РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

дисциплина: Компьютерный практикум

по статистическому данных анализ

Студент: Доре Стевенсон Эдгар

Группа: НКН-бд-01-19

МОСКВА

2022 г.

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

Цель работы:

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

Ход работы:

Повторила примеры из раздела 6.2

6.2.1. Решение обыкновенных дифференциальных уравнений

6.2.1.1. Модель экспоненциального роста

6.2.1. Решение обыкновенных дифференциальных уравнений

6.2.1.1. Модель экспоненциального роста

In [1]:

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

```
  Updating registry at `C:\Users\Admin\.julia\registries\General`
 Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
```

In [2]:

```
# 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)
```

Out[2]:

```
retcode: Success
Interpolation: automatic order switching interpolation
t: 5-element Array{Float64,1}:
 0.0
 0.10042494449239292
 0.35218603951893646
 0.6934436028208104
 1.0
u: 5-element Array{Float64,1}:
 1.0
```

In [3]:

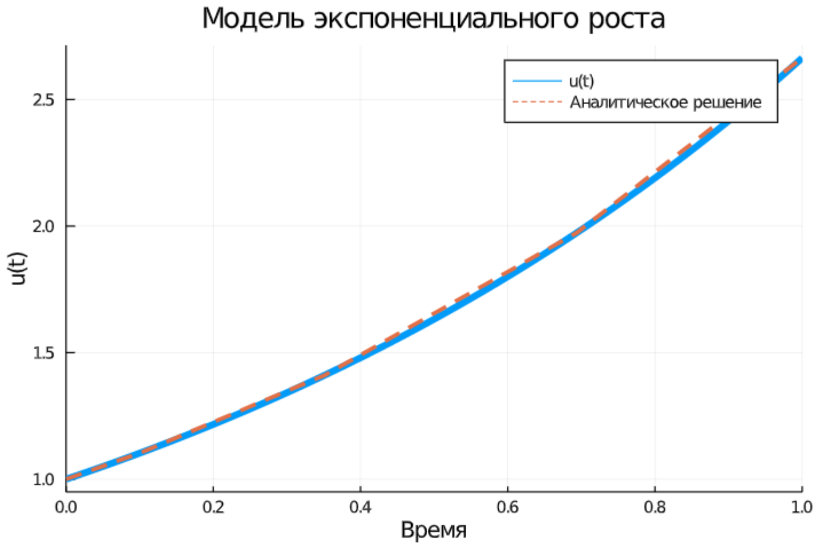
```
# подключаем необходимые пакеты:
Pkg.add("Plots")
using Plots
```

```
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
```

In [4]:

```
# 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")
```

Out[4]:

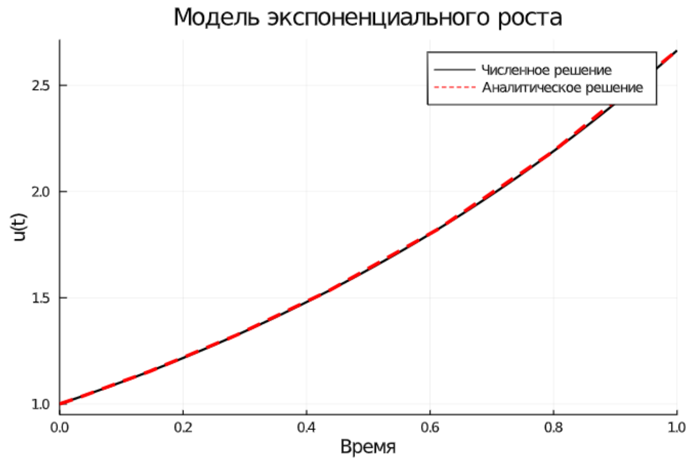


In [5]:

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

```
retcode: Success
Interpolation: automatic order switching interpolation
t: [0.0, 0.04127492324135852, 0.14679917846877366, 0.28631546412766684, 0.4381941361169628, 0.6118924302028597, 0.7985659100883
337, 0.9993516479536952, 1.0]
u: [1.0, 1.0412786454705882, 1.1547261252949712, 1.3239095703537043, 1.5363819257509728, 1.8214895157178692, 2.187139644829622
3, 2.662763824115295, 2.664456241933517]
```

Out[5]:



6.2.1.2. Система Лоренца

6.2.1.2. Система Лоренца

In [6]:
```
# подключаем необходимые пакеты:
import Pkg
Pkg.add("DifferentialEquations")
using DifferentialEquations, Plots;
```

```
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
```

In [7]:
```
# 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
```

Out[7]:
```
lorenz! (generic function with 1 method)
```

In [8]:
```
# задаём начальное условие:
u0 = [1.0,0.0,0.0]
# задаём знанчения параметров:
p = (10,28,8/3)
# задаём интервал времени:
tspan = (0.0,100.0)
# решение:
prob = ODEProblem(lorenz!,u0,tspan,p)
sol = solve(prob)
```

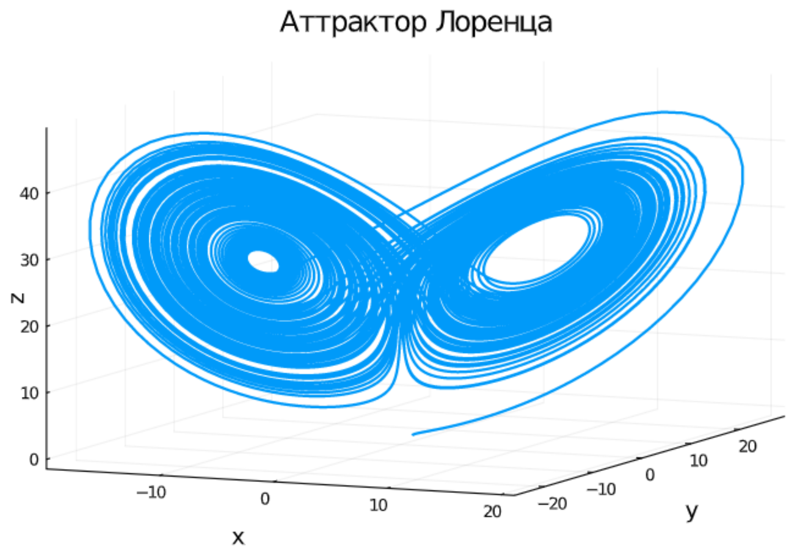
Out[8]:
```
retcode: Success
Interpolation: automatic order switching interpolation
t: 1294-element Array{Float64,1}:
   0.0
   3.5678604836301404e-5
   0.0003924646531993154
   0.003262407754410573
   0.009058075635317072
   0.01695646895607931
```

In [9]:
```
# подключаем необходимые пакеты:
Pkg.add("Plots")
using Plots
```

```
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
```

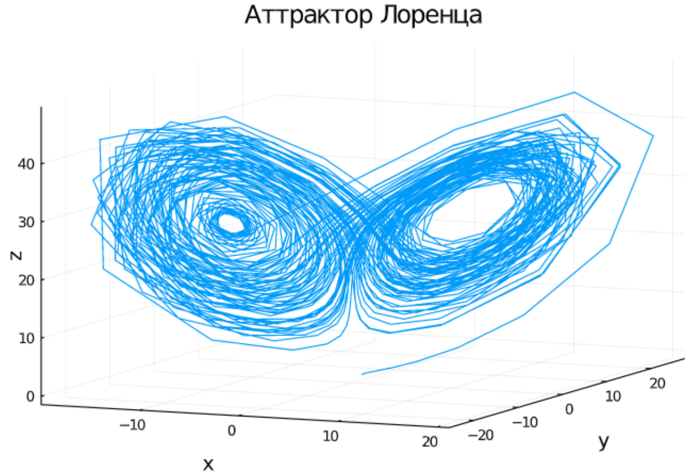
In [10]:
```
# строим график:
plot(sol, vars=(1,2,3), lw=2, title="Аттрактор Лоренца", xaxis="x",yaxis="y", zaxis="z",legend=false)
```

