Моделирование Солнечной системы

Выполнил А.А. Змайлов, А. Мирончик, И. Марченков

Научный руководитель А.С. Байгашов

Аннотация

Работа посвящена моделированию Солнечной системы, а также изучению влияния Солнца на планеты Солнечной системы. Был создан аппарат моделирования гравитационно-связанных систем с использованием возможностей среды Python по численному решению уравнений и визуализации их результатов. Было показана зависимость орбитальной скорости планет от их массы и удалённости от Солнца.

Введение

Моделирование гравитационно-связанных систем – отличный способ проиллюстрировать разнообразные законы небесной механики, астрономии и физики вообще. В частности, на моделях удобно демонстрировать зависимости динамических параметров движения небесных объектов от их физических характеристик – масс и начальных положений по отношению к другим объектам. Для этого необходимо выполнить численное решение набора уравнений, начальных и граничных условий к ним, описывающих движение тел в центральном гравитационном поле. В рамках настоящей работы рассматривается Солнечная система, моделирование динамики которой и является её целью.

Постановка задачи

Для описания движения планет вокруг Солнца необходимо решить дифференциальную систему уравнений, описывающих проекции силы притяжения Солнца, действующую на каждую из планет:

$$\overrightarrow{F_G} = \{F_{Gx}, F_{Gy}\}$$

$$\begin{cases} F_{Gx} = -\dfrac{GMm}{(x^2+y^2)^{\frac{3}{2}}} \cdot x \\ F_{Gy} = -\dfrac{GMm}{(x^2+y^2)^{\frac{3}{2}}} \cdot y \end{cases}$$

где M – масса Солнца, m – масса планеты, x и y – её координаты в системе, связанной с неподвижным Солнцем.

$$\begin{cases} \frac{dx}{dt} = v_x \\ \frac{dv_x}{dt} = -\frac{GM}{(x^2+y^2)^{\frac{3}{2}}} \cdot x \\ \frac{dy}{dt} = v_y \\ \frac{dv_y}{dt} = -\frac{GM}{(x^2+y^2)^{\frac{3}{2}}} \cdot y \end{cases}$$

Начальные условия

Для решения поставленной задачи необходимо определить следующие начальные условия:

G=6.67*10**(-11) – ускорение свободного падения

sun_mass=1.9*10**30 – масса Солнца

Марс	x0_mars= 0	v_x0_mars=24100	y0_mars=-228*10**9	v_y0_mars=0
Венера	x0_ven=-108*10**9	v_x0_ven=0	y0_ven=0	v_y0_ven=-35000
Сатурн	x0_saturn=0	v_x0_saturn=-9690	y0_saturn=1430*10**9	v_y0_saturn=0
Юпитер	x0_yup=0	v_x0_yup=-13070	y0_yup=778.57*10**9	v_y0_yup=0
Земля	x0_earth=0	v_x0_earth=-29783	y0_earth=149*10**9	v_y0_earth=0
Меркурий	x0_merc=0	v_x0_merc=47870	y0_merc=46*10**9	v_y0_merc=0
Нептун	x0_neptun = 0	v_x0_neptun = -5434.9	y0_neptun = 4553*10**9	v_y0_neptun = 0
Уран 0	x0_uran = 0	v_x0_uran = -6810	y0_uran = 3000*10**9	v_y0_uran = 0

Результаты моделирования

Численное решение описанной системы уравнений и его визуализация было проведено с использованием среды Python и её открытых библиотек. В результате выполненной работы были получены анимации движения планет по своим орбитам вокруг Солнца (рис. 1).

