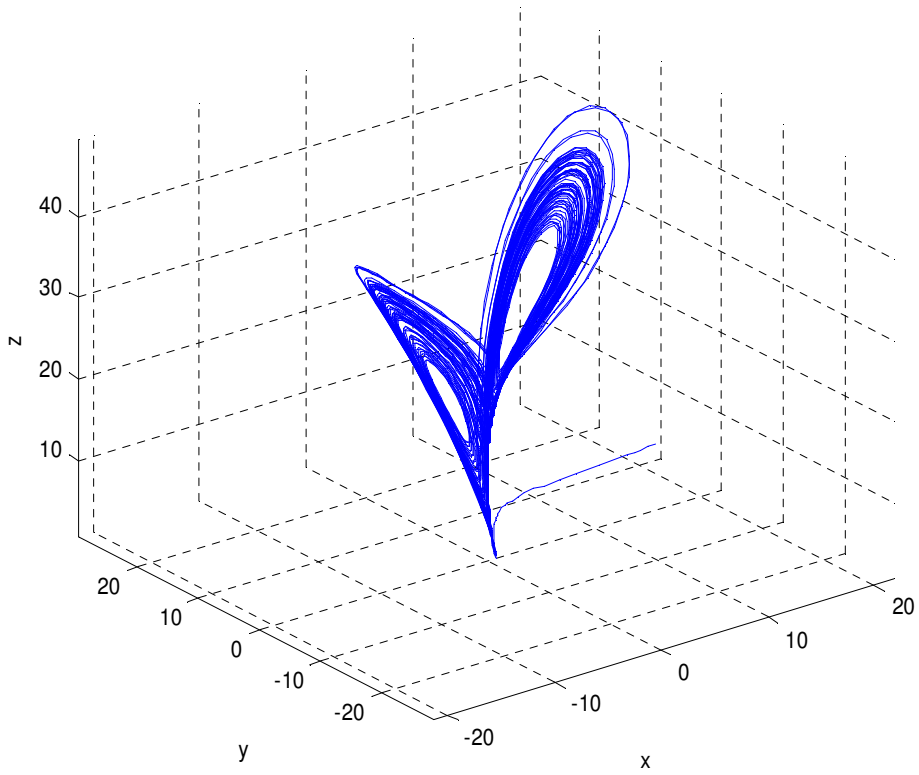
Carmelo Camacho González
Fernando J. Manobel Ponce



ECUACIONES DIFERENCIALES Y MÉTODOS NUMÉRICOS.

ALUMNOS: FERNANDO J.MANOBEL PONCE.

CARMELO CAMACHO GONZALEZ.

PROFESOR. ANTONIO ALGABA.

AÑO ACADEMICO 2008/2009

0 INDICE

1. RESOLUCIÓN NUMÉRICA DE ECUACIONES Y SISTEMAS DE ECUACIONES.

1.1.- RESOLUCIÓN NUMERICA DE ECUACIONES Y SISTEMAS DE ECUACIONES NO LINEALES.

Sea la ecuación general f(x)=0, a partir de un valor inicial p0, cercano a la solución de la ecuación dada p, el algoritmo de Newton Raphson, expuesto a continuación, nos da una aproximación a la solución exacta.

$$P_0 = P$$

$$P_n = P_{n-1} - \frac{f(p_{n-1})}{f'(P_{n-1})}, \quad n \geq 1,$$

Para sistemas de ecuaciones no lineales , con n-ecuaciones y n-incognitas, el algoritmo de Newton-Raphson quedará del siguiente modo.

$$P_0 = P$$

$$Df(p_n)Z_n = -f(p_n),$$

$$P_{n+1} = P_n + Z_n, \quad n \geq 0,$$

Ecuaciones no lineales. Problema 1.

Encontrar la única solución en [0,1] de la ecuación

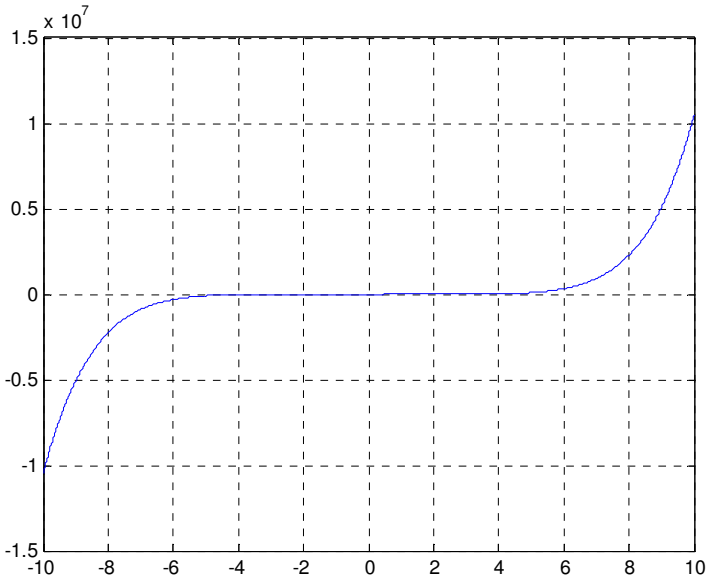
$$\boldsymbol{f(x) = x^7 + 5x^5 + 2x^2 + x - 1}$$

a) **Construir un gráfico de y=f(x), en el intervalo [-10,10], para obtener una idea de las raices existentes.**

Comenzamos definiendo las funciones en archivos, para posteriormente emplearlos en el algoritmo de N_R.

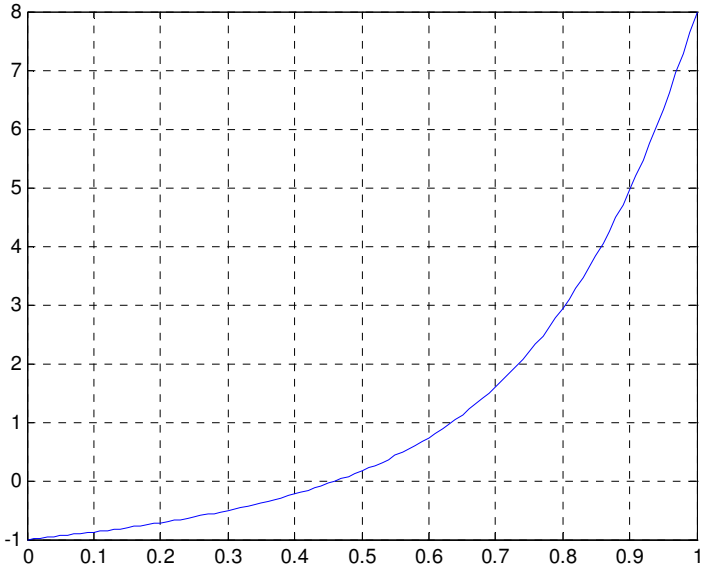
```
x=-10:0.01:10;
>> y=x.^7+5*x.^5+2*x.^2+x-1;
>> plot(x,y)
```

Carmelo Camacho González
Fernando J. Manobel Ponce



b) Construir un nuevo gráfico en [0,1] para obtener una nueva aproximación de la raiz de la ecuación:

```
x=0:0.01:1;
y=x.^7+5*x.^5+2*x.^2+x-1;
 plot(x,y)
 grid on
```



c) Aplicar N-R en p0=0,4, con Nmax=200 y un criterio de paro igual a:
[pn-pn-1]<10^-10,
[f(pn)]<10^-10

La solucion aproximada es:
p1 = 0.46234002719882 alcanzada en k = 5 iteraciones.

Ecuaciones no lineales. Problema 1 Propuesto

Hallar todas las raices de la ecuación $x^3 - 7x + 2 = 0$.
Realizar un análisis gráfico para calcular las aproximaciones iniciales necesarias en el método de Newton, tomando como criterio de paro |pn – pn-1|<10^-7, y 50 iteraciones máximo. Construir gráfico que muestre la convergencia de las iteraciones a una de las raices existentes.

Si derivamos la función, obtenemos la siguiente:
$f'(x) = 3x^2 - 7$

Vemos que tenemos un máximo en $x = -\sqrt{\frac{7}{3}}$

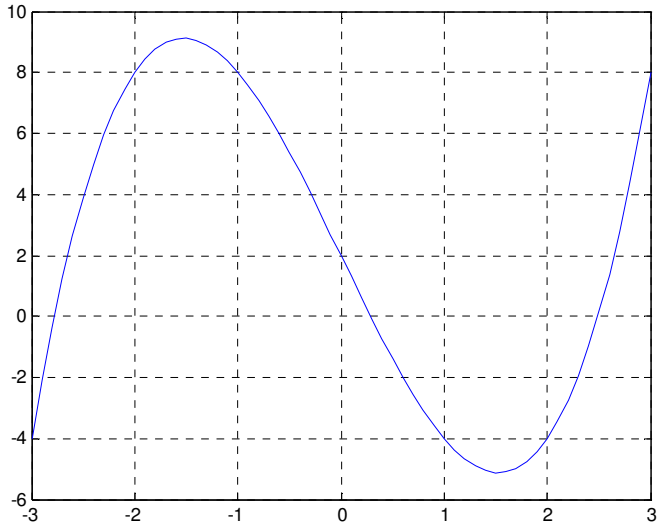
Un mínimo en el $x = \sqrt{\frac{7}{3}}$

Nos quedan por tanto los intervalos

$$\left(-\infty, -\frac{7}{3}\right)\left(-\frac{7}{3}, \frac{7}{3}\right)\left(\frac{7}{3}, \infty\right)$$

Vemos que la gráfica viene de –∞, y va hasta +∞, por lo que es posible hasta 3 raices reales en la función.

Por Bolzano, podemos demostrar numéricamente un cambio de signo en los tres intervalos, Observando la gráfica, podemos ver la existencia de 3 puntos de corte,.



El primer punto lo tenemos aproximadamente en x=-2.8.

El segundo punto lo tenemos aproximadamente en x=0.25

El tercer punto lo tenemos aproximadamente en x=2.5

Aplicamos el método de Newton para hallar las raices con unos errores inferiores a los indicados en el problema.

El algoritmo empleado para hallar las raíces es el siguiente:

NR.m.

```
format long
N=50; %numero maximo de iteraciones
TOL=10^(-7);
p0=-3;%condicion inicial, cercana a la solución que buscamos
c=0;
for k=1:N
p1=p0-f(p0)/fprima(p0);
if abs(p1-p0)<TOL & abs(f(p1))<TOL
'la solucion aproximada es', p1, 'alcanzada en',k,'iteraciones'
c=1;
break
end
p0=p1;
end
if c==0
'el metodo no converge para', N, 'iteraciones con la condicion inicial y TOL dada'
end
```

Introducimos los archivos para resolver la función:

f.m

```
function y=f(x)
y=x^3-7*x+2;
```

fprima.m

```
%Definimos la derivada de la función f
function y=fprima(x)
y=3*x.^2-7;
```

Comenzamos suponiendo un punto inicial p0=-3. Obtenemos:

```
NR
ans =
la solucion aproximada es
p1 =
  -2.77845711825839
alcanzada en
k =
     5
Iteraciones
```

Buscamos el siguiente punto, partiendo de p0=0 (Cambiaremos en la línea correspondiente del algoritmo, el valor inicial de búsqueda, para que la solución converga a la raíz buscada).

```
NR
la solucion aproximada es
p1 =
   0.28916854644831
alcanzada en
k =
     4
iteraciones
```

Vamos al tercer punto, partiendo de p0=2

```
NR
la solucion aproximada es
p1 =
   2.48928857181008
alcanzada en
k =
     6
iteraciones
```

Sistemas de ecuaciones no lineales. Problema 3.

Se trata de resolver el sistema:

$$3x - cos(yz) - \frac{1}{2} = 0$$
$$x^2 - 81(y + 0.1)^2 + sen(z) + 1.06 = 0$$
$$e^{-xy} + 20z + \frac{10\pi - 3}{3} = 0$$

Estudiar la convergencia del método de Newton-Raphson con las especificaciones: vector inicial $P_{0=}(0.1, 0.1, -0.1)$, y como criterio de paro $||P\_n - P\_(n-1)|| < 10^{-10}$ y un número máximo de iteraciones 1000.

Imprimir las 10 primeras iteraciones, asi como la norma infinito de la diferencia de dos iteraciones consecutivas.

Para hallar el problema por Newton-Rahpson, tendremos que aplicar la expresión siguiente:

$$P_{n+1} = P_n - \frac{f(p_n)}{f'(P_n)},$$

Tendremos que empezar metiendo los vectores en la fórmula anterior.

Observamos que tenemos una matriz con un vector, La derivada de la matriz será al Jacobiano de la misma:

Tendremos por tanto que aplicar 3 algoritmos:

- Hallar f(pn).

  ftrid.m

```
%Dentro de un sistema de ENL, será el primer algoritmo a emplear, crear la
%función.
%define la funcion tridimensional (llamada ftrid) p(x,y,z)----->f(f1,f2,f3)
function f=ftrid(p)
x=p(1);y=p(2);z=p(3);
f(1)=3*x-cos(y*z)-1/2; %Cada una de las funciones definidas en el problema.
f(2)=x^2-81*(y+0.1)^2+sin(z)+1.06;
f(3)=exp(-x*y)+20*z+(10*pi-3)/3;
```

- Hallar la derivada (Jacobiano).

  jactrid.m

```
%En un S EDNL, será el segundo programa a emplear, será la derivada de la
%función anterior, en matrices hablamos de jacobiano.
```

```
%este fichero jacftrid.m, nos define el jacobiano de ftrid
%metemos directamente las derivadas de la función.
function JA=jacftrid(p)
x=p(1);y=p(2);z=p(3);
JA(1,:)=[3 z*sin(y*z) y*sin(y*z)];
JA(2,:)=[2*x -162*(y+0.1) cos(z)];
JA(3,:)=[-y*exp(-x*y) -x*exp(-x*y) 20];
```

- Aplicar la función de Newton.

Newtonprob3.m

```
%Solucion por el metodo de Newton-Raphson, el sistema esta definido en
%ftrid y el jacobiano de f en jacftrid
'Introducir por teclado la tolerancia'
TOL=input('TOL=');
'Condicion inicial'
x=input('condicion inicial x=');
c=1;
z=-jacftrid(x)\ftrid(x)';
while norm(z,inf)>=TOL
x=x+z';
z=-jacftrid(x)\ftrid(x)';
c=c+1;
if c>1000 %si en 1000 iteraciones no converge mensaje de error
error('no se alcanzo dicha tolerancia')
end
end
'se alcanzo la tolerancia ', TOL
'en la iteracion,'
c
'solucion aproximada,'
x+z'
```

Una vez definidos todos los algoritmos, aplicamos los mismos.
Nos basta con aplicar el último, pues hace llamada a las funciones definidos en los otros dos.

newtonprob3
Introducir por teclado la tolerancia

```
TOL=10^-10
Condicion inicial
condicion inicial x=[0.1 0.1 -0.1]
se alcanzo la tolerancia
TOL =
   1.000000000000000e-010
en la iteracion,
c =
    6
solucion aproximada,
   5.000000000000000e-001    9.036283026834329e-018   -5.235987755982988e-001
```

Para imprimir las 10 primeras iteraciones, así como la norma entre dos iteraciones consecutivas, aplicaremos un algoritmo, el cual nos volverá a pedir las condiciones iniciales, así como el nº de iteraciones, obteniendo los resultados en 4 columnas.
La última columna nos muestra el error o norma entre dos iteraciones consecutivas.

imprimirEDNL.m

```
%Para imprimir un nº de iteraciones definidas, así como el error obtenido,
%lo haremos del siguiente modo.
x=input('condicion inicial x=')
n=input('número de iteraciones')
for k=1:n %Será el nº de iteraciones que queramos imprimir
   y=x-(jacftrid(x)\ftrid(x)')'; %De nuevo aplicamos N-R de un moo distinto al anterior
   A(k,:)=[y norm(x-y,inf)];
   x=y;
end
A
```

```
>> imprimirEDNL
condicion inicial x=[0.1 0.1 -0.1]
número de iteraciones10

A =
   4.998696729264286e-001    1.946684853741811e-002   -5.215204719358306e-001
4.215204719358306e-001
   5.000142401642189e-001    1.588591370293900e-003   -5.235569643476383e-001
1.787825716712421e-002
```

5.000001134678342e-001 1.244478332153407e-005 -5.235984500728894e-001
1.576146586972366e-003
5.000000000070757e-001 7.757857169962316e-010 -5.235987755780071e-001
1.244400753581707e-005
5.000000000000000e-001 4.281872937125380e-018 -5.235987755982989e-001
7.757857127143586e-010
5.000000000000000e-001 9.036283026834329e-018 -5.235987755982988e-001
1.110223024625157e-016
5.000000000000000e-001 -9.442926788531291e-018 -5.235987755982989e-001
1.110223024625157e-016
5.000000000000000e-001 9.036283026834333e-018 -5.235987755982988e-001
1.110223024625157e-016
5.000000000000000e-001 -9.442926788531288e-018 -5.235987755982989e-001
1.110223024625157e-016
5.000000000000000e-001 9.036283026834336e-018 -5.235987755982988e-001
1.110223024625157e-016

Sistema de ecuaciones no lineales. Problema propuesto nº3

Sea el sistema de ecuaciones no lineales:

F(x,y)=xy-x-y+1=0

***G*(*x, y*) = 3x²y-x²-12xy+4x+12y-4=0**

Aplicar el método de Newton con criterio de paro $||P\_n - P\_(n-1)|| < 10^{-10}$ **y las condiciones iniciales**

P = [0,9*;* 0,4]*;*

Al igual que en el caos anterior comenzaremos definiendo los algoritmos, y posteriormente aplicaremos N-R.

ftrid1.m

```
function f=ftrid1(p)
x=p(1);y=p(2);
f(1)=x*y-x-y+1; %Cada una de las funciones definidas en el problema.
f(2)=3*x.^2*y-x.^2-12*x*y+4*x+12*y-4;
```

jactrid1.m

```
function JA=jacftrid1(p)
x=p(1);y=p(2);
JA(1,:)=[y-1 x-1];
JA(2,:)=[6*x*y-2*x-12*y+4 3*x^2-12*x+12];
```

Newtonprob3_1.m

```
%Solucion por el metodo de Newton-Raphson, el sistema esta definido en
%ftrid y el jacobiano de f en jacftrid
'Introducir por teclado la tolerancia'
TOL=input('TOL=');
'Condicion inicial'
x=input('condicion inicial x=');
c=1;
z=-jacftrid1(x)\ftrid1(x)';
while norm(z,inf)>=TOL
x=x+z';
z=-jacftrid1(x)\ftrid1(x)';
c=c+1;
if c>1000 %si en 1000 iteraciones no converge mensaje de error
error('no se alcanzo dicha tolerancia')
end
end
'se alcanzo la tolerancia ', TOL
'en la iteracion,'
c
'solucion aproximada,'
x+z'
```

```
newtonprob3_1
Introducir por teclado la tolerancia
TOL=10^-10
Condicion inicial
condicion inicial x=[0.9,0.4]
se alcanzo la tolerancia
TOL =
   1.000000000000000e-010
en la iteracion,
c =
    5
solucion aproximada,
  1.00000000000000   0.33333333333333
```

1.2.- RESOLUCIÓN DE SISTEMAS DE ECUACIONES LINEALES.

Teniendo un sistema de n-ecuaciones con n-incognitas. Representaremos el sistema como Ax=b.

Los métodos iterativos que vamos a plantear transforman el sistema dado en otro equivalente de la forma x=Tx+c, para una matriz fija T y un vector c.

Para ver si T converge, tendremos que verificar que el radio espectral del mismo es menor de 1.

Radio espectral lo definimos como el máximo de los autovalores de la matriz.

Estudiaremos tres métodos distintos con la resolución del problema expuesto, unos pueden converger y otros no.

La forma definida anteriormente será los métodos de Jacobi, y de Gauss-Seidel (fijos).

El tercer método introducido será el denominado algoritmo de relajación, y consiste, en aplicar al método de gauus-Seidel una condición particular . (w=1), el método convergerá si 0<w<2, en los demás casos también podría converger.

Procedemos a la realización del problema aplicando cada uno de los métodos indicados anteriormente

Problema 2.

Una persona espera impacientemente a otra que llega con retraso, de manera que camina nerviosa a lo largo de una calle, se mueve hacia la izquierda con frecuencia triple que a la derecha. Supongamos posiciones numeradas en la calle de 0 a 20, de manera que, partiendo de *x*0 = 1 y *x*20 = 0, las posiciones a lo largo del tiempo se obtendrán resolviendo el sistema tridiagonal:

$$x_k = \frac{3}{4}x_{k-1} + \frac{1}{4}x_{k+1}$$

(1) Resolver el sistema planteado utilizando el operador \ de Matlab.

Para ésto comenzamos definiendo un algoritmo que defina el sistema como una matriz tridiagonal.

También conocemos la matriz b.

Tenemos la opción de montar por tanto el sistema Ax=b, y hallar x, operando con Matlab de la forma A\B.

sistemaproblema2.m

```
function[A,B]=sistemaproblema2(n) %para pedir los datos
A(1,1)=1; % primer elemento.El resto de la fila es cero, por defecto
A(n,n)=1; % ultimo elemento.Si no lo definimos el resto de elementos es cero.
for i=2:n-1 % el bucle empiza en la segunda fila y termina en la penultima
for j=1:n
if i==j %si actua un comaprador lógico, el == es doble
A(i,j)=-1; %si se cumple, para aquí.Esta es la diagonal principal
elseif i==j-1 %si no se cumple continua.
A(i,j)=0.25; %ésta es una por encima de la diagonal
elseif i==j+1
A(i,j)=0.75; %ésta es una por debajo de la diagonal
end
end
end
B=zeros(n,1);%ésta matriz la hemos construido como columna, tiene n fila y una columna
B(1)=1; %aquí hemos metido el valor de la columna
```

```
[A,B]=sistemaproblema2(21)

A =
   1.0000        0        0        0        0        0        0        0        0        0
   0.7500   -1.0000    0.2500        0        0        0        0        0        0        0
        0    0.7500   -1.0000    0        0        0        0        0        0        0
             ..........................................................
             ..........................................................
        0        0        0        0        0    0.7500   -1.0000    0.2500        0        0    0
        0        0        0        0        0        0    0.7500   -1.0000    0.2500        0    0
        0        0        0        0        0        0        0    0.7500   -1.0000    0.2500    0
        0        0        0        0        0        0        0        0    0.7500   -1.0000        0
        0        0        0        0        0        0        0        0        0        0        0.2500
        0                                                                                        1.0000
        0                         .                                .                              .          0
B =  [  1  0  0  0  0.....................0]'
```

Una vez obtenido el sistema de éste modo en Matlab, operamos para hallar x.

```
A\B
1.00000000000000
   0.99999999942641
   0.99999999770562
   0.99999999254327
```

0.99999997705622
0.99999993059508
0.99999979121164
0.99999937306132
0.99999811861037
0.99999435525753
0.99998306519898
0.99994919502336
0.99984758449648
0.99954275291584
0.99862825817392
0.99588477394817
0.98765432127091
0.96296296323914
0.88888888914382
0.66666666685787
0

(2) Aplicar el método iterativo de Jacobi para obtener la solución, analizando previamente la convergencia del método.

Una vez que llegamos a ése punto, para continuar aplicando un método lo normal es verificar antes la convergencia del mismo.

Para la convergencia hallaremos el radio espectral, mediante el algoritmo correspondiente:

radioespectraljacobi.m

```
function re=radioespectraljacobi
%
% Datos de salida
% re es el radio espectral de la matriz de Jacobi
%
format long
% lee la matriz de los coeficientes del sistema de ecuaciones Ax=B
A=sistemaproblema2(8);
% calculo de las matrices D,L,U
D=diag(diag(A)); L=tril(A)-D; U=triu(A)-D; %cogemos parte de la matriz+diag, que es lo que hace tril o triu,
% calculo del radio espectral
re=max(abs(eig(-inv(D)*(L+U))));%ésta función es la definida para hallar T por Jacobi.
```

```
re=radioespectraljacobi
re =
  0.85536319397709
```

Observamos que el mismo es menor que 1, por lo que previsiblemente el método debe converger.

A continuación aplicaremos el algoritmo correspondiente de Jacobi, para resolver el problema anterior.

La Matriz T para Jacobi, es de la forma:

T=D^-1*(L+U)

Ésta expresión la hemos aplicado en el algoritmo anterior para el radio espectral, por lo que tenemos definido las matrices L y U.

Necesitamos para aplicar Jacobi, únicamente la condición inicial P, a partir del cual el método comenzará a iterar hasta hallar la solución:

P=zeros(21,1);

Desde aquí podremos aplicar el algoritmo de Jacobi para ver la solución obtenida:

jacobi.m

```
function X=Jacobi(P,tol,max)
%
% Datos de entrada
% P es el dato inicial, un vector de orden nx1, es la condicion inicial del
% sistema.
% tol es la tolerancia, condicion de parada, diferencia entre dos
% operaciones.
% max es el numero maximo de iteraciones, condicion de parada
%
% Datos de salida
% X es una matriz de orden nx1 con la aproximacion a la
% solucion de AX=B obtenida por el metodo de Jacobi
%
% en el ejemplo P=zeros(21,1), rellena un vector columna con ceros
format long
% lee la matriz A, matriz cuadrada invertible de orden n,
% y el vector B, de orden nx1, al ejecutar el fichero sistemaproblema2
[A,B]=sistemaproblema2(8); % toma del sistemaproblema2 las dos matrices
n=length(B);
c=0;
for k=1:max
% construccion de la siguiente iteracion por el metodo de Jacobi
for j=1:n
X(j)=(B(j)-A(j,[1:j-1,j+1:n])*P([1:j-1,j+1:n]))/A(j,j); % formula matematica de jacobi
end
% condicion de paro
```

```
ea=abs(norm(X'-P)); % valora absoluto de la norma (' es la traspuesta)
if ea<tol % comparador
disp('Se alcanzo la tolerancia con exito en la iteracion')
k
disp('y la solucion aproximada es')
X=X';
c=1;
break % fin de bucle for
else
P=X';
end
end
% no cumplimiento de la condicion de paro
if c==0
disp('El metodo no cumple el criterio de paro propuesto para')
max
disp('iteraciones.')
end
```

```
X=Jacobi(P,10^-10,200)
Se alcanzo la tolerancia con exito en la iteracion
k =
   179
y la solucion aproximada es
X =
   1.00000000000000
   0.99999999942639
   0.99999999770559
   0.99999999254317
   0.99999997705603
   0.99999993059461
   0.99999979121085
   0.99999937305956
   0.99999811860759
   0.99999435525167
   0.99998306519020
   0.99994919500580
   0.99984758447144
   0.99954275286838
   0.99862825811010
   0.99588477383528
   0.98765432113193
```

0.96296296302187

0.88888888892474

0.66666666663335

0

(3) Aplicar el método iterativo de Gauss-Seidel para obtener la solución, analizando previamente la convergencia del método.

Al igual que en el método anterior, tendremos que definir el radio espectral para éste método, lo haremos con el algoritmo correspondiente.
Recordemos que para definir la T por Gauss-Seidel, lo haremos de la forma:

T=(D+L)^-1 * U

```
function re=radioespectralGaussSeidel
% re es el radio espectral de la matriz de GaussSeidel
%
format long
% lee la matriz de los coeficientes del sistema de ecuaciones Ax=B
A=sistemaproblema2(8);
% calculo de las matrices D,L,U
D=diag(diag(A)); L=tril(A)-D; U=triu(A)-D;
% calculo del radio espectral
re=max(abs(eig(-inv(D+L)*U)));
```

```
re=radioespectralGaussSeidel
re =
0.73164619361068
```

Al ser de valores inferiores a 1, el método debe converger sin problema.
Aplicamos ahora el algoritmo de gauss-Seidel, Al igual que antes debemos partir de un punto inicial, por lo que también empleamos:

P=zeros(21,1);.

Gausseidel.m

```
function X=gaussSeidel(P,tol,max)
%
% Datos de entrada
% P es el dato inicial, un vector de orden nx1
% tol es la tolerancia
% max es el numero maximo de iteraciones
```

```
%
% Datos de salida
% X es una matriz de orden nx1 con la aproximacion a la
% solucion de AX=B obtenida por el metodo de GaussSeidel
%
format long
[A,B]=sistemaproblema2(8);
n=length(B);
c=0;
for k=1:max
% construccion de la siguiente iteracion por el metodo de GaussSeidel
for j=1:n
if j==1
X(1)=(B(1)-A(1,2:n)*P(2:n))/A(1,1);
elseif j==n
X(n)=(B(n)-A(n,1:n-1)*(X(1:n-1))')/A(n,n);
else
X(j)=(B(j)-A(j,1:j-1)*(X(1:j-1))'-A(j,j+1:n)*P(j+1:n))/A(j,j);
end
end
% condicion de paro
ea=abs(norm(X'-P));
if ea<tol
disp('Se alcanzo la tolerancia con exito en la iteracion')
k
disp('y la solucion aproximada es')
X=X';
c=1;
break
else
P=X';
end
end
% no cumplimiento de la condicion de paro
if c==0
disp('El metodo no cumple el criterio de paro propuesto para')
max
disp('iteraciones.')
end
```

X=GaussSeidel(P,10^-10,200)

Se alcanzo la tolerancia con exito en la iteracion

k =

83

y la solucion aproximada es

X =

1.00000000000000
0.99999999942633
0.99999999770540
0.99999999254279
0.99999997705531
0.99999993059345
0.99999979120888
0.99999937305682
0.99999811860325
0.99999435524658
0.99998306518257
0.99994919499936
0.99984758446226
0.99954275286838
0.99862825811010
0.99588477386556
0.98765432116920
0.96296296312277
0.88888888902648
0.66666666676986
0

(4) Experimenta con distintos valores de w en el método S.O.R. para acelerar la convergencia. Calcular el radio espectral de la matriz del método.

Al igual que en los casos anteriores elaboramos y aplicamos el algoritmo para ver el rádio espectral del método.

Para hallar el radio espectral, aplicaremos un valor inicial de w, (0<w<2), recordemos que el caso de Gauss-Seidel es igual que éste, pero en el caso de G-S, w=1.

Para hallar T=((w*L+D)^-1*(w-1)*D+w*U).

Aplicamos una w=1.4, hallando el siguiente radio espectral:

radioespectralSOR.m

```
function re=radioespectralSOR(w)
%
% Datos de entrada
```

```
% w es el valor de omega, esta entre 0 y 2, y hay que probar que valor de
% omega da un r espectral mas proximo a cero, que tendrá mejor grado de
% convergencia.
% Datos de salida
% re es el radio espectral de la matriz de Jacobi
%
format long
% lee la matriz de los coeficientes del sistema de ecuaciones Ax=B
A=sistemaproblema2(8);
% calculo de las matrices D,L,U
D=diag(diag(A)); L=tril(A)-D; U=triu(A)-D;
% calculo del radio espectral
re=max(abs(eig(-inv(w*L+D)*((w-1)*D+w*U))));
```

```
re=radioespectralSOR(1.4)
re =
   0.40000000000000
```

(Los resultados obtenidos con 1,3, o 1,5 son mayores que los obtenidos con éste valor).

Observamos que inicialmente la convergencia debe ser más rápida que con los métodos anteriores, pues el radio espectral es más cercano a 0.

Tras indicar el mismo valor inicial de P, aplicamos el algoritmo de SOR, obteniendo el siguiente resultado:

sor.m

```
function X=SOR(P,w,tol,max)
%
% Datos de entrada
% P es el dato inicial, un vector de orden nx1
% w es el valor de omega
% tol es la tolerancia
% max es el numero maximo de iteraciones
%
% Datos de salida
% X es una matriz de orden nX1 con la ultima iteracion en la aproximacion
% a la solucion de AX=B obtenida por el metodo SOR
%
format long
% lee la matriz A, matriz cuadrada invertible de orden n,
% y el vector B, de orden nx1, al ejecutar el fichero sistemaproblema2
[A,B]=sistemaproblema2(8);
```

```
n=length(B);
c=0;
for k=1:max
%construccion de la siguiente iteracion por el metodo de SOR
for j=1:n
if j==1
X(1)=(1-w)*P(j)+ w*(B(1)-A(1,2:n)*P(2:n))/A(1,1);
elseif j==n
X(n)=(1-w)*P(j)+ w*(B(n)-A(n,1:n-1)*(X(1:n-1))')/A(n,n);
else
X(j)=(1-w)*P(j)+ w*(B(j)-A(j,1:j-1)*(X(1:j-1))' ... % los tres puntos es para decir que la instrucción
sigue en la siguiente
-A(j,j+1:n)*P(j+1:n))/A(j,j);
end
end
%condicion de paro
ea=abs(norm(X'-P));
if ea<tol
disp('Se alcanzo la tolerancia con exito en la iteracion')
k
disp('y la solucion aproximada es')
X=X';
c=1;
break
else
P=X';
end
end
if c==0
disp('El metodo no cumple el criterio de paro propuesto para')
max
disp('iteraciones.')
end
```

```
X=SOR(P,1.4,10^-10,200)
Se alcanzo la tolerancia con éxito en la iteracion
k =
   31
y la solucion aproximada es
```

X =

1.00000000000046
0.99999999942792
0.99999999770619
0.99999999254513
0.99999997705696
0.99999993059737
0.99999979121257
0.99999937306409
0.99999811861148
0.99999435526081
0.99998306520005
0.99994919502822
0.99984758449612
0.99954275292197
0.99862825817431
0.99588477395570
0.98765432127089
0.96296296324755
0.88888888914348
0.66666666686808
0

(5) El problema descrito es un problema de frontera para una ecuación en diferencias, cuya solución exacta es:

$$x_k = 1 - \frac{3^k - 1}{3^{20} - 1}$$

Calcula estos valores para k = 0; 1; 2; .. ; 20 y compara los resultados con los obtenidos en los apartados anteriores.

Conociendo la función exacta, lo que haremos será preparar un algoritmo con el cálculo de la misma, y con una llamada a cada uno de los algoritmos antes empleados, de modo que obtengamos una diferencia de todos los errores obtenidos en las operaciones anteriores.

Problema2a5.m

```
function error=problema2a5
% calculo de la solucion exacta X
X(1)=1; for k=2:21
X(k)=1-(3^(k-1)-1)/(3^20-1); % el bucle for se reajusta porque en el contador del
```

```
% for no puede empezar en cero como la formula iterativa de las soluciones del sistema dado.
end
% calculo de las aproximaciones
P=zeros(21,1);
S2=jacobi(P,10^(-10),200);
S3=gausssseidel(P,10^(-10),200);
S4=sor(P,1.4,10^(-10),100);
[A,B]=sistemaproblema2(21);
S1=A\B;
format long e
error=[abs(X'-S2),abs(X'-S3),abs(X'-S4),abs(X'-S1)];
%error, da 3 columnas de las comparaciones o diferencias entre la solucion
%real y las resultas por jacobi, gausssseidel, sor y el oprador de matlab
%A\B
```

```
error =
  Columns 1 through 3

                   0                   0   4.611866444292900e-013
  1.143529715363911e-014   7.516209876712310e-014   1.513789094076401e-012
  3.352873534367973e-014   2.197131365733185e-013   5.662137425588298e-013
  1.001421168211891e-013   4.779510121011299e-013   1.859290499339750e-012
  1.919575609576896e-013   9.162670622231417e-013   7.390754674929667e-013
  4.672928710647284e-013   1.631805801594055e-012   2.290945211314011e-012
  7.913669719528116e-013   2.763900219804327e-012   9.319212068703564e-013
  1.764144386129374e-012   4.506395256953510e-012   2.769229290322528e-012
  2.787992059438693e-012   7.120970479945754e-012   1.109112801600531e-012
  5.859979168576501e-012   1.094790924582867e-011   3.287259353612626e-012
  8.784528660044089e-012   1.641009550468198e-011   1.070366018041113e-012
  1.755784406753946e-011   2.399669352115552e-011   4.858446978062148e-012
  2.503386387076034e-011   3.421263272684882e-011   3.587130592563881e-013
  4.746192328042298e-011   4.746192328042298e-011   6.129097229745639e-012
  6.381761785689832e-011   6.381761785689832e-011   3.910205492729801e-013
  1.128847015863244e-010   8.260847561558649e-011   7.534639578921087e-012
  1.389811599139534e-010   1.017078643528180e-010   1.643130076445232e-014
  2.172707569414456e-010   1.163639185008947e-010   8.412048835282349e-012
  2.190814196723068e-010   1.173352526251392e-010   3.380629109983602e-013
  2.245132968425878e-010   8.800149498000565e-011   1.021771556253270e-011
                   0                   0                   0

  Column 4

                   0
```

0
1.110223024625157e-016
1.110223024625157e-016
2.220446049250313e-016
1.110223024625157e-016
3.330669073875470e-016
4.440892098500626e-016
4.440892098500626e-016
4.440892098500626e-016
4.440892098500626e-016
6.661338147750939e-016
6.661338147750939e-016
5.551115123125783e-016
5.551115123125783e-016
5.551115123125783e-016
5.551115123125783e-016
5.551115123125783e-016
5.551115123125783e-016
4.440892098500626e-016
0

Sistema de ecuaciones lineales. Ejercicio 2 propuesto.

2. Consideremos el sistema de ecuaciones Ax = b, dado por

$$\begin{pmatrix} 10 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 4 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 4 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 4 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 4 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 4 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 4 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 10 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{pmatrix} = \begin{pmatrix} 9 \\ 5 \\ -2 \\ 2 \\ 1 \\ 6 \\ 2 \\ -10 \end{pmatrix}.$$

a) Resolverlo utilizando tanto el método de Jacobi como el de Gauss-Seidel con las siguientes especificaciones para ambos casos: el vector inicial es el origen, la tolerancia es 10^-10*;* medida en la norma euclidea, y el número máximo de iteraciones que se permite es 100. ¿Convergen ambos métodos para las especificaciones dadas?

b) Resolver el sistema dado mediante el método SOR para *w* = 1.03 obtenido en el apartado anterior con las especificaciones del apartado (a).

Al igual que en el ejercicio anterior tendremos que comenzar creando el sistema en Matlab. El mismo queda definido, una vez modificado el algoritmo:

sistemaproblema3.m

```
function[A,B]=sistemaproblema3(n) %para pedir los datos
for i=2:n-1 % el bucle empiza en la segunda fila y termina en la penultima
for j=2:n-1
   A(:,1)=1;
   A(:,n)=1;
   A(1,:)=1;
   A(n,:)=1;
   A(1,1)=10;
   A(n,n)=10;

if i==j %si actua un comaprador lógico, el == es doble
A(i,j)=4; %si se cumple, para aquí.Esta es la diagonal principal
elseif i==j-1 %si no se cumple continua.
A(i,j)=1; %ésta es una por encima de la diagonal
elseif i==j+1
A(i,j)=1; %ésta es una por debajo de la diagonal
end
end
end

B=[9,5,-2,2,1,6,2;-10];%ésta matriz la hemos construido como columna
```

```
[A,B]=sistemaproblema3(8)

A =
   10    1    1    1    1    1    1    1
    1    4    1    0    0    0    0    1
    1    1    4    1    0    0    0    1
    1    0    1    4    1    0    0    1
    1    0    0    1    4    1    0    1
    1    0    0    0    1    4    1    1
    1    0    0    0    0    1    4    1
    1    1    1    1    1    1    1   10
B =
    9
    5
   -2
```

```
    2
    1
    6
    2
  -10
```

Comenzaremos verificando el sistema por el método de Jacobi, para ello comenzamos calculando el radio espectral para asegurar que converge:

```
re=radioespectraljacobi
re =
   0.83919946455443

X=Jacobi(P,10^-10,50)
El metodo no cumple el criterio de paro propuesto para
max =
    50
iteraciones.
X =
   0.70497636584656
   1.67231850455223
  -0.98828054873707
   0.98174324825884
  -0.23776894686311
   1.67025603662879
   0.25768435821076
  -1.40613474526456
```

Observamos que el método no consigue bajar del error en el número de iteraciones propuestas aunque el radio espectral es menor que 1.

A continuación probaremos con el método de Gauss-Seidel, siguiendo los mismos criterios que anteriormente:

```
re=radioespectralGaussSeidel
re =
   0.22199069994159

X=gaussSeidel(P,10^-10,50)
Se alcanzo la tolerancia con exito en la iteracion
k =
    18
y la solucion aproximada es
```

```
X =
   0.70499911676157
   1.67234092055045
  -0.98825080460730
   0.98177517546201
  -0.23773701965781
   1.67028578075817
   0.25770677420716
  -1.40611199434742
```

Vemos que para el caso de Gauss Seidel, si se cumplen las condiciones propuestas.

Siguiendo el apartado b, resolveremos el problema por SOR, para w=1.03.

```
re=radioespectralSOR(1.03)
re =
   0.19147292778787

X=SOR(P,1.03,10^-10,50)
Se alcanzo la tolerancia con exito en la iteracion
k =
    16

y la solucion aproximada es
X =
   0.70499911676823
   1.67234092053369
  -0.98825080460557
   0.98177517545630
  -0.23773701965450
   1.67028578075220
   0.25770677420430
  -1.40611199434565
```

2.- ECUACIONES DIFERENCIALES DE PRIMER ORDEN

2.1.- ESTUDIO DE BIFURCACIONES Y PUNTOS DE EQUILIBRIO. APLICACIÓN DE 'EZPLOT'

1.- Consideremos ecuación diferencial x = f(x,L), calcular los puntos de bifurcación y dibujar el diagrama de bifurcaciones correspondiente cuando:

a) $\boldsymbol{f(x,L) = 3x^3 - 3x - L}$

Hallar los puntos de equilibrio y sus estabilidades cuando

$\boldsymbol{L = -2, 0, \frac{2\sqrt{3}}{3}}$

Comenzamos analizando la ecuación inicial, verificando si existen bifurcaciones y cual es el tipo , para ello, debemos saber que para que exista una bifurcación se debe cumplir:

$$f(x,L) = 0$$

$$\frac{\partial f(x,L)}{\partial x} = 0$$

$$\mathrm{f(x,L)} = 3\mathrm{x}^3 - 3\mathrm{x} - \mathrm{L} = 0$$

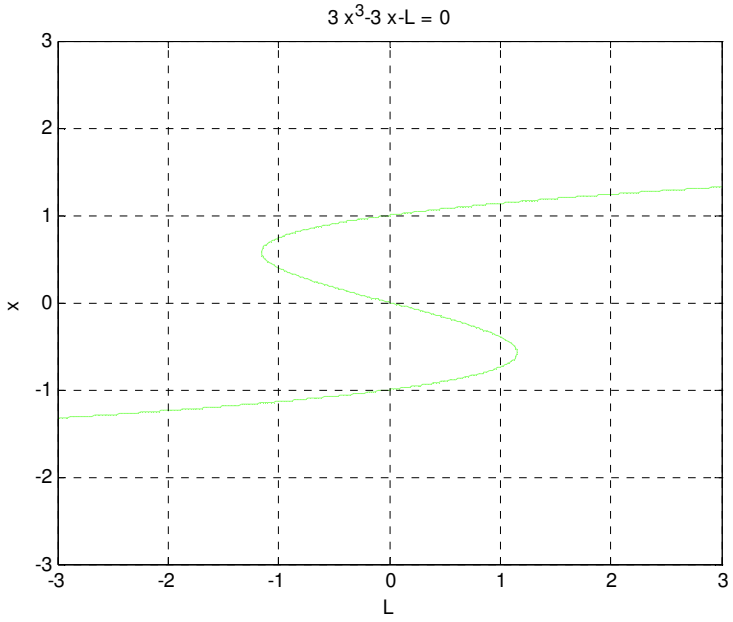
$$\frac{\partial f(x,L)}{\partial x} = 9x^2 - 3 = 0 \qquad x = \pm\frac{1}{\sqrt{3}}$$

$$\text{Si } x = +\frac{1}{\sqrt{3,}}, \quad L = -\frac{2}{\sqrt{3}}$$

$$\text{Si } x = -\frac{1}{\sqrt{3,}}, \quad L = \frac{2}{\sqrt{3}}$$

Podemos asegurar por tanto que tenemos dos puntos de bifurcación.
Comenzaremos analizando el diagrama de bifurcación de la función anterior, verificacndo que los puntos anteriores coinciden con un cambio de comportamiento en cuanto a los puntos de equilibrio, para ello emplearemos la función ezplot, del siguiente modo:

```
>>ezplot('3*x^3-3*x-L',[-3,3,-3,3])
>> grid on
```



En ésta gráfica confirmamos que en los puntos calculados anteriormente, pasamos de 1 a 2 ó de 2 a 3 puntos de equilibrio.

La bifurcación es del tipo histéresis.

