Problemas de contorno: Series de Fourier y aproximación numérica de soluciones.

Capítulo 5 - Libro* + *Hoja de problemas 8* + *Complementos

Resumen de contenidos:

- **Resoluciones explícitas (si se puede)**
- **Desarrollos en serie de Fourier**
- **Aproximación de soluciones: métodos de Galerkin y de elementos finitos**

% Los dos apartados de este cuaderno, relativos a series de Fourier y aproximación de soluciones, se complementan con los cuadernos 9 y 10 respectivamente.

Primera parte: Problemas de contorno/series de Fourier

*% **Problemas de contorno**: verificamos el teorema de la alternativa de Fredholm, para un modelo de viga de segundo orden, sujeta en los extremos: EIy''+Ty=p(x), x en (0,l), y(0)=0, y(l)=0.*

% Tomando l=1, T/EI=pi^2:

>> syms t

>> dsolve('D2y+pi^2*y=0')

ans =

C1*sin(pi*t)+C2*cos(pi*t)

% las constantes de integración C1 y C2 se determinan, si es posible, con las condiciones de contorno

>> dsolve('D2y+pi^2*y=0','y(0)=0','y(1)=0')

ans =

C1*sin(pi*t)

*% Así pues, obtenemos soluciones distintas de la trivial; de alguna manera esto implica que no se puede predecir el comportamiento de la viga: en términos matemáticos, no existe solución única de cualquier problema de contorno no homogéneo asociado, tal y como se ve a continuación, introduciendo distintos datos no homogéneos (p(t)= exp(t) o p(t)=sin(2*pi*t), y EI=1)*

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

Warning: Explicit solution could not be found.
> In dsolve at 333

ans =

[empty sym]

% no existe solucion para p=exp(t)

>> dsolve('D2y+pi^2*y=sin(2*pi*t)','y(0)=0','y(1)=0')

ans =

sin(pi*t)*C2-1/3*sin(2*pi*t)/pi^2

>> pretty(ans)

```
                                  sin(2 pi t)
                sin(pi t) C2 - 1/3 -----------
                                        2
                                      Pi
```

*% para p(t)=sin(2*pi*t) hay infinitas soluciones, una para cada valor de C2. Así, por ejemplo, para T/EI=pi^2: la solución no se puede aproximar numéricamente. Esto se comprueba en la última parte de este cuaderno. Para otras relaciones entre T y EI tales que T/EI distinto de k^2pi^2, existe una única solución del problema homogéneo (la trivial), y de todos los problemas no homogéneos asociados que se consideren. Tomamos T=EI=1*

>> dsolve('D2y+y=0','y(0)=0','y(1)=0')

ans =

0

>> dsolve('D2y+y=sin(2*pi*t)','y(0)=0','y(1)=0')

ans =

-sin(2*pi*t)/(-1+4*pi^2)

>> dsolve('D2y+y=exp(t)','y(0)=0','y(1)=0')

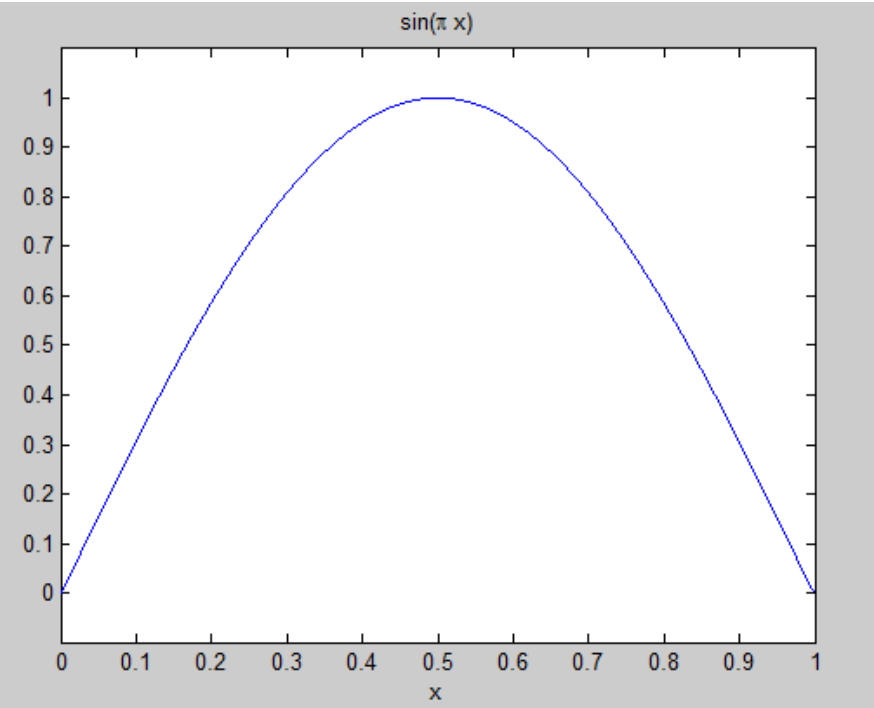
ans =

-1/2*sin(t)*(-cos(1)+exp(1))/sin(1)-1/2*cos(t)+1/2*exp(t)

*% La razón de esto es que las llamadas "cargas de pandeo'' en el modelo: EI y''+Ty=0, y(0)=0, y(1)=0, están dadas por T/EI=(k pi)^2, para k=1,2,3,….. "los valores propios". Así para la relación T/EI=pi^2, se tiene la llamada "carga crítica de pandeo", y la forma "en que se deforma la viga", para p(x)=0, está dada por la gráfica de la función sin(pi*x) (salvo constante multiplicativa):*

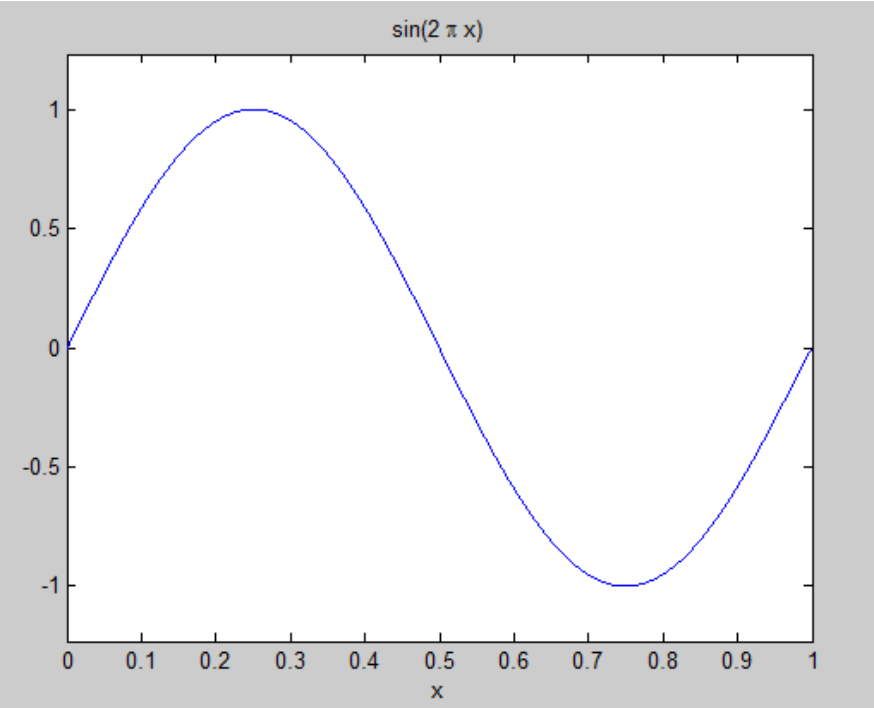
>> syms x

>> ezplot(sin(pi*x),[0,1])

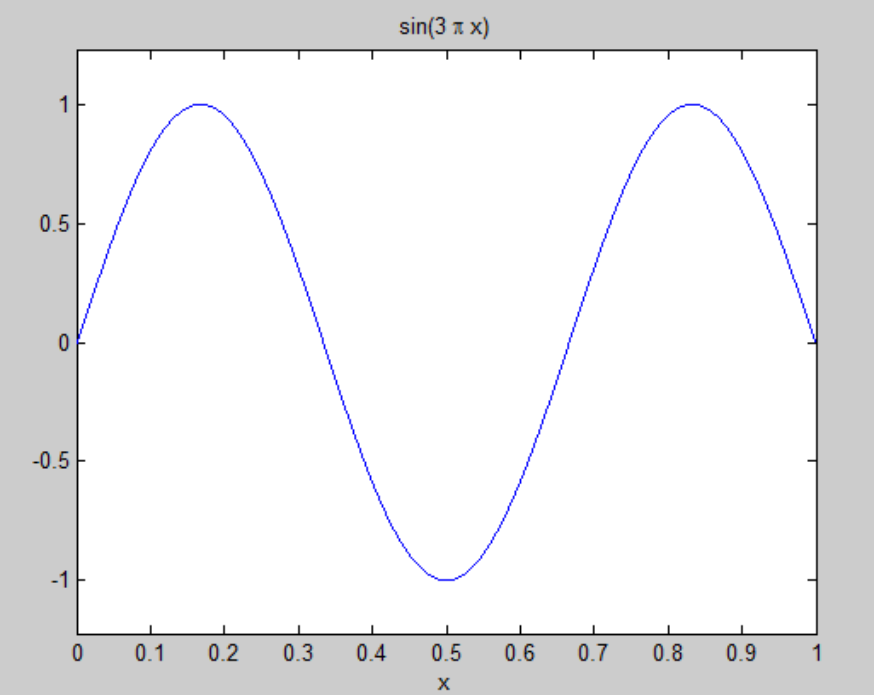


% De manera análoga para los valores de T/EI= 4pi^2 y 9pi^2 se tienen las gráficas de la solución de EIy''+Ty=0, y(0)=0, y(1)=0 (salvo constante multiplicativa):

>> ezplot(sin(2*pi*x),[0,1])



>> ezplot(sin(3*pi*x),[0,1])



*% Por otro lado, para cada valor propio k^2pi^2, hay una función propia asociada, y_k=sin(k*pi*x); comprobamos que funciones propias asociadas a distintos valores propios son ortogonales entre sí en (0,1):*

>> int(sin(k*pi*t)*sin(j*pi*t),0,1)

??? Undefined function or variable 'k'

% obviamente, da error, dado que hay que definir las variable simbólicas k y j para la integración

>> syms k j

>> int(sin(k*pi*t)*sin(j*pi*t),0,1)

ans =

-(-k*cos(k*pi)*sin(j*pi)+j*sin(k*pi)*cos(j*pi))/pi/(-k^2+j^2)

>> pretty(ans)

```
                  -k cos(k pi) sin(j pi) + j sin(k pi) cos(j pi)
                - ---------------------------------------------
                                      2    2
                                pi (-k  + j )
```

% si k es distinto de j la integral calculada vale 0, y para k=j, se tiene

>> int((sin(k*pi*t))^2,0,1)

ans =

1/2*(-cos(k*pi)*sin(k*pi)+k*pi)/k/pi

>> pretty(ans)

```
                                    -cos(k pi) sin(k pi) + k pi
                               1/2 ---------------------------
                                              k pi
```

*% **Desarrollos en serie de Fourier**: Cualquier función f(x) continua a trozos en [0,1] se escribe en términos de las funciones propias calculadas sin(k*pi*x), como una suma sum_{k=1}^infty c_k sin(k*pi*x) donde las constantes c_k son los coeficientes de Fourier: c_k=int(f(x)*sin(k*pi*x),0,1)/ int((sin(k*pi*x))^2,0,1). La función Matlab seriefourier.m lee n, dibuja las sumas parciales n-ésimas de la serie de Fourier y compara la gráfica con la de la función f(x) (f(x)=x), permitiéndonos analizar el tipo de convergencia que se tiene al variar la función f*

>> type seriefourier

>> help seriefourier *% nos da información sobre la función*

```
OCW Universidad de Cantabria
Cálculo Simbólico y Numérico en Ecuaciones Diferenciales
%
 La funcion seriefourier.m lee n el numero de funciones propias para
 aproximar una funcion dada  f(x);
 calcula los coeficientes  de Fourier c_j de la función f que definimos
 dentro de la función, y la aproximacion de la solucion:
 u=sum_{j=1}^n (c_j*sin(j*pi*t))

 sin(k*pi*x), k=1,2,3,..son las funciones propias
 del problema de valores propios
 y''+\lambda y=0, x en (0,1), y(0)=0,y(1)=0.
%
```

% ejecutamos el fichero, "el programa", para distintos valores de n

>> seriefourier(1)

f =

t

aproximacion =

.6366*sin(3.142*t)

% vemos que el primer término de la serie nos da una aproximación muy mala de f=x; las aproximaciones, con dos o tres términos de la serie, siguen siendo malas como se observa en la gráfica obtenida con las instrucciones "hold on" y "gtext(' comentario')"

>> hold on

>> seriefourier(2)

f =

t

aproximacion =

.6366*sin(3.142*t)-.3183*sin(6.284*t)

>> seriefourier(3)

f =

t

aproximacion =

.6366*sin(3.142*t)-.3183*sin(6.284*t)+.2122*sin(9.426*t)

>> seriefourier(4)

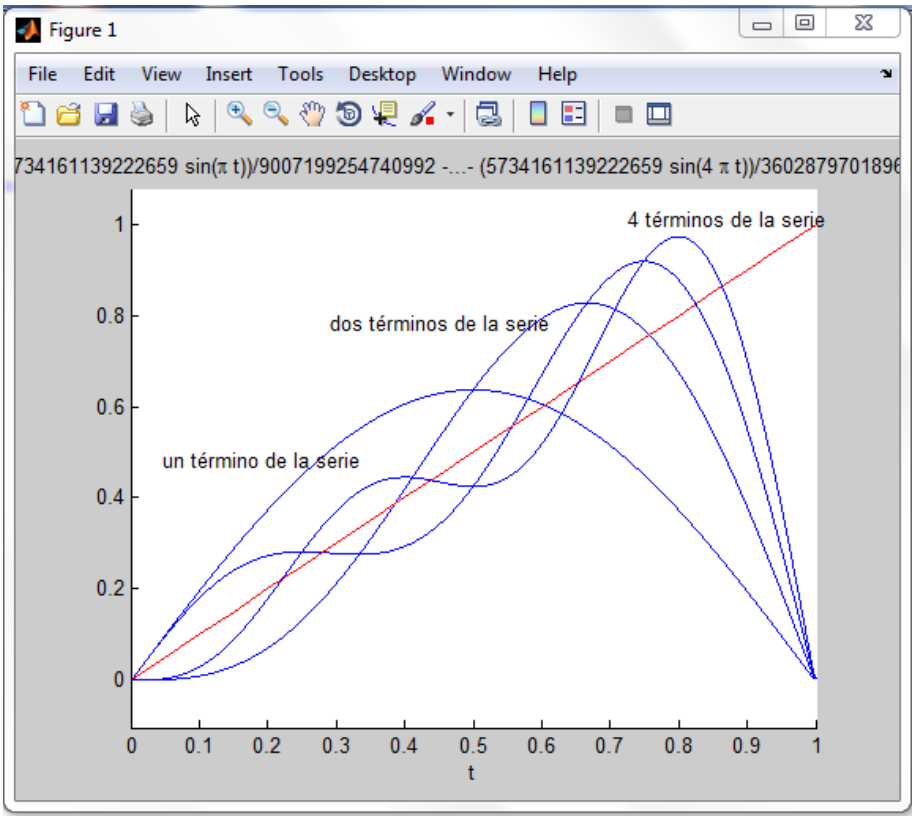
f =

t

aproximacion =

.6366*sin(3.142*t)-.3183*sin(6.284*t)+.2122*sin(9.426*t)-.1592*sin(12.57*t)

>> hold off



>> seriefourier(10)

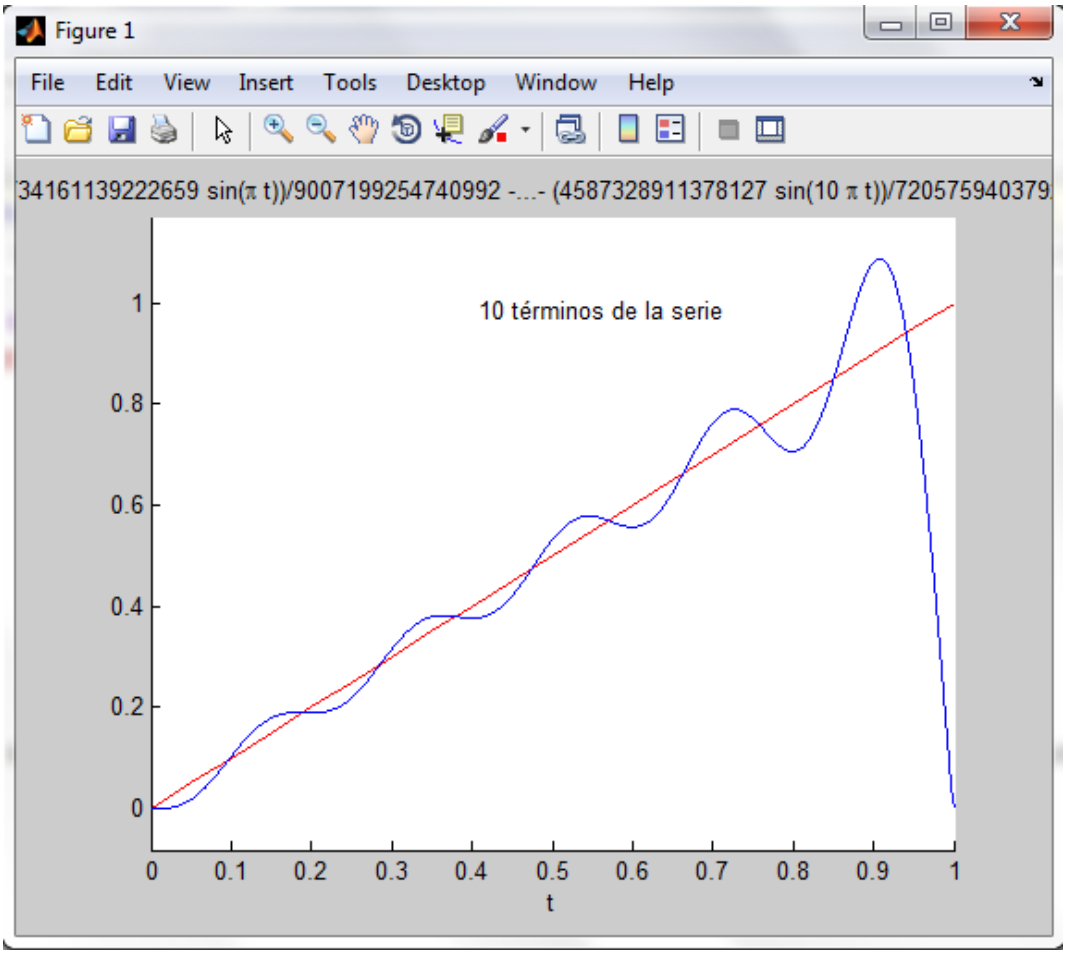
f =

t

aproximacion =

.6366*sin(3.142*t)-.3183*sin(6.284*t)+.2122*sin(9.426*t)-
.1592*sin(12.57*t)+.1273*sin(15.71*t)-.1061*sin(18.85*t)+.9095e-1*sin(21.99*t)-
.7958e-1*sin(25.14*t)+.7074e-1*sin(28.28*t)-.6366e-1*sin(31.42*t)

