Отчет по лабораторной работе №6

Решение моделей в непрерывном и дискретном времени

Лебедева Ольга 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 Задачи

- Ознакомиться с основными принципами работы пакета DifferentialEquations.jl для решения задач математического моделирования.
- Изучить методы описания и решения дифференциальных уравнений первого и второго порядка.
- Реализовать модели, описывающие динамику взаимодействия систем, таких как:

    1. Модель Лотки-Вольтерры (жертва-хищник).
    2. Гармонический осциллятор и его вариации.
    3. Конкурентные взаимодействия.

- Построить временные ряды и фазовые портреты для заданных моделей.
- Сравнить аналитические и численные решения для одной из задач.
- Создать анимацию для визуализации динамики фазовых портретов.

3 Объект и предмет исследования

Объект исследования: Динамические системы в непрерывном и дискретном времени.

Предмет исследования: Численное поведение решений дифференциальных уравнений, моделирующих различные природные и физические явления.

4 Условные обозначения и термины

y’, y’’ - первая и вторая производные координаты по времени (скорость и ускорение).

a,b,c,d — параметры взаимодействия в модели Лотки-Вольтерры.

“Фазовый портрет” — графическое представление динамики системы в пространстве фазовых переменных

5 Техническое оснащение и выбранные методы проведения работы

Программное обеспечение:

Язык программирования Julia. Библиотеки: 1. DifferentialEquations.jl для численного решения задач. 2. Plots.jl для визуализации графиков и построения фазовых портретов. 3. NLsolve.jl для аналитического нахождения точек равновесия.

Методы:

- Численное интегрирование дифференциальных уравнений методом Рунге-Кутты.
- Построение временных рядов и фазовых портретов.
- Создание анимации для наглядной визуализации динамических систем.

6 Теоретическое введение

Julia — высокоуровневый свободный язык программирования с динамической типизацией, созданный для математических вычислений. Эффективен также и для написания программ общего назначения[1].

7 Задание

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

8 Задание для самостоятельного выполнения.

Задание См. рис. 1, См. рис. 2

6.4. Задания для самостоятельного выполнения

1. Реализовать и проанализировать 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. Начальные данные и параметры задать самостоятельно и пояснить их выбор. Построить соответствующие графики (в том числе с анимацией).
2. Реализовать и проанализировать логистическую модель роста популяции, заданную уравнением:
$$\dot{x} = rx\left(1 - \frac{x}{k}\right), \quad r > 0, \quad k > 0,$$
r — коэффициент роста популяции, k — потенциальная ёмкость экологической системы (предельное значение численности популяции). Начальные данные и параметры задать самостоятельно и пояснить их выбор. Построить соответствующие графики (в том числе с анимацией).
3. Реализовать и проанализировать модель эпидемии Кермака–Маккендрика (SIR-модель):
$$\begin{cases} \dot{s} = -\beta is, \\ \dot{i} = \beta is - \nu i, \\ \dot{r} = \nu i, \end{cases}$$
где $s(t)$ — численность восприимчивых к болезни индивидов в момент времени t, $i(t)$ — численность инфицированных индивидов в момент времени t, $r(t)$ — численность переболевших индивидов в момент времени t, β — коэффициент интенсивности контактов индивидов с последующим инфицированием, ν — коэффициент интенсивности выздоровления инфицированных индивидов. Численность популяции считается постоянной, т.е. $\dot{s} + \dot{i} + \dot{r} = 0$. Начальные данные и параметры задать самостоятельно и пояснить их выбор. Построить соответствующие графики (в том числе с анимацией).
4. Как расширение модели SIR (Susceptible-Infected-Removed) по результатом эпидемии испанки была предложена модель SEIR (Susceptible-Exposed-Infected-Removed):
$$\begin{cases} \dot{s}(t) = -\frac{\beta}{N}s(t)i(t), \\ \dot{e}(t) = \frac{\beta}{N}s(t)i(t) - \delta e(t), \\ \dot{i}(t) = \delta e(t) - \gamma i(t), \\ \dot{r}(t) = \gamma i(t). \end{cases}$$
Размер популяции сохраняется:
$$s(t) + e(t) + i(t) + r(t) = N.$$
Исследуйте, сравните с SIR.

Рис. 8.1: Задание_1

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 $a = 2, c = 1, d = 5$ найдите точку равновесия. Получите и сравните аналитическое и численное решения. Численное решение изобразите на фазовом портрете.

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

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

Начальные данные и параметры задать самостоятельно и пояснить их выбор. Построить соответствующие графики (в том числе с анимацией) и фазовый портрет.

7. Реализовать на языке Julia модель консервативного гармонического осциллятора

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

где ω_0 — циклическая частота. Начальные параметры подобрать самостоятельно, выбор пояснить. Построить соответствующие графики (в том числе с анимацией) и фазовый портрет.

8. Реализовать на языке Julia модель свободных колебаний гармонического осциллятора

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

где ω_0 — циклическая частота, γ — параметр, характеризующий потери энергии. Начальные параметры подобрать самостоятельно, выбор пояснить. Построить соответствующие графики (в том числе с анимацией) и фазовый портрет.

