Tarea 2, Métodos Numéricos para la Ciencia y la Ingeniería

Jaime Castillo Lara

La tarea trabaja un sistema físico que consiste en una pelota rebotando (con movimiento solo en el eje vertical) sobre un suelo que se mueve sinusoidalmente.

Para el movimiento del suelo nos daremos la función:

$$y(t) = Asen(wt)$$

Donde A es la amplitud de la oscilación y w es la frecuencia de la misma.

La regla de choque para los choques inelásticos entre la pelota y el suelo, estará dada por:

$$v_p'(t^*) = (1+\eta)v_s(t^*) - \eta v_p(t^*)$$

Donde t^* es el momento del choque, v_p es la velocidad de la pelota justo antes del choque, v_p' es la velocidad de la pelota justo después del choque, v_s es la velocidad del suelo en el instante del choque, y η es el coeficiente de restitución.

En nuestro programa, tendremos una función resta() y funciones que ajustaran el tiempo para cada choque, las cuales deberán llamar a variables que se sobrescribirán después de cada choque. Para no tener problemas con esto, iniciaremos el codigo definiendo variables globales para todo el programa. Escribimos:

```
def definiciones():
    global pos_init
    pos_init = 0
    global vel_init
    vel_init = 15
    global w
    w= 1.79
    global ajt
    ajt = 0
```

donde pos_init: Posición de la pelota en el instante del último choque con el suelo.

vel_init: Velocidad de la pelota en el instante del último choque con el suelo.

(nos damos el valor para t=0)

ajt: El tiempo que ha pasado desde el último choque entre la pelota y el suelo.

Nos damos además la frecuencia de oscilación w=1.79

Para determinar para cada momento la distancia entre la pelota y el suelo, definimos la función resta(), que calculará la diferencia entre las posiciones de ambos. Escribimos:

```
def resta(time):
   global pos_init
   global vel_init
   global w
   return pos_init + (vel_init*tiempoajustado(time)) -
(tiempoajustado(time)*tiempoajustado(time)/2) - np.sin(w*time)
```

La función resta() recibe como parámetro el tiempo, y toma como condiciones iniciales pos_init y vel_init. Sabiendo que $\text{posición} = \text{distancia} * \text{tiempo}$, determinamos la posición de la pelota a partir de las condiciones iniciales, y la posición del suelo con la función seno que nos dimos. Finalmente, se restan ambas posiciones, y la función nos dará la distancia entre ambos objetos para cada tiempo.

Notamos que para la pelota, ocupamos para la variable temporal: tiempoajustado(time)

Y para el suelo, ocupamos solamente time.

Esto se debe a que utilizaremos contabilizadores del tiempo distintos para la pelota y para el suelo. Para el suelo, el tiempo irá creciendo normalmente desde el inicio de la modelación. Sin embargo, para la pelota, necesitamos que el tiempo se reinicie después de cada choque. Con este objetivo definimos la variable ajt.

Definimos entonces dos funciones:

```
def ajustartiempo(t):
    global ajt
    ajt = t
```

La función ajustartiempo() actualiza la variable ajt cada vez que ocurre un choque.

```
def tiempoajustado(t):
    global ajt
    return t – ajt
```

La función tiempoajustado() toma la variable ajt y registra el tiempo desde el último choque.

Para conocer la velocidad de la pelota inmediatamente después de cada choque, definimos la función choque(). Escribimos:

```
def choque(coef,velp,vels):
    return ((1+coef)*vels) - (coef*velp)
```

A la función se le ingresa la velocidad del suelo en el instante del choque (vels) y la velocidad de la pelota justo antes del choque (velp), y retorna la velocidad de la pelota inmediatamente después del choque.

Para modelar la cinemática de todo el sistema, definimos la función func_cinm(). A la cual se le ingresarán el número de choques que se desean modelar (Ns) y el coeficiente de restitución (coef).

Inicializamos entonces escribiendo:

```
def func_cinm(Ns,coef):
    global vel_init
    global pos_init
    epsilon = 0.01
    t=0;
    nchoques = 0
    dato = resta(t+epsilon)
    vp=0
    vs=0
    tiempo=[]
    vel=[]
```

Vemos que la función utilizará las variables globales vel_init y pos_init. Epsilon será el delta de tiempo que nos daremos para nuestra simulación discreta. Definimos además la variable “dato” que nos dará la distancia entre la pelota y el suelo. Inicializamos el tiempo (t), el número de choques (nchoques). Creamos dos arreglos, uno donde se irán registrando los tiempos y las velocidades para cada choque.

