# *Resolución numérica de ED y sistemas.*

***Capítulo 1 - Libro (1.7)*** **+** ***Capítulo 2- Libro*** **+** ***Hoja de problemas 4***

**Resumen de contenidos:**

- **ED de primer orden: problemas de Cauchy**
- **Soluciones explícitas y numéricas.**
- **Intervalos de definición de soluciones y control del paso**
- **Ecuaciones de segundo orden y sistemas**
- **Método de tiro para problemas de contorno**

*% En este cuaderno se trabaja principalmente con métodos numéricos para la resolución de problemas de valores iniciales. Se compararan soluciones explícitas con numéricas, y con aproximaciones de soluciones obtenidas por otros métodos. En particular, se analizan los teoremas de existencia y unicidad de solución, e intervalos de definición de soluciones mediante ejemplos.*

*% La idea principal del cuaderno es experimentar con los métodos numéricos de Euler y Runge-Kutta, de orden dos y cuatro, así como otros implementados en Matlab. Entre otras, utilizamos las funciones "eul", "rk2" y "rk4" proporcionadas por J. Polking (Rice University, Texas, USA), y "ode45", para resolver numéricamente problemas de Cauchy para ED y sistemas diferenciales. En este contexto, también se abordan problemas de contorno. El cuaderno se complementa con los cuadernos segundo y tercero del curso (ver observación final).*

*% Comenzamos con varios ejemplos sobre análisis de errores de redondeo y el significado sobre la dependencia continua de la solución con respecto a datos iniciales; también sobre el intervalo de definición de la solución.*

*% **Ejemplo 1:** Evaluamos el determinante 1000 *(6.010 * 6.000-2.004 * 18.04), dependiendo de que usemos para los cálculos una cifra decimal, dos o tres. Se trata del ejercicio 12 de la Sección 1.7 del libro de apuntes. Se tienen los resultados bien distintos:*

***>> (6.010*6.000-2.004*18.04)*1000***

ans =

-92.1600

```
>> (6.01*6.00-2.00*18.04)*1000

ans =

  -20.0000

>> (6.0*6.0-2.0*18.0)*1000

ans =

     0

% Así, es importante trabajar con la máxima precisión posible.

% Ejemplo 2: influencia sobre la solución de un error pequeño en el dato inicial.
  Dada una ecuación lineal y'=y+t-3, que se resuelve explícitamente en (-infinito,
  infinito), se comete un error de una milésima en la condición inicial (error de
  redondeo, de lectura u otro tipo), y analizamos la solución en puntos alejados del
  punto inicial

% A continuación, llamamos sol a la solución exacta (esto es, la solución analítica) y
  solap la solución exacta con el error en el dato inicial: calculamos explícitamente con
  "dsolve" y hacemos las gráficas en distintos intervalos. En particular, se comparan
  ambas soluciones en t=1 y t=log(10^6)

>> syms t

>> sol=dsolve('Dy=t+y-3', 'y(0)=2')

sol =

  2 - t


>> solap=dsolve('Dy=t+y-3', 'y(0)=2.001')

solap =

    exp(t)/1000 - t + 2

% La diferencia sol-solap= - exp(t)/1000. Comparamos las soluciones en t=1
```

***>> subs(solap,t,1)-subs(sol,t,1)***

ans =

0.0027

*% comparando las soluciones en t=log(10^6)*

***>> subs(solap,t,log(10^6))-subs(sol,t,log(10^6))***

ans =

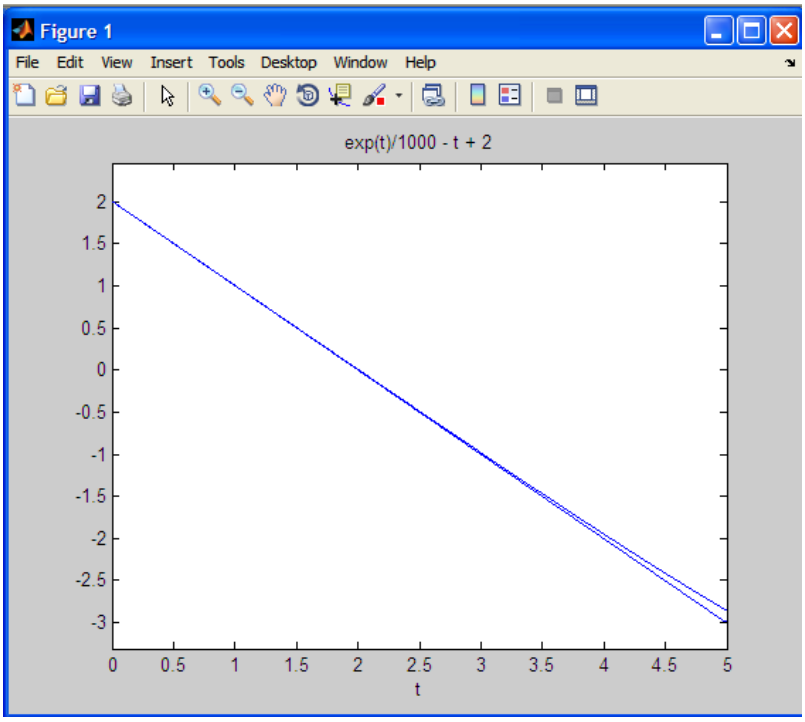
1.0000e+003

*% en cuanto a las gráficas, casi coinciden en [0,5] mientras que son muy distintas en intervalos más grandes*

***>> ezplot(sol,[0,5])***

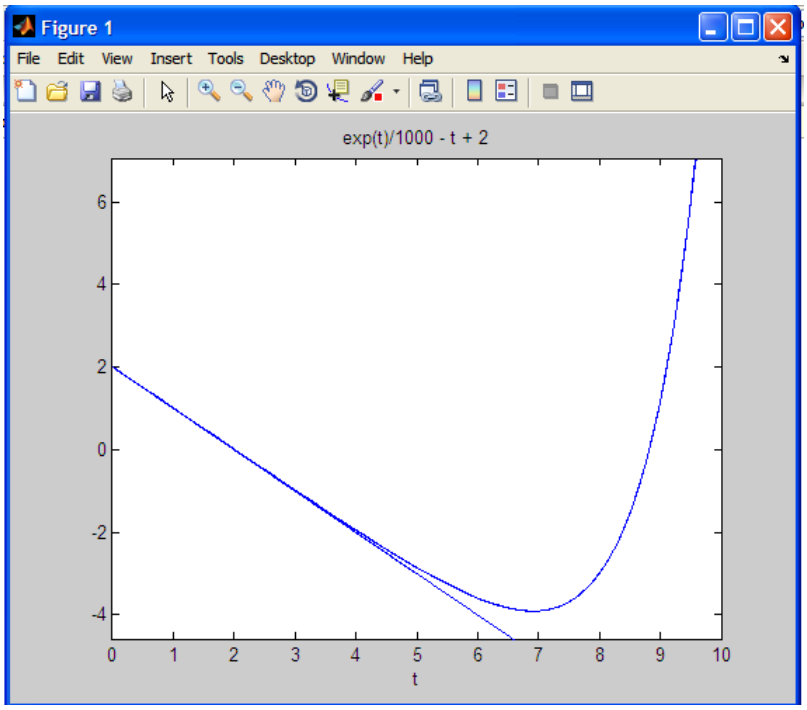
***>> hold on***

***>> ezplot(solap,[0,5])***



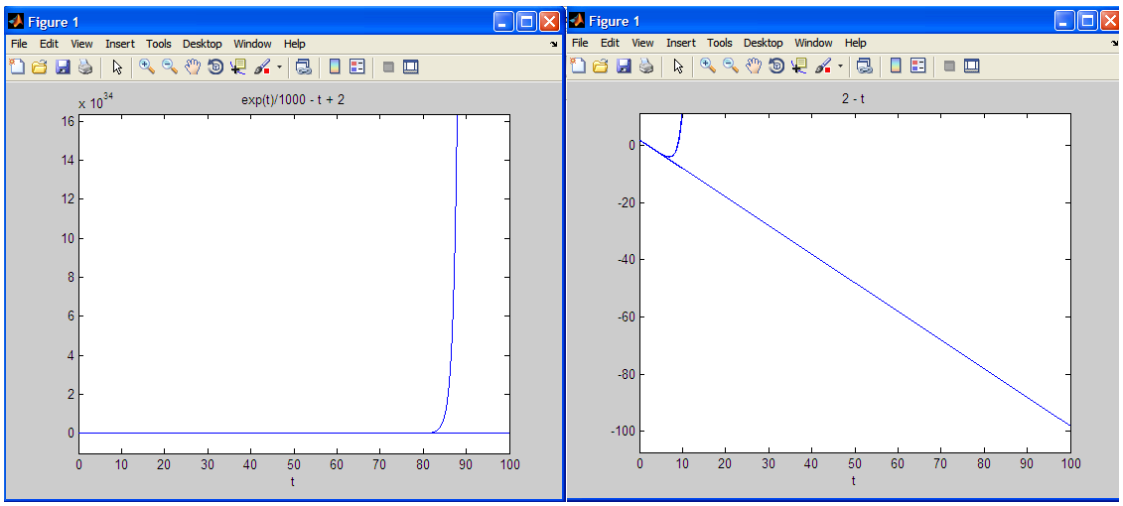
***>> ezplot(sol,[0,10])***

***>> ezplot(solap,[0,10])***




***>> ezplot(solap,[0,100])***

***>> ezplot(sol,[0,100])***



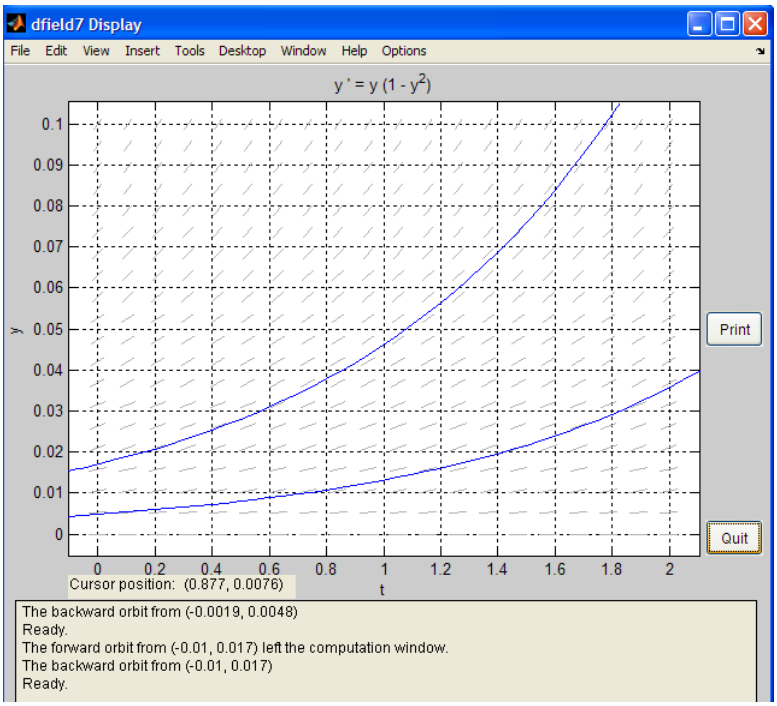

*% Se ven diferencias importantes entre la solución exacta y la solución aproximada a medida que agrandamos el intervalo de comparación: e.g., diferencias grandes en [0,100]. Si no se dan valores de referencias a los ejes, el resultado gráfico puede variar, como se aprecia en estas dos últimas gráficas.*

***>> hold off***

*% Ocurre que los errores en los datos iniciales pueden afectar a las soluciones considerablemente en puntos alejados del dato inicial. En el siguiente ejemplo (ya considerado en el tercer cuaderno) se observa el mismo efecto con el campo de direcciones asociado a una ecuación no lineal. Para las ED no lineales, de manera general, el error no está controlado por una función exponencial*

*% **Ejemplo 3**: Soluciones de y'=y(1-y^2) verificando 0<y(0)=<0.01 y con 0.01<y(0)<0.02 (t en [0,2])*

***>> dfield7***




*% Observamos como condiciones iniciales cercanas nos llevan a soluciones muy distintas a medida que nos alejamos de los puntos iniciales (comparar en t=2, por ejemplo)*

*% **Ejemplo 4:** Mostramos que la solución de un problema de valor inicial puede no estar definida en un intervalo elegido al azar (por muy regular que sea la ED). Para ello, consideramos el problema de Cauchy para una ED no lineal y'=t^2+y^2, y(0)=1.*

*% Matlab no resuelve explícitamente este problema con funciones elementales (ver segundo cuaderno). Intentamos aproximar la solución. Comparamos la aproximación obtenida con el campo de direcciones, con varios términos del desarrollo en serie de Taylor y con métodos numéricos*

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

ans =

(4*t*gamma(3/4)*besselk(-3/4, (t^2*i)/2)*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))

*% con esta solución de Matlab, estamos en el campo complejo y buscamos soluciones reales. La versión 2007 de Matlab nos da la solución real*

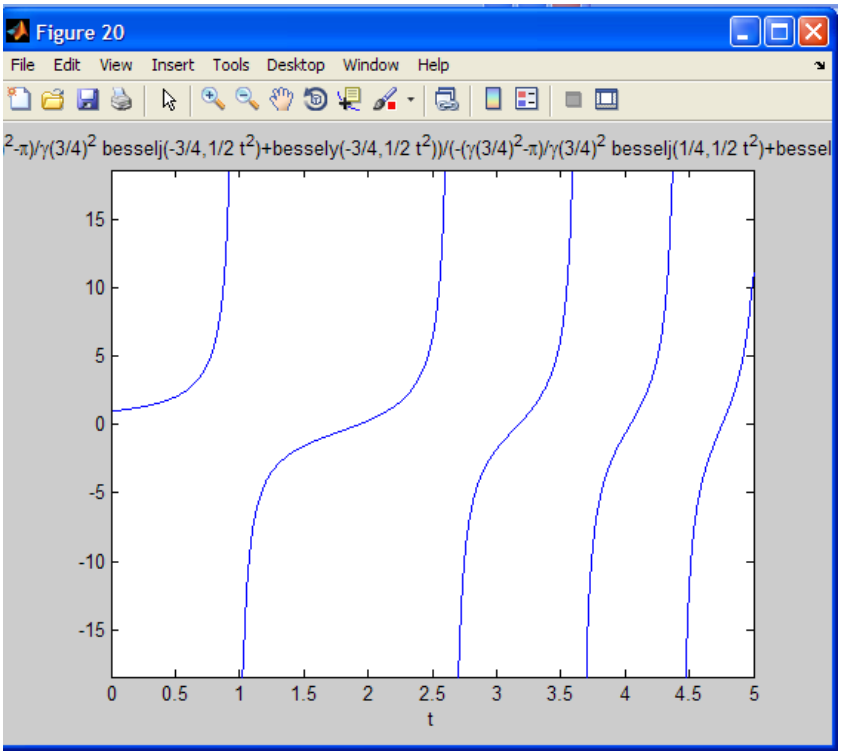
***>> dsolve('Dy=t^2+y^2','y(0)=1')*** *% con Matlab 2007*

ans =

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

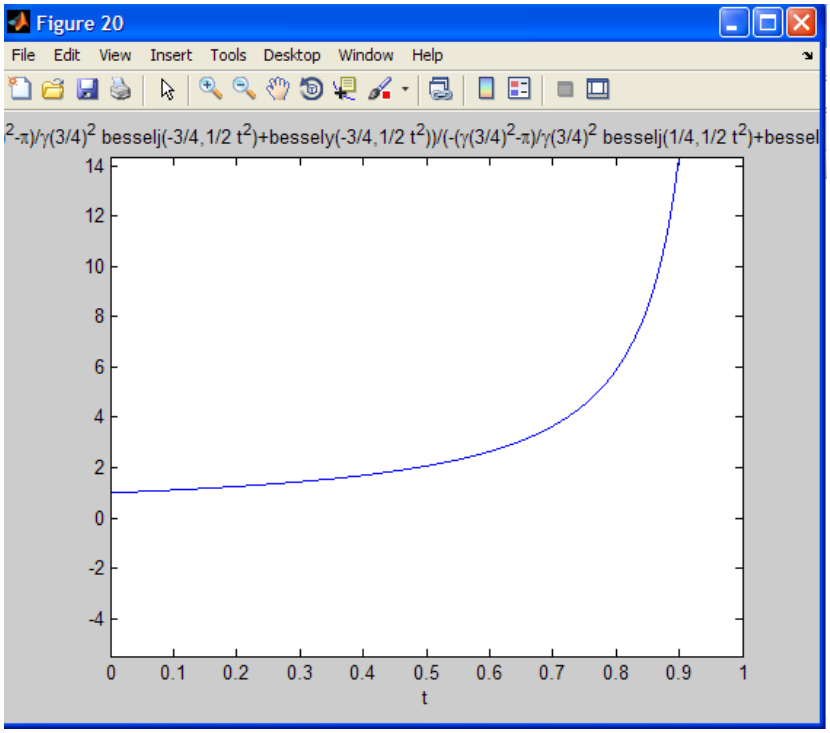
*% y al hacer la gráfica en el intervalo [0,5] vemos que hay asíntotas; esto es, la solución no está definida en todo el intervalo [0,5]*

***>> ezplot(ans,[0,5])***



*% La solución de nuestro problema de Cauchy debe estar definida en t=0, luego, en la gráfica, se trataría de la primera rama. Nos restringimos al intervalo [0,1]*

***>> ezplot(ans,[0,1])***




*% Comparamos los cinco primeros términos del desarrollo en serie de Taylor de la solución que nos da Matlab con los calculados a mano (esto es a partir de la ED y la condición inicial, y(0)+y'(0)t+(y''(0)/2)t^2+(y'''(0)/6)t^3+(y''''(0)/24))t^4): 1+t+t^2+(4/3)*t^3+(28/24)*t^4)*

***>> taylor(ans,5)*** *% con Matlab 2007*

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

***>> vpa(ans,5)*** *% visualizamos el resultado mejor*

ans =

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

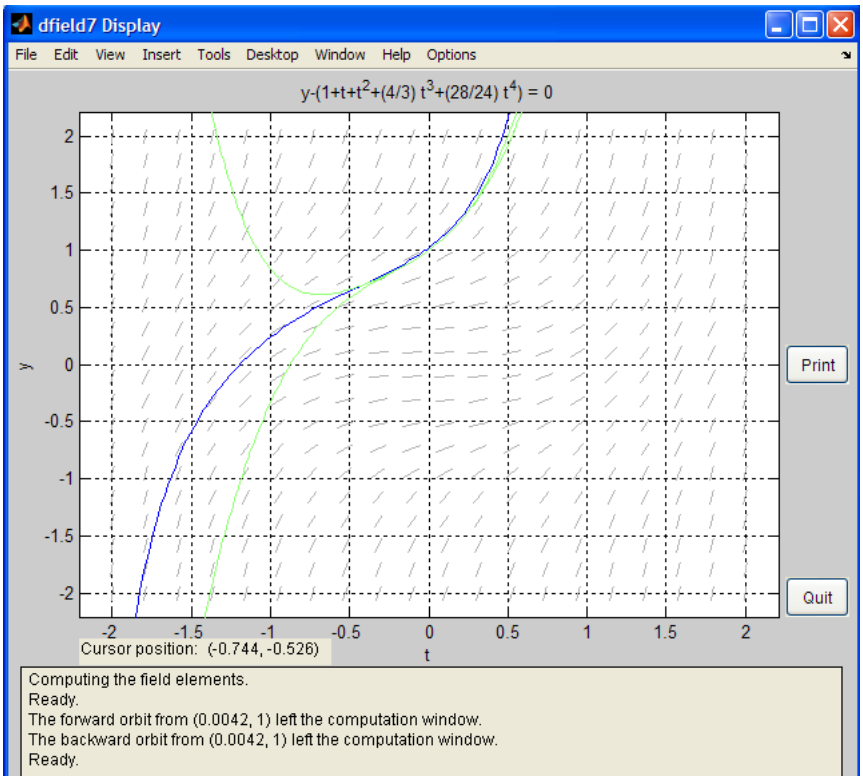
*% y vemos que la solución y(t) se aproxima por los primeros términos desarrollo en serie de Taylor 1+t+t^2+(4/3)*t^3+(28/24)*t^4)+…*

*% Abajo, comparamos gráficamente la solución numérica que nos proporciona el entorno "dfield" (en azul, tangente a los vectores del campo) con la aproximada por los distintos términos del desarrollo en serie (gráficas en verde)*

***>> dfield7***

***>> ezplot('y-(1+t+t^2+(4/3)*t^3)') )*** *% 4 términos del desarrollo en serie*

***>> ezplot('y-(1+t+t^2+(4/3)*t^3+(28/24)*t^4)')*** *% 5 términos del desarrollo en serie*




*% vemos que, a medida que aumentamos el número de términos del desarrollo, aumenta el intervalo de aproximación, pero no conocemos el error. A continuación trabajamos con los métodos numéricos que nos permiten obtener la gráfica en azul sin usar el entorno dfield.*

# *Problemas de Cauchy: aproximación numérica de la solución*

*% Además de las funciones Matlab eul.m, rk2.m y rk4.m, para aproximación numérica de soluciones de problemas de valores iniciales, se proporciona una función para definir nuestra ED: ejer10.m. Dicha función ejer10.m está asociada a la ED y'=t^2+y^2 (ejercicio 10, sección 1.7 del libro de apuntes),*

```
function dydt = ejer10(t,y)
dydt = t^2+y^2;
```

*y podemos modificar con el editor de Matlab para crear la función f(t,y) de la ED y'=f(t,y) con la que trabajemos. Por ejemplo, para la ED y'=t+y, modificamos cambiando t^2+y^2 por t+y, y el nombre de la función "ejer10" por "ejerlineal"*

***>> type ejer10***

***>> type ejerlineal***

*% De manera general las funciones Matlab "eul", "rk2" y "rk4" se utilizan como "ode45", añadiendo la posibilidad de introducir el tamaño del paso h:*

*[t,y]=eul('funcionf',[t0,t0+delta],y0,h)*

*resuelve numéricamente el problema de valor inicial y'=f(t,y), y(t0)=y0 en el intervalo [t0,t0+delta] (o [t0,t0-delta] si h negativo). funcionf es el nombre del fichero Matlab (M-file) en que hemos definido la función f(t,y). Lo que nos devuelve Matlab son los vectores t e y: y es la solución numérica para tamaño de paso h; t es el vector de abscisas donde aproximamos la solución, y es el vector ordenadas; número de componentes que depende del tamaño del paso (e.g., si proporcionamos h y delta/h es un número natural, dicho número es: (delta/h)+1))*

***>> type eul***

***>> type rk2***

***>> type rk4***

***>> help eul***

```
 eul         Integrates a system of ordinary differential equations using
             Euler's method.  See also ode45 and odedemo.
             [t,y] = eul('yprime', tspan, y0) integrates the system
             of ordinary differential equations described by the m-file
             yprime.m over the interval tspan = [t0,tfinal] and using initial
             conditions y0.
             [t, y] = eul(F, tspan, y0, ssize) uses step size ssize

 INPUT:
  F     - String containing name of user-supplied problem description.
        Call: yprime = fun(t,y) where F = 'fun'.
        t      - Time (scalar).
        y      - Solution vector.
        yprime - Returned derivative vector; yprime(i) = dy(i)/dt.
 tspan = [t0, tfinal], where t0 is the initial value of t, and tfinal is
           the final value of t.
 y0    - Initial value vector.
ssize - The step size to be used. (Default: ssize = (tfinal - t0)/100).


  OUTPUT:
 t  - Returned integration time points (column-vector).
 y  - Returned solution, one solution row-vector per tout-value.

 The result can be displayed by: plot(t,y).
```

