Ecuaciones diferenciales con Maxima

José Antonio Vallejo

Facultad de Ciencias
UASLP

Índice

Prefacio

Antes de nada, esto no es un curso habitual de ecuaciones diferenciales: no se parte de cero, no hay demostraciones, etc. Estas notas están pensadas como un complemento a un curso de carácter teórico y tienen como objetivo ocuparse *únicamente* de los aspectos computacionales, es decir, cómo calcular la solución a una ecuación diferencial de manera efectiva y eficiente.
Sin embargo, aquí no hablaremos del cálculo de subespacios cíclicos, factores invariantes, formas de Jordan, sistemas fundamentales de soluciones, etc. Todas esas construcciones van implementadas en los algoritmos de cálculo que el software simbólico moderno (Maxima, Mathematica, Maple, etc.) ya trae "out of the box". La idea es no perderse en medio de un algoritmo tedioso si eso lo puede hacer una máquina más rápido y mejor. Se trata entonces de aprender a usar el software para resolver en la práctica los problemas que se nos planteen, dejándonos tiempo libre para ocuparlo pensando en cuestiones más interesantes desde un punto de vista matemático.

El software con que trabajaremos es Maxima (con su interfaz wxMaxima). Maxima es un sistema algebraico computacional (CAS por sus siglas en inglés), es decir, un paquete de programas que permiten hacer cálculos numéricos y simbólicos. Por lo que respecta a la posibilidad de hacer cálculos numéricos, Maxima se comporta como una calculadora avanzada de gran precisión, pero lo realmente importante es que ofrece la posibilidad (ausente en las calculadoras científicas) de hacer cálculos literales, simbólicos, con variables que no tienen ningún valor asignado. Así, por ejemplo, si le pedimos a Maxima que calcule la derivada de la función $\sin(x)$, nos devuelve $\cos(x)$, no calcula la derivada numérica en un punto, sino que lo hace para cualquier valor que tome la variable x (como haría mentalmente un matemático).
Naturalmente, para poder conseguir esto es necesario que la interacción con Maxima se haga por medio de reglas muy precisas. Para nosotros puede ser lo mismo $\sin(x)$ que $\mathrm{sen}(x)$, pero para Maxima no lo es. Si queremos que el programa haga cálculos para nosotros, debemos darle la información sobre lo que queremos hacer en su propio lenguaje y esta es la parte más complicada, pues requiere de un aprendizaje previo por parte del usuario de la sintaxis que utiliza Maxima. En la primera sección, repasaremos brevemente los comandos básicos de Maxima, los comandos más avanzados se analizarán a medida que vayan apareciendo. Hay muchos manuales y tutoriales en la red (algunos de ellos, excelentes, en español), por lo que se anima al lector a consultarlos para cualquier aspecto no tratado aquí. Asi mismo, el programa Maxima tiene un sistema de ayuda muy completo, al que se accede en al interfaz gráfica mediante el menú "Help" (o "Ayuda" si se ha configurado la interfaz en español).

Respecto a las notaciones existentes para las ecuaciones diferenciales, hay muchas y utilizaremos cualquiera de ellas indistintamente, algo a lo que debe acostumbrarse el lector. Las más comunes consisten en indicar una derivada por medio del apóstrofe ', por ejemplo, si $y = y(x)$ es una función de una variable

que denominaremos x, su derivada respecto de esa variable es

$$y'(x) = \frac{dy}{dx}.$$

Esta notación es apropiada hasta las terceras derivadas: se usa y, y', y'' e incluso y''', pero para las derivadas de orden superior esto se hace incómodo y se suele indicar el orden de derivación mediante un símbolo como (n). Así:

$$y^{(n)}(x) = \frac{d^n y}{dx^n}.$$

En Física, cuando la variable independiente se interpreta como el tiempo t, es frecuente utilizar un punto $\cdot$ en lugar de '. Pondremos entonces

$$\dot{y}(t) = \frac{dy}{dt}, \quad \ddot{y}(t) = \frac{d^2y}{dt^2}$$

(en Física no es frecuente utilizar derivadas de orden mayor que 2, excepto en algunos campos muy concretos).

Quisiera mostrar aquí mi agradecimiento y respeto por el magnífico trabajo que los desarrolladores de Maxima llevan a cabo, así como la ayuda (desinteresada y muy eficiente) que prestan en la lista de Maxima. Mil gracias a: Alexey Beshenov, David Billinghurst, Robert Dodier, Richard Fateman, Richard Hennessy, Dieter Kaiser, Stavros Macrakis, Mario Rodríguez, Viktor Toth, Raymond Toy, Jaime Villate, Andrej Vodopivec y Barton Willis.

1. Introducción a Maxima

1.1. Aritmética básica

Maxima puede operar como una calculadora científica avanzada. Las operaciones básicas como sumas, productos, potencias o cocientes se indican con los símbolos habituales (y los operadores también siguen las reglas de precedencia habituales):

```
(%i1) 4*(1+3)^2/5;
```

$$(\%o1) \qquad \frac{64}{5}$$

Maxima numera cada entrada y salida con unos contadores `%in` y `%on`.
Las constantes numéricas más conocidas se indican con el símbolo % delante. Así, `%i` es la unidad imaginaria, la base de logaritmos neperianos es `%e`, `%pi` es el número pi. También disponemos de las funciones trigonométricas comunes:

```
(%i2) sin(%pi/2);
```

$$(\%o2) \qquad 1$$

y de sus inversas:

```
(%i3) atan(1);
```

$$(\%o3) \qquad \frac{\pi}{4}$$

Maxima trabaja en forma simbólica siempre que aparecen números enteros. Por ejemplo, la siguiente expresión la trata tal cual, sin modificarla:

```
(%i4) 2/sqrt(2);
```

$$(\%o4) \qquad \frac{2}{\sqrt{2}}$$

Para conocer su valor numérico usamos la orden `float`. Aprovechamos para ver cómo hacer referencia a un comando o resultado anterior, mediante los contadores `%in` o `%on`. El contador % hace referncia a la salida del último comando escrito (de manera que abajo daría lo mismo si escribiéramos `float(%)`:

```
(%i5) float(%o4);
```

(%o5) 1.414213562373095

En el caso de que necesitemos ese valor con más digitales, podemos modificar el valor de la variable global `fpprec` (que por defecto es 15). Las variables se asignan (o modifican) escribiendo su nombre seguido de dos puntos y el valor que se les da:

```
(%i6) fpprec:25;
```

(%o6) 25

Para recuperar el valor numérico de $\frac{2}{\sqrt{2}} = \sqrt{2}$, ahora debemos recurrir al comando `bfloat` (de "big float"):

```
(%i7) bfloat(sqrt(2));
```

(%o7) 1.414213562373095048801689b0

Este comando expresa su salida acabando en bN, lo cual quiere decir que la expresión resultante lleva un factor 10^N. En el siguiente ejemplo se ve más claro: la presencia de $b3$ al final indica que hay que multiplicar por 10^3:

```
(%i8) 1000+bfloat(sqrt(2));
```

(%o8) 1.001414213562373095048802b3

1.2. Números complejos

Maxima trabaja sin problemas con números complejos. Podemos definir dos variables que representen los números $3+2i$ y $-1-i$ (obsérvese que escribimos un asterisco * para indicar el producto por i):

```
(%i9) z:3+2*%i;
```

(%o9) $2\,i+3$

```
(%i10) w:-1-%i;
```

(%o10) $-i-1$

Ahora, podemos hacer las operaciones que queramos con ellos:

```
(%i11) z+w;
```

(%o11) $i+2$

```
(%i12) realpart(w);
```

(%o12) -1

```
(%i13) imagpart(z);
```

(%o13) 2

Se pueden definir variables con subíndices, como en este caso que construímos z_0:

```
(%i14) z[0]:z*w;
```

(%o14) $(-i-1)\,(2\,i+3)$

Fijémonos en que Maxima trata las variables de forma simbólica, no las modifica ni las convierte a valores numéricos. A veces esto no es el comportamiento que nos gustaría:

```
(%i15) exp(%);
```

(%o15) $e^{(-i-1)\,(2\,i+3)}$

En este caso, podemos expandir las operaciones primero y definir una variable después:

```
(%i16) z*w;
```

(%o16) $(-i-1)\,(2\,i+3)$

```
(%i17) expand(%);
```

(%o17) $-5\,i-1$

```
(%i18) z[1]:%;
```

(%o18) $-5\,i-1$

```
(%i19) exp(%);
```

(%o19) $e^{-5\,i-1}$

Podemos expresar este mismo complejo mediante la fórmula de DeMoivre:

```
(%i20) demoivre(%);
```

(%o20) $e^{-1}\,\left(cos\left(5\right)-i\,sin\left(5\right)\right),$

en forma rectangular (cartesiana):

```
(%i21) rectform(2*exp(z));
```

(%o21) $2\,e^{3}\,i\,sin\left(2\right)+2\,e^{3}\,cos\left(2\right)$

o en forma polar:

```
(%i22) polarform(w);
```

(%o22) $\sqrt{2}\,e^{-\frac{3\,i\,\pi}{4}}$

El apóstrofe ' delante de algún comando (o variable) sirve para indicarle al programa que no evalúe lo que le sigue. Esto es útil, por ejemplo, para escribir ecuaciones:

```
(%i23) 'z[0]=z[0];
```

(%o23) $z_0=\left(-i-1\right)\,\left(2\,i+3\right)$

```
(%i24) 'z[1]=z[1];
```

(%o24) $z_1=-5\,i-1$

1.3. Simplificación de expresiones

Veamos ahora cómo simplificar expresiones:

```
(%i25) 2/a+4/sqrt(b);
```

(%o25) $$\frac{4}{\sqrt{b}}+\frac{2}{a}$$

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

(%o26) $$\frac{2\sqrt{b}+4\,a}{a\sqrt{b}}$$

`ratsimp` (rational simplification) es el comando básico (junto con `simplify`, pero éste último no trabaja con expresiones racionales que contienen variables), pero a veces se necesita algo más potente:

```
(%i27) a/b+sqrt(a/b);
```

(%o27) $$\frac{a}{b}+\sqrt{\frac{a}{b}}$$

```
(%i28) ratsimp(%);
```

(%o28) $$\frac{\sqrt{\frac{a}{b}}\,b+a}{b}$$

Para estos casos tenemos `radcan`, que es capaz de simplificar radicales, potencias y logaritmos:

```
(%i29) radcan(%);
```

(%o29) $$\frac{\sqrt{a}\,\sqrt{b}+a}{b}$$

```
(%i30) (a^b)^c;
```

(%o30) $$\left(a^{b}\right)^{c}$$

```
(%i31) radcan(%);
```

(%o31) $$a^{b\,c}$$

En el siguiente ejemplo, calculamos la descomposición de 12! en factores primos:

```
(%i32) factor(12!);
```

(%o32) $$2^{10}\,3^5\,5^2\,7\,11$$

Podemos expandir la fórmula del binomio de Newton:

```
(%i33) expand((a+b)^4);
```

(%o33) $$b^4+4\,a\,b^3+6\,a^2\,b^2+4\,a^3\,b+a^4$$

1.4. Ecuaciones y gráficos

Pasemos ahora a la resolución de ecuaciones, un aspecto básico de cualquier programa de cálculo simbólico. El comando `solve` trabaja con ecuaciones polinomiales, tanto explícitas como con parámetros. Las soluciones se devuelven en forma de una lista (una expresión cuyos elementos van entre corchetes [ ]):

```
(%i34) solve(a*x^2+b*x+c=0,x);
```

(%o34) $$[x=-\frac{\sqrt{b^2-4\,a\,c}+b}{2\,a},x=\frac{\sqrt{b^2-4\,a\,c}-b}{2\,a}]$$

El comando también resuelve sistemas de ecuaciones algebraicas. Las ecuaciones y las variables se pasan como dos listas independientes:

```
(%i35) solve([2*x+3*y=2,-x+5*y=-1],[x,y]);
```

(%o35) $$[[x=1,y=0]]$$

También se pueden resolver sistemas con parámetros (aquí, a es un parámetro y las soluciones se expresan en función de él):

```
(%i36) solve([a*x+3*y=2,-x+5*y=-1],[x,y]);
```

(%o36) $$[[x=\frac{13}{5\,a+3},y=-\frac{a-2}{5\,a+3}]]$$

Para calcular raíces de polinomios, tenemos `allroots`:

```
(%i37) allroots(x^3-5*x^2+11*x-15);
```

(%o37) $[x = 2.0\,i + .9999999999999999, x = .9999999999999999 - 2.0\,i, x = 3.0]$

O bien, si sólo nos interesan las raíces reales:

```
(%i38) realroots(x^3-5*x^2+11*x-15);
```

(%o38) $[x = 3]$

`solve` también es capaz de resolver algunas ecuaciones no lineales:

```
(%i39) solve(asin(cos(3*x))*(f(x)-1)=0,x);
'solve' is using arc-trig functions to get a solution.
Some solutions will be lost.
```

(%o39) $[x = \frac{\pi}{6}, f(x) = 1]$

Fijémonos en que Maxima avisa del uso de funciones trigonmétricas inversas, que son multivaluadas, por lo que puede que no proporcione *todas* las soluciones. La ecuación a resolver se pasa en la forma "expresión=0". Si no se especifica una ecuación, omitiendo el sigo =, Maxima entiende que la expresión esta igualada a 0:

```
(%i40) solve(asin(cos(3*x))*(f(x)-1),x);
'solve' is using arc-trig functions to get a solution.
Some solutions will be lost.
```

(%o40) $[x = \frac{\pi}{6}, f(x) = 1]$

Para resolver ecuaciones no lineales más complicadas hay que recurrir a algún método numérico. Estos métodos no están implementados en las rutinas básicas de Maxima (para evitar un uso desmesurado de la memoria del sistema). Hay que cargarlos con la orden `load`. Por ejemplo, para resolver ecuaciones no lineales mediante en método de Newton, cargamos el paquete `newton1`:

```
(%i41) load(newton1)$
```

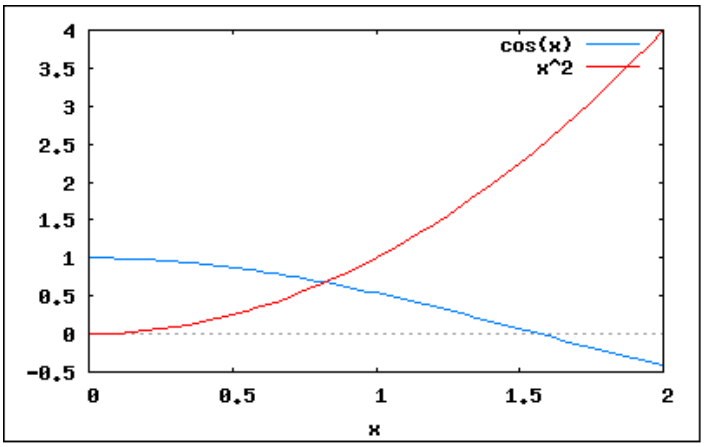
Ahora, planteamos la solución a la ecuación $\cos(x) = x^2$. Hay que dar la ecuación a resolver en la forma "expresión=0" ($\cos(x) - x^2 = 0$), indicar la variable (x), un valor inicial a partir del cual iterar el método (aquí $x_0 = 0.5$) y el valor de parada, en este caso 0.001:

```
(%i42) newton(cos(x)-x^2,x,0.5,0.001);
```

(%o42) .8241461317281952

Podemos comprobar que la solución es correcta graficando las funciones para ver dónde se cortan. El comando *básico* para graficar es `wxplot2d` (hay otros más avanzados como `wxdraw2d`). Para graficar las funciones, las pasamos en una lista. En otra lista se da el valor de la variable y sus valores mínimo y máximo (en este orden):

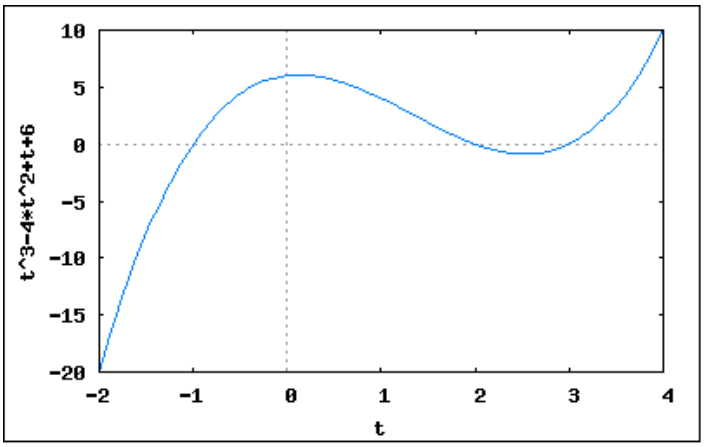
```
(%i43) wxplot2d([cos(x),x^2],[x,0,2]);
```



(%o43)

Si sólo se va a representar una función, no hace falta ponerla en una lista:

```
(%i44) wxplot2d(t^3-4*t^2+t+6,[t,-2,4]);
```



(%o44)

El paquete `mnewton` resuelve sistemas de funciones no lineales. Las ecuaciones se pasan en una lista, luego va otra lista con las variables (en este caso x e y) y una tercera lista con unos valores iniciales para las variables que estimemos cerca de los valores de las soluciones (para obtener estos valores iniciales puede ser útil representar primero las gráficas de las ecuaciones):

```
(%i45) load(mnewton)$
(%i46) mnewton([x+3*log(x)-y^2, 2*x^2-x*y-5*x+1],[x,y],[5,5]);
0 errors, 0 warnings
```

(%o46) $[[x = 3.756834008012769, y = 2.779849592817897]]$

1.5. Límites, derivadas e integrales

El cálculo de límites no supone ninguna dificultad: el comando `limit` admite una expresión, la variable y el valor al que tiende. Aquí calculamos $\lim_{a\to\pi} 2\cos a$:

```
(%i47) limit(2*cos(a),a,%pi);
```

(%o47) -2

También se peuden calcular límites cuando la variable tiende a $+\infty$ (`inf`) ó $-\infty$ (`minf`).

```
(%i48) limit(4/(1+c^2),c,inf);
```

(%o48) 0

En el caso de que el límite sea indeterminado, Maxima lo indica con `und` (de undetermined):

```
(%i49) limit(1/x,x,0);
```

(%o49) *und*

(en estos casos, se puede estudiar el límite lateral, con `limit(1/x,x,0,minus)` y `limit(1/x,x,0,plus)`).
Las derivadas se calculan con el comando `diff`, indicando la función y la variable:

```
(%i50) diff(cos(x^2),x);
```

(%o50) $$-2\,x\,sin\left(x^2\right)$$

Se pueden calcular derivadas de orden superior sin más que añadir el orden de derivación. Por ejemplo, aquí calculamos la derivada segunda de $\cos(x^2)$:

```
(%i51) diff(cos(x^2),x,2);
```

(%o51) $$-2\,sin\left(x^2\right)-4\,x^2\,cos\left(x^2\right)$$

Recordando que el apóstrofo deja sin evaluar un comando, podemos escribir ecuaciones como

```
(%i52) 'diff(cos(x^2),x,2)=diff(cos(x^2),x,2);
```

(%o52) $$\frac{d^2}{d\,x^2}\,cos\left(x^2\right)=-2\,sin\left(x^2\right)-4\,x^2\,cos\left(x^2\right)$$

El comando para integrar es `integrate`:

```
(%i53) integrate(1/(2+3*x^2),x);
```

(%o53) $$\frac{atan\left(\frac{3\,x}{\sqrt{6}}\right)}{\sqrt{6}}$$

Si queremos calcular integrales definidas, basta con añadir los extremos de integración:

```
(%i54) integrate(exp(x)*cos(x)^2,x,0,%pi);
```

(%o54) $$\frac{3\,e^{\pi}}{5}-\frac{3}{5}$$

```
(%i55) 'integrate(exp(x)*cos(x)^2,x,0,%pi)=
integrate(exp(x)*cos(x)^2,x,0,%pi);
```

(%o55) $$\int_0^{\pi}e^x\,cos\left(x\right)^2dx=\frac{3\,e^{\pi}}{5}-\frac{3}{5}$$

2. Ecuaciones de primer orden

2.1. Ecuaciones separables

El comando básico para resolver ecuaciones diferenciales de primer y segundo orden en Maxima es `ode2`, cuya sintaxis es `ode2(`*eqn,vard,vari*`)` siendo *eqn* la ecuación que nos interesa *resuelta con respecto a la derivada* y *vard, vari* la variable dependiente y la independiente, respectivamente. Su uso es muy sencillo, como se verá en los siguientes ejemplos.

Para determinar las soluciones a la ecuación $xy(1+y^2)\mathrm{d}x-(1+x^2)\mathrm{d}y=0$, como se ha señalado, hay que resolver primero con respecto a la derivada. En este caso, la ecuación equivalente en la que ya aparece explícitamente la derivada es:

$$xy(1+y^2)-(1+x^2)\frac{\mathrm{d}y}{\mathrm{d}x}=0.$$

Resulta cómodo definir primero la ecuación y luego pasársela a Maxima:

```
(%i1) ec1:x*y*(1+y^2)-(1+x^2)*'diff(y,x)=0;
```

$$(\%o1)\qquad x\,y\,\left(y^2+1\right)-\left(x^2+1\right)\,\left(\frac{d}{d\,x}\,y\right)=0$$

```
(%i2) ode2(ec1,y,x);
```

$$(\%o2)\qquad -\frac{log\left(y^2+1\right)-2\,log\left(y\right)}{2}=\frac{log\left(x^2+1\right)}{2}+\%c$$

Nota importante: Fijémonos en que al definir la ecuación se ha utilizado el comando `'diff`, con el apóstrofo ' delante. Recordemos que esto sirve para indicarle a Maxima que no evalúe la derivada, de manera que se tenga efectivamente una ecuación. Si no pusiéramos el apóstrofo, Maxima trataría de derivar y con respecto a x y, al no haber definido previamente que $y=y(x)$ (mediante el comando `depends`, por ejemplo), esa derivada daría 0 y no habría nada que resolver.

El objetivo final es poder calcular la función solución para un valor inicial determinado, lo cual determina automáticamente la constante $\%c$. En Maxima podemos indicar el valor inicial de un problema de primer orden mediante el comando `ic1`, con la sintaxis `ic1(`*eqn,*$x=x_0, y=y_0$`)`, siendo *eqn* la ecuación proporcionada por `ode2`, x_0 el punto inicial e y_0 el valor de la función variable dependiente en el punto inicial. Por ejemplo, supongamos que en el caso que estamos tratando queremos hallar la solución $y=y(x)$ tal que $y(-1)=1$. Basta con hacer:

```
(%i3) ic1(%o2,x=-1,y=1);
```

(%o3) $$-\frac{\log(y^2+1)-2\log(y)}{2}=\frac{\log(x^2+1)-2\log(2)}{2}$$

Ahora, veamos cómo simplificar esta ecuación. Lo más sencillo es utilizar la función `logcontract`, que simplifica expresiones como $\log(A)+\log(B)$ para dar $\log(AB)$ y similares (hay un modificador que hace lo contrario: `logexpand` tiene por defecto el valor `false`, pero si se declara `logexpand:true`, cuando se detecten expresiones como $\log(a^b)$, automáticamente se convierten a $a\log(b)$, $\log(ab)$ pasa a ser $\log(a)+\log(b)$, etc.) Podemos hacer que, además, se simplifiquen factores racionales (convirtiendo por ejemplo $\frac{1}{2}\log(a)$ en $\log(\sqrt{a})$) con los siguientes comandos:

```
(%i4) logconcoeffp:'logconfun$
(%i5) logconfun(m):=featurep(m,integer) or ratnump(m)$

(%i6) logcontract(%o3);
```

(%o6) $$\log(\frac{|y|}{\sqrt{y^2+1}})=\log(\frac{\sqrt{x^2+1}}{2})$$

Aunque cuando se resuelve una ecuación como esta "a mano" es común simplificar algo más, por ejemplo escribiendo la constante c como $c=\log(k)$ y simplificando los logaritmos de manera que resulte una expresión visualmente más agradable:

$$(1+x^2)(1+y^2)=ky^2,$$

esto no es imprescindible. Dado que el logaritmo es una función monótona creciente, es inyectiva. Lo mismo ocurre con la función elevar al cuadrado; por tanto, esta ecuación equivale a

$$\frac{y^2}{y^2+1}=\frac{x^2+1}{4}.$$

Finalmente, le pedimos a Maxima que nos la resuelva para y:

```
(%i7) solve(y^2*4=(x^2+1)*(y^2+1),y);
```

(%o7) $$[y=-\frac{\sqrt{x^2+1}}{\sqrt{3-x^2}},y=\frac{\sqrt{x^2+1}}{\sqrt{3-x^2}}]$$

Una característica muy interesante de Maxima es que almacena en una variable llamada `method` el método que ha empleado para resolver la última ecuación diferencial. En este caso:

```
(%i8) method;
```

(%o8) *separable*

lo que nos dice que ha utilizado el método de separación de variables (el lector debería reconocer que la ecuación propuesta, $xy(1+y^2)-(1+x^2)\frac{\mathrm{d}y}{\mathrm{d}x}=0$ es de variables separables).

Ahora que hemos visto con detalle el método general, podemos pasar a calcular las soluciones de algunas ecuaciones con mayor rapidez. Por ejemplo, resolvamos la ecuación de variables separables

$$e^x \sec(y)dx + (1+e^x)\sec(y)\tan(y)dy = 0,$$

con las condiciones iniciales $y_0 = \frac{\pi}{3}$ si $x_0 = 3$:

```
(%i9) ec2:exp(x)*sec(y)+(1+exp(x))*sec(y)*tan(y)*'diff(y,x)=0;
```

(%o9) $$(e^x+1)\ sec\,(y)\ tan\,(y)\left(\frac{d}{d\,x}\,y\right)+e^x\,sec\,(y)=0$$

```
(%i10) ode2(ec2,y,x);
```

(%o10) $$-log\,(sec\,(y))=log\,(e^x+1)+\ \%c$$

```
(%i11) ic1(%o11,x=3,y=%pi/3);
```

(%o11) $$-log\,(sec\,(y))=log\,(e^x+1)-log\left(e^3+1\right)-log\,(2)$$

```
(%i12) logcontract(%);
```

(%o12) $$-log\,(sec\,(y))=log\left(\frac{e^x+1}{2\,e^3+2}\right)$$

```
(%i13) solve(sec(y)=(2*%e^3+2)/(exp(x)+1),y);
'solve' is using arc-trig functions to get a solution.
Some solutions will be lost.
```

(%o13) $$[y=asec\left(\frac{2\,e^3}{e^x+1}+\frac{2}{e^x+1}\right)]$$

Fijémonos en que Maxima efectivamente reconoció que la ecuación era separable:

```
(%i14) method;
```

(%o14) *separable*

2.2. Ecuaciones exactas y homogéneas

Otro tipo frecuente de ecuaciones de primer orden son las exactas. Maxima no tiene mayores dificultades con ellas. Consideremos la ecuación $2xydx-(x^2-y^2)dy=0$:

```
(%i15) ec3:2*x*y-(x^2-y^2)*'diff(y,x)=0;
```

$$(\%o15) \qquad 2\,x\,y-\left(x^2-y^2\right)\,\left(\frac{d}{d\,x}\,y\right)=0$$

```
(%i16) ode2(ec3,y,x);
```

$$(\%o16) \qquad -\frac{y}{y^2+x^2}=\%c$$

```
(%i17) solve(%o17,y);
```

$$(\%o17) \qquad [y=-\frac{\sqrt{1-4\,\%c^2\,x^2}+1}{2\,\%c},y=\frac{\sqrt{1-4\,\%c^2\,x^2}-1}{2\,\%c}]$$

Veamos que la ecuación fue tratada como exacta:

```
(%i18) method;
```

$$(\%o18) \qquad exact$$

Y comprobemos que se cumple la condición de exactitud, esto es, si la ecuación se expresa como $M(x,y)dx+N(x,y)dy=0$ entonces

$$\frac{\partial M}{\partial y}=\frac{\partial N}{\partial x}:$$

```
(%i19) diff(2*x*y,y)-diff(x^2-y^2,x);
```

$$(\%o19) \qquad 0$$

Las ecuaciones homogéneas se resuelven de la misma forma:

```
(%i20) ec4:(a*x-b*y)+(b*x-a*y)*'diff(y,x)=0;
```

$$(\%o20) \qquad (b\,x-a\,y)\,\left(\frac{d}{d\,x}\,y\right)-b\,y+a\,x=0$$

```
(%i21) ode2(ec4,y,x);
```

$$(\%o21) \qquad \%c\,x = e^{-\frac{(b+a)\,log\left(\frac{y+x}{x}\right)+(a-b)\,log\left(\frac{y-x}{x}\right)}{2\,a}}$$

```
(%i22) method;
```

$$(\%o22) \qquad homogeneous$$

Notemos que con un poco más de trabajo, esta solución se puede poner en una forma más sugerente:

```
(%i23) radcan(%o21);
```

$$(\%o23) \qquad \%c\,x = \frac{x\,(y-x)^{\frac{b-a}{2\,a}}}{(y+x)^{\frac{b+a}{2\,a}}}$$

Finalmente, simplificando por x y elevando a la potencia adecuada, si hacemos $k = c^{-2a}$ nos queda:

$$k = (y+x)^{a+b}(y-x)^{a-b}.$$

Finalizamos esta sección con otra ecuación homogénea.

```
(%i24) ec5:3*x^2+2*x*y+4*y^2+(20*x^2+6*x*y+y^2)*'diff(y,x)=0;
```

$$(\%o24) \qquad \left(y^2+6\,x\,y+20\,x^2\right)\left(\frac{d}{d\,x}\,y\right)+4\,y^2+2\,x\,y+3\,x^2=0$$

```
(%i25) ode2(ec5,y,x);
```

$$(\%o25) \qquad \%c\,x = \frac{x\,\left(y+3\,x\right)}{y^2+7\,x\,y+x^2}$$

```
(%i26) method;
```

$$(\%o26) \qquad homogeneous$$

```
(%i27) solve(%c*x*(y^2+7*x*y+x^2)=x*(y+3*x),y);
```

```
(%o27)
```

$$[y = -\frac{\sqrt{45\ \%c^2\,x^2-2\,\%c\,x+1}+7\,\%c\,x-1}{2\,\%c},$$
$$y = \frac{\sqrt{45\ \%c^2\,x^2-2\,\%c\,x+1}-7\,\%c\,x+1}{2\,\%c}]$$

2.3. Factores integrantes

Las ecuaciones del tipo $M(x,y)dx+N(x,y)dy=0$ a veces pueden convetirse en exactas con la ayuda de un factor integrante, una función $\mu(x,y)$ tal que al multiplicar por ella la ecuación, para obtener $\mu M dx+\mu N dy=0$, resulta que

$$\frac{\partial(\mu M)}{\partial y}=\frac{\partial(\mu N)}{\partial x}.$$

Maxima es capaz de encontrar estos factores integrantes, aunque no nos dice explícitamente cuál ha sido la función μ utilizada. Para recuperarla, invocamos a la variable `intfactor`. Veamos un ejemplo: se trata de integrar la ecuación $xdy+ydx=x^2y^2dx$,

```
(%i28) ec6:x*'diff(y,x)-(x^2)*(y^2)-y=0;
```

(%o28) $$x\left(\frac{d}{d\,x}\,y\right)-x^2\,y^2-y=0$$

```
(%i29) ode2(ec6,y,x);
```

(%o29) $$-\frac{x^3\,y+3\,x}{3\,y}=\%c$$

```
(%i30) method;
```

(%o30) $$exact$$

Vemos aquí que Maxima ha reducido la ecuación a una exacta. Para saber qué factor integrante ha empleado, hacemos:

```
(%i31) intfactor;
```

(%o31) $$\frac{1}{y^2}$$

Finalmente, podemos expresar la solución en forma explícita:

```
(%i32) solve((x^3)*y+3*x+3*y*%c=0,y);
```

(%o32) $$[y=-\frac{3\,x}{x^3+3\,\%c}]$$

Un ejemplo más. Vamos a integrar la ecuación

$$\left(\frac{y^2}{2}+2ye^x\right)dx+(y+e^x)dy=0:$$

```
(%i33) ec7:'diff(y,x)=-((y^2)/2+2*y*exp(x))/(y+exp(x));
```

$$(\%o33) \qquad \frac{d}{d\,x}\,y = \frac{-\frac{y^2}{2} - 2\,e^x\,y}{y + e^x}$$

```
(%i34) ode2(ec7,y,x);
```

$$(\%o34) \qquad e^x\,y^2 + 2\,e^{2\,x}\,y = \%c$$

```
(%i35) solve(%,y);
```

$$(\%o35) \qquad [y = -e^{-x}\left(e^{\frac{x}{2}}\sqrt{e^{3\,x} + \%c} + e^{2\,x}\right), y = e^{-x}\left(e^{\frac{x}{2}}\sqrt{e^{3\,x} + \%c} - e^{2\,x}\right)]$$

```
(%i36) method;
```

$$(\%o36) \qquad exact$$

```
(%i37) intfactor;
```

$$(\%o37) \qquad e^x$$

2.4. La ecuación lineal de primer orden. Ecuación de Bernoulli

La forma general de una ecuación lineal de primer orden es

$$\frac{dy}{dx} + yP(x) = Q(x). \tag{1}$$

Cuando $Q(y) = 0$, se dice que es una ecuación lineal homogénea. La teoría de las ecuaciones lineales es bien conocida. De hecho, se tiene:

Teorema (Lagrange)**.** *La solución general de la ecuación lineal de primer orden (1) está dada por la función*

$$y(x) = e^{-\int P(x)dx}\left(c + \int Q(x)e^{P(x)dx}dx\right).$$

Estas ecuaciones son importantísimas y Maxima las resuelve sin problemas. Consideremos por ejemplo la integración de la ecuación $y' + 2xy = x$:

```
(%i38) ec8:'diff(y,x)+2*x*y=x;
```

$$(\%o38) \qquad \frac{d}{d\,x}\,y + 2\,x\,y = x$$

```
(%i39) ode2(ec8,y,x);
```

$$(\%o39) \qquad y = e^{-x^2}\,\left(\frac{e^{x^2}}{2} + \%c\right)$$

```
(%i40) method;
```

$$(\%o40) \qquad linear$$

Otro ejemplo: la ecuación $y' + \sin(x)y = 2xe^{\cos(x)}$,

```
(%i41) ec9:'diff(y,x)+(sin(x))*y=2*exp(cos(x));
```

$$(\%o41) \qquad \frac{d}{d\,x}\,y + sin\,(x)\;y = 2\,e^{cos(x)}$$

```
(%i42) ode2(ec9,y,x);
```

$$(\%o42) \qquad y = (2\,x + \%c)\;e^{cos(x)}$$

```
(%i43) method;
```

$$(\%o43) \qquad linear$$

En este tipo de ecuaciones, no es necesario que originalmente se tenga la y' aislada. Maxima puede despejarla. Como ejemplo, consideremos $x^2y' + xy = x^4 + x^2$:

```
(%i44) ec10:(x^2)*'diff(y,x)+x*y=x^4+x^2;
```

$$(\%o44) \qquad x^2\,\left(\frac{d}{d\,x}\,y\right) + x\,y = x^4 + x^2$$

```
(%i45) ode2(ec10,y,x);
```

(%o45) $$y = \frac{\frac{\left(x^2+1\right)^2}{4} + \%c}{x}$$

```
(%i46) method;
```

(%o46) *linear*

La ecuación de Bernoulli tiene la forma

$$y' + P(x)y = Q(x)y^n,$$

y se reduce a la lineal por medio de un cambio de variable. No es preciso recordarlo, puesto que Maxima ya lo tiene implementado. Consideremos por ejemplo la ecuación $y' - \frac{3}{x}y = y^5$:

```
(%i47) ec11:'diff(y,x)-3*y/x=y^5;
```

(%o47) $$\frac{d}{d\,x}\,y - \frac{3\,y}{x} = y^5$$

```
(%i48) ode2(ec11,y,x);
```

(%o48) $$y = \frac{x^3}{\left(\%c - \frac{4\,x^{13}}{13}\right)^{\frac{1}{4}}}$$

```
(%i49) method;
```

(%o49) *bernoulli*

2.5. Ecuaciones de primer orden y grado superior. Ecuaciones de Clairault, Lagrange y Riccati

Para tratar con estos casos, es conveniente cargar el paquete `contrib_ode`, ya que la rutina `ode2` no es capaz de trabajar con ellos.
Veamos cómo resolver la ecuación

$$x\left(\frac{dy}{dx}\right)^2 - (1+xy)\frac{dy}{dx} + y = 0.$$

Lo primero, como hemos mencionado, es cargar el paquete `contrib_ode`, cuyo uso es idéntico al de `ode2`:

```
(%i50) load(contrib_ode)$;

(%i51) ec12:x*'diff(y,x)^2-(1+x*y)*'diff(y,x)+y=0;
```

$$(\%o51) \qquad x\left(\frac{d}{d\,x}\,y\right)^2-(x\,y+1)\,\left(\frac{d}{d\,x}\,y\right)+y=0$$

```
(%i52) contrib_ode(ec12,y,x);
(%o52)
```

$$x\left(\frac{d}{d\,x}\,y\right)^2-(x\,y+1)\,\left(\frac{d}{d\,x}\,y\right)+y=0$$

```
first order equation not linear in y'
```

$$[y=log\,(x)+\,\%c, y=\,\%c\,e^x]$$

Fijémonos en que Maxima nos avisa de que la ecuación no es lineal en la derivada.
Como antes, podemos saber qué método ha utilizado para hallar la solución:

```
(%i53) method;
```

$$(\%o53) \qquad factor$$

Con este paquete, se pueden resolver ecuaciones de tipos menos comunes, como la ecuación de Clairault $\left(\frac{dy}{dx}\right)^2+x\frac{dy}{dx}-y=0$:

```
(%i54) ec13:'diff(y,x)^2+x*'diff(y,x)-y=0;
```

$$(\%o54) \qquad \left(\frac{d}{d\,x}\,y\right)^2+x\,\left(\frac{d}{d\,x}\,y\right)-y=0$$

```
(%i55) contrib_ode(ec13,y,x);
(%o55)
```

$$\left(\frac{d}{d\,x}\,y\right)^2+x\,\left(\frac{d}{d\,x}\,y\right)-y=0$$

```
first order equation not linear in y'
```

$$[y=\,\%c\,x+\,\%c^2, y=-\frac{x^2}{4}]$$

```
(%i56) method;
```

(%o56) *clairault*

También se pueden resolver ecuaciones del tipo de Lagrange, como

$$\frac{dy}{dx} = (x+y)^2 :$$

```
(%i57) ec14:'diff(y,x)=(x+y)^2;
```

(%o57) $$\frac{d}{d\,x}\,y = (y+x)^2$$

```
(%i58) contrib_ode(ec14,y,x);
```

(%o58)
$[[x = \%c - atan\left(\sqrt{\%t}\right), y = -x-\sqrt{\%t}], [x = atan\left(\sqrt{\%t}\right)+\%c, y = \sqrt{\%t}-x]]$

```
(%i59) method;
```

(%o59) *lagrange*

Finalmente, consideremos una ecuación de Riccati

$$\frac{dy}{dx} - (1-2x)y + y^2 = 2x.$$

Esta ecuación es bastante complicada, de hecho, nótese que en la solución aparece la función error $erf(x) = \int_0^x e^{-t^2} dt$:

```
(%i60) ec15:'diff(y,x)-(1-2*x)*y+y^2=2*x;
```

(%o60) $$\frac{d}{d\,x}\,y + y^2 - (1-2\,x)\;y = 2\,x$$

```
(%i61) contrib_ode(ec15,y,x);
(%o61)
...trying factor method...
solving 3 equations in 2 variables
```

$$[y = \frac{e^{-x}\left(e^{x}\left(\sqrt{\pi}\,\%c\,e^{x^2+\frac{1}{4}}\,erf\left(\frac{2\,x+1}{2}\right)+\sqrt{\pi}\,e^{x^2}\right)+2\,\%c\right)}{\sqrt{\pi}\,\%c\,e^{x^2+\frac{1}{4}}\,erf\left(\frac{2\,x+1}{2}\right)+\sqrt{\pi}\,e^{x^2}}]$$

```
(%i62) method;
```

(%o62) *riccati*

3. Ecuaciones de orden superior y sistemas

3.1. Ecuaciones lineales de segundo orden

La forma general de la ecuación diferencial lineal de segundo orden es

$$\frac{d^2x}{dt^2} + a_1(t)\frac{dx}{dt} + a_2(t)x = b(t),$$

donde a_1, a_2, b son funciones reales (no consideraremos aquí ecuaciones en el dominio complejo). Si $b(t) \equiv 0$, se dice que la ecuación es homogénea.
Las ecuaciones de segundo orden se integran igual que las de primero, mediante `ode2`, la única diferencia es que ahora nos aparecerán dos constantes de integración, que Maxima denotará $\%k1$, $\%k2$. Consideremos por ejemplo la ecuación $x'' + x' - 2x = 0$:

```
(%i63) ec16:'diff(x,t,2)+'diff(x,t)-2*x=0;
```

$$(\%o63) \qquad \frac{d^2}{d\,t^2}\,x + \frac{d}{d\,t}\,x - 2\,x = 0$$

```
(%i64) ode2(ec16,x,t);
```

$$(\%o64) \qquad x = \%k1\,e^{t} + \%k2\,e^{-2\,t}$$

También aquí podemos conocer el método que Maxima ha empleado para hallar la solución:

```
(%i65) method;
```

$$(\%o65) \qquad constcoeff$$

Claramente, ha sido el de coeficientes constantes. Maxima también sabe trabajar con el método de variación de constantes, como se ve en el siguiente ejemplo con la ecuación $x'' - x = t^2e^t$:

```
(%i66) ec17:'diff(x,t,2)-'diff(x,t)=(t^2)*exp(t);
```

$$(\%o66) \qquad \frac{d^2}{d\,t^2}\,x - \frac{d}{d\,t}\,x = t^2\,e^{t}$$

```
(%i67) ode2(ec17,x,t);
```

$$(\%o67) \qquad x = \frac{\left(t^3 - 3\,t^2 + 6\,t - 6\right)\,e^{t}}{3} + \%k1\,e^{t} + \%k2$$

```
(%i68) method;
```

(%o68) $variationofparameters$

Digamos algunas palabras sobre las condiciones iniciales en la solución a una ecuación de segundo orden. Ahora, al existir dos constantes de integración, para especificar una solución concreta hay que dar un punto inicial por el que pase la solución $y(x_0) = y_0$ y el valor de la derivada de la función en ese punto $y'(x_0) = y'_0$. Esto se hace a través del comando `ic2`.
Veamos un ejemplo: supongamos que queremos encontrar la solución a la ecuación $y'' - 6y' + 9y = 0$ que en $x_0 = 0$ vale $y_0 = 1$ y tiene derivada $y'_0 = -1$. Primero resolvemos la ecuación:

```
(%i69) ec18:'diff(y,x,2)-6*'diff(y,x)+9*y=0;
```

(%o69) $$\frac{d^2}{d\,x^2}\,y - 6\,\left(\frac{d}{d\,x}\,y\right) + 9\,y = 0$$

```
(%i70) ode2(ec18,y,x);
```

(%o70) $$y = (\,\%k2\,x + \,\%k1)\;e^{3\,x}$$

Y a continuación imponemos las condiciones iniciales:

```
(%i71) ic2(%o70,x=0,y=1,'diff(y,x)=-1);
```

(%o71) $$y = (1 - 4\,x)\;e^{3\,x}$$

Hay otra manera de especificar una solución concreta (o, si se quiere, de eliminar las constantes k_1 y k_2): dando el valor que ha de tener la función solución en dos puntos distintos. Esto se conoce como "especificar las condiciones de contorno", y en Maxima se puede hacer mediante el comando `bc2`. En el ejemplo anterior, si hubiésemos querido la solución que en $x = 0$ vale $y = 2$ y en $x = 3$ vale $y = 4$, hubiésemos hecho

```
(%i72) bc2(%o70,x=0,y=2,x=3,y=4);
```

(%o72) $$y = \left(2 - \frac{e^{-9}\;\left(2\,e^{9} - 4\right)\;x}{3}\right)\;e^{3\,x}$$

En general, `ode2` sabe trabajar bien con las ecuaciones de segundo orden con *coeficientes constantes*. No es capaz, sin embargo, de tratar con ecuaciones de coeficientes variables; en este caso nos devuelve `false` como respuesta, indicando que no sabe resolver la ecuación:

```
(%i73) ec19:'diff(x,t,2)+(1-t)*'diff(x,t)-t*x=0;
```

$$(\%o73) \qquad \frac{d^2}{dt^2}\,x + (1-t)\left(\frac{d}{dt}\,x\right) - t\,x = 0$$

```
(%i74) ode2(ec19,x,t);
```

$$(\%o74) \qquad false$$

Para el caso de coeficientes variables, el paquete `contrib_ode` dispone del comando `odelin`, que funciona igual que `ode2`, pero incorpora rutinas capaces de tratar con ecuaciones de segundo orden *homogéneas* con coeficientes variables (el caso de ecuaciones no homogéneas no está implementado):

```
(%i75) odelin(ec19,x,t);
(%o75)
...trying factor method...
solving 3 equations in 2 variables
```

$$\{e^{-t}, e^{-t-\frac{1}{2}}\,erf\left(\frac{i\,t}{\sqrt{2}} + \frac{i}{\sqrt{2}}\right)\}$$

Nótese que aquí se ha obtenido una solución compleja espuria. La función real que nos proporciona `odelin`, e^{-t} es la solución correcta, como es inmediato comprobar.

3.2. Sistemas de ecuaciones diferenciales de primer orden

Maxima implementa un algoritmo para resolver sistemas de ecuaciones diferenciales lineales mediante transformadas de Laplace. El comando correspondiente es `desolve`. Este comando presenta algunas diferencias con `ode2`, la más importante es que las funciones que intervienen en las ecuaciones se han de declarar con su dependencia en la variable correspondiente; es decir, no se debe escribir algo como $4y$, sino que debe declararse $4y(x)$ si es que y depende de x. Las ecuaciones del sistema, naturalmente, se pasan en forma de lista y las funciones respecto de las cuales queremos resolver el sistema también. Un ejemplo aclarará todo esto. Supongamos que queremos resolver el sistema (donde la variable independiente es t):

$$\begin{cases} x' = -x - 4y - 6t + 3 \\ y' = x + 3y + 3t \\ z' = -8x + 4y - 2z \end{cases}$$

Utilizando el comando `desolve` haríamos:

```
(%i76) e1:'diff(x(t),t)=-x(t)-4*y(t)-6*t+3;
```

$$(\%o76) \qquad \frac{d}{dt}x(t) = -4\,y(t) - x(t) - 6\,t + 3$$

```
(%i77) e2:'diff(y(t),t)=x(t)+3*y(t)+3*t;
```

$$(\%o77) \qquad \frac{d}{dt}y(t) = 3\,y(t) + x(t) + 3\,t$$

```
(%i78) e3:'diff(z(t),t)=-8*x(t)+4*y(t)-2*z(t);
```

$$(\%o78) \qquad \frac{d}{dt}z(t) = -2\,z(t) + 4\,y(t) - 8\,x(t)$$

```
(%i79) desolve([e1,e2,e3],[x(t),y(t),z(t)]);
(%o79)
```

$$\begin{aligned}[x(t) = -4y(0)te^t - 2x(0)te^t - 6te^t + (x(0)+3)e^t + 6t - 3,\\ y(t) = 2y(0)te^t + x(0)te^t + 3te^t + y(0)e^t - 3t,\\ z(t) = \frac{40y(0)te^t}{3} + \frac{20x(0)te^t}{3} +\\ 20te^t - \frac{(28y(0)+44x(0)+132)e^t}{9} +\\ \frac{(9z(0)+28y(0)+44x(0)-111)e^{-2t}}{9}\\ -30t + 27]\end{aligned}$$

Fijémonos en que la solución está expresada en términos de los valores iniciales $x(0)$, $y(0)$ y $z(0)$. Para sustituir estas constantes por unos valores prefijados, digamos $x(0) = 0$, $y(0) = 1$, $z(0) = -1$, podemos utilizar el comando `ev`, cuya sintaxis es `ev(expr,arg1,...,argn)`, siendo *expr* la expresión en la que se quieren hacer las evaluaciones o sustituciones y *argi*, $i = 1, ..., n$, (los argumentos) cualquier regla que determine esas sustituciones: pueden ser asignaciones de variables, ecuaciones, funciones, etc...
En el sistema anterior, con las condiciones iniciales $x(0) = 0$, $y(0) = 1$, $z(0) = -1$, haríamos:

```
(%i80) ev(%o8,x(0)=0,y(0)=1,z(0)=1);
(%o80)
```

$$[x(t) = -10te^t + 3e^t + 6t - 3,$$
$$y(t) = 5te^t + e^t - 3t,$$
$$z(t) = \frac{100te^t}{3} - \frac{160e^t}{9} - \frac{74e^{-2t}}{9} - 30t + 27]$$

Si las condiciones iniciales se conocen de antemano, también podemos pasárselas a `desolve` *antes* de calcular la solución general (con lo cual evitamos que resulte una expresión muy larga e inmanejable, pues las condiciones iniciales ya estarán evaluadas). Esto se hace con el comando `atvalue`, en la forma `atvalue(`*expr*`,`*x*`=`*a*`,`*b*`)`, que asigna el valor b a la expresión *expr* cuando x toma el valor a.
Por ejemplo, para resolver el sistema

$$\begin{cases} x' = 3x + 8z + t \\ y' = 3x - y + 6z + t \\ z' = -2x - 5z + t \end{cases}$$

con las condiciones iniciales

$$x(0) = y(0) = z(0) = 1,$$

primero declaramos las ecuaciones:

```
(%i81) e4:'diff(x(t),t)=3*x(t)+8*z(t)+t;
```

$$(\%o81) \qquad \frac{d}{dt}\,x\,(t) = 8\,z\,(t) + 3\,x\,(t) + t$$

```
(%i82) e5:'diff(y(t),t)=3*x(t)-y(t)+6*z(t)+t;
```

$$(\%o82) \qquad \frac{d}{dt}\,y\,(t) = 6\,z\,(t) - y\,(t) + 3\,x\,(t) + t$$

```
(%i83) e6:'diff(z(t),t)=-2*x(t)-5*z(t)+t;
```

$$(\%o83) \qquad \frac{d}{dt}\,z\,(t) = -5\,z\,(t) - 2\,x\,(t) + t$$

Imponemos las condiciones iniciales, con el comando `atvalue`:

```
(%i84) atvalue(x(t),t=0,1);
```

$$(\%o84) \qquad 1$$

```
(%i85) atvalue(y(t),t=0,1);
```

(%o85) 1

```
(%i86) atvalue(z(t),t=0,1);
```

(%o86) 1

y finalmente integramos el sistema:

```
(%i87) desolve([e4,e5,e6],[x(t),y(t),z(t)]);
(%o87)
```

$$[x\,(t) = 24\,t\,e^{-t} + 26\,e^{-t} + 13\,t - 25,$$
$$y\,(t) = 18\,t\,e^{-t} + 20\,e^{-t} + 10\,t - 19,$$
$$z\,(t) = -12\,t\,e^{-t} - 10\,e^{-t} - 5\,t + 11]$$

Este método tiene la desventaja de que deja fijados los valores de $x(0)$, $y(0)$ y $z(0)$, de modo que si no nos damos cuenta podemos cometer errores más adelante al utilizar $x(t)$, $y(t)$ ó $z(t)$ como nuevas funciones. Para borrar la asignación que le hemos dado a la función $x(t)$ mediante `atvalue`, hacemos:

```
(%i88) remove(x,atvalue);
```

(%o88) *done*

```
(%i89) x(0);
```

(%o89) $x\,(0)$

Notemos que escribimos x y no $x(t)$ (esto es así porque el comando `remove` actúa sobre átomos: el nombre de la función $x(t)$, x, es un átomo mientras que la propia función no lo es).

3.3. Ecuaciones lineales de orden $n > 2$ y coeficientes constantes

Es sabido que cualquier ecuación diferencial lineal de orden $n > 2$, como

$$a_n(t)\frac{d^n x}{dt^n} + \cdots + a_1(t)\frac{dx}{dt} + a_0(t)x = b(t),$$

puede reducirse a un sistema de ecuaciones de primer orden. Dado que, como hemos visto, Maxima puede resolver una amplia clase de estos sistemas, es de esperar que también se puedan resolver ecuaciones diferenciales de orden mayor que 2.
De hecho, el método estándar para tratar tales ecuaciones es la transformada de Laplace, que es precisamente el que implementa `desolve`, así que lo podremos usar para integrar ecuaciones diferenciales lineales, con la condición de que sean de coeficientes *constantes*. Como ejemplo, consideremos la ecuación

$$\frac{d^3y}{dx^3}+\frac{d^2y}{dx^2}+\frac{dy}{dx}+y-x=0$$

```
(%i94) ec20:'diff(y(x),x,3)+'diff(y(x),x,2)+'diff(y(x),x)+y(x)-x=0;
```

(%o94)

```
(%i95) desolve(ec20,y(x));
(%o95)
```

$$y(x)=\frac{sin\left(x\right)\,\left(\left.\frac{d^2}{d\,x^2}\,y\left(x\right)\right|_{x=0}+2\,\left(\left.\frac{d}{d\,x}\,y\left(x\right)\right|_{x=0}\right)+y\left(0\right)-1\right)}{2}+\frac{e^{-x}\,\left(\left.\frac{d^2}{d\,x^2}\,y\left(x\right)\right|_{x=0}+y\left(0\right)+1\right)}{2}-\frac{cos\left(x\right)\,\left(\left.\frac{d^2}{d\,x^2}\,y\left(x\right)\right|_{x=0}-y\left(0\right)-1\right)}{2}+x-1$$

De nuevo, vemos que la solución contiene las condiciones iniciales. La forma de evaluar estas condiciones una vez que se dispone de la solución no es sencilla, debido a la presencia de derivadas. En este caso, es mejor dar las condiciones iniciales previamente con el comando `atvalue`, como en la sección anterior.
Por ejemplo, para resolver la ecuación $y'''+y''-2y'=5-e^x$ con las condiciones iniciales $y(0)=0=y'(0)$, $y''(0)=1$, procederemos como sigue. Primero, definimos la ecuación:

```
(%i96) ec21:'diff(y(x),x,3)+'diff(y(x),x,2)-2*'diff(y(x),x)=5-exp(x);
```

(%o96) $$\frac{d^3}{d\,x^3}\,y\left(x\right)+\frac{d^2}{d\,x^2}\,y\left(x\right)-2\,\left(\frac{d}{d\,x}\,y\left(x\right)\right)=5-e^x$$

y a continuación las condiciones iniciales:

```
(%i97) atvalue('diff(y(x),x,2),x=0,1);
```

(%o97) 1

```
(%i198) atvalue('diff(y(x),x),x=0,0);
```

(%o98) 0

```
(%i99) atvalue(y(x),x=0,0);
```

(%o99) 0

Finalmente, resolvemos la ecuación:

```
(%i100) desolve(ec21,y(x));
```

$$(\%o100) \qquad y\left(x\right)=-\frac{x\,e^{x}}{3}+\frac{22\,e^{x}}{9}-\frac{7\,e^{-2\,x}}{36}-\frac{5\,x}{2}-\frac{9}{4}$$

y simplificamos el resultado:

```
(%i101) fullratsimp(%);
```

$$(\%o101) \qquad y\left(x\right)=-\frac{e^{-2\,x}\,\left(\left(12\,x-88\right)\,e^{3\,x}+\left(90\,x+81\right)\,e^{2\,x}+7\right)}{36}$$

Para evitar problemas si más tarde usamos el símbolo y para denotar otra función, eliminamos la asignación hecha con `atvalue` sobre y y sus derivadas:

```
(%i102) remove(y,atvalue);
```

(%o102) *done*

4. Teoría de la estabilidad

4.1. Estabilidad según Lyapunov y estabilidad asintótica

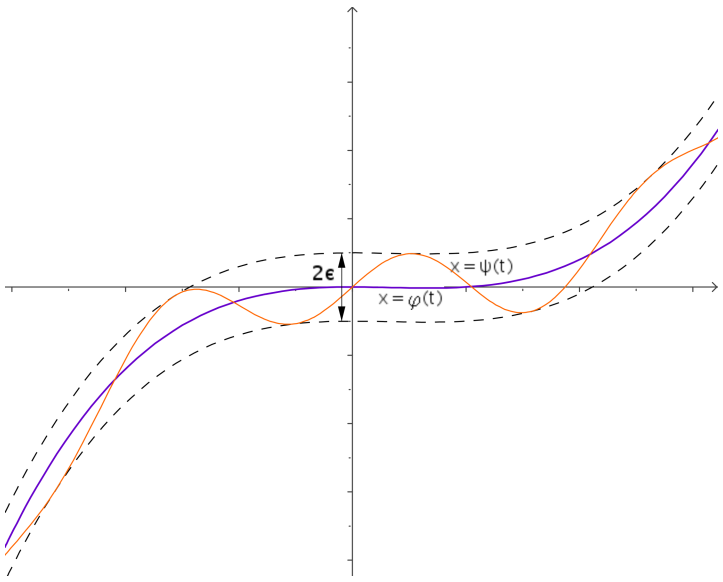
Dado un sistemas de ecuaciones diferenciales (no necesariamente lineales)

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}, t) \tag{2}$$

denotaremos a sus funciones soluciones por $\mathbf{x} = \varphi(t)$, $\mathbf{x} = \psi(t)$, etc. Obviamente, aquí es $\mathbf{x}(t) \in \mathbb{R}^n$, y $f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$.
La solución $\mathbf{x} = \varphi(t)$ del sistema (2) se dice que es *estable (en sentido Lyapunov)* si y sólo si para todo $\varepsilon > 0$ existe un δ_ε tal que si otra solución $\mathbf{x} = \psi(t)$ cumple $||\varphi(t_0) - \psi(t_0)|| < \delta_\varepsilon$ (i.e, en un instante inicial), entonces

$$||\varphi(t) - \psi(t)|| < \varepsilon, \quad \forall t > t_0$$

(i.e, en cualquier otro instante posterior).
Gráficamente, tenemos la siguiente situación:



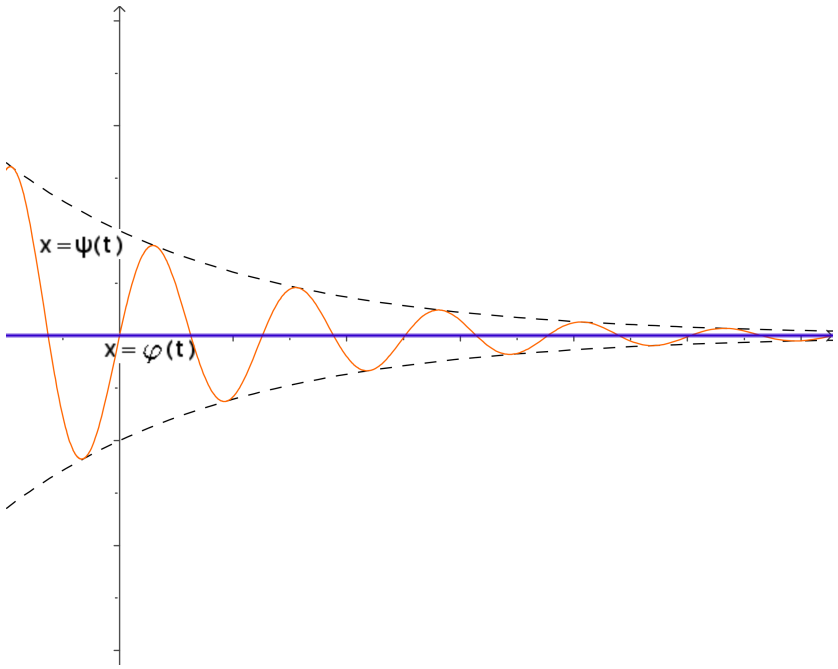
En palabras: una solución es estable si cualquier otra solución que está cerca de ella en un instante inicial, permanece cerca de ella en cualquier instante posterior (esto *no* implica que las soluciones converjan, como se aprecia en la figura una puede oscilar alrededor de la otra, relacionado con esto tenemos la definición de estabilidad asintótica más abajo). Por el contrario, si una solución es inestable significa que al menos hay otra solución que inicialmente está cerca de ella y al cabo de un cierto tiempo se ha separado tanto como queramos.

Una solución $\mathbf{x} = \varphi(t)$ del sistema (2) se dice que es *asintóticamente estable* si y sólo si es estable y, además, existe un $\delta > 0$ tal que para toda otra solución

$\mathbf{x} = \psi(t)$ que cumpla $||\varphi(t_0) - \psi(t_0)|| < \delta$ ocurre que

$$||\varphi(t) - \psi(t)|| \to 0, \quad si \ t \to 0.$$

Nuevamente en palabras, una solución φ es estable si (además de ser estable) cualquier otra solución ψ que en un cierto instante pase cerca de ella, al hacer $t \to 0$ se aproxima a φ tanto como queramos.
Gráficamente:



Las correspondientes nociones de estabilidad y estabilidad asintótica para una ecuación diferencial de orden n, cuya forma general es

$$x^{(n)}(t) = f(t, x(t), x'(t), ..., x^{(n-1)}(t),$$

son las de su sistema de ecuaciones de primer orden asociado, obtenido definiendo

$$x_1(t) := x(t), x_2(t) := x'(t), ..., x_n(t) := x^{(n-1)}(t),$$

y que tiene el aspecto

$$\begin{cases} x_1'(t) = x_2(t), \\ x_2'(t) = x_3(t), \\ \quad \vdots \qquad \vdots \\ x_{n-1}'(t) = x_n(t), \\ x_n'(t) = -a_0(t)x_1(t) - a_1(t)x_2(t) - \cdots - a_{n-1}(t)x_n(t) + b(t). \end{cases} \tag{3}$$

Desde el punto de vista práctico, la estabilidad de un sistema reviste una especial importancia. Por ejemplo, en Física las mediciones nunca son exactas, sino que vienen afectadas por un cierto error. Supongamos que estamos realizando mediciones sobre un sistema físico cuya evolución está descrita por el sistema de ecuaciones

$$\begin{cases} \dot{x} = x + y \\ \dot{y} = y. \end{cases}$$

Este sistema tiene la propiedad de que su única solución con las condiciones iniciales $x(0) = 0$, $y(0) = 0$ es la aplicación constante igual a $(0, 0)$:

```
(%i1) ec1:'diff(x(t),t)=x(t)+y(t);
```

$$(\%o1) \qquad \frac{d}{d\,t}\,x\,(t) = y\,(t) + x\,(t)$$

```
(%i2) ec2:'diff(y(t),t)=y(t);
```

$$(\%o2) \qquad \frac{d}{d\,t}\,y\,(t) = y\,(t)$$

```
(%i3) desolve([ec1,ec2],[x(t),y(t)]);
```

$$(\%o3) \qquad [x\,(t) = y\,(0)\ t\,e^{t} + x\,(0)\ e^{t}, y\,(t) = y\,(0)\ e^{t}]$$

```
(%i4) ev(%,x(0)=0,y(0)=0);
```

$$(\%o4) \qquad [x\,(t) = 0, y\,(t) = 0]$$

Supongamos también que nuestros aparatos de medida son tales que tienen un error del orden de 10^{-3} (por ejemplo, utilizamos una regla cuyas divisiones son de 1mm). Si en el instante inicial $t = 0$ medimos x e y obteniendo $(x(0), y(0)) = (0.001, 0.001)$, ¿qué implicará la estabilidad en este caso?. En principio, como el error de nuestros aparatos es de 10^{-3}, las medidas que hemos hecho deberían expresarse como

$$(x(0), y(0)) = (0.001, 0.001) \pm 0.001$$

y así se ve más claro que estas medidas son compatibles tanto con el hecho de que el sistema se encuentre en el estado $(x(0), y(0)) = (0, 0)$ como con el que se encuentre en el $(x(0), y(0)) = (0.002, 0.002)$. La diferencia es que si se da el primer caso, el sistema permanecerá en $(0, 0)$ indefinidamente, según `(%o4)`, mientras que si se da el segundo caso la evolución del sistema estará regida por la ecuación obtenida fijando las condiciones iniciales $(x(0), y(0)) = (0.002, 0.002)$:

```
(%i5) ev(%o3,x(0)=0.002,y(0)=0.002);
```

$$(\%o5) \qquad [x(t) = 0.002\,t\,e^t + 0.002\,e^t, y(t) = 0.002\,e^t]$$

Entonces, si nuestra unidad de medida para x, y es el metro y para t son segundos, al cabo de $t = 5$ segundos el sistema se encontrará en la posición

```
(%i6) float(ev(%o5,t=5));
```

$$(\%o6) \qquad [x(5.0) = 1.780957909230919, y(5.0) = .2968263182051532]$$

Es decir: desde un punto de vista práctico, no podemos asegurar nada sobre el comportamiento del sistema. El motivo, claramente, es la inestabilidad del mismo, puesta de manifiesto con el comportamiento de estas dos soluciones.

Nota: Fijémonos en que, por definición, la estabilidad asintótica implica estabilidad, pero el recíproco no es cierto. Un ejemplo trivial lo proporciona la ecuación $\dot{x} = 0$, cuyas soluciones son de la forma $x(t) = k$, con $k \in \mathbb{R}$ constante. Todas estas soluciones son estables (basta tomar $\delta = \varepsilon$ en la definición), pero ninguna es asintóticamente estable (si una solución $\varphi(t)$ corresponde a k_1 y otra $\psi(t)$ a k_2, siempre se mantienen a una distancia $||\varphi(t) - \psi(t)|| = |k_1 - k_2|$).

Si el sistema (2) es tal que el miembro derecho no depende explícitamente de t, esto es, $\dot{\mathbf{x}}(t) = f(\mathbf{x})$, se dice que el sistema es *autónomo*. Se llaman entonces *soluciones de equilibrio* a las funciones constantes $\mathbf{x}(t) = \mathbf{x}^*$, donde $\mathbf{x}^*$ es una solución a la ecuación $f(\mathbf{x}) = 0$. Veamos un ejemplo: se trata de estudiar la estabilidad de la solución de equilibrio $x = 0$ a la ecuación $\dot{x} = x^2$ (ésta es la única solución de equlibrio de este sistema). Según acabamos de ver, la estabilidad significa que soluciones $\varphi(t)$ cuya gráfica está en algún momento t_0 cerca del eje de abcisas $x = 0$, permanecerán cerca del mismo para valores de t muy alejados de t_0 (si es con $t < t_0$ se dice que hay estabilidad por la izquierda, y si es con $t > t_0$, que hay estabilidad por la derecha. "Estabilidad" sin más significa estabilidad por la izquierda y por la derecha).
Calculamos con Maxima la solución general de esta ecuación:

```
(%i7) ode2('diff(x,t)=x^2,x,t);
```

$$(\%o7) \qquad -\frac{1}{x} = t + \%c$$

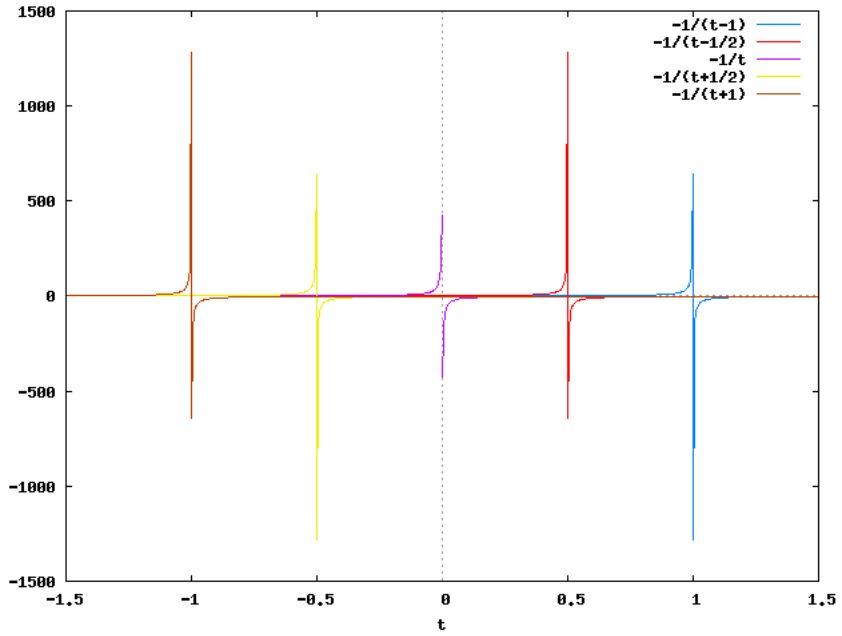
```
(%i8) solve(%,x);
```

$$(\%o8) \qquad [x = -\frac{1}{t + \%c}]$$

A continuación, representamos algunas de estas soluciones (para los valores de la constante de integración $c \in \{-1, -1/2, 0, 1/2, 1\}$ y sobre el intervalo $[-1.5, 1.5]$):

```
(%i9) wxplot2d(makelist(-1/(t+(i/2)),i,-2,2),[t,-1.5,1.5]);
```

(%o9)



Si quisiéramos guardar la imagen que genera `gnuplot`, haríamos (este comando guarda la imagen con formato PNG (Portable Network Graphics) y el nombre *inestable.png*):

```
(%i10) plot2d(
makelist(-1/(t+(i/2)),i,-2,2),
[t,-1.5,1.5],
[gnuplot_preamble, "set terminal png; set output 'inestable.png'"]
)$
```

A la vista del gráfico, resulta claro que $x = 0$ no es estable: hay soluciones que están cerca de ella casi para cualquier valor de t, pero de repente "explotan" en una singularidad y se van a infinito (las funciones elegidas tienen sus singularidades en los puntos $t = \pm\frac{i}{2}$).

Otro ejemplo. Vamos a estudiar gráficamente la estabilidad de la solución de equilibrio $x = 1$ de la ecuación $\dot{x} = 1 - x^2$ (esta ecuación admite dos soluciones de equilibrio: $x = \pm 1$). Comenzamos pidiéndole a Maxima que resuelva la ecuación:

```
(%i11) ode2('diff(x,t)=1-x^2,x,t);
```

(%o11)
$$\frac{log\,(x+1)-log\,(x-1)}{2}=t+\,\%c$$

Notemos que $x=1$ es una solución singular de la ecuación, que no está contenida en esta fórmula (si tratamos de imponer la condición inicial $x=1$ para $t=0$, Maxima se quejará de que aparece log(0), ¿por qué?).
Queremos ver qué ocurre con soluciones a la ecuación que están cerca del eje horizontal $x=1$. Para ello, vamos a considerar algunas soluciones cuyas condiciones iniciales sean tales que en $t=1$ estén cerca de $x=1$; por ejemplo, tomaremos las soluciones que en $t=1$ valen 1.01, 1.001 y 1.0001:

```
(%i12) makelist(ic1(%o11,t=1,x=1+10^(-i)),i,2,4);
(%o12)
```

$$[\frac{log\,(x+1)-log\,(x-1)}{2}=\frac{2\,t+log\,(100)+log\left(\frac{201}{100}\right)-2}{2},$$
$$\frac{log\,(x+1)-log\,(x-1)}{2}=\frac{2\,t+log\,(1000)+log\left(\frac{2001}{1000}\right)-2}{2},$$
$$\frac{log\,(x+1)-log\,(x-1)}{2}=\frac{2\,t+log\,(10000)+log\left(\frac{20001}{10000}\right)-2}{2}]$$

```
(%i13) logcontract(%o12);
(%o13)
```

$$[-\frac{log\left(\frac{x-1}{x+1}\right)}{2}=\frac{2\,t+log\,(201)-2}{2},$$
$$-\frac{log\left(\frac{x-1}{x+1}\right)}{2}=\frac{2\,t+log\,(2001)-2}{2},$$
$$-\frac{log\left(\frac{x-1}{x+1}\right)}{2}=\frac{2\,t+log\,(20001)-2}{2}]$$

Aquí se ve claramente que la estructura de estas soluciones es una ecuación del tipo
$$\frac{x-1}{x+1}=\exp(-2t-\log(2\cdot 10^i+1)+2),$$
que podemos pedirle a Maxima que nos resuelva:

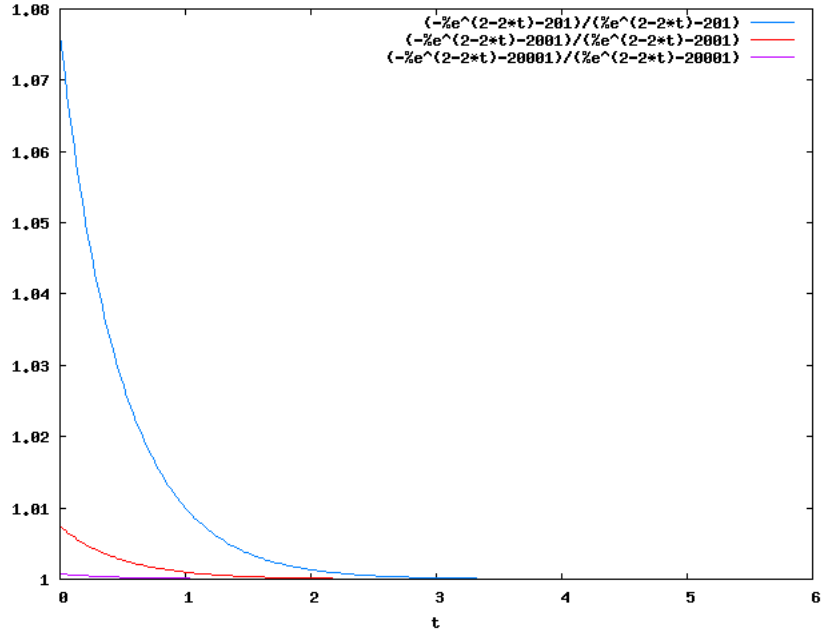
```
(%i14) solve((x-1)/(x+1)=exp(-2*t-log(2*(10^i)+1)+2),x);
```

(%o14)
$$[x=-\frac{e^{2-2\,t}+2\,10^i+1}{e^{2-2\,t}-2\,10^i-1}]$$

Con esto, ya podemos representar gráficamente estas soluciones cercanas a la $x=1$:

```
(%i15) wxplot2d(
makelist(-(%e^(2-2*t)+2*10^i+1)/(%e^(2-2*t)-2*10^i-1),i,2,4),
[t,0,6]);
```

(%o15)



Se ve claramente que a partir de $\delta = 0.01$ (es decir, valores $\delta < 0.01$), las funciones se aproximan muy rápidamente a $x = 1$. Según la definición es, pues, una solución asintóticamente estable.
Como ejercicio, el lector puede estudiar de qué tipo es la otra solución de equilibrio de este sistema, $x = -1$.

4.2. Estabilidad de soluciones de sistemas lineales

En esta sección, particularizaremos el estudio a los sistemas lineales. En este caso, (2) se escribe como

$$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + b(t), \tag{4}$$

donde $\mathbf{x}(t) \in \mathbb{R}^n$ para cada $t \in [t_0, t_1]$ y $A(t) \in \mathcal{M}at_{n\times n}(\mathbb{R})$.
Las soluciones a (4) *no* forman un espacio vectorial (esto es, la suma de soluciones y el producto de una solución por un escalar no son soluciones, en general). Sin embargo, la diferencia de dos soluciones *sí* es una solución, lo que se traduce en que éstas tienen estructura de espacio afín (de hecho, $n-$dimensional). Como consecuencia de esta propiedad, es inmediato el siguiente resultado:

Teorema. *Una solución* $\mathbf{x} = \varphi(t)$ *del sistema lineal* (4) *es estable (respectivamente, asintóticamente estable) si y sólo si la solución trivial* $\mathbf{x} \equiv 0$ *del sistema homogéneo asociado*

$$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t)$$

es estable (respectivamente, asintóticamente estable).

Nota importante: Como consecuencia de este resultado, la estabilidad o estabilidad asintótica *de un sistema lineal* es una característica propia del sistema: o todas las soluciones son estables o no lo es ninguna. Así, diremos que un sistema es estable si lo son todas sus soluciones, e inestable en caso contrario.

Otra consecuencia es que el estudio de la estabilidad de un sistema no homogéneo se reduce al estudio de la estabilidad del sistema homogéneo asociado. De hecho, si el sistema homogéneo es tal que la matriz A cumple que $\det(A(t)) \neq 0$ para todo t, entonces todo se reduce a estudiar la estabilidad de la única solución de equilibrio $\mathbf{x} = 0$.

El problema de todo esto es que los criterios de estabilidad que existen para este caso se dan en términos de una matriz fundamental del sistema, para obtener la cual no hay métodos generales. Únicamente en casos especiales (como $A(t) = A$ una matriz constante, que veremos más adelante) se pueden dar resultados más concretos, referidos directamente a la matriz A.

Quizás el resultado más sencillo a este respecto sea el siguiente.

Teorema. *Si todas las soluciones de un sistema lineal*

$$\dot{\mathbf{x}}(t) = A(t)\mathbf{x}(t) + b(t)$$

son acotadas, entonces el sistema es estable.

Como ejemplo, consideremos el sistema bidimensional

$$\begin{cases} \dot{x} = y + \cos(t) \\ \dot{y} = -x + \sin(t). \end{cases}$$

Resolviéndolo mediante `desolve`:

```
(%i16) desolve(
['diff(x(t),t)=y(t)+cos(t),'diff(y(t),t)=-x(t)+sin(t)],[x(t),y(t)]
);
```

(%o16)
$$[x\left(t\right)=\left(y\left(0\right)+1\right)\ sin\left(t\right)+x\left(0\right)\ cos\left(t\right),y\left(t\right)=y\left(0\right)\ cos\left(t\right)-x\left(0\right)\ sin\left(t\right)]$$

Dado que $|\sin(t)|, |\cos(t)| \leq 1$, aplicando la desigualdad triangular se obtiene la acotación (para cada curva solución que pasa por un punto $(x(0), y(0))$):

$$\begin{aligned} |x(t)| &\leq |y(0) + 1| + |x(0)| \\ |y(t)| &\leq |y(0)| + |x(0)| \end{aligned}$$

por lo que el sistema es estable.
Señalemos, sin embargo, la limitación de este criterio: sólo nos habla de estabilidad cuando lo importante, en al práctica, es la estabilidad asintótica.

4.3. Estabilidad de sistemas lineales autónomos

Consideraremos en esta sección el caso en que la matriz de coeficientes A *no* depende del parámetro t. En este caso, se tiene un sistema autónomo.
Como ya se ha señalado, para este caso particular si es posible enunciar resultados sobre la estabilidad del sistema que sólo hacen referencia a la matriz A. El más habitual de éstos es el siguiente.

Teorema. *Sea el sistema lineal autónomo*

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + b(t).$$

Entonces:

(a) El sistema es asintóticamente estable si y sólo si los valores propios de la matriz A tienen todos parte real estrictamente *negativa.*

(b) El sistema es estable si y sólo existen valores propios de A con parte real nula (es decir, imaginarios puros), siendo todos ellos raíces simples *del polinomio mínimo anulador de A, y los restantes (si existen) tienen parte real negativa.*

(c) El sistema es inestable en otro caso.

Nota importante: Fijémonos en que si se da el supuesto (b), por (a) el sistema es estable pero nunca asintóticamente estable.

En el enunciado del teorema, aparece el polinomio mínimo de A (no su polinomio característico). Recordemos que éste es el (único) polinomio mónico $m_A(x) = x^n + c_{n-1}x^{n-1} + \cdots + c_1x + c_0$ de menor grado tal que A satisface su ecuación asociada, esto es, $m_A(A) = 0$. Por el Teorema de Cayley-Hamilton, el polinomio mínimo siempre es un divisor del característico. Se cumple, pues, que si el polinomio característico de A es de la forma $\chi_A(x) = (x-\lambda_1)^{n_1} \cdots (x-\lambda_k)^{n_k}$ entonces $m_A(x) = (x-\lambda_1)^{m_1} \cdots (x-\lambda_k)^{m_k}$ para ciertos naturales $1 \leq m_i \leq n_i$. Es útil saber que si todas las raíces del polinomio característico son simples, entonces $\chi_A(x) = m_A(x)$.

Veamos un ejemplo. Estudiemos en función del parámetro $a \in \mathbb{R}$ la estabilidad del sistema diferencial $\dot{\mathbf{x}}(t) = A\mathbf{x}(t) + b(t)$, donde

$$A = \begin{pmatrix} 3-a & 2 & 1 \\ -3 & -(2+a) & 0 \\ a-3 & a-3 & -4 \end{pmatrix}$$

```
(%i17) A:matrix([3-a,2,1],[-3,-(2+a),0],[a-3,a-3,-4]);
```

(%o17)
$$\begin{pmatrix} 3-a & 2 & 1 \\ -3 & -a-2 & 0 \\ a-3 & a-3 & -4 \end{pmatrix}$$

Para estudiar los invariantes algebraicos de una matriz, conviene cargar el paquete `eigen`:

```
(%i18) load(eigen)$
```

El polinomio característico $\chi_A(x)$, por ejemplo, se obtiene haciendo:

```
(%i19) charpoly(A,x);
```

(%o19)
$(-x-4)\ (-x-a-2)\ (-x-a+3)-(a-3)\ (-x-a-2)+6\ (-x-4)-3\ (a-3)$

```
(%i20) factor(%);
```

(%o20) $$-(x+3)\ (x+a-1)\ (x+a+1)$$

donde se ve claramente que los valores propios son $\lambda_1 = -3$, $\lambda_2 = 1-a$ y $\lambda_3 = -(1+a)$, todos ellos simples (por lo que en este caso son también las raíces del polinomio mínimo). Una vez que se tiene definida una matriz como la A, el cálculo de los valores y vectores propios se puede hacer de forma directa. El comando `eigenvectors` nos devuelve una "lista de listas", donde la primera lista contiene los valores propios con sus multiplicidades y la segunda lista nos da los vectores propios correspondientes. El comando `eigenvalues` nos da una lista con los valores propios y otra lista con sus correspondientes multiplicidades:

```
(%i21) eigenvalues(A);
```

(%o21) $$[[-3, -a-1, 1-a], [1,1,1]],$$

resultado que coincide con el que ya habíamos obtenido calculando el polinomio característico.
En este caso es $\chi_A(x) = m_A(x)$, pero en general podemos calcular el polinomio mínimo sin cálculos previos, cargando el paquete `diag` que contiene los comandos `jordan` y `minimalPoly`. Combinados, obtenemos el polinomio mínimo:

```
(%i22) load(diag)$
(%i23) minimalPoly(jordan(A));
```

(%o23) $$(x+3)\ (x+a-1)\ (x+a+1)$$

Veamos ahora cuándo se da la estabilidad asintótica. Para ello, los tres valores propios deben ser estrictamente negativos. Como $\lambda_1 = -3$ cumple esta condición, sólo debemos averiguar cuándo ocurre que $\lambda_2 = 1-a$ y $\lambda_3 = -(1+a)$ son estrictamente negativos.
Para resolver inecuaciones, podemos recurrir al comando `fourier_elim`, cargando el paquete con el mismo nombre. Las desigualdades se pasan como una lista y en otra lista se dan las variables independientes:

```
(%i24) load(fourier_elim)$

(%i25) fourier_elim([1-a<0,-1-a<0],[a]);
```

(%o25) $[1 < a]$

Así, para $1 < a$ tenemos estabilidad asintótica.
Para estudiar la estabilidad, debemos admitir la posibilidad de que haya valores propios con parte real nula. En nuestro caso, esto sólo puede darse si $1 - a = 0$ ó $-(1 + a) = 0$, es decir, cuando $a = 1$ ó $a = -1$. El segundo caso conduce a $\lambda_2 = 2 > 0$ y, por tanto, a un sistema inestable. En el primer caso, resulta $\lambda_2 = 0$ y $\lambda_3 = -2$, con lo que estamos en el supuesto (b) del teorema y hay estabilidad (pero no asintótica). Para resumir:

1. El sistema es asintóticamente estable si $a > 1$.
2. El sistema es estable si $a = 1$.
3. El sistema es inestable si $a < 1$.

Nota: Antes de que se generalizase el uso de equipos de cómputo, se dedicó mucho esfuerzo a encontrar criterios para saber cuándo las raíces de un polinomio dado tenían parte real negativa. Uno de estos resultados se cita en la literatura con mucha frecuencia: el Teorema de Routh-Hurwitz. Sin embargo, las posibilidades de un sistema de álgebra computacional como Maxima, sumadas a la potencia de cálculo de cualquier computadora de sobremesa actual, hacen que tales criterios sean cada vez menos útiles desde el punto de vista práctico (aunque mantienen su interés teórico). Por ejemplo, en caso de que `eigenvalues` no nos devuelva los resultados en una forma "tratable", podemos calcular las raíces de cualquier polinomio numéricamente de manera casi inmediata y con una aproximación muy buena con el comando `allroots`, o bien con el `realroots` si sólo nos interesan las raíces reales. Por esta razón, no trataremos estos resultados aquí.

5. Clasificación de órbitas en sistemas autónomos planos

Como se sabe del Álgebra Lineal, dada una matriz cualquiera (con coeficientes en un cuerpo $\mathbb{K}$)

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdot & \cdot & \cdots & \cdot \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

siempre se puede poner en una forma más simple, diagonal o casi diagonal, mediante una transformación de semejanza $A \mapsto P^{\top} AP$. La matriz de paso P se obtiene a partir de un estudio de los valores y vectores propios de A.
Ahora bien, en el caso de una matriz real 2×2,

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

los valores y vectores propios resultan de la ecuación del polinomio característico, que en este caso se puede calcular como:

```
(%i1) A:matrix([a,b],[c,d]);
```

(%o1)
$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

```
(%i2) load(nchrpl)$
(%i3) %chi ('A)=ncharpoly(A,%lambda);
```

(%o3)
$$\chi\left(A\right) = a\,d + \lambda\,\left(-d - a\right) - b\,c + \lambda^2$$

En esta expresión, reconocemos la traza de A, $\mathrm{tr}(A) = a + d$ y su determinante $\det(A) = ad - bc$, de modo que

$$\chi(A) = \lambda^2 - \mathrm{tr}(A)\lambda + \det(A).$$

No es de extrañar, entonces, que la forma "canónica" que adopte A dependa de los valores de $\mathrm{tr}(A)$ y $\det(A)$.
Por ejemplo, se sabe (de nuevo del curso de Álgebra Lineal) que si los valores propios son reales y distintos $\lambda_1 \neq \lambda_2$, entonces la matriz A se puede transformar mediante una semejanza P (cuyas columnas son precisamente los vectores propios) en la matriz

$$\begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \tag{5}$$

Las condiciones para que esto suceda consisten en que el discriminante del polinomio característico sea estrictamente positivo, esto es, que sea

$$\mathrm{tr}(A)^2 - 4\mathrm{det}(A) > 0.$$

Esto es lo mismo que decir que el punto en el plano dado por $(\mathrm{tr}(A), \mathrm{det}(A))$ está *por debajo* de la parábola $\mathrm{det}(A) = \frac{1}{4}\mathrm{tr}(A)^2$. De hecho, se tiene la siguiente información algebraica sobre los valores propios:

1. Si $\mathrm{tr}(A)^2 - 4\mathrm{det}(A) > 0$ (puntos por debajo de la parábola $\mathrm{det}(A) = \frac{1}{4}\mathrm{tr}(A)^2$), entonces λ_1, λ_2 son reales y distintos.
2. Si $\mathrm{tr}(A)^2 - 4\mathrm{det}(A) = 0$ (puntos sobre la parábola $\mathrm{det}(A) = \frac{1}{4}\mathrm{tr}(A)^2$), entonces $\lambda_1 = \lambda_2 \in \mathbb{R}$
3. Si $\mathrm{tr}(A)^2 - 4\mathrm{det}(A) < 0$ (puntos por encima de la parábola $\mathrm{det}(A) = \frac{1}{4}\mathrm{tr}(A)^2$), λ_1, λ_2 son complejos conjugados.

Nota: Fijémonos también en lo siguiente. Expresando el polinomio característico como producto de monomios $(\lambda - \lambda_i)$, resulta

```
(%i4) expand((%lambda -%lambda[1])*(%lambda -%lambda[2]));
```

$$(\%o4) \qquad \lambda^2 - \lambda_2\,\lambda - \lambda_1\,\lambda + \lambda_1\,\lambda_2$$

y comparando con `(%o3)`, vemos que

$$\mathrm{tr}(A) = \lambda_1 + \lambda_2, \;\; \mathrm{det}(A) = \lambda_1\lambda_2.$$

Consideremos ahora un sistema lineal autónomo de ecuaciones en el plano

$$\dot{\mathbf{x}}(t) = A\mathbf{x}(t) \tag{6}$$

(con $\mathbf{x} = (x, y)$ función vectorial) y veamos cómo clasificar el comportamiento de sus soluciones (u órbitas) en función de la matriz A.
Estas soluciones son curvas paramétricas $x = x(t)$, $y = y(t)$, pero a nosotros nos va a interesar su forma explícita $y = y(x)$ en el plano, obtenida eliminando t en las ecuaciones paramétricas (se habla entonces de $y = y(x)$ como de una órbita en el plano de fases (x, y)).[1]

[1] Consideremos un sistema físico autónomo descrito por una ecuación diferencial de segundo orden, como la de Newton, $\ddot{x}(t) = F(x, \dot{x})$. Por el teorema de existencia y unicidad de soluciones al problema d Cauchy, la evolución del sistema queda determinada una vez que se da un par de valores (condiciones iniciales) $(x_0, \dot{x}_0)$; es por esto que al plano con coordenadas $(x, \dot{x})$ se le llama *espacio de estados* o *espacio de fases* del sistema. Como ya sabemos, una ecuación de segundo orden es equivalente a un sistema de dos ecuaciones de primer orden; en este caso las ecuaciones serían $\dot{x} = y$, $\dot{y} = F(x, y)$, cuyas soluciones son curvas paramétricas en el plano de fases.

5.1. Caso $\det(A) < 0$

Aquí es $\lambda_1\lambda_2 < 0$, por lo que los valores propios de A son reales y distintos (de signo contrario). Dado que uno de ellos siempre es (real) positivo la solución nula (y, por tanto, el sistema) es *inestable*.
Para representar las órbitas, nada mejor que resolver explícitamente el sistema. Supondremos los índices asignados de manera que $\lambda_1 < 0 < \lambda_2$. Como ya hemos comentado, la matriz A puede expresarse mediante una semejanza en la forma (5) y entonces la solución al sistema de ecuaciones (6) viene dada por:

```
(%i5) eq1:'diff(x(t),t)=%lambda[1]*x(t);
```

(%o5)
$$\frac{d}{d\,t}\,x\,(t) = \lambda_1\,x\,(t)$$

```
(%i6) eq2:'diff(y(t),t)=%lambda[1]*y(t);
```

(%o6)
$$\frac{d}{d\,t}\,y\,(t) = \lambda_1\,y\,(t)$$

```
(%i7) desolve([eq1,eq2],[x(t),y(t)]);
```

(%o7)
$$[x\,(t) = x\,(0)\;e^{\lambda_1\,t}, y\,(t) = y\,(0)\;e^{\lambda_1\,t}]$$

Esta es la solución paramétrica. Para obtener la solución explícita, despejamos t de ambas funciones e igualamos los resultados, despejando y de la ecuación resultante:

```
(%i8) solve(x=x[0]*exp(%lambda[1]*t),t);
```

(%o8)
$$[t = \frac{log\left(\frac{x}{x_0}\right)}{\lambda_1}]$$

```
(%i9) solve(y=y[0]*exp(%lambda[2]*t),t);
```

(%o9)
$$[t = \frac{log\left(\frac{y}{y_0}\right)}{\lambda_2}]$$

```
(%i10) radcan(
solve(log(x/x[0])*%lambda[2]/%lambda[1]=log(y/y[0]),y)
);
```

(%o10) $$[y = \frac{y_0\, x^{\frac{\lambda_2}{\lambda_1}}}{x_0^{\frac{\lambda_2}{\lambda_1}}}]$$

Escrito de una forma un poco más elegante, esto es

$$y = y_0 \left(\frac{x}{x_0}\right)^{\frac{\lambda_2}{\lambda_1}} \tag{7}$$

Este caso se denomina *punto de silla*.
Por ejemplo, consideremos el sistema

$$\begin{cases} \dot{x} = 3x - 2y \\ \dot{y} = 4x - 3y \end{cases}$$

Los valores propios de su matriz de coeficientes los podemos calcular con el comando `eigenvalues`:

```
(%i11) eigenvalues(matrix([3,-2],[4,-3]));
```

(%o11) $$[[-1, 1], [1, 1]]$$

Luego $\lambda_1 = -1$, $\lambda_2 = 1$. Las órbitas del sistema en el plano de fases, según hemos visto, vienen dadas por (7) que en este caso se reduce a

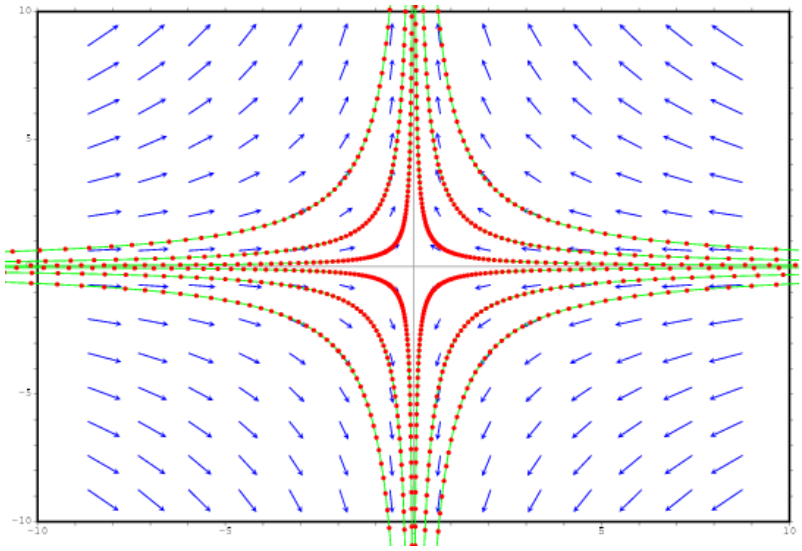
$$y = y_0 \left(\frac{x_0}{x}\right)$$

Esta es la ecuación de las órbitas para el sistema expresado en forma canónica. El sistema original tiene el mismo comportamiento cualitativo, pero las órbitas se transforman mediante la matriz P de paso (en general, las órbitas del sistema en forma canónica y las del sistema original difieren en rotaciones y homotecias). Para representar las órbitas, utilizaremos el paquete `plotdf`:

```
(%i12) load(plotdf)$
```

Una vez que tenemos el paquete cargado, representamos el sistema introduciendo únicamente el miembro derecho de su ecuación en forma de lista. En el gráfico que aparece, al hacer pulsar con el botón izquierdo del ratón en un punto (x_0, y_0), automáticamente se dibuja la (única) órbita que tiene ese punto como condición inicial. Por ejemplo, las órbitas del sistema canónico son:

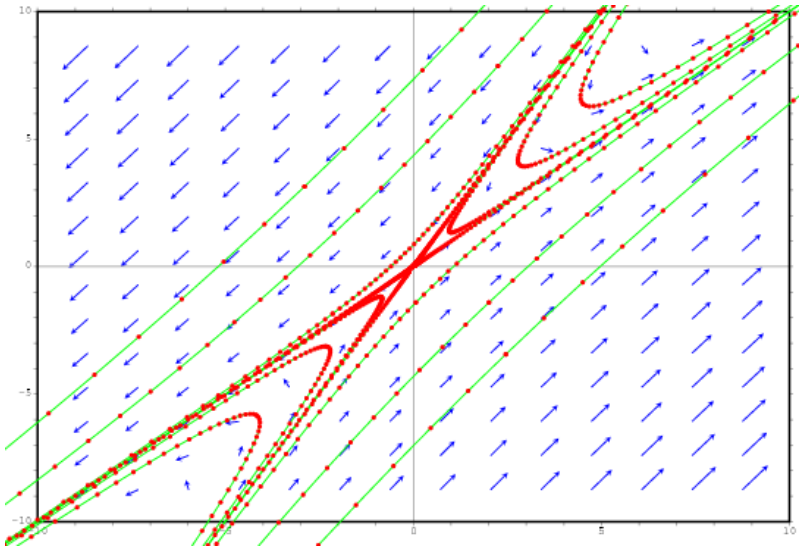
```
(%i13) plotdf([-x,y]);
```



(%o13)

Y las del sistema original:

```
(%i14) plotdf([3*x-2*y,4*x-3*y]);
```



(%o13)

5.2. Caso $\mathbf{tr}(A)^2 \geq 4\mathbf{det}(A) > 0$

Aquí, los valores propios son reales no nulos, distintos y tienen el mismo signo: $\lambda_1 \cdot \lambda_2 > 0$. Por tanto, habrá un caso asintóticamente estable (cuando

