Metody Obliczeniowe Fizyki i Techniki
Laboratorium 12
Rozwiązywanie równań dynamiki Newtona z automatyczną kontrolą błędu i doborem kroku czasowego

Krzysztof Tondera III rok FT

29.03.2021

1 Przedstawienie Problemu

Celem laboratorium jest wyliczenie orbity ciała o parametrach ruchu zbliżonych do komety Halleya. W chwili początkowej kometa znajdujęe się w peryhelium orbity (0,0.586 au) i porusza się z prędkością (54600 $\frac{m}{s}$, 0). Słońcu przypisano masę M=1.989$\cdot10^{30}$kg i nieruchamiamy je w początku układu odniesienia. Stała grawitacji G=6.6741$\cdot10^{-11}$ $\frac{m^3}{kg\ s^2}$. jednostka astronomiczna au=149 597 870 700 m. Ruch ciała w potencjale słońca opisują równania:

$$\frac{dx}{dt} = v_x \tag{1}$$

$$\frac{dy}{dt} = v_y \tag{2}$$

$$\frac{dv_x}{dt} = -G\frac{M}{r^3}x = a_x \tag{3}$$

$$\frac{dv_y}{dt} = -G\frac{M}{r^3}y = a_y \tag{4}$$

[2]:
```
def ax(x,y):
    r=np.sqrt(x**2+y**2)
    return -G*m_s*x/r**3

def ay(x,y):
    r=np.sqrt(x**2+y**2)
    return -G*m_s*y/r**3

def position(r,v,dt):
    return r+v*dt

def velocity(v,a,dt):
    return v+a*dt
```

2 Jawny Schemat Eulera

Wykorzystując schemat Eulera obliczono tor komety przy 3 obrotach dookoła słońca (t=365*24*3600*75*3s, jako okres pełnego obiegu przyjęto 75 lat). Zrealizowano schemat za pomocą poniższej funkcji:

[3]:
```
def fun_euler(x0,y0,vx0,vy0,dt):
    tab_x=[x0]
    tab_y=[y0]
    tab_t=[0]


    vx=vx0
    vy=vy0
    for i in range(int(365*24*3600*75*3/dt)):
        x = tab_x[-1]
        y = tab_y[-1]
        v_xn = velocity(vx, ax(x,y), dt)
        v_yn = velocity(vy, ay(x,y), dt)
        x_n = position(x, vx, dt)
        y_n = position(y, vy, dt)

        vy = v_yn
        vx = v_xn
        tab_x.append(x_n)
        tab_y.append(y_n)
        tab_t.append(tab_t[i]+dt)

    return tab_x,tab_y,tab_t
```

Dla kroku czasowego Δt=3600s zrealizowano poniższe wykresy:

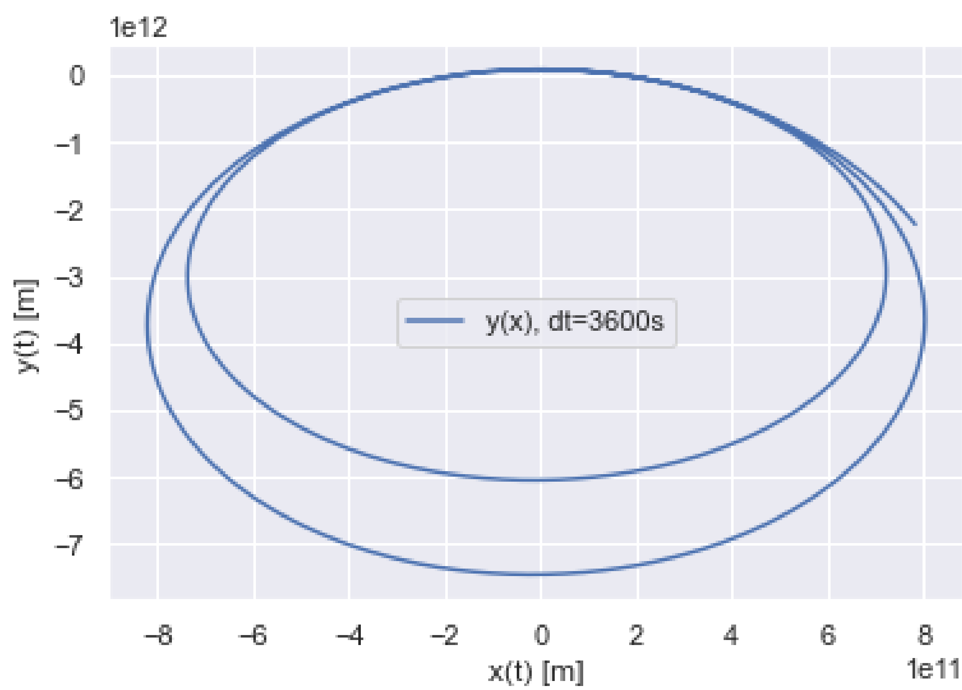

Figure 1: Portret fazowy dla schematu Eulera

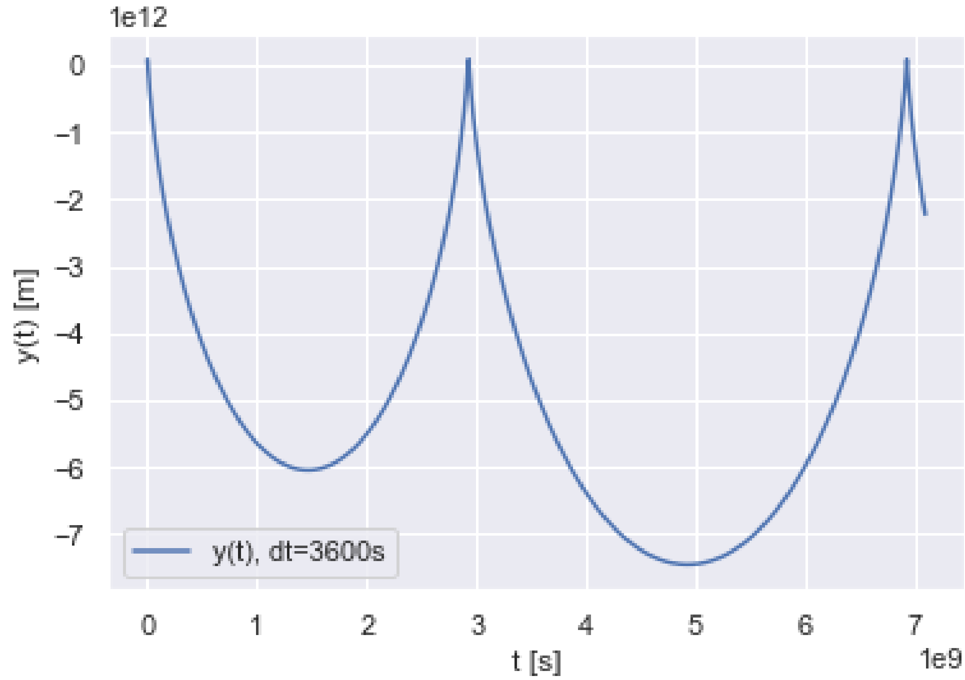

Figure 2: Zależność y(t) w schemacie Eulera

Policzono że czas trwania obliczeń programu to 26.32s.

3 Metoda RK4

Zaimplementowano metodę RK4 zgodnie ze schematem z wykładu:

[5]:
```
def fun_RK4(x0,y0,vx,vy,dt):

    k_11=vx
    k_12=vy
    k_13=ax(x0,y0)
    k_14=ay(x0,y0)

    k_21=vx+dt*k_13/2
    k_22=vy+dt*k_14/2
    k_23=ax(x0+dt*k_11/2,y0+dt*k_12/2)
    k_24=ay(x0+dt*k_11/2,y0+dt*k_12/2)

    k_31=vx+dt*k_23/2
    k_32=vy+dt*k_24/2
    k_33=ax(x0+dt*k_21/2,y0+dt*k_22/2)
    k_34=ay(x0+dt*k_21/2,y0+dt*k_22/2)

    k_41=vx+dt*k_33
    k_42=vy+dt*k_34
    k_43=ax(x0+dt*k_31,y0+dt*k_32)
    k_44=ay(x0+dt*k_31,y0+dt*k_32)

    x_t=x0+dt*(k_11+2*k_21+2*k_31+k_41)/6
```

```
    y_t=y0+dt*(k_12+2*k_22+2*k_32+k_42)/6
    vx_t=vx+dt*(k_13+2*k_23+2*k_33+k_43)/6
    vy_t=vy+dt*(k_14+2*k_24+2*k_34+k_44)/6

    return x_t,y_t,vx_t,vy_t


def calc_RK4(x0,y0,vx0,vy0,dt):
    tab_x=[x0]
    tab_y=[y0]
    tab_t=[0]



    vx=vx0
    vy=vy0
    for i in range(int(365*24*3600*75*3/dt)):
        x = tab_x[-1]
        y = tab_y[-1]

        x_n,y_n,vx,vy=fun_RK4(x,y,vx,vy,dt)

        tab_x.append(x_n)
        tab_y.append(y_n)
        tab_t.append(tab_t[i]+dt)

    return tab_x,tab_y,tab_t
```

Za pomocą poniższych funkcji wygenerowano wykresy podobne jak w poprzednim punkcie:

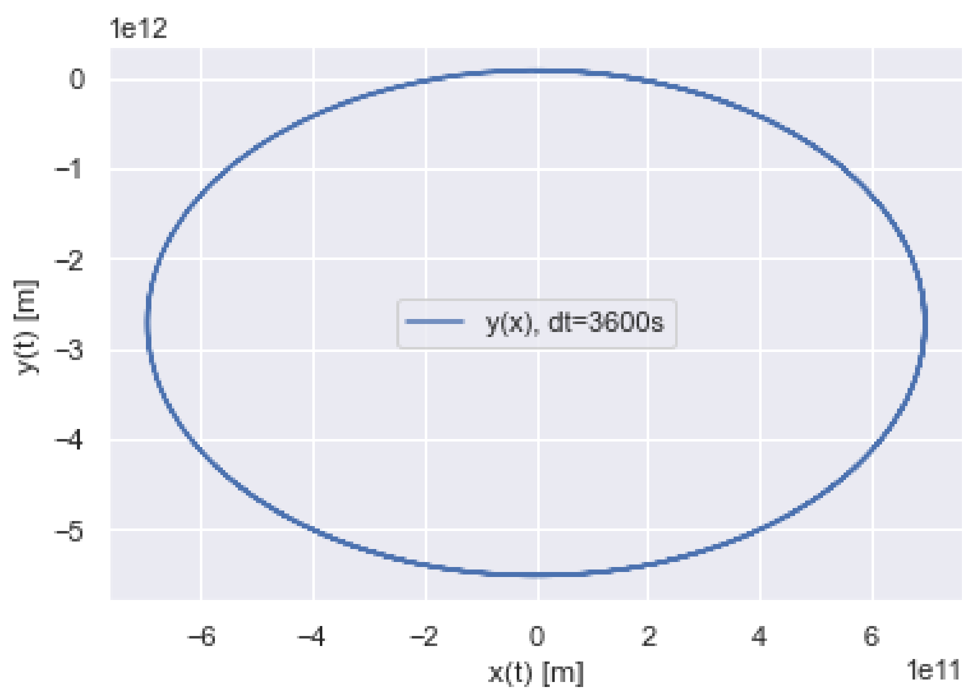

Figure 3: Portret fazowy dla metody RK4

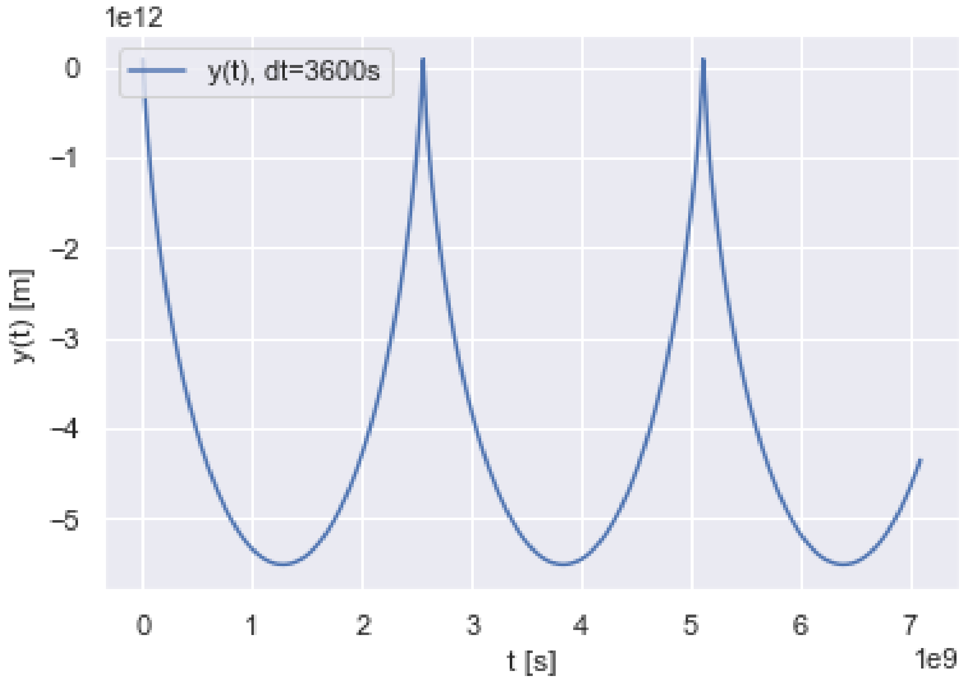

Figure 4: Zależność y(t) w metodznie RK4

Czas wykonywania obliczeń: 100.6 s. Widać od razu że obliczenia schematem Eulera przebiegają szybciej niż w metodzie RK4.

4 Metoda Eulera z kontrolą błędu i automatycznym doborem kroku czasowego

W tym punkcie monitorowano błędy dla x i y i jako błąd ε przyjmowano ten który jest większy. Następnie wygenerowano wykresy jak w poprzednich zadaniach dla tolerancji błędu 1000m oraz 1m. Zaimplementowano schemat za pomocą funkcji:

[36]:
```
def fun_euler_at(tol,x0,y0,vx0,vy0,dt0):
    tab_x=[x0]
    tab_y=[y0]
    tab_t=[0]
    tab_dt=[dt0]

    t=0
    c=0.9
    n=1 #w metodzie Eulera
    vx=vx0
    vy=vy0
    dt=dt0

    while(365*24*3600*75*3>t):
        x = tab_x[-1]
        y = tab_y[-1]

        x1=position(x,vx,dt)
        y1=position(y,vy,dt)
```

```
        x2=position(x,vx,dt/2)
        y2=position(y,vy,dt/2)

        vx2=velocity(vx,ax(x,y),dt/2)
        vy2=velocity(vy,ay(x,y),dt/2)

        x3=position(x2,vx2,dt/2)
        y3=position(y2,vy2,dt/2)

        eps_x=(x3-x1)/(2**n-1)
        eps_y=(y3-y1)/(2**n-1)

        if abs(eps_x)>abs(eps_y):
            eps=eps_x
        else:
            eps=eps_y

        if eps<=tol:
            tab_x.append(x1)
            tab_y.append(y1)
            tab_t.append(tab_t[-1]+dt)
            tab_dt.append(dt)
            vx=velocity(vx,ax(x,y),dt)
            vy=velocity(vy,ay(x,y),dt)
            t+=dt

        dt=c*dt*abs(tol/eps)**(1/(n+1))
    return tab_x,tab_y,tab_t,tab_dt
```

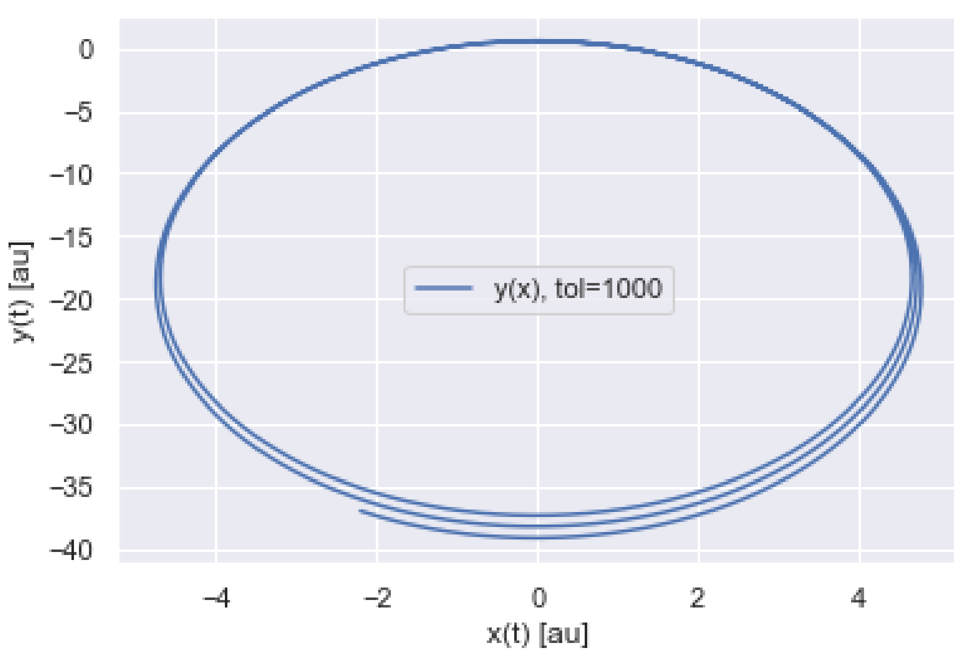

Figure 5: Portret fazowy dla schematu Eulera, tolerancja=1000m

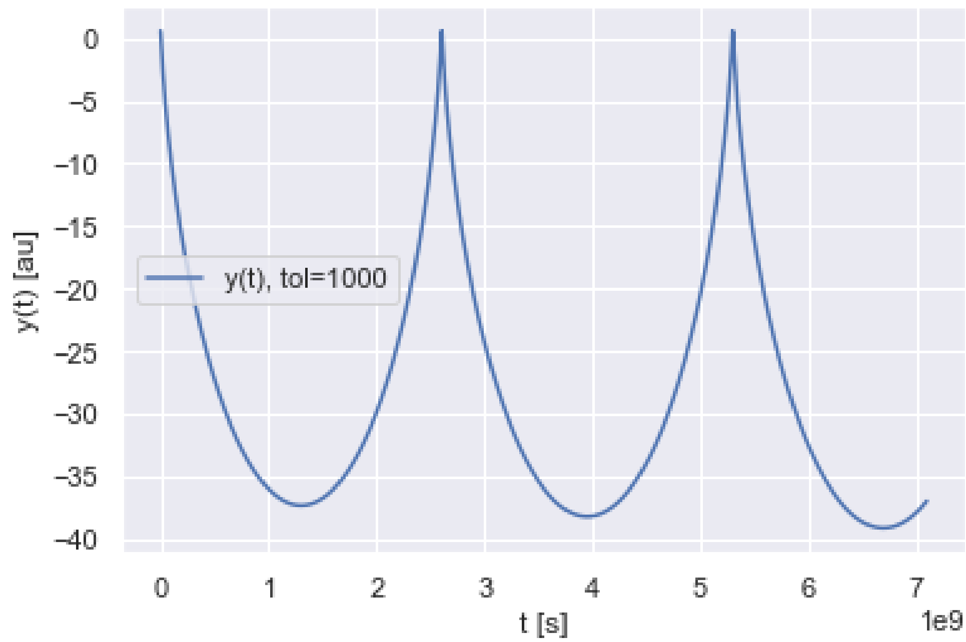

Figure 6: Zależność y(t) w schemacie Eulera, tolerancja=1000m

Czas wykonywania programu: 12.40s.

