Introducción al cálculo simbólico con *Maxima*

José A. Alonso Jiménez

Grupo de Lógica Computacional
Dpto. de Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

Sevilla, 1 de Julio de 2010 (versión de 18 de septiembre de 2010)

Índice general

Capítulo 1

Introducción a Maxima

1.1. Ejercicios resueltos

1.1.1. Primeros pasos con Maxima

1.1.1.1 Ejercicio. Calcular el valor de $5 + \left(\frac{2}{3}\right)^{-2}$.

Solución:

```
(%i1)  5+(2/3)^(-2);
(%o1)
```

$$\frac{29}{4}$$

1.1.2. Maxima como calculadora

1.1.2.1 Ejercicio. Calcular el factorial de 100.

Solución:

```
(%i2)  100!;
(%o2)
```

$$933262154439441526816992388562[98 digits]916864000000000000000000000000$$

1.1.2.2 Ejercicio. Observad que no se han mostrado todas las cifras. Obtener todas las cifras cambiando la pantalla 2D.

Solución:

```
(%i3)  set_display(ascii)$
(%i4)  100!;
(%o4)

       933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000000
(%i5)  set_display(xml)$
```

1.1.2.3 Ejercicio. Calcular la raíz cuadrada de 4.

Solución:

```
(%i6)  sqrt(4);
(%o6)
```

$$2$$

1.1.2.4 Ejercicio. Calcular la raíz cuadrada de 5.

Solución:

```
(%i7)  sqrt(5);
(%o7)
```

$$\sqrt{5}$$

1.1.2.5 Ejercicio. Observad que la raíz de 5 la muestra de forma simbólica. Calcular el valor aproximado del resultado anterior

Solución:

```
(%i8)  float(%);
(%o8)
```

$$2{,}23606797749979$$

1.1.2.6 Ejercicio. Asignar a la variable a el valor 5^2.

Solución:

```
(%i9)  a : 5^2;
(%o9)
```

$$25$$

1.1.2.7 Ejercicio. Calcular el valor de $\sqrt{a}+\frac{1}{a}$.

Solución:

```
(%i10)  sqrt(a)+1/a;
(%o10)
```

$$\frac{126}{25}$$

1.1.2.8 Ejercicio. Asignar a la variable c el valor b^2.

Solución:

```
(%i11)  c:b^2;
(%o11)
```

$$b^2$$

1.1.2.9 Ejercicio. Calcular el valor de la raíz cuadrada de c. Observad que Maxima puede trabajar a nivel simbólico.

Solución:

```
(%i12)  sqrt(c);
(%o12)
```

$$|b|$$

1.1.2.10 Ejercicio. Calcular el valor de $a+A$. Observad que Maxima distingue entre letras minúsculas y mayúsculas.

Solución:

```
(%i13)  a+A;
(%o13)
```

$$A+25$$

1.1.2.11 Ejercicio. Calcular los valores de `Exp(0)` y de `exp(0)`.

Solución:

```
(%i14)  Exp(0);
(%o14)
```

$$\mathrm{Exp}\,(0)$$

```
(%i15)  exp(0);
(%o15)
```

$$1$$

1.1.2.12 Ejercicio. Calcular el logaritmo neperiano del número e.

Solución:

```
(%i16)  log(%e);
(%o16)
```

$$1$$

1.1.2.13 Ejercicio. Calcular el valor de la constante π.

Solución:

```
(%i17)  %pi;
(%o17)
```

$$\pi$$

1.1.2.14 Ejercicio. Calcular el valor aproximado de π.

Solución:

```
(%i18)  float(%pi);
(%o18)
```

$$3{,}141592653589793$$

1.1.2.15 Ejercicio. Calcular el valor de π con 50 cifras decimales.

Solución:

```
(%i19)  fpprec : 50$
(%i20)  bfloat(1000*%pi);
(%o20)
```

$$3{,}14159265358979323846264338327950288419716939937510_B \times 10^3$$

```
(%i21)  set_display(ascii)$
(%i22)  bfloat(1000*%pi);
(%o22)
```

$$3{,}14159265358979323846264338327950288419716939937510_B \times 10^3$$

```
(%i23)  set_display(xml)$
```

1.1.3. Los números complejos

1.1.3.1 Ejercicio. Calcular la raíz cuadrada de -1.

Solución:

```
(%i24)  sqrt(-1);
(%o24)
```

$$i$$

1.1.3.2 Ejercicio. Calcular el cuadrado de la unidad imaginaria.

Solución:

```
(%i25)  %i^2;
(%o25)
```

$$-1$$

1.1.3.3 Ejercicio. Asignar a z el número complejo $\dfrac{(1+i)^2}{1-2i}$.

Solución:

```
(%i26)  z : (1+%i)^2/(1-2*%i);
(%o26)
```

$$\frac{(i+1)^2}{1-2\,i}$$

1.1.3.4 Ejercicio. Calcular la forma cartesiana de z.

Solución:

```
(%i27)  rectform(%);
(%o27)
```

$$26$$

1.1.3.5 Ejercicio. Calcular la parte real de z.

Solución:

```
(%i28)  realpart(z);
(%o28)
```

$$-\frac{4}{5}$$

1.1.3.6 Ejercicio. Calcular la parte imaginaria de z.

Solución:

```
(%i29)  imagpart(z);
(%o29)
```

$$\frac{2}{5}$$

1.1.3.7 Ejercicio. Calcular el módulo de z.

Solución:

```
(%i30)  abs(z);
(%o30)
```

$$\frac{2}{\sqrt{5}}$$

1.1.3.8 Ejercicio. Calcular el argumento de z.

Solución:

```
(%i31)  carg(z);
(%o31)
```

$$\arctan 2 + \frac{\pi}{2}$$

1.1.3.9 Ejercicio. Calcular la forma polar de z.

Solución:

```
(%i32)  polarform(z);
(%o32)
```

$$\frac{2\,e^{i\left(\arctan 2+\frac{\pi}{2}\right)}}{\sqrt{5}}$$

1.1.3.10 Ejercicio. Calcular la forma algebraica de z^4.

Solución:

```
(%i33)  rectform(z^4);
(%o33)
```

$$-\frac{384\,i}{625}-\frac{112}{625}$$

1.1.4. Cálculos algebraicos básicos

1.1.4.1 Ejercicio. Borrar los valores de todas las variables.

Solución:

```
(%i34)  remvalue(all)$
```

1.1.4.2 Ejercicio. Asignar a la variable y la expresión $(a+b)^4$.

Solución:

```
(%i35)  y : (a+b)^4;
(%o35)
```

$$(b+a)^4$$

1.1.4.3 Ejercicio. Desarrollar la expresión anterior.

Solución:

```
(%i36)  expand(%);
(%o36)
```

$$35$$

1.1.4.4 Ejercicio. Factorizar la expresión anterior.

Solución:

```
(%i37)  factor(%);
(%o37)
```

$$2^2\,3^2$$

1.1.4.5 Ejercicio. Factorizar la expresión $x^4 - 1$.

Solución:

```
(%i38)  factor(x^4-1);
(%o38)
```

$$(x-1)\,(x+1)\left(x^2+1\right)$$

1.1.4.6 Ejercicio. Sustituir x por $\dfrac{3}{z}$ en la expresión anterior.

Solución:

```
(%i39)  subst(3/z,x,%);
(%o39)
```

$$38$$

1.1.4.7 Ejercicio. Simplificar la expresión anterior.

Solución:

```
(%i40)  ratsimp(%);
(%o40)
```

$$39$$

1.1.5. Ecuaciones y sistemas de ecuaciones

1.1.5.1 Ejercicio. Resolver la ecuación $3x^2 - 17x + 10 = 0$.

Solución:

```
(%i42)  solve(3*x^2-17*x+10=0);
(%o42)
```

$$\left[x = 5, x = \frac{2}{3}\right]$$

1.1.5.2 Ejercicio. Resolver la ecuación $x^2 + 9 = 0$.

Solución:

```
(%i43)  solve(x^2+9=0);
(%o43)
```

$$[x = -3\,i, x = 3\,i]$$

1.1.5.3 Ejercicio. Resolver la ecuación $1 + z + z^2 = 0$.

Solución:

```
(%i44)  solve(1+z+z^2=0);
(%o44)
```

$$\left[z = -\frac{\sqrt{3}\,i + 1}{2}, z = \frac{\sqrt{3}\,i - 1}{2}\right]$$

1.1.5.4 Ejercicio. Resolver la ecuación $ax^2 + bx + c = 0$.

Solución:

```
(%i45)  solve(a*x^2+b*x+c=0,x);
(%o45)
```

$$\left[x = -\frac{\sqrt{b^2 - 4\,a\,c} + b}{2\,a}, x = \frac{\sqrt{b^2 - 4\,a\,c} - b}{2\,a}\right]$$

1.1.5.5 Ejercicio. Borrar el valor de la variable y.

Solución:

```
(%i46)  kill(y)$
```

1.1.5.6 Ejercicio. Asignar a la variable S el sistema de ecuaciones

$$\begin{cases} mx + y & = 1, \\ x - m^2 y & = m \end{cases}$$

Solución:

```
(%i47)  S : [m*x+y=1, x-m^2*y=m];
(%o47)
```

$$\left[y + m\,x = 1, x - m^2\,y = m\right]$$

1.1.5.7 Ejercicio. Resolver el sistema S respecto de las variables x e y.

Solución:

```
(%i48)  solve(S,[x,y]);
(%o48)
```

$$\left[\left[x = \frac{m}{m^2 - m + 1}, y = -\frac{m - 1}{m^2 - m + 1}\right]\right]$$

1.1.5.8 Ejercicio. Calcular una raíz de la ecuación $\cos(x) = x$ entre 0 y π.

Solución:

```
(%i49)  find_root(cos(x)=x,x,0,%pi);
(%o49)
```

0,73908513321516

1.1.6. Gráficas de funciones

1.1.6.1 Ejercicio. Dibujar las gráficas de las funciones

$$\begin{cases} y & = \cos(x) \\ y & = x \end{cases}$$

Solución:

Vamos a hacer las gráficas cambiando los rangos de las variables.

En primer lugar, con la x entre -5 y 5. La entrada es

```
(%i50)  plot2d([cos(x), x], [x,-5,5])$
```

y la gráfica está en la figura 1.1.

En segundo lugar, con la x entre -2 y 2. La entrada es

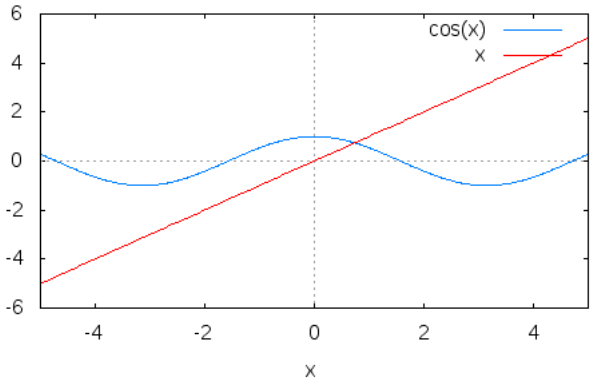

Figura 1.1: Gráfica de cos(x) y de x

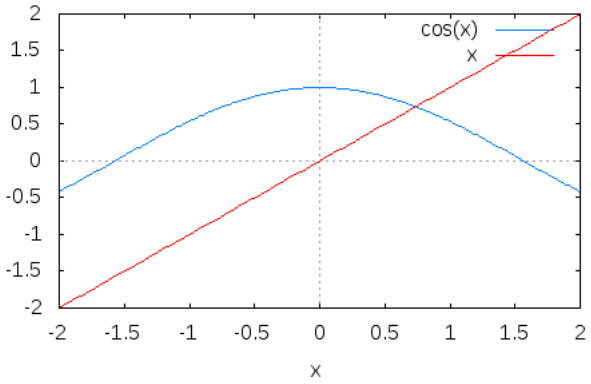

Figura 1.2: Gráfica de cos(x) y de x con x entre -2 y 2

```
(%i51)  plot2d([cos(x), x], [x,-2,2])$
```

y la gráfica está en la figura 1.2.

En tercer lugar, con la x y la y entre 0 y 1. La entrada es

```
(%i52)  plot2d([cos(x), x], [x,0,1],[y,0,1])$
```

y la gráfica está en la figura 1.3.

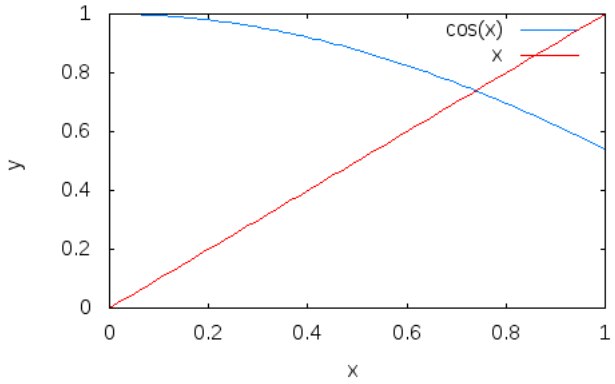

Figura 1.3: Gráfica de cos(x) y de x con x e y entre 0 y 1

1.1.7. Obtención de la ayuda para una función

1.1.7.1 Ejercicio. Obtener la ayuda de la función `is`.

Solución:

```
(%i53)  ? is;
 – Function: is (<expr>)
     Attempts to determine whether the predicate <expr> is provable
     from the facts in the 'assume' database.
```

1.1.7.2 Ejercicio. Comprobar si la raíz cuadrada de t^2 es igual a t.

Solución:

```
(%i54)  is(sqrt(t^2)=t);
(%o54)
```

false

1.1.7.3 Ejercicio. Comprobar si la raíz cuadrada de t^2 es igual al valor absoluto de t.

Solución:

```
(%i55)  is(sqrt(t^2)=abs(t));
(%o55)
                                true
```

1.1.7.4 Ejercicio. Comprobar si t es positivo.

Solución:

```
(%i56)  is(t>0);
(%o56)
                              unknown
```

1.2. Ejercicios propuestos

1.2.1. Ejercicio 1: Cálculo aritmético

1.2.1.1 Ejercicio. Definir la constante a igual a

$$\sqrt[3]{20+14\sqrt{2}}+\sqrt[3]{20-14\sqrt{2}}$$

Solución:

```
(%i1)  a : (20+14*sqrt(2))^(1/3) + (20-14*sqrt(2))^(1/3);
(%o1)
```

$$\left(7\,2^{\frac{3}{2}}+20\right)^{\frac{1}{3}}+\left(20-7\,2^{\frac{3}{2}}\right)^{\frac{1}{3}}$$

1.2.1.2 Ejercicio. Calcular el valor numérico de a. ¿A qué entero se aproxima?
Nota: Usar la función `round`.

Solución:

```
(%i2)  round(float(a));
(%o2)
```

$$4$$

1.2.1.3 Ejercicio. Confirmar la conjetura con la orden `is`.

Solución: Podemos confirmar que se aproxima

```
(%i3)  is(abs(a-4)<10^-9);
(%o3)
```

true

No podemos confirmar la igualdad.

```
(%i4)  is(a=4);
(%o4)
```

false

1.2.2. Ejercicio 2: Cálculo trigonométrico

1.2.2.1 Ejercicio. Escribir el número

$$\left(\operatorname{sen}\frac{\pi}{3}+\cos\frac{\pi}{3}\right)^9$$

en la forma $a+bc^d$, donde a, b, c y d son números racionales.
Nota: Cambiar el valor de la variable `%piargs` a `true` y usar `radcan` para la simplificación de radicales.

Solución:

```
(%i5)  %piargs : true$
(%i6)  radcan((sin(%pi/3)+cos(%pi/3))^9);
(%o6)
```

$$\frac{17\,3^{\frac{5}{2}}+265}{32}$$

```
(%i7)  %piargs : false$
```

1.2.2.2 Ejercicio. Asignar a las variables a, b, c y d los valores obtenidos en el apartado anterior y calcular el valor de

$$\left(\operatorname{sen}\frac{\pi}{3}+\cos\frac{\pi}{3}\right)^9-a+bc^d$$

Solución:

```
(%i8)  %piargs : true$
(%i9)  a:265/32$
(%i10)  b:17/32$
(%i11)  c:3$
(%i12)  d:5/2$
(%i13)  radcan(radcan((sin(%pi/3)+cos(%pi/3))^9)-(a+b*c^d));
(%o13)
```

$$0$$

```
(%i14)  %piargs : false$
```

1.2.3. Ejercicio 3: Cálculo con precisión determinada

1.2.3.1 Ejercicio. Calcular la cifra 49 del número π.

Solución:

```
(%i38)  fpprec : 51;
(%o38)
```

$$51$$

```
(%i39)  bfloat(%pi);
(%o39)
```

$$3{,}14159265358979323846264338327950288419716939937511_B \times 10^0$$

Por tanto, la cifra 49 de π es 1.

Nota: Otra forma de calcularlo es mediante

```
floor(mod(bfloat(%pi)*10**49,10));
```

1.2.4. Ejercicio 4: Raices y factorización de un polinomio

1.2.4.1 Ejercicio. Asignarle a p el polinomio $x^4 - x^3 - 7x^2 - 8x - 6$.

Solución:

```
(%i17)  p : x^4-x^3-7*x^2-8*x-6;
(%o17)
```

$$x^4 - x^3 - 7\,x^2 - 8\,x - 6$$

1.2.4.2 Ejercicio. Calcular las raíces reales de p.

Solución:

```
(%i18)  solve(p);
(%o18)
```

$$\left[x = 1 - \sqrt{7}, x = \sqrt{7} + 1, x = -\frac{\sqrt{3}\,i + 1}{2}, x = \frac{\sqrt{3}\,i - 1}{2}\right]$$

Por tanto, las raíces reales de p son $1 + \sqrt{7}$ y $1 - \sqrt{7}$.

1.2.4.3 Ejercicio. Factorizar al máximo el polinomio p.

Solución:

```
(%i19)  (x-sqrt(7)-1)*(x+sqrt(7)-1)*
        (x-(sqrt(3)*%i-1)/2)*(x+(sqrt(3)*%i+1)/2);
(%o19)
```

$$\left(x-\sqrt{7}-1\right)\left(x+\sqrt{7}-1\right)\left(x-\frac{\sqrt{3}\,i-1}{2}\right)\left(x+\frac{\sqrt{3}\,i+1}{2}\right)$$

```
(%i20)  ratsimp(%);
(%o20)
```

$$x^4-x^3-7\,x^2-8\,x-6$$

1.2.5. Ejercicio 5: Cálculo con números complejos

1.2.5.1 Ejercicio. Asignar a z el número complejo

$$\left(\frac{1-i\sqrt{3}}{1+i}\right)^{20}$$

Solución:

```
(%i21)  z : ((1-%i*sqrt(3))/(1+%i))^20;
(%o21)
```

$$\frac{\left(1-\sqrt{3}\,i\right)^{20}}{\left(i+1\right)^{20}}$$

1.2.5.2 Ejercicio. Calcular la parte real y la parte imaginaria de z.
Nota: Usar `radcan` para simplificarla.

Solución:

```
(%i22)  realpart(z);
(%o22)
```

$$512$$

```
(%i23)  imagpart(z);
(%o23)
```

$$-\frac{-40\,3^{\frac{23}{2}}+5168\,3^{\frac{17}{2}}-25840\,3^{\frac{15}{2}}+167960\,3^{\frac{11}{2}}-142120\,3^{\frac{9}{2}}-5168\,3^{\frac{7}{2}}+380\,3^{\frac{5}{2}}-20\,\sqrt{3}}{1024}$$

```
(%i24)  radcan(imagpart(z));
(%o24)
```

$$512\sqrt{3}$$

1.2.5.3 Ejercicio. Calcular el módulo y el argumento de z.

Solución:

```
(%i25)  radcan(abs(z));
(%o25)
```

$$1024$$

```
(%i26)  radcan(carg(z));
(%o26)
```

$$\arctan\sqrt{3}$$

1.2.6. Ejercicio 6: Gráficos para conjeturar soluciones y su cálculo

1.2.6.1 Ejercicio. Con la ayuda de la representación gráfica, conjeturar el número de soluciones de $\text{sen}(x) = 1 - x^4$.

Solución: La entrada para obtener la gráfica es

```
(%i27)  plot2d([sin(x),1-x^4], [x,-2,2],[y,-2,2])$
```

y la gráfica está en la figura 1.4.

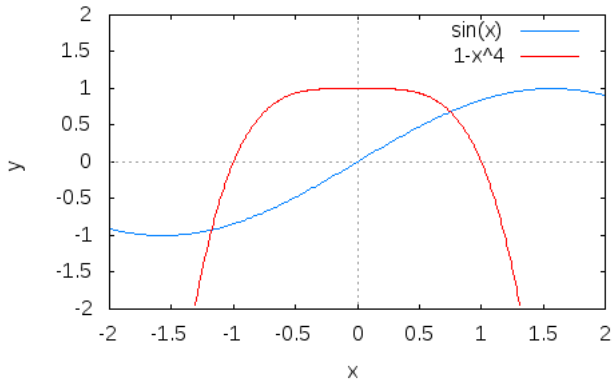

Figura 1.4: Gráfica para conjeturar solución

Tiene dos raíces: una entre -1.5 y -1 y la otra entre 0.5 y 1.

1.2.6.2 Ejercicio. Dar una aproximación de cada solución.

Solución:

```
(%i28)  find_root(sin(x)=1-x^4, x, -1.5, -1);
(%o28)
```

$$-1{,}177703485350082$$

```
(%i29)  find_root(sin(x)=1-x^4, x, 0.5, 1);
(%o29)
```

$$0{,}75080729992228$$

1.2.7. Ejercicio 7: Solución de sistemas lineales

1.2.7.1 Ejercicio. Borrar el valor de todas las variables.

Solución:

```
(%i30)  remvalue(all)$
```

1.2.7.2 Ejercicio. Resolver el sistema lineal

$$\begin{cases} x + 2y &= 1, \\ 2x + 4y &= 2 \end{cases}$$

Nota: Usar el menú "Ecuaciones / Resolver sistema lineal".

Solución:

```
(%i31)  linsolve([x+2*y=1, 2*x+4*y=2], [x,y]);
  solve: dependent equations eliminated: (2)
(%o31)
```

$$[x = 1 - 2\ \%\mathrm{r1}, y = \%\mathrm{r1}]$$

1.2.7.3 Ejercicio. Resolver el sistema lineal

$$\begin{cases} x + 2y &= 1, \\ 2x + 4y &= 7 \end{cases}$$

Solución:

```
(%i32)  linsolve([x+2*y=1, 2*x+4*y=7], [x,y]);
(%o32)
                                  []
```

1.2.7.4 Ejercicio. Resolver el sistema lineal

$$\begin{cases} x+z & = y, \\ 2ax-y & = 2a^2, \\ y-2z & = 2 \end{cases}$$

en función del parámetro a.

Solución:

```
(%i33)  linsolve([x+z=y, 2*a*x-y=2*a^2, y-2*z=2],[x,y,z]);
(%o33)
                          [x = a + 1, y = 2 a, z = a - 1]
```

Capítulo 2

Funciones de una variable

2.1. Ejercicios resueltos

2.1.1. Definición de funciones

2.1.1.1 Ejercicio. Definir, usando el operador `:=`, la función

$$f(x) = \text{sen}(x) - x$$

Solución:

```
(%i1)  f(x):=sin(x)-x;
(%o1)
```

$$f(x) := \sin x - x$$

2.1.1.2 Ejercicio. Definir, usando `lambda`, la función g que a un x le asigna $\text{sen}(x)$.

Solución:

```
(%i2)  g:lambda([x],sin(x));
(%o2)
```

$$\lambda([x], \sin x)$$

2.1.1.3 Ejercicio. Definir, usando `define`, la función

$$h(x) = \sqrt{1+x^2} - 2x$$

Solución:

```
(%i3)  define(h(x),sqrt(1+x^2)-2*x);
```

```
(%o3)
```

$$h(x) := \sqrt{x^2+1} - 2x$$

2.1.2. Límites y asíntotas

2.1.2.1 Ejercicio. Calcular el límite de $f(x)$ cuando x tiende a infinito.

Solución:

```
(%i4)  limit(f(x),x,inf);
(%o4)
```

$$-\infty$$

2.1.2.2 Ejercicio. Calcular el límite de $\dfrac{g(x)}{x}$ cuando x tiende a cero.

Solución:

```
(%i5)  limit(g(x)/x,x,0);
(%o5)
```

$$1$$

2.1.2.3 Ejercicio. Calcular el límite de $h(x)$ cuando x tiende a menos infinito.

Solución:

```
(%i6)  limit(h(x),x,minf);
(%o6)
```

$$\infty$$

2.1.2.4 Ejercicio. Calcular el límite de $\dfrac{1}{1-t^2}$ cuando t tiende a 1 por la izquierda.

Solución:

```
(%i7)  'limit(1/(1-t^2),t,1,minus)=limit(1/(1-t^2),t,1,minus);
(%o7)
```

$$\lim_{t\uparrow 1} \frac{1}{1-t^2} = \infty$$

2.1.2.5 Nota. Cuando una expresión va precedida del operador `'`, significa que dicha expresión no se evalúa.

2.1.2.6 Ejercicio. Calcular el límite de $\dfrac{1}{1-t^2}$ cuando t tiende a 1 por la derecha.

Solución:

```
(%i8)  'limit(1/(1-t^2),t,1,plus)=limit(1/(1-t^2),t,1,plus);
(%o8)
```

$$\lim_{t\downarrow 1} \frac{1}{1-t^2} = -\infty$$

2.1.2.7 Ejercicio. Calcular los 2 primeros términos del polinomio de Taylor de $h(x)$ en un entorno del infinito y, a partir de él, el límite de $h(x)$ cuando x tiende a infinito.

