Лабораторная работа №6

Компьютерный практикум по статистическому анализу данных

Николаев Дмитрий Иванович

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

2.1 Повторение примеров

Повторим примеры, представленные в лабораторной работе ([1]).

2.1.1 Решение обыкновенных дифференциальных уравнений

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

$$u(t) = f(u(t), p, t),$$

где $f(u(t), p, t)$ — нелинейная модель (функция) изменения $u(t)$ с заданным начальным значением $u(t_0) = u_0$, p — параметры модели, t — время.

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

2.1.1.1 Модель экспоненциального роста

Реализуем модель экспоненциального роста ([2.1-2.5]).

Лабораторная работа №6. Решение моделей в непрерывном и дискретном времени

Повторение примеров

6.2.1. Решение обыкновенных дифференциальных уравнений

Напомним, что обыкновенное дифференциальное уравнение (ОДУ) описывает изменение некоторой переменной $\boldsymbol{u}$:

$$u(t) = f(u(t), p, t),$$

где $f(u(t), p, t)$ — нелинейная модель (функция) изменения $u(t)$ с заданным начальным значением $u(t_0) = u_0$, p — параметры модели, t — время.

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

6.2.1.1. Модель экспоненциального роста

Рассмотрим пример использования этого пакета для решение уравнения модели экспоненциального роста, описываемую уравнением

$$u'(t) = au(t), \quad u(0) = u_0,$$

где a — коэффициент роста. Предположим, что заданы следующие начальные данные $a = 0.98$, $u(0) = 1.0$, $t \in [0; 1.0]$.

Аналитическое решение модели имеет вид:

$$u(t) = u_0 e^{at} u(t).$$

Численное решение в Julia будет иметь следующий вид:

```
[1]: # подключаем необходимые пакеты:
import Pkg
Pkg.add("DifferentialEquations")
```

```
    Updating registry at `C:\Users\User\.julia\registries\General.toml`
   Resolving package versions...
  No Changes to `C:\Users\User\.julia\environments\v1.8\Project.toml`
  No Changes to `C:\Users\User\.julia\environments\v1.8\Manifest.toml`
```

Рис. 2.1: Модель экспоненциального роста (1)

```
[2]: using DifferentialEquations
```

```
[ Info: Precompiling DifferentialEquations [0c46a032-eb83-5123-abaf-570d42b7fbaa]
┌ Warning: Replacing docs for `SciMLBase.sol :: Union{Tuple, Tuple{D}, Tuple{S}, Tuple{N}, Tuple{T}} where {T, N, S, D}` in module `SciMLBase`
└ @ Base.Docs docs\Docs.jl:240
```

```
[3]: # 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)
```

```
[3]: (0.0, 1.0)
```

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

```
[4]: retcode: Success
Interpolation: specialized 4th order "free" interpolation, specialized 2nd order "free" stiffness-aware interpolation
t: 5-element Vector{Float64}:
 0.0
 0.10042494449239292
 0.35218603951893646
 0.6934436334555072
 1.0
u: 5-element Vector{Float64}:
 1.0
 1.1034222047865465
 1.4121908848175448
 1.9730384867968267
 2.664456142481423
```

Рис. 2.2: Модель экспоненциального роста (2)

Построение графика, соответствующего полученному решению:

[5]:
```
# подключаем необходимые пакеты:
Pkg.add("Plots")
using Plots
```

```
   Resolving package versions...
  No Changes to `C:\Users\User\.julia\environments\v1.8\Project.toml`
  No Changes to `C:\Users\User\.julia\environments\v1.8\Manifest.toml`
```

[6]:
```
# строим графики:
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")
```

[6]:

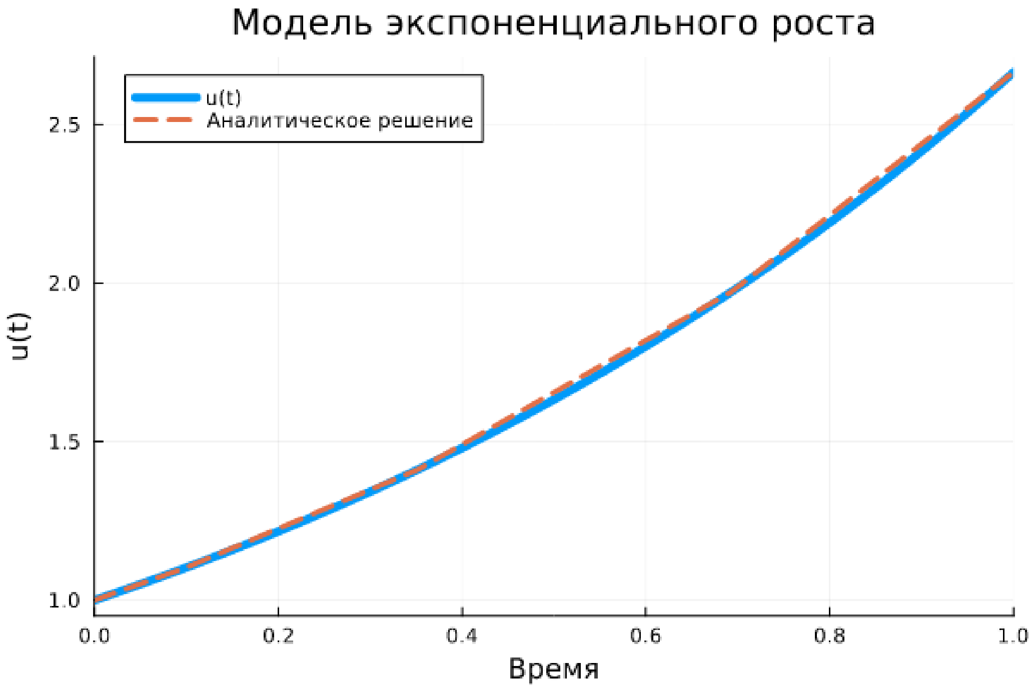


Рис. 2.3: Модель экспоненциального роста (3)

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

[7]:
```
# задаём точность решения:
sol = solve(prob,abstol=1e-8,reltol=1e-8)
println(sol)
```

```
ODESolution{Float64, 1, Vector{Float64}, Nothing, Nothing, Vector{Float64}, Vector{Vector{Float64}}, ODEProblem{Float64, Tuple{Float64, Float64}, false, SciMLBase.NullParameters, ODEFunction{false, SciMLBase.AutoSpecialize, typeof(f), LinearAlgebra.UniformScaling{Bool}, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, typeof(SciMLBase.DEFAULT_OBSERVED), Nothing, Nothing}, Base.Pairs{Symbol, Union{}, Tuple{}, NamedTuple{(), Tuple{}}}, SciMLBase.StandardODEProblem}, CompositeAlgorithm{Tuple{Vern7{typeof(OrdinaryDiffEq.trivial_limiter!), typeof(OrdinaryDiffEq.trivial_limiter!), Static.False}, Rodas4{0, false, Nothing, typeof(OrdinaryDiffEq.DEFAULT_PRECS), Val{:forward}, true, nothing}}, OrdinaryDiffEq.AutoSwitchCache{Vern7{typeof(OrdinaryDiffEq.trivial_limiter!), typeof(OrdinaryDiffEq.trivial_limiter!), Static.False}, Rodas4{0, false, Nothing, typeof(OrdinaryDiffEq.DEFAULT_PRECS), Val{:forward}, true, nothing}, Rational{Int64}, Int64}}, OrdinaryDiffEq.CompositeInterpolationData{ODEFunction{false, SciMLBase.AutoSpecialize, typeof(f), LinearAlgebra.UniformScaling{Bool}, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, Nothing, typeof(SciMLBase.DEFAULT_OBSERVED), Nothing, Nothing}, Vector{Float64}, Vector{Float64}, Vector{Vector{Float64}}, OrdinaryDiffEq.CompositeCache{Tuple{OrdinaryDiffEq.Vern7Co
```

Рис. 2.4: Модель экспоненциального роста (4)

```
Number of function 2 evaluations:                  0
Number of W matrix evaluations:                    0
Number of linear solves:                           0
Number of Jacobians created:                       0
Number of nonlinear solver iterations:             0
Number of nonlinear solver convergence failures:   0
Number of rootfind condition calls:                0
Number of accepted steps:                          8
Number of rejected steps:                          0
Maximum eigenvalue recorded:                       1, [1, 1, 1, 1, 1, 1, 1, 1, 1], SciMLBase.ReturnCode.Success)
```

[8]:
```
# 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")
```

[8]:

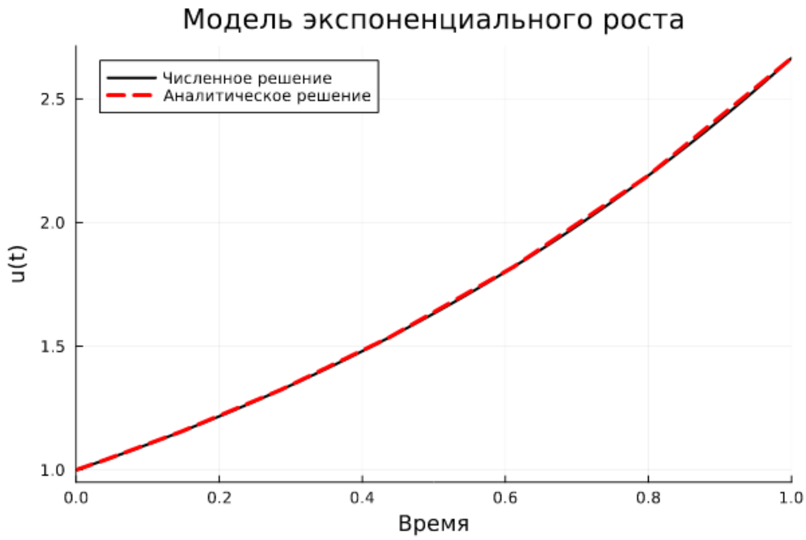


Рис. 2.5: Модель экспоненциального роста (5)

2.1.1.2 Система Лоренца

Реализуем динамическую систему Лоренца ([2.6-2.10]).

6.2.1.2. Система Лоренца

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

$$\begin{cases} \dot{x} = \sigma(y - x), \\ \dot{y} = \rho x - y - xz, \\ \dot{z} = xy - \beta z, \end{cases}$$

где σ, ρ и β — параметры системы (некоторые положительные числа, обычно указывают $\sigma = 10$, $\rho = 28$ и $\beta = \frac{8}{3}$).

Система 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:

```
[9]: # 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

[9]: lorenz! (generic function with 1 method)
```

Рис. 2.6: Система Лоренца (1)

```
[11]: # 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);

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

[12]: retcode: Success
Interpolation: specialized 4th order "free" interpolation, specialized 2nd order "free" stiffness-aware interpolation
t: 1263-element Vector{Float64}:
   0.0
   3.5678604836301404e-5
   0.0003924646531993154
   0.0032624077544510573
   0.009058075635317072
   0.01695646895607931
   0.02768995855685593
   0.04185635042021763
   0.06024041165841079
   0.08368541255159562
   0.11336499649094857
   0.1486218182609657
   0.18703978481550704
   ⋮
  99.05535949898116
  99.14118781914485
  99.22588252940076
  99.30760258626904
  99.39665422328268
  99.49536147459878
  99.58822928767293
  99.68983993598462
  99.77864535713971
  99.85744078539504
  99.93773320913628
 100.0
```

Рис. 2.7: Система Лоренца (2)

```
 -----
u: 1263-element Vector{Vector{Float64}}:
 [1.0, 0.0, 0.0]
 [0.9996434557625105, 0.0009988049817849058, 1.781434788799208e-8]
 [0.9961045497425811, 0.010965399721242457, 2.146955365838907e-6]
 [0.9693591634199452, 0.08977060667778931, 0.0001438018342266937]
 [0.9242043615038835, 0.24228912482984957, 0.0010461623302512404]
 [0.8800455868998046, 0.43873645009348244, 0.0034242593451028745]
 [0.8483309847495312, 0.6915629321083602, 0.008487624590227805]
 [0.8495036669651213, 1.0145426355349096, 0.01821208962127994]
 [0.9139069574560097, 1.4425599806525806, 0.03669382197085303]
 [1.088863826836895, 2.052326595543049, 0.0740257368585531]
 [1.4608627354936607, 3.0206721193016133, 0.16003937020467585]
 [2.162723488309695, 4.633363843843712, 0.37711740539408584]
 [3.3684644104189387, 7.26769410983553, 0.936355641713984]
 ⋮
 [12.265454131109882, 12.598146409807255, 31.546057337607913]
 [10.48677626670755, 6.494631680470132, 33.669742813875764]
 [6.893277189568002, 3.1027383340030155, 29.77818388970318]
 [4.669609096878053, 3.061564434452441, 25.1424735017959]
 [4.188801916573263, 4.617474401440693, 21.09864175382292]
 [5.559603854699961, 7.905631612648314, 18.79323210016923]
 [8.556629716266505, 12.533041060088328, 20.6623639692711]
 [12.280585075547771, 14.505154761545633, 29.332088452699942]
 [11.736883151600804, 8.279294641640229, 34.68007510231878]
 [8.10973327066804, 3.2495066495235854, 31.97052076740117]
 [4.958629886040755, 2.194919965065022, 26.948439650907677]
 [3.8020065515435855, 2.787021797920187, 23.420567509786622]
```

Рис. 2.8: Система Лоренца (3)

Фазовый портрет:

[14]:
```
# строим график:
plot(sol, idxs=(1,2,3), lw=2, title="Аттрактор Лоренца",
    xaxis="x",yaxis="y", zaxis="z",legend=false)
```

[14]:

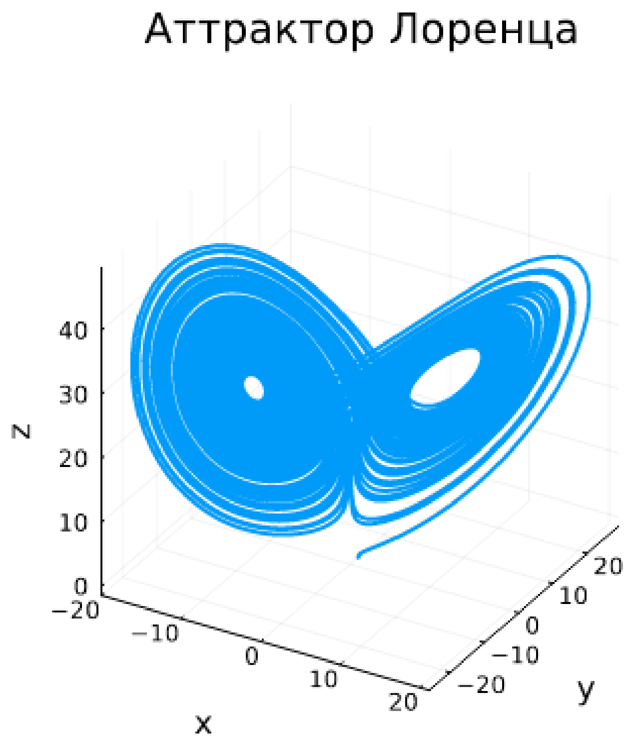


Рис. 2.9: Система Лоренца (4)

