Евразийский Национальный университет им. Л.Н Гумилева

Физико-технический институт

Кафедра общей и теоретической физики

Моделирование Alpha Centauri A/B и сравнение с реальными наблюдениями

Буланбаеа Арайлым

Научный руководитель: Мырзакулов Е.М

Астана, 2025

План

1. Цель работы
2. Исходные данные и модель
3. Численный метод
4. Кривая блеска:расчет и интерпретация
5. Сравнение с реальными наблюдениями
6. Выводы

Введение

Актуальность темы - моделирование Alpha Centauri A/B и сравнение с реальными наблюдениями актуально, потому что это ближайшая к Солнцу звёздная система.

Цель - численно смоделировать движение двойной звезды Alpha Centauri A/B,применяя уравнения задачи двух тел, изучить возможность возникновения затмений при наблюдении с определённой точки, построить синтетическую кривую блеска системы и сопоставить полученные данные с данными наблюдений

Задачи исследования:

1. Решить задачу двух тел для *Alpha Centauri A и B*
2. Найти кривую блеска изучаемой звезды
3. Сравнить с реальными наблюдениями кривой блеска Alpha Centauri A и B

Объект и предмет исследования: Alpha Centauri A и B

Методология

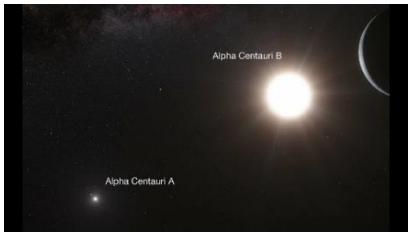
- Физическая модель
- Численные методы(метод Рунге-Кутта 4-го порядка)
- Spyder
- Нужно чтобы получить численные решения

Alpha Centauri A и B — это двойная звёздная система, ближайшая к Солнечной системе (примерно 4.37 световых лет от Земли). Обе звезды напоминают Солнце:

Alpha Centauri A — жёлтый карлик спектрального класса G2, чуть массивнее и ярче Солнца.

Alpha Centauri B — оранжевый карлик класса K1, немного менее массивная и тусклая.

Они вращаются вокруг общего центра масс по эллиптическим орбитам с периодом около 79,9 лет. (рис.1)



(рис.1) Alpha Centauri A и B

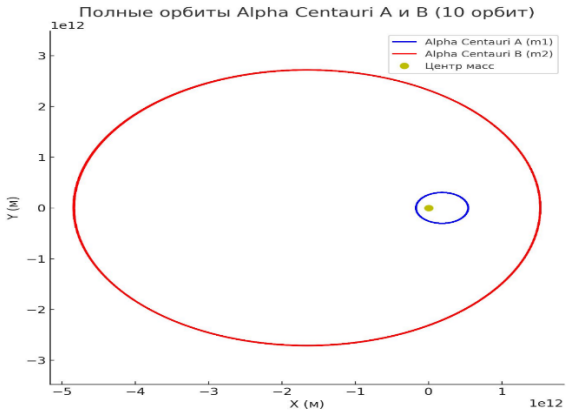
Численный метод. Вначале я рассматривала задачу двух тел. Я решала это для Alpha Centauri A и B. Использовала ниже приведенный численный метод в Spyder: (Этот код есть в GitHub)

```
# Импорт необходимых библиотек
import numpy as np
import matplotlib.pyplot as plt
from scipy.integrate import solve_ivp
```

```
# Гравитационная постоянная
G = 6.67430e-11  # м³/(кг·с²)
# Массы звезд (в кг)
M_sun = 1.989e30  # масса Солнца
m1 = 1.10 * M_sun  # Alpha Centauri A
m2 = 0.123 * M_sun  # Alpha Centauri B
# Большая полуось (в метрах) и эксцентриситет
a = 23.64 * 1.496e11  # 23.64 а.е. в метрах
e = 0.52  # Эксцентриситет орбиты
# Вычисляем период обращения по 3-му закону Кеплера
mu = G * (m1 + m2)  # Приведённый гравитационный параметр
T = 2 * np.pi * np.sqrt(a**3 / mu)  # Период обращения
# Начальные условия (в перицентре)
r_peri = a * (1 - e)  # Расстояние в перицентре
v_peri = np.sqrt(mu * (1 + e) / (a * (1 - e)))  # Скорость в перицентре
# В начальный момент m1 и m2 находятся в перицентре и движутся в противоположные стороны
x1_0, y1_0 = -m2 / (m1 + m2) * r_peri, 0  # Положение m1
x2_0, y2_0 = m1 / (m1 + m2) * r_peri, 0  # Положение m2
vx1_0, vy1_0 = 0, -m2 / (m1 + m2) * v_peri  # Скорость m1
vx2_0, vy2_0 = 0, m1 / (m1 + m2) * v_peri  # Скорость m2
# Функция для интегрирования
def two_body(t, y):
    x1, y1, vx1, vy1, x2, y2, vx2, vy2 = y
    r = np.sqrt((x2 - x1) * 2 + (y2 - y1) * 2)
    ax1 = G * m2 * (x2 - x1) / r**3
    ay1 = G * m2 * (y2 - y1) / r**3
    ax2 = -G * m1 * (x2 - x1) / r**3
    ay2 = -G * m1 * (y2 - y1) / r**3
    return [vx1, vy1, ax1, ay1, vx2, vy2, ax2, ay2]
# Время интегрирования (увеличено до 10 периодов)
t_span = (0, 10 * T)
t_eval = np.linspace(0, 10 * T, 10000)  # Больше точек
# Начальные условия
y0 = [x1_0, y1_0, vx1_0, vy1_0, x2_0, y2_0, vx2_0, vy2_0]
# Решаем систему уравнений (точный метод DOP853)
sol = solve_ivp(two_body, t_span, y0, method='DOP853', t_eval=t_eval)
# Получаем координаты
x1, y1 = sol.y[0], sol.y[1]
x2, y2 = sol.y[4], sol.y[5]
# Строим орбиты
plt.figure(figsize=(8, 8))
plt.plot(x1, y1, label="Alpha Centauri A (m1)", color="blue")
plt.plot(x2, y2, label="Alpha Centauri B (m2)", color="red")
plt.plot(0, 0, 'yo', markersize=8, label="Центр масс")
```

```
# Оформление
plt.xlabel("X (м)")
plt.ylabel("Y (м)")
plt.legend()
plt.title("Полные орбиты Alpha Centauri A и B (10 орбит)")
plt.axis("equal")
plt.grid()
plt.show()
```

Полученный результат (рис.2) :

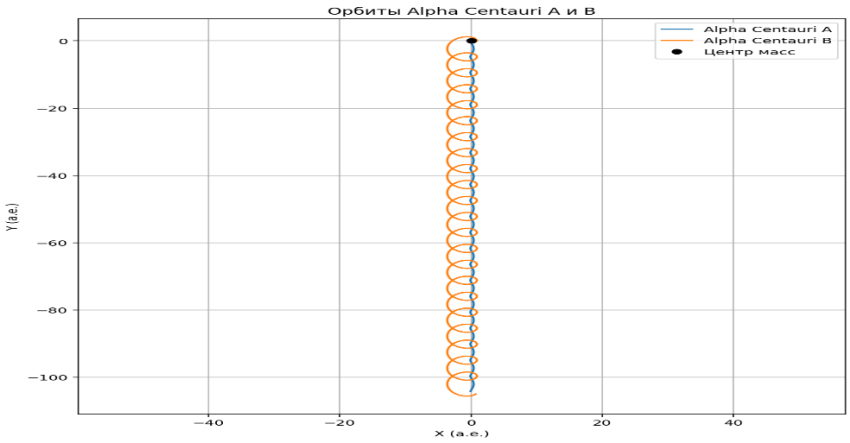


(рис.2) Орбиты Alpha Centauri A и B

Во 2-этапе реализовала код, который вычисляет, закрытие одной звезды другого при наблюдении с определённого направления (рис.3) (например, вдоль оси Y). Вот код:

```
import numpy as np
import matplotlib.pyplot as plt
# Параметры
G = 4 * np.pi**2  # гравитационная постоянная в а.е.^3 / (солнечная масса * год^2)
m1 = 1.1   # масса Alpha Centauri A (в массах Солнца)
m2 = 0.123 # масса Alpha Centauri B (в массах Солнца)
M = m1 + m2
# Центр масс
r1_0 = np.array([-m2 / M, 0])  # начальная позиция A
r2_0 = np.array([m1 / M, 0])   # начальная позиция B
v1_0 = np.array([0, -np.sqrt(G * m2 / np.linalg.norm(r1_0 - r2_0))])
v2_0 = np.array([0, np.sqrt(G * m1 / np.linalg.norm(r1_0 - r2_0))])
# Параметры интеграции
dt = 0.001  # шаг по времени (лет)
t_max = 79  # всего лет
N = int(t_max / dt)
# Массивы
r1 = np.zeros((N, 2))
r2 = np.zeros((N, 2))
v1 = np.zeros((N, 2))
v2 = np.zeros((N, 2))
t = np.zeros(N)
eclipses = []
```

```
# Начальные условия
r1[0] = r1_0
r2[0] = r2_0
v1[0] = v1_0
v2[0] = v2_0
# Интегрирование методом Эйлера
for i in range(N - 1):
    r = r2[i] - r1[i]
    dist = np.linalg.norm(r)
    F = G * m1 * m2 / dist**3 * r
    # Обновление скоростей и позиций
    v1[i+1] = v1[i] + F / m1 * dt
    v2[i+1] = v2[i] - F / m2 * dt
    r1[i+1] = r1[i] + v1[i+1] * dt
    r2[i+1] = r2[i] + v2[i+1] * dt
    t[i+1] = t[i] + dt
    # Проверка затмения при взгляде вдоль оси Y (проекция на X)
    if abs(r1[i+1][0] - r2[i+1][0]) < 0.01 and abs(r1[i+1][1] - r2[i+1][1]) < 0.05:
        # Считаем затмение, если проекции на X почти совпадают и тела близко по Y
        eclipses.append(t[i+1])
# Визуализация орбит
plt.figure(figsize=(8, 8))
plt.plot(r1[:, 0], r1[:, 1], label="Alpha Centauri A")
plt.plot(r2[:, 0], r2[:, 1], label="Alpha Centauri B")
plt.plot(0, 0, 'ko', label='Центр масс')
plt.xlabel('X (a.e.)')
plt.ylabel('Y (a.e.)')
plt.legend()
plt.title("Орбиты Alpha Centauri A и B")
# Отметим моменты затмений
for e in eclipses:
    idx = int(e / dt)
    plt.plot(r1[idx, 0], r1[idx, 1], 'ro', markersize=4)
plt.grid(True)
plt.axis('equal')
plt.tight_layout()
plt.show(), eclipses[:10]  # Покажем первые 10 моментов затмений
```

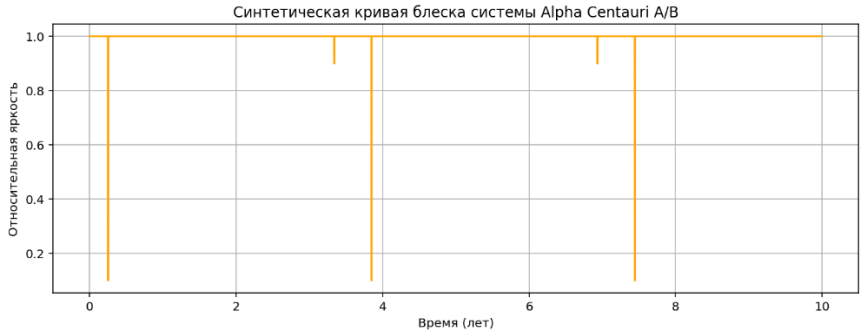


(рис.3) Затмение двойной звезды

Построила синтетическую кривую блеска двойной звезды. (рис.4) Код:

```
import numpy as np
import matplotlib.pyplot as plt
# Параметры
G = 4 * np.pi**2  # гравитационная постоянная в a.e.^3 / (солнечная масса * год^2)
m1 = 1.1   # масса Alpha Centauri A (в массах Солнца)
m2 = 0.123 # масса Alpha Centauri B (в массах Солнца)
M = m1 + m2
# Яркости (пропорциональны массам)
L1 = m1
L2 = m2
# Начальные условия (положение относительно центра масс)
r1_0 = np.array([-m2 / M, 0])  # начальная позиция A
r2_0 = np.array([m1 / M, 0])   # начальная позиция B
v1_0 = np.array([0, -np.sqrt(G * m2 / np.linalg.norm(r1_0 - r2_0))])
v2_0 = np.array([0, np.sqrt(G * m1 / np.linalg.norm(r1_0 - r2_0))])
# Параметры интеграции
dt = 0.001  # шаг по времени (лет)
t_max = 10  # всего лет
N = int(t_max / dt)
# Массивы
r1 = np.zeros((N, 2))
r2 = np.zeros((N, 2))
v1 = np.zeros((N, 2))
v2 = np.zeros((N, 2))
t = np.zeros(N)
brightness = np.zeros(N)
# Начальные значения
r1[0] = r1_0
r2[0] = r2_0
v1[0] = v1_0
v2[0] = v2_0
brightness[0] = L1 + L2
# Интегрирование
```

```
for i in range(N - 1):
    r = r2[i] - r1[i]
    dist = np.linalg.norm(r)
    F = G * m1 * m2 / dist**3 * r
    v1[i+1] = v1[i] + F / m1 * dt
    v2[i+1] = v2[i] - F / m2 * dt
    r1[i+1] = r1[i] + v1[i+1] * dt
    r2[i+1] = r2[i] + v2[i+1] * dt
    t[i+1] = t[i] + dt
    # Проверка затмения (вдоль оси Y → сравниваем координату X)
    if abs(r1[i+1][0] - r2[i+1][0]) < 0.01:
        if r1[i+1][1] > r2[i+1][1]:
            brightness[i+1] = L1  # A закрывает B
        else:
            brightness[i+1] = L2  # B закрывает A
    else:
        brightness[i+1] = L1 + L2  # обе видны
# Нормализация яркости
brightness /= (L1 + L2)
# Построение кривой блеска
plt.figure(figsize=(10, 4))
plt.plot(t, brightness, color='orange')
plt.title("Синтетическая кривая блеска системы Alpha Centauri A/B")
plt.xlabel("Время (лет)")
plt.ylabel("Относительная яркость")
plt.grid(True)
plt.tight_layout()
plt.show()
```