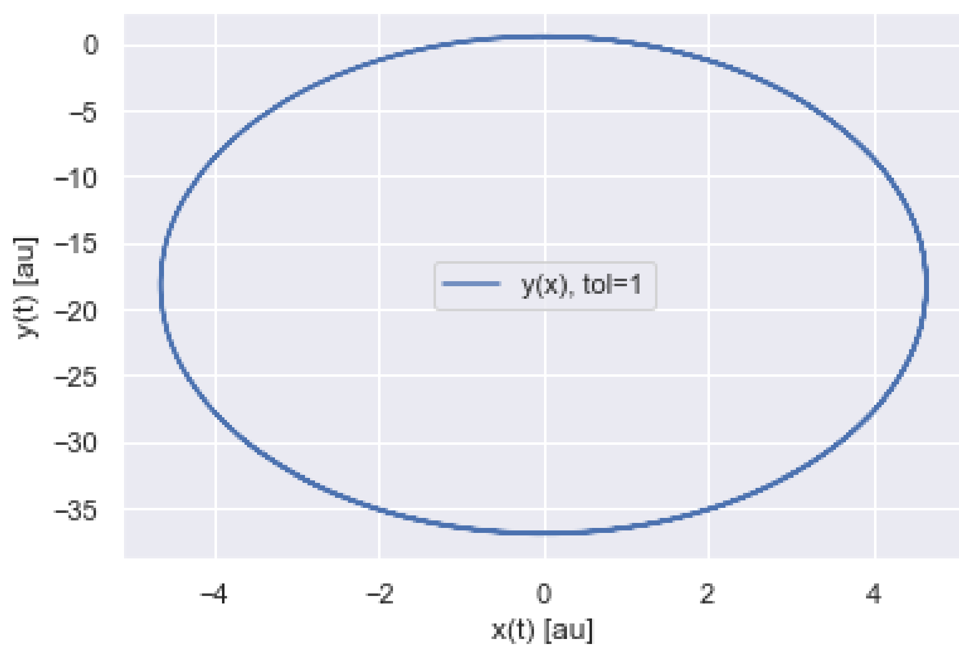

Figure 7: Portret fazowy dla schematu Eulera, tolerancja=1m

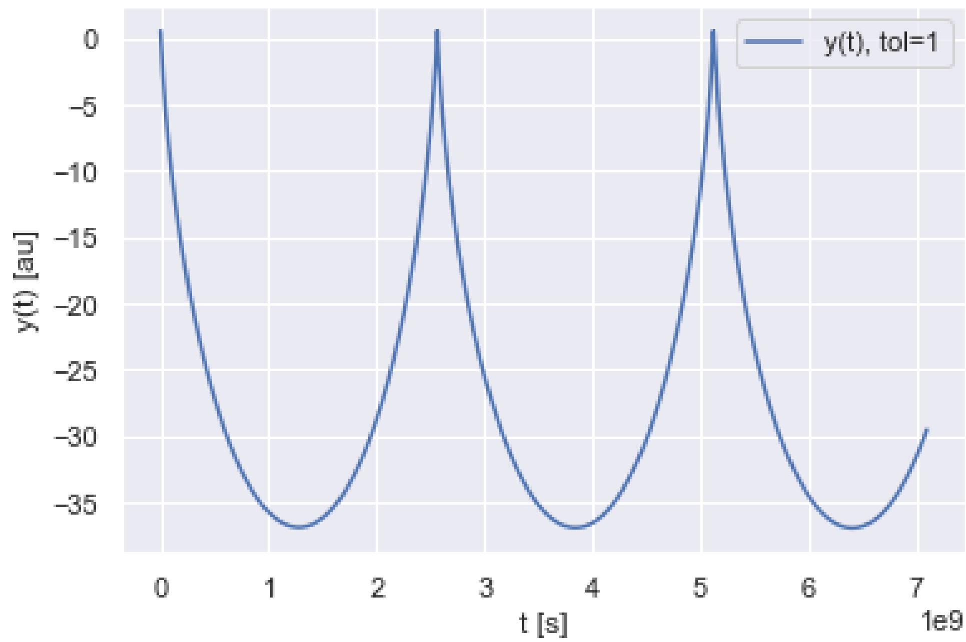

Figure 8: Zależność y(t) w schemacie Eulera, tolerancja=1m

Czas wykonywania programu: 381.79s.
Widać że jest bardzo duża róznica pomiędzy czasem wykonywania programu w zależności od toleracnji błędu.

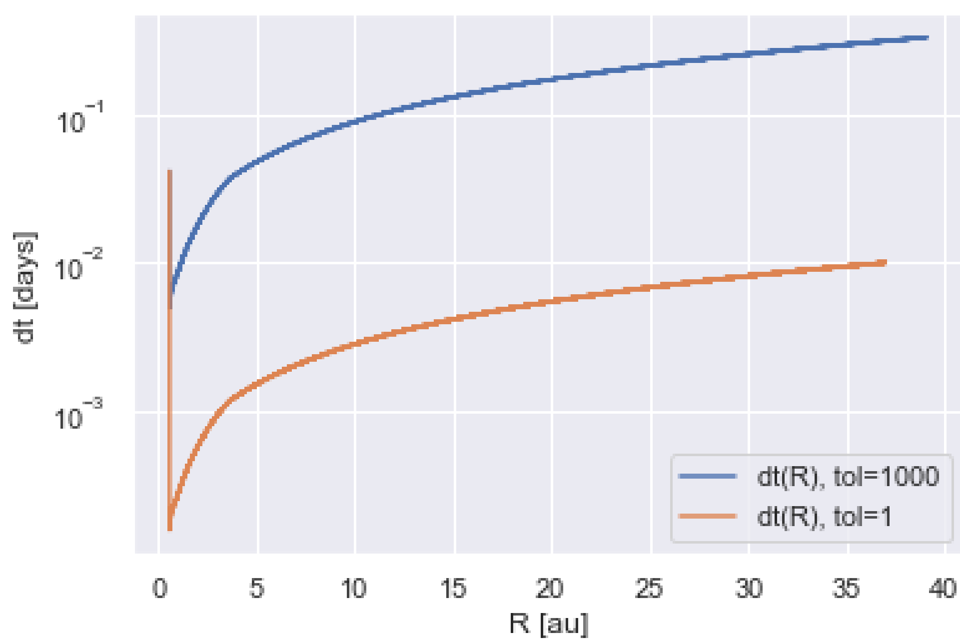

Figure 9: Zależność kroku czasowego Δt od odległości komety od słońca dla obu tolerancji w metodzie Eulera

5 Metoda RK4 z kontrolą błędu i automatycznym doborem kroku czasowego

Powtórzono kroki z punktu 4 dla metody RK4 i zaimplementowano poniższą funkcję:

[40]:
```
def fun_RK4_at(tol,x0,y0,vx0,vy0,dt0):
    tab_x=[x0]
    tab_y=[y0]
    tab_t=[0]
    tab_dt=[dt0]

    t=0
    c=0.9
    n=4 #w metodzie RK4
    vx=vx0
    vy=vy0
    dt=dt0

    while(365*24*3600*75*3>t):
        x = tab_x[-1]
        y = tab_y[-1]

        x1,y1,vx1,vy1=fun_RK4(x,y,vx,vy,dt)

        x2,y2,vx2,vy2=fun_RK4(x,y,vx,vy,dt/2)

        x3,y3,vx3,vy3=fun_RK4(x2,y2,vx2,vy2,dt/2)

        eps_x=(x3-x1)/(2**n-1)
        eps_y=(y3-y1)/(2**n-1)

        if abs(eps_x)>abs(eps_y):
            eps=eps_x
        else:
            eps=eps_y

        if eps<=tol:
            tab_x.append(x1)
            tab_y.append(y1)
            tab_t.append(tab_t[-1]+dt)
            tab_dt.append(dt)
            vx=vx1
            vy=vy1
            t+=dt

        dt=c*dt*abs(tol/eps)**(1/(n+1))
    return tab_x,tab_y,tab_t,tab_dt
```

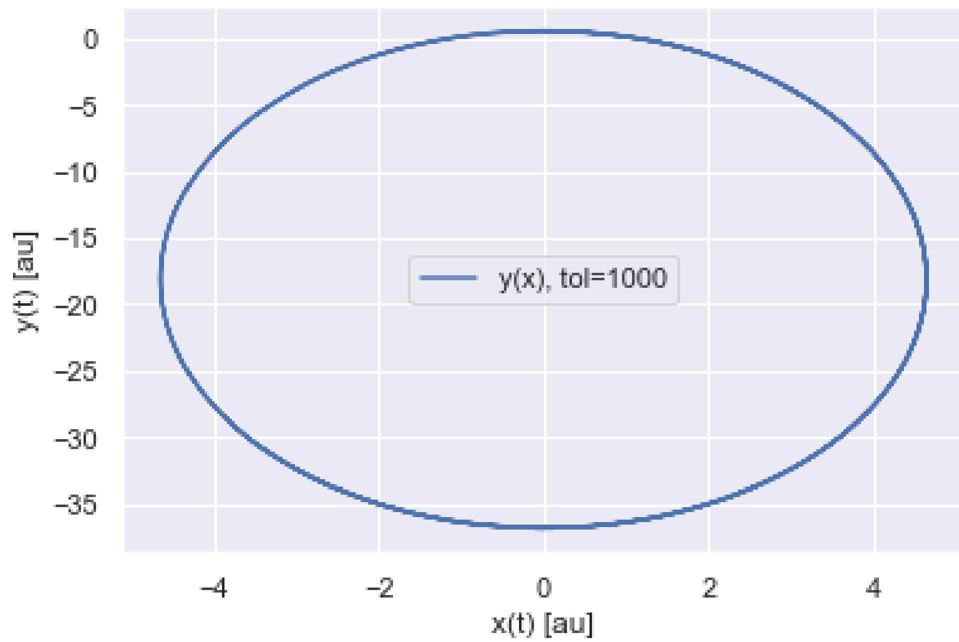

Figure 10: Portret fazowy w metodzie RK4 dla toleracnji=1000m

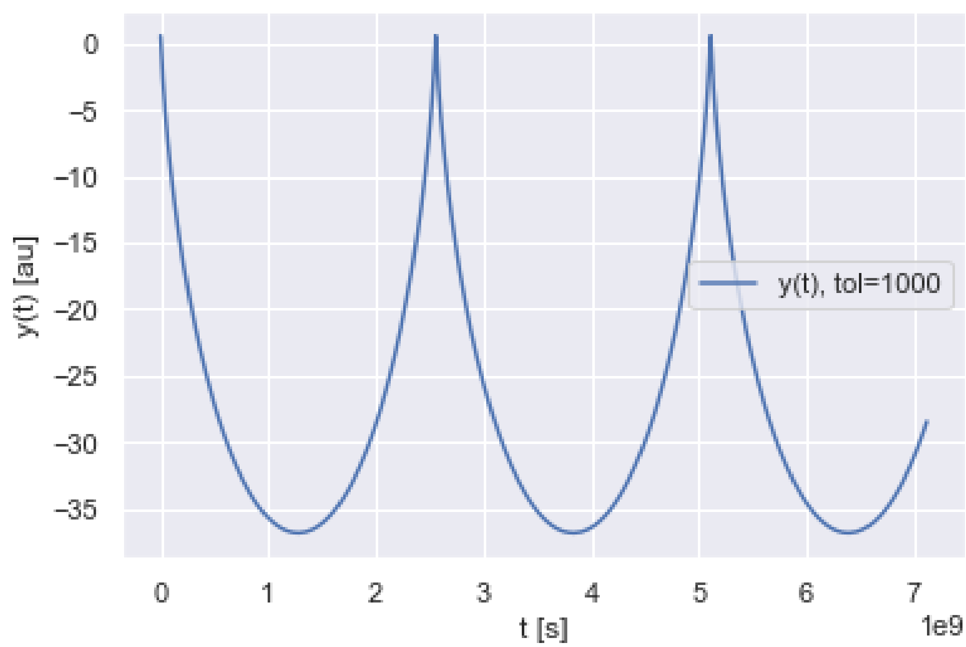

Figure 11: Zależność y(t) w metodzie RK4 dla tolerancji=1000m

Czas wykonywania programu: 2.96s.

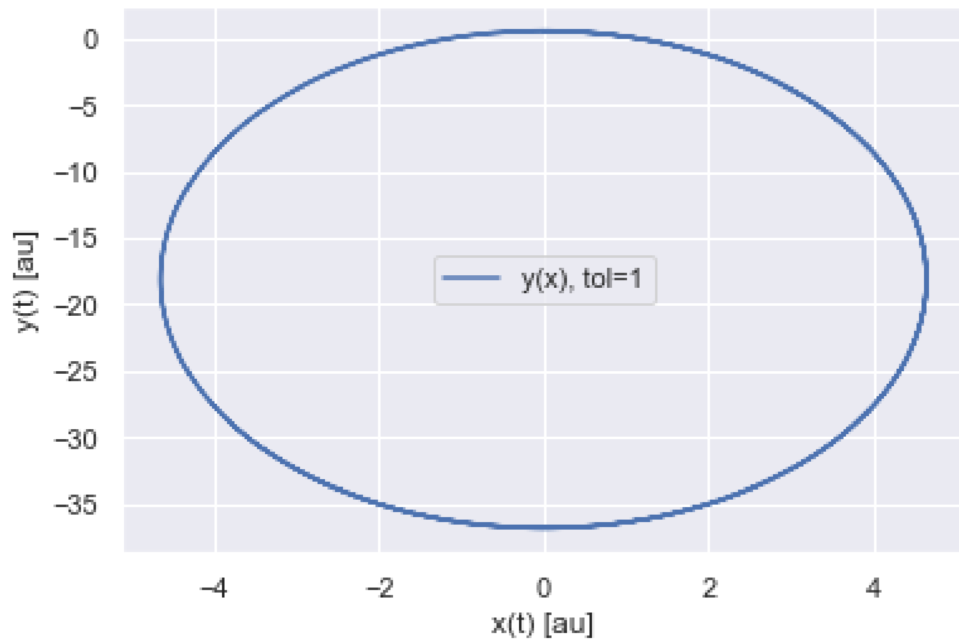

Figure 12: Portret fazowy w metodzie RK4 dla toleracnji=1m

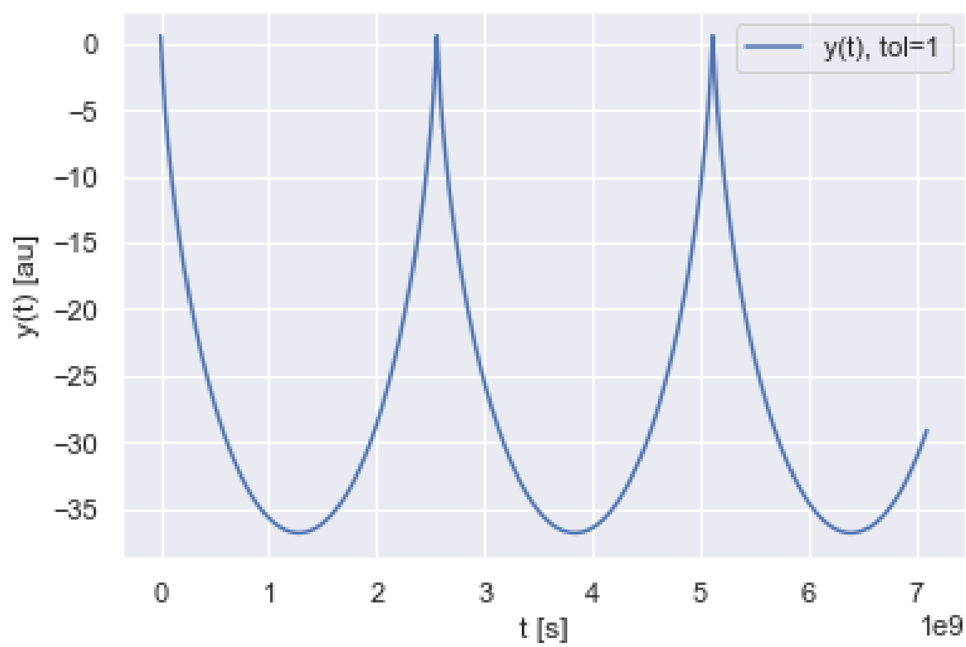

Figure 13: Zależność y(t) w metodzie RK4 dla tolerancji=1m

Czas wykonywania programu: 0.71s. Nie porównywalnie mniejszy czas wykonywania programu między Metodą Eulera a Metodą RK4 z automatycznym doborem kroku czasowego i kontrolą błędów

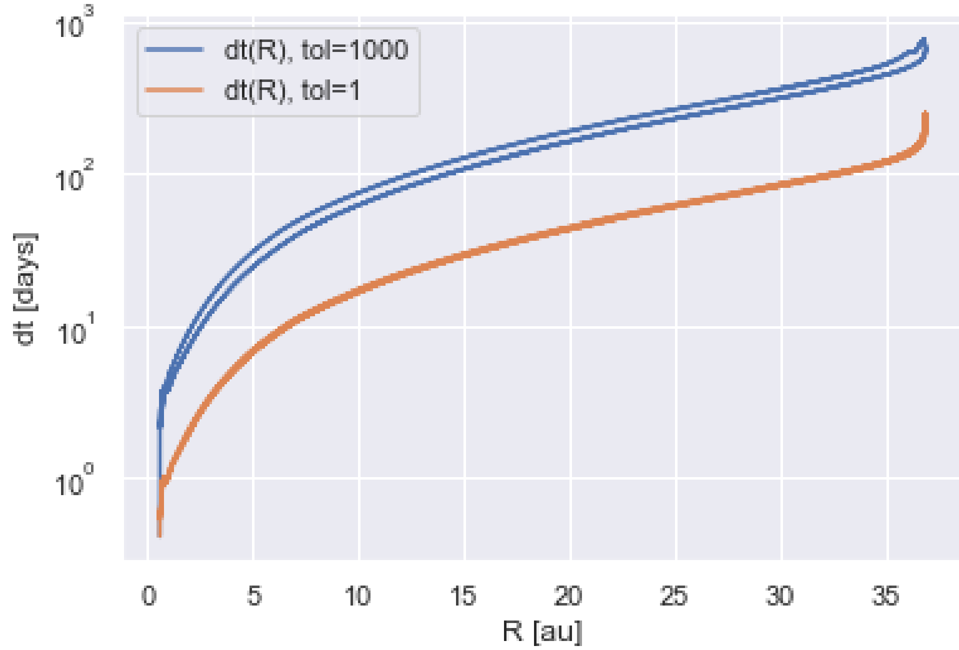

Figure 14: Zależność kroku czasowego Δt od odległości komety od słońca dla obu tolerancji w metodzie RK4