

# Amortiguado

March 23, 2017

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
In [2]: # constantes generales
g= 9.81 # [m/s^2]
```

## 0.1 Péndulo ideal

A efectos de presentar código Python útil para graficar la respuesta temporal de sistemas a continuación se muestra un ejemplo para la dinámica del péndulo ideal según  $\psi(t) = \psi_0 \cos(\omega t + \phi_0)$ .

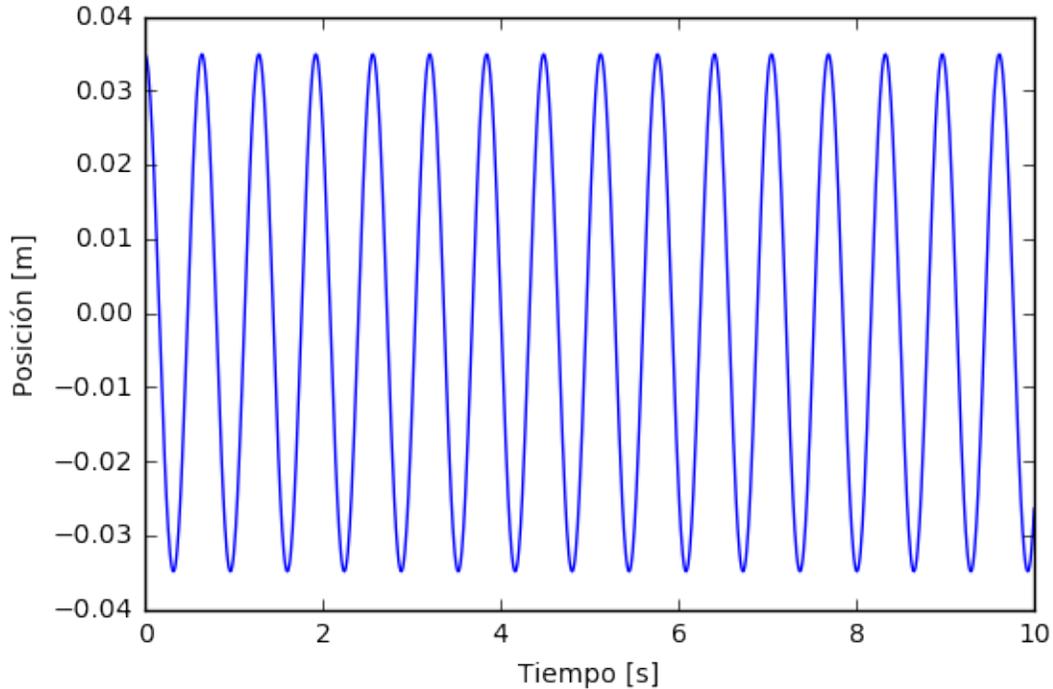
La amplitud en el tiempo inicial  $\psi_0$  se nota con la variable `pen_0`. La fase  $\phi_0$  como `phi_0`.

```
In [3]: # PARÁMETROS MODIFICABLES: condiciones iniciales péndulo
pen_0= 2 # [m] amplitud inicial
pen_0*= np.pi/180 # grados -> radianes
phi_0= 0 # fase inicial
l=1 # [m] longitud de la cuerda
```

```
In [4]: # Función que define la dinámica del péndulo
omega= g/l # frecuencia de oscilación
def pen(t):
    return pen_0 *np.cos(omega*t+phi_0)
```

```
In [5]: # graficación péndulo
tempo= np.linspace(0,10,int(1E3))
outPen= pen(tempo)
plt.plot(tempo,outPen)
plt.ylabel('Posición [m]')
plt.xlabel('Tiempo [s]')
```

```
Out[5]: <matplotlib.text.Text at 0x7fcad6968be0>
```



## 0.2 Oscilador sin resistencia

Si la posición de una masa  $x$  es la de equilibrio  $x_0$  mas un apartamiento  $\psi$ , y la fuerza  $\vec{F} = m\ddot{x}\hat{x}$  entonces  $\vec{F} = m\ddot{\psi}$ .

Un resorte de constante elástica  $k$  provee la fuerza, por tanto  $-k\psi = m\ddot{\psi}$ . De aquí se despeja  $\ddot{\psi} = \omega\psi$  con  $\omega = \sqrt{\frac{k}{m}}$ .

Propuesta la solución  $\psi(t) = Ce^{\lambda t}$  queda  $(\lambda^2 + \omega^2)Ce^{\lambda t} = 0$ . Luego  $\lambda_{1,2} = \pm i\omega$  y entonces  $\psi(t) = C_1e^{i\omega t} + C_2e^{-i\omega t}$ .

Haciendo uso de la igualdad de Euler  $e^{\pm i\theta} = \cos \theta + i \sin \theta$ , agrupamos para obtener  $\psi(t) = (C_1 + C_2) \cos \omega t + i(C_1 - C_2) \sin \omega t$ .

Este conjunto de constantes puede determinarse usando los datos a tiempo inicial  $\psi(0) = C_1 + C_2$  y  $\dot{\psi}(0) = i(C_1 - C_2)\omega$ .

Por tanto

$$\psi(t) = \psi(0) \cos \omega t + \frac{\dot{\psi}(0)}{\omega} \sin \omega t.$$

Esto puede ser escrito con una fase  $\phi = \arctan\left(\frac{\dot{\psi}(0)}{\omega\psi(0)}\right)$  y amplitud  $U = \sqrt{\psi^2(0) + \left(\frac{\dot{\psi}(0)}{\omega}\right)^2}$ , quedando

$$\psi(t) = U \cos(\omega t + \phi).$$

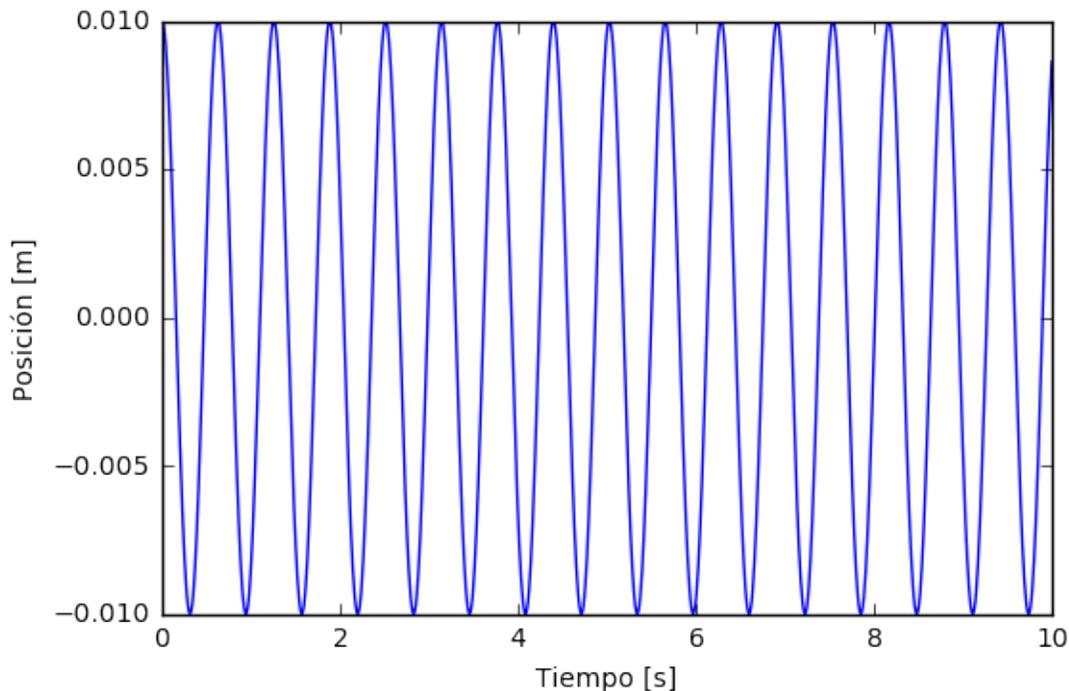
```
In [6]: # PARÁMETROS MODIFICABLES: condiciones oscilador
m= 1E-1 # [kg] masa oscilador
k= 1 # [N/m] constante elástica resorte
```

```
psi_0= 1E-2 # [m] amplitud oscilación a tiempo inicial
ppsi_0= 0E-2 # [m/s] velocidad oscilación a tiempo inicial
```

```
In [7]: # Función que define la dinámica del oscilador
omega= k/m
def psiOsc(t):
    return psi_0 * (np.cos(omega*t)+ (ppsi_0/omega)* np.sin(omega* t) )
```

```
In [8]: # graficación oscilador
tiempo= 10 # [s]
tempo= np.linspace(0,10,int(1E3))
outOsc= psiOsc(tempo)
plt.axis([0,tiempo,-psi_0,psi_0])
plt.plot(tempo,outOsc)
plt.ylabel('Posición [m]')
plt.xlabel('Tiempo [s]')
```

Out[8]: <matplotlib.text.Text at 0x7fcad6980c88>



### 0.3 Oscilador amortiguado

Se agrega una fuerza opuesta y proporcional a la velocidad  $-c\dot{\psi}$ , así la 2.a ley queda  $m\ddot{\psi} = -c\dot{\psi} - k\psi$  y entonces  $\ddot{\psi} + \frac{c}{m}\dot{\psi} + \omega\psi = 0$ .

Usualmente se denomina  $\Gamma = \frac{c}{m}$  así  $\lambda_{1,2} = -\frac{\Gamma}{2} \pm \sqrt{\left(\frac{\Gamma}{2}\right)^2 - \omega^2}$ . De acuerdo a si el valor dentro de la raíz  $\omega' = \sqrt{\left(\frac{\Gamma}{2}\right)^2 - \omega^2}$  es negativo, positivo o nulo se presentan distintas dinámicas.

### 0.3.1 oscilador subamortiguado

Si el amortiguamiento es débil,  $\frac{\Gamma}{2} < \omega$ , entonces subsiste el comportamiento oscilatorio,  $\omega' = i\sqrt{\left(\frac{\Gamma}{2}\right)^2 - \omega^2}$  y la dinámica responde a  $\psi(t) = A_1 e^{-\frac{\Gamma}{2}t} \cos |\omega'|t + A_2 e^{-\frac{\Gamma}{2}t} \sin |\omega'|t$ .

Contemplando las condiciones iniciales esta solución usualmente se escribe

$$\psi(t) = e^{-\frac{\Gamma}{2}t} \left( \psi(0) \cos \omega't + \frac{\dot{\psi}(0) + \frac{\Gamma}{2}\psi(0)}{\omega'} \sin \omega't \right),$$

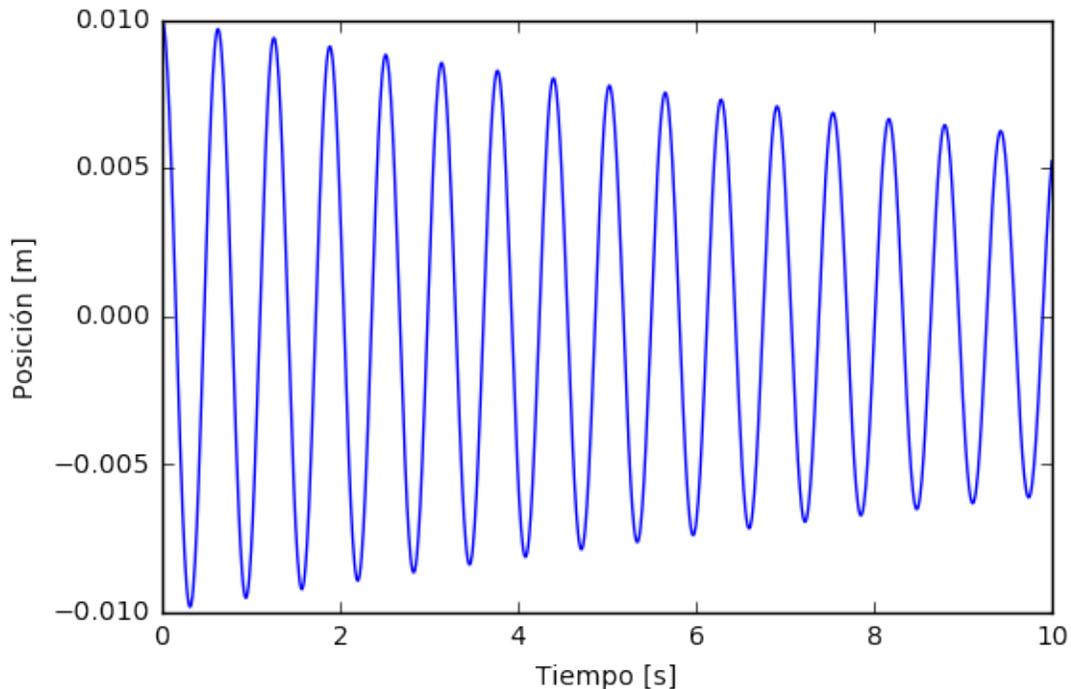
donde aquí  $\omega'$  se refiere al valor absoluto de la raíz.

```
In [9]: # PARÁMETROS MODIFICABLES: condiciones adicionales subamortiguado
c= 1E-2 # [kg/s^2]
gamma= c/m # [1/s]
omegaPrima= np.sqrt(np.abs((gamma/2)**2- omega**2))
```

```
In [10]: # dinámica del subamortiguado
def psiOscSub(t):
    return np.exp((-gamma/2)*t) * ( psi_0 * np.cos(omegaPrima*t) + ((psi_0_0
```

```
In [11]: # graficación subamortiguado
outOscSub= psiOscSub(tiempo)
plt.axis([0,tiempo,-psi_0,psi_0])
plt.plot(tiempo,outOscSub)
plt.ylabel('Posición [m]')
plt.xlabel('Tiempo [s]')
```

```
Out[11]: <matplotlib.text.Text at 0x7fcad68133c8>
```



### 0.3.2 Oscilador sobreamortiguado

Si por el contrario el amortiguamiento prima sobre la oscilación,  $\frac{\Gamma}{2} > \omega$ , los valores de  $\omega'$  son reales. Basta recordar que  $e^{\pm\theta} = \cosh \theta \pm \sinh \theta$  para arribar a la solución,

$$\psi(t) = e^{-\frac{\Gamma}{2}t} \left( \psi(0) \cosh \omega' t + \frac{\dot{\psi}(0) + \frac{\gamma}{2}\psi(0)}{\omega'} \sinh \omega' t \right),$$

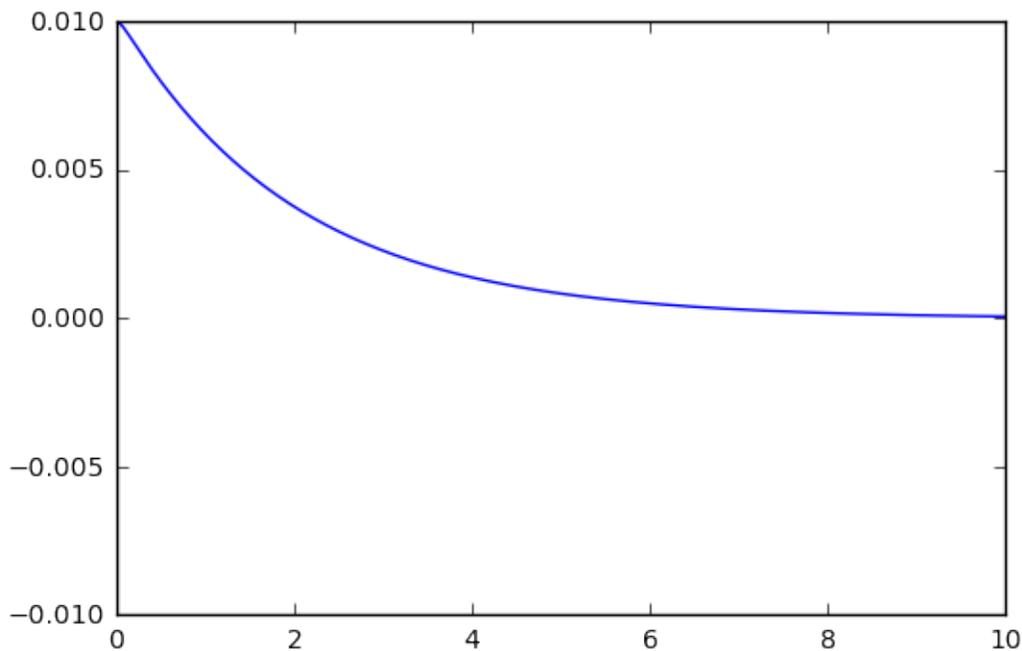
ahora con  $\omega'$  real.

```
In [12]: # PARÁMETROS MODIFICABLES: condiciones adicionales sobremortiguado
c= 2.1E0# [kg/s^2]
gamma= c/m # [1/s]
```

```
In [13]: # dinámica del sobremortiguado
def psiOscSob(t):
    return np.exp((-gamma/2)*t) * ( psi_0 * np.cosh(omegaPrima*t) + ((psi_0
```

```
In [14]: # graficación sobreamortiguado
outOscSob= psiOscSob(tiempo)
plt.axis([0,tiempo,-psi_0, psi_0])
plt.plot(tiempo,outOscSob)
```

