Лабораторная работа №6

Линейная алгебра

Клюкин Михаил Александрович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Задание

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

2. Выполните задания для самостоятельной работы.

3 Выполнение лабораторной работы

3.1 Решение обыкновенных дифференциальных уравнений

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

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

3.1.1 Модель экспоненциального роста

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

Численное решение в Julia будет иметь следующий вид и график соответсвующий полученному решению (рис. 3.1 - 3.2):

```julia
import Pkg
using DifferentialEquations
```

```julia
# задаём описание модели с начальными условиями:
a = 0.98
f(u,p,t) = a*u
u0 = 1.0

# задаём интервал времени:
tspan = (0.0,1.0)

# решение:
prob = ODEProblem(f,u0,tspan)
sol = solve(prob)
```

```
retcode: Success
Interpolation: 3rd order Hermite
t: 5-element Vector{Float64}:
 0.0
 0.10042494449239292
 0.3521860297865888
 0.6934436122197829
 1.0
u: 5-element Vector{Float64}:
 1.0
 1.1034222047865465
 1.4121908713484919
 1.9730384457359198
 2.6644561424814266
```

```julia
using Plots

# строим графики:
plot(sol, linewidth=5,title="Модель экспоненциального роста", xaxis="Время",yaxis="u(t)",l

plot!(sol.t, t->1.0*exp(a*t),lw=3,ls=:dash,label="Аналитическое решение")
```

Рис. 3.1: Численное решение

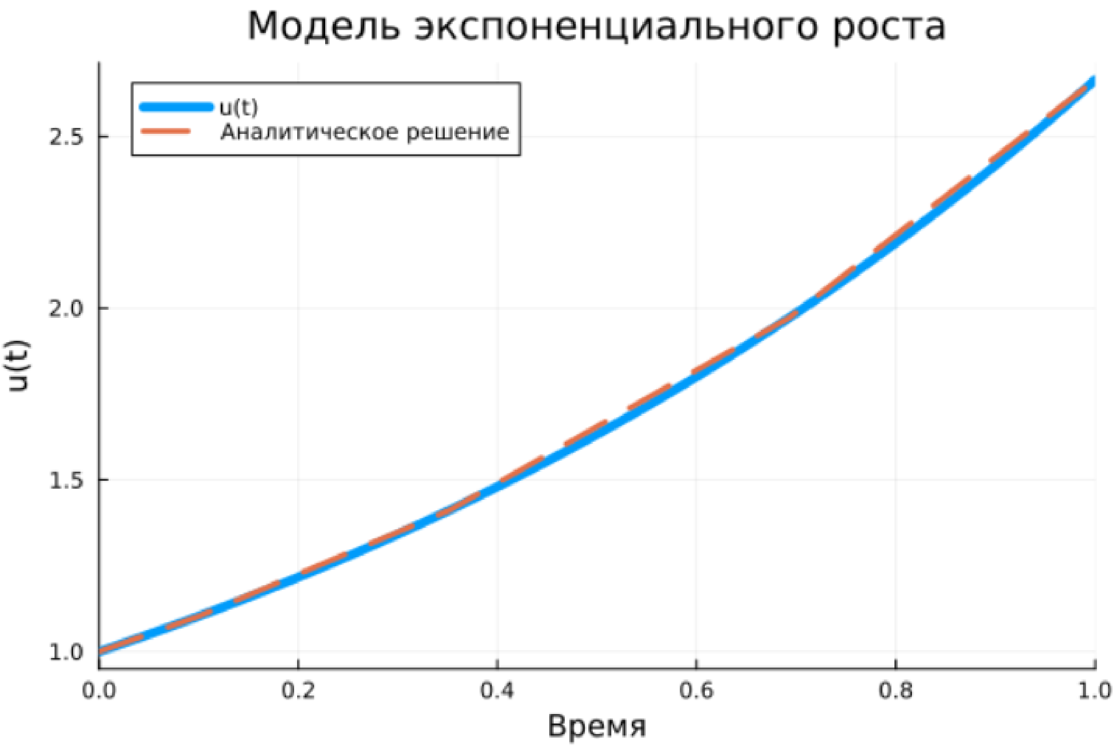

Рис. 3.2: График численного решения

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. 3.3 - 3.4):

```
# задаём точность решения:
sol = solve(prob,abstol=1e-8,reltol=1e-8)
println(sol)

# строим график:
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")
```

Рис. 3.3: Построение графика численного решения

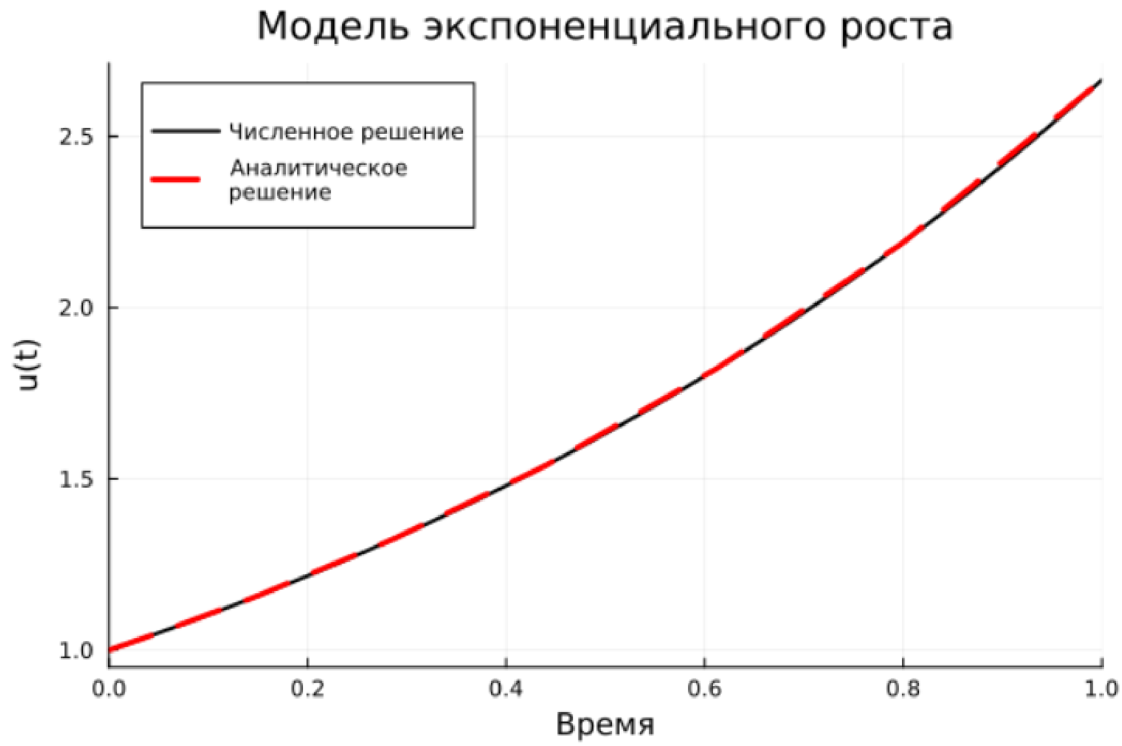

Рис. 3.4: График численного рещения

3.1.2 Система Лоренца

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

Система (6.2) 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 последующем 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 и график (рис. 3.5 - 3.6):

```
# задаём точность решения:
sol = solve(prob,abstol=1e-8,reltol=1e-8)
println(sol)

# строим график:
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")

# задаём описание модели:
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

# задаём начальное условие:
u0 = [1.0,0.0,0.0]

# задаём значения параметров:
p = (10,28,8/3)

# задаём интервал времени:
tspan = (0.0,100.0)

# решение:
prob = ODEProblem(lorenz!,u0,tspan,p)
sol = solve(prob)
```

Рис. 3.5: Численное решение системы Лоренца

```
# строим график:
plot(sol, vars=(1,2,3), lw=2, title="Аттрактор Лоренца", xaxis="x",yaxis="y", zaxis="z",legend=false)
```

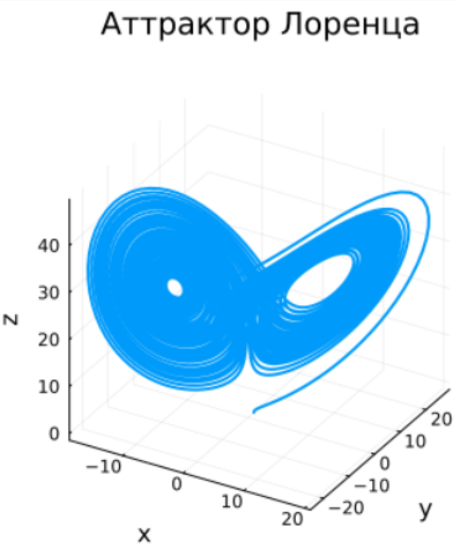

Рис. 3.6: Аттрактор Лоренца

Можно отключить интерполяцию (рис. 3.7).

```
# отключаем интерполяцию:
plot(sol,vars=(1,2,3),denseplot=false, lw=1, title="Аттрактор Лоренца",
xaxis="x",yaxis="y", zaxis="z",legend=false)
```

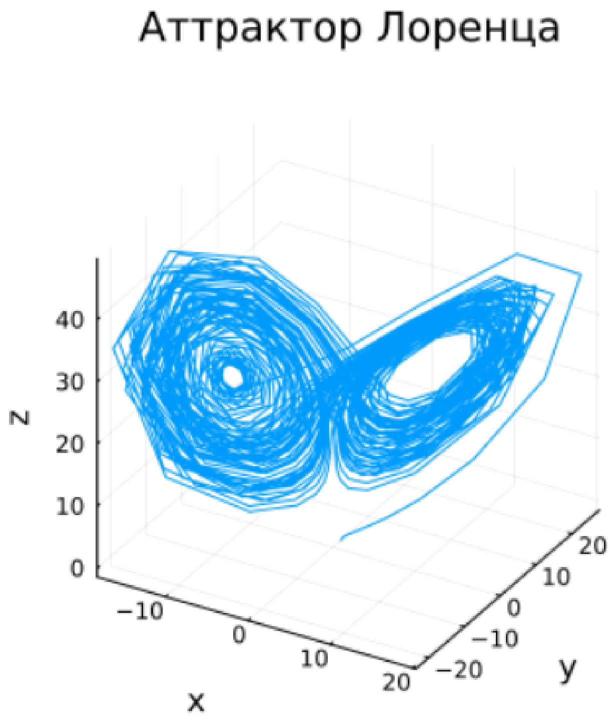

Рис. 3.7: Аттрактор Лоренца (интерполяция отключена)

3.2 Модель Лотки-Вольтерры

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

Численное решение в Julia будет иметь следующий вид, а также график (рис. 3.8 - 3.9).

```julia
Pkg.add("ParameterizedFunctions")
using ParameterizedFunctions, DifferentialEquations, Plots;
```

```
    Resolving package versions...
      Project No packages added to or removed from `~/.julia/environments/v1.12/Project.toml`
     Manifest No packages added to or removed from `~/.julia/environments/v1.12/Manifest.toml`
```

```julia
# задаём описание модели:
lv! = @ode_def LotkaVolterra begin
    dx = a*x - b*x*y
    dy = -c*y + d*x*y
end a b c d

# задаём начальное условие:
u0 = [1.0,1.0]
# задаём значения параметров:
p = (1.5,1.0,3.0,1.0)
# задаём интервал времени:
tspan = (0.0,10.0)

# решение:
prob = ODEProblem(lv!,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")
```

Рис. 3.8: Численное решение для модели Лотки-Волтерры

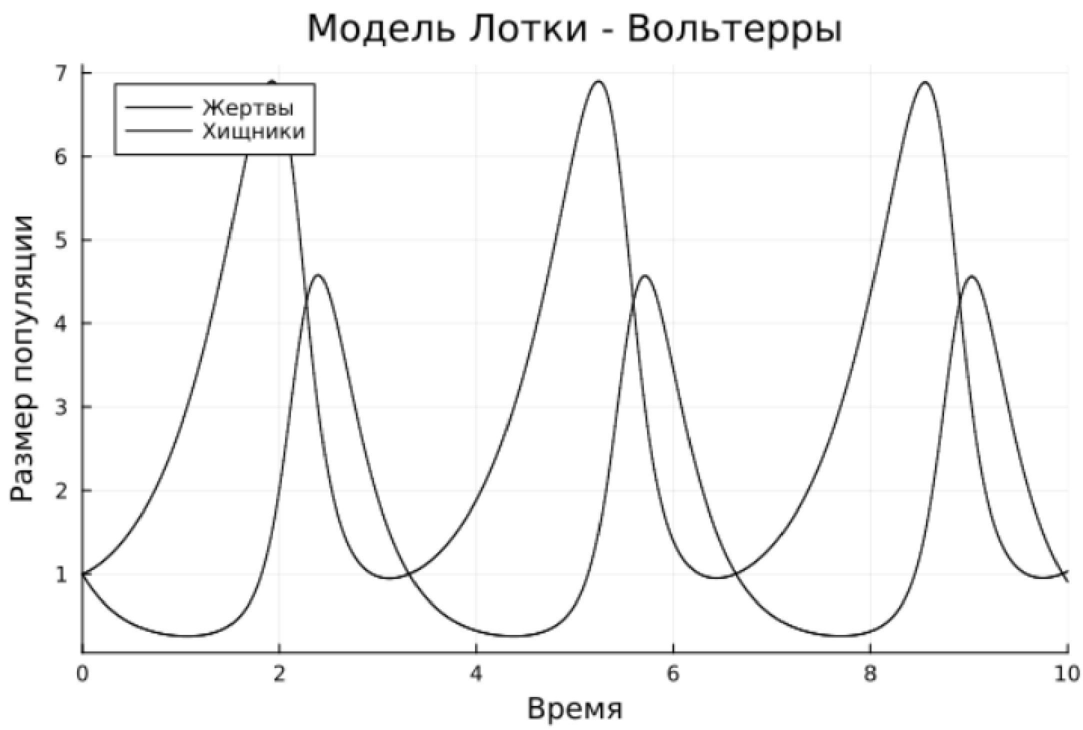

Рис. 3.9: График численного решения для модели Лотки-Волтерры

Фазовый портрет (рис. 3.10).

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

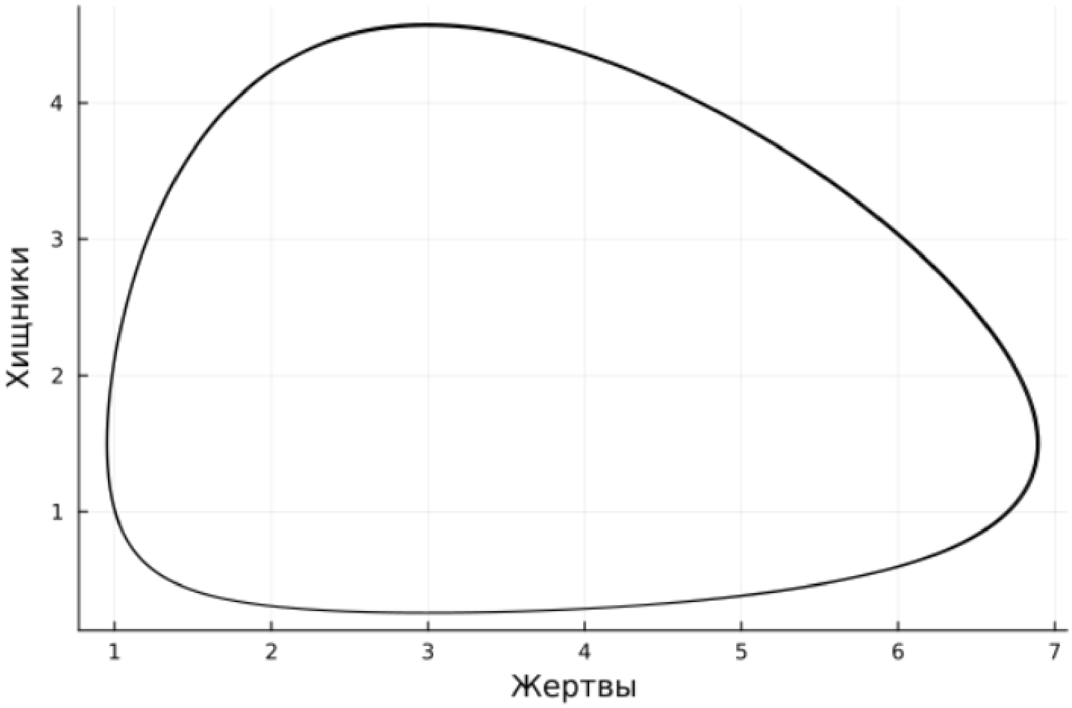

Рис. 3.10: Фазовый портрет для модели Лотки-Волтерры

4 Задания для самостотельного выполнения

Выполнение задания 1 (рис. 4.1 - 4.2):

```
f(u, p, t) = a * u
b = 0.5
c = 0.1
a = b - c
u0 = 1.0
tspan = (0.0, 10.0)
prob = ODEProblem(f, u0, tspan)
sol = solve(prob)
plot(sol, title="Модель Мальтуса", label="Численное решение", color="blue", linewidth=2)
```

Рис. 4.1: Численное решение для модели Мальтуса

```
anim = @animate for i in 1:length(sol.t)
    plot(sol.t[1:i], sol.u[1:i], linewidth=2, title="Модель Мальтуса", xlabel="Время", ylabel="Численность популяции", legend=false,
end
gif(anim, "maltus.gif", fps=15)
```

```
[ Info: Saved animation to /home/sc24/rudn/2026-1--study--computer-practice/labs/lab6/maltus.gif
```

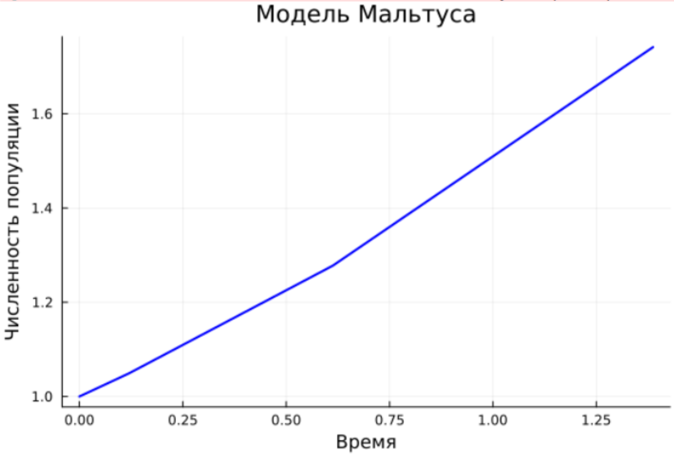

Рис. 4.2: График для модели Мальтуса

Выполнение задания 2 (рис. 4.3 - 4.4):

```
f(u, p, t) = r * u * (1 - u/k)
r = 0.5
k = 100
u0 = 1.0
tspan = (0.0, 20.0)
prob = ODEProblem(f, u0, tspan)
sol = solve(prob)
plot(sol, title="Логистическая модель роста популяции", label="Численность популяции", color="blue", linewidth=2)
```

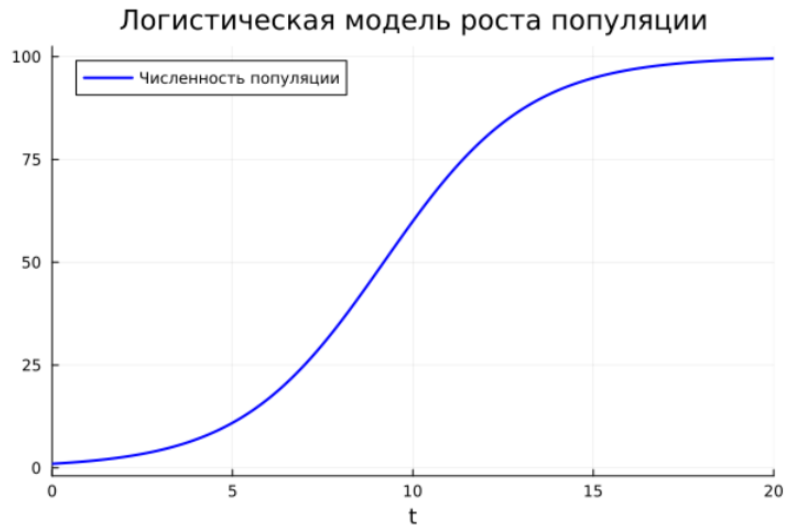

Рис. 4.3: Численное решение для логичтической модели роста популяции

```
anim = @animate for t in 0:0.5:20
    plot(sol.t[sol.t .<= t], sol.u[sol.t .<= t], label="Численное решение", xlim=(0,20), ylim=(0,k), lw=2, color="blue",
end
gif(anim, "logistic.gif", fps=10)
```

```
[ Info: Saved animation to /home/sc24/rudn/2026-1--study--computer-practice/labs/lab6/logistic.gif
```

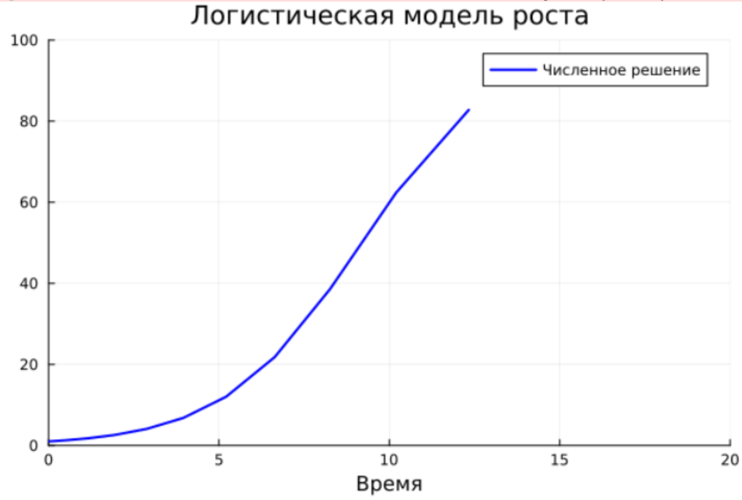

Рис. 4.4: График для логичтической модели роста популяции

Выполнение задания 3 (рис. 4.5 - 4.7):

```
function sir(u, p, t)
    beta, v = p
    s, i, r = u
    ds = -beta * s * i
    di = beta * s * i - v * i
    dr = v * i
    return [ds, di, dr]
end
```

```
sir (generic function with 1 method)
```

```
u0 = [0.99, 0.01, 0.0]
beta = 0.3
v = 0.1
p = (beta, v)
tspan = (0.0, 100.0)
prob_sir = ODEProblem(sir, u0, tspan, p)
sol_sir = solve(prob_sir, Tsit5())
plot(sol_sir, title="SIR", label=["S" "I" "R"])
```

Рис. 4.5: Численное решение для модели SIR

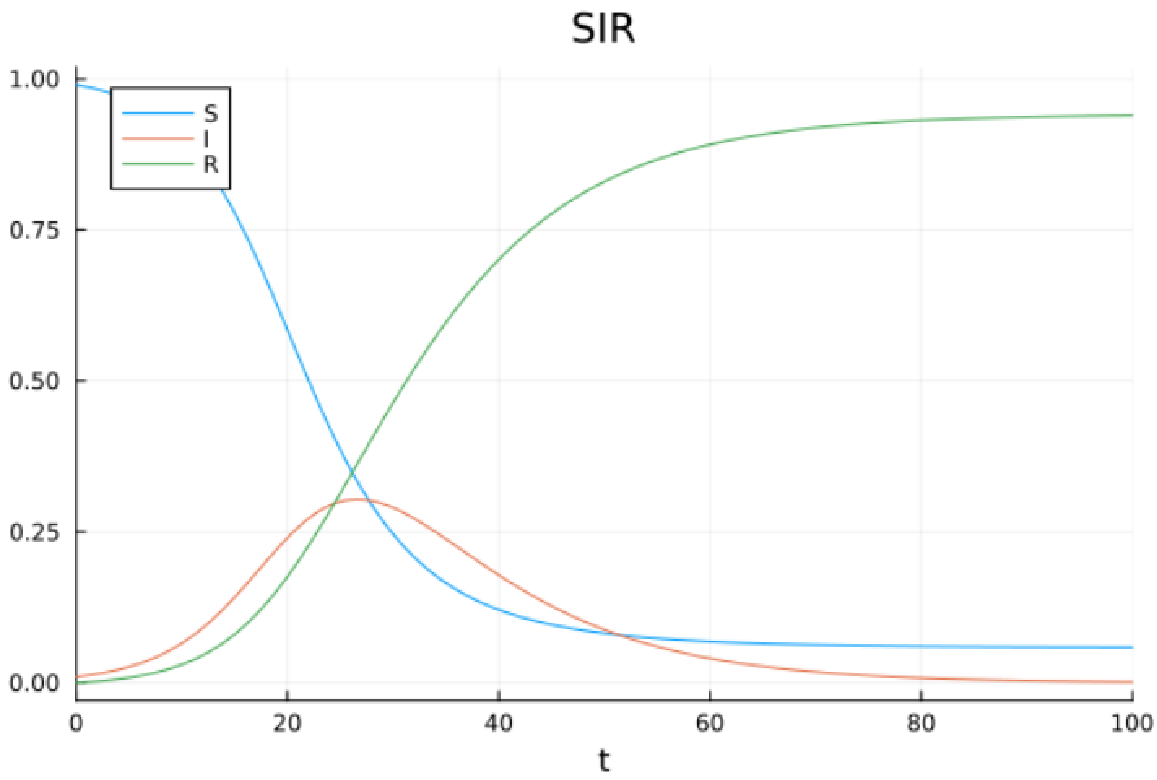

Рис. 4.6: График для модели SIR

```
anim = @animate for t in 1:length(sol_sir.t)
    plot(sol_sir.t[1:t], [u[1] for u in sol_sir.u[1:t]], label="Восприимчивые", color=:blue, linewidth=2)
    plot!(sol_sir.t[1:t], [u[2] for u in sol_sir.u[1:t]], label="Инфицированные", color=:red, linewidth=2)
    plot!(sol_sir.t[1:t], [u[3] for u in sol_sir.u[1:t]], label="Выздоровевшие", color=:green, linewidth=2)
    title!("Модель эпидемии SIR")
    xlabel!("Время")
    ylabel!("Численность")
end
gif(anim, "sir.gif", fps=120)
```

```
[ Info: Saved animation to /home/sc24/rudn/2026-1--study--computer-practice/labs/lab6/sir.gif
```

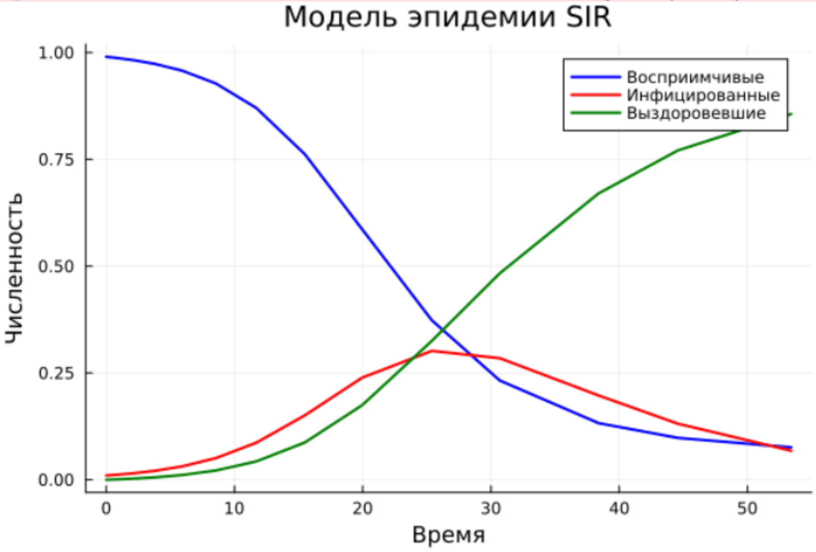

Рис. 4.7: График для модели SIR

Выполнение задания 4 (рис. 4.8 - 4.9):

```julia
function seir(u,p,t)
    (s,e,i,r) = u
    (beta, gamma, delta) = p
    N = s + e + i + r
    ds = -(beta * s * i) / N
    de = (beta * s * i) / N - delta * e
    di = delta * e - gamma * i
    dr = gamma * i
    return [ds, de, di, dr]
end
```

```
seir (generic function with 1 method)
```

```julia
dt_ = 0.1
u0 = [980.0, 10.0, 10.0, 0.0]
p = [0.3, 0.1, 0.2]
tmax = 60.0
tspan = (0.0, tmax)
prob_1 = ODEProblem(seir, u0, tspan, p)
sol = solve(prob_1, dt = dt_)
plot(sol, title="SEIR", label=["S" "E" "I" "R"])
```

Рис. 4.8: Численное решение для модели SEIR

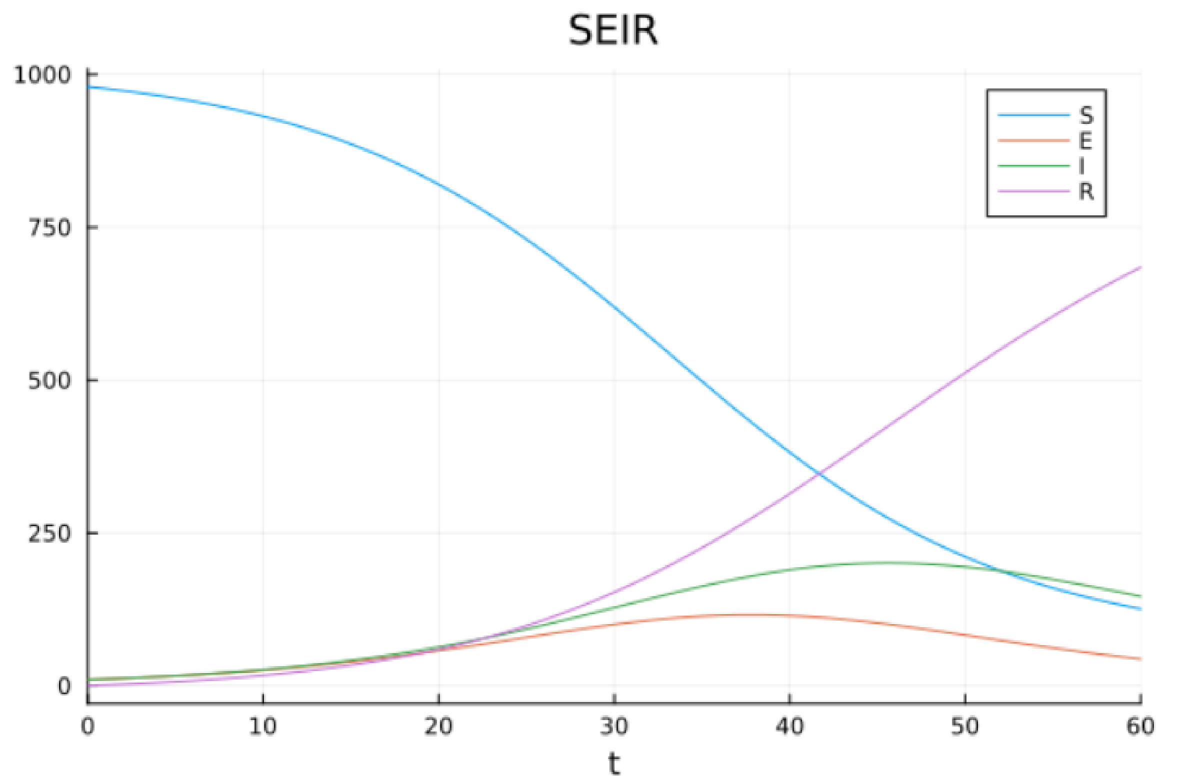

Рис. 4.9: График для модели SEIR

Выполнение задания 5 (рис. 4.10 - 4.12):

```
a = 2
c = 1
d = 5
X0 = [0.5, 0.3]
tspan = 50
X1 = zeros(tspan)
X2 = zeros(tspan)
X1[1] = X0[1]
X2[1] = X0[2]
for t in 1:tspan-1
    X1[t+1] = a * X1[t] * (1 - X1[t]) - X1[t] * X2[t]
    X2[t+1] = -c * X2[t] + d * X1[t] * X2[t]
end
plot(1:tspan, X1, label="Жертвы(X1)", xlabel="Время", ylabel="Численность", title="Численное решение")
plot!(1:tspan, X2, label="Хищники(X2)")
```

Рис. 4.10: Численное решение для дискретной модели Лотки-Вольтерры

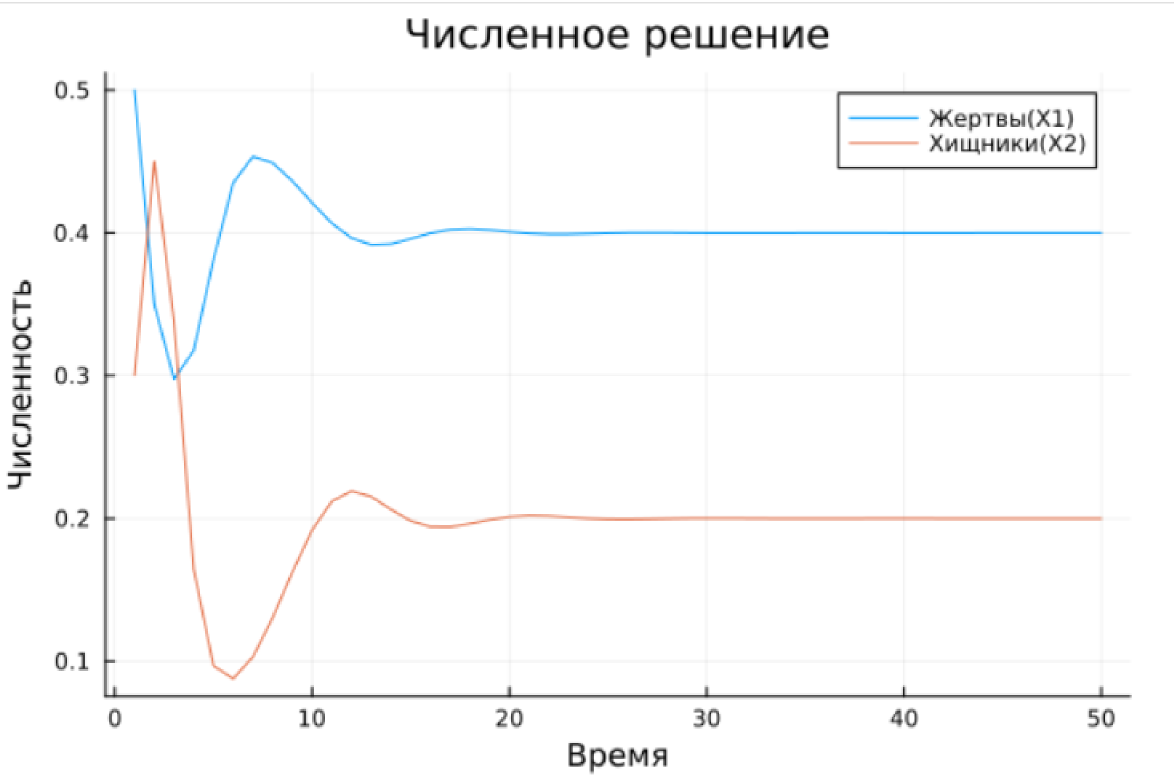

Рис. 4.11: График для дискретной модели Лотки-Вольтерры

```
plot(X1, X2, xlabel="Жертвы(X1)", ylabel="Хищники(X2)", title="Фазовый портрет")
```

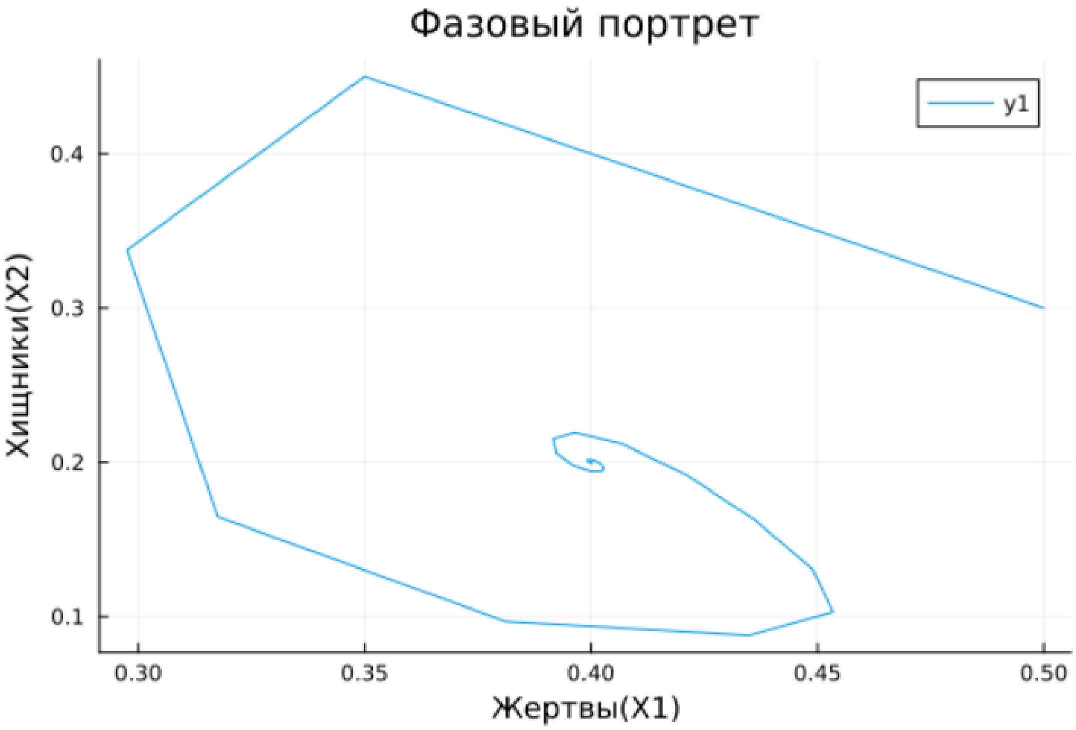

Рис. 4.12: Фазовый портрет для дискретной модели Лотки-Вольтерры

Выполнение задания 6 (рис. 4.13 - 4.15):

```
function competition(du, u, p, t)
    alpha, beta = p
    x, y = u
    du[1] = alpha * x - beta * x * y
    du[2] = alpha * y - beta * x * y
end
```

```
competition (generic function with 1 method)
```

```
tspan = (0.0, 50.0)
u0 = [10.0, 5.0]
p = [1, 0.01]
prob = ODEProblem(competition, u0, tspan, p)
sol = solve(prob, Tsit5())
plot(sol, label=["Популяция x" "Популяция y"], title="Модель конкуренции", xlabel="Время", ylabel="Популяция", linewidth=2)
```

Рис. 4.13: Численное решение для модели отбора на основе конкурентных отношений

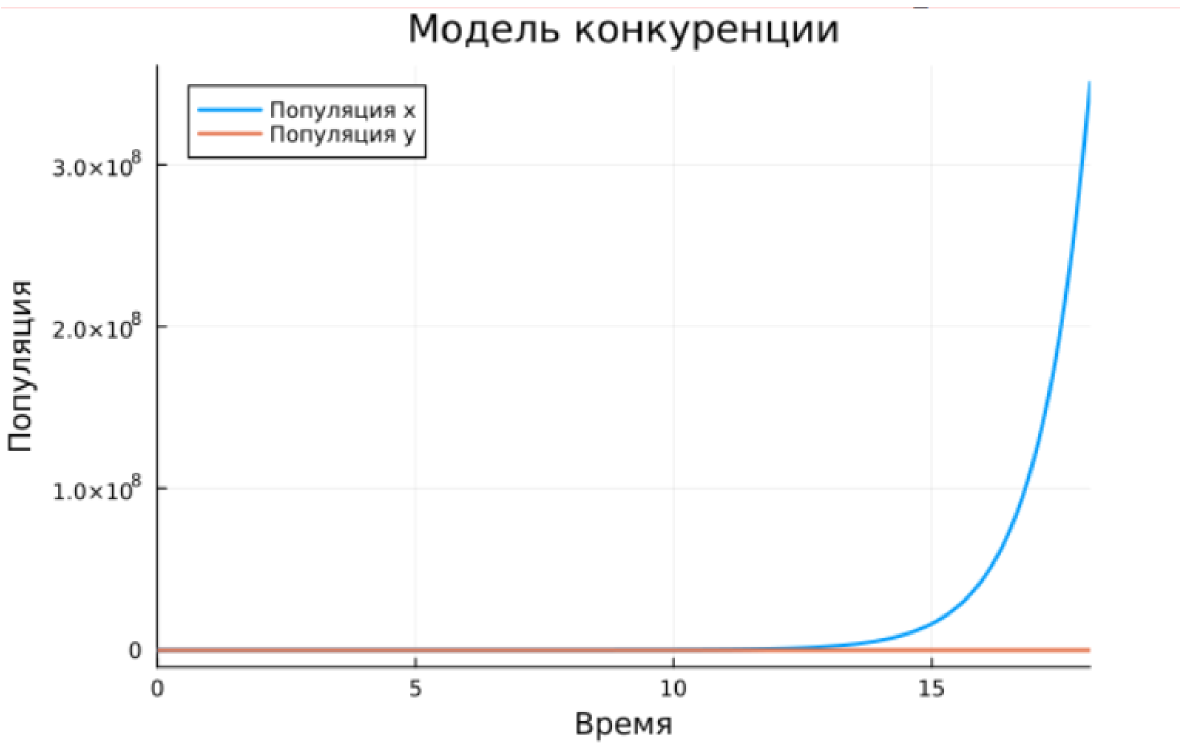

Рис. 4.14: График для модели отбора на основе конкурентных отношений

```
plot(sol, idxs=(1, 2), title="Конкурентные отношения")
```

Рис. 4.15: Фазовый портрет для модели отбора на основе конкурентных отношений

Выполнение задания 7 (рис. 4.16 - 4.18):

```
function harmonic(u, p, t)
    x, y = u
    Y, omega = p
    dx = y
    dy = -2 * Y * y - omega^2 * x
    return [dx, dy]
end
```

```
harmonic (generic function with 1 method)
```

```
tspan = (0.0, 2*pi)
u0 = [-0.5, 0]
p = [0, 3.5]
prob = ODEProblem(harmonic, u0, tspan, p)
sol = solve(prob, Tsit5())
plot(sol, title="Гармонический осциллятор")
```

Рис. 4.16: Численное решение для модели консервативного гармонического осциллятора

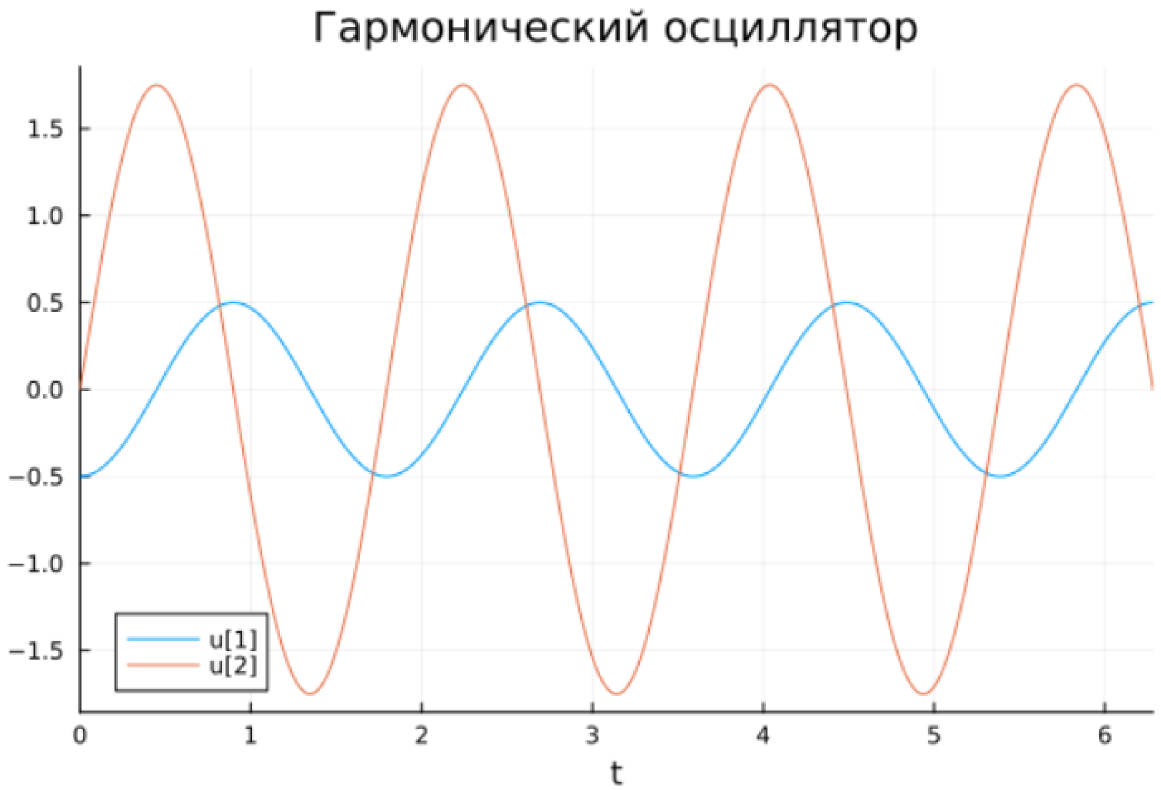

Рис. 4.17: График для модели консервативного гармонического осциллятора

```
plot(sol, idxs=(1,2), title="Фазовый портрет")
```

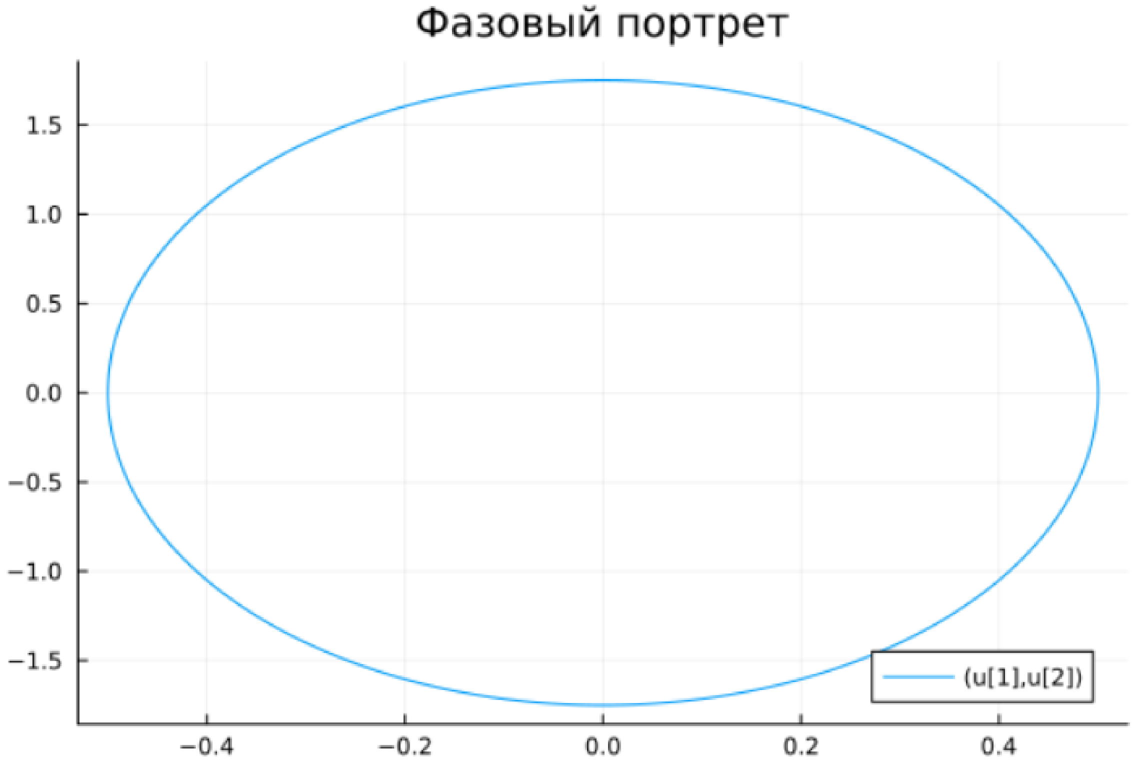

Рис. 4.18: Фазовый портрет для модели консервативного гармонического осциллятора

Выполнение задания 8 (рис. 4.19 - 4.20):

```
tspan = (0.0, 5 * pi)
u0 = [-0.5, 1]
p = [0.4, 3.5]
prob = ODEProblem(harmonic, u0, tspan, p)
sol = solve(prob, Tsit5())
plot(sol, title="Модель свободных колбаний гармонического осциллятора")
```