Estudiaremos la estabilidad de la bifurcación, del siguiente modo:

Los puntos de bifurcación son, según hemos hallado antes $x = \pm\frac{1}{\sqrt{3}}$

También tenemos que $\frac{\partial f(x,L)}{\partial x} = 9x^2 - 3$

Para cualquier valor $x < -\frac{1}{\sqrt{3}}$, $\frac{\partial f(x,L)}{\partial x} > 0$, $Inestable$

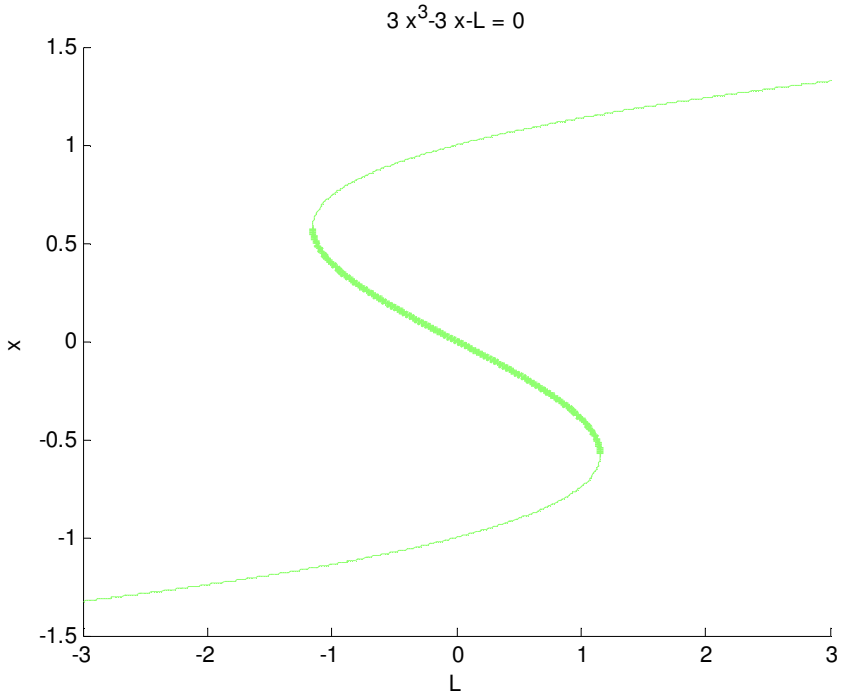
Para cualquier valor $-\frac{1}{\sqrt{3}} < x < \frac{1}{\sqrt{3}}$, $\frac{\partial f(x,L)}{\partial x} < 0$, $Estable$

Para cualquier valor $x > \frac{1}{\sqrt{3}}$, $\frac{\partial f(x,L)}{\partial x} > 0$, $Inestable$

Quedará por tanto del modo siguiente: (La parte gruesa será la zona de la bifurcación estable).

Para representarla por trozos, hemos aplicado ezplot del siguiente modo:

```
ezplot('3*x^3-3*x-L',[-3,2/sqrt(3)],[-3,-1/sqrt(3)])
hold on
ezplot('3*x^3-3*x-L',[-2/sqrt(3),2/sqrt(3)],[-1/sqrt(3),1/sqrt(3)])
ezplot('3*x^3-3*x-L',[-2/sqrt(3),3],[1/sqrt(3),3])
```



Siguiendo con el problema, le daremos valores a la función, confirmando los diferentes equilibrios en función de L

Observamos que la ecuación es no lineal, por lo que tendremos que emplear un sistema de resolución, como, Newton-Raphson.

Cuando L=-2, la función queda como:

$\boldsymbol{f(x) = 3x^3 - 3x + 2}$

La derivada de la función queda:

$\frac{df(x)}{dx} = 9x^2 - 3$

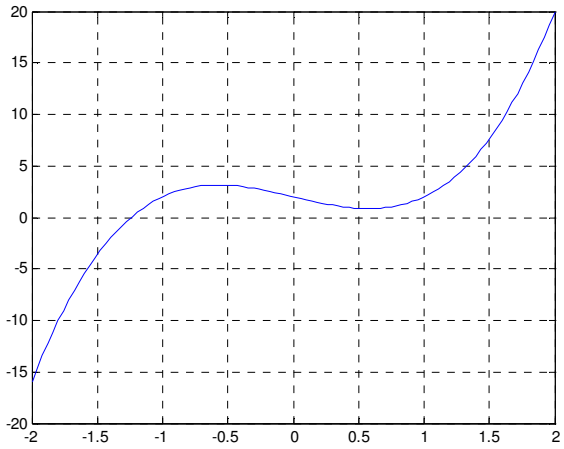
Según la gráfica anterior, nos debe salir una sola raíz inestable.

Introducimos los datos en una gráfica para confirmar el número de raíces.

```
>> x=linspace(-2,2,100);
>> f=3*x.^3-3*x+2;
>> plot(x,f)
>> grid on
```

Obtenemos una sola raíz.

Carmelo Camacho González
Fernando J. Manobel Ponce



Para hallar la solución exacta de la ecuación lo haremos empleando el algoritmo deNewton Raphson, empleado en el apartado 1.1 de éste documento.

NR.m.
(Nº máximo iteraciones 50, error 10^-4).

Introducimos los archivos para resolver la función:
f.m

```
function y=f(x)
y=3*x^3-3*x+2;
```

fprima.m

```
%Definimos la derivada de la función f
function y=fprima(x)
y=9*x.^2-3;
```

Introduciendo los datos en el algoritmo, partiendo de $P_0 = -1.5$, quedan los siguientes parámetros:

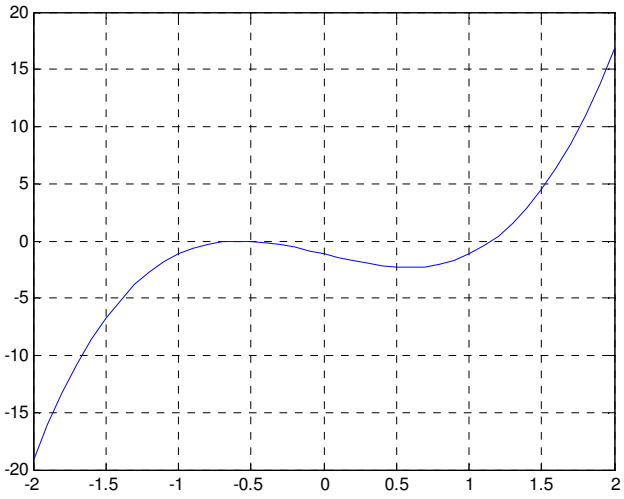
```
NR
ans =
la solucion aproximada es
p1 =
  -1.24001180975226
ans =
alcanzada en
k =
```

4

iteraciones

Éste será un punto de equilibrio de la ecuación diferencial, sustituyendo el mismo en la derivada, nos sale >0, por lo que el equilibrio es **inestable**.

Haciendo la misma operación para $L = \frac{2\sqrt{3}}{3}$, Analizando la gráfica anterior, observamos que éste punto de cambio a 2 raices,Debemos tener un punto de equilibrio estable, y otro inestable. Comenzamos representando la función:



Aparentemente tiene dos raices, lo verificamos con N-R (Cond. Inicial p0=-1 y p0=1).
Tendremos que cambiar las funciones:

f.m

```
function y=f(x)
y=3*x^3-3*x-(2*sqrt(3)/3);
```

fprima.m en éste caso se mantiene.

```
NR
ans =
la solucion aproximada es
p1 =
  -0.57741365250348
alcanzada en
```

k =

13

iteraciones

Si sustituimos en la derivada de la función, observamos que es <0, por lo tanto el punto de equilibrio es **estable**.

Buscamos el segundo punto (p0=1)

>> NR

la solucion aproximada es

p1 =

1.15470053838797

alcanzada en

k =

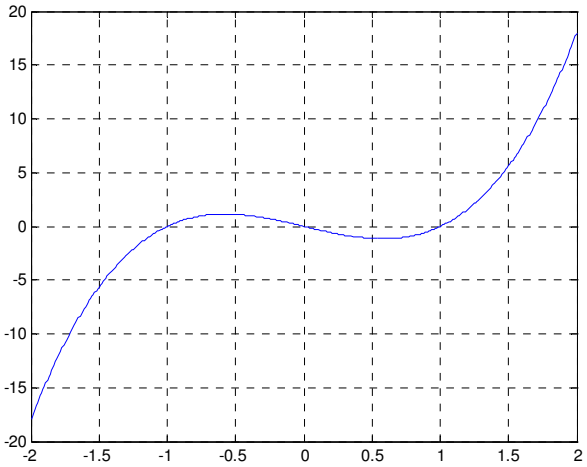
4

iteraciones

Sustituyendo en la derivada de la función, obtenemos que es>0,por lo tanto el punto de equilibrio es **inestable**.

Repitiendo la misma operación con L=0, observando la gráfica de bifurcaciones, debemos obtener 3 soluciones, 2 inestables, y una estable entre los valores -0.58<x<0.58.

La representación nos queda del siguiente modo.



Aquí tendremos en principio 3 puntos de equilibrio, en 0,1 y -1

En el punto 0, la derivada saldrá <0, el punto será **estable**.

En los puntos 1 y -1 la derivada es>0, el equilibrio es **inestable**.

Queda por tanto demostrado el estudio de bifurcaciones

.

b) Suponemos a continuación la ecuación diferencial:

$$\boldsymbol{f(x,L)=(x-L)(x^2-L)}$$

Hallar los puntos de equilibrio y sus estabilidades cuando

$$\boldsymbol{L=-2,0,\frac{2\sqrt{3}}{3}}$$

Comenzamos analizando la ecuación inicial, verificando si existen bifurcaciones y cual es el tipo , para ello, debemos saber que para que exista una bifurcación se debe cumplir:

$$f(x,L)=0$$

$$\frac{\partial f(x,L)}{\partial x}=0$$

$$\mathrm{f(x,L)=(x-L)(x^2-L)=0}$$

$$\frac{\partial f(x,L)}{\partial x}=3x^2-2xL-L=0 \qquad x=\frac{L\pm\sqrt{L(L+3)}}{3}$$

Si $L=0,\quad x=0\quad Tendremos\ un\ punto\ de\ equilibrioen\ (0,0)$

Si $L<0,\quad x=L,\quad Tendremos\ un\ solo\ punto\ de\ equilibrio$

Si $L>0,\quad Tendremos\ 3\ puntos\ de\ equilibrio$

Por el estudio realizado, el tipo de bifurcación que tenemos corresponde a una Pitchfork, pasamos de 1 a 3 puntos de equilibrio.
A continuación, estudiaremos las estabilidades:

Tenemos que $\frac{\partial f(x,L)}{\partial x}=3x^2-2xL-L$

Para cualquier valor $L<0,\ Para\ todo\ valor\ de\ x\ \ \frac{\partial f(x,L)}{\partial x}>0,\qquad Inestable$

Para $L=0,\quad Para\ todo\ valor\ de\ x,\ \ \frac{\partial f(x,L)}{\partial x}>0,\qquad Inestable$

Para $L>0\ ,\qquad x=\frac{L\pm\sqrt{L(L+3)}}{3}$

Si $x<\frac{L-\sqrt{L(L+3)}}{3}\qquad \frac{\partial f(x,L)}{\partial x}>0,\qquad Inestable$

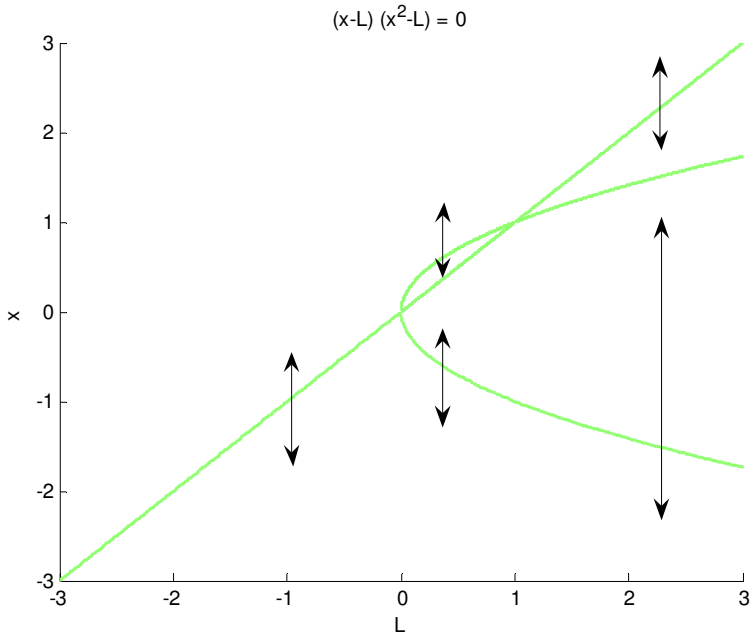
Si $\frac{L-\sqrt{L(L+3)}}{3}<x<\frac{L+\sqrt{L(L+3)}}{3}\qquad \frac{\partial f(x,L)}{\partial x}<0,\qquad Estable$

Si $x>\frac{L+\sqrt{L(L+3)}}{3}\qquad \frac{\partial f(x,L)}{\partial x}>0,\qquad Inestable$

Representamos la bifurcación con la ayuda de la función ‘ezplot’, quedará del siguiente modo:

```
ezplot('(x-L)*(x^2-L)',[-3,3],[-3,3])
```

Hemos marcado mediante flechas los equilibrios inestables.



A continuación, daremos valores a L, según las indicaciones del problema, verificando los resultados obtenidos en el diagrama de bifurcaciones.

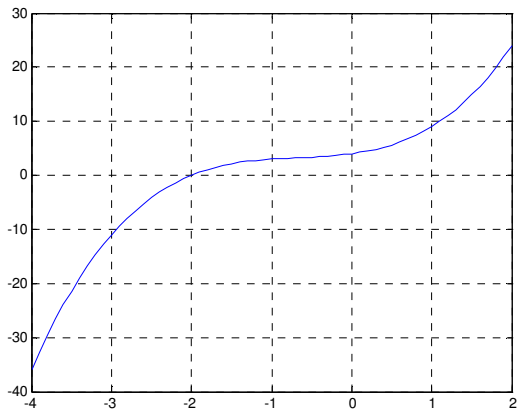
Para L=-2

$$f(x) = (x + 2)(x^2 + 2)$$

$$\frac{\partial f(x)}{\partial x} = 3x^2 + 4x + 2$$

Debemos obtener un único punto de equilibrio inestable.

Representamos la función mediante Matlab, y luego la resolvemos.



Tenemos un solo punto de equilibrio en -2.

Sustituyendo en la derivada, obtenemos un valor positivo, por lo que el equilibrio, tal y como esperábamos, es inestable.

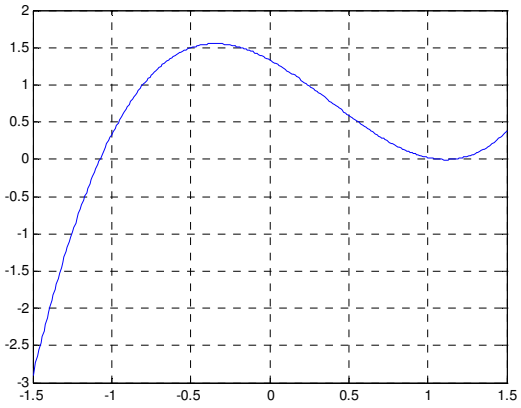
Para $L = \frac{2\sqrt{3}}{3}$

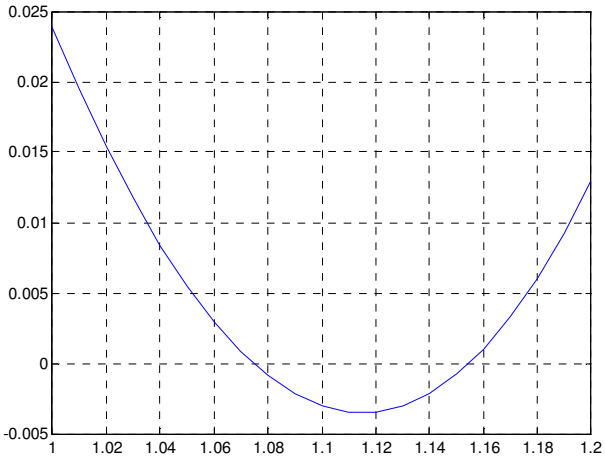
$$f(x) = \left(x - \frac{2\sqrt{3}}{3}\right)\left(x^2 - \frac{2\sqrt{3}}{3}\right)$$

$$\frac{\partial f(x)}{\partial x} = 3x^2 - \frac{4\sqrt{3}x}{3} - \frac{2\sqrt{3}}{3} = 0$$

Según el diagrama de bifurcaciones, debemos obtener 3 raices, siendo dos inestables y una estable, el equilibrio estable debe estar contenido en el intervalo definido por los dos puntos de equilibrio inestable.

Tras representar la función:





Observamos la existencia de 3 puntos e equilibrio.
Empleamos Newton Raphson, para conseguir los mismos, en éste caso con un error menor de 10e-4.

Partimos de p0=-1, obteniendo:

p1 =
-1.07456993182354.
Sustituyendo en la derivada, observamos que es >0, por lo que tenemos un **equilibrio inestable**.

Para ver el siguiente punto, partimos de p0=1.06, obteniendo.
p1 =
1.07456993182354.
Sustituyendo en la derivada obtenemos que es <0, por lo que tenemos un **equilibrio estable.**

Para ver el último punto partimos de p0=1.16, obteniendo:
p1 =
1.15470053837925
Sustituyendo en la derivada obtenemos un resultado >0, por lo que tenemos un **equilibrio inestable**.

Para $L = 0$

$$f(x) = x^3$$
$$\frac{\partial f(x)}{\partial x} = 3x^2$$

En éste caso, tenemos el punto de bifurcación justamente en el x=0, tal y como podemos ver en el diagrama de bifurcaciones correspondiente, es el punto límite en el que pasamos de 1 a 3 puntos de equilibrio (Pitchfork).

c) Suponemos a continuación la ecuación diferencial:

$$\boldsymbol{f(x, L) = (L - x^3)}$$

Hallar los puntos de equilibrio y sus estabilidades cuando

$\boldsymbol{L = -2, 0, \frac{2\sqrt{3}}{3}}$

Comenzamos analizando la ecuación inicial, verificando si existen bifurcaciones y cual es el tipo , para ello, debemos saber que para que exista una bifurcación se debe cumplir:

$$f(x, L) = 0$$

$$\frac{\partial f(x, L)}{\partial x} = 0$$

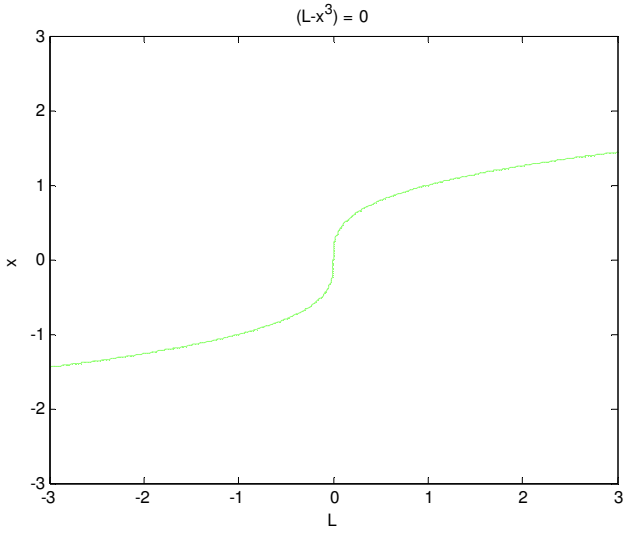
$$\mathrm{f(x, L)} = (\mathrm{L} - \mathrm{x}^3)=0$$
$$\frac{\partial f(x,L)}{\partial x} = -3x^2 = 0$$

Para $x = 0, \quad L = 0 \quad Tendremos\ un\ punto\ de\ bifurcación\ en\ (0,0)$

En éste caso podemos observar, que tenemos un punto de equilibrio para todos los valores de L y x (excepto en el 0,0), además todos los equilibrios son estables, pues la derivada de la función es siempre negativa.

Si representamos la función con ayuda del 'ezplot', obtenemos la siguiente gráfica:

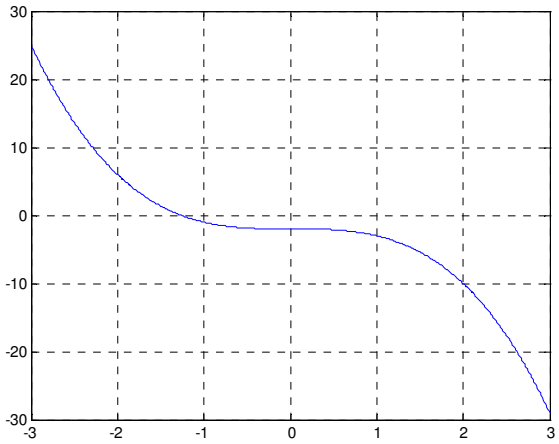
```
>> ezplot('(L-x^3)',[-3,3],[-3,3])
```



Para L=-2

$$\mathrm{f(x)} = (-2 - \mathrm{x}^3)$$
$$\frac{\partial f(x)}{\partial x} = -3x^2$$

Representamos la función con Matlab, podemos ver que obtenemos una sola raíz.



Por Newton-Raphson, hallamos la solución, partiendo de p0=-1, quedando

p1 =

-1.25992104989487.

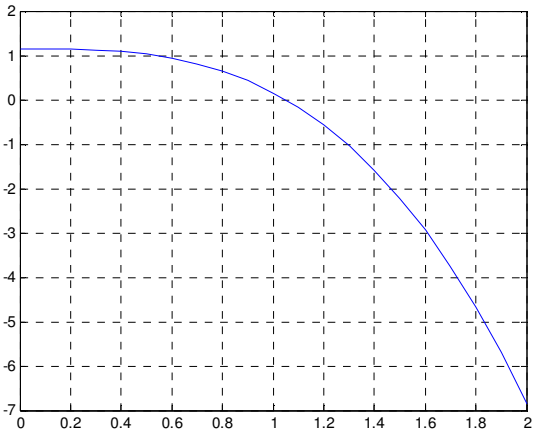
Sustituyendo en la derivada, obtenemos un valor <0, por o que tenemos un **equilibrio estable**.

Para $L = \frac{2\sqrt{3}}{3}$

$$f(x) = \left(\frac{2\sqrt{3}}{3} - x^3\right)$$

$$\frac{\partial f(x)}{\partial x} = -3x^2$$

Representamos la función observando que tenemos un único punto de equilibrio



Por N-R obtenemos el siguiente valor:

p1 =

1.04911506342165.

Sustituyendo en la derivada obtenemos un valor >0, por lo que el **equilibrio es estable**.

Para $L = 0$

$$\mathrm{f(x)} = -\mathrm{x}^3$$
$$\frac{\partial f(x)}{\partial x} = -3x^2$$

En éste caso, tenemos el punto de bifurcación justamente en el x=0.

d) Sea la ecuación unidimensional

$$\boldsymbol{\dot{x} = x(x^2 + \gamma x + \mu), x \in R; \gamma, \mu \in R}$$

Hallar equilibrios y estabilidades.

Observando la ecuación anterior, la misma es del tipo:

$$\dot{x} = \gamma + \mu x + x^3 = f(x, \gamma, \mu)$$

Por lo que será del tipo Cúspide

Para hallar los equilibrios: $\dot{x} = 0$

Soluciones:

$$x = 0$$
$$(x^2 + \gamma x + \mu) = 0, \qquad x = \frac{-\gamma \pm \sqrt{\gamma^2 - 4\mu}}{2}$$

Para que sea una bifurcación, no olvidemos que:

$$f'(x) = 3x^2 + 2\gamma x + \mu = 0$$

Viendo la primera solución: x=0,

$f'(x) = \mu = 0$, Será la primera solución.

Para las siguientes soluciones, resolveremos el sistema:

$$-(x^2 + \gamma x + \mu) = 0$$
$$3x^2 + 2\gamma x + \mu = 0$$
$$\_\_\_\_\_$$
$$2x^2 + \gamma x = 0 \longrightarrow \gamma = -2x$$

Sustituimos en $f'(x) = 3x^2 + 2\gamma x + \mu \longrightarrow 3x^2 - 4x^2 + \mu = 0 \longrightarrow x^2 = \mu$

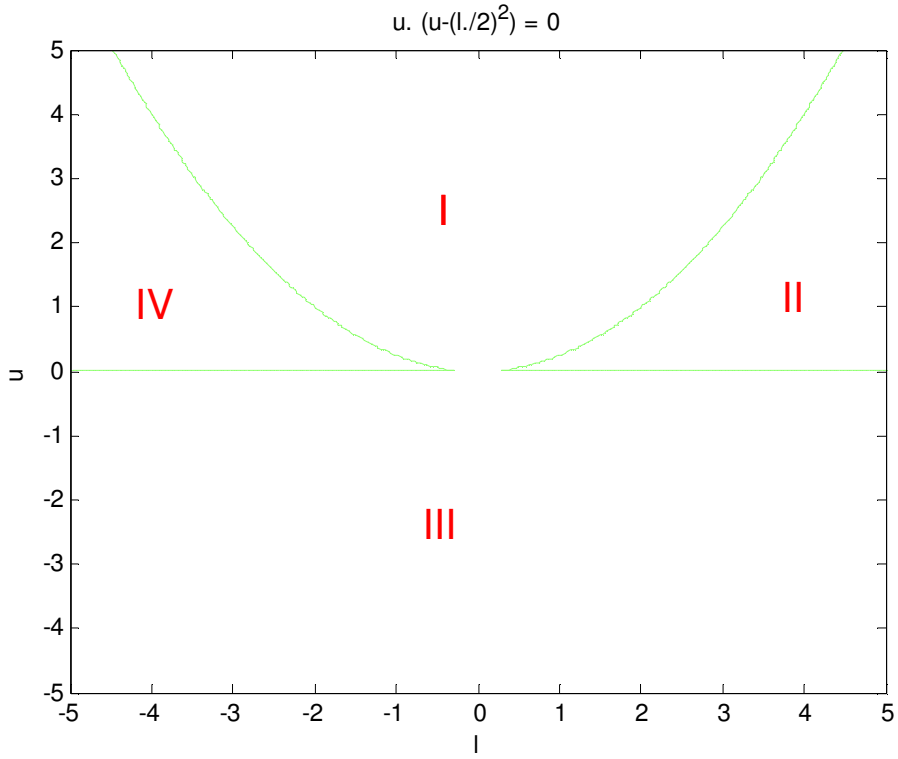
$$x = \frac{-\gamma}{2}, \qquad x = \sqrt{\mu}$$

$$\mu - \left(\frac{\gamma}{2}\right)^2 = 0$$

Con esto ya tenemos ambas soluciones, a continuación uniremos ambas y representaremos con Matlab:

$$\mu(\mu - \left(\frac{\gamma}{2}\right)^2) = 0$$

```
>> ezplot('u.*(u-(l./2)^2)',[-5,5],[-5,5])'
```



Observamos la representación de la bifurcación.

A continuación verificaremos el número de equilibrios existentes en cada una de las zonas marcadas:

Zona I

$\mu > 0$, $\gamma - 2\sqrt{\mu} < 0$ Para que se mantenga en el interior del area I.(Pues $\mu = \frac{\gamma^2}{4}$ es el límite del area)

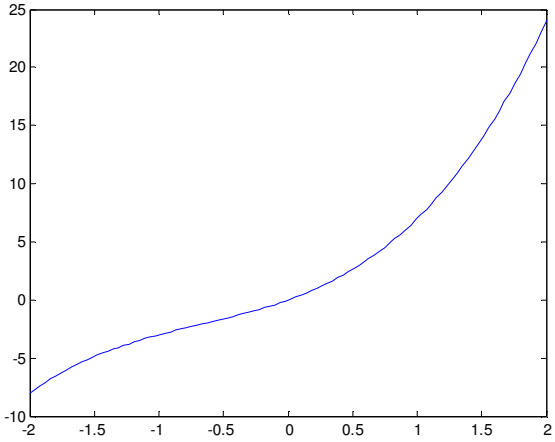
Para ello, $\gamma < 2\sqrt{\mu}$, X= $\frac{\gamma \pm \sqrt{\gamma^2 - 4\mu}}{2}$,

No existe solución para la ecuación anterior, por lo tanto, solo tendremos un punto de equilibrio en ésta zona

Para verificar lo anterior, damos valores que estén contenidos en la zona, para verificar lo anteriormente analizado. P.ejemplo $\gamma = 2, \mu = 4 \longrightarrow (Se\ entiende\ que\ \mu = u, \gamma = l)$

Nos queda por tanto la ecuación: $x(x^2 + 2x + 4) = 0$

```
>> x=linspace(-2,2,100);
>> y=x.*(x.^2+2*x+4);
>> plot(x,y)
```



Si resolvemos por Newton-Raphson, obtenemos la siguiente raíz:

```
p1 =
   7.602734853857181e-011
```

Estabilidades:

Para estudiar la estabilidad, tendremos que ver las posibles soluciones sobre la derivada:

$$f'(x) = 3x^2 + 2\gamma x + \mu.$$

Podemos ver que para cualquier valor que cumpla las condiciones anteriores, la derivada es mayor que cero, por lo que el equilibrio es inestable.

Zona II

Condiciones:

$$\gamma > 0, \qquad \mu > 0$$
$$\gamma - 2\sqrt{\mu} > 0$$

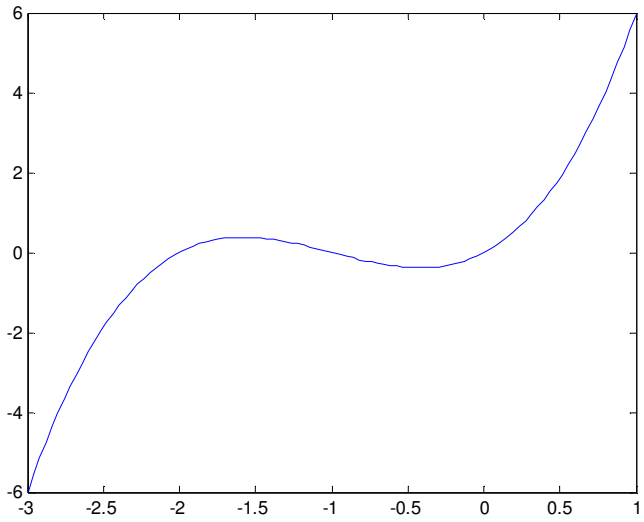
Observando, X= $\frac{\gamma \pm \sqrt{\gamma^2 - 4\mu}}{2}$, Vemos que la resolución de la ecuación de 2º grado tiene dos soluciones, por lo tanto en ésta zona, tendremos tres raíces.

Comprobamos dando valores a la ecuación, de modo que se cumplan las condiciones anteriores:

$$\gamma = 3, \mu = 2 \longrightarrow (Se\ entiende\ que\ \mu = u, \gamma = l)$$

Nos queda por tanto la ecuación: $x(x^2 + 3x + 2) = 0$

```
>> x=linspace(-3,1,100);
>> y=x.*(x.^2+3*x+2);
>> plot(x,y)
```



Verificamos con Newton-Raphson.

Volvemos a indicar el modo de hacerlo, en cuanto a los archivos f.m, y fprima.m, tendremos que modificarlo para cada zona que estimamos, pues cambian las ecuaciones.

Asimismo, los valores iniciales, tal y como hemos indicado anteriormente deben ser modificados en el algoritmo NR.

f.m

```
function y=f(x)
y=x*(x.^2+3*x+2);
```

fprima.m

```
function y=fprima(x)
y=3*x.^2+6*x+2;
```

Para P0=-2.5.

p1 =

-2.00000000001506

alcanzada en

k =

5

Para P0=-1.5.

la solucion aproximada es

p1 =

-1

alcanzada en

k =

2

iteraciones

Para P0=1

NR

la solucion aproximada es

p1 =

2.288975687763956e-012

alcanzada en

k =

6

Iteraciones.

Estabilidades:

Para estudiar la estabilidad, tendremos que ver las posibles soluciones sobre la derivada:

$$f'(x) = 3x^2 + 2\gamma x + \mu.$$

Si vemos la solución de la ecuación anterior, obtendremos:

$$x=\frac{-2\gamma\pm\sqrt{4\gamma^2-12\mu}}{6}=\frac{-\gamma\pm\sqrt{\gamma^2-3\mu}}{3}.$$

Si las soluciones x de la ecuación cumplen:

$x > \frac{-\gamma+\sqrt{\gamma^2-3\mu}}{3}$ entonces f'(x)>0, inestable.

$\frac{-\gamma-\sqrt{\gamma^2-3\mu}}{3} < x < \frac{-\gamma+\sqrt{\gamma^2-3\mu}}{3}$ entonces f'(x)<0, estable

$x < \frac{-\gamma-\sqrt{\gamma^2-3\mu}}{3}$ entonces f'(x)>0, Inestable

Esto se puede verificar aplicando el ejemplo con números, estudiado anteriormente.

$$\gamma = 3, \mu = 2$$

$$f'(x) = 3x^2 + 6x + 2.$$

$$x = \frac{-6 \pm \sqrt{12}}{6}, \quad x_1 = -0.42264, \; x_2 = -1.5773$$

Si verificamos con los resultados obtenidos en el ejemplo anterior, obtenemos los siguientes resultados:

Para x =0, $x > \frac{-\gamma+\sqrt{\gamma^2-3\mu}}{3}$=-0.42264 f'(x)=2>0, Inestable.

Para x=-1, $\frac{-\gamma-\sqrt{\gamma^2-3\mu}}{3} < x < \frac{-\gamma+\sqrt{\gamma^2-3\mu}}{3}$ → -1.5773<-1<-0.42264 → f'(x)=-1<0, Estable.

Para x=-2, $x < \frac{-\gamma-\sqrt{\gamma^2-3\mu}}{3}$ → -2<-1.5773 → f'(x)=2>0 Inestable

Vemos que se cumplen las condiciones anteriores.

Zona III

Condiciones:

$$\mu < 0$$

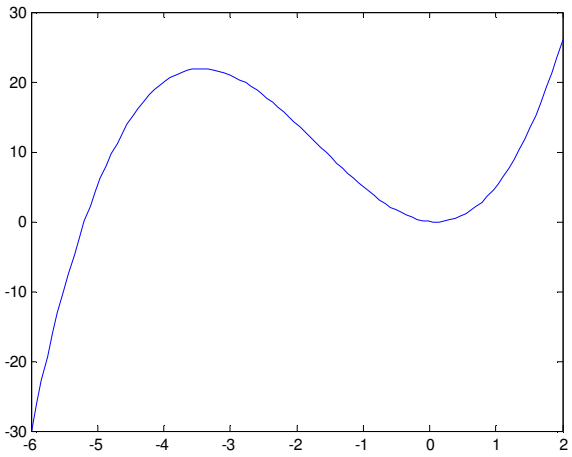
Observando, X= $\frac{\gamma \pm \sqrt{\gamma^2-4\mu}}{2}$, Vemos que la resolución de la ecuación de 2º grado tiene dos soluciones, por lo tanto en ésta zona, tendremos tres raíces.

Comprobamos dando valores a la ecuación, de modo que se cumplan las condiciones anteriores:

$$\gamma = 5, \mu = -1 \longrightarrow (Se\ entiende\ que\ \mu = u, \gamma = l)$$

Nos queda por tanto la ecuación: $x(x^2 + 5x - 1) = 0$

```
>> x=linspace(-6,2,100);
>> y=x.*(x.^2+5*x-1);
>> plot(x,y)
```



Por Newton-Raphson:

P0=-5.

p1 =

-5.19258240683994

P0=-1

p1 =

-4.196064587665967e-014

P0=1

p1 =

0.19258240357800

Estabilidades:

Para estudiar la estabilidad,lo haremos del mismo modo que antes.

$$f'(x) = 3x^2 + 2\gamma x + \mu.$$

Si vemos la solución de la ecuación anterior, obtendremos:

$x=\frac{-2\gamma\pm\sqrt{4\gamma^2-12\mu}}{6} = \frac{-\gamma\pm\sqrt{\gamma^2-3\mu}}{3}$.

Si las soluciones x de la ecuación cumplen:

$x > \frac{-\gamma+\sqrt{\gamma^2-3\mu}}{3}$ entonces f'(x)>0, inestable.

$\frac{-\gamma-\sqrt{\gamma^2-3\mu}}{3} < x < \frac{-\gamma+\sqrt{\gamma^2-3\mu}}{3}$ entonces f'(x)<0, estable

$x < \frac{-\gamma-\sqrt{\gamma^2-3\mu}}{3}$ entonces f'(x)>0, Inestable

Esto se puede verificar aplicando el ejemplo con números, estudiado anteriormente.

$$\gamma = 5, \mu = -1$$

$$f'(x) = 3x^2 + 10x - 1$$

$$x = \frac{-10 \pm \sqrt{112}}{6}, \quad x_1 = 0.09716, \; x_2 = -3.4305$$

Si verificamos con los resultados obtenidos en el ejemplo anterior, obtenemos los siguientes resultados:

Para x =0.19, $x > \frac{-\gamma+\sqrt{\gamma^2-3\mu}}{3}$=0.09716 f'(x)>0, Inestable.

Para x=-4.19e-4, $\frac{-\gamma-\sqrt{\gamma^2-3\mu}}{3} < x < \frac{-\gamma+\sqrt{\gamma^2-3\mu}}{3}$ → -3.4305<-4.19e-4<0.09716 → f'(x)<0, Estable.

Para x=-5.19, $x < \frac{-\gamma-\sqrt{\gamma^2-3\mu}}{3}$ → -5.19<-3.4305 → f'(x)>0 Inestable

Vemos que se cumplen las condiciones anteriores.

Zona IV

Condiciones:

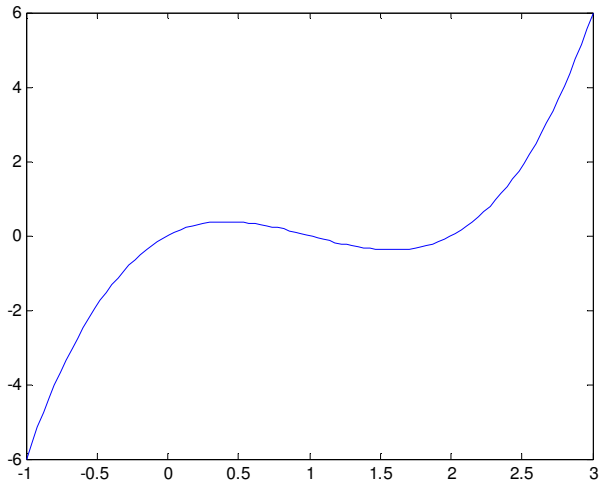
$$\gamma < 0, \qquad \mu > 0$$
$$\gamma + 2\sqrt{\mu} < 0$$

Comprobamos dando valores a la ecuación, de modo que se cumplan las condiciones anteriores:

$$\gamma = -3, \mu = 1 \longrightarrow (Se\ entiende\ que\ \mu = u, \gamma = l)$$

Nos queda por tanto la ecuación: $x(x^2 - 3x + 1) = 0$

```
>> x=linspace(-1,3,100);
>> y=x.*(x.^2-3*x+1);
>> plot(x,y)
```



Por Newton-Raphson:

P0=0

p1 =

0

P0=1

p1 =

0.38196601125184

P0=2

p1 =

2.61803398874990

Estabilidades:

Para estudiar la estabilidad,lo haremos del mismo modo que antes.

$$f'(x) = 3x^2 + 2\gamma x + \mu.$$

Si vemos la solución de la ecuación anterior, obtendremos:

$x=\frac{-2\gamma\pm\sqrt{4\gamma^2-12\mu}}{6} = \frac{-\gamma\pm\sqrt{\gamma^2-3\mu}}{3}$.

Si las soluciones x de la ecuación cumplen:

$x > \frac{-\gamma+\sqrt{\gamma^2-3\mu}}{3}$ entonces f'(x)>0, inestable.

$\frac{-\gamma-\sqrt{\gamma^2-3\mu}}{3} < x < \frac{-\gamma+\sqrt{\gamma^2-3\mu}}{3}$ entonces f'(x)<0, estable

$x < \frac{-\gamma-\sqrt{\gamma^2-3\mu}}{3}$ entonces f'(x)>0, Inestable

Observamos que ls condiciones son las mismas que en los casos anteriores, por lo que no vamos a demostrar el caso numérico en ésta ocasión.

Zona V

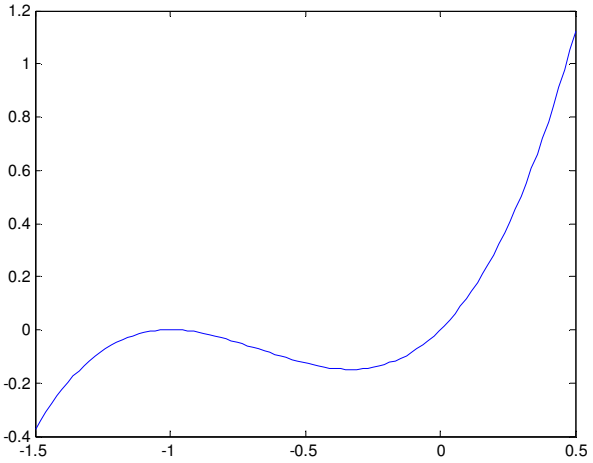
Aún queda una región que estudiar, que es aquella en la que se cumple que

$$\mu = \frac{\gamma^2}{4} \quad \text{, y} \quad \gamma > 0,$$

Serán los límites entre el área I - II, y I – IV

Comprobamos por ejemplo, para los valores $\gamma = 2$, por lo que $\mu = 1$.

```
>> x=linspace(-1.5,0.5,100);
>> y=x.*(x.^2+2*x+1);
>> plot(x,y)
```



Si obtenemos las raíces, veremos que tenemos x=0, y x=-1, por lo que se cumple la existencia de 2 raices en la zona denominada V.

En el caso de $\mu = 0$, (Eje abcisas de la representación de la bifurcación, y separación del sistema con la zona 3, también tendremos dos soluciones:

X=0 doble, x=-γ

En el caso de las estabilidades, se cumplen las mismas condiciones que en los casos anteriores.

2.2. ECUACIONES DIFERENCIALES DE PRIMER ORDEN. ESTUDIOS MEDIANTE DFIELD.

2.- Supongamos que originariamente hay P toneladas de peces en un lago y que su evolución viene dada por la ecuación logística con razón de reproducción r=1 y capacidad de soporte del medio k=10, es decir:

$$\frac{dP}{dt} = rP(1 - \frac{P}{k})$$

a) Calcular la población al cabo de tres meses si en el instante t=0, hay una tonelada de peces. Hallar el orden de cada método utilizado. ¿Qué ocurre con la población a largo plazo?, hallar los equilibrios y obtener el retrato de fases de dicha ecuación.

Si hacemos un estudio inicial de los puntos de equilibrio de la función, quedará del siguiente modo:

$$P' = P\left(1 - \frac{P}{10}\right) \quad , \quad Si\ P' = 0,$$

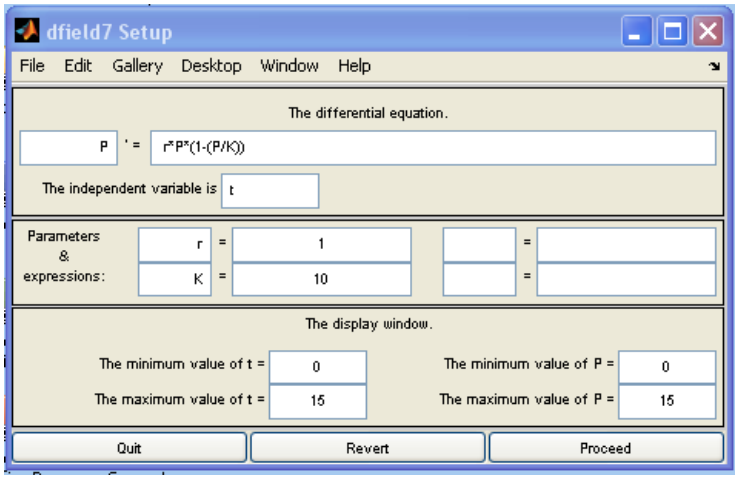
$$Obtendremos\ P = 0, y\ P = 10, serán\ puntos\ de\ equilibrio.$$

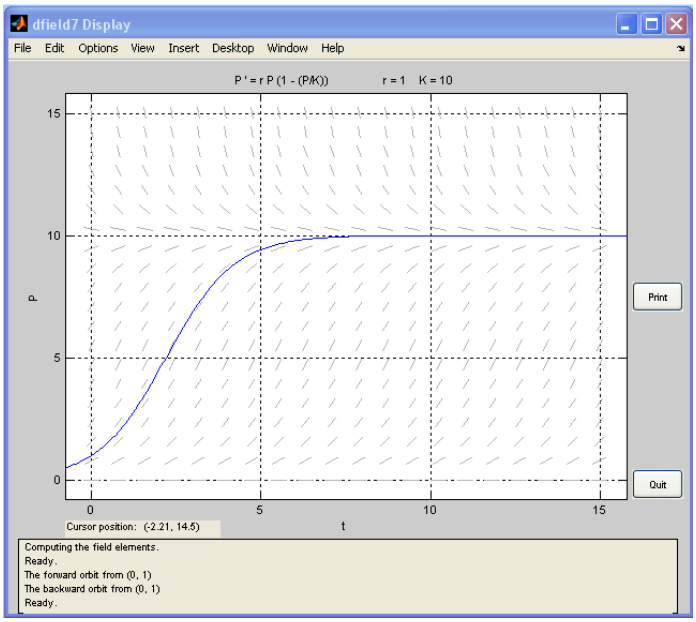
Para ver la estabilidad de los puntos, derivamos la función anterior:

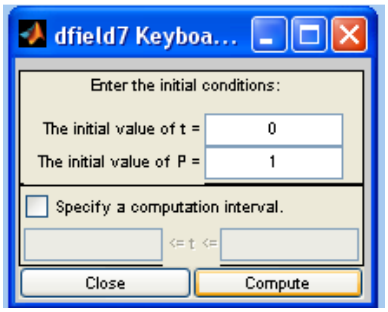
$$dP' = 10 - 2P, \qquad Para\ P = 0, dP' > 0, Inestable.$$

$$Para\ P = 10, dP' < 0, Estable.$$

Aplicando Dfield, observaremos los puntos de equilibrio con sus estabilidades, tal y como hamos visto de forma teórica..







Observamos que a largo plazo, con éstas condiciones iniciales la población tiende a estabilizarse aproximadamente en P=10 Tn.(Equilibrio estable).

b) Empezamos a pescar, a razón de 3 toneladas al mes los 3 primeros meses, y 0,5 toneladas los restantes meses.