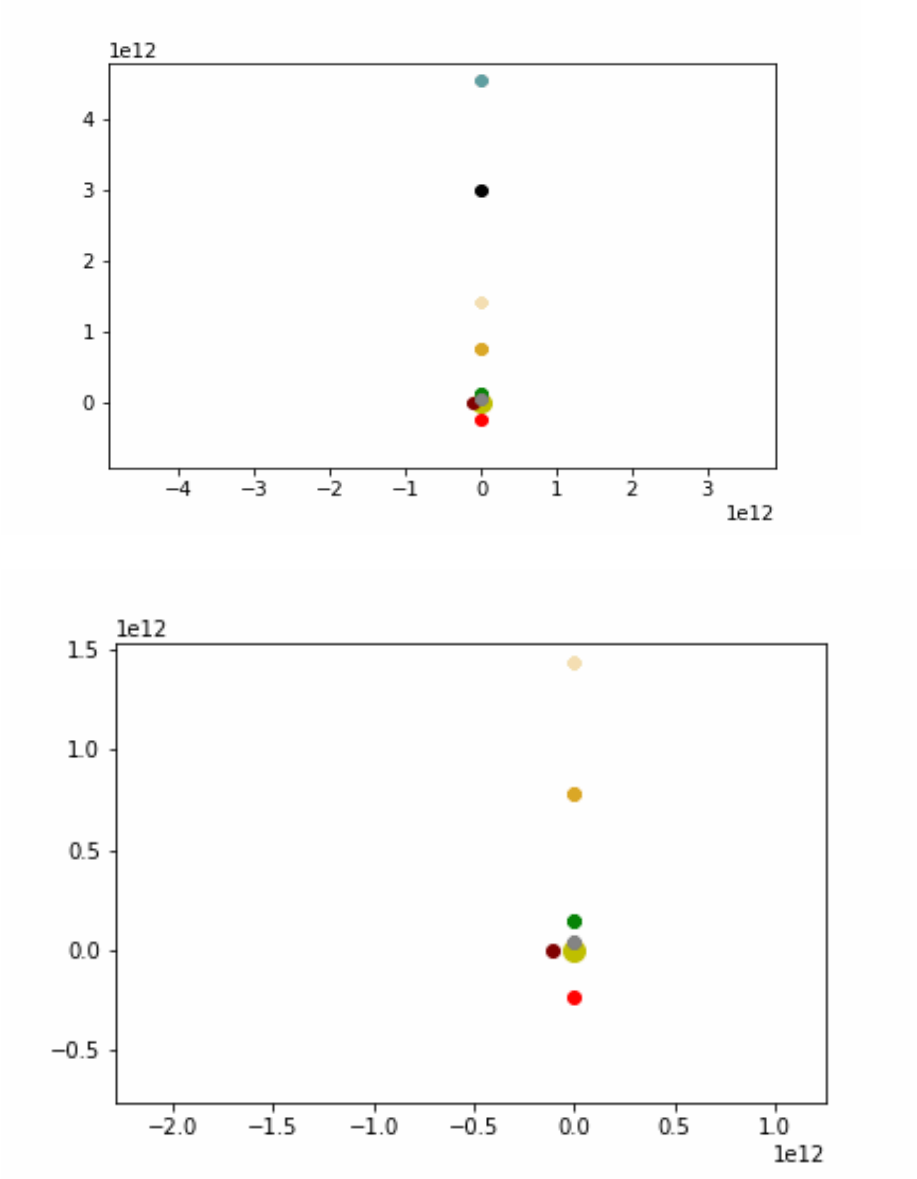

Рис. 1. Солнечная система

Заключение

В ходе выполнения работы было наглядно продемонстрирована возможность использования среды Python для моделирования несложных гравитационно-связанных

систем, что может быть широко использовано в целях иллюстрации различных физических законов. Ключевым преимуществом такого подхода является возможность быстро изменять различные начальные параметры моделируемой системы с тем, чтобы нагляднее продемонстрировать работу того или иного закона. Например, изменив массу Солнца, легко наблюдать, как при этом поменяются траектории движения планет Солнечной системы. Таким образом, такая модель может активно использоваться в образовательных и демонстрационных целях.

Приложение

Листинг кода решения задачи:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import odeint
from matplotlib.animation import ArtistAnimation
sec_y=365*24*60*60
sec_d=24*60*60
years=5
t=np.arange(0,years*sec_y,sec_d*10)


def grav_func(z,t):
     (x_mars, v_x_mars, y_mars, v_y_mars,
      x_ven, v_x_ven, y_ven, v_y_ven,
      x_saturn, v_x_saturn, y_saturn, v_y_saturn,
      x_yup, v_x_yup, y_yup, v_y_yup,
      x_earth, v_x_earth, y_earth, v_y_earth,
      x_merc, v_x_merc, y_merc, v_y_merc,
      x_neptun,v_x_neptun, y_neptun,v_y_neptun,
      x_uran, v_x_uran, y_uran, v_y_uran) = z


     dxdt_mars= v_x_mars
     dv_xdt_mars= -G*sun_mass* x_mars/(x_mars**2+y_mars**2)**1.5
     dydt_mars= v_y_mars
     dv_ydt_mars= -G*sun_mass* y_mars/(x_mars**2+y_mars**2)**1.5
```

```
dxdt_ven= v_x_ven
dv_xdt_ven= -G*sun_mass* x_ven/(x_ven**2+y_ven**2)**1.5
dydt_ven= v_y_ven
dv_ydt_ven= -G*sun_mass* y_ven/(x_ven**2+y_ven**2)**1.5

dxdt_saturn= v_x_saturn
dv_xdt_saturn= -G*sun_mass* x_saturn/(x_saturn**2+y_saturn**2)**1.5
dydt_saturn= v_y_saturn
dv_ydt_saturn= -G*sun_mass* y_saturn/(x_saturn**2+y_saturn**2)**1.5

dxdt_yup= v_x_yup
dv_xdt_yup= -G*sun_mass* x_yup/(x_yup**2+y_yup**2)**1.5
dydt_yup= v_y_yup
dv_ydt_yup= -G*sun_mass* y_yup/(x_yup**2+y_yup**2)**1.5