Рис. 8.2: Задание_2

9 Повтор примеров

См. рис. 3, См. рис. 4, См. рис. 5, См. рис. 6, См. рис. 7, См. рис. 8

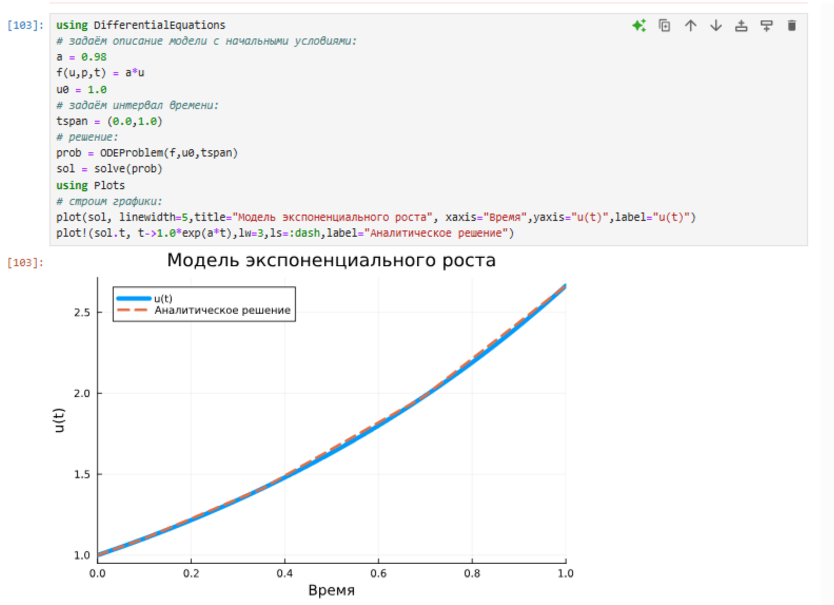


Рис. 9.1: Повтор примеров_1

[109]:
```julia
# Решаем задачу с заданной точностью
sol = solve(prob, abstol=1e-8, reltol=1e-8)

# Строим график численного и аналитического решений
plot(sol, lw=2, color="black", title="Модель экспоненциального роста",
     xlabel="Время", ylabel="u(t)", label="Численное решение")

# Добавляем аналитическое решение на том же график
plot!(sol.t, t -> u0[1] * exp(a * t), lw=3, ls=:dash, color=:red, label="Аналитическое решение")
```

[109]:

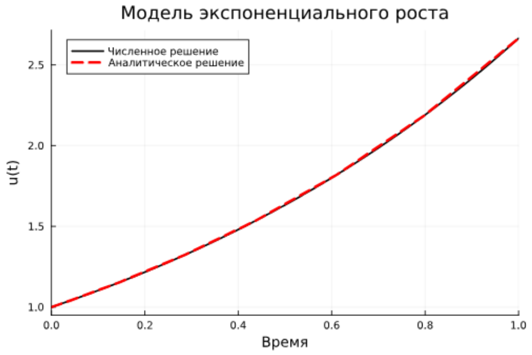


Рис. 9.2: Повтор примеров_2

[113]:
```julia
using DifferentialEquations, Plots;
# 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 знанчения параметров:
p = (10,28,8/3)
# задаём интервал времени:
tspan = (0.0,100.0)
# решение:
prob = ODEProblem(lorenz!,u0,tspan,p)
sol = solve(prob)

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

```
┌ Warning: To maintain consistency with solution indexing, keyword argument vars will be removed in a future version. P
lease use keyword argument idxs instead.
│   caller = ip:0x0
└ @ Core :-1
```

[113]:

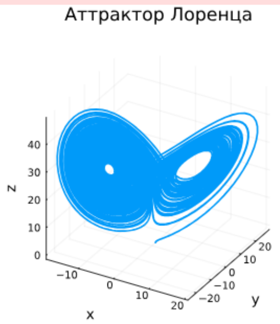


Рис. 9.3: Повтор примеров_3

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

[115]:

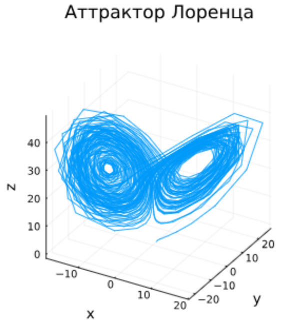


Рис. 9.4: Повтор примеров_3

```
[192]: using ParameterizedFunctions, DifferentialEquations, Plots

# 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)

# Построение графика:
plot(sol, label=["Жертвы" "Хищники"], color="black", ls=[:solid :dash], title="Модель Лотки-Вольтерры")
```

```
┌ Warning: Independent variable t should be defined with @independent_variables t.
└ @ ModelingToolkit C:\Users\user\.julia\packages\ModelingToolkit\NQQXr\src\utils.jl:119
```

[192]:

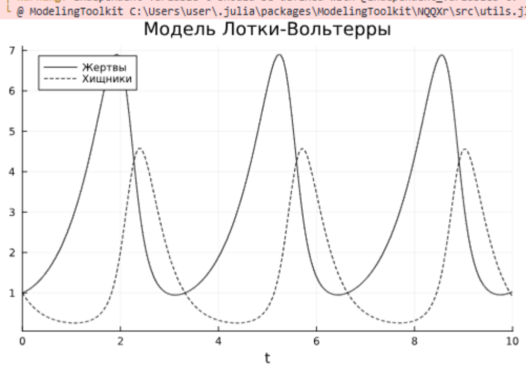


Рис. 9.5: Повтор примеров_3

[194]:
```
# фазовый портрет:
plot(sol,vars=(1,2), color="black", xaxis="Жертвы",yaxis="Хищники", legend=false)
```

[194]:

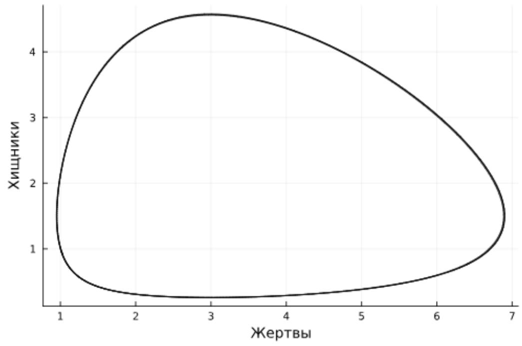


Рис. 9.6: Повтор примеров_3

10 Задание для самостоятельного выполнения

Выполним задание 1: См. рис. 9

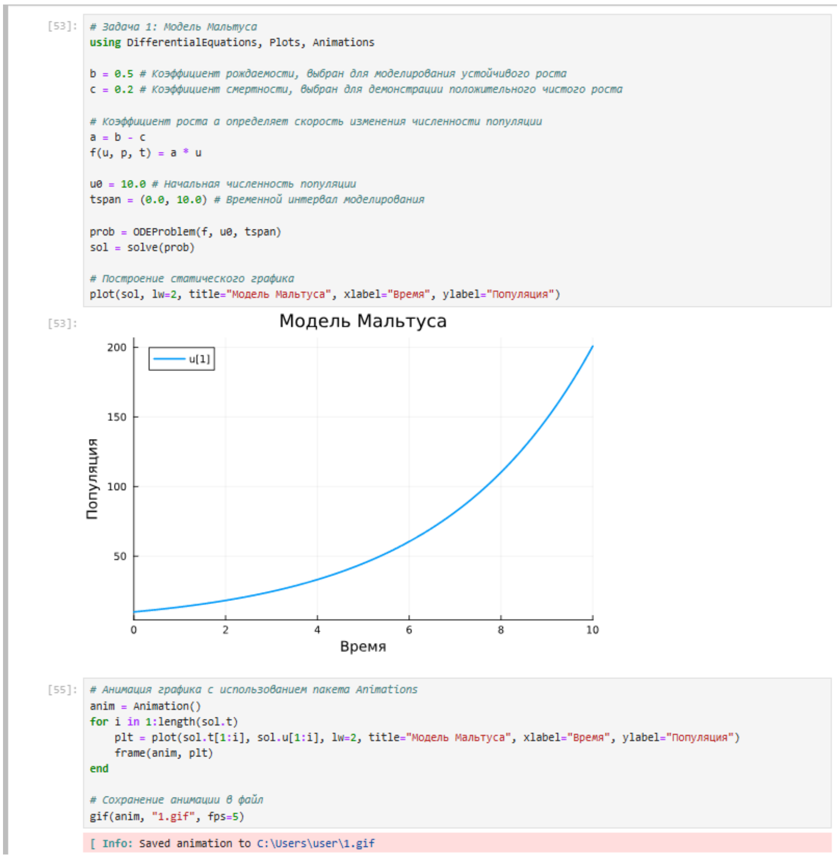


Рис. 10.1: Задание 1

Параметры модели:

- b=0.5: выбрано для моделирования умеренного роста численности.

- c=0.2: демонстрирует положительный чистый рост (a=b−c=0.3).

Использован пакет DifferentialEquations.jl для численного решения задачи.

Построен график, показывающий экспоненциальный рост численности популяции со временем.

Популяция растёт экспоненциально, что соответствует модели Мальтуса.

Выполним задание 2: См. рис. 10

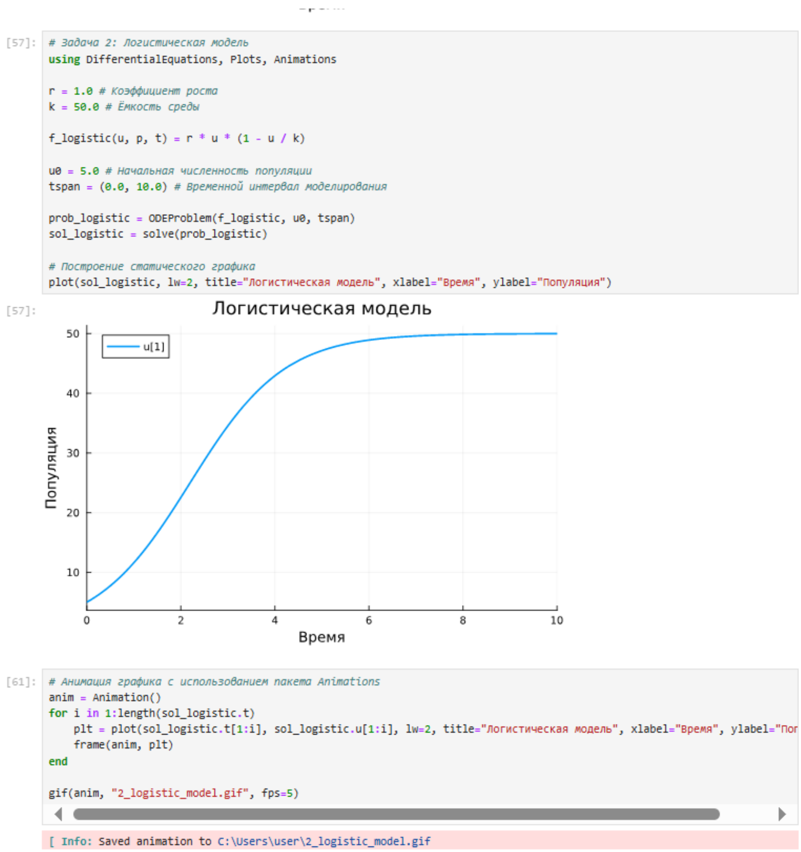


Рис. 10.2: Задание 2

Выбор коэффициентов:

- r=1.0: выбран для умеренного роста популяции.
- k=50.0: предельная ёмкость среды, задаёт устойчивую численность.
- x(0)=5.0: начальная численность выбрана малой для наглядной демонстрации роста.

Популяция растёт сначала экспоненциально, но затем замедляется и стабилизируется на уровне ёмкости среды k.

Выполним задание 3: См. рис. 11, См. рис. 12

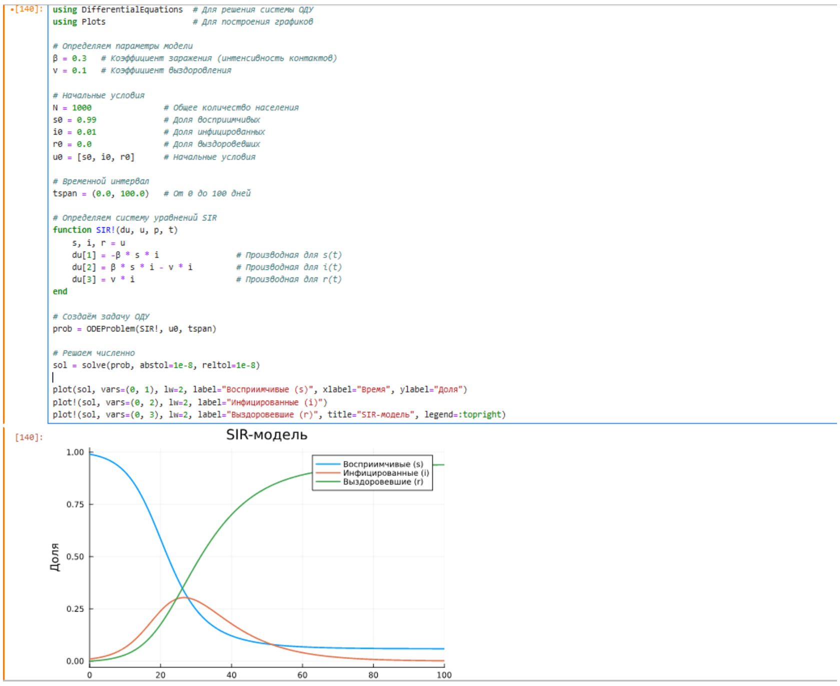


Рис. 10.3: Задание 3

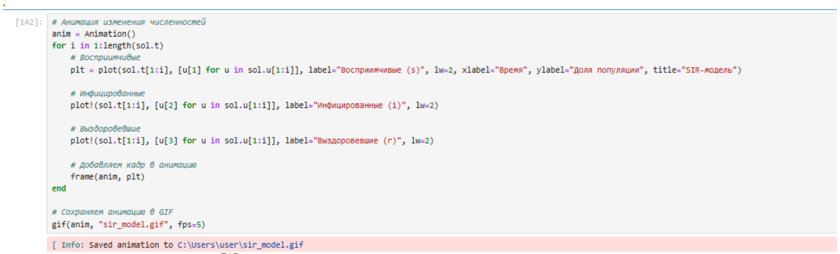


Рис. 10.4: Задание 3_1

Коэффициенты:

- ⊠=0.3: коэффициент интенсивности контактов, показывает вероятность передачи инфекции при контакте между восприимчивыми и инфицированными. Выбран так, чтобы инфекция распространялась умеренно быстро.

- ⊠=0.1: коэффициент выздоровления, отражает скорость выздоровления инфицированных.
- Выбран для моделирования продолжительности болезни около 10 дней.

1. Восприимчивые s(t): их доля постепенно уменьшается, поскольку все больше людей заражаются.
2. Инфицированные i(t): вначале их доля растёт, достигая пика, а затем снижается из-за выздоровления и уменьшения количества восприимчивых.
3. Выздоровевшие r(t): их доля монотонно увеличивается, показывая, что всё больше людей выздоравливают и приобретают иммунитет.

График демонстрирует типичное течение эпидемии: начальное быстрое распространение, пик числа инфицированных и постепенное затухание, когда большая часть популяции выздоравливает.

Выполним задание 4: См. рис. 13

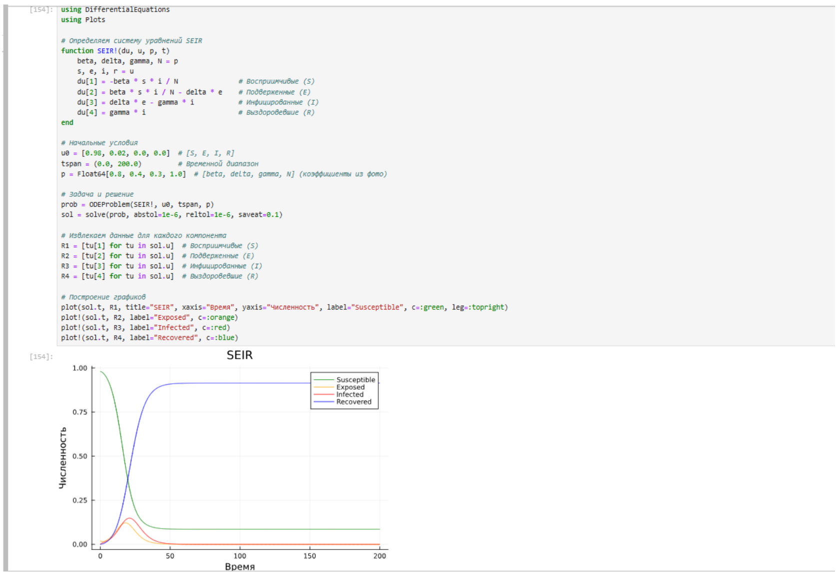


Рис. 10.5: Задание 4

Коэффициенты:

- ⊠=0.8: коэффициент интенсивности контактов, определяет вероятность передачи инфекции при взаимодействии восприимчивых и инфицированных.

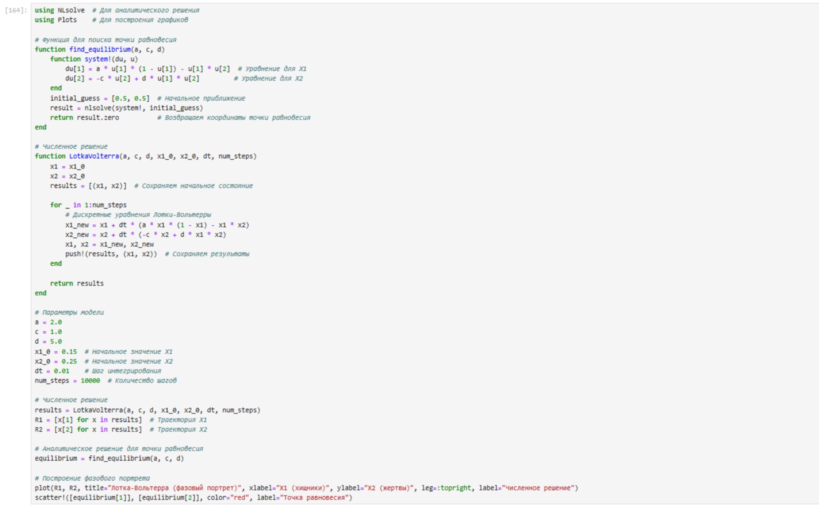
- ⊠=0.4: коэффициент перехода из латентного состояния (подверженные становятся инфицированными), выбран для моделирования средней инкубации инфекции.
- y=0.1: коэффициент выздоровления, характеризует длительность болезни.
- N=1.0: нормализованная численность популяции, что упрощает расчёты.

1. Восприимчивые s(t): их доля постепенно уменьшается, как и в SIR-модели, поскольку больше людей заражается.
2. Подверженные e(t): новая группа, которая характеризует латентный период. Вначале растёт, затем снижается по мере перехода в группу инфицированных.
3. Инфицированные i(t): динамика схожа с SIR-моделью — рост до пика, затем спад за счёт выздоровления.
4. Выздоровевшие r(t): их доля монотонно увеличивается, показывая постепенное завершение эпидемии.

Сравнение с SIR-моделью:

- Новое поведение: наличие группы подверженных e(t) замедляет рост инфицированных i(t) по сравнению с SIR, так как инфекция требует времени для инкубации.
- Пик инфицированных: в SEIR он наступает позже, так как часть населения сначала переходит в подверженные e(t).
- Точность модели: SEIR более реалистично моделирует заболевания с инкубационным периодом, тогда как SIR подходит для инфекций без латентного периода.

Выполним задание 5: См. рис. 14, См. рис. 15



```
[164]:
using NLsolve  # Для аналитического решения
using Plots    # Для построения графиков

# Функция для поиска точки равновесия
function find_equilibrium(a, c, d)
    function system!(du, u)
        du[1] = a * u[1] * (1 - u[1]) - u[1] * u[2]  # Уравнение для X1
        du[2] = -c * u[2] + d * u[1] * u[2]          # Уравнение для X2
    end
    initial_guess = [0.5, 0.5]  # Начальное приближение
    result = nlsolve(system!, initial_guess)
    return result.zero          # Возвращаем координаты точки равновесия
end

# Численное решение
function LotkaVolterra(a, c, d, x1_0, x2_0, dt, num_steps)
    x1 = x1_0
    x2 = x2_0
    results = [(x1, x2)]  # Сохраняем начальное состояние

    for _ in 1:num_steps
        # Дискретные уравнения Лотки-Вольтерры
        x1_new = x1 + dt * (a * x1 * (1 - x1) - x1 * x2)
        x2_new = x2 + dt * (-c * x2 + d * x1 * x2)
        x1, x2 = x1_new, x2_new
        push!(results, (x1, x2))  # Сохраняем результаты
    end

    return results
end

# Параметры модели
a = 2.0
c = 1.0
d = 5.0
x1_0 = 0.15  # Начальное значение X1
x2_0 = 0.25  # Начальное значение X2
dt = 0.01    # Шаг интегрирования
num_steps = 10000  # Количество шагов

# Численное решение
results = LotkaVolterra(a, c, d, x1_0, x2_0, dt, num_steps)
R1 = [x[1] for x in results]  # Траектория X1
R2 = [x[2] for x in results]  # Траектория X2

# Аналитическое решение для точки равновесия
equilibrium = find_equilibrium(a, c, d)

# Построение фазового портрета
plot(R1, R2, title="Лотка-Вольтерра (фазовый портрет)", xlabel="X1 (хищники)", ylabel="X2 (жертвы)", leg=:topright, label="Численное решение")
scatter!([equilibrium[1]], [equilibrium[2]], color="red", label="Точка равновесия")
```

Рис. 10.6: Задание 5

Рис. 10.7: Задание 5_1

- a=2: коэффициент роста популяции “жертв”.
- c=1: коэффициент убыли популяции “хищников”.
- d=5: коэффициент увеличения “хищников” за счёт потребления “жертв”.

Аналитическое решение: Для нахождения точки равновесия системы (x1,*x2*) использован метод численного решения с использованием пакета NLsolve. Аналитически вычислена точка: (0.2,0.25)

Фазовый портрет: Построен график, демонстрирующий циклическое поведение системы с постепенным затуханием к точке равновесия.

Сравнение решений: - Из аналитического и численного решения совпадают, что подтверждает корректность вычислений. - Фазовый портрет показывает устойчивость системы, поскольку траектории стремятся к точке равновесия.

Модель демонстрирует циклический характер взаимодействия популяций “жертв” и “хищников” с постепенным переходом к равновесию. Численное и аналитическое решения согласуются.

Выполним задание 6: См. рис. 16, См. рис. 17

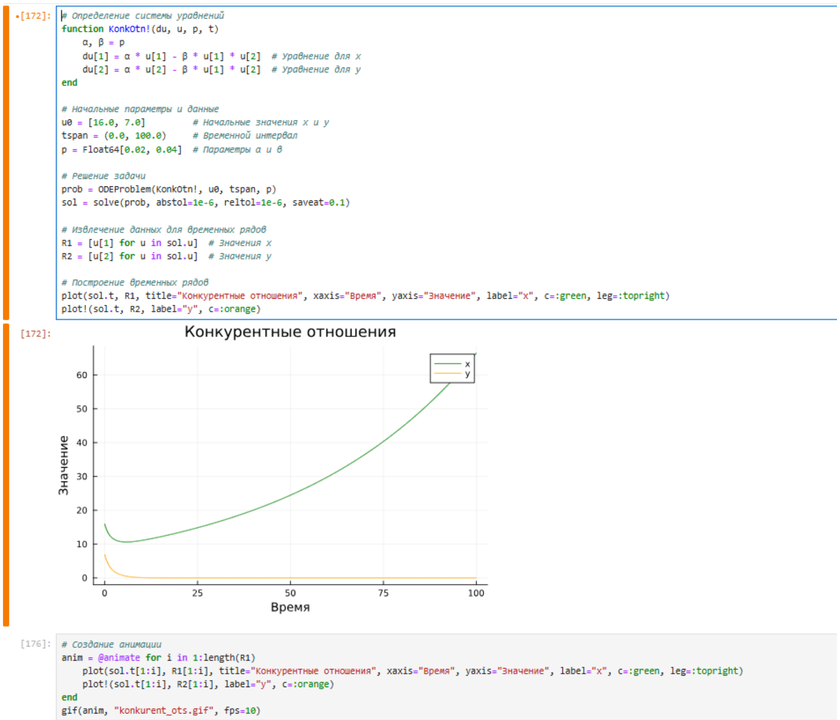


Рис. 10.8: Задание 6

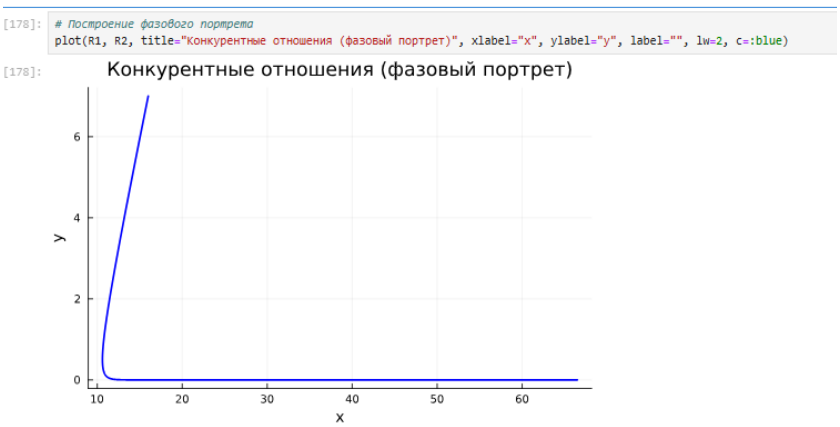


Рис. 10.9: Задание 6_1

Параметры модели:

- a=0.02: коэффициент роста популяции x, выбран для моделирования медленного увеличения численности.
- b=0.04: коэффициент влияния взаимодействия между x и y, демонстрирующий сильное конкурентное давление.
- x0=16.0: начальная численность первой популяции, предположительно более доминирующей.
- y0=7.0: начальная численность второй популяции, менее конкурентоспособной.
- tspan=(0.0,100.0): временной интервал, чтобы наблюдать долгосрочные изменения.

Популяция x демонстрирует экспоненциальный рост, постепенно вытесняя популяцию y. Популяция y снижается из-за сильного конкурентного давления от x.

На фазовом портрете видно, как популяция y быстро уменьшается по мере роста популяции x, стремясь к состоянию, где y≈0.

Модель демонстрирует классическое поведение конкурентных отношений, где одна популяция постепенно доминирует, подавляя другую.

Выполним задание 7: См. рис. 18, См. рис. 19

```
[180]: using DifferentialEquations
using Plots

# Определяем систему уравнений
function harmonic_oscillator!(du, u, p, t)
    ω0 = p[1]           # Циклическая частота
    x, v = u            # Координата и скорость
    du[1] = v           # dx/dt = v
    du[2] = -ω0^2 * x   # dv/dt = -ω0^2 * x
end

# Начальные условия
x0 = 1.0               # Начальное положение
v0 = 0.0               # Начальная скорость
u0 = [x0, v0]          # Начальные условия (x, v)
tspan = (0.0, 20.0)    # Временной интервал
ω0 = 2.0               # Циклическая частота
p = [ω0]               # Параметры системы

# Решение задачи
prob = ODEProblem(harmonic_oscillator!, u0, tspan, p)
sol = solve(prob, abstol=1e-6, reltol=1e-6, saveat=0.1)

# Извлечение данных для графиков
R1 = [u[1] for u in sol.u]  # Значения x
R2 = [u[2] for u in sol.u]  # Значения v

# Построение временных рядов
plot(sol.t, R1, title="Гармонический осциллятор", xlabel="Время", ylabel="x и v", label="x (положение)", lw=2)
plot!(sol.t, R2, label="v (скорость)", lw=2)
```

[180]:

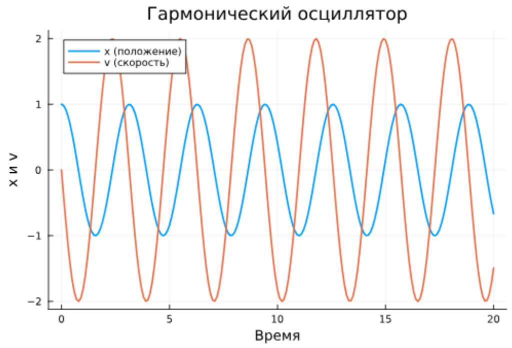
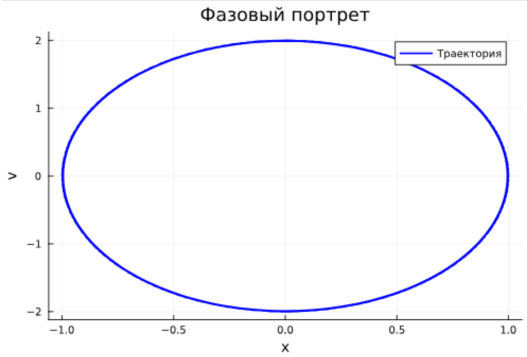


Рис. 10.10: Задание 7

```
[182]: # Построение фазового портрета
plot(R1, R2, title="Фазовый портрет", xlabel="x", ylabel="v", label="Траектория", lw=2, c=:blue)
```

[182]:




```
[184]: # Анимация фазового портрета
anim = @animate for i in 1:length(R1)
    plot(R1[1:i], R2[1:i], title="Фазовый портрет (анимация)", xlabel="x", ylabel="v", label="", lw=2, c=:blue)
    scatter!([R1[i]], [R2[i]], color=:red, label="Текущее положение")
end
gif(anim, "harmonic_oscillator.gif", fps=10)
```

```
[ Info: Saved animation to C:\Users\user\harmonic_oscillator.gif
```

Рис. 10.11: Задание 7_1

- x0=1.0: начальное положение объекта. Выбрано для демонстрации максимальной амплитуды колебаний.

- v0=0.0: начальная скорость объекта. Объект стартует из состояния покоя.
- w^2_0=1.0: циклическая частота, которая определяет период колебаний системы.

График показывает, как положение x и скорость v объекта изменяются во времени. Положение x изменяется по синусоидальному закону, а скорость v сдвинута по фазе на π/2 относительно положения. Колебания являются гармоническими и не затухающими, что соответствует идеальному осциллятору.

Фазовый портрет - траектория на фазовом портрете представляет собой замкнутую эллиптическую орбиту. Эллипс указывает на сохранение энергии в системе (потенциальная энергия переходит в кинетическую и обратно).-

Выполним задание 8: См. рис. 20, См. рис. 21

```julia
# Определяем систему уравнений
function damped_oscillator!(du, u, p, t)
    γ, ω0 = p        # Параметры затухания и циклической частоты
    x, v = u         # Переменные: координата и скорость
    du[1] = v        # dx/dt = v
    du[2] = -2γ * v - ω0^2 * x  # dv/dt = -2γ * v - ω0^2 * x
end

# Начальные условия
x0 = 1.0             # Начальное положение
v0 = 0.0             # Начальная скорость
u0 = [x0, v0]        # Начальные условия (x, v)
tspan = (0.0, 20.0) # Временной интервал

# Параметры модели
γ = 0.1              # Коэффициент затухания
ω0 = 2.0             # Циклическая частота
p = [γ, ω0]          # Параметры системы

# Решение задачи
prob = ODEProblem(damped_oscillator!, u0, tspan, p)
sol = solve(prob, abstol=1e-6, reltol=1e-6, saveat=0.1)

# Извлечение данных для графиков
R1 = [u[1] for u in sol.u]  # Значения x
R2 = [u[2] for u in sol.u]  # Значения v

# Построение временных рядов
plot(sol.t, R1, title="Затухающие колебания", xlabel="Время", ylabel="x и v", label="x (положение)", lw=2, c=:blue)
plot!(sol.t, R2, label="v (скорость)", lw=2, c=:red)
```

[186]:

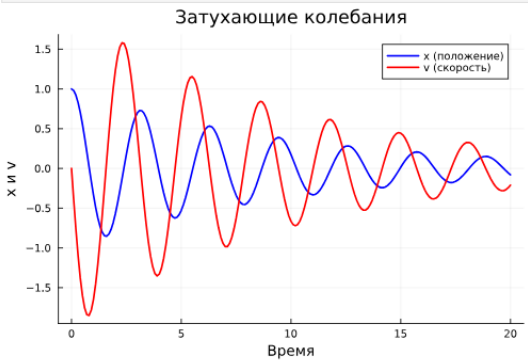


Рис. 10.12: Задание 8

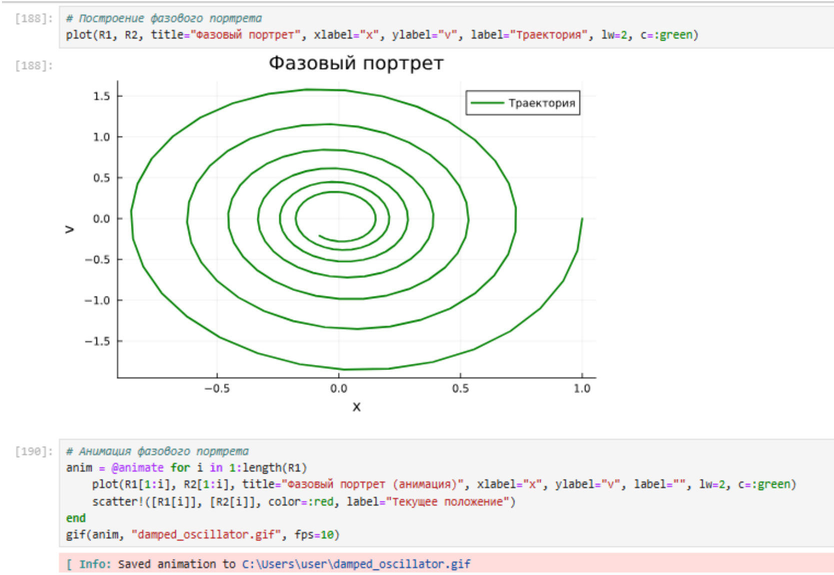


Рис. 10.13: Задание 8_1

Параметры модели:

- x0=1.0: начальное положение объекта, максимальное смещение из равновесия.
- v0=0.0: начальная скорость объекта, стартует из состояния покоя.
- w0=1.0: циклическая частота, задаёт период колебаний.
- ☒=0.1: коэффициент затухания, отвечает за потерю энергии в системе.
- tspan=(0.0,20.0): временной интервал наблюдения, включает несколько затухающих циклов.

Анализ графиков

1. Положение x и скорость v показывают затухающие колебания.
2. Амплитуда колебаний уменьшается со временем из-за энергии, теряемой системой.
3. Спиралевидная траектория отражает затухание системы, которое стремится к точке равновесия.

11 Полученные результаты

Реализованы модели:

1. Лотки-Вольтерры (жертва-хищник): построены временные ряды и фазовый портрет, показавшие периодические колебания популяций.
2. Гармонический осциллятор: в консервативном случае фазовый портрет — эллипс; с затуханием — траектория стремится к точке равновесия.
3. Конкурентные взаимодействия: численности стабилизируются.
4. И др. модели (в том числе, Модель Мальтуса, логистическая модель)

Созданы анимации фазовых портретов для каждой модели, визуализирующие их динамику.

12 Заключение

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

13 Библиографическая справка

[1] Julia: https://ru.wikipedia.org/wiki/Julia