



## **HERRAMIENTAS INFORMÁTICAS PARA MATEMÁTICAS**

(Apuntes elaborados por el equipo docente para el curso 2020-21)

### **Tema 2**

## **ELEMENTOS BÁSICOS DE LOS LENGUAJES MATEMÁTICOS**

En este tema se presentan los elementos básicos que se emplearán en las herramientas matemáticas Scilab y Máxima para la solución de problemas. Estos elementos básicos se usarán tanto en la solución de los problemas matemáticos complejos como en los más sencillos. Por esta razón, el dominio de la sintaxis de estos elementos será un paso fundamental en el buen aprendizaje de estas herramientas.

El tema está dividido en los siguientes apartados:

- Operadores, variables y expresiones
- Estructuras de datos y matrices
- Operaciones y funciones elementales
- Representaciones gráficas

### **2.1 OPERADORES, VARIABLES Y EXPRESIONES**

En este apartado se mostrarán los operadores matemáticos básicos como son la multiplicación o la suma. Se mostrará como se definen variables y los distintos tipos de datos a emplear, que pueden ser desde números enteros a cadenas de caracteres, o variables simbólicas. Se dejará para desarrollar en apartados posteriores el uso y las operaciones de matrices y otras estructuras de datos como son las listas.

#### **2.1.1 Operadores y expresiones**

Tanto Scilab como Maxima permiten realizar distintos tipos de operaciones. Se podrá clasificar las operaciones matemáticas en tres grandes grupos.

- Operaciones aritméticas
- Operaciones lógicas
- Operaciones de comparación

### Operaciones aritméticas

Las operaciones aritméticas se realizan con operandos numéricos, ya sean entre valores numéricos definidos y/o variables simbólicas. Las operaciones aritméticas básicas empleadas en Scilab y en Maxima se muestran en la siguiente tabla:

Operación	Scilab y Maxima
Suma	+
Resta	-
Multiplicación	*
División	/
Potenciación	^ o **

**Tabla 2.1** Operaciones aritméticas básicas.

En primer lugar se muestra un ejemplo de Maxima realizando operaciones aritméticas simbólicas mezcladas con valores numéricos.

```
(%i8) 7*b+a;
(%o8) 7 b+a

(%i9) (%-3*b)/c;
(%o9)  $\frac{4 b+a}{c}$ 

(%i10) (%)^0.5;
(%o10)  $\left(\frac{4 b+a}{c}\right)^{0.5}$ 
```

**Código 2.1** Ejemplo de operaciones aritméticas en Maxima mezclando valores numéricos con variables simbólicas.

A continuación se muestra un ejemplo de como realizan las mismas operaciones numéricas las dos herramientas.

Scilab	Maxima
-->5+7 ans =  12.	(%i1) 5+7; (%o1) 12
-->ans-6 ans =  6.	(%i2) %-6; (%o2) 6
-->ans/1.5 ans =  4.	(%i3) %/1.5; (%o3) 4.0
-->ans*7;   -->ans^0.5 ans =  5.2915026	(%i4) %*7\$   (%i5) %^0.5; (%o5) 5.291502622129181

**Tabla 2.2** Ejemplo de operaciones aritméticas.

A pesar del parecido en la sintaxis de las operaciones, se observan algunas diferencias. La llamada al valor de la última evaluación en Scilab se realiza empleando `ans`. Maxima emplea el símbolo `%`. Otra diferencia se refiere a la manera para no mostrar la salida de una evaluación. Scilab emplea el caracter `;` y Maxima emplea el caracter `$`.

### Operaciones lógicas

Las operaciones lógicas tienen como operandos elementos booleanos y dan como resultado un valor booleano (verdadero o falso). Este tipo de datos se analizará en los siguientes apartados junto con el resto de tipos.

Operación	Scilab	Maxima
y	<code>&amp;</code>	<code>and</code>
o	<code> </code>	<code>or</code>
no	<code>~</code>	<code>not</code>

**Tabla 2.3** Operaciones lógicas básicas.

Operadores lógicos más complejos pueden ser contruidos a partir de combinaciones de los operadores `and`, `or` y `not`.

### Operaciones relacionales

Los operadores relacionales permiten comparar operandos numéricos dando como resultado un valor booleano. Este tipo de operaciones son especialmente interesantes para el control de flujo de programas. También tienen gran interés para el control de errores.

Operación	Scilab	Maxima
igualdad	<code>==</code>	<code>=</code>
desigualdad	<code>~=</code> ó <code>&lt;&gt;</code>	<code>#</code>
menor	<code>&lt;</code>	<code>&lt;</code>
menor o igual	<code>&lt;=</code>	<code>&lt;=</code>
mayor	<code>&gt;</code>	<code>&gt;</code>
mayor o igual	<code>&gt;=</code>	<code>&gt;=</code>

**Tabla 2.4** Operaciones relaciones.

Existe una diferencia importante en el uso de estos operadores en las dos herramientas. Scilab realiza la sustitución numérica de los valores numéricos y comprueba si cumple el operador. Los operadores relacionales empleados en Maxima deben ser evaluados empleando las funciones `is()` o `maybe()`.

Es posible realizar operaciones relacionales entre variables simbólicas con Maxima. A continuación se muestra un sencillo ejemplo que compara la igualdad o desigualdad de una misma variable.

```
(%i12) is(a=a);
(%o12) true

(%i15) is(a#a);
(%o15) false
```

**Código 2.2** Comparación de variable simbólica en Maxima.

A continuación se muestra un ejemplo del empleo de algunos operadores relacionales. Se puede observar en el ejemplo que las expresiones de entrada %i13 y %i16 no evalúan la igualdad.

Scilab	Maxima
-->12==4 ans =	<pre>(%i13) 12=4; (%o13) 12=4</pre>
F	<pre>(%i14) is(12=4); (%o14) false</pre>
-->12.5>4 ans =	<pre>(%i15) is(12.5&gt;4); (%o15) true</pre>
T	<pre>(%i16) 4#2; (%o16) 4#2</pre>
-->4<>2 ans =	<pre>(%i17) maybe(4#2); (%o17) true</pre>
T	

**Tabla 2.5** Ejemplo de operaciones relacionales.

Existen otros muchos operadores que los mostrados anteriormente para ambas herramientas. Se recomienda consultar el apartado *operadores* de la ayuda de Maxima y el capítulo 3 del manual de Baudin para Scilab.

En este apartado se han visto los operadores para elementos unitarios. En apartados posteriores veremos operadores para matrices.

### 2.1.2 Variables

En el apartado anterior se han empleado las herramientas matemáticas como calculadoras. Tanto Scilab como Maxima permiten emplear variables para almacenar datos o bien emplear estas variables para realizar cálculos simbólicos.

Las variables pueden ser nombradas con una combinación de los caracteres ASCII, siempre teniendo cuidado de no utilizar como nombre de variable un término reservado. La asignación de un valor a una variable se muestra en el siguiente ejemplo.

Scilab	Maxima
-->x=17 x =	<pre>(%i1) x:17; (%o1) 17</pre>
17.	<pre>(%i2) x; (%o2) 17</pre>
-->a=%T a =	<pre>(%i3) a:true; (%o3) true</pre>
T	<pre>(%i4) a; (%o4) true</pre>

**Tabla 2.6** Ejemplo de asignación de valor a variables.

### Constantes

A diferencia de las variables, las constantes almacenan valores que no cambian. Habitualmente expresan constantes matemáticas conocidas. Tanto en Scilab como en Maxima se emplea en muchos casos % como prefijo para algunos nombres reservados para representar constantes. El nombre de estas constantes no deben ser empleadas como nombre para variables. En la siguiente tabla se muestran algunas de las constantes más importantes.

Constante	Scilab	Maxima
$\pi$	%pi	%pi
Constante de Euler	%e	%e
Número imaginario	%i	%i
Booleano verdadero	%T ó %t	true ó True
Booleano falso	%F ó %f	false ó False
Infinito positivo	-	inf

**Tabla 2.7** Algunas constantes en Maxima y Scilab.

### 2.1.3 Tipos de datos

Las herramientas matemáticas deben ser capaces de manipular distintos tipos de números como son los enteros, los de coma flotante de la precisión requerida, o números de tipo complejo, pero además cadenas de caracteres, tipos booleanos, o en el caso del cálculo simbólico expresiones polinómicas. Tanto Scilab como Maxima permiten el empleo de gran variedad de tipos, permitiendo una gran versatilidad en el desarrollo de programas de alto nivel.

#### Tipos numéricos

En Scilab las variables numéricas por defecto se definen como un tipo de coma flotante de doble precisión. Los tipos numéricos permiten la conversión entre sus distintos tipos, por ejemplo entre coma flotante y enteros de la precisión deseada. A continuación se puede ver un ejemplo de la conversión de tipos numéricos en Scilab.

```
-->a=12.23
a =

12.23

-->type(a)
ans =

1.

-->a=int8(a)
a =

12

-->type(a)
ans =

8.

-->a=complex(2,4)
a =

2. + 4.i

-->type(a)
ans =

1.
```

**Código 2.3** Conversión de tipos con Scilab.

Tipo	Código <code>type ( )</code>
Real o complejo	1
Polinomio	2
Booleano	4
Enteros de cualquier precisión	8
Cadena de caracteres	10
Lista	15

**Tabla 2.8** Código `type ( )` de tipos en Scilab.

En el ejemplo mostrado en el código 2.3 se puede ver como al asignarle un valor de tipo `real` a una variable en Scilab se convierte en una variable de tipo `real`. Posteriormente, se convierte esta variable de tipo `real` a tipo entero de 8 bits con signo. Al consultar de nuevo a la función `type ( )`, devuelve el código correspondiente. Ahora se le asigna un valor complejo, donde el primer argumento de `complex ( )` es la parte real y el segundo la parte imaginaria. Algunos de los códigos que devuelve la función `type ( )` se muestran en la tabla adjunta.

En Máxima existen distintos tipos numéricos:

- *Enteros* (73647).
- *Racionales*, representados por el cociente de números enteros ( $\frac{7}{9}$ ).

- *Coma flotante o decimales y decimales de doble precisión* (1.76235464).
- *Números complejos*, donde los coeficientes del número complejo deben ser del tipo decimal ( $6.23 + 3.56 * i$ ).

En el siguiente ejemplo se puede observar como se define la variable  $c$  como tipo racional, pudiendo evaluar su valor decimal (`float`).

```
(%i1) a:5;
(%o1) 5

(%i2) b:4;
(%o2) 4

(%i3) c:a/b;
(%o3)  $\frac{5}{4}$ 

(%i4) float(c);
(%o4) 1.25
```

**Código 2.4** Conversión de tipos en Maxima

En el siguiente ejemplo se puede observar como la representación de los números decimales y enteros en formato decimal, llamada notación científica, donde las potencias de 10 se expresa en Maxima con el carácter `b`. En Scilab se emplea el carácter `D`. La precisión de la salida del número decimal en Scilab se define con la expresión `format()`, donde los parámetros de la expresión son el tipo de formato. En nuestro ejemplo, el formato científico es ``e`` seguido por el número de total de dígitos. En Maxima la precisión del tipo decimal se define empleando la expresión `fpprec`.

Scilab	Maxima
<pre>--&gt;x=1/3 x =  0.3333333  --&gt;format(20)  --&gt;x x =  0.33333333333333331  --&gt;format('e',25)  --&gt;x x =  3.333333333333333148D-01</pre>	<pre>(%i29) fpprec:20\$ bfloat(1/3); (%o30) 3.3333333333333331483b-1</pre>

**Tabla 2.9** Precisión numérica.

### Otros tipos

Además de los tipos numéricos, las herramientas matemáticas pueden manipular otros tipos de datos. En Maxima, el tipo cadena de caracteres se define como una secuencia de caracteres entre comillas "cadena". En Scilab, la definición de una cadena de caracteres se realiza de manera similar a Maxima, pero en este caso empleando comillas simples 'cadena'.

Scilab	Maxima
<pre>--&gt;frase='Esto es una frase.' frase = Esto es una frase.</pre>	<pre>(%i31) frase:"Esto es una frase."; (%o31) Esto es una frase.</pre>

**Tabla 2.10** Definición de cadenas de caracteres.

Scilab permite la definición de tipo polinómico, dirigido a la manipulación simbólica. Sin embargo, debido a la potencia de Maxima para el cálculo simbólico, no se podrá de relieve las capacidades de cálculo simbólico que permite Scilab.

