Estudio de la oscilación de un péndulo simple

Víctor Mira Ramírez

15 de febrero de 2024

Profesora: María Reyes Calvo Urbina
Lugar: Universidad de Alicante

Índice

1. Introducción y Datos

1.1. Abstract

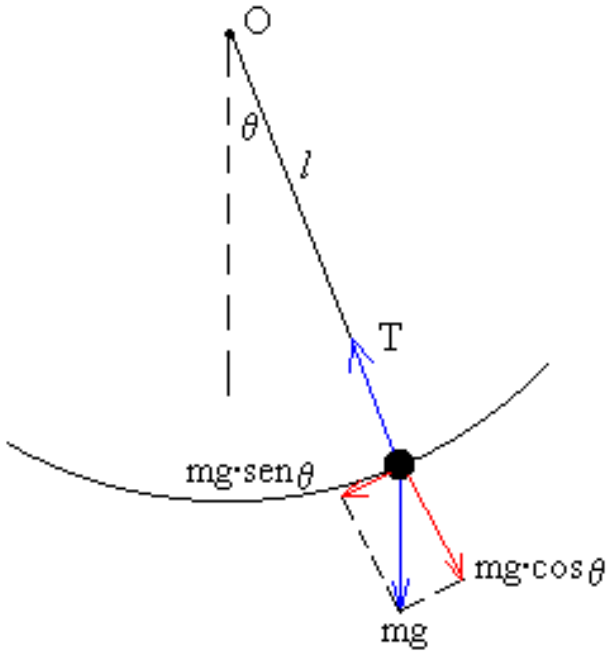
Este experimento estudia el movimiento de un péndulo simple: el ángulo que describe, su velocidad angular, la tensión en la cuerda y el periodo. Analizaremos el caso con rozamiento y el caso sin él. La práctica incluye gráficas realizadas con códigos en python que representan el comportamiento del péndulo para varios ángulos iniciales y en diferentes intervalos de tiempo.

1.2. Datos Empíricos

Masa del péndulo	1kg
Longitud de la cuerda	1m

1.3. Diagramas y formulaciones matemáticas

El siguiente diagrama muestra un péndulo simple y las fuerzas que actúan sobre él. Podemos descomponer el peso en una componente radial ($P_r = mg\cos(\theta)$) y en una componente tangencial ($P_t = mg\sin(\theta)$)



Estudiamos el caso tanto en sentido radial, $T - mg\cos(\theta) = ma_c = ml\omega^2$, donde a_c es la aceleración centrípeta ($\frac{v^2}{r}$), quedándonos la ecuación para el periodo:

$$\boxed{T = mg\cos(\theta) + ml\omega^2}$$

y el caso en sentido tangencial, $-mg\sin(\theta) = ma_t = ml\alpha = ml\frac{d^2\theta}{dt^2}$, donde a_t es la aceleración tangencial ($\frac{dv}{dt}$), quedándonos la ecuación para el ángulo:

$$\boxed{\frac{d^2\theta}{dt^2} + \frac{g}{l}\sin(\theta) = 0}$$

Con esta ecuación diferencial, conseguimos el ángulo en función al tiempo, $\omega(t)$, a partir de la cual podemos averiguar la velocidad angular, $\omega(t) = \frac{d\theta}{dt}$ y la aceleración angular $\alpha(t) = \frac{d^2\theta}{dt^2}$
Resolver esta ecuación diferencial es algo complicado, pero existe un método rápido para obtener soluciones para ángulos pequeños.

1.3.1. Ángulos Pequeños

La ecuación $\frac{d^2\theta}{dt^2} + \frac{g}{l}\sin(\theta) = 0$ puede simplificarse si, considerando ángulos pequeños, $\sin(\theta) = 0$, obteniendo la fórmula $\frac{d^2\theta}{dt^2} + \frac{g}{l}\theta) = 0$, que tiene una solución exacta,

$$\boxed{\theta = \theta_0 \cos(\gamma t) \qquad \theta_0 = \theta(t=0)}$$

donde γ es la frecuencia angular, $\gamma^2 = \frac{g}{l} \quad \longrightarrow \quad \gamma = \sqrt{\frac{g}{l}}$

y como $\omega = \frac{d\theta}{dt}$:

$$\boxed{\omega = \theta_0 \gamma \left(-\sin(\gamma t)\right)}$$

1.3.2. Solución Numérica

Discretizando el tiempo y usando el método de Euler para pasos de Δt pequeños,
$\omega = \frac{d\theta}{dt} \quad \longrightarrow \quad \frac{d^2\theta}{dt^2} + \frac{g}{l}\sin(\theta) = 0 \quad \longrightarrow \quad \frac{d\omega}{dt} + \frac{g}{l}\sin(\theta) = 0 \quad \longrightarrow \quad \Delta\omega = \left[\frac{-g}{l}\sin(\theta)\right]\Delta t$
y por tanto, obtenemos para la velocidad angular la ecuación:

$$\boxed{\omega_{i+1} = \omega_i - \left(\frac{g}{l}\sin(\theta_i)\right)\Delta t}$$

y como $\omega = \frac{d\theta}{dt} \quad \rightarrow \quad \Delta\theta = \omega\Delta t$, obtenemos para el ángulo la ecuación:

$$\boxed{\theta_{i+1} = \theta_i + \omega_i \Delta t}$$

1.3.3. Energías

Trivialmente sabemos que la Energía Mecánica será la suma de la Energía Potencial y de la Energía Cinética. En el caso sin rozamiento, habrá una conservación de la Energía Mecánica, ya que no existirá ninguna energía disipatoria que la reduzca. Ese es el caso con rozamiento, donde el rozamiento irá poco a poco reduciendo nuestra energía mecánica total hasta llegar al punto donde no oscilará.

