Матвей Полубрюхов

m9175172036@gmail.com

Аннотация

[Заинтересуйте читателя с помощью аннотации (как правило, это краткое содержание документа).   
Если вы готовы добавить свой текст — просто щелкните здесь и введите его.]

Моделирование кривой блеска затменной звёздной системы

[Подзаголовок документа]

Оглавление

[Цели проекта 2](#_Toc108831117)

[Используемый «инструментарий» 2](#_Toc108831118)

[Алгоритм работы кода 2](#_Toc108831119)

[1. Вычисление истинной аномалии 2](#_Toc108831120)

# Цели проекта

* Изучить затменные звёздные системы, закономерность изменения звёздной величины, различия кривых блеска для разных типов звёздных систем
* Создать программу, выполняющую моделирование двойной звёздной системы
* Проанализировать полученные графические изображения изменения параметров во времени

# Используемый «инструментарий»

* Язык программирования Python 3.10. Выбор обусловлен лёгкостью и быстротой написания кода, понятностью синтаксиса, что я поставил приоритетней скорости работы кода.
* Интегрированная среда разработки JetBrains PyCharm Professional EAP. На данный момент это одна из лучших по функционалу IDE для выбранного ЯП.
* Распределённая система контроля версий Git for Windows. Самая известная и простая в изучении.
* Подключаемые библиотеки (список см. [Dependencies (README.md)](https://github.com/PM-95025/scientificProject/network/dependencies) и [requirements.txt](https://github.com/PM-95025/scientificProject/blob/master/requirements.txt)

# Алгоритм работы кода

1. Первоначальная настройка (инициализация)
2. Главный цикл (итерация по времени)
   1. Вычисление истинной аномалии
   2. Расчёт расстояния между центрами в видимой плоскости (обозначен за х на рис.2)
   3. Проверка дисков звёзд на наложение
   4. Подсчёт суммарной звёздной величины
   5. Добавление данных в массивы
3. Инициализация matplotlib
4. Построение графиков

### 1. Вычисление истинной аномалии

Производится путём численного решения уравнения Кеплера методом итераций:

def elliptical\_orbit(t: float, T: float, e: float) -> float:  
 M: float = t / T \* 2\*pi  
 E: float = M  
   
 for n in range(20):  
 E = e \* sin(E) + M  
   
 phi: float = 2 \* degrees(atan(tan(E/2) / sqrt((1-e) / (1+e))))  
   
 if t/T > 0.5:  
 phi += 360  
   
 return phi

### 2. Расчёт расстояния в проекции на картинную плоскость

* def apparent\_distance(system, fi: float) -> float:if system.e == 0:  
   return system.a \* abs(sin(radians(fi)))  
     
   return abs((system.p \* sin(radians(system.periapsis\_argument + fi))) / (1 + system.e \* cos(radians(fi))))
* def inclination\_correction(system, fi: float) -> float:return system.a \* sin(radians(system.inclination)) \* cos(radians(system.ascending\_node\_longitude)) \* cos(radians(fi))
* fi = time2phi(t, system.period, system.e)  
  ap = apparent\_distance(system, fi)  
  x: float = sqrt(ap\*\*2 + inclination\_correction(system, fi)\*\*2)

### 3. Проверка дисков звёзд на наложение

Производится с использование ранее вычисленного расстояния в проекции на картинную плоскость

