PROBLEMA 1: SISTEMA RESONANTE DE MUELLES.

```
clearvars
clear all
clc
```

Apartado 1:

Haciendo:

$$x_1 = x(t)$$

$$x_2 = x'(t)$$

Se tiene que:

$$x_2 = x_1'$$

Y en la ecuación diferencial:

$$x_2' = 4\sin(5t) - 25x_1$$

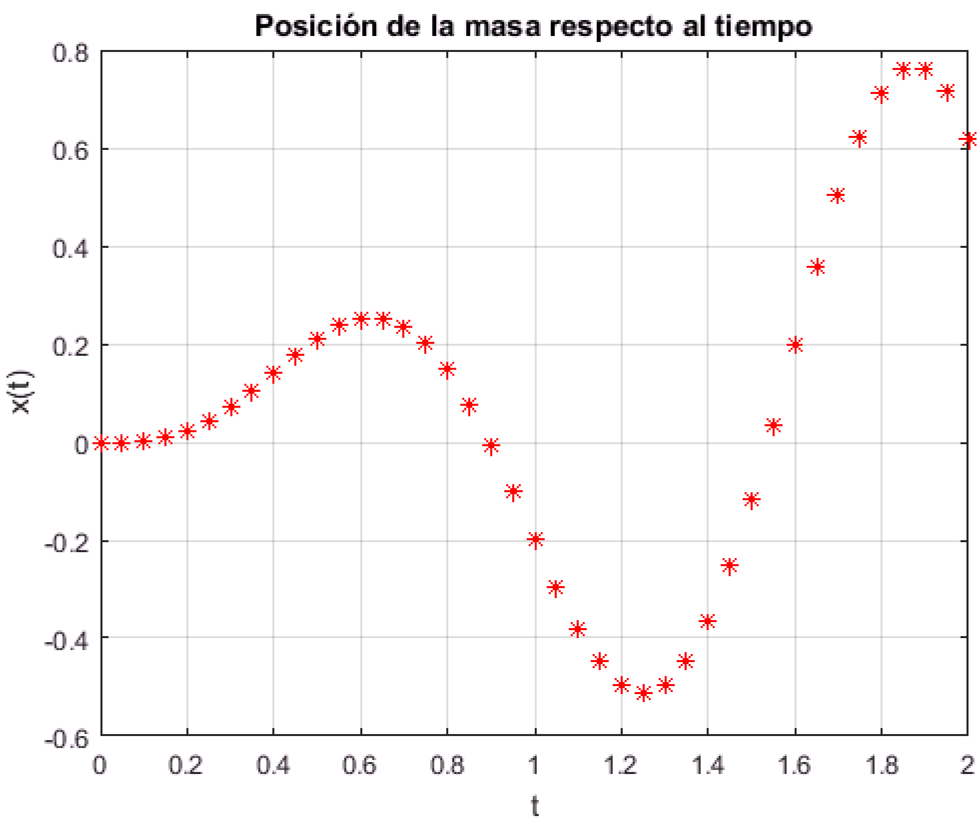
Luego, el sistema de ecuaciones diferenciales de primer orden es:

$$x_1' = x_2$$

$$x_2' = 4\sin(5t) - 25x_1$$

Apartado 2:

```
z = @(t,z) [z(2); 4*sin(5*t)-25*z(1)];
t0 = 0; tf = 2; nsi = 40; dt = (tf-t0)/nsi; z0 = [0;0];
[tsolHeun,xsolHeun] = HeunOrden2(z,t0,tf,z0,dt,nsi);
```



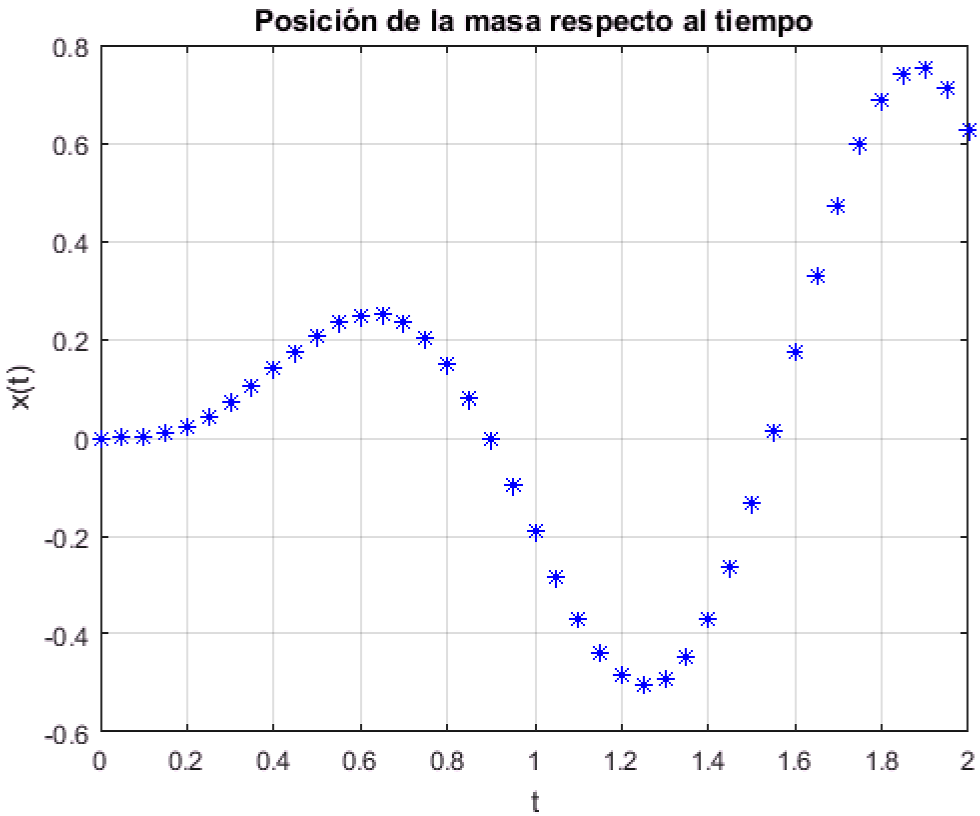

```
xfindtHeun = [0:0.25:2];
for i = 1:length(xfindtHeun)
    jx = find(tsolHeun == xfindtHeun(i));
    xfindHeun(i) = xsolHeun(1,jx);
end
TfindHeun = table(xfindtHeun',xfindHeun');
TfindHeun.Properties.VariableNames = {'Instante' 'Posicion'};
TfindHeun
```

TfindHeun = *9×2 table*

Instante	Posicion
0	0
0.25	0.043576
0.5	0.2108
0.75	0.20143
1	-0.19988
1.25	-0.51067
1.5	-0.11685
1.75	0.6243
2	0.61907

Apartado 3:

```
[tsolRK4,xsolRK4] = RungeKutta4(z,t0,tf,z0,dt,nsi);
```




```
xfindtRK4 = [0:0.25:2];
for i = 1:length(xfindtRK4)
    jx = find(tsolRK4 == xfindtRK4(i));
    xfindRK4(i) = xsolRK4(1,jx);
end
TfindRK4 = table(xfindtRK4',xfindRK4');
TfindRK4.Properties.VariableNames = {'Instante' 'Posicion'};
TfindRK4
```

TfindRK4 = *9×2 table*

Instante	Posicion
0	0
0.25	0.04439
0.5	0.2081
0.75	0.20044
1	-0.19015
1.25	-0.50235
1.5	-0.13299
1.75	0.59648
2	0.62775

PROBLEMA 2: PROBLEMA DE VALOR INICIAL:

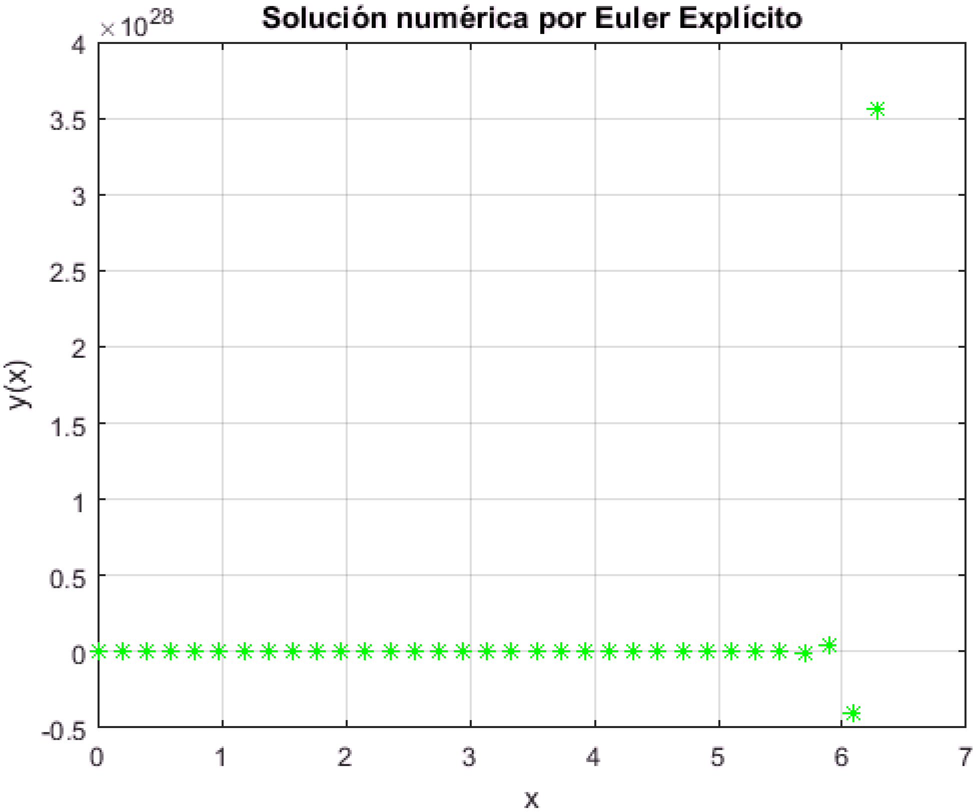
Apartado 1:

```
EDO = @(x,y,l) l.*(-y+sin(x));
ysol = @(x,l) (l/(1+l^2))*exp(-l*x)+(l^2/(1+l^2))*sin(x)-(l/(1+l^2))*cos(x);
lambda = 50;
f = @(x,y) EDO(x,y,lambda);
```

```
yexact = @(x) ysol(x,lambda);
x0 = 0; xf = 2*pi; y0 = 0; nsi = 32; dx =(xf-x0)/nsi;
```

SOLUCIÓN POR EULER EXPLÍCITO

```
[xsolEuExp,ysolEuExp] = EulerExplicito(f,x0,y0,dx,nsi);
```



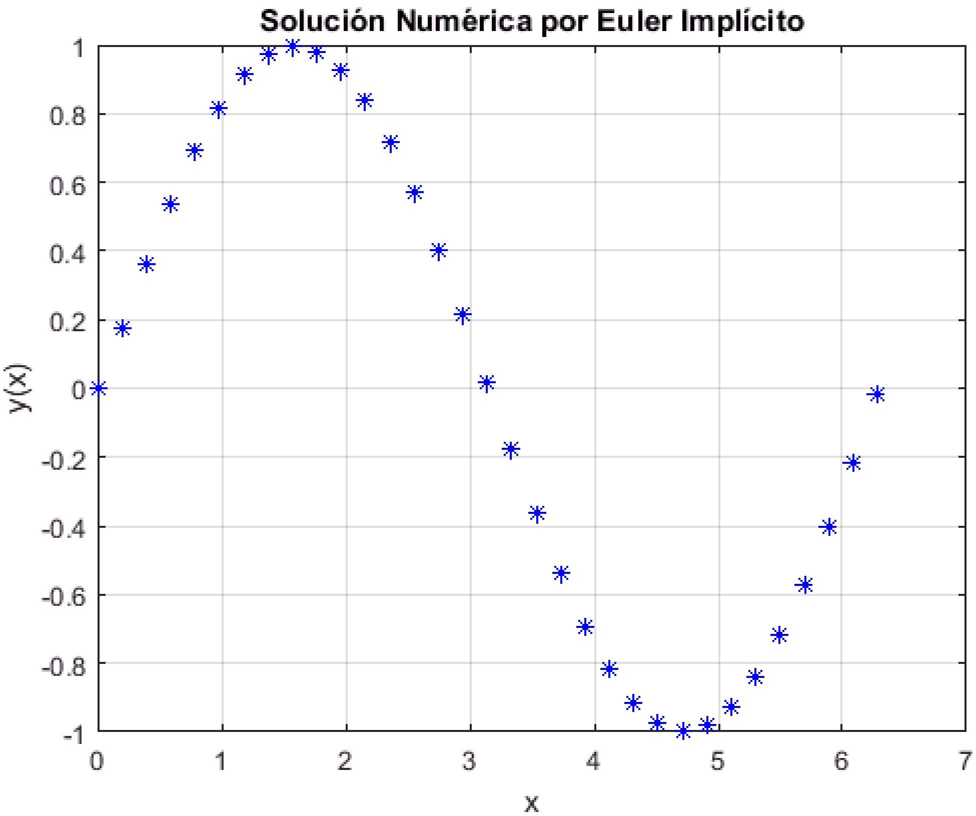
Error máximo cometido:

```
ErrExp = abs(yexact(xsolEuExp)-ysolEuExp);
jEmaxExp = find(ErrExp == max(ErrExp));
EmaxExp = ErrExp(jEmaxExp)
```

```
EmaxExp = 3.5545e+28
```

SOLUCIÓN POR EULER IMPLÍCITO

```
x0 = 0; xf = 2*pi; y0 = 0; nsi = 32; dx =(xf-x0)/nsi;
[xsolEuImp,ysolEuImp] = EulerImplicito(f,x0,y0,dx,nsi);
```



Error máximo cometido:

```
ErrImp = abs(yexact(xsolEuImp)-ysolEuImp);
jEmaxImp = find(ErrImp == max(ErrImp));
EmaxImp = ErrImp(jEmaxImp)
```

```
EmaxImp = 0.0019
```

Apartado 2:

Proponemos una tolerancia tal que a partir de ello se obtenga el número de subintervalos mínimos que satisfacen un buen ajuste o buena aproximación.

```
es = 10^(-2);
x0 = 0; xf = 2*pi; y0 = 0; n0 = 100; dx =(xf-x0)/n0;
EstimarSubintervalo(f,x0,y0,dx,n0,yexact,es,xf)
```

```
A partir de 263 subintervalos, se tendrá un error máximo de 0.010000.
```

PROBLEMA 3: OSCILADOR DE VAN DER POL.

Haciendo:

$$x_1 = x(t)$$

$$x_2 = x'(t)$$

Se tiene que:

$$x_2 = x_1'$$

Y en la ecuación diferencial:

$$x_2' = \mu(1 - x_1^2)x_2 - x_1$$

Luego, el sistema de ecuaciones diferenciales de primer orden es:

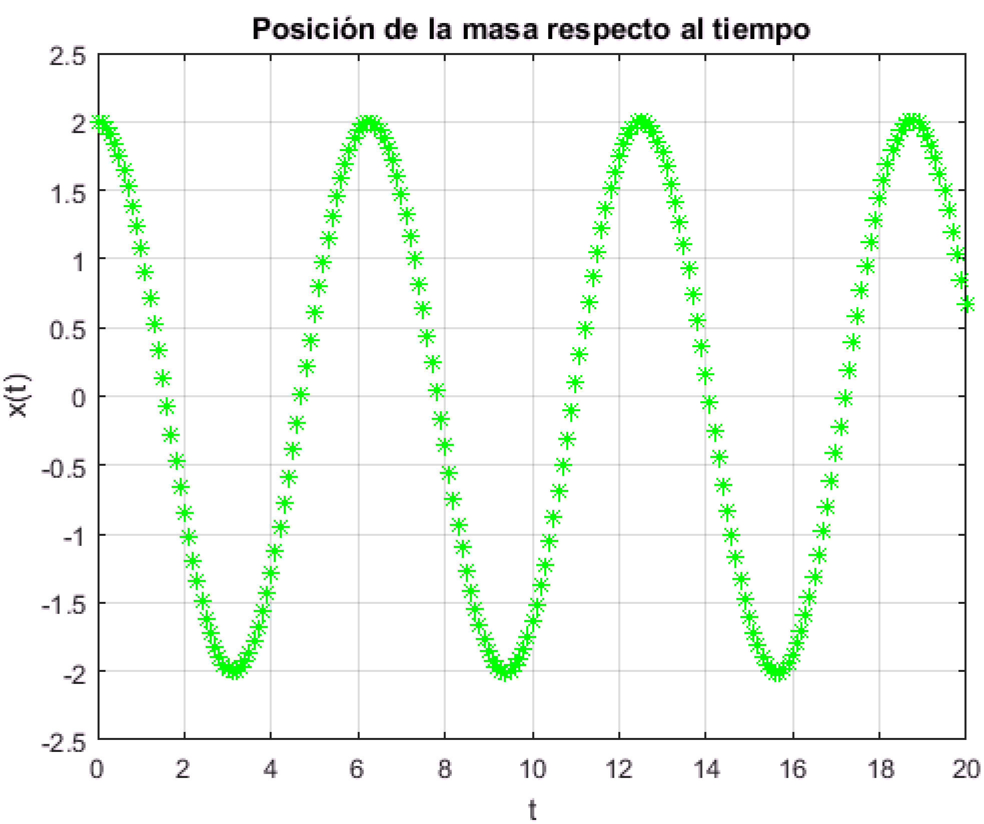
$$x_1' = x_2$$

$$x_2' = \mu(1 - x_1^2)x_2 - x_1$$

```
sist = @(t,z,u) [z(2);u*(1-z(1)^2)*z(2)-z(1)];
z0 =[2;0]; h = 0.1; t0 = 0; tf = 20;
tfind = [2:6:14 16];
```

Apartado 1:

```
u = 0;
z = @(t,z) sist(t,z,u);
[tsolAB2u0,zsolAB2u0] = AdamsBashforth2(z,z0,t0,h,tf);
```



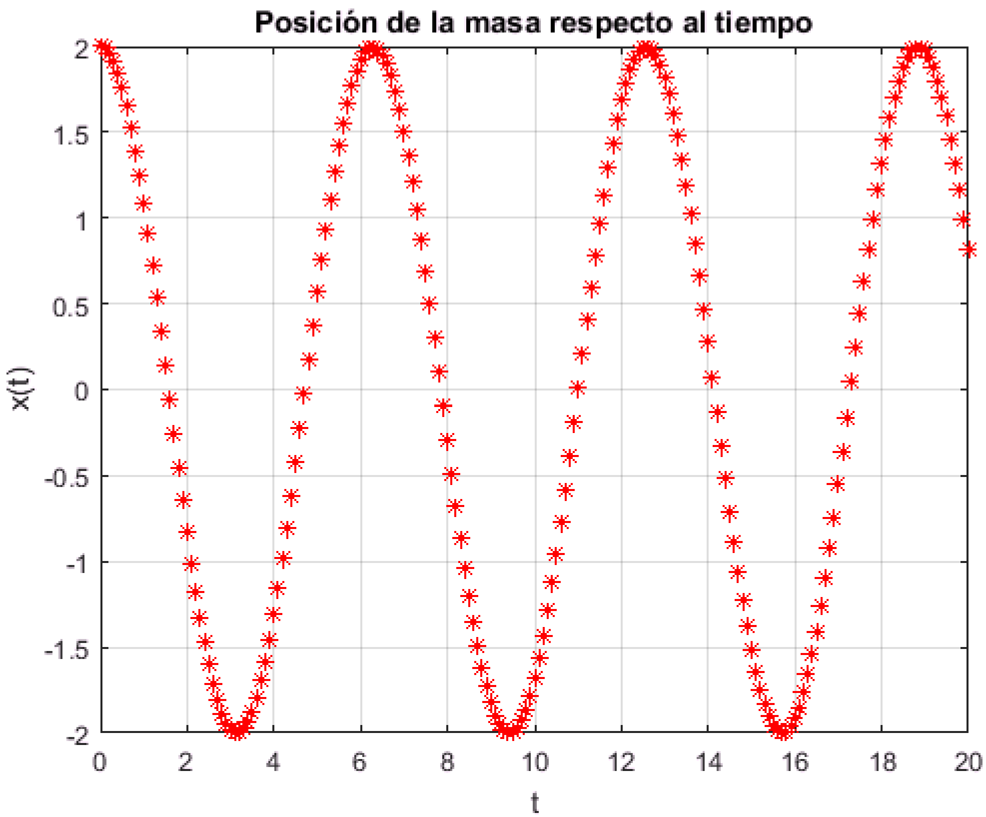

```
for i = 1:length(tfind)
    jt = find(tsolAB2u0 == tfind(i));
    zfindAB2u0(i) = zsolAB2u0(1,jt);
end
TfindAB2u0 = table(tfind',zfindAB2u0');
```

```
TfindAB2u0.Properties.VariableNames = {'Instante' 'Posicion'};
TfindAB2u0
```

```
TfindAB2u0 = 4×2 table
    Instante    Posicion
    ________    ________

     2          -0.84717
     8          -0.35709
    14           0.15818
    16           -1.8804
```

```
[tsolAB4u0,zsolAB4u0] = AdamsBashforth4(z,z0,t0,h,tf);
```



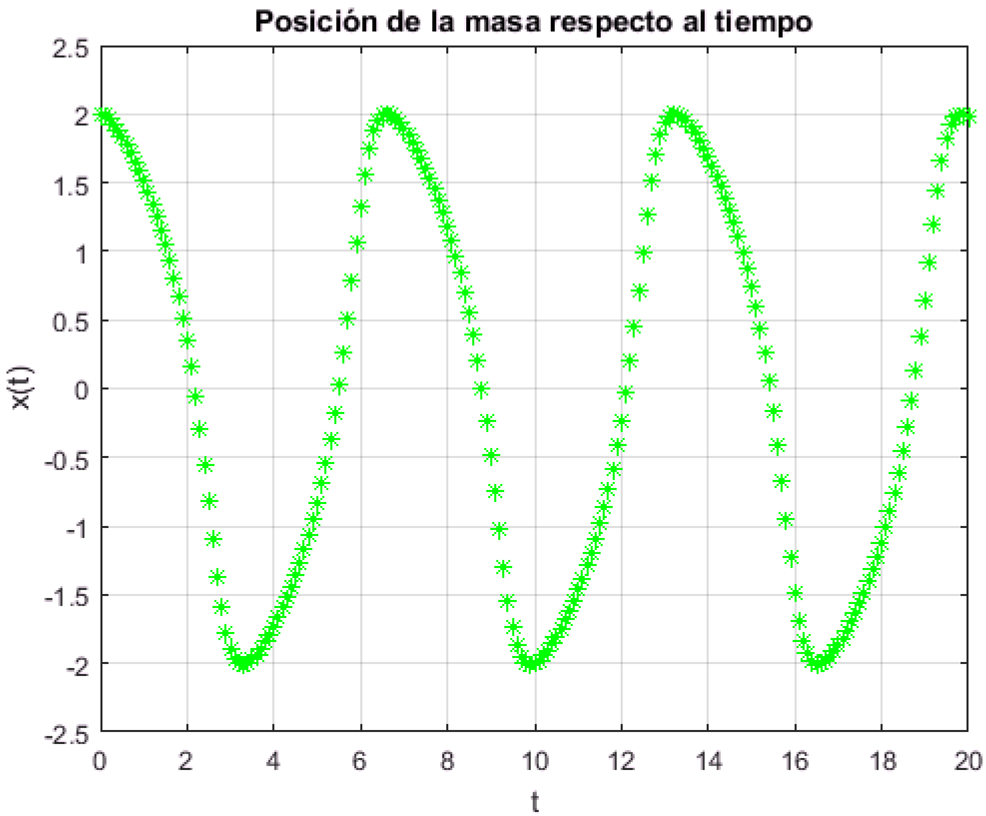

```
for i = 1:length(tfind)
    jt = find(tsolAB4u0 == tfind(i));
    zfindAB4u0(i) = zsolAB4u0(1,jt);
end
TfindAB4u0 = table(tfind',zfindAB4u0');
TfindAB4u0.Properties.VariableNames = {'Instante' 'Posicion'};
TfindAB4u0
```

```
TfindAB4u0 = 4×2 table
    Instante    Posicion
    ________    ________

     2          -0.83218
     8          -0.29046
    14           0.27439
    16           -1.9155
```

Apartado 2:

```
u = 1;
z = @(t,z) sist(t,z,u);
[tsolAB2u1,zsolAB2u1] = AdamsBashforth2(z,z0,t0,h,tf);
```

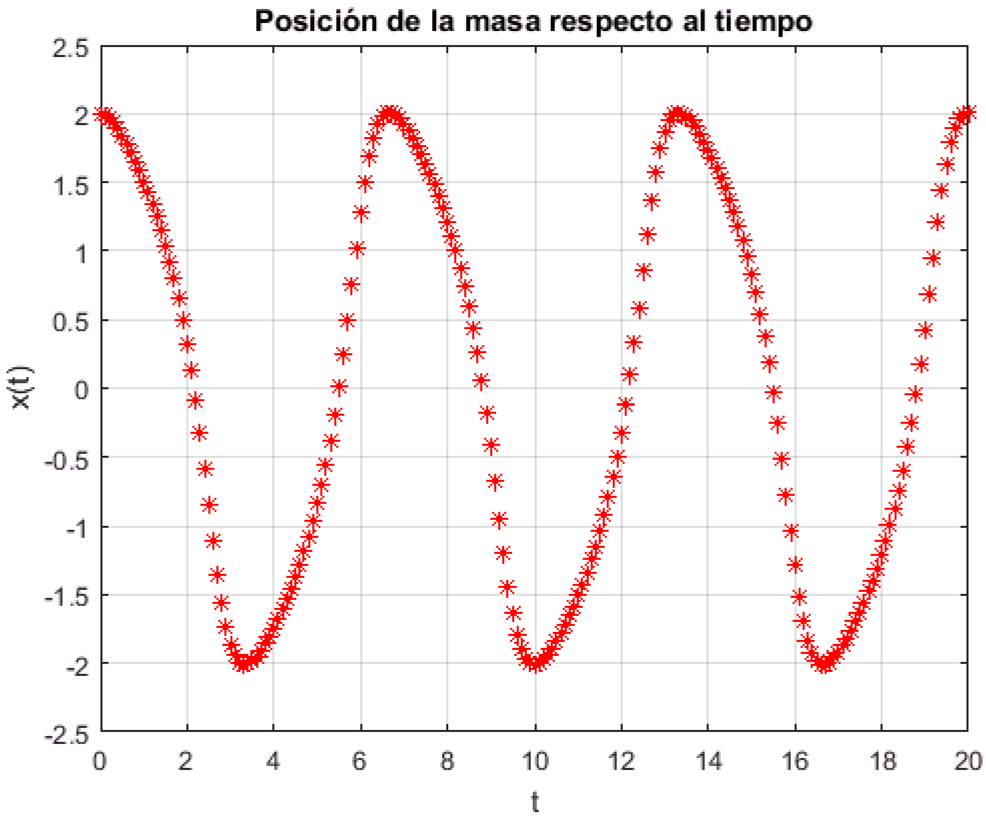



```
for i = 1:length(tfind)
    jt = find(tsolAB2u1 == tfind(i));
    zfindAB2u1(i) = zsolAB2u1(1,jt);
end
TfindAB2u1 = table(tfind',zfindAB2u1');
TfindAB2u1.Properties.VariableNames = {'Instante' 'Posicion'};
TfindAB2u1
```

TfindAB2u1 = 4×2 table

Instante	Posicion
2	0.34428
8	1.1813
14	1.6894
16	-1.4804

```
[tsolAB4u1,zsolAB4u1] = AdamsBashforth4(z,z0,t0,h,tf);
```




```
for i = 1:length(tfind)
    jt = find(tsolAB4u1 == tfind(i));
    zfindAB4u1(i) = zsolAB4u1(1,jt);
end
TfindAB4u1 = table(tfind',zfindAB4u1');
TfindAB4u1.Properties.VariableNames = {'Instante' 'Posicion'};
TfindAB4u1
```

TfindAB4u1 = *4×2 table*

Instante	Posicion
2	0.3234
8	1.214
14	1.7397
16	-1.2894

CÓDIGO DE LOS MÉTODOS NUMÉRICOS USADOS

MÉTODO DE HEUN

```
function [x,z] = HeunOrden2(df,t0,tf,z0,h,n)
    x = [t0:h:tf];
    z(:,1) = z0;
    for i = 1:n
        k1 = h*df(x(i),z(:,i));
        k2 = h*df(x(i)+h,z(:,i)+k1);
        z(:,i+1) = z(:,i)+(1/2)*(k1+k2);
    end
    plot(x,z(1,:),'r*')
```

```
    xlabel('t')
    ylabel('x(t)')
    title('Posición de la masa respecto al tiempo')
    grid on
end
```

<u>MÉTODO DE RUNGE KUTTA DE ORDEN 4</u>

```
function [x,z] = RungeKutta4(df,t0,tf,z0,h,n)
    x = [t0:h:tf];
    z(:,1) = z0;
    for i = 1:n
        k1 = df(x(i),z(:,i));
        k2 = df(x(i)+(h/2),z(:,i)+(h/2)*k1);
        k3 = df(x(i)+(h/2),z(:,i)+(h/2)*k2);
        k4 = df(x(i)+h,z(:,i)+h*k3);
        z(:,i+1) = z(:,i)+(h/6)*(k1+2*k2+2*k3+k4);
    end
    plot(x,z(1,:),'b*')
    xlabel('t')
    ylabel('x(t)')
    title('Posición de la masa respecto al tiempo')
    grid on
end
```

<u>MÉTODO DE EULER EXPLÍCITO</u>

```
function [x,y] = EulerExplicito(f,x0,y0,h,n)
    y(1) = y0;
    x(1) = x0;
    for i = 1:n
        y(i+1) = y(i) + h*f(x(i),y(i));
        x(i+1) = x(i) + h;
    end
    plot(x,y,'g*')
    xlabel('x')
    ylabel('y(x)')
    title('Solución numérica por Euler Explícito')
    grid on
end
```

Con este algoritmo estiramos el número de subintevarlos necesarios:

```
function EstimarSubintervalo(f,x0,y0,h,n,ysol,es,xf)
    while (1)
        [x,y] = SolEuler(f,x0,y0,h,n);
        Err =  abs(ysol(x)-y);
        jEmax = find(Err == max(Err));
        Emax = Err(jEmax);
        if Emax <= es || n > 500
            break;
        end
        n = n + 1;
        h =(xf-x0)/n;
    end
    nsi = n;
    fprintf('A partir de %d subintervalos, se tendrá un error máximo de %f.\n',nsi,es)
end
function [x,y] = SolEuler(f,x0,y0,h,n)
```

```
    y(1) = y0;
    x(1) = x0;
    for i = 1:n
        y(i+1) = y(i)+h*f(x(i),y(i));
        x(i+1) = x(i)+h;
    end
end
```

MÉTODO DE EULER IMPLÍCITO

```
function [x,y] = EulerImplicito(f,x0,y0,h,n)
    x(1) = x0;
    y(1) = y0;
    for i = 1:n
        x(i+1) = x0+i*h;
        g = @(u) u-y(i)-h*f(x(i+1),u);
        y(i+1) = fzero(g,y(i));
    end
    plot(x,y,'b*')
    xlabel('x')
    ylabel('y(x)')
    title('Solución Numérica por Euler Implícito')
    grid on
end
```

MÉTODO DE ADAMS BASHFORTH DE ORDEN 2

```
function [t,z] = AdamsBashforth2(df,z0,t0,h,tf)
    t = [t0:h:tf];
    n = (tf-t0)/h;
    nAB2 = 1;
    z(:,1) = z0;
    for i = 1:nAB2
        k1 = df(t(i),z(:,i));
        k2 = df(t(i)+(h/2),z(:,i)+(h/2)*k1);
        k3 = df(t(i)+(h/2),z(:,i)+(h/2)*k2);
        k4 = df(t(i)+h,z(:,i)+h*k3);
        z(:,i+1) = z(:,i)+(h/6)*(k1+2*k2+2*k3+k4);
    end
    for i = 2:n
        AB12 = (3/2)*df(t(i),z(:,i));
        AB22 = (-1/2)*df(t(i-1),z(:,i-1));
        z(:,i+1) = z(:,i)+h*(AB12+AB22);
    end
    plot(t,z(1,:),'g*')
    xlabel('t')
    ylabel('x(t)')
    title('Posición de la masa respecto al tiempo')
    grid on
end
```

MÉTODO DE ADAMS BASHFORTH DE ORDEN 4

```
function [t,z] = AdamsBashforth4(df,z0,t0,h,tf)
    t = [t0:h:tf];
    n = (tf-t0)/h;
    nAB4 = 3;
    z(:,1) = z0;
    for i = 1:nAB4
```

```
        k1 = df(t(i),z(:,i));
        k2 = df(t(i)+(h/2),z(:,i)+(h/2)*k1);
        k3 = df(t(i)+(h/2),z(:,i)+(h/2)*k2);
        k4 = df(t(i)+h,z(:,i)+h*k3);
        z(:,i+1) = z(:,i)+(h/6)*(k1+2*k2+2*k3+k4);
    end
    for i = 4:n
        AB14 = (55/24)*df(t(i),z(:,i));
        AB24 = (-59/24)*df(t(i-1),z(:,i-1));
        AB34 = (37/24)*df(t(i-2),z(:,i-2));
        AB44 = (-9/24)*df(t(i-3),z(:,i-3));
        z(:,i+1) = z(:,i) + h*(AB14+AB24+AB34+AB44);
    end
    plot(t,z(1,:),'r*')
    xlabel('t')
    ylabel('x(t)')
    title('Posición de la masa respecto al tiempo')
    grid on
end
```