Лабораторная работа 6

Петрушов Дмитрий, 1032212287

2024 г.

Российский университет дружбы народов, Москва, Россия

Цель работы

Цель работы

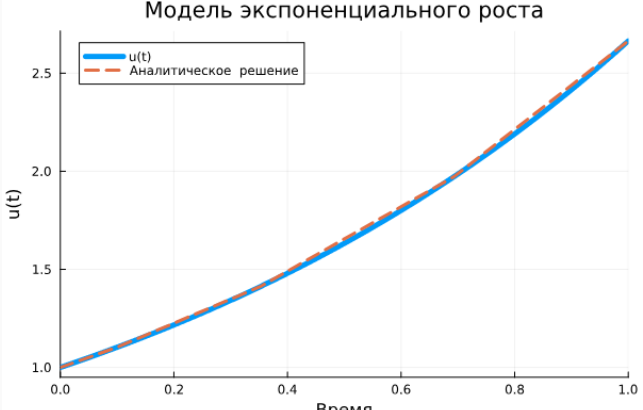
Основной целью работы является освоение специализированных пакетов для решения задач в непрерывном и дискретном времени.

Выполнение 1 части

Модель экспоненциального роста

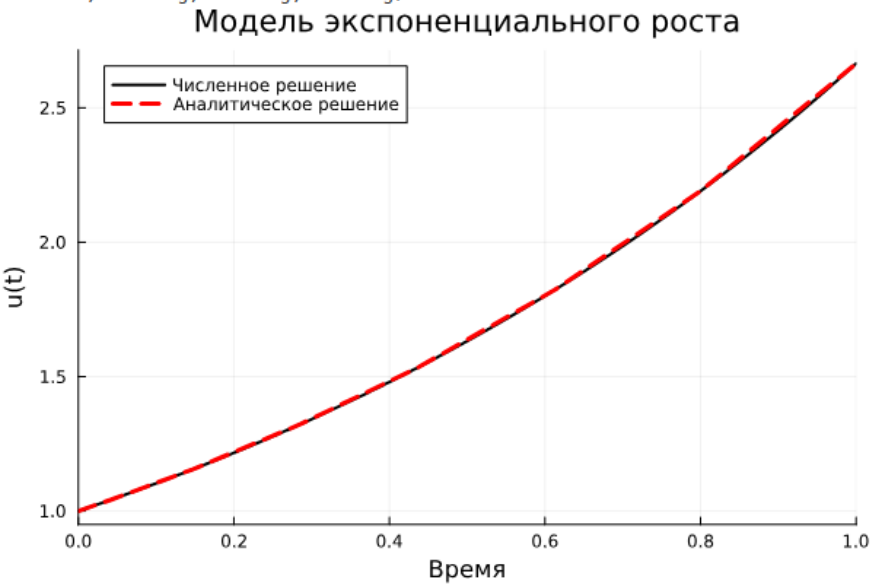
```
# подключаем необходимые пакеты:
Pkg.add("Plots")
using Plots
# строим графики:
plot(sol, linewidth=5,title="Модель экспоненциального роста", xaxis="Время",yaxis="u(t)",label="u(t)")
plot!(sol.t, t->1.0*exp(a*t),lw=3,ls=:dash,label="Аналитическое  решение")
```

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



```
Success, nothing, nothing, nothing)
```

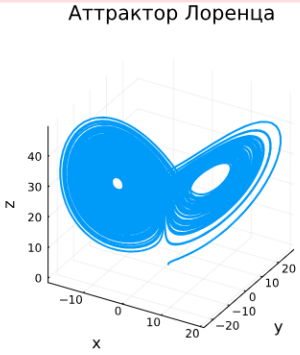
[5]:



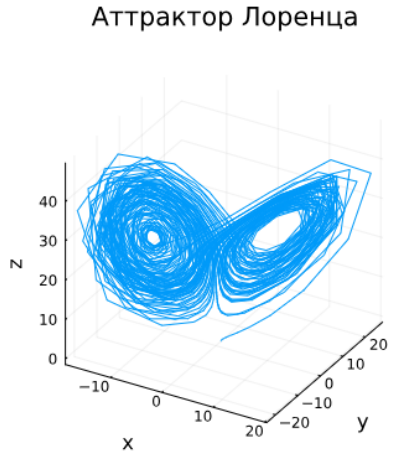
Система Лоренца

```
# 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
```



