Отчет по лабораторной работе №6

Решение моделей в непрерывном и дискретном времени

Легиньких Галина Андреевна

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 (раздел 6.4).

3 Выполнение лабораторной работы

1. Для начала я повторила все примеры. Тут их не так много. Первый пример это модель экспоненциального роста. Для решения обыкновенных дифференциальных уравнений (ОДУ) в Julia можно использовать пакет diffrentialEquations.jl. Численное решение в Julia будет иметь следующий вид: (рис. 3.1)

```
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
```

Рис. 3.1: Модель экспоненциального роста. Численное решение.

Далее получила графика, соответствующий полученному решению: (рис. 3.2)

[3]:

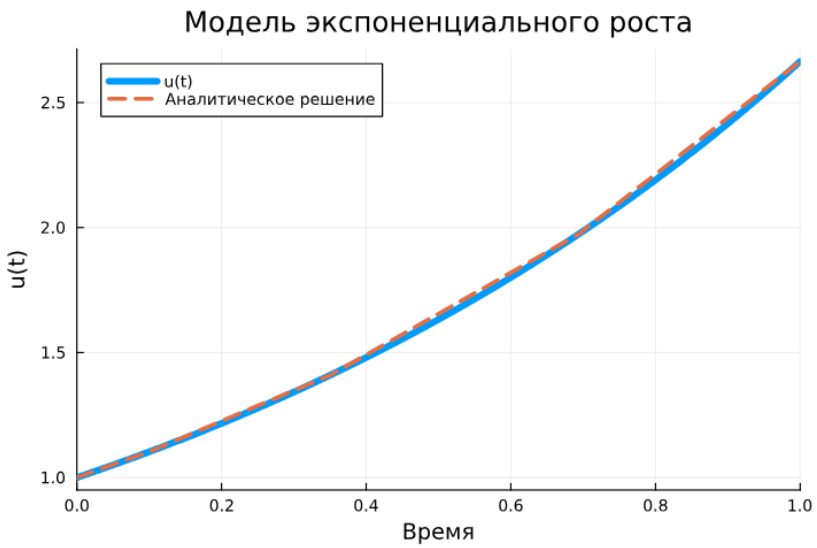

Рис. 3.2: Модель экспоненциального роста. График.

Если требуется задать точность решения, то можно воспользоваться параметрами abstol (задаёт близость к нулю) и reltol (задаёт относительную точность). (рис. 3.3)

```
thing, nothing, :linear, :Standard, OrdinaryDiffEqCore.trivial_limiter!)), false,
10, 3, 9//10, 9//10, 2, false, 5, 2), 2, 1.0, OrdinaryDiffEqTsit5.Tsit5ConstantCa
che(), OrdinaryDiffEqVerner.Vern7ConstantCache(true), #undef, #undef, #undef, #un
def), nothing, false), true, 0, SciMLBase.DEStats(82, 0, 0, 0, 0, 0, 0, 0, 0, 0,
8, 0, 0.0), [2, 2, 2, 2, 2, 2, 2, 2], SciMLBase.ReturnCode.Success, nothing, n
othing, nothing)
```

[5]:

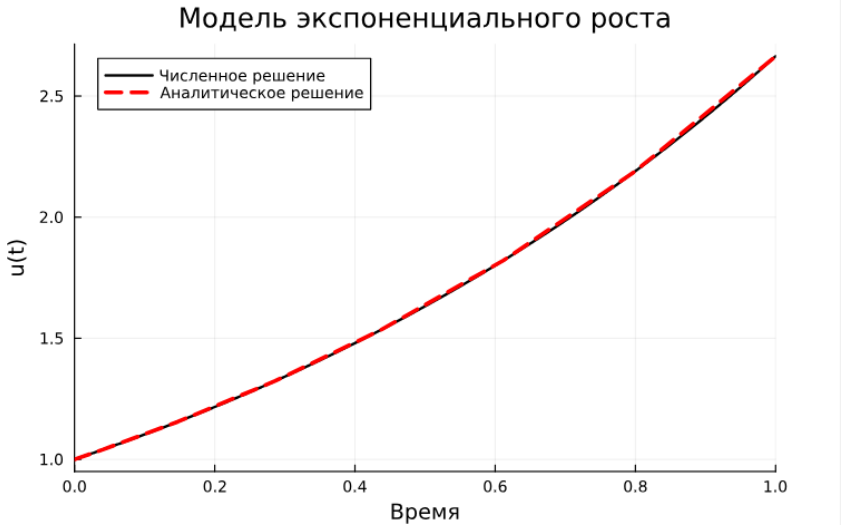

Рис. 3.3: Модель экспоненциального роста. График с точным решением.

2. Далее рассмотрела систему Лоренца. Численное решение в Julia будет иметь следующий вид: (рис. 3.4) (рис. 3.5) (рис. 3.6)

```
retcode: Success
Interpolation: 3rd order Hermite
t: 1292-element Vector{Float64}:
   0.0
   3.567860483630140e-5
   0.0003924646531993154
   0.0032624087311758 73
   0.0090580766226861 89
   0.0169564709017674 3
   0.0276899601164208 83
   0.0418563522196183 44
   0.0602404118654932 96
   0.0836854121090992 4
   0.1133649933610391 1
   0.1486218139342663 2
   0.1870397836913972
   ⋮
  99.25245983911758
  99.3123010170333
  99.37100059024901
  99.43545175575305
  99.50217600300971
  99.56297541572351
  99.62622492183432
  99.69561088424294
  99.77387244562912
  99.86354266863755
  99.93826978918452
 100.0
```

Рис. 3.4: Аттрактор Лоренца. Численное решение t.

```
u: 1292-element Vector{Vector{Float64}}:
 [1.0, 0.0, 0.0]
 [0.9996434557625105, 0.0009988049817849058, 1.781434788799189e-8]
 [0.9961045497425811, 0.010965399721242457, 2.1469553658389193e-6]
 [0.9693591548287857, 0.0897706331002921, 0.00014380191884671585]
 [0.9242043547708632, 0.24228915014052968, 0.0010461625485930237]
 [0.8800455783133068, 0.43873649717821195, 0.003424260078582332]
 [0.8483309823046307, 0.6915629680633586, 0.008487625469885364]
 [0.8495036699348377, 1.0145426764822272, 0.01821209108471829]
 [0.9139069585506618, 1.442559985646147, 0.03669382222358562]
 [1.0888638225734468, 2.0523265829961646, 0.07402573595703686]
 [1.460862686672, 3.020672001462966, 0.1600393577289759]
 [2.1627233814115288, 4.6333636054125975, 0.37711736638953464]
 [3.368464366588119, 7.267694015527519, 0.9363556169983378]
 ⋮
 [14.543594291970454, 8.633828572600308, 40.30844424656028]
 [9.527702621805666, 0.0875695733614436, 37.04712768883169]
 [4.5637551148141755, -2.4307213825654537, 31.150857751066518]
 [1.2013409155396158, -2.429012698730855, 25.83593282347909]
 [-0.4985909866565941, -2.2431908075030083, 21.591758421186338]
 [-1.3554328352527145, -2.5773570617802326, 18.48962628032902]
 [-2.1618698772305467, -3.5957801801676297, 15.934724265473792]
 [-3.433783468673715, -5.786446127166032, 14.065327938066913]
 [-5.971873646288483, -10.261846004477597, 14.060290896024572]
 [-10.941900618598972, -17.312154206417734, 20.65905960858999]
 [-14.71738043327772, -16.96871551014668, 33.06627229408802]
 [-13.714517151605754, -8.323306384833089, 38.798231477265624]
```

Рис. 3.5: Аттрактор Лоренца. Численное решение u(t).

