

# **Лабораторная Работа №6**

Решение моделей в непрерывном и дискретном времени

---

Козлов В.П.

Российский университет дружбы народов им. Патриса Лумумбы, Москва, Россия

## Докладчик

---

- Козлов Всеволод Павлович
- НФИбд-02-22
- Российский университет дружбы народов
- [1132226428@pfur.ru]

## **Выполнение лабораторной работы**

---

## Цель работы

---

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

## Задание

---

1. Используя Jupyter Lab, повторите примеры из раздела 6.2.
2. Выполните задания для самостоятельной работы (раздел 6.4).

# Библиотеки

```
# Подключение необходимых пакетов
import Pkg
Pkg.add("DifferentialEquations")
Pkg.add("Plots")
Pkg.add("ParameterizedFunctions")
|
using DifferentialEquations, Plots, ParameterizedFunctions

Updating registry at `C:\Users\vsvld\.julia\registries\General.toml`
Resolving package versions...
Installed SciMLPublic ━━━━━━━━ v1.0.0
Installed Accessors ━━━━━━━━ v0.1.42
Installed OrdinaryDiffEqRosenbrock ━━━━━━ v1.18.1
Installed LoggingExtras ━━━━━━ v1.2.0
Installed OrdinaryDiffEqRKN ━━━━ v1.5.0
```

Figure 1: Библиотеки

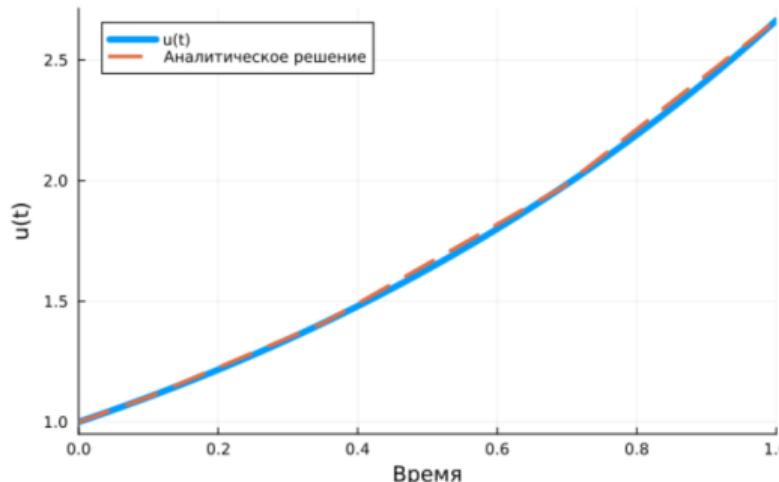
# Экспоненциальный рост

```
# Пример 1: Модель экспоненциального роста
a = 0.98
f(u,p,t) = a*u
u0 = 1.0
tspan = (0.0,1.0)

prob = ODEProblem(f,u0,tspan)
sol = solve(prob)

plot(sol, linewidth=5,title="Модель экспоненциального роста",
      xaxis="Время",yaxis="u(t)",label="u(t)")
plot!(sol.t, t->1.0*exp(a*t),lw=3,ls=:dash,label="Аналитическое решение")
```

2]: Модель экспоненциального роста



# Повышенная точность

```
[3]: # Решение с повышенной точностью
sol = solve(prob,abstol=1e-8,reltol=1e-8)
plot(sol, lw=2, color="black", title="Модель экспоненциального роста",
      xaxis="Время",yaxis="u(t)",label="Численное решение")
plot!(sol.t, t->1.0*exp(a*t),lw=3,ls=:dash,color="red",label="Аналитическое решение")
```

[3]:

Модель экспоненциального роста

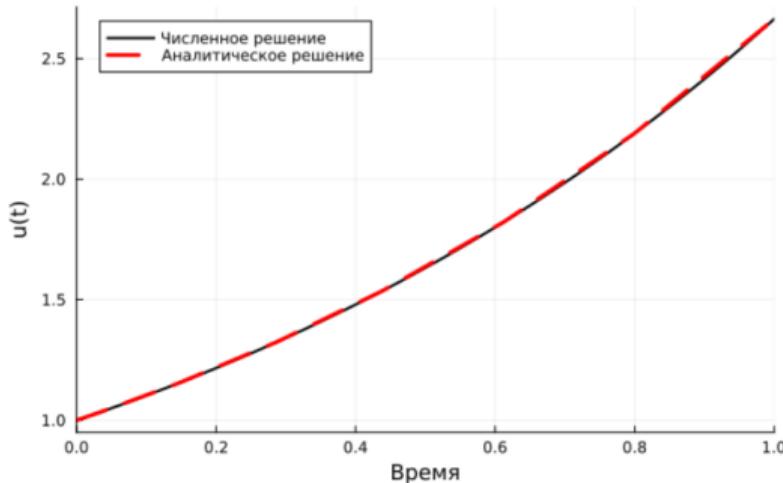


Figure 3: Повышенная точность

## Система Лоренца

---

![Система Лоренца(image/4.png){ #fig:004 width=70% }

# Модель Лотки-Вольтерры

```
: # Пример 3: Модель Лотки-Вольтерры
lvl = @ode_def LotkaVolterra begin
    dx = a*x - b*x*y
    dy = -c*y + d*x*y
end a b c d

u0 = [1.0,1.0]
p = (1.5,1.0,3.0,1.0)
tspan = (0.0,10.0)

prob = ODEProblem(lvl,u0,tspan,p)
sol = solve(prob)
plot(sol, label = ["Жертвы" "Хищники"], color="black", ls=:solid :dash,
      title="Модель Лотки - Вольтерры", xaxis="Время", yaxis="Размер популяции")

# Фазовый портрет
plot(sol,vars=(1,2), color="black", xaxis="Жертвы", yaxis="Хищники", legend=false)
```

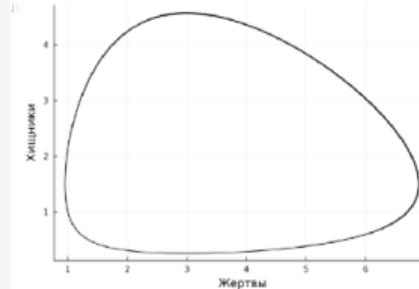


Figure 4: Модель Лотки-Вольтерры

## Задание №2

```
[11]: # ===== ЗАДАНИЕ 2: Модель Мальтуса (экспоненциальный рост) =====
# Реализовать модель роста численности изолированной популяции
function malthus_model(du, u, p, t)
    a = p
    du[1] = a * u[1]
end

u0 = [100.0] # начальная численность популяции
a = 0.1      # коэффициент роста (рождаемость - смертность)
tspan = (0.0, 50.0)
prob = ODEProblem(malthus_model, u0, tspan, a)
sol = solve(prob)

plot(sol, label="Модель Мальтуса", xaxis="Время", yaxis="Численность популяции",
     title="Экспоненциальный рост популяции", size=(500, 300))
```

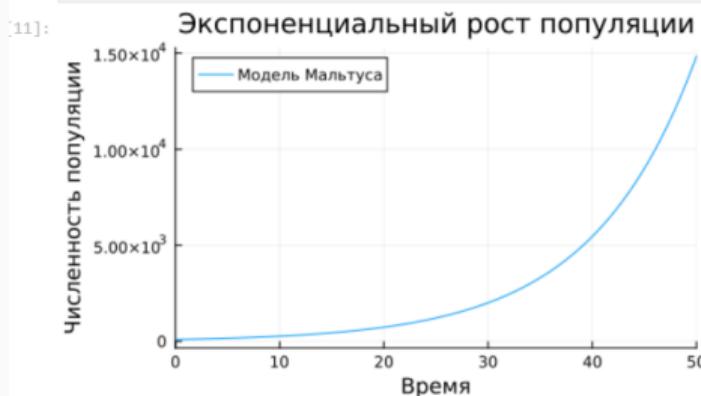


Figure 5: Задание №2

## Задание №3

```
# ===== ЗАДАНИЕ 3: Логистическая модель роста популяции =====
# Реализовать логистическую модель роста популяции
function logistic_model(du, u, p, t)
    r, k = p
    du[1] = r * u[1] * (1 - u[1]/k)
end

u0 = [10.0]      # начальная численность популяции
r = 0.2          # коэффициент роста
k = 1000.0        # емкость среды
tspan = (0.0, 100.0)
prob = ODEProblem(logistic_model, u0, tspan, (r, k))
sol = solve(prob)

plot(sol, label="Логистический рост", xaxis="Время", yaxis="Численность популяции",
     title="Логистическая модель роста популяции", size=(500, 300))
```

Логистическая модель роста популяции

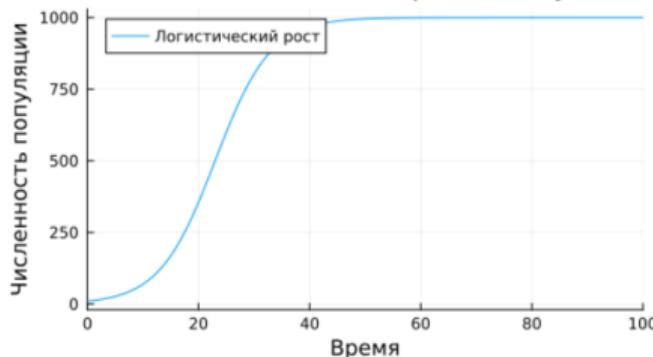


Figure 6: Задание №3

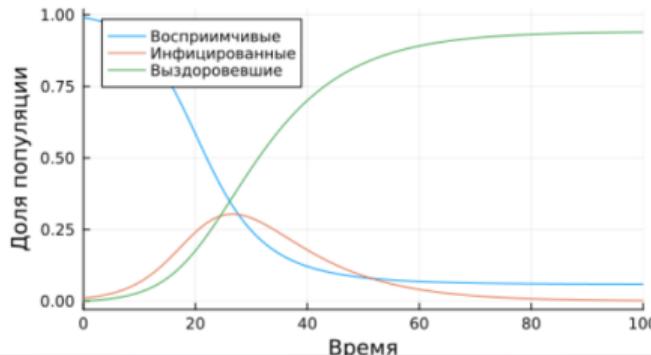
# Задание №4

```
# ===== ЗАДАНИЕ 4: Модель эпидемии Кермака-Маккендрока (SIR) =====
# Реализовать SIR-модель эпидемии
function sir_model(du, u, p, t)
    s, i, r = u
    β, ν = p
    du[1] = -β * i * s      # ds/dt
    du[2] = β * i * s - ν * i # di/dt
    du[3] = ν * i            # dr/dt
end
u₀ = [0.99, 0.01, 0.0] # s, i, r
β = 0.3                # коэффициент заражения
ν = 0.1                # коэффициент выздоровления
tspan = (0.0, 100.0)
prob = ODEProblem(sir_model, u₀, tspan, (β, ν))
sol = solve(prob)

plot(sol, label=["Восприимчивые" "Инфицированные" "Выздоровевшие"],
      title="SIR-модель эпидемии", xlabel="Время", ylabel="Доля популяции", size=(500, 300))
```

4]:

SIR-модель эпидемии



## Задание №5

```
[]: # ===== ЗАДАНИЕ 5: Модель SEIR =====
# Реализовать и исследовать SEIR-модель эпидемии
function seir_model(du, u, p, t)
    s, e, i, r = u
    β, δ, γ, N = p

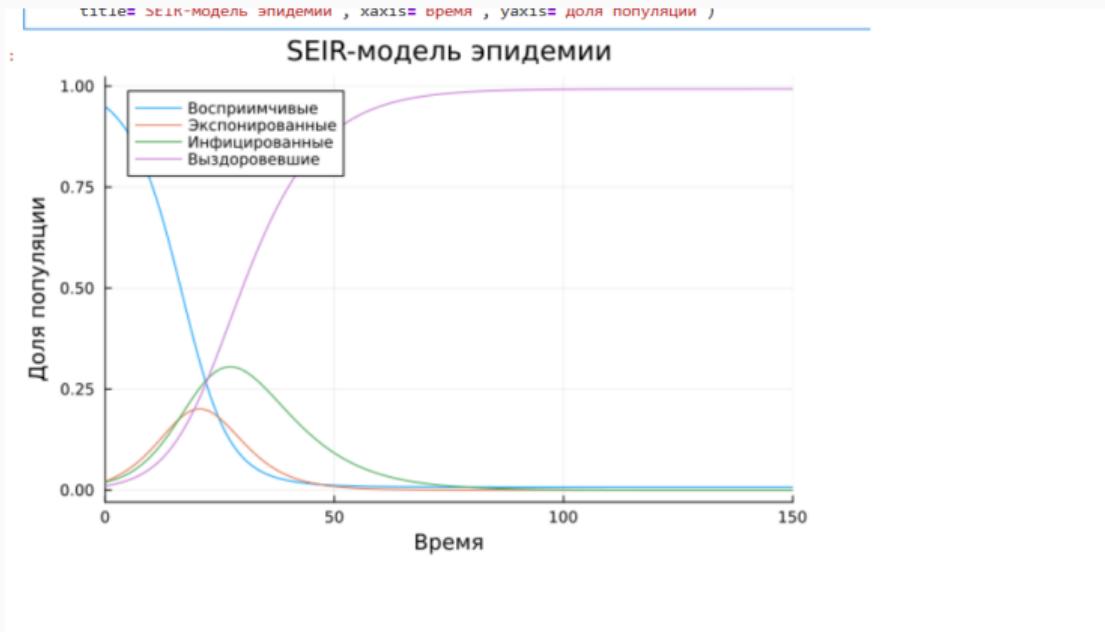
    du[1] = -β/N * s * i          # ds/dt
    du[2] = β/N * s * i - δ * e # de/dt
    du[3] = δ * e - γ * i        # di/dt
    du[4] = γ * i                 # dr/dt
end

N = 1.0 # общая численность популяции (нормированная)
u0 = [0.95, 0.02, 0.02, 0.01] # s, e, i, r
β = 0.5 # коэффициент заражения
δ = 0.2 # коэффициент перехода из экспонированных в инфицированные
γ = 0.1 # коэффициент выздоровления
tspan = (0.0, 150.0)
prob = ODEProblem(seir_model, u0, tspan, (β, δ, γ, N))
sol = solve(prob)

plot(sol, label=["Восприимчивые" "Экспонированные" "Инфицированные" "Выздоровевшие"],
      title="SEIR-модель эпидемии", xaxis="Время", yaxis="Доля популяции")
```

Figure 8: Задание №5

## Задание №5



**Figure 9:** Задание №5

# Задание №6

```
# ===== ЗАДАНИЕ 6: Дискретная модель Лотки-Вольтерры =====
# Фазовый портрет дискретной модели Лотки-Вольтерры

using LinearAlgebra

# Параметры модели:
a = 2.0    # коэффициент роста жертв
c = 1.0    # коэффициент смертности хищников
d = 5.0    # коэффициент превращения жертв в хищников

# Несколько начальных условий для построения фазового портрета
initial_conditions = [
    [0.1, 0.1],  # мало жертв и хищников
    [0.3, 0.2],  # умеренное количество
    [0.6, 0.1],  # много жертв, мало хищников
    [0.2, 0.5],  # мало жертв, много хищников
    [0.5, 0.4]   # сбалансированно
]

# Функция дискретной модели Лотки-Вольтерры
function discrete_lotka_volterra(X, p, t)
    a, c, d = p
    x1, x2 = X

    X1_next = a * x1 * (1 - x1) - x1 * x2  # изменение популяции жертв
    X2_next = -c * x2 + d * x1 * x2          # изменение популяции хищников

    return [X1_next, X2_next]
end

# Создаем фазовый портрет
plt = plot(size=(800, 600), title="Фазовый портрет дискретной модели Лотки-Вольтерры",
           xlabel="Жертвы (x1)", ylabel="Хищники (x2)",
           xticks=0:0.1:1.0, yticks=0:0.1:1.0)
```

Figure 10: Задание №6

# График

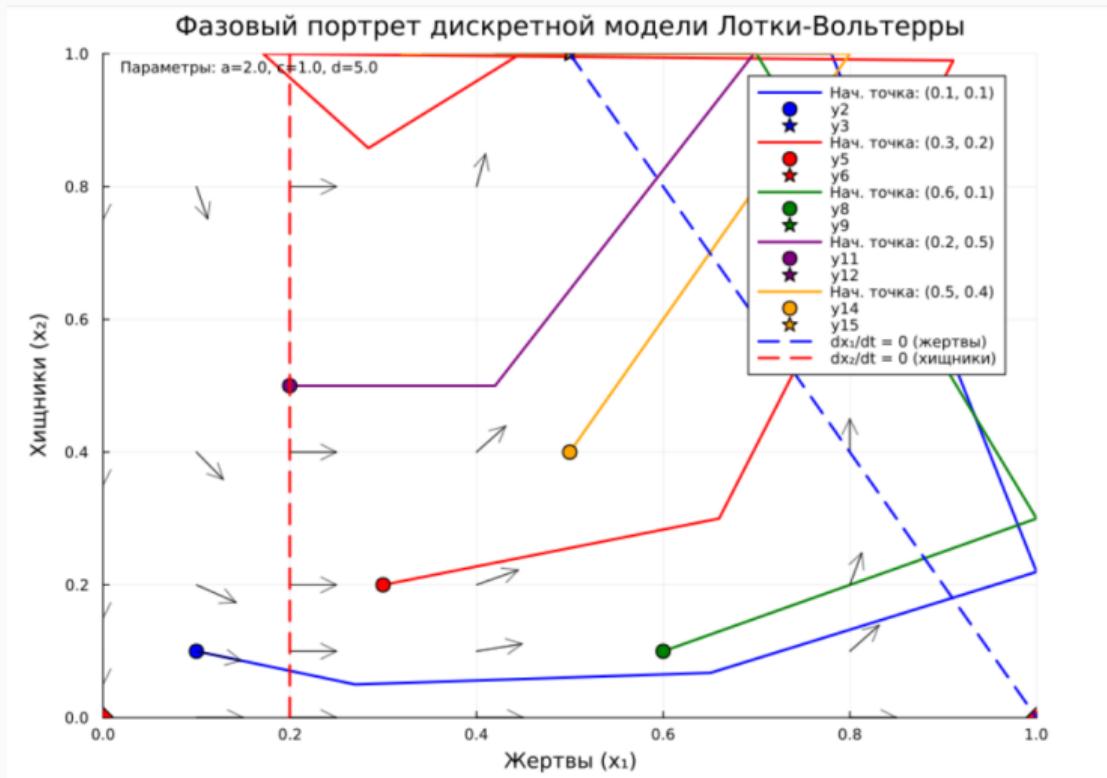


Figure 11: График

# Задание №7

```
: # ===== ЗАДАНИЕ 7: Модель отбора на основе конкурентных отношений =====
# Реализовать модель конкурентных отношений с разными параметрами для видов

using DifferentialEquations, Plots, LinearAlgebra

# Модель конкурентных отношений с разными коэффициентами
function competition_model(du, u, p, t)
    x, y = u
    a1, a2, β = p # a1 - рост вида 1, a2 - рост вида 2, β - конкуренция

    du[1] = a1 * x - β * x * y # изменение вида 1
    du[2] = a2 * y - β * x * y # изменение вида 2
end

# Параметры модели:
# a1 = 0.15 - более высокий коэффициент роста вида 1
# a2 = 0.08 - более низкий коэффициент роста вида 2
# β = 0.005 - умеренная конкуренция
a1 = 0.15
a2 = 0.08
β = 0.005

u0 = [50.0, 30.0] # начальные численности: вид 1 = 50, вид 2 = 30
tspan = (0.0, 50.0)
prob = ODEProblem(competition_model, u0, tspan, (a1, a2, β))
sol = solve(prob)

println("Параметры модели:")
println("a1 = $a1 (коэффициент роста вида 1)")
println("a2 = $a2 (коэффициент роста вида 2)")
println("β = $β (коэффициент конкуренции)")
println("Начальные условия: вид 1 = $(u0[1]), вид 2 = $(u0[2]))"

# График динамики популяций во времени
```

Figure 12: Задание №7

# График

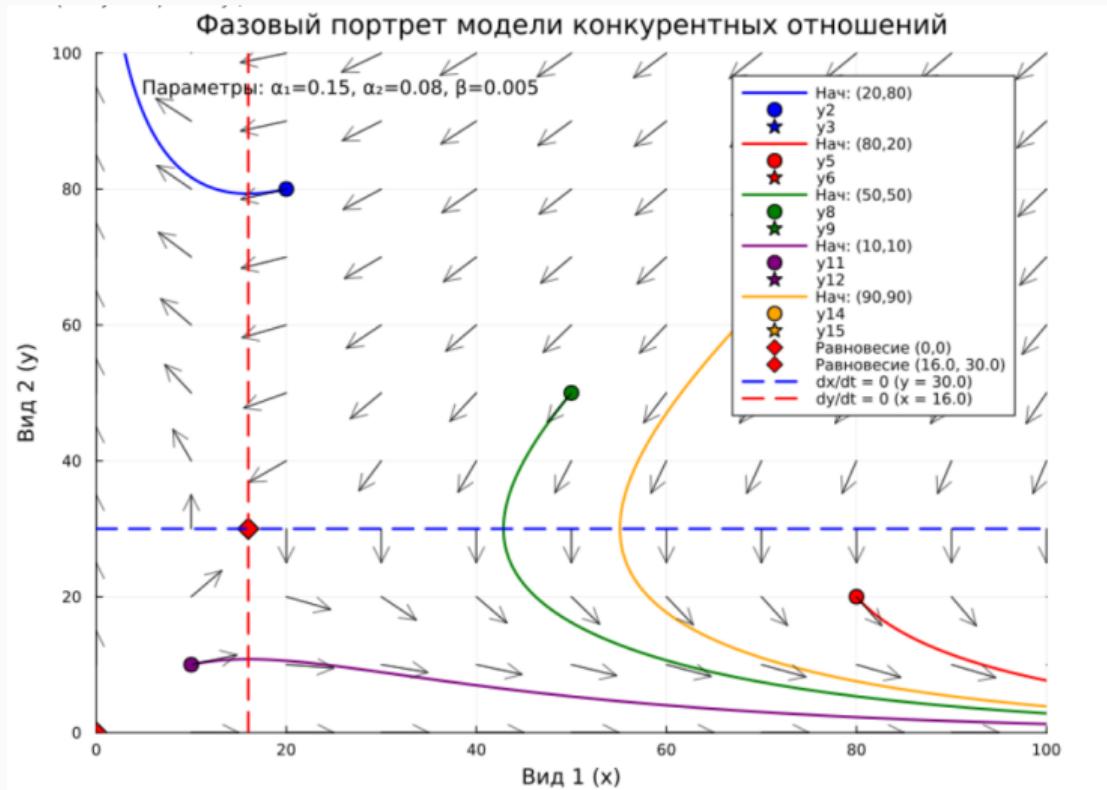


Figure 13: График

## Задание №8

```
|: # ===== ЗАДАНИЕ 8: Модель консервативного гармонического осциллятора =====
# Реализовать модель гармонического осциллятора без затухания
function harmonic_oscillator(du, u, p, t)
    x, v = u
    w0 = p

    du[1] = v           # dx/dt = v
    du[2] = -w0^2 * x  # dv/dt = -w0^2x
end

u0 = [1.0, 0.0]      # начальное положение и скорость
w0 = 1.0             # циклическая частота
tspan = (0.0, 10π)
prob = ODEProblem(harmonic_oscillator, u0, tspan, w0)
sol = solve(prob)

plot(sol, label=["Положение" "Скорость"], title="Гармонический осциллятор",
      xaxis="Время", yaxis="Значение")

# Фазовый портрет
plot(sol, vars=(1,2), label="Фазовый портрет", xaxis="Положение", yaxis="Скорость",
      title="Фазовый портрет гармонического осциллятора", size=(500, 300))
```

Figure 14: Задание №8

# График

Фазовый портрет гармонического осциллятора

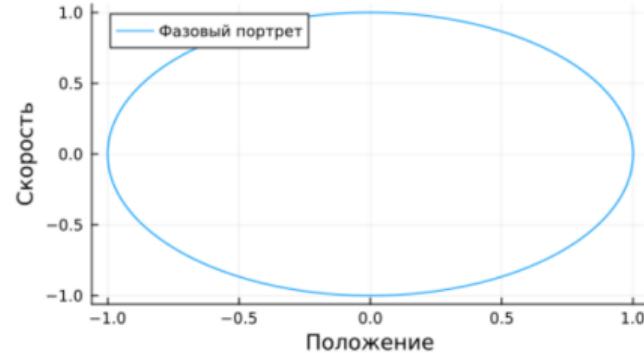


Figure 15: График

# Задание №9

```
# ===== ЗАДАНИЕ 9: Модель свободных колебаний гармонического осциллятора =====
# Реализовать модель гармонического осциллятора с затуханием
function damped_oscillator(du, u, p, t)
    x, v = u
    ω₀, γ = p

    du[1] = v # dx/dt = v
    du[2] = -2γ*v - ω₀^2 * x # dv/dt = -2γv - ω₀²x
end

u₀ = [1.0, 0.0] # начальное положение и скорость
ω₀ = 1.0 # циклическая частота
γ = 0.1 # коэффициент затухания
tspan = (0.0, 20.0)
prob = ODEProblem(damped_oscillator, u₀, tspan, (ω₀, γ))
sol = solve(prob)

plot(sol, label=["Положение" "Скорость"], title="Затухающий гармонический осциллятор",
      xaxis="Время", yaxis="Значение")

# Фазовый портрет
plot(sol, vars=(1,2), label="Фазовый портрет", xaxis="Положение", yaxis="Скорость",
      title="Фазовый портрет затухающего осциллятора")
```

Figure 16: Задание №9

# График

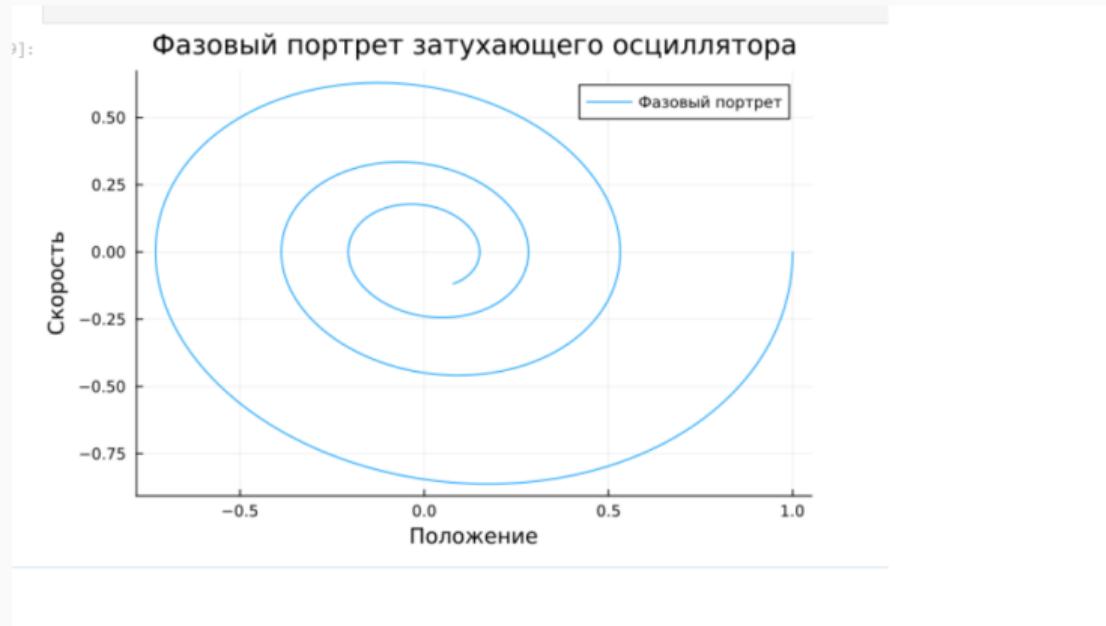


Figure 17: График

## Выводы

Освоил специализированные пакеты для решения задач в непрерывном и дискретном времени.