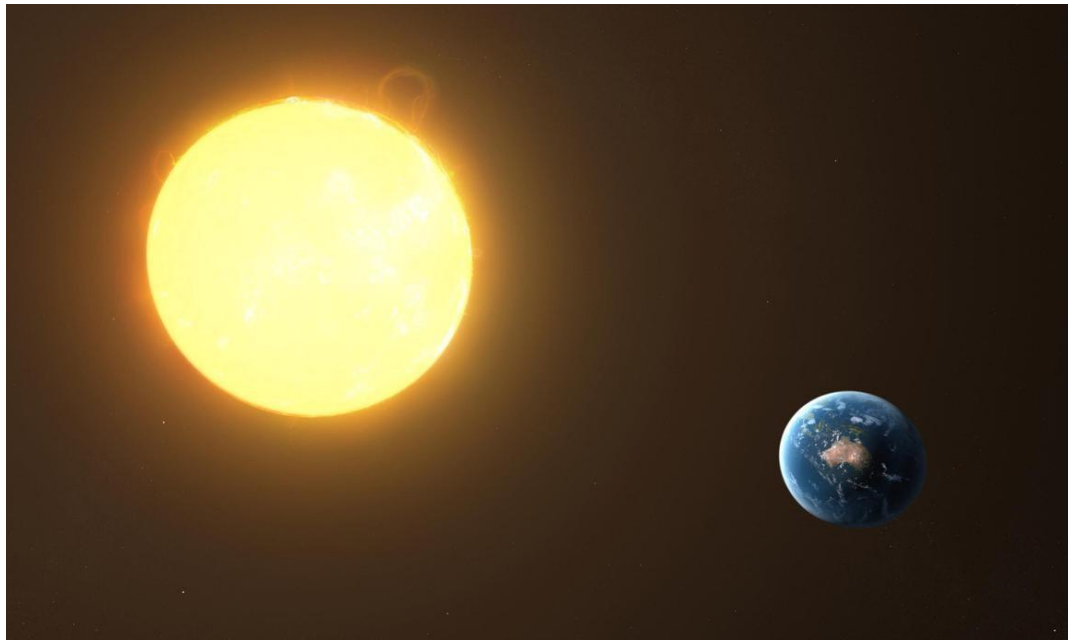
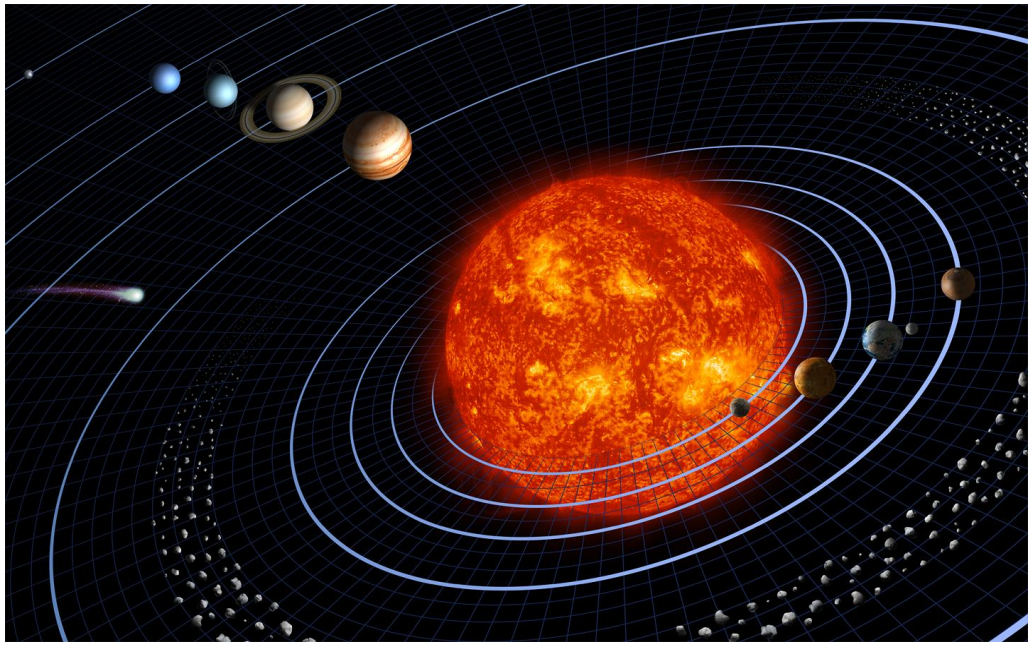
Sistema planetario

IFC – Segundo semestre
Miguel Durán Vera
David Villa Blanco
Universidad de Oviedo
Facultad de ciencias

