МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
НАБЕРЕЖНОЧЕЛНИНСКИЙ ИНСТИТУТ (ФИЛИАЛ)

Г. Х. Тазмеев, А.Т. Галиакбаров, Х. К. Тазмеев,

И. К. Хафизов, А. З. Хазивалиев

Физика в цифровой лаборатории: MATLAB-модели и эксперименты

Учебное пособие к практикуму по физике

Набережные Челны

2025

УДК 53(075.8):004.94
Т13

Печатается по рекомендации Учебно-методической комиссии Высшей технической школы Набережночелнинского института (филиала) Казанского (Приволжского) федерального университета (протокол №1.3.2.37-01/02 от 09.09.2025 г.)

Работа выполнена за счет гранта Академии наук Республики Татарстан, предоставленного молодым кандидатам наук (постдокторантам) с целью защиты докторской диссертации, выполнения научно-исследовательских работ, а также выполнения трудовых функций в научных и образовательных организациях Республики Татарстан в рамках Государственной программы Республики Татарстан «Научно-технологическое развитие Республики Татарстан», соглашение № 42/2024-ПД от 16.12.2024 г.

Рецензенты:
А. Н. Илюхин, кандидат технических наук, доцент
С. К. Савицкий, кандидат педагогических наук, доцент

Т13 **Физика в цифровой лаборатории: MATLAB-модели и эксперименты:** учебно-методическое пособие / Г. Х. Тазмеев, И. К. Хафизов, А.Т. Галиакбаров, Х. К. Тазмеев, А. З. Хазивалиев / – Набережные Челны: Отдел информации и связей с общественностью Набережно-челнинского института КФУ, 2025. – 94 с. – 1065 КБ (PDF) / – Текст: электронный

Учебное пособие подготовлено в соответствии с рабочей программой дисциплины «Физика» и предназначено для студентов технических направлений, осваивающих экспериментальные и вычислительные методы исследования физических явлений с применением пакета MATLAB и цифровых лабораторных средств.

1. Поединок с Голиафом

Давайте решим задачу, связанную с пращой Давида. Праща - это древнее оружие, представляющее собой веревку с карманом для камня. Камень раскручивают вокруг головы, создавая центростремительное ускорение, и отпускают в нужный момент, чтобы поразить цель. Этот пример поможет лучше понять физику движения по параболической траектории и как можно использовать MATLAB для решения таких задачек.

1.1 Теоретическая часть

Параболическое движение - это движение тела, подверженного постоянному ускорению (например, ускорению свободного падения), но при этом движущегося с начальной скоростью, составляющей угол с горизонтом. Оно описывается траекторией в виде параболы, где горизонтальная скорость постоянна, а вертикальная изменяется со временем из-за ускорения g.
Основные уравнения для параболического движения:

Горизонтальное движение:

$$x(t) = v_0 \cdot \cos(\theta) \cdot t \tag{1.1}$$

где $x(t)$ - горизонтальное расстояние в момент времени t,
v_0 - начальная скорость,
θ - угол запуска.

Вертикальное движение:

$$y(t) = v_0 \cdot \sin(\theta) \cdot t - \frac{1}{2} g t^2 \tag{1.2}$$

где $y(t)$ - высота в момент времени t,
g - ускорение свободного падения.

Эти уравнения описывают положение тела в любой момент времени.

Ускорение свободного падения определяет, насколько быстро объект теряет высоту. Чем больше g, тем быстрее материал будет опускаться, а траектория станет более крутой. Если g уменьшится (например, на Луне), то объект будет падать медленнее, и его траектория будет менее крутой, а время полета увеличится.

Закон сохранения энергии говорит, что механическая энергия (сумма кинетической и потенциальной энергии) сохраняется, если нет сопротивления воздуха. На высоте максимума траектории кинетическая энергия будет минимальной, а потенциальная - максимальной. Во время падения кинетическая энергия увеличивается, а потенциальная уменьшается. Однако из-за сопротивления воздуха механическая энергия не сохраняется полностью в реальных условиях.

1.2 Практическая часть

Условия задачи:

1. Расстояние между Давидом и Голиафом $d = X$ м.
2. Высота Голиафа $h = X$ м.
3. Начальная скорость $v_0 = X$ м/с.
4. Угол запуска $\theta = X°$.

Необходимо найти, попадет ли Давид в голову Голиафа, и, если попадет, на какой высоте будет это попадание.

Шаг 1: Преобразование угла в радианы
Поскольку угол задан в градусах, для использования в тригонометрических функциях в MATLAB, его нужно преобразовать в радианы:

$$\theta_{\text{rad}} = \theta \cdot \frac{\pi}{180} \quad (1.3)$$

Шаг 2: Нахождение горизонтальной и вертикальной составляющих начальной скорости

Найдем горизонтальную (v_{0x}) и вертикальную (v_{0y}) компоненты начальной скорости v_0:

$$v_{0x} = v_0 \cdot \cos(\theta_{\text{rad}})$$

$$v_{0y} = v_0 \cdot \sin(\theta_{\text{rad}}) \quad (1.5)$$

Шаг 3: Вычисление времени полета до точки попадания

Так как камень летит по горизонтали с постоянной скоростью v_{0x}, время полета до попадания можно вычислить по формуле:

$$t = \frac{d}{v_{0x}} \quad (1.6)$$

Шаг 4: Вычисление высоты в момент времени t

Для вычисления высоты камня в момент времени t используем уравнение для вертикального движения:

$$y(t) = v_{0y} \cdot t - \frac{1}{2} g t^2 \quad (1.7)$$

где $g = 9.81$ м/с^2 - ускорение свободного падения.

Шаг 5: Проверка попадания

Чтобы проверить попадание, нужно убедиться, что высота $y(t)$ в момент времени t примерно равна высоте головы Голиафа h:

$$| y(t) - h | \leq 0.1 \quad (1.8)$$

Если это условие выполняется, значит, камень попадет в голову Голиафа.

Пример решения задачи в MATLAB:

```
% Исходные данные
d = X;   % Расстояние до Голиафа (м), подставьте свое значение
h = X;    % Высота Голиафа (м), подставьте свое значение
v_0 = X; % Начальная скорость (м/с), подставьте свое значение
theta = X; % Угол запуска (градусы), подставьте свое значение

% Константа
g = 9.81; % ускорение свободного падения (м/с^2) на Земле, можно менять данные из других планет

% Преобразование угла в радианы
theta_rad = deg2rad(theta);

% Горизонтальная и вертикальная компоненты скорости
v_0x = v_0 * cos(theta_rad);
v_0y = v_0 * sin(theta_rad);

% Время полета до попадания
t = d / v_0x;

% Высота в момент времени t
y_t = ?????????

% Проверка попадания
if abs(y_t - h) <= 0.1
   fprintf('С углом %.2f° и скоростью %.2f м/с камень попадет в голову Голиафа!\n', theta, v_0);
   fprintf('Высота попадания: %.2f м\n', y_t);
else
```

```
    fprintf('С углом %.2f° и скоростью %.2f м/с камень не попадет в голову Голиафа.\n', theta, v_0);
    fprintf('Высота в момент попадания: %.2f м\n', y_t);
end
```

С углом 20.00° и скоростью 22.00 м/с камень попадет в голову Голиафа!
Высота попадания: 2.69 м

Рисунок 1.1 Пример вывода решения

Это означает, что с указанным углом и скоростью камень действительно попадет в голову Голиафа, причем высота попадания будет 2.69 метра, что очень близко к 2.77 метрам - высоте головы Голиафа.

1.3 Задачи на практическую часть

1. Подставьте в строчку кода
% Высота в момент времени t
y_t = ?????????
уравнение для вертикального движения
2. Подставьте свои исходные данные таким образом, чтобы камень, выпущенный из пращи, попал в Голиафа.
3. Придумайте свой фантастический сценарий и запустите его успешно, где место действия будет происходить не на Земле с $g = 9.81$ м/с^2, а на другой планете, данные возьмите из интернета.

1.4 Контрольные вопросы

1. Как изменится траектория движения камня, если угол запуска увеличится? Что произойдет, если скорость увеличится? Объясните влияние каждого из этих факторов на дальность полета и высоту.

2. Почему время полета не зависит от высоты Голиафа?
3. Почему для проверки попадания важно учитывать высоту Голиафа в момент времени попадания, а не просто проверять, достиг ли камень расстояния d?
4. Как закон сохранения энергии объясняет движение камня в параболической траектории? Почему кинетическая энергия в начале и в конце полета не равна?
5. Как ускорение свободного падения g влияет на траекторию камня и его время полета? Как бы изменилась траектория, если бы ускорение свободного падения было меньше или больше?
6. Почему в реальных условиях (с сопротивлением воздуха) траектория камня будет отличаться от идеальной параболы? Как это повлияет на попадание в Голиафа?
7. Какое влияние на точность попадания могут иметь погрешности в измерении начальной скорости или угла запуска? Как можно уменьшить эти погрешности?

2. Индиана Джонс и катящийся камень

Индиана Джонс оказался в древнем храме, и теперь ему предстоит спастись от катящегося на него гигантского камня. Камень катится по наклонному склону, а профессор должен добежать до выхода из храма, чтобы избежать столкновения. Однако, камень набирает скорость, и задача состоит в том, чтобы вычислить, успеет ли наш герой добежать до выхода или столкнется с камнем.

2.1 Теоретическая часть

Кинематика движения

При решении задачи рассматриваются два типа движения:
Равноускоренное движение (движение камня) – тело движется с постоянным ускорением.
Равномерное движение (движение Индианы Джонса) – тело движется с постоянной скоростью.

Уравнения движения

Равноускоренное движение (без начальной скорости)

Если объект движется с ускорением a и начальная скорость $v_0 = 0$, его положение во времени определяется формулой:

$$x = x_0 + \frac{1}{2}at^2 \qquad (2.1)$$

где
x_0 – начальное положение (для камня $x_0 = 0$),
a – ускорение (камень движется под действием силы тяжести),
t – время.
Скорость камня в момент времени t определяется по формуле:

$$v = v_0 + at \tag{2.2}$$

Поскольку $v_0 = 0$, то:

$$v = at \tag{2.3}$$

Равномерное движение

Если объект движется с постоянной скоростью v, его положение во времени рассчитывается так:

$$x = x_0 + vt \tag{2.4}$$

где:

x_0 – начальная позиция (для Индианы Джонса $x_0 = X$ м),

v – постоянная скорость,

t – время.

Динамика движения

Силы, действующие на камень

Камень находится на наклонной поверхности, и на него действуют следующие силы:

Сила тяжести $F_g = mg$, направленная вертикально вниз.

Составляющая силы тяжести вдоль наклонной плоскости:

$$F_{\parallel} = mg\sin\theta \tag{2.5}$$

Эта сила вызывает ускоренное движение камня.

Реакция опоры

$$N = mg\cos\theta, \tag{2.6}$$

которая компенсирует перпендикулярную составляющую силы тяжести.

Ускорение камня

Из второго закона Ньютона:

$$ma = mg\sin\theta \tag{2.7}$$

Сокращая массу, получаем ускорение:

$$a = g\sin\theta \tag{2.8}$$

где $g \approx 9.81$ м/с^2.

Анализ столкновения

Столкновение происходит, когда позиции Индианы Джонса и камня совпадают:

$$x_{\text{stone}}(t) = x_{\text{Ind}}(t) \tag{2.9}$$

Подставляя уравнения движения:

$$\frac{1}{2}at^2 = x_0 + v_{\text{Ind}}t \tag{2.10}$$

Решая квадратное уравнение относительно t, можно найти момент столкновения.

1.2 Практическая часть

Условия задачи

Камень:

Камень катится с постоянным ускорением $a = X$ м/с^2 по наклонной поверхности.

Начальная скорость камня $v_0 = 0$ м/с.

Начальная позиция камня $x_0 = 0$ м.

Индиана Джонс:

Профессор находится в начале храма на позиции $x_0 = X$ м.

Его скорость постоянна и равна $v_{\text{Ind}} = X$ м/с.

Индиана должен добежать до выхода, расположенного в точке $x_{\text{exit}} = X$ м.

Задача:

Необходимо определить, успеет ли Индиана добежать до выхода до того, как камень его догонит. Если Индиана не успеет, нужно определить, на каком расстоянии от выхода это произойдет.

Цель задачи:

1. Рассчитать траекторию движения как камня, так и Индианы Джонса.
2. Построить графики их движения по времени.
3. Найти момент времени, когда профессор добежит до выхода или камень его догонит.
4. Проверить, успеет ли Индиана Джонс добежать до выхода до того, как камень его догонит.

Уравнения движения:

Движение камня: Камень движется по наклонной поверхности с ускорением. Его движение описывается уравнением:

$$x_{\text{stone}}(t) = \frac{1}{2} a t^2 \tag{2.11}$$

где

$x_{\text{stone}}(t)$ - позиция камня в момент времени t,

$a = X$ м/с^2 - ускорение камня,

t - время.

Движение Индианы Джонса: профессор двигается с постоянной скоростью. Его положение определяется уравнением:

$$x_{\text{Ind}}(t) = x_0 + v_{\text{Ind}} t \tag{2.12}$$

где

$x_{\text{Ind}}(t)$ - позиция Индианы Джонса в момент времени t,

$x_0 = \text{X}$ м - начальная позиция Индианы Джонса,

$v_{\text{Ind}} = X$ м/с - скорость Индианы Джонса,

t - время.

Условия для спасения профессора:

Если в какой-то момент t камень не догоняет Индиану Джонса до момента $x_{\text{Ind}}(t) = x_{\text{exit}}$, то Индиана Джонс успевает выйти из храма.

Если Индиану Джонса догоняет камень, то нужно найти момент времени, когда это произойдет и вычислить расстояние до выхода в этот момент.

Шаги решения:

1. Моделирование движения:

Время будет моделироваться от $t = 0$ до некоторого конечного времени, например, $t = 10$ секунд, чтобы увидеть, как меняются позиции Индианы и камня.

2. Построение графиков положения Индианы Джонса и камня по времени.

Нахождение момента столкновения:

Чтобы найти момент столкновения, нужно решить систему уравнений:

$$\frac{1}{2}at^2 = x_0 + v_{\text{Ind}}t \qquad (2.13)$$

Если $t_{\text{collision}} > t_{\text{exit}}$, значит, Индиана Джонс успевает добежать до выхода до столкновения.

3. Анимация движения:

Для визуализации движения можно использовать анимацию, показывая на графике, как изменяются позиции камня и Индианы Джонса с течением времени.

