

# Лабораторная работа №6

## Решение моделей в непрерывном и дискретном времени

---

Ким Реачна<sup>1</sup>

13 декабря, 2023, Москва, Россия

<sup>1</sup>Российский Университет Дружбы Народов

# Цели и задачи

---

# Цель лабораторной работы

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

1. Используя Jupyter Lab, повторите примеры.
2. Выполните задания для самостоятельной работы.

# Процесс выполнения лабораторной работы

---

# Модель экспоненциального роста

```
[2]: using DifferentialEquations
      # 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)

      # решение:
      prob = ODEProblem(f,u0,tspan)
      sol = solve(prob)

[2]: 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. 1: Численное решение модель экспоненциального роста

# Модель экспоненциального роста

```
[3]: # подключаем необходимые пакеты:  
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")
```

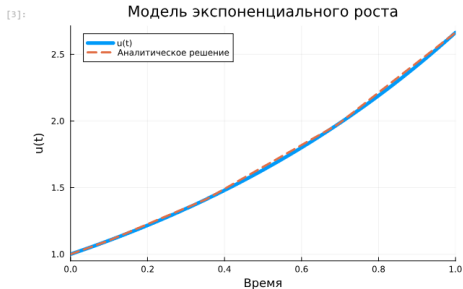


Рис. 2: График модель экспоненциального роста

# Система Лоренца

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

Resolving package versions...
No Changes to 'C:\Users\Reachna\.julia\environments\v1.9\Project.toml'
No Changes to 'C:\Users\Reachna\.julia\environments\v1.9\Manifest.toml'

[7]: using DifferentialEquations, Plots;
# задаём описание модели:
function lorenz!(du,u,p,t)
    o,p,β = p
    du[1] = o*(u[2]-u[1])
    du[2] = u[1]*(p-u[3]) - u[2]
    du[3] = u[1]*u[2] - β*u[3]
end
# задаём начальные условия:
u0 = [1.0,0.0,0.0]
# задаём значения параметров:
p = (10,28,8/3)
# задаём интервал времени:
tspan = (0.0,100.0)
# решение:
prob = ODEProblem(lorenz!,u0,tspan,p)
sol = solve(prob)

[7]: 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
 0.00039246460531993154
 0.0052624077544510573
 0.009858075535317072
 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
 00 406152174160878
```

Рис. 3: Численное решение динамической системой Лоренца



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

[9]:

Аттрактор Лоренца

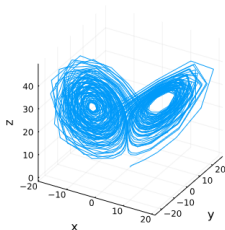


Рис. 4: Аттрактор Лоренца (интерполяция отключена)

# Модель Лотки–Вольтерры

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

Resolving package versions...
No Changes to 'C:\Users\Reachna\.julia\environments\v1.9\Project.toml'
No Changes to 'C:\Users\Reachna\.julia\environments\v1.9\Manifest.toml'

[11]: using ParameterizedFunctions, DifferentialEquations, Plots;
# задаём описание модели:
lv1 = @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{lv1,u0,tspan,p}
sol = solve(prob)

[11]: 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
 0.670821987497083
 0.9444046158046306
 1.2674601546821105
 1.6192913303893046
 1.9869754428624007
 2.2640902393538296
 2.5125484298063063
 2.7468200290123062
 3.0380065851974147
 ⋮
 6.455762090996754
 6.780496138817711
 7.171040059920871
 7.584863345264154
 7.978868981329682
 8.48316543760351
 8.719248247740158
 8.949206788834692
 9.200185054623292
 9.438029017301554
 9.711808134779586
10.0
```

Рис. 5: Численное решение модель Лотки–Вольтерры

# Модель Лотки–Вольтерры

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

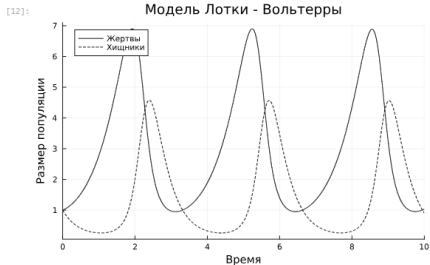


Рис. 6: Модель Лотки–Вольтерры: динамика изменения численности популяций

# Модель Лотки–Вольтерры

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

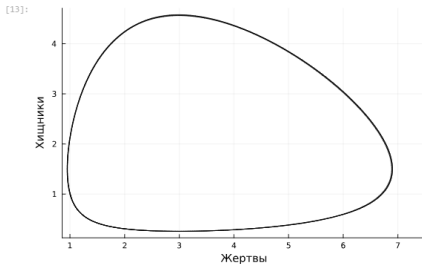


Рис. 7: Модель Лотки–Вольтерры: фазовый портрет

# Модель Мальтуса

```
[14]: # Task1
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)

[14]: retcode: Success
Interpolation: specialized 4th order "free" interpolation, specialized 2nd order "free" stiffness-aware interpolation
t: 12-element Vector{Float64}:
 0.0
 0.07579340539309044
 0.2176538131796436
 0.39326275375009306
 0.6100444793398203
 0.8636787203353302
 1.1544101119687582
 1.4789340537388638
 1.8349001265017795
 2.219134461733416
 2.628731787861167
 3.0
u: 12-element Vector{Vector{Float64}}:
 [2.0]
 [2.327358634990142]
 [3.0908767890047213]
 [4.391507855871063]
 [6.774976441549192]
 [11.251525438510586]
 [20.12503871207196]
 [30.513500897099114]
 [78.48706775956025]
 [169.25231460681178]
 [383.9709586782193]
 [806.8145670268354]
```

Рис. 8: Решение модели Мальтуса

# Модель Мальтуса

```
plot(sol, label = "Численность изолированной популяции  $x(t)$ ", color="green", ls="solid", title="Модель Мальтуса",  
      xaxis="Время", yaxis="Размер изолированной популяции")
```

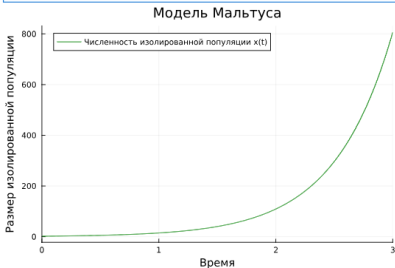


Рис. 9: График модели Мальтуса

# Логистическая модель роста популяции

```
[17]: # Task2
using ParameterizedFunctions, DifferentialEquations, Plots;

# задаем описание модели:
lv! = @ode_def Logistic_population begin
    dx = r*x*(1-x/k)
end r k
# задаем начальное условие:
u0 = [1.0]
# задаем значения параметров:
p = (0.9, 20)
# задаем интервал времени:
tspan = (0.0, 10.0)
# решение:
prob = ODEProblem(lv!, u0, tspan, p)
sol = solve(prob)

[17]: retcode: Success
Interpolation: specialized 4th order "free" interpolation, specialized 2nd order "free" stiffness-aware interpolation
t: 14-element Vector{Float64}:
 0.0
 0.10320330193850687
 0.3855506045099877
 0.780748965506008
 1.262015691559725
 1.8586159628422565
 2.574933530608521
 3.471498551774082
 4.571529609523612
 5.629314234929612
 6.930091225213617
 8.078262639019435
 9.531767314565881
10.0
u: 14-element Vector{Vector{Float64}}:
 [1.0]
 [1.092018818522065]
 [1.3860627615966585]
 [1.9212436877635002]
 [2.816808473652035]
 [4.379382291381814]
 [6.9638339217858904]
 [10.89715151483962]
 [15.262798385676023]
 [17.860400164058895]
 [19.28300459695191]
 [19.73872139608262]
 [19.928358494866384]
 [19.95293645513508]
```

Рис. 10: Решение логистическую модель роста популяции

# Логистическая модель роста популяции

```
[18]: plot(sol, label = "Численность популяции  $x(t)$ ", color="blue", ls=[:solid], title="Логистическая модель роста популяции",  
        xaxis="Время", yaxis="Размер популяции")
```

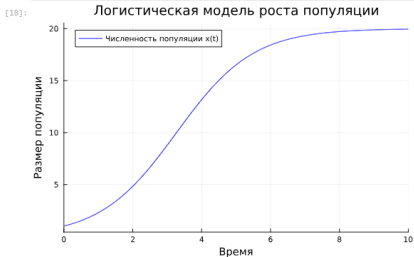


Рис. 11: График логистическую модель роста популяции



```
# задаём описание модели:
lv1 = @ode_def SIR begin
  ds = - b*i*s
  di = b*i*s - v*i
  dv = 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(lv1,u0,tspan,p)
sol = solve(prob)

retcode: Success
Interpolation: specialized 4th order "free" interpolation, specialized 2nd order "free" stiffness-aware interpolation
t: 19-element Vector{Float64}:
 0.0
 0.00888145925786733
 0.674649456183469
 1.9774587638268786
 3.928688933845557
 6.371588738903415
 9.52414865378298
 13.099293783864182
 17.0272982736833
 22.927215428937856
 27.195723313968843
 33.36658873512655
 39.87152643668008
 49.090534048044405
 57.69126316873367
 69.09753551513025
 81.37728197451536
 95.06634285664659
100.0
u: 19-element Vector{Vector{Float64}}:
 [1.0, 0.1, 0.0]
 [0.9979636107859043, 0.10162869618330198, 0.0004076931107937434]
 [0.9821139347502068, 0.11427647441254161, 0.003609590837251619]
 [0.9414409947662143, 0.1464982846000639, 0.012068720633721717]
 [0.864254086672518, 0.2065639776328575, 0.029176413679850716]
 [0.74121657128916, 0.28889894251007387, 0.0598924858006701]
 [0.5567300467580422, 0.42613510184975667, 0.11713485139220119]
 [0.360654706841008, 0.535379239828872, 0.20396602917610487]
 [0.20739657233924386, 0.5779798290842139, 0.3146235985765423]
 [0.09060823642513902, 0.5202223125052752, 0.4801694510695858]
 [0.05338199089370889, 0.4606394458344585, 0.585978564322846]
 [0.028391022566481953, 0.35937261970727175, 0.71223635772624631]
```

Рис. 12: Решение SIR модель

```
plot(sol, label = ["Восприимчивые" "Инфицированные" "Переболевшие"], color=["blue" "green" "red"], ls=[:solid :dash :dot],  
title="Модель эпидемии Кермака-Маккендрика SIR", xaxis="Время", yaxis="Размер популяции")
```

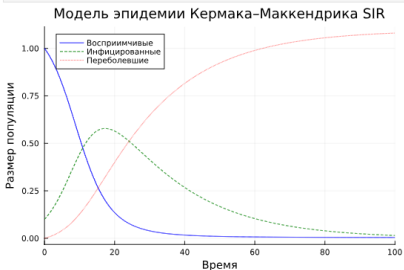


Рис. 13: График SIR модель

```
[23]: # Task4
using ParameterizedFunctions, DifferentialEquations, Plots;
N = 1.0
# задаём описание модели:
lv1 = @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 β γ δ

initialInfect = 0.1
# задаём начальные условия:
u0 = [(N - 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(lv1, u0, tspan, p)
sol = solve(prob)

[23]: retcode: Success
Interpolation: specialized 4th order "free" interpolation, specialized 2nd order "free" stiffness-aware interpolation
t: 25-element Vector{Float64}:
 0.0
 0.024423707511123237
 0.21983937298994613
 0.672446110582935
 1.3433111385495904
 2.2048531822239386
 3.1191976283637796
 4.685982405908369
 6.3524541259891905
 8.357303010682124
10.779894718728759
13.70832499771059
17.247759058759296
21.418982739513048
26.11587299819979
31.347128286227476
37.434253818037966
45.57822727276479
51.92822542671895
59.56986630189384
67.80367237385267
75.3587774263594
84.11521635193901
93.88039928820192
100.0
u: 25-element Vector{Vector{Float64}}:
 1.0 0.0 0.0 0.0
 0.9755762924888767 0.002442370751123237 0.022446110582935 0.0000000000000000
 0.9501606194100538 0.004378765858791263 0.04162123418120876 0.0000000000000000
 0.924744946231231 0.006317531717562526 0.07913880181879124 0.0000000000000000
 0.8993292730524091 0.008256297576383751 0.116656912919979 0.0000000000000000
 0.8739136001735862 0.01019506343557507 0.154175024021958 0.0000000000000000
 0.8484979272947633 0.0121338292946663 0.191693136123937 0.0000000000000000
 0.8230822544159404 0.01407260515375757 0.229211248225916 0.0000000000000000
 0.7976665815371175 0.0160113810128488 0.266729360323895 0.0000000000000000
 0.7722509086582946 0.0179501568719401 0.304247472425874 0.0000000000000000
 0.7468352357794717 0.0198889327310314 0.341765584527853 0.0000000000000000
 0.7214195629006488 0.0218277085901227 0.379283696629832 0.0000000000000000
 0.6960038900218259 0.0237664844492140 0.416801808731811 0.0000000000000000
 0.670588217143003 0.0257052603083053 0.454320020833790 0.0000000000000000
 0.645172544264180 0.0276440361673966 0.491838232935769 0.0000000000000000
 0.619756871385357 0.0295828120264879 0.529356445037748 0.0000000000000000
 0.594341198506534 0.0315215878855792 0.566874657139727 0.0000000000000000
 0.568925525627711 0.0334603637446705 0.604392869241706 0.0000000000000000
 0.543509852748888 0.0353991396037618 0.641911081343685 0.0000000000000000
 0.518094179870065 0.0373379154628531 0.679429293445664 0.0000000000000000
 0.492678506991242 0.0392766913219444 0.716947505547643 0.0000000000000000
 0.467262834112419 0.0412154671810357 0.754465717649622 0.0000000000000000
 0.441847161233596 0.0431542430401270 0.791983929751601 0.0000000000000000
 0.416431488354773 0.0450930188992183 0.829502141853580 0.0000000000000000
 0.391015815475950 0.0470317947583096 0.867020353955559 0.0000000000000000
 0.365600142597127 0.0489705706174009 0.904538566057538 0.0000000000000000
 0.340184469718304 0.0509093464764922 0.942056778159517 0.0000000000000000
 0.314768796839481 0.0528481223355835 0.979574990261496 0.0000000000000000
 0.289353123960658 0.0547868981946748 1.017093202363475 0.0000000000000000
 0.263937451081835 0.0567256740537661 1.054611414465454 0.0000000000000000
 0.238521778203012 0.0586644499128574 1.092129626567433 0.0000000000000000
 0.213106105324189 0.0606032257719487 1.129647838669412 0.0000000000000000
 0.187690432445366 0.0625420016310400 1.167166050771391 0.0000000000000000
 0.162274759566543 0.0644807774901313 1.204684262873370 0.0000000000000000
 0.136859086687720 0.0664195533492226 1.242202474975349 0.0000000000000000
 0.111443413808897 0.0683583292083139 1.279720687077328 0.0000000000000000
 0.086027740930074 0.0702971050674052 1.317238899179307 0.0000000000000000
 0.060612068051251 0.0722358809264965 1.354757111281286 0.0000000000000000
 0.035196395172428 0.0741746567855878 1.392275323383265 0.0000000000000000
 0.009780722293605 0.0761134326446791 1.429793535485244 0.0000000000000000
 0.000000000000000 0.0780522085037704 1.467311747587223 0.0000000000000000
 0.000000000000000 0.0800000000000000 1.504830000000000 0.0000000000000000
```

Рис. 14: Решение SEIR-модель

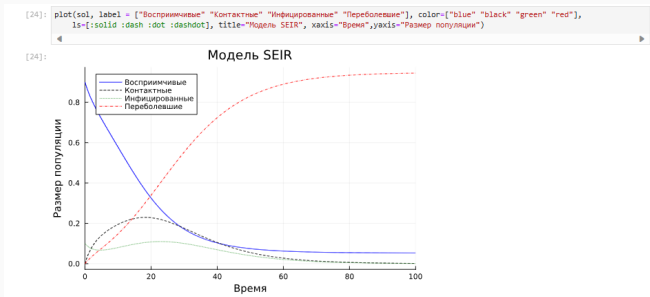


Рис. 15: График SEIR-модель

# Дискретной модели Лотки–Вольтерры

```
[26]: import Pkg
      Pkg.add("LaTeXStrings")

      Resolving package versions...
      No Changes to 'C:\Users\Reachna\.julia\environments\v1.9\Project.toml'
      No Changes to 'C:\Users\Reachna\.julia\environments\v1.9\Manifest.toml'

[27]: # Task5
      using DifferentialEquations, Plots, ParameterizedFunctions, LaTeXStrings

      # задаём значения 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. 16: Решение модели Лотки–Вольтерры

# Дискретной модели Лотки–Вольтерры

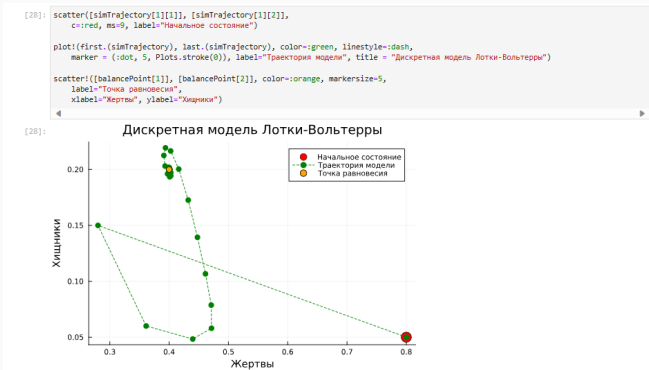


Рис. 17: График модели Лотки–Вольтерры

# Модель роста популяции в условиях коконкуренции

```
[10]: # Task6
using ParameterizedFunctions, DifferentialEquations, Plots;
# задача описана на русском
lv1 = @ode_def CompetitiveSelectionModel begin
    dx = a*x - b*x*y
    dy = a*y - b*x*y
end a b
# задаем начальные условия:
u0 = [1.0, 1.4]
# задаем значения параметров:
p = (0.5, 0.2)
# задаем интервал времени:
tspan = (0.0, 10.0)
# решаем:
prob = ODEProblem{lv1, u0, tspan, p}
sol = solve(prob)

[10]: retcode: Success
Interpolation: specialized 4th order "free" interpolation, specialized 2nd order "free" stiffness-aware interpolation
t: 20-element Vector{Float64}:
 0.0
 0.10061515958673816
 0.6620095910816169
 1.4745515040811719
 2.384274376329455
 3.4538271544345127
 4.562673961852109
 5.67839475174588
 6.818742680948912
 7.174117888180357
 7.581652232104449
 7.967287684662207
 8.211743148258813
 8.476131118051619
 8.7219313710483542
 8.961403080998073
 9.201605421462836
 9.457562710292689
 9.735425791326456
10.0
u: 20-element Vector{Vector{Float64}}:
 [1.0, 1.4]
 [1.028424415994334, 1.4554136185342876]
 [1.1340867943700582, 1.691933380954628]
 [1.2518748556042107, 2.0090783181538007]
 [1.2907822965384151, 2.408347267277165]
 [1.1633365332504924, 3.412641939340822]
 [0.8307911711859107, 4.74668561486651]
 [0.3930825927897952, 7.33789081212312]
 [0.12651304828955845, 11.803651811087418]
 [0.83764043695860551, 14.488643868861088]
 [0.812491811797030182, 17.729609345536696]
 [0.80426266579768024, 20.85489224691554]
 [0.801245992590926548, 24.27939085309544]
 [0.80036185736489178275, 27.709728708908833]
 [0.560318083134166-5, 31.364442854341014]
 [2.244580754326122-5, 35.12009587775201]
 [4.313586083100942e-6, 39.86555971687741]
 [6.934576244296251e-7, 45.26283767286127]
 [7.440000000000000e-8, 49.99999999999999]
```

Рис. 18: Решение модели роста популяции в условиях коконкуренции

# Модель роста популяции в условиях коконкуренции

```
[31]: plot(soi, label = ["1-й биологический вид" "2-ой биологический вид"], color=["blue" "red"], ls=["solid :dash"],  
        title="Модель роста популяции в условиях конкуренции", xaxis="Время",yaxis="Размер популяции")
```

[31]: Модель роста популяции в условиях конкуренции

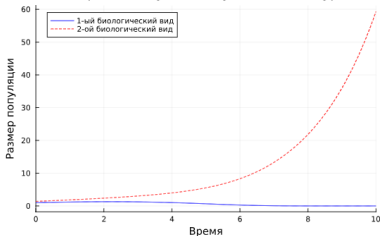


Рис. 19: График модели роста популяции в условиях коконкуренции



# Модель консервативного гармонического осциллятора

```
[34]: # задаём описание модели:
lv1 = @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(lv1,u0,tspan,p)
sol = solve(prob)

[34]: retcode: Success
Interpolation: specialized 4th order "free" interpolation, specialized 2nd order "free" stiffness-aware interpolation
t: 31-element Vector{Float64}:
 0.0
 0.07500007943195412
 0.2069885812216689
 0.35309669557584694
 0.5285194634228536
 0.7514914350697009
 1.0072081570278821
 1.2779920001493048
 1.5687719973957674
 1.8626764818913302
 2.2229737198679334
 2.5850540372165933
 2.9526997573066778
 ⋮
 5.742944245724529
 6.171924002188556
 6.584586020436235
 7.010469902680138
 7.433867666627458
 7.849769901055738
 8.282275752485367
 8.684259757356292
 9.126171387660838
 9.52416872011131
 9.970147020711996
10.0
u: 31-element Vector{Vector{Float64}}:
 [1.0, 1.0]
 [1.0640413705392677, 0.6864865930281919]
 [1.1166550813709244, 0.11102078370062098]
 [1.0853087396797187, -0.5370470078797774]
 [0.9360868170800958, -1.7401954273629171]
```

Рис. 20: Решение модель консервативного гармонического осциллятора

# График модели консервативного гармонического осциллятора

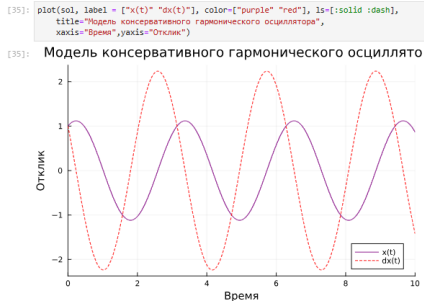


Рис. 21: График модели консервативного гармонического осциллятора

# Фазовый портрет модели консервативного гармонического осциллятора

```
[36]: # фазовый портрет:  
plot(sol, vars=(1,2), color="black", title="Фазовый портрет", xaxis="x(t)", yaxis="dx(t)", legend=false)
```

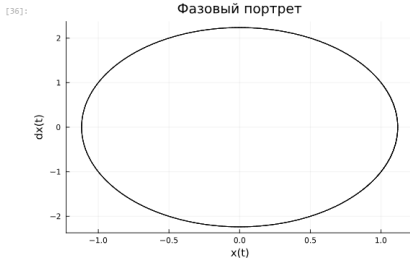


Рис. 22: Фазовый портрет

# Модель свободных колебаний гармонического осциллятора

```
[38]: # Task8
# 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)

[38]: retcode: Success
Interpolation: specialized 4th order "free" interpolation, specialized 2nd order "free" stiffness-aware interpolation
t: 31-element Vector{Float64}:
 0.0
 0.07472295275624102
 0.2111474073618621
 0.370998115535819
 0.5569932690427111
 0.7935651279047608
 1.0592494385311997
 1.3434677044587358
 1.634559088704463
 1.9695771278946115
 2.298725814870145
 2.6637691425219807
 3.0363220157596453
 ⋮
 5.737115484620817
 6.172863902331125
 6.563858473632751
 6.9841201699413915
 7.37336534430487
 7.815553300011508
 8.212080450073843
 8.639453151000238
 9.029651151704007
 9.479283944498775
 9.87771453843803
10.0
u: 31-element Vector{Vector{Float64}}:
 [0.5, 1.0]
 [0.566294838471426, 0.7739308462880993]
```

Рис. 23: Решение модели свободных колебаний гармонического осциллятора

# График модели свободных колебаний гармонического осциллятора

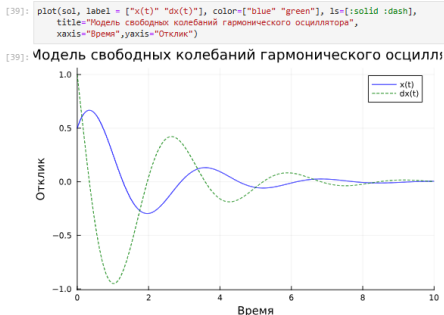


Рис. 24: График модели свободных колебаний гармонического осциллятора

# Фазовый портрет модели свободных колебаний гармонического осциллятора

```
[40]: # фазовый портрет:  
plot(sol, vars=(1,2), color="black", title="Фазовый портрет", xaxis="x(t)", yaxis="dx(t)", legend=false)
```

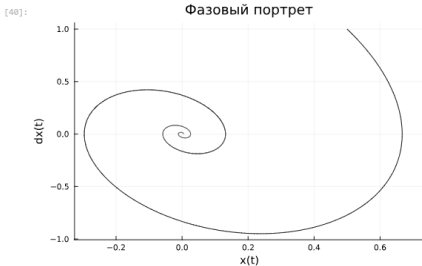


Рис. 25: Фазовый портрет

## Выводы по проделанной работе

---

Освоила специализированных пакетов для решения задач в непрерывном и дискретном времени.