Energía Potencial

Sabiendo que la fórmula de la Energía Potencial es $E_p = -mgh$ y que la altura del péndulo en cada momento viene dada por $\cos(\theta)l$, obtenemos para la Energía Potencial de nuestro péndulo la ecuación:

$$\boxed{E_p = -mgl\cos(\theta)}$$

Energía Cinética

Sabiendo que la fórmula de la Energía Cinética es $E_c = \frac{1}{2}mv^2$ y que la velocidad tangencial se relaciona con la angular con la fórmula $v = \omega r$, donde r será la longitud de la cuerda del péndulo l, obtenemos para la Energía Cinética de nuestro péndulo la ecuación

$$\boxed{E_c = \frac{1}{2}m\omega^2 l^2}$$

1.3.4. Péndulo con Fricción

La fuerza de rozamiento viene definida por la ecuación $F_r = -k\omega$, que debemos agregar a nuestra ecuación diferencial inicial, quedándonos la nueva ecuación diferencial:

$$-mg\sin(\theta) - k\omega = ml\alpha = ml\frac{d^2\theta}{dt^2}$$

$$\boxed{\frac{d^2\theta}{dt^2} + \frac{g}{l}\sin(\theta) - \frac{k}{ml}\frac{d\theta}{dt} = 0}$$

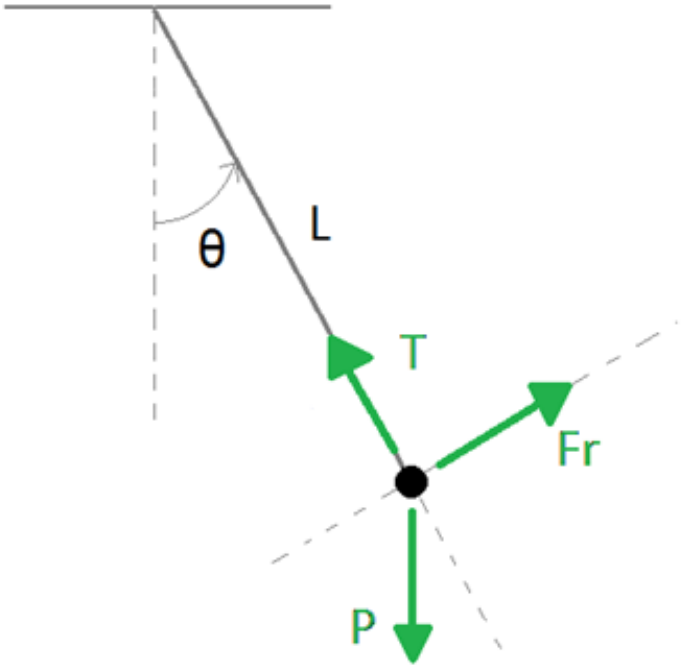
Simplemente hemos añadido el término $-\frac{k}{ml}\frac{d\theta}{dt} = 0$ a nuestra ecuación diferencial.

Actualizando nuestra solución numérica tenemos las siguientes ecuaciones para el caso con rozamiento:

$$\boxed{\omega_{i+1} = \omega_i + \left[\frac{-g}{l}\sin(\theta_i) + \frac{k}{ml}\omega_i\right]\Delta t}$$

$$\boxed{\theta_{i+1} = \theta_i + \omega_i\Delta t}$$

El siguiente diagrama representa el caso estudiado, donde T es la tensión y F_r es la fuerza de rozamiento que hemos añadido.

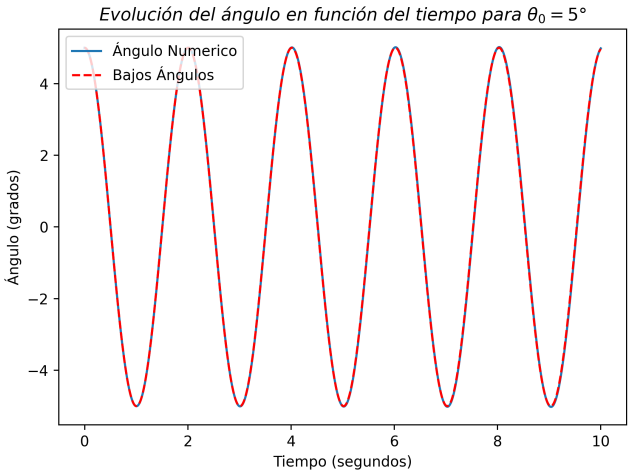


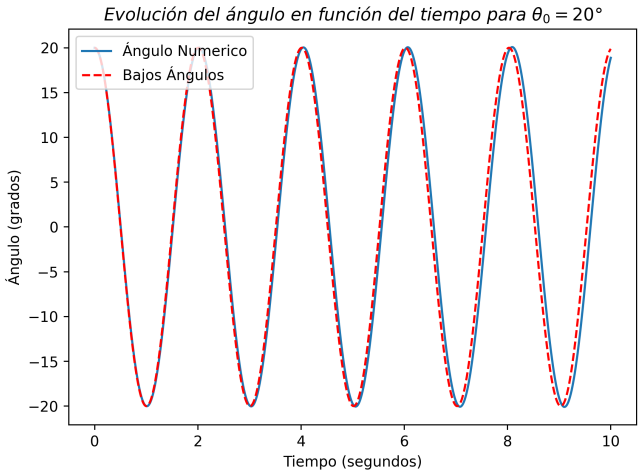
2. Parte a) - Péndulo sin Fricción

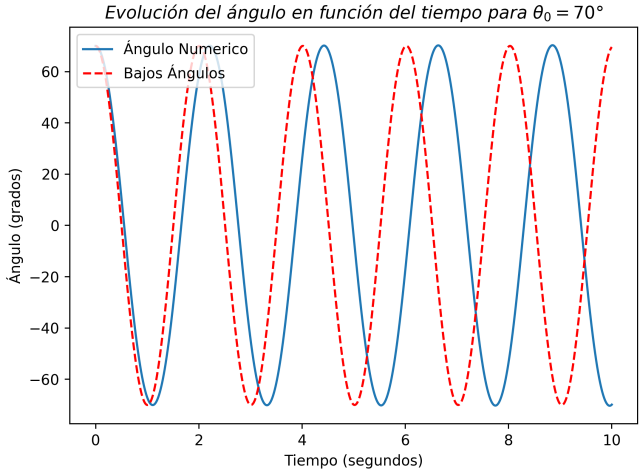
En esta sección estudiaremos las gráficas del movimiento del péndulo usando dos soluciones diferentes, una solución numérica de la ecuación diferencial, y otra para ángulos pequeños. Observaremos como para ángulos mayores a 15 grados, la solución numérica es la única que nos proporciona unos resultados más fiables.