Índice

Objetivos





Descripción física y matemática

- Leyes de la gravitación de Newton:

$$\vec{F}_{12} = G \cdot \frac{M_1 M_2}{r_{12}^3} \cdot \vec{r}_{12}$$

$$\vec{F}_1 = M_1 \cdot \vec{a}_1$$

$$\vec{a}_1 = G \cdot \frac{M_2}{r_{12}^3} \cdot \vec{r}_{12}$$

- Integrar las ecuaciones diferenciales que describen las trayectorias:

$$\frac{d\vec{v}_1}{dt} = \vec{a}_1 = G \cdot \frac{M_2}{r_{12}^3} \cdot \vec{r}_{12}$$

$$\frac{d\vec{x}_1}{dt} = \vec{v}_1$$

- Ecuaciones diferenciales → trayectoria

Métodos numéricos

Método de Euler

$$y_1 = y_0 + h \cdot f(t_0, y_0)$$

$$\frac{dy}{dt} = f(t_0, y_0)$$

$$y_n = y_{n-1} + h \cdot f(t_{n-1}, y_{n-1})$$

Método de Runge-Kutta (General)

$$y_n = y_{n-1} + h \sum_{i=1}^{s} b_i k_i$$

$$k_i = f(t_n + hc_i, y_n + h \sum_{j=1}^{s} a_{ij} k_j) \quad i = 1, \ldots, s$$

Python

- Astropy: obtenemos datos iniciales con ’get_body_barycentric_posvel’.

- Funciones:
  - r
  - modulo
  - acel
  - diffPlanets
  - matrizafila
  - filaamatriz
  - matrizacubo
  - funder

```
def diffPlanets(p):
        w=filaamatriz(p)
        diff=zeros_like(w)
        for i in range(len(w)):
                x1,y1,z1,vx1,vy1,vz1=w[i]
                dx_dt=vx1
                dy_dt=vy1
                dz_dt=vz1
                dvx_dt=0
                dvy_dt=0
                dvz_dt=0
                for j in range(len(w)):
                        if i==j: continue
                        x2,y2,z2,vx2,vy2,vz2=w[j]
                        R=r(x1,x2,y1,y2,z1,z2)
                        m=M[j]
                        dvx_dt+=acel(R,m)[0] #vamos sumando cada aceleración obtenida
                        dvy_dt+=acel(R,m)[1]
                        dvz_dt+=acel(R,m)[2]
                diff[i]=array([dx_dt,dy_dt,dz_dt,dvx_dt,dvy_dt,dvz_dt])

        integrando=matrizafila(diff)
        return integrando
```

Python

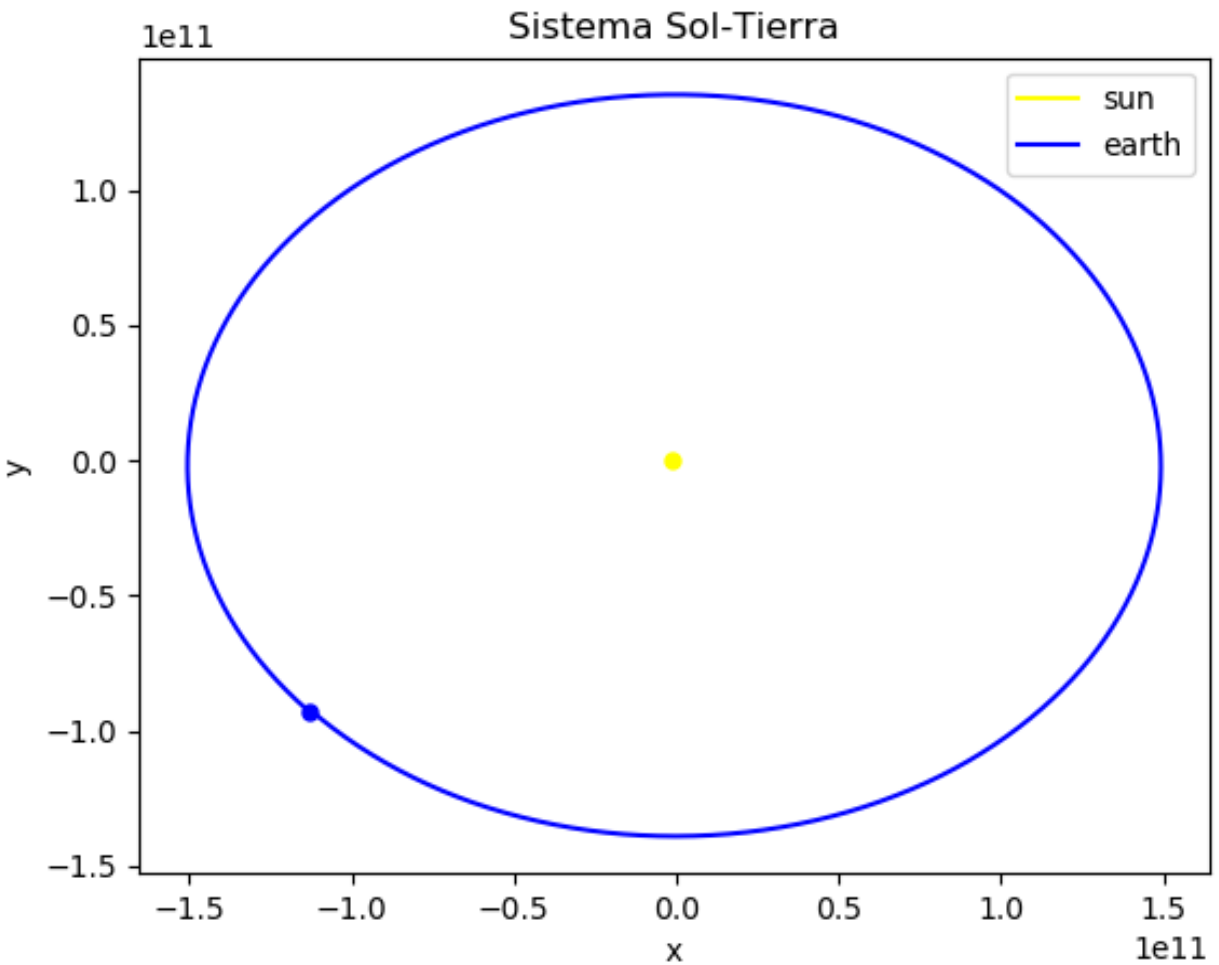
- solve_ivp: permite obtener trayectoria a partir de diferenciales y valores iniciales.

```
result=solve_ivp(funder,[tiempo[0],tiempo[-1]],datosTolinea,t_eval=tiempo,method='RK23').y


trayectorias=matrizacubo(result)
```

