EDP: Modelos de propagación de ondas, vibraciones y difusión del calor.

Capítulo 6 - Libro **+** ***Hoja de problemas 9*** **+** ***Videos***

Resumen de contenidos:

- **Problemas de valor iniciales para EDP de primer orden: ondas**
- **Problemas de valores inicial para EDP de segundo orden: ondas y calor**
- **Problemas mixtos para la ecuación de ondas (EDP segundo orden)**
- **Modelos de vibraciones de vigas (EDP cuarto orden)**
- **Problemas de contorno para la ecuación de Laplace**
- **Modelos de vibraciones de membranas**

% Este cuaderno contiene enlaces a videos que modelan distintos fenómenos como la difusión del calor, la propagación de ondas o las vibraciones de membranas. Se consideran modelos/ecuaciones adimensionales. Algunas gráficas se repiten para dar una visión completa del contenido de cada una de las secciones del cuaderno.

Modelos de propagación de ondas 1-D: problemas de Cauchy para EDP de primer orden en dominios no acotados

% El problema u_t+cu_x=0, u(x,0)=f(x) modela, e.g., la propagación de una onda en un canal 1-D; u(x,t) representa, e.g., la altura del punto x del medio en el tiempo t, sabiendo que en el tiempo t=0 la forma de la onda está dada por la función f(x). La solución es la onda u=f(x-ct) que viaja en la dirección positiva del eje de las x sin cambio de forma con velocidad constante c (si c>0), y viaja en la dirección negativa si c<0; |c| es la velocidad.

% Ejemplo: u_t+cu_x=0, u(x,0)=1/(1+x^2); c=2. Dibujamos la superficie solución

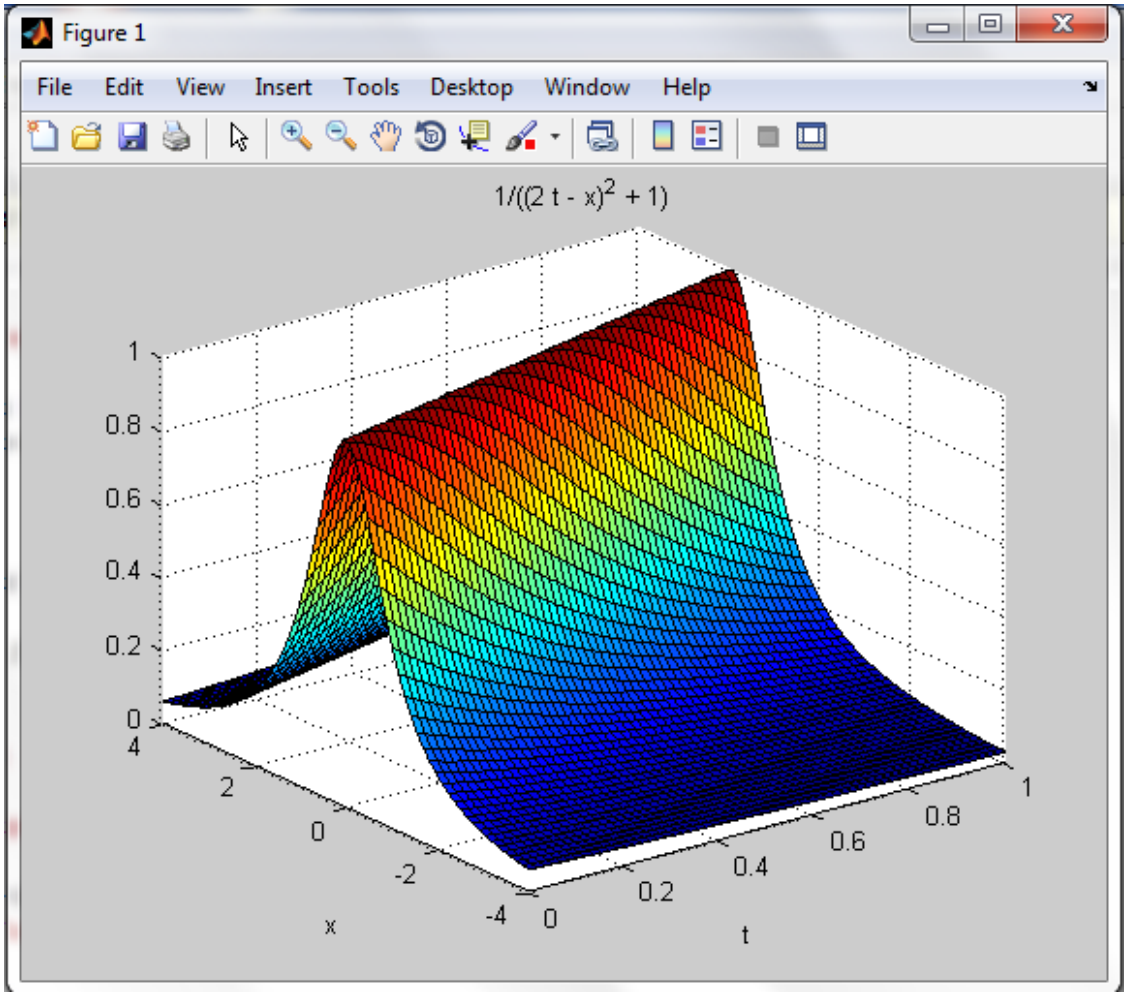
>> syms x t

>> u=1/(1+(x-2*t)^2)

u =

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

>> ezsurf(u,[0,1],[-4,4])



% El fichero viajaonda.m contiene instrucciones para que se dibuje la superficie solución u=f(x+ct), con f=1/(x^2+1), la forma de la onda para t=0,1,2,3,4, y la superposición de las cinco gráficas ("la onda en 5 instantes de tiempo distintos"), y para distintos valores de c positivos y negativos. Se observa como la onda avanza más hacia la izquierda, en el mismo tiempo, para c más grande:

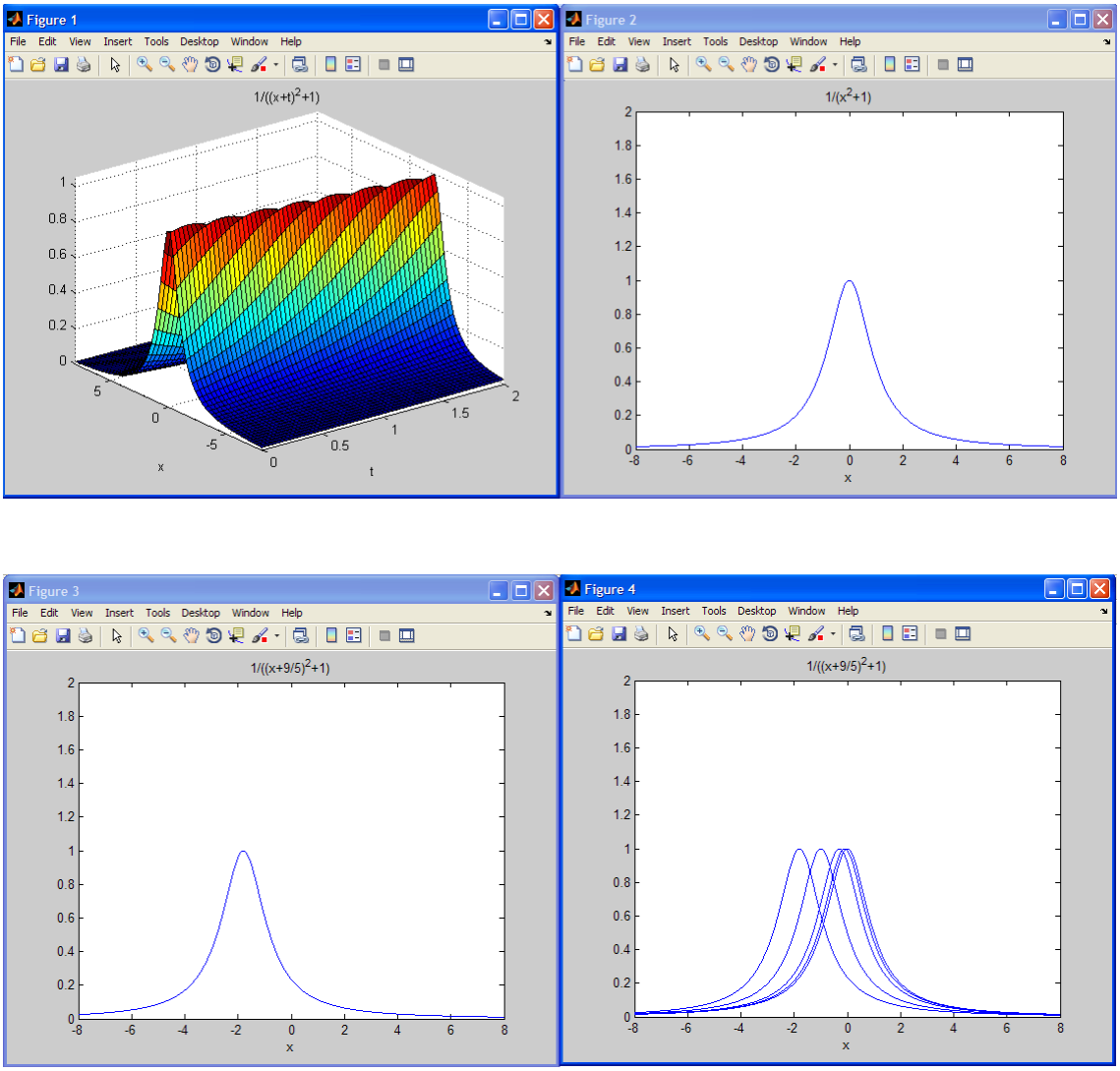
>> viajaonda(1)

u =

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

w1 =

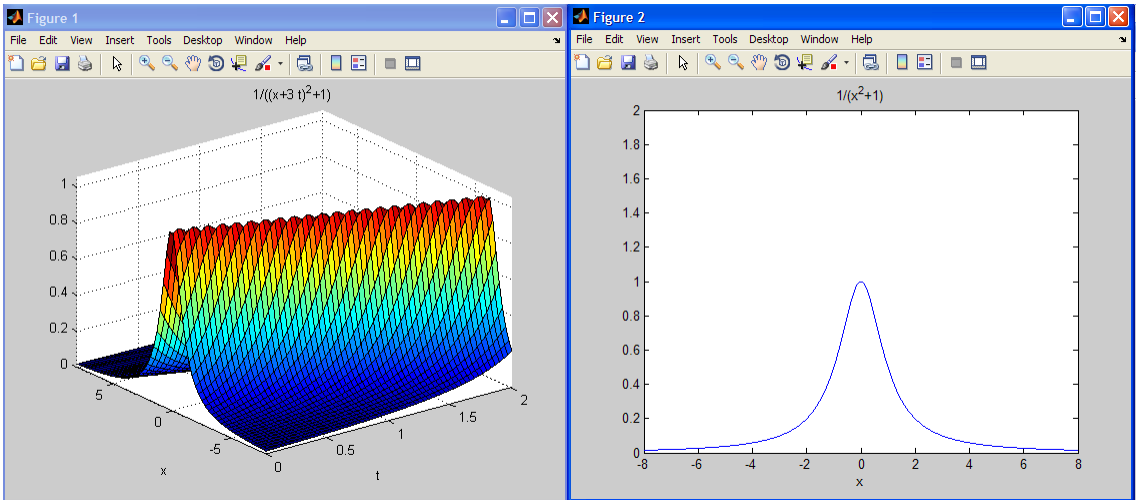
1/(x^2+1)

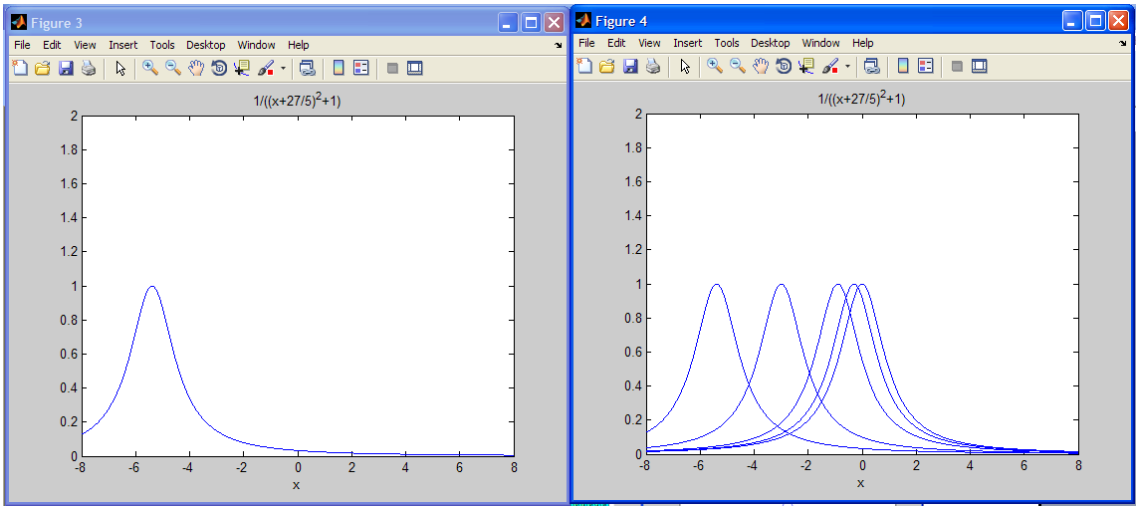


>> viajaonda(3)

u =

1/((x+3*t)^2+1)



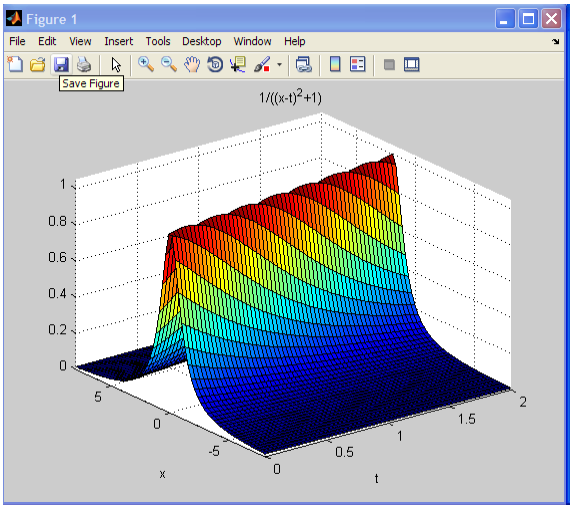


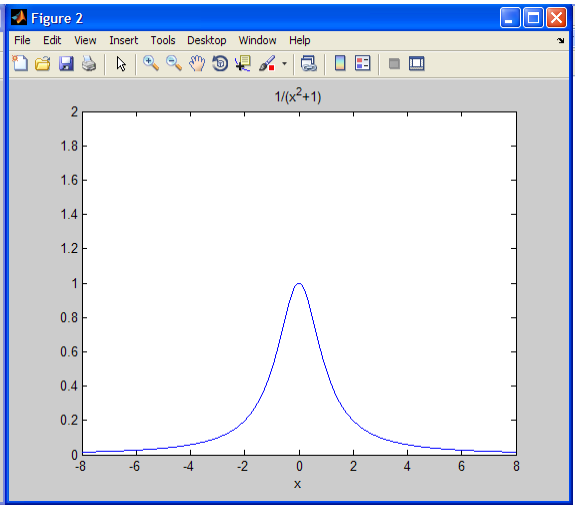
% Similar fenómeno se observa con la onda avanzando hacia la derecha, para c<0

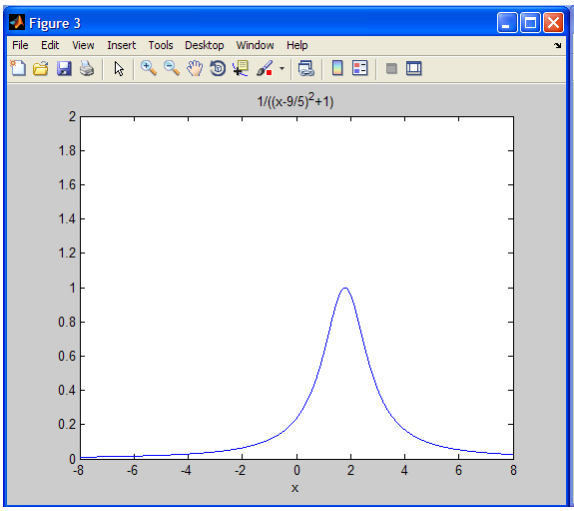
>> viajaonda(-1)

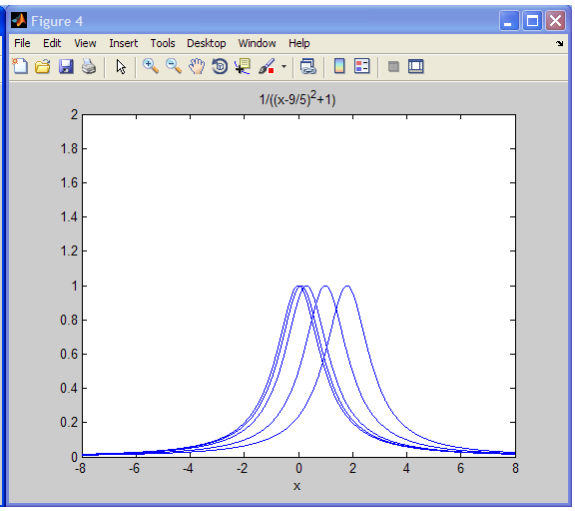
u =

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





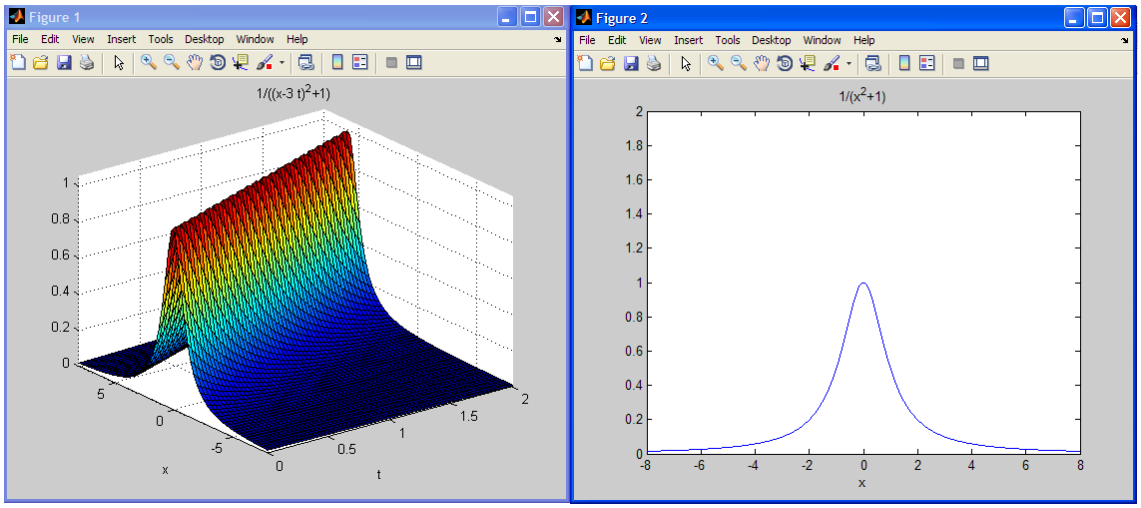


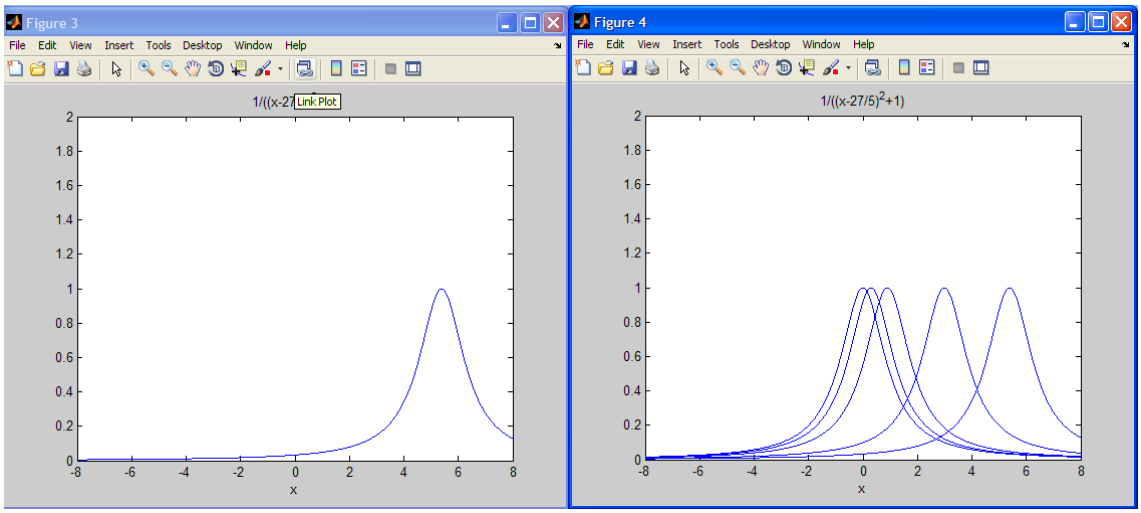


>> viajaonda(-3)

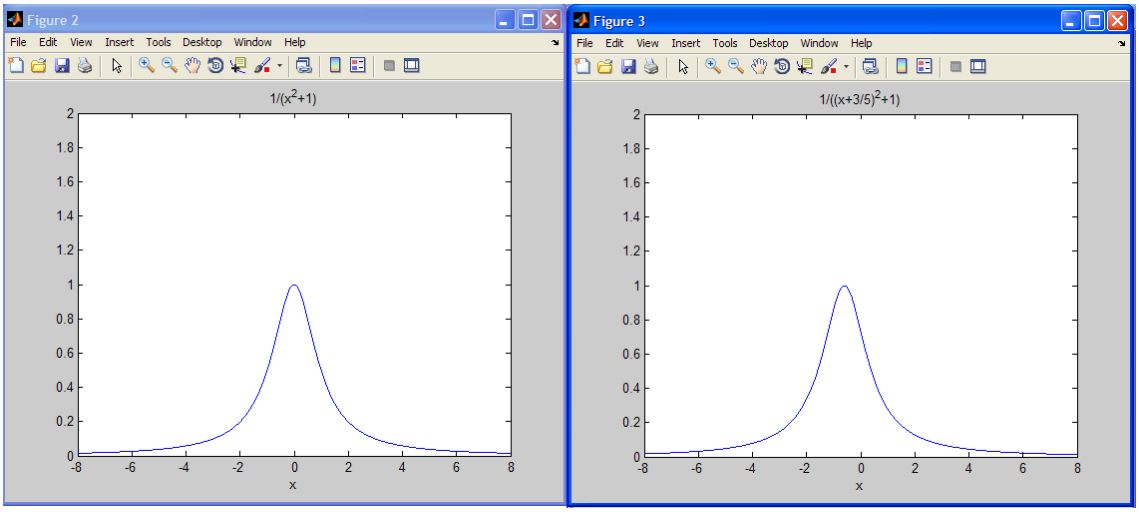
u =

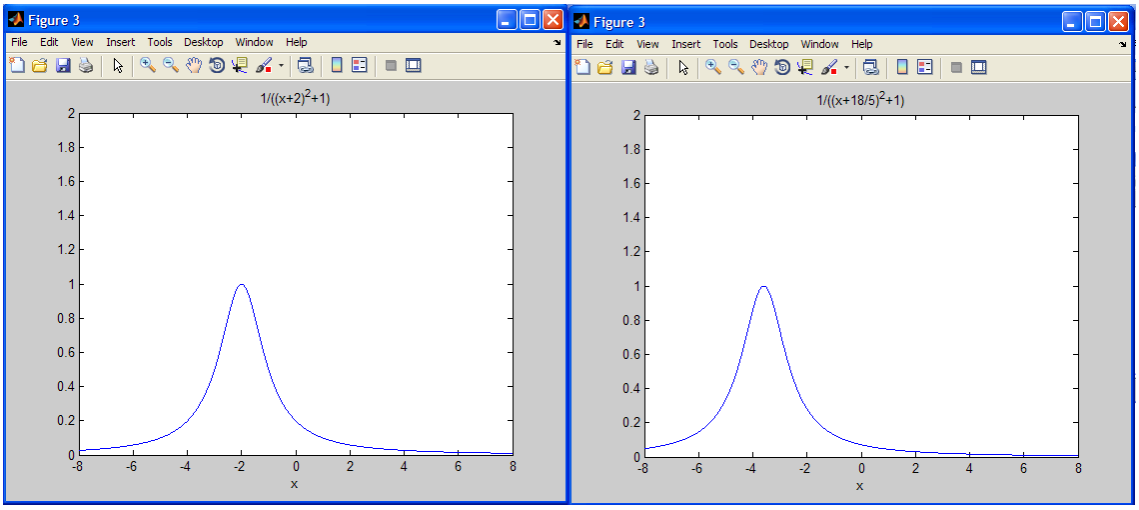
1/((x-3*t)^2+1)





% Las figuras de arriba nos muestran la superficie y la forma de la onda en t=0 (primera fila); la forma de la onda en t=4 y la superposición, para los distintos valores de t (segunda fila). Las figuras de abajo muestran la evolución de las ondas, que se obtendría sacando fotos en cada instante de tiempo, mientras que con un "movie" veríamos el movimiento: esto lo permiten los comandos Matlab "moviein", "getframe" y "movie"





% dicho movimiento hacia la izquierda se ve ejecutando la función viajaondassolobis.m

>> viajaondassolobis ***% ver movie: viajaondasde***

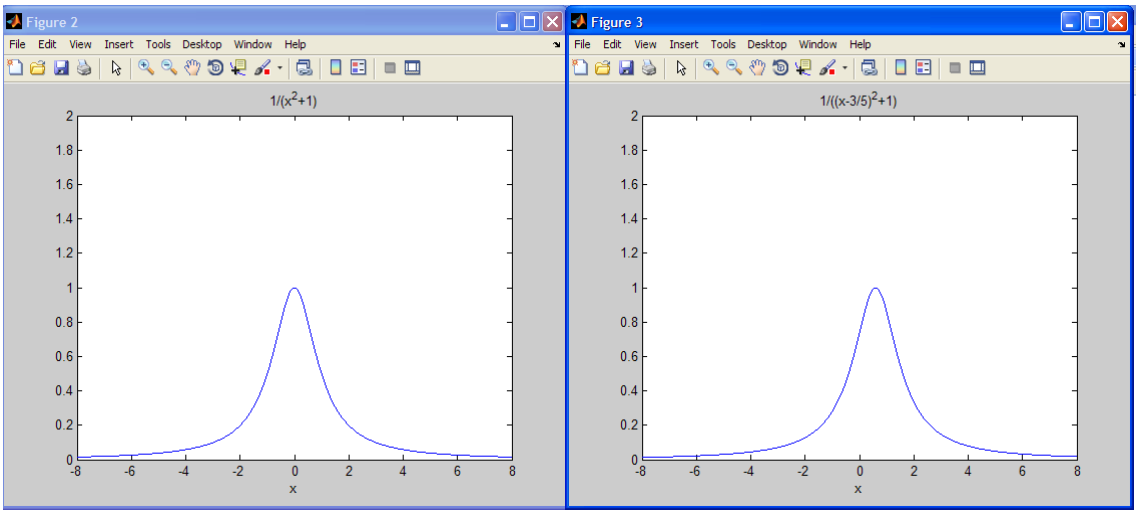
```
c =
    1


h =

1/(x^2+1)
```

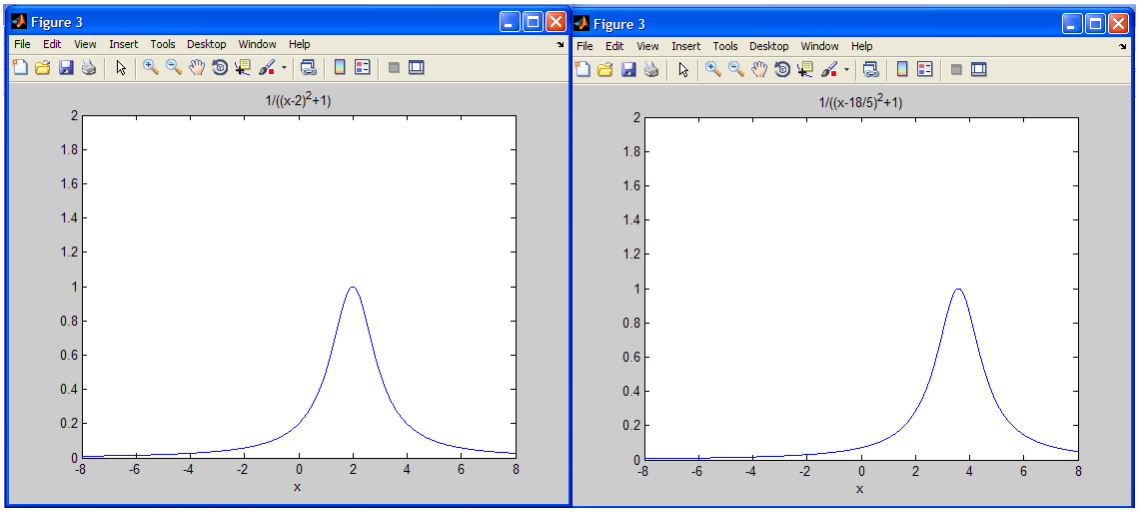
% o hacia la derecha ejecutando viajaondassolodbis.m. h es el dato inicial

>> viajaondassolodbis ***% ver movie: viajaondasiz***

c = 1

h =1/(x^2+1)





>> type viajaondas

>> type viajaondassolobis

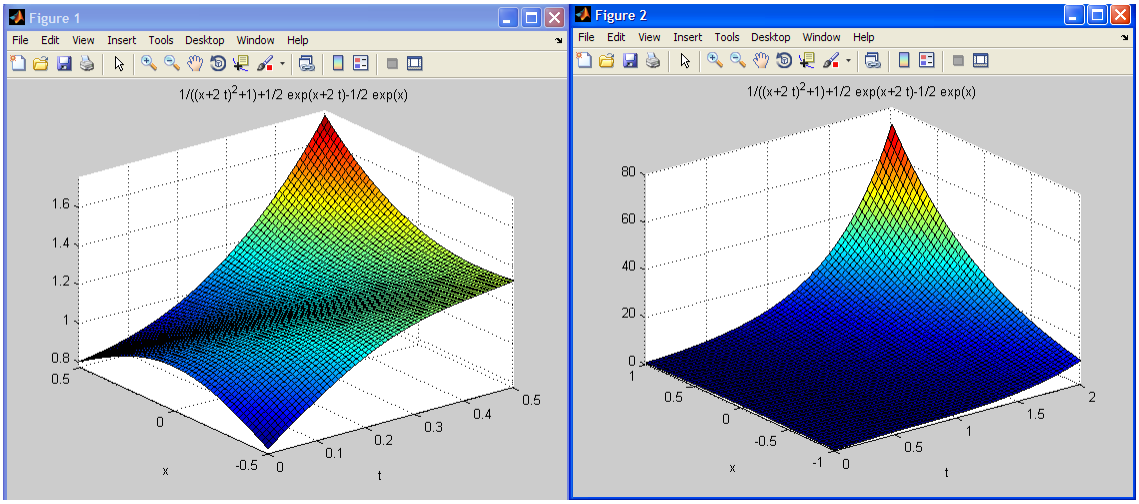
>> type viajaondassolodbis

% Este tipo de fenómeno (ondas viajeras sin cambio de forma) no ocurre si la ecuación es no homogénea. Así, adaptando la función Matlab viajaonda.m, simulamos otros efectos.