2.1. Cálculo del Ángulo, Velocidad Angular y la Tensión

2.1.1. Evolución del Ángulo en función del tiempo





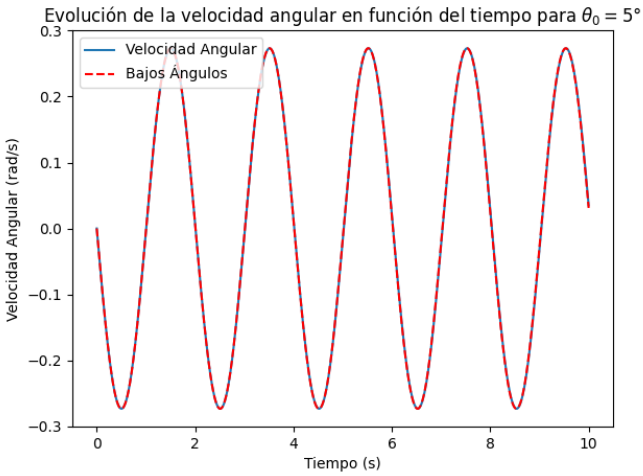


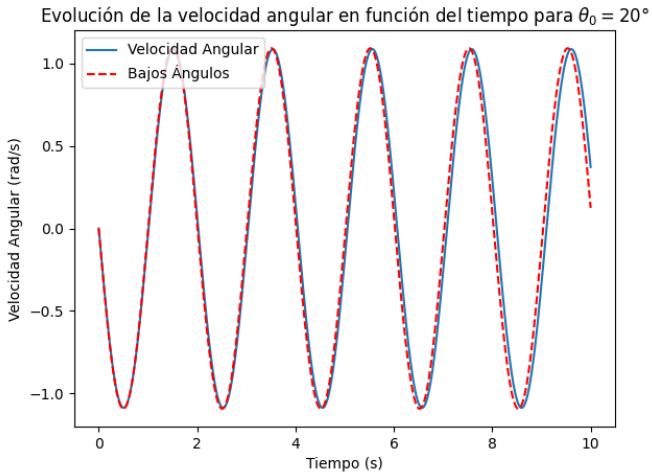
Comentar como para 5° ambas soluciones coinciden casi a la perfección, para 20 ° los métodos empiezan a distanciarse, y para 70° las soluciones se desfasan completamente en menos de 10 segundos.

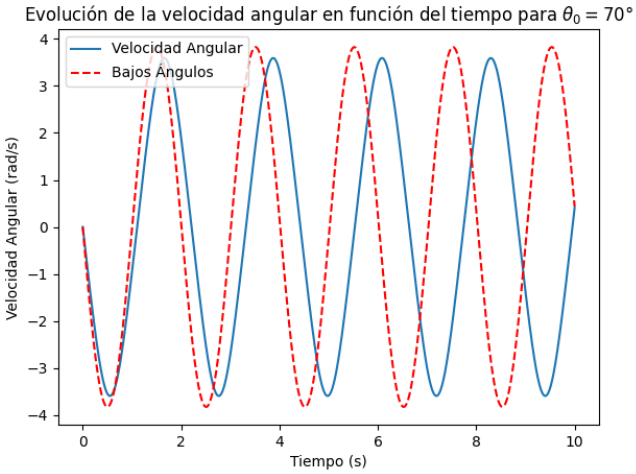
Código que calcula estas gráficas:

```
ang = int(input('Introduce el Angulo (deg): '))
file = input('Introduce el nombre de la foto: ') + ".png"

from matplotlib.pyplot import plot,xlabel,ylabel,savefig,legend,subplots_adjust,title
from numpy             import zeros,sin,cos,radians,degrees

l,g,n,dt = 1,9.8,100000,0.0001
tiempo,w,theta,angulo = zeros(n),zeros(n),zeros(n),zeros(n)
theta[0], angulo[0] = radians(ang), radians(ang)

for i in range(1,n):
    tiempo[i] = tiempo[i-1] + dt
    w[i] = w[i-1] - (g/l)*sin(angulo[i-1])*dt
    angulo[i] = angulo[i-1] + w[i-1]*dt
for i in range(len(angulo)):
    angulo[i] = degrees(angulo[i])
plot(tiempo,angulo,label="Angulo Numerico")

for i in range(1,n):
    tiempo[i] = i * dt
    theta[i] = radians(ang) * cos(((g/l)**0.5)*tiempo[i])
for i in range(len(theta)):
    theta[i] = degrees(theta[i])
plot(tiempo,theta,'r--',label="Bajos Angulos")

subplots_adjust(left=0.1, bottom=0.11, right=0.95, top=0.9)
title((r'Evolucion del angulo en funcion del tiempo para $\theta_0 = {}\degree$').format(ang))
legend(loc="upper left")
xlabel('Tiempo (segundos)')
ylabel('Angulo (grados)')
savefig(file,dpi=350)
```

2.1.2. Evolución de la Velocidad Angular en función del tiempo





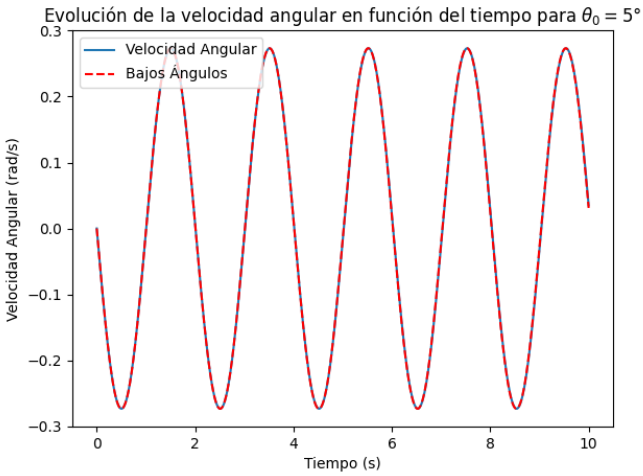


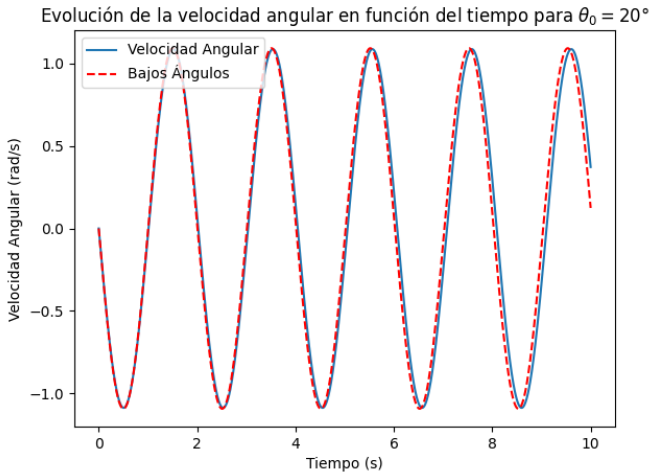
Al igual que con la posición, la velocidad angular en la solución para ángulos pequeños se desfasa respecto de la solución numérica para ángulos superiores a unos 15°