Out[10]:



In [11]:
```
# отключаем интерполяцию:
plot(sol,vars=(1,2,3),denseplot=false, lw=1, title="Аттрактор Лоренца", xaxis="x",yaxis="y", zaxis="z",legend=false)
```

Out[11]:



6.2.2. Модель Лотки–Вольтерры

6.2.2. Модель Лотки–Вольтерры

In [12]:
```
# подключаем необходимые пакеты:
import Pkg
Pkg.add("ParameterizedFunctions")
using ParameterizedFunctions, DifferentialEquations, Plots;
```

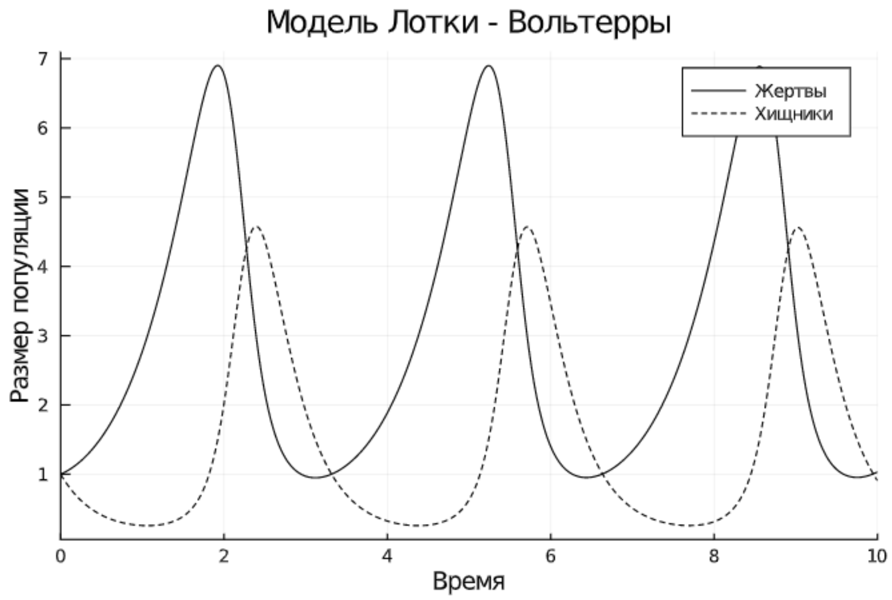
```
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
```

In [13]:
```
# 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
```

Out[13]:
```
(::LotkaVolterra{var"###ParameterizedDiffEqFunction#333",var"###ParameterizedTGradFunction#334",var"###ParameterizedJacobianFunction#335",Nothing,Nothing,ODESystem}) (generic function with 1 method)
```

In [14]:
```
# задаём начальное условие:
u0 = [1.0,1.0]
# задаём значения параметров:
p = (1.5,1.0,3.0,1.0)
# задаём интервал времени:
tspan = (0.0,10.0)
# решение:
prob = ODEProblem(lv!,u0,tspan,p)
sol = solve(prob)
plot(sol, label = ["Жертвы" "Хищники"], color="black", ls=[:solid :dash], title="Модель Лотки - Вольтерры",
xaxis="Время",yaxis="Размер популяции")
```

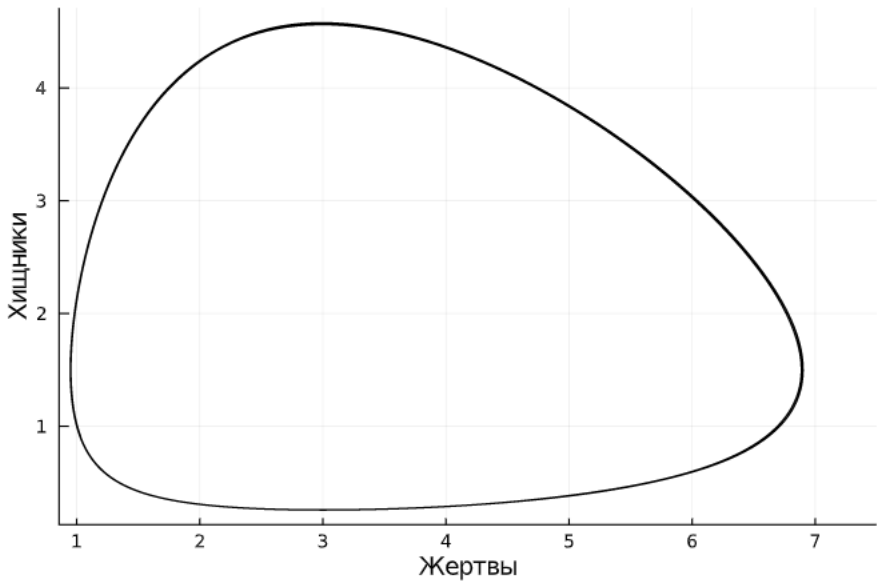
Out[14]:



In [15]:

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

Out[15]:



Задания для самостоятельного выполнения

1. Реализовать и проанализировать модель роста численности изолированной популяции (модель Мальтуса): $\dot{x} = ax$, $a = b - c$. 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).

Задание 1

In [47]:

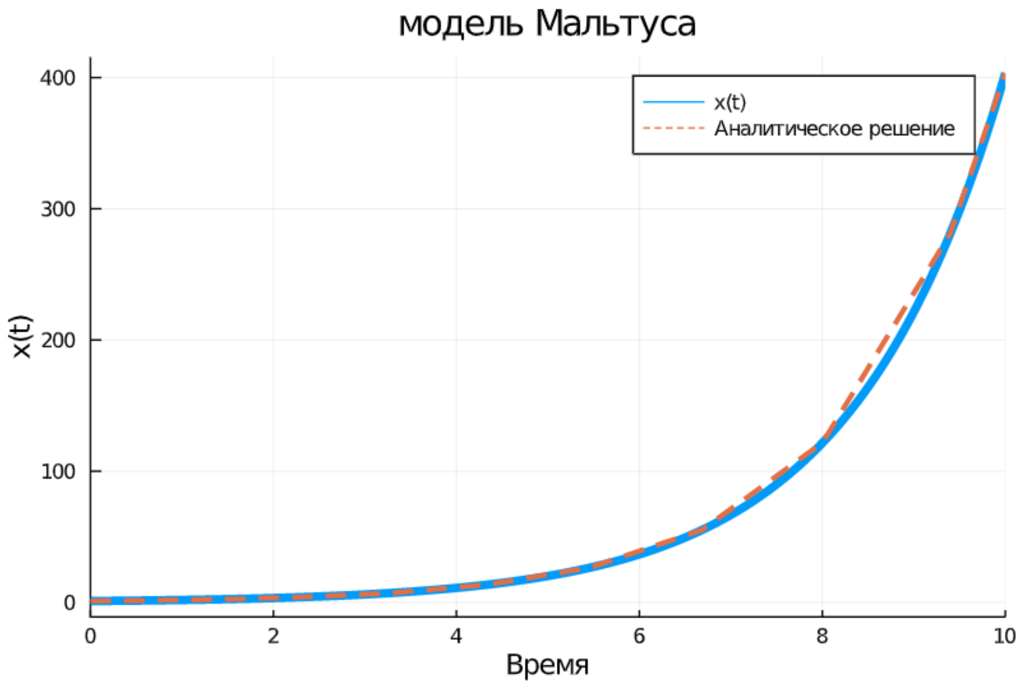
```
using DifferentialEquations
# задаём описание модели с начальными условиями:
b = 0.7 #коэффициент рождаемости
c = 0.1 #коэффициент смертности
a = b-c # коэффициент роста популяции
f(x,p,t) = a*x
x0 = 1.0
# задаём интервал времени:
tspan = (0.0,10.0)
# решение:
prob = ODEProblem(f,x0,tspan)
sol = solve(prob)
```

