КАФЕДРА

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №4

Модели динамических систем - 1

по курсу: Компьютерное моделирование

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. № _____ _____ _____
подпись, дата инициалы, фамилия

Санкт-Петербург 2024

Цель работы

Цель настоящей работы: освоить приемы моделирования непрерывных процессов в MatLab Simulink

Ход работы

1. Самостоятельно ознакомиться со справочными сведениями относительно приложения MatLab Simulink.
2. Построить графики непрерывной (не)линейной модели решения дифференциального уравнения.
3. Разработать модель Simulink для решения дифференциального уравнения.
4. Построить графики дискретной (не)линейной модели решения разностного уравнения.
5. Разработать модель Simulink для решения разностного уравнения (системы уравнений).
6. Получить сравнительные графики поведения моделей при разных параметрах дифференциального уравнения, параметра дискретизации и настроек Simulink.
7. Составить и представить преподавателю отчет о работе.

Вариант №14

1) $(1+x^2)y' + y\sqrt{1+x^2} = xy, \;\; y(0) = 1.$

2) $\begin{cases} \dfrac{dx}{dt} = 2x - y \\ \dfrac{dy}{dt} = x + 2y \end{cases}, \;\; x(0) = -1, \;\; y(0) = 0.$

Графики непрерывной модели решения уравнений

1. Шаг 1: Анализ уравнения

Уравнение содержит несколько ключевых элементов, которые могут накладывать ограничения на область определения:

1. $1 + x^2$:
   - Это выражение всегда больше 0 для всех $x \in R$, так как $x^2> = 0$.
   - Оно не вызывает проблем при делении.
2. $\sqrt{1 + x^2}$:
   - Подкоренное выражение $1 + x^{\wedge 2}$ также всегда положительно для $x \in R$, так как $1 + x^2> = 1> 0$.
   - Следовательно, квадратный корень определен для всех x.
3. Правая часть уравнения x*y:
   - Здесь также нет ограничений, так как произведение x * y определено для любых значений x и y.

Вывод 1: Уравнение не содержит явных ограничений на x. Теоретически область определения уравнения — вся вещественная ось $x \in R$.

2. Шаг 2: Приведение уравнения к нормальной форме

Приведем уравнение к виду, пригодному для анализа и решения. Разделим обе стороны на $1 + x^2$ (всегда больше нуля):

$$y' = \frac{xy - y\sqrt{1 + x^2}}{1 + x^2}$$

Теперь уравнение в нормальной форме, и его правая часть f(x, y) выражена как:

$$f(x, y) = \frac{xy - y\sqrt{1 + x^2}}{1 + x^2}$$

Проверим, нет ли особенностей у правой части:

- $1 + x^2 > 0$, деление всегда корректно.
- Все выражения остаются определенными для $x \in R$ и любых y.

Вывод 2: Правая часть уравнения определена для всех $x \in R$.

3. Шаг 3: Начальное условие

Уравнение дополнено начальным условием:

$y(0) = 1$.

Начальное значение находится в области определения правой части f (x, y), поэтому решение можно искать на $x \in R$.

4. Шаг 4: Теоретическая область определения решения

Согласно теореме существования и единственности для дифференциальных уравнений первого порядка:

1. Если правая часть f (x, y) и её частная производная df / dy непрерывны в некоторой окрестности точки (x_0, y_0), то у уравнения существует единственное решение в этой области.

В данном случае:

- f (x, y) непрерывна, так как $1 + x^2 > 0$, $\sqrt{1 + x^{\wedge 2}}$ определено.
- df / dy также непрерывна, так как это рациональная функция без особенностей.

Следовательно, решение существует и определено на всей вещественной оси.

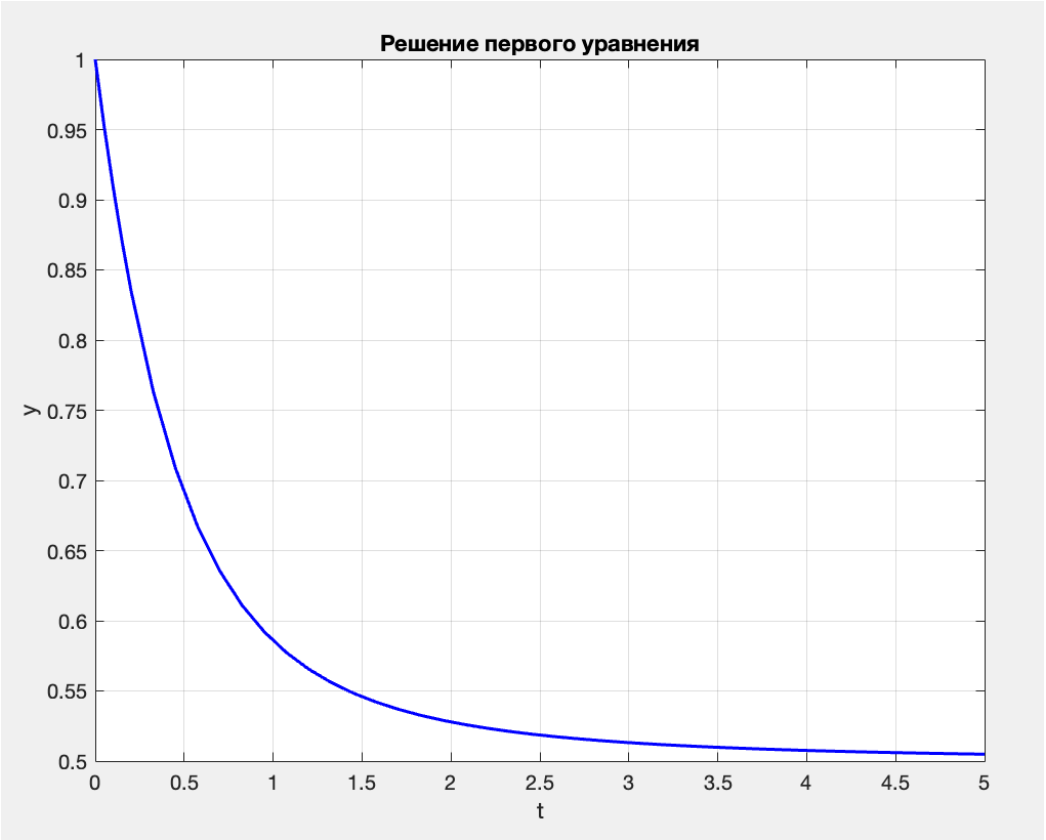

Рисунок 1 – график решения первого уравнения

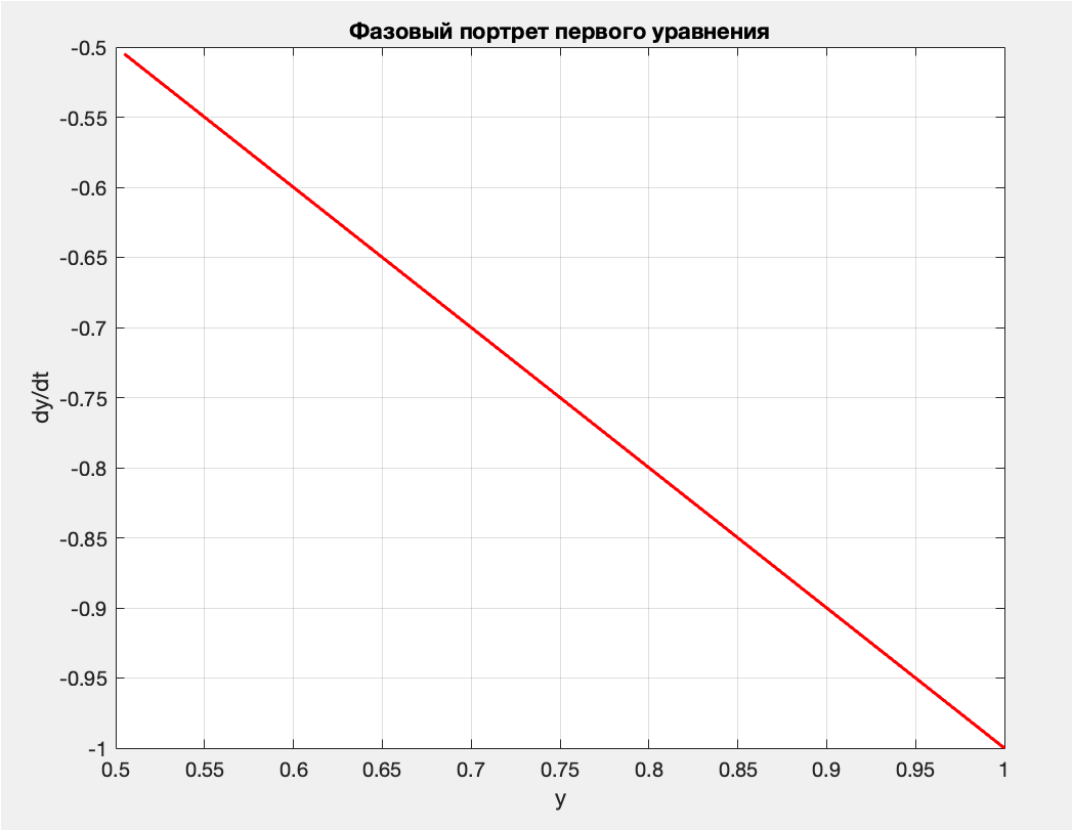

Рисунок 2 – фазовый портрет первого уравнения

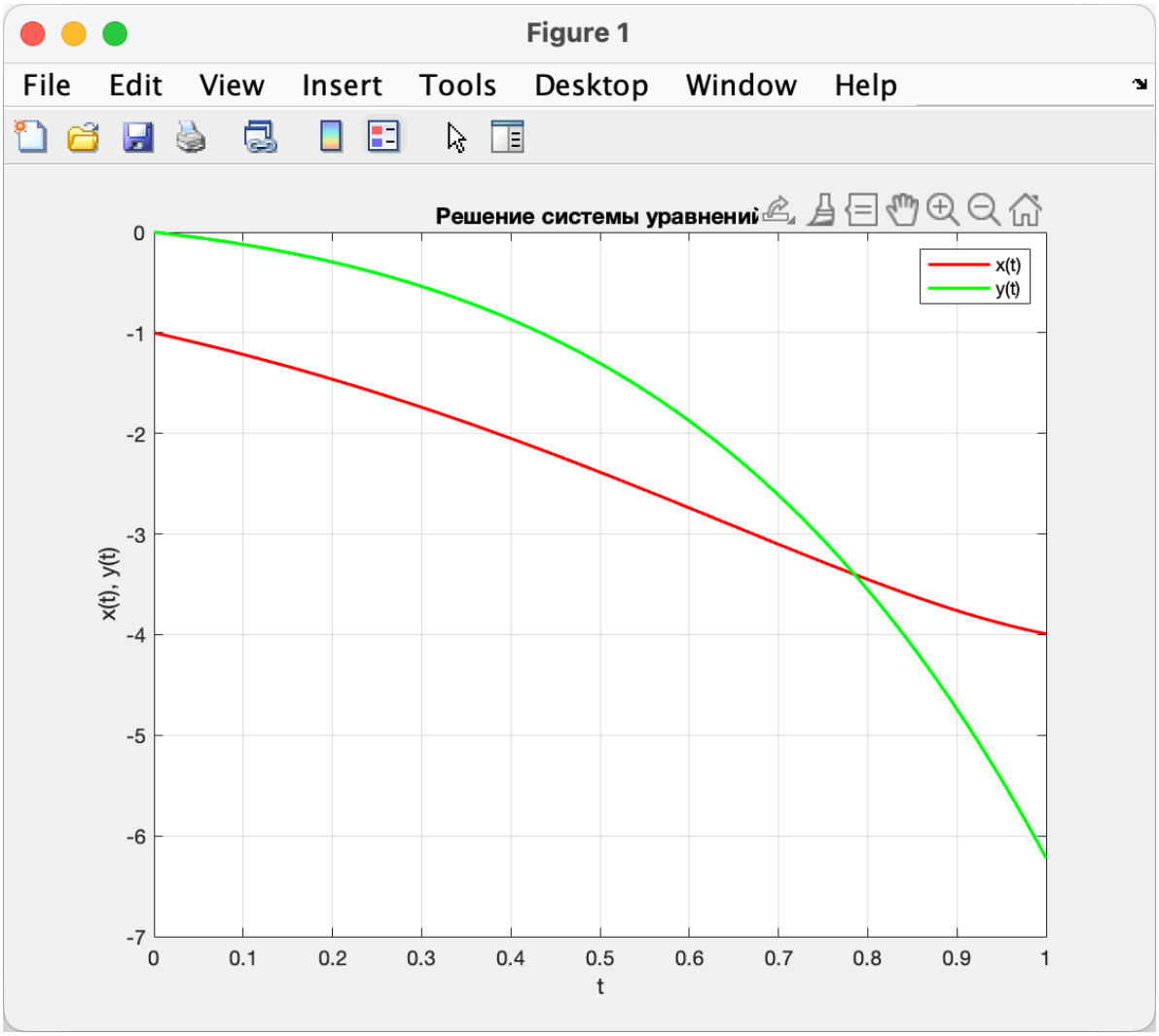

Рисунок 3 – график решения второго уравнения

Модели Simulink

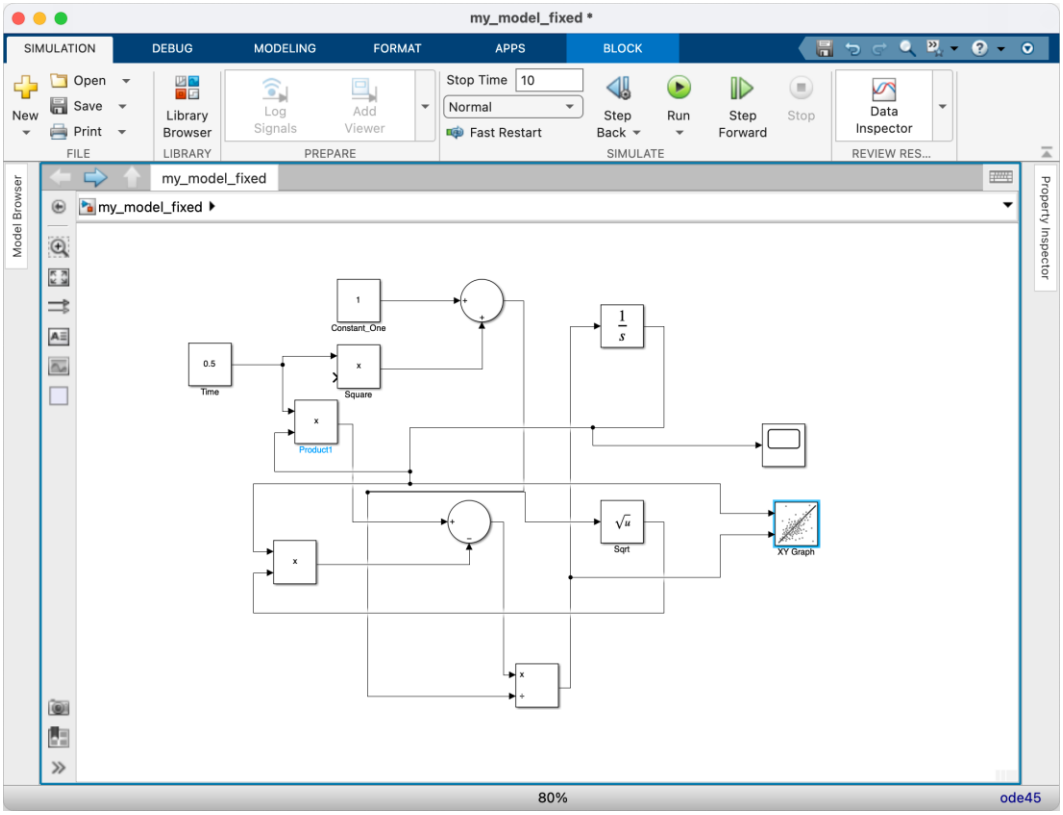

Рисунок 4 – модель для первого уравнения

Модель решает дифференциальное уравнение методом численного интегрирования. Основные блоки и их функции:

1. Constant (Time):

Генерирует постоянное значение времени t = 0.5 для вычисления выражений, связанных с t. Этот блок играет роль источника данных для дальнейших вычислений.

2. Integrator:

Реализует численное интегрирование, то есть вычисляет значение функции y(t), основываясь на выражении для dy/dt. Начальное условие интегратора задается как y (0) = 1.

3. Math Function (Sqrt):

Вычисляет квадратный корень, который используется в уравнении.

4. Product (Square):

Возводит t в квадрат ($t^{\wedge 2}$) для последующего вычисления 1 + $t^{\wedge 2}$.

5. Sum (Sum1):

Складывает 1 и $t^{\wedge 2}$, формируя выражение 1 + $t^{\wedge 2}$.

6. Product (Product1):

Вычисляет произведение t * y.

7. Product (Product2):

Вычисляет произведение y sqrt {1 + $t^{\wedge 2}$}.

8. Sum (Sum2):

Выполняет вычитание t * y - y √{1 + $t^{\wedge 2}$}, формируя числитель для уравнения.

9. Divide:

Реализует деление числителя (t*y - y √{1 + $t^{\wedge 2}$}) на знаменатель (1 + $t^{\wedge 2}$), что соответствует выражению для dy/dt.

10. Scope:

Визуализирует решение уравнения y(t) в виде графика, позволяя оценить поведение функции во времени.

11. XY Graph:

Построение фазового портрета, отображающего зависимость y(t) от dy/dt, что помогает проанализировать динамику системы.

Логика работы модели

1. Входные данные:

Генерируются значения времени t и начальные условия y (0) = 1.

2. Численные операции:

На основе заданного уравнения производятся расчёты 1 + $t^{\wedge 2}$, t * t y, y * sqrt{1 + $t^{\wedge 2}$} , а затем вычисляется dy/dt с помощью блока деления.

3. Интеграция:

Значение dy/dt передаётся в блок Integrator, который интегрирует его для получения функции y(t).

4. Визуализация:

Результаты отображаются на графике времени (Scope) и фазовом портрете (XY Graph).

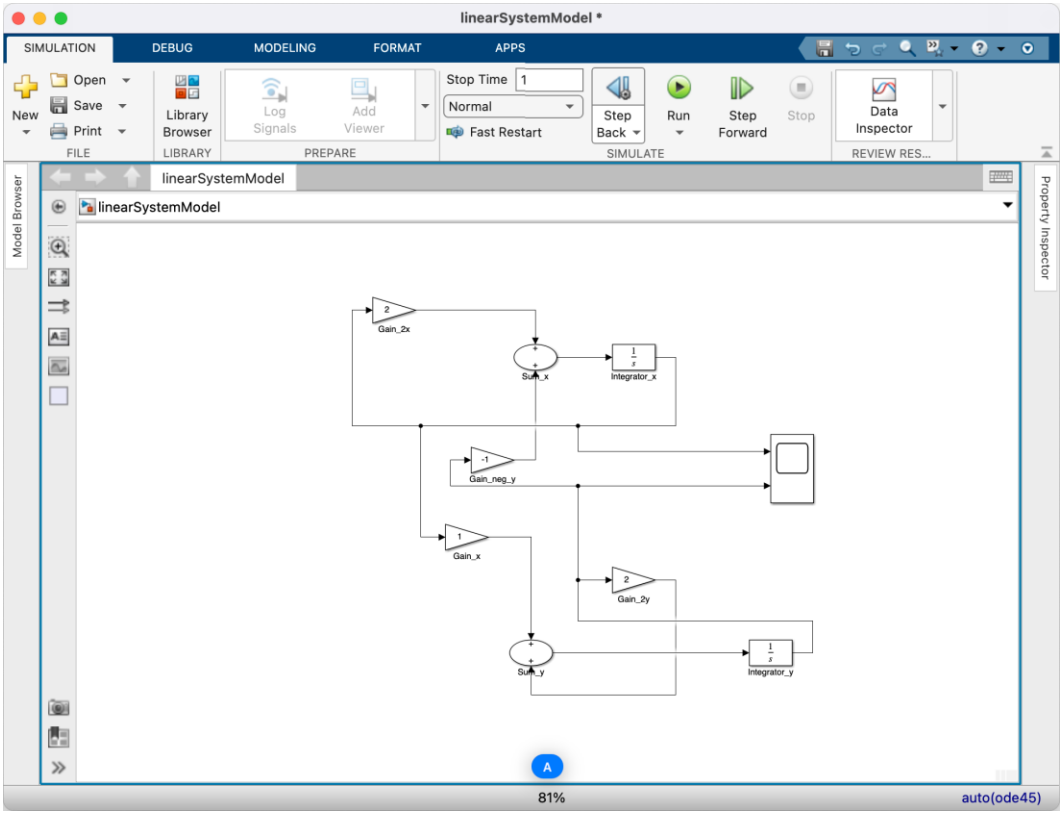

Рисунок 5 – модель для второго уравнения

Описание модели

Модель Simulink, созданная для решения системы дифференциальных уравнений, включает различные блоки для моделирования динамики двух переменных x(t) и y(t). Основные компоненты модели:

1. Интеграторы:

Integrator_x и Integrator_y: Эти блоки интегрируют функции x(t) и y(t), соответственно. Начальные условия установлены как x(0) и y(0). Эти интеграторы используются для вычисления значений x(t) и y(t) на основе их производных.

2. Сумматоры:

Sum_x: Суммирует входные значения для вычисления производной dx/dt. Он получает два входа: x и y, которые затем складываются.

Sum_y: Суммирует входные значения для вычисления производной dy/dt = x y. Он также получает два входа: x и y, которые складываются для получения производной dy/dt.

3. Коэффициенты (Gain):

Gain_4x: Умножает x(t) на коэффициент 4, что необходимо для выражения 4x в уравнении dx/dt.

Gain_1y: Умножает y(t) на коэффициент 1, что необходимо для выражения y в уравнении dx/dt.

Gain_neg05x: Умножает x(t) на коэффициент, что необходимо для выражения в уравнении dy/dt.

Gain_2y: Умножает y(t) на коэффициент 2, что необходимо для выражения 2y в уравнении dy/dt.

4. Scope:

Блок Scope используется для визуализации результатов. Он отображает значения x(t) и y(t) во времени. Модель настроена на два входных порта, что позволяет одновременно отображать оба сигнала на графике.

5. Линии соединений:

Линии соединяют блоки, обеспечивая правильную передачу сигналов. Например:

x(t) передается из интегратора Integrator_x в блок Gain_4x, а затем результат идет в Sum_x.

Аналогично, y(t) передается в блок Gain_1y, результат которого также идет в Sum_x.

Процесс аналогичен для уравнения для y(t), где блоки Gain_neg05x и Gain_2y используются для вычисления производной dy/dt, которая затем интегрируется в Integrator_y.

Основные этапы работы модели:

1. Начальные условия x(0) и y(0) задаются в интеграторах.
2. Производные dx/dt и dy/dt вычисляются с использованием коэффициентов и сумматоров.
3. Результаты передаются в соответствующие интеграторы для обновления значений x(t) и y(t).
4. Визуализация значений x(t) и y(t) происходит с помощью блока Scope, где можно наблюдать динамику системы.

Сохранение и запуск модели:

После создания и настройки модели, она сохраняется и запускается с использованием стандартного решателя. Результаты моделирования можно наблюдать на Scope, который отображает графики зависимости x(t) и y(t) от времени.

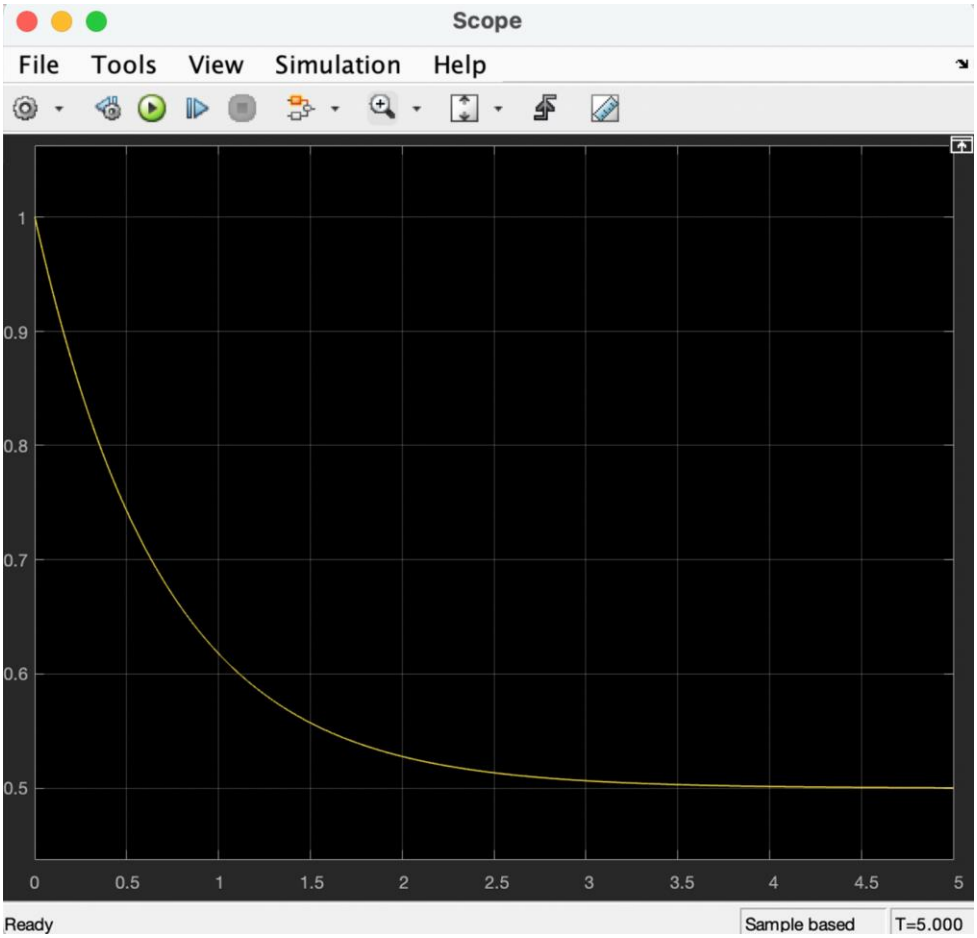

Рисунок 6 – график первой модели

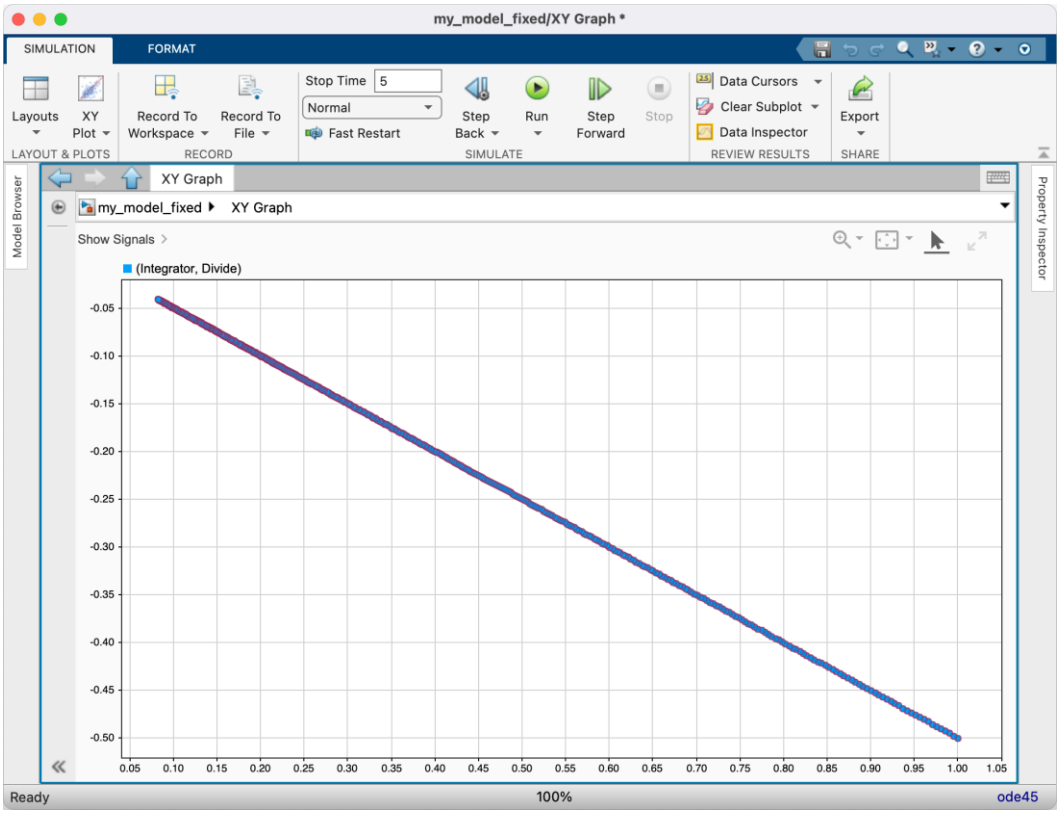

Рисунок 7 – фазовый портрет первой модели

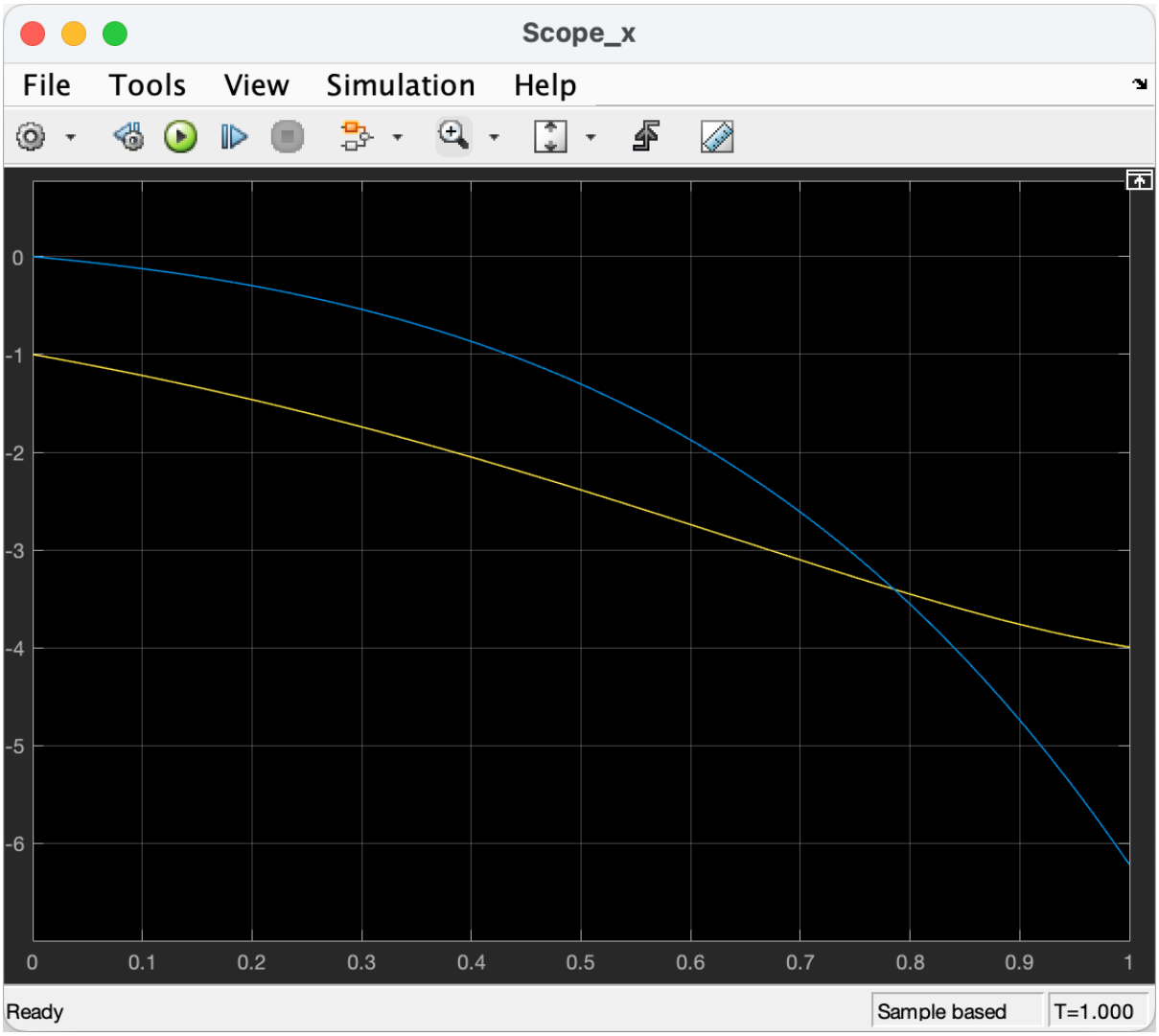

Рисунок 8 – график второй модели

Сравнение дискретных моделей с разными шагами дискретизации

Для дискретизации дифференциальных уравнений используем метод Эйлера, который аппроксимирует производные на основе значений функции в предыдущий момент времени. Рассмотрим дискретизацию для каждого из уравнений.

1. Дискретизация первого уравнения:
Для первого уравнения:

$$(1 + x^2)y' + y\sqrt{1 + x^2} = xy, y(0) = 1$$

Используем метод Эйлера для численного решения этого уравнения. Метод Эйлера позволяет выразить решение в следующем виде:

$$y_{n+1} = y_n + h * \frac{x_n y_n - y_n\sqrt{1 + x_n^2}}{1 + x_n^2}$$

Где: y_n — значение функции y в момент времени t_n , h — шаг дискретизации, x_n — значение переменной x в момент времени t_n .
Таким образом, для каждого шага h мы вычисляем значение функции y_{n+1} с использованием этой формулы.

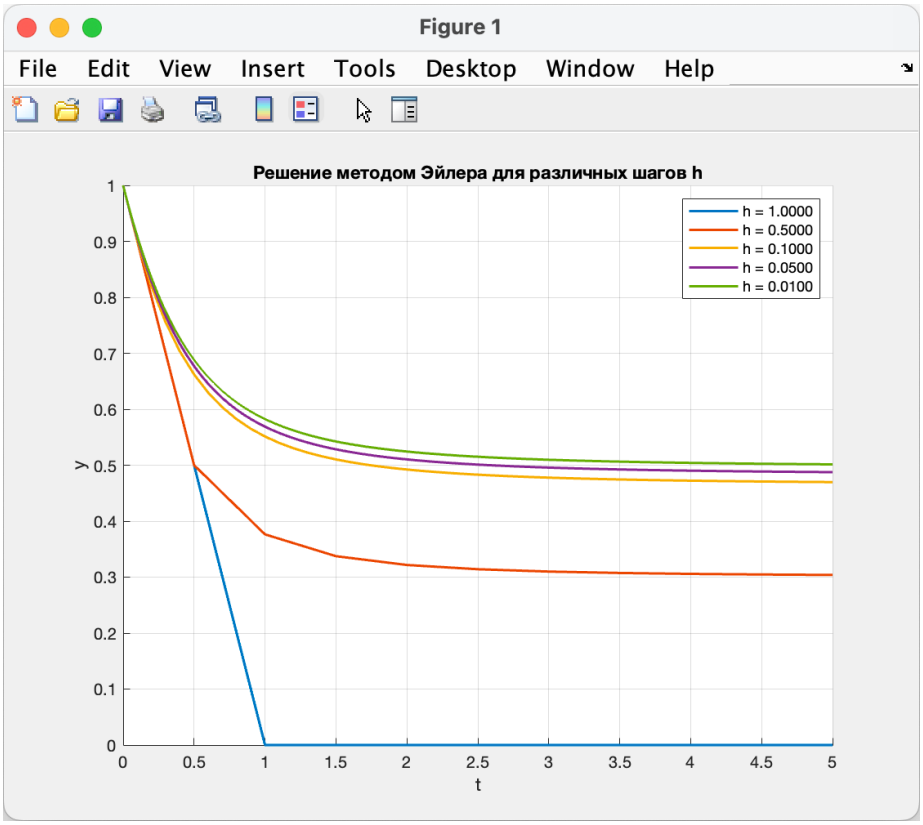

Рисунок 9 – решение с разными шагами дискретизации в Matlab для первого уравнения

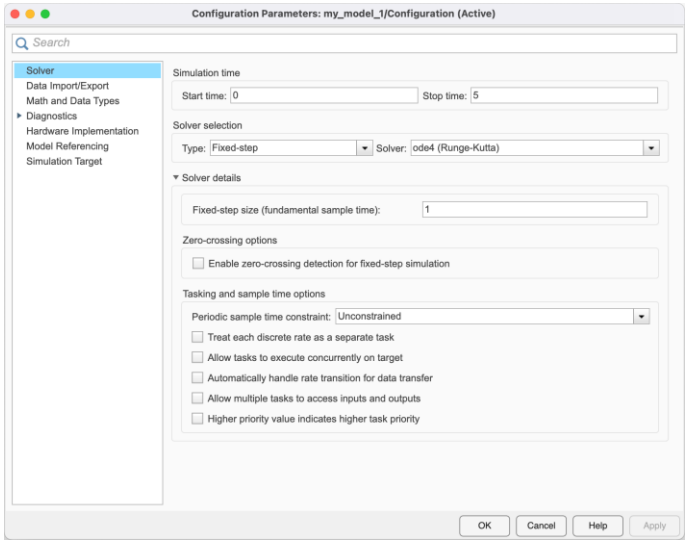

Рисунок 10 – настройка шага дискретизации в Simulink

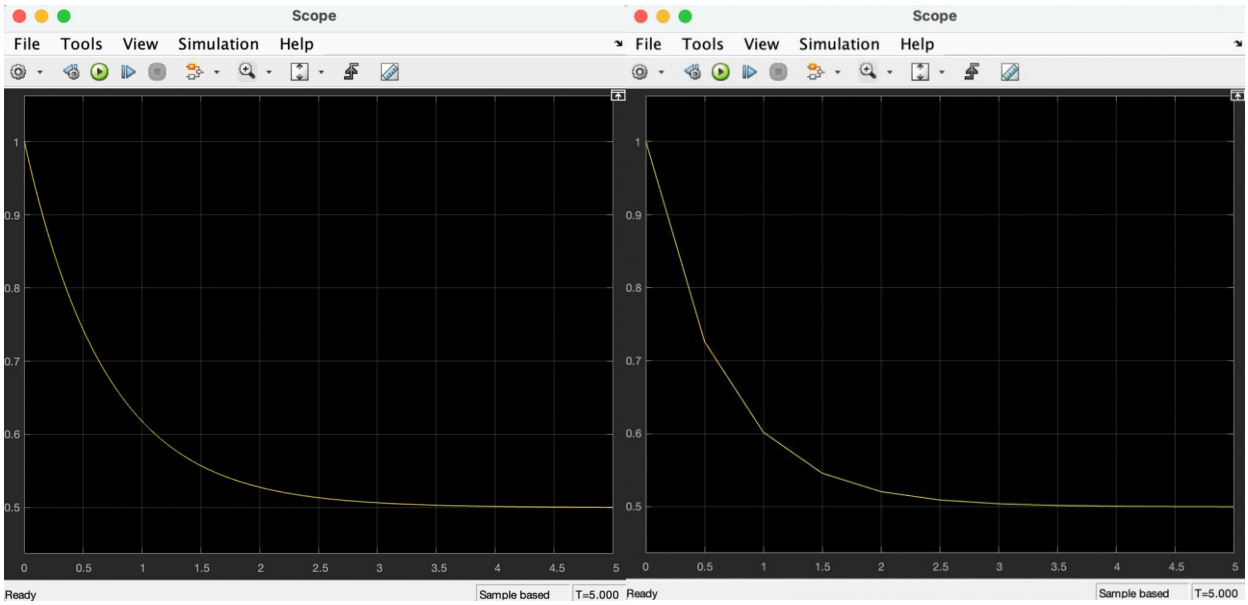

Рисунок 11 - сравнение непрерывного решения с дискретным h = 0.5 (Simulink)

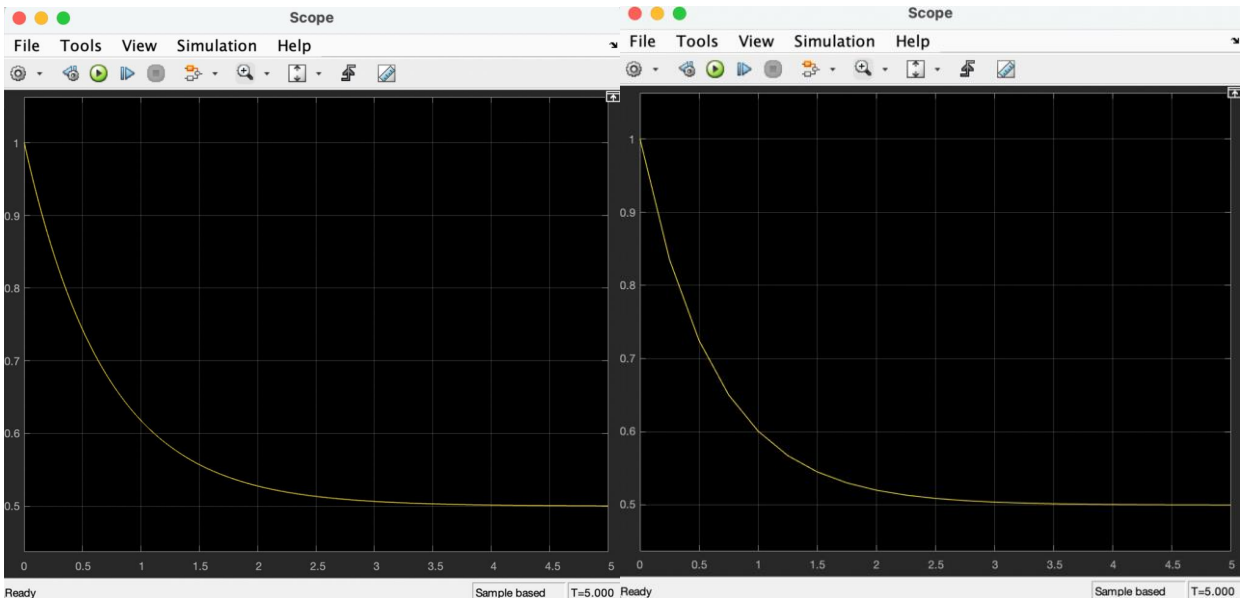

Рисунок 12 - сравнение непрерывного решения с дискретным h = 0.01(Simulink)

2. Дискретизация системы уравнений (второе уравнение):

Для системы дифференциальных уравнений:

$$\frac{\mathrm{dx}}{dt} = 2x - y$$
$$\frac{\mathrm{dy}}{dt} = x + 2y$$
$$x(0) = -1, y(0) = 0$$

Дискретизация по методу Эйлера для этой системы выглядит следующим образом:

$$y_{n+1} = y_n + h * \frac{x_n y_n - y_n\sqrt{1 + x_n^2}}{1 + x_n^2}$$

Дискретизация по методу Эйлера для этой системы выглядит следующим образом:

$$x_{n+1} = x_n + h(2x_n - y_n)$$
$$y_{n+1} = y_n + h(x_n + 2y_n)$$

Где: x_n и y_n — значения функций x и y в момент времени t_n, и h — шаг дискретизации.

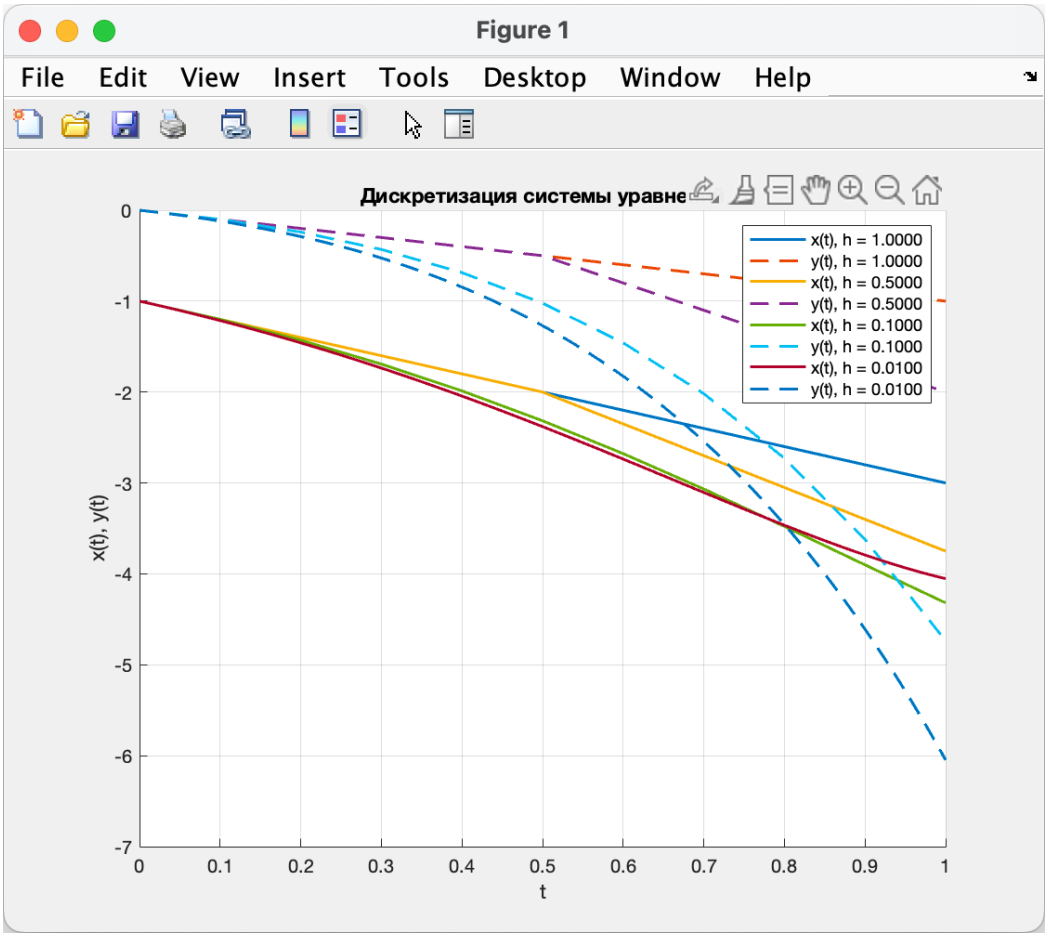

Рисунок 13 - решение с разными шагами дискретизации в Matlab для системы

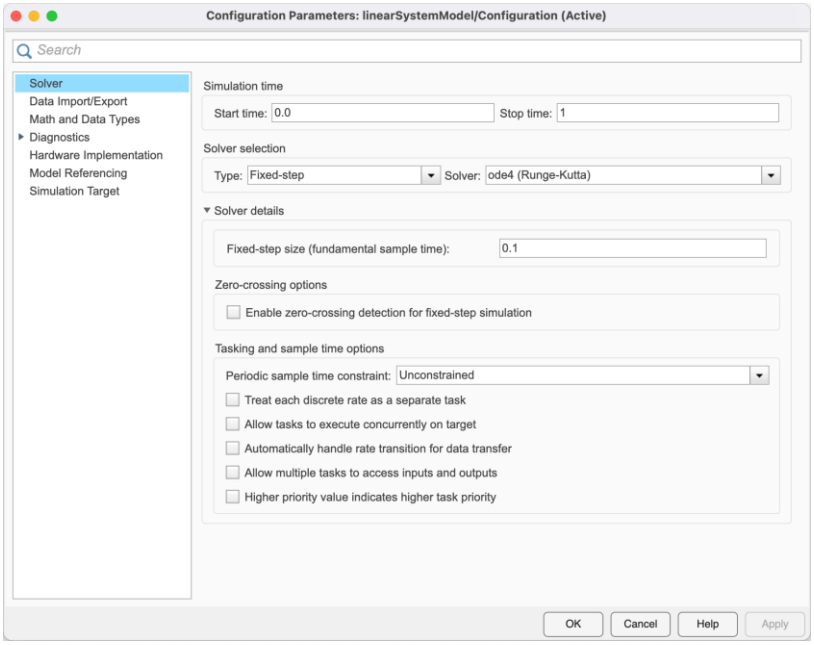

Рисунок 14 – настройка шага дискретизации в Simulink

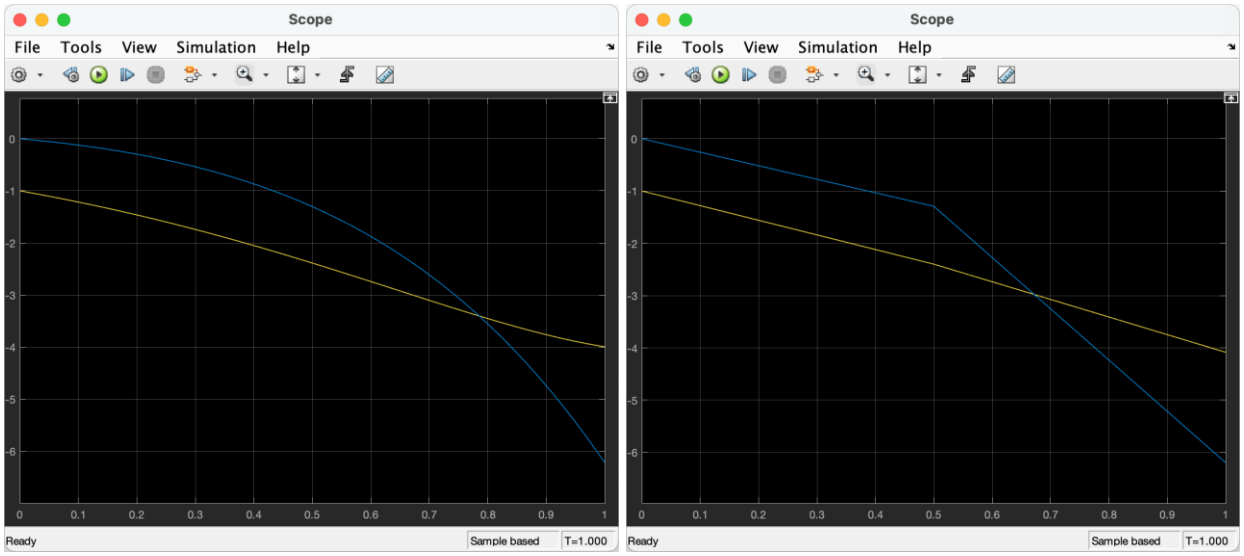

Рисунок 15 – сравнение непрерывного решения с дискретным h = 0.5 (Simulink)

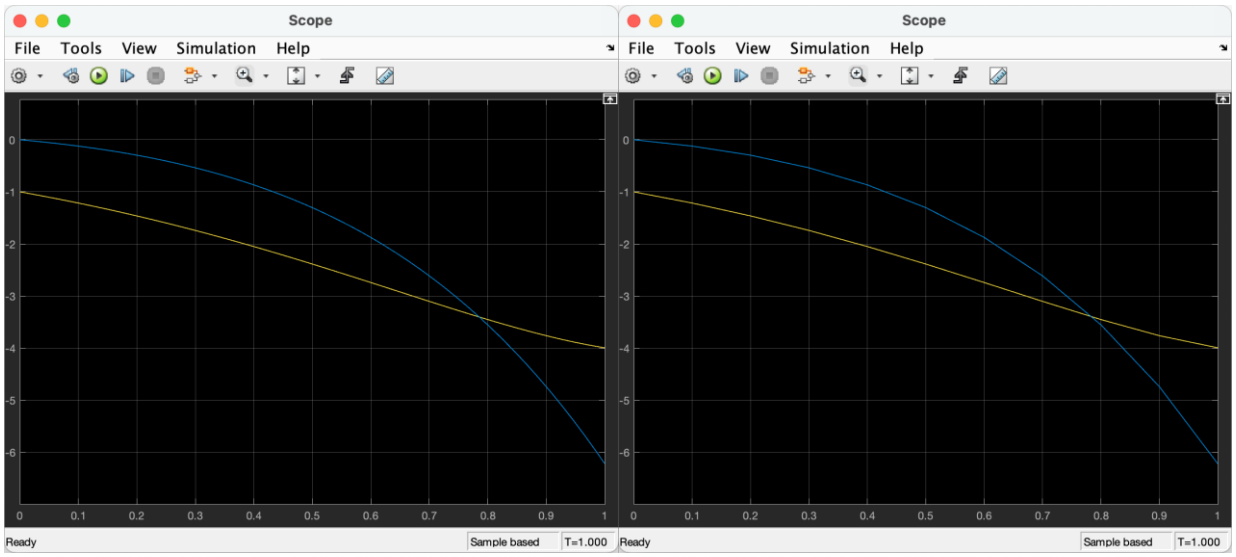

Рисунок 16 – сравнение непрерывного решения с дискретным h = 0.1 (Simulink)

Вывод

В ходе выполнения лабораторной работы было установлено, что дискретное решение системы дифференциальных уравнений может быть приближено к непрерывному с высокой точностью при условии использования малого шага дискретизации h.

В процессе работы было реализовано численное решение с применением метода Эйлера и выполнено сравнение его результатов с точным решением, полученным для непрерывной модели. Анализ показал, что уменьшение шага дискретизации приводит к снижению погрешности и позволяет добиться практически полного совпадения дискретного решения с непрерывным.

Это подтверждает, что численные методы позволяют моделировать сложные системы с высокой степенью точности при соблюдении условий выбора оптимального значения шага дискретизации. Полученные результаты наглядно продемонстрировали важность корректной настройки параметров численных методов для достижения высокой точности моделирования.

Листинг

```
tspan = [0, 5];
y0 = 1;

dydt = @(t, y) (t * y - y * sqrt(1 + t^2)) / (1 + t^2);

[t, y] = ode45(dydt, tspan, y0);

% График решения
figure;
plot(t, y, 'b-', 'LineWidth', 1.5);
xlabel('t');
ylabel('y');
title('Решение первого уравнения');
grid on;

% Фазовый портрет
dy_vals = (t .* y - y .* sqrt(1 + t.^2)) ./ (1 + t.^2);

figure;
plot(y, dy_vals, 'r-', 'LineWidth', 1.5);
xlabel('y');
ylabel("dy/dt");
title('Фазовый портрет первого уравнения');
grid on;



% Решение системы уравнений:
% dx/dt = 2x - y, dy/dt = x + 2y, x(0) = -1, y(0) = 0
tspan = [0, 1]; % Интервал времени
xy0 = [-1; 0]; % Начальные условия [x(0); y(0)]

% Определение правой части системы
dxy = @(t, xy) [2 * xy(1) - xy(2); xy(1) + 2 * xy(2)];

% Решение системы с использованием метода Рунге-Кутты 4-го порядка
options = odeset('RelTol', 1e-8, 'AbsTol', 1e-10);  % Установка параметров точности
[t, xy] = ode45(dxy, tspan, xy0, options);

% Построение графиков x(t) и y(t)
figure;
plot(t, xy(:, 1), 'r-', 'LineWidth', 1.5); hold on;
plot(t, xy(:, 2), 'g-', 'LineWidth', 1.5);
xlabel('t');
ylabel('x(t), y(t)');
legend('x(t)', 'y(t)');
title('Решение системы уравнений');
grid on;



% Параметры и шаги дискретизации
h_values = [1, 0.5, 0.1, 0.01, 0.001, 0.0001]; % Шаги дискретизации
t_end = 5; % Конечное время
figure;
hold on;

for h = h_values
    N = ceil(t_end / h) + 1; % Количество точек
    t = linspace(0, t_end, N); % Временная шкала
```

```
    y = zeros(1, N); % Значения y
    y(1) = 1; % Начальное значение y

    for k = 1:N-1
        % Вычисление y(k+1) методом Эйлера
        y_prime = (t(k) * y(k) - y(k) * sqrt(1 + t(k)^2)) / (1 + t(k)^2);
        y(k+1) = y(k) + h * y_prime;
    end

    plot(t, y, 'LineWidth', 1.5, 'DisplayName', sprintf('h = %.2f', h));
end

xlabel('t');
ylabel('y');
title('Дискретное решение первого уравнения с разными шагами');
legend show;
grid on;



% Параметры и шаги дискретизации
h_values = [1, 0.5, 0.1, 0.01, 0.001, 0.0001]; % Шаги дискретизации
t_end = 5; % Конечное время
figure;
hold on;

for h = h_values
    N = ceil(t_end / h) + 1; % Количество точек
    t = linspace(0, t_end, N); % Временная шкала

    x = zeros(1, N); % Значения x
    y = zeros(1, N); % Значения y
    x(1) = -1; % Начальное значение x
    y(1) = 0; % Начальное значение y

    for k = 1:N-1
        % Вычисление x(k+1) и y(k+1) методом Эйлера
        x_next = x(k) + h * (2*x(k) - y(k));
        y_next = y(k) + h * (x(k) + 2*y(k));
        x(k+1) = x_next;
        y(k+1) = y_next;
    end

    plot(t, x, 'LineWidth', 1.5, 'DisplayName', sprintf('h = %.2f', h));
end

xlabel('t');
ylabel('x');
title('Дискретное решение второго уравнения с разными шагами');
legend show;
grid on;



% Открываем новую модель Simulink
modelName = 'my_model_fixed';
open_system(new_system(modelName));

% Блок Constant (вместо Clock)
add_block('simulink/Sources/Constant', [modelName, '/Time']);
set_param([modelName, '/Time'], 'Value', '0.5', 'Position', [50, 50, 100, 100]);
```

```
% Блок Интегратора для решения уравнения
add_block('simulink/Continuous/Integrator', [modelName, '/Integrator']);
set_param([modelName, '/Integrator'], 'Position', [200, 50, 250, 100]);

% Блок для вычисления sqrt(1 + t^2)
add_block('simulink/Math Operations/Math Function', [modelName, '/Sqrt']);
set_param([modelName, '/Sqrt'], 'Function', 'sqrt', 'Position', [50, 150, 100, 200]);

% Блок для возведения t^2
add_block('simulink/Math Operations/Product', [modelName, '/Square']);
set_param([modelName, '/Square'], 'Position', [50, 100, 100, 150]);

% Блок для 1 + t^2
add_block('simulink/Math Operations/Sum', [modelName, '/Sum1']);
set_param([modelName, '/Sum1'], 'Inputs', '|++', 'Position', [150, 100, 200, 150]);

% Блок для t * y
add_block('simulink/Math Operations/Product', [modelName, '/Product1']);
set_param([modelName, '/Product1'], 'Position', [150, 200, 200, 250]);

% Блок для y * sqrt(1 + t^2)
add_block('simulink/Math Operations/Product', [modelName, '/Product2']);
set_param([modelName, '/Product2'], 'Position', [150, 250, 200, 300]);

% Блок для t * y - y * sqrt(1 + t^2)
add_block('simulink/Math Operations/Sum', [modelName, '/Sum2']);
set_param([modelName, '/Sum2'], 'Inputs', '|+-', 'Position', [250, 200, 300, 250]);

% Блок для деления на (1 + t^2)
add_block('simulink/Math Operations/Divide', [modelName, '/Divide']);
set_param([modelName, '/Divide'], 'Position', [300, 250, 350, 300]);

% Блок Scope для графика решения
add_block('simulink/Sinks/Scope', [modelName, '/Scope']);
set_param([modelName, '/Scope'], 'Position', [400, 50, 450, 100]);

% Блок XY Graph для фазового портрета
add_block('simulink/Sinks/XY Graph', [modelName, '/XY Graph']);
set_param([modelName, '/XY Graph'], 'Position', [400, 150, 450, 200]);

% Соединяем блоки

% Подключаем блоки для вычисления 1 + t^2
add_line(modelName, 'Time/1', 'Square/1');  % t -> t^2
add_line(modelName, 'Square/1', 'Sum1/2'); % t^2 -> 1 + t^2
add_block('simulink/Sources/Constant', [modelName, '/Constant_One']);
set_param([modelName, '/Constant_One'], 'Value', '1', 'Position', [100, 50, 150, 100]);
add_line(modelName, 'Constant_One/1', 'Sum1/1'); % 1 -> 1 + t^2
add_line(modelName, 'Sum1/1', 'Sqrt/1');  % 1 + t^2 -> sqrt(1 + t^2)

% Вычисляем t * y
add_line(modelName, 'Time/1', 'Product1/1'); % t
add_line(modelName, 'Integrator/1', 'Product1/2'); % y

% Вычисляем y * sqrt(1 + t^2)
add_line(modelName, 'Integrator/1', 'Product2/1'); % y
add_line(modelName, 'Sqrt/1', 'Product2/2');       % sqrt(1 + t^2)

% Подключаем к сумматору t * y - y * sqrt(1 + t^2)
add_line(modelName, 'Product1/1', 'Sum2/1'); % t * y
add_line(modelName, 'Product2/1', 'Sum2/2'); % - y * sqrt(1 + t^2)
```

```
% Деление на (1 + t^2)
add_line(modelName, 'Sum2/1', 'Divide/1'); % числитель
add_line(modelName, 'Sum1/1', 'Divide/2'); % знаменатель

% Связываем делитель с интегратором
add_line(modelName, 'Divide/1', 'Integrator/1'); % dy/dt -> y

% Подключаем Scope
add_line(modelName, 'Integrator/1', 'Scope/1'); % y -> Scope

% Подключаем XY Graph для фазового портрета
add_line(modelName, 'Integrator/1', 'XY Graph/1'); % y -> ось X
add_line(modelName, 'Divide/1', 'XY Graph/2');     % dy/dt -> ось Y

% Настраиваем начальное условие интегратора
set_param([modelName, '/Integrator'], 'InitialCondition', '1');

% Настраиваем параметры моделирования
set_param(modelName, 'Solver', 'ode45', 'StartTime', '0', 'StopTime', '10', 'MaxStep', '0.01');

% Запускаем моделирование
sim(modelName);

% Открываем Scope
open_system([modelName, '/Scope']);



% Название модели
modelName = 'second_equation_model_adjusted'; % Уникальное имя модели
new_system(modelName); % Создание новой модели

% Открываем модель
open_system(modelName);

%% Добавление блоков в модель Simulink

% Позиции блоков (x, y, ширина, высота)
blockSize = [30, 30]; % Размер блоков

% Добавляем интеграторы для x и y
add_block('simulink/Continuous/Integrator', [modelName '/Integrator_x'], ...
    'Position', [100, 100, 100+blockSize(1), 100+blockSize(2)], ...
    'InitialCondition', '0.5'); % Начальное условие для x(t)

add_block('simulink/Continuous/Integrator', [modelName '/Integrator_y'], ...
    'Position', [100, 200, 100+blockSize(1), 200+blockSize(2)], ...
    'InitialCondition', '0'); % Начальное условие для y(t)

% Добавляем сумматоры для dx/dt и dy/dt
add_block('simulink/Math Operations/Sum', [modelName '/Sum_x'], ...
    'Position', [250, 100, 250+blockSize(1), 100+blockSize(2)], ...
    'Inputs', '++');

add_block('simulink/Math Operations/Sum', [modelName '/Sum_y'], ...
    'Position', [250, 200, 250+blockSize(1), 200+blockSize(2)], ...
    'Inputs', '+-');

% Добавляем коэффициенты (Gain) для x и y
add_block('simulink/Math Operations/Gain', [modelName '/Gain_4x'], ...
    'Position', [400, 50, 400+blockSize(1), 50+blockSize(2)], ...
    'Gain', '4'); % Увеличенный коэффициент для x
```

```
add_block('simulink/Math Operations/Gain', [modelName '/Gain_1y'], ...
    'Position', [400, 150, 400+blockSize(1), 150+blockSize(2)], ...
    'Gain', '1'); % Коэффициент для y

add_block('simulink/Math Operations/Gain', [modelName '/Gain_neg05x'], ...
    'Position', [400, 250, 400+blockSize(1), 250+blockSize(2)], ...
    'Gain', '-0.5'); % Отрицательный коэффициент для x

add_block('simulink/Math Operations/Gain', [modelName '/Gain_2y'], ...
    'Position', [400, 350, 400+blockSize(1), 350+blockSize(2)], ...
    'Gain', '2'); % Увеличенный коэффициент для y

% Добавляем линии соединений
% Линии для dx/dt = 4x - y
add_line(modelName, 'Integrator_x/1', 'Gain_4x/1');
add_line(modelName, 'Integrator_y/1', 'Gain_1y/1');
add_line(modelName, 'Gain_4x/1', 'Sum_x/1');
add_line(modelName, 'Gain_1y/1', 'Sum_x/2');
add_line(modelName, 'Sum_x/1', 'Integrator_x/1', 'autorouting', 'on');

% Линии для dy/dt = -0.5x - 2y
add_line(modelName, 'Integrator_x/1', 'Gain_neg05x/1');
add_line(modelName, 'Integrator_y/1', 'Gain_2y/1');
add_line(modelName, 'Gain_neg05x/1', 'Sum_y/1');
add_line(modelName, 'Gain_2y/1', 'Sum_y/2');
add_line(modelName, 'Sum_y/1', 'Integrator_y/1', 'autorouting', 'on');

% Добавляем блок Scope для отображения x(t) и y(t)
add_block('simulink/Sinks/Scope', [modelName '/Scope'], ...
    'Position', [600, 150, 650, 200]);

% Настройка Scope для отображения двух сигналов
set_param([modelName, '/Scope'], 'NumInputPorts', '2'); % Установка двух входов

% Соединяем Scope с выходами интеграторов
add_line(modelName, 'Integrator_x/1', 'Scope/1'); % Подключаем x(t)
add_line(modelName, 'Integrator_y/1', 'Scope/2'); % Подключаем y(t)

%% Сохранение и запуск модели
save_system(modelName);
disp('Модель Simulink успешно создана.');

% Открываем модель
open_system(modelName);

% Запуск симуляции на 5 секунд
simOut = sim(modelName, 'StopTime', '5');

% Отображение результатов
disp('Симуляция завершена. Проверьте результаты на Scope.');


% Очистка рабочей среды и закрытие моделей
clear; clc; close all;
bdclose all;

%% Название модели
modelName = 'linearSystemModel';
new_system(modelName); % Создание новой модели
open_system(modelName);

%% Параметры системы
% dx/dt = 2x - y
```

```
% dy/dt = x + 2y

% Начальные условия
x0 = -1; % x(0)
y0 = 0;  % y(0)

% Размеры блоков для удобства размещения
blockWidth = 50;
blockHeight = 30;

% Позиции блоков
x_pos = 100; y_pos = 100; % Интеграторы
x_out_pos = x_pos + 300; % Выход x
y_out_pos = y_pos + 200; % Выход y

%% Добавление блоков

% Интеграторы для x и y
add_block('simulink/Continuous/Integrator', [modelName '/Integrator_x'], ...
    'Position', [x_pos, y_pos, x_pos + blockWidth, y_pos + blockHeight], ...
    'InitialCondition', num2str(x0));

add_block('simulink/Continuous/Integrator', [modelName '/Integrator_y'], ...
    'Position', [x_pos, y_pos + 100, x_pos + blockWidth, y_pos + 100 + blockHeight], ...
    'InitialCondition', num2str(y0));

% Сумматоры для dx/dt и dy/dt
add_block('simulink/Math Operations/Sum', [modelName '/Sum_x'], ...
    'Inputs', '++', ...
    'Position', [x_pos - 100, y_pos, x_pos - 100 + blockWidth, y_pos + blockHeight]);

add_block('simulink/Math Operations/Sum', [modelName '/Sum_y'], ...
    'Inputs', '++', ...
    'Position', [x_pos - 100, y_pos + 100, x_pos - 100 + blockWidth, y_pos + 100 + blockHeight]);

% Gain блоки для коэффициентов
add_block('simulink/Math Operations/Gain', [modelName '/Gain_2x'], ...
    'Gain', '2', ...
    'Position', [x_pos - 200, y_pos, x_pos - 200 + blockWidth, y_pos + blockHeight]);

add_block('simulink/Math Operations/Gain', [modelName '/Gain_neg_y'], ...
    'Gain', '-1', ...
    'Position', [x_pos - 200, y_pos + 50, x_pos - 200 + blockWidth, y_pos + 50 + blockHeight]);

add_block('simulink/Math Operations/Gain', [modelName '/Gain_x'], ...
    'Gain', '1', ...
    'Position', [x_pos - 200, y_pos + 100, x_pos - 200 + blockWidth, y_pos + 100 + blockHeight]);

add_block('simulink/Math Operations/Gain', [modelName '/Gain_2y'], ...
    'Gain', '2', ...
    'Position', [x_pos - 200, y_pos + 150, x_pos - 200 + blockWidth, y_pos + 150 + blockHeight]);

% Scopes для визуализации результатов
add_block('simulink/Sinks/Scope', [modelName '/Scope_x'], ...
    'Position', [x_out_pos, y_pos, x_out_pos + blockWidth, y_pos + blockHeight]);

add_block('simulink/Sinks/Scope', [modelName '/Scope_y'], ...
    'Position', [x_out_pos, y_pos + 100, x_out_pos + blockWidth, y_pos + 100 + blockHeight]);

%% Соединение блоков

% Сумматоры -> Интеграторы
add_line(modelName, 'Sum_x/1', 'Integrator_x/1');
```

```
add_line(modelName, 'Sum_y/1', 'Integrator_y/1');

% Интеграторы -> Scopes
add_line(modelName, 'Integrator_x/1', 'Scope_x/1');
add_line(modelName, 'Integrator_y/1', 'Scope_y/1');

% Gain блоки -> Сумматоры (dx/dt = 2x - y)
add_line(modelName, 'Gain_2x/1', 'Sum_x/1');
add_line(modelName, 'Gain_neg_y/1', 'Sum_x/2');

% Gain блоки -> Сумматоры (dy/dt = x + 2y)
add_line(modelName, 'Gain_x/1', 'Sum_y/1');
add_line(modelName, 'Gain_2y/1', 'Sum_y/2');

% Обратная связь от интеграторов -> Gain блоки
add_line(modelName, 'Integrator_x/1', 'Gain_2x/1');
add_line(modelName, 'Integrator_x/1', 'Gain_x/1');
add_line(modelName, 'Integrator_y/1', 'Gain_neg_y/1');
add_line(modelName, 'Integrator_y/1', 'Gain_2y/1');

%% Сохранение и запуск модели
save_system(modelName);
disp('Модель Simulink успешно создана и сохранена.');

% Запуск симуляции
simOut = sim(modelName, 'StartTime', '0', 'StopTime', '10');

% Открытие модели
open_system(modelName);




% Начальные условия
y0 = 1;          % y(0)
tspan = [0, 5];   % Временной интервал

% Шаги дискретизации
h_values = [1, 0.5, 0.1, 0.05, 0.01];

% Определение правой части уравнения
dydt = @(t, y) (t * y - y * sqrt(1 + t^2)) / (1 + t^2);

figure; hold on;

% Цикл по разным шагам дискретизации
for h = h_values
    % Число точек дискретизации
    N = ceil((tspan(2) - tspan(1)) / h) + 1;
    t = linspace(tspan(1), tspan(2), N);

    % Инициализация массива для y
    y = zeros(1, N);
    y(1) = y0; % Начальное значение

    % Метод Эйлера
    for n = 1:N-1
        y(n+1) = y(n) + h * dydt(t(n), y(n));
    end

    % Построение графика
    plot(t, y, 'LineWidth', 1.5, 'DisplayName', sprintf('h = %.4f', h));
end
```

```
xlabel('t');
ylabel('y');
title('Решение методом Эйлера для различных шагов h');
legend show;
grid on;


% Решение уравнения с ode45
[t_ode, y_ode] = ode45(dydt, tspan, y0);

% График решения
figure;
plot(t_ode, y_ode, 'b-', 'LineWidth', 1.5);
xlabel('t');
ylabel('y');
title('Решение уравнения с помощью ode45');
grid on;


% Вычисление производной y' для фазового портрета
dy_vals = (t_ode .* y_ode - y_ode .* sqrt(1 + t_ode.^2)) ./ (1 + t_ode.^2);

% Построение фазового портрета
figure;
plot(y_ode, dy_vals, 'r-', 'LineWidth', 1.5);
xlabel('y');
ylabel("dy/dt");
title('Фазовый портрет');
grid on;




% Решение системы уравнений методом Эйлера
% dx/dt = 2x - y, dy/dt = x + 2y, x(0) = -1, y(0) = 0

% Начальные условия
x0 = -1; % x(0)
y0 = 0;  % y(0)
tspan = [0, 1]; % Временной интервал

% Шаги дискретизации
h_values = [1, 0.5, 0.1, 0.01];

figure; hold on;

% Цикл по разным шагам дискретизации
for h = h_values
    N = ceil((tspan(2) - tspan(1)) / h) + 1;
    t = linspace(tspan(1), tspan(2), N);

    % Инициализация массивов
    x = zeros(1, N);
    y = zeros(1, N);

    % Начальные условия
    x(1) = x0;
    y(1) = y0;

    % Метод Эйлера
    for n = 1:N-1
        dx = 2 * x(n) - y(n);
```

```
        dy = x(n) + 2 * y(n);
        x(n+1) = x(n) + h * dx;
        y(n+1) = y(n) + h * dy;
    end

    % Построение графиков x(t) и y(t)
    plot(t, x, 'LineWidth', 1.5, 'DisplayName', sprintf('x(t), h = %.4f', h));
    plot(t, y, '--', 'LineWidth', 1.5, 'DisplayName', sprintf('y(t), h = %.4f', h));
end

xlabel('t');
ylabel('x(t), y(t)');
title('Дискретизация системы уравнений');
legend show;
grid on;
```