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.

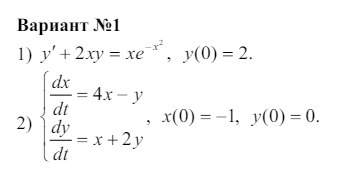
3. Разработать модель Simulink для решения дифференциального уравнения.

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.



**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;

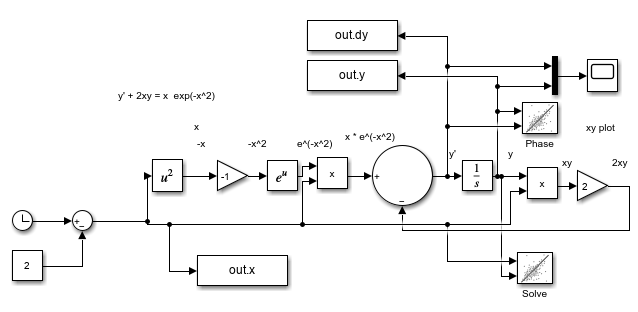
xlabel('x');

ylabel('y');

title('График');

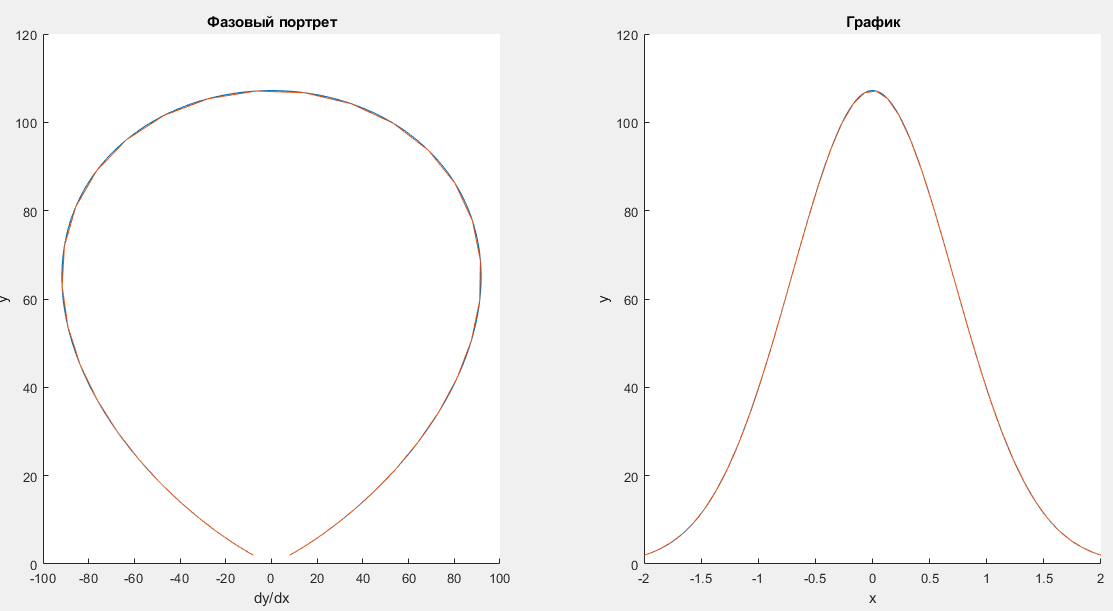
1. Построение модели Simulink для решения дифференциального уравнения

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.

1. 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)');

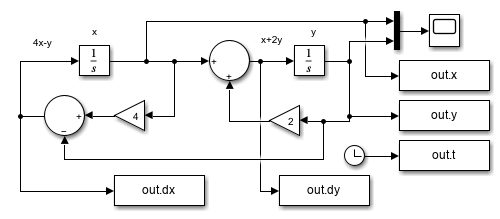
xlabel('t');

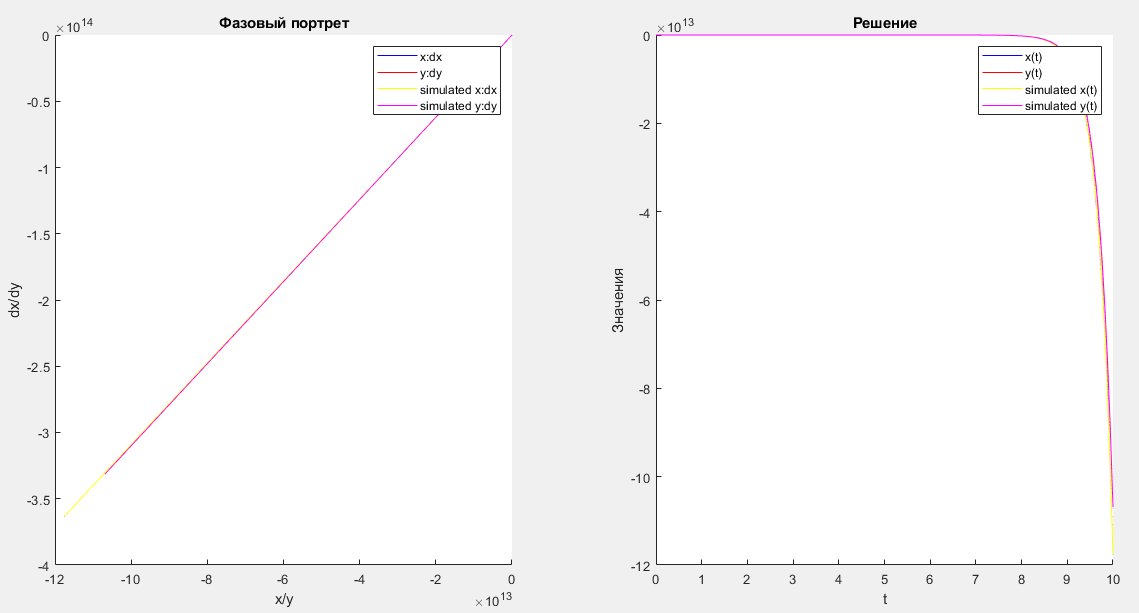
ylabel('Значения');

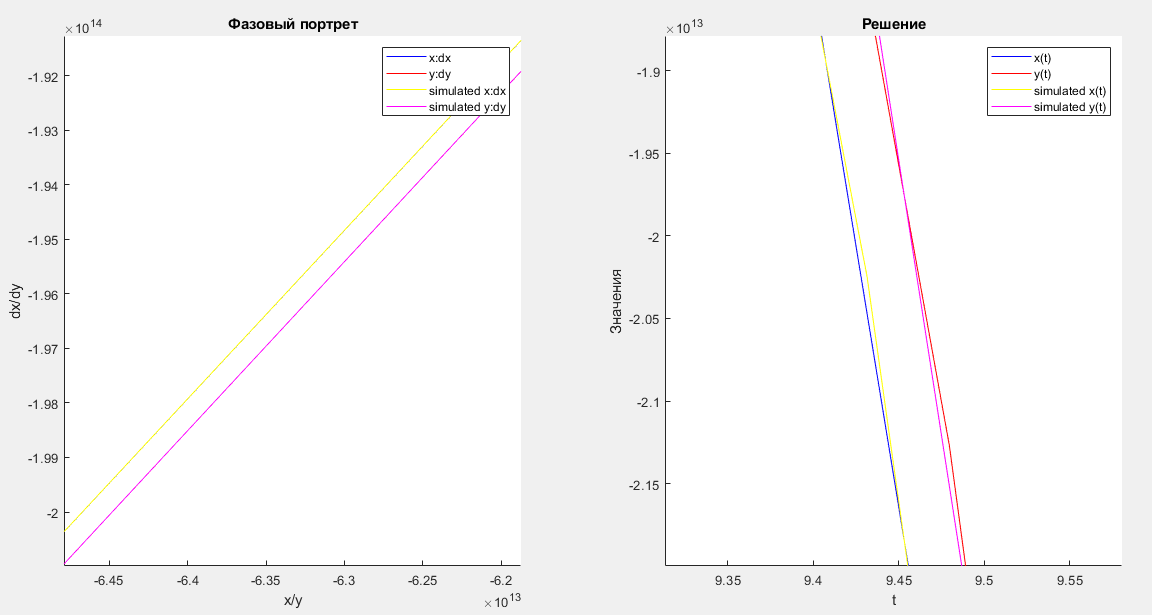
title('Решение');

1. Построение модели Simulink для решения системы дифференциальных уравнений

Для решения системы дифференциальных уравнений в Simulink была создана модель:







**ВЫВОДЫ**

В ходе выполнения первой лабораторной работы №2 были описаны функциональные модели для моделирования исходных данных.

Самостоятельно построена линейная полиномиальная модель второго порядка, имеющая вид

f1(x) = 0.0408 \* x^2 + 0.019 \* x + 8.06.

С помощью средств программного пакета Matlab построены полиномиальные модели порядков 3-6, визуально как наилучшая определена модель с порядком p = 5, имеющая вид

f2(x) = -0.0009583\*x^5 + 0.01439\*x^4 + -0.07634\*x^3 + 0.2342\*x^2 + -0.2716\*x + 8.261.

Также построена модель f3:

f3(x) = (81.1604 \* x + 462.178) ^ (1/3).

С помощью скорректированного коэффициента детерминации выбрана наилучшая модель – f2. Коэффициент имеет следующие значения для каждой из функций:

Radj f1 = 0.987704

Radj f2 = 0.999652

Radj f3 = 0.770832.

Приобретены навыки использования систем моделирования случайных временных рядов с помощью функциональных моделей.

Приложение 1 Исходный код программы

x = [1, 2, 3, 4, 5, 6, 7];

y = [8.16, 8.25, 8.41, 8.76, 9.2, 9.78, 10.1];

n = size(x, 2);

k = 1;

A = [sum(x.^4), sum(x.^3), sum(x.^2);

sum(x.^3), sum(x.^2), sum(x);

sum(x.^2), sum(x), n];

b = [sum((x.^2) .\* y);

sum(x .\* y);

sum(y)];

coeff = A\b;

syms x1

hold on;

scatter(x, y , "k");

f2 = fit(transpose(x), transpose(y), "poly3");

plot(f2, "r");

f2 = fit(transpose(x), transpose(y), "poly4");

plot(f2, "g");

f2 = fit(transpose(x), transpose(y), "poly5");

plot(f2, "b");

f2 = fit(transpose(x), transpose(y), "poly6");

plot(f2, "c");

legend("Исходные даные", "p = 3", "p = 4", "p = 5", "p = 6");

hold off;

f1 = coeff(1) \* x1 ^ 2 + coeff(2) \* x1 + coeff(3);

f2 = fit(transpose(x), transpose(y), "poly5");

f3 = (x1 \* 81.1604 + 462.178) ^ (1/3);

fprintf("Radj f1 - %s\n", f1);

fprintf("Radj f2 - %s\n", f2);

fprintf("Radj f3 - %s\n", f3);

syms r

y1 = zeros(1, n);

y2 = zeros(1, n);

y3 = zeros(1, n);

**for** i = 1:n

y1(1, i) = subs(f1, x1, i);

y2(1, i) = f2(i);

y3(1, i) = subs(f3, x1, i);

**end**

Radj = 1 - (1 - r) \* (n - 1) / (n - k);

Radj1 = subs(Radj, r, R(x, y, y1));

Radj2 = subs(Radj, r, R(x, y, y2));

Radj3 = subs(Radj, r, R(x, y, y3));

fprintf("Radj f1 - %f\n", Radj1);

fprintf("Radj f2 - %f\n", Radj2);

fprintf("Radj f3 - %f\n", Radj3);

plot(x, y, "ok");

hold on;

fplot(f1, [x(1), x(n)]);

plot(f2);

fplot(f3, [x(1), x(n)]);

legend("off");

hold off;

**function** [result] = **R**(X, Y, approximatedY)

% Посчитать коэффициент детерминации модели

% X - координаты исходных точек по X

% Y - координаты исходных точек по Y, сответствующие вектору X

% approximatedY - аппроксимированные значения Y, соответствующие вектору X

% кол-во точек

n = size(X, 2);

% функция одной переменной, k = 1

k = 1;

dispersion = sum((approximatedY - Y) .^ 2) / (n - k - 1);

result = 1 - (dispersion) / (std(Y) ^ 2);

**end**