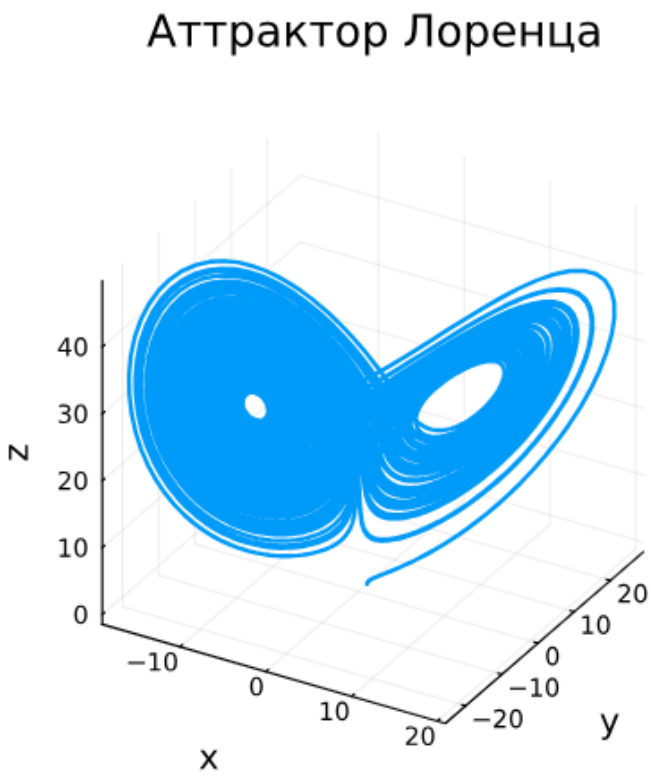

Рис. 3.6: Аттрактор Лоренца. График.

Можно отключить интерполяцию. (рис. 3.7)

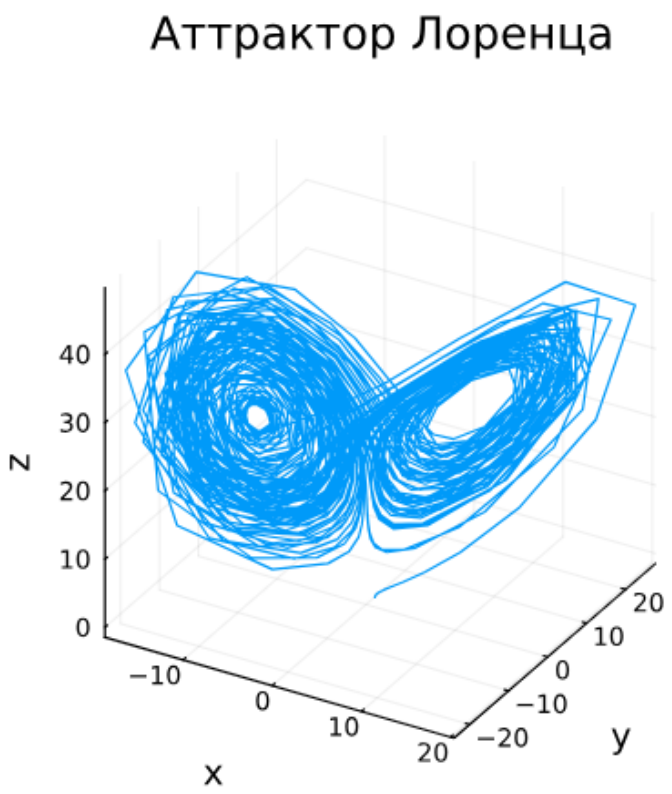

Рис. 3.7: Аттрактор Лоренца. График с отключенной интерполяцией.

3. Рассмотрела Модель Лотки–Вольтерры. Модель Лотки–Вольтерры описывает взаимодействие двух видов типа «хищник – жертва», где x - количество жертв, y — количество хищников, t — время, alfa, beta, gamma, delta — коэффициенты, отражающие взаимодействия между видами (в данном случае alfa — коэффициент рождаемости жертв, gamma — коэффициент убыли хищников, beta — коэффициент убыли жертв в результате взаимодействия с хищниками, delta — коэффициент роста численности хищников). Численное решение в Julia будет иметь следующий вид: (рис. 3.8) (рис. 3.9)

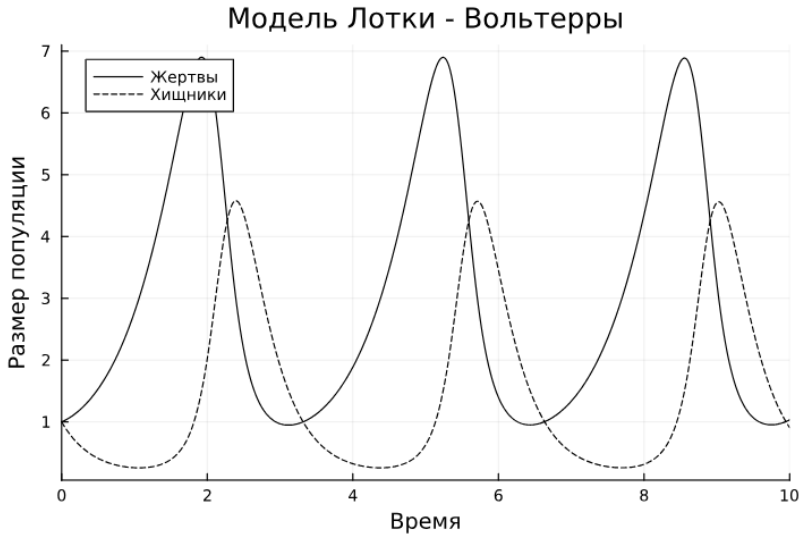

Рис. 3.8: Модель Лотки–Вольтерры: динамика изменения численности популяций

[15]:

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

[15]:

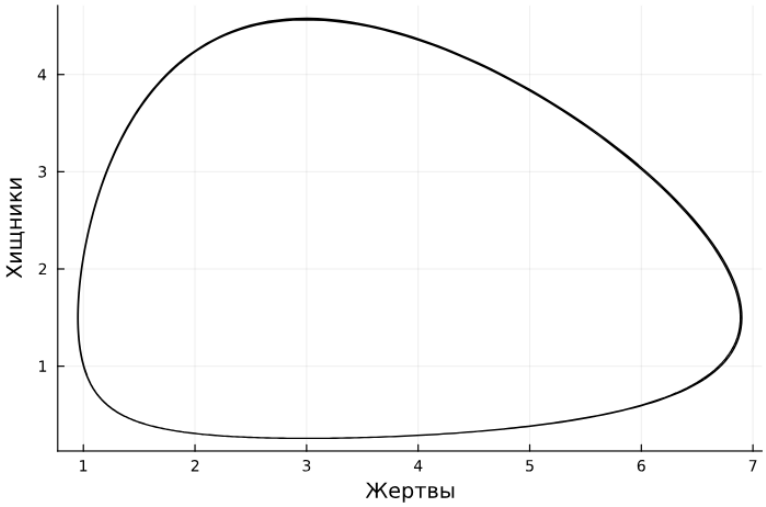

Рис. 3.9: Модель Лотки–Вольтерры: фазовый портрет

10. Приступила к заданиям для самостоятельной работы. Нумерация соответствует.

- Задание 1 (рис. 3.10) (рис. 3.11) (рис. 3.12) (рис. 3.13)

[45]:

```
# Параметры модели
b = 0.5   # Коэффициент рождаемости
c = 0.2   # Коэффициент смертности
a = b - c # Коэффициент роста популяции
x0 = 10.0 # Начальная численность популяции
tspan = (0.0, 10.0) # Временной диапазон

# Уравнение Мальтуса
function malthus!(dx, x, p, t)
    dx[1] = a * x[1]
end

# Решение задачи
prob = ODEProblem(malthus!, [x0], tspan)
sol = solve(prob, Tsit5(), saveat=0.1)

# Построение графика
plot(sol.t, sol[1, :], xlabel="Время", ylabel="Численность",
    title="Модель Мальтуса", lw=2)
```

Рис. 3.10: Задание 1. Код

[45]:

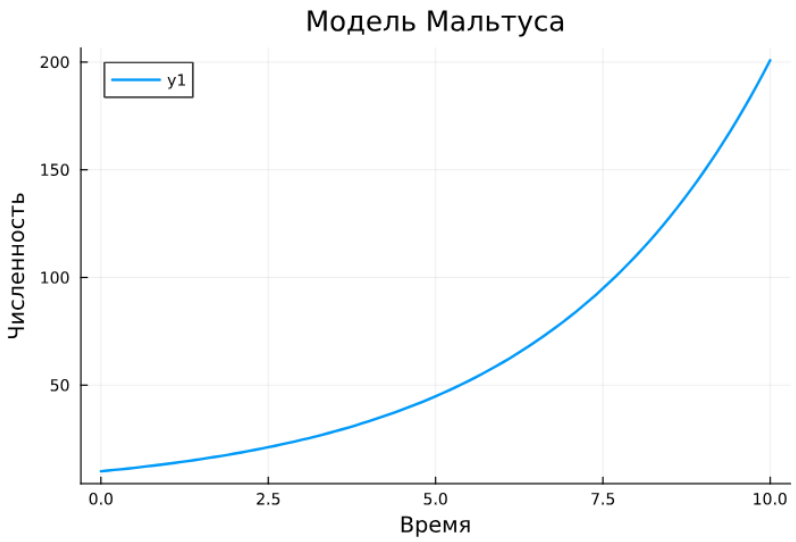

Рис. 3.11: Задание 1. График

[47]:

```
# Анимация
anim = @animate for t in 1:length(sol.t)
    plot(sol.t[1:t], sol[1, 1:t], xlabel="Время", ylabel="Численность",
         title="Модель Мальтуса", lw=2, legend=false, xlims=(0, tspan[2]),
        ylims=(0, maximum(sol[1, :])))
end
gif(anim, "malthus_model.gif", fps=15)
```

Рис. 3.12: Задание 1. Код анимации

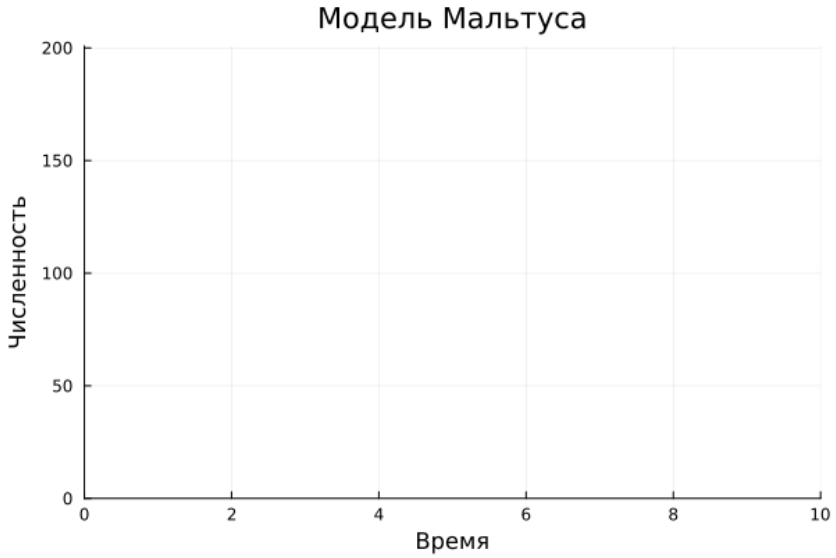

Рис. 3.13: Задание 1. Анимация

- Задание 2 (рис. 3.14) (рис. 3.15) (рис. 3.16) (рис. 3.17)

[49]:

```
# Параметры модели
r = 0.5    # Коэффициент роста популяции
k = 100.0  # Предельная ёмкость среды
x0 = 10.0  # Начальная численность популяции
tspan = (0.0, 20.0) # Временной диапазон

# Логистическое уравнение
function logistic!(dx, x, p, t)
    dx[1] = r * x[1] * (1 - x[1] / k)
end

# Решение задачи
prob = ODEProblem(logistic!, [x0], tspan)
sol = solve(prob, Tsit5(), saveat=0.1)

# Построение графика
plot(sol.t, sol[1, :], xlabel="Время", ylabel="Численность",
    title="Логистическая модель", lw=2)
```

Рис. 3.14: Задание 2. Код

[49]:

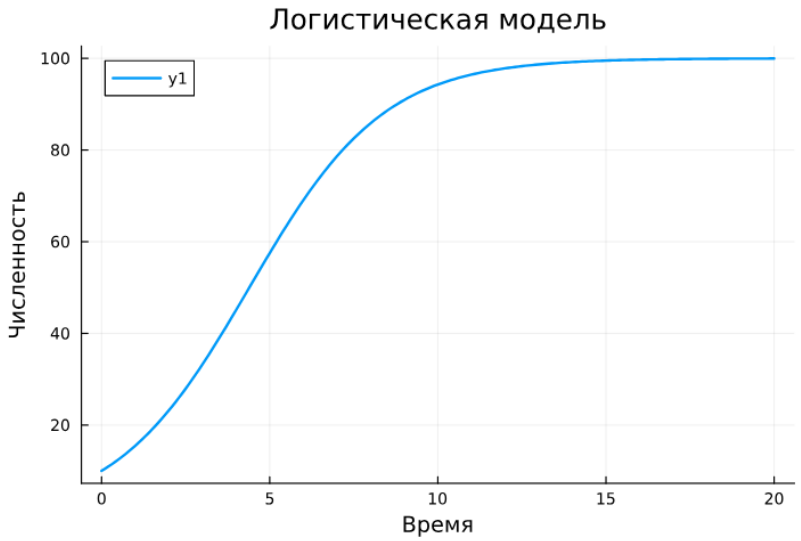

Рис. 3.15: Задание 2. График

[51]:

```
anim = @animate for t in 1:length(sol.t)
    plot(sol.t[1:t], sol[1, 1:t], xlabel="Время", ylabel="Численность",
         title="Логистическая модель", lw=2, legend=false, xlims=(0, tspan[2]),
        ylims=(0, maximum(sol[1, :])))
end
gif(anim, "logistic_model.gif", fps=15)
```