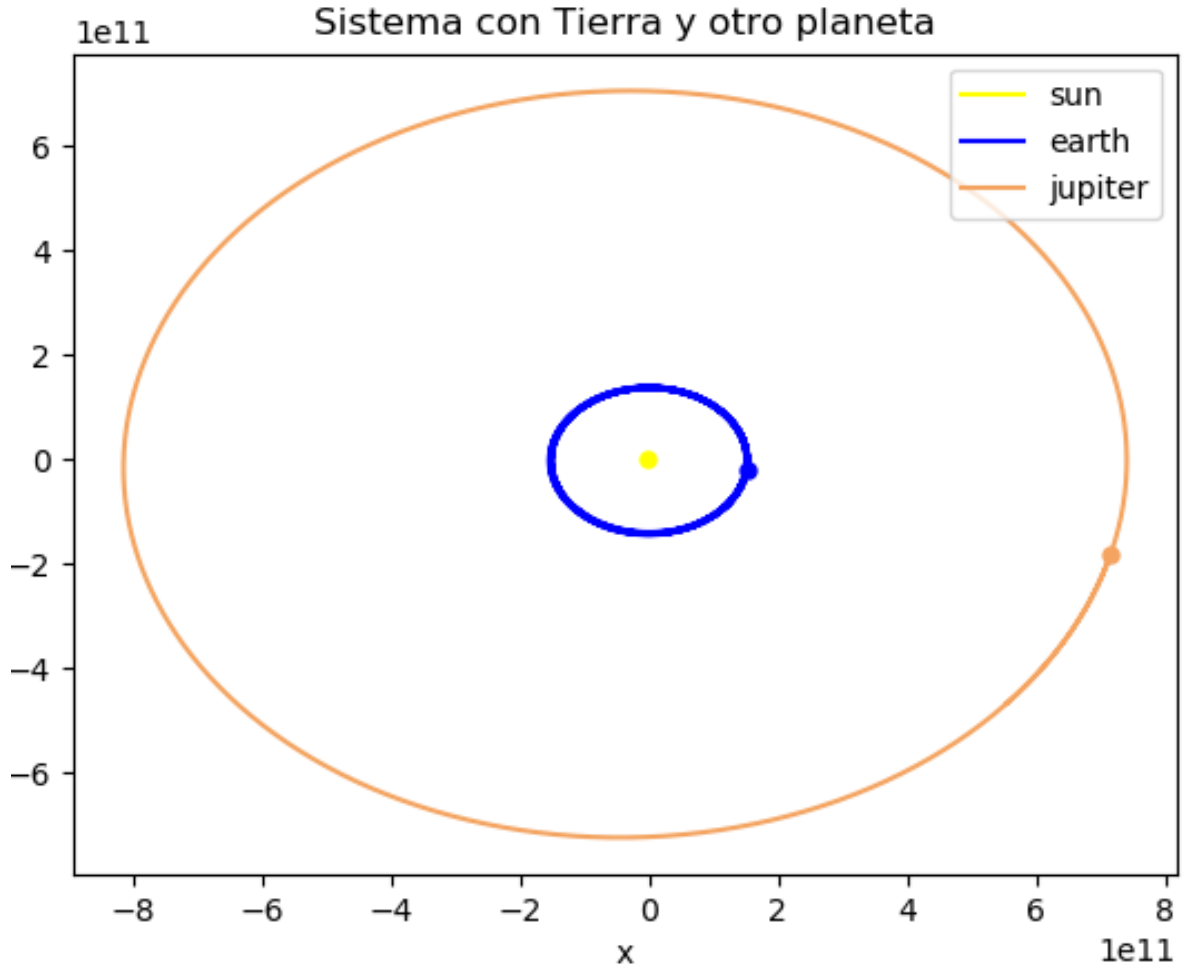
- Creación de la representación: 2D, 3D y vpython.

```
for i in range(len(trayectorias)):
    cuerpo=trayectorias[i]
    x=cuerpo[0,:int(seg[i]/nin)]
    y=cuerpo[1,:int(seg[i]/nin)]
    if i>4: mark=10
    else: mark=5
    plt.plot(x,y,colors[i],label=planets[i]) #creamos una representación de todas las trayectorias
    plt.plot([x[-1]],[y[-1]],markersize=mark,color=colors[i],marker='o')  #circulo que representa la posicion final de cada planeta

plt.legend(loc=1)
plt.title('Sistema con Tierra y otro planeta')
plt.xlabel('x')
plt.ylabel('y')
plt.savefig("Tierra y Júpiter.png")
plt.show()
```

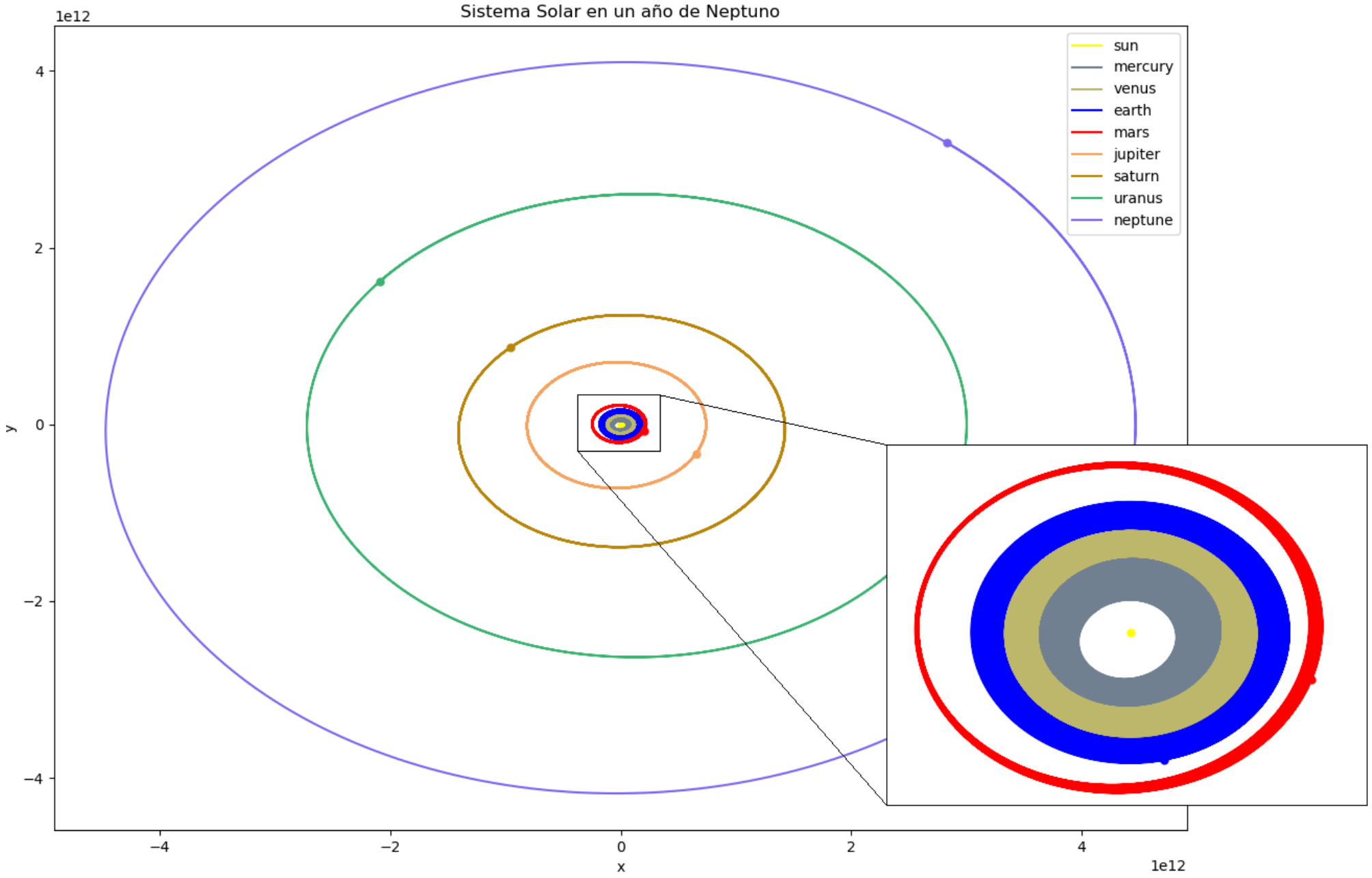
Sistema Tierra-Sol

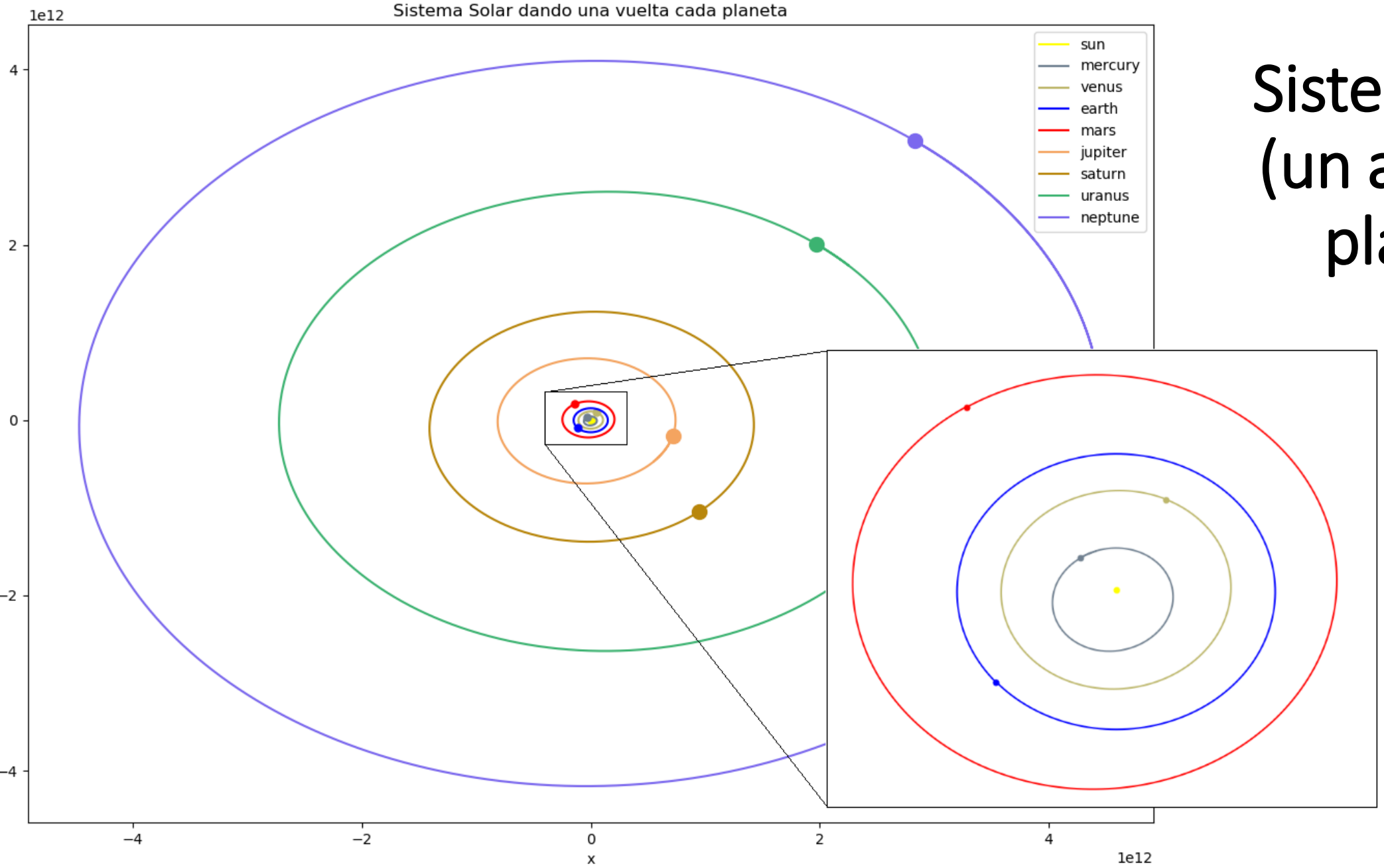