Out[47]:

```
retcode: Success
Interpolation: automatic order switching interpolation
t: 13-element Array{Float64,1}:
  0.0
  0.11077877679647585
  0.4740804384690646
  1.0081836371595618
  1.6416723513672835
  2.4197287550956883
  3.314796369840269
  4.332328815819821
  5.457963155008864
  6.684698795774086
  8.001406462043974
  9.398672684176248
 10.0
```

график, просчитав для проверки аналитическое решение. Численность популяции действительно увеличивается.

```julia
using Plots
# строим графики:
plot(sol, linewidth=5,title="модель Мальтуса", xaxis="Время",yaxis="x(t)",label="x(t)")
plot!(sol.t, t->1.0*exp(a*t),lw=3,ls=:dash,label="Аналитическое решение")
```

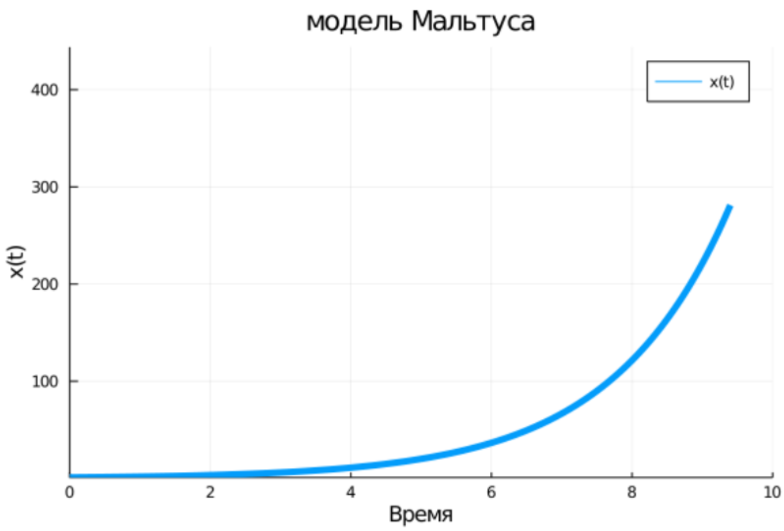


анимация данного графика.

```julia
animate(sol, fps = 7, "1.Maltus.gif",  linewidth=5,title="модель Мальтуса", xaxis="Время",yaxis="x(t)",label="x(t)")

```

```
┌ Info: Saved animation to
│   fn = C:\Users\Admin\1.Maltus.gif
└ @ Plots C:\Users\Admin\.julia\packages\Plots\5ItHH\src\animation.jl:104
```



2. Реализовать и проанализировать логистическую модель роста популяции, заданную уравнением: $\dot{x} = rx\left(1 - \frac{x}{k}\right)$, $r > 0$, $k > 0$, 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).

В данной модели первое слагаемое дает информацию о неограниченном росте популяции. Второе — о влиянии внутривидовой конкуренции (отрицательном влиянии взаимодействия двух особей одного вида) на скорость роста популяции.

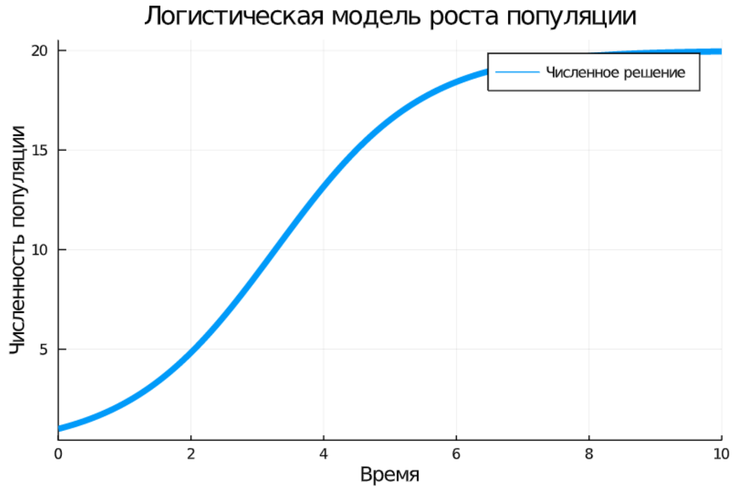
Задание 2

```
# задаём описание модели с начальными условиями:
r = 0.9 #коэффициент роста популяции
k = 20 #потенциальная ёмкость экологической системы
f(x,p,t) = r*x*(1-x/k)
x0 = 1.0
# задаём интервал времени:
tspan = (0.0,10.0)
# решение:
prob = ODEProblem(f,x0,tspan)
sol = solve(prob)
```

```
retcode: Success
Interpolation: automatic order switching interpolation
t: 14-element Array{Float64,1}:
  0.0
  0.10320330193850687
  0.3855506045099877
  0.780748965506008
  1.262015691559725
  1.8586158648823017
  2.5749333150313944
  3.4714981889836993
  4.5715292448819005
  5.629313666416045
  6.930090935678242
  8.078262058777629
  9.531766731892224
 10.0
u: 14-element Array{Float64,1}:
  1.0
  1.092018818522065
```

график данной модели.

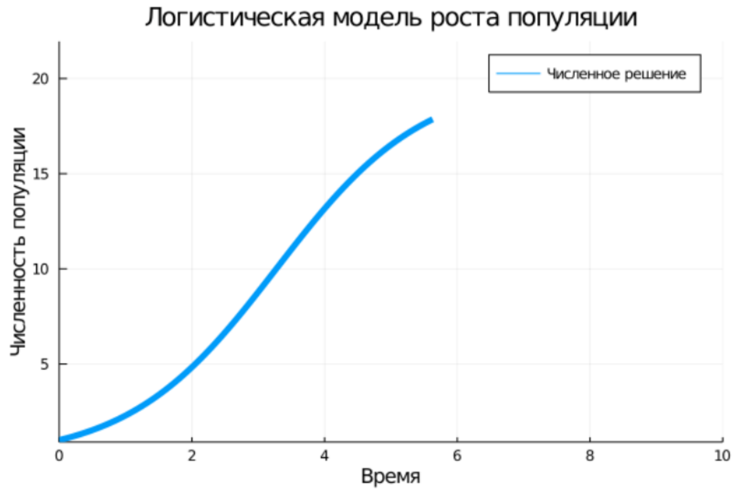
```
# строим графики:
plot(sol, linewidth=5,title="Логистическая модель роста популяции", xaxis="Время",yaxis="Численность популяции",label="Числе
#plot!(sol.t, t->k/(exp(-r*t) + 1), lw=3,ls=:dash,label="Аналитическое решение")
```



анимация

```
animate(sol, fps = 7, "2.logistModel.gif",  linewidth=5,title="Логистическая модель роста популяции",xaxis="Время",yaxis="Чи
```

```
┌ Info: Saved animation to
│   fn = C:\Users\Admin\2.logistModel.gif
└ @ Plots C:\Users\Admin\.julia\packages\Plots\5ItHH\src\animation.jl:104
```



