Отчет по лабораторной работе 6

Петрушов Дмитрий, 1032212287

Содержание

1 Цель работы **5**

2 Выполнение лабораторной работы **6**

2.1 Решение обыкновенных дифференциальных уравнений 6

2.2 Модель экспоненциального роста 6

2.3 Система Лоренца 8

2.4 Модель Лотки–Вольтерры 10

2.5 Самостоятельное выполнение 12

3 Вывод **29**

Список литературы **30**

Список иллюстраций

2.1 График модели экспоненциального роста 7
2.2 График модели экспоненциального роста (задана точность решения) 8
2.3 Аттрактор Лоренца 9
2.4 Аттрактор Лоренца (интерполяция отключена) 10
2.5 Модель Лотки–Вольтерры: динамика изменения численности популяций 11
2.6 Модель Лотки–Вольтерры: фазовый портрет 12
2.7 Решение задания №1 13
2.8 график №1 14
2.9 Решение задания №2 15
2.10 График №2 16
2.11 Решение задания №3 17
2.12 График №3 18
2.13 Решение задания №4 19
2.14 График №4 20
2.15 Решение задания №5 21
2.16 График №5 22
2.17 Решение задания №6 23
2.18 График №6 24
2.19 Решение задания №7 25
2.20 График №7 26
2.21 Решение задания №8 27
2.22 График №8 28

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 Решение обыкновенных дифференциальных уравнений

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

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

2.2 Модель экспоненциального роста

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

Численное решение в Julia будет иметь следующий вид, а также график, соответствующий полученному решению (рис. [2.1]):

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

```
   Resolving package versions...
  No Changes to `~/.julia/environments/v1.10/Project.toml`
  No Changes to `~/.julia/environments/v1.10/Manifest.toml`
```

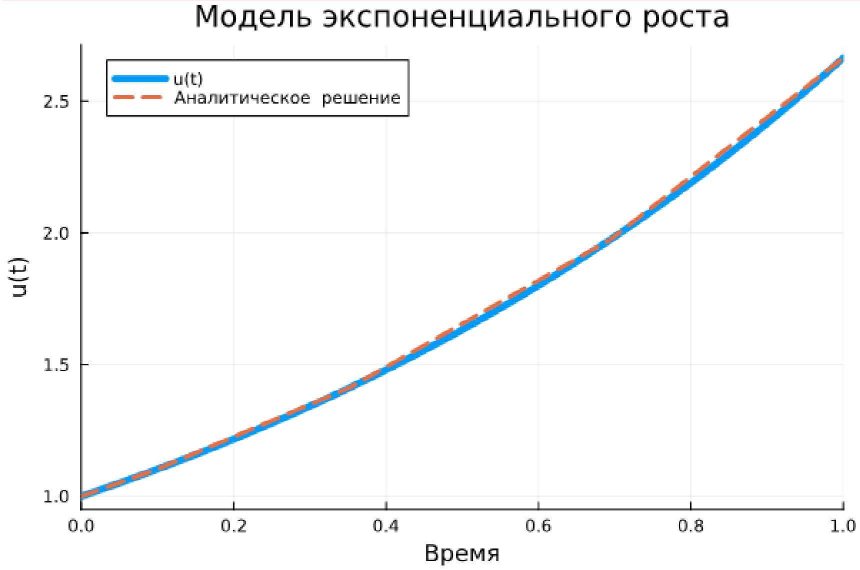

Рис. 2.1: График модели экспоненциального роста

При построении одного из графиков использовался вызов sol.t, чтобы захватить массив моментов времени. Массив решений можно получить, воспользовавшись sol.u.

Если требуется задать точность решения, то можно воспользоваться параметрами abstol (задаёт близость к нулю) и reltol (задаёт относительную точность). По умолчанию эти параметры имеют значение abstol = 1e-6 и reltol = 1e-3.

Для модели экспоненциального роста (рис. [2.2]):

