaciones lineales:

(%i5)
```
ecu1:C0-C1+C2-C3+C4-C5=8$
```

(%i6)
```
ecu2:C0-3/5·C1+1/25·C2+9/25·C3-51/125·C4+477/3125·C5=10$
```

(%i7)
```
ecu3:C0-1/5·C1-11/25·C2+7/25·C3+29/125·C4-961/3125·C5=11$
```

(%i8) `ecu4:C0+1/5·C1−11/25·C2−7/25·C3+29/125·C4+961/3125·C5=18$`

(%i9) `ecu5:C0+3/5·C1+1/25·C2−9/25·C3−51/125·C4−477/3125·C5=20$`

(%i10) `ecu6:C0+C1+C2+C3+C4+C5=34$`

Resolvemos el sistema anterior:

(%i11) `linsolve([ecu1,ecu2,ecu3,ecu4,ecu5,ecu6], [C0,C1,C2,C3,C4,C5]);`

(%o11) $[C0=\frac{2249}{144}, C1=\frac{3043}{336}, C2=\frac{1775}{504}, C3=-\frac{175}{216}, C4=\frac{625}{336}, C5=\frac{14375}{3024}]$

Para asignar cada solución a la variable correspondiente podemos hacer lo siguiente:

(%i12) `[C0,C1,C2,C3,C4,C5]:[rhs(%[1]),rhs(%[2]),rhs(%[3]),rhs(%[4]),rhs(%[5]),rhs(%[6])];`

(%o12) $[\frac{2249}{144}, \frac{3043}{336}, \frac{1775}{504}, -\frac{175}{216}, \frac{625}{336}, \frac{14375}{3024}]$

Por lo tanto, la función aproximada será:

(%i13) `f:C0+C1·legendre_p(1,x)+C2·legendre_p(2,x)+C3·legendre_p(3,x)+C4·legendre_p(4,x)+legendre_p(`

(%i14) `f:expand(%);`

(f) $\frac{14375\,x^5}{384}+\frac{3125\,x^4}{384}-\frac{8375\,x^3}{192}-\frac{325\,x^2}{192}+\frac{7367\,x}{384}+\frac{1863}{128}$

Procedemos a introducir los datos:

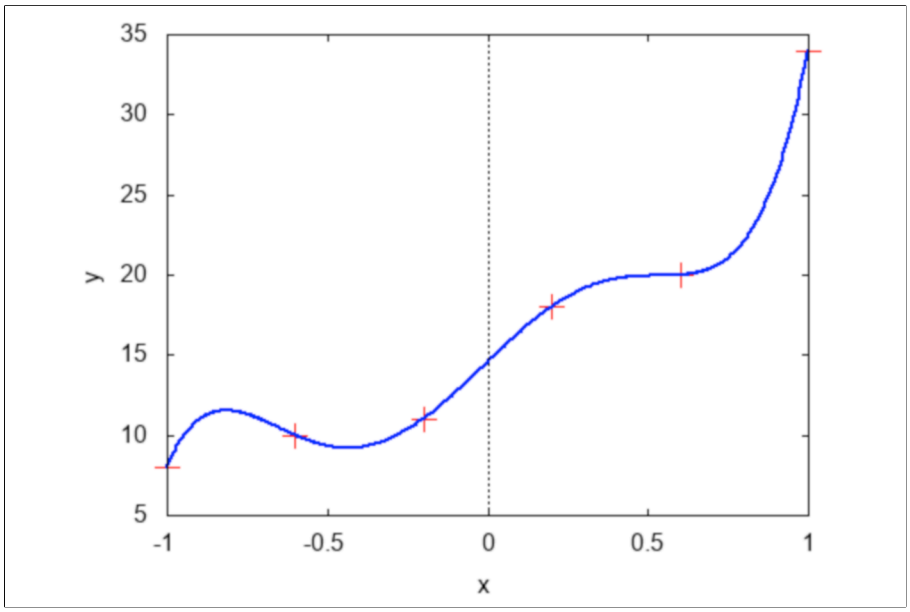
(%i15) `datos:[[-1,8],[-3/5,10],[-1/5,11],[1/5,18],[3/5,20],[1,34]];`

(datos) $[[-1,8],[-\frac{3}{5},10],[-\frac{1}{5},11],[\frac{1}{5},18],[\frac{3}{5},20],[1,34]]$

Para finalizar, haremos la gráfica con los datos y con la interpolación:

(%i16)
```
wxplot2d([[discrete,datos],f], [x,-1,1],[style, [points,5,2], [lines,2,1]],[point_type, plus
```

(%t16)



(%i17)
```
kill(all)$
```

FIN