Universidad Nacional San Agustin de Arequipa

Facultad de Ingenierias de Produccion y Servicios

Escuela Profesional de Ingenieria de Sistemas

Fisica Computacional

Alumno:

Fuentes Paredes Nelson Alejandro

Mayo 2020

[ ]:

```python

```

[1]:

```python
%matplotlib inline
#%matplotlib notebook
```

1 Importando librerias

[2]:

```python
import math
import numpy as np
from matplotlib import pyplot as plt
from matplotlib.animation import FuncAnimation
from IPython.display import HTML
from mpl_toolkits import mplot3d
```

1.1 Sea un resorte con k = 0.1 N/m unido a una masa m = 0.2 kg se mueve horizontalmente en un medio donde no hay fricción. Si las condiciones iniciales son x = 2 m y vx = 0

Declaración de variables del sistema

[3]:

```python
h=0.1
k=0.1
m=0.2
c=0.0
F0=0.0
w=0.0
t=0
tfin=100
p=np.array([2])
v=np.array([0])
```

Ejecución

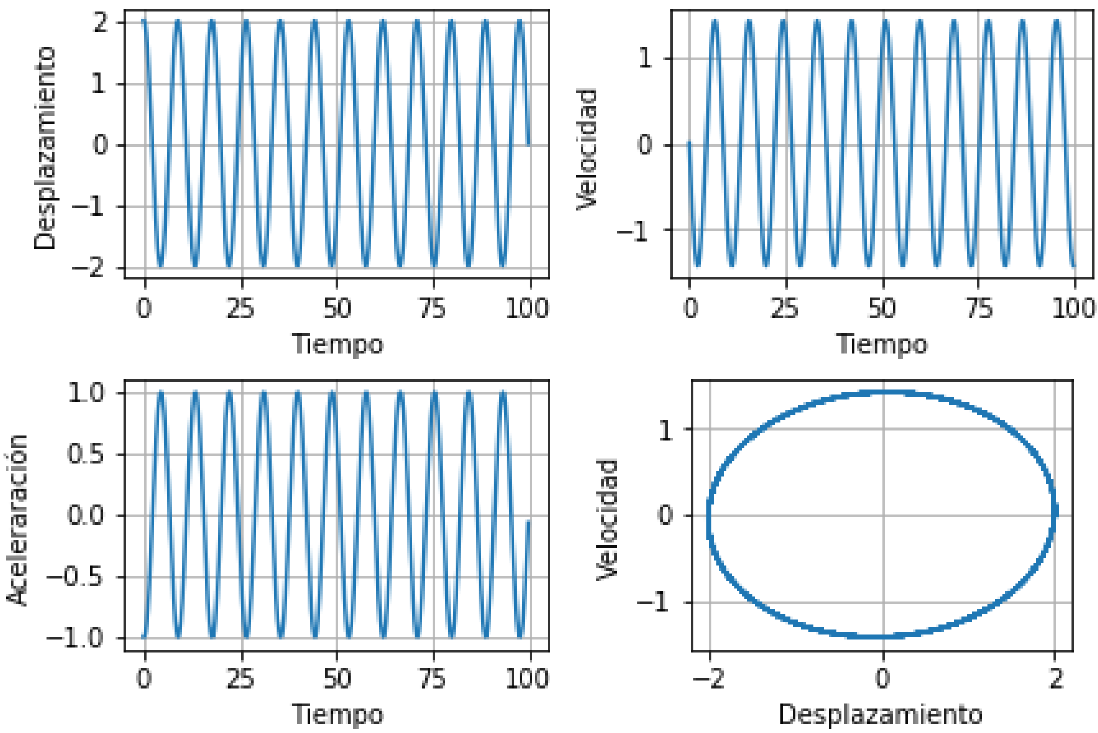
[4]:

```python
a=np.array([-k*p[0]/m-c*v[0]/m+F0/m*math.cos(w*t)])
pt=[t]
pv=[v[0]]
px=[p[0]]
pa=[a[0]]
u =[p[0]**2*k*0.5]
K=[0.5*m*v[0]**2]
E = [ u[0] + K[0] ]
for ts in np.arange(t+h, tfin, h):
    a =np.array([-k*p[0]/m-c*v[0]/m+F0/m*math.cos(w*ts)])
    v = v + a*h
    p  = p  + v*h
    u_ = p[0]**2*k/2
```

```
    u.append(u_)
    pt.append(ts)
    px.append(p[0])
    pv.append(v[0])
    pa.append(a[0])
    K_ = 0.5*m*v[0]**2
    K.append(K_)
    E.append( u_ + K_ )
```

(a) x − t, v − t, a − t y v − x en cuatro gráficos utilizando la instrucción subplot

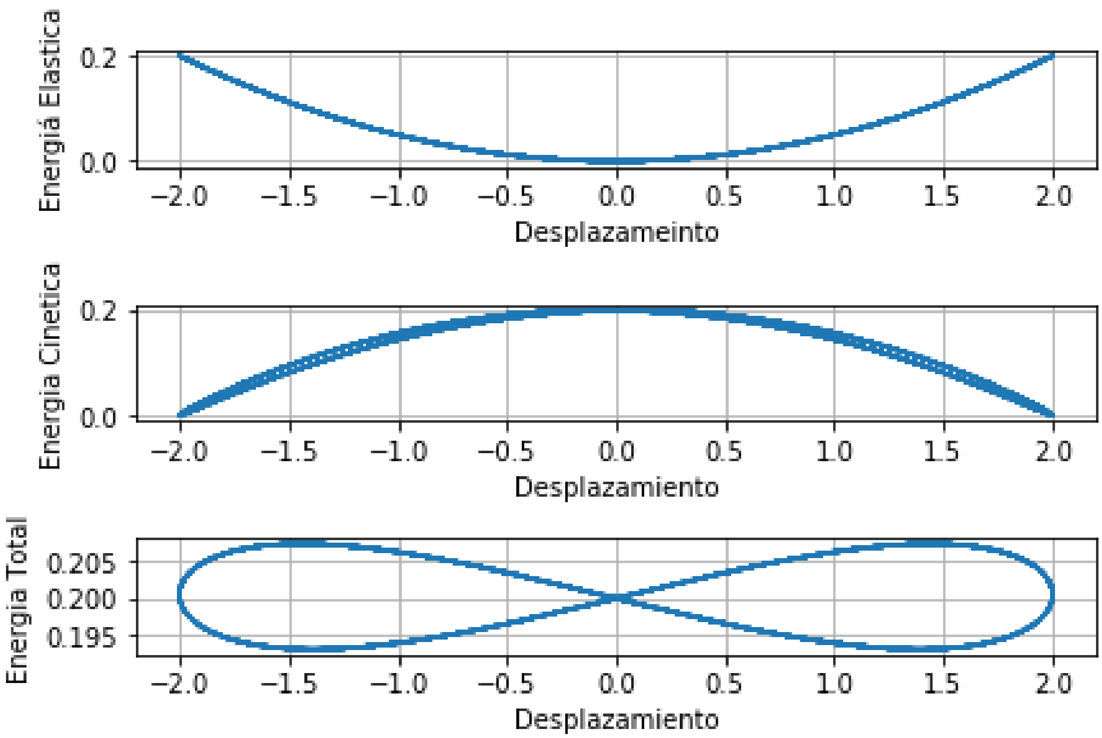
[5]:
```
fig, axes = plt.subplots(2, 2, constrained_layout=True)
axes[0][0].plot(pt, px)
axes[0][0].grid()
axes[0][0].set(xlabel = 'Tiempo', ylabel='Desplazamiento')

axes[0][1].plot(pt, pv)
axes[0][1].grid()
axes[0][1].set(xlabel = 'Tiempo', ylabel='Velocidad')

axes[1][0].plot(pt, pa)
axes[1][0].grid()
axes[1][0].set(xlabel = 'Tiempo', ylabel='Aceleraración')

axes[1][1].plot(px, pv)
axes[1][1].grid()
axes[1][1].set(xlabel = 'Desplazamiento', ylabel='Velocidad')
axes[1][1].set_aspect('equal')

#def animate(i):
#    line.set_data(pt[:i], px[:i])
#    line2.set_data(pt[:i], pv[:i])
#    line3.set_data(pt[:i], pa[:i])
#    line4.set_data(px[:i], pv[:i])
#    return [line, line2, line3, line4]

#ani = FuncAnimation(fig, animate, interval=1)
#HTML(ani.to_jshtml())
plt.show()
```



(b)U − x, K − x, E − x en un sólo gráfico

[6]:
```
fig, axes = plt.subplots(3, 1, constrained_layout=True)
axes[0].plot(px, u)
axes[0].grid()
axes[0].set(ylabel = 'Energiá Elastica' , xlabel='Desplazameinto')
axes[0]
axes[1].plot(px, K)
axes[1].grid()
axes[1].set(ylabel = 'Energia Cinetica', xlabel='Desplazamiento')

axes[2].plot(px, E)
axes[2].grid()
axes[2].set(ylabel = 'Energia Total', xlabel='Desplazamiento')

#def animate(i):
#    line.set_data(px[:i], u[:i])
#    line2.set_data(px[:i], K[:i])
#    line3.set_data(px[:i], E[:i])
#    return [line, line2, line3]

#ani = FuncAnimation(fig, animate, interval=1)
#HTML(ani.to_jshtml())
```

