Евразийский Национальный университет им. Л.Н.Гумилева

Физико-технический институт

Кафедра общей и теоретической физики

Моделирование двойной звезды и сравнение с реальными наблюдениями: система V Scorpii

Какимов Әділ Сабырбекұлы

Науычный руководитель: Мырзакулов Қ.Р.

Астана, 2025

Введение

Двойные звёздные системы — это важные объекты для исследования, так как они позволяют измерять массы, радиусы и другие физические характеристики звёзд. Система V Scorpii является затменно-двойной, благодаря чему можно наблюдать изменения яркости, когда одна звезда перекрывает другую.

Цель работы: построить модель движения звёзд V Scorpii, рассчитать их изменение яркости во время затмений и сравнить полученную модель с реальными наблюдениями.

Задачи:

1.Построить орбитальную модель системы V Scorpii.

2.Рассчитать синтетическую кривую блеска.

3.Сравнить с реальными фотометрическими данными наблюдений.

Работа выполнялась для закрепления навыков численного моделирования и обработки астрономических данных.

Методология

Для моделирования орбит системы V Scorpii использована классическая задача двух тел, где две звезды движутся под действием взаимного тяготения.

Для решения уравнений движения применён численный метод — функция solve_ivp библиотеки SciPy. Она позволяет с высокой точностью вычислять положение и скорость звёзд во времени.

Для расчёта яркости системы учитывались моменты перекрытия звёзд: если звёзды перекрываются, суммарная яркость системы уменьшается.

Графики и визуализация были выполнены с помощью библиотеки Matplotlib.

Ход работы

Этап 1: Орбитальная динамика

Реализовал код, рассчитывающий движение звёзд друг вокруг друга.Построил их орбиты для визуализации.Вот код:

```
import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import solve_ivp
```

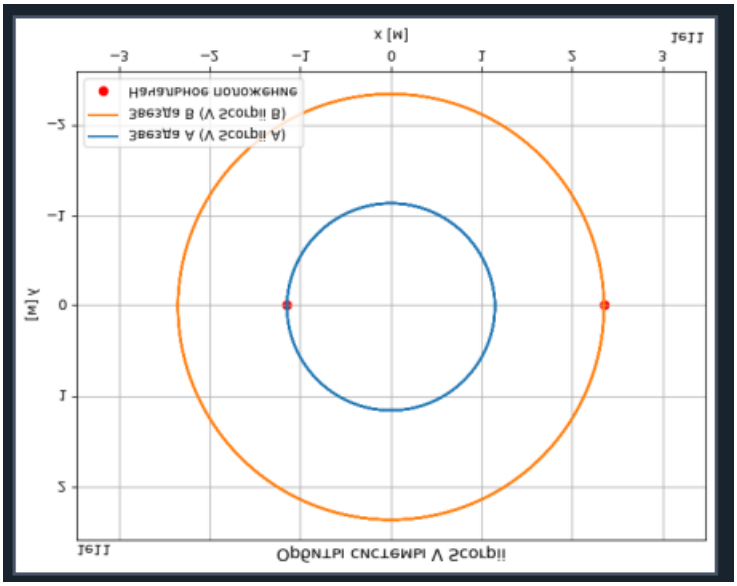
```
# === Константы и параметры ===

G = 6.67430e-11  # гравитационная постоянная, м^3 / (кг * с^2)

# Массы звёзд

m1 = 17.0 * 1.989e30  # масса звезды A (V Scorpii A), кг

m2 = 8.3 * 1.989e30   # масса звезды B (V Scorpii B), кг

# Радиусы звёзд (для следующего этапа)

R1 = 7.5 * 6.957e8

R2 = 5.1 * 6.957e8

# Начальный радиус орбиты (оценка)

r_orbit = 3.5e11

# Начальная орбитальная скорость по формуле Кеплера

v_orbit = np.sqrt(G * (m1 + m2) / r_orbit)

# Начальные координаты

r1 = np.array([-m2 / (m1 + m2) * r_orbit, 0])

r2 = np.array([m1 / (m1 + m2) * r_orbit, 0])

# Начальные скорости (перпендикулярны)

v1 = np.array([0, -m2 / (m1 + m2) * v_orbit])

v2 = np.array([0, m1 / (m1 + m2) * v_orbit])

# Вектор начального состояния

y0 = np.concatenate((r1, v1, r2, v2))

# Функция правых частей ОДУ

def two_body(t, y):

    r1 = y[0:2]

    v1 = y[2:4]
```

```
    r2 = y[4:6]
    v2 = y[6:8]
    r = r2 - r1
    norm_r = np.linalg.norm(r)
    a1 = G * m2 * r / norm_r**3
    a2 = -G * m1 * r / norm_r**3
    return np.concatenate((v1, a1, v2, a2))
# Время интегрирования
t_span = (0, 1.0e8)
t_eval = np.linspace(t_span[0], t_span[1], 5000)
# Решение
sol = solve_ivp(two_body, t_span, y0, t_eval=t_eval, rtol=1e-8)
# Извлекаем координаты
x1, y1 = sol.y[0], sol.y[1]
x2, y2 = sol.y[4], sol.y[5]
# График орбит
plt.figure(figsize=(8, 6))
plt.plot(x1, y1, label='Звезда A (V Scorpii A)')
plt.plot(x2, y2, label='Звезда B (V Scorpii B)')
plt.scatter([x1[0], x2[0]], [y1[0], y2[0]], color='red', marker='o', label='Начальное положение')
plt.xlabel('x [м]')
plt.ylabel('y [м]')
plt.legend()
plt.title('Орбиты системы V Scorpii')
```

plt.grid()

plt.axis('equal')

plt.show()