## **2.2 ESTRUCTURA DE DATOS Y MATRICES**

Las variables permiten almacenar, tanto en Scilab como en Maxima más de un dato y de distintos tipos en una misma variable. Para almacenar varios datos de un solo tipo se van a emplear las matrices. Para almacenar datos de distinto tipo, se pueden emplear otros tipos, como son las listas o estructuras, entre otros.

### **2.2.1 Estructuras de datos**

Hay varios tipos de estructuras de datos que se deben contemplar, los arrays, las listas y las estructuras. Tanto Scilab como Maxima tienen sus tipos equivalentes, ya que estos conceptos vienen heredados de lenguajes de programación de más bajo nivel.

#### Arrays

Tanto en Maxima como Scilab se pueden emplear arrays. Este tipo de estructura de datos permite almacenar datos de distintos tipos. En Scilab se llaman `cell array`, y en Maxima simplemente `array`. Un array en Scilab es un caso particular de lista. Sin embargo, en Maxima, el acceso a los datos del array es más rápido que si los datos hubieran sido almacenados en listas.

Se muestra a continuación algunos ejemplos de cómo declarar un array, y acceder y definir los valores de un array.



```

[ (%i1) array(un_array_en_Maxima,3,2);
  (%o1) un_array_en_Maxima

[ (%i2) un_array_en_Maxima[0,0]:"Hola"$

[ (%i3) un_array_en_Maxima[0,1]:37.23$

[ (%i5) un_array_en_Maxima[2,0]:x+y$

[ (%i6) un_array_en_Maxima[2,0];
  (%o6) y+x

[ (%i7) listarray(un_array_en_Maxima);
  (%o7) [Hola, 37.23, #####, #####, #####, #####, y+x, #####, #####, #####, #####, #####]

```

### Código 2.5 Declaración de un array con elementos de distintos tipos en Maxima.

En este primer ejemplo se observa como se puede declarar un array en Maxima empleando la función `array()`. Esta función tiene como argumentos, el nombre que tendrá el array, y los siguientes definen la dimensión del array. Se puede observar que los índices del array comienzan por 0, por lo que el array del ejemplo, `un_array_en_Maxima`, tendrá una dimensión de 4x3. En las expresiones de entrada 2, 3 y 4 se introducen los valores de tres términos del array empleando la sintaxis de []. Se puede observar que los tipos empleados son distintos. En la expresión de entrada 6 se solicita la salida de un valor específico. Finalmente se podrán solicitar todos los valores del array almacenados empleando la función `listarray()`.

```

--> un_array_en_scilab=cell(3,2)
un_array_en_scilab =

      [0x0 constant]  [0x0 constant]
      [0x0 constant]  [0x0 constant]
      [0x0 constant]  [0x0 constant]

--> un_array_en_scilab{1,1}='hola';

--> un_array_en_scilab{2,1}=37.23;

--> un_array_en_scilab{3,2}=%T;

--> un_array_en_scilab{3,2}
ans =

      T

--> un_array_en_scilab
un_array_en_scilab =

      [1x1 string ]  [0x0 constant]
      [1x1 constant] [0x0 constant]
      [0x0 constant] [1x1 boolean ]

```

### Código 2.6 Declaración de un array con elementos de distintos tipos en Scilab.

En este ejemplo para Scilab, se realizan las mismas operaciones que en el ejemplo anterior con Maxima. Se define un array empleando la función `cell()`, donde los

argumentos son las dimensiones del array. Los índices del array comienzan por 1, por lo que la dimensión del array definido es de 3x2.

Para introducir datos en el array se debe emplear la sintaxis siguiente,

`Nombre_el_array{dim a, dim b, ...} = valor asignado.`

El acceso al contenido del array se realiza empleando la siguiente sintaxis,

`Nombre_el_array{dim a, dim b, ...}`

Se puede acceder a los tipos de los elementos almacenados en el array llamando simplemente al nombre del array.

Si se desea acceder a todo el contenido del array, se puede emplear la siguiente sintaxis,

`Nombre_el_array{: , : , ...}.`

### Estructuras

Las estructuras son contenedores que almacenan la información de forma estructurada.

La información de las estructuras se almacena en campos que describen las características de los datos que se almacenan en su nombre. Esta es la razón por la que en Scilab el título de los campos deben ser cadenas de caracteres. Se verá a continuación un ejemplo de una estructura que almacena información sobre satélites del sistema solar.

```
-->satelite1=struct('Nombre','Ganimedes',...
-->'Masa',1.482d23,...
-->'Diametro', 5262.4d3,...
-->'Temperatura',113,...
-->'T_rot',[7 3 42.6])
satelite1 =

Nombre: "Ganimedes"
Masa: 1.482D+23
Diametro: 5262400
Temperatura: 113
T_rot: [7,3,42.6]
```

#### **Código 2.7** Declaración de una estructura de datos en Scilab.

En este ejemplo se puede ver como se crea una estructura mediante la función `struct()`, donde se introducen los nombres de los campos y la asignación de su valor. Se puede acceder al contenido de cualquier campo de la estructura empleando la sintaxis, `Nombre_estructura.campo`.

**Ejemplo 2.1** *Cálculo del número de minutos de rotación y de la estimación de la densidad del satélite Ganimedes accediendo a los datos de la estructura del código 2.7.*

Se accede a los datos del vector del campo del periodo de rotación, que tiene un formato de [días horas minutos], para calcular el número de minutos del periodo de rotación del satélite. Seguidamente, se hace una estimación de la densidad, a partir de la masa y del diámetro del satélite en [Kg/m<sup>3</sup>].

```
-->Minutos_rot=satelite1.T_rot(1)*24*60+..
-->satelite1.T_rot(2)*60+..
-->satelite1.T_rot(3)
Minutos_rot =

10302.6

-->Densidad_sat1=satelite1.Masa/((4/3)*%pi*(satelite1.Diametro/2)^3)
Densidad_sat1 =

1942.2182
```

### **Código 2.8** Operaciones empleando campos de una estructura en Scilab.

#### Listas

Este tipo de estructura de datos dinámica es considerado en Maxima y Scilab. En Maxima, todas las estructuras de datos son listas, ya que el lenguaje Lisp que utiliza como lenguaje de base, está orientado a su manipulación. Comúnmente, este tipo de contenedor ofrece numerosas herramientas para manipular su contenido, como acceder a todos o partes de los datos u ordenarlos.

En Scilab se consideran tres tipos de listas, las listas ordinarias `list()`, las listas de tipo typed, `tlist()`, y las listas de tipo typed orientadas a matrices, `mlist()`. La primera, la lista ordinaria tiene propiedades muy parecidas a los arrays. Las listas de tipo typed, son muy útiles para estructurar información. Se verá en el siguiente ejemplo el uso de una lista de tipo typed, donde se almacenan algunas propiedades de elementos químicos.

```
-->elementos=tlist(['Lista_elementos','Campos','Plutonio','Radio'],..
-->['Masa_Atm','Radio','P_fusion','Densidad'],[244,175,912.5,19816],..
-->[266.02,215,973,5000])
elementos =

elementos(1)

!Lista_elementos Campos Plutonio Radio !

elementos(2)

!Masa_Atm Radio P_fusion Densidad !

elementos(3)

244.    175.    912.5    19816.

elementos(4)

266.02    215.    973.    5000.

-->disp(elementos.Campos(2)+':'+string(elementos.Plutonio(2)))

Radio:175
```

### **Código 2.9** Declaración de una lista en Scilab.

Se puede observar que en las listas `tlist()`, el primer termino de la lista almacena los campos, y los siguientes términos los datos asociados a dichos campos con el mismo orden. El primer campo del primer término no se contabiliza como campo.

En el ejemplo, se puede observar como acceder a la información de la lista empleando tanto el índice como el nombre del campo.

Las listas de tipo `typed` orientada a matrices, es similar a las de tipo `typed` ordinario, pero tienen como peculiaridad que no permiten el acceso a datos empleando índices.

Las listas en Maxima son parecidas a las listas ordinarias de Scilab. Se definen empleando `[ ]` en la definición de la variable.

```
(%i1) Plutonio: ["Pu", "Actinidos", ["Masa", "Densidad", "P_Fusion"], [244, 19816, 912.5]];
(%o1) [Pu, Actinidos, [Masa, Densidad, P_Fusion], [244, 19816, 912.5]]

(%i2) Propiedades: part(Plutonio, 3);
(%o2) [Masa, Densidad, P_Fusion]

(%i3) Val_Propiedades: part(Plutonio, 4);
(%o3) [244, 19816, 912.5]

(%i7) concat(Propiedades[2], ":", Val_Propiedades[2]);
(%o7) Densidad: 19816
```

**Código 2.10** Declaración de una lista en Maxima.

Se han aprovechado los ejemplos anteriores para mostrar la salida de caracteres en consola combinando cadenas de caracteres y números. En el caso de Scilab se han empleado `disp()` y la función `string()`, que convierte una variable numérica en una cadena de caracteres. En Maxima se ha empleado la función `concat()`.

## 2.2.2 Matrices

Este tipo de variable permite almacenar un conjunto de datos de manera ordenada. Las matrices pueden tener la dimensión deseada. El tipo de datos debe ser el mismo y el acceso a un dato de la matriz se suele realizar mediante los índices de la posición que ocupa. Una de las grandes utilidades matemáticas que tienen las herramientas de matemáticas de alto nivel es la manipulación matricial. En áreas de las matemáticas, de la física y de las ingenierías su uso es indispensable. Tanto Scilab como Maxima ofrecen una gran cantidad de funciones para manipular matrices. A continuación se muestra como se define una matriz y como se accede a los datos en ambos lenguajes.

```
-->impar=[1 3 5;7 9 11;13 15 17]
impar =

    1.    3.    5.
    7.    9.   11.
   13.   15.   17.

-->impar(2,1)
ans =

    7.

-->impar(3,:)
ans =

   13.   15.   17.

-->n_impar=[impar(1,:);impar(3,:);impar(2,:)]
n_impar =

    1.    3.    5.
   13.   15.   17.
    7.    9.   11.
```

**Código 2.11** Definición de los componentes de una matriz en Scilab.

Se puede observar en el ejemplo anterior de Scilab, que las matrices se definen introduciendo los datos entre corchetes `[]`, donde los componentes de las filas están separados por espacios y las columnas por `;`. El acceso a un término de la matriz se realiza mediante el índice que ocupa entre paréntesis `()`. Se puede emplear el término `:` para definir un rango de índices, si sólo se emplea `:` se rellena con todos los elementos de la fila. Si se emplea `1:3`, únicamente se toman los términos 1, 2 y 3. En el ejemplo se puede observar que la construcción de matrices mediante otras matrices es inmediata, sin embargo debe cuidarse la concordancia entre las dimensiones de las matrices.

Se verá a continuación como declarar una matriz con componentes simbólicas y como se accede a sus componentes en Maxima. Se emplea la función `matrix()` para asignar los valores a los componentes de la matriz. Para acceder a un término de la matriz se emplea el nombre de la matriz con los índices del término entre corchetes `[]`.

```
(%i16) M:matrix([a,b],[c,d]);
      (%o16) 
$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$


(%i17) M[2,2];
      (%o17) d
```

**Código 2.12** Asignación de valores simbólicos a una matriz en Maxima.

En el ejemplo siguiente se realizan las mismas operaciones de matriz que en el ejemplo anterior con Scilab. Se va a emplear la función `submatrix()`, que permite eliminar componentes de la matriz. Los dos primeros coeficientes numéricos de la función son las filas y las columnas a eliminar. La función `addrow()` permite concatenar matrices con la misma dimensión de filas.

```

(%i10) impar:matrix([1,3,5],[7,9,11],[13,15,17]);
(%o10)
1  3  5
7  9 11
13 15 17

(%i14) impar[2,1];
(%o14) 7

(%i15) submatrix(1,2,impar);
(%o15)
13 15 17

(%i16) addrow(submatrix(2,3,impar),submatrix(1,3,impar),submatrix(1,2,impar));
(%o16)
1  3  5
7  9 11
13 15 17

```

**Código 2.13** Definición de los componentes de una matriz en Maxima.

Las operaciones más sencillas entre matrices y otros números, como las multiplicaciones y potenciación, por ejemplo, pueden interpretarse de dos maneras distintas.

$$A \cdot B = C$$

Donde  $C$ , puede ser la operación elemento a elemento,

$$c_{ij} = a_{ij} \cdot b_{ij}$$

o bien, el producto matricial,

$$c_{ij} = \sum_{k=1}^n a_{ik} \cdot b_{kj}$$

donde  $n$  es el número de columnas de  $A$  y el número de filas de  $B$ .