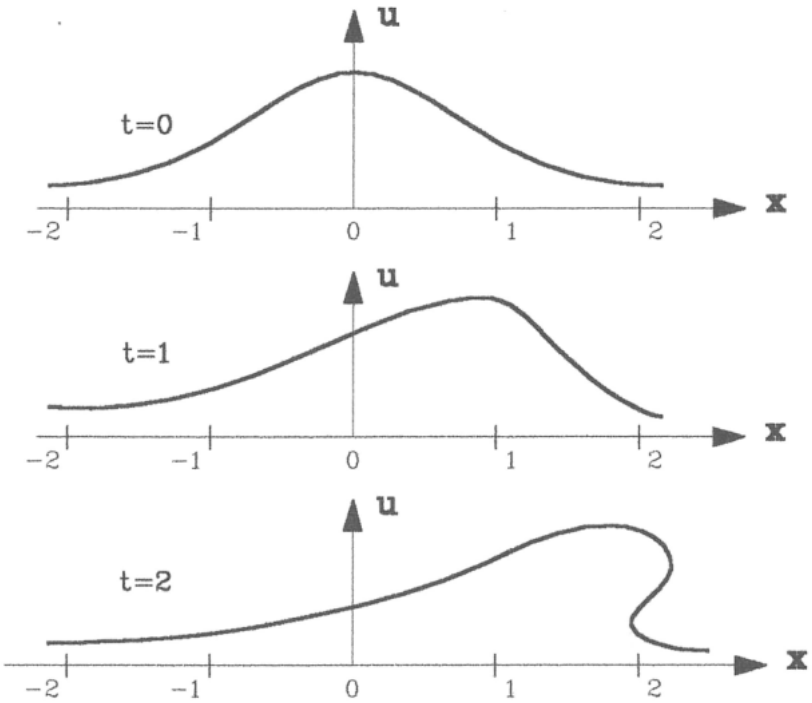
*% Ejemplo: u_t- 2u_x=e^x, u(x,0)= 1/(x^2+1). Se resuelve utilizando la separación de variables para encontrar una solución particular de la EDP y después homogeneizando esta EDP. Solución: u(x,t)= 1/((x+2*t)^2+1)+exp(x+2*t)/2-exp(x)/2*

```
>> syms t x

>> u=1/((x+2*t)^2+1)+exp(x+2*t)/2-exp(x)/2

u =

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

>> diff(u,t)-2*diff(u,x)-exp(x) % verificación de solución de la EDP

ans =

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

>> simplify(ans) % simplifica y demuestra que la respuesta es 0

ans =

0

>> subs(u,t,0) % verificación de la condición inicial

ans =

1/(x^2+1)

>> ezsurf(u,[0,0.5],[-0.5,0.5]) % dibujando la superficie solución

>> figure(2)

>> ezsurf(u,[0,2],[-1,1])
```

% tenemos las gráficas dependiendo de los intervalos (observamos que la exponencial se hace muy grande enseguida y por eso conviene tomar intervalos pequeños). Podemos apreciar que ya no tenemos una superficie con forma de "túnel", y que, al cortar por planos t=constante, cambia de forma de la onda, como consecuencia del término no homogéneo en la EDP



% El fenómeno de la propagación de ondas sin cambio de forma tampoco ocurre si la EDP es no lineal, o no es de coeficientes no constantes. Ejemplo: problema de valor inicial para la ecuación de Burgers: u_t+uu_x=0, u(x,0)=1/(x^2+1) , en la figura de abajo (ver libro de apuntes) vemos como "la onda rompe" pasado un tiempo



Modelos de propagación de ondas y difusión del calor en 1-D: problemas de Cauchy para EDP de segundo orden en dominios no acotados

% Partiendo de la ecuación de primer orden, otro aspecto interesante de la propagación de ondas es la superposición de ondas: dos ondas que viajan sin cambio de forma, en sentidos contrarios, con la misma velocidad. Esto nos lleva a la ecuación de ondas de segundo orden u_tt-c^u_xx=0 y a los problemas de valores iniciales para EDP.

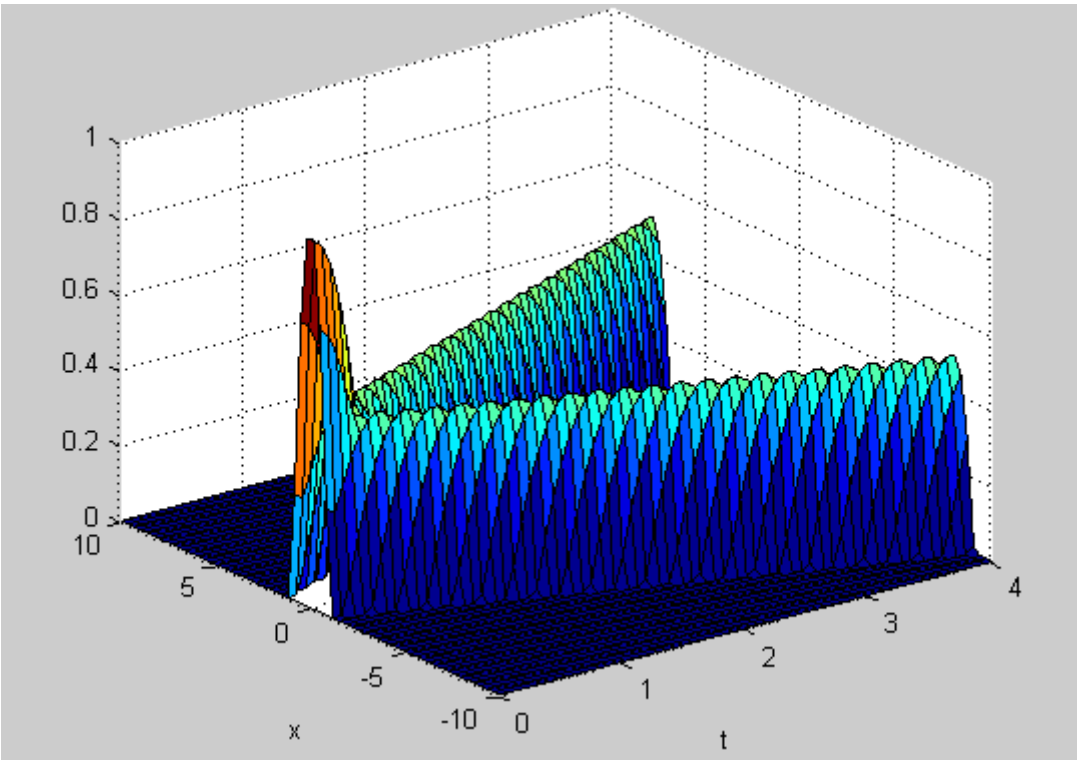
% Es un problema de Cauchy para la ecuación de ondas de segundo orden en dimensión 1 (1-D), también llamada ecuación de la cuerda vibrante: modela las vibraciones de una cuerda de longitud infinita que en estado de equilibrio se encuentra en el eje de las x, en el instante t=0 se desplaza de su posición de equilibrio, toma la forma de la función f(x) y empieza a vibrar con velocidad inicial de cada punto x, g(x). Las vibraciones u oscilaciones se producen en un plano vertical, u(x,t) representa, e.g., la altura del punto x de la cuerda en el tiempo t . c es una constante que depende de las características físicas del medio.

*% u_tt-c^2u_xx=0, u(x,0)=f(x), u_t(x,0)=g(x); la solución viene dada por la fórmula de d'Alembert: u=(f(x+ct)+f(x-ct))/2+ (1/(2c))*int(g,x-ct,x+ct) que muestra fenómenos que recuerdan a los que se observan al tirar una piedra en el centro de un estanque muy grande con el agua en reposo.*

% En particular, para f(x) un dato localizado y g(x)=0, dicha fórmula nos da la división de una onda en dos (de altura máxima ½ de la de la onda inicial) que viajan en direcciones distintas sin cambio de forma. Ejemplo: f = (x^2-1) en (-1,1) y f=0 fuera, y g=0; la fórmula de d'Alembert nos da

```
>> syms x t

>> f=(heaviside(x+1)-heaviside(x-1))*(x^2-1)

f =

-(x^2 - 1)*(heaviside(x - 1) - heaviside(x + 1))

>> u=(subs(f,x,x+2*t)+subs(f,x,x-2*t))/2

u =

- (((2*t - x)^2 - 1)*(heaviside(x - 2*t - 1) - heaviside(x - 2*t + 1)))/2 - (((2*t + x)^2 -
1)*(heaviside(2*t + x - 1) - heaviside(2*t + x + 1)))/2
```

>> ezsurf(-u,[0,4],[-10,10])



% La fórmula de d'Alembert con perturbación localizada para f en [-1,1], g=0, se encuentra en el fichero dalembert1112.m usando la función de heaviside (que permite una escritura simplificada de f), o en dalembert0708.m para versiones previas de Matlab (sin utilizar heaviside). Dado que la solución es superposición de dos ondas que viajan en distinto sentido y con igual velocidad, pasado un tiempo, se separan las ondas, la onda inicial se va deformando durante un tiempo hasta que se divide en dos con altura máxima ½ de la de la onda original, y la misma forma; estas ondas nunca se vuelven a encontrar.

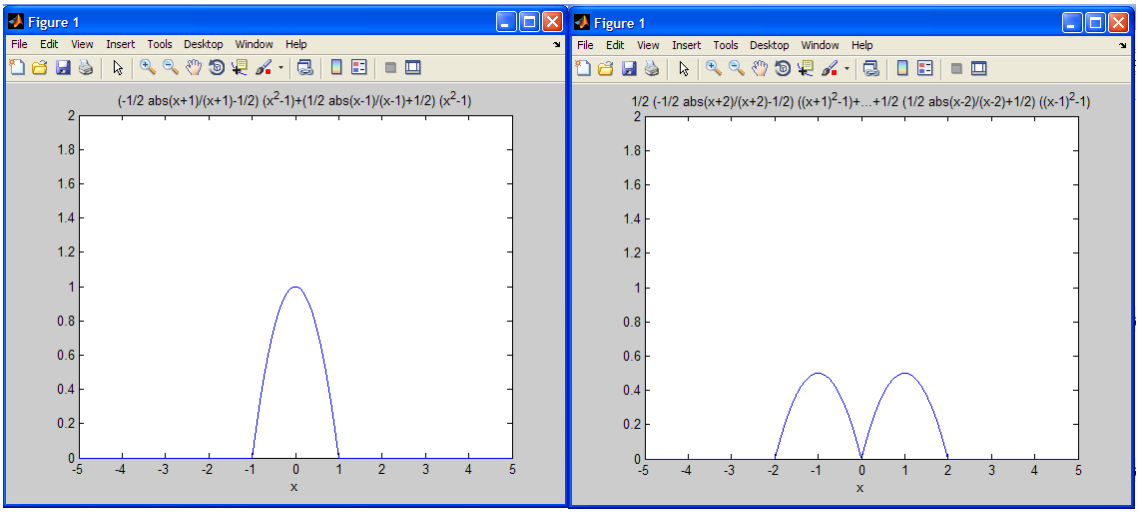
>> dalembert0708 *% f es el dato inicial*

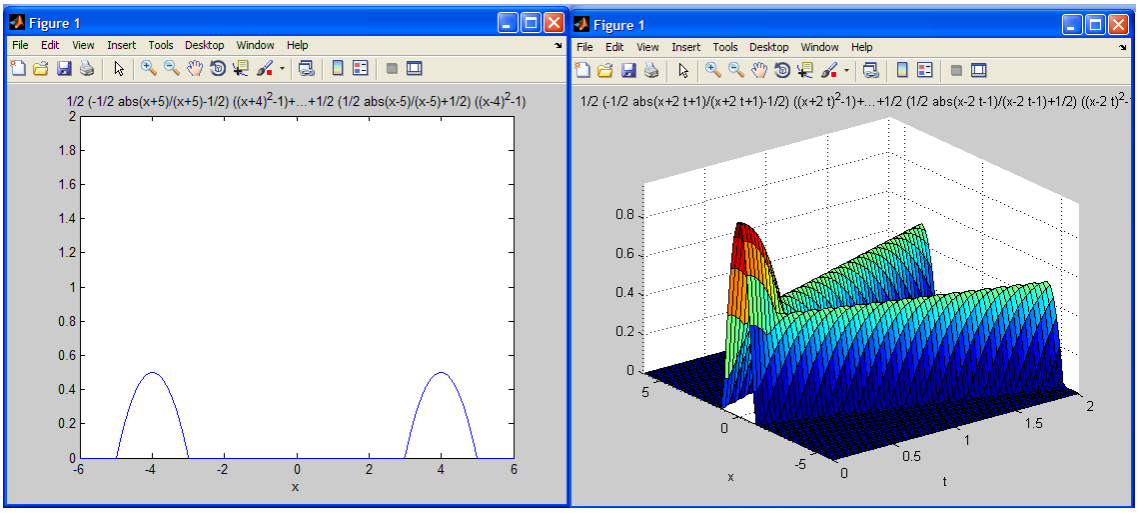
f=

-(((abs(x+1)/(x+1)) +1)/2) *(x^2-1)+(((abs(x-1)/(x-1)) +1)/2)*(x^2-1);

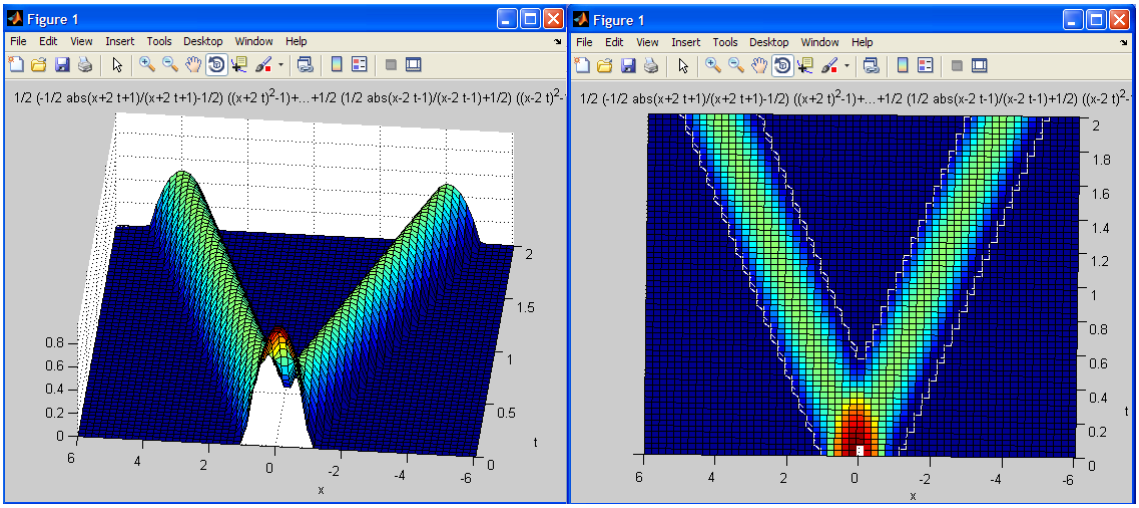
solucion =

1/2*(-1/2*abs(x+2*t+1)/(x+2*t+1)-1/2)*((x+2*t)^2-1)+1/2*(1/2*abs(x+2*t-1)/(x+2*t-1)+1/2)*((x+2*t)^2-1)+1/2*(-1/2*abs(x-2*t+1)/(x-2*t+1)-1/2)*((x-2*t)^2-1)+1/2*(1/2*abs(x-2*t-1)/(x-2*t-1)+1/2)*((x-2*t)^2-1)





% en la superficie vemos como lo que parece "la entrada de un túnel" se modifica: disminuye de altura con el tiempo y se ensancha durante un tiempo, y luego se divide en dos "túneles" que no vuelven a encontrarse nunca.

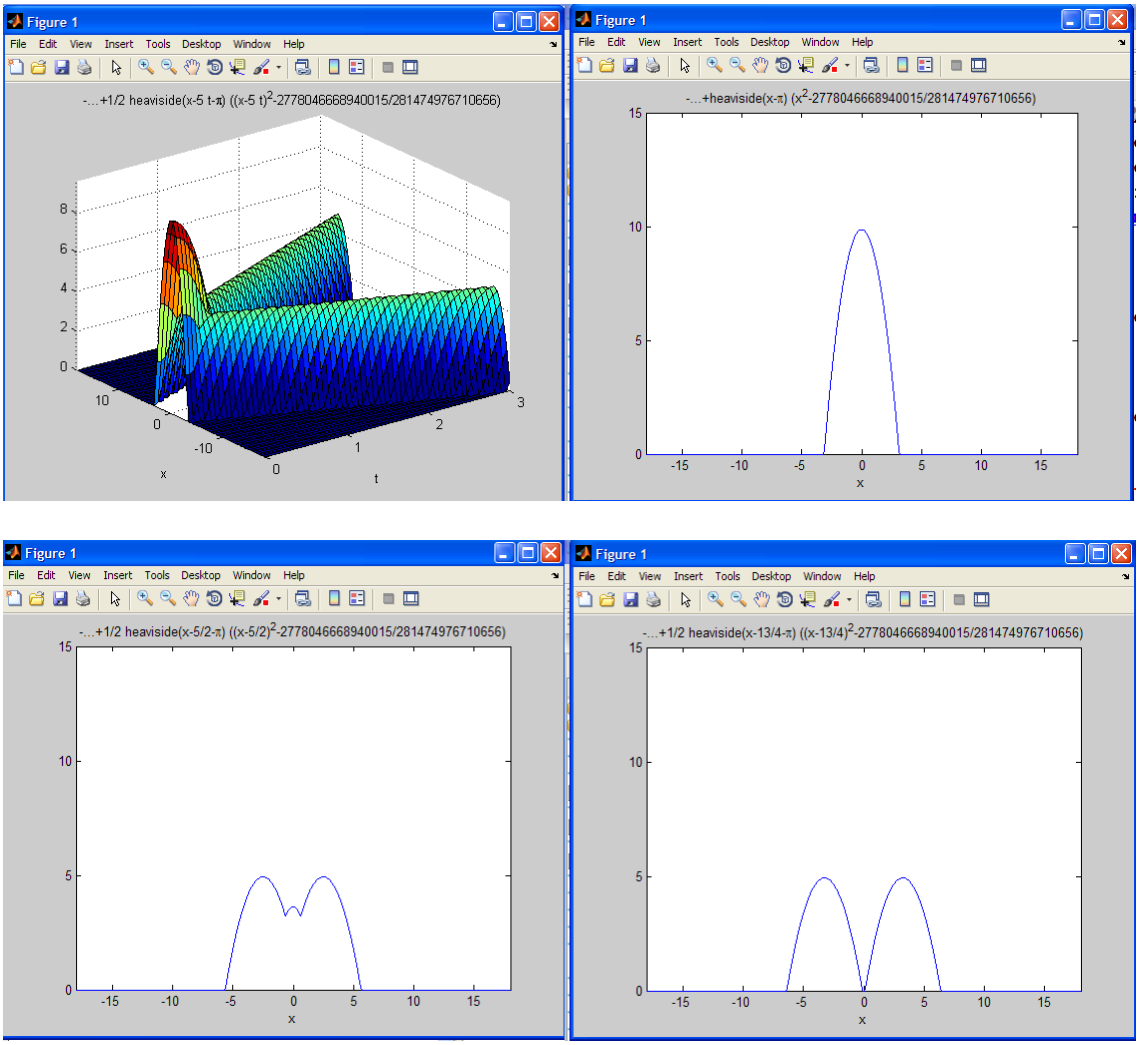


% este mismo efecto se tiene en dalembert1112, donde g=0 y f=(x^2-pi^2) en [-pi, pi] y cero fuera; esto es, el dato inicial está localizado en (-pi^2,pi^2)

>> dalembert1112 *% f es el dato inicial localizado en (-pi^2,pi^2)*

f =

heaviside(x - pi)*(x^2 - 2778046668940015/281474976710656) - heaviside(pi + x)*(x^2 - 2778046668940015/281474976710656)

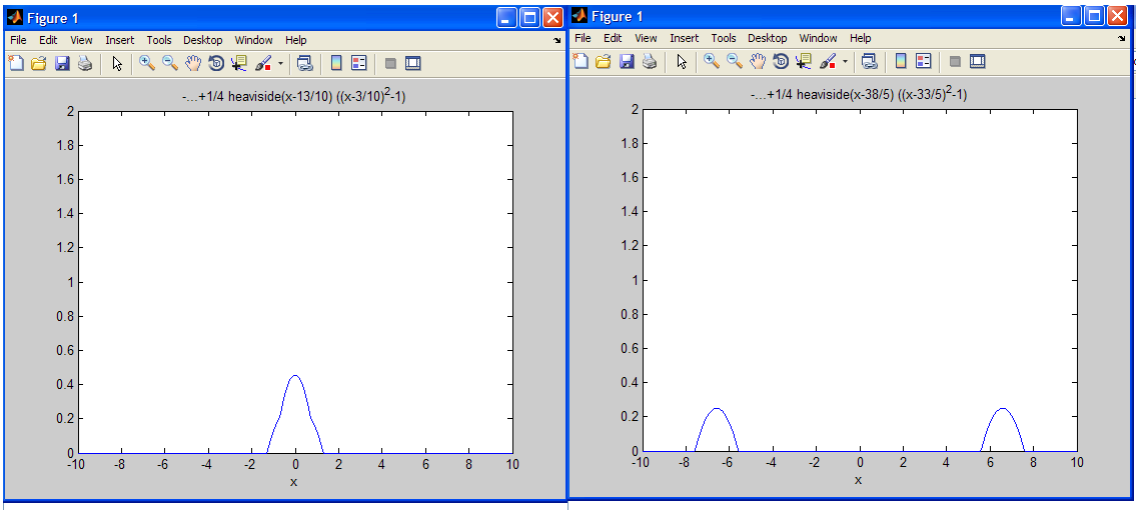


% El movimiento de la onda lo vemos ejecutando dalemmoviendoondassolo.m. Así podemos ver como rápidamente las dos ondas desaparecen de la ventana de observación y nunca más volveríamos a verlas a no ser que choquen contra algo como lo que se simula la función Matlab en reflejaondasdospuntos.m

>> dalemmoviendoondassolo ***% ver movie: dalemondas***

t = 0

t = 0.1000



0.3000

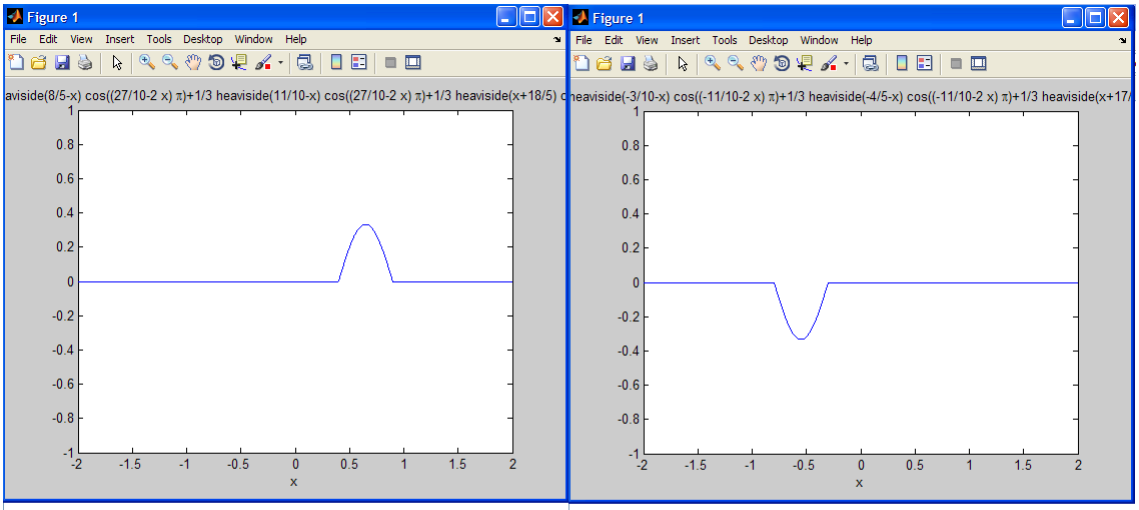
t =

6.6000

t =

>> reflejaondasdospuntos ***% ver movie: reflejaondas***

u =
heaviside(3/4-t)*(1/3*heaviside(x-t+1/4)*cos(2*(x-t)*pi)-1/3*heaviside(x-t-1/4)*cos(2*(x-t)*pi))+(1/3*heaviside(x-t+1/4)*cos(2*(x-t)*pi)-1/3*heaviside(x-t-1/4)*cos(2*(x-t)*pi)-1/3*heaviside(9/4-t-x)*cos(2*(2-t-x)*pi)+1/3*heaviside(7/4-t-x)*cos(2*(2-t-x)*pi))*heaviside(11/4-t)*heaviside(t-3/4)*heaviside(1-x)+(-1/3*heaviside(9/4-t-x)*cos(2*(2-t-x)*pi)+1/3*heaviside(7/4-t-x)*cos(2*(2-t-x)*pi)+1/3*heaviside(x+17/4-t)*cos(2*(x+4-t)*pi)-1/3*heaviside(x+15/4-t)*cos(2*(x+4-t)*pi))*heaviside(t-11/4)*heaviside(4-t)*heaviside(x+1)

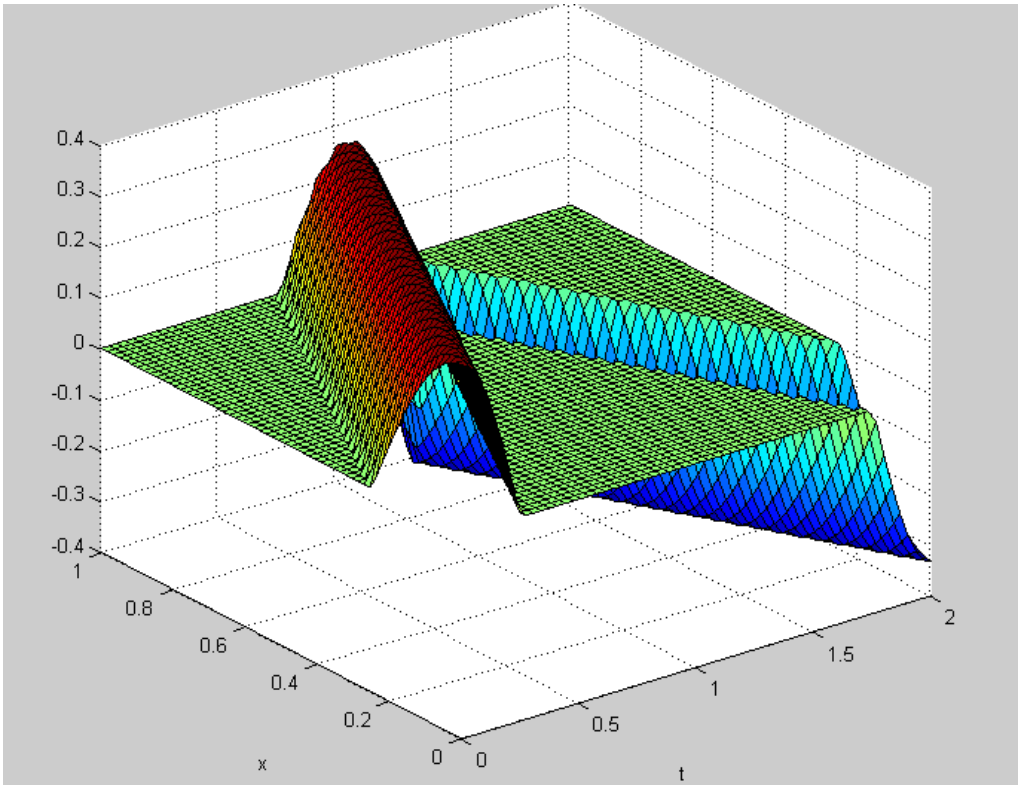



```
    0.6500

tdis =
```

```
    2.5500

tdis =
```

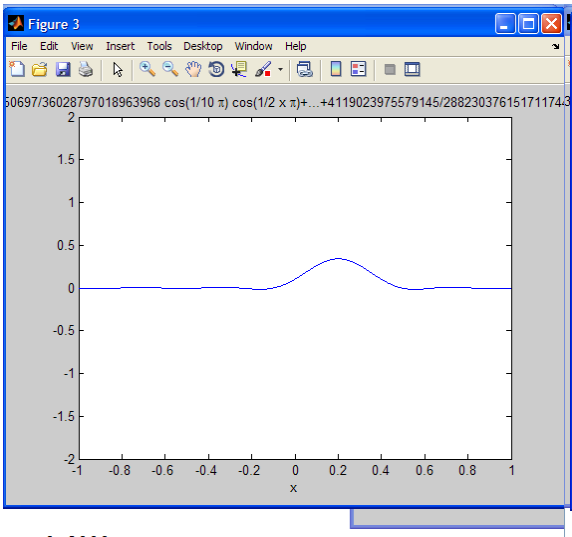
% La función u en reflejaondasdospuntos.m nos da la solución en un intervalo de tiempo, y las gráficas arriba son los cortes por planos t=dis de la superficie z=u(t,x) (gráfica de abajo)

>> ezsurf(-u,[0,2],[0,1])



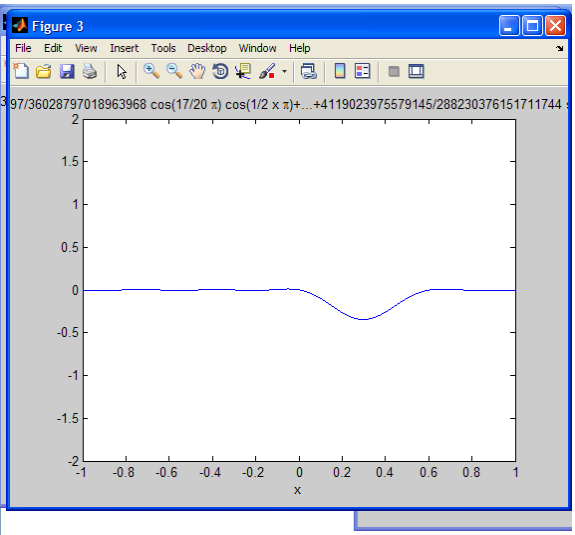
% la onda, localizada en el instante inicial, puede volver si choca contra algo como son los puntos x=1 y x=-1 en las figuras de arriba, de manera que va por arriba hasta que choca y vuelve por abajo hasta que choca de nuevo y así estaría indefinidamente. Esto se puede modelar con desarrollos en serie de Fourier (modelo de cuerda vibrante que tiene una longitud finita, sujeta en los extremos, con unos datos iniciales adecuados), si bien conviene observar que se trata de una aproximación de la solución por un número de términos de una serie

>> fourierrefleja(5) ***% ver movie: fourierefleja***



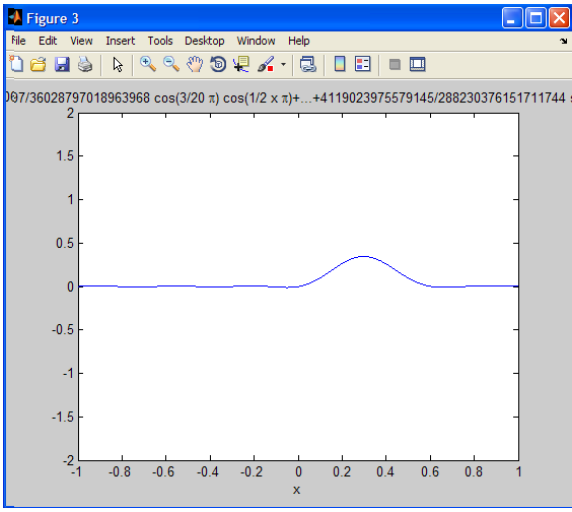
0.2000

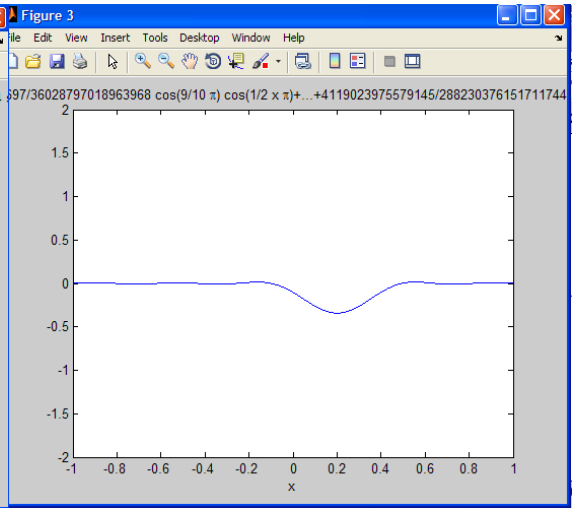
dt =



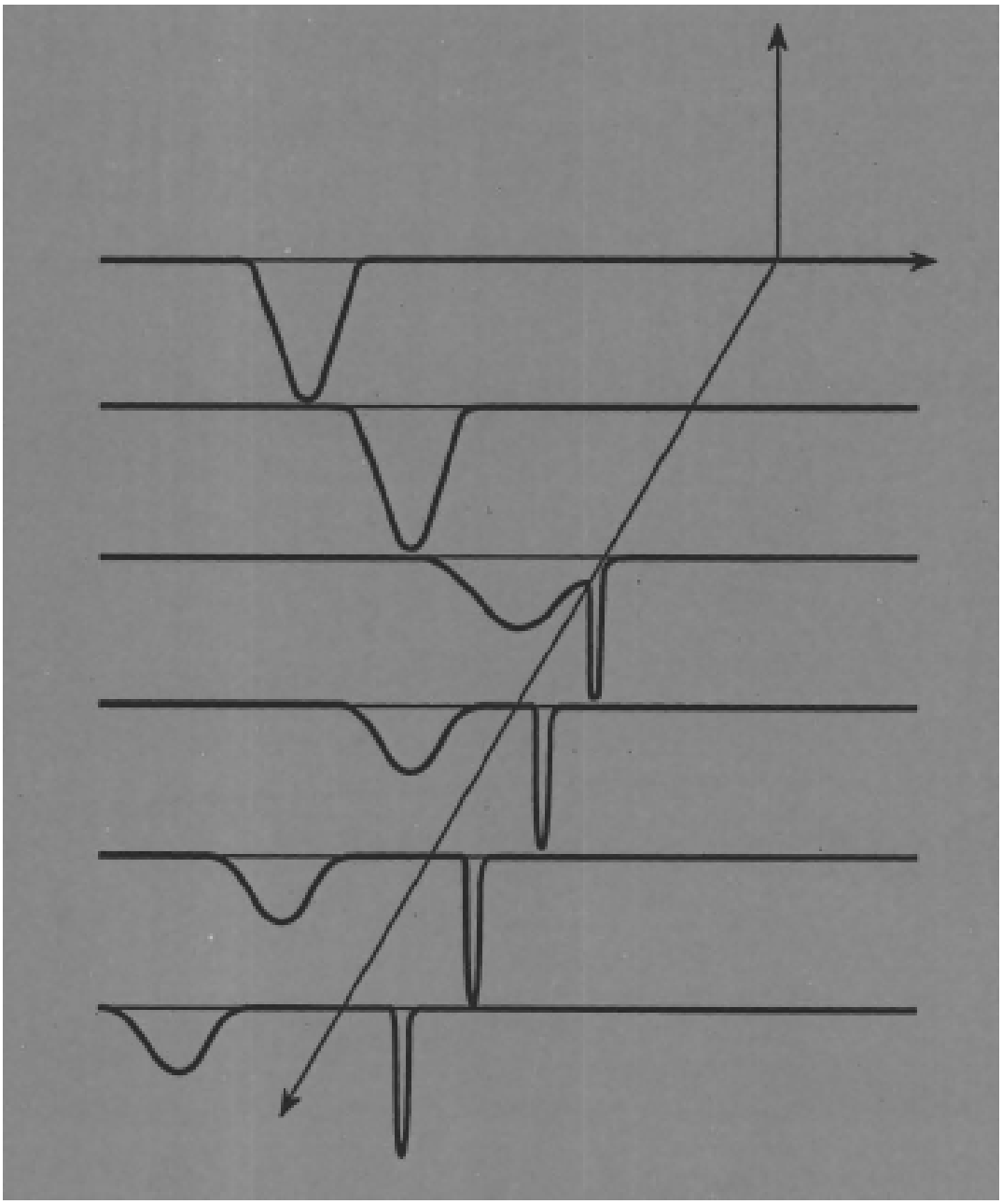
1.6000

dt =





% Como vemos abajo, el efecto observado con la perturbación localizada en las funciones "dalembert", tampoco se tendría si la onda choca contra un medio de distintas características físicas, de distinta densidad por ejemplo, como nos muestra la onda de la caratula del libro de apuntes

