Aula 13

Melhorando a solução de equações diferenciais com aproximações de 2ª e 4ª ordem

Aos 40 minutos terei de reiniciar a sessão zoom... e vocês tem que se ligar de novo!

Próxima aula será em skype meet-now: enviarei um link (vamos ver se funciona sem limite de 40 min)

De volta à lei de Newton do arrefecimento

$$\frac{dT}{dt} = -\alpha(T - T_{ar})$$

Solução exata:

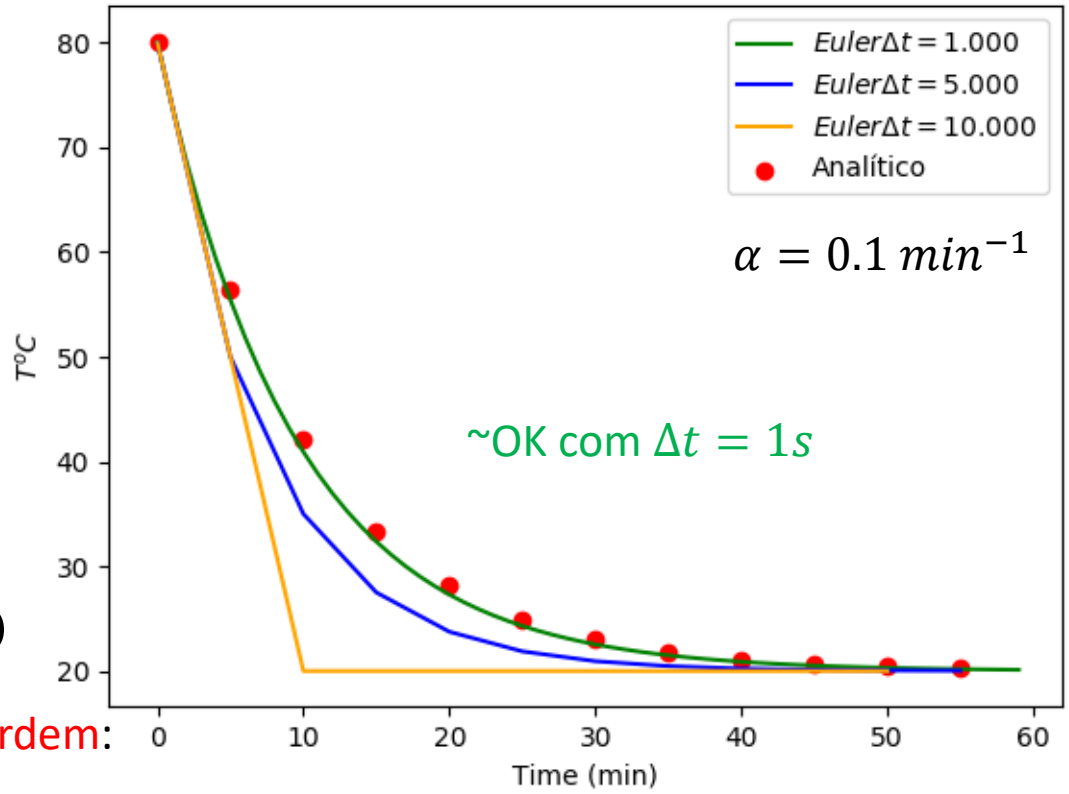
$T = (T_0 - T_{ar})e^{-\alpha t}$

Método de Euler:

$T(t + \Delta t) = T(t) - \alpha \Delta t(T(t) - T_{ar})$

Trata-se de uma aproximação de 1ª ordem:

$$\left(\frac{dT}{dt}\right)_{t=t} \approx \frac{T(t + \Delta t) - T(t)}{\Delta t} + E(\Delta t)$$



Diferenças centradas (2ª ordem)

$$\left(\frac{dy}{dt}\right)_{t=a} \approx \frac{y(a+\Delta t) - y(a-\Delta t)}{2\Delta t} + E\left(\Delta t^2\right)$$

Que se pode escrever:

$$\left(\frac{dy}{dt}\right)_{t=a+\Delta t/2} \approx \frac{y(a+\Delta t/2) - y(a-\Delta t/2)}{\Delta t} + E\left(\Delta t^2\right)$$

ou

$$T(t+\Delta t) = T(t) - \alpha\Delta t\left(T\left(t+\frac{\Delta t}{2}\right) - T_{ar}\right)$$

$$T(t+\Delta t) = T(t) - \alpha\Delta t\left(\frac{T(t+\Delta t) + T(t)}{2} - T_{ar}\right)$$

$$T(t+\Delta t)\left(1+\frac{\alpha\Delta t}{2}\right)=T(t)-\alpha\Delta t\left(\frac{T(t)}{2}-T_{ar}\right)$$

ou

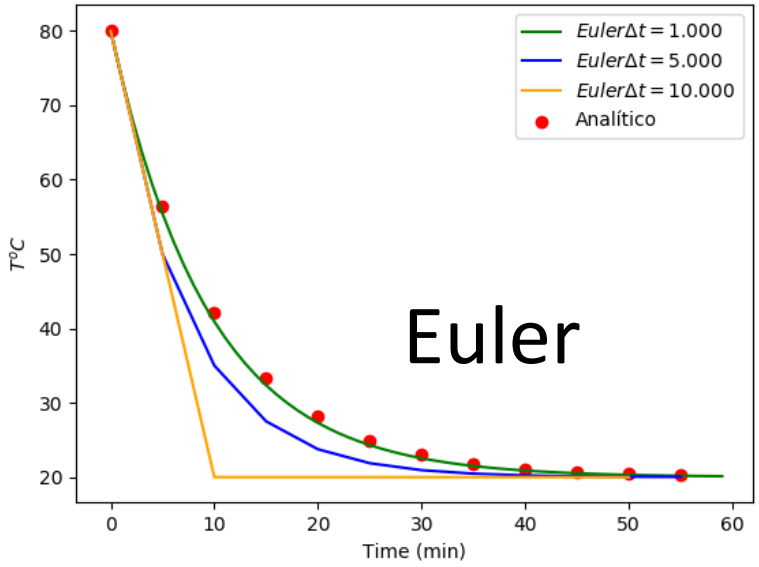
$$T(t+\Delta t)=\left[T(t)-\alpha\Delta t\left(\frac{T(t)}{2}-T_{ar}\right)\right]\left(1+\frac{\alpha\Delta t}{2}\right)^{-1}$$