La función captura viene dada por:

H(t)= 3 si t<3

0,5 si t>3

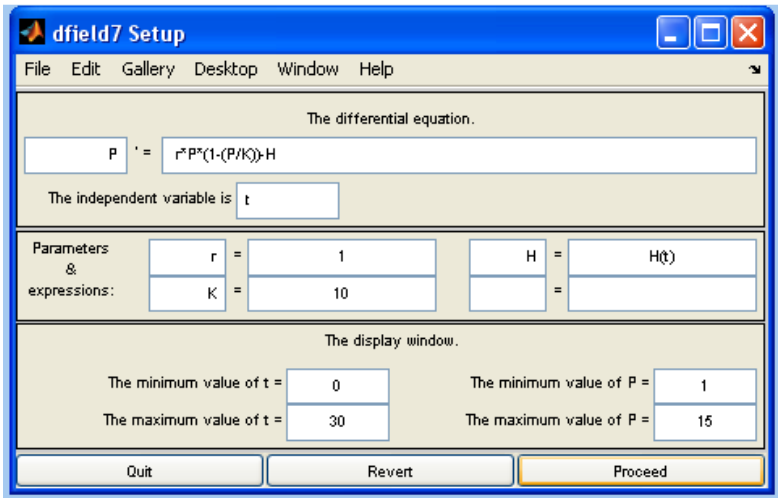
Dibujar la trayectoria correspondiente a la función captura, ¿Cuál es la población existente a los 18 meses?, calcular el valor mínimo de la población

Para introducir la función captura en Matlab, usaremos un archivo independiente, al cual llamaremos desde Dfield, lo haremos del siguiente modo:

Carmelo Camacho González
Fernando J. Manobel Ponce

H.m

```
function v=H(t)
v=3*(t<=3)+0.5*(t>3);
```



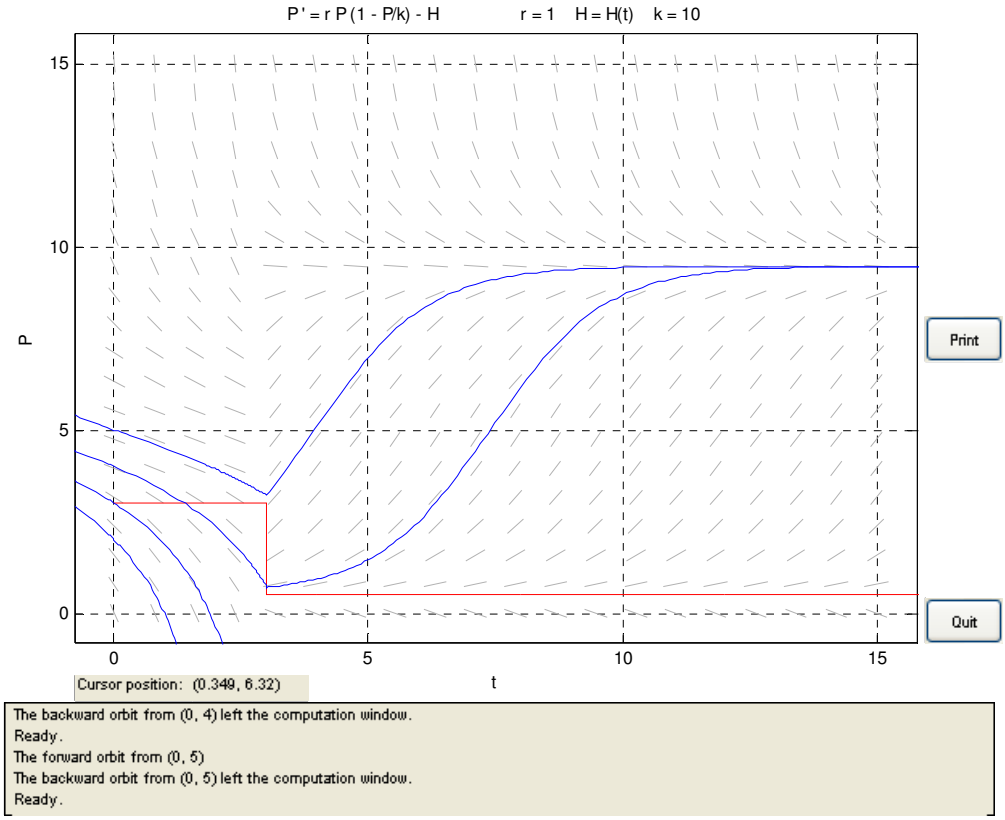
Observamos que si vamos metiendo valores iniciales de peces, y aplicando la función de pesca, hasta que no tengamos un valor inicial, por encima de un mínimo, la solución no tiende a equilibrarse, observamos que para P=2 y 3, la solución tiende a 0 (extinción), sin embargo, a partir de 3.9575 tiende a estabilizarse en entornos cercanos a la solución de equilibrio.

Para incluir la función pesca en la gráfica, lo haremos del siguiente modo:

```
>> t=0:0.001:20;
>> Y=H(t);
>> plot(t,Y,'r')
```

La solución trayectoria y la función captura la obtenemos en la siguiente gráfica:

Carmelo Camacho González
Fernando J. Manobel Ponce



Para obtener la población a los 18 meses, así como el máximo o mínimo lo haremos del siguiente modo:

Meteremos los datos manuales, entre los valores 0-18 meses, con una población inicial de 10 Ton.

Iremos a option-Export solution data. Y nos dará un archivo al que ha exportado todas éstas soluciones, será éste con el que tengamos que trabajar.

Para obtener el valor mínimo de la grafica, lo haremos del siguiente modo:

Exportaremos los datos, por lo que nos dará un archivo en Matlab, donde hemos obtenido los mismos:

Para obtener los resultados en Matlab:

(Solo mostramos alguno de los valores obtenidos.

```
A=[dfdata1.t;dfdata1.P]'
A =
         0   10.0000
    1.0443    7.9168
    2.1144    6.7989
    3.0002    6.1630
    4.0159    7.7600
    5.0095    8.6787
    6.0718    9.1477
```

7.1091 9.3410
8.0178 9.4135
9.0545 9.4488
10.1014 9.4630
11.1023 9.4684
12.0240 9.4705
13.0824 9.4715
14.3200 9.4719
15.0274 9.4720
16.1858 9.4721
17.0929 9.4721
18.0000 9.4721

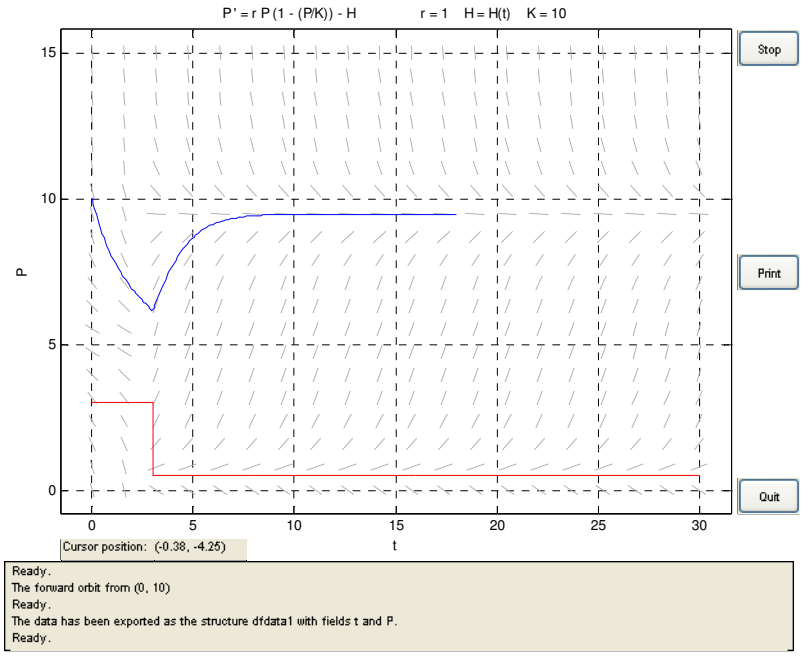
Podemos observar como partimos de las 10 toneladas, y como va variando la población inicialmente, cuando comenzamos la actividad de la pesca, posteriormente la misma se va normalizando estabilizándose en 9,4721 a los 18 meses.

Para obtener el valor mínimo de la serie.

min(dfdata1.P)

ans =

6.1627



c) Supongamos que la función captura H(t) es constante en todo instante. ¿Qué ocurre con los puntos de equilibrio al variar ésta constante en el intervalo

[0,3]?¿Para qué valor de la constante se produce un cambio en el nº de puntos de equilibrio?. Dibujar el diagrama de bifurcaciones correspondiente. ¿Cómo se llama éste tipo de bifurcación?.

Para estudiar la función, comenzaremos del siguiente modo:

$$P' = P\left(1 - \frac{P}{10}\right) - H, \qquad H = [0,3],$$

Para hallar los puntos de equilibrio:

$$P' = 10P - P^2 - 10H, \quad P = \frac{-10 \pm \sqrt{100 - 40H}}{-2}, \qquad P = 5 \pm \sqrt{25 - 10H}$$

Podemos deducir tres posibles soluciones:
Si H< 2,5, 25-10H >0, tendremos dos soluciones distintas y reales.
Si H=2,5, 25-10H = 0, tendremos una única solución real.
Si H>2,5, 25-10C<0, tendremos soluciones complejas.(No existen puntos de equilibrio).

Para ver la estabilidad, derivamos la función, obteniendo:

$$dP' = -2P + 10$$

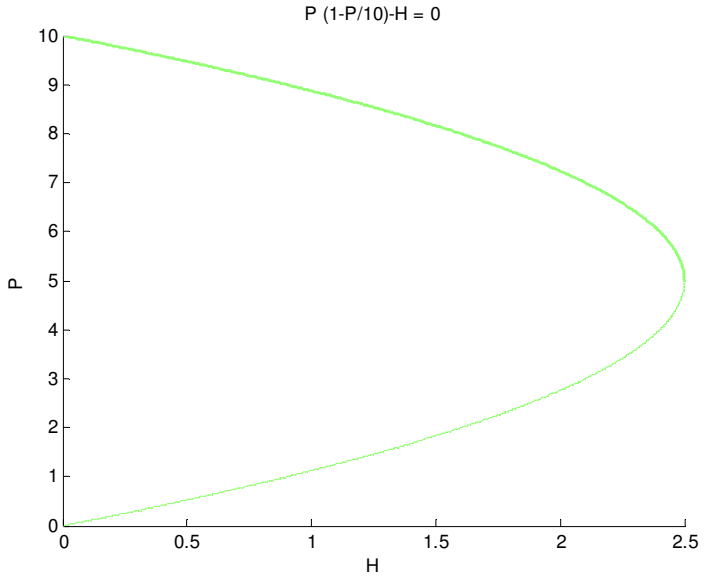
Si P>5, las soluciones serán estables.
Si P<5, las soluciones serán inestables.

Para representar el diagrama de bifurcaciones, emplearemos la función ezplot, del siguiente modo:

```
ezplot('P*(1-P/10)-H',[0,2.5],[0,5])
hold on
ezplot('P*(1-P/10)-H',[0,2.5],[5,10])
```

(La parte gruesa de la gráfica marca las soluciones estables).

Carmelo Camacho González
Fernando J. Manobel Ponce



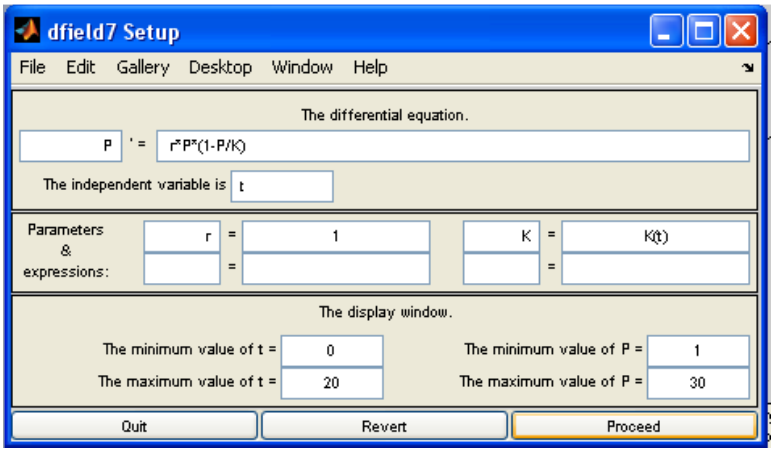
d) Si suponemos que la capacidad soporte del medio varía linealmente con el tiempo *k=10+t, y que en el instante t=0 hay una tonelada de peces, dibujar la trayectoria correspondiente así como la función soporte. ¿Cuál es la población al cabo de 5,10,15 y 18 meses?¿Como varía la población a largo plazo?.

Para resolver éste apartado, hemos supuesto que k ya no es constante, introduciendo un nuevo archivo .m

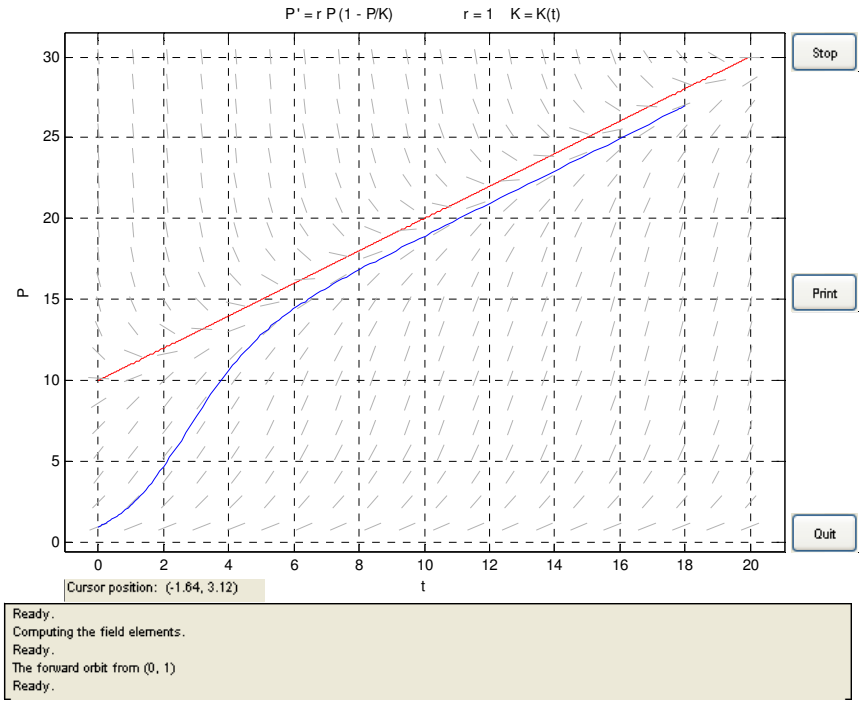
Evidentemente partimos de la premisa que no hay función captura, pues en ese caso, partiendo de 1 tonelada de peces, la especie desaparecería independiente del valor lineal de k.

```
function s=K(t);
s=10+t;
```

Lo introducimos en el dfield del siguiente modo.



Los resultados obtenidos los podemos observar en la siguiente gráfica.



En rojo tenemos la capacidad soporte del medio, y en azul como varía la población de los mismos.

Los resultados solicitados son los siguientes:

Al cabo de 5 meses la población es de:

0	1.0000
1.0213	2.3778
2.0465	4.8524
3.0904	8.0792
4.1436	11.0250
4.9558	12.7633
5.0000	**12.8454**

Al cabo de 10 meses, la población quedará:

5.0490	12.9352
6.0641	14.5359
7.0309	15.7605
8.1273	16.9839
9.0764	17.9801
9.8992	18.8232
10.0000	**18.9258**

Al cabo de 15 meses, la población continúa del siguiente modo:

10.1755 19.1040
11.0193 19.9572
12.2247 21.1701
13.0807 22.0295
14.0334 22.9850
14.9252 23.8790
15.0000 23.9540

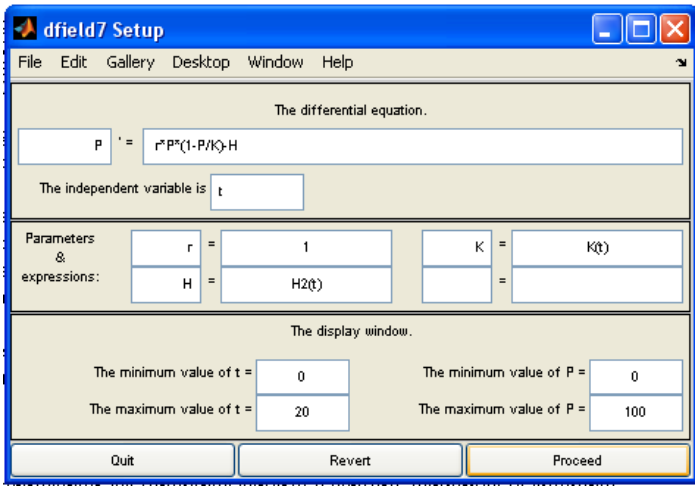
A los 18 meses, la población queda del siguiente modo:

15.0933 24.0475
16.2709 25.2275
17.5677 26.5267
18.0000 26.9597

Si seguimos representando a largo plazo, observamos que la población sigue aumentando de forma lineal.

e) Si suponemos que tanto la capacidad soporte como la función captura varían linealmente con el tiempo, k=10+t, H(t)=1+0.2t, dibujar las trayectorias correspondientes , así como las funciones soporte y captura, cuando P(0)=1.4 y P(0)=2. ¿Cómo varía la población a largo plazo?.

Comenzamos metiendo los archivos.m, tanto de soporte como captura, y modificamos la entrada de datos en dfield7, del siguiente modo:

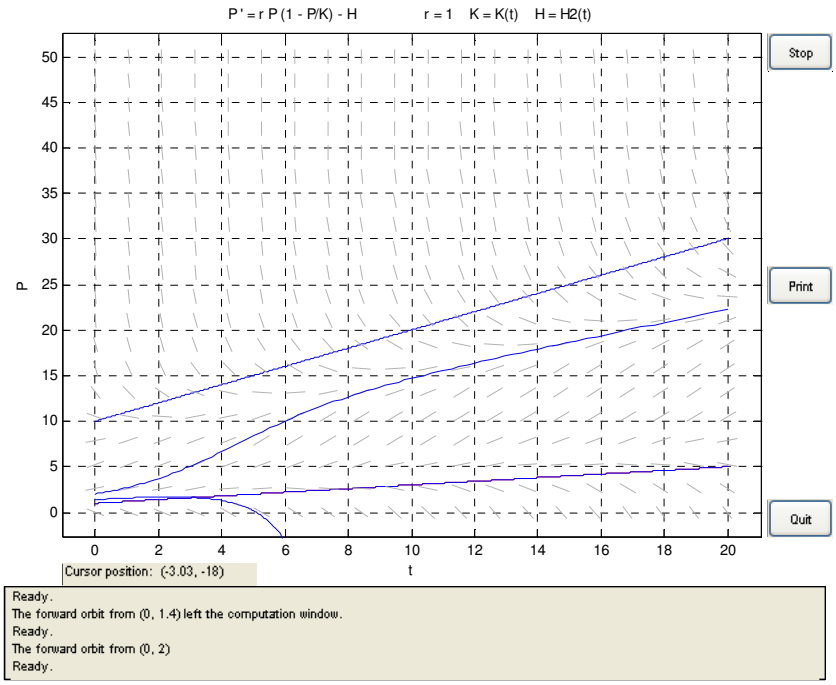


Para representar las funciones soporte y captura, copiamos lo siguiente:

```
>> t=0:0.001:20;
Y=H2(t);
```

```
plot(t,Y,'r')
>> Z=K(t);
>> plot(t,Z,'b')
```

Suponiendo p(0)=1.4, y posteriormente suponiendo p(0)=2 obtenemos la siguiente gráfica:



Obtenemos dos líneas de pendientes constante, que corresponde, la de abajo a la función captura, y la de arriba, a la función soporte, asimismo, vemos dos soluciones en la gráfica. La primera, con una P(0)=1,4, lo que nos muestra es que la población de peces tiende a desaparecer, con los dos parámetros anteriores, esta desaparición se produce aproximadamente a los 6 meses, sin embargo, podemos observar que en el caso de P(0)=2, la población de peces tiende a una solución estable, mostrando por tanto un crecimiento lineal.

3.- Un circuito RC está modelado por la ecuación:

$$RC\frac{dVc}{dt} + Vc = V(t)$$

Donde Vc es el voltaje a través del condensador. Supongamos que el tiempo es medido en segundos, la resistencia R=2,3 ohm, y la capacitancia C=1,2 F.

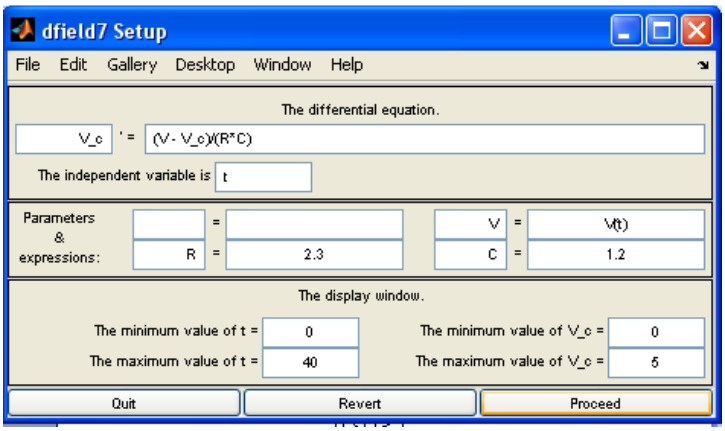
a) Si el voltaje suministrado por la fuente de tensión viene dado por:

V(t) $3\ si\ 0 \leq t < 5$

$$0 \; si \; t \geq 5$$

Dibujar en el intervalo de tiempo [0,40], la solución correspondiente a la ecuación inicial Vc(0)=0, y la función V(t). ¿Cuál es el valor de Vc en t=6 para la solución que cumple Vc(5)=4?

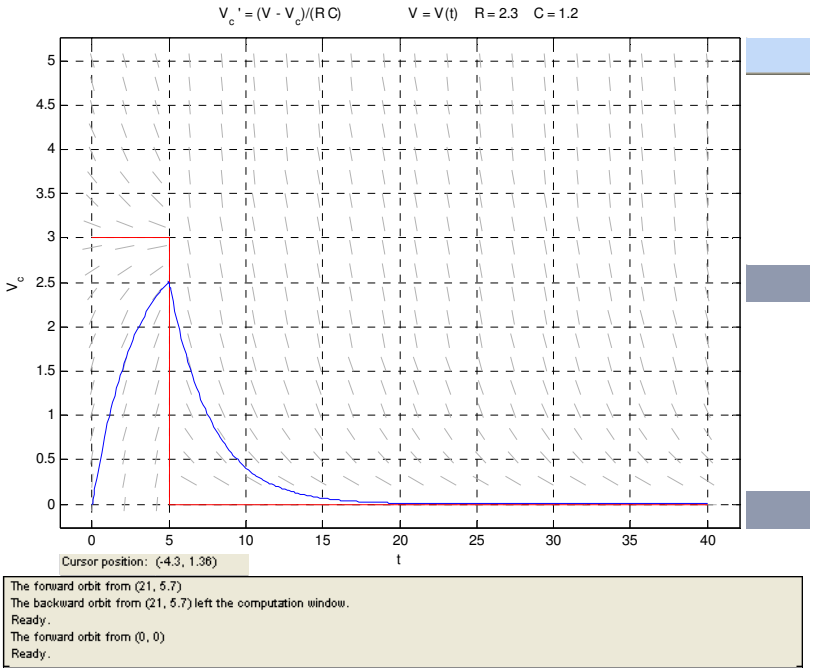
Introducimos los valores en dfield7, del siguiente modo.



Introducimos la función que define la carga inicial del siguiente modo:

```
function v=H(t)
v=3*(t<=3)+0.5*(t>3);
```

El resultado obtenido es el siguiente:

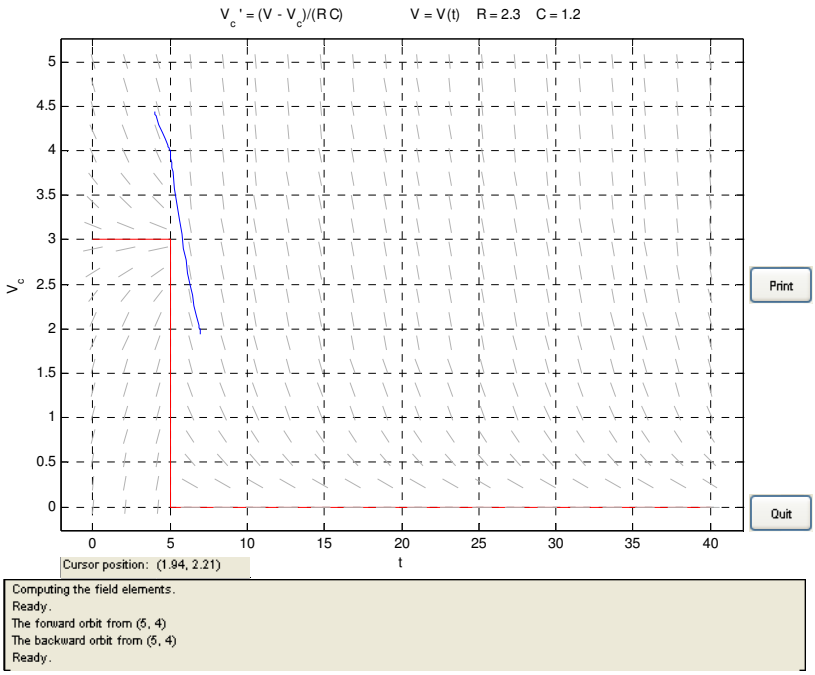


Observamos que los valores obtenidos en el condensador son los siguientes:

0	0
4.9997	2.5098
5.0002	2.5096
5.9607	1.7720
6.1002	1.6846
7.0781	1.1820
8.0562	0.8293
9.0344	0.5818
10.0129	0.4081
10.9918	0.2863
11.9711	0.2008
13.0912	0.1338
14.0723	0.0938
15.0549	0.0657
16.0396	0.0460
17.0272	0.0321
18.0187	0.0224
19.0164	0.0156
20.0222	0.0109
25.0040	0.0018
30.2528	0.0003
35.0451	0.0000
40.0000	0.0000

Observamos que al aplicar tensión al condensador los primeros 5 segúndos (Gráfica en rojo), el mismo comienza a cargarse (max 2.598 volt), cuando dejamos de aplicar tensión, éste se comienza a descargar hasta llegar 0 voltios (descarga completa).

El valor de Vc, en t=6, parala solución que cumple Vc(5)=4, es la siguiente:



Obtendremos los datos de la gráfica con la orden 'export data', y lo planteamos en Matlab, con la siguiente orden, obteniendo:

A=[dfdata1.t ; dfdata1.V_c]'

A =

5.0000	4.0000
5.0086	3.9875
5.0172	3.9751
5.0259	3.9627
5.0345	3.9503
5.1035	3.8528
5.1725	3.7577
5.2415	3.6649
5.3105	3.5744
5.4527	3.3949
5.5948	3.2245
5.7370	3.0626
5.8791	2.9089
5.9094	2.8772
5.9396	2.8459
5.9698	2.8149
6.0000	**2.7842**

b) A continuación supondremos que la onda de entrada vendrá dada por una onda cuadrada de amplitud 3 voltios, periodo 32 segundos y ciclo activo del 25%:

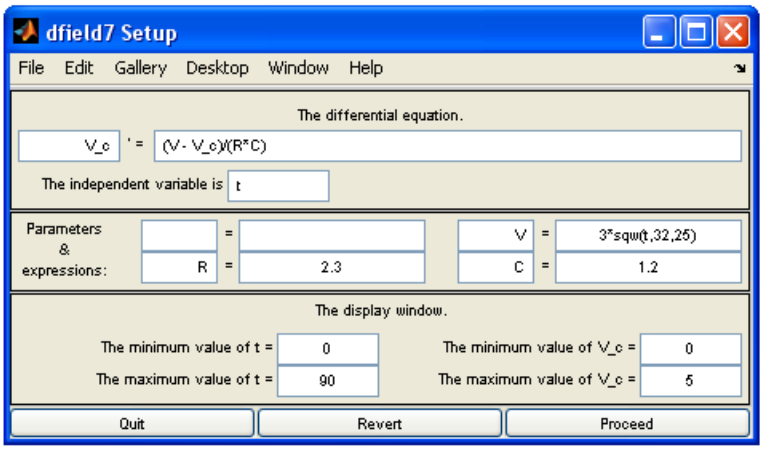
V(t)=3sqw(t,32,25).

Dibujar en el intervalo de tiempo [0,90], a solución correspondiente a la condición inicial Vc(0)=0, así como la función de entrada de voltaje V(t). Repetir lo mismo bajando el periodo a 16 y 4 segundos.¿Qué ocurre al disminuir el periodo?

Introduciremos la función cuadrada, en forma de archivo.m, quedando del siguiente modo:

```
function y=sqw(t,T,d)
% t es el tiempo
% T es el periodo
% d es el ciclo de funcionamiento
r=mod(t,T);
y=r<(d*T)/100;
```

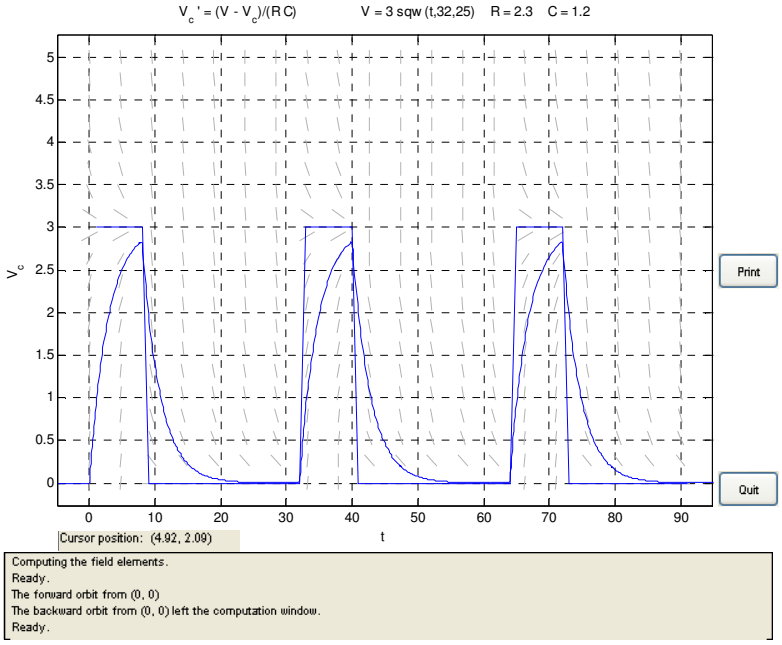
A continuación introduciremos los valores en dfield7, como sigue.



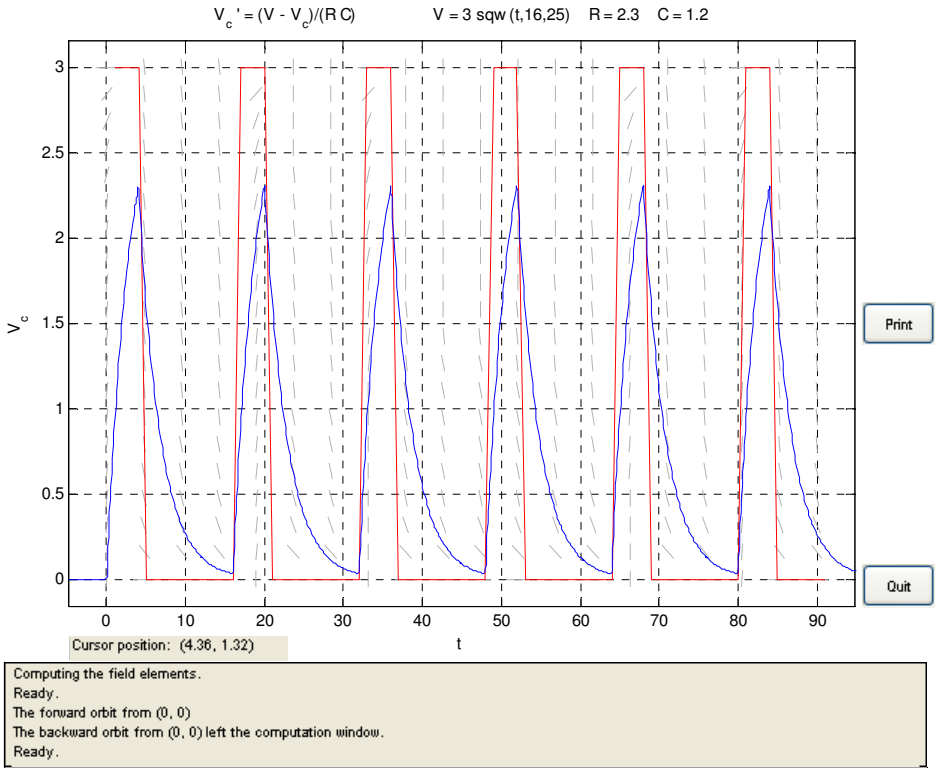
Representaremos las funciones de entrada y la función voltaje, quedando del siguiente modo:

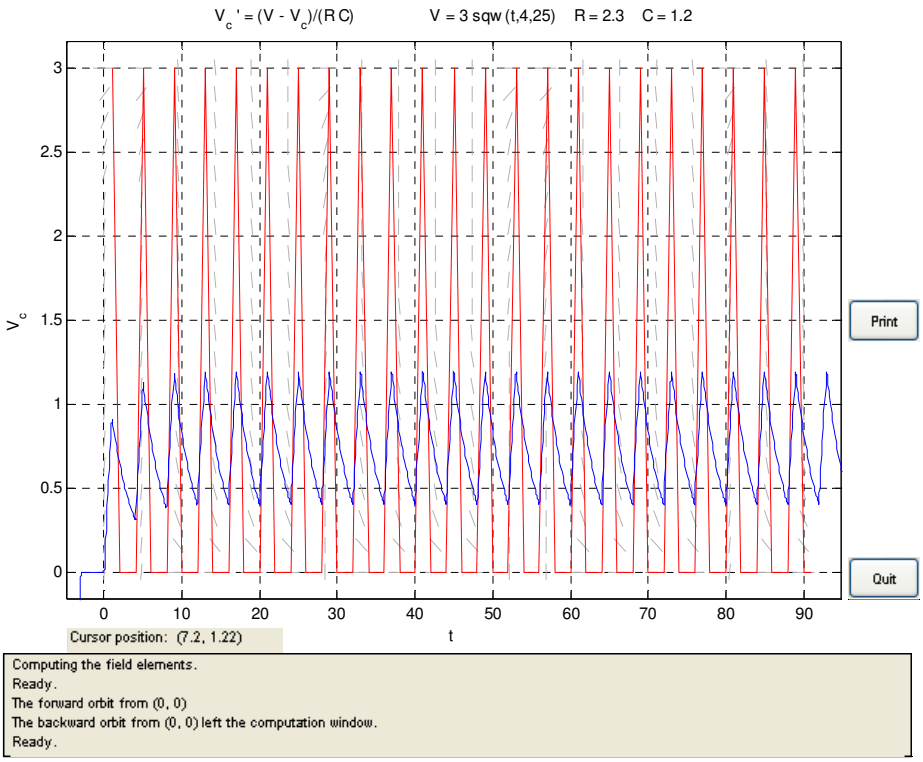
```
t=0:90;
>> plot(3*sqw(t,32,25),'b')
```

Carmelo Camacho González
Fernando J. Manobel Ponce



A continuación, representamos las funciones anteriores disminuyendo el periodo de carga inicial.





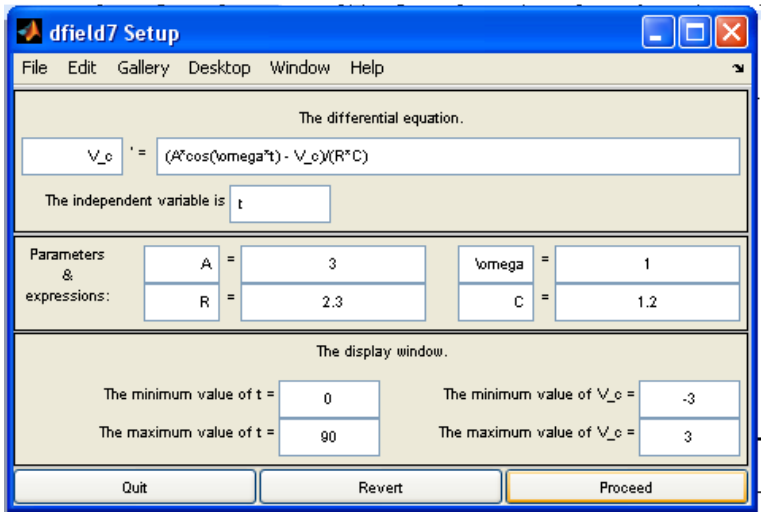
Observamos que a medida que disminuimos los periodos en los que aplicamos tensión a sistema, la tensión obtenida por el condensador disminuye, pues al mismo no le da tiempo a cargarse o descargarse.

c) Consideramos finalmente que la solución ahora viene dada por:

V(t)= Acos(wt), donde A=3, w=1,30,60, dibujar la solución correspondiente a la condición inicialVc(0)=0 y la función V(t). ¿Cómo varían la amplitud A y la frecuencia w de la señal de salida?.

Introduciremos la nueva función en el dfield7, del modo siguiente:

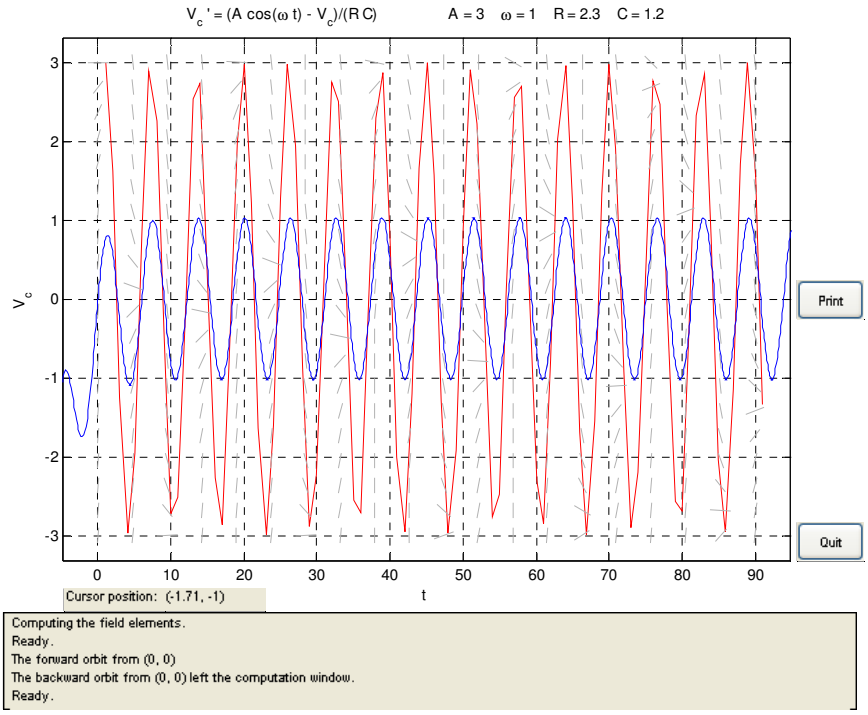
Carmelo Camacho González
Fernando J. Manobel Ponce



Representamos la función de entrada en color rojo, para una amplitud A=3, y w=1

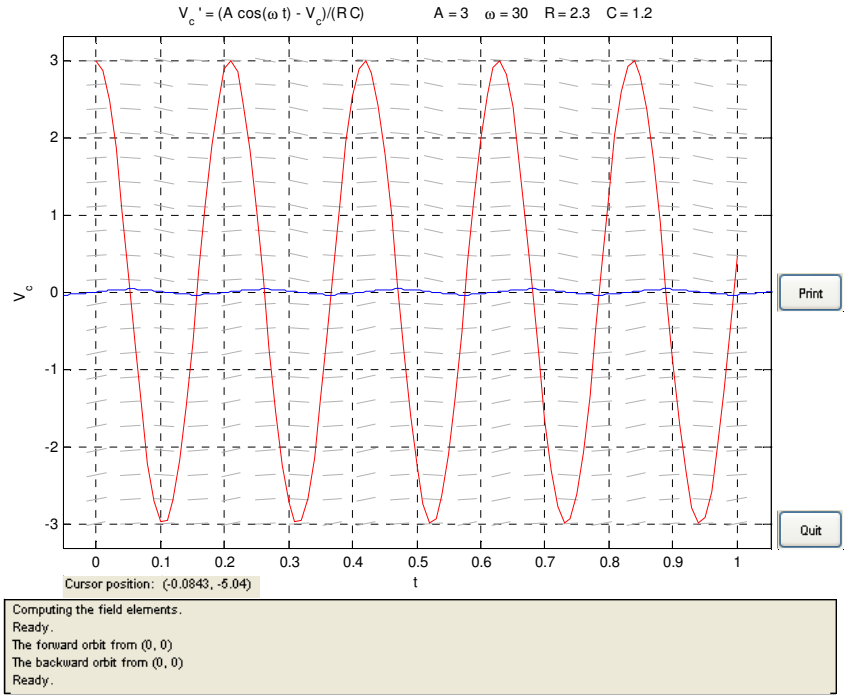
```
>> plot(3*cos(1*t),'r');
```

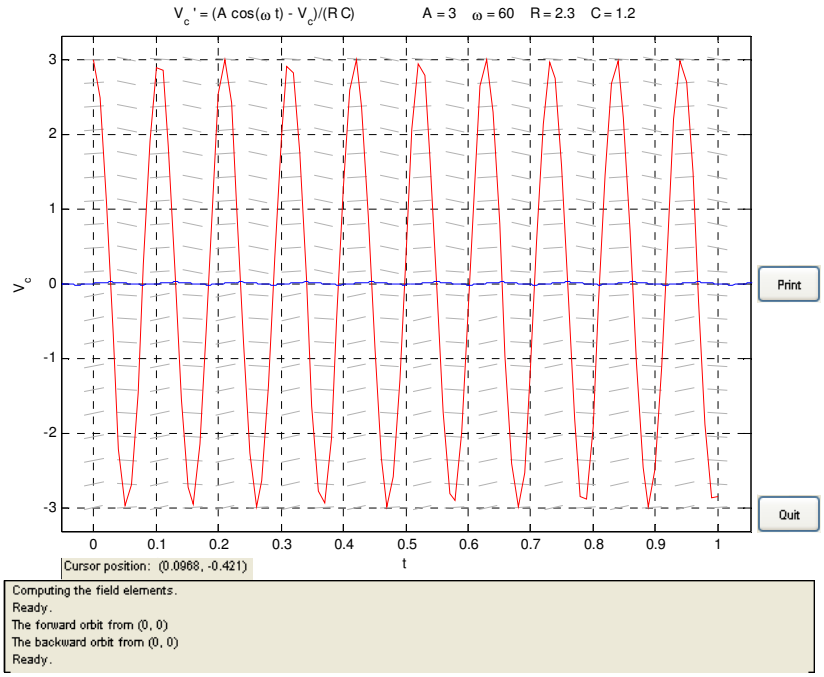
Obtenemos el siguiente resultado:



Si ahora varíamos las soluciones con w=30 y w=60, obtenemos las siguientes gráficas:

Carmelo Camacho González
Fernando J. Manobel Ponce





Podemos observar, que incrementando el periodo el condensador “deja” de cargarse y descargarse, al ser periodos tan cortos de tiempo.

4.- Un tanque con la forma de cilindro circular recto tiene una fuga de agua por un agujero circular en su fondo. Si no consideramos la fricción y la contracción del chorro en el agujero, la altura h del agua en el tanque viene dada por la elución:

$$\frac{dh}{dt} = -\frac{Ah}{Aw}\sqrt{2gh}$$

Ah y Aw son las areas transversales del agujero y del agua, respectivamente.

a) Si la altura inicial del agua es H, tomando g=9,8 m/s², obtener la solución correspondiente y su intervalo de definición en función de Ah,Aw y H.

Partimos de la premisa de que la ecuación anterior es una ecuación diferencial separable, por lo que bastará con despejar la parte dependiente de h, y la dependiente de t, e integrar por separado.
Obtendremos los siguientes resultados.

$$-\frac{Aw}{\left(Ah\sqrt{2gh}\right)}dh = dt$$

Sabiendo que los límites de h, están entre h(t) y H, procedemos a integrar ambos términos, obteniendo:

$$\int_H^{h(t)} -\frac{Aw}{\left(Ah\sqrt{2gh}\right)}dh = \int_{t0}^{t1} dt = -\frac{Aw}{Ah\sqrt{2g}}x2\left(\sqrt{h(t)} - \sqrt{H}\right) = t$$

Despejando h(t) para obtener la solución particular, tendremos:

$$h(t) = \sqrt{H} - \frac{Ah\sqrt{\frac{g}{2}}t}{Aw}$$

Para hallar el intervalo de la existencia de la función, partiremos desde el instante inicial t=0, al instante en el que la altura h(t)=0.
En la ecuación anterior por lo tanto igualamos h(t)=0, y despejamos t.

$$\sqrt{H} - \frac{Ah\sqrt{\frac{g}{2}}t}{Aw} = 0\,, \quad \frac{\sqrt{H}Aw\sqrt{2}}{Ah\sqrt{g}} = t,$$

$$t = Aw\sqrt{2H}/Ah\sqrt{g}$$

El intervalo quedara por tanto:

$$(0, Aw\sqrt{2H}/Ah\sqrt{g})$$

b) Si el tanque mide 3m de altura, 61 cm de radio, y el agujero tiene 1.27 cm de radio, ¿Cuanto tardará en vaciarse si el tanque está lleno al principio?