dxdt_earth= v_x_earth
dv_xdt_earth= -G*sun_mass* x_earth/(x_earth**2+y_earth**2)**1.5
dydt_earth= v_y_earth
dv_ydt_earth= -G*sun_mass* y_earth/(x_earth**2+y_earth**2)**1.5

dxdt_merc= v_x_merc
dv_xdt_merc= -G*sun_mass* x_merc/(x_merc**2+y_merc**2)**1.5
dydt_merc= v_y_merc
dv_ydt_merc= -G*sun_mass* y_merc/(x_merc**2+y_merc**2)**1.5

dxdt_neptun = v_x_neptun
  dv_xdt_neptun = -G * sun_mass * x_neptun / (x_neptun**2 + y_neptun**2)**1.5
dydt_neptun = v_y_neptun
dv_ydt_neptun = -G * sun_mass * y_neptun / (x_neptun**2 + y_neptun**2)**1.5

dxdt_uran= v_x_uran
dv_xdt_uran= -G*sun_mass* x_uran/(x_uran**2+y_uran**2)**1.5
dydt_uran= v_y_uran
dv_ydt_uran= -G*sun_mass* y_uran/(x_uran**2+y_uran**2)**1.5
```

```
    return (dxdt_mars, dv_xdt_mars, dydt_mars, dv_ydt_mars,
            dxdt_ven, dv_xdt_ven, dydt_ven, dv_ydt_ven,
            dxdt_saturn,dv_xdt_saturn, dydt_saturn, dv_ydt_saturn,
            dxdt_yup,dv_xdt_yup,dydt_yup, dv_ydt_yup,
            dxdt_earth,dv_xdt_earth,dydt_earth, dv_ydt_earth,
            dxdt_merc,dv_xdt_merc,dydt_merc, dv_ydt_merc,
            dxdt_neptun,dv_xdt_neptun,dydt_neptun,dv_ydt_neptun,
            dxdt_uran, dv_xdt_uran,dydt_uran,dv_ydt_uran)

x0_mars= 0
v_x0_mars=24100
y0_mars=-228*10**9
v_y0_mars=0

x0_ven=-108*10**9
v_x0_ven=0
y0_ven=0
v_y0_ven=-35000

x0_saturn=0
v_x0_saturn=-9690
y0_saturn=1430*10**9
v_y0_saturn=0

x0_yup=0
v_x0_yup=-13070
y0_yup=778.57*10**9
v_y0_yup=0

x0_earth=0
v_x0_earth=-29783
y0_earth=149*10**9
```

```
v_y0_earth=0


x0_merc=0
v_x0_merc=47870
y0_merc=46*10**9
v_y0_merc=0


x0_neptun = 0
v_x0_neptun = - 5434.9
y0_neptun = 4553*10**9
v_y0_neptun = 0


x0_uran = 0
v_x0_uran = -6810
y0_uran = 3000*10**9
v_y0_uran =  0


z0=(x0_mars,v_x0_mars,y0_mars, v_y0_mars,
    x0_ven,v_x0_ven, y0_ven,v_y0_ven,
    x0_saturn, v_x0_saturn,y0_saturn,v_y0_saturn,
    x0_yup, v_x0_yup, y0_yup,  v_y0_yup,
    x0_earth, v_x0_earth, y0_earth, v_y0_earth,
    x0_merc, v_x0_merc, y0_merc, v_y0_merc,
    x0_neptun,v_x0_neptun,y0_neptun,v_y0_neptun,
    x0_uran,v_x0_uran,y0_uran,v_y0_uran)


G=6.67*10**(-11)
sun_mass=1.9*10**30
sol=odeint(grav_func, z0, t)


fig=plt.figure()
planets=[]


for i in range(0, len(t), 1):
```

```
    sun, =plt.plot([0], [0], 'yo', ms=10)

    mars_line, = plt.plot(sol[i,0], sol[i, 2], 'ro')

    ven_line, = plt.plot(sol[i,4], sol[i, 6],'o', color= 'maroon')

    saturn_line, = plt.plot(sol[i,8], sol[i, 10], 'o', color='wheat')

    yup_line, = plt.plot(sol[i,12], sol[i, 14], 'o', color='goldenrod')

    earth_line, = plt.plot(sol[i,16], sol[i,18], 'o', color='green')

    merc_line, = plt.plot(sol[i,20], sol[i,22], 'o', color='grey')

    neptun_line, = plt.plot(sol[i,24], sol[i,26], 'o', color='cadetblue')

    uran_line, = plt.plot(sol[i,28], sol[i,30], 'o', color='black')

    planets.append([sun,  mars_line, ven_line,
                    saturn_line, yup_line, earth_line,
                    merc_line, neptun_line, uran_line])

ani = ArtistAnimation(fig, planets, interval= 50)
plt.axis('equal')
ani.save('solarsys20.gif')
```