Se puede observar esta diferencia en las operaciones en Scilab. Donde la operación  $*$  corresponde al producto matricial y el mismo símbolo precedido por un punto,  $\cdot$ , la operación elemento a elemento.

```
-->A=2*ones(3,3)
A =

    2.    2.    2.
    2.    2.    2.
    2.    2.    2.

-->B=3*ones(3,3)
B =

    3.    3.    3.
    3.    3.    3.
    3.    3.    3.

-->A*B
ans =

    18.    18.    18.
    18.    18.    18.
    18.    18.    18.

-->A.*B
ans =

    6.    6.    6.
    6.    6.    6.
    6.    6.    6.
```

**Código 2.14** Operaciones de multiplicación con Scilab.

Maxima también contempla esta posibilidad en el producto. En el siguiente ejemplo se observa como el operador `*` realiza el producto elemento a elemento y el operador `.` realiza el producto matricial. En el ejemplo se observa la multiplicación de un escalar por una matriz de elementos unidad, `ones(dim_filas, dim_filas)`. En el caso de la multiplicación por un escalar el resultado de emplear un operador u otro nos dará el mismo resultado.

```
(%i8) A:matrix([2,2,2],[2,2,2],[2,2,2]);
(%o8) [ 2 2 2
        2 2 2
        2 2 2 ]

(%i9) B:matrix([3,3,3],[3,3,3],[3,3,3]);
(%o9) [ 3 3 3
        3 3 3
        3 3 3 ]

(%i10) A*B;
(%o10) [ 6 6 6
         6 6 6
         6 6 6 ]

(%i11) A.B;
(%o11) [ 18 18 18
         18 18 18
         18 18 18 ]
```

**Código 2.15** Operaciones de multiplicación con Maxima.

Se puede observar como las mismas operaciones matriciales realizadas de manera numérica por Maxima pueden ser realizadas de manera simbólica, incluyendo incluso valores numéricos.

```
(%i16) M:matrix([a,b],[c,d]);
(%o16)  $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ 

(%i18) N:matrix([u,v],[x,y]);
(%o18)  $\begin{bmatrix} u & v \\ x & y \end{bmatrix}$ 

(%i19) M+N;
(%o19)  $\begin{bmatrix} u+a & v+b \\ x+c & y+d \end{bmatrix}$ 

(%i20) M*N;
(%o20)  $\begin{bmatrix} a u & b v \\ c x & d y \end{bmatrix}$ 

(%i21) M.N;
(%o21)  $\begin{bmatrix} b x + a u & b y + a v \\ d x + c u & d y + c v \end{bmatrix}$ 

(%i22) M*5;
(%o22)  $\begin{bmatrix} 5 a & 5 b \\ 5 c & 5 d \end{bmatrix}$ 
```

**Código 2.16** Operaciones aritméticas de matrices con componentes simbólicas con Maxima.

En la siguiente tabla se observa la diferencia entre los operadores básicos matriciales de Maxima y Scilab.

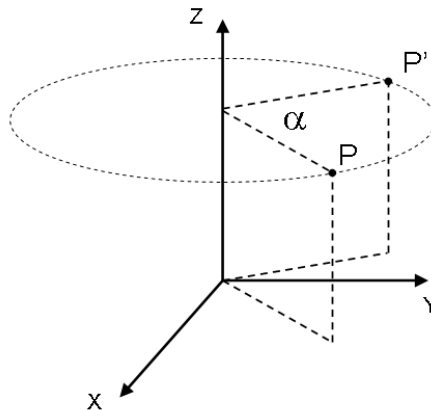
Operaciones	Scilab	Maxima
Producto elemento-elemento	. *	*
Producto matricial	*	.
División elemento-elemento	. /	/
División matricial	/	
Potencia elemento-elemento	. ^	^
Potencia matricial	^	^^

**Tabla 2.11** Operaciones entre matrices.

Además de los operadores básicos mostrados sobre matrices, ambas herramientas pueden realizar gran número de operaciones, como la inversión, transposición, cálculo de autovalores y definición de matrices diagonales, de identidad y nulas de manera inmediata. Estas operaciones se mostrarán en el siguiente apartado.



**Ejemplo 2.2** Realizar una transformación de rotación sobre una coordenada cartesiana  $(i,j,k)$ , con origen en  $(0,0,0)$ . Donde el ángulo de giro  $\alpha=\pi/4$ , corresponde al eje  $z$ .



La rotación del sistema respecto a eje  $z$  corresponde a la siguiente operación,

$$\begin{pmatrix} i' \\ j' \\ k' \end{pmatrix} = \begin{pmatrix} \cos \alpha & \sin \alpha & 0 \\ -\sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} i \\ j \\ k \end{pmatrix}$$

Solución del ejemplo propuesto para Scilab para el punto  $(1,1,1)$  y  $\alpha=\pi/4$ .

```
-->Pos=[1;1;1];

-->alpha=%pi/4;

-->M_rot=[cos(alpha) sin(alpha) 0; -sin(alpha) cos(alpha) 0; 0 0 1];

-->Pos_Rot=M_rot*Pos
Pos_Rot =

    1.4142136
    1.110D-16
    1.
```

**Código 2.17** Resolución de problema 2.2 con Scilab.

La resolución simbólica del problema se puede realizar en Maxima de la siguiente manera,

```

(%i14) Pos:matrix([i],[j],[k]);
      [ i ]
      [ j ]
      [ k ]

(%i15) M_Rot:matrix([cos(alpha),sin(alpha),0],
      [-sin(alpha),cos(alpha),0],
      [0,0,1]);
      [ cos(alpha)  sin(alpha)  0 ]
      [ -sin(alpha) cos(alpha)  0 ]
      [ 0          0          1 ]

(%i16) Pos_rot:M_Rot.Pos;
      [ sin(alpha) j + cos(alpha) i ]
      [ cos(alpha) j - sin(alpha) i ]
      [ k ]
    
```

**Código 2.18** Resolución de problema 2.2 con Maxima.

Observa como se pueden incluir valores en Maxima para calcular el valor del ejercicio propuesto. El formato para sustituir las variables simbólicas por los valores numéricos se puede realizar mediante la estructura `variable,num`. La diferencia que se obtiene entre el cálculo realizado entre Maxima y Scilab es que Maxima ha realizado el cálculo exacto de la solución, mientras que Scilab ha realizado el calculo numérico aproximado.

```

(%i17) alpha:%pi/4$
      i:1$
      j:1$
      k:1$
      Pos_rot,num;
      [ sqrt(2) ]
      [ 0 ]
      [ 1 ]

(%o21)
    
```

**Código 2.19** Resolución de problema 2.2 con Maxima, sustitución de valores de las variables.

## ***EJERCICIOS:***

**Ejercicio 2.1** Suma a cada elemento de la matriz  $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$  el valor correspondiente a la suma de sus índices en Maxima.

**Ejercicio 2.2** Comprueba que el producto matricial y la potencia matricial con exponente 2 son equivalentes, emplea la matriz numérica en Scilab.

$$A \cdot A = A^2$$

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

**Ejercicio 2.3** *Calcula una matriz donde sus elementos correspondan elemento a elemento con la raíz cuadrada de la siguiente matriz en Maxima.*

$$A = \begin{pmatrix} u & v \\ w & z \end{pmatrix}$$

$$B = \begin{pmatrix} \sqrt{u} & \sqrt{v} \\ \sqrt{w} & \sqrt{z} \end{pmatrix}$$

## 2.3 OPERACIONES Y FUNCIONES ELEMENTALES

En este apartado se va a dividir el conjunto de operaciones y funciones elementales en 4 grandes grupos.

- Operaciones trigonométricas
- Operaciones entre polinomios
- Operaciones matriciales
- Otras operaciones elementales

### 2.3.1 Operaciones trigonométricas

Las herramientas matemáticas Scilab y Maxima permiten realizar las operaciones trigonométricas más importantes. Desde las más básicas como son el seno, el coseno y la tangente, como las funciones inversas de estas, o las hiperbólicas, entre otras.

Para obtener la información más detallada sobre el uso de cada una de ellas, consulte los manuales de ayuda de cada herramienta.

Habitualmente, las funciones trigonométricas más sencillas tienen una sintaxis del tipo: función(argumento), como por ejemplo  $\sin(\pi/2)$ , expresión válida para ambas herramientas.

En primer lugar se mostrará un ejemplo en Maxima sobre el uso simbólico de algunas funciones trigonométricas.

```
(%i14) sin(a+b)/tan(c)^2;
(%o14)  $\frac{\sin(b+a)}{\tan(c)^2}$ 

(%i15) atan(%)+acoth(d);
(%o15)  $\operatorname{acoth}(d) + \operatorname{atan}\left(\frac{\sin(b+a)}{\tan(c)^2}\right)$ 
```

**Código 2.20** Operaciones trigonométricas simbólicas con Maxima

Las operaciones trigonométricas en Maxima pueden ser simplificadas, ofreciendo soluciones más comprensibles al usuario. Estas funciones de simplificación tienen especial importancia en el cálculo algebraico y la manipulación de funciones trigonométricas simbólicas, debido a que las soluciones no se muestran habitualmente de manera compacta. Las simplificaciones y la factorización se estudiarán en el tema 3.

A continuación se muestra como se realizan algunos cálculos trigonométricos numéricos y simbólicos con Maxima y Scilab,

Scilab	Maxima
<pre>--&gt;tan(%pi/4) ans =  1.</pre>	<pre>(%i57) tan(%pi/4); (%o57) 1</pre>
<pre>--&gt;sin(%pi/3)^2+cos(%pi/3)^2 ans =  1.</pre>	<pre>(%i58) sin(%pi/3)^2+cos(%pi/3)^2; (%o58) 1</pre>
<pre>--&gt;sin(%pi/4) ans =  0.7071068</pre>	<pre>(%i59) sin(0.5)^2+cos(0.5)^2; (%o59) 1.0</pre>
<pre>--&gt;cotg(asec(2)) ans =  0.5773503</pre>	<pre>(%i60) sin(%pi/4); (%o60) <math>\frac{1}{\sqrt{2}}</math></pre>
	<pre>(%i61) cot(asec(2)); cot(asec(2.0)); (%o61) <math>\frac{1}{\sqrt{3}}</math> (%o62) 0.57735026918963</pre>

**Tabla 2.12** Algunas operaciones trigonométricas con Maxima y Scilab.

Se puede observar que en la tabla anterior las operaciones trigonométricas realizadas en Scilab son evaluadas numéricamente. Sin embargo, en Maxima, algunas funciones se han calculado simbólicamente y otras de manera numérica. La razón, es que Maxima tiene un conjunto de soluciones simbólicas para determinados valores. Observa que las dos primeras expresiones se han calculado simbólicamente, mientras que la tercera se ha evaluado numéricamente. Se puede forzar que Maxima calcule numéricamente la solución, tal y como se puede ver en la quinta expresión, si en vez de emplear 2 como argumento se utiliza el valor 2.0.

### 2.3.2 Operaciones polinómicas

Las operaciones polinómicas tienen sentido dentro de la manipulación simbólica de este tipo de expresiones. Los polinomios están compuestos por sumas y restas de potencias naturales de variables, multiplicadas por coeficientes simbólicos o numéricos. En el siguiente ejemplo para Maxima, se puede ver como se han definido varios polinomios en función de la variable  $x$ , y como se realizan varias operaciones básicas entre ellos, con resultado no simplificado.

```
(%i69) P1:x^2-2*x+1;
(%o69) x^2-2 x+1

(%i71) P2:a*x^2;
(%o71) a x^2

(%i80) P3:c(x+1);
(%o80) c(x+1)

(%i73) P1*P2;
(%o73) a x^2 (x^2-2 x+1)

(%i74) P1/P2;
(%o74)  $\frac{x^2-2 x+1}{a x^2}$ 

(%i82) exp(P1/(P2+P3));
(%o82)  $\%e^{\frac{x^2-2 x+1}{c(x+1)+a x^2}}$ 
```

**Código 2.21** Operaciones polinómicas con Maxima.

Las simplificaciones de expresiones polinómicas se explicarán con detalle en el tema 3.

