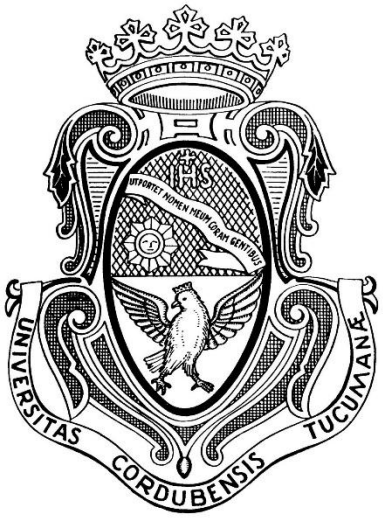
UNIVERSIDAD NACIONAL DE CÓRDOBA

Facultad de Ciencias Exactas Físicas y Naturales



Métodos Numéricos

Flotante cónico

Profesor: Gabriel Maggio

Año: 2014

Stigliano, Agustín Federico

Carrera: IME

Matrícula: 35260049

INTRODUCCION

En el presente trabajo se explicara el método de Euler para determinar los valores de una funciones mediante la ecuación de la derivada segunda de dicha función, luego se mostrara la codificación de un programa que resuelve este problema presentando en detalle todos los aspectos que tuvieron en cuenta. La función en cuestión responde a un fenómeno físico de un flotante que oscila sobre un punto de equilibrio a partir de una altura inicial dada. El flotante tiene permitido moverse solamente a lo largo del eje vertical, y su oscilación queda determinada por la siguiente ecuación diferencial:

$$\frac{d^2y}{dt^2} = g(1 - ay^3)$$

Donde,

g es la aceleración de la gravedad 9,8m/s2;
a es una constante que depende de la densidad del flotante y sus dimensiones (16m−3);
y es la distancia de la base del flotante respecto a la superficie del líquido [m].

En donde el objetivo es determinar los valores de la posición en función del tiempo, la amplitud y periodo de las oscilaciones, mediante un algoritmo codificado sobre una plataforma informática que nos permita desarrollar una sucesión de cálculos a gran velocidad.

A continuación se explicara brevemente el desarrollo del método de Euler para luego poder mostrar cómo se aplicó en este caso:

1) Dada una función f(x), Euler propone que f ‘(x) es

$$f^{'}(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

   Lo cual si tomamos un h infinitesimal, podremos afirmar que obtenemos un valor aproximado de la derivada de f(x) para cada punto.

2) Luego despejando de la ecuación anterior tenemos los valores de la función en un punto, pueden ser calculados mediante el valor de la función un instante anterior, o sea:

$$f(x1) = f(x0 + h) \cong f(x0) + h * f'(x0)$$

...

$$f(x_n) = f(x_{n-1} + h) \cong f(x_{n-1}) + h * f'(x_{n-1})$$

3) Para poder determinar los valores sucesivos que toma una función hay que establecer las condiciones iniciales de partida para tomar como referencia.

Vamos a desarrollar la forma en la que se procedió para aplicar el método de Euler, pero primero vamos de definir cómo y(t), función primitiva o de orden 0, Dy(t), derivada de orden 1, y DDy(t) derivada de orden 2, h el paso, y n el número de la iteración correspondiente:

1) Determinamos las condiciones iniciales para la configuración de un escenario físico a medir, en este caso le damos un valor a y(0), o sea una altura inicial respecto a la posición de equilibrio, y Dy(0)=0.0 ya que para cualquier instante inicial consideramos al flotante quieto o que no manifiesta velocidad alguna. Entonces tenemos nuestras condiciones iniciales,

   :y(0) = A (donde A pertenece a los reales)

   :Dy(0) = 0.0

   2) Ahora podemos evaluar la ecuación de DDy(t) para el instante t=0

$$DDy(0) = g(1 - aA^3)$$

$$DDy(0) = 9.8 * (1 - 16A^3)$$

   3) Luego según el valor que se la haya dado a A es necesario determinar un paso “h” infinitesimalmente pequeño y constante para poder aproximar Dy(t) por medio de DDy(t), y a y(t) por medio de Dy(t), o sea:

$$Dy(1) = Dy(0) + h * DDy(0)$$
$$y(1) = y(0) + h * Dy(0)$$
$$DDy(1) = 9.8 * (1 - 16 * y(1)^3)$$

$$Dy(2) = Dy(1) + h * DDy(1)$$
$$y(2) = y(1) + h * Dy(1)$$
$$DDy(2) = 9.8 * (1 - 16 * y(2)^3)$$

…

$$Dy(n) = Dy(n-1) + h * DDy(n-1)$$
$$y(n) = y(n-1) + h * Dy(n-1)$$
$$DDy(n) = 9.8 * (1 - 16 * y(n)^3)$$

Hubo una serie de consideraciones especiales que se tuvieron en cuenta a la hora de codificar el algoritmo para que este funcione eficientemente, estas fueron:

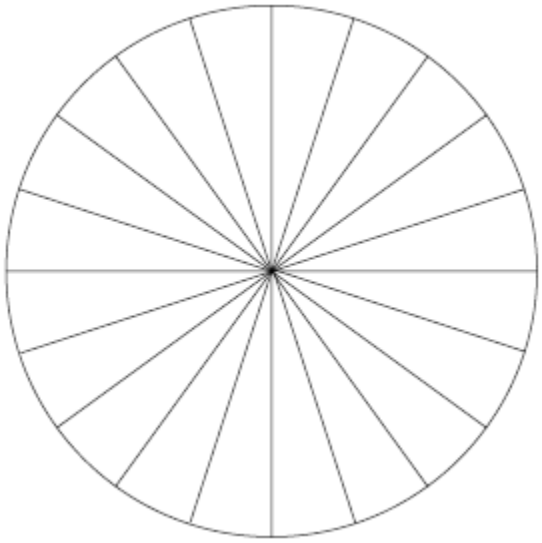
-Error en la programación.

-Elección del paso adecuado.

Se logró minimizar los errores gracias a un análisis previo del comportamiento de la función según las condiciones iniciales que se propusieran.

Por ejemplo:

Los valores de la aceleración (DDy) se representancomo curva exponencial (para ¼ de ciclo) cada vez más pronunciada para valor de A cada vez mayores, por lo que el paso h debe ser elegido con un criterio no lineal para diferentes valores de A.Esto se solucionó considerando una circunferencia de radio A, de perímetro igual al módulo del período de la oscilación, y subdividiendo el circulo en n partes, donde n es el número de iteraciones deseadas por ciclo, por lo tanto el programa automáticamente calculo un paso h recomendado para cada valor de A en funcióndel seno de la razón del periodo por la cantidad de iteraciones deseadas por ciclo, por lo tanto, si para cualquier A se cumple que Periodo << iteraciones por ciclo, entonces por razón de limites notables, si el seno de un ángulo pequeño tiende a ser el mismo ángulo, el sen(periodo/iteraciones) se ajusta perfecto a ser un valor aproximado del h con menor error, gráficamente podría explicarse de la siguiente manera.



Viéndolo como una rueda de bicicleta, a medida que aumentamos la cantidad de iteraciones por ciclos, la razón periodo iteraciones es cada vez menor (aumentando la cantidad de rayos de la rueda), por lo que el valor de h (el ángulo entre cualquiera de dos rayos consecutivos) se vuelve cada vez más preciso, esto se debe gracias a que conocemos el valor del periodo de antemano.

APLICACIÓN DEL METODO EN PYTHON

Para que el programa funcione de manera eficiente fue necesario conocer de antemano que resultados eran los esperados para cada valor inicial de altura que se le dé, en este caso se podría guiar al programa por el camino en el que arrojase los resultado con el menor error posible. El programa corre empezando por el **módulo de carga de valores iniciales**, en donde se clasifican los valores de y(t) y se crean las variables necesarias para el funcionamiento del programa.

La clasificación de los valores de y sigue la siguiente forma:

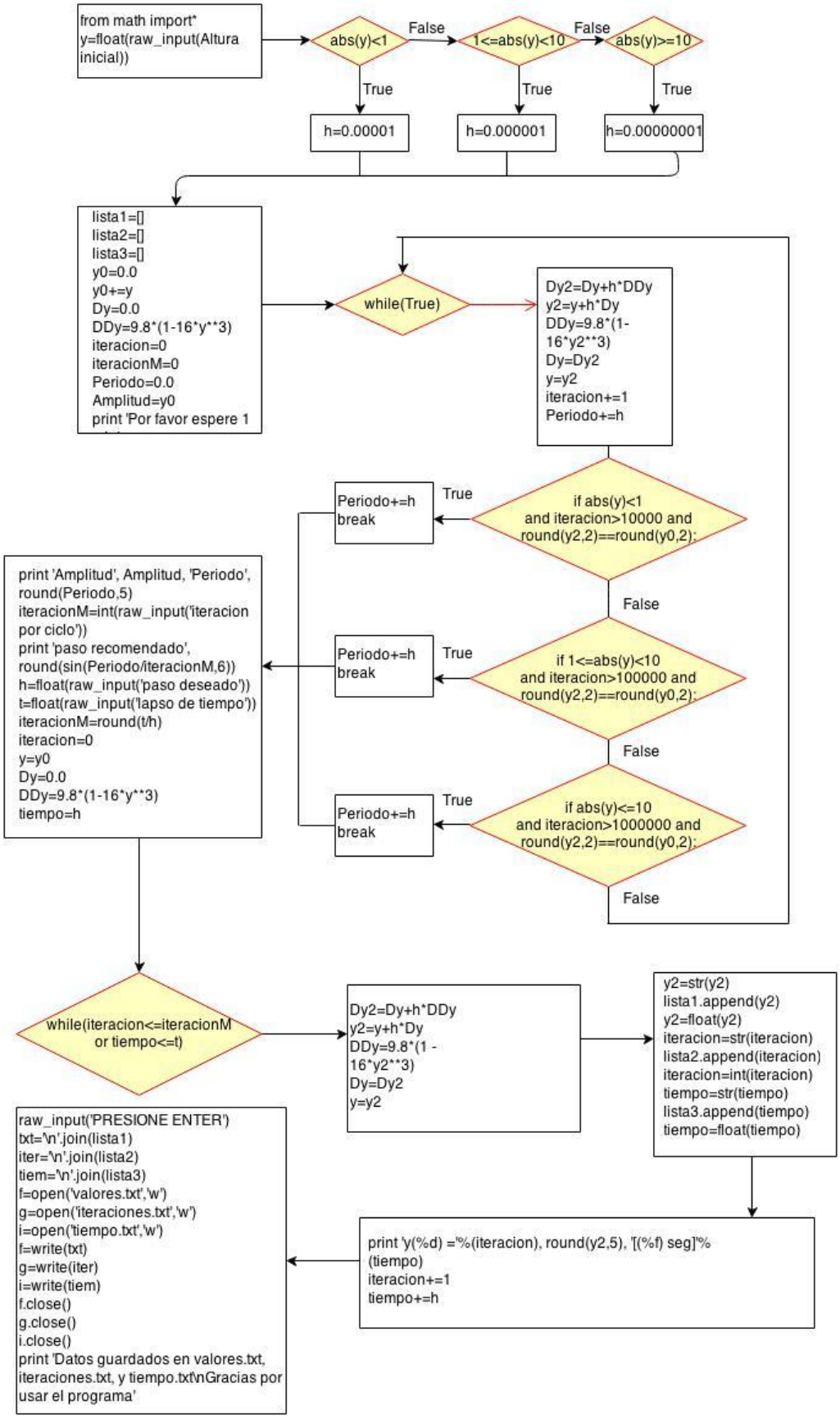
- Valores pequeños: abs(y) < 1
- Valores medios: $1 \leq abs(y) < 10$
- Valores grandes: $abs(y) \geq 10$

