

# Лабораторная работа №6

## Решение моделей в непрерывном и дискретном времени

- Выполнил:
- Студент группы: НПИбд-02-21
- Студенческий билет: № 1032217060
- ФИО студента: Королев Адам Маратович

# Цели работы:

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

# Выполнение работы:

## 1. Используя Jupyter Lab, повторите примеры из раздела 6.2.

```
# Лабораторная работа №6. Решение моделей в непрерывном и дискретном времени
# 6.2. Предварительные сведения

import Pkg
Pkg.add(["DifferentialEquations", "Plots"])

using DifferentialEquations, Plots

# 6.2.1.1. 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)

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

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

# 6.2.1.2. Система Лоренца
function lorenz!(du, u, p, t)
    σ, ρ, β = p
    du[1] = σ * (u[2] - u[1])
    du[2] = u[1] * (ρ - u[3]) - u[2]
    du[3] = u[1] * u[2] - β * u[3]
end

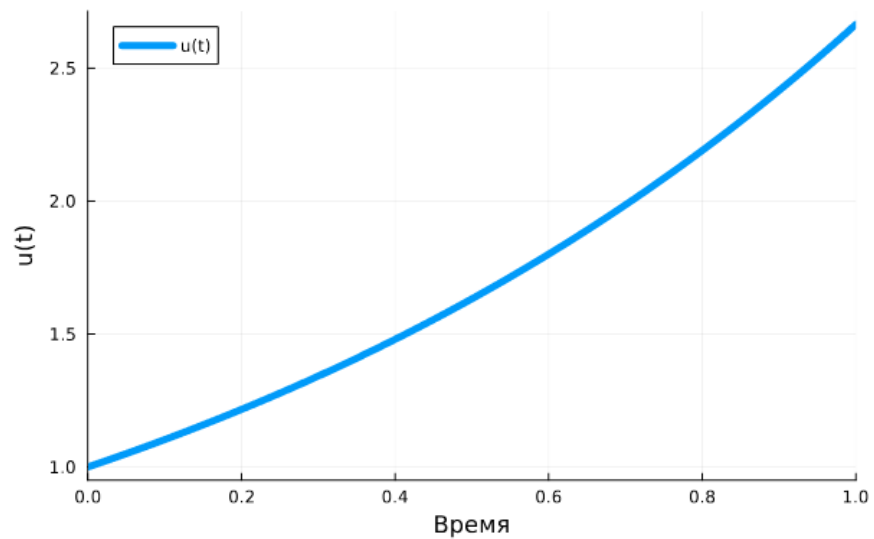
u0 = [1.0, 0.0, 0.0]
p = (10, 28, 8/3)
tspan = (0.0, 100.0)

prob = ODEProblem(lorenz!, u0, tspan, p)
sol = solve(prob)

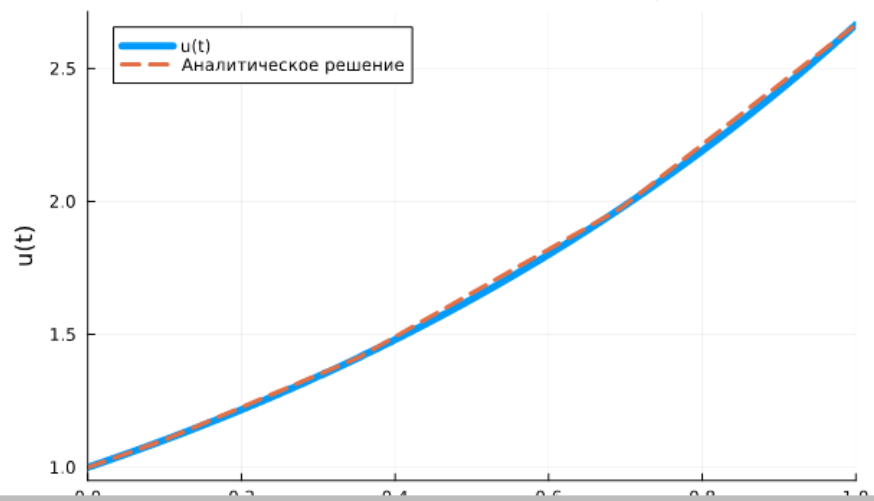
display(plot(sol, vars=(1, 2, 3), lw=2, title="Аттрактор Лоренца", xaxis="x", yaxis="y", zaxis="z", legend=false))

# Фазовый портрет без интерполяции
display(plot(sol, vars=(1, 2, 3), denseplot=false, lw=1, title="Аттрактор Лоренца (без интерполяции)", xaxis="x", yaxis="y", zaxis="z", legend=false))
```

Модель экспоненциального роста



Модель экспоненциального роста



```
# Фазовый портрет без интерполяции
display(plot(sol, vars=(1, 2, 3), denseplot=false, lw=1, title="Аттрактор Лоренца (без интерполяции)", хaxis="x", уaxis="y", zaxis="z", legend=false))

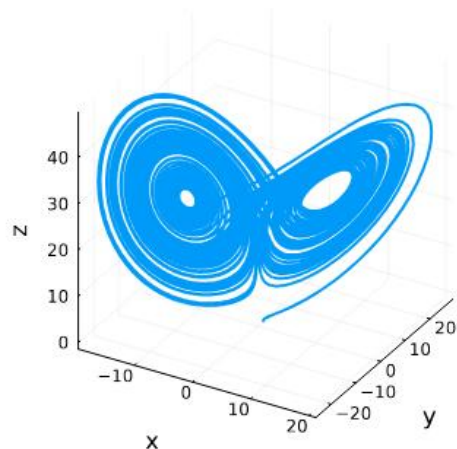
# 6.2.2. Модель Лотки-Вольтерры
function lotka_volterra!(du, u, p, t)
    a, b, c, d = p
    du[1] = a * u[1] - b * u[1] * u[2]
    du[2] = -c * u[2] + d * u[1] * u[2]
end

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

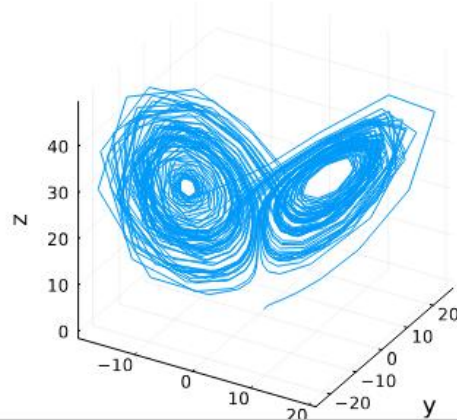
prob = ODEProblem(lotka_volterra!, u0, tspan, p)
sol = solve(prob)

display(plot(sol, label=["Жерты" "Хищники"], title="Модель Лотки-Вольтерры", хaxis="Время", уaxis="Популяция"))
display(plot(sol, vars=(1, 2), color="black", хaxis="Жерты", уaxis="Хищники", legend=false))
```

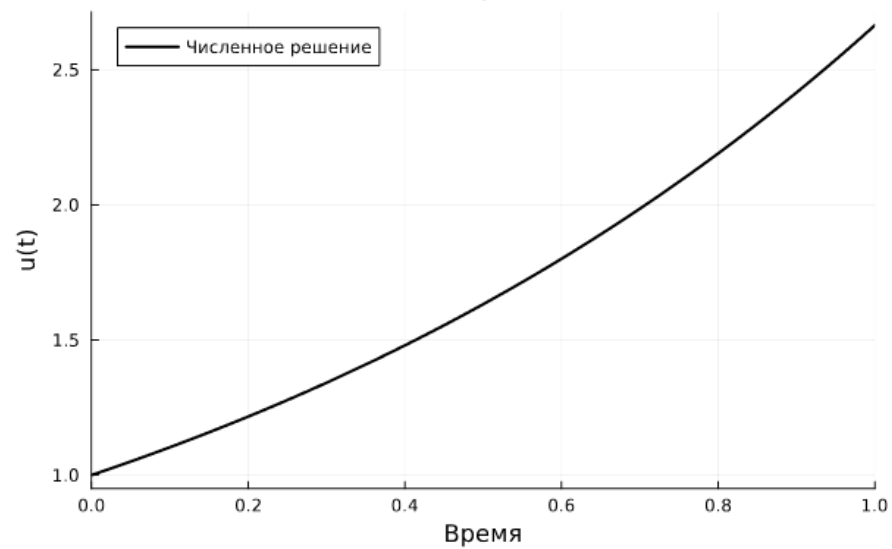
время  
Аттрактор Лоренца



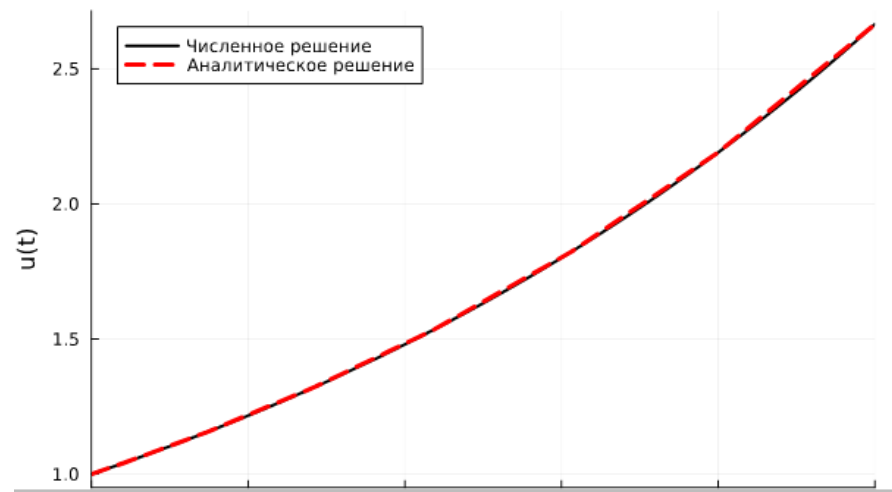
Аттрактор Лоренца (без интерполяции)



Модель экспоненциального роста (с высокой точност

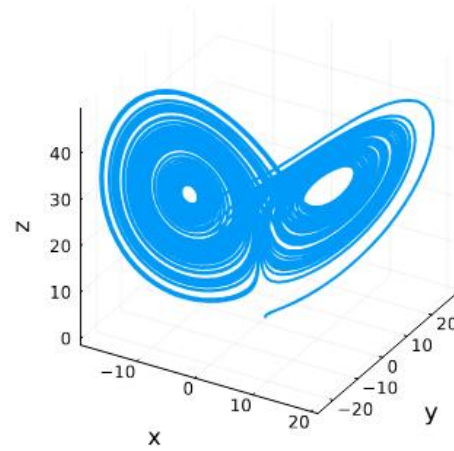


Модель экспоненциального роста (с высокой точност

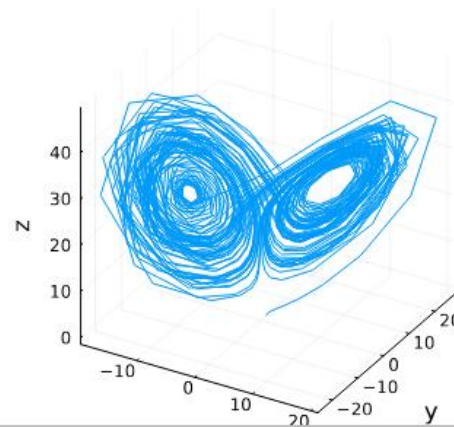


время

### Аттрактор Лоренца



### Аттрактор Лоренца (без интерполяции)



## 2. Выполните задания для самостоятельной работы (раздел 6.4).

```
# 1. Модель Мальтуса
import Pkg
Pkg.add("DifferentialEquations")
Pkg.add("Plots")
using DifferentialEquations, Plots

# Параметры
b = 0.5 # рождаемость
c = 0.2 # смертность
a = b - c
u0 = 10.0

# Модель
f(u, p, t) = a * u
prob = ODEProblem(f, u0, (0.0, 10.0))
sol = solve(prob)

plot(sol, linewidth=3, title="Модель Мальтуса", xlabel="Время", ylabel="Численность популяции")
display(plot(sol))

# 2. Логистическая модель
r = 0.5
k = 100
f_log(u, p, t) = r * u * (1 - u / k)
prob_log = ODEProblem(f_log, u0, (0.0, 20.0))
sol_log = solve(prob_log)

plot(sol_log, linewidth=3, title="Логистическая модель", xlabel="Время", ylabel="Численность популяции")
display(plot(sol_log))

# 3. SIR-модель
beta = 0.3
nu = 0.1
function sir!(du, u, p, t)
    s, i, r = u
    du[1] = -beta * s * i
    du[2] = beta * s * i - nu * i
    du[3] = nu * i
end
```

```

# 5. Дискретная модель Лотки-Вольтерры
a_lv = 2
c_lv = 1
d_lv = 5
function discrete_lv(x1, x2)
    return (a_lv * x1 * (1 - x1) - x1 * x2, -c_lv * x2 + d_lv * x1 * x2)
end

x1 = 0.1
x2 = 0.1
results = [(x1, x2)]
for t in 1:50
    x1, x2 = discrete_lv(x1, x2)
    push!(results, (x1, x2))
end

plot(first.(results), last.(results), linewidth=3, title="Дискретная модель Лотки-Вольтерры", xlabel="X1", ylabel="X2")
display(plot(first.(results), last.(results)))

# 6. Модель конкуренции
alpha = 0.5
beta = 0.1
function competition!(du, u, p, t)
    x, y = u
    du[1] = alpha * x - beta * x * y
    du[2] = alpha * y - beta * x * y
end
u0_comp = [1.0, 0.5]
prob_comp = ODEProblem(competition!, u0_comp, (0.0, 20.0))
sol_comp = solve(prob_comp)

plot(sol_comp, linewidth=3, label=["X" "Y"], title="Модель конкуренции", xlabel="Время", ylabel="Численность")
display(plot(sol_comp))

# 7. Гармонический осциллятор
omega0 = 2 * pi
function harmonic!(du, u, p, t)
    du[1] = u[2]
    du[2] = -omega0^2 * u[1]
end
u0_harm = [1.0, 0.0]
prob_harm = ODEProblem(harmonic!, u0_harm, (0.0, 10.0))

```

```

# 3. SIR-модель
beta = 0.3
nu = 0.1
function sir!(du, u, p, t)
    s, i, r = u
    du[1] = -beta * s * i
    du[2] = beta * s * i - nu * i
    du[3] = nu * i
end
u0_sir = [0.99, 0.01, 0.0]
prob_sir = ODEProblem(sir!, u0_sir, (0.0, 100.0))
sol_sir = solve(prob_sir)

plot(sol_sir, linewidth=3, label=["S" "I" "R"], title="SIR-модель", xlabel="Время", ylabel="Доля популяции")
display(plot(sol_sir))

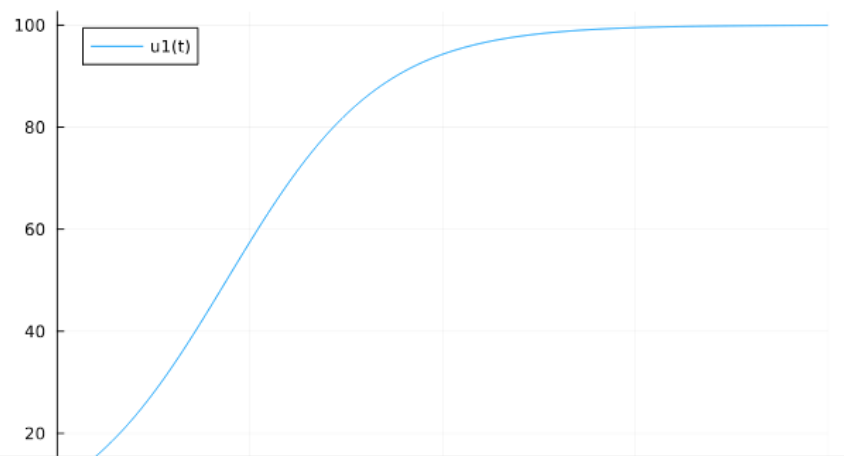
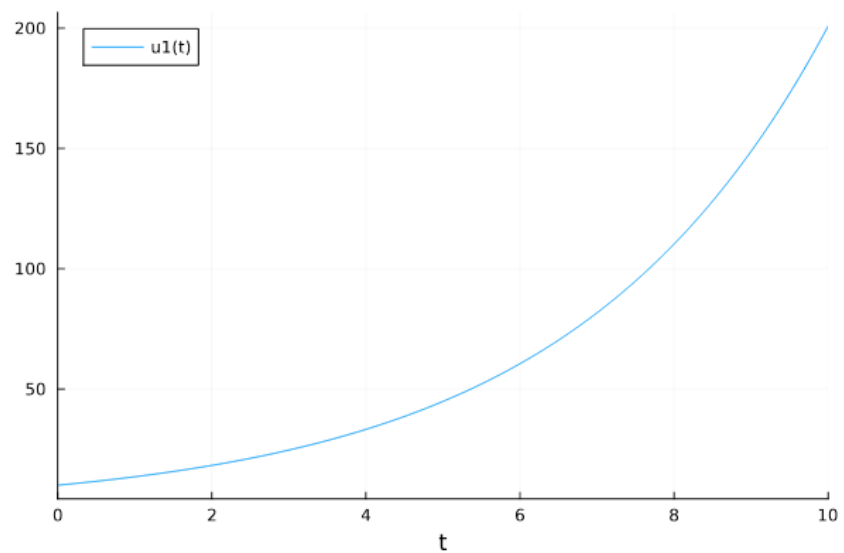
# 4. SEIR-модель
beta_seir = 0.3
nu_seir = 0.1
delta = 0.2
N = 1.0
function seir!(du, u, p, t)
    s, e, i, r = u
    du[1] = -beta_seir / N * s * i
    du[2] = beta_seir / N * s * i - delta * e
    du[3] = delta * e - nu_seir * i
    du[4] = nu_seir * i
end
u0_seir = [0.99, 0.01, 0.0, 0.0]
prob_seir = ODEProblem(seir!, u0_seir, (0.0, 100.0))
sol_seir = solve(prob_seir)

plot(sol_seir, linewidth=3, label=["S" "E" "I" "R"], title="SEIR-модель", xlabel="Время", ylabel="Доля популяции")
display(plot(sol_seir))

# 5. Дискретная модель Лотки-Вольтерры
a_lv = 2
c_lv = 1
d_lv = 5
function discrete_lv(x1, x2)

```



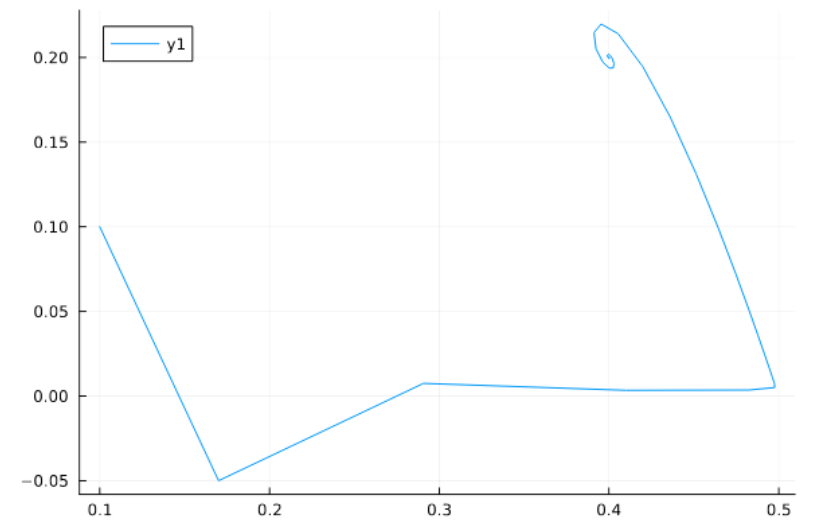
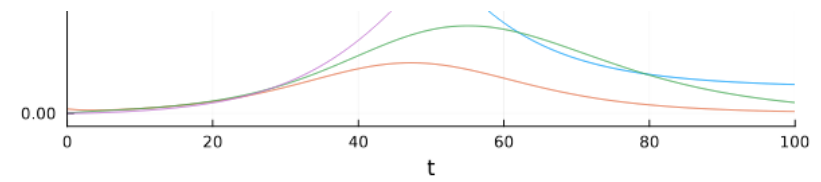
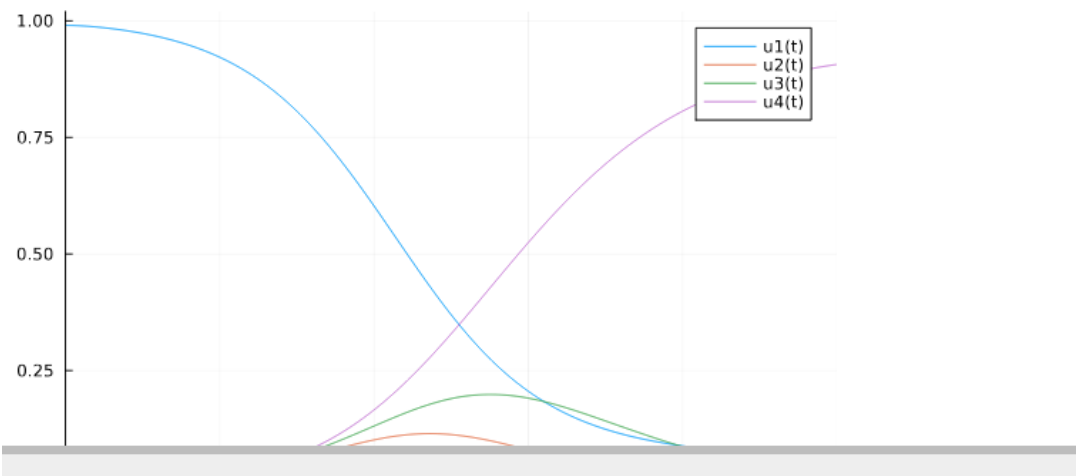
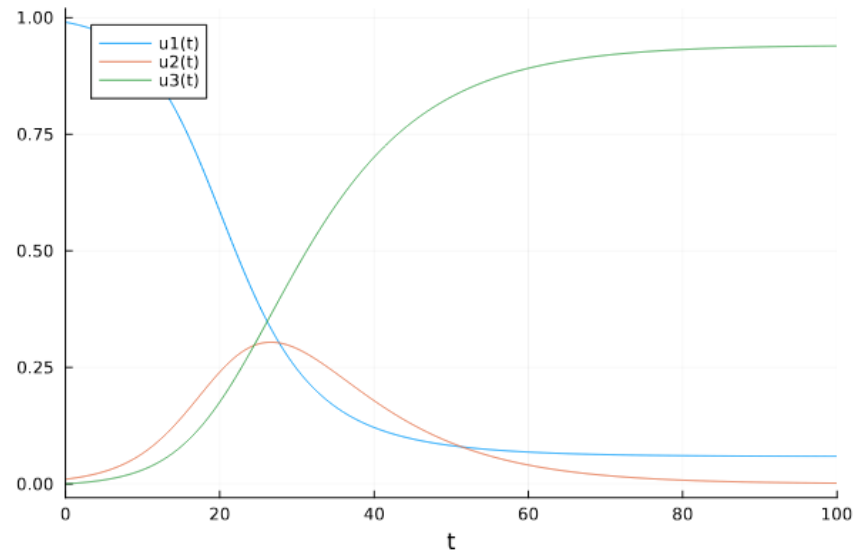
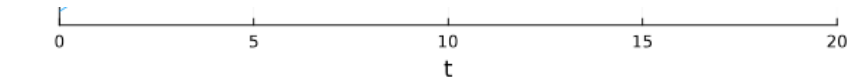


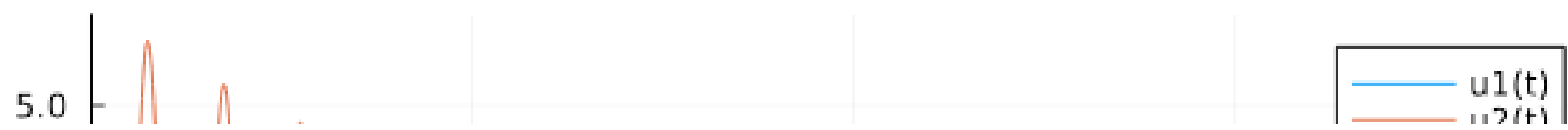
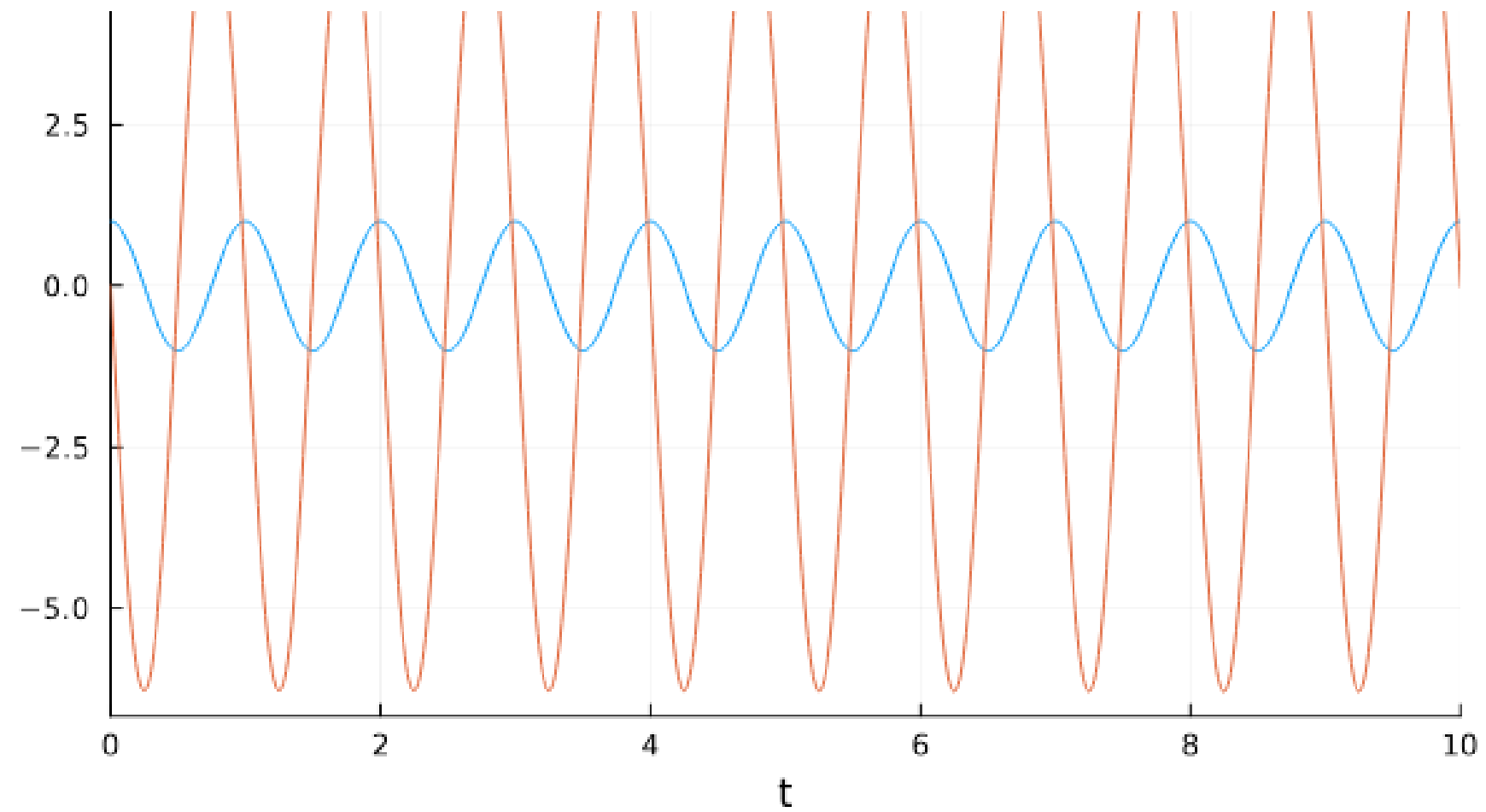
```
# 7. Гармонический осциллятор
omega0 = 2 * pi
function harmonic!(du, u, p, t)
    du[1] = u[2]
    du[2] = -omega0^2 * u[1]
end
u0_harm = [1.0, 0.0]
prob_harm = ODEProblem(harmonic!, u0_harm, (0.0, 10.0))
sol_harm = solve(prob_harm)

plot(sol_harm, linewidth=3, label=["x(t)" "dx/dt"], title="Гармонический осциллятор", xlabel="Время", ylabel="Амплитуда")
display(plot(sol_harm))

# 8. Свободные колебания гармонического осциллятора
omega0_damped = 2 * pi
gamma = 0.1
function damped_harmonic!(du, u, p, t)
    du[1] = u[2]
    du[2] = -2 * gamma * u[2] - omega0_damped^2 * u[1]
end
u0_damped = [1.0, 0.0]
prob_damped = ODEProblem(damped_harmonic!, u0_damped, (0.0, 20.0))
sol_damped = solve(prob_damped)

plot(sol_damped, linewidth=3, label=["x(t)" "dx/dt"], title="Свободные колебания осциллятора", xlabel="Время", ylabel="Амплитуда")
display(plot(sol_damped))
```







# Выводы:

- В процессе выполнения работы мною были освоены специализированные пакеты для решения задач в непрерывном и дискретном времени.
-