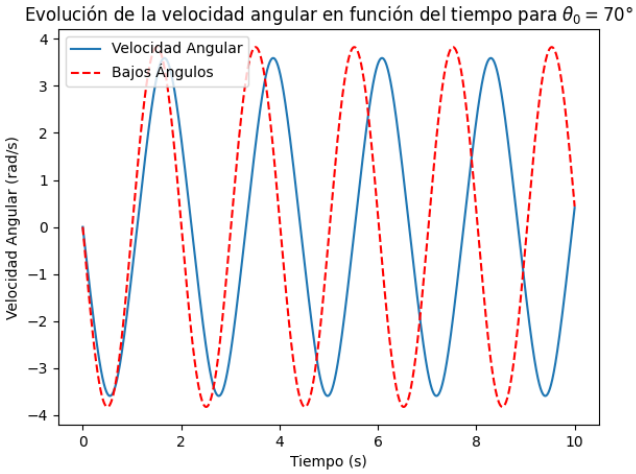
Código que calcula estas gráficas:

```
ang = int(input('Introduce el Angulo (deg): '))
file = input('Introduce el nombre de la foto: ') + ".png"

from matplotlib.pyplot import plot,xlabel,ylabel,savefig,legend,subplots_adjust,title
from numpy             import zeros,sin,cos,radians

l,masa,g,n,dt = 1,1,9.8,10000,0.001
tiempo,w,theta,wb = zeros(n),zeros(n),zeros(n),zeros(n)
theta[0] = radians(ang)

for i in range(1,n):
    tiempo[i] = i * dt
    theta[i] = radians(ang) * cos(((g/l)**0.5)*tiempo[i])
    wb[i] = radians(ang) * ((g/l)**0.5) * -1*sin(((g/l)**0.5)*tiempo[i])

for i in range(1,n):
    tiempo[i] = tiempo[i-1] + dt
    theta[i] = theta[i-1] + w[i-1]*dt
    w[i] = w[i-1] - ((g/l)*sin(theta[i]))*dt

plot(tiempo,w,label="Velocidad Angular")
plot(tiempo,wb,'r--',label="Bajos Angulos")

subplots_adjust(left=0.12, bottom=0.11, right=0.93, top=0.9)
title((r'Evolucion de la velocidad angular en funcion del tiempo para $\theta_0 = {}\degree$').format(ang))
legend(loc="upper left")
xlabel('Tiempo (s)')
ylabel('Velocidad Angular (rad/s)')
savefig(file)
```

2.1.3. Evolución de la Tensión en función del tiempo







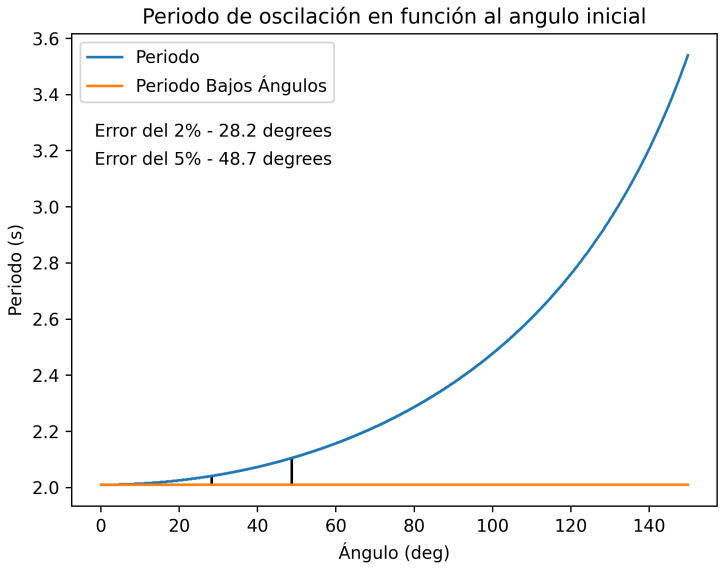
De la misma forma que con la posición y la velocidad angular, las soluciones se desfasan en ángulos superiores a 15° rápidamente.

Concluimos que siempre que el ángulo sea pequeño y el tiempo en el que lo apliquemos sea razonable, la solución para ángulos pequeños es perfectamente válida, y equiparable a la numérica.

Código que calcula estas gráficas:

```
ang = int(input('Introduce el Angulo (deg): '))
file = input('Introduce el nombre de la foto: ') + ".png"

from matplotlib.pyplot import plot,xlabel,ylabel,savefig,legend,title,subplots_adjust
from numpy             import zeros,sin,cos,radians

l,masa,g,n,dt = 1,1,9.8,10000,0.001
tiempo,w,theta,tension = zeros(n),zeros(n),zeros(n),zeros(n)

theta[0] = radians(ang)
tension[0] = masa*g*cos(theta[0])

for i in range(1,n):
    tiempo[i] = tiempo[i-1] + dt
    theta[i] = theta[i-1] + w[i-1]*dt
    w[i] = w[i-1] - ((g/l)*sin(theta[i]))*dt
    tension[i] = masa*(g*cos(theta[i-1]) + masa * l * w[i-1]**2)

plot(tiempo,tension,label="Tension")

subplots_adjust(left=0.12, bottom=0.11, right=0.93, top=0.9)
title((r'Evolucion de la tension en funcion del tiempo para $\theta_0 = {}\degree$').format(ang))
legend(loc="upper left")
xlabel('Tiempo (s)')
ylabel('Tension (N)')
savefig(file)
```

2.2. Cálculo del periodo de oscilación



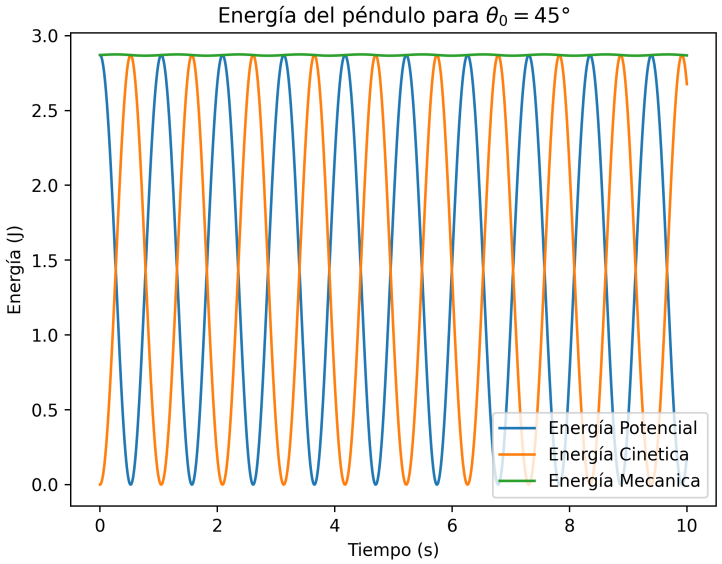
Después de calcular el periodo con ambos métodos para distintos ángulos iniciales, obtenemos esta gráfica. Para obtener el periodo hay que tener en cuenta cuando el ángulo deja de tomar una dirección y cambia a la otra, y después multiplicarlo por dos. Si lo pensamos la derivada del ángulo (la velocidad angular) será 0 siempre que alcance un máximo, y el periodo será el tiempo que tarda el péndulo en alcanzar dos de estos máximos (ida y vuelta).

El programa también calcula cuando la solución numérica dista de más de un 2 % y 5 % de la de ángulos pequeños. Esto resulta ser a partir de 28.2 y 48.7 grados respectivamente. Estos resultados están en línea con lo ya estudiado, que la solución para ángulos pequeños es eso, para ángulos pequeños, y diferirá de la realidad en mayor grado cuanto mayor sea el ángulo incial que usemos.

Código que calcula la gráfica:

```
file = input('Introduce el nombre de la foto: ') + ".png"

from matplotlib.pyplot import plot,legend,xlabel,ylabel,savefig,text,axvline,title,subplots_adjust
from numpy             import cos,sin,zeros,radians,degrees

l,masa,g,n,ang,c = 1,1,9.8,10000,0,1500
da,dt =              radians(0.1),0.001
T,Tp,angulo =        [],[],[]
t,w,th =             zeros(n),zeros(n),zeros(n)

def periodo(theta):
    th[0] = theta
    for i in range(1,n):
        t[i] = t[i-1] + dt
        th[i] = th[i-1] + w[i-1]*dt
        w[i] = w[i-1] - ((g/l)*sin(th[i]))*dt
        if i > 1 and th[i-2] < th[i-1] < th[i]:
            T = 2*t[i-1]
            break
    return T

def periodop(theta):
    th[0] = theta
    for i in range(1,n):
        t[i] = i * dt
        th[i] = theta * cos(((g/l)**0.5)*t[i])
        if i > 1 and th[i-2] < th[i-1] < th[i]:
            T = 2*t[i-1]
            break
    return T

def separacion(a,b,per):
    for i in range (len(a)):
        if a[i] >= b[i]:
            ma = a[i]
            mi = b[i]
        else:
            ma = b[i]
            mi = a[i]
        if round((100 * (ma - mi))/ma) == per:
            return i

for i in range(c):
    angulo.append(degrees(ang))
    ang += da
    T.append(periodo(ang))
    Tp.append(periodop(ang))

axvline(x=round(angulo[separacion(T,Tp,5)],2),color='k', linestyle='-', ymin=0.05,ymax=0.1)
axvline(x=round(angulo[separacion(T,Tp,2)],2),color='k', linestyle='-', ymin=0.05,ymax=0.06)
plot(angulo,T,label="Periodo")
plot(angulo,Tp,label="Periodo Bajos Angulos")
text(-4.5,3.15,"  Error del 5% - {} degrees".format(round(angulo[separacion(T,Tp,5)],2)))
text(-4.5,3.25,"  Error del 2% - {} degrees".format(round(angulo[separacion(T,Tp,2)],2)))
subplots_adjust(left=0.12, bottom=0.11, right=0.93, top=0.9)
title('Periodo de oscilacion en funcion al angulo inicial')

legend(loc="upper left")
xlabel('Angulo (deg)')
ylabel('Periodo (s)')
savefig(file,dpi=350)
```

2.3. Cálculo de las energías en el péndulo



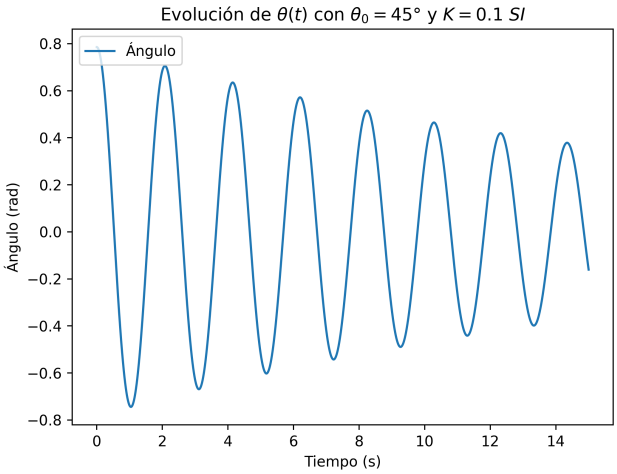
Tomando 10 segundos de la oscilación de un péndulo sin rozamiento que empezó a oscilar desde un ángulo de 45 grados, obtenemos esta gráfica para las energías. Observamos como inicialmente solo tiene energía potencial, y la energía cinética es cero. Conforme avanza el tiempo, estas dos se intercambian siempre estando a media fase de distancia entre sí. De esta forma, la suma de ellas en cada instante, la energía mecánica, tendrá un valor constante que en este caso ronda los 2.8J

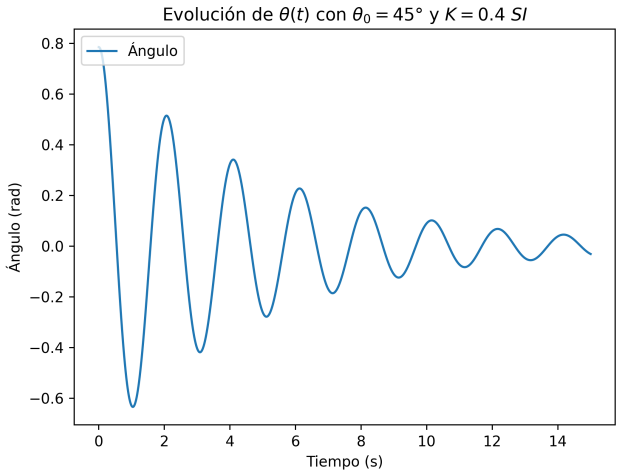
Código que calcula la gráfica:

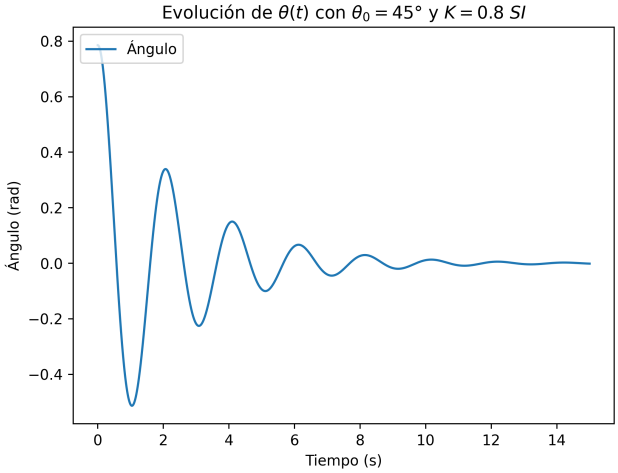
```
ang = int(input('Introduce el Angulo (deg): '))
file = input('Introduce el nombre de la foto: ') + ".png"
seg = int(input('Introduce los segundos a graficar: '))

from matplotlib.pyplot import plot,legend,xlabel,ylabel,savefig,title,subplots_adjust
from numpy             import cos,sin,zeros,radians

l,masa,g,n,dt = 1,1,9.8,1000*seg,0.001
t,w,th,ep,ec,em = zeros(n),zeros(n),zeros(n),zeros(n),zeros(n),zeros(n)

th[0] = radians(ang)
ep[0] = masa*g*(l-l*cos(th[0]))

for i in range(1,n):
    t[i] = t[i-1] + dt
    th[i] = th[i-1] + w[i-1]*dt
    w[i] = w[i-1] - ((g/l)*sin(th[i]))*dt
    ep[i] = masa*g*(l-l*cos(th[i]))
    ec[i] = 1/2*masa*(w[i]**2)*(l**2)
    em[i] = ep[i]+ec[i]
em[0]=ep[0]

plot(t,ep,label="Energia Potencial")
plot(t,ec,label="Energia Cinetica")
plot(t,em,label="Energia Mecanica")
subplots_adjust(left=0.12, bottom=0.11, right=0.93, top=0.9)
title((r'Energia del pendulo para $\theta_0 = {}\degree$').format(ang))

legend(loc="lower right")
xlabel('Tiempo (s)')
ylabel('Energia (J)')
savefig(file,dpi=350)
```

3. Parte b) - Péndulo con Fricción

3.1. Cálculo del Ángulo y la Velocidad Angular

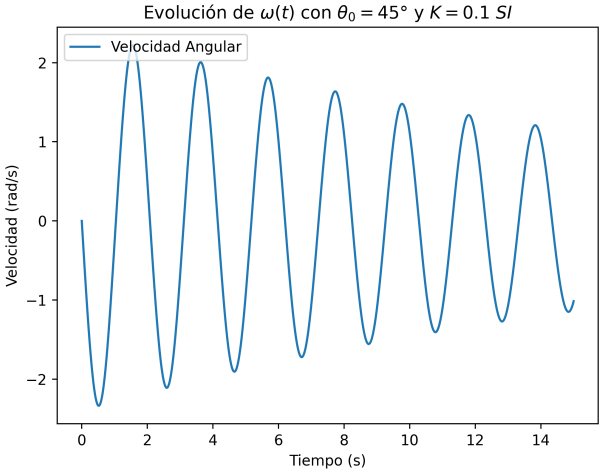


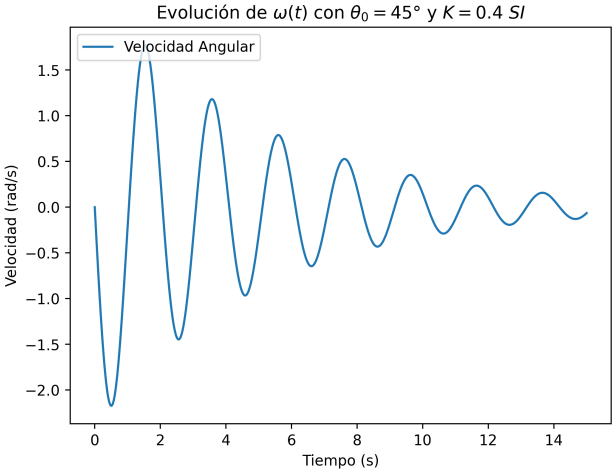


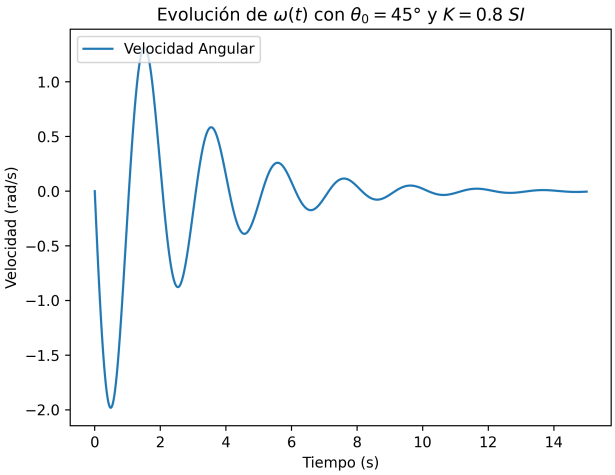


Observamos como un péndulo soltado desde el mismo ángulo inicial, 45 grados, y estudiado con diferentes valores para la K durante 15 segundos, el ángulo máximo que descrirá el péndulo será siempre el ángulo inicial, ya que conforme pase el tiempo las oscilaciones cada vez serán de menor tamaño.

Y es que para valores de K muy grandes, como es 0.8 en la última figura, en la primera oscilación no llega a alcanzar ni la mitad del ángulo desde el que se soltó, ya que la mayor parte de la energía potencial ha sido disipada.







Observamos aquí la velocidad angular para el mismo caso observado con el ángulo, un péndulo soltado desde 45 grados y graficado durante 15 segundos para varios valores de K.

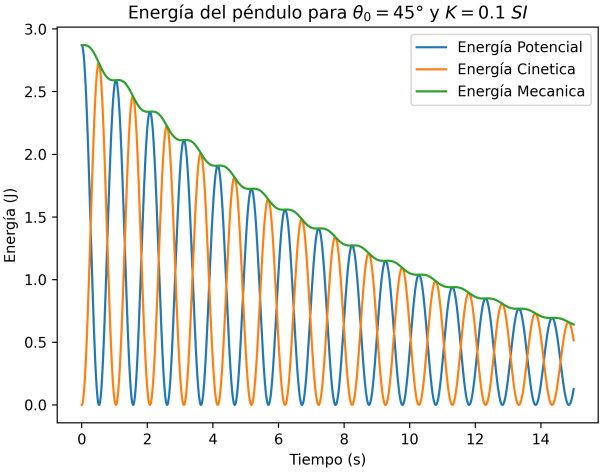
Al igual que con el ángulo, la velocidad angular alcanzará su pico máximo en la primera oscilación (no en el instante $t = 0$ ya que ahí $\omega = 0$) y irá descendiendo en mayor o menor grado en función del tamaño de la K utilizada.

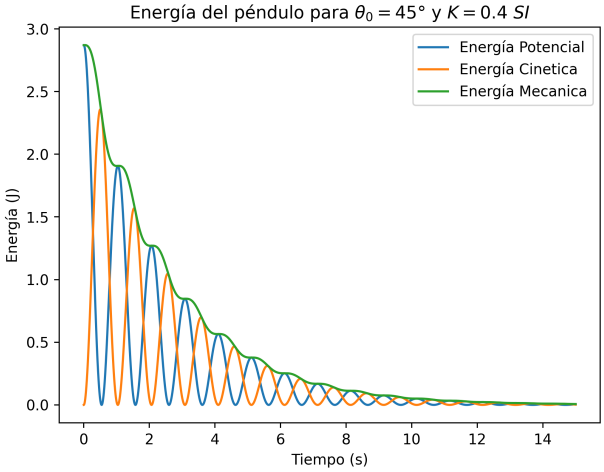
Código que calcula las gráficas:

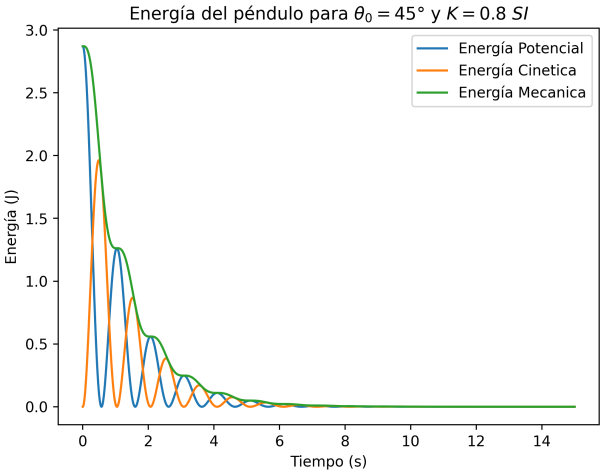
```
ang = int(input('Introduce el Angulo (deg): '))
k = float(input('Introduce la K: '))
file = input('Introduce el nombre de la TH: ') + ".png"
file1 = input('Introduce el nombre de la W: ') + ".png"
seg = int(input('Introduce los segundos a graficar: '))

import matplotlib.pyplot as plt
from numpy import zeros,radians,cos,sin

l,g,n,dt = 1,9.8,1000*seg,0.001
t,w,th = zeros(n),zeros(n),zeros(n)
th[0] = radians(ang)

for i in range(1,n):
    t[i] = t[i-1] + dt
    th[i] = th[i-1] + w[i-1]*dt
    w[i] = w[i-1] - ((g/l)*sin(th[i]) + w[i-1]*k)*dt

plt.subplots_adjust(left=0.12, bottom=0.11, right=0.93, top=0.9)
plt.title((r'Evolucion de $\theta(t)$ con $\theta_0 = {}\degree$ y $K = {}\ SI$').format(ang,k))
plt.plot(t,th,label="Angulo")
plt.legend(loc="upper left")
plt.xlabel('Tiempo (s)')
plt.ylabel('Angulo (rad)')
plt.savefig(file,dpi=350)
plt.figure()

plt.subplots_adjust(left=0.12, bottom=0.11, right=0.93, top=0.9)
plt.title((r'Evolucion de $\omega(t)$ con $\theta_0 = {}\degree$ y $K = {}\ SI$').format(ang,k))
plt.plot(t,w,label="Velocidad Angular")
plt.legend(loc="upper left")
plt.xlabel('Tiempo (s)')
plt.ylabel('Velocidad (rad/s)')
plt.savefig(file1,dpi=350)
```

Este código calcula tanto la velocidad angular como el ángulo en un solo código. El programa pregunta por un ángulo inicial y una K, y calcula ambas gráficas.

3.2. Cálculo de las energías







Ahora observamos la variación de la energía para el mismo caso que con la velocidad angular y el ángulo, un péndulo soltado desde 45 grados, graficado durante 15 segundos y con varios valores de K.