Рис. 3.16: Задание 2. Код анимации

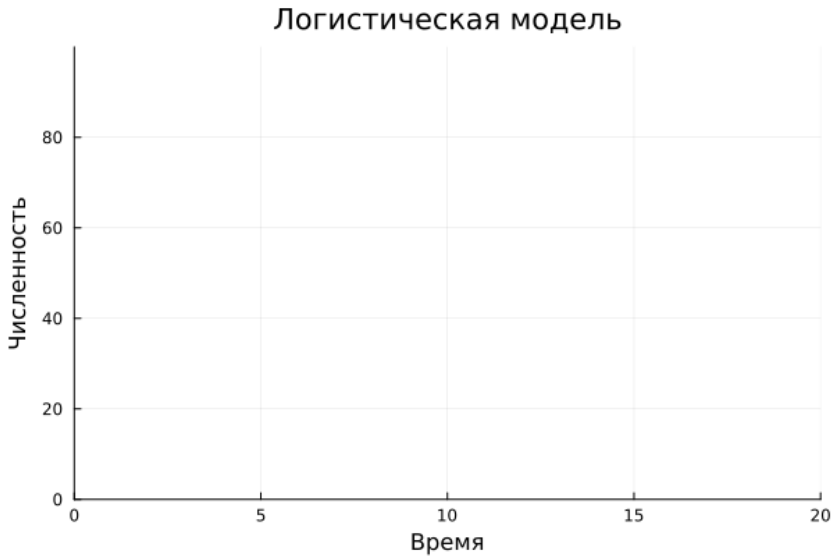

Рис. 3.17: Задание 2. Анимация

- Задание 3 (рис. 3.18) (рис. 3.19) (рис. 3.20) (рис. 3.21)

[53]:

```
# Параметры модели
β = 0.3    # Коэффициент инфицирования
v = 0.1    # Коэффициент выздоровления
s0 = 0.99  # Доля восприимчивых индивидов
i0 = 0.01  # Доля инфицированных индивидов
r0 = 0.0   # Доля выздоровевших индивидов
tspan = (0.0, 100.0) # Временной диапазон

# Уравнения SIR-модели
function sir!(du, u, p, t)
    s, i, r = u
    du[1] = -β * s * i      # ds/dt
    du[2] = β * s * i - v * i # di/dt
    du[3] = v * i           # dr/dt
end

# Решение задачи
u0 = [s0, i0, r0]
prob = ODEProblem(sir!, u0, tspan)
sol = solve(prob, Tsit5(), saveat=0.1)

# Построение графиков
plot(sol.t, sol[1, :], label="S (восприимчивые)", xlabel="Время",
    ylabel="Доля популяции", lw=2)
plot!(sol.t, sol[2, :], label="I (инфицированные)")
plot!(sol.t, sol[3, :], label="R (выздоровевшие)", title="SIR-модель эпидемии")
```

Рис. 3.18: Задание 3. Код

[53]:

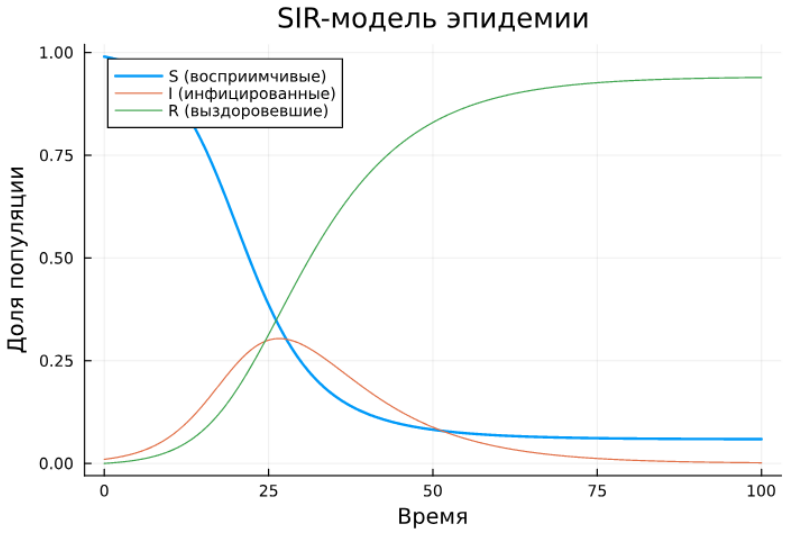

Рис. 3.19: Задание 3. График

[55]:

```
anim = @animate for t in 1:length(sol.t)
    plot(sol.t[1:t], sol[1, 1:t], label="S (восприимчивые)", xlabel="Время",
        ylabel="Доля популяции", lw=2, legend=false)
    plot!(sol.t[1:t], sol[2, 1:t], label="I (инфицированные)", legend=false)
    plot!(sol.t[1:t], sol[3, 1:t], label="R (выздоровевшие)",
        title="SIR-модель эпидемии", legend=false)
end
gif(anim, "sir_model.gif", fps=15)
```

Рис. 3.20: Задание 3. Код анимации

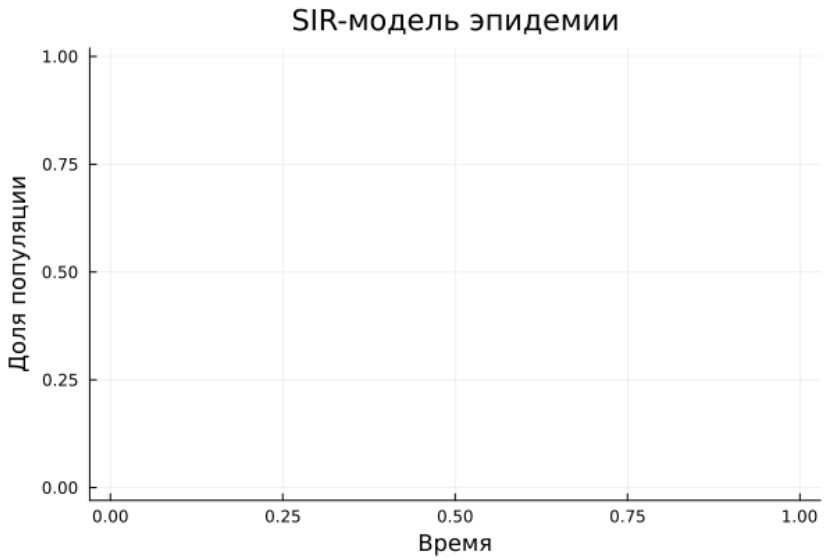

Рис. 3.21: Задание 3. Анимация

- Задание 4 (рис. 3.22) (рис. 3.23) (рис. 3.24) (рис. 3.25)

```julia
# Параметры модели
β = 0.3    # Коэффициент инфицирования
δ = 0.2    # Скорость перехода из E в I
γ = 0.1    # Коэффициент выздоровления
N = 1.0      # Размер популяции
s0 = 0.99    # Доля восприимчивых индивидов
e0 = 0.01    # Доля индивидов в латентной фазе
i0 = 0.0     # Доля инфицированных индивидов
r0 = 0.0     # Доля выздоровевших индивидов
tspan = (0.0, 100.0) # Временной диапазон

# Уравнения SEIR-модели
function seir!(du, u, p, t)
    s, e, i, r = u
    du[1] = -β * s * i / N       # ds/dt
    du[2] = β * s * i / N - δ * e # de/dt
    du[3] = δ * e - γ * i       # di/dt
    du[4] = γ * i                # dr/dt
end

# Решение задачи
u0 = [s0, e0, i0, r0]
prob = ODEProblem(seir!, u0, tspan)
sol = solve(prob, Tsit5(), saveat=0.1)

# Построение графиков
plot(sol.t, sol[1, :], label="S (восприимчивые)", xlabel="Время",
    ylabel="Доля популяции", lw=2)
plot!(sol.t, sol[2, :], label="E (латентные)")
plot!(sol.t, sol[3, :], label="I (инфицированные)")
plot!(sol.t, sol[4, :], label="R (выздоровевшие)",title="SEIR-модель эпидемии")
```

Рис. 3.22: Задание 4. Код

[57]:

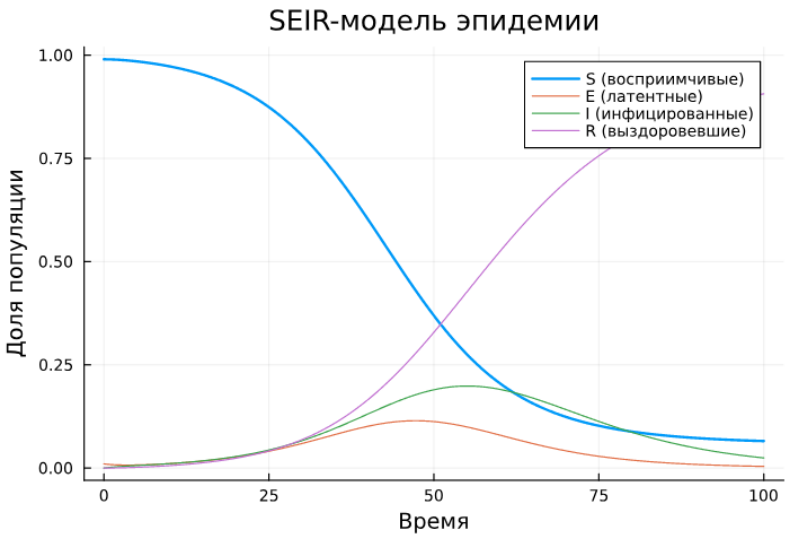

Рис. 3.23: Задание 4. График

[59]:

```
# Анимация
anim = @animate for t in 1:length(sol.t)
    plot(sol.t[1:t], sol[1, 1:t], label="S (восприимчивые)", xlabel="Время",
        ylabel="Доля популяции", lw=2, legend=false)
    plot!(sol.t[1:t], sol[2, 1:t], label="E (латентные)", legend=false)
    plot!(sol.t[1:t], sol[3, 1:t], label="I (инфицированные)", legend=false)
    plot!(sol.t[1:t], sol[4, 1:t], label="R (выздоровевшие)",
        title="SEIR-модель эпидемии", legend=false)
end
gif(anim, "seir_model.gif", fps=15)
```

Рис. 3.24: Задание 4. Код анимации

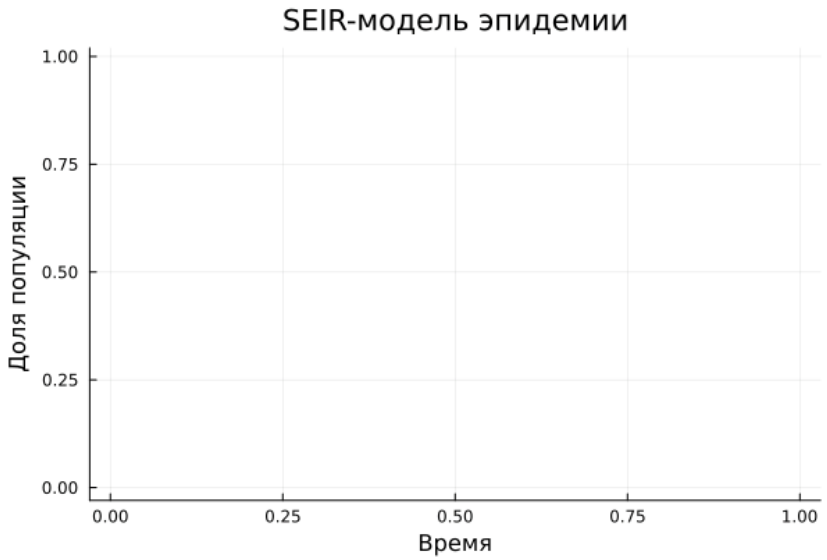

Рис. 3.25: Задание 4. Анимация

- Задание 5 (рис. 3.26) (рис. 3.27) (рис. 3.28)

```julia
# Параметры модели
a = 2.0  # Коэффициент роста популяции X1
c = 1.0  # Коэффициент убыли популяции X2
d = 5.0  # Влияние X1 на X2
# Начальные данные
X1_0 = 0.5   # Начальное значение X1
X2_0 = 0.5   # Начальное значение X2
t_max = 50   # Количество шагов моделирования

# Аналитическое решение для точки равновесия
X1_eq = c / d
X2_eq = a * X1_eq * (1 - X1_eq)
println("Аналитическое решение:")
println("Точка равновесия: X1_eq = $X1_eq, X2_eq = $X2_eq")

# Численное моделирование
X1 = [X1_0]
X2 = [X2_0]
for t in 1:t_max
    X1_next = a * X1[end] * (1 - X1[end]) - X1[end] * X2[end]
    X2_next = -c * X2[end] + d * X1[end] * X2[end]
    push!(X1, X1_next)
    push!(X2, X2_next)
end

# Построение графиков
plot(1:t_max+1, X1, label="X1", xlabel="Время (t)", ylabel="Популяция", lw=2,
    title="Динамика популяций")
plot!(1:t_max+1, X2, label="X2", lw=2)
```

```
Аналитическое решение:
Точка равновесия: X1_eq = 0.2, X2_eq = 0.32000000000000006
```

Рис. 3.26: Задание 5. Код

[69]:

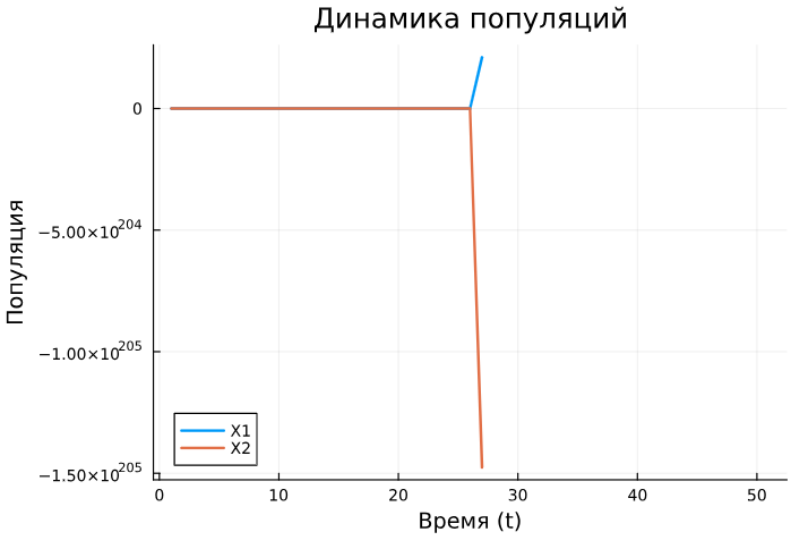

Рис. 3.27: Задание 5. График

[67]:

```
# 2. Фазовый портрет
plot(X1, X2, xlabel="X1", ylabel="X2", title="Фазовый портрет", legend=false, lw=
scatter!([X1_eq], [X2_eq], color=:red, label="Точка равновесия", ms=5)
```

[67]:

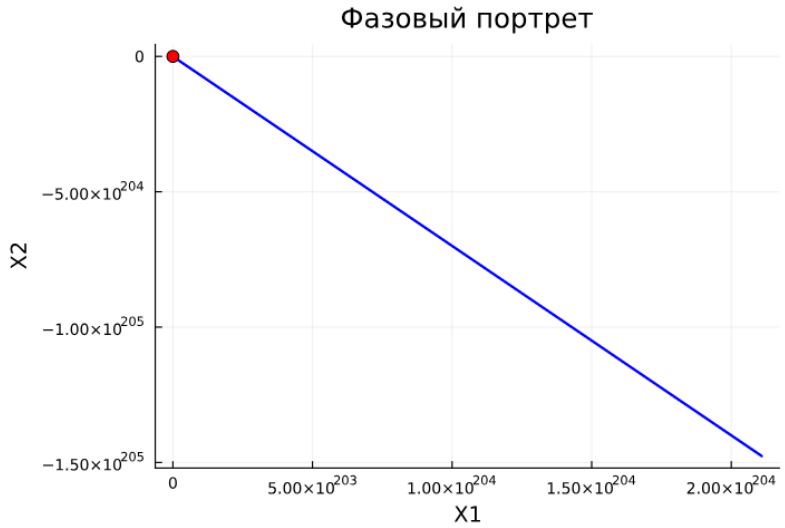

Рис. 3.28: Задание 5. Фазовый портрет

- Задание 6 (рис. 3.29) (рис. 3.30) (рис. 3.31) (рис. 3.32)

```
# Параметры модели
α = 1.0
β = 0.1
x0 = 10.0
y0 = 5.0
tspan = (0.0, 50.0)

# Уравнения модели
function competition!(du, u, p, t)
    x, y = u
    du[1] = α * x - β * x * y
    du[2] = α * y - β * x * y
end

# Решение задачи
u0 = [x0, y0]
prob = ODEProblem(competition!, u0, tspan)
sol = solve(prob, Tsit5(), saveat=0.1)

# Построение графика
plot(sol.t, sol[1, :], label="x (вид 1)", xlabel="Время", ylabel="Численность",
    lw=2)
plot!(sol.t, sol[2, :], label="y (вид 2)", title="Модель конкурентного отбора")
```

Рис. 3.29: Задание 6. Код

[71]:

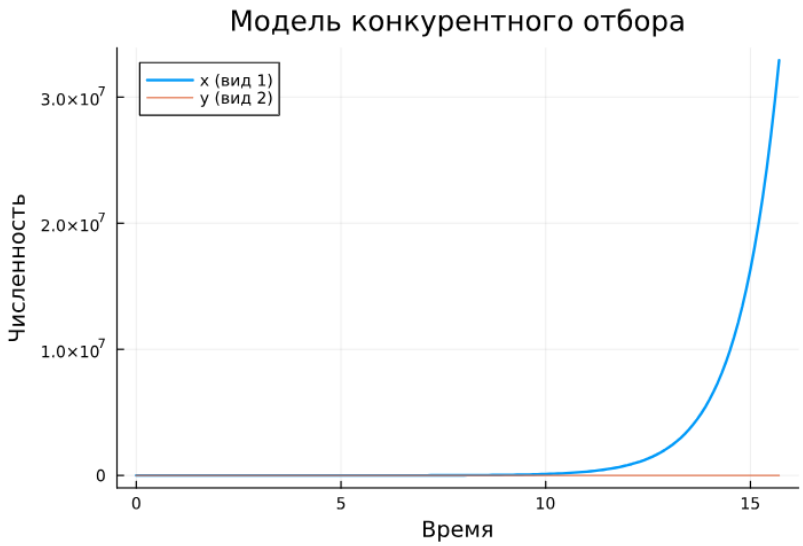

Рис. 3.30: Задание 6. График

[*]:

```
# Анимация
anim = @animate for t in 1:length(sol.t)
    plot(sol.t[1:t], sol[1, 1:t], label="x (вид 1)", xlabel="Время",
        ylabel="Численность", lw=2, legend=false)
    plot!(sol.t[1:t], sol[2, 1:t], label="y (вид 2)",
        title="Модель конкурентного отбора", legend=false)
end
gif(anim, "competition_model.gif", fps=15)
```

Рис. 3.31: Задание 6. Код анимации

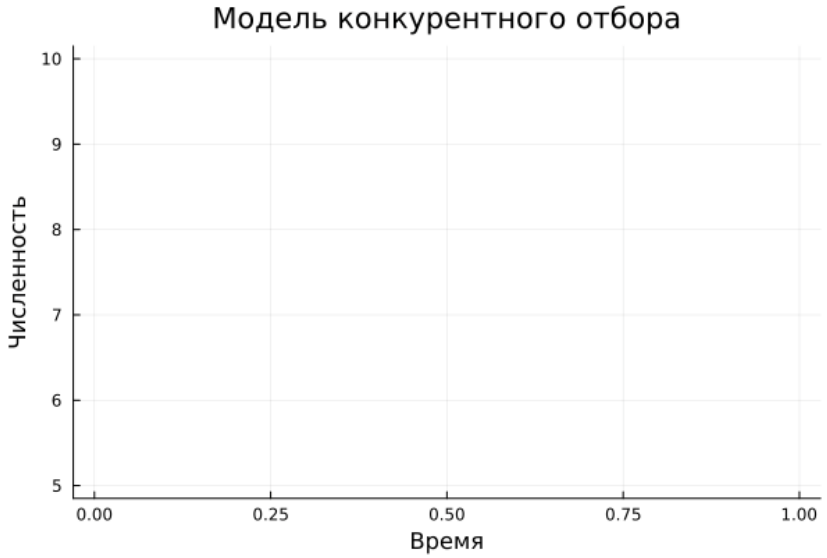

Рис. 3.32: Задание 6. Анимация

- Задание 7 (рис. 3.33) (рис. 3.34) (рис. 3.35) (рис. 3.36) (рис. 3.37) (рис. 3.38) (рис. 3.39)

[81]:

```
# Параметры
ω0 = 2.0  # Циклическая частота
x0 = 1.0  # Начальное отклонение
y0 = 0.0  # Начальная скорость
tspan = (0.0, 10.0)  # Временной диапазон

# Уравнение гармонического осциллятора
function harmonic_oscillator!(du, u, p, t)
    x, v = u  # u = [x, v], где x - координата, v - скорость
    du[1] = v  # dx/dt = v
    du[2] = -ω0^2 * x  # dv/dt = -ω0^2 * x
end

# Начальные условия
u0 = [x0, y0]

# Решение задачи
prob = ODEProblem(harmonic_oscillator!, u0, tspan)
sol = solve(prob, Tsit5(), saveat=0.1)

# Построение графиков
# 1. График x(t)
plot(sol.t, sol[1, :], xlabel="Время (t)", ylabel="x(t)",
    title="Консервативный гармонический осциллятор", lw=2)
```

Рис. 3.33: Задание 7. Код

[81]:

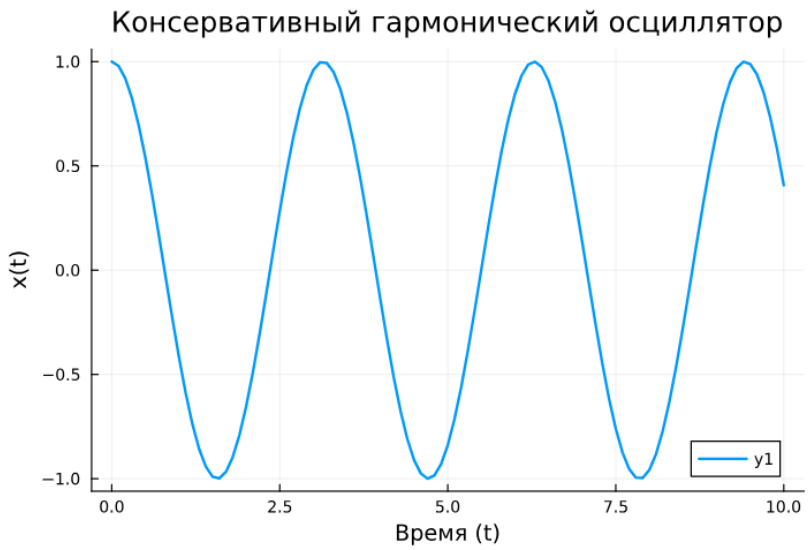

Рис. 3.34: Задание 7. График

[83]:

```
# 2. Фазовый портрет (x, v)
plot(sol[1, :], sol[2, :], xlabel="x", ylabel="v", title="Фазовый портрет", lw=2)
```

[83]:

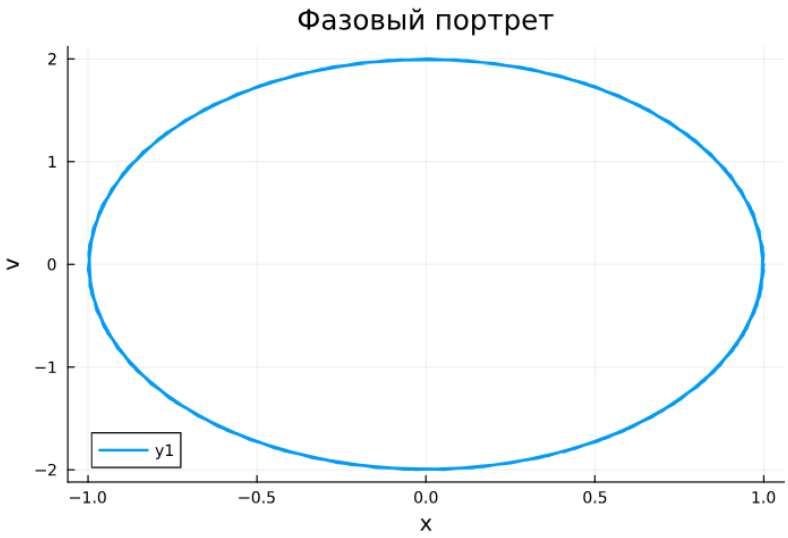

Рис. 3.35: Задание 7. Фазовый портрет

```
# Анимация графика x(t)
anim_xt = @animate for t in 1:length(sol.t)
    plot(sol.t[1:t], sol[1, 1:t], xlabel="Время (t)", ylabel="x(t)",
         title="Консервативный гармонический осциллятор", lw=2, legend=false, xl
end
gif(anim_xt, "harmonic_xt.gif", fps=15)
```

Рис. 3.36: Задание 7. Код анимация для графика

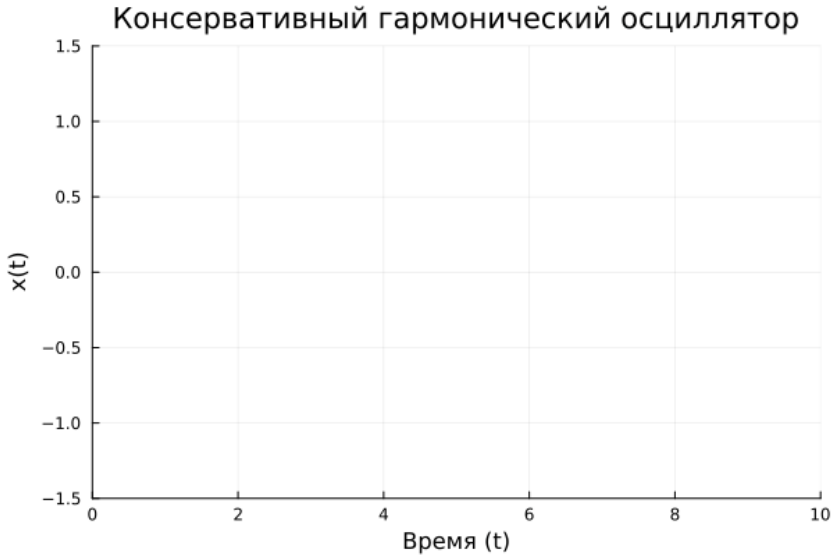

Рис. 3.37: Задание 7. Анимация графика

```
# Анимация фазового портрета
anim = @animate for t in 1:length(sol.t)
    plot(sol.t, sol[1, :], xlabel="Время (t)", ylabel="x(t)",
        title="Консервативный гармонический осциллятор", lw=2)
end
gif(anim, "harmonic_oscillator.gif", fps=15)
```

Рис. 3.38: Задание 7. Код анимация для фазового портрета

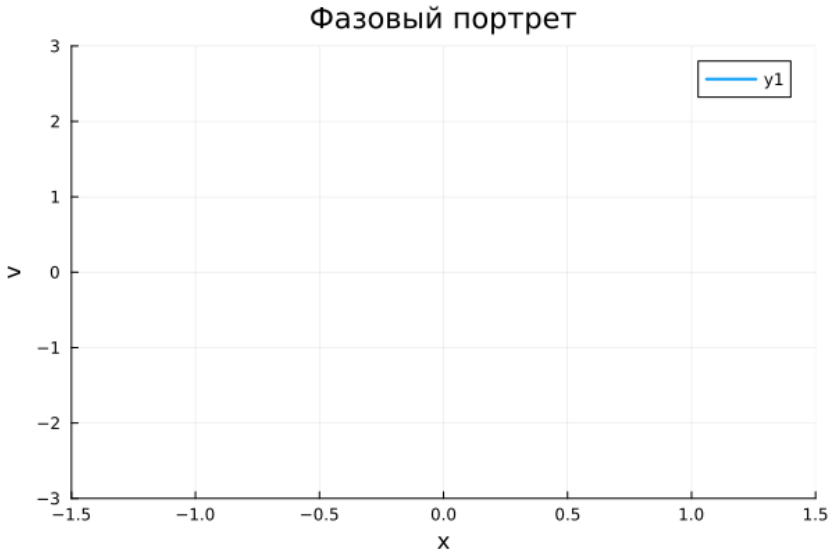

Рис. 3.39: Задание 7. Анимация фазового портрета

- Задание 8 (рис. 3.40) (рис. 3.41) (рис. 3.42) (рис. 3.43) (рис. 3.44) (рис. 3.45) (рис. 3.46)

[93]:

```
# Параметры
ω0 = 2.0  # Циклическая частота
γ = 0.2   # Коэффициент потерь
x0 = 1.0  # Начальное отклонение
y0 = 0.0  # Начальная скорость
tspan = (0.0, 10.0)  # Временной диапазон

# Уравнение гармонического осциллятора с демпфированием
function damped_oscillator!(du, u, p, t)
    x, v = u  # u = [x, v], где x - координата, v - скорость
    du[1] = v  # dx/dt = v
    du[2] = -2 * γ * v - ω0^2 * x  # dv/dt = -2γv - ω0^2 * x
end

# Начальные условия
u0 = [x0, y0]

# Решение задачи
prob = ODEProblem(damped_oscillator!, u0, tspan)
sol = solve(prob, Tsit5(), saveat=0.1)

# Построение графиков
# 1. График x(t)
plot(sol.t, sol[1, :], xlabel="Время (t)", ylabel="x(t)",
    title="Гармонический осциллятор с потерями энергии", lw=2)
```

