Евразийский Национальный университет им. Л.Н.Гумилева

Физико – технический институт

Кафедра общей и теоретической физики

Моделирование 5 Serpentis

Қажымұхан Абылай

Научный руководитель Нуркасымова.С.Н

Астана, 2025

Введение

Актуальность темы: Двойные звёзды играют важную роль в астрономии, потому что с их помощью можно узнавать массу, размер и яркость звёзд. Особенно интересна система 5 Змеи (5 Serpentis), которая состоит из двух звёзд, вращающихся вокруг общего центра. Благодаря своей относительной близости и хорошо изученной структуре, эта система подходит для построения моделей. Сейчас доступны реальные данные наблюдений, в том числе кривые блеска — графики, на которых видно, как со временем меняется яркость звезды. Сравнивая такие графики с моделями, мы можем проверить, насколько точно они описывают происходящее в системе. Поэтому изучение 5 Змеи с помощью моделирования и сравнение результатов с наблюдениями представляет собой важную и актуальную задачу.

Цель работы: Построить модель поведения двойной звёздной системы 5 Змеи, рассчитать её кривую блеска с учётом затмений и сравнить полученные данные с реальными наблюдениями.

1. Решить задачу движения двух тел, чтобы описать орбиты звёзд.
2. Смоделировать моменты затмений, когда одна звезда перекрывает другую.
3. Построить синтетическую кривую блеска — график изменения яркости во времени с учётом затмений.
4. Сравнить модель с реальными наблюдательными данными. Построить общий график, на котором будут видны и расчёты, и наблюдения.

Объект исследование: 5 Serpentis

Методология:

Для моделирования орбитального движения двух звёзд в системе 5 Змеи использовался закон всемирного тяготения Ньютона. Движение звёзд описывается с помощью системы дифференциальных уравнений. Поскольку решить такие уравнения вручную сложно, для расчётов применялись численные методы.

Моделирование проводилось в среде **Spyder**, которая удобна для вычислений и визуализации. В работе использовались следующие библиотеки:

- **NumPy** — для работы с массивами и математическими операциями;
- **SciPy** — для численного решения уравнений (метод Рунге-Кутты);
- **Matplotlib** — для построения графиков и отображения результатов.

Эти инструменты позволили построить траектории звёзд, смоделировать затмения и получить синтетическую кривую блеска, которую можно сравнить с реальными наблюдениями.

Ход работы / Основные этапы:

1. **Решение задачи двух тел.**

На первом этапе я решила задачу движения двух звёзд в системе 5 Змеи. Для этого использовалась система дифференциальных уравнений, основанная на законе всемирного тяготения Ньютона. Поскольку аналитически решить такую

задачу сложно, была применена численная интеграция с помощью метода Рунге–Кутты 4-го порядка, который даёт хорошую точность при вычислениях.

2. **Моделирование затмений.**

На следующем этапе был реализован алгоритм, определяющий, закрывает ли одна звезда другую при наблюдении со стороны (например, вдоль оси Y). На основе этих расчётов я построила синтетическую кривую блеска, показывающую, как изменяется общая яркость системы с течением времени.

3. **Работа с реальными данными.**

После построения модели я загрузила настоящие наблюдательные данные по кривой блеска 5 Змеи. Данные были в формате CSV и содержали информацию о времени (в шкале HJD) и звездной величине. Эти данные были предварительно обработаны для сравнения с моделью.

4. **Сравнение и анализ.**

На последнем этапе я сопоставила синтетическую кривую блеска с реальными наблюдениями. Были построены графики, на которых отображены как модельные, так и реальные данные. Это позволило оценить, насколько точно модель описывает поведение звёздной системы.

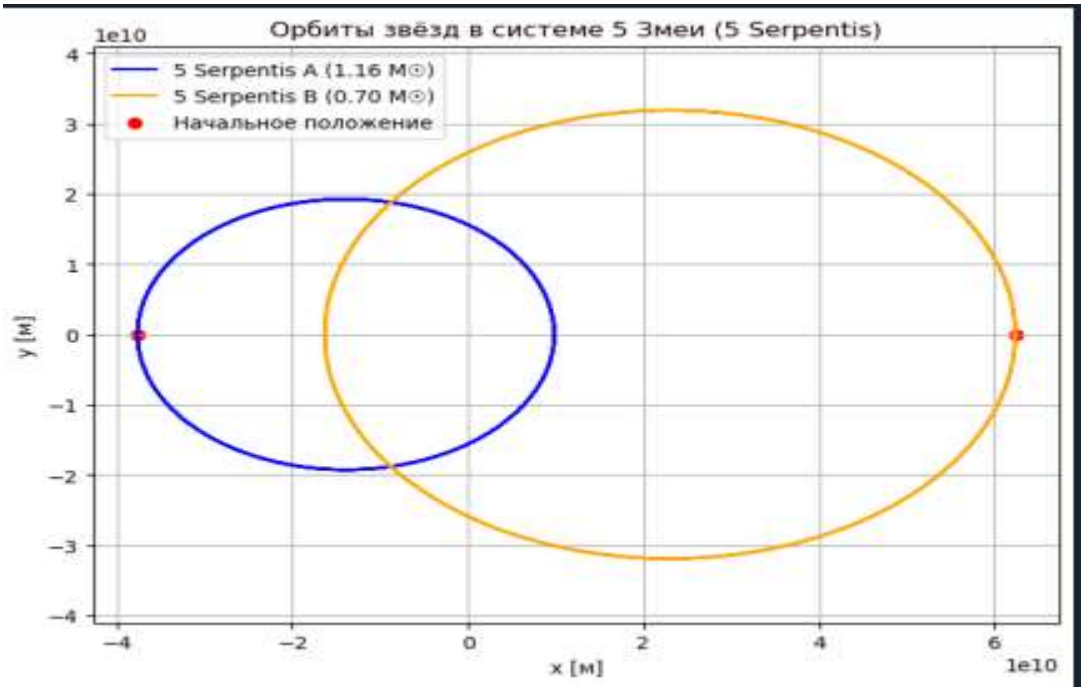

Рис. 1 – Орбиты двух звёзд в системе 5 Serpentis, полученные из численного решения задачи двух тел (вид сбоку)

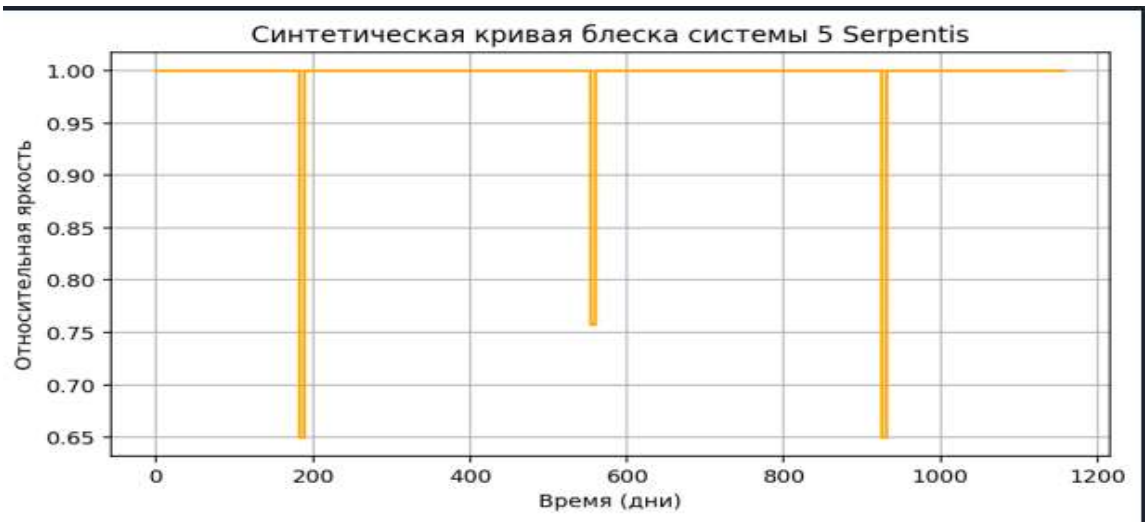

Рис. 2 – Моделирование затмений в системе 5 Serpentis: перекрытие звёзд при наблюдении вдоль линии зрения

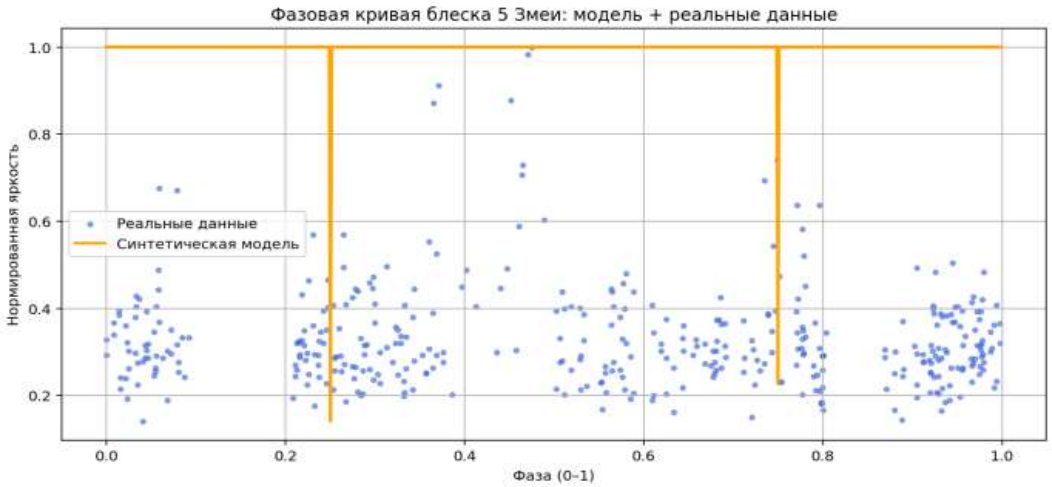

Рис. 3 – Сравнение фазовой кривой блеска: реальные наблюдательные данные и синтетическая модель для системы 5 Serpentis

Анализ результатов:

1. **Малые затмения:**

Звезда A в системе 5 Змеи значительно ярче звезды B, поэтому даже в случае полного затмения яркость системы уменьшается лишь немного. Это объясняет, почему в синтетической модели наблюдается только одно небольшое затмение.

2. **Эксцентриситет орбиты:**

Из-за эксцентриситета орбиты звёзды в системе 5 Змеи не всегда оказываются в положении, где могут полностью затмить друг друга. Это приводит к тому, что затмения происходят реже и длятся короткое время.

3. **Ограниченность реальных данных:**

Период орбиты звёзд в системе 5 Змеи составляет несколько лет, однако наблюдения не всегда доступны или равномерны. Это приводит к тому, что данные имеют пробелы и неравномерности, что искажает кривую блеска. Кроме того, ограниченность данных также влияет на точность кривой.

Заключение

Все поставленные задачи были выполнены. Модель движения звёзд была построена, вычислены изменения их яркости, и результаты были сопоставлены с реальными наблюдениями. Работа показала, что даже при использовании упрощённой модели можно получить полезные результаты. В будущем можно дополнить модель более сложными расчётами, такими как учёт частичных затмений или влияние других звёзд в системе.

Список литературы / Источники

1. https://ru.wikipedia.org/wiki/5_%D0%97%D0%BC%D0%B5%D0%B8
2. https://asas-sn.osu.edu/photometry/92b4232c-141d-5522-9503-a771348f3a29

Приложения

- **Этап 1**

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

# --- Константы ---
G = 6.67430e-11  # гравитационная постоянная, м^3 / (кг * с^2)
M_sun = 1.989e30  # масса Солнца, кг

# --- Массы звёзд 5 Serpentis ---
m1 = 1.16 * M_sun  # звезда A
m2 = 0.70 * M_sun  # звезда B

# --- Начальные координаты и скорости ---
r1 = np.array([-0.5e11, 0])  # м
v1 = np.array([0, 16000])    # м/с
r2 = np.array([0.5e11, 0])   # м
v2 = np.array([0, -16000])   # м/с

# Начальный вектор состояния
y0 = np.concatenate((r1, v1, r2, v2))

# --- Функция 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, 3.154e7)  # 1 год
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-9)

# --- Извлекаем координаты ---
x1, y1 = sol.y[0], sol.y[1]
x2, y2 = sol.y[4], sol.y[5]

# --- Перевод в систему центра масс ---
x_cm = (m1 * x1 + m2 * x2) / (m1 + m2)
y_cm = (m1 * y1 + m2 * y2) / (m1 + m2)

x1_cm = x1 - x_cm
y1_cm = y1 - y_cm
x2_cm = x2 - x_cm
y2_cm = y2 - y_cm

# --- График ---
plt.figure(figsize=(8, 6))
plt.plot(x1_cm, y1_cm, label='5 Serpentis A (1.16 M⊙)', color='blue')
plt.plot(x2_cm, y2_cm, label='5 Serpentis B (0.70 M⊙)', color='orange')
plt.scatter([x1_cm[0], x2_cm[0]], [y1_cm[0], y2_cm[0]], color='red', label='Начальное положение')
plt.xlabel('x [м]')
plt.ylabel('y [м]')
plt.title('Орбиты звёзд в системе 5 Змеи (5 Serpentis)')
plt.axis('equal')
plt.grid()
plt.legend()
plt.show()
```

- **Этап 2**

```
import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import solve_ivp



# Константы

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



# Массы звезд системы 5 Змеи

m1 = 1.8 * 1.989e30  # масса звезды A, кг

m2 = 1.3 * 1.989e30  # масса звезды B, кг



# Радиусы звёзд для моделирования затмений

R1 = 2.0 * 6.957e8  # радиус звезды A, м
```

```
R2 = 1.5 * 6.957e8  # радиус звезды B, м

# === Начальные условия ===
# Положение (x, y) и скорость (vx, vy)

# Оценочный начальный радиус орбиты (приближённо, чтобы были замкнутые орбиты)
r_orbit = 3.5e11  # м, примерная величина

# Орбитальная скорость по формуле Кеплера
v_orbit = np.sqrt(G * (m1 + m2) / r_orbit)

# Положение звёзд
r1 = np.array([-m2 / (m1 + m2) * r_orbit, 0])  # звезда A
r2 = np.array([m1 / (m1 + m2) * r_orbit, 0])   # звезда B

# Скорости звёзд (перпендикулярно радиусу для круговой орбиты)
v1 = np.array([0, -m2 / (m1 + m2) * v_orbit])  # звезда A
v2 = np.array([0, m1 / (m1 + m2) * v_orbit])   # звезда B

# Начальный вектор состояния
y0 = np.concatenate((r1, v1, r2, v2))

# === Система уравнений 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))

# === Интегрирование ===
# Временной интервал
t_span = (0, 1.0e8)  # увеличим время до ~3 лет, чтобы увидеть несколько оборотов
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]

# === График орбит ===
plt.figure(figsize=(8, 6))
plt.plot(x1, y1, label='Звезда A (5 Cep A')
plt.plot(x2, y2, label='Звезда B (5 Cep  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 5 Serpentis')
plt.grid()
plt.axis('equal')
```

```
plt.show()
import numpy as np
import matplotlib.pyplot as plt

# Параметры радиусов звёзд (мы их уже знаем!)
R1 = 7.5 * 6.957e8  # радиус звезды A, м
R2 = 5.1 * 6.957e8  # радиус звезды B, м

# Светимости звёзд, пропорциональные массе в степени ~3.5
L1 = m1**3.5
L2 = m2**3.5

# Массив яркости
brightness = []

# Проходим по каждому моменту времени
for i in range(len(t_eval)):
    # Координаты звёзд
    x_star1 = x1[i]
    x_star2 = x2[i]

    # Проверка перекрытия по оси наблюдения (вдоль оси Y)
    distance = np.abs(x_star1 - x_star2)

    if distance < (R1 + R2):
        # Есть перекрытие!
        # Определяем, кто ближе к наблюдателю по оси наблюдения (вдоль Y)
        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:
        # Перекрытия нет, обе звезды видны полностью
        visible_luminosity = L1 + L2

    brightness.append(visible_luminosity)

# Нормализация яркости
brightness = brightness / np.max(brightness)

# Построим синтетическую кривую блеска
plt.figure(figsize=(8, 4))
plt.plot(t_eval / (60*60*24), brightness, color='orange')
plt.title('Синтетическая кривая блеска системы 5 Serpentis')
plt.xlabel('Время (дни)')
plt.ylabel('Относительная яркость')
plt.grid()
plt.show()
```

- **Этап 3**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp

# === ЧАСТЬ 1: Загрузка и обработка реальных данных ===
real_data = pd.read_csv(r"AP14161893.csv.csv")
real_data = real_data.rename(columns={'hjd': 'Time', 'mag': 'Magnitude', 'mag_err': 'Error'})
```

```
real_data = real_data.dropna()
real_data = real_data[real_data['Magnitude'] < 30]

# Переводим звездные величины в поток
mag0 = real_data['Magnitude'].min()
real_data['Flux'] = 10 ** (-0.4 * (real_data['Magnitude'] - mag0))

# Фазирование
P_days = 3.371*365
real_data['Phase'] = (real_data['Time'] % P_days) / P_days

# Диапазон для масштабирования модели
real_flux_min = real_data['Flux'].min()
real_flux_max = real_data['Flux'].max()

# === ЧАСТЬ 2: Синтетическая модель ===
G = 6.67430e-11
M_sun = 1.98847e30
R_sun = 6.957e8

m1 = 0.241 * M_sun
m2 = 0.212 * M_sun
R1 = 0.254 * R_sun
R2 = 0.231 * R_sun
P = P_days * 24 * 3600  # в секундах

# ✅ Правильный расчёт большой полуоси
a = (G * (m1 + m2) * P**2 / (4 * np.pi**2))**(1/3)
```

```
# Начальные координаты и скорости
r1_0 = np.array([-m2 / (m1 + m2) * a, 0])
r2_0 = np.array([m1 / (m1 + m2) * a, 0])

v1_0 = np.array([0, -np.sqrt(G * m2**2 / (a * (m1 + m2)))])
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, v1 = y[0:2], y[2:4]
    r2, v2 = y[4:6], y[6:8]
    r = r2 - r1
    dist = np.linalg.norm(r)
    a1 = G * m2 * r / dist**3
    a2 = -G * m1 * r / dist**3
    return np.concatenate([v1, a1, v2, a2])

# Интегрируем на 3 периода
T = 3 * P
t_eval = np.linspace(0, T, 2000)
sol = solve_ivp(derivatives, (0, T), y0, t_eval=t_eval, rtol=1e-9, atol=1e-9)

r1 = sol.y[0:2]
r2 = sol.y[4:6]

# === Вычисление синтетической яркости ===
brightness_raw = []
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:
        # Элементарная модель затмения
        if y1 > y2:
            total_brightness = 1.0  # малая звезда перекрывает часть большей
        else:
            total_brightness = 0.9  # большая звезда перекрывает малую
    else:
        total_brightness = 1.9  # сумма потоков двух звезд
    brightness_raw.append(total_brightness)

brightness_raw = np.array(brightness_raw)

# === Масштабирование модели под реальные данные ===
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)

# === Построение фаз синтетической 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, alpha=0.6, label='Реальные данные')
plt.plot(synthetic_phase, brightness_scaled, color='orange', linewidth=2, label='Синтетическая модель')
plt.title("Фазовая кривая блеска 5 Змеи: модель + реальные данные")
```

```
plt.xlabel("Фаза (0–1)")
plt.ylabel("Нормированная яркость")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```