МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Крымский федеральный университет им. В.И. Вернадского»

ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

«Моделирование Двойного Математического Маятника»

обязательное задание по дисциплине

«Математическое компьютерное моделирование»

студента 2 курса группы ПИ-222(2)

Федько Руслана Диляверовича

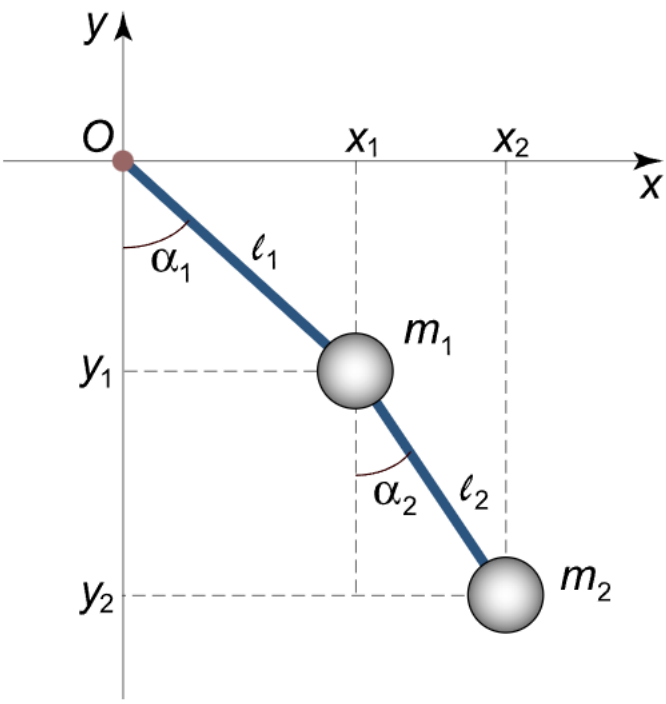
направления подготовки 09.03.04 «Программная инженерия»

Кандидат технических наук, доцент, заведующий кафедрой компьютерной инженерии и моделирования	________________ (оценка) ________________ (подпись, дата)	Милюков В.В.

Симферополь, 2024

Цель: смоделировать движение двойного математического маятника.

Физическая постановка задачи:



The kinetic and potential energy of the pendulums (respectively T and V) are expressed by the formulas

$$T = \frac{m_1 v_1^2}{2} + \frac{m_2 v_2^2}{2} = \frac{m_1\left(\dot{x}_1^2 + \dot{y}_1^2\right)}{2} + \frac{m_2\left(\dot{x}_2^2 + \dot{y}_2^2\right)}{2}, \quad V = m_1 g y_1 + m_2 g y_2.$$

v – скорость, является первой производной от положения(от x’^2 + y’^2).

V2(потенциальная = mgh) где h = y (с рисунка выше).

Лагранжиан (функция Лагранжа), в простейшем случае, - разность между кинетической и потенциальной энергией системы, записанная как функция переменных состояния системы (обобщенных координат и их производных по времени - обобщенных скоростей).

$$L = T - V = T_1 + T_2 - (V_1 + V_2) = \frac{m_1}{2}\left(\dot{x}_1^2 + \dot{y}_1^2\right) + \frac{m_2}{2}\left(\dot{x}_2^2 + \dot{y}_2^2\right) - m_1 g y_1 - m_2 g y_2.$$

Берем производные от координат и получаем

$$\dot{x}_1 = l_1 \cos\alpha_1 \cdot \dot{\alpha}_1, \quad \dot{x}_2 = l_1 \cos\alpha_1 \cdot \dot{\alpha}_1 + l_2 \cos\alpha_2 \cdot \dot{\alpha}_2,$$

$$\dot{y}_1 = l_1 \sin\alpha_1 \cdot \dot{\alpha}_1, \quad \dot{y}_2 = l_1 \sin\alpha_1 \cdot \dot{\alpha}_1 + l_2 \sin\alpha_2 \cdot \dot{\alpha}_2.$$

Преобразуем T1(для первого маятника) и T2(для второго и получим следующее)

$$T_1 = \frac{m_1}{2}\left(\dot{x}_1^2 + \dot{y}_1^2\right) = \frac{m_1}{2}\left(l_1^2 \dot{\alpha}_1^2 \cos^2\alpha_1 + l_1^2 \dot{\alpha}_1^2 \sin^2\alpha_1\right) = \frac{m_1}{2} l_1^2 \dot{\alpha}_1^2,$$

$$T_2 = \frac{m_2}{2}\left(\dot{x}_2^2 + \dot{y}_2^2\right) = \frac{m_2}{2}\left[\left(l_1\dot{\alpha}_1 \cos\alpha_1 + l_2\dot{\alpha}_2 \cos\alpha_2\right)^2 + \left(l_1\dot{\alpha}_1 \sin\alpha_1 + l_2\dot{\alpha}_2 \sin\alpha_2\right)^2\right]$$

Преобразуем скорости, высчитаем y (через гипотенузу и угол между ними)

$$V_1 = m_1 g y_1 = -m_1 g l_1 \cos\alpha_1,$$

$$V_2 = m_2 g y_2 = -m_2 g \left(l_1 \cos\alpha_1 + l_2 \cos\alpha_2\right).$$

В результате система лагранжиана принимает следующий вид.

$$L = T - V = T_1 + T_2 - (V_1 + V_2) =$$

$$\left(\frac{m_1}{2} + \frac{m_2}{2}\right) l_1^2 \dot{\alpha}_1^2 + \frac{m_2}{2} l_2^2 \dot{\alpha}_2^2 + m_2 l_1 l_2 \dot{\alpha}_1 \dot{\alpha}_2 \cos\left(\alpha_1 - \alpha_2\right) + \left(m_1 + m_2\right) g l_1 \cos\alpha_1 + m_2 g l_2 \cos\alpha_2.$$

Теперь мы можем записать уравнения Лагранжа (иногда их называют уравнениями Эйлера-Лагранжа):

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\alpha}_i} - \frac{\partial L}{\partial \alpha_i} = 0, \quad i = 1, 2.$$

Частные производные в этих уравнениях выражаются следующими формулами

$$\frac{\partial L}{\partial \dot{\alpha}_1} = (m_1 + m_2)\, l_1^2 \dot{\alpha}_1 + m_2 l_1 l_2 \dot{\alpha}_2 \cos(\alpha_1 - \alpha_2)\,,$$

$$\frac{\partial L}{\partial \alpha_1} = -m_2 l_1 l_2 \dot{\alpha}_1 \dot{\alpha}_2 \sin(\alpha_1 - \alpha_2) - (m_1 + m_2)\, g l_1 \sin\alpha_1,$$

$$\frac{\partial L}{\partial \dot{\alpha}_2} = m_2 l_2^2 \dot{\alpha}_2 + m_2 l_1 l_2 \dot{\alpha}_1 \cos(\alpha_1 - \alpha_2)\,,$$

$$\frac{\partial L}{\partial \alpha_2} = m_2 l_1 l_2 \dot{\alpha}_1 \dot{\alpha}_2 \sin(\alpha_1 - \alpha_2) - m_2 g l_2 \sin\alpha_2.$$

Следовательно, первое уравнение Лагранжа можно записать в виде

$$\frac{d}{dt}\left[(m_1 + m_2)\, l_1^2 \dot{\alpha}_1 + m_2 l_1 l_2 \dot{\alpha}_2 \cos(\alpha_1 - \alpha_2)\right] + m_2 l_1 l_2 \dot{\alpha}_1 \dot{\alpha}_2 \sin(\alpha_1 - \alpha_2) + (m_1 + m_2)\, g l_1 \sin\alpha_1 = 0,$$

$$\Rightarrow (m_1 + m_2)\, l_1^2 \ddot{\alpha}_1 + m_2 l_1 l_2 \ddot{\alpha}_2 \cos(\alpha_1 - \alpha_2) + m_2 l_1 l_2 \dot{\alpha}_2^2 \sin(\alpha_1 - \alpha_2) + (m_1 + m_2)\, g l_1 \sin\alpha_1 = 0.$$

Аналогично выводим второе дифференциальное уравнение

$$\frac{d}{dt}\left[m_2 l_2^2 \dot{\alpha}_2 + m_2 l_1 l_2 \dot{\alpha}_1 \cos(\alpha_1 - \alpha_2)\right] - m_2 l_1 l_2 \dot{\alpha}_1 \dot{\alpha}_2 \sin(\alpha_1 - \alpha_2) + m_2 g l_2 \sin\alpha_2 = 0,$$

$$\Rightarrow m_2 l_2^2 \ddot{\alpha}_2 + m_2 l_1 l_2 \ddot{\alpha}_1 \cos(\alpha_1 - \alpha_2) - m_2 l_1 l_2 \dot{\alpha}_1^2 \sin(\alpha_1 - \alpha_2) + m_2 g l_2 \sin\alpha_2 = 0.$$

