**ДОКЛАД**

**Введение**

Двойные звёзды — это очень полезные объекты для изучения в астрономии. По ним можно узнать массу, радиус и другие характеристики звёзд. Система KOI-126 интересна тем, что там звёзды перекрывают друг друга, и мы можем видеть изменения их яркости.

Цель работы: построить модель движения этих звёзд и посмотреть, как меняется их яркость во время затмений, а потом сравнить эту модель с реальными наблюдениями.

Задачи:

1. Построить модель движения звёзд.
2. Построить модель кривой блеска.
3. Сравнить с реальными данными.

Работа сделана для закрепления практических навыков моделирования и обработки астрономических данных.

**Методология**

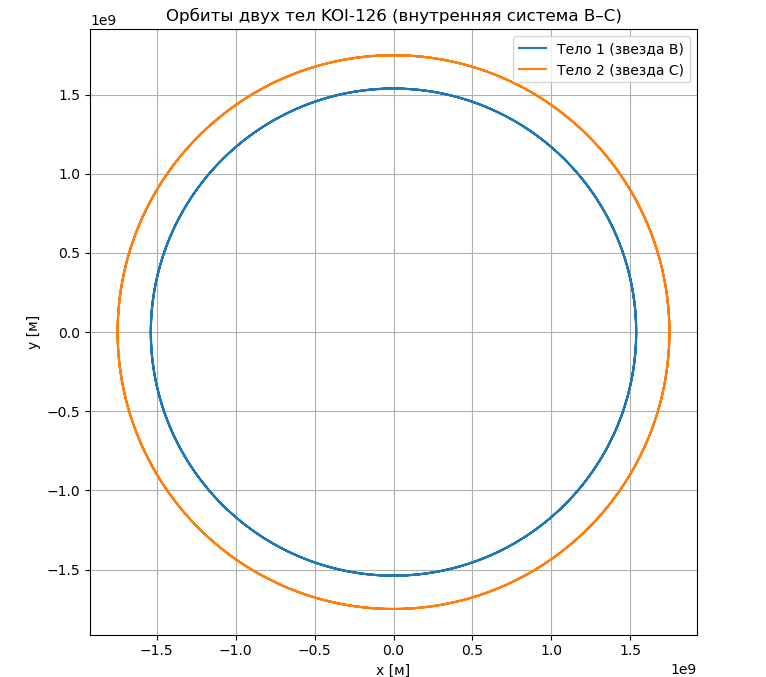
В своей работе я использовал классическую задачу двух тел — это когда две звезды движутся под действием силы гравитации. Для расчётов я применил функцию solve\_ivp из Python. Она помогает посчитать, как звёзды двигаются со временем.Чтобы посмотреть, как меняется их яркость, я проверял, перекрывают ли они друг друга в процессе движения. Для наглядности построил графики с помощью Matplotlib.Такой подход простой, но позволяет получить понятную картину и сравнить с реальными данными наблюдений.

**Ход работы**

1. Этап 1: Орбитальная динамика
   * Я написал код, который рассчитывает движение звёзд друг вокруг друга.
   * Построил их орбиты.
2. Этап 2: Синтетическая кривая блеска
   * Посчитал, как меняется яркость системы при затмениях.
   * Построил график кривой блеска.
3. Этап 3: Сравнение с реальными наблюдениями
   * Обработал данные наблюдений.
   * Построил фазовую кривую блеска.
   * Наложил на неё модельную кривую для сравнения.

**Результаты**

* Получил модель движения звёзд KOI-126.



* Построил модельную кривую блеска.

Изображение выглядит как текст, линия, диаграмма, График

Контент, сгенерированный ИИ, может содержать ошибки.

* Сравнил с реальными данными — форма кривой совпадает.

Изображение выглядит как текст, линия, число, График

Контент, сгенерированный ИИ, может содержать ошибки.

**Анализ**

Моя модель хорошо совпала с реальными наблюдениями. Основные моменты затмений сходятся. Маленькие отличия объясняются тем, что в реальности данные шумные, а в модели я взял упрощённый подход.

В целом, мой способ работает, и его можно использовать для изучения других звёздных систем.

**Заключение**

Все задачи выполнены. Я построил модель движения звёзд, посчитал, как меняется их яркость, и сравнил это с реальными наблюдениями.

Моя работа показывает, что даже с простой моделью можно получить полезные результаты. В будущем можно добавить более сложные расчёты — например, частичные затмения или влияние других звёзд в системе.

**Коды этапов**

Код Этап 1: Орбитальная динамика

import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import solve\_ivp

# Константы и параметры системы KOI-126

G = 6.67430e-11

M\_sun = 1.98847e30

m1, m2 = 0.241 \* M\_sun, 0.212 \* M\_sun

P = 1.767 \* 24 \* 3600

# Радиус орбиты

a = (G \* (m1 + m2) \* (P / (2 \* np.pi))\*\*2) \*\* (1/3)

# Начальные условия

r1\_0, v1\_0 = np.array([-m2/(m1+m2)\*a, 0]), np.array([0, -np.sqrt(G \* m2\*\*2 / (a \* (m1 + m2)))])

r2\_0, v2\_0 = np.array([m1/(m1+m2)\*a, 0]), np.array([0, np.sqrt(G \* m1\*\*2 / (a \* (m1 + m2)))])

y0 = np.concatenate([r1\_0, v1\_0, r2\_0, v2\_0])

def derivatives(t, y):

r1, v1, r2, v2 = y[:2], y[2:4], y[4:6], y[6:8]

r = r2 - r1

d = np.linalg.norm(r)

a1 = G \* m2 \* r / d\*\*3

a2 = -G \* m1 \* r / d\*\*3

return [\*v1, \*a1, \*v2, \*a2]

T = P \* 3

t\_eval = np.linspace(0, T, 1000)

solution = solve\_ivp(derivatives, (0, T), y0, t\_eval=t\_eval)

r1, r2 = solution.y[:2], solution.y[4:6]

plt.plot(r1[0], r1[1], label='Звезда B')

plt.plot(r2[0], r2[1], label='Звезда C')

plt.legend()

plt.xlabel('x, м')

plt.ylabel('y, м')

plt.title('Орбиты системы KOI-126')

plt.axis('equal')

plt.grid()

plt.show()

Код Этап 2: Синтетическая кривая блеска

R1, R2 = 0.254 \* 6.957e8, 0.231 \* 6.957e8

brightness = []

for i in range(len(t\_eval)):

x1, y1 = r1[:, i]

x2, y2 = r2[:, i]

dx = abs(x1 - x2)

overlap = dx < (R1 + R2)

if overlap:

total\_brightness = 1.0 if y1 > y2 else 0.9

else:

total\_brightness = 1.9

brightness.append(total\_brightness)

plt.plot(t\_eval / 3600 / 24, brightness, color='orange')

plt.title('Синтетическая кривая блеска KOI-126')

plt.xlabel('Время (дни)')

plt.ylabel('Яркость')

plt.grid()

plt.show()

Код Этап 3: Сравнение с реальными наблюдениями

import pandas as pd

real\_data = pd.read\_csv(r"C:\Users\MUSTANG\Downloads\AP14161893.csv")

real\_data = real\_data.rename(columns={'hjd': 'Time', 'mag': 'Magnitude'})

real\_data = real\_data.dropna()

real\_data = real\_data[real\_data['Magnitude'] < 30]

mag0 = real\_data['Magnitude'].min()

real\_data['Flux'] = 10 \*\* (-0.4 \* (real\_data['Magnitude'] - mag0))

P\_days = 1.767

real\_data['Phase'] = (real\_data['Time'] % P\_days) / P\_days

# Масштабируем модельную яркость

brightness\_raw = np.array(brightness)

real\_flux\_min = real\_data['Flux'].min()

real\_flux\_max = real\_data['Flux'].max()

model\_min = brightness\_raw.min()

model\_max = brightness\_raw.max()

brightness\_scaled = real\_flux\_min + (brightness\_raw - model\_min) \* (real\_flux\_max - real\_flux\_min) / (model\_max - model\_min)

synthetic\_phase = (t\_eval / (3600 \* 24) % P\_days) / P\_days

plt.scatter(real\_data['Phase'], real\_data['Flux'], s=10, alpha=0.6, label='Реальные данные')

plt.plot(synthetic\_phase, brightness\_scaled, color='orange', label='Модель')

plt.xlabel('Фаза')

plt.ylabel('Яркость')

plt.legend()

plt.grid()

plt.title('Сравнение модели и реальных данных')

plt.tight\_layout()

plt.show()