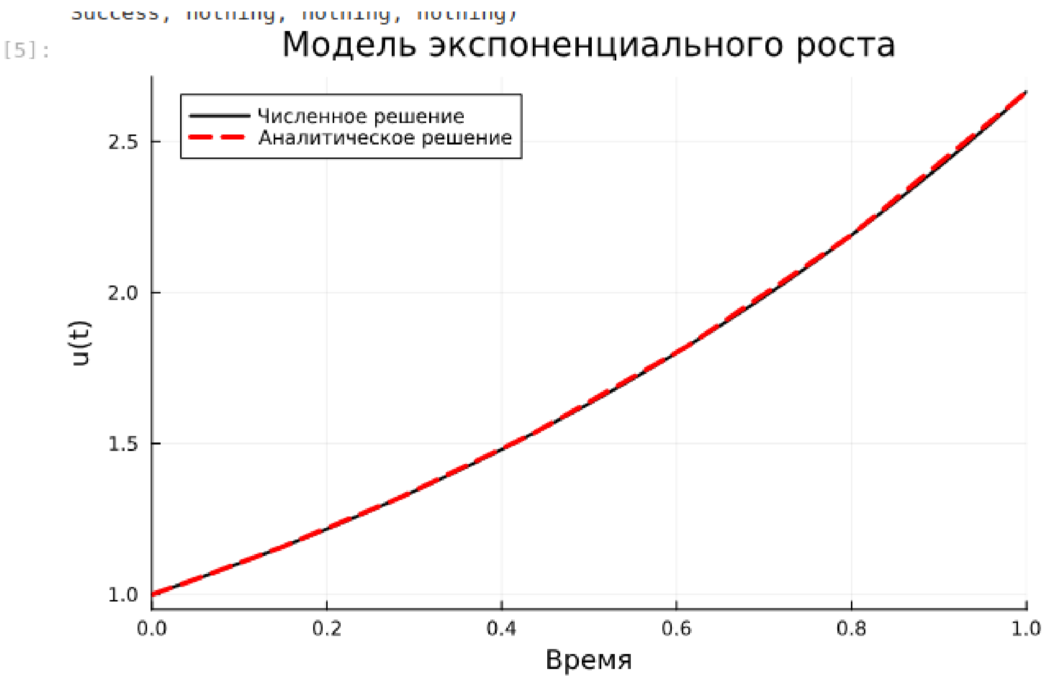

Рис. 2.2: График модели экспоненциального роста (задана точность решения)

2.3 Система Лоренца

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

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 (рис. [2.3]):

```
# подключаем необходимые пакеты:
Pkg.add("Plots")
using Plots
plot(sol, vars=(1,2,3), lw=2, title="Аттрактор Лоренца", xaxis="x",yaxis="y", zaxis="z",legend=false)
```

```
   Resolving package versions...
  No Changes to `~/.julia/environments/v1.10/Project.toml`
  No Changes to `~/.julia/environments/v1.10/Manifest.toml`
┌ Warning: To maintain consistency with solution indexing, keyword argument vars will be removed in a fut
ure version. Please use keyword argument idxs instead.
│   caller = ip:0x0
└ @ Core :-1
```

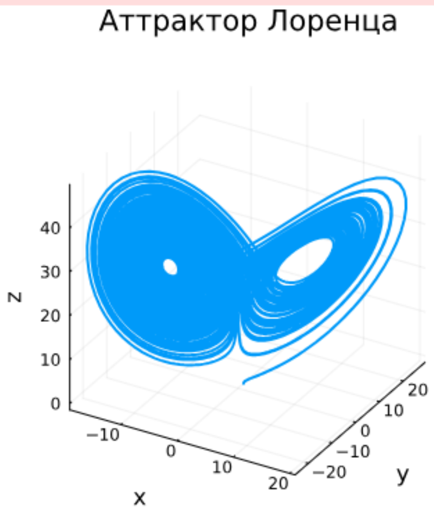

Рис. 2.3: Аттрактор Лоренца

Можно отключить интерполяцию (рис. [2.4]):

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

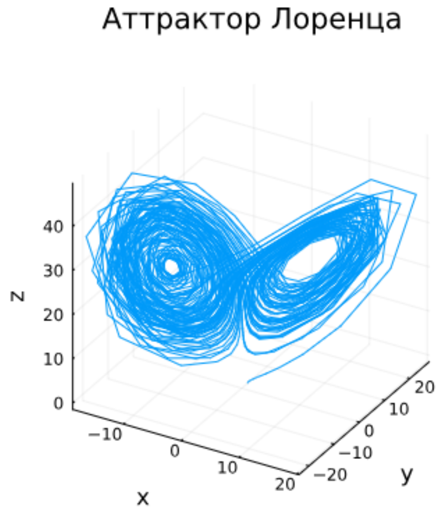

Рис. 2.4: Аттрактор Лоренца (интерполяция отключена)

2.4 Модель Лотки–Вольтерры

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

Численное решение в Julia будет иметь следующий вид (рис. [2.5]):

```
[1.8164214705302744, 4.064057991958618]
[1.146502825635759, 2.791173034823897]
[0.955798652853089, 1.623563316340748]
[1.0337581330572414, 0.9063703732075853]
```

```
color="black", ls=[:solid :dash], title="Модель Лотки - Вольтерры", xaxis="Время",yaxis="Размер популяции'
```

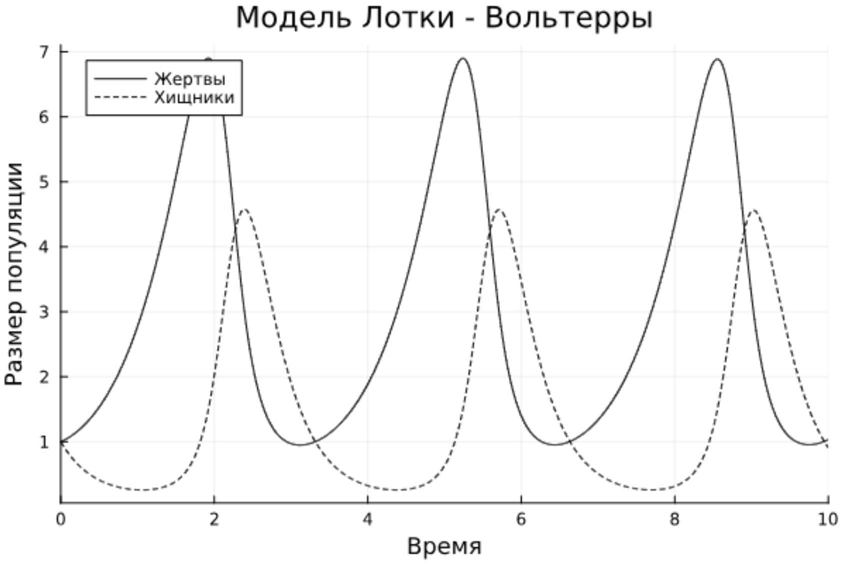

Рис. 2.5: Модель Лотки–Вольтерры: динамика изменения численности популяций

Фазовый портрет (рис. [2.6]):

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

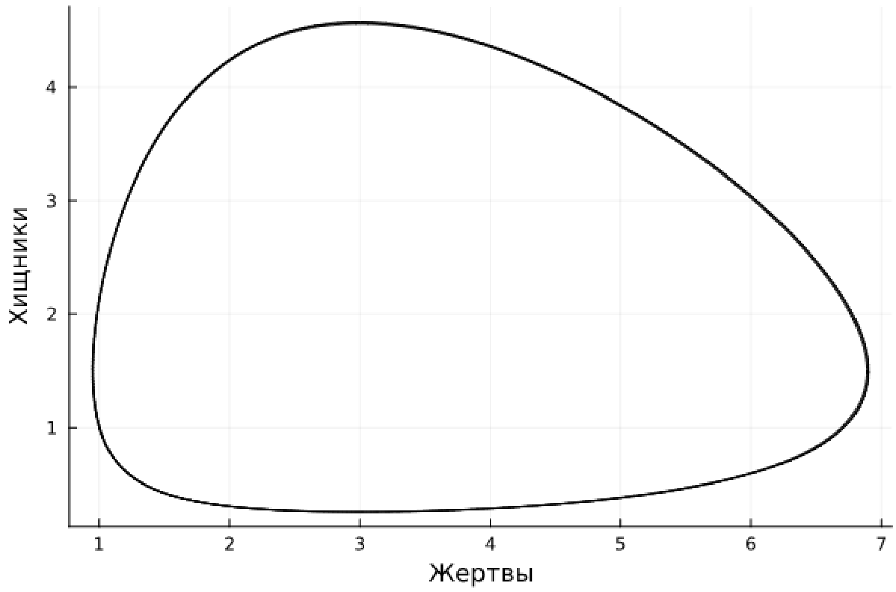

Рис. 2.6: Модель Лотки–Вольтерры: фазовый портрет

2.5 Самостоятельное выполнение

Выполнение задания №1 (рис. [2.7]):

```
using ParameterizedFunctions, DifferentialEquations, Plots;

# задаём описание модели:
lv! = @ode_def Malthus begin
    dx = a*x
end a

# задаём начальное условие:
u0 = [2]
# задаём значения параметров:
b = 3.0
c = 1.0
p = (b - c)
# задаём интервал времени:
tspan = (0.0, 3.0)

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

Рис. 2.7: Решение задания №1

График №1 (рис. [2.8]):

```
animate(sol, fps=7, "Malthus.gif", label = "Численность изолированной популяции x(t)", colo
    xaxis="Время", yaxis="Размер изолированной популяции")
```

```
┌ Info: Saved animation to
│   fn = /Users/anastasia/Desktop/Учеба универ/Практикум по телекоммуникациям/6 lab/Malthus
└ @ Plots /Users/anastasia/.julia/packages/Plots/YdauZ/src/animation.jl:104
```

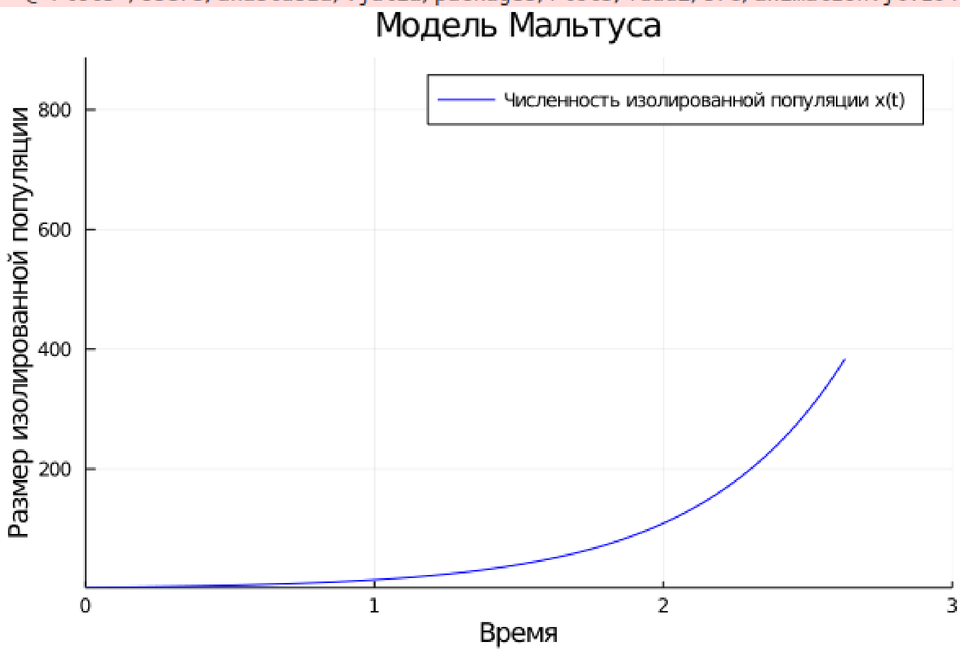

Рис. 2.8: график №1

Выполнение задания №2 (рис. [2.9]):

```
# задаём описание модели:
lv! = @ode_def Logistic_population begin
    dx = r*x*(1 - x/k)
    end r k

# задаём начальное условие:
u0 = [1.0]

# задаём значения параметров:
p = (0.9, 20)
# задаём интервал времени:
tspan = (0.0, 10.0)

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

Рис. 2.9: Решение задания №2

График №2 (рис. [2.10]):

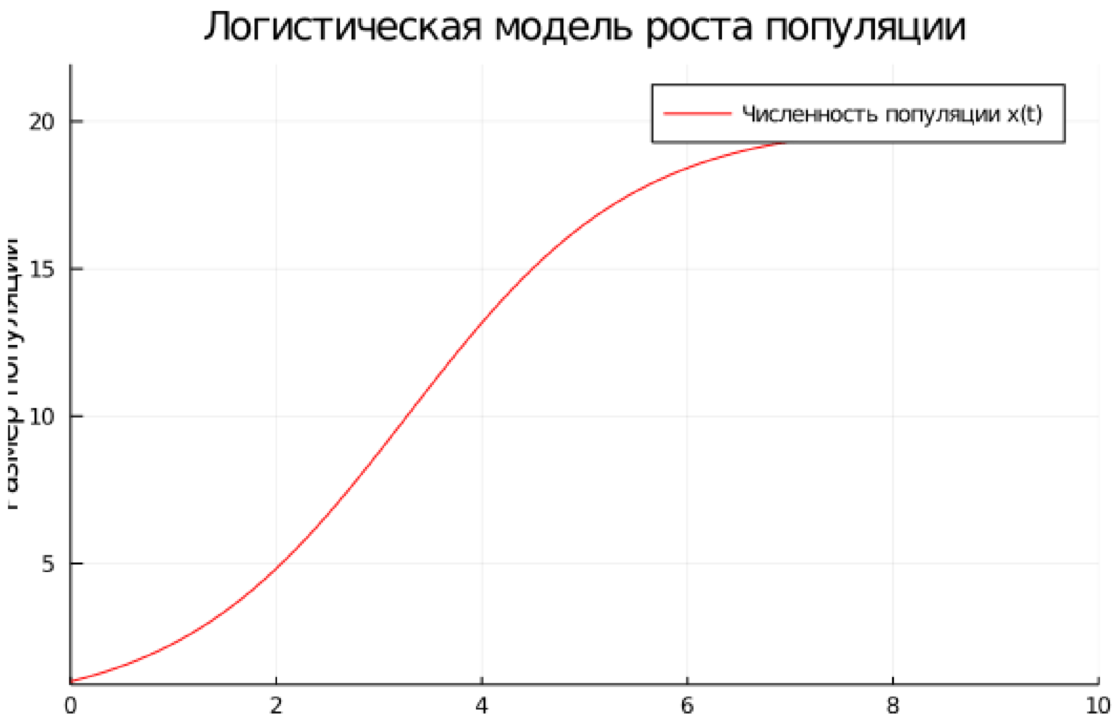

Рис. 2.10: График №2

Выполнение задания №3 (рис. [2.11]):

```
# задаём описание модели:
lv! = @ode_def SIR begin
ds = - b*i*s
di = b*i*s - v*i
dr = v*i
end b v

# задаём начальное условие:
u0 = [1.0, 0.1, 0]
# задаём значения параметров:
p = (0.25, 0.05)
# задаём интервал времени:
tspan = (0.0, 100.0)

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

Рис. 2.11: Решение задания №3

График №3 (рис. [2.12]):

```
animate(sol, fps=7, "SIR.gif", label = ["Восприимчивые" "Инфицированные" "Переболевшие"],
    color=["blue" "red" "green"], ls=[:solid :dash :dot], title="Модель SIR",
    xaxis="Время",yaxis="Размер популяции")
```

```
┌ Info: Saved animation to
│   fn = /Users/anastasia/Desktop/Учеба универ/Практикум по телекоммуникациям/6 lab/SIR.gif
└ @ Plots /Users/anastasia/.julia/packages/Plots/YdauZ/src/animation.jl:104
```

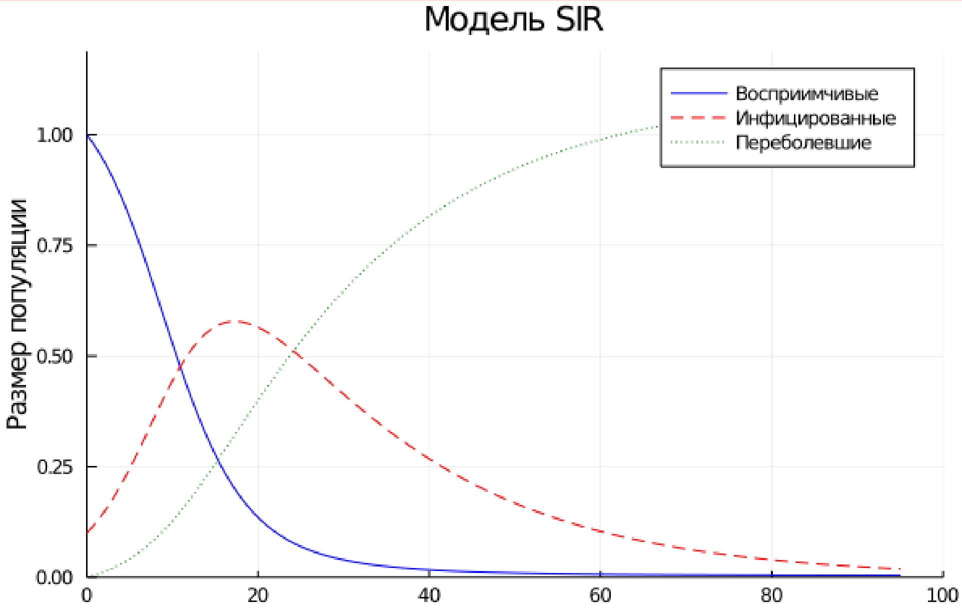

Рис. 2.12: График №3

Выполнение задания №4 (рис. [2.13]):

```
M = 1.0

# задаём описание модели:
lv! = @ode_def SEIR begin
ds = -(β/M)*s*i
de = (β/M)*s*i - δ*e
di = δ*e - γ*i
dr = γ*i
end β γ δ

initialInfect = 0.1
# задаём начальное условие:
u0 = [(M - initialInfect), 0.0, initialInfect, 0.0]
# задаём значения параметров:
p = (0.6, 0.2, 0.1)
# задаём интервал времени:
tspan = (0.0, 100.0)

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

Рис. 2.13: Решение задания №4

График №4 (рис. [2.14]):

```
animate(sol, fps=7, "SEIR.gif", label = ["Восприимчивые" "Контактные" "Инфицированные"
    color=["blue" "orange" "red" "green"], ls=[:solid :dash :dot :dashdot],
    title="Модель SEIR",
    xaxis="Время",yaxis="Размер популяции")
```

```
┌ Info: Saved animation to
│   fn = /Users/anastasia/Desktop/Учеба универ/Практикум по телекоммуникациям/6 lab/SE
└ @ Plots /Users/anastasia/.julia/packages/Plots/YdauZ/src/animation.jl:104
```

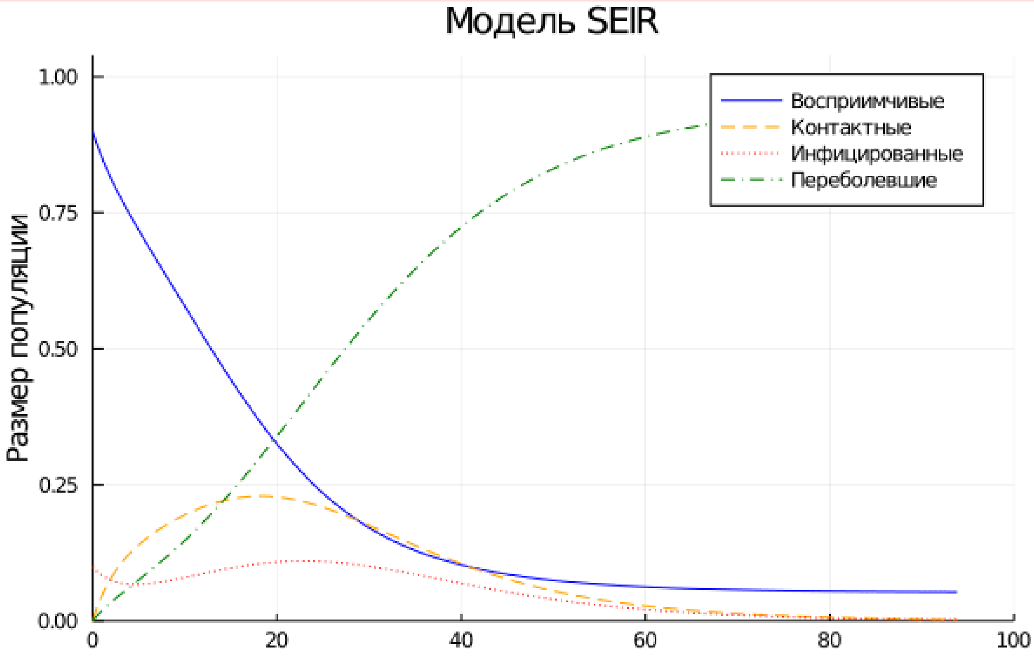

Рис. 2.14: График №4

Выполнение задания №5 (рис. [2.15]):

```julia
using DifferentialEquations, Plots, ParameterizedFunctions, LaTeXStrings

# задаём значнения параметров:
a, c, d = 2, 1, 5

# задаем функцию для дискретной модели
next(x1, x2) = [(a*x1*(1 - x1) - x1*x2), (-c*x2 + d*x1*x2)]

# рассчитываем точку равновесия
balancePoint = [(1 + c)/d ,(d*(a - 1)-a*(1 + c))/d]

# задаём начальное условие:
u0 = [0.8, 0.05]
modelingTime = 100

simTrajectory = Array{Union{Nothing, Array}}(nothing, modelingTime)

for t in 1:modelingTime
    simTrajectory[t] = []
    if(t == 1)
        simTrajectory[t] = u0
    else
        simTrajectory[t] = next(simTrajectory[t-1]...)
    end
end
```

Рис. 2.15: Решение задания №5

График №5 (рис. [2.16]):

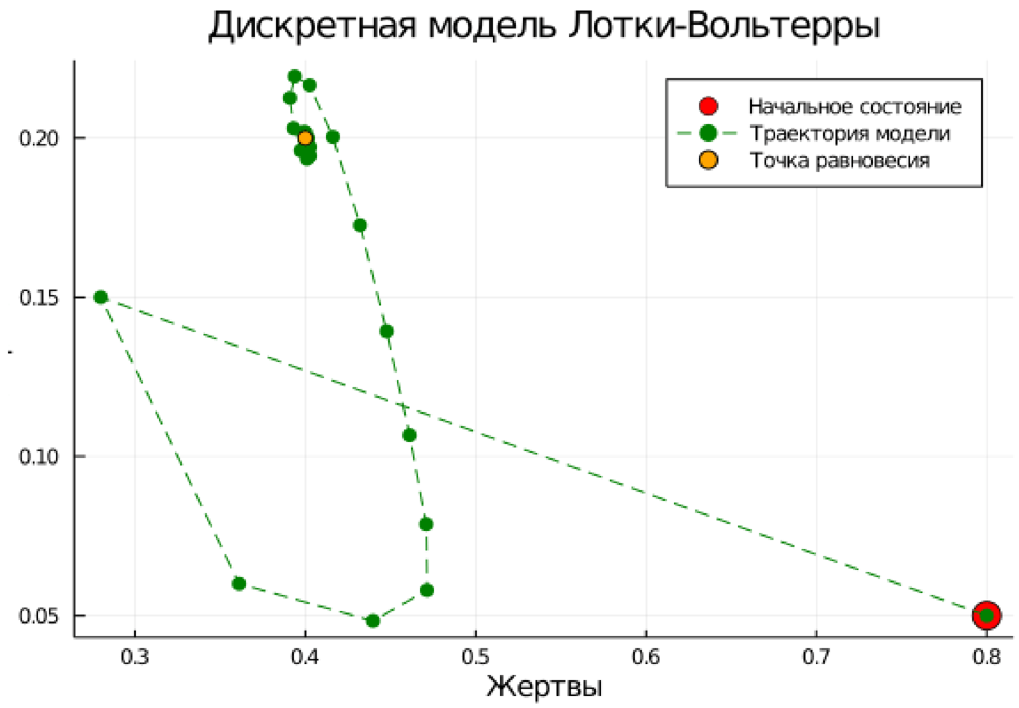

Рис. 2.16: График №5

Выполнение задания №6 (рис. [2.17]):

```
# задаём описание модели:
lv! = @ode_def CompetitiveSelectionModel begin
dx = a*x - b*x*y
dy = a*y - b*x*y
    end a b

# задаём начальное условие:
u0 = [1.0, 1.4]
# задаём значения параметров:
p = (0.5, 0.2)
# задаём интервал времени:
tspan = (0.0, 10.0)

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

Рис. 2.17: Решение задания №6

График №6 (рис. [2.18]):

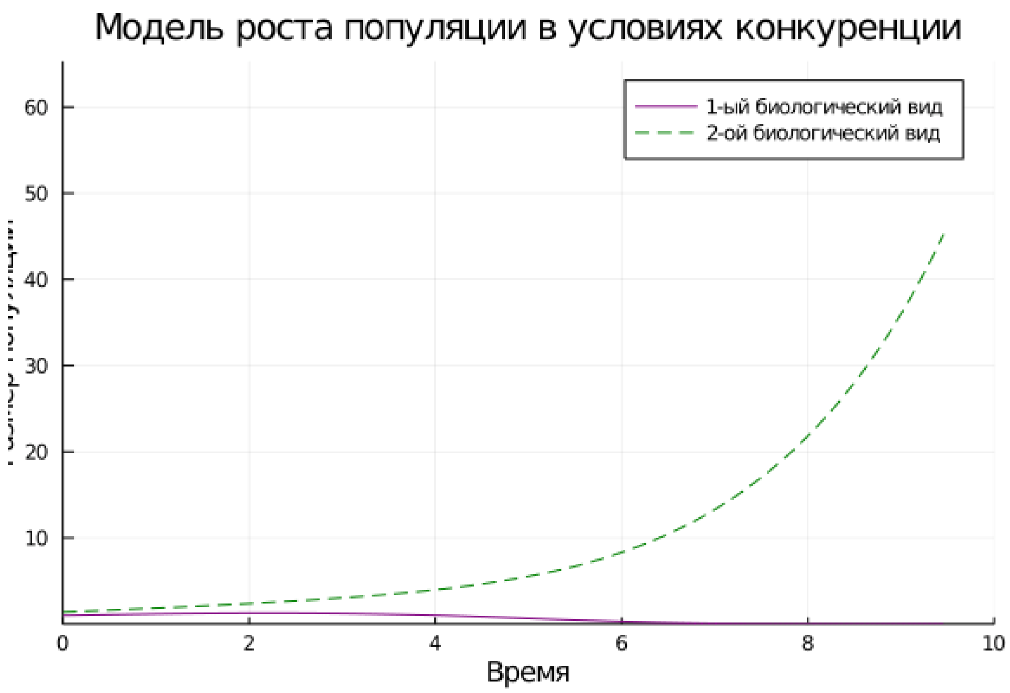

Рис. 2.18: График №6

Выполнение задания №7 (рис. [2.19]):

```
# задаём описание модели:
lv! = @ode_def classicOscillator begin
dx = y
dy = -(w0^2)*x
end w0

# задаём начальное условие:
u0 = [1.0, 1.0]
# задаём значения параметров:
p = (2.0)
# задаём интервал времени:
tspan = (0.0, 10.0)

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

Рис. 2.19: Решение задания №7

График №7 (рис. [2.20]):

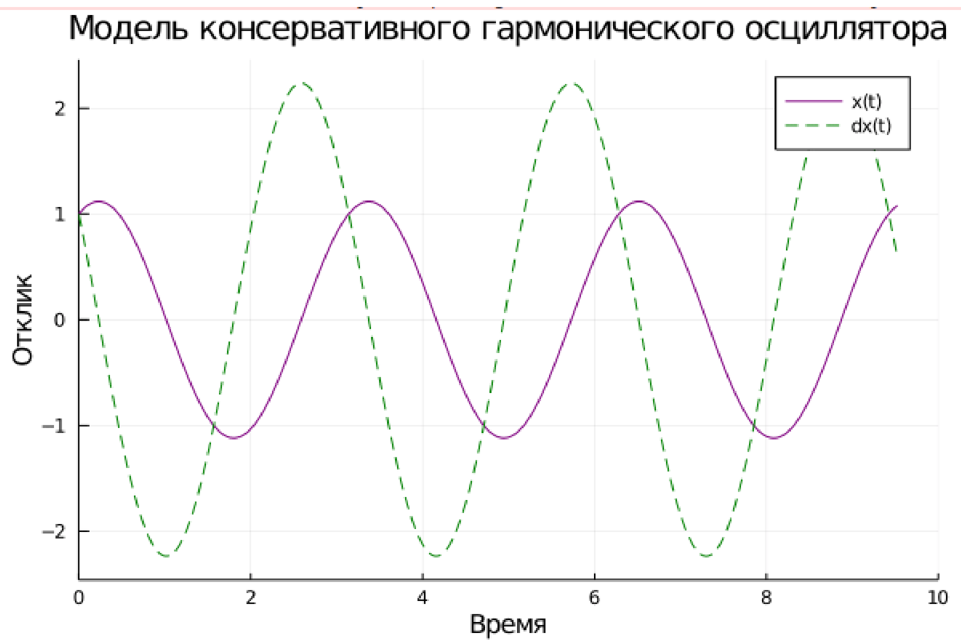

Рис. 2.20: График №7

Выполнение задания №8 (рис. [2.21]):

```
# задаём описание модели:
lv! = @ode_def Oscillator begin
dx = y
dy = -2*v*y - (w0^2)*x
end v w0

# задаём начальное условие:
u0 = [0.5, 1.0]
# задаём значения параметров:
p = (0.5, 2.0)
# задаём интервал времени:
tspan = (0.0, 10.0)

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

Рис. 2.21: Решение задания №8

График №8 (рис. [2.22]):

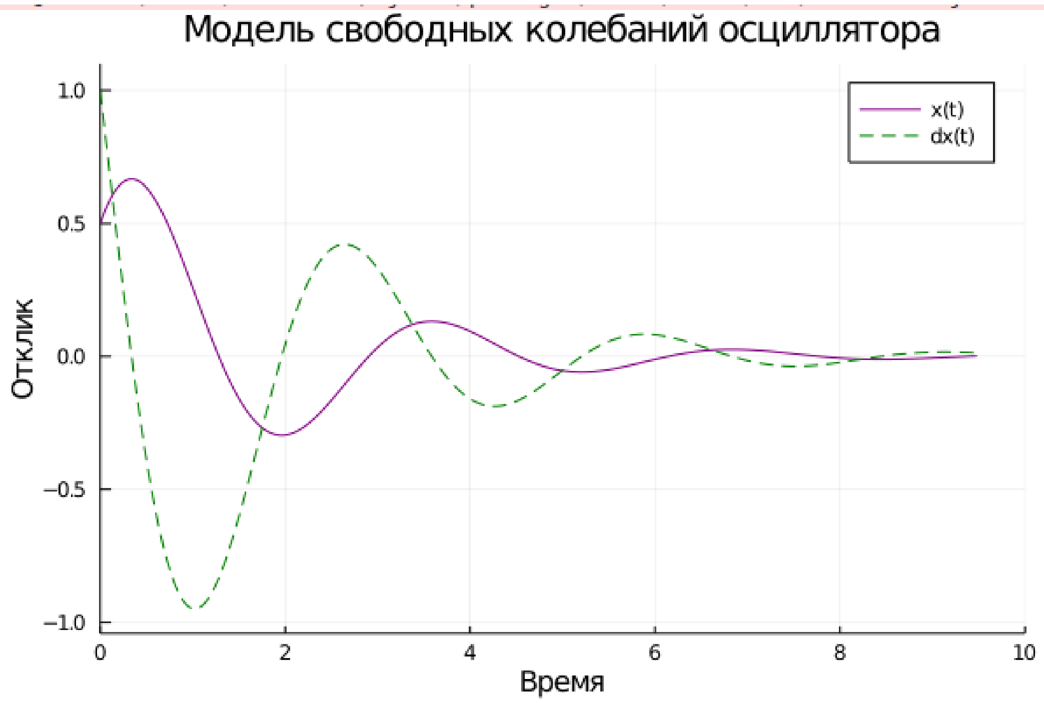

Рис. 2.22: График №8

3 Вывод

В ходе выполнения лабораторной работы были освоены специализированные пакеты для решения задач в непрерывном и дискретном времени.

Список литературы