ГБОУ "Президентский ФМЛ № 239"

Моделирование эволюции планетной системы TRAPPIST – 1

Годовой проект по информатике

Автор: Клопова-Сапоровская Ирина, 10-2

Цель

Проанализировать эволюцию планетной системы TRAPPIST-1 с помощью метода численного моделирования.

Расшифровка цели

Численное моделирование на данный момент является важнейшим методом астрофизических исследований. Это в первую очередь вызвано тем, что астрономия все еще остается наблюдательной наукой, то есть в ней невозможен прямой эксперимент над исследуемым объектом. Но этот факт не мешает ученым строить динамические модели, основываясь на доступной информации, получаемой от объектов космоса. Модель построена, после чего необходимо доказать ее работоспособность. Как это сделать? Вот тут и приходит на помощь численное моделирование. Переведя свою модель в код и сверив результат расчёта с наблюдаемыми на разных стадиях эволюции объектами можно подтвердить (или же опровергнуть) свою теорию. Если результаты совпали с наблюдениями, то с какой-то точностью модель верна. Кроме того, если уже есть достоверная модель, то она обладает предсказательной силой, а значит по ней можно рассчитать, во что перейдет объект в результате своей эволюции с течением времени. Реализация этих идей является целью моего проекта.

TRAPPIST-1 - это планетная система, состоящая из главной звезды (красного карлика) и 7 планет, открытая 2 мая 2016 года. На данный момент известно большое количество планетных систем у звезд различных типов, однако для моделирования был выбрана эта система по следующим причинам: она крайне интересна для исследования, поскольку не совсем ясно, каким образом система из такого количества тел с учетом компактности их орбит смогла сохранить устойчивость (хотя бы за время ее наблюдения, т.е. примерно 2 года), благодаря ее относительной близости к Солнечной системе параметры орбит планет известны с хорошей точностью, что позволяет моделировать поведение системы на больших временных масштабах, параметры орбит и наклон картинной плоскости у системы TRAPPIST-1 очень удачны, потому как освобождает от необходимости дополнительных вычислений.

Система TRAPPIST-1 удобна еще и тем, что на ней можно продемонстрировать сразу две стороны численного моделирования: уже оговоренные ранее проверка модели и экстраполяция эволюции системы на большие промежутки времени.

Результатом проек та будет построение графика, на который нанесены орбиты планет, по которому можно будет сделать вывод об устойчивости системы, а также вывод о достоверности используемой модели.

Исходные и выходные данные

Исходные данные – это полученные из реальных наблюдений значения масс тел системы (double), их радиус-векторов (double), периодов (double) и начальных фаз (double).

Выходные данные – график, по оси абсцисс отложена x – координата в астрономических единицах, по оси ординат y – координата в астрономических единицах. Начало координат совпадает с центром масс системы. Еще одним результатом будет график, характеризующий изменения суммарной энергии системы в процессе моделирования.

Математическая модель

Основной формулой в моделировании эволюции системы является закон всемирного тяготения:

$$\vec{F}_{12} = \frac{G * m * M}{r_{21}^3} * \overrightarrow{r_{21}}$$

F – сила, с которой первое тело действует на второе

G – гравитационная постоянная

M – масса первого тела

m – масса второго тела

r – радиус-вектор от второго тела к первому

Используется формула для вычисления скорости при равноускоренном движении:

$$\vec{V} = \overrightarrow{V_0} + \vec{a} * t$$

V_0 – вектор начальной скорости

V – вектор скорости через время t

a – ускорение

t – время

Формула для вычисления перемещения:

$$\vec{r} = \overrightarrow{r_0} + \vec{V} * t$$

r – радиус-вектор через время t

r_0 – начальный радиус-вектор

Формула для проекции на оси абсцисс и ординат в декартовой системе координат:

$$r_x = r * \cos\varphi$$

$$r_y = r * \sin\varphi$$

r – полное перемещение

r_x – перемещение по x

r_y – перемещение по y

φ – угол между вектором перемещения и осью x

Формула расчета линейной скорости при движении по кругу через угловую скорость и радиус-вектор точки:

$$V = \frac{2\pi}{P} * R$$

R – радиус-вектор из барицентра в тело

P – период тела

Формула потенциальной энергии тела в гравитационном поле другого тела:

$$E_{\text{п}} = -\frac{G * m * M}{2 * r}$$

$E_{\text{п}}$ - значение потенциальной энергии

Формула кинетической энергии движения тела:

$$E_{\text{к}} = \frac{m * V^2}{2}$$

$E_{\text{к}}$ – значение кинетической энергии

Анализ используемой структуры данных

Все входные данные хранятся в массивах. Данные хранить необходимо, поскольку они используются на протяжении всего решения. Рассчитываемые величины скоростей и координат также необходимо хранить в массивах (уже двумерных, поскольку рассчитываются координаты и скорости для каждого из тел системы на каждом шаге), поскольку на основании первого рассчитывается второе, а на основании координат строится график.

Кроме того, в процессе решения идет заполнение массива суммарной энергии системы на каждом шаге, по которому в конечном итоге тоже будет построен график.

Типы данных во всех массивах – double, поскольку необходимо рассчитывать модель с как можно большей точностью.

Алгоритм решения задачи

Задача – построение графика, на котором отражены траектории тел системы за заданный промежуток времени. График строится по точкам, которые рассчитывает программа. Количество точек задается количеством шагов, каждый из которых отделен от последующего промежутком времени dt, задаваемым в программе. Рассмотрим метод расчета одной точки.

Точка характеризует положение тела на своей орбите в заданный момент времени. Эта точка определяется начальными данными, а также гравитационным взаимодействием между телом и остальными 7 телами данной системы. Чтобы рассчитать положение тела на следующем шаге, делается предположение, по которому ускорение тела, которое равно векторной сумме ускорений, сообщаемых телу остальными семью телами системы, остается постоянным на промежутке времени, равном dt. С таким предположением можно достичь хорошей точности, если брать небольшие значения dt (много меньше значений

орбитальных периодов тел). Это ускорение раскладывается на две составляющие (по количеству осей в ДСК), и с их помощью рассчитывается скорость на момент следующего шага по формуле для равноускоренного движения. Далее для этого же шага делается еще одно приближение, по которому уже скорость тела во время шага считается за константу. Таким образом, зная x- и y-компоненты скоростей, рассчитываются координаты тела для следующего шага. Данный метод необходимо структурировать, чтобы иметь возможность написать для него программу.

Пошаговый алгоритм:

1) В массивы записываются начальные данные о радиус-векторах, массах, фазах и периодах. Для точности выбирается подходящая для данных значений параметров система единиц, и все константы переводятся в эту систему. В случае TRAPPIST-1 удобно измерять расстояния в астрономических единицах, массы в массах Земли, а время в сутках. В эту систему переведена гравитационная постоянная из закона всемирного тяготения.
2) Двумерные массы скоростей и координат заполняются нулями. Одномерный массив энергий заполняется нулями. Переменная, характеризующая начальную энергию системы, в которую на одном из следующих шагов будет записана рассчитанная начальная энергия, инициализируется нулем.
3) В цикле по всем телам рассчитываются начальные значения скоростей и координат на основании фаз, радиус-векторов и периодов, в предположении круговых орбит этих тел. Нулевое тело, звезду, можно условно считать неподвижной, поскольку ее масса во много раз превосходит суммарную массу остальных тел системы.
4) Рассчитываем в цикле по телам суммарную энергию системы в нулевой момент времени. Энергия состоит из суммы кинетических энергий всех тел в нулевой момент времени, а также потенциальной энергии их попарных взаимодействий, которая рассчитывается в еще одном цикле по телам, причем, что естественно, влияние тела самого на себя отсутствует, а значит, не включается в расчет.
5) Далее идет основная часть программы. Ее удобнее представить в виде схемы:
   Цикл по шагам(шаг n):
   Цикл по телам (тело i):
   Обнуление ускорений
   Цикл по телам (тело k):
   Расчет суммарных ускорений по x и y для тела i, возникающих в результате воздействия а него k тел

   Расчет координат и скоростей на n+1 шаг

   Цикл по телам(тело i):

   Суммирование кинетических энергий тел

   Суммирование потенциальных в цикле и прибавление к ним

   кинетических.