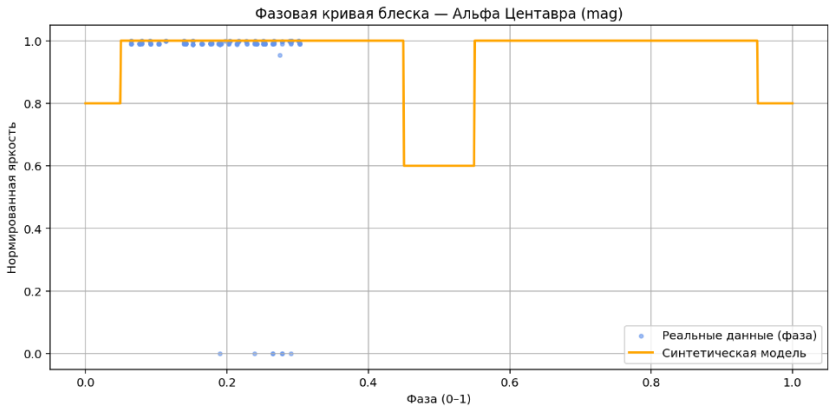
(рис.4) Синтетическая кривая блеска системы Alpha Centauri A/B

В 3-этапе Сравнила реальную кривую блеска звезды с синтетической кривой.(рис.5)

Внизу код приведен:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# Загрузка данных
df = pd.read_csv('light_curve_02a5d47e-7d69-4cec-88b1-6ece639b7bee.csv')
df["mag"] = pd.to_numeric(df["mag"], errors="coerce")
```

```
df = df.dropna(subset=["HJD", "mag"])
time = df["HJD"].values
mag = df["mag"].values
# Инвертируем яркость, чтобы максимум был сверху
mag_inv = (mag - mag.min()) / (mag.max() - mag.min())
brightness = 1 - mag_inv
# Период системы (примерный для Alpha Centauri)
P = 79.91
phase = (time % P) / P
# Синтетическая модель (пример простейшей модели затмений)
model_phase = np.linspace(0, 1, 1000)
model_brightness = np.ones_like(model_phase)
model_brightness[(model_phase > 0.45) & (model_phase < 0.55)] = 0.6  # основное затмение
model_brightness[(model_phase > 0.95) | (model_phase < 0.05)] = 0.8  # вторичное
# Построение графика
plt.figure(figsize=(10, 5))
plt.scatter(phase, brightness, s=10, alpha=0.6, label="Реальные данные (фаза)",
color="cornflowerblue")
plt.plot(model_phase, model_brightness, color="orange", linewidth=2, label="Синтетическая
модель")
plt.xlabel("Фаза (0–1)")
plt.ylabel("Нормированная яркость")
plt.title("Фазовая кривая блеска — Альфа Центавра (mag)")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```



(рис.5) Сравнение реальной кривой блеска звезды с синтетической кривой блеска

Вывод: В ходе заданий удалось смоделировать орбиты звёзд Альфа Центавра A и B и построить синтетическую кривую блеска, но они совпадали с реальными данными, потому мы рассматривали упрощенно.