% con 10 términos vamos teniendo una mejor aproximación, al menos, en términos de áreas por debajo de las gráficas. Necesitamos añadir más términos en el desarrollo en serie para observar esto mejor



>> hold off

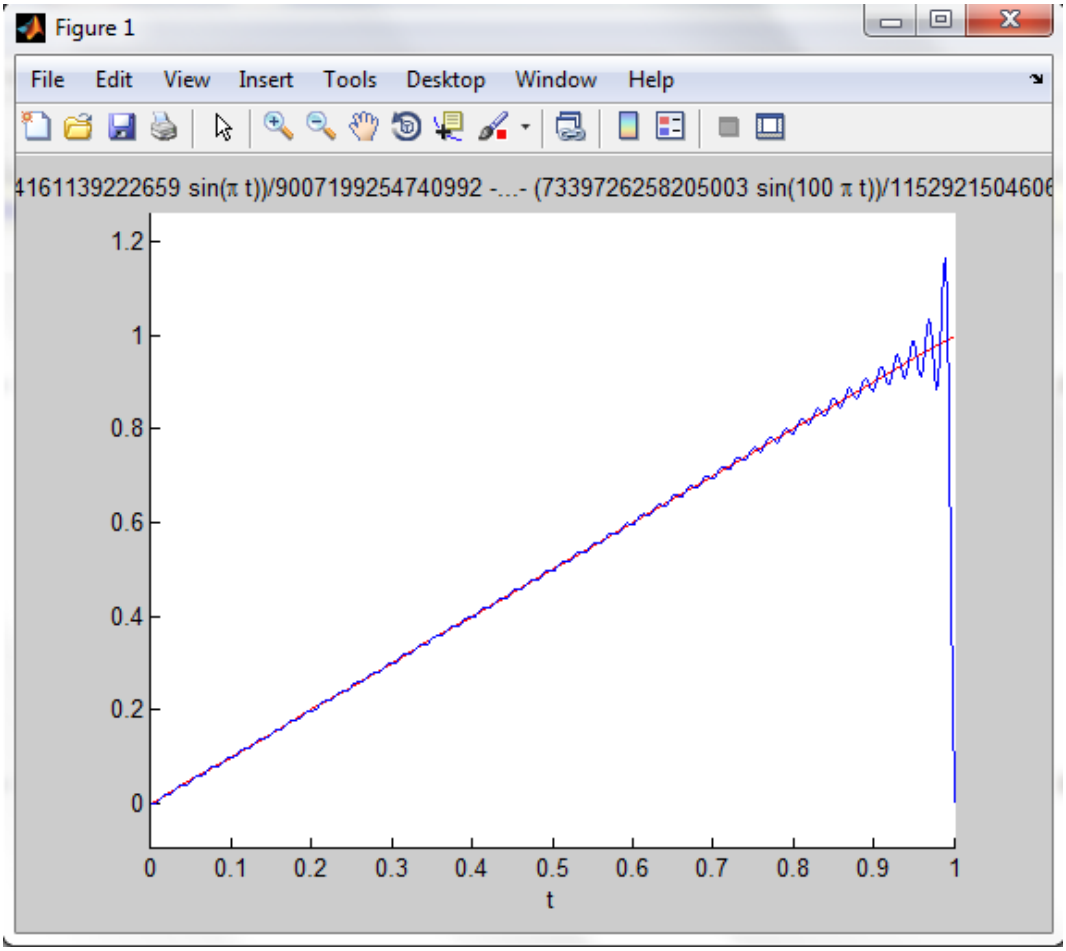
>> seriefourier(100)

f =

t

aproximacion =

-.1061e-1*sin(188.5*t)-.7958e-1*sin(25.14*t)+.6366*sin(3.142*t)-.7764e-2*sin(257.6*t)+.7670e-2*sin(260.8*t)-.5305e-1*sin(37.70*t)+.6563e-2*sin(304.8*t)+.8058e-2*sin(248.2*t)+.4897e-1*sin(40.85*t)-.6920e-2*sin(289.1*t)+.6431e-2*sin(311.1*t)-.7958e-2*sin(251.4*t)+.8721e-2*sin(229.4*t)-.2449e-1*sin(81.69*t)-.7403e-2*sin(270.2*t)+.7317e-2*sin(273.4*t)-.8162e-2*sin(245.1*t)+.2358e-1*sin(84.83*t)-.9947e-2*sin(201.1*t)-.1273e-1*sin(157.1*t)-.9362e-2*sin(213.7*t)+.1157e-1*sin(172.8*t)+.2546e-1*sin(78.55*t)+.8966e-2*sin(223.1*t)-.6773e-2*sin(295.3*t)+.1481e-1*sin(135.1*t)+.1079e-1*sin(185.4*t)-.8603e-2*sin(232.5*t)-.1989e-1*sin(100.5*t)-.1137e-1*sin(176.0*t)+.7153e-2*sin(279.6*t)-.1061*sin(18.85*t)+.2195e-1*sin(91.12*t)-.1592e-1*sin(125.7*t)+.9502e-2*sin(210.5*t)-.1592*sin(12.57*t)-.1027e-1*sin(194.8*t)+.1632e-1*sin(122.5*t)+.1201e-1*sin(166.5*t)+.2768e-1*sin(72.27*t)+.1248e-1*sin(160.2*t)+.1117e-1*sin(179.1*t)-.1872e-1*sin(106.8*t)-.3537e-1*sin(56.56*t)+.7860e-2*sin(254.5*t)+.9095e-1*sin(21.99*t)-.3183e-1*sin(62.84*t)+.5787e-1*sin(34.56*t)+.4244e-1*sin(47.13*t)+.8488e-2*sin(235.6*t)-.9646e-2*sin(207.4*t)+.1415e-1*sin(141.4*t)+.8268e-2*sin(241.9*t)-.3979e-1*sin(50.27*t)-.3183*sin(6.284*t)-.1326e-1*sin(150.8*t)+.6845e-2*sin(292.2*t)+.1355e-1*sin(147.7*t)-.7234e-2*sin(276.5*t)-.2122e-1*sin(94.26*t)+.3032e-1*sin(65.98*t)-.2274e-1*sin(87.98*t)+.3351e-1*sin(59.70*t)+.1011e-1*sin(197.9*t)+.1819e-1*sin(110.0*t)+.6701e-2*sin(298.5*t)-.8842e-2*sin(226.2*t)+.1721e-1*sin(116.3*t)-.6631e-2*sin(301.6*t)+.2122*sin(9.426*t)-.6366e-1*sin(31.42*t)+.1273*sin(15.71*t)+.2054e-1*sin(97.40*t)-.2653e-1*sin(75.41*t)-.1098e-1*sin(182.2*t)+.3745e-1*sin(53.41*t)-.1675e-1*sin(119.4*t)-.7579e-2*sin(263.9*t)-.1224e-1*sin(163.4*t)+.1299e-1*sin(154.0*t)-.4547e-1*sin(43.99*t)+.1044e-1*sin(191.7*t)+.7074e-1*sin(28.28*t)-.1447e-1*sin(138.2*t)-.1768e-1*sin(113.1*t)+.1929e-1*sin(103.7*t)-.9095e-2*sin(219.9*t)-.2894e-1*sin(69.12*t)-.1384e-1*sin(144.5*t)-.1179e-1*sin(169.7*t)+.9226e-2*sin(216.8*t)-.1516e-1*sin(132.0*t)+.1553e-1*sin(128.8*t)+.9794e-2*sin(204.2*t)+.6996e-2*sin(285.9*t)-.6496e-2*sin(307.9*t)+.7490e-2*sin(267.1*t)-.8377e-2*sin(238.8*t)-.6366e-2*sin(314.2*t)-.7074e-2*sin(282.8*t)

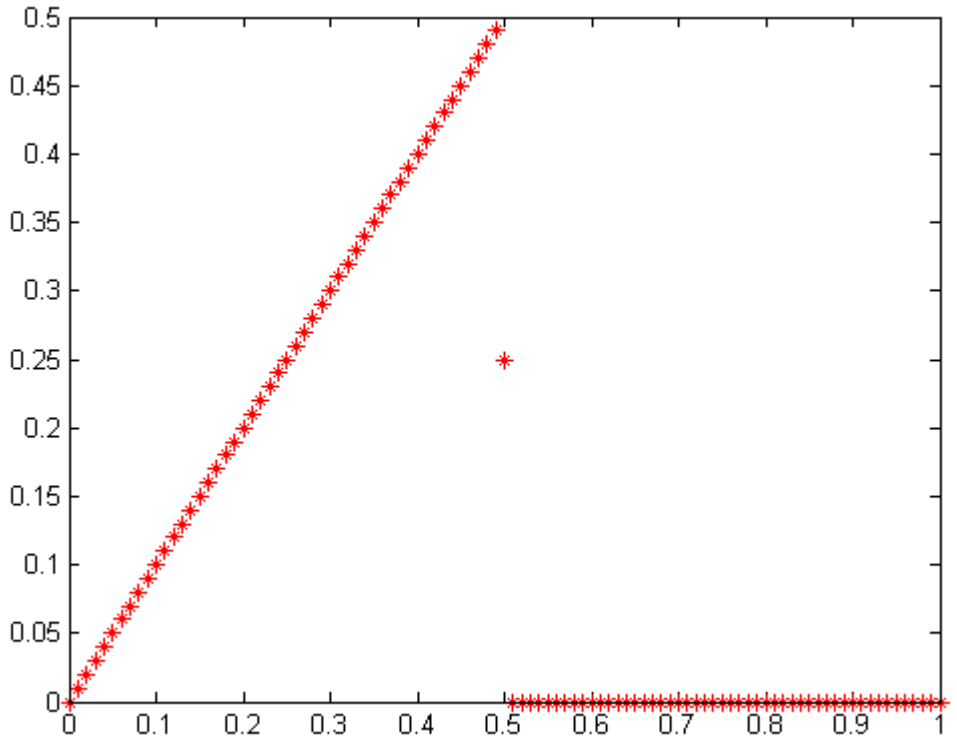


% vemos que la aproximación de los 100 primeros términos es una buena aproximación en el sentido de áreas por debajo de las gráficas (la gráfica de la función f(x)=x está en rojo). La aproximación es buena en los puntos de continuidad de f(x), y es muy mala en el extremo x=1, como consecuencia de que los términos de la serie toman el valor 0 en x=1, y, f(1)=1.

% Repetimos el experimento para la función discontinua en x=1/2, pero que verifica las condiciones de contorno de los términos del desarrollo en x=0 y x=1: la función f(x)=x en [0,1/2], f(x)=0 en (1/2,0]. La función f(x) está definida dentro de la función Matlab seriefourier_disc.m como

```
>> syms t

>> f=t*(heaviside(t)-heaviside(t-1/2));

% y su gráfica, en rojo, con el conjunto de comandos

>> dt=0:0.01:1;

>> df=subs(f,t,dt);
```

>> plot(dt,df,'*r')



% observamos que en t=1/2, nuestra función vale 1/2, no 1/4 como aparece en la gráfica (punto intermedio marcado en rojo); esto es debido a la definición de la función de heaviside por Matlab, que varía dependiendo de la versión, como se ve con help en las versiones 2011 y 2008 de Matlab

>> help heaviside *% versión 2008 de Matlab*

```
heaviside    Step function.
    heaviside(X) is 0 for X < 0, 1 for X > 0, and .5 for X == 0.
    heaviside(X) is not a function in the strict sense.
```

>> help heaviside *% versión 2011 de Matlab*

```
HEAVISIDE    Step function.
    HEAVISIDE(X) is 0 for X < 0, 1 for X > 0, and NaN for X == 0.
    HEAVISIDE(X) is not a function in the strict sense.
```

% ejecutando el fichero seriefourier_disc, para 10 términos en el desarrollo en serie, se obtiene

>> type seriefourier_disc

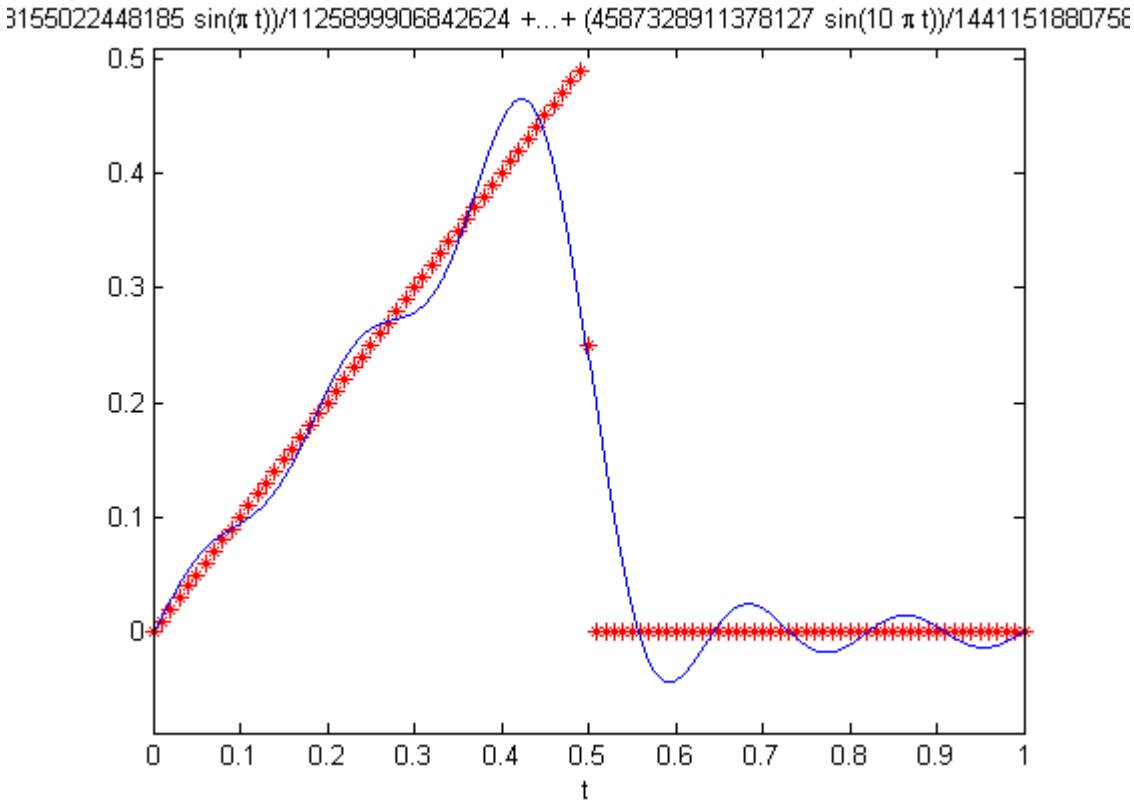
>> seriefourier_disc(10)

f =

-t*(heaviside(t - 1/2) - heaviside(t))

aproximacion =

0.05305*sin(18.85*t) - 0.02252*sin(9.425*t) - 0.004136*sin(21.99*t) + 0.2026*sin(3.142*t) + 0.1592*sin(6.283*t) - 0.07958*sin(12.57*t) - 0.03979*sin(25.13*t) + 0.002502*sin(28.27*t) + 0.008106*sin(15.71*t) + 0.03183*sin(31.42*t)



% vemos la aproximación en términos de áreas por debajo de la curva, mientras que, y si volvemos a ejecutar con 100 términos, observamos la mejora de esta "convergencia" así como la convergencia puntual en [0,1/2), y en (1/2,0], mientras que en x=1/2 el valor de la serie es el valor medio de la función (f(1/2+)+f(1/2-))/2=1/4 (que no coincide con f(1/2)=1/2)