Можно отключить интерполяцию:

[15]:
```
# отключаем интерполяцию:
plot(sol,vars=(1,2,3),denseplot=false, lw=1, title="Аттрактор Лоренца",
    xaxis="x",yaxis="y", zaxis="z",legend=false)
```

[15]:

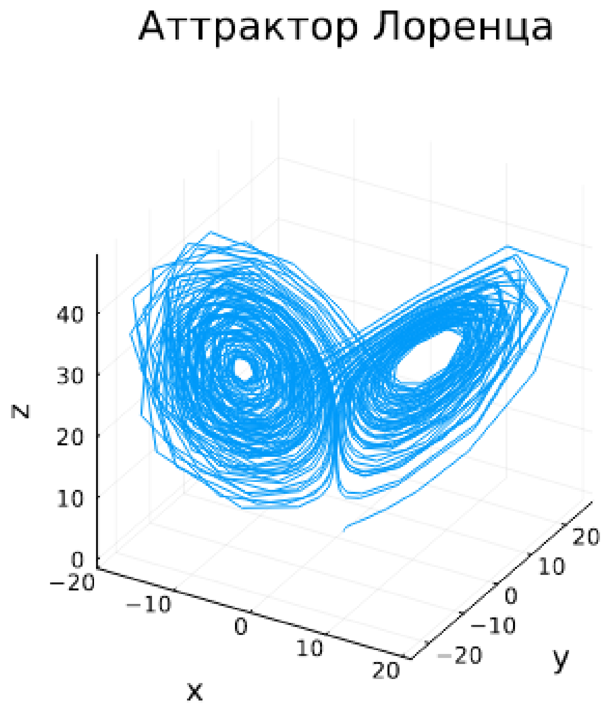


Рис. 2.10: Система Лоренца (5)

2.1.2 Модель Лотки–Вольтерры

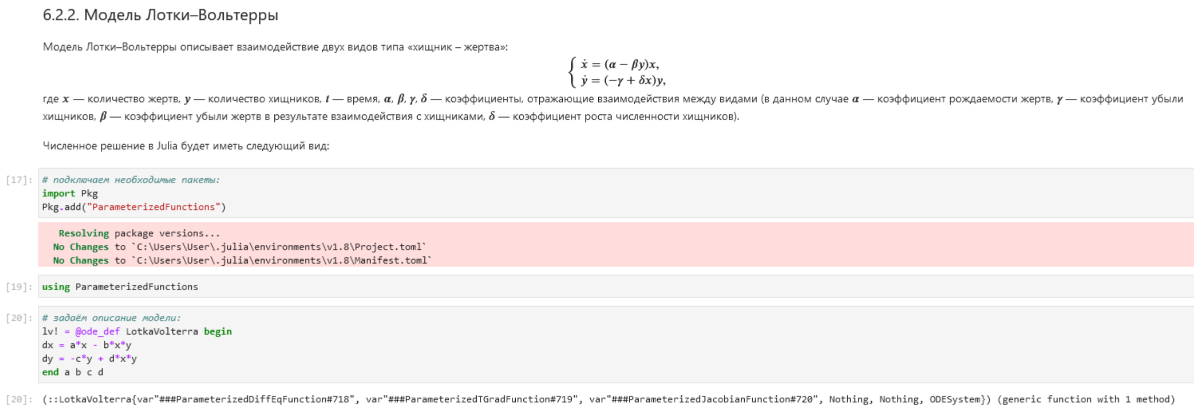
Модель Лотки–Вольтерры описывает взаимодействие двух видов типа «хищник – жертва»:

$$\begin{cases} \dot{x} = (\alpha - \beta y)x, \\ \dot{y} = (-\gamma + \delta x)y, \end{cases}$$

где x — количество жертв, y — количество хищников, t — время, α, β, γ, δ — коэффициенты, отражающие взаимодействия между видами (в данном случае

α — коэффициент рождаемости жертв, γ — коэффициент убыли хищников, β — коэффициент убыли жертв в результате взаимодействия с хищниками, δ — коэффициент роста численности хищников).

Реализуем модель Лотки-Вольтерры ([2.11-2.15]).


6.2.2. Модель Лотки–Вольтерры

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

$$\begin{cases} \dot{x} = (\alpha - \beta y)x, \\ \dot{y} = (-\gamma + \delta x)y, \end{cases}$$

где x — количество жертв, y — количество хищников, t — время, α, β, γ, δ — коэффициенты, отражающие взаимодействия между видами (в данном случае α — коэффициент рождаемости жертв, γ — коэффициент убыли хищников, β — коэффициент убыли жертв в результате взаимодействия с хищниками, δ — коэффициент роста численности хищников).

Численное решение в Julia будет иметь следующий вид:

[17]:
```
# подключаем необходимые пакеты:
import Pkg
Pkg.add("ParameterizedFunctions")
```

```
  Resolving package versions...
 No Changes to `C:\Users\User\.julia\environments\v1.8\Project.toml`
 No Changes to `C:\Users\User\.julia\environments\v1.8\Manifest.toml`
```

[19]:
```
using ParameterizedFunctions
```

[20]:
```
# 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
```

[20]:
```
(::LotkaVolterra{var"###ParameterizedDiffEqFunction#718", var"###ParameterizedTGradFunction#719", var"###ParameterizedJacobianFunction#720", Nothing, Nothing, ODESystem}) (generic function with 1 method)
```

Рис. 2.11: Модель Лотки–Вольтерры (1)

```
[23]: # 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);
```

```
[24]: # решение:
prob = ODEProblem(lv!,u0,tspan,p)
sol = solve(prob)
```

```
[24]: retcode: Success
Interpolation: specialized 4th order "free" interpolation, specialized 2nd order "free" stiffness-aware interpolation
t: 34-element Vector{Float64}:
  0.0
  0.0776084743154256
  0.23264513699277584
  0.4291185174543143
  0.6790821987497083
  0.9444046158046306
  1.2674601546021105
  1.6192913303893046
  1.9869754428624007
  2.2640902393538296
  2.5125484290863063
  2.7468280298123062
  3.0380065851974147
  ⋮
  6.455762090996754
  6.780496138817711
  7.171040059920871
  7.584863345264154
  7.978068981329682
  8.48316543760351
  8.719248247740158
  8.949206788834692
  9.200185054623292
  9.438029017301554
  9.711808134779586
 10.0
```

Рис. 2.12: Модель Лотки–Вольтерры (2)

```
u: 34-element Vector{Vector{Float64}}:
 [1.0, 1.0]
 [1.0454942346944578, 0.8576684823217127]
 [1.1758715885138267, 0.639459570317544]
 [1.4196809607170826, 0.4569962601282084]
 [1.876719395008001, 0.3247334292791l314]
 [2.5882500645533466, 0.26336255535952163]
 [3.8607089092207665, 0.2794458098285253]
 [5.750812667710396, 0.5220072537934558]
 [6.814978999130169, 1.917782632839066]
 [4.3929992925714245, 4.194670792850584]
 [2.1008562663496626, 4.31694049248469]
 [1.2422757654297396, 3.1073646247560807]
 [0.9582720921023357, 1.7661433892230374]
 ⋮
 [0.952206525526163, 1.4383448433913901]
 [1.1004623776276266, 0.7526620730760382]
 [1.5991134291557523, 0.3903181675223147]
 [2.614253967788294, 0.26416945387525886]
 [4.241076127191749, 0.3051236762921916]
 [6.791123785297795, 1.134528779714611з]
 [6.265370675764892, 2.74169350754023]
 [3.7807651118880545, 4.431165685863461]
 [1.816420140681761, 4.064056625315978]
 [1.146502140769072S, 2.7911706616216976]
 [0.9557986135403302, 1.6235622951850799]
 [1.0337581256020607, 0.9063703842886133]
```

Рис. 2.13: Модель Лотки–Вольтерры (3)

```
[26]: plot(sol, label = ["Жертвы" "Хищники"], color="black", ls=[:solid :dash], title="Модель Лотки - Вольтерры",
          xaxis="Время",yaxis="Размер популяции")
```

[26]:

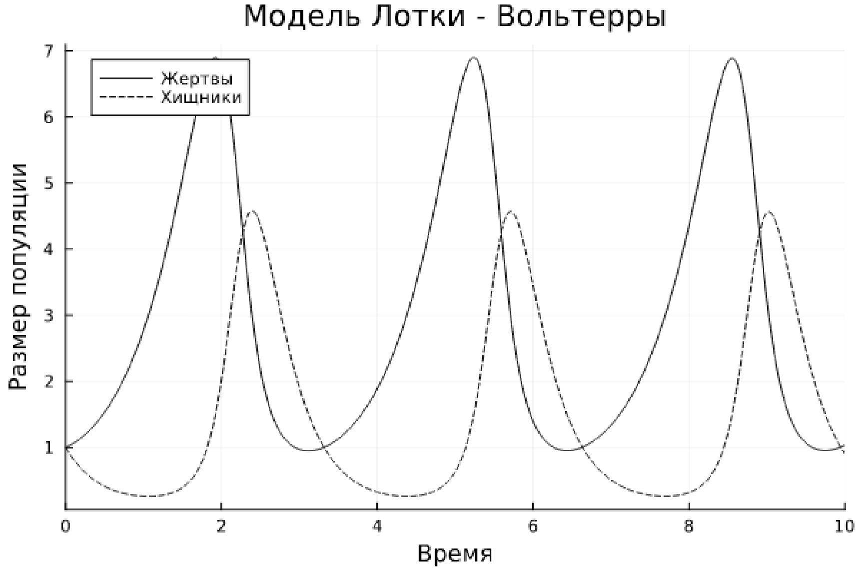


Рис. 2.14: Модель Лотки–Вольтерры (4)

Фазовый портрет:

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

[28]:

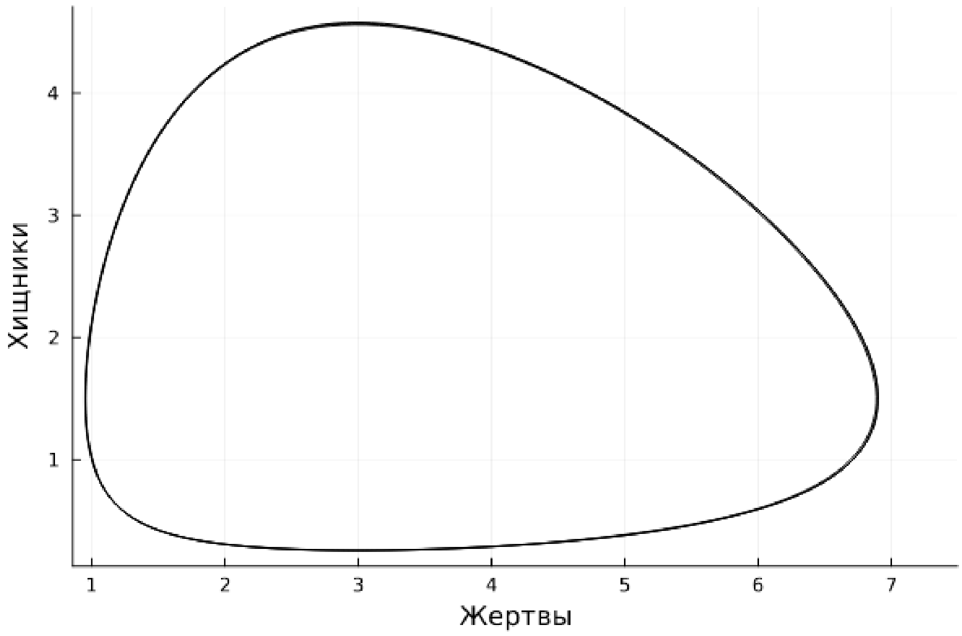


Рис. 2.15: Модель Лотки–Вольтерры (5)

2.2 Самостоятельная работа

1. Реализуем и проанализируем модель роста численности изолированной популяции — модель Мальтуса ([2.16-2.19]) с анимацией ([**gif:001?**]).

Самостоятельное задание

[42]:
```
using LaTeXStrings
```

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).

[32]:
```
# задаём описание модели с начальными условиями:
b = 4.2 # чуть больше 4 детей на семью
c = 2.1 # двое умирают
a = b - c
Maltus(u,p,t) = a*u
u0_1 = 1.0;
```

[39]:
```
# задаём интервал времени:
tspan_1 = (0.0,1.0)
```

[39]:
```
(0.0, 1.0)
```

[71]:
```
# решение:
prob_1 = ODEProblem(Maltus,u0_1,tspan_1)
sol_1 = solve(prob_1, abstol=1e-16, reltol=1e-16)
```

[71]:
```
retcode: Success
Interpolation: specialized 9th order lazy interpolation, specialized 4rd order "free" stiffness-aware interpolation
t: 63-element Vector{Float64}:
 0.0
 0.00915890363384417
 0.016579777630371828
 0.023921681410520583
 0.031394474505082974
 0.04002297337896621
 0.048059121461500494
 0.056642844965357
```

Рис. 2.16: Задание 6.4.1. Модель Мальтуса (1)

```
 0.06483751194505144
 0.07459464203321045
 0.08277236411239146
 0.09214225439582731
 0.10178079670017122
 ⋮
 0.7830430925256849
 0.8044017477419722
 0.8257702262777016
 0.8466694115956777
 0.8688960913954831
 0.8890305790641818
 0.9099448451019394
 0.9290961810244789
 0.9477639708777722
 0.9710765229336767
 0.9900513828337181
 1.0
u: 63-element Vector{Float64}:
 1.0
 1.0194198567875812
 1.0354307596418606
 1.0515187324306086
 1.0681502310906115
 1.08768136674715
 1.1061927394698512
 1.1263135723974413
 1.1458638137762651
 1.1695847230844163
 1.1898437127037809
 1.2134878505555127
 1.238300249927902
 ⋮
 5.177853014068694
 5.415383340363222
 5.663927041306394
 5.918042671140687
 6.200821553679382
 6.4686292009032975
 6.759062349330959
 7.036437324041015
 7.317760502217603
 7.684925618552657
 7.997331343374798
 8.166169912567648
```

Рис. 2.17: Задание 6.4.1. Модель Мальтуса (2)

```
[101]: # строим графики:
plot(sol_1, linewidth=5,title="Модель Мальтуса, где a = $a",
    xaxis="Время",yaxis="Численность популяции",label=L"$\dot{x} = ax$", legendfontsize=16)
```

[101]:

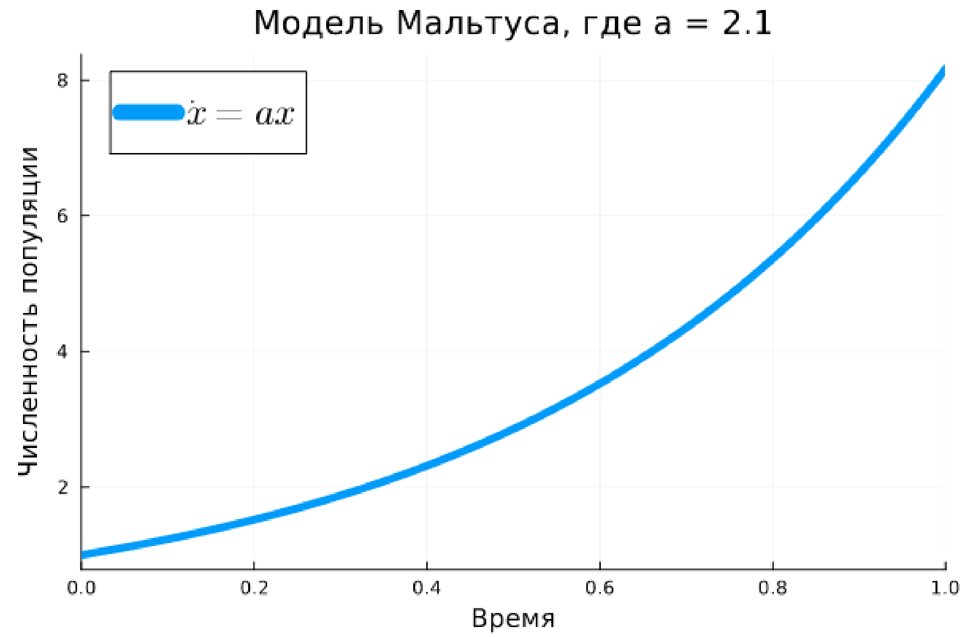


Рис. 2.18: Задание 6.4.1. Модель Мальтуса (3)

```
[84]: plot(sol_1, idxs=(0, 1), label = L"$u(t)$",
        title = "Анимация модели Мальтуса")
@gif for i ∈ 1:length(sol_1.u)
    scatter!((sol_1.t[i], sol_1.u[i]),
        ms = 3,
        color =:red,
        legend=false)
end
```

```
[ Info: Saved animation to C:\Users\User\Documents\work\study\2023-2024\Statistical_Analysis_computer-practise\computer-practice\labs\lab06\report\report\tmp.gif
```

[84]:

Рис. 2.19: Задание 6.4.1. Модель Мальтуса (4)

Рис. 2.20: Задание 6.4.1. Анимация модели Мальтуса

2. Реализуем и проанализируем логистическую модель роста популяции ([2.21-2.24]) с анимацией ([**gif:002?**]).

2. Реализовать и проанализировать логистическую модель роста популяции, заданную уравнением:

$$\dot{x} = rx(1 - \frac{x}{k}), \quad r > 0, k > 0,$$

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).

[86]:
```
# задаём описание модели с начальными условиями:
r = 2.1  # Популяция растет
k = 100 # Ограничение популяции
Logistic(u,p,t) = r*u*(1 - u/k)
u0_2 = 1.0; # Начальная популяция
```

[87]:
```
# задаём интервал времени:
tspan_2 = (0.0,5.0);
```

[89]:
```
# решение:
prob_2 = ODEProblem(Logistic,u0_2,tspan_2)
sol_2 = solve(prob_2, abstol=1e-12, reltol=1e-12)
```

[89]:
```
retcode: Success
Interpolation: specialized 9th order lazy interpolation, specialized 4rd order "free" stiffness-aware interpolation
t: 47-element Vector{Float64}:
 0.0
 0.02557129468588408
 0.07027392683941402
 0.1292981705139264
 0.19099650718606195
 0.2593943135157928
 0.33103713793150485
 0.4068732224998999
 0.4854476682373941
 0.5669163250279707
 0.6505911928998332
 0.7365118890180666
 0.824537484002157
 ⋮
```

Рис. 2.21: Задание 6.4.2. Логистическая модель роста популяции (1)

```
 3.481841649322189
 3.607413263333669
 3.737945568410705
 3.867103902707648
 3.999410228846137
 4.133649617117162
 4.27144608385097
 4.412968938376174
 4.559102753690844
 4.710386048062807
 4.867700635619142
 5.0
u: 47-element Vector{Float64}:
  1.0
  1.0545859167209892
  1.1571803397367753
  1.307885043836402
  1.486122298793529
  1.7117440394183117
  1.9841436385614546
  2.3187393463318813
  2.723393695909831
  3.215215015929423
  3.809332571623973
  4.528465981664943
  5.398301140761318
  ⋮
 93.80076790254658
 95.16829757362294
 96.28368336650614
 97.14129639552722
 97.81965334362793
 98.34640509830051
 98.75673291967054
 99.07341893375003
 99.3166056103863
 99.50168172764944
 99.64137356707081
 99.72813029773977
```

Рис. 2.22: Задание 6.4.2. Логистическая модель роста популяции (2)

[103]:

```
# строим график:
plot(sol_2, linewidth=5,title="Логистическая модель роста популяции, где r = $r, k = $k", titlefontsize = 11,
    xaxis="Время",yaxis="Численность популяции",label=L"$\dot{x} = rx(1 - \frac{x}{k})$", legendfontsize=16)
```

[103]:

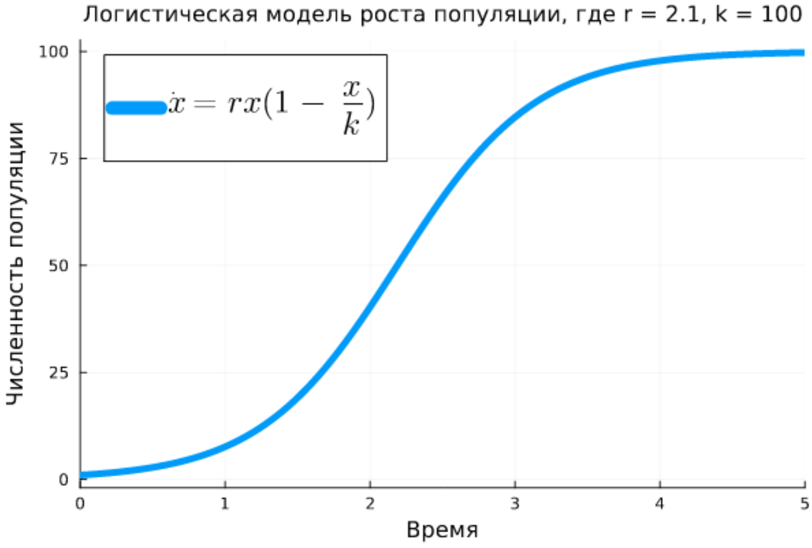


Рис. 2.23: Задание 6.4.2. Логистическая модель роста популяции (3)

[92]:

```
plot(sol_2, idxs=(0, 1), label = L"$u(t)$",
        title = "Анимация логистической модели роста популяции")
@gif for i ∈ 1:length(sol_2.u)
    scatter!((sol_2.t[i], sol_2.u[i]),
        ms = 3,
        color =:red,
        legend=false)
end
```

```
[ Info: Saved animation to C:\Users\User\Documents\work\study\2023-2024\Statistical_Analysis_computer-practise\computer-practice\labs\lab06\report\report\tmp.gif
```

[92]:

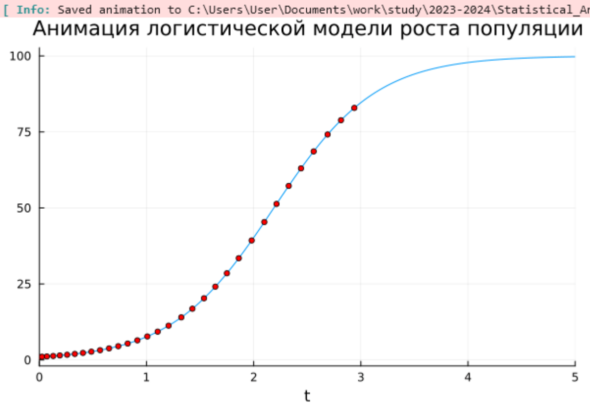


Рис. 2.24: Задание 6.4.2. Логистическая модель роста популяции (4)

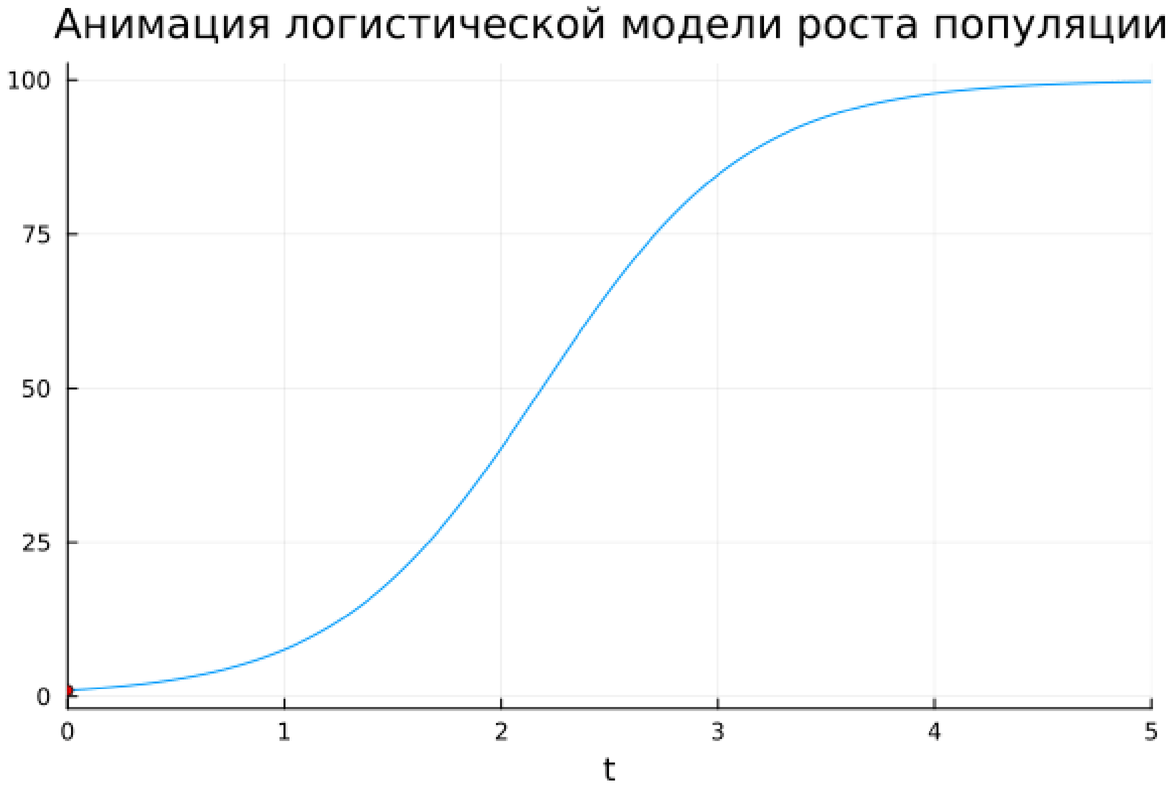


Рис. 2.25: Задание 6.4.2. Анимация логистической модели роста популяции

3. Реализуем и проанализируем модель эпидемии Кермака–Маккендрика — SIR-модель ([2.26-2.29]) с анимацией ([**gif:003?**]).

3. Реализовать и проанализировать модель эпидемии Кермака–Маккендрика (SIR-модель):

$$\begin{cases} \dot{s} = -\beta \imath s, \\ \dot{\imath} = \beta \imath s - \nu \imath, \\ \dot{r} = \nu \imath, \end{cases}$$

где $s(t)$ — численность восприимчивых к болезни индивидов в момент времени t, $\imath(t)$ — 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. $\dot{s} + \dot{\imath} + \dot{r} = 0$. Начальные данные и параметры задать самостоятельно и пояснить их выбор. Построить соответствующие графики (в том числе с анимацией).

[104]:
```
# задаём описание модели:
function SIR!(du,u,p,t)
    β, ν = p
    du[1] = -β*u[1]*u[2]
    du[2] = β*u[1]*u[2] - ν*u[2]
    du[3] = ν*u[2]
end
```

[104]:
```
SIR! (generic function with 1 method)
```

[148]:
```
# задаём начальное условие:
u0_3 = [1000.0, 1.0, 0.0]
# задаём значения параметров (β, ν):
p_3 = (0.02, 2)
# задаём интервал времени:
tspan_3 = (0.0,1.5);
```

[158]:
```
# решение:
prob_3 = ODEProblem(SIR!,u0_3,tspan_3,p_3)
sol_3 = solve(prob_3, abstol=1e-8, reltol=1e-8)
```

[158]:
```
retcode: Success
Interpolation: specialized 7th order lazy interpolation, specialized 3rd order "free" stiffness-aware interpolation
t: 63-element Vector{Float64}:
 0.0
 0.019428660504419996
 0.03103030076140588
 0.046548992414566434
 0.06036452698837914
 0.07528982785392146
 0.08987302010494794
 0.10479621908168664
 0.11969815632827989
```

Рис. 2.26: Задание 6.4.3. SIR-модель (1)

```
 0.10479621908168664
 0.11969815632827989
 0.13475915642728148
 0.14991706841831004
 0.16526698934205905
 0.18088183839045802
 ⋮
 0.9561587778815379
 0.9877355126548749
 1.0213013154709596
 1.057170834264557
 1.0957302548429728
 1.1374700813772423
 1.1830320363924256
 1.233299967722454
 1.2896015810740609
 1.3543106317923201
 1.4347985931997127
 1.5
u: 63-element Vector{Vector{Float64}}:
 [1000.0, 1.0, 0.0]
 [999.5349394257488, 1.4185436994052925, 0.0465168748459432]
 [999.1691628881451, 1.747718867024899, 0.08311824482999013]
 [998.5441982627083, 2.310115492669723, 0.1456862446219985]
 [997.8210047226144, 2.9608580031977985, 0.21813727418783002]
 [996.8101774356394, 3.870330475052082, 0.3194920893085434]
 [995.5249741383911, 5.026518985355521, 0.44850687625329416]
 [993.8144363799813, 6.565086272469757, 0.6204773475489226]
 [991.5890093455097, 8.566334390432047, 0.8446562640582364]
 [988.6586110624191, 11.200769644259358, 1.140619293321447]
 [984.814052340275, 14.655704161616733, 1.5302434981081592]
 [979.7373455960992, 19.215578610127878, 2.0470757937727875]
 [972.9966571150727, 25.265879639983837, 2.7374632449433656]
 ⋮
 [1.1880866922377886, 326.27080457544645, 673.5411087323158]
 [0.9729671690742073, 306.51101101116956, 693.5160218197562]
 [0.7973383368096075, 286.77951604136695, 713.4231456218234]
 [0.6537335784825186, 267.06520024506517, 733.2810661764523]
 [0.536162588984245, 247.356526895993, 753.1073105150227]
 [0.439785974836303, 227.63797701071206, 772.9222370144516]
 [0.36067934392160567, 207.8871968288709, 792.7521238272075]
 [0.2956326419479863, 188.06507247267325, 812.6392948853787]
 [0.24196844323299724, 168.08770754413726, 832.6703240126297]
 [0.19730103104997262, 147.7247070574961, 853.077991911454]
 [0.15838852485212057, 125.79565894403567, 875.0459525311122]
 [0.1358078148887278, 110.43721244121176, 890.4269797438996]
```

Рис. 2.27: Задание 6.4.3. SIR-модель (2)

```
[159]: plot(sol_3, label = ["Восприимчивые" "Инфицированные" "Выздоровевшие"], color=["black" "red" "green"], ls=:solid, title="Модель эпидемии SIR, где β = $(p_3[1]), ν = $(p_3[2])",
    xaxis="Время",yaxis="Число индивидов")
```

[159]:

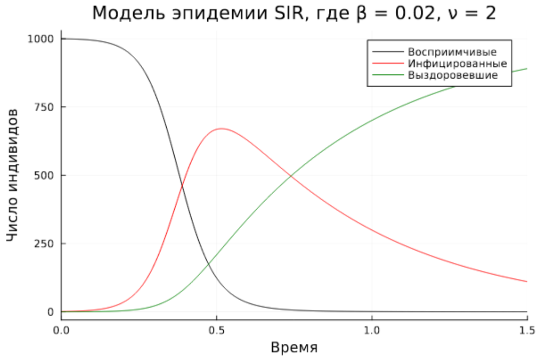


Рис. 2.28: Задание 6.4.3. SIR-модель (3)

```
[160]: plot(sol_3, label = ["Восприимчивые" "Инфицированные" "Выздоровевшие"], color=["black" "red" "green"], ls=:solid, title="Анимация модели эпидемии SIR",
    xaxis="Время",yaxis="Число индивидов")
@gif for i ∈ 1:length(sol_3.u)
    scatter!((sol_3.t[i], sol_3.u[i][2]),
        ms = 3,
        color =:red,
        legend=false)
end
```

```
[ Info: Saved animation to C:\Users\User\Documents\work\study\2023-2024\Statistical_Analysis_computer-practise\computer-practice\labs\lab06\report\report\tmp.gif
```

[160]:

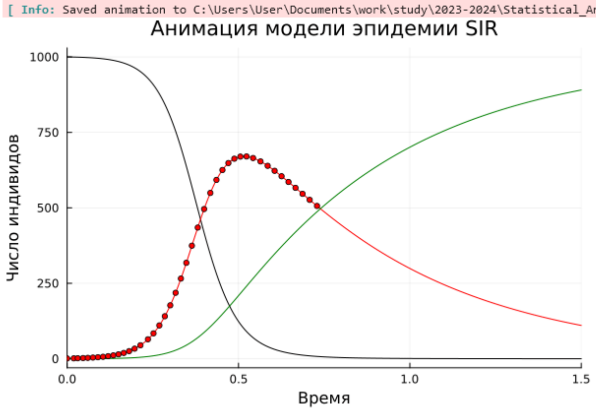


Рис. 2.29: Задание 6.4.3. SIR-модель (4)

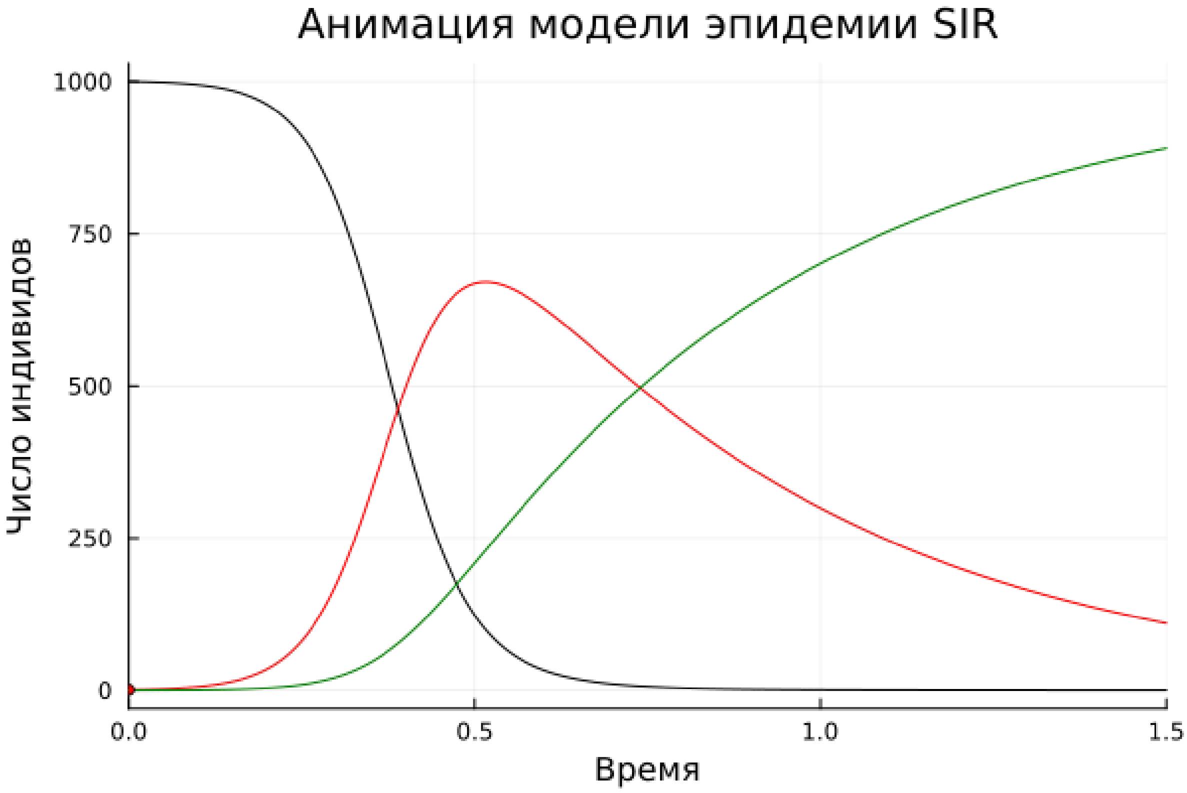


Рис. 2.30: Задание 6.4.3. Анимация SIR-модели

4. Исследуем и сравним расширение модели SIR (Susceptible-Infected-Removed) — модель SEIR (Susceptible-Exposed-Infected-Removed) ([2.31-2.34]) с анимацией ([**gif:004?**]).

4. Как расширение модели SIR (Susceptible-Infected-Removed) по результатом эпидемии испанки была предложена модель SEIR (Susceptible-Exposed-Infected-Removed):

$$\begin{cases} \dot{s}(t) = -\frac{\beta}{N} s(t)\imath(t), \\ \dot{e}(t) = \frac{\beta}{N} s(t)\imath(t) - \delta e(t), \\ \dot{\imath}(t) = \delta e(t) - \gamma\imath(t), \\ \dot{r}(t) = \gamma\imath(t). \end{cases}$$

Размер популяции сохраняется:

$$s(t) + e(t) + \imath(t) + r(t) = N.$$

Исследуйте, сравните с SIR.

```
[161]: # задаём описание модели:
SEIR! = @ode_def SEIR begin
ds = -β/N*s*i
de = β/N*s*i - δ*e
di = δ*e - γ*i
dr = γ*i
end N β δ γ
```

```
[161]: (::SEIR{var"###ParameterizedDiffEqFunction#4940", var"###ParameterizedTGradFunction#4941", var"###ParameterizedJacobianFunction#4942", Nothing, Nothing, ODESystem}) (generic function with 1 method)
```

```
[205]: # задаём начальное условие:
u0_4 = [1000.0, 10.0, 1.0, 0.0]
# задаём значения параметров (N β δ γ):
p_4 = (1111, 6.0, 2.0, 1.0)
# задаём интервал времени:
tspan_4 = (0.0, 8.0);
```

```
[206]: # решение:
prob_4 = ODEProblem(SEIR!,u0_4,tspan_4,p_4)
sol_4 = solve(prob_4, abstol=1e-12, reltol=1e-12)
```

```
[206]: retcode: Success
Interpolation: specialized 9th order lazy interpolation, specialized 4rd order "free" stiffness-aware interpolation
t: 102-element Vector{Float64}:
 0.0
 0.020382163204281788
 0.04668077162216282
 0.07464937140656683
 0.10623781272295595
```

Рис. 2.31: Задание 6.4.4 SEIR-модель (1)

```
 0.3462978763041009
 0.39443426105910573
 0.4449320040234373
 ⋮
 6.660747149822965
 6.77463567452797
 6.889829290478822
 7.006363795802013
 7.124391656821628
 7.244012953805387
 7.365342527138759
 7.48853015352197
 7.613719040182397
 7.741043991632618
 7.870705989068766
 8.0
u: 102-element Vector{Vector{Float64}}:
 [1000.0, 10.0, 1.0, 0.0]
 [999.868975522941, 9.729081760633262, 1.3776797612017566, 0.02426295522398142]
 [999.6402576235291, 9.453670539776526, 1.839447555477261, 0.06662428121713568]
 [999.3271824125596, 9.244174215600447, 2.3040180521425335, 0.1246253196974137]
 [998.8916633979782, 9.100700907000231, 2.8022948864816613, 0.2053408085399178]
 [998.3343358859566, 9.0454694126127, 3.3115120774343585, 0.3086826239962734]
 [997.6284150717698, 9.088361548891783, 3.8435633515994985, 0.4396600277388804]
 [996.7650865302086, 9.23930154494432, 4.395642832587444, 0.5999690922596957]
 [995.7235240999514, 9.50711252447585, 4.975804558509255, 0.7935588170634731]
 [994.4862825920128, 9.89937345869507, 5.590562253586535, 1.0237816957056267]
 [993.0289918519918, 10.425134749715014, 6.250554954648523, 1.2953184436447132]
 [991.3249818127523, 11.094223520776131, 6.967462460487337, 1.6133322059843052]
 [989.3375845103593, 11.92048818520434, 7.757002416704328, 1.9849248877320531]
 ⋮
 [5.513734173096658, 1.9299208924281654, 40.59465499480366, 962.9616899396718]
 [5.384474201871158, 1.6521167026358916, 36.6091144975088, 967.3542945979844]
 [5.269309990648219, 1.414765023272131, 32.95828221176552, 971.3576427743144]
 [5.166654142236698, 1.2119799012437364, 29.620733583478675, 975.0006323730412]
 [5.075032815580003, 1.038552678021309, 26.572720854129063, 978.3136936522699]
 [4.993216069947593, 0.8901556240991626, 23.793460469161435, 981.3231678367921]
 [4.920124072384285, 0.7630946871061477, 21.263061114858196, 984.0537201256517]
 [4.854797643630382, 0.65420621406112, 18.962277750565153, 986.5287183917437]
 [4.796405570779902, 0.5608210729044483, 16.873422956068648, 988.7693504002474]
 [4.744223572783781, 0.4806826318985952, 14.980203394548937, 990.7948904007691]
 [4.697591194427492, 0.4118417618021664, 13.26661464973535, 992.6239523940354]
 [4.656789889070201, 0.3538520473116087, 11.75010059432728, 994.2392574692914]
```

Рис. 2.32: Задание 6.4.4 SEIR-модель (2)

```
[207]: plot(sol_4, label = ["Восприимчивые" "Незащищенные" "Инфицированные" "Выздоровевшие"], color=["black" "purple" "red" "green"], ls=:solid,
           title="Модель эпидемии SEIR, где β = $(p_4[2]), δ = $(p_4[3]), γ = $(p_4[4])", titlefontsize=11,
           xaxis="Время",yaxis="Число индивидов")
```

[207]:

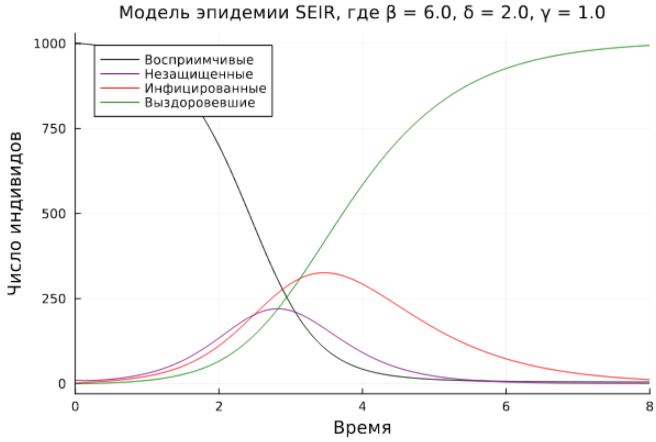


Рис. 2.33: Задание 6.4.4 SEIR-модель (3)

```
[209]: plot(sol_4, label = ["Восприимчивые" "Незащищенные" "Инфицированные" "Выздоровевшие"], color=["black" "purple" "red" "green"], ls=:solid,
           title="Анимация модели эпидемии SEIR, где β = $(p_4[2]), δ = $(p_4[3]), γ = $(p_4[4])", titlefontsize=11,
           xaxis="Время",yaxis="Число индивидов")
       @gif for i ∈ 1:length(sol_4.u)
           scatter!((sol_4.t[i], sol_4.u[i][3]),
               ms = 3,
               color =:red,
               legend=false)
           scatter!((sol_4.t[i], sol_4.u[i][2]),
               ms = 3,
               color =:purple,
               legend=false)
       end
```

```
[ Info: Saved animation to C:\Users\User\Documents\work\study\2023-2024\Statistical_Analysis_computer-practise\computer-practice\labs\lab06\report\report\tmp.gif
```

[209]:

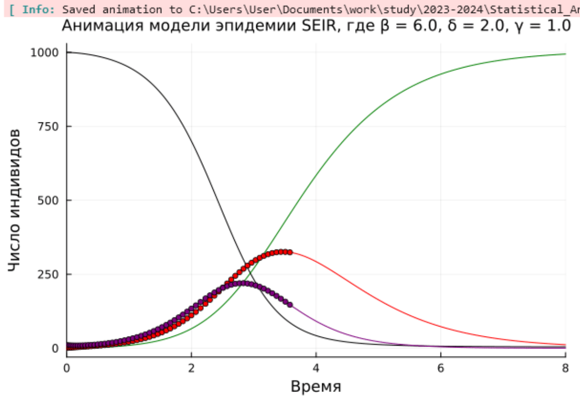


Рис. 2.34: Задание 6.4.4 SEIR-модель (4)

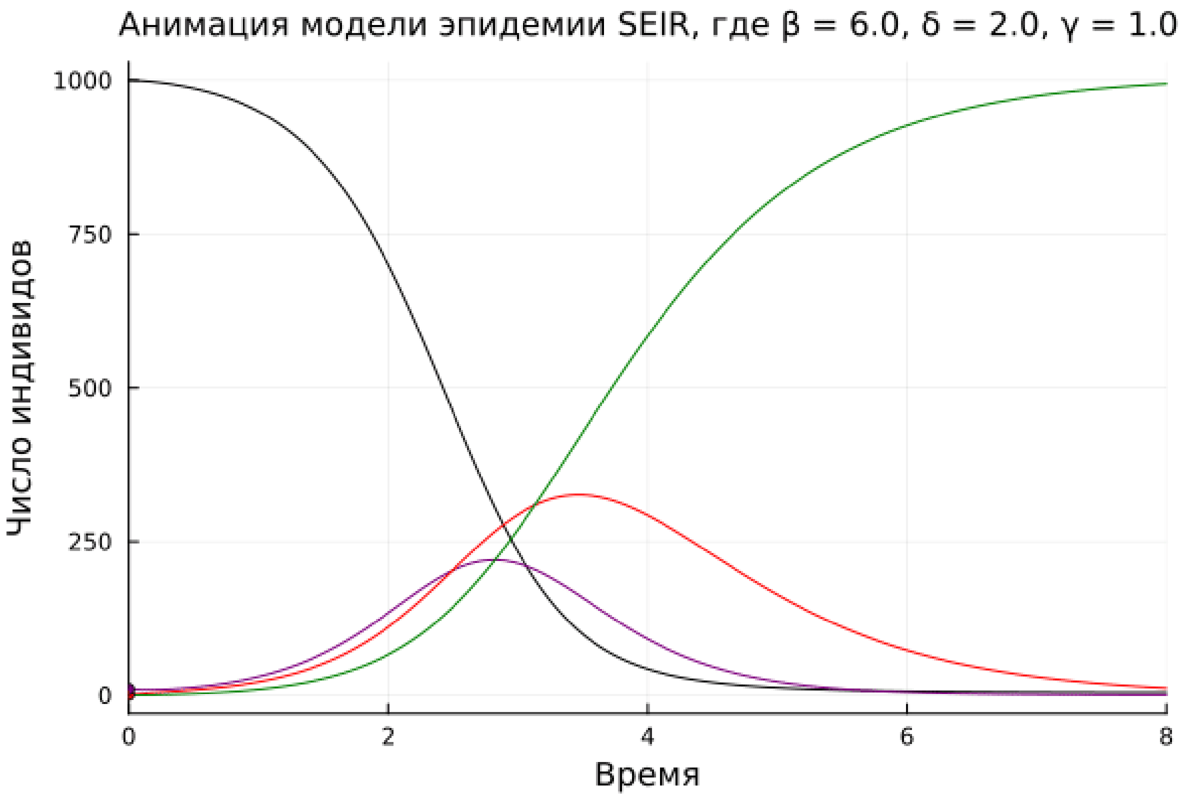


Рис. 2.35: Задание 6.4.4 Анимация SEIR-модели

5. Реализуем и проанализируем дискретную модель Лотки-Вольтерры ([2.36-2.40]) с анимацией ([**gif:005?**]).

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.

```
[210]: # задаём описание модели:
Discrete_lv! = @ode_def DiscreteLotkaVolterra begin
dx = a*x*(1 - x) - x*y
dy = -c*y + d*x*y
end a c d
```

```
[210]: (::DiscreteLotkaVolterra{var"###ParameterizedDiffEqFunction#6422", var"###ParameterizedTGradFunction#6423", var"###ParameterizedJacobianFunction#6424", Nothing, Nothing, ODESystem}) (generic function with 1 m
ethod)
```

```
[211]: # задаём начальное условие:
u0_5 = [1.0,1.0]
# задаём значения параметров (a, c, d):
p_5 = (2, 1, 5)
# задаём интервал времени:
tspan_5 = (0.0,100.0);
```

```
[212]: # решение:
prob_5 = ODEProblem(Discrete_lv!, u0_5, tspan_5, p_5)
sol_5 = solve(prob_5, abstol=1e-12, reltol=1e-12)
```

```
[212]: retcode: Success
Interpolation: specialized 9th order lazy interpolation, specialized 4rd order "free" stiffness-aware interpolation
t: 293-element Vector{Float64}:
   0.0
   0.023933673062369404
   0.056549426341570126
   0.0918546039316109
   0.13157913167634175
   0.17623998604311483
   0.2256889594785779
   0.2723828322821779
   0.32165420771033587
```

Рис. 2.36: Задание 6.4.5 Дискретная модель Лотки-Вольтерры (1)

```
  0.32165420771033587
  0.37122971905692975
  0.42278215285380416
  0.4764297496513247
  0.5347420142156215
  ⋮
 89.0731860409692
 89.96511604393481
 90.93042976050276
 91.91706073863338
 92.86472583729973
 93.91152283948671
 94.89784659478913
 95.93873953516206
 97.03622910204601
 98.06762574737543
 99.24469633467092
100.0
u: 293-element Vector{Vector{Float64}}:
 [1.0, 1.0]
 [0.9757714161865111, 1.0988794266806567]
 [0.9417514627049559, 1.2436430769136675]
 [0.9035550129208922, 1.4128843796902384]
 [0.8588695418863462, 1.6176375910710448]
 [0.8066309921585056, 1.8632072388536296]
 [0.7467904620186603, 2.1488319825950906]
 [0.6890708878732563, 2.4250689586907943]
 [0.6278784323821484, 2.715030487416396]
 [0.5671568570961832, 2.9960927384342595]
 [0.5061686446433937, 3.2675084273006783]
 [0.44626818724848427, 3.5184702569286443]
 [0.3864388591902479, 3.7470229990694213]
 ⋮
 [0.1999999954241612, 1.599999966196368]
 [0.20000000292596892, 1.5999999614558014]
 [0.20000000525348705, 1.599999998332996]
 [0.20000000098740978, 1.6000000253948319]
 [0.1999999970719864, 1.6000000158932777]
 [0.19999999775589372, 1.5999999906615574]
 [0.20000000082360433, 1.5999999851641937]
 [0.20000000193455958, 1.5999999990470877]
 [0.20000000018494923, 1.6000000094785958]
 [0.199999998832913, 1.6000000042618405]
 [0.19999999951771108, 1.5999999949974806]
 [0.2000000003678781, 1.5999999947672527]
```

Рис. 2.37: Задание 6.4.5 Дискретная модель Лотки-Вольтерры (2)

[228]:

```
# строим график:
plot(sol_5, label = ["Жертвы" "Хищники"], color="black", ls=[:solid :dash],
    title="Дискретная модель Лотки - Вольтерры, где a = $(p_5[1]), c = $(p_5[2]), d = $(p_5[3])",
    titlefontsize = 11,
    xaxis="Время",yaxis="Размер популяции")
```

[228]:

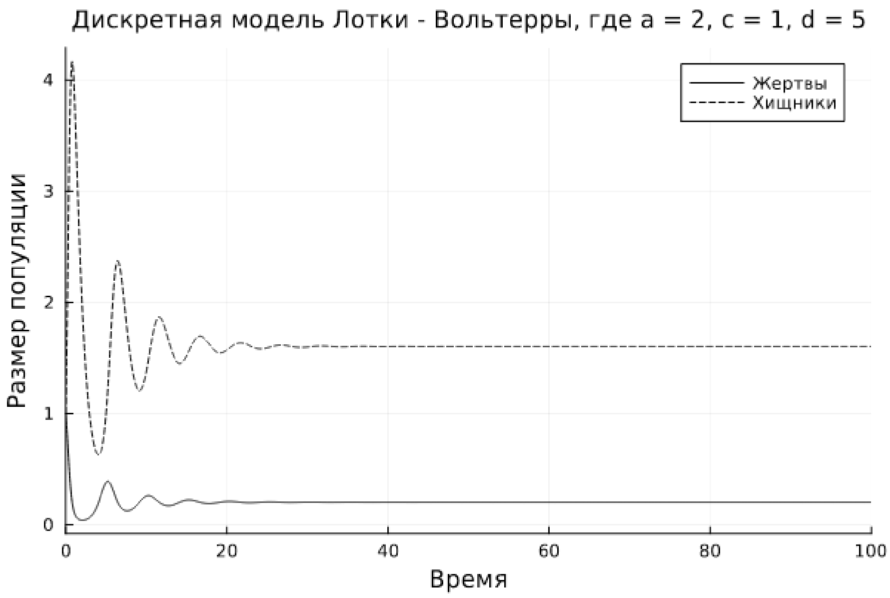
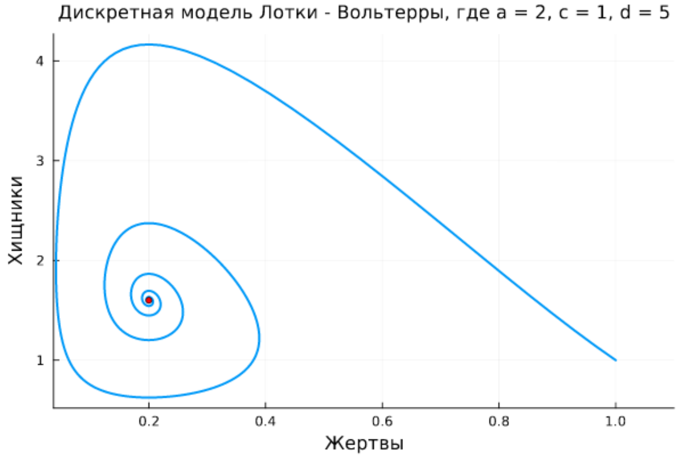


Рис. 2.38: Задание 6.4.5 Дискретная модель Лотки-Вольтерры (3)

[229]:
```
# Фазовый портрет
plot(sol_5, idxs=(1,2), lw=2,
    title="Дискретная модель Лотки - Вольтерры, где a = $(p_5[1]), c = $(p_5[2]), d = $(p_5[3])",
    titlefontsize = 11,
    xaxis="Жертвы",yaxis="Хищники",legend=false)
# Отметим точку равновесия
scatter!((sol_5.u[end][1], sol_5.u[end][2]), color =:red, ms = 3)
```

[229]:



Точка равновесия - $(0.2, 1.6)$ (по аналогии с непрерывной моделью, можно найти x координату особой точки - $x =^{\gamma} /_{\delta} =^{c} /_{d} =^{1} /_{5} = 0.2$)

Рис. 2.39: Задание 6.4.5 Дискретная модель Лотки-Вольтерры (4)

[230]:
```
plot(sol_5, idxs=(1,2), lw=2,
    title="Анимация дискретной модели Лотки - Вольтерры, где a = $(p_5[1]), c = $(p_5[2]), d = $(p_5[3])",
    titlefontsize = 10,
    xaxis="Жертвы",yaxis="Хищники",legend=false)
@gif for i ∈ 1:length(sol_5.u)
    scatter!((sol_5.u[i][1], sol_5.u[i][2]),
        ms = 3,
        color =:purple,
        legend=false)
end
```

```
[ Info: Saved animation to C:\Users\User\Documents\work\study\2023-2024\Statistical_Analysis_computer-practise\computer-practice\labs\lab06\report\report\tmp.gif
```

[230]:

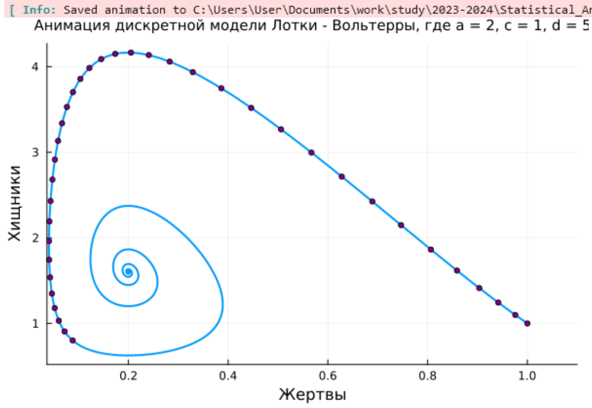


Рис. 2.40: Задание 6.4.5 Дискретная модель Лотки-Вольтерры (5)

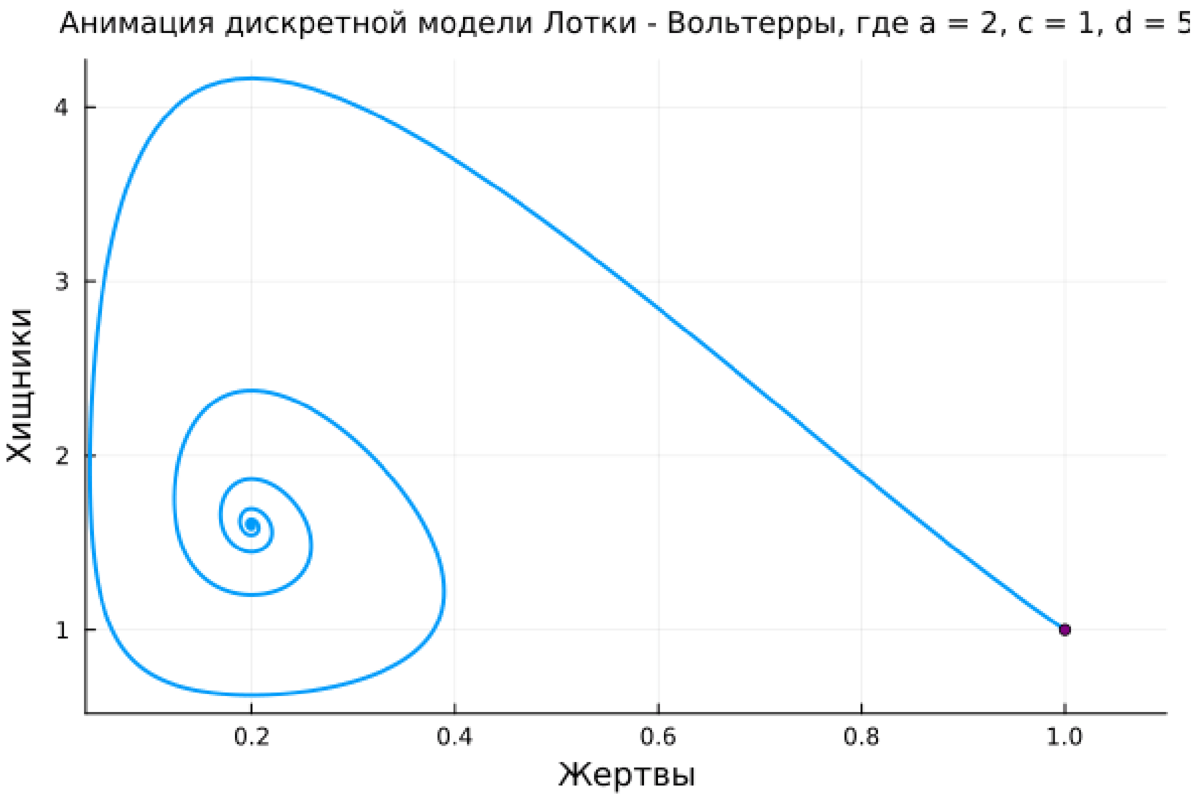


Рис. 2.41: Задание 6.4.5 Анимация дискретной модели Лотки-Вольтерры

6. Реализуем модель отбора на основе конкурентных отношений ([2.42-2.46]) с анимацией ([**gif:006?**]).

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

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

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

[246]:
```
# задаём описание модели:
Concurrent! = @ode_def ConcurrentRelations begin
dx = α*x - β*x*y
dy = α*y - β*x*y
end α β
```

[246]:
```
(::ConcurrentRelations{var"###ParameterizedDiffEqFunction#9384", var"###ParameterizedTGradFunction#9385", var"###ParameterizedJacobianFunction#9386", Nothing, Nothing, ODESystem}) (generic function with 1 met
hod)
```

[247]:
```
# задаём начальное условие:
u0_6 = [10.0,7.0]
# задаём значения параметров (α, β):
p_6 = (1.5, 2.0)
# задаём интервал времени:
tspan_6 = (0.0,1.0);
```

[248]:
```
# решение:
prob_6 = ODEProblem(Concurrent!, u0_6, tspan_6, p_6)
sol_6 = solve(prob_6, abstol=1e-12, reltol=1e-12)
```

[248]:
```
retcode: Success
Interpolation: specialized 9th order lazy interpolation, specialized 4rd order "free" stiffness-aware interpolation
t: 57-element Vector{Float64}:
 0.0
 0.007087361517120001
 0.011572834285818512
 0.017733526518712985
 0.02336363261357824
 0.029788666423539605
 0.03633429574423381
 0.04343652489310536
 0.05090694550599869
 0.058909545950189206
 0.06740779039919279
 0.07647890860091436
 0.08613301116009514
 ⋮
```

Рис. 2.42: Задание 6.4.6 Модель отбора на основе конкурентных отношений (1)

```
 0.07647890860091436
 0.08613301116009514
 ⋮
 0.7598988981443658
 0.782541912210457
 0.8052686664838677
 0.8281174911343668
 0.8511291268829188
 0.8743473120647968
 0.8978191184513504
 0.9215960025536083
 0.9457340509501627
 0.9702953343602193
 0.9953488637651823
 1.0
u: 57-element Vector{Vector{Float64}}:
 [10.0, 7.0]
 [9.207740187748813, 6.175676929976707]
 [8.787995587686826, 5.735463191045128]
 [8.291872561088297, 5.211000854856821]
 [7.904442433814394, 4.7974421008136785]
 [7.524129833807606, 4.387040870065517]
 [7.191747472816011, 4.023705470140905]
 [6.882136856409725, 3.6801641967837546]
 [6.603214864781218, 3.3651602986086075]
 [6.347815535915543, 3.070657456580873]
 [6.116808437119743, 2.79760782562222]
 [5.907683397063587, 2.5430108344206785]
 [5.720011986171117, 2.306260581062883]
 ⋮
 [9.380211759066299, 0.001329105023872399]
 [9.703795976402699, 0.0008926086745578159]
 [10.03996896189608, 0.0005896744393536682]
 [10.389808298910623, 0.00038263954972584963]
 [10.754546767801868, 0.00024349421946603028]
 [11.135596591921283, 0.00015167234947098172]
 [11.53457408296595, 9.228443721311987e-5]
 [11.95334158529712, 5.471460394318872e-5]
 [12.394042436149075, 3.152256497306574e-5]
 [12.85916334716283, 1.7590598739446114e-5]
 [13.35160087985934, 9.472094272332633e-6]
 [13.445075631708178, 8.420694000484729e-6]
```

Рис. 2.43: Задание 6.4.6 Модель отбора на основе конкурентных отношений (2)

```
[250]: # строим график:
plot(sol_6, label = [L"$x(t)$" L"$y(t)$"], color="black", ls=[:solid :dash], title="Модель конкурентных отношений, где α = $(p_6[1]), β = $(p_6[2])",
    xaxis="Время",yaxis="Число индивидов")
```

[250]:

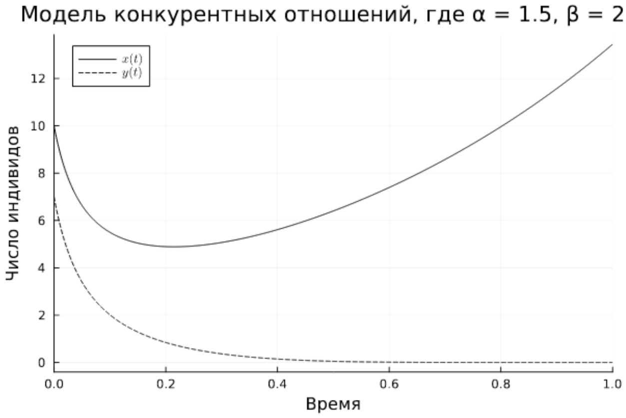


Рис. 2.44: Задание 6.4.6 Модель отбора на основе конкурентных отношений (3)

```
[251]: # Фазовый портрет
plot(sol_6, idxs=(1,2), lw=2, title="Модель конкурентных отношений, где α = $(p_6[1]), β = $(p_6[2])",
    xaxis="x",yaxis="y",legend=false)
```

[251]:

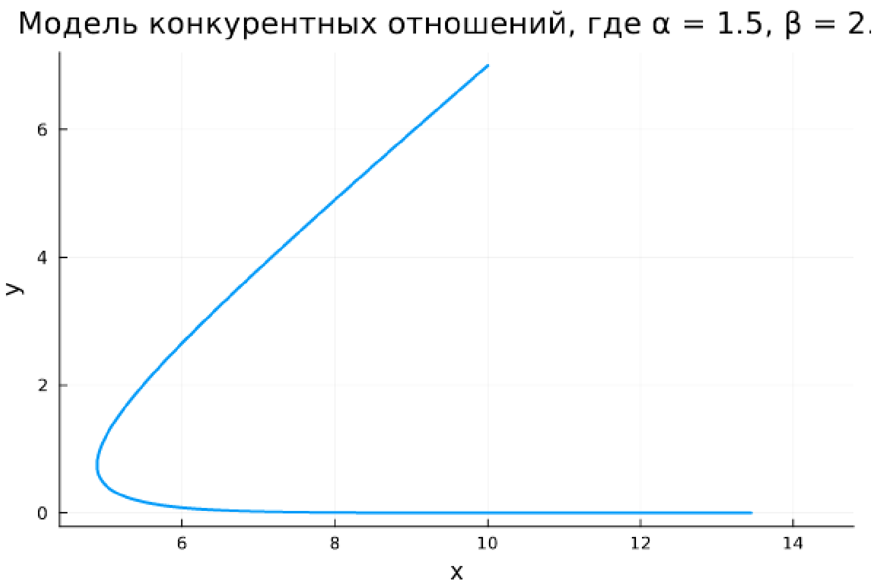


Рис. 2.45: Задание 6.4.6 Модель отбора на основе конкурентных отношений (4)

```
[245]: plot(sol_6, idxs=(1,2), lw=2, title="Модель конкурентных отношений, где α = $(p_6[1]), β = $(p_6[2])",
           xaxis="x",yaxis="y",legend=false)
       @gif for i ∈ 1:length(sol_6.u)
           scatter!((sol_6.u[i][1], sol_6.u[i][2]),
               ms = 3,
               color =:purple,
               legend=false)
       end
```

```
[ Info: Saved animation to C:\Users\User\Documents\work\study\2023-2024\Statistical_Analysis_computer-practise\computer-practice\labs\lab06\report\report\tmp.gif
```

[245]:

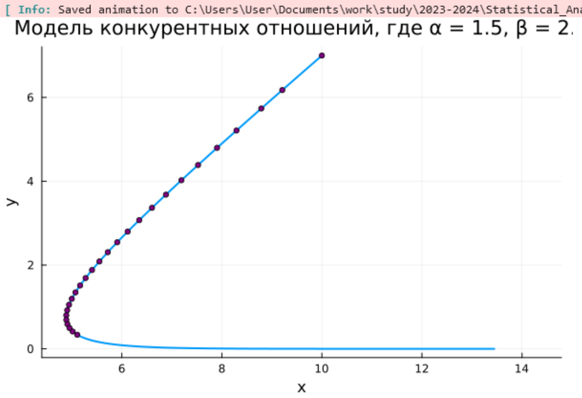


Рис. 2.46: Задание 6.4.6 Модель отбора на основе конкурентных отношений (5)

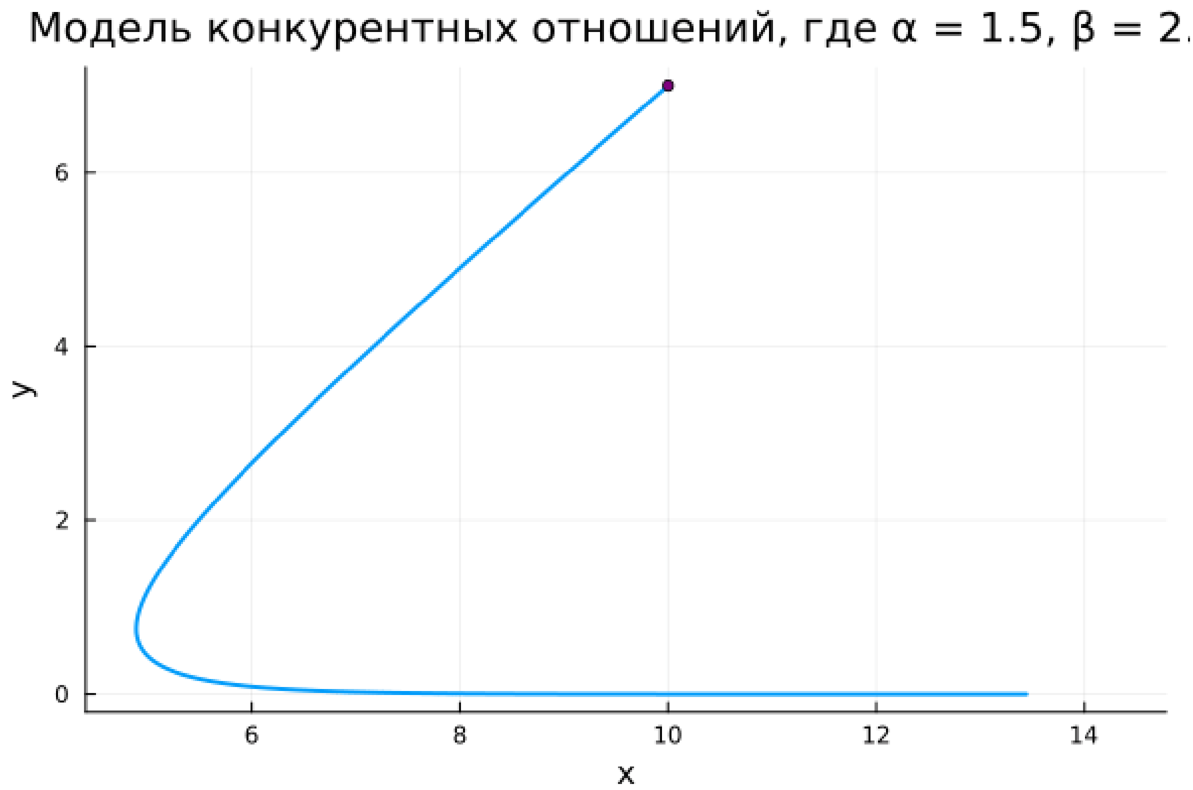
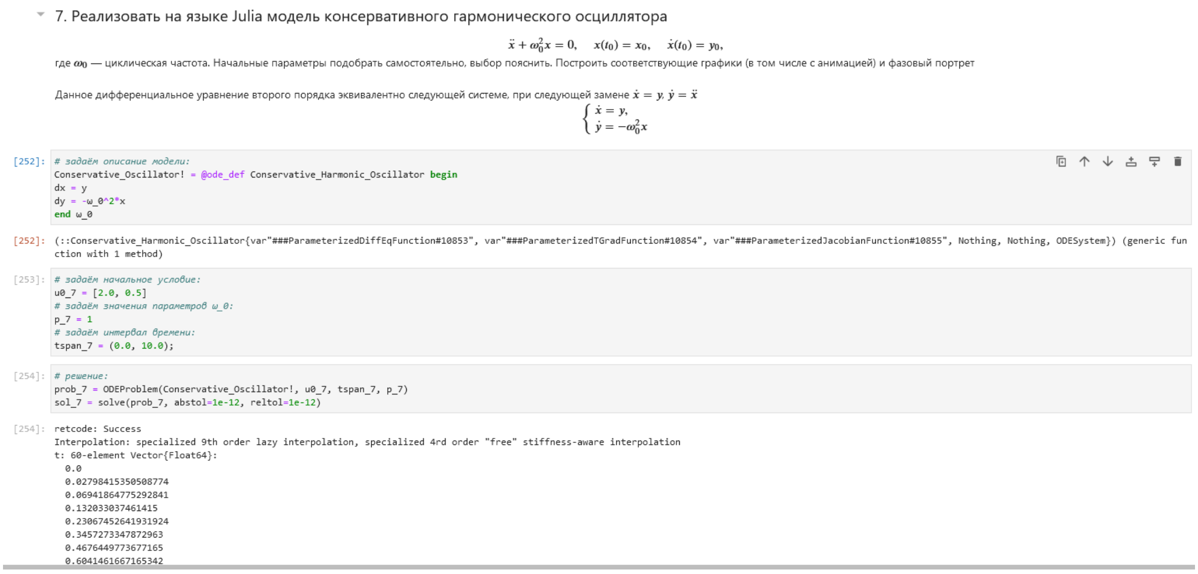


Рис. 2.47: Задание 6.4.6 Анимация модели отбора на основе конкурентных отношений

7. Реализуем модель консервативного гармонического осциллятора ([2.48-2.52]) с анимацией ([**gif:007?**]).

   Данное дифференциальное уравнение второго порядка эквивалентно следующей системе, при замене $\dot{x} = y, \dot{y} = \ddot{x}$ получим

$$
\begin{cases}
\dot{x} = y, \\
\dot{y} = -\omega_0^2 x.
\end{cases}
$$



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

Данное дифференциальное уравнение второго порядка эквивалентно следующей системе, при следующей замене $\dot{x} = y, \dot{y} = \ddot{x}$

$$
\begin{cases}
\dot{x} = y, \\
\dot{y} = -\omega_0^2 x
\end{cases}
$$

```
[252]: # задаём описание модели:
Conservative_Oscillator! = @ode_def Conservative_Harmonic_Oscillator begin
dx = y
dy = -ω_0^2*x
end ω_0

[252]: (::Conservative_Harmonic_Oscillator{var"###ParameterizedDiffEqFunction#10853", var"###ParameterizedTGradFunction#10854", var"###ParameterizedJacobianFunction#10855", Nothing, Nothing, ODESystem}) (generic fun
ction with 1 method)

[253]: # задаём начальное условие:
u0_7 = [2.0, 0.5]
# задаём значения параметров ω_0:
p_7 = 1
# задаём интервал времени:
tspan_7 = (0.0, 10.0);

[254]: # решение:
prob_7 = ODEProblem(Conservative_Oscillator!, u0_7, tspan_7, p_7)
sol_7 = solve(prob_7, abstol=1e-12, reltol=1e-12)

[254]: retcode: Success
Interpolation: specialized 9th order lazy interpolation, specialized 4rd order "free" stiffness-aware interpolation
t: 60-element Vector{Float64}:
  0.0
  0.02798415350508774
  0.06941864775292841
  0.132033037461415
  0.23067452641931924
  0.3457273347872963
  0.4676449773677165
  0.6041461667165342
```

Рис. 2.48: Задание 6.4.7. Модель консервативного гармонического осциллятора (1)

```
 0.9031173585671005
 1.0644100601365012
 1.231788046470059
 1.4028500315146972
 ⋮
 8.053776000357075
 8.23742653304688
 8.419999815861585
 8.606298863867748
 8.794047158401023
 8.983479226698481
 9.172928539224669
 9.3615897761321
 9.548578625364959
 9.73334538293183
 9.915131120565922
10.0
u: 60-element Vector{Vector{Float64}}:
 [2.0, 0.5]
 [2.013207188851619, 0.44384323218469907]
 [2.0298644397398378, 0.3600699324848906]
 [2.0484174651939235, 0.2323486352197925]
 [2.0613419109689177, 0.029487727667799505]
 [2.051098975318877, -0.207347518542793]
 [2.0106574285400134, -0.4552545497376837]
 [1.9300046461766427, -0.7246254658349875]
 [1.8055777016032617, -0.9949317380971839]
 [1.6309605227405621, -1.2609392424941943]
 [1.4072918743048068, -1.506495795054093]
 [1.1366464236904212, -1.7198938651881326]
 [0.827280856645323, -1.8882813307947994]
 ⋮
 [0.09311812118569404, -2.0594487164061066]
 [-0.2845441641512404, -2.0418213973429173]
 [-0.650529460271926, -1.9562237656562262]
 [-1.0016111434785118, -1.8018809942001095]
 [-1.320325933895388, -1.5832685900639776]
 [-1.5948383678423337, -1.3063271337830744]
 [-1.812308690171817, -0.9826175306433377]
 [-1.9644355096665913, -0.6252944333358922]
 [-2.0464352939549966, -0.24920390778480764]
 [-2.0573862414573907, 0.1310032574469876]
 [-1.9998019265495428, 0.5007916278939031]
 [-1.9501536135975635, 0.6685064572405073]
```

Рис. 2.49: Задание 6.4.7. Модель консервативного гармонического осциллятора (2)

[261]:
```
# строим график:
plot(sol_7, label = ["Координата колебаний" "Скорость колебаний"], color="black", ls=[:solid :dash],
    title="Модель консервативного гармонического осциллятора, где ω = $(p_7)",
    titlefontsize = 10,
    xaxis="Время",yaxis="Число")
```

[261]:

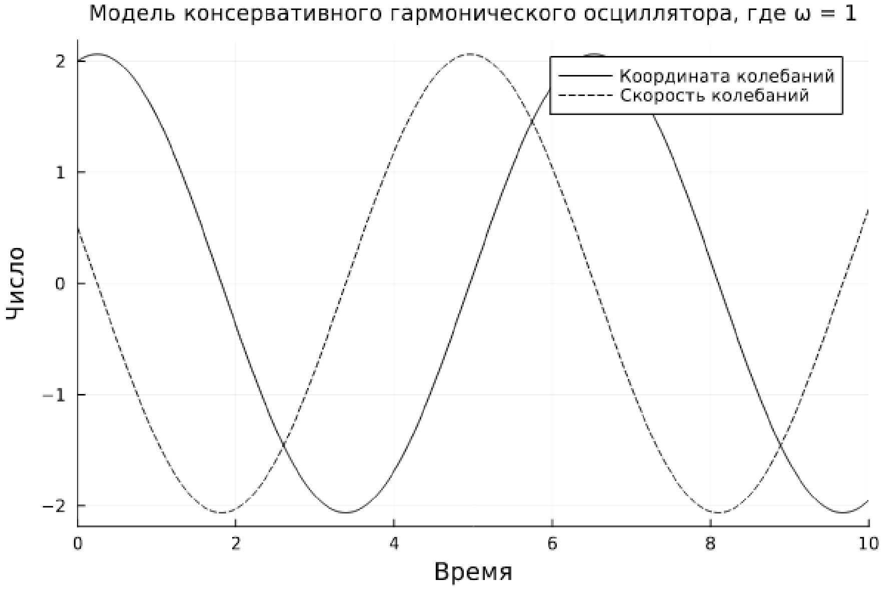


Рис. 2.50: Задание 6.4.7. Модель консервативного гармонического осциллятора (3)

[263]:
```
# Фазовый портрет
plot(sol_7, idxs=(1,2), lw=2, title="Модель консервативного гармонического осциллятора, где ω = $(p_7)",
    titlefontsize = 10,
    label = L"$\ddot{x} + ω^2x = 0$",
    xaxis="Координата",yaxis="Скорость",legend=true)
```

[263]:

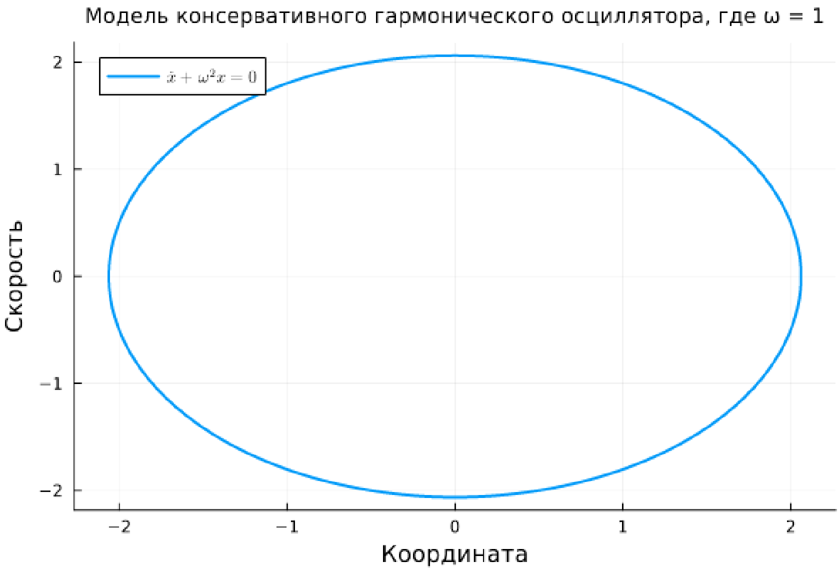


Рис. 2.51: Задание 6.4.7. Модель консервативного гармонического осциллятора (4)

[264]:
```
plot(sol_7, idxs=(1,2), lw=2, title="Модель консервативного гармонического осциллятора, где ω = $(p_7)",
    titlefontsize = 10,
    label = L"$\ddot{x} + ω^2x = 0$",
    xaxis="Координата",yaxis="Скорость",legend=true)
@gif for i ∈ 1:length(sol_7.u)
    scatter!((sol_7.u[i][1], sol_7.u[i][2]),
        ms = 3,
        color =:purple,
        legend=false)
end
```

```
[ Info: Saved animation to C:\Users\User\Documents\work\study\2023-2024\Statistical_Analysis_computer-practise\computer-practice\labs\lab06\report\report\tmp.gif
```

[264]:

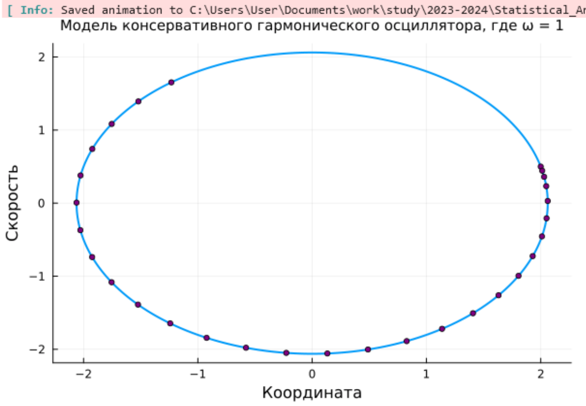


Рис. 2.52: Задание 6.4.7. Модель консервативного гармонического осциллятора (5)

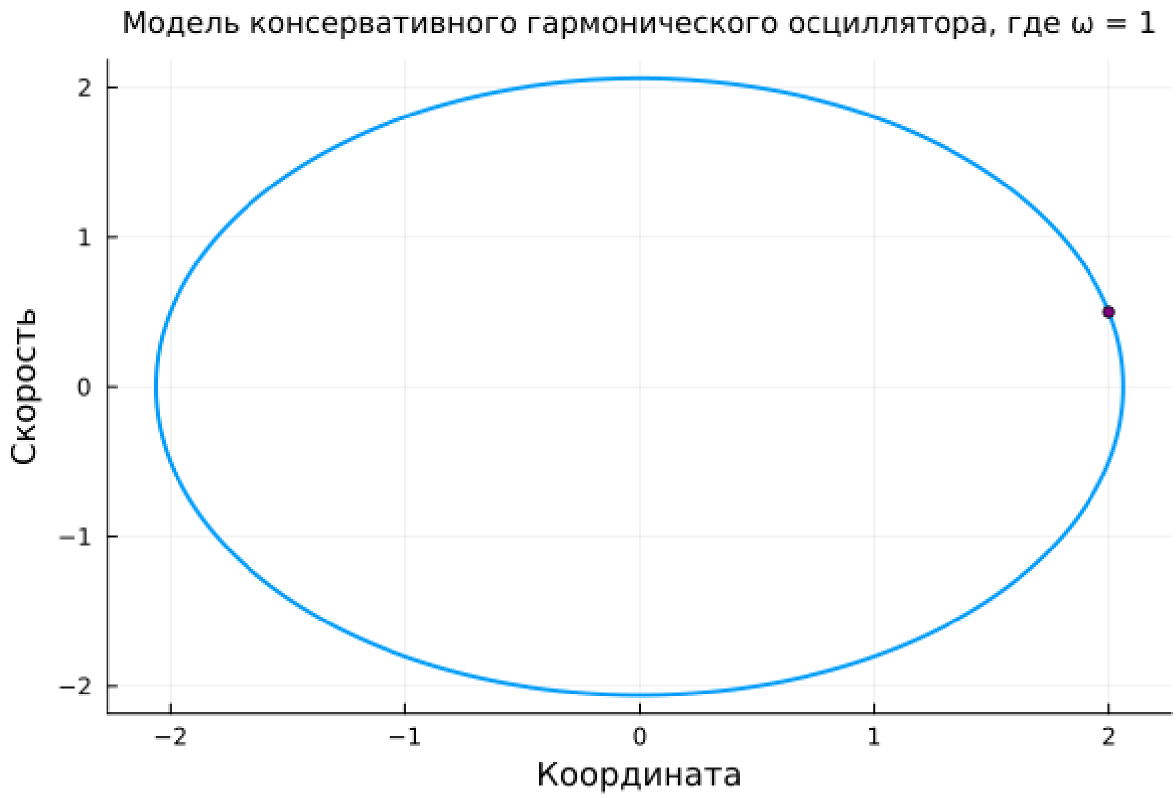
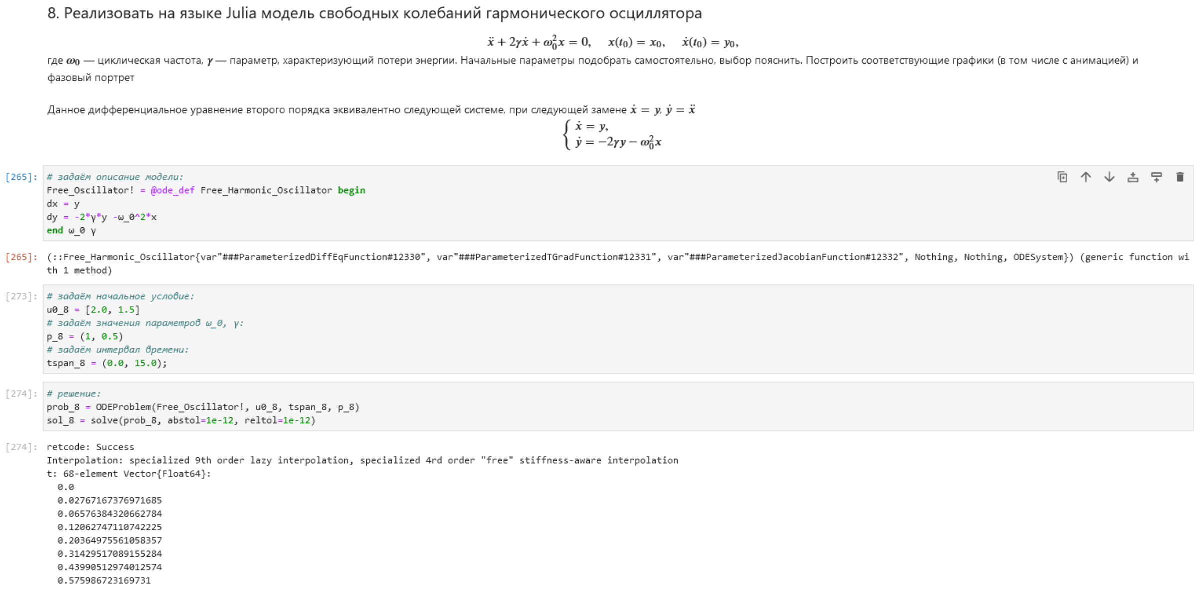


Рис. 2.53: Задание 6.4.7. Анимация модели консервативного гармонического осциллятора

8. Реализуем модель свободных колебаний гармонического осциллятора ([2.54-2.58]) с анимацией ([**gif:008?**]).

   Данное дифференциальное уравнение второго порядка эквивалентно следующей системе, при замене $\dot{x} = y$, $\dot{y} = \ddot{x}$ получим

$$\begin{cases} \dot{x} = y, \\ \dot{y} = -2\gamma y - \omega_0^2 x. \end{cases}$$



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

Данное дифференциальное уравнение второго порядка эквивалентно следующей системе, при следующей замене $\dot{x} = y$, $\dot{y} = \ddot{x}$

$$\begin{cases} \dot{x} = y, \\ \dot{y} = -2\gamma y - \omega_0^2 x \end{cases}$$

[265]:
```
# задаём описание модели:
Free_Oscillator! = @ode_def Free_Harmonic_Oscillator begin
dx = y
dy = -2*γ*y -ω_0^2*x
end ω_0 γ
```

[265]:
```
(::Free_Harmonic_Oscillator{var"###ParameterizedDiffEqFunction#12330", var"###ParameterizedTGradFunction#12331", var"###ParameterizedJacobianFunction#12332", Nothing, Nothing, ODESystem}) (generic function wi
th 1 method)
```

[273]:
```
# задаём начальное условие:
u0_8 = [2.0, 1.5]
# задаём значения параметров ω_0, γ:
p_8 = (1, 0.5)
# задаём интервал времени:
tspan_8 = (0.0, 15.0);
```

[274]:
```
# решение:
prob_8 = ODEProblem(Free_Oscillator!, u0_8, tspan_8, p_8)
sol_8 = solve(prob_8, abstol=1e-12, reltol=1e-12)
```

[274]:
```
retcode: Success
Interpolation: specialized 9th order lazy interpolation, specialized 4rd order "free" stiffness-aware interpolation
t: 68-element Vector{Float64}:
  0.0
  0.02767167376971685
  0.06576384320662784
  0.12062747110742225
  0.20364975561058357
  0.31429517089155284
  0.43990512974012574
  0.575986723169731
```

Рис. 2.54: Задание 6.4.8. Модель свободных колебаний гармонического осциллятора (1)

```
 1.0341836155144524
 1.2005149785476612
 1.371928437205861
 ⋮
11.619806913385595
11.921418413883345
12.226732925114058
12.536753259109174
12.852766544138023
13.176271521189184
13.508712511861063
13.850202600860118
14.197991411398606
14.549467666504146
14.904284087252652
15.0
u: 68-element Vector{Vector{Float64}}:
 [2.0, 1.5]
 [2.040174597084184, 1.4039200752992564]
 [2.091173159796163, 1.2742198299933483]
 [2.15607455963035, 1.0927632044646372]
 [2.235809317865667, 0.8305657205350633]
 [2.3094472273044913, 0.505139897143591]
 [2.3514610967112377, 0.1699570561494495]
 [2.352232137722313, -0.15135286044440752]
 [2.3083692827921056, -0.4457483477334786]
 [2.2195668588739172, -0.7043803952153457]
 [2.088945734749728, -0.9189463009818896]
 [1.9214831146237241, -1.0853973055163337]
 [1.724721170293343, -1.2014234643146011]
 ⋮
 [-0.009971020074893587, 0.002060181408535856]
 [-0.009034555120240895, 0.004004595845149711]
 [-0.007618758662172584, 0.005139850156343678]
 [-0.005941980884614819, 0.005568536821660773]
 [-0.004192539825928475, 0.005419149866138974]
 [-0.0025248304844454686, 0.004832853215649195]
 [-0.0010588132896883328, 0.00395439579451112]
 [0.00011783799156908241, 0.0029284187498433184]
 [0.0009545589826912164, 0.0018947780896308329]
 [0.0014525192775024075, 0.0009642120076003075]
 [0.0016538522454997494, 0.00020380119153903003]
 [0.0016650920851569242, 3.359351061003857e-5]
```

Рис. 2.55: Задание 6.4.8. Модель свободных колебаний гармонического осциллятора (2)

```
[275]: # строим график:
plot(sol_8, label = ["Координата колебаний" "Скорость колебаний"], color="black", ls=[:solid :dash],
    title="Модель свободных колебаний гармонического осциллятора, где ω = $(p_8[1]), γ = $(p_8[2])",
    titlefontsize = 8,
    xaxis="Время",yaxis="Число")
```

[275]:

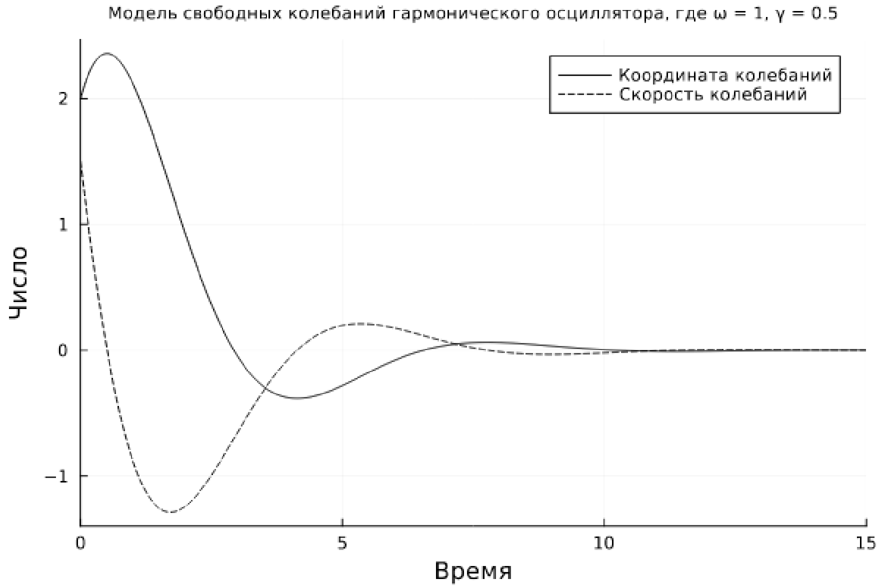


Рис. 2.56: Задание 6.4.8. Модель свободных колебаний гармонического осциллятора (3)

```
[276]: # Фазовый портрет
plot(sol_8, idxs=(1,2), lw=2, title="Модель свободных колебаний гармонического осциллятора, где ω = $(p_8[1]), γ = $(p_8[2])",
    titlefontsize = 8,
    label = L"$\ddot{x} + 2γ\dot{x} + ω^2x = 0$",
    xaxis="Координата",yaxis="Скорость",legend=true)
```

[276]:

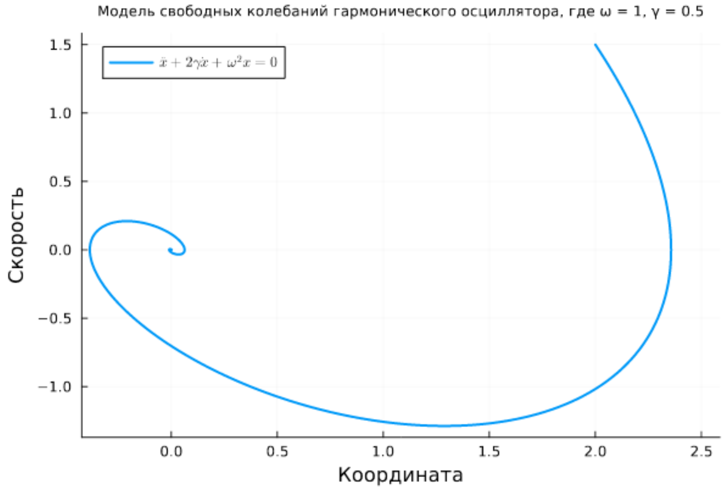


Рис. 2.57: Задание 6.4.8. Модель свободных колебаний гармонического осциллятора (4)

```
[277]: plot(sol_8, idxs=(1,2), lw=2, title="Модель свободных колебаний гармонического осциллятора, где ω = $(p_8[1]), γ = $(p_8[2])",
    titlefontsize = 8,
    label = L"$\ddot{x} + 2γ\dot{x} + ω^2x = 0$",
    xaxis="Координата",yaxis="Скорость",legend=true)
@gif for i ∈ 1:length(sol_8.u)
    scatter!((sol_8.u[i][1], sol_8.u[i][2]),
        ms = 3,
        color =:purple,
        legend=false)
end
```

```
[ Info: Saved animation to C:\Users\User\Documents\work\study\2023-2024\Statistical_Analysis_computer-practise\computer-practice\labs\lab06\report\report\tmp.gif
```

[277]:

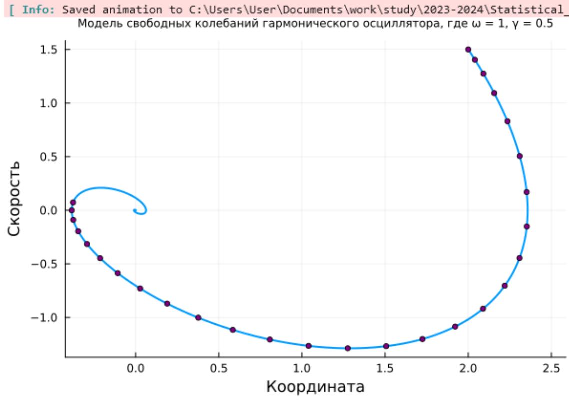


Рис. 2.58: Задание 6.4.8. Модель свободных колебаний гармонического осциллятора (5)

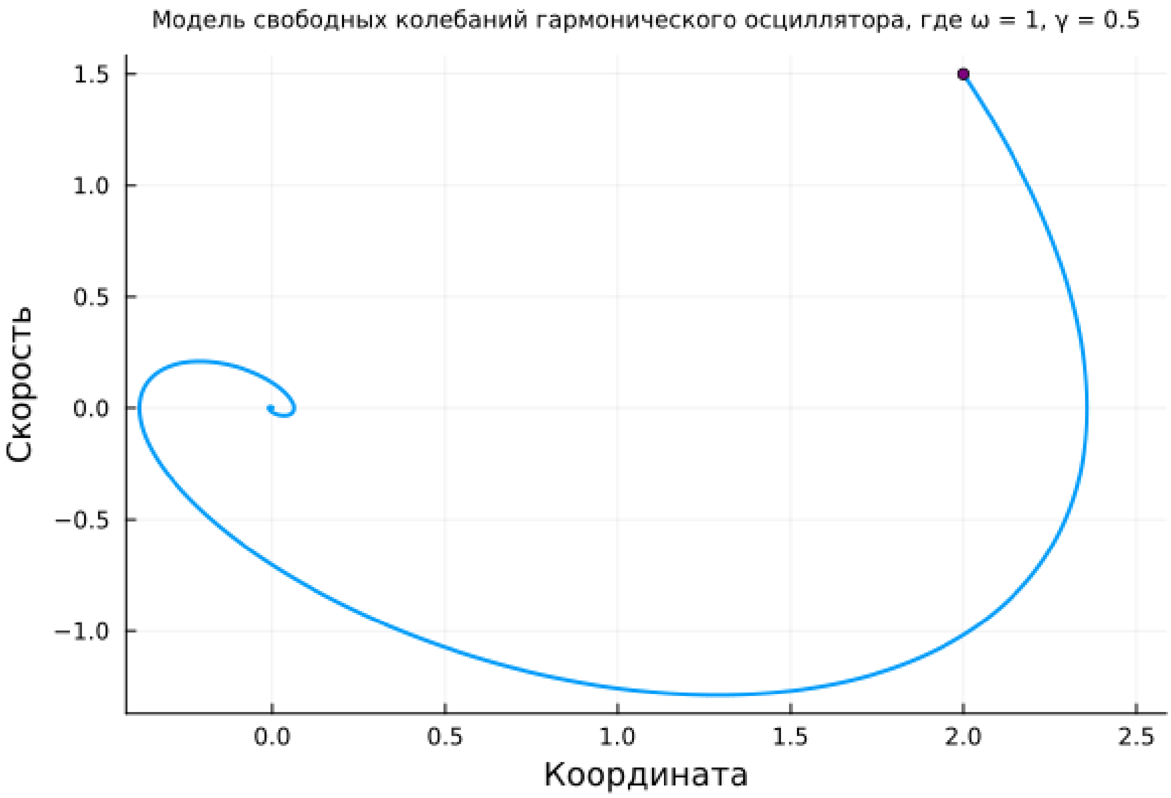


Рис. 2.59: Задание 6.4.8. Анимация модели свободных колебаний гармонического осциллятора

3 Выводы

В ходе выполнения лабораторной работы я освоил специализированные пакеты Julia для решения задач в непрерывном и дискретном времени.

Список литературы

1. Королькова А. В., Кулябов Д. С. Лабораторная работа № 6. Решение моделей в непрерывном и дискретном времени [Электронный ресурс]. RUDN, 2023. URL: https://esystem.rudn.ru/pluginfile.php/2231357/mod_resource/content/2/006-lab_f_du.pdf.