Важно отметить, что данный этап программы может занять некоторый ощутимый промежуток времени, зависящий от длины шага, количества шагов и мощности компьютера.

6) Осталось вывести результаты. Подписываются графики, оси, после чего выводятся сами графики.

Листинг

```
import numpy as np
import matplotlib.pyplot as plt
import math
```
Подключение библиотек. Math для математических функций (например, квадратного корня), matplotlib.pyplot для построения графиков, а numpy для работы с массивами и для математических функций.
```
deg2rad = np.pi/180.0
```
Инициализация переменной, равной коэффициенту перевода из градусной меры углов в радианную.
```
N_bodies = 8
dt = 0.001
N_steps = 9000
```
Инициализация количества тел, количества шагов и промежутка времени между шагами.
```
G = 8.89042e-10
```
Гравитационная постоянная, пересчитанная в нужную систему единиц (а.е.^3/масса_Земли/сутки^2).
```
m = [29600, 1.02, 1.16, 0.297, 0.772, 0.934, 1.148, 0.331]
R = [0.0, 0.01150,  0.01576, 0.02219, 0.02916, 0.03836, 0.0467, 0.0617]
P = [0.0, 1.51087637, 2.42180746, 4.049959, 6.099043, 9.205585, 12.354473, 18.767953]
phase = [0.0, 0.000, 217.470, 300.378, 142.558, 323.471, 269.932, 42.487]
```
Задание начальных параметров системе в выбранной системе единиц.
```
x = np.tile(0.0, (N_bodies, N_steps))
y = np.tile(0.0, (N_bodies, N_steps))
vx = np.tile(0.0, (N_bodies, N_steps))
vy = np.tile(0.0, (N_bodies, N_steps))
E = np.tile(0.0, N_steps - 1)
E0 = 0.0
```
Инициализация массивов координат, скоростей и энергий, в которые в процессе работы программы будут записываться рассчитываемые данные.
```
for i in range(1, N_bodies):
    x[i, 0] = R[i]*np.cos(phase[i]*deg2rad)
    y[i, 0] = R[i]*np.sin(phase[i]*deg2rad)
    vx[i, 0] = -(2.0*np.pi*R[i]/P[i])*np.sin(phase[i]*deg2rad)
    vy[i, 0] = (2.0*np.pi*R[i]/P[i])*np.cos(phase[i]*deg2rad)
for i in range(0, N_bodies):
    E0 =E0+ m[i]*(vx[i,0]**2+vy[i,0]**2)/2.0
    for j in range(0, N_bodies):
        if j == i: continue
        E0=E0-G*m[j]*m[i]/(2.0*math.sqrt((x[j,0]-x[i,0])**2+(y[i,0]-y[j,0])**2))
```
Расчет начальных координат и скоростей тел и начальной энергии системы в целом.
```
for n in range (0, N_steps - 1): #запуск цикла по шагам
    for i in range (0, N_bodies):  #запуск цикла для расчета координат и скоростей на следующем шаге для тела i
        dvx = 0.0#обнуление ускорения по оси x
```

```
        dvy = 0.0#обнуление ускорения по оси y
        for k in range(0, N_bodies):#запуск цикла, которые рассчитывает ускорения, вызываемые влиянием на тело i другими телами системы
            if i == k: continue #тело само на себя не действует
            dx = x[i, n] - x[k, n] #разность x-координат тела i и тела k
            dy = y[i, n] - y[k, n] # разность y-координат тела i и тела k
            dr = (dx*dx + dy*dy)**0.5 #расчет расстояния между телами k и i
            dvx = dvx - G*m[k]/dr**2*dx/dr #прибавление приращения ускорения от тела k по оси x к суммарному возмущающему ускорению по этой же оси, вычисленному в результате прохождения по тому же циклу по телам от 0 до k-1
            dvy = dvy - G*m[k]/dr**2*dy/dr # прибавление приращения ускорения от тела k по оси y к суммарному возмущающему ускорению по этой же оси, вычисленному в результате прохождения по тому же циклу по телам от 0 до k-1

        x[i, n + 1] = x[i, n] + vx[i, n]*dt  #расчет координаты по оси x на шаге n+1
        y[i, n + 1] = y[i, n] + vy[i, n]*dt    #расчет координаты по оси y на шаге n+1
        vx[i, n + 1] = vx[i, n] + dvx*dt  #расчет скорости по x на шаге n+1
        vy[i, n + 1] = vy[i, n] + dvy*dt # расчет скорости по y на шаге n+1

    for i in range(0, N_bodies):#запуск цикла по телам
        E[n] =E[n]+ m[i]*(vx[i,n]**2+vy[i,n]**2)/2.0/E0 #в каждом цикле прибавляется кинетическая энергия тела номера i к посчитанной ранее суммарной энергии
        for j in range(0, N_bodies): #в цикле по телам идет суммирование потенциальных энергий взаимодействия тела номера i и тела номера j
            if j == i: continue #тело само с собой не взаимодействует
            E[n]=E[n]-G*m[j]*m[i]/(2.0*math.sqrt((x[j,n]-x[i,n])**2+(y[i,n]-y[j,n])**2)) /E0 #расчет и суммирование
```

Заполнение двумерных массивов скоростей и координат тел, а также одномерного массива энергий системе в результаты пошагового просчета каждой величины.

```
plt.xlabel('AU')
plt.ylabel('AU')
plt.title('TRAPPIST-1')
plt.plot(x[0,:], y[0,:],'.')
for i in range(1, N_bodies):
    plt.plot(x[i,:], y[i,:])
plt.show()
```

Построение траекторий тел на графике.

```
plt.xlabel('Steps')
plt.ylabel('E/E0')
plt.title('Energy')
plt.plot(E)
plt.show()
```

Построение графика зависимости отнормированной энергии системы в зависимости от номера шага.

Пример работы программы

Входные данные (полученные в результате наблюдений):

Данные о планетах:

	Масса (Массы Земли)	Большая полуось (а.е.)	Период (сутки)	Орбитальная фаза (градусы)

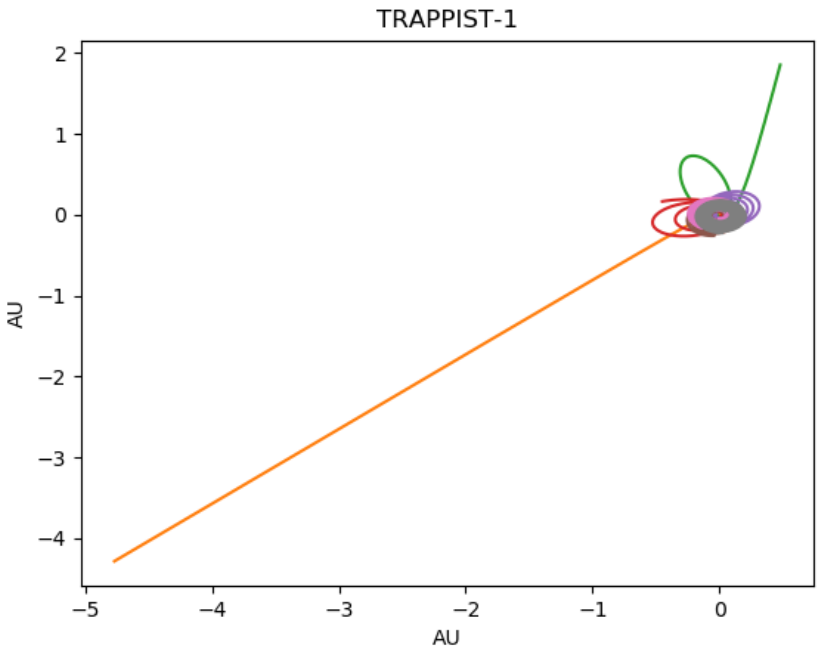
1	1.02	0.01150	1.51087637	0.000
2	1.16	0.01576	2.42180746	217.470
3	0.297	0.02219	4.049959	300.378
4	0.772	0.02916	6.099043	142.558
5	0.934	0.03836	9.205585	323.471
6	1.148	0.0467	12.354473	269.932
7	0.331	0.0617	18.767953	42.487

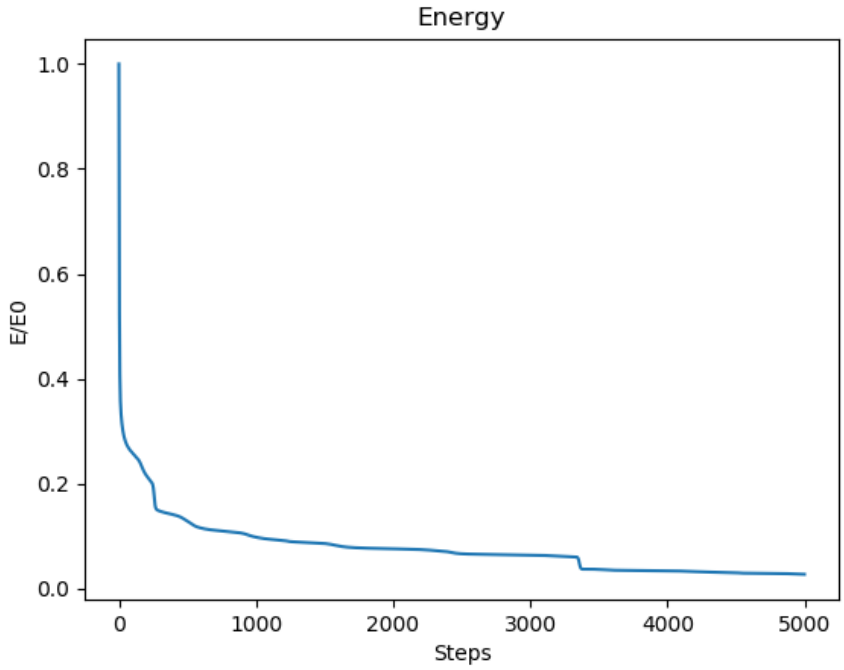
Данные о главной звезде:

Масса (Массы Земли)
29600

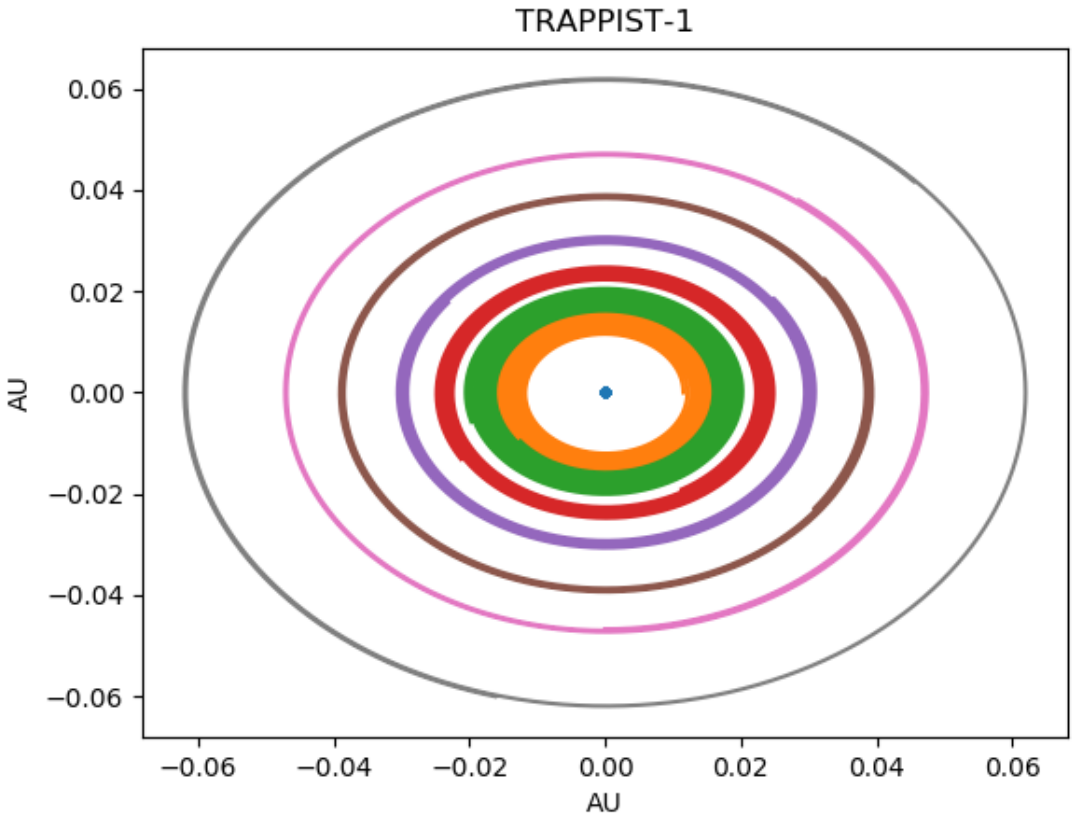
Особенность численного моделирования состоит в том, что его результат сильно зависит от выбранной величины шага и количества шагов. Ниже приведены результаты выполнения программы для разных величин этих параметров.

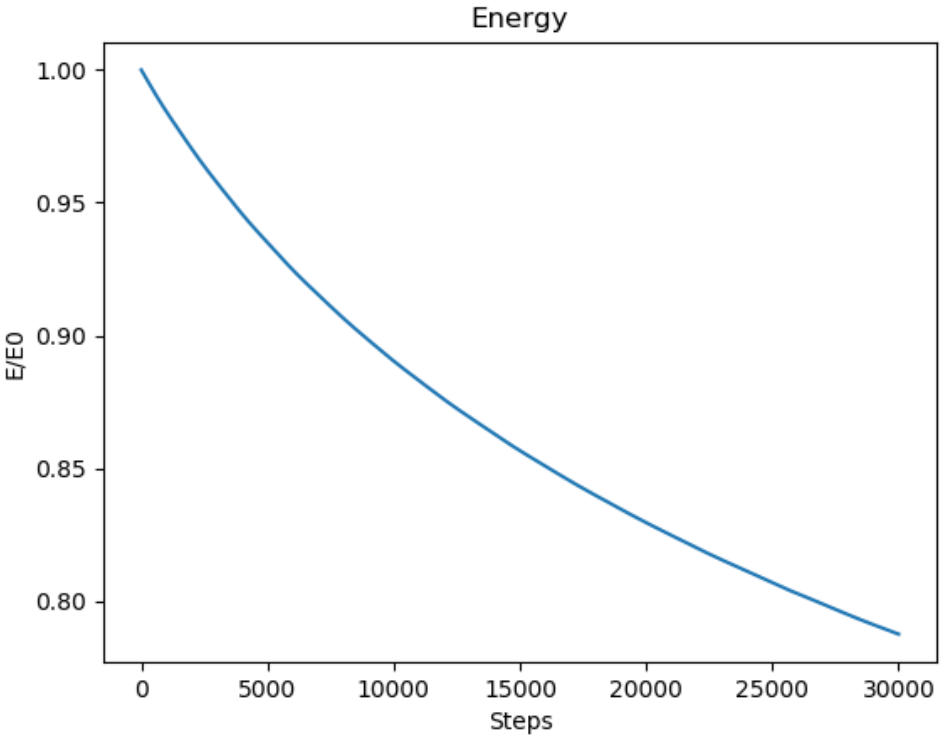
Большой шаг:





А вот что будет, если взять маленький шаг и достаточное для большого количества оборотов системы:





Анализ правильности решения

Во-первых, очевидно, что результат моделирования очень зависит от точности введенных данных, величины шага и промежутка между шагами, поэтому не совсем корректно называть результат правильным или неправильным. Тем более, что сверить результат практически не с чем, поскольку систему наблюдают пока только 2 года, а следовательно, о ее стабильности на больших временных масштабах по экспериментальным данным судить не приходится. Но все-таки существует параметр, опираясь на который, можно говорить о жизнеспособности той или иной модели. Это энергия. В подобной замкнутой системе энергия должна сохраняться, поэтому чем лучше для расчета работает закон сохранения энергии, тем вероятнее, что система будет эволюционировать именно таким образом. Как видно из результатов, модель, по которой система остается относительно стабильной, имеет хороший график энергии, то есть от начальной энергии теряется только 20%. Из этого можно сделать вывод, что система скорее всего стабильна, а самая близкая к действительному положению вещей модель – это модель с практически замкнутыми траекториями.

Во-вторых, из эксперимента известно, что система стабильна на промежутках времени минимум в два года. То, что при моделировании получилась похожая картина говорит о том, что выбранная модель выполняется с хорошей точностью.