Sistema 2 planetas



Sistema Solar

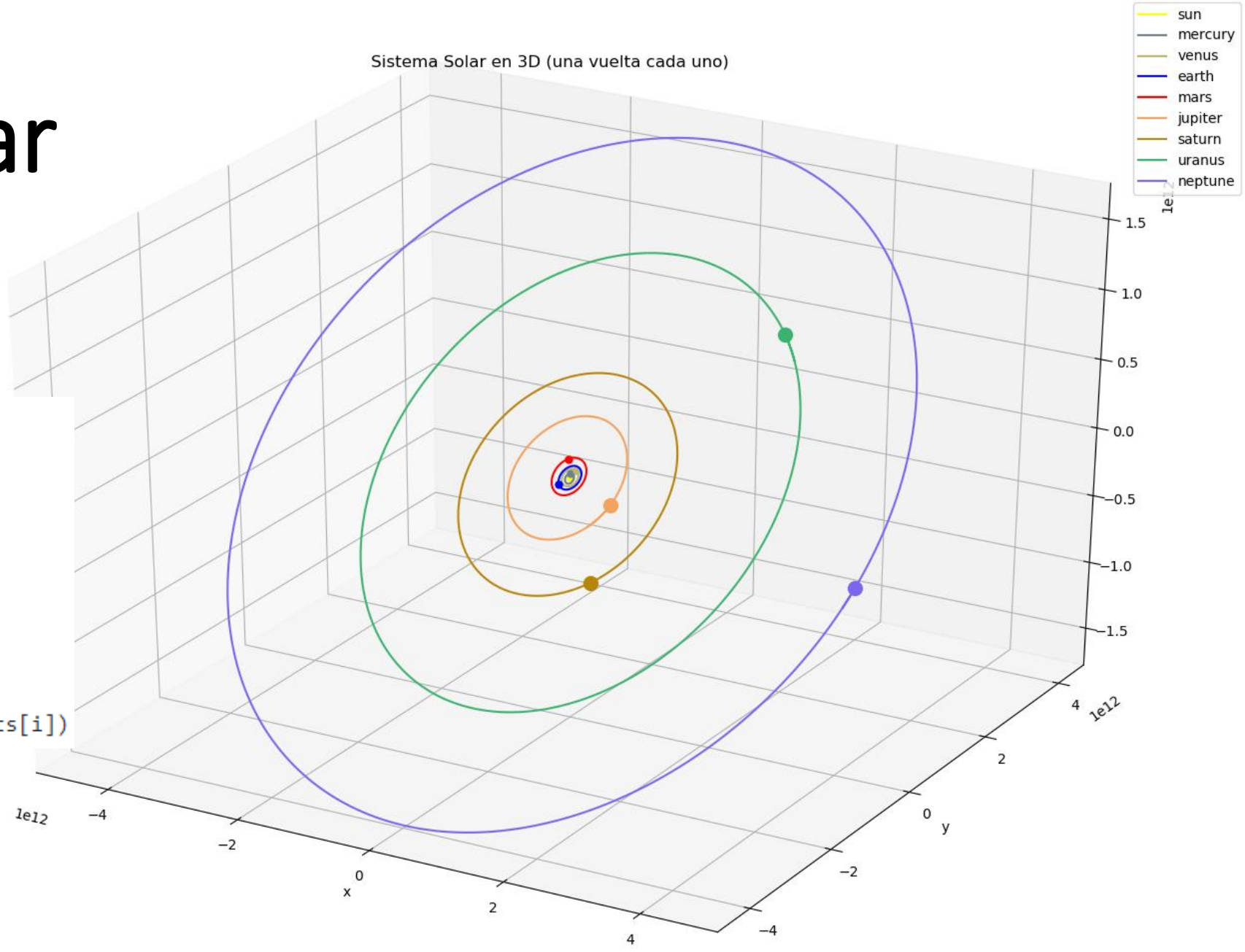