### 2.3.3 Operaciones matriciales

En las matemáticas, como en otras disciplinas científicas, gran número de operaciones matemáticas requieren del uso de matrices. Scilab y Maxima tienen gran número de funciones que permiten operar con matrices, además de las más sencillas ya vistas anteriormente como las sumas o multiplicaciones, se pueden realizar operaciones específicas como el cálculo de la inversa, la transpuesta o el determinante, entre otras.

#### Cálculo de la inversa

Maxima puede calcular la inversa de una matriz empleando la función `invert(Matriz)`. También puede ser expresada mediante la potenciación no conmutativa de exponente -1, `Matriz^^-1`.

En la siguiente figura se muestra el cálculo de la inversa de una matriz de 2x2 de manera simbólica, donde todos sus componentes son distintos.

```
(%i83) M:matrix([a,b],[c,d]);
(%o83)  $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ 

(%i84) invert(M);
(%o84)  $\begin{bmatrix} \frac{d}{a d-b c} & -\frac{b}{a d-b c} \\ -\frac{c}{a d-b c} & \frac{a}{a d-b c} \end{bmatrix}$ 
```

**Código 2.22** Inversa de una matriz con Maxima.

Scilab realiza la misma función de manera numérica, en este caso sobre una matriz de 3x3.

```
-->A=[1,2,3;4,5,5;7,8,9]
A =

    1.    2.    3.
    4.    5.    5.
    7.    8.    9.

-->inv(A)
ans =

- 0.8333333 - 1.    0.8333333
 0.1666667    2. - 1.1666667
 0.5         - 1.    0.5
```

### Código 2.23 Inversa de una matriz con Scilab.

#### Cálculo del determinante

Maxima calcula el determinante de una matriz empleando la función `determinant(Matriz)`, empleando un método similar a la eliminación de Gauss.

```
(%i85) M:matrix([a,b],[c,d]);
(%o85)  $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ 

(%i86) determinant(M);
(%o86)  $a d - b c$ 
```

### Código 2.24 Determinante de una matriz con Maxima.

Seguidamente se muestra un ejemplo del cálculo del determinante con Scilab. Scilab emplea la función `det(Matriz)`.

```
-->A=[1,2,3;4,5,5;7,8,9];

-->det(A)
ans =

- 6.
```

### Código 2.25 Determinante de una matriz con Scilab.

#### Cálculo de la transpuesta

La matriz transpuesta se realiza en Maxima mediante la función `transpose(Matriz)`.

```
(%i90) M:matrix([a,b],[c,d]);
(%o90)  $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ 

(%i91) transpose(M);
(%o91)  $\begin{bmatrix} a & c \\ b & d \end{bmatrix}$ 
```

### Código 2.26 Transpuesta de una matriz con Maxima.

En Scilab la transpuesta de una matriz se calcula mediante el operador "'",

```
-->A=[1,2,3;4,5,5;7,8,9];

-->A'
ans =

    1.    4.    7.
    2.    5.    8.
    3.    5.    9.
```

**Código 2.27** Transpuesta de una matriz con Scilab.

Una vez analizadas algunas de las operaciones más básicas, se muestra como se pueden realizar otras operaciones de gran utilidad,

```
(%i110) M:matrix([a,b],[c,d]);
(%o110) [ a b
         c d ]

(%i111) rank(M);
(%o111) 2

(%i112) M.ident(2);
(%o112) [ a b
         c d ]

(%i113) M.zeromatrix(2,2);
(%o113) [ 0 0
         0 0 ]
```

**Código 2.28** Algunas operaciones matriciales de utilidad con Maxima.

En el ejemplo anterior se calcula el rango de la matriz, `rank(Matriz)`. Se declara matrices unidad de orden N, `ident(N)`, y matrices nulas de dimensión NxM, `zeromatrix(N,M)`.

Numéricamente, Scilab tiene el mismo juego de funciones que las mostradas en el ejemplo anterior.

```
-->A=[1,2,3;4,5,5;7,8,9];

-->rank(A)
ans =

    3.

-->A*eye(3)
ans =

    1.    2.    3.
    4.    5.    5.
    7.    8.    9.

-->A*zeros(3)
ans =

    0.    0.    0.
    0.    0.    0.
    0.    0.    0.
```

**Código 2.29** Algunas operaciones matriciales de utilidad con Scilab.

Donde el rango de la matriz se calcula empleando la función `rank(Matriz)`. La matriz unidad de orden  $N$ , `eye(N)`, y la matriz nula de dimensión  $N$ , `zeros(N)`.

Se ilustrará el uso de las operaciones matriciales mediante unos sencillos ejemplos.

### **Ejemplo 2.3** Demostración de ortogonalidad de una matriz de 2x2.

De manera resumida, una matriz ortogonal debe cumplir que su matriz inversa y su matriz transpuesta sean iguales, por lo tanto:

$$A \cdot A^t = I$$

Se resuelve la ecuación dada de manera simbólica para una matriz de 2x2 con Maxima.

```
(%i7) M:matrix([a,b],[c,d]);
(%o7) [a b]
      [c d]

(%i8) determinant(M);
(%o8) a d - b c

(%i9) invert(M)=transpose(M);
(%o9) [d/(a d - b c)  -b/(a d - b c)] = [a c]
      [-c/(a d - b c)  a/(a d - b c)]   [b d]
```

**Código 2.30** Ejercicio 2.3 con Maxima.

Las ecuaciones obtenidas son las siguientes:



$$\frac{d}{|M|} = a ; \frac{-b}{|M|} = c ; \frac{-c}{|M|} = b ; \frac{a}{|M|} = d$$

sólo puede cumplirse cuando el determinante de la matriz sea  $ad - bc = 1$  y ( $d = a$ ) y ( $-b = c$ )

En el tema 4 se verá como resolver sistemas de ecuaciones no lineales.

### **Ejemplo 2.4** Cálculo de los autovalores de una matriz de 2x2.

Los autovalores son las soluciones del polinomio de  $\lambda$ ,

$$q(\lambda) = |M - \lambda I|$$

Se resuelve este problema para una expresión general simbólica con Maxima,

```
(%i17) M:matrix([a,b],[c,d]);
(%o17)  $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ 

(%i18) M-l*ident(2);
(%o18)  $\begin{bmatrix} a-l & b \\ c & d-l \end{bmatrix}$ 

(%i19) q_1:determinant(%);
(%o19)  $(a-l)(d-l)-bc$ 

(%i20) solve(q_1,l);
(%o20)  $[l = -\frac{\sqrt{d^2-2ad+4bc+a^2}-d-a}{2}, l = \frac{\sqrt{d^2-2ad+4bc+a^2}+d+a}{2}]$ 

(%i21) eigenvalues(M);
(%o21)  $[[-\frac{\sqrt{d^2-2ad+4bc+a^2}-d-a}{2}, \frac{\sqrt{d^2-2ad+4bc+a^2}+d+a}{2}], [1, 1]]$ 
```

**Código 2.31** Ejercicio 2.4 con Maxima.

Una vez obtenido el polinomio en función de  $l$ , se resuelve la ecuación mediante la función `solve(ecuación, variable)`.

Se puede observar que hay una función específica para calcular los autovalores de una matriz, `eigenvalues(Matriz)`.

### **2.3.4 Otras operaciones elementales**

Tanto en el cálculo simbólico de Maxima, como en el cálculo numérico de Scilab, hay otras operaciones, como el cálculo del factorial de un número natural, el cálculo de logaritmos y exponenciales, tanto para cualquier base, como para matrices. También existe la función de raíz cuadrada. A continuación se muestra una tabla con algunas funciones importantes para las dos herramientas.

Función	Scilab	Maxima
Factorial	<code>factorial()</code>	<code>factorial()</code>
Logaritmo natural	<code>log()</code>	<code>log()</code>
Exponencial natural	<code>exp()</code>	<code>exp()</code>
Raíz cuadrada	<code>sqrt()</code>	<code>sqrt()</code>

**Tabla 2.13** Otras operaciones con Maxima y Scilab.

Seguidamente se muestran algunos ejemplos de estas funciones para Maxima y Scilab.

Scilab	Maxima
<pre>--&gt;factorial(7) ans =  5040.</pre>	<pre>(%i64) 7!; factorial(7); (%o64) 5040 (%o65) 5040</pre>
<pre>--&gt;exp(log(15)) ans =  15.</pre>	<pre>(%i66) log(exp(a+b)); (%o66) b+a</pre>
<pre>--&gt;sqrt(27^2) ans =  27.</pre>	<pre>(%i67) exp(A+B); M:matrix([a,b],[c,d]); exp(M); (%o67) %e<sup>B+A</sup> (%o69) <math>\begin{bmatrix} e^a &amp; e^b \\ e^c &amp; e^d \end{bmatrix}</math></pre>
	<pre>(%i70) sqrt((z+4)^4); (%o70) (z+4)<sup>2</sup></pre>

**Tabla 2.14** Ejemplo de otras operaciones con Maxima y Scilab.

## EJERCICIOS:

**Ejercicio 2.4** Define las siguientes expresiones en Maxima.

a) 
$$\frac{\sin(x^2 + e^{xy} + y^2)}{\log(xy)}$$

b) 
$$f_1(x) = x^2, \quad f_2(x, y) = x + \sin(y), \quad f_3(x, y) = e^{\frac{f_1}{f_2}}$$

**Ejercicio 2.5** Calcula el valor de las expresiones del problema anterior con los valores  $x=1$  e  $y=0.5$  empleando Scilab.

**Ejercicio 2.6** *Calcula las siguientes expresiones con Maxima,*

$$N = x^2 + ax + c$$

$$M = bx^3 + dx + 1$$

$$P = \frac{N}{M}$$

$$Q = e^P$$

$$R = \sin(Q)$$

## 2.4 REPRESENTACIONES GRÁFICAS

La representación gráfica es una herramienta fundamental en todas las disciplinas científicas, ya que permite mostrar resultados que pueden ser analizados de manera más rápida e intuitiva. Maxima y Scilab cuentan con unas potentes funciones para la representación gráfica. En este apartado se mostrarán algunas de estas funciones. Las funciones gráficas cuentan con numerosos parámetros para mejorar el aspecto de sus representaciones, sin embargo no se hará hincapié en estas características, y se dará mayor importancia a la muestra de los resultados.

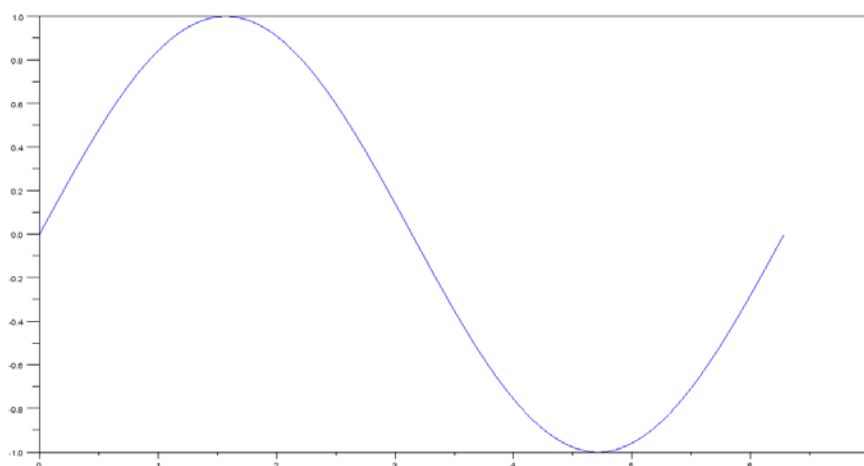
### 2.4.1 Gráficas en 2 dimensiones

En los ejemplos 1 y 2 del tema 1 ya se mostró como emplear algunas sentencias para generar gráficas en 2 dimensiones con Scilab y Maxima. Una manera sencilla de obtener una representación gráfica en 2 dimensiones, realizando una sustitución numérica de la función  $\sin(x)$ , con  $x=[0,2\pi]$  en Scilab se podría realizar de la siguiente manera,

```
-->x=0:0.01:2*pi;
-->y=sin(x);
-->plot(x,y)
```

**Código 2.32** Empleo de la función `plot()` con Scilab.

El resultado se muestra en la siguiente gráfica. La gráfica está formada por 629 puntos equidistantes (0.01) y evaluadas en  $y$ . La función empleada es `plot()`.