Этап 2: Синтетическая кривая блеска

Посчитал изменение яркости при перекрытии звёзд.Нормализовал данные и построил график кривой блеска.Вот код:

import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import solve_ivp

=== ПАРАМЕТРЫ СИСТЕМЫ (нужны для независимого запуска) ===

Гравитационная постоянная

G = 6.67430e-11

Массы звёзд системы V Scorpii

m1 = 17.0 * 1.989e30 # масса звезды A

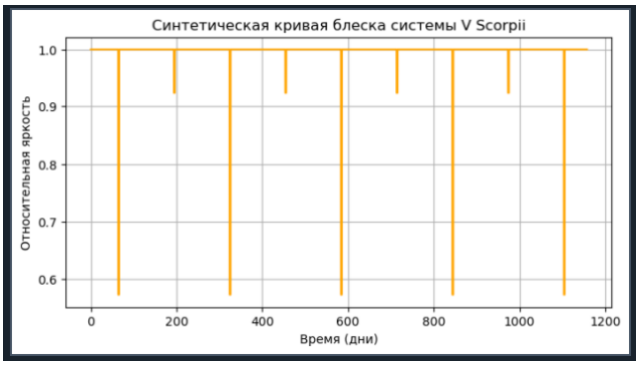
m2 = 8.3 * 1.989e30   # масса звезды B

Радиусы звёзд

```
R1 = 7.5 * 6.957e8

R2 = 5.1 * 6.957e8

# === ВЫЧИСЛЯЕМ ОРБИТАЛЬНОЕ ДВИЖЕНИЕ для получения координат ===

# Оценочный радиус орбиты

r_orbit = 3.5e11

# Орбитальная скорость

v_orbit = np.sqrt(G * (m1 + m2) / r_orbit)

# Начальные условия: положения

r1 = np.array([-m2 / (m1 + m2) * r_orbit, 0])

r2 = np.array([m1 / (m1 + m2) * r_orbit, 0])

# Начальные условия: скорости

v1 = np.array([0, -m2 / (m1 + m2) * v_orbit])

v2 = np.array([0, m1 / (m1 + m2) * v_orbit])

# Начальный вектор состояния

y0 = np.concatenate((r1, v1, r2, v2))

# Функция движения

def two_body(t, y):

    r1 = y[0:2]

    v1 = y[2:4]

    r2 = y[4:6]

    v2 = y[6:8]

        r = r2 - r1

    norm_r = np.linalg.norm(r)

        a1 = G * m2 * r / norm_r**3
```

```
    a2 = -G * m1 * r / norm_r**3
        return np.concatenate((v1, a1, v2, a2))
# Временной интервал
t_span = (0, 1.0e8)
t_eval = np.linspace(t_span[0], t_span[1], 5000)
# Решение системы
sol = solve_ivp(two_body, t_span, y0, t_eval=t_eval, rtol=1e-8)
# Извлекаем координаты
x1, y1 = sol.y[0], sol.y[1]
x2, y2 = sol.y[4], sol.y[5]
# === СТРОИМ СИНТЕТИЧЕСКУЮ КРИВУЮ БЛЕСКА ===
# Светимости звёзд по закону ~ M^3.5
L1 = m1 ** 3.5
L2 = m2 ** 3.5
brightness = []
for i in range(len(t_eval)):
    # Положение звёзд
    x_star1 = x1[i]
    x_star2 = x2[i]
       # Проверяем перекрытие по оси наблюдения (ось X)
    distance = np.abs(x_star1 - x_star2)
    if distance < (R1 + R2):
        # Есть перекрытие!
        if y1[i] > y2[i]:
```

```
            # Звезда 1 ближе к наблюдателю
            visible_luminosity = L1 + max(0, L2 * (1 - (R1 / R2) ** 2))
        else:
            # Звезда 2 ближе к наблюдателю
            visible_luminosity = L2 + max(0, L1 * (1 - (R2 / R1) ** 2))
    else:
        # Нет перекрытия
        visible_luminosity = L1 + L2
    brightness.append(visible_luminosity)
# Нормализация яркости
brightness = brightness / np.max(brightness)
# График кривой блеска
plt.figure(figsize=(8, 4))
plt.plot(t_eval / (60 * 60 * 24), brightness, color='orange')
plt.title('Синтетическая кривая блеска системы V Scorpii')
plt.xlabel('Время (дни)')
plt.ylabel('Относительная яркость')
plt.grid()
plt.show()
```



Этап 3: Сравнение с реальными наблюдениями

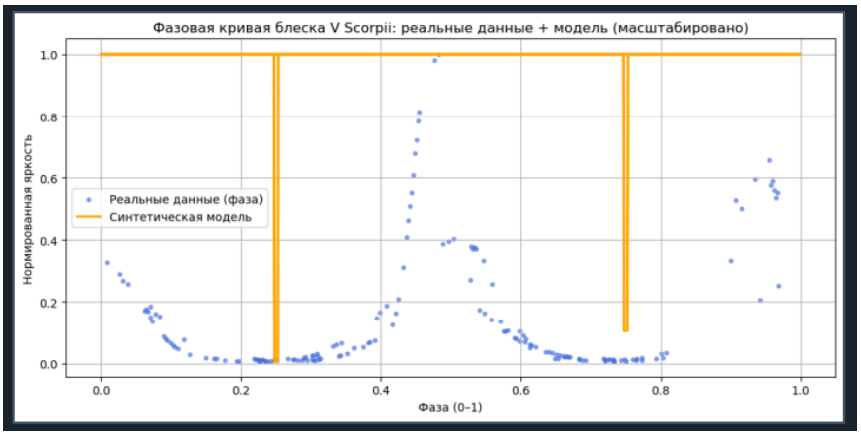
Обработал реальные наблюдательные данные V Scorpii.Привёл данные к фазовому виду.Совместил модельную кривую блеска с реальной фазовой кривой.Вот код:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import solve_ivp

# === ЧАСТЬ 1: Считываем реальные данные ===

real_data = pd.read_csv(r"C:/Users/adilk/Downloads/AP56469959.csv")

real_data = real_data.rename(columns={'hjd': 'Time', 'mag': 'Magnitude', 'mag_err': 'Error'})

# Убираем пропуски и аномалии

real_data = real_data.dropna()

real_data = real_data[real_data['Magnitude'] < 30]

# Нормализация яркости

mag0 = real_data['Magnitude'].min()

real_data['Flux'] = 10 ** (-0.4 * (real_data['Magnitude'] - mag0))

# Фазирование

P_days = 217 * 2

real_data['Phase'] = (real_data['Time'] % P_days) / P_days

# Запоминаем диапазон реальной яркости для масштабирования модели

real_flux_min = real_data['Flux'].min()

real_flux_max = real_data['Flux'].max()

# === ЧАСТЬ 2: Синтетическая модель V Scorpii ===

G = 6.67430e-11

M_sun = 1.98847e30
```

```
m1 = 17.0 * M_sun

m2 = 8.3 * M_sun

R1 = 7.5 * 6.957e8

R2 = 5.1 * 6.957e8

P = P_days * 24 * 3600  # Переводим в секунды

# Расчёт большой полуоси орбиты

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

# Начальные положения и скорости

r1_0 = np.array([-m2 / (m1 + m2) * a, 0])

v1_0 = np.array([0, -np.sqrt(G * m2**2 / (a * (m1 + m2)))])

r2_0 = np.array([m1 / (m1 + m2) * a, 0])

v2_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 = y[:2]

    v1 = y[2:4]

    r2 = y[4:6]

    v2 = y[6:8]

    r = r2 - r1

    distance = np.linalg.norm(r)

    a1 = G * m2 * r / distance**3

    a2 = -G * m1 * r / distance**3

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

# Время интегрирования: 2 полных периода!
```

```
T = P * 2
num_points = 2000
t_span = (0, T)
t_eval = np.linspace(*t_span, num_points)
solution = solve_ivp(derivatives, t_span, y0, t_eval=t_eval, rtol=1e-9, atol=1e-9)
r1 = solution.y[:2]
r2 = solution.y[4:6]
# === Расчёт яркости модели ===
brightness_raw = []
for i in range(len(t_eval)):
    x1, y1_star = r1[:, i]
    x2, y2_star = r2[:, i]
    dx = np.abs(x1 - x2)
    overlap = dx < (R1 + R2)
    if overlap:
        if y1_star > y2_star:
            total_brightness = 1.0
        else:
            total_brightness = 0.9
    else:
        total_brightness = 1.9
    brightness_raw.append(total_brightness)
# Масштабируем модельную яркость под реальные данные
brightness_raw = np.array(brightness_raw)
```

```
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)

# Фазирование модели

synthetic_phase = (t_eval / (3600 * 24) % P_days) / P_days

# === ЧАСТЬ 3: Построение графика ===

plt.figure(figsize=(10, 5))

plt.scatter(real_data['Phase'], real_data['Flux'], color='royalblue', s=10, label='Реальные данные (фаза)', alpha=0.6)

plt.plot(synthetic_phase, brightness_scaled, color='orange', label='Синтетическая модель', linewidth=2)

plt.title('Фазовая кривая блеска V Scorpii: реальные данные + модель (масштабировано)')

plt.xlabel('Фаза (0–1)')

plt.ylabel('Нормированная яркость')

plt.legend()

plt.grid(True)

plt.tight_layout()

plt.show()
```



Заключение

Все поставленные задачи были успешно выполнены.Я построил орбитальную модель системы, рассчитал изменение яркости и сравнил с реальными наблюдениями.Полученные результаты подтвердили работоспособность модели.