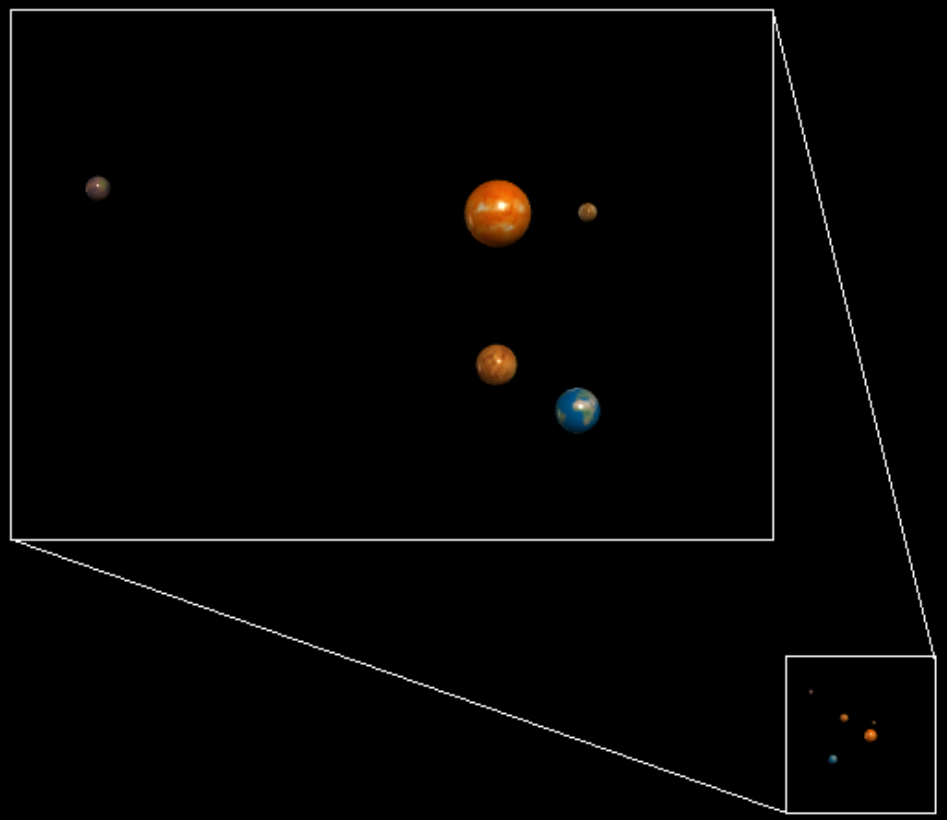


Sistema Solar (un año cada planeta)

