Ejemplo01-6-1Shi-es

April 25, 2025

1 Ejemplo de perfil de leva con movimiento armónico simple

1.1 Seguidor alternante de rodillo

El seguidor de movimiento alternativo, radial y de rodillo, de una leva de placa debe subir 2 pulg con movimiento armónico simple en 180° de rotación de la leva, y retornar con movimiento armónico simple en los 180° restantes. Si el radio del rodillo es de 0.375 pulg y el del círculo primario es de 2 pulg constrúyase el diagrama de desplazamientos, la curva de paso y el perfil de la leva para una rotación de ésta en el mismo sentido que el movimiento de las manecillas del reloj.

1.2 Librerías

Se ha de utilizar la librería `DiskCamMechanismLibrary`, la cual se puede encontrar en este link, asi como `matplotlib` y `numpy`.

[4]:
```
from DiskCamMechanismLibrary import PDCamRollerFollower
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.animation import FuncAnimation
import matplotlib.animation as animation

import matplotlib as mpl
mpl.rcParams['figure.dpi'] = 300
```

1.3 Movimiento armónico simple

Las ecuaciones de movimiento armónico simple para subida y descenso del seguidor se muestran a continuación:

$$
\begin{aligned}
y &= \frac{L}{2}\left(1 - \cos\theta\right) \\
y' &= \frac{L}{2}\sin\theta \\
y'' &= \frac{L}{2}\cos\theta
\end{aligned}
$$

donde L es el desplazamiento máximo que alcanza el seguidor y θ es la posición angular de la leva.

Se agrega el siguiente código de python para calcular, desplazamiento, velocidad y aceleración del seguidor.

[6]:
```
def MovArmonicoSimple(th,L):
    y = 0.5*L*(1-np.cos(th))
    yp = 0.5*L*np.sin(th)
    ypp = 0.5*L*np.cos(th)
    return y,yp,ypp
```

1.4 Datos del problema:

$$L = 2 \text{ pulg}$$
$$r_{\text{primario}} = 2 \text{ pulg}$$
$$r_{\text{rodillo}} = 0.375 \text{ pulg}$$

[8]:
```
L=2
Rbase=2 #radio primario
rd=0.375 #radio rodillo
Rbroca=3/16 # Radio de la broca (centro de la leva)
excentricidad = 0.0
posAngularSeguidor = np.pi/2 # posicion angular del seguidor en radianes
theta = np.linspace(0,1,500)*2*np.pi # barrido angular de cero a 2pi radianes
# calcular desplazamiento, velocidad, aceleracion
y,yp,ypp = MovArmonicoSimple(theta,L)

# Agrupar datos en diccionario, para otros parametros consultar la␣
 ↪documentacion de DiskCamMechanismLibrary
CamData={'theta':theta,
         'y':y,
         'yp':yp,
         'ypp':ypp,
         'Rbase':Rbase,
         'Rhole':Rbroca,
         'epsilon':excentricidad,
         'FollowerAng':posAngularSeguidor,
         'Followerwidth': 4/16,
         'turn_direction':'clockwise',
         'Rroller':rd
        }
```

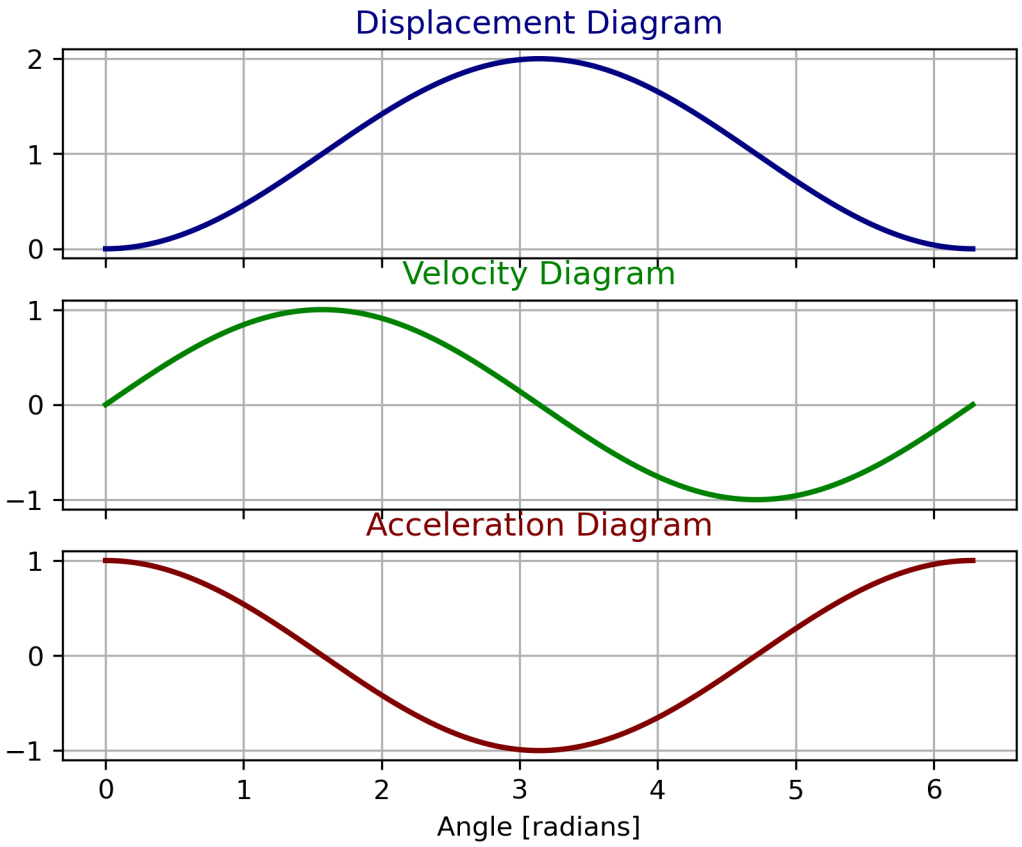
1.5 Calcular el perfil de la Leva

[10]:
```
Leva=PDCamRollerFollower(**CamData)
```

1.6 Diagrama de movimiento

[12]:
```
figMD=plt.figure()
Leva.PlotMotionDiagram(figMD)
```

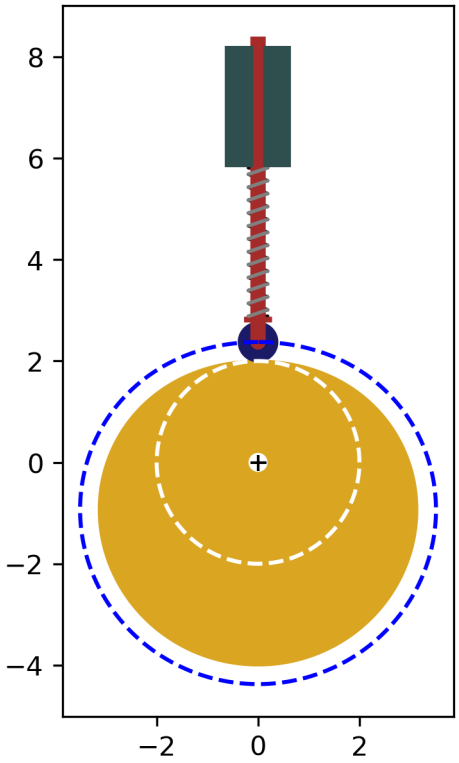
[12]:
```
array([<Axes: title={'center': 'Displacement Diagram'}>,
       <Axes: title={'center': 'Velocity Diagram'}>,
       <Axes: title={'center': 'Acceleration Diagram'}, xlabel='Angle
[radians]'>],
      dtype=object)
```



1.7 Graficar el perfil de la leva

[14]:
```
figPCam=plt.figure()
Leva.PlotCamRollerFollower(figPCam)
```

[14]:
```
<Axes: >
```



Los datos de las coordenadas del perfil se encuentran en los atributos `Leva.Xp` y `Leva.Yp`:

[16]:
```
print(Leva.Xp[0:10]) # Just a few data
print(Leva.Yp[0:10]) # Just a few data
```

```
[ 7.65404249e-17 -2.71713421e-02 -5.43432515e-02 -8.15162928e-02
 -1.08691026e-01 -1.35868003e-01 -1.63047766e-01 -1.90230845e-01
 -2.17417755e-01 -2.44608993e-01]
[2.         1.99990096 1.99960379 1.99910835 1.99841441 1.99752165
 1.99642963 1.99513786 1.99364574 1.99195256]
```

Los datos de la curva de paso se encuentran en los atributos `Leva.Xr` y `Leva.Yr`:

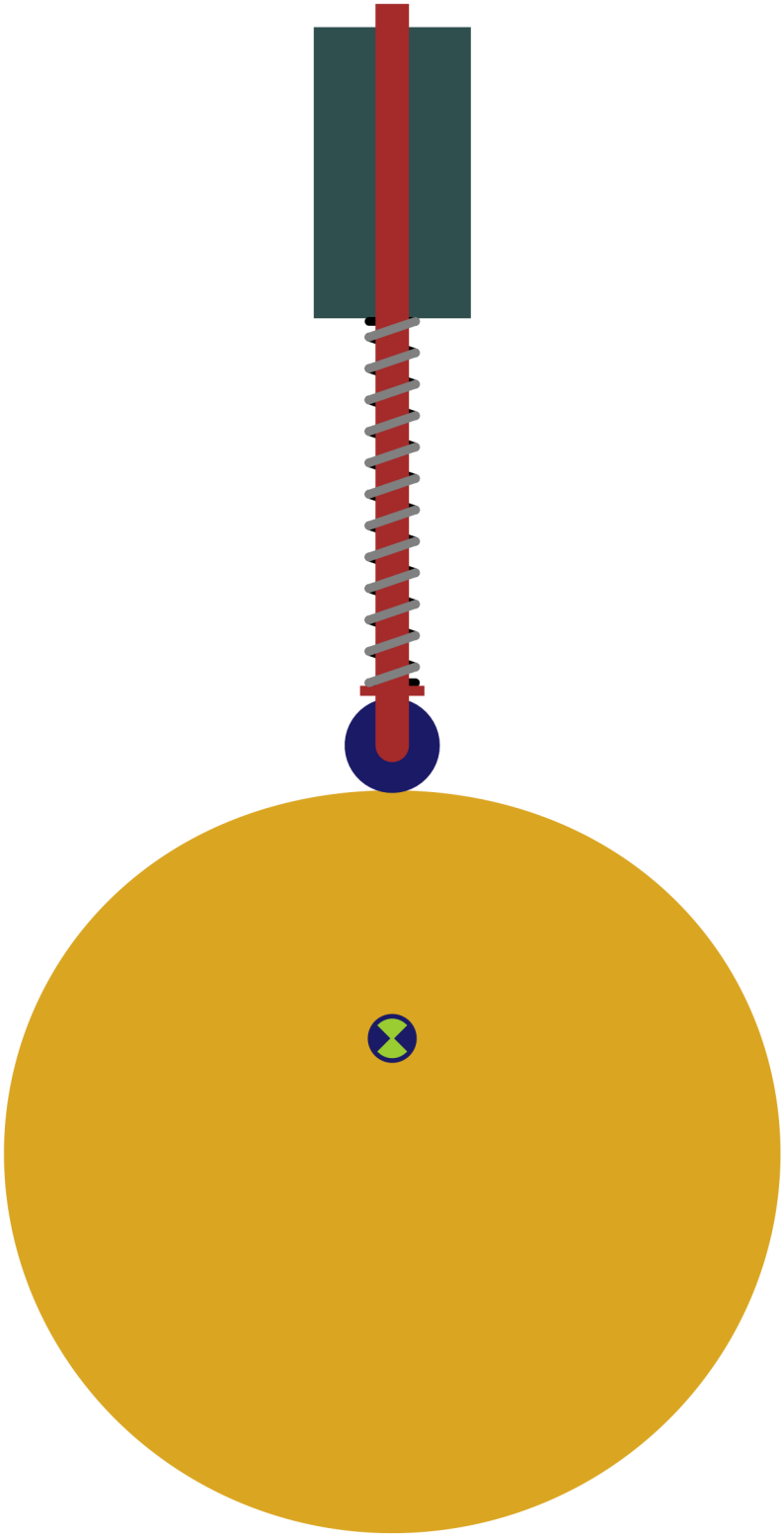
[18]:
```
print(Leva.Xr[0:10]) # Just a few data
print(Leva.Yr[0:10]) # Just a few data
```

```
[ 1.45426807e-16 -2.99051480e-02 -5.98115427e-02 -8.97204269e-02
 -1.19633035e-01 -1.49550591e-01 -1.79474300e-01 -2.09405350e-01
 -2.39344903e-01 -2.69294096e-01]
[2.375      2.37489099 2.37456392 2.3740186  2.37325474 2.37227194
 2.37106969 2.36964733 2.36800412 2.3661392 ]
```