En primer lugar según la magnitud de y(t) se toma un paso h exageradamente chico, y se le pide al programa que calcule una cantidad inmensa de iteraciones aplicando el método de Euler lo que llamamos **módulo de cálculo de datos previos aproximados**, en donde a partir de un valor dado de iteraciones el programa compare los valores actuales de y con el valor inicial, y para cuando por medio de un redondeo el valor actual sea igual al valor inicial el programa pueda determinar el tiempo que tarda el flotante en salir de su condición inicial hasta volver al mismo punto.

Luego pasamos al **modulo de carga de datos**en donde se muestra por pantalla al usuario el periodo y la amplitud de la oscilación en función de la altura inicial dada.Ahora el usuario determina cuantas iteraciones por ciclo va a querer calcular y el programa automáticamente indica cual es el paso conveniente para que los valores sean los más precisos y eficientes posibles, el usuario también ingresa la ventana de tiempo en la que el programa tiene que correr y el paso que desea. Concluye el modulo haciendo un **reinicio de valores** para pasar al módulo siguiente.

El programa corre el **modulo programa principal** mostrando por pantalla el valor de la posición en función del tiempo para cada iteración, a medida que se realizan y se muestran dichos cálculos también se van guardando en unas listas en el **módulo de almacenamiento de datos** y el programa finaliza con el **módulo de exportación de datos** grabando en el directorio del programa tres archivos .txt, uno con los valores sucesivos de y, otro con los valores de las iteraciones, y uno más con el tiempo transcurrido para cada iteración, los archivos se llaman “valores.txt”, “iteraciones.txt”, “tiempo.txt” respectivamente. De esta forma se facilita la exportación de información del programa y permite graficar la función en otros programas sin tener que pasar los datos manualmente.

Ahora presentamos el diagrama de flujo del programa y luego la codificación del algoritmo exportada directamente del archivo flotante_conico.py.