**Figura 2.1** Representación gráfica de 2D con la función `plot()` en Scilab.

De manera similar, Maxima permite realizar este mismo tipo de representación de puntos de una función. En este caso también se representan 8 puntos de la función  $\sin(x)$ ,  $x=[0,3.5]$ . Es común el uso de la función `plot2d()` para las representaciones de funciones de 2D. Para la representación de puntos definidos se emplea el parámetro de la función `discrete`.

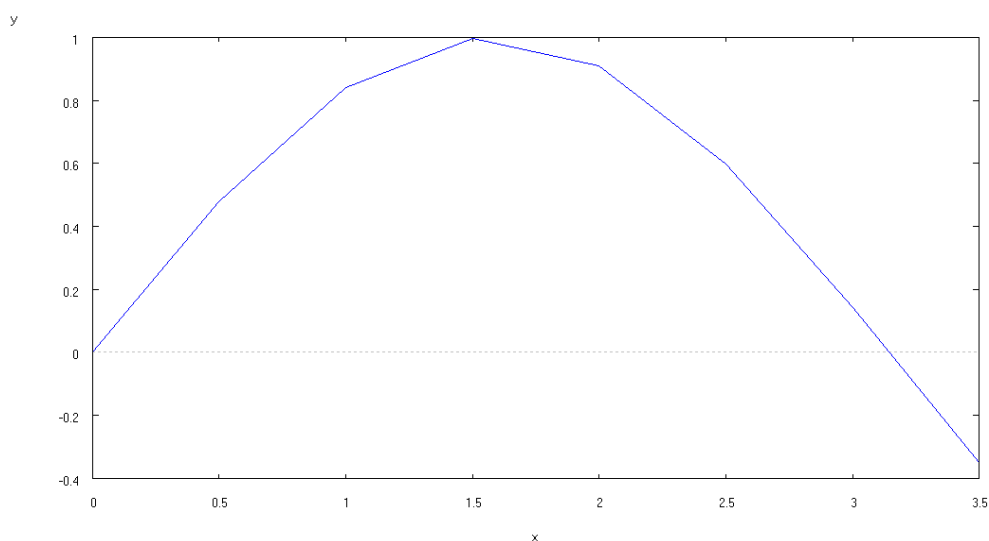
```
(%i71) x:[0,0.5,1,1.5,2,2.5,3,3.5];
(%o71) [0,0.5,1,1.5,2,2.5,3,3.5]

(%i72) y:sin(x);
(%o72) [0,0.4794255386042,sin(1),0.99749498660405,sin(2),0.59847214410396,sin(3),-
0.35078322768962]

(%i73) plot2d([discrete,x,y]);
```

**Código 2.33** Representación de puntos definidos en Maxima.

Dando como resultado la siguiente gráfica, donde cada punto discreto se une mediante una línea.



**Figura 2.2** Representación gráfica de puntos discretos en 2D mediante la función `plot2d()` en Maxima.

Puede ser de interés mostrar en la misma gráfica valores discretos y funciones simbólicas.

En Scilab, todos los valores son discretos, por lo tanto, esta tarea se realiza representando graficas con más de una función.

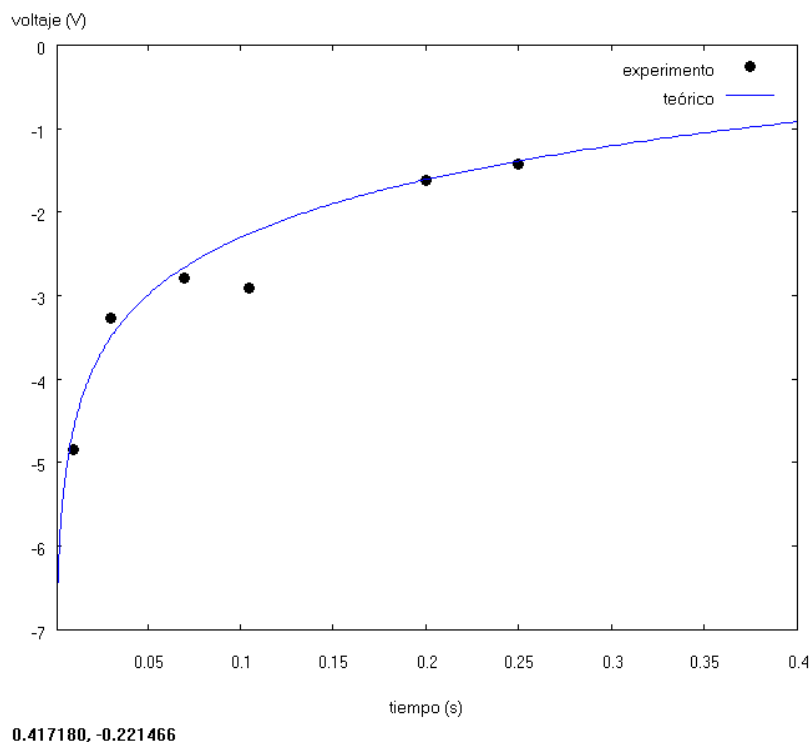
En Maxima se puede hacer dentro de una sola llamada a la función `plot2d()`. En el siguiente ejemplo, se desea representar simultáneamente los valores experimentales y los valores teóricos calculados.

```
(%i70) xy_exp:[0.01,-4.84], [0.03,-3.27], [0.07,-2.79], [0.105,-2.91], [0.2,-1.62], [0.25,-1.42]];
(%o70) [[0.01,-4.84],[0.03,-3.27],[0.07,-2.79],[0.105,-2.91],[0.2,-1.62],[0.25,-1.42]]

(%i86) plot2d([[discrete, xy_exp],log(1)], [1,0.001,0.4],
[style, [points,2,5,1], [lines,1,1]],
[legend,"experimento","teórico"],
[xlabel,"tiempo (s)"], [ylabel,"voltaje (V)"])$
```

**Código 2.34** Representación en la misma gráfica de puntos discretos y de una función simbólica en Maxima.


En el ejemplo anterior, en primer lugar se declaran los pares de puntos del experimento. Dentro de la función, se definen los puntos discretos que se desean representar gráficamente y la función. Se aprovecha este ejemplo del uso de esta función para ilustrar el manejo de algunos parámetros para definir el aspecto de la función `plot2d()`. El parámetro `style`, define el aspecto de `points` y `lines` de las gráficas. Estos parámetros corresponden con el formato de los puntos y de las líneas que los unen. El parámetro `legend`, define los nombres de las leyendas para cada gráfica. Los parámetros `xlabel` e `ylabel` definen los nombres de los ejes. Puede observar que las definiciones siguen el orden en que se han declarado las funciones. El resultado del uso del código de Maxima es el siguiente,



**Figura 2.3** Representación gráfica de puntos discretos y función en Maxima.

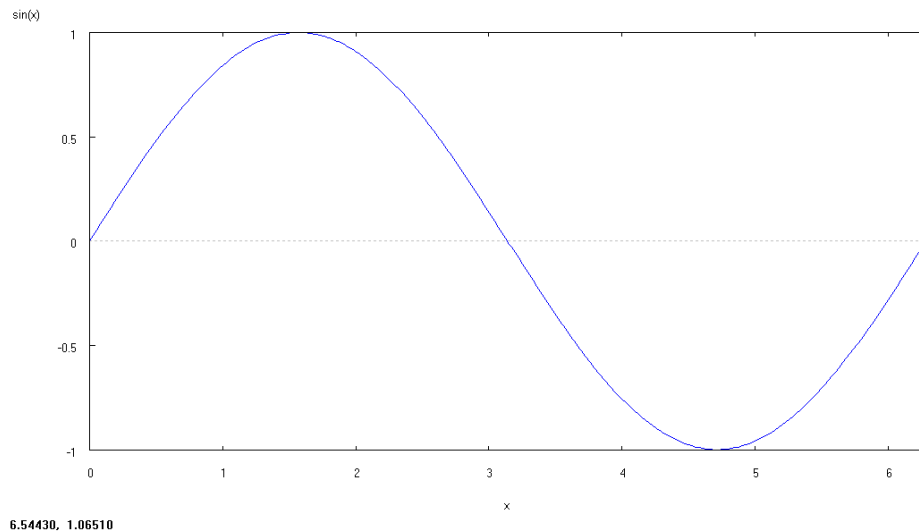
Sin embargo, debe decirse que Maxima está orientada a la representación gráfica de la función simbólica. La función `plot2d()`, también puede realizar esta tarea. En este caso, la sintaxis de la función será:

`plot2d(función, [variable, min_var, max_var]),`

 (%i2) `plot2d(sin(x), [x, 0, %pi*2])$`

**Código 2.35** Uso simple de la función `plot2d()`.

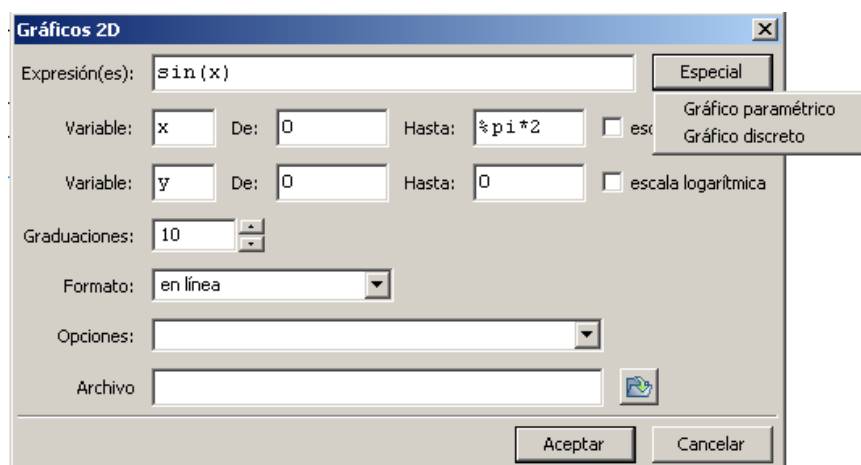
Esta expresión da como resultado la siguiente gráfica,



**Figura 2.4** Representación gráfica de una función simbólica en 2D (`plot2d()`) con Maxima.

Una de las propiedades de la ventana gráfica de Maxima es que muestra la posición del cursor en el borde inferior izquierdo.

Maxima permite la representación gráfica desde la propia ventana de wxMaxima, accediendo desde el menú *Gráfico*, se pueden hacer representaciones en 2 y 3 dimensiones. En la siguiente figura se ilustra el ejemplo anterior empleando el interface gráfico que ofrece wxMaxima para la función en 2 dimensiones.



**Figura 2.5** Ventana de wxMaxima para la representación gráfica de funciones.

Se puede representar funciones paramétricas con Maxima, empleando el parámetro de `plot2d()`, `parametric`.

