Resolución explícita de ecuaciones diferenciales ordinarias de primer orden.

Capítulo 1- Libro **+** ***Hoja de problemas 2***

Resumen de contenidos:

- **Comando "dsolve" y fórmulas integrales: distintas versiones de Matlab**
- **ED de variables separadas, homogéneas y reducibles a ellas**
- **Ecuaciones diferenciales exactas y reducibles a ellas**
- **ED lineales y reducibles a ellas: ecuaciones de Riccati**
- **Modelos diferenciales**

% Seguimos el Capítulo 1 del libro de apuntes: resolución de ED y'=f(x,y), donde x es la variable independiente e y(x) la función incógnita. Se buscan soluciones generales y soluciones particulares, y se resuelven problemas de Cauchy: y'=f(x,y), y(x_0)=y_0. Usamos "dsolve" intentado interpretar el resultado en términos de la teoría general.

% El resultado, en la resolución de ED, puede depender de la versión de Matlab. Intentamos trabajar con Matlab 2011, volviendo a versiones anteriores para comparar soluciones, si Matlab 2011 no resuelve o si no da un resultado adecuado (ver la relación de Matlab con Maple, e.g., en las direcciones de internet
http://www-h.eng.cam.ac.uk/help/tpl/programs/Matlab/maplesymbolic.html
http://www.mathworks.es/es/help/symbolic/index.html
http://en.wikibooks.org/wiki/MATLAB_Programming/Symbolic_Toolbox)

% ***Ejemplo (ejercicio 1.1, d):*** *y'= y*

% Se trata de una ED de variables separadas y'=f(x)g(y): la solución general está dada de la forma int(1/g(y),y)-int(f(x),x)-C=0, siendo C la constante de integración. Resolvemos usando la fórmula y con el comando "dsolve".

>> syms y x

>> int(1/y,y)-int(1,x)

ans =

log(y) – x

% Así, se tiene la solución general en forma implícita log(y)-x=C ; despejando y=y(x), se tiene la solución en forma explícita y=C exp(x)

% El comando "dsolve" intenta darnos la solución en forma explícita. La forma de introducir la derivada y' es con Dy

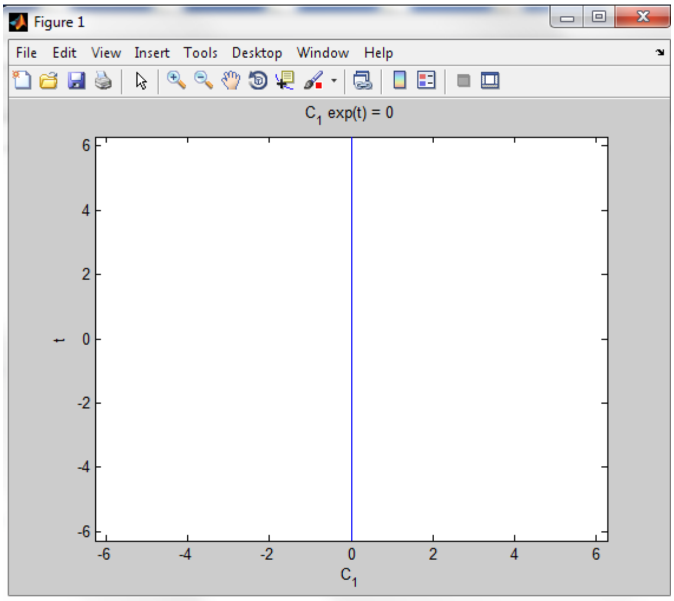
>> dsolve('Dy=y')

ans =

C1*exp(t)

% Por defecto, vemos que integra con respecto a t (t variable independiente), y y=C1 exp(t) es la solución general, con C1 la constante de integración . Dependiendo de la versión de Matlab, al ir resolviendo distintas ED, Matlab va cambiando el nombre de la constante de integración denotándolas con C1, C2, C3, … o con C1 siempre.

>> ezplot(ans) *% no dibuja soluciones de la ED*



% Obviamente "ezplot(ans)" no proporciona la gráfica de las curvas solución. Para hacer gráficas de soluciones hay que dar valores a C1 (con una condición inicial, e.g.)

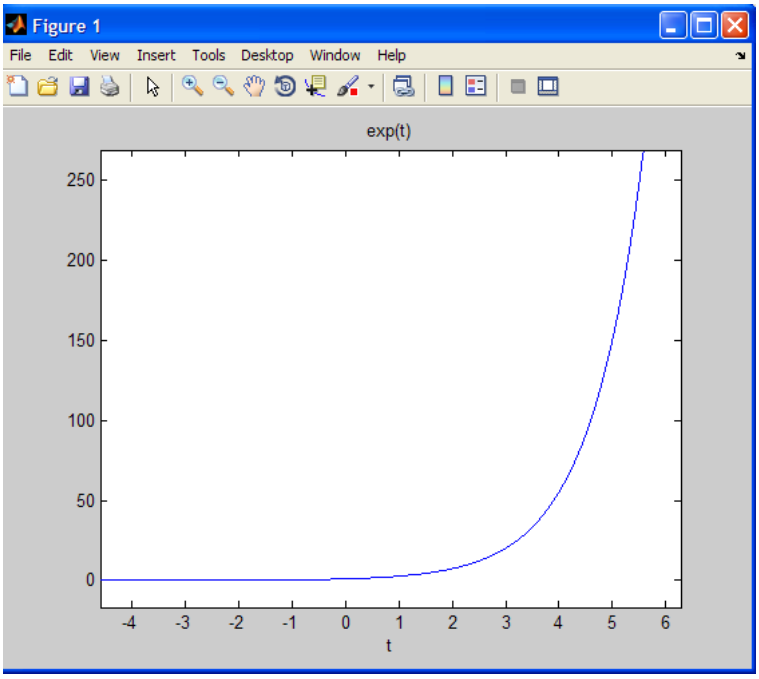
% Imponiendo una condición inicial y(0)=1: solución con gráfica pasando por (0,1)

>> dsolve('Dy=y','y(0)=1')

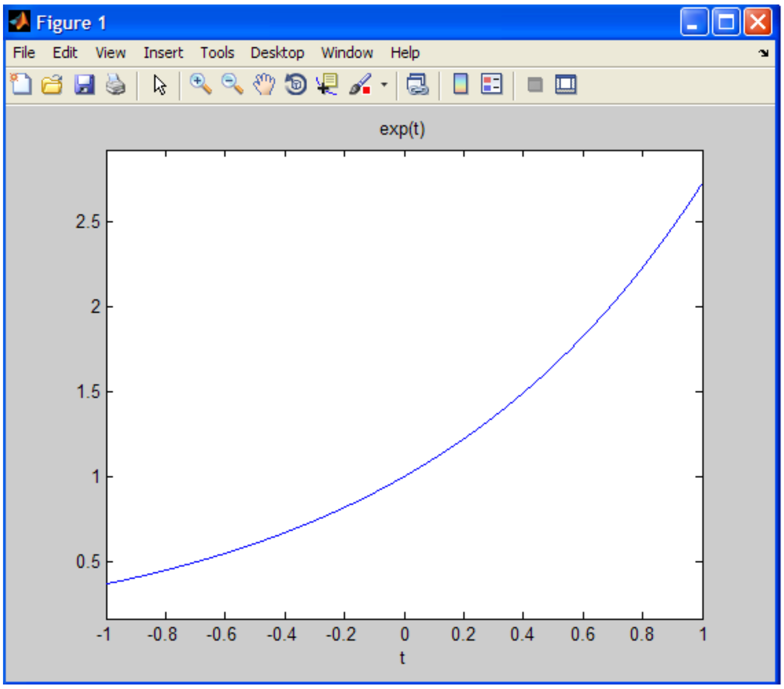
ans =

exp(t)

>> ezplot(ans) *% por defecto lo hace la gráfica en un intervalo que elige*



>> ezplot(ans,[-1,1]) *% gráfica de la solución en [-1,1]*



Sobre soluciones explícitas e intervalos de definición de éstas

*% **Ejemplo (ejercicio 1.2)**: una ED lineal y'=t+x*

>> dsolve('Dy=t+y') *% solución general que depende de una constante de integración*

ans =

C3*exp(t) - t – 1

>> dsolve('Dy=x+y') *% no entiende que x es la variable independiente*

ans =

C5*exp(t) – x

% no nos da la solución, aunque incluye las variables t y x: ha tomado t como la variable independiente y x como una constante (C5 la constante de integración). Se puede pedir que integre con respecto a x

>> dsolve('Dy=x+y','x')

ans =

C8*exp(x) - x – 1

% ha tomado x como variable independiente, y ahora sí se tiene la solución general (la misma que arriba con "dsolve('Dy=t+y')": ahora C8 es la constante de integración.

% Resolviendo un problema de Cauchy asociado a la ED lineal y'=t+y: para esta ED cualquier solución está definida en toda la recta real

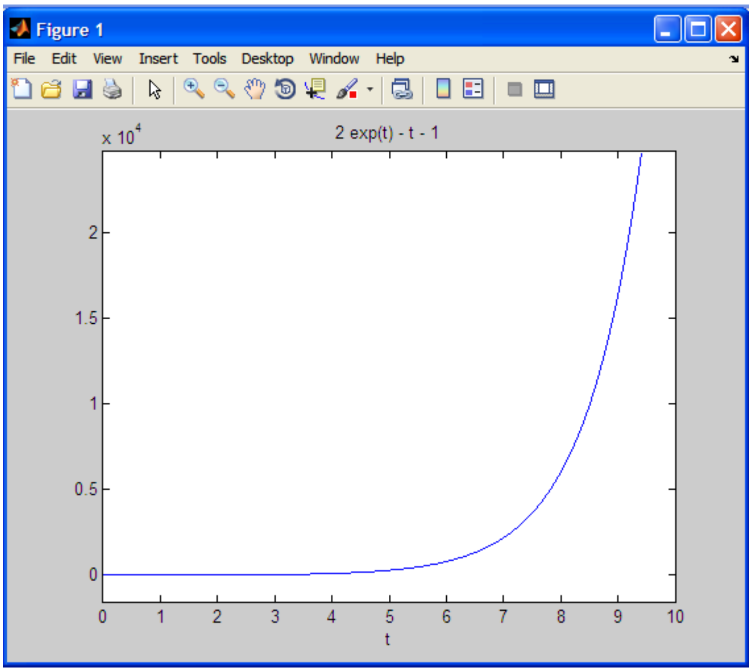
>> dsolve('Dy=t+y','y(0)=1')

ans =

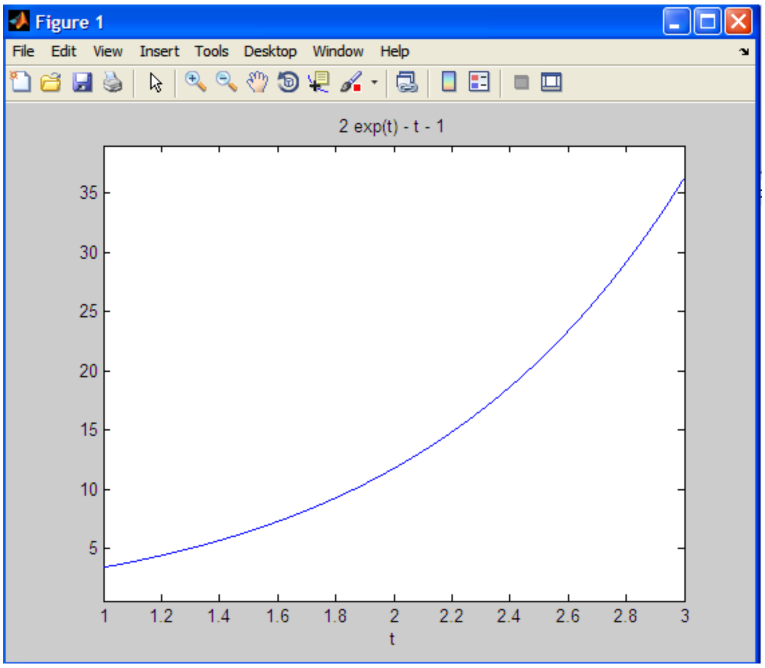
2*exp(t) - t - 1

% haciendo la gráfica en distintos intervalos

>> ezplot(ans,[0,10])



>> ezplot(ans,[1,3])



% si queremos que la x sea la variable independiente, se lo podemos indicar también en la resolución del P. de Cauchy,

>> dsolve('Dy=x+y','y(0)=1','x')

ans =

-1-x+2*exp(x)

% __Ejemplo (ejercicio 1.1, b):__ ED de variables separadas y'=-t/y, cuya solución general en forma implícita es x^2+y^2=C^2, como se obtiene aplicando las fórmulas integrales