```
Out[14]: [<matplotlib.lines.Line2D at 0x7fcad68c4400>]
```



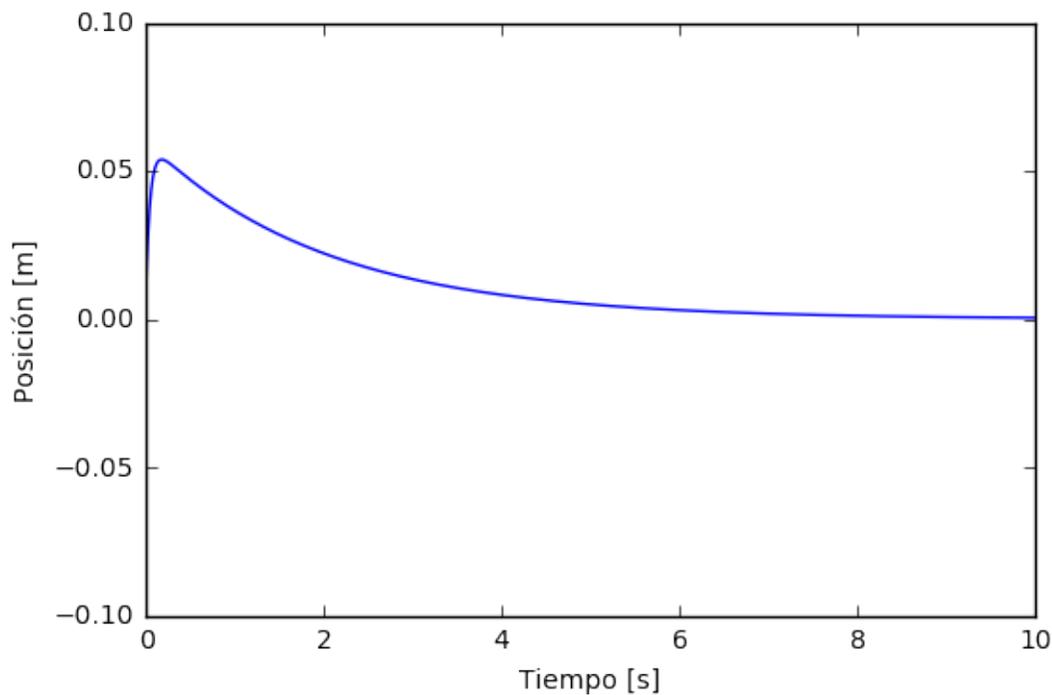
### Sobreamortiguado con velocidad inicial

```
In [15]: # PARÁMETROS MODIFICABLES: condiciones adicionales subamortiguado - con v
c= 2.1E0# [kg/s^2]
gamma= c/m # [1/s]
ppsi_0= 1E0 # [m/s] velocidad inicial
```

```
In [16]: # dinámica sobreamortiguado
def psiOscSobVel(t):
    return np.exp((-gamma/2)*t)* ( psi_0 * np.cosh(omegaPrima*t)+ ((ppsi_0
```

```
In [17]: # graficación sobreamortiguado
outOscSobVel= psiOscSobVel(tiempo)
plt.axis([0,tiempo,-10*psi_0, 10*psi_0])
plt.plot(tiempo,outOscSobVel)
plt.ylabel('Posición [m]')
plt.xlabel('Tiempo [s]')
```

```
Out[17]: <matplotlib.text.Text at 0x7fcad671da20>
```



### 0.3.3 Oscilador críticamente amortiguado

El caso en que  $\frac{\Gamma}{2} = \omega$  hace que  $\lambda_1 = \lambda_2 = -\omega$ , así  $\psi(t) = (A_1 + A_2t)e^{-\omega t}$ . La dinámica termina siendo dada por

$$\psi(t) = \left[ \psi(0) + \left( \dot{\psi}(0) + \omega\psi(0)t \right) \right] e^{-\omega t}$$

```
In [18]: # PARÁMETROS MODIFICABLES: condiciones adicionales críticamente amortiguado
gamma= 2* omega # [1/s]
```

```
In [19]: # dinámica críticamente amortiguado
def psiOscCrit(t):
    return (psi_0+ (ppsi_0+ omega* psi_0)* t)* np.exp(-omega* t)
```

```
In [20]: # graficación críticamente amortiguado
outOscCrit= psiOscCrit(tiempo)
plt.axis([0,tiempo,-psi_0, psi_0])
plt.plot(tiempo,outOscCrit)
plt.ylabel('Posición [m]')
plt.xlabel('Tiempo [s]')
```

```
Out[20]: <matplotlib.text.Text at 0x7fcad66e2da0>
```