### **Ejemplo 2.5** Representación gráfica de una elipse.

En este caso se representa una elipse mediante ecuaciones paramétricas:

$$x = a \cos(t)$$

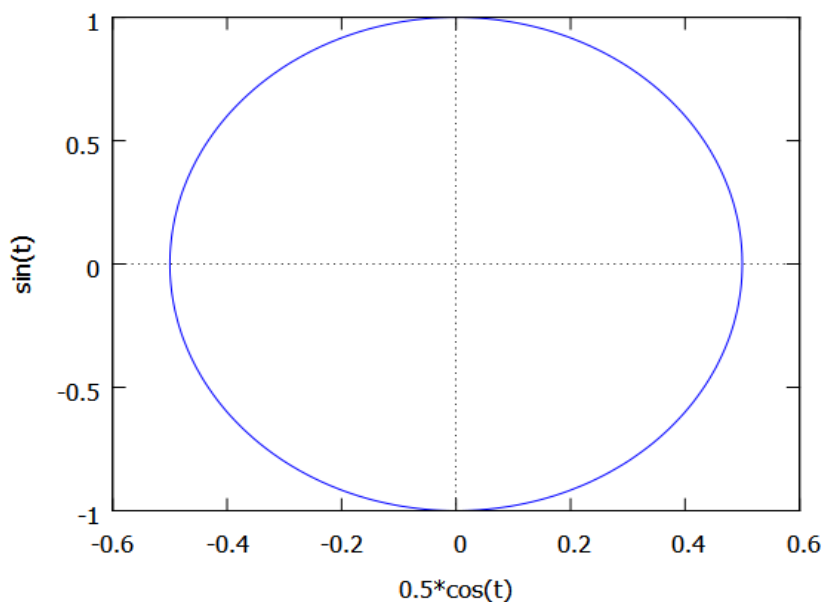
$$y = b \sin(t)$$

Para obtener la representación gráfica de estas ecuaciones paramétricas se emplea el código mostrado a continuación,

```
(%i2) plot2d([parametric, 0.5*cos(t), sin(t), [t,0,%pi*2]]);
```

**Código 2.36** Representación gráfica de funciones paramétricas del ejercicio 2.5.

En este caso, después de emplear el parámetro `parametric`, se introduce las dos funciones. El siguiente vector de parámetros corresponde a la variable y los valores mínimo y máximo de esta. Finalmente, se indica el rango de puntos que debe componer la gráfica. En este caso, se dibuja la elipse entre  $[0, 2\pi]$ . Como resultado se obtiene la siguiente representación,



**Figura 2.6** Representación gráfica de una función paramétrica del ejercicio 2.5.

**Ejemplo 2.6** Dibujar una espiral representada por las siguientes ecuaciones, donde la espiral da 5 vueltas completas.

$$x = t \cos(t)$$

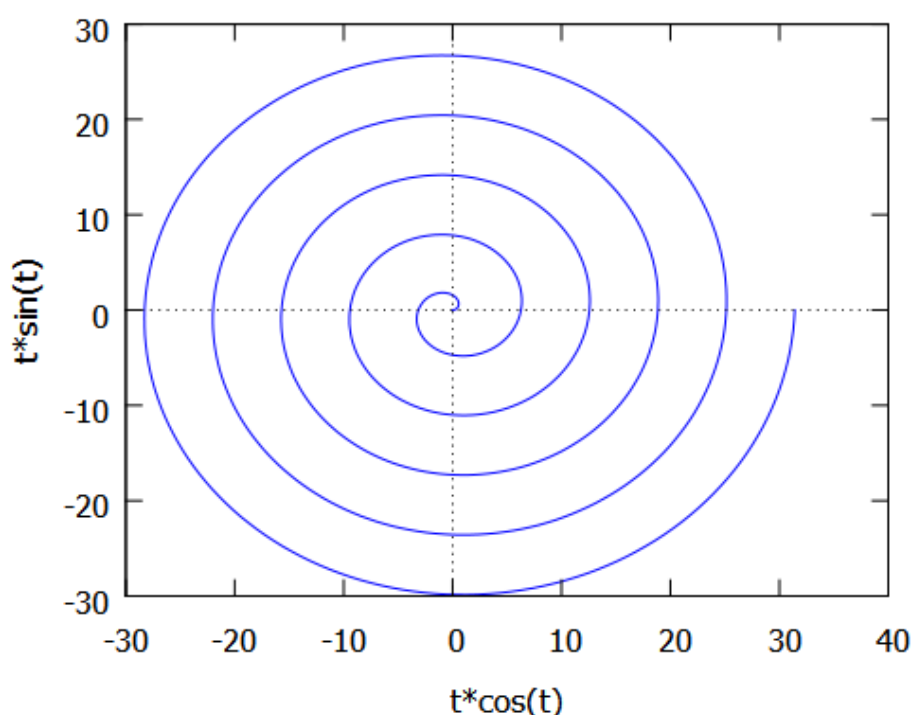
$$y = t \sin(t)$$

Para representar esta función, se emplea la representación gráfica paramétrica de Maxima.

```
(%i1) plot2d([parametric, t*cos(t), t*sin(t), [t, 0, %pi*2*5]]);
```

**Código 2.37** Representación gráfica de funciones paramétricas del ejercicio 2.6.

La gráfica obtenida se muestra a continuación,



**Figura 2.7** Representación gráfica de una espiral con Maxima.

**Ejemplo 2.7** Representación gráfica de la función  $3x^2 - y^2 = 6$  y de las asíntotas,

$$y = \sqrt{\frac{6}{2}}x$$

$$y = -\sqrt{\frac{6}{2}}x$$

La función planteada es una hipérbola. En su forma canónica se puede expresar como,

$$\frac{x^2}{2} - \frac{y^2}{6} = 1$$



Las funciones hipérbola no está evaluadas para algunos valores de  $x$ . Se puede representar esta función en forma paramétrica mediante las siguientes ecuaciones,

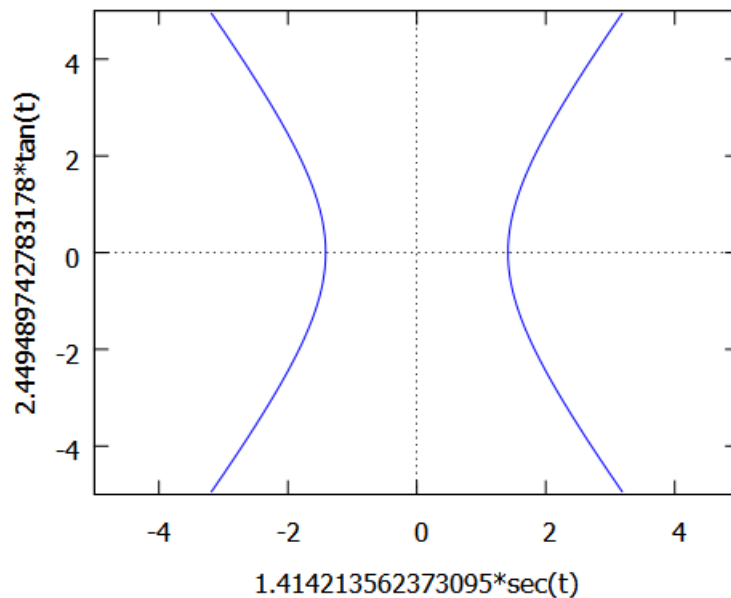
$$x = \sqrt{2} \sec(t)$$

$$y = \sqrt{6} \tan(t)$$

Empleando la siguiente sentencia para Maxima, se obtiene la representación mostrada a continuación.

```
(%i3) plot2d([parametric, (2^0.5)*sec(t), (6^0.5)*tan(t), [t,0,%pi*2]],
[x,-5,5], [y,-5,5]);
```

**Código 2.38** Representación gráfica de una hipérbola en forma paramétrica.



**Figura 2.8** Representación gráfica de una hipérbola con Maxima.

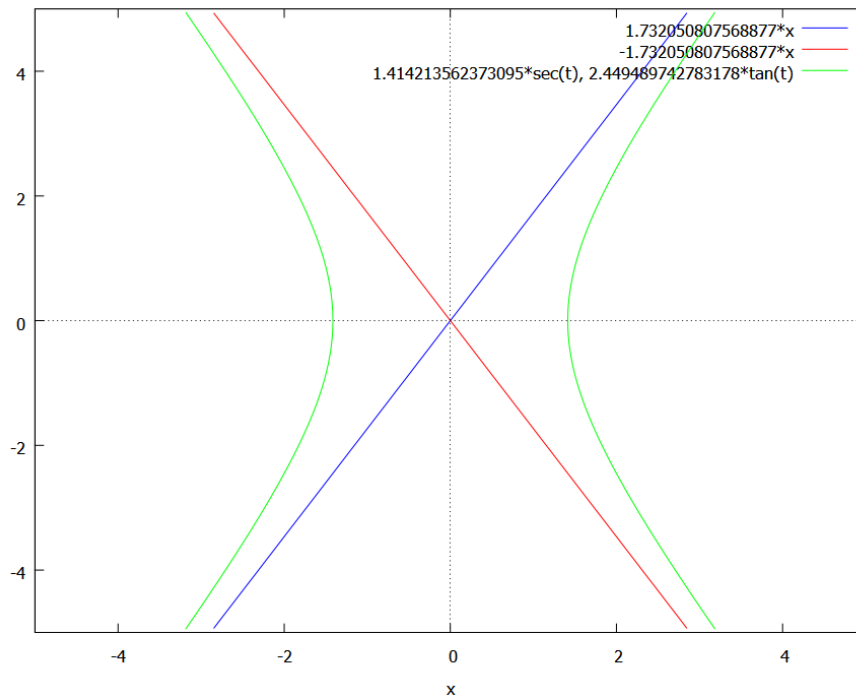
Maxima permite representar en la misma gráfica funciones y curvas paramétricas. Se emplea la siguiente sentencia en Maxima,

```
--> plot2d([(3^0.5)*x, (-3^0.5)*x, [parametric, (2^0.5)*sec(t), (6^0.5)*tan(t),
[t,0,%pi*2]]], [x,-5,5], [y,-5,5]);
```

**Código 2.38** Representación gráfica de una hipérbola en forma paramétrica y sus asíntotas.

Se puede observar que las funciones se han descrito en primer lugar y como las curvas paramétricas emplean internamente las variables  $x$  e  $y$ . También se definen simultáneamente los límites de la gráfica para los dos grupos de funciones.

El resultado de esta expresión se muestra a continuación,



**Figura 2.9** Representación gráfica de una hipérbola y sus asíntotas con Maxima.

En Maxima se puede usar indistintamente la función `plot2d()` o `wxplot2d()`. La diferencia es que la primera produce una salida independientemente de wxMaxima y la segunda incluye como salida del propio programa la representación gráfica.

Se verá a continuación como representar varias funciones en la misma gráfica. Scilab puede representar funciones de manera más compacta que como ha sido mostrado anteriormente.

**Ejemplo 2.8** Representa una familia de funciones en la misma gráfica.

$$f_1(x) = x \cos(x)$$

$$f_2(x) = x \cos^2(x)$$

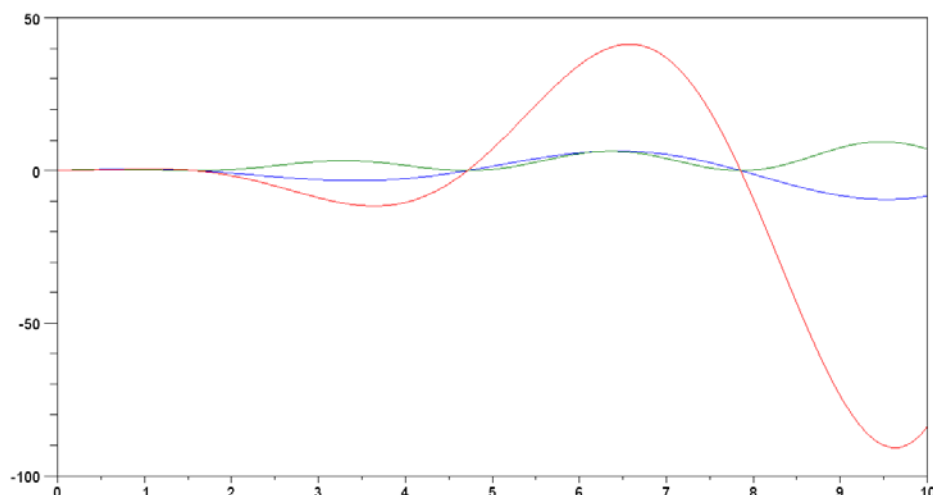
$$f_3(x) = x^2 \cos(x)$$

El código de Scilab empleado para representar simultáneamente esta familia de funciones se muestra a continuación.

```
-->x=[0:0.01:10]';
-->plot(x,[x.*cos(x) x.*(cos(x)).^2 (x.^2).*cos(x)])
```

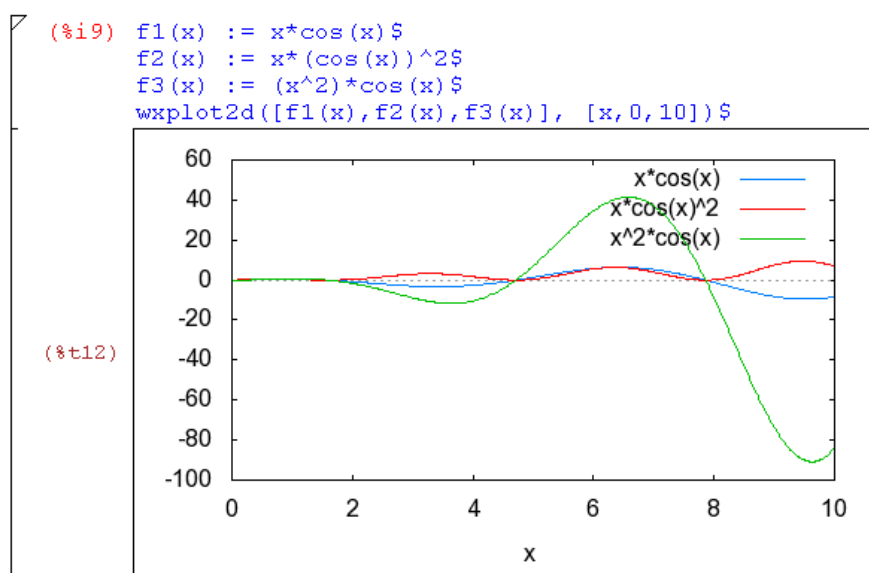
**Código 2.39** Representación gráfica de una familia de funciones con Scilab.

Observa que a diferencia del ejemplo anterior del empleo de la función `plot()`, esta función obliga a introducir un vector de la variable  $x$  con el formato adecuado. El resultado se muestra a continuación.



**Figura 2.10** Representación de varias funciones en la misma gráfica con Scilab.

Para dibujar más de una gráfica en Maxima se debe emplear el siguiente formato de `plot2d()`.



**Figura 2.11** Código y representación de varias funciones en la misma gráfica con Maxima.

En este caso se definen las funciones  $f_1$ ,  $f_2$ ,  $f_3$  previamente.

## 2.4.2 Graficas en 3 dimensiones

En este apartado se mostrarán algunas de las representaciones gráficas en tres dimensiones que ofrecen Scilab y Maxima. Las capacidades de representación gráfica se van a ilustrar mediante varios ejemplos.

**Ejemplo 2.9** Representación gráfica de la siguiente función,

$$f(x, y) = e^x \sin^2(y)$$

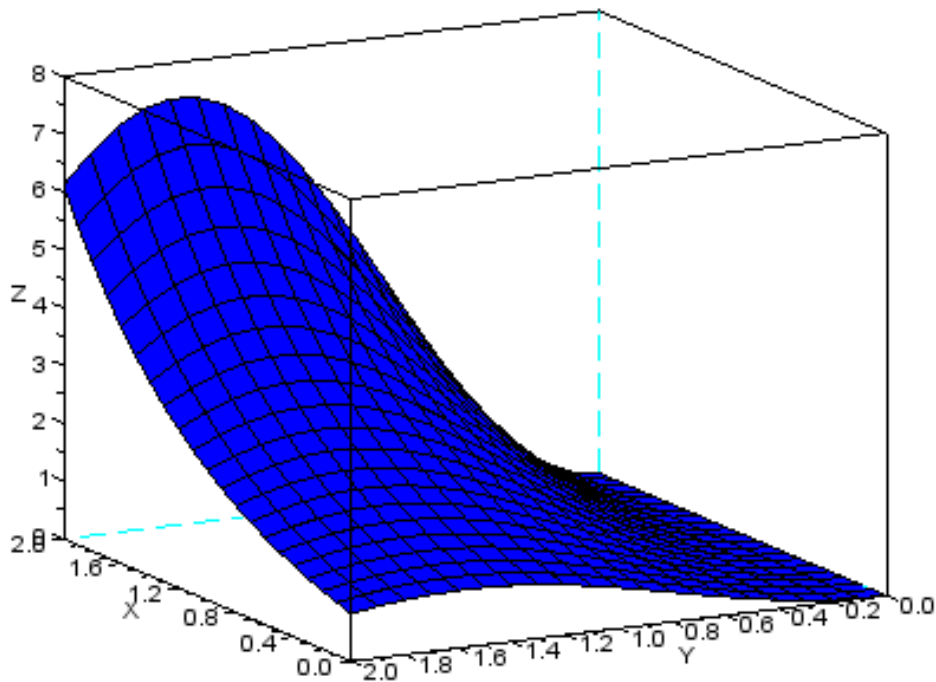
$$x \in [0, 2], y \in [0, 2]$$

A continuación se muestra el código de Scilab empleado, observa que las matrices de datos de la cuadrícula y de la función deben tener el formato adecuado.

```
-->x=[0:0.1:2]';
-->y=[0:0.1:2]';
-->z=exp(x)*sin(y')^2;
-->plot3d(x,y,z)
```

**Código 2.40** Representación gráfica mediante la función `plot3d()` en Scilab.

El resultado de ejecutar el código anterior se muestra en la siguiente figura,



**Figura 2.12** Representación gráfica empleando la función `plot3d()`.

En este caso se ha empleado la función `plot3d()` de Scilab, el uso es muy parecido a la función `fplot3d()`, que se muestra en el siguiente ejemplo. La figura ha sido girada manualmente mediante la herramienta de giro de la ventana gráfica. En el siguiente ejemplo se muestra como se puede configurar el ángulo de la vista.

**Ejemplo 2.10** Representación gráfica de la función de Rosenbrock,

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$

$$x \in [-2, 2], y \in [-1, 3]$$

La función de Rosenbrock es una función muy empleada para la evaluación de algoritmos de optimización. Esta función tiene la peculiaridad de no tener un único mínimo, como se podrá ver a continuación. Se va a obtener una representación gráfica en 3D de la función de Rosenbrock empleando el siguiente código de Scilab.

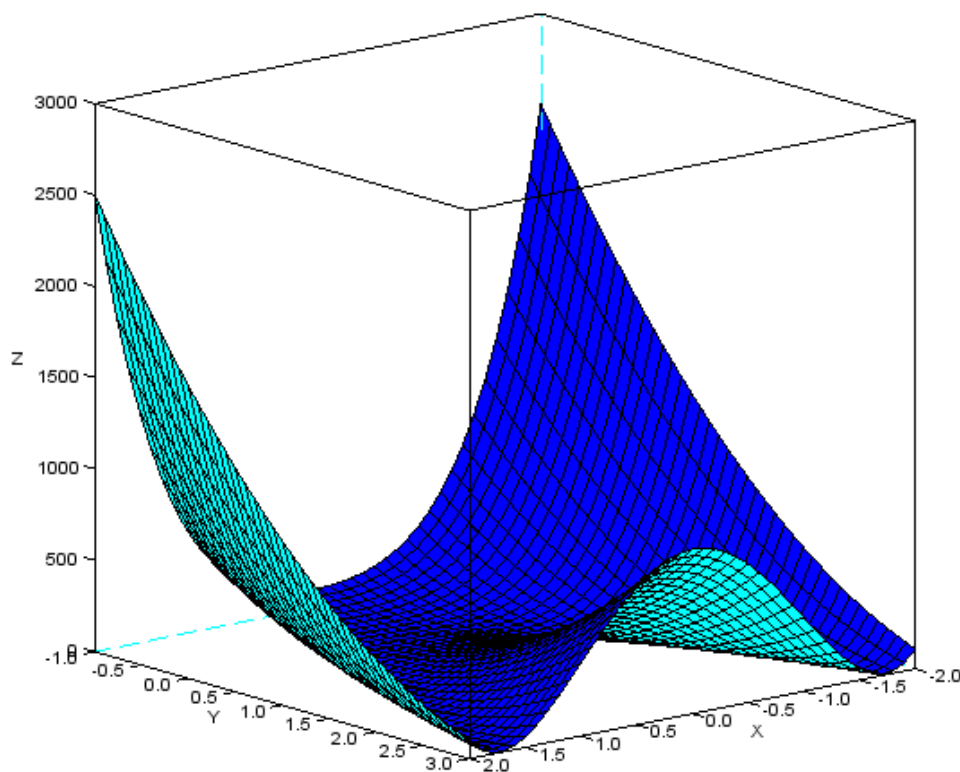
```
-->x=-2:0.1:2;  
-->y=-1:0.1:3;  
-->def('z=f(x,y)', 'z=(1-x)^2+100*(y-x^2)^2');  
-->fplot3d(x,y,f,alpha=0.3,theta=50)
```

**Código 2.41** Representación gráfica de la función de Rosenbrock en Scilab.

En primer lugar se definen los puntos de las variables que se desean representar, definiendo un vector para cada variable. Posteriormente se define una función  $z(x, y)$ , que representa la función de Rosenbrock.

En la función `fplot3d()`, se definen las variables y la función a representar. En este caso se han añadido dos parámetros, que representan los ángulos de visualización de la gráfica.

En resultado obtenido es el siguiente,



**Figura 2.13** Función de Rosenbrock con Scilab.

**Ejemplo 2.11** Obtener estimativamente el mínimo de la siguiente función a partir de su representación gráfica,

$$S(x, y) = 2 \left( xy + \frac{V}{x} + \frac{V}{y} \right)$$

$$V = 1$$

En este ejercicio será adecuado el uso de la representación gráfica mediante contornos. Este tipo de gráfica consiste en una representación de 2 dimensiones donde los ejes son las variables y mediante curvas se representan los valores de la función que tienen el mismo valor.

Se representa la gráfica del contorno de la función  $S(x, y)$  empleando Maxima. En primer lugar se define la función a representar. Posteriormente se emplea la función `contour_plot()` para obtener una gráfica de contorno.

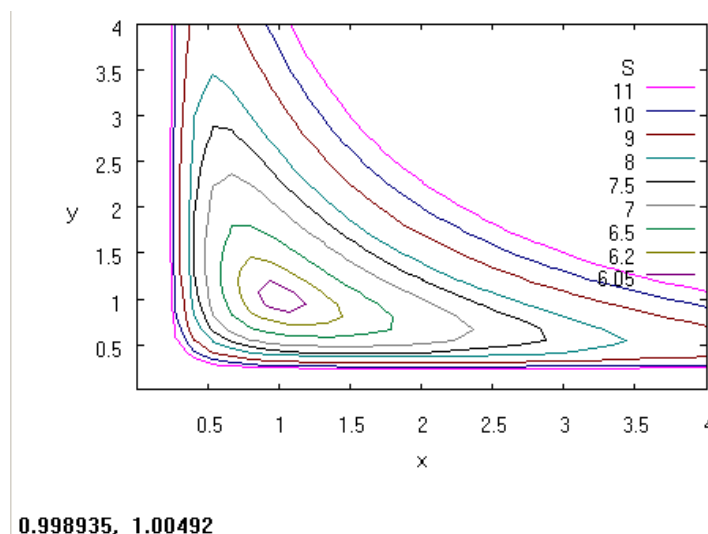
La sintaxis de esta función sin parámetros especiales es,  
`contour_plot(función, [v1, lim_inf_v1, lim_sup_v1], [v2, lim_inf_v2, lim_sup_v2])`.

Para controlar los valores de las curvas del contorno se asignan valores discretos empleando el parámetro de `gnuplot_preamble`, `cntrparam levels discrete`.

```
(%i68) S(x,y):=2*(x*y+(1/x)+(1/y))$
contour_plot(S, [x, 0.01, 4], [y, 0.01, 4],
[gnuplot_preamble, "set contour;
set cntrparam levels discrete 6.05, 6.2, 6.5, 7, 7.5, 8, 9, 10, 11"]);
```

**Código 2.42** Empleo de contornos en Maxima.

En la siguiente figura se muestra el resultado de ejecutar este código, donde se puede observar que al posicionar el cursor sobre la posición aproximada correspondiente al mínimo, el valor de las coordenadas están cercanas a los valores  $x=1$  e  $y=1$ . Por lo tanto el valor mínimo de la función estaría en torno a este valor  $S(1,1) = 6$ .



**Figura 2.14** Contorno de la función  $S(x, y)$  para determinados valores discretos en Maxima.

**Ejemplo 2.12** Representación gráfica del contorno y 3D de la función Griewank, para dos variables y  $x_1, x_2 \in [-6, 6]$ .

$$f(x_1, x_2, \dots, x_n) = 1 + \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right)$$

Esta función es empleada habitualmente para la evaluación de algoritmos de optimización global. La función Griewank es una función multimodal y admite el número de dimensiones deseadas.

