

Лабораторная работа №6

Компьютерный практикум по статистическому анализу данных

Еюбоглу Тимур

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
2.1	Решение обыкновенных дифференциальных уравнений	6
2.2	Модель экспоненциального роста	6
2.3	Система Лоренца	8
2.4	Модель Лотки–Вольтерры	10
2.5	Самостоятельное выполнение	12
3	Вывод	18
4	Список литературы. Библиография	19

Список иллюстраций

2.1	График модели экспоненциального роста	7
2.2	График модели экспоненциального роста (задана точность решения)	8
2.3	Аттрактор Лоренца	9
2.4	Аттрактор Лоренца (интерполяция отключена)	10
2.5	Модель Лотки–Вольтерры: динамика изменения численности популяций	11
2.6	Модель Лотки–Вольтерры: фазовый портрет	12
2.7	Решение задания №1	13
2.8	Решение задания №2	14
2.9	Решение задания №3	15
2.10	Решение задания №4	15
2.11	Решение задания №5	16
2.12	Решение задания №6	16
2.13	Решение задания №7	17
2.14	Решение задания №8	17

Список таблиц

1 Цель работы

Основной целью работы является освоение специализированных пакетов для решения задач в непрерывном и дискретном времени.

2 Выполнение лабораторной работы

2.1 Решение обыкновенных дифференциальных уравнений

Вспомним, что обыкновенное дифференциальное уравнение (ОДУ) описывает изменение некоторой переменной u .

Для решения обыкновенных дифференциальных уравнений (ОДУ) в Julia можно использовать пакет `differentialEquations.jl`.

2.2 Модель экспоненциального роста

Рассмотрим пример использования этого пакета для решения уравнения модели экспоненциального роста, описываемую уравнением, где a — коэффициент роста.

Численное решение в Julia будет иметь следующий вид, а также график, соответствующий полученному решению (рис. 2.1):



Рис. 2.1: График модели экспоненциального роста

При построении одного из графиков использовался вызов `sol.t`, чтобы захватить массив моментов времени. Массив решений можно получить, воспользовавшись `sol.u`.

Если требуется задать точность решения, то можно воспользоваться параметрами `abstol` (задаёт близость к нулю) и `reltol` (задаёт относительную точность). По умолчанию эти параметры имеют значение `abstol = 1e-6` и `reltol = 1e-3`.

Для модели экспоненциального роста (рис. 2.2):



Рис. 2.2: График модели экспоненциального роста (задана точность решения)

2.3 Система Лоренца

Динамической системой Лоренца является нелинейная автономная система обыкновенных дифференциальных уравнений третьего порядка.

Система получена из системы уравнений Навье–Стокса и описывает движение воздушных потоков в плоском слое жидкости постоянной толщины при разложении скорости течения и температуры в двойные ряды Фурье с последующим усечением до первых-вторых гармоник.

Решение системы неустойчиво на аттракторе, что не позволяет применять классические численные методы на больших отрезках времени, требуется использовать высокоточные вычисления.

Численное решение в Julia будет иметь следующий вид (рис. 2.3):

1.2. Система Лоренца

```
* Задаём описание модели
function lorentz(du, u, p, t)
    θ, ρ, β = p
    du[1] = θ * (u[2] - u[1])
    du[2] = u[1] * (ρ - u[3]) - u[2]
    du[3] = u[1] * u[2] - β * u[3]
end

* Задаём начальные условия
u0 = [1.0, 0.0, 0.0]

* Задаём значения параметров
p = (10, 28, 8 / 3)

* Задаём интервал времени
tspan = (0.0, 100.0)

* Решаем
prob = ODEProblem(lorentz, u0, tspan, p)
sol = solve(prob, Tsit5())

* Строим график
plot(sol, idxs=(1, 2, 3), lw=1, title="Аттрактор Лоренца", xaxis="x", yaxis="y", zaxis="z", legend=false)
```



Рис. 2.3: Аттрактор Лоренца

Можно отключить интерполяцию (рис. 2.4):

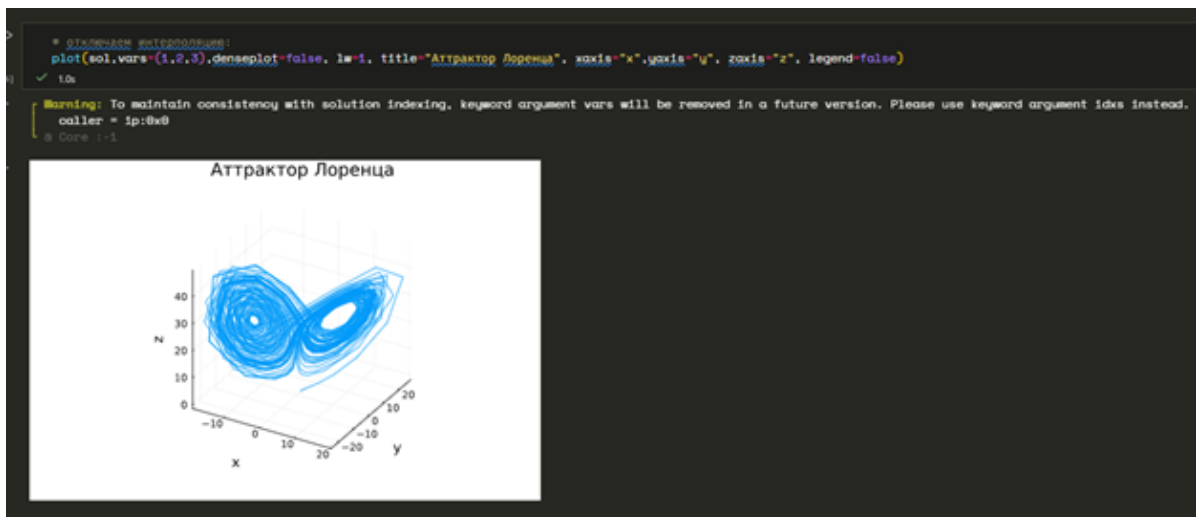


Рис. 2.4: Аттрактор Лоренца (интерполяция отключена)

2.4 Модель Лотки–Вольтерры

Модель Лотки–Вольтерры описывает взаимодействие двух видов типа «хищник – жертва».

Численное решение в Julia будет иметь следующий вид (рис. 2.5):

2. Модель Лотки–Вольтерры

```

# Определяем модель Лотки-Вольтерры
function lotka_volterra!(du, u, p, t)
    x, y = u
    a, b, c, d = p
    du[1] = a * x - b * x * y # Уравнение для жертв
    du[2] = -c * y + d * x * y # Уравнение для хищников
end

# Начальные условия
u0 = [1.0, 1.0] # начальные популяции жертв и хищников
# Параметры модели
p = [1.5, 1.0, 3.0, 1.0] # a, b, c, d
# Интервал времени
tspan = (0.0, 10.0)
# Создаём задачу
prob = ODEProblem(lotka_volterra!, u0, tspan, p)
# Решаем задачу
sol = solve(prob, Tsit5())
# Построение графика
plot(sol, label=["Жертвы" "Хищники"], color="black",
     linestyle = [:solid :dash], title="Модель Лотки-Вольтерры",
     xlabel="Время", ylabel="Размер популяции")

```

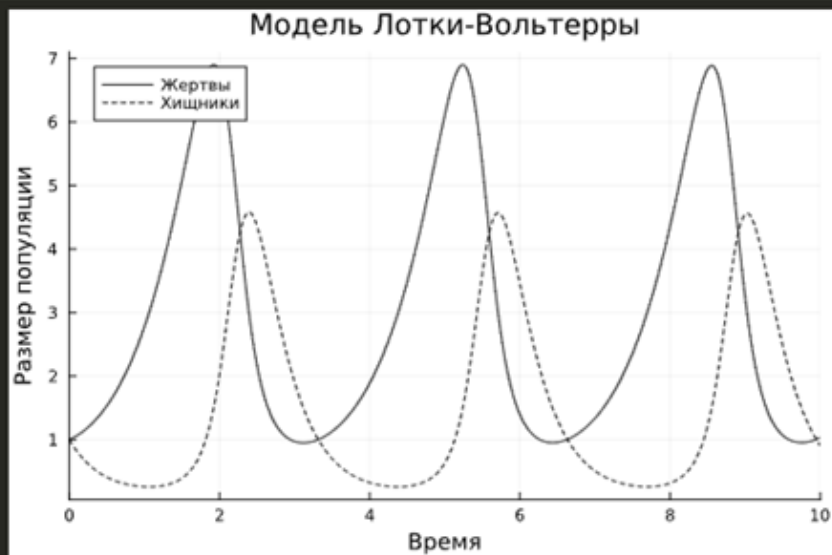


Рис. 2.5: Модель Лотки–Вольтерры: динамика изменения численности популяций

Фазовый портрет (рис. 2.6):

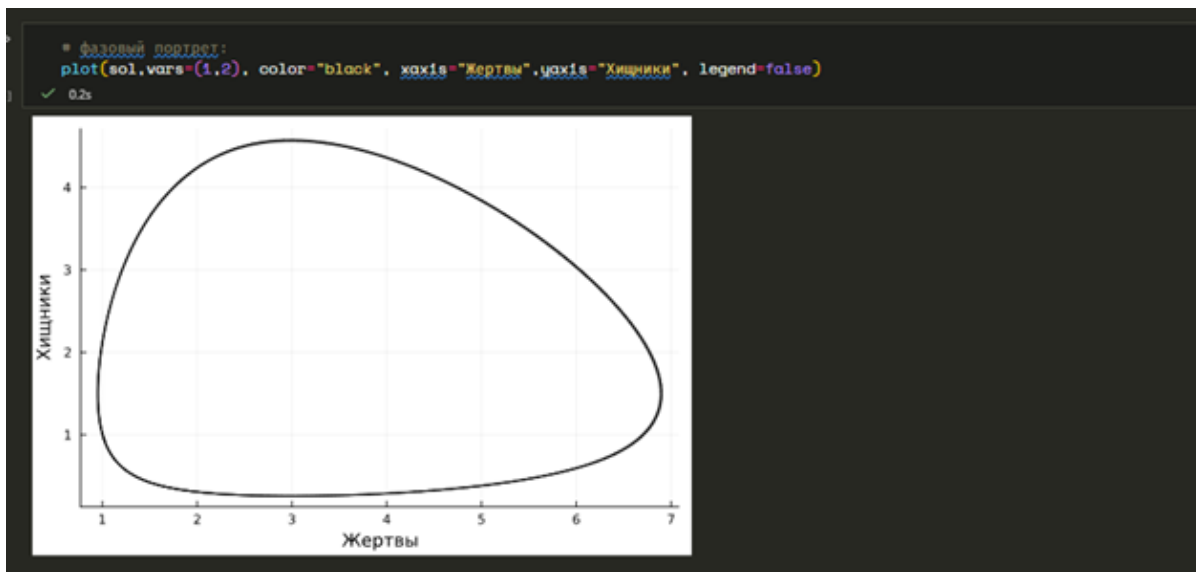


Рис. 2.6: Модель Лотки–Вольтерры: фазовый портрет

2.5 Самостоятельное выполнение

Выполнение задания №1 (рис. 2.7):

Самостоятельное выполнение

1) Реализовать и проанализировать модель роста численности изолированной популяции (модель Мальтуса). Начальные данные и параметры задать самостоятельно и пояснить их выбор. Построить соответствующие графики (в том числе с анимацией):

```
# Определённые параметры
b = 1.0
c = 0.2
a = b - c
u0 = 1.0 # Начальная численность популяции
tspan = (0.0, 10.0) # Интервал времени

# Описание модели
f(u, p, t) = a * u # Уравнение роста популяции

# Задача задачи
prob = ODEProblem(f, u0, tspan)

# Решение задачи
sol = solve(prob)

# Визуализация решения
anim = @animate for i in 1:length(sol.t)
    plot(sol.t[1:i], sol.u[1:i], linewidth=3, title="Модель Мальтуса", xlabel="Время", ylabel="Численность популяции", legend=false)
end

# Сохранение анимации в файл
gif(anim, "malthus_population_growth.gif", fps=10)
```

[Info: Saved animation to `C:\Users\Igor\OneDrive\Desktop\grafical\malthus_population_growth.gif`

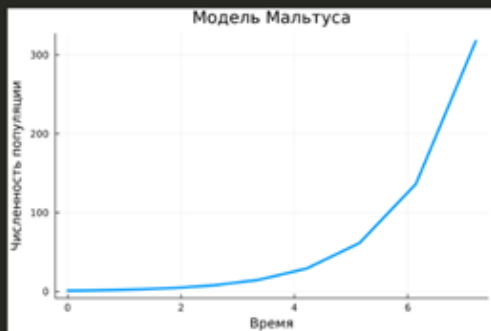


Рис. 2.7: Решение задания №1

Выполнение задания №2 (рис. 2.8):

выбор. Построить соответствующие графики (в том числе с анимацией):

```

% Определение модели
function logistic1(du, u, p, t)
    r, k = p
    du[1] = r * u[1] * (1 - u[1] / k)
end
% Начальные условия
u0 = [1.0] % Начальная численность популяции
% Параметры
r = 0.5 % Коэффициент роста
k = 10.0 % Емкость экосистемы
p = (r, k)
% Интервал времени
tspan = (0.0, 20.0)
% Решение
prob = ODEProblem(logistic1, u0, tspan, p)
sol = solve(prob, Tsit5()) % Используем Tsit5 для не-жесткой задачи
% Создание объекта анимации
anim = Animation()
% Построение анимации
for t in 0:0.5:20
    plot(sol, vars=(0, 1), linewidth=2, color=:blue, title="Логистическая модель роста",
        xlabel="Время", ylabel="Популяция", legend=false)
    scatter!([t], [sol(t)[1]], color=:red, label="", markersize=5) % Добавляем текущую точку
    frame(anim) % Добавляем кадр в анимацию
end
% Сохранение анимации
gif(anim, "logistic_growth.gif", fps=10) % Сохраняем анимацию в файл
3.4s

```

[Info: Saved animation to c:\Users\timur\OneDrive\Desktop\md\grafica\logistic_growth.gif]

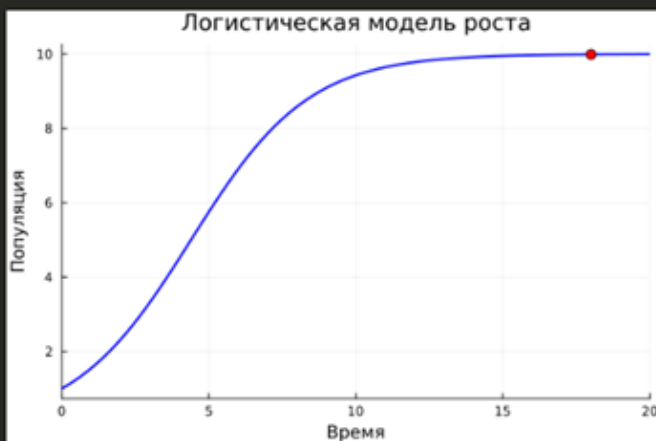


Рис. 2.8: Решение задания №2

Выполнение задания №3 (рис. 2.9):

```

# Переопределим модель
function sirf(S, I, R, p, t)
  S, I, R = p
  dS[1] = -beta * S * I # Восприимчивые
  dS[2] = -beta * S * I - mu * I # Восприимчивые
  dI[1] = beta * S * I # Больные
  dI[2] = -beta * S * I - mu * I # Больные
end

# Начальные условия
u0 = [0.99, 0.01, 0.0] # 99% восприимчивые, 1% инфицированные
# Параметры
beta = 0.5 # коэффициент передачи инфекции
mu = 0.1 # коэффициент естественной
rho = [0, 0]
# Решаем задачу
tspan = [0, 0, 100, 0]
# Рисуем
prob = ODEProblem(sirf, u0, tspan, p)
sol = solve(prob, Tsit5()) # Решаем задачу Tsit5 с помощью решателя
# Выводим график
u = zeros{Float64, 2, 2} for i in 1:length(sol.t)
  solf(sol.t[i], u[1]) for u in sol.u[i, 1], label="Восприимчивые", color=blue, linewidth=2
  solf(sol.t[i], u[2]) for u in sol.u[i, 2], label="Восприимчивые", color=red, linewidth=2
  solf(sol.t[i], u[3]) for u in sol.u[i, 3], label="Восприимчивые", color=green, linewidth=2
end
title("Модель эпидемии SIR")
xlabel("Время")
ylabel("Восприимчивые")
end

# Сохраняем график
gif("sir_model_animation.gif", fps=100)

```

✓ 100%

Info: Based animation to [@JuliaTimeSeriesPlots/DevTools/infographic/sir_model_animation.gif](#)

Время	Восприимчивые (Blue)	Восприимчивые (Red)	Восприимчивые (Green)
0	1.00	0.00	0.00
5	0.98	0.02	0.00
10	0.95	0.05	0.00
15	0.75	0.15	0.10

Выполнение задания N^o4 (рис. 2.10):

```

* Параметры модели
function setup(ds, w, p, T)
    S = 0; I = 0; R = 0;
    S = 0; I = 0; R = 0;
    ds[1] = S + w * 1 / N; # Восприимчивые
    ds[2] = I + w * 1 / N; # Инфицированные
    ds[3] = S + w * 1 / N; # Восприимчивые
    ds[4] = S + w * 1 / N; # Восприимчивые
    ds[5] = S + w * 1 / N; # Восприимчивые
end

* Параметры модели
w = [0.01, 0.0, 0.01, 0.0] # 0% восприимчивых, 0% инфицированных
* Параметры
S = 0.5 # Восприимчивые: половина населения
I = 0.5 # Инфицированные: половина населения
R = 0.5 # Восприимчивые: половина населения
w = 1.0 # Скорость заражения
p = [S, I, R, N]
* Параметры модели
tspan = [0, 100]
* Модель
prob = ODEProblem.setup(ds, w, tspan, p)
sol = solve(prob, Tsit5()) # Решение Tsit5 для модели
* Параметры модели
plot(sol, labels=["Восприимчивые", "Инфицированные", "Восстановленные", "Выведенные"],
      title="Модель эпидемии SEIR", xlabel="Время", ylabel="Плотность", xlims=[0, 100])

```

15

Выполнение задания №5 (рис. 2.11):

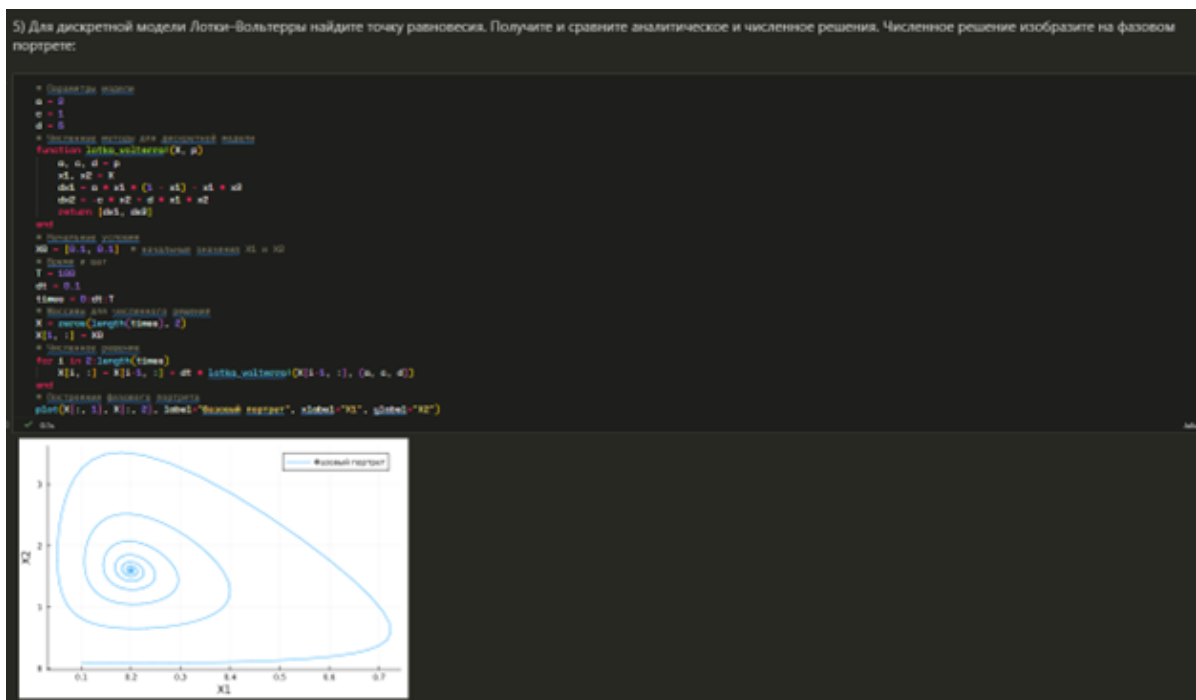


Рис. 2.11: Решение задания №5

Выполнение задания №6 (рис. 2.12):

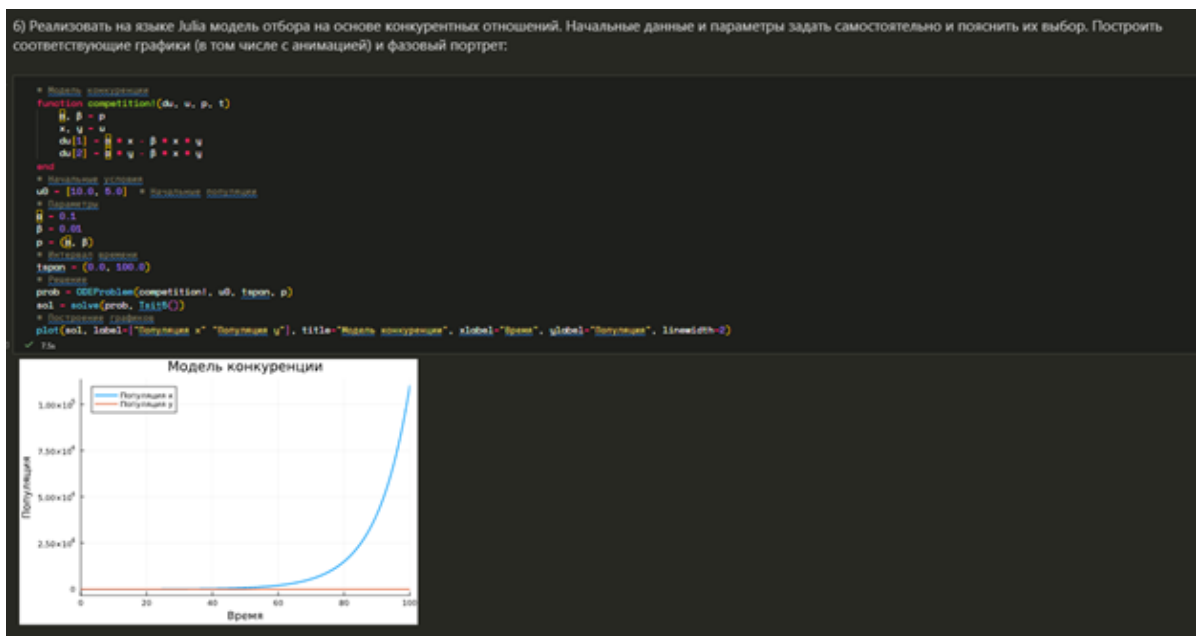


Рис. 2.12: Решение задания №6

Выполнение задания №7 (рис. 2.13):

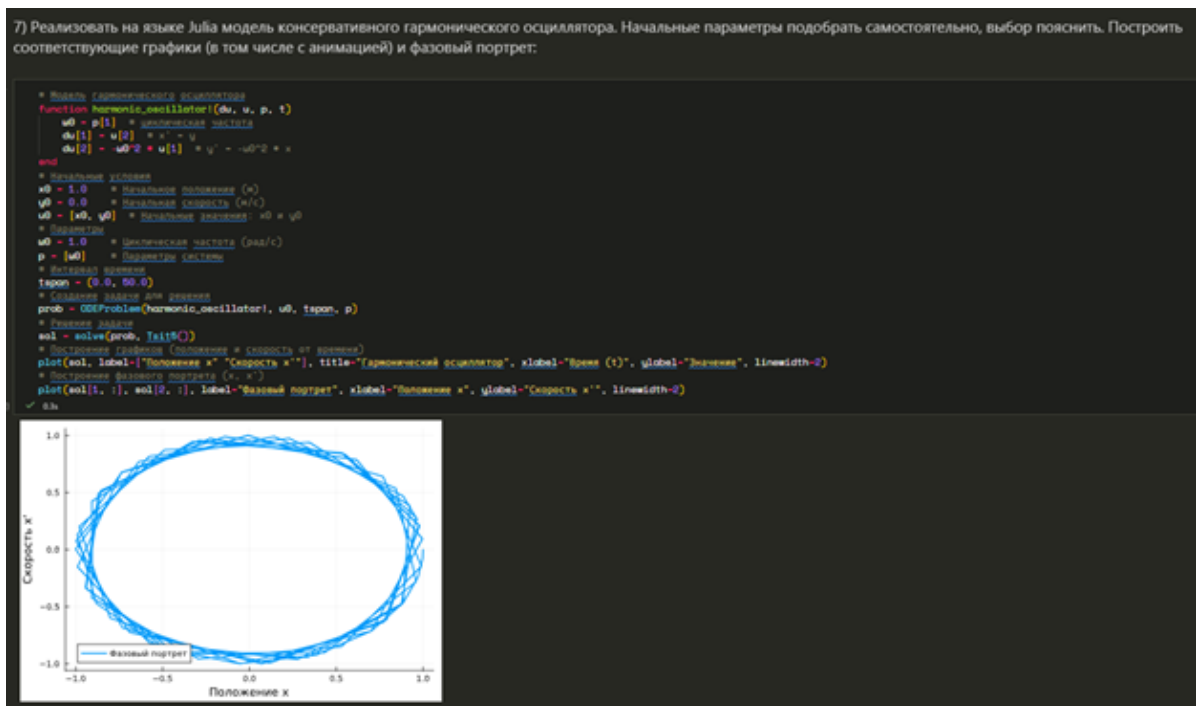


Рис. 2.13: Решение задания №7

Выполнение задания №8 (рис. 2.14):

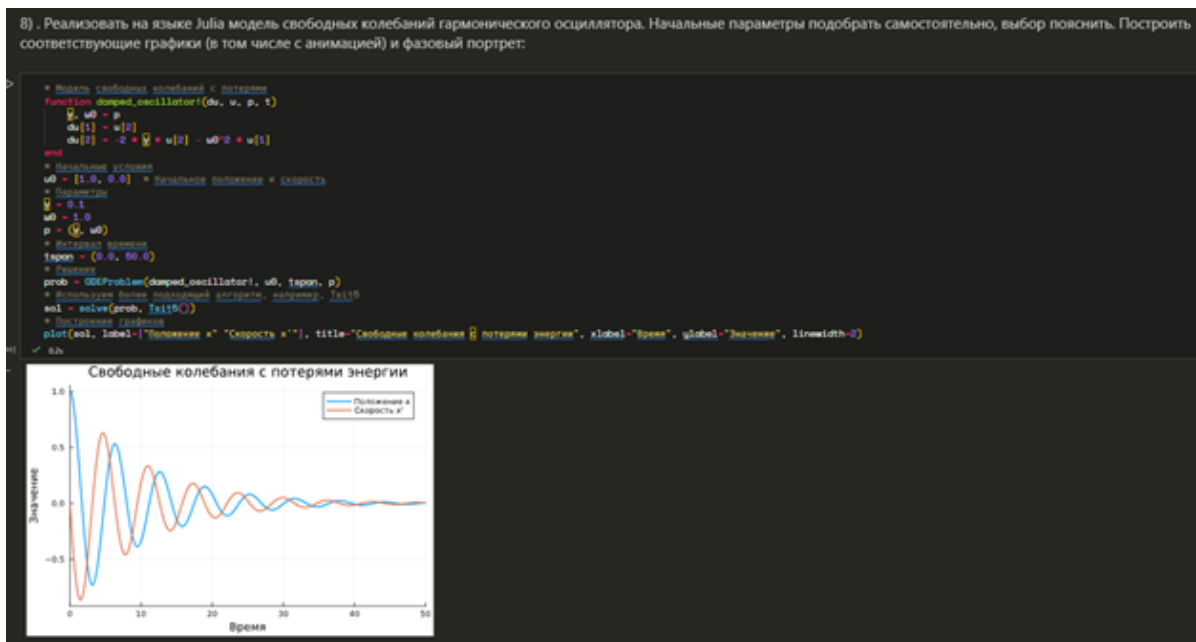


Рис. 2.14: Решение задания №8

3 Вывод

В ходе выполнения лабораторной работы были освоены специализированные пакеты для решения задач в непрерывном и дискретном времени.

4 Список литературы. Библиография

[1] Julia Documentation: <https://docs.julialang.org/en/v1/>