>> syms x y

>> int(y,y)+int(x,x)

ans =

x^2/2 + y^2/2

% Las curvas solución son por tanto las circunferencias centradas en el origen. La solución en forma explícita se tiene despejando y en x^2+y^2=C^2 (en forma inversa despejando x, y en forma paramétrica con x=Ccos(t), y=Csin(t), por ejemplo)

>> dsolve('Dy=-t/y') *% por defecto t es la variable independiente*

ans =

 2^(1/2)*(C3 - t^2/2)^(1/2)
-2^(1/2)*(C3 - t^2/2)^(1/2)

% es la solución general en forma explícita, que se obtiene al despejar y de y^2/2+t^2/2=C3, con C3 la constante de integración

>> pretty(ans)

```
+-                        -+
|         /        2 \1/2  |
|     1/2 |       t   |    |
|    2    | C3 - -- |      |
|         \       2  /     |
|                          |
|          /        2 \1/2 |
|      1/2 |       t   |   |
|   - 2    | C3 - -- |     |
|          \       2  /    |
+-                        -+
```

% aunque se ve más rápido con Matlab 2008, donde despeja de y^2+t^2=C

```
>> dsolve('Dy=-t/y')

ans =

 (-t^2+C1)^(1/2)
-(-t^2+C1)^(1/2)
```

% Si queremos que x sea la variable de integración hay que indicarlo en el comando "dsolve", introduciendo 'x'

```
>> dsolve('Dy=-x/y')  % integra con respecto a t

ans =

 2^(1/2)*(C7 - t*x)^(1/2)
-2^(1/2)*(C7 - t*x)^(1/2)

>> x

Undefined function or variable 'x'.
```

% este resultado de arriba depende de la versión de Matlab pero ha resuelto una ED distinta a la que queríamos

```
>> dsolve('Dy=-x/y','x') % entiende que x es la variable independiente

ans =

 2^(1/2)*(C11 - x^2/2)^(1/2)
-2^(1/2)*(C11 - x^2/2)^(1/2)
```

% Un P. de Cauchy asociado: encontrar la solución con gráfica pasando por (0,1)

```
>> dsolve('Dy=-t/y','y(0)=1')

ans =

2^(1/2)*(1/2 - t^2/2)^(1/2)
```

% Matlab se ha quedado solamente con una solución: la semi-circunferencia de radio 1 para t en (-1,1). Comparamos con la solución cuya gráfica pasa por (1,0) (donde la pendiente es "infinito")

>> dsolve('Dy=-t/y','y(1)=0')

ans =

```
 2^(1/2)*(1/2 - t^2/2)^(1/2)
-2^(1/2)*(1/2 - t^2/2)^(1/2)
```

% __Ejemplo (ejercicio 1.1, f):__ ED de variables separadas y'=sin(x)sin(y). Resolvemos primero usando fórmulas integrales, y luego con "dsolve": vemos que el resultado depende de la versión de Matlab.

>> syms y x

>> int(1/sin(y),y)-int(sin(x),x)

ans =

log(tan(y/2)) + cos(x)

% solución general log(tan(y/2)) + cos(x)=C

% "dsolve" intenta darnos la solución explícita: con Matlab 2008,

>> dsolve('Dy=sin(y)*sin(t)');

>> pretty(ans)

```
                                                             2
            exp(-cos(t)) C1        -exp(-2 cos(t)) C1   + 1
   atan(2 ----------------------, -----------------------)
                                2                          2
          1 + exp(-2 cos(t)) C1   1 + exp(-2 cos(t)) C1
```

% y con Matlab 2011, nos da también las soluciones particulares que encuentra

>> dsolve('Dy=sin(y)*sin(t)')

ans =

```
                    0
 2*atan(exp(C23 - cos(t)))
```

*% y=0 es una solución particular, faltan otras soluciones particulares como y= k*pi con k entero. La otra es la solución general en forma explícita y=2*atan(exp(C23 - cos(t))). Intentamos hacer gráficas de soluciones con "ezplot"*

>> ezplot(ans) *% no entiende qué debe dibujar*

Error using inlineeval (line 15)
Error in inline expression ==> matrix([[0], [2.*atan(exp(C3 - cos(t)))]])
…………………………………………..
…………………………………………..
% Se puede definir C23 como una variable simbólica para substituirla por un valor, y hacer una gráfica, o varias…

>> syms t C23

>> dsolve('Dy=sin(y)*sin(t)')

```
ans =

                     0
 2*atan(exp(C23 - cos(t)))
```

>> w=subs(2*atan(exp(C23 - cos(t))),C23,1) *% así identificamos C23 con 1*

```
w =

2*atan(exp(1 - cos(t)))
```

>> ezplot(w,[-5,5])

>>w=subs(2*atan(exp(C23 - cos(t))),C23,2) *% el valor de C23 es ahora 2*

```
w =

2*atan(exp(2 - cos(t)))
```

>> hold on

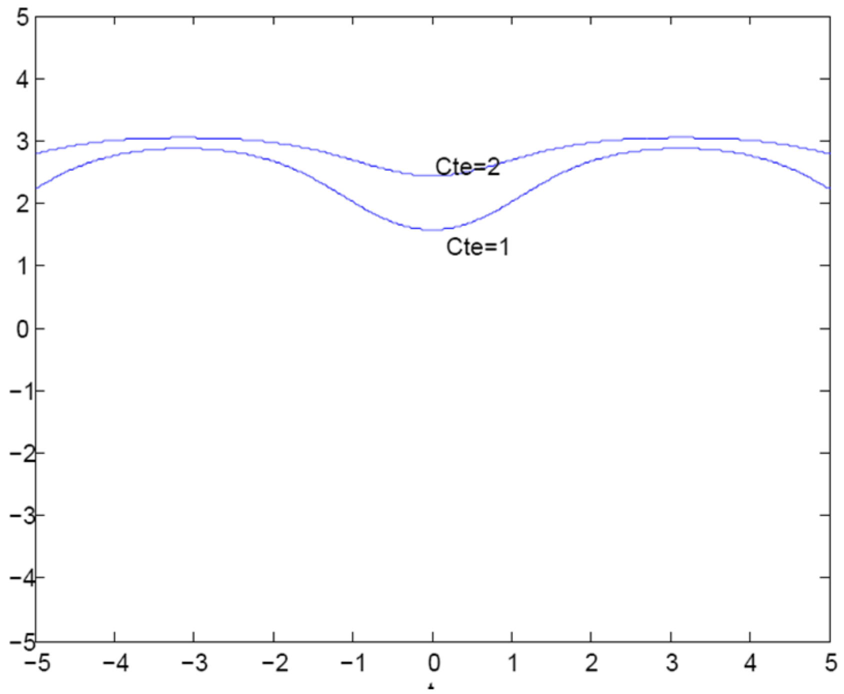
>> ezplot(w,[-5,5])

>> axis([-5,5,-5,5]) *% ajustamos los ejes*

>> gtext('Cte=1') *% ponemos comentarios*

>> gtext('Cte=2')

>> hold off



*% **Ejemplo (ejercicio 1.4, a):** una ecuación de Riccati y'=-(y/t)+y^2-(1/t^2) de la que se conoce una solución particular yp=1/t, por lo que se sabe resolver de manera explícita; Matlab busca soluciones explícitas*

>> dsolve('Dy=-(y/t)+y^2-(1/t^2)')

ans =

```
                          -1/t
 1/(t^3*(C13 + 1/(2*t^2))) - 1/t
```

% de las soluciones que nos devuelve, la primera, yp=-1/t, es una solución particular; la segunda es la solución general con C13 la constante de integración. En general, ningún valor de dicha constante nos da la solución particular

% Verificamos que yp=-1/t que es solución de la ED

>> syms t

>> yp=-1/t

yp =

-1/t

>> diff(yp)+(yp/t)-yp^2+1/t^2

ans =

0

% Verificamos que up=1/t es también otra solución particular

>> up=1/t

up =

1/t

>> diff(up)+(up/t)-up^2+1/t^2

ans =

0

% Se trata de una ED no definida en t=0: está definida en (-infinito,0) y en (0, infinito). Resolvemos un P. de Cauchy asociado

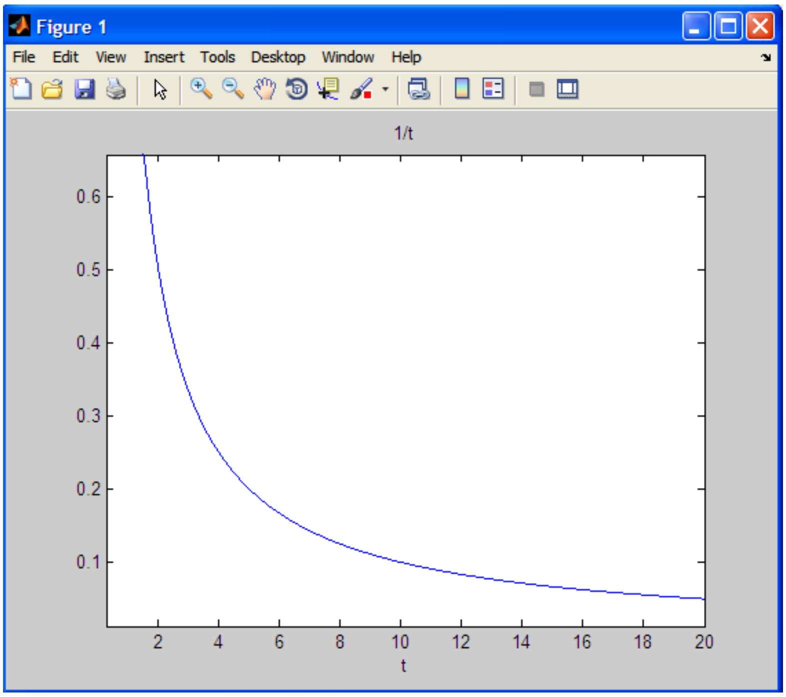
>> dsolve('Dy=-(y/t)+y^2-(1/t^2)','y(1)=1')

ans =

1/t

% ED no definida en t=0, la solución con gráfica que pasa por (1,1) no está definida tampoco en t=0, tiende a infinito para t->0, y está definida en (0,infinito)

>> ezplot(ans,[0.3,20])



% Otra solución que puede no estar definida en todo (0, infinito) es la del P. de Cauchy

>> dsolve('Dy=-(y/t)+y^2-(1/t^2)','y(2)=1') *% y(t_0)=y_0, con t_0=2, y_0=1*

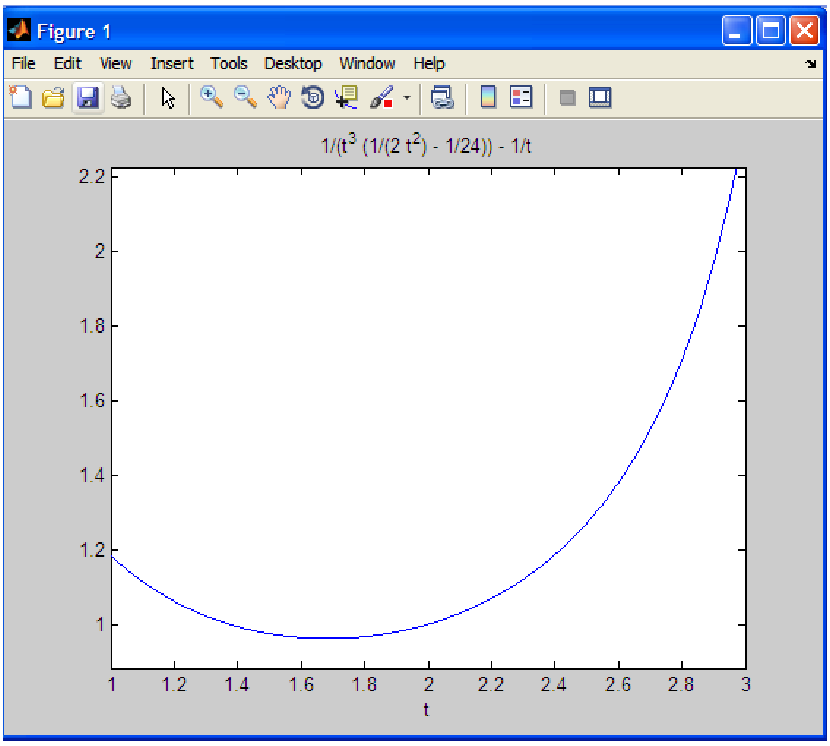
ans =

1/(t^3*(1/(2*t^2) - 1/24)) - 1/t

>> pretty(ans)

```
           1              1
  ------------------ - -
   3 /   1          \    t
  t  | ---- - 1/24 |
     |    2         |
     \ 2 t          /
```

>> ezplot(ans,[1,3]) *% intervalo que contiene a t_0=2*



% Con las versiones 2007 y 2008 de Matlab se obtiene una solución distinta, aparentemente, como se ve abajo, pero, las dos soluciones del problema de Cauchy coinciden, al menos localmente, tal y como asegura el teorema de existencia y unicidad de solución pasando por un punto.

>> dsolve('Dy=-(y/t)+y^2-1/t^2') *% solución general con Matlab 2007*

ans =

-tanh(log(t)-C1)/t

>> dsolve('Dy=-(y/t)+y^2-(1/t^2)','y(2)=1') *% solución tal que y(2)=1 con Matlab 2007*

ans =

-tanh(log(t)-log(2)-atanh(2))/t

>> pretty(ans)

```
   tanh(log(t) - log(2) - atanh(2))
 - --------------------------------
                  t
```

% La solución en un entorno de t_0=2 es la misma, como se ve puede comprobar con las gráficas de esta solución y de la función u=1/(t^3(1/(2*t^2) - 1/24)) - 1/t (solución calculada con la versión 2011 de Matlab)*

>> ezplot(ans,[1,3]) *% gráfica de la solución del P. de Cauchy en [1,3]*

% dibujamos con puntos la solución encontrada con Matlab 2011. Esta solución es la misma que se obtiene reduciendo la ED a una lineal con los cambios: y=u+1/t, z=1/u.

>> u=1/(t^3*(1/(2*t^2) - 1/24)) - 1/t

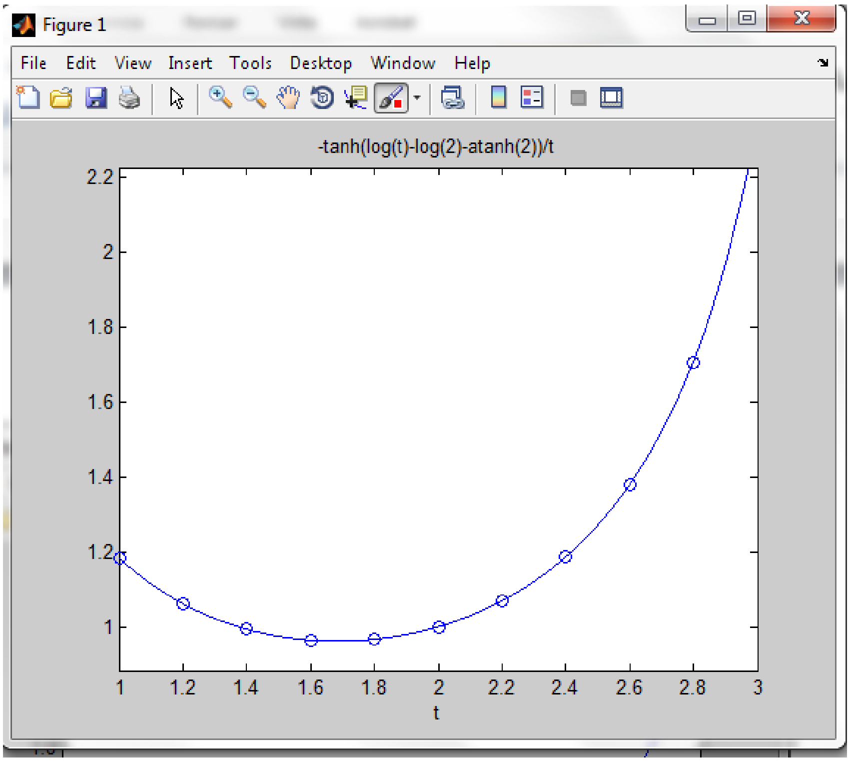
>> dt=1 :0.2 :3 ;

>> du=subs(u,t,dt) ;

>> hold on

>> plot(dt,du,'o')

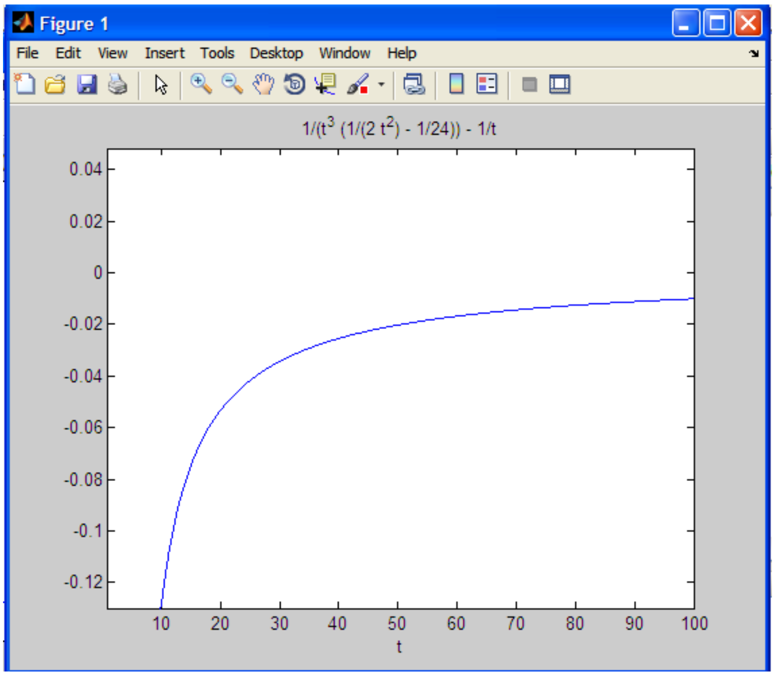
>> hold on



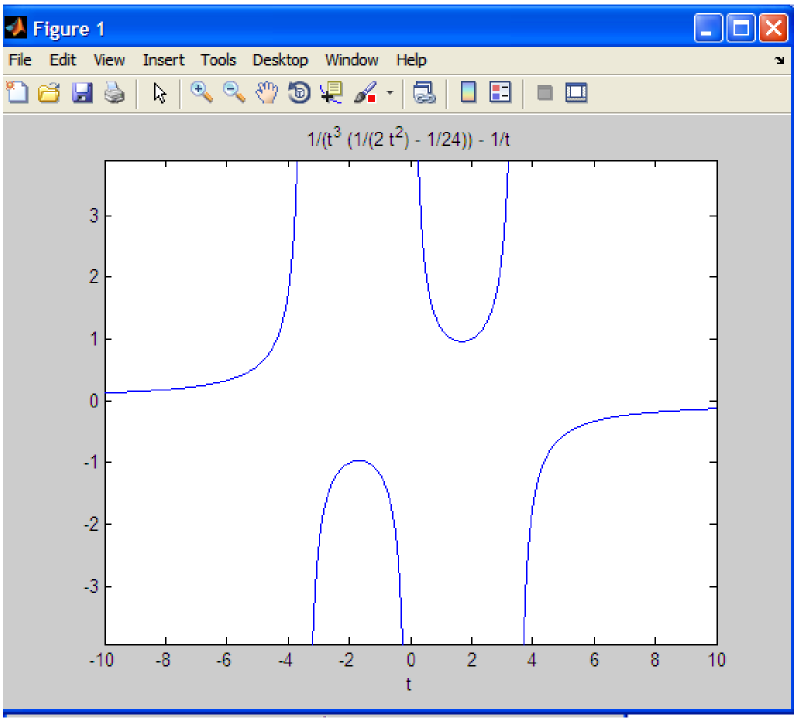
% Así, aunque no se sepa despejar, o demostrar que se tiene la misma solución, con una gráfica podemos comprobar que dos funciones, soluciones de un mismo P. de Cauchy, coinciden (por supuesto, siempre que dicho problema tenga solución única). Sin embargo, esta solución puede no estar definida en (0, infinito): puede que tenga

asíntotas. Esto es lo que pasa con nuestra solución: la solución u= 1/(t^3(1/(2*t^2) - 1/24)) - 1/t del P. de Cauchy y'=-(y/t)+y^2-(1/t^2) , y(2)=1, está definida sólo en (0,sqrt(12)), y las gráficas en intervalos aleatorios (abajo) pueden no darnos la información deseada….*

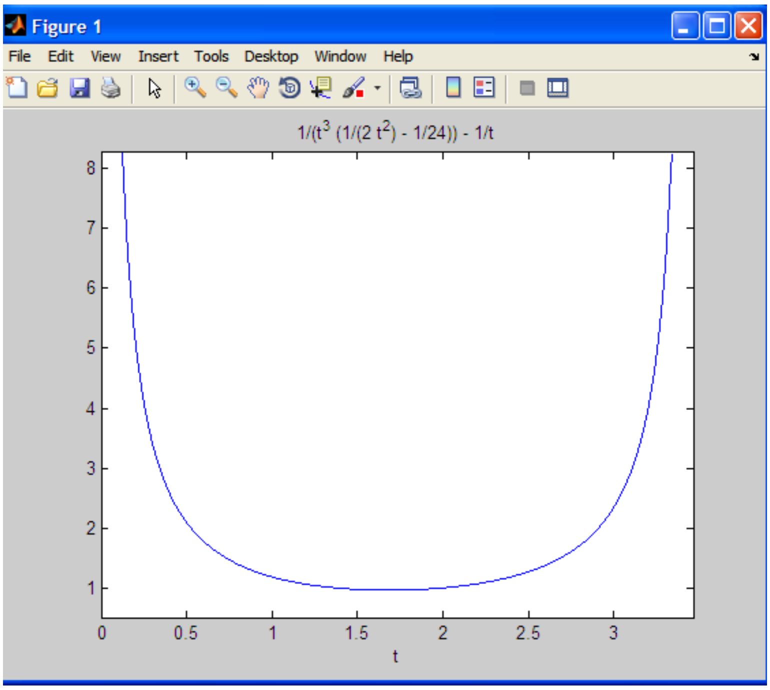
>> ezplot(ans,[1,100])



>> ezplot(ans,[-10,10])



>> ezplot(ans,[0,sqrt(12)]) *% es el intervalo de definición de la solución*



% esta es la misma gráfica que se obtiene con "ezplot(u,[0,sqrt(12)])"

>> clear all

ED homogéneas y lineales: soluciones en forma implícita y explícita

% Ecuaciones homogéneas como: y'=(t-y)/(t+y) o y'=(t+y)/(t-y). Con las distintas versiones de Matlab estas ecuaciones han pasado de resolverse con el comando "dsolve" (versiones 2007 y 2008), a no resolverse (versiones 5.3, 2010 y 2011), de una manera que puede parecer aleatoria.

*% **Ejemplo (ejercicio 1.5, b):** la ED y'=(t-y)/(t+y). Resolvemos utilizando distintas versiones de Matlab y vemos que se obtienen distintas expresiones de soluciones.*

% con la versión 2007 de Matlab (similar con Matlab 2008)

>> syms t

>> dsolve('Dy=(t-y)/(t+y)')

```
ans =

 (-t*C1+(2*t^2*C1^2+1)^(1/2))/C1
 (-t*C1-(2*t^2*C1^2+1)^(1/2))/C1
```

>> pretty(ans)

```
[                2    2      1/2]
[-t C1 + (2 t   C1   + 1)      ]
[-------------------------------]
[              C1               ]
[                               ]
[                2    2      1/2]
[-t C1 - (2 t   C1   + 1)      ]
[-------------------------------]
[              C1               ]
```

% Arriba, se tiene la solución general, con C1 la constante de integración. Comparando con la solución que nos dan las fórmulas integrales, los signos positivo y negativo acompañando a la raíz pueden interpretarse como el resultado de despejar la función incógnita y(t) en la solución general en forma implícita.

% Con la versión 2011 de Matlab

>> dsolve('Dy=(t-y)/(t+y)')

ans =

```
                 t*(2^(1/2) - 1)
                -t*(2^(1/2) + 1)
 t*((exp(C4 - 2*log(t)) + 2)^(1/2) - 1)
-t*((exp(C4 - 2*log(t)) + 2)^(1/2) + 1)
```

>> pretty(ans)

```
+-                                          -+
|                  1/2                        |
|              t (2     - 1)                  |
|                                             |
|                   1/2                       |
|             - t (2     + 1)                 |
|                                             |
|                                 1/2         |
|   t ((exp(C6 - 2 log(t)) + 2)      - 1)     |
|                                             |
|                                  1/2        |
|  - t ((exp(C6 - 2 log(t)) + 2)      + 1)    |
+-                                          -+
```

% se trata de soluciones explícitas; a mano, utilizando las fórmulas integrales, o con otra versión de Matlab, puede que no tengamos esta expresión de la solución general (C6 la constante de integración). Las dos primeras son soluciones particulares como se comprueba sin más que derivar y substituir en la ED y'=(t-y)/(t+y), o con las instrucciones

>> syms a t

>> u=a*t

u =

a*t

>> diff(u)-(-u+t)/(u+t)

ans =

a - (t - a*t)/(t + a*t)

>> simplify(ans)

ans =

a - 2/(a + 1) + 1

>> solve(ans,a)

ans =

2^(1/2) - 1
- 2^(1/2) – 1

*% Las rectas y=a*t son soluciones de y'=(t-y)/(t+y) cuando a=2^(1/2)-1 ó a=-2^(1/2)-1*

% Otras ecuaciones homogéneas, no se resuelven de manera simple con las versiones 2010 y 2011 de Matlab, mientras que versiones previas de Matlab (versiones en 2003-2008) las pueden resolver, y dar el mismo resultado que aplicando las fórmulas integrales. Lo vemos abajo

*% **<u>Ejemplo (ejercicio 1.5, a):</u>** la ED y'=(t+y)/(t-y)*

>> dsolve('Dy=(t+y)/(t-y)') *% con Matlab 2007 (similar con Matlab 2008)*

Warning: Explicit solution could not be found; implicit solution returned.
> In dsolve at 315
ans =

-1/2*log((t^2+y^2)/t^2)+atan(y/t)-log(t)-C1 = 0

>> dsolve('Dy=(t+y)/(t-y)') *% con Matlab 2011*

```
Warning: The solutions are parametrized by the symbols:
z = solve(ln(u5^2 + 1) - 2*arctan(u5) = 2*C15 - 2*ln(t), u5)

> In dsolve at 176

ans =

 t*z
 t*i
-t*i
```

% además de una advertencia da un resultado en el campo complejo…

>> help i

```
i  Imaginary unit.
   As the basic imaginary unit SQRT(-1), i and j are used to enter
   complex numbers.  For example, the expressions 3+2i, 3+2*i, 3+2j,
   3+2*j and 3+2*sqrt(-1) all have the same value.

   Since both i and j are functions, they can be overridden and used
   as a variable.  This permits you to use i or j as an index in FOR
   loops, etc.

   See also j.

   Reference page in Help browser
     doc i
```

% Se trata de una ecuación que se sabe resolver, y podemos hacerlo utilizando las formulas integrales. Haciendo el cambio de variable u=y/t, u't+u= (1+u)/(1-u), reducimos a ED de variables separadas

>> clear all

>> syms t y u

>> int(1/(((1+u)/(1-u))-u),u)-int(1/t,t)

```
ans =

atan(u) - log(u^2 + 1)/2 - log(t)
```

>> subs(ans,u,y/t)

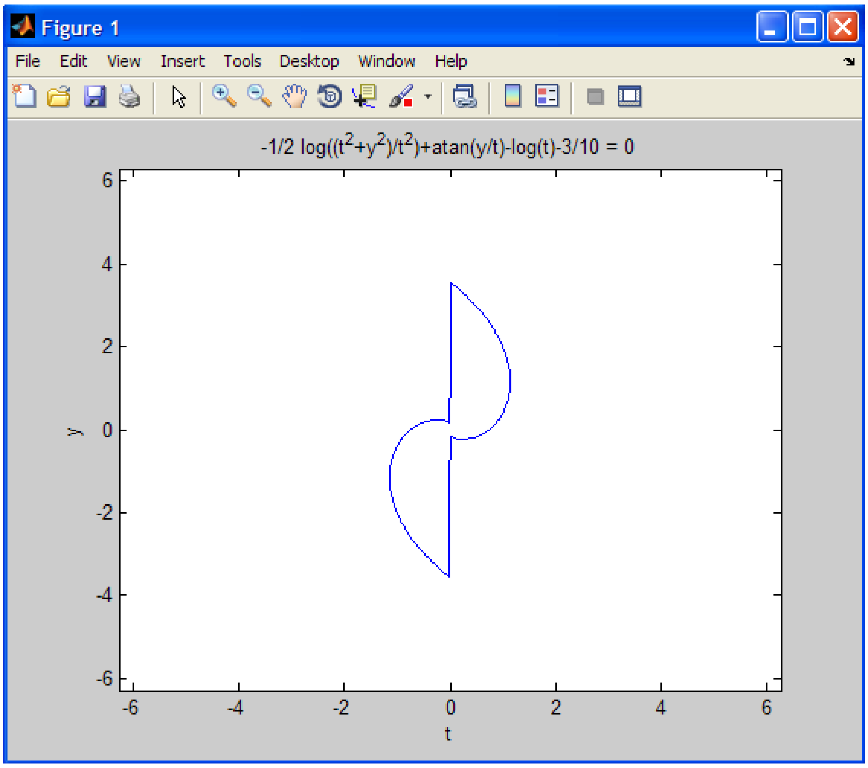
```
ans =

atan(y/t) - log(y^2/t^2 + 1)/2 - log(t)
```

% La solución general es: atan(y/t) - log(y^2/t^2 + 1)/2 - log(t)=C, también encontrada con Matlab 2007.

% Gráficas de soluciones para distintos valores de la constante C: C=0.3, C=1, C=0.1, C=-0.1, -C=0.5

>> ezplot(-1/2*log((t^2+y^2)/t^2)+atan(y/t)-log(t)-3/10)

% la misma gráfica se obtiene con "ezplot(ans-3/10)"



% Para C=3/10 vemos que hay problemas en t=0; pasa lo mismo con otros valores de C

>> ezplot(ans-1)

>> hold on

>> gtext('C=1')

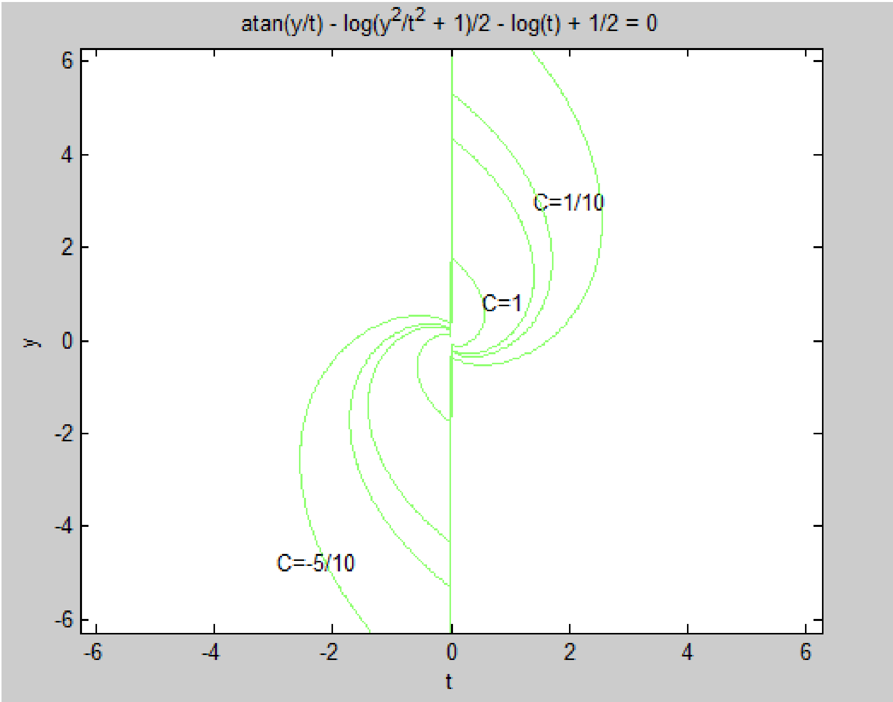
>> ezplot(ans-0.1)

>> gtext('C=1/10')

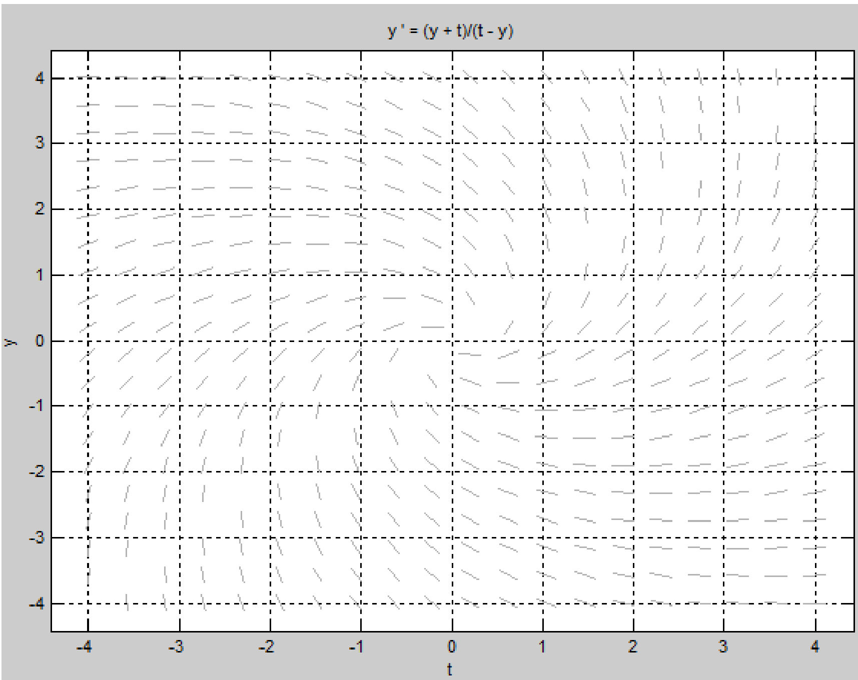
>> ezplot(ans+0.1)

>> ezplot(ans+0.5)

>> gtext('C=-5/10')

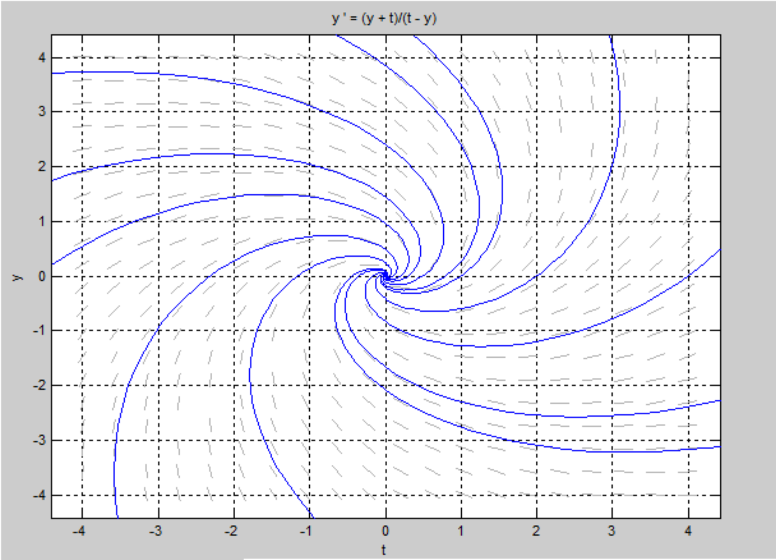


% Se observa que las gráficas no cambian si introducimos el valor absoluto dentro de los logaritmos, y que nos dan una idea parcial de las curvas solución que son tangentes a los vectores del campo (abajo; ver tercer cuaderno de curso)



*% Pero también podemos escribir la solución en coordenadas polares r=K*exp(theta), siendo r^2=t^2+y^2, t=rcos(theta), y=rsin(theta), y K la constante de integración. Abajo el campo de direcciones (con "dfield8") y el conjunto de instrucciones que nos proporciona la gráfica de las soluciones para valores de K=1,2,4,8,10,0.1,0.3,0.5,0.8…*

```
>> syms tt
>> dfield8
>> ezpolar(exp(tt),[-10,10])
>> ezpolar(2*exp(tt),[-10,10])
>> ezpolar(4*exp(tt),[-10,10])
>> ezpolar(8*exp(tt),[-10,10])
>> ezpolar(0.5*exp(tt),[-10,10])
>> ezpolar(0.1*exp(tt),[-10,10])
>> ezpolar(0.3*exp(tt),[-10,10])
>> ezpolar(0.8*exp(tt),[-10,10])
>> ezpolar(0.2*exp(tt),[-10,10])
>> ezpolar(10*exp(tt),[-10,10])
>> ezpolar(0.05*exp(tt),[-10,10])
```



>> clear all

*% **Ejemplo:** la ED lineal (del ejemplo 11 del libro de apuntes) xy'+2y=4x^2. Consideramos primero un problema de valor inicial asociado a la ED*

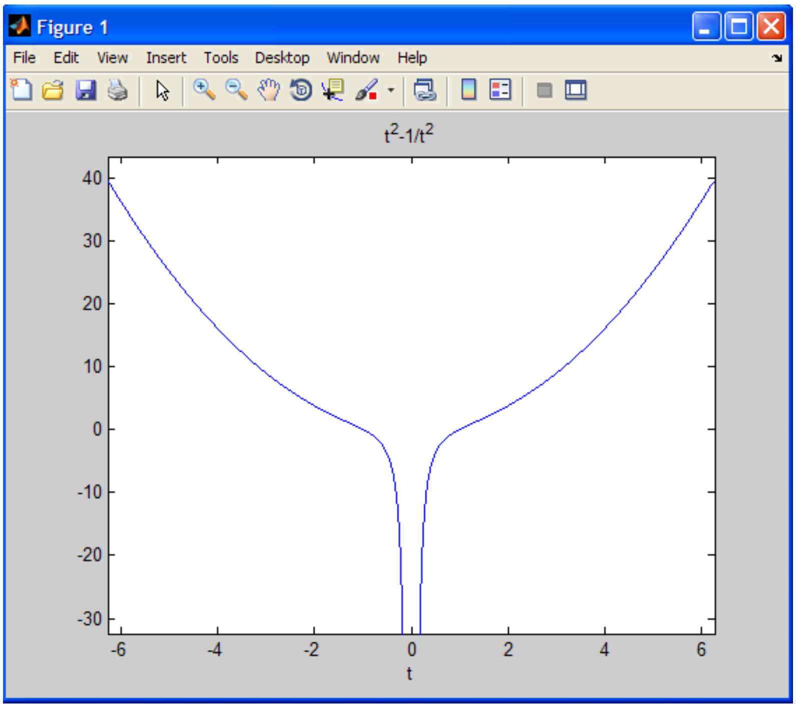
>> dsolve('Dy+2*y/t=4*t','y(1)=0')

ans =

t^2 - 1/t^2

% La teoría asegura que la solución de este P. de Cauchy está definida en (0,infinito); si hacemos la gráfica, tenemos las dos ramas de la función definida en "ans" (ver gráfica 10, Capítulo 1, en el libro de apuntes para mejor explicación del dibujo en el contexto de todas las soluciones): obviamente "ans" también es solución de la ED en (-infinito,0)

>> ezplot(ans)



% Pero podemos calcular la solución general y hacer gráficas para distintos valores de la constante de integración C1, obteniendo el dibujo de todas las curvas solución, esto es, de unas cuantas significativas. Por ejemplo, usamos el conjunto de instrucciones que viene a continuación

>> syms t C1

>> u=dsolve('Dy+2*y/t=4*t') *% solución general*

u =

(t^4+C1)/t^2

>> subs(u,C1,0) *% una de las soluciones, para el valor de la constante C1=0*

ans =

t^2

>> ezplot(ans,[-10,10]) *% gráfica de la solución en [-10,10]*

>> hold on *% para que siga dibujando en el mismo entorno gráfico*

>> subs(u,C1,1) *% otra solución para C1=1*

ans =

(t^4+1)/t^2

>> ezplot(ans,[-10,10])

>> subs(u,C1,2)

ans =

(t^4+2)/t^2

>> ezplot(ans,[-10,10])

>> gtext ('C1>0') *% texto a introducir en la gráfica*

>> gtext ('C1=0')

>> subs(u,C1,-2)

ans =

(t^4-2)/t^2

>> ezplot(ans,[-10,10])

>> subs(u,C1,-1)

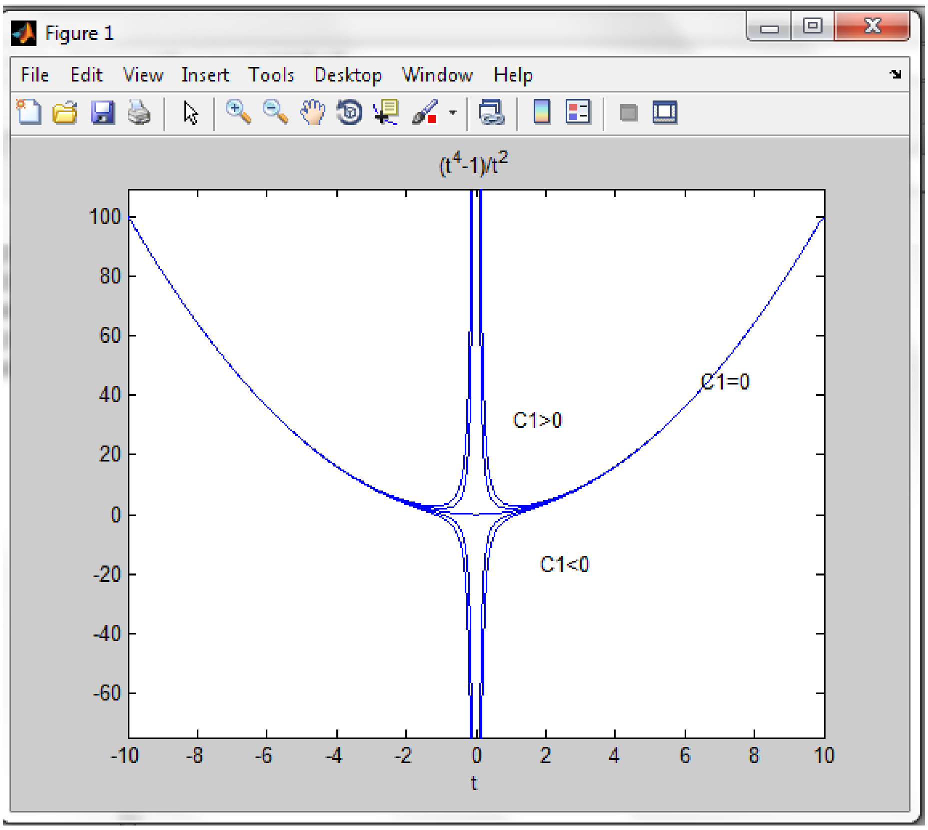
ans =

(t^4-1)/t^2

>> ezplot(ans,[-10,10])

>> gtext ('C1<0')

>> hold off *% para que deje de superponer gráficos*



% Ajustando los ejes con "axis" podemos tener mejor visualización de las soluciones. También jugando con los intervalos donde hacemos la gráfica; ver, por ejemplo, la gráfica generada con el conjunto de comandos de abajo, notando la aproximación asintótica de todas las soluciones a la solución y=t^2 para t>0 y t<0.

>> subs(u,C1,0)

ans =

t^2
>> ezplot(ans,[-5,5])

>> hold on

>> subs(u,C1,4)

ans =

(t^4+4)/t^2

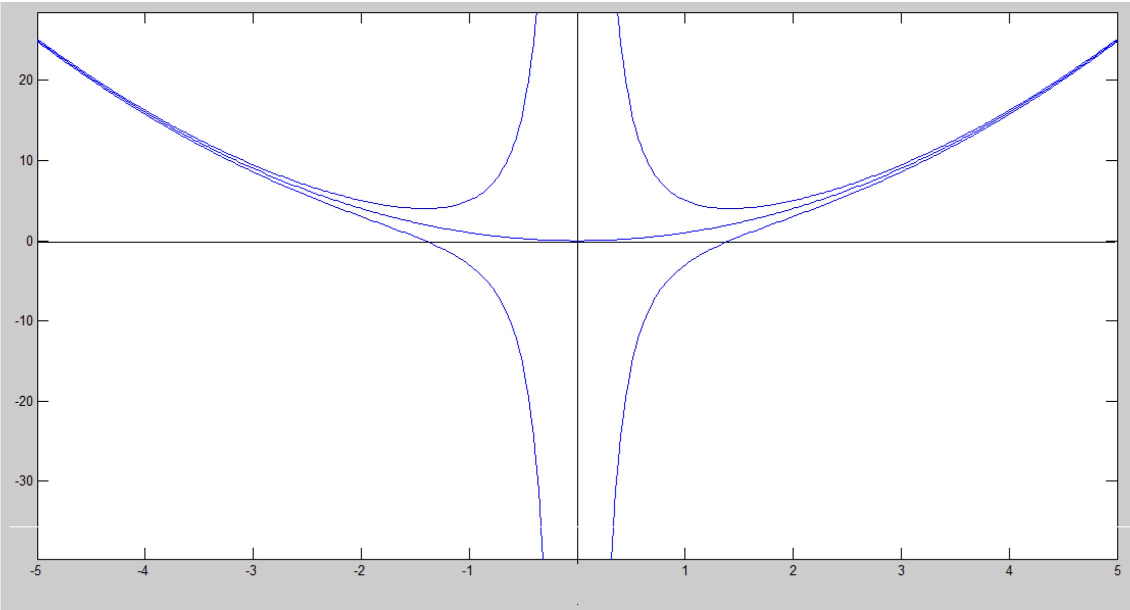
>> ezplot(ans,[-5,5])

>> subs(u,C1,-4)

ans =

(t^4-4)/t^2

>> ezplot(ans,[-5,5])



*% **Ejemplos (ejercicio 1.6):** las ED y'=(a t+b y+c)/(aa t+bb y+cc), con a,b,c,aa,bb,cc constantes. Dependiendo de la relación entre las constantes, estas ED se reducen a ED de variables separadas o a ED homogéneas. Según la versión de Matlab que utilicemos, "dsolve" nos puede devolver soluciones en que intervienen funciones no utilizadas de manera habitual, e incluso funciones desconocidas (Matlab no nos da información sobre dichas funciones).*

*% Una ED reducible a ED de variables separadas (haciendo z=2y+t, **ejercicio 1.6,a**)*

>> dsolve('Dy=(2*y+t+1)/(2*y+t-1)') *% con Matlab 2008*

ans =

-1/2*t-2/3*lambertw(-1/4*exp(-1/4)*exp(-9/4*t)*C1)-1/6

>> help lambertw

```
LAMBERTW Lambert's W function.
   W = LAMBERTW(X) is the solution to w*exp(w) = x.
   W = LAMBERTW(K,X) is the K-th branch of this multi-valued function.
   Reference: Robert M. Corless, G. H. Gonnet, D. E. G. Hare,
   D. J. Jeffrey, and D. E. Knuth, "On the Lambert W Function",
   Advances in Computational Mathematics, volume 5, 1996, pp. 329-359.

   Overloaded methods:
      sym/lambertw

   Reference page in Help browser
      doc lambertw
```

>> dsolve('Dy=(2*y+t+1)/(2*y+t-1)') *% con Matlab 2011*

ans =

```
                                                  - t/2 - 1/6
 - t/2 + exp(1/2 - (9*t)/4 - (9*C22)/2)/exp(wrightOmega(log(3/2) - (9*t)/4 - (9*C22)/2 +
1/2 + pi*i)) - 1/6
```

% aparece la unidad imaginaria, además de otra función de la que podemos obtener información

>> help wrightOmega

```
wrightOmega Wright omega function.
   W = wrightOmega(X) is a solution of the equation Y + log(Y) = X.

   Reference:
      [1] Corless, R. M. and Jeffrey, D. J. "The Wright omega Function."
      In Artificial Intelligence, Automated Reasoning, and Symbolic
      Computation (Ed. J. Calmet, B. Benhamou, O. Caprotti, L. Henocque
      and V. Sorge). Berlin: Springer-Verlag, pp. 76-89, 2002.

   Overloaded methods:
      sym/wrightOmega

   Reference page in Help browser
      doc wrightOmega
```

% ED reducible a ED homogénea ***(ejercicio 1.6, b)***

>> dsolve('Dy=(2*y+t+1)/(2*y+2*t-1)') *% se resuelve con Matlab 2008*

Warning: Explicit solution could not be found; implicit solution returned.
> In dsolve at 315

ans =

-1/2*log((-(2*t-4)^2+2*(-2*y-3)^2)/(2*t-4)^2)-2^(1/2)*atanh((-2*y-3)/(2*t-4)*2^(1/2))-log(2*t-4)-C1 = 0

>> dsolve('Dy=(2*y+t+1)/(2*y+2*t-1)') *% con Matlab 2011*

Warning: Explicit solution could not be found; implicit solution returned.
> In dsolve at 168

ans =

(t - 2)/(2^(1/2) + 2) - t + 1/2

1/2 - (t - 2)/(2^(1/2) - 2) – t

solve(12*X8108*(4*X8108 + 2*t - 2*2^(1/2)*X8108 - 2*2^(1/2)*t + 2^(1/2) + 2)^(2^(1/2)) + 8*t*(4*X8108 + 2*t - 2*2^(1/2)*X8108 - 2*2^(1/2)*t + 2^(1/2) + 2)^(2^(1/2)) + (4*X8108 + 2*t - 2*2^(1/2)*X8108 - 2*2^(1/2)*t + 2^(1/2) + 2)^(2^(1/2)) + 4*X8108^2*(4*X8108 + 2*t - 2*2^(1/2)*X8108 - 2*2^(1/2)*t + 2^(1/2) + 2)^(2^(1/2)) - 2*C34^(7^(1/3)/7)*(4*X8108 + 2*t + 2*2^(1/2)*X8108 + 2*2^(1/2)*t - 2^(1/2) + 2)^(2^(1/2)) - 2*t^2*(4*X8108 + 2*t - 2*2^(1/2)*X8108 - 2*2^(1/2)*t + 2^(1/2) + 2)^(2^(1/2)) = 0, X8108)

>> help X8108

X8108 not found.

Use the Help browser search field to search the documentation, or
type "help help" for help command options, such as help for methods.

% Así, con la versión 2011 de Matlab puede resultar difícil saber que nos devuelve.

*% **<u>Ejemplo (ejercicio 1.5):</u>** Resolvemos una ecuación homogénea y’=(x+y)/(x-y) reduciéndola a ED exacta, y aplicando las fórmulas integrales para resolver ésta (ver otras resoluciones en páginas anteriores con “dsolve” y otras fórmulas integrales)*

% Se comprueba que y’=(x+y)/(x-y) admite un factor integrante (x^2+y^2)^{-1}. Multiplicamos numerado y denominador por este factor

>> syms x y

>> p=(x+y)/(x^2+y^2)

p =

(x + y)/(x^2 + y^2)

>> q=(y-x)/(x^2+y^2)

q =

-(x - y)/(x^2 + y^2)

>> diff(p,y)-diff(q,x) *% comprobación de ecuación diferencial exacta*

ans =

2/(x^2 + y^2) - (2*y*(x + y))/(x^2 + y^2)^2 - (2*x*(x - y))/(x^2 + y^2)^2

>> simplify(ans)

ans =

0

% la ED y'=-p/q se resuelve como ED exacta, haciendo las integrales:

>> F= int(p,x)-int(q-diff(int(p,x),y),y)

F =

1/2*log(x^2+y^2)+atan(x/y)

*% solución general 1/2*log(x^2+y^2)+atan(x/y)=C que podemos comparar con la obtenida con "dsolve" y Matlab 2008 (relacionando atan(x/y) y atan(y/x))*

>> dsolve('Dy=(t+y)/(t-y)')

Warning: Explicit solution could not be found; implicit solution returned.
> In dsolve at 315

ans =

-1/2*log((t^2+y^2)/t^2)+atan(y/t)-log(t)-C1 = 0

*% **Observación:** en general, como con cualquier software matemático, debemos saber que queremos y esperamos de Matlab para obtener una información deseada sobre soluciones de ED. Dependiendo de las versiones de Matlab, dicha información parece cambiar, como se está poniendo de manifiesto a lo largo de este cuaderno.*

ED de Riccati: soluciones por desarrollos en serie de potencias

% Resolvemos varias ED de Riccati: y'+p(x)y^2+q(x)y=h(x)

% Si se conoce una solución particular y_p de dicha ED, el cambio y=y_p+z^{-1} nos lleva a la ED lineal de primer orden z'− (q(x) + 2p(x)y_p(x))z = p(x), que se resuelve con fórmulas integrales .

*% Si no se conoce una solución particular, el cambio y= v'/(v*p) nos lleva a la ED lineal de segundo orden v'' + (q(x) –p'(x)/p(x))v'− h(x)p(x)v = 0, que, dependiendo de la regularidad de p(x), puede resolverse, expresándose la solución en términos de funciones elementales o de desarrollos en serie de potencias (se consideran en el cuaderno sexto del curso). A menudo Matlab nos da la solución de una ecuación de Riccati en términos de las funciones de Bessel o Airy, u otras funciones conocidas que se expresan con desarrollos en serie de potencias (ver por ejemplo, el cuaderno sexto, y http://www.mathworks.es/es/help/symbolic/airy-and-bessel-functions-1.html).*

*% Se ha resuelto una ED de Riccati y'+y/t-y^2+(1/t^2)=0 de la que se conoce y_p=-1/t (**ejercicio 1.4, a**), viendo los distintos resultados dependiendo la versión de Matlab (ver comentarios en la primera parte del cuaderno). Por ejemplo,*

```
>> dsolve('Dy+y/t-y^2+(1/t^2)=0') % con Matlab 2011

ans =

                        -1/t
 1/(t^3*(C4 + 1/(2*t^2))) - 1/t
```

% Consideramos otras dos ED de Riccati de las que no se conoce una solución particular, y vemos los resultados con "dsolve". Lo hacemos con Matlab 2007 o Matlab 2008; las versiones 2010 y 2011 pueden dar problemas, como se ve abajo

% __Ejemplo (ejercicio 1.4, b):__ la ED y'=t^2+y^2

>> dsolve('Dy=t^2+y^2')

ans =

-t*(C1*besselj(-3/4,1/2*t^2)+bessely(-3/4,1/2*t^2))/(C1*besselj(1/4,1/2*t^2)+bessely(1/4,1/2*t^2))

>> pretty(ans)

```
                            2                        2
     t (C1 besselj(-3/4, 1/2 t ) + bessely(-3/4, 1/2 t ))
   - ----------------------------------------------------
                          2                        2
        C1 besselj(1/4, 1/2 t ) + bessely(1/4, 1/2 t )
```

% Nos da la solución en términos de las funciones de Bessel y la constante de integración C1

>> help Bessel

```
BESSEL Bessel functions of various kinds.
   Bessel functions are solutions to Bessel's differential
   equation of order NU:
         2                    2    2
        x * y'' +  x * y' + (x - nu ) * y = 0

   There are several functions available to produce solutions to
   Bessel's equations.  These are:

      BESSELJ(NU,Z)    Bessel function of the first kind
      BESSELY(NU,Z)    Bessel function of the second kind
      BESSELI(NU,Z)    Modified Bessel function of the first kind
      BESSELK(NU,Z)    Modified Bessel function of the second kind
      BESSELH(NU,K,Z)  Hankel function
      AIRY(K,Z)        Airy function

   See the help for each function for more details.
```

% Si se impone una condición inicial, bajo condiciones de existencia y unicidad de solución (que tenemos), se pueden obtener diversos términos del desarrollo en serie de Taylor de la solución.

% Por ejemplo, los n términos del desarrollo en serie de Taylor de sin(t) se obtienen con el comando "taylor"

>> u=sin(t)

u =

sin(t)

>> taylor(u,100) *% pedimos 100 términos del desarrollo*

ans =

```
-
t^99/93326215443944152681699238856266700490715968264381621468592963895
2175999932299156089414639761565182862536979208272237582511852109168640
0000000000000000000000 +
t^97/96192759682482119853328425949563698712343813919172976158104477319
333745612481875498805879175589072651261284189679678167647067832320000
000000000000000000 -
t^95/10329978488239059262599702099394727095397746340117372869212250571
234293987594703124871765375385424468563282236864226607350415360000000
000000000000000 +
t^93/11567725070816415747592051623062404362147532295764135351861422812
132468071214673152152032895168448453038389962893870780907520000000000
00000000000 -
t^91/13520015276784029625516656875949514214758686647690667779174173459
715367077155999476568528395475044942775116833676800819200000000000000
0000000 +
t^89/16507955160908461081216919262453619309839666236496541854913520707
833171034378509739399912570787600662729080382999756800000000000000000
000 -
t^87/21077572983795277172136005186993895952297837380613562123229725112
14654115727593174080683423236414793504734471782400000000000000000000 +
t^85/28171041143805502769494794422606115948005663433057420640510191275
2560026159795933451040286452340924018275123200000000000000000000 -
t^83/39455239697206586511897471180120610571436503407643446275224357528
36975156299662933487959194010377087090688000000000000000000000 +
t^81/57971260207473679858797342315781091054123572447316259587458650497
16390179693892056256184534249745940480000000000000000000 -
t^79/89461821307829752868514417153983165206980821677957190721386806322
7837990693501860533361810841010176000000000000000000 +
t^77/14518309202828586963407078408630828498374037922420835884678157468
8061991349156420080065207861248000000000000000000 -
t^75/24809140811395398091946477116594033660926243886570122837795894512
655842677572867409443815424000000000000000000 +
t^73/44701154615126843408912571381250511100768007002829050158190800923
70422104067183317016903680000000000000000 -
t^71/85047858856786231752116764423992601028858460812079623588643076338
```

8588680378079017697280000000000000000 +
t^69/1711224524281413113724683388812728390922705448935203693936480409232572797541406474240000000000000000 -
t^67/36471110918188685288249859096605464427167635314049524593701628500267962436943872000000000000000 +
t^65/82476505920824706667231703067854962521862585513454374929221231343889557749760000000000000000 -
t^63/1982608315404440064116146708361898137544773690227268628106279599612729753600000000000000 +
t^61/507580213877224798800856812176625227226004528988036003099405939480985600000000000000 -
t^59/1386831185456898357379390197203894063459028767726874325408212949401600000000000000 +
t^57/4052691950487721675568060190543232213498038479622660214518448128000000000000 -
t^55/12696403353658275925965100847566516959580321051449436762275840000000000000 +
t^53/4274883284060025564298013753389399649690343788366813724672000000000000 -
t^51/1551118753287382280224243016469303211063259720016986112000000000000 +
t^49/6082818640342675608722521633212953768875528313792102400000000000 -
t^47/2586232415111681806429643551536119799691976323891200000000000 +
t^45/1196222208654801945619631614956577150643837337600000000000 -
t^43/604152630633738356373551320685139975072645120000000000 +
t^41/33452526613163807108170062053440751665152000000000 -
t^39/20397882081197443358640281739902897356800000000 +
t^37/13763753091226345046315979581580902400000000 -
t^35/10333147966386144929666651337523200000000 +
t^33/8683317618811886495518194401280000000 -
t^31/8222838654177922817725562880000000 +
t^29/8841761993739701954543616000000 -
t^27/10888869450418352160768000000 + t^25/15511210043330985984000000 -
t^23/25852016738884976640000 + t^21/51090942171709440000 -
t^19/121645100408832000 + t^17/355687428096000 - t^15/1307674368000 +
t^13/6227020800 - t^11/39916800 + t^9/362880 - t^7/5040 + t^5/120 - t^3/6 + t

% en el desarrollo vemos una aproximación de los coeficientes, de las distintas potencias de t, por números racionales; se puede visualizar mejor:

>> vpa(ans,5) *% nos permite visualizar mejor el formato racional*

ans =

- 1.0715*10^(-156)*t^99 + 1.0396*10^(-152)*t^97 - 9.6806*10^(-149)*t^95 + 8.6447*10^(-145)*t^93 - 7.3964*10^(-141)*t^91 + 6.0577*10^(-137)*t^89 - 4.7444*10^(-133)*t^87 + 3.5497*10^(-129)*t^85 - 2.5345*10^(-125)*t^83 + 1.725*10^(-121)*t^81 - 1.1178*10^(-117)*t^79 + 6.8879*10^(-114)*t^77 - 4.0308*10^(-110)*t^75 + 2.2371*10^(-106)*t^73 - 1.1758*10^(-102)*t^71 + 5.8438*10^(-99)*t^69 - 2.7419*10^(-95)*t^67 + 1.2125*10^(-91)*t^65 - 5.0439*10^(-88)*t^63 + 1.9701*10^(-84)*t^61 - 7.2107*10^(-81)*t^59 + 2.4675*10^(-77)*t^57 - 7.8762*10^(-74)*t^55 + 2.3392*10^(-70)*t^53 - 6.447*10^(-67)*t^51 + 1.644*10^(-63)*t^49 - 3.8666*10^(-60)*t^47 + 8.3597*10^(-57)*t^45 - 1.6552*10^(-53)*t^43 + 2.9893*10^(-50)*t^41 - 4.9025*10^(-47)*t^39 + 7.2655*10^(-44)*t^37 - 9.6776*10^(-41)*t^35 + 1.1516*10^(-37)*t^33 - 1.2161*10^(-34)*t^31 + 1.131*10^(-31)*t^29 - 9.1837*10^(-29)*t^27 + 6.447*10^(-26)*t^25 - 3.8682*10^(-23)*t^23 + 1.9573*10^(-20)*t^21 - 8.2206*10^(-18)*t^19 + 2.8115*10^(-15)*t^17 - 7.6472*10^(-13)*t^15 + 1.6059*10^(-10)*t^13 - 2.5052*10^(-8)*t^11 + 2.7557*10^(-6)*t^9 - 0.00019841*t^7 + 0.0083333*t^5 - 0.16667*t^3 + t

>> taylor(u,10) *% 10 primeros términos del desarrollo*

ans =

t^9/362880 - t^7/5040 + t^5/120 - t^3/6 + t

% Podemos comprobar que efectivamente son los 10 primeros términos del desarrollo en serie de potencias de la función sin(t).
% Volviendo la ecuación de Riccati y'=t^2+y^2, buscar soluciones que pasen por (0,1), por ejemplo, nos permite identificar la constante C1 en la solución general obtenida

>> syms t

>> u=dsolve('Dy=t^2+y^2','y(0)=1') *% problema de Cauchy*

u =

-t*((-gamma(3/4)^2+pi)/gamma(3/4)^2*besselj(-3/4,1/2*t^2)+bessely(-3/4,1/2*t^2))/((gamma(3/4)^2+pi)/gamma(3/4)^2*besselj(1/4,1/2*t^2)+bessely(1/4,1/2*t^2))

>> taylor(u,5)

ans =

1/2*(2*(gamma(3/4)^2+pi)/gamma(3/4)/pi+2/pi*gamma(3/4))*gamma(3/4)+
1/2*(2*(gamma(3/4)^2+pi)/gamma(3/4)/pi+2/pi*gamma(3/4))*gamma(3/4)*t+
1/2*(2*(gamma(3/4)^2+pi)/gamma(3/4)/pi+2/pi*gamma(3/4))*gamma(3/4)*t^2+
1/2*(2/3/gamma(3/4)+2*(gamma(3/4)^2+pi)/gamma(3/4)/pi+2/pi*gamma(3/4))*
gamma(3/4)*t^3+1/2*(13/6*(gamma(3/4)^2+pi)/gamma(3/4)/pi+13/6/pi*gamma(3/4
)+1/6/gamma(3/4))*gamma(3/4)*t^4

% Se trata efectivamente de constantes (en términos de la función Gamma, ver help abajo) acompañando a las distintas potencias de t. Para determinar el valor aproximado de las constantes usamos "vpa"

>> vpa(ans,5)

ans =

1.0000+1.0000*t+1.0000*t^2+1.3333*t^3+1.1666*t^4

% utilizando la ED y la condición inicial, se pueden obtener todos los términos del desarrollo en serie de la solución: obtenemos los cinco primeros
y'=x^2+y^2, y(0)=1; implica y'(0)=1
y''=2x+2yy'; implica y''(0)= 2
y'''=2+2y'y'+2yy' ; implica y'''(0)=8
y''''= 4y''(y')+ 2y'y''+2yy''' ; implica que y''''(0)=16+4+8
*Así, y(x)= 1+t+t^2+4/3*t^3+28/24*t^4+…. comprobamos que coincide con el desarrollo en serie de Taylor de la solución que nos ha dado Matlab arriba*

>> vpa(1+t+t^2+4/3*t^3+28/24*t^4,5)

ans =

1.+t+t^2+1.3333*t^3+1.1667*t^4

>> help gamma *% la función Gamma se define en términos integrales*

```
GAMMA  Gamma function.
   Y = GAMMA(X) evaluates the gamma function for each element of X.
   X must be real.  The gamma function is defined as:

      gamma(x) = integral from 0 to inf of t^(x-1) exp(-t) dt.

   The gamma function interpolates the factorial function.  For
   integer n, gamma(n+1) = n! (n factorial) = prod(1:n).

   Class support for input X:
      float: double, single
```

% La ED y'=t^2+y^2 no se resuelve de manera adecuada con algunas versiones de Matlab posteriores a 2008, como comprobamos abajo.

>> dsolve('Dy=t^2+y^2') *% con Matlab 2011*

```
ans =

-((C6*besselj(1/4, t^2/2) + C7*bessely(1/4, t^2/2))/(2*t^(1/2)) +
t^(1/2)*(C6*(t*besselj(-3/4, t^2/2) - besselj(1/4, t^2/2)/(2*t)) + C7*(t*bessely(-3/4,
t^2/2) - bessely(1/4, t^2/2)/(2*t))))/(t^(1/2)*(C6*besselj(1/4, t^2/2) + C7*bessely(1/4,
t^2/2)))
```

% "dsolve", con Matlab de 2011, nos da dos constantes de integración, y deberíamos tener sólo una constante; además, si intentamos identificarlas imponiendo la condición inicial, nos da una advertencia y una solución en el campo complejo

>> dsolve('Dy=t^2+y^2','y(0)=1')

```
 Warning: Explicit solution could not be found; implicit solution returned.
> In dsolve at 168
Warning: The solutions are parametrized by the symbols:
z = solve([limit((4*t*gamma(3/4)*besselK(-3/4, (t^2*I)/2)*I -
2^(1/2)*4^(1/4)*C42*PI^(3/2)*t*besselI(-3/4,
(t^2*I)/2)*I)/(4*gamma(3/4)*besselK(1/4, (t^2*I)/2) +
2^(1/2)*4^(1/4)*C42*PI^(3/2)*besselI(1/4, (t^2*I)/2)), t = 0) = 1], C42, VectorFormat =
TRUE)

> In dsolve at 176

ans =

(t*gamma(3/4)*besselk(-3/4, (t^2*i)/2)*4*i - 2^(1/2)*4^(1/4)*pi^(3/2)*t*z*besseli(-
3/4, (t^2*i)/2)*i)/(4*gamma(3/4)*besselk(1/4, (t^2*i)/2) +
2^(1/2)*4^(1/4)*pi^(3/2)*z*besseli(1/4, (t^2*i)/2))
```

% como antes, los términos "gamma(3/4)" son constantes, mientras que las funciones de Bessel se definen a través de desarrollos en serie de potencias, pero aparece la " i" de la unidad imaginaria que no es necesaria, y no aparece en otras versiones de Matlab. Tampoco el desarrollo en serie de Taylor da un resultado satisfactorio

>> taylor(ans,5)

ans =

((pi^6*t*besseli(1/4, t^2*(i/2))^3*besseli(-3/4, t^2*(i/2))*(i/16))/(gamma(3/4)^4*besselk(1/4, (i*t^2)/2)^4) + (pi^6*t*besseli(1/4, t^2*(i/2))^4*besselk(-3/4, t^2*(i/2))*(i/16))/(gamma(3/4)^4*besselk(1/4, (i*t^2)/2)^5))*z^4 + ((2^(1/2)*4^(1/4)*pi^(9/2)*t*besseli(1/4, t^2*(i/2))^2*besseli(-3/4, t^2*(i/2))*(-i/16))/(gamma(3/4)^3*besselk(1/4, (i*t^2)/2)^3) + (2^(1/2)*4^(1/4)*pi^(9/2)*t*besseli(1/4, t^2*(i/2))^3*besselk(-3/4, t^2*(i/2))*(-i/16))/(gamma(3/4)^3*besselk(1/4, (i*t^2)/2)^4))*z^3 + ((pi^3*t*besseli(1/4, t^2*(i/2))*besseli(-3/4, t^2*(i/2))*(i/4))/(gamma(3/4)^2*besselk(1/4, (i*t^2)/2)^2) + (pi^3*t*besseli(1/4, t^2*(i/2))^2*besselk(-3/4, t^2*(i/2))*(i/4))/(gamma(3/4)^2*besselk(1/4, (i*t^2)/2)^3))*z^2 + ((2^(1/2)*4^(1/4)*pi^(3/2)*t*besseli(-3/4, t^2*(i/2))*(-i/4))/(gamma(3/4)*besselk(1/4, t^2*(i/2))) + (2^(1/2)*4^(1/4)*pi^(3/2)*t*besseli(1/4, t^2*(i/2))*besselk(-3/4, t^2*(i/2))*(-i/4))/(gamma(3/4)*besselk(1/4, (i*t^2)/2)^2))*z + (t*besselk(-3/4, t^2*(i/2))*i)/besselk(1/4, t^2*(i/2))

>> help besselj

```
besselj Bessel function of the first kind.
  J = besselj(NU,Z) is the Bessel function of the first kind, J_nu(Z).
  The order NU need not be an integer, but must be real.
  The argument Z can be complex.  The result is real where Z is positive.

  If NU and Z are arrays of the same size, the result is also that size.
  If either input is a scalar, it is expanded to the other input's size.
  If one input is a row vector and the other is a column vector, the
  result is a two-dimensional table of function values.

  J = besselj(NU,Z,0) is the same as besselj(NU,Z).

  J = besselj(NU,Z,1) scales J_nu(z) by exp(-abs(imag(z)))

  [J,IERR] = besselj(NU,Z) also returns an array of error flags.
      ierr = 1   Illegal arguments.
      ierr = 2   Overflow.  Return Inf.
      ierr = 3   Some loss of accuracy in argument reduction.
      ierr = 4   Complete loss of accuracy, z or nu too large.
      ierr = 5   No convergence.  Return NaN.

  Examples:

      besselj(3:9,(0:.2:10)') generates the entire table on page 398
      of Abramowitz and Stegun, Handbook of Mathematical Functions.

  besselj uses a MEX interface to a Fortran library by D. E. Amos.

  Class support for inputs NU and Z:
     float: double, single

  See also bessely, besseli, besselk, besselh.

  Overloaded methods:
     sym/besselj

  Reference page in Help browser
     doc besselj
```

*% **Observación:** distintos aspectos de las soluciones de esta ED (y'=y^2+t^2) serán objeto de estudio en los próximos cuadernos de curso (e.g., cuadernos 3 y 4)*

*% **Ejemplo (ejercicio 1.4, c):** la ED de Riccati y'=x^2+exp(x)y^2 no se resuelve con Matlab (versiones en 1999—2011), si bien, el mensaje "warning" es distinto dependiendo de la versión.*

```
>> dsolve('Dy=t^2+y^2*exp(t)') % con Matlab 2008

??? Error using ==> maple at 129
Error, invalid input: rhs received [diff(y(t),t) = t^2+y(t)^2*exp(t)], which is not valid for its 1st argument, expr

Error in ==> dsolve at 365
  RHS{j} = ...

ans =

[ empty sym ]
```

```
>> dsolve('Dy=t^2+exp(t)*y^2')  % con Matlab 2010

{Warning: Explicit solution could not be found.}
> In <a href="matlab:
opentoline('C:\MATLAB\R2010b\toolbox\symbolic\symbolic\dsolve.m',101,1)">dsolve at 101</a>

ans =

[ empty sym ]
```

```
>> dsolve('Dy=t^2+exp(t)*y^2')  % con Matlab 2011

Warning: Explicit solution could not be found.
> In dsolve at 161

ans =

[ empty sym ]
```

*% **Observación:** hay que reducir esta ED (y'=x^2+exp(x)y^2) a una ED de segundo orden y resolver con desarrollos en serie de potencias, lo cual se hace en el cuaderno sexto del curso.*

Modelos matemáticos relacionados con ED de primer orden

% Se trata de modelos considerados en la sección 1.6 del libro de apuntes. Utilizamos las versiones 2007 o 2008 de Matlab (algunas versiones posteriores pueden dar problemas, como los descritos antes en el cuaderno).

***% Modelo de caída libre de cuerpos (ejercicio 2)**: variables del problema*
t tiempo
v=v(t) velocidad
h=h(t) altura
h0, v0 altura y velocidad iniciales
h0-h(t) espacio recorrido
m masa del cuerpo
grav constante de la gravedad
k constante relacionada con el rozamiento

% Modelo no lineal: amortiguación proporcional a v^2

>> syms t m k v0 h0 grav v

>> V=dsolve('m*Dv=m*grav-k*v^2','v(0)=v0')

ans =

tanh((t*(m*grav*k)^(1/2)+m*atanh(k*v0/(m*grav*k)^(1/2)))/m)*(m*grav*k)^(1/2)/k

>> pretty(V)

```
                  1/2                  k v0
      t (m grav k)     + m atanh(-------------)
                                            1/2
                               (m grav k)                1/2
 tanh(----------------------------------------) (m grav k)
                        m
 -------------------------------------------------------------
                             k
```

% para encontrar h(t) se integra h=- int(V,t)

>> h=-int(V,t)

h =

1/2/k*m*log(tanh((t*(m*grav*k)^(1/2)+m*atanh(k*v0/(m*grav*k)^(1/2)))/m)-1)+1/2/k*m*log(1+tanh((t*(m*grav*k)^(1/2)+m*atanh(k*v0/(m*grav*k)^(1/2)))/m))

>> pretty(h)

```
                        1/2                  k v0
             t (m grav k)    + m atanh(-------------)
                                                1/2
                                      (m grav k)
     m log(tanh(-------------------------------------------) - 1)
                                m
  1/2 ------------------------------------------------------------
                              k

                                     1/2                  k v0
                          t (m grav k)    + m atanh(-------------)
                                                             1/2
                                                   (m grav k)
              m log(1 + tanh(-------------------------------------------))
                                             m
       + 1/2 ------------------------------------------------------------
                                           k
```

% la constante de integración, que no aparece (por defecto) en la integración, hay que determinarla imponiendo h(0)=h0

% Otra forma de encontrar la altura en función del tiempo es considerar la ecuación diferencial de segundo orden para h(t), y entonces, v(t)=- h'(t). Abajo la incógnita y(t) es la altura, D2y indica la derivada segunda

>> syms t m k v0 h0 grav v

>> altura=dsolve('m*D2y=-m*grav+k*(Dy)^2','y(0)=h0','Dy(0)=-v0')

altura =

1/2*(2*grav^(1/2)*k^(1/2)*t*m^2+m^(5/2)*log(4*grav*m^5*k^2/(exp(2/m^(1/2)*grav^(1/2)*k^(1/2)*t)*k*m^(5/2)*grav^(1/2)+exp(2/m^(1/2)*grav^(1/2)*k^(1/2)*t)*k^(3/2)*m^2*v0+m^(5/2)*grav^(1/2)*k-k^(3/2)*m^2*v0)^2)+2*m^(3/2)*k*h0)/m^(3/2)/k

>> pretty(altura)

```
    /                                                /
    |      1/2  1/2      2    5/2              5 2   /  |
1/2 |2 grav    k     t m  + m    log(4 grav m  k   /   |
    |                                             /    |
    \                                                \

                1/2  1/2
            grav    k    t     5/2     1/2
      exp(2 ---------------) k m    grav
                  1/2
                 m

                   1/2  1/2                                              \2
               grav    k    t   3/2  2       5/2     1/2      3/2  2    |
       + exp(2 ---------------) k    m  v0 + m    grav    k - k    m  v0|  )
                     1/2                                                |
                    m                                                   /

                    \
            3/2     |   /   3/2
       + 2 m    k h0|  /  (m    k)
                    | /
                    /
```

% Modelo lineal: amortiguación proporcional a v

>> altura_lineal=dsolve('m*D2y=-m*grav-k*Dy', 'y(0)=h0','Dy(0)=-v0')

altura_lineal =

m/k^2*exp(-k/m*t)*(k*v0-m*grav)-m/k*grav*t+(k^2*h0-m*k*v0+m^2*grav)/k^2

>> pretty(altura_lineal)

```
        k t
m exp(- ---)  (k v0 - m grav)                  2                2
         m                      m grav t   k  h0 - m k v0 + m  grav
---------------------------  -  --------  +  ------------------------
              2                    k                      2
             k                                           k
```

>> velocidad=-diff(altura_lineal,t)

velocidad =

1/k*exp(-k/m*t)*(k*v0-m*grav)+m/k*grav

>> pretty(velocidad)

```
            k t
   exp(- ---)  (k v0 - m grav)
            m                       m grav
   -------------------------- + ------
                k                     k
```

% Las constantes se determinan a partir de valores iniciales (velocidad inicial v0 y altura inicial h0), y si no se conocieran otros datos, como la constante de amortiguación k, se podrían tomar medidas experimentalmente, como la altura en distintos tiempos, para determinarlos.

% Modelos de crecimiento de poblaciones (ejercicio 3): *variables del problema*
t tiempo
n0 población inicial
ninf valor hacia el que tiende la solución
gam constante de proporcionalidad ("tasa de crecimiento")
y(t) población en el tiempo t
(ver también ejercicio 1 de la hoja 3 de problemas)

% Modelo Verhulst: solucion explícita

>> syms t n0 ninf gam

>> poblacion=dsolve('Dy=gam*y*(1-y/ninf)','y(0)=n0')

poblacion =

n0*ninf/(n0+exp(-gam*t)*ninf-exp(-gam*t)*n0)

>> pretty(poblacion)

```
                  n0 ninf
  ---------------------------------------
  n0 + exp(-gam t) ninf - exp(-gam t) n0
```

% Dado que se ha resuelto la ED, se pueden hacer las gráficas de las soluciones dependiendo de los valores de las constantes gam, ninf y n0. Dichas constantes se determinan a partir de datos reales de la población. Abajo, las gráficas de varias soluciones para distintas condiciones iniciales n0, y el conjunto de comandos que nos las proporcionan. Por simplicidad, de manera simbólica, tomamos gam=0.2 y ninf=2 (t en [0,10])

>> N=subs(subs(poblacion,ninf,2),gam,0.2) *% población para gam=0.2 y ninf=2*

N =

2*n0/(n0+2*exp(-1/5*t)-exp(-1/5*t)*n0)

>> ezplot(subs(N,n0,0),[0,10]) *% tomamos n0=0*

>> hold on

>> ezplot(subs(N,n0,0.5),[0,10]) *% tomamos n0=0.5*

>> ezplot(subs(N,n0,0.2),[0,10])

>> ezplot(subs(N,n0,0.5),[0,10])

>> ezplot(subs(N,n0,1.5),[0,10])

>> ezplot(subs(N,n0,2),[0,10])

>> ezplot(subs(N,n0,2.5),[0,10])

>> ezplot(subs(N,n0,1.25),[0,10])

>> ezplot(subs(N,n0,2.25),[0,10])

>> ezplot(subs(N,n0,3.25),[0,10])

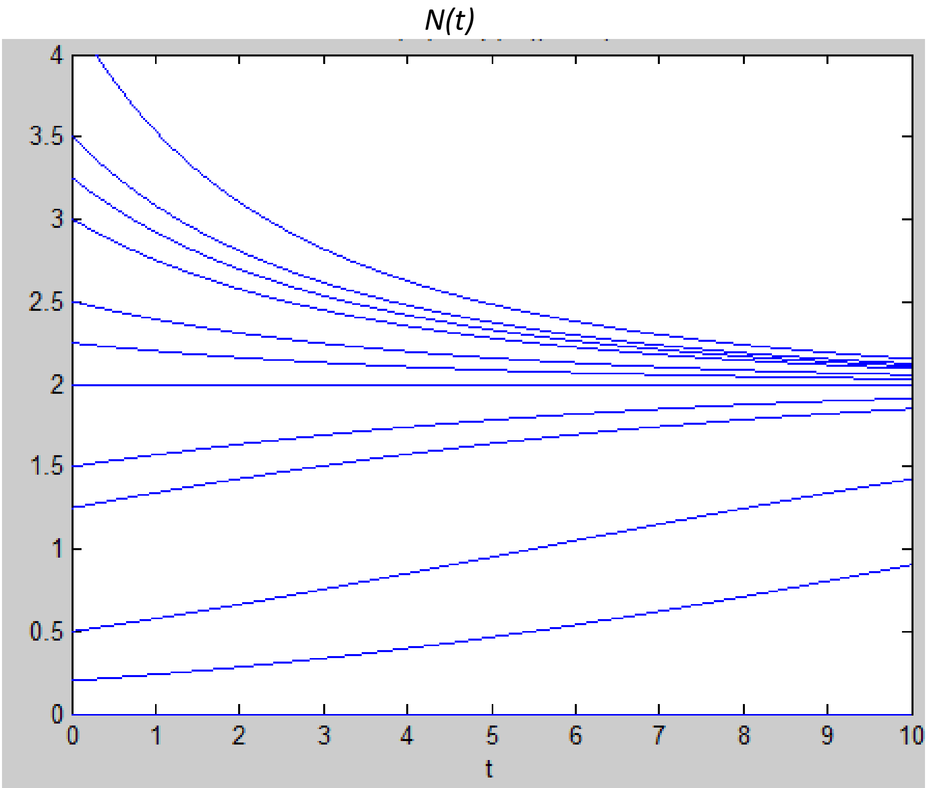
>> ezplot(subs(N,n0,4.25),[0,10])

>> ezplot(subs(N,n0,3),[0,10])

>> ezplot(subs(N,n0,3.5),[0,10])

>> axis([0 10 0 4]) *% ajustamos ejes*

>> hold off



% El campo de direcciones (entorno "dfield", ver el tercer cuaderno de curso) nos da una idea del crecimiento de las soluciones sin necesidad de resolver la ED (esto es, idea del "comportamiento cualitativo de las soluciones"). Utilizamos "dfield5" para representar las soluciones y añadir comentarios. Primero consideramos gam positivo y luego gam negativo. En la gráfica, N(t) representa el tamaño de la población.

% Tomamos gam=0.2, ninf=2, t en [0,10], y en [0,4].

>> dfield5

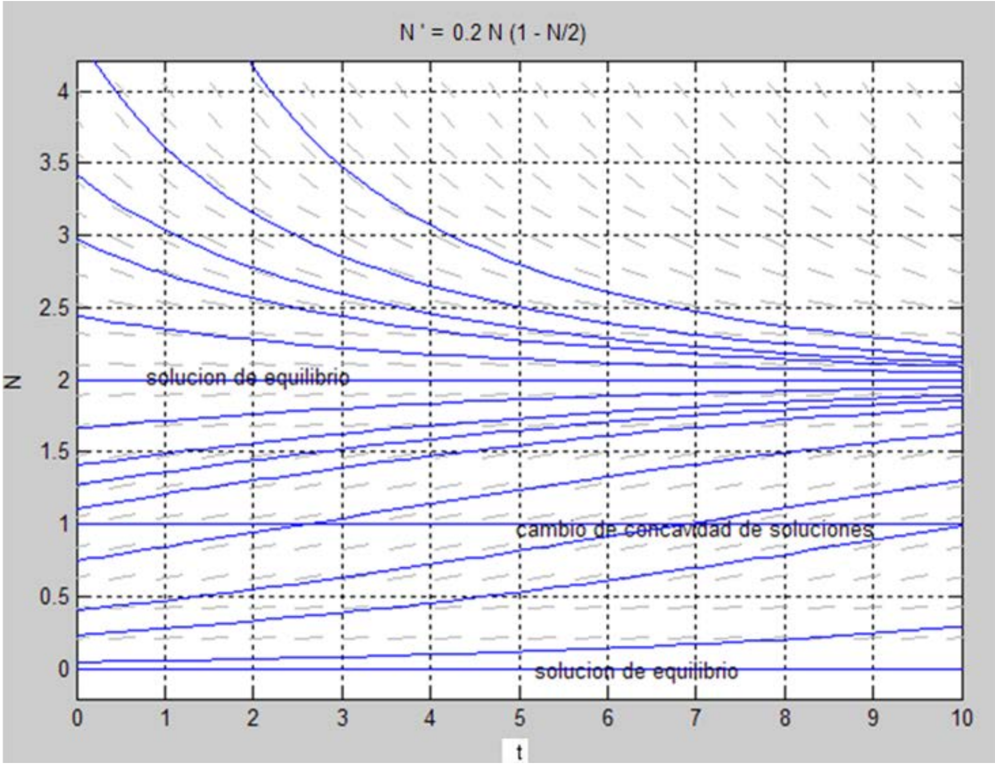
>> ezplot('N=1/2')

>> ezplot('N=1',[0,10])

% N=ninf/2 es la recta donde hay un cambio de concavidad de las curvas solución

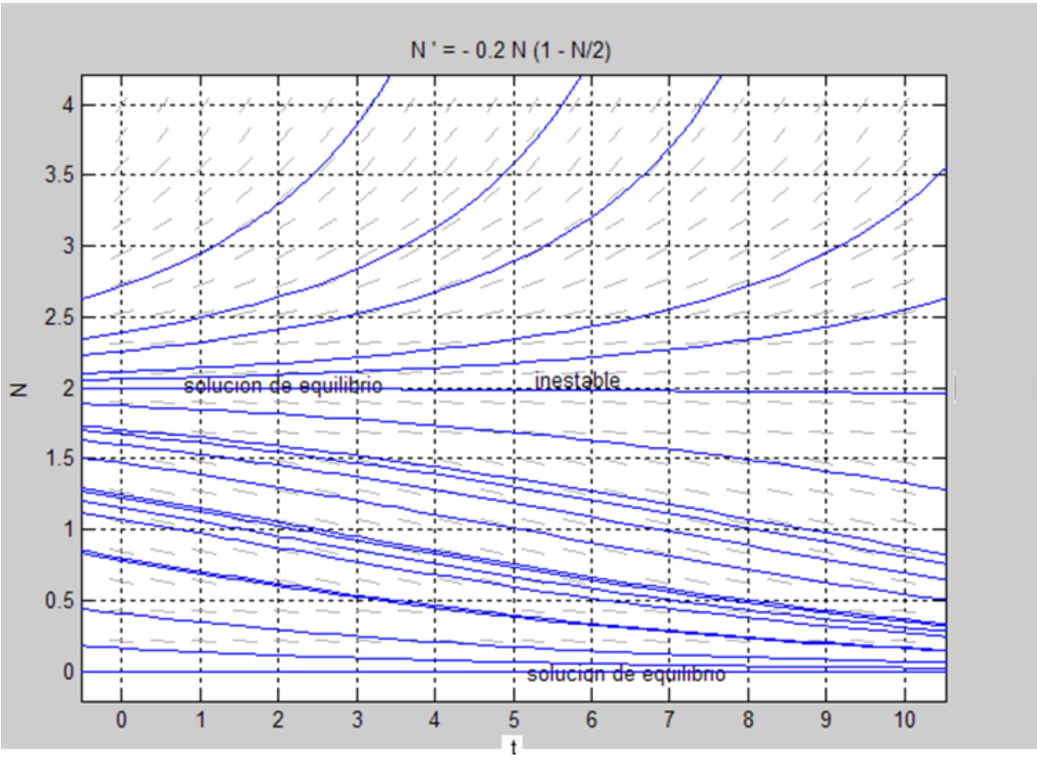
>> gtext('cambio de concavidad de soluciones')

>> gtext('solucion de equilibrio')



% Para gam >0, N=0 y N=ninf son soluciones de equilibrio (N'(t)=0). N=0 es una solución de equilibrio "inestable". N=ninf es una solución de equilibrio "asintóticamente estable".

% Abajo gam<0: tomamos gam=-0.2, ninf=2, t en [0,10], y en [0,4].



% Para gam<0, N=inf es una solución de equilibrio inestable y N=0 es asintóticamente estable. En cualquier caso, observamos que las gráficas de dos soluciones distintas no se cortan nunca.

*% **Observación:** Modelos diferenciales y problemas de estabilidad de soluciones para ED se segundo orden y sistemas autónomos se consideran en el cuaderno quinto del curso.*