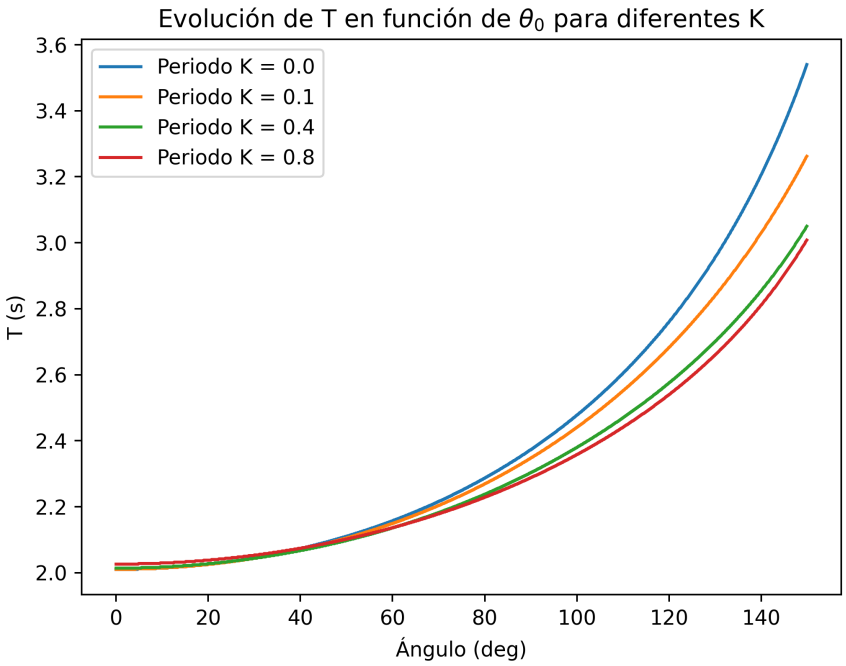
La energía mecánica no se conserva, va descendiendo a lo largo del tiempo ya que estamos en un caso donde existe una fuerza disipatoria (la fuerza de rozamiento) que la va haciendo disminuir. En cada instante la energía mecánica es la suma de la potencial y la cinética.

Es interesante observar que el descenso de la energía mecánica no es una curva suave, si no que tiene una forma casi ondulatoria. Esto se debe a que la energía mecánica se disipará en mayor medida cuanto más rápido vaya el péndulo, es decir cuanto mayor sea la energía cinética. Cuando el péndulo alcanza uno de los extremos, la energía cinética será 0, y por tanto no habrá disipación. Es en estos instantes donde la energía mecánica será plana y no descenderá, lo que hace que obesrvemos estos picos en la gráfica.

Código que calcula la gráfica:

```
ang = int(input('Introduce el Angulo (deg): '))
k = float(input('Introduce la K: '))
file = input('Introduce el nombre de la foto: ') + ".png"
seg = int(input('Introduce los segundos a graficar: '))

from matplotlib.pyplot import plot,xlabel,ylabel,savefig,legend,title,subplots_adjust
from numpy             import zeros,radians,cos,sin

l,masa,g,dt,n = 1,1,9.8,0.001,1000*seg
t,w,th,ep,ec,em = zeros(n),zeros(n),zeros(n),zeros(n),zeros(n),zeros(n)

th[0] = radians(ang)
ep[0] = masa*g*(l-l*cos(th[0]))
em[0] = ep[0]

for i in range(1,n):
    t[i] = t[i-1] + dt
    th[i] = th[i-1] + w[i-1]*dt
    w[i] = w[i-1] - ((g/l)*sin(th[i]) + w[i-1]*k)*dt
    ep[i] = masa*g*(l-l*cos(th[i]))
    ec[i] = 1/2*masa*(w[i]**2)*(l**2)
    em[i] = ep[i]+ec[i]

plot(t,ep,label="Energia Potencial")
plot(t,ec,label="Energia Cinetica")
plot(t,em,label="Energia Mecanica")

subplots_adjust(left=0.12, bottom=0.11, right=0.93, top=0.9)
title((r'Energia del pendulo para $\theta_0 = {}\degree$ y $K = {}\ SI$').format(ang,k))
legend(loc="upper right")
xlabel('Tiempo (s)')
ylabel('Energia (J)')
savefig(file,dpi=350)
```

3.3. Cálculo del Periodo en función de K



Por último graficaremos la evolución del periodo de la primera oscilación en el caso con rozamiento. Puesto que dependiendo del ángulo inicial desde el cual se suelte el péndulo afectará a la amplitud del ángulo, el periodo se verá afectado a su vez. Es por eso que lo graficamos en función al ángulo inicial.

Debo mencionar que lo que se calcula para esta gráfica es en realidad el tiempo en recorrer media oscilación, que luego se multiplica por dos para aproximar el periodo total. Hago esto por motivos de eficiencia del código, que de otra forma requeriría más tiempo de ejecución. El periodo calculado diferirá con el de una oscilación completa, ya que durante la segunda parte de la oscilación seguirá habiendo rozamiento. Esto da cabida a cierto grado de error en el cálculo de la gráfica, que se agravará cuanto mayor sea la K, y mayor el rozamiento, pero que no cambiaría de forma fundamental los resultados concluidos.

Podemos observar como para una K = 0, es decir sin rozamiento, el periodo será siempre mayor al del resto de curvas, que serán más bajas cuanto mayor sea la K, como era de esperar.

Código que calcula la gráfica:

```
file = input('Introduce el nombre de la foto: ') + ".png"
k = list(map(float,input('Introduce las K: ').split()))

from matplotlib.pyplot import plot,legend,xlabel,ylabel,savefig,title,subplots_adjust
from numpy             import cos,sin,zeros,radians,degrees

l,g,ang,c,n = 1,9.8,0,1500,1000*15
da,dt =       radians(0.1),0.001
T0,angulo =   [],[]
t,w,th =      zeros(n),zeros(n),zeros(n)

def periodo(theta,j):
    T = 0
    th[0] = theta
    for i in range(1,n):
        t[i] = t[i-1] + dt
        th[i] = th[i-1] + w[i-1]*dt
        w[i] = w[i-1] - ((g/l)*sin(th[i]) + w[i-1]*k[j])*dt
        if i > 1 and th[i-2] < th[i-1] < th[i]:
            T = 2*t[i-1]
            break
    return T

for j in range(len(k)):
    for i in range(c):
        angulo.append(degrees(ang))
        ang += da
        T0.append(periodo(ang,j))
    plot(angulo,T0,label="Periodo K = {}".format(k[j]))
    T0.clear()
    angulo.clear()
    ang = 0

subplots_adjust(left=0.12, bottom=0.11, right=0.93, top=0.9)
title(r'Evolucion de T en funcion de $\theta_0$ para diferentes K')
legend(loc="upper left")
xlabel('Angulo (deg)')
ylabel('T (s)')
savefig(file,dpi=350)
```

Este código grafica la evolución del periodo en función del ángulo inicial para tantas K como le sean introducidas, si se le introduce una única K, solo graficará una, pero si se le introducen 4, graficará 4.

El tiempo de ejecución variará drásticamente en función de cuantas curvas tenga que calcular.