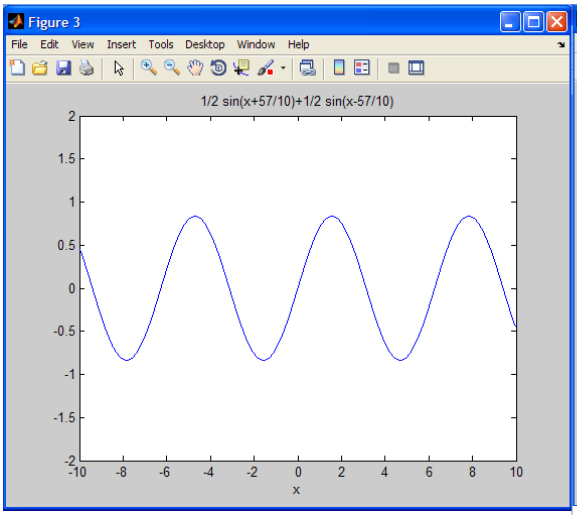


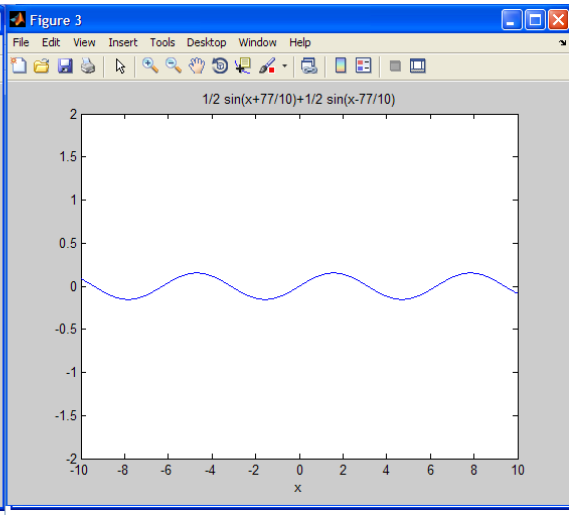
% Por otro lado, si la perturbación inicial no está localizada, pero la velocidad inicial es cero, el efecto de superposición de ondas que viajan en distintas direcciones sin cambio de forma se ve ejecutando dalemmoviendoondassolosenos.m: f=sin(x), g=0

>> dalemmoviendoondassolosenos ***% ver movie: dalemondasseno***

h =

sin(x)



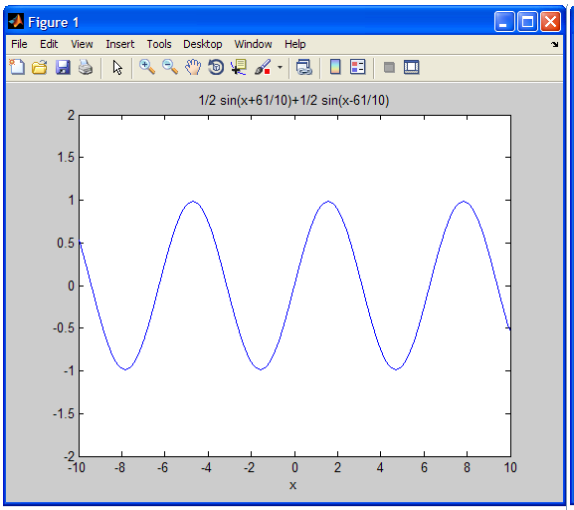


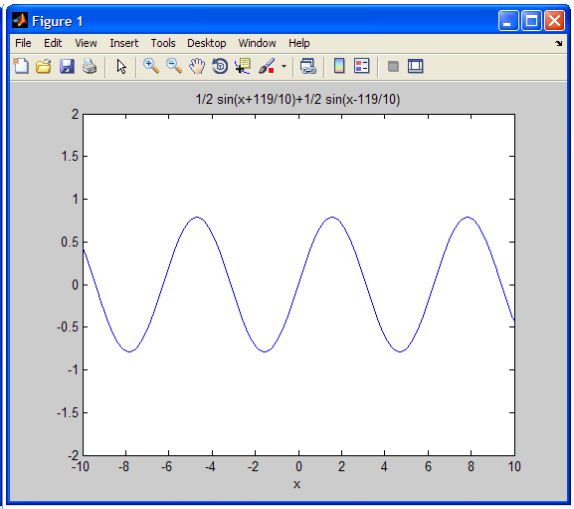

```
    5.7000
```

```
t =
```

```
    7.7000
```

```
t =
```





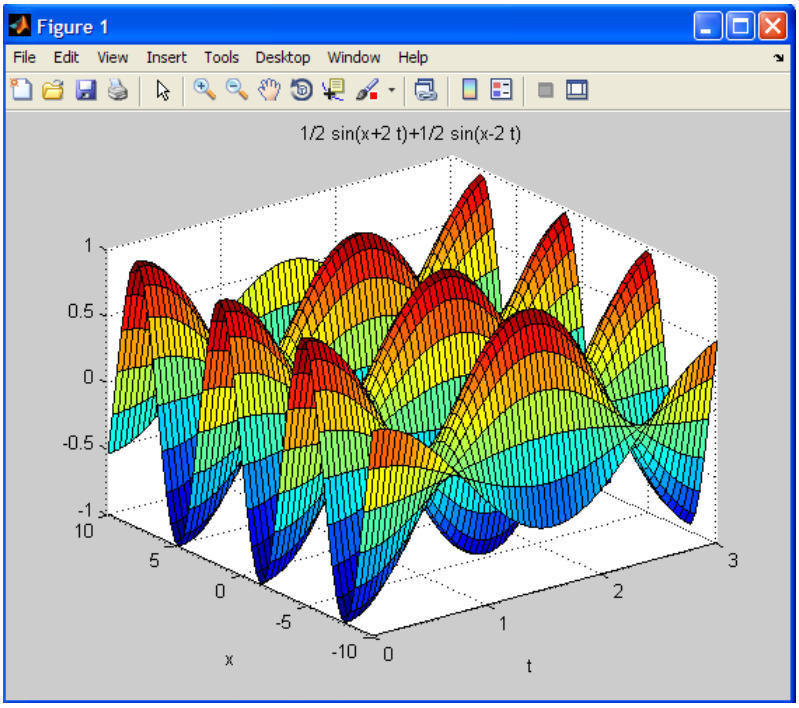

```
t =

    6.1000
```

```
t =

   11.9000
```

% El efecto que se observa ahora es que los puntos de la cuerda se mueven de arriba a abajo, o al revés, en un plano vertical, sin avanzar la onda. Esto es, no se observa propagación para f=sin(x), g=0, solo oscilaciones en un plano transversal, como consecuencia de la superposición de dos ondas con igual forma inicial e igual velocidad, que viajan en direcciones opuestas sin cambio de forma: "onda estacionaria"

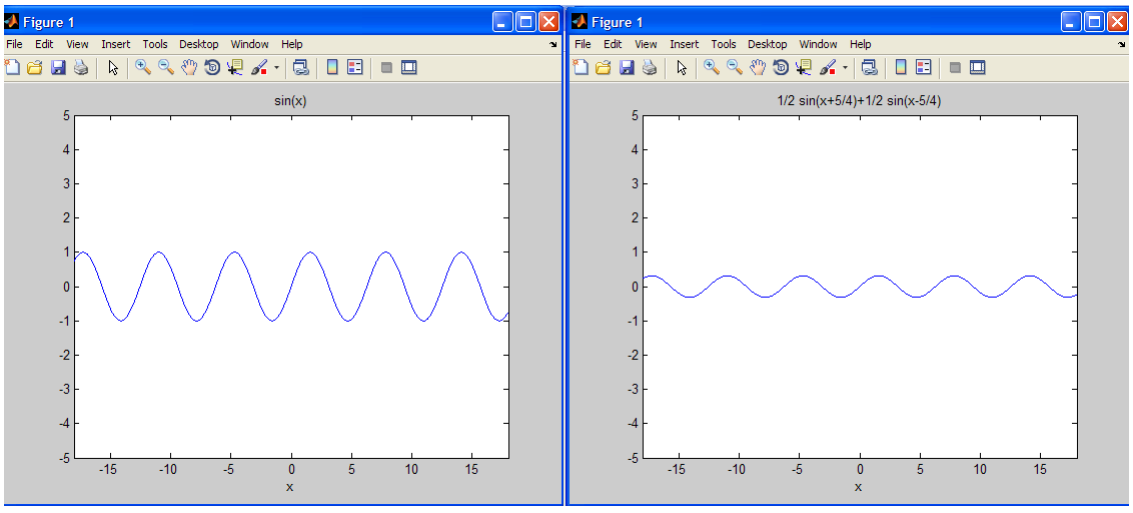
```
>> syms s t x

>> f=sin(s)

>> u=(subs(f,s,x+2*t)+subs(f,s,x-2*t))/2

u =

1/2*sin(x+2*t)+1/2*sin(x-2*t)

>> simple(u)

simplify:

sin(2*t + x)/2 - sin(2*t - x)/2


radsimp:


. . . . . .


ans =

cos(2*t)*sin(x)

% se tiene una onda estacionaria
```

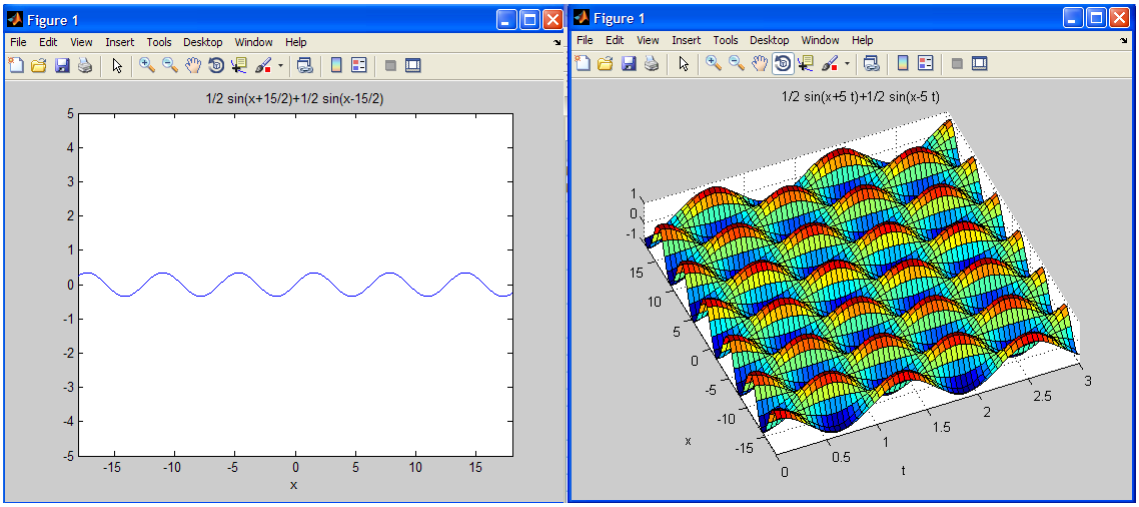
>> ezsurf(u,[0,3],[-10,10]) *% nos da la superficie solución para f=sin(x), g=0*



% Con f=sin(x), g=0, y la función Matlab dalembert1112s.m se tiene

>> dalembert1112s





% Aunque la superficie es parecida, el efecto de la propagación de ondas es distinto para f=sin(x), g=cos(x), c=2 (función dalemsenocoseno)

>> dalemsenocoseno ***% ver movie: dalemondassenocos***

f =

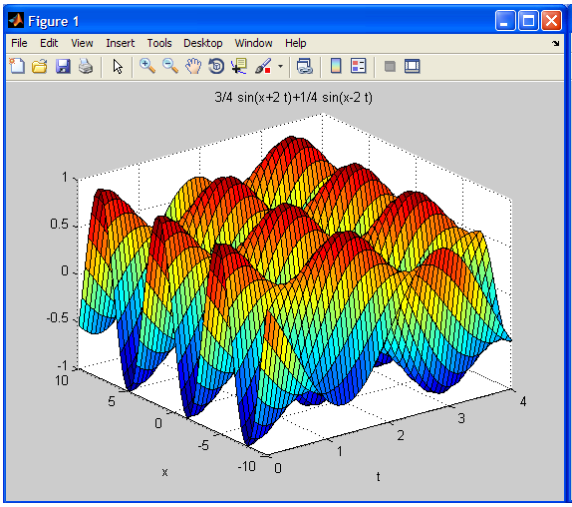
sin(x)

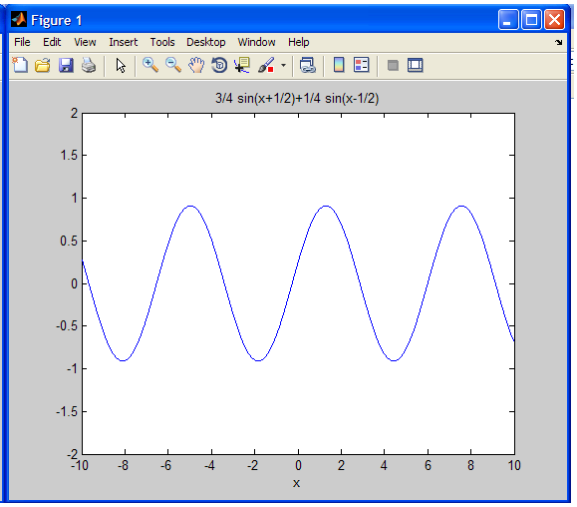
g =

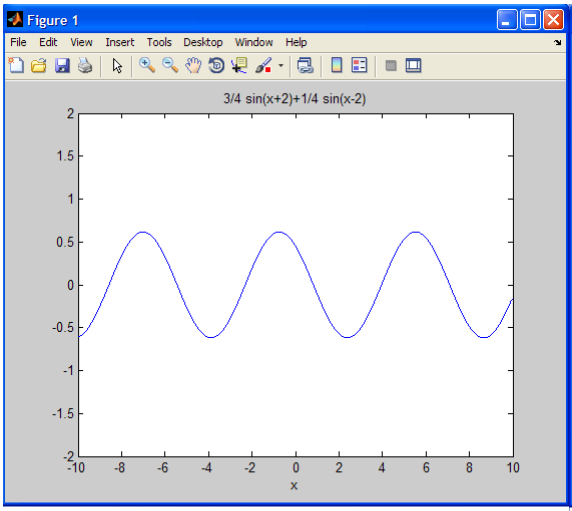
cos(x)

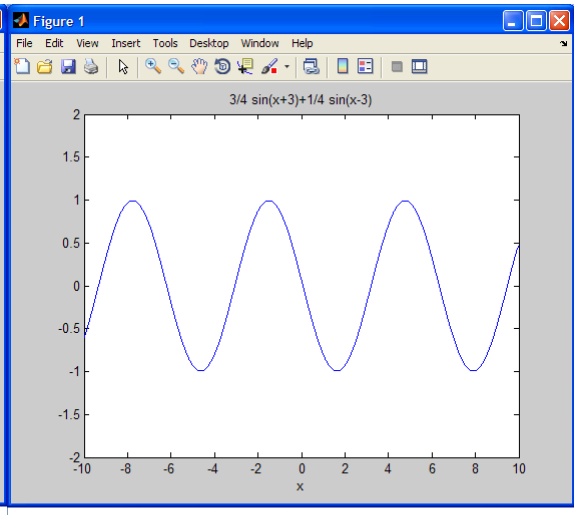
w =

3/4*sin(x+1)+1/4*sin(x-1)









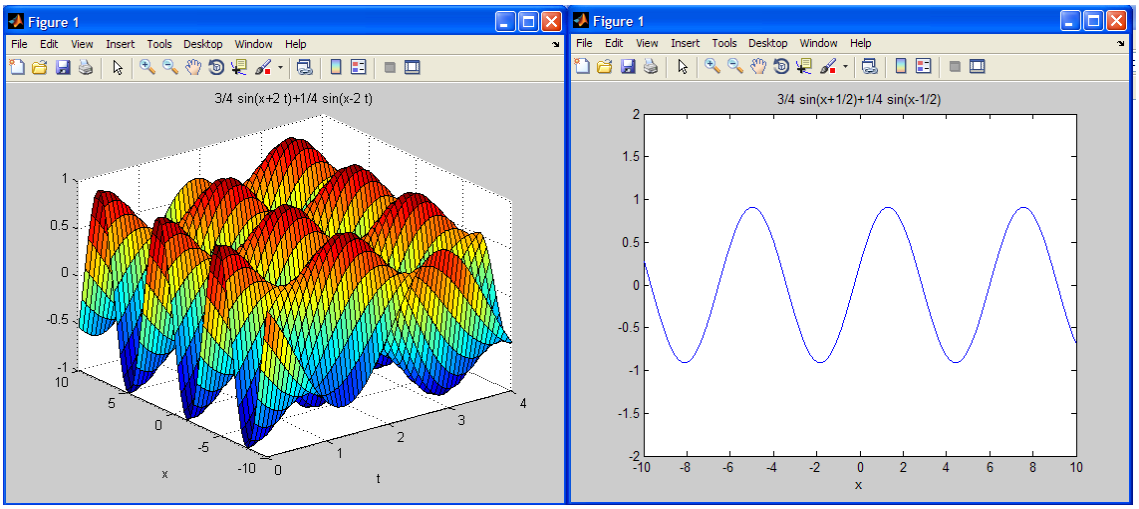

```
w =

3/4*sin(x+2)+1/4*sin(x-2)
```

```
w =

3/4*sin(x+3)+1/4*sin(x-3)
```

% abajo, para c=2, f=sin(x), g=cos(x), la fórmula de d'Alembert (distintas posibilidades para hacer la integral definida), y la gráfica de la superficie solución y de la forma de la onda en t=1

```
>> clear all

>> syms t x

>> f=sin(x)

f =

sin(x)

>> syms s % s variable para la integración, x variable espacial, t variable temporal

>> g=cos(s)

g =

cos(s)

>> u=(subs(f,x,x+2*t)+subs(f,x,x-2*t))/2+(1/4)*int(g,x-2*t,x+2*t) ; % la fórmula

u =

3/4*sin(x+2*t)+1/4*sin(x-2*t)

% también se puede definir

>> f=sin(s) ;

>> u=(subs(f,s,x+2*t)+subs(f,s,x-2*t))/2+(1/4)*int(g,x-2*t,x+2*t); % da el mismo
  resultado


>> ezsurf(u,[0,3],[-10,10])

% forma de la onda para t=1
```

>> ezplot(subs(u,t,1),[-10,10])



% el mismo resultado pero para f y g particularizados en la misma fórmula de d'Alembert se tiene con las instrucciones

>> clear all

>> syms s t x

>> u=(sin(x+2*t)+sin(x-2*t))/2+(1/4)*int(cos(s),s,x-2*t,x+2*t)

u =

sin(x - 2*t)/2 + sin(2*t + x)/2 + cos(t)*cos(x)*sin(t)

>> simplify(u)

ans =

(3*sin(2*t + x))/4 - sin(2*t - x)/4

% Los ficheros que se han ejecutado para EDP de segundo orden se pueden visualizar abajo usando type:

>> type dalembert0708

>> type dalembert1112

>> type dalembert1112s

>> type dalemmoviendoondassolo

>> type dalemmoviendoondassolosenos

% El proceso de difusión del calor del calor es bien distinto del de la propagación de las ondas. Lo comprobamos experimentalmente, simulando dicha difusión, en modelos 1-D, esto es, la difusión de calor en barras conductoras de longitud infinita. Así, consideramos el problema de valor inicial u_t-a^2u_xx=0, u(x,0)=f(x) ; a es una constante que depende de las características físicas de la barra, u(x,t) modela la temperatura del punto x en el tiempo t, cuando se sabe que en el instante de tiempo t=0 temperatura de cada punto x de la barra que es f(x).

% La solución nos la da la fórmula de Poisson ,
*u(x,t)= (1/(sqrt(pi*t)*2*a))*int(f*exp(-(x-s)^2/4*t*a^2),s,-inf ,inf),*
que explica fenómenos conocidos como el hecho de que la difusión del calor tenga "velocidad infinita". Tomamos temperaturas iniciales (f(x)) modeladas por funciones sinusoidales o localizadas en puntos o intervalos en el instante inicial t=0.

% Si por ejemplo a=1 y f(x)=sin(x), la solución se encuentra fácilmente por separación de variables: u(x,t)=sin(x)exp(-t). El dibujo de la solución nos muestra que la temperatura en cada punto de la barra tiende a anularse (esto es en unos puntos aumenta y en otros disminuye; hay puntos que permanecen a temperatura cero (x=kpi), pero de manera general la temperatura tiende a homogeneizarse en todos los puntos de la barra

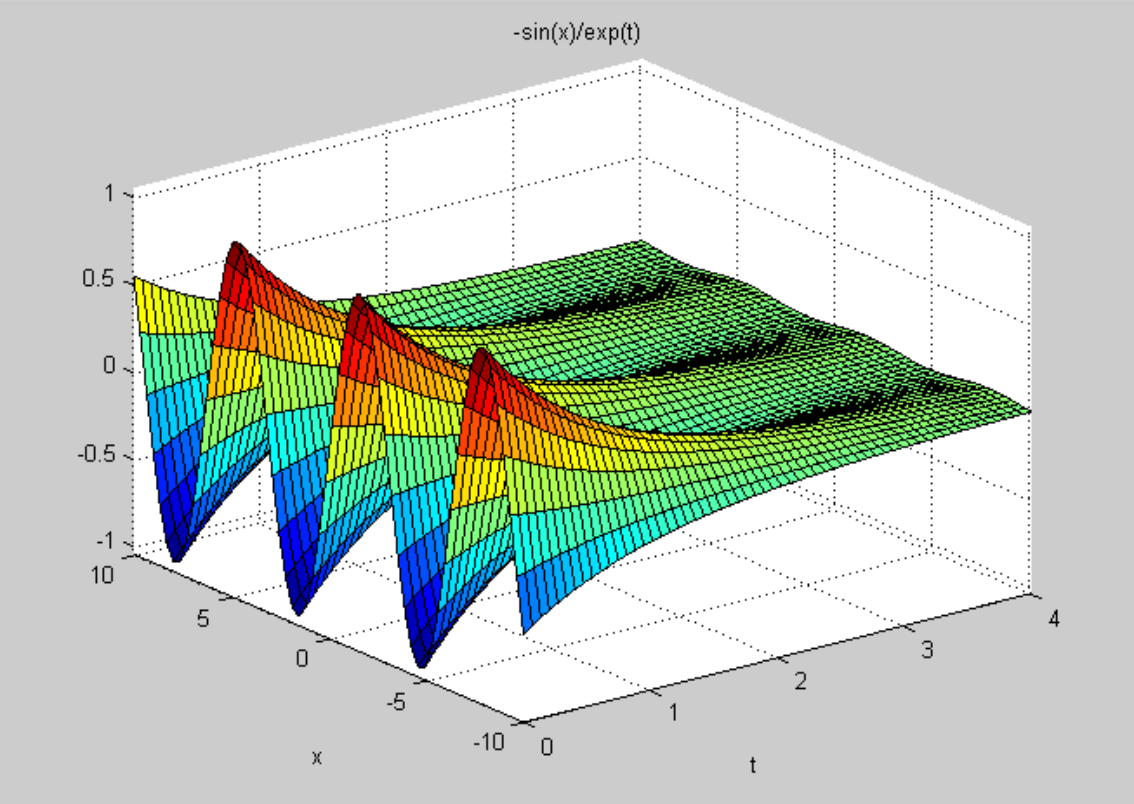
>> ezsurf(-u,[0,4],[-10,10])

>> u=sin(x)*exp(-t)

u =

sin(x)/exp(t)

>> ezsurf(-u,[0,4],[-10,10])



% la gráfica de la distribución de la temperatura para distintos tiempos se tiene con

>> ezplot(subs(u,t,0.2),[-10,10]) *% temperatura en t=0.2*

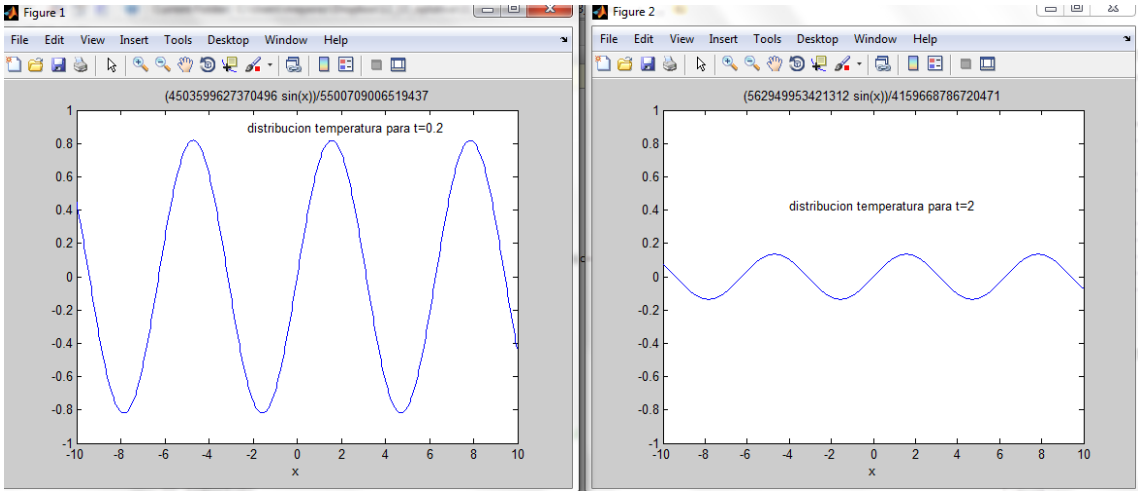
>> gtext('distribucion temperatura para t=0.2')

>> figure(2)

>> ezplot(subs(u,t,2),[-10,10]) *% temperatura en t=2*

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

>> gtext('distribucion temperatura para t=2')



% Si la distribución de la temperatura en el instante inicial está localizada, por ejemplo, en (-1,1) y modelada por la función triangular f(x)=x+1 para x en [-1,0], f(x)=1-x para x en [0,1]; la solución nos la da la fórmula de Poisson. Tomamos a=1 y definimos f

>> syms s x t

>> f=(heaviside(s+1)-heaviside(s))*(s+1)+(heaviside(s)-heaviside(s-1))*(1-s)

f =

(s - 1)*(heaviside(s - 1) - heaviside(s)) + (s + 1)*(heaviside(s + 1) - heaviside(s))

>> Poisson=(1/(sqrt(pi*t)*2))*int(f*exp(-(x-s)^2/(4*t)),s,-1,1) *% version 2011 de Matlab*

Poisson =

((pi^(1/2)*x*erfi((x*(-1/t)^(1/2))/2 - (-1/t)^(1/2)/2))/(-1/t)^(1/2) -
(pi^(1/2)*(erf(((1/t)^(1/2)*(x - 1))/2) - erf(((1/t)^(1/2)*(x + 1))/2)))/(1/t)^(1/2) -
(4*t)/exp(x^2/(4*t)) + (pi^(1/2)*x*erfi((x*(-1/t)^(1/2))/2 + (-1/t)^(1/2)/2))/(-1/t)^(1/2)
- (2*pi^(1/2)*x*erfi((x*(-1/t)^(1/2))/2))/(-1/t)^(1/2) +
(2*t)/(exp(1/(4*t))*exp(x^2/(4*t))*exp(x/(2*t))) +
(2*t*exp(x/(2*t)))/(exp(1/(4*t))*exp(x^2/(4*t))))/(2*pi^(1/2)*t^(1/2))

% no hace integrales y nos lo deja en términos de la función erfi

>> pretty(Poisson)

```
/                            /    / / 1 \1/2         \      / / 1 \1/2         \ \
|                            |    | | - |    (x - 1) |      | | - |    (x + 1) | |
|                          1/2 |    | \ t /            |      | \ t /            | |
|   1/2                   pi   | erf| ---------------- | - erf| ---------------- | |
| pi    x erfi(#1 - #3)        \    \        2         /      \         2        / /   4 t
| --------------------- - ------------------------------------------------------------ - --- +
|      /   1 \1/2                                 / 1 \1/2                              #2
|      | - - |                                    | - |
\      \   t /                                    \ t /

                                                                      \
                                                                      |
                                                                      |
    1/2                         1/2                                   |
   pi    x erfi(#1 + #3)   2 pi    x erfi(#1)     2 t      2 t #4 |        1/2  1/2
   ---------------------- - ------------------ + -------- + ------ | / (2 pi    t   )
        /   1 \1/2              /   1 \1/2       #5 #2 #4   #5 #2  |
        | - - |                 | - - |                          |
        \   t /                 \   t /                          /

where

        /   1 \1/2
      x | - - |
        \   t /
  #1 = ------------
             2

          /  2  \
          | x   |
  #2 = exp| --- |
          \ 4 t /

       /   1 \1/2
       | - - |
       \   t /
  #3 = ----------
             2

          /  x  \
  #4 = exp| --- |
          \ 2 t /

          /  1  \
  #5 = exp| --- |
          \ 4 t /
```

>> help erfi

erfi not found.

Use the Help browser search field to search the documentation, or

type "help help" for help command options, such as help for methods.

% help no proporciona ayuda sobre la función erfi; se puede ver la definición, por ejemplo en http://en.wikipedia.org/wiki/Error_function.
Evaluaciones de esta variable (función) Poisson en puntos puede ayudarnos a representar, pero la gráfica de la función con ezsurf da error como se ve abajo

>> subs(Poisson,{x,t},{1,1})

ans =

(2/exp(1) - 4/exp(1/4) + pi^(1/2)*erf(1) - pi^(1/2)*erfi(i)*i + pi^(1/2)*erfi(i/2)*2*i + 2)/(2*pi^(1/2))

>> vpa(ans,5)

ans =

0.21516

>> ezsurf(Poisson,[0.01,10],[-10,10])

Error using inlineeval (line 15)
Error in inline expression ==> ((pi.^(1./2).*x.*erfi((x.*(-1./t).^(1./2))./2 - (-1./t).^(1./2)./2))./(-1./t).^(1./2) -
……..
Undefined function 'erfi' for input arguments of type 'double'.

Error in inline/feval (line 34)
INLINE_OUT_ = inlineeval(INLINE_INPUTS_, INLINE_OBJ_.inputExpr, INLINE_OBJ_.expr); ……..
……..

% Todo esto depende de la versión de Matlab. Arriba se ha utilizado Matlab 2011, pero la versión de 2008 nos lo deja indicado en términos de la función erf; esto es, tampoco hace las integrales de la fórmula de Poisson, pero se trata de integrales tabuladas o que calcula numéricamente, y podemos dibujar la distribución de la temperatura

>> syms s x t

>> f=(heaviside(s+1)-heaviside(s))*(s+1)+(heaviside(s)-heaviside(s-1))*(1-s)

f =

(heaviside(s+1)-heaviside(s))*(s+1)+(heaviside(s)-heaviside(s-1))*(1-s)

>> Poisson=(1/(sqrt(pi*t)*2))*int(f*exp(-(x-s)^2/(4*t)),s,-1,1) *% con Matlab 2008*

Poisson =

1/2/(pi*t)^(1/2)*(2*t*exp(-1/4/t)*exp(1/2*x/t)*exp(-1/4*x^2/t)-2*x*t^(1/2)*pi^(1/2)*erf(1/2*x/t^(1/2))+x*t^(1/2)*pi^(1/2)*erf(1/2*(x-1)/t^(1/2))+2*t*exp(-1/4/t)*exp(-1/2*x/t)*exp(-1/4*x^2/t)+x*t^(1/2)*pi^(1/2)*erf(1/2*(x+1)/t^(1/2))-pi^(1/2)*t^(1/2)*erf(1/2*(x-1)/t^(1/2))+pi^(1/2)*t^(1/2)*erf(1/2*(x+1)/t^(1/2))-4*t*exp(-1/4*x^2/t))

>> pretty(Poisson)

```
     /                                          2
     |                                         x
1/2  |2 t exp(- 1/4 1/t) exp(1/2 x/t) exp(- 1/4 ----)
     |                                         t
     \

              1/2   1/2           x          1/2   1/2          -1 + x
    - 2 x t     pi    erf(1/2 ----) + x t     pi    erf(1/2 ------)
                                1/2                          1/2
                               t                            t

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

         1/2   1/2          1 + x       1/2  1/2          -1 + x
    + x t     pi    erf(1/2 -----) - pi     t    erf(1/2 ------)
                             1/2                          1/2
                            t                            t

                                                2  \
        1/2  1/2          1 + x                 x   |    /        1/2
    + pi     t    erf(1/2 -----) - 4 t exp(- 1/4 ----)|   /   (pi t)
                             1/2                  t   | /
                            t                        /
```

>> help erf

ERF Error function.

Y = ERF(X) is the error function for each element of X. X must be real. The error function is defined as:

*erf(x) = 2/sqrt(pi) * integral from 0 to x of exp(-t^2) dt*

……

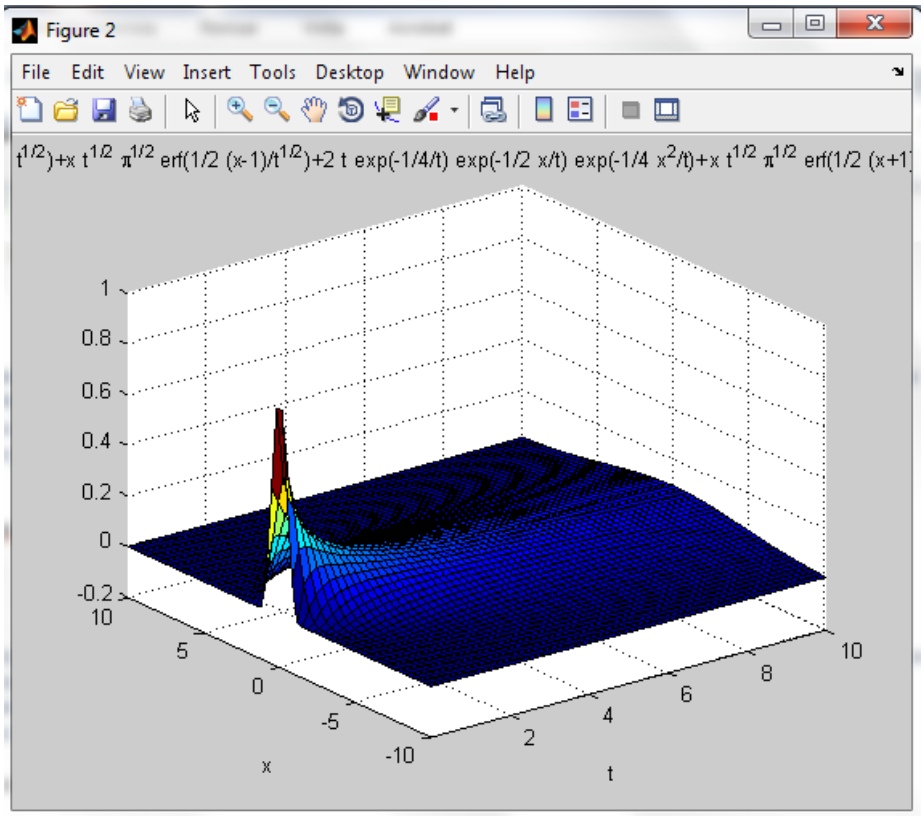
% La evaluación en el punto (1,1) nos da lo mismo en ambas versiones:

>> subs(Poisson,{x,t},{1,1}); vpa(ans,5)

ans =

0.21516

>> ezsurf(Poisson,[0.01,10],[-10,10])



% la distribución de temperaturas en distintos tiempos nos lo da el corte por planos t=constante. Nos muestra como todos los puntos de la barra tienden a enfriarse: esto es, a temperatura 0

>> ezplot(subs(Poisson,t,0.2),[-10,10])

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

>> gtext('distribucion de la temperatura en t=0.2')

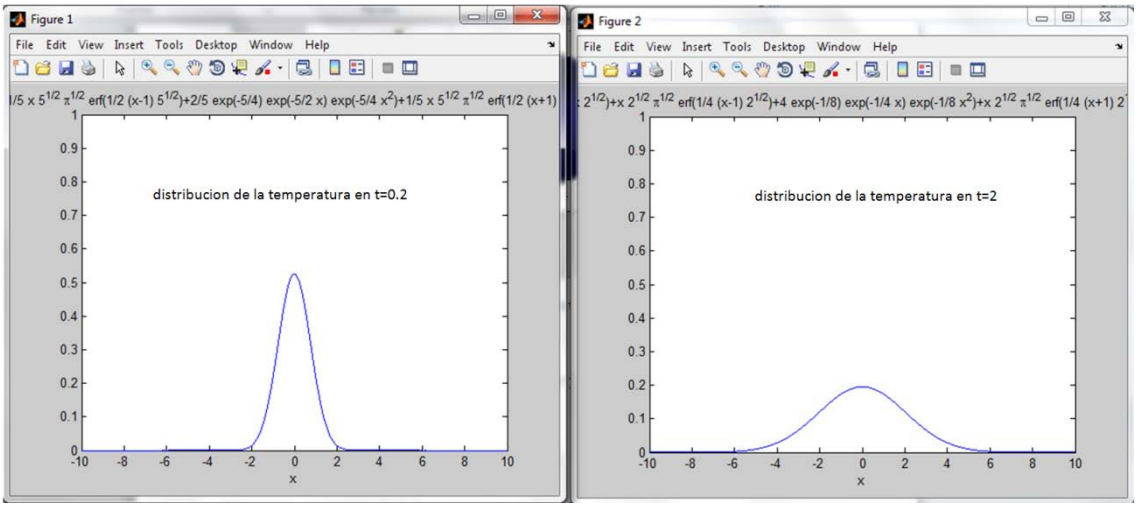
>> ezplot(subs(Poisson,t,2),[-10,10])

>> figure(2)

>> ezplot(subs(Poisson,t,2),[-10,10])

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

>> gtext('distribucion de la temperatura en t=2')



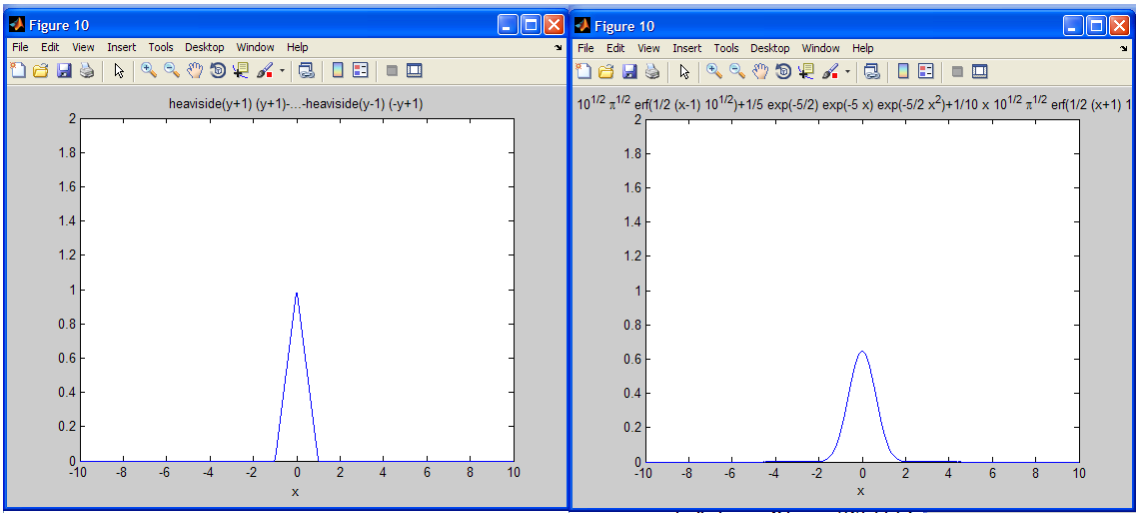
% Ejecutamos las funciones Matlab calor1.m y calorsurface.m que nos proporcionan unas gráficas análogas a las de las ondas, pero con significado físico bien distinto del de la propagación de las ondas; lo que se visualiza modela la variación de la temperatura con el tiempo. Los datos iniciales son f=sin(x) y f con soporte localizado en [-1,1]

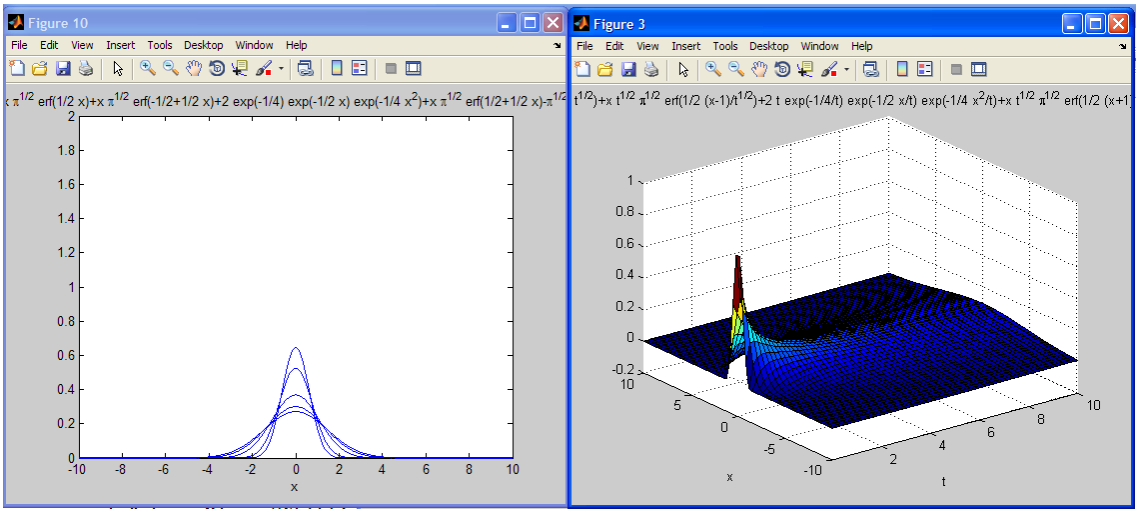
>> calorsurface ***% ver movie: calorpoiss***

% Para f(x) "localizada" en el intervalo [-1,1], vemos cómo a medida que avanza el tiempo la temperatura va disminuyendo en estos tiempos, en este intervalo, pero aumenta considerablemente en puntos cercanos a [-1,1], tendiendo a homogeneizarse en todos puntos. La superficie solución nos refleja bien la situación, bien distinta de la de las ondas: "la entrada del túnel" se va ensanchando a medida que disminuye su altura hasta llegar casi a cero.

solucion F. Poisson =

$$
\begin{aligned}
&\int_0^{1/2} 2\,t\,\exp\left(-\tfrac{1}{4}\,\frac{1}{t}\right)\exp\left(\tfrac{1}{2}\,\frac{x}{t}\right)\exp\left(-\tfrac{1}{4}\,\frac{x^2}{t}\right)\,dt \\
&\quad -2\,x\,t^{1/2}\,\pi^{1/2}\,\operatorname{erf}\left(\tfrac{1}{2}\,\frac{x}{t^{1/2}}\right) + x\,t^{1/2}\,\pi^{1/2}\,\operatorname{erf}\left(\tfrac{1}{2}\,\frac{-1+x}{t^{1/2}}\right) \\
&\quad +2\,t\,\exp\left(-\tfrac{1}{4}\,\frac{1}{t}\right)\exp\left(-\tfrac{1}{2}\,\frac{x}{t}\right)\exp\left(-\tfrac{1}{4}\,\frac{x^2}{t}\right) \\
&\quad +x\,t^{1/2}\,\pi^{1/2}\,\operatorname{erf}\left(\tfrac{1}{2}\,\frac{1+x}{t^{1/2}}\right) - \pi^{1/2}\,t^{1/2}\,\operatorname{erf}\left(\tfrac{1}{2}\,\frac{-1+x}{t^{1/2}}\right) \\
&\quad +\pi^{1/2}\,t^{1/2}\,\operatorname{erf}\left(\tfrac{1}{2}\,\frac{1+x}{t^{1/2}}\right) - 4\,t\,\exp\left(-\tfrac{1}{4}\,\frac{x^2}{t}\right)\left|\int_0^{(\pi\,t)^{1/2}}\right.
\end{aligned}
$$





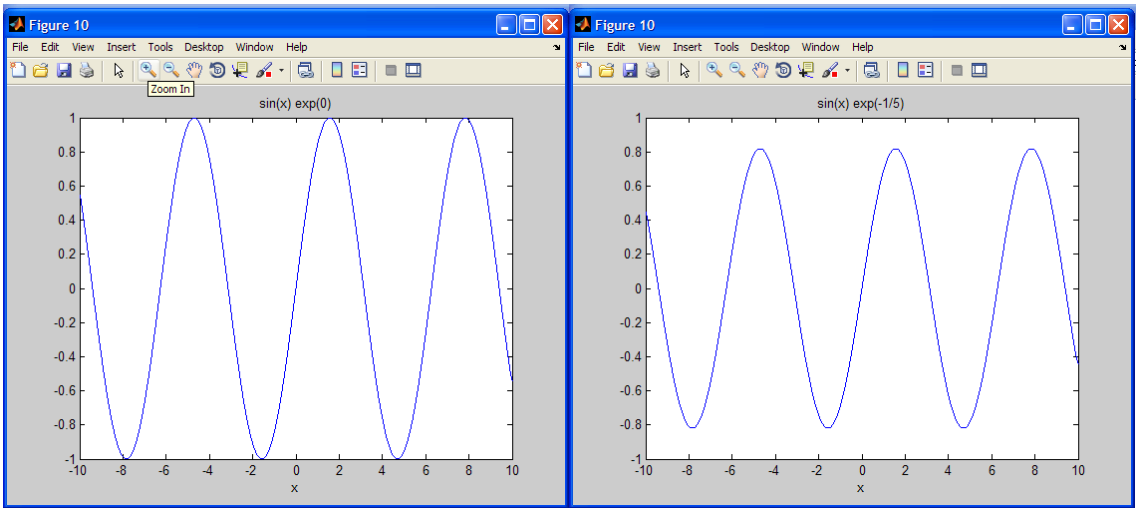
% Un proceso bien distinto se observa ejecutando calor1.m cuando temperatura dada en t=0, está distribuida a lo largo de toda la barra, por ejemplo f=sin(x), o f=cos(x) como se ha visto antes. Así vemos la distribución de la temperatura un tiempo inicial, y pasado otro tiempo, la temperatura de la barra tiende a homogeneizarse a cero, es decir, los puntos a temperatura inicial 1, se enfrían y los de temperatura -1 se calientan hasta llegar en todos a temperatura 0. Obviamente, hay puntos de temperatura 0 siempre como consecuencia de que la solución es u=sin(x)exp(-t)

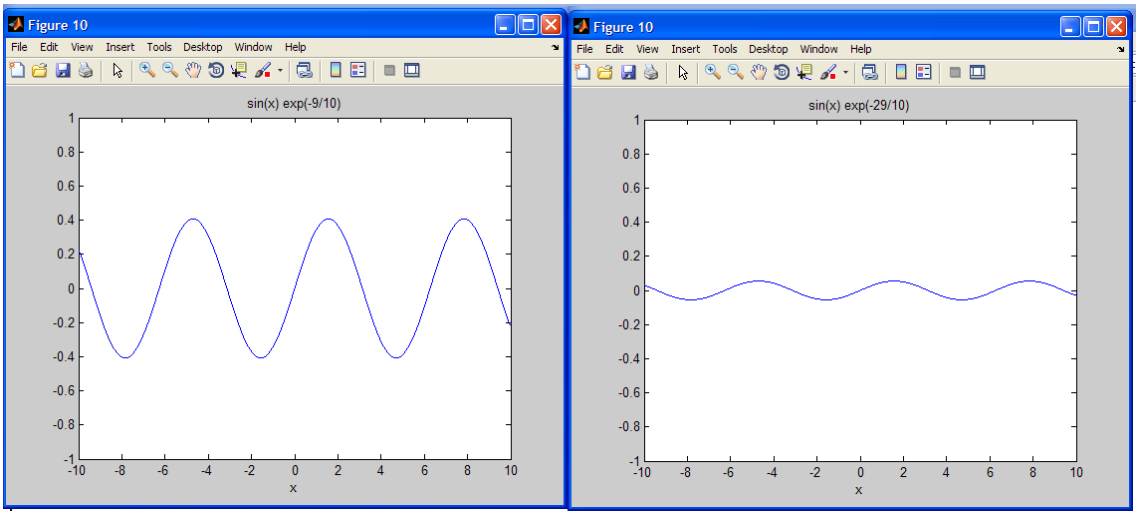
>> calor1 ***% ver movie: calor1***

h =

sin(x)*exp(-s)

% h solución





% Las gráficas muestran la temperatura de la barra en distintos tiempos; a medida que t crece la temperatura se va haciendo cero en todos los puntos

>> type calor1

>> type calorsurface

% Un proceso distinto es el que se tiene cuando la barra se somete a una fuente de calor en el tiempo, es decir, la temperatura inicial es cero pero u_t-u_xx=cos(x) como se plantea en el problema que sigue. No se puede esperar que la temperatura decrezca a 0, sino que se estabilice en cos(x) en cada punto de la barra. La solución particular de la EDP no homogénea se encuentra, primero, por separación de variables y luego, una vez homogeneizado, se procede de nuevo por separación de variables o mediante la fórmula de Poisson

% Solución del problema u_t-u_xx=cos(x), u(x,0)=0

>> clear all

>> syms t x

>> u=-cos(x)*exp(-t)+cos(x)

u =

-cos(x)*exp(-t)+cos(x)

% verificación de EDP u_t-u_xx=cos(x)

>> diff(u,t)-diff(u,x,2)-cos(x)

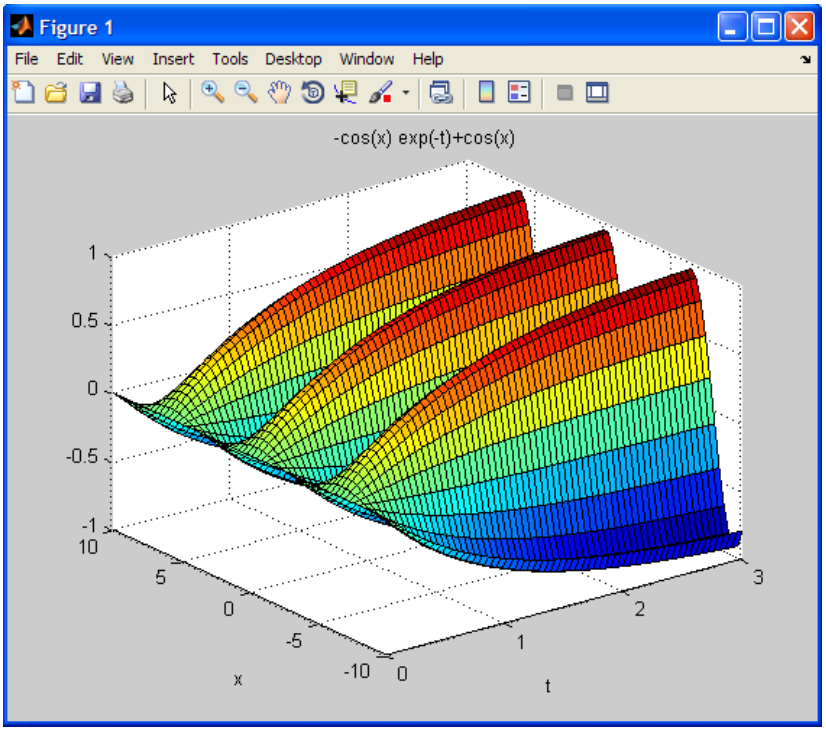
ans =0

% verificación de condición inicial u(x,0)=0

>> subs(u,t,0)

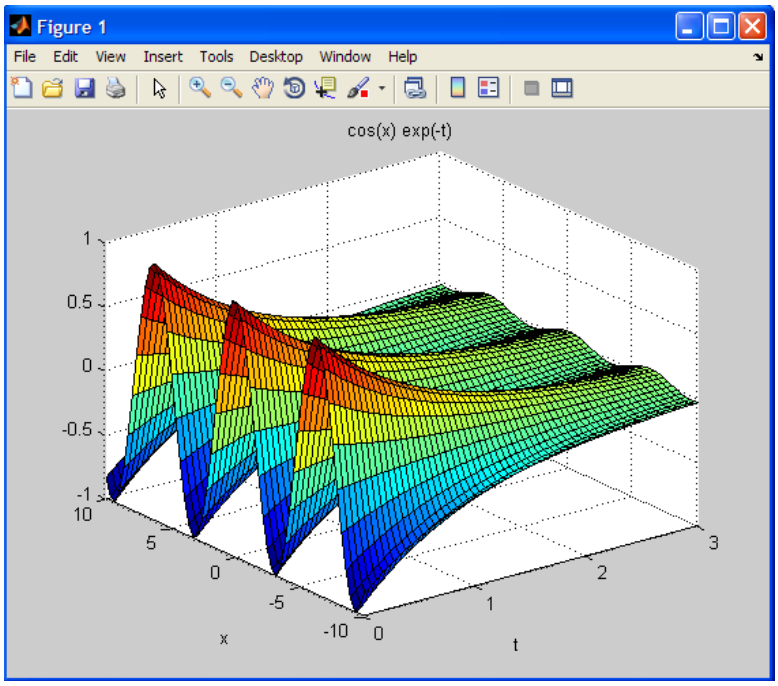
ans = 0

>> ezsurf(u,[0,3],[-10,10])

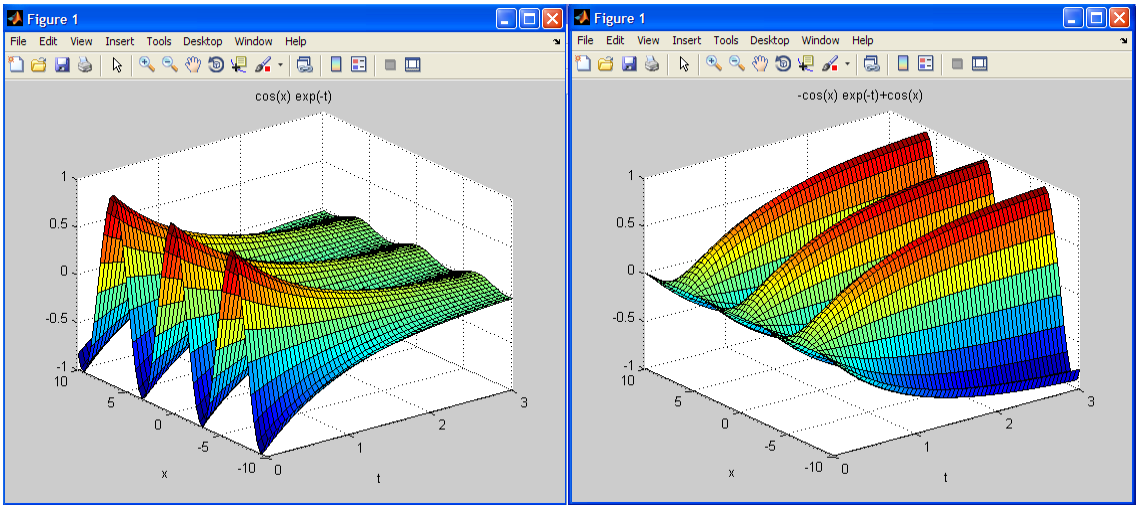


% vemos que la temperatura de cada punto x tiende a estabilizarse en cos(x) cuando el tiempo crece. Comparamos con el caso de temperatura en x dada por cos(x) para t=0

>> ezsurf(cos(x)*exp(-t),[0,3],[-10,10])



% Se comparan las dos para ver los distintos efectos,



Modelos de vibraciones de cuerdas y vigas: problemas mixtos para EDP de segundo orden, y de cuarto orden.

EDP Cuarto orden: Modelo de las vibraciones transversales de una viga, con extremos simplemente soportados

$$\begin{cases} u_{tt} + u_{xxxx} = 0\,, x \in (0,\pi), t > 0\,, \\ u(x,0) = \sin 2x\,,\ x \in [0,\pi]\,, \\ u_t(x,0) = 3\sin 2x\,,\ x \in [0,\pi] \\ u(0,t) = u(\pi,t) = 0\,, t \geq 0\,, \\ u_{xx}(0,t) = u_{xx}(\pi,t) = 0\,, t \geq 0\,. \end{cases}$$

Solución: Métododo de separación de variables.

$$u(x,t) = sin(2x)(\cos(4t) + \frac{3}{4}\sin(4t))$$

*% Se supone que la viga en estado de equilibrio se encuentra en el intervalo [0,pi], se desplaza de su posición de equilibrio en el tiempo t=0, toma la forma de la función sin(2*x), y empieza a vibrar cada punto con una velocidad 3sin(2*x). Los extremos se denominan simplemente soportados (ejercicio 20 de la Seccion 7.5 del libro de apuntes). Ejecutamos el fichero vibraviga.m donde tenemos instrucciones para que nos muestre la gráfica de la superficie solución y de la forma de la viga en distintos tiempos. El movimiento se ve ejecutando moviendoviga.m*

>> vibraviga

u =

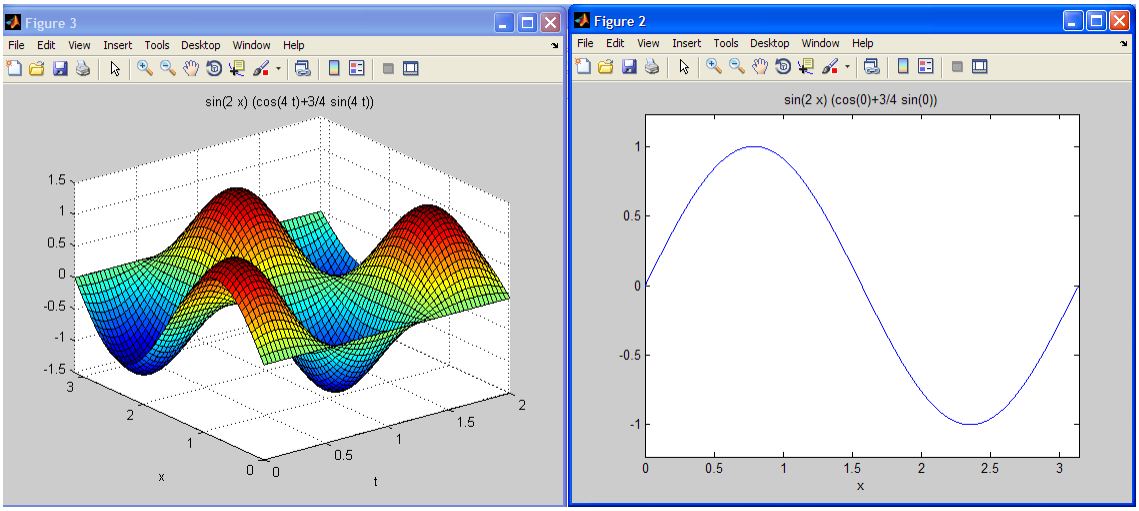
sin(2*x)*(cos(4*t)+3/4*sin(4*t))

w1 =

sin(2*x)

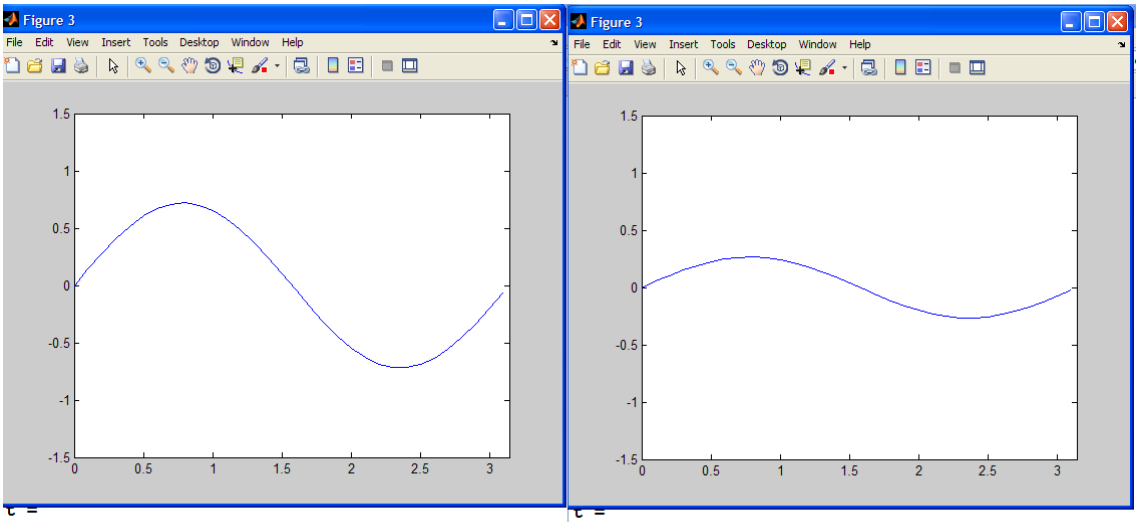
w2 =

sin(2*x)*(cos(4)+3/4*sin(4))



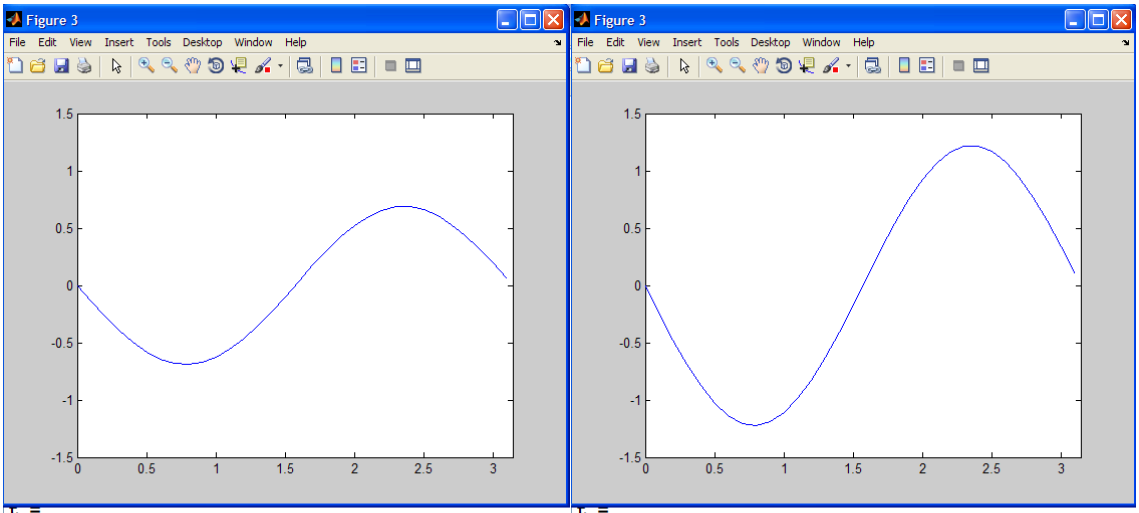
% La gráfica nos muestra la superficie solución y la forma inicial de la viga

>> moviendoviga ***% ver movie: movieviga***




```
t =

    0.3000
```

```
t =

    0.4000
```



% Las gráficas nos muestra la forma de la viga en distintos tiempos: cada punto oscila en un plano vertical, indefinidamente, como en el proceso de superposición de ondas que viajan en direcciones distintas sin cambio de forma.

% verificación de solución

>> clear all

>> syms t x

>> u=sin(2*x)*(cos(4*t)+(3/4)*sin(4*t))

u =

sin(2*x)*(cos(4*t)+3/4*sin(4*t))

% verificación de EDP

>> diff(u,t,2)+diff(u,x,4)

ans =

sin(2*x)*(-16*cos(4*t)-12*sin(4*t))+16*sin(2*x)*(cos(4*t)+3/4*sin(4*t))

>> simplify(ans)

ans =

0

% verificación de condiciones iniciales

>> subs(u,t,0)

ans =

sin(2*x)

>> subs(diff(u,t),t,0)

ans =

3*sin(2*x)

% verificación de condiciones de contorno

>> subs(u,x,0)

ans =

0

>> subs(u,x,pi)

ans =

0

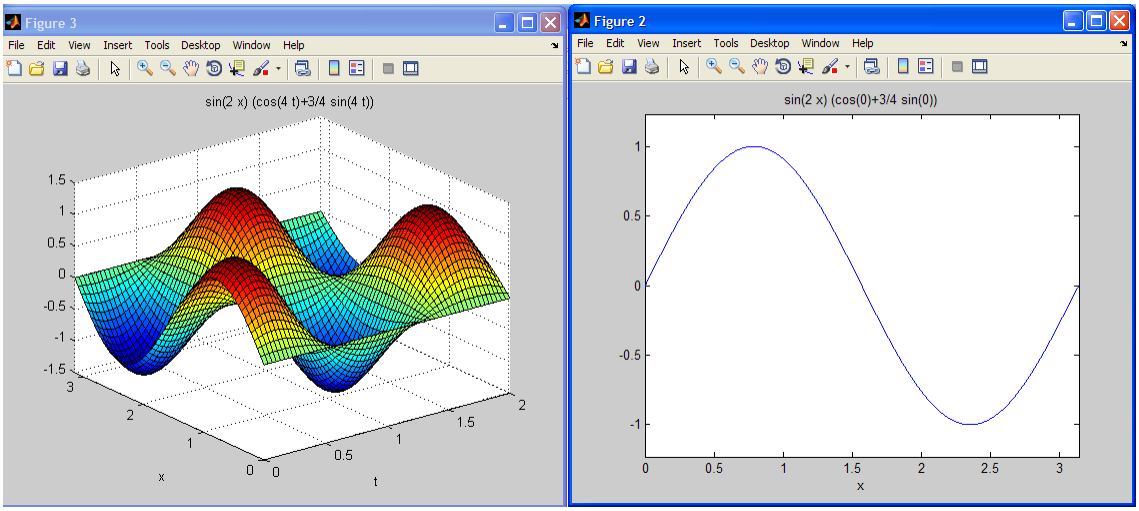
>> subs(diff(u,x,2),x,pi)

ans =

0

```
>> subs(diff(u,x,2),x,0)

ans =

0

% Resumen de instrucciones utilizadas sin mostrar el resultado ni las gráficas

>> u=sin(2*x)*(cos(4*t)+(3/4)*sin(4*t));

>> diff(u,t,2)+diff(u,x,4);

>> simplify(ans);

>> subs(u,t,0);

>> subs(diff(u,t),t,0);

>> subs(u,x,pi);

>> subs(u,x,0);

>> subs(diff(u,x,2),x,0);

>> subs(diff(u,x,2),x,pi);

% superficie solución

>> ezsurf(u,[0,5],[0,pi])

% forma de la viga para t=1, t=2,...

>> ezplot(subs(u,t,1),[0,pi])

>> ezplot(subs(u,t,1),[0,pi])

>> ezplot(subs(u,t,2),[0,pi])

>> ezplot(subs(u,t,3),[0,pi])

>> ezplot(subs(u,t,3.1),[0,pi])

>> ezplot(subs(u,t,3.5),[0,pi])
```

>> ezplot(subs(u,t,2.5),[0,pi])



>> type vibraviga

>> type moviendoviga

% Modelos de segundo orden: se considera el problema mixto, que modela las vibraciones de una cuerda, sujeta en los extremos, que en el instante de tiempo t=0, se desplaza de su posición de equilibrio, toma la forma de la función f(x) y empieza a vibrar en un plano transversal, cada punto x con una velocidad inicial g(x).

% u_{tt}-(3^2)u_xx=0, x en [0,1], t>0

% u(x,0)=f(x), u_t(x,0)=g(x), x en [0,1]

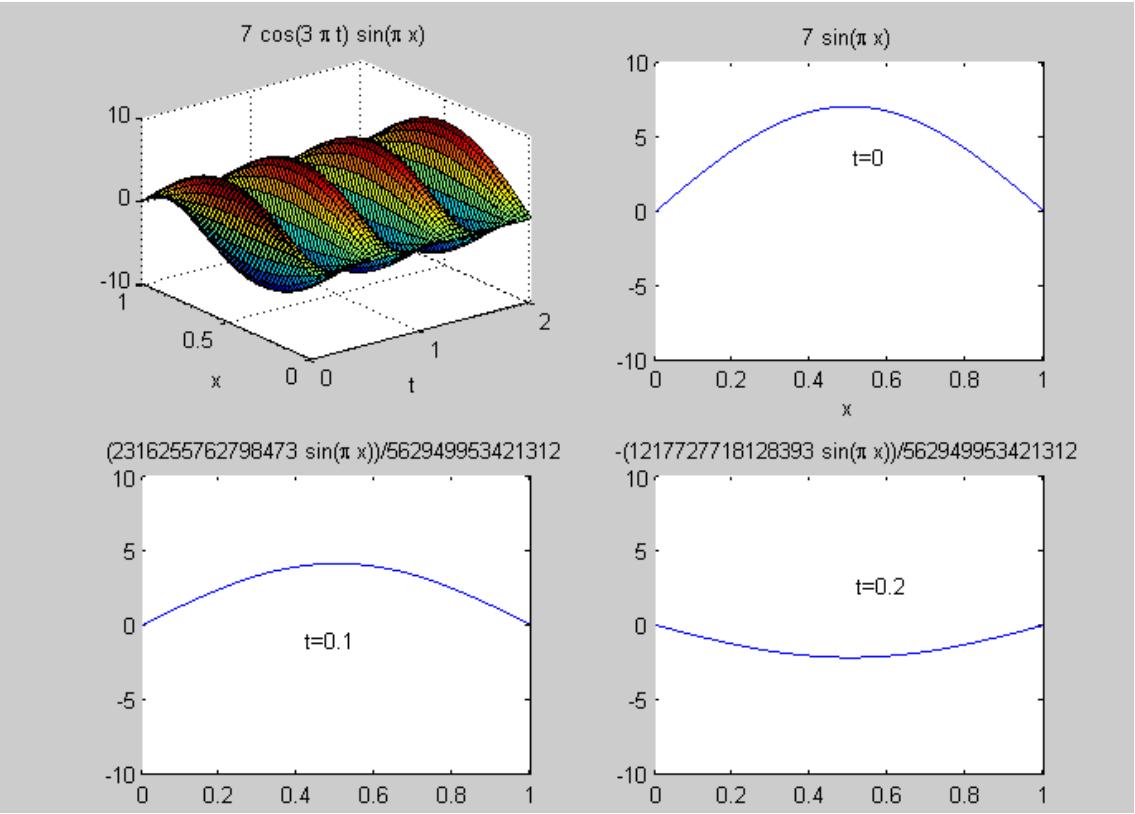
% u(0,t)=0, u(1,t)=0, t>=0

*% la función Matlab mionda2.m dibuja la forma de la cuerda en distintos tiempos, cuando los datos iniciales son funciones propias del problema de valores propios al que se llega por separación de variables. Esto es, dibuja ondas estacionarias, f=7*sin(pi*x), g=0*

>> type mionda2

% ejecutando mionda2.m nos da

>> mionda2



*% No obstante la sensación de movimiento nos lo la da la presentación de las ondas para distintos t; e.g., mionda1.m dibuja la forma de la cuerda para distintos t. El movimiento es similar al que se observa con dalemoviendoondassolosenos.m. Como se observa, cada punto x oscila en una línea vertical entre -7 y 7 dependiendo del tiempo t (esto es, la posición del punto x es sin(pi*x)*7*cos(3*pi*t)). Se puede programar de tal forma que cambie el dato inicial con n. Así, la función ondasestacionarias.m lee n que implica distintas fases iniciales.*

>> type mionda1

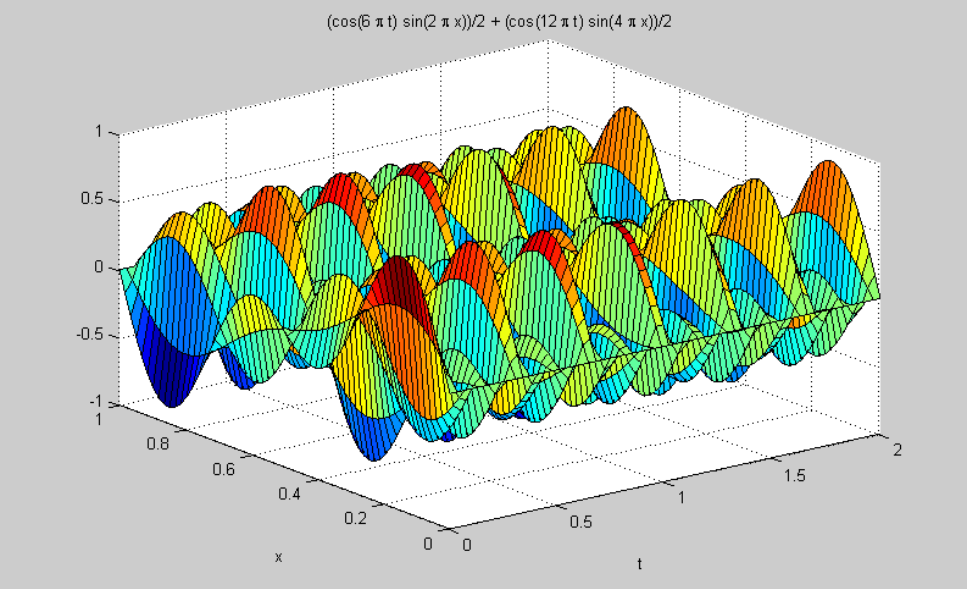
>> type ondasestacionarias

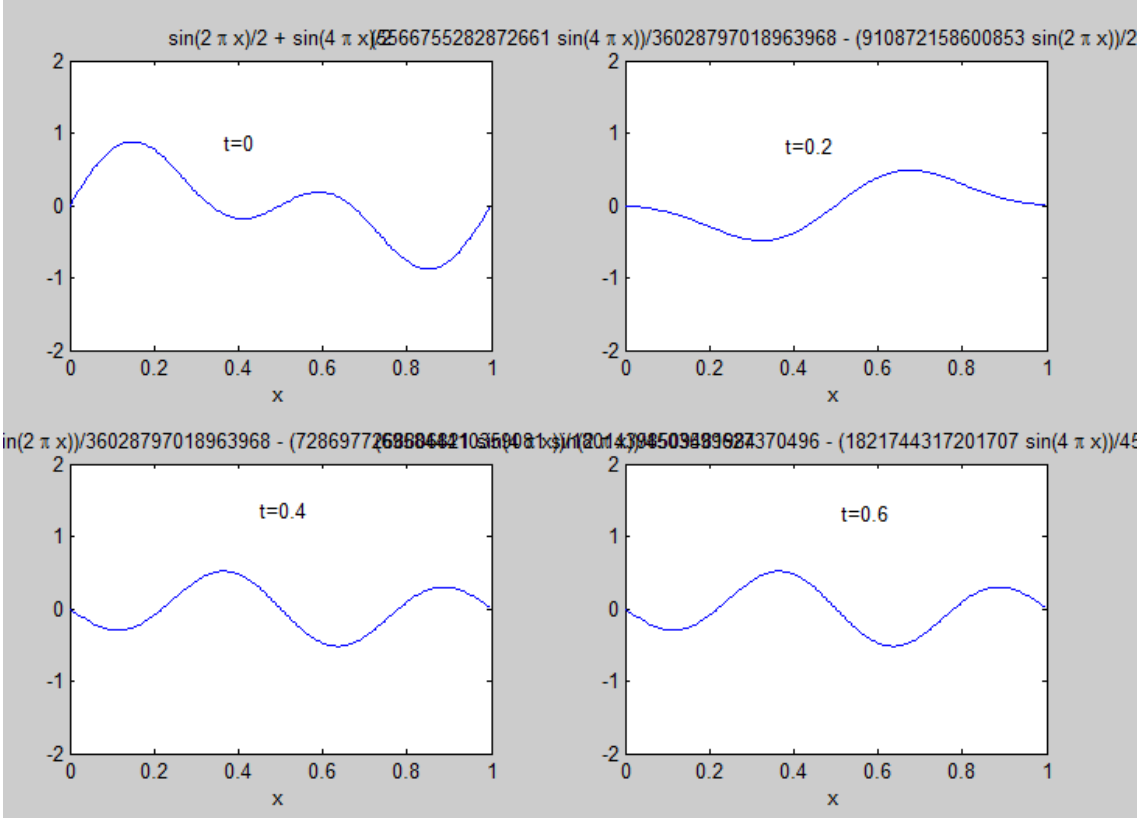
*% la función Matlab miondacomp.m dibuja la forma de la cuerda para distintos tiempos; primero la superficie solución y luego cuatro dibujos. De nuevo la sensación de movimiento se obtiene ejecutando ondaestacionariacomp.m. Las condiciones iniciales que se han considerado son u(x,0)=sin(2*pi*x)/2+sin(4*pi*x)/2, u_t(x,0)=0 y no se trata ya de ondas estacionarias, sino que se desplazan de alguna forma . Es la superposición de dos ondas estacionarias. El movimiento que se observa es similar al de dalemsenocoseno.m*

>> type miondacomp

>> type ondaestacionanariascomp

>> miondacomp





% Para condiciones iniciales generales u(x,0)=f(x), u_t(x,0)=g(x), se necesita un análisis de Fourier. Se crea una función que lea n y calcule los n términos del desarrollo en serie de Fourier de la solución, esto es, la superposición de n de ondas utilizando los coeficientes de Fourier de f y g

% Una función Matlab para g=0, y para f=((x-1/2)^2-1/16) con soporte localizado en [1/4,3/4],

$$f(x) = \begin{cases} (x-(\frac{1}{2}))^2 - \frac{1}{4^2} & \text{si } x \in [\frac{1}{4}, \frac{3}{4}] \\ 0 & \text{si } x \in [0,1],\ x \notin [\frac{1}{4}, \frac{3}{4}] \end{cases}$$

está implementada en la función vibracionescuerda(n) que lee el número de términos de la suma que queremos utilizar para la aproximación y nos dibuja la forma de la cuerda para distintos tiempos:

>> type vibracionescuerda

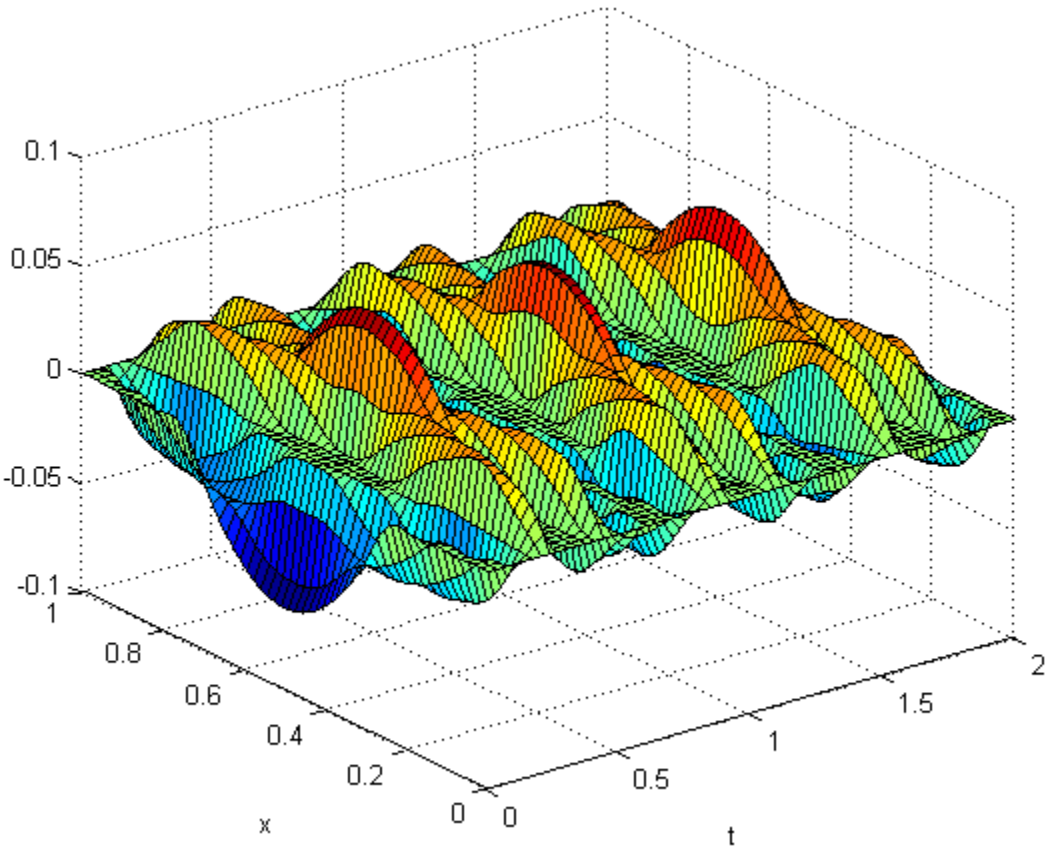
% como en el desarrollo de la serie de Fourier (cuaderno 6), ejecutando el programa, vamos viendo la forma de la cuerda en distintos tiempos. Ya, en t=0, observamos que a más términos nos da mejor aproximación de la función, esto es, de la condición inicial. Los tiempos que se van poniendo dentro del programa, seguidos de pausas después del dibujo para que nos dé tiempo a verle con detalle.

>> vibracionescuerda(20) ***% ver movie: vibracuerda***

solucion =

-.3915e-1*sin(3.142*x)*cos(9.426*t)+.2268e-1*sin(9.426*x)*cos(28.28*t)-.4272e-2*sin(15.71*x)*cos(47.13*t)-.3456e-2*sin(21.99*x)*cos(65.98*t)+.1519e-2*sin(28.28*x)*cos(84.83*t)+.1321e-2*sin(34.56*x)*cos(103.7*t)+.1000e-36*sin(37.70*x)*cos(113.1*t)-.7648e-3*sin(40.85*x)*cos(122.5*t)-.6909e-3*sin(47.13*x)*cos(141.4*t)+.1000e-37*sin(50.27*x)*cos(150.8*t)+.4587e-3*sin(53.41*x)*cos(160.2*t)+.4235e-3*sin(59.70*x)*cos(179.1*t)

% son los 20 primeros términos del desarrollo en serie de la solución exacta, esto es, la aproximación. La superficie solución para t en [0,2].


0.1
0.05
0
-0.05
-0.1
1
0.8
0.6
0.4
0.2
0
x
0
0.5
1
1.5
2
t

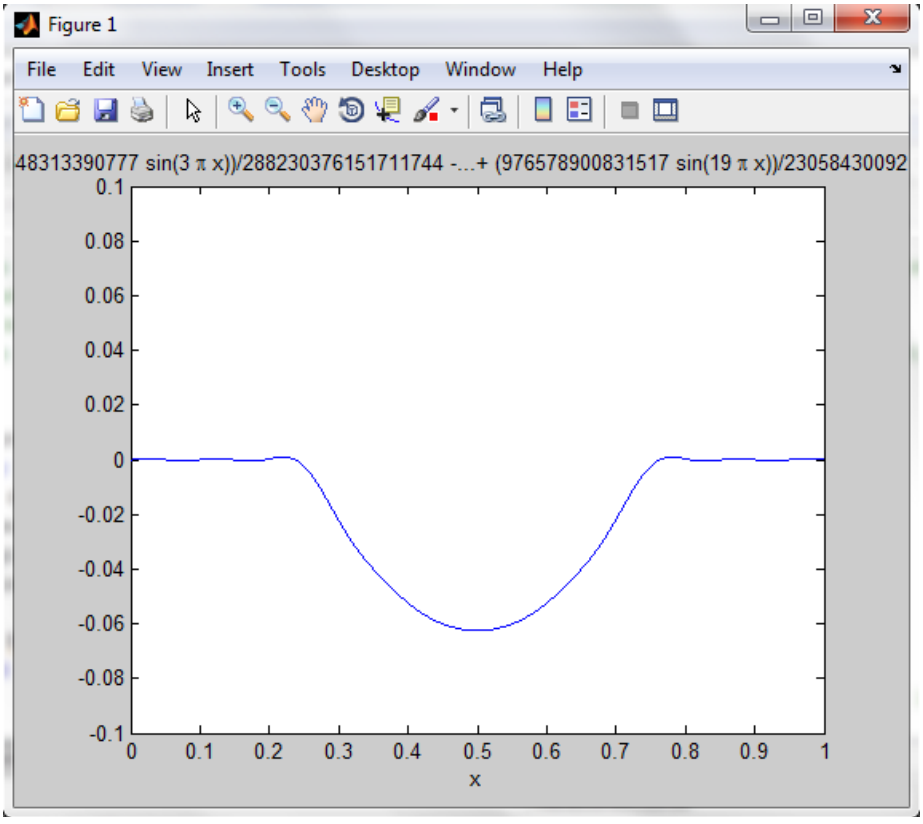
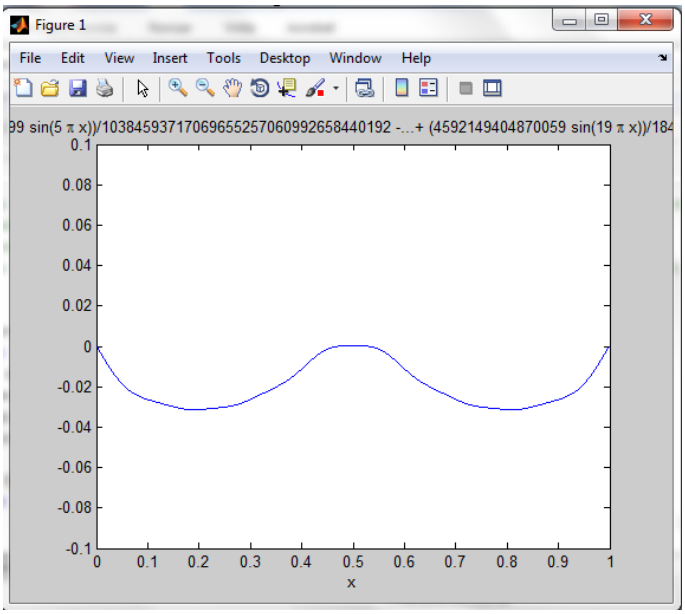
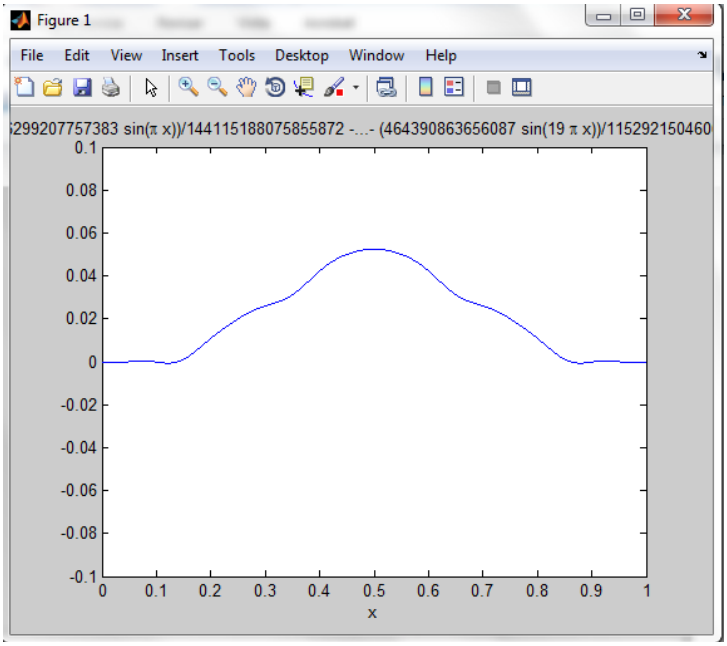


Figure 1
File Edit View Insert Tools Desktop Window Help
48313390777 sin(3 π x))/288230376151711744 -...+ (976578900831517 sin(19 π x))/23058430092
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
-0.1
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
x

% t=0, forma inicial de la cuerda (aproximación con los 20 primeros términos del desarrollo en serie de Fourier)



% t=0.1

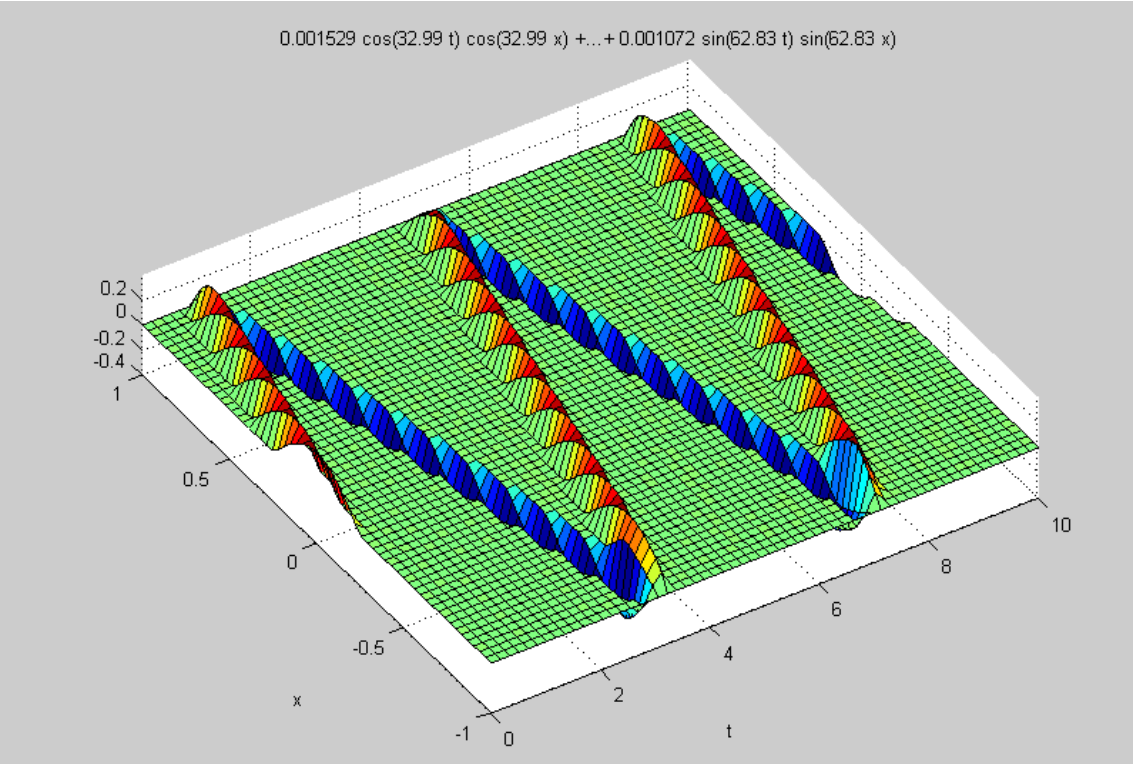


% t=0.2

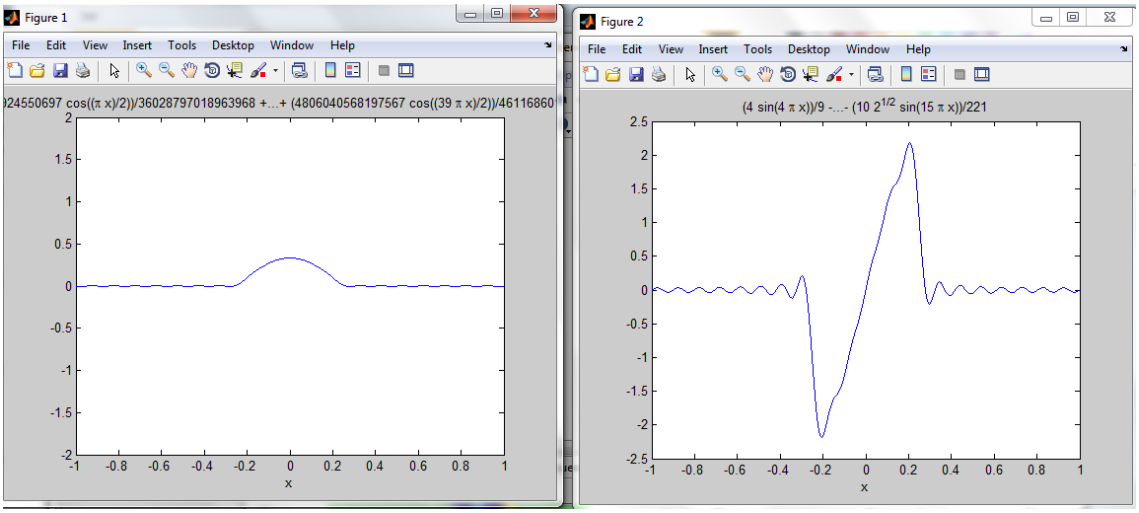
% Con el desarrollo en serie de Fourier y con datos iniciales localizados se pueden conseguir otros fenómenos como 'ondas que chocan' contra un punto y vuelven, hasta chocar contra otro. Esto está programado en la función Matlab fourierrefleja.m que, de nuevo lee el número de términos con el que queremos aproximar la solución, nos devuelve la gráfica de los datos iniciales, la superficie solución y la forma de la onda en distintos tiempos; en todos los casos hablamos de aproximación utilizando los n términos del desarrollo en serie de Fourier. El movimiento se ve en el video "fourierefleja". La superficie solución nos muestra cómo la onda inicial viaja hacia la izquierda hasta chocar contra el extremo de la cuerda fijo x=-1, y luego vuelve hacia la derecha sin cambio de forma hasta chocar con el otro extremo fijo x=1, y así continua indefinidamente chocando contra estos puntos y volviendo. El problema descrito es un modelo para la ecuación de la cuerda vibrante de longitud finita, sujeta en los extremos.

% la ejecución para n=20 nos proporciona

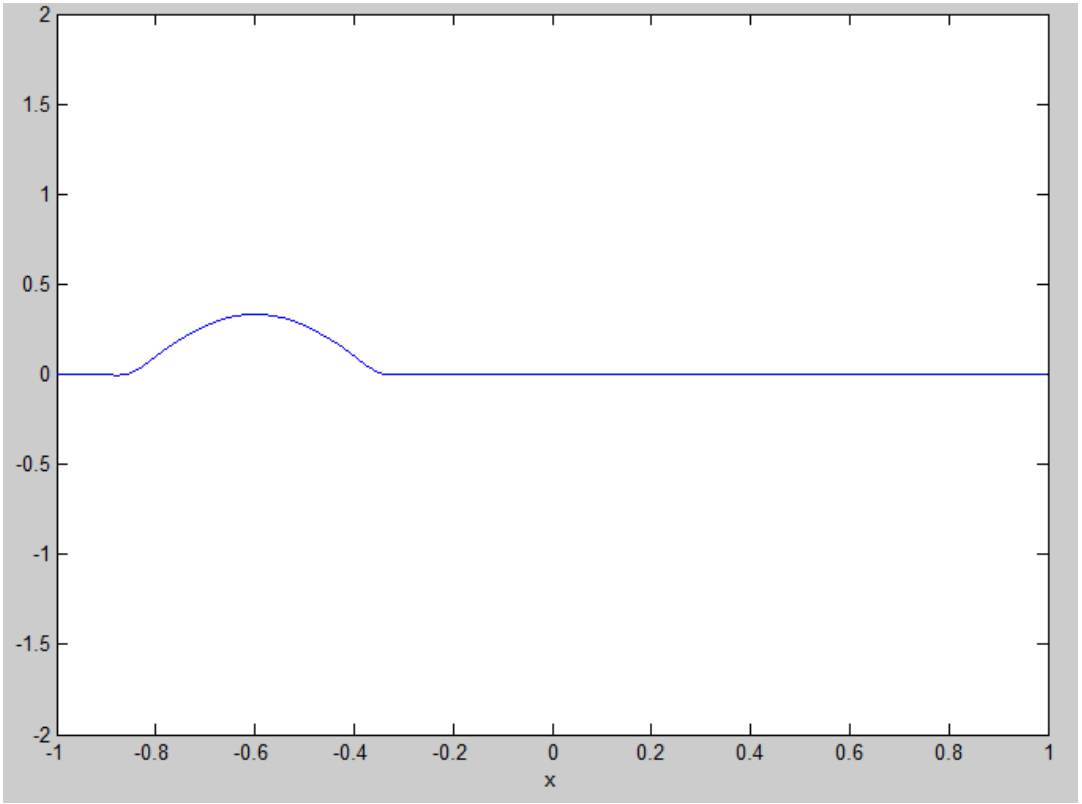
>> fourierrefleja(20) ***% ver movie: fourierefleja / reflejaondas***



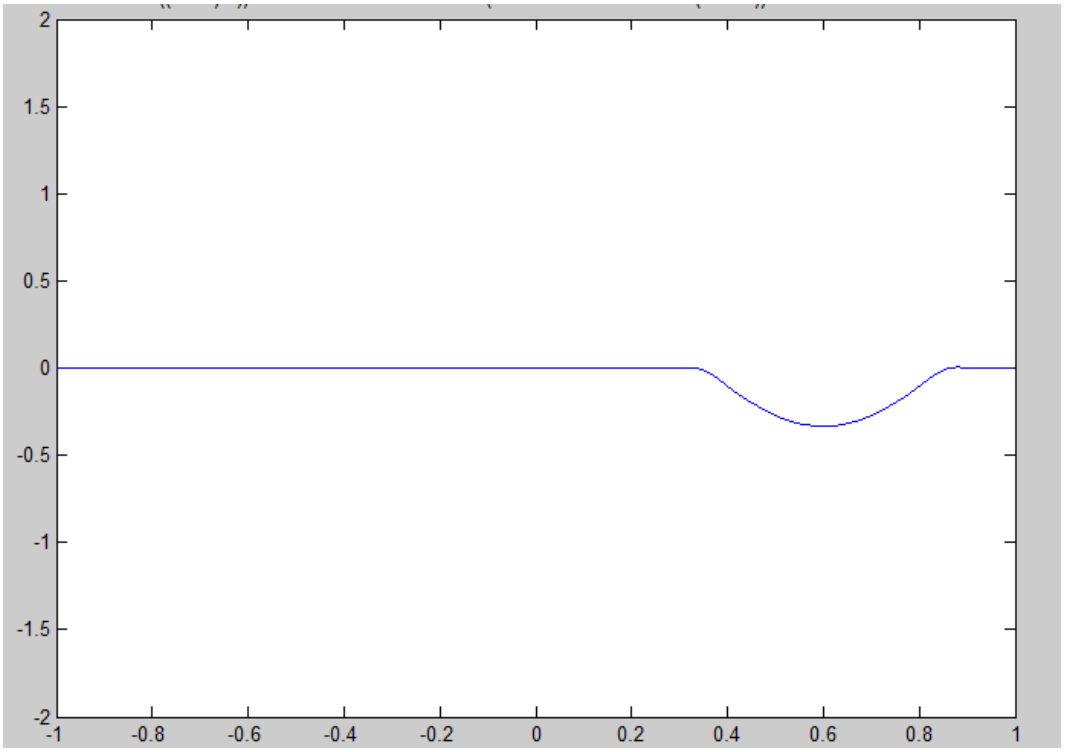
% la aproximación de los datos iniciales f y g con 20 términos del desarrollo en serie de Fourier



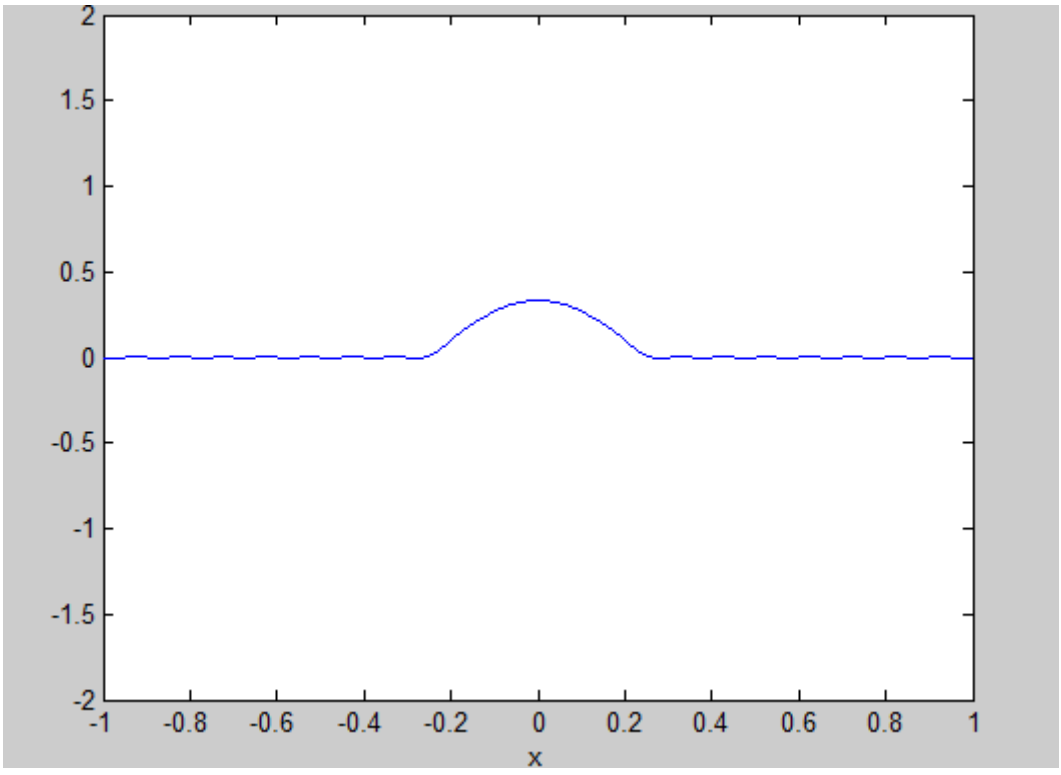
% la forma de la onda (aproximación) pasado un tiempo



% la forma de la onda (aproximación) pasado otro tiempo

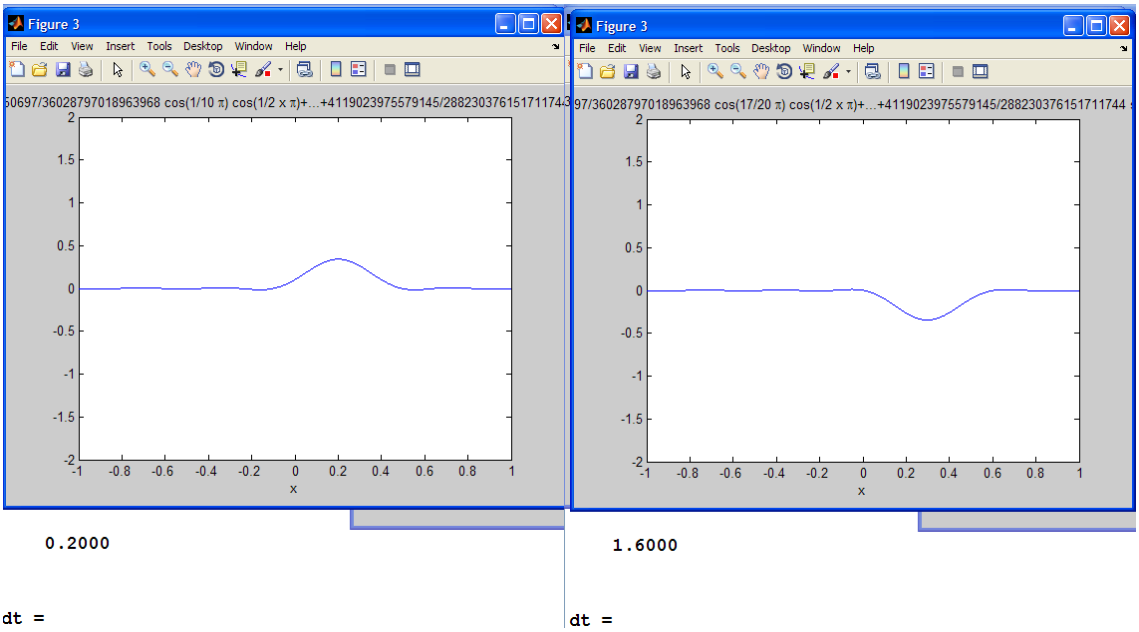


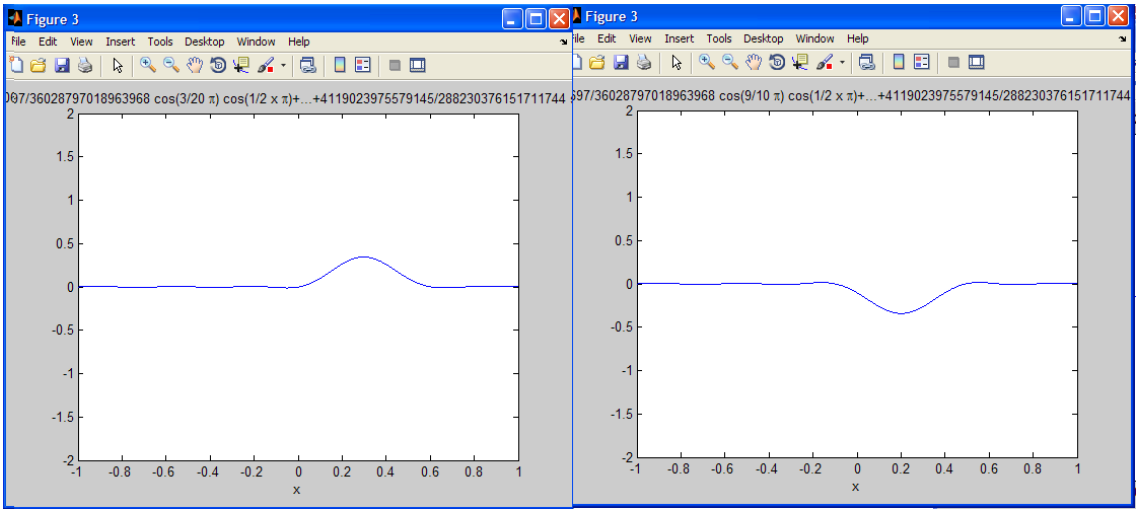
% vemos que la onda continúa así indefinidamente



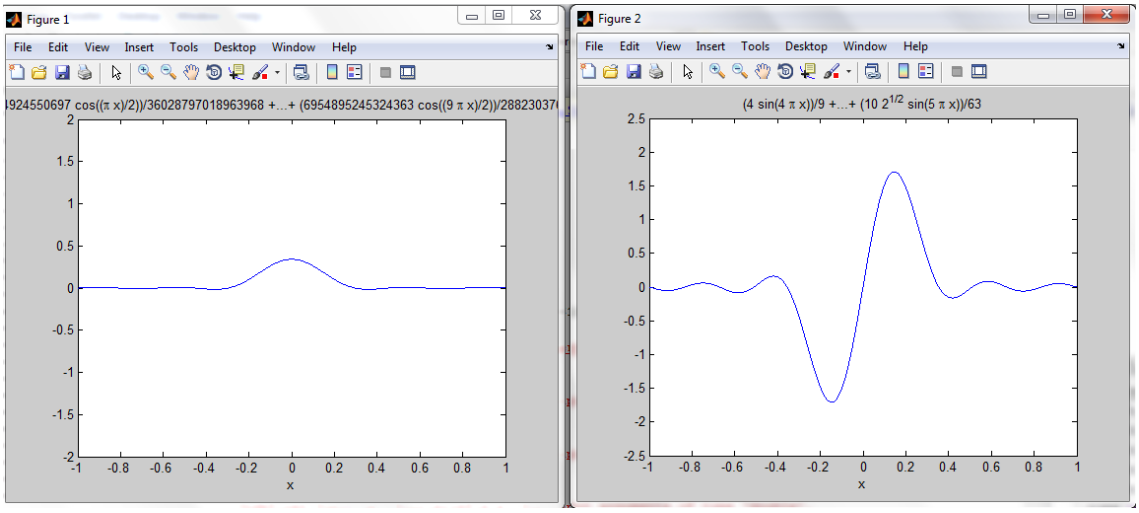
% obviamente lo que vemos con el movie es la vibración de la cuerda en el periodo de tiempo que le pongamos

>> fourierrefleja(5)



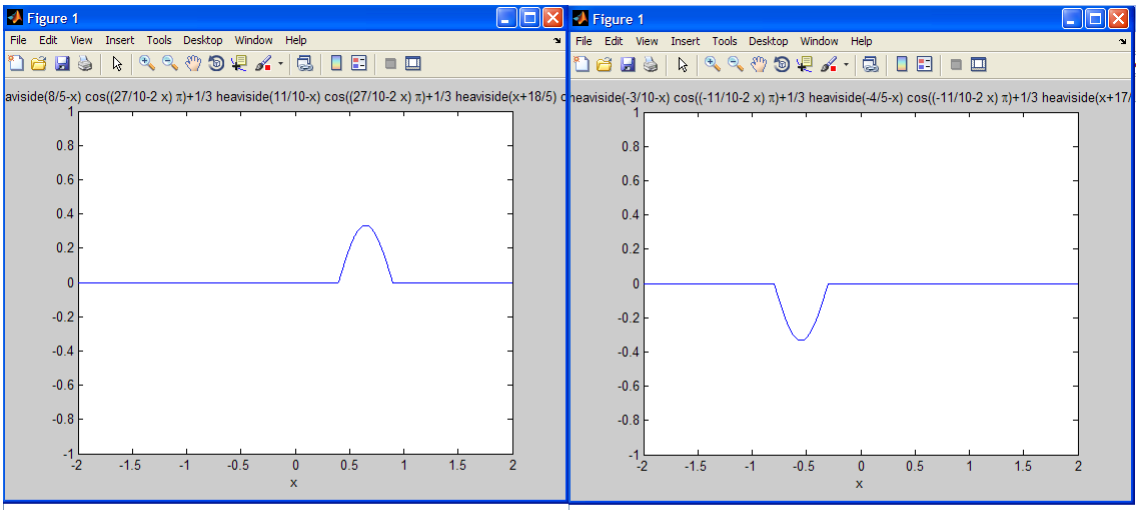


% Observamos que para n=5, fourierreleja(5) nos proporciona peor aproximación de los datos iniciales, por ejemplo:



% Este efecto se puede simular con la fórmula de d'Alembert, como se ha comentado en la parte relativa a ondas, siendo en este caso lo que representamos la solución exacta, pero se necesita un análisis preciso de los tiempos en los que la onda choca con los puntos fijos

>> reflejaondasdospuntos ***% ver movie: reflejaondas***




```
    0.6500

tdis =
```

```
    2.5500

tdis =
```

Modelos de EDP con 2 variables espaciales: Vibraciones elementales / ondas estacionarias / membranas circulares y cuadradas

% Buscamos modos propios de vibración de membranas circulares (membranas que ocupan un dominio circular de radio R_1 o una corona circular comprendida entre los radios R_int y R_1)

$$\frac{\partial^2 u}{\partial t^2} - a^2 \Delta u = 0, \quad t > 0, \quad x^2 + y^2 < R_1$$

$$\frac{\partial^2 u}{\partial t^2} - a^2 \Delta u = 0, \quad t > 0, \quad 0 < R_{int} < x^2 + y^2 < R_1$$

% aquí, y en lo que sigue, a es una constante (alpha abajo) fija que, para los ejemplos, tomamos con el valor 1.

% De cara a resolver el problema se separan variables u=T(t)U(r,theta), con r y theta las coordenadas polares.

% Si se buscan vibraciones radiales la función U solo depende de r; se tiene la EDP unidimensional en la variable espacial r:

$$\frac{\partial^2 u}{\partial t^2} = \alpha^2 \left(\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) \right),$$

% y por separación de variables se llega al problema de valores propios singular

$$\begin{cases} \frac{d}{dr}(r\frac{dy}{dr}) + \lambda r y = 0, & r \in (0, r_1), \\ y(r_1) = 0, & y, \frac{dy}{dr} \text{ acotadas cuando } r \to 0. \end{cases}$$

*% Se buscan los lambda tales que exista solución no nula con y(1)=0 (si se trata de una membrana circular de radio 1), y que en el punto r=0 esta solución esté definida. La ecuación (r*y'(r))'+lambda *r*y=0 se reduce a una ecuación de Bessel de orden 0, y de las dos soluciones (las funciones de Bessel de primera y segunda especie) nos quedamos con la de primera especie pues la segunda tiene una singularidad de tipo logarítmico en el 0. El cambio de variable, para obtener la ecuación de Bessel t^2y''+ty'+t^2y=0, es t=sqrt(lambda)*r. Resolvemos la ED de Bessel con dsolve:*

>> syms t

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

ans =

C1*besselj(0,t)+C2*bessely(0,t)

>> help besselj

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

     Reference page in Help browser
          doc besselj
```

% En el "help" de arriba, pinchando en doc besselj , Matlab nos lleva a la ayuda y nos muestra los desarrollos en serie de potencias que definen las funciones de Bessel de primera especie. Pasa parecido con las funciones de Bessel de segunda especie

>> help bessely

```
bessely Bessel function of the second kind.
    Y = bessely(NU,Z) is the Bessel function of the second kind, Y_nu(Z).
    The order NU need not be an integer, but must be real.
    The argument Z can be complex.  The result is real where Z is positive.
    . . . . .
```

% Además podemos hacer la gráfica de dichas funciones, observando el comportamiento singular de las de segunda especie

>> syms x

>> ezplot(besselj(0,x),[0,20])

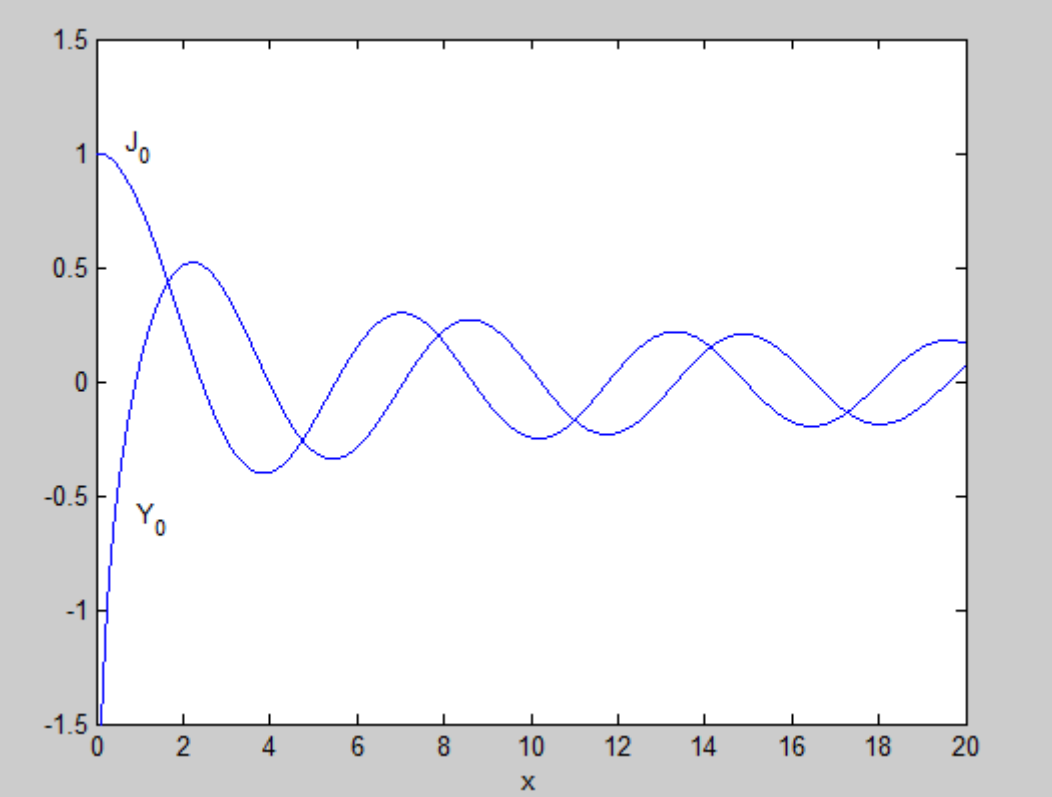
>> hold on

>> ezplot(bessely(0,x),[0,20])

>> axis([0,20,-1.5,1.5])

>> gtext('Y_0')

>> gtext('J_0')



% Resolvemos la ecuación de Bessel de orden mu con dsolve

>> syms t mu

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

ans =

C5*besselj(mu, t) + C6*bessely(mu, t)

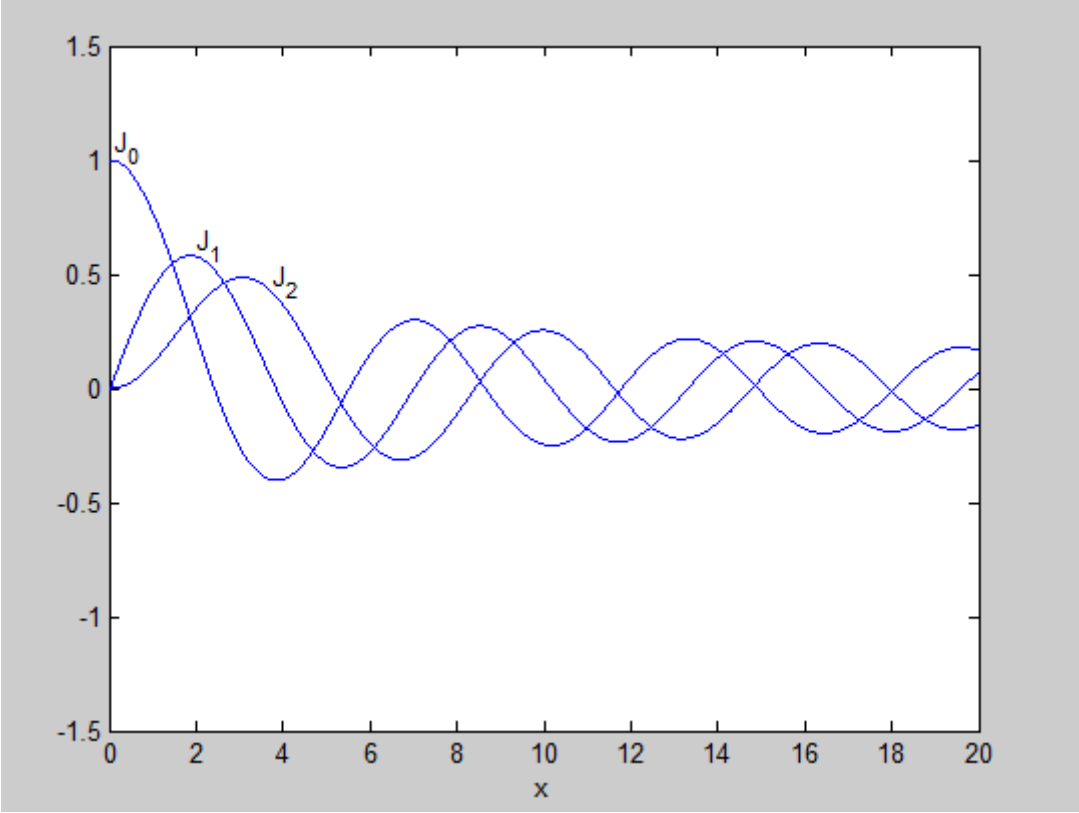
% Las funciones de Bessel de primera especie besselj tienen propiedades especiales; la única que no se anula en x=0 es J_0, mientras que los ceros de cada una de ellas separan y son separados por los de la otra. Abajo dibujamos las funciones de Bessel para índices enteros

>> syms x

>> ezplot(besselj(0,x),[0,20])

>> ezplot(besselj(1,x),[0,20])

```
>> ezplot(besselj(0,x),[0,20])

>> hold on

>> ezplot(besselj(1,x),[0,20])

>> axis([0,20,-1.5,1.5])

>> gtext('J_0')

>> gtext('J_1')

>> ezplot(besselj(2,x),[0,20])

>> axis([0,20,-1.5,1.5])

>> gtext('J_2')
```




```
% También las funciones de Bessel de segunda especie tienen propiedades especiales;
  todas ellas tienen un comportamiento singular en el cero. Vemos que los ceros se
  intercalan al igual que pasa con las funciones de Bessel de primera especie.
```

>> hold off

>> ezplot(bessely(0,x),[0,20])

>> hold on

>> gtext('Y_0')

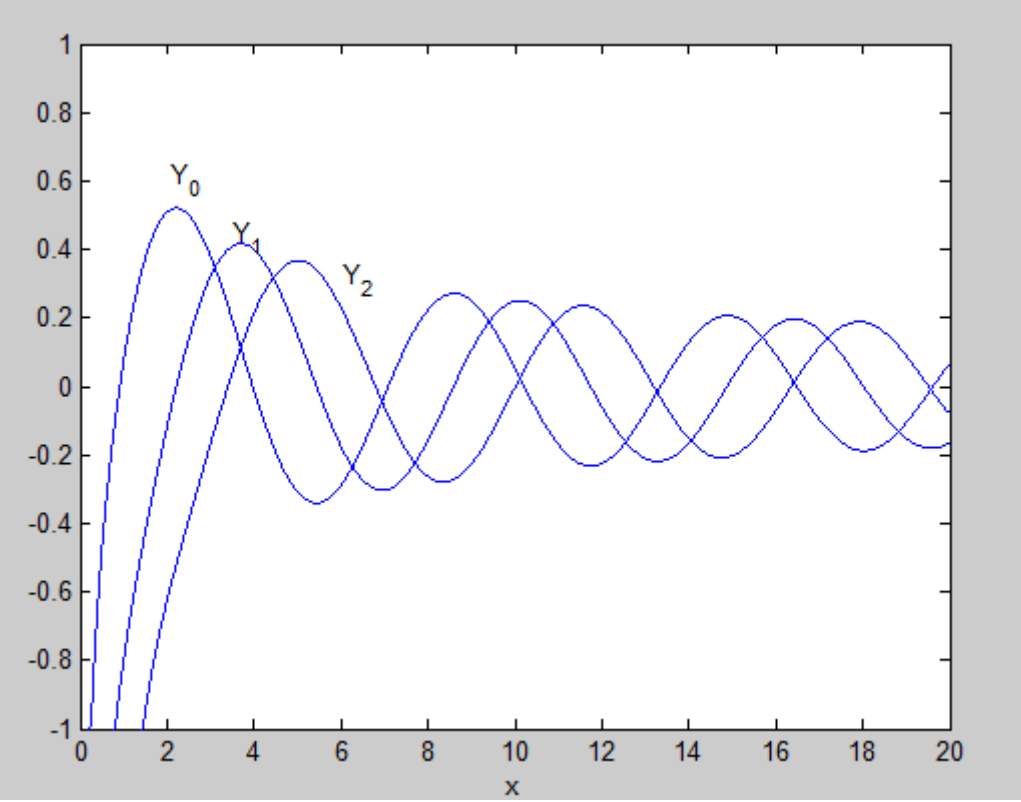
>> ezplot(bessely(1,x),[0,20])

>> gtext('Y_1')

>> ezplot(bessely(2,x),[0,20])

>> gtext('Y_2')

>> axis([0,20,-1,1])



% Mientras las raíces del polinomio indicial r^2-mu^2, no coincidan o la diferencia no sea un natural, las funciones besselj(-mu,x) también están definidas como series de potencias multiplicada por x^{-mu}, y las funciones de Bessel de segunda especie bessely(mu,x) se definen como combinación líneal de besselj(mu,x) y besselj(-mu,x).

% Algunas funciones de Bessel tienen expresiones relativamente simples, como las de órdenes 1/ 2 y 3/2 que se expresan en términos de las funciones trigonométricas:

>> bessely(1/2,x)

ans =

-(2^(1/2)*cos(x))/(pi^(1/2)*x^(1/2))

>> pretty(ans)

```
     1/2
    2    cos(x)
  - -----------
      1/2  1/2
    pi    x
```

>> bessely(3/2,x)

ans =

-(2^(1/2)*(sin(x) + cos(x)/x))/(pi^(1/2)*x^(1/2))

>> pretty(ans)

```
     1/2 /          cos(x) \
    2    | sin(x) + ------ |
         \            x    /
  - ------------------------
            1/2  1/2
          pi    x
```

>> besselj(3/2,x)

ans =

-(2^(1/2)*(cos(x) - sin(x)/x))/(pi^(1/2)*x^(1/2))

>> pretty(ans)

```
     1/2 /          sin(x) \
    2    | cos(x) - ------ |
         \            x    /
  - ------------------------
            1/2  1/2
          pi    x
```

*% Todas estas propiedades, y otras de las funciones de Bessel, son esenciales en la descripción de las vibraciones. Las soluciones de u_tt-alpha^2(u_xx+u_yy)=0, independientes del ángulo theta (en coordenadas polares x=r*cos(theta), y =r*sin(theta) son soluciones de*

$$\frac{\partial^2 u}{\partial t^2} = \alpha^2 \left(\frac{1}{r} \frac{\partial}{\partial r} \left(r \frac{\partial u}{\partial r} \right) \right),$$

*y buscando soluciones en forma de separación de variables, y considerando que se buscan soluciones definidas en r=0, se llega a las soluciones elementales ("ondas estacionarias") u=J_0(sqrt(lambda)*r) * sin(sqrt(lambda)*t) , donde sqrt(lambda) son los ceros de la función de Bessel de primera especie de orden cero*

% Una manera simple de buscar estos ceros es hacer la gráfica de J_0, y utilizar la instrucción Matlab ginput, que nos da aproximadamente las coordenadas de un punto que marquemos en la gráfica.

>> ezplot(besselj(0,x),[0,2.5])

>> ginput

ans =

2.4050 0.0010

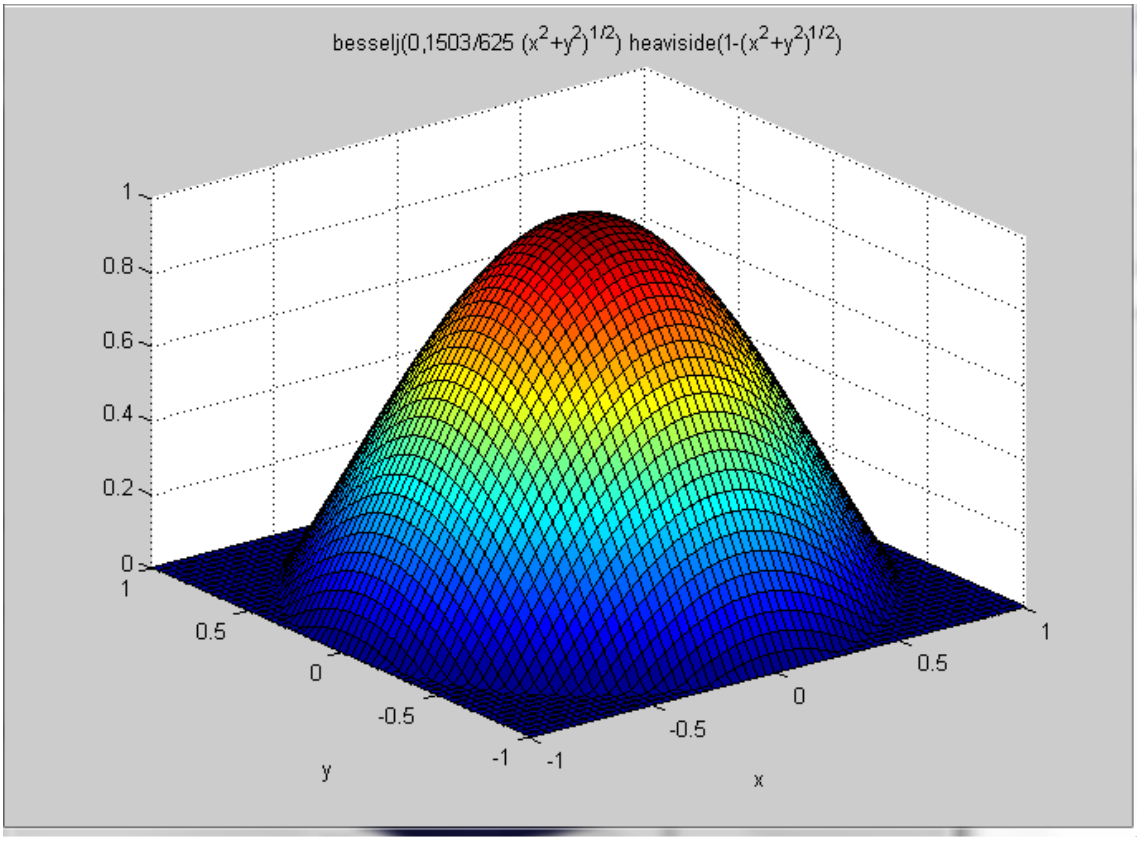
% Los ceros de la función de Bessel de orden cero se encuentran tabulado en la literatura sobre dichas funciones; el primero es 2.4048 (esto es, la aproximación del primero), y este número es bastante cercano del calculado arriba. Ver por ejemplo en la dirección de internet
http://mathworld.wolfram.com/notebooks/SpecialFunctions/BesselFunctionZeros.nb
Así, hacemos la gráfica aproximada del primer modo propio de vibración de una membrana circular de radio 1 sujeta en el borde:

>> u=besselj(0,2.4048*sqrt(x^2+y^2))*heaviside(1-sqrt(x^2+y^2))

u =

besselj(0,1503/625*(x^2+y^2)^(1/2))*heaviside(1-(x^2+y^2)^(1/2))

>> ezsurf(u,[-1,1,-1,1])



*% Para las condiciones iniciales u(x,y,0)=0, u_t(x,y,0)=J_0(2.4048*r) , las vibraciones asociadas (aproximaciones de vibraciones) vienen simuladas por: w=besselj(0,2.4048*sqrt(x^2+y^2))*heaviside(1-sqrt(x^2+y^2))*sin(t*2.4048) (o esta misma función de Bessel multiplicada por una constante y cambiando seno por coseno, dependiendo de las condiciones iniciales). Aquí la función heaviside se ha utilizado solo para extender la solución por cero fuera del círculo unidad.*

>> syms t x y

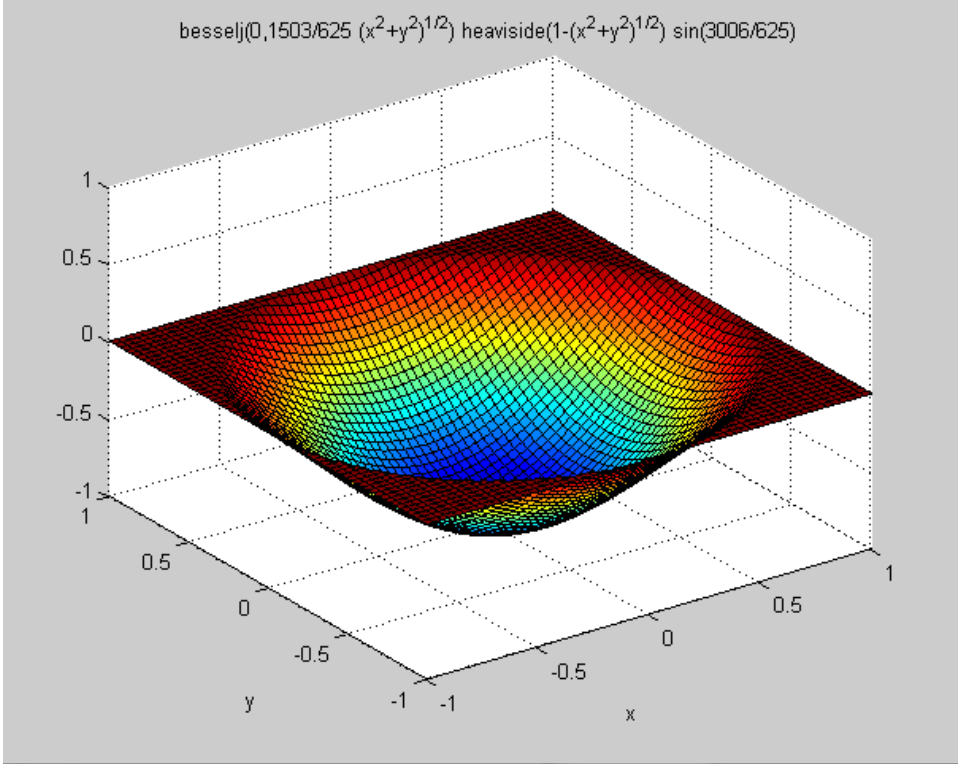
>> w=besselj(0,2.4048*sqrt(x^2+y^2))*heaviside(1-sqrt(x^2+y^2))*sin(t*2.4048)

w =

besselj(0,1503/625*(x^2+y^2)^(1/2))*heaviside(1-(x^2+y^2)^(1/2))*sin(1503/625*t)

>> ezsurf(subs(w,t,2),[-1,1,-1,1])

>> axis([-1,1,-1,1,-1,1])



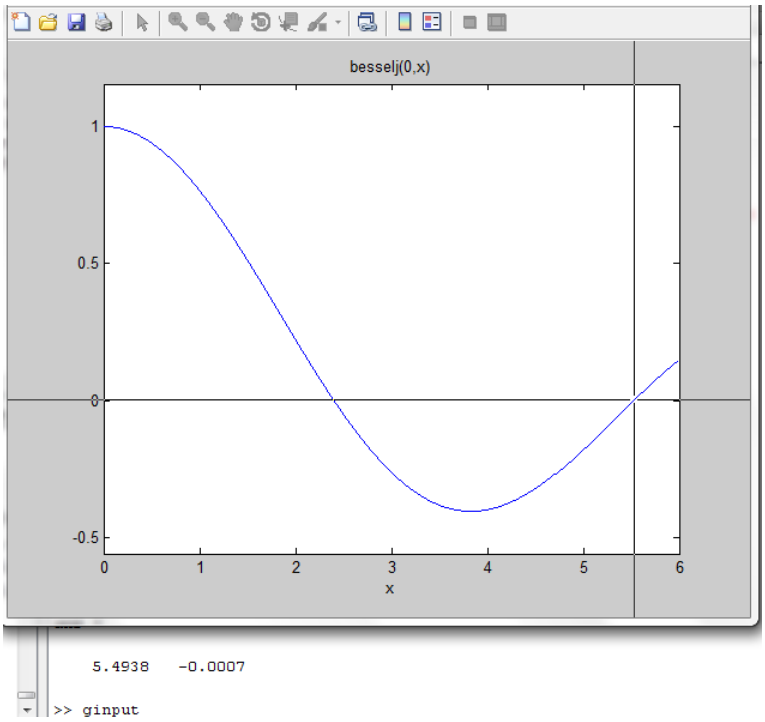
% el movimiento se ve con la función Matlab vibracionmembrana.m, que ejecutamos

>> vibracionmembrana ***% ver movie: vibramembrana***

% en relación al segundo modo propio de vibración, que presenta oscilaciones en el sentido radial sólo, se puede obtener de forma análoga

>> ezplot(besselj(0,x),[0,6])

>> ginput



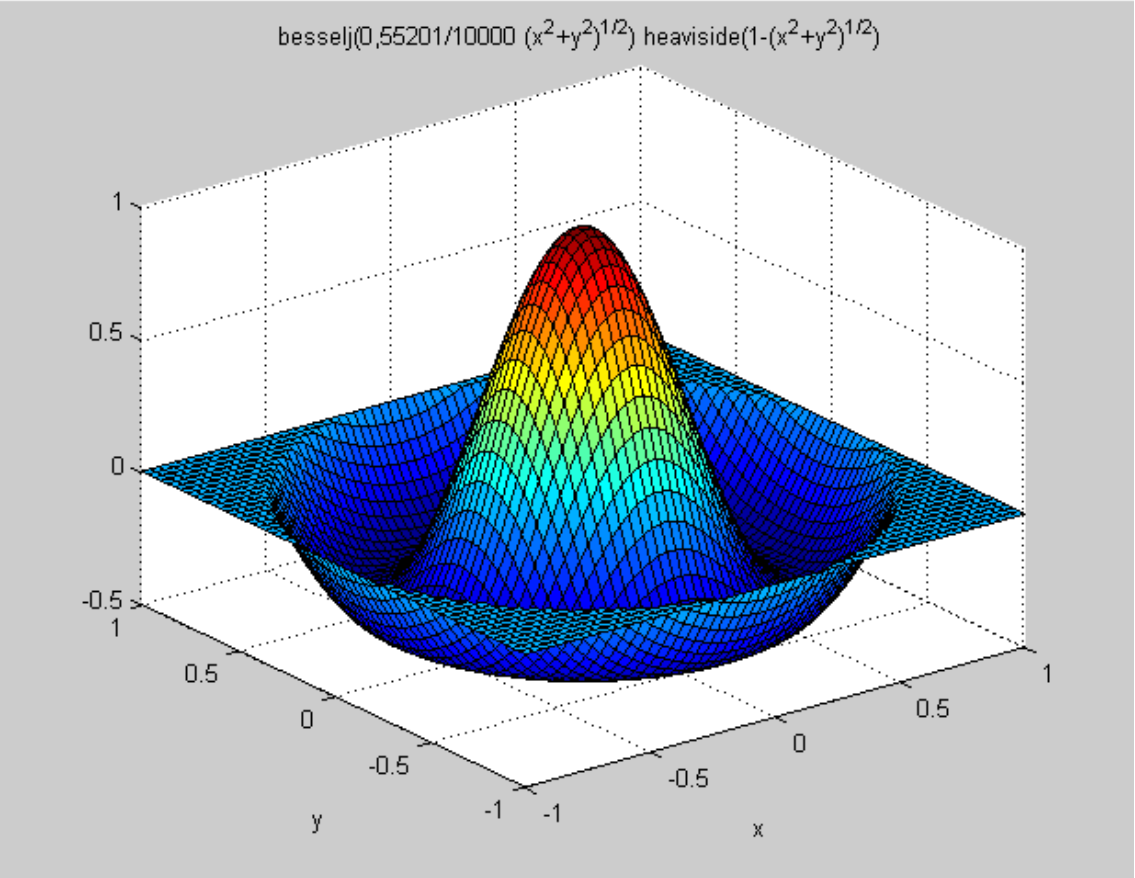
% el segundo cero tabulado en los libros es 5.5201, bastante cercano al que hemos obtenido arriba con ginput, se puede mejorar, pero una aproximación del modo propio asociado al valor propio aproximado (5.5201)^2 es

>> u=besselj(0,5.5201*sqrt(x^2+y^2))*heaviside(1-sqrt(x^2+y^2))

u =

besselj(0,55201/10000*(x^2+y^2)^(1/2))*heaviside(1-(x^2+y^2)^(1/2))

>> ezsurf(u,[-1,1,-1,1])



% las vibraciones asociadas se ven ejecutando vibracionmembranamodo2.m

>> vibracionmembranamodo2 ***% ver movie: vibramembranamodo2***

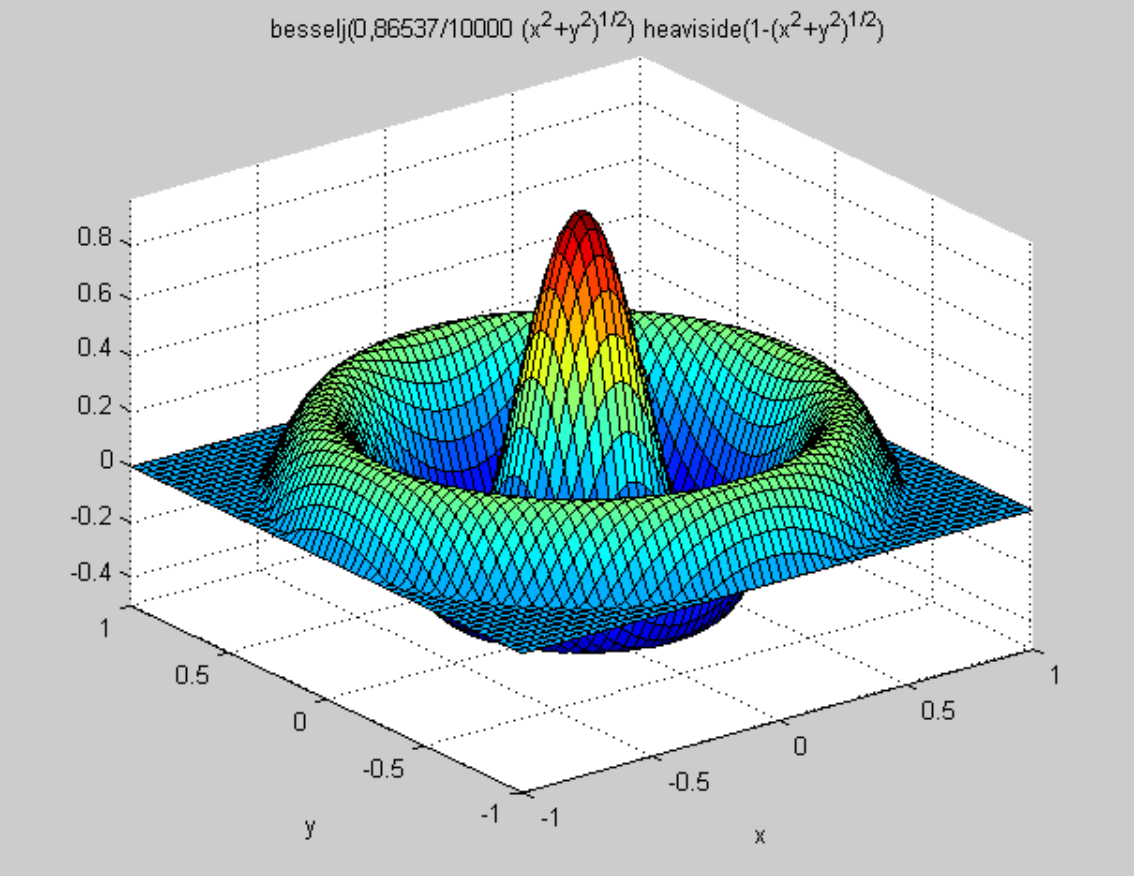
% Así también se puede tener el tercer modo propio de vibración como

>> u=besselj(0, 8.6537*sqrt(x^2+y^2))*heaviside(1-sqrt(x^2+y^2))

u =

besselj(0,86537/10000*(x^2+y^2)^(1/2))*heaviside(1-(x^2+y^2)^(1/2))

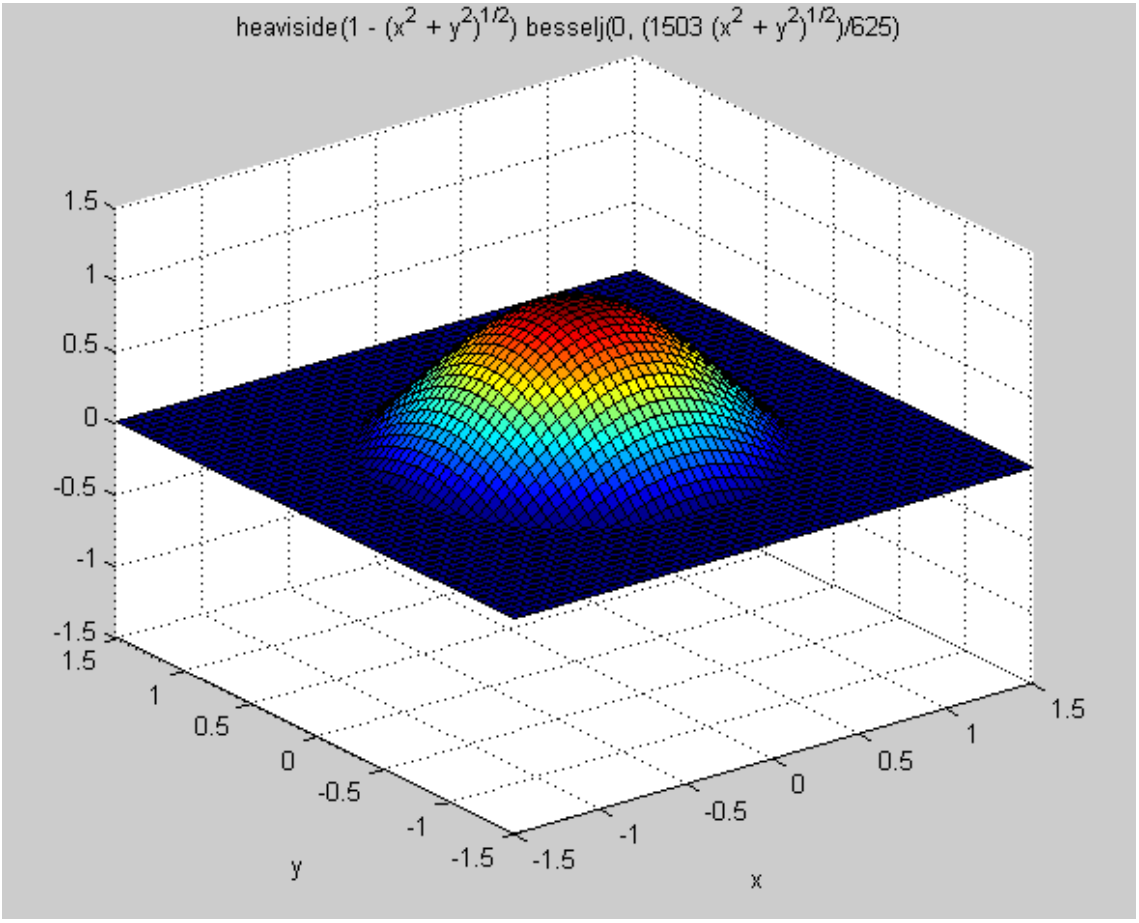
>> ezsurf(u,[-1,1,-1,1])



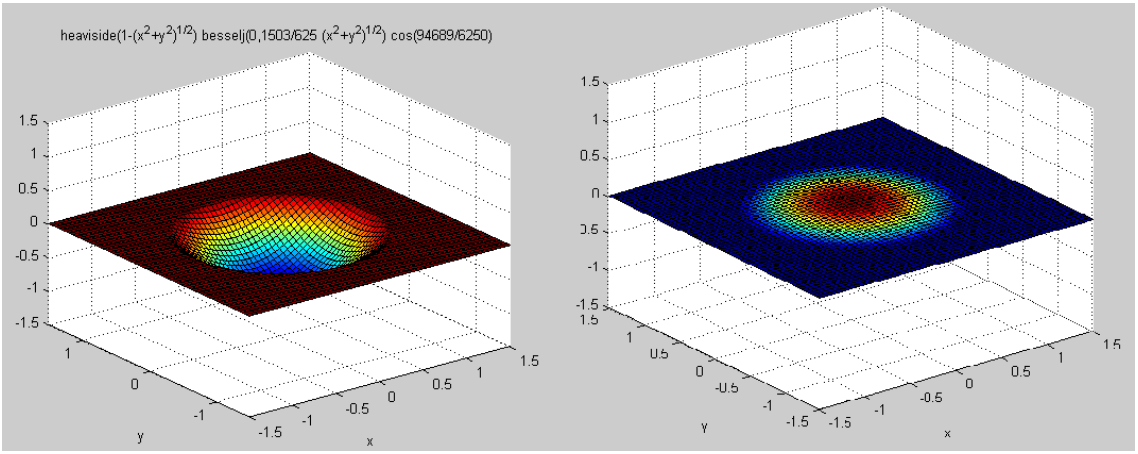
% propongo ver los movies que nos muestran las vibraciones asociadas al primer y segundo modo propio de vibración de la membrana, y la forma de la membrana para distintos t supuesto, que las condiciones iniciales para iniciar la vibración son u(x,y,0) dada por el modo propio y u_t=0

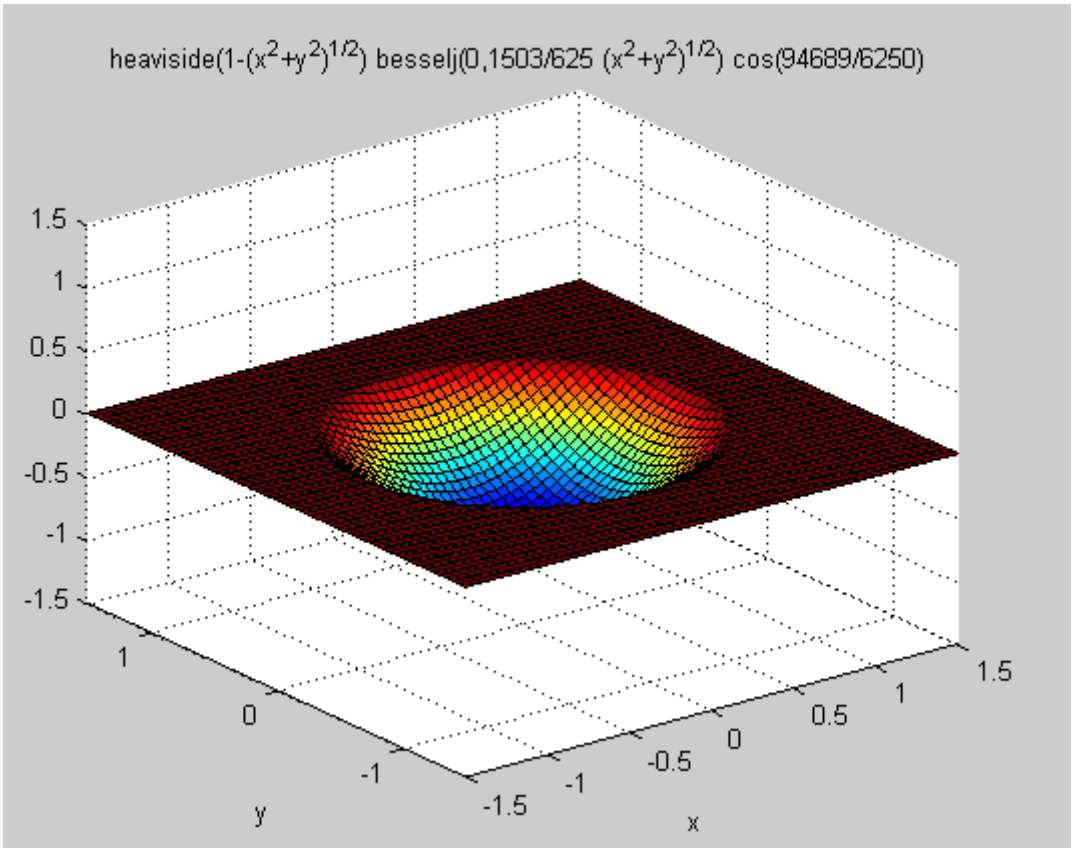
>> vibramembrana ***% ver movie: vibramembrana***

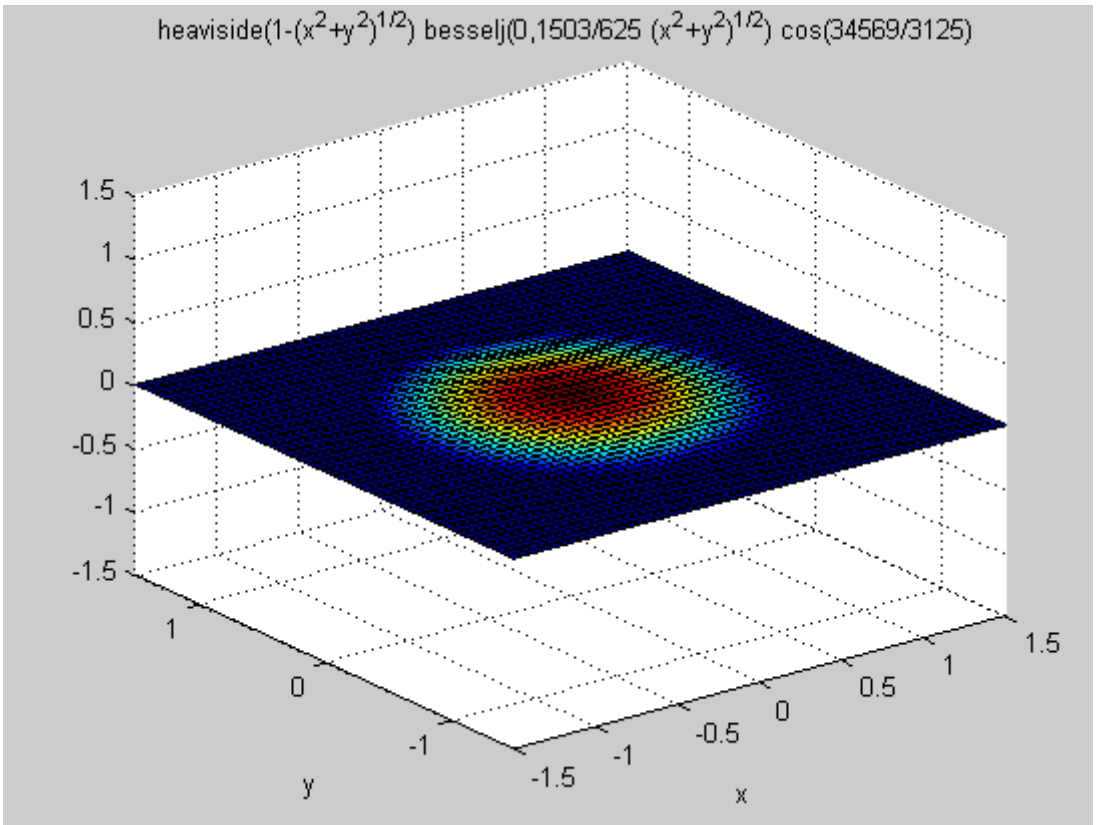
% primer modo propio de vibración: forma inicial de la membrana



% forma de la membrana para distintos valores de t

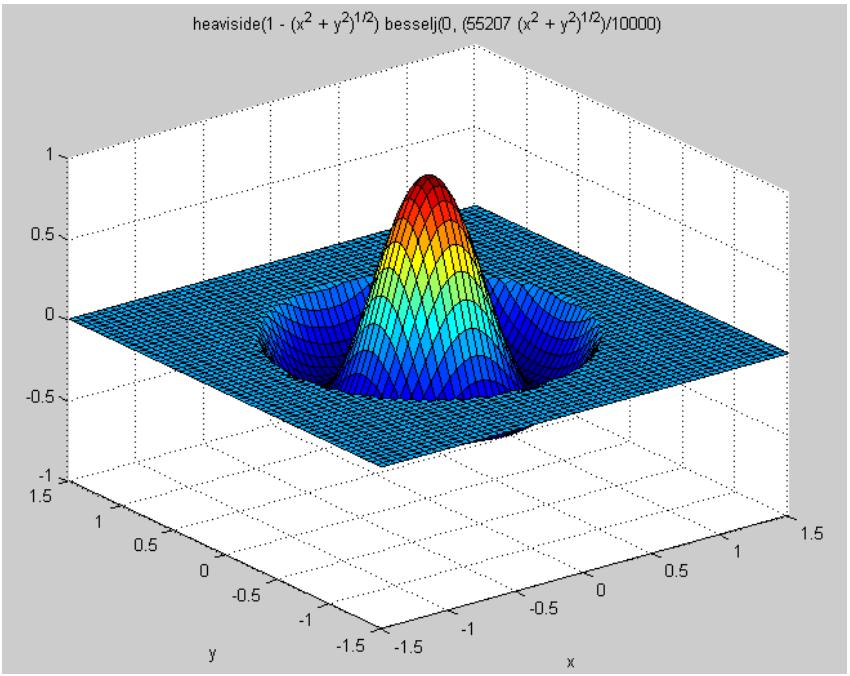



heaviside(1-(x^2+y^2)^1/2) besselj(0,1503/625 (x^2+y^2)^1/2) cos(94689/6250)
1.5
1
0.5
0
-0.5
-1
-1.5
y
x

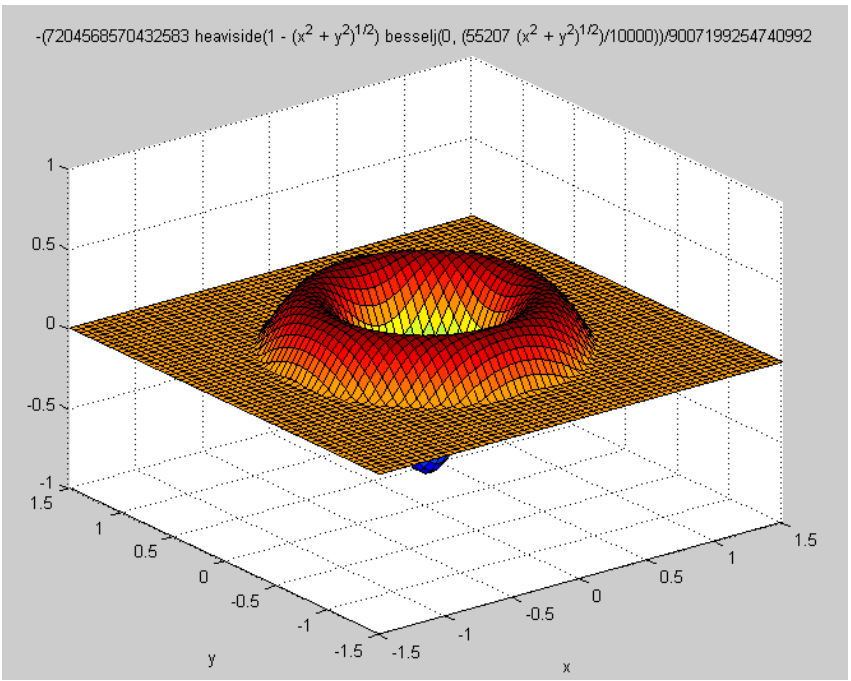

heaviside(1-(x^2+y^2)^1/2) besselj(0,1503/625 (x^2+y^2)^1/2) cos(34569/3125)
1.5
1
0.5
0
-0.5
-1
-1.5
y
x

>> vibracionmembranamodo2 *% ver movie: vibramembranamodo2*

% segundo modo propio de vibración (para vibraciones radiales): forma inicial de la membrana



% forma de la membrana pasado un tiempo



% Los ficheros que contienen las instrucciones Matlab para el movimiento:

>> type vibracionmembrana

>> type vibracionmembranamodo2

% En el caso de una membrana con forma de corona circular, sujeta en los bordes (en dos circunferencias concéntricas con radios r_int y r_1), para calcular los modos propios de vibración hay que resolver el problema de valores propios

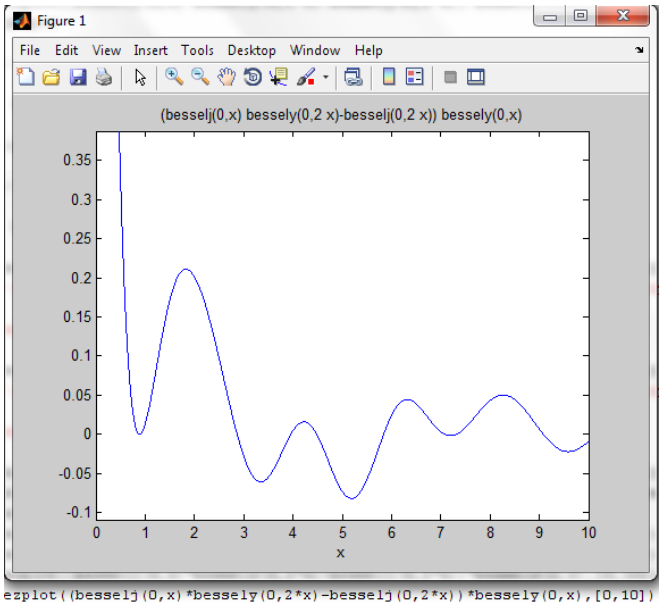
$$\begin{cases} \frac{d}{dr}(r\frac{dy}{dr}) + \lambda r y = 0\,, & r \in (r_{int}, r_1), \\ y(r_1) = 0 \quad , & y(r_{int}) = 0\,. \end{cases}$$

% supuesto que los radios son r_int=1 y r_1=2, se demuestra que los valores propios son los lambda que hacen cero el determinante

C1 J_0(sqrt(\lambda))+C2 Y_0(sqrt(lambda))=0
C1 J_0(2 sqrt(\lambda))+C2 Y_0(2 sqrt(lambda))=0

de tal manera que se puede despejar C1 en función C2, y entonces las soluciones son las raíces de la ecuación
*(besselj(0,x)*bessely(0,2*x)-besselj(0,2*x))*bessely(0,x),*

y un simple gráfico nos muestra que tiene una infinidad de raíces



% Así, para r_int=1 y r_1=2, el conjunto de instrucciones

>> ezplot(besselj(0,2*x)*bessely(0,x)/besselj(0,x),[0,2.5^2])

>> hold on

>> ezplot(bessely(0,2*x),[0,2.5^2])

% nos muestra que hay un cero entre 3 y 4, que visualizamos mejor con

>> hold off

>> ezplot(bessely(0,2*x),[3,4])

>> hold on

>> ezplot(besselj(0,2*x)*bessely(0,x)/besselj(0,x),[3,4])

>> ginput *% pinchando en el punto de corte*

ans =

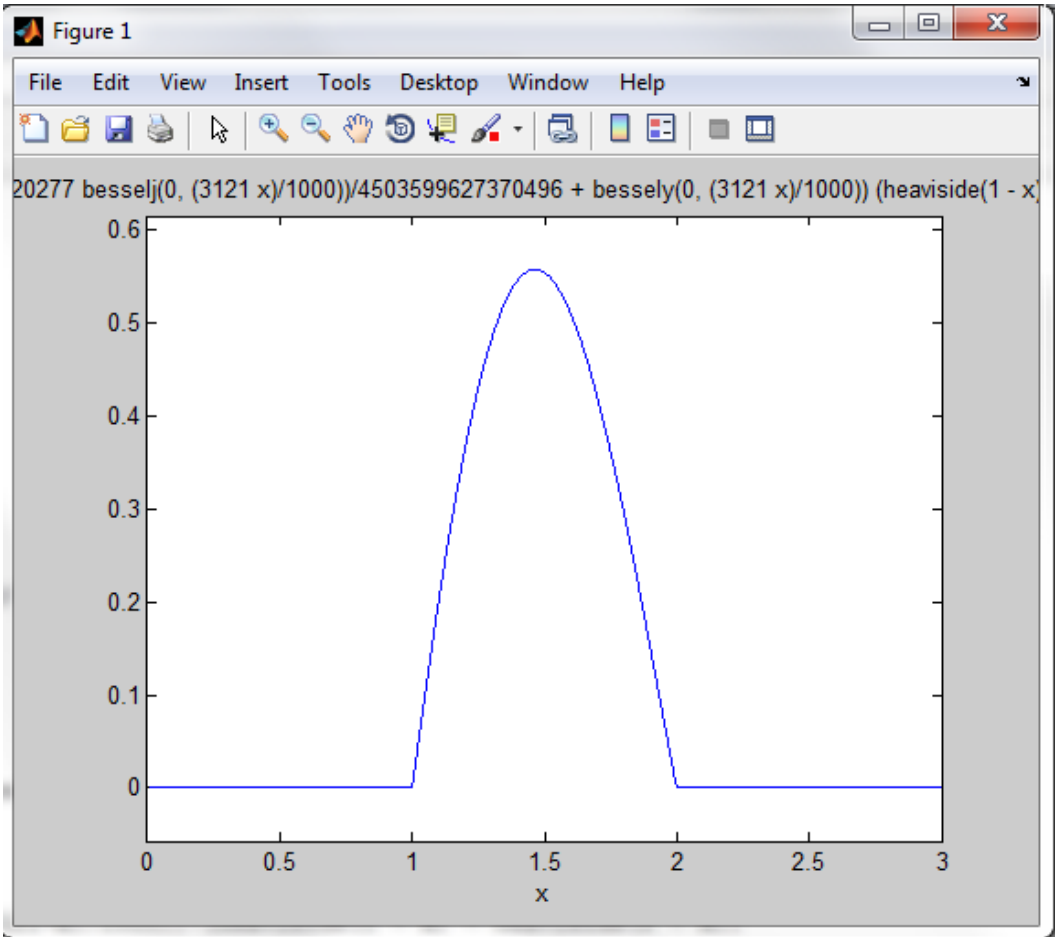
3.1210 -0.2372

>> w=((bessely(0,2* 3.1210)/besselj(0,2* 3.1210))*besselj(0, 3.1210 *x)- bessely(0, 3.1210 *x)) *(heaviside(2-x)-heaviside(1-x))

w =

((5089116369520277*besselj(0, (3121*x)/1000))/4503599627370496 + bessely(0, (3121*x)/1000))*(heaviside(1 - x) - heaviside(2 - x))

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



% nos muestra que la función es positiva entre 1 y 2, luego, por simetría radial, el primer modo propio de vibración de la membrana con forma circular y su gráfica se obtienen con las instrucciones

>> u=((bessely(0,2* 3.1210)/besselj(0,2* 3.1210))*besselj(0, 3.1210 *sqrt(x^2+y^2))-bessely(0, 3.1210 *sqrt(x^2+y^2))) *(heaviside(2-sqrt(x^2+y^2))-heaviside(1-sqrt(x^2+y^2)))

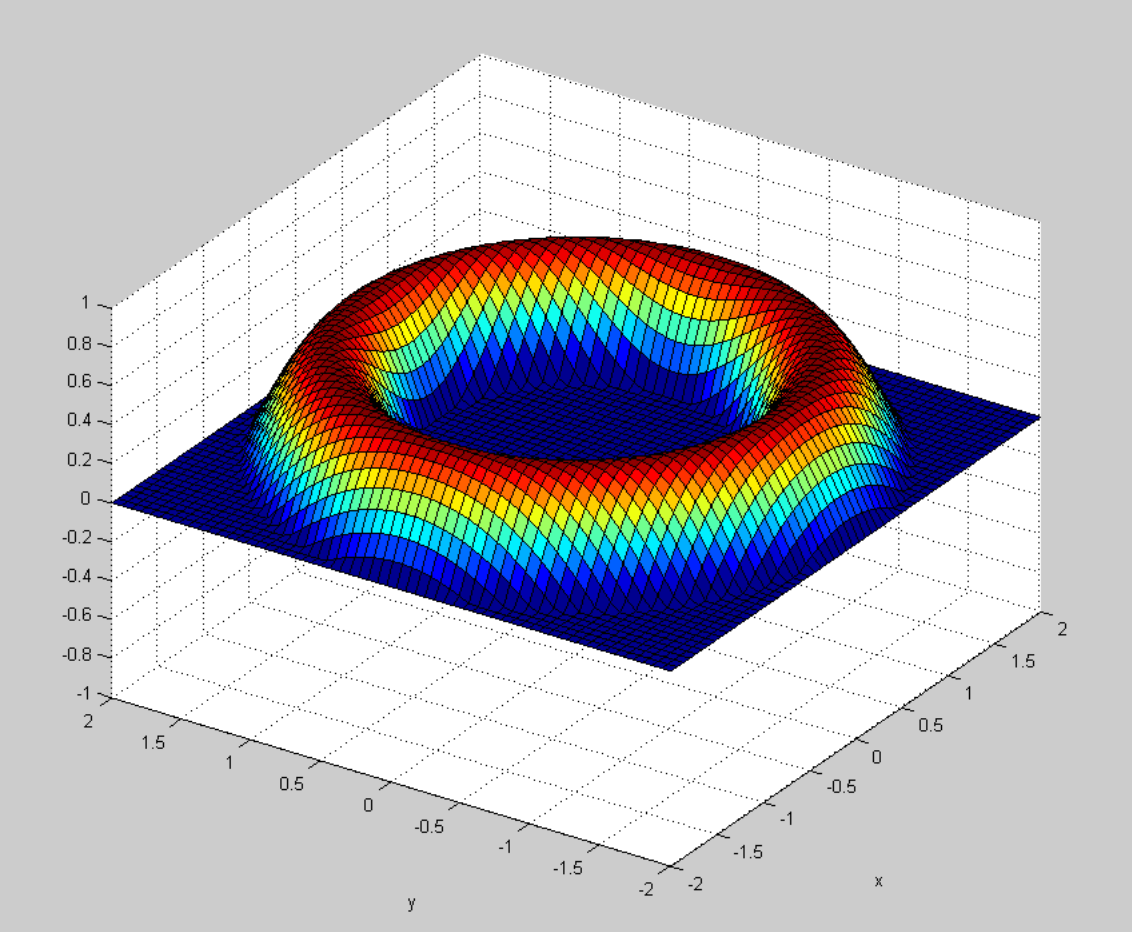
u =

((5089116369520277*besselj(0, (3121*(x^2 + y^2)^(1/2))/1000))/4503599627370496 + bessely(0, (3121*(x^2 + y^2)^(1/2))/1000))*(heaviside(1 - (x^2 + y^2)^(1/2)) - heaviside(2 - (x^2 + y^2)^(1/2)))

>> ezsurf(u,[-2,2],[-2,2])

>> axis([-2,2,-2,2,-1,1])

>> title('')



*% Todos estos modos propios de vibración encontrados en función de las funciones de Bessel de orden 0, tanto para la membrana circular como para la membrana con forma de corona, nos permiten resolver problemas de vibraciones u_tt-(u_xx+u_yy)=0, u(x,y,0)=f(r), u_t(x,y,0)=g(r), u(x,y,t)=0 , en el borde, de manera totalmente análoga a las vibraciones de una cuerda; sin embargo, no nos permiten hacerlo si f y g dependen de theta. De cara a obtener todas las vibraciones, la separación de variables nos lleva a una serie de valores propios que son las raíces de las ecuaciones J_n(sqrt(lambda))=0 , con n=0,1,2,. . . . Las dos funciones propias para el operador de Laplace asociadas a este cero (valor de lambda encontrado) son, para n=1,2,3, . . . :u1=J_n(sqrt(lambda)r)*cos(n*theta), u2=J_n(sqrt(lambda)r)*sin(n*theta) .*
Si n=0, se tiene solo la función propia u0=J_0 (sqrt(lambda)r), que coincide con el calculado para las vibraciones que solo capturan las oscilaciones radiales.

*% Así por ejemplo, para n=3, se considera el tercer cero x (ver tablas de ceros de J_n; sqrt(lambda)=13.0152) / J_3(x)=0, y el conjunto de instrucciones de abajo, nos aproxima la gráfica del modo propio de vibración asociado, que como vemos contiene oscilaciones en las direcciones radiales y angulares; mayor cero de J_n implica más oscilaciones radiales mientras que mayor n implica más oscilaciones angulares. La vibración asociada se obtiene multiplicando estas funciones propias por cos(sqrt(lambda)*t), o por sin(sqrt(lambda)*t), en función de las condiciones iniciales que se tengan.*

>> syms x y r theta

>> theta=atan(y/x)

theta =
atan(y/x)

>> r=sqrt(x^2+y^2)

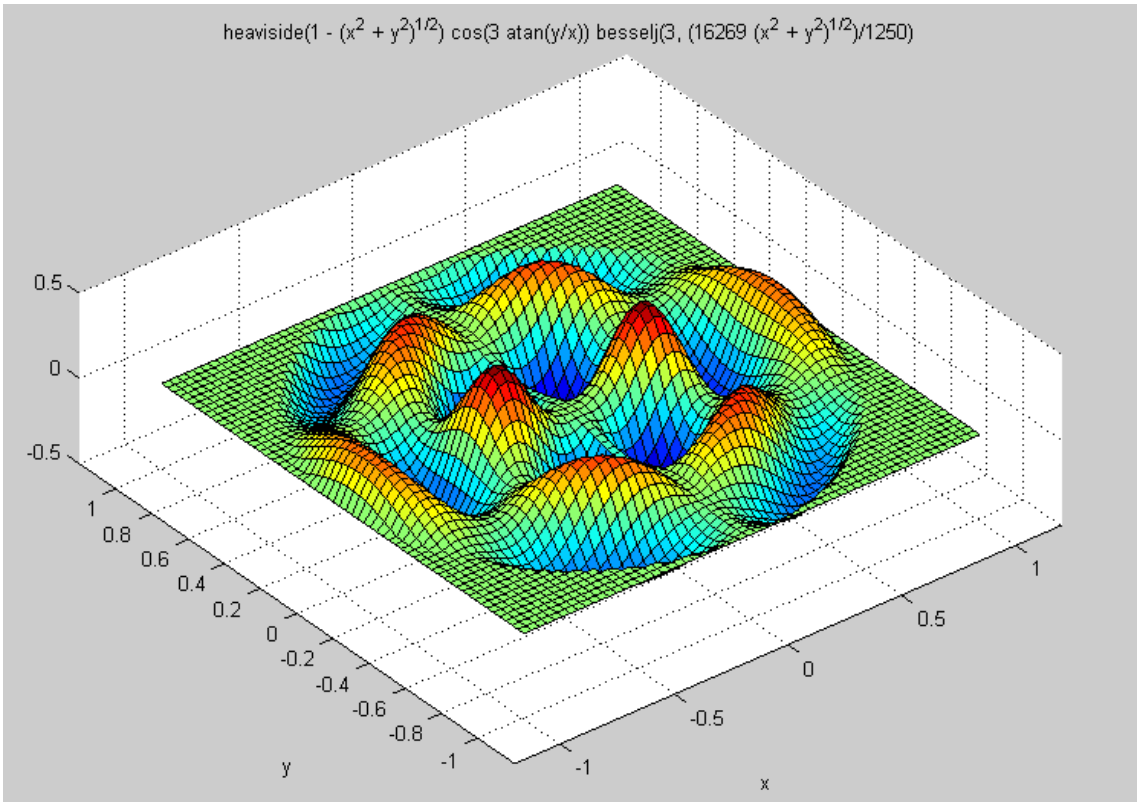
r =
(x^2 + y^2)^(1/2)

>> u=heaviside(1-r)*besselj(3,13.0152*r)*cos(3*theta)

u =
heaviside(1 - (x^2 + y^2)^(1/2))*cos(3*atan(y/x))*besselj(3, (16269*(x^2 + y^2)^(1/2))/1250)

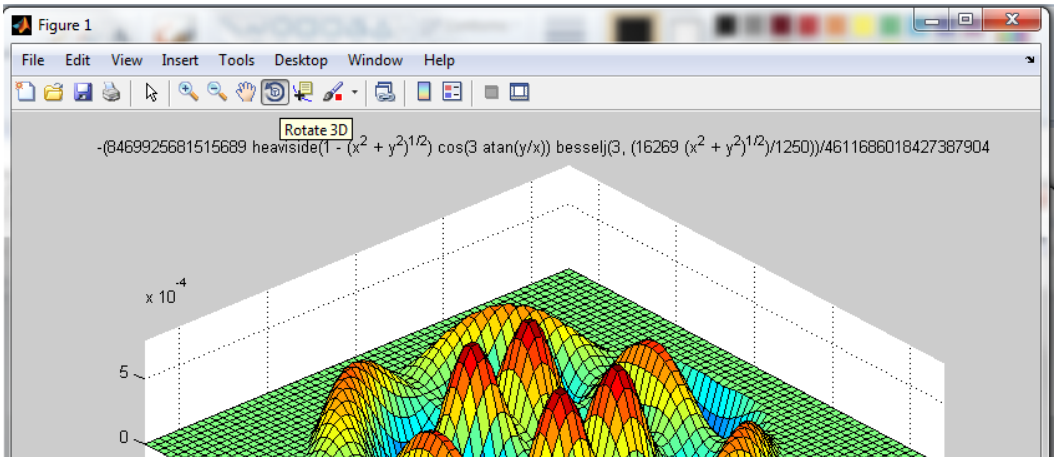
>> ezsurf(u,[-1,1,-1,1])

>> axis([-1.2,1.2,-1.2,1.2,-0.5,0.5])



% También puede introducirse la dependencia en t, siendo esta de arriba la forma de la membrana para t=0,

% Abajo la forma para t=15 y para t=pi/26, estas instrucciones pueden agruparse en un fichero para ver el movimiento, pero también utilizando el editor de gráficas (primero rotar, poner la figura de forma agradable y luego ir pulsando un cursor, tenemos la sensación de movimiento, aunque no tan precisa ni controlada para distintos t como con el fichero)



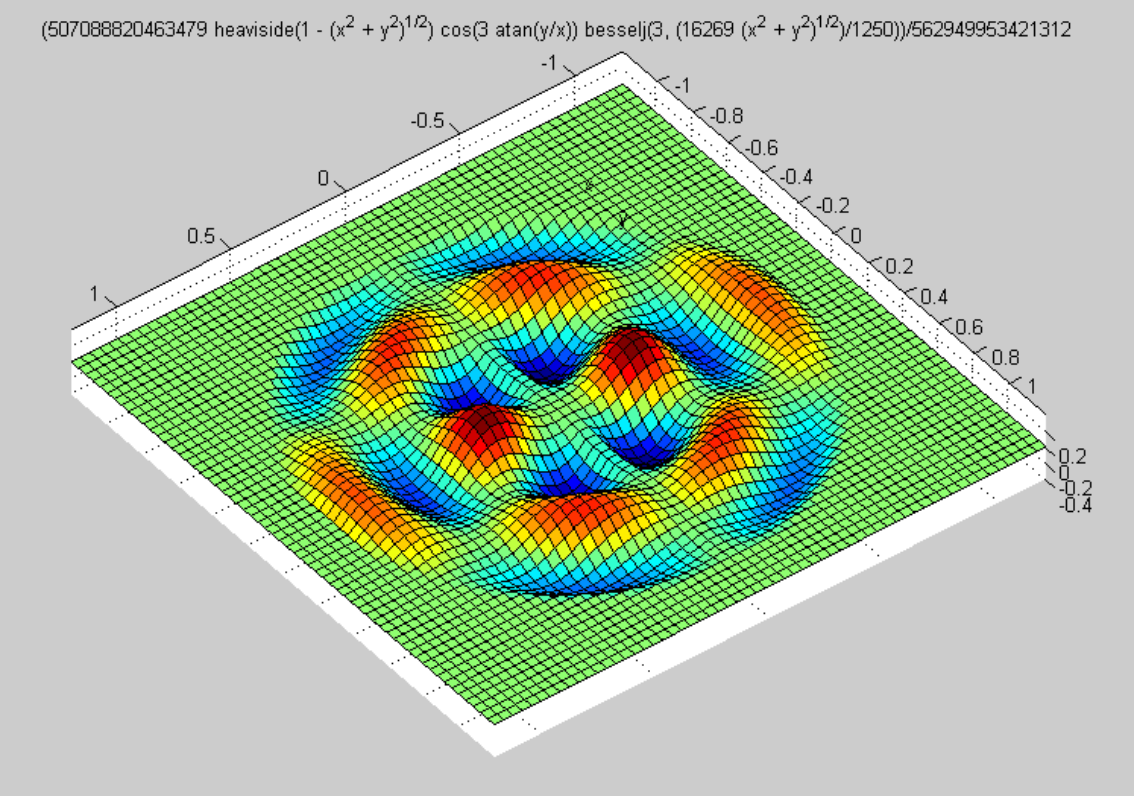
>> syms t

>> formamembra_en_t=heaviside(1-r)*besselj(3,13.0152*r)*cos(3*theta)*cos(13.0152*t)

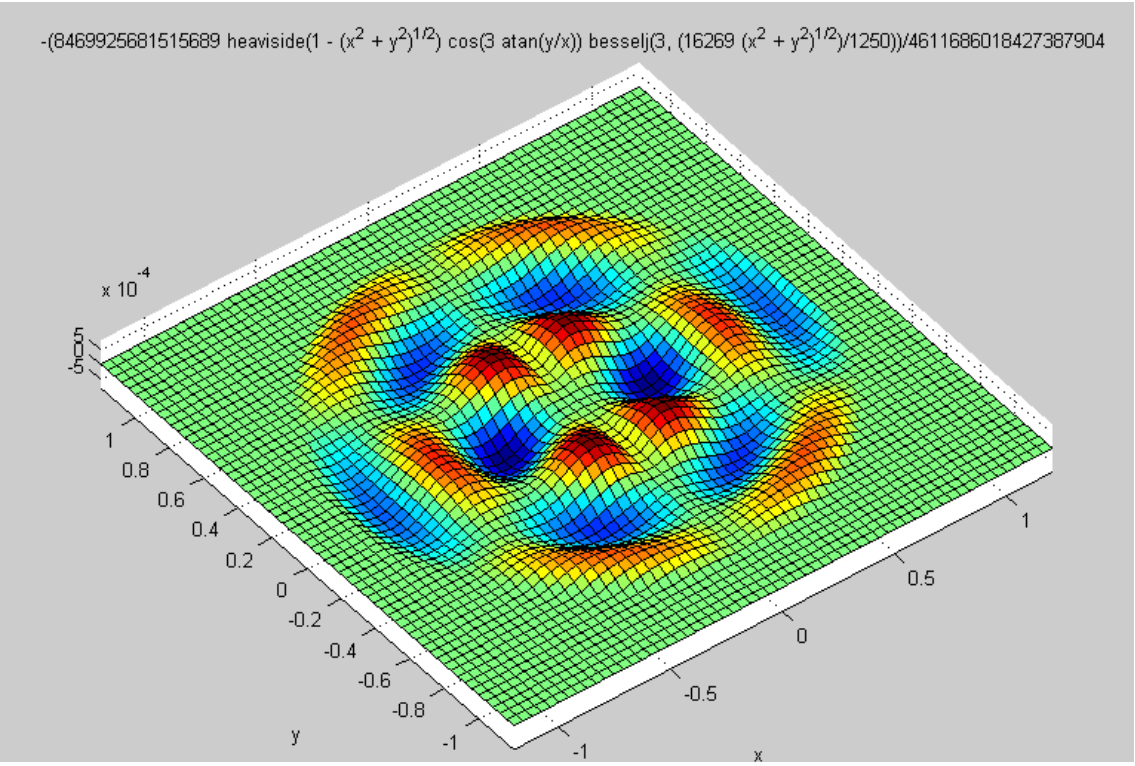
formamembra_en_t =

cos((16269*t)/1250)*heaviside(1 - (x^2 + y^2)^(1/2))*cos(3*atan(y/x))*besselj(3, (16269*(x^2 + y^2)^(1/2))/1250)

>> ezsurf(subs(formamembra_en_t,t,15),[-1.2,1.2,-1.2,1.2])



>> ezsurf(subs(formamembra_en_t,t,pi/(13*2)),[-1.2,1.2,-1.2,1.2])



% Para ver la simulación de las vibraciones en un intervalo de tiempo; esto es, la solución de:
u_tt-(u_xx+u_yy)=0, t>0, x^2+y^2<1,
u(x,y,t)=0 , si t>0, x^2+y^2=1,
*u(x,y,0)=J_3(13.0152*t)*cos(3*theta), x^2+y^2<1,*
u_t(x,y,0)=0 , x^2+y^2<1,
ejecutamos vibracionmembranaangular.m

>> vibracionmembranaangular ***% ver movie: vibramembranaang***

>> type vibracionmembranaangular

% Finalmente, proponemos un problema que modela las vibraciones de una membrana cuadrada donde se puede utilizar un desarrollo en serie de Fourier para resolverlo. Se supone que el dato inicial está localizado en un subdominio. Por ejemplo, supuesto el dominio D ocupado por la membrana es D= (0,1)x(0,1), la forma inicial de la membrana está dada por una función con soporte en (1/4,3/4)x(1/4,3/4),

$$f(x) = \begin{cases} (x - (\frac{1}{2}))^2 - \frac{1}{4^2} & \text{si } x \in [\frac{1}{4}, \frac{3}{4}] \\ 0 & \text{si } x \in [0,1],\ x \notin [\frac{1}{4}, \frac{3}{4}] \end{cases}$$

% la solución del problema

u_tt-(u_xx+u_yy)=0, t>0,
u(x,y,t)=0 , si t>0, x=0, o y= 0, o x=1 o y=1
u(x,y,0)= f(x)g(y), u_t(x,y,0)=0

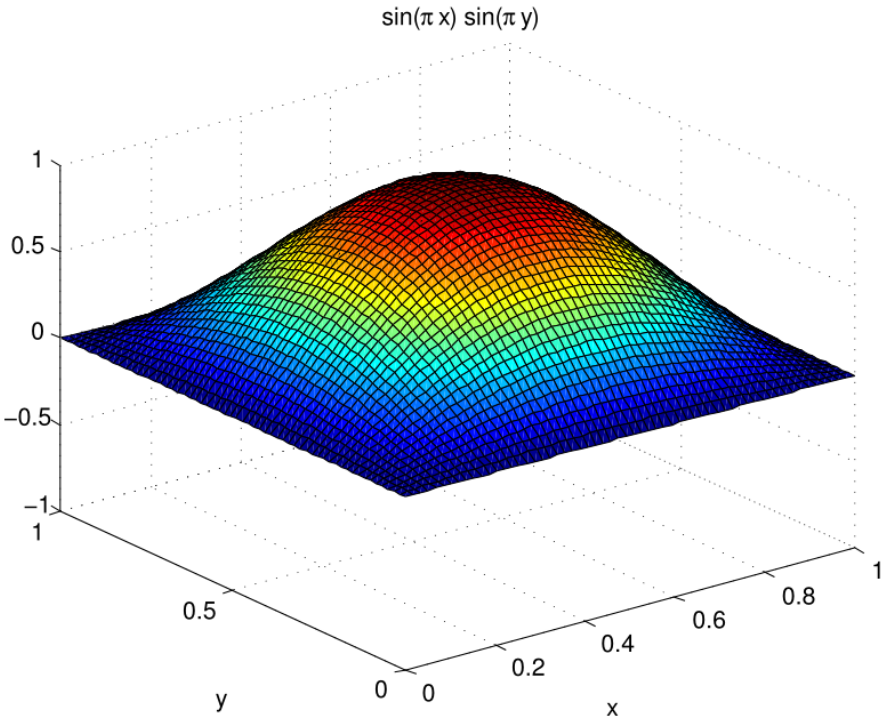
está dada por

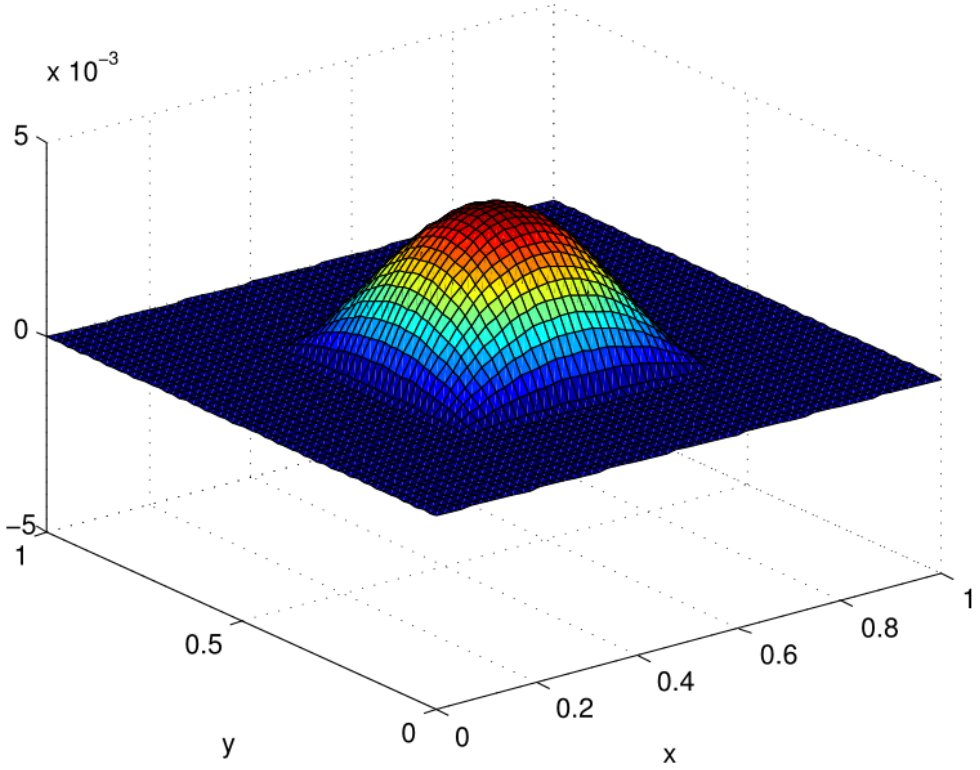
$$u(x,y,t)=\sum_{k=1}^{\infty}\sum_{n=1}^{\infty}\alpha_{k,n}\cos(\sqrt{n^2+k^2}\pi t)\sin(k\pi x)\sin(n\pi y)$$

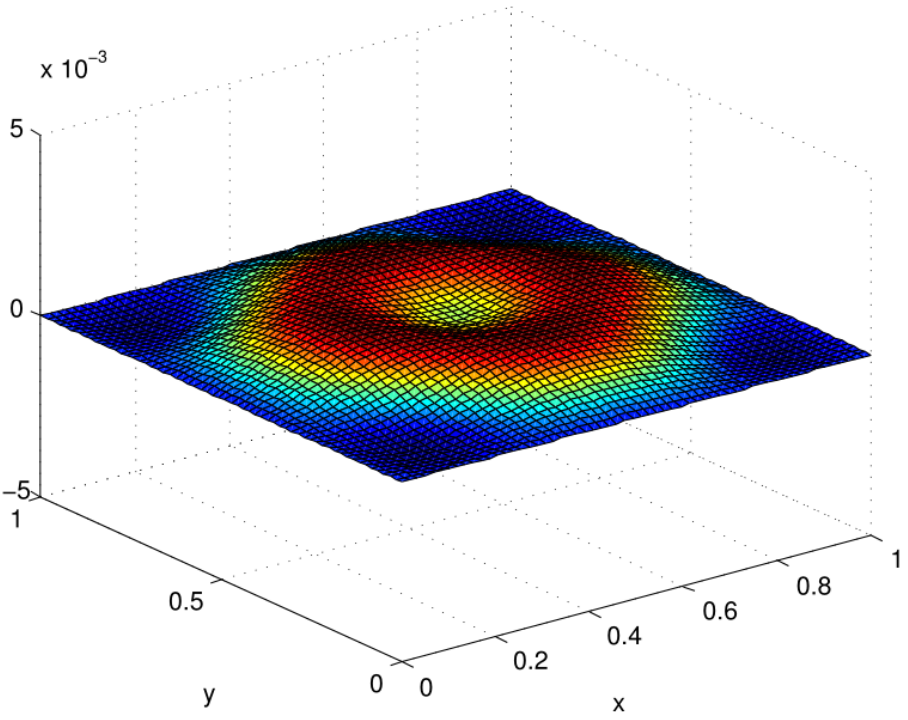
donde $\alpha_{k,n}$

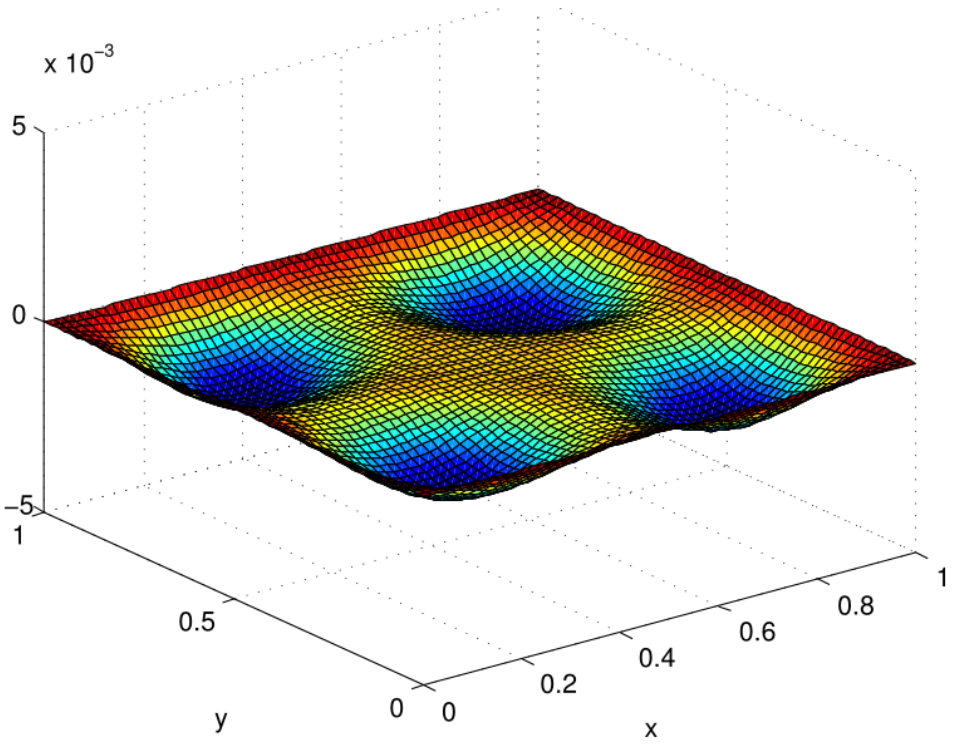
$$\alpha_{k,n}=4\int_{1/4}^{3/4}\int_{1/4}^{3/4}\left((x-(\frac{1}{2}))^2-\frac{1}{4^2}\right)\left((y-(\frac{1}{2}))^2-\frac{1}{4^2}\right)\sin(k\pi y)\sin(n\pi x)$$

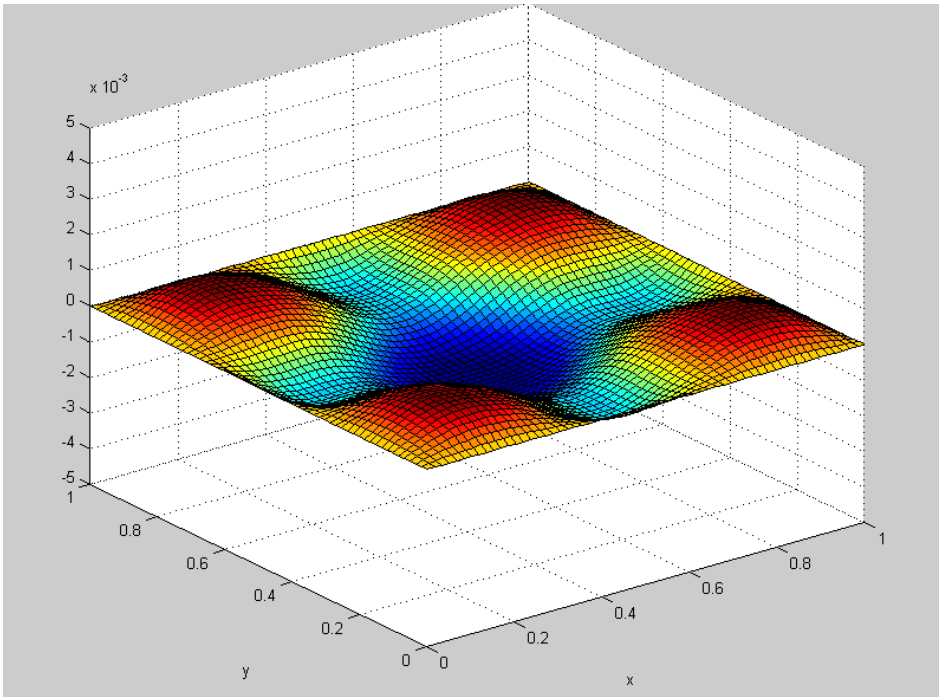
% Los siguientes dibujos muestran la gráfica del primer modo propio de vibración de la membrana y la gráfica de la solución del problema planteado (la forma de la membrana) en distintos instantes de tiempo t=0, t=3, t=5 y t=8




x 10^−3
5
0
−5
1
0.5
y
0
0
0.2
0.4
0.6
0.8
1
x


x 10^−3
5
0
−5
1
0.5
y
0
0
0.2
0.4
0.6
0.8
1
x





*% para simular las vibraciones de la membrana***% ver movie: vibramembranacuadrada**