Sistema Solar en 3D

```
planet3D=plt.figure(figsize=(14,10))
ax=Axes3D(planet3D)

for i in range(len(trayectorias)):
        cuerpo=trayectorias[i]
        x=cuerpo[0]
        y=cuerpo[1]
        z=cuerpo[2]
        ax.plot(x,y,z,colors[i],label=planets[i])
```



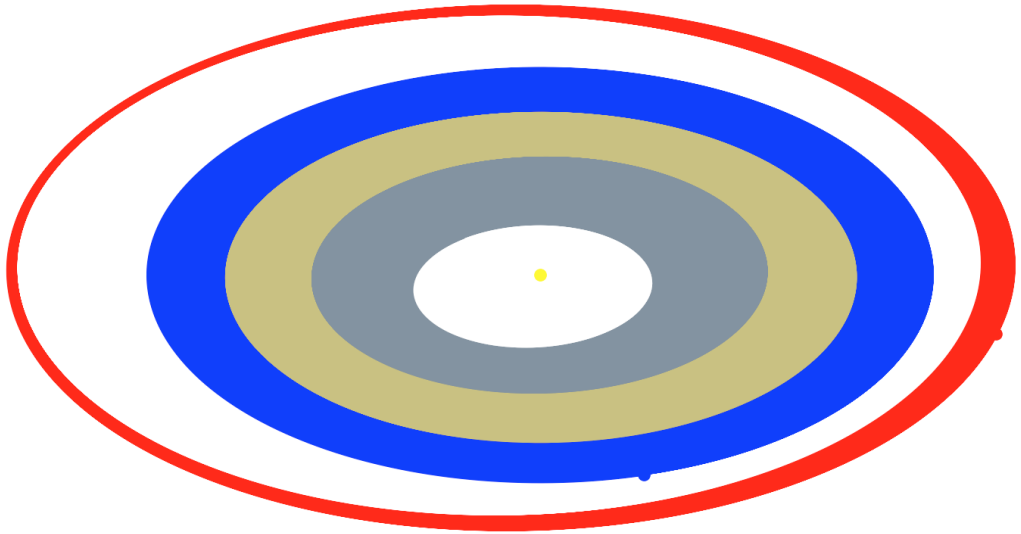


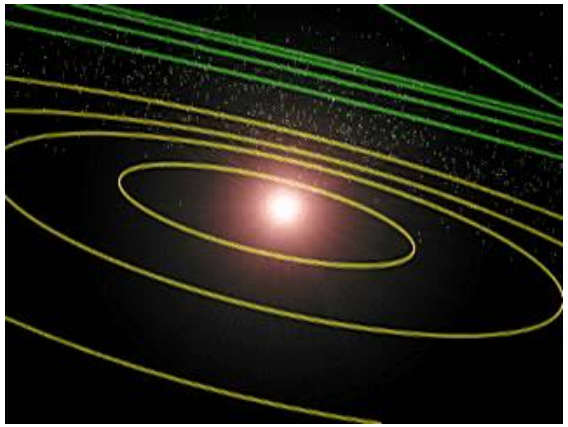
Simulación en Vpython

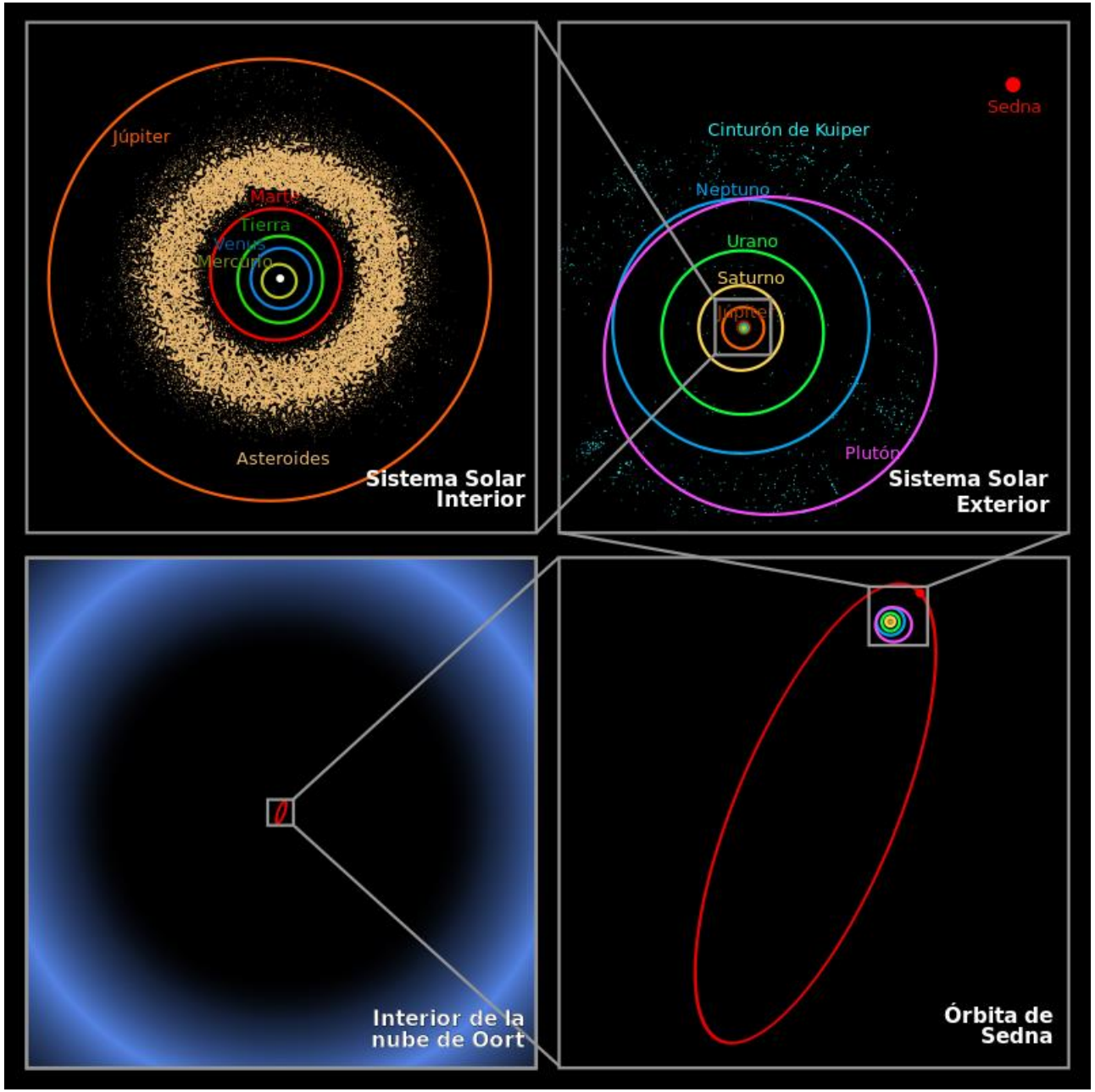
Simulación en Vpython

```
for j in range(len(trayectorias)):
                cuerpo = trayectorias[j]
                x=cuerpo[0]
                y=cuerpo[1]
                z=cuerpo[2]
                if j==6:
                        circle1=shapes.circle(radius=18)
                        circle2=shapes.circle(radius=12)#creamos discos para Saturno y para Urano
                        disc=extrusion(pos=vector(x[j]/AU*10,y[j]/AU*10,z[j]/AU*10),axis=vector(1,0,0),path=[vector(0,0.01,0),vector(0,-0.01,0)],shape=[circle1,circle2],texture={'fil
                if j==7:
                        circle3=shapes.circle(radius=8)
                        circle4=shapes.circle(radius=7)
                        disc2=extrusion(pos=vector(x[j]/AU*10,y[j]/AU*10,z[j]/AU*10),axis=vector(1,0,0),path=[vector(0.03,0.01,0),vector(-0.03,-0.01,0)],shape=[circle3,circle4],textu
                bola=sphere(pos=vector(x[0]/AU*10,y[0]/AU*10,z[0]/AU*10),radius=tam[j]/5,texture={'file':text[j],'place':'all'}) #creamos una esfera para cada cuerpo
                planetas.append(bola)#lista donde se guardan todas las esferas


for i in range(len(trayectorias[0][0])): #va por todos los tiempo que solve_ivp ha analizado y actualizando todas las posiciones en cada caso.
        for j in range(len(trayectorias)):
                rate(100)
                cuerpo = trayectorias[j]
                x=cuerpo[0]
                y=cuerpo[1]
                z=cuerpo[2]
                bola = planetas[j] #escoge la esfera correspondiente
                bola.pos = vector(x[i]/AU*10,y[i]/AU*10,z[i]/AU*10)
                if j==6:
                        disc.pos=bola.pos
                if j==7:
                        disc2.pos=bola.pos
```

Conclusiones







Referencias

- scipy.integrate.solve ivp. Scipy.org.
- Método de Euler. Wikipedia.com.
- Método de Runge-Kutta. Wikipedia.com.
- L.F. Shampine P. Bogacki. “A 3(2) pair of Runge - Kutta formulas”. En: (1989).
- Julio M. Fernandez Díaz y Rosario Díaz Crespo. “Introducción a la Física Computacional: Problemas de Física resueltos numéricamente”.

Enlace a Github con el código: GitHub - DavidVB10/Solar_system: