Лабораторная работа №6

Леснухин Даниил Дмитриевич Российский университет дружбы народов Москва

Цель работы

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

Задание

1. Используя JupyterLab, повторите примерыи. При этом дополните графики обозначениями осей координат, легендой с названиями траекторий, названиями графиков и т.п.
2. Выполните задания для самостоятельной работы.

Теоретическое введение

Julia – высокоуровневый свободный язык программирования с динамической типизацией, созданный для математических вычислений [@julialang]. Эффективен также и для написания программ общего назначения. Синтаксис языка схож с синтаксисом других математических языков, однако имеет некоторые существенные отличия.

Для выполнения заданий была использована официальная документация Julia [@juliadoc].

Выполнение лабораторной работы

Задания для самостоятельной работы

1. Реализуем и проанализируем модель роста численности изолированной популяции(модель Мальтуса):

$$\dot{x} = ax, \quad a = b - c,$$

где $x(t)$ – численность изолированной популяции в момент времени t,a – коэффициент роста популяции, b – коэффициент рождаемости, c – коэффициент смертности. Построим соответствующие графики (в том числе с анимацией)

```julia
using DifferentialEquations, Plots

function maltus_model(du, u, p, t)
    a = p
    du[1] = a * u[1]
end

a = 0.2
u0 = [10.0]
tspan = (0.0, 20.0)

prob = ODEProblem(maltus_model, u0, tspan, a)
sol = solve(prob)


plot(sol, lw=2, title="Модель Мальтуса: экспоненциальный рост", xaxis="Время", yaxis="Размер популяции", label="численность", legend=:topleft)

anim = @animate for i in 1:length(sol.t)
    plot(sol.t[1:i], sol[1, 1:i], lw=2, title="Модель Мальтуса: экспоненциальный рост",
     xaxis="Время", yaxis="Размер популяции", label="численность", legend=:topleft,
     xlim=(0, 20), ylim=(0, maximum(sol[1, :]) * 1.1))
end

gif(anim, "malthus_model.gif", fps=10)
```

Рис. 1: Задание2

2. Реализуем и проанализируем логистическую модель роста популяции:

$$\dot{x} = rx(1 - \frac{x}{k}), \quad r > 0, \quad k > 0,$$

где r – коэффициент роста популяции, k – потенциальная ёмкость экологической системы (предельное значение численности популяции). Построим соответствующие графики (в том числе с анимацией)

3. Реализуем и проанализируем логистическую модель эпидемии Кермака–Маккендрика (SIR-модель):

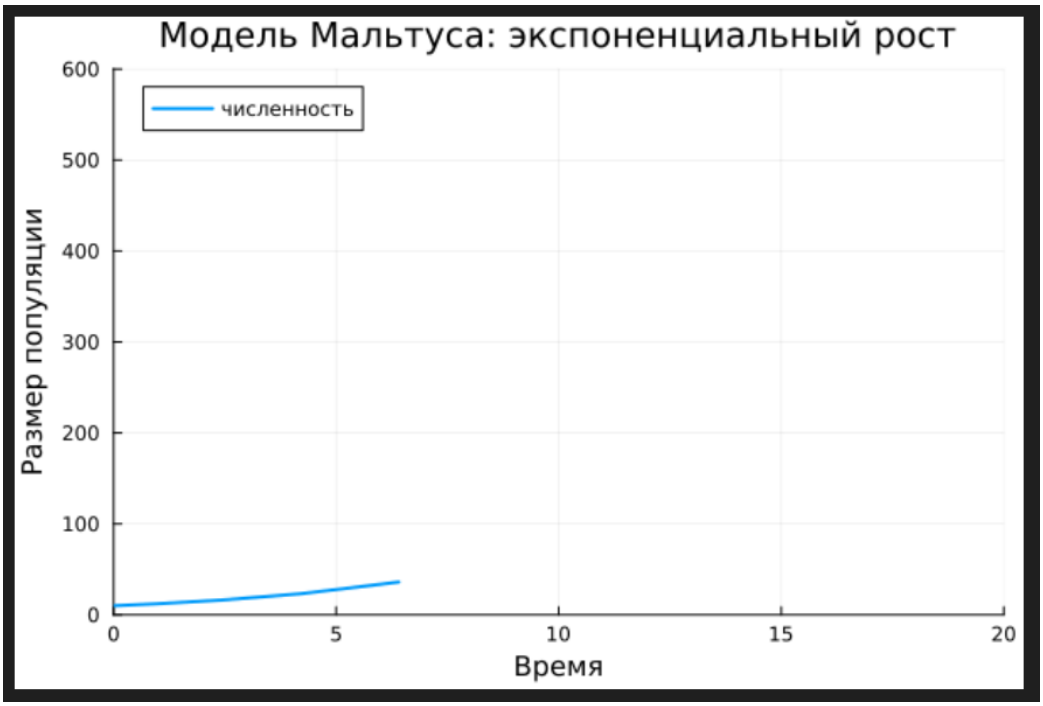


Рис. 2: Задание1

```julia
function logistic_model(du, u, p, t)
    r, k = p
    du[1] = r * u[1] * (1 - u[1] / k)
end

r = 0.5
k = 100.0
u0 = [10.0]

tspan = (0.0, 30.0)

p = (r, k)

prob = ODEProblem(logistic_model, u0, tspan, p)
sol = solve(prob)

plot(sol, lw=2, title="Модель Вергула: логистический рост", xaxis="Время", yaxis="Размер популяции", label="численность", legend=:topleft)

hline!([k], lw=2, ls=:dash, color=:red, label="Ёмкость среды")

anim = @animate for i in 1:length(sol.t)
    plot(sol.t[1:i], sol[1, 1:i], lw=2, title="Модель Вергула: логистический рост",
     xaxis="Время", yaxis="Размер популяции", label="численность", legend=:topleft,
     xlim=(0, 30), ylim=(0, k * 1.1))
    hline!([k], lw=2, ls=:dash, color=:red, label="Ёмкость среды")
end

gif(anim, "logistic_model.gif", fps=10)
```

Рис. 3: Задание2

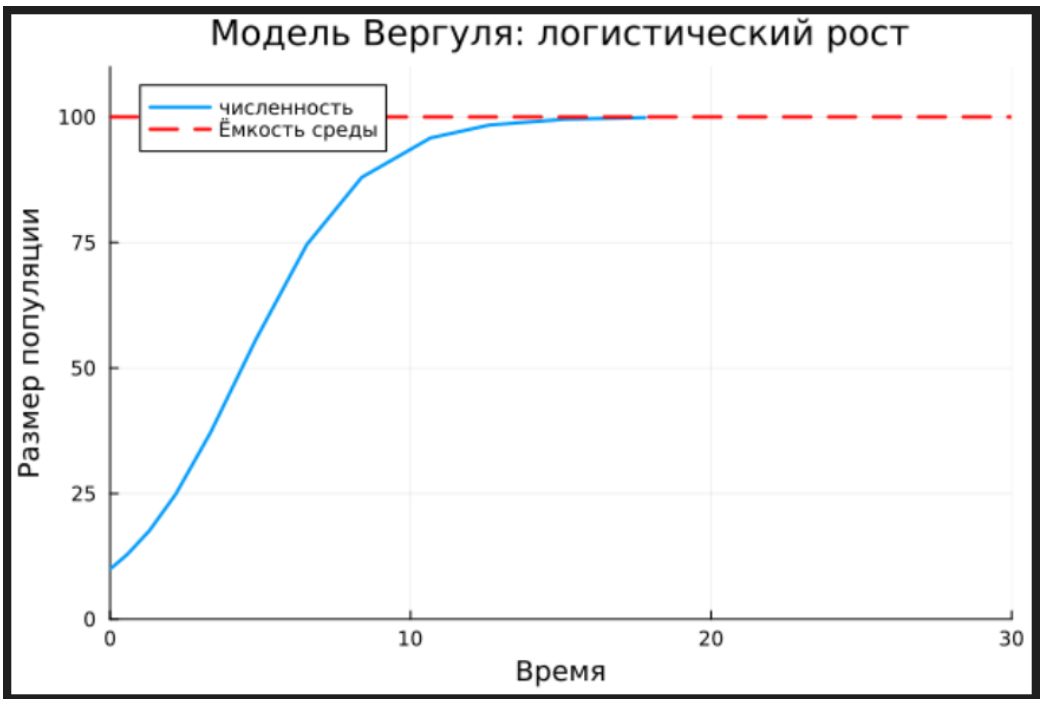


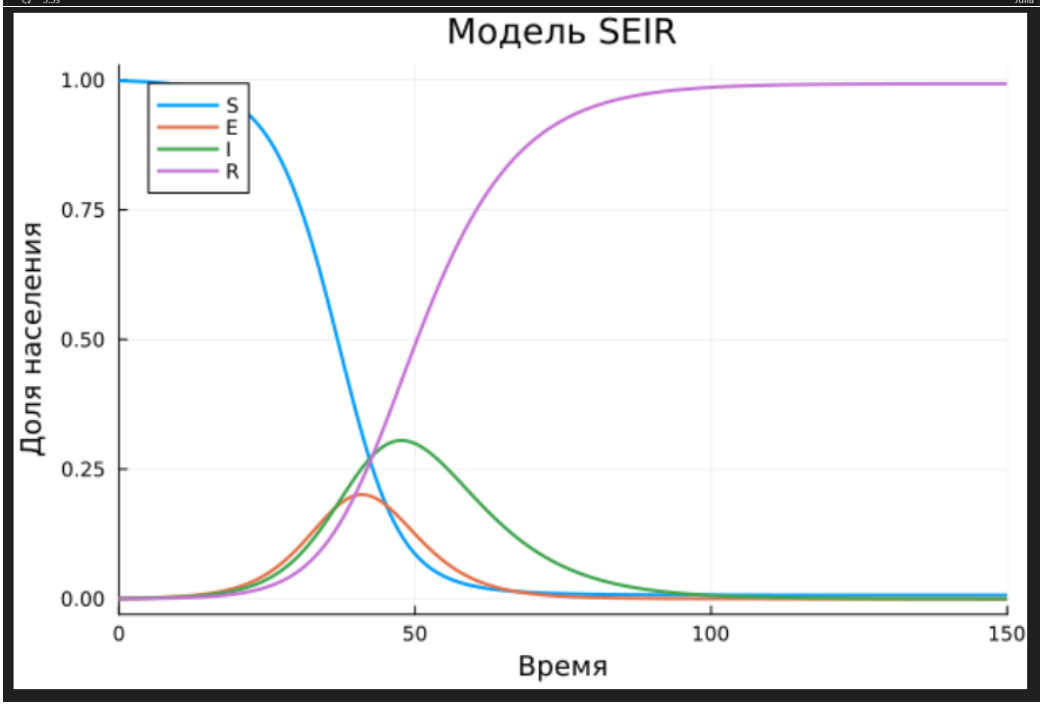
Рис. 4: Задание2

$$\begin{cases} \dot{S} = -\beta IS, \\ \dot{I} = \beta IS - \gamma I, \\ \dot{R} = \gamma I, \end{cases}$$

где S – численность восприимчивой популяции, I – численность инфицированных, R – численность удаленной популяции (в результате смерти или выздоровления), и N — это сумма этих трёх, а β и γ - это коэффициенты заболеваемости и выздоровления соответственно

```julia
function seir_model!(du, u, p, t)
    S, E, I, R = u
    β, σ, γ, N = p
    du[1] = -β * S * I / N
    du[2] = β * S * I / N - σ * E
    du[3] = σ * E - γ * I
    du[4] = γ * I
end

β = 0.5
σ = 0.2
γ = 0.1
N = 1000.0
u0 = [999.0/N, 0.0, 1.0/N , 0.0]  # S, E, I, R
tspan = (0.0, 150.0)

p = (β, σ, γ, N)
prob = ODEProblem(seir_model!, u0, tspan, p)
sol = solve(prob)

plot(sol, lw=2, title="Модель SEIR распространения инфекции", xaxis="Время", yaxis="Доля населения", label=["Восприимчивые (S)" "Экспонированные (E)" "Инф
```

3.5s Julia



5. Для дискретной модели Лотки–Вольтерры:

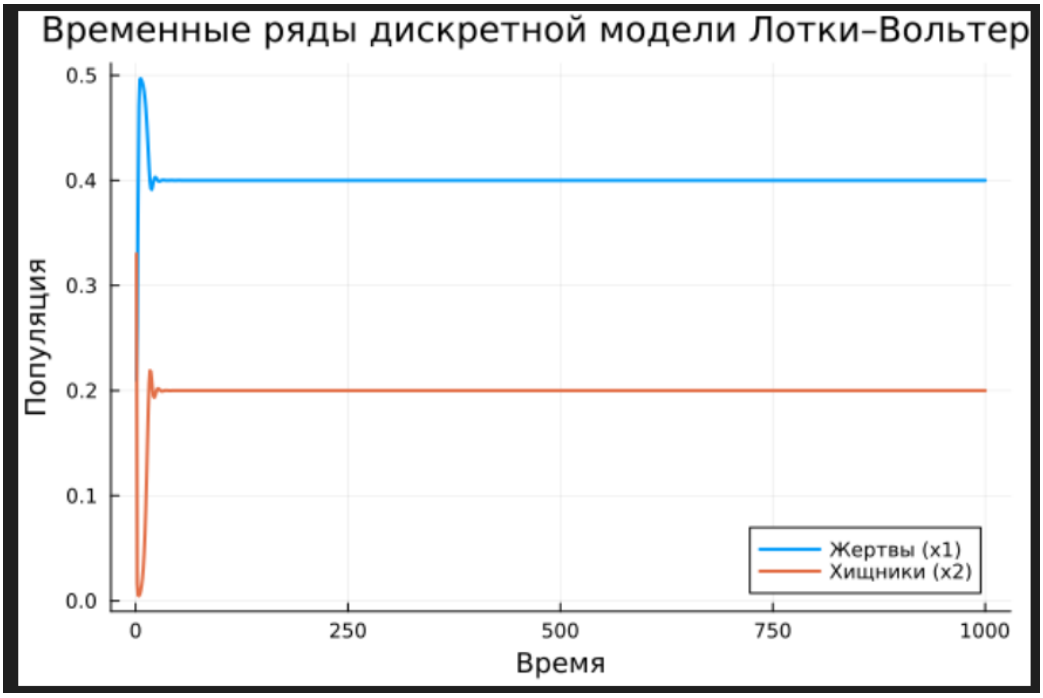
$$\begin{cases} X_1(t+1) = aX_1(t)(1 - X_1(t)) - X_1(t)X_2(t), \\ X_2(t+1) = -cX_2(t) - dX_1(t)X_2(t). \end{cases}$$

с начальными данными $a = 2, c = 1, d = 5$ найдем точку равновесия. Получим и сравним аналитическое и численное решения

$$\begin{cases} \dot{S} = -\dfrac{\beta}{N} IS, \\ \dot{E} = \dfrac{\beta}{N} IS - \delta E, \\ \dot{I} = \delta E - \gamma I, \\ \dot{R} = \gamma I, \end{cases}$$

```julia
using Plots

function discrete_lotka_volterra(x1, x2, a, c, d)
    x1_next = a * x1 * (1 - x1) - x1 * x2
    x2_next = -c * x2 + d * x1 * x2
    return x1_next, x2_next
end

a = 2
c = 1
d = 5

# равновесная точка
x1_eq = c / d
x2_eq = a * x1_eq * (1 - x1_eq)

println("Равновесная точка: (x1, x2) = ($x1_eq, $x2_eq)")

T = 1000

# создаём массивы
x1_vals = zeros(T)
x2_vals = zeros(T)

# начальные условия (небольшое отклонение)
x1_vals[1] = x1_eq + 0.01
x2_vals[1] = x2_eq + 0.01
```

```
# итерации
for t in 1:T-1
    x1_vals[t+1], x2_vals[t+1] =
        discrete_lotka_volterra(x1_vals[t], x2_vals[t], a, c, d)
end

# фазовый портрет
scatter(x1_vals, x2_vals,
        markersize=1,
        alpha=0.5,
        title="Фазовый портрет дискретной модели Лотки-Вольтерры",
        xaxis="Жертвы (x1)",
        yaxis="Хищники (x2)",
        legend=false)

scatter!([x1_eq], [x2_eq], markersize=8, color=:red,
        label="Равновесная точка")
# временные ряды
plot(1:T, x1_vals,
     lw=2,
     title="Временные ряды дискретной модели Лотки-Вольтерры",
     xaxis="Время",
     yaxis="Популяция",
     label="Жертвы (x1)")
plot!(1:T, x2_vals, lw=2, label="Хищники (x2)")
```



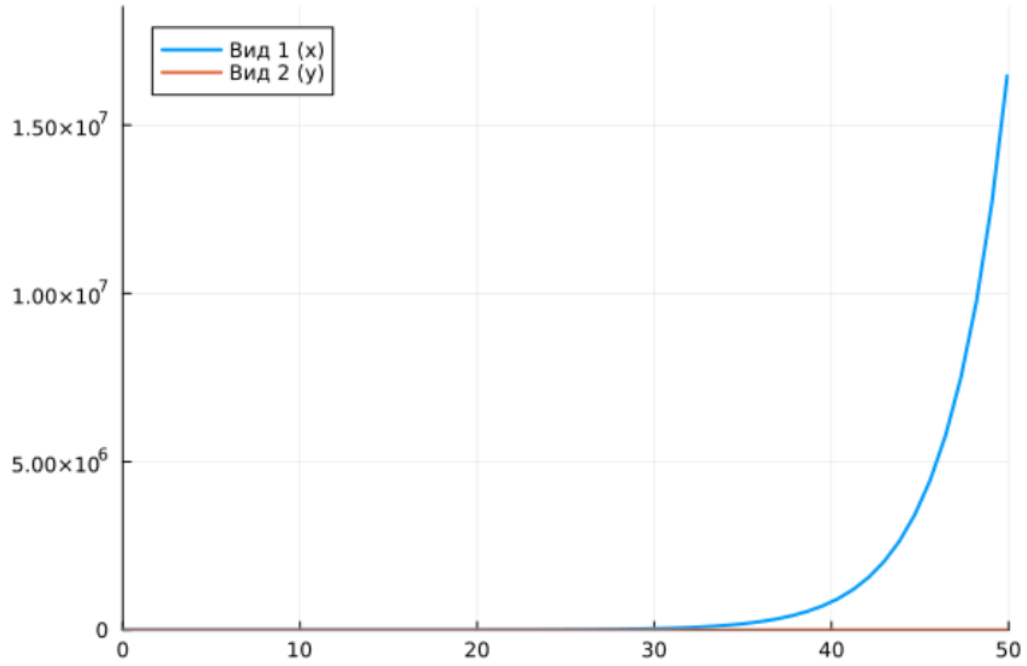
6. Реализуем на языке Julia модель отбора на основе конкурентных отношений:

$$\begin{cases} \dot{x} = \alpha x - \beta xy, \\ \dot{y} = \alpha y - \beta xy, \end{cases}$$

Построим соответствующие графики (в том числе с анимацией) и фазовый портрет

```julia
using DifferentialEquations, Plots

function competition_model(du, u, p, t)
    x, y = u
    a, b = p
    du[1] = a * x - b * x * y
    du[2] = a * y - b * x * y
end

a = 0.3
b = 0.1
p = (a, b)

u0 = [10.0, 5.0]
tspan = (0.0, 50.0)

prob = ODEProblem(competition_model, u0, tspan, p)
sol = solve(prob)

# График во времени
plot(sol, lw=2,
     title="Модель конкуренции",
     xaxis="Время",
     yaxis="Размер популяции",
     label=["Вид 1 (x)" "Вид 2 (y)"],
     legend=:topleft)

# Фазовый портрет
```

✓ 3.4s

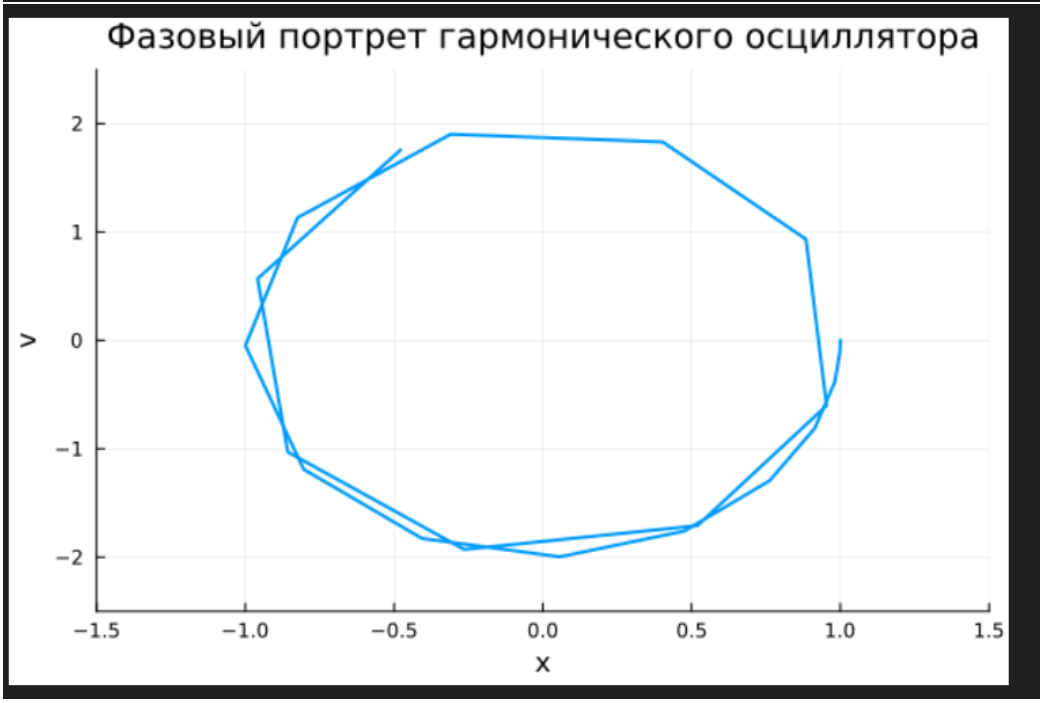


7. Реализуем на языке Julia модель консервативного гармонического осциллятора:

$$\ddot{x} + \omega_0^2 = 0, x(t_0) = x_0, \dot{x}(t_0) = y_0.$$

Построим соответствующие графики (в том числе с анимацией) и фазовый портрет

```julia
function harmonic_oscillator(du, u, p, t)
    x, v = u
    w0 = p

    du[1] = v
    du[2] = -w0^2 * x
end

w0 = 2.0

u0 = [1.0, 0.0]
tspan = (0.0, 10.0)


prob = ODEProblem(harmonic_oscillator, u0, tspan, w0)
sol = solve(prob)

plot(sol, lw=2, title="Гармонический осциллятор", xaxis="Время", yaxis="x(t)", label="Численное решени

plot(sol, vars=(1,2), lw=2, title="Фазовый портрет гармонического осциллятора", xaxis="x", yaxis="v",

anim = @animate for i in 1:length(sol.t)
    plot(sol[1,1:i], sol[2,1:i], lw=2,
        title="Фазовый портрет гармонического осциллятора",
        xaxis="x",
        yaxis="v",
        legend=false,
```



8. Реализуем на языке Julia модель свободных колебаний гармонического

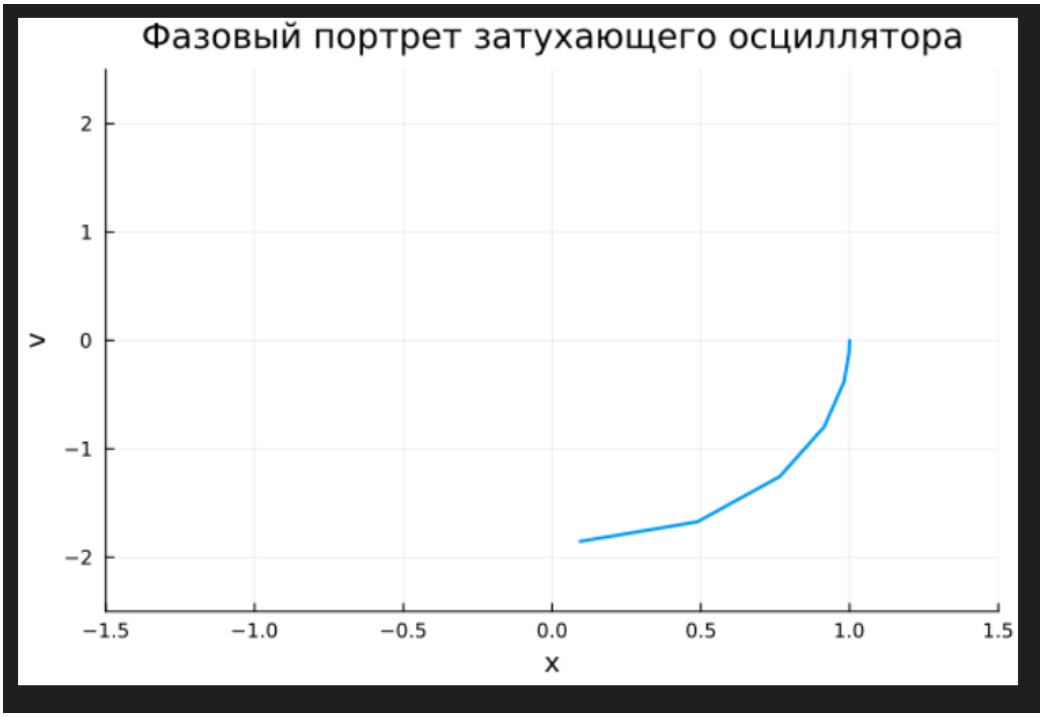
осциллятора:

$$\ddot{x} + 2\gamma\dot{x} + \omega_0^2 = 0, x(t_0) = x_0, \dot{x}(t_0) = y_0.$$

Построим соответствующие графики (в том числе с анимацией) и фазовый портрет

```
function damped_oscillator(du, u, p, t)
    x, v = u
    w0, γ = p

    du[1] = v
    du[2] = -2*γ*v - w0^2 * x
end

w0 = 2.0
γ = 0.1

u0 = [1.0, 0.0]
tspan = (0.0, 20.0)
p = (w0, γ)
prob = ODEProblem(damped_oscillator, u0, tspan, p)

sol = solve(prob)

plot(sol, lw=2, title="Затухающие колебания", xaxis="Время", yaxis="x(t)", label="Численное решение")
plot(sol, vars=(1,2), lw=2, title="Фазовый портрет затухающего осциллятора", xaxis="x", yaxis="v", legend=false)

anim = @animate for i in 1:length(sol.t)
    plot(sol[1,1:i], sol[2,1:i], lw=2,
        title="Фазовый портрет затухающего осциллятора",
        xaxis="x",
        yaxis="v",
        legend=false,
        xlim=(-1.5, 1.5),
        ylim=(-2.5, 2.5))
```

✓ 2.1s



Выводы

В результате выполнения данной лабораторной работы я освоил специализированные пакеты для решения задач в непрерывном и дискретном времени.