Solución:

```
(%i9)  taylor(h(x),x,inf,2);
(%o9)
```

$$-x+\frac{1}{2\,x}+\cdots$$

```
(%i10)  limit(%, x, inf);
(%o10)
```

$$-\infty$$

2.1.3. Derivación

2.1.3.1 Ejercicio. Calcular la derivada de $h(x)$ respecto de x.

Solución:

```
(%i11)  'diff(h(x),x)=diff(h(x),x);
(%o11)
```

$$\frac{d}{d\,x}\left(\sqrt{x^2+1}-2\,x\right)=\frac{x}{\sqrt{x^2+1}}-2$$

2.1.3.2 Ejercicio. Definir, usando `define`, la función $p(x)$ como la derivada de $h(x)$ respecto de x.

Solución:

```
(%i12)  define(p(x),diff(h(x),x));
(%o12)
```

$$p(x) := \frac{x}{\sqrt{x^2+1}} - 2$$

Se puede definir de manera simplificada:

```
(%i13)  define(p(x),ratsimp(diff(h(x),x)));
(%o13)
```

$$p(x) := -\frac{2\sqrt{x^2+1} - x}{\sqrt{x^2+1}}$$

2.1.3.3 Ejercicio. Calcular la segunda derivada de la función f respecto de x.

Solución:

```
(%i14)  diff(f(x),x,2);
(%o14)
```

$$-\sin x$$

2.1.3.4 Ejercicio. Calcular la ecuación reducida de la tangente a la curva definida por h, en el punto de abcisa $x = 2$.

Solución:

```
(%i21)  y=expand(taylor(h(x),x,2,1));
(%o21)
```

$$y = \frac{2x}{\sqrt{5}} - 2x + \sqrt{5} - \frac{4}{\sqrt{5}}$$

```
(%i22)  y=factor(expand(taylor(h(x),x,2,1)));
(%o22)
```

$$y = -\frac{2\sqrt{5}x - 2x - 1}{\sqrt{5}}$$

2.1.4. Funciones definidas a trozos

2.1.4.1 Ejercicio. Definir la función

$$d(x) = \begin{cases} 0, & \text{si } x < 0 \\ x^3, & \text{si } 0 \leq x \leq 1 \\ 1, & \text{si } x > 1 \end{cases}$$

Solución:

```
(%i25)  d(x) := if x<0 then 0
                elseif x<=1 then x^3
                else 1;
(%o25)
```

$$d(x) := \textbf{if } x < 0 \textbf{ then } 0 \textbf{ else } x^3$$

2.1.5. Representación gráfica

2.1.5.1 Ejercicio. Dibujar la gráfica de $d(x)$ para x entre -2 y 4 e y entre -1 y 2.

Solución: La entrada es

```
(%i26)  plot2d(d(x), [x,-2,4], [y,-1,2])$
```

y la gráfica está en la figura 2.1.

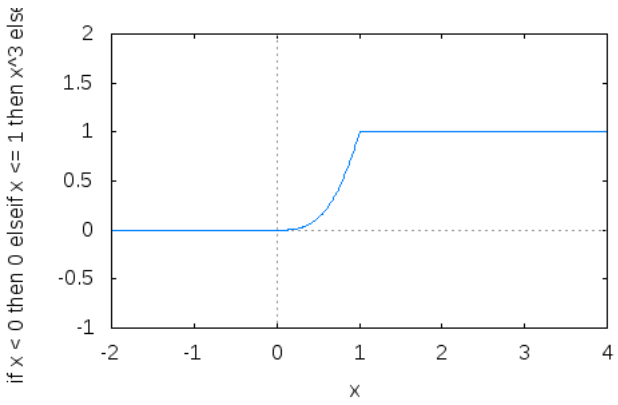

Figura 2.1: Gráfica de d(x)

2.1.5.2 Ejercicio. Dibujar la gráfica de $h(x)$ para x entre -2 y 4 e y entre -5 y 1, usando retícula.

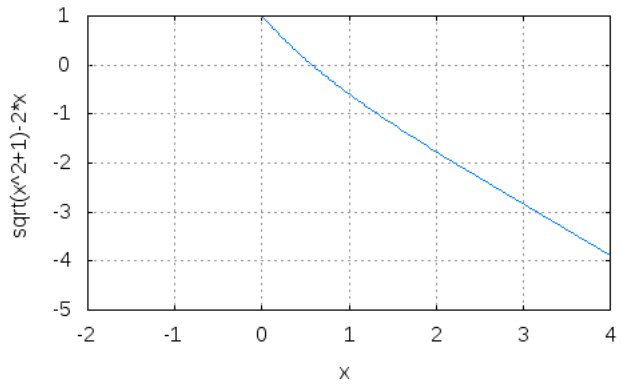

Figura 2.2: Gráfica de h(x)

Solución: La entrada es

```
(%i27)  plot2d(h(x), [x,-2,4], [y,-5,1],
          [gnuplot_preamble, "set grid "])$
```

y la gráfica está en la figura 2.2.

2.1.5.3 Ejercicio. Dibujar las gráficas de $g(x)$, x y $x - \dfrac{x^3}{6}$ para x entre $-\pi$ y π e y entre $-1{,}5$ y $1{,}5$, mostrando los ejes de coordenadas.

Solución: La entrada es

```
(%i28)  plot2d([g(x),x,x-x^3/6] ,
          [x,-%pi,%pi] ,
          [y,-1.5,1.5] ,
          [gnuplot_preamble, "set zeroaxis"])$
```

y la gráfica está en la figura 2.3.

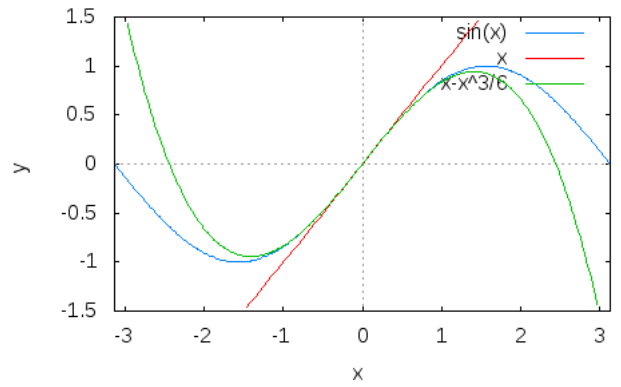

Figura 2.3: Gráfica de 3 funciones

2.1.5.4 Ejercicio. Dibujar las gráficas de $g(x)$, x y $x - \dfrac{x^3}{6}$ para x entre $-\pi$ y π e y entre $-1{,}5$ y $1{,}5$, mostrando los ejes de coordenadas y retículas.

Solución:

```
(%i29)  plot2d([g(x),x,x-x^3/6] ,
     [x,-%pi,%pi] ,
     [y,-1.5,1.5] ,
     [gnuplot_preamble, "set zeroaxis; set grid"])$
```

y la gráfica está en la figura 2.4.

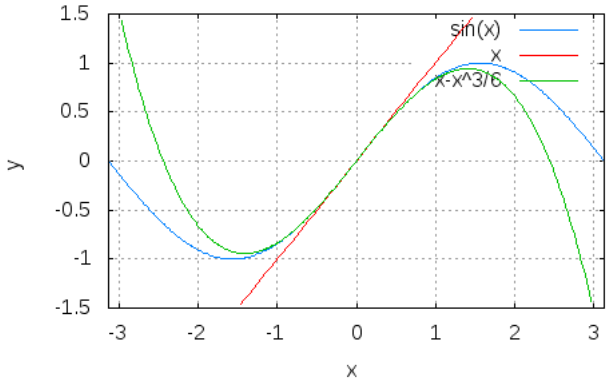

Figura 2.4: Gráfica de 3 funciones con retícula

2.2. Ejercicios propuestos con soluciones

2.2.1. Ejercicio 1: Funciones definidas por partes

2.2.1.1 Ejercicio. Sean a y b dos números reales. Se considera la función f definida sobre $\mathbb{R}$ por

$$f(x) = \begin{cases} \dfrac{e^x - 1}{x}, & \text{si } x > 0 \\ ax + b, & \text{si } x \leq 0 \end{cases}$$

Definir la función f usando el condicional `if ... then ... else`

Solución:

```
(%i1)  f(x) := if x>0 then (%e^x-1)/x else a*x+b;
(%o1)
```

$$f(x) := \textbf{if } x > 0 \textbf{ then } \frac{e^x - 1}{x} \textbf{ else } a\,x + b$$

2.2.1.2 Ejercicio. El procedimiento `limit` no puede evaluar comandos del tipo `if...then` Por ello, para determinar el límite de f en cero por la derecha se necesita precisar en qué intervalo se encuentra x. Esto puede hacerse con la función `assume`.

Escribir la expresión `assume(x>0)`, después calcular el límite de f en cero por la derecha. Se puede eliminar la hipótesis sobre x por `forget(x>0)`

Solución:

```
(%i2)  assume(x>0)$
(%i3)  limit(f(x), x, 0, plus);
(%o3)
```

$$1$$

```
(%i4)  forget(x>0)$
```

2.2.1.3 Ejercicio. Deducir el valor de b para el que f es continua en $\mathbb{R}$.

Solución:

```
(%i39)  assume(x<=0)$
(%i40)  limit(f(x), x, 0, minus);
(%o40)
```

$$b$$

```
(%i41)  forget(x<=0)$
```

Por tanto, $b = 1$.

2.2.1.4 Ejercicio. Calcular la derivada de f en cero por la derecha.

Solución:

```
(%i42)  assume(x>0)$
(%i43)  define(dfp(x),diff(f(x),x,1));
(%o43)
```

$$\mathrm{dfp}\,(x) := \frac{e^x}{x} - \frac{e^x - 1}{x^2}$$

```
(%i44)  limit(dfp(x), x, 0, plus);
(%o44)
```

$$\frac{1}{2}$$

```
(%i45)  forget(x>0)$
```

2.2.1.5 Ejercicio. Calcular el valor de a para el que f es derivable en cero.

Solución:

```
(%i10)  assume(x<=0)$
(%i11)  define(dfm(x),diff(f(x),x,1));
(%o11)
```

$$\mathrm{dfm}\,(x) := a$$

Por tanto, $a = 1/2$.

2.2.2. Ejercicio 2: Estudio de funciones

2.2.2.1 Ejercicio. Definir la función g tal que $g(x) = 2x - \sqrt{1+x^2}$.

Solución:

```
(%i12)  g(x) := 2*x-sqrt(1+x^2);
(%o12)
```

$$g\,(x) := 2\,x - \sqrt{1+x^2}$$

2.2.2.2 Ejercicio. Calcular los límites de g en $+\infty$ y en $-\infty$.

Solución:

```
(%i13)  limit(g(x), x, inf);
(%o13)
```

$$\infty$$

```
(%i14)  limit(g(x), x, minf);
(%o14)
```

$$-\infty$$

2.2.2.3 Ejercicio. Dibujar la gráfica de la función g.

Solución: La entrada es

```
(%i15)  plot2d([g(x)], [x,-50,150])$
```

y la gráfica está en la figura 2.5.

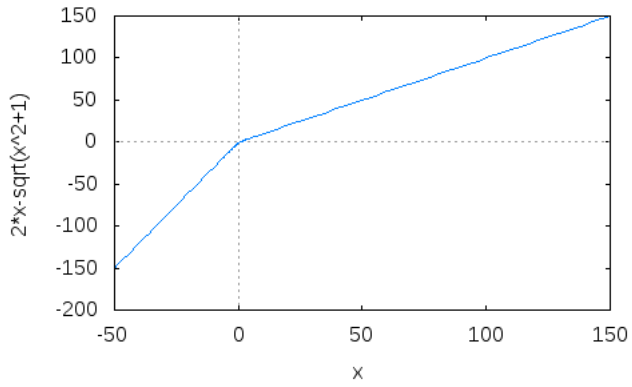

Figura 2.5: Gráfica de la función g

2.2.2.4 Ejercicio. Calcular $g'(x)$.

Solución:

```
(%i16)  define(dg(x),diff(g(x),x,1));
(%o16)
```

$$\mathrm{dg}\,(x) := 2 - \frac{x}{\sqrt{x^2+1}}$$

2.2.2.5 Ejercicio. Resolver la ecuación $g(x) = 0$.

Solución:

```
(%i17)  r : find_root(g(x), x, -50, 150);
(%o17)
```

$$0{,}57735026918963$$

2.2.2.6 Ejercicio. Determinar los intervalos de crecimiento de g.

Solución:

```
(%i18)  define(dg2(x),diff(g(x),x,2));
(%o18)
```

$$\mathrm{dg2}\,(x) := \frac{x^2}{(x^2+1)^{\frac{3}{2}}} - \frac{1}{\sqrt{x^2+1}}$$

```
(%i19)  radcan(%);
(%o19)
```

$$\mathrm{dg2}\,(x) := -\frac{\sqrt{x^2+1}}{x^4+2\,x^2+1}$$

```
(%i20)  is(dg2(x)<0);
(%o20)
```

true

Por tanto, g es creciente en todo $\mathbb{R}$.

2.2.2.7 Ejercicio. Calcular las ecuaciones reducidas de las asíntotas de g.

Solución: En primer lugar, vamos a aproximarnos a $+\infty$

```
(%i21)  g(1000),numer;
(%o21)
```

$$999{,}999500000125$$

```
(%i22)  g(10000),numer;
(%o22)
```

$$9999{,}999949999999$$

Veamos la aproximación entre g y la recta $y = ax + b$ en $+\infty$:

```
(%i23)  limit(g(x)/(a*x+b), x, inf);
```

```
(%o23)
```
$$\frac{1}{a}$$

Por tanto, la asíntota es $y = x$. En efecto,

```
(%i24)  limit(g(x)/x, x, inf);
(%o24)
```
$$1$$

En segundo lugar, vamos a aproximarnos a $-\infty$

```
(%i25)  g(-1000),numer;
(%o25)
```
$$-3000{,}000499999875$$

```
(%i26)  g(-10000),numer;
(%o26)
```
$$-30000{,}00005$$

Veamos la aproximación entre g y la recta $y = ax + b$ en $-\infty$:

```
(%i27)  limit(g(x)/(a*x+b), x, minf);
(%o27)
```
$$\frac{3}{a}$$

Por tanto, la asíntota es $y = 3x$. En efecto,

```
(%i28)  limit(g(x)/(3*x), x, minf);
(%o28)
```
$$1$$

2.2.3. Ejercicio 3: Desarrollos trigonométricos

2.2.3.1 Ejercicio. Desarrollar $\cos(3t)$ en función de $\cos t$.

Solución:

```
(%i29)  trigexpand(cos(3*t));
(%o29)
```

$$(\cos t)^3 - 3\cos t\,(\sin t)^2$$

```
(%i30)  trigsimp(%);
(%o30)
```

$$4\,(\cos t)^3 - 3\cos t$$

2.2.3.2 Ejercicio. Desarrollar $\cos(4t)$ en función de $\cos(t)$.

Solución:

```
(%i31)  trigexpand(cos(4*t));
(%o31)
```

$$(\sin t)^4 - 6\,(\cos t)^2\,(\sin t)^2 + (\cos t)^4$$

```
(%i32)  trigsimp(%);
(%o32)
```

$$8\,(\cos t)^4 - 8\,(\cos t)^2 + 1$$

2.2.3.3 Ejercicio. Desarrollar $\cos(5t)$ en función de $\cos(t)$.

Solución:

```
(%i33)  trigexpand(cos(5*t));
(%o33)
```

$$5\cos t\,(\sin t)^4 - 10\,(\cos t)^3\,(\sin t)^2 + (\cos t)^5$$

```
(%i34)  trigsimp(%);
(%o34)
```

$$16\,(\cos t)^5 - 20\,(\cos t)^3 + 5\cos t$$

2.2.3.4 Ejercicio. Determinar los polinomios T_n de la variable x tales que para todo $t \in \mathbb{R}$, $\cos(nt) = T_n(\cos t)$ para $n \in \{3,4,5\}$.

Solución:

```
(%i35)  T3(x) :=  4*x^3- 3*x ;
(%o35)
```

$$T3(x) := 4x^3 - 3x$$

```
(%i36)  T4(x) :=  8*x^4- 8*x^2+1 ;
(%o36)
```

$$T4(x) := 8x^4 - 8x^2 + 1$$

```
(%i37)  T5(x) := 16*x^5-20*x^3+5*x ;
(%o37)
```

$$T5(x) := 16x^5 - 20x^3 + 5x$$

2.2.3.5 Ejercicio. Representar las funciones T_3, T_4 y T_5 en la misma gráfica.

Solución: La entrada es

```
(%i38)  plot2d([T3(x), T4(x), T5(x)], [x,-1,1])$
```

y la gráfica está en la figura 2.6.

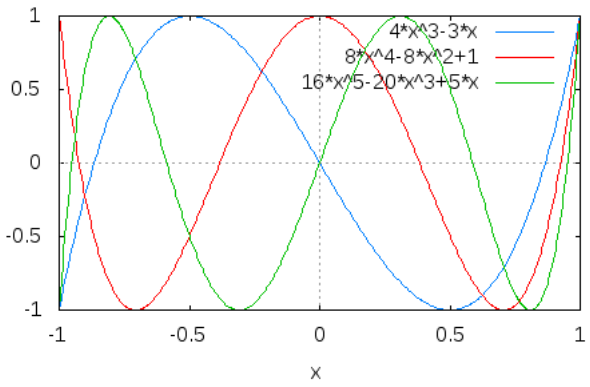

Figura 2.6: Gráfica de las funciones T_3, T_4 y T_5

Capítulo 3

Aritmética

3.1. Ejercicios resueltos

3.1.1. Divisores y división entera

3.1.1.1 Ejercicio. Asignarle a la variable a el valor 16,800 y a la variable b el valor -990.

Solución:

```
(%i1)  a:16800$
(%i2)  b:-990$
```

3.1.1.2 Ejercicio. Calcular el conjunto de los divisores positivos de b.

Solución:

```
(%i3)  divisors(b);
(%o3)
```

$$\{1, 2, 3, 5, 6, 9, 10, 11, 15, 18, 22, 30, 33, 45, 55, 66, 90, 99, 110, 165, 198, 330, 495, 990\}$$

3.1.1.3 Ejercicio. Calcular la suma de los divisores de b.

Solución:

```
(%i4)  divsum(b);
(%o4)
```

$$2808$$

3.1.1.4 Ejercicio. Calcular el cociente y el resto de la división euclídea de a entre b.

Solución:

```
(%i5)  divide(a,b);
(%o5)
```

$$[-16, 960]$$

3.1.1.5 Ejercicio. Asignarle a las variables q y r el cociente y el resto de la división euclídea de a entre b.

Solución:

```
(%i6)  [q,r]:%;
(%o6)
```

$$[-16, 960]$$

3.1.1.6 Ejercicio. Comprobar, usando `is`, que el dividendo (a) es igual al divisor (b) por el cociente (q) más el resto (r).

Solución:

```
(%i7)  is(a=b*q+r);
(%o7)
```

true

3.1.1.7 Ejercicio. Calcular el resto de dividir 17 entre 5.

Solución:

```
(%i8)  mod(17,5);
(%o8)
```

2

3.1.2. Máximo común divisor y mínimo común múltiplo

3.1.2.1 Ejercicio. Calcular el máximo común divisor de a y b.

Solución:

```
(%i9)  gcd(a,b);
(%o9)
```

$$30$$

3.1.2.2 Ejercicio. Calcular el mínimo común múltiplo de a y b.

Solución:

```
(%i10)  lcm(a,b);
(%o10)
```

$$\mathrm{lcm}\,(16800, -990)$$

3.1.3. Números primos

3.1.3.1 Ejercicio. Comprobar si los números 101 y 1001 son primos.

Solución:

```
(%i11)  primep(101);
(%o11)
```

$$\textbf{true}$$

```
(%i12)  primep(1001);
(%o12)
```

$$\textbf{false}$$

3.1.3.2 Ejercicio. Calcular el mayor primo menor que 1001.

Solución:

```
(%i13)  prev_prime(1001);
(%o13)
```

$$997$$

3.1.3.3 Ejercicio. Calcular el menor primo mayor que 1001.

Solución:

```
(%i14)  next_prime(1001);
(%o14)
```

$$1009$$

3.1.3.4 Ejercicio. Descomponer 2520 en factores primos.

Solución:

```
(%i15)  factor(2520);
(%o15)
```

$$2^3\,3^2\,5\,7$$

3.1.3.5 Ejercicio. Calcular la descomposición de 2520 en factores primos.

Solución:

```
(%i16)  ifactors(2520);
(%o16)
```

$$[[2,3],[3,2],[5,1],[7,1]]$$

3.1.3.6 Ejercicio. Descomponer $2^{67}-1$ en factores primos.

Solución:

```
(%i17)  factor(2^67-1);
(%o17)
```

$$193707721\,761838257287$$

3.1.4. Programación básica

3.1.4.1 Ejercicio. Escribir un programa para calcular la descomposición en factores primos de los números $100, 105, \ldots, 125$.

Solución:

```
(%i18)  for i from 100 step 5 thru 125 do (print(i,"=",factor(i)))$
```

escribe

$$\begin{aligned} 100 &= 2^2\,5^2 \\ 105 &= 3\,5\,7 \\ 110 &= 2\,5\,11 \\ 115 &= 5\,23 \\ 120 &= 2^3\,3\,5 \\ 125 &= 5^3 \end{aligned}$$

3.1.4.2 Ejercicio. Un número entero n es perfecto si la suma de sus divisores positivos distintos de n es igual a n.

Escribir un programa que calcule los números perfectos menores que 500.

Solución:

```
(%i19)  for k from 1 thru 500 do
            (if k=divsum(k)-k then print(k, "es perfecto"))$
```

escribe

6 es perfecto
28 es perfecto
496 es perfecto

3.2. Ejercicios propuestos con soluciones

3.2.1. Ejercicio 1: Divisores

3.2.1.1 Ejercicio. Asignarle a la variable a el valor 2.460 y a la b el 3.030.

Solución:

```
(%i1)  a:2460$
(%i2)  b:3030$
```

3.2.1.2 Ejercicio. Calcular el conjunto D_1 de los divisores positivos de a.

Solución:

```
(%i3)  D1:divisors(a);
(%o3)
```

$$\{1, 2, 3, 4, 5, 6, 10, 12, 15, 20, 30, 41, 60, 82, 123, 164, 205, 246, 410, 492, 615, 820, 1230, 2460\}$$

3.2.1.3 Ejercicio. Calcular el conjunto D_2 de los divisores positivos de b.

Solución:

```
(%i4)  D2:divisors(b);
(%o4)
```

$$\{1, 2, 3, 5, 6, 10, 15, 30, 101, 202, 303, 505, 606, 1010, 1515, 3030\}$$

3.2.1.4 Ejercicio. Calcular, usando la función `intersection`, el conjunto D de los divisores comunes de a y b.

Solución:

```
(%i5)  D:intersection(D1,D2);
(%o5)
```

$$\{1, 2, 3, 5, 6, 10, 15, 30\}$$

3.2.1.5 Ejercicio. Calcular el máximo común divisor de a y b.

Solución:

```
(%i6)  gcd(a,b);
(%o6)
```

$$30$$

3.2.1.6 Ejercicio. Calcular el mínimo común múltiplo de a y b.

Solución:

```
(%i7)  lcm(a,b);
(%o7)
```

$$\text{lcm}\,(2460, 3030)$$

3.2.2. Ejercicio 2: Estudio de grandes factoriales

3.2.2.1 Ejercicio. Asignarle a la variable n el valor 2,008!

Solución:

```
(%i8)  n:2008!$
```

3.2.2.2 Ejercicio. ¿Cuántas cifras tiene n en base 10?

Solución:

```
(%i9)  bfloat(n);
(%o9)
```

$$8{,}64364185767107_B \times 10^{5761}$$

Por tanto, tiene 5762 cifras.

3.2.2.3 Ejercicio. Calcular la descomposición de n en productos de factores primos.

Solución:

```
(%i10)  ifactors(n);
(%o10)
```

$$[[2, 2001], [3, 1000], [5, 500], [7, 331], [11, 199], [13, 165], [17, 124], [19, 110], [23, 90], [29, 71], [31, 66], [$$

3.2.2.4 Ejercicio. ¿Con cuántos ceros termina n?

Solución: Puesto que el exponente de 5 en la descomposición anterior es 500, n termina con 500 ceros.

3.2.3. Ejercicio 3: Guión para el primo 2008

3.2.3.1 Ejercicio. Escribir un programa para asignarle a la variable `sol3` el término que ocupa la posición 2008 en la sucesión de números primos ordenados de manera creciente.

Solución:

```
(%i13)  sol3:2$
(%i14)  for i:2 thru 2008 do sol3:next_prime(sol3)$
(%i15)  sol3;
(%o15)
```

17467

3.2.4. Ejercicio 4: Guión para el número de primos menores que 100000

3.2.4.1 Ejercicio. Escribir un programa para asignarle a la variable `sol4` el número de primos inferiores a 100,000.

Solución:

```
(%i16)  sol4:0$
(%i17)  for i:2 thru 100000 do
            (if primep(i) then sol4: sol4+1 else sol4)$
(%i18)  sol4;
(%o18)
```

9592

3.2.5. Ejercicio 5: Estudio del primo 9592–ésimo

3.2.5.1 Ejercicio. Escribir un programa para asignarle a la variable `sol5` el término que ocupa la posición 9592 en la sucesión de números primos ordenados de manera creciente.

Solución:

```
(%i19)  sol5:2$
(%i20)  for i:2 thru 9592 do sol5:next_prime(sol5)$
(%i21)  sol5;
(%o21)
```

99991

3.2.5.2 Ejercicio. Comprobar si `sol5` es el mayor primo menor que 100,000.

Solución:

```
(%i22)  is(sol5=prev_prime(100000));
(%o22)
```

true

Capítulo 4

Sucesiones y recursión

4.1. Ejercicios resueltos

4.1.1. Formas de generar una sucesión

4.1.1.1 Ejercicio. Definir la sucesión u_n cuyo término general es $1-\left(-\frac{1}{2}\right)^n$.

Solución:

```
(%i1)  u[n]:=1-(-1/2)^n;
(%o1)
```

$$u_n := 1-\left(\frac{-1}{2}\right)^n$$

4.1.1.2 Ejercicio. Calcular el término que ocupa la posición 20 de la sucesión anterior.

Solución:

```
(%i2)  u[20];
(%o2)
```

$$\frac{1048575}{1048576}$$

4.1.1.3 Ejercicio. Calcular el valor decimal del término anterior.

Solución:

```
(%i3)  float(%);
(%o3)
```

$$0{,}99999904632568$$

4.1.1.4 Ejercicio. Calcular el límite de la sucesión u_n.

Solución:

```
(%i4)  limit(u[n],n,inf);
(%o4)
```

$$1$$

4.1.1.5 Ejercicio. Definir, por recursión, la sucesión

$$v_n = \begin{cases} 1, & \text{si } n = 0 \\ \dfrac{v_{n-1}}{2} + 3, & \text{si } n > 0 \end{cases}$$

Solución:

```
(%i5)  v[0] :  1$
(%i6)  v[n] := v[n-1]/2+3$
```

4.1.1.6 Ejercicio. Calcular la lista de pares $[k, v_k]$ para $0 \leq k \leq 9$.

Solución:

```
(%i7)  makelist([k,v[k]],k,0,9);
(%o7)
```

$$\left[[0,1], \left[1,\frac{7}{2}\right], \left[2,\frac{19}{4}\right], \left[3,\frac{43}{8}\right], \left[4,\frac{91}{16}\right], \left[5,\frac{187}{32}\right], \left[6,\frac{379}{64}\right], \left[7,\frac{763}{128}\right], \left[8,\frac{1531}{256}\right], \left[9,\frac{3067}{512}\right]\right]$$

4.1.1.7 Ejercicio. Asignarle a la variable `terminos` como valor la lista de pares $[k, v'_k]$ para $0 \leq k \leq 5$, donde v'_k es la expresión decimal de v_k.

Solución:

```
(%i8)  terminos : makelist([k,float(v[k])],k,0,9);
(%o8)
```

[[0, 1,0] , [1, 3,5] , [2, 4,75] , [3, 5,375] , [4, 5,6875] , [5, 5,84375] , [6, 5,921875] , [7, 5,9609375] , [8, 5,98046875

4.1.1.8 Ejercicio. Dibujar los puntos de la lista `terminos`.

Solución: La entrada es

```
(%i9)  plot2d([discrete,terminos],
         [style, points],
         [xlabel,"n"],
         [ylabel,"v[n]"])$
```

y la gráfica está en la figura 4.1.

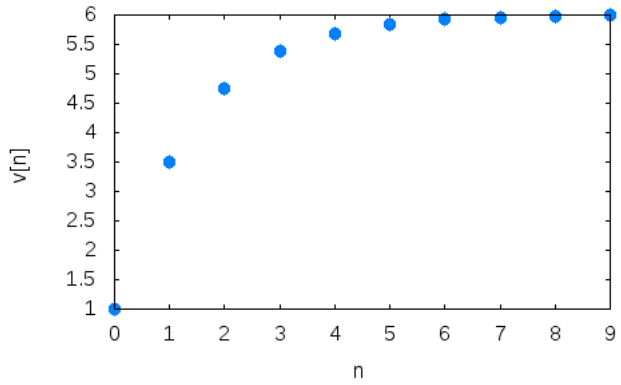

Figura 4.1: Gráfica de la sucesión v

4.1.1.9 Ejercicio. Calcular la lista de pares $[k, v'_k]$ para k entre 100 y 105, donde v'_k es la expresión decimal de v_k.

Solución:

```
(%i10)  makelist([k,float(v[k])],k,100,105);
(%o10)
```

$$[[100,6{,}0],[101,6{,}0],[102,6{,}0],[103,6{,}0],[104,6{,}0],[105,6{,}0]]$$

4.1.1.10 Ejercicio. Calcular el límite de la sucesión v.

Solución:

```
(%i11)  limit(v[n],n,inf);
   Maxima encountered a Lisp error:
    Error in PROGN [or a callee]: Bind stack overflow.
   Automatically continuing.
   To enable the Lisp debugger set *debugger-hook* to nil.
```

Se observa que Maxima no calcula el límite de esta sucesión.

4.1.2. Recurrencias

4.1.2.1 Ejercicio. Cargar el paquete `solve_rec` para resolver recurrencias.

Solución:

```
(%i12)  load(solve_rec)$
```

4.1.2.2 Ejercicio. Resolver la ecuación recurrente $w_n = \frac{w_{n-1}}{2} + 3$

Solución:

```
(%i13)  solve_rec(w[n]=w[n-1]/2+3,w[n]);
(%o13)
```

$$w_n = \frac{\%k_1}{2^n} - 3\,2^{1-n} + 2\,3$$

Se puede simplificar

```
(%i14)  ratsimp(%);
(%o14)
```

$$w_n = \frac{3\,2^{n+1} + \%k_1 - 6}{2^n}$$

4.1.2.3 Ejercicio. Resolver la ecuación recurrente $w_n = \frac{w_{n-1}}{2} + 3$ con la condición inicial $w_0 = 1$.

Solución:

```
(%i15)  solve_rec(w[n]=w[n-1]/2+3,w[n],w[0]=1);
(%o15)
```

$$w_n = \frac{1}{2^n} - 3\,2^{1-n} + 2\,3$$

Se puede simplificar

```
(%i16)  ratsimp(%);
(%o16)
```

$$w_n = \frac{3\,2^{n+1} - 5}{2^n}$$

4.1.2.4 Ejercicio. Calcular w_2.

Solución:

```
(%i17)  w[2];
(%o17)
```

$$\frac{19}{4}$$

4.1.2.5 Ejercicio. Definir la sucesión w_n como la solución de

$$w_n = \begin{cases} 1, & \text{si } n = 0 \\ \dfrac{w_{n-1}}{2} + 3, & \text{si } n > 0 \end{cases}$$

Solución: La solución de la ecuación se obtiene con

```
(%i18)  ratsimp(solve_rec(w[n]=w[n-1]/2+3,w[n],w[0]=1));
(%o18)
```

$$w_n = \frac{3\,2^{n+1} - 5}{2^n}$$

La definición de w_n es

```
(%i19)  define(w[n],rhs(%));
(%o19)
```

$$w_n := \frac{3\,2^{n+1} - 5}{2^n}$$

4.1.2.6 Ejercicio. Calcular w_2.

Solución:

```
(%i20)  w[2];
(%o20)
```

$$\frac{19}{4}$$

4.1.2.7 Ejercicio. Calcular el desarrollo de w_n.

Solución:

```
(%i21)  expand(w[n]);
(%o21)
```

$$6 - \frac{5}{2^n}$$

4.1.2.8 Ejercicio. Borra el valor de la variable u.

Solución:

```
(%i22)  kill(u)$
```

4.1.2.9 Ejercicio. Resolver las ecuaciones que definen la sucesión de Fibonacci

$$u_n = \begin{cases} 0, & \text{si } n = 0 \\ 1, & \text{si } n = 1 \\ u_{m+1} + u_m, & \text{si } n = m+2 \end{cases}$$

Solución:

```
(%i23)  solve_rec(u[n+2]=u[n+1]+u[n],u[n],u[0]=0,u[1]=1);
(%o23)
```

$$u_n = \frac{\left(\sqrt{5}+1\right)^n}{\sqrt{5}\,2^n} - \frac{\left(\sqrt{5}-1\right)^n (-1)^n}{\sqrt{5}\,2^n}$$

4.1.2.10 Ejercicio. Definir la sucesión uC como la forma cerrada de la sucesión de Fibonacci.

Solución:

```
(%i24)  define(uC[n],rhs(%));
(%o24)
```

$$\mathrm{uC}_n := \frac{\left(\sqrt{5}+1\right)^n}{\sqrt{5}\,2^n} - \frac{\left(\sqrt{5}-1\right)^n (-1)^n}{\sqrt{5}\,2^n}$$

4.1.2.11 Ejercicio. Calcular el décimo término de la sucesión de Fibonacci usando su forma cerrada.

Solución:

```
(%i25)  uC[10];
(%o25)
```

$$\frac{\left(\sqrt{5}+1\right)^{10}}{1024\sqrt{5}} - \frac{\left(\sqrt{5}-1\right)^{10}}{1024\sqrt{5}}$$

Su simplificación es

```
(%i26)  ratsimp(%);
```

```
(%o26)
```

55

4.1.3. Representación gráfica de uns sucesión

4.1.3.1 Ejercicio. Asignarle a la variable `puntos` la lista de los puntos correspondientes a los 9 primeros puntos de la sucesión de Fibonacci calculados con la forma cerrada.

Solución:

```
(%i27)  puntos : makelist([k,ratsimp(uC[k])],k,0,8);
(%o27)
          [[0,0],[1,1],[2,1],[3,2],[4,3],[5,5],[6,8],[7,13],[8,21]]
```

4.1.3.2 Ejercicio. Representar gráficamente los puntos correspondientes a los 9 primeros puntos de la sucesión de Fibonacci calculados con la forma cerrada.

Solución: La entrada es

```
(%i28)  plot2d([discrete,puntos],[x,0,9],[style,points])$
```

y la gráfica está en la figura 4.2.

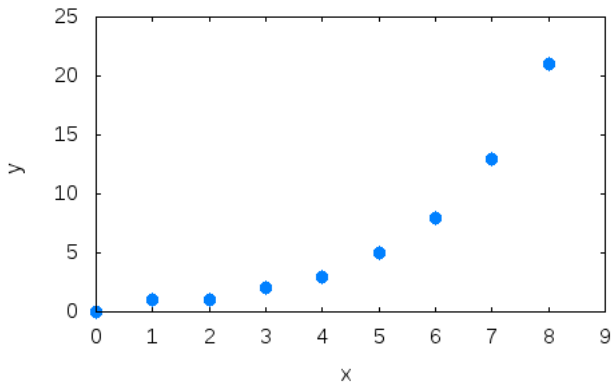

Figura 4.2: Gráfica de la sucesión de Fibonacci

4.1.4. Sucesiones definidas por sumatorios

4.1.4.1 Ejercicio. Calcular la suma de los cuadrados de los n primeros números

Solución:

```
(%i29)  sum(k^2,k,1,n);
(%o29)
```

$$\sum_{k=1}^{n} k^2$$

4.1.4.2 Ejercicio. Cargar el paquete `simplify_sum`.

Solución:

```
(%i30)  load(simplify_sum) $
```

4.1.4.3 Ejercicio. Calcular la suma de los cuadrados de los n primeros números

Solución:

```
(%i31)  'sum(k^2,k,1,n)=simplify_sum(sum(k^2,k,1,n));
(%o31)
```

$$\sum_{k=1}^{n} k^2 = \frac{2\,n^3+3\,n^2+n}{6}$$

4.1.4.4 Ejercicio. Descomponer en factores la suma anterior.

Solución:

```
(%i32)  factor(%);
(%o32)
```

$$\sum_{k=1}^{n} k^2 = \frac{n\,(n+1)\,(2\,n+1)}{6}$$

4.1.4.5 Ejercicio. Calcular la suma de la serie de término $\frac{1}{k^2}$.

Solución:

```
(%i33)  'limit(sum(1/k^2,k,1,n),n,inf) =
         simplify_sum(sum(1/k^2,k,1,inf));
(%o33)
```

$$\lim_{n\to\infty} \sum_{k=1}^{n} \frac{1}{k^2} = \frac{\pi^2}{6}$$

4.1.4.6 Ejercicio. Calcular la suma de 1 a n de $k\binom{n}{k}$.

Solución:

```
(%i33)  'sum(k*binomial(n,k),k,1,n) =
         simplify_sum(sum(k*binomial(n,k),k,1,n));
(%o34)
```

$$\sum_{k=1}^{n} k\binom{n}{k} = n\,2^{n-1}$$

4.1.5. Productos y factoriales

4.1.5.1 Ejercicio. Calcular el producto de los cuadrados de los 5 primeros números.

Solución:

```
(%i35)  product(k^2,k,1,5);
(%o35)
```

$$14400$$

4.1.5.2 Ejercicio. Definir la sucesión q_n cuyo término n–ésimo es

$$\frac{\binom{2n}{n}}{4^n}$$

Solución:

```
(%i36)  q[n] := binomial(2*n,n)/4^n;
(%o36)
```

$$q_n := \frac{\binom{2n}{n}}{4^n}$$

4.1.5.3 Ejercicio. Expresar q_n mediante factoriales.

Solución:

```
(%i37)  makefact(q[n]);
(%o37)
```

$$\frac{(2n)!}{4^n\,n!^2}$$

4.1.5.4 Ejercicio. Expresar $\frac{q_{n+1}}{q_n}$ mediante factoriales.

Solución:

```
(%i38)  q[n+1]/q[n];
(%o38)
```

$$\frac{\binom{2\,(n+1)}{n+1}}{\binom{4\,2\,n}{n}}$$

```
(%i39)  makefact(%);
(%o39)
```

$$\frac{n!^2\ (2\ (n+1))!}{4\ (n+1)!\ (2\,n)!\ (2\ (n+1)-n-1)!}$$

4.1.5.5 Ejercicio. Simplificar los factoriales de la expresión anterior.

Solución:

```
(%i40)  minfactorial(%);
(%o40)
```

$$\frac{2\ (n+1)-1}{2\ (2\ (n+1)-n-1)}$$

4.1.5.6 Ejercicio. Simplificar la expresión anterior.

Solución:

```
(%i41)  ratsimp(%);
(%o41)
```

$$\frac{2\,n+1}{2\,n+2}$$

4.1.6. Sucesiones del tipo $u_{n+1} = f(u_n)$

4.1.6.1 Ejercicio. Definir la función f como la función coseno.

Solución:

```
(%i42)  f(x):=cos(x);
(%o42)
```

$$f\,(x) := \cos x$$

4.1.6.2 Ejercicio. Redefinir la sucesión u por recurrencia:

$$u_n = \begin{cases} 1{,}5, & \text{si } n = 0 \\ f(u_{n-1}), & \text{si } n > 0 \end{cases}$$

Solución:

```
(%i43)  kill(u)$
(%i44)  u[0] :   1.5$
(%i45)  u[n] := f(u[n-1])$
```

4.1.6.3 Ejercicio. Definir la lista L_1 de los puntos (u_k, u_{k+1}) y la lista L_2 de los puntos $(u_{k+1}, uk + 1)$ para k entre 0 y 10.

Solución:

```
(%i46)  L1:makelist([u[k],u[k+1]],k,0,10)$
(%i47)  L2:makelist([u[k+1],u[k+1]],k,0,10)$
```

4.1.6.4 Ejercicio. Definir L como la lista obtenida intercalando los puntos de L_1 y L_2; es decir, L es `[L1[1],L2[1],L1[2],L2[2],L1[3],L2[3],...]`.

Solución:

```
(%i48)  L : join(L1,L2)$
```

4.1.6.5 Ejercicio. Dibujar, sobre una misma gráfica, la función f, la función $y = x$ y los puntos de la lista L.

Solución: La entrada es

```
(%i49)  wxplot2d([f(x),x,[discrete,L]], [x,0,%pi/2], [y,0,1.2],
          [style, [lines,2,1], [lines,1,2], [linespoints,1,1,3,1]],
          [gnuplot_preamble, "set grid"],
          [legend, "y=f(x)", "y=x", "sucesión u"])$
```

y la gráfica está en la figura 4.3.

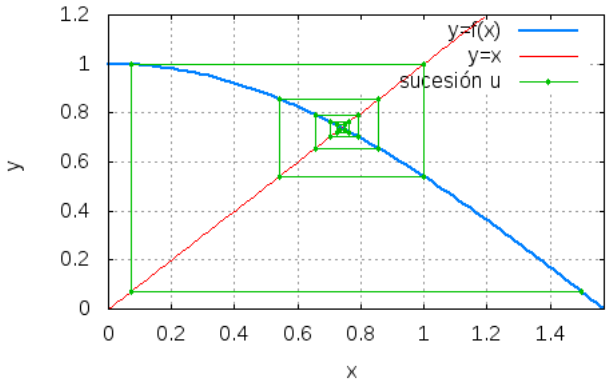

Figura 4.3: Gráfica de la sucesión u_n

4.2. Ejercicios propuestos

4.2.1. Ejercicio 1: Sucesión de Fibonacci

4.2.1.1 Ejercicio. La sucesión de Fibonacci está definida como

$$f_n = \begin{cases} 0, & \text{si } n = 0 \\ 1, & \text{si } n = 1 \\ f_{n-1} + f_{n-2}, & \text{si } n > 1 \end{cases}$$

Definir f_n como el n–ésimo término de la sucesión de Fibonacci.

Solución:

```
(%i1)  f[0] : 0$
(%i2)  f[1] : 1$
(%i3)  f[n] := f[n-1]+f[n-2]$
```

4.2.1.2 Ejercicio. Definir la lista l_1 cuyos elementos son los 20 primeros términos de la sucesión de Fibonacci.

Solución:

```
(%i4)  l1: makelist([n,f[n]],n,0,20);
(%o4)
```

$$[[0,0],[1,1],[2,1],[3,2],[4,3],[5,5],[6,8],[7,13],[8,21],[9,34],[10,55],[11,89],[12,144],[13,233],$$

4.2.1.3 Ejercicio. Calcular el término que ocupa la posición 20 en la sucesión de Fibonacci.

Solución:

```
(%i5)  f[20];
(%o5)
```

$$6765$$

4.2.1.4 Ejercicio. Calcular el término de posición 20 de la sucesión de Fibonacci de forma iterativa, usando sólo dos variables: a y b.

Solución:

```
(%i6)  a:0$
(%i7)  b:1$
(%i8)  for i:1 thru 20 do (c:a, d:b, a:d, b:c+d)$
(%i9)  b;
(%o9)
```

10946

4.2.1.5 Ejercicio. Definir la sucesión g, que calcule el término n–ésimo de la sucesión de Fibonacci de forma iterativa, usando sólo dos variables: a y b. Usando la función g, calcular el término de posición 20 de la sucesión de Fibonacci.

Solución:

```
(%i10)  g[n]:= block ([a:0,b:1,c,d],
                for i:1 thru n do (c:a, d:b, a:d, b:c+d),
                a)$
```

Solución:

```
(%i11)  g[20];
(%o11)
```

6765

4.2.1.6 Ejercicio. Comprobar si se puede obtener el término 800 de la sucesión de Fibonacci mediante alguna de las dos funciones f ó g.

Solución: Con la g se puede

```
(%i12)  g[800];
(%o12)
```

69283081864224717136290077681328518273399124385204820718966040597691435587278383112277 16

Con la f no se puede

```
(%i13)  f[800];
```

Maxima encountered a Lisp error:
Error in PROGN [or a callee]: Bind stack overflow.
Automatically continuing.
To enable the Lisp debugger set *debugger-hook* to nil.

4.2.2. Ejercicio 2: Series

4.2.2.1 Ejercicio. Definir s_n como la suma de los n primeros términos de la sucesión $\frac{(-1)^{k+1}}{k!}$.

Solución:

```
(%i14)  s[n] := sum((-1)^(k+1)/k!,k,1,n);
(%o14)
```

$$s_n := \text{sum}\left(\frac{(-1)^{k+1}}{k!}, k, 1, n\right)$$

4.2.2.2 Ejercicio. Calcular los valores exactos de s_1, s_2, s_5 y s_9.

Solución:

```
(%i15)  makelist('s[i]=s[i],i,[1,2,5,9]);
(%o15)
```

$$\left[s_1 = 1, s_2 = \frac{1}{2}, s_5 = \frac{19}{30}, s_9 = \frac{28673}{45360}\right]$$

4.2.2.3 Ejercicio. Calcular los valores decimales aproximados de s_{20} y s_{50}.

Solución:

```
(%i16)  makelist('s[i]=s[i],i,[20,50]), numer;
(%o16)
```

$$[s_{20} = 0{,}63212055882856, s_{50} = 0{,}63212055882856]$$

4.2.2.4 Ejercicio. Cargar el paquete `simplify_sum` y calcular la suma de la serie s_n.

Solución:

```
(%i17)  load(simplify_sum)$
(%i17)  'sum((-1)^(k+1)/k!,k,1,inf) =
        simplify_sum(sum((-1)^(k+1)/k!,k,1,inf));
(%o18)
```

$$\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k!} = \frac{\sqrt{\pi}\,\text{bessel\_i}\left(\frac{1}{2}, \frac{1}{2}\right)}{\sqrt{e}}$$

4.2.3. Ejercicio 3: Recurrencias

Un banco presta un capital K al t por ciento aunual, que se reembolsa en N años, con anualidades x constante. Sea $c_0 = K$ y sea c_n el capital pendiente de pagar después de la n–ésima anualidad. Entonces, $c_{n+1} = (1+t)c_n - x$.

4.2.3.1 Ejercicio. Expresar c_n de manera explícita en función de n, K, t y x.

Solución:

```
(%i19)  load(solve_rec)$
(%i20)  kill(c)$
(%i21)  solve_rec(c[n]=(1+t)*c[n-1]-x,c[n],c[0]=K);
(%o21)
```

$$c_n = (t+1)^n\ K - \frac{(t+1)^n\ x}{t} + \frac{x}{t}$$

```
(%i22)  ratsimp(%);
(%o22)
```

$$c_n = \frac{t\ (t+1)^n\ K + \left(1-(t+1)^n\right)\ x}{t}$$

4.2.3.2 Ejercicio. Se sabe que $c_N = 0$. Deducir el valor de x en función de K, t y N.

Solución:

```
(%i23)  define(c[n],rhs(%));
(%o23)
```

$$c_n := \frac{t\ (t+1)^n\ K + \left(1-(t+1)^n\right)\ x}{t}$$

```
(%i24)  solve(c[N],x);
(%o24)
```

$$\left[x = \frac{t\ (t+1)^N\ K}{(t+1)^N - 1}\right]$$

4.2.3.3 Ejercicio. Calcular el importe de una anualidad, cuando $K = 100000$, $t = 5{,}5\,\%$ y $N = 15$.

Solución:

```
(%i25)  subst([K=100000,t=5.5,N=15],%);
(%o25)
```

$$[x = 550000{,}000000352]$$

4.2.4. Ejercicio 4: Recurrencia a partir de funciones

4.2.4.1 Ejercicio. Definir la función $f(x) = \dfrac{x}{3-2x}$.

Solución:

```
(%i26)  f(x) := x/(3-2*x);
(%o26)
```

$$f(x) := \frac{x}{3-2x}$$

4.2.4.2 Ejercicio. Definir la sucesión u_n tal que

$$u_n = \begin{cases} 2, & \text{si } n = 0 \\ f(u_{n-1}), & \text{si } n > 0 \end{cases}$$

Solución:

```
(%i27)  u[0] : 2$
(%i28)  u[n] := f(u[n-1])$
```

4.2.4.3 Ejercicio. Calcular u_1, u_2 y u_9.

Solución:

```
(%i29)  makelist('u[i]=u[i],i,[1,2,9]);
(%o29)
```

$$\left[u_1 = -2, u_2 = -\frac{2}{7}, u_9 = -\frac{2}{19681}\right]$$

4.2.4.4 Ejercicio. Dibujar, en la misma gráfica, la función f, la recta de ecuación $y = x$ y los puntos de coordenada $(u_k, f(u_k))$ para $0 \leq k \leq 15$.

Solución: Los puntos son

```
(%i30)  puntos : makelist([u[k],f(u[k])],k,0,15)$
```

La entrada es

```
(%i31)  plot2d([f(x),x,[discrete,puntos]],[x,-9,9],[y,-9,9],
                [style, lines,lines,linespoints],
                [gnuplot_preamble, "set key left top"])$
```

y la gráfica está en la figura 4.4.

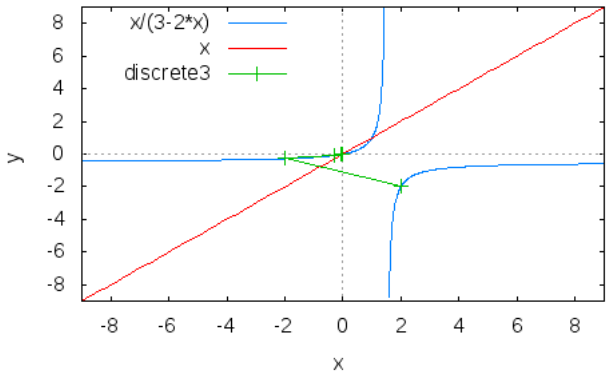

Figura 4.4: Gráfica de la sucesión

4.2.4.5 Ejercicio. Conjeturar la monotonía de la sucesión u_n y su límite.

Solución: La sucesión u_n es monótona decreciente y su límite es 0.

4.2.4.6 Ejercicio. Resolver la ecuación $f(x) = x$. Llamar a las raíces a y b.

Solución: La solución es

```
(%i34)  solve(f(x)=x,x);
(%o34)
```

$$[x = 0, x = 1]$$

Las raíces son

```
(%i35)  a:0$
(%i36)  b:1$
```

4.2.4.7 Ejercicio. Definir la sucesión $w_n = \dfrac{u_n - a}{u_n - b}$.

Solución:

```
(%i37)  w[n] := (u[n]-a)/(u[n]-b);
(%o37)
```

$$w_n := \frac{u_n - a}{u_n - b}$$

4.2.4.8 Ejercicio. Calcular los 10 primeros términos de la sucesión w_n.

Solución:

```
(%i38)  makelist(w[n],n,0,9);
(%o38)
```

$$\left[2,\frac{2}{3},\frac{2}{9},\frac{2}{27},\frac{2}{81},\frac{2}{243},\frac{2}{729},\frac{2}{2187},\frac{2}{6561},\frac{2}{19683}\right]$$

4.2.4.9 Ejercicio. Comprobar que w_n es una progresión geométrica y calcular su razón.

Solución: Se observa que w_n es la progresión geométrica de término inicial 2 y razón 1/3.

4.2.4.10 Ejercicio. Deducir la expresión de u_n en función de n.

Solución:

```
(%i39)  solve((x-a)/(x-b)=2/3^n,x);
(%o39)
```

$$\left[x=-\frac{2}{3^n-2}\right]$$

Por tanto, $u_n=\dfrac{-2}{3^n-2}$. Puede comprobarse como sigue

```
(%i40)  makelist([u[n],-2/(3^n-2)],n,0,5);
(%o40)
```

$$\left[[2,2],[-2,-2],\left[-\frac{2}{7},-\frac{2}{7}\right],\left[-\frac{2}{25},-\frac{2}{25}\right],\left[-\frac{2}{79},-\frac{2}{79}\right],\left[-\frac{2}{241},-\frac{2}{241}\right]\right]$$

Capítulo 5

Programación

5.1. Ejercicios resueltos

5.1.1. Funciones y procedimientos

5.1.1.1 Ejercicio. Se definen las variables a y b y el procedimiento `ej_proc(n)` como sigue

```
a:0$
b:1$
ej_proc(n):=block([a,k], a:2, k:3, a+b+k-n)$
```

Calcular los valores de `ej_proc(10)`, a y k.

Solución: La definición es

```
(%i1)  a:0$
(%i2)  b:1$
(%i3)  ej_proc(n):=block([a,k], a:2, k:3, a+b+k-n)$
```

El cálculo es

```
(%i4)  ej_proc(10);
(%o4)
```

$$-4$$

```
(%i5)  a;
(%o5)
```

$$0$$

```
(%i6)  k;
```

```
(%o6)
```

$$k$$

5.1.2. Estructura condicional

5.1.2.1 Ejercicio. Un número natural es perfecto si es igual a la suma de sus divisores positivos distintos de él mismo. Definir el procedimiento `perfecto(n)` que se verifique si n es un número perfecto. Por ejemplo,

```
(%i1) perfecto(2.5)$ perfecto(-3)$ perfecto(6)$ perfecto(9)$
2.5 no es un número natural
-3 no es un número natural
6 es un número perfecto
9 no es un número perfecto
```

Solución:

```
(%i7)  perfecto(n):=block(
         if n#floor(n) or n<0 then print(n," no es un número natural")
         elseif n=divsum(n)-n then print(n," es un número perfecto")
         else print(n," no es un número perfecto"))$
```

5.1.3. Iteración con el bucle para (for)

5.1.3.1 Ejercicio. Definir, por iteración con `for`, el procedimiento `primos` tal que `primos(n)` es la lista de los números primos menores o iguales que n. Por ejemplo,

```
(%i1) primos(50);
(%o1) [2,3,5,7,11,13,17,19,23,29,31,37,41,43,47]
```

Solución: La definición es

```
(%i12)  primos(n):=block([lista,k],
            lista : [],
            for k from 1 thru n do
               (if primep(k) then lista : cons(k,lista)),
            reverse(lista))$
```

El cálculo es

```
(%i13)  primos(50);
(%o13)
```

$$[2,3,5,7,11,13,17,19,23,29,31,37,41,43,47]$$

5.1.3.2 Ejercicio. Definir, por recursión, la función `primosR` tal que `primosR(n)` es la lista de los números primos menores o iguales que n. Por ejemplo,

```
(%i1) primosR(50);
(%o1) [2,3,5,7,11,13,17,19,23,29,31,37,41,43,47]
```

Solución:

```
(%i14)  primosR(n) := reverse(primosRaux(n))$
(%i15)  primosRaux(n) :=
            if n=0 then []
            elseif primep(n) then cons(n,primosRaux(n-1))
            else primosRaux(n-1)$
(%i16)  primosR(50);
(%o16)
                [2,3,5,7,11,13,17,19,23,29,31,37,41,43,47]
```

5.1.4. Iteración con el bucle mientras (while)

5.1.4.1 Ejercicio. Definir, por iteración con `while`, el procedimiento `biseccion` tal que `biseccion(f)` pregunta por el valor inferior, a, el valor superior, b, y el error, e y calcula la solución de $f(x) = 0$ entre a y b con un error menor que e. Por ejemplo,

```
(%i1) (g(x):=x^2-2, biseccion(g));
solución aproximada de f(x)=0 en [a,b]
escribe a 0;
escribe b 2;
escribe el error 0.001;
la solución x0 está entre 1.4140625 y 1.4150390625
(%o47) 1.4150390625
```

Solución:

```
(%i17)  biseccion(f):=block([a,b,e],
            print("solución aproximada de f(x)=0 en [a,b]"),
            a : read("escribe a "),
            b : read("escribe b "),
            e : read("escribe el error "),
            while (b-a)>e do
               (c : (a+b)/2,
                if f(a)*f(c)<0 then b:c else a:c),
```

```
            print("la solución x0 está entre ",float(a),"
                  y ",float(b)))$
```

5.1.4.2 Ejercicio. Definir, por recursión, la función *biseccionR* tal que $biseccionR(f,a,b,e)$ es la solución de $f(x) = 0$ entre a y b con un error menor que e. Por ejemplo,

```
(%i1) (g(x):=x^2-2, biseccionR(g,0,2,0.001)), numer;
(%o1) 1.4150390625
```

Solución:

```
(%i18)  biseccionR(f,a,b,e) := block([c],
            if b-a <= e then b
            else (c : (a+b)/2,
                  if f(a)*f(c)<0 then biseccionR(f,a,c,e)
                  else biseccionR(f,c,b,e)))$
```

5.2. Ejercicios propuestos

5.2.1. Ejercicio 1: Tangente a una curva

5.2.1.1 Ejercicio. Definir la función *tangente* tal que $tangente(f,a)$ es la ecuación de la tangente a la función f en el punto de abscisa a. Por ejemplo,

```
(%i1) (f(x):=x^3, tangente(f,2));
(%o1) y=12*(x-2)+8
```

Solución: La defición es

```
(%i1) tangente(f,a) := block ([m],
          m : at(diff(f(x),x),x=a),
          y = ratsimp(f(a)+m*(x-a)))$
```

El cálculo es

```
(%i2) (f(x):=x^3, tangente(f,2));
(%o2)
```

$$y = 12\,x - 16$$

5.2.1.2 Ejercicio. Calcular la tangente a $f(x) = \ln(\tan(|x|)$ en el punto de abscisa $-\pi/12$.

Solución:

```
(%i3) (f(x)=ln(tan(abs(x))), tangente(f,-%pi/12));
(%o3)
```

$$y = \frac{18\,\pi^2\,x + \pi^3}{864}$$

5.2.2. Ejercicio 2: Signos del trinomio

5.2.2.1 Ejercicio. Definir el procedimiento *signosTrinomio* tal que $signosTrinomio(a,b,c)$ es la tabla de la variación de los signos del trinomio $ax^2 + bx + c$. Por ejemplo,

```
(%i1) signosTrinomio(1,-2,1);
(%o1) [[[-inf,1],+],[1,0],[[1,inf],+]]
(%i2) signosTrinomio(-1,2,-1);
(%o2) [[[-inf,1],-],[1,0],[[1,inf],-]]
(%i3) signosTrinomio(1,-3,2);
(%o3) [[[-inf,1],+],[1,0],[[1,2],-],[2,0],[[2,inf],+]]
(%i4) signosTrinomio(-1,3,-2);
```

```
(%o4) [[[-inf,2],-],[2,0],[[2,1],+],[1,0],[[1,inf],-]]
(%i5) signosTrinomio(1,0,1);
(%o5) [[[-inf,inf],+]]
(%i6) signosTrinomio(-1,0,-1);
(%o6) [[[-inf,inf],-]]
```

Se supone que a es distinto de cero.

Solución:

```
(%i4)  signosTrinomio(a,b,c) := block ([d : b^2-4*a*c,
                                        e : -b/(2*a),
                                        x1,
                                        x2],
          x1 : (-b-sqrt(d))/(2*a),
          x2 : (-b+sqrt(d))/(2*a),
          if d=0 then if a>0 then [[[minf,e],"+"],[e,0],[[e,inf],"+"]]
                              else  [[[minf,e],"-"],[e,0],[[e,inf],"-"]]
          elseif d>0  then if a>0 then [[[minf,x1],"+"],[x1,0],[[x1,x2]
                                   else [[[minf,x1],"-"],[x1,0],[[x1,x2]
          else if a>0 then [[[minf,inf],"+"]]
                      else [[[minf,inf],"-"]])$
```

```
(%i5)  signosTrinomio(1,-2,1);
(%o5)
```

$$[[[-\infty,1],+],[1,0],[[1,\infty],+]]$$

```
(%i6)  signosTrinomio(-1,2,-1);
(%o6)
```

$$[[[-\infty,1],-],[1,0],[[1,\infty],-]]$$

```
(%i7)  signosTrinomio(1,-3,2);
(%o7)
```

$$[[[-\infty,1],+],[1,0],[[1,2],-],[2,0],[[2,\infty],+]]$$

```
(%i8)  signosTrinomio(-1,3,-2);
```

```
(%o8)
```
$$[[[-\infty,2],-],[2,0],[[2,1],+],[1,0],[[1,\infty],-]]$$

```
(%i9)  signosTrinomio(1,0,1);
(%o9)
```
$$[[[-\infty,\infty],+]]$$

```
(%i10)  signosTrinomio(-1,0,-1);
(%o10)
```
$$[[[-\infty,\infty],-]]$$

5.2.2.2 Ejercicio. Calcular la tabla de la variación de los signos del trinomio $-6x^2 - 3x + 14/3$.

Solución:

```
(%i11)  signosTrinomio(-6,-3,14/3);
(%o11)
```
$$\left[\left[\left[-\infty,\frac{2}{3}\right],-\right],\left[\frac{2}{3},0\right],\left[\left[\frac{2}{3},-\frac{7}{6}\right],+\right],\left[-\frac{7}{6},0\right],\left[\left[-\frac{7}{6},\infty\right],-\right]\right]$$

5.2.3. Ejercicio 3: Simulación aleatoria

Se lanza un dado cúbico equilibrado hasta que se obtiene la cara 6 por primera vez. Se designa por X la variable aleatorio que cuenta el número de lanzamientos efectuados. Se dice que X es el tiempo de espera del primer 6.

5.2.3.1 Ejercicio. Definir el procedimiento $X()$ que simule una serie de lanzamientos del dado y devuelva el número de lanzamientos realizados para obtener el 6 por primera vez.

Solución: Solución recursiva

```
(%i12)  X() := Xaux(1)$
(%i13)  Xaux(n) :=  if 1+random(6)=6 then n else Xaux(n+1)$
```

Solución iterativa

```
(%i14)  X() := for cuenta:0 do
                  (if 1+random(6) = 6 then return(cuenta))$
```

5.2.3.2 Ejercicio. Con la ayuda del bucle `for`, definir el procedimiento $simulacion(n)$ que simule una serie de n lanzamientos y devuelva la lista de frecuencia de los eventos $[X = i]$ para $1 \leq i \leq 60$. Por ejemplo,

```
(%i1) simulacion(1000);
(%o1) [0,145,115,104,88,61,65,53,51,50,40,28,30,29,27,13,21,18,10,6,8,
       5,1,2,4,3,1,0,1,0,0,1,0,1,0,0,1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0
       0,0,0]
```

Solución: La definición es

```
(%i15)  simulacion(n) := block ([i],
            remarray(X),
            array(X,60),
            fillarray(X,makelist(0,i,1,60)),
            for a:1 thru  n do
               (i : X(),
                X[i] : X[i]+1),
            listarray(X))$
```

El cálculo es

```
(%i16)  simulacion(1000);
(%o16)
```

[153, 159, 110, 104, 69, 69, 60, 38, 48, 23, 23, 20, 15, 16, 14, 13, 16, 13, 5, 5, 5, 3, 1, 1, 5, 2, 0, 3, 3, 0, 0, 0, 0, 0, 0, 0

5.2.3.3 Ejercicio. Definir la función *media* tal que $media(n)$ es el valor medio de X en n lanzamientos. Calcular tres veces $media(1000)$.

Solución: La definición es

```
(%i17)  media(n) := (simulacion(n),
                     float(sum (X[i]*i,i,0,60)/n))$
```

El cálculo es

```
(%i18)  media(100);
(%o18)
```

5,81

```
(%i19)  media(100);
(%o19)
                                4,56

(%i20)  media(100);
(%o20)
                                4,53
```

5.2.4. Ejercicio 4: Conjetura de Goldbach

La conjetura de Goldbach afirma que todo número natural par mayor que 3 se puede escribir como la suma de dos números primos. Por ejemplo, $4 = 2 + 2$, $20 = 3 + 17$, $50 = 3 + 47$.

5.2.4.1 Ejercicio. Definir la función *goldbach* tal que $goldbach(n)$ es una descomposición de n como suma de dos números primos. Por ejemplo,

```
(%i1) goldbach(20);
(%o1) [3,17]
```

Indicación: Iterar los primos desde $x = 2$ a $n/2$ hasta que $n - x$ sea primo.

Solución:

```
(%i21)  goldbach(n) := block ([x],
            for x:2 next next_prime(x) thru n/2 do
               if primep(n-x) then return([x,n-x]))$
```

5.2.4.2 Ejercicio. Descomponer 2010 como suma de dos primos.

Solución:

```
(%i22)  goldbach(2010);
(%o22)
                              [7,2003]
```

5.2.4.3 Ejercicio. Definir la función *goldbachTodas* tal que $goldbachTodas(n)$ es la lista de todas las descomposiciones de n como suma de dos números primos x e y con $x \leq y$. Por ejemplo,

```
(%i1) goldbachTodas(20);
(%o1) [[7,13],[3,17]]
```

Solución: La definición es

```
(%i23)  goldbachTodas (n) := block ([todas:[]],
            for x:2 thru n/2 do
               if primep(x) and primep(n-x)
               then todas : cons([x,n-x],todas),
            todas)$
```

El cálculo es

```
(%i24)  goldbachTodas(20);
(%o24)
```

$$[[7,13],[3,17]]$$

5.2.4.4 Ejercicio. Calcular el número de descomposiciones de 2010 como suma de dos primos.

Solución:

```
(%i25)  length(goldbachTodas(2010));
(%o25)
```

$$84$$

Capítulo 6

Matrices con *Maxima*

6.1. Ejercicios resueltos

6.1.1. Definición de una matriz

6.1.1.1 Ejercicio. Definir la matriz

$$M = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}$$

Solución:

```
(%i1)  M:matrix([2,1,1],[1,2,1],[1,1,2]);
(%o1)
```

$$\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}$$

6.1.1.2 Ejercicio. Definir la matriz

$$N = \begin{pmatrix} 2 & 0 & 3 \\ 4 & 1 & 5 \end{pmatrix}$$

Solución:

```
(%i2)  N:matrix([2,0,3],[4,1,5]);
(%o2)
```

$$\begin{pmatrix} 2 & 0 & 3 \\ 4 & 1 & 5 \end{pmatrix}$$

6.1.1.3 Ejercicio. Definir la matriz A de orden 3×3 cuyo elemento a_{ij} es $\dfrac{(-1)^{i+j}}{i+j}$.

Solución:

```
(%i3)  a[i,j]:=(-1)^(i+j)/(i+j) $
(%i4)  A:genmatrix(a,3,3);
(%o4)
```

$$\begin{pmatrix} \frac{1}{2} & -\frac{1}{3} & \frac{1}{4} \\ -\frac{1}{3} & \frac{1}{4} & -\frac{1}{5} \\ \frac{1}{4} & -\frac{1}{5} & \frac{1}{6} \end{pmatrix}$$

6.1.1.4 Ejercicio. Definir la matriz B de orden 2×2.

Solución:

```
(%i5)  B:genmatrix(b,2,2);
(%o5)
```

$$\begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix}$$

6.1.2. Operaciones con matrices

6.1.2.1 Ejercicio. Calcular la suma de las matrices M y $2A$.

Solución:

```
(%i6)  M+2*A ;
(%o6)
```

$$\begin{pmatrix} 3 & \frac{1}{3} & \frac{3}{2} \\ \frac{1}{3} & \frac{5}{2} & \frac{3}{5} \\ \frac{3}{2} & \frac{3}{5} & \frac{7}{3} \end{pmatrix}$$

6.1.2.2 Ejercicio. Calcular el producto de las matrices M y N.

Solución:

```
(%i7)  N.M;
(%o7)
```

$$\begin{pmatrix} 7 & 5 & 8 \\ 14 & 11 & 15 \end{pmatrix}$$

6.1.2.3 Ejercicio. Calcular M^5.

Solución:

```
(%i8)  M^^5;
(%o8)
```

$$\begin{pmatrix} 342 & 341 & 341 \\ 341 & 342 & 341 \\ 341 & 341 & 342 \end{pmatrix}$$

6.1.2.4 Ejercicio. Calcular el rango de la matriz N.

Solución:

```
(%i9)  rank(N);
(%o9)
```

$$2$$

6.1.2.5 Ejercicio. Calcular el determinante de la matriz A.

Solución:

```
(%i10)  determinant(A);
(%o10)
```

$$\frac{1}{43200}$$

6.1.2.6 Ejercicio. Calcular la inversa de la matriz A.

Solución:

```
(%i11)  invert(A);
(%o11)
```

$$\begin{pmatrix} 72 & 240 & 180 \\ 240 & 900 & 720 \\ 180 & 720 & 600 \end{pmatrix}$$

6.1.2.7 Ejercicio. Calcular la traspuesta de la matriz N.

Solución:

```
(%i12)  transpose(N);
(%o12)
```

$$\begin{pmatrix} 2 & 4 \\ 0 & 1 \\ 3 & 5 \end{pmatrix}$$

6.1.3. Diagonalización de matrices cuadradas

6.1.3.1 Ejercicio. Calcular el polinomio característico de la matriz M.

Solución:

```
(%i13)  charpoly(M,x);
(%o13)
```

$$2\,x+\left(\left(2-x\right)^2-1\right)\,\left(2-x\right)-2$$

```
(%i14)  expand(%);
(%o14)
```

$$-x^3+6\,x^2-9\,x+4$$

6.1.3.2 Ejercicio. Calcular los autovalores de la matriz M.

Solución:

```
(%i15)  solve(%=0,x) ;
(%o15)
```

$$[x=4,x=1]$$

```
(%i16)  factor(charpoly(M,x));
(%o16)
```

$$-\left(x-4\right)\,\left(x-1\right)^2$$

```
(%i17)  eigenvalues(M);
(%o17)
```

$$[[4,1],[1,2]]$$

6.1.3.3 Ejercicio. Calcular los autovectores de la matriz M.

Solución:

```
(%i18)  eigenvectors(M);
(%o18)
```

$$[[[4,1],[1,2]],[[[1,1,1]],[[1,0,-1],[0,1,-1]]]]$$

6.1.3.4 Ejercicio. Calcular la matriz P que es la matriz de paso de la base canónica de $\mathbb{R}^3$ a una base de autovectores.

Solución:

```
(%i19)  P:transpose( matrix([1,1,1],[1,0,-1],[0,1,-1]) ) ;
(%o19)
```

$$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & -1 & -1 \end{pmatrix}$$

6.1.3.5 Ejercicio. Calcular la matriz Q que es la inversa de P.

Solución:

```
(%i20)  Q:invert(P);
(%o20)
```

$$\begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \end{pmatrix}$$

6.1.3.6 Ejercicio. Calcular la matriz diagonal D cuyos valores en la diagonal son los autovalores de M.

Solución:

```
(%i21)  D:diag_matrix(4,1,1);
(%o21)
```

$$\begin{pmatrix} 4 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

6.1.3.7 Ejercicio. Calcular el producto de las matrices P, D y Q.

Solución:

```
(%i22)  P.D.Q;
(%o22)
```

$$\begin{pmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{pmatrix}$$

6.1.3.8 Ejercicio. Comprobar que el producto de las matrices P, D y Q es igual a la matriz M.

Solución:

```
(%i23)  is(P.D.Q = M);
(%o23)
```

true

6.2. Ejercicios propuestos

6.2.1. Ejercicio 1: Cálculo con matrices con 1 parámetro

6.2.1.1 Ejercicio. Definir la matriz

$$M(k) = \begin{pmatrix} 2 & -1 & 1 \\ -1 & k & 1 \\ 1 & 1 & 2 \end{pmatrix}$$

para $k \in \mathbb{R}$.

Solución:

```
(%i1)  M(k) := matrix([2,-1,1],[-1,k,1],[1,1,2]);
(%o1)
```

$$M(k) := \begin{pmatrix} 2 & -1 & 1 \\ -1 & k & 1 \\ 1 & 1 & 2 \end{pmatrix}$$

6.2.1.2 Ejercicio. Calcular el determinante de $M(k)$.

Solución:

```
(%i2)  determinant(M(k));
(%o2)
```

$$2\,(2k-1) - k - 4$$

```
(%i3)  expand(%);
(%o3)
```

$$3k - 6$$

6.2.1.3 Ejercicio. Determinar los valores de k para los que $M(k)$ es invertible.

Solución:

```
(%i4)  solve(%,k);
(%o4)
```

$$[k = 2]$$

Por tanto, $M(k)$ es inversible para k distinto de 2.

6.2.1.4 Ejercicio. Calcular la inversa de $M(k)$.

Solución:

```
(%i5)  invert(M(k));
(%o5)
```

$$\begin{pmatrix} \frac{2k-1}{2(2k-1)-k-4} & \frac{3}{2(2k-1)-k-4} & \frac{-k-1}{2(2k-1)-k-4} \\ \frac{3}{2(2k-1)-k-4} & \frac{3}{2(2k-1)-k-4} & -\frac{3}{2(2k-1)-k-4} \\ \frac{-k-1}{2(2k-1)-k-4} & -\frac{3}{2(2k-1)-k-4} & \frac{2k-1}{2(2k-1)-k-4} \end{pmatrix}$$

```
(%i6)  ratsimp(%);
(%o6)
```

$$\begin{pmatrix} \frac{2k-1}{3k-6} & \frac{1}{k-2} & -\frac{k+1}{3k-6} \\ \frac{1}{k-2} & \frac{1}{k-2} & -\frac{1}{k-2} \\ -\frac{k+1}{3k-6} & -\frac{1}{k-2} & \frac{2k-1}{3k-6} \end{pmatrix}$$

6.2.1.5 Ejercicio. Calcular los autovalores de $M(k)$.

Solución:

```
(%i7)  eigenvalues(M(k));
(%o7)
```

$$\left[\left[-\frac{\sqrt{k^2-2k+9}-k-1}{2}, \frac{\sqrt{k^2-2k+9}+k+1}{2}, 3\right], [1,1,1]\right]$$

6.2.1.6 Ejercicio. Determinar los k para los que $M(k)$ tiene autovalores múltiples.

Solución:

```
(%i8)  [x,y,z] : %[1];
(%o8)
```

$$\left[-\frac{\sqrt{k^2-2k+9}-k-1}{2}, \frac{\sqrt{k^2-2k+9}+k+1}{2}, 3\right]$$

```
(%i9)  realroots(x=y);
(%o9)
```

$$[k=0]$$

```
(%i10)  solve(x=z);
(%o10)
```

$$\left[k=\sqrt{k^2-2k+9}+5\right]$$

```
(%i12)  solve(y=z);
(%o12)
```

$$\left[k = 5 - \sqrt{k^2 - 2k + 9}\right]$$

Por tanto, sólo para $k = 0$ tiene autovalores múltiples.

6.2.2. Ejercicio 2: Inversas de matrices triangulares

6.2.2.1 Ejercicio. Definir las matrices $A(k)$ (para $k \in \mathbb{N}$) tales que $A(k)$ es la matriz triangular superior de orden $n+1$ cuyo término general es

$$a_{ij} = \begin{cases} \binom{j-1}{i-1} & \text{si } i \leq j \\ 0, & \text{si } i > j \end{cases}$$

Solución:

```
(%i14)  a[i,j] := if i <= j then binomial(j-1,i-1) else 0$
(%i15)  A(n) := genmatrix(a,n+1,n+1);
(%o15)
```

$$A(n) := \text{genmatrix}(a, n+1, n+1)$$

6.2.2.2 Ejercicio. Calcular las matrices $A(1)$, $A(2)$ y $A(5)$.

Solución:

```
(%i16)  A(1);
(%o16)
```

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

```
(%i17)  A(2);
(%o17)
```

$$\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}$$

```
(%i18)  A(5);
```

```
(%o18)
```

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 1 & 3 & 6 & 10 \\ 0 & 0 & 0 & 1 & 4 & 10 \\ 0 & 0 & 0 & 0 & 1 & 5 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

6.2.2.3 Ejercicio. Calcular las inversas de las matrices $A(1)$, $A(2)$ y $A(5)$.

Solución:

```
(%i19)  invert(A(1));
(%o19)
```

$$\begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}$$

```
(%i20)  invert(A(2));
(%o20)
```

$$\begin{pmatrix} 1 & -1 & 1 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{pmatrix}$$

```
(%i21)  invert(A(5));
(%o21)
```

$$\begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 \\ 0 & 1 & -2 & 3 & -4 & 5 \\ 0 & 0 & 1 & -3 & 6 & -10 \\ 0 & 0 & 0 & 1 & -4 & 10 \\ 0 & 0 & 0 & 0 & 1 & -5 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

6.2.2.4 Ejercicio. Conjeturar cuál es la inversa de $A(n)$ y definirla como $B(n)$.

Solución:

```
(%i22)  b[i,j] := (-1)^(i+j)*a[i,j]$
(%i23)  B(n) := genmatrix(b,n+1,n+1)$
(%i24)  B(5);
```

```
(%o24)
```

$$\begin{pmatrix} 1 & -1 & 1 & -1 & 1 & -1 \\ 0 & 1 & -2 & 3 & -4 & 5 \\ 0 & 0 & 1 & -3 & 6 & -10 \\ 0 & 0 & 0 & 1 & -4 & 10 \\ 0 & 0 & 0 & 0 & 1 & -5 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

6.2.2.5 Ejercicio. Comprobar la conjetura para n entre 1 y 10.

Solución:

```
(%i25)  apply ("and", makelist(is(invert(A(n))=B(n)),n,1,10));
(%o25)
```

true

6.2.3. Ejercicio 3: Matrices que conmutan con una dada

El objetivo de este ejercicio es determinar las matrices cuadradas X de orden 2 que conmutan con la matriz

$$A = \begin{pmatrix} 1 & -5 \\ -5 & 3 \end{pmatrix}$$

6.2.3.1 Ejercicio. Definir la matriz

$$A = \begin{pmatrix} 1 & -5 \\ -5 & 3 \end{pmatrix}$$

Solución:

```
(%i26)  A : matrix([1,-5],[-5,3]);
(%o26)
```

$$\begin{pmatrix} 1 & -5 \\ -5 & 3 \end{pmatrix}$$

6.2.3.2 Ejercicio. Definir la matriz X cuyos términos son a, b, c, d.

Solución:

```
(%i27)  kill(a,b,c,d)$
(%i28)  X : matrix([a,b],[c,d]);
```

```
(%o28)
```

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

6.2.3.3 Ejercicio. Definir $M = AX - XA$

Solución:

```
(%i29)  M : A.X-X.A;
(%o29)
```

$$\begin{pmatrix} 5b-5c & -5d-2b+5a \\ 5d+2c-5a & 5c-5b \end{pmatrix}$$

6.2.3.4 Ejercicio. Resolver el sistema lineal de 4 ecuaciones con 4 incógnitas $M = 0$. *Indicación*: Antes de resolverlo, asignarle a la variable `globalsolve` el valor `true`.

Solución:

```
(%i30)  globalsolve : true$
(%i31)  solve([M[1,1],M[1,2],M[2,1],M[2,2]]);
  solve: dependent equations eliminated: (4 3)
(%o31)
```

$$\left[\left[d : \%\text{r2}, a : \frac{5\ \%\text{r2}+2\ \%\text{r1}}{5}, c : \%\text{r1}, b : \%\text{r1}\right]\right]$$

6.2.3.5 Ejercicio. Definir las matrices B que son soluciones de la ecuación $M = 0$.

Solución:

```
(%i32)  B : matrix([(5*v+2*u)/5,u],[u,v]);
(%o32)
```

$$\begin{pmatrix} \frac{5v+2u}{5} & u \\ u & v \end{pmatrix}$$

6.2.3.6 Ejercicio. Comprobar que A y B conmutan.

Solución:

```
(%i33)  is(A.B=B.A);
(%o33)
```

true

Capítulo 7

Gráficos y animaciones

7.1. Ejercicios resueltos

7.1.1. Gráficos en el plano con plot2d

Coordenadas cartesianas

7.1.1.1 Ejercicio. Dibujar la gráfica de $\text{sen}(2x)$ para x entre -2π y 2π.

Solución:

```
(%i1)  plot2d(sin(2*x),[x,-2*%pi,2*%pi])$
```

La gráfica está en la figura 7.1.

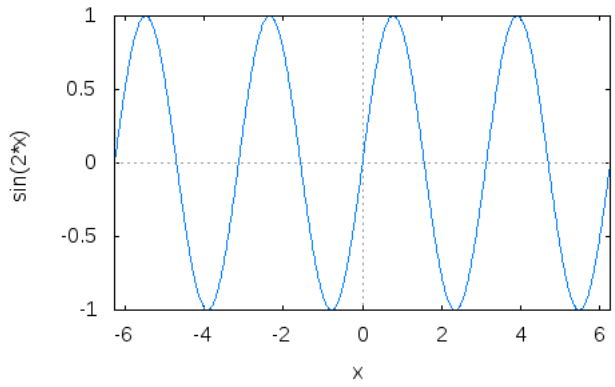

Figura 7.1: Gráfica de $\text{sen}(2x)$

7.1.1.2 Ejercicio. Dibujar las gráficas de x^2 y de $\sqrt{2x}$ para x entre -2 y 2.

Solución:

```
(%i2)  plot2d([x^2,sqrt(2*x)],[x,-2,2])$
```

La gráfica está en la figura 7.2.

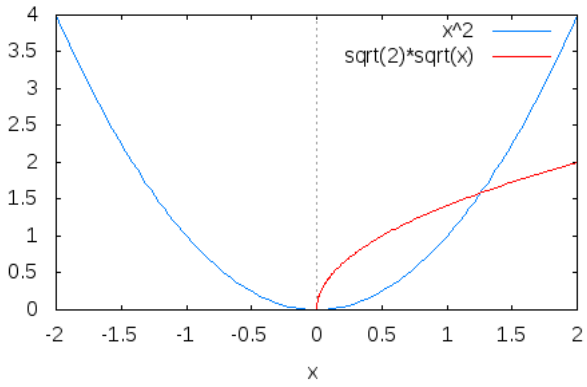

Figura 7.2: Gráfica de x^2 y de $\sqrt{2x}$

7.1.1.3 Ejercicio. Dibujar la gráfica de $\dfrac{x}{x^2-4}$ para x entre -6 y 6 e y entre -6 y 6.

Solución:

```
(%i3)  plot2d(x/(x^2-4),[x,-6,6],[y,-6,6])$
```

La gráfica está en la figura 7.3.

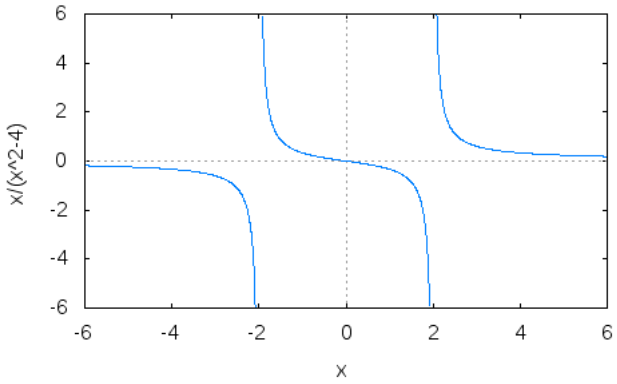

Figura 7.3: Gráfica de $\dfrac{x}{x^2-4}$

7.1.1.4 Ejercicio. Dibujar la gráfica de $\dfrac{x}{x^2-4}$ para x entre -6 y 6 e y entre -6 y 6 con ambos ejes con el mismo tamaño en pantalla.

Solución:

```
(%i4)  plot2d(x/(x^2-4),[x,-6,6],[y,-6,6],
              [gnuplot_preamble, "set size ratio 1;"])$
```

La gráfica está en la figura 7.4.

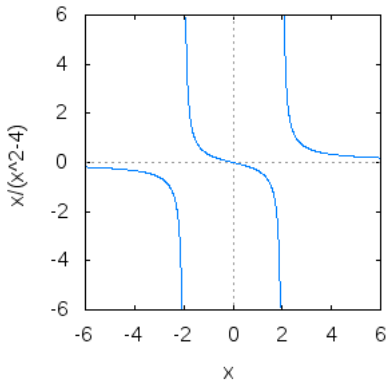

Figura 7.4: Gráfica de $\dfrac{x}{x^2-4}$ con escala unitaria

7.1.1.5 Ejercicio. Dibujar la gráfica de $\dfrac{x}{x^2-4}$ para x entre -6 y 6 e y entre -6 y 6 con una malla.

Solución:

```
(%i5)  plot2d(x/(x^2-4),[x,-6,6],[y,-6,6],
              [gnuplot_preamble, "set grid;"])$
```

La gráfica está en la figura 7.5.

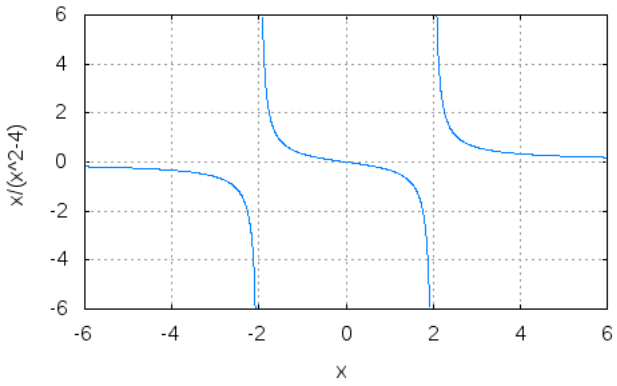

Figura 7.5: Gráfica de $\dfrac{x}{x^2-4}$ con malla

7.1.1.6 Ejercicio. Dibujar las gráficas de $\sqrt{1-x^2}$ y de $-\sqrt{1-x^2}$ para x e y entre -1 y 1.

Solución:

```
(%i6)  plot2d([sqrt(1-x^2),-sqrt(1-x^2)],[x,-1,1],[y,-1,1])$
```

La gráfica está en la figura 7.6.

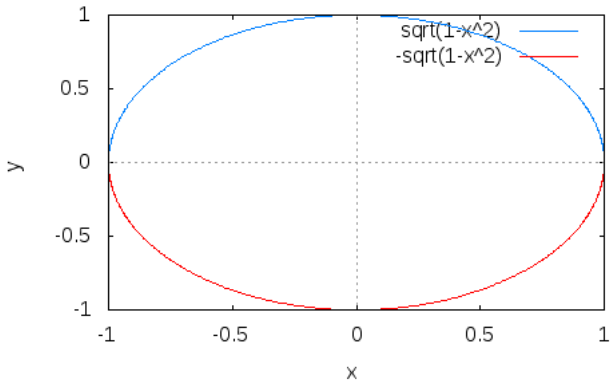

Figura 7.6: Gráfica de $\sqrt{1-x^2}$ y de $-\sqrt{1-x^2}$

7.1.1.7 Ejercicio. Dibujar las gráficas de $\sqrt{1-x^2}$ y de $-\sqrt{1-x^2}$ para x e y entre -1 y 1 con ambos ejes con el mismo tamaño en pantalla.

Solución:

```
(%i7)  plot2d([sqrt(1-x^2),-sqrt(1-x^2)],[x,-1,1],[y,-1,1],
                [gnuplot_preamble, "set size ratio 1;"])$
```

La gráfica está en la figura 7.7.

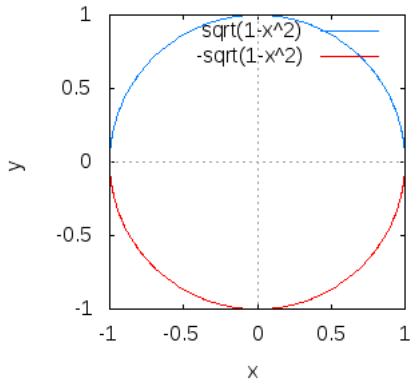

Figura 7.7: Gráfica de $\sqrt{1-x^2}$ y de $-\sqrt{1-x^2}$ con escala unitaria

Gráficas de funciones definidas a trozos

7.1.1.8 Ejercicio. Dibujar la gráfica de la función

$$f(x) = \begin{cases} \sqrt{-x}, & \text{si } x < 0 \\ x^3, & \text{en caso contrario} \end{cases}$$

Solución:

```
(%i8)  f(x):= if x<0 then sqrt(-x) else x^3$
(%i9)  plot2d(f(x),[x,-9,9],[y,-1,6])$
```

La gráfica está en la figura 7.8.

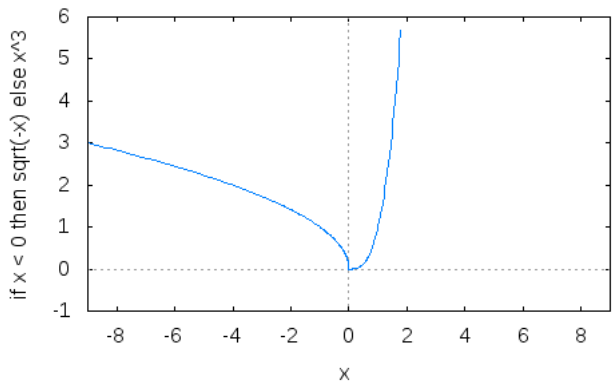

Figura 7.8: Gráfica de función definida a trozos

Gráficos en coordenadas paramétricas

7.1.1.9 Ejercicio. Un astroide es una curva trazada por un punto fijo de un círculo de radio r que rueda sin deslizar dentro de otro círculo fijo de radio $4r$. Sus ecuaciones paramétricas son:

$$\begin{cases} x = \cos(t)^3 \\ y = \text{sen}(t)^3 \end{cases}$$

Dibujar un astroide.

Solución:

```
(%i10)  plot2d(
              ['parametric, cos(t)^3, sin(t)^3,
              [t, 0, 2*%pi],
              [nticks, 300]])$
```

La gráfica está en la figura 7.9.

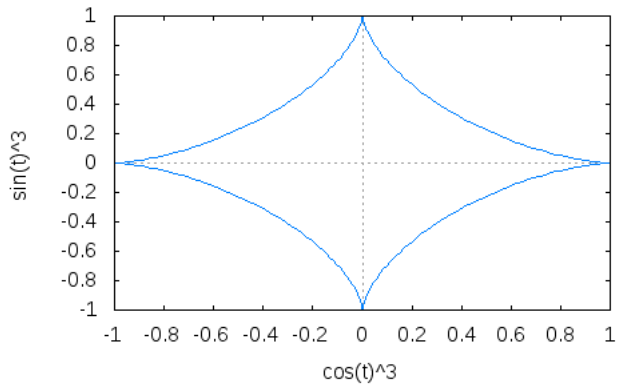

Figura 7.9: Gráfica de un astroide

Gráficas de curvas poligonales

7.1.1.10 Ejercicio. Dibujar el segmento que une los puntos (0,6) y (5,1).

Solución: Una forma es

```
(%i11)  plot2d([discrete,[0,5],[6,1]], [x,-5,5])$
```

Otra forma es

```
(%i12)  plot2d([discrete,[[0,6],[5,1]]], [x,-5,5])$
```

La gráfica está en la figura 7.10.

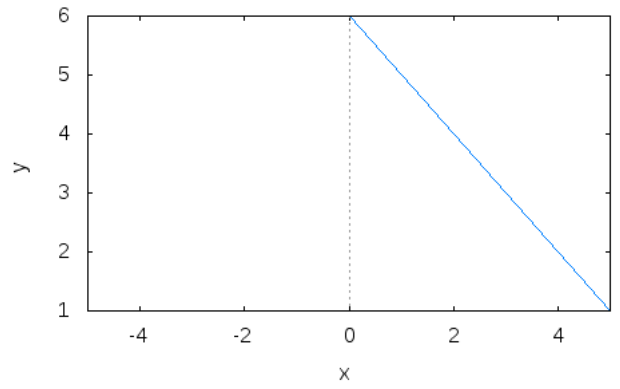

Figura 7.10: Gráfica de un segmento

7.1.1.11 Ejercicio. Dibujar la recta que une los puntos (0,6), (5,1) y (8,3).

Solución: Una forma es

```
(%i13)  plot2d([discrete,[0,5,8],[6,1,3]], [x,-5,10])$
```

Otra forma es

```
(%i14)  plot2d([discrete,[[0,6],[5,1],[8,3]]], [x,-5,10])$
```

La gráfica está en la figura 7.11.

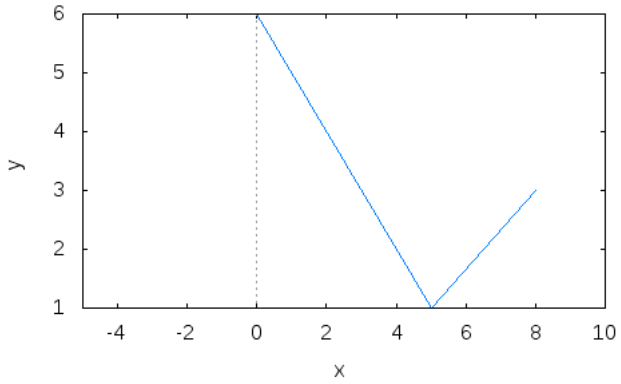

Figura 7.11: Gráfica de un ángulo

7.1.1.12 Ejercicio. Dibujar el rombo de vértices $(-1,0)$, $(0,-2)$, $(1,0)$ y $(0,2)$.

Solución:

```
(%i15)  plot2d([discrete,[-1,0,1,0,-1],[0,-2,0,2,0]], [x,-5,5])$
```

La gráfica está en la figura 7.12.

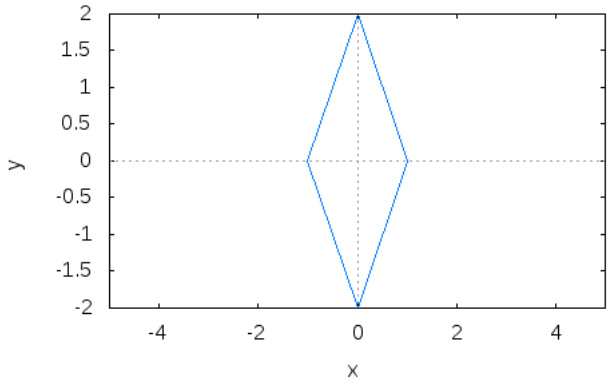

Figura 7.12: Gráfica de un rombo

7.1.1.13 Ejercicio. Dibujar los puntos $(0,0)$, $(2,0)$ y $(2,2)$.

Solución:

```
(%i16)  xy:[[0,0],[2,0],[2,2]]$
(%i17)  plot2d([discrete,xy],[style,points])$
```

La gráfica está en la figura 7.13.

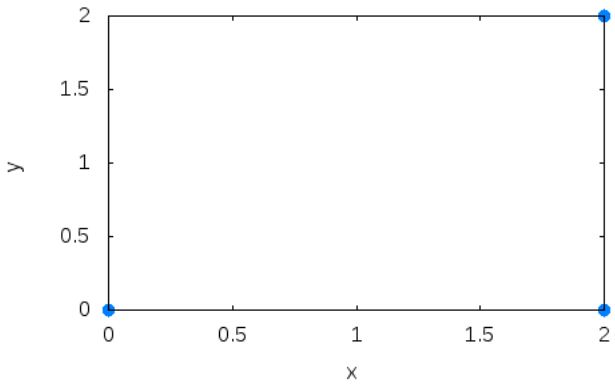

Figura 7.13: Gráfica de tres puntos

7.1.1.14 Ejercicio. Dibujar los puntos (0,0), (2,0) y (2,2) con ancho 10 y color rojo.

Solución:

```
(%i18)  plot2d([discrete,xy],[style,[points,10,2]])$
```

La gráfica está en la figura 7.14.

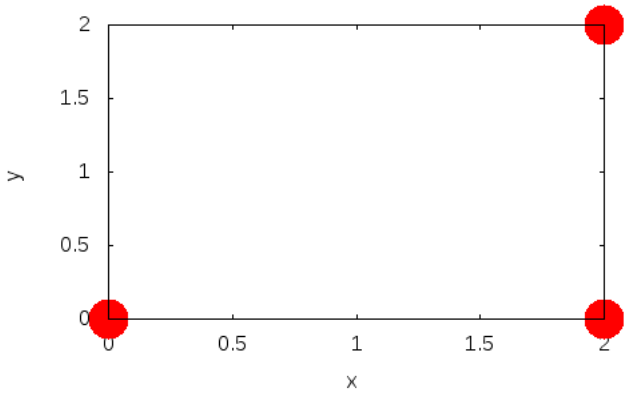

Figura 7.14: Gráfica de tres puntos coloreados

7.1.1.15 Ejercicio. Dibujar el triángulo rectángulo de vértices (0,0), (2,0) y (2,2) con ancho 5 y los lados en rojo.

Solución:

```
(%i19)  plot2d([discrete,[0,2,2,0],[0,0,2,0]],[style,[lines,5,2]])$
```

La gráfica está en la figura 7.15.

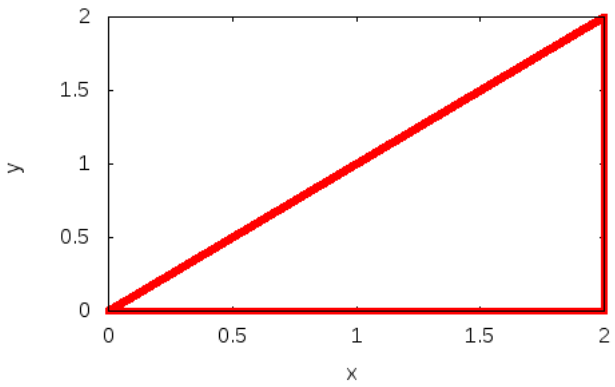

Figura 7.15: Gráfica de un triángulo coloreado

7.1.1.16 Ejercicio. Dibujar el triángulo rectángulo de vértices (0,0), (2,0) y (2,2) con la x entre -3 y 5, la y entre -1 y 3, el ancho 5 y los lados en verde.

Solución:

```
(%i20)  plot2d(
             [discrete,[0,2,2,0],[0,0,2,0]],
             [x,-3,5],
             [y,-1,3],
             [style,[lines,5,3]])$
```

La gráfica está en la figura 7.16.

7.1.1.17 Ejercicio. Dibujar el triángulo rectángulo de vértices (0,0), (2,0) y (2,2) con la x entre -3 y 5, la y entre -1 y 3, el ancho 5 y los lados en rojo y los vértices como puntos azules.

Solución:

```
(%i21)  plot2d(
             [[discrete,[0,2,2,0],[0,0,2,0]],[discrete,xy]],
             [x,-3,5],[y,-1,3],
             [style,[lines,5,2],[points,5,1,1]])$
```

La gráfica está en la figura 7.17.

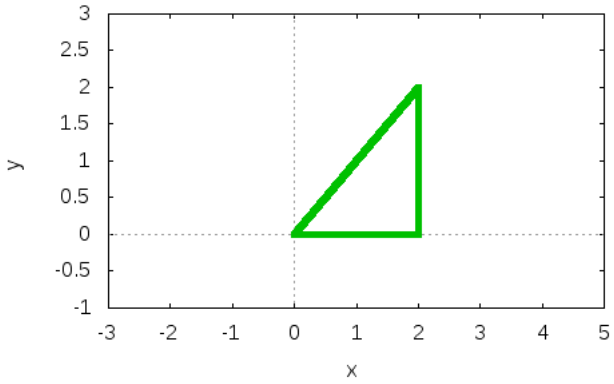

Figura 7.16: Gráfica de un triángulo coloreado en verde

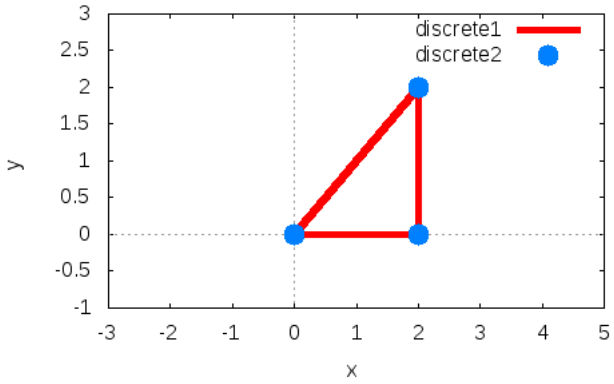

Figura 7.17: Gráfica de un triángulo con lados y vértices coloreados

7.1.2. Gráficos con draw

7.1.2.1 Ejercicio. Cargar el módulo `draw`.

Solución:

```
(%i22)  load(draw)$
```

7.1.2.2 Ejercicio. Definir `coseno` como el objeto que correpondiente al gráfico en azul de la función $\cos(x)$ para x entre 0 y 4π.

Solución:

```
(%i23)  coseno:gr2d(color=blue,
                    explicit(cos(x),x,0,4*%pi))$
```

7.1.2.3 Ejercicio. Dibujar el gráfico de `coseno`.

Solución:

```
(%i24)  draw(coseno)$
```

La gráfica está en la figura 7.18.

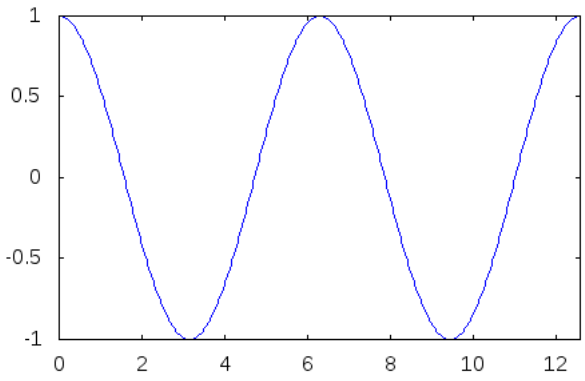

Figura 7.18: Gráfica con *draw*

7.1.2.4 Ejercicio. Dibujar en rojo el gráfico de $\text{sen}(x)$ para x entre 0 y 4π.

Solución:

```
(%i25)  draw2d(color=red,
               explicit(sin(x),x,0,4*%pi))$
```

La gráfica está en la figura 7.19.

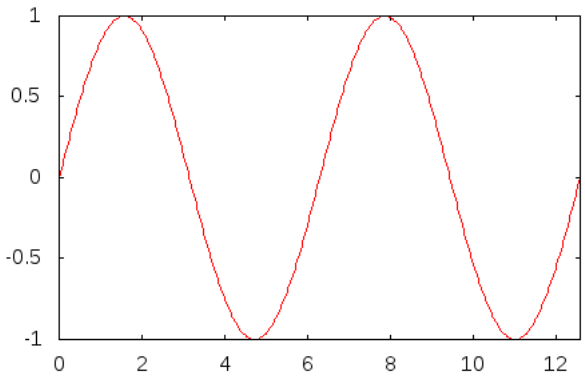

Figura 7.19: Gráfica de sen(x) con *draw*

7.1.2.5 Ejercicio. Dibujar la elipse de ecuación

$$\begin{cases} x = 2\cos(t), \\ y = 5\,\mathrm{sen}(t) \end{cases}$$

Solución: Una forma es definir la elipse y después dibujarla

```
(%i26)  d1: gr2d(title="Elipse",
                 nticks=30,
                 parametric(2*cos(t),5*sin(t),t,0,2*%pi))$
(%i27)  draw(scene1)$
```

La gráfica está en la figura 7.20.

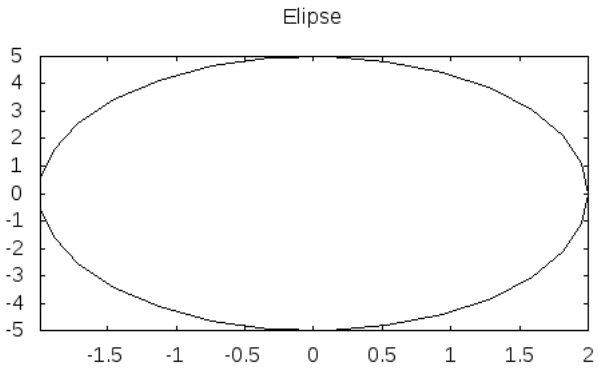

Figura 7.20: Gráfica de elipse

Otra forma es dibujarla directamente

```
(%i28)  draw2d(title="Elipse",
               color=blue,
```

```
                nticks=30,
                parametric(2*cos(t),5*sin(t),t,0,2*%pi))$
```

La gráfica está en la figura 7.21.

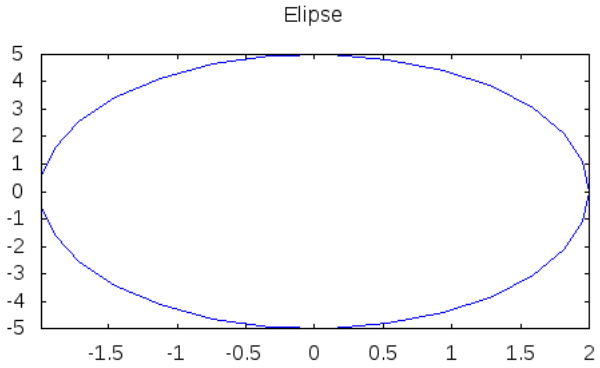

Figura 7.21: Gráfica de elipse

7.1.2.6 Ejercicio. Dibujar las gráficas de x^2, para x entre -1 y 1, y de la curva definida por

$$\begin{cases} x = 2\cos(t), \\ y = 5\,\mathrm{sen}(t) \end{cases}$$

Solución:

```
(%i29)  draw2d(color=red,
                explicit(x^2,x,-1,1),
                color=blue,
                nticks=30,
                parametric(cos(t),sin(t),t,0,2*%pi))$
```

La gráfica está en la figura 7.22.

7.1.2.7 Ejercicio. Dibujar las gráficas de $\cos(x)$, para x entre 0 y 4π, y de x^3, para x entre -5 y 5 en el marco común $[0,\pi] \times [-2,2]$.

Solución:

```
(%i30)  draw2d(color=blue,
                explicit(cos(x),x,0,4*%pi),
                color=red,
                explicit(x^3,x,-5,5),
                xrange=[0,%pi],yrange=[-2,2])$
```

La gráfica está en la figura 7.23.

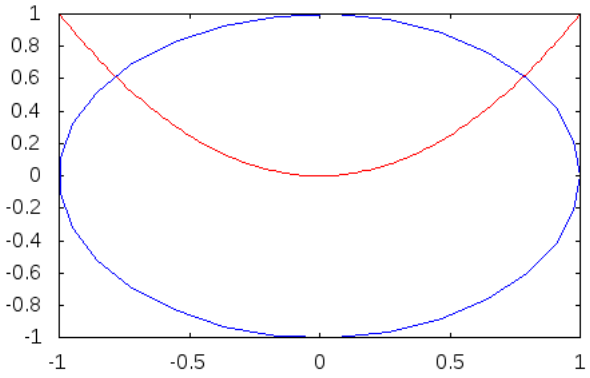

Figura 7.22: Gráfica de dos curvas con *draw*

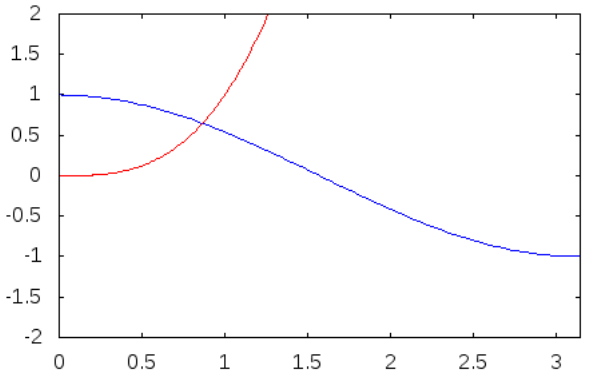

Figura 7.23: Gráfica de dos curvas con *draw* con marco común

7.1.2.8 Ejercicio. Dibujar la gráfica de la función implícita definida por $xy = 1$ para x e y entre -3 y 3. Además dibujar una malla y titular la ventana como "Hipérbola".

Solución:

```
(%i31)  draw2d(color=blue,
               nticks=100,
               implicit(x*y=1,x,-3,3,y,-3,3),
               grid=true,
               title="Hipérbola")$
```

La gráfica está en la figura 7.24.

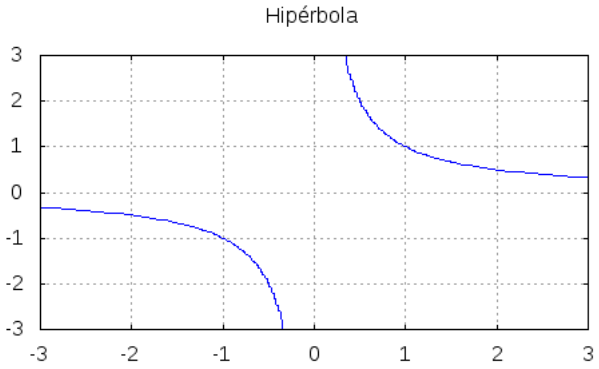

Figura 7.24: Gráfica de una función implícita

7.1.2.9 Ejercicio. Dibujar la gráfica de $e^{x/2}$, para x entre -1 y 2. Además, etiquetar el eje x con "Tiempo", el eje y con "Habitantes" y la ventana con "Evolución de la población".

Solución:

```
(%i32)  draw2d(color=blue,
               explicit(exp(x/2),x,-2,2),
               xlabel="Tiempo",
               ylabel="Habitantes",
               title="Evolución de la población")$
```

La gráfica está en la figura 7.25.

7.1.2.10 Ejercicio. Dibujar la gráfica de $\cos(x)$, para x entre 0 y 10, rellenado de azul la región entre la curva y la parte inferior de la ventana.

Solución:

```
(%i33)  draw2d(fill_color=blue,
               filled_func=true,
```

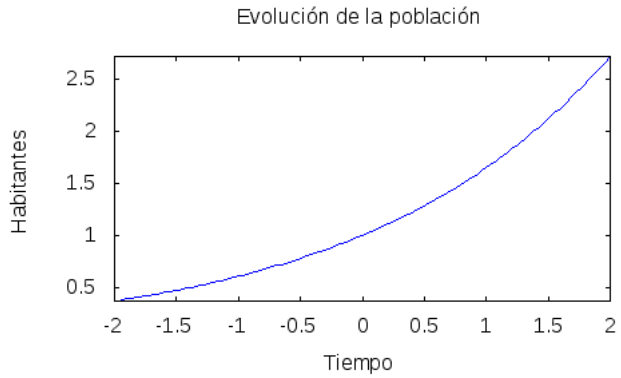

Figura 7.25: Gráfica de la evolución de la población

```
explicit(cos(x),x,0,10))$
```

La gráfica está en la figura 7.26.

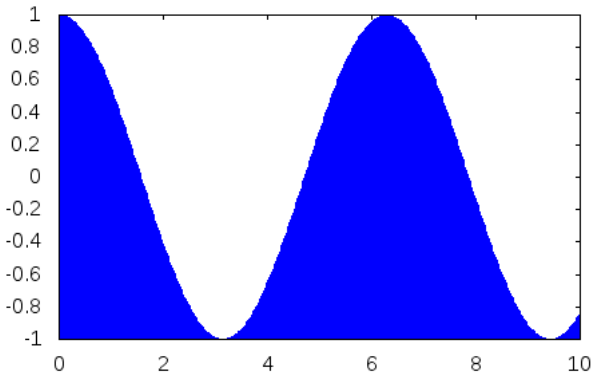

Figura 7.26: Gráfica con relleno

7.1.2.11 Ejercicio. Dibujar la gráfica de $\cos(x)$, para x entre 0 y 10, rellenado de azul la región entre la curva y la gráfica de $\text{sen}(x)$.

Solución:

```
(%i34)  draw2d(fill_color=blue,
               filled_func=sin(x),
               explicit(cos(x),x,0,10))$
```

La gráfica está en la figura 7.27.

7.1.2.12 Ejercicio. Dibujar la gráfica de $\cos(x)$, para x entre 0 y 10, rellenado de azul la región entre la curva y la gráfica de $\text{sen}(x)$. Además, dibujar de rojo y grosor 5 la gráfica

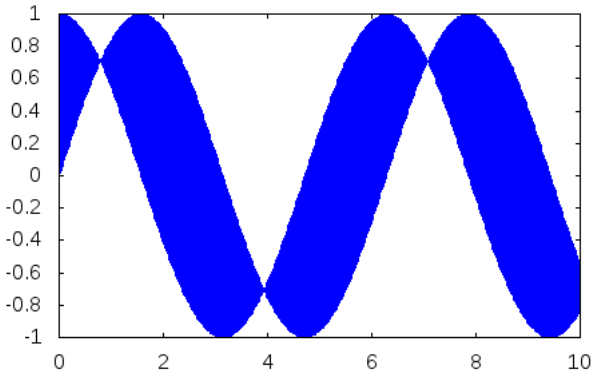

Figura 7.27: Gráfica con relleno entre dos curvas

de sen(x) y de amarillo y grosor 5 la gráfica de cos(x).

Solución:

```
(%i35)  draw2d(filled_func=sin(x),
               fill_color=blue,
               explicit(cos(x),x,0,10),
               filled_func=false,
               color=red,line_width=5,
               explicit(sin(x),x,0,10),
               color=yellow,line_width=5,
               explicit(cos(x),x,0,10))$
```

La gráfica está en la figura 7.28.

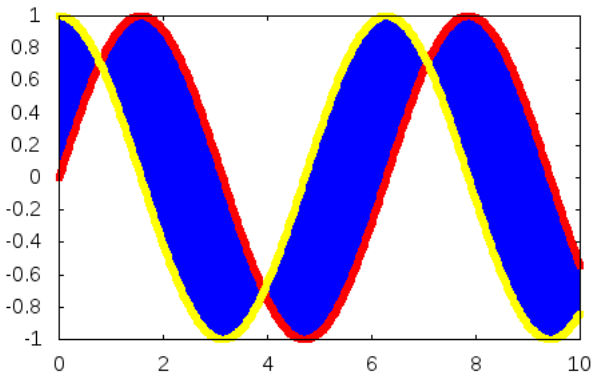

Figura 7.28: Gráfica con relleno entre dos curvas y bordes

7.1.2.13 Ejercicio. Definir la función $f(x) = x^3 - 2x^2 - x + 2$.

Solución:

```
(%i36)  f(x):=x^3-2*x^2-x+2$
```

7.1.2.14 Ejercicio. Definir la función $df(x)$ que es la derivada de $f(x)$.

Solución:

```
(%i37)  define(df(x),diff(f(x),x))$
```

7.1.2.15 Ejercicio. Definir la función $tangente(x,a)$ que es la tangente a $f(x)$ en el punto de abscisa a.

Solución:

```
(%i38)  tangente(x,a):=f(a)+df(a)*(x-a)$
(%i39)  tangente(x,a);
(%o39)
```

$$\left(3\,a^2-4\,a-1\right)\,\left(x-a\right)+a^3-2\,a^2-a+2$$

7.1.2.16 Ejercicio. Dibujar la gráfica de $f(x)$ y su tangente en $x=1$. Además, escribir las leyendas "funcion" para $f(x)$ y "tangente" para su tangente y dibujar la malla.

Solución:

```
(%i40)  draw2d(color=blue,key="funcion",
               explicit(f(x),x,-2,3),
               color=red,key="tangente",
               explicit(tangente(x,1),x,-2,3),
               grid=true)$
```

La gráfica está en la figura 7.29.

7.1.2.17 Ejercicio. Dibujar las gráficas de las funciones implícitas

$$\begin{cases} y^2 = x^3 - 2x + 1 \\ x^3 + y^3 = 3xy^2 - x - 1 \end{cases}$$

Además, la primera dibujarla en azul con la etiqueta $y^2 = x^3 - 2x + 1$, la segunda dibujarla en rojo y con la etiqueta $x^3 + y^3 = 3xy^2 - x - 1$, etiquetar la ventana con "Dos funciones implicitas" y, finalmente, dibujar la malla.

Solución:

```
(%i41)  draw2d(color=blue,
               key="y^2=x^3-2x+1",
```

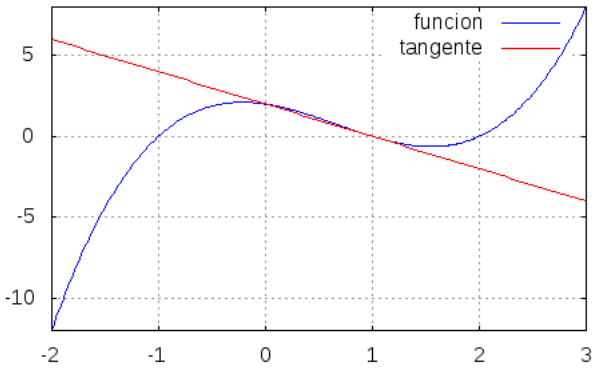

Figura 7.29: Gráfica de la tangente

```
implicit(y^2=x^3-2*x+1, x, -4,4, y, -4,4),
color=red,
key="x^3+y^3 = 3xy^2-x-1",
implicit(x^3+y^3 = 3*x*y^2-x-1, x,-4,4, y,-4,4),
title="Dos funciones implicitas",
grid=true)$
```

La gráfica está en la figura 7.30.

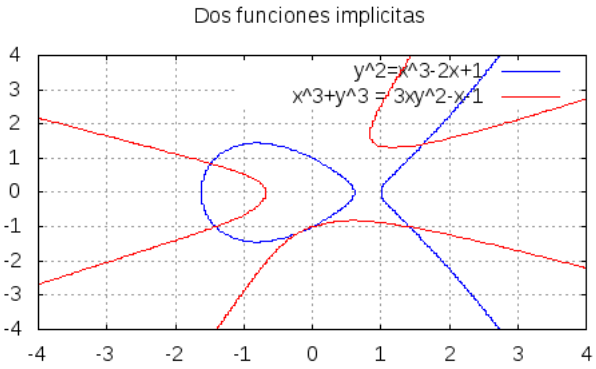

Figura 7.30: Gráfica de dos funciones implícitas

7.1.2.18 Ejercicio. Dibujar los siguientes rectángulos

- el de vértices opuestos (-2,-2) y (6,-1) en verde con los lados punteados on grosor 6
- el de vértices opuestos (9,4) y (2,-1) en rojo con los lados de grosor 2

La ventana es la $[-3,10] \times [-3,4{,}5]$.

Solución:
La gráfica está en la figura 7.31.

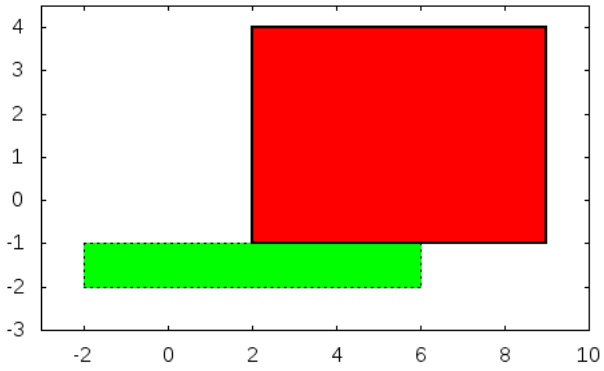

Figura 7.31: Gráfica de rectángulos

7.1.2.19 Ejercicio. Dibujar:

- la elipse de centro (0,6), semieje horizontal de longitud 3, semieje vertical de longitud 2, ángulo inicial 270 y amplitud -270; dibujar el borde de verde con grosor 5 y el relleno de rojo.
- la elipse de centro (2.5,6), semieje horizontal de longitud 2, semieje vertical de longitud 3, ángulo inicial 30 y amplitud -90; dibujar el borde de azul con grosor 5 y sin relleno.

La dimensiones de la ventana son [-3,6] $\times$ [2,9]

Solución:

```
(%i43)  draw2d(fill_color  = red,
               color = green,
               transparent = false,
               line_width  = 5,
               ellipse(0,6,3,2,270,-270),
               transparent = true,
               color = blue,
               line_width  = 3,
               ellipse(2.5,6,2,3,30,-90),
               xrange  = [-3,6],
               yrange  = [2,9] )$
```

La gráfica está en la figura 7.32.

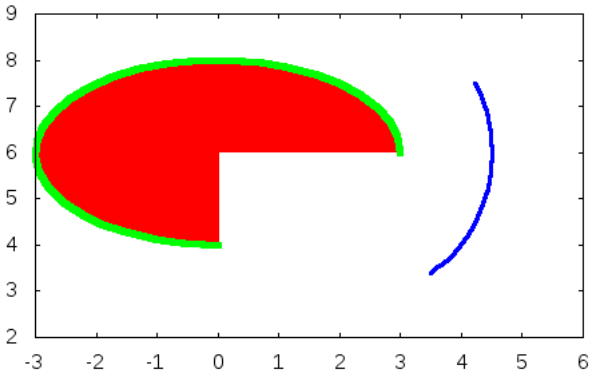

Figura 7.32: Gráfica de elipses

7.1.2.20 Ejercicio. Dibujar 100 puntos aleatorios en [0,10]×[0,10]. Los puntos dibujarlos como círculos azules con grosor 2.

Solución:

```
(%i44)  draw2d(color=blue,
               point_type=filled_circle,
               point_size=2,
               points(makelist([random(10.0),random(10.0)],k,1,150)))$
```

La gráfica está en la figura 7.33.

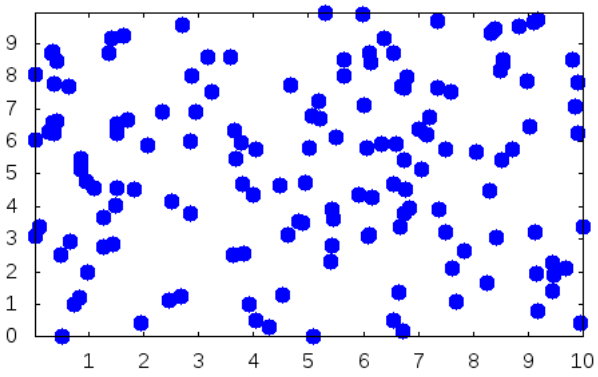

Figura 7.33: Gráfica de puntos aleatorios

7.1.2.21 Ejercicio. Dibujar 10 puntos aleatorios en [0,10]×[0,10]. Los puntos dibujarlos como cuadrados naranjas con grosor 3.

Solución:

```
(%i45)  draw2d(color=orange-red,
```

```
                point_type=filled_square,
                point_size=3,
                points(makelist(random(10.0),k,1,10),
                       makelist(random(10.0),k,1,10)))$
```

La gráfica está en la figura 7.34.

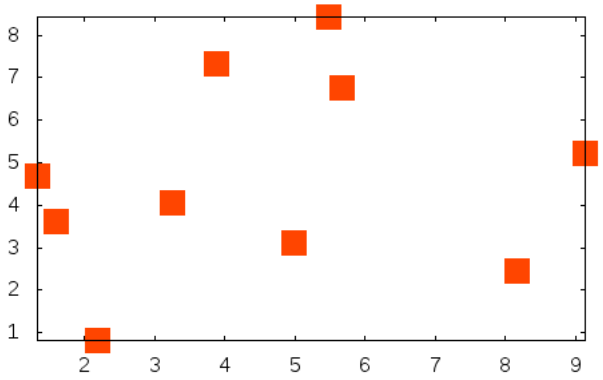

Figura 7.34: Gráfica de puntos aleatorios naranjas

7.1.2.22 Ejercicio. Dibujar en la ventana de dimensiones [0,12]×[0,10]

- el vector de origen (0,1) y desplazamiento (5,5) con cabeza de longitud 1,
- el vector de origen (3,1) y desplazamiento (5,5) con la cabeza vacía y
- el vector de origen (6,1) y desplazamiento (5,5) con cabezas en ambos extremos sin rellenar y punteado.

Solución:

```
(%i46)  draw2d(xrange  = [0,12],
               yrange  = [0,10],
               head_length = 1,
               vector([0,1],[5,5]),
               head_type = 'empty,
               vector([3,1],[5,5]),
               head_both = true,
               head_type = 'nofilled,
               line_type = dots,
               vector([6,1],[5,5]))$
```

La gráfica está en la figura 7.35.

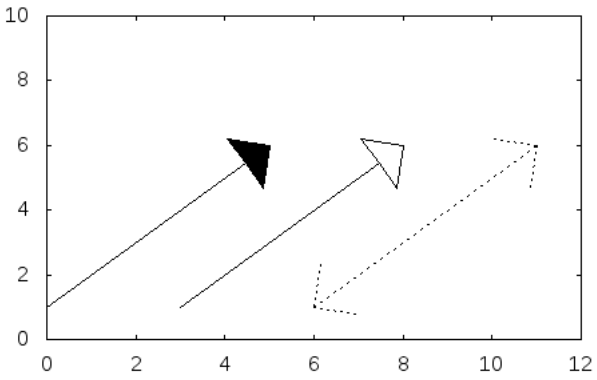

Figura 7.35: Gráfica de vectores

7.1.3. Animaciones gráficas

Animaciones del seno

7.1.3.1 Ejercicio. Crear una animación de la función $\text{sen}(x+n)$, donde el parámetro n va a tomar los valores desde 1 a 20 y el marco es $[-2\pi, 2\pi] \times [-1,1,1,1]$.

Solución:

```
(%i47) with_slider(n,
                   makelist(i,i,1,20),
                   sin(x+n),
                   [x,-2*%pi,2*%pi],[y,-1.1,1.1])$
```

La gráfica inicial está en la figura 7.36.

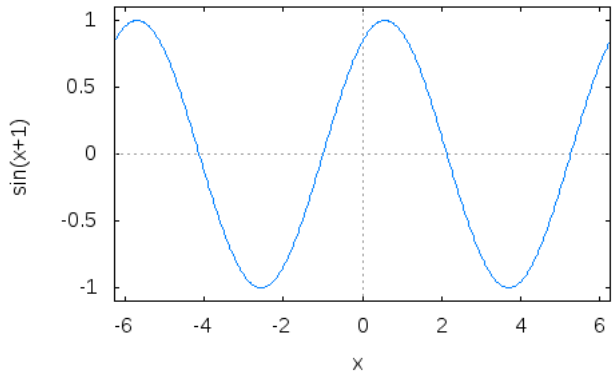

Figura 7.36: Gráfica inicial de $\text{sen}(x+n)$

7.1.3.2 Ejercicio. Crear una animación de la función $\text{sen}(nx)$, donde el parámetro n va a tomar los valores desde 1 a 20 y el marco es $[-2\pi, 2\pi] \times [-1,1,1,1]$.

Solución:

```
(%i48)  with_slider(n,
                    makelist(i,i,1,20),
                    sin(x*n),
                    [x,-2*%pi,2*%pi],[y,-1.1,1.1])$
```

La gráfica inicial está en la figura 7.37.

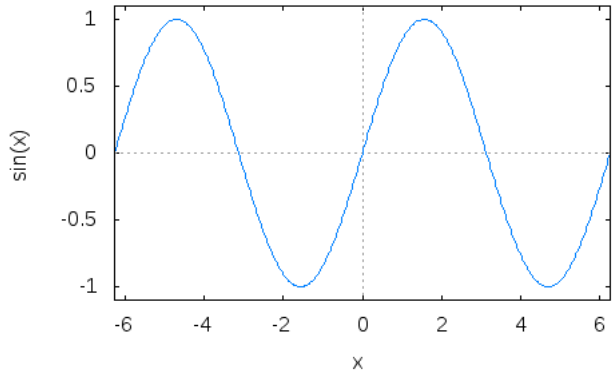

Figura 7.37: Gráfica inicial de sen(xn)

Secantes y tangentes a una parábola

7.1.3.3 Ejercicio. Definir la función f tal que $f(x) = \dfrac{x^2}{4}$.

Solución:

```
(%i49)  f(x):= x^2/4;
(%o49)
```

$$f(x) := \frac{x^2}{4}$$

7.1.3.4 Ejercicio. Dibujar la gráfica de f.

Solución:

```
(%i50)  plot2d([f(x)], [x,-5,5])$
```

La gráfica está en la figura 7.38.

7.1.3.5 Ejercicio. Definir la función *linea* tal que $linea(m,a)$ es la recta de pendiente m que pasa por el punto $(a, f(a))$.

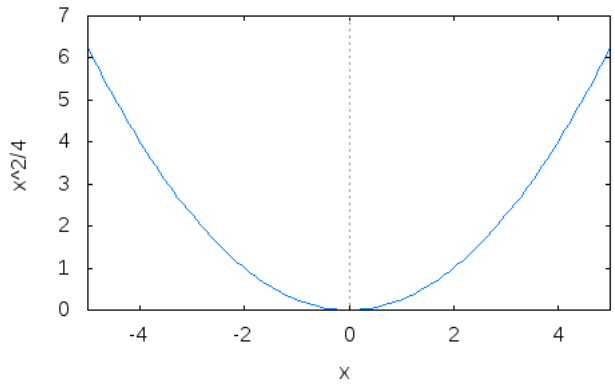

Figura 7.38: Gráfica de la función $f(x) = x^2/4$

Solución:

```
(%i51)  linea(m,a):= m*(x-a)+f(a);
(%o51)
```

$$\text{linea}\,(m,a) := m\ (x-a) + f\,(a)$$

7.1.3.6 Ejercicio. Calcular $linea(0{,}5, 2)$.

Solución:

```
(%i52)  linea(0.5,2);
(%o52)
```

$$0{,}5\ (x-2) + 1$$

7.1.3.7 Ejercicio. Dibujar las gráficas de f y $linea(0{,}5, 2)$ para x entre -4 y 4.

Solución:

```
(%i53)  plot2d([f(x),linea(0.5,2)], [x,-4,4])$
```

La gráfica está en la figura 7.39.

7.1.3.8 Ejercicio. Crear una animación de las funciones $f(x)$ y $linea(m, 2)$, donde el parámetro m va a tomar los valores $-1 + 0{,}2i$ para i desde 0 a 20 y el marco es [0,4] $\times$ [-1,4].

Solución:

```
(%i54) with_slider(
         m,                              /* parámetro */
```

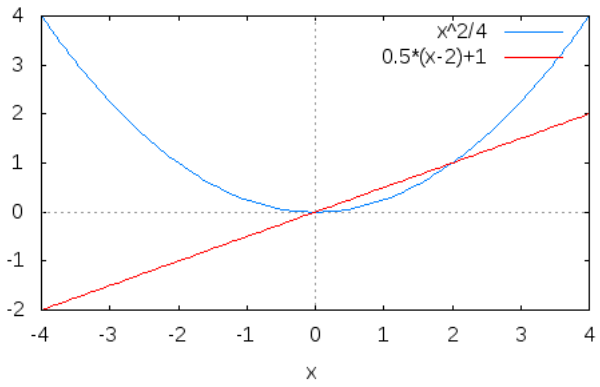

Figura 7.39: Gráfica de la función $f(x) = x^2/4$ y una secante

```
makelist(-1+0.2*i,i,0,20), /* valores del parámetro */
[f(x),linea(m,2)],         /* funciones a representar*/
[x,0,4],[y,-1,4])$
```

La gráfica inicial está en la figura 7.40.

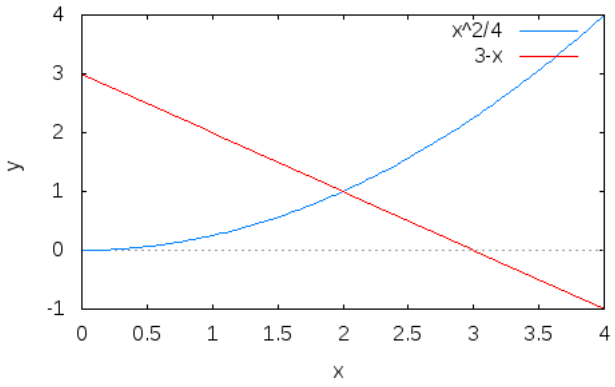

Figura 7.40: Gráfica inicial de las secantes

7.1.3.9 Ejercicio. Repetir la animación anterior usando `with_slider_draw`.

Solución:

```
(%i55)  with_slider_draw(
          m,
          makelist(-1+0.2*i,i,0,20),
          color=blue,
          explicit(f(x),x,0,4),
          color=red,
          explicit(linea(m,2),x,0,4),
```

```
            xrange=[0,4], yrange=[-1,4])$
```

7.1.3.10 Ejercicio. Crear una animación que dibuje las tangentes a la parábola $f(x)$.

Solución:

```
(%i56)  with_slider_draw(
          m,
          makelist(-2+0.2*i,i,0,20),
          color=blue,
          explicit(f(x),x,-4,4),
          color=red,
          explicit(linea(m,2*m),x,-4,4),
          xrange=[-4,4], yrange=[-1,4])$
```

La gráfica inicial está en la figura 7.41.

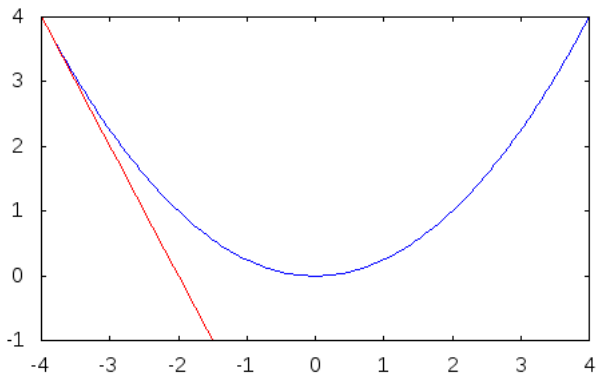

Figura 7.41: Gráfica inicial de las tangentes

Polinomios de Taylor

7.1.3.11 Ejercicio. Definir la función $f(x) = \text{sen}(x) + \cos(x)$.

Solución:

```
(%i57)  kill(f)$
(%i58)  f(x):= sin(x) + cos(x)$
```

7.1.3.12 Ejercicio. Definir las funciones `polinomio_taylor`, para k entre 1 y 20, tales que `polinomio_taylor[k]` es el polinomio de los k primeros términos del polinomio de Taylor de $f(x)$ en $x = 0$. Por ejemplo,

```
(%i1) polinomio_taylor[7](x);
(%o1) 1+x-x^2/2-x^3/6+x^4/24+x^5/120-x^6/720-x^7/5040+...
```

Solución:

```
(%i59)  makelist(define(polinomio_taylor[k](x),
                        trunc(taylor(f(x),x,0,k))),k,1,20)$
(%i60)  polinomio_taylor[7](x);
(%o60)
```

$$1+x-\frac{x^2}{2}-\frac{x^3}{6}+\frac{x^4}{24}+\frac{x^5}{120}-\frac{x^6}{720}-\frac{x^7}{5040}+\cdots$$

7.1.3.13 Ejercicio. Crear una animación de las funciones $f(x)$ y `polinomio_taylor[k](x)` , donde el parámetro k varía desde 1 a 20 en la ventana de dimensiones [-10,10] $\times$ [-2,3].

Solución:

```
(%i61) with_slider_draw(
         k,
         makelist(n,n,1,20),
         line_width=2, color=blue, key="función",
         explicit(f(x),x,-10,10),
         color=red, key=sconcat("polinomio de taylor ",k),
         explicit(polinomio_taylor[k](x),x,-10,10),
         xaxis=true,yaxis=true,
         yrange=[-3,3])$
```

La gráfica inicial está en la figura 7.42.

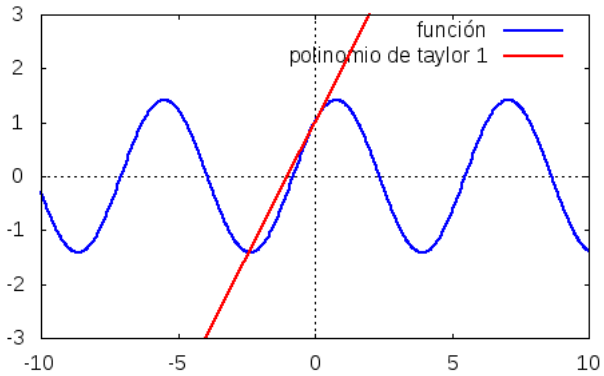

Figura 7.42: Gráfica inicial de polinomios de Taylor

Capítulo 8

Misceláneas de ejercicios

8.1. Ejercicios propuestos

8.1.1. Suma de los enteros menores de 1000 que son múltiplos de 3 ó 5

8.1.1.1 Ejercicio. (Problema 1 del Proyecto Euler) Los números naturales menores que 10 que son múltiplos de 3 ó 5 son 3, 5, 6 y 9. La suma de estos múltiplos es 23.

Definir la función *euler*1 tal que *euler*1(n) es la suma de todos los múltiplos de 3 ó 5 menores que n. Por ejemplo,

$$euler1(10) = 23$$

Calcular la suma de todos los múltiplos de 3 ó 5 menores que 1000.

Solución:

```
(%i1)  euler1(n) := block([s:0],
         for i:3 while i<n do
            (if (mod(i,3)=0 or mod(i,5)=0) then s:s+i),
         s)$
(%i2)  euler1(1000);
(%o2)
```

233168

8.1.2. Suma de los términos pares de la sucesión de Fibonacci menores que 4.000.000

8.1.2.1 Ejercicio. (Problema 2 del Proyecto Euler) Cada término de la sucesión de Fibonacci se obtiene sumando los dos anteriores. Comenzando con 1 y 2, los 10 primeros

términos son

$$1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$$

Definir la función *euler*2 tal que *euler*2(n) es la suma de los términos pares $(2, 8, 34, \ldots)$ de la sucesión de Fibonacci que son menores que n. Por ejemplo,

$$euler2(40) = 2 + 8 + 34 = 44$$

Calcular *euler*2(4000000).

Solución: La definición recursiva de la sucesión de Fibonacci es

```
(%i3)  fib[0] : 1$
(%i4)  fib[1] : 2$
(%i5)  fib[n] := fib[n-1]+fib[n-2]$
```

La definición de *euler*2 es

```
(%i6)  euler2(n) := block ([s:0, f],
         for i:1 do
            (f:fib[i],
             if f>= n then return(s)
             elseif evenp(f) then s:s+f),
         s)$
(%i7)  euler2(40);
(%o7)
```

$$44$$

Otra definición es

```
(%i8)  euler2b(n) := block ([s:0],
         for i:1 while fib[i]<n do
            (if evenp(fib[i]) then s:s+fib[i]),
         s)$
(%i9)  euler2b(40);
(%o9)
```

$$44$$

El cálculo es

```
(%i10)  euler2(4000000);
(%o10)
```

$$4613732$$

```
(%i11)  euler2b(4000000);
```

```
(%o11)
```

4613732

8.1.3. Mayor factor primo de un número

8.1.3.1 Ejercicio. (Problema 3 del Proyecto Euler) Definir la función $euler3$ tal que $euler3(n)$ es el mayor factor primo de n. Por ejemplo,

$$euler3(18) = 3$$

¿Cuál es el mayor factor primo del número 600851475143?

Solución:

```
(%i12)  euler3(n) := first(last(ifactors(n)))$
(%i13)  euler3(13195);
(%o13)
```

29

```
(%i14)  euler3(600851475143);
(%o14)
```

6857

8.1.4. Mayor capicúa producto de números de n cifras

8.1.4.1 Ejercicio. (Problema 4 del Proyecto Euler) Un capicúa es un número que es igual leído de izquierda a derecha que de derecha a izquierda. El mayor capicúa formado por el producto de dos números de 2 cifras es $9009 = 91 \times 99$.

Definir la función $euler4$ tal que $euler4(n)$ es el mayor capicúa que es el producto de dos números de n cifras. Por ejemplo,

$$euler4(2) = 9009$$

Calcular $euler4(3)$.

Solución: Se usa la librería de procesamiento de cadenas.

```
(%i15)  load(stringproc)$
```

Se define la función auxiliar $capicua$ tal que $capicua(n)$ se verifica si n es capicúa. Por ejemplo,

$$capicua(232) = true$$

```
(%i16)  capicua(n) := block([c:string(n)],
          sequal(c,sreverse(c)))$
```

La definición de *euler*4 es

```
(%i17)  euler4(n) := block([m:0,c],
          for a:10^(n-1) thru 10^n-1 do
           for b:10^(n-1) thru a do
            (c:a*b,
             if capicua(c) then m:max(m,c)),
           m)$
(%i18)  euler4(2);
(%o18)
```

$$9009$$

Vamos a dar otra definición sin usar cadenas.
$inverso(n)$ es el número obtenido escribiendo n en orden inverso. Por ejemplo,

$$inverso(305) = 503$$

```
(%i19)  inverso(n) := block([x:0],
          for i:n next quotient(i,10) while i>0 do
             x : 10*x+mod(i,10),
          x)$
```

$capicua1(n)$ se verifica si n es capicúa. Por ejemplo,

$$capicua1(303) = true \quad capicua1(304) = false$$

```
(%i20)  capicua1(n) := is(n = inverso(n))$
```

La segunda definición de *euler*4 es

```
(%i21)  euler4a(n) := block([m:0,c],
          for a:10^(n-1) thru 10^n-1 do
           for b:10^(n-1) thru a do
            (c:a*b,
             if capicua1(c) then m:max(m,c)),
           m)$
```

8.1.5. Menor número divisible por los números de un intervalo

8.1.5.1 Ejercicio. (Problema 5 del Proyecto Euler) Definir la función *euler*5 tal que $euler5(a, b)$ es el menor número divisible por los números desde a hasta b. Por ejemplo,

$$euler5(1, 10) = 2520$$

Calcular el menor número divisible por los números del 1 al 20.

Solución:

```
(%i22)  load("functs")$
(%i23)  euler5(a,b) := apply(lcm,makelist(i,i,a,b))$
(%i24)  euler5(1,10);
(%o24)
                                2520

(%i25)  euler5(1,20);
(%o25)
                              232792560
```

8.1.6. Número de cifras

8.1.6.1 Ejercicio. Definir, por recursión, la función $numeroCifrasR$ tal que $numeroCifrasR(n)$ es el número de cifras de n. Por ejemplo,

$$numeroCifrasR(42704) = 5$$

Solución:

```
(%i26)  numeroCifrasR(n) :=
          if n<10 then 1
          else 1+numeroCifrasR(quotient(n,10))$
(%i27)  numeroCifrasR(42704);
(%o27)
                                5
```

Solución:

8.1.6.2 Ejercicio. Definir, por iteración, la función $numeroCifrasI$ tal que $numeroCifrasI(n)$ es el número de cifras de n. Por ejemplo,

$$numeroCifrasI(42704) = 5$$

Solución:

```
(%i28)  numeroCifrasI(n) := block([x:0],
          while n>= 10 do
             (x:x+1,
              n:quotient(n,10)),
          x+1)$
(%i29)  numeroCifrasI(42704);
(%o29)
                                5
```

8.1.7. Imagen inversa de un número

8.1.7.1 Ejercicio. Definir, por recursión, la función *imagenR* tal que $imagenR(n)$ es el número obtenido escribiendo las cifras de n de derecha a izquierda. Por ejemplo,

$$imagenR(37052) = 25073$$

Solución:

```
(%i30) imagenR(n) :=
         if n<10 then n
         else mod(n,10)*10^(numeroCifrasR(n)-1) +
              imagenR(quotient(n,10))$
(%i31)  imagenR(37052);
(%o31)
                              25073
```

8.1.7.2 Ejercicio. Definir, por iteración, la función *imagenI* tal que $imagenI(n)$ es el número obtenido escribiendo las cifras de n de derecha a izquierda. Por ejemplo,

$$imagenI(37052) = 25073$$

Solución:

```
(%i32)  imagenI(n) := block([x:0,k:numeroCifrasI(n)-1],
          while n>10 do
             (x:x+mod(n,10)*10^k,
              n:quotient(n,10),
              k:k-1),
          n+x)$
(%i33)  imagenI(37052);
```

```
(%o33)
```

25073

8.1.8. Cuadrado de la suma menos la suma de los cuadrados

8.1.8.1 Ejercicio. (Problema 6 del proyecto Euler) El cuadrado de la suma de los 10 primeros números es

$$(1+2+...+10)^2 = 55^2 = 3025$$

La suma de los cuadrados de los 10 primeros números es

$$1^2+2^2+...+10^2 = 385$$

Luego, la diferencia entre el cuadrado de la suma de los 10 primeros números y la suma de los cuadrados de los 10 primeros números es $3025-385 = 2640$.

Definir la función *euler*6 tal que $euler6(n)$ es la diferencia entre el cuadrado de la suma de los n primeros números y la suma de los cuadrados de los n primeros números. Por ejemplo,

$$euler6(10) = 2640$$

Usando *euler*6, calcular la diferencia entre el cuadrado de la suma de los 100 primeros números y la suma de los cuadrados de los 100 primeros números.

Solución:

```
(%i34)  euler6(n) := sum(i,i,1,n)^2 - sum(i^2,i,1,n)$
(%i35)  euler6(10);
(%o35)
```

2640

```
(%i36)  euler6(100);
(%o36)
```

25164150

8.1.9. Terna pitagórica de suma dada

8.1.9.1 Ejercicio. (Problema 9 del proyecto Euler) Una terna pitagórica es una terna de números naturales (a,b,c) tal que $a<b<c$ y $a^2+b^2=c^2$. Por ejemplo $(3,4,5)$ es una terna pitagórica.

Definir la función *ternaPitagoricaSuma* tal que $ternaPitagoricaSuma(x)$ es una terna pitagórica cuya suma es x. Por ejemplo,

$$ternaPitagoricaSuma(12) = [3,4,5]$$

Existe exactamente una terna pitagórica tal que $a+b+c = 1000$. Calcular el producto abc.

Solución: $ternaPitagoricaSuma(x)$ es una terna pitagórica cuya suma es x.

```
(%i37)  ternaPitagoricaSuma(x) := block([c,s],
          for a:1 thru x do
             for b:a+1 thru x-a do
                (c:x-a-b,
                 if is(a^2+b^2=c^2) then s:[a,b,c]),
          s)$
(%i38)  ternaPitagoricaSuma(12);
(%o38)
                                [3,4,5]
```

El cálculo se hará de dos formas. La primera forma de calcularlo es

```
(%i39)  ([a,b,c]:ternaPitagoricaSuma(1000),a*b*c);
(%o39)
                                31875000
```

La segunda forma de calcularlo es

```
(%i40)  apply("*",ternaPitagoricaSuma(1000));
(%o40)
                                31875000
```

8.1.10. Suma de primos menores que uno dado

8.1.10.1 Ejercicio. (Problema 10 del Proyecto Euler) Definir la función *sumaPrimoMenores* tal que $sumaPrimoMenores(n)$ es la suma de los primos menores que n. Por ejemplo,

$$sumaPrimoMenores(10) = 17$$

Solución:

```
(%i41)  sumaPrimoMenores(n) := block([x:0,s:0],
          while x<n do
```

```
                (s:s+x,
                 x:next_prime(x)),
              s)$
(%i42)  sumaPrimoMenores(10);
(%o42)
                                17
```

8.1.11. Menor número triangular con más de n divisores

8.1.11.1 Ejercicio. (Problema 12 del Proyecto Euler) La sucesión de los números triangulares se obtiene sumando los números naturales. Así, el 7^o número triangular es

$$1+2+3+4+5+6+7=28$$

Los primeros 10 números triangulares son

$$1,3,6,10,15,21,28,36,45,55,\ldots$$

Los divisores de los primeros 7 números triangulares son:

$$\begin{array}{rl} 1: & 1 \\ 3: & 1,3 \\ 6: & 1,2,3,6 \\ 10: & 1,2,5,10 \\ 15: & 1,3,5,15 \\ 21: & 1,3,7,21 \\ 28: & 1,2,4,7,14,28 \end{array}$$

Como se puede observar, 28 es el menor número triangular con más de 5 divisores.

Definir la función *euler*12 tal que *euler*12(n) es el menor número triangular con más de n divisores. Por ejemplo,

$$euler12(5)=28$$

Solución: En primer lugar, se define la función *nDivisores* tal que *nDivisores*(n) es el número de divisores de n. Por ejemplo, *nDivisores*$(28)=6$.

```
(%i44)  nDivisores(x) := block([n:0],
          for y:1 thru x do
             (if is(mod(x,y)=0) then n:n+1),
          n)$
(%i45)  nDivisores(28);
(%o45)
                                6
```

La definición de *euler*12 es

```
(%i46)  euler12(n) := block ([s:0],
          for x:1 while (nDivisores(s)<=n) do
              s:s+x,
          s)$
(%i47)  euler12(5);
(%o47)
```

$$28$$

8.1.12. Número de puntos dentro del círculo de radio n

8.1.12.1 Ejercicio. Definir la función *circulo* tal que $circulo(n)$ es el la cantidad de pares de números naturales (x, y) que se encuentran dentro del círculo de radio n. Por ejemplo,

$$circulo(3) = 9$$
$$circulo(4) = 15$$
$$circulo(5) = 22$$

Solución:

```
(%i48)  circulo(n) := block([s:0],
          for x:0 while x<n do
             (for y:0 while x^2+y^2<n^2 do s:s+1),
          s)$
(%i49)  circulo(5);
(%o49)
```

$$22$$

8.2. Ejercicios de exámenes

8.2.1. Primo que ocupa el lugar n

8.2.1.1 Ejercicio. Definir la función *primo* tal que $primo(n)$ es el n–ésimo número primo. Por ejemplo,

$$primo(5) = 11$$

Solución: La sucesión de los primos es

```
(%i50)  primo[1] : 2$
(%i51)  primo[n] := next_prime(primo[n-1])$
(%i52)  primo[5];
(%o52)  11
```

La definición recursiva de *primo* es

```
(%i53)  primo(n) :=
          if n= 1 then 2 else next_prime(primo(n-1))$
```

La definición iterativa de *primo* es

```
(%i54)  primo(n) := block([x:1],
          for i:1 thru n do x:next_prime(x),
          x)$
(%i55)  primo(5);
(%o55)
```

$$11$$

8.2.2. Suma de las cifras de un número

8.2.2.1 Ejercicio. Definir la función *sumaCifras* tal que $sumaCifras(n)$ es la suma de las cifras del número n. Por ejemplo,

$$sumaCifras(325) = 10$$

Solución:

```
(%i56)  sumaCifras(n) :=
          if n<10 then n
          else sumaCifras(quotient(n,10)) + mod(n,10)$
(%i57)  sumaCifras(325);
(%o57)
```

$$10$$

8.2.3. Primos con suma par

8.2.3.1 Ejercicio. Definir la función *numeroPrimosSumaPar* tal que $numeroPrimosSumaPar(n)$ es la cantidad de elementos del conjunto de los n primeros primos tales que la suma de sus cifras es par. Por ejemplo,

$$numeroPrimosSumaPar(10) = 5$$

Solución: En primer lugar se define la función *primosSumaPar* tal que $primosSumaPar(n)$ es la lista de elementos del conjunto de los n primeros primos tales que la suma de sus cifras es par

```
(%i58)  primosSumaPar(n) := block([x:[]],
          if n=0 then x
          elseif evenp(sumaCifras(primo(n)))
                 then cons(primo(n),primosSumaPar(n-1))
          else primosSumaPar(n-1))$
(%i59)  primosSumaPar(20);
(%o59)
```

$$[71,59,53,37,31,19,17,13,11,2]$$

La definición recursiva de *numeroPrimosSumaPar* es

```
(%i60) numeroPrimosSumaPar(n) :=
         if n=0 then 0
         elseif evenp(sumaCifras(primo(n)))
                then 1+numeroPrimosSumaPar(n-1)
         else numeroPrimosSumaPar(n-1)$
(%i61) numeroPrimosSumaPar(10);
(%o61)
```

$$5$$

La definición iterativa de *numeroPrimosSumaPar* es

```
(%i62)  numerosPrimosSumaPar2(n) := block([x:0],
          for i:1 thru n do
             (if evenp(sumaCifras(primo(n))) then x:1+x),
          x)$
```

8.2.3.2 Ejercicio. Definir función *puntos* tal que $puntos(n)$ es la lista de los puntos de la forma $[x,y]$ donde x toma los valores $0,10,20,\ldots,10n$ e y es la cantidad de elementos del conjunto de los x primeros primos tales que la suma de sus cifras es par. Por ejemplo,

$$puntos(5) = [[0,0],[10,5],[20,10],[30,17],[40,21],[50,23]]$$

Solución:

```
(%i63)  puntos(n) :=
          makelist([10*i,numeroPrimosSumaPar(10*i)],i,0,n)$
(%i64)  puntos(5);
(%o64)
```

$$[[0,0],[10,5],[20,10],[30,17],[40,21],[50,23]]$$

8.2.3.3 Ejercicio. Dibujar en el mismo marco la gráfica de $y = x/2$ y los puntos de $puntos(10)$ para x entre 0 y 100.

Solución:

```
(%i65)  plot2d([[discrete,puntos(10)],x/2],
                [x,0,100],
                [style,points,line])$
```

La gráfica está en la figura 8.1.

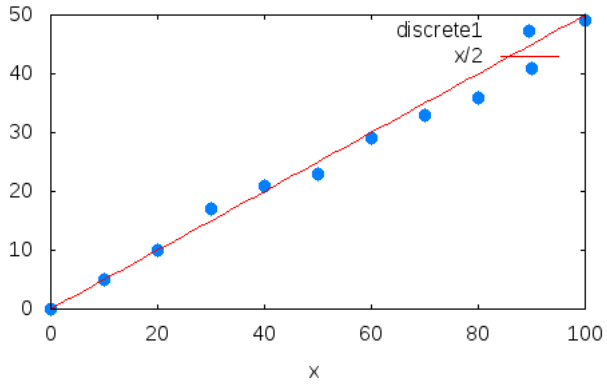

Figura 8.1: Gráfica de primos de suma par

Como se observa, el número de primos de suma par es la mitad del número de primos.

8.2.4. Aproximación de π

8.2.4.1 Ejercicio. Definir la función *calculaPi* tal que $calculaPi(n)$ es la aproximación del número π calculada mediante la expresión

$$4\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\ldots+\frac{(-1)^n}{2n+1}\right)$$

Por ejemplo,

$$\begin{aligned} calculaPi(3) &= 2{,}895238095238095 \\ calculaPi(300) &= 3{,}144914903558851 \end{aligned}$$

Solución:

```
(%i66)  calculaPi(n) := float(4*sum((-1)**x/(2*x+1),x,0,n))$
(%i67)  calculaPi(3);
```

```
(%o67)
                              2,895238095238095

(%i68)  calculaPi(300);
(%o68)
                              3,144914903558851
```

8.2.5. Número de ceros del factorial de n

8.2.5.1 Ejercicio. Definir la función *numeroCerosFactorial* tal que $numeroCerosFactorial(n)$ es el número de ceros con los que termina el factorial de n. Por ejemplo,

$$numeroCerosFactorial(17) = 3$$

Solución: En primer lugar se define la función *numeroCeros* tal que $numeroCeros(n)$ es el número de ceros en los que termina n.

```
(%i69)  numeroCeros(n) :=
          if is(mod(n,10)=0) then 1+numeroCeros(quotient(n,10))
          else 0$
(%i70)  numeroCeros(35400);
(%o70)
                                    2
```

La definición de *numerosCerosFactorial* es

```
(%i71)  numerosCerosFactorial(n) := numeroCeros(n!)$
(%i72)  numerosCerosFactorial(17);
(%o72)
                                    3
```

8.3. Más ejercicios

8.3.1. Números felices

Este ejercicio se basa en el artículo "Happy Numbers" publicado el 23 de julio de 2010 en ProgrammingPraxis y su respuesta en Bonsai.

Según la Wikipedia, un número feliz se define por el siguiente proceso. Se comienza reemplazando el número por la suma del cuadrado de sus cifras y se repite el proceso hasta que se obtiene el número 1 o se entra en un ciclo que no contiene al 1. Aquellos números para los que el proceso termina en 1 se llaman números felices y los que entran en un ciclo sin 1 se llaman números desgraciados.

Por ejemplo, 7 es un número feliz porque

$$\begin{array}{rlll} 7 & \to 7^2 & & = 49 \\ & \to 4^2+9^2 & = 16+81 & = 97 \\ & \to 9^2+7^2 & = 81+49 & = 130 \\ & \to 1^2+3^2+0^2 & = 1+9+0 & = 10 \\ & \to 1^2+0^2 & = 1+0 & = 1 \end{array}$$

Pero 17 es un número desgraciado porque

$$\begin{array}{rlll} 17 & \to 1^2+7^2 & = 1+49 & = 50 \\ & \to 5^2+0^2 & = 25+0 & = 25 \\ & \to 2^2+5^2 & = 4+25 & = 29 \\ & \to 2^2+9^2 & = 4+81 & = 85 \\ & \to 8^2+5^2 & = 64+25 & = 89 \\ & \to 8^2+9^2 & = 64+81 & = 145 \\ & \to 1^2+4^2+5^2 & = 1+16+25 & = 42 \\ & \to 4^2+2^2 & = 16+4 & = 20 \\ & \to 2^2+0^2 & = 4+0 & = 4 \\ & \to 4^2 & & = 16 \\ & \to 1^2+6^2 & = 1+36 & = 37 \\ & \to 3^2+7^2 & = 9+49 & = 58 \\ & \to 5^2+8^2 & = 25+64 & = 89 \end{array}$$

que forma un bucle al repetirse el 89.

El objetivo del ejercicio es definir una función que calcule todos los números felices hasta un límite dado.

8.3.1.1 Ejercicio. Definir la función $cifras$ tal que $cifras(n)$ es la lista de las cifras de n. Por ejemplo,

$$cifras(325) = [3,2,5]$$

Solución:

Maxima

```
cifras(n) :=
   cifrasAux(n,[])$

cifrasAux(n,xs) :=
   if n<10 then cons(n,xs)
   else cifrasAux(quotient(n,10),cons(mod(n,10),xs))$
```

Un ejemplo del cálculo con la traza es

```
(%i26) trace(all);
(%o26) [cifras, cifrasAux]
(%i27) cifras(325);
1 Enter cifras [325]
 1 Enter cifrasAux [325, []]
  2 Enter cifrasAux [32, [5]]
   3 Enter cifrasAux [3, [2, 5]]
   3 Exit  cifrasAux [3, 2, 5]
  2 Exit  cifrasAux [3, 2, 5]
 1 Exit  cifrasAux [3, 2, 5]
1 Exit  cifras [3, 2, 5]
(%o27) [3, 2, 5]
(%i28) untrace;
(%o28) untrace
```

8.3.1.2 Ejercicio. Definir la función *caminoALaFelicidad* tal que *caminoALaFelicidad*(n) es la lista de los números obtenidos en el proceso de la determinación si n es un número feliz: se comienza con la lista $[n]$, ampliando la lista con la suma del cuadrado de las cifras de su primer elemento y se repite el proceso hasta que se obtiene el número 1 o se entra en un ciclo que no contiene al 1. Por ejemplo,

```
(%i1) caminoALaFelicidad(7);
(%o1) [1, 10, 130, 97, 49, 7]
(%i2) caminoALaFelicidad(17);
(%o2) [89, 58, 37, 16, 4, 20, 42, 145, 89, 85, 29, 25, 50, 17]
```

Solución:

Maxima

```
caminoALaFelicidad(n) := caminoALaFelicidadAux([n])$

caminoALaFelicidadAux(vs) := block([x:first(vs), xs:rest(vs)],
   if is(x=1) then vs
   elseif member(x,xs) then vs
   else caminoALaFelicidadAux(cons(lsum(y^2,y,cifras(x)),vs)))$
```

8.3.1.3 Ejercicio. Definir la función *esFeliz* tal que *esFeliz*(n) se verifica si n es un número feliz. Por ejemplo,

```
esFeliz(7)   =  true
esFeliz(17)  =  false
```

Solución:

Maxima

```
esFeliz(n) := is(first(caminoALaFelicidad(n)) = 1)$
```

8.3.1.4 Ejercicio. Definir la función *numerosFelices* tal que *numerosFelices*(n) es la lista de los números felices menores que n. Por ejemplo,

```
numerosFelices(50)  =  [1,7,10,13,19,23,28,31,32,44,49]
```

Solución:

Maxima

```
numerosFelices(n) :=
   reverse(numerosFelicesAux(n-1))$

numerosFelicesAux(n) :=
   if is(n=0) then []
   elseif esFeliz(n) then cons(n,numerosFelicesAux(n-1))
   else numerosFelicesAux(n-1)$
```

Puede definirse por recursión final como sigue

Maxima

```
numerosFelices2(n) :=
   numerosFelices2Aux(n-1,[])$

numerosFelices2Aux(n,xs) :=
   if is(n=0) then xs
   elseif esFeliz(n) then numerosFelices2Aux(n-1,cons(n,xs))
   else numerosFelices2Aux(n-1,xs)$
```

También puede definirse por iteración como sigue

Maxima

```
numerosFelices3(n) := block([xs:[]],
   for x:n-1 step -1 thru 1 do
     (if esFeliz(x) then xs:cons(x,xs)),
   xs)$
```

8.3.1.5 Ejercicio. Compilar todas las funciones anteriores.

Solución:

```
(%i10)  compile(all)$
   Compiling /tmp/gazonk_6160_0.lsp.
   End of Pass 1.
   End of Pass 2.
   OPTIMIZE levels: Safety=2, Space=3, Speed=3
   Finished compiling /tmp/gazonk_6160_0.lsp.
   Compiling /tmp/gazonk_6160_0.lsp.
   End of Pass 1.
   End of Pass 2.
```

8.3.1.6 Ejercicio. Calcular el tiempo necesario para calcular la cantidad de números felices hasta 500, 1000 y 2000 con cada una de las 3 definiciones de *numerosFelices*.

Solución:

```
(%i1) showtime:true$
Evaluation took 0.0000 seconds (0.0000 elapsed)
(%i2) length(numerosFelices(500));
Evaluation took 0.4800 seconds (0.5100 elapsed)
(%o2) 76
(%i3) length(numerosFelices2(500));
Evaluation took 0.4400 seconds (0.4500 elapsed)
(%o3) 76
(%i4) length(numerosFelices3(500));
Evaluation took 0.4900 seconds (0.5000 elapsed)
(%o4) 76
(%i5) length(numerosFelices(1000));
Evaluation took 0.9100 seconds (0.9000 elapsed)
(%o6) 142
(%i6) length(numerosFelices2(1000));
Error in PROGN [or a callee]: Bind stack overflow.
(%i7) length(numerosFelices3(1000));
Evaluation took 1.0100 seconds (1.0200 elapsed)
```

```
(%o7) 142
(%i8) length(numerosFelices(2000));
Error in PROGN [or a callee]: Bind stack overflow.
(%i9) length(numerosFelices3(2000));
Evaluation took 1.9600 seconds (1.9600 elapsed)
(%o9) 298
(%i10) showtime:false$
```

En resumen,

x	Def. 1	Def. 2	Def. 3
500	0.48	0.44	0.49
1000	0.91	error	1.01
2000	error	error	1.96

8.3.1.7 Ejercicio. Definir la función *puntos* tal que $puntos(a)$ es la lista de los pares $[x, y]$ tales que x es un número de la forma $100n$ con n entre 1 y a e y es la cantidad de números felices menores que x. Por ejemplo,

$$puntos(5) = [[100, 19], [200, 32], [300, 44], [400, 66], [500, 76]]$$

Solución:

Maxima

```
puntos(a) :=
   makelist([x,length(numerosFelices3(x))],
            x,makelist(100*n,n,1,a))$
```

8.3.1.8 Ejercicio. Cargar la librería de inferencia estadística.

Solución:

Maxima

```
load("stats")$
```

8.3.1.9 Ejercicio. Definir la función *regresion* tal que regresion(n) que es la recta de regresión para los $puntos(n)$. Por ejemplo,

```
(%i1) regresion(5);
(%o1) 0.148 x + 3.000000000000036
```

Indicación: Usar la función `simple_linear_regression` de la librería *stats*.

Solución:

Maxima

```
regresion(n) :=
   take_inference(model,simple_linear_regression(puntos(n)));
```

8.3.1.10 Ejercicio. Calcular los valores de *puntos*(10) y *regresion*(10).

Solución: El cálculo es

```
(%i1) puntos(10);
(%o1) [[100,19],[200,32],[300,44],[400,66],[500,76],[600,81],
       [700,100],[800,111],[900,124],[1000,142]]
(%i2) regresion(10);
(%o2) 0.13290909090909 x + 6.399999999999992
```

8.3.1.11 Ejercicio. Dibujar en una figura los puntos de *puntos*(10) y la recta *regresion*(10).

Maxima

```
plot2d([[discrete,puntos(10)],regresion(10)],
       [x,100,1000],
       [style,points,lines],
       [gnuplot_term,png],
       [gnuplot_out_file,"NumerosFelices_1.png"])$
```

La gráfica está en la figura 8.2.

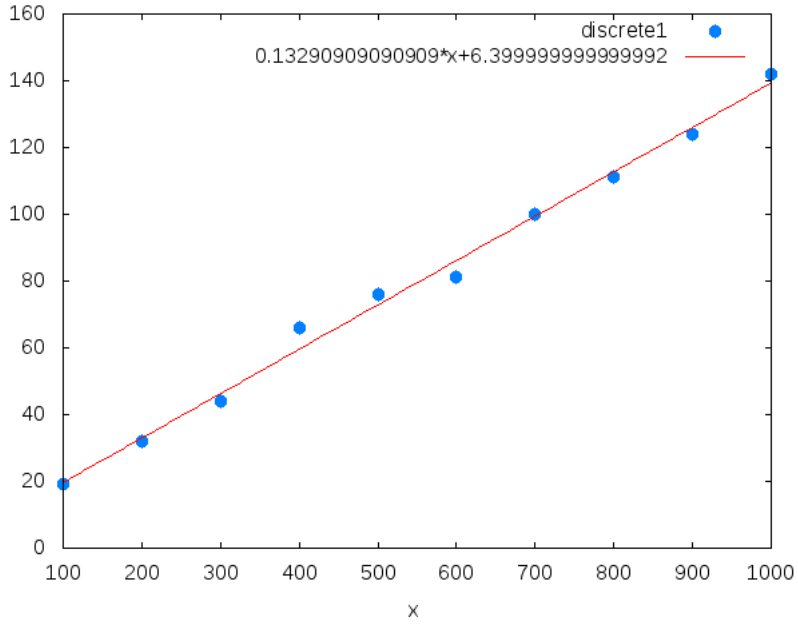

Figura 8.2: Gráfica de números felices

Apéndice A

Resumen de *Maxima*

A.1. Hoja de cálculo

- « `;` » evalúa una expresión y devuelve el resultado. Por ejemplo `1+2/3;`
- « `$` » evalúa una expresión sin devolver el resultado. Por ejemplo `a:2 $`
- « `%` » es el valor del último cálculo efectuado
- `? plot2d` muestra la ayuda sobre la función `plot2d`
- `example(expand)` muestra ejemplos de utilización de la función `expand`
- `kill(all)` reinicia el sistema

A.2. Operadores

- `+ , - , * , /` son las cuatro operaciones usuales
- « `^` » es el operador potencia. Por ejemplo, `x^3` es x^3
- « `#` » es la relación no igual (distinto)
- `= , <, <= , >, >=` son los operadores de comparación
- « `:` » es el operador de asignación. Por ejemplo `a:3` asigna el valor 3 a la variable a.
- « `:=` » es el operador para definir una función.
- « `=` » indica una ecuación en Maxima.
- « `!` » es el factorial de un entero natural. Por ejemplo, $5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$.
- « `.` » es el producto de dos matrices.

A.3. Constantes

- `%pi` es el número $\pi \approx 3,14159$
- `%e` es el número e $= \exp(1) \approx 2,7183$
- `%i` es la unidad imaginaria $\sqrt{-1}$
- `true` es el valor lógico "verdadero"
- `false` es el valor lógico "falso"
- `inf` es $+\infty$
- `minf` es $-\infty$

A.4. Números reales

A.4.1. Funciones usuales

- `abs(x)` valor absoluto de x
- `floor(x)` parte entera de x
- `round(x)` redondeo entero de x
- `sqrt(x)` raiz cuadrada de x
- `exp(x)` es e^x
- `log(x)` logaritmo neperiano de x
- Funciones trigonométricas: `sin(x)`, `cos(x)`, `tan(x)`
- Funciones trigonométricas inversas: `asin(x)`, `acos(x)`, `atan(x)`

A.4.2. Valores aproximados

- `float(x)` es el valor decimal aproximado de x
- `bfloat(x)` es el valor aproximado de x en notación científica
- `fpprec:20` fija la aproximación dada por `bfloat` a 20 cifras (por defecto es 16)

A.4.3. Trigonometría

- `trigexpand(a)` desarrolla la expresión trigonométrica a utilizando las fórmulas de la suma de seno y coseno. Por ejemplo, `trigexpand(cos(x+y))` devuelve $\cos x \cos y - \sin x \sin y$
- `trigreduce(a)` reduce un polinomio trigonométrico a. Por ejemplo, `trigreduce(sin(x)^` devuelve $\frac{3\sin x - \sin(3x)}{4}$
- `trigsimp(a)` simplifica la expresión trigonométrica a usando la relación $\cos^2 t + \sin^2 t = 1$ y sustituyendo $\tan t$ por $\frac{\sin t}{\cos t}$
- `load(ntrig)` carga el paquete, lo que permite obtener los valores exactos de $\sin x$, $\cos x$ y $\tan x$ cuando x es un múltiplo de $\pi/10$

A.5. Aritmética entera

Sean a y b dos enteros. Sean n y p dos números naturales.

- `divide(a,b)` es la división euclídea de a por b. El resultado es una lista cuyo primer elemento es el cociente y el segundo el resto
- `divisors(a)` es el conjunto de los divisores positivos de a
- `divsum(a)` es la suma de los divisores positivos de a
- `quotient(a,b)` es el cociente de la división euclídea de a por b
- `mod(a,b)` es el resto de la división euclídea de a por b
- `gcd(a,b)` es el máximo común divisor de a y b
- `load(functs) $ lcm(a,b)` es el mínimo común múltiplo de a y b
- `primep(p)` se verifica si p es primo
- `prev_prime(n)` es el mayor primo menor que n
- `next_prime(n)` es el menor primo mayor que n
- `factor(n)` es la descomposición de n en producto de factores primos
- `ifactors(n)` es la lista de la descomposición de n en producto de factores primos
- `binomial(n,p)` es el coeficiente binomial $\binom{n}{p}$
- `random(n)` es un entero natural, elegido al azar entre 0 y $n-1$ cuando $n \in \mathbb{N}^*$

A.6. Números complejos

Sea z un número complejo.

- `realport(z)` es la parte real de z
- `imagport(z)` es la parte imaginaria de z
- `conjugate(z)` es el conjugado de z
- `abs(z)` es el módulo de z
- `carg(z)` es el argumento de z (en $]-\pi, \pi]$)
- `rectform(z)` es la forma cartesiana de z
- `polarform(z)` es la forma polar de z

A.7. Cálculo algebraico

Sean P y Q dos polinomios.

- `R: x^4-4*x^2-5` define el polinomio $R = X^4 - 4X^2 - 5$
- `expand(P)` desarrolla P
- `factor(P)` factoriza P
- `gfactor(P)` factoriza P en $\mathbb{C}$
- `solve(P,x)` raices complejas de P

- `subst(4,x,P)` valor de P cuando $X = 4$
- `ratcoef(P,x^3)` es el coeficiente del término en X^3 de P
- `divide(P,Q,x)` es la lista formada por el cociente y el resto de la división de P por Q.
- `portfrac(P/Q,x)` descompone la función racional P/Q en elementos simples
- `ratsimp(expr)` simplifica la expresión racional `expr` (reduce a común denominador)
- `subst(1/z,x,expr)` sustituye x por $1/z$ en la expresión `expr`

A.8. Funciones numéricas

A.8.1. Definición de funciones

- `f(x):=x^2+2*x-3`
- `define(f(x),x^2+2*x-3)`
- `f:lambda([x],x^2+2*x-3)`

A.8.2. Límites, tangentes y asíntotas

- `limit(sin(x)/x,x,0)` límite en 0
- `limit(1/x,x,0,plus)` límite en 0 por la derecha
- `limit(1/x,x,0,minus)` límite en 0 por la izquierda
- `limit(1/x,x,inf)` límite en $+\infty$
- `limit(x*exp(x),x,minf)` límite en $-\infty$
- `taylor(f(x),x,a,n)` desarrollo de Taylor de f respecto de x en el punto a con n términos

A.8.3. Derivación

- `diff(f(x),x)` es la derivada $f'(x)$
- `diff(f(x),x,2)` es la derivada segunda $f''(x)$
- `define(g(x),diff(f(x),x))` define $g(x)$ como $f'(x)$
- `taylor(f(x),x,a,1)` es la ecuación reducida de la tangente a f en el punto $(a, f(a))$

A.8.4. Representación de funciones

- `plot2d([f(x),g(x)],[x,x1,x2],[y,y1,y2])` representa las funciones f y g en la región $[x_1, x_2] \times [y_1, y_2]$

A.8.5. Integrales

- `integrate(f(x),x)` es una primitiva de la función f
- `integrate(f(x),x,a,b)` es la integral $\int_a^b f(x)dx$
- `romberg(f(x),x,a,b)` es una aproximación de la integral $\int_b^a f(x)dx$

A.9. Ecuaciones

A.9.1. Resolución de ecuaciones

- `solve(x^3-2x^2+x=2,x)` es la resolución exacta en $\mathbb{C}$
- `find_root(x^5=1+x,x,1,2)` solución aproximada en $[1,2]$
- `allroots(x^5=1+x)` valores aproximados de todas las soluciones (reales y complejas)

A.9.2. Sistemas lineales

Para resolver el sistema $\begin{cases} 3x+2y=1 \\ x-y=2 \end{cases}$

- `S1:[3*x+2*y=1,x-y=2]`
- `solve(S1,[x,y])`

A.10. Listas

- `cons(x,l)` es la lista obtenida añadiendo el elemento `x` a la lista `l`.
- `L:makelist(k^2,k,0,9)` asigna a L la lista con los cuadrados de los 10 primeros naturales
- `L[2]:5` sustituye el segundo elemento de la lista L por 5
- `length(L)` longitud de la lista L.
- `first(L)` primer elemento de L
- `second(L)` segundo elemento de L
- `last(L)` último elemento de L
- `member(x,L)` se verifica si x pertenece a la lista L
- `append(xs,ys)` concatena las listas
- `join(l,m)` construye una lista, intercalando los elementos de l y m. Se obtiene la lista $[l[1], m[1], l[2], m[2], l[3], m[3], \ldots]$.
- `sort(L)` ordenación de los elementos de L en orden creciente.
- `map(f,L)` aplicación de la función f a todos los elementos de L
- `create_list(e,x1,l1,...,xn,ln)` crea la lista evaluando la expresión `e` con sus argumentos en las listas `l1,...,ln`.
- `delete(x,ys,n)` elimina las `n` primeras ocurrencias de `x` en `ys`.

A.11. Sumas y productos

A.11.1. Sumas finitas

- `sum(1/k^2,k,1,10)` es la suma de los inversos de los cuadrados de los enteros entre 1 y 10.

A.11.2. Productos finitos

- `product(sqrt(k),k,1,10)` es el producto de las raices cuadradas de los enteros entre 1 y 10.

A.11.3. Sumas infinitas

- `load(simplify_sum) $ sum(1/k^2,k,1,inf) $ simplify_sum(%)` calcula la suma de la serie $\sum_{k=1}^{n} \frac{1}{k^2}$

A.12. Programación

A.12.1. Sintaxis de un programa

- Sintaxis general

```
nombre(parámetros de entrada) := block([variables locales],
   <instrucción 1>,
   ....
   <instrucción n>,
   /* -------Comentario------ */)
```

- Ejemplo simple de programa que suma dos números
`suma(a,b) := block([c], c:a+b, return(c))`

A.12.2. Estructura condicional

- Sintaxis

```
if (condición)
   then (<instrucción1> , <instrucción2>)
   else (<instrucción3> , <instrucción4>)
```

A.12.3. Estructuras iterativas

- Bucle **For** para escribir la tabla del 7:
  `for k from 1 thru 10 do (print("7 veces",k,ïgual a",7*k))`
- Bucle **While** para escribir la tabla del 7:
  `k:1 $ while k<11 do (print("7 veces",k,ïgual a",7*k), k:k+1)`

A.13. Matrices

A.13.1. Construcción de matrices

- Por extensión:
  `A:matrix([1,2,3],[4,5,6],[7,8,-9])`
  define la matriz $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & -9 \end{pmatrix}$
- Mediante una función:
  `M:genmatrix(lambda([i,j],i+j),3,2)`
  define la matriz $M \in \mathfrak{M}_{3,2}(\mathbb{R})$ cuyos elementos son de la forma $m_{i,j} = i + j$.

A.13.2. Matrices particulares

- `zeromatrix(5,3)` es la matriz nula de orden 5×3
- `ident(5)` es la matriz identidad I_5
- `diag_matrix(a,b,c)` es la matriz diagonal $\begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix}$

A.13.3. Operaciones con matrices

- `A+B` es la suma de las matrices A y B
- `3*A` es el producto de la matriz A por 3
- `A.B` es el producto de las matrices A y B
- `A^^3` es la matriz A elevada al cubo
- `invert(A)` es la inversa de la matriz A
- `transpose(A)` es la traspuesta de la matriz A
- `determinant(A)` es el determinante de A