Пример MATLAB кода:

```
% Дано:
a = X;              % ускорение камня, м/с^2
v_Ind = X;          % скорость Индианы Джонса, м/с
x0_stone = 0;       % начальная позиция камня, м
x0_Ind = X;         % начальная позиция Индианы Джонса, м
x_exit = X;         % позиция выхода, м
t_max = 10;         % максимальное время моделирования (в
секундах)
time_steps = 500;   % количество шагов для моделирования
времени

% Вектор времени
t = linspace(0, t_max, time_steps);

% Позиции камня и Индианы Джонса
x_stone = ???????????;      % положение камня
x_Ind = ??????????;     % положение Индианы Джонса

% Положение камня и Индианы Джонса относительно выхода
collision_time = NaN;
for i = 1:length(t)
   if x_stone(i) >= x_Ind(i) && x_Ind(i) < x_exit
      collision_time = t(i);
```

```
        break;
    end
end
% Результат
if ~isnan(collision_time)
    fprintf('Камень догонит Индиану Джонаса через %.2f секунд.\n',
collision_time);
    fprintf('На момент столкновения Индиана Джонс находится на
расстоянии %.2f м от выхода.\n', x_exit - x_Ind(find(t ==
collision_time)));
else
    fprintf('Индиана Джонс успевает добежать до выхода без
столкновения.\n');
end

% Графики движения камня и Индианы Джонса
figure;
hold on;
plot(x_stone, t, 'r', 'LineWidth', 2);  % движение камня
plot(x_Ind, t, 'b', 'LineWidth', 2);    % движение Индианы Джонса
plot(x_exit * ones(size(t)), t, 'g--', 'LineWidth', 2); % линия выхода
xlabel('Позиция (м)');
ylabel('Время (с)');
title('Движение Индианы Джонса и катящегося камня');
legend('Камень', 'Индиана Джонс', 'Выход');
grid on;
hold off;
```

Пояснения к коду:

1. **Моделирование движения**:

Используем уравнения для движения камня и Индианы Джонса. Камень начинает движение с нулевой скорости, но с ускорением.

2. **Проверка столкновения**:

Мы ищем момент времени, когда Индиана Джонса выйдет из храма или когда камень догонит Индиану. В результате вычисляется время столкновения и положение Индианы в этот момент.

3. **График**:

Строим график движения как камня, так и Индианы, и отображаем линию выхода, чтобы увидеть, успевает ли Индиана добежать до выхода до того, как камень его догонит.

Что нужно учесть:

Если Индиана Джонс не успевает добежать до выхода и сталкивается с камнем, нужно точно определить место столкновения.

Если время столкновения больше времени, необходимого для достижения выхода, Индиана Джонс спасется.

Пример вывода решения:

«Камень догонит Индиану Джонаса через 2.97 секунд.
На момент столкновения Индиана находится на расстоянии 37.03 м от выхода»

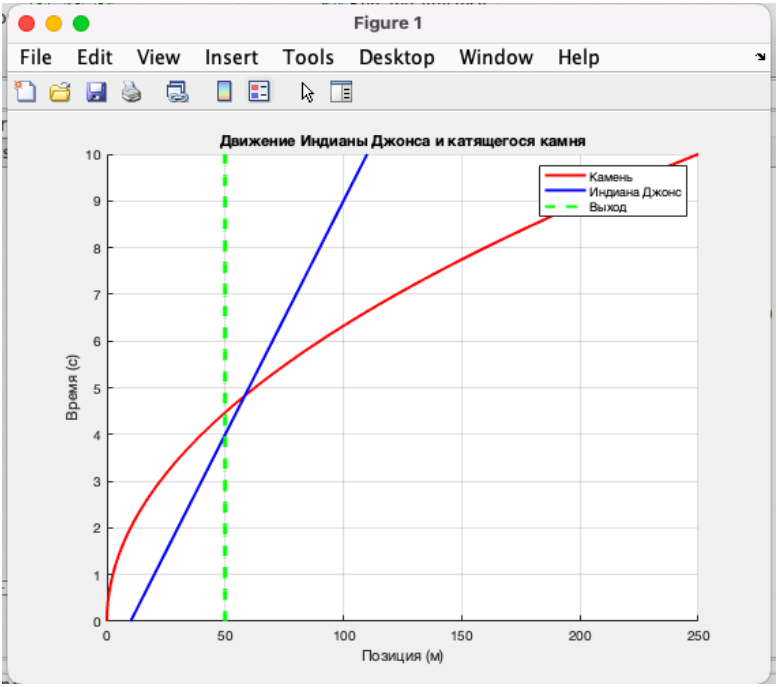

Рисунок 2.1 - Пример графика, при котором Индиана Джонс спасается.

2.3 Задачи на практическую часть

1. Дополните строчки кода

```
% Позиции камня и Индианы Джонса
x_stone = ???????????;       % положение камня
x_Ind = ??????????;     % положение Индианы Джонса
```

системами уравнений

2. Подставьте свои исходные данные таким образом, чтобы Индиана Джонс сумел спастись от катящегося камня.

2.4 Контрольные вопросы

1. Какой вид движения совершает камень? Какие уравнения описывают его движение? Какой вид движения совершает Индиана Джонс? Чем оно отличается от движения камня?
2. Как рассчитать положение камня в любой момент времени, если известно его начальное ускорение? Как рассчитать положение профессора в любой момент времени, если он движется с постоянной скоростью?
3. Какие силы действуют на камень при его движении по наклонной поверхности?
4. Почему камень катится с ускорением, а Индиана Джонс бежит с постоянной скоростью?
5. Как можно определить ускорение камня, если известен угол наклона поверхности и сила тяжести?
6. Как изменилось бы поведение камня, если бы он не катился, а просто скользил без трения?
7. Как можно использовать подобную модель для анализа движения реальных объектов, например, автомобилей или поездов?
8. Как изменилось бы поведение камня, если бы он не катился, а просто скользил без трения?
9. Можно ли использовать эту модель для расчета движения лавины? Какие параметры пришлось бы изменить?

3. Парадокс близнецов

Парадокс близнецов - это мысленный эксперимент, который является следствием теории относительности Эйнштейна. Он иллюстрирует, как время проходит с разной скоростью для двух объектов, движущихся с разными скоростями, даже если их первоначальные условия одинаковы.

3.1 Теоретическая часть

Описание парадокса

В парадоксе близнецов один из близнецов путешествует на космическом корабле со скоростью, близкой к скорости света, а другой остаётся на Земле. Согласно теории относительности, время для движущегося объекта замедляется относительно покоящегося.

Теория относительности и замедление времени

Замедление времени - это явление, при котором время для объекта, движущегося с большой скоростью, проходит медленнее, чем для объекта, который находится в покое относительно этого объекта. Это явление описывается специальной теорией относительности Альберта Эйнштейна.

Если один из близнецов движется с постоянной скоростью v относительно другого, то время для движущегося близнеца будет протекать медленнее, чем для того, кто остаётся на Земле. Это можно выразить через формулу замедления времени:

$$T = \frac{T_0}{\sqrt{1-\frac{v^2}{c^2}}} \tag{3.1}$$

где

T - время, прошедшее для движущегося близнеца (который путешествует на скорости v),

T_0 - время, прошедшее для неподвижного близнеца (который остаётся на Земле),

v - скорость движения близнеца,

c - скорость света в вакууме ($c = 3 \times 10^8$ м/с).

Эта формула показывает, что с увеличением скорости v время для движущегося объекта замедляется.

Интерпретация результатов

Для неподвижного близнеца (на Земле) время проходит нормально, в то время как для движущегося близнеца время замедляется.

Это явление происходит из-за того, что в теории относительности время и пространство являются взаимосвязанными и зависят от скорости движения объекта. Когда скорость близнеца приближается к скорости света, эффект замедления времени становится всё более заметным.

Теория относительности находит широкое применение в различных областях науки и техники. Специальная теория относительности (СТО) используется для изучения движения тел с высокими скоростями, включая скорости, близкие к скорости света. Она особенно актуальна в физике и астрономии, начиная с XX века. Общая теория относительности (ОТО) применима для исследования движения тел в гравитационных полях любой интенсивности, если квантовыми эффектами можно пренебречь. ОТО позволяет объяснить многие физические явления, такие как смещение перигелия Меркурия, и активно используется в астрофизике и системах глобального позиционирования (GPS), где учитываются релятивистские поправки для точного определения координат.

3.2 Практическая часть

Для визуализации парадокса близнецов с помощью MATLAB мы можем создать простую анимацию движения корабля вдоль оси X, где ось Y будет показывать разницу в возрасте между близнецами. Ниже приведён пример кода, который создаёт такую анимацию.

Пример MATLAB кода:

```
L = X; % Длина пути в световых годах
v = X; % Скорость корабля в долях от скорости света
c = 1;   % Скорость света

% Функция Лоренца
gamma = @(v) 1 / sqrt(1 - v^2);

% Разница в возрасте
delta_t = @(x) gamma(v) * x / c;

% Создаем фигуру и оси
figure('Position', [100, 100, 800, 600]);
h = animatedline('Color','b');
xlabel('Расстояние (световые годы)');
ylabel('Разница в возрасте (годы)');
title('Парадокс Близнецов');
grid on;

% Рассчитываем максимальную разницу в возрасте
max_delta_t = max(delta_t(L));

% Задаем увеличенный предел по оси Y
upper_y_limit = 2 * max_delta_t; % Увеличение в два раза

% Устанавливаем пределы по осям
```

```
axis([0 L 0 upper_y_limit]);

% Устанавливаем разметку по годам на оси Y
yticks(0:ceil(upper_y_limit)); % Подробная разметка по каждому
году

% Устанавливаем подробную разметку на оси X
xticks(0:L/20:L);

% Анимация
for x = linspace(0, L, 400)
   addpoints(h, x, delta_t(x));
   drawnow;
   pause(0.005);            % Пауза для плавности анимации
end
```

Пояснения к коду:

Параметры: Мы задаём длину пути L, общее время полёта T и скорость корабля v относительно скорости света c.

Функция Лоренца (gamma) используется для вычисления фактора замедления времени.

Разница в возрасте (delta_t) рассчитывается исходя из пройденного расстояния и скорости корабля.

Анимация: В цикле отрисовывается движение корабля по оси X и соответствующая разница в возрасте по оси Y.

Этот код создаст график, на котором мы сможем наблюдать, как изменяется возраст одного из близнецов относительно другого по мере движения корабля.

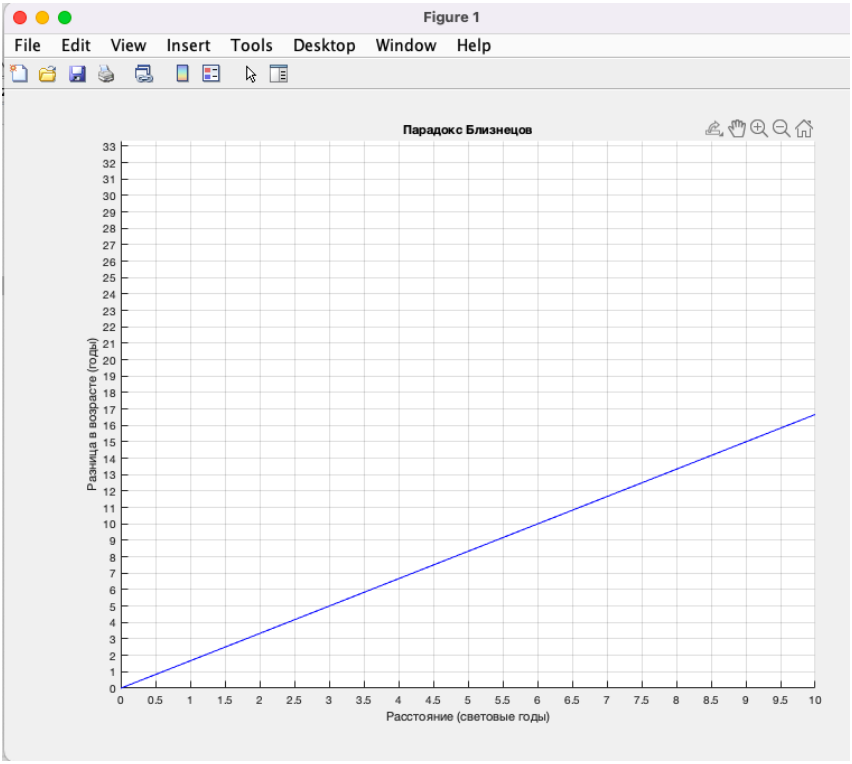

Рисунок 3.1 - Пример вывода решения

3.3 Задачи на практическую часть

1. Подставьте свои значения параметров: длины пути в световых годах, скорости корабля в долях от скорости света. И посмотрите, как изменится график.
3. Измените константу скорости света и посмотрите на другие результаты
4. Посмотрите в справочниках сколько нужно лететь до ближайших экзопланет и посмотрите результат подставив свои данные
5. Найдите в коде место, где можно изменить масштаб графика и поменяйте его.

3.4 Контрольные вопросы

1. Объясните суть парадокса близнецов. Почему возникает разница в возрасте между близнецами после путешествия?
2. Опишите, почему время течет медленнее для космонавта, путешествующего на высокой, близкой к скорости света.
3. Как релятивистский фактор γ влияет на разницу в возрасте между близнецами?
4. Приведите пример реального эксперимента, подтверждающего замедление времени.

4. Путешествие Одиссея

После долгих лет странствий Одиссей плывет домой в Итаку на своем корабле. Однако его путь лежит через бурные воды, где высокие волны могут затопить судно. Ваша задача – правильно подобрать параметры модели движения волны, чтобы Одиссей благополучно достиг родного берега.

4.1 Теоретическая часть

Волны на поверхности воды представляют собой механические колебания, которые можно описать с помощью синусоидальной функции:

$$y(x,t) = A \sin((2\pi/\lambda)(x - ct)) \quad (4.1)$$

где:

A – амплитуда волны (максимальная высота относительно средней линии),

λ – длина волны (расстояние между двумя гребнями),

c – скорость распространения волны,

x – координата вдоль поверхности воды,

- t – время.

Корабль испытывает воздействие волны, и его устойчивость зависит от величины амплитуды волны: при больших значениях A риск затопления увеличивается.

