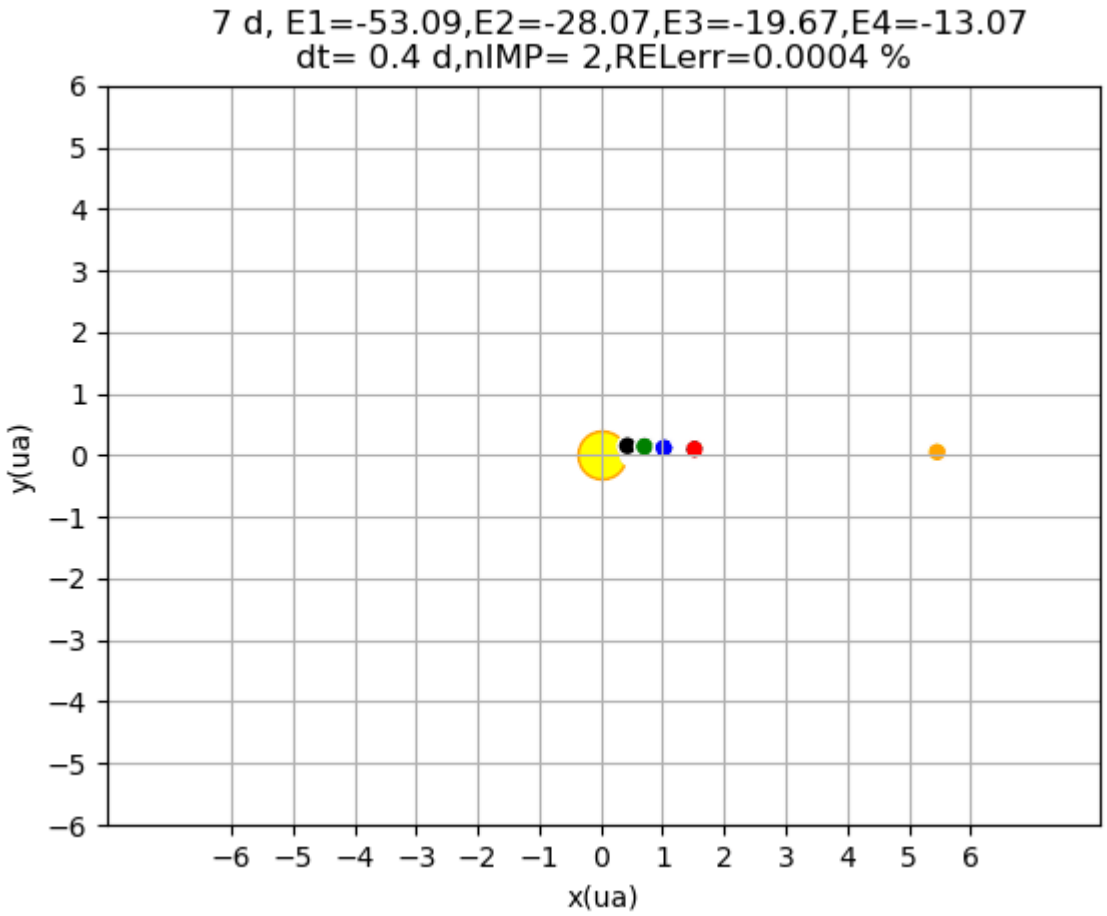
Aula 10

Equações diferenciais com condições fronteira num ponto



Problemas de valores iniciais

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

$$\frac{d^2\vec{r}}{dt^2} = \vec{g} \quad , \qquad \vec{g} = const$$

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

$$\frac{d^2\vec{r}}{dt^2} = \vec{g} = g_x\vec{e}_x + g_y\vec{e}_y, \qquad \vec{g} = -\frac{GM_{Sol}}{d^2}\vec{u}$$

$$\frac{d^2\vec{r}}{dt^2} = \frac{d\vec{v}}{dt} = \vec{g} - 2\vec{\Omega} \times \vec{v}$$

Arrefecimento de um corpo por convecção

Lei de Newton do arrefecimento

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

Em que α é o coeficiente de transferência e $T(t = 0) = T_0$ é a condição inicial.

Com $\alpha = const$, existe solução analítica:

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

A solução numérica de uma equação diferencial

Exige a sua transformação num sistema de equações algébricas.

A transformação mais simples consiste na utilização de diferenças finitas.

Diferenças finitas de 1ªordem

Série de Taylor

$$y(x+\Delta x) = y(x) + \frac{dy}{dx}\Delta x + \frac{1}{2}\frac{d^2y}{dx^2}\Delta x^2 + \frac{1}{3!}\frac{d^3y}{dx^3}\Delta x^3 + \cdots$$

Desprezando os termos de ordem 2 e superior:

$$y(x+\Delta x) = y(x) + \frac{dy}{dx}\Delta x + E(\Delta x^2)$$

ou

$$\left(\frac{dy}{dx}\right)_{x=a} \approx \frac{y(a+\Delta x) - y(a)}{\Delta x} + E(\Delta x)$$

Se for $T(t)$, temos a diferença avançada:

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

Diferenças finitas centradas (2ª ordem)

Fazendo

$$y(x+\Delta x) = y(x) + \frac{dy}{dx}\Delta x + \frac{1}{2}\frac{d^2y}{dx^2}\Delta x^2 + \frac{1}{3!}\frac{d^3y}{dx^3}\Delta x^3 + \cdots$$

$$y(x-\Delta x) = y(x) - \frac{dy}{dx}\Delta x + \frac{1}{2}\frac{d^2y}{dx^2}\Delta x^2 - \frac{1}{3!}\frac{d^3y}{dx^3}\Delta x^3 + \cdots$$

Subtraindo

$$y(x+\Delta x) - y(x-\Delta x) = 2\frac{dy}{dx}\Delta x + E(\Delta x^4)$$

ou

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

Diferenças finitas

Fazendo

$$y(x+\Delta x) = y(x) + \frac{dy}{dx}\Delta x + \frac{1}{2}\frac{d^2y}{dx^2}\Delta x^2 + \frac{1}{3!}\frac{d^3y}{dx^3}\Delta x^3 + \cdots$$

$$y(x-\Delta x) = y(x) - \frac{dy}{dx}\Delta x + \frac{1}{2}\frac{d^2y}{dx^2}\Delta x^2 - \frac{1}{3!}\frac{d^3y}{dx^3}\Delta x^3 + \cdots$$

Somando

$$y(x+\Delta x) + y(x-\Delta x) = 2y(x) + \frac{d^2y}{dx^2}\Delta x^2 + E(\Delta x^4)$$

ou

$$\left(\frac{d^2y}{dx^2}\right)_{x=a} \approx \frac{y(a+\Delta x) + y(a-\Delta x) - 2y(a)}{\Delta x^2} + E(\Delta x^2)$$

Método de Euler para o arrefecimento convectivo

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

Usamos diferenças avançadas (1ª ordem)

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

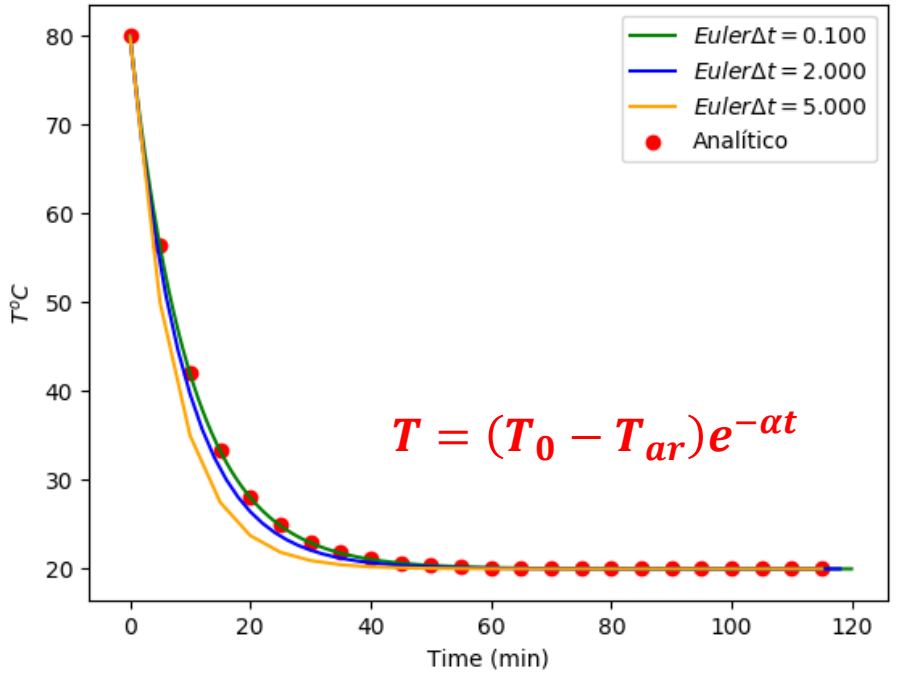
Resolvendo para o futuro temos uma recursão

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

A solução depende da condição inicial $T(t = 0) = T_0$ e dos parâmetros T_{ar}, α.

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

```
import numpy as np
import matplotlib.pyplot as plt
Tar=20; Tinicial=80; alpha=0.1
cores=['green','blue','orange']; kc=0
for dt in[0.1,2,5]:
    tempo=np.arange(0.,120.,dt)
    n=len(tempo)
    T=np.zeros(tempo.shape)
    T[0]=Tinicial
    for k in range(1,n): #Euler
        T[k]=T[k-1]-alpha*(T[k-1]-Tar)*dt
    plt.plot(tempo,T,color=cores[kc],label=r'$Euler \Delta t=%6.3f$'%(dt))
    plt.xlabel('Time (min)'); plt.ylabel(r'$T ^{o} C$');    kc=kc+1
tempoAn=np.arange(0.,120.,5)
Tan=Tar+(Tinicial-Tar)*np.exp(-alpha*tempoAn)
plt.scatter(tempoAn,Tan,color='red',label='Analítico')
plt.legend()
```



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

Movimento balístico $(g = const)$

$$\vec{F} = m\vec{a} \Longrightarrow \frac{d\vec{v}}{dt} = \frac{d^2\vec{r}}{dt^2} = \vec{g} = -g\vec{k}$$

Com $\vec{r} = x\vec{\imath} + y\vec{\jmath} + z\vec{k}$

Trata-se de uma equação (vetorial) de segunda ordem para a posição.

Vamos escrever como um sistema de 6 equações de 1ª ordem: de facto 2 são triviais ($u, v = const$).

$$\begin{cases} \frac{du}{dt} = 0 \\ \frac{dv}{dt} = 0 \\ \frac{dw}{dt} = -g \\ \frac{dx}{dt} = u \\ \frac{dy}{dt} = v \\ \frac{dz}{dt} = w \end{cases}$$

Método de Euler

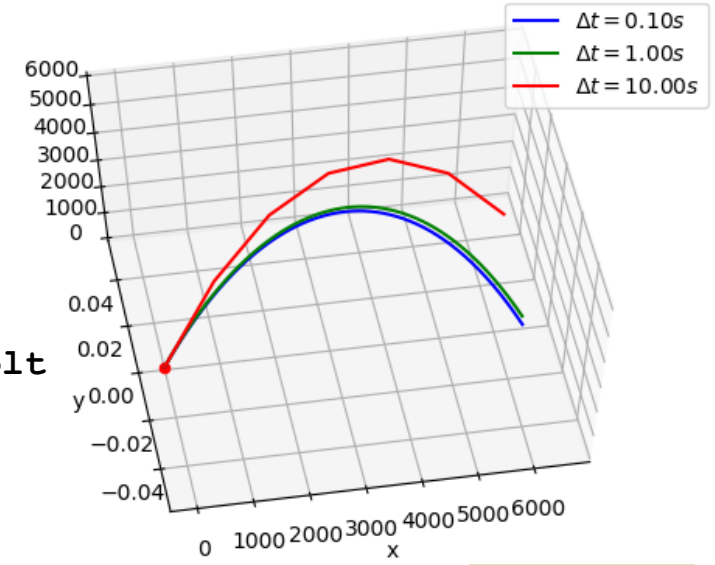
$$\begin{cases} \frac{du}{dt} = 0 \\ \frac{dv}{dt} = 0 \\ \frac{dw}{dt} = -g \\ \frac{dx}{dt} = u \\ \frac{dy}{dt} = v \\ \frac{dz}{dt} = w \end{cases}$$

$$w(t + \Delta t) = w(t) - g\Delta \mathrm{t}$$
$$x(t + \Delta t) = x(t) + u\Delta t$$
$$y(t + \Delta t) = y(t) + v\Delta t$$
$$z(t + \Delta t) = z(t) + w(t)\Delta t$$

Com condições iniciais

$$x(0) = x_0$$
$$y(0) = y_0$$
$$z(0) = z_0$$
$$u(0) = u$$
$$v(0) = v$$
$$w(0) = w_0$$

$$\frac{d\vec{v}}{dt} = \frac{d^2\vec{r}}{dt^2} = \vec{g}$$




```
import numpy as np;import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
plt.close('all');g=9.8065
x0=0;y0=0;z0=0 #posição inicial
u=100;v=0;w0=320 #velocidade inicial
timeInt=2*w0/g #tempo máximo de integração
fig=plt.figure();ax=fig.add_subplot(111, projection='3d')
c=['blue','green','red'];kc=-1;ax.scatter(x0,y0,z0,color='red')
for dt in[0.1,1,10]:
    kc=kc+1;tempo=np.arange(0.,timeInt,dt);n=len(tempo)
    X=np.zeros((n));Y=np.copy(X);Z=np.copy(X)
    X[0]=x0;Y[0]=y0;Z[0]=z0;w=w0
    for kt in range(1,n):
        X[kt]=X[kt-1]+u*dt; Y[kt]=Y[kt-1]+v*dt; Z[kt]=Z[kt-1]+w*dt
        w=w-g*dt
    ax.plot(xs=X,ys=Y,zs=Z,color=c[kc],label=r'$\Delta t=%6.2f s $'%(dt))
plt.legend()
ax.set_zlim(0,6000);ax.set_xlabel('x');ax.set_ylabel('y')
```

Funciona com Δt pequeno

Oscilador harmónico (lei de Hooke)

Oscilador 1D: 2 equações diferenciais de 1ª ordem:

$$\frac{dv}{dt} = \frac{F}{m} = -\frac{k}{m}x$$

$$\frac{dx}{dt} = v$$

A aceleração é variável (mais complicado que o caso $g = const$), depende só da posição.

(Tal como no movimento balístico) o sistema deve conservar energia mecânica:

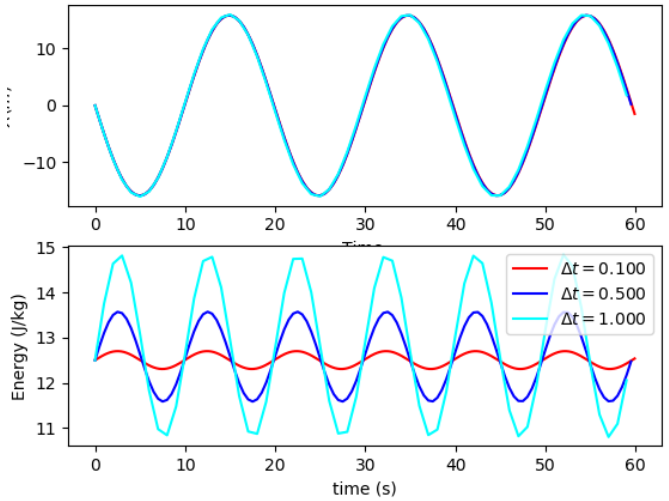
$$\frac{E_M}{m} = \frac{1}{2}\,v^2 + \frac{kx^2}{2} = const$$

Existe solução analítica:

$$x = A\cos(\omega t + \phi)$$

Onde a Amplitude (A) e a fase inicial (ϕ) dependem das condições iniciais, e $\omega = \frac{2\pi}{T} = \sqrt{\frac{k}{m}}$

Oscilador: método de Euler




```
import numpy as np
import matplotlib.pyplot as plt
xinicial=0; vinicial=-5; kapa=0.1 # k/m (lei de Hooke)
cores=['red','blue','cyan','green']; kc=0 #line color
for dt in[0.1,0.5,1.]:
    tempo=np.arange(0.,60.,dt); n=len(tempo)
    X=np.zeros(tempo.shape); V=np.zeros(tempo.shape)
    X[0]=xinicial; V[0]=vinicial
    for k in range(1,n):
        V[k]=V[k-1]-kapa*X[k-1]*dt
        X[k]=X[k-1]+V[k]*dt
    plt.figure(2); plt.subplot(2,1,1)
    plt.plot(tempo,X,color=cores[kc],label=r'$ \Delta t=%6.3f$' %(dt))
    plt.xlabel('Time '); plt.ylabel(r'$X (m)$')
    EM=V**2/2+kapa*X**2/2
    plt.subplot(2,1,2)
    plt.plot(tempo,EM,color=cores[kc],label=r'$ \Delta t=%6.3f$' %(dt))
    plt.ylabel('Energy (J/kg)'); plt.xlabel('time (s)'); plt.legend()
    kc=kc+1
```

Energia varia
2% com $\Delta t = 0.1s$
20% com $\Delta t = 1s$

Comentários

O método de Euler é muito simples. Converge para a solução analítica no $\lim_{\Delta t \to 0}$. Mas é pouco preciso e pode não ser estável. É possível obter métodos melhores com um pequeno custo adicional.

Vamos desenvolver um método mais preciso no próximo exemplo.

De volta ao movimento balístico $(g = const)$

$$\frac{d^2\vec{r}}{dt^2} = \vec{g} = -g\vec{k}$$

Euler:

$$\begin{aligned} w(t+\Delta t) &= w(t) - g\Delta \mathrm{t} \\ x(t+\Delta t) &= x(t) + u\Delta t \\ y(t+\Delta t) &= y(t) + v\Delta t \\ z(t+\Delta t) &= z(t) + w(t)\Delta t \end{aligned}$$

Para satisfazer o teorema da média devia ser:

$$z(t+\Delta t) = z(t) + w\left(t + \frac{\Delta t}{2}\right)\Delta t$$

$$\begin{cases} \frac{du}{dt} = 0 \\ \frac{dv}{dt} = 0 \\ \frac{dw}{dt} = -g \\ \frac{dx}{dt} = u \\ \frac{dy}{dt} = v \\ \frac{dz}{dt} = w \end{cases}$$

Método do ponto médio

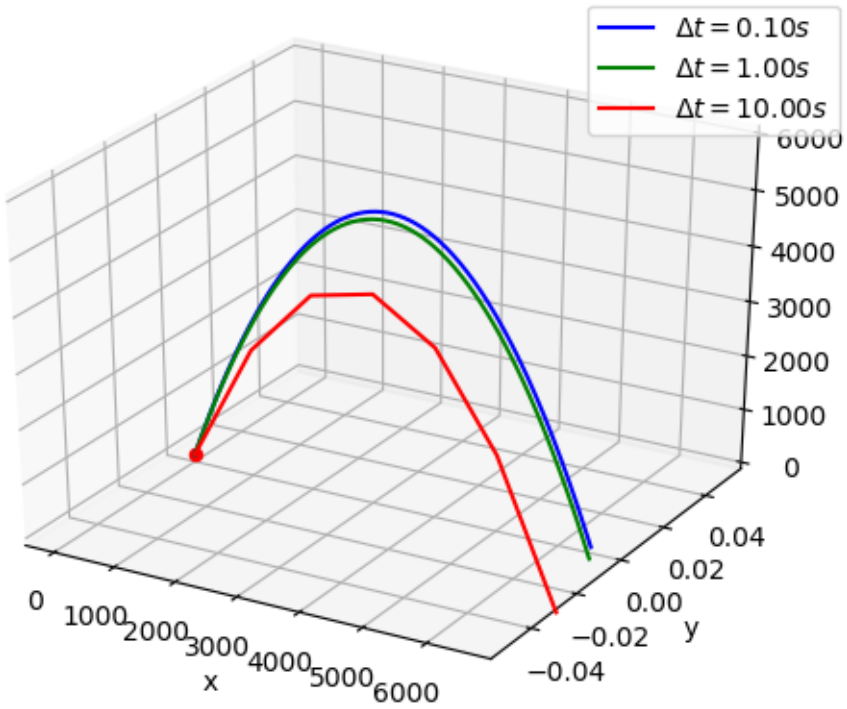
$$z(t+\Delta t) = z(t) + w\left(t + \frac{\Delta t}{2}\right)\Delta t$$

```
import numpy as np;import matplotlib.pyplot as plt
plt.close('all');g=9.8065
x0=0;y0=0;z0=0 #posição inicial
u=100;v=0;w0=320 #velocidade inicial
timeInt=2*w0/g #tempo máximo de integração
fig=plt.figure();ax=fig.add_subplot(111, projection='3d')
c=['blue','green','red'];kc=-1;ax.scatter(x0,y0,z0,color='red')
for dt in[0.1,1,10]:
    kc=kc+1;tempo=np.arange(0.,timeInt,dt);n=len(tempo)
    X=np.zeros((n));Y=np.copy(X);Z=np.copy(X)
    X[0]=x0;Y[0]=y0;Z[0]=z0;w=w0
    for kt in range(1,n):
        wM=w; w=w-g*dt; wH=0.5*(wM+w)
        X[kt]=X[kt-1]+u*dt; Y[kt]=Y[kt-1]+v*dt; Z[kt]=Z[kt-1]+wH*dt
    ax.plot(xs=X,ys=Y,zs=Z,color=c[kc],label=r'$\Delta t=%6.2f s $'%(dt))
plt.legend()
ax.set_zlim(0,6000);ax.set_xlabel('x');ax.set_ylabel('y')
```

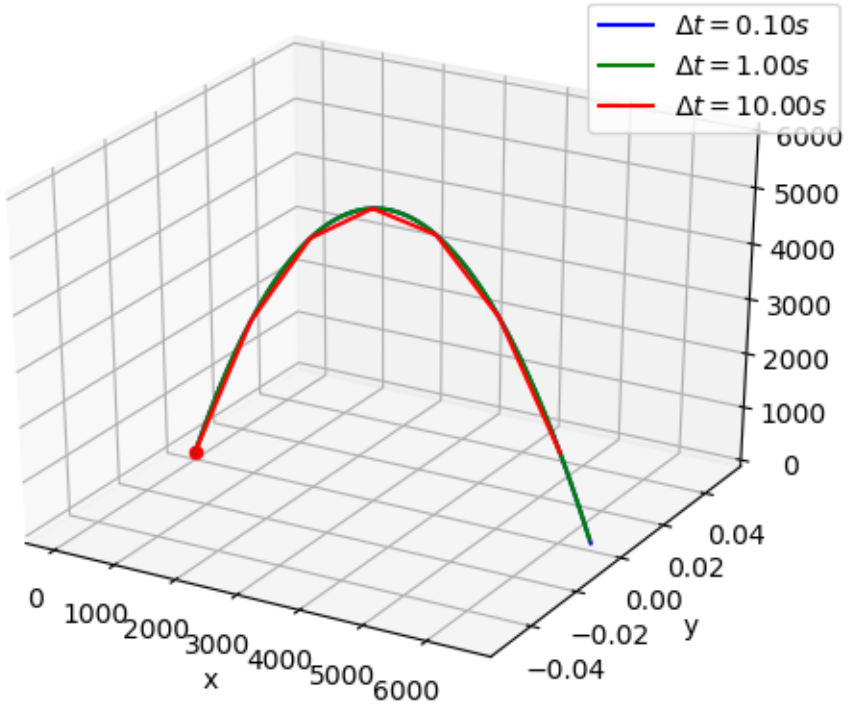
Notar a ordem das instruções

Comparação

$$z(t + \Delta t) = z(t) + w(t)\Delta t$$



$$z(t + \Delta t) = z(t) + w\left(t + \frac{\Delta t}{2}\right)\Delta t$$



O método do ponto médio

Corresponde a uma aproximação de segunda ordem:

$$\left(\frac{dy}{dt}\right)_{t=t_0+\Delta t/2} \approx \frac{y(t_0+\Delta t)-y(t_0)}{\Delta t} + E(\Delta t^2)$$

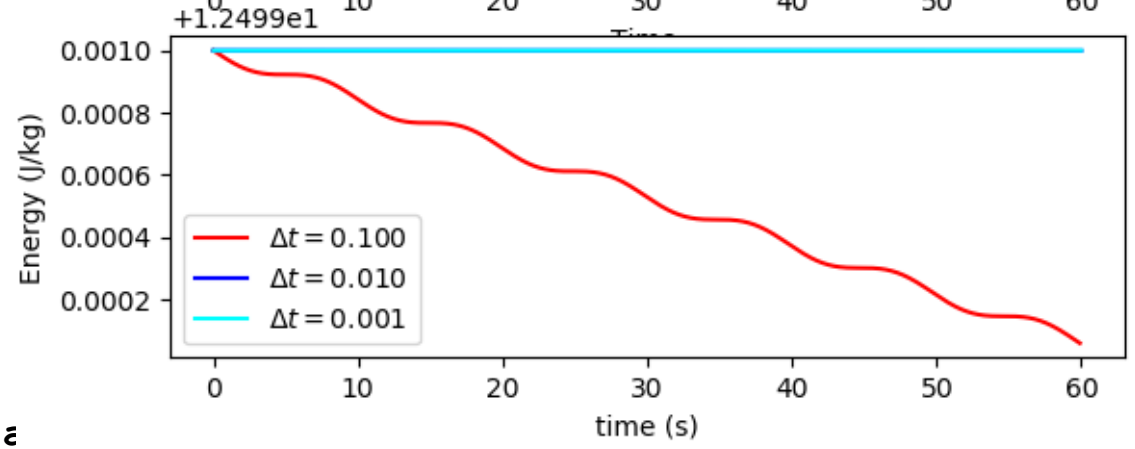
pois trata-se de uma diferença centrada no passo intermédio $t_0 + \Delta t/2$.

O método de Euler usa uma diferença avançada (de 1ª ordem)

$$\left(\frac{dy}{dt}\right)_{t=t_0} \approx \frac{y(t_0+\Delta t)-y(t_0)}{\Delta t} + E(\Delta t)$$

É possível construir aproximações de ordem mais elevada, com a introdução de ainda mais passos intermédios: métodos de Runge-Kutta.

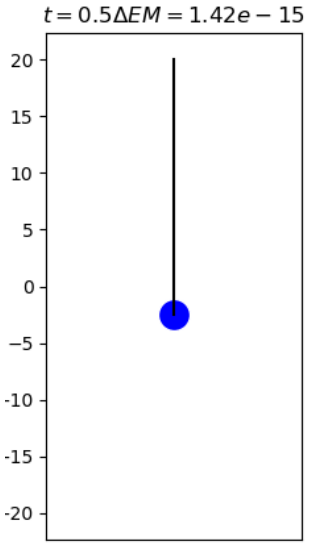
Oscilador com ponto médio




```
import numpy as np
import matplotlib.pyplot a
xinicial=0; vinicial=-5; kapa=0.1 # k/m (lei de Hooke)
cores=['red','blue','cyan','green']; kc=0 #line color
for dt in[0.1,0.5,1.]:
    tempo=np.arange(0.,60.,dt); n=len(tempo)
    X=np.zeros(tempo.shape); V=np.zeros(tempo.shape)
    X[0]=xinicial; V[0]=vinicial
    for k in range(1,n):
        V[k]=V[k-1]-kapa*X[k-1]*dt
        X[k]=X[k-1]+0.5*(V[k]+V[k-1])*dt
        V[k]=V[k-1]-kapa*0.5*(X[k-1]+X[k])*dt
        X[k]=X[k-1]+0.5*(V[k]+V[k-1])*dt
```

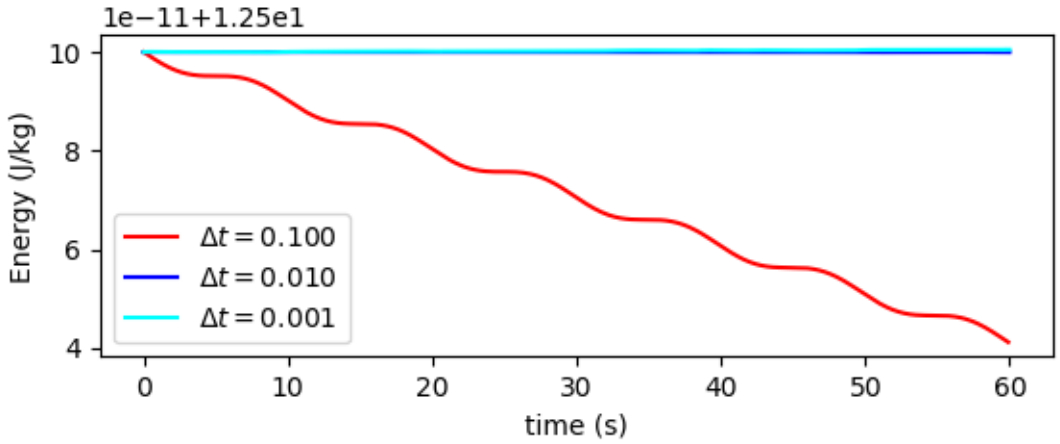
...

Energia varia 10^{-4} com $\Delta t = 0.1s$



Como a aceleração é variável é possível melhorar… até ao erro de arredondamento $\Delta t = 0.1s$

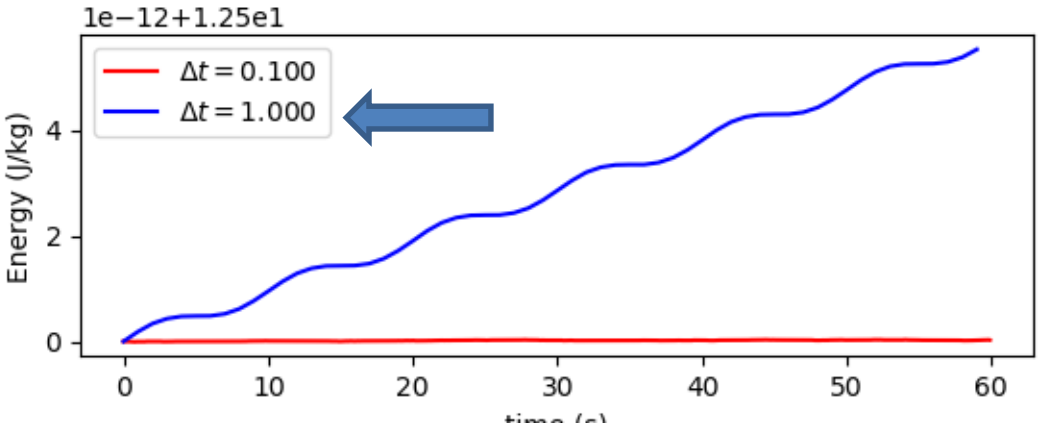
```
for k in range(1,n):
        V[k]=V[k-1]-kapa*X[k-1]*dt
        X[k]=X[k-1]+0.5*(V[k]+V[k-1])*dt
        for improve in range(3):
            V[k]=V[k-1]-kapa*0.5*(X[k-1]+X[k])*dt
            X[k]=X[k-1]+0.5*(V[k]+V[k-1])*dt
```



$$\frac{6 \times 10^{-11}}{12.5} \approx 5 \times 10^{-12}$$

É possível iterar mais… até ao erro de arredondamento agora com $\Delta t = 1s$

```
for k in range(1,n):
        V[k]=V[k-1]-kapa*X[k-1]*dt
        X[k]=X[k-1]+0.5*(V[k]+V[k-1])*dt
        for improve in range(8):
            V[k]=V[k-1]-kapa*0.5*(X[k-1]+X[k])*dt
            X[k]=X[k-1]+0.5*(V[k]+V[k-1])*dt
```



$$\frac{6 \times 10^{-12}}{12.5} \approx 5 \times 10^{-13}$$

Os métodos numéricos tornam-se mesmo interessantes…

Em problemas sem solução analítica.

Movimento planetário $\vec{g}$ variável, $E_M = const$

$$\frac{d^2\vec{r}}{dt^2} = \vec{g} = g_x\vec{e}_x + g_y\vec{e}_y$$

$$\vec{g} = -\frac{GM_{Sol}}{d^2}\vec{u} = -\frac{GM_{Sol}}{x^2+y^2}\frac{x\vec{e}_x + y\vec{e}_y}{\sqrt{x^2+y^2}} = -\frac{GM_{Sol}}{(x^2+y^2)^{3/2}}\left(x\vec{e}_x + y\vec{e}_y\right)$$

d é a distância do centro do "planeta" ao centro do Sol. $\vec{u}$ é o versor (vetor unitário): $\vec{u} = \frac{\vec{r}}{|\vec{r}|} = \frac{\vec{r}}{d}$

Unidades:

$$1\ u.a. \approx 150 \times 10^6 km = 1.5 \times 10^{11} m$$

$$1\ ano \approx 365.25\ dias$$

Vamos admitir que os planetas se movem no plano da eclítica $(\mathrm{x}, \mathrm{y}, z = 0)$.

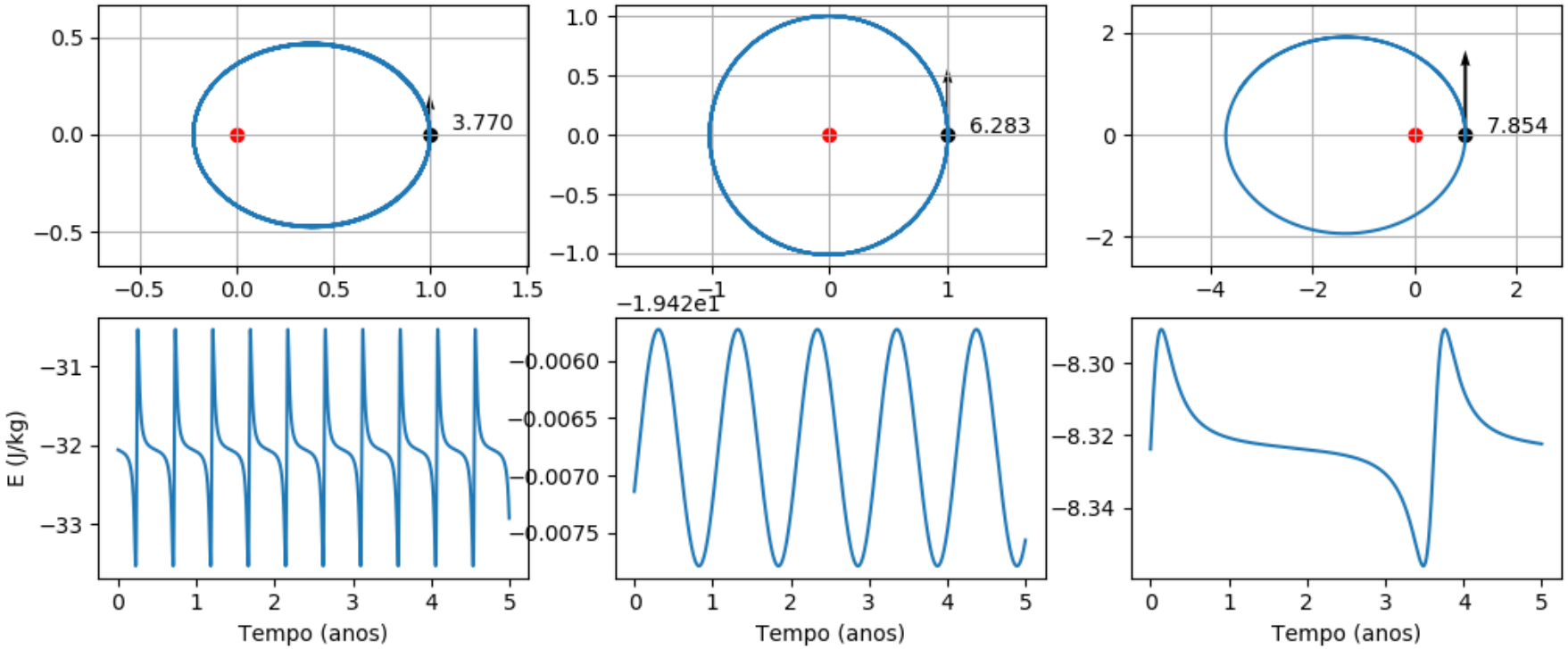
Trajetória planetária (Euler)

```
def traject(x0,y0,vx0,vy0,dt,n):
    G=6.67E-11; M=1.99e30; ua=1.5e11; ano=365.25*24*3600;
    GM=G*M*ano**2*ua**(-3);
    x=x0;y=y0;vx=vx0;vy=vy0;
    t=np.arange(0,n*dt,dt);n=len(t)
    X=np.array(t);Y=np.array(t);E=np.array(t)
    X[0]=x0;Y[0]=y0
    E[0]=(vx**2+vy**2)/2.-GM/np.sqrt(x**2+y**2)
    for k in range(1,n):
        aa=-GM*(x**2+y**2)**(-3./2.);
        ax=aa*x;ay=aa*y
        vx=vx+ax*dt;vy=vy+ay*dt
        x=x+vx*dt;y=y+vy*dt
        X[k]=x;Y[k]=y
        E[k]=(vx**2+vy**2)/2.-GM/np.sqrt(x**2+y**2) #Energia
    return X,Y,t,E
```

Teste do método de Euler

```
import numpy as np;import matplotlib.pyplot as plt
from math import pi
kp=1 #indice de subplot
for vy0 in[1.2*pi,2*pi,2.5*pi]:
    x0=1;y0=0;vx0=0;dt=0.001;n=5/(dt); #5 anos de simulação
    plt.subplot(2,3,kp)
    plt.scatter(x0,y0,color='black') #plot da posição inicial do planeta
    plt.quiver(x0,y0,vx0,vy0,scale=40)
    plt.text(x0+0.01,y0+0.02,'%8.3f ' % (vy0))
    plt.scatter(0.,0.,color='red') #plot da posição do Sol
    plt.grid() #traça grelha
    plt.axis('equal') #escala isotrópica
    [X,Y,T,E]=traject(x0,y0,vx0,vy0,dt,n);
    plt.plot(X,Y)
    plt.subplot(2,3,3+kp)
    plt.plot(T,E) #Energia mecânica
    plt.xlabel('Tempo (anos)')
    if kp==1:
        plt.ylabel('E (J/kg)')
    kp=kp+1
```

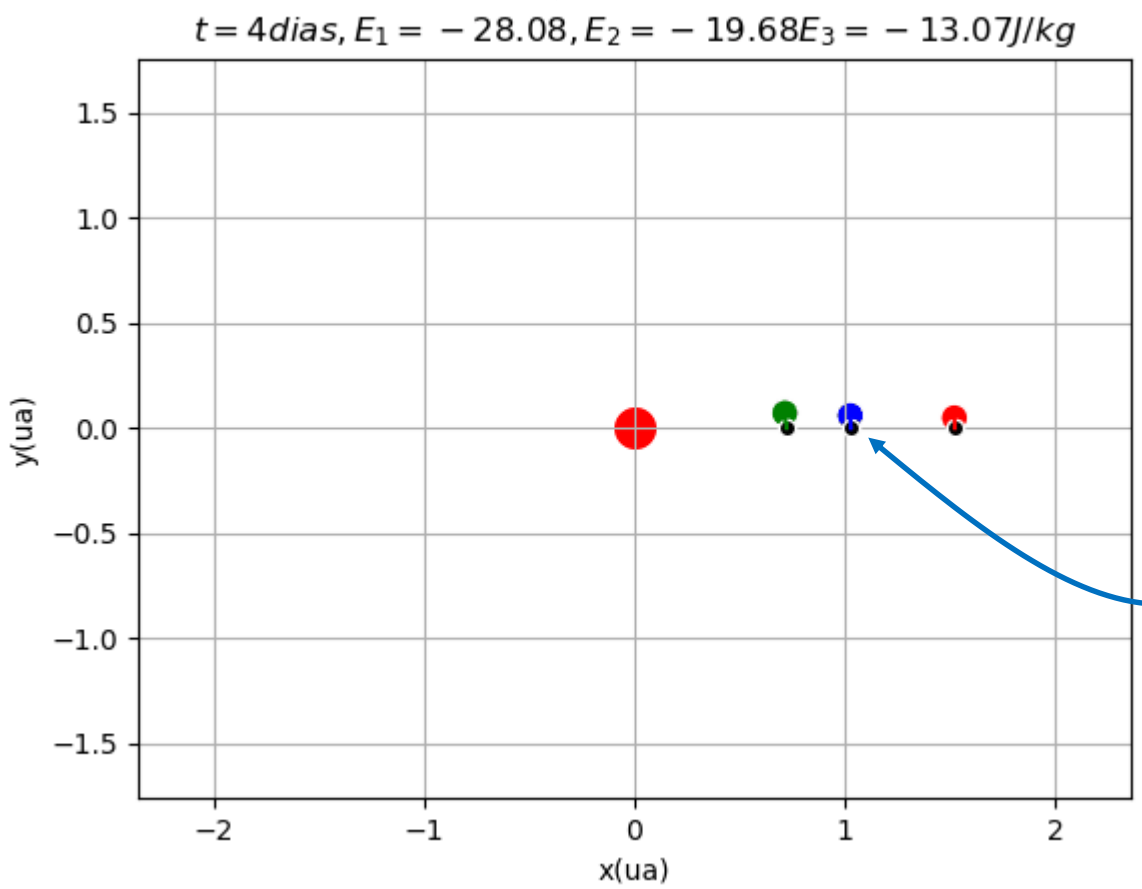
Teste do método de Euler $\Delta t = 0.001$ ano



Energia devia ser constante! O seu ciclo annual é espúrio.

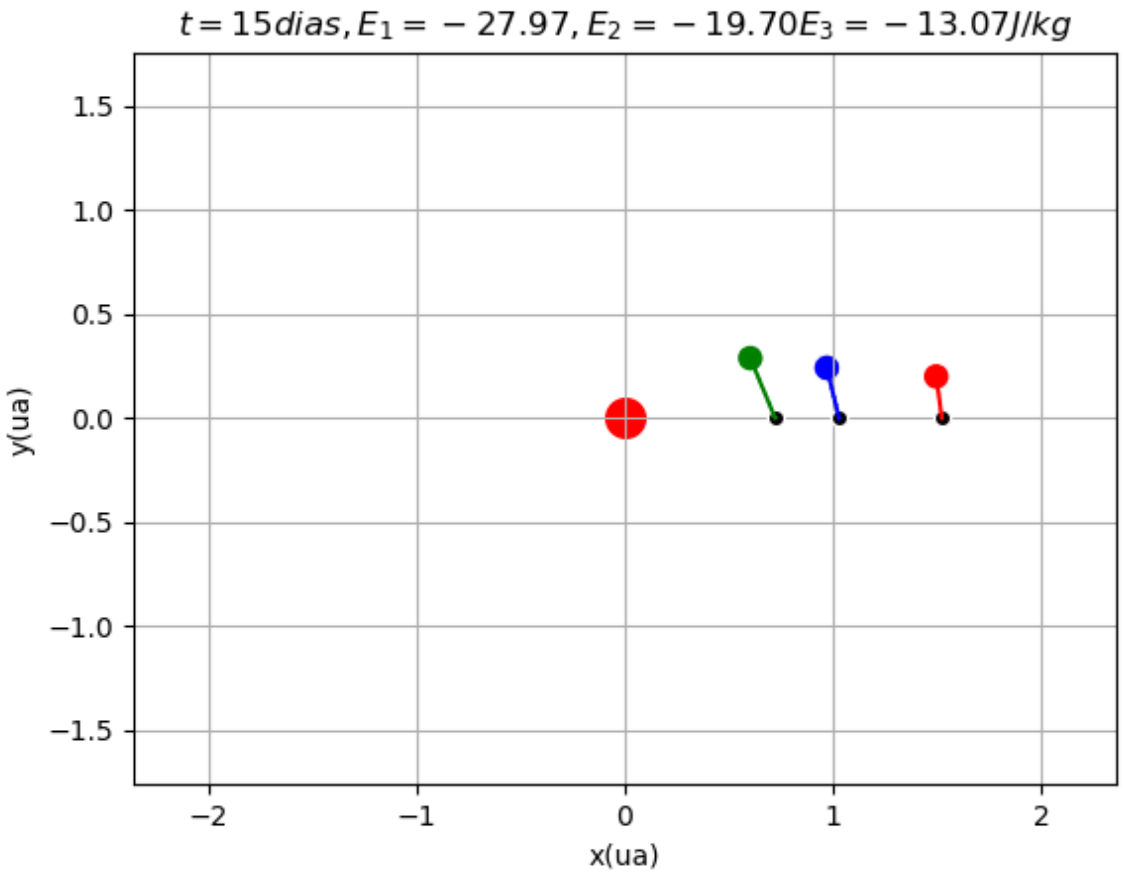
3 planetas $\Delta t = 0.01$ anos

Maior acelaração, Energia varia +



Terra
1ano/ciclo

3 planetas $\Delta t = 0.04$ anos



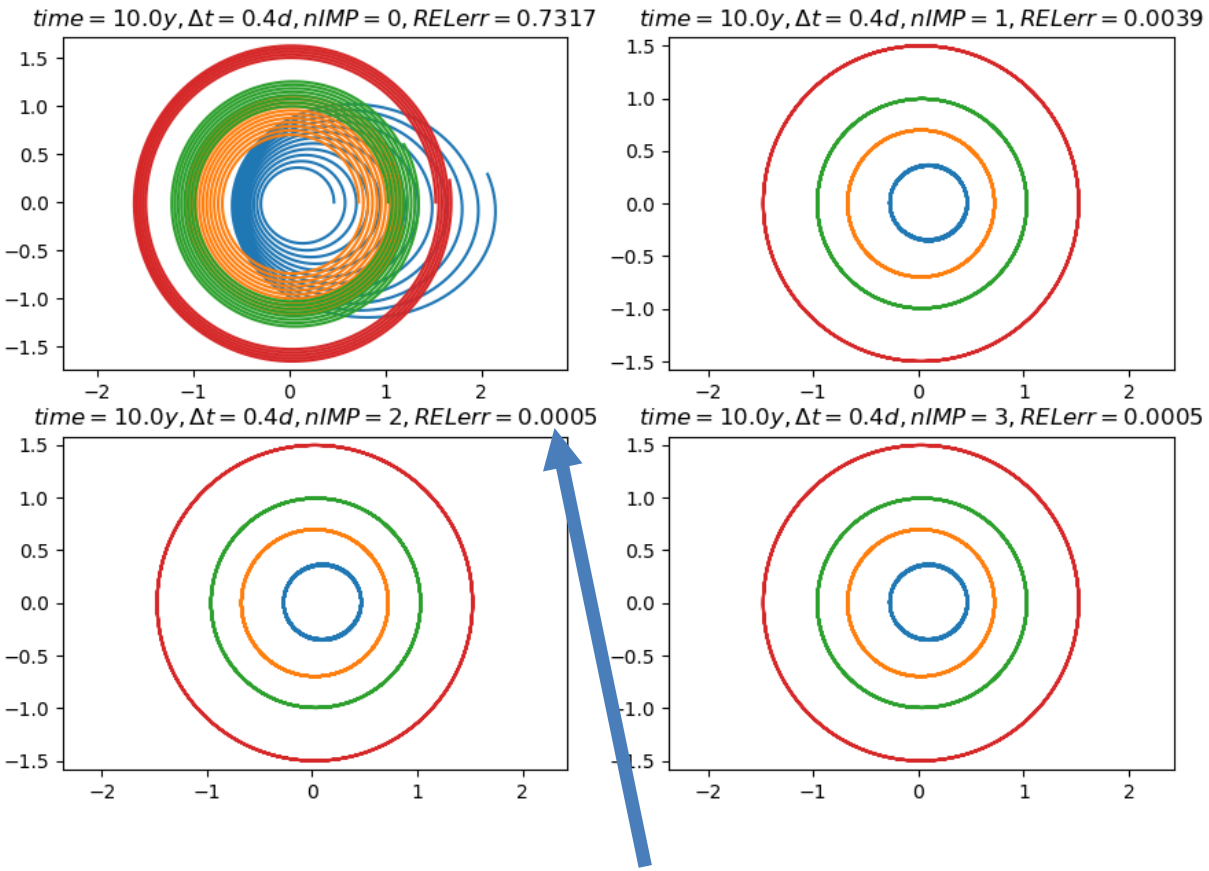
Ponto médio com n planetas

```
def traject(x0,y0,vx0,vy0,dt,n,movie,markYT,nIMP):
    G=6.67E-11;M=1.99e30;ua=1.5e11;ano=365.25*24*3600;GM=G*M*ano**2*ua**(-3);
    nPlanets=len(x0); erro=0; t=np.arange(0,n*dt,dt); n=len(t)
    X=np.zeros((n,nPlanets));Y=np.copy(X);U=np.copy(X);V=np.copy(X);E=np.copy(X)
    X[0]=x0;Y[0]=y0;U[0]=vx0;V[0]=vy0
    E[0]=(U[0]**2+V[0]**2)/2.-GM/np.sqrt(X[0]**2+Y[0]**2)
    for k in range(1,n):
        aa=-GM*(X[k-1]**2+Y[k-1]**2)**(-3./2.); ax=aa*X[k-1]; ay=aa*Y[k-1]
        U[k]=U[k-1]+ax*dt; V[k]=V[k-1]+ay*dt
        X[k]=X[k-1]+0.5*(U[k]+U[k-1])*dt #ponto médio
        Y[k]=Y[k-1]+0.5*(V[k]+V[k-1])*dt
        for improve in range(nIMP):
            x=0.5*(X[k]+X[k-1]); y=0.5*(Y[k]+Y[k-1]); #ponto médio
            aa=-GM*(x**2+y**2)**(-3./2.); ax=aa*x; ay=aa*y
            U[k]=U[k-1]+ax*dt; V[k]=V[k-1]+ay*dt
            X[k]=X[k-1]+0.5*(U[k]+U[k-1])*dt
            Y[k]=Y[k-1]+0.5*(V[k]+V[k-1])*dt
        E[k]=(U[k]**2+V[k]**2)/2.-GM/np.sqrt(X[k]**2+Y[k]**2)
        erro=max(erro,abs((E[k,0]-E[0,0])/E[0,0]))
return X,Y,t,E,erro
```

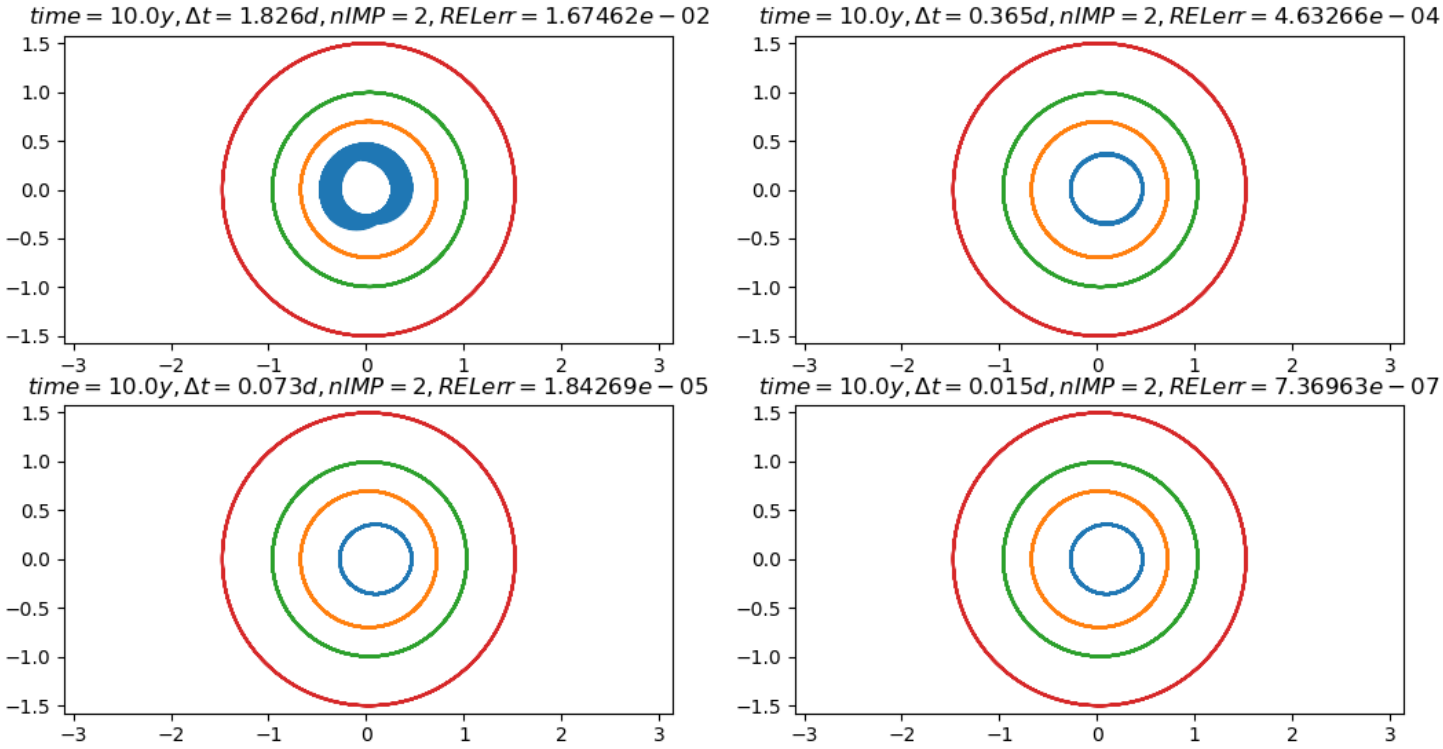
4 planetas terrestres

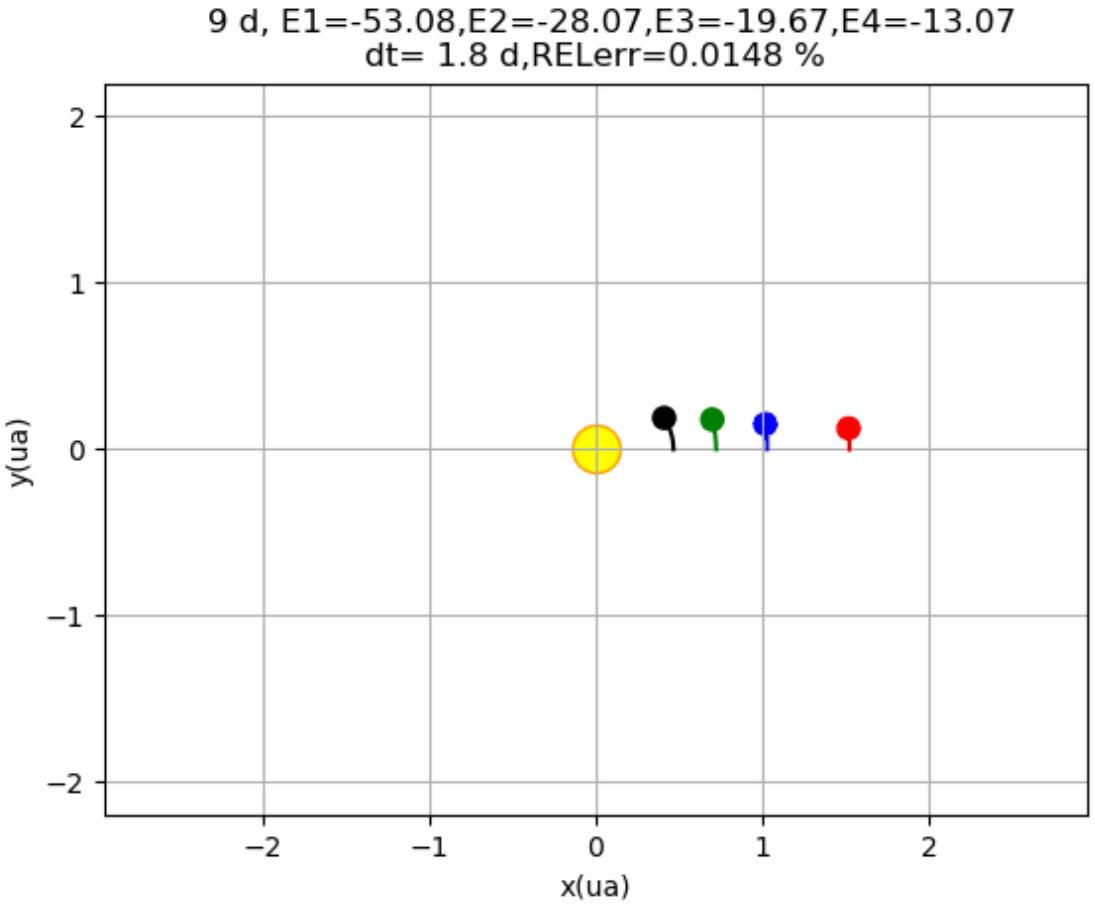
```
import numpy as np; import matplotlib.pyplot as plt; from math import pi
x0=np.array([0.7/1.5,0.723,1.03,1.524]); #Mercurio,Venus,Terra,Marte
vy0=np.array([2.5*pi,2.3*pi,2*pi-0.225,1.6*pi]) #Mercurio,Venus,Terra,Marte
y0=np.zeros(x0.shape);vx0=np.zeros(x0.shape)
dt=0.001;nIMP=2;kp=0
for nIMP in[0,1,2,3]:  #testando a iteração do ponto médio
    plt.figure();kp=kp+1;n=int(10./(dt))+1;
    plt.scatter(0.,0.,color='yellow',edgecolor='orange',s=300) #Sol
    plt.grid();plt.xticks(np.linspace(-2,2,9));
    plt.yticks(np.linspace(-2,2,9))
    [X,Y,T,E,MAXerro]=traject(x0,y0,vx0,vy0,dt,n,movi,markYT,nIMP);
    plt.subplot(2,2,kp);plt.plot(X,Y); plt.axis('equal')
    plt.title(r'$time=%3.1f y,\Delta t=%6.1f d,nIMP=%2i,RELerr=%6.4f $' \
              %(n*dt,dt*365.25,nIMP,MAXerro))
```

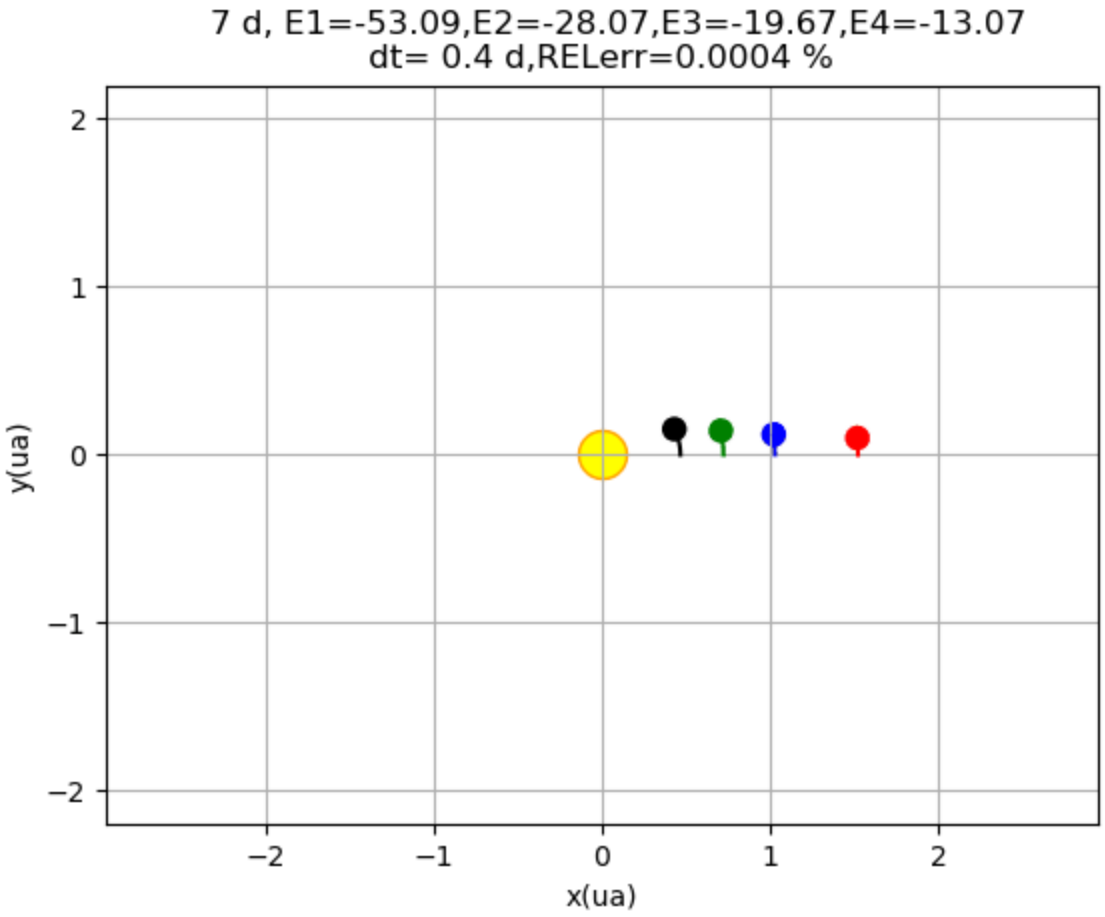
4 planetas terrestres, $\Delta t = 0.4$ dias



4 planetas terrestres, $nIMP = 2$




9 d, E1=-53.08,E2=-28.07,E3=-19.67,E4=-13.07
dt= 1.8 d,RELerr=0.0148 %
y(ua)
x(ua)
2
1
0
−1
−2
−2
−1
0
1
2


7 d, E1=-53.09,E2=-28.07,E3=-19.67,E4=-13.07
dt= 0.4 d,RELerr=0.0004 %
2
1
0
−1
−2
y(ua)
−2
−1
0
1
2
x(ua)