Таким образом, нелинейную систему двух дифференциальных уравнений Лагранжа можно записать в виде

$$\begin{cases}(m_1+m_2)\,l_1\ddot{\alpha}_1+m_2l_2\ddot{\alpha}_2\cos(\alpha_1-\alpha_2)+m_2l_2\dot{\alpha}_2^2\sin(\alpha_1-\alpha_2)+(m_1+m_2)\,g\sin\alpha_1=0\\ l_2\ddot{\alpha}_2+l_1\ddot{\alpha}_1\cos(\alpha_1-\alpha_2)-l_1\dot{\alpha}_1^2\sin(\alpha_1-\alpha_2)+g\sin\alpha_2=0\end{cases}.$$

Уравнения движения, получаемые из уравнений Эйлера — Лагранжа, можно записать следующим образом(случай когда массы и длинны двух маятников совпадают)

$$\begin{cases}\dot{\theta}_1=\dfrac{6}{m\ell^2}\dfrac{2p_{\theta_1}-3\cos(\theta_1-\theta_2)p_{\theta_2}}{16-9\cos^2(\theta_1-\theta_2)}\\ \\ \dot{\theta}_2=\dfrac{6}{m\ell^2}\dfrac{8p_{\theta_2}-3\cos(\theta_1-\theta_2)p_{\theta_1}}{16-9\cos^2(\theta_1-\theta_2)}.\end{cases}$$

$$\begin{cases}\dot{p}_{\theta_1}=\dfrac{\partial L}{\partial\theta_1}=-\frac{1}{2}m\ell^2\left[\dot{\theta}_1\dot{\theta}_2\sin(\theta_1-\theta_2)+3\frac{g}{\ell}\sin\theta_1\right]\\ \\ \dot{p}_{\theta_2}=\dfrac{\partial L}{\partial\theta_2}=-\frac{1}{2}m\ell^2\left[-\dot{\theta}_1\dot{\theta}_2\sin(\theta_1-\theta_2)+\frac{g}{\ell}\sin\theta_2\right].\end{cases}$$

Последние четыре уравнения являются явными формулами для временной эволюций системы с заданным текущим состоянием. Невозможно продвинуться дальше и интегрировать эти уравнения аналитически, чтобы получить формулы для θ1 и θ2 как функции от времени. Однако возможно выполнить численное интегрирование, используя метод Рунге — Кутты

Метод Рунге — Кутты 4-го порядка.

Задача состоит в том, чтобы найти значение неизвестной функции y в заданной точке x.

Метод Рунге-Кутты находит приблизительное значение y для заданного x . С помощью метода Рунге-Кутты 4-го порядка можно решить только обыкновенные дифференциальные уравнения первого порядка.

Ниже приведена формула, используемая для вычисления следующего значения yn + 1 из предыдущего значения yn. Значение n равно 0, 1, 2, 3, ...(x – x0) / ч. Здесь h - высота ступени и xn + 1 = x0 + h. Меньший размер шага означает большую точность.

$$\begin{aligned}
K_1 &= hf(x_n, y_n) \\
K_2 &= hf(x_n + \tfrac{h}{2}, y_n + \tfrac{k_1}{2}) \\
K_3 &= hf(x_n + \tfrac{h}{2}, y_n + \tfrac{k_2}{2}) \\
K_4 &= hf(x_n + h, y_n + k_3) \\
y_{n+1} &= y_n + k_1/6 + k_2/3 + k_3/3 + k_4/6 + O(h^5)
\end{aligned}$$

Формула в основном вычисляет следующее значение yn + 1, используя текущее значение yn плюс средневзвешенное значение с четырьмя приращениями.

k1 - это приращение, основанное на наклоне в начале интервала, с использованием y

k2 - это приращение, основанное на наклоне в средней точке интервала, с использованием y + hk1/2.

k3 - это снова приращение, основанное на наклоне в средней точке, с использованием y + hk2 / 2.

k4 - это приращение, основанное на наклоне в конце интервала, с использованием y + hk3.

Имплементация:

Рисуем поле

```
1+ usages
function setup() :void  {
    createCanvas(400, 400);
}
```

Задаем начальные параметры тела

```
let m :number  = parseFloat(document.getElementById( elementId: 'mass').value); // Масса(кг)
let l :number  = parseFloat(document.getElementById( elementId: 'large').value); // Длина шнура (см)
let gr :number  = -parseFloat(document.getElementById( elementId: 'gravity').value); // Ускорение свободного падения
let init :(number)[]  = [Math.PI, Math.PI, 0, 0]; // Данные инициализации для алгоритма RK4(Начальное положение) (два угла, д
let poses1 :any[]  = []; // Массивы координат для отрисовки предыдущих позиаций  первого маятника
let poses2 :any[]  = [];// Массивы координат для отрисовки предыдущих позиаций  второго маятника

let isDrawing :boolean  = false; // Флажок переключителя
```

Функция переключатель

```
function toggleDrawing() :void  { //Функция переключатель
    m = parseFloat(document.getElementById( elementId: 'mass').value);
    l =  parseFloat(document.getElementById( elementId: 'large').value);
    gr = -parseFloat(document.getElementById( elementId: 'gravity').value);
    init = [Math.PI/2, Math.PI/2, 0, 0];
    poses1 = [];
    poses2 = [];
    isDrawing = !isDrawing;
}
```

Считаем ускорение для первого и второго маятника. На основе диф.уравнений.

```
// Считаем ускорение для первого маятника
1+ usages
function f(t1, t2, p1, p2) {
    return (
        ((6 / (m * l * l)) * (2 * p1 - 3 * Math.cos( x: t1 - t2) * p2)) /
        (16 - 9 * Math.cos( x: t1 - t2) * Math.cos( x: t1 - t2))
    );
}
// Считаем ускорение для второго маятника
1+ usages
function g(t1, t2, p1, p2) {
    return (
        ((6 / (m * l * l)) * (8 * p2 - 3 * Math.cos( x: t1 - t2) * p1)) /
        (16 - 9 * Math.cos( x: t1 - t2) * Math.cos( x: t1 - t2))
    );
}
```

Считаем скорость первого и второго маятника на основе ускорения

```
// Считаем скорости первого маятника на основе его ускорения
1+ usages
function j(t1, t2, p1, p2) {
    return (
        -0.5 *
        m *
        l *
        l *
        (f(t1, t2, p1, p2) * g(t1, t2, p1, p2) * Math.sin( x: t1 - t2) +
            ((3 * gr) / l) * Math.sin(t1))
    );
}

//считаем скорости второго маятника на основе его ускорения
1+ usages
function k(t1, t2, p1, p2) {
    return (
        -0.5 *
        m *
        l *
        l *
        (-1 * f(t1, t2, p1, p2) * g(t1, t2, p1, p2) * Math.sin( x: t1 - t2) +
            (gr / l) * Math.sin(t2))
    );
}
```

Считаем новые углы и скорости при помощи интегратора RK4.

```
function RK4(init) : [any, any, any, any]  {
    const t1 = init[0]; // Присваиваю начальные данные Начальный угол первого маятника
    const t2 = init[1]; // Начальный угол второго маятника
    const p1 = init[2]; //  начальная скорость первого маятника
    const p2 = init[3]; // начальная скорость второго маятника

    const h : number  = 0.08; // Шаг интегирования

    // k0 - это приращение, основанное на наклоне в начале интервала, с использованием y
    // k1 - это приращение, основанное на наклоне в средней точке интервала, с использованием y + hk1/2.
    // k2 - это снова приращение, основанное на наклоне в средней точке, с использованием y + hk2 / 2.
    // k3 - это приращение, основанное на наклоне в конце интервала, с использованием y + hk3.

    const k0 : number  = h * f(t1, t2, p1, p2); // Приращение к первому углу
    const l0 : number  = h * g(t1, t2, p1, p2); // Приращение к второму углу
    const m0 : number  = h * j(t1, t2, p1, p2);  // Приращение к первой угловой скорости
    const n0 : number  = h * k(t1, t2, p1, p2); // Приращение к второй угловой скорости

    const k1 : number  = h * f( t1: t1 + 0.5 * k0,  t2: t2 + 0.5 * l0,  p1: p1 + 0.5 * m0,  p2: p2 + 0.5 * n0);
    const l1 : number  = h * g( t1: t1 + 0.5 * k0,  t2: t2 + 0.5 * l0,  p1: p1 + 0.5 * m0,  p2: p2 + 0.5 * n0);
    const m1 : number  = h * j( t1: t1 + 0.5 * k0,  t2: t2 + 0.5 * l0,  p1: p1 + 0.5 * m0,  p2: p2 + 0.5 * n0);
    const n1 : number  = h * k( t1: t1 + 0.5 * k0,  t2: t2 + 0.5 * l0,  p1: p1 + 0.5 * m0,  p2: p2 + 0.5 * n0);

    const k2 : number  = h * f( t1: t1 + 0.5 * k1,  t2: t2 + 0.5 * l1,  p1: p1 + 0.5 * m1,  p2: p2 + 0.5 * n1);
    const l2 : number  = h * g( t1: t1 + 0.5 * k1,  t2: t2 + 0.5 * l1,  p1: p1 + 0.5 * m1,  p2: p2 + 0.5 * n1);
    const m2 : number  = h * j( t1: t1 + 0.5 * k1,  t2: t2 + 0.5 * l1,  p1: p1 + 0.5 * m1,  p2: p2 + 0.5 * n1);
    const n2 : number  = h * k( t1: t1 + 0.5 * k1,  t2: t2 + 0.5 * l1,  p1: p1 + 0.5 * m1,  p2: p2 + 0.5 * n1);

    const k3 : number  = h * f( t1: t1 + k2,  t2: t2 + l2,  p1: p1 + m2,  p2: p2 + n2);
    const l3 : number  = h * g( t1: t1 + k2,  t2: t2 + l2,  p1: p1 + m2,  p2: p2 + n2);
    const m3 : number  = h * j( t1: t1 + k2,  t2: t2 + l2,  p1: p1 + m2,  p2: p2 + n2);
    const n3 : number  = h * k( t1: t1 + k2,  t2: t2 + l2,  p1: p1 + m2,  p2: p2 + n2);

    //
    const nt1 = t1 + (1 / 6) * (k0 + 2 * k1 + 2 * k2 + k3); // К начальному значению прибавляю приращение функции на шаг интегрирования
    const nt2 = t2 + (1 / 6) * (l0 + 2 * l1 + 2 * l2 + l3);
    const np1 = p1 + (1 / 6) * (m0 + 2 * m1 + 2 * m2 + m3);
    const np2 = p2 + (1 / 6) * (n0 + 2 * n1 + 2 * n2 + n3);

    return [nt1, nt2, np1, np2];
```

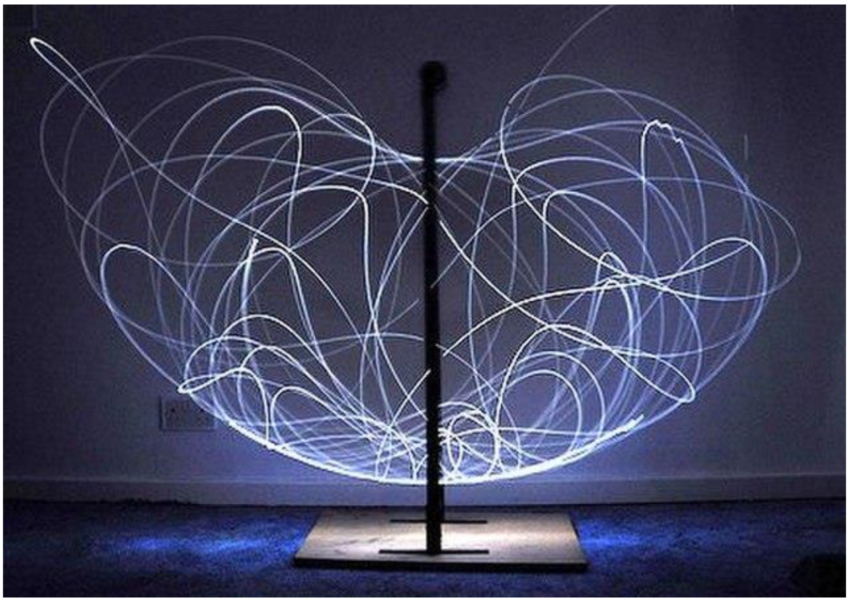
Рисуем новые положения маятников на основе вычислений из RK4.

```
function draw() : void  {
    background(220);

    if (isDrawing) {

        init = RK4(init); // Получение новых углов через шаг времени

        const t1 = init[0]; // Угол t1 в момент времени t
        const t2 = init[1];// Угол t2 в момент времени t
        let x1 : number  = l * Math.sin(t1) + 200; // считаем координаты по прямоугольному треугольнику
        let y1 : number  = -l * Math.cos(t1) + 200;// считаем координаты по прямоугольному треугольнику
        poses1.push({x1, y1});

        let x2 : number  = l * Math.sin(t2) + x1; // считаем координаты по прямоугольному треугольнику
        let y2 : number  = -l * Math.cos(t2) + y1; // считаем координаты по прямоугольному треугольнику
        poses2.push({x2, y2});

        // if (poses1.length > 50) poses1.shift();
        // if (poses2.length > 100) poses2.shift();

        for (let p : number  = 0; p < poses1.length - 1; p++) {
            stroke(color(255, 0, 0, 100));
            line(poses1[p].x1, poses1[p].y1, poses1[p + 1].x1, poses1[p + 1].y1);
        }

        for (let p : number  = 0; p < poses2.length - 1; p++) {
            stroke(color(0, 0, 255, 100));
            line(poses2[p].x2, poses2[p].y2, poses2[p + 1].x2, poses2[p + 1].y2);
        }

        stroke(color(0, 0, 0, 255));
        line(200, 200, x1, y1);
        fill(color(255, 0, 0, 255));
        circle(x1, y1, m*10);
        line(x1, y1, x2, y2);
        fill(color(0, 0, 255, 255));
        circle(x2, y2, m*10);
    }
}
```

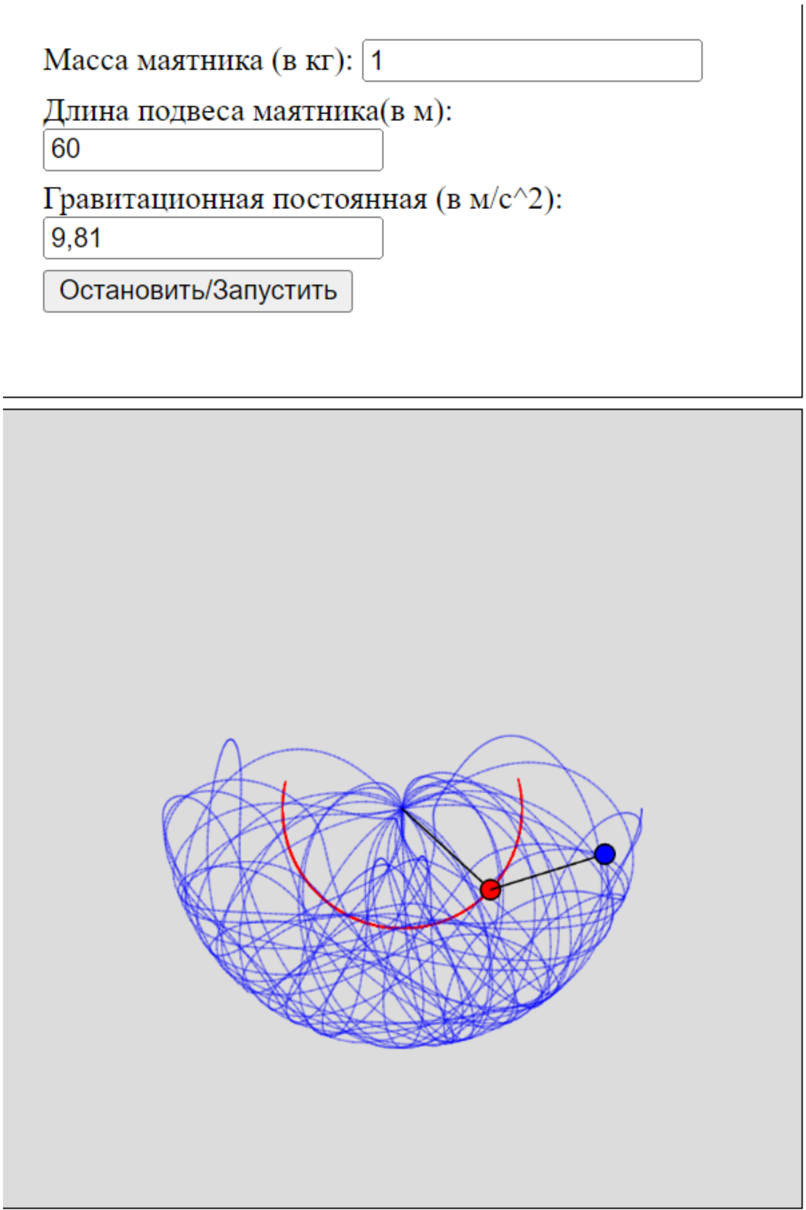
Тестирование:

Запускаем программу и получаем следующий результат моделирования.

Эталон из реальной жизни выглядит вот так



Получаем следующий результат после запуска



Вывод: в ходе работе я смоделировал двойной математический маятник, изучил метод Рунге-Кутты 4 порядка.