МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ИНСТИТУТ НЕПРЕРЫВНОГО И ДИСТАНЦИОННОГО ОБРАЗОВАНИЯ

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

ОЦЕНКА

ПРЕПОДАВАТЕЛЬ

д-р техн. наук, профессор		С.И. Колесникова
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

Моделирование динамических систем в Симулинк

по дисциплине: Компьютерное моделирование

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ гр. №	Z1431		М.Д. Быстров
	номер группы	подпись, дата	инициалы, фамилия

Студенческий билет № 2021/3572

Санкт-Петербург 2024

ЦЕЛЬ РАБОТЫ

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

ЗАДАНИЕ

1. Самостоятельно ознакомиться со справочными сведениями относительно приложения MatLab Simulink.

2. Построить фазовый портрет и графики во временной области непрерывной 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 №1

1) $y' + 2xy = xe^{-x^2}$, $y(0) = 2.$

2) $\begin{cases} \dfrac{dx}{dt} = 4x - y \\ \dfrac{dy}{dt} = x + 2y \end{cases}$, $x(0) = -1$, $y(0) = 0.$

ХОД РАБОТЫ

1. Построение графиков решения дифференциального уравнения и фазового портрета

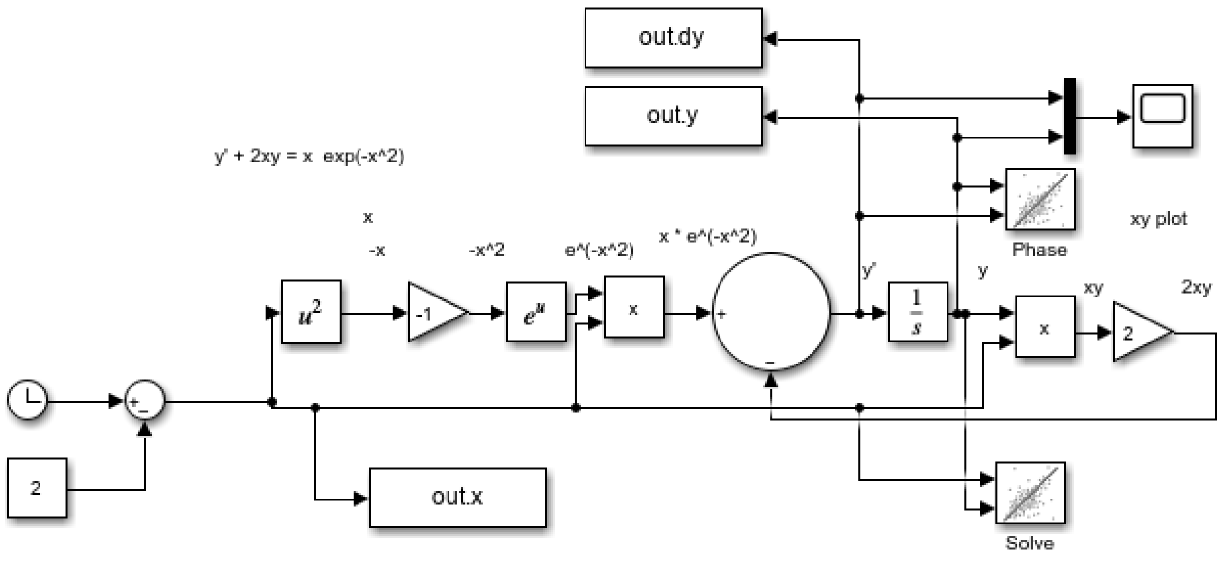
Для решения дифференциального уравнения в Matlab может быть применена функция ode45.

```
% Определение функции, задающей дифференциальное уравнение
diff_eq = @(x, y) x * exp(-x^2) - 2*x*y;

% Решение дифференциального уравнения для заданного диапазона x
x = linspace(-2, 2, 1000);
y0 = 2; % Начальное условие y(0) = 2
[t, y] = ode45(diff_eq, x, y0);

% получение значения производной для каждого значения функции и
аргумента
% для построения фазового портрета
diffValues = zeros(size(y, 1), 1);

for i = 1:size(y,1)
    diffValues(i) = diff_eq(x(i), y(i));
end

% Построение фазового портрета
figure
subplot(1, 2, 1);
hold on;
plot(diffValues, y);
plot(out.dy, out.y);
hold off;
xlabel('dy/dx');
ylabel('y');
title('Фазовый портрет');

% Построение графика решения
subplot(1, 2, 2);
hold on;
plot(t, y);
plot(out.x, out.y);
hold off;
xlabel('x');
ylabel('y');
    title('График');
```

2. Построение модели Simulink для решения дифференциального

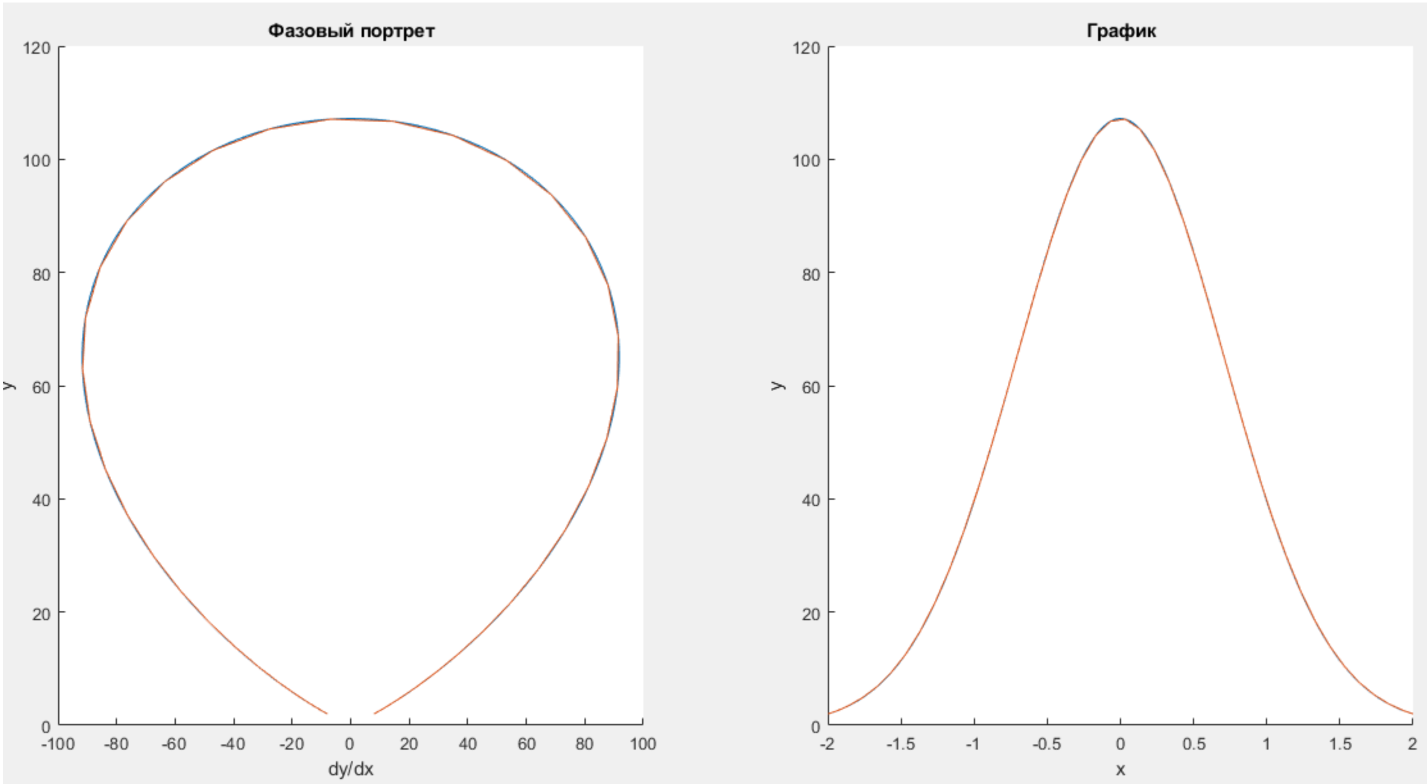
уравнения

Модель Simulink для решения дифференциального уравнения имеет следующий вид:



Значение аргумента x задается элементом Clock.

Для сравнения расчетов Matlab и Simulink фазовые портреты и графики решения отображены на одних осях.



Графики (красное начертание – Simulink, синее начертание – скрипт Matlab) совпадают, что говорит о корректности построенной модели.

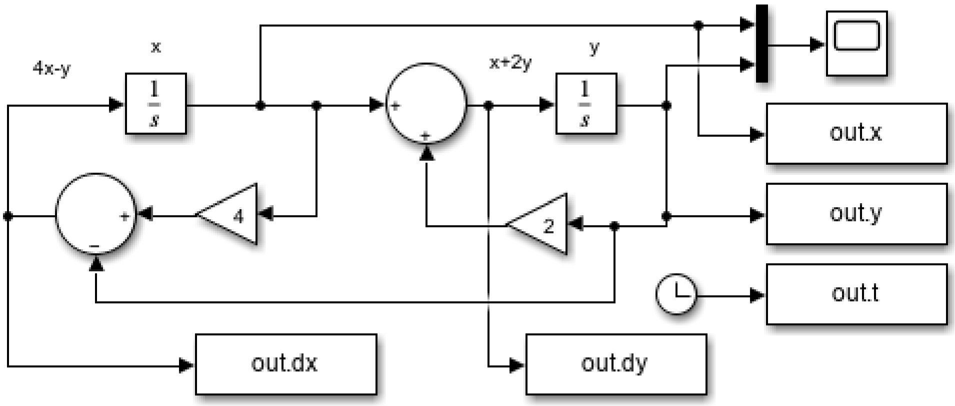
3. Построение фазового портрета и графиков системы дифференциальных уравнений

Для поиска решения системы дифференциальных уравнений в Matlab также используется функция ode45. Система уравнений может быть решена с помощью следующего набора команд:

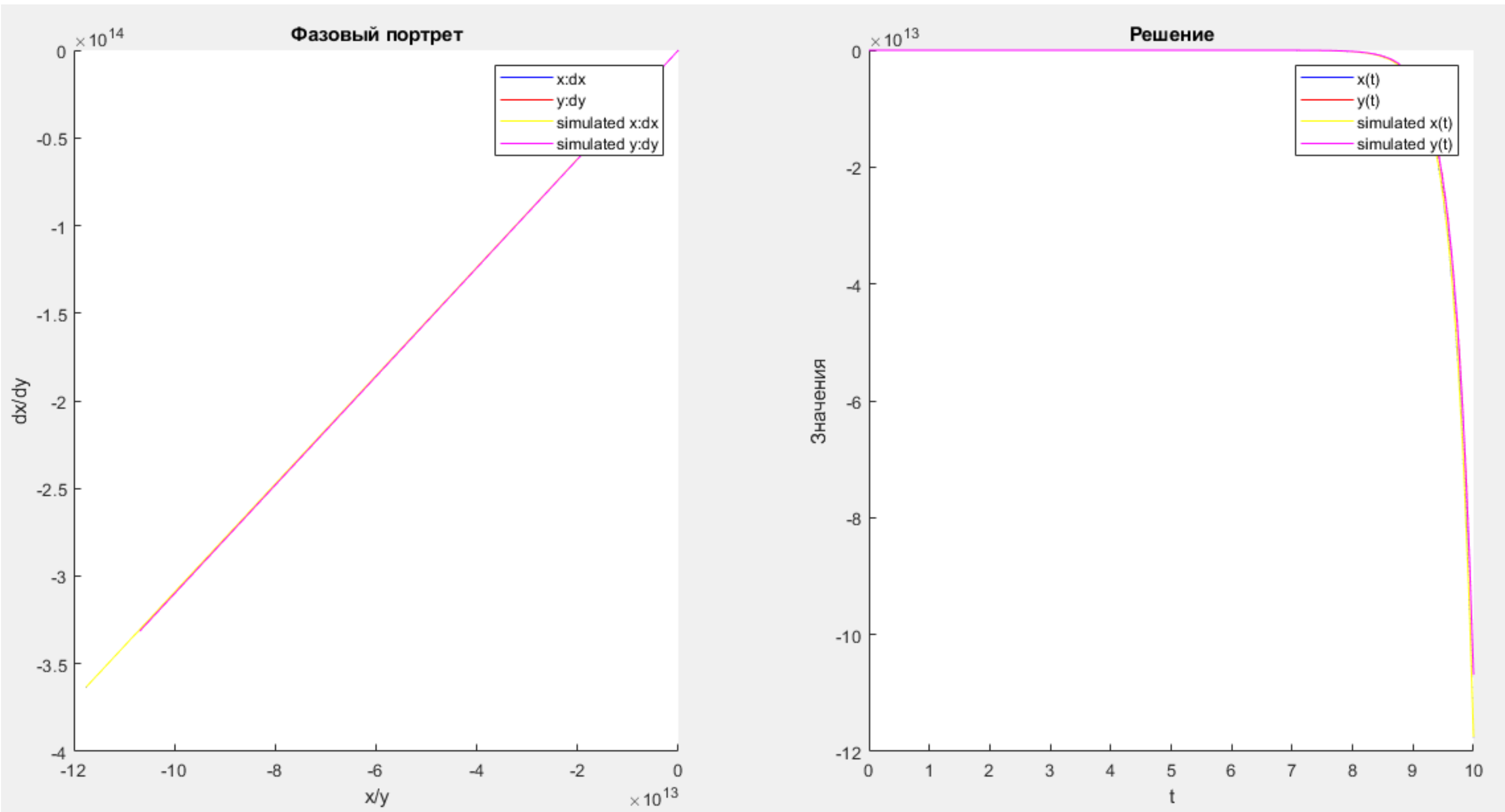
```
% Определение системы дифференциальных уравнений
dxdt = @(t, x) 4*x(1) - x(2);
dydt = @(t, x) x(1) + 2*x(2);

% Начальные условия
x0 = -1;
y0 = 0;
tspan = [0 10]; % Диапазон времени для решения

% Решение системы дифференциальных уравнений
[t, sol] = ode45(@(t, x) [dxdt(t, x); dydt(t, x)], tspan, [x0; y0]);

% Нахождение значений производной для построения фазового портрета
dx = zeros(size(t, 1), 1);
dy = zeros(size(t, 1), 1);

for i = 1 : size(t, 1)
    dx(i) = dxdt(t(i), sol(i, :));
    dy(i) = dydt(t(i), sol(i, :));
end

% Построение фазового портрета
figure
subplot(1, 2, 1);
hold on;
plot(sol(:, 1), dx, "b");
plot(sol(:, 2), dy, "r");
plot(out.x, out.dx, "y");
plot(out.y, out.dy, "m");
hold off;
legend('x:dx', 'y:dy', 'simulated x:dx', 'simulated y:dy');
xlabel('x/y');
ylabel('dx/dy');
title('Фазовый портрет');

% Построение графиков x(t) и y(t)
subplot(1, 2, 2);
hold on;
plot(t, sol(:, 1), 'b', t, sol(:, 2), 'r');
plot(out.t, out.x, 'y', out.t, out.y, 'm');
hold off;
legend('x(t)', 'y(t)', 'simulated x(t)', 'simulated y(t)');
xlabel('t');
ylabel('Значения');
title('Решение');
```

4. Построение модели Simulink для решения системы дифференциальных уравнений

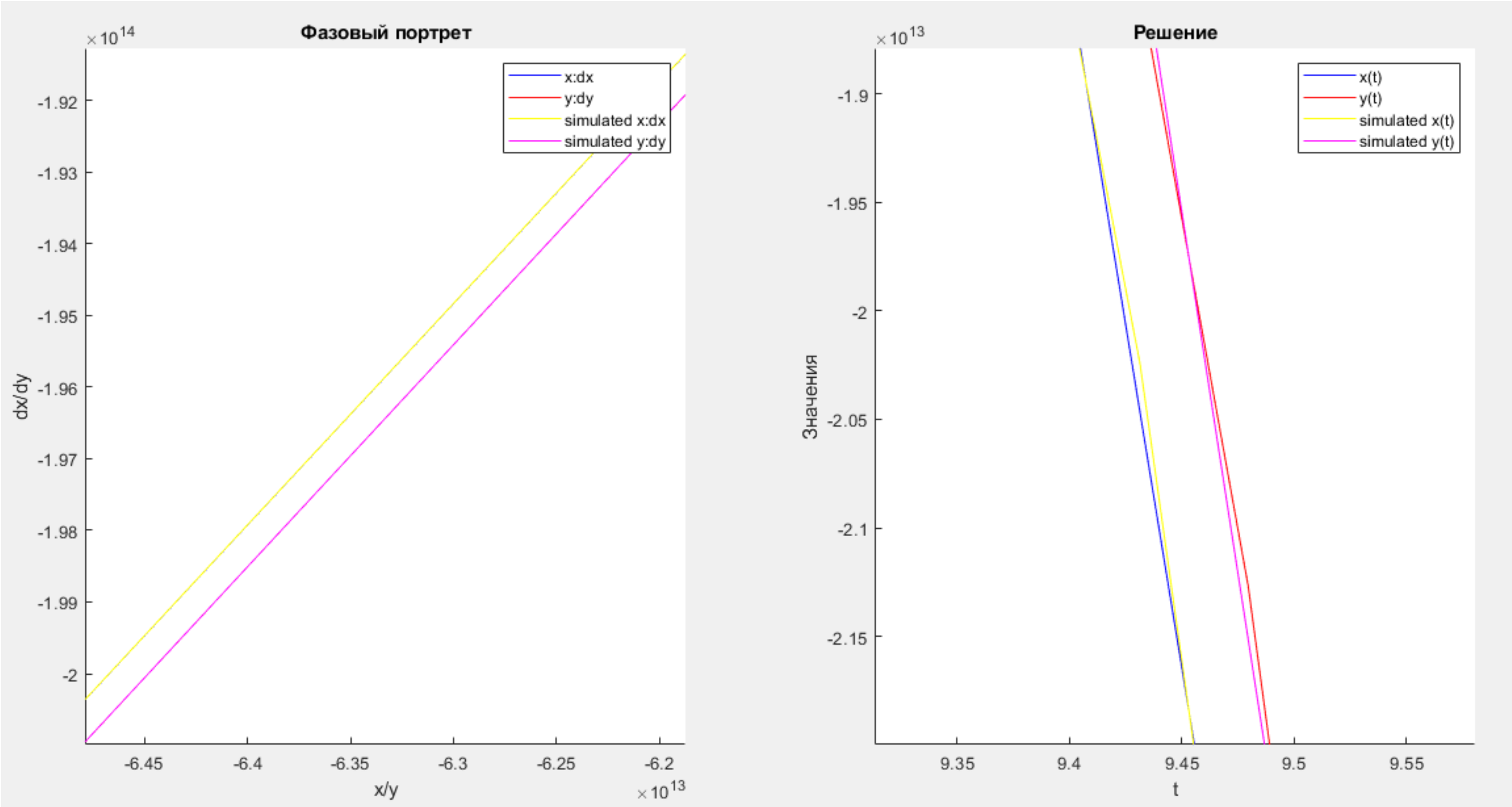
Для решения системы дифференциальных уравнений в Simulink была создана модель:



Построены сравнительные графики фазового портрета и решения, полученные из расчетов Matlab и модели Simulink.



В более крупном масштабе видна разница в точности между расчетами:

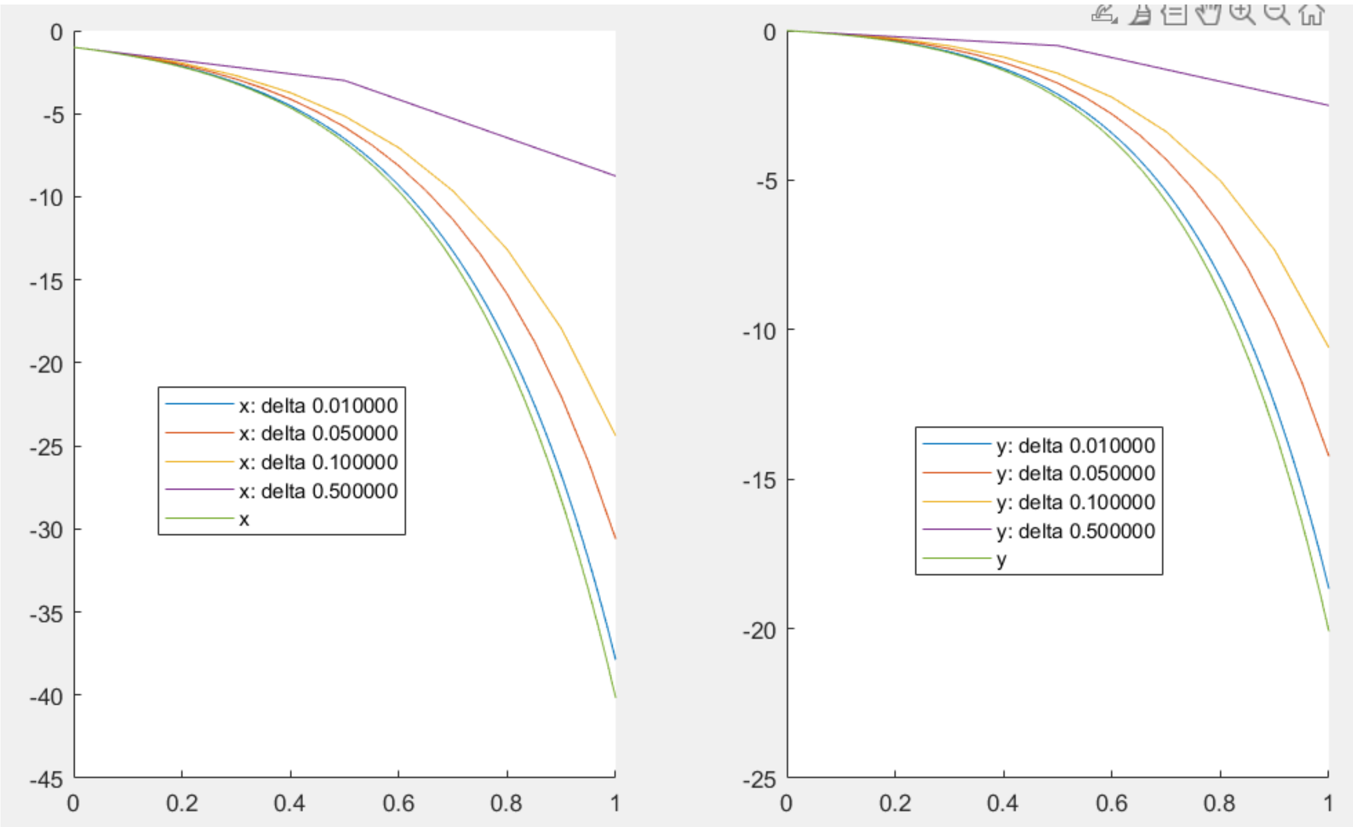


5. Построение графиков дискретной (нелинейной) модели решения разностного уравнения.

Для построения графиков дискретного решения разностного уравнения была реализована модель решения разностного уравнения с помощью метода Эйлера.

```
% решение системы двух дифференциальных уравнений методом
дискретизации
% Эйлера
function [t, x, y] = eiler(nextX, nextY, startX, startY, fromT, toT,
delta)

    % кол-во шагов алгоритма для заданного параметра дискретизации
    steps = int32((toT - fromT) ./ delta) + 1;

    t = zeros(steps, 1);
    x = zeros(steps, 1);
    y = zeros(steps, 1);

    prevX = startX;
    prevY = startY;

    % первые элементы - начальные значения
    x(1) = startX;
    y(1) = startY;
    t(1) = fromT;

    for i = 2:steps

        x(i) = nextX(prevX, prevY, delta);
        y(i) = nextY(prevX, prevY, delta);

        localT = delta * double(i - 1);
        t(i) = localT;

        prevX = x(i);
        prevY = y(i);
    end
end
```

Построены графики решения системы дифференциального уравнения для каждой из функций:



На левом графике отображены функция x и показания её моделей дискретизации, на правом – соответствующие графики для функции y.

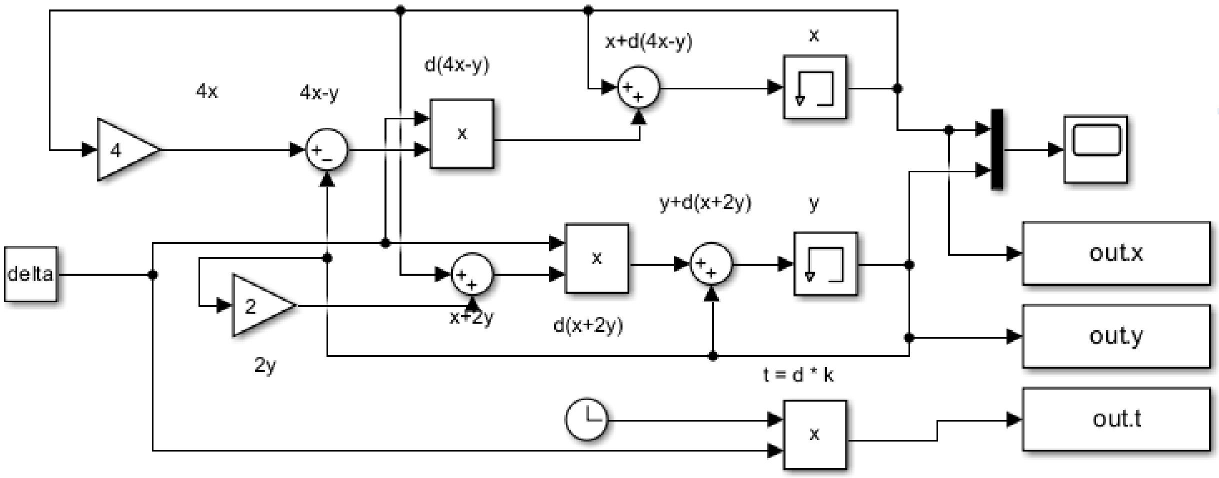
На графике видно, что ближе всего к реальной функции (зеленое начертание) находится модель, которая использует наименьший параметр дискретизации.

6. Разработка модели Simulink для решения разностного уравнения

Модель Simulink для решения системы разностных уравнений

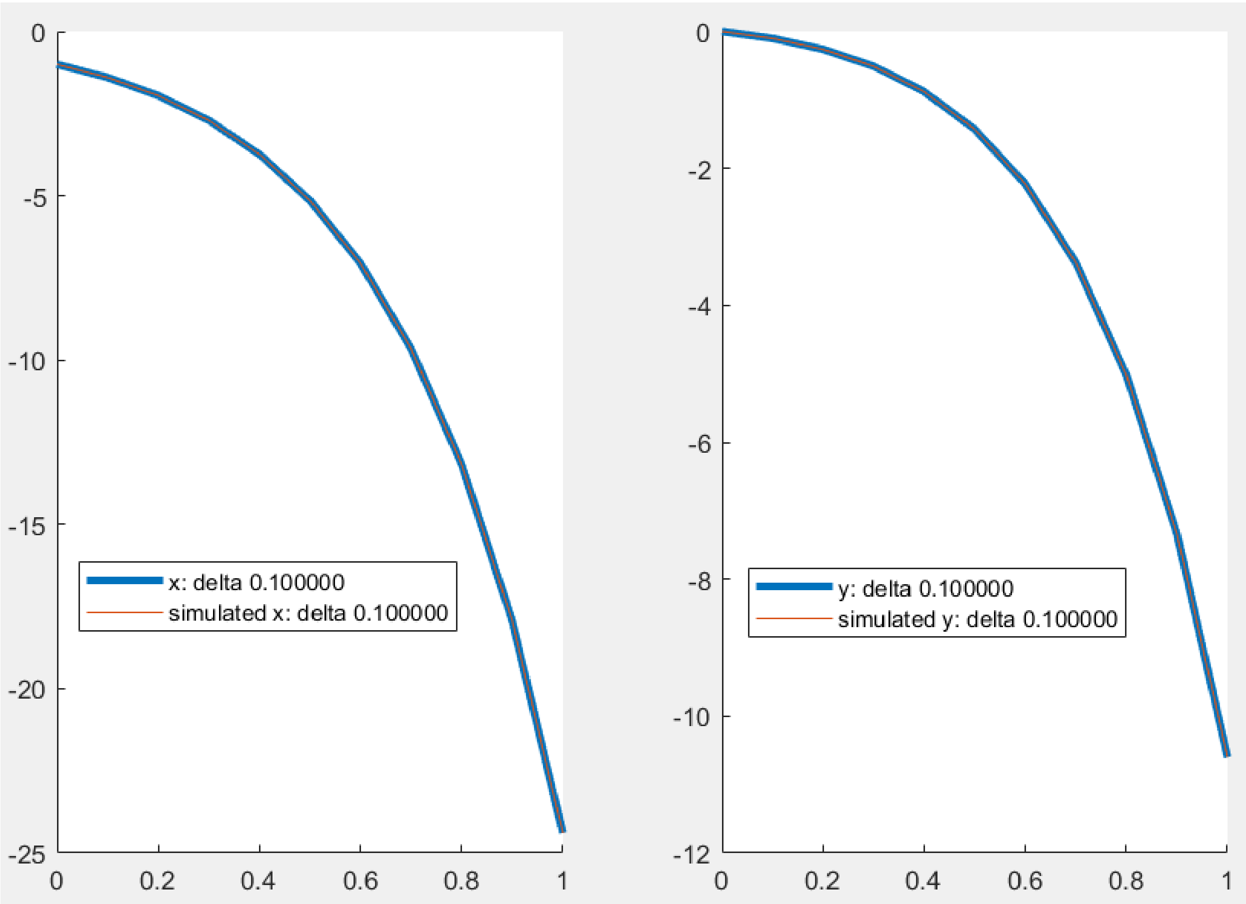
$$x(t+1) = x(t) + d(4x(t) - y(t))$$
$$y(t+1) = y(t) + d(x(t) + 2y(t))$$

полученных из системы дифференциальных уравнений методом дискретизации Эйлера, имеет вид:



Для хранения значений функции между итерациями используется блок "Memory" с установкой начальных значений для x(0) и y(0).

Выполнена проверка совпадения расчетов с помощью инструкций и модели Simulink.



На графиках видно, что решения совпадают.

ВЫВОДЫ

В ходе выполнения лабораторной работы №3 были созданы модели для решения дифференциальных уравнений, а также систем дифференциальных и разностных уравнений в системе Simulink.

Написана программа для проверки корректной работы модели решения дифференциального уравнения и системы дифференциальных, произведено сравнение фазовых графиков и графиков решения.

Система уравнения дискретизирована по схеме Эйлера, приведены графики сравнения схемы решения системы уравнений с помощью дискретного метода и реального решения. Наибольшее приближение к реальному решению показала модель с наименьшим параметром дискретизации.

Произведена проверка Simulink модели решения системы разностных уравнений с помощью скрипта Matlab. Продемонстрирована эквивалентность решений на отрезке [0,1].

Приобретены навыки моделирования в программном пакете Matlab Simulink.

Приложение 1 Исходный код программы

1. lab3_12.m

```
% Определение функции, задающей дифференциальное уравнение
diff_eq = @(x, y) x * exp(-x^2) - 2*x*y;

% Решение дифференциального уравнения для заданного диапазона x
x = linspace(-2, 2, 1000);
y0 = 2; % Начальное условие y(0) = 2
[t, y] = ode45(diff_eq, x, y0);

% получение значения производной для каждого значения функции и
аргумента
% для построения фазового портрета
diffValues = zeros(size(y, 1), 1);

for i = 1:size(y,1)
    diffValues(i) = diff_eq(x(i), y(i));
end

% Построение фазового портрета
figure
subplot(1, 2, 1);
hold on;
plot(diffValues, y);
plot(out.dy, out.y);
hold off;
xlabel('dy/dx');
ylabel('y');
title('Фазовый портрет');

% Построение графика решения
subplot(1, 2, 2);
hold on;
plot(t, y);
plot(out.x, out.y);
hold off;
xlabel('x');
ylabel('y');
title('График');
```

2. lab3_34.m

```
% Определение системы дифференциальных уравнений
dxdt = @(t, x) 4*x(1) - x(2);
dydt = @(t, x) x(1) + 2*x(2);

% Начальные условия
x0 = -1;
y0 = 0;
tspan = [0 1]; % Диапазон времени для решения

% Решение системы дифференциальных уравнений
[t, sol] = ode45(@(t, x) [dxdt(t, x); dydt(t, x)], tspan, [x0; y0]);

% Нахождение значений производной для построения фазового портрета
dx = zeros(size(t, 1), 1);
dy = zeros(size(t, 1), 1);

for i = 1 : size(t, 1)
    dx(i) = dxdt(t(i), sol(i, :));
```

```
    dy(i) = dydt(t(i), sol(i, :));
end

% Построение фазового портрета
figure
subplot(1, 2, 1);
hold on;
plot(sol(:, 1), dx, "b");
plot(sol(:, 2), dy, "r");
plot(out.x, out.dx, "y");
plot(out.y, out.dy, "m");
hold off;
legend('x:dx', 'y:dy', 'simulated x:dx', 'simulated y:dy');
xlabel('x/y');
ylabel('dx/dy');
title('Фазовый портрет');

% Построение графиков x(t) и y(t)
subplot(1, 2, 2);
hold on;
plot(t, sol(:, 1), 'b', t, sol(:, 2), 'r');
plot(out.t, out.x, 'y', out.t, out.y, 'm');
hold off;
legend('x(t)', 'y(t)', 'simulated x(t)', 'simulated y(t)');
xlabel('t');
ylabel('Значения');
    title('Решение');
```

3. lab3_56.m

```
% x(t+1) = x(t) + d(4x(t) - y(t))
% y(t+1) = y(t) + d(x(t) + 2y(t))

nextX = @(xt, yt, delta) xt + delta .* (4 .* xt - yt);
nextY = @(xt, yt, delta) yt + delta .* (xt + 2 .* yt);

% Начальные условия
x0 = -1;
y0 = 0;

% набор различных параметров дискретизации
 % deltas = [0.01, 0.05, 0.1, 0.5];
deltas = [0.1];

deltaNum = size(deltas, 2);

% интервал расчета
fromT = 0;
toT = 1;

timeSpan = [fromT toT];

oldT = t;

for i = 1:deltaNum

    delta = deltas(i);

    steps = int32((toT - fromT) ./ delta) + 1;

    % запуск симуляции для заданного параметра дискретизации
    out = sim("lab3_3.slx");

    simulatedT = out.t(1 : steps);
    simulatedX = out.x(1:size(simulatedT, 1));
```

```
    simulatedY = out.y(1:size(simulatedT, 1));


    [t, x, y] = eiler(nextX, nextY, x0, y0, fromT, toT, delta);

    subplot(1,2,1);
    hold on;
    plot(t, x, "DisplayName", sprintf("x: delta %f", delta),
"LineWidth", 3);
    plot(simulatedT, simulatedX, "DisplayName", sprintf("simulated x:
delta %f", delta));
    hold off;
    subplot(1,2,2);
    hold on;
    plot(t, y, "DisplayName", sprintf("y: delta %f", delta),
"LineWidth", 3);
    plot(simulatedT, simulatedY, "DisplayName", sprintf("simulated y:
delta %f", delta));
    hold off;

end

% hold off;
subplot(1,2,1);
% hold on;
% plot(oldT, sol(:, 1), "DisplayName", "x");
% hold off;
legend;
subplot(1,2,2);
% hold on;
% plot(oldT, sol(:, 2), "DisplayName", "y");
% hold off;
legend;

% решение системы двух дифференциальных уравнений методом
дискретизации
% эйлера
function [t, x, y] = eiler(nextX, nextY, startX, startY, fromT, toT,
delta)

    % кол-во шагов алгоритма для заданного параметра дискретизации
    steps = int32((toT - fromT) ./ delta) + 1;

    t = zeros(steps, 1);
    x = zeros(steps, 1);
    y = zeros(steps, 1);

    prevX = startX;
    prevY = startY;

    % первые элементы - начальные значения
    x(1) = startX;
    y(1) = startY;
    t(1) = fromT;

    for i = 2:steps

        x(i) = nextX(prevX, prevY, delta);
        y(i) = nextY(prevX, prevY, delta);

        localT = delta * double(i - 1);
        t(i) = localT;

        prevX = x(i);
        prevY = y(i);
```

```
    end
end
```