Исследование движения двух звёзд

Евгений Никитин

Июнь 2023

1 Введение

В своём прошлом исследовании о благоприятных условиях для возникновения жизни на экзопланетах, я подробно рассматривал лишь внутренние климотообразующие факторы (связанные конкретно с экзопланетой). При этом соблюдалось строгое ограничение: только одна звезда в системе. Естественно, что, изучая подобные общие явления, как климат экзопланет, нельзя обойти стороной случай, когда в системе находится более одного светила. Для продолжения исследования необходимо понять принцип движения двух звёзд в системе и выяснить, как случай с двумя движущимися источниками тепла может влиять на колебания климата планеты. Для этого необходимо спроектировать компьютерную модель движения двух звёзд.

Цель работы: Изучить механизм движения двух тел в гравитационном поле и построить динамичную компьютерную модель этой системы.

Задачи:

- Сформулировать задачу, изобразить её условия графически.
- Получить выражения необходимые для вычисления постоянных величин системы, используя механику Лагранжа.
- Определить функциональную зависимость координат системы от времени.
- На основе полученных выражений и зависимостей построить модель, демонстрирующую механизм движения двух тел под действием гравитационной силы.

2 Теоретическая часть

2.1 Формулировка задачи

Для начала сформулируем задачу математически. Рассмотрим трёхмерную систему координат (x; y; z), и две материальные точки с массами m_1 и m_2. Проведём к каждому из тел радиусы-векторы $\vec{r_1}$ и $\vec{r_2}$ соответственно, а также вектор смещения $\vec{r}$, направленный от m_1 к m_2 (рис. 1) и вектор $\vec{R}$ – радиус-вектор центра массы. Векторы $\vec{r}$ и $\vec{R}$ определяются выражениями:

$$\vec{r} = \vec{r_1} - \vec{r_2}$$
$$\vec{R} = \frac{m_1\vec{r_1} + m_2\vec{r_2}}{m_1 + m_2}$$

Решая систему относительно $\vec{r_1}$ и $\vec{r_2}$, получаем уравнения для двух векторов:

$$\vec{r_1} = \vec{R} + \frac{m_2}{m_1 + m_2}\vec{r}$$
$$\vec{r_2} = \vec{R} - \frac{m_1}{m_1 + m_2}\vec{r}$$

Удобно расположить начало координат в центре инерции системы, записав уравнения радиус-векторов в виде:

$$\vec{r_1} = \frac{m_2}{m_1 + m_2}\vec{r}$$
$$\vec{r_2} = -\frac{m_1}{m_1 + m_2}\vec{r}$$

Движение двух тел относительно друг друга будет происходить в одной плоскости, *перпендикулярной вектору момента импульса системы* $\vec{M}$ [2, с. 90], это будет использовано в дальнейшем. Следующая задача: определить положение каждого тела в любой момент времени.

2.2 Функция Лагранжа системы

Теоретическая механика описывает гравитационное движение двух тел, *используя механику Лагранжа*, введённую в 18 веке Луи Лагранжем. Функция Лагранжа системы двух тел определяет её состояние в момент времени t [1, с. 46]:

$$L\left(r\right) = \frac{m\dot{\vec{r}}^2}{2} - \frac{\alpha}{|\vec{r}|} \tag{1}$$

Учитывая, что α - некий коэффициент потенциальной энергии и $\alpha < 0$.

2.3 Сохраняющиеся величины

Постоянные или сохраняющиеся величины системы – параметры, не изменяющиеся со временем.

Обобщенная энергия системы определяется в теоретической механике как [1, с. 24]:

$$E = \sum_{i=1}^{S} \frac{\partial L}{\partial \dot{q}_i} \dot{q}_i - L$$

Полная энергия сохраняется, так как функция Лагранжа (а значит состояние системы) не зависит явно от времени. Учитывая, что S - число материальных точек системы, и $\dot{q}_1 \equiv \ \dot{r}_1(t_0)$ и $\dot{q}_2 \equiv \ \dot{r}_2(t_0)$, получаем:

$$E = \frac{m_1 \dot{r}_1 + m_2 \dot{r}_2}{2} + U(r_0)$$

Обобщенный момент импульса системы (учитывая, что $\dot{\vec{R}} = 0$) [1, с. 30]:

$$M = \sum_{i=1}^{N} [\vec{r}_i \times \vec{p}_i]$$

Сохранение момента импульса связано с тем, что система обладает *изотропией пространства.*

2.4 Функциональная зависимость координат от времени

В разделе 2.1 упоминалось, что движение двух тел происходит в одной плоскости, перпендикулярной вектору момента импульса. В этой плоскости вводятся полярные координаты: длина вектора смещения r и угол поворота φ. Их изменение со временем характеризуют следующие интегралы [1, с. 47]:

Зависимость φ от r для движения в центральном поле:

Зависимость угла поворота φ от изменения вектора r определяет траекторию движения тела и даётся следующим выражением:

$$\varphi = \int_{r_0}^{r} \frac{\frac{M}{r^2}dr}{\sqrt{2mE + 2m\alpha\frac{1}{r} - M^2\frac{1}{r^2}}}$$

Введя замену $a = 2mE$; $b = 2m\alpha$; $c = M$; и выделив полный квадрат:

$$\varphi = \int_{r_0}^{r} \frac{\frac{M}{r^2}dr}{\sqrt{(-\left(\frac{c}{r} - \frac{b}{2c}\right)^2 + \frac{b^2}{4c} - a)}}$$

Выполним простое интегрирование:

$$\varphi = arccos\frac{\frac{p}{r} - 1}{e}$$

Перепишем зависимость как *уравнение конического сечения*:

$$r = \frac{p}{1 + e\cos\phi} \tag{2}$$

Где e – эксцентриситет орбиты, а p – орбитальный параметр, равные соответственно:

$$e = \sqrt{\frac{2M^2E}{m\alpha^2} + 1} \tag{3}$$

$$p = \frac{M^2}{m\alpha} \tag{4}$$

Для эллиптической орбиты эксцентриситет соответствует:

$$0 < e < 1$$

И, соответственно, полная механическая энергия $E < 0$.

Зависимость r от t для движения в центральном поле:

Зависимость длины вектора r от времени t дается интегральным выражением:

$$t = \pm \int_{r_0}^{r} \frac{dr}{\sqrt{\frac{2}{m}\left(E + \frac{\alpha}{r}\right) - \frac{M^2}{m^2 r^2}}}$$

Выполним преобразования, учитывая, что орбита эллиптическая ($E < 0$):

$$t = \pm \sqrt{\frac{m}{2|E|}} \int_{r_0}^{r} \frac{rdr}{\sqrt{-r^2 + \frac{\alpha}{|E|} - \frac{M^2}{2m|E|}}}$$

Учитывая, что для эллиптической орбиты справедливо [1, с. 52]:

$$a = \frac{p}{1 - e^2} = \frac{\alpha}{2|E|}$$

Где, a – *большая полуось эллипса.* Подставляя в интегральное выражение, получим:

$$t = \sqrt{\frac{ma}{\alpha}} \int_{r_0}^{r} \frac{rdr}{\sqrt{a^2 e^2 - (r - a^2)^2}}$$

С помощью подстановки [1, с. 54]:

$$r - a = -aecos\xi$$

Запишем интеграл в виде:

$$t = \sqrt{\frac{ma^3}{\alpha}} \int_{r_0}^{r} (1 - ecos\xi)\, d\xi = \sqrt{\frac{ma^3}{\alpha}} (\xi - esin\xi)$$

Таким образом, при помощи введения параметра и разделив переменные, окончательно запишем неявную зависимость r(t):

$$r = a(1 - ecos\xi)$$

$$t = \sqrt{\frac{ma^3}{\alpha}} (\xi - esin\xi) \qquad (5)$$

Где ξ- *эксцентрическая аномалия* (рис.2). Связь ξ и ϕ:

$$cos\xi = \frac{e + cos\varphi}{1 + ecos\varphi}$$

$$\phi = \xi + 2arctan\frac{\beta \sin \xi}{1 - \beta \cos \xi} \qquad (6)$$

Где β - некий коэффициент, равный $\beta = \frac{e}{1+\sqrt{1-e^2}}$

2.5 Зависимость эксцентриситета от скорости $e(v)$, первичные параметры моделируемой системы

В ходе исследования была выявлена непредвиденная проблема – неопределённость начальных скоростей тел. Исходя из функции Лагранжа, формула (1), видно, что она содержит в себе начальные положения и начальные скорости тел. Начальное положение тела определяется либо как его радиус-вектор $\vec{r_i}$, либо более явно, как начальное расстояние между телами r_0. Однако начальные скорости тел (а тем более их вектор) определить не так легко. Для того чтобы выяснить, *выбор каких скоростей физически обоснован, необходимо получить зависимость* $e(v)$, её можно выразить из формулы (3). Направим векторы скоростей под углом $\gamma = 60$ к соединяющему их вектору, будем считать, что векторы скоростей тел одинаковы и направлены противоположно относительно друг друга.(Рис. 2):

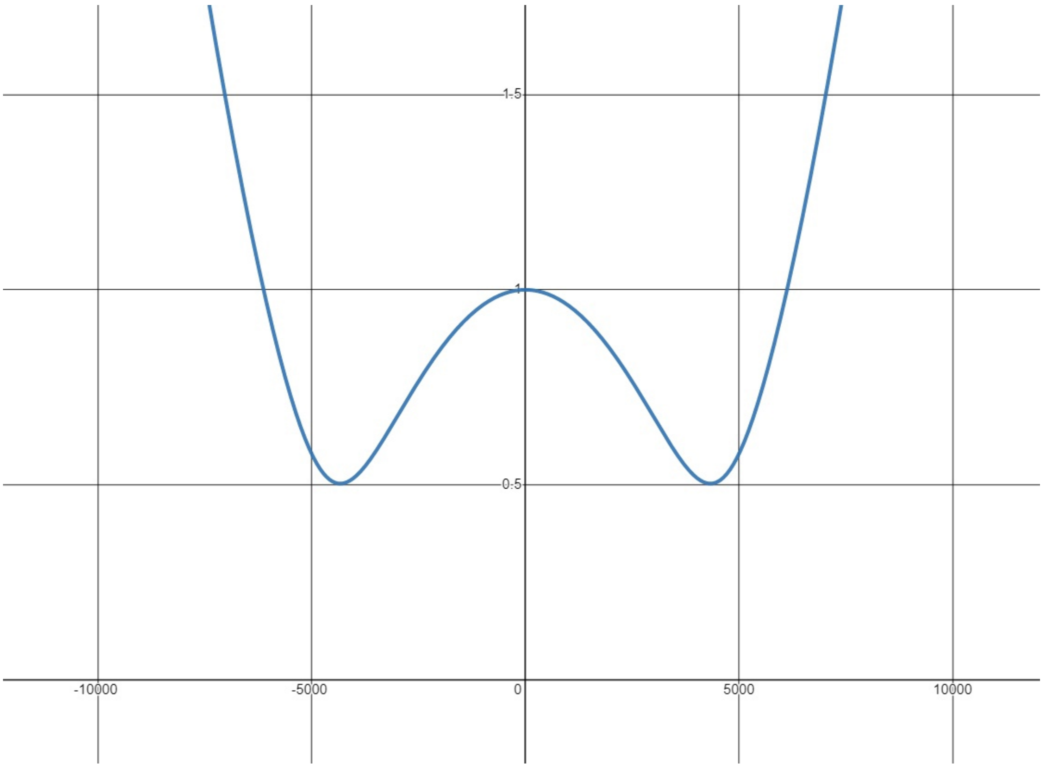
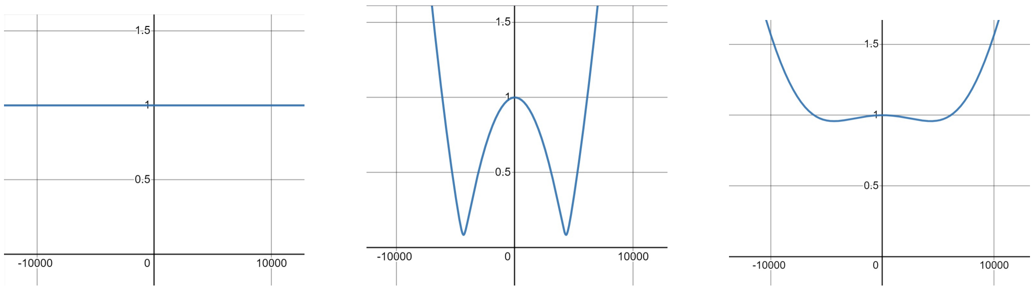


Рис. 1: Зависимость эксцентриситета от скорости тел

График симметричный, с тремя экстремумами, отрицательные значения горизонтальной оси соответствуют противоположным направлениям векторов скорости тел. Заметим, что при отсутствии скорости $e = 1$, это значит что траектория будет иметь вид параболы и движение будет финитно, а двойная система не образуется. Видно, что с увеличением скорости, вплоть до некоторой точки минимума, эксцентриситет уменьшается, это значит, что траектория движения становится более похожа на окружность. После точки минимума, значение функции начинает монотонно возрастать, а значит орбита из эллиптической переходит в параболическую, а затем в гиперболическую и система также перестаёт быть замкнутой. Для эллиптической орбиты характерно $0 < e < 1$.



(a) График при $\gamma = 0$ (b) График при $\gamma = \frac{\pi}{2}$ (c) График при $\gamma \to \pi$

Рис. 2: Вариации зависимости $e(v)$ при разных направлениях $\vec{v}$

Попробуем поменять направление скоростей тел. Если векторы скоростей будут направлены под углом $\gamma = 0$, то есть от одного тела к другому, то траектория будет параболической при любых скоростях. Если же изменять угол между радиус-вектором и вектором скорости до $\gamma = \frac{\pi}{2}$, график станет круче, но его минимумы не достигнут оси x, значит выполняется первый закон Кеплера. Изменяя угол направления скорости далее до π, график начнет выпрямляться.

3 Создание компьютерной модели

Процесс создания модели делится на три этапа: *вычисление постоянных величин системы, создание координатной области и анимация движения тел.* Для работы модели понадобятся модули NumPy, matplotlib, math, а также модуль openpyxl, необходимый для работы с Excel таблицами.

3.1 Вычисление констант computing.py

Введём функцию, необходимую для возведения в отрицательную степень.

```
def power(a, n):
    res = 1
    for k in range(abs(n)):
        res *= a
    if n >= 0:
        return res
    else:
        return 1 / res
```

Запишем физические постоянные (гравитационнная, значение астрономической единицы в метрах, значение солнечной массы в килограммах).

```
G = 6.67 * power(10, -11)
solar_mass = 1.98847 * power(10, 30)
a_u = 149597870700
```

Запишем массу, расстояние между телами (соответствующие, например, системе Альфа-Центавра АБ), значение скорости и угол между вектором расстояния и вектором скорости.

```
m1 = 1.0788 * solar_mass
m2 = 0.9092 * solar_mass
r0 = 23.45 * a_u
v = 4000
alfa = 60
```

Вычислим постоянные значения системы, включая эксцентриситет, орбитальный параметр, большую полуось, максимальное и минимальное расстояние между телами и период обращения.

```
m = (m1*m2)/(m1+m2)
E = ( v**2 * (m1 + m2) / 2 ) - ( G*m1*m2 / r0 )
M = ( (m2 / (m1 + m2)) * r0 * (m1 * v) * sin(alfa)) + ( (m1 /
                                (m1 + m2)) * r0 * (m2 * v) *
                               sin(alfa))

e = sqrt((2*E*power(M, 2)) / (m * power(G*m1*m2, 2)) + 1)
p = (power(M, 2)) / (m * (G*m1*m2)) / a_u

major_axis = p/(1 - power(e,2))

r_max = major_axis * (1 + e)
r_min = major_axis * (1 - e)
```

```
period = 2 * np.pi * sqrt(( (major_axis) * a_u)**3 / (G * (m1
                                                   + m2)))
```

Полученные данные запишем в таблицу для дальнейшей работы.

3.2 Построение координатной системы coordinates.py

Теперь используем функциональную зависимость времени от эксцентрической аномалии (5). С помощью неё нам необходимо проанализировать, как изменятеся угол поворота вектора смещения от времени - $\phi(t)$. Математически это сделать невозможно, так что используем для этого алгоритм. Создадим массив всех возможных значений эксцентрической аномалии ξ, от 0 до 2π, с достаточно малым шагом, например 0.001.

```
x = np.arange(0, (2 * np.pi + 0.001), 0.001)
```

Далее, выберем эталонное значения, на которое каждое следующее значение времени должно быть больше предыдущего, например 0.5 года.

```
time_piece = 0.5
```

Вычислим время для каждого малого угла поворота из массива, и самые близкие к эталонному значения апроксимируем с конечной точностью.

```
selected = 0
chosen = [100, 0, 0]

approximate_dots = {}
chosen_dots = {}

for i in x:
    y = sqrt(m * a**3 / p) / 31536000 * (i - e * np.sin(i))

    if y >= (time_piece - 0.1) and y <= (time_piece + 0.1):
        partial = abs(time_piece - y)
        approximate_dots[i] = [y, partial]

    elif y > (float(time_piece) + 0.1):

        for j in approximate_dots:
            selected = approximate_dots[j][1]

            if selected < chosen[0]:
                chosen = [selected, approximate_dots[j][0], j
                                                        ]
            else:
                continue
```

```
        chosen_dots[chosen[2]] = chosen[1]
        chosen = [100, 0, 0]
        approximate_dots = {}

        time_piece += 0.5
    else:
        continue
```

Записываем результат в таблицу.

```
count = 1

for i in chosen_dots:
    sheet['A' + str(count)] = i
    sheet['B' + str(count)] = chosen_dots[i]
    count += 1
```

Завершающим шагом будет перевести значения ξ в значения угла ϕ, с помощью упомянутой ранее зависимости (6). Объявим функцию coordinates_computing(phase), которую используем в дальнейшем.

```
phase = []
b = e / ( 1 + sqrt((1 - e**2)) )
def coordinates_computing(phase):
    for i in range(1, sheet.max_row, 1):
        eccentric_anomaly = sheet['A' + str(i)].value
        true_anomaly = eccentric_anomaly + 2 * atan (b * sin(
                                           eccentric_anomaly) / (
                                           1 - b * cos(
                                           eccentric_anomaly)))
        phase.append(true_anomaly)

    return phase
```

Таким образом мы получим массив, где время изменяется на одну и ту же величину, а угол ϕ изменяется по какому-то закону.

3.3 Анимация модели animation.py

Модель должна отрисовывать состояние системы в каждый малый момент времени. Для этого необходимо использовать модуль matplotlib. Для графического изображения орбит в плоскости полярных координат (r, ϕ) введём другую прямоугольную систему координат $(x; y)$, с началом в центре инерции. Используя уравнение полярной системы координат (2), найдём длину вектора r для каждого угла ϕ и, затем, спроектируем конец вектора на оси ox и oy. Запишем все проекции в массив и, таким образом, получим координаты всех возможных точек орбит каждого тела.

```
x_trajectory = [[], []]
y_trajectory = [[], []]

def r(fi):
    r = ((p) / (1 + e * cos(fi)))
    u = r * np.cos(fi)
    w = r * np.sin(fi)
    return [u, w, r]

def trajectory(fi):
    x_trajectory[0].append((m2 / (m1 + m2)) * (r(fi)[0]))
    y_trajectory[0].append((m2 / (m1 + m2)) * (r(fi)[1]))

    x_trajectory[1].append((m1 / (m1 + m2)) * -(r(fi)[0]))
    y_trajectory[1].append((m1 / (m1 + m2)) * -(r(fi)[1]))

fi_angel = np.arange(0, 2*np.pi + 0.1, 0.1)
for i in fi_angel:
    trajectory(i)
```

Создадим базовое полотно matplotlib, укажим переменные, обозначающие радиус-векторы тел и сами тела, зададим максимальные и минимальные значения осей *ox* и *oy*, зависящие от максимального и минимального расстояния между телами.

```
fi_angel = np.arange(0, 2*np.pi + 0.1, 0.1)

fig, ax = plt.subplots()

quiver_1 = ax.quiver(0, 0, 0, 0, angles='xy', scale_units='xy
                               ', scale=1, color="b")
quiver_2 = ax.quiver(0, 0, 0, 0, angles='xy', scale_units='xy
                               ', scale=1, color="b")

body_1 = ax.scatter(x_trajectory[0][0], y_trajectory[0][0],
                               color="C0")
body_2 = ax.scatter(x_trajectory[1][0], y_trajectory[1][0],
                               color='C1')

ax.set_xlim(-round(r_max, 0) - 5, round(r_max, 0) + 5)
ax.set_ylim(-round(r_max, 0) - 5, round(r_max, 0) + 5)

ax.grid()
fig.suptitle('Motion of two bodies')
```

Создадим функцию, анимирующую движение тел по орбитам. Эта функция будет использовать данные, полученные в файле coordinates.py, она должна принимать все записанные нами значения угла ϕ, соответствующие равным промежуткам времени, и поворачивать вектор смещения r на принятую величину.

```
def motion(phase):
    global body_1, body_2, year, quiver_1, quiver_2

    body_1.remove()
    body_2.remove()

    quiver_1.remove()
    quiver_2.remove()

    body_1 = ax.scatter((m2 / (m1 + m2)) * (r(phase)[0]), (m2
                                    / (m1 + m2)) * (r(phase)[
                                   1]), color="C0")
    body_2 = ax.scatter((m1 / (m1 + m2)) * -(r(phase)[0]), (
                                   m1 / (m1 + m2)) * -(r(
                                   phase)[1]), color='C1')

    quiver_1 = ax.quiver(0, 0, (m2 / (m1 + m2)) * (r(phase)[0
                                   ]), (m2 / (m1 + m2)) * (r(
                                   phase)[1]), angles='xy',
                                   scale_units='xy', scale=1,
                                    color="C0")
    quiver_2 = ax.quiver(0, 0, (m1 / (m1 + m2)) * -(r(phase)[
                                   0]), (m1 / (m1 + m2)) * -(
                                   r(phase)[1]), angles='xy',
                                    scale_units='xy', scale=1
                                   , color='C1')
```

Для получения массива со значениями углов поворота, используем объявленную ранее функцию, передав в неё пустой список.

```
phase = coordinates_computing([])
```

Свяжем функцию обновления полотна с массивом значений угла ϕ с помощью встроенного в модуль matplotlib метода FuncAnimation.

```
motion_ani = FuncAnimation(
    fig,
    motion,
    frames=phase,
    interval=0,
    repeat=True)

plt.show()
input()
```

Финальный внешний вид модели представлен на (Рис. 3).

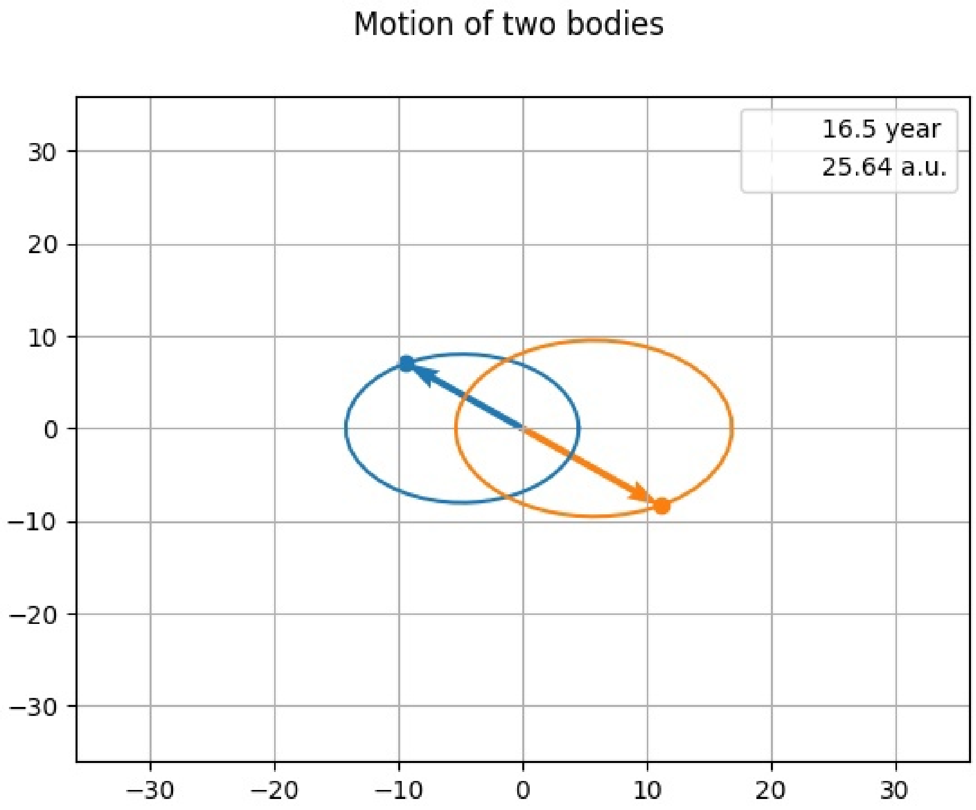


Рис. 3: Интерфейс модели

4 Заключение

4.1 Итоги работы с теоретическим материалом

В ходе исследования фундаментальных физических закономерностей и работы с математическими зависимостями, необходимых для достижения основной цели проекта, были получены уникальные научные данные:

- Была подробна сформулирована общая задача двух тел и решена в целях построения компьютерной модели.
- Неявная зависимость длины вектора смещения от времени была сведена, с помощью компьютерных вычислений, к явному виду для любой конфигурации физической системы.

- Построена важная математическая модель, определяющая эксцентриситет орбиты в зависимости от неизвестной начальной скорости и от известных параметров системы.

Всё это даёт исчерпывающее описание характера движения двух массивных тел.

4.2 Достижение основной цели

Главные продукты этой работы:

- Качественный компьютерный алгоритм, анализирующий двойную звёздную систему, основываясь на теоретических данных, обновляющий в ходе работы базу данных.

- Гибкая компьютерная модель, демонстрирующая динамически изменяющуюся двойную систему, и которую возможно использовать в поставленных целях.