1.8 Animación de la leva

[20]:
```
fig, ax=plt.subplots()
ax.set_axis_off()
init_func=Leva.initAnim(ax),
dpi=100
width = 1920/dpi
hight = 1080/dpi
fig.set_size_inches(width,hight)

anim3 = FuncAnimation(fig, Leva, frames=np.arange(1000),
                      interval=100, blit=False)
plt.show()
```



1.9 Guardar la animación de la leva en un archivo

[ ]:
```
writer = animation.writers['ffmpeg'](fps=30)
anim3.save('Leva01.mp4',writer=writer,dpi=dpi)
```

1.10 Código completo

[ ]:
```
#%% Librerias
from DiskCamMechanismLibrary import PDCamRollerFollower
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.animation import FuncAnimation
import matplotlib.animation as animation

# %% Movimiento armonico simple
def MovArmonicoSimple(th,L):
    y = 0.5*L*(1-np.cos(th))
    yp = 0.5*L*np.sin(th)
    ypp = 0.5*L*np.cos(th)
    return y,yp,ypp

# %% Datos del problema
L=2
Rbase=2 #radio primario
rd=0.375 #radio rodillo
Rbroca=3/16 # Radio de la broca (centro de la leva)
excentricidad = 0.0
posAngularSeguidor = np.pi/2 # posicion angular del seguidor en radianes
theta = np.linspace(0,1,500)*2*np.pi # barrido angular de cero a 2pi radianes
# calcular desplazamiento, velocidad, aceleracion
y,yp,ypp = MovArmonicoSimple(theta,L)

# Agrupar datos en diccionario, para otros parametros consultar la␣
 ↪documentacion de DiskCamMechanismLibrary
CamData={'theta':theta,
         'y':y,
         'yp':yp,
         'ypp':ypp,
         'Rbase':Rbase,
         'Rhole':Rbroca,
         'epsilon':excentricidad,
         'FollowerAng':posAngularSeguidor,
         'Followerwidth': 4/16,
         'turn_direction':'clockwise',
         'Rroller':rd
        }
```

```
#%% Calcular el perfil de la Leva
Leva=PDCamRollerFollower(**CamData)

#%% Diagrama de movimiento
figMD=plt.figure()
Leva.PlotMotionDiagram(figMD)

#%% Graficar el perfil de la leva
figPCam=plt.figure()
Leva.PlotCamRollerFollower(figPCam)

#%% Animación de la leva
fig, ax=plt.subplots()
ax.set_axis_off()
init_func=Leva.initAnim(ax),
dpi=100
width = 1920/dpi
hight = 1080/dpi
fig.set_size_inches(width,hight)

anim3 = FuncAnimation(fig, Leva, frames=np.arange(1000),
                      interval=100, blit=False)
plt.show()

#%% Guardar la animación de la leva en un archivo
writer = animation.writers['ffmpeg'](fps=30)
anim3.save('Leva01.mp4',writer=writer,dpi=dpi)
```