***>> help ode45***

```
ode45  Solve non-stiff differential equations, medium order method.
    [TOUT,YOUT] = ode45(ODEFUN,TSPAN,Y0) with TSPAN = [T0 TFINAL] integrates
    the system of differential equations y' = f(t,y) from time T0 to TFINAL
    with initial conditions Y0. ODEFUN is a function handle. For a scalar T
    and a vector Y, ODEFUN(T,Y) must return a column vector corresponding
    to f(t,y). Each row in the solution array YOUT corresponds to a time
    returned in the column vector TOUT.  To obtain solutions at specific
    times T0,T1,...,TFINAL (all increasing or all decreasing), use TSPAN =
    [T0 T1 ... TFINAL].

    [TOUT,YOUT] = ode45(ODEFUN,TSPAN,Y0,OPTIONS) solves as above with default
    integration properties replaced by values in OPTIONS, an argument created
    with the ODESET function. See ODESET for details. Commonly used options
    are scalar relative error tolerance 'RelTol' (1e-3 by default) and vector
    of absolute error tolerances 'AbsTol' (all components 1e-6 by default).
    If certain components of the solution must be non-negative, use
    ODESET to set the 'NonNegative' property to the indices of these
    components.

    ode45 can solve problems M(t,y)*y' = f(t,y) with mass matrix M that is
    nonsingular. Use ODESET to set the 'Mass' property to a function handle
    MASS if MASS(T,Y) returns the value of the mass matrix. If the mass matrix
    is constant, the matrix can be used as the value of the 'Mass' option. If
    the mass matrix does not depend on the state variable Y and the function
    MASS is to be called with one input argument T, set 'MStateDependence' to
    'none'. ODE15S and ODE23T can solve problems with singular mass matrices.

……..
……..
```

## *Ejercicio 1*

*% Resolución y aproximación de la solución de y'=t+y, y(0)=1 en [0,3]. Se compara la solución explícita con "dsolve" y la numérica con distintos métodos implementados en "eul", "rk2", "rk4" y "ode45". Comparamos el error con gráficas, y el error en t=3*

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

ans =

 -1-t+2*exp(t)

% valor exacto (con los errores de redondeo del ordenador)

>> v_exacto=-1-3+2*exp(3)

v_exacto =

  36.1711

>> ezplot(ans,[0,3])

>> hold on

% en el fichero ejerlineal.m introducimos las instrucciones
```

```
function dydt = ejerlineal(t,y)
dydt = t+y;
```

```
>> [t,y]=eul('ejerlineal',[0,3],1,0.1);[t,y]

ans =

         0    1.0000
    0.1000    1.1000
    0.2000    1.2200
    0.3000    1.3620
    0.4000    1.5282
    0.5000    1.7210
    0.6000    1.9431
    0.7000    2.1974
    0.8000    2.4872
    0.9000    2.8159
```

```
   1.0000    3.1875
   1.1000    3.6062
   1.2000    4.0769
   1.3000    4.6045
   1.4000    5.1950
   1.5000    5.8545
   1.6000    6.5899
   1.7000    7.4089
   1.8000    8.3198
   1.9000    9.3318
   2.0000   10.4550
   2.1000   11.7005
   2.2000   13.0805
   2.3000   14.6086
   2.4000   16.2995
   2.5000   18.1694
   2.6000   20.2364
   2.7000   22.5200
   2.8000   25.0420
   2.9000   27.8262
   3.0000   30.8988
```

*% la primera columna contiene el vector abscisa ("la discretización de t"), y la segunda, el vector ordenada, esto es, el valor aproximado de la solución en t. Por ejemplo, con el método de Euler y para el tamaño de paso h=0.1, en t=2, la aproximación de la solución es 10.4550 mientras que en t=3, la aproximación es 30.8988.*

*% Repetimos para leer la aproximación en t=3, para distintos tamaños de paso y con distintos métodos*

***>> [t,y]=eul('ejerlineal',[0,3],1,0.1);[t,y]***

```
ans =

   0         1.0000
…….       …….
…….       …….
3.0000   30.8988
```

***>> v_exacto-30.8988***

```
ans =

   5.2723
```

***>> hold on***

***>> plot(t,y,'--')*** *% gráfica en línea discontinua*

***>> gtext('h=0.1')*** *% añadimos comentarios a las gráficas con gtext*

***>> gtext('eul')***

*% disminuimos el tamaño del paso h y comparamos*

***>> [t,y]=eul('ejerlineal',[0,3],1,0.05);[t,y]***

```
ans=
   0        1.0000
.........    ..........
..........   ..........
3.0000   33.3584
```

***>> plot(t,y,'g')***

***>> gtext('h=0.05 eul')***

***>> 33.3584-v_exacto***

```
ans =

  -2.8127
```

*% seguimos disminuyendo h y comparando*

***>> [t,y]=eul('ejerlineal',[0,3],1,0.01);[t,y]***

```
ans =

   0        1.0000
  .........  ..........
  .........  ..........
 3.0000   35.5769
```

***>> v_exacto-35.5769***

```
ans =

   0.5942
```

***>> plot(t,y,'y')***

***>> gtext('h=0.01_eul')***

*% cambiamos de método (método de Euler mejorado) para el primer h, h=0.1, y comparamos*

***>> [t,y]=rk2('ejerlineal',[0,3],1,0.1);[t,y]***

```
ans =

      0       1.0000
  0.1000      1.1100
   ……..       ……..
   ……..       ……..
  2.9000     32.2856
  3.0000     35.9851
```

***>> v_exacto-35.9851***

```
ans =

   0.1860
```

*% Así, con rk2 y h=0.1 tenemos un error mucho más pequeño que con eul y h=0.001; menos operaciones también.*

*% Cambiamos de método (método de Runge-Kutta) para el mismo h, h=0.1, y comparamos*

***>> [t,y]=rk4('ejerlineal',[0,3],1,0.1);[t,y]***

```
ans =

   0        1.0000
……..        ……..
……..        ……..
3.0000    36.1710
```

***>> v_exacto-36.1710***

```
ans =

  7.3846e-005
```

*% con rk4 y h=0.1 tenemos una aproximación mucho mejor que con "eul" y h=0.001 y que con rk2 y h=0.1.*

***>> format long***

***>> [t,y]=rk4('ejerlineal',[0,3],1,0.1);[t,y]***

```
ans =

   0                   1.000000000000000
……….                  ……..
……….                  ……..
3.000000000000000  36.170981439329736
```

***>> v_exacto-36.170981439329736***

```
ans =

   9.240704559942969e-005
```

*% Cambiamos a ode45 y comparamos.*

***>> [t,y]=ode45('ejerlineal',[0,3],1);[t,y]***

```
ans =

   0                   1.000000000000000
……….                  ……..
……….                  ……..
3.000000000000000  36.171116531462054
```

***>> v_exacto-36.171116531462054***

```
ans =

  -4.268508671856353e-005
```

***>> [t,y]=rk4('ejerlineal',[0,3],1,0.05);[t,y]***

```
ans =

   0                   1.000000000000000
……….                  ……..
……….                  ……..
3.000000000000000  36.171067825652521
```

***>> v_exacto-36.171067825652521***

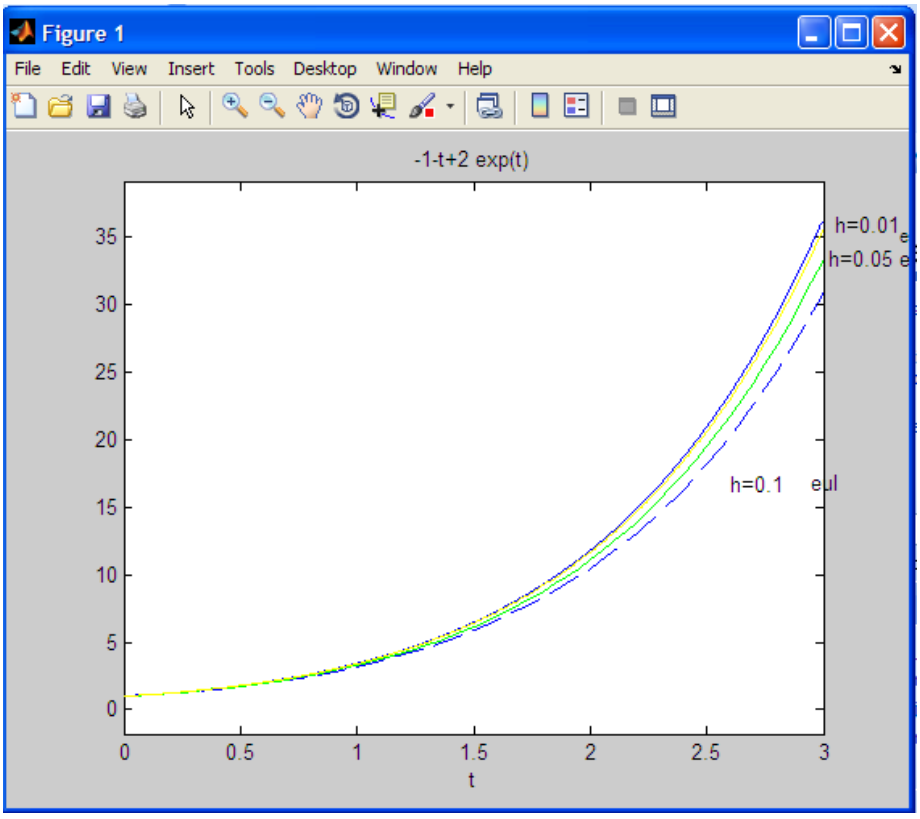
ans =

6.020722814525925e-006

*% con ode45 el error es del mismo orden que con rk4 y h=0.1. El error es más grande que con rk4 y h=0.05, pero éste último necesita más operaciones.*

*% Para ode45 y otras funciones Matlab como ode23 (métodos Runge-Kutta, con control de paso) : ver la guía de Matlab, y las direcciones de internet*
*http://www.mathworks.es/es/help/matlab/ref/ode45.html*
*http://es.m.wikipedia.org/wiki/M%C3%A9todo_de_Dormand-Prince*
*http://www.mathworks.es/es/help/matlab/ref/ode23.html*

*% Abajo, en la gráfica, la línea continua azul muestra la solución exacta, las otras son las aproximaciones con el método de Euler para distintos valores de h. Se ve que a partir de un h las gráficas se superponen y no tiene sentido hacer más. Aunque gráficamente parece que no hay error, numéricamente se ha visto que sí lo hay.*



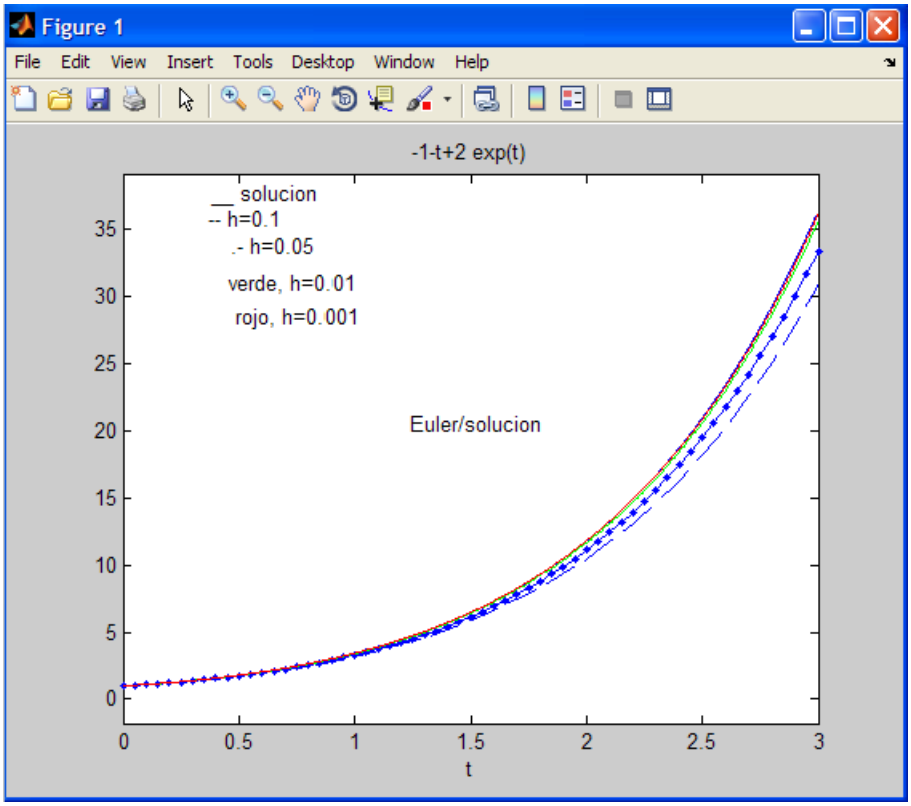

*% Abajo, con la misma idea, gráficas de la solución y aproximaciones con el método de Euler*

***>> [t,y]=eul('ejerlineal',[0,3],1,0.001);[t,y]***

ans =

| | |
|---|---|
| 0 | 1.0000 |
| 0.0010 | 1.0010 |
| 0.0020 | 1.0020 |
| 0.0030 | 1.0030 |
| ..... | ..... |
| ..... | ..... |
| 2.9940 | 35.8771 |
| 2.9950 | 35.9159 |
| 2.9960 | 35.9549 |
| 2.9970 | 35.9938 |
| 2.9980 | 36.0328 |
| 2.9990 | 36.0718 |
| 3.0000 | 36.1109 |

***>> plot(t,y,'r')***

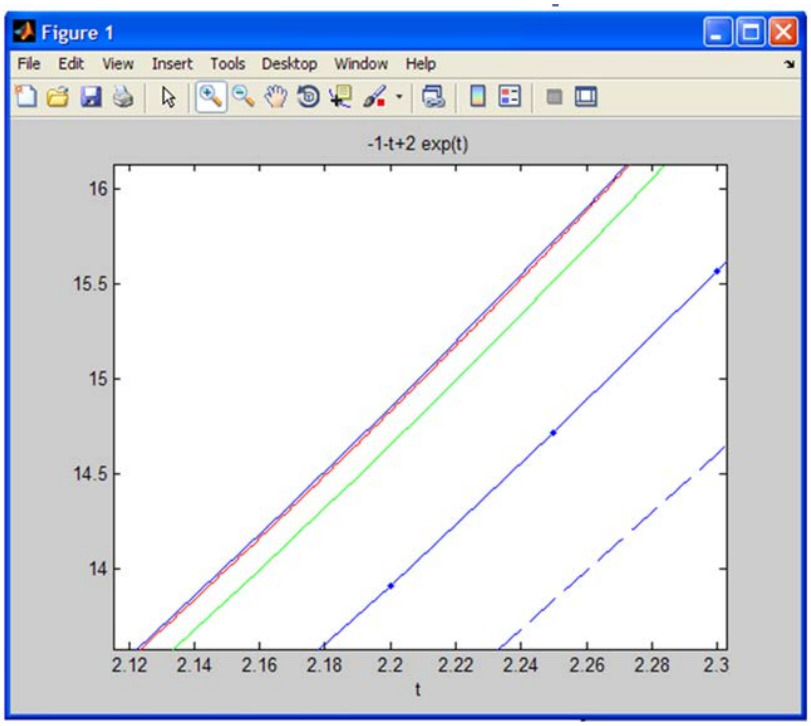


***>> 36.1109-(2*exp(3) - 3 - 1)*** *% error en t=3, con eul y h=0.001*

ans =

-0.0602

*% El error en t=3, con eul y h=0.001, es más pequeño que con rk2 y h=0.1. Aunque aparentemente la gráfica de esta aproximación (en rojo) se superpone con la de la solución (línea continua en azul), la ampliación de la figura (abajo) muestra que son distintas*




*% **Observación:** En la resolución de la ED lineal y'= t+y, se ha podido medir el error (solución exacta- aproximada) y comprobar que el error es menor a medida que disminuye el tamaño del paso h, y para el mismo h, el error es mayor a medida que nos alejamos del punto inicial. Esto es lo que muestra la fórmula del el error del método (ver libro de apuntes)*

$$|e_N| \leq \frac{Ch}{2L}(e^{\delta L} - 1)$$

*donde C y L son ciertas constantes y delta mide la amplitud del intervalo de aproximación. Sin embargo, al disminuir el h demasiado, pueden aumentar los errores de redondeo (ver bibliografía en el libro de apuntes)*

*% Realmente, para medir el error hay que evaluar (solución exacta- aproximada) en cada t. El fichero ejecutoeul1213.m contiene las instrucciones para esta comparación: nos da el máximo del valor absoluto de los errores para cada tamaño de paso, esto es, un vector con tamaños de paso, y un vector con errores; además se comparan las gráficas*

```
t=0:0.05:3; x=-1-t+2*exp(t);
h_vector=[h_vector,h];
 [s,y]=eul('ejerlineal',[0,3],1,h);
plot(t,x,s,y,'o');
xlabel('variable independiente');ylabel('solucion y(t)');
title('solucion exacta --, numerica o')
z=-1-s+2*exp(s);
maxerror=max(abs(z-y));
err_vector=[err_vector,maxerror];
h_vector
err_vector
h=h/2;
```

***>> type ejecutoeul1213***

*% Para ejecutar el programa, inicializamos h*

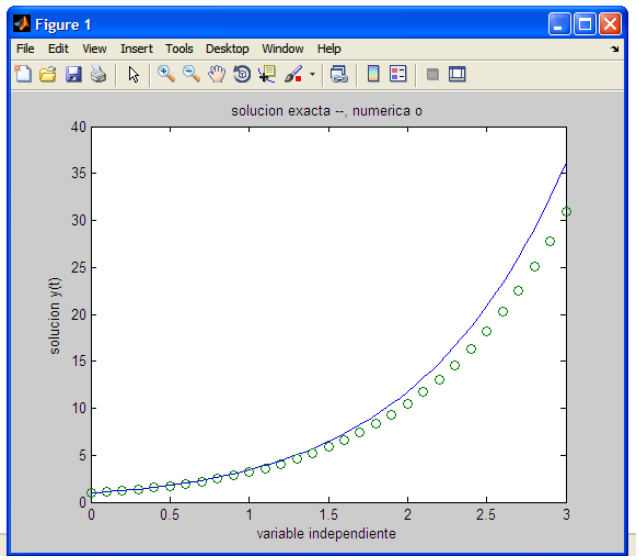
***>> h_vector=[]; err_vector=[]; h=0.1***

```
h =

    0.1000
```

***>> ejecutoeul1213***

```
h_vector =

    0.1000


err_vector =

    5.2723
```

```
>> h_vector=[]; err_vector=[]; h=0.1

h =

   0.100000000000000

>> format short
>> %inicializando  vemos los errores con ejecutoeul
>> h_vector=[]; err_vector=[]; h=0.1

h =

    0.1000

>> ejecutoeul1213

h_vector =

    0.1000

err_vector =

    5.2723

>>
```




***>> ejecutoeul1213***

h_vector =

0.1000 0.0500

err_vector =

5.2723 2.8127

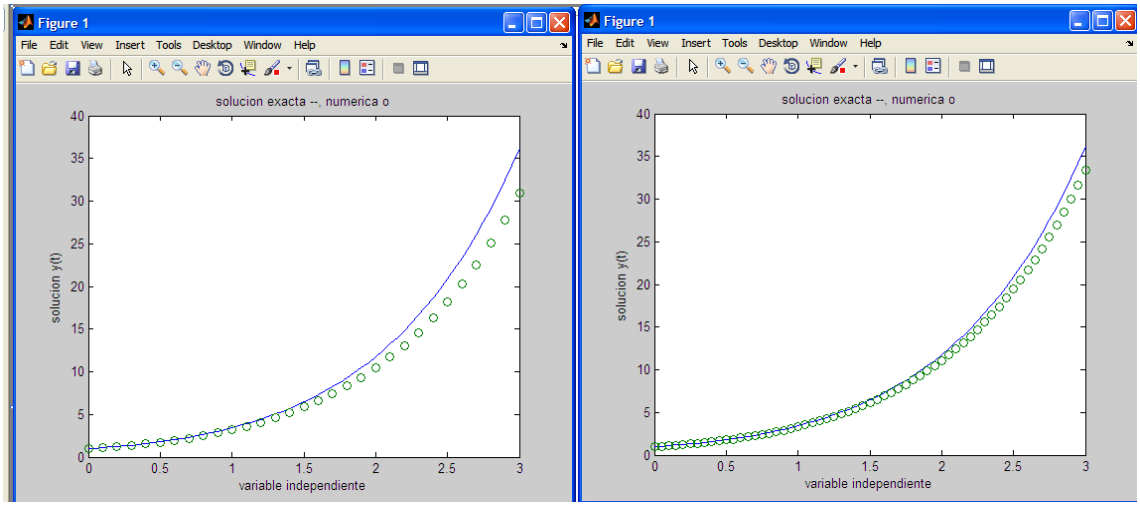
***>> ejecutoeul1213***

h_vector =

0.1000 0.0500 0.0250

err_vector =

5.2723 2.8127 1.4548




***>> ejecutoeul1213***

h_vector =

0.1000 0.0500 0.0250 0.0125

err_vector =

5.2723 2.8127 1.4548 0.7401

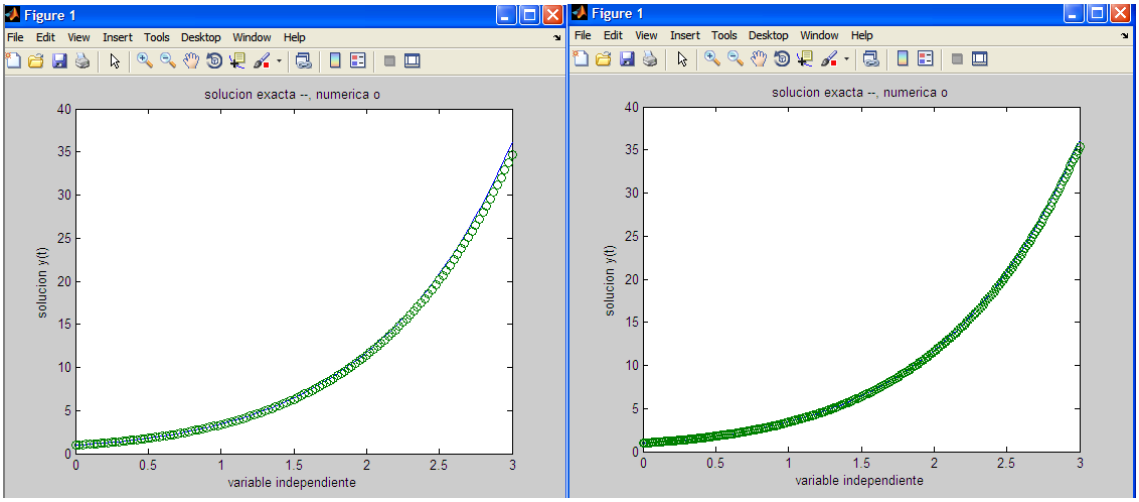
***>> ejecutoeul1213***

h_vector =

0.1000 0.0500 0.0250 0.0125 0.0063

err_vector =

5.2723 2.8127 1.4548 0.7401 0.3733




*% De nuevo vemos que el error es mayor a medida que nos alejamos del dato inicial. A partir de aquí ya no se observa variación en las gráficas, pero vemos que sí se comete error (las componentes del vector err_vector no se anulan)*

***>> ejecutoeul1213***

h_vector =

0.1000 0.0500 0.0250 0.0125 0.0063 0.0031

err_vector =

5.2723 2.8127 1.4548 0.7401 0.3733 0.1875

***>> ejecutoeul1213***

h_vector =

0.1000 0.0500 0.0250 0.0125 0.0063 0.0031 0.0016

err_vector =

5.2723 2.8127 1.4548 0.7401 0.3733 0.1875 0.0939

***>> ejecutoeul1213***

h_vector =

0.1000 0.0500 0.0250 0.0125 0.0063 0.0031 0.0016 0.0008

err_vector =

5.2723 2.8127 1.4548 0.7401 0.3733 0.1875 0.0939 0.0470

***>> ejecutoeul1213***

h_vector =

0.1000 0.0500 0.0250 0.0125 0.0063 0.0031 0.0016 0.0008 0.0004

err_vector =

5.2723 2.8127 1.4548 0.7401 0.3733 0.1875 0.0939 0.0470 0.0235

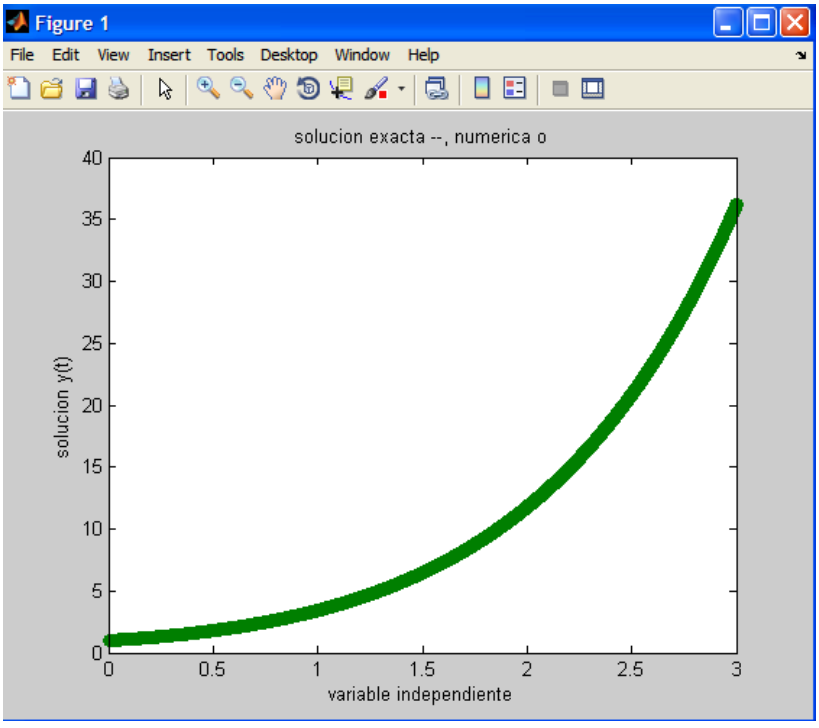
***>> ejecutoeul1213***

h_vector =

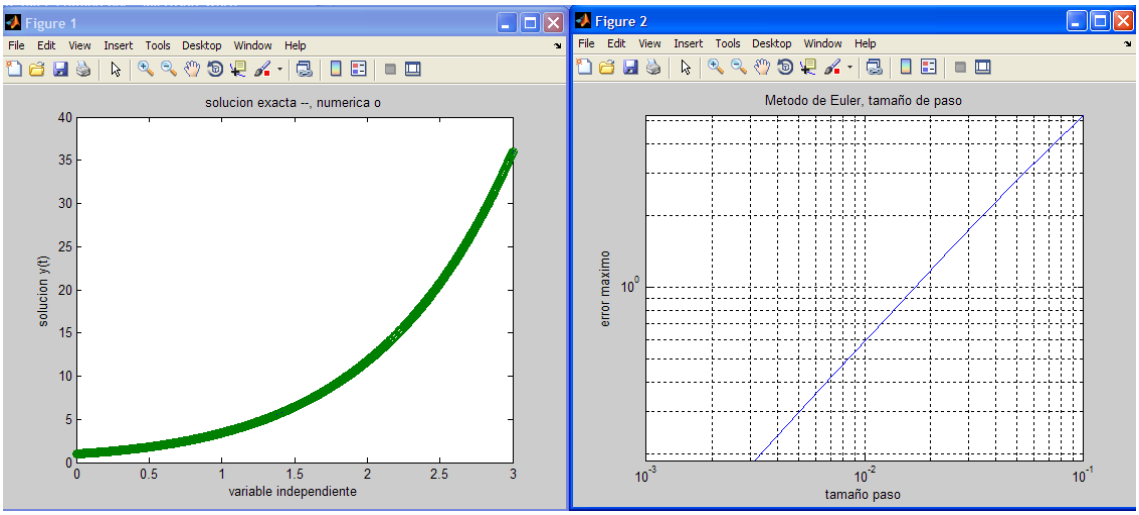
0.1000 0.0500 0.0250 0.0125 0.0063 0.0031 0.0016 0.0008 0.0004 0.0002

err_vector =

5.2723 2.8127 1.4548 0.7401 0.3733 0.1875 0.0939 0.0470 0.0235 0.0118

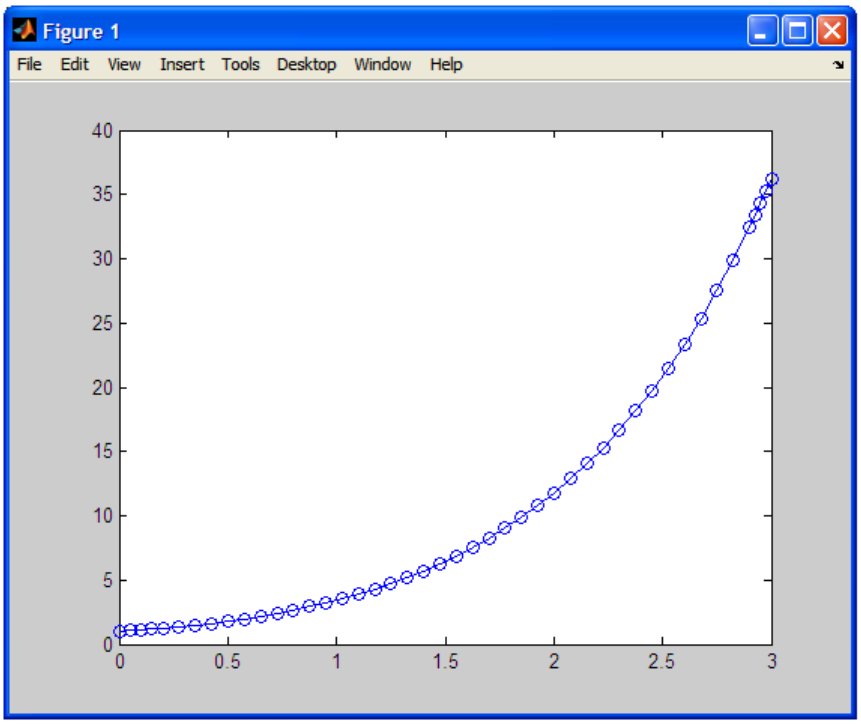


*% Como se ha observado, la gráfica de comparación de soluciones no presenta variaciones a partir de un h, pero podemos hacer una gráfica en una escala logarítmica (abajo, a la derecha) donde se muestra el error en términos del paso*




*% **Nota:** en diversas versiones de Matlab (R2008a,R2011b, e.g.) la instrucción*

***>> ode45('ejerlineal',[0,3],1)***

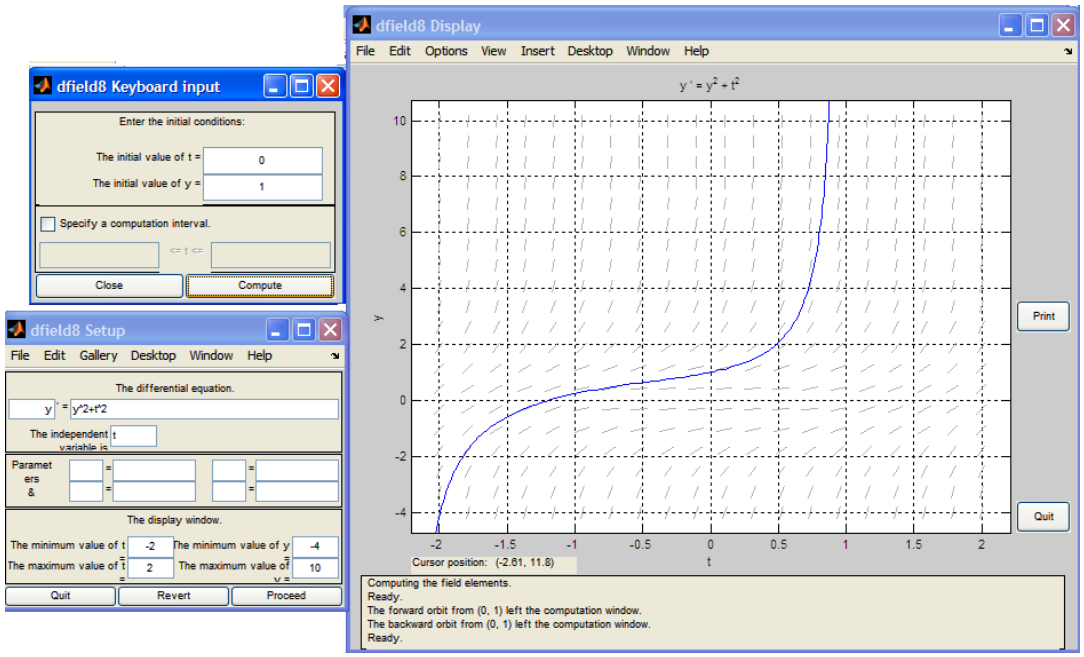



*% nos da la gráfica de la solución numérica en dos formatos: en línea continua y con puntos. Por defecto, los "circulitos" nos indican el tamaño del paso utilizado, y vemos que no es constante.*

## *Ejercicio 2*

*% Aproximación de la solución del problema de Cauchy: y'=t^2+y^2, y(0)=1. Como se ha visto en el ejemplo 4, al principio del cuaderno, la solución de este problema (que se sabe que existe y es única) puede presentar una asíntota en [0,1]. Nos planteamos encontrarla numéricamente. También analizamos cuestiones relacionadas con el tamaño del paso y el error, dado que no podemos comparar con la solución exacta.*

*% Un primer dibujo con "dfield", nos indica que la solución no está definida en todo el intervalo [0,1], es decir, parece que tiene una asíntota*




*% Hacemos la gráfica de la solución, resolviendo numéricamente. Se trata del ejercicio 10 de la sección 1.7 del libro de apuntes (ver también Ejemplo 13 en el libro)*

10. Se considera el problema de Cauchy

$$\begin{cases} y' &= x^2 + y^2 \\ y(0) &= 1, \end{cases}$$

a). Encontrar, mediante el método de Euler, los valores aproximados de la solución en los puntos $x = i \times h$, $i = 1, 2, 3, 4, 5$ para los valores del paso $h = 0.1$ y $h = 0.05$

b). Intentar encontrar la aproximación numérica de la solución para los pasos $h = 0.1$, $h = 0.05$, en el intervalo $[0, 1]$. ¿Qué se observa cerca del punto $x = 1$ ?. Encontrar el tamaño del paso tal que $|\varphi_h(0, 9) - 14.3| \leq 0.05$.

c). Utilizando como valor inicial el valor aproximado de la solución en $x = 0.9$, obtenido en el apartado anterior (puede, por ejemplo, tomarse el valor $y(0.9) \approx 14.27$, obtenido mediante el método de Runge-Kutta para el paso $h = 0.05$), aplicar el método de Euler para los pasos $h = 0.01$ y $h = 0.005$, para intentar aproximar la solución del problema dado, en el intervalo $[0.9, 1]$. Utilizando los resultados obtenidos, dar un valor aproximado de $\omega^+$ (ver ejemplo 13).

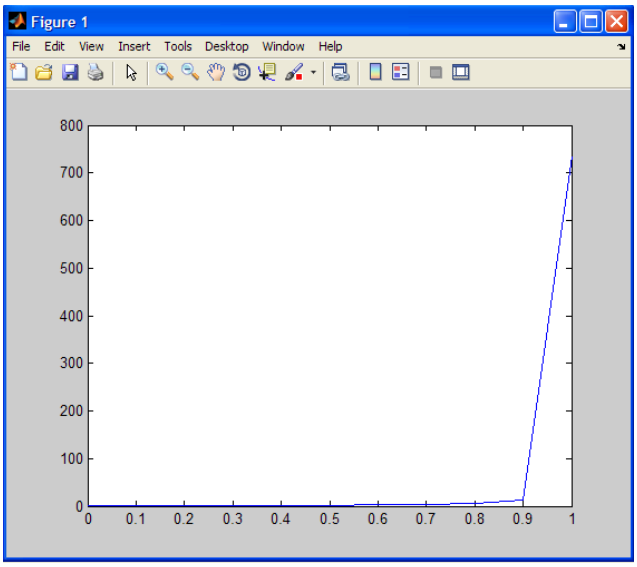
*% Intentamos aproximar la solución en [0,1] con el método de Runge-Kutta (que es el que nos da menor error para el mismo tamaño de paso); previamente, en ejer10.m se ha definido la función f(t,y)=t^2+y^2*

***>> [t,y]=rk4('ejer10',[0,1],1,0.1);[t,y]*** *% h=0.1*

ans =

| | |
|---|---|
| 0 | 1.0000 |
| 0.1000 | 1.1115 |
| 0.2000 | 1.2530 |
| 0.3000 | 1.4397 |
| 0.4000 | 1.6961 |
| 0.5000 | 2.0670 |
| 0.6000 | 2.6439 |
| 0.7000 | 3.6522 |
| 0.8000 | 5.8420 |
| 0.9000 | 14.0218 |
| 1.0000 | 735.0991 |

***>> plot(t,y)***



*% Parece una aproximación muy mala, pues crece muy deprisa entre 0.9 y 1; de ahí la gráfica. Hay que disminuir h; se repite cambiando h por h/2*

***>> [t,y]=rk4('ejer10',[0,1],1,0.05);[t,y]***

```
ans =

  1.0e+005 *

        0         0.0000
   0.0000    0.0000
   ……..       ……..
   ……..       ……..
   0.0000    0.0005
   0.0000    1.7586
```

*% no se leen bien los valores numéricos; lo ponemos en formato para lectura más cómoda*

***>> format short g***

***>> [t,y]=rk4('ejer10',[0,1],1,0.05);[t,y]***

```
ans =
         0         1
      0.05    1.0527
       0.1    1.1115
      0.15    1.1777
       0.2     1.253
      0.25    1.3395
       0.3    1.4397
      0.35     1.557
       0.4    1.6961
      0.45    1.8632
       0.5     2.067
      0.55    2.3206
       0.6     2.644
      0.65    3.0694
       0.7    3.6529
      0.75    4.5015
       0.8    5.8481
      0.85    8.3128
       0.9    14.271
      0.95    46.578
         1  1.7586e+005
```

***>> plot(t,y)***

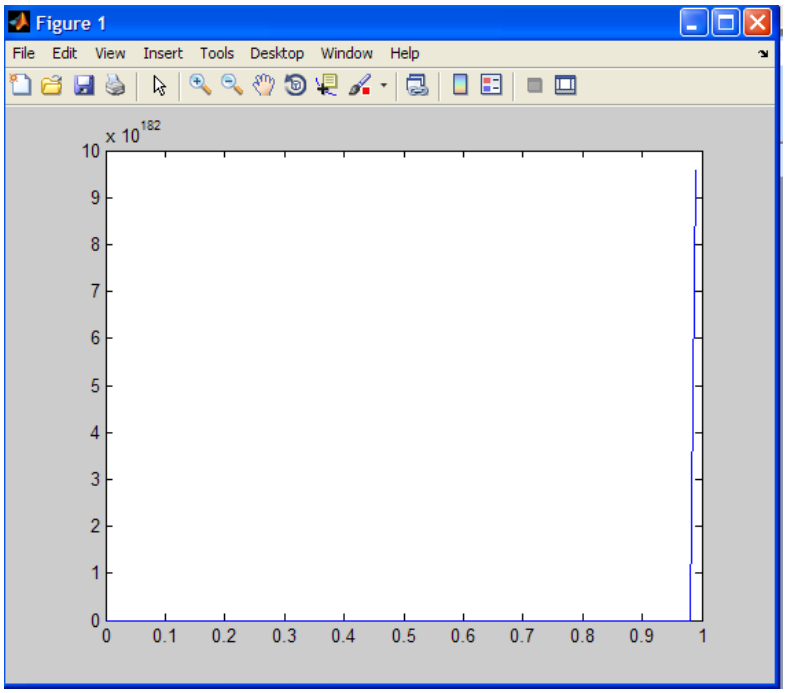
*% Seguimos teniendo una gráfica inapropiada…Seguimos disminuyendo h*

***>> [t,y]=rk4('ejer10',[0,1],1,0.01);[t,y]***

ans =

| 0 | 1 |
|---|---|
| 0.01 | 1.0101 |
| 0.02 | 1.0204 |
| 0.03 | 1.0309 |
| 0.04 | 1.0417 |
| 0.05 | 1.0527 |
| 0.06 | 1.0639 |
| 0.07 | 1.0754 |
| 0.08 | 1.0871 |
| 0.09 | 1.0992 |
| 0.1 | 1.1115 |
| 0.11 | 1.1241 |
| 0.12 | 1.137 |
| …… | …… |
| …… | …… |
| 0.85 | 8.3155 |
| 0.86 | 9.0777 |
| 0.87 | 9.9923 |
| 0.88 | 11.11 |
| 0.89 | 12.508 |
| 0.9 | 14.305 |
| 0.91 | 16.702 |
| 0.92 | 20.061 |
| 0.93 | 25.106 |
| 0.94 | 33.531 |
| 0.95 | 50.436 |
| 0.96 | 101.16 |
| 0.97 | 899.78 |
| 0.98 | 3.8528e+013 |
| 0.99 | 9.5921e+182 |
| 1 | Inf |

***>> plot(t,y)***




*% Con cualquier h que tomemos, parece que no vamos a poder aproximar la solución más allá de t=0.96, puesto que la aproximación (y/o la solución) crece muy deprisa entre 0.96 y 1; para h=0.01, la aproximación en t=1 nos da infinito (Inf)*

*para h=0.05*

| | |
|---|---|
| 0.95 | 46.578 |
| 1 | 1.7586e+005 |

*para h=0.05*

| | |
|---|---|
| 0.96 | 101.16 |
| 0.97 | 899.78 |
| 0.98 | 3.8528e+013 |
| 0.99 | 9.5921e+182 |
| 1 | Inf |

*% Dado que no podemos establecer comparaciones de la solución numérica con la exacta, nos preguntamos con qué h nos quedamos para tener una buena aproximación de la solución. Esto puede depender de los valores numéricos del problema.*

% Para que el h no sea muy pequeño, tomamos el intervalo [0,0.9], donde la solución no crece tan deprisa, y luego un h tal que |y_h(0.9)-y_{h/2}(0.9)| <0.05 (establecemos una "tolerancia de error"). Este valor 0.05 es un error "de tolerancia" que podemos poner dependiendo del problema (esto es, "relativizando" el error). Y, entre los h que satisfacen esta relación, tomamos el h más grande (ahorrando cálculos, e.g.)

*% Tomamos [0,0.9], y empezamos con h=0.1*

***>> [t,y]=rk4('ejer10',[0,0.9],1,0.1);[t,y]***

ans =

```
      0    1.0000
 0.1000    1.1115
 0.2000    1.2530
 0.3000    1.4397
 0.4000    1.6961
 0.5000    2.0670
 0.6000    2.6439
 0.7000    3.6522
 0.8000    5.8420
 0.9000   14.0218
```

*% disminuimos h a la mitad*

***>> [t,y]=rk4('ejer10',[0,0.9],1,0.05);[t,y]***

ans =

```
      0    1.0000
 0.0500    1.0527
 0.1000    1.1115
 0.1500    1.1777
 0.2000    1.2530
 0.2500    1.3395
 0.3000    1.4397
 0.3500    1.5570
 0.4000    1.6961
 0.4500    1.8632
 0.5000    2.0670
 0.5500    2.3206
 0.6000    2.6440
 0.6500    3.0694
 0.7000    3.6529
 0.7500    4.5015
 0.8000    5.8481
 0.8500    8.3128
 0.9000   14.2712
```

*% como |y_0.1(0.9)-y_0.05(0.9)|= |14.0218- 14.2712| > 0.05 continuamos disminuyendo el h*

***>> [t,y]=rk4('ejer10',[0,0.9],1,0.01);[t,y]***

ans =

| 0 | 1 |
|---|---|
| 0.01 | 1.0101 |
| 0.02 | 1.0204 |
| 0.03 | 1.0309 |
| 0.04 | 1.0417 |
| 0.05 | 1.0527 |
| 0.06 | 1.0639 |
| 0.07 | 1.0754 |
| …… | …… |
| …… | …… |
| 0.33 | 1.5078 |
| 0.34 | 1.532 |
| …… | …… |
| …… | …… |
| 0.83 | 7.1176 |
| 0.84 | 7.6705 |
| 0.85 | 8.3155 |
| 0.86 | 9.0777 |
| 0.87 | 9.9923 |
| 0.88 | 11.11 |
| 0.89 | 12.508 |
| 0.9 | 14.305 |

*% como |y_0.05(0.9)-y_0.01(0.9)|= |14.2712 -14.305|< 0.05, nos quedamos con h=0.05 para aproximar la solución.*

*% De hecho, como se ve abajo, por mucho que disminuyamos el paso ya, no obtenemos mucha mejor aproximación a no ser que pongamos el error de tolerancia más pequeño y pidamos más cifras decimales de precisión cambiando el formato….*

***>> [t,y]=rk4('ejer10',[0,0.9],1,0.005);[t,y]***

ans =

| 0 | 1 |
|---|---|
| 0.005 | 1.005 |
| 0.01 | 1.0101 |
| 0.015 | 1.0152 |
| 0.02 | 1.0204 |
| 0.025 | 1.0256 |
| 0.03 | 1.0309 |

| | |
|---|---|
| ..... | ..... |
| ..... | ..... |
| 0.8 | 5.8486 |
| 0.805 | 6.0281 |
| 0.81 | 6.2188 |
| 0.815 | 6.4218 |
| ..... | ..... |
| ..... | ..... |
| 0.875 | 10.522 |
| 0.88 | 11.11 |
| 0.885 | 11.768 |
| 0.89 | 12.508 |
| 0.895 | 13.346 |
| 0.9 | 14.305 |

***>> [t,y]=rk4('ejer10',[0,0.9],1,0.001);[t,y]***

ans =

| | |
|---|---|
| 0 | 1 |
| 0.001 | 1.001 |
| 0.002 | 1.002 |
| 0.003 | 1.003 |
| ..... | ..... |
| ..... | ..... |
| 0.893 | 12.998 |
| 0.894 | 13.17 |
| 0.895 | 13.346 |
| 0.896 | 13.528 |
| 0.897 | 13.714 |
| 0.898 | 13.905 |
| 0.899 | 14.102 |
| 0.9 | 14.305 |

***>> format long g***

***>> [t,y]=rk4('ejer10',[0,0.9],1,0.005);[t,y]***

ans =

| | |
|---|---|
| 0 | 1 |
| 0.005 | 1.0050251673995 |
| ..... | ..... |
| ..... | ..... |
| 0.83 | 7.11759331327505 |

| | |
|---|---|
| 0.835 | 7.3838355657157 |
| 0.84 | 7.6705337984891 |
| 0.845 | 7.9801441962061 |
| 0.85 | 8.3155329292533 |
| 0.855 | 8.68006542788068 |
| 0.86 | 9.07772004566337 |
| 0.865 | 9.51323425677129 |
| 0.87 | 9.99229479540636 |
| 0.875 | 10.5217879588312 |
| 0.88 | 11.110133520046 |
| 0.885 | 11.7677367540758 |
| 0.89 | 12.5076103744221 |
| 0.895 | 13.3462458703378 |
| 0.9 | 14.3048592806911 |

***>> [t,y]=rk4('ejer10',[0,0.9],1,0.001);[t,y]***

ans =

| 0 | 1 |
|---|---|
| 0.001 | 1.0010010013345 |
| 0.002 | 1.00200401068537 |
| 0.003 | 1.00300903609479 |
| ……. | …… |
| ……. | …… |
| 0.896 | 13.5275940388036 |
| 0.897 | 13.7139140587879 |
| 0.898 | 13.905417310063 |
| 0.899 | 14.1023233521297 |
| 0.9 | 14.3048643247043 |

*% vemos que efectivamente cambian las aproximaciones en t=0.9: para h=0.005, y(0.9) se aproxima por 14.3048592806911, mientras que para h=0.001, se aproxima por 14.3048643247043.*

*% Así un test de parada para evaluar la solución aproximada es el que acabamos de describir, pero comparando en cada componente del vector t (los t_i componentes que sean comunes en ambas discretizaciones): Si |y_h(t_i)-y_{h/2}(t_i)|<error de tolerancia (e.g., 0.05) tomar la solución aproximada obtenida con h. Test de este tipo deberían estar incluidos en "un buen programa" de resolución numérica.*

***>> format short g***

***>> [t,y]=rk4('ejer10',[0,0.9],1,0.05);[t,y]***

ans =

| | |
|---|---|
| 0 | 1 |
| 0.05 | 1.0527 |
| 0.1 | 1.1115 |
| 0.15 | 1.1777 |
| 0.2 | 1.253 |
| 0.25 | 1.3395 |
| 0.3 | 1.4397 |
| 0.35 | 1.557 |
| 0.4 | 1.6961 |
| 0.45 | 1.8632 |
| 0.5 | 2.067 |
| 0.55 | 2.3206 |
| 0.6 | 2.644 |
| 0.65 | 3.0694 |
| 0.7 | 3.6529 |
| 0.75 | 4.5015 |
| 0.8 | 5.8481 |
| 0.85 | 8.3128 |
| 0.9 | 14.271 |

***>> plot(t,y)***

*% Tomando h=0.05, la aproximación de la solución obtenida en 0.9 es 14.271.*
*Intentamos continuar la aproximación de la solución en [0.9,1], pero tomando ahora el dato inicial y(0.9)=14.271, es decir, aproximamos la solución del problema de Cauchy: y'=t^2+y^2, y(0.9)=14.271*

***>> [t,y]=rk4('ejer10',[0.9,1],14.271,0.01);[t,y]***

ans =

| | |
|---|---|
| 0.9 | 14.271 |
| 0.91 | 16.656 |
| 0.92 | 19.995 |
| 0.93 | 25.002 |
| 0.94 | 33.347 |
| 0.95 | 50.021 |
| 0.96 | 99.526 |
| 0.97 | 829.67 |
| 0.98 | 1.1969e+013 |
| 0.99 | 7.2137e+174 |
| 1 | Inf |

*% De nuevo vemos un crecimiento muy rápido de 0.96 a 0.97, y mucho mayor entre 0.96 y 1. Tomamos un h más pequeño (deberíamos utilizar el test de parada anterior o similar…)*

***>> [t,y]=rk4('ejer10',[0.9,1],14.271,0.005);[t,y]***

ans =

| | |
|---|---|
| 0.9 | 14.271 |
| 0.905 | 15.372 |
| 0.91 | 16.656 |
| 0.915 | 18.174 |
| 0.92 | 19.995 |
| 0.925 | 22.221 |
| 0.93 | 25.003 |
| 0.935 | 28.581 |
| 0.94 | 33.351 |
| 0.945 | 40.03 |
| 0.95 | 50.052 |
| 0.955 | 66.76 |
| 0.96 | 100.17 |
| 0.965 | 199.52 |
| 0.97 | 1678.5 |
| 0.975 | 2.8233e+013 |
| 0.98 | 2.0236e+176 |
| 0.985 | Inf |
| 0.99 | Inf |
| 0.995 | Inf |
| 1 | Inf |

*% vemos un crecimiento muy rápido de 0.965 a 0.97, y mucho mayor entre 0.96 y 1. Seguimos disminuyendo h*

***>> [t,y]=rk4('ejer10',[0.9,1],14.271,0.001);[t,y]***

ans =

| | |
|---|---|
| 0.9 | 14.271 |
| 0.901 | 14.478 |
| 0.902 | 14.692 |
| 0.903 | 14.912 |
| 0.904 | 15.138 |
| 0.905 | 15.372 |
| 0.906 | 15.613 |
| 0.907 | 15.861 |

```
    .....      .....
    .....      .....
    0.96     100.24
   0.961     111.41
   0.962     125.38
   0.963     143.35
   0.964     167.34
   0.965     200.96
   0.966      251.5
   0.967     335.99
   0.968     505.75
   0.969     1016.9
    0.97     9238.1
   0.971  5.6435e+014
   0.972  4.3074e+186
   0.973        Inf
   0.974        Inf
   0.975        Inf
    .....      .....
    .....      .....
   0.998        Inf
   0.999        Inf
       1        Inf
```

*% para estos dos valores de h, h=0.005 y h=0.001, el error de tolerancia en 0.96 es menor que 0.1 (podríamos hacerle más pequeño disminuyendo h de nuevo), mientras que los valores aproximados en 0.97 son muy distintos. No parece razonable fiarnos de ellos y nos hace pensar que la solución crece muy deprisa entre 0.96 y 0.97, y tiene una asíntota antes de t=0.97, entre 0.06 y 0.97.*

*% tomamos como buena la aproximación con rk4 y h=0.005 en [0.9, 0.96], y continuamos la gráfica (abajo, en amarillo)*

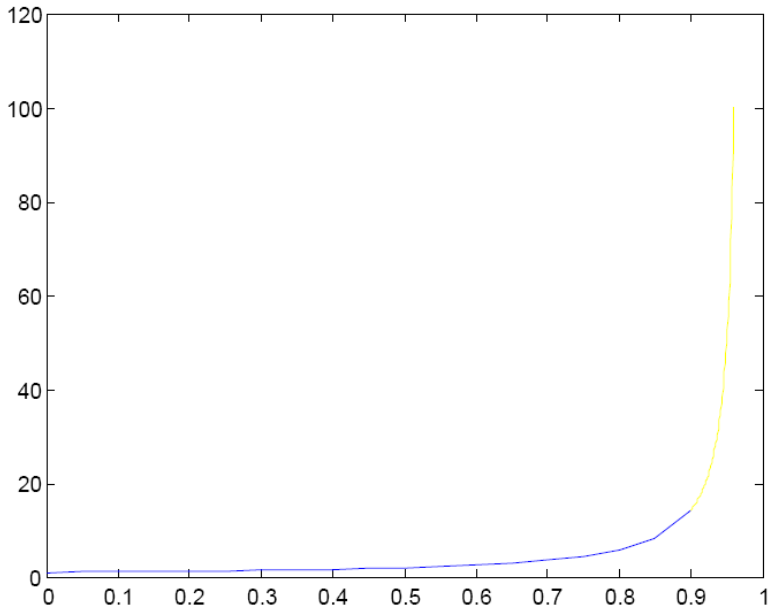
***>> [t,y]=rk4('ejer10',[0.9,0.96],14.271,0.005);[t,y]***

ans =

```
    0.9     14.271
  0.905     15.372
   0.91     16.656
  0.915     18.174
   0.92     19.995
  0.925     22.221
   0.93     25.003
  0.935     28.581
```

| | |
|---|---|
| 0.94 | 33.351 |
| 0.945 | 40.03 |
| 0.95 | 50.052 |
| 0.955 | 66.76 |
| 0.96 | 100.17 |

***>> plot(t,y,'y')***




*% La gráfica obtenida muestra la aproximación de la solución del problema propuesto, con distintos tamaños de paso en [0,0.9] y en [0.9,0.96], lo que, entre otras cosas ahorra cálculos.*

*% A continuación comparamos nuestra resolución/aproximación con la que nos proporciona "ode45": esta función Matlab no necesita leer el tamaño del paso h y detecta la asíntota.*

*% Comenzamos con el problema, intentando aproximar en [0,1]; el resultado es:*

***>> [t,y]=ode45('ejer10',[0,1],1);[t,y]***

Warning: Failure at t=9.698087e-001. Unable to meet integration tolerances without reducing the step size below the smallest value allowed (1.776357e-015) at time t.
> In ode45 at 371

ans =

| | |
|---|---|
| 0 | 1 |
| 0.025 | 1.0256 |
| 0.05 | 1.0527 |
| 0.075 | 1.0812 |
| 0.1 | 1.1115 |
| 0.125 | 1.1436 |
| 0.15 | 1.1777 |
| 0.175 | 1.2141 |
| 0.2 | 1.253 |
| 0.225 | 1.2947 |
| 0.25 | 1.3395 |
| 0.275 | 1.3877 |
| ……. | …… |
| ……. | …… |
| 0.4 | 1.6961 |
| 0.425 | 1.7757 |
| 0.45 | 1.8632 |
| 0.475 | 1.9598 |
| ……. | …… |
| ……. | …… |
| 0.6 | 2.644 |
| 0.625 | 2.8413 |
| 0.65 | 3.0694 |
| 0.675 | 3.3363 |
| ……. | …… |
| ……. | …… |
| 0.8 | 5.8486 |
| 0.81841 | 6.5725 |
| 0.83682 | 7.4878 |
| ……. | …… |
| ……. | …… |
| 0.90048 | 14.403 |
| 0.90943 | 16.544 |
| 0.91596 | 18.568 |
| 0.9225 | 21.127 |
| 0.92903 | 24.5 |
| 0.93556 | 29.192 |
| 0.93876 | 32.206 |
| 0.94195 | 35.897 |
| 0.94515 | 40.54 |
| 0.94835 | 46.583 |
| 0.95066 | 52.263 |

```
0.95298      59.442
 0.9553      68.902
0.95762      82.047
0.95876       90.54
 0.9599      100.94
0.96104      114.04
0.96218       131.1
  0.963      147.06
0.96383      167.22
0.96465      193.78
0.96547      230.65
0.96588      254.58
0.96629      283.91
0.96669      320.85
 0.9671      369.02
0.96739      413.87
0.96768      470.53
0.96797      545.13
0.96827      648.65
0.96841      716.08
0.96856      798.74
 0.9687      902.93
0.96885      1038.9
0.96895        1165
0.96905      1324.3
0.96916      1533.9
0.96926      1824.7
0.96931      2014.7
0.96936      2247.7
0.96942      2541.6
0.96947      2925.2
 0.9695       3280
0.96954        3728
0.96958      4317.3
0.96961      5134.5
0.96963        5670
.............      ..........
.............      ..........
0.96979       44933
0.96979       50152
.............      ..........
.............      ..........
 0.9698       96363
 0.9698  1.1454e+005
 0.9698  1.2652e+005
```

…… ……
…… ……
0.96981 9.0834e+005
0.96981 1.0035e+006
0.96981 1.4613e+006
…… ……
…… ……
0.96981 8.8918e+006
0.96981 1.0065e+007
…… ……
…… ……
0.96981 1.4769e+007
…… ……
…… ……
0.96981 1.7809e+008
0.96981 1.9889e+008
0.96981 2.2519e+008
0.96981 2.5963e+008
0.96981 2.9095e+008
…… ……
…… ……
0.96981 1.1404e+014
0.96981 1.2517e+014
…… ……
…… ……
0.96981 2.7311e+014
0.96981 3.2145e+014

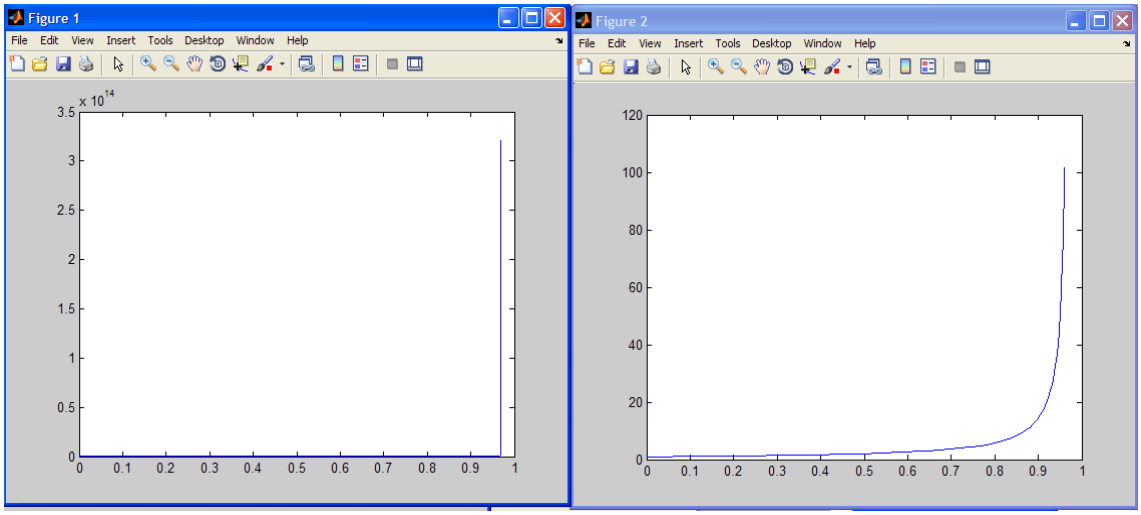
*% Vemos valores muy grandes de la aproximación a medida que nos acercamos a 0.97; evitamos los vectores solución para leer mejor la advertencia de Matlab*

***>> [t,y]=ode45('ejer10',[0,1],1);***

Warning: Failure at t=9.698087e-001. Unable to meet integration tolerances without reducing the step size below the smallest
value allowed (1.776357e-015) at time t.
> In ode45 at 371

***>> plot(t,y)***

*% y de nuevo, el dibujo de la solución calculada no nos da idea del comportamiento de la solución; hay que hacer la representación en [0, 0.96] para tener esta idea, tal y como muestran las gráficas de abajo*




*% ver también crecimiento de la solución entre 0.96 y 0.97 y la gráfica en [0,0.967] donde se ve mejor la existencia de la asíntota cercana a t=0.97.*

***>> format short g***

***>> [t,y]=ode45('ejer10',[0,0.97],1);[t,y]***

Warning: Failure at t=9.697958e-001. Unable to meet integration tolerances without reducing the step size below the smallest value allowed (1.776357e-015) at time t.
> In ode45 at 371

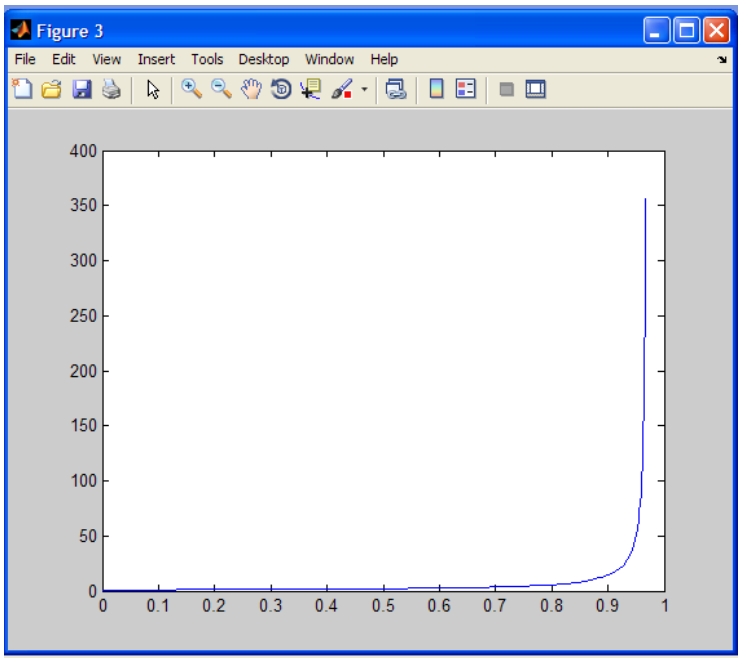
ans =

| | |
|---|---|
| 0 | 1 |
| 0.02425 | 1.0249 |
| 0.0485 | 1.051 |
| ..... | ..... |
| ..... | ..... |
| 0.9551 | 67.945 |
| 0.95819 | 86.134 |
| 0.95907 | 93.213 |
| 0.95995 | 101.55 |
| 0.96083 | 111.51 |
| 0.96171 | 123.66 |
| 0.96259 | 138.88 |
| 0.96347 | 158.14 |
| 0.96435 | 183.6 |
| 0.96523 | 219.13 |
| 0.96563 | 240.22 |
| 0.96603 | 265.73 |

| | |
|---|---|
| 0.96643 | 297.28 |
| 0.96683 | 337.45 |
| 0.96723 | 391.37 |
| 0.96763 | 462.7 |
| 0.96803 | 565.95 |
| 0.96843 | 733.68 |
| 0.96853 | 790.6 |
| 0.96863 | 857.02 |
| 0.96873 | 935.61 |
| 0.96883 | 1030.2 |
| 0.96892 | 1146.4 |
| 0.96902 | 1291.2 |
| 0.96912 | 1477.8 |
| 0.96922 | 1728.7 |
| 0.96927 | 1903.7 |
| 0.96932 | 2117.3 |
| 0.96938 | 2384.6 |
| 0.96943 | 2730.4 |
| 0.96947 | 3066.5 |
| 0.96951 | 3492.2 |
| 0.96955 | 4054.9 |
| 0.96959 | 4840.4 |
| 0.96961 | 5333.8 |
| 0.96963 | 5936.4 |
| 0.96965 | 6692.1 |
| 0.96967 | 7671.7 |
| 0.96968 | 8613.1 |
| 0.96969 | 9804.6 |
| 0.96971 | 11378 |
| 0.96972 | 13570 |
| 0.96973 | 14961 |
| 0.96974 | 16661 |
| 0.96974 | 18795 |
| 0.96975 | 21567 |
| 0.96975 | 24206 |
| 0.96976 | 27545 |
| 0.96976 | 31950 |
| 0.96977 | 38082 |
| 0.96977 | 41999 |
| 0.96977 | 46793 |
| 0.96978 | 52819 |
| 0.96978 | 60655 |
| 0.96978 | 68062 |
| 0.96978 | 77427 |
| 0.96978 | 89773 |

0.96979 1.0694e+005
…… ……
…… ……
0.96979 9.3242e+005
0.96979 1.0398e+006
0.96979 1.1752e+006
…… ……
…… ……
0.9698 9.3093e+006
0.9698 1.0715e+007
…… ……
…… ……
0.9698 9.5275e+007
0.9698 1.0826e+008
0.9698 1.2534e+008
…… ……
…… ……
0.9698 8.5882e+008
0.9698 9.9409e+008
0.9698 1.1814e+009
…… ……
…… ……
0.9698 9.3715e+009
0.9698 1.0354e+010
…… ……
…… ……
0.9698 9.1768e+010
0.9698 1.0389e+011
…… ……
…… ……
0.9698 9.5107e+011
0.9698 1.0658e+012
…… ……
…… ……
0.9698 9.612e+012
0.9698 1.1126e+013
…… ……
…… ……
0.9698 8.8235e+013
0.9698 1.0489e+014
…… ……
…… ……
0.9698 4.8691e+014
0.9698 6.2266e+014

***>> [t,y]=ode45('ejer10',[0,0.967],1);plot(t,y)***




*% En los vectores (t,y), que nos devuelve ode45, vemos que el tamaño del paso es variable: se va haciendo mucho más pequeño entre 0.96 y 0.97; esto es debido a otro test de control de paso de ode45 que detecta el crecimiento rápido de la solución. De manera general, un test de parada, que se puede usar e incorporar en un buen programa para detectar variaciones rápidas de soluciones, es: si la distancia entre dos puntos consecutivos, en una iteración de paso h, es más grande que una tolerancia que establezcamos, entonces disminuir el h.*

*% Abajo, vemos que ode45 no aproxima en todo el intervalo, [0,0.97], y sí lo hace en [0,0.968], para los errores de tolerancia que tiene incorporados por defecto.*

***>> [t,y]=ode45('ejer10',[0,1],1);*** *% no aproxima en todo el intervalo*

Warning: Failure at t=9.698087e-001. Unable to meet integration tolerances without reducing the step size below the smallest value allowed
(1.776357e-015) at time t.
> In ode45 at 309

***>> [t,y]=ode45('ejer10',[0,0.97],1);*** *% no aproxima en todo el intervalo*

Warning: Failure at t=9.697958e-001. Unable to meet integration tolerances without reducing the step size below the smallest value allowed
(1.776357e-015) at time t.
> In ode45 at 309

***>> [t,y]=ode45('ejer10',[0,0.968],1);*** *% no da ninguna alerta*

*% **Comparación gráfica de nuestra resolución con rk4, y con ode45 en [0,0.96]:** abajo vemos la similitud de las gráficas obtenidas con rk4 para distintos tamaños de paso en [0,0.9] y en [0.9, 0.96], y la obtenida con ode45*

***>> [t,y]=rk4('ejer10',[0,0.9],1,0.05);***

***>> plot(t,y)***

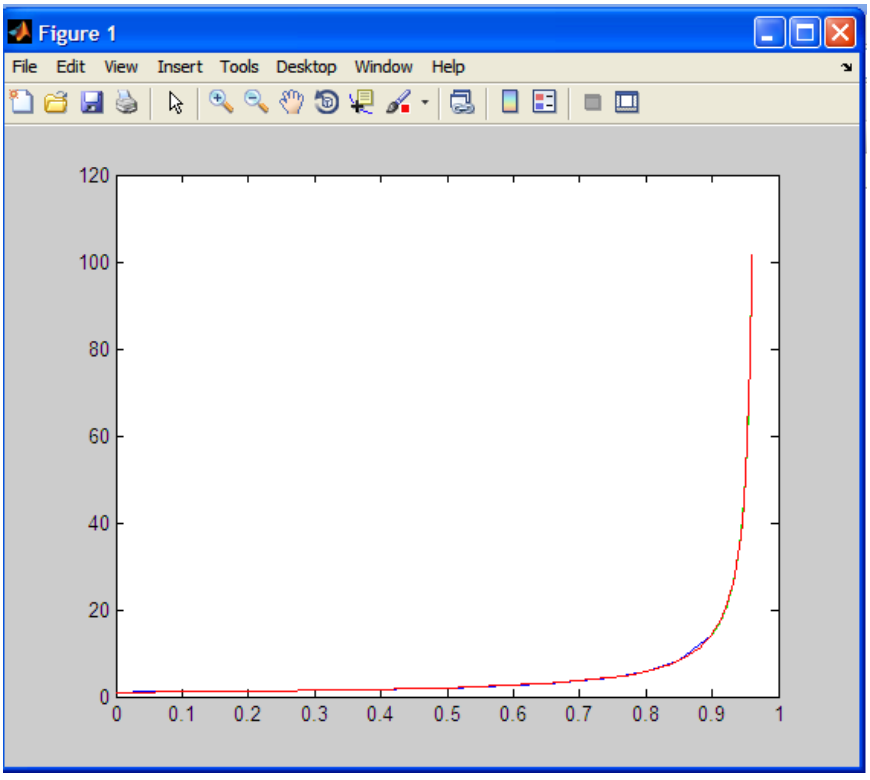
***>> hold on***

***>> [t,y]=rk4('ejer10',[0.9,0.96],14.271,0.005);***

***>> plot(t,y,'y')***

***>> [t,y]=ode45('ejer10',[0,0.96],1);***

***>> plot(t,y,'r')***



***% Observaciones sobre el tamaño del paso y su control con ode45.***

*% La función ode45, está programada para darnos la solución con una estimación del error dependiente de dos números RelTol y AbsTol, que por defecto toman los valores 10^{-3} y 10^{-6} respectivamente: con RelTol se controla el error cometido dependiendo del valor de la aproximación, evitando en cada iteración que el tamaño del paso se haga muy pequeño si la aproximación es muy grande. Estos valores de tolerancia relativos y absolutos, RelTol y AbsTol, se pueden disminuir para obtener una mejor aproximación de la solución. Damos una idea gráfica de este control de paso.*

*% Podemos hacer una gráfica que nos muestre como varía el tamaño del paso. Para esto, en un fichero tamagnopaso1213.m hemos introducido instrucciones para comparar gráficas y tamaños del paso con rk4 (donde el h es constante) y con ode45 (donde se va ajustando h)*

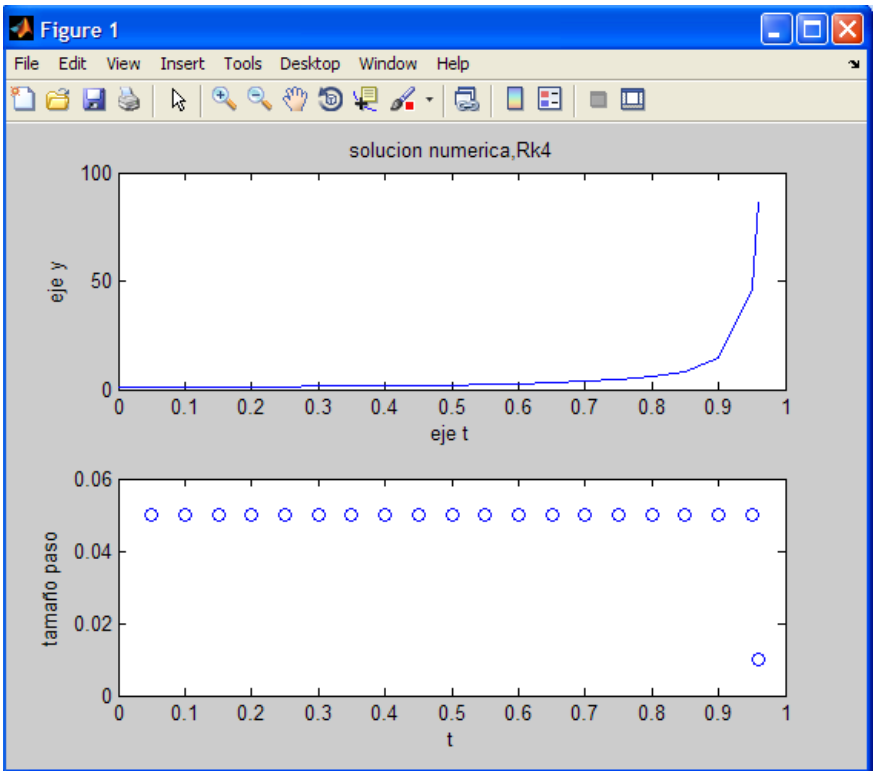
***>> type tamagnopaso1213***

***>> h=0.05*** *% tomamos este paso en [0,0.96] para rk4,*

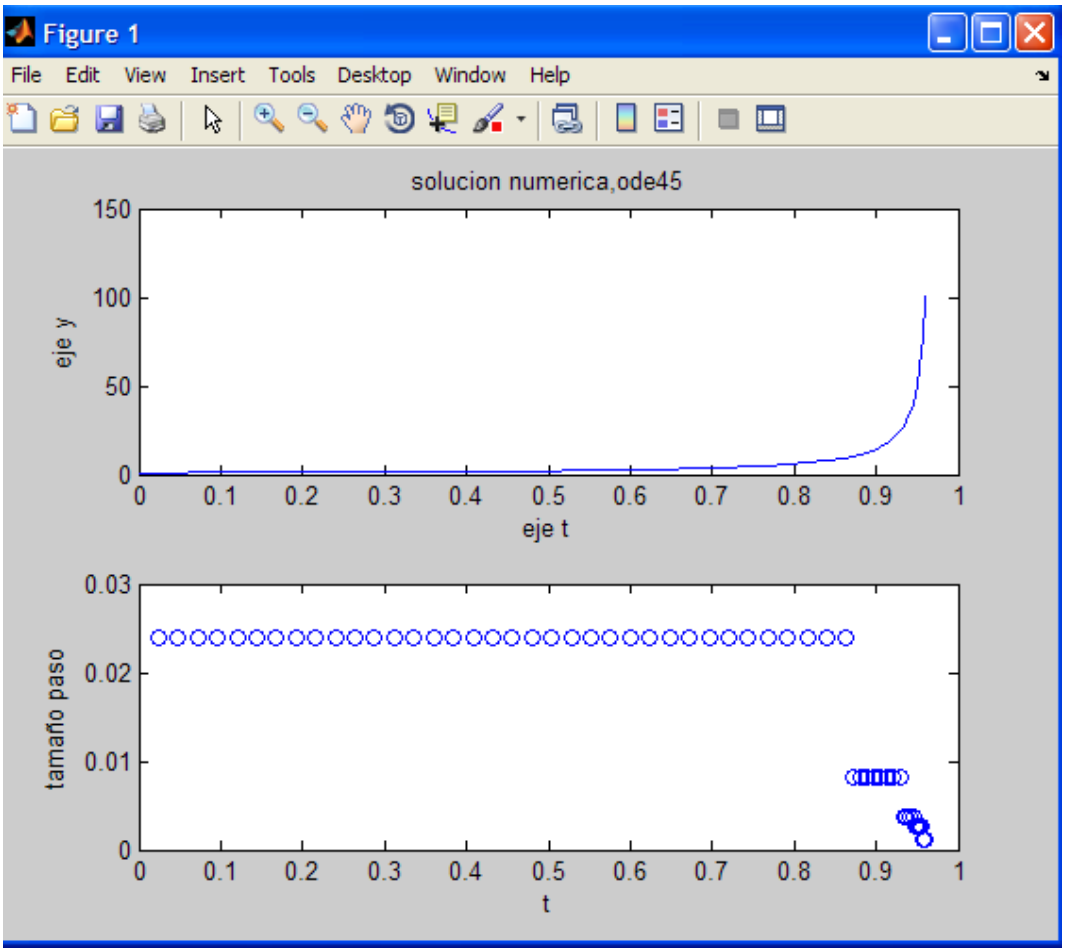
h =

0.0500

***>> tamagnopaso1213***



*% Dibujo que nos proporciona rk4 para h=0.05. Se observa una gráfica angulosa de la aproximación entre 0.9 y 0.96 mientras que el tamaño de paso sólo cambia en la última iteración para ajustar el intervalo (i.e., dependiendo de 0.96/h)*

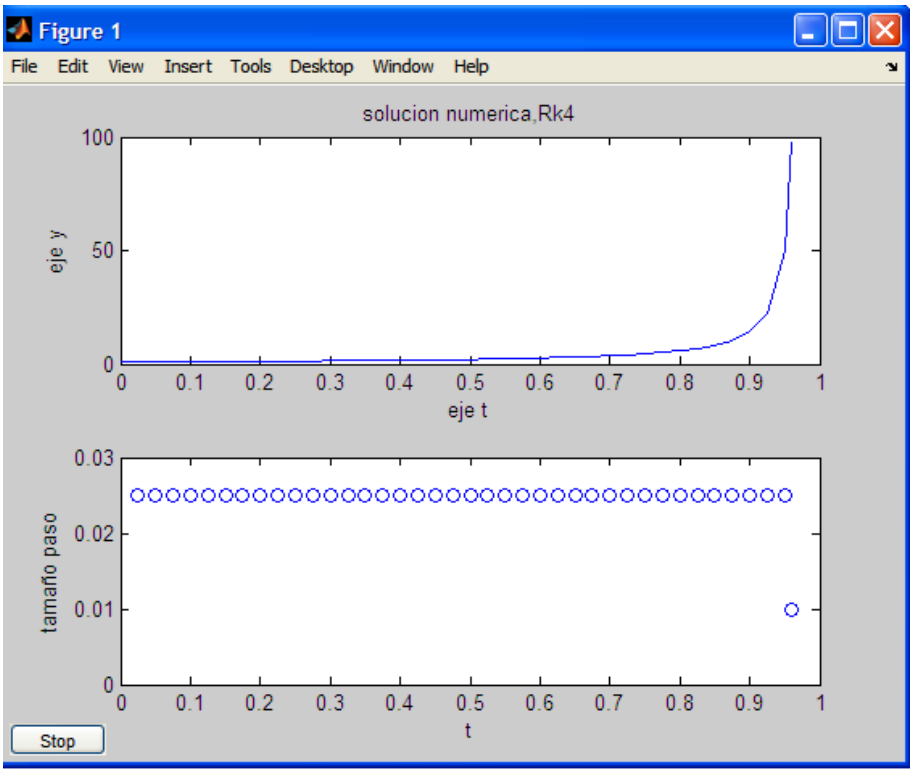


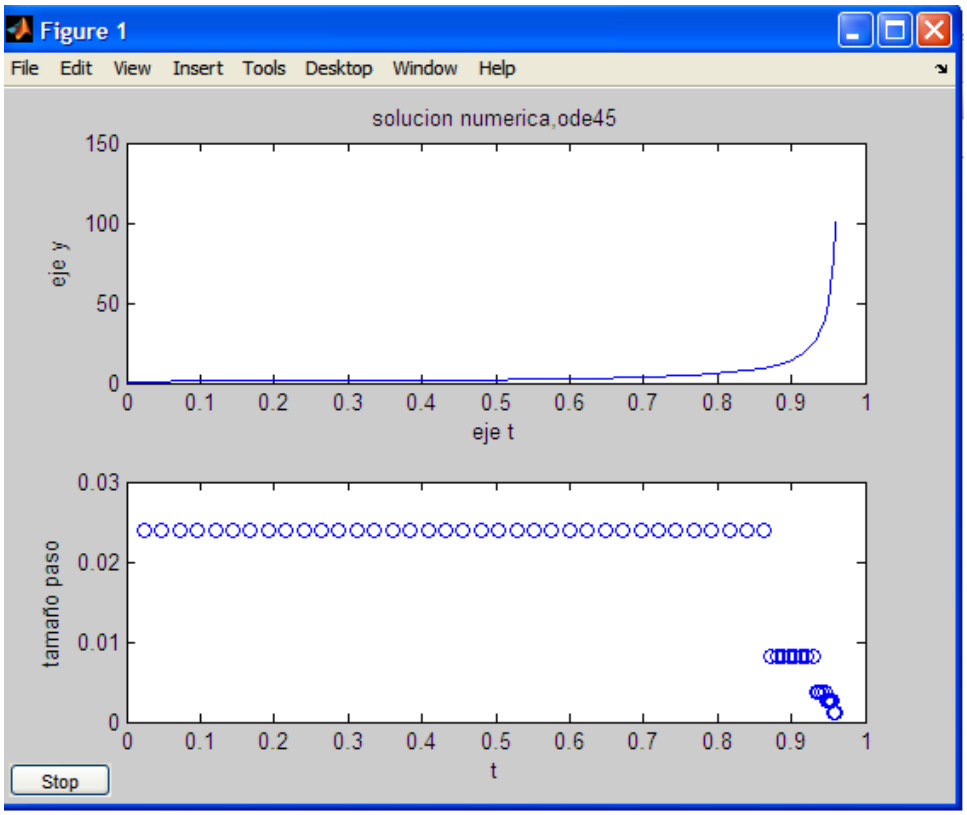

*% Dibujo que nos proporciona ode45 con paso variable. Vemos que le toma constante (entre 0.02 y 0.03) en [0, 0.8], y luego va disminuyéndole, ajustándole para tener un error de tolerancia adecuado para t entre 0.9 y 0.96*

*% Si disminuimos el h para rk4 (ode45 no se ve afectado), e.g., h=0.025 mejora la aproximación con rk4 pero sigue siendo peor cerca de 0.96 (tal y como se ve abajo)*

***>> h=0.025***

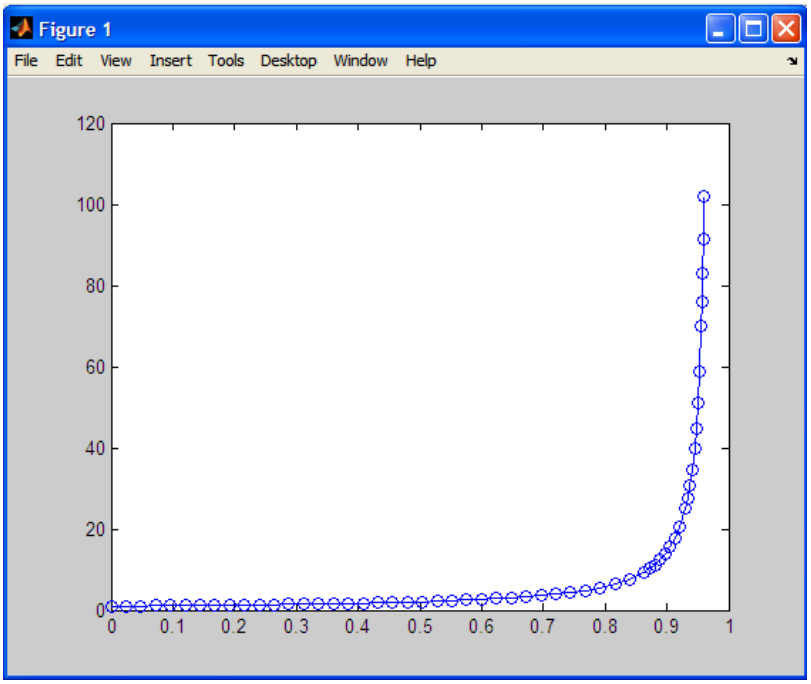
***>> tamagnopaso1213***



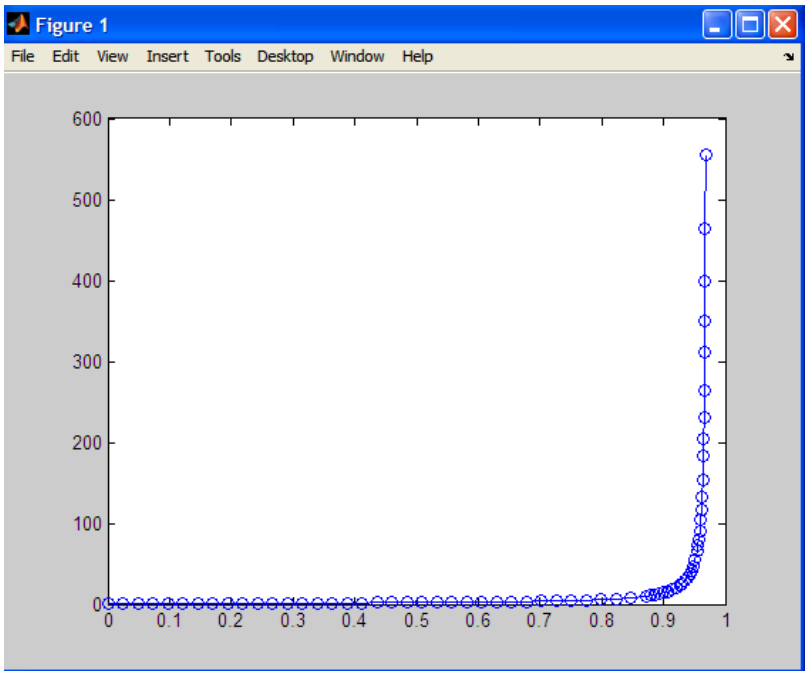



*% Esta característica de paso variable se observa con la función ode45, dependiendo de la versión de Matlab: abajo, en distintos intervalos, ode45 no nos da números pero sí idea del tamaño del paso*

***>> ode45('ejer10',[0,0.96],1);***




***>> ode45('ejer10',[0,0.968],1);***



*% Como se ha comentado antes, en la función Matlab ode45, se pueden cambiar errores de tolerancia en aras a obtener una mejor aproximación de la solución. Explicamos esto analizando los vectores tamaño del paso y las opciones de ode45.*

***>> help ode45***

```
ode45  Solve non-stiff differential equations, medium order method.
   [TOUT,YOUT] = ode45(ODEFUN,TSPAN,Y0) with TSPAN = [T0 TFINAL] integrates
   the system of differential equations y' = f(t,y) from time T0 to TFINAL
   with initial conditions Y0. ODEFUN is a function handle. For a scalar T
   and a vector Y, ODEFUN(T,Y) must return a column vector corresponding
   to f(t,y). Each row in the solution array YOUT corresponds to a time
   returned in the column vector TOUT.  To obtain solutions at specific
   times T0,T1,...,TFINAL (all increasing or all decreasing), use TSPAN =
   [T0 T1 ... TFINAL].

   [TOUT,YOUT] = ode45(ODEFUN,TSPAN,Y0,OPTIONS) solves as above with default
   integration properties replaced by values in OPTIONS, an argument created
   with the ODESET function. See ODESET for details. Commonly used options
   are scalar relative error tolerance 'RelTol' (1e-3 by default) and vector
   of absolute error tolerances 'AbsTol' (all components 1e-6 by default).
   If certain components of the solution must be non-negative, use
   ODESET to set the 'NonNegative' property to the indices of these
   components.

   ode45 can solve problems M(t,y)*y' = f(t,y) with mass matrix M that is
   nonsingular. Use ODESET to set the 'Mass' property to a function handle
   MASS if MASS(T,Y) returns the value of the mass matrix. If the mass matrix
   is constant, the matrix can be used as the value of the 'Mass' option. If
   the mass matrix does not depend on the state variable Y and the function
   MASS is to be called with one input argument T, set 'MStateDependence' to
   'none'. ODE15S and ODE23T can solve problems with singular mass matrices.

   [TOUT,YOUT,TE,YE,IE] = ode45(ODEFUN,TSPAN,Y0,OPTIONS) with the 'Events'
   property in OPTIONS set to a function handle EVENTS, solves as above
   while also finding where functions of (T,Y), called event functions,
   are zero. For each function you specify whether the integration is
   to terminate at a zero and whether the direction of the zero crossing
   matters. These are the three column vectors returned by EVENTS:
   [VALUE,ISTERMINAL,DIRECTION] = EVENTS(T,Y). For the I-th event function:
   VALUE(I) is the value of the function, ISTERMINAL(I)=1 if the integration
   is to terminate at a zero of this event function and 0 otherwise.
   DIRECTION(I)=0 if all zeros are to be computed (the default), +1 if only
   zeros where the event function is increasing, and -1 if only zeros where
   the event function is decreasing. Output TE is a column vector of times
   at which events occur. Rows of YE are the corresponding solutions, and
   indices in vector IE specify which event occurred.

   SOL = ode45(ODEFUN,[T0 TFINAL],Y0...) returns a structure that can be
   used with DEVAL to evaluate the solution or its first derivative at
   any point between T0 and TFINAL. The steps chosen by ode45 are returned
   in a row vector SOL.x.  For each I, the column SOL.y(:,I) contains
   the solution at SOL.x(I). If events were detected, SOL.xe is a row vector
   of points at which events occurred. Columns of SOL.ye are the corresponding
   solutions, and indices in vector SOL.ie specify which event occurred.

   Example
        [t,y]=ode45(@vdp1,[0 20],[2 0]);
        plot(t,y(:,1));
     solves the system y' = vdp1(t,y), using the default relative error
     tolerance 1e-3 and the default absolute tolerance of 1e-6 for each
     component, and plots the first component of the solution.

   Class support for inputs TSPAN, Y0, and the result of ODEFUN(T,Y):
     float: double, single
```

```
See also ode23, ode113, ode15s, ode23s, ode23t, ode23tb, ode15i,
         odeset, odeplot, odephas2, odephas3, odeprint, deval,
         odeexamples, rigidode, ballode, orbitode, function_handle.
```

*% Las distintas opciones se cambian con el comando "odeset" (vemos que hay otras aparte de RelTol y AbsTol)*

***>> help odeset***

```
ODESET Create/alter ODE OPTIONS structure.
   OPTIONS = ODESET('NAME1',VALUE1,'NAME2',VALUE2,...) creates an integrator
   options structure OPTIONS in which the named properties have the
   specified values. Any unspecified properties have default values. It is
   sufficient to type only the leading characters that uniquely identify the
   property. Case is ignored for property names.

   OPTIONS = ODESET(OLDOPTS,'NAME1',VALUE1,...) alters an existing options
   structure OLDOPTS.
ODESET Create/alter ODE OPTIONS structure.
   OPTIONS = ODESET('NAME1',VALUE1,'NAME2',VALUE2,...) creates an integrator
   options structure OPTIONS in which the named properties have the
   specified values. Any unspecified properties have default values. It is
   sufficient to type only the leading characters that uniquely identify the
   property. Case is ignored for property names.

   OPTIONS = ODESET(OLDOPTS,'NAME1',VALUE1,...) alters an existing options
   structure OLDOPTS.

   OPTIONS = ODESET(OLDOPTS,NEWOPTS) combines an existing options structure
   OLDOPTS with a new options structure NEWOPTS. Any new properties
   overwrite corresponding old properties.

   ODESET with no input arguments displays all property names and their
   possible values.

 ODESET PROPERTIES

 RelTol - Relative error tolerance  [ positive scalar {1e-3} ]
   This scalar applies to all components of the solution vector, and
   defaults to 1e-3 (0.1% accuracy) in all solvers.  The estimated error in
   each integration step satisfies e(i) <= max(RelTol*abs(y(i)),AbsTol(i)).

 AbsTol - Absolute error tolerance  [ positive scalar or vector {1e-6} ]
   A scalar tolerance applies to all components of the solution vector.
   Elements of a vector of tolerances apply to corresponding components of
   the solution vector. AbsTol defaults to 1e-6 in all solvers. See RelTol.

 NormControl -  Control error relative to norm of solution  [ on | {off} ]
   Set this property 'on' to request that the solvers control the error in
   each integration step with norm(e) <= max(RelTol*norm(y),AbsTol). By
   default the solvers use a more stringent component-wise error control.

 Refine - Output refinement factor  [ positive integer ]
   This property increases the number of output points by the specified
   factor producing smoother output. Refine defaults to 1 in all solvers
   except ODE45, where it is 4. Refine does not apply if length(TSPAN) > 2
   or the ODE solver returns the solution as a structure.
  ..............................
..............................
```

*% Resultado sin modificar RelTol y AbsTol, el conjunto de comandos*

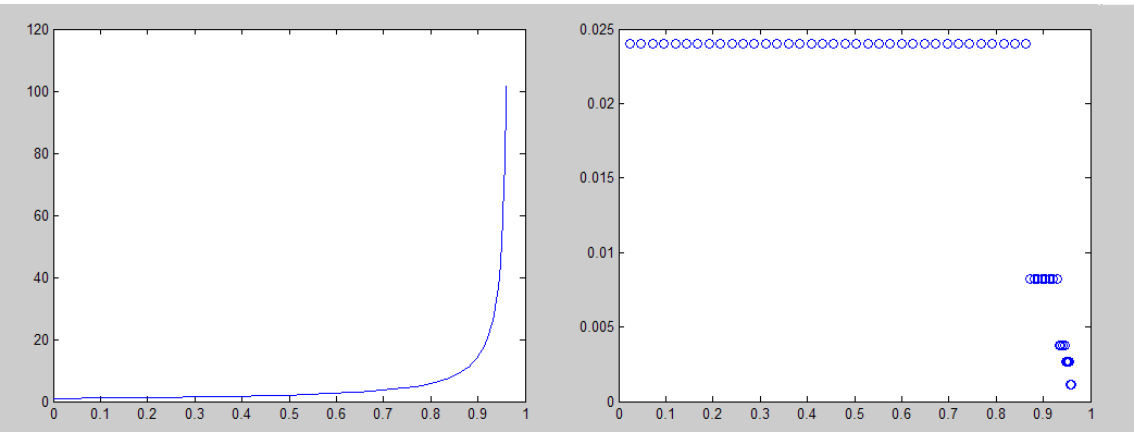
***>> [t,y]=ode45('ejer10',[0,0.96],1);***

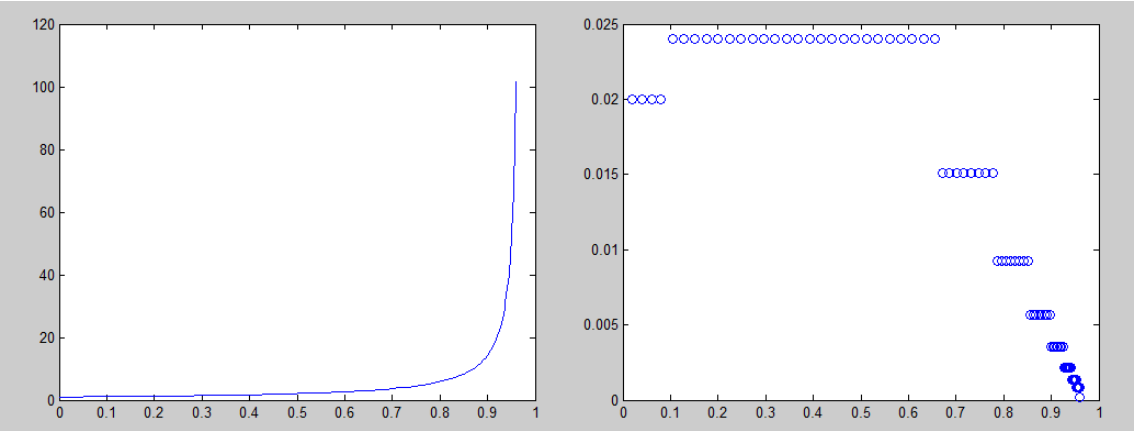
***>> dt=diff(t); tdt=t(2:length(t));***

***>> plot(tdt,dt,'o')***

***>> plot(t,y)***

*% nos proporciona la gráfica de la aproximación y del tamaño del paso (con círculos): diff(t) nos da el vector incremento para t con n+1 componentes.*




*% Las segundas gráficas se han obtenido modificando "odeset", esto es, con el conjunto de comandos*

***>> options=odeset('RelTol',1e-5,'AbsTol',1e-8)***

***>> [t,y]=ode45('ejer10',[0,0.96],1,options);***

***>> dt=diff(t); tdt=t(2:length(t));***

***>> plot(tdt,dt,'o')***

***>> plot(t,y)***

*% Aparentemente las dos gráficas de la solución son la misma pero ha cambiado el tamaño del paso y, por tanto, los puntos de aproximación.*

***>> format long***

***>> [t,y]=ode45('ejer10',[0,0.9],1);[t,y]***

ans =

| | |
|---|---|
| 0 | 1.000000000000000 |
| 0.022500000000000 | 1.023021860516800 |
| 0.045000000000000 | 1.047151579359794 |
| ..... | ..... |
| ..... | ..... |
| 0.890109422701895 | 12.525112091014282 |
| 0.895054711350947 | 13.356187059600886 |
| 0.900000000000000 | 14.305101082768962 |

***>> dt1=diff(t)*** *% tamaños de paso que toma ode45 sin modificar RelTol y AbsTol*

dt1 =

0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000

```
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.022500000000000
0.017554711350947
0.017554711350947
0.017554711350947
0.017554711350947
0.004945288649053
0.004945288649053
0.004945288649053
0.004945288649053
```

*% introducimos errores de tolerancia más pequeños*

***>> options=odeset('RelTol',1e-5,'AbsTol',1e-8)***

```
options =

    AbsTol: 1e-008
       BDF: []
    Events: []
     .....
     .....
    Refine: []
    RelTol: 1e-005
     Stats: []
Vectorized: []
MStateDependence: []
   MvPattern: []
  InitialSlope: []
```

*% como vemos, en el listado hay otras propiedades que se pueden cambiar*

***>> [t,y]=ode45('ejer10',[0,0.9],1,options);[t,y]***

```
ans =
                0   1.000000000000000
  0.020000000000000   1.020410920374718
  0.040000000000000   1.041688482521341
  ..................  ..................
  ..................  ..................
  0.894205366842064  13.205370737436846
  0.897102683421032  13.733112386004239
  0.900000000000000  14.304627837082519
```

***>> dt2=diff(t)*** *% tamaños de paso que toma ode45 con las opciones introducidas*

```
dt2 =

  0.020000000000000
  0.020000000000000
  0.020000000000000
  0.020000000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
  0.022500000000000
```

```
  0.022500000000000
  0.016885439380394
  0.016885439380395
  0.016885439380395
  0.016885439380394
  0.016885439380394
  0.016885439380395
  0.016885439380395
  0.016885439380394
  0.010320277151277
  0.010320277151277
  0.010320277151277
  0.010320277151277
  0.010320277151277
  0.010320277151277
  0.010320277151277
  0.010320277151277
  0.006345625178845
  0.006345625178845
  0.006345625178845
  0.006345625178845
  0.006345625178845
  0.006345625178845
  0.006345625178845
  0.006345625178845
  0.002897316578968
  0.002897316578968
  0.002897316578968
  0.002897316578968
```

*% El tamaño del paso es menor con el cambio introducido en odeset: comparamos el tamaño de los vectores diff(t) con las distintas opciones usadas*

***>> length(dt1)***

```
ans =

   44
```

***>> length(dt2)***

```
ans =

   56
```

## *Ejercicios 3 y 4*

*% Se trata de ejemplos de problemas de valores iniciales en los no que tiene garantizada la existencia y unicidad de solución pasando por un punto. En particular, la unicidad de solución no se puede garantizar debido a que la función f(t,y) no es regular (su derivada parcial con respecto a y no es continua en el valor inicial). Analizamos algunos resultados que nos dan los métodos numéricos (en general, no se tiene garantizada la convergencia de un método numérico y el resultado puede ser imprevisible). Son los ejercicios 15 y 16 de la sección 1.7 del libro de apuntes*

15. La ecuación $y' = 2|y|^{\frac{1}{2}}$ admite al menos las soluciones $y = 0$ e $y = x|x|$ soluciones pasando por $(0,0)$ definidas en $(-\infty, \infty)$. Comprobar que con el método de Euler sólo encontramos una aproximación de la primera para cualquier paso $h$.

16. ¿Qué se puede decir sobre la unicidad de solución del problema

$$\begin{cases} y' &= y|y|^{-3/4} + x \sin \frac{\pi}{x} \\ y(0) &= 0 \end{cases} ?.$$

Aplicar el método de Euler para encontrar la aproximación de la solución en $[0, 0.2]$: Encontrar dos valores del tamaño del paso $h$, tal que la solución numérica en $x = 0.2$ sea positiva y dos valores de $h$ tales que sea negativa. ¿A qué se debe este resultado?. Aplicar el método de Runge-Kutta para aproximar numéricamente la solución en $[0, 1]$ para los valores de los pasos $h = \frac{1}{15}, \frac{1}{20}, \frac{1}{10}, \frac{1}{27}$, respectivamente. Hacer una gráfica de la solución. ¿Qué se observa?.

*% Para el primero, demostramos que cualquier método numérico sólo nos da la solución y=0, y sabemos (por un cálculo elemental) que hay muchas soluciones. Para el segundo, dependiendo del método y del tamaño del paso, obtenemos, aleatoriamente, aproximación hacia una u otra curva "posibles soluciones": al no tener garantizado que exista solución única, los métodos no tienen por qué converger a nada. Ambos ejemplos muestran que tenemos que verificar que estamos en las condiciones del teorema de existencia y unicidad de solución, para poder aproximarla.*

*% Para el ejercicio 3, modificamos la función ejer10.m cambiando f(t,y)= t^2+y^2 por f(t,y)= 2*sqrt(|y|) y guardamos en el fichero ejer3.m. Intentamos aproximar la solución que pasa por (0,0) con distintos métodos (rk2, rk4 y ode45) y distintos tamaños de paso, pero sólo se obtiene como solución la que pasa por el (0,0), la y=0. Comprobamos que y= x^2 es también una solución cuando x>0, que no obtenemos numéricamente.*

***>> type ejer3***

***>> help abs***

*ABS Absolute value.*
*ABS(X) is the absolute value of the elements of X. When*
*X is complex, ABS(X) is the complex modulus (magnitude) of*
*the elements of X.*

***>> [t,y]=rk4('ejer3',[0,1],0,0.05);[t,y]***

ans =

| | |
|---|---|
| 0 | 0 |
| 0.05 | 0 |
| 0.1 | 0 |
| 0.15 | 0 |
| 0.2 | 0 |
| 0.25 | 0 |
| 0.3 | 0 |
| 0.35 | 0 |
| 0.4 | 0 |
| 0.45 | 0 |
| 0.5 | 0 |
| 0.55 | 0 |
| 0.6 | 0 |
| 0.65 | 0 |
| 0.7 | 0 |
| 0.75 | 0 |
| 0.8 | 0 |
| 0.85 | 0 |
| 0.9 | 0 |
| 0.95 | 0 |
| 1 | 0 |

***>> plot(t,y)*** *% gráfica de la solución y=0, en línea continua*

***>> [t,y]=rk4('ejer3',[0,1],0,0.01);[t,y]***

ans =

| | |
|---|---|
| 0 | 0 |
| 0.01 | 0 |
| 0.02 | 0 |
| ..... | ... |
| ..... | ... |

```
    0.97         0
    0.98         0
    0.99         0
       1         0
```

***>> [t,y]=rk2('ejer3',[0,1],0,0.01);[t,y]***

```
ans =

       0         0
    0.01         0
    0.02         0
    .....        ...
    .....        ...
    0.97         0
    0.98         0
    0.99         0
       1         0
```

***>> [t,y]=ode45('ejer3',[0,1],0);[t,y]***

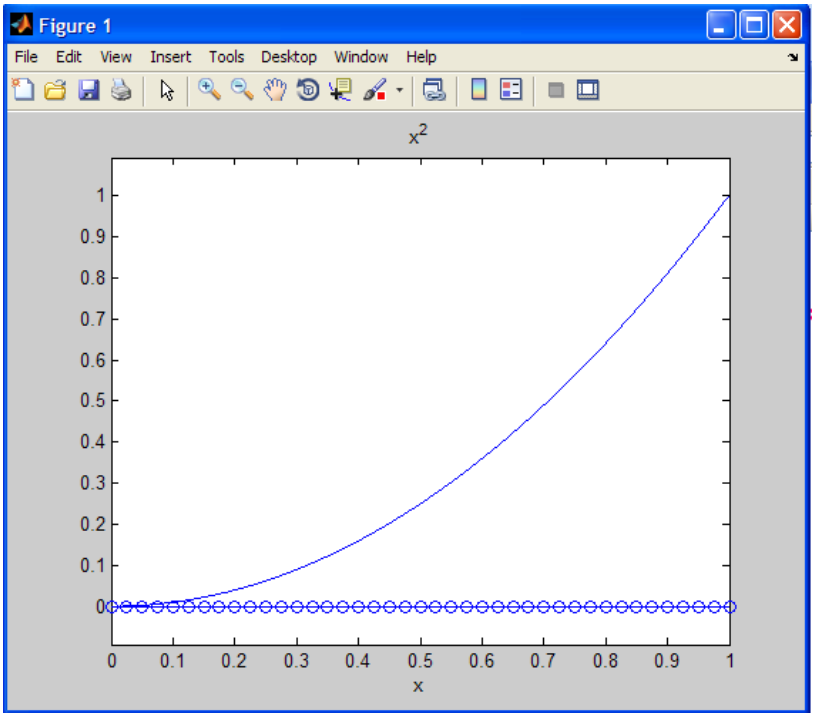
```
ans =

       0         0
   0.025         0
    .....        ...
    .....        ...

   0.975         0
       1         0
```

***>> hold on***

***>> plot(t,y,'o')*** *% gráfica de la solución numérica en línea continua*

***>> syms x***

***>> ezplot('x^2',[0,1])*** *% gráfica de la solución y=x^2, para x>0*




*% **Observación:** numéricamente no se ha detectado más que una solución: la gráfica con círculos*

*% Para el ejercicio 4, modificamos el fichero ejer10.m para introducir la nueva función f(t,y) usando las estructuras de control "if-then-else".*

```
function z=ejer16(x,y)
if x==0,
  if y==0,
    z=0;
  else
    z=y*(abs(y))^(-3/4);
   end
 else
   if y==0
     z=x*sin(pi/x);
   else
z=y*(abs(y))^(3/4)+x*sin(pi/x);
   end
end
```

***>> type ejer16***

*% Utilizamos distintos métodos numéricos, con distintos tamaños de paso y hacemos la gráfica de la solución que vamos obteniendo*

***>> format short g***

***>> [t,y]=rk4('ejer16',[0,0.4],0,1/10);[t,y]***

```
ans =
         0         0
       0.1  -0.00078877
       0.2   -0.021921
       0.3   -0.071489
       0.4    -0.11488
```

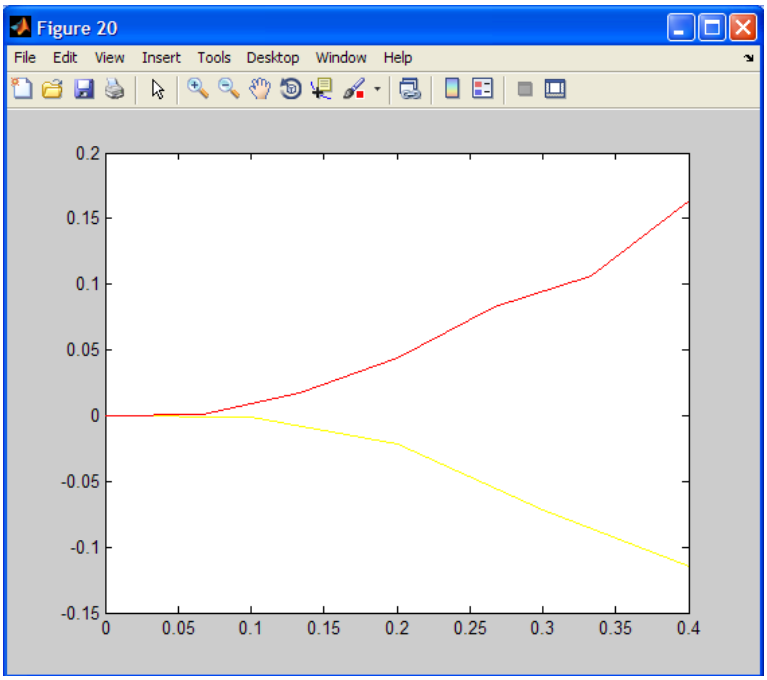
***>> plot(t,y,'y')***

***>> hold on***

***>> [t,y]=rk4('ejer16',[0,0.4],0,1/15);[t,y]***

```
ans =
         0         0
  0.066667  0.0004613
   0.13333   0.017793
       0.2   0.043725
   0.26667   0.083124
   0.33333    0.10634
       0.4    0.16323
```

***>> plot(t,y,'r')***



*% Notemos que estos h tomados no son muy pequeños, así que seguimos disminuyendo h pensando que la gráfica se pueda aproximar a una posible solución*

***>> [t,y]=rk4('ejer16',[0,0.4],0,1/20);[t,y]***

ans =

```
         0         0
      0.05  -0.00031761
       0.1    -0.015767
      0.15     -0.034536
       0.2    -0.061213
      0.25     -0.079559
       0.3     -0.11796
      0.35     -0.15182
       0.4     -0.16806
```

***>> plot(t,y,'.-')***

***>> [t,y]=rk4('ejer16',[0,0.4],0,1/27);[t,y]***

ans =

```
         0         0
  0.037037  0.00022526
  0.074074   0.0083956
   0.11111     0.02207
   0.14815    0.035381
   0.18519    0.051919
   0.22222    0.071978
   0.25926    0.095378
    0.2963     0.10739
   0.33333     0.12314
   0.37037     0.15163
       0.4     0.18126
```

***>> plot(t,y,'--')***

***>> [t,y]=rk4('ejer16',[0,0.4],0,0.0001);[t,y]***

ans =

```
         0         0
    0.0001 -9.8459e-008
    0.0002 -3.3625e-006
```

```
    0.0003 -8.2235e-006
    ..........  ..............
    ..........  ..............
    0.3994    -0.19495
    0.3995    -0.19498
    0.3996     -0.195
    0.3997    -0.19503
    0.3998    -0.19505
    0.3999    -0.19508
      0.4      -0.19511
```

***>> plot(t,y,'o')***

***>> [t,y]=rk4('ejer16',[0,0.4],0,0.000001);[t,y]***

ans =

```
         0         0
    1e-006    2.2605e-010
    ..........  ..........
    ..........  ..........
  0.002381   0.00021655
  0.002382   0.00021667
    ..........  ..........
    ..........  ..........
   0.39978    0.2062
    ..........  ..........
    ..........  ..........
    0.39999   0.20643
    ..........  ..........
    ..........  ..........
       0.4     0.20644
```

***>> plot(t,y,'*')***

*% Cambiamos de método: ode45 toma un paso variable, pequeño, intentando ajustar la aproximación y nos da una solución negativa*

***>> [t,y]=ode45('ejer16',[0,0.4],0);[t,y]***

ans =

```
         0         0
  0.0010746  -6.6225e-006
  0.0021491  -5.6317e-005
```

```
   0.0032237   -0.0001638
   0.0042982   -0.00030378
      .....       .....
      .....       .....

    0.39257     -0.19269
       0.4        -0.19469
```

*% Otras funciones Matlab, como ode23, comienzan con un tamaño de paso más pequeño y nos dan una solución positiva*

***>> [t,y]=ode23('ejer15',[0,0.4],0);[t,y]***

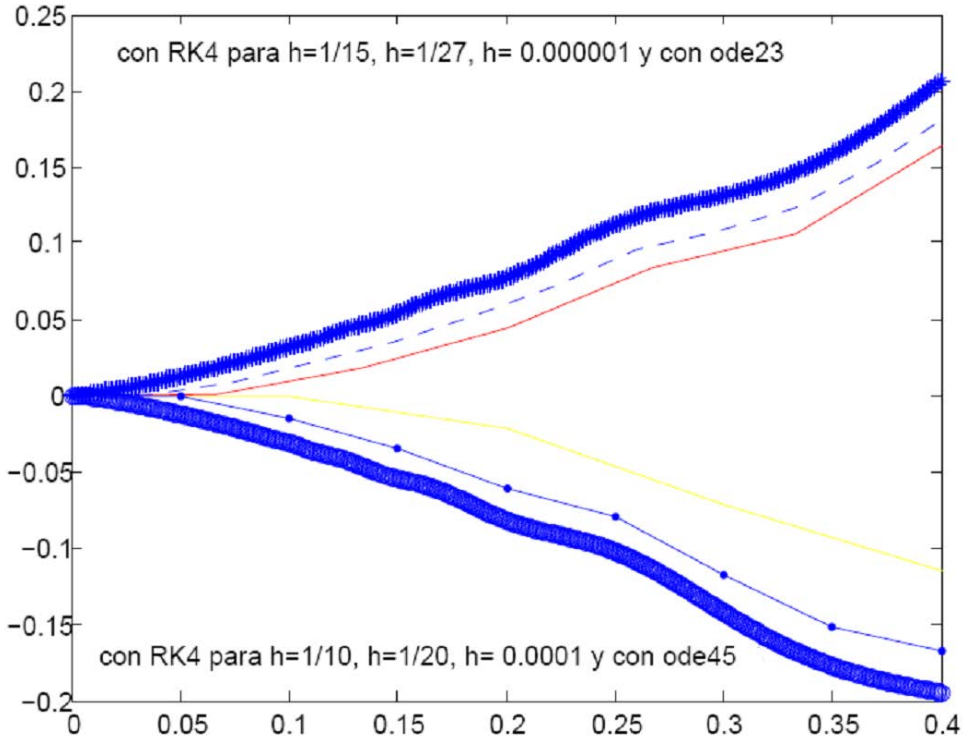
ans =

```
         0          0
    0.0003125  2.5561e-008
     0.000625  1.2406e-005
      .....       .....
      .....       .....
     0.35125     0.15913
     0.38286     0.18816
        0.4      0.20616
```

***>> gtext('con RK4 para h=1/10, h=1/20, h= 0.0001 y con ode45')***

***>> gtext('con RK4 para h=1/15, h=1/27, h= 0.000001 y con ode23')***

***% Observación:*** *Con rk4, para h=1/10, h=1/20 y h= 0.0001, y con ode45, obtenemos soluciones negativas (valores de y<0). Con rk4, para h=1/15, h=1/27 y h=0.000001, y con ode23 se tienen soluciones positivas. Parece que hay dos soluciones numéricas, y que aleatoriamente se encuentra una u otra aproximación dependiendo del tamaño de paso y del método (posiblemente modificando RelTol y AbsTol en ode23 y ode45 se obtengan distintas aproximaciones; ver ayuda en Matlab para el método implementado en cada función y conveniencia de uso según ED).*




***Ejercicio 5:***

*% Sobre influencia de los errores en los datos iniciales. Repetimos el segundo ejemplo del cuaderno utilizando métodos numéricos en vez de la resolución explícita. Se observa el mismo tipo de comportamiento: comparamos sólo las gráficas. Se trata del ejercicio 13 de la sección 1.7 del libro de apuntes*

13. Considerar el problema de Cauchy:

$$\begin{cases} y' &= x+y-3 \\ y(0) &= 2, \end{cases}$$

cuya solución exacta es $\varphi(x) = 2 - x$ en $(-\infty, \infty)$. Tomar como condición inical $y(0) = 2.001$ y calcular la solución exacta $\varphi_1(x)$. Comparar los valores de $\varphi(x)$ y $\varphi_1(x)$ para $x = 1$, para $x = \log 10^6$, y en general para $x \to \infty$. ¿Qué se puede decir, en general, sobre el buen planteamiento de un problema de Cauchy?.

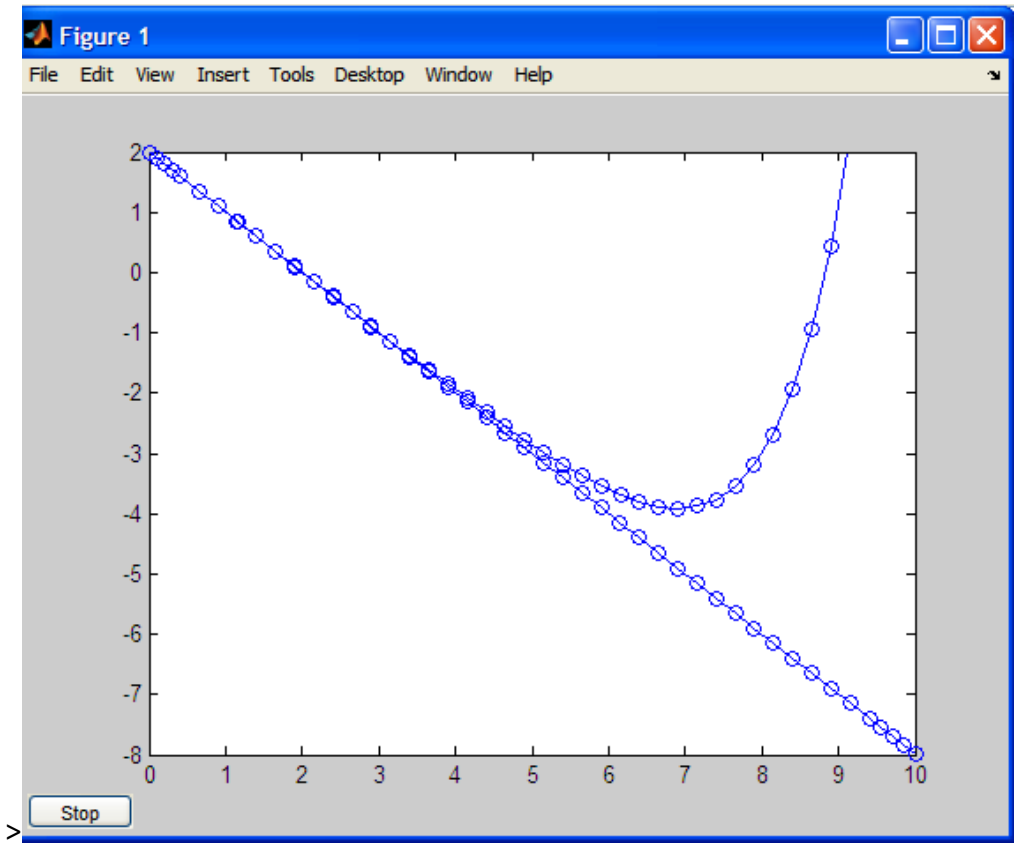
*% creamos la función Matlab ejerlinealdatoinicial.m*

***>> type ejerlinealdatoinicial***

***>> ode45('ejerlinealdatoinicial',[0,10],2);***

***>> hold on***

***>> ode45('ejerlinealdatoinicial',[0,10],2.001);***




>

*% las soluciones son casi coincidentes cerca de t=0, y luego son muy distintas.*

## *Aproximación numérica y resolución explícita de sistemas*

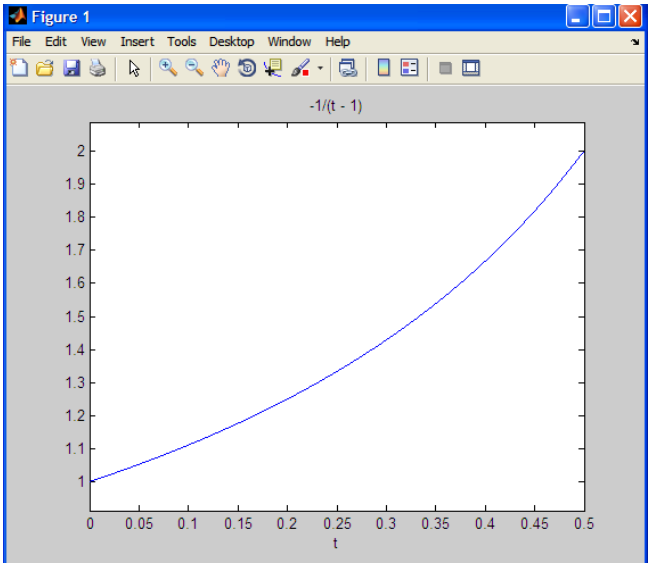
*% Seguimos el esquema del cuaderno para las ED de primer orden. Las ED de segundo orden, o de orden superior, se reducen a sistemas diferenciales de primer orden. A modo de introducción, resolvemos numéricamente un problema de Cauchy para una ED de segundo orden, del que se conoce la solución exacta (ver ejemplo 21 del libro de apuntes; sección 2.1). Comparamos la solución exacta y la solución aproximada con distintos métodos numéricos.*

***Ejercicio 1 (segunda parte de la hoja 4, sobre ED de segundo orden):***

*% Se considera el problema de valores iniciales y''=2y^3, y(0)=1, y'(0)=1. "dsolve" no nos da la solución, pero podemos comprobar que y=1/(1-t) es la solución exacta.*

```
>> dsolve('D2y=2*y^3','y(0)=1','Dy(0)=1')

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

 ans =

[ empty sym ]

>> syms t

>> u=1/(1-t)

u =

-1/(t - 1)

>> diff(u,2)-2*u^3 % verificación de solución

ans =

0

>> subs(u,0) % verificación de condición inicial y(0)=1

ans =

    1

>> subs(diff(u),0) % verificación de condición inicial y'(0)=1

ans =

    1
```

***>> ezplot(u,[0,1/2])*** *% gráfica de la solución en [0,1/2]*




*% A continuación, resolvemos numéricamente en [0,1/2] utilizando eul, rk2 y rk4 con distintos tamaños de paso h. Comparamos soluciones para deducir que rk4 "es el mejor". Resolvemos también con ode45.*

*% Reducimos la ED a sistema diferencial haciendo y1=y2, y2=y': y1'=y2, y2'=2(y1)^3 (y1, y2 las funciones incógnita). Introducimos la función vectorial definiendo el sistema en el fichero fejemplo21.m., e.g., fpendulo.m (sistema y1'=y2, y2'=-sin(y1) )*

```
function dydt = fpendulo(t,y)
dydt = [y(2); -sin(y(1))];
```

```
function dydt = fejemplo21(t,y)
dydt = [y(2); 2*(y(1))^3];
```

***>> type fejemplo21***

***>> clear t*** *% para no confundir la variable simbólica con el vector que nos da eul abajo*

***>>hold on*** *% para superponer gráficas*

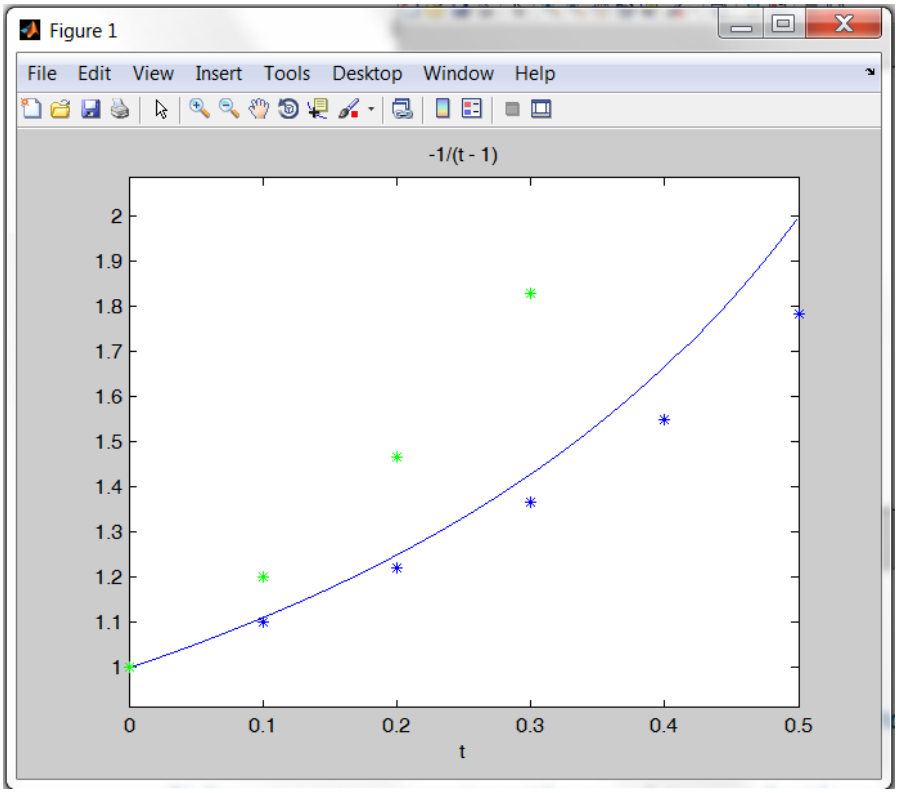
***>> [t,y]=eul('fejemplo21', [0,0.5],[1,1],0.1);[t,y]*** *% h=0.1*

ans =

| | | |
|---|---|---|
| 0 | 1.0000 | 1.0000 |
| 0.1000 | 1.1000 | 1.2000 |
| 0.2000 | 1.2200 | 1.4662 |
| 0.3000 | 1.3666 | 1.8294 |
| 0.4000 | 1.5496 | 2.3398 |
| 0.5000 | 1.7835 | 3.0840 |

*% la segunda columna es la aproxima de la solución u, y la tercera la aproximación de la derivada u'; el vector con condiciones iniciales es [1,1]*

***>> plot(t,y,'*')*** *% nos da la gráfica de la aproximación de u, y, de u' (puntos en verde)*




***>> plot(t,y(:,1),'* r')*** *% nos da la gráfica de la aproximación de u (puntos en rojo, abajo)*

*% Comparamos aproximación numérica y solución exacta en t=1/2*

***>> 1/0.5***

***ans =***

***2***

*% 2 es el valor de la solución u(0.5), mientras que la aproximación de la solución que hemos obtenido en t=0.5, con eul y h=0.1, es 1.7835. Hay que acercarse al valor 2 disminuyendo h. Comparamos gráficamente.*

***>> [t,y]=eul('fejemplo21', [0,0.5],[1,1],0.05);[t,y]*** *% h=0.05*

ans =

```
      0    1.0000    1.0000
 0.0500    1.0500    1.1000
 0.1000    1.1050    1.2158
 0.1500    1.1658    1.3507
 0.2000    1.2333    1.5091
 0.2500    1.3088    1.6967
 0.3000    1.3936    1.9209
 0.3500    1.4897    2.1916
 0.4000    1.5992    2.5221
 0.4500    1.7253    2.9311
 0.5000    1.8719    3.4448
```

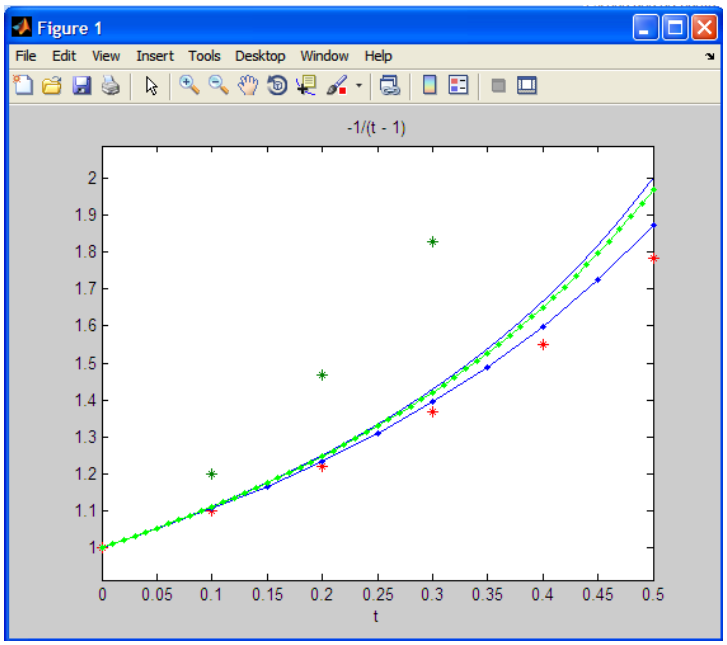
***>> plot(t,y(:,1),'.-')***

***>> [t,y]=eul('fejemplo21', [0,0.5],[1,1],0.01);[t,y]***

ans =

```
      0       1.0000    1.0000
 0.0100    1.0100    1.0200
 0.0200    1.0202    1.0406
 0.0300    1.0306    1.0618
 0.0400    1.0412    1.0837
 0.0500    1.0521    1.1063
   ..........................
   ..........................
 0.4300    1.7356    3.0024
 0.4400    1.7656    3.1070
 0.4500    1.7966    3.2171
 0.4600    1.8288    3.3331
 0.4700    1.8621    3.4554
 0.4800    1.8967    3.5845
 0.4900    1.9325    3.7210
 0.5000    1.9698    3.8654
```

***>> plot(t,y(:,1),'.- g')***




*% Cambiamos de método, esto es, usamos rk2 o rk4 con tamaño de paso más grande, y continuamos haciendo gráficas…*

***>> [t,y]=rk2('fejemplo21', [0,0.5],[1,1],0.1);[t,y]***

ans =

| | | |
|---|---|---|
| 0 | 1.0000 | 1.0000 |
| 0.1000 | 1.1100 | 1.2331 |
| 0.2000 | 1.2470 | 1.5575 |
| 0.3000 | 1.4221 | 2.0274 |
| 0.4000 | 1.6536 | 2.7440 |
| 0.5000 | 1.9732 | 3.9128 |

***>> plot(t,y(:,1),'.- r')***

***>> [t,y]=rk4('fejemplo21', [0,0.5],[1,1],0.1);[t,y]***

ans =

| | | |
|---|---|---|
| 0 | 1.0000 | 1.0000 |
| 0.1000 | 1.1111 | 1.2346 |
| 0.2000 | 1.2500 | 1.5625 |
| 0.3000 | 1.4285 | 2.0408 |
| 0.4000 | 1.6666 | 2.7778 |
| 0.5000 | 1.9998 | 4.0001 |

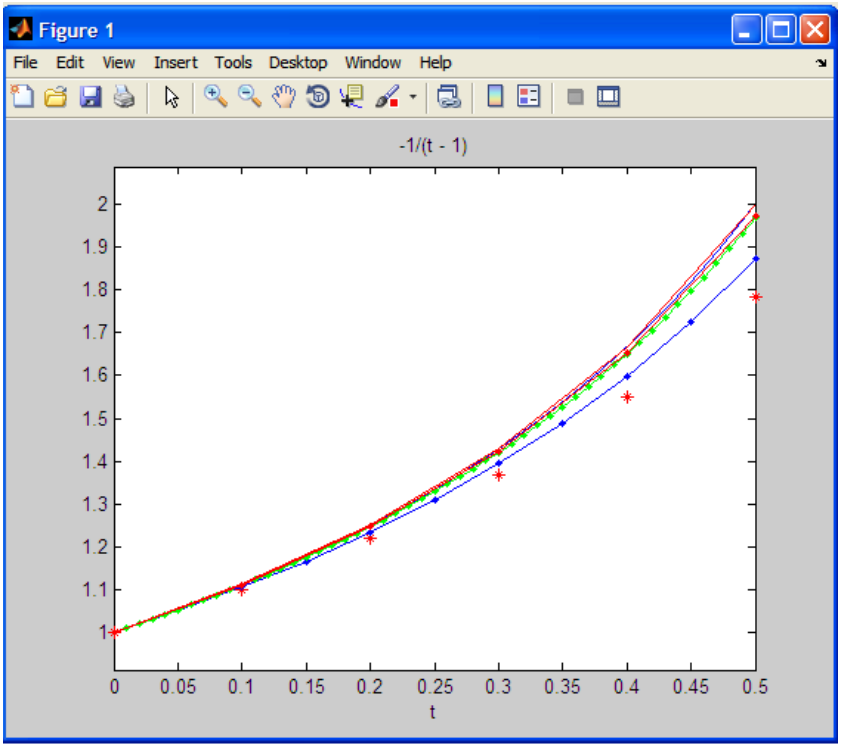
*% con rk4 y h=0.1, se tiene mejor aproximación que con rk2 y h=0.1, y que con eul y h=0.01. Probamos con ode45, que no necesita leer tamaño de paso.*

***>> [t,y]=ode45('fejemplo21', [0,0.5],[1,1]);[t,y]***

ans =

```
      0    1.0000    1.0000
 0.0125    1.0127    1.0255
 ..........................
 ..........................
 0.4750    1.9048    3.6281
 0.4875    1.9512    3.8073
 0.5000    2.0000    4.0000
```

***>> plot(t,y(:,1),'r')***




***>> hold off***

*% Aunque aparentemente la aproximación en t=2 es el valor exacto de la solución (también para la derivada u'(0.5)=4) , vemos que se sigue cometiendo error con*

***>>format long***

***>> [t,y]=ode45('fejemplo21', [0,0.5],[1,1]);[t,y]***

ans =

| | | |
|---|---|---|
| 0 | 1.000000000000000 | 1.000000000000000 |
| 0.012500000000000 | 1.012658234691001 | 1.025476712607703 |
| 0.025000000000000 | 1.025641032255032 | 1.051939525338238 |
| ..... | ..... | ..... |
| ..... | ..... | ..... |
| 0.475000000000000 | 1.904762184449091 | 3.628118482404317 |
| 0.487500000000000 | 1.951219548086893 | 3.807256208868066 |
| 0.500000000000000 | 2.000000025727396 | 3.999999718270436 |

*% Así, también con ode45 se comete un error numérico.*

*% Comparamos gráficas de distintas aproximaciones con la de la solución exacta, con el conjunto de comandos*

***>> syms t***

***>> u=1/(1-t)***

***>> ezplot(u,[0,1/2])***

***>> clear t***

***>>hold on***

***>> [t,y]=eul('fejemplo21',[0,1/2],[1,1],0.05);***

***>>plot(t,y(:,1),'--')***

***>> [t,y]=eul('fejemplo21',[0,1/2],[1,1],0.01);***

***>>plot(t,y(:,1),'y')***

***>> [t,y]=rk2('fejemplo21',[0,1/2],[1,1],0.1);***

***>>plot(t,y(:,1),'g')***

***>>[t,y]=rk4('fejemplo21',[0,1/2],[1,1],0.1);***

***>> plot(t,y(:,1),'r')***

***>> gtext('eul -- h=0.51')***

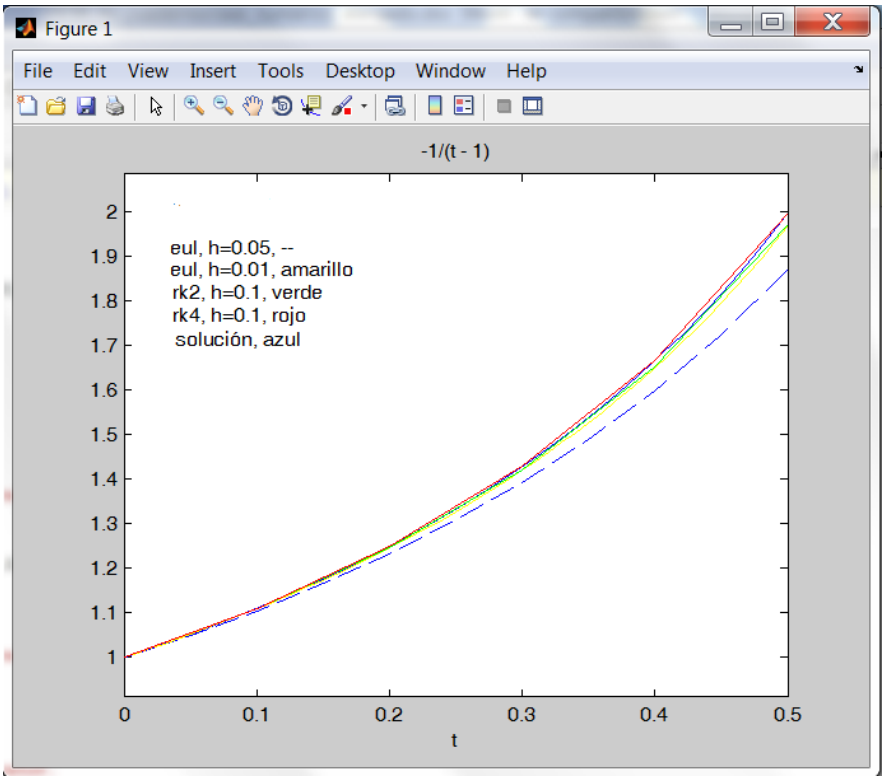
***>> gtext('eul, h=0.05, --')***

***>> gtext('eul, h=0.01, amarillo')***

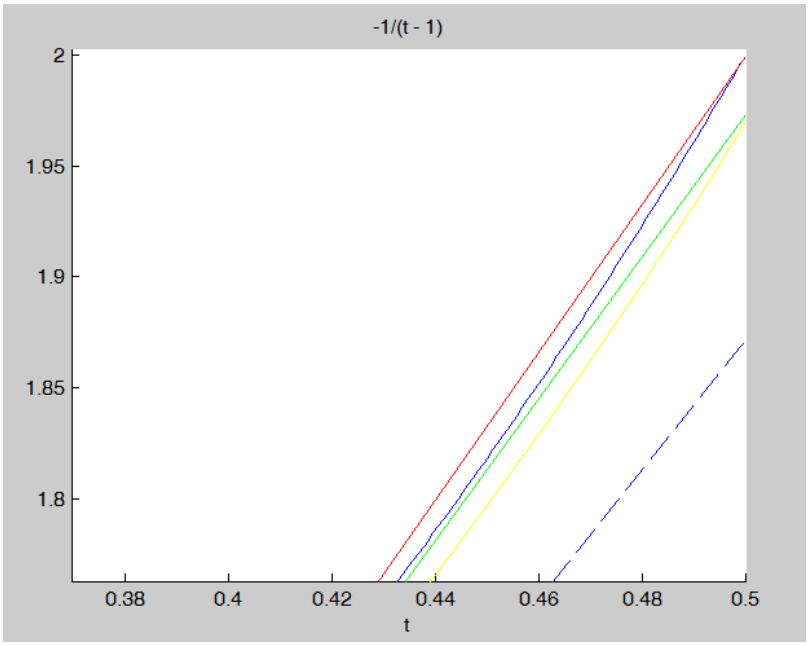
***>> gtext('rk2, h=0.1, verde')***

***>> gtext('rk4, h=0.1, rojo')***

***>> gtext('solución, azul')***




*% ampliando la gráfica, vemos que las aproximaciones son distintas entre sí y distintas de la solución exacta.*




*% A continuación, también a modo de introducción, resolvemos un sistema numéricamente usando "ode45"; como se puede resolver explícitamente, se compara la solución numérica con la solución exacta.*

***Ejemplo:***

$$\begin{cases} y_1' &= 4y_1 + 5y_2 + e^x \\ y_2' &= -2y_1 - 2y_2 \\ y_1(0) &= 3\ ,\ y_2(0) = -2 \end{cases}$$

***>> dsolve('Dy1=4*y1+5*y2+exp(t)','Dy2=-2*y1-2*y2', 'y1(0)=3','y2(0)=-2')***

ans =

y2: [1x1 sym]
y1: [1x1 sym]

*% hay que indicar dónde queremos guardar las dos componentes de la solución Y1 e Y2,*

```
>> [Y1,Y2]=dsolve('Dy1=4*y1+5*y2+exp(t)','Dy2=-2*y1-2*y2', 'y1(0)=3','y2(0)=-2')

Y1 =

3*exp(t)


Y2 =

-2*exp(t)
```

*% valor de las componentes en t=1*

```
>> v1=3*exp(1)

v1 =

   8.1548

>> v2=-2*exp(1)

v2 =

  -5.4366
```

*% Para resolver numéricamente, introducimos la función vectorial definiendo nuestro sistema en el fichero fsistema.m. Modificando, e.g., fpendulo.m (sistema y1'=y2, y2'=-sin(y1) )*

```
function dydt = fpendulo(t,y)
dydt = [y(2); -sin(y(1))];
```

```
function dydt = fsistema(t,y)
dydt = [4*y(1)+5*y(2)+exp(t); -2*y(1)-2*y(2)];
```

```
>> type fsistema

>> [t,y]=ode45('fsistema',[0,1],[3,-2]);[t,y]

ans =

         0    3.0000   -2.0000
    0.0250    3.0759   -2.0506
    0.0500    3.1538   -2.1025
    .....     .....    .....
    .....     .....    .....
```

```
    0.9500    7.7571   -5.1714
    0.9750    7.9535   -5.3023
    1.0000    8.1548   -5.4366
```

*% La aproximación de la solución está en las columnas segunda (aproximación de la primera componente de la solución Y1) y tercera (Y2). Nos referimos a estas componentes mediante y(:,1) e y(:,2) respectivamente. Aunque aparentemente la solución exacta coincide con la numérica en t=1, basta con considerar el formato de escritura más largo para ver que no es así*

***>> format long***

***>> [t,y]=ode45('fsistema',[0,1],[3,-2]);[t,y]***

ans =

```
                   0   3.000000000000000  -2.000000000000000
   0.025000000000000   3.075945343827411  -2.050630232997753
   0.050000000000000   3.153813252801293  -2.102542175263521
   ..................  ..................  ..................
   ..................  ..................  ..................
   0.975000000000000   7.953501522737760  -5.302334351932623
   1.000000000000000   8.154845427650947  -5.436563612106495
```

***>> v2+5.436563612106495***

ans =

```
  -4.481159621150255e-008
```

***>> v1-8.154845427650947***

ans =

```
   5.772619005028901e-008
```

*% como siempre, hay un error numérico en la aproximación. Este error no se observa en las gráficas (abajo)*

***>> plot(t,y)*** *% gráficas de las dos componentes de la aproximación (t,y(:,1)) y (t,y(:,2))*

***>> plot(t,y,'*')*** *% cambiamos el formato de dibujo*

***>> hold on***

***>> syms t***

***>> [Y1,Y2]=dsolve('Dy1=4*y1+5*y2+exp(t)','Dy2=-2*y1-2*y2', 'y1(0)=3','y2(0)=-2')***

Y1 =

3*exp(t)

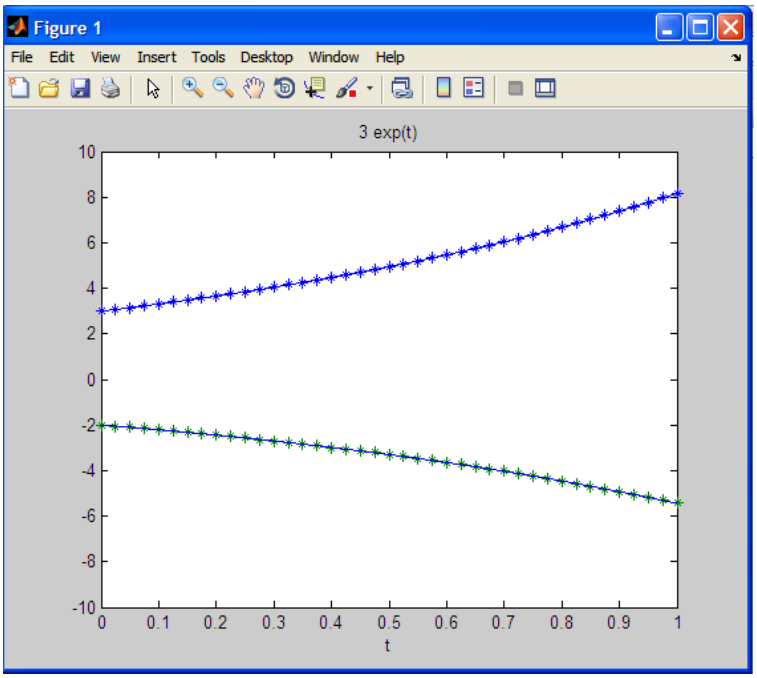
Y2 =

-2*exp(t)

***>> ezplot(Y1,[0,1])*** *% gráfica de la primera componente de la solución*

***>> ezplot(Y2,[0,1])*** *% gráfica de la segunda componente de la solución*

***>> axis([0,1,-10,10])***




## *Problemas de contorno: el método de tiro*

*% Dado el problema y''=f(x,y,y'), y(0)=0, y(1)=0, supuesto que tenga solución única, para aproximarla, utilizamos "ode45" resolviendo numéricamente y''=f(x,y,y'), y(0)=0, y'(0)=beta, para un beta que tenemos que ajustar de tal manera que y(1) valga 0 o se aproxime a 0. De los valores numéricos que nos da ode45, nos interesa el valor de la primera componente, y(:,1), en el punto 1. Comparamos las gráficas y los valores de y(:,1) en el punto 1 para distintos betas.*

***Ejercicio 5 (segunda parte de la hoja 4 sobre ED de segundo orden):***

$$y'' + y' + y^2x^3 = 1, \quad x \in (0,1)$$
$$y(0) = 0, \quad y(1) = 0$$

*% Consideramos la ED junto con las condiciones iniciales y(0)=0, y'(0)=beta. Tomamos beta =1, -1, 0.5, -0.5,..., y resolvemos estos problemas de Cauchy con "ode45" viendo que los primeros valores de beta nos dan muy malas aproximaciones de la solución del problema de contorno, y que empiezan a mejorar con beta=-0.5*

*% Se trata de una ED de segundo orden no lineal, reducimos a sistema y1'=y2 e y2'=1-y2-y1^2*x^3, e introducimos la función vectorial en el fichero fmitiro.m, por ejemplo, modificando fpendulo.m*

```
function dydt = fpendulo(t,y)
dydt = [y(2); -sin(y(1))];
```

```
function dydx=fmitiro(x,y)
dydx = [y(2); 1-(y(1))^2*x^3-y(2)];
```

***>> type fmitiro***

*% empezamos tomando y(0)=0, y'(0)=beta=1*

***>> [t,y]=ode45('fmitiro',[0,1],[0,1]);[t,y]***

ans =

| | | |
|---|---|---|
| 0 | 0 | 1.0000 |
| 0.0001 | 0.0001 | 1.0000 |
| 0.0001 | 0.0001 | 1.0000 |
| 0.0002 | 0.0002 | 1.0000 |
| 0.0002 | 0.0002 | 1.0000 |
| 0.0005 | 0.0005 | 1.0000 |
| ..... | ..... | ..... |
| ..... | ..... | ..... |
| 0.9313 | 0.9185 | 0.9056 |
| 0.9485 | 0.9340 | 0.8951 |
| 0.9657 | 0.9492 | 0.8836 |
| 0.9828 | 0.9643 | 0.8712 |
| 1.0000 | 0.9791 | 0.8577 |

***>> plot(t,y(:,1))***

*% tenemos una muy mala aproximación ya que y(1) se aproxima por 0.97… y necesitamos que se aproxime a 0, así que tomamos beta =-1*

**>> [t,y]=ode45('fmitiro',[0,1],[0,-1]);[t,y]** *% y(0)=0, y'(0)=beta=-1*

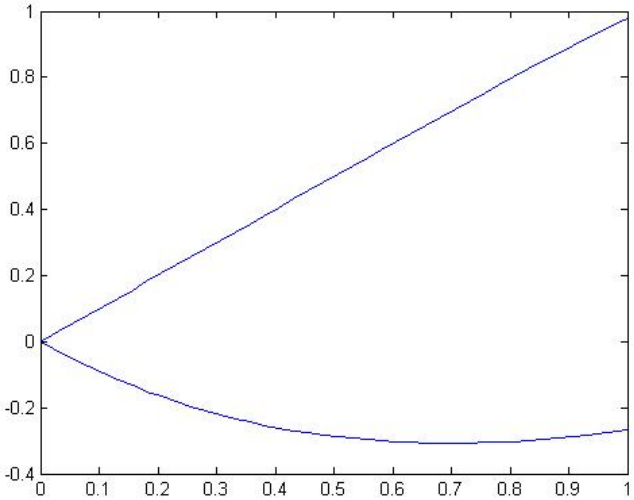
ans =

| | | |
|---|---|---|
| 0 | 0 | -1.0000 |
| 0.0001 | -0.0001 | -0.9999 |
| 0.0001 | -0.0001 | -0.9998 |
| 0.0002 | -0.0002 | -0.9997 |
| 0.0002 | -0.0002 | -0.9996 |
| 0.0005 | -0.0005 | -0.9991 |
| 0.0007 | -0.0007 | -0.9986 |
| …… | …… | …… |
| …… | …… | …… |
| 0.9063 | -0.2879 | 0.1794 |
| 0.9313 | -0.2832 | 0.1981 |
| 0.9485 | -0.2797 | 0.2106 |
| 0.9657 | -0.2760 | 0.2229 |
| 0.9828 | -0.2720 | 0.2350 |
| 1.0000 | -0.2679 | 0.2468 |

*% tenemos una mala aproximación (y(1) se aproxima por -0.26…), así que tomamos beta =0.5. Las gráficas nos muestran las soluciones para beta=1 y beta=-1, así como la necesidad de seguir ajustando beta*

***>> hold on***

***>> plot(t,y(:,1))***



***>> [t,y]=ode45('fmitiro',[0,1],[0,0.5]);[t,y]*** *% y(0)=0, y'(0)=beta=0.5*
ans =

| | | |
|---|---|---|
| 0 | 0 | 0.5000 |
| 0.0001 | 0.0001 | 0.5001 |
| 0.0002 | 0.0001 | 0.5001 |
| 0.0003 | 0.0002 | 0.5002 |
| 0.0004 | 0.0002 | 0.5002 |
| 0.0009 | 0.0005 | 0.5005 |
| ……. | …… | …… |
| ……. | …… | …… |
| 0.9720 | 0.6540 | 0.7574 |
| 0.9813 | 0.6610 | 0.7559 |
| 0.9907 | 0.6681 | 0.7543 |
| 1.0000 | 0.6751 | 0.7524 |

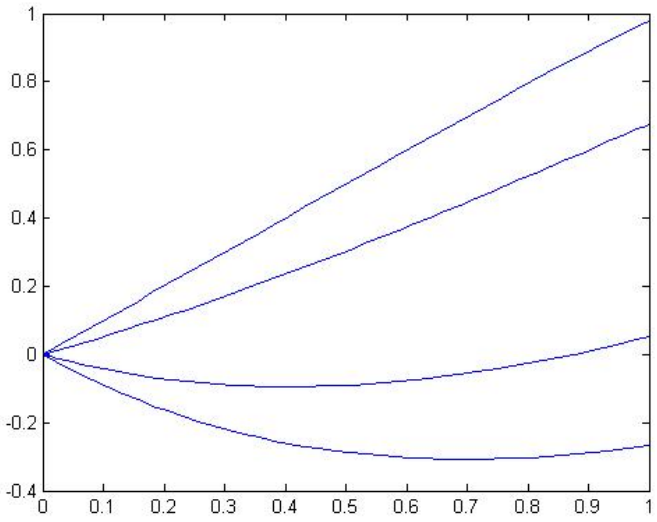
***>> plot(t,y(:,1))***

***>> [t,y]=ode45('fmitiro',[0,1],[0,-0.5]);[t,y]*** *% y(0)=0, y'(0)=beta=-0.5*

ans =

| | | |
|---|---|---|
| 0 | 0 | -0.5000 |
| 0.0001 | -0.0001 | -0.4998 |
| 0.0002 | -0.0001 | -0.4997 |
| 0.0003 | -0.0002 | -0.4995 |
| 0.0004 | -0.0002 | -0.4994 |
| 0.0009 | -0.0005 | -0.4986 |
| …… | ……. | ……. |
| …… | ……. | ……. |
| 0.9720 | 0.0394 | 0.4322 |
| 0.9813 | 0.0434 | 0.4374 |
| 0.9907 | 0.0475 | 0.4426 |
| 1.0000 | 0.0517 | 0.4478 |

***>> plot(t,y(:,1))***




***>> gtext('beta=0,5')***

***>> gtext('beta=-0,5')***

*% Con beta=-0.5 estamos bastante más cerca de y(1)=0; elegimos otro beta entre 0.5 y -0.5, y seguimos ajustando, abajo y(0)=0, y'(0)=beta=-0.57 y otros con los que nos vamos aproximando más a la solución del problema de contorno.*

***>> [t,y]=ode45('fmitiro',[0,1],[0,-0.57]);[t,y]*** *% y(0)=0, y'(0)=beta=-0.57*

ans =

| | | |
|---|---|---|
| 0 | 0 | -0.5700 |
| 0.0001 | -0.0001 | -0.5699 |
| 0.0002 | -0.0001 | -0.5697 |
| 0.0003 | -0.0002 | -0.5696 |
| ..... | ..... | ..... |
| ..... | ..... | ..... |
| 0.9550 | -0.0111 | 0.3950 |
| 0.9662 | -0.0067 | 0.4018 |
| 0.9775 | -0.0021 | 0.4085 |
| 0.9887 | 0.0025 | 0.4151 |
| 1.0000 | 0.0072 | 0.4216 |

***>> [t,y]=ode45('fmitiro',[0,1],[0,-0.58]);[t,y]*** *% y(0)=0, y'(0)=beta=-0.58*

ans =

| | | |
|---|---|---|
| 0 | 0 | -0.5800 |
| 0.0001 | -0.0001 | -0.5799 |
| 0.0002 | -0.0001 | -0.5797 |
| 0.0003 | -0.0002 | -0.5796 |
| 0.0003 | -0.0002 | -0.5795 |
| 0.0008 | -0.0005 | -0.5788 |
| ..... | ..... | ..... |
| ..... | ..... | ..... |
| 0.9040 | -0.0364 | 0.3593 |
| 0.9290 | -0.0273 | 0.3751 |
| 0.9540 | -0.0177 | 0.3905 |
| 0.9655 | -0.0132 | 0.3974 |
| 0.9770 | -0.0086 | 0.4043 |
| 0.9885 | -0.0039 | 0.4111 |
| 1.0000 | 0.0009 | 0.4178 |

***>> plot(t,y(:,1))***

***>> gtext('beta=-0,58')***

***>> [t,y]=ode45('fmitiro',[0,1],[0,-0.5813]);[t,y]*** *% y(0)=0, y'(0)=beta=-0.5813*

ans =

| | | |
|---|---|---|
| 0 | 0 | -0.5813 |
| 0.0001 | -0.0001 | -0.5812 |
| 0.0002 | -0.0001 | -0.5810 |
| 0.0003 | -0.0002 | -0.5809 |
| 0.0003 | -0.0002 | -0.5808 |
| ..... | ..... | ..... |
| ..... | ..... | ..... |
| 0.9654 | -0.0140 | 0.3969 |
| 0.9770 | -0.0094 | 0.4038 |
| 0.9885 | -0.0047 | 0.4106 |
| 1.0000 | 0.0001 | 0.4174 |

***>> [t,y]=ode45('fmitiro',[0,1],[0,-0.5815]);[t,y]*** *% y(0)=0, y'(0)=beta=-0.5815*

ans =

| | | |
|---|---|---|
| 0 | 0 | -0.5815 |
| 0.0001 | -0.0001 | -0.5814 |
| 0.0002 | -0.0001 | -0.5812 |
| 0.0003 | -0.0002 | -0.5811 |
| 0.0003 | -0.0002 | -0.5810 |
| ..... | ..... | ..... |
| ..... | ..... | ..... |
| 0.9654 | -0.0141 | 0.3968 |
| 0.9770 | -0.0095 | 0.4037 |
| 0.9885 | -0.0048 | 0.4105 |
| 1.0000 | -0.0001 | 0.4173 |

***>> [t,y]=ode45('fmitiro',[0,1],[0,-0.5814]);[t,y]*** *% y(0)=0, y'(0)=beta=-0.5814*

ans =

| | | |
|---|---|---|
| 0 | 0 | -0.5814 |
| 0.0001 | -0.0001 | -0.5813 |
| 0.0002 | -0.0001 | -0.5811 |
| 0.0003 | -0.0002 | -0.5810 |
| ..... | ..... | ..... |
| ..... | ..... | ..... |
| 0.9654 | -0.0141 | 0.3968 |
| 0.9770 | -0.0095 | 0.4037 |
| 0.9885 | -0.0048 | 0.4106 |
| 1.0000 | -0.0000 | 0.4173 |

***>> plot(t,y(:,1),'y')***

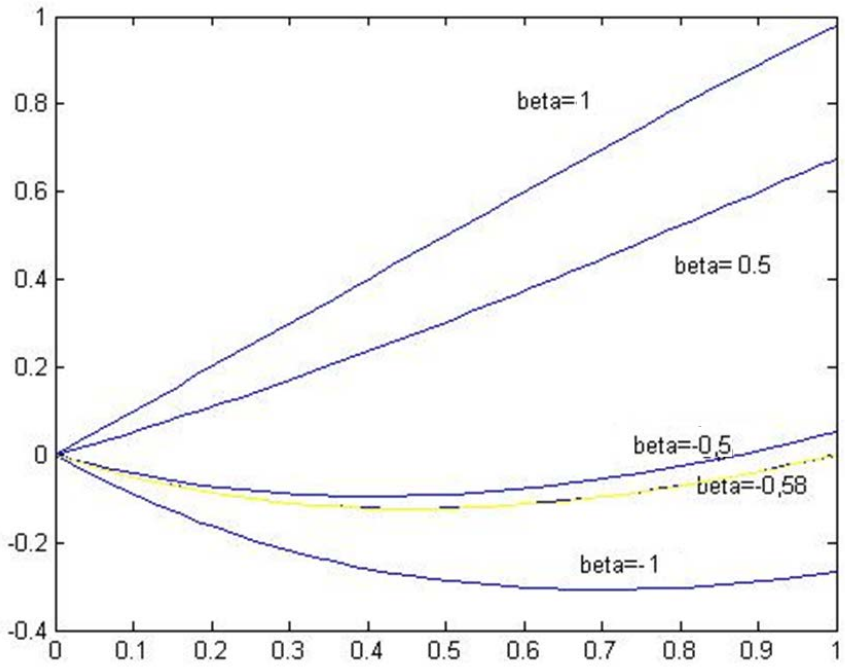
*% beta=-0.5814 da la aproximación de y(1) por 0., pero cambiando el formato de lectura vemos que dicha aproximación no es 0, sino -0.000002515627454*

***>>format long***

***>> [t,y]=ode45('fmitiro',[0,1],[0,-0.5814]);[t,y]***

ans =

| | | |
|---|---|---|
| 0 | 0 | -0.581400000000000 |
| 0.000086408201978 | -0.000050231825136 | -0.581263359972886 |
| 0.000172816403957 | -0.000100451843962 | -0.581126731752081 |
| …… | …… | …… |
| …… | …… | …… |
| 0.965439038525851 | -0.014073424715079 | 0.396830587338042 |
| 0.976959359017234 | -0.009461946267669 | 0.403737951346444 |
| 0.988479679508617 | -0.004771343753789 | 0.410567097110926 |
| 1.000000000000000 | -0.000002515627454 | 0.417318503535054 |




*% De los betas elegidos, beta= -0.5814 es el que mejor aproximación nos proporciona; pero en la gráfica, no se nota ya mucha diferencia entre beta=-0.58 y este otro valor beta= -0.5814 (gráfica en amarillo).*

***% Observación:** no se ha abordado el problema de la existencia y unicidad de solución del problema de contorno para la ecuación no lineal y''+y'+y^2x^3=1 para las ecuaciones lineales de los ejercicios que siguen, esto es consecuencia del teorema de la alternativa de Fredholm (ver sección 5.1 del libro de apuntes y cuadernos octavo y décimo del curso)*

***Ejercicio 6 (segunda parte de la hoja 4 sobre ED de segundo orden):***

*% Aplicamos el método de tiro para resolver un problema de contorno*

$$y'' - y = x^2, \quad x \in (0,1)$$
$$y(0) = 0, \quad y(1) = 0$$

*% Resolvemos el problema de valores iniciales y''-y=x^2, y(0)=0, y'(0)=beta, probando con diversos valores de beta (incluyendo 1,-1,0.5,-0.5 y otros), y haciendo las gráficas de las soluciones en distintos colores, hasta llegar ajustar beta=-0.073. Como se puede encontrar una solución explícita, se resuelve, y se compara la solución exacta con la aproximación.*

*% Introducimos la función vectorial en fcont1.m*

***>> type fcont1***

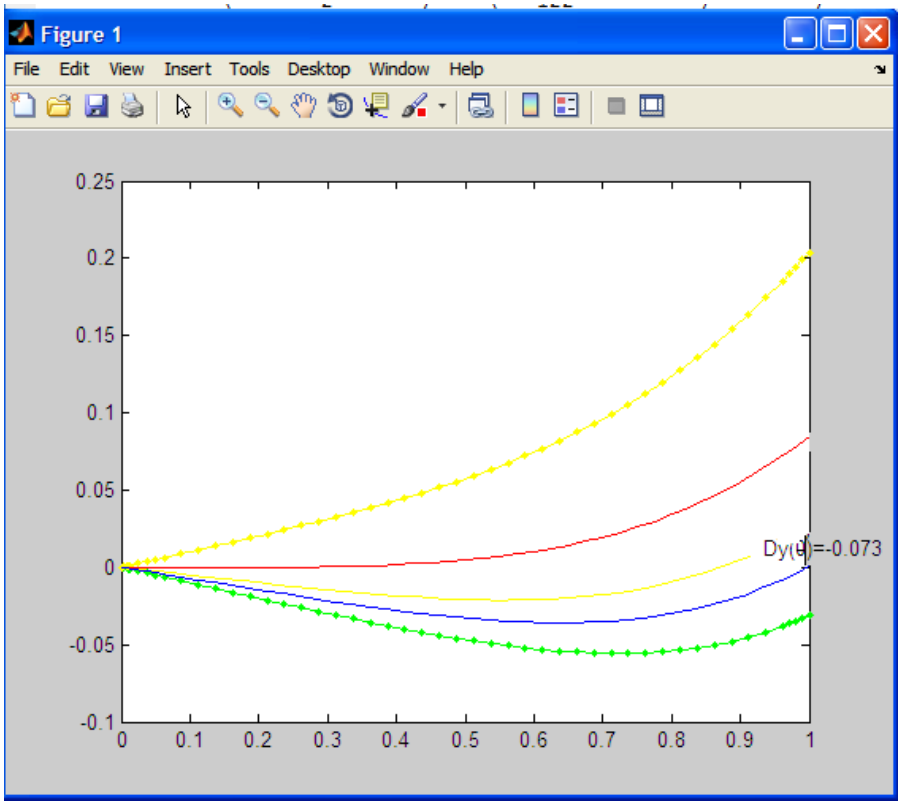
***>> format long***

***>> [t,y]=ode45('fcont1',[0,1],[0,-0.073]);[t,y]***

ans =

| | | |
|---|---|---|
| 0 | 0 | -0.073000000000000 |
| 0.000688188063427 | -0.000050237732577 | -0.073000017177861 |
| 0.001376376126855 | -0.000100475488685 | -0.073000068276880 |
| 0.002064564190282 | -0.000150713291444 | -0.073000152645228 |
| 0.002752752253709 | -0.000200951163525 | -0.073000269631087 |
| 0.006193692570846 | -0.000452142325870 | -0.073001321010506 |
| ..... | ..... | ..... |
| ..... | ..... | ..... |
| 0.989001489898737 | -0.002183382967043 | 0.226889590182192 |
| 0.992667659932492 | -0.001344990307469 | 0.230482396453027 |
| 0.996333829966246 | -0.000493371261129 | 0.234104985125714 |
| 1.000000000000000 | 0.000371583629911 | 0.237757503443994 |

***>> gtext('Dy(0)=-0.073')***




***>> dsolve('D2y-y=t^2','y(0)=0','y(1)=0')*** *% resolución explícita*

ans =

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

***>> pretty(ans)***

```
exp(t) (3 exp(1) - 2)     2    3 exp(1) - 2 exp(2)
---------------------  - t  - -------------------  - 2
     exp(2) - 1                exp(t) (exp(2) - 1)
```

*% de hecho se puede derivar y calcular y'(0)*

***>> diff(ans)***

ans =

(exp(t)*(3*exp(1) - 2))/(exp(2) - 1) - 2*t + (3*exp(1) - 2*exp(2))/(exp(t)*(exp(2) - 1))

***>> subs(ans,t,0)***

ans =

-0.0733

*% que es bastante próxima del beta que hemos tomado para la gráfica amarilla, es decir, para la aproximación numérica.*

***Ejercicio 7 (segunda parte de la hoja 4 sobre ED de segundo orden):***

$$y'' - x^2 y = e^x, \quad x \in (0,1)$$
$$y(0) = 0, \quad y(1) = 0$$

*% Matlab no resuelve y''-t^2y=exp(t) (en términos de funciones elementales) y menos el problema de contorno dado, pero se puede comprobar que el problema de contorno homogéneo asociado tiene sólo la solución trivial. Como consecuencia, el problema dado tiene solución única y se puede proceder como en los ejercicios anteriores para su aproximación: ver resultados en el cuaderno 10.*

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

Warning: Possibly missing solutions. [solvini]
Warning: Explicit solution could not be found.
> In dsolve at 161

ans =

[ empty sym ]

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

ans =

0

*% **Observación:** Modelos matemáticos en que intervienen sistemas diferenciales y ED de orden dos o superior se consideran en el cuaderno quinto. También, los problemas de contorno se consideran en los cuadernos octavo y décimo del curso.*

---