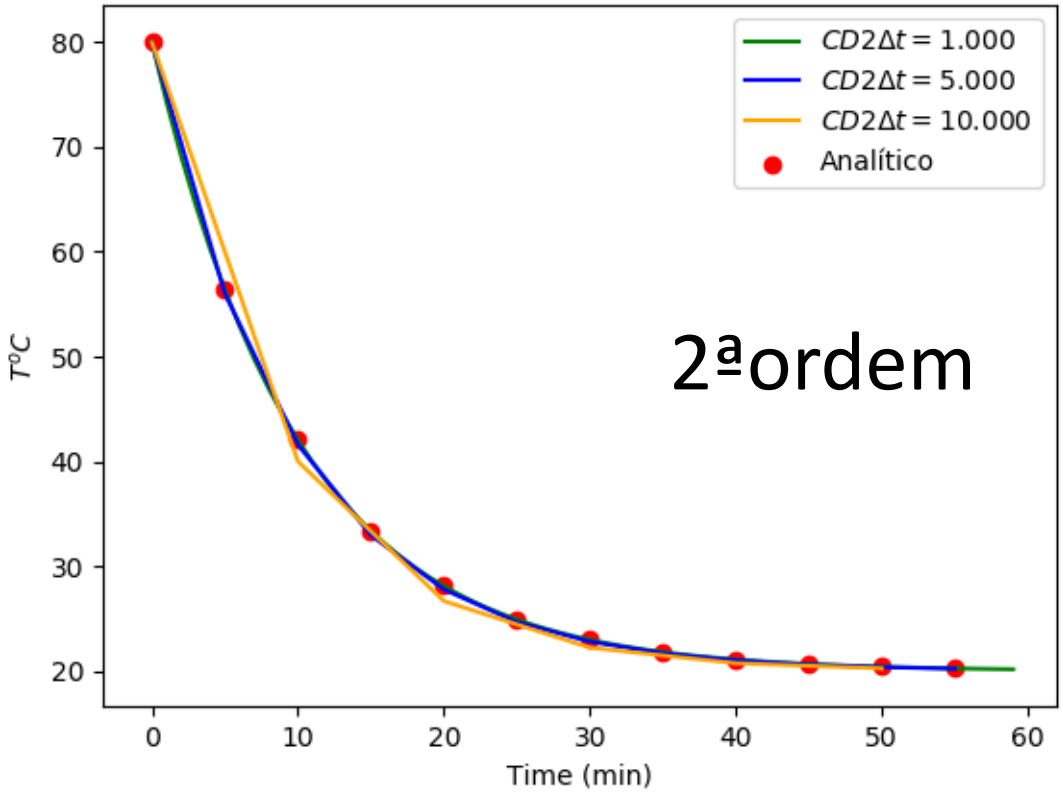
Em vez de (Euler):

$$T(t+\Delta t)=T(t)-\alpha\Delta t(T(t)-T_{ar})$$

```
kc=0
for dt in dts:
    tempo=np.arange(0.,timemax,dt)
    n=len(tempo)
    T=np.zeros(tempo.shape)
    T[0]=Tinicial
    for k in range(1,n):
        T[k]=(T[k-1]-alpha*(0.5*T[k-1]-Tar)*dt)\
             /(1+0.5*alpha*dt)
    plt.plot(tempo,T,color=cores[kc],\
             label=r'$CD2 \Delta t=%6.3f$'%(dt))
    plt.xlabel('Time (min)')
    plt.ylabel(r'$T ^{o} C$')
    kc=kc+1
```

Dif Centrada vs Euler





Perfeito com $\Delta t = 5s$

Aproximação de 4ª ordem de Runge-Kutta

$$\frac{\partial \phi}{\partial t} = f(\phi, t), \Delta t = t^{n+1} - t^n$$

$$k_1 = \Delta t \, f(\phi^n, t^n)$$

$$k_2 = \Delta t \, f\left(\phi^n + \frac{k_1}{2}, t^n + \frac{\Delta t}{2}\right)$$

$$k_3 = \Delta t \, f\left(\phi^n + \frac{k_2}{2}, t^n + \frac{\Delta t}{2}\right)$$

$$k_4 = \Delta t \, f(\phi^n + k_3, t^n + \Delta t)$$

$$\phi^{n+1} = \phi^n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}$$

Runge-Kutta

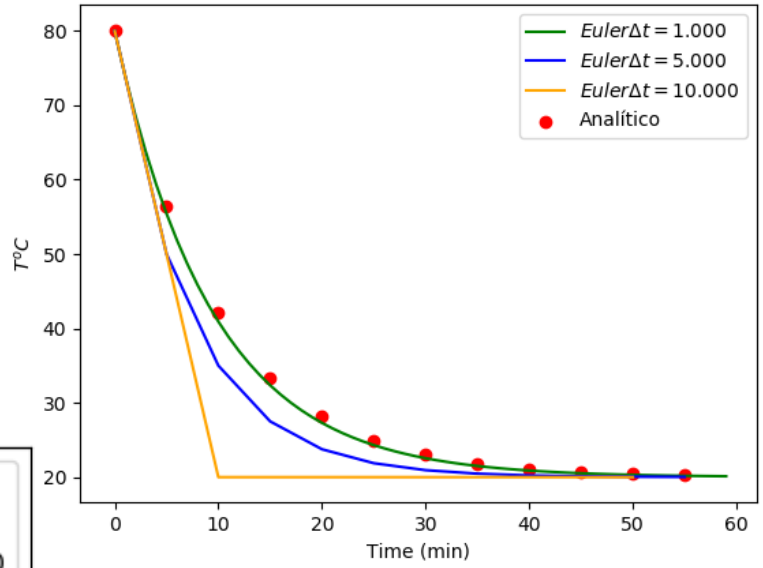
```
def RK4x(x,dxdt,dt):
    k1=dxdt(x)*dt
    k2=dxdt(x+k1/2)*dt
    k3=dxdt(x+k2/2.)*dt
    k4=dxdt(x+k3)*dt
    xP=x+1./6.*(k1+2*k2+2*k3+k4)
    return xP
def dTdt(T):
    Tar=20.
    alpha=0.1
    c=-alpha*(T-Tar)
    return c
```

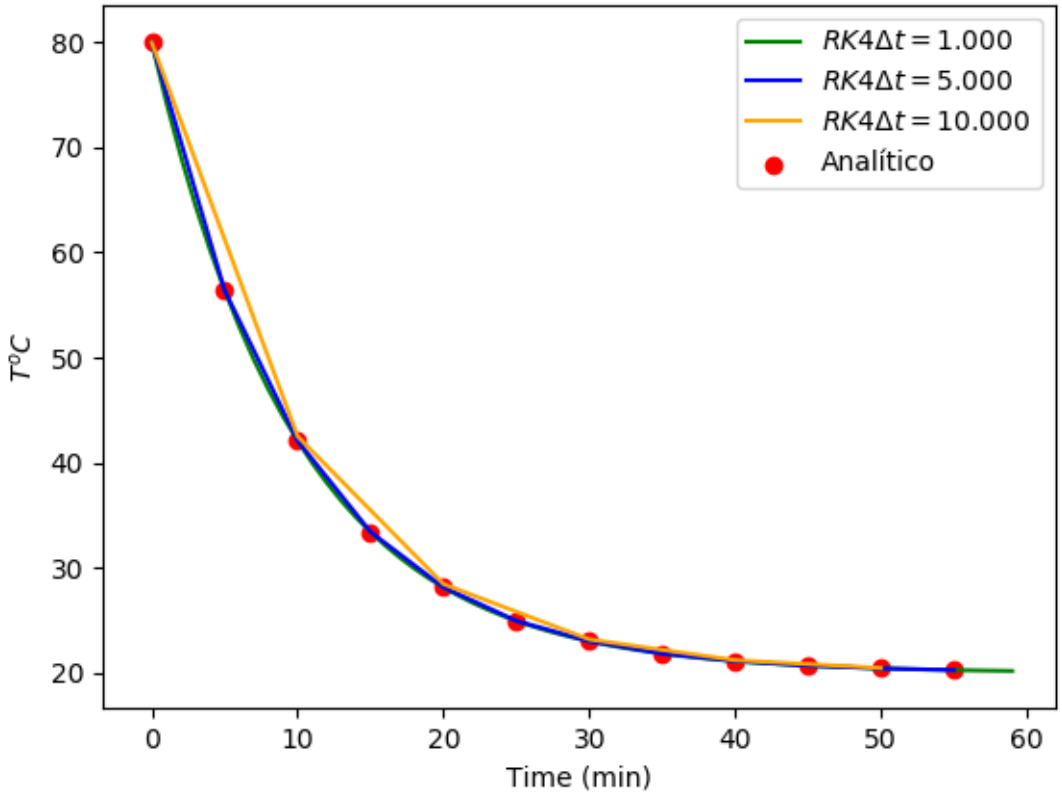
$$\frac{\partial \phi}{\partial t} = f(\phi, t)$$

Não depende de t

```
kc=0
for dt in dts:
    …
    T[0]=Tinicial
    for k in range(1,n):
        T[k]=RK4x(T[k-1],dTdt,dt)
    …
```

RK4 vs Euler





Perfeito com $\Delta t = 10s$

Um problema mais comlplicado: O pêndulo gravítico

$$\vec{a} = \frac{\vec{F}}{m}$$

Velocidade angular:

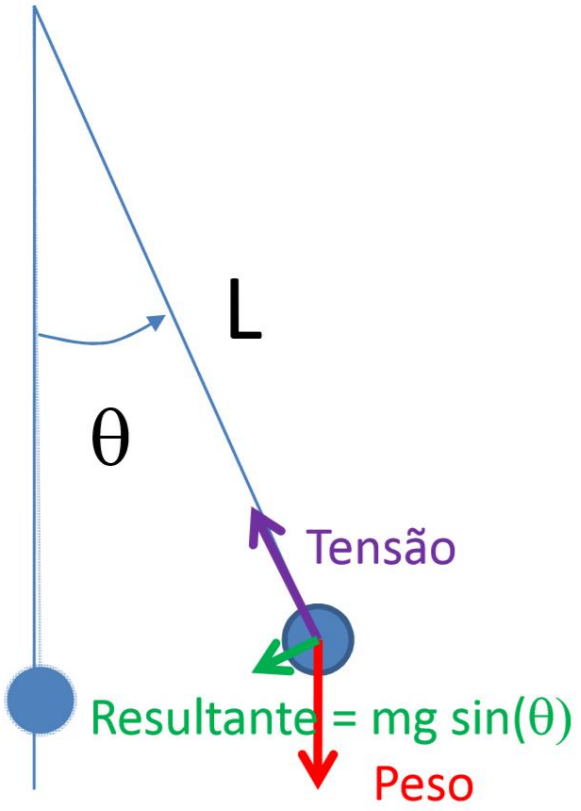
$$\omega = \frac{d\theta}{dt}$$

velocidade linear:

$$v = L\omega$$

Lei de Newton:

$$a = \frac{dv}{dt} = \frac{F}{m} = -g \sin\theta$$



Limite das pequenas oscilações

$$\frac{dv}{dt} = L\frac{d\omega}{dt} = L\frac{d^2\theta}{dt^2} = -g\,\sin\theta$$

No caso de o pêndulo oscilar muito perto do equilíbrio (θ pequeno), podemos usar a aproximação:

$$\sin\theta = \theta + \frac{\theta^3}{6} + \frac{3\theta^5}{40} + \frac{5\theta^7}{112} + \cdots \approx \theta$$

E a equação reduz-se à equação do oscilador harmónico:

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\theta = 0$$

Lembra-se que na lei de Hooke (mola ideal) se tinha:

$$\frac{d^2x}{dt^2} + \frac{k}{m}x = 0$$

Solução analítica (θ pequeno)

Pode verificar-se (por substituição) que a equação do oscilador harmónico tem, neste caso, a solução

$$\theta(t) = A\cos\left(\frac{2\pi t}{T} + \phi\right)$$

Onde o período é dado por (não depende da massa, nem da amplitude):

$$T = 2\pi\sqrt{\frac{L}{g}}$$

e A(amplitude) e ϕ (fase inicial), dependem das condições iniciais.

Se $\omega(t = 0) = 0$, $A = |\theta_0|$

A solução numérica permite…

Estudar o comportamento do pêndulo para oscilações de grande amplitude.

Neste caso a solução não é uma sinusoide mas é uma função periódica cujo período pode ser obtido analiticamente como: θ_0 é a amplitude

$$T = 4\sqrt{\frac{L}{g}}\int_{\theta}^{\theta_0}\frac{1}{\sqrt{\theta-\theta_0}}d\theta$$

Notar que $\lim_{\theta_0\to\pi} T = \infty$

Pode mostrar-se que:

$$T = 2\pi\sqrt{\frac{L}{g}}\sum_{n=0}^{\infty}\left(\left(\frac{(2n)!}{(2^n n!)^2}\right)^2 \sin^{2n}\frac{\theta_0}{2}\right)$$

Energia

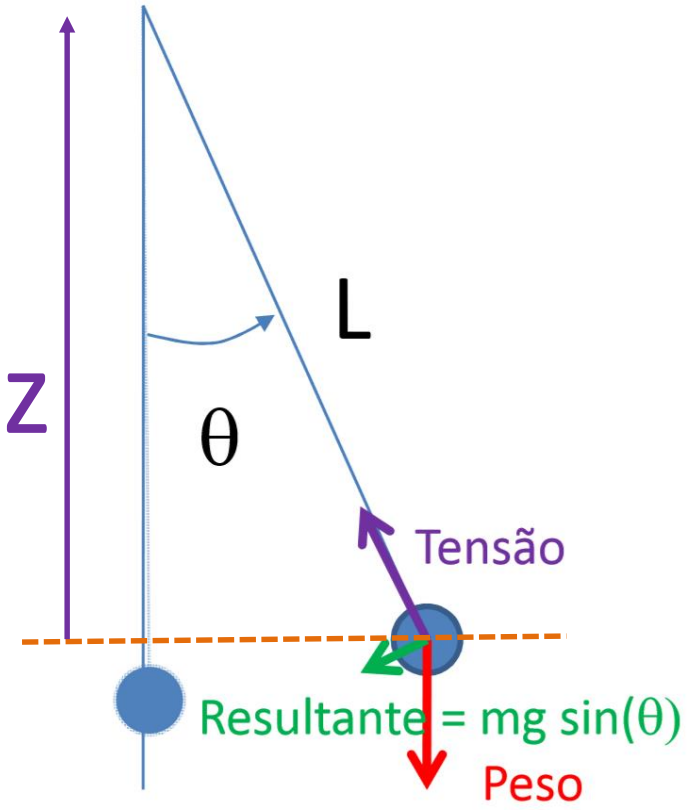
Apesar de não existir uma solução explícita, existe a garantia de que o pêndulo deve conservar energia mecânica:

$$\frac{E_M}{m} = \frac{1}{2}v^2 + gz = \frac{1}{2}(L\omega)^2 + g(L - L\cos\theta)$$

$$\omega = \frac{d\theta}{dt}$$

$$v = L\omega$$

Onde se convencionou $z = 0$, em $\theta = 0$.



Simulação python

```
import numpy as np
import matplotlib.pyplot as plt
L=1 #comprimento do pêndulo
g=9.80665;nIMP=2
theta0=-5./180.*np.pi #amplitude 5 graus em radianos
omega0=0. #velocidade angular inicial
T=2*np.pi*np.sqrt(L/g) #periodo (solução analítica)
t=np.linspace(0.,10*T,1001) #vetor de tempos (10T)
dt=t[1]-t[0] #passo de tempo
n=len(t)
theta=np.zeros(t.shape); omega=np.copy(theta)
theta[0]=theta0; omega[0]=omega0
domovie=False #opcional
pngs=[] #lista de frames
```

Parte 2 (1º passo Euler, depois ponto médio)

```
for kt in range(1,n):
    omega[kt]=omega[kt-1]-g/L*np.sin(theta[kt-1])*dt
    theta[kt]=theta[kt-1]\
              +0.5*(omega[kt]+omega[kt-1])*dt
    for improve in range(nIMP):
        omega[kt]=omega[kt-1]-g/L*np.sin\
                 (0.5*(theta[kt-1]+theta[kt]))*dt
        theta[kt]=theta[kt-1]\
                  +0.5*(omega[kt]+omega[kt-1])*dt
```

Parte 3 (frames)

```
if domovie:
    xis=L*np.sin(theta[kt])
    yps=-L*np.cos(theta[kt])
    plt.scatter([-L*1.2,L*1.2],[-L*1.2,1.2*L],color='white')
    # marca espaço para fixar o tamanho da imagem
    plt.plot([0,xis],[0,yps],color='black')
    plt.scatter([xis],[yps],color='red',s=200,zorder=2)
    plt.title(r'$t=%4.2f $'%(t[kt]))
    plt.axis('equal')
    plt.pause(0.001)
    frame='Pend'+str(kt)+'.png'
    pngs.append(frame)
    plt.savefig(frame)
    plt.clf()
```

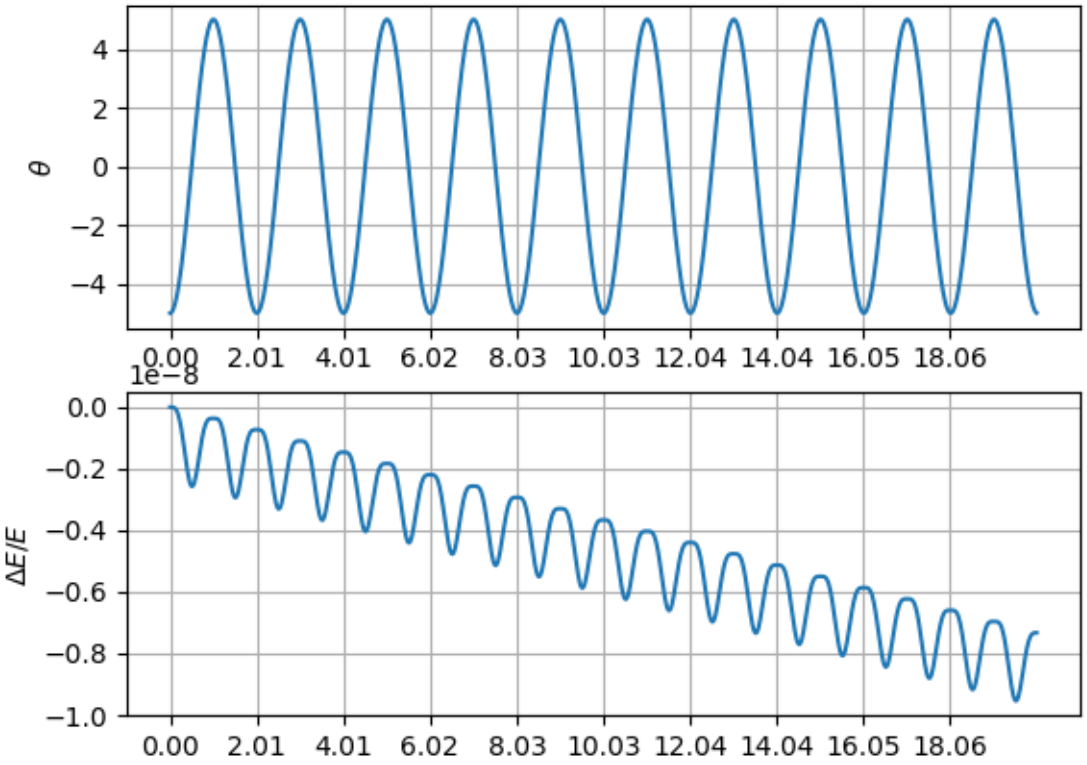
Parte 4 (movie)

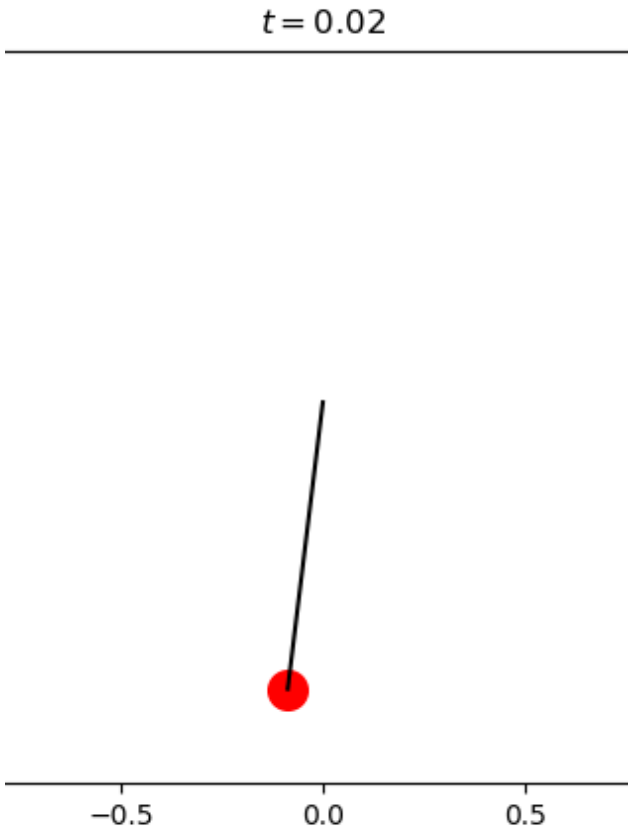
```
import imageio
import os
if len(pngs)!=0: # caso domovie==False
    for frame in pngs:
        images.append(imageio.imread(frame))
        os.remove(frame) #limpa espaço de disco
    imageio.mimsave(movie+'.gif',\
                    images,duration=0.05)
```

Gráficos finais

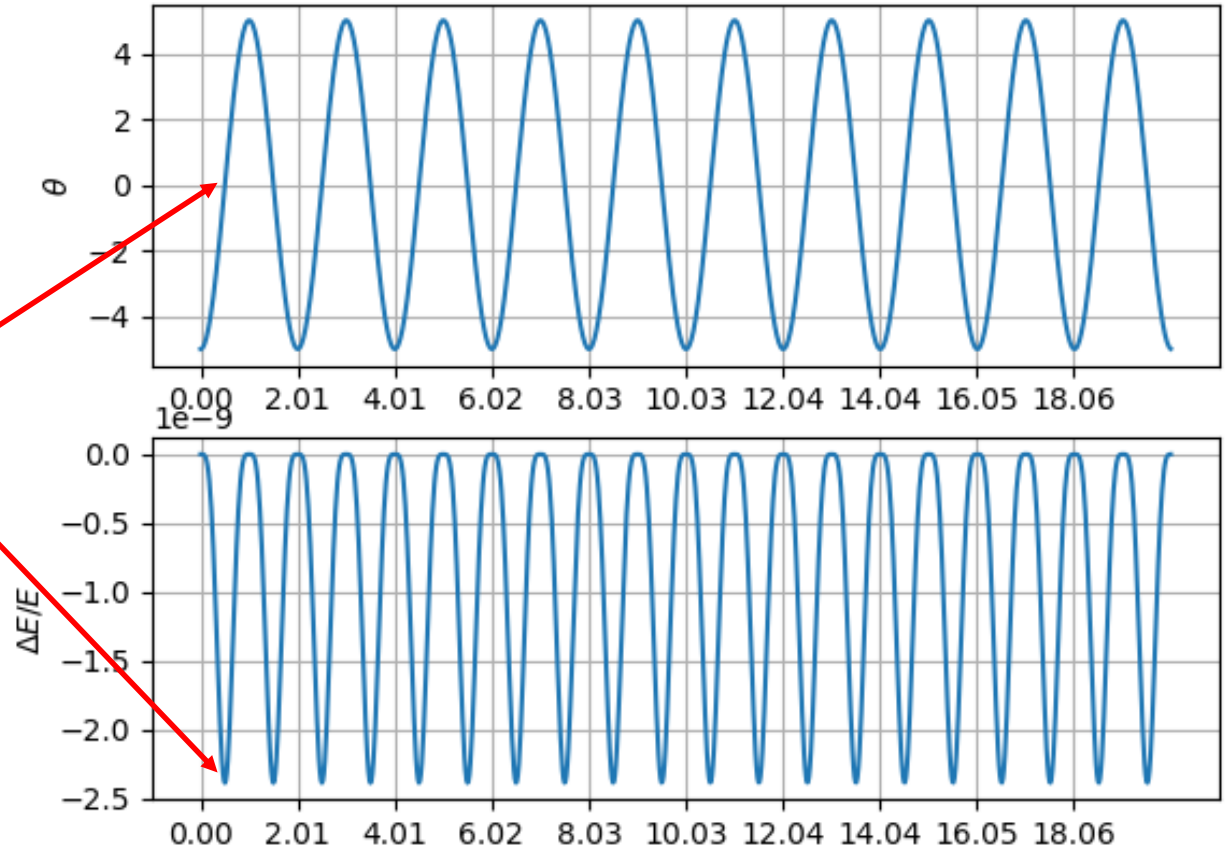
```
plt.figure(2)
plt.subplot(2,1,1)
plt.plot(t,theta*180./np.pi); #conversão para graus
plt.xticks(np.arange(min(t),max(t),T)) #periodos
plt.grid()
plt.ylabel(r'$\theta$')
plt.subplot(2,1,2)
EM=0.5*(L*omega)**2-L*np.cos(theta)*g
plt.plot(t,(EM-EM[0])/EM[0]); #erro relativo
plt.xticks(np.arange(min(t),max(t),T))
plt.grid()
plt.ylabel(r'$\Delta E/E$')
```

Pequena amplitude $\theta_0 = 5°$ `nIMP=2`



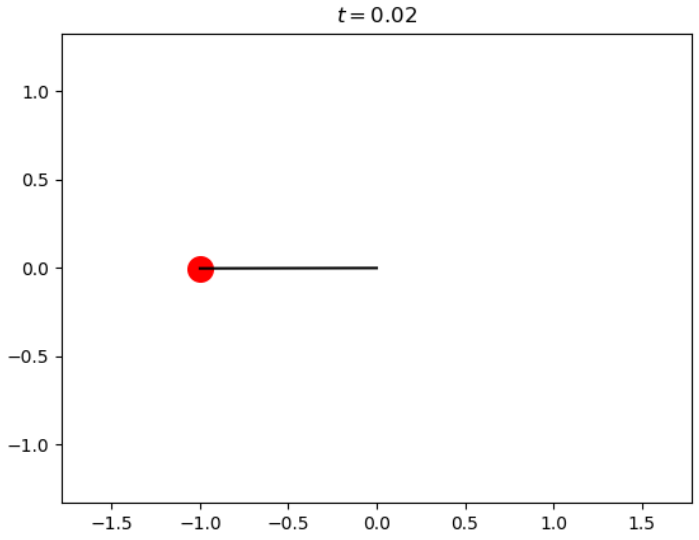


Pequena amplitude $\theta_0 = 5°$ `nIMP=5`

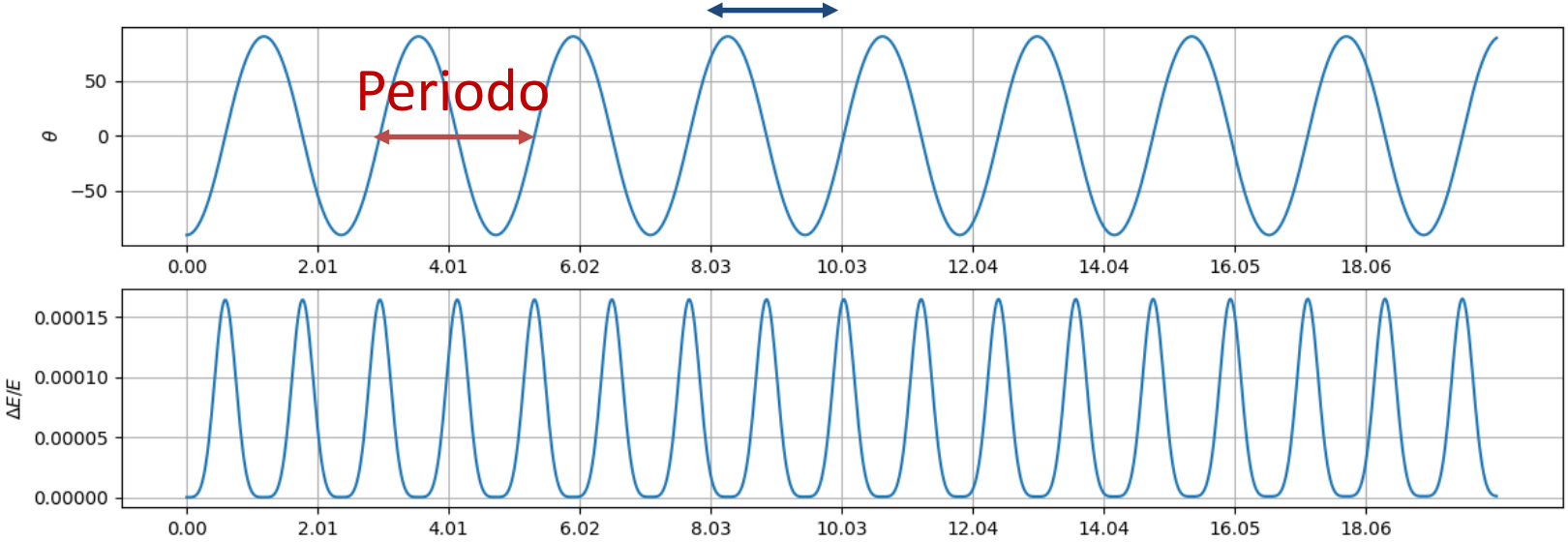


Max velocidade

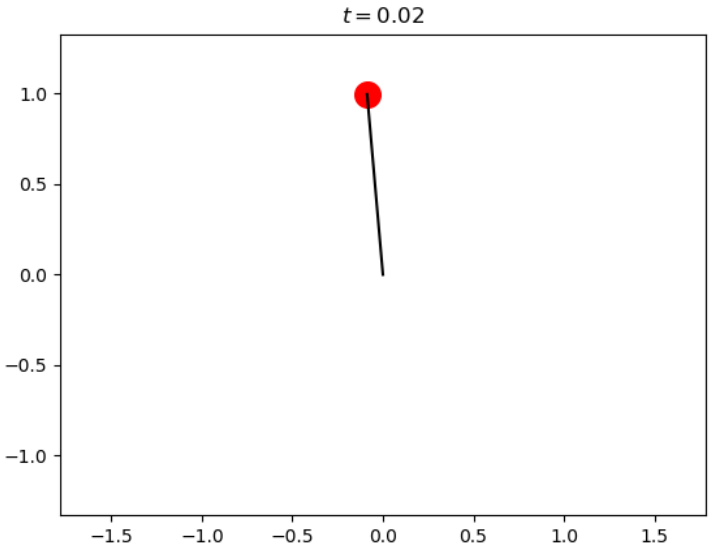
nIMP=2, $\theta_0 = 90°$



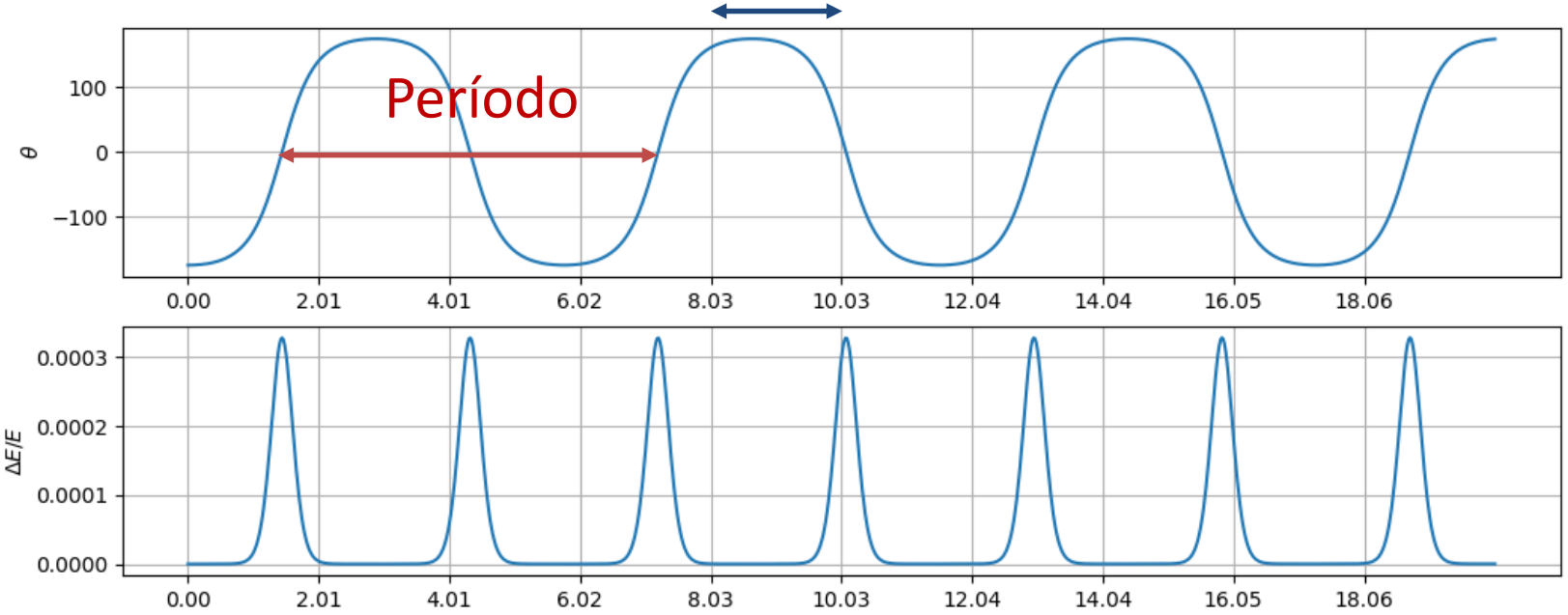
$$2\pi\sqrt{\frac{L}{g}}$$



nIMP=2, $\theta_0 = 175°$

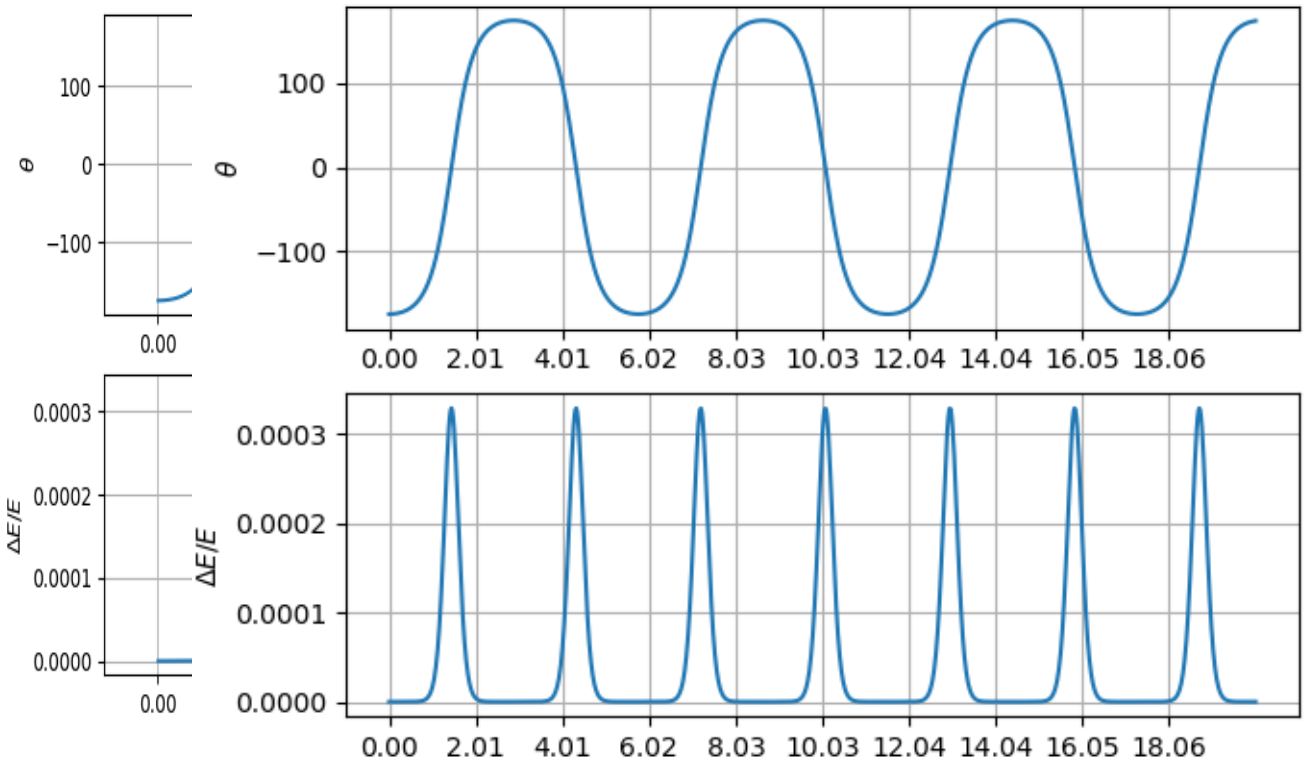


$$2\pi\sqrt{\frac{L}{g}}$$



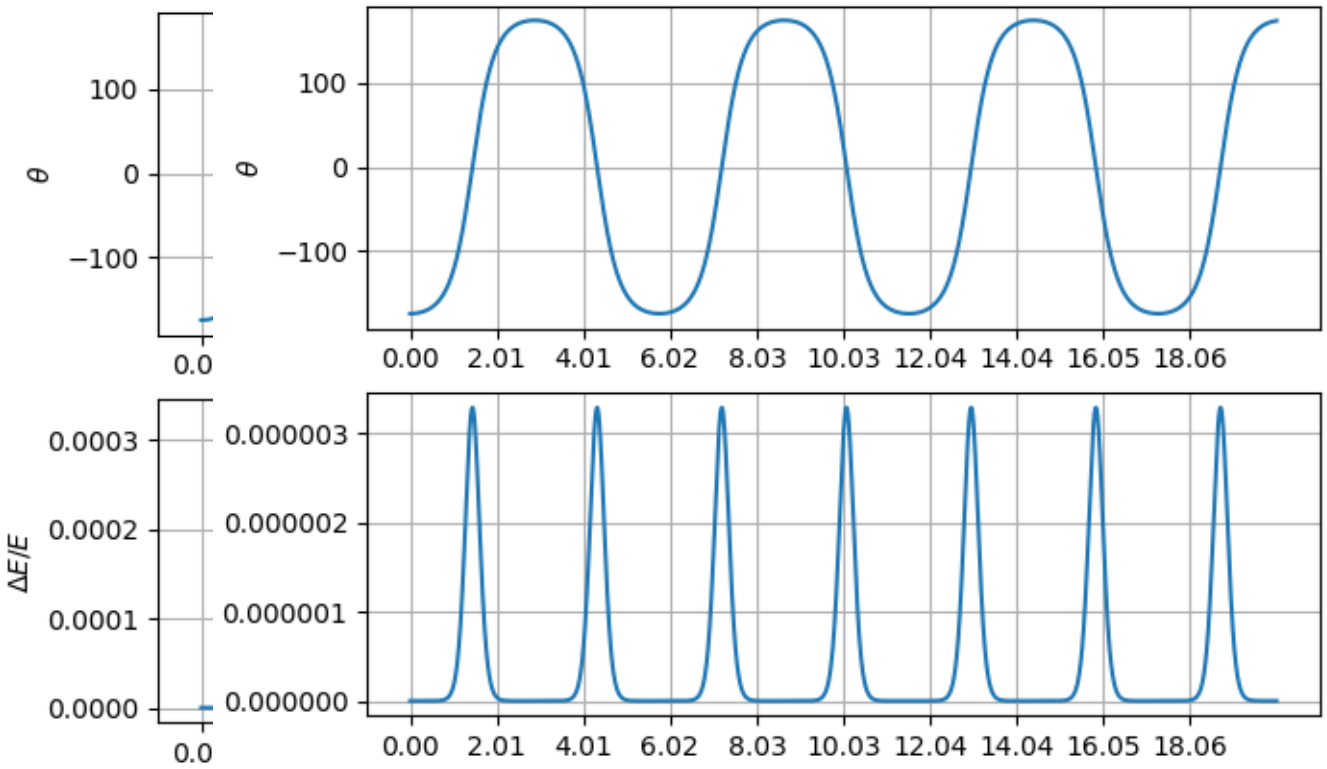
Tentando melhorar a conservação de Energia

nIMP=10, $\Delta t = \frac{T_{ana}}{100}$ (não tem impacto)



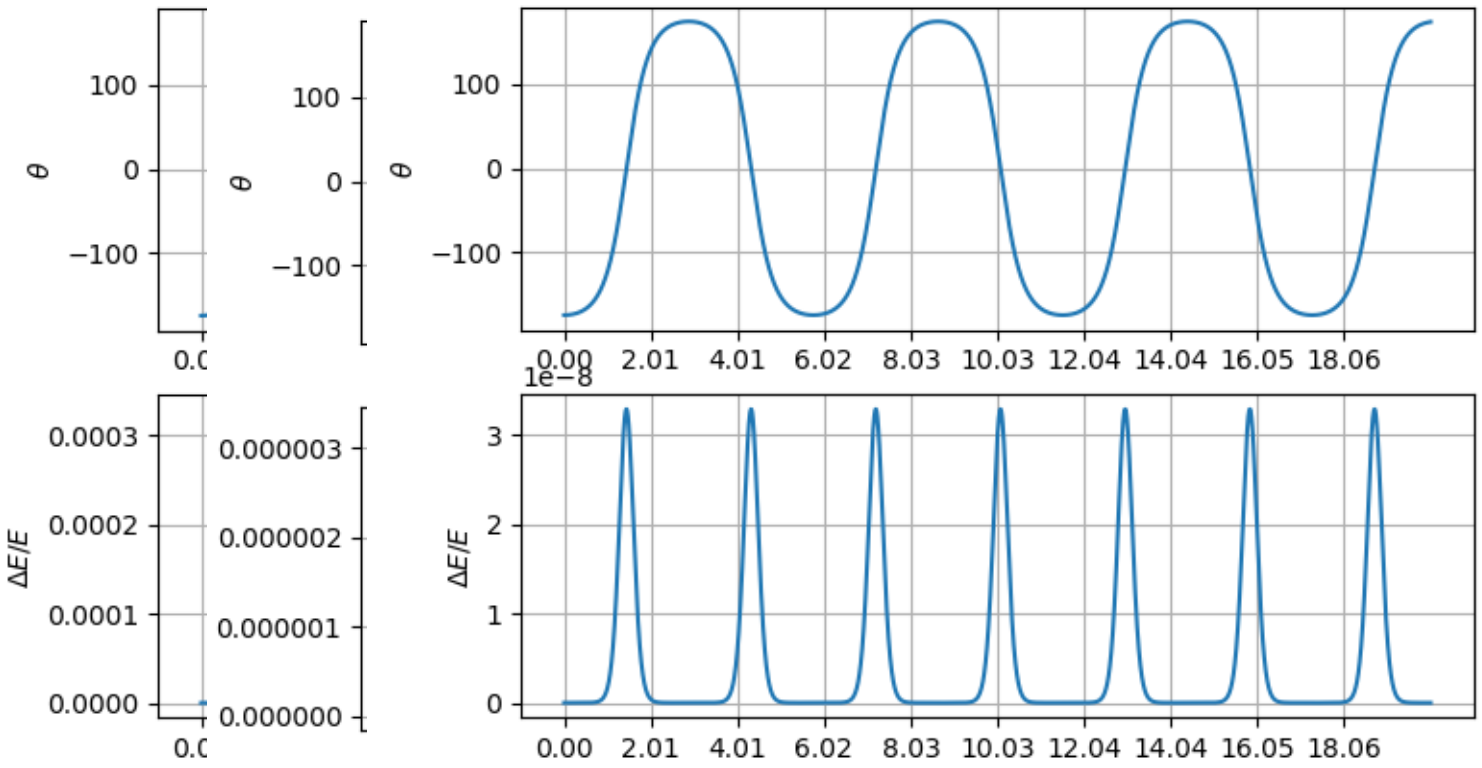
Tentando melhorar a conservação de Energia

nIMP=2, $\Delta t = \frac{T_{ana}}{1000}$

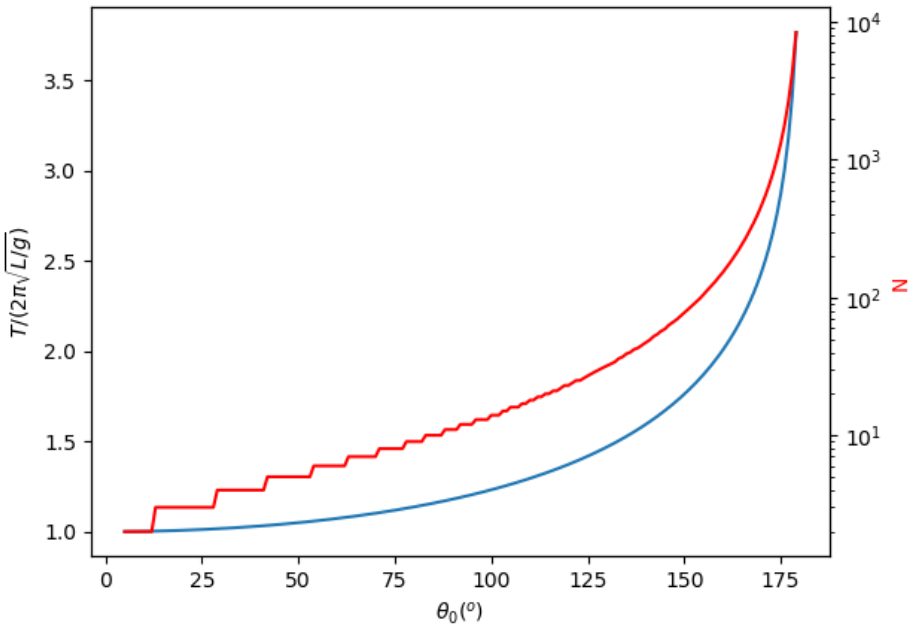


Tentando melhorar a conservação de Energia

nIMP=2, $\Delta t = \frac{T_{ana}}{10000}$



$$T = 2\pi\sqrt{\frac{L}{g}}\sum_{n=0}^{\infty}\left(\left(\frac{(2n)!}{(2^n n!)^2}\right)^2 \sin^{2n}\frac{\theta_0}{2}\right)$$

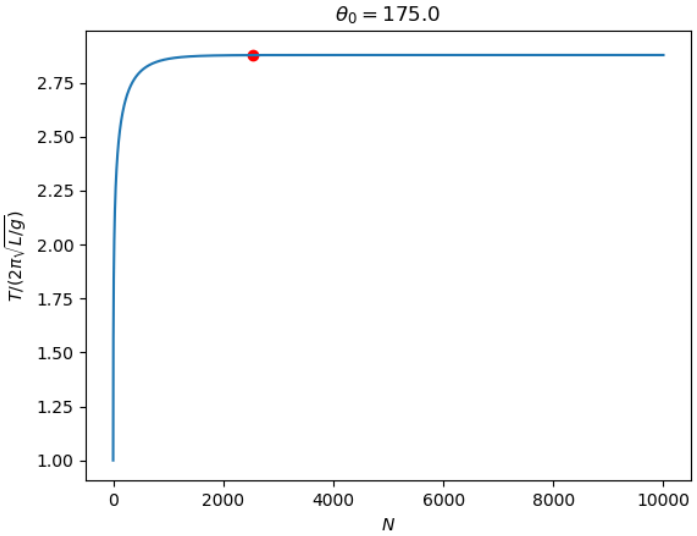


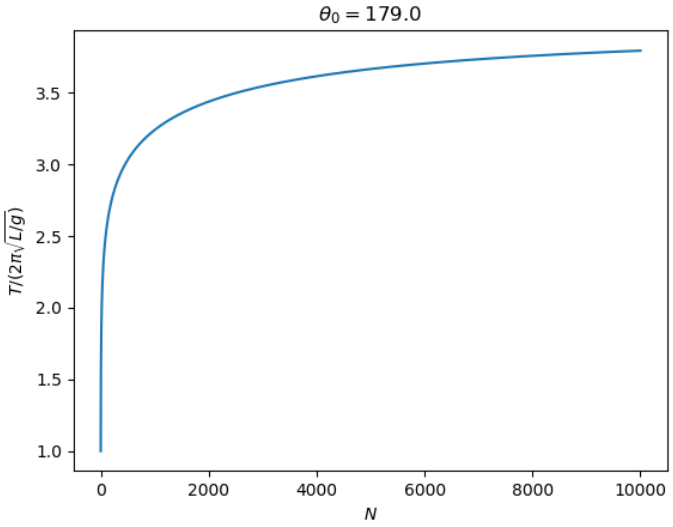

```
plt.figure(3);fig,ax1=plt.subplots()
angs=np.arange(5.,180.,1.)*np.pi/180.
Ns=np.zeros(angs.shape,dtype=int);
Ts=np.zeros(angs.shape,dtype=float)
from math import factorial
k=-1;T=2*np.pi*np.sqrt(L/g)
for theta0 in angs:
    k=k+1;  Ts[k]=0
    maxERR=1e-6
    N=-1;SER=1e10
    while SER>maxERR:
        N=N+1
        SER=(factorial(2*N)/(2**N*factorial(N))**2)**2*(np.sin(theta0/2))**(2*N)
        Ts[k]=Ts[k]+SER
    Ts[k]=Ts[k]*T; Ns[k]=N
ax1.plot(angs/np.pi*180,Ts/T)
ax1.set_xlabel(r'$\theta_0 (^{o})$');ax1.set_ylabel(r'$T/(2\pi \sqrt{L/g})$')
ax2=ax1.twinx()
ax2.plot(angs/np.pi*180,Ns,color='red')
ax2.set_yscale('log');ax2.set_ylabel('N',color='red')
```

$$T = 2\pi\sqrt{\frac{L}{g}}\sum_{n=0}^{\infty}\left(\left(\frac{(2n)!}{(2^n n!)^2}\right)^2 \sin^{2n}\frac{\theta_0}{2}\right)$$

#CONVERGÊNCIA DA SÉRIE pode ser LENTA

```
plt.figure(5)
fig,ax=plt.subplots()
Nm=10000;TH0=175.*np.pi/180.
plt.title(r'$\theta_0=%4.1f$'%(TH0*180/np.pi))
TNs=np.zeros((Nm+1),dtype=float)
k=0
TNs[0]=T*(factorial(2*k)/(2**k*factorial(k))**2)\
       **2*(np.sin(TH0/2))**(2*k)
for k in range(1,Nm+1):
    TNs[k]=TNs[k-1]+T*(factorial(2*k)\
           /(2**k*factorial(k))**2)**2\
           *(np.sin(TH0/2))**(2*k)
ax.plot(np.linspace(0,Nm,Nm+1),TNs/T)
ax.set_xlabel(r'$N$');
ax.set_ylabel(r'$T/(2\pi \sqrt{L/g})$');
```





O pêndulo gravítico de grande amplitude

Constitui um problema fortemente não linear.

Os métodos simples aqui descritos são suficientemente bons para descrever o seu comportamento, mas só conservam energia com elevada precisão se o passo de tempo for muito pequeno.