Si tenemos en cuenta la fricción y la contracción del agua en el agujero, al ecuación diferencial anterior se convierte en:

$$\frac{dh}{dt} = -c\frac{Ah}{Aw}\sqrt{2gh}$$

Donde 0<c<1, ¿Cuanto tardará en vaciarse si c=0.6?

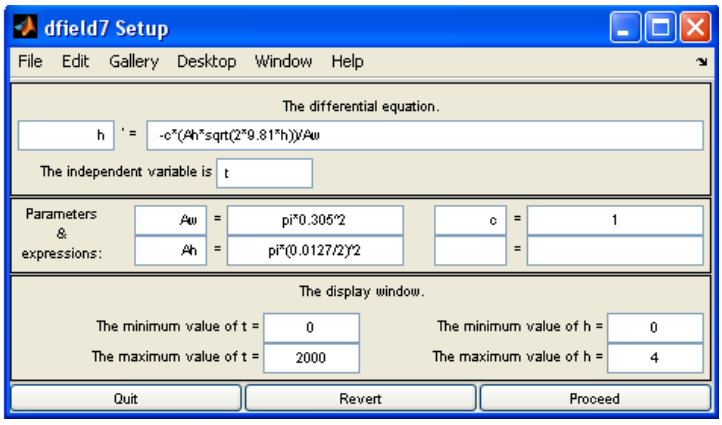
Para obtener la solución, es fácil, conociendo la ecuación general del sistema, pues la misma quedará:

$$h(t) = \sqrt{3} - \pi \times \frac{0.127^2\sqrt{\frac{g}{2}}}{\pi \times 0.61^2}t$$

Si suponemos la altura final h(t)=0, despejamos t, obteniendo 1805.2 segundos.

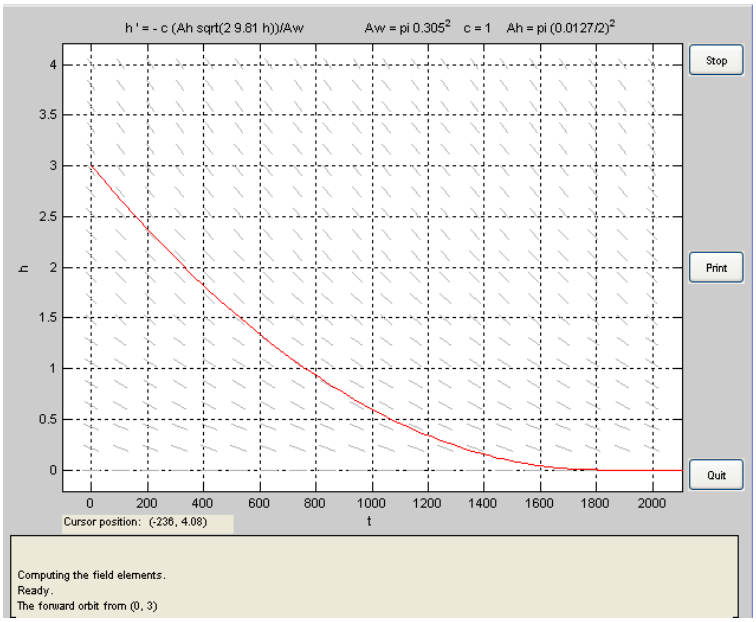
Si representamos directamente la variación de la ecuación en función del tiempo, obtenemos una función lineal.

Resolveremos la ecuación diferencial utilizando el dfield.

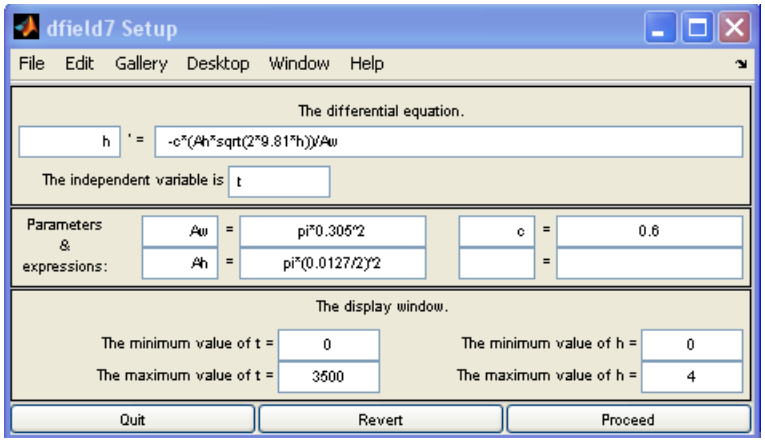


El resultado obtenido, gráficamente es la siguiente:

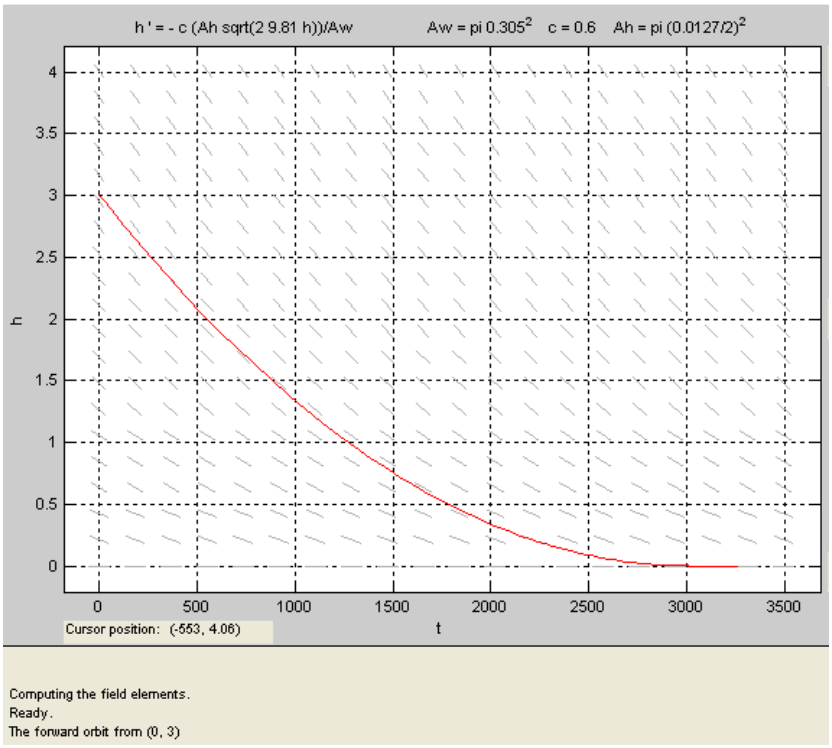
Carmelo Camacho González
Fernando J. Manobel Ponce



Si suponemos el caso para la existencia de fricción y contracción (c=0,6), la solucion quedará del siguiente modo:



Carmelo Camacho González
Fernando J. Manobel Ponce



El valor obtenido es de apr.3000 seg → 50 min

3.- SISTEMAS DE ECUACIONES DIFERENCIALES.

3.1. ANÁLISIS DE SISTEMAS DE ECUACIONES DIFERENCIALES. EMPLEO DE PPLANE Y ODESOLVE.

1.- Un circuito electrico RLC está modelado por la ecuación (oscilador armónico)

$$\boldsymbol{L\frac{d^2Q(t)}{dt^2}+R\frac{dQ(t)}{dt}+\frac{Q(t)}{C}=E(t).}$$

Donde R es la resistencia, L es la inductancia de la bobina, C la capacidad del condensador que almacena una carga Q(t) y E(t) una fuente de tensión que produce una intensidad I(t).

a) Probar que la ecuación anterior se puede escribir de la forma

$$\boldsymbol{Q'(t)=I(t)}$$

$$\boldsymbol{I'(t)=F(t)-2bI(t)-a^2Q(t)}$$

Donde a y b son constantes.

Partiendo de la ecuación inicial:

$$LQ''(t)+RQ'(t)+\frac{Q(t)}{C}=E(t)$$

Sabiendo que la I(t) es la derivada de la carga respecto al tiempo (Q'(t)), podemos colocar la ecuación anterior de la forma:

$$Q''(t)=\frac{E(t)}{L}-\left(\frac{R}{L}\right)I(t)-\frac{Q(t)}{LC}$$

A continuación podemos denominar cada término como consideremos oportuno, quedando:

$$\frac{E(t)}{L}=F(t)$$

$$\frac{R}{L}=2b,\qquad b=R/2L$$

$$\frac{1}{LC}=a^2,\quad a=1/\sqrt{LC}$$

Sustituyendo las denominaciones en la ecuación anterior, la misma nos quedará de la forma:

$$I'(t) = F(t) - 2bI(t) - a^2Q(t)$$

b) Suponemos que la resistencia y la tensión son nulas, E(t)=R=0 (oscilador armónico no amortiguado). Hallar la solución general y el periodo de las soluciones. Para a=1,2, dibujar las trayectorias y la órbita correspondiente a las condiciones iniciales (Q(0)=1, I(0)=0) y (Q(0)=0, I(0)=1).. ¿Cómo varían la amplitud y frecuencia de las soluciones?¿Que tipo de configuración presenta la solución de equilibrio?.

Para hallar las soluciones, lo podemos hacer como una ec. Diferencial de 2º órden con coeficientes constantes:

$$Q''(t) + \left(\frac{R}{L}\right)Q'(t) + \frac{Q(t)}{LC} = E(t).$$

En éste apartado, nos están diciendo que no tenemos ni aporte de energía externa, ni resistencia en el circuito, por lo que la ecuación anterior quedará de la forma:

$$Q''(t) + \frac{Q(t)}{LC} = 0$$

(Antes hemos llamado a 1/LC=a², seguiremos con esa notación para resolver el sistema).

Para la realización de éstos problemas, hallaremos la ecuación característica (de los autovalores), y de ese modo obtendremos la solución general tipo.

Podemos tener varias posibilidades:

1. Autovalores m_1 y m_2, reales y distintos, la ecuación quedará por tanto de la forma

$$y(t) = C_1e^{m_1t} + C_2e^{m_2t}$$

2.- Autovalores $m_1 = m_2$, reales, la ecuación quedará de la forma

$$y(t) = C_1e^{m\,t} + C_2e^{m\,t}$$

3.- Autovalores complejos, del tipo c+-di, la solución quedará del modo

$$y(t) = e^{c\,t}(C_1\,\text{c?s}(dt) + C_2\sin(dt))$$

En el caso que estamos estudiando, obtendremos dos autovalores complejos, quedando la solución por tanto según lo indicado en el apartado 3.

Sol: $x^2 = -a^2,\ m = \pm ia,\ \ c + di = 0 + ai$

La solución general quedará de la forma:

$$Q(t) = (C_1 \cos(at) + C_2 sen(at))$$

Para hallar los valores de C1 y C2 que dependen de las condiciones iniciales, obtendremos otra ecuación derivando la primera, por lo que obtendremos:

$$I(t) = Q'(t) = -C_1 a \text{ sen(at)} + C_2\, a \cos(at)$$

Sustituyendo en ambas ecuaciones para t=0

$$Q(0) = C_1$$

$$I(0) = C_2\, a, \quad \rightarrow \quad C_2\sqrt{1/LC}, \quad \dashrightarrow \quad C_2 = \frac{I(0)}{a}$$

La solución quedará de la forma:

$$Q(t) = Q(0)\cos(a\, t) + \frac{I(0)}{a} sen(at)$$

Representando la trayectoria de la solución anterior, nos debe quedar periódica, y por lo tanto, la órbita debe ser cerrada.

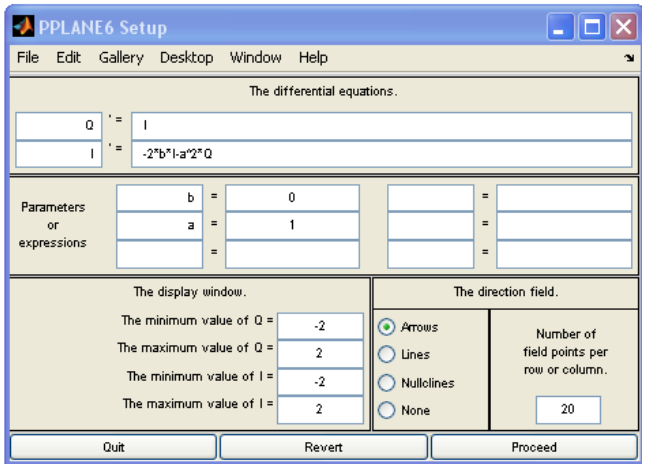
Para realizar los ejercicios con Matlab, emplearemos la aplicación PPLANE, para comenzar a trabajar con el mismo tendremos que introducir el problema como un sistema de ecuaciones de primer órden, por lo que emplearemos las ecuaciones del apartado A.

$$Q'(t) = I(t)$$
$$I'(t) = -2bI(t) - a^2 Q(t)$$

Tendremos que introducir los valores de

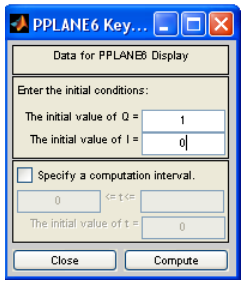
$$a = 1/\sqrt{LC} \qquad b = R/2L$$

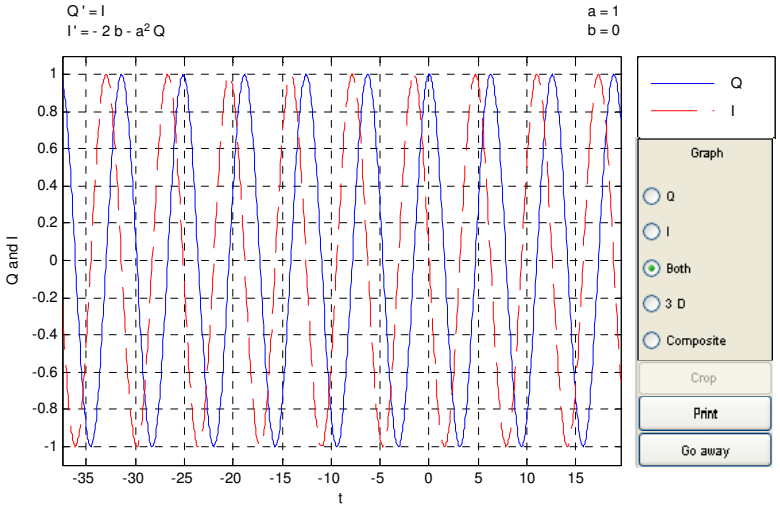
Para el caso de éste apartado, tenemos a=1 y b=0



Carmelo Camacho González
Fernando J. Manobel Ponce

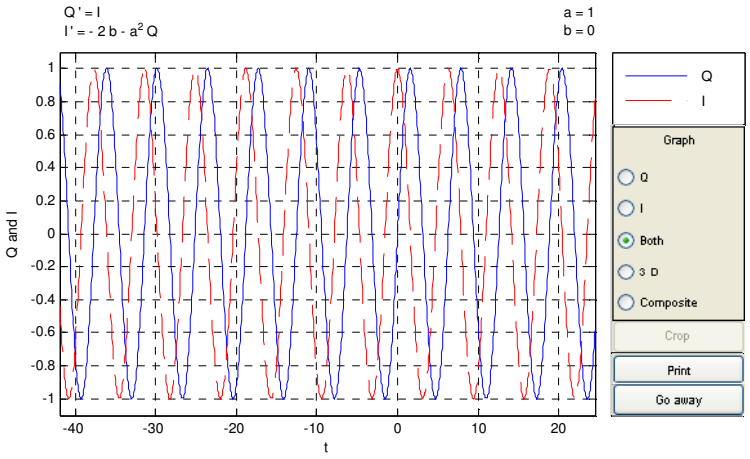
Para los datos iniciales Q(0)=1 e I(0)=0, las trayectorias quedan del siguiente modo.





Para Q(0)=0 e I(0)=1.

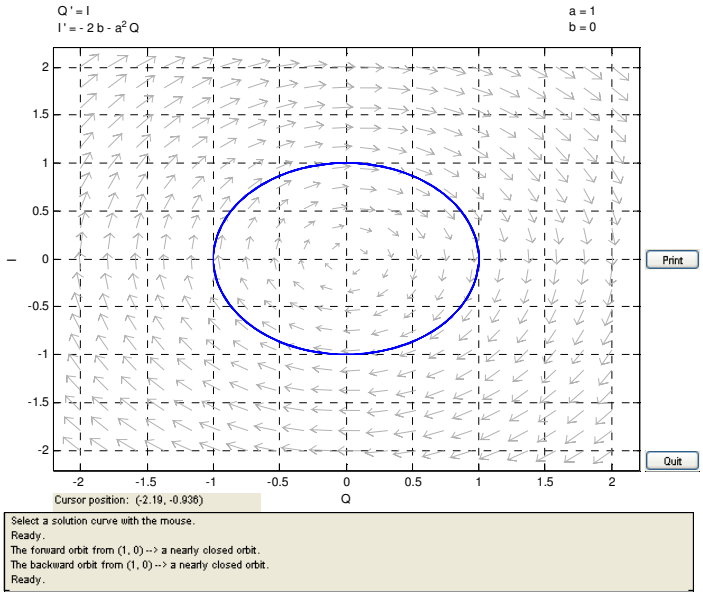
La trayectoria queda:



Las diferencias en las dos gráficas anteriores se encuentran únicamente en cierto desplazamiento de angulos entre ellas.

La órbita para ambos casos se superponen.

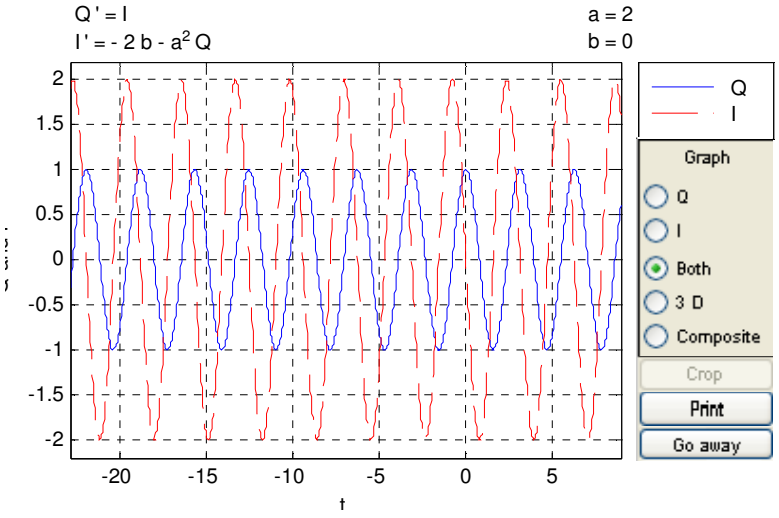
Carmelo Camacho González
Fernando J. Manobel Ponce



Para el caso de a=2, las gráficas quedan del modo siguiente:

Q(0)=1

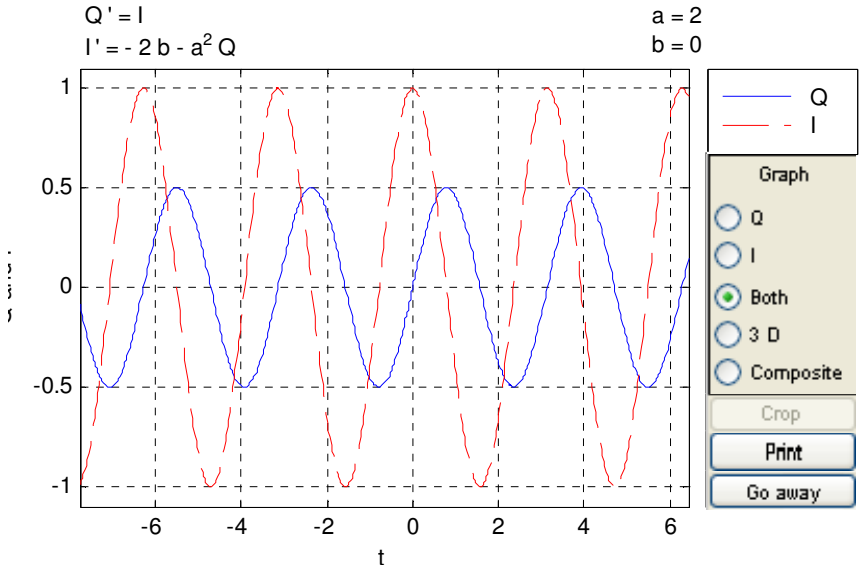
I(0)=0



Q(0)=0

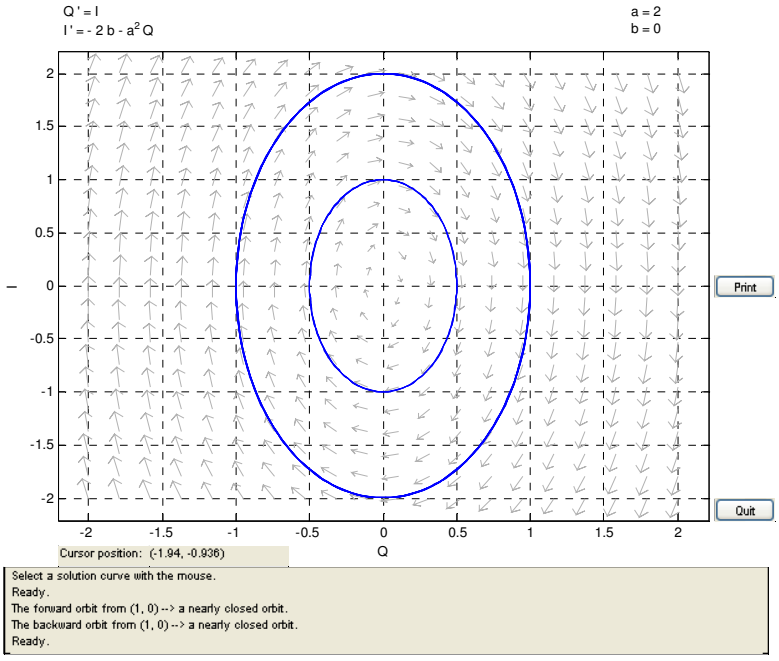
I(0)=1

Carmelo Camacho González
Fernando J. Manobel Ponce



En las trayectorias antes representadas, se observan las variaciones de amplitud y periodo, tema que hemos demostrado en la parte anterior.

La representación de la órbita nos queda del modo siguiente.(el caso de la órbita más pequeña es el equivalente a la segunda trayectoria representada.)



c) Supongamos que ahora la E(t)=0 (oscilador armónico amortiguado). Hallar la solución general cuando b>a (sobreamortiguado), b=a (críticamente amortiguado) y b<a (subamortiguado). ¿Qué ocurre en éste último caso cuando R→0?..

Dibujar trayectorias y órbitas para a=1 b=2; a=b=1, a=2 b=1, con unas condiciones iniciales (Q(0)=1, I(0)=0 y Q(0)=0 y I(0)=1). ¿Qué tipo de configuración presenta la solución de equilibrio en cada caso?.

Para comenzar éste apartado, volveremos a hallar los autovalores del sistema completo, diagonalizando. |A-m*I|=0, sabiendo que el sistema está compuesto por las dos ecuaciones siguientes:

$$Q'(t) = I(t)$$
$$I'(t) = -2bI(t) - a^2Q(t)$$

$$A = \begin{bmatrix} 0 & 0 \\ -a^2 & -2b \end{bmatrix}$$

$$[A - mI] = 0, \rightarrow \begin{bmatrix} 0-m & 0 \\ -a^2 & -2b-m \end{bmatrix} = 0 \rightarrow m^2 + 2bm + a^2$$

Las soluciones quedarán de la forma:

$$m = -b \pm \sqrt{(b^2 - a^2)}$$

Sobre éste punto aplicaremos las diferentes opciones que muestra el apartado del problema, obteniendo los autovalores de uno u otro modo, y por tanto la solución general de la ecuación será de las formas indicadas en el apartado anterior.

Asimismo, para saber el tipo de configuración de la solución de equilibrio en cada caso, tendremos que tener en cuenta los siguiente casos:

1.- m_1>0, m_2>0. Nodo Inestable.
2.- m_1<0, m_2<0. Nodo asintóticamente estable.
3.- $m_1 = m_2$. → si >0 Nodo asintóticamente inestable.
Si <0 Nodo asintóticamente estable.
4.- m_1>0, m_2<0 → Silla.
5.- m=a+-bi → a>0 espiral inestable,
a<0 espiral estable.
6.- m=+-bi → Centro

Nota. El punto 6 ha sido el caso del apartado c.

CASO b>a, observamos que obtendremos 2 autovalores positivos distintos, siendo la solución de la forma siguiente:

$$y(t) = C_1 e^{m_1 t} + C_2 e^{m_2 t}$$

Observando la ecuación solución, $b^2-a^2 > 0$

También podemos afirmar que

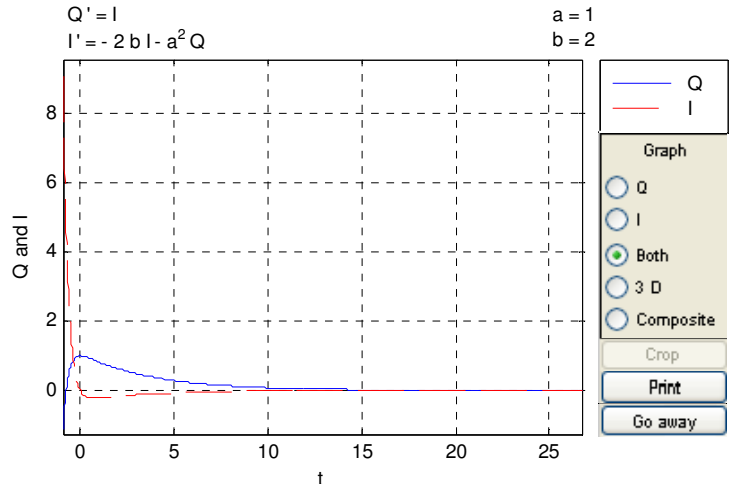
$$b > \sqrt{(b^2 - a^2)}$$

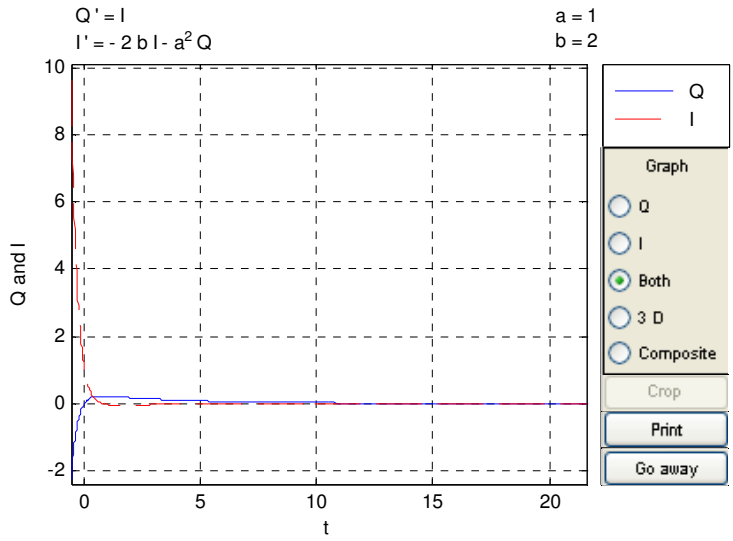
por tanto ambos autovalores obtenidos serán <0, estaremos en un caso 2. (m_1<0, m_2<0. Nodo asintóticamente estable.),

La caída de energía en el sistema nos llevará a un punto de estabilidad.

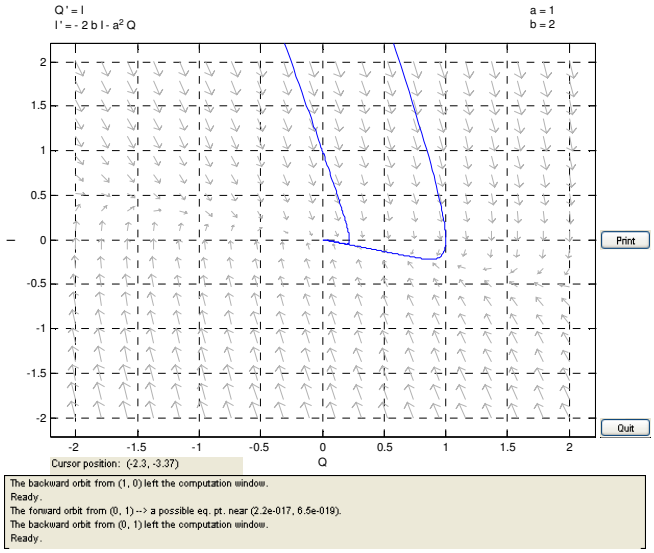
Lo comprobamos empleando el PPlane:

Lo representamos para los valores definidos en la figura, observamos ambas soluciones, para los valores iniciales Q=0, I=1, y Q=1,I=0, vemos que la respuesta tiende hasta el punto de equilibrio, tal y como hemos indicado anteriormente.





Carmelo Camacho González
Fernando J. Manobel Ponce



CASO b=a (oscilador críticamente amortiguado),

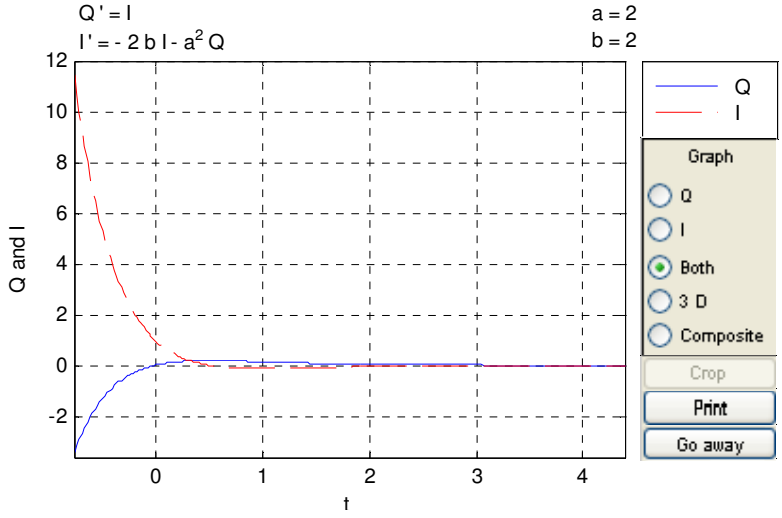
En éste caso, obtenemos que m1=m2, y es <0, por lo tanto estamos en el caso de un nodo asintóticamente estable.

La ecuación quedará por tanto de la forma:

$$y(t) = C_1 e^{m\,t} + C_2 e^{m\,t}$$

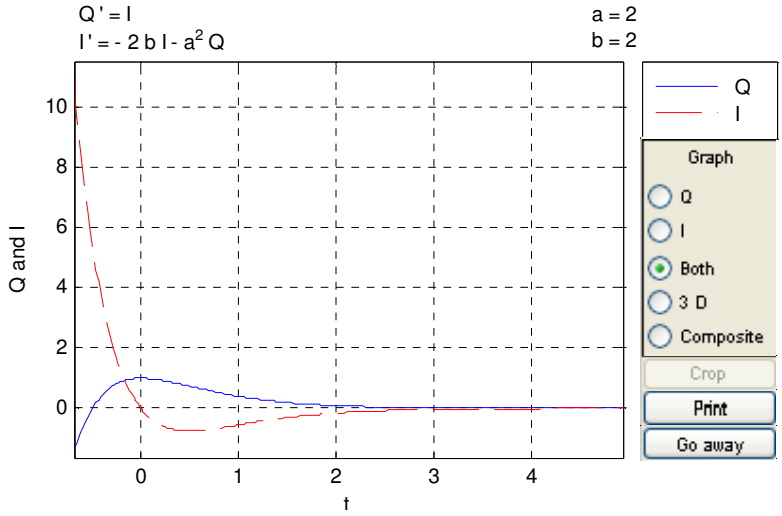
Comprobaremos lo anterior empleando el algoritmo de Matlab.

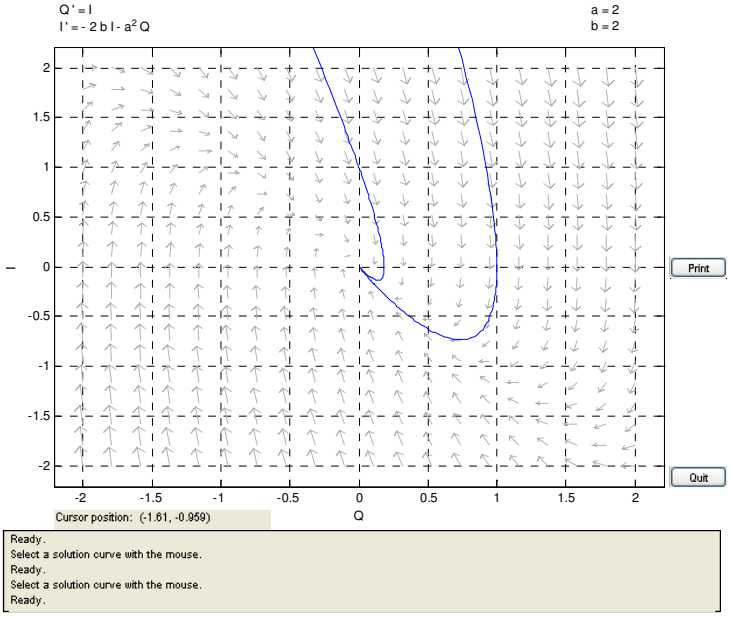
Para Condiciones iniciales Q=0, e I=1.



Para Condiciones iniciales I=0, y Q=1.

Carmelo Camacho González
Fernando J. Manobel Ponce





Observamos que al igual que en el caso anterior, la solución tiene hacia el un punto estable, pero de un modo más brusco que en el caso anterior.

CASO b<a, (Oscilador subamortiguado).

Si vamos a la ecuación de los autovalores, tal y como la estamos analizando, observamos:

$m = -b \pm \sqrt{(b^2 - a^2)}$, en éste caso el interior de la raíz será negativo, y por tanto tendremos dos soluciones complejas

De los diferentes casos expuestos, corresponderá al 5, al ser b<0, tendremos por tanto una espiral estable.

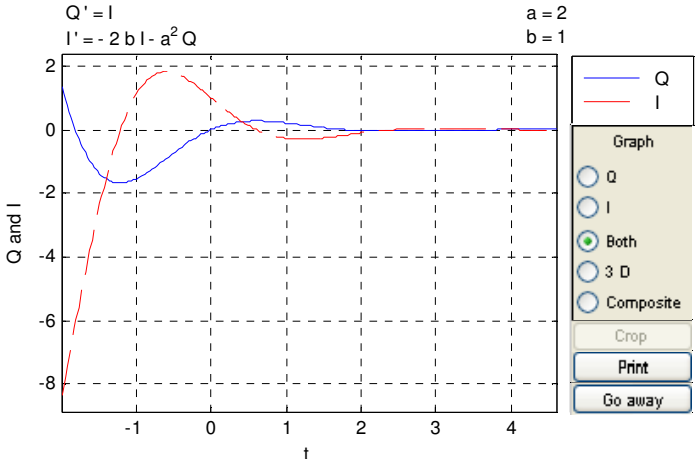
La solución general de la ecuación será de la forma:

$$y(t) = e^{c\,t}(C_1 \cos(dt) + C_2 \operatorname{sen}(dt))$$

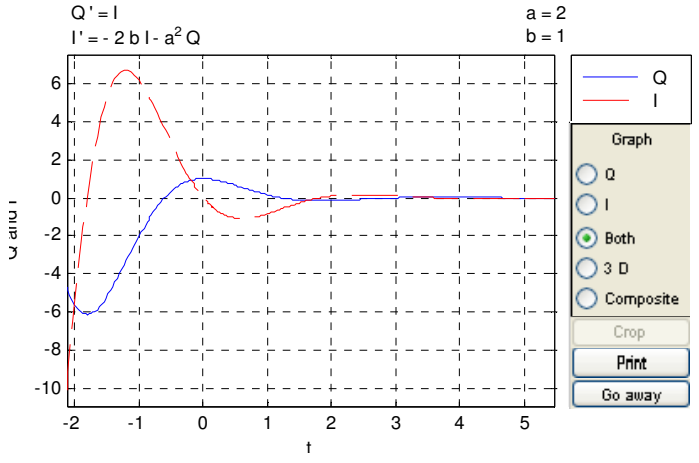
Lo confirmamos representándolo con a ayuda del Matlab.

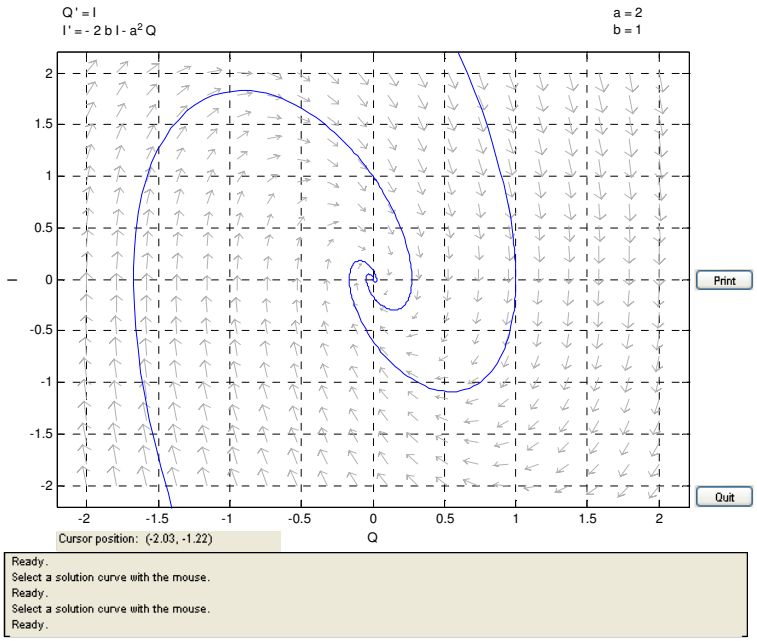
(En las ecuaciones representadas b=c).

Para C.Inicial Q=0 e I=1,



Para la C. Inicial Q=1 e I=0, nos queda:





d) En el caso del oscilador subamortiguado, integrar numéricamente el sistema empleando los comandos de Matlab, en el intervalo $\left[0, \frac{6\pi}{\sqrt{3}}\right]$, para Q(0)=1 e I(0)=0. ¿En que valores la carga Q(t) alcanza los valores máximos?.

En éste caso empleamos un algoritmo externo para efectuar la integración, y otro archivo para definir la función:

ode45sistema.m

```
function S=ode45sistema(ftysis,a,b,Za,M)
%
% Datos de entrada
% ftysis es la funcion, almacenada como una cadena de caracteres
% a y b son los extremos del intervalo
% Za=[x1(a),...,xn(a)] es la condici¶on inicial
% M es el n¶umero de pasos
% Datos de salida
% S=[T X] donde T es el vector con las abscisas y
% X=[x1(t),...,xn(t)] es el vector con las ordenadas
h=(b-a)/M;
T=a:h:b;
[T X]=ode45(ftysis,T,Za);
format long
S=[T X];
```

ftysis.m

```
function Z=ftysis(t,E)
Q=E(1); I=E(2);
```

```
Z=[I, -2*I-4*Q]';
```

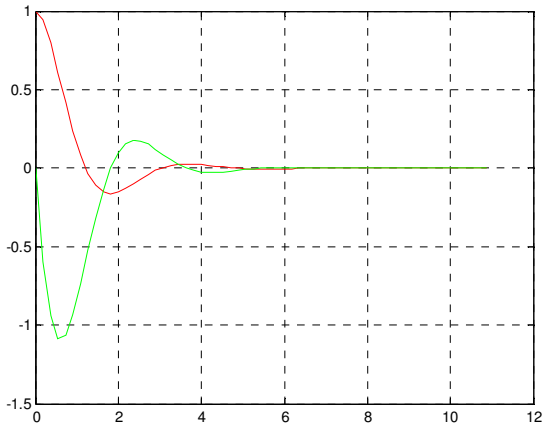
La solución obtenida será del tipo matricial, obteniendo en la primera columna, el intervalo que le hemos pedido (en nuestro caso 60 pasos), en la columna siguiente el valor de Q(t), y en el siguiente el valor de I(t).

```
>> S=ode45sistema('ftysis',0,6*pi/sqrt(3),[1,0],60)

S =

            0   1.00000000000000                   0
  0.18137993642342   0.94210973773204  -0.59526342941474
  0.36275987284684   0.79890546623564  -0.94429110422317
  0.54413980927027   0.61220899644033  -1.08427290889920
  0.72551974569369   0.41535138238746  -1.06300374930233
  0.90689968211711   0.23317948055738  -0.93254745687254
  1.08827961854053   0.08090560871785  -0.73963684378352
  1.26965955496395  -0.03384739830558  -0.52501878126227
  1.45103949138737  -0.10994441376309  -0.31812613931104
  1.63241942781080  -0.15097205427942  -0.13962657133036
  1.81379936423422  -0.16301175194574  -0.00018182643448
  1.99517930065764  -0.15360906352736   0.09692723213625
  2.17655923708106  -0.13029736114851   0.15389319650896
  2.35793917350448  -0.09985591385938   0.17675090378146
  2.53931910992790  -0.06777485253634   0.17334325844551
  2.72069904635133  -0.03805554977012   0.15206653919118
  2.90207898277475  -0.01322877020684   0.12065866515073
  3.08345891919817   0.00549384222772   0.08563068618925
  3.26483885562159   0.01791818811953   0.05193603463270
  3.44621879204501   0.02460480563559   0.02280222804289
  3.62759872846844   0.02658095633112   0.00005375088281
  3.80897866489186   0.02504560833202  -0.01578339194982
  3.99035860131528   0.02125182846009  -0.02508555290084
  4.17173853773870   0.01628614193674  -0.02881215197560
  4.35311847416212   0.01105805670011  -0.02826879109440
  4.53449841058554   0.00621033227888  -0.02479626824281
  4.71587834700897   0.00216157249420  -0.01968258689335
  4.89725828343239  -0.00089168906453  -0.01396768538827
  5.07863821985581  -0.00291951967447  -0.00847495713884
  5.26001815627923  -0.00401085679973  -0.00372449609082
  5.44139809270265  -0.00433374827740  -0.00001278009534
  5.62277802912608  -0.00408404170534   0.00257048138654
  5.80415796554950  -0.00346577498411   0.00408854922718
  5.98553790197292  -0.00265646599642   0.00469719926833
  6.16691783839634  -0.00180395450953   0.00460933710406
  6.34829777481976  -0.00101357352570   0.00404384974195
  6.52967771124318  -0.00035315019262   0.00321006773746
  6.71105764766661   0.00014480446623   0.00227875191721
  6.89243758409003   0.00047561919759   0.00138247431752
  7.07381752051345   0.00065380638579   0.00060825778453
  7.25519745693687   0.00070636679199   0.00000280934965
  7.43657739336029   0.00066609735414  -0.00041866414407
  7.61795732978372   0.00056507622064  -0.00066609216243
  7.79933726620714   0.00043341785722  -0.00076590817769
  7.98071720263056   0.00029429135732  -0.00075139931831
  8.16209713905398   0.00016548916574  -0.00065963713018
  8.34347707547740   0.00005775685397  -0.00052350187346
  8.52485701190082  -0.00002350070851  -0.00037184963265
  8.70623694832425  -0.00007742565115  -0.00022567391899
  8.88761688474767  -0.00010649590893  -0.00009928769252
  9.06899682117109  -0.00011518772170  -0.00000070569198
  9.25037675759451  -0.00010844489714   0.00006795994136
  9.43175669401793  -0.00009201629227   0.00010836667391
```

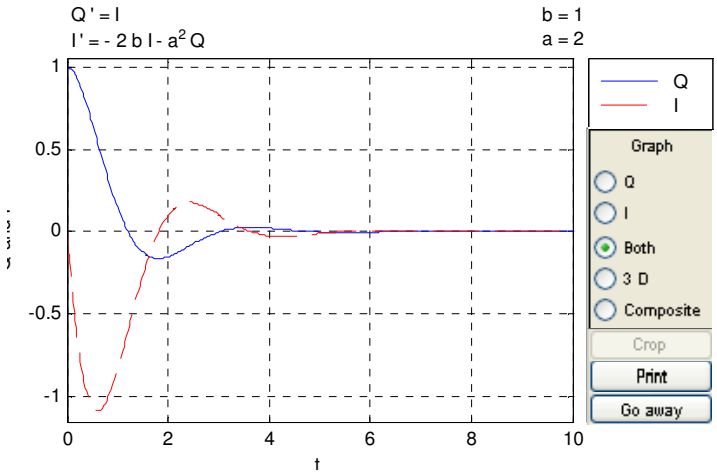
```
 9.61313663044136  -0.00007076817488   0.00012473984834
 9.79451656686478  -0.00004806586608   0.00012235354068
 9.97589650328820  -0.00002703398725   0.00010737772299
10.15727643971162  -0.00000957168207   0.00008551962529
10.33865637613504   0.00000368328314   0.00006078568982
10.52003631255846   0.00001251074025   0.00003690240402
10.70141624898189   0.00001729316412   0.00001643191040
10.88279618540531   0.00001874112560   0.00000033872526
```

Para representar los resultados obtenidos:

```
>> plot(S(:,1),S(:,2),'r') %rojo V(t).
>> hold on
>> grid on
>> plot(S(:,1),S(:,3),'g') %verde I(t)
```



Representamos de nuevo la gráfica obtenida con Pplane, con un intervalo definido, para verificar que coinciden las mismas.



e) Supongamos que F(t)=F0cos(wt) (oscilaciones forzadas) y b<a. Hallar la solución general para a=2, b=1, F0=1, y w=2, dibujar las trayectorias y las órbitas correspondientes a las condiciones iniciales (Q(0)=1,I(0)=0), y (Q(0)=10, I(0)=10). Dibujar alguna de las soluciones comprobando el crecimientote la amplitud de las oscilaciones para valores de b y w próximos a los valores de resonancia.

Si $F(t) = e^{-bt}cos(wt)$, ¿Se produce resonancia?.

En éste apartado interviene un esfuerzo externo en el sistema para que el mismo siga funcionando.

La ecuación será del tipo $F(t) = x'' + cx' + dx$

Tendremos que hallar la solución general y a partir de ésta, hallaremos una solución particular.

La indicaciones dadas en el problema, nos dice que es el caso estudiado en el apartado 3, b<a, oscilador subamortiguado.

La solución general de la ecuación será de la forma:

$$Y(t) = e^{(c\,t)}\big(c_1\,\text{c?s}(dt) + c_2 sen(dt)\big) + F_0\text{c?s}\,(wt)$$

Nuestra ecuación tendrá ahora dos apartados, un apartado inicial, correspondiente a la ecuación homogenea, la cual dependerá e los valores iniciales y que al ser subamortiguada tenderá a cero, (Será la parte transitoria), y un segundo componente que será la parte de la ecuación diferencial no homogénea (será la parte estacionaria).

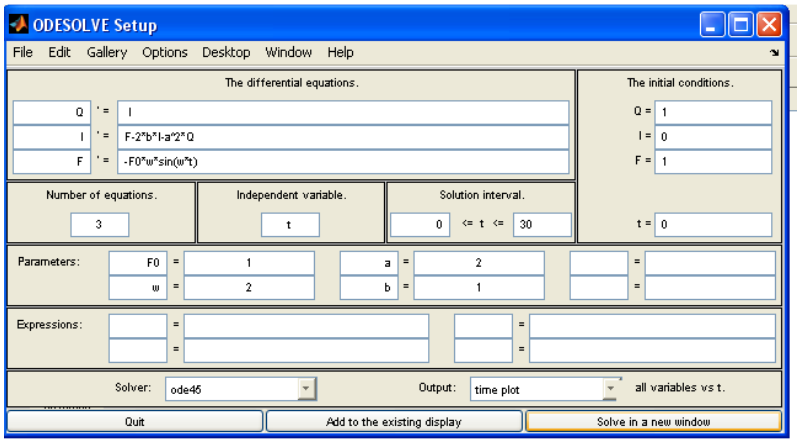
Para hacer el problema mediante Matlab, emplearemos el algoritmo ODESOLVE, e incluiremos una tercera ecuación con el aporte externo.

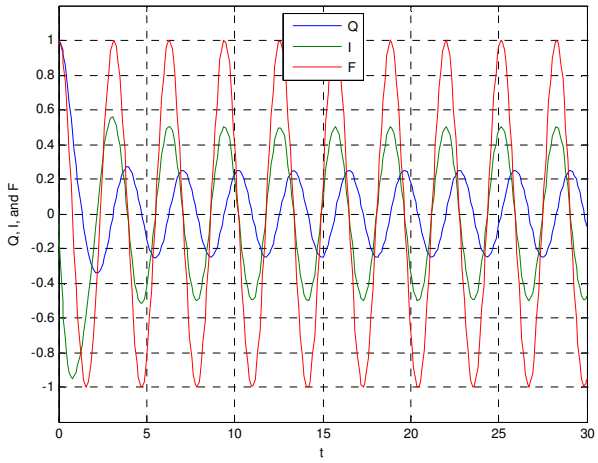
$$F(t) = F_0\text{c?s}\,(wt)$$

Derivamos para emplearla como ecuación, obteniendo:

$$F'(t) = F_0\text{w sen}\,(wt)$$

Si dibujamos la opción con los valores iniciales Q(0)=1, I(0)=0, obtenemos los siguientes parámetros:





Podemos observar en la gráfica anterior la existencia de una componente inicial transitoria, la cual se va convirtiendo a homogénea, debido a la influencia de la fuerza externa.

2.-La ecuación de Duffing

$$mx'' + cx' + kx + lx^3 = 0$$

es un modelo de un sistema masa-resorte no lineal y no forzado.

a) ¿Existe algún valor de los parámetros para los que el sistema anterior es un sistema hamiltoniano? En caso afirmativo encontrar la función hamiltoniana *H(x, y).*

Un sistema de ecuaciones lineales se llama sistema Hamiltoniano, si existe una función real H(x,y) tal que:

$$\frac{\partial x}{\partial t} = \frac{\partial H}{\partial y}$$

$$\frac{\partial y}{\partial t} = \frac{-\partial H}{\partial x}$$

Para toda x e y, la función H se denomina la función Hamiltoniana del sistema.

En un sistema Hamiltoniano, todos los puntos de equilibrio son centros o sillas.

Para que el sistema sea Hamiltoniano el mismo debe ser conservativo.

Para comprobar la existencia de un sistema Hamiltoniano, comenzaremos hallando las dos ecuaciones lineales :

$$mx'' + cx' + kx + lx^3 = 0$$

Hacemos x'=y

$$y' = \frac{-cy - kx - lx^3}{m}$$

Tenemos por tanto:

$$\frac{\partial x}{\partial t} = f(x,y) = y$$

$$\frac{\partial y}{\partial t} = g(x,y) = \frac{-cy - kx - lx^3}{m}$$

Comenzamos hallando inicialmente H(x,y), para ello despejaremos la ecuación siguiente, e integramos:

$$\frac{\partial x}{\partial t} = \frac{\partial H}{\partial y} \rightarrow \partial H = \frac{\partial x}{\partial t}\partial y \longrightarrow \int \partial H = \int y \partial y \longrightarrow H(x,y) = \frac{y^2}{2} + k$$

Debemos determinar k, para ello derivaremos la constante respecto a x, e igualaremos a la otra ecuación que define el Hamiltoniano:

(Hemos comentado antes que el sistema es conservativo,por lo tanto el rozamiento c=0.)

$$k'(x) = g(x,y) = \frac{\partial y}{\partial t} = \frac{-kx - lx^3}{m} = -\frac{dH}{dx}.$$

Integramos ahora respecto a x.

$$\int k'(x) = \int \frac{-kx - lx^3}{m} dx \longrightarrow k(x) = -\frac{kx^2}{2m} - \frac{lx^4}{4m}$$

Tenemos por tanto que el sistema Hamiltoniano queda del modo siguiente:

$$H(x,y) = \frac{y^2}{2} - \frac{kx^2}{2m} - \frac{lx^4}{4m}$$

b) Consideremos el caso en el que *k* = 16, Y *l* = 4. Calcular los puntos de equilibrio y clasificarlos. Usando el ordenador obtener el plano de fases correspondientes a diversas soluciones con diferentes Constantes de amortiguación, c= 0 (caso no amortiguado) y c

= 1,4. Para dichas órbitas dibuja los diagramas *t* frente a *x,* así como *t* frente a *y.* ¿Qué diferencias observas entre el caso no amortiguado y el caso amortiguado?

Partimos de las ecuaciones:

$$x' = y$$
$$y' = -cy - 16x - 4x^3$$

Para hallar los puntos de equilibrio:

$$y = 0$$
$$-16x - 4x^3 = 0$$

Obtenemos:

x=0.

x=+-2i.

Tenemos por tanto un solo punto de equilibrio en (0,0).

Para saber el tipo de equilibrio que tenemos, al igual que en el ejercicio anterior, calcularemos los autovalores,

Como tenemos un sistema no lineal, para proceder al cálculo de los autovalores, tendremos que hallar el Jacobiano de la función:

$$J = \begin{Bmatrix} \frac{\partial x}{\partial x} & \frac{\partial x}{\partial y} \\ \frac{\partial y}{\partial x} & \frac{\partial y}{\partial y} \end{Bmatrix} \longrightarrow \begin{Bmatrix} 0 & 1 \\ -16 - 12x^2 & -c \end{Bmatrix}$$

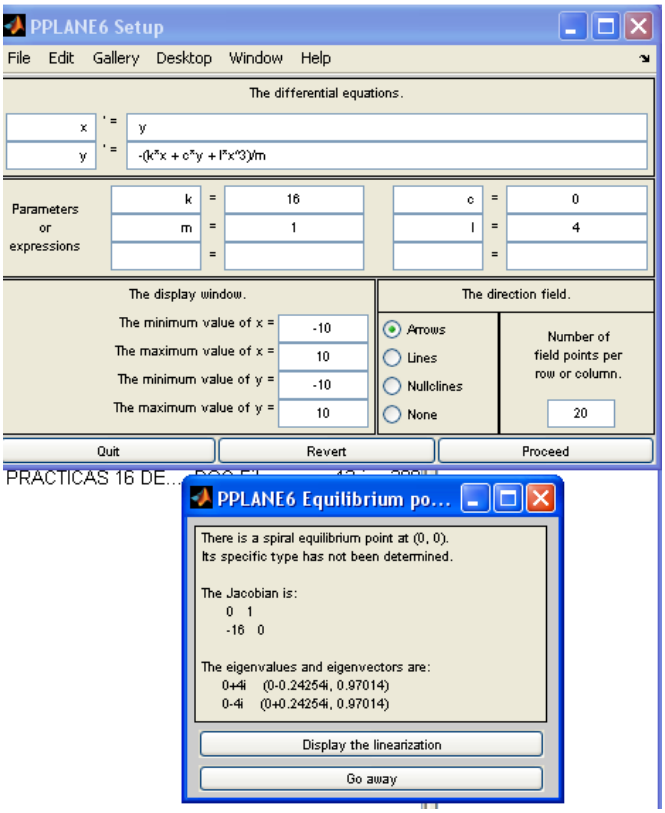
Para el caso no amortiguado c=0.

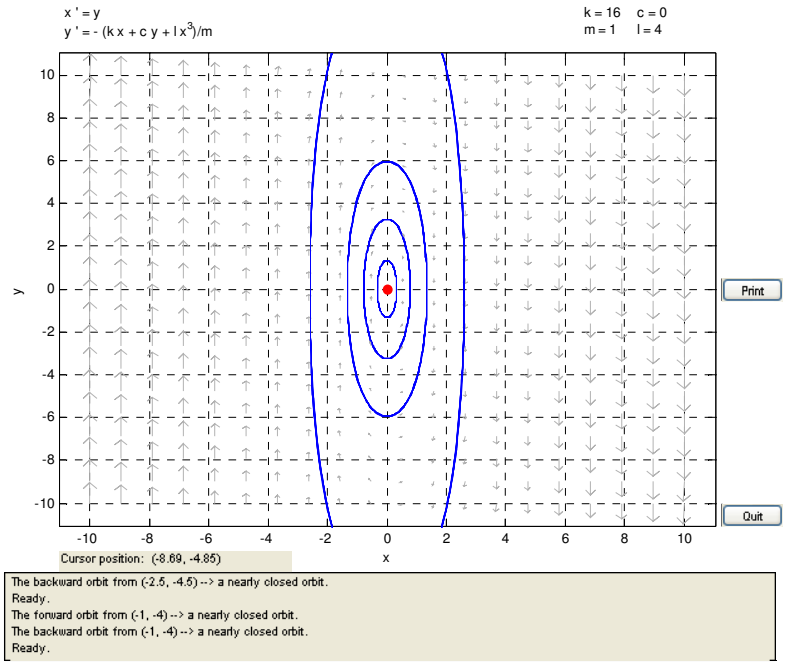
Tenemos que el único punto de equilibrio obtenido es el (0,0), si sustituimos en la Matriz del Jacobiano, y hallamos los autovalores, queda del siguiente modo:

$$[A - \gamma I] = \begin{Bmatrix} 0-\gamma & 1 \\ -16 - 12x^2 & -\gamma \end{Bmatrix} = \begin{Bmatrix} 0-\gamma & 1 \\ -16 & -\gamma \end{Bmatrix} = \gamma^2 + 16;\ \gamma = \pm 4i$$

Podemos observar que los autovalores son del tipo m=+-bi → Centro

Si introducimos las ecuaciones en el Pplane, observamos lo hallado anteriormente:





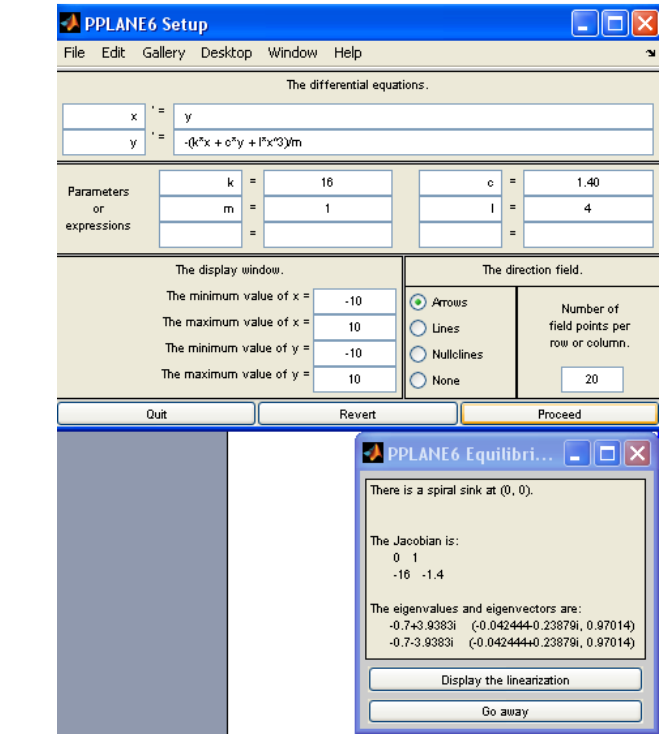
A continuación, hallamos los autovalores, para c=1.4.

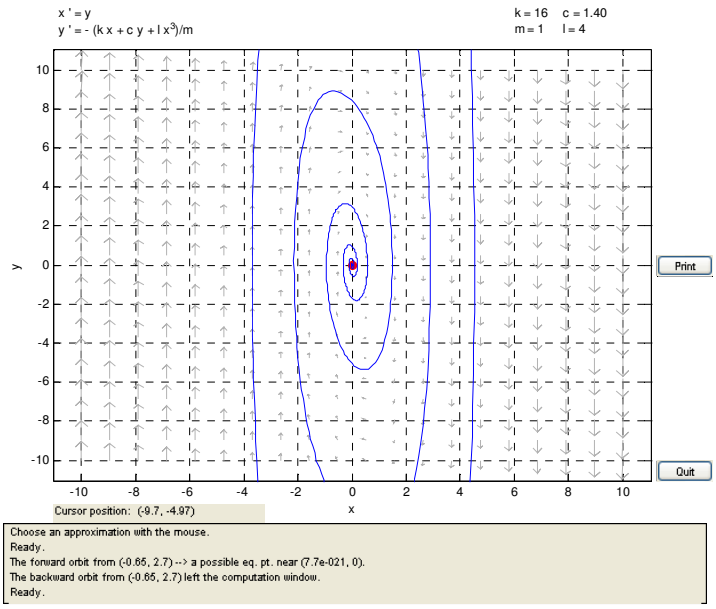
$$[A-\gamma I]=\begin{Bmatrix}-\gamma & 1\\ -16-12x^2 & 1.4-\gamma\end{Bmatrix}=\begin{Bmatrix}-\gamma & 1\\ -16 & 1.4-\gamma\end{Bmatrix}=\gamma^2+1.4\gamma+16;$$

$$\gamma=\frac{-1.4\pm\sqrt{1.4^2-64}}{2};\quad \gamma=-0.7\pm 3.93i$$

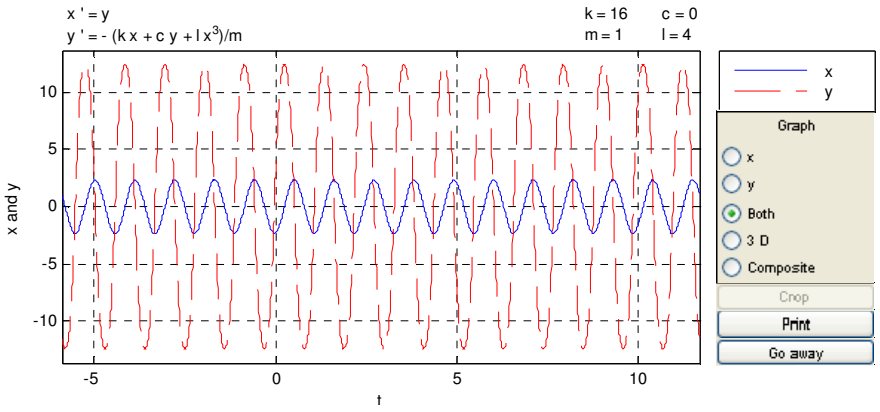
Tenemos los autovalores del tipo m=-a+-bi → a<0 espiral estable.

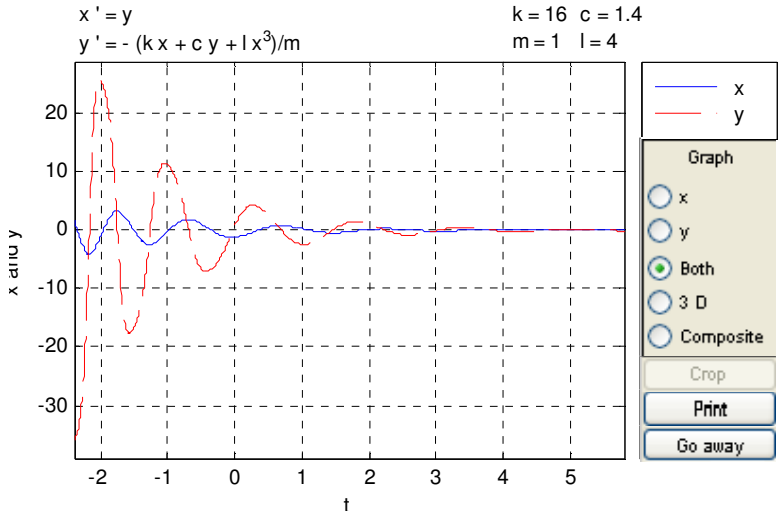
Introducimos los datos en Pplane, verificando lo obtenido anteriormente.





Representando los diagramas de fase en ambos casos observamos lo siguiente:





Observamos la diferencia de un sistema conservativo, a otro que no lo es, en el primero al no existir rozamiento, el sistema sigue oscilando indefinidamente, y en el el segundo caso, al no existir una fuerza externa, y si un rozamiento, el sistema tiende a detenerse.

c) Hacer lo mismo para el caso en el que *k* = 16, Y *l* = -4. ¿Existen órbitas heteroclinas conectando dos equilibrios? Dibujar las variedades estables e inestables de los equilibrios tipo silla. Dibujar las curvas de nivel *H(x,y)* = 15 y *H(x,y)* = 16.

Repetimos el proceso anterior, ahora desde las ecuaciones:

$$x' = y$$

$$y' = -cy - 16x + 4x^3$$

Buscamos los equilibrios:

$$y = 0$$

$$x(-16+4x^2)=0$$

Obtenemos los siguientes puntos de equilibrio:

(0,0), (-2,0), (2,0).

Hallamos el jacobiano, obteniendo posteriormente los autovalores, y de éste modo los tipos de equilibrio que tenemos.

$$J=\begin{Bmatrix}\frac{\partial x}{\partial x} & \frac{\partial x}{\partial y}\\ \frac{\partial y}{\partial x} & \frac{\partial y}{\partial y}\end{Bmatrix}\longrightarrow\begin{Bmatrix}0 & 1\\ -16+12x^2 & -c\end{Bmatrix}$$

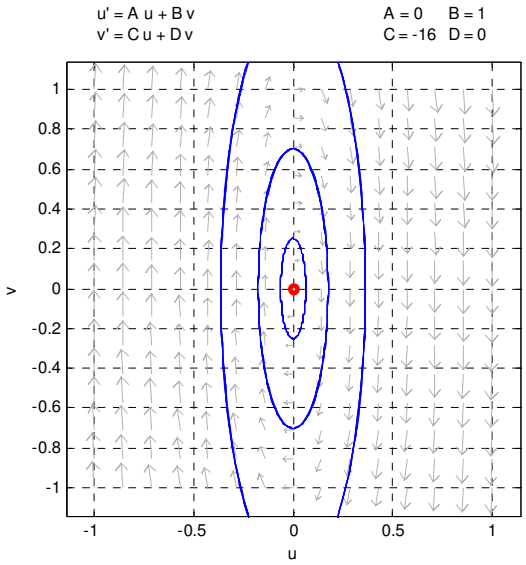
Para el caso c=0.

Punto de equilibrio (0,0).

$$[A-\gamma I]=\begin{Bmatrix}-\gamma & 1\\ -16+12x^2 & -\gamma\end{Bmatrix}=\begin{Bmatrix}-\gamma & 1\\ -16 & -\gamma\end{Bmatrix}=\gamma^2+16;\ \gamma=\pm 4i$$

Tenemos un punto de equilibrio tipo centro.

Si representamos la función obtenemos:



Para el caso c=0.

Punto de equilibrio (-2,0).

$$[A-\gamma I]=\begin{Bmatrix}-\gamma & 1\\ -16+12x^2 & -\gamma\end{Bmatrix}=\begin{Bmatrix}-\gamma & 1\\ 32 & -\gamma\end{Bmatrix}=\gamma^2-32;\ \gamma=\pm\sqrt{32}$$

Los equilibrios son del tipo m=+-a, por lo que tenemos equilibrio tipo silla.

Si linealizamos obtenemos:

Para el punto de equilibrio (2,0), el resultado es igual al anterior, como podemos ver.

$$[A-\gamma I]=\begin{Bmatrix}-\gamma & 1\\ -16+12x^2 & -\gamma\end{Bmatrix}=\begin{Bmatrix}-\gamma & 1\\ 32 & -\gamma\end{Bmatrix}=\gamma^2-32;\ \gamma=\pm\sqrt{32}$$

Para hallar las variables estable e inestables de los equilibrio tipo silla, tendremos que obtener los autovectores, procedentes de los autovalores del punto de equilibrio.

Por ejemplo en el caso del equilibrio (-2,0), tenemos los autovalores hallados anteriormente:

$$\gamma_1=+\sqrt{32}>0\longrightarrow Tenemos\ la\ variedad\ inestable$$
$$\gamma_2=-\sqrt{32}<0\longrightarrow Tenemos\ la\ variedad\ estable$$

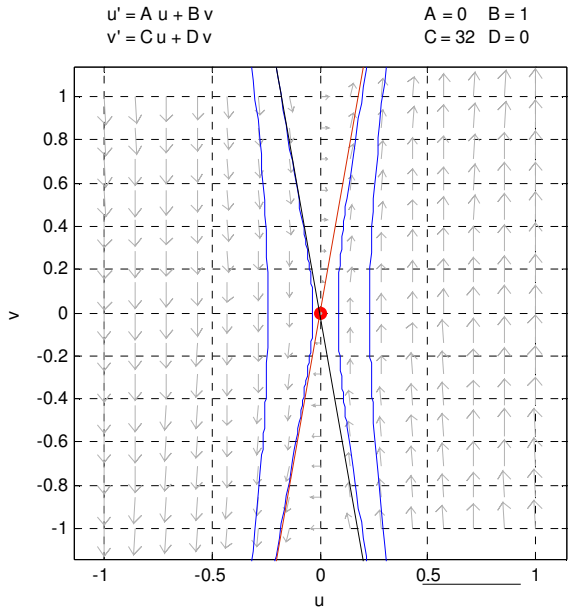
Para el caso de la variedad inestable.

$(A-\gamma I)\vec{x}=0\longrightarrow$ $\begin{pmatrix}-\sqrt{32} & 1\\ 32 & -\sqrt{32}\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}$=0 $\rightarrow -\sqrt{32}x_1+x_2=0\longrightarrow(1,\sqrt{32})$

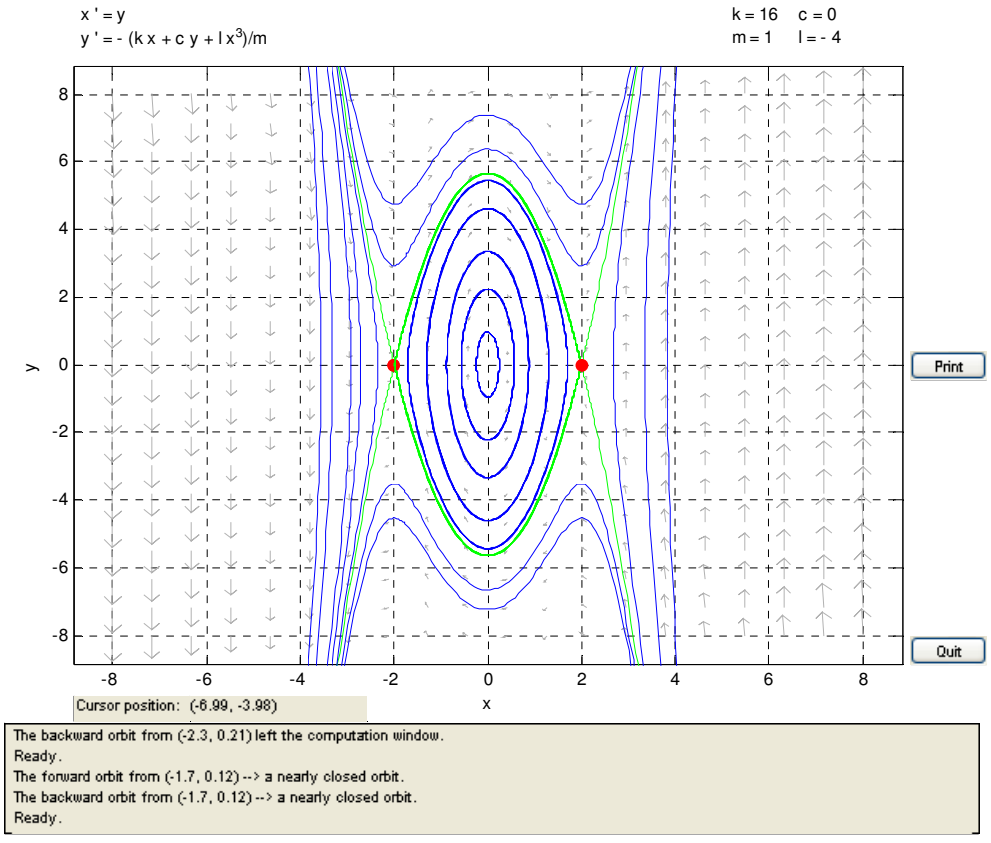
Para el caso de la variedad estable.

$(A-\gamma I)\vec{x}=0\longrightarrow$ $\begin{pmatrix}\sqrt{32} & 1\\ 32 & \sqrt{32}\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}$=0 $\rightarrow \sqrt{32}x_1+x_2=0\longrightarrow(1,-\sqrt{32})$

Carmelo Camacho González
Fernando J. Manobel Ponce



En la figura anterior, tenemos linealizado, el equilibrio correspondiente al punto (-2,0), con algunas líneas de equilbrio,y las variedades inestables (rojo) y estable(negro) del punto silla. Como vemos, coincide con los puntos hallados de forma teórica.(los cuales nos definen las rectas dibujadas).

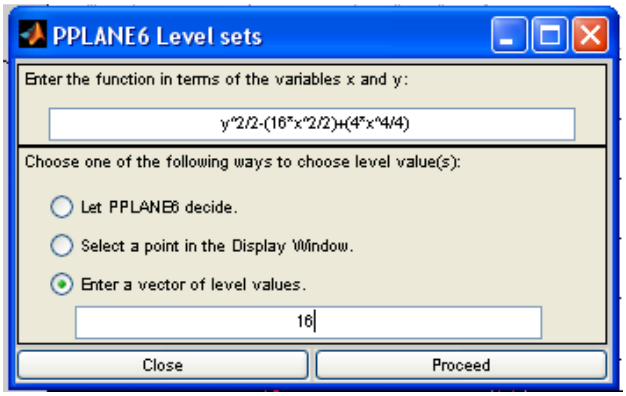


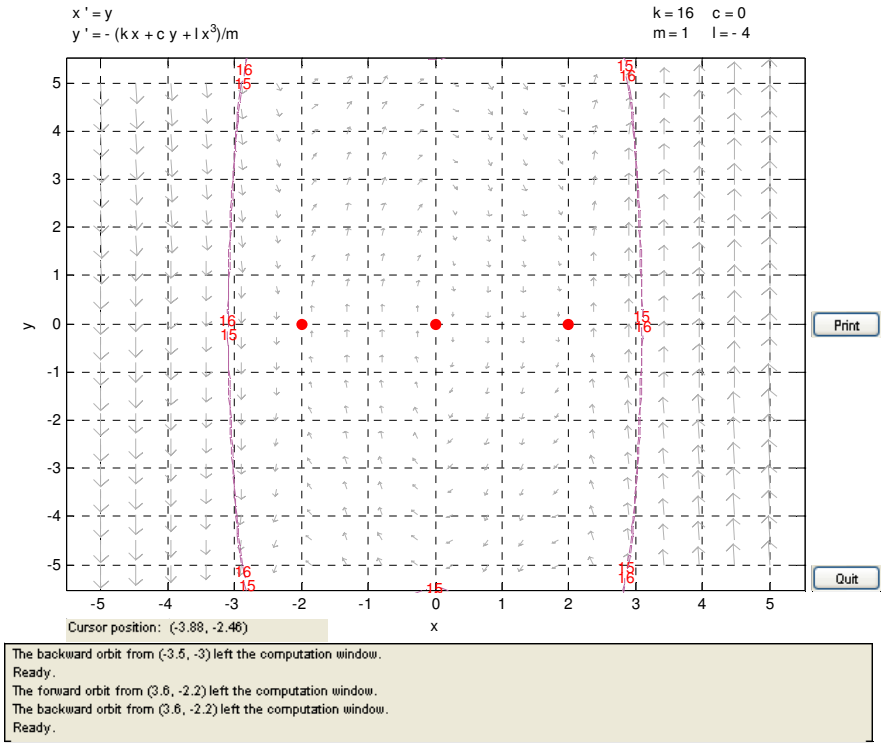
En a figura anterior, representamos el diagrama de fases completo, con los tres puntos de equilibrio, éste podemos observar la existencia de órbitas heteroclinas, uniendo los puntos sillas, situados en el (-2,0) y (2,0).

En cuanto a las curvas de nivel de un sistema Hamiltoniano, son las trayectorios de los planos de fase del sistema original.

Partimos por tanto de la función de Hamiltoniano, hallado en el primer apartado, sustituyendo los valores conocidos (k,l,m y c), y sacamos las curvas de nivel con el Pplane.

$$H(x,y) = \frac{y^2}{2} - \frac{kx^2}{2m} - \frac{lx^4}{4m} \longrightarrow H(x,y) = \frac{y^2}{2} - \frac{16x^2}{2} + \frac{4x^4}{4}$$





En el punto anterior, tenemos las curvas de nivel para H(x,y)=15, y 16.

Analizaremos el sistema con el caso amortiguado, c=1,4.

Buscamos los equilibrios:

$$y = 0$$
$$x(-16 + 4x^2) = 0$$

Obtenemos los mismos puntos de equilibrio que en el caso anterior.

(0,0), (-2,0), (2,0).

Hallamos el jacobiano, obteniendo posteriormente los autovalores, y de éste modo los tipos de equilibrio que tenemos.

$$J = \begin{Bmatrix} \frac{\partial x}{\partial x} & \frac{\partial x}{\partial y} \\ \frac{\partial y}{\partial x} & \frac{\partial y}{\partial y} \end{Bmatrix} \longrightarrow \begin{Bmatrix} 0 & 1 \\ -16 + 12x^2 & -1.4 \end{Bmatrix}$$

Punto de equilibrio (0,0).

$$[A - \gamma I] = \begin{Bmatrix} -\gamma & 1 \\ -16 + 12x^2 & -1.4 - \gamma \end{Bmatrix} = \begin{Bmatrix} -\gamma & 1 \\ -16 & -1.4 - \gamma \end{Bmatrix} = \gamma^2 + 1{,}4\gamma + 16;$$
$$\gamma = \frac{-1.4 \pm \sqrt{1.4^2 - 64}}{2} \longrightarrow \gamma = -0.7 \pm 3.9383i$$

Observamos que en el (0,0), los autovalores son del tipo –a+-bi, por lo que los equilibrios son del tipo espiral estable.

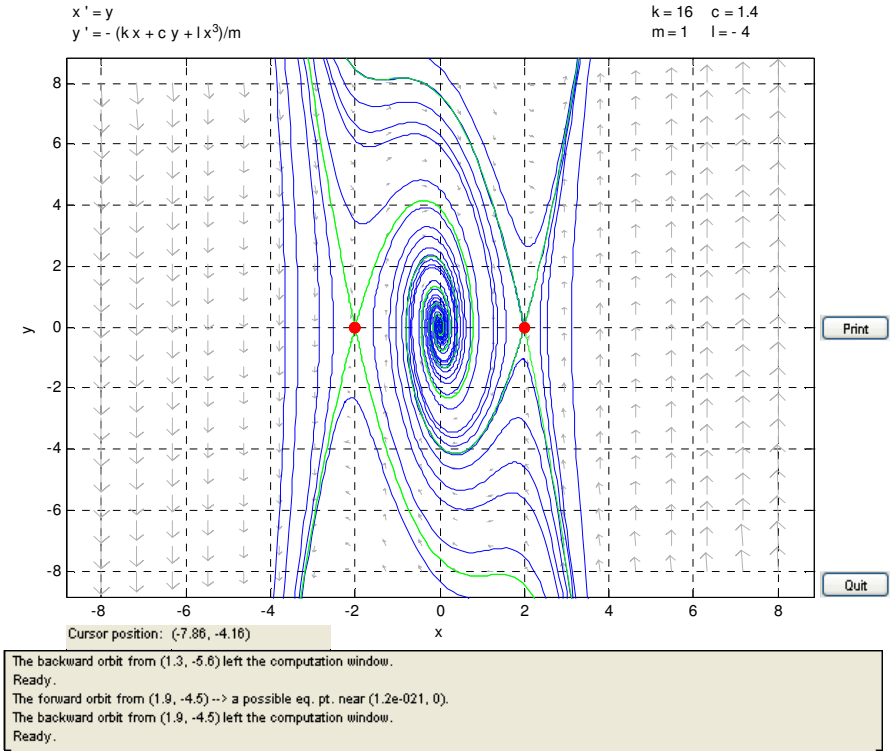
Punto de equilibrio (2,0).

$$[A - \gamma I] = \begin{Bmatrix} -\gamma & 1 \\ -16 + 12x^2 & -1.4 - \gamma \end{Bmatrix} = \begin{Bmatrix} -\gamma & 1 \\ 32 & -1.4 - \gamma \end{Bmatrix} = \gamma^2 + 1{,}4\gamma - 32;$$
$$\gamma = \frac{-1.4 \pm \sqrt{1.4^2 + 128}}{2} \longrightarrow \gamma = \pm 5$$

Punto de equilibrio (-2,0). Los autovalores obtenidos son iguales a los anteriores.

En los puntos (2,0) y (-2,0), tenemos equilibrios del tipo m1,-m2, por o que tenemos equilibro tipo silla.

La representación del plano de fase queda del siguiente modo:



d) Considerar por último el caso en el que *k* = -1, Y *l* = 1. Encontrar y clasificar todos los puntos de equilibrio para cada uno de los casos c = 0, y c = 1. ¿En qué caso existen órbitas homoclinas?

Ahora partimos del siguiente sistema de ecuaciones:

$$x' = y$$

$$y' = -cy + x - x^3$$

Buscamos los equilibrios:

$$y = 0$$

$$x(1 - x^2) = 0$$

Hallando los puntos de equilibrio en este caso, obtenemos los siguientes.

(0,0),(1,0) y (-1,0).

Procedemos igual que en los casos anteriores, primero para un sistema sin amortiguar c=0, y luego con un sistema amortiguado, c=1.

Para el caso c=0.

Punto de equilibrio (0,0).

$$[A - \gamma I] = \begin{Bmatrix} -\gamma & 1 \\ 1 - 3x^2 & -\gamma \end{Bmatrix} = \begin{Bmatrix} -\gamma & 1 \\ 1 & -\gamma \end{Bmatrix} = \gamma^2 - 1;\ \gamma = \pm 1$$

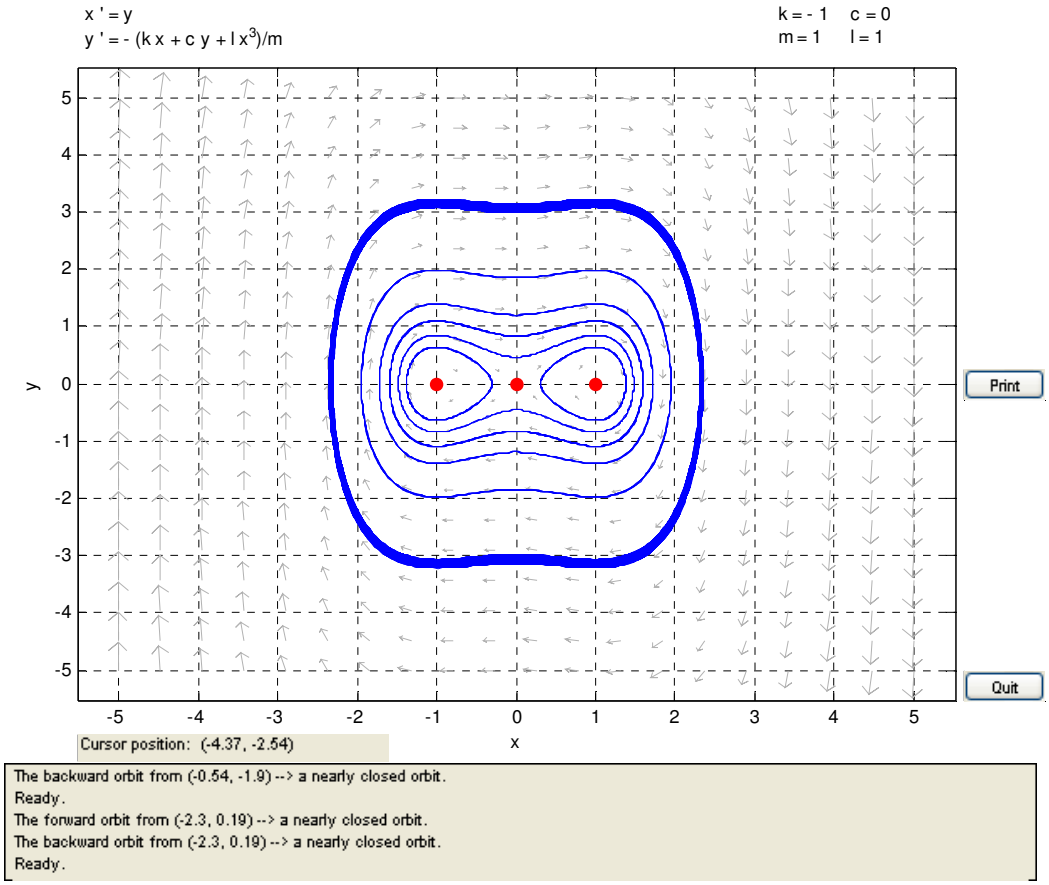
tenemos dos autovalores (+-1), por lo que tenemos un equilibrio tipo silla.

Punto de equilibrio (-1,0).

$$[A-\gamma I]=\begin{Bmatrix}-\gamma & 1\\ 1-3x^2 & -\gamma\end{Bmatrix}=\begin{Bmatrix}-\gamma & 1\\ -2 & -\gamma\end{Bmatrix}=\gamma^2+2;\ \gamma=\pm\sqrt{2}i$$

En el caso (-1,0),los autovalores son del tipo +-1.41 i, por lo que tenemos equilibrio del tipo centro, al igual que en el caso (0,1).

El equilibrio de fase queda representado por la siguiente figura:



Repetimos el caso para c=1, quedando del siguiente modo.

Punto de equilibrio (0,0).

$$[A-\gamma I]=\begin{Bmatrix}-\gamma & 1\\ 1-3x^2 & -1-\gamma\end{Bmatrix}=\begin{Bmatrix}-\gamma & 1\\ 1 & -1-\gamma\end{Bmatrix}=\gamma^2+\gamma-1;\ \gamma=0.6180,-1.6180$$

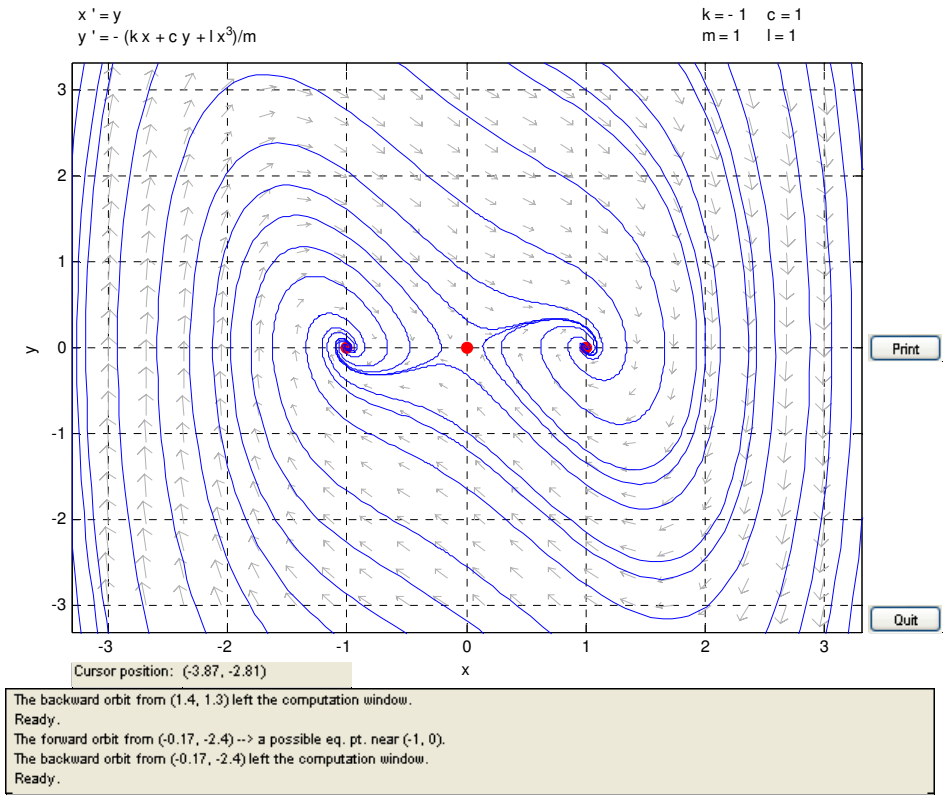
tenemos dos autovalores (+-m), por lo que tenemos un equilibrio tipo silla.

Punto de equilibrio (-1,0).

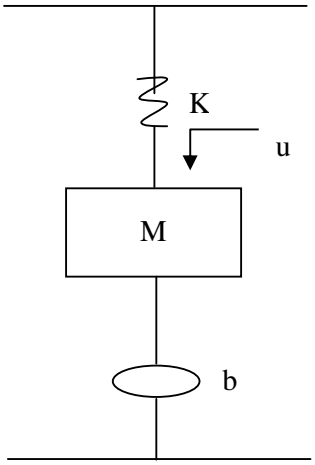
$$[A-\gamma I]=\begin{Bmatrix}-\gamma & 1\\ 1-3x^2 & -1-\gamma\end{Bmatrix}=\begin{Bmatrix}-\gamma & 1\\ -2 & -1-\gamma\end{Bmatrix}=\gamma^2+\gamma+2;\ \gamma=-0.5\pm 1.3229i$$

En el caso (-1,0),los autovalores son del tipo –a+-bi, por lo que tenemos equilibrio del tipo espiral estable, al igual que en el caso (1,0).

Representando con el Pplane, obtenemos el siguientes diagrama de fases.



3.2 MODELOS MECÁNICOS (ode45)



1.- La ecuación que describe el movimiento de dicho sistema mecánico es la siguiente:

Con m=1; b=0, k=8, g=0 y u=cos(omega*t).

a) **Representa la gráfica, del espacio y la velocidad frente al tiempo, para 0≤t≤20.**
b) **Calcula también aproximadamente y(15).**
c) **Sabiendo que la amplitud viene en función de b, k y omega. Representar la función A-omega para diferentes valores de b, y comenta los resultados obtenidos.**

---

a) Primero debemos crear el archivo f1Mec.m, para definir la función de forma adecuada para poder ser usado por el comando ode45.

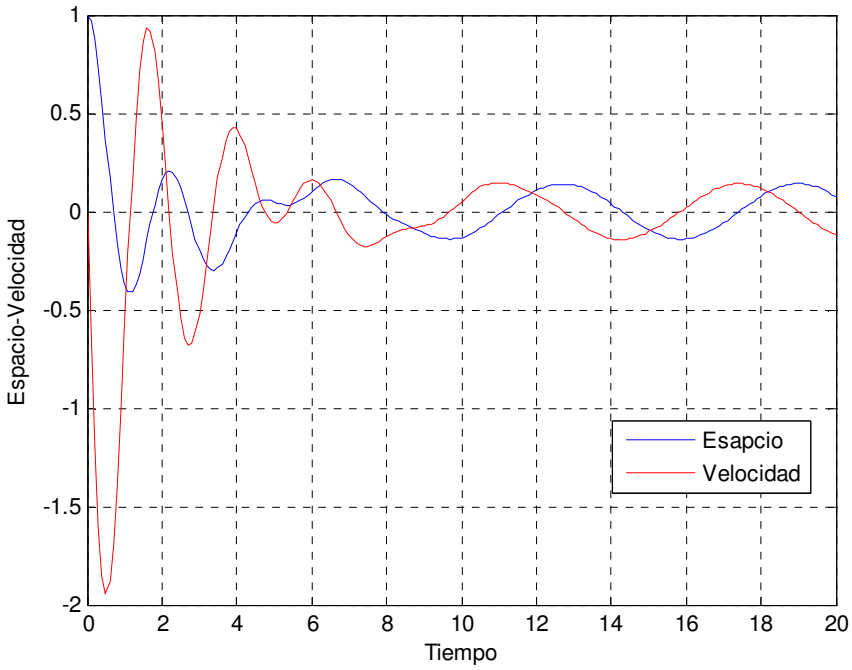
```
function z=f1Mec(t,y);
m=1;    %Masa.
b=1;    %Amortiguación del muelle
k=8;    %Constante de elasticidad del muelle.
omega=1;
u=cos(omega.*t); %Fuerza externa, como acción de un terremoto, viento, etc,...
g=0; %Gravedad
z(1)=(y(2)).*(1/m);
z(2)=(-k.*y(1)-b.*y(2)+u+g).*(1/m);
z=z';
```

Para usar el comando ode45 usaremos el algoritmo ode45sistema.m, es el siguiente:

```
function S=ode45sistema(f,a,b,Za,M)
%DATOS DE ENTRRADA
%f función, la cual esta almacenada como una cadena de caracteres
%a,b extremos del intervalo
%Za condiciones iniciales
%M es el número de pasos
h=(b-a)/M;
T=a:h:b;
[T X]=ode45(f,T,Za);
format long
S=[T X];
```

Ahora ya estamos preparados para ello:

```
S=ode45sistema(f,a,b,Za,M)
>> S=ode45sistema('f1Mec',0,20,[1 0],200);
>> plot(S(:,1),S(:,2),S(:,1),S(:,3),'r')
>> grid on
>> xlabel('Tiempo')
>> ylabel('Espacio-Velocidad')
>> legend('Esapcio','Velocidad','location','best')
```



b)

```
>> S=ode45sistema('f1Mec',0,20,[1 0],20)

S =
                0   1.00000000000000                0
  1.00000000000000  -0.36473416597396  -0.61933872617066
  2.00000000000000   0.16309913695295   0.45385590504976
  3.00000000000000  -0.19849069525214  -0.52077730658162
  4.00000000000000  -0.11009711906302   0.42275591393463
  5.00000000000000   0.04750394972072  -0.05787407790327
  6.00000000000000   0.09946856302431   0.16175632876414
  7.00000000000000   0.14211427707314  -0.12164556976960
  8.00000000000000  -0.01639839615993  -0.12854977586090
  9.00000000000000  -0.10995471117339  -0.07408346061125
 10.00000000000000  -0.13316597814337   0.05254767955723
 11.00000000000000  -0.01734627352077   0.14714282997158
 12.00000000000000   0.10686373203086   0.08645196998679
 13.00000000000000   0.13531747384817  -0.03696738750388
 14.00000000000000   0.03930158154176  -0.13812795384569
 15.00000000000000  -0.09370153629519  -0.10507996386913
 16.00000000000000  -0.13956053850319   0.02063723899734
 17.00000000000000  -0.05793206617356   0.12925775588211
 18.00000000000000   0.07752123028933   0.11834199696832
 19.00000000000000   0.14134652421079  -0.00122307221175
 20.00000000000000   0.07542885955116  -0.11959704831759
```

b) Creamos el archivo Amp.m para la función amplitud:

Amp.m

```
function y=Amp(omega)
k=8;
b=0.01;
y=1./sqrt(((k-(omega.^2) ).^2+b.^2.*(omega.^2)));
>> omega=linspace(0,6,100);
>> y=Amp(omega);
>> plot(omega,y)
```

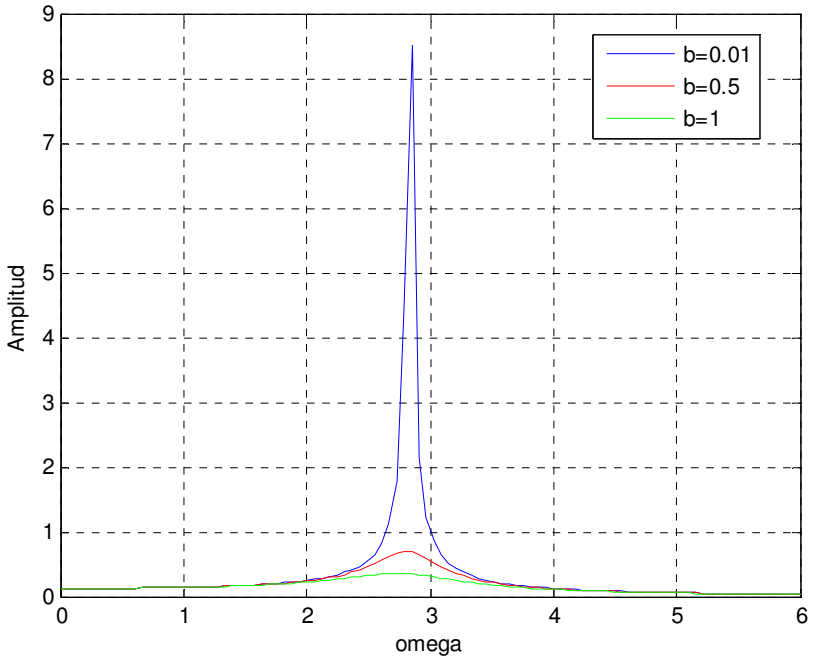
Amp.m

```
function y=Amp(omega)
k=8;
b=0.5;
y=1./sqrt(((k-(omega.^2) ).^2+b.^2.*(omega.^2)));
>> y=Amp(omega);
>> hold on
>> plot(omega,y,'r')
```

Amp.m

```
function y=Amp(omega)
```

```
k=8;
b=1;
y=1./sqrt(((k-(omega.^2) ).^2+b.^2.*(omega.^2)));
>> y=Amp(omega);
>> hold on
>> plot(omega,y,'g')
>> grid on
>> xlabel('omega')
>> ylabel('Amplitud')
>> legend('b=0.01','b=0.5','b=1','location','best')
```



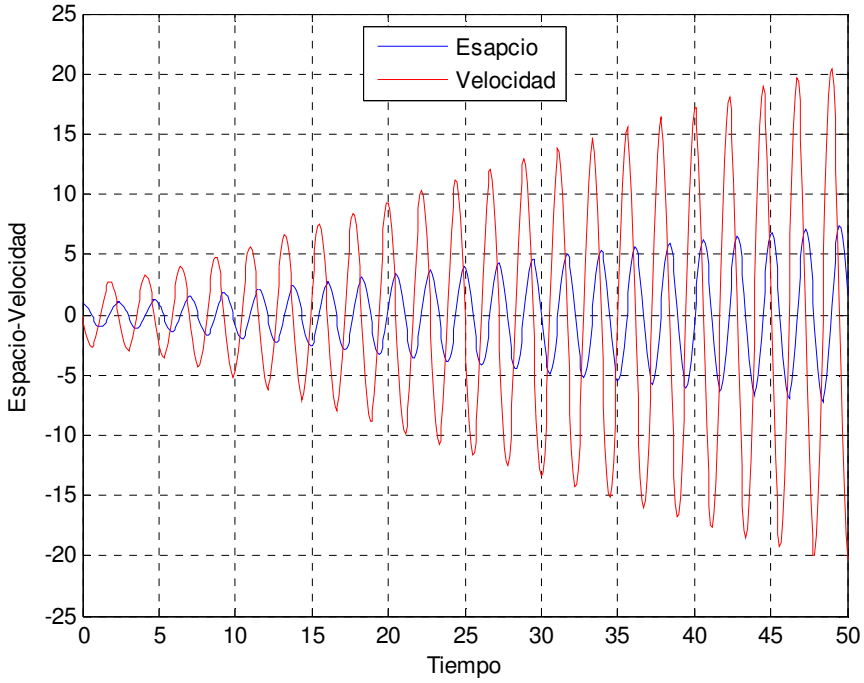
Como podemos observar omega siempre es máxima en el mismo punto para todas las funciones, precisamente en este punto es cuando la amplitud también es máxima, sobre todo cuando eliminamos o ajustamos a un valor muy pequeño el coeficiente de amortiguamiento, b.

Para dicha omega y b el sistema entraría en resonancia, veamos los efectos que tendría.

```
f1Mec.m
function z=f1Mec(t,y);
m=1;    %Masa.
b=0.01;    %Amortiguación del muelle
k=8;    %Constante de elasticidad del muelle.
omega=2.8141;
u=cos(omega.*t); %Fuerza externa, como acción de un terremoto, viento, etc,...
```

```
g=0;


z(1)=(y(2)).*(1/m);
z(2)=(-k.*y(1)-b.*y(2)+u+g).*(1/m);
z=z';
```

```
>> S=ode45sistema('f1Mec',0,50,[1 0],500);
>> plot(S(:,1),S(:,2),S(:,1),S(:,3),'r')
>> grid on
>> xlabel('Tiempo')
>> ylabel('Espacio-Velocidad')
>> legend('Esapcio','Velocidad','location','best')
```



Como se puede comprobar el sistema entra en resonancia y no llegaría nunca a un equilibrio.

2.- El puente del estrecho de Tacoma, fue abierto el 1 de Julio 1940 al transito rodado, tubo un coste de 6 millones de dólares. El 7 de Noviembre del mismo año, durante un vendaval, el puente se rompió y se vino abajo. Esto fue ocasionado porque la superficie de rodadura oscilaba exageradamente con el viento.

Este tipo de puente era colgante y lo hacía de unos cables verticales unidos a cables soportados por torres. Si consideramos que los cables son largos resortes, entonces es tentador modelar las oscilaciones de la superficie de rodamiento con una ecuación de oscilador armónico. Es posible que el viento proporcione un forzamiento periódico, podríamos así suponer que el colapso se debió a la resonancia.

El asunto no es tan simple, para obtener las ecuaciones que describan el movimiento del puente nos basaremos en los estudios de *"A. C. Lazer y P. J. McKenna"* . Suponemos que el puente solo oscila hacia arriba y hacia abajo, por tanto solo nos hará falta una variable, y(t).

Para y=0, el puente estará en reposo, y>0, los cables estarían flojos y para y<0, estarían tensos.

La ecuación en si desarrollada por *Lazer* y *Mckenna* para describir el movimiento de un puente colgante en un día sin viento sería:

$$\frac{d^2y}{dt^2} + \alpha\frac{dy}{dt} + \beta y + c(y) = -g$$

El primer termino es la aceleración, el segundo, surge del amortiguamiento, se cree que el termino α es pequeño debido a que los puentes colgantes son estructuras flexibles, el tercer término vendría de la rigidez del puente, por último la función c(y) tiene en cuenta el jalón del cable:

$$c(y) = \begin{cases} \gamma y, & si \;\; y < 0 \\ 0, & si \;\; y \geq 0 \end{cases}$$

Por supuesto g es la constante de gravedad. Podemos convertir dicha ecuación diferencial en un sistema autónomo de ecuaciones de la siguiente forma.

$$\begin{cases} \dfrac{dy}{dt} = v \\ \dfrac{dv}{dt} = -\beta y - c(y) - \alpha v - g \end{cases}$$

Podemos simplificar el sistema de la siguiente forma:

$$\begin{cases} \dfrac{dy}{dt} = v \\ \dfrac{dv}{dt} = -h(y) - \alpha v - g \end{cases}$$

Con:

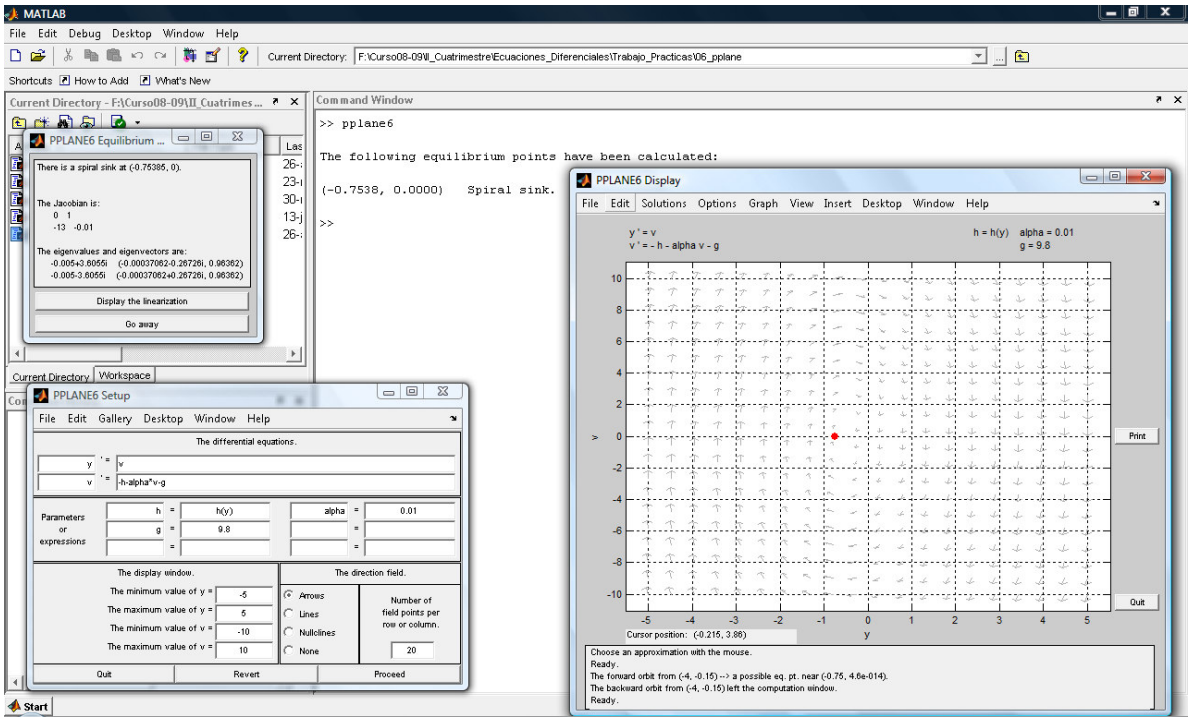
$$h(y) = \begin{cases} ay, & si \;\; y < 0 \\ by, & si \;\; y \geq 0 \end{cases}$$

Para estudiar el ejemplo de una forma numérica daremos los siguientes valores, a=17, b=13, α=0.01 y g=9.8. Para el estudio de este sistema utilizaremos el algoritmo pplane6 de Matlab.

Primero definiremos la función h(y), de la siguiente forma:

h.m

```
function z=h(y)
z=17.*y;
if y<0
   z=13.*y;
   if y>=0
   end
end
```



Como podemos comprobar el sistema solo tiene un punto de equilibrio, en (-0.7538,0), un foco estable. De forma que cualquier punto tiende de forma muy lenta al punto de equilibrio.

A pesar de lo complicado de la empresa, supondremos por simplicidad, que el viento ejerce un empuje de la forma λsin(μt). Al ser imposible que los vientos den un forzamiento de amplitud o frecuencia contante estudiaremos un rango de ellas.

Por ejemplo escogeremos un μ=4 y un λ=<0.05 (muy pequeña), toda solución periódica tenderá al punto de equilibrio (-0.7538,0), por tanto nuestro sistema de ecuaciones podría quedar de la siguiente manera.

$$\begin{cases} \dfrac{dy}{dt} = v \\ \dfrac{dv}{dt} = -17y - 0.01v - 9.8 + \lambda \sin(4t) \end{cases}$$

Esta vez resolveremos el sistema de ecuaciones mediante el comando ode45 de Matlab, definiremos igualmente las funciones necesarias y usaremos el algoritmo del ejercicio 1 de esta sección:

h.m

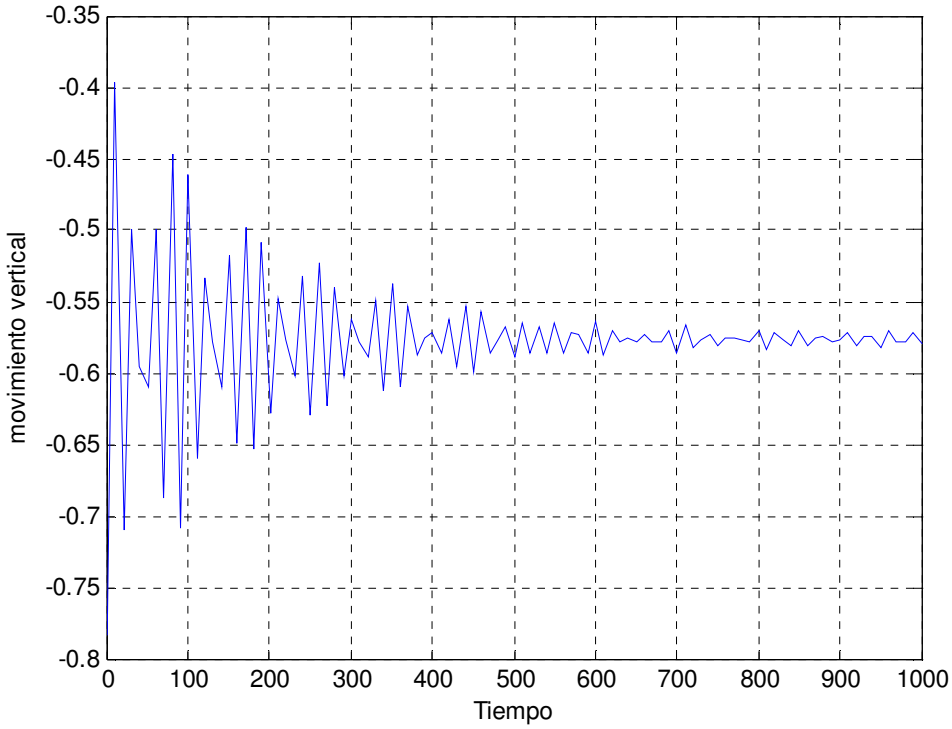
```
function z=h(yy(1))
z=17.*yy(1);
if yy(1)<0
   z=13.*yy(1);
   if yy(1)>=0
   end
end
```

fvi.m

```
function Z=fvi(t,yy)
alpha=0.01;
g=9.8;
la=0.005;
nu=4;
Z(1)=yy(2);
Z(2)=-17.*yy(1)-alpha.*yy(2)-g+la.*(sin(nu.*t));
Z=Z';
```

```
>> S=ode45sistema('fvi',1,1000,[-0.7834 0],100);
>> plot(S(:,1),S(:,2))
>> grid on
>> ylabel('movimiento vertical')
>> xlabel('Tiempo')
```

Carmelo Camacho González
Fernando J. Manobel Ponce

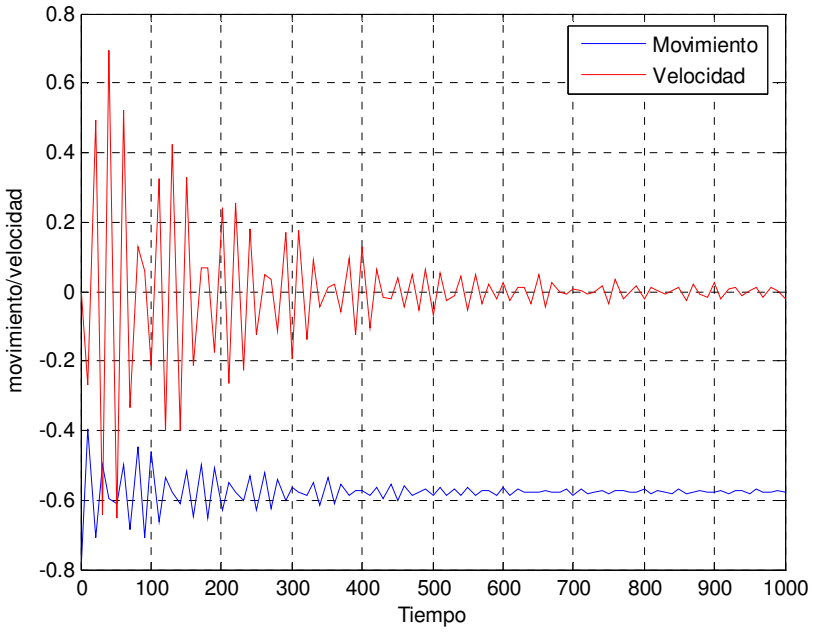


Como vemos si el término λ es muy pequeño al final, el puente tiende poco a poco a su posición de equilibrio.

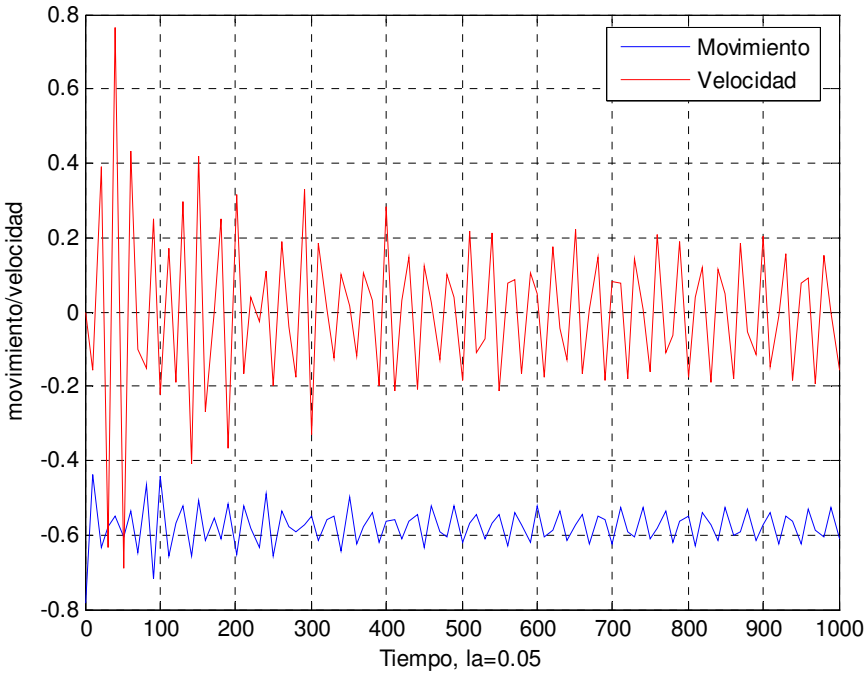
Veamos ahora la componente velocidad como varía:

```
>> plot(S(:,1),S(:,2),S(:,1),S(:,3),'r')
>> ylabel('movimiento/velocidad')
>> xlabel('Tiempo')
>> grid on
>> legend('Movimiento','Velocidad','location','best')
```

Carmelo Camacho González
Fernando J. Manobel Ponce

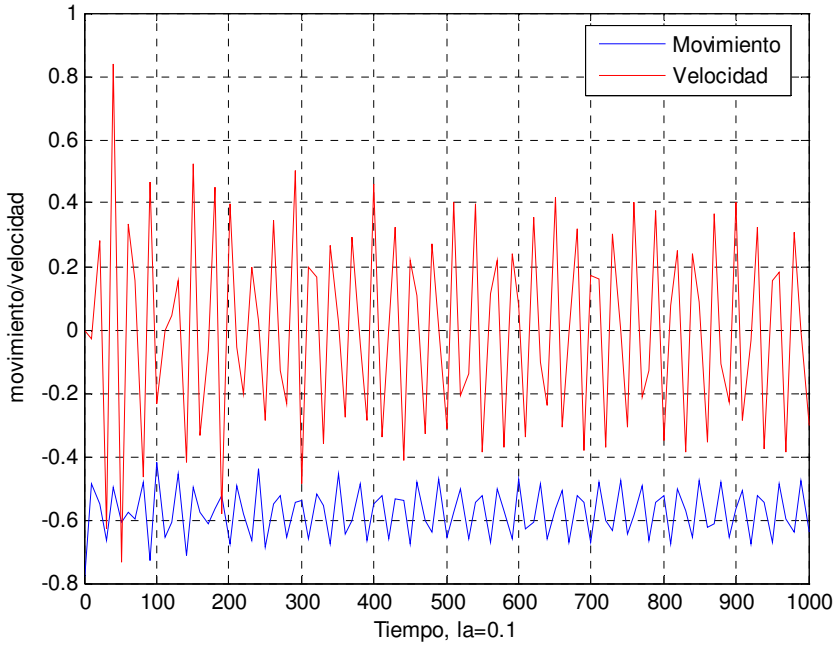


Para λ=0.05.

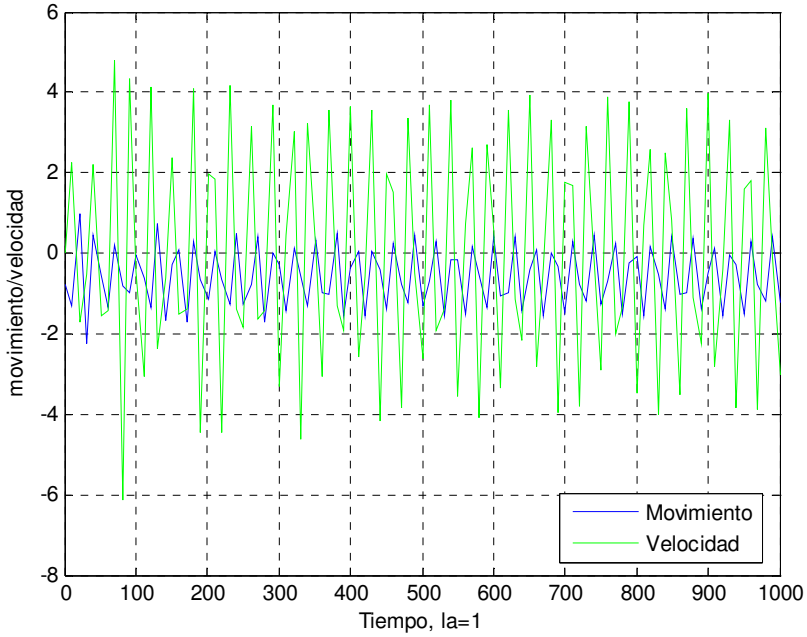


Para λ=0.1

Carmelo Camacho González
Fernando J. Manobel Ponce

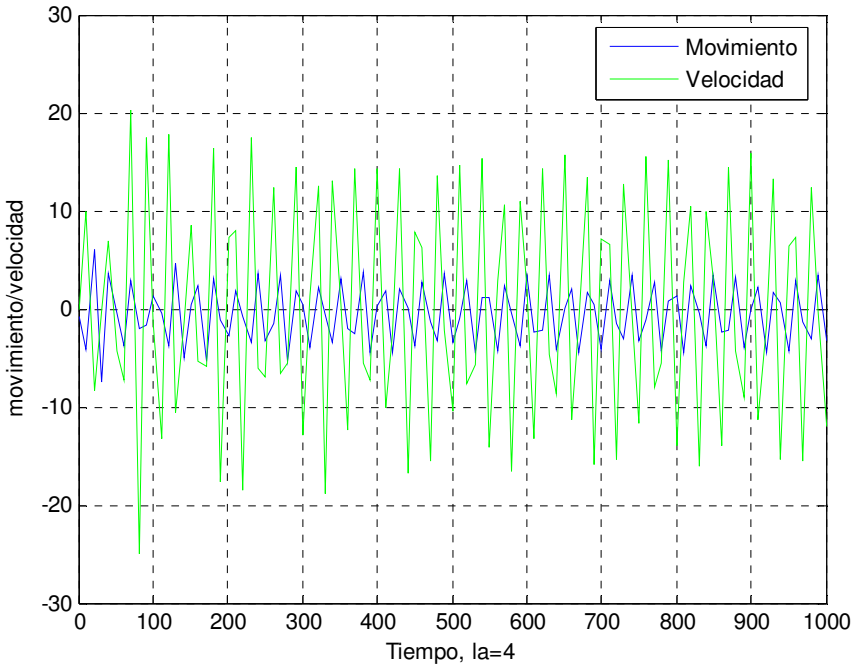


Para λ=1



Para λ=4

Carmelo Camacho González
Fernando J. Manobel Ponce



Como podemos observar cada vez que se aumenta el forzamiento del puente este oscila más y más. Aun así, se recuerda que esto es solo un ejemplo muy simple de este tipo de problemas.

4. PROBLEMAS DE CONTORNO.

4.1 DIFERENCIAS FINITAS Y DISPARO LINEAL

1.- Sea el problema de contorno:

$$y'' = y' + 2y + \cos(t) \;\; ; \;\; y(0) = -0.3 \;\; ; \;\; y\left(\frac{\pi}{2}\right) = -0.1 \;\; ; \;\; 0 < y < \left(\frac{\pi}{2}\right)$$

Cuya solución es: $y(t) = -\frac{1}{10}(\sin(t) + 3\cos(t))$

Hallar la solución y su derivada, así como el error cometido para cada una de ellas, cuando el tamaño de paso sea h=pi/4 y h=pi/8.

Usaré el algoritmo de Diferencias finitas para la resolución de la ecuación diferencial.

```
function D=diferenciasfinitas(p,q,r,a,b,alfa,beta,M)

h=((b-a)/(M+1));
T=linspace(a+h,b-h,M);

% construccion de la matriz tridiagonal de orden MxM
for i=1:M
A(i,i)=2+h.^2.*feval(q,T(i));
end
for i=1:M-1
A(i,i+1)=-1+(h/2).*feval(p,T(i));
A(i+1,i)=-1-(h/2).*feval(p,T(i+1));
end

% construccion del vector de los terminos independientes de orden 1xM
B(1)=-(h.^2).*feval(r,T(1))+(1+(h/2).*feval(p,T(1))).*alfa;
B(M)=-(h.^2).*feval(r,T(M))+(1-(h/2).*feval(p,T(M))).*beta;
for i=2:M-1
B(i)=-(h.^2).*feval(r,T(i));
end

% resolución del sistema tridiagonal para hallar la solución Y
Y=A\B';
Y=[alfa;Y;beta];

% vector con las abscisas
T=[a T b];
errorY=abs(Y-YE(T)');
% construcción del vector con la derivada de la solución Yprima y error
Yprima(1)=(Y(2)-Y(1))/h;
Yprima(M+2)=(Y(M+2)-Y(M+1))/h;
for i=2:M+1
Yprima(i)=(Y(i+1)-Y(i-1))/(2.*h);
end
y' + 2y + c□s(t)
Yprimaprima=Yprima'+2.*Y+cos(T)'; %AQUI SE COLOCARÁ LA ECUACIÓN DEL PROBLEMA EN CUESTIÓN.
```

```
37. errorYprima=abs((dYE(T))-Yprima);
38.
39. % matriz con los datos de salida
40. format long
41. disp('        T                Y                errorY          Yprima          errorYprima
    Yprimaprima')
42. D=[T' Y errorY Yprima' errorYprima' Yprimaprima];
```

Defino las siguientes funciones:

p.m

```
function y=p(t)
y=1;
```

q.m

```
function y=q(t)
y=2;
```

r.m

```
function y=r(t)
y=cos(t);
```

YE.m

```
function y=YE(T);
y=-(1/10).*(sin(T)+3.*cos(T));
```

dYE.m

```
function y=dYE(T)
%Derivada de la solución exacta del problema.
y=-(1/10).*(cos(T)-3.*sin(T));
```

Para h=pi/4.

```
>> D=diferenciasfinitas('p','q','r',0,pi/2,-0.3,-0.1,1)

        T              Y          errorY        Yprima         errorYprima       Yprimaprima
D =
        0        -0.3000      0.0000      0.1863          0.2863          0.5863
   0.7854        -0.1537      0.1292      0.1273          0.0141          0.5271
   1.5708        -0.1000      0.0000      0.0683          0.2317          -0.1317
```

Para h=pi/8

```
>> D=diferenciasfinitas('p','q','r',0,pi/2,-0.3,-0.1,3)

        T              Y          errorY        Yprima         errorYprima       Yprimaprima
D =
        0        -0.3000      0.0000     -0.0399      0.0601      0.3601
   0.3927        -0.3157      0.0003      0.0218      0.0007      0.3143
```

0.7854	-0.2829	0.0001	0.1384	0.0030	0.2797
1.1781	-0.2070	0.0002	0.2329	0.0060	0.2016
1.5708	-0.1000	0.0000	0.2725	0.0275	0.0725

2.- Representemos por u el potencial electrostático entre dos esferas metálicas concéntricas de radios R_1 y R_2 con $R_1<R_2$, tales que el potencial de la esfera interior se mantenga constante en V1 voltios y potencial de la esfera exterior en 0 voltios. El potencial en la región situada entre ambas esferas está regido por la ecuación de Laplace, que en esta aplicación particular se reduce a:

$$\frac{d^2u}{dr^2} = -\frac{2}{r}\frac{du}{dr} \qquad R_1 < r < R_2 \quad u(R_1) = V_1; \quad u(R_2) = 0 \quad V_1 = 110; \quad R_1 = 2mm; \quad R_2 = 4mm$$

a) **Aproxime u(3) con 40 subintervalos por medio del algoritmo de disparo lineal.**

b) **Compare los resultados obtenidos en el apartado a con el potencial real u(3), donde**

$$u(r) = \frac{V_1 R_1}{r}\left(\frac{R_2 - r}{R_2 - R_1}\right)$$

```
function D=disparo(F1,F2,a,b,alfa,beta,M)

% Solución del sistema F1
h=(b-a)/M;
T=linspace(a,b,M+1);
Za=[alfa,0];
[T,Z]=ode45(F1,T,Za);
U=Z(:,1);
Uprima=Z(:,2);

% Solución del sistema F2
Za=[0,1];
[T,Z]=ode45(F2,T,Za);
V=Z(:,1);
Vprima=Z(:,2);

% Solución del problema de contorno y error cometido
Y=U+(beta-U(M+1))*V/V(M+1);

%Solución exacta y error.
%errorY=abs((YE(T))-Y);

% Derivada de la solución del problema de contorno y error cometido
Yprima=Uprima+(beta-U(M+1))*Vprima/V(M+1);

%Error en Yprima.

errorYprima=abs((dYE(T))-Yprima);%AQUI CAMBIOS

% Yprimaprima.
```

```
31. Yprimaprima=-(2./T).*Yprima;
32.
33. % Datos de salida
34. format long
35. disp('     T       Y       errrorY    Yprima          errorYprima      YprimaPrima')
36. D=[T,Y,errorY,Yprima,errorYprima,Yprimaprima];
```

Para responder al apartado a y b usaré el algoritmo de disparo lineal con las siguientes funciones:

p.m

```
function y=p(t)
y=(-2./t);
```

q.m

```
function y=q(t)
y=0;
```

r.m

```
function y=q(t)
y=0;
```

YE.m

```
function y=YE(T);
y=((4-T)*110)./T;
```

dYE.m

```
function y=dYE(T)
y=(-4./T.^2).*110;
```

ftysis1.m

```
function Z=ftysis1(t,Z)
u1=Z(1);
u2=Z(2);
Z=[u2,u2.*p(t)+u1.*q(t)+r(t)]';
```

ftysis2.m

```
function Z=ftysis2(t,Z)
v1=Z(1);
v2=Z(2);
Z=[v2,v2.*p(t)+v1.*q(t)]';
```

```
>> D=disparo('ftysis1','ftysis2',2,4,110,0,40);
>> D(21,1:3)
          R                                U                          errorU
  3.00000000000000  36.66666536310768   0.00000130355898
```

3.- Considere la deflexión de una viga con los extremos soportados sujetos a una carga uniforme. El problema con valor de frontera que rige esta situación física es:

$$\frac{d^2\omega}{dx^2} = \frac{S}{EI}\omega + \frac{qx}{2EI}(x-l),\ \ 0 < x < l,\ \ \omega(0) = \omega(l) = 0$$

Suponga que la viga es de acero y del tipo W10.

a) **Aproxime la deflexión de la viga w(x) cada 6 plg mediante el método de las diferencias finitas.**

b) **Dada la ecuación real. ¿Es el error máximo en el intervalo menor que 0.02 pulgadas.**

c) **La ley estatal de la construcción estipula que $\max_0<x<l w(x)<1/300$ ¿Cumple la viga con el código estatal?.**

Para responder a los apartados a) y b), escribimos los siguientes archivos de Matlab para usarlos con el de Diferencias Finitas.

OJO QUE HAY QUE PASAR LA *"q"* A lb/plg, SI LA DEJAMOS EN lb/pie NO TENDRÍA LAS MISMAS UNIDADES QUE EL RESTO.

p.m

```
function y=p(t)
y=0;
```

q.m

```
function y=q(t)
S=1000;
E=30000000;
I=625;
y=S/(E*I);
```

r.m

```
function y=r(t)
q=100/11.99;  %OJO que he pasado de lb/pie a lb/plg.
l=120;
E=30000000;
I=625;
y=(q*t*(t-l))/(2*E*I);
```

YE.m

```
function y=YE(T);
c1=7.7042537e+004;
a=2.3094010e-004;
c2=7.9207462e+004;
b=-4.1666666e-003;
l=120;
c=-1.5625e+005;
y=c1.*exp(a.*T)+c2.*exp(-a.*T)+b.*(T-l).*T+c;
% Es la función exacta.
```

El algoritmo de diferencias finitas que hemos confeccionado incluye la función de la solución exacta para que muestre el error cometido en todo momento y así poder comprobar de una sola vez si el error es mayor al que se indica en el apartado b).

Procedemos a operar con Matlab:

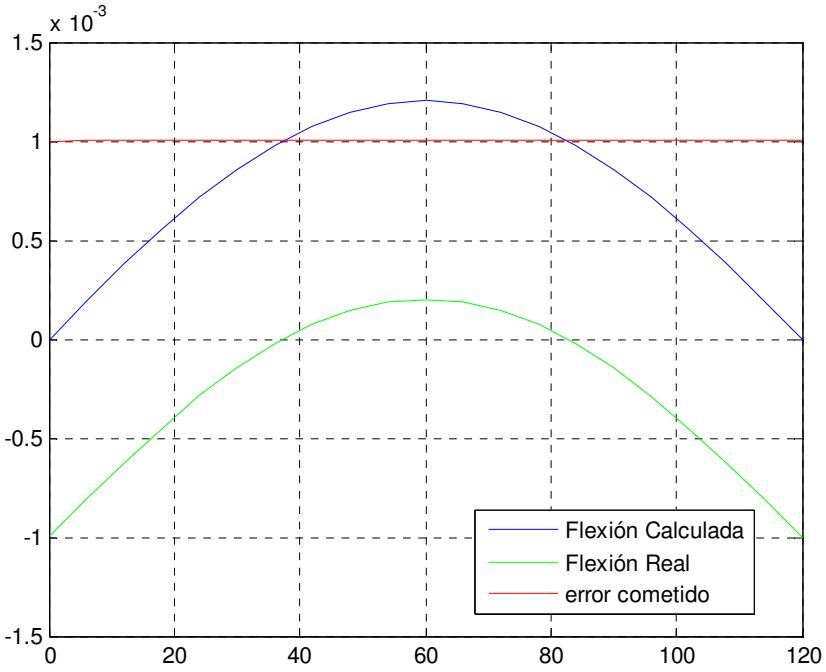
```
>> D=diferenciasfinitas('p','q','r',0,120,0,0,19);
>> error=max(D(:,3))
error = 0.00100460465060
```

Como podemos comprobar el error es menor a las 0.02 plg indicadas en el apartado b).

A continuación paso a representar las graficas de la flexión calculada, la real y el error cometido:

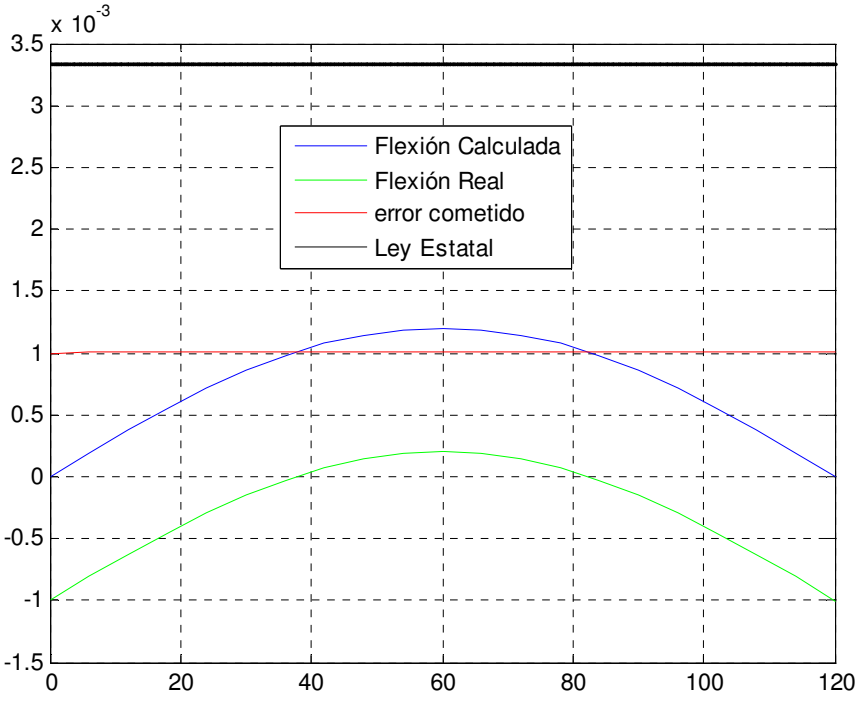
```
>> plot(D(:,1),D(:,2))
>> T=D(:,1);
>> y=YE(T);
>> hold on
>> plot(T,y,'g')
>> hold on
>> plot(D(:,1),D(:,3),'r')
>> grid on
>> hold on
>> legend('Flexión Calculada','Flexión Real','error cometido','location','best')
```

Carmelo Camacho González
Fernando J. Manobel Ponce



Para el apartado c) añadimos las siguientes órdenes para completar la gráfica:

```
>> x=linspace(0,120,1000);
>> ley=(1/300);
>> hold on
>> plot(x,ley,'black')
>> legend('Flexión Calculada','Flexión Real','error cometido','Ley Estatal','location','best')
```



Como vemos la viga cumple con la ley estatal sobradamente, **OJO**, el ejercicio resuelto en clase da otros valores ya que a tomado la q=100lb/pie y no la ha pasado a lb/plg.

4.- La deflexión de una placa rectangular larga y uniformemente cargada, y que se encuentra bajo una fuerza de tensión axial, se rige por la ecuación diferencial de segundo orden. Siendo S la fuerza axial y q la intensidad de carga uniforme. La deflexión W a lo largo de la longitud elemental de esta está dada por:

$$\frac{d^2\omega}{dx^2} = \frac{S}{D}\omega + \frac{qx}{2D}(x-l),\ \ 0 < x < l,\ \ \omega(0) = \omega(l) = 0$$

Aproximar la flexión a intervalos de 1 plg.

Resolveremos el ejercicio por el mismo método que antes, para ello construimos las siguientes funciones en Matlab:

p.m

```
function y=p(t)
y=0;
```

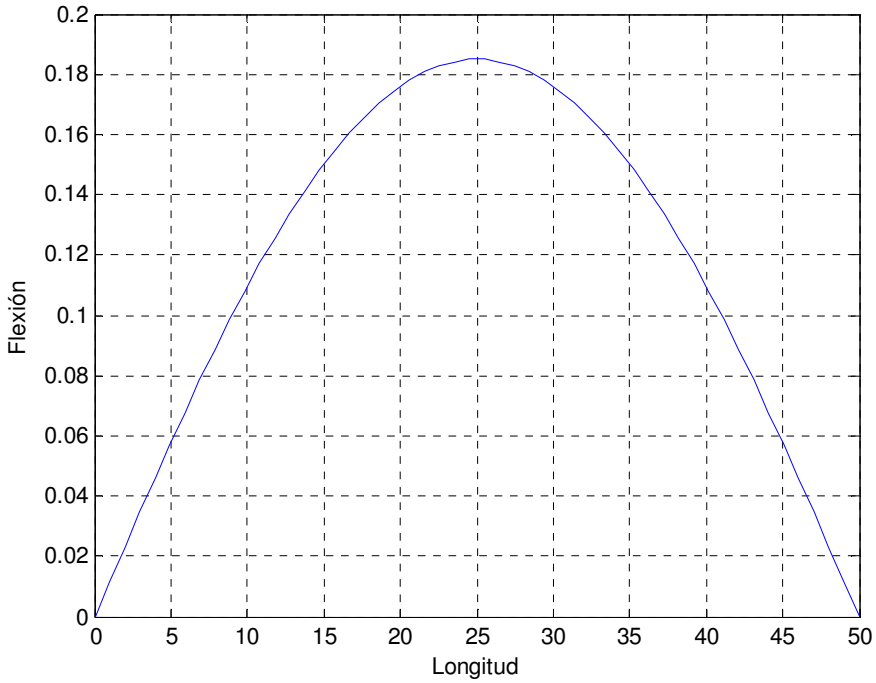
q.m

```
function y=q(t)
S=100;
D=8.8e+7;
y=S/D;
```

r.m

```
function y=r(t)
q=200;
l=50;
D=8.8e+7
y=((q*t)/(2*D)).*(t-l);
```

```
>> D=diferenciasfinitas('p','q','r',0,50,0,0,50);
>> plot(D(:,1),D(:,2))
>> grid on
>> xlabel('Longitud')
>> ylabel('Flexión')
```



5.- Compara el error cometido entre los algoritmos de Diferencias finitas y Disparo Lineal con la ecuación diferencial siguiente, sabiendo la solución exacta.

$$\frac{d^2y}{dx^2} = -\frac{2}{x}\frac{dy}{dx} + \frac{2}{x^2}y + \frac{\sin(\log x)}{x^2} \rightarrow \begin{cases} y(1) = 1 \\ y(2) = 2 \end{cases}$$

Solución Exacta: $y(x) = c_1 x - \frac{c_2}{x} - \frac{3}{10}\sin(\log x) - \frac{1}{10}\cos(\log x)$

Lo primero que debemos hacer es calcular las constantes de integración c_1 y c_2, para los valores de y(x) dados en x=1 y x=2, para ello usaré el comando *solve* de Matlab en el siguiente algoritmo, al que llamamos evalR2:

evalR2.m

```
S=solve('c1-c2-((3/10)*sin(ln(1)))-((1/10)*cos(ln(1)))-1=0','2*c1-(c2/2)-((3/10)*sin(ln(2)))-
((1/10)*cos(ln(2)))-2=0','c1,c2');
M=[eval(S.c1),eval(S.c2)]
```

```
>> evalR2
M =   1.14574151535366   0.04574151535366
```

Disparo Lineal:

p.m

```
function y=p(t)
y=(-2./t);
```

q.m

```
function y=q(t)
y=2./(t.^2);
```

r.m

```
function y=r(t)
y=(sin(log(x))./(t.^2);
```

```
function y=YE(T);
c1=1.14574151535366;
c2=0.04574151535366;
y=(c1.*T)-(c2./T)-((3/10)*sin(log(T)))-((1/10)*cos(log(T)))-1
```

ftysis1.m

```
function Z=ftysis1(t,Z)
u1=Z(1);
u2=Z(2);
Z=[u2,u2.*p(t)+u1.*q(t)+r(t)]';
```

ftysis2.m

```
function Z=ftysis2(t,Z)
v1=Z(1);
v2=Z(2);
Z=[v2,v2.*p(t)+v1.*q(t)]';
```

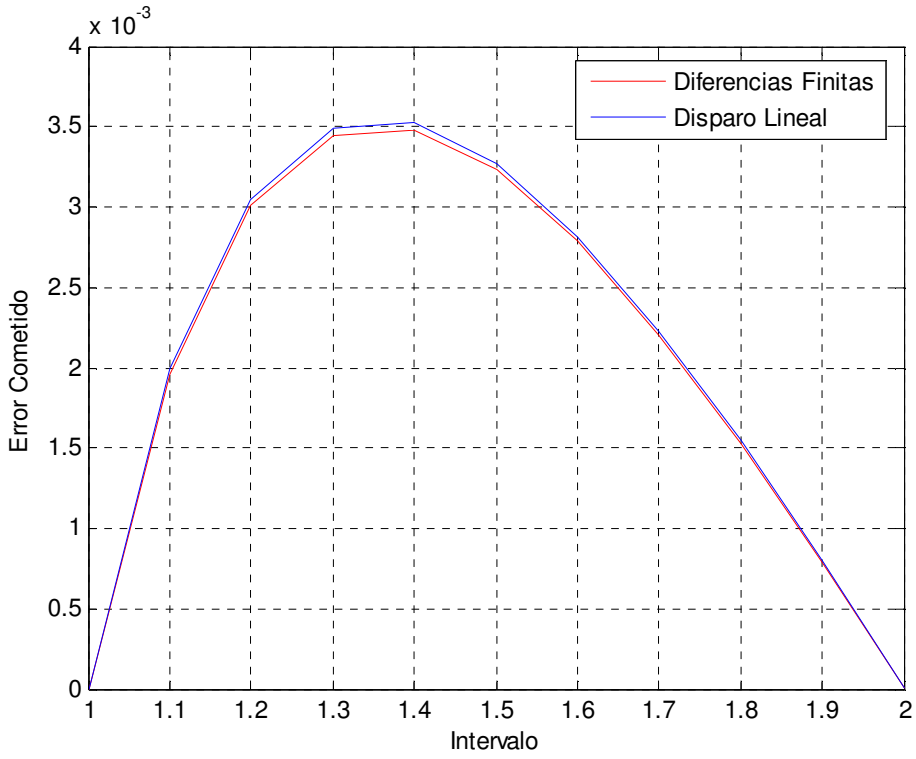
```
>> D=disparo('ftysis1','ftysis2',1,2,1,2,10)
```

x	y	error-y
1.00000000000000	1.00000000000000	0
1.10000000000000	1.09262922569844	0.00199268036538
1.20000000000000	1.18708479181925	0.00304938569248
1.30000000000000	1.28338233099811	0.00349147903769
1.40000000000000	1.38144592881779	0.00352060687397
1.50000000000000	1.48115940097066	0.00326723506754
1.60000000000000	1.58239244946667	0.00281799278118
1.70000000000000	1.68501395382835	0.00223167156306
1.80000000000000	1.78889852922954	0.00154891362602
1.90000000000000	1.89392950569297	0.00079825561400
2.00000000000000	2.00000000000000	0.00000000000000

Para el algoritmo de Diferencia Finitas usaré las mismas funciones, esta vez me arroja los siguientes resultados:

```
>> D=diferenciasfinitas('p','q','r',1,2,1,2,9)

        x                   y                errory
D =
   1.00000000000000   1.00000000000000            0
   1.10000000000000   1.09260052072014   0.00196397538708
   1.20000000000000   1.18704312879507   0.00300772266830
   1.30000000000000   1.28333687021440   0.00344601825398
   1.40000000000000   1.38140204622861   0.00347672428479
   1.50000000000000   1.48112026211222   0.00322809620909
   1.60000000000000   1.58235989569346   0.00278543900797
   1.70000000000000   1.68498901838456   0.00220673611927
   1.80000000000000   1.78888174619375   0.00153213059023
   1.90000000000000   1.89392109915713   0.00078984907816
   2.00000000000000   2.00000000000000   0.00000000000000
```

Comparativa de los errores en ambos métodos de cálculo:



Como podemos observar el método de Diferencias Finitas nos da mejor resultado.

6.- Por un tubo cilíndrico circula un fluido caliente. Debido a que la presión es muy alta, las paredes del tubo son gruesas. Para esta situación, la ecuación diferencial que relaciona las temperaturas de la pared metálica con la distancia radial es:

$$r\frac{d^2u}{dr^2}+\frac{du}{dr}=0$$

donde r es la distancia radial desde la línea central y u la temperatura.

Calcula u(1.67) para un tubo cuyo radio interior es 1 cm y cuyo radio exterior es de 2 cm, si el fluido está a 540°C y la temperatura de la circunferencia exterior está 20°C. Dibujar la gráfica de $\frac{d^2u}{dr^2}$ en el intervalo [1,2].

Haré el problema mediante el algoritmo de Disparo Lineal, y usaré las siguientes funciones:

p.m
```
function y=p(t)
y=-1/t;
```

q.m
```
function y=p(t)
y=-0;
```
r.m
```
function y=p(t)
y=-0;
```

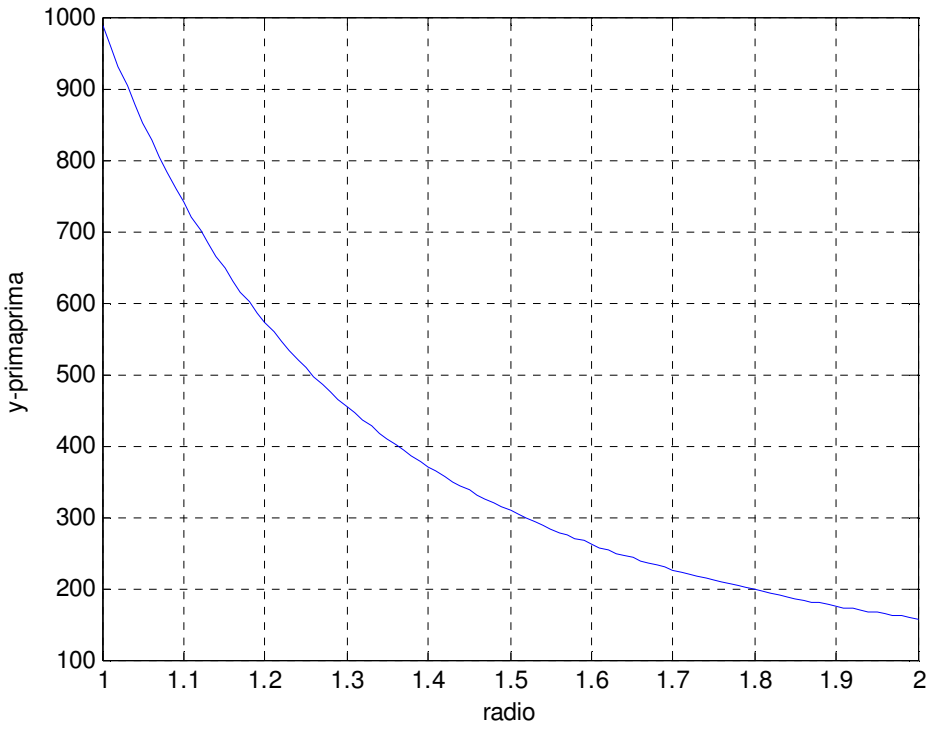
ftysis1.m
```
function Z=ftysis1(t,Z)
u1=Z(1);
u2=Z(2);
Z=[u2,u2.*p(t)+u1.*q(t)+r(t)]';
```

ftysis2.m
```
function Z=ftysis2(t,Z)
v1=Z(1);
v2=Z(2);
Z=[v2,v2.*p(t)+v1.*q(t)]';
```

```
>> D=disparo('ftysis1','ftysis2',1,2,540,20,100)
        x                       y                   y-primaprima
D =
  1.0e+002 *
   0.01000000000000   5.40000000000000   9.89304993812826
   0.01010000000000   5.30209229002920   9.59372790695850
.
```

```
.
   0.01660000000000   1.39325552536896   2.39116686882659
   0.01670000000000   1.35373111548493   2.35668965828764
   0.01680000000000   1.31453964907339   2.32306280534897
.
.
   0.01990000000000   0.23154874801147   1.58962791229486
   0.02000000000000   0.20000000000000   1.57322259823625
```

```
>> plot(D(:,1),D(:,3))
>> grid on
>> xlabel('radio')
>> ylabel('y-primaprima')
```



7.- Sea el problema de contorno:

$$y'' = -\left(\frac{4}{x}\right)y' - \left(\frac{2}{x^2}\right)y + 2\left(\frac{logx}{x^2}\right);\quad y(1) = \frac{1}{2};\quad y(2) = \log 2;\quad 1 \le x \le 2$$

con una solución única dada por:

$$y(x) = \frac{4}{x} - \frac{2}{x^2} + logx - \frac{3}{2}$$

Usar el método de las Diferencias Finitas para aproximar la solución y su derivada en x=1,55. ¿Cuál es el error cometido?

p.m

```
function y=p(t)
y=-(4./t);
```

q.m

```
function y=q(t)
y=-2./(t.^2);
```

r.m

```
function y=r(t)
y=2.*(log(t)./(t.^2));
```

YE.m

```
function y=YE(T);
y=(4./T)-(2./(T.^2))+log(T)-(3/2);
```

dYE.m

```
function y=dYE(T);
y=-(4./(T.^2))+(4./(T.^3))+(1./T);
```

```
D=diferenciasfinitas(p,q,r,a,b,alfa,beta,M)
>> D=diferenciasfinitas('p','q','r',1,2,0.5,log(2),19)
```

x	Y	errorY	Yprima	errorYprima
1.00000000000000	0.50000000000000	0	0.88656594209959	0.11343405790041
1.05000000000000	0.54432829710498	0.00007328032784	0.78895407413500	0.00934064146034
1.10000000000000	0.57889540741350	0.00011415322901	0.61541273339829	0.00684774466802
1.15000000000000	0.60586957044481	0.00013409309801	0.48011216053887	0.00505668260676
1.20000000000000	0.62690662346739	0.00014062222899	0.37412684212602	0.00375647175565
1.25000000000000	0.64328225465741	0.00013870334320	0.29080387527175	0.00280387527175
1.30000000000000	0.65598701099456	0.00013162226672	0.22513166311503	0.00210025665167
1.35000000000000	0.66579542096891	0.00012155554190	0.17329859623969	0.00157680586221
1.40000000000000	0.67331687061853	0.00010994011977	0.13238028312625	0.00118494784929
1.45000000000000	0.67903344928154	0.00009771686808	0.10011507814773	0.00089001766965
1.50000000000000	0.68332837843330	0.00008549254736	0.07474110346383	0.00066702938976
1.55000000000000	**0.68650755962792**	**0.00007364846784**	**0.05487664710324**	**0.00049780785649**

1.60000000000000	0.68881604314363	0.00006241389789	0.03943152033429
0.00036902033429			
1.65000000000000	0.69045071166135	0.00005191594354	0.02754075484071
0.00027081021540			
1.70000000000000	0.69157011862770	0.00004221358629	0.01851458473094
0.00019583854734			
1.75000000000000	0.69230217013445	0.00003332097453	0.01180041256470
0.00013860498452			
1.80000000000000	0.69275015988417	0.00002522337711	0.00695367281667
0.00009496225426			
1.85000000000000	0.69299753741611	0.00001788809944	0.00361535610401
0.00006176599089			
1.90000000000000	0.69311169549457	0.00001127192605	0.00149455863120
0.00003662015620			
1.95000000000000	0.69314699327923	0.00000532616051	0.00035485065375
0.00001769055327			
2.00000000000000	0.69314718055995	0.00000000000000	0.00000374561426
0.00000374561426			

5. PROBLEMAS DE ECUACIÓN DEL CALOR

Para la resolución de éstos problemas emplearemos los algoritmos necesarios, según el método solicitado por los problemas, éstos algoritmos contendrán los elementos necesarios para obtener las solicitudes más habituales en éste tipo de problemas.

Para ésta primera parte, partiremos de un algoritmo que converge en todos los casos, para unas condiciones de contorno determinadas (Crank-Nicholson). A medida que el problema nos vaya cambiando las condiciones de contorno, iremos indicando las líneas que debemos modificar para el correcto funcionamiento del mismo.

```
1.  % Crank-Nicolson tipo Dirichlet-Dirichlet
2.  function [W]=CNDD(c,L,m,N,T)
3.  h=L/m;
4.  k=T/N;
5.  la=c^2*k/h^2;
6.
7.  %Cálculo de A,B
8.  A=tridiagonal(1+la,-la/2,m-1);
9.  B=tridiagonal(1-la,la/2,m-1);
10.
11. %Cálculo de Wo
12. for i=1:m-1
13. x(i)=(i*h);
14. ejex= [(x(1)-h);x';L];% Variable eje x para poder utilizarla para graficas y calculo de
15. % error
16. W(i)=ff(i*h);
17. U(i)=u(x(i),T); % Función de la solución exacta.
18. end
19.
20. W=W';
21. U=U';
22. WW(1,:)=[aa(0);W;bb(0)]'; %1ª fila de la matriz temperaturas
23.
24. %Cálculo de W1,...,WN
25. for j=0:N-1
26. J(1,j+1)=j*k;
27. tiempo=[J';T]; % Variable tiempo para poder utilizarla para gráficas
28.
29. %Cálculo de bj
30. for i=1:m-1
31. b(i)=k*gg(i*h,(j+1/2)*k);
32. end
33. b(1)=b(1)+la/2*(aa((j+1)*k)+aa(j*k));
34. b(m-1)=b(m-1)+la/2*(bb((j+1)*k)+bb(j*k));
35.
36. W=A\(B*W+b');
```

```
WW(j+2,:)=[aa((j+1)*k);W;bb((j+1)*k)]';
Werror=WW(j+2,:);% ;Lo utilizo para el programa error.
end

Error=norm(U-W);
W=[aa(T);W;bb(T)];

%___________FUNCIÓN U, Solución exacta
UU=[u(0,T);U;u(L,T)];

%_____________________Cálculo del MAX valor__

MAX=WW(1,1);
x=0;
t=0;
ite=0;
pos=[1 1];
for i=1:N+1
    for j=1:m+1
        if MAX==WW(i,j)
        pos=[pos;[i j]];
        MAX=[MAX;WW(i,j)];
        x=[x;(i-1)*h];
        t=[t;(j-1)*k];
         end
      if MAX<WW(i,j)
         x=(j-1)*h;
         t=(i-1)*k;
         MAX=WW(i,j);
         pos=[i j];
       end
   end
end

Maximos=[MAX x t pos];

%_____________________Cálculo del MIN valor__

MIN=WW(1,1);
x=0;
t=0;
pos2=[1 1];
for i=1:N+1
   for j=1:m+1
      if MIN==WW(i,j)
        MIN=[MIN;WW(i,j)];
        pos2=[pos2;[i j]];
        x=[x;(i-1)*h];
        t=[t;(j-1)*k];
        end
       if MIN>WW(i,j)
        x=(j-1)*h;
        t=(i-1)*k;
```

```
            MIN=WW(i,j);
            pos2=[i j];
          end
      end
end
 Minimos=[MIN x t pos2];

%_____________________GRAFICA DE U (ejex, Tiempo final)__
figure
title ('Temperatura INCIAL-FINAL')
          hold on
          plot(ejex,WW(1,:),'g')
          hold on
          grid on
          hold on
          plot(ejex,WW(j+2,:)','b') % Solo la calculada
          hold on
          plot(ejex,UU,'r') % Solo real
          hold on
          legend('W0:Inicial','WW:Calculada Final','UU:Real Final')
          xlabel('x:longitud de la barra')
          ylabel('Temperatura')

          %_____________________HACEMOS MALLA__
          figure
          r=linspace(0,L,m+1);
          s=linspace(0,T,N+1);
          [rr,ss]=meshgrid(r,s);
          surf(rr,ss,WW);
          xlabel('Longitud')
          ylabel('Tiempo')
          zlabel('Temperaturas')

          %_____________________FIGURA DE X FIJO PARA TODO T __
          figure
          Title('Temperaturas x=1/2')
          hold on
          grid on
          hold on
          x=tiempo;
          y=WW(:,(m/2)+1);
          plot(x,y,'b')
          xlabel('Tiempo Transcurrido')
          ylabel('Temperatura')
```

Algoritmo para calcular un m y N acorde al error máximo que pida el problema.

```
function [W,norma,m,N]=CNDDerror(c,L,m,N,T,Tol)
cont=0;
[W,ejex,WW]=CNDD(c,L,m,N,T);
m=2*m; N=2*N;
cont=cont+1;
[W2,ejex,WW]=CNDD(c,L,m,N,T);
```

```
% Para poder compararlos me tengo que quedar con los impares.
Wcomparada=W2(1:2^cont:m+1);
while norm(W-Wcomparada)>Tol
W=Wcomparada;
m=2*m; N=N*2;
cont=cont+1;
cont;
[W2,ejex,WW]=CNDD(c,L,m,N,T);
Wcomparada=W2(1:2^cont:m+1);
end
norma=norm(W-Wcomparada);
m;
N;
```

Resolver los siguientes problemas relativos a la ecuación del calor, con un error menor a 10^{-4}, la solución exacta para todas es:

$$\boldsymbol{u(x,t) = \sin(x) + \cos(t)}$$

1. **Resolver:**

$$\boldsymbol{U_t = U_{xx} + \big(\sin(x) - sin(t)\big) \quad 0 \le x \le 2\pi;\ \ t \ge 0.}$$

$$\boldsymbol{u(0,t) = \cos(t)\,;\ \ u(2\pi,t) = \cos(t)\,;\ \ u(x,0) = 1 + \sin(x)}$$

Como vemos estas son condiciones Dirichlet-Dirichlet. Usaré el algoritmo de Cranck Nicholson.

Para hacerlo con un error menor al indicado, primero debo correr mi algoritmo CNDD.m, siempre pidiendo al algoritmo como datos de salida los que le hacen falta para funcionar al algoritmo de error:

```
[Werror,ejex,WW]=CNDD(c,L,m,N,T);
```

```
>> [Werror,ejex,WW]=CNDD(1,2*pi,10,15,1);
```

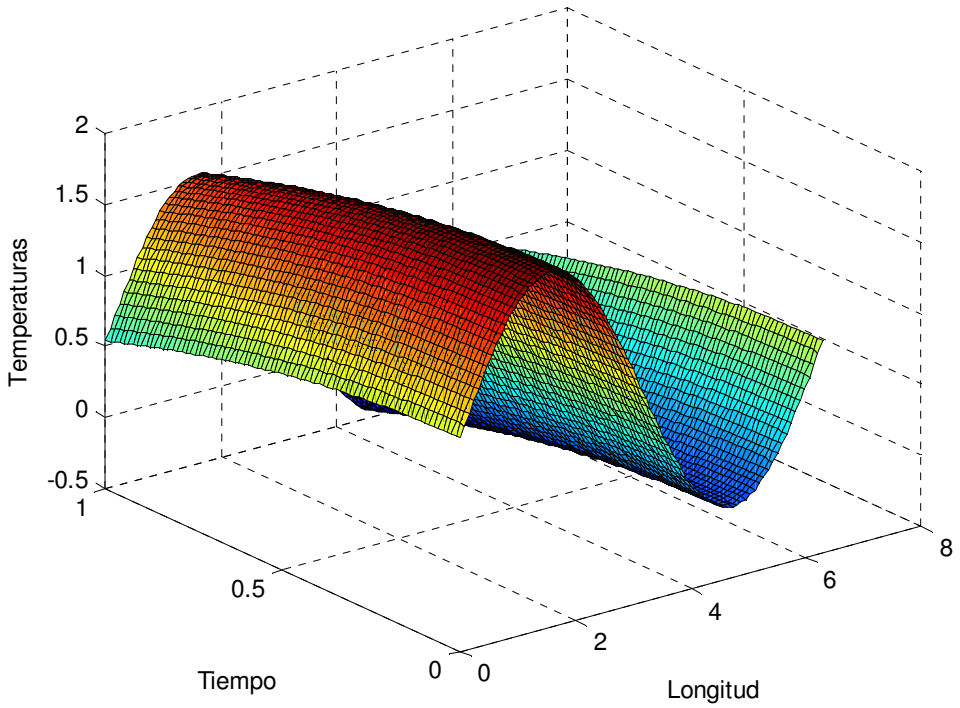
Acto seguido haré correr el algoritmo CNDDerror y este será el que me de los valores de m y N para resolver el problema con un error menor al del enunciado.

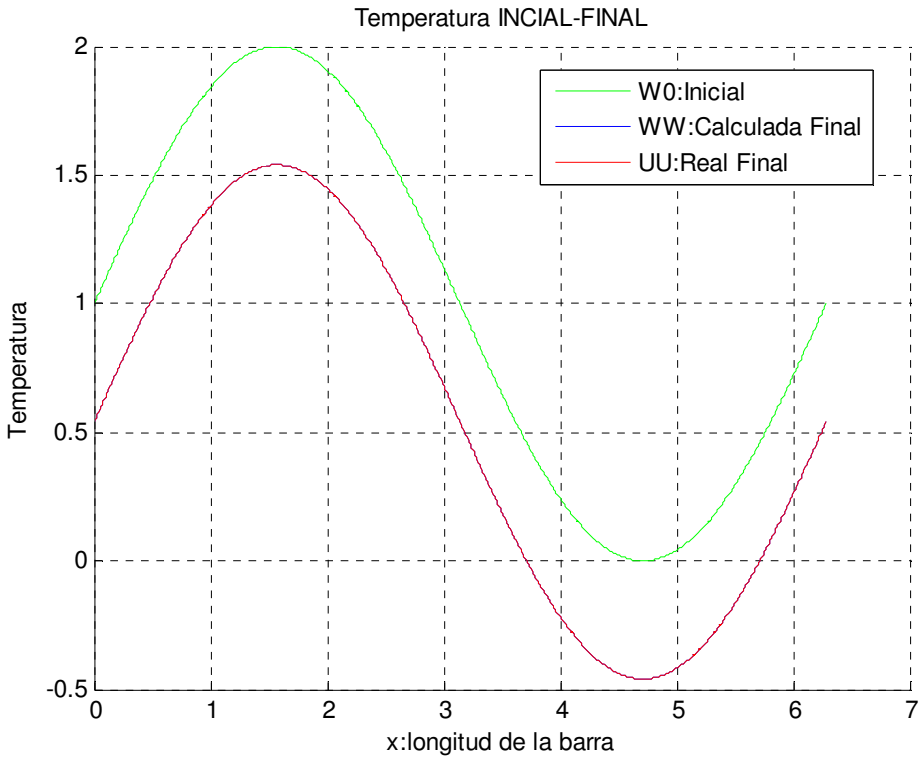
```
[W,norma,m,N]=CNDDerror(c,L,m,N,T,Tol)
>>  [W,norma,m,N]=CNDDerror(1,2*pi,10,15,1,0.0001)
m = 640
N = 960
```

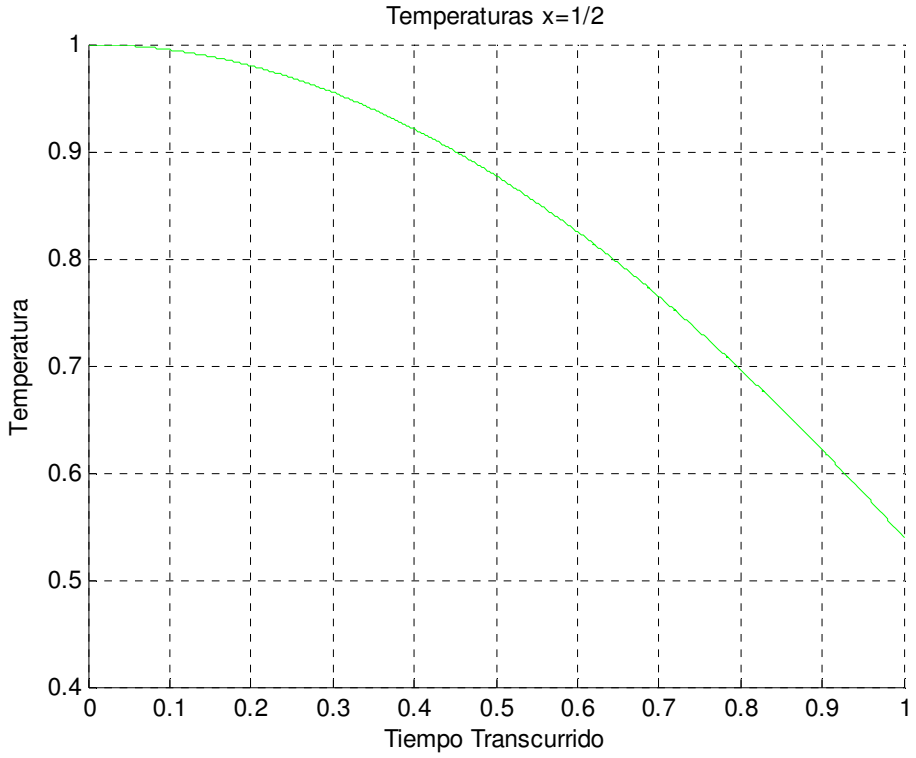
Ahora estoy en condiciones de poder resolver el problemas en las condiciones indicadas.

```
>> [Maximos,Minimos,Error]=CNDD(1,2*pi,640,960,1)
  Temperatura   Posicion  Tiempo Fila    Columna
```

```
Maximos =
   2.0000    1.5708         0    1.0000  161.0000
Minimos =
  -0.4597    4.7124    1.0000  961.0000  481.0000

Error =
 9.0824e-005
```




Temperatura INCIAL-FINAL
W0:Inicial
WW:Calculada Final
UU:Real Final
Temperatura
x:longitud de la barra
2
1.5
1
0.5
0
-0.5
0
1
2
3
4
5
6
7


Temperaturas x=1/2
Temperatura
Tiempo Transcurrido
1
0.9
0.8
0.7
0.6
0.5
0.4
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

2. Resolver:

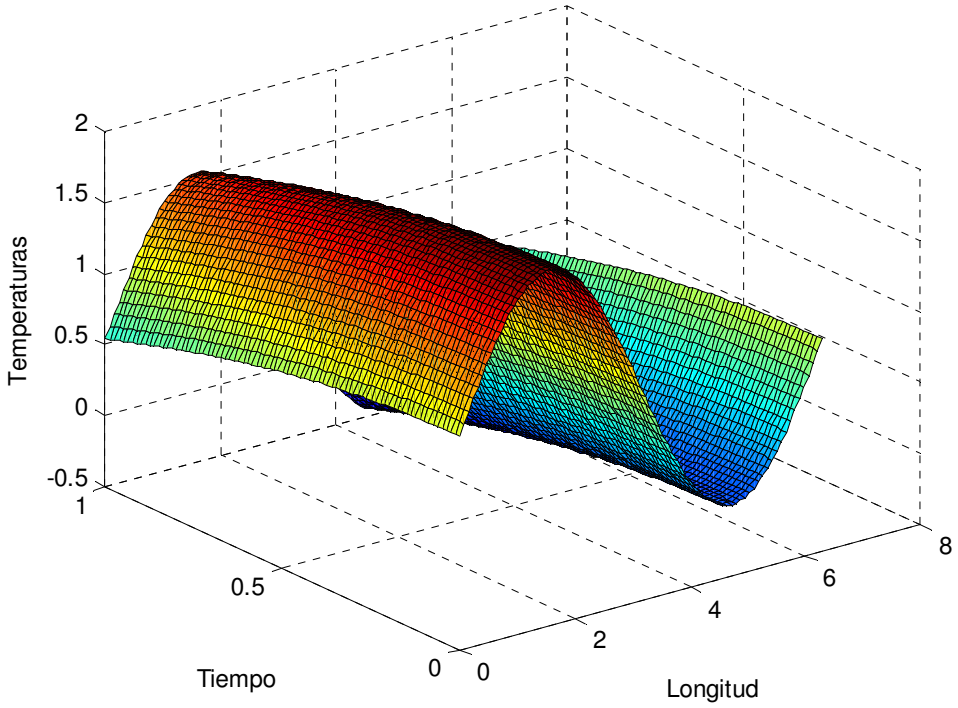
$$U_t = U_{xx} + \left(\mathbf{sin}(x) - sin(t)\right) \quad 0 \le x \le 2\pi;\ \ t \ge 0.$$
$$u(0,t) = \cos(t)\,;\ \ u_x(2\pi,t) = 1;\ \ u(x,0) = 1 + \sin(x)$$

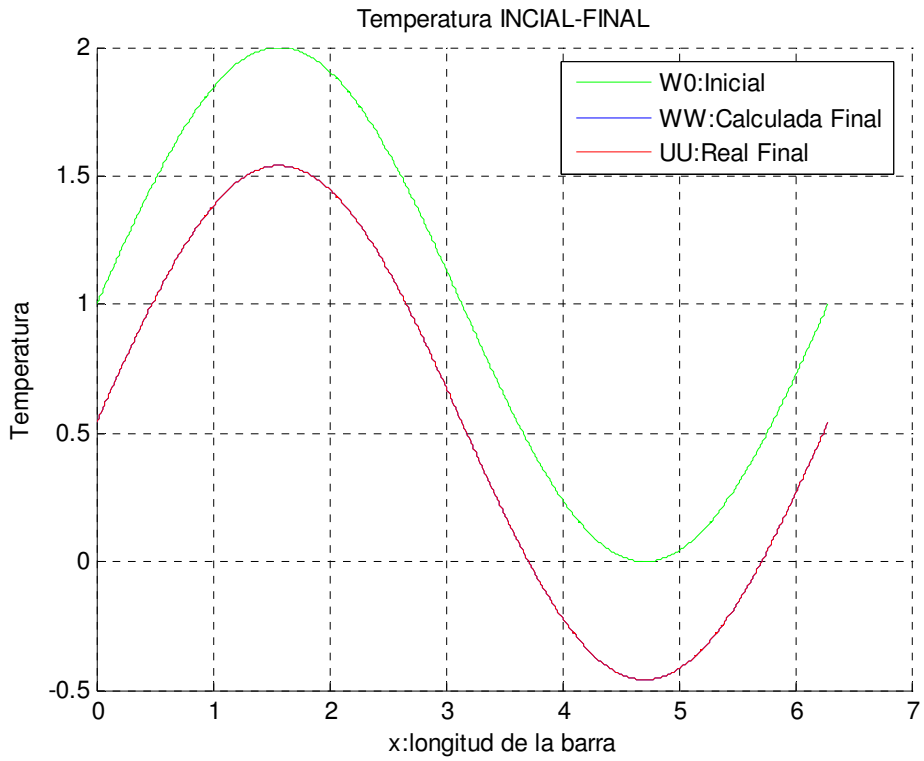
Como vemos estas son condiciones Dirichlet-Neumann. Usaré el algoritmo de Cranck Nicholson. Y actuaré de la misma forma que en el ejercicio anterior. Aunque antes debemos cambiar algunas líneas del algoritmo CNDD.m.

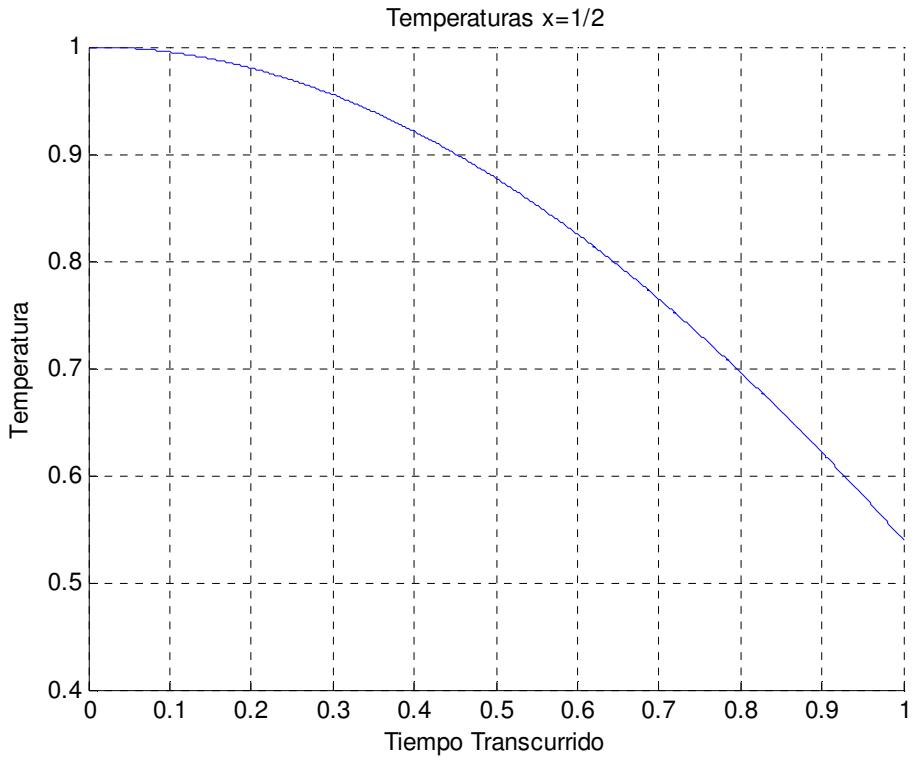
Cambio la línea	Por la siguiente
2	function [W]=CNDN(c,L,m,N,T)
22	WW(1,:)=[aa(0);W]';
37	WW(j+2,:)=[aa((j+1)*k);W]';
42	W=[aa(T);W];
45	UU=[u(0,T);U];

```
[W]=CNDN(c,L,m,N,T);
>> [Werror,ejex,WW]=CNDN(1,2*pi,10,15,1);
[norma,m,N]=CNDNerror(c,L,m,N,T,Tol)
>> [norma,m,N]=CNDNerror(1,2*pi,10,15,1,0.0001)
norma = 7.4477e-005
m = 640
N = 960
>> [Minimos,Maximos,Error]=CNDN(1,2*pi,640,960,1)
Temperatura     Posicion  Tiempo Fila    Columna
Minimos =
   -0.4597    4.7124    1.0000  961.0000  481.0000
Maximos =
    2.0000    1.5708         0    1.0000  161.0000
Error = 1.0744e-004
```

Carmelo Camacho González
Fernando J. Manobel Ponce







3. Resolver

$$U_t = U_{xx} + \left(\mathbf{sin}(x) - sin(t)\right) \quad 0 \le x \le 2\pi;\ \ t \ge 0.$$

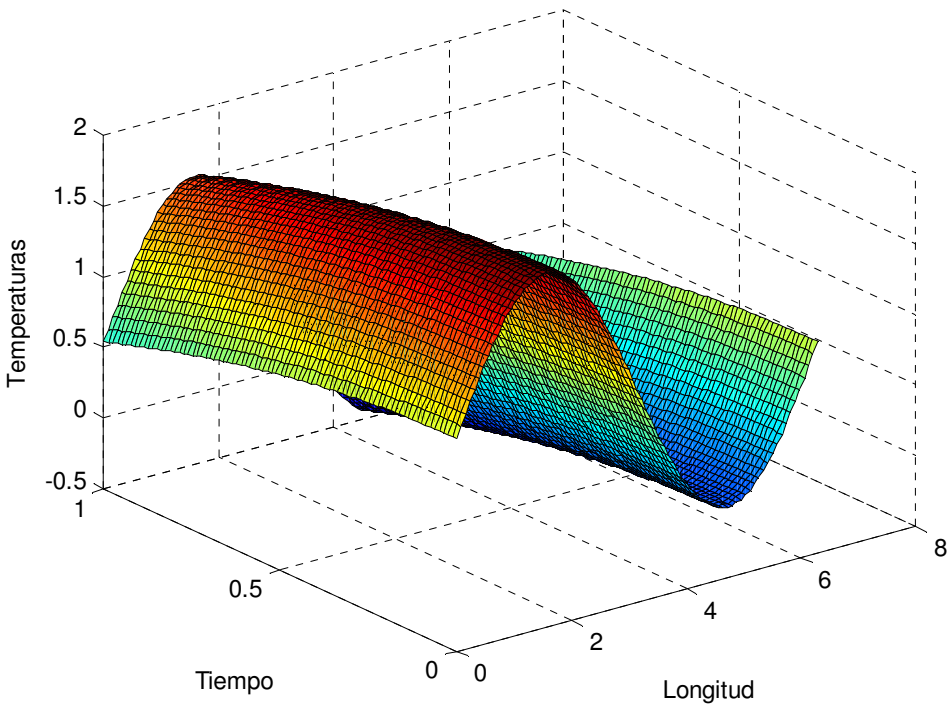
$$u_x(0,t) = 1;\ \ u_x(2\pi,t) = 1;\ \ u(x,0) = 1 + \mathbf{sin}(x)$$

Como vemos estas son condiciones Neumann-Neumann. Usaré el algoritmo de Cranck Nicholson. Y actuaré de la misma forma que en el ejercicio anterior. Aunque antes debemos volver a cambiar algunas líneas del algoritmo CNDD.m.

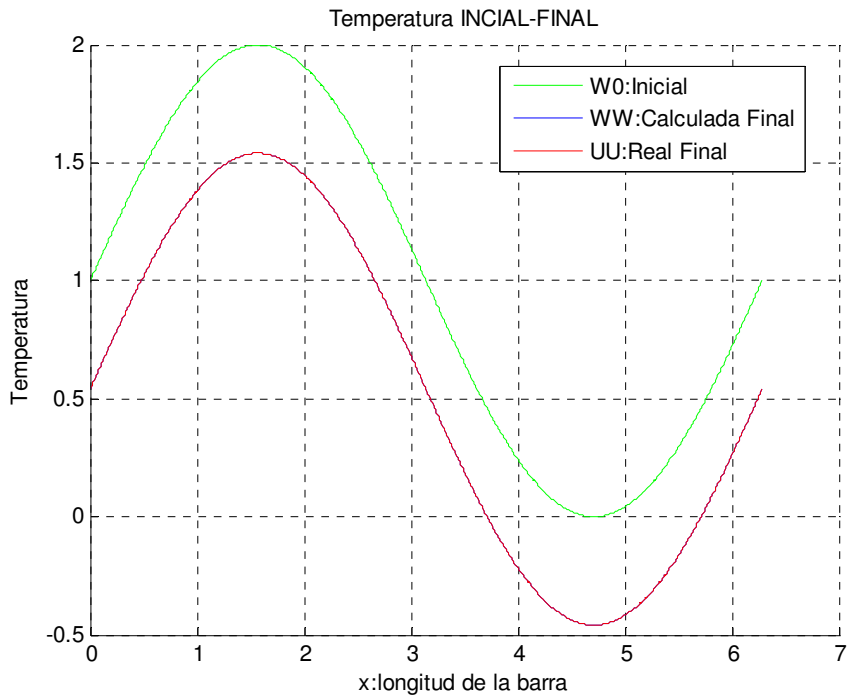
Cambio la línea	Por la siguiente
2	function [W]=CNNN(c,L,m,N,T)
22	WW(1,:)=[W]';
37	WW(j+2,:)=[W]';
42	W=[W];
45	UU=[U];

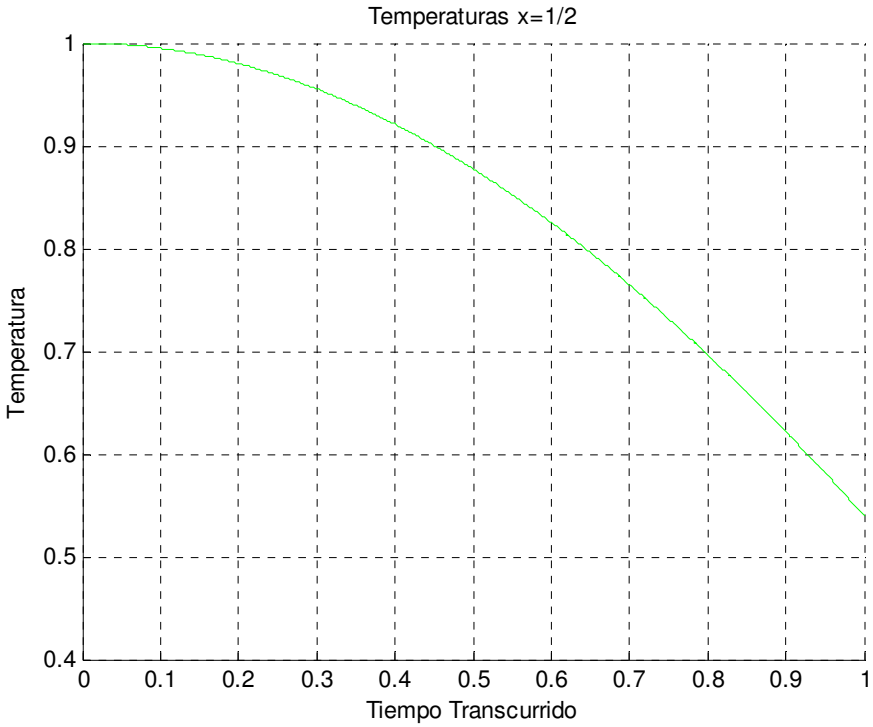
[W]=CNNN(c,L,m,N,T);
>> [Werror,ejex,WW]=CNNN(1,2*pi,10,15,1);
[norma,m,N]=CNNNerror(c,L,m,N,T,Tol)
>> [norma,m,N]=CNNNerror(1,2*pi,10,15,1,0.0001)

```
norma = 6.5115e-005
m = 640
N = 960
>> [Minimos,Maximos,Error]=CNNN(1,2*pi,640,960,1)
  Temperatura  Posicion  Tiempo  Fila Columna
Minimos =
   -0.4597    4.7124    1.0000  961.0000  481.0000
Maximos =
    2.0000    1.5708         0    1.0000  161.0000

Error =
 1.2214e-004
```



Carmelo Camacho González
Fernando J. Manobel Ponce





CONCLUSIONES:

Se puede observar que con los tres tipos de condiciones los resultados son los mismos en cuanto a máximos y mínimos, e incluso el error. El más aproximado a la solución exacta es dado en condiciones D-D, en condiones N-N y D-N, no distan casi nada del resultado obtenido en el mejor de los casos.

4.- Compara los métodos Progresivo, Regresivo y de Cranck Nicholson, para la resolución del siguiente problema:

$$U_t = U_{xx} + e^x(-\cos(t) - sin(t)) \quad 0 \le x \le 1; \ t \ge 0.$$

$$u_x(0,t) = \cos(t); \ u_x(1,t) = e * \cos(t); \ u(x,0) = e^x$$

Cuya solución real es:

$$u(x,t)e^x\cos(t)$$

Método Progresivo, se aclara que para el método progresivo, he forzado a que λ sea 1/6, ya que esta es la que nos dará mejor aproximación.

Algoritmos usados:

MÉTODO PROGRESIVO:

```
function [W,U]= ProgresivoNN(c,L,m,T)

la=(1/6); % Es el óptimo;
h=L/m;
K= (la*h^2)/(c^2);
N=((c^2)*T)/(la*(h^2));

A=tridiagonal(1,0,m+1);
B=tridiagonal(1-2*la,la,m+1);
B(1,2)=2*la;
B(m+1,m)=2*la;

% Calculo de Wo
for i=0:m
W(i+1)=f(i*h);
end
W=W';
Wo=W;
WW(1,:)=[W]';
for i=0:m
x(i+1)=i*h;
U(i+1)=u(x(i+1),T);
end

% Calculo de W1.....Wn
for j=0:N-1
J(1,j+1)=j*K;
for i=0:m
b(i+1)=K*gg(i*h,(j+1)*K);
end

```

```
b(1)=b(1)-2*la*h*(a(j+1)*K);
b(m+1)=b(m+1)+2*la*h*(bb(j+1)*K);

%W=A\(B*W+b')
W=B*W+b';

%ALMACENAMOS TODA LA MATRIZ ENTERA
WW(j+2,:)=[W]';
end
U=U';
W=W;
```

MÉTODO REGRESIVO:

```
function [W,U]=implicitoneumann(c,L,m,N,T)

h=L/m;
k=T/N;
la=c^2*k/h^2;

%Cálculo de A,B
A=tridiagonal(1+2*la,-la,m+1);
A(m+1,m)=-(2*la);
A(1,2)=-(2*la);
B=tridiagonal(1,0,m+1);  %B es la matriz identidad

%Cálculo de Wo
for i=0:m
x(i+1)=i*h;
ejex=[x'];
W(i+1)=ff(i*h); %ff es la condición inicial --> u(x,0)=f(x)
U(i+1)=u(x(i+1),T);
end
W=W';
U=U';
%para añadir la matriz
WW(1,:)=[W]';
%Cálculo de W1,...,WN
for j=0:N-1
J(1,j+1)=j*k;
tiempo=[J';T];
%Cálculo de bj
for i=0:m
b(i+1)=k*gg(i*h,(j+1)*k);
end
b(1)=b(1)-2*la*h*aa((j+1)*k);
b(m+1)=b(m+1)+2*la*h*bb((j+1)*k);

W=A\(B*W+b');
%para añadir la matriz
WW(j+2,:)=[W]';
```

```
38. Werror=WW(j+2,:);
39. end
40. Error=norm(U'-Werror);
41. Error=max(abs(U'-Werror));
42. end
43. W=[W];
```

MÉTODO CrancK-Nicholson:

Emplearemos el algoritmo utilizado en el apartado anterior, con las condiciones de contorno Newman-Newman (El mismo que en el ejercicio 3).

Empezamos creando los archivos de las funciones de contorno en Matlab, que serán las mismas para todos los algoritmos:

aa.m

```
function y=aa(t)
y=cos(t);
```

bb.m

```
function y=bb(t)
y=(exp(1)).*cos(t);
```

ff.m

```
function y=ff(x)
y=exp(x);
```

gg.m

```
function y=gg(x,t)
y=(exp(x)).*(-cos(t)-sin(t));
```

u.m

```
function y=u(x,t)
y=(exp(x)).*cos(t);
```

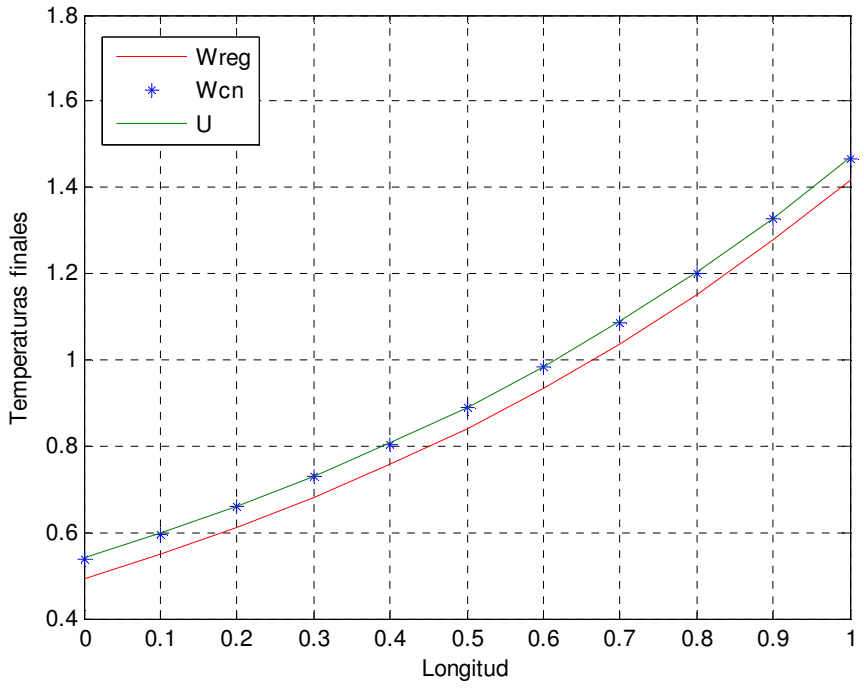
A continuación resolvemos el problema con todos los métodos definidos, y creamos una tabla donde podemos observar los errores obtenidos con cada uno de ellos, en comparación con los resultados reales.

```
[W]= ProgresivoNN(c,L,m,T)
>> [W]= ProgresivoNN(1,1,10,1);
>> Wprog=W;

[W]=implicitoneumann(c,L,m,N,T)
>> [W]=implicitoneumann(1,1,10,15,1);
```

```
>> Wreg=W;


[W]=CNNN(c,L,m,N,T)
>> [W,U]=CNNN(1,1,10,15,1);
>> Wcn=W;

>> x=0:0.1:1;
>> U=u(x,1)';
>> S=[Wprog Wreg Wcn U]

S =
  -0.41759319915346   0.49341305887012   0.53895774412540   0.54030230586814
  -0.42261891454269   0.55006068844557   0.59567335226944   0.59712639541468
  -0.43622375065815   0.61253325373625   0.65834968624230   0.65992672662765
  -0.45681167175231   0.68148432852462   0.72760489763211   0.72933182632973
  -0.48257317273517   0.75763715673168   0.80413907118558   0.80603632408662
  -0.51151935894887   0.84179155312575   0.88872213067204   0.89080790429313
  -0.54145213867422   0.93483153573551   0.98220420413156   0.98449498941668
  -0.56994439910080   1.03773376179442   1.08552442003048   1.08803523129451
  -0.59431544929493   1.15157684658818   1.19972218658220   1.20246489546840
  -0.61157521804271   1.27755165301286   1.32595554228438   1.32892923247855
  -0.61853515977663   1.41697264906763   1.46546084646318   1.46869393991589
```

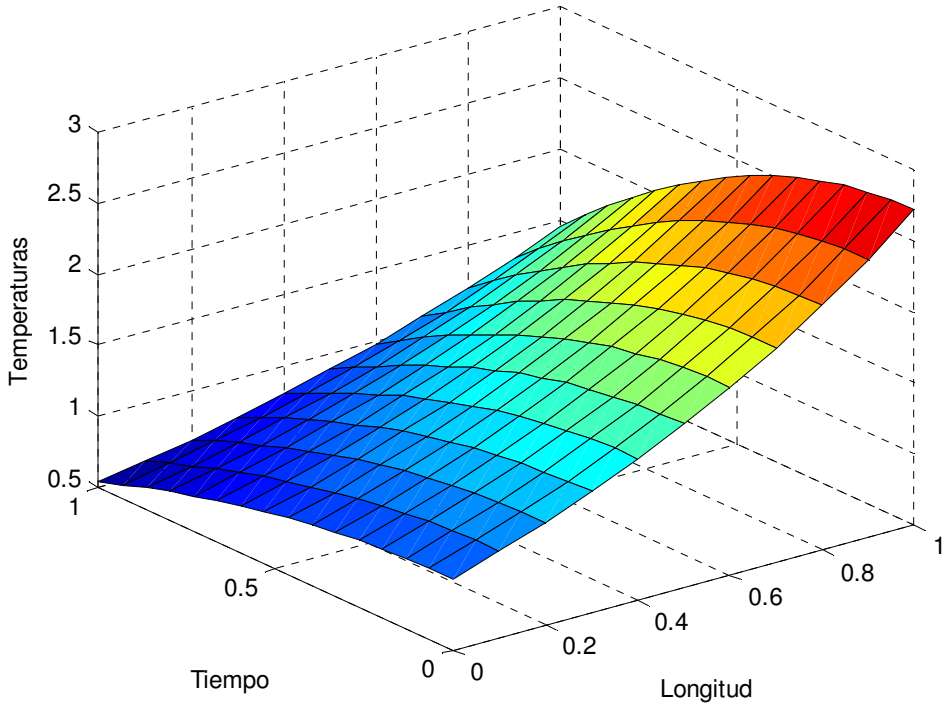
Como vemos el Método que más se aproxima a la solución real es el de Crank Nicholson, De todas formas, a continuación se representa una grafica explicativa, dejando fuera de la gráfica el resultado que nos proporciona el método progresivo, ya que es evidente que está muy lejos de la solución exacta:

```
>> plot(x,S(:,2),'r',x,S(:,3),'*',x,S(:,4))
>> grid on
>> legend('Wreg','Wcn','U','location','best')
>> xlabel('Longitud')
>> ylabel('Temperaturas finales')
```



```
>> Errores=[abs(Wprog-U) abs(Wreg-U) abs(Wcn-U)]
Errores =
  0.95789550502160   0.04688924699802   0.00134456174274
  1.01974530995738   0.04706570696911   0.00145304314524
  1.09615047728579   0.04739347289139   0.00157704038535
  1.18614349808204   0.04784749780511   0.00172692869762
  1.28860949682178   0.04839916735494   0.00189725290104
  1.40232726324200   0.04901635116738   0.00208577362109
  1.52594712809090   0.04966345368117   0.00229078528512
  1.65797963039531   0.05030146950009   0.00251081126402
  1.79678034476333   0.05088804888022   0.00274270888620
  1.94050445052126   0.05137757946569   0.00297369019417
  2.08722909969252   0.05172129084826   0.00323309345270
```

Lógicamente podríamos ajustar los parámetros m y N para afinar mucho más en el error, pero el ejercicio solo pretendía poder hacer una comparativa de cual es el mejor método a priori.



5.- Aproxima la función que se plantea a continuación mediante los métodos Regresivo y de Cranck Nicholson.

$$U_t = 0.86U_{xx}\,; \quad 0 < x < 2\,; \;\; 0 < t < 1.$$

$$u(0,t) = 0\,; \;\; u(2,t) = 0$$

$$u(x,0) = \begin{cases} \frac{4}{3}x \quad si\; 0 \le x \le 0.75; \\ 1 \;\; si \;\; 0.75 \le x \le 1.25; \\ \frac{4}{3}(2-x) \;\; si \;\; 1.25 \le x \le 2; \end{cases}$$

Para el método de Cranck_Nicholson, usaré el algoritmo del ejercicio 1, para el método regresivo usaré el del ejercicio anterior con los siguientes cambios:

Cambio la línea	Por la siguiente
1	function [W]=implicitodirilech(c,L,m,N,T)
23	WW(1,:)=[aa(0);W;bb(0)]';
37	WW(j+2,:)=[aa((j+1)*k);W;bb((j+1)*k)]';
43	W=[aa(T);W;bb(T)];

Primero crearé las funciones de las condiciones de contorno para los dos algoritmos que voy a usar:

aa.m

```
function y=aa(t)
y=0;
```

bb.m

```
function y=bb(t)
y=0;
```

ff.m

```
function y=ff(x)
if x>=0
   if x<=0.75
      y=(4/3)*x;
   end
   if x>=0.75
      if x<=1.25
         y=1;
      end
      if x>=1.25
         if x<=2
```

```
            (4/3)*(2-x);
          end
        end
    end
end
```

gg.m

```
function y=gg(x,t)
y=0;
```

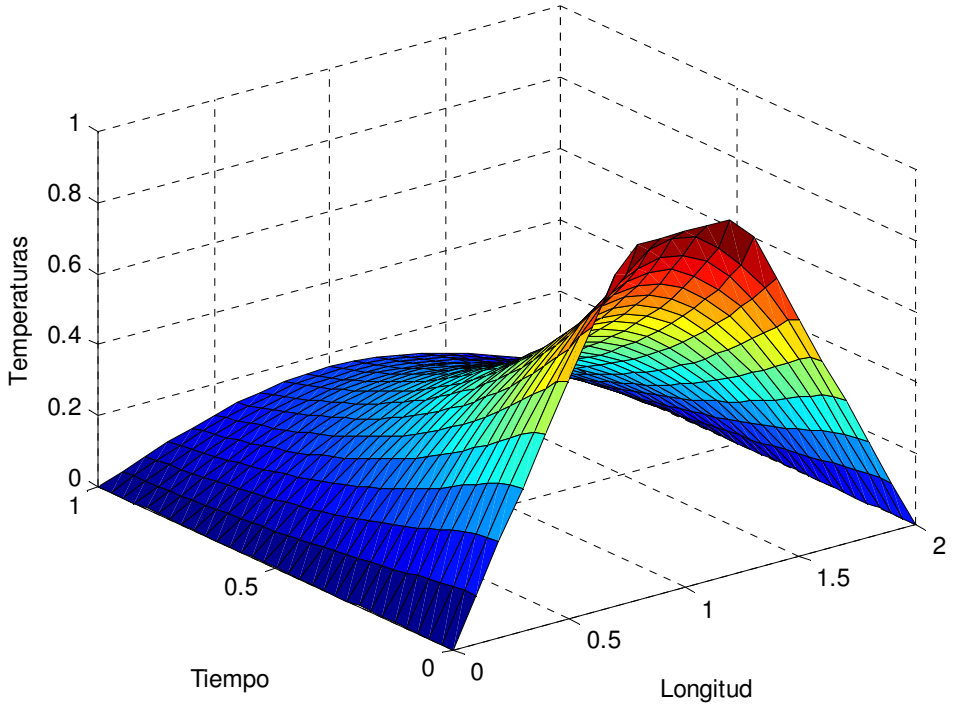
A continuación se procede con el Matlab:

```
[W]=implicitodirichlet(c,L,m,N,T)
>> [W]=implicitodirichlet(0.86,2,20,30,1)
W =
                0
   0.02663137452290
   0.05260700617317
   0.07728729647446
   0.10006453868728
   0.12037788071193
   0.13772713530367
   0.15168509750847
   0.16190806576918
   0.16814430722291
   0.17024025826083
   0.16814430722291
   0.16190806576918
   0.15168509750847
   0.13772713530367
   0.12037788071193
   0.10006453868728
   0.07728729647446
   0.05260700617317
   0.02663137452290
                0
>> Wreg=W;

>> Wreg=W;

[W]=CNDD(c,L,m,N,T)
>> [W]=CNDD(0.86,2,20,30,1)
W =
                0
   0.02523914911427
   0.04985682499499
   0.07324686515416
   0.09483331783711
   0.11408466940996
   0.13052686889704
   0.14375506958389
   0.15344354461239
   0.15935371996763
   0.16134009453265
   0.15935371996763
   0.15344354461239
```

```
   0.14375506958389
   0.13052686889704
   0.11408466940996
   0.09483331783711
   0.07324686515416
   0.04985682499499
   0.02523914911427
                 0

>> Wcn=W;

>> Diferencias=abs(Wreg-Wcn)
Diferencias =
                 0
   0.00139222540863
   0.00275018117818
   0.00404043132031
   0.00523122085017
   0.00629321130197
   0.00720026640663
   0.00793002792458
   0.00846452115680
   0.00879058725528
   0.00890016372818
   0.00879058725528
   0.00846452115680
   0.00793002792458
   0.00720026640663
   0.00629321130197
   0.00523122085017
   0.00404043132031
   0.00275018117818
   0.00139222540863
                 0
```

Nuevamente la diferencia entre un método y otro es muy pequeña, como podemos observar.

Carmelo Camacho González
Fernando J. Manobel Ponce



6. PROBLEMAS DE ECUACIÓN DE ONDAS

Este es el algoritmo que usaremos para resolver los problemas planteados de ondas, el que aquí se expone es para condiciones Direlech-Direlech, el cual se modificará cuando sea necesario.

```
1.  function [W]=ondasDD(c,L,m,N,T)
2.  h=L/m;
3.  k=T/N;
4.  la=(c*k)/h;
5.
6.  %Cálculo de la matriz A
7.  A=tridiagonal(2*(1-la^2),la^2,m-1);
8.
9.  %Cálculo de W0 y W1
10. for i=1:m-1
11.
12. W0(i)=ff(i*h); %ff es la función de condición inicial.
13. W1(i)=((1-la^2)*ff(i*h)+(la^2/2)*(ff((i+1)*h)+ff((i-1)*h))+k*v(i*h)+(k^2/2)*gg(i*h,0));v    %gg
    es la function de impulso.
14. end
15.
16. W0=W0';
17. MW(1,:)=[aa(0);W0;bb(0)]';
18. W1=W1';
19. MW(2,:)=[aa(k);W1;bb(k)]';
20.
21. %Cálculo de W2,W3..., WN
22. for j=1:N-1
23. J(1,j+1)=j*k; %da una matriz indicando los periodos de tiempo
24.
25. %calculo de bj
26. for i=1:m-1
27. b(i)=k^2*gg(i*h,j*k);
28. end
29.
30. b(1)=b(1)+la^2*aa(j*k);
31. b(m-1)=b(m-1)+la^2*bb(j*k);
32.
33. W=A*W1-W0+b';
34. MW(j+2,:)=[aa((j+1)*k);W;bb((j+1)*k)]';
35. W0=W1;
36. W1=W;
37. W=[aa(T);W;bb(T)];
38. end
39.
40. % Solución Exacta.
41. for i=1:m+1
42. x(i)=(i-1)*h;
43. U(i)=u(x(i),T);
```

```
44. end
45.
46. %_____Eje x para usarlo en graficas
47. ejex=x';
48.
49. % Matriz WWf de Posiciones y tiempos, mediante adicción de una fila y columna más.
50. WWf=[x;MW];
51. J=J';
52. J;
53. Y=[0;J;T];
54. Y;
55. WWf=[Y WWf];
56.
57. %_____Variable tiempo para poder hacer graficas_____
58. tiempo=[J;T];
59.
60. % Incluyendo solución exacta que proporciona de datos, incluye los extremos
61. for i=1:N+1
62. for j=1:m+1
63. MU(i,j)=u((j-1)*h,(i-1)*k);
64. end
65. end
66.
67. error=norm(MU-MW);
68.
69. %_____DIBUJA LA P INICIAL, FINAL CALCULADA Y REAL__
70. figure
71. %plot(ejex,UU,'g');
72. hold on
73. plot(ejex,WW(1,:),'r',ejex,W,'b')
74. grid
75. legend('P final real','P Inicial','P final calculada','location','best')
76. xlabel('x:longitud de la barra')
77. ylabel('Temperatura')
78.
79. %_____FIGURA DE X FIJO PARA TODO T __
80. figure
81. hold on
82. Title('Temperaturas')
83. hold on
84. grid on
85. hold on
86. x=tiempo;
87. y1=WW(:,1);
88. y2= WW(:,(m/2)+1);
89. y3=WW(:,m+1);
90. plot(x,y1,'b',x,y2,'g',x,y3,'r')
91. hold on
92. legend('T(0)','T(L/2)','T(L)','Location','Best')
93. xlabel('Tiempo Transcurrido')
94. ylabel('Temperatura')
95.
```

```
%_____________________HACEMOS MALLA__
figure
r=linspace(0,L,m+1);
s=linspace(0,T,N+1);
        [rr,ss]=meshgrid(r,s);
        surf(rr,ss,WW);
        xlabel('Longitud')
        ylabel('Tiempo')
        zlabel('Temperaturas')

        %_____________________Cálculo del MAX valor__

        MAX=MW(1,1);
        x=0;
        t=0;
        ite=0;
        pos=[1 1];
        for i=1:N+1
        for j=1:m+1
        if MAX==MW(i,j)
        pos=[pos;[i j]];
        MAX=[MAX;MW(i,j)];
        x=[x;(i-1)*h];
        t=[t;(j-1)*k];
        end
        if MAX<MW(i,j)
        x=(j-1)*h;
        t=(i-1)*k;
        MAX=MW(i,j);
        pos=[i j];
        end
        end
        end
        Maximos=[MAX x t pos];

        %_____________________Cálculo del MIN valor__

        MIN=MW(1,1);
        x=0;
        t=0;
        pos2=[1 1];
        for i=1:N+1
        for j=1:m+1
        if MIN==MW(i,j)
        MIN=[MIN;MW(i,j)];
        pos2=[pos2;[i j]];
        x=[x;(i-1)*h];
        t=[t;(j-1)*k];
        end
        if MIN>MW(i,j)
        x=(j-1)*h;
        t=(i-1)*k;
        MIN=MW(i,j);
```

```
149.        pos2=[i j];
150.        end
151.        end
152.        end
153.        Minimos=[MIN x t pos2];
```

Algoritmo de Error.

```
1.  function [W,norma,m,N]=ondaserrorDD(c,L,m,N,T,Tol)
2.  cont=0;
3.  [W,WWf,MW,ejex]=ondasDD(c,L,m,N,T);
4.  m=2*m; N=2*N;
5.  cont=cont+1;
6.  [W2,WWf,MW,ejex]=ondasDD(c,L,m,N,T);
7.  % Para poder compararlos me tengo que quedar con los impares.
8.  Wcomparada=W2(1:2^cont:m+1);
9.  while norm(W-Wcomparada)>Tol
10. W=Wcomparada;
11. m=2*m; N=N*2;
12. cont=cont+1;
13. cont;
14. [W2,MW,WWf,ejex]=ondasDD(c,L,m,N,T);
15. Wcomparada=W2(1:2^cont:m+1);
16. end
17. norma=norm(W-Wcomparada);
18. m;
19. N;
```

1.- Resolver el siguiente planteamiento:

$$U_{tt} = U_{xx} - 6xt$$

Con las siguientes condiciones de contorno:

$$u(0,t) = 0 \ ; \ u(1,t) = \sin(1)\sin(t) + t$$

Y las siguientes condiciones iniciales:

$$u(x,0) = 0 \ ; \ u_t(x,0) = \sin(x) + x^3$$

Siendo la solución exacta:

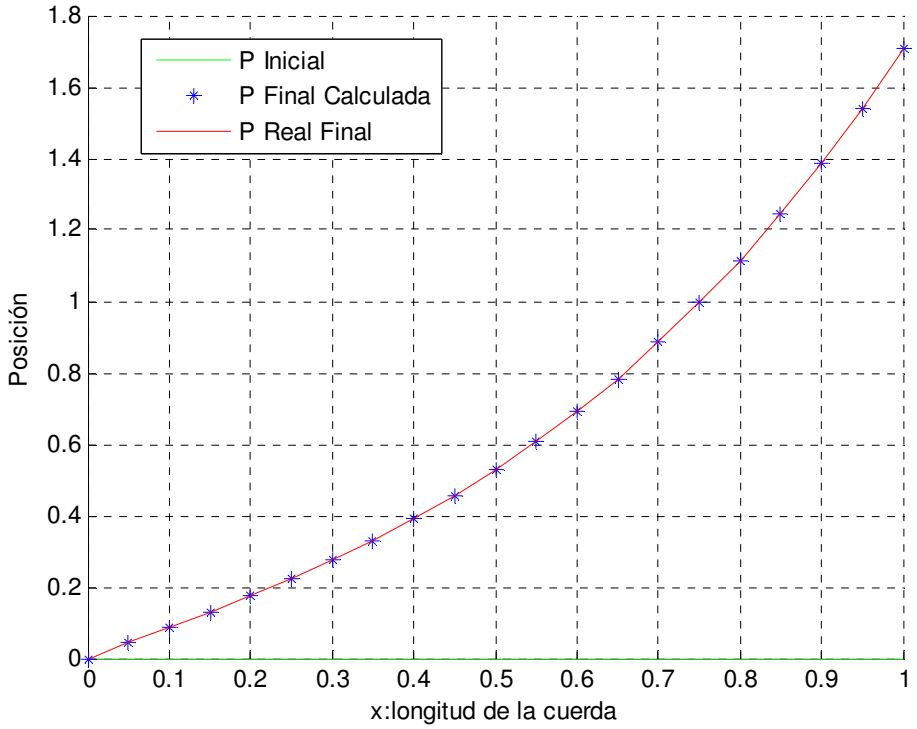
$$u(x,t) = \sin(x)\sin(t) + x^3 t$$

Crea la una matriz cuyas columnas correspondan a la solución calculada, la exacta y la diferencia entre las dos, para t=0,5. Además de la representación gráfica de la posición inicial, final calculada y la final real. Todo ello con un error menor a 10^{-3}.

Antes de empezar debo calcular un *"m"* y un *"N"* apropiados para que los resultados tengan la tolerancia exigida.

Para ello primero debo dar al algoritmo de error unos datos de partida, son los siguientes:

```
[W,WWf,MW,ejex]=ondasDD(c,L,m,N,T)
>> [W,WWf,MW,ejex]=ondasDD(1,1,10,15,1);
[W,norma,m,N]=ondaserrorDD(c,L,m,N,T,Tol);
>> [W,norma,m,N]=ondaserrorDD(1,1,10,15,1,1e-3)
>>m
m = 20
>>N
N = 30
[W]=ondasDD(c,L,m,N,T)
>> [W,MW,ejex]=ondasDD(1,1,20,30,1);
N=30;
>> uCalculada=MW(((N/2)+1),:)';
>> x=ejex;
>> uReal=feval('u',x,0.5);
>> Diferencia=(abs(uReal-uCalculada));
>> Solucion=[uCalculada uReal Diferencia]
Solucion =
            0                   0                   0
   0.02402834504062   0.02402379014659   0.00000455489403
   0.04837178794980   0.04836268954660   0.00000909840320
   0.07334557632026   0.07333195714916   0.00001361917110
   0.09926525681872   0.09924715092056   0.00001810589816
   0.12644682378832   0.12642427641841   0.00002254736991
   0.15520686667106   0.15517993424704   0.00002693242402
   0.18586271405344   0.18583146602553   0.00003124802791
   0.21873255258386   0.21869709850368   0.00003545408018
   0.25413540502802   0.25409608546554   0.00003931956248
   0.29239075983300   0.29234884706593   0.00004191276707
   0.33381839819057   0.33377710625162   0.00004129193895
   0.37874060767125   0.37870402192622   0.00003658574503
   0.42748540490983   0.42745431852594   0.00003108638388
   0.48038212422093   0.48035441168228   0.00002771253865
   0.53775609985173   0.53773252965769   0.00002357019404
   0.59993744647952   0.59991883025051   0.00001861622901
   0.66726004340337   0.66724551287717   0.00001453052620
   0.74005651662443   0.74004692555132   0.00000959107311
   0.81866460286362   0.81865966649266   0.00000493637096
   0.90342268011133   0.90342268011133                   0

>> [W,MW,ejex,tiempo,UU]=ondasDD(1,1,20,30,1);
>>figure
>>plot(ejex,MW(1,:),'g',ejex,MW(j+2,:),'*',ejex,UU,'r')
>>hold on
>>legend('P Inicial','P Final Calculada','P Real Final','location','best')
>>xlabel('x:longitud de la cuerda')
>>ylabel('Posición')
>>grid on
```



2.- Resolver el siguiente planteamiento:

$$U_{tt} = U_{xx} + U_x + 2 \;;\quad 0 \le x \le 1 \;;\; t \ge 0$$

Con las siguientes condiciones de contorno:

$$u_x(0,t) = 0 = aa.m \;;\; u(1,t) = t^2 = bb.m$$

Y las siguientes condiciones iniciales:

$$u(x,0) = 0 = ff.m \;;\; u_t(x,0) = 0 = v.m$$

Y sea, y(t) la solución del P.V.I.:

$$\begin{cases} y'' + ty' - 2y = 2, \\ y(0) = 0, \;\; y'(0) = 0, \end{cases}$$

Se pide:

a) **Calcular u(1/2,t), con error menor que 10^{-4}, para t [0,1]. Dibujar la función.**
b) **Calcular |u(t,t)-y(t)+1|, para t[0,1] y dibujar la función.**
c) **Calcular:**

$$\max_{t\epsilon[0,1]} \left| u^2\left(\frac{1}{2}, t\right) + y^2(t) \right|$$

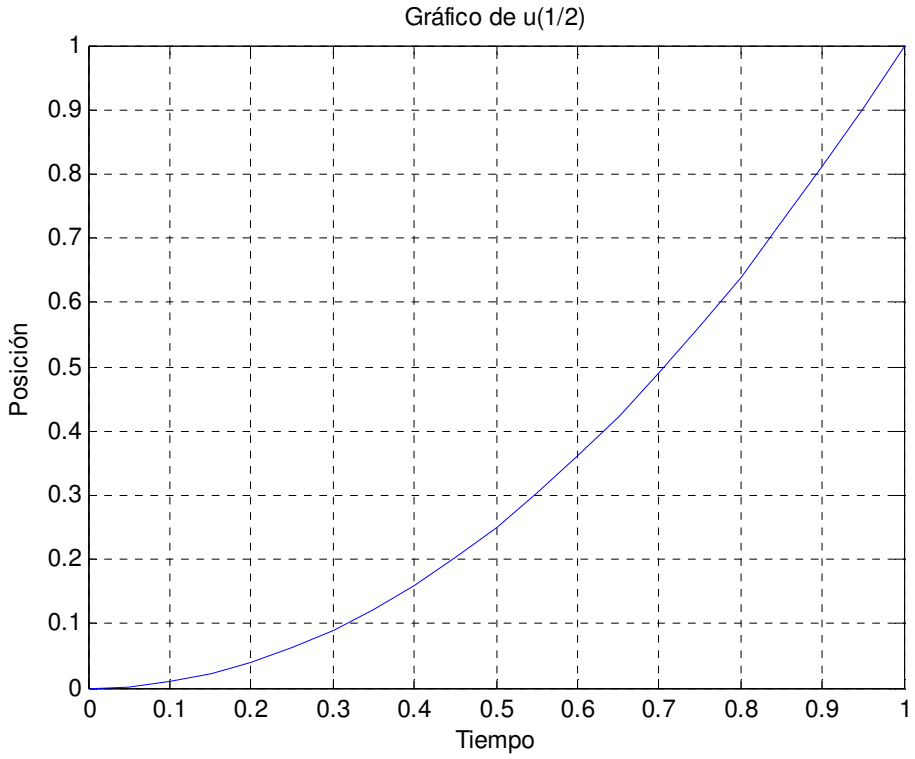
Donde se alcanza al máximo.

En este caso tengo condiciones Neumann-Direlech y un parametro adicional luego debo modificar mi algoritmo inicial. Los cambios son los siguientes:

Cambio la línea	Por la siguiente
1	function [W]=OndasND(c,L,m,N,T)
7	A=tridiagonalSI(m,2*(1-la^2),la^2+(k^2/(2*h)),la^2-(k^2/(2*h))); A(1,2)=2*(la^2);
13	W1(i+1)=(1-la^2)*ff(i*h)+(((la^2/2)+(k^2/(4*h)))*ff((i+1)*h))+(((la^2/2)-(k^2/(4*h)))*ff((i-1)*h))+k*v(i*h)+(k^2/2)*gg((i*h),0);
17	W=[W;bb(T)];
19	MW(2,:)=[W1;bb(k)]';
34	MW(j+2,:)=[W;bb((j+1)*k)]';
37	W=[W;bb(T)];

El algoritmo error no cambia en nada. Solo tengo que crear los archivos de las funciones aa, bb, ff y v. en este caso omito el u.m ya que no nos dan la solución exacta del problema. Sabemos que la función g(x,t)=2, recogida en el archivo gg.m

```
[W,WWf,MW,tiempo,ejex]=ondasND(c,L,m,N,T)
>> [W,WWf,MW,tiempo,ejex]=ondasND(1,1,10,10,1);
[m,N]=ondaserrorND(c,L,m,N,T,Tol)
>> [m,N]=ondaserrorND(1,1,10,10,1,0.0001)
m = 20
N = 20
>> [tiempo,MW]=ondasND(1,1,20,20,1);
>> y=MW(:,11);
>> x=tiempo;
>> plot(x,y)
>> hold on
>> grid on
>> title('Gráfico de u(1/2,t)')
>> xlabel('Tiempo')
>> ylabel('Posición')
```




```
>>u=diag(MW);
```

La ecuación diferencial dada la resolveré mediante el comando ode45, pero primero debo crear los siguientes archivos:

fvi.m

```
function Z=fvi(t,yy)
Z(1)=yy(2);
Z(2)=-t.*yy(2)+2.*yy(1)+2;
Z=Z';
```

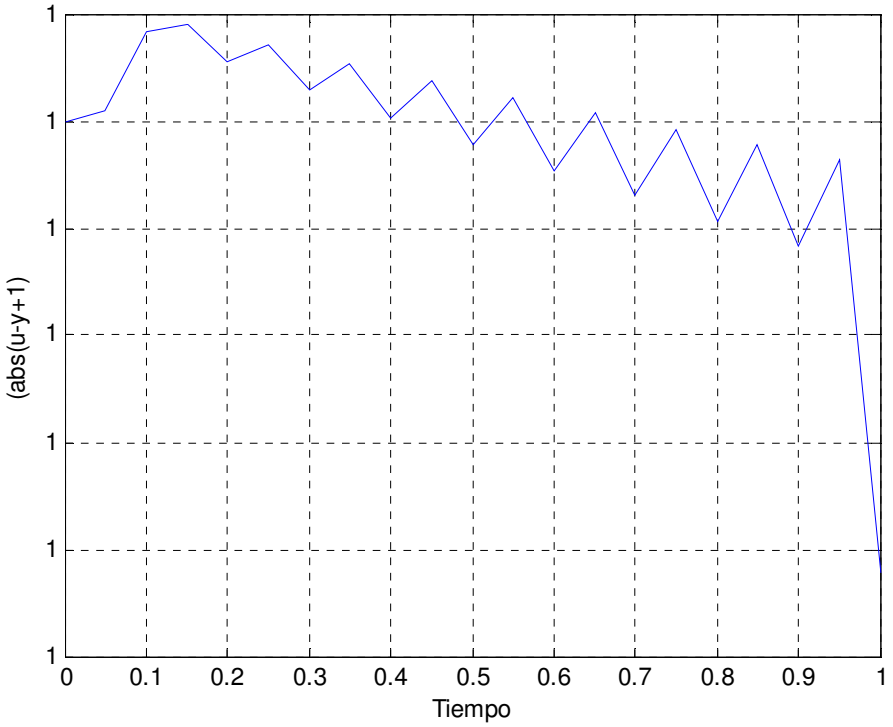
Ode45sistema

```
function S=ode45sistema(f,a,b,Za,M)
h=(b-a)/M;
T=a:h:b;
[T X]=ode45(f,T,Za);
format long
S=[T X];
```

En este caso M=length(u)-1, para que el vector y(t) y el vector u, sean de la misma longitud.

```
>> M=length(u)-1;
>> S=ode45sistema('fvi',0,1,[0 0],M);
>> y=S(:,2);
>> Sol=abs(u-y+1)

Sol =
   1.00000000000000
   1.00000000044666
   1.00000000413003
   1.00000000451837
   1.00000000273383
   1.00000000358296
   1.00000000140476
   1.00000000271303
   1.00000000012951
   1.00000000189541
   0.99999999889516
   1.00000000111739
   0.99999999768931
   1.00000000036679
   0.99999999650015
   0.99999999963210
   0.99999999531656
   0.99999999890253
   0.99999999412818
   0.99999999816807
   0.99999997893938

>> y=S(:,2);
>> x=tiempo;
>> plot(x,(abs(u-y+1)))
>> plot(x,(abs(u-y)))
```

```
>> plot(x,(abs(u-y+1)))
>> grid on
>> xlabel('Tiempo')
>> ylabel('(abs(u-y+1)')
```



3.- Resolver el siguiente planteamiento:

$$U_{tt} = U_{xx} - 20x^3 + 20t^3 \;\; ; \quad 0 \le x \le 1 \;\; ; \;\; t \ge 0$$

Siendo la solución real de U la siguiente:

$$u(x,t) = t^5 + x^5$$

Se pide:

a) **Razonar cuales son las condiciones de contorno e iniciales para la resolución del planteamiento mediante condiciones Dirilech-Dirilech y Neumann-Neuman.**
b) **Resolver por cada uno de los métodos con valores de m=20 y N=30 y comparar en cada algoritmo el error que se comete en u(x,1).**
c) **Dibujar la gráfica de u(x,1/3) en condiciones Dirilech-Dirilech, con un error menor a 10^{-2}.**
d) **Dibujar la gráfica de u(1/5,t). con 0<t<1 y un error menor a 10^{-2}, para condiciones Neumann-Dirilech.**

a) Las condiciones para Dirilech-Dirilech las deduzco de la siguiente forma:

$$Condiciones\ de\ contorno \begin{cases} u(0,t) = t^5 + 0^5 = t^5 \rightarrow aa.m \\ u(1,t) = t^5 + 1^5 = 1 + t^5 \rightarrow bb.m \end{cases}$$

$$Condiciones\ iniciales \begin{cases} u(x,0) = t^5 + x^5 = x^5 + 0^5 = x^5 \rightarrow ff.m \\ u_t = 5t^4 \rightarrow u_t(x,0) = 5 * 0^5 = 0 \rightarrow v.m \end{cases}$$

Las condiciones para Neumann-Neumann las deduzco de la siguiente forma:

$$Condiciones\ de\ contorno \begin{cases} u_x = 5x^4 \rightarrow u_x(0,t) = 5 * 0^5 = 0 \rightarrow aa.m \\ u_x = 5x^4 \rightarrow u_x(0,t) = 5 * 1^5 = 5 \rightarrow bb.m \end{cases}$$

$$Condiciones\ iniciales \begin{cases} u(x,0) = t^5 + x^5 = x^5 + 0^5 = x^5 \rightarrow ff.m \\ u_t = 5t^4 \rightarrow u_t(x,0) = 5 * 0^5 = 0 \rightarrow v.m \end{cases}$$

b) Para resolver el planteamiento mediante condiciones Dirilech-Dirilech uso el algoritmo expuesto al principio de este capitulo tal cual:

```
[W,ejex,UU]=ondasDD(c,L,m,N,T)
>> [W,ejex,UU]=ondasDD(1,1,20,30,1);
>> Wdd=W;
```

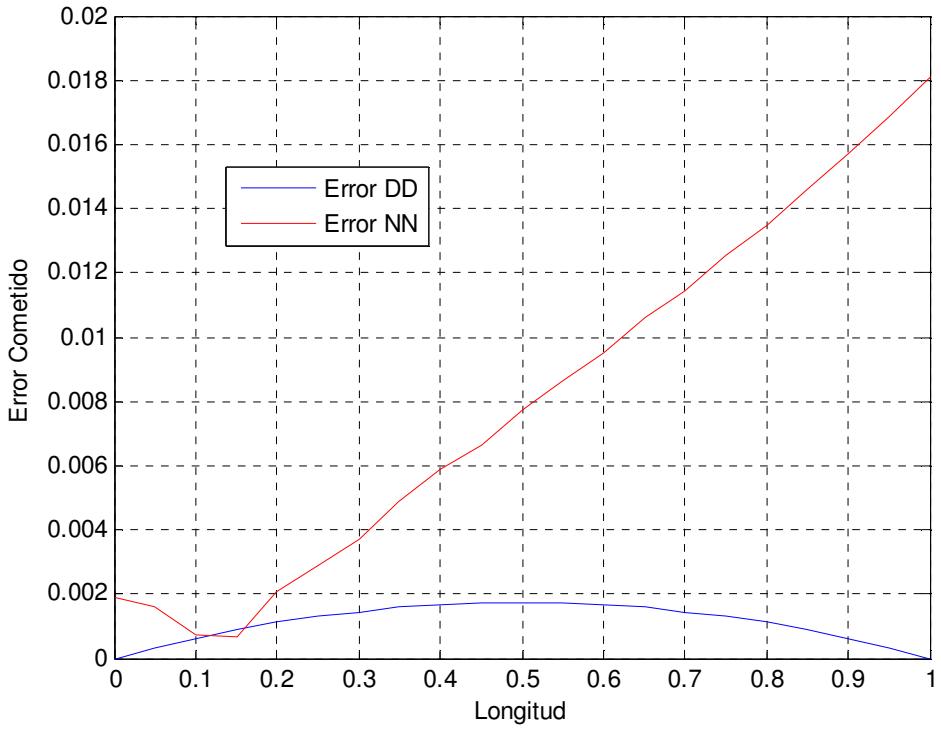
Sin embargo para usar las condiciones Neumann-Neumann, si que debo hacer los siguientes cambios:

Cambio la línea	Por la siguiente
1	function [W]=ondasDD(c,L,m,N,T)
17	MW(1,:)=[W0]';
19	MW(2,:)=[W1]';
34	MW(j+2)=[W];
37	W=[W];

```
[W,ejex,UU]=ondasNN(c,L,m,N,T)
>> [W,ejex,UU]=ondasNN(1,1,20,30,1);
>> Wnn=W;
>> Ed=abs(UU-Wdd);
>> Enn=abs(UU-Wnn);
>> M=[Wdd Edd Wnn Enn]

M =
   1.0000         0    1.0019    0.0019
   1.0003    0.0003    1.0016    0.0016
   1.0006    0.0006    1.0007    0.0007
   1.0010    0.0009    0.9994    0.0007
   1.0014    0.0011    0.9983    0.0021
   1.0023    0.0013    0.9981    0.0029
   1.0039    0.0015    0.9987    0.0037
   1.0068    0.0016    1.0003    0.0049
   1.0119    0.0017    1.0044    0.0059
   1.0202    0.0017    1.0118    0.0066
```

```
   1.0330    0.0017    1.0235    0.0078
   1.0520    0.0017    1.0417    0.0086
   1.0794    0.0017    1.0683    0.0095
   1.1176    0.0016    1.1054    0.0106
   1.1695    0.0015    1.1567    0.0114
   1.2386    0.0013    1.2247    0.0126
   1.3288    0.0011    1.3142    0.0135
   1.4446    0.0009    1.4291    0.0146
   1.5911    0.0006    1.5748    0.0157
   1.7741    0.0003    1.7569    0.0169
   2.0000         0    1.9819    0.0181
```

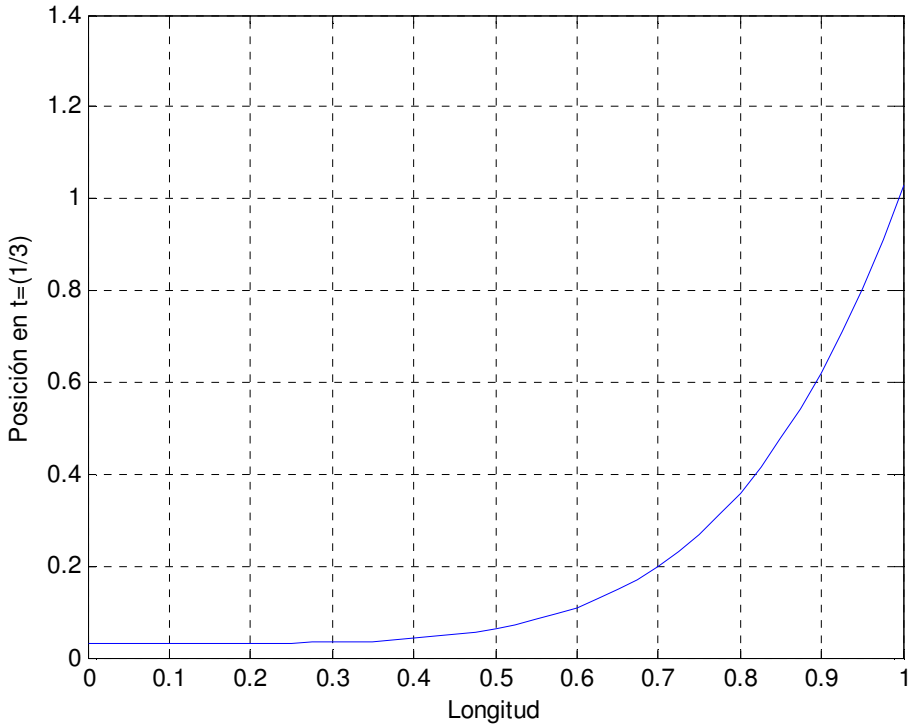
```
>> plot(ejex,Edd,ejex,Enn,'r')
>> grid on
>> xlabel('Longitud')
>> ylabel('Error Cometido')
>> legend('Error DD','Error NN')
```



c)Procedemos como hemos hecho hasta ahora, corriendo el algoritmo ondasDD con los argumentos de salidas iguales a los que le hacen falta como entrada al ondaserrorDD y así obtendremos el m y N correcto, pero como mínimo N debe ser múltiplo a 3 ya que nos pide la posición de la cuerda en t=1/3.

```
>> [W,WWf,MW,ejex]=ondasDD(1,1,10,15,1);
>> [m,N]=ondaserrorDD(1,1,10,15,1,0.01)
m = 40
N = 60
>> [W,ejex,MW]=ondasDD(1,1,40,60,1);
>> x=ejex;
>> y=MW(31,:);
```

```
>> plot(x,y)
>> grid on
>> xlabel('Longitud')
>> ylabel('Posición en t=(1/3)')
```



d) Para usar las condiciones Neumann-Dirilech, haremos los siguientes cambios:

Cambio la línea	Por la siguiente
1	function [W]=ondasND(c,L,m,N,T)
17	MW(1,:)=[W0;bb(T)]';
19	MW(2,:)=[W1;bb(k)]';
34	MW(j+2)=[W;bb(j+1)*k]';
37	W=[W,bb(T)];

```
[W,WWf,MW,tiempo,ejex]=ondasND(c,L,m,N,T)
>> [W,WWf,MW,tiempo,ejex]=ondasND(1,1,10,10,1);
>> [m,N]=ondaserrorND(1,1,10,10,1,0.001)
 [m,N]=ondaserrorND(c,L,m,N,T,Tol)
m = 80
N = 80
>> [tiempo,MW]=ondasND(1,1,80,80,1);
>> x=tiempo;
>> m=80;
>> y=MW(:,(m/5)+1);
>> plot(x,y)
>> legend('P(1/5)','location','best')
>> xlabel('Tiempo transcurrido')
>> ylabel('Posición')
>> grid on
```

Carmelo Camacho González
Fernando J. Manobel Ponce