from math import*
y=float(raw_input(Altura inicial))
abs(y)<1
False
1<=abs(y)<10
False
abs(y)>=10
True
True
True
h=0.00001
h=0.000001
h=0.00000001
lista1=[]
lista2=[]
lista3=[]
y0=0.0
y0+=y
Dy=0.0
DDy=9.8*(1-16*y**3)
iteracion=0
iteracionM=0
Periodo=0.0
Amplitud=y0
print 'Por favor espere 1
while(True)
Dy2=Dy+h*DDy
y2=y+h*Dy
DDy=9.8*(1-16*y2**3)
Dy=Dy2
y=y2
iteracion+=1
Periodo+=h
if abs(y)<1 and iteracion>10000 and round(y2,2)==round(y0,2):
True
Periodo+=h
break
False
if 1<=abs(y)<10 and iteracion>100000 and round(y2,2)==round(y0,2):
True
Periodo+=h
break
False
if abs(y)<=10 and iteracion>1000000 and round(y2,2)==round(y0,2):
True
Periodo+=h
break
False
print 'Amplitud', Amplitud, 'Periodo', round(Periodo,5)
iteracionM=int(raw_input('iteracion por ciclo'))
print 'paso recomendado', round(sin(Periodo/iteracionM,6))
h=float(raw_input('paso deseado'))
t=float(raw_input('lapso de tiempo'))
iteracionM=round(t/h)
iteracion=0
y=y0
Dy=0.0
DDy=9.8*(1-16*y**3)
tiempo=h
while(iteracion<=iteracionM or tiempo<=t)
Dy2=Dy+h*DDy
y2=y+h*Dy
DDy=9.8*(1 - 16*y2**3)
Dy=Dy2
y=y2
y2=str(y2)
lista1.append(y2)
y2=float(y2)
iteracion=str(iteracion)
lista2.append(iteracion)
iteracion=int(iteracion)
tiempo=str(tiempo)
lista3.append(tiempo)
tiempo=float(tiempo)
print 'y(%d) ='%(iteracion), round(y2,5), '[(%f) seg]'%(tiempo)
iteracion+=1
tiempo+=h
raw_input('PRESIONE ENTER')
txt='\n'.join(lista1)
iter='\n'.join(lista2)
tiem='\n'.join(lista3)
f=open('valores.txt','w')
g=open('iteraciones.txt','w')
i=open('tiempo.txt','w')
f=write(txt)
g=write(iter)
i=write(tiem)
f.close()
g.close()
i.close()
print 'Datos guardados en valores.txt, iteraciones.txt, y tiempo.txt\nGracias por usar el programa'

```
frommathimport*

##CARGA DE VALORES INICIALES
y= float(raw_input('ALTURA INICIAL : '))
if abs(y)<1:
  h=0.00001
if abs(y)>=1 and abs(y)<10:
h=0.000001
ifabs(y)>=10:
  h=0.00000001

lista1= []
lista2= []
lista3= []
y0= 0.0
y0+= y
Dy=0.0
DDy=9.8*(1-16*y**3)
iteracion= 0
iteracionM= 0
Periodo=0.0
Amplitud=abs(y0)

#MODULO DE CALCULO DE DATOS PREVIOS APROXIMADOS
print '\nPor favor espere 1 min\n'
while (True):
Dy2=Dy+h*DDy
  y2=y+h*Dy
  DDy=9.8*(1-16*y2**3)
  Dy=Dy2
  y=y2
iteracion+=1
Periodo+=h
if abs(y)<1 and iteracion>10000 and round(y2,2)==round(y0,2):
Periodo+=h
break
if abs(y)>=1 and abs(y)<10 and iteracion>100000 and round(y2,2)==round(y0,2):
Periodo+=h
break
if abs(y)>=10 and iteracion>1000000 and round(y2,2)==round(y0,2):
Periodo+=h
break

#MODULO DE CARGA DE DATOS
print 'Amplitud max : ', Amplitud, '\nPeriodo [seg] : ', round(Periodo,5)
iteracionM=int(raw_input('n° iteraciones por ciclo deseadas(aprox): '))
print 'paso recomendado (h): ', round(sin(Periodo)/iteracionM,6), '\n',
h=float(raw_input('pasodeseado= '))
t=float(raw_input('lapso de tiempo [seg]= '))
iteracionM=round(t/h)

#REINICIAR VALORES
iteracion=0
y= y0
Dy= 0.0
DDy=9.8*(1-16*y**3)
tiempo=h
```

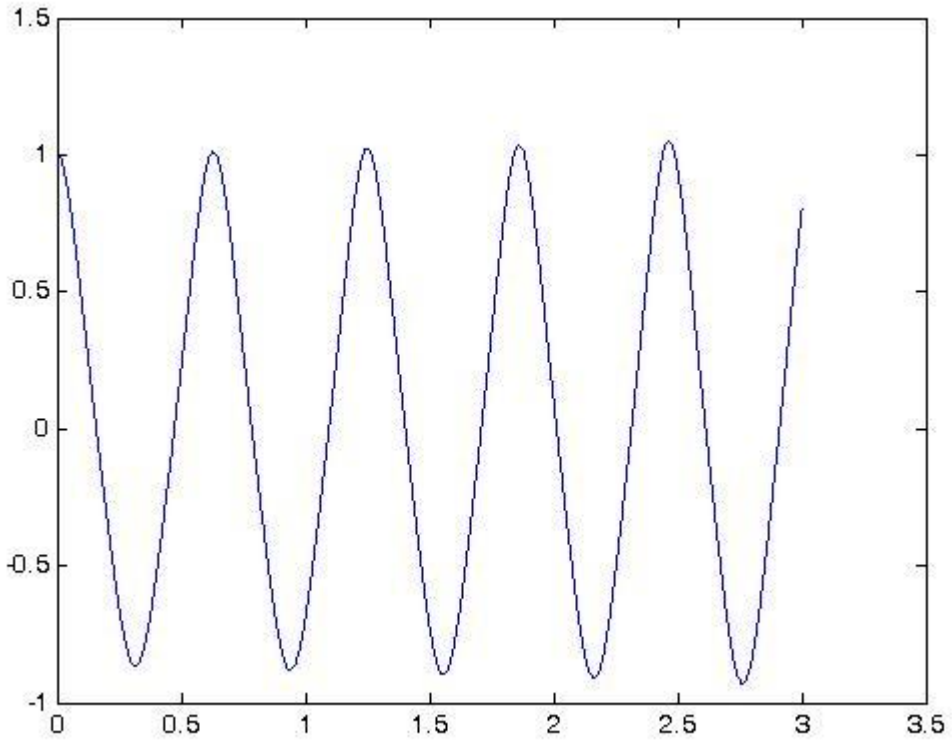
```
#PROGRAMA PRINCIPAL
while (iteracion<=iteracionMor tiempo<=t):
  Dy2=Dy+h*DDy
  y2=y+h*Dy
  DDy=9.8*(1-16*y2**3)
  Dy=Dy2
  y=y2
  #MODULO DE ALMACENAMIENTO DE DATOS
  y2= str(y2)
lista1.append(y2)
  y2= float(y2)
iteracion=str(iteracion)
lista2.append(iteracion)
iteracion=int(iteracion)
tiempo=str(tiempo)
lista3.append(tiempo)
tiempo=float(tiempo)
  #--------------------------------
print 'y(%d) ='%(iteracion), round(y2,5), ' [%f seg]'%(round(tiempo,5))
iteracion+=1
tiempo+=h

#MODULO DE EXPORTACION DE DATOS
raw_input('\nPRESIONE ENTER')
txt = '\n'.join(lista1)
iter= '\n'.join(lista2)
tiem= '\n'.join(lista3)
f = open ('valores.txt', 'w')
g = open ('iteraciones.txt', 'w')
i = open ('tiempo.txt', 'w')
f.write(txt)
g.write(iter)
i.write(tiem)
f.close()
g.close()
i.close()
print 'Datos guardados en valores.txt, iteraciones.txt y tiempo.txt\nGracias por usar el programa'
```

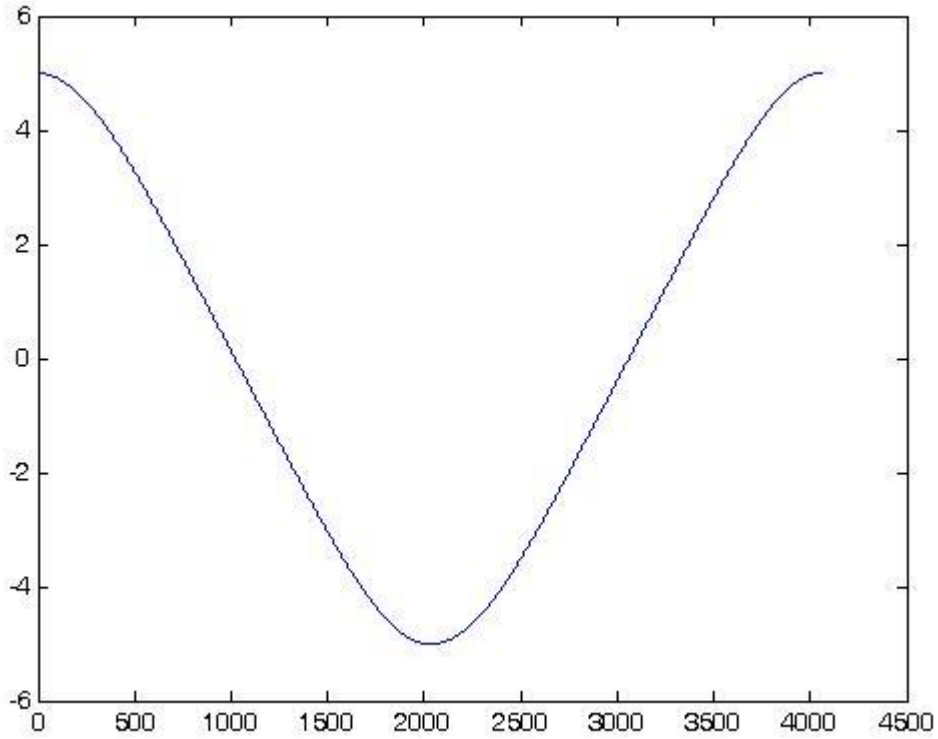
Para la **verificación y comprobación** de los datos se exportaron los datos desde el python al notepad, de una gama de valores iniciales tanto pequeños medianos y grandes, positivos y negativos; para luego graficarlos en matlab. Se buscaron diferentes alternativas, como por ejemplo, visualizar si el periodo que arroja el programa es el determinado para la oscilación correspondiente, verificar que la función sea continua en todos los puntos según el paso recomendado; y por último se tomó como modelo en paralelo el Excel, tomando los mismos valores iniciales, el mismo paso, para el mismo lapso de tiempo. No sólo que para el tiempo determinado la hoja de cálculos llega a exactamente a la misma posición y(t), sino también que llega en la misma cantidad de iteraciones; lo que los errores propios de la codificación son despreciables.

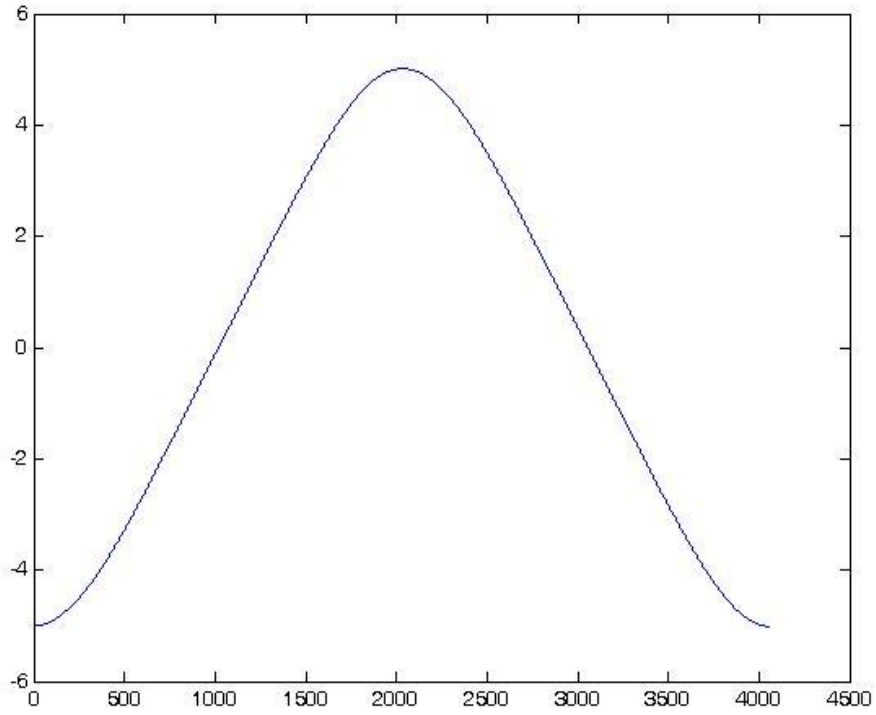
Pasamos a mostrar las gráficas,

1) Se tomó un valor de y(0)=1, h=0.005, t=3 seg y se verificaron las gráficas de los valores que arrojaron el Excel y el python, se usó la función 'hold_on' por lo cual las curvas están superpuestas, pero la simetría no nos permite notar la diferencia.



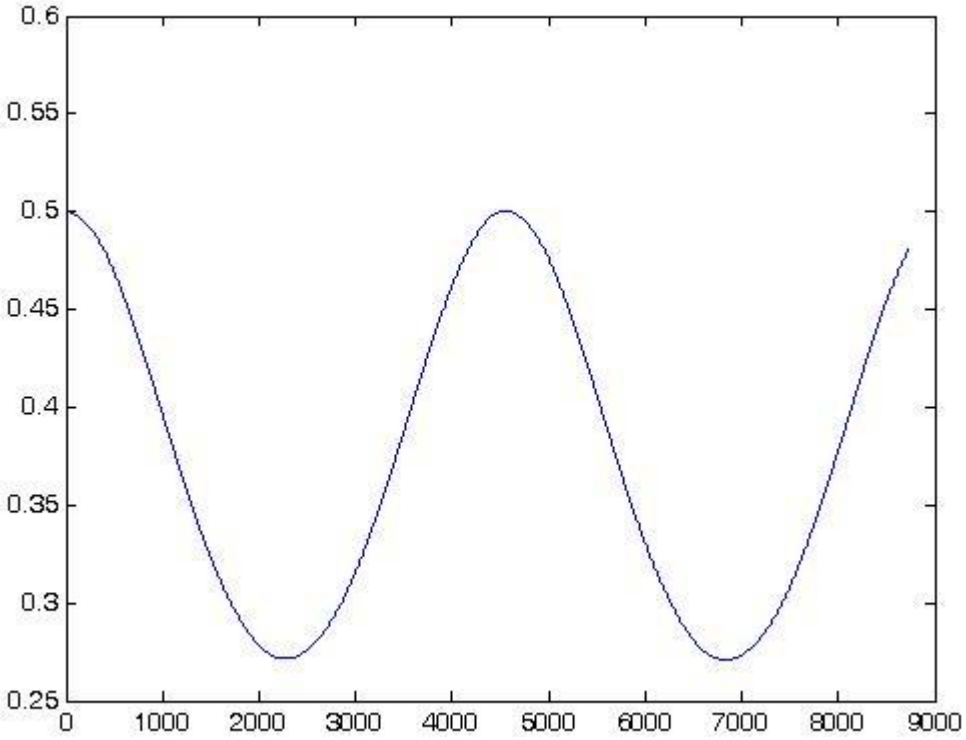
2) Un período para y(0) = 5, se verifica que cuando se calcula para y(0) = -5 la grafica es idéntica aunque invertida:

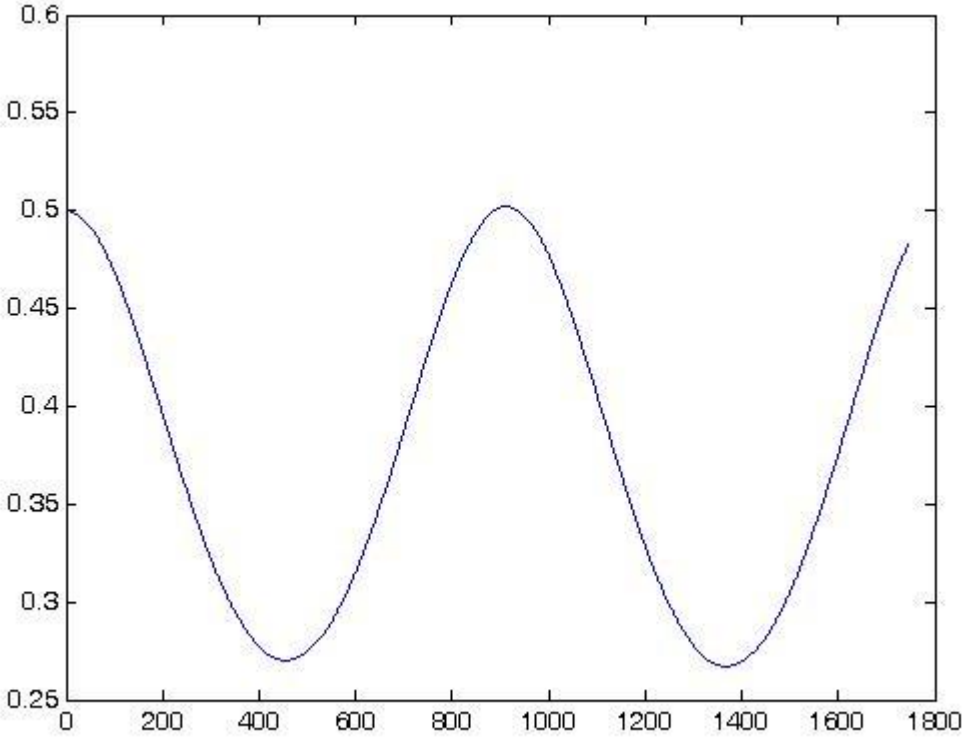




3) y(0) = 0.5, periodo= 0.71569 seg para t = 1.43138 (2ciclos)usando siempre el paso h recomendado, esta vez solicitando que realice 4000 iteraciones por ciclo en el primer gráfico, y 800 iteraciones por ciclo en el segundo.

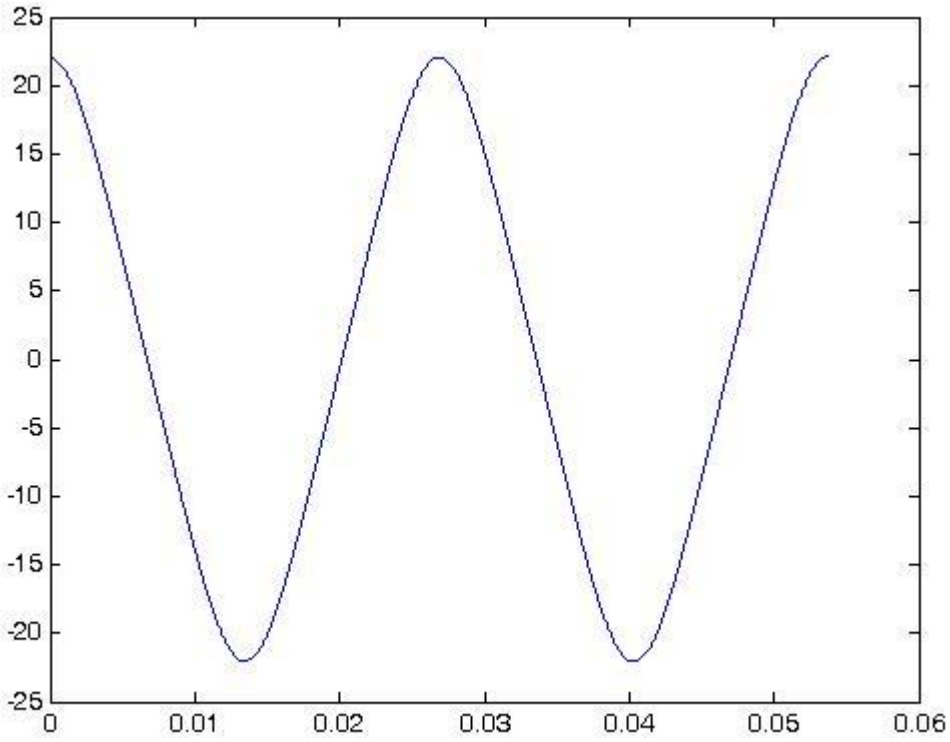
En el primero llega en 8728 iteraciones (728 iteraciones de más), y en el segundo en 1747 iteraciones (147 iteraciones de más).

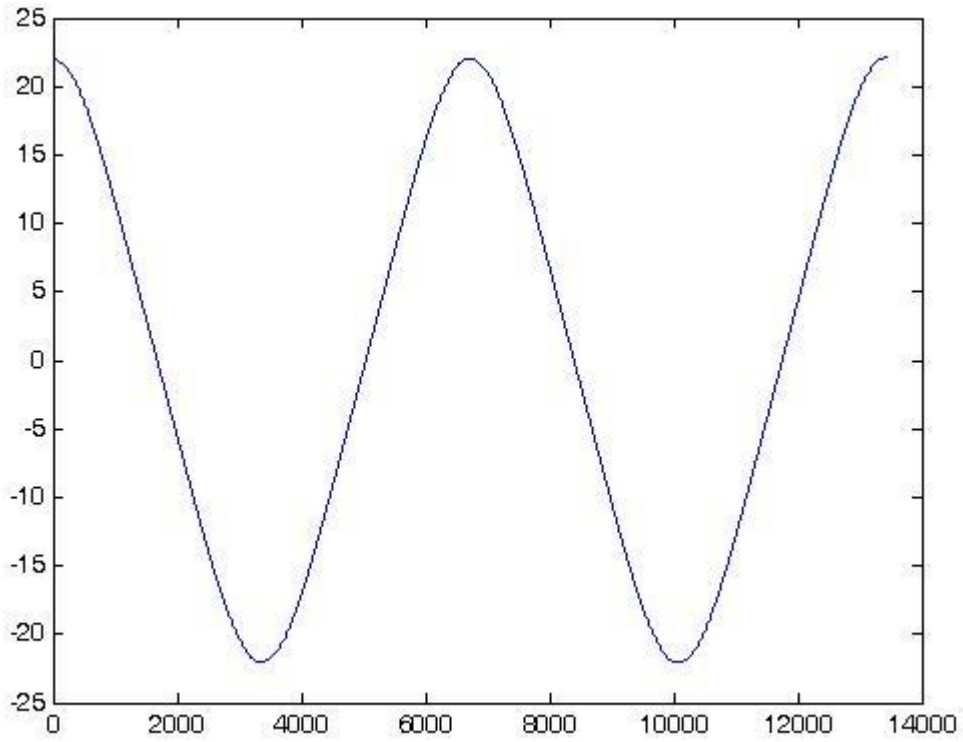

0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0
1000
2000
3000
4000
5000
6000
7000
8000
9000


0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0
200
400
600
800
1000
1200
1400
1600
1800

4) y(0) = 22m, periodo = 0.02684 , h = 0.000004 (recomendado para 6000 iteraciones por ciclo) , t = 0.05368 (Periodo*2), el programa concluye en 13420 (el error está en la estimación de iteraciones por ciclo en este caso en 1420 iteraciones más).

Se muestran las gráficas de la posición en el eje de las ordenadas, con el tiempo y las iteraciones en el eje de las abscisas, respectivamente.





En la carpeta de la presentación del informe está la hoja de cálculos de Excel, donde simplemente reemplazando los valores de y(0), h y, t te muestra los resultados por pantalla.También se encuentran en un .rar las gráficas correspondientes a las comprobaciones aquí presentadas, que por un problema de soft han sido manipuladas debilitando su resolución.

Aclaración: las comprobaciones aquí presentadas son las consideradas más pertinentes a simple vista, pero el programa ha sido testeado en varias ocasiones comparando sus resultados con Excel y matlab, donde la presentación de dichas experiencias son muy extensas para exponer en éste informe, ya que involucraban una cantidad de cálculos donde no alcanzarían las hojas para presentar los resultados.

CONCLUSION

El trabajo de laboratorio de métodos numéricos fue en general muy enriquecedor, tanto en la construcción de un algoritmo para luego ponerlo en marcha mediante un programa (PythonG) para que, por el método de Euler resolver una ecuación diferencial de segundo orden con aplicación a la física, pero también por la manera de como presentar dicha información dentro del programa, y en explayar toda la experiencia en un informe detallado viéndose uno forzado a explicar cosas que uno da por sabidas, haciéndolo entendible para cualquier persona con conocimientos básicos en informática aunque éste desconozca el método.

La idea principal fue crear un programa con la mayor automatización posible en sus cálculos, lo cual fue muy accesible debido a que el método de Euler tiene un desarrollo muy sencillo y da muestra de cómo resultados aproximados pueden ser tan semejantes a la realidad, tanto como para tomar el error absoluto como despreciable para todos los fines prácticos.

Fueron necesarias varias horas frente a la máquina para asegurarse que todo funcionara correctamente, y para verificar que la información obtenida era la buscada, además de todo el trabajo de escritorio para dar forma a éste proyecto de forma seria, tanto como para entender por completo los fines prácticos del método, sino también para su presentación ante el profesor que finalmente será quien evaluara todo el proceso y el resultado final.

Vale aclarar que la forma de resolver el problema de flotante cónico no es única, creo que podría haber muchas versiones diferentes al programa aquí presentado siendo todas éstas igual de válidas que flotante_conico.py mientras que los resultados sean también comprobables.

Para consultas o información sobre la codificación del algoritmo af.stigliano@gmail.com.