Para iterar, escribimos:

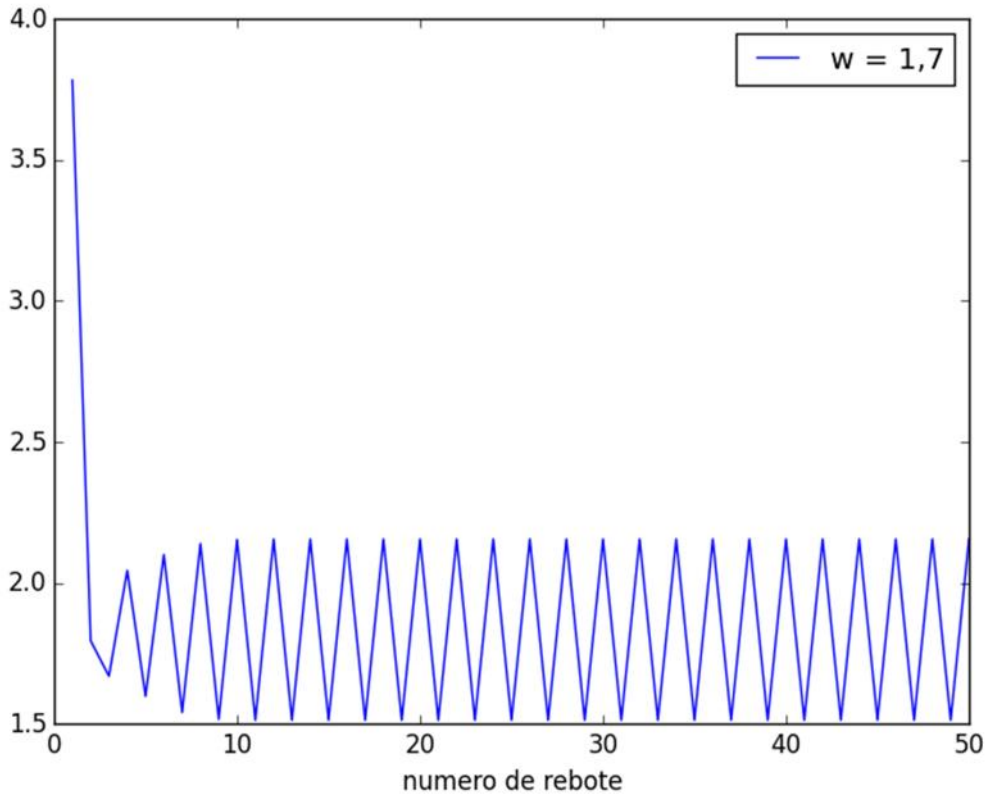
```
while nchoques<Ns:
      while dato>0  :
         t= t+epsilon
         dato = resta(t)
      tiempo.append(brentq(resta,t-epsilon,t))
      t=tiempo[nchoques]
      vp=(vel_init)-(tiempoajustado(t))
      vs=w*np.cos(w*t)
      vel.append(choque(coef,vp,vs))
      pos_init= np.sin(w*t)
      vel_init= vel[nchoques]
      ajustartiempo(t)
      nchoques+=1
      t=t+epsilon
      dato=resta(t)
   return tiempo , vel
```

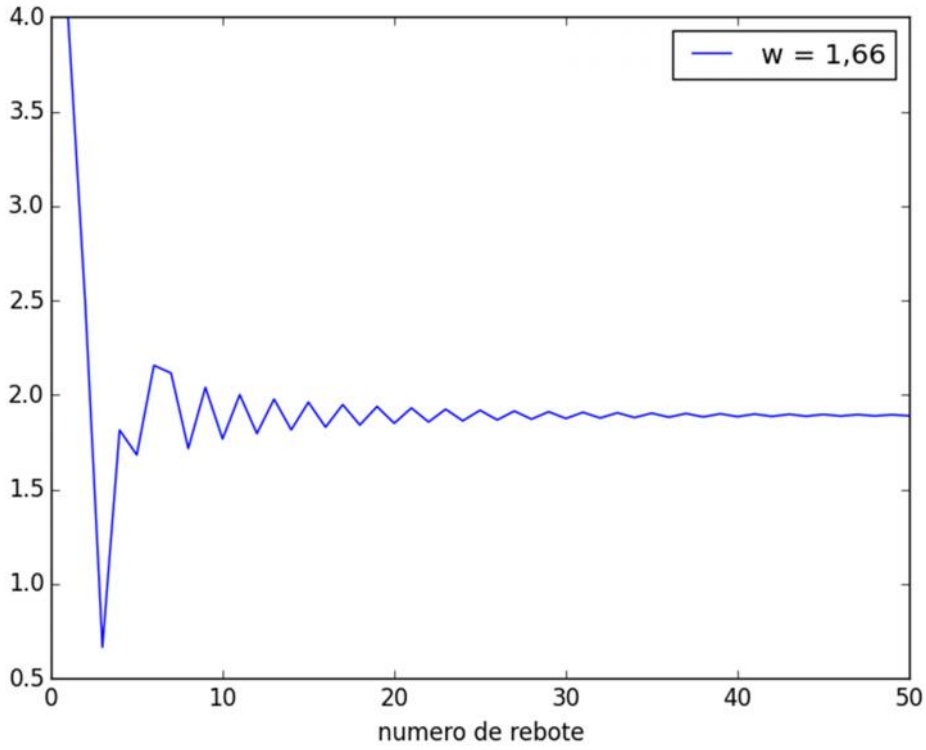
El primer while hará que la iteración se repita mientras el número de choques sea menor al total que buscamos modelar. El segundo while hará que la simulación continúe mientras la pelota y el suelo no choquen. La función brentq() encuentra el cero de una función, para un intervalo dado. Por lo tanto, (brentq(resta,t-epsilon,t)) nos entregará el momento del choque de la iteración correspondiente. Así, tiempo.append(brentq(resta,t-epsilon,t)) hará que el momento del choque para cada iteración se vaya registrando en el arreglo de tiempo. Para conocer la velocidad del suelo, derivamos la función seno que nos dimos para la posición. Por lo tanto, escribimos vs=w*np.cos(w*t).

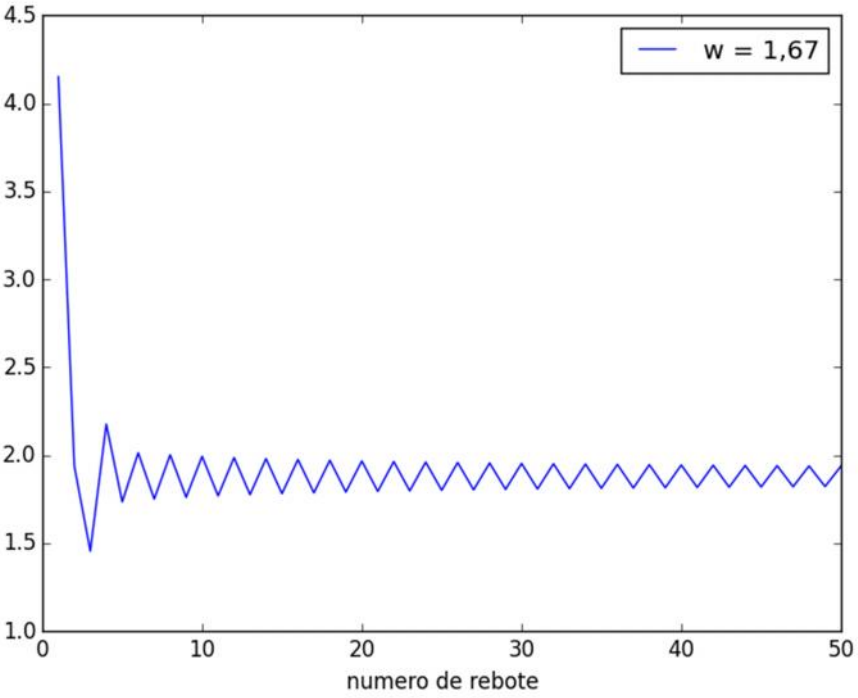
Por cinemática, sabemos que: $v_{final} = v_{inicial} - at$. Dado que estamos trabajando con aceleración de gravedad g=1, tendremos simplemente $v_{final} = v_{inicial} - tiempo.$

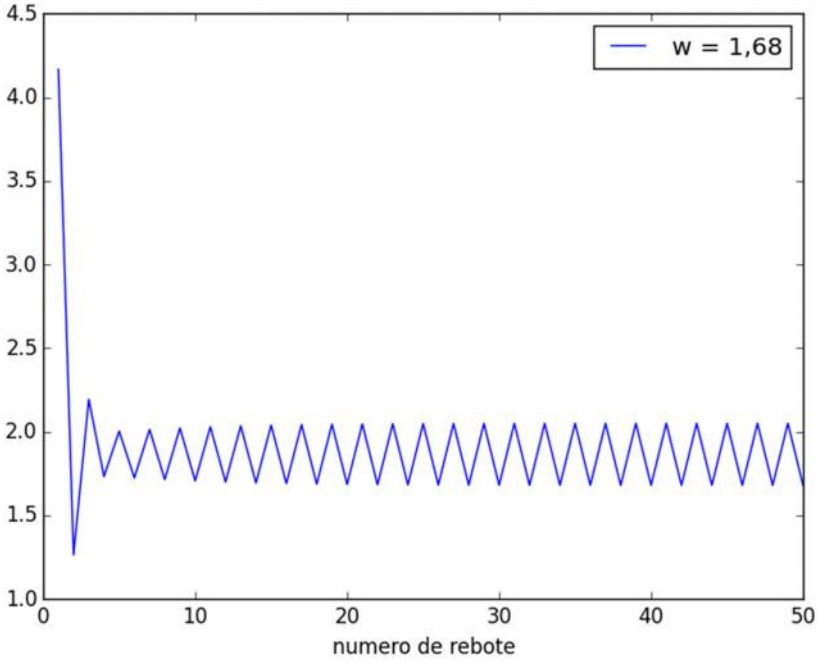
Por lo tanto, para conocer la velocidad de la pelota, escribimos vp=(vel_init)-(tiempoajustado(t)). Para ir registrando la velocidad de la pelota justo después de cada choque, utilizamos la función choque() que definimos con este objetivo, y la vamos registrando en el arreglo correspondiente, Por lo tanto, escribimos vel.append(choque(coef,vp,vs)). Dado que en el instante del choque la posición de la pelota será la misma que la del suelo, para reiniciar las condiciones iniciales, escribimos pos_init= np.sin(w*t). Dado que la velocidad de la pelota justo después del choque quedó registrada en el arreglo, para utilizarla como condición inicial para la siguiente iteración, escribimos vel_init= vel[nchoques]. Finalmente, para terminar la iteración, ajustamos el contador de tiempo para la pelota ajustartiempo(t), agregamos una unidad al contador de choques nchoques+=1, y pedimos que nos retorne el tiempo y la velocidad de la pelota para cada choque return tiempo , vel.

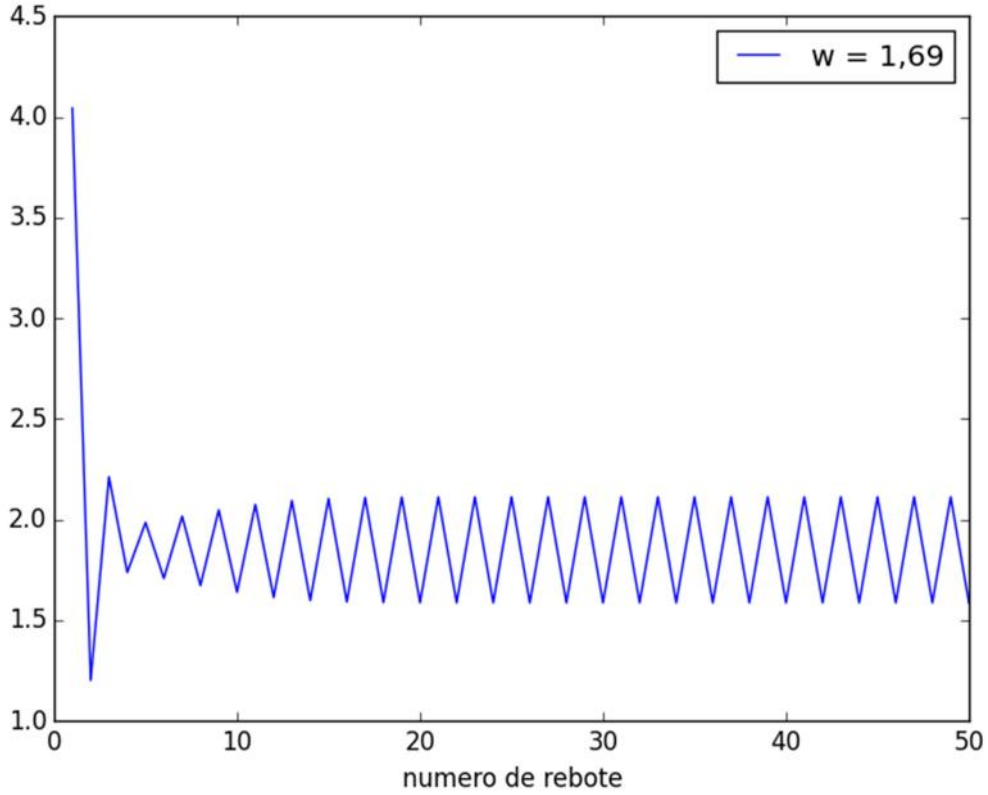
Una vez que tenemos estos valores, graficamos los resultados para distintos w, con lo que encontramos:




w = 1,66
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0
10
20
30
40
50
numero de rebote


w = 1,67
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
10
20
30
40
50
numero de rebote


w = 1,68
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
10
20
30
40
50
numero de rebote


w = 1,69
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0
10
20
30
40
50
numero de rebote