La ecuación que se empleará para la representación de 3D será,

$$z(x, y) = 1 + \frac{1}{4000}(x^2 + y^2) - \cos(x) \cos\left(\frac{y}{\sqrt{2}}\right)$$

Un rango típico para este sistema es  $x, y \in [-20, 20]$ . Un ejemplo de código en Scilab para obtener este resultado se muestra a continuación,

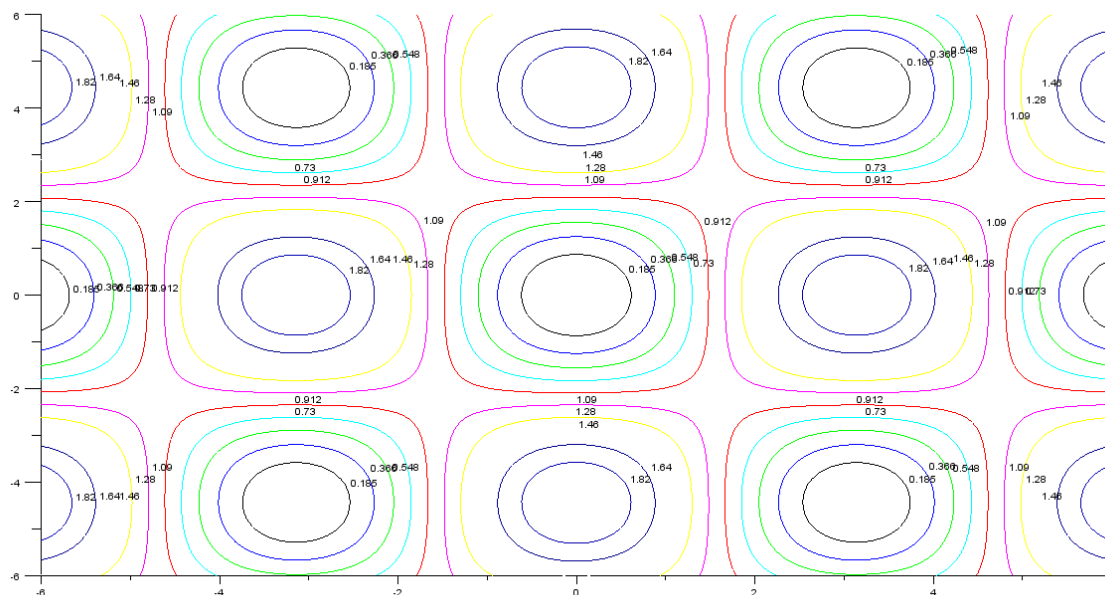
```
-->y=linspace ( -6 , 6 , 100 );
-->x=linspace ( -6 , 6 , 100 );

-->function f = z(x,y);
-->f=1+(1/4000)*(x.^2+y.^2)-cos(x)*cos(y/2^0.5)
-->endfunction

-->contour (x,y,z,10)
```

**Código 2.43** Representación grafica de la función de Griewank mediante contornos en Scilab.

En este caso para visualizar mejor el resultado se ha seleccionado el rango  $x, y \in [-6, 6]$ . En este código se ha empleado la función `linspace()`, para conseguir 100 elementos equiespaciados en el rango deseado. El manejo de esta función es más cómoda que definir la distancia entre los puntos. Después se ha creado la función `z`. En apartados posteriores se mostrará las características de estas estructuras. El resultado obtenido es el siguiente,



**Figura 2.15** Contorno de la función Griewank con Scilab.

A continuación se muestra como realizar esta misma tarea con Maxima.

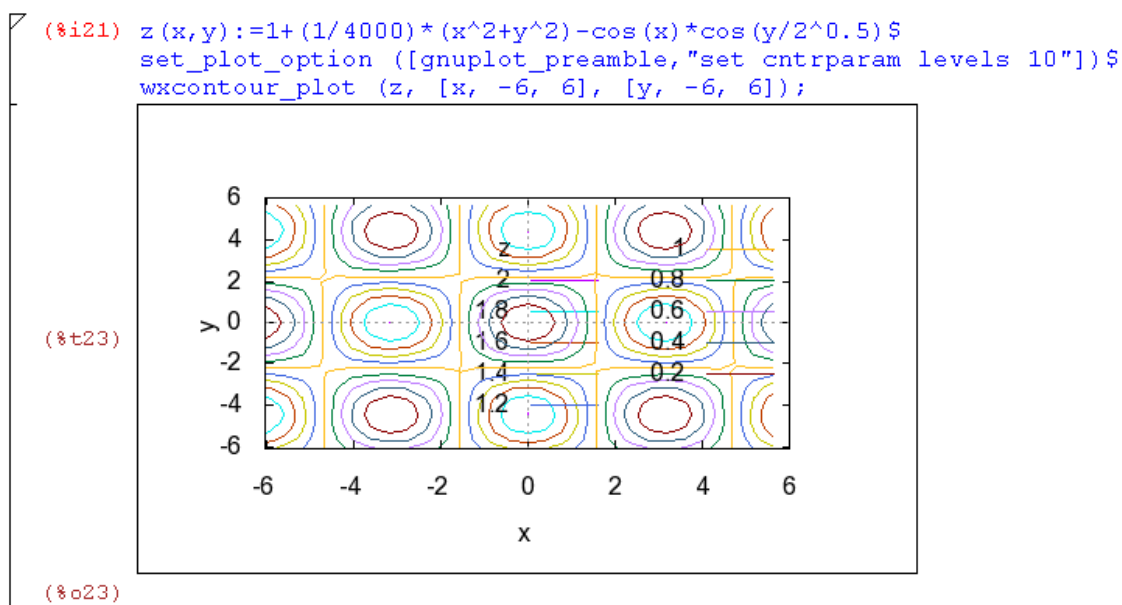
```
(%i18) z(x,y):=1+(1/4000)*(x^2+y^2)-cos(x)*cos(y/2^0.5)$
       set_plot_option([gnuplot_preamble,"set cntrparam levels 10"])$
       contour_plot(z,[x,-6,6],[y,-6,6]);
```

**Código 2.44** Representación grafica de la función de Griewank mediante contornos en Maxima.

En primer lugar se define  $z$  como función de  $x$  e  $y$ . Las funciones gráficas de Maxima admiten multitud de opciones. La manera de introducirlas es mediante la función `set_plot_option()`. En este caso se define el número de niveles equiespaciados que se desea representar en la gráfica de contorno. Esta es una propiedad de `gnuplot`, `cntrparam levels`. Finalmente se llama a la función `contour_plot()`, que tiene como argumentos la función a representar, y en los vectores, la variable y el rango correspondiente.

El resultado obtenido es el siguiente,





**Figura 2.16** Código y representación grafica de la función de Griewank mediante contornos en Maxima mediante la función `wxcontour_plot()`.

Seguidamente se muestra como realizar la representación en 3D de esta misma función.

```
-->x=linspace (-6 , 6 , 100 );

-->y=linspace (-6 , 6 , 100 );

-->function f = z(x,y);
-->f=1+(1/4000)*(x.^2+y.^2)-cos(x)*cos(y/2^0.5);
-->endfunction

-->[X,Y] = ndgrid(x,y);

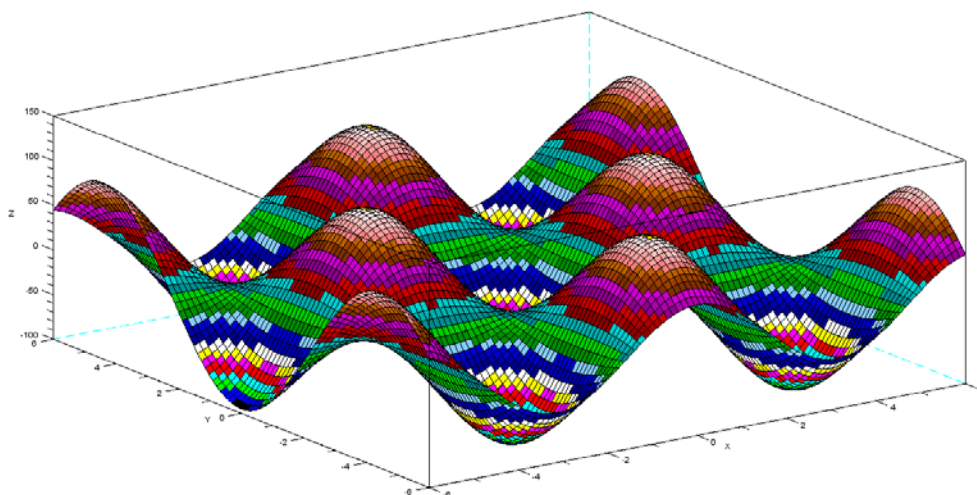
-->Z=z(X,Y);

-->surf(X,Y,Z)
```

**Código 2.45** Representación grafica de la función de Griewank en 3D con Scilab.

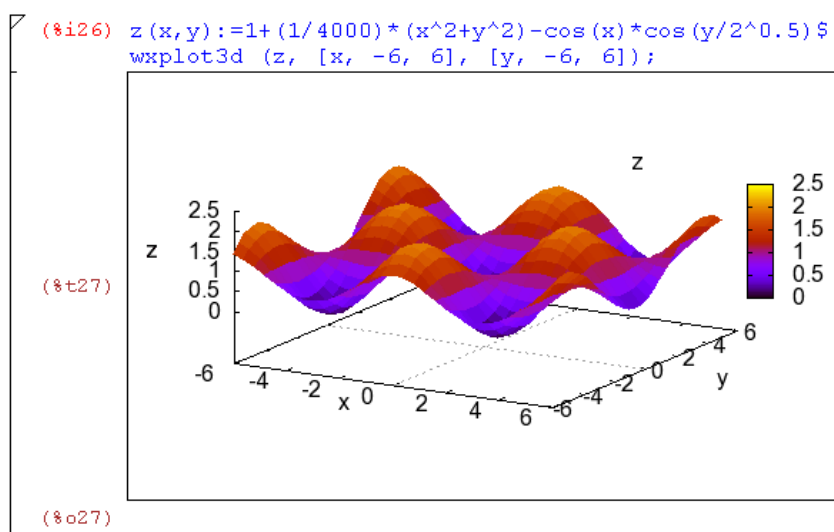
Como se puede ver, el código comienza de la misma manera que en el caso de graficar el contorno. La diferencia entre el contorno y una superficie en 3D, es que se debe definir todos los puntos de la superficie. En primer lugar se realiza un mallado de las variables  $x$  e  $y$ , empleando la función `ndgrid()`, en la que se introduce el vector en el rango de cada variable y se obtiene una matriz de todos los puntos de la superficie espaciados como se definió con la función `linspace()`. Se evalúan los puntos del mallado en la función  $z$  y finalmente se emplea la función `surf()` para obtener una gráfica de la superficie.

El resultado obtenido es el siguiente,



**Figura 2.16** Representación grafica de la función de Griewank mediante una superficie en Scilab.

A continuación se realizará la misma tarea en Maxima. La manera de graficar en Maxima una superficie es muy sencilla. Se puede observar como se ha resuelto el problema de igual modo a como se hizo con el contorno, pero empleando la función `plot3d()`.



**Figura 2.17** Código y representación grafica de la función de Griewank mediante una superficie en Maxima.

## ***EJERCICIOS:***

**Ejercicio 2.7** Representa las siguientes funciones en la misma gráfica, empleando Maxima y Scilab,

$$y_1 = x^2 + 1$$

$$y_2 = 3x - 2$$

$$y_3 = e^{-x}$$

$$x \in [0, 3]$$

**Ejercicio 2.8** Representa gráficamente la conocida curva paramétrica de mariposa empleando Maxima y Scilab.

$$x = \sin(t) \left( e^{\cos(t)} - 2 \cos(4t) - \sin^5 \left( \frac{t}{12} \right) \right)$$

$$y = \cos(t) \left( e^{\cos(t)} - 2 \cos(4t) - \sin^5 \left( \frac{t}{12} \right) \right)$$

**Ejercicio 2.9** Representa gráficamente la superficie de una esfera de radio 1, centrada en el punto (1,1,1).