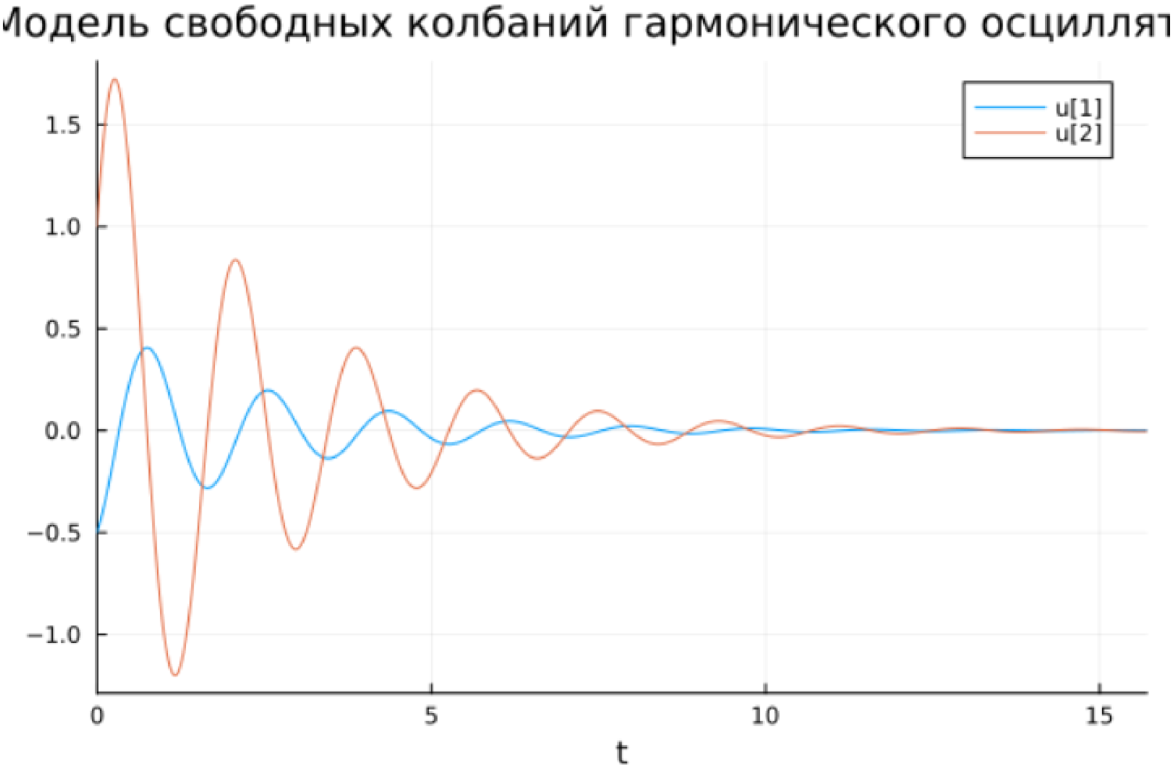

Рис. 4.19: Численное решение для модели свободный колебаний гармонического осциллятора

```
plot(sol, vars=(1,2), title="Фазовый портрет")
```

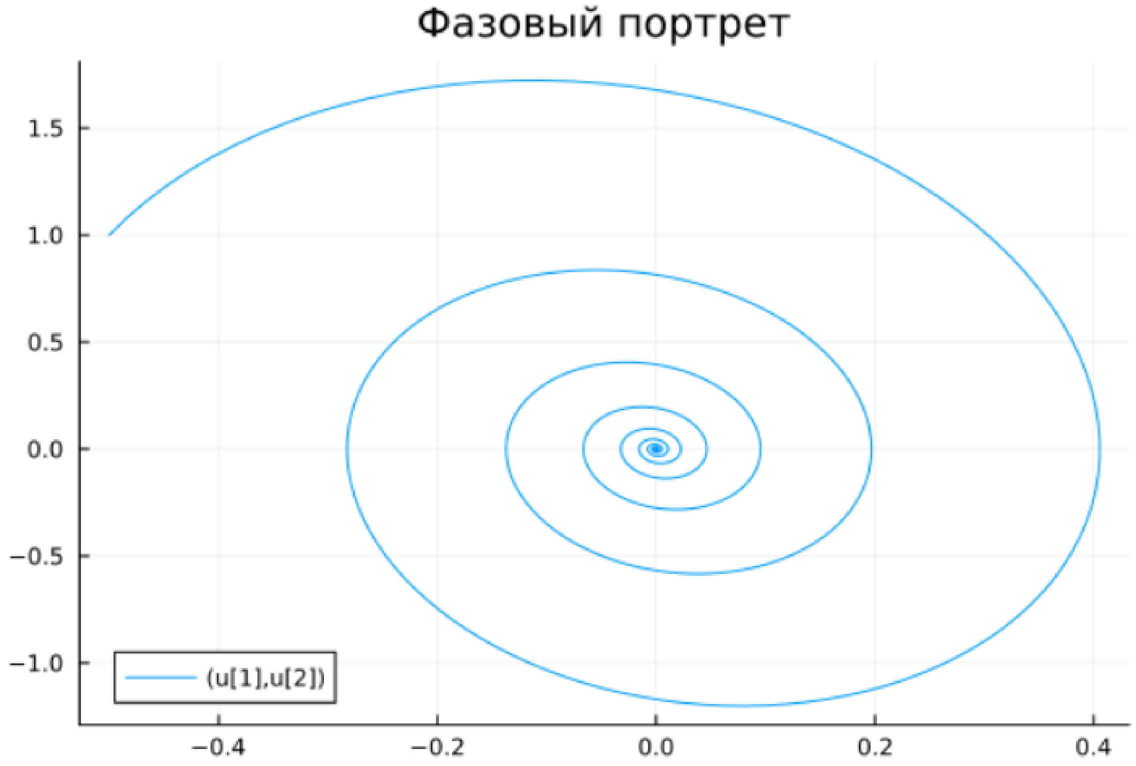

Рис. 4.20: График для модели свободный колебаний гармонического осциллятора

5 Вывод

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