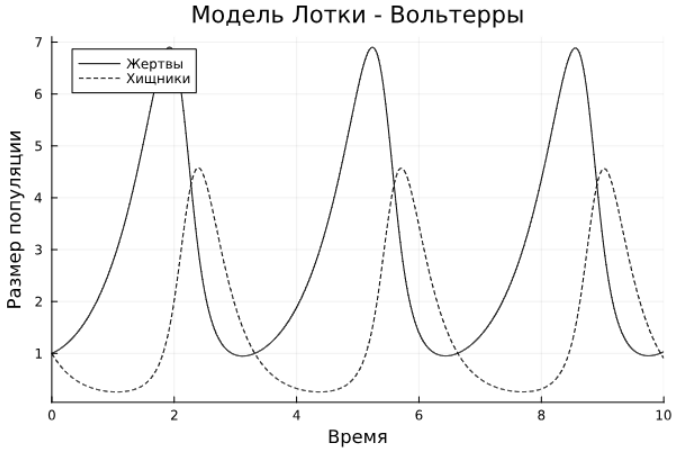
```
# отключаем интерполяцию:
plot(sol,vars=(1,2,3),denseplot=false, lw=1, title="Аттрактор Лоренца", xaxis="x",yaxis="y", zaxis="z",leg
```



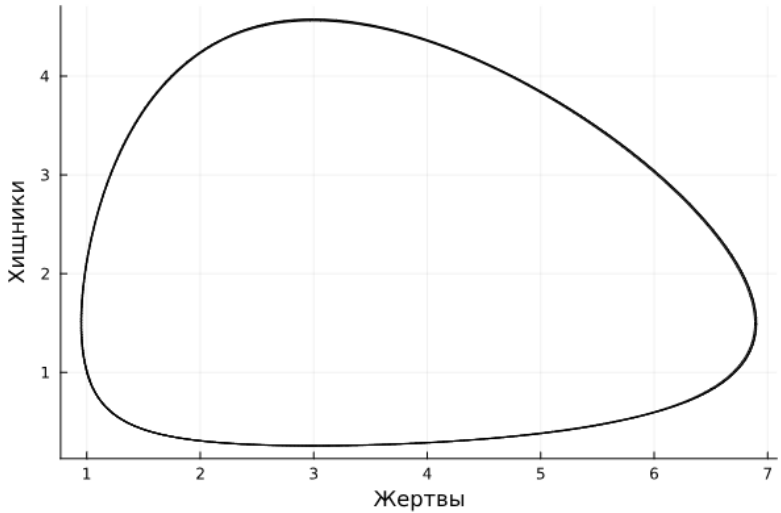
Модель Лотки–Вольтерры

```
[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"
```



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



Самостоятельное выполнение

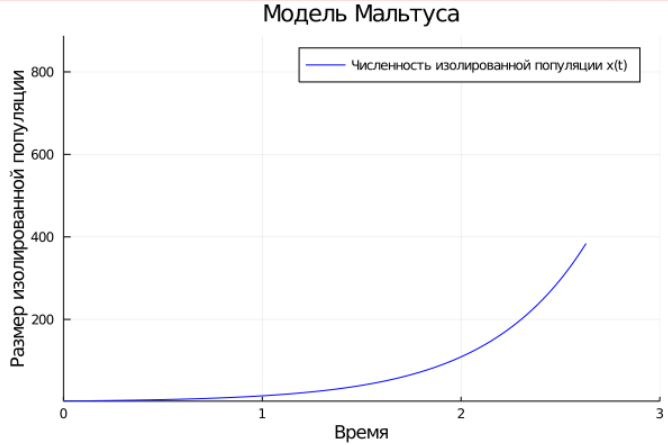
Выполнение задания №1

```
]: 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)
```

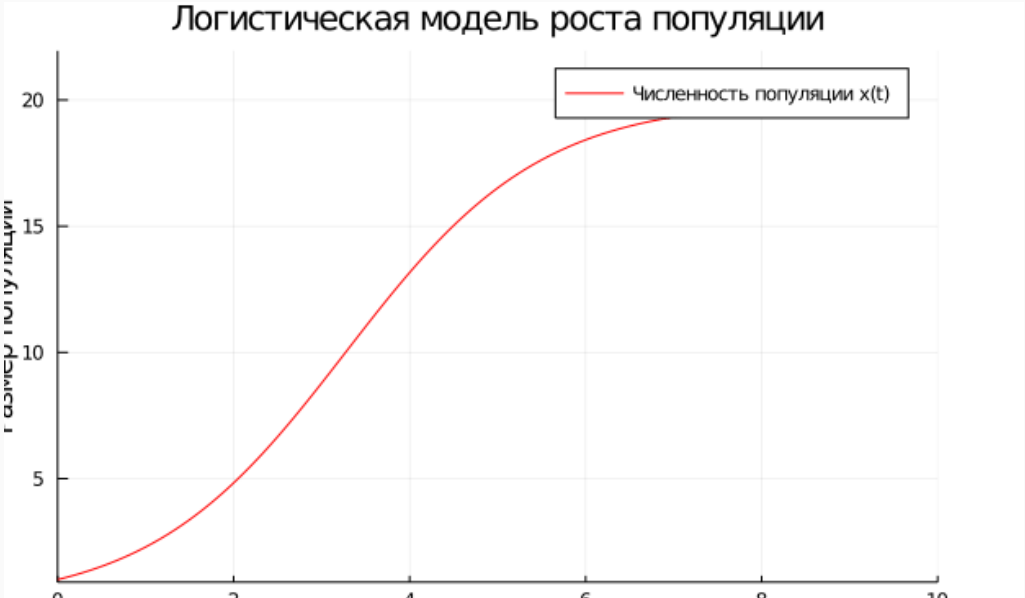
График №1

```
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
```



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

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

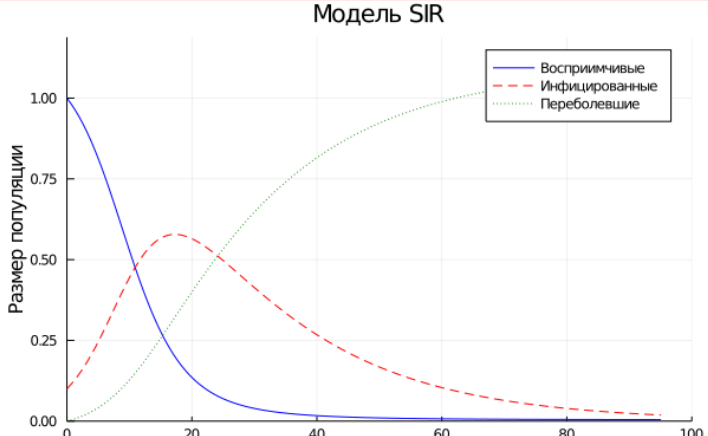

Логистическая модель роста популяции
Численность популяции x(t)
20
15
10
5
0
2
4
6
8
10

```
# 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 знанчения параметров:
p = (0.25, 0.05)
# задаём интервал времени:
tspan = (0.0, 100.0)

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

График №3

```
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
```



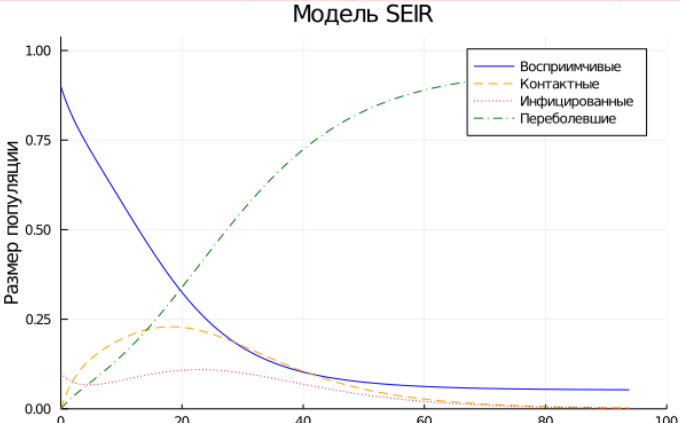
Выполнение задания №4

```
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 знанчения 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)
```

График №4

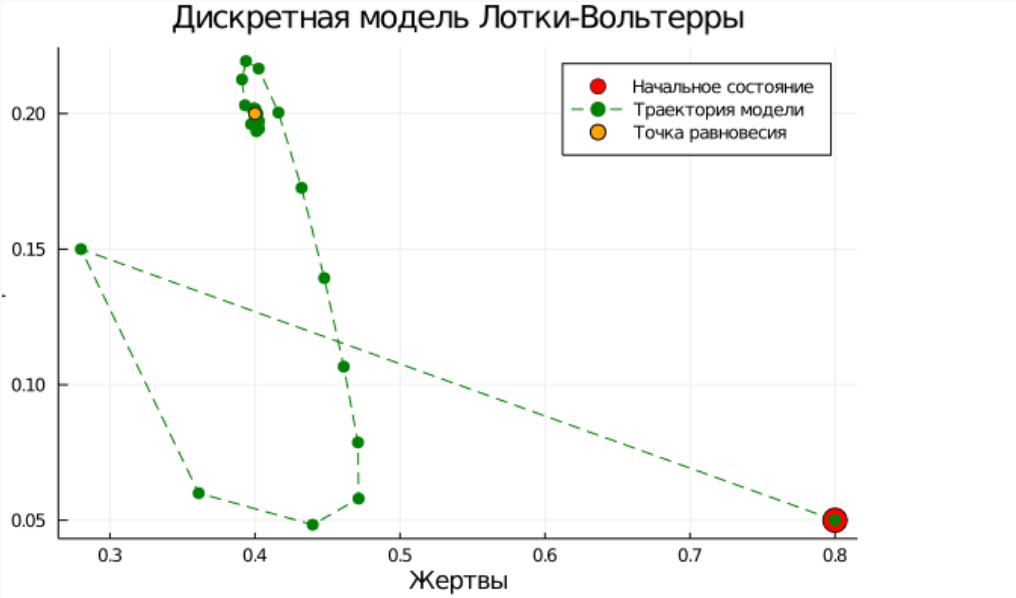
```
animate(sol, fps=7, "SEIR.gif", label = ["Восприимчивые" "Контактные" "Инфицированные"
    color=["blue" "orange" "red" "green"], ls=[:solid :dash :dot :dashdot],
    title="Модель SEIR",
    xaxis="Время",yaxis="Размер популяции")
```

```
┌ 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
```

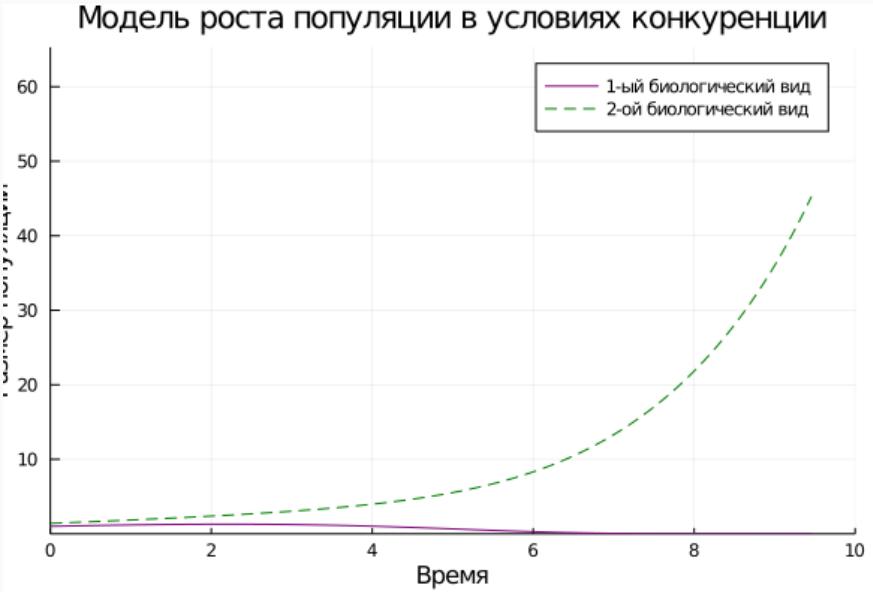


Выполнение задания №5

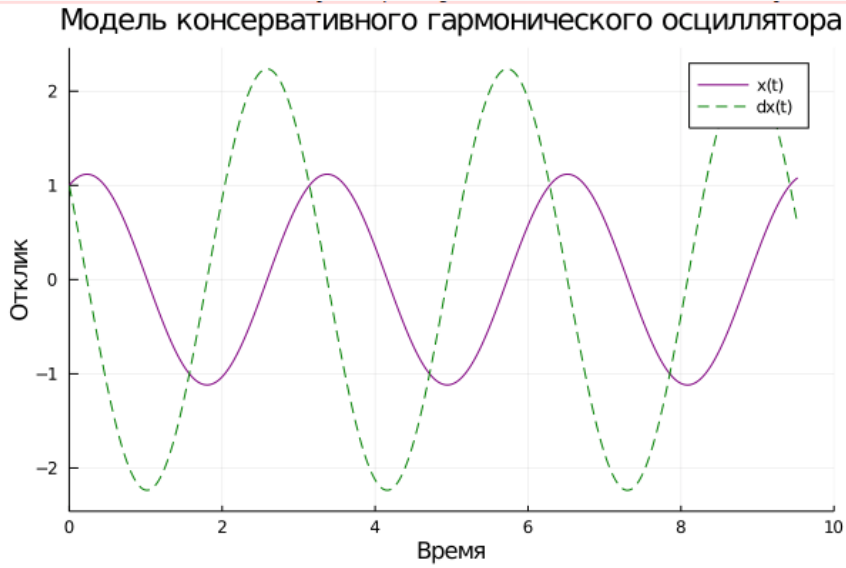
```julia
using DifferentialEquations, Plots, ParameterizedFunctions, LaTeXStrings

# задаём знанчения 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
```


Дискретная модель Лотки-Вольтерры
Начальное состояние
Траектория модели
Точка равновесия
0.20
0.15
0.10
0.05
0.3
0.4
0.5
0.6
0.7
0.8
Жертвы

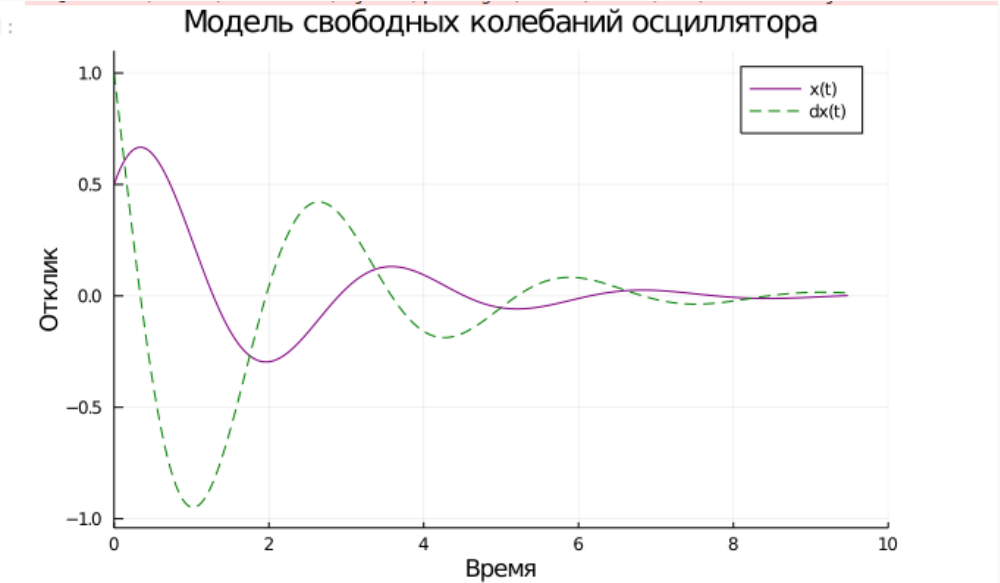
```
# задаём описание модели:
lv! = @ode_def CompetitiveSelectionModel begin
dx = a*x - b*x*y
dy = a*y - b*x*y
    end a b

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

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


Модель роста популяции в условиях конкуренции
1-ый биологический вид
2-ой биологический вид
60
50
40
30
20
10
0
2
4
6
8
10
Время

```
# 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 знанчения параметров:
p = (2.0)
# задаём интервал времени:
tspan = (0.0, 10.0)

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


Модель консервативного гармонического осциллятора
x(t)
dx(t)
Отклик
Время
2
1
0
−1
−2
0
2
4
6
8
10

```
# 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 знанчения параметров:
p = (0.5, 2.0)
# задаём интервал времени:
tspan = (0.0, 10.0)

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


Модель свободных колебаний осциллятора
x(t)
dx(t)
Отклик
Время
1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
8
10

Выводы

Выводы

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