4.2 Практическая часть

Пример MATLAB кода:

```
x = 1:.1:100;          % Горизонтальное расстояние
lambda = X;            % Длина волны, подставьте свои значения
c = X;                 % Скорость волны, подставьте свои значения
A = X;                 % Амплитуда волны, подставьте свои значения
```

```
% Инициализация графика
figure;
hold on;
xlabel('Distance (m)');
ylabel('Wave Height (m)');
box off;
axis([0 100 -5 5]);

% Свойства корабля
ship_x = 75;            % Положение корабля
ship_w = 8;             % Длина корабля
ship_h = 0.5;           % Высота корабля
mast_h = 3;             % Высота мачты
sail_w = 3;             % Ширина паруса
is_sinking = false;     % Индикатор затопления
sink_depth = 0;         % Глубина погружения
ship_direction = 1;     % Направление корабля (1 - вправо, -1 - влево)

% Анимация
for t = 0:.1:20

    % Вычисление высоты волны в каждый момент времени
    y = A * sin((??????????????)*(x-c*t));

    % Отображение волны
    hwave = fill([x,100,0],[y,-5,-5],[.01 .44 .61],'edgecolor','b');

    % Высота корабля на x = 75
    y_ship = y(x == ship_x) - sink_depth;

    % Условие затопления: если волна слишком высокая, корабль начнет тонуть
```

```
    if ~is_sinking && y_ship < -1
        is_sinking = true;
    end

    % Если корабль тонет, он постепенно уходит вниз
    if is_sinking
        sink_depth = sink_depth + 0.05;
        if sink_depth > 5  % Когда корабль уходит под воду полностью, останавливаем анимацию
            break;
        end
    end

    % Определяем форму корпуса корабля (длинный прямоугольник)
    hull_x = [ship_x-ship_w/2, ship_x+ship_w/2, ship_x+ship_w/2, ship_x-ship_w/2];
    hull_y = [y_ship, y_ship, y_ship+ship_h, y_ship+ship_h];

    % Рисуем корабль
    hship = fill(hull_x, hull_y, [.5 .2 .1], 'EdgeColor', 'k');

    % Добавляем мачту
    mast_x = [ship_x, ship_x];
    mast_y = [y_ship+ship_h, y_ship+ship_h+mast_h];
    hmast = plot(mast_x, mast_y, 'k', 'LineWidth', 2);

    % Добавляем парус в направлении движения корабля
    sail_x = [ship_x, ship_x - sail_w, ship_x];
    sail_y = [y_ship+ship_h+mast_h, y_ship+ship_h+mast_h/2, y_ship+ship_h];
    hsail = fill(sail_x, sail_y, [1 1 1], 'EdgeColor', 'k');
```

```
    % Отображение кадра
    drawnow;
    pause(0.1);
    delete([hwave hship hmast hsail]);
end

% Если корабль затонул, отобразим его на дне
if is_sinking
    text(ship_x, -4.5, 'Корабль затонул!', 'HorizontalAlignment', 'center', 'FontSize', 12, 'FontWeight', 'bold', 'Color', 'r');
else
    % Сообщение об успехе
    text(ship_x, 4.5, 'Одиссей справился!', 'HorizontalAlignment', 'center', 'FontSize', 12, 'FontWeight', 'bold', 'Color', 'g');
end
```

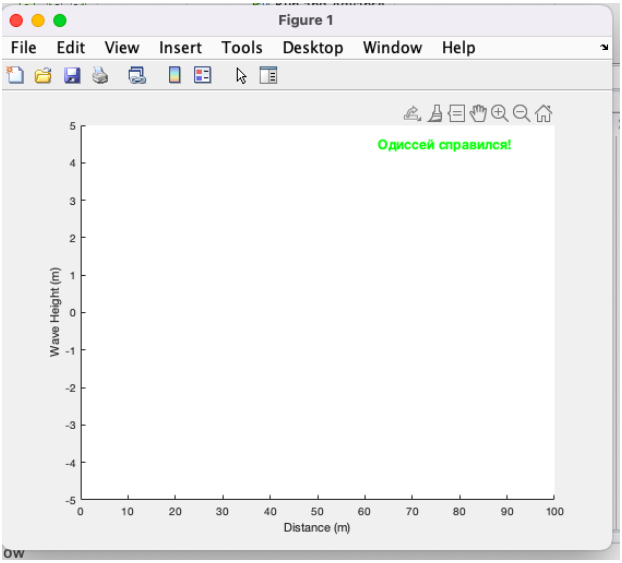

Рисунок 4.1 Пример вывода решения «Одиссей справился!»

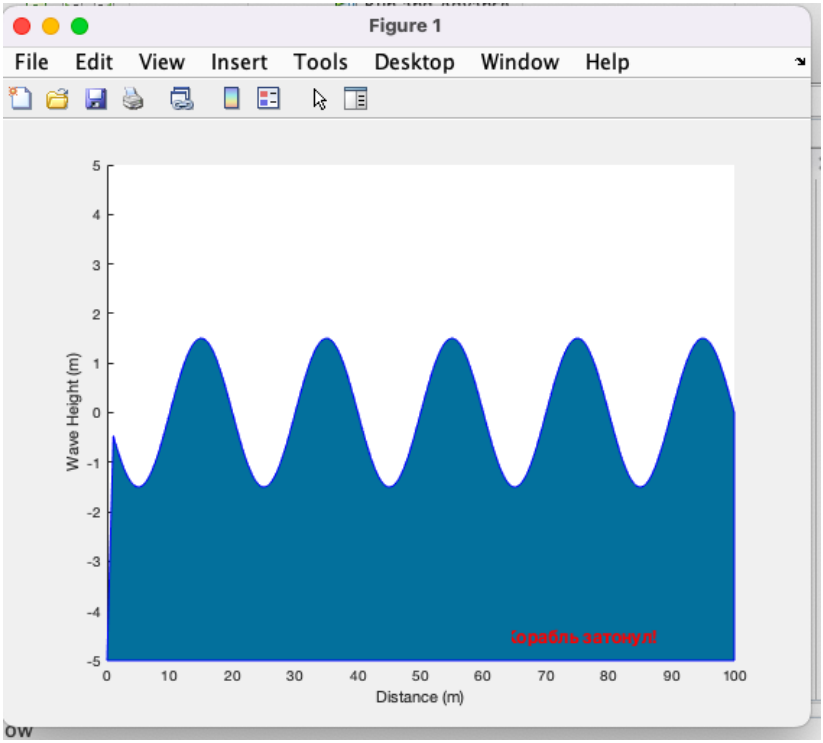

Рисунок 4.2 Пример вывода решения «Корабль затонул!»

4.3 Задачи на практическую часть

1. Дополните строчку кода

% Вычисление высоты волны в каждый момент времени

вместо ? добавьте формулу волнового числа K

y = A * sin((??????????????)*(x-c*t));

2. Подберите параметры волны (длину, скорость, амплитуду) таким образом, чтобы Одиссей успешно закончил свое путешествие и не затонул.

3. Изучите как работает анимация корабля, какие строчки кода за это отвечают, попробуйте поменять параметры корабля и посмотрите, что получится.

4.4 Контрольные вопросы

1. Как влияет амплитуда волны на вероятность затопления корабля?
2. Почему корабли устойчивее на длинных волнах, чем на коротких?
3. Как можно рассчитать вероятность затопления корабля при заданных параметрах волны?
4. Что такое волновое число k?

5. Танец электрона

Представьте, что вы проводите исследование траектории электрона в классическом атоме водорода. В отсутствие внешнего магнитного поля электрон движется по эллиптической орбите, обусловленной центральным кулоновским притяжением между электроном и протоном. Однако при наложении магнитного поля на систему добавляется дополнительная сила Лоренца, которая изменяет траекторию электрона, вызывая эффект, напоминающий движение в вращающейся системе координат.

5.1 Теоретическая часть

Классическая модель атома водорода

В классической модели атома водорода электрон рассматривается как точечная частица, движущаяся под влиянием кулоновской силы, действующей со стороны протона. Кулоновская сила описывается законом обратных квадратов:

$$\mathbf{F} = -\frac{1}{r^2}\,\hat{\mathbf{r}} \tag{5.1}$$

где r – расстояние между частицами, а знак минус указывает на притяжение.

Уравнения движения:
При отсутствии внешних полей уравнения движения записываются в виде системы дифференциальных уравнений, где первые три уравнения описывают изменение координат, а следующие – изменение скоростей (ускорения), зависящие от кулоновской силы.

Влияние магнитного поля

Сила Лоренца:
При наложении магнитного поля на систему, движущаяся заряженная частица испытывает дополнительную силу Лоренца, которая определяется по формуле:

$$\mathbf{F}_{\text{Лоренца}} = q\,(\mathbf{v} \times \mathbf{B}) \tag{5.2}$$

Если магнитное поле направлено вдоль оси z (то есть $\mathbf{B} = (0,0,B)$), то компоненты силы будут:

$$\begin{aligned} F_x &= qB\,v_y \\ F_y &= -qB\,v_x \\ F_z &= 0 \end{aligned} \tag{5.3}$$

В данном коде параметр ω связан с величиной магнитного поля, а дополнительные слагаемые в уравнениях ускорения моделируют влияние силы Лоренца с учётом выбранных безразмерных величин.

Полная система уравнений движения

Координаты:

$$\frac{dx}{dt} = v_x, \quad \frac{dy}{dt} = v_y, \quad \frac{dz}{dt} = v_z \tag{5.4}$$

Ускорения:

$$\frac{dv_x}{dt} = -\frac{x}{r^3} + 2\omega\,v_y,$$

$$\frac{dv_y}{dt} = -\frac{y}{r^3} - 2\omega\,v_x, \tag{5.5}$$

$$\frac{dv_z}{dt} = -\frac{z}{r^3}, \tag{5.6}$$

где

$$r = \sqrt{x^2 + y^2 + z^2}. \quad (5.7)$$

Преобразование системы координат

Для визуализации динамики движения применяется преобразование координат в вращающуюся систему:

$$X_1 = X\cos(\omega T) - Y\sin(\omega T)$$

$$Y_1 = X\sin(\omega T) + Y\cos(\omega T) \quad (5.8)$$

Это преобразование помогает увидеть, как движение электрона выглядит с точки зрения системы, вращающейся относительно лабораторной. Оно «выключает» эффект поворота, вызванного магнитным полем, и позволяет более наглядно анализировать влияние кулоновской и магнитной сил.

Численное решение уравнений

В MATLAB для решения системы ОДУ используется функция ode45 – адаптивный метод Рунге–Кутты. Он позволяет с высокой точностью вычислять траекторию частицы даже при наличии нелинейных членов (например, $-1/r^3$) и сил, зависящих от скорости.

Преимущества метода:

Адаптивный шаг интегрирования, позволяющий автоматически подбирать шаг при изменении характера движения.

Высокая точность, особенно важная для задач с сильными нелинейностями.

Физический смысл параметров задачи

ω**:**
Параметр, связанный с величиной магнитного поля. При

увеличении ω усиливается влияние магнитного поля, что приводит к более заметным эффектам поворота траектории.

Начальные скорости (v_{tang} и v_{0z}):
Определяют начальное вращательное и продольное движение электрона. Изменяя их, можно моделировать различные типы орбит – от замкнутых до хаотичных.

t_{fin}:
Время интегрирования, задающее длительность наблюдения траектории.

Анализ результатов моделирования

Первый график:
Отображает траекторию электрона в стационарной системе координат, демонстрируя влияние кулоновской силы и магнитного поля.

Второй график:
Построен после преобразования координат, что позволяет увидеть траекторию в системе. Это помогает выделить те компоненты движения, которые непосредственно связаны с воздействием магнитного поля.

5.2 Практическая часть

Пример MATLAB кода:

```
% Классический атом водорода в магнитном поле. Решение дифференциального уравнения
function [y] = maggy(~, x, ~, om)
   if nargin < 1
      tfin = input('tfin=');
      om   = input('omega Lor.=');
```

```
v0z  = input('v0z=');
v0y  = input('vtang=');
A0 = [1 0 0 0 v0y v0z];
options = odeset('RelTol', 1e-6, 'AbsTol', 1e-6*[1 1 1 1 1 1]);
[T, A] = ode45('E2_2', [0 tfin], A0, options, om);

if (v0z == 0) && (A0(3) == 0)
   r  = sqrt(A(:,1).^2 + A(:,2).^2);
   fi = atan2(A(:,2), A(:,1));
   polarplot(fi, r);
   s = strcat('Omega Lorenz = ', num2str(om));
   text(1.2*cos(pi/3), 1.2*sin(pi/3), s);
else
   % Построение первого графика
   X = A(:,1);
   Y = A(:,2);
   Z = A(:,3);
   plot3(X, Y, Z, 'color', [1 1 1]);
   xlabel('x'); ylabel('y'); zlabel('z');
   hold on;
   XX = [X(1); X(2)];
   YY = [Y(1); Y(2)];
   ZZ = [Z(1); Z(2)];
   pl = line(XX, YY, ZZ, 'color', 'k');
   [m, ~] = size(X);
   for i = 3:m
      XX = [X(i-1); X(i)];
      YY = [Y(i-1); Y(i)];
      ZZ = [Z(i-1); Z(i)];
      set(pl, 'xdata', XX, 'ydata', YY, 'zdata', ZZ);
      drawnow;
   end
   plot3(X, Y, Z, 'color', 'k');
```

```
        grid on;

        % Построение второго графика (вращение системы координат)
        X1 = X .* cos(om*T) - Y .* sin(om*T);
        Y1 = X .* sin(om*T) + Y .* cos(om*T);
        figure;
        plot3(X1, Y1, Z, 'color', [1 1 1]);
        xlabel('x'); ylabel('y'); zlabel('z');
        hold on;
        XX = [X1(1); X1(2)];
        YY = [Y1(1); Y1(2)];
        ZZ = [Z(1); Z(2)];
        pl = line(XX, YY, ZZ, 'color', 'k');
        for i = 3:m
           XX = [X1(i-1); X1(i)];
           YY = [Y1(i-1); Y1(i)];
           ZZ = [Z(i-1); Z(i)];
           set(pl, 'xdata', XX, 'ydata', YY, 'zdata', ZZ);
           drawnow;
        end
        plot3(X1, Y1, Z, 'color', 'k');
        grid on;
     end
  else
     y(1) = x(4);
     y(2) = x(5);
     y(3) = x(6);
     r3 = ((x(1))^2 + (x(2))^2 + (x(3))^2)^1.5;
     y(4) = -x(1)/r3 + 2*om*x(5);
     y(5) = -x(2)/r3 - 2*om*x(4);
     y(6) = -x(3)/r3;
     y = y';
```

```
    end
end
```

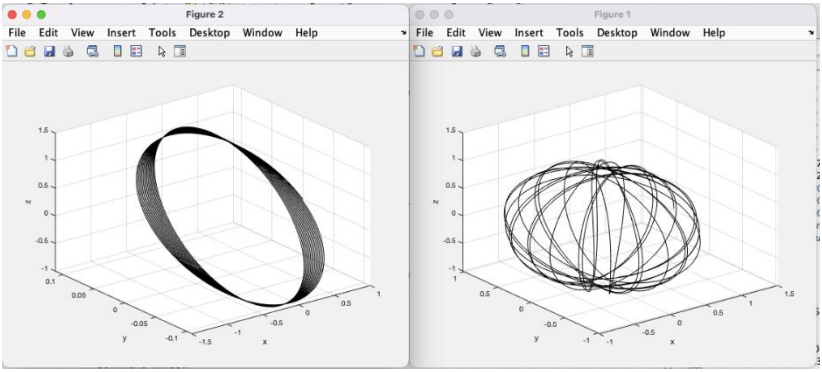

Рисунок 5.1 - Пример вывода решения

пример интересных рисунков для v=0, t=10
om=1.125 (4), 0.9877 (3), 0.7943 (2), 0.523 (4')
t=500 om=0.05 vz=1 vyan=0
или t=250 om=0.03 vz=0.3 vtan=0.3

5.3 Задачи на практическую часть

1. Подставьте свои значения параметров и посмотрите на результат.
tfin=???
omega Lor.=??
v0z=?
vtang=?

2. Исследуйте случай плоского движения в случае сильного магнитного поля, когда omega Lor >> 1

5.4 Контрольные вопросы

1. Какие силы действуют на электрон в классической модели атома водорода при отсутствии внешнего магнитного поля?
Как изменяется система сил при введении магнитного поля?
2. Объясните, что такое сила Лоренца и как она математически описывается для частицы, движущейся в магнитном поле, направленном вдоль оси z.
3. Как изменяется динамика движения электрона при увеличении или уменьшении параметра ω?
4. Как изменение начальных скоростей v0z и vtang влияет на форму орбиты электрона?

6. Ультрафиолетовая катастрофа

Ультрафиолетовая катастрофа - это историческое название проблемы, возникшей при описании спектра излучения абсолютно чёрного тела с помощью классической физики. Согласно классическому закону Рэлея–Джинса, чем короче длина волны, тем сильнее должна возрастать интенсивность излучения, что приводило к парадоксальному результату: бесконечной энергии в области ультрафиолетовых волн. Однако эксперименты показывали, что реальный спектр «обрывается» и не уходит в бесконечность. Разрешить это противоречие удалось Максу Планку в 1900 году, когда он предположил, что энергия излучается порциями (квантами), а не непрерывно. Это стало одним из ключевых шагов в развитии квантовой механики и позволило корректно описать спектр абсолютно чёрного тела.

6.1 Теоретическая часть

Классическая теория и закон Рэлея–Джинса

Согласно закону Рэлея–Джинса, спектральная плотность излучения абсолютно чёрного тела увеличивается пропорционально квадрату частоты. Это означает, что при уменьшении длины волны (увеличении частоты) интенсивность излучения должна возрастать неограниченно, что приводит к расходимости интеграла и предсказанию бесконечной энергии излучения в ультрафиолетовой области. Это явление назвали ультрафиолетовой катастрофой, так как оно не соответствовало экспериментальным данным, согласно которым интенсивность излучения уменьшается при малых длинах волн.

Гипотеза Планка

В 1900 году Макс Планк предложил новый подход, который позволил решить проблему ультрафиолетовой катастрофы. Он предположил, что энергия излучения не является непрерывной, а квантуется, то есть излучается дискретными порциями (квантами) с энергией:

$$E = h\nu \qquad (6.1)$$

где h - постоянная Планка, ν - частота излучения.
Благодаря этому предположению была выведена формула Планка, которая корректно описывает спектральное распределение излучения абсолютно чёрного тела:

$$I(\lambda) = (2\pi hc^2) / (\lambda^5(\exp(hc / (\lambda k\_BT)) - 1)) \qquad (6.2)$$

Эта формула показала, что интенсивность излучения уменьшается при малых длинах волн, что соответствует экспериментальным наблюдениям.

Историческая и научная значимость

1. Разрешение противоречия: Формула Планка устраняет ультрафиолетовую катастрофу и соответствует экспериментальным данным.
2. Рождение квантовой механики: Введение квантования энергии стало основой для дальнейшего развития квантовой физики.
3. Практическое применение: Квантовая теория излучения нашла применение в астрофизике, термодинамике и других науках.

Таким образом, ультрафиолетовая катастрофа была одной из первых серьёзных проблем, которые классическая физика не могла объяснить. Решение этой проблемы положило начало

развитию квантовой механики и открыло новые горизонты в понимании природы света и материи.

6.2 Практическая часть

Пример MATLAB кода:

```
% Задание параметров
T = XXX;                        % температура в Кельвинах
lambda_nm = linspace(200, 3000, 500);   % диапазон длин волн в нанометрах [200 нм - 3000 нм]
lambda = lambda_nm * 1e-9;              % перевод в метры
kB = X.XXXXXXX;                 % постоянная Больцмана [Дж/К]
h = X.XXXXXXXX;                 % постоянная Планка [Дж·с]
c = XXX;                        % скорость света [м/с]

% Вычисление спектральной интенсивности по формуле Планка (Вт/(м^2·м))
I_planck = (2*pi*h*c^2) ./ (lambda.^5) ./ (exp(h*c./(lambda*kB*T)) - 1);

% Вычисление спектральной интенсивности по Рэйли-Джинсу (Вт/(м^2·м))
I_RJ = (2*pi*c*kB*T) ./ (lambda.^4);

% Перевод из Вт/(м^2·м) в Вт/(м^2·нм)
I_planck_nm = I_planck * 1e-9;
I_RJ_nm = I_RJ * 1e-9;

% Создание окна и задание логарифмической шкалы по оси Y
figure;
set(gca, 'Yscale', 'log');
```

```
% Создание анимированных линий (задаём имена для легенды)
h_RJ_line = animatedline('Color','r','LineWidth',2,
'DisplayName','Рэйли-Джинс');
h_planck_line = animatedline('Color','b','LineWidth',2,
'DisplayName','Планк');

xlabel('Длина волны, нм');
ylabel('Спектральная интенсивность, Вт/(м^2·нм)');
title('Сравнение спектральной интенсивности излучения
абсолютно чёрного тела');

% Предварительно создаём легенду (она обновится, когда добавим
другие объекты)
legend('show');

% Задаём пределы осей (ось X от 200 до 3000 нм)
axis([200 3000 1e-15 1e12]);
hold on;

% Устанавливаем шаг по оси X: 100 нм
xticks(200:100:3000);

% Выделение ультрафиолетовой области (200–400 нм)
line([200 200], [1e-15, 1e12], 'Color', 'g', 'LineStyle', '-',
'LineWidth', 2, ...
   'DisplayName', 'УФ область');
line([400 400], [1e-15, 1e12], 'Color', 'g', 'LineStyle', '-',
'LineWidth', 2, ...
   'DisplayName', 'УФ область');
text(210, 1e11, 'УФ область', 'Color', 'g', 'FontSize', 12);

% Анимация: поочерёдное добавление точек для каждой модели
for i = length(lambda):-1:1
```

```
    addpoints(h_RJ_line, lambda_nm(i), I_RJ_nm(i));
    addpoints(h_planck_line, lambda_nm(i), I_planck_nm(i));
    drawnow;
    pause(0.01);  % регулировка скорости анимации
end
```

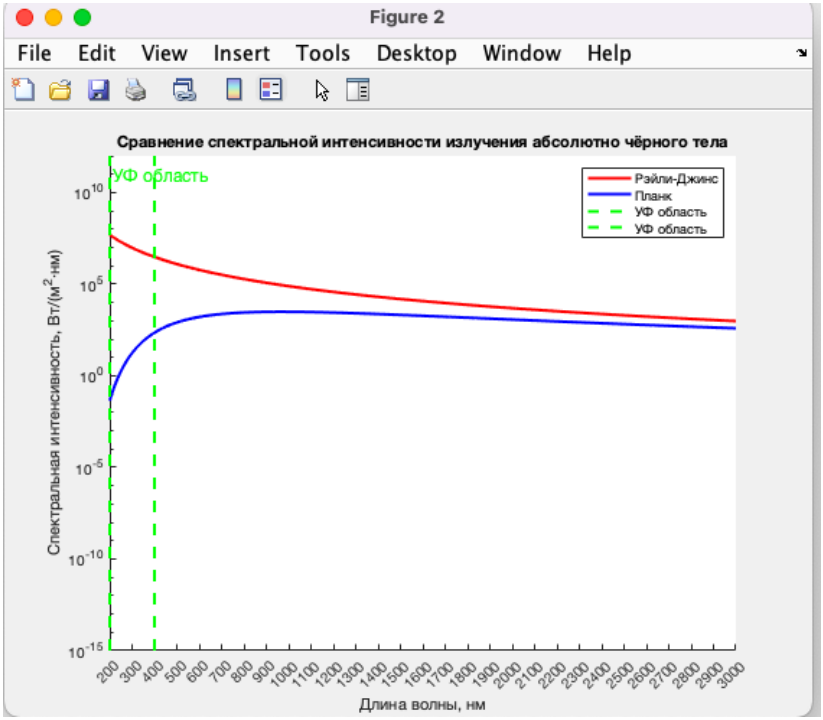

Рисунок 6.1 Пример вывода решения при T = 3000K

6.3 Задачи на практическую часть

1. Подставьте правильные значения: постоянной Больцмана, постоянной Планка, скорости света
2. Подставьте в код температуру в Кельвинах и посмотрите на результат. Повторите с разными значениями.

6.4 Контрольные вопросы

1. Что такое ультрафиолетовая катастрофа? Почему она возникла?
2. В чём суть закона Рэлея–Джинса? Какую проблему он создавал в классической физике?
3. Почему классическая физика предсказывала бесконечную интенсивность излучения при малых длинах волн?
4. В чём заключается основная идея гипотезы Планка о квантовании энергии?
5. Какие фундаментальные физические принципы были пересмотрены в результате решения проблемы ультрафиолетовой катастрофы?

7. Основы Simulink

Simulink - это мощная среда моделирования и симуляции динамических систем, интегрированная с MATLAB. Одной из её ключевых возможностей является моделирование электрических и электронных схем. Это позволяет не только визуализировать работу электрических цепей, но и анализировать поведение системы до её физической реализации.

7.1.1 Теоретическая часть. Зарядка и разрядка конденсатора

Основы работы конденсатора

Конденсатор - это пассивный электрический элемент, способный накапливать электрический заряд. При подключении к источнику напряжения, между его обкладками накапливается заряд, создавая электрическое поле.

Основное уравнение для напряжения на конденсаторе:

$$i(t) = C\frac{dV(t)}{dt} \tag{7.1}$$

где

$i(t)$ - ток через конденсатор,

C - ёмкость (в фарадах),

$V(t)$ - напряжение на конденсаторе.

Ионистор (суперконденсатор) - это устройство для накопления энергии, сочетающее свойства конденсатора и аккумулятора.

Он работает как обычный конденсатор, но имеет намного большую ёмкость (в миллифарадах и даже фарадах) за счёт

использования двойного электрического слоя и специальных электродов (обычно на основе углеродных наноматериалов).

Преимущества и недостатки ионисторов

Преимущества:

Очень быстрая зарядка и разрядка. Заряжаются за секунды, в отличие от аккумуляторов, которые требуют минут и часов.
Долговечность. Выдерживают более 1 миллиона циклов заряд/разряд без существенной потери ёмкости.
Высокая мощность. Могут быстро отдавать или принимать большую мощность - идеально для импульсных нагрузок (торможение, вспышки, стартеры).
Низкий внутренний импеданс. Меньше тепловых потерь и высокая эффективность при быстрой передаче энергии.
Безопасность. Не взрываются и не воспламеняются, как литиевые аккумуляторы.
Экологичность. Не содержат тяжёлых металлов и токсичных веществ.
Работа в широком температурном диапазоне. Могут работать в условиях холода или жары, где аккумуляторы выходят из строя.

Недостатки:

Низкая энергоёмкость. Хранят меньше энергии, чем аккумуляторы - не подходят для длительной автономной работы.
Низкое рабочее напряжение. Обычно 2.7–5 В на один элемент. Для работы с более высокими напряжениями требуется сборка нескольких последовательно (что усложняет схему).
Саморазряд. Быстрее теряют заряд во времени по сравнению с аккумуляторами - не подходят для долгосрочного хранения энергии.
Размер и масса. Чтобы накопить столько же энергии, сколько аккумулятор, требуется больше пространства.

Цена за ёмкость.Хотя по стоимости за цикл ионисторы выгоднее, начальная цена за ватт-час энергии - выше.

Процесс зарядки

Когда конденсатор подключён к источнику постоянного напряжения через резистор, он начинает **заряжаться**. Напряжение на конденсаторе изменяется по экспоненциальному закону:

$$V_C(t) = V_{\text{ист}}\left(1 - e^{-t/RC}\right) \quad (7.2)$$

где

$V_{\text{ист}}$ - напряжение источника,

R - сопротивление цепи,

C - ёмкость конденсатора,

t - время.

Конденсатор теоретически никогда не достигает ровно 100% заряда**,** так как экспоненциальная функция приближается к напряжению асимптотически. Поэтому на практике считается, что конденсатор полностью зарядился**,** когда напряжение достигло:

Таблица 7.1 - Зависимость степени заряда конденсатора от времени (в постоянных времени τ

Процент от Vист	Время
~63% 1τ	~63% 1τ
~86% 2τ	~86% 2τ
~95% 3τ	~95% 3τ
~98% 4τ	~98% 4τ
~99% 5τ (считается полным зарядом)	~99% 5τ *(считается полным зарядом)*

Процесс разрядки

Если заряженный конденсатор отключить от источника и подключить на нагрузку (например, через резистор), он начнёт **разряжаться**. Напряжение при этом убывает по формуле:

$$V_C(t) = V_0 e^{-t/RC} \quad (7.3)$$

где V_0 - начальное напряжение на конденсаторе.

Моделирование в Simulink

Для моделирования процессов зарядки и разрядки в **Simulink с** использованием **Simscape Electrical**, применяются следующие компоненты:

Источник постоянного напряжения (**DC Voltage Source**),

Резистор (**Resistor**),

Конденсатор (**Capacitor**),

Переключатель (**Ideal Switch**) или ключ (**Controlled Switch**) - для переключения между зарядкой и разрядкой,

Заземление (**Electrical Reference**),

Измерительные блоки (**Voltage Sensor**, **Current Sensor**),

Визуализация (**Scope**).

Принцип построения модели

Собирается RC-цепь: источник - резистор - конденсатор.

Для наблюдения процесса зарядки и разрядки реализуется управление переключателем.

Подключаются измерительные блоки и осциллографы для отображения напряжения и тока во времени.

Устанавливается временной интервал моделирования, позволяющий проследить полную зарядку и разрядку.

7.1.2 Практическая часть. Зарядка и разрядка конденсатора

«Моделирование зарядки и разрядки конденсатора в Simulink»

Цель: Построить модель зарядки и разрядки конденсатора в Simulink с визуализацией на осциллографе (Scope).

Построение модели зарядки конденсатора

1. Откройте MATLAB и в командном окне введите Simulink, либо нажмите на кнопку Simulink в меню HOME
2. Откроется Simulink Start Page.
3. Нажмите Blank Model, чтобы создать новую модель.
4. Дважды кликните в рабочем поле и начните вводить:
   Series RLC Branch, затем выберите и перетащите блок в модель.
5. Дважды кликните по блоку и:
   Выберите Branch type на R
   Установите Measurements → Branch Voltage and Current
   Подтвердите действия, у вас должен поменяться рисунок.
6. Скопируйте и вставьте такой же блок
   Выберите Branch type на C
   Установите ёмкость на 1e-3 mF (для более наглядной зарядной кривой);
   Поставьте галочку Set the initial capacitor voltage
   Задайте начальное напряжение на конденсаторе 0 В (чтобы показать зарядку с нуля).
   Подтвердите действия
   Поверните блок (Ctrl + R).
7. Сделайте соответствующее соединение между блоками.
8. Добавьте блок DC Voltage Source (второе всплывающее окно)
9. Соедините источник с RC-цепью. «Замкните» модель.

10. Добавьте блок Voltage Measurement и подключите его параллельно конденсатору.

11. Добавьте блок Scope и соедините его с выходом измерителя напряжения.

12. Введите и добавьте блок powergui - он обязателен при использовании элементов библиотеки powerlib.

13. Установите время моделирования на 0.02 сек, так как зарядка происходит быстро, и на большом интервале кривая будет неразличима.

14. Нажмите Run (Запуск модели).

15. Откройте Scope кликнув 2 раза

Вы увидите кривую зарядки конденсатора.

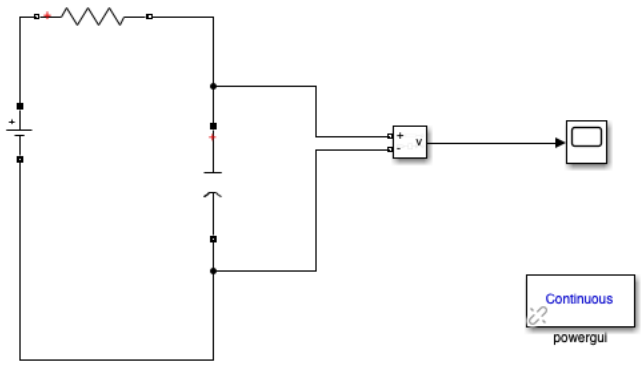

Рисунок 7.1 - Пример схемы зарядки конденсатора

Построение модели разрядки конденсатора

1. Удалите источник питания из схемы.
2. Соедините цепь так, чтобы конденсатор разряжался через резистор.

3. Дважды кликните по конденсатору и установите начальное напряжение: 10 В - это моделирует уже заряженный конденсатор.
4. Нажмите Run.
5. Откройте Scope

Вы увидите кривую разрядки конденсатора.

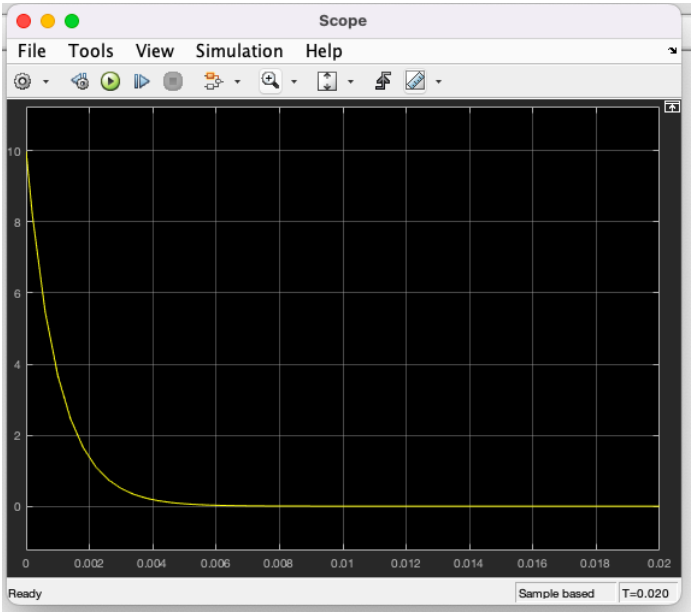

Рисунок 7.2 - Пример осциллограммы разрядки конденсатора

Таким образом, вы смоделировали два ключевых переходных процесса: **зарядку** и **разрядку** конденсатора, что позволяет визуализировать временные зависимости напряжения в RC-цепи.

7.1.3. Задачи на практическую часть. Зарядка и разрядка конденсатора

1. Найдите, за какое время напряжение упадёт примерно до 40% от начального (это соответствует $\tau = RC$). Значения выберите сами. График осциллографа должен совпасть с вычислениями.
2. Найдите время полной зарядки конденсатора.

7.1.4. Контрольные вопросы. Зарядка и разрядка конденсатора

1. Что такое постоянная времени $\tau\tau$ и как она рассчитывается?
2. Через какое время напряжение на конденсаторе достигает примерно 63% от максимального?
3. Почему конденсатор не может зарядиться до 100% мгновенно?
4. Сколько времени примерно нужно для "полной" зарядки или разрядки конденсатора?
5. Для чего используется блок Scope?
6. Какие блоки необходимы для моделирования RC-цепи в Simulink?
7. В чем преимущества и недостатки ионисторов?

7.2.1 Теоретическая часть. Фильтр низких частот

Фильтр низких частот (ФНЧ, Low Pass Filter, LPF) - это электронная цепь, которая пропускает сигналы с низкой частотой и подавляет (ослабляет) сигналы с высокой частотой.
Он используется в аналоговой и цифровой технике для выделения полезных низкочастотных сигналов, подавления шумов, сглаживания и других целей.

Принцип работы:
Низкие частоты проходят практически без изменений.

Высокие частоты "уходят" через конденсатор на землю, ослабляя сигнал на выходе.

Конденсатор работает как переменный "короткозамыкающий" путь для высоких частот.

Частота среза

Частота среза - это частота, при которой сигнал на выходе уменьшается до **70.7% от максимального значения** (или –3 дБ).

$$f_c = \frac{1}{2\pi RC} \tag{7.4}$$

На этой частоте фильтр начинает активно "резать" сигнал.

При частоте **меньше** f_c - сигнал проходит почти без искажений.
При **частоте =** f_c - амплитуда выхода ≈ 0.707 от входа.
При **частоте >** f_c - выход быстро ослабевает.

Где применяются ФНЧ?

Сглаживание сигнала после АЦП (аналогово-цифрового преобразования);

Удаление высокочастотных шумов;

Аудиотехника (фильтрация высоких частот);

Измерительные системы (подавление помех);

Импульсные источники питания (фильтрация пульсаций).

7.2.2 Практическая часть. Фильтр низких частот

1. Откройте MATLAB, затем откройте Simulink.
2. Создайте новую модель.
3. Нажмите на пустое место и введите Series RLC Branch.
4. Скопируйте ещё один такой же блок.
5. Добавьте источник переменного напряжения AC Voltage Source.
6. Соедините все элементы между собой.

7. Измените один из блоков Series RLC Branch на **резистор R** и установите значение сопротивления **1 кОм**.
8. Измените второй блок Series RLC Branch на **конденсатор C** и задайте ёмкость **2.2 мкФ**.
9. Добавьте блок измерения напряжения Voltage Measurement и подключите его параллельно с конденсатором.
10. Добавьте блок Scope для отображения сигналов.
11. Добавьте блок powergui - он необходим для моделирования с компонентами Simscape Electrical.
Теперь обратим внимание на расчёты.
Мы будем наблюдать поведение фильтра при разных частотах.
Сначала - при **50 Гц**.
12. Установите время моделирования Stop time на **0.1 сек**.
13. Задайте частоту источника **50 Гц**.
14. Добавьте блок Mux (для объединения сигналов в один вектор).
15. Скопируйте ещё один блок измерения напряжения Voltage Measurement и подключите его параллельно источнику (на входе).
Теперь нажмите кнопку **Run**.
Откройте Scope и нажмите **Auto Scale**.
Синяя линия - это **входной сигнал**, желтая - **отфильтрованный выход**.
Как видно, при низкой частоте сигнал практически полностью проходит через фильтр.
Теперь зададим частоту **454.545 Гц**.
Как видно, проходит только малая часть сигнала - примерно **10–15 В**.
Цепь эффективно ослабляет компоненту высокого диапазона.
Теперь зададим частоту **1000 Гц**, снова запустим симуляцию и откроем Scope.
Как видно, проходит совсем небольшой сигнал.
Это показывает, что наша схема пропускает низкие частоты и подавляет высокие частоты.

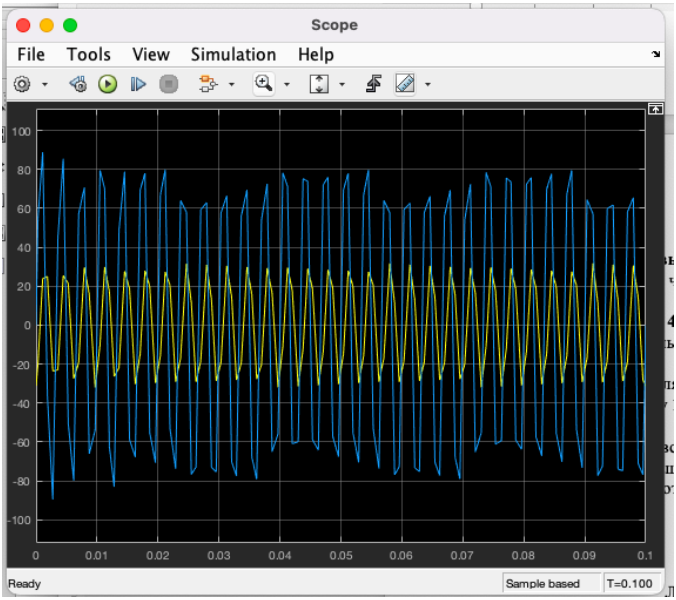

Рисунок 7.3 - Пример осциллограммы отфильтрованного сигнала

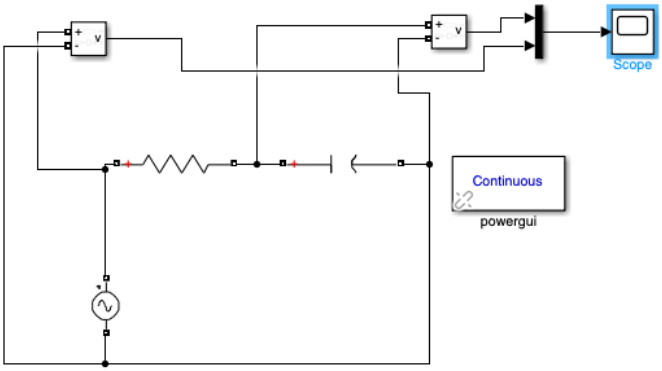

Рисунок 7.4 - Пример схемы

7.2.3. Задачи на практическую часть. Фильтр низких частот

1. Вычислить частоту среза по формуле (7.4) и подтвердить её экспериментально, подставив свои значения RC. Провести моделирование при частотах: 100 Гц, 500 Гц, 1000 Гц

7.2.4. Контрольные вопросы. Фильтр низких частот

1. Что такое фильтр низких частот? Какова его основная функция?
2. Что такое частота среза? Как её можно вычислить?
3. Что изменится в поведении фильтра, если увеличить ёмкость конденсатора?
4. Как ФНЧ может быть использован в реальных устройствах?

8. Второй закон

В Солнечную систему входит загадочный корабль - *Оумуамуа*. Его обшивка покрыта следами микрометеоритов, а внутренние отсеки - мертвой тишиной. При расшифровке бортового журнала становится ясно: корабль создан разумной цивилизацией, все пассажиры находились в состоянии криосна.

Однако, функционирует лишь ИИ корабля, который сообщает: «Все параметры системы были в норме. Энергоснабжение, температура, радиационная защита - всё было под контролем.» Люди анализируют данные и приходят к выводу: корабль был спроектирован идеально - но с одной ошибкой: не была учтена неумолимо растущая энтропия замкнутой системы.

Даже в самой совершенной системе локальные сбои, деградация информации и нестабильность компонентов накапливаются. Идеальный порядок уступает месту хаосу.

ИИ корабля задаёт вопрос:

– Как называется эта формула, которая разрушила наш идеальный корабль?

8.1 Теоретическая часть

Энтропия в термодинамике

Энтропия - это фундаментальное понятие в термодинамике, обозначающее меру беспорядка или степень рассеяния энергии в системе. Она также интерпретируется как измерение необратимости процессов.

Если энергия в системе становится менее доступной для совершения работы, мы говорим, что энтропия увеличивается.

Второй закон термодинамики

Во всех замкнутых (изолированных) системах энтропия со временем не убывает.
Это означает, что самопроизвольные процессы в природе всегда происходят в направлении увеличения энтропии. Например:
Горячее тело остывает, но не нагревается само.
Газ равномерно распределяется по объёму, но не собирается обратно в угол сосуда.
Этот закон объясняет, почему время воспринимается как направленное: потому что энтропия растёт.

Статистическое определение энтропии

Физик Людвиг Больцман связал макроскопическое понятие энтропии с микросостояниями системы:

$$S = k \cdot \ln(W) \tag{8.1}$$

где
S - энтропия,
k - постоянная Больцмана (приблизительно 1.38×10^{-23} Дж/К),
W - число возможных микросостояний, соответствующих одному макросостоянию.
Чем больше вариантов внутреннего устройства системы при том же внешнем виде - тем выше энтропия.

Примеры роста энтропии

Лёд тает → упорядоченный кристалл превращается в более хаотичную жидкость.
Перемешивание красителя в воде → невозможно вернуть прежнее состояние.

Сгорание топлива → энергия уходит в тепло, и её уже нельзя полностью собрать обратно.

Энтропия и информационная теория

В теории информации (Шеннон) энтропия также описывает неопределённость в сообщении. Чем больше вариантов возможных символов - тем выше информационная энтропия.
Это подчеркивает глубокую связь между физическим беспорядком и информационной неопределённостью.

8.2 Практическая часть

Пример MATLAB кода:

```
% ------------------------------
% Энтропия по Больцману: S = k * log(W)
% Где W = 1 + (число разрушенных ячеек)
% ------------------------------

% Параметры системы
N = 100;                % Общее число ячеек жизни
rows = 10; cols = 10;   % Размерность отображения
t_max = 100;            % Время моделирования
k = 1;                  % Условная постоянная Больцмана
decay_rate = 0.03;      % Скорость деградации жизни

% Подготовка графиков
figure('Position', [100 100 1000 400]);

% Визуализация состояния корабля
subplot(1,2,1);
life_plot = imagesc(zeros(rows, cols));
```

```
colormap([0 0 0; 0 1 0]);  % Черный = мертвая клетка, Зеленый = живая
title('Состояние биосистемы на корабле');
axis off;

% График энтропии
subplot(1,2,2);
entropy_plot = plot(0, 0, 'LineWidth', 2);
xlabel('Время');
ylabel('Энтропия S');
title('Энтропия: S = k \cdot log(W), W = 1 + число разрушений');
xlim([0 t_max]);
ylim([0 k * log(N+1)]);
grid on;
hold on;

% Инициализация
alive_cells = N;
entropy_history = zeros(1, t_max);

for t = 1:t_max
    % Экспоненциальное убывание числа живых ячеек
    alive_cells = round(N * exp(-decay_rate * t));
    alive_cells = max(alive_cells, 0);

    % Формируем состояние: живые + мёртвые
    state = [ones(1, alive_cells), zeros(1, N - alive_cells)];
    state = state(randperm(N));
    state_matrix = reshape(state, rows, cols);

    % Обновляем изображение
    set(life_plot, 'CData', state_matrix);
```

```
    % Уравнение Больцмана - W = 1 + число разрушенных
    W = 1 + (N - alive_cells);
    XXXXXXXXXXX; % Введите формулу энтропии выбитую на могильном камне Больцмана
    entropy_history(t) = S;

    % Обновляем график энтропии
    set(entropy_plot, 'XData', 1:t, 'YData', entropy_history(1:t));

    pause(0.05);
end
```

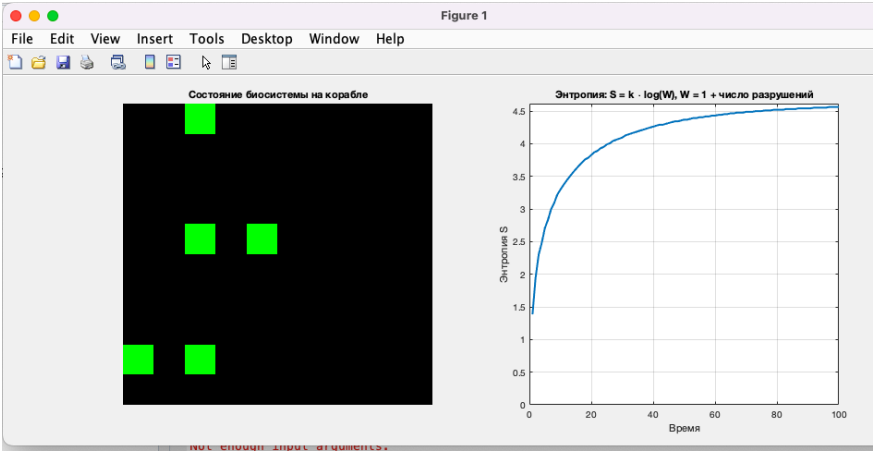

Рисунок 8.1 - Пример вывода решения

1. После строки W = 1 + (N - alive_cells); напишите формулу энтропии что бы запустить симуляцию.

2. Попробуйте изменять начальные параметры: Время моделирования, Скорость деградации жизни. Посмотрите, как будет изменяться симуляция

8.4 Контрольные вопросы

1. Что такое энтропия с точки зрения термодинамики?
2. Почему во всех замкнутых системах энтропия со временем не убывает?
3.Какая связь между числом микросостояний W и энтропией S?
4. Что происходит с графиком энтропии при полном разрушении всех ячеек?
5. Какие параметры в реальном космическом корабле могли бы привести к незаметному, но необратимому росту энтропии?

9. Контроль преломления

Станция Элион, Сахара. 2064 год.
После серии солнечных бурь человечество утратило большую часть спутниковой инфраструктуры. Связь с внешним миром фрагментарна и нестабильна. Один из немногих действующих объектов - изолированный купол связи в пустыне, защищающий архивы и экспериментальный квантовый канал.
Купол построен из метаматериала с переменным показателем преломления и способен фильтровать световые импульсы по их углу падения. Это позволяет защититься от высокочастотных излучений и возможных вредоносных протоколов.
Сегодня система зафиксировала необычный фотонный импульс, приближающийся с орбиты под нестандартным углом. Сигнатура частично совпадает с медицинскими маяками времён до катастрофы, но сигнал может быть поддельным.
Ваша задача как оператора:
проанализировать параметры (n_1, n_2, θ_1) и принять решение - пропустить импульс (если он преломляется) или изменить условия так, чтобы он отразился.
Выбор может стоить связи. Или безопасности.

9.1 Теоретическая часть

Когда свет переходит из одной оптической среды в другую, он изменяет направление - это называется преломлением. Этот процесс подчиняется закону Снеллиуса.

Закон Снеллиуса

Закон Снеллиуса описывает соотношение между углом падения и углом преломления света:

$$N_1 \cdot \sin(\theta_1) = n_2 \cdot \sin(\theta_2) \quad (9.1)$$

где
n_1 - показатель преломления первой среды (например, стеклянного купола);
n_2 - показатель преломления второй среды (например, воздуха);
θ_1 - угол падения (между лучом и нормалью);
θ_2 - угол преломления.

Условия преломления и отражения

Если $\sin(\theta_2) \leq 1$, свет преломляется и проходит в новую среду.
Если $\sin(\theta_2) > 1$, преломление невозможно - происходит полное внутреннее отражение.

Критический угол

При переходе света из более плотной среды в менее плотную существует критический угол θ_кр, при превышении которого происходит полное отражение:

$$\Theta\_кр = \arcsin(n_2 / n_1), \text{ при } n_1 > n_2 \quad (9.2)$$

9.2 Практическая часть

Пример MATLAB кода:

```
function sahara
   % Контроль преломления

   f = figure('Name', 'Контроль преломления', ...
      'Position', [300 200 760 560], 'Color', [0.1 0.1 0.1]);

   % Поля ввода
   uicontrol(f, 'Style', 'text', 'String', 'n1 (внутри купола):', ...
```

```
    'Position', [30 470 130 20], 'HorizontalAlignment', 'left', ...
    'ForegroundColor', 'w', 'BackgroundColor', f.Color);
n1_edit = uicontrol(f, 'Style', 'edit', 'String', '1.52', ...
    'Position', [160 470 60 25]);

uicontrol(f, 'Style', 'text', 'String', 'n₂ (атмосфера):', ...
    'Position', [30 430 130 20], 'HorizontalAlignment', 'left', ...
    'ForegroundColor', 'w', 'BackgroundColor', f.Color);
n2_edit = uicontrol(f, 'Style', 'edit', 'String', '1.00', ...
    'Position', [160 430 60 25]);

uicontrol(f, 'Style', 'text', 'String', 'θ₁ (угол сигнала, °):', ...
    'Position', [30 390 150 20], 'HorizontalAlignment', 'left', ...
    'ForegroundColor', 'w', 'BackgroundColor', f.Color);
theta1_edit = uicontrol(f, 'Style', 'edit', 'String', '60', ...
    'Position', [180 390 60 25]);

% Критический угол
critical_txt = uicontrol(f, 'Style', 'text', 'String', 'θ_кр = -', ...
    'Position', [500 470 220 20], ...
    'ForegroundColor', [0.9 0.9 0.9], ...
    'BackgroundColor', f.Color, ...
    'FontAngle', 'italic', 'HorizontalAlignment', 'left');

% Кнопка запуска
uicontrol(f, 'Style', 'pushbutton', 'String', 'Проверить сигнал', ...
    'FontSize', 12, 'BackgroundColor', [0.6 0.7 1], ...
    'Position', [280 430 200 50], ...
    'Callback', @(~,~) runSimulation());

% Реплика
story_txt = uicontrol(f, 'Style', 'text', 'String', '', ...
    'Position', [280 390 440 30], 'FontSize', 11, ...
```

```
    'ForegroundColor', [0.9 0.9 1], ...
    'BackgroundColor', f.Color, ...
    'FontAngle', 'italic', 'HorizontalAlignment', 'left');

% График
ax = axes(f, 'Units', 'pixels', 'Position', [100 50 550 300]);
axis(ax, 'equal');
xlim(ax, [-1, 1]);
ylim(ax, [-0.5, 0.5]);
hold(ax, 'on');
grid(ax, 'on');
set(ax, 'Color', [0.05 0.05 0.05]);

function runSimulation()
    cla(ax);
    line(ax, [-1 1], [0 0], 'Color', 'w', 'LineWidth', 2); % граница

    % Ввод
    n1 = str2double(n1_edit.String);
    n2 = str2double(n2_edit.String);
    theta1_deg = str2double(theta1_edit.String);
    theta1 = deg2rad(theta1_deg);
    L = 0.8;

    % Критический угол
    if n1 > n2
        theta_crit = rad2deg(asin(n2 / n1));
        critical_txt.String = sprintf('θ_кр = %.2f°', theta_crit);
    else
        critical_txt.String = 'θ_кр = - (нет полного отражения)';
    end

    % Закон Снеллиуса
```

```
    XXXXXXXXXXXXXXX;

    % Падающий луч
    x_end = -L * cos(theta1);
    y_end = L * sin(theta1);
    for t = 0:0.05:1
        x = t * x_end;
        y = t * y_end;
        plot(ax, [0, x], [0, y], 'b-', 'LineWidth', 2);
        pause(0.01);
        if t < 1
            cla(ax); line(ax, [-1 1], [0 0], 'Color', 'w', 'LineWidth', 2);
        end
    end
    text(ax, x_end, y_end + 0.05, 'Сигнал', 'Color', 'b');

    % Решение
    if abs(sin_theta2) > 1
        theta_reflect = theta1;
        x2 = L * cos(theta_reflect);
        y2 = L * sin(theta_reflect);
        for t = 0:0.05:1
            x = t * x2; y = t * y2;
            plot(ax, [0, x], [0, y], 'r--', 'LineWidth', 2);
            pause(0.01);
            if t < 1
                cla(ax); line(ax, [-1 1], [0 0], 'Color', 'w', 'LineWidth', 2);
                plot(ax, [0, x_end], [0, y_end], 'b-', 'LineWidth', 2);
            end
        end
        text(ax, x2, y2 + 0.05, 'Отражение', 'Color', 'r');
        story_txt.String = 'Сигнал отражён. Купол заблокировал
внешний импульс.';
```

```
            story_txt.ForegroundColor = [1 0.4 0.4];
        else
            theta2 = asin(sin_theta2);
            x3 = L * cos(theta2);
            y3 = -L * sin(theta2);
            for t = 0:0.05:1
                x = t * x3; y = t * y3;
                plot(ax, [0, x], [0, y], 'g-', 'LineWidth', 2);
                pause(0.01);
                if t < 1
                    cla(ax); line(ax, [-1 1], [0 0], 'Color', 'w', 'LineWidth', 2);
                    plot(ax, [0, x_end], [0, y_end], 'b-', 'LineWidth', 2);
                end
            end
            text(ax, x3, y3 - 0.05, 'Проникновение', 'Color', 'g');
            story_txt.String = 'Сигнал прошёл через купол. Данные
приняты системой.';
            story_txt.ForegroundColor = [0.5 1 0.5];
        end
    end
end
```

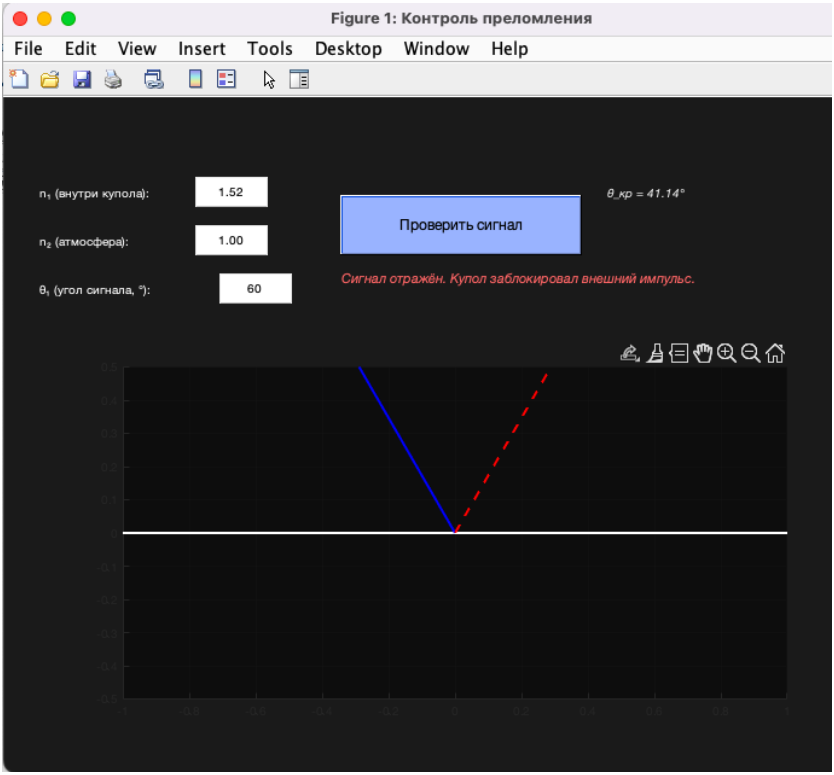

Рисунок 9.1 - Пример вывода решения

9.3 Задачи на практическую часть

1. В строке % Закон Снеллиуса запишите необходимую формулу. Подсказка [redacted]
2. Попробуйте изменять начальные параметры: n1 (внутри купола), n2 (атмосфера), θ (угол сигнала). Посмотрите, как будет изменяться симуляция.

9.4 Контрольные вопросы

1. Сформулируйте закон Снеллиуса. Какие параметры в него входят?
2. Что происходит, если sin(θ2)>1sin? Почему это физически невозможно?
3. Что такое критический угол? Как он вычисляется?
4. В каких реальных технологиях сегодня используется принцип полного внутреннего отражения?

10. RTL-SDR

Современные беспроводные технологии стремительно развиваются, охватывая такие области, как мобильная связь, спутниковая навигация, цифровое телевидение, радиолокация и Интернет вещей (IoT). Изучение принципов работы радиосистем становится важной частью подготовки специалистов в области телекоммуникаций, радиотехники и информационных технологий.

Одним из доступных и мощных инструментов для изучения радиосигналов является устройство **RTL-SDR (Realtek Software Defined Radio)** - программно-определяемое радио, позволяющее принимать широкий диапазон радиочастот (от 24 МГц до 1.7 ГГц) и анализировать сигналы с помощью программных средств. Благодаря поддержке в MATLAB, стало возможным выполнять сложную обработку радиосигналов, строить спектры, проводить демодуляцию и исследовать радиочастотную среду в реальном времени.

10.1 Теоретическая часть. FM радио

Введение в SDR (Программно-Определяемое Радио)

Программно-определяемое радио (SDR, Software Defined Radio) - это технология, позволяющая реализовать функции традиционного радио (приём, передача, обработка сигналов) с помощью программного обеспечения вместо аппаратных компонентов. Такая гибкость позволяет быстро адаптироваться к различным стандартам радиосвязи, проводить исследования радиоспектра и реализовывать сложные алгоритмы обработки сигналов.

Один из наиболее доступных SDR-ресиверов - это RTL-SDR, основанный на ТВ-тюнере RTL2832U. Хотя изначально он предназначался для приёма цифрового телевидения, благодаря возможности прямого доступа к радиосигналу его можно использовать для приёма широкого диапазона частот - от 500 кГц до 1.7 ГГц.

Одно из практических применений SDR - приём спутниковых сигналов. С помощью приёмника RTL-SDR и программных средств, таких как MATLAB или GNU Radio, можно подключаться к вещательным спутникам, например NOAA, и принимать метеорологические изображения Земли в реальном времени. SDR позволяет наблюдать спектр спутникового сигнала, демодулировать его и анализировать передаваемые данные, что делает технологию удобной для учебных и исследовательских целей.

FM-радиовещание

FM (Frequency Modulation, частотная модуляция) - это метод модуляции, при котором информация передаётся изменением частоты несущего сигнала. В отличие от амплитудной модуляции (AM), FM-модуляция более устойчива к шумам, потому что информация в ней передаётся изменением частоты несущего сигнала, а не амплитуды, как в AM.

Большинство помех и шумов проявляются именно как изменения амплитуды, поэтому они слабо влияют на частотно модулированный сигнал. Кроме того, при приёме FM используется ограничитель амплитуды, который удаляет амплитудные колебания, оставляя только частотные изменения, содержащие полезную информацию.

Таким образом, FM обеспечивает лучшее качество звука и меньший уровень шумов, особенно в условиях радиопомех.

Диапазон вещания FM-радиостанций обычно составляет от 88 МГц до 108 МГц. Стандартное отклонение несущей частоты при FM-модуляции составляет ±75 кГц.

Принцип работы FM-приёмника с RTL-SDR в MATLAB

Приём сигнала осуществляется с помощью RTL-SDR и блока comm.SDRRTLReceiver.
Демодуляция FM-сигнала выполняется в MATLAB с использованием специального объекта comm.FMBroadcastDemodulator, который выделяет аудиосигнал из модулированного радиосигнала.
Анализ спектра можно провести с помощью инструментов MATLAB, таких как dsp.SpectrumAnalyzer, что позволяет визуализировать мощность сигнала на различных частотах.
Воспроизведение аудио производится с помощью блока audioDeviceWriter, отправляющего обработанный аудиосигнал на звуковую карту компьютера.

10.2 Практическая часть. FM радио

Пример MATLAB кода:

```
function radio_gaga()
% Главное окно
f = figure('Name', 'RTL-SDR FM Радио', ...
    'Position', [100 100 600 450], ...
    'NumberTitle', 'off', ...
    'Resize', 'off');

% Метка и поле ввода частоты
uicontrol(f, 'Style', 'text', ...
```

```
        'Position', [30 400 120 25], ...
        'String', 'Частота (МГц):', ...
        'FontSize', 10, 'HorizontalAlignment', 'left');

    freqEdit = uicontrol(f, 'Style', 'edit', ...
        'Position', [150 400 100 25], ...
        'String', '100');

    % Кнопка Старт/Стоп
    startBtn = uicontrol(f, 'Style', 'togglebutton', ...
        'String', 'Старт', ...
        'Position', [270 400 100 25], ...
        'Callback', @startCallback);

    % Область для отображения спектра
    ax = axes('Parent', f, ...
        'Units', 'pixels', ...
        'Position', [50 60 500 320]);

    % Объявляем переменные в функции
    radio = [];
    audioPlayer = [];
    isRunning = false;

    % Заглушки доступа, чтобы MATLAB понял, что переменные
используются
    %#ok<*NASGU>

    % Обработчик кнопки
    function startCallback(src, ~)
        % Ссылаемся на переменные извне, чтобы подавить
предупреждение
        dummy = {radio, audioPlayer, isRunning};
```

```
    if get(src, 'Value') == 1
        set(startBtn, 'String', 'Стоп');
        freqMHz = str2double(freqEdit.String);

        if isnan(freqMHz) || freqMHz < ?? || freqMHz > ??
            errordlg('Введите частоту от 88 до 108 МГц');
            set(startBtn, 'Value', 0);
            set(startBtn, 'String', 'Старт');
            return;
        end

        fs = 240000;
        samplesPerFrame = 1024;

        try
            radio = comm.SDRRTLReceiver(...
                'CenterFrequency', freqMHz * 1e6, ...
                'SampleRate', fs, ...
                'EnableTunerAGC', true, ...
                'SamplesPerFrame', samplesPerFrame, ...
                'OutputDataType', 'double');

            audioPlayer = audioDeviceWriter(...
                'SampleRate', 48000, ...
                'BufferSize', 4096);

            data_prev = 0;
            isRunning = true;
            iter = 0;

            while isRunning && ishandle(f) && get(startBtn, 'Value')
== 1
```

```
            [data, len] = radio();
            if len > 0
                iter = iter + 1;

                if mod(iter, 10) == 0
                  [pxx, fxx] = periodogram(data, [], [], fs, 'centered');
freqMHz = fxx / 1e6 + freqMHz;  % Переводим в МГц с учётом
центральной частоты
plot(ax, freqMHz, 10*log10(pxx));
title(ax, 'Спектр радиосигнала');
xlabel(ax, 'Частота (МГц)');
ylabel(ax, 'Мощность (дБ)');
                    drawnow;
                end

                phase = angle(data .* conj([data_prev; data(1:end-1)]));
                data_prev = data(end);
                audio = downsample(phase, 5);
                audioPlayer(audio);
              end
            end

        catch ME
            errordlg(['Ошибка: ' ME.message]);
        end

        if ~isempty(audioPlayer), release(audioPlayer); end
        if ~isempty(radio), release(radio); end
        isRunning = false;
        set(startBtn, 'Value', 0);
        set(startBtn, 'String', 'Старт');

    else
```

```
            isRunning = false;
        end
    end
end
```

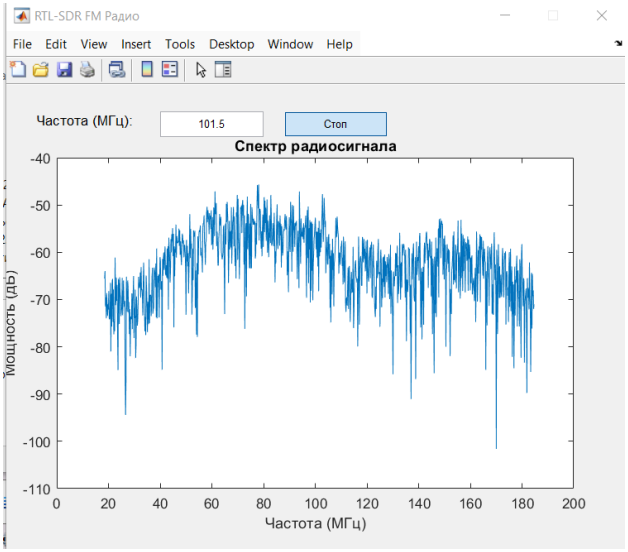

Рисунок 10.1 - Пример вывода радиостанции

10.3 Задачи на практическую часть. FM радио

1. Перед началом, подключите донгл RTL-SDR с антенной в порт usb.
2. В сети интернет найдите частоты радиостанции, на которых идет вещание в вашем населенном пункте.
3. В строке if isnan(freqMHz) || freqMHz < ?? || freqMHz > ?? вместо ?? введите частоты, которые вы теоретически можете поймать (от скольки и до скольки)
4. Запустите код, в окошке «Частота (МГц)» вбейте планируемую частоту радиостанции, послушайте, идет ли сигнал.

10.4 Практическая часть. Сигнал от пульта, рации

Пример MATLAB кода:

```
function Detector()
    f = figure('Name', 'Спектр сигнала', ...
        'Position', [100 100 700 400], ...
        'NumberTitle', 'off', ...
        'Resize', 'off');

    % Ввод частоты
    uicontrol(f, 'Style', 'text', ...
        'Position', [30 360 160 25], ...
        'String', 'Частота приёма (МГц):', ...
        'FontSize', 10, 'HorizontalAlignment', 'left');

    freqEdit = uicontrol(f, 'Style', 'edit', ...
        'Position', [200 360 100 25], ...
        'String', '433.92');

    % Кнопка запуска
    toggleBtn = uicontrol(f, 'Style', 'togglebutton', ...
        'String', 'Старт приёма', ...
        'Position', [320 360 120 25], ...
        'Callback', @startCallback);

    % График
    ax = axes('Parent', f, ...
        'Units', 'pixels', ...
        'Position', [70 60 560 270]);
    xlabel(ax, 'Частота (МГц)');
    ylabel(ax, 'Мощность (дБ)');
    title(ax, 'Спектр сигнала');
    grid(ax, 'on');
```

```
% Параметры приёма
fs = 250000;
samplesPerFrame = 4096;
threshold = 0.2;
powerCutoff_dB = -40;  % □ Новый порог отображения

radio = [];
isRunning = false;

function startCallback(~, ~)
    if get(toggleBtn, 'Value') == 1
        set(toggleBtn, 'String', 'Стоп');
        isRunning = true;

        freqMHz = str2double(freqEdit.String);
        if isnan(freqMHz)
            errordlg('Введите корректную частоту в МГц');
            return;
        end

        try
            radio = comm.SDRRTLReceiver(...
                'CenterFrequency', freqMHz * 1e6, ...
                'SampleRate', fs, ...
                'SamplesPerFrame', samplesPerFrame, ...
                'EnableTunerAGC', false, ...
                'TunerGain', 40, ...
                'OutputDataType', 'double');

            while isRunning && ishandle(f)
                [data, len] = radio();
                if len > 0
```

```
                amp = abs(data);
                if max(amp) > threshold
                    [pxx, fxx] = periodogram(data, [], [], fs, 'centered');
                    fPlot = fxx / 1e6 + freqMHz;
                    pwr_dB = 10 * log10(pxx);

                    % □ Фильтрация: только > -40 дБ
                    pwr_dB(pwr_dB < powerCutoff_dB) = NaN;

                    plot(ax, fPlot, pwr_dB, '-');
                    xlabel(ax, 'Частота (МГц)');
                    ylabel(ax, 'Мощность (дБ)');
                    title(ax, sprintf('Спектр @ %.2f МГц (порог: > %d
дБ)', freqMHz, powerCutoff_dB));
                    xlim(ax, [freqMHz - 0.2, freqMHz + 0.2]);
                    ylim(ax, [powerCutoff_dB, -5]);
                    grid(ax, 'on');
                    drawnow;
                end
            end
        end

    catch ME
        errordlg(['Ошибка: ' ME.message]);
    end

    if ~isempty(radio)
        release(radio);
    end
    isRunning = false;
    set(toggleBtn, 'Value', 0);
    set(toggleBtn, 'String', 'Старт приёма');
```

```
        else
            isRunning = false;
        end
    end
end
```

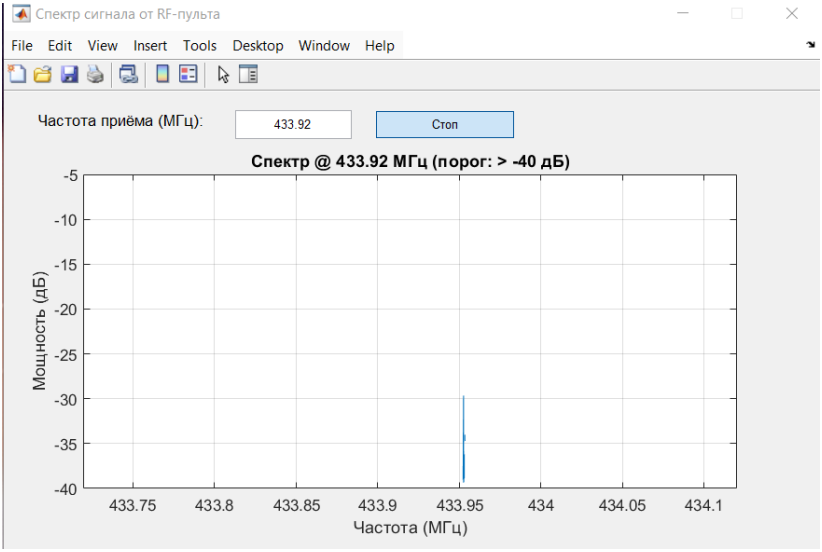

Рисунок 10.2 - Пример вывода радиочастоты от пульта

10.5 Задачи на практическую часть. Нахождение сигнала от пульта, рации

1. Перед началом, подключите донгл RTL-SDR с антенной в порт usb.
2. Запустите код, в окошке «Частота (МГц)» вбейте по очереди частоты 315 МГц и 433МГц, начните прием на этих частотах.
3. Нажимайте и удерживайте кнопки на пульте что бы убедиться, что сигнал действительно появился.

4. Включите рацию, настройте частоту на нем любую частоту. Эту же частоту вбейте в вашей программе, убедитесь, что сигнал действительно появился.

Рисунок 10.3 - Фото необходимого оборудования для данной работы

10.6 Контрольные вопросы

1. Что означает аббревиатура SDR и в чём заключается её основное преимущество?
2. Какие функции традиционного радиоприёмника реализуются программно в SDR?
3. В каком диапазоне частот ведётся FM-радиовещание?
4. В чём суть частотной модуляции (FM)?
5. Какую роль выполняет приёмник RTL-SDR в данной лабораторной работе?
6. Почему FM-модуляция более устойчива к шумам, чем AM-модуляция?
7. Можно ли с помощью RTL-SDR принимать сигналы со спутников? Приведите пример.
8. В чём практическая ценность SDR-подхода для обучения и исследований?

11. Сферическая симуляция тлеющей плазмы

Плазменный шар представляет собой прозрачную стеклянную сферу, внутри которой находится разреженный газ (обычно неон, аргон или смесь инертных газов).
В центре шара расположен электрод, к которому подается переменное высокочастотное напряжение. В результате в газе создаются ионизированные каналы - плазменные нити, которые следуют линиям электрического поля и визуально тянутся к стенкам шара.
Электроны, ускоряясь в поле, сталкиваются с атомами газа, вызывая их возбуждение и последующее излучение фотонов - именно это и создает характерное свечение.
При касании руки к шару изменяется распределение потенциала, и часть нитей концентрируется в сторону точки касания - эффект взаимодействия с внешним проводящим телом.

11.1 Теоретическая часть. Модель в MATLAB

В работе используется упрощённая двумерная модель (вид сверху), в которой: центральный электрод моделируется в виде круга радиуса a, внешний радиус шара - R, частицы (электроны) движутся под действием радиального поля с амплитудой V0 и частотой f0, демпфирование ν имитирует столкновения с нейтральными атомами, добавлен стохастический шум noise_amp для реалистичного мерцания нитей, реализовано “прикосновение” к стеклу - мягкая притяжка нитей к движущейся точке на границе.

11.2 Практическая часть.

Пример MATLAB кода:
function ploaz()

```
%% -------- Параметры --------
R  = 1.00;       % внешний радиус "шара" (усл. ед.)
a  = 0.08;       % радиус центрального электрода
V0 = 14.0;       % амплитуда "напряжения"
f0 = 2.2;        % частота мерцания (Гц, визуальная)
Bexp = 1.0;      % спад |E| ~ 1/r^Bexp (0.8..1.5 - красиво)
nu = 2.0;        % демпфирование скорости (упругие столкновения с
нейтралами)
noise_amp = 0.9;  % "дрожание" нитей (боковая шумовая сила)
acc_scale = 0.01; % масштаб ускорения от поля (зрелищность)
Np = 1800;       % число частиц
trailCount = 160; % сколько частиц рисовать со следами
trailLen   = 40;  % длина следа
dt  = 0.008;     % шаг интегрирования (с)
steps = 8000;    % количество шагов анимации
seed = 1;        % для воспроизводимости

% Доп. "палец" у стекла (притягивает нити к точке касания)
touch_on = true;
touch_strength = 0.9;     % 0..1
touch_move_speed = 0.25;  % рад/с (медленное скольжение точки
касания)

%% -------- Инициализация --------
rng(seed);
% Позиции частиц рождаем вблизи центра
r0 = a + (0.03 + 0.02*rand(Np,1))*(R-a);
th = 2*pi*rand(Np,1);
x = r0.*cos(th);  y = r0.*sin(th);

% Начальные скорости небольшие
vx = 0.2*randn(Np,1); vy = 0.2*randn(Np,1);
```

```
% Поднабор для следов
idxTrail = randperm(Np, min(trailCount, Np));
trailX = nan(trailLen, numel(idxTrail));
trailY = nan(trailLen, numel(idxTrail));

% Геометрия "пальца" у стекла
touch_angle = 0;  % начальное положение
touch_R = 0.98*R;

%% -------- Фигура --------
figure('Color','k','Name','Plasma Ball','Position',[60 60 900 900]);
ax = axes('Color','k'); axis(ax,'equal'); axis(ax,'off'); hold(ax,'on');
xlim(R*[-1.05 1.05]); ylim(R*[-1.05 1.05]);

% Рисуем внешнюю сферу и центральный электрод
t = linspace(0,2*pi,400);
plot(R*cos(t), R*sin(t), '-', 'Color',[.8 .8 .9], 'LineWidth',1.2);       %
стекло
plot(a*cos(t), a*sin(t), '-', 'Color',[.7 .7 .9], 'LineWidth',1.0);    % катод

% Несколько "направляющих" слабых лучей (для глубины)
for k=1:18
   ang = 2*pi*k/18;
   plot([a*cos(ang), R*cos(ang)], [a*sin(ang), R*sin(ang)], '-', ...
      'Color',[.25 .35 .55], 'LineWidth',0.5);
end

% Графика частиц и следов
hPts = scatter(x, y, 8, 'filled');
hPts.CData = repmat([0.85 0.9 1.0], Np, 1);   % холодное свечение

hTrails = gobjects(numel(idxTrail),1);
for k=1:numel(idxTrail)
```

```
    hTrails(k) = plot(nan, nan, '-', 'LineWidth',1.0, 'Color',[0.3 0.75 1 0.35]);
end

% "Палец" (точка касания) - тонкий полупрозрачный маркер у стекла
hTouch = plot(nan, nan, 'o', 'MarkerSize',6, 'MarkerFaceColor',[1 .85 .3], ...
    'MarkerEdgeColor','none', 'Color',[1 .85 .3 .4]);

ttl = title(ax,'Plasma Ball - AC glow','Color',[.95 .98 1]);
set(ttl,'FontSize',14);

info = text(-0.98*R, -1.05*R, '', 'Color',[.85 .9 1], 'FontName','Consolas');

%% -------- Основной цикл --------
t0 = tic;
for it = 1:steps
    time = it*dt;

    % Переменное напряжение (визуальное мерцание)
    Vt = V0 * sin(2*pi*f0*time);
    % Радиальное поле: E = sign(Vt) * (R/r)^Bexp * e_r
    r = hypot(x,y);
    % защита от r<a
    rEff = max(r, a*1.0);
    Er_mag = acc_scale * (abs(Vt)+0.5) .* (R./rEff).^Bexp; % +0.5 для "тления" при малом V

    % Единичный радиальный вектор
    ex = x./rEff; ey = y./rEff;
```

```
    % Сила от радиального поля: направляем от центра к стеклу, чтобы "нити" тянулись наружу
    % (художественная вольность: для красоты пусть всегда тянет наружу)
    ax_field = Er_mag .* ex;
    ay_field = Er_mag .* ey;

    % Добавим "прикосновение": мягкая аттракция к точке на стекле
    if touch_on
        touch_angle = wrapToPi(touch_angle + touch_move_speed*dt);
        tx = touch_R*cos(touch_angle);
        ty = touch_R*sin(touch_angle);
        hx = tx - x; hy = ty - y;
        dist = sqrt(hx.^2 + hy.^2) + 1e-6;
        % направляющая к "пальцу", сила убывает с расстоянием
        ax_touch = touch_strength * (hx ./ dist);
        ay_touch = touch_strength * (hy ./ dist);
    else
        tx = nan; ty = nan; ax_touch = 0; ay_touch = 0;
    end

    % Поперечная "дрожь" (стохастика, задаёт извилистость нитей)
    % Берём направление, перпендикулярное радиальному
    nx = -ey; ny = ex;
    xi = randn(Np,1); eta = randn(Np,1);
    ax_noise = noise_amp * (0.6*xi.*nx + 0.4*eta.*ny);

    % Полное ускорение с демпфированием скорости (псевдо-коллизии)
    ax = ax_field + ax_touch + ax_noise - nu*vx;
    ay = ay_field + ay_touch + ax_noise*0 - nu*vy; % поперечный шум уже в ax_noise; можно добавить в ay по желанию
```

```
% Интегрирование
vx = vx + ax*dt; vy = vy + ay*dt;
x = x + vx*dt;   y = y + vy*dt;

% Границы: внешнее стекло r>=R - "прилипание" и пере-рождение у центра
r = hypot(x,y);
hitR = r>=R;
if any(hitR)
   % часть "скользит" вдоль стекла, часть пере-рождается
   slide = hitR & (rand(Np,1)<0.35);
   if any(slide)
      % скорость вдоль касательной
      exR = x(slide)./r(slide); eyR = y(slide)./r(slide);
      % тангенциальная компонента текущей скорости
      vn = vx(slide).*exR + vy(slide).*eyR;
      vtx = vx(slide) - vn.*exR;
      vty = vy(slide) - vn.*eyR;
      spd = hypot(vtx,vty)+1e-6;
      vtx = vtx./spd .* (0.6+0.4*rand(sum(slide),1));
      vty = vty./spd .* (0.6+0.4*rand(sum(slide),1));
      vx(slide) = vtx; vy(slide) = vty;
      x(slide) = exR*(R-1e-3); y(slide) = eyR*(R-1e-3);
   end

   regen = hitR & ~slide;
   if any(regen)
      % Рождение возле центра с небольшим разбросом угла
      rr = a + (0.02 + 0.02*rand(sum(regen),1))*(R-a);
      th = 2*pi*rand(sum(regen),1);
      x(regen) = rr.*cos(th);  y(regen) = rr.*sin(th);
      vx(regen) = 0.2*randn(sum(regen),1);
```

```
        vy(regen) = 0.2*randn(sum(regen),1);
    end
end

% Центральный электрод: мягкое отражение наружу
r = hypot(x,y);
hitA = r<a;
if any(hitA)
    exA = x(hitA)./max(r(hitA),1e-6); eyA = y(hitA)./max(r(hitA),1e-
6);
    vn = vx(hitA).*exA + vy(hitA).*eyA;
    vx(hitA) = vx(hitA) - 2*vn.*exA;
    vy(hitA) = vy(hitA) - 2*vn.*eyA;
    x(hitA)  = exA*(a+1e-3); y(hitA) = eyA*(a+1e-3);
end

% Следы
trailX = [x(idxTrail)'; trailX(1:end-1,:)];
trailY = [y(idxTrail)'; trailY(1:end-1,:)];

% Обновление графики
if mod(it,1)==0
    set(hPts,'XData',x,'YData',y);
    for k=1:numel(idxTrail)
        set(hTrails(k),'XData',trailX(:,k),'YData',trailY(:,k));
    end
    if touch_on
        set(hTouch,'XData',tx,'YData',ty);
    end

    fps = it / max(1e-9, toc(t0));
    info.String = sprintf('V0=%.1f, f=%.1f Hz  |  N=%d  |  dt=%.3f  |
fps=%.1f', ...
```

```
                    V0, f0, Np, dt, fps);
        drawnow limitrate
    end
end
end
```

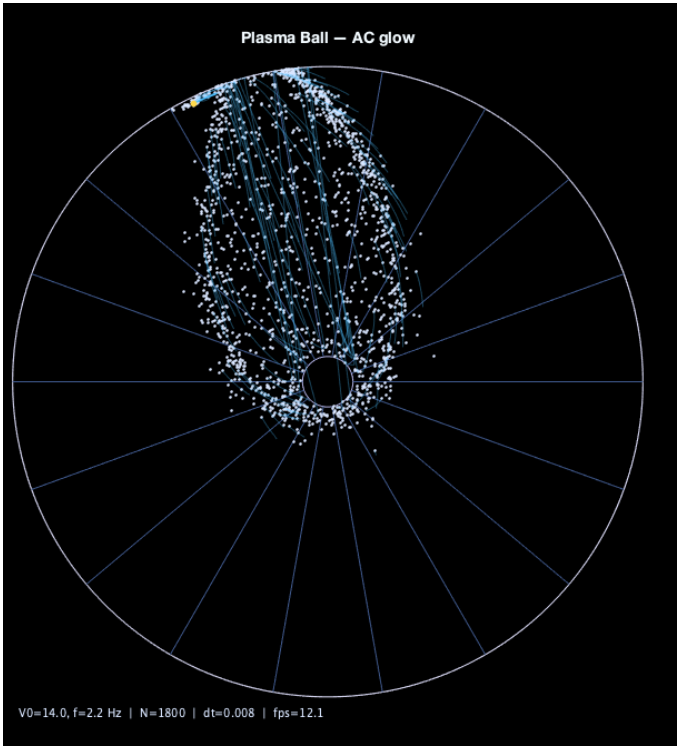

Рисунок 11.1 Пример «прикосновения» сверху плазменного шара

11.3 Задачи на практическую часть

1. Запустите программу и наблюдайте за динамикой плазменных нитей.

2. Измените параметры в начале скрипта: V0 - амплитуда напряжения (влияет на яркость и скорость нитей); f0 - частота мерцания (от 1 до 5 Гц); noise_amp - уровень "дрожания" нитей; touch_strength - сила притяжения к "пальцу".
3. Наблюдайте, как изменяется поведение нитей при варьировании параметров.

11.4 Контрольные вопросы

1. Почему в плазменном шаре наблюдаются именно разветвленные нити, а не сплошное свечение?
2. Какое физическое явление обеспечивает свечение газа в шаре?
3. Почему при касании рукой часть нитей тянется к точке касания?
4. Как изменится форма нитей при увеличении амплитуды V0?
5. Что в модели обозначает коэффициент демпфирования ν?
6. Какие ограничения у данной модели по сравнению с реальным плазменным шаром?

СОДЕРЖАНИЕ

Список литературы

1. Кондратьев А.С., Ляпцев А.В. Физика. Задачи на компьютере. - М.: ФИЗМАТЛИТ, 2008. - 400 с. - ISBN 978-5-9221-0917-8.
2. Физика: учебник и практикум для вузов / В. А. Ильин, Е. Ю. Бахтина, Н. Б. Виноградова, П. И. Самойленко; под редакцией В. А. Ильина. - Москва : Издательство Юрайт, 2025. - 399 с. - (Высшее образование). - ISBN 978-5-9916-6343-4.

Электронное учебное издание

Гаяз Харисович Тазмеев

Азат Талгатович Галиакбаров

Харис Каюмович Тазмеев

Ильназ Клирменович Хафизов

Альберт Занфирович Хазивалиев

ФИЗИКА В ЦИФРОВОЙ ЛАБОРАТОРИИ: MATLAB-МОДЕЛИ И ЭКСПЕРИМЕНТЫ

В авторской редакции

Редактор
Г. Ф. Таипова

Компьютерная верстка
Н.Н. Савицкая

Подписано к использованию 27.10.2025. Объем 1094 КБ
Уч.-изд. л. 2,10. Заказ 1890
Отдел информации и связей с общественностью
Набережночелнинского института
Казанского (Приволжского) федерального университета

423810, г. Набережные Челны, Новый город, проспект Мира, 68/19. тел. (8552) 38-47-68, e-mail: ic-nchi-kpfu@mail.