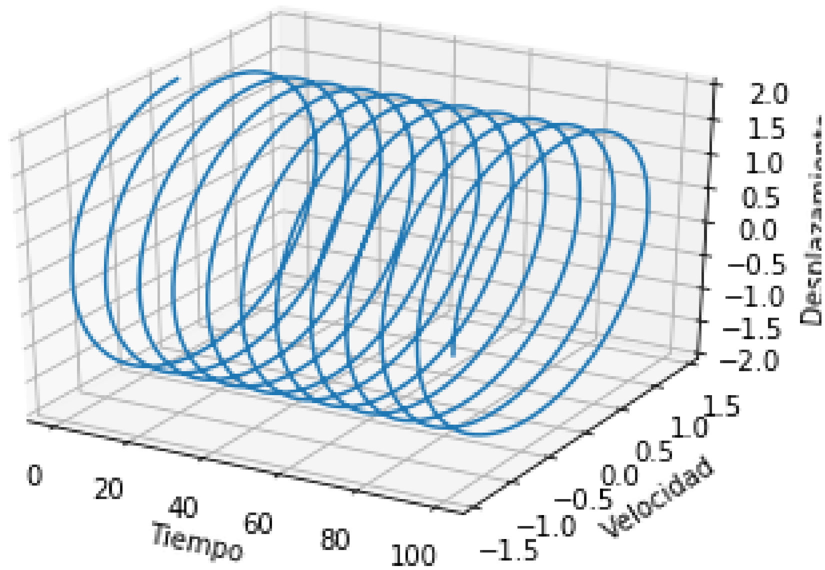
```
plt.show()
```



(c) v − x − t

[7]:
```
fig = plt.figure()
axes = plt.axes(projection='3d')
axes.plot3D(pt, pv, px)
axes.set(xlabel='Tiempo', ylabel='Velocidad',zlabel='Desplazamiento')
```

[7]:
```
[Text(0.5, 0, 'Desplazamiento'),
 Text(0.5, 0, 'Velocidad'),
 Text(0.5, 0, 'Tiempo')]
```



1.2 Sea un resorte con k = 0.1 N/m unido a una masa m = 0.2 kg se mueve horizontalmente en un medio c = 0.05 Ns/m donde hay fricción. Si las condiciones iniciales son x = 0 y vx = −2 m/s.

Declaración de variables del sistema

[8]:
```
h=0.1
k=0.1
m=0.2
c=0.05
F0=0.0
w=0.0
t=0
tfin=100
p=np.array([0])
v=np.array([-2])
```

Ejecución

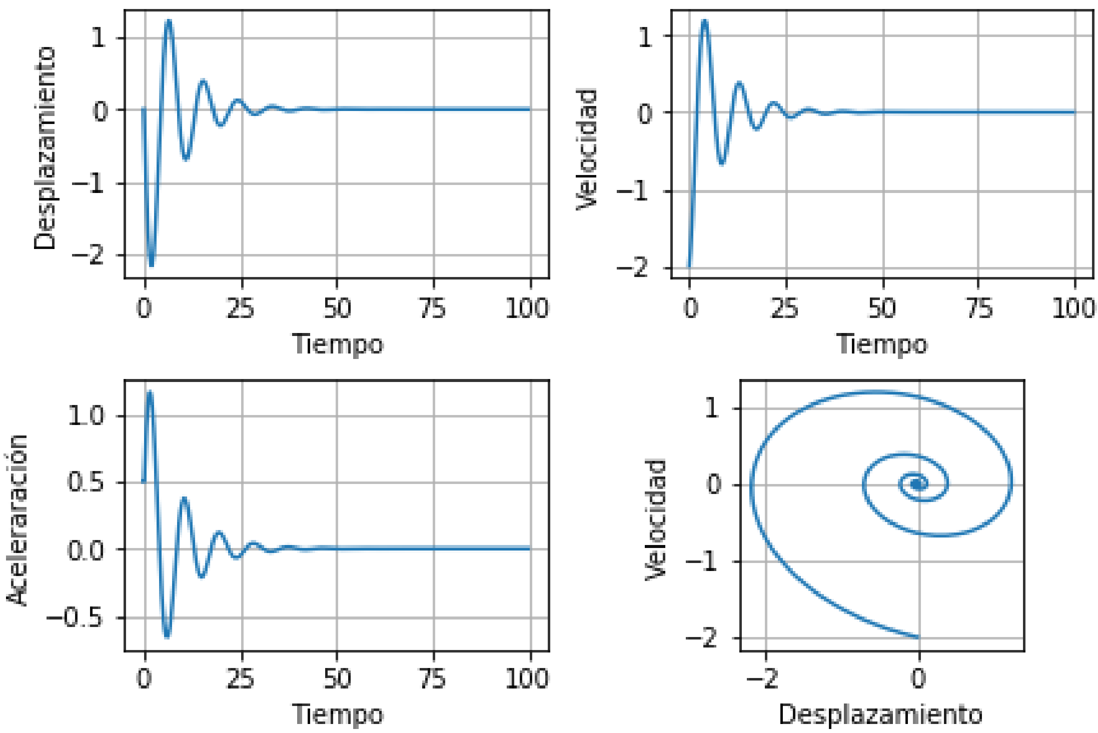
[9]:
```
a=np.array([-k*p[0]/m-c*v[0]/m+F0/m*math.cos(w*t)])
pt=[t]
pv=[v[0]]
px=[p[0]]
pa=[a[0]]
u =[p[0]**2*k*0.5]
K=[0.5*m*v[0]**2]
E = [ u[0] + K[0] ]
```

```python
for ts in np.arange(t+h, tfin, h):
    a =np.array([-k*p[0]/m-c*v[0]/m+F0/m*math.cos(w*ts)])
    v = v + a*h
    p  = p  + v*h
    u_ = p[0]**2*k/2
    u.append(u_)
    pt.append(ts)
    px.append(p[0])
    pv.append(v[0])
    pa.append(a[0])
    K_ = 0.5*m*v[0]**2
    K.append(K_)
    E.append( u_ + K_ )
```