Рис. 3.40: Задание 8. Код

[93]:

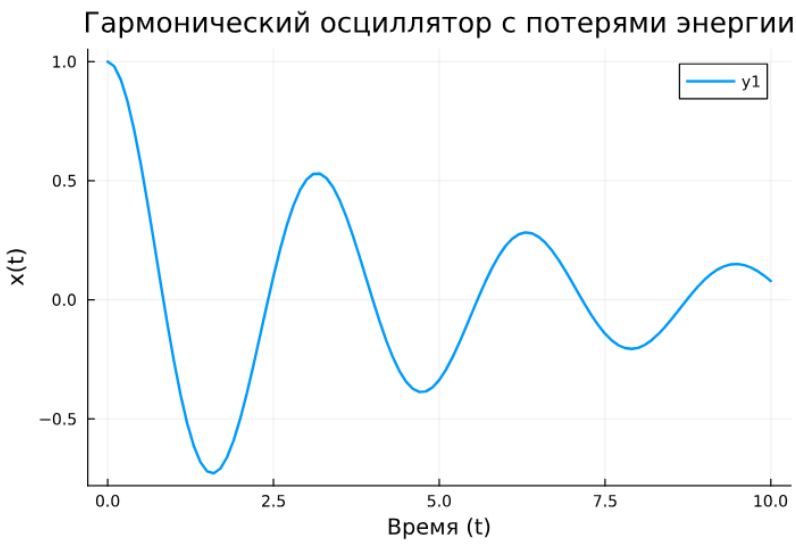

Рис. 3.41: Задание 8. График

[95]:

```
# 2. Фазовый портрет (x, v)
plot(sol[1, :], sol[2, :], xlabel="x", ylabel="v", title="Фазовый портрет",
    lw=2)
```

[95]:

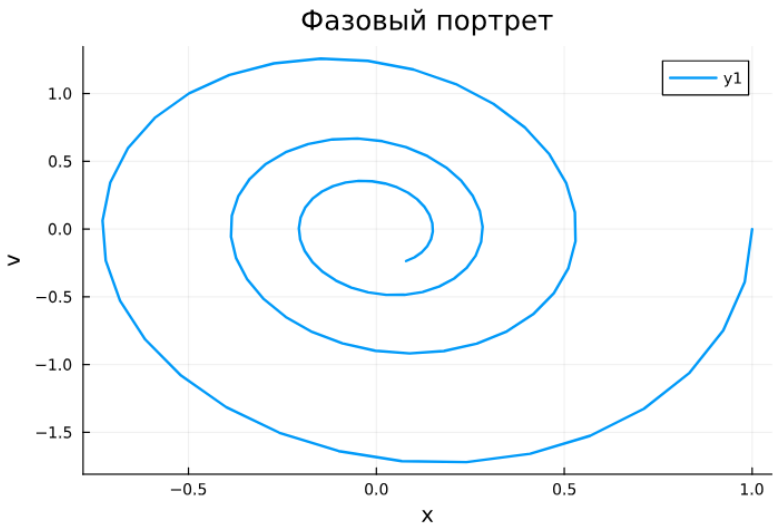

Рис. 3.42: Задание 8. Фазовый портрет

```
# Анимация графика x(t)
anim_xt = @animate for t in 1:length(sol.t)
    plot(sol.t[1:t], sol[1, 1:t], xlabel="Время (t)", ylabel="x(t)",
         title="Гармонический осциллятор с потерями энергии", lw=2, legend=false
end
gif(anim_xt, "damped_xt.gif", fps=15)
```

Рис. 3.43: Задание 8. Код анимация для графика

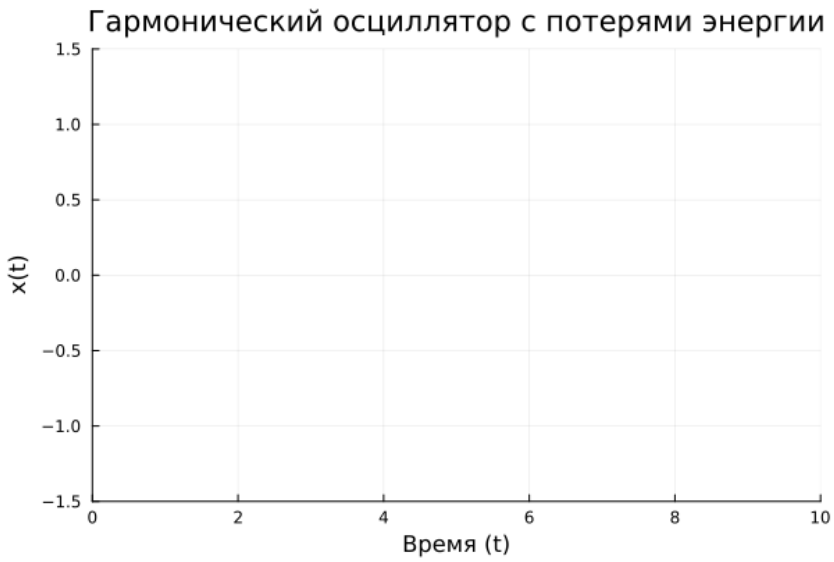

Рис. 3.44: Задание 8. Анимация графика

```
# Анимация фазового портрета
anim = @animate for t in 1:length(sol.t)
    plot(sol[1, 1:t], sol[2, 1:t], xlabel="x", ylabel="v", title="Фазовый портре
end
gif(anim, "damped_oscillator.gif", fps=15)
```

Рис. 3.45: Задание 8. Код анимация для фазового портрета

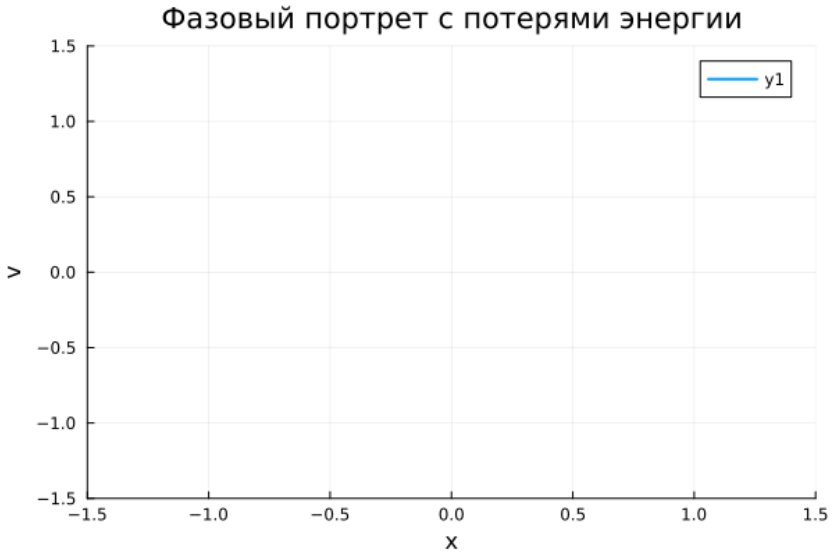

Рис. 3.46: Задание 8. Анимация фазового портрета

4 Вывод

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