$\lambda_1, \lambda_2 < 0$) y otro inestable (cuando $\lambda_1, \lambda_2 > 0$).
El análisis de las soluciones explícitas es exactamente el mismo de antes (ya que los valores propios son reales distintos, la matriz es diagonalizable). Lo que cambia es que ahora el factor $\frac{\lambda_2}{\lambda_1}$ es positivo y que el comportamiento cualitativo es distinto si $\lambda_1, \lambda_2 < 0$ ó $\lambda_1, \lambda_2 > 0$.
Veamos un ejemplo de cada caso. El primero es el sistema

$$\begin{cases} \dot{x} = y \\ \dot{y} = -2 * x + 3 * y \end{cases}$$

```
(%i15) eigenvalues(matrix([0,1],[-2,3]));
```

(%o15) $$[[1, 2], [1, 1]]$$

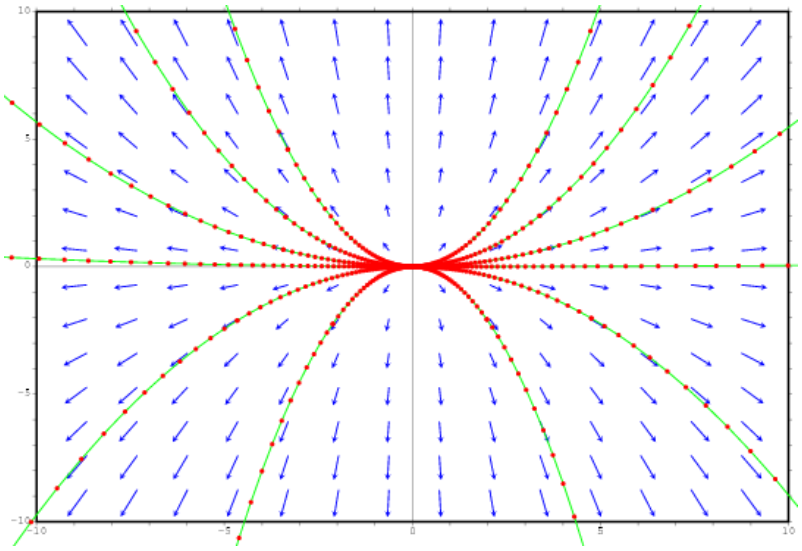
Sus valores propios son, pues, $\lambda_1 = 1$ y $\lambda_2 = 2$, ambos positivos. La solución explícita, obtenida sustituyendo valores en (7), es

$$y = y_0 \left(\frac{x}{x_0} \right)^2$$

que es una colección de parábolas. Representamos las órbitas, primero del sistema en forma canónica (diagonalizado):

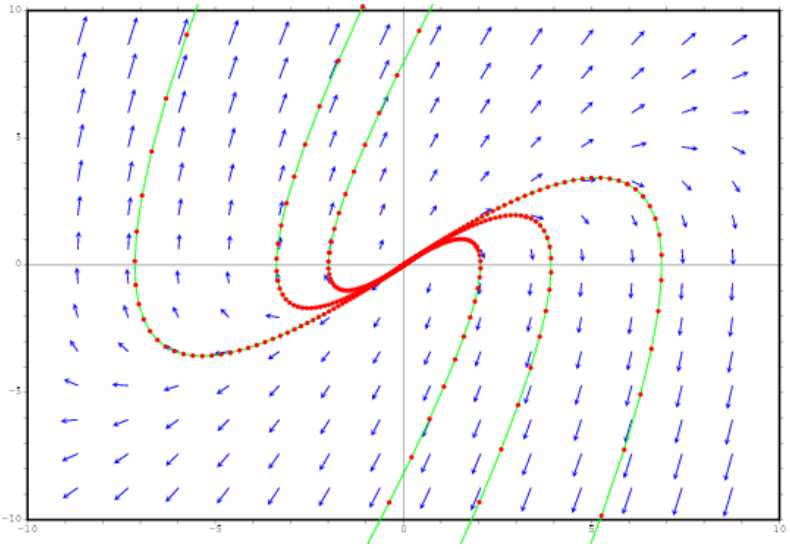
```
(%i16) plotdf([x,2*y]);
```

(%o16)



y después del sitema original:

```
(%i17) plotdf([y,-2*x+3*y]);
```

(%o17) 

Este tipo de sistemas se denominan *nodos inestables*.
Veamos ahora un ejemplo de sistema con dos valores propios negativos:

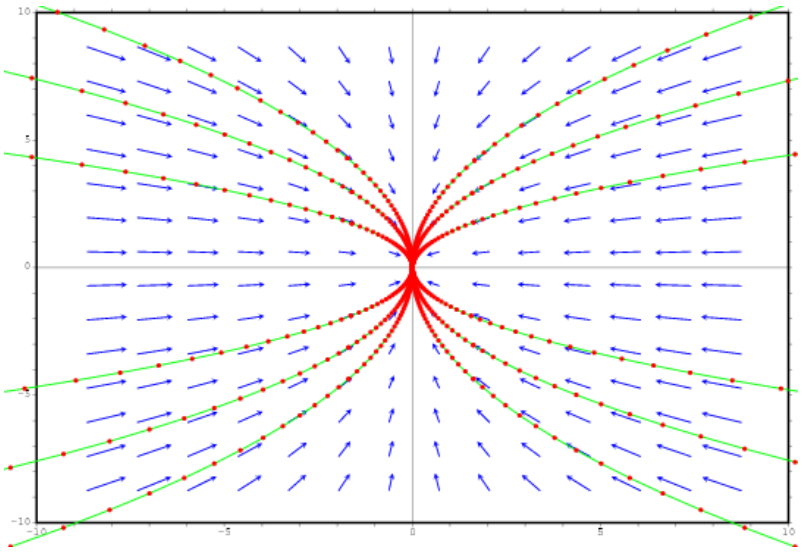
$$\begin{cases} \dot{x} = -y \\ \dot{y} = 2 * x - 3 * y \end{cases}$$

```
(%i19) eigenvalues(matrix([0,-1],[2,-3]));
```

(%o19) $[[-2,-1],[1,1]]$

Órbitas del sistema diagonalizado:

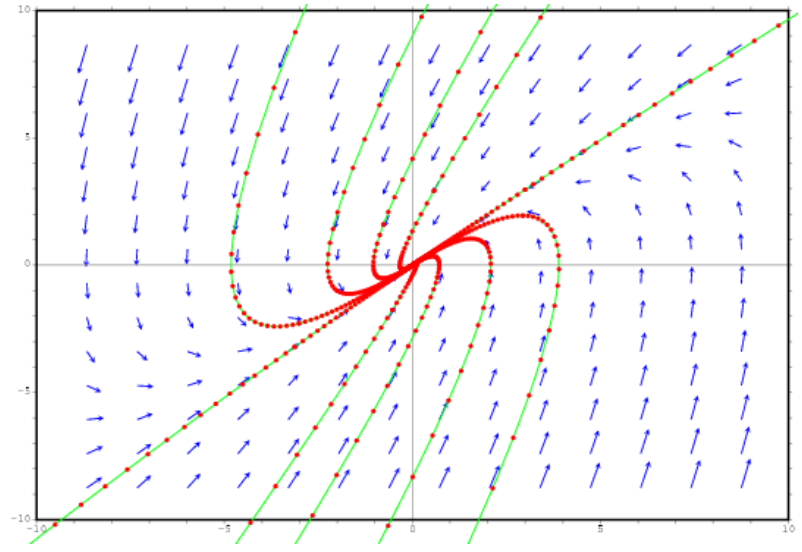
```
(%i20) plotdf([-2*x,-y]);
```



(%o20)

Y del sistema original:

```
(%i21) plotdf([-y,2*x-3*y]);
```



(%o21)

Estos sistemas se denominan *nodos estables*.

5.3. Caso $\mathbf{tr}(A)^2 = 4\mathbf{det}(A) > 0$

En este caso, los valores propios son reales e iguales $\lambda_1 = \lambda_2 = \lambda \in \mathbb{R}$. Como antes, se tendrá estabilidad asintótica si $\lambda < 0$ o bien inestabilidad si $\lambda > 0$. Hay dos subcasos posibles.

5.3.1. Caso diagonalizable

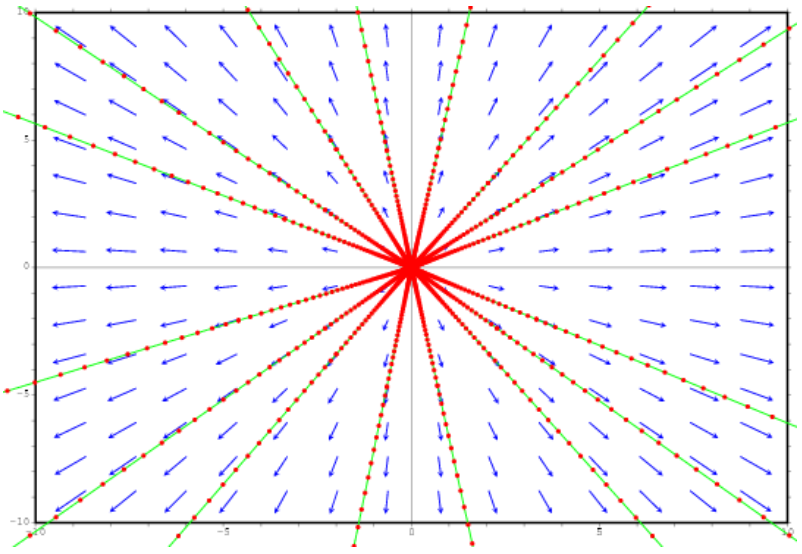
Si la matriz A sólo posee un valor propio doble λ, puede que sea diagonalizable o que no. Si es diagonalizable, existe una matriz P que la pasa a la forma

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

Las soluciones (triviales) al sistema que determina A son semirrectas que parten del origen (no contienen al origen, sin embargo, que es un punto singular). Debido a la representación gráfica de las órbitas (aquí en el caso $\lambda = 1$):

```
(%i22) plotdf([x,y]);
```

(%o22)



Estos sistemas se llaman *nodos estrella* (estables si $\lambda > 0$ e inestables si $\lambda < 0$). Un ejemplo de nodo estrella estable lo proporciona el sistema (ya en forma diagonal):

$$\begin{cases} \dot{x} = -x \\ \dot{y} = -y \end{cases}$$

5.3.2. Caso no diagonalizable

Puede ocurrir que A admita un valor propio doble pero que no sea diagonalizable. En ese caso, siempre existe una matriz de paso P que la convierte a la forma casi-diagonal

$$\begin{pmatrix} \lambda & 1 \\ 0 & \lambda \end{pmatrix} \tag{8}$$

En este caso, Maxima puede encontrar la solución explícita procediendo como hicimos antes:

```
(%i23) ec1:'diff(x(t),t)=%lambda*x(t)+y(t);
```

(%o23) $$\frac{d}{d\,t}\,x\,(t) = y\,(t) + \lambda\,x\,(t)$$

```
(%i24) ec2:'diff(y(t),t)=%lambda*y(t);
```

(%o24) $$\frac{d}{d\,t}\,y\,(t) = \lambda\,y\,(t)$$

```
(%i25) desolve([ec1,ec2],[x(t),y(t)]);
```

(%o25) $$[x\,(t) = y\,(0)\;t\,e^{\lambda\,t} + x\,(0)\;e^{\lambda\,t}, y\,(t) = y\,(0)\;e^{\lambda\,t}]$$

```
(%i26) solve(x=(y[0]*t+x[0])*exp(%lambda*t),t);
```

(%o26) $$[t = -\frac{e^{-\lambda\,t}\;\left(x_0\,e^{\lambda\,t} - x\right)}{y_0}]$$

```
(%i27) solve(y=y[0]*exp(%lambda*t),t);
```

(%o27) $$[t = \frac{log\left(\frac{y}{y_0}\right)}{\lambda}]$$

```
(%i28) solve(
log(y/y[0])=
-%lambda*(%e^(-%lambda*t)*(x[0]*%e^(%lambda*t)-x))/y[0],y
);
```

(%o28) $$[y = y_0\, e^{\frac{\lambda\, e^{-\lambda\, t}\, x}{y_0} - \frac{x_0\, \lambda}{y_0}}]$$

Los sistemas de este tipo se denominan *nodos impropios* (o *nodos de tangente simple*).
Un ejemplo de esta situación lo proporciona el sistema

$$\begin{cases} \dot{x} = y \\ \dot{y} = -x + 2y \end{cases}$$

Su matriz de coeficientes no es diagonalizable, pero se puede pasar a la forma canónica (8). Maxima nos lo muestra con los comandos `jordan` y `dispJordan`, del paquete `diag`:

```
(%i29) load(diag)$
(%i30) jordan(matrix([0,1],[-1,2]));
```

(%o30) $$[[1, 2]]$$

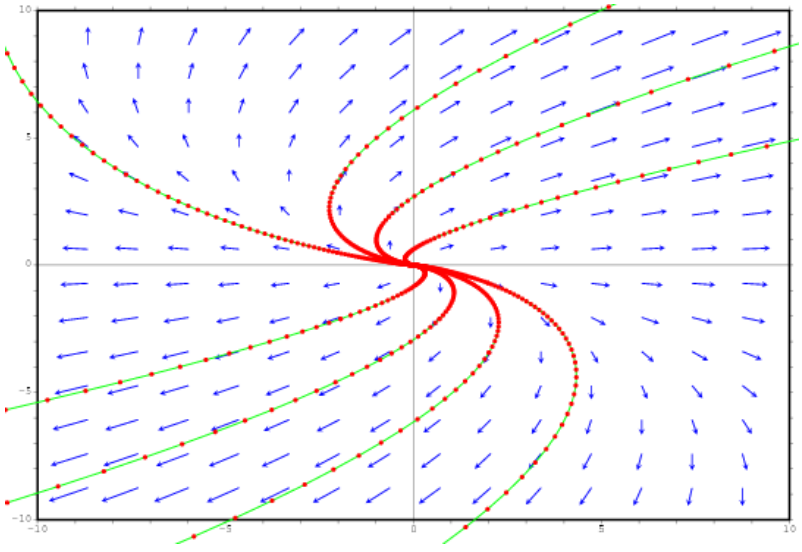
```
(%i31) dispJordan(%);
(%o31)
```

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

La representación de las órbitas del sistema en forma canónica es:

```
(%i32) plotdf([x+y,y]);
```

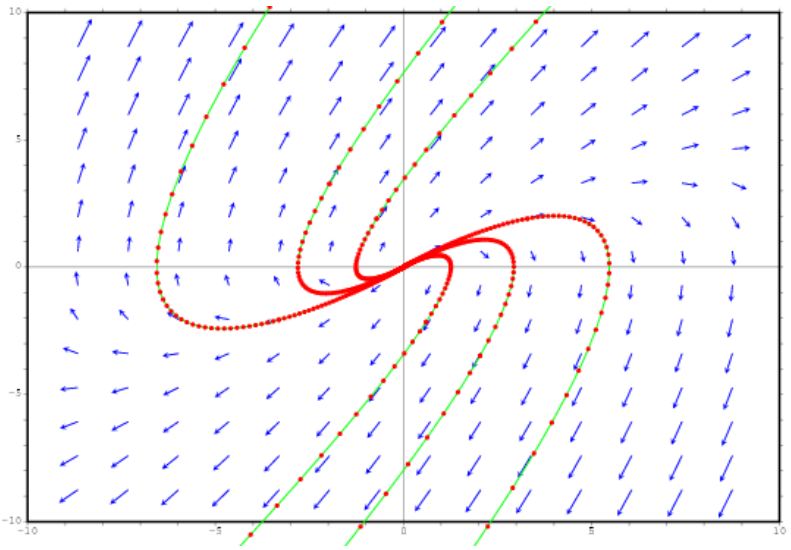
(%o32)



Y la del sistema en su forma original:

```
(%i33) plotdf([y,-x+2*y]);
```

(%o33)



Éste era un caso inestable. El caso estable ($\lambda < 0$) es idéntico, salvo porque las órbitas se dirigen al origen en lugar de alejarse.
Nota: No estamos haciendo mención de los casos degenerados, en los que unos de los valores propios es nulo. El lector puede estudiar fácilmente estos casos triviales, observando que conducen a órbitas situadas sobre los ejes coordenados en los sistemas canónicos.

5.4. Caso $\mathbf{tr}(A)^2 < 4\mathbf{det}(A)$, $\mathbf{det}(A) > 0$

En este caso, los valores propios son números complejos conjugados, $\lambda_1 = a + bi$ y $\lambda_2 = a - bi$ (con $b \neq 0$). De acuerdo con lo que ya sabemos, el sistema será asintóticamente estable si $a < 0$ e inestable si $a > 0$. Si $a = 0$ hay estabilidad, pero *no* asintótica.
Recurriendo a la forma compleja (o complejificación) de la matriz A de coeficientes del sistema, siempre se puede encontrar una matriz de paso P tal que A se transforma en

$$\begin{pmatrix} a & -b \\ b & a \end{pmatrix}$$

Maxima resuelve fácilmente el sistema de ecuaciones diferenciales asociado:

```
(%i34) eq1:'diff(x(t),t)=a*x(t)-b*y(t);
```

(%o34)
$$\frac{d}{d\,t}\,x\,(t) = a\,x\,(t) - b\,y\,(t)$$

```
(%i35) eq2:'diff(y(t),t)=b*x(t)+a*y(t);
```

(%o35)
$$\frac{d}{d\,t}\,y\,(t) = a\,y\,(t) + b\,x\,(t)$$

```
(%i36) assume(notequal(b,0));
```

(%o36)
$$[notequal\,(b,0)]$$

```
(%i37) radcan(desolve([eq1,eq2],[x(t),y(t)]));
(%o37)
```

$$[x\,(t) = x\,(0)\;e^{a\,t}\,cos\,(b\,t) - y\,(0)\;e^{a\,t}\,sin\,(b\,t)\,,$$
$$y\,(t) = x\,(0)\;e^{a\,t}\,sin\,(b\,t) + y\,(0)\;e^{a\,t}\,cos\,(b\,t)]$$

Aquí también habrá que distinguir casos.

5.4.1. Caso $a = 0$

Los valores propios son imaginarios puros $\pm bi$. Las soluciones entonces son

```
(%i38) subst(a=0,%);
```

(%o38)
$$[x\,(t) = x\,(0)\;cos\,(b\,t) - y\,(0)\;sin\,(b\,t)\,, y\,(t) = x\,(0)\;sin\,(b\,t) + y\,(0)\;cos\,(b\,t)]$$

La representación gráfica de las órbitas típicamente proporciona elipses. Este tipo de sistemas se denominan *centros*.
Por ejemplo, consideremos el sistema

$$\begin{cases} \dot{x} = x + 2y \\ \dot{y} = -x - y \end{cases}$$

Calculamos los valores propios:

```
(%i39) eigenvalues(matrix([1,2],[-1,-1]));
```

(%o39)
$$[[-i, i], [1, 1]]$$

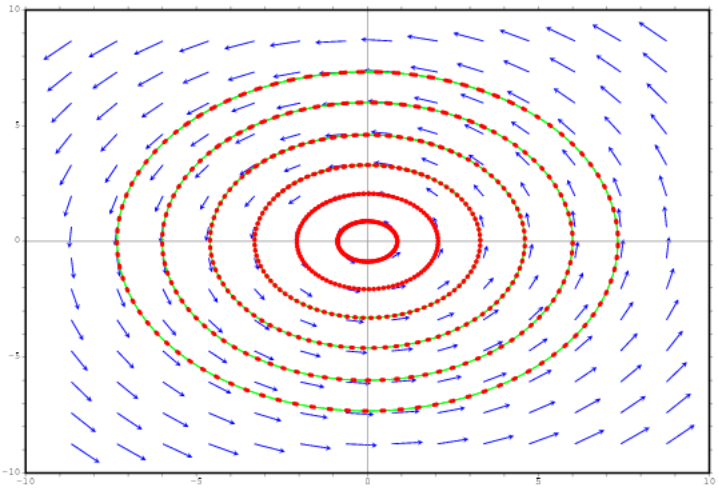
Por tanto, $b = 1$. Las soluciones del sistema vienen dadas por

```
(%i39) subst(b=1,%o38);
```

(%o9) $[x(t) = x(0)\ cos(t) - y(0)\ sin(t), y(t) = x(0)\ sin(t) + y(0)\ cos(t)]$

Ahora, podemos obtener la representación gráfica de las órbitas en el plano de fases. Primero, las del sistema en forma canónica:

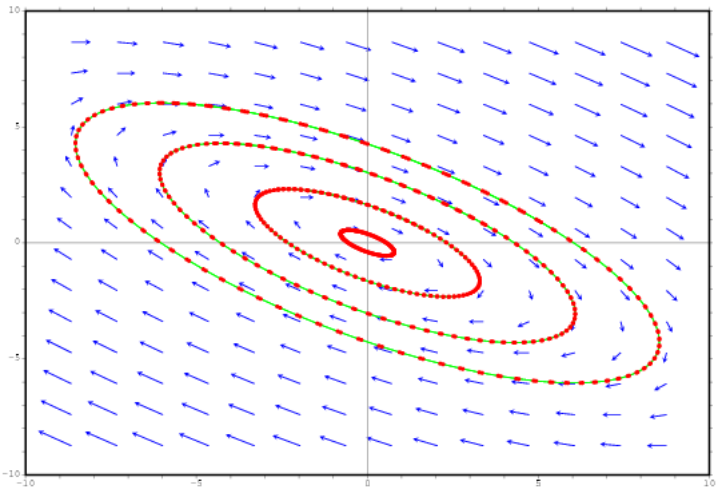
```
(%i40) plotdf([-y,x]);
```



(%o40)

Y las del sistema original:

```
(%i41) plotdf([x+2*y,-x-y]);
```



(%o41)

Gráficamente se aprecia con toda claridad que el sistema es estable pero no asintóticamente estable.

5.4.2. Caso $a \neq 0$

Ahora, los valores propios son $a \pm bi$. Ambos casos son iguales, salvo un cambio de signo y el correspondiente cambio en la orientación de las órbitas (que siguen siendo las mismas como conjuntos de puntos).
Un ejemplo en el que la parte real es $a > 0$ está dado por el sistema

$$\begin{cases} \dot{x} = 2x + 2y \\ \dot{y} = -x \end{cases}$$

```
(%i42) eigenvalues(matrix([2,2],[-1,0]));
```

(%o42) $$[[1 - i, i + 1], [1, 1]]$$

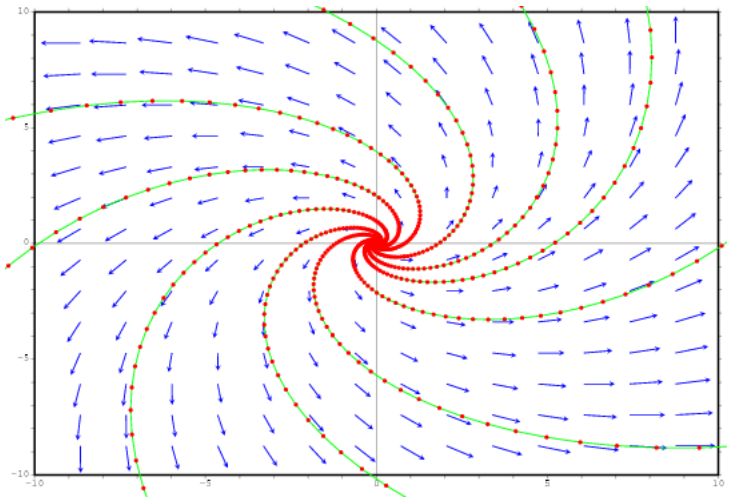
Así, $a \pm bi = 1 \pm i$. Las órbitas del sistema canónico, cuya matriz es

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

se representan en el siguiente gráfico:

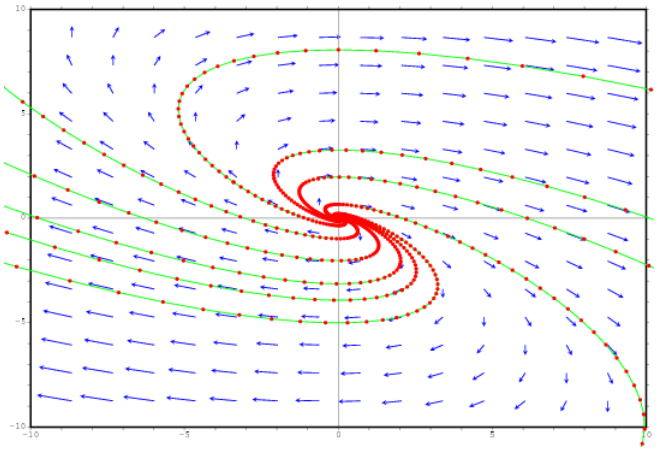
```
(%i43) plotdf([x-y,x+y]);
```

(%o43)



Por razones obvias, un sistema de este tipo se denomina un *foco o espiral inestable*. Las órbitas del sistema original son:

```
(%i44) plotdf([2*x+2*y,-x]);
```



(%o44)

Por último, damos un ejemplo de sistema donde la parte real de los valores propios es negativa (con lo cual el sistema es asintóticamente estable):

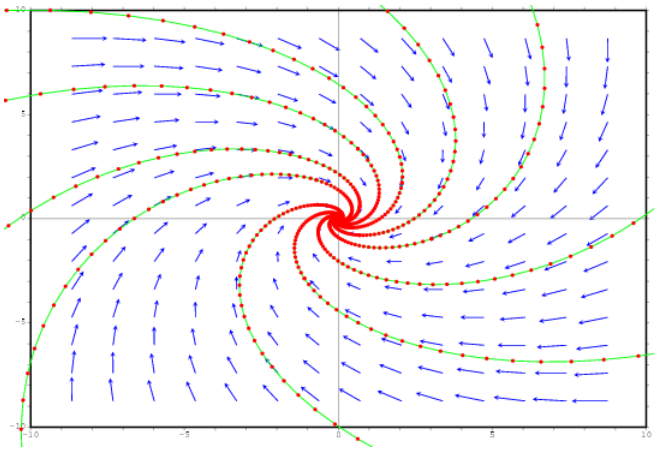
$$\begin{cases} \dot{x} = -2x - 2y \\ \dot{y} = x \end{cases}$$

```
(%i45) eigenvalues(matrix([-2,-2],[1,0]));
```

(%o45) $$[[-i-1, i-1], [1, 1]]$$

órbitas del sistema en forma canónica:

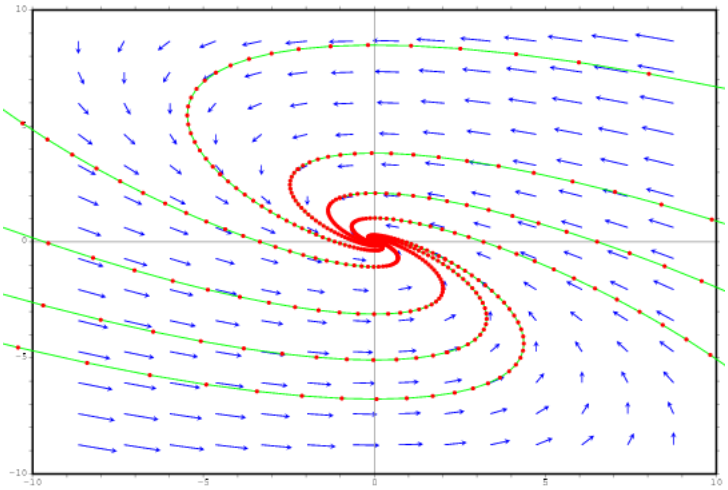
```
(%i46) plotdf([-x+y,-x-y]);
```



(%o46)

Y del sistema original:

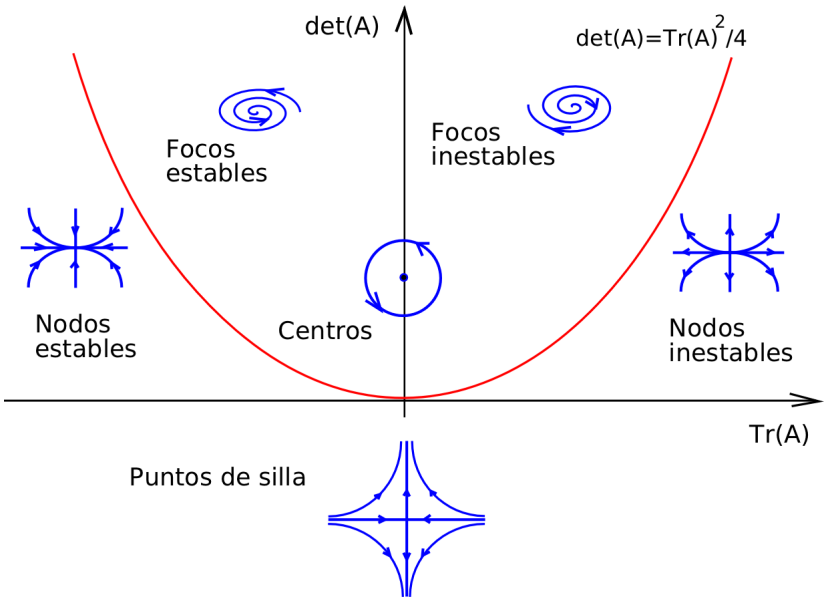
```
(%i47) plotdf([-2*x-2*y,x]);
```



(%o47)

Este tipo de sistemas, se conoce con el nombre de *focos o espirales estables*.

Como resumen de todo lo que hemos visto hasta ahora, tenemos el siguiente diagrama:



6. Análisis de sistemas no lineales

6.1. El método directo de Lyapunov

Para concretar ideas, consideremos un sistema descrito por las siguientes ecuaciones en el plano:

$$\begin{cases} \dot{x} = f(x,y) \\ \dot{y} = g(x,y) \end{cases}$$

Si (x^*, y^*) es un punto de equilibrio ocurre que $f(x^*, y^*) = 0 = g(x^*, y^*)$. Vamos a estudiar al estabilidad de la solución constante $(x(t), y(t)) = (x^*, y^*)$, pero bajo unas hipótesis particulares:

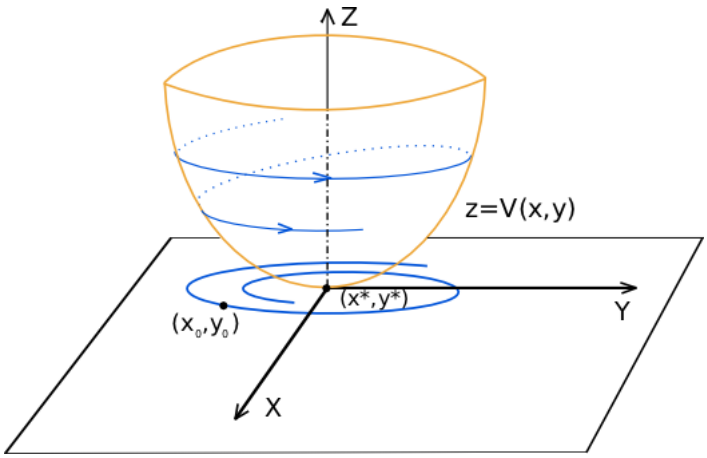
1. Supondremos que existe una función diferenciable $V(x,y)$ definida en un entorno de (x^*, y^*) que toma valores positivos, de manera que su gráfica queda en el semiespacio por encima del plano.

2. Supondremos también que la superficie $z = V(x,y)$ sólo posee un mínimo y que éste se encuentra precisamente en el punto (x^*, y^*).

3. Por último, supondremos que la función $V(x,y)$ sobre las trayectorias del sistema diferencial es decreciente, es decir: si $z(t) = V(x(t), y(t))$,

$$\frac{dz}{dt} \leq 0.$$

Gráficamente, la situación es como en la siguiente figura.



La última de las condiciones anteriores es la más importante. Significa que si recorremos una solución $(x(t), y(t))$ del sistema que para $t = t_0$ pasa por (x_0, y_0), entonces debemos ir "descendiendo" por la superficie. Por la forma de $V(x,y)$ ocurre que $(x(t), y(t))$ debe aproximarse a (x^*, y^*) y esto, precisamente, es la condición de estabilidad asintótica de la solución de equlibrio en (x^*, y^*).

Nota: Aún cuando esta motivación se ha hecho en un contexto bidimensional, el método de Lyapunov es aplicable (con las modificaciones obvias) a sistemas de n ecuaciones.
La principal ventaja del método de Lyapunov es que, cuando se puede encontrar la función V asegura la estabilidad asintótica, además de que es aplicable a sistemas no necesariamente lineales ni autónomos.
A una función como la $V(x, y)$ del enunciado se la denomina *función de Lyapunov*. No existen reglas generales para obtener funciones de Lyapunov en un problema dado, pero una buena manera de enfocar estos problemas siempre es tratar de encontrar una V que sea una forma cuadrática de las variables del problema (en este caso, x e y). Desde el punto de vista físico, frecuentemente se interpreta a V como una cierta "energía potencial" del sistema, por lo que también es frecuente denotarla por E.

Consideremos, por ejemplo, la (única) solución de equilibrio $x^* = (0, 0)$ del sistema:

$$\begin{cases} \dot{x} = -2xy \\ \dot{y} = x^2 - y^3 \end{cases}$$

Busquemos, de acuerdo con lo dicho, una función en la forma

$$E(x, y) = \alpha x^{2p} + \beta y^{2q}.$$

Fijémonos en que tendremos que calcular $E(x(t), y(t))$, con lo que nos aparecerán (al aplicar la regla de la cadena) $\dot{x} = -2xy$, $\dot{y} = x^2 - y^3$; así, declaramos las dependencias con t y hacemos:

```
(%i1) depends([x,y],[t]);
```

$$(\%o1) \qquad [x(t), y(t)]$$

```
(%i2) gradef(x(t),-2*x(t)*y(t));
```

$$(\%o2) \qquad x(t)$$

```
(%i3) gradef(y(t),x(t)^2-y(t)^3);
```

$$(\%o3) \qquad y(t)$$

```
(%i4) E(x,y):=%alpha*(x(t))^(2*p)+%beta*(y(t))^(2*q);
```

$$(\%o4) \qquad E(x, y) := \alpha\, x(t)^{2\,p} + \beta\, y(t)^{2\,q}$$

Ahora, podemos calcular la derivada implícita $\frac{dE}{dt}$:

```
(%i5) diff(E(x,y),t);
```

$$(\%o5) \qquad 2\,\beta\,q\,y\,(t)^{2\,q-1}\left(x\,(t)^2-y\,(t)^3\right)-4\,\alpha\,p\,x\,(t)^{2\,p}\,y\,(t)$$

```
(%i6) expand(%);
```

$$(\%o6) \qquad -2\,\beta\,q\,y\,(t)^{2\,q+2}+2\,\beta\,q\,x\,(t)^2\,y\,(t)^{2\,q-1}-4\,\alpha\,p\,x\,(t)^{2\,p}\,y\,(t)$$

Queremos que sea $\frac{dE}{dt} \leq 0$. Como el término $-2\,\beta\,q\,y\,(t)^{2\,q+2}$ es claramente definido negativo si $\beta\,q \geq 0$, debemos preocuparnos por anular el resto, es decir, debemos tomar unos valores de los parámetros que hagan cero la expresión

```
(%i7) a(p,q,%alpha,%beta):=
+2*%beta*q*x(t)^2*y(t)^(2*q-1)-4*%alpha*p*x(t)^(2*p)*y(t);
```

$$(\%o7) \qquad a\,(p,q,\alpha,\beta):=2\,\beta\,q\,x\,(t)^2\,y\,(t)^{2\,q-1}-4\,\alpha\,p\,x\,(t)^{2\,p}\,y\,(t)$$

Teniendo en cuenta al condición $\beta\,q \geq 0$, por simple inspección vemos que la elección $p=1=q$, $\alpha=1$, $\beta=2$ es acertada (puede haber otras):

```
(%i8) a(1,1,1,2);
```

$$(\%o8) \qquad 0$$

Sustituyendo en $E(x,y)$, obtenemos la función de Lyapunov buscada:

```
(%i9) V(x,y)=subst([p=1,q=1,%alpha=1,%beta=2],E(x,y));
```

$$(\%o9) \qquad V\,(x,y)=2\,y\,(t)^2+x\,(t)^2$$

```
(%i10) 'diff(V(x,y),t)=
expand(subst([p=1,q=1,%alpha=1,%beta=2],diff(E(x,y),t)));
```

$$(\%o10) \qquad \frac{d}{d\,t}\,V\,(x,y)=-4\,y\,(t)^4$$

Obviamente, hay ocasiones en las que es preciso aplicar más de un método. Como muestra, estudiemos el sistema

$$\begin{cases}\dot{x}=(1-\alpha x^2-\beta y^2)(2y-x)\\ \dot{y}=-(1-\alpha x^2-\beta y^2)(x+y)\end{cases}$$

tratando de determinar la estabilidad asintótica de la solución nula y la existencia de valores particulares de α y β que hagan que *todas* las soluciones sean asintóticamente estables.
Comenzamos observando que para $\alpha, \beta = 0$ el sistema es lineal:

$$\begin{cases} \dot{x} = -x + 2y \\ \dot{y} = -x - y \end{cases}$$

con polinomio mínimo dado por:

```
(%i11) A:matrix([-1,2],[-1,-1]);
```

(%o11)
$$\begin{pmatrix} -1 & 2 \\ -1 & -1 \end{pmatrix}$$

```
(%i12) load(diag)$
(%i13) minimalPoly(jordan(A));
```

(%o13)
$$\left(x - \sqrt{2}\,i + 1\right)\left(x + \sqrt{2}\,i + 1\right)$$

Vemos que las raíces son $\lambda_1 = -1 + \sqrt{2}i$, $\lambda_2 = -1 - \sqrt{2}i$, ambas con parte real estrictamente negativa, por lo que *todas* las soluciones son asintóticamente estables en el caso $\alpha, \beta = 0$.
Supongamos ahora que $\alpha, \beta \neq 0$. Para estudiar la etabilidad asintótica del equilibrio en $(0, 0)$ buscamos una función de Lyapunov, por el procedimiento que ya hemos descrito:

```
(%i14) depends([x,y],[t]);
```

(%o14)
$$[x(t), y(t)]$$

```
(%i15) gradef(x(t),(1-%alpha*x^2-%beta*y^2)*(2*y(t)-x(t)));
```

(%o15)
$$x(t)$$

```
(%i16) gradef(y(t),-(1-%alpha*x^2-%beta*y^2)*(x(t)+y(t)));
```

(%o16)
$$y(t)$$

```
(%i17) E(x,y):=a*(x(t))^(2*p)+b*(y(t))^(2*q);
```

(%o17) $$E\left(x,y\right):=a\,x\left(t\right)^{2\,p}+b\,y\left(t\right)^{2\,q}$$

```
(%i18) diff(E(x,y),t);
(%o18)
```

$$\begin{aligned}&2\,b\,q\,y\left(t\right)^{2\,q-1}\,\left(y\left(t\right)+x\left(t\right)\right)\,\left(\beta\,y^2+\alpha\,x^2-1\right)\\&+2\,a\,p\,x\left(t\right)^{2\,p-1}\,\left(2\,y\left(t\right)-x\left(t\right)\right)\,\left(-\beta\,y^2-\alpha\,x^2+1\right)\end{aligned}$$

Llegados a este punto, lo primero que se nos ocurre es tomar $p=1=q$, para que los términos de la forma xy^{2q-1} y $x^{2p-1}y$ se hagan homogéneos:

```
(%i19) subst([p=1,q=1],%o18);
(%o19)
```

$$\begin{aligned}&2\,b\,y\left(t\right)\,\left(y\left(t\right)+x\left(t\right)\right)\,\left(\beta\,y^2+\alpha\,x^2-1\right)\\&+2\,a\,x\left(t\right)\,\left(2\,y\left(t\right)-x\left(t\right)\right)\,\left(-\beta\,y^2-\alpha\,x^2+1\right)\end{aligned}$$

Lo siguiente es tratar de cancelar los términos que contienen el producto cruzado xy, es decir, intentar hacer

$$2byx(\beta y^2+\alpha x-1)-4axy(\beta y^2+\alpha x-1)=0.$$

Claramente, esto se consigue tomando $b=2a$. Con esta suposición, llegamos a:

```
(%i18) factor(subst([b=2*a],%));
```

(%o18) $$2\,a\,\left(2\,y\left(t\right)^2+x\left(t\right)^2\right)\,\left(\beta\,y^2+\alpha\,x^2-1\right)$$

Una región en la que es fácil encontrar una función de Lyapunov es aquella en la cual $\beta y^2+\alpha x^2-1\leq 0$, pues para ello basta con tomar $\alpha,\beta\leq 0$. En ese caso, con cualquier valor $a\geq 0$ (y el correspondiente $b=2a$) llegamos a una función de Lyapunov; por ejemplo, haciendo $b=2a=2$ y llamando $\alpha=-A$, $\beta=-B$, con $A,B\geq 0$:

```
(%i19) V(x,y)=subst([b=2,a=1,p=1,q=1],E(x,y));
```

(%o19) $$V\left(x,y\right)=2\,y\left(t\right)^2+x\left(t\right)^2$$

```
(%i23) 'diff(V,z)=factor(subst([b=2,a=1,%beta=-B,%alpha=-A],%o18));
```

(%o23)
$$\frac{d}{d\,z}\,V = -2\left(2\,y\,(t)^2 + x\,(t)^2\right)\left(y^2\,B + x^2\,A + 1\right)$$

Es obvio entonces que $V(x,y) \geq 0$ y que $\frac{dV}{dt} \leq 0$, por lo que en la región dada por $\alpha, \beta \leq 0$, la solución nula es asintóticamente estable. El lector encontrará instructivo tratar de averiguar qué ocurre si $\alpha > 0$ ó $\beta > 0$ (quizás leyendo las secciones siguientes).

6.2. La técnica de linealización

La habitual fuerza elástica que actúa sobre un péndulo viene dada por $F_e = -k \sin\theta$ (con $k > 0$) donde θ es el desplazamiento angular respecto de la posición de equilibrio. Un péndulo amortiguado está sujeto a una fuerza adicional que es proporcional a la velocidad y que tiende a frenar el movimiento, $F_a = -a\dot{\theta}$ (con $a > 0$). De hecho, en esta notación es $k = g/l$, de modo que aproximando $g = 10ms^{-2}$ y $l = 1m$, podemos tomar $k = 10s^{-2}$. Supondremos también que la masa del péndulo vale 1 y que ajustamos la escala de tiempo de manera que $k = 1$. En estas condiciones, la ecuación de Newton para el péndulo es simplemente

$$\ddot{\theta} + a\dot{\theta} + \sin\theta = 0,$$

que puede escribirse, si hacemos $\theta = x$, $\dot{\theta} = y$, como

$$\left\{ \begin{array}{rcl} \dot{x} & = & y \\ \dot{y} & = & -ay - \sin x \end{array} \right. . \tag{9}$$

Éste es claramente un sistema no lineal, dependiente de un parámetro a que es el amortiguamiento. La técnica estándar para tratar tales sistemas consiste en estudiar el movimiento alrededor de una posición de equilibrio viéndolo como una perturbación de la solución correspondiente. Un punto (x_e, y_e) es una posición de equilibrio para el sistema

$$\left\{ \begin{array}{rcl} \dot{x} & = & f(x,y) \\ \dot{y} & = & g(x,y) \end{array} \right. \tag{10}$$

si se cumple $f(x_e, y_e) = 0 = g(x_e, y_e))$. Se considera entonces un sistema lineal asociado, obtenido desarrollando por Taylor f y g alrededor de (x_e, y_e):

$$\left\{ \begin{array}{rcl} \dot{x} & = & D_1 f(x_e,y_e)(x - x_e) + D_2 f(x_e,y_e)(y - y_e) \\ \dot{y} & = & D_1 g(x_e,y_e)(x - x_e) + D_2 g(x_e,y_e)(y - y_e) \end{array} \right. .$$

Fijémonos en que haciendo un cambio de coordenadas

$$x - x_e \mapsto x, y - y_e \mapsto y$$

este sistema es

$$\left(\begin{array}{c} \dot{x} \\ \dot{y} \end{array} \right) = J_{(x_e,y_e)} F \left(\begin{array}{c} x \\ y \end{array} \right) \tag{11}$$

donde $F(x,y) = (f(x,y), g(x,y))$ y $J_{(x_e,y_e)}F$ es su Jacobiano en el punto de equilibrio (x_e, y_e). La relación entre el sistema linealizado (11) y el original (10) puede describirse como sigue. El punto (x_e, y_e) se dice que es hiperbólico si todos los valores propios de $J_{(x_e,y_e)}F$ tienen parte real distinta de cero. Entonces, se tiene:

Teorema (Grossmann-Hartman)**.** *Si (x_e, y_e) es un punto hiperbólico del sistema no lineal (10), el comportamiento de éste en un entorno de (x_e, y_e) es topológicamente equivalente al comportamiento del sistema linealizado (11) en un entorno del origen* $(0,0)$.

Esto quiere decir que existe un homeomorfismo que lleva las órbitas del sistema (10) alrededor del punto (x_e, y_e) en órbitas del sistema (11) alrededor del $(0,0)$ y, por tanto, el comportamiento cualitativo de los dos sistemas es el mismo. Así pues, esta técnica reduce el estudio de sistemas no lineales al de sistemas lineales en un entorno del origen. El comportamiento de estos últimos ya lo hemos analizado en una sección anterior, por lo que estamos en condiciones de poder decir algo sobre los sistemas no lineales, como veremos con detalle en la siguiente subsección.

6.3. El péndulo amortiguado

Vamos a aplicar estas ideas al péndulo amortiguado (9). En este caso, tenemos que $f(x,y) = y$, $g(x,y) = ay + \sin x$. Las posiciones de equilibrio vienen dadas por las soluciones a

$$\begin{cases} y = 0 \\ -ay - \sin x = 0 \end{cases},$$

y vamos a considerar, por simplicidad, el comportamiento de este sistema no lineal alrededor del equilibrio $(x_e, y_e) = (0,0)$. El primer paso para poder aplicar el Teorema de Grossmann-Hartman es comprobar que tenemos un punto hiperbólico, y para ello Maxima dispone del comando `jacobian`, que nos devuelve la matriz Jacobiana de una función vectorial como la $F(x,y) = (y, -ay - \sin(x))$:

```
(%i1) jacobian([y,-a*y-*sin(x)],[x,y]);
(%o1)
```

0 errors, 0 warnings

$$\begin{pmatrix} 0 & 1 \\ -cos\,(x) & -a \end{pmatrix}$$

```
(%i2) A:subst([x=0,y=0],%);
(%o2)
```

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

Calculamos los valores propios:

```
(%i3) eigenvalues(A);
```

(%o3) $$[[-\frac{\sqrt{a^2-4}+a}{2},\frac{\sqrt{a^2-4}-a}{2}],[1,1]]$$

Claramente, el comportamiento del sistema depende de si $a < 2$ ó $a > 2$. Estudiemos el primer caso con el valor particular $a = 1$ (correspondiente a un amortiguamiento débil):

```
(%i4) A1:matrix([0,1],[-1,-1]);
(%o4)
```

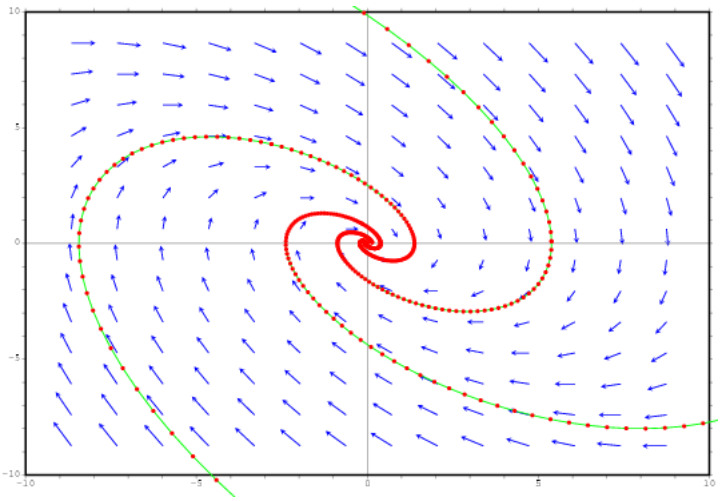
$$\begin{pmatrix} 0 & 1 \\ -1 & -1 \end{pmatrix}$$

```
(%i5) eigenvalues(A1);
```

(%o5) $$[[-\frac{\sqrt{3}\,i+1}{2},\frac{\sqrt{3}\,i-1}{2}],[1,1]]$$

En este caso, los valores propios tienen parte real negativa, de modo que las soluciones se van desvaneciendo (aproximándose al valor constante 0) a medida que oscilan. El punto de equilibrio es, pues, un atractor. Podemos verlo gráficamente, para ello, necesitamos cargar primero el paquete `plotdf`:

```
(%i6) load(plotdf)$
(%i7) plotdf([y,-x-y]);
```



(%o7)

Veamos qué ocurre si $a > 2$ (en este caso se dice que el sistema está sobreamortiguado). Ahora tenemos la matriz

```
(%i8) A3:matrix([0,1],[-1,-3]);
(%o8)
```

$$\begin{pmatrix} 0 & 1 \\ -1 & -3 \end{pmatrix}$$

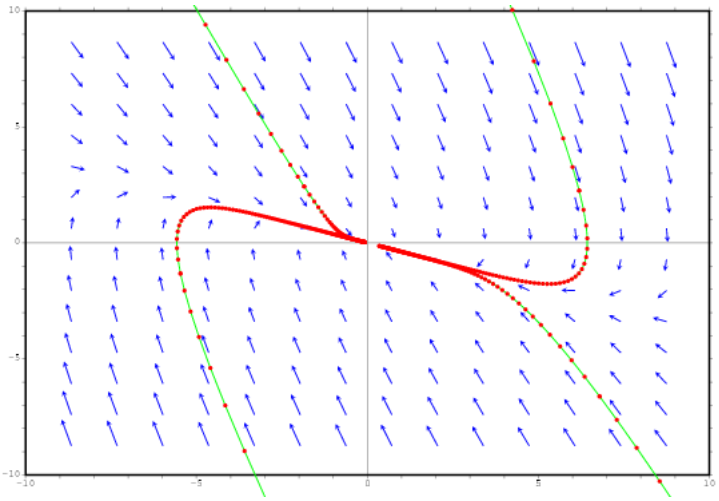
```
(%i9) eigenvalues(A3);
```

$$(\%o9) \qquad [[-\frac{\sqrt{5}+3}{2}, \frac{\sqrt{5}-3}{2}], [1,1]]$$

De nuevo los valores propios tienen parte real negativa, lo cual significa que el punto de equilibrio es un atractor. Pero en este caso no hay oscilaciones, sino que se alcanza el equilibrio de manera casi instantánea, esto se ve en el diagrama de fases porque las trayectorias "no dan vueltas" alrededor del origen, sino que van directas hacia él.

```
(%i10) plotdf([y,-x-3*y]);
```

(%o10)



Para apreciar la complejidad de este sistema, vamos a analizar lo que ocurre en otro punto de equilibrio del sistema original. Esta vez, consideramos el $(\pi, 0)$.

```
(%i11) jacobian([y,-a*y-sin(x)],[x,y]);
(%o11)
```

0 errors, 0 warnings

$$\begin{pmatrix} 0 & 1 \\ -cos\,(x) & -a \end{pmatrix}$$

```
(%i12) B:subst([x=%pi,y=0],%);
(%o12)
```

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

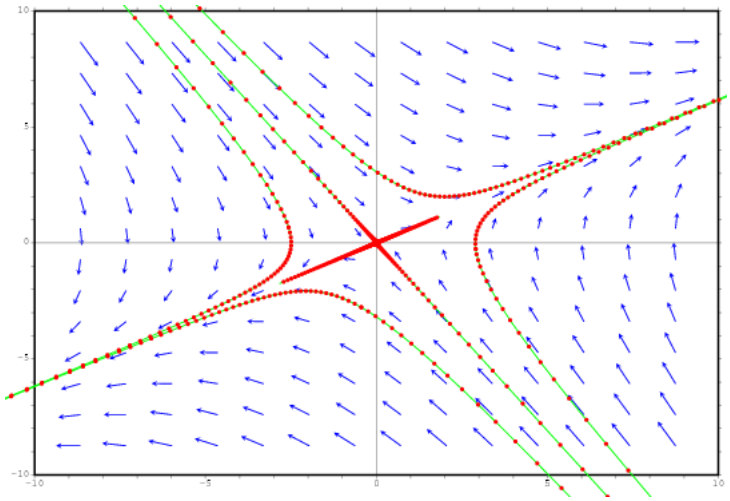
```
(%i13) eigenvalues(B);
```

$$(\%o13) \qquad [[-\frac{\sqrt{a^2+4}+a}{2}, \frac{\sqrt{a^2+4}-a}{2}], [1,1]]$$

Aquí los valores propios son reales pero de signo contrario. Por tanto, el punto de equilibrio $(\pi, 0)$ es un punto de silla: todas las soluciones excepto las separatrices se alejan del equilibrio:

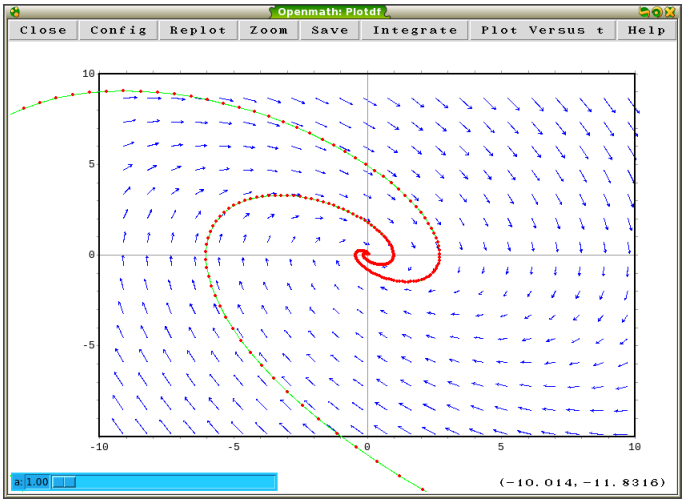
```
(%i14) plotdf([y,x-y]);
```

(%o14)

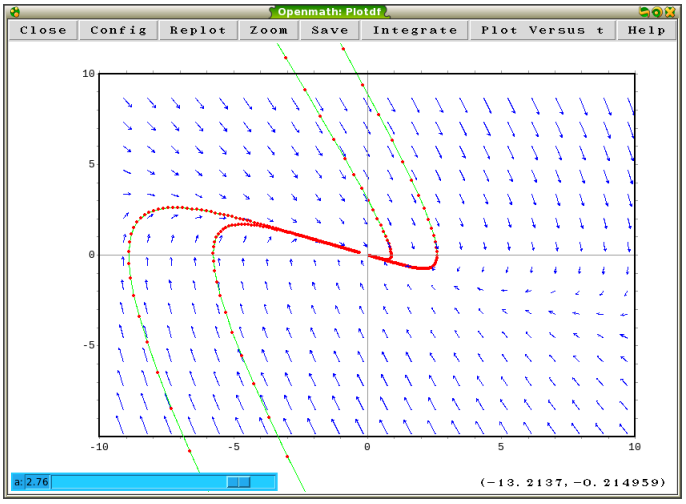


Podemos hacer esto de manera mucho más eficiente aprovechando las capacidades gráficas de Maxima y GNUplot, introduciendo un deslizador (slider) que modifique los valores del parámetro a; de esta forma, se visualiza mucho más claramente la dependencia del comportamiento con los parámetros:

```
(%i15) plotdf([y,-x-a*y],[x,y],[parameters,"a"],
[sliders,"a=1:3"]);
```


Openmath: Plotdf
Close
Config
Replot
Zoom
Save
Integrate
Plot Versus t
Help
10
5
0
-5
-10
-5
0
5
10
a: 1.00
(−10.014, −11.8316)

(%o15)


Openmath: Plotdf
Close
Config
Replot
Zoom
Save
Integrate
Plot Versus t
Help
10
5
0
-5
-10
-5
0
5
10
a: 2.76
(−13.2137, −0.214959)

(%o15)

7. Métodos numéricos para problemas de valores iniciales

7.1. Nociones básicas sobre los métodos numéricos

Consideremos el problema de valores iniciales

$$\begin{cases} \dot{x} = f(t,x),\ t > t_0 \\ x(t_0) = x_0 \end{cases} \tag{12}$$

Utilizaremos la notación empleada en Física, donde un punto sobre una función indica derivación respecto del tiempo, porque resulta útil pensar en esta ecuación como en la descripción del movimiento de una partícula a través de su posición $x(t)$ si conocemos su velocidad $\dot{x}(t)$. También supondremos que conocemos la posición en el instante inicial, esto es, $x(t_0) = x_0$. Sin pérdida de generalidad, cuando no se especifique lo contrario supondremos $t_0 = 0$.
La idea es calcular un valor aproximado de $x(t)$ en una serie de instantes $t_0, t_1, ..., t_n$ y aproximar la solución $x = x(t)$ por una función interpolante entre estos nodos aproximados. Si queremos calcular la solución en un intervalo $[t_0, T]$, podemos dividirlo en n subintervalos uniformes haciendo

$$T = t_0 + nh$$

donde h es la longitud de cada subintervalo, $h = \frac{T-t_0}{n}$. La tarea consiste entonces en calcular los valores

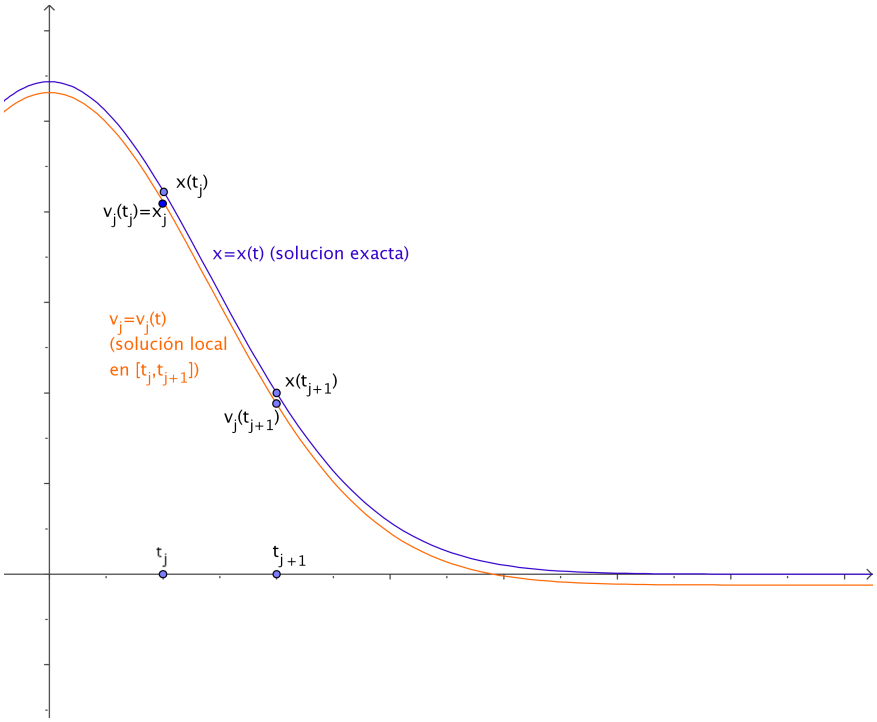
$$x_j \equiv x(t_j) = x(t_0 + jh).$$

con $j = 1, ..., n$. Esto lo haremos partiendo del dato inicial $x_0 = x(t_0)$ (que es exacto), mediante la solución aproximada de una secuencia de problemas *locales* de valores iniciales definidos en cada intervalo $[t_j, t_{j+1}]$:

$$\begin{cases} \dot{v}_j = f(t,v_j),\ t > t_j \\ v_j(t_j) = x_j \end{cases} \tag{13}$$

Cada uno de estos problemas locales determina una solución aproximada $v_j(t)$ que pasa por x_j en el instante t_j y que puede seguirse hasta el instante t_{j+1},

donde toma el valor $v_j(t_{j+1})$.



Con el fin de calcular una aproximación x_j, para cada $j = 1, ..., n$, del problema *global* (12) basta con resolver aproximadamente cada uno de los problemas locales (13), mediante algún algoritmo o fórmula que nos proporcione como salida una aproximación del valor $v_j(t_{j+1})$, x_{j+1}, a partir del valor inicial local x_j (o a partir de algunos valores previos x_k con $k \leq j$).

Hay que hacer notar que los problemas locales (13) *no son independientes*: la solución de cada uno de ellos depende de la solución hallada para los anteriores, a través de la condición inicial x_j, y todos dependen del dato inicial global x_0 y de la fórmula o algoritmo empleado.

Un método numérico para un problema de valores iniciales consiste en una fórmula que asigna un valor a x_{j+1} a partir de x_j, o de varios x_k con $k \leq j$, de tal manera que estos valores x_{j+1} aproximen la solución exacta del problema aproximado (13), es decir, $x_{j+1} \approx v_j(t_{j+1})$.
Si la fórmula es de la forma

$$x_{j+1} = F(x_j, x_{j+1}, ..., x_{j-l}) \tag{14}$$

se dice que el método es *explícito*. Si la fórmula no es explícita, es decir, es del tipo

$$G(x_{j+1}, x_j, ..., x_{j-l}) = 0, \tag{15}$$

el método se dice *implícito*. En este último caso, frecuentemente necesitaremos de otra aproximación (un algoritmo aproximado) para recuperar x_{j+1}. Las

ecuaciones (14), (15) se denominan ecuaciones en diferencias y en el siguiente apartado veremos cómo Maxima puede resolverlas de forma exacta en muchos casos.
Un método se dice *unipaso* si x_{j+1} sólo depende de x_j y *multipaso* en otro caso (de paso k si x_{j+1} depende de k valores anteriores).
Dado que un método numérico involucra dos aproximaciones (primero, la del problema global por una serie de problemas locales; segundo, la de la solución exacta de estos problemas locales por una aproximada), tendremos dos tipos de error. El *error local* de un método numérico en t_{j+1} se define por:

$$\varepsilon_L(t_{j+1}) = v_j(t_{j+1}) - x_{j+1}.$$

Definimos el *error global* de un método numérico en t_{j+1} como

$$\varepsilon_G(t_{j+1}) = x(t_{j+1}) - x_{j+1}.$$

Fijémonos en que, por no ser los problemas locales independientes, el error global *no* es la suma de los errores locales.

Un método se dice que es *consistente* si para cada j, $\varepsilon_L(t_{j+1})$ tiende a 0 cuando $h \to 0$. Se dice que es *convergente* en T si $\varepsilon_G(t_{j+1})$ tiende a 0 cuando $h \to 0$. La consistencia de un método está relacionada con su precisión; usualmente, la consistencia se prueba encontrando un $p > 0$ tal que $O(\varepsilon_L) = O(h^p)$. Este p es el llamado *orden local* del método y lo normal es que el método se construya exigiendo que el desarrollo de Taylor de la solución aproximada que proporciona coincida con el desarrollo de Taylor hasta orden p de v_j en cada t_j. Bajo condiciones muy generales (por ejemplo, que la función $f(x,t)$ sea Lipschitz), se puede probar que la convergencia implica la consistencia. Lo difícil es saber qué condiciones deben darse para que la consistencia implique la convergencia; esta condición adicional es la estabilidad, de modo que podemos establecer que

$$\textit{consistencia} \;+\; \textit{estabilidad} \;\Rightarrow\; \textit{convergencia}.$$

El *orden global* sólo puede definirse cuando el método converge. En la mayoría de los casos, sin embargo, cuando se da la convergencia se tiene que el orden global es igual a $p-1$.

7.2. El método de Euler

Para ilustrar la forma en que se construyen los métodos numéricos de solución de ecuaciones diferenciales, vamos a considerar el caso más sencillo: una ecuación de primer orden

$$\frac{dx}{dt} \equiv \dot{x} = f(x,t).$$

Para obtener los valores $x(t_0), ..., x(t_n)$ se parte del valor inicial y de una estimación del valor medio de la derivada en cada intervalo $[t_i, t_{i+1}]$. Para ello, se utiliza $f(x,t)$, que da el valor de la derivada en el instante t.
El método de Euler proporciona una relación de recurrencia entre los $x(t_n)$ a

partir de la función $f(x,t)$. Se basa en la observación de que el límite que aparece en la definición de la derivada se puede aproximar por un cociente incremental, si la longitud h es suficientemente pequeña.
En efecto, de

$$\dot{x}(t_n) = \lim_{h\to 0} \frac{x(t_n+h)-x(t_n)}{h} = \lim_{h\to 0} \frac{x_{n+1}-x_n}{h}$$

resulta, si h es pequeño,

$$x_{n+1} \simeq x_n + h\dot{x}(t_n),$$

es decir,

$$x_{n+1} \simeq x_n + hf(t_n, x_n).$$

La ecuación recurrente

$$x_{n+1} = x_n + hf(t_n, x_n)$$

se toma entonces como una aproximación, que define el *método de Euler*. Notemos que el conocimiento de la condición inicial $x(t_0) = x_0$ nos permite utilizar esta recurrencia para calcular todos los valores $x(t_n)$, que es lo que queremos. Si la longitud h es muy pequeña, el número de puntos t_n y $x(t_n)$ es muy grande y eso nos permitirá, por ejemplo, dibujar la gráfica de la solución $x(t)$ con gran aproximación.
Vamos a utilizar Maxima para estudiar mediante el método de Euler una ecuación sencilla:

$$\dot{x} = a - x$$

donde $a \in \mathbb{R}$ es una cierta constante. Naturalmente, esta ecuación puede resolverse de manera exacta y de hecho por eso la hemos elegido, para poder comparar los resultados aproximados con los exactos.
Vamos a ver que el método es convergente. Lo haremos viendo explícitamente que las soluciones aproximadas convergen a la solución exacta cuando $h \to 0$.
Comenzamos con la ecuación recursiva, que en este caso es

$$x_{n+1} = x_n + h(a - x_n) = (1-h)x_n + ha.$$

Lo que nos interesa no es esta ecuación en sí, sino su solución; es decir, queremos tener una expresión para x_n en función de x_0, h y a, que son los datos de que disponemos. Para resolver sucesiones recurrentes, en Maxima tenemos el comando `solve_rec`, que se carga con la orden:

```
(%i1) load(solve_rec)$;
```

El uso del comando `solve_rec` es muy sencillo, sólo hay que darle la recurrencia e indicarle la variable:

```
(%i2) solve_rec(x[n]=(1-h)*x[n-1]+h*a, x[n]);
```

$$(\%o2) \qquad x_n = -a\,(1-h)^n + \,\%k_1\,(1-h)^n + a$$

Aquí aparece una constante k_1. Sustituyendo el valor $t = 0$, vemos que es precisamente $k_1 = x_0$. Así, nuestra ecuación básica es

$$x_n = x_0(1-h)^n - a(1-h)^n + a$$

Supongamos por simplicidad que el intervalo donde estamos considerando la solución es del tipo $[0, t]$, con $t > 0$. Entonces, será $h = \frac{t}{n}$ y sustituyendo nos queda

$$x_n = x_0(1-\frac{t}{n})^n - a(1-\frac{t}{n})^n + a$$

Comprobemos entonces que cuando $h \to 0$, o sea $n \to \infty$, la solución converge a la exacta:

```
(%i3) assume(t>0);
```

(%o3) $[t > 0]$

```
(%i4) limit(-a*((1-(t/n))^n)+(x[0])*((1-(t/n))^n) + a,n,inf);
```

(%o4) $-a\,e^{-t} + x_0\,e^{-t} + a$

```
(%i5) factor(%o4);
```

(%o5) $e^{-t}\left(a\,e^{t} - a + x_0\right)$

```
(%i6) ode2('diff(x,t)=a-x,x,t);
```

(%o6) $x = e^{-t}\left(a\,e^{t} + \%c\right)$

```
(%i7) ic1(%o6,t=0,x=x[0]);
```

(%o7) $x = e^{-t}\left(a\,e^{t} - a + x_0\right)$

Esta solución exacta coincide plenamente con el límite `(%o5)` de las soluciones aproximadas. El método de Euler (para la ecuación particular que estamos considerando) es, pues, convergente.

7.3. Implementación del método de Euler

Ahora que conocemos la base del método, su implementación numérica está clara: se trata de crear una variable (una lista) en la que iremos almacenando los valores de los distintos nodos t_i y los correspondientes valores $x_i = x(t_i)$, para después representarlos gráficamente. Haremos esto para distintos valores de h cada vez más pequeños, con el fin de visualizar como las soluciones aproximadas se acercan a la exacta al hacer $h \to 0$. Los valores elegidos son $h = 0.3, 0.2, 0.1$. También, haremos $a = 0.5$ y tomaremos el punto inicial $x(0) = 1$. Al valor $h = 0.3$ le corresponde la lista `lista1`, a $h = 0.2$ la `lista2` y a $h = 0.1$ la `lista3`. Notemos también que a medida que hacemos h más pequeño, hay que aumentar el número de pasos en la recurrencia para cubrir el mismo intervalo de definición de la solución, que aquí tomaremos como $t \in [0, 6]$, como consecuencia de la relación $6 = hn$. Por ejemplo, con intervalos de longitud $h = 0.3$ se necesitan 20 iteraciones, pero cuando $h = 0.1$ es preciso tomar $n = 60$.
Con estas observaciones en mente, el algoritmo debería resultar claro. Comenzamos construyendo la solución aproximada para $h = 0.3$:

```
(%i8) h1:0.3;
```

(%o8) 0,3

```
(%i9) x1[0]:1;
```

(%o9) 1

```
(%i10) x1[n] := x1[n-1] - h1*x1[n-1]+(0.5)*h1;
```

(%o10) $x1_n := x1_{n-1} - h1\, x1_{n-1} + 0{,}5\, h1$

```
(%i11) lista1 : makelist([i*h1, x1[i]], i, 0, 20)$
```

Suprimimos la salida de este comando con el signo $, pues se trata de una lista enorme.
A continuación, construimos la solución para $h = 0.2$:

```
(%i12) h2:0.2;
```

(%o12) 0,2

```
(%i13) x2[0]:1;
```

(%o13) 1

```
(%i14) x2[n] := x2[n-1] - h2*x2[n-1]+(0.5)*h2;
```

(%o14) $x2_n := x2_{n-1} - h2\,x2_{n-1} + 0{,}5\,h2$

```
(%i15) lista2 : makelist([i*h2, x2[i]], i, 0, 30)$
```

Y finalmente para $h = 0.1$:

```
(%i16) h3:0.1;
```

(%o16) 0,1

```
(%i17) x3[0]:1;
```

(%o17) 1

```
(%i18) x3[n] := x3[n-1] - h3*x3[n-1]+(0.5)*h3;
```

(%o18) $x3_n := x3_{n-1} - h3\,x3_{n-1} + 0{,}5\,h3$

```
(%i19) lista3 : makelist([i*h3, x3[i]], i, 0, 60)$
```

Construimos también una lista con los valores que toma la solución exacta para $a = 0.5$ en los mismos 60 nodos que la solución con $h = 0.3$, para poder representarlas todas juntas y hacer una comparación gráfica:

```
(%i20) listaex:makelist([n*h3,float(0.5*(1+exp(-n*h3)))],n,0,60)$
```

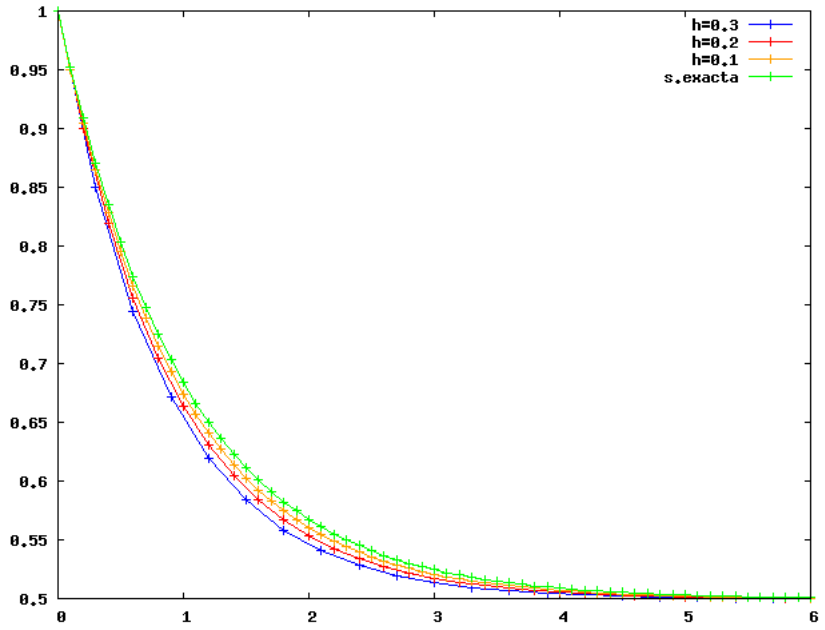
Ya estamos en condiciones de representar los datos. Cargamos para ello el paquete `draw`:

```
(%i21) load(draw)$
```

y especificamos las opciones que queremos para el gráfico dentro del comando `wxdraw2d` (existe otro comando `draw2d` que dibuja el gráfico fuera de la ventana de Maxima, en el graficador Gnuplot. El comando `wxdraw2d` embebe el gráfico *dentro* de la sesión de Maxima). Estas opciones son, por ejemplo, que en cada serie de datos se unan los puntos mediante segmentos de recta (`points_joined=true`), el color de puntos y segmentos (`color=blue`) y la etiqueta para el gráfico resultante (`key="h=0.3"`). Las opciones para cada conjunto de datos se escriben *delante* del nombre de ese conjunto (`points(lista1)`):

```
(%i22) wxdraw2d(
 points_joined=true,color=blue,key="h=0.3",points(lista1),
 points_joined=true,color=red,key="h=0.2",points(lista2),
 points_joined=true,color=orange,key="h=0.1",points(lista3),
 points_joined=true,color=green,key="s.exacta",points(listaex));
```

(%t22)



(%o22) $[gr2d\,(points, points, points, points)]$

El resultado no deja lugar a dudas sobre el comportamiento de las soluciones y de la idea detrás del método de Euler.

Nota: Si quisiéramos guardar el gráfico que genra Gnuplot, bastaría con añadir la opción `terminal=formato` al final del comando `(%i22)`, (antes del último paréntesis) donde `formato` puede ser `jpg`, `png`, `ps`, etc.

7.4. Los métodos de Runge-Kutta

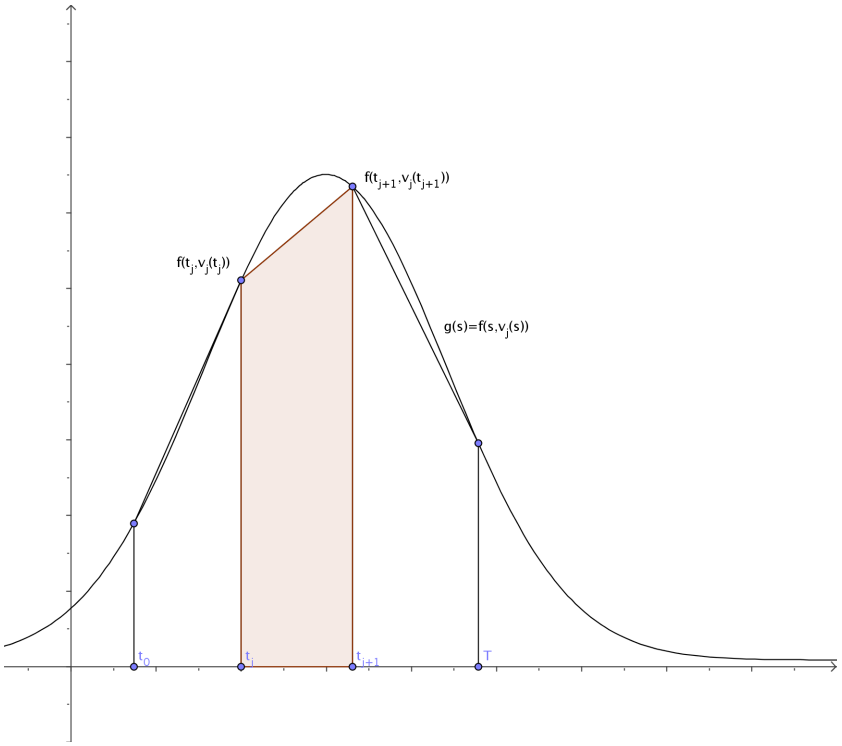
Para estudiar el problema local

$$\begin{cases} \dot{v}_j = f(t, v_j),\ t > t_j \\ v_j(t_j) = x_j, \end{cases} \tag{16}$$

se puede aprovechar el Teorema Fundamental del Cálculo, que nos da la identidad

$$v_j(t_{j+1}) - v_j(j_j) = \int_{t_j}^{t_{j+1}} f(s, v_j(s))\mathrm{d}s.$$

Para resolver de manera aproximada esta ecuación, podemos refinar la construcción de la integral como sumas de Riemann (basada en rectángulos) empleando trapecios. En cada subintervalo $[t_j, t_{j+1}]$ el trapecio determinado por las alturas $f(t_j, v_j(t_j))$ y $f(t_{j+1}, v_j(t_{j+1}))$ nos da una aproximación relativamente buena a la integral anterior.



Así, podemos considerar el método numérico definido por la "fórmula trapezoidal":

$$x_{j+1} - x_j = \frac{h}{2}\left(f(t_j, x_j) + f(t_{j+1}, x_{j+1})\right) \tag{17}$$

Esta fórmula define un método implícito consistente con orden local 3 y, cuando converge, orden global 2.
Fijémonos en que la fórmula anterior contiene incrementos de la forma que ya había aparecido al considerar el método de Euler, $hf(t_j, x_j)$. Una forma de obtener métodos explícitos de alto orden, consiste en componer este tipo de incrementos eulerianos con fórmulas implícitas. Los métodos de Runge-Kutta precisamente se basan en la composición del método de Euler explícito con fórmulas trapezoidales; por ejemplo, el método de Runge-Kutta de orden 2 consiste en tomar en la fórmula (17) la aproximación de x_{j+1} dada por la fórmula de Euler: $x_{j+1} = x_j + hf(t_j, x_j)$. Es decir, si

$$K_1 = hf(t_j, x_j)$$

y

$$K_2 = hf(t_{j+1}, x_j + K_1)$$

entonces el método de Runge-Kutta de orden 2 consiste en la ecuación

$$x_{j+1} = x_j + \frac{1}{2}(K_1 + K_2).$$

Este método resulta ser consistente con orden local $p = 3$ y, cuando converge, su orden global es 2. También notemos que el método requiere que se evalúe la función f dos veces. De forma parecida se definen los métodos de Runge-Kutta de orden superior.
En particular, el método de Runge-Kutta de orden 4 (popularmente conocido como RK4) se construye a partir de una secuencia de 4 incrementos de Euler que requieren la evaluación de f otras 4 veces, a partir de

$$\begin{aligned} K_1 &= hf(t_j, x_j) \\ K_2 &= hf(t_{j+\frac{1}{2}}, x_j + \frac{1}{2}K_1) \\ K_3 &= hf(t_{j+\frac{1}{2}}, x_j + \frac{1}{2}K_2) \\ K_4 &= hf(t_{j+1}, x_j + K_3) \end{aligned}$$

(donde $t_{j+\frac{1}{2}}$ significa $t_{j+\frac{1}{2}} = \frac{1}{2}(t_{j+1} - t_j)$) mediante la ecuación

$$x_{j+1} = x_j + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4).$$

Este método es unipaso, consistente de orden local $p = 5$ y, cuando converge, de orden global 4.

Nota: Uno puede plantearse qué ocurre con los métodos de orden superior o, en otras palabras, ¿por qué es tan popular el método RK4?. Claramente, aumentando el orden se aumenta la precisión del método. La respuesta estriba en el hecho de que aumentar el orden a partir de 4 implica aumentar extraordinariamente al complejidad de los cálculos. John C. Butcher ha demostrado que no existe ningún método de Runge-Kutta de orden global $p = 5$ con cinco etapas (evaluaciones de f), y de hecho los máximos órdenes que se pueden alcanzar para los diversos números de etapas vienen dados por la siguiente tabla, donde m representa el número de etapas:

$$\begin{aligned} p(m) &= m, \quad para \quad m = 1, 2, 3, 4 \\ p(m) &= m - 1, \ para \ m = 5, 6, 7 \\ p(m) &= m - 2, \ para \ m = 8, 9 \\ p(m) &\leq m - 2 \ para \ m \geq 10 \end{aligned}$$

Así, por ejemplo, el método de Runge-Kutta de cinco etapas proporciona el mismo orden que RK4, pero con un coste computacional más elevado. A la

vista de la tabla, se comprende que RK4 proporciona el compromiso óptimo entre precisión y complejidad y por eso es el más usado.

Veamos un ejemplo típico en el cual el método RK4 funciona muy bien. Se trata de estudiar numéricamente la solución a la ecuación diferencial

$$\dot{x} = 2t\left(e^{-t^2} - x\right).$$

que pasa por el punto $(t = 0, x = 2)$.
Cargamos el paquete `dynamics`, que contiene el comando `rk`. Éste es el que utilizaremos para realizar cálculos, pues el programa al que llama implementa el método de Runge-Kutta de orden 4. También cargamos el paquete `draw` (si no lo tuviéramos cargado ya) para hacer las representaciones gráficas.

```
(%i23) load(dynamics)$
(%i24) load(draw)$
```

Definimos la ecuación:

```
(%i25) eq1:'diff(x,t)=2*t*(exp(-t^2)-x);
```

$$(\%o25) \qquad \frac{d}{d\,t}\,x = 2\,t\,\left(e^{-t^2} - x\right)$$

Para poder hacernos una idea de cuán buena es la solución proporcionada por RK4, hemos elegido una ecuación que se puede resolver de forma exacta.Su solución con las condiciones iniciales dadas se puede calcular como ya hemos visto:

```
(%i26) ode2(%o3,x,t);
```

$$(\%o26) \qquad x = \left(t^2 + \%c\right)\,e^{-t^2}$$

```
(%i27) ic1(%o4,t=0,x=2);
```

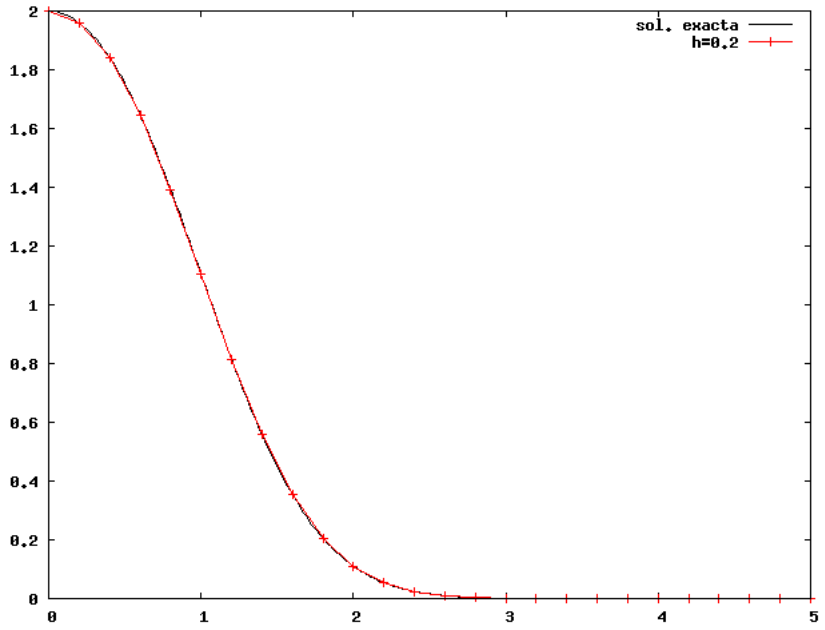
$$(\%o27) \qquad x = \left(t^2 + 2\right)\,e^{-t^2}.$$

Ahora, aplicamos el método RK4. La sintaxis es: `rk(`*expr*$, x, x_0, [t, t_0, T, h]$`)`, donde *expr* es el miembro derecho de $\dot{x}(t) = f(x, t)$, x es la variable dependiente, t es la variable dependiente, x_0 es el valor que toma $x(t)$ en $t = t_0$, T es el valor hasta el cual queremos extender la solución y h es el tamaño de los subintervalos que utilizaremos.
En este ejemplo, vamos a calcular la solución en el intervalo $[t_0 = 0, T = 5]$, utilizando un tamaño de paso $h = 0.2$. La salida de `rk` es una lista que contiene todos los valores (t_j, x_j); obviamente, esta lista no es lo que nos interesa, sino su representación gráfica. Lo que hay que hacer entonces es almacenar esta lista con un nombre, para luego poder usarla haciendo referencia a ese nombre. Así, el comando a utilizar es:

```
(%i28) tabla1:rk(2*t*(%e^(-t^2)-x),x,2,[t,0,5,0.2])$
```

Ahora, representamos la solución aproximada frente a la exacta (esta última se puede graficar en `draw2d` o `wxdraw2d` utilizando la opción *explicit* en lugar de *points*, que es la que usamos para conjuntos discretos de datos. La sintaxis es *explicit(f(var),var,varmin,varmax)*, que es autoexplicativa):

```
(%i29) wxdraw2d(
key="sol. exacta",explicit((t^2+2)*exp(-t^2),t,0,5),
points_joined=true,color="red",key="h=0.2",points(tabla1)
);
```



(%o29) $[gr2d\,(explicit, points)]$

En este ejemplo, la aproximación es asombrosamente buena. No siempre ocurre así, como se puede ver en el siguiente ejemplo. Se trata de resolver aproximadamente la ecuación

$$y' = y + \sin(x),$$

con las condiciones iniciales $y = 1$ cuando $x = 0$. Esta ecuación, para poder comparar, también se puede resolver de manera exacta:

```
(%i30) eq2:'diff(y,x)=y+sin(x);
```

(%o30) $$\frac{d}{d\,x}\,y = y + sin\,(x)$$

```
(%i31) ode2(%,y,x);
```

$$(\%o31) \qquad y = e^{x}\left(\frac{e^{-x}\,\left(-sin\left(x\right)-cos\left(x\right)\right)}{2}+\%c\right)$$

```
(%i32) ic1(%,x=0,y=1);
```

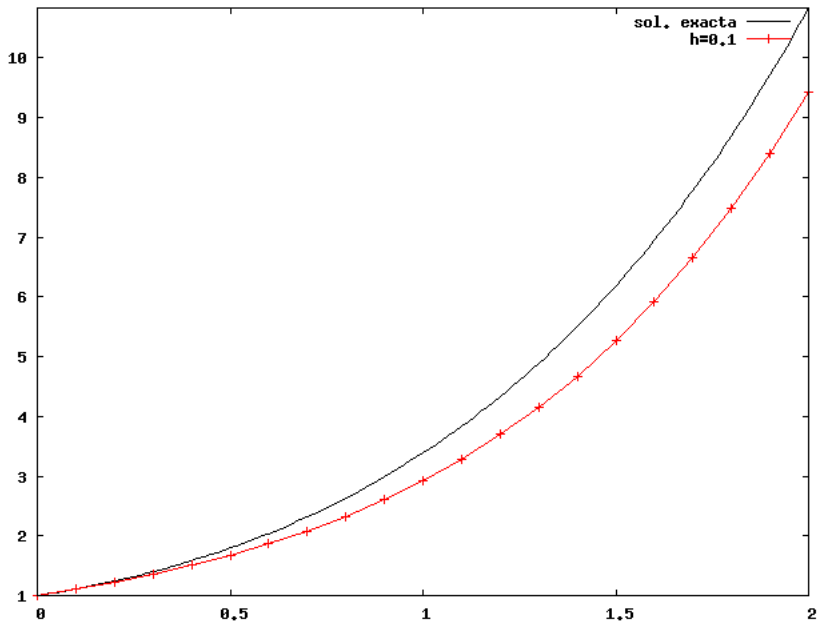
$$(\%o32) \qquad y = -\frac{sin\left(x\right)+cos\left(x\right)-3\,e^{x}}{2}$$

La solución aproximada, con un tamaño de paso $h = 0.1$, la obtenemos mediante:

```
(%i33) tabla2:rk(-(sin(x)+cos(x)-3*%e^x)/2,y,1,[x,0,2,0.1])$
```

y ahora podemos representar ambas en un mismo gráfico:

```
(%i34) wxdraw2d(
key="sol. exacta",explicit(-(sin(x)+cos(x)-3*%e^x)/2,x,0,2),
points_joined=true,color="red",key="h=0.1",points(tabla2)
);
```



$$(\%o34) \qquad [gr2d\left(explicit, points\right)]$$

Aquí vemos que la aproximación es más o menos buena sólo en el intervalo $[0, 0.5]$, mientras que en el ejemplo anterior, con un valor de h mayor $h = 0.2$, el ajuste era excelente en todo el intervalo $[0, 5]$. ¿Por qué ocurre esto?. La respuesta es muy sencilla: recordemos que el orden (global) del método RK4 es precisamente 4. Esto quiere decir, *grosso modo*, que garantizamos que la solución aproximada es tan buena como la que proporciona el polinomio de Taylor de 4^o grado de la solución. Pero aquí la solución está dominada por una exponencial, que contiene polinomios de *todos* los órdenes, por lo que a partir de $x = 1$, la aproximación con términos de cuarto orden únicamente no podrá reproducir el comportamiento global de la solución (suele decirse, con otras palabras, que "la exponencial crece más deprisa que cualquier polinomio".

El comando `rk` también es capaz de trabajar con sistemas de ecuaciones de primer orden. Esto nos permite poder aplicar el método RK4 a ecuaciones de orden $n \geq 2$ que, como sabemos (véase (3)), son equivalentes a un sistema de ecuaciones de primer orden. Veamos un ejemplo: supongamos que tenemos la ecuación

$$y'' - y = 2\cos(x)e^x$$

con las condiciones iniciales, $(x = 0, y = 1)$, $(x = 0, \dot{y} = 0)$. Nos interesa la solución definida en el intervalo $[0, 3]$. Para integrarla numéricamente, escribimos su sistema equivalente; definimos $x_1(t) = y(t)$, $x_2(t) = y'(t)$, y obtenemos:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_1 + 2\cos(t)e^t. \end{cases} \tag{18}$$

Antes de nada, observemos que esta ecuación se puede resolver exactamente, lo cual nos servirá para saber qué tan bueno es el método aproximado:

```
(%i35) ec:'diff(y,x,2)-y=2*exp(x)*cos(x);
```

$$(\%o35) \qquad \frac{d^2}{d\,x^2}\,y - y = 2\,e^x\,cos\,(x)$$

```
(%i36) ode2(%,y,x);
```

$$(\%o36) \qquad y = \frac{4\,e^x\,sin\,(x) - 2\,e^x\,cos\,(x)}{5} + \%k1\,e^x + \%k2\,e^{-x}$$

```
(%i37) ic2(%,x=0,y=1,'diff(y,x)=0);
```

$$(\%o37) \qquad y = \frac{4\,e^x\,sin\,(x) - 2\,e^x\,cos\,(x)}{5} + \frac{e^x}{2} + \frac{9\,e^{-x}}{10}$$

Ahora, resolvemos el sistema equivalente (18) mediante el método RK4, tomando $h = 1$. La sintaxis del comando `rk` es la misma de antes, sólo que hay que pasarle una lista con las dos ecuaciones en lugar de una sola ecuación, una lista con las dos variables dependientes en lugar de una sola variable dependiente y uan lista con las dos condiciones iniciales en lugar de una sola condición inicial:

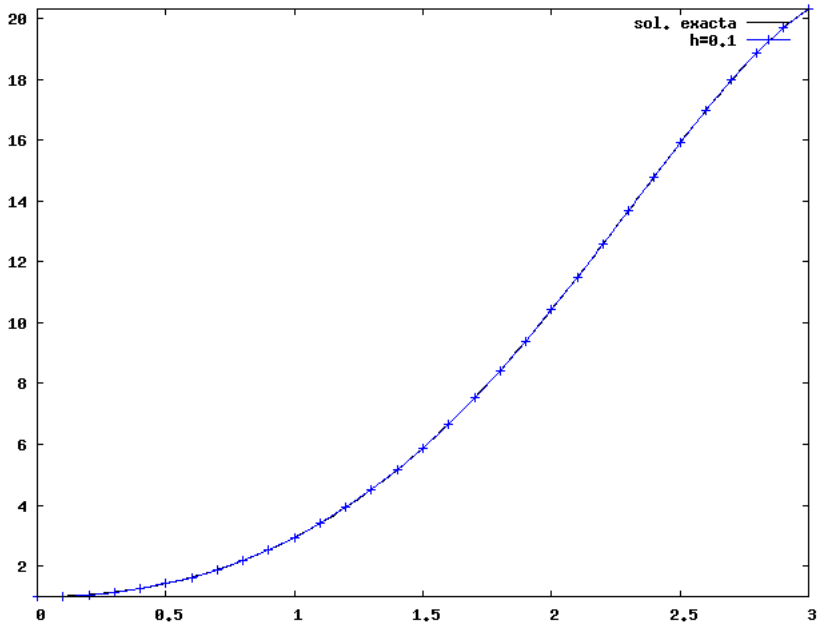
```
(%i38) tabla:
rk([x[2],x[1]+2*exp(t)*cos(t)],[x[1],x[2]],[1,0],[t,0,3,0.1])$
```

En la lista `tabla` hemos guardado los datos de las ternas $(t_j, (x_1)_j, (x_2)_j)$, esto es, de las (t_j, y_j, y'_j). Para representar gráficamente la solución, sólo nos interesan las dos primeras componentes de cada elemento de la lista. Entonces, construimos otra lista `tablab` cuyos elementos son precisamente esos (t_j, y_j):

```
(%i39) tablab:
makelist([tabla[i][1],tabla[i][2]],i,1,length(tabla))$
```

Si no lo estuviera, cargaríamos el paquete `draw`. Para representar juntas la solución exacta y la aproximada, hacemos:

```
(%i40) wxdraw2d(
key="sol. exacta",
explicit(%e^x*((4*sin(x)-2*cos(x))/5+1/2)+(9*%e^(-x))/10,x,0,3),
points_joined=true,color="blue",key="h=0.1",points(tablab)
);
```



(%o40) $[gr2d\,(explicit, points)]$

Consideremos ahora este otro caso:

$$y'' - y = e^{\cos(x)}.$$

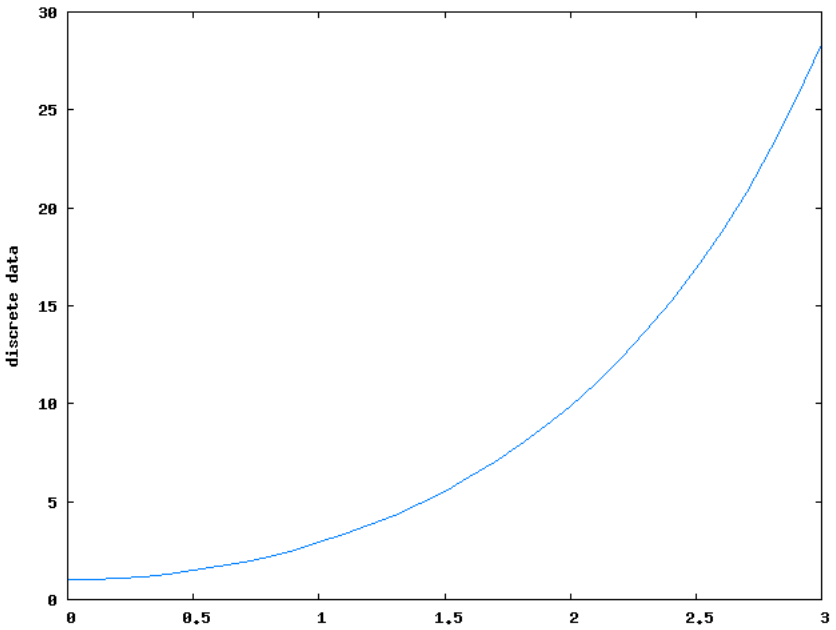
con las mismas condiciones iniciales anteriores, $(x = 0, y = 1)$, $(x = 0, \dot{y} = 0)$. Evidentemente, la presencia de la función $e^{\cos(x)}$ hace que esta ecuación ya no se pueda tratar en forma exacta.
El sistema equivalente de esta ecuación es:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_1 + e^{\cos(t)}. \end{cases}$$

Para integrarla numéricamente, según hemos visto, basta con hacer:

```
(%i41) tabla2:
rk([x[2],x[1]+exp(cos(t))],[x[1],x[2]],[1,0],[t,0,3,0.1])$
(%i42) tabla2b:
makelist([tabla2[i][1],tabla2[i][2]],i,1,length(tabla2))$
(%i43) wxplot2d([discrete,tabla2b]);
```

(%o43)



GNU Free Documentation License

Version 1.2, November 2002

51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. Applicability and definitions

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "**Document**", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "**you**". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "**Modified Version**" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "**Secondary Section**" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter

of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "**Invariant Sections**" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "**Cover Texts**" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "**Transparent**" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "**Opaque**".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "**Title Page**" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "**Entitled XYZ**" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "**Acknowledgements**", "**Dedications**", "**Endorsements**", or "**History**".) To "**Preserve the Title**" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers

are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. Verbatim copying

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. Copying in quantity

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. Modifications

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network

location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. Combining documents

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the

original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. Collections of documents

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. Aggregation with independent works

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. Translation

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders,

but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. Termination

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. Future revisions of this license

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.