(a) x − t, v − t, a − t y v − x en cuatro gráficos utilizando la instruccián subplot

[10]:

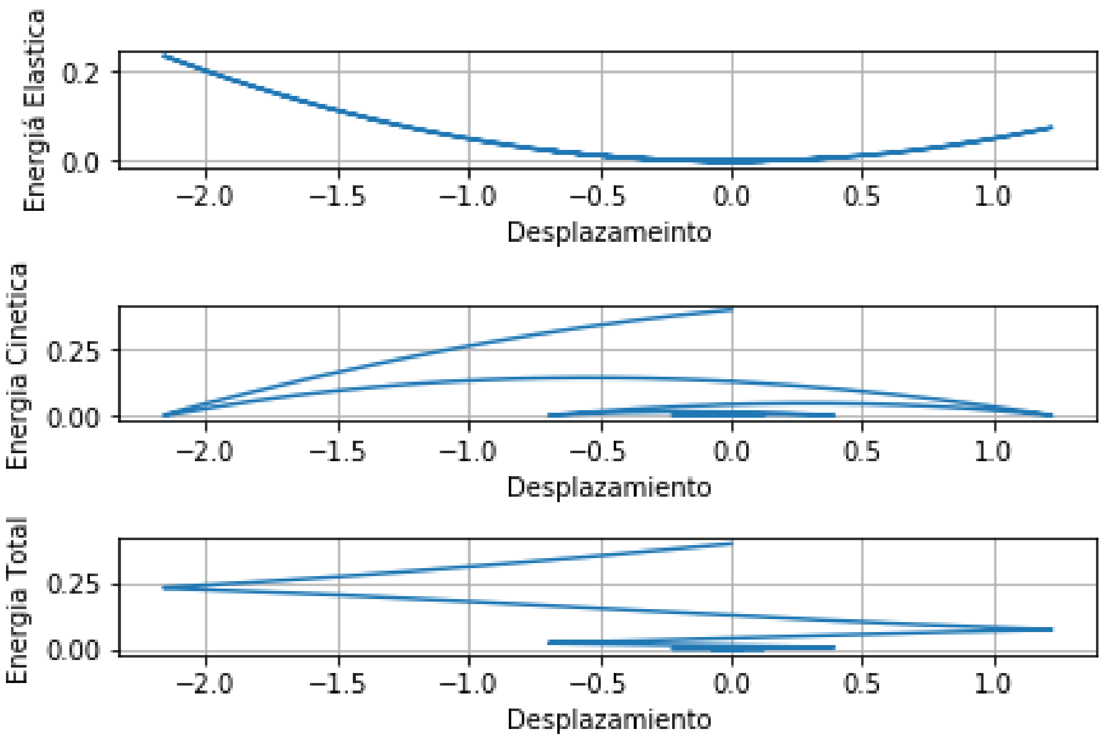
```python
fig, axes = plt.subplots(2, 2, constrained_layout=True)
axes[0][0].plot(pt, px)
axes[0][0].grid()
axes[0][0].set(xlabel = 'Tiempo', ylabel='Desplazamiento')

axes[0][1].plot(pt, pv)
axes[0][1].grid()
axes[0][1].set(xlabel = 'Tiempo', ylabel='Velocidad')

axes[1][0].plot(pt, pa)
axes[1][0].grid()
axes[1][0].set(xlabel = 'Tiempo', ylabel='Aceleraración')

axes[1][1].plot(px, pv)
axes[1][1].grid()
axes[1][1].set(xlabel = 'Desplazamiento', ylabel='Velocidad')
axes[1][1].set_aspect('equal')

#def animate(i):
#    line.set_data(pt[:i], px[:i])
#    line2.set_data(pt[:i], pv[:i])
#    line3.set_data(pt[:i], pa[:i])
#    line4.set_data(px[:i], pv[:i])
#    return [line, line2, line3, line4]

#ani = FuncAnimation(fig, animate, interval=1)
#HTML(ani.to_jshtml())
plt.show()
```



(b) U − x, K − x, E − x en un sólo gráfico

[11]:
```
fig, axes = plt.subplots(3, 1, constrained_layout=True)
axes[0].plot(px, u)
axes[0].grid()
axes[0].set(ylabel = 'Energiá Elastica' , xlabel='Desplazameinto')
axes[0]
axes[1].plot(px, K)
axes[1].grid()
axes[1].set(ylabel = 'Energia Cinetica', xlabel='Desplazamiento')

axes[2].plot(px, E)
axes[2].grid()
axes[2].set(ylabel = 'Energia Total', xlabel='Desplazamiento')

#def animate(i):
#    line.set_data(px[:i], u[:i])
#    line2.set_data(px[:i], K[:i])
#    line3.set_data(px[:i], E[:i])
#    return [line, line2, line3]

#ani = FuncAnimation(fig, animate, interval=1)
#HTML(ani.to_jshtml())
```

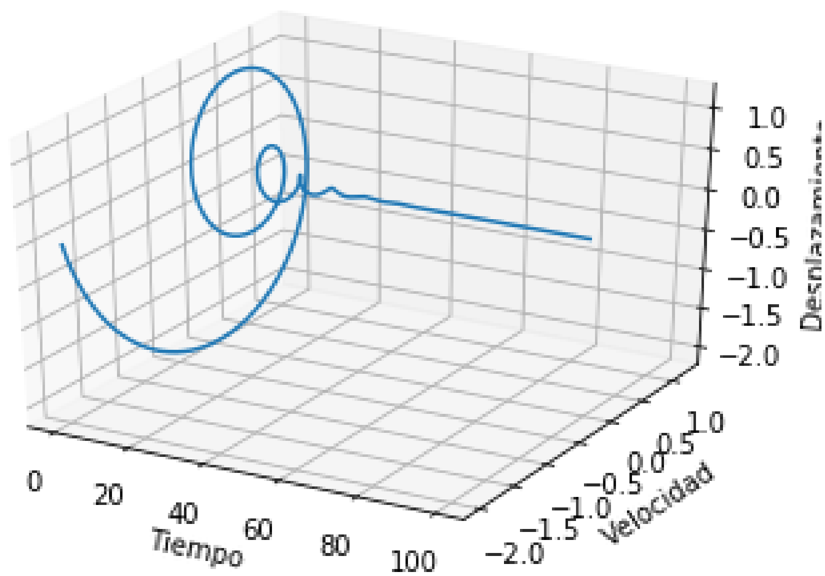
```
plt.show()
```



(c) v − x − t

[12]:
```
fig = plt.figure()
axes = plt.axes(projection='3d')
axes.plot3D(pt, pv, px)
axes.set(xlabel='Tiempo', ylabel='Velocidad',zlabel='Desplazamiento')
```

[12]:
```
[Text(0.5, 0, 'Desplazamiento'),
 Text(0.5, 0, 'Velocidad'),
 Text(0.5, 0, 'Tiempo')]
```



1.3 Sea un resorte con k = 0.1 N/m unido a una masa m = 0.2 kg se mueve horizontalmente en un medio c = 0.05 Ns/m donde hay fricción. Luego actúa una fuerza externa cuya amplitud F0 = 0.2 N y = 0.2 rad/s. Si las condiciones iniciales son x = −1 m y vx = 1 m/s, grafique

Declarado las variables del sistema

[13]:
```
h=0.1
k=0.1
m=0.2
c=0.05
F0=0.2
w=0.2
t=0
tfin=100
p=np.array([-1])
v=np.array([1])
```

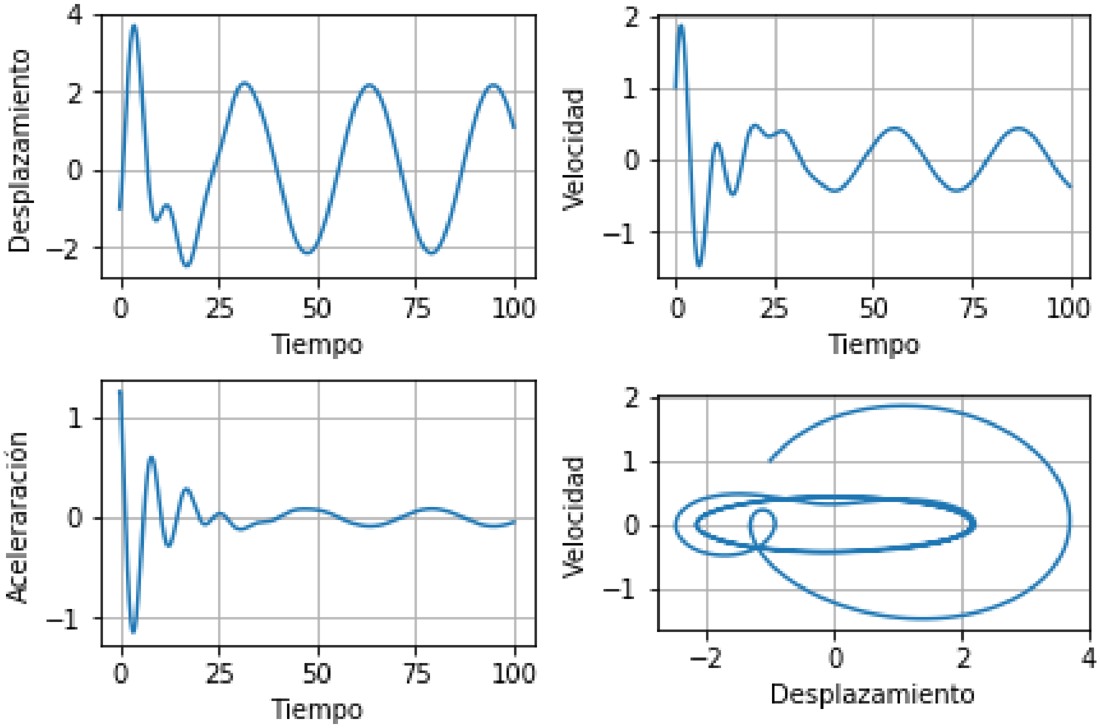
Ejecutando

[14]:
```
a=np.array([-k*p[0]/m-c*v[0]/m+F0/m*math.cos(w*t)])
pt=[t]
pv=[v[0]]
px=[p[0]]
pa=[a[0]]
u =[p[0]**2*k*0.5]
K=[0.5*m*v[0]**2]
```

```python
E = [ u[0] + K[0] ]
for ts in np.arange(t+h, tfin, h):
    a =np.array([-k*p[0]/m-c*v[0]/m+F0/m*math.cos(w*ts)])
    v = v + a*h
    p  = p  + v*h
    u_ = p[0]**2*k/2
    u.append(u_)
    pt.append(ts)
    px.append(p[0])
    pv.append(v[0])
    pa.append(a[0])
    K_ = 0.5*m*v[0]**2
    K.append(K_)
    E.append( u_ + K_ )
```

(a) x − t, v − t, a − t y v − x en cuatro gráficos utilizando la instruccián subplot

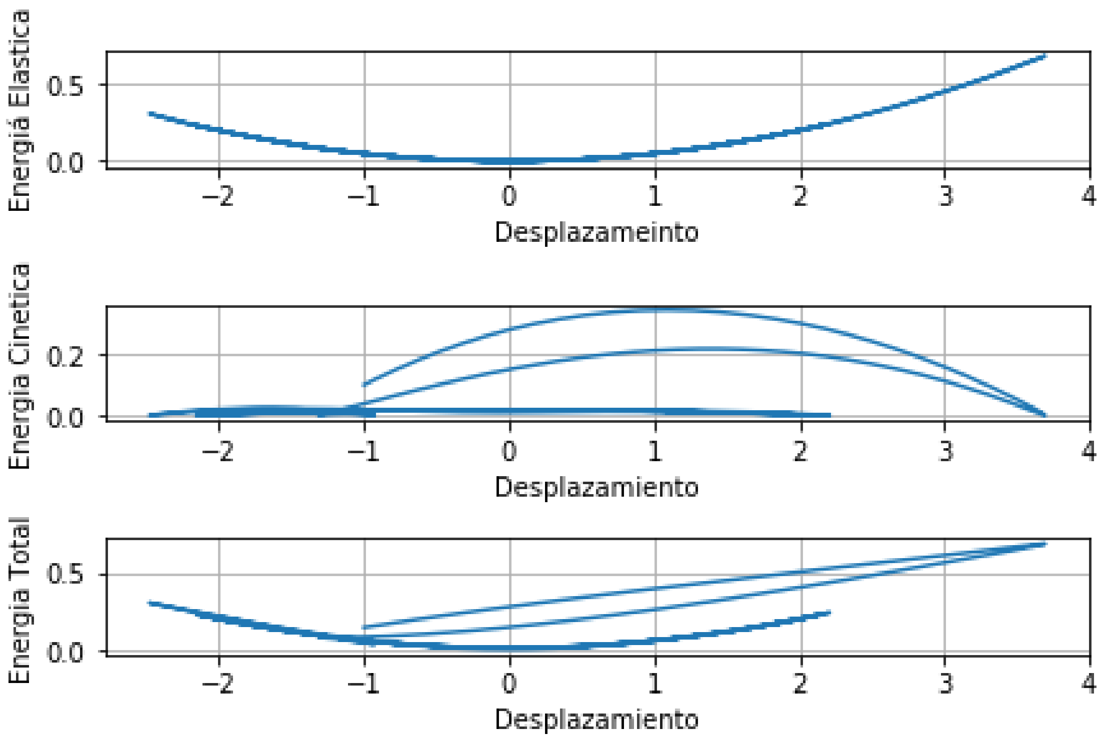
[15]:
```python
fig, axes = plt.subplots(2, 2, constrained_layout=True)
axes[0][0].plot(pt, px)
axes[0][0].grid()
axes[0][0].set(xlabel = 'Tiempo', ylabel='Desplazamiento')

axes[0][1].plot(pt, pv)
axes[0][1].grid()
axes[0][1].set(xlabel = 'Tiempo', ylabel='Velocidad')

axes[1][0].plot(pt, pa)
axes[1][0].grid()
axes[1][0].set(xlabel = 'Tiempo', ylabel='Aceleraración')

axes[1][1].plot(px, pv)
axes[1][1].grid()
axes[1][1].set(xlabel = 'Desplazamiento', ylabel='Velocidad')
axes[1][1].set_aspect('equal')

#def animate(i):
#    line.set_data(pt[:i], px[:i])
#    line2.set_data(pt[:i], pv[:i])
#    line3.set_data(pt[:i], pa[:i])
#    line4.set_data(px[:i], pv[:i])
#    return [line, line2, line3, line4]

#ani = FuncAnimation(fig, animate, interval=1)
#HTML(ani.to_jshtml())
plt.show()
```



(b) U − x, K − x, E − x en un sólo gráfico

[16]:
```
fig, axes = plt.subplots(3, 1, constrained_layout=True)
axes[0].plot(px, u)
axes[0].grid()
axes[0].set(ylabel = 'Energiá Elastica' , xlabel='Desplazameinto')
axes[0]
axes[1].plot(px, K)
axes[1].grid()
axes[1].set(ylabel = 'Energia Cinetica', xlabel='Desplazamiento')

axes[2].plot(px, E)
axes[2].grid()
axes[2].set(ylabel = 'Energia Total', xlabel='Desplazamiento')

#def animate(i):
#     line.set_data(px[:i], u[:i])
#     line2.set_data(px[:i], K[:i])
#     line3.set_data(px[:i], E[:i])
#     return [line, line2, line3]

#ani = FuncAnimation(fig, animate, interval=1)
#HTML(ani.to_jshtml())
```

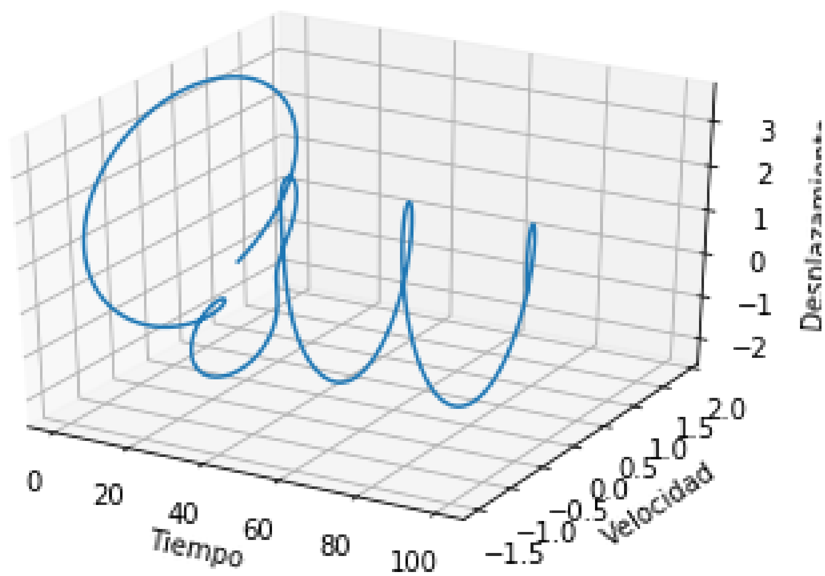
```
plt.show()
```



(c) v − x − t

[17]:
```
fig = plt.figure()
axes = plt.axes(projection='3d')
axes.plot3D(pt, pv, px)
axes.set(xlabel='Tiempo', ylabel='Velocidad',zlabel='Desplazamiento')
```