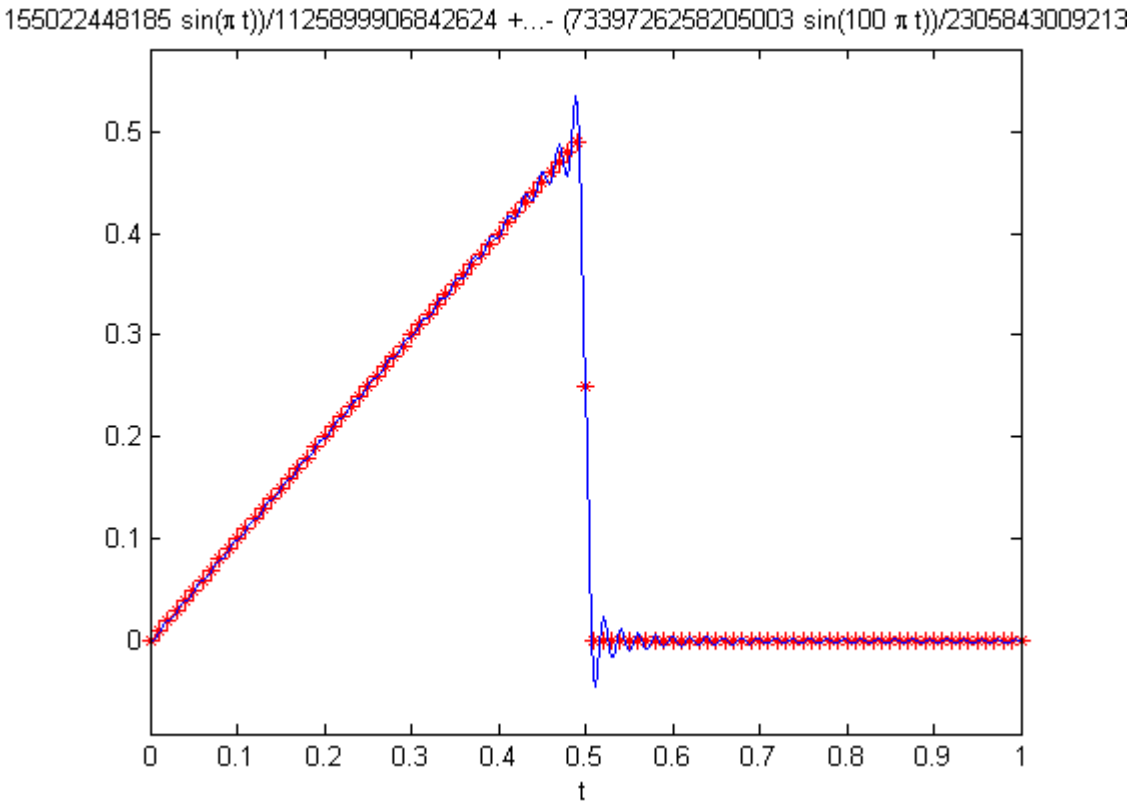
>> seriefourier_disc(100)

f =

-t*(heaviside(t - 1/2) - heaviside(t))

aproximacion =

0.0001205*sin(128.8*t) + 0.003882*sin(257.6*t) - 2.942e-5*sin(260.8*t) + 0.0004595*sin(65.97*t) + 0.007579*sin(131.9*t) - 0.003789*sin(263.9*t) + 2.805e-5*sin(267.0*t) - 0.0001096*sin(135.1*t) + 0.003701*sin(270.2*t) - 2.677e-5*sin(273.3*t) - 0.001675*sin(34.56*t) + 0.01447*sin(69.12*t) - 0.007234*sin(138.2*t) - 0.003617*sin(276.5*t) + 2.558e-5*sin(279.6*t) + 0.0001001*sin(141.4*t) + 0.003537*sin(282.7*t) - 2.447e-5*sin(285.9*t) - 0.0003831*sin(72.26*t) + 0.00692*sin(144.5*t) - 0.00346*sin(289.0*t) + 2.343e-5*sin(292.2*t) - 9.173e-5*sin(147.7*t) + 0.003386*sin(295.3*t) - 2.245e-5*sin(298.5*t) - 0.02252*sin(9.425*t) + 0.05305*sin(18.85*t) - 0.02653*sin(37.7*t) - 0.01326*sin(75.4*t) - 0.006631*sin(150.8*t) - 0.003316*sin(301.6*t) + 2.154e-5*sin(304.7*t) + 8.44e-5*sin(153.9*t) + 0.003248*sin(307.9*t) - 2.068e-5*sin(311.0*t) + 0.0003242*sin(78.54*t) + 0.006366*sin(157.1*t) - 0.003183*sin(314.2*t) - 7.791e-5*sin(160.2*t) + 0.001199*sin(40.84*t) + 0.01224*sin(81.68*t) - 0.006121*sin(163.4*t) + 7.214e-5*sin(166.5*t) - 0.000278*sin(84.82*t) + 0.005895*sin(169.6*t) - 6.699e-5*sin(172.8*t) - 0.004136*sin(21.99*t) + 0.02274*sin(43.98*t) - 0.01137*sin(87.96*t) - 0.005684*sin(175.9*t) + 6.237e-5*sin(179.1*t) + 0.000241*sin(91.11*t) + 0.005488*sin(182.2*t) - 5.821e-5*sin(185.4*t) - 0.0009006*sin(47.12*t) + 0.01061*sin(94.25*t) - 0.005305*sin(188.5*t) + 5.446e-5*sin(191.6*t) - 0.0002109*sin(97.39*t) + 0.005134*sin(194.8*t) - 5.106e-5*sin(197.9*t) + 0.2026*sin(3.142*t) + 0.1592*sin(6.283*t) - 0.07958*sin(12.57*t) - 0.03979*sin(25.13*t) - 0.01989*sin(50.27*t) - 0.009947*sin(100.5*t) - 0.004974*sin(201.1*t) + 4.796e-5*sin(204.2*t) + 0.0001861*sin(103.7*t) + 0.004823*sin(207.3*t) - 4.514e-5*sin(210.5*t) + 0.0007012*sin(53.41*t) + 0.009362*sin(106.8*t) - 0.004681*sin(213.6*t) + 4.256e-5*sin(216.8*t) - 0.0001654*sin(110.0*t) + 0.004547*sin(219.9*t) - 4.02e-5*sin(223.1*t) + 0.002502*sin(28.27*t) + 0.01768*sin(56.55*t) - 0.008842*sin(113.1*t) - 0.004421*sin(226.2*t) + 3.803e-5*sin(229.3*t) + 0.000148*sin(116.2*t) + 0.004301*sin(232.5*t) - 3.603e-5*sin(235.6*t) - 0.0005613*sin(59.69*t) + 0.008377*sin(119.4*t) - 0.004188*sin(238.8*t) + 3.418e-5*sin(241.9*t) - 0.0001332*sin(122.5*t) + 0.004081*sin(245.0*t) - 3.247e-5*sin(248.2*t) + 0.008106*sin(15.71*t) + 0.03183*sin(31.42*t) - 0.01592*sin(62.83*t) - 0.007958*sin(125.7*t) - 0.003979*sin(251.3*t) + 3.089e-5*sin(254.5*t)



% En el caso en que la función f(x) a aproximar sea de clase C^2 en [0,1] y verifique las condiciones de contorno del problema de valores propios, esto es, de los términos de la serie, la aproximación mejora considerablemente (se trata de una convergencia uniforme), y los tres o cuatro primeros términos del desarrollo nos dan ya una muy buena aproximación de la solución como puede observarse para la función f(x)=x(x-1) introducida en la función Matlab seriefouriercc.m

```
>> syms t

>> f=t*(t-1);

% hacemos la gráfica en rojo

>> dt=0:0.01:1;

>> df=subs(f,t,dt);

>> plot(dt,df,'*r')
```

>> type seriefouriercc

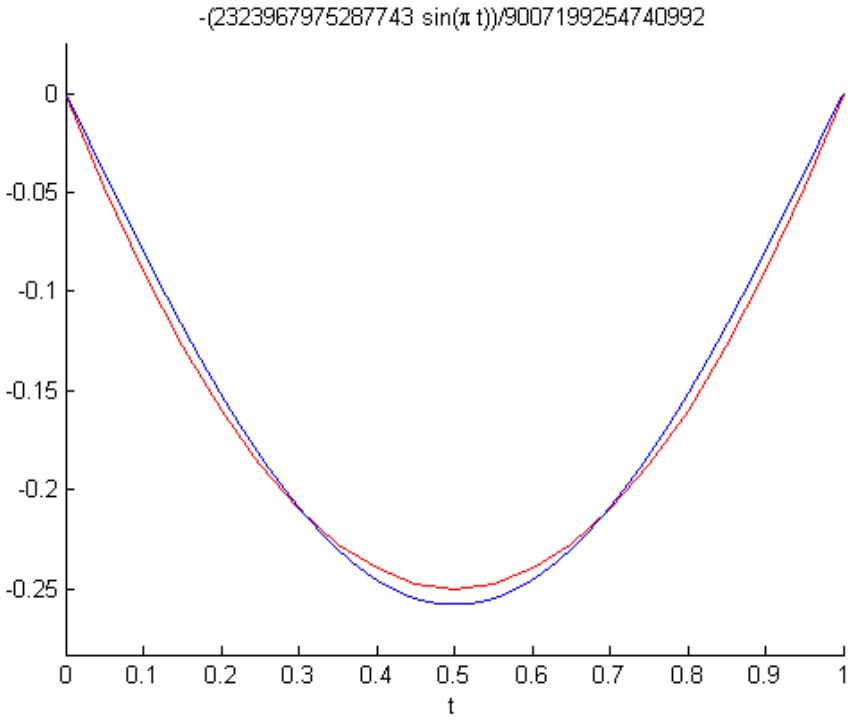
>> seriefouriercc(1);

f =

t*(t - 1)

aproximacion =

-0.258*sin(3.142*t)



% la primera aproximación en azul, mediante un solo término del desarrollo es ya bastante buena; abajo vemos que es mejor la aproximación por tres términos, que aparentemente coincide con la solución (¡siempre se comete un error!)

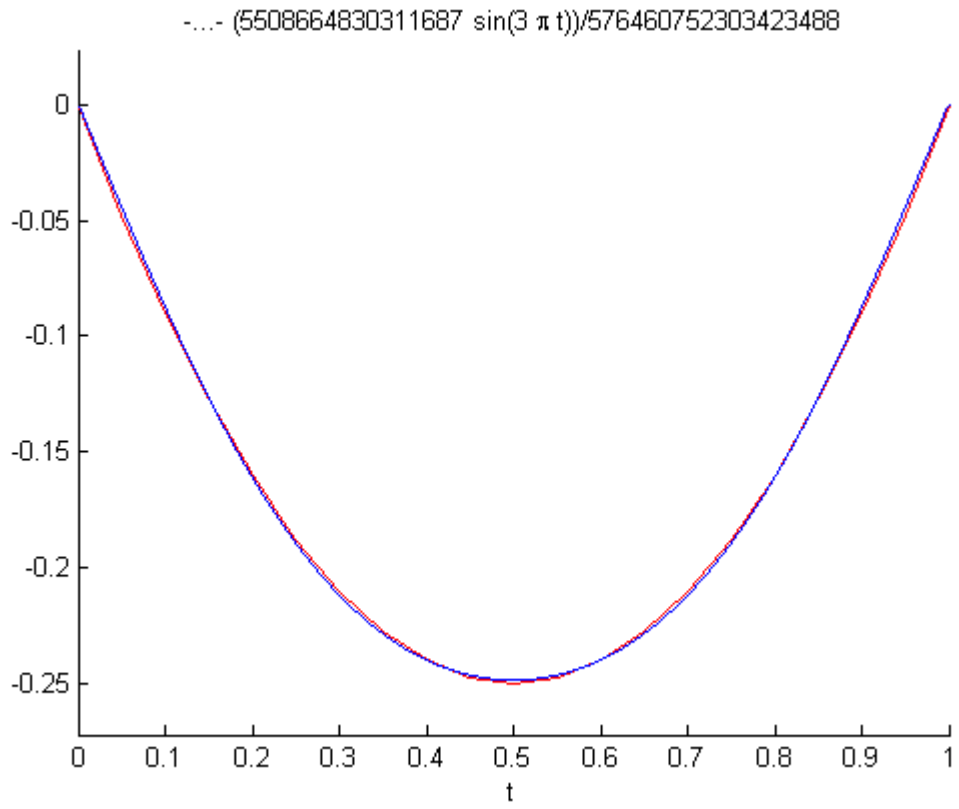
>> seriefouriercc(3)

f =

t*(t - 1)

aproximacion =

- 0.009556*sin(9.425*t) - 0.258*sin(3.142*t)



*% **Resolución explícita de ED de segundo orden:** recordamos que y'' es D2y en Matlab (y' es Dy respectivamente)*

>> dsolve('D2y-y=t^2')

ans =

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

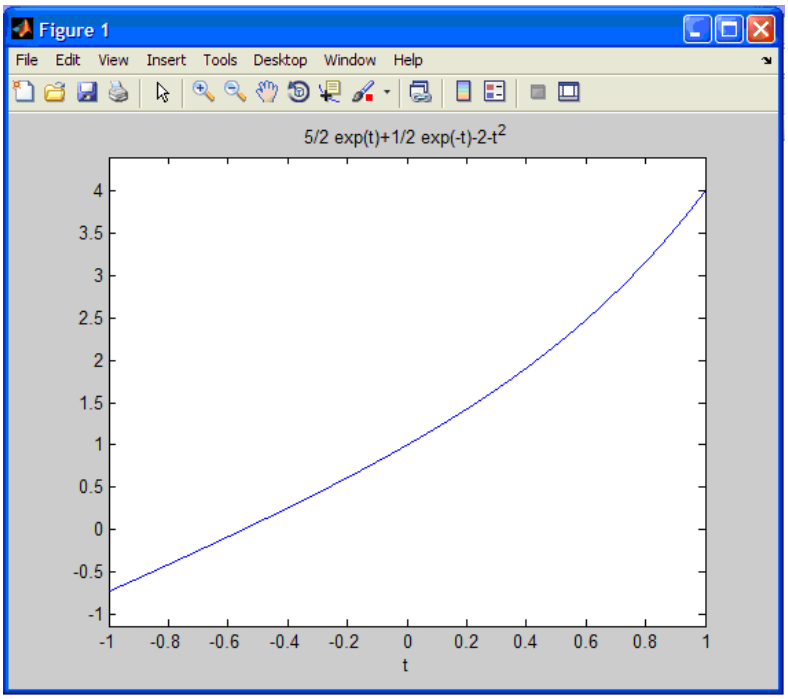
% C1 y C2 son las constantes de integración, y se determinan imponiendo condiciones iniciales o de contorno

>> dsolve('D2y-y=t^2','y(0)=1','Dy(0)=2'); *% ejemplo de Problema de Cauchy con solución definida en (-infinito,infinito)*

ans =

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

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



>> dsolve('D2y-y=t^2','y(0)=0','y(1)=0') *% ejemplo de problema de contorno con solución definida en (0,1)*

ans =

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

% pretty, simple, simplify y vpa ayudan a visualizar mejor

>> pretty(ans)

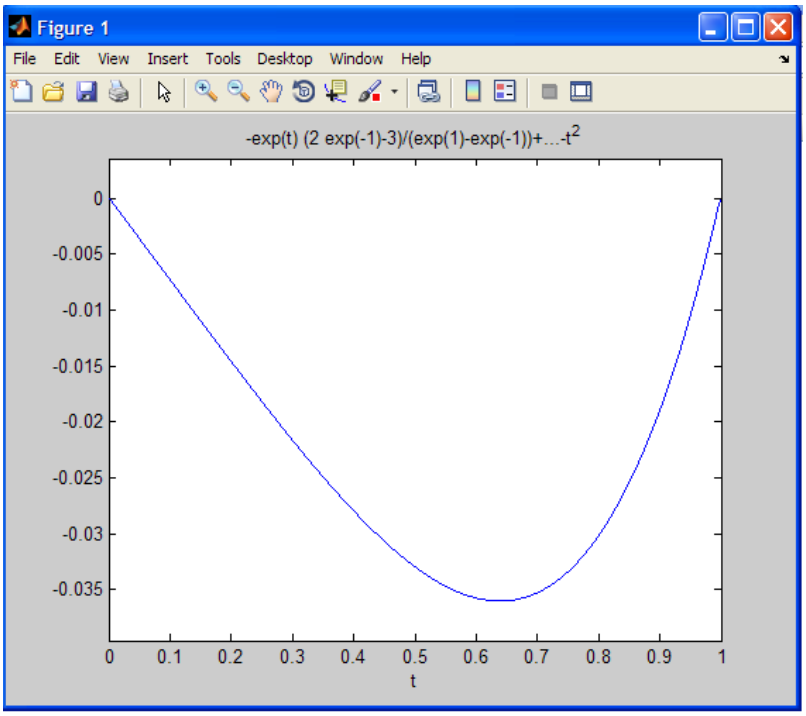
```
  exp(t) (-3 + 2 exp(-1))   exp(-t) (2 exp(1) - 3)         2
- ----------------------- + ----------------------- - 2 - t
     exp(1) - exp(-1)          exp(1) - exp(-1)
```

>> vpa(ans,5)

ans =

.96333*exp(t)+1.0367*exp(-1.*t)-2.-1.*t^2

>> ezplot(ans,[0,1]) *% se dibuja en el intervalo donde está definida la solución*



% Ejemplo de problema de contorno con solución única que Matlab no resuelve explícitamente: y''-t^3y=exp(-t^2), y(0)=0, y(1)=0

>> dsolve('D2y-t^3*y=0','y(0)=0','y(1)=0') *% tiene solo la solución trivial*

ans =

0

>> dsolve('D2y-t^3*y=0')

ans =

t^(1/2)*(C1*besseli(1/5,2/5*t^(5/2))+C2*besselk(1/5,2/5*t^(5/2)))

>> dsolve('D2y-t^3*y=exp(-t^2)')

ans =

1/5*t^(1/2)*(5*C2*besseli(1/5,2/5*t^(5/2))+5*besselk(1/5,2/5*t^(5/2))*C1+2*Int(t^(1/2)*besselk(1/5,2/5*t^(5/2))*exp(-t^2),t)*besseli(1/5,2/5*t^(5/2))-2*Int(t^(1/2)*besseli(1/5,2/5*t^(5/2))*exp(-t^2),t)*besselk(1/5,2/5*t^(5/2)))

% vemos que integra la ecuación homogénea, pero no la no homogénea, y por tanto no resuelve el problema de contorno:

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

Warning: Explicit solution could not be found.
> In dsolve at 333

ans =

[empty sym]

% sin embargo, como dicho problema tiene solución única, se puede aproximar la solución numéricamente utilizando, por ejemplo, las funciones Matlab galerkin.m y/o elementosfinitos.m

Segunda parte: aproximaciones numéricas

% Las funciones Matlab galerkin.m y elementosfinitos.m convenientemente adaptadas permiten aproximar soluciones de problemas de contorno usando como funciones de base las funciones propias de un problema de valores propios (es "una base de funciones"), y las de los elementos finitos (funciones lineales a trozos asociadas a una partición del intervalo donde se define el problema), respectivamente. Leen n, el número de elementos de la base que se quiere utilizar para la aproximación.

% Concretamente, ejecutando "galerkin(n)", se calcula la aproximación de la solución del problema de contorno
y''- y=x^2, x en (0,1), y(0)=0, y(1)=0,
por el método de Galerkin, utilizando como funciones de base las funciones propias del problema de valores propios
y''+lambda y=0, x en (0,1), y(0)=0,y(1)=0.

*Dichas funciones propias son sin(k*pi*x), k=1,2,.. asociadas a los valores propios lambda=k^2pi^2. Lee n el número de elementos de la base y nos devuelva el vector c que nos permite calcular la aproximación de la solución*
*(u=) y_n(t)=sum_{j=1}^n (c_j*sin(j*pi*t))*
Dado que para este problema se conoce la solución explícita, la función galerkin.m compara la aproximación u con la solución.

% En el proceso de cálculo hay que hay que resolver un sistema: Ac=f, donde A es la matriz de coeficientes, f es el término independiente y c es la solución del sistema. El siguiente conjunto de instrucciones permite definir A y f y resolver el sistema.

```
syms t %variable simbólica
A=ones(n,n); % inicializa  matriz y  vector
f=ones(n,1);
for i=1:n
  f(i)=-double(int(sin(i*pi*t)*t^2,0,1));
  for j=1:n
    if i==j
      A(i,j)=double(int((i*pi*cos(i*pi*t))^2,0,1)+int((sin(pi*i*t))^2,0,1));
        else
      A(i,j)=double(int(i*pi*cos(i*pi*t)*j*pi*cos(j*pi*t),0,1)+
int(sin(i*pi*t)*sin(j*pi*t),0,1));
     end
    end
  end
%
matriz_sistema=A
termino_independiente=f
%
c=A\f; % resolviendo el sistema
coeficientes_solucion=c
```

% El siguiente conjunto de instrucciones define la aproximación de la solución y nos proporciona su gráfica en línea discontinua

```
u=0;
for i=1:n
  u=u+c(i)*sin(pi*i*t);
end
%
solu_aproximada=vpa(u,4)% una forma de visualizar la aproximación
%
dt=0:0.05:1;
du=subs(u,t,dt);
ddu=double(du);
plot(dt,ddu,'--') % se podría utilizar ezplot
```

```
% Si no se sabe resolver la ecuación diferencial el programa acabaría aquí. En este
  ejemplo concreto, se calcula la solución del problema de contorno y se compara la
  gráfica de la solución con la de u obtenida arriba
```

```
v=dsolve('D2y-y=t^2','y(0)=0','y(1)=0');
solu_exacta=v
  %
hold on
ezplot(v,[0,1])
hold off
```

```
% Por la misma razón se pueden calcular explícitamente los coeficientes de Fourier de
  la solución v, alpha(i), y compararlos con los coeficientes c(i) que nos da la resolución
  del sistema. Se comprueba que se aproximan (para N grande).
```

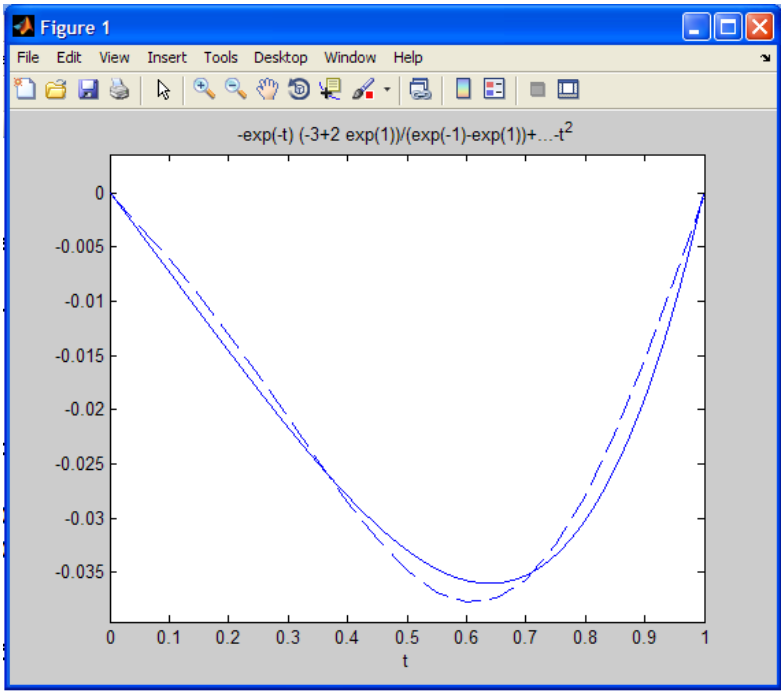
```
alpha=ones(n,1);
for i=1:n
   alpha(i)=double(int(sin(i*pi*t)*v,0,1)/int((sin(pi*i*t))^2,0,1));
end
coeficientes_fourier=alpha
%
errorfourier_galerkin=alpha-c
```

```
% En este ejemplo concreto, se cometen pocos errores de cálculo dado que, por las
  características del problema y base utilizada (ED de coeficientes constantes,
  ortogonalidad de las funciones propias y de sus derivadas,…), se tiene que A es una
  matriz diagonal y se puede demostrar que los coeficientes c(i) coinciden con los
  alpha(i).
```

```
% Se ejecuta el fichero para distintos n viendo que la aproximación de la solución es
  muy buena con pocos elementos de la base, se comparan gráficas y soluciones
```

>> galerkin(2)

```
matriz_sistema =

   5.4348     0
        0   20.2392


termino_independiente =

   -0.1893
    0.1592


coeficientes_solucion =

   -0.0348
    0.0079


solu_aproximada =

-.3483e-1*sin(3.142*t)+.7864e-2*sin(6.284*t)



solu_exacta =

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


coeficientes_fourier =

   -0.0348
    0.0079


errorfourier_galerkin =

  1.0e-015 *

   -0.1874
    0.1006
```



>> galerkin(4)

matriz_sistema =

```
   5.4348    0         0         0
        0   20.2392    0         0
        0         0   44.9132    0
        0         0         0   79.4568
```

termino_independiente =

```
  -0.1893
   0.1592
  -0.1013
   0.0796
```

coeficientes_solucion =

```
  -0.0348
   0.0079
  -0.0023
   0.0010
```

solu_aproximada =

-.3483e-1*sin(3.142*t)+.7864e-2*sin(6.284*t)-.2256e-2*sin(9.426*t)+.1002e-2*sin(12.57*t)

solu_exacta =

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

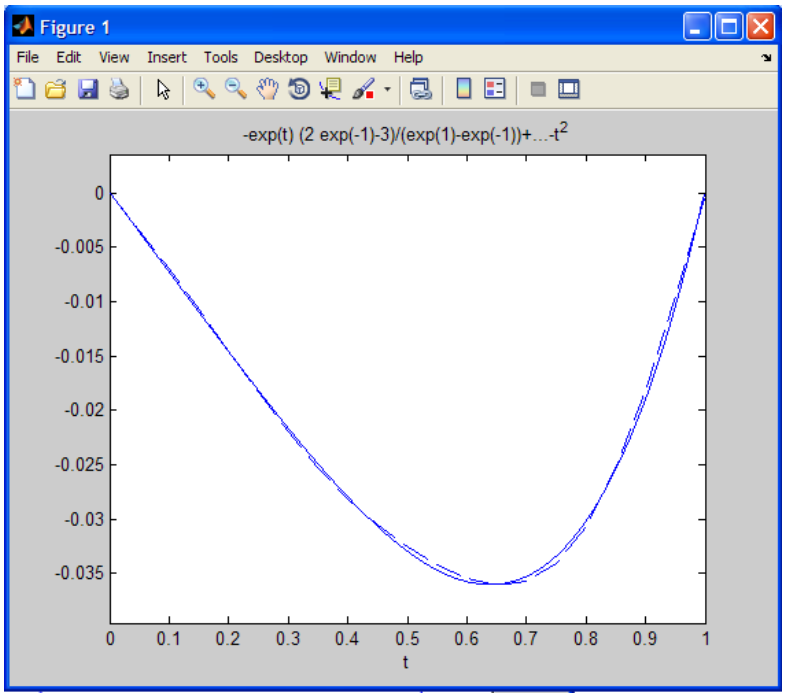
coeficientes_fourier =

-0.0348
0.0079
-0.0023
0.0010

errorfourier_galerkin =

1.0e-015 *

-0.1874
0.1006
-0.0681
0.0510

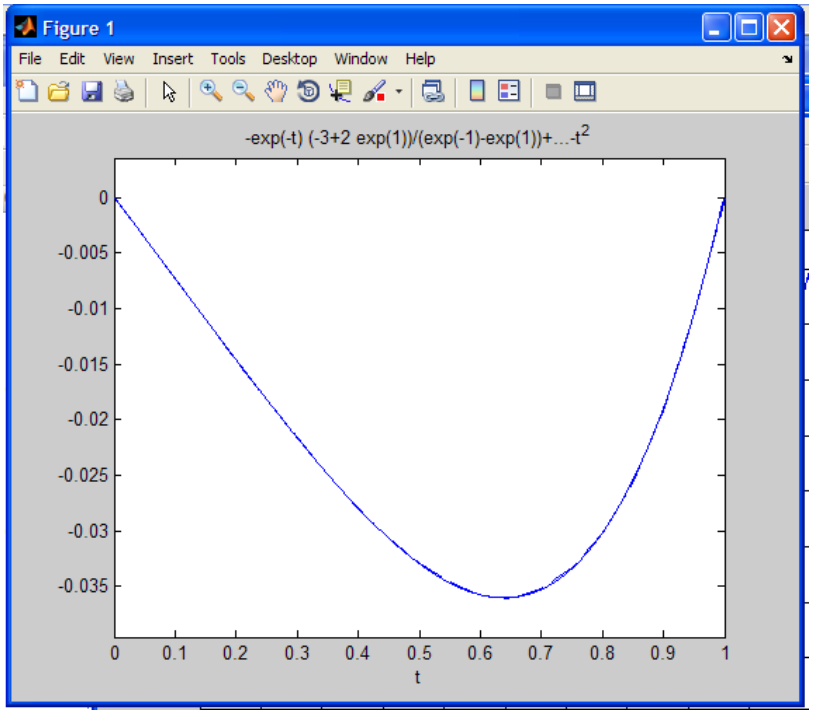


% Buenas aproximaciones por las características del problema: los coeficientes de Fourier coinciden con la solución del sistema

>> galerkin(8)

```
matriz_sistema =

  Columns 1 through 6

    5.4348    0         0         0         0         0
         0   20.2392    0         0         0         0
         0         0   44.9132    0         0         0
         0         0         0   79.4568    0         0
         0         0         0         0  123.8701    0
         0         0         0         0         0  178.1529
         0         0         0         0         0         0
         0         0         0         0         0         0

  Columns 7 through 8

         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
  242.3053    0
         0  316.3273


termino_independiente =

   -0.1893
    0.1592
   -0.1013
    0.0796
   -0.0626
    0.0531
   -0.0451
    0.0398
```

```
coeficientes_solucion =

   -0.0348
    0.0079
   -0.0023
    0.0010
   -0.0005
    0.0003
   -0.0002
    0.0001

solu_aproximada =

-.3483e-1*sin(3.142*t)+.7864e-2*sin(6.284*t)-.2256e-2*sin(9.426*t)+.1002e-
2*sin(12.57*t)-.5056e-3*sin(15.71*t)+.2978e-3*sin(18.85*t)-.1861e-
3*sin(21.99*t)+.1258e-3*sin(25.14*t)

solu_exacta =

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

coeficientes_fourier =

   -0.0348
    0.0079
   -0.0023
    0.0010
   -0.0005
    0.0003
   -0.0002
    0.0001


errorfourier_galerkin =

  1.0e-015 *

   -0.1874
    0.1006
   -0.0681
    0.0510
   -0.0411
    0.0343
   -0.0294
    0.0257
```



>> type galerkin

% Se modifica el programa para otro problema en que no se conoce la solución explícita, aunque se sabe que existe y es única:
y''- x^2y = exp(x), x en (0,1), y(0)=0, y(1)=0.
Se observa que la matriz A no es diagonal, complicándose los cálculos. Se puede establecer un test de parada comparando gráficas.

>> 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]

% el problema homogéneo asociado tiene sólo la solución trivial

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

ans =

0

% Una simple modificación dentro de la función Matlab galerkin.m de las integrales que aparecen definiendo f y A

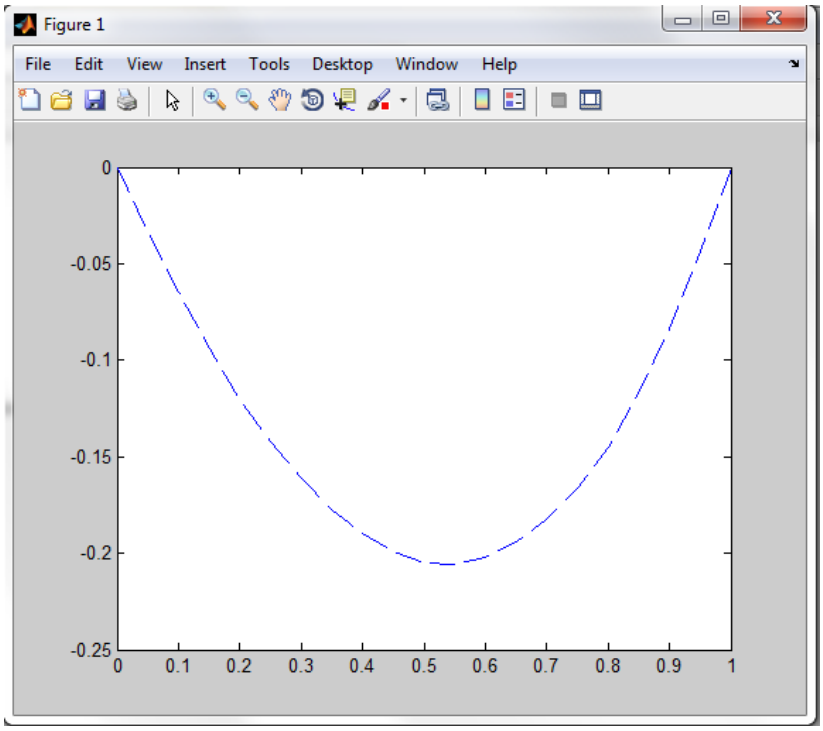
```
f(i)=-double(int(sin(i*pi*t)*exp(t),0,1));
%
A(i,j)=double(int((i*pi*cos(i*pi*t))^2,0,1)+int((sin(pi*i*t))^2*t^2,0,1));
%
A(i,j)=double(int(i*pi*cos(i*pi*t)*j*pi*cos(j*pi*t),0,1)+int(sin(i*pi*t)*sin(j*pi*t)*t^
2,0,1));
```

% Así, ejecutando el fichero para n=6, nos devuelve la matriz del sistema simétrica pero con todos los términos no nulos

matriz_sistema =

5.0761	-0.0901	0.0190	-0.0072	0.0035	-0.0020
-0.0901	19.8995	-0.0973	0.0225	-0.0092	0.0047
0.0190	-0.0973	44.5771	-0.0993	0.0237	-0.0100
-0.0072	0.0225	-0.0993	79.1219	-0.1001	0.0243
0.0035	-0.0092	0.0237	-0.1001	123.5357	-0.1005
-0.0020	0.0047	-0.0100	0.0243	-0.1005	177.8188

% y la gráfica de la solución aproximada (abajo) que ya no varía mucho para n más grandes



% Comentarios análogos a los de la función galerkin.m se tienen con elementosfinitos.m: ejecutando "elementosfinitos(n)", la función nos proporciona una aproximación de la solución del problema de contorno
y''- y=x^2, x en (0,1), y(0)=0, y(1)=0,
utilizando las funciones de base lineales a trozos, asociadas a los nodos internos de una partición {x_i}_{i=0}^{n+1} del intervalo [0,1], y compara dicha aproximación con la solución explícita. Esto es, se toman las funciones de base phi_i(x) que son polinomios de grado 1 en los sub-intervalos [x_i,x_{i+1}], continuas en [0,1], se anulan en x=0 y x=1, y toman el valor 1 en el nodo x_i y 0 en el resto de los nodos (''hat functions'').

*% La función elementosfinitos.m lee n, el número de nodos del intervalo (0,1); calcula la matriz del sistema A y el término independiente f, y nos devuelve un vector c: c(i) es la solución aproximada en el nodo x_i=i*h, i=1,2,…,n.*

% El siguiente conjunto de instrucciones define los nodos y las funciones que intervienen en la ED

```
h=1/(n+1);
x=ones(n+2,1);
for i=1:n+2
  x(i)=(i-1)*h;
end
%
syms t; % variable simbólica
q=-1;  % término que aparece multiplicando a y(x): y''+q(x)y=r(x)
r=t^2; % término independiente: y''+q(x)y=r(x)
```

% el siguiente conjunto de instrucciones define y resuelve el sistema Ac=f (A matriz tridiagonal simétrica nxn, f término independiente, vector c aproximación de la solución en los nodos internos) y se dibuja la solución aproximada en línea discontinua

```
R=ones(n,1);% inicializando vectores
S=ones(n,1);
Q=ones(n,1);
f=ones(n,1);
a1=ones(n,1);
amenos1=ones((n-1),1);
mas1=ones((n-1),1);
% definiendo las integrales sobre cada segmento [x_i,x_{i+1}]
for i=1:n
  R(i)=double(int(q*(t-x(i+2))^2,x(i+1),x(i+2)));
  Q(i)= double(int(q*(t-x(i))^2,x(i),x(i+1)));
  S(i)=double(int(q*(t-x(i))*(t-x(i+1)),x(i),x(i+1)));
  f(i)=double((1/h)*int(r*(t-x(i)),x(i),x(i+1))+(1/h)*int(r*(-t+x(i+2)),x(i+1),x(i+2)));
end
S1menos=S(2:n);

% la matriz  A se define a través de los  vectores que definen  las diagonales y
subdiagonales
 %
a1= (1/h^2)*(-2*h*ones(n,1)+Q+R); % A(i,i)
%
amenos1=(1/h^2)*(h*ones((n-1),1)-S1menos); % A(i,i-1)
%
amas1=amenos1; % A(i,i+1)
%
A=diag(a1)+diag(amas1,1)+diag(amenos1,-1);
%
matriz_sistema=A
termino_independiente=f
 %
c=A\f; % resolviendo el sistema
%
coeficientes_solucion=c
%
Nodo_y_Aproximacionsolucion=[x,[0;c;0]]
% gráfica de la solución
plot(x,[0;c;0],'--')
```

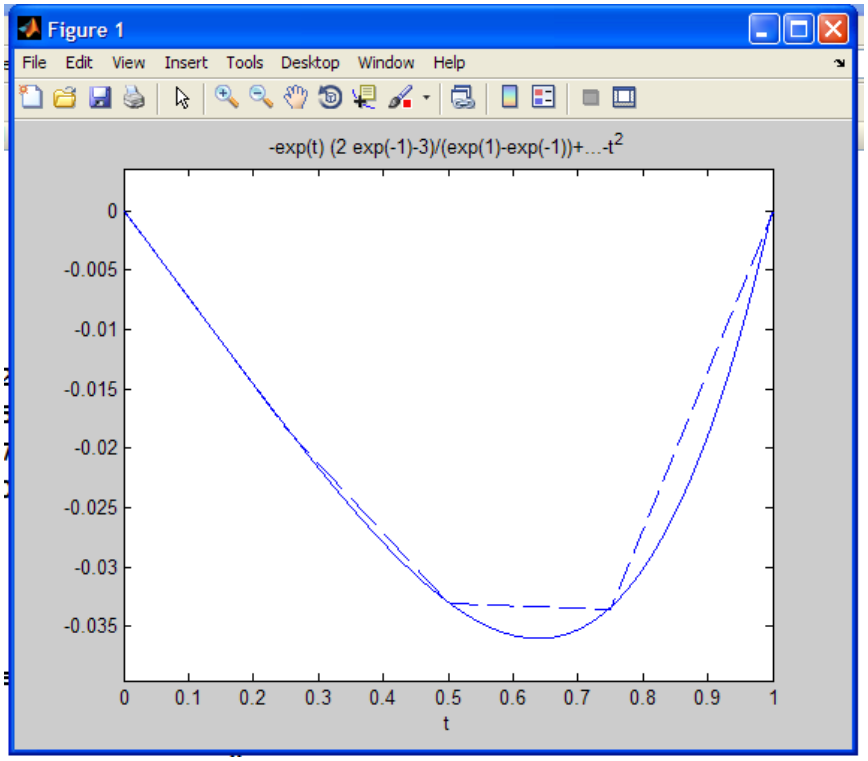
% Si no se sabe resolver la ecuación diferencial, el programa acabaría aquí. En este ejemplo, se calcula la solución explícita del problema de contorno, se evalúa dicha solución en los nodos, y se comparan los valores numéricos y las gráficas

```
% definiendo la solución exacta y comparando gráficas
%
v=dsolve('D2y-y=t^2','y(0)=0','y(1)=0');
solu_exacta=v
%
hold on
%
ezplot(v,[0,1])
 % calculando errores en nodos c_i-v(x_i)
nodos=x
solucionaproximadaennodos=[0;c;0];
solucionexactaennodos=subs(v,t,x);
%
error=solucionaproximadaennodos-solucionexactaennodos
```

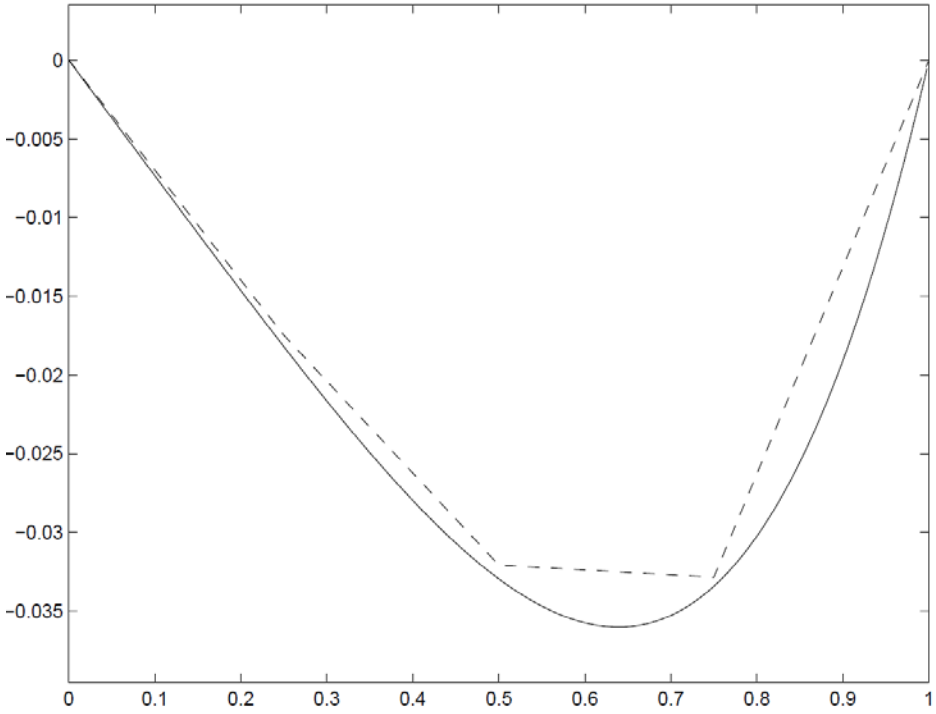
% Se observa que, en este ejemplo concreto, los errores en los nodos son relativamente pequeños para n pequeño. Esto es debido a las características del problema (se calculan primitivas, por ejemplo). Se ejecuta el fichero para distintos n viendo gráficamente cómo la aproximación se acerca a la solución para n grande y también cómo va disminuyendo el error en los nodos.

```
>> elementosfinitos(3)

matriz_sistema =

   -8.1667    3.9583         0
    3.9583   -8.1667    3.9583
         0    3.9583   -8.1667

termino_independiente =

    0.0182
    0.0651
    0.1432
```

```
coeficientes_solucion =

   -0.0183
   -0.0331
   -0.0336


Nodo_y_Aproximacionsolucion =

         0         0
    0.2500   -0.0183
    0.5000   -0.0331
    0.7500   -0.0336
    1.0000         0


solu_exacta =

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



nodos =

         0
    0.2500
    0.5000
    0.7500
    1.0000


error =

  1.0e-003 *

         0
   -0.0873
   -0.1600
   -0.1653
    0.0000
```



% En principio, el comando "double" delante de la integral nos devuelve una aproximación numérica de la integral aunque no se sepa calcular la primitiva, pero conviene observar que si las integrales no se pueden calcular explícitamente, y se introduce una cuadratura numérica simple (aproximación por el punto central), entonces el error aumenta, como nos muestra la figura de abajo



>> elementosfinitos(4)

matriz_sistema =

```
 -10.1333    4.9667         0         0
   4.9667  -10.1333    4.9667         0
        0    4.9667  -10.1333    4.9667
        0         0    4.9667  -10.1333
```

termino_independiente =

```
   0.0093
   0.0333
   0.0733
   0.1293
```

coeficientes_solucion =

```
  -0.0147
  -0.0281
  -0.0359
  -0.0303
```

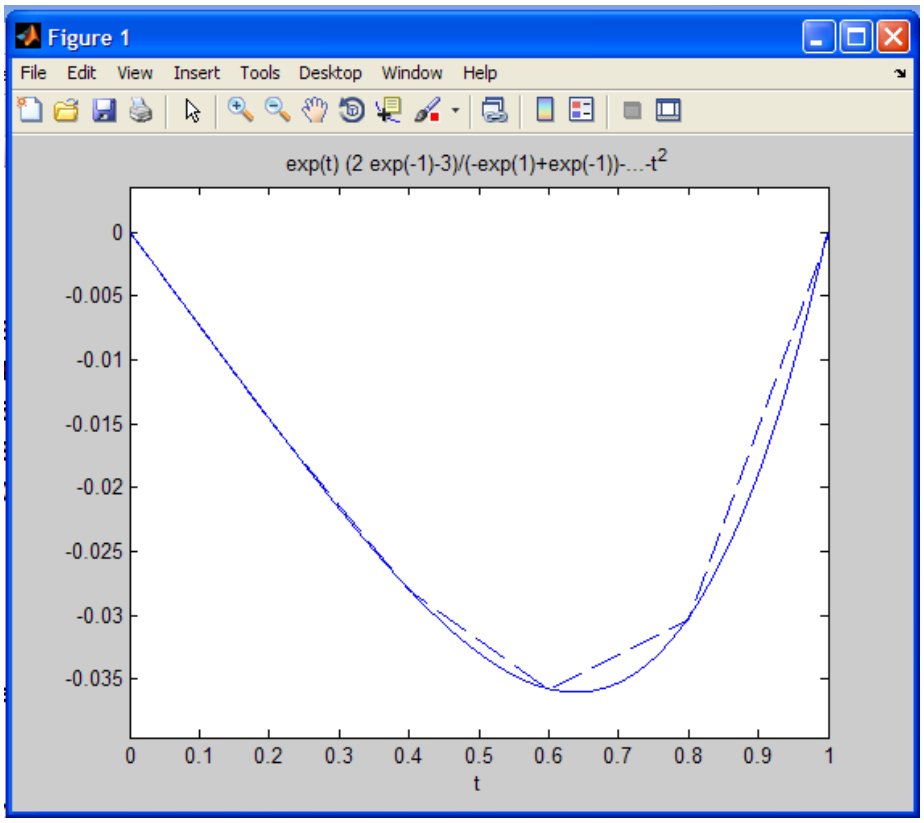
Nodo_y_Aproximacionsolucion =

```
         0         0
    0.2000   -0.0147
    0.4000   -0.0281
    0.6000   -0.0359
    0.8000   -0.0303
    1.0000         0
```

solu_exacta =

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

```
nodos =

         0
    0.2000
    0.4000
    0.6000
    0.8000
    1.0000


error =

  1.0e-003 *

         0
   -0.0442
   -0.0854
   -0.1107
   -0.0953
    0.0000
```



>> elementosfinitos(8)

matriz_sistema =

Columns 1 through 6

-18.0741	8.9815	0	0	0	0
8.9815	-18.0741	8.9815	0	0	0
0	8.9815	-18.0741	8.9815	0	0
0	0	8.9815	-18.0741	8.9815	0
0	0	0	8.9815	-18.0741	8.9815
0	0	0	0	8.9815	-18.0741
0	0	0	0	0	8.9815
0	0	0	0	0	0

Columns 7 through 8

0	0
0	0
0	0
0	0
0	0
8.9815	0
-18.0741	8.9815
8.9815	-18.0741

termino_independiente =

0.0016
0.0057
0.0126
0.0222
0.0345
0.0496
0.0674
0.0880

coeficientes_solucion =

-0.0082
-0.0162
-0.0239
-0.0304
-0.0349
-0.0359
-0.0319
-0.0207

Nodo_y_Aproximacionsolucion =

0	0
0.1111	-0.0082
0.2222	-0.0162
0.3333	-0.0239
0.4444	-0.0304
0.5556	-0.0349
0.6667	-0.0359
0.7778	-0.0319
0.8889	-0.0207
1.0000	0

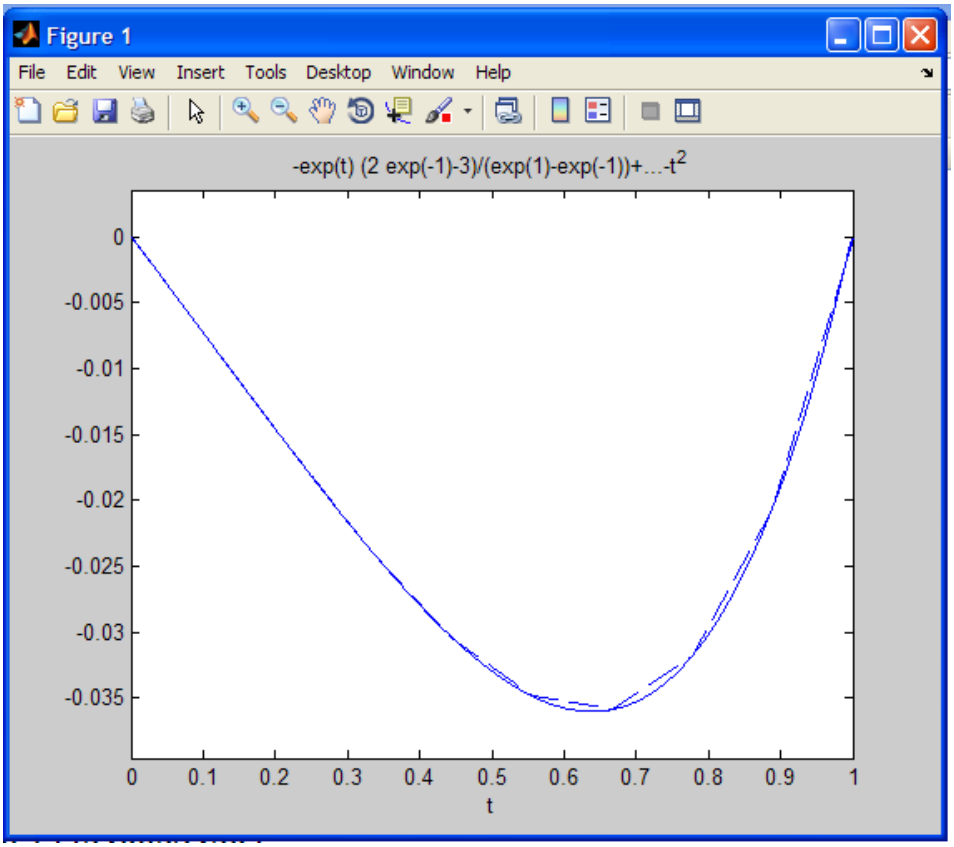
solu_exacta =

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

nodos =

0
0.1111
0.2222
0.3333
0.4444
0.5556
0.6667
0.7778
0.8889
1.0000

```
error =

  1.0e-004 *

         0
   -0.0744
   -0.1487
   -0.2202
   -0.2830
   -0.3274
   -0.3407
   -0.3056
   -0.2008
    0.0000
```



>> elementosfinitos(20)

matriz_sistema =

Columns 1 through 6

-42.0317	20.9921	0	0	0	0
20.9921	-42.0317	20.9921	0	0	0
0	20.9921	-42.0317	20.9921	0	0
0	0	20.9921	-42.0317	20.9921	0
0	0	0	20.9921	-42.0317	20.9921
0	0	0	0	20.9921	-42.0317
0	0	0	0	0	20.9921
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

……
……

Columns 19 through 20

0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0
0	0

```
         0         0
         0         0
         0         0
         0         0
         0         0
   20.9921         0
  -42.0317   20.9921
   20.9921  -42.0317
```

termino_independiente =

```
    0.0001
    0.0004
    0.0010
    0.0017
    0.0027
    0.0039
    0.0053
    0.0069
    0.0088
    0.0108
    0.0131
    0.0156
    0.0183
    0.0212
    0.0243
    0.0277
    0.0312
    0.0350
    0.0390
    0.0432
```

coeficientes_solucion =

```
   -0.0035
   -0.0070
   -0.0105
   -0.0139
   -0.0174
   -0.0207
   -0.0239
   -0.0269
   -0.0296
```

-0.0319
-0.0339
-0.0352
-0.0360
-0.0359
-0.0349
-0.0328
-0.0295
-0.0247
-0.0184
-0.0102

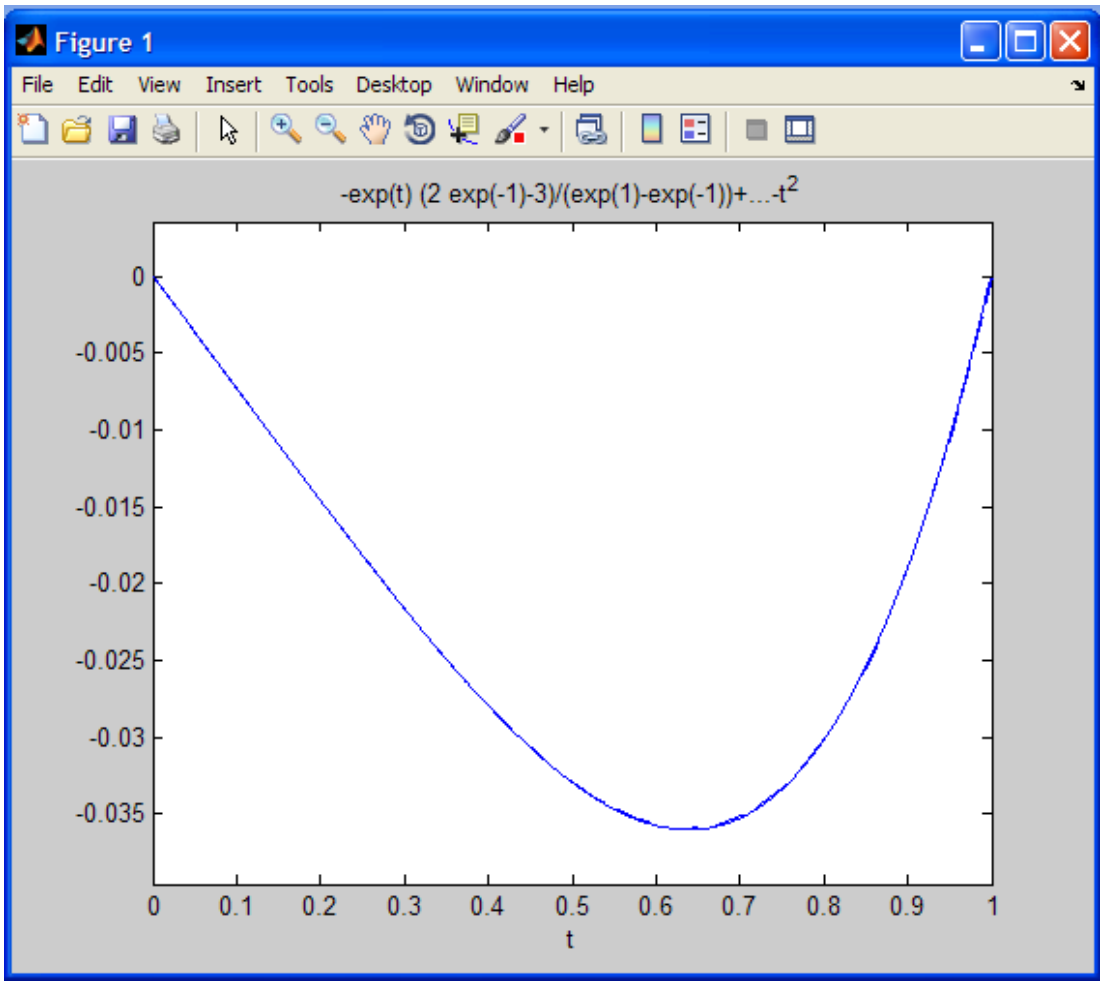
Nodo_y_Aproximacionsolucion =

0	0
0.0476	-0.0035
0.0952	-0.0070
0.1429	-0.0105
0.1905	-0.0139
0.2381	-0.0174
0.2857	-0.0207
0.3333	-0.0239
0.3810	-0.0269
0.4286	-0.0296
0.4762	-0.0319
0.5238	-0.0339
0.5714	-0.0352
0.6190	-0.0360
0.6667	-0.0359
0.7143	-0.0349
0.7619	-0.0328
0.8095	-0.0295
0.8571	-0.0247
0.9048	-0.0184
0.9524	-0.0102
1.0000	0

solu_exacta =

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

```
error =

 1.0e-005 *

      0
 -0.0580
```



>> type elementosfinitos

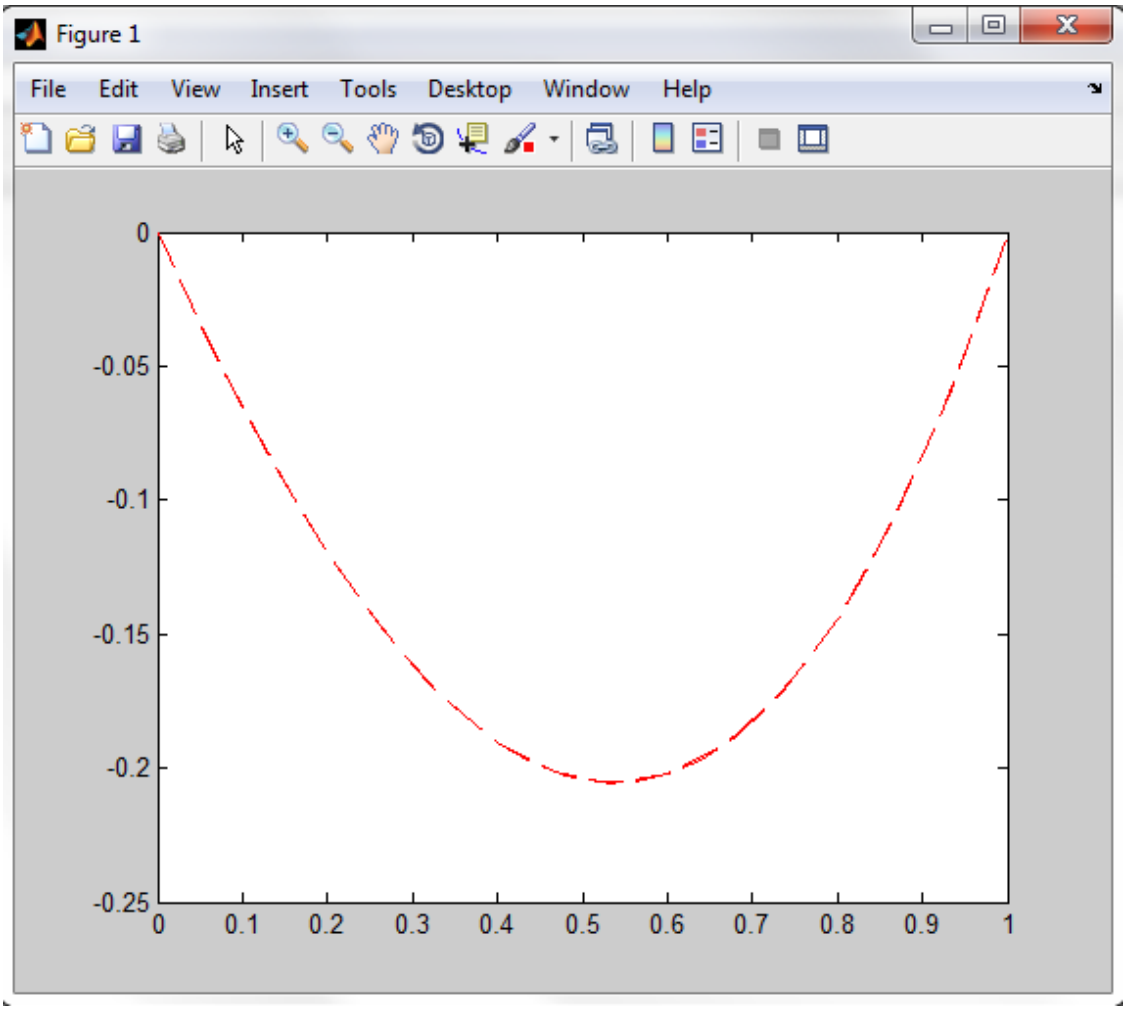
% Cambiando q por –t^2 y r por exp(t) en la función elementosfinitos.m; esto es,

```
syms t; % variable simbólica
q=-1;  % término que aparece multiplicando a y(x): y''+q(x)y=r(x)
r=t^2; % término independiente: y''+q(x)y=r(x)
```

% se cambia por

```
syms t; % variable simbólica
q=-t^2;  % término que aparece multiplicando a y(x): y''+q(x)y=r(x)
r=exp(t); % término independiente: y''+q(x)y=r(x)
```

% y se aproxima la solución del problema de contorno planteado antes (con la función galerkin), en el que no se conoce la solución explícita, aunque se sabe que existe y es única: y'' - x^2y = exp(x), x en (0,1), y(0)=0, y(1)=0.
Se comparan las gráficas para distintos valores de h, abajo para n=24 y n=14 respectivamente, viendo que se aproximan bastante



% También observamos que la aproximación en los nodos que coinciden tampoco varia en las cuatro primeras cifras decimales (se observa variación a partir de la quinta cifra decimal, utilizando format long). Esto puede servir de test de parada

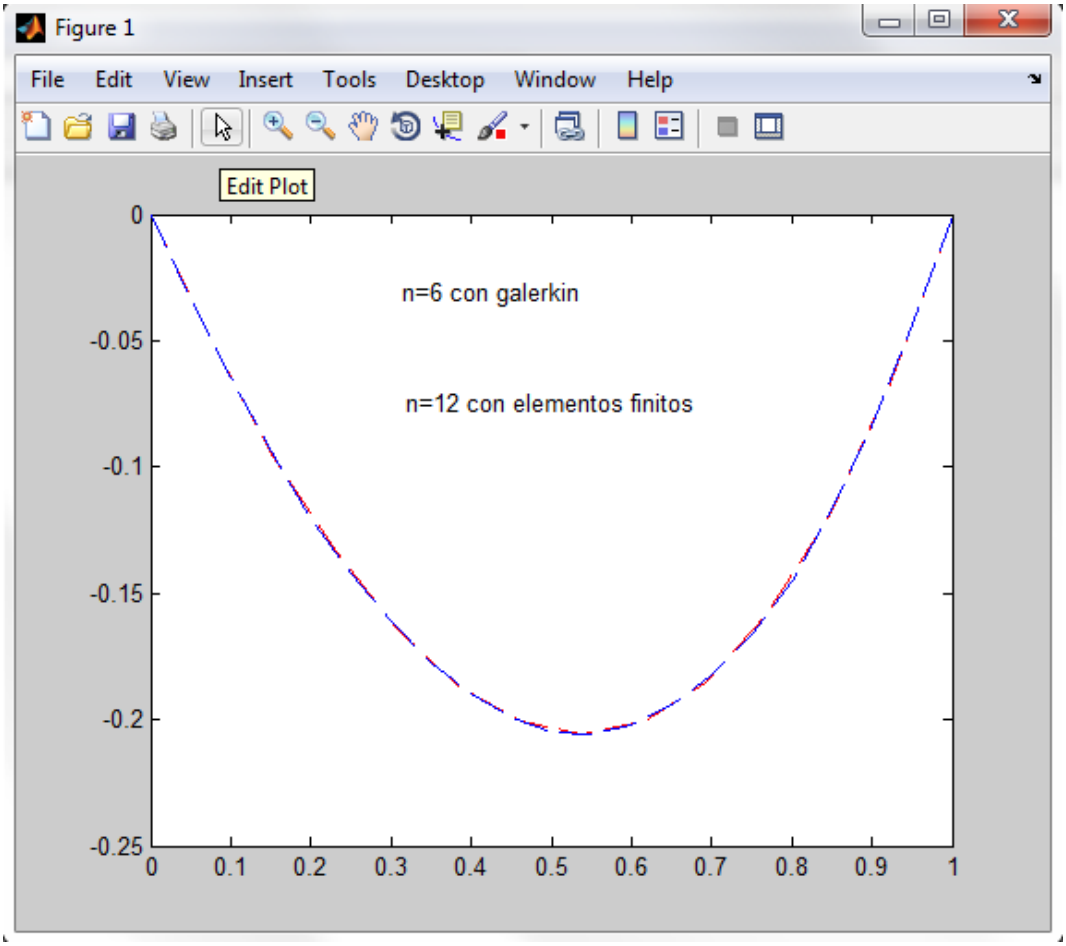
nodos_y_aproximacion =

0	0
0.0400	-0.0274
0.0800	-0.0531
0.1200	-0.0770
0.1600	-0.0992
0.2000	-0.1195
0.2400	-0.1378
0.2800	-0.1541
0.3200	-0.1684
0.3600	-0.1804
0.4000	-0.1902
0.4400	-0.1976
0.4800	-0.2027
0.5200	-0.2052
0.5600	-0.2051
0.6000	-0.2023
0.6400	-0.1967
0.6800	-0.1882
0.7200	-0.1767
0.7600	-0.1621
0.8000	-0.1442
0.8400	-0.1228
0.8800	-0.0979
0.9200	-0.0693
0.9600	-0.0367
1.0000	0

nodos_y_aproximacion =

0	0
0.0667	-0.0447
0.1333	-0.0846
0.2000	-0.1195
0.2667	-0.1489
0.3333	-0.1726
0.4000	-0.1902
0.4667	-0.2013
0.5333	-0.2055
0.6000	-0.2023
0.6667	-0.1914
0.7333	-0.1722
0.8000	-0.1442
0.8667	-0.1066
0.9333	-0.0589
1.0000	0

% Abajo se comparan las gráficas con galerkin.m y elementosfinitos.m para n=6 y 12 respectivamente



*% **Ejercicios: Modificaciones de galerkin.m y elementosfinitos.m***
(ecuación(p(x)y')'+q(x)y=r(x), condiciones de contorno no homogéneas, condiciones de contorno de tipo Neumann, problemas que no tienen solución única, etc.)

% Ejercicio *con la función p(x) distinta de 1*

$$\begin{cases} \left(\frac{e^{-(t+2)}}{(t+1)}y'\right)' + \frac{e^{-(t+2)}}{(t+1)^2}y = 1 \quad , \quad t \in (0,1) \\ y(0) = 0 \quad , \quad y(1) = 0 \end{cases}$$

% Como consecuencia de hacer el cambio de x por t+2 se tiene

$$\begin{cases} \left(\frac{e^{-x}}{(x-1)}y'\right)' + \frac{e^{-x}}{(x-1)^2}y = 1 \quad , \quad x \in (2,3) \\ y(2) = 0 \quad , \quad y(3) = 0 \end{cases}$$

% y que se resuelve con el comando dsolve despejando y''

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

```
>> pretty(v)

  3 exp(3) (t + 2)                          exp(t) (6 exp(2) - exp(3))    t exp(2) exp(t) (t + 4)
  ---------------- - exp(t + 2) (t + 2) - --------------------------- + -----------------------
  2 (2 exp(1) - 3)                              2 exp(1) - 3                        2
```

% La función galerking7.m nos aproxima a la solución. Tal y como se muestra abajo, los resultados dependen de la versión de Matlab. De manera general, si no se encuentran primitivas de las integrales, Matlab 2011 nos devuelve "Warning: Explicit integral could not be found" pero hace la aproximación numérica

>> galerking7(5) *% con la versión de Matlab 2011*

Warning: Explicit integral could not be found.
……
……

matriz_sistema =

-0.3138	-0.2267	-0.0921	-0.0863	-0.0499
-0.2267	-1.2429	-0.6364	-0.2191	-0.1884
-0.0921	-0.6364	-2.7906	-1.2475	-0.3927
-0.0863	-0.2191	-1.2475	-4.9572	-2.0616
-0.0499	-0.1884	-0.3927	-2.0616	-7.7427

termino_independiente =

0.6366
0
0.2122
0
0.1273

coeficientes_solucion =

-2.3578
0.4917
-0.1352
0.0630
-0.0231

solu_aproximada =

0.4917*sin(6.283*t) - 2.358*sin(3.142*t) - 0.1352*sin(9.425*t) + 0.06298*sin(12.57*t) - 0.02313*sin(15.71*t)

solu_exacta =

(3*exp(3)*(t + 2))/(2*(2*exp(1) - 3)) - exp(t + 2)*(t + 2) - (exp(t)*(6*exp(2) - exp(3)))/(2*exp(1) - 3) + (t*exp(2)*exp(t)*(t + 4))/2

coeficientes_fourier =

-2.3609
0.4934
-0.1375
0.0652
-0.0308

errorfourier_galerkin =

-0.0031
0.0017
-0.0022
0.0022
-0.0076

>> galerking7(5) *% con la versión de Matlab 2008*

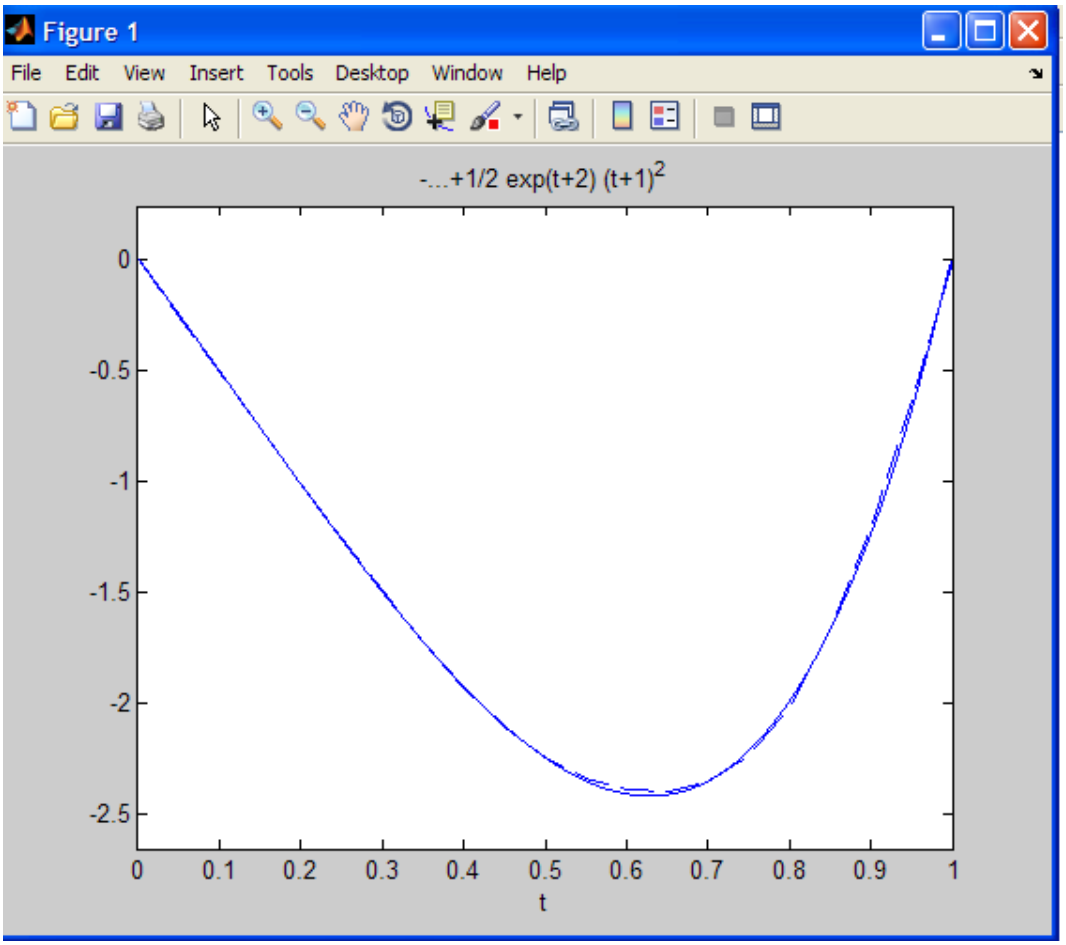
```
matriz_sistema =

  -0.3138       -0.2267 - 0.0000i   -0.0921 - 0.0000i   -0.0863 - 0.0000i  -0.0499 - 0.0000i
  -0.2267 + 0.0000i   -1.2429           -0.6364 - 0.0000i  -0.2191 + 0.0000i   -0.1884
  -0.0921 - 0.0000i    -0.6364           -2.7906           -1.2475 + 0.0000i       -0.3927
  -0.0863 + 0.0000i  -0.2191           -1.2475 + 0.0000i  -4.9572           -2.0616 + 0.0000i
  -0.0499          -0.1884 + 0.0000i  -0.3927 - 0.0000i  -2.0616 + 0.0000i  -7.7427


termino_independiente =

    0.6366
         0
    0.2122
         0
    0.1273

coeficientes_solucion =

  -2.3578 + 0.0000i
   0.4917 - 0.0000i
  -0.1352 + 0.0000i
   0.0630 - 0.0000i
  -0.0231 + 0.0000i

solu_aproximada =

(-2.358+.4438e-35*i)*sin(3.142*t)+(.4917-.4777e-35*i)*sin(6.284*t)+(-.1352+.1136e-
35*i)*sin(9.426*t)+(.6298e-1-.3237e-36*i)*sin(12.57*t)+(-.2313e-1+.1885e-
36*i)*sin(15.71*t)


solu_exacta =

-1/2*(t+2)*(exp(1)*exp(2)-4*exp(3))/(-3+2*exp(1))+1/2*exp(t)*(-
8*exp(3)+3*exp(2))/(-3+2*exp(1))+1/2*exp(t+2)*(t+1)^2


Warning: Imaginary parts of complex X and/or Y arguments ignored
> In galerking7 at 72
```

coeficientes_fourier =

-2.3609
0.4934
-0.1375
0.0652
-0.0308

errorfourier_galerkin =

-0.0031 - 0.0000i
0.0017 + 0.0000i
-0.0022 - 0.0000i
0.0022 + 0.0000i
-0.0076 - 0.0000i



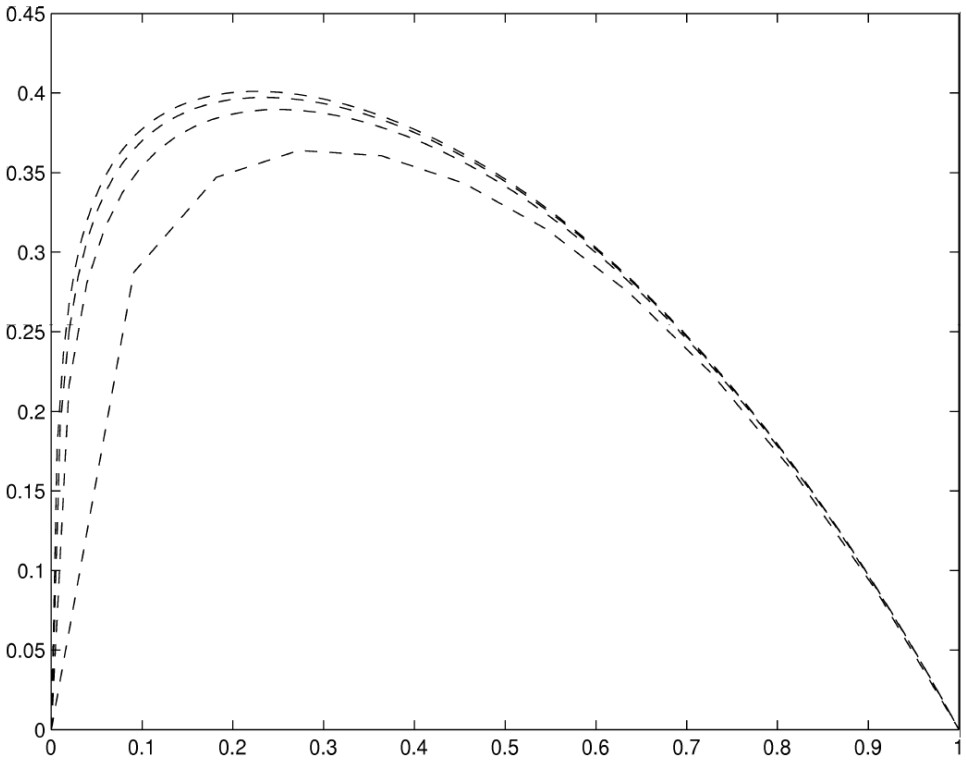
% Se debe ignorar la parte imaginaria pues en formato largo nos da que esta parte imaginaria es 0, nos hace la gráfica de la solución exacta y la de la aproximada, y se ve como el error mejora a medida que aumenta el número de términos de la suma (n)

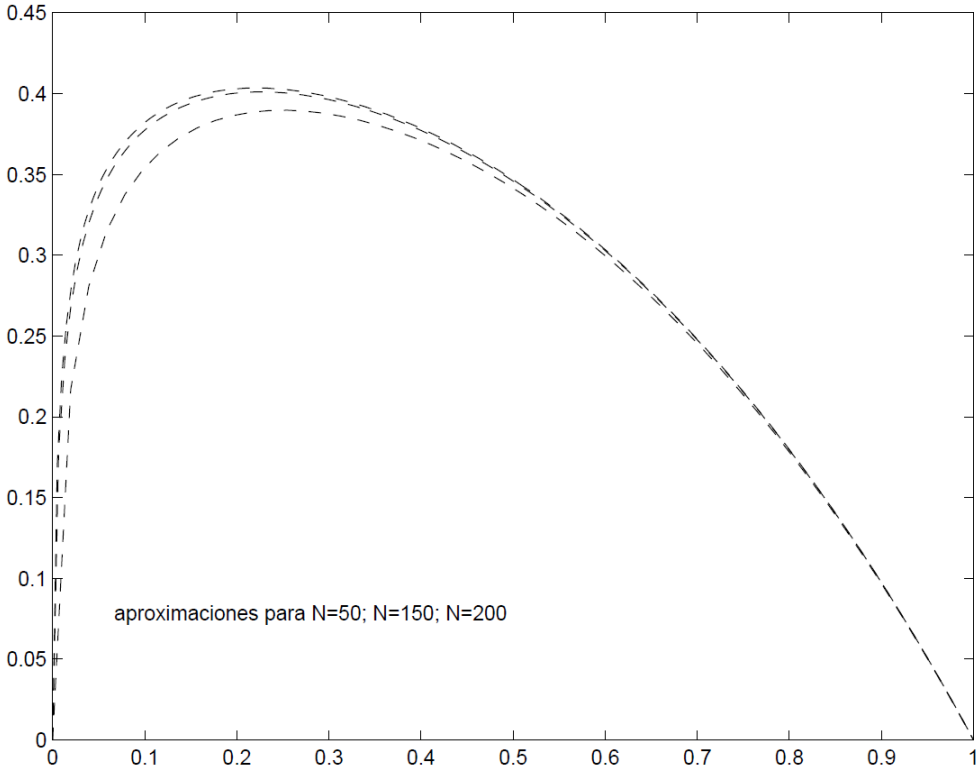
>> type galerking7

% En problemas con p(x) anulándose en x=0 (por ejemplo), como pasa con

$$\begin{cases} -(xy')' + 2y = e^x \quad , \quad x \in (0,1) \\ y(0) = 0 \quad , \quad y(1) = 0 \end{cases}$$

% Puede necesitarse n muy grande para tener una buena aproximación de la solución. Esto puede ser consecuencia del crecimiento rápido de la solución cerca del extremo en el que se anula p(x). La gráfica de abajo para distintos valores de n muestra que se necesitan más de 200 elementos para tener una buena aproximación de la solución





*% **Ejercicios** con solución explícita no conocida y con condiciones de contorno no necesariamente homogéneas*

$$\begin{cases} y'' - x^3 y = e^{x^2} & , \quad x \in (0,1) \\ y(0) = 0 & , \quad y(1) = 0 \end{cases}$$

$$\begin{cases} y'' - x^3 y = e^{x^2} & , \quad x \in (0,1) \\ y(0) = 0 & , \quad y(1) = 1 \end{cases}$$

% Para el primer problema se compraran soluciones con galerkin.m y con elementosfinitos.m. Se pide el h o el n de tal manera que el valor aproximado en un punto no cambie para n más grandes (h más pequeños).

% Para el segundo problema, se pueden homogenizar las condiciones de contorno, restando a y(x) una función polinómica que verifique estas condiciones (solución particular de las condiciones de contorno), y luego sumar esta función.

>> galerkinexg1(4)

matriz_sistema =

```
   5.0218   -0.0743    0.0285   -0.0101
  -0.0743   19.8547   -0.0844    0.0338
   0.0285   -0.0844   44.5340   -0.0873
  -0.0101    0.0338   -0.0873   79.0795
```

termino_independiente =

```
  -0.8800
   0.2296
  -0.3751
   0.1301
```

coeficientes_solucion =

```
  -0.1750
   0.0109
  -0.0083
   0.0016
```

solu_aproximada =

-.1750*sin(3.142*t)+.1087e-1*sin(6.284*t)-.8287e-2*sin(9.426*t)+.1609e-2*sin(12.57*t)

>> elementosfinitosexg1(9)

matriz_sistema =

-20.0001	9.9999	0	0	0	0	0	0	0
9.9999	-20.0006	9.9997	0	0	0	0	0	0
0	9.9997	-20.0019	9.9993	0	0	0	0	0
0	0	9.9993	-20.0043	9.9985	0	0	0	0
0	0	0	9.9985	-20.0084	9.9972	0	0	0
0	0	0	0	9.9972	-20.0145	9.9954	0	0
0	0	0	0	0	9.9954	-20.0230	9.9929	0
0	0	0	0	0	0	9.9929	-20.0343	9.9897
0	0	0	0	0	0	0	9.9897	-20.0488

termino_independiente =

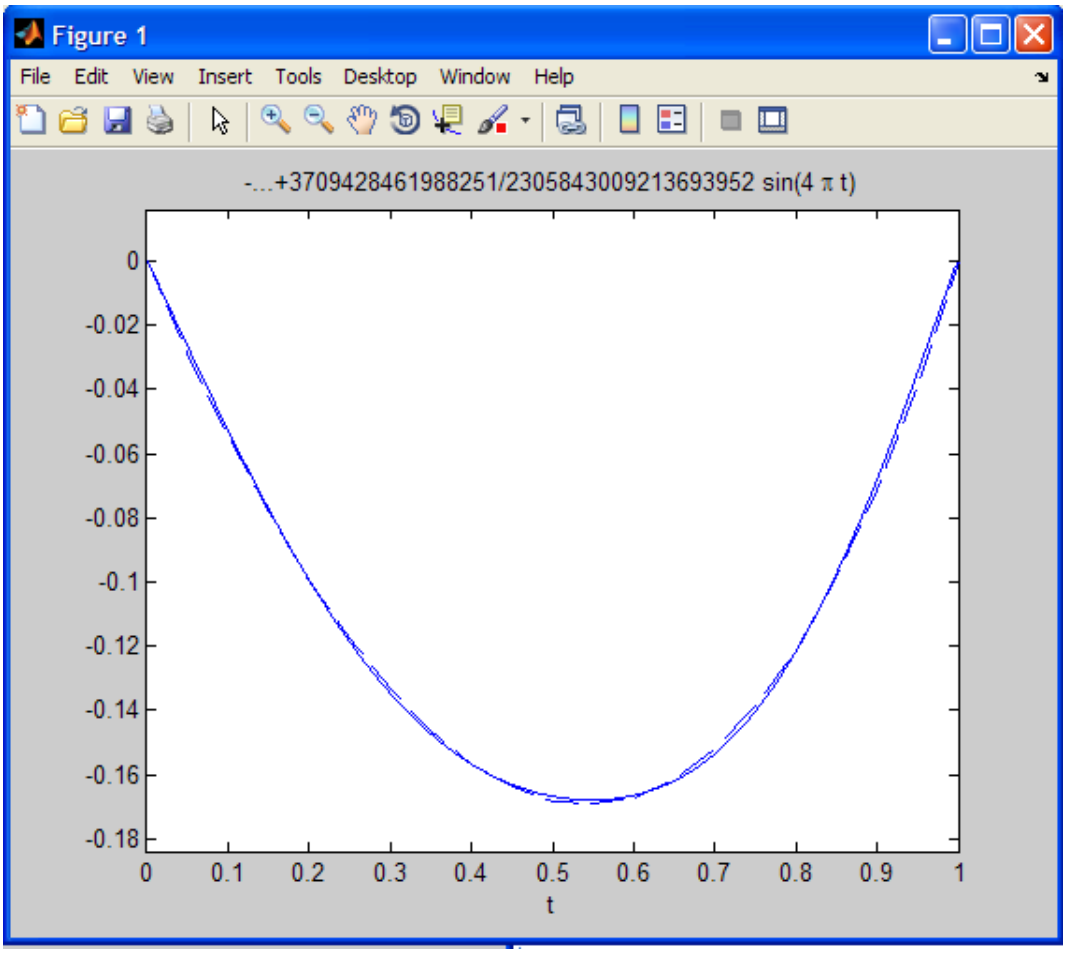
0.1012
0.1043
0.1096
0.1176
0.1287
0.1437
0.1638
0.1904
0.2258

coeficientes_solucion =

-0.0547
-0.0992
-0.1334
-0.1566
-0.1682
-0.1671
-0.1519
-0.1210
-0.0715

Nodo_y_Aproximacionsolucion =

0	0
0.1000	-0.0547
0.2000	-0.0992
0.3000	-0.1334
0.4000	-0.1566
0.5000	-0.1682
0.6000	-0.1671
0.7000	-0.1519
0.8000	-0.1210
0.9000	-0.0715
1.0000	0



% Observación importante, para este ejemplo, para n más grandes, la versión 2011 de Matlab funciona bien, pero la versión 2008 da problemas: errores importantes por problemas de evaluación de las integrales a partir del coeficiente 6 o siete. Esto se comprueba si evaluamos

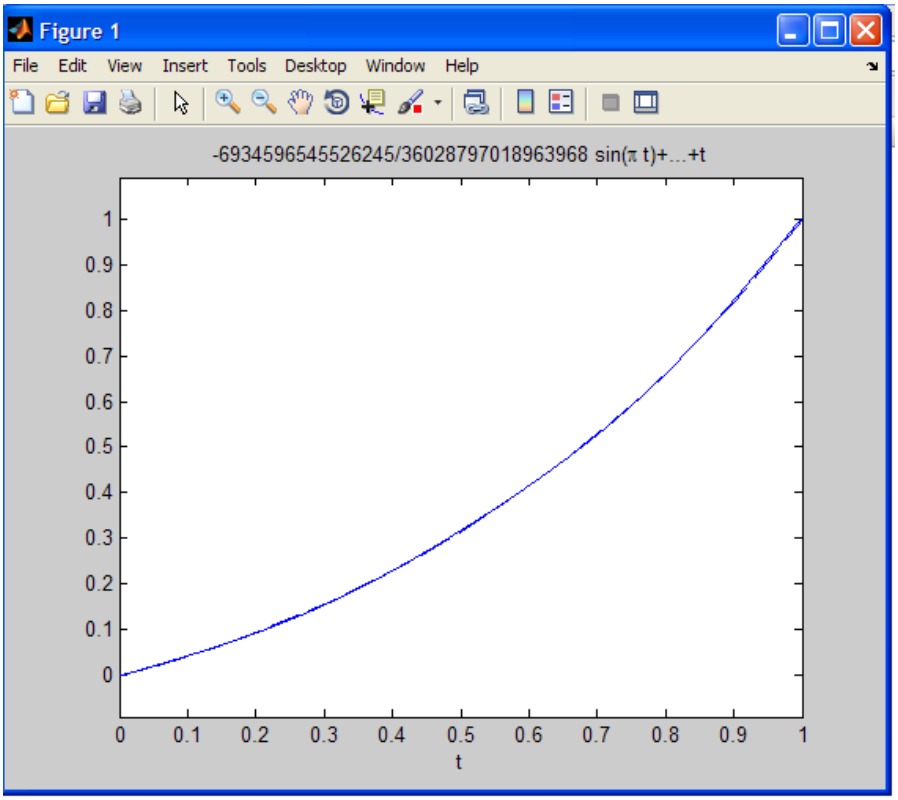
```
>> syms t

>> double(int(exp(t^2)*sin(9*pi*t),0,1))

ans =

  1.0000e+052

>> double(int(exp(t^2)*sin(5*pi*t),0,1))

ans =

    0.2322

% mientras que con el comando quad (para integración numérica)

>> quad('exp(t.^2).*sin(9*pi*t)',0,1)

ans =

    0.1307

>> quad('exp(t.^2).*sin(5*pi*t)',0,1)

ans =

    0.2322

% sin embargo, la modificación con quad en el lazo que define el vector f

for j=1:n
    f(j)= quad('exp(t.^2).*sin(j*pi*t)',0,1)
end
```

tampoco funciona

```
% Para el problema no homogéneo, se hace lo mismo sumando la función up=t, es
  decir, se resuelve
  y''-t^3y=t^4+exp(t), y(0)=0, y(1)=0
  y luego se suma up. Abajo la comparación de las gráficas (EF en línea discontinua)

>> elementosfinitosexg1nh(19);

>> hold on
```

>> galerkinexg1nh(4) ;

>> hold off



>> type elementosfinitosexg1

>> type elementosfinitosexg1nh

>> type galerkinexg1

>> type galerkinexg1nh(n)

*% **Ejercicio:** problema de contorno con una condición de tipo Neumann*

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

% La modificación de elementosfinitos.m supone introducir una ecuación más para el mismo n. La modificación de galerkin.m puede suponer cambiar de base para obtener una mejor aproximación de la solución.

>> elemfiniejerneumann(6)

nodos =

```
         0
    0.14286
    0.28571
    0.42857
    0.57143
    0.71429
    0.85714
         1
```

matriz_sistema =

```
   -14.095    6.9762         0         0         0         0         0
    6.9762   -14.095    6.9762         0         0         0         0
         0    6.9762   -14.095    6.9762         0         0         0
         0         0    6.9762   -14.095    6.9762         0         0
         0         0         0    6.9762   -14.095    6.9762         0
         0         0         0         0    6.9762   -14.095    6.9762
         0         0         0         0         0    6.9762   -7.0476
```

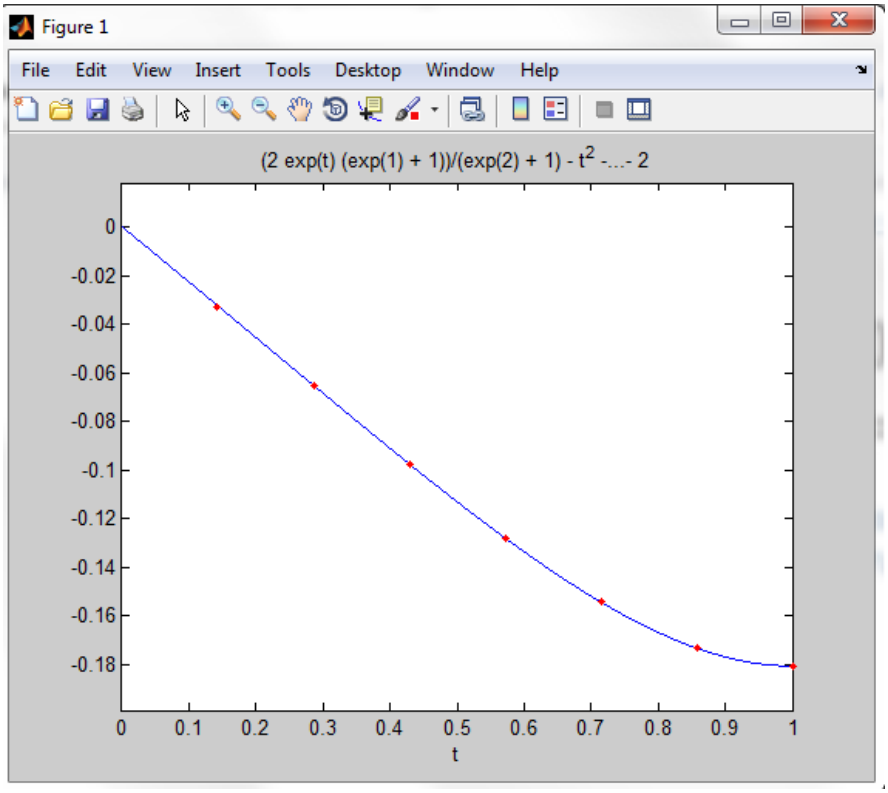
termino_independiente =

```
    0.0034
    0.0121
    0.0267
    0.0471
    0.0734
    0.1054
    0.0649
```

coeficientes_solucion =

```
   -0.0326
   -0.0653
   -0.0976
   -0.1281
   -0.1545
   -0.1735
   -0.1809
```

```
solu_exacta =

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


solucionaproximadaennodos =

         0
   -0.0326
   -0.0653
   -0.0976
   -0.1281
   -0.1545
   -0.1735
   -0.1809


solucionexactaennodos =

         0
   -0.0325
   -0.0652
   -0.0975
   -0.1280
   -0.1543
   -0.1733
   -0.1807


error =

  1.0e-003 *

         0
   -0.0372
   -0.0753
   -0.1143
   -0.1523
   -0.1864
   -0.2117
   -0.2215
```



% Los puntos rojos en la gráfica son los valores aproximados de la solución en los nodos; esto es lo que nos da la función elemfiniejerneumann.m

% Modificamos galerkin.m introduciendo una nueva base de funciones propias que verifiquen las condiciones de contorno del problema y(0)=0, y'(1)=0; con n=2 se tiene

>> galerkinnbc(2)

matriz_sistema =

```
   1.7337         0
        0   11.6033
```

termino_independiente =

```
  -0.2945
   0.1092
```

coeficientes_solucion =

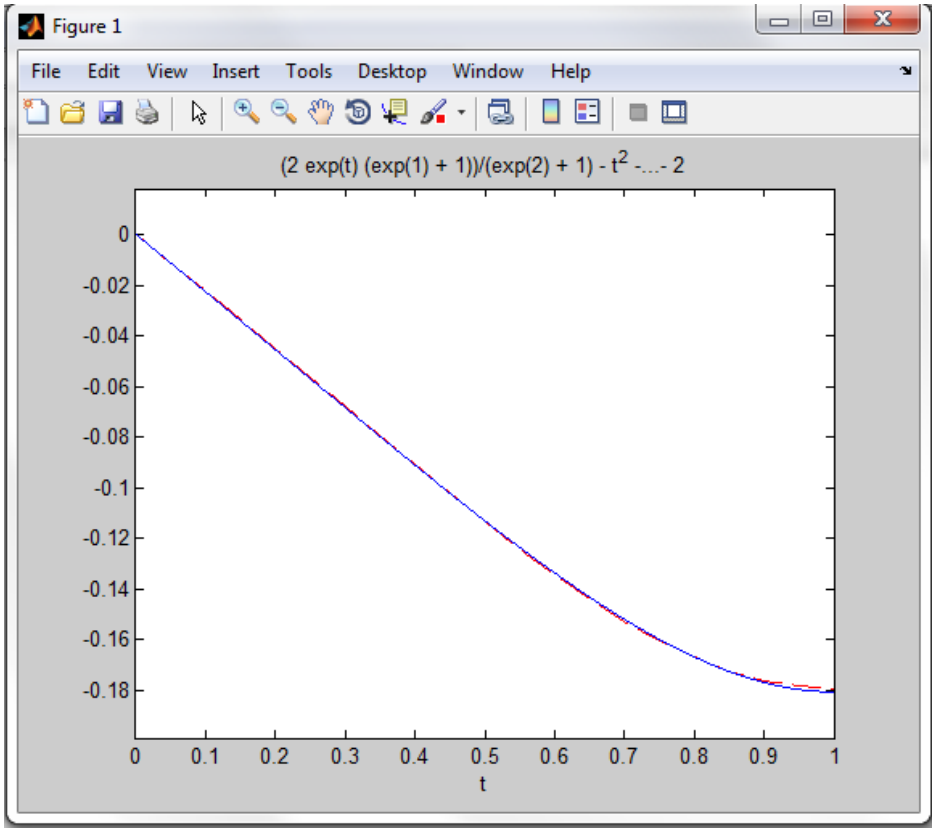
```
  -0.1699
   0.0094
```

solu_aproximada =

0.009409*sin(4.712*t) - 0.1699*sin(1.571*t)

solu_exacta =

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



% la gráfica en rojo es la aproximación de la solución con n=2 y galerkinnbc.m

>> type elemfiniejerneumann

>> type galerkinnbc

% Terminamos el cuaderno con los ejercicios de las primeras páginas: ejercicios para los que no se tiene la existencia y unicidad de solución del problema de contorno:

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

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

*% El problema de contorno homogéneo asociado tiene solución no trivial. Se demuestra que el primer problema de contorno no tiene solución y Matlab lo detecta devolviendo "empty" (no la encuentra). El segundo problema tiene infinitas soluciones (sin(pi*t)*C2-1/3*sin(2*pi*t)/pi^2 con C2 cualquier constante). Sin esta consideración previa, si intentamos aproximar la solución modificando los ficheros galerkin.m y elementosfinitos.m, se obtienen resultados muy dispares como se ve a continuación.*

% Para el primer problema galerkin.m (convenientemente modificado) no nos puede dar ninguna solución: la matriz A es singular y como se ve abajo nos da un valor muy grande (que no es fiable) para c(1)

```
matriz_sistema =

  Columns 1 through 8

    0.0000         0         0         0         0         0         0         0
         0   14.8044         0         0         0         0         0         0
         0         0   39.4784         0         0         0         0         0
         0         0         0   74.0220         0         0         0         0
         0         0         0         0  118.4353         0         0         0
         0         0         0         0         0  172.7181         0         0
         0         0         0         0         0         0  236.8705         0
         0         0         0         0         0         0         0  310.8925
         0         0         0         0         0         0         0         0
         0         0         0         0         0         0         0         0
```

```
  Columns 9 through 10

       0       0
       0       0
       0       0
       0       0
       0       0
       0       0
       0       0
       0       0
  394.7842  0
       0  488.5454


Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 6.412194e-019.
> In galerking8_9_nounibis at 46

termino_independiente =

    -0.31831
     0.15915
     -0.1061
    0.079577
   -0.063662
    0.053052
   -0.045473
    0.039789
   -0.035368
    0.031831


coeficientes_solucion =

 -1.0161e+015
    0.010751
   -0.0026876
    0.0010751
  -0.00053753
   0.00030716
  -0.00019197
   0.00012798
 -8.9588e-005
  6.5155e-005
```

% Para el segundo problema, con la misma matriz singular, galerkin.m sí encuentra solución (debido a que f(1) es también cero, y además, se puede resolver el resto del sistema dividiendo por los elementos de la diagonal principal). La solución que encuentra para aproximar, en este caso, es una de las encontradas explícitamente con C2=0

>> v=dsolve('D2y+pi^2*y=sin(2*pi*t)','y(0)=0','y(1)=0')

v =

sin(pi*t)*C2-1/3*sin(2*pi*t)/pi^2

% galerking.m, convenientemente modificada, para n=10, nos devuelve

matriz_sistema =

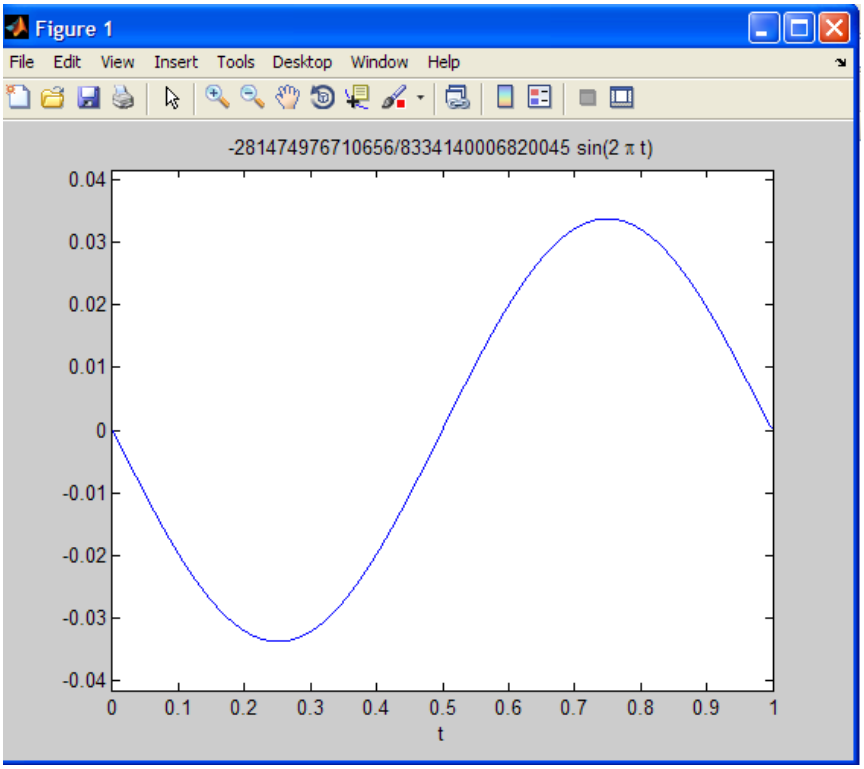
Columns 1 through 8

```
  0.0000    0         0         0         0         0         0         0
       0   14.8044    0         0         0         0         0         0
       0         0   39.4784    0         0         0         0         0
       0         0         0   74.0220    0         0         0         0
       0         0         0         0  118.4353    0         0         0
       0         0         0         0         0  172.7181    0         0
       0         0         0         0         0         0  236.8705    0
       0         0         0         0         0         0         0  310.8925
       0         0         0         0         0         0         0         0
       0         0         0         0         0         0         0         0
```

Columns 9 through 10

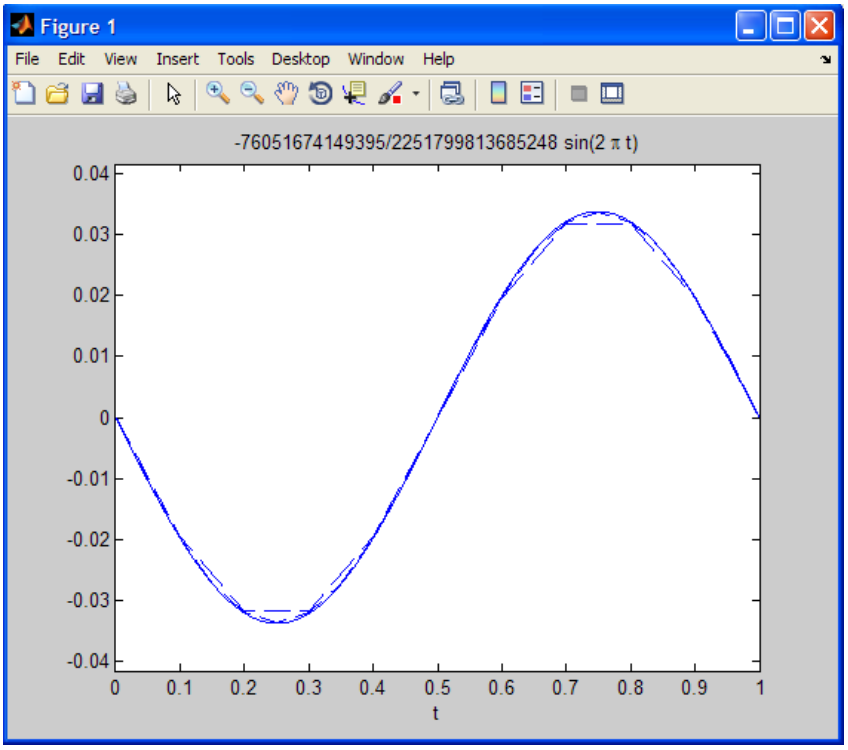
```
       0         0
       0         0
       0         0
       0         0
       0         0
       0         0
       0         0
       0         0
  394.7842       0
       0   488.5454
```

```
termino_independiente =

     0
-0.5000
     0
     0
     0
     0
     0
     0
     0
     0

Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 6.412194e-019.
> In galerking8_9_nounibis at 46

coeficientes_solucion =

     0
-0.0338
     0
     0
     0
     0
     0
     0
     0
     0

solu_aproximada =

-.3377e-1*sin(6.284*t)
```



*% que es la gráfica de la función -1/3*sin(2*pi*t)/pi^2*

% La función elementofinitos.m, convenientemente modificada, también encuentra aproximación a esta solución: para h=1/10, h=1,20 h=1/30 nos proporciona las aproximaciones en la gráfica de abajo



% Para el problema sin solución Matlab lo advierte:

>> v=dsolve('D2y+pi^2*y=t','y(0)=0','y(1)=0')

Warning: Explicit solution could not be found.
> In dsolve at 333

v =

[empty sym]

% galerkin.m no da solución como se ha indicado antes (mejor dicho, nos da una solución que no es fiable con el primer coeficiente de la aproximación del orden de 10^{15})

solu_aproximada =

-.1016e16*sin(3.142*t)+.1075e-1*sin(6.284*t)-.2688e-2*sin(9.426*t)+.1075e-2*sin(12.57*t)-.5375e-3*sin(15.71*t)+.3072e-3*sin(18.85*t)-.1920e-3*sin(21.99*t)+.1280e-3*sin(25.14*t)-.8959e-4*sin(28.28*t)

% la función elementosfinitos.m, modificada convenientemente, nos da soluciones que no se van aproximando a ninguna, como se ve en la gráfica la relativa a h=1/10, h=1/20, h=1/30, h=1/40