Resolución de ecuaciones y sistemas diferenciales lineales: Soluciones en series de potencias.

Capítulo 2 - Libro* + *Capitulo 3 - Libro* + *Hoja de problemas 6

Resumen de contenidos:

- **Fórmulas integrales para ED de segundo orden**
- **Ecuaciones de Airy, Bessel, Schrödinger,…**
- **Solución por desarrollos en serie de potencias**
- **Formulas integrales para sistemas lineales**

```
% Este cuaderno está dedicado a la aplicación de los métodos de reducción de orden y
  de variación de parámetros, y a la aproximación por desarrollos en serie de potencias
  de soluciones, tanto de ecuaciones, como de sistemas diferenciales lineales. De
  manera general, si se conoce una solución de la ED homogénea asociada, se puede
  reducir el orden. Si no, se busca la solución como una serie de potencias (ver teoremas
  en la sección 2.6 del libro de apuntes)

% Resolvemos ED de segundo orden aplicando  fórmulas. Si se puede, se comparan los
  resultados con los obtenidos usando "dsolve" y con el desarrollo en serie de potencias.
  Como dichos resultados pueden cambiar dependiendo de la versión de Matlab
  utilizada, comparamos algunos de ellos.

% Ejemplo: (2x+1)y''-4(x+1)+4y=0, con y1=x+1 .

>> syms x

>> y1=x+1

y1 =

x + 1

% Comprobamos que es solución de la ED y calculamos la solución general
```

```
>> (2*x+1)*diff(y1,2)-4*(x+1)*diff(y1)+4*y1 % y1 verifica (2x+1)y1''-4(x+1)y1'+4y1=0

ans =

0

 % De manera general, para la ED y''+p(x)y'+q(x)y=r(x): si se conoce una solución y_1(x)
   de y''+p(x)y'+q(x)y=0, se calcula otra solución y_2(x)=c(x)y_1(x), con c(x)=int((exp(
   int(-p(x) ))/y_1(x)^2), y, se calcula una solución particular de la ED dada
   yp=k1*y_1+k2*y_2, donde k1=int(r*y_2/w(y_1,y_2)), k2=int(r*y_1/w(y_1,y_2)), y
   w(y_1,y_2) indica el Wronskiano de y_1 e y_2.

% Calculamos y2= c(x)y1 tomando en la fórmula de c(x), p(x)= -4(x+1)/(2x+1)

>> exp(int((4*(x+1))/(2*x+1)))

ans =

exp(2*x + log(x + 1/2))

>> c= int(ans/(x+1)^2)

 c =

exp(2*x)/(2*(x + 1))

>> y2=y1*c

y2 =

exp(2*x)/2

% Comprobamos que esta y2 es también solución de (2x+1)y''-4(x+1)y'+4y=0.

>> (2*x+1)*diff(y2,2)-4*(x+1)*diff(y2)+4*y2

ans =

2*exp(2*x) + 2*exp(2*x)*(2*x + 1) - exp(2*x)*(4*x + 4)
```

>> simplify(ans)

ans =

0

% Así, las dos soluciones linealmente independientes son y1=x+1 e y2=exp(2x)/2. La solución general de (2x+1)y''-4(x+1)y'+4y=0 es y= A(x+1)+B(exp(2x)), con A y B constantes. Como vemos abajo, esta ED también la resuelvenlas versiones 2007--2011 de Matlab

>> syms t

>> dsolve('(2*t+1)*D2y-4*(t+1)*Dy+4*y=0') *% con Matlab 2011*

ans =

C2*(t + 1) + (C3*exp(2*t))/2

>> syms t

>> dsolve('(2*t+1)*D2y-4*(t+1)*Dy+4*y=0') *% con Matlab 2007 y 2008*

ans =

C1*t+C1+C2*exp(2*t)

% notamos que, aunque en distinto formato, se trata de la misma solución general.

% Utilizamos estas dos soluciones y1 e y2 calculadas para resolver la ED no homogénea (2x+1)y''-4(x+1)y'+4y=1. Para las fórmulas de yp, p=-4(x+1)/(2x+1) y r=1/(2x+1)

>> y1=x+1

y1 =

x + 1

>> y2=exp(2*x)

y2 =

exp(2*x)

>> w=y1*diff(y2)-y2*diff(y1) *% el Wronskiano*

w =

2*exp(2*x)*(x + 1) - exp(2*x)

>> r=1/(2*x+1)

r =

1/(2*x + 1)

>> k1=-int(r*y2/w)

k1 =

1/(4*x + 2)

>> k2=int(r*y1/w)

k2 =

-1/(4*exp(2*x)*(2*x + 1))

>> yp=k1*y1+k2*y2

yp =

(x + 1)/(4*x + 2) - 1/(4*(2*x + 1))

% yp es la solución particular y que se simplifica mucho en este caso:

>> pretty(yp) *% para visualizar yp mejor*

```
  x + 1        1
------- - -----------
4 x + 2   4 (2 x + 1)
```

>> simplify(yp)

ans =

1/4

% comprobación de solución particular

>> (2*x+1)*diff(yp,2)-4*(x+1)*diff(yp)+4*yp-1

ans =

(4*(x + 1))/(4*x + 2) - 1/(2*x + 1) - (2*x + 1)*(2/(2*x + 1)^3 + 8/(4*x + 2)^2 - (32*(x + 1))/(4*x + 2)^3) - (4*x + 4)*(1/(2*(2*x + 1)^2) + 1/(4*x + 2) - (4*(x + 1))/(4*x + 2)^2) - 1

>> simplify(ans)

ans =

0

% Solución general de (2x+1)y''-4(x+1)y'+4y=1: y=A(x+1)+B exp(2x)+1/4, o lo que es lo mismo, y=A(x+1)+B exp(2x)+ (x + 1)/(4x + 2) - 1/(4(2x + 1)). Utilizando dsolve para la resolución se encuentra también esta solución general:*

>> dsolve('(2*t+1)*D2y-4*(t+1)*Dy+4*y=1')

ans =

C20*(t + 1) - 1/(4*(2*t + 1)) + (t + 1)/(4*t + 2) + (C21*exp(2*t))/2

>> simplify(ans)

ans =

C20 *(t +1)+ (C21*exp(2*t))/2 + 1/4

*% Observamos que, si no se conoce una solución de y''+p(x)y'+q(x)=0 , de manera general, y''+p(x)y'+qx)y=r(x) no se sabe resolver. Un ejemplo en que dsolve no calcula la solución (**ejercicio 1.b**) es: y''+exp(t)y'+(1+t^2)y=1; sin embargo, como se observa abajo, el resultado en pantalla depende de la versión de Matlab:*

>> dsolve('D2y+exp(t)*Dy+(1+t^2)*y=1') *% Matlab 2011*

ans =

```
(exp(t/2)*(C13*whittakerM(-1/2, (- t^2 - 1)^(1/2), exp(t)) - C14*whittakerW(-1/2, (- t^2
  - 1)^(1/2), exp(t)) - int(exp(exp(t)/2 - t/2)*whittakerM(-1/2, (- t^2 - 1)^(1/2), exp(t)),
  t)*whittakerW(-1/2, (- t^2 - 1)^(1/2), exp(t)) + int(exp(exp(t)/2 - t/2)*whittakerW(-
  1/2, (- t^2 - 1)^(1/2), exp(t)), t)*whittakerM(-1/2, (- t^2 - 1)^(1/2),
  exp(t))))/(exp(exp(t)/2)*diff(whittakerM(-1/2, (- t^2 - 1)^(1/2), exp(t)),
  t)*whittakerW(-1/2, (- t^2 - 1)^(1/2), exp(t)) - exp(exp(t)/2)*diff(whittakerW(-1/2, (-
  t^2 - 1)^(1/2), exp(t)), t)*whittakerM(-1/2, (- t^2 - 1)^(1/2), exp(t)))
```

>>pretty(ans) *% visualizamos mejor*

```
        /                    /                                /                            \
   / t \ |                    |     / exp(t)   t \             |     / exp(t)   t \          |
exp| - | | C13 #2 - C14 #1 -  |  exp| ------ - - | #2 dt #1 +  |  exp| ------ - - | #1 dt #2 |
   \ 2 / \                   /      \   2      2 /            /      \   2      2 /          /
-------------------------------------------------------------------------------------------------
                   / exp(t) \                     / exp(t) \
                exp| ------ | #1 diff(#2, t) - exp| ------ | #2 diff(#1, t)
                   \   2    /                     \   2    /

where

                           2     1/2
   #1 = whittakerW(-1/2, (- t  - 1)   , exp(t))

                           2     1/2
   #2 = whittakerM(-1/2, (- t  - 1)   , exp(t))
```

>> help whittakerM *% este help no nos da información*

> whittakerM 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.

% Esto es, con la versión 2011 de Matlab, además de dejar la solución en términos de funciones que pueden resultar "desconocidas" en el curso, y de las que help no nos da información, hay una integral que no calcula:
*int(exp(exp(t)/2 - t/2)*whittakerW(-1/2, (- t^2 - 1)^(1/2), exp(t)), t).*

No obstante, se observa que las funciones que aparecen en la solución (Whittaker(..)) están bien definidas y la definición se puede encontrar, por ejemplo en
http://www.mathworks.es/es/help/symbolic/whittakerw.html
http://en.wikipedia.org/wiki/Whittaker_function

% Probamos con las versiones 2009, 2010 y 2008 de Matlab, viendo que no nos dan ninguna solución

>> dsolve('D2y+exp(t)*Dy+(1+t^2)*y=1')

{Warning: Explicit solution could not be found.}
> In <a href="matlab:
opentoline('D:\Programme\MATLAB\R2009a\toolbox\symbolic\dsolve.m',120,1)">dsolve at 120

ans =
[empty sym]

% dsolve, tampoco resuelve la homogénea

>> dsolve('D2y+exp(t)*Dy+(1+t^2)*y=0')

{???? Error using ==> mupadmex
Error in MuPAD command: argument must be of 'Type::Arithmetical' [exp]

>> dsolve('D2y+exp(t)*Dy+(1+t^2)*y=0') *% con Matlab 2010*

{???? Error using ==> mupadmex
Error in MuPAD command: argument must be of 'Type::Arithmetical' [exp]

Error in ==> <a href="matlab:
opentoline('C:\MATLAB\R2010b\toolbox\symbolic\symbolic\@sym\sym.m',2018,0)">sym.sym>sym.mupadmexnout at 2018
out = mupadmex(fcn,args{:});

>> dsolve('D2y+exp(t)*Dy+(1+t^2)*y=0') *% con Matlab 2008*

ans =

DESol({diff(_Y(t),$(t,2))+exp(t)*diff(_Y(t),t)+_Y(t)+_Y(t)*t^2},{_Y(t)})

% Abajo, otra ED y''+exp(t)y'+(1+t^2)y'=0, del mismo tipo, en la que no aparece y explícitamente tampoco la resuelve Matlab 2008: deja la solución en términos integrales. Ahora una solución es y1=1, pero parece que no se pueden calcular las integrales en el proceso de reducción de orden. Esto es algo que se ha observado ya con ED de primer orden.

>> dsolve('D2y+exp(t)*Dy+(1+t^2)*Dy=0')

ans =

C5 + C6*int(1/exp(t + exp(t) + t^3/3), t)

% Matlab 2011, lejos de resolver la misma ED, nos da como respuesta

>> dsolve('D2y+exp(t)*Dy+(1+t^2)*Dy=0')

ans =

C19 + C20*int(1/exp(t + exp(t) + t^3/3), t, IgnoreAnalyticConstraints)

*% **La ecuación de Airy** (**ejercicio 4**) y''-xy=0: es un ejemplo de ecuación donde no se conoce una solución expresada en términos de funciones elementales (como las funciones trignométricas, exponenciales, polinómicas, etc.)*

>> dsolve('D2y-t*y=0') *% con Matlab 2011*

ans =

C21*airyAi(t, 0) + C22*airyBi(t, 0)

>> dsolve('D2y-t*y=0') *% con Matlab 2008*

ans =

C1*AiryAi(t)+C2*AiryBi(t)

% En ambos casos tenemos la solución en términos de las conocidas funciones de Airy (se determinan por su desarrollo en serie de potencias) . Vemos que la forma de expresar dichas funciones depende de la versión de Matlab

>> help airy *% versiones 2008 y 2011*

```
airy   Airy functions.

   W = airy(Z) is the Airy function, Ai(z), of the elements of Z.
   W = airy(0,Z) is the same as airy(Z).
   W = airy(1,Z) is the derivative, Ai'(z).
   W = airy(2,Z) is the Airy function of the second kind, Bi(z).
   W = airy(3,Z) is the derivative, Bi'(z).
   If the argument Z is an array, the result is the same size.

   [W,IERR] = airy(K,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 too large.
      ierr = 5   No convergence.  Return NaN.

   The relationship between the Airy and modified Bessel functions is:

      Ai(z) = 1/pi*sqrt(z/3)*K_1/3(zeta)
      Bi(z) = sqrt(z/3)*(I_-1/3(zeta)+I_1/3(zeta))
      where zeta = 2/3*z^(3/2)

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

   See also besselj, bessely, besseli, besselk.

   Reference page in Help browser
      doc airy
```

>> help airyAi *% versiones 2008 y 2011*

```
airyAi 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.
```

>> mhelp airyAi *% version 2011 de Matlab*

```
Error using mhelp (line 24)
The MHELP command is not available.
```

>> mhelp AiryAi *% version 2008 de Matlab*

AiryAi, AiryBi - The Airy Ai and Bi wave functions

Calling Sequence
AiryAi(x)

AiryBi(x)

AiryAi(n, x)

AiryBi(n, x)

Parameters
n - algebraic expression (the order or index)

x - algebraic expression (the argument)

Description
- The Airy wave functions (Definition/Airy_function) AiryAi and AiryBi are linearly independent solutions for w in the equation w''- z*w=0. Specifically,

AiryAi(z) = c1*0F1(; 2/3; z^3/9) - c2*z*0F1(; 4/3; z^3/9)
AiryBi(z) = 3^(1/2) * [c1*0F1(; 2/3; z^3/9) + c2*z*0F1(; 4/3; z^3/9)]

where 0F1 is the generalized hypergeometric function, c1 = AiryAi(0) and c2 = -AiryAi'(0).

- The two argument forms are used to represent the derivatives, so AiryAi(1, x) = D(AiryAi)(x) and AiryBi(1, x) = D(AiryBi)(x). Note that all higher derivatives can be written in terms of the 0'th and 1st derivatives.

Note also that AiryAi(3, x^2) is the 3rd derivative of AiryAi(x) evaluated at x^2, and not the 3rd derivative of AiryAi(x^2).

- The Airy functions are related to Bessel functions of order n/3 for n=-2,-1,1,2 (see the examples below).

Examples
> AiryAi(0);

$$\frac{1/3}{3}\,\frac{3}{\mathrm{GAMMA}(2/3)}$$

> AiryBi(0);

$$\frac{5/6}{3}\,\frac{3}{\mathrm{GAMMA}(2/3)}$$

> AiryAi(1.23);

0.1021992656

> AiryBi(-3.45+2.75*I);

-16.85910551 - 32.61659997 I

> AiryAi(1,x);

```
                          AiryAi(1, x)

> AiryBi(2,x);

                          x AiryBi(x)

> convert(AiryAi(x), Bessel);

                          3 1/2                    3 1/2    3 1/3
       -x BesselI(1/3, 2/3 (x )   ) + BesselI(-1/3, 2/3 (x )   ) (x )
    1/3 ------------------------------------------------------------
                                3 1/6
                               (x )

> convert(AiryBi(1,x), Bessel);

      1/2   2                  3 1/2      3 2/3                  3 1/2
     3    (x  BesselI(2/3, 2/3 (x )   ) + (x )    BesselI(-2/3, 2/3 (x )   ))
  1/3 ------------------------------------------------------------------------
                                3 1/3
                               (x )

> diff(AiryAi(sin(x)), x);

                       cos(x) AiryAi(1, sin(x))

> diff(AiryBi(n,x),x);

                         AiryBi(n + 1, x)

> (D@@5)(AiryBi);

                                  2
                  z -> 4 z AiryBi(z) + z  AiryBi(1, z)

> series(AiryAi(x),x,4);

            1/3          1/6                  1/3
           3            3   GAMMA(2/3)          3       3     4
     1/3 ---------- - 1/2 ---------- x + 1/18 ---------- x  + O(x )
        GAMMA(2/3)           Pi                GAMMA(2/3)


See Also
inifcn, convert[Airy], AiryZeros, Bessel, convert[Bessel]
```

% La solución general de la ecuación de Airy está dada (ver Sección 2.6 del libro de apuntes) por el desarrollo en serie de potencia

$$y = a_0(1+\frac{1}{3.2}x^3+\cdots+\frac{1}{3n(3n-1).(3n-3)(3n-4).\cdots.(6.5)(3.2)}x^{3n}+\cdots)+$$

$$+a_1(x+\frac{1}{4.3}x^4+\cdots+\frac{1}{(3n+1)(3n).(3n-2)(3n-3).\cdots.(7.6)(4.3)}x^{3n+1}+\cdots)$$

*% donde a_0= y(0) y a_1=y'(0). La solución que nos da Matlab en términos de las funciones de Airy Ai y Bi, y(t)= C1*AiryAi(t)+C2*AiryBi(t), contiene constantes que no tienen por qué ser a_0 y a_1, es decir, de manera general no son los datos iniciales. También, de manera general, las dos funciones que definen los desarrollos encontrados arriba (buscando y=a_0+a_1x+a_2x^2+a_3x^4+….a_nx^n+….), son combinación lineal de éstas dos funciones de Airy ("de primera y segundo clase"), y al revés Ai y Bi son combinación lineal de las series de potencias encontradas arriba. Esto que acabamos de mencionar se comprueba, por ejemplo, imponiendo distintas condiciones iniciales en la ecuación de Airy*

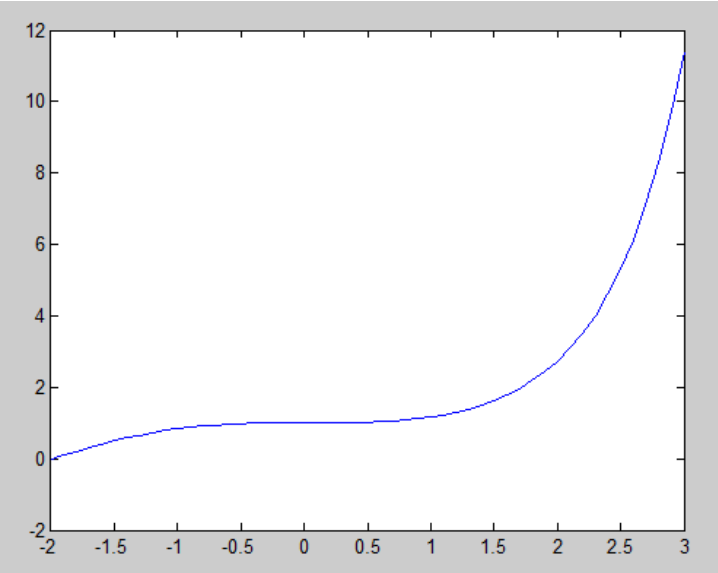
% La serie de potencias que acompaña a a_0 (como consecuencia del teorema de existencia y unicidad de solución para un problema de Cauchy) es para Matlab

```
>> dsolve('D2y-t*y=0','y(0)=1','Dy(0)=0')
```

```
ans =

1/2*3^(2/3)*gamma(2/3)*AiryAi(t)+1/2*3^(1/6)*gamma(2/3)*AiryBi(t)
```

% para hacer la gráfica utilizamos plot (ezplot da error)

```
>> dt=-2:0.1:3;

>> da=subs(ans,t,dt);

>> dA=double(da);

>> plot(dt,dA)
```



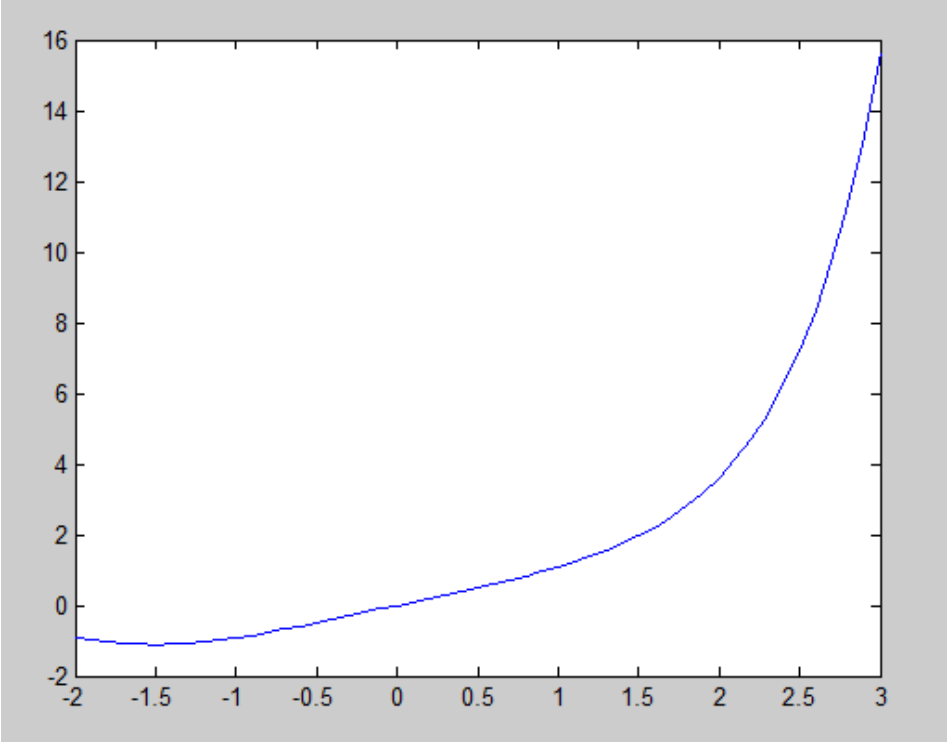
% La serie de potencias que acompaña a a_1 es para Matlab

>> dsolve('D2y-t*y=0','y(0)=0','Dy(0)=1')

ans =

-1/3*3^(5/6)*pi/gamma(2/3)*AiryAi(t)+1/3*3^(1/3)*pi/gamma(2/3)*AiryBi(t)

% Repitiendo el conjunto de comandos de arriba, con plot, se tiene la gráfica



% Estas gráficas se pueden comparar con las gráficas de las funciones de Airy Ai y Bi en el mismo intervalo. Por ejemplo, hacemos la gráfica de Ai en (-2,3):

>> dsolve('D2y-t*y=0')

ans =

*C1*AiryAi(t)+C2*AiryBi(t)*

>> syms C1 C2

>> dsolve('D2y-t*y=0')

ans =

C1*AiryAi(t)+C2*AiryBi(t)

>> subs(ans,C1,1)

ans =

AiryAi(t)+C2*AiryBi(t)

>> subs(ans,C2,0)

ans =

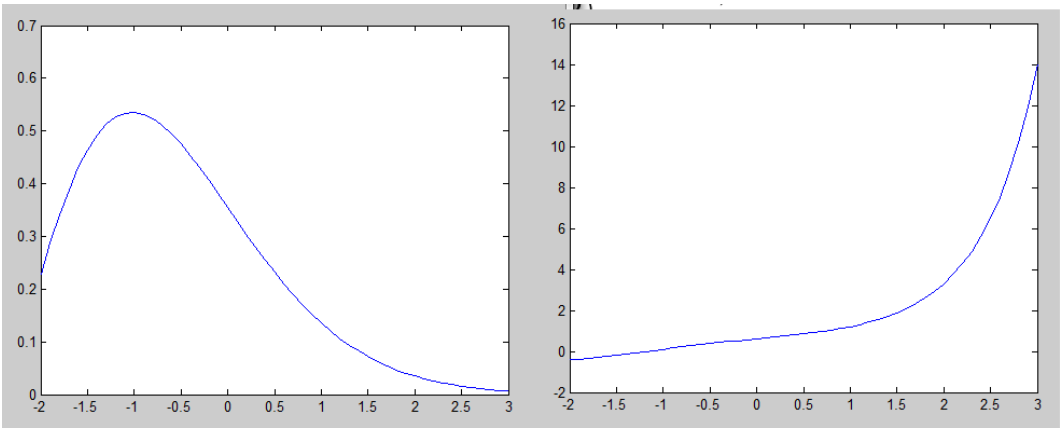
AiryAi(t)

>> da=subs(ans,t,dt);

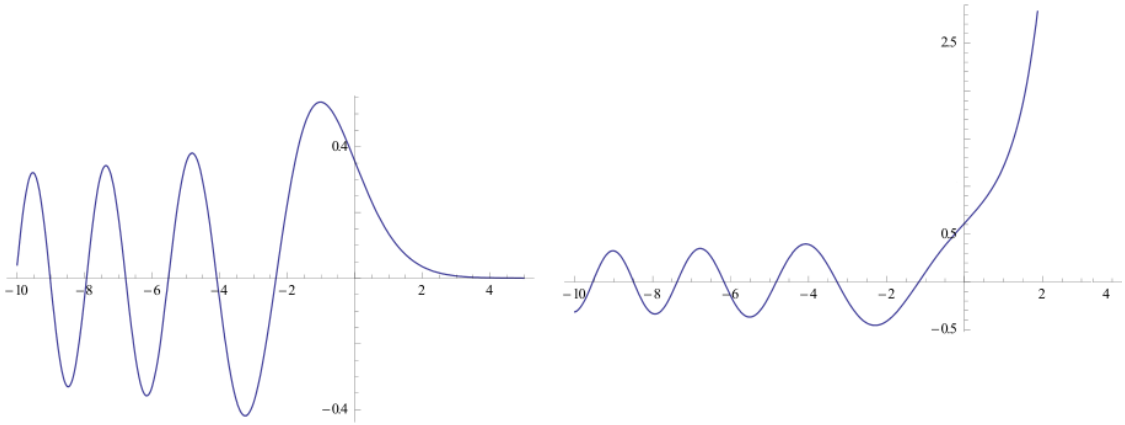
>> dA=double(da);

>> plot(dt,dA)

% Bi se dibuja repitiendo los pasos de arriba con C1=0 y C2=1. En las gráficas de abajo a la izquierda tenemos Ai, y a la derecha Bi, en el intervalo (-2,3)



% y, lo mismo, en el intervalo (-10,4)



% Observamos que las constantes que aparecen en ambas soluciones, obtenidas con dsolve (para a_0=1, a_1=0 y a_0=0 y a_1=1 repectivamente), están expresadas en términos de la función Gamma:

>> help gamma

```
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

   See also gammaln, gammainc, psi.

   Overloaded methods:
      sym/gamma

   Reference page in Help browser
      doc gamma
```

% Nota: Estas ecuaciones de Airy, y otras como las de Legendre, Bessel, Schrödinger, etc., son muy estudiadas por su interés en diversos modelos de la ciencia y la técnica, así como por las propiedades de las funciones con este nombre (ver cuaderno 9). Además, se pueden definir todos los términos del desarrollo en serie de manera recursiva en una sola fórmula, lo cual no pasa con otras ecuaciones diferenciales. Dependiendo de la versión que se utilice, Matlab devuelve la solución de una u otra

forma, y es conveniente buscar ayuda, bien la que proporciona Matlab, Wikipedia u otras para definiciones o propiedades. Por ejemplo, ver para las funciones de Airy:

http://www.mathworks.es/es/help/symbolic/airy.html
http://en.wikipedia.org/wiki/Airy_function
http://es.m.wikipedia.org/wiki/Funci%C3%B3n_de_Airy

También, hay que tener en cuenta que las variables que utiliza Matlab al devolvernos la solución pueden ser complejas (¡aparece la unidad imaginaria!).

% La teoría para resolver estas ecuaciones de Airy, y''+exp(x)y'+(1+x^2)y'=0, u otras como y''+h(x)y=0 con h(x) analítica en 0, se encuentra en la Sección 2.6 del libro de apuntes (Teorema 7). De manera general se busca solución de la forma
y= a_0+a_1x+a_2x^2+a_3x^4+….a_nx^n+….
donde los a_n se van obteniendo recursivamente en términos de a_0 y a_1, aunque, a veces, es difícil determinar fórmulas generales para soluciones con todos los términos del desarrollo, de aquí el interés de las funciones Matlab soluschroedinger.m y desapotenciasn.m , que analizaremos más adelante, para aproximar las soluciones.

***% La ecuación de Bessel:** x^2y''+xy'+(mu^2-x^2)y=0, con mu constante (**ejercicio 5**).*
Matlab nos da las soluciones de esta ecuación, para cualquier valor de mu, en términos de las funciones de Bessel de primera o segunda especie (las unas acotadas en el 0, mientras que las otras presentan comportamientos singulares, conteniendo términos con potencias de x de exponente negativo o términos logarítmicos). La forma de buscar la solución es y
y= x^r (a_0+a_1x+a_2x^2+a_3x^4+….a_nx^n+….)
para algún r a determinar (ver Teorema 8 del libro de apuntes): deben ser raíces del polinomio indicial r^2-mu^2. Analizamos ejemplos con distintas raíces

% Para mu=0 las dos raíces coinciden

```
>> dsolve('t^2*D2y+t*Dy+(t^2)*y=0')

ans =

C15*besselj(0, t) + C16*bessely(0, t)
```

% Las soluciones besselj y bessely son las funciones de Bessel de orden 0, de primera y segunda especie, respectivamente.

% Para mu=2, las raíces del polinomio inicial difieren en un número entero

>> dsolve('t^2*D2y+t*Dy+(t^2-4)*y=0')

ans =

C18*besselj(2, t) + C19*bessely(2, t)

% Para mu^2=4/3

>> dsolve('t^2*D2y+t*Dy+(t^2-4/3)*y=0')

ans =

C21*besselj(-(2*3^(1/2))/3, t) + C22*bessely(-(2*3^(1/2))/3, t)

>> dsolve('t^2*D2y+t*Dy+(t^2-11/3)*y=0')

ans =

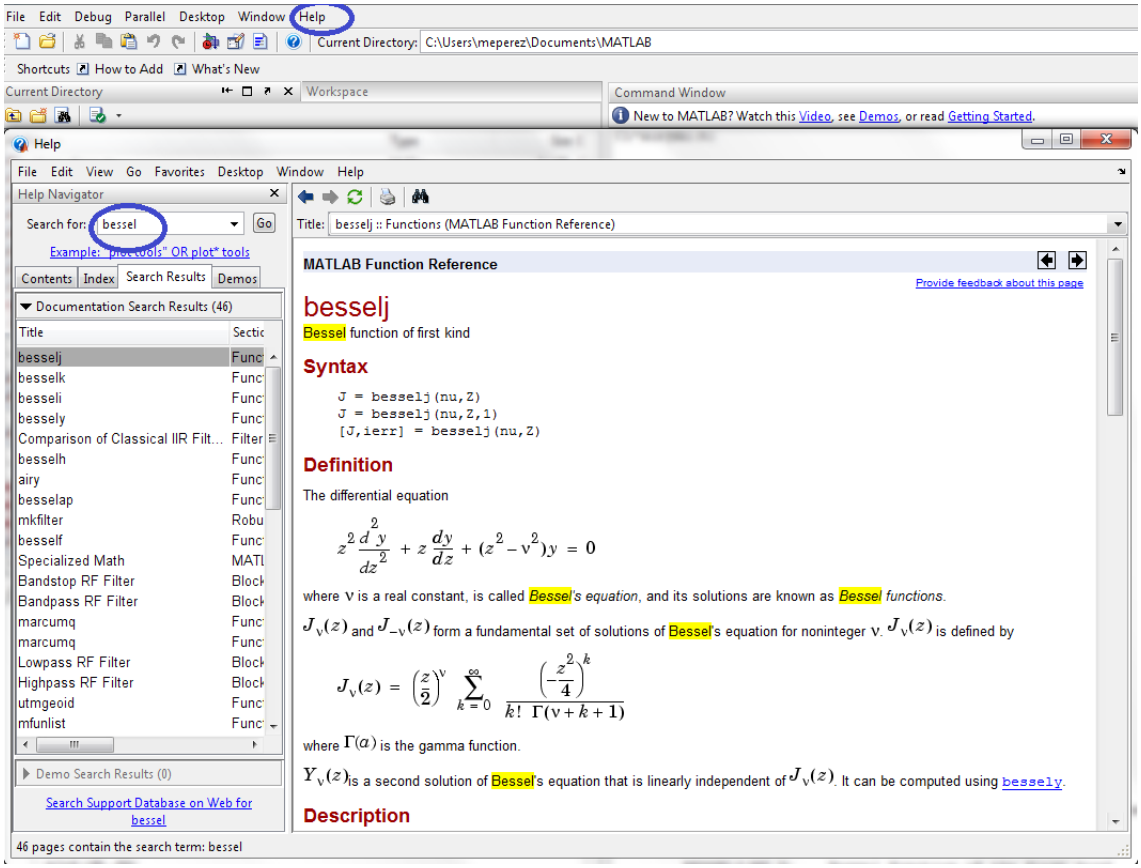
C24*besselj(-(3^(1/2)*11^(1/2))/3, t) + C25*bessely(-(3^(1/2)*11^(1/2))/3, t)

% Notemos que aunque Matlab nos da las soluciones en términos de las funciones de Bessel de primera y segunda especie, J_mu= Besselj(mu,.) y Y_mu=Bessely(mu,.), para cualquier mu, que no sea un natural, todas las soluciones se tienen como combinación lineal de las funciones J_mu y J_{-mu}:

$$Y_\nu(z) = \frac{J_\nu(z)\cos(\nu\pi) - J_{-\nu}(z)}{\sin(\nu\pi)}$$

% help en el entorno Matlab, o búsquedas en internet, nos proporcionan las definiciones precisas de todas las funciones de Bessel, como la función de arriba, y ésta de abajo

$$J_\nu(z) = \left(\frac{z}{2}\right)^\nu \sum_{k=0}^{\infty} \frac{\left(-\frac{z^2}{4}\right)^k}{k!\ \Gamma(\nu+k+1)}$$



http://www.mathworks.es/es/help/matlab/ref/besselj.html
http://www.mathworks.es/es/help/matlab/ref/besselk.html
http://en.wikipedia.org/wiki/Bessel_function

% Finalmente, observamos que en el cuaderno 9 del curso, se muestra el interés de estas funciones, por ejemplo, en modelos de vibraciones.

*% Hemos visto que Matlab no siempre nos expresa la solución de la ED en términos de funciones conocidas. No obstante, dado el carácter analítico de los coeficientes, en muchas ED, se sabe que existe solución que se puede expresar como un desarrollo en serie de potencias, obteniendo los términos de dicho desarrollo recursivamente a partir de los datos iniciales. Esto se puede obtener con pequeñas modificaciones de las funciones "desapotenciasn.m" y "soluchrodinger.m": esto es, tantos términos de dicho desarrollo como queramos. Estas funciones contienen instrucciones para aproximar la solución de las ecuaciones (**ejercicios 1--3**): y''+exp(x)y' +(1+x^2)y=0, y, y''+(3-x^2) y=0 respectivamente.*

% En ambas ED se busca y = a0+a1x+a2x^2+a3x^4+….anx^n+…. Se calculan los primeros términos ai en función de a0 y a1 hasta llegar a una fórmula recursiva que nos permita calcular an en función de los ai anteriores; fórmula que varía para cada ecuación.

% La función desapotenciasn.m lee los datos iniciales y(0) e y'(0), o lo que es lo mismo los términos a0 y a1 del desarrollo en serie a0+a1x+a2x^2+a3x^3+….+anx^n+… de la solución, lee el número N de términos que pedimos de dicho desarrollo, y nos proporciona estos términos junto con una gráfica de la aproximación de la solución del problema de Cauchy
y''+exp(x)y' +(1+x^2)y=0, y(0)=a0, y'(0)=a1

Las fórmulas recursivas para obtener los N términos:

```
a(1)=a1;
a(2)=-(a(1)+a0)/2;
c(1)=a(1)+2*a(2);
a(3)=-(a(1)+c(1))/6;
c(2)=3*a(3)+2*a(2)+a(1)/2;
a(4)=-(a0+a(2)+c(2))/12;
for i=3:N-2
   z=0;
   for j=1:i
      z=z+a(i+1-j)*(i-j+1)/factorial(j);
   end
   c(i)=z+a(i+1)*(i+1);
   a(i+2)=-((a(i-2)+a(i)+c(i))/((i+2)*(i+1)));
end
```

% y la aproximación:

```
syms x;
h=a0;
for i=1:N
   h=h+a(i)*x^i;
end
```

>> type desapotenciasn

>> desapotenciasn(1,0,10) *% y(0)=a0=1, y(1)=a1=0, N=10*

nterminosdesarrollo =

- (3497398567766801*x^10)/147573952589676412928 + (1436068452607735*x^9)/4611686018427387904 - (41*x^8)/40320 - x^7/168 + (11*x^6)/720 - x^5/120 + x^3/6 - x^2/2 + 1

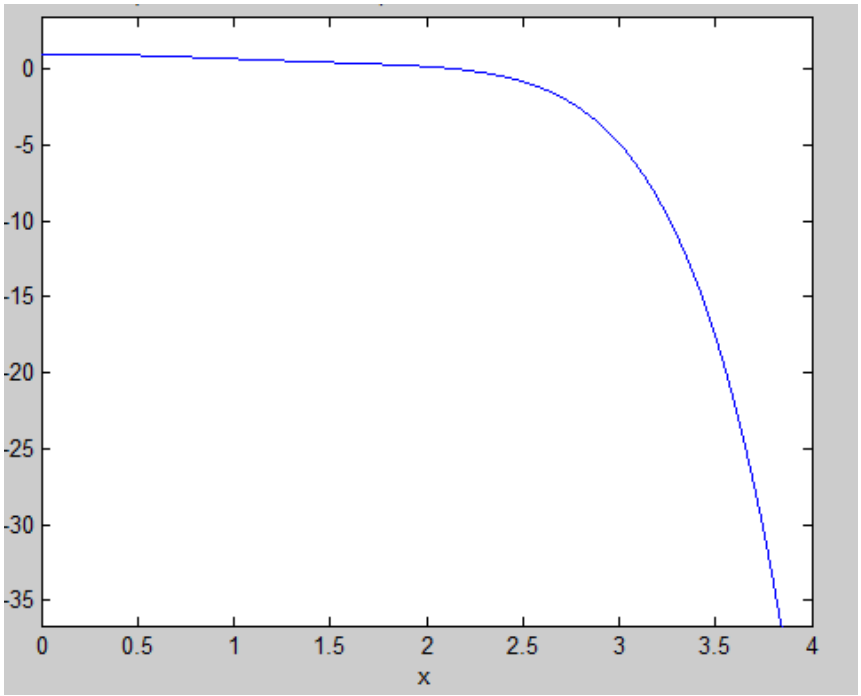
% para no ver estas expresiones "raras", largas (cocientes de números), que Matlab da por defecto, podemos pedir que nos de unas cifras decimales de lo que se pueda convertir a número en la expresión:

>> syms x

>> vpa((1436068452607735*x^9)/4611686018427387904 - (3497398567766801*x^10)/147573952589676412928 - (41*x^8)/40320 - x^7/168 + (11*x^6)/720 - x^5/120 + x^3/6 - x^2/2 + 1,5)

ans =

0.0003114*x^9 - 0.000023699*x^10 - 0.0010169*x^8 - 0.0059524*x^7 + 0.015278*x^6 - 0.0083333*x^5 + 0.16667*x^3 - 0.5*x^2 + 1.0



% a continuación, pedimos 100 términos, 1000 términos y comparamos gráficas

>>gtext('n=10')

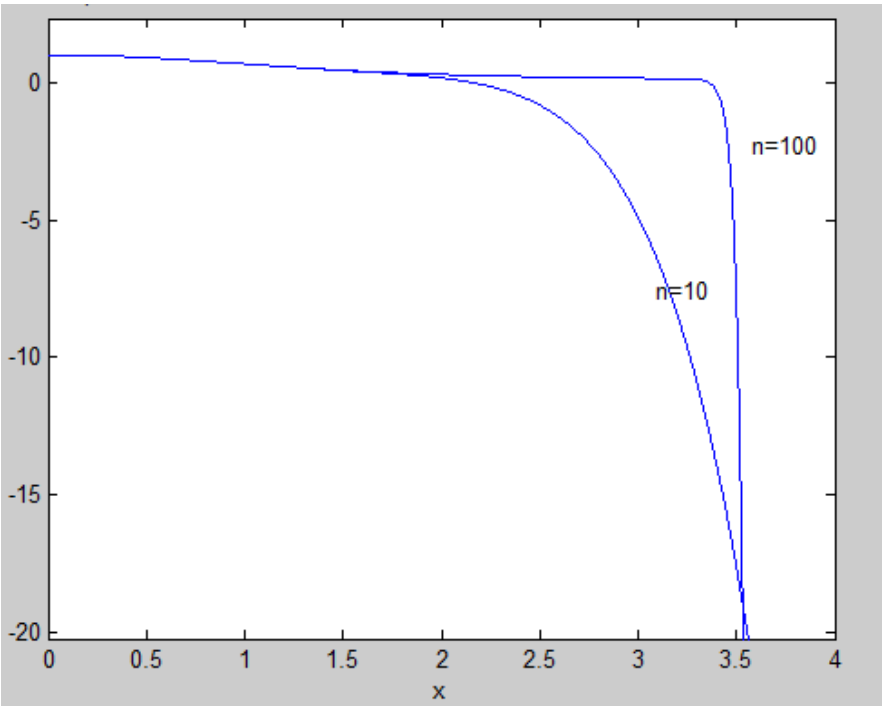
>> hold on

>> desapotenciasn(1,0,100)

nterminosdesarrollo =

(2929867606776523*x^100)/1725436586697640946858688965569256363112777243042596638790631055949824 - ……
……

- (41*x^8)/40320 - x^7/168 + (11*x^6)/720 - x^5/120 + x^3/6 - x^2/2 + 1



% los puntos suspensivos indican los términos muy largos, que evitamos escribir, como se ve abajo. Podríamos introducir dentro del programa desapotenciasn.m comandos del tipo "vpa" o "pretty", para visualizar mejor.

>> desapotenciasn(1,0,100)

nterminosdesarrollo =

2929867606776523/172543658669764094685868896556925636311277724304259663879063105594982 4*x^100 - 3281899462594259/1380349269358112757486951172455405090490221794434077311032504844759859 2*x^99 - 3238645589981309/1078397866686025591786680603480785226945485776901622899244144409968 64*x^98 - 2545696389300915/1347997333357531989733350754350981533681857221127028624055180512460 8*x^97 - 2388352769923891/3369993333938299743333768858774538342046430528175715601379512811 52*x^96 - 292563368113419/2106245833371143733958360553673408646377901908010982225086219550 72*x^95 + 4861086558842459/1684996666696914987166688442938726917102321526408785780068975640576*x^94 + 8685637643762195/2106245833371143733958360553673408646377901908010982225086219550 72*x^93 + 5628753957192893/2632807291713929667447950692091760807972377385013727781357774438 4*x^92 + 8886830443741241/1316403645856964833723975346045880403986188692506863890678887219 2*x^91 + 5026424766833557/6582018229284824168619876730229402019930943462534319453394436096*x^90 - 2541418035054153/4113761393303015105387422956393376262456839664083949658371522 56*x^89 - 1312556466800919/2571100870814384440867139347745860164035524790052468536482201 6*x^88 - 351992157508443/1606938044258990275541962092341162602522202993782792835301376*x^87 - 6980447090927247/1285550435407192220433569673872930082017762395026234268241100 8*x^86 + 1760648588994951/1285550435407192220433569673872930082017762395026234268241100 8*x^85 + 7636427843186895/8034690221294951377709810461705813012611014968913964176506 88*x^84 + 5667466919270665/1004336277661868922213726307713226626576376871114245522063 36*x^83 + 4934524308364687/2510840694154672305534315769283066566440942177785613805158 4*x^82 + 7915429490867233/2510840694154672305534315769283066566440942177785613

051584*x^81 -
3632411696307393/313855086769334038191789471160383320805117772223201725
6448*x^80 -
2347292390874129/196159429230833773869868419475239575503198607639501078528*x^79 -
5352594563725525/980797146154168869349342097376197877515993038197505392
64*x^78 -
7089038371689529/490398573077084434674671048688098938757996519098752696
32*x^77 -
2880069501277485/980797146154168869349342097376197877515993038197505392
64*x^76 +
782313077713907/383123885216472214589586756787577295904684780545900544*x^75 +
4848959025958497/383123885216472214589586756787577295904684780545900544*x^74 +
1061736973064247/239452428260295134118491722992235809940427987841187848*x^73 +
3428078643085037/478904856520590268236983445984471619880855975682375688*x^72 -
5858722968519833/239452428260295134118491722992235809940427987841187848*x^71 -
7522981371067771/299315535325368917648114653740294762425534984801484 8*x^70 -
2074632016134175/187072209578355573530071658587684226515959365500928*x^69 -
1284799878647567/467680523945888933825179146469210566289898413752328*x^68 +
5920568141101961/149657767662684458824057326870147381212767492400742 4*x^67 +
311601404138195/730750818665451459101842416358141509827966271488*x^66 +
7018416937576999/292300327466180583640736966543256603931186508595 2*x^65 +
2772464742657809/365375409332725729550921208179070754913983135744*x^64 +
6330316591584243/730750818665451459101842416358141509827966271488*x^63 -
5461032802120115/913438523331814323877303020447676887284957839 36*x^62 -
659888150930963/142724769270595988105828596944949513638274662 4*x^61 -
5012247907651601/285449538541191976211657193889899027276549324 8*x^60 -
591382963248101/178405961588244985132285746181186892047843328*x^59 +
4398629961963473/713623846352979940529142984724747568191373312*x^58 +
448733357877013/557518629963265578538392956816209037649510 4*x^57 +
8005272764200025/223007451985306231415357182726483615059804 16*x^56 +

2403504198754263/27875931498163278926919647840810451882475552*x^55 -
2302101368036973/11150372599265311570767859136324180752990208*x^54 -
8956917647124565/69689828745408197317299119602026129706188 8*x^53 -
90680557328707/13611294676837538538534984297270728458 24*x^52 -
4032691742672741/21778071482940061661655974875633165533184*x^51 -
4365661706129733/43556142965880123323311949751266331066368*x^50 +
2611430505671355/13611294676837538538534984297270728458 24*x^49 +
7775773251971553/68056473384187692692674921486353642291 2*x^48 +
2960128586757869/85070591730234615865843651857942052864*x^47 +
5634975868887121/17014118346046923173168730371588410572 8*x^46 -
2929929048830885/10633823966279326983230456482242756608*x^45 -
1218799340308081/66461399789245793645190353014017228 8*x^44 -
60776189797711/10384593717069655257060992658440192*x^43 -
8773550689038173/13292279957849158729038070602803445 76*x^42 +
821058129955289/20769187434139310514121985316880384*x^41 +
720567833591597/25961484292674138142652481646100 48*x^40 +
1150125286588865/12980742146337069071326240823050 24*x^39 +
5087073442338325/51922968585348276285304963292200 96*x^38 -
947391015677331/16225927682921336339157801028812 8*x^37 -
802653923553437/20282409603651670423947251286016*x^36 -
4852162005190077/40564819207303340847894502572032*x^35 -
8474540498636661/81129638414606681695789005144064*x^34 +
8947335523097517/10141204801825835211973625643008*x^33 +
3320410373643097/63382530011411470074835160268 8*x^32 +
4428210667099235/31691265005705735037417580134 4*x^31 +
6397245031474829/12676506002282294014967032053 76*x^30 -
2569600322995265/19807040628566084398385987584*x^29 -
6159297294701703/99035203142830421991929937 92*x^28 -
6511077172655227/49517601571415210995964968 96*x^27 +
67329480710387/77371252455336267181195264*x^26 +
2665293617952199/15474250491067253436239052 8*x^25 +
2397857366729415/38685626227668133590597632*x^24 +
6261776889629159/77371252455336267181195264*x^23 -
5288780617255571/19342813113834066795298816*x^22 -
1146080265447361/60446290980731458735308 8*x^21 -
2737320543462567/60446290980731458735308 8*x^20 +
6303983040594249/19342813113834066795298816*x^19 +
5761993159052817/15111572745182864683827 2*x^18 +
5917855484284525/37778931862957161709568*x^17 +
2893393636108155/18889465931478580854784*x^16 -
3686222898590609/47223664828696452136 96*x^15 -
6195017982070121/23611832414348226068 48*x^14 -
639872074928875/737869762948382064 64*x^13 +

```
2679736851870653/295147905179352825856*x^12 +
7070586427964067/737869762948382064 64*x^11 -
3497398567766801/14757395258967641 2928*x^10 +
1436068452607735/461168601842738790 4*x^9 - 41/40320*x^8 - 1/168*x^7 +
11/720*x^6 - 1/120*x^5 + 1/6*x^3 - 1/2*x^2 + 1
```

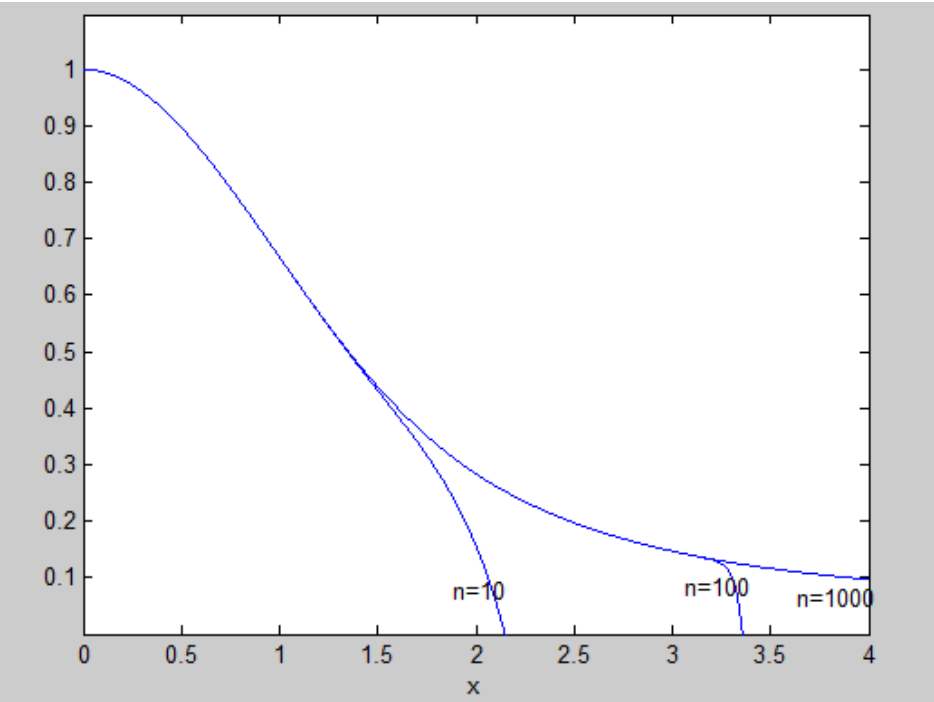
% comparamos con 1000 términos

>> desapotenciasn(1,0,1000)

nterminosdesarrollo =

```
(3*x^494)/10120112665365530917624767335945865352477832488207105917845067901371516978399767344598019185071856224759353893215840595569490436869289673843350669997036925496075871213828318068223345387104660817061988383923637253428100374171234634930905167782457977817040502825617938477616670730761525126609316375432300313165385387054674739 2 -(51*x^493)/20240…

……
……

(631315775791781*x^265)/25326633110845904287795458152411872259597450147964092407200056943912675850908863198240399468671287806934801554024052668349579779513011323900676726282433860394660533468026791526 4 – …….

……
……

 + (1436068452607735*x^9)/461168601842738790 4 - (41*x^8)/40320 - x^7/168 +
(11*x^6)/720 - x^5/120 + x^3/6 - x^2/2 + 1
```

>>gtext('n=1000')



% Observamos que a medida que aumenta el número de términos aumenta el intervalo de aproximación de las solución: definida en (-infinito, infinito).
Por otro lado, dependiendo del orden en que se hagan los dibujos, Matlab puede tomar distinto escalado de ejes; así, dependiendo de la escala en el eje de ordenadas, puede cambiar la idea que nos da la aproximación como se ve en la gráfica de abajo con el conjunto de comandos

>> desapotenciasn(1,0,1000);

>> hold on

>> gtext('n=1000')

>> desapotenciasn(1,0,100);

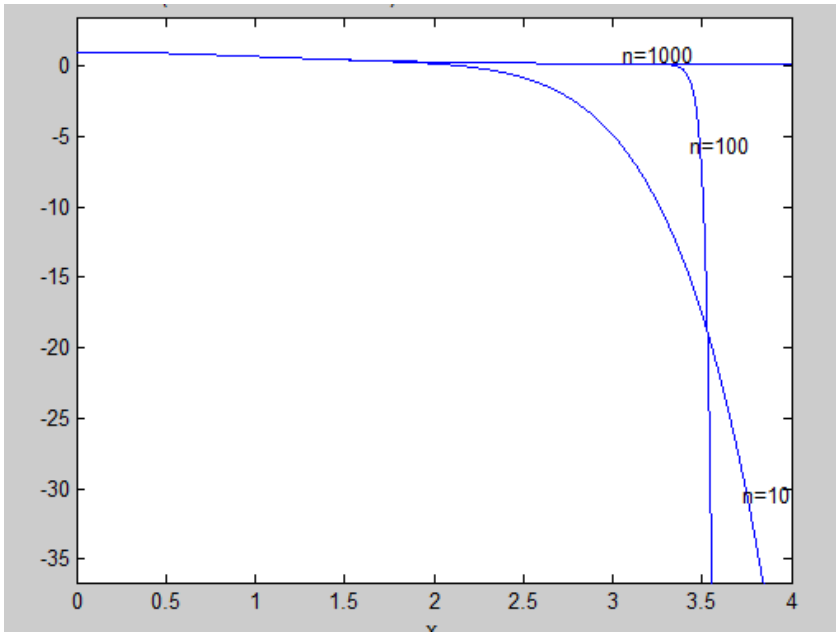
>> hold on

>> gtext('n=100')

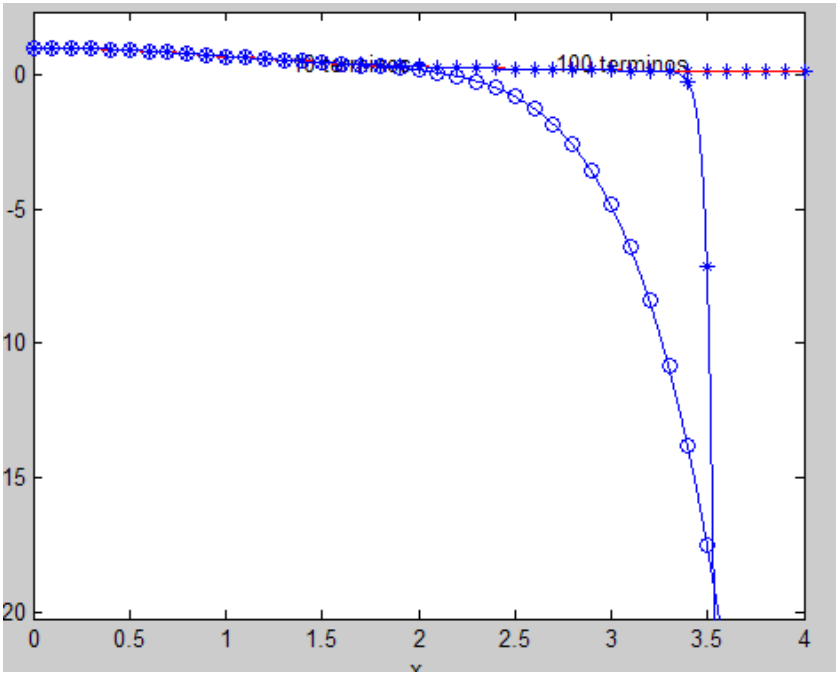
>> desapotenciasn(1,0,10);

>> hold on

>> gtext('n=10')



Abajo, con "plot" la gráfica de las mismas funciones: 'o' corresponde a la aproximación de la solución con 10 términos del desarrollo, '' con n=100, y en rojo con 1000 términos.*



% Ecuaciones de tipo Schrödinger *(unidimensional, estacionaria): y''+h(x)y=0, con h(x)=a^2-V(x). Se trata de ecuaciones importantes en múltiples campos de la ciencia y la técnica. No siempre las resuelve Matlab, de aquí el interés de construir funciones Matlab como soluschroedinger.m que nos aproxime tantos términos del desarrollo en serie de la solución como se quiera.*

% Resolvemos y''+(3-x^2)y=0 con distintas condiciones iniciales

>> dsolve('D2y+(3-t^2)*y=0') *% la solución general*

ans =

C1*exp(-1/2*t^2)*t-i*C2*pi^(1/2)*exp(1/2*t^2)+C2*exp(-1/2*t^2)*pi*erf(i*t)*t

>> dsolve('D2y+(3-t^2)*y=0', 'y(0)=1','Dy(0)=0')

ans =

i/pi^(1/2)*(-i*pi^(1/2)*exp(1/2*t^2)+exp(-1/2*t^2)*pi*erf(i*t)*t)

>> dsolve('D2y+(3-t^2)*y=0', 'y(0)=0','Dy(0)=1')

ans =

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

*% De este último resultado (la ED con condiciones iniciales y(0)=0,y'(0)=1) se deduce que una solución de la ecuación homogénea es y1= exp(- t^2/2)t. Esto permite reducir el orden y encontrar la solución general de la no homogénea: vemos que Matlab la encuentra en términos integrales (no encuentra las primitivas de manera explícita). Como consecuencia, para cualesquiera condiciones iniciales, no se va a conocer la solución explícita. Por ejemplo, para condiciones iniciales y(0)=1 e y'(0)=0, la solución se expresa también en términos de la función erf (en términos integrales, ver primer cuaderno, con help): "erf(x) = 2/sqrt(pi) * integral from 0 to x of exp(-t^2) dt".*
Pero como se ve abajo esto también depende de la versión de Matlab

% La función soluschroedinger.m nos proporciona los N primeros términos del desarrollo en serie de la solución. Lee N y las condiciones iniciales a0 y a1. Las fórmulas recursivas para obtener los coeficientes son

```
a(1)=a1;
a(2)=-a0*3/2;
a(3)=-a(1)/2;
a(4)=(a0-3*a(2))/12;
for i=3:N-2
   a(i+2)=(a(i-2)-3*a(i))/((i+2)*(i+1));
end
```

>> type soluschroedinger

% para a0=1, a1=0, N=100

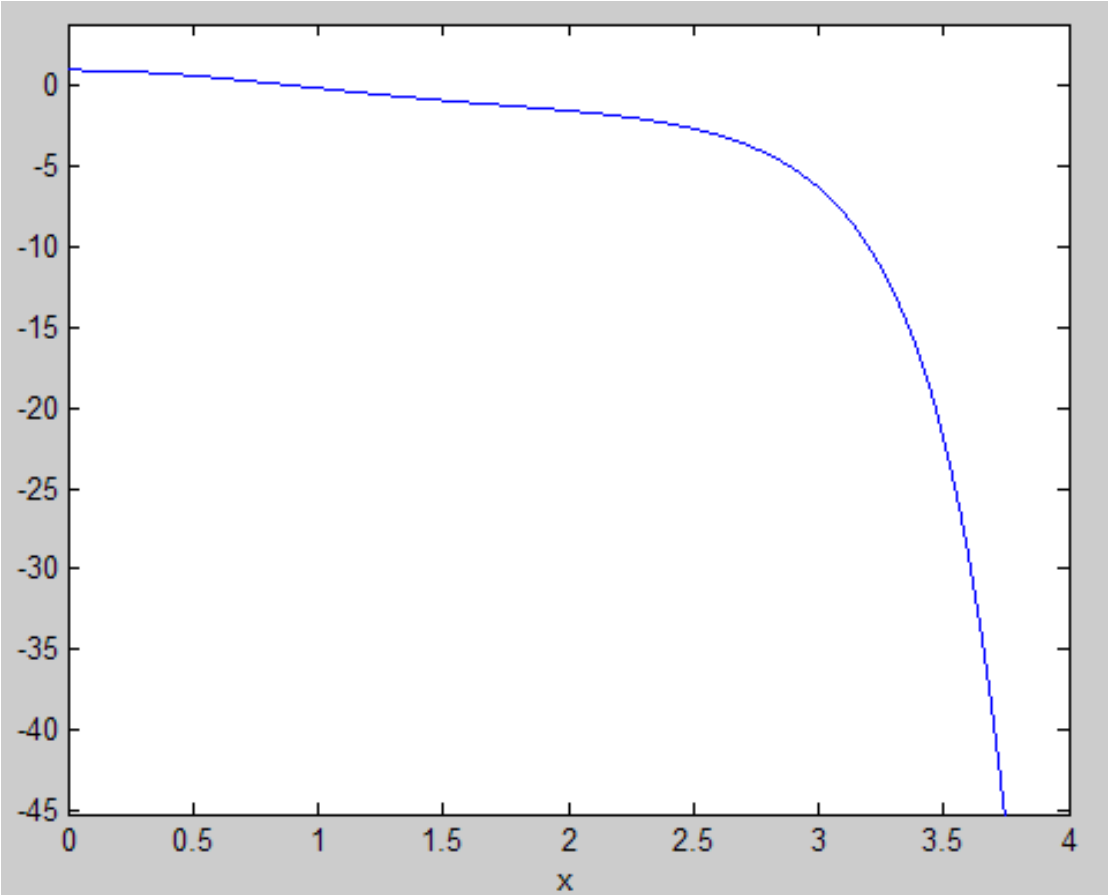
>> soluschroedinger(1,0,100)

nterminosdesarrollo =

(1082089187094889*x^100)/208592483976651375233888838493120323691670363511391872065140782013888645095765678713179891302 4 - (6699234030472215*x^98)/1303703024854071095211805240582002023072939771946199200407129887586804031848535491957374320 64 + (317111807823997*x^96)/6365737426045269019588892776279306753285838730206050783237938904232441561760427206823116 8 - (3767979964990127*x^94)/7957171782556586274486115970349133441607298412757563479047423630290551952200534008528896 + (5471930016006035*x^92)/1243308091024466605388455620367052100251140376993369293601159942232898742531333438832 64 - (7785417551482063*x^90)/1942668892225729070919461906823518906642406839052139521251812409738904285205208498176 + (1352423032352233*x^88)/3794275180128377091639574036764685364535950857523710002444946112771297432041422848 - (3679435596293841*x^86)/1185710993790117841137366886488964176417484642976159375764045660241030447512944 64 + (2441632060167111*x^84)/9263367138985295633885678800695032628261598773251245123156606720633050371194 88 - (6337667846143507*x^82)/2894802230932904885589274625217197696331749616641014100986439600197828240998 4 + (8014962212038731*x^80)/4523128485832663883733241601901871400518358776001584532791311875309106626 56 - (2475336736411971*x^78)/1766847064778384329583297500742918515827483896

75618958121606201292619776 +
(743919411720587*x^76)/690174634679056378743475586227702545245110897217038655516252422379929 6 -
(6980245278656683*x^74)/862718293348820473429344482784628181556388621521298319395315527974912 +
(3977358164179283*x^72)/673998666678765994866675377175490766840928610563514312027590256230 4 -
(4416501398957343*x^70)/105312291668557186697918027683670432318895095400549111254310977536 +
(2378452962241783*x^68)/822752278660603021077484591278675252491367932816789931674304512 -
(2492477517024307*x^66)/128555043540719222043356967387293008201776239502623426824110 08 +
(5057458990578563*x^64)/401734511064747568885490523085290650630550748445698208825344 -
(1245997617790373*x^62)/156927543384667019095894735580191660402558886111600862822 4 +
(2372405820257539*x^60)/490398573077084434674671048688098938757996519098752696 32 -
(8759007874489193*x^58)/306499108173177771671669405430061836723747824436720435 2 +
(3895451282097757*x^56)/239452428260295134118491722992235809940427987841187 84 -
(6704449887874935*x^54)/748288838313422294120286634350736906063837462003712 +
(693015305152899*x^52)/14615016373309029182036848327162830196559325429 76 -
(4424722673433401*x^50)/182687704666362864775460604089535377456991567872 +
(1691096919045983*x^48)/14272476927059598810582859694494951363827466 24 -
(4976166666503015*x^46)/89202980794122492566142873090593446023921664 +
(6984880174561705*x^44)/27875931498163278926919647840810451882475 52 -
(1175551330920815*x^42)/10889035741470030830827987437816582766592 +
(6011997005919707*x^40)/13611294676837538538534984297270728458 24 -
(7343794780092071*x^38)/42535295865117307932921825928971026432 +
(2117605232018043*x^36)/33230699894622896822595176507008614 4 -
(2322318127859991*x^34)/10384593717069655257060992658440192 +
(4775859476695655*x^32)/64903710731685345356631204115251 2 -
(4641405633940465*x^30)/20282409603651670423947251286016 +
(4191482089866997*x^28)/63382530011411470074835160268 8 -
(1775215146924303*x^26)/99035203142830421991929937 92 +
(172602487288207*x^24)/38685626227668133590597632 -
(7979149494245733*x^22)/77371252455336267181195264 +

```
(5206740546357605*x^20)/241785163922925834941235 2 -
(6223671824006247*x^18)/151115727451828646838272 +
(6561817031498399*x^16)/944473296573929042739 2 -
(6208892033726517*x^14)/590295810358705651712 +
(4987536210706027*x^12)/36893488147419103232 - (73*x^10)/48384 +
(179*x^8)/13440 - (23*x^6)/240 + (11*x^4)/24 - (3*x^2)/2 + 1
```



% Introduciendo en la función soluschrodinger.m vpa(nterminosdesarrollo,2), y ejecutando de nuevo, visualizamos mejor la solución, y obtenemos esta misma gráfica de arriba

>> soluschroedinger_vpa(1,0,100)

nterminosdesarrollo =

5.2e-79*x^100 - 5.1e-77*x^98 + 5.0e-75*x^96 - 4.7e-73*x^94 + 4.4e-71*x^92 - 4.0e-69*x^90 + 3.6e-67*x^88 - 3.1e-65*x^86 + 2.6e-63*x^84 - 2.2e-61*x^82 + 1.8e-59*x^80 - 1.4e-57*x^78 + 1.1e-55*x^76 - 8.1e-54*x^74 + 5.9e-52*x^72 - 4.2e-50*x^70 + 2.9e-48*x^68 - 1.9e-46*x^66 + 1.3e-44*x^64 - 7.9e-43*x^62 + 4.8e-41*x^60 - 2.9e-39*x^58 + 1.6e-37*x^56 - 9.0e-36*x^54 + 4.7e-34*x^52 - 2.4e-32*x^50 + 1.2e-30*x^48 - 5.6e-29*x^46 + 2.5e-27*x^44 - 1.1e-25*x^42 + 4.4e-24*x^40 - 1.7e-22*x^38 + 6.4e-21*x^36 - 2.2e-19*x^34 + 7.4e-18*x^32 - 2.3e-16*x^30 + 6.6e-15*x^28 - 1.8e-13*x^26 + 4.5e-12*x^24 - 1.0e-10*x^22 + 2.2e-9*x^20 - 4.1e-8*x^18 + 6.9e-7*x^16 - 1.1e-5*x^14 + 1.4e-4*x^12 - 1.5e-3*x^10 + 0.013*x^8 - 0.096*x^6 + 0.46*x^4 - 1.5*x^2 + 1.0

% A continuación, con dsolve, vemos que la solución nos da de forma distinta en las versiones de Matlab 2008 y 2011, que deben ser la misma por el teorema de existencia y unicidad de solución del problema de Cauchy. También podemos obtener tantos términos como queramos del desarrollo en serie de la solución usando "taylor" y comprobar que, en ambas versiones, coinciden

>> dsolve('D2y+(3-t^2)*y=0', 'y(0)=1','Dy(0)=0') *% con Matlab 2008*

ans =

i/pi^(1/2)*(-i*pi^(1/2)*exp(1/2*t^2)+exp(-1/2*t^2)*pi*erf(i*t)*t)

>> taylor(ans,10) *% nos da 10 términos del desarrollo en serie de Taylor de la solución*

ans =

1-3/2*t^2+11/24*t^4-23/240*t^6+179/13440*t^8

>> dsolve('D2y+(3-t^2)*y=0', 'y(0)=1','Dy(0)=0') *% con Matlab 2011*

ans =

exp(t^2/2)*hypergeom([1], [1/2], -t^2)

>> taylor(ans,10)

ans =

(179*t^8)/13440 - (23*t^6)/240 + (11*t^4)/24 - (3*t^2)/2 + 1

% vemos que dichos desarrollos coinciden, y coinciden con los obtenidos con

>> soluschroedinger(1,0,9)

nterminosdesarrollo =

(179*x^8)/13440 - (23*x^6)/240 + (11*x^4)/24 - (3*x^2)/2 + 1

% Para las condiciones iniciales y(0)=1, y'(0)=0, en que se encuentra la solución explícita con dsolve, vemos que la solución coincide (después de simplificación) en las versiones de Matlab 2008 y 2011 respectivamente

>> dsolve('D2y+(3-t^2)*y=0', 'y(0)=0','Dy(0)=1') *% versión 2008*

ans =

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

>> dsolve('D2y+(3-t^2)*y=0', 'y(0)=0','Dy(0)=1') *% versión 2011*

ans =

(t*exp(t^2/2))/exp(t^2)

% soluschroedinger(1,0,100) nos da los 100 primeros términos del desarrollo en serie de la solución; dichos términos deben coincidir con los 100 primeros términos del desarrrollo en serie de Taylor de la solución explícita encontrada exp(t^2/2)t. Lo vemos a continuación.

>> ut100=vpa(taylor(exp(-1/2*x^2)*x,100),2)

ut100 =

- 2.9e-78*x^99 + 2.9e-76*x^97 - 2.7e-74*x^95 + 2.6e-72*x^93 - 2.4e-70*x^91 + 2.1e-68*x^89 - 1.9e-66*x^87 + 1.6e-64*x^85 - 1.4e-62*x^83 + 1.1e-60*x^81 - 8.9e-59*x^79 + 7.0e-57*x^77 - 5.3e-55*x^75 + 3.9e-53*x^73 - 2.8e-51*x^71 + 2.0e-49*x^69 - 1.3e-47*x^67 + 8.8e-46*x^65 - 5.7e-44*x^63 + 3.5e-42*x^61 - 2.1e-40*x^59 + 1.2e-38*x^57 - 6.8e-37*x^55 + 3.7e-35*x^53 - 1.9e-33*x^51 + 9.6e-32*x^49 - 4.6e-30*x^47 + 2.1e-28*x^45 - 9.3e-27*x^43 + 3.9e-25*x^41 - 1.6e-23*x^39 + 6.0e-22*x^37 - 2.1e-20*x^35 + 7.3e-19*x^33 - 2.3e-17*x^31 + 7.0e-16*x^29 - 2.0e-14*x^27 + 5.1e-13*x^25 - 1.2e-11*x^23 + 2.7e-10*x^21 - 5.4e-9*x^19 + 9.7e-8*x^17 - 1.6e-6*x^15 + 2.2e-5*x^13 - 2.6e-4*x^11 + 2.6e-3*x^9 - 0.021*x^7 + 0.12*x^5 - 0.5*x^3 + x

>> soluschroedinger(0,1,100) *% 100 términos del desarrollo en serie de la solución*

nterminosdesarrollo

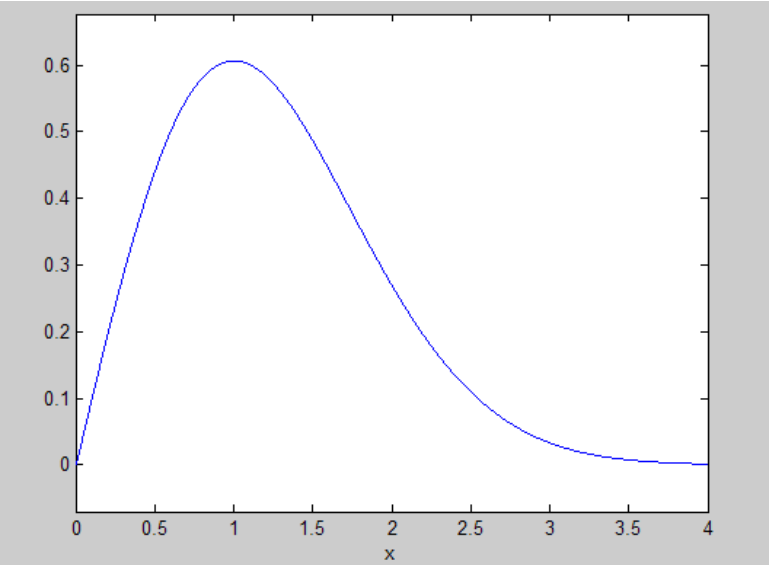
(4663802020167463*x^97)/16296287810675888690147565507275025288411747149327490005089123594835050398106693649467179008 - (6091496516137095*x^99)/20859248397665137523388883849312032369167036351139187206514078201388864509576567871317989130 24 +
……
……

(6212541674450187*x^27)/31691265005705735037417580134 4 + (630961263811347*x^25)/12379400392853802748991242 24 - (7571535165736165*x^23)/61897001964269013744956211 2 + (5205430426443613*x^21)/19342813113834066795298816 - (1626697008263629*x^19)/30223145490365729367654 4 + (3660068268593165*x^17)/37778931862957161709568 - (3660068268593165*x^15)/23611832414348226068 48 + x^13/46080 - x^11/3840 + x^9/384 - x^7/48 + x^5/8 - x^3/2 + x

>> soluschroedinger_vpa(0,1,100) *% mismo resultado utilizando vpa(.,2)*

nterminosdesarrollo =

- 2.9e-78*x^99 + 2.9e-76*x^97 - 2.7e-74*x^95 + 2.6e-72*x^93 - 2.4e-70*x^91 + 2.1e-68*x^89 - 1.9e-66*x^87 + 1.6e-64*x^85 - 1.4e-62*x^83 + 1.1e-60*x^81 - 8.9e-59*x^79 + 7.0e-57*x^77 - 5.3e-55*x^75 + 3.9e-53*x^73 - 2.8e-51*x^71 + 2.0e-49*x^69 - 1.3e-47*x^67 + 8.8e-46*x^65 - 5.7e-44*x^63 + 3.5e-42*x^61 - 2.1e-40*x^59 + 1.2e-38*x^57 - 6.8e-37*x^55 + 3.7e-35*x^53 - 1.9e-33*x^51 + 9.6e-32*x^49 - 4.6e-30*x^47 + 2.1e-28*x^45 - 9.3e-27*x^43 + 3.9e-25*x^41 - 1.6e-23*x^39 + 6.0e-22*x^37 - 2.1e-20*x^35 + 7.3e-19*x^33 - 2.3e-17*x^31 + 7.0e-16*x^29 - 2.0e-14*x^27 + 5.1e-13*x^25 - 1.2e-11*x^23 + 2.7e-10*x^21 - 5.4e-9*x^19 + 9.7e-8*x^17 - 1.6e-6*x^15 + 2.2e-5*x^13 - 2.6e-4*x^11 + 2.6e-3*x^9 - 0.021*x^7 + 0.12*x^5 - 0.5*x^3 + x



% Comparando gráficas, resulta que la gráfica con soluschroedinger(0,1,100), y la del desarrollo en serie de Taylor coinciden: esto es, la gráfica es la misma con el conjunto de comandos

>> dsolve('D2y+(3-t^2)*y=0', 'y(0)=0','Dy(0)=1');

>> taylor(ans,100);

>> ezplot(ans,0,4)

% o el conjunto de comandos

>> syms x

>> ezplot(taylor(exp(-1/2*x^2)*x,100),0,4)

% También se puede comprobar que los desarrollos que nos da soluschrodinger y taylor son los mismos. Para esta comprobación pedimos menos términos

>> y1=dsolve('D2y+(3-t^2)*y=0','y(0)=0','Dy(0)=1')

y1 =

(t*exp(t^2/2))/exp(t^2)

>> simplify(y1)

ans =

t/exp(t^2/2)

>> taylor(y1,10)

ans =

t^9/384 - t^7/48 + t^5/8 - t^3/2 + t

>> soluschroedinger(0,1,10)

nterminosdesarrollo =

x^9/384 - x^7/48 + x^5/8 - x^3/2 + x

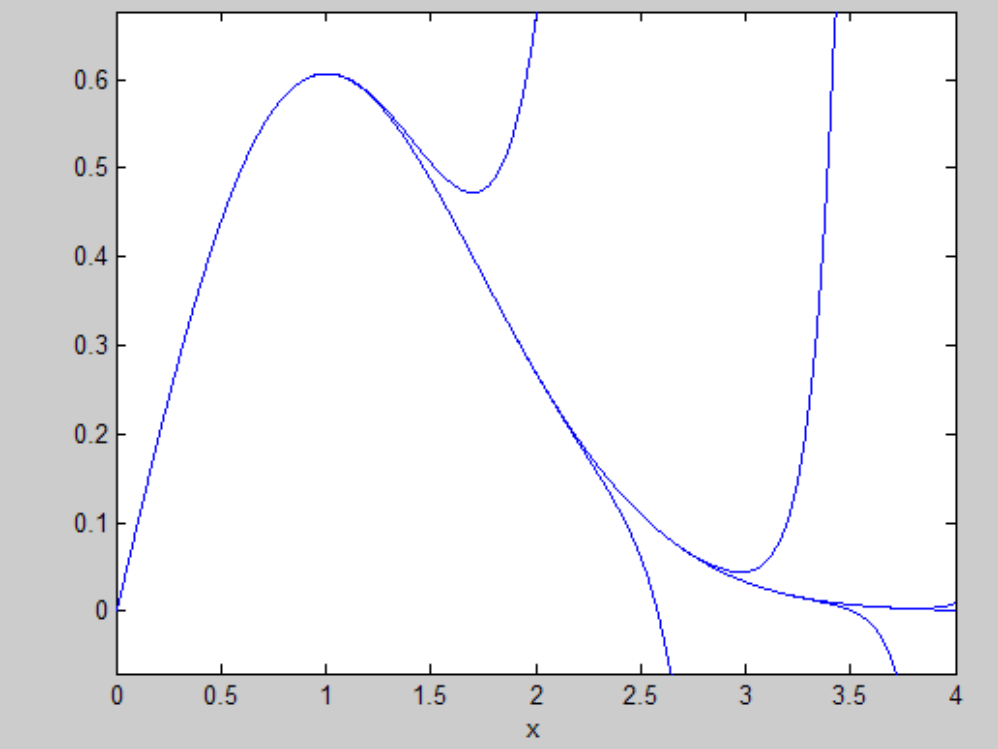
% lo ponemos en términos de variable independiente t

>> y2=t^9/384 - t^7/48 + t^5/8 - t^3/2 + t

y2 =

t^9/384 - t^7/48 + t^5/8 - t^3/2 + t

```
>> taylor(y1,10)-y2

ans =

0

% Abajo la gráfica obtenida con la serie de comandos

>> y1=dsolve('D2y+(3-t^2)*y=0','y(0)=0','Dy(0)=1');

>> ezplot(y1,0,4)

>> hold on

>>soluschroedinger(0,1,10);

>> soluschroedinger(0,1,20);

>> soluschroedinger(0,1,30);

>> soluschroedinger(0,1,40);

>> soluschroedinger(0,1,50);

>> soluschroedinger(0,1,60);

>> soluschroedinger(0,1,70);

>> soluschroedinger(0,1,80);

>> soluschroedinger(0,1,90);

>> soluschroedinger(0,1,100);
```



% gráfica de la solución y de su aproximación por 10, 20, 30, 40 ,…,100 términos. Vemos que en este intervalo (ventana de observación de la solución), a partir de la aproximación con 60 términos, ya no se aprecia diferencia. Abajo, la gráfica de las aproximaciones con el n correspondiente, y ajustando los ejes

>> hold off

>> soluschroedinger(0,1,10);

>>gtext('n=10')

>> soluschroedinger(0,1,20);

>>gtext('n=20')

>> soluschroedinger(0,1,30);

>>gtext('n=30')

>> soluschroedinger(0,1,40);

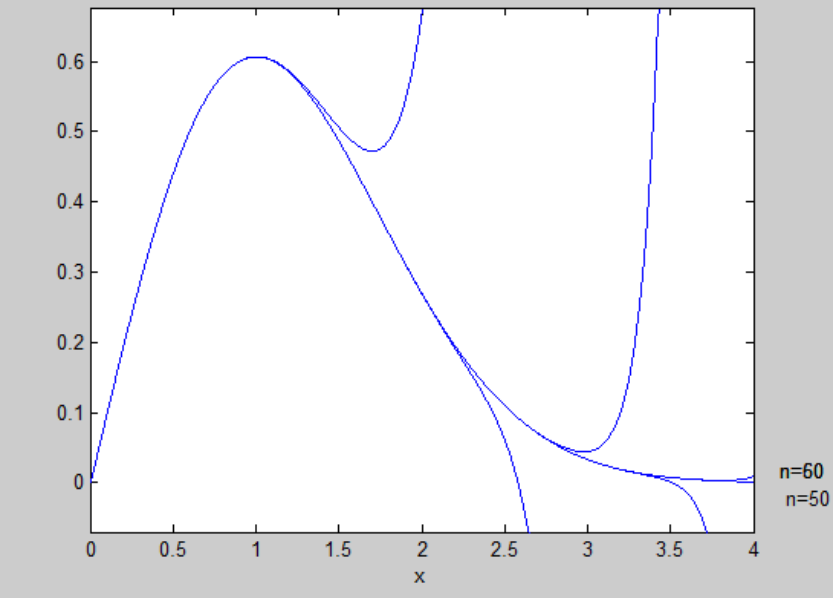
>>gtext('n=40')

>> soluschroedinger(0,1,50);

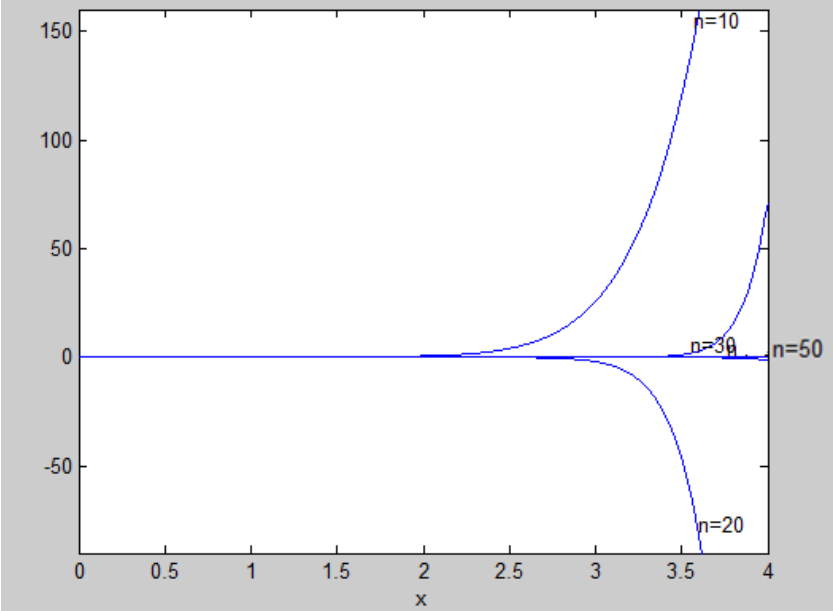
>>gtext('n=50')

>> soluschroedinger(0,1,60);

>>gtext('n=60')



>>axis([0,4,-90,160])



*% Una ecuación de tipo Schrodinger, con potencial bien distinto, es y''+(a^2-1/x^2)y=0 (**ejercicio 5**). Esta ED se reduce a una ecuación de Bessel con el cambio de variable y=u sqrt(x). La nueva ED es: u'' + u'/x+ (a^2u-(5/4)u/x^2)=0 con variable independiente x e incógnita u. Multiplicando por x^2 y haciendo t=ax se llega a la ecuación de Bessel: t^2u''+tu'+(x^2-5/4)u=0 para mu^2=5/4 . La solución con dsolve se tiene en términos de las funciones de Bessel de primera y segunda especie:*

>> syms a

>> dsolve('D2y+(a^2-1/t^2)*y=0')

ans =

t^(1/2)*(C1*besselj(1/2*5^(1/2),t*a)+bessely(1/2*5^(1/2),t*a)*C2)

*% Otras ecuaciones lineales de segundo orden, que se obtienen mediante un cambio de variable de **ecuaciones de Riccati**, son por ejemplo: v''+x^2v=0 y v''+v'+exp(x)x^2v=0. La primera se obtiene de y'=x^2+y^2 y la segunda de y'= x^2+exp(x)y^2 (ver hoja 2, ejercicio 4).*

*% Dada la ecuación de Riccati (de primer orden, no lineal, evidentemente) y'+p(x)y^2+q(x)y=h(x), para reducirla a ED lineal de segundo orden se hace el cambio de v=exp(int (p(x)*y(x))). Resuelta esta ED se obtiene la solución con y=(v'/(vp)). Al igual que se ha observado antes, dependiendo de la versión de Matlab, la solución puede ser distinta.*

% Los resultados con la versión 2008 de Matlab son

>> dsolve('Dy=t^2+y^2') *% ED de Riccati,*

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 )
```

```
% se tiene la solución en términos de una constante C1, de acuerdo con el cambio
  comentado.

>> dsolve('D2y+t^2*y=0') % ED de segundo orden a la que se reduce

ans =

t^(1/2)*(C1*besselj(1/4,1/2*t^2)+bessely(1/4,1/2*t^2)*C2)

>> dsolve('Dy=t^2+exp(t)*y^2') % ED de Riccati

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

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

>> dsolve('D2y+Dy+exp(t)*t^2*y=0') % ED de segundo orden a la que se reduce

ans =

DESol({diff(_Y(t),$(t,2))+diff(_Y(t),t)+exp(t)*t^2*_Y(t)},{_Y(t)})

% los resultados con la versión 2011 de Matlab son

>> dsolve('Dy=t^2+y^2') % ED de Riccati

ans =

-((C15*besselj(1/4, t^2/2) + C16*bessely(1/4, t^2/2))/(2*t^(1/2)) +
t^(1/2)*(C15*(t*besselj(-3/4, t^2/2) - besselj(1/4, t^2/2)/(2*t)) + C16*(t*bessely(-3/4,
t^2/2) - bessely(1/4, t^2/2)/(2*t))))/(t^(1/2)*(C15*besselj(1/4, t^2/2) +
C16*bessely(1/4, t^2/2)))

% se observa que este resultado no puede ser correcto pues nos da la solución en
  términos de dos constantes de integración en vez de una sola constante.
```

>> dsolve('D2y+t^2*y=0') *% ED de segundo orden*

ans =

(C3*(t^2*i)^(1/2)*besselk(1/4, (t^2*i)/2))/(pi^(1/2)*t^(1/2)) +
(2^(1/2)*4^(1/4)*C2*pi*(t^2*i)^(1/2)*besseli(1/4, (t^2*i)/2))/(4*t^(1/2)*gamma(3/4))

% podría ser una solución en el campo complejo

>> dsolve('Dy=t^2+exp(t)*y^2') *% ED de Riccati*

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

ans =

[empty sym]

>> dsolve('D2y+Dy+exp(t)*t^2*y=0') *% ED de segundo orden a la que se reduce*

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

ans =

[empty sym]

*% En relación a sistemas y desarrollos en serie de potencias, se puede seguir la misma idea, si no se conoce una solución del sistema homogéneo, para reducir de orden, por ejemplo: adaptar las funciones desapotenciasn.m o soluschroedinger.m para encontrar la aproximación de la solución vectorial en forma de serie de potencias. Esto suele pasar si el sistema no es de coeficientes constantes o de tipo Euler. En cambio, si conocemos una matriz fundamental de soluciones, se puede aplicar el método de variación de parámetros. Tomamos como ejemplo: resolver en (-1,1) el problema de Cauchy (**ejercicio 7**):*

$$\begin{cases} y_1' &= -\frac{x}{1-x^2}y_1 + \frac{1}{1-x^2}y_2 + 1 \\ y_2' &= \frac{1}{1-x^2}y_1 - \frac{x}{1-x^2}y_2 + 1 \end{cases}$$

$$y_1(0) = 0\,,\, y_2(0) = 1\,.$$

% Primero, intentamos resolver con dsolve. La versión 2008 de Matlab lo hace, y no lo hace la versión 2011, tal y como se ve abajo. Luego, aplicamos fórmulas integrales para resolver y comparamos las soluciones obtenidas de distintas formas.

% con Matlab 2008

>> [Y1,Y2]=dsolve('Dy1=(-t/(1-t^2))*y1+(1/(1-t^2))*y2+1','Dy2=(1/(1-t^2))*y1+(-t/(1-t^2))*y2+1')

Y1 =

log(t+1)+C1+log(t+1)*t-t+C2*t-1

Y2 =

log(t+1)*t+log(t+1)-t-1+C1*t+C2

% C1 y C2 son las constantes de integración, pero Matlab 2008 también resuelve el problema de Cauchy asociado

>> [Y1,Y2]=dsolve('Dy1=(-t/(1-t^2))*y1+(1/(1-t^2))*y2+1','Dy2=(1/(1-t^2))*y1+(-t/(1-t^2))*y2+1', 'y1(0)=0', 'y2(0)=1')

Y1 =

log(t+1)+log(t+1)*t+t

Y2 =

log(t+1)*t+log(t+1)+1

% verificamos que, en efecto, se trata de la solución del sistema

>> diff(Y1)-((-t/(1-t^2))*Y1+(1/(1-t^2))*Y2+1) *% verificación de la primera ecuación*

ans =

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

>> simplify(ans)

ans =

0

>> diff(Y2)-((-t/(1-t^2))*Y2+(1/(1-t^2))*Y1+1) *% verificación de la segunda ecuación*

ans =

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

>> simplify(ans)

ans =

0

>> subs(Y1,t,0)

ans =

0

>> subs(Y2,t,0)

ans =

1

% mientras que, como se ve abajo, la versión 2011 de Matlab no resuelve ni el sistema ni el problema de Cauchy asociado

>> [Y1,Y2]=dsolve('Dy1=(-t/(1-t^2))*y1+(1/(1-t^2))*y2+1','Dy2=(1/(1-t^2))*y1+(-t/(1-t^2))*y2+1')

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

Y1 =

[empty sym]

Y2 =

[]

>> [Y1,Y2]=dsolve('Dy1=(-t/(1-t^2))*y1+(1/(1-t^2))*y2+1','Dy2=(1/(1-t^2))*y1+(-t/(1-t^2))*y2+1', 'y1(0)=0', 'y2(0)=1')

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

Y1 =

[empty sym]

Y2 =

[]

% En este ejemplo, como se trata de un sistema lineal (con coeficientes no constantes) y se conoce la matriz fundamental, aplicamos el método de variación de parámetros para encontrar la solución general del sistema y la del problema de Cauchy asociado

>> syms t c1 c2

>> phi=[1,t; t,1] *% definimos la matriz fundamental*

phi =

[1, t]
[t, 1]

>> y=phi*[c1;c2]+phi*int(inv(phi)*[1;1]) *% es la solución general; c1 y c2 constantes*

y =

c1+t*c2+atanh(t)+1/2*log(t-1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(t-1)+1/2*log(t+1))
t*c1+c2+atanh(t)+1/2*log(t-1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(t-1)+1/2*log(t+1))

% Si se quieren calcular c1 y c2, para que se verifiquen las condiciones iniciales, imponemos y(0)=[0;1]. Para ello, nos damos cuenta que los logaritmos tal y como aparecen no son funciones reales para t<1, pues t-1 es negativo; esto es debido a que Matlab al integrar 1/(t-1) da

>> int(1/(t-1))

ans =

log(t - 1)

% Por tanto, la solución que obtenida puede ser correcta fuera del intervalo (-1,1), pero en (-1,1) hay que cambiar log(t-1) por log(-t+1). Así

Y =
*c1+t*c2+atanh(t)+1/2*log(-t+1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(-t+1)+1/2*log(t+1))*
*t*c1+c2+atanh(t)+1/2*log(-t+1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(-t+1)+1/2*log(t+1))*

y, esta función vectorial se define, por ejemplo :

>>Y=[c1+t*c2+atanh(t)+1/2*log(-t+1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(-t+1)+1/2*log(t+1));t*c1+c2+atanh(t)+1/2*log(t+1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(-t+1)+1/2*log(t+1))]

Y =

c1+t*c2+atanh(t)+1/2*log(-t+1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(-t+1)+1/2*log(t+1))
t*c1+c2+atanh(t)+1/2*log(-t+1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(-t+1)+1/2*log(t+1))

>> C1=subs(c1+t*c2+atanh(t)+1/2*log(-t+1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(-t+1)+1/2*log(t+1)),t,0)

C1 =

c1+atanh(0)+log(1)

>> C2=subs(t*c1+c2+atanh(t)+1/2*log(-t+1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(-t+1)+1/2*log(t+1)),t,0)

C2 =

c2 +atanh(0)+log(1)

% Como Y(0)=[0;1]=[C1;C2], despejamos c1=0 y c2=1. Así la solución del problema de Cauchy se tiene al substituir obtiene al substituir, en Y, c1 por -0 y c2 por 1:

>> subs(subs(Y,c1,0),c2,1)

ans =

t+atanh(t)+1/2*log(-t+1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(-t+1)+1/2*log(t+1))
1+atanh(t)+1/2*log(-t+1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(-t+1)+1/2*log(t+1))

*% "ans" es ahora la solución vectorial, solución del problema de Cauchy dado, con componentes funciones reales que deberían coincidir en el intervalo (-1,1) con las obtenidas con el comando dsolve (esto es debido al teorema de existencia y unicidad de solución). Comparando, vemos la dificultad de demostrarlo con Matlab (al menos en este intervalo): consideramos la primera componente de esta solución y la obtenida dsolve, log(t+1)+log(t+1)*t+t, las restamos y simplificamos*

>> log(t+1)+log(t+1)*t+t- (t+atanh(t)+1/2*log(-t+1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(-t+1)+1/2*log(t+1)))

ans =

1/2*log(t+1)+log(t+1)*t-atanh(t)-1/2*log(-t+1)-t*(atanh(t)+1/2*log(-t+1)+1/2*log(t+1))

>> simplify(ans)

ans =

1/2*log(t+1)+1/2*log(t+1)*t-atanh(t)-1/2*log(-t+1)-t*atanh(t)-1/2*t*log(-t+1)

>> log(t+1)+log(t+1)*t+t- (t+atanh(t)+1/2*log(-t+1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(-t+1)+1/2*log(t+1)))

ans =

1/2*log(t+1)+log(t+1)*t-atanh(t)-1/2*log(-t+1)-t*(atanh(t)+1/2*log(-t+1)+1/2*log(t+1))

>> simple(ans)

simplify:
……
…….

ans =

-1/2*(-log(t+1)+2*atanh(t)+log(-t+1))*(t+1)

% como Matlab no lo simplifica (no nos devuelve "0"), comprobamos al igual que antes, que tenemos la solución exacta, y el que coincidan en (-1,1) es consecuencia, por ejemplo, del teorema de existencia y unicidad de solución para problemas de valores iniciales.

% Definimos

>>YY1= t+atanh(t)+1/2*log(-t+1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(-t+1)+1/2*log(t+1))

```
>>YY2= 1+atanh(t)+1/2*log(-t+1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(-
t+1)+1/2*log(t+1))

% y repetimos la comprobación de solución

>> diff(YY1)-((-t/(1-t^2))*YY1+(1/(1-t^2))*YY2+1) % verificación de la primera ecuación

ans =

1/(1-t^2)-1/2/(-t+1)+1/2/(t+1)+atanh(t)+1/2*log(-t+1)+1/2*log(t+1)+t*(1/(1-t^2)-
1/2/(-t+1)+1/2/(t+1))+t/(1-t^2)*(t+atanh(t)+1/2*log(-
t+1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(-t+1)+1/2*log(t+1)))-1/(1-
t^2)*(1+atanh(t)+1/2*log(-t+1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(-t+1)+1/2*log(t+1)))

>> simplify(ans)

ans =

0

>> diff(YY2)-((-t/(1-t^2))*YY2+(1/(1-t^2))*YY1+1) % verificación de la segunda

ans =

1/(1-t^2)-1/2/(-t+1)+1/2/(t+1)+atanh(t)+1/2*log(-t+1)+1/2*log(t+1)+t*(1/(1-t^2)-
1/2/(-t+1)+1/2/(t+1))+t/(1-t^2)*(1+atanh(t)+1/2*log(-
t+1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(-t+1)+1/2*log(t+1)))-1/(1-
t^2)*(t+atanh(t)+1/2*log(-t+1)+1/2*log(t+1)+t*(atanh(t)+1/2*log(-t+1)+1/2*log(t+1)))-
1

 >> simplify(ans)

ans =

0

% verificación de condiciones iniciales

>> subs(YY1,t,0)

ans =

    0
```

>> subs(YY2,t,0)

ans =

1

% Así, tenemos la solución del problema de Cauchy asociado al sistema en (-1,1). Si repetimos los cálculos con la versión 2011 de Matlab

>> syms t c1 c2

>> phi=[1,t; t,1]

phi =

[1, t]
[t, 1]

>> y=phi*[c1;c2]+phi*int(inv(phi)*[1;1]) *% es la solución general; c1 y c2 constantes*

y =

c1 - log(t - 1)/2 + log(t + 1)/2 + log(t^2 - 1)/2 + c2*t + t*(log(t + 1)/2 - log(t - 1)/2 + log(t^2 - 1)/2)

c2 - log(t - 1)/2 + log(t + 1)/2 + log(t^2 - 1)/2 + c1*t + t*(log(t + 1)/2 - log(t - 1)/2 + log(t^2 - 1)/2)

% de nuevo vemos que la versión 2011 de Matlab nos una solución general aparentemente distinta (en este caso al resolver las integrales); no obstante, si repetimos el proceso anterior, tomando las dos componentes del vector reales e imponiendo las condiciones iniciales y(0)=[0;1], se debe tener la misma solución que utilizando Matlab 2008 con esta fórmula integral y con dsolve. Esto es consecuencia del teorema de existencia y unicidad de solución para el problema de Cauchy.

% Nota: como se sabe, las ecuaciones lineales de orden n (respectivamente los sistemas lineales con n ecuaciones), con coeficientes constantes, se saben resolver siempre que se puedan encontrar explícitamente las raíces del polinomio característico (respectivamente los valores propios del determinante) asociado. Para las ED no homogéneas asociadas, también hay que poder encontrar una solución particular. Las ecuaciones y sistemas de tipo Euler se resuelven reduciendo a coeficientes constantes. Las funciones soluschroedinger.m y desapotenciasn.m se pueden adaptar para aproximar ecuaciones (sistemas, respectivamente) lineales de orden n>2.

Diseño y Maquetación: ***Antonio J. González Pérez***