[17]:
```
[Text(0.5, 0, 'Desplazamiento'),
 Text(0.5, 0, 'Velocidad'),
 Text(0.5, 0, 'Tiempo')]
```



1.4 Grafique en una misma ventana el oscilador amortiguado y forzado con las mismas condiciones iniciales del problema anterior y ubique en que lugar del tiempo y espacio empieza a predominar la fuerza externa

Variables del Sistema

[18]:
```
h=0.1
k=0.1
m=0.2
c=0.05
w=0.2
t=0
tfin=100


F0=0.2
p0=np.array([-1])
v0=np.array([1])

F1=0
p1=np.array([-1])
v1=np.array([1])
```

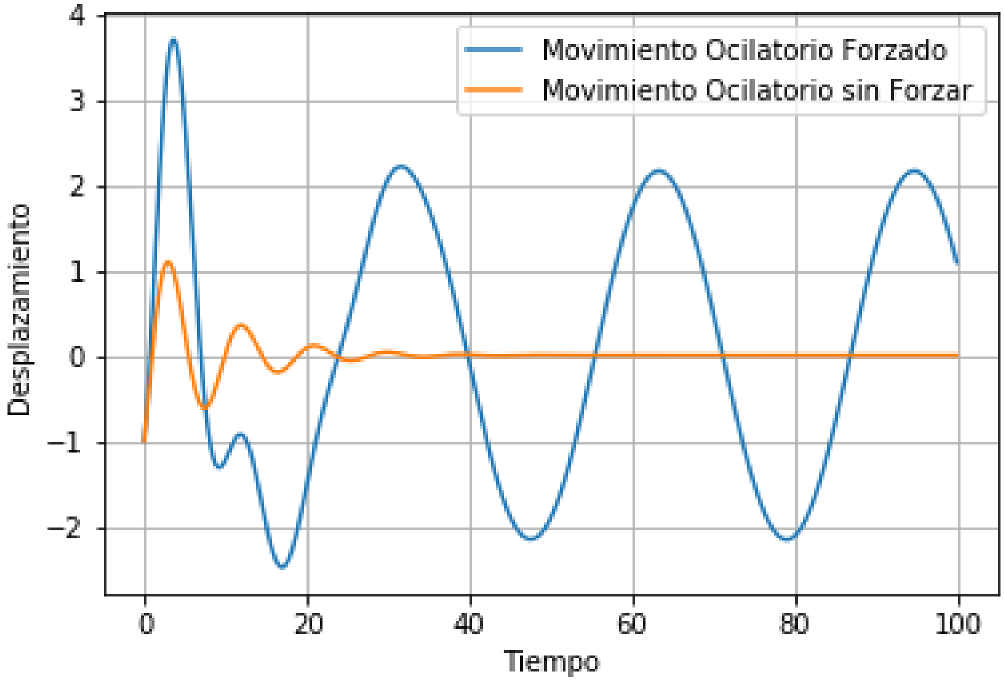
Ejecutando

[19]:
```
a0=np.array([-k*p0[0]/m-c*v0[0]/m+F0/m*math.cos(w*t)])
pv0=[v0[0]]
```

```
px0=[p0[0]]
pa0=[a0[0]]

a1=np.array([-k*p1[0]/m-c*v1[0]/m+F1/m*math.cos(w*t)])
pv1=[v1[0]]
px1=[p1[0]]
pa1=[a1[0]]

pt=[t]

for ts in np.arange(t+h, tfin, h):
    a0 = np.array([-k*p0[0]/m-c*v0[0]/m+F0/m*math.cos(w*ts)])
    v0 = v0 + a0*h
    p0 = p0 + v0*h

    a1 = np.array([-k*p1[0]/m-c*v1[0]/m+F1/m*math.cos(w*ts)])
    v1 = v1 + a1*h
    p1 = p1 + v1*h

    pt.append(ts)

    px0.append(p0[0])
    pv0.append(v0[0])
    pa0.append(a0[0])

    px1.append(p1[0])
    pv1.append(v1[0])
    pa1.append(a1[0])
```

Graficar

[20]:
```
fig = plt.Figure()
plt.plot(pt, px0, label='Movimiento Ocilatorio Forzado')
plt.plot(pt, px1, label='Movimiento Ocilatorio sin Forzar')
plt.xlabel('Tiempo')
plt.ylabel('Desplazamiento')
plt.grid()
plt.legend()
plt.show()
```



Movimiento Ocilatorio Forzado
Movimiento Ocilatorio sin Forzar
Desplazamiento
Tiempo
4
3
2
1
0
−1
−2
0
20
40
60
80
100