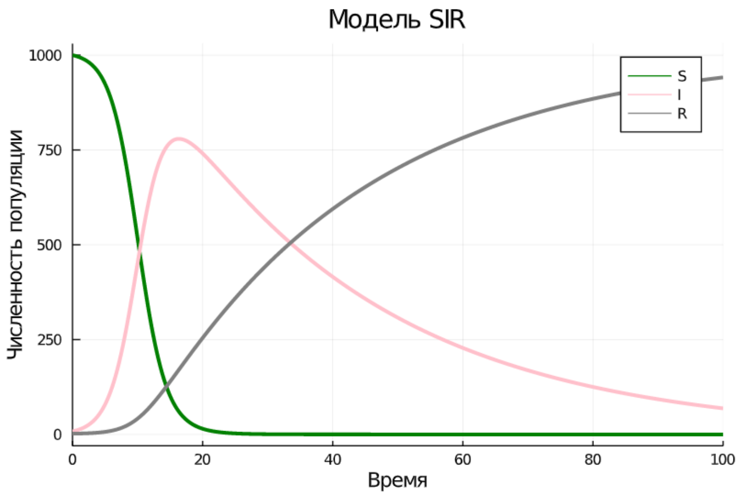
3. Реализовать и проанализировать модель эпидемии Кермака–Маккендрика (SIRмодель): $\begin{cases} \dot{s} = -\beta is, \\ \dot{i} = \beta is - \nu i, \\ \dot{r} = \nu i, \end{cases}$ где $s(t)$ — численность восприимчивых к болезни индивидов в момент времени t, $i(t)$ — численность инфицированных индивидов в момент времени t, $r(t)$ — численность переболевших индивидов в момент времени t, β — коэффициент интенсивности контактов индивидов с последующим инфицированием, ν — коэффициент интенсивности выздоровления инфицированных индивидов. Численность популяции считается постоянной, т.е. $\dot{s} + \dot{i} + \dot{r} = 0$. Начальные данные и параметры задать самостоятельно и пояснить их выбор. Построить соответствующие графики (в том числе с анимацией).

Задание 3

```julia
# задаём описание модели:
sir! = @ode_def SIR begin
    ds = -β*i*s
    di = β*i*s - v*i
    dr = v*i
end β v
# задаём начальное условие
u0 = [1000,8,2]
# задаём значения параметров
p = (0.0005,0.03)
# задаём интервал времени
tspan = (0.0,100.0)
# решение:
prob = ODEProblem(sir!,u0,tspan,p)
sol = solve(prob)
```

```
retcode: Success
Interpolation: automatic order switching interpolation
t: 32-element Array{Float64,1}:
   0.0
   0.12899112413728822
   0.6144106027322935
   1.3597509926247686
   2.236148846713095
   3.3275727507998507
   4.5907424955670395
   6.05411637988411
   7.731450937554388
   9.665161065499406
  12.056610671590565
  14.27325916231578
```

```julia
# строим график
plot(sol, linewidth=3,label = ["S" "I" "R"], color=[:green :pink :gray], title="Модель SIR", xaxis="Время",yaxis="Численност
```



анимация:

```
1  animate(sol, fps = 7, "3.SIR.gif",  linewidth=3,label = ["S" "I" "R"], color=[:green :pink :gray], title="Модель SIR",xaxis=
```

```
┌ Info: Saved animation to
│   fn = C:\Users\Admin\3.SIR.gif
└ @ Plots C:\Users\Admin\.julia\packages\Plots\5ItHH\src\animation.jl:104
```

4. Как расширение модели SIR (Susceptible-Infected-Removed) по результатом эпидемии испанки была предложена модель SEIR (Susceptible-Exposed-Infected-Removed): $\dot{s}(t) = -\beta N s(t)i(t), \dot{e}(t) = \beta N s(t)i(t) - \delta e(t), \dot{i}(t) = \delta e(t) - \gamma i(t), \dot{r}(t) = \gamma i(t)$. Размер популяции сохраняется: $s(t) + e(t) + i(t) + r(t) = N$. Исследуйте, сравните с SIR.

   SIR-модель предоставляет приближенную оценку динамики распространения эпидемии, но реальный процесс протекания болезни более сложен и включает две стадии и различные формы заболевания. SEIR-модель добавляет к трем состояниям SIR-модели четвертое состояние - зараженный в инкубационном периоде.

начальные параметры:

S – восприимчивые индивидуумы с 3 лет = 0.8

E – зараженные индивидуумы без симптомов = 0

I – инфицированные индивидуумы с симптомами = 0.2

R – вылеченные индивидуумы = 0

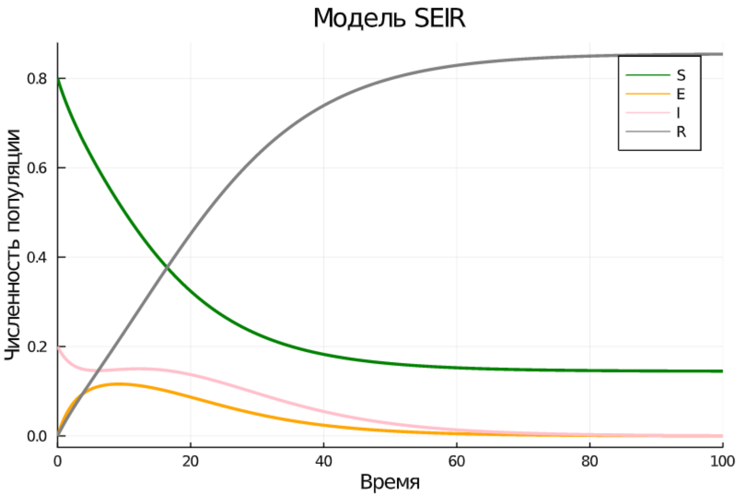
Задание 4

```
# задаём описание модели:
N = 1.0
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 β δ γ
# задаем начальные значения:
u0 = [0.8, 0.0, 0.2, 0]
# задаем значения параметров:
p = (0.3, 0.2, 0.15)
# задаём интервал времени:
tspan = (0.0,100.0)
# решение:
prob = ODEProblem(seir!,u0,tspan,p)
sol = solve(prob)
```

```
retcode: Success
Interpolation: automatic order switching interpolation
t: 26-element Array{Float64,1}:
   0.0
   0.0249065866279741
   0.21568258329525578
   0.6533402693543651
   1.3011805187737153
   2.134437064823257
   3.214278086890176
   4.543969802941188
   6.171034775597869
   8.12655008352053
```

график

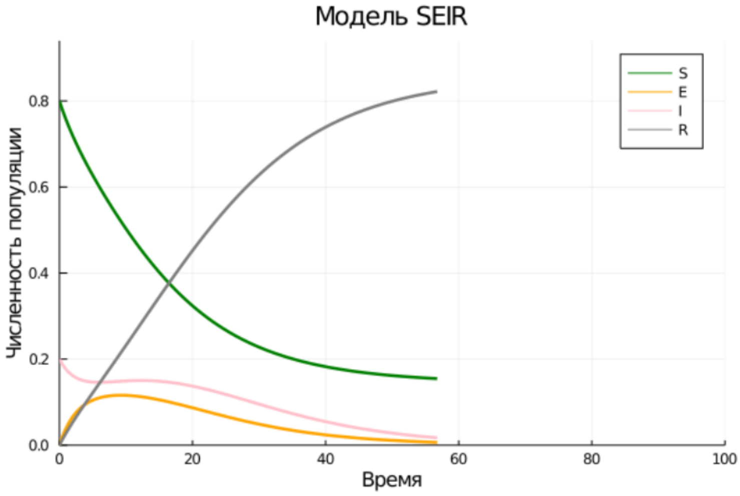
```
# строим график
plot(sol, linewidth=2.5,label = ["S" "E" "I" "R"], color=[:green :orange :pink :gray], title="Модель SEIR", xaxis="Время",ya
```



анимация:

```
animate(sol, fps = 7, "4.SEIR.gif", linewidth=2.5,label = ["S" "E" "I" "R"], color=[:green :orange :pink :gray], title="Моде
```

```
Info: Saved animation to
  fn = C:\Users\Admin\4.SEIR.gif
@ Plots C:\Users\Admin\.julia\packages\Plots\5ItHH\src\animation.jl:104
```



5. Для дискретной модели Лотки–Вольтерры: $\{ X1\,(t+1) = aX1\,(t)(1 - X1\,(t)) - X1\,(t)X2\,(t),\ X2\,(t+1) = -cX2\,(t) + dX1\,(t)X2\,(t).$ 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.

jupyter lab6 Last Checkpoint: Last Wednesday at 4:24 PM (autosaved)

File Edit View Insert Cell Kernel Widgets Help   Not Trusted   Julia 1.5.0

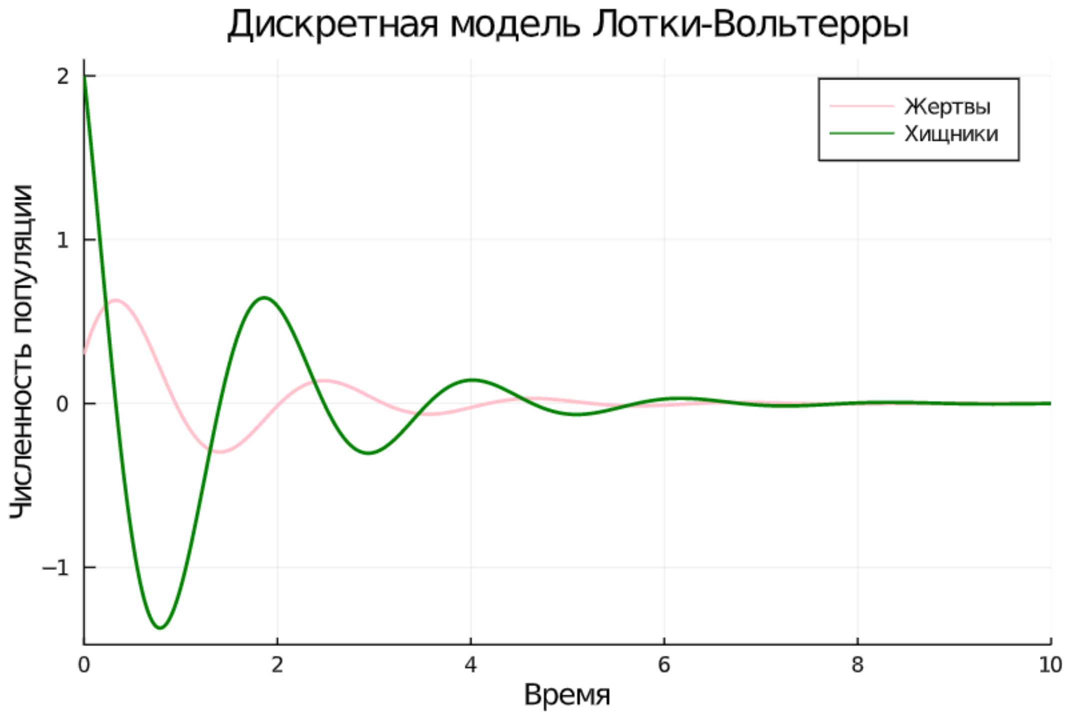
Задание 5

In [28]:
```
# задаём начальные данные
a = 2
c = 1
d = 5
# задаём начальные данные
X1 = [0.75]
X2 = [0.03]
t = 100
# задаём описание модели
dLV1(x, y) = a*x*(1-x) - x*y
dLV2(x, y) = -c*y + d*x*y
# аналитическое решение для нахождения точки равновесия
equibPoint = [(1+c)/d ,(d*(a-1)-a*(1+c))/d]
for i in 1:t
    append!(X1, dLV1(X1[i], X2[i]))
    append!(X2, dLV2(X1[i], X2[i]))
end
```

In [29]:
```
# строим график
plot(sol, linewidth=2,label = ["Жертвы" "Хищники"], color=[:pink :green],
    title="Дискретная модель Лотки-Вольтерры", xaxis="Время",yaxis="Численность популяции")
```

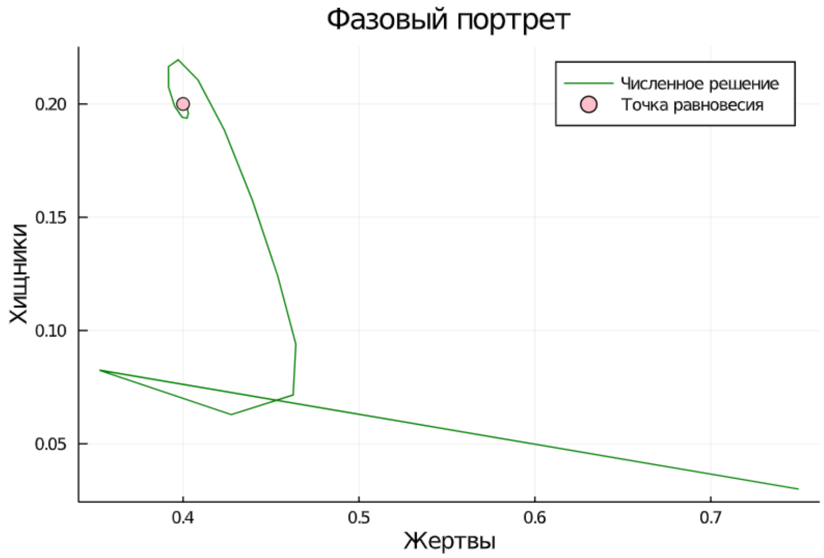
Out[29]:

Построил график:



Фазовый портрет и точка равновесия:

```
# фазовый портрет:
plot(X1, X2, title="Фазовый портрет", xlabel="Жертвы", ylabel="Хищники", label="Численное решение", c=:green)
#точка равновесия
scatter!([equibPoint[1]], [equibPoint[2]], c=:pink, shape=:circle, ms=5, label="Точка равновесия")
```



6. Реализовать на языке Julia модель отбора на основе конкурентных отношений: $\{\dot{x} = \alpha x - \beta xy, \dot{y} = \alpha y - \beta xy$. Начальные данные и параметры задать самостоятельно и пояснить их выбор. Построить соответствующие графики (в том числе с анимацией) и фазовый портрет.

Модель отбора на основе конкурентных отношений - работает при рассмотрении конкурентных взаимодействий любой природы: биохимических соединений, различного типа оптической активности, конкурирующих клеток, особей, популяций. Ее модификации применяются для описания конкуренции в экономике.

Согласно такой модели, симметричное состояние сосуществования обоих видов является неустойчивым, один из взаимодействующих видов обязательно вымрет, а другой размножится до бесконечности.

Задал начальные параметры:

a=0.1, b=0.3

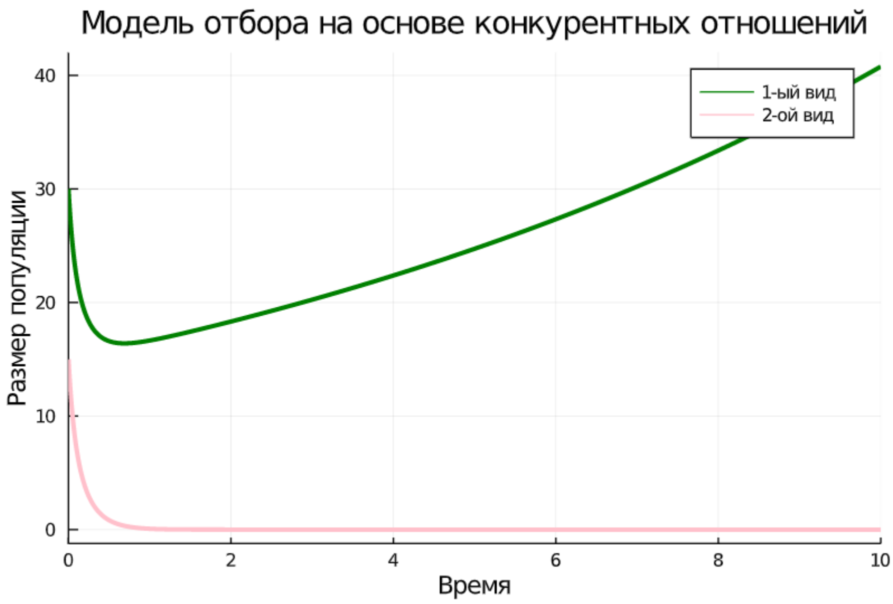
Задание 6

```
# 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 = [30, 15]
# задаём значения параметров:
p = (0.1, 0.3)
# задаём интервал времени:
tspan = (0.0, 10.0)
# решение:/
prob = ODEProblem(lv!,u0,tspan,p)
sol = solve(prob)
```

```
retcode: Success
Interpolation: automatic order switching interpolation
t: 40-element Array{Float64,1}:
  0.0
  0.04034715742273399
  0.09607885394064145
  0.2111732806369
  0.27873322877160844
  0.39571176776657246
  0.5037416779064952
  0.633159184860776
  0.7659632869461704
  0.9094880744126153
  1.057025359662743
```

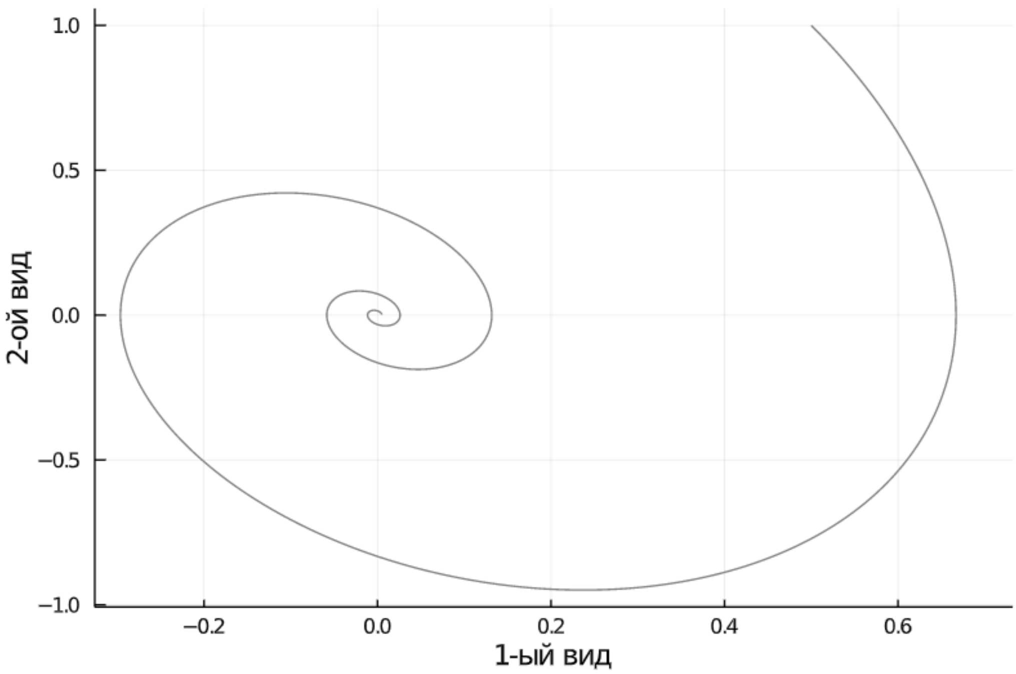
график:

```
# строим график
plot(sol, linewidth=3, label = ["1-ый вид" "2-ой вид"], color=["green" "pink"],
    title="Модель отбора на основе конкурентных отношений", xaxis="Время",yaxis="Размер популяции")
```



Построил фазовый портрет:

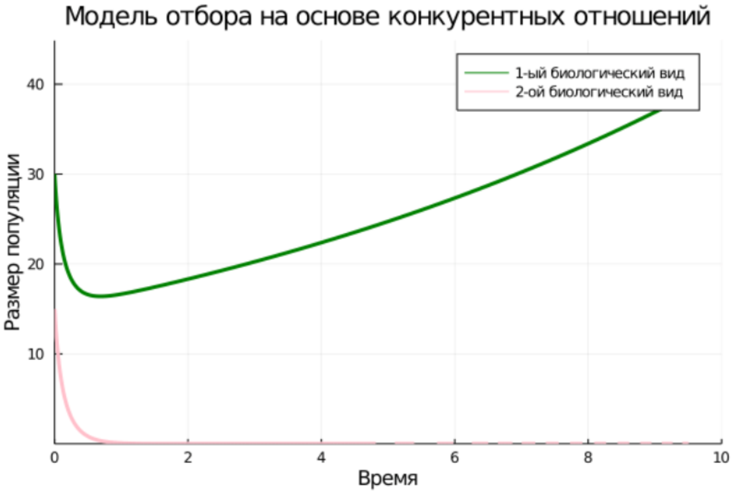
```
# фазовый портрет:
plot(sol, vars=(1,2), color="gray", xaxis="1-ый вид", yaxis="2-ой вид", legend=false)
```



Сделал анимацию:

```
animate(sol, fps=7,linewidth=3, "6.otborKonkOtnoshenii.gif", label = ["1-ый биологический вид" "2-ой биологический вид"], co
    title="Модель отбора на основе конкурентных отношений", xaxis="Время",yaxis="Размер популяции")
```

```
┌ Info: Saved animation to
│   fn = C:\Users\Admin\6.otborKonkOtnoshenii.gif
└ @ Plots C:\Users\Admin\.julia\packages\Plots\5ItHH\src\animation.jl:104
```



7. Реализовать на языке Julia модель консервативного гармонического осциллятора $\ddot{x} + \omega 2\ 0x = 0,\ x(t0\ ) = x0\ ,\ \dot{x}(t0\ ) = y0$, где $\omega 0$ — циклическая частота. Начальные параметры подобрать самостоятельно, выбор пояснить. Построить соответствующие графики (в том числе с анимацией) и фазовый портрет.

Гармонический осциллятор (в классической механике) — система, которая при выведении её из положения равновесия испытывает действие возвращающей силы, пропорциональной смещению: F=−kxF=−kx, где — постоянный коэффициент. Если — единственная сила, действующая на систему, то систему
называют простым или консервативным гармоническим осциллятором. Свободные колебания такой системы представляют собой периодическое движение около положения равновесия (гармонические колебания). Частота и амплитуда при этом постоянны, причём частота не зависит от амплитуды.

Задал начальные параметры

Циклическая частота = 3.1

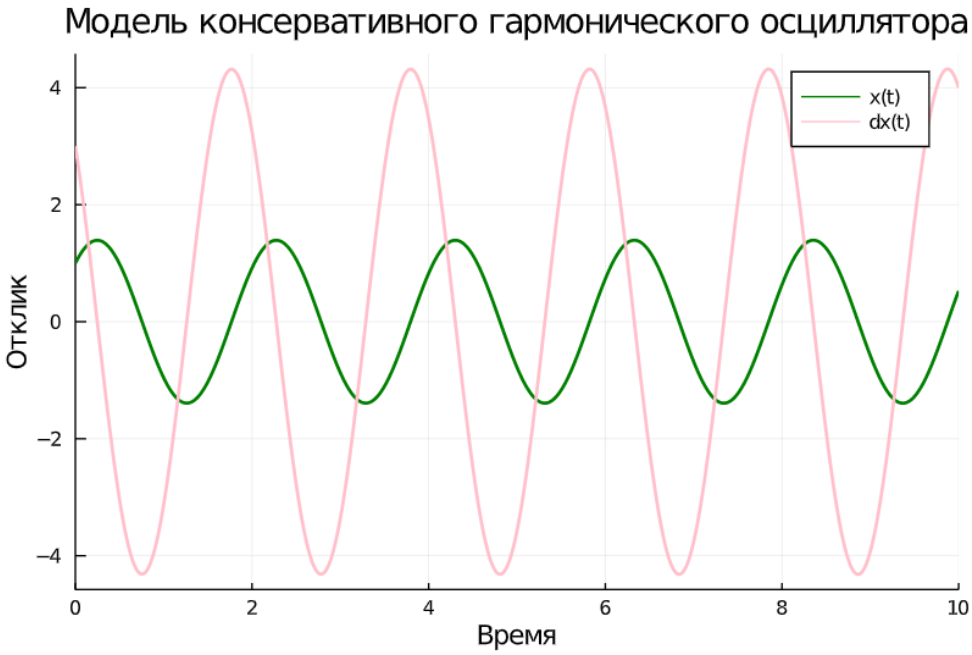
Задание 7

```
# задаём описание модели:
lv! = @ode_def Oscillator begin
dx = y
dy = -(w0^2)*x
end w0
# задаём начальное условие:
u0 = [1.0, 3.0]
# задаём значения параметров:
p = (3.1)
# задаём интервал времени:
tspan = (0.0, 10.0)
# решение:
prob = ODEProblem(lv!,u0,tspan,p)
sol = solve(prob)
```

```
retcode: Success
Interpolation: automatic order switching interpolation
t: 43-element Array{Float64,1}:
  0.0
  0.06360821298615765
  0.16088581827314855
  0.26988253193930023
  0.39135260806537686
  0.5394008803306547
  0.712376246058562
  0.8984394445965864
  1.0867554443482423
  1.3088786271222241
  1.5230676891712787
  1.7582107111566605
  2.0007197236844996
```

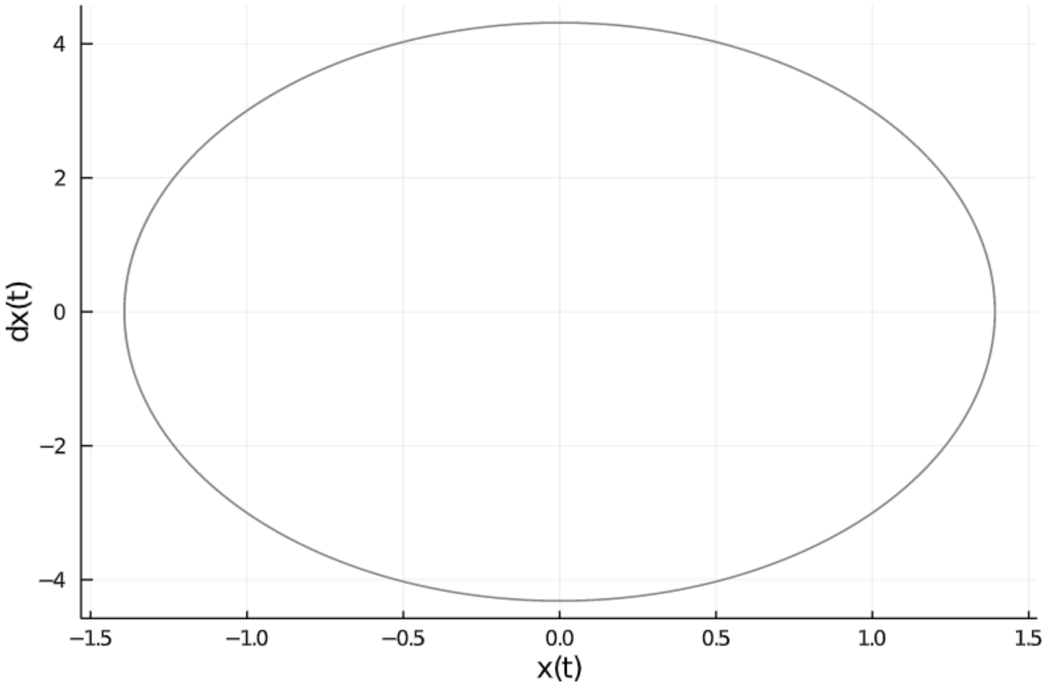
график:

```
# строим график
plot(sol, linewidth=2, label = ["x(t)" "dx(t)"], color=["green" "pink"],
    title="Модель консервативного гармонического осциллятора", xaxis="Время",yaxis="Отклик")
```



фазовый портрет:

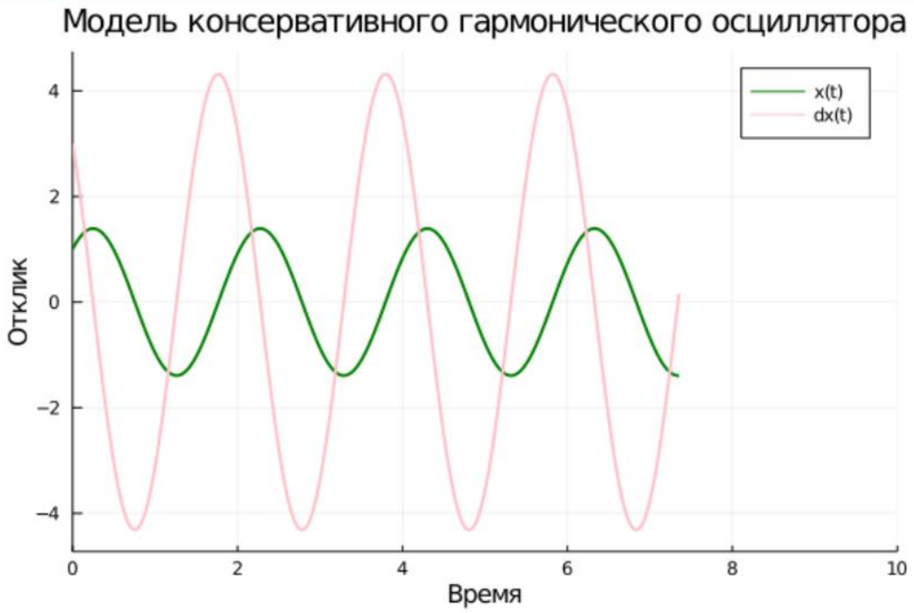
```
# фазовый портрет:
plot(sol, vars=(1,2), color="gray", xaxis="x(t)", yaxis="dx(t)", legend=false)
```



анимациz:

```
animate(sol, fps=7,linewidth=2, "7.garmonOscil.gif", label = ["x(t)" "dx(t)"], color=[:green :pink],
    title="Модель консервативного гармонического осциллятора", xaxis="Время",yaxis="Отклик")
```

```
┌ Info: Saved animation to
│   fn = C:\Users\Admin\7.garmonOscil.gif
└ @ Plots C:\Users\Admin\.julia\packages\Plots\5ItHH\src\animation.jl:104
```



8. Реализовать на языке Julia модель свободных колебаний гармонического осциллятора $\ddot{x} + 2\gamma\dot{x} + \omega_0^2 x = 0$, $x(t_0) = x_0$, $\dot{x}(t_0) = y_0$, 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.

Эта модель называется линейным гармоническим осциллятором. Уравнение свободных колебаний гармонического осциллятора имеет вид $\ddot{x} + 2\gamma\dot{x} + \omega_0^2 x = 0$.

Задал начальные параметры:

циклическая частота = 0.7

параметр, характеризующий потери энергии = 3

При таких параметрах график в скором времени должен превратиться в одну прямую линию.

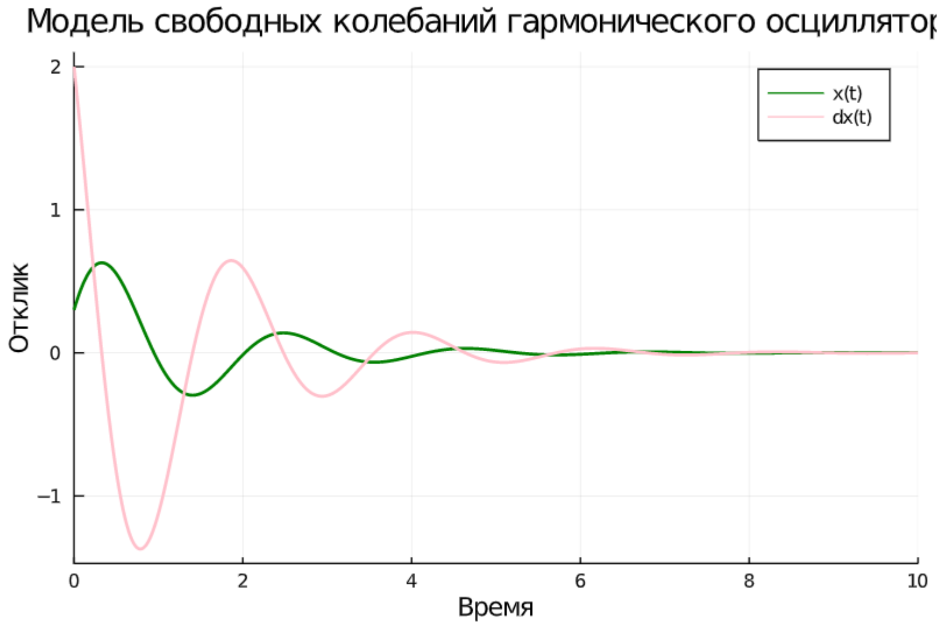
Задание 8

```
# задаём описание модели:
lv! = @ode_def Oscillator2 begin
dx = y
dy = -2*v*y - (w0^2)*x
end v w0
# задаём начальное условие:
u0 = [0.3, 2.0]
# задаём знанчения параметров:
p = (0.7, 3.0)
# задаём интервал времени:
tspan = (0.0, 10.0)
# решение:
prob = ODEProblem(lv!,u0,tspan,p)
sol = solve(prob)
```

```
retcode: Success
Interpolation: automatic order switching interpolation
t: 42-element Array{Float64,1}:
  0.0
  0.05948770714858458
  0.15779264119134842
  0.27017595788354126
  0.4045072775413231
  0.545131971590469
  0.7341354441308688
  0.9246935238717611
  1.1265150011938885
  1.3405858737189154
  1.57033033798381
```

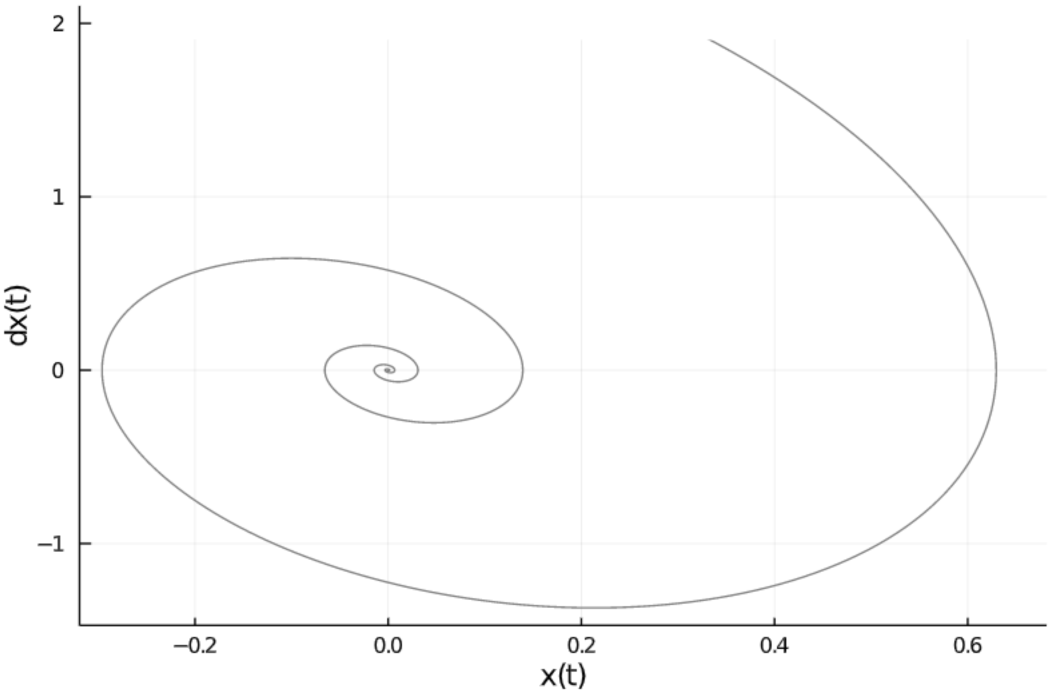
график:

```
# строим график
plot(sol, linewidth=2, label = ["x(t)" "dx(t)"], color=["green" "pink"],
    title="Модель свободных колебаний гармонического осциллятора", xaxis="Время",yaxis="Отклик")
```



Фазовый портрет:

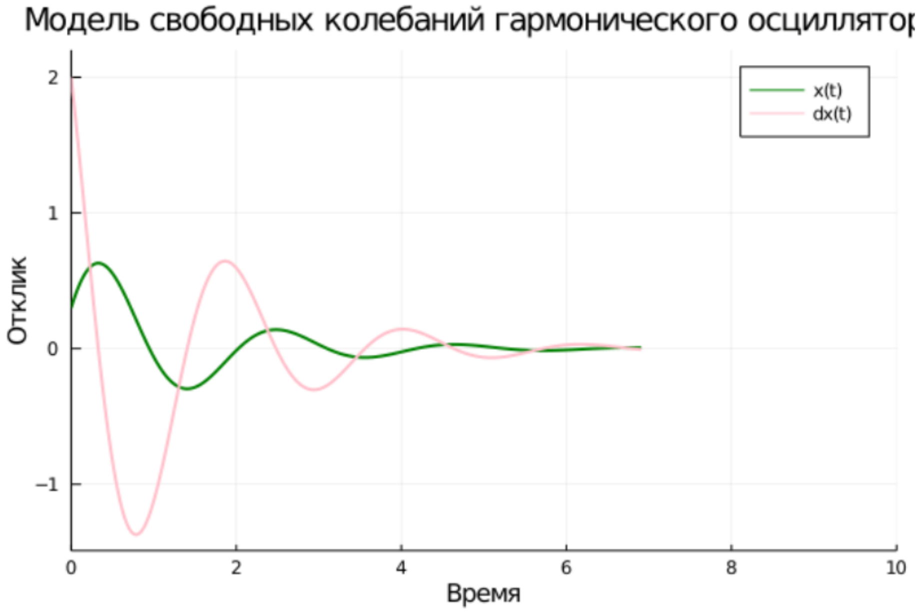
```
# фазовый портрет:
plot(sol, vars=(1,2), color="gray", xaxis="x(t)", yaxis="dx(t)", legend=false)
```



анимация

```
animate(sol, fps=7,linewidth=2, "8.kolebOscil.gif", label = ["x(t)" "dx(t)"], color=[:green :pink],
    title="Модель свободных колебаний гармонического осциллятора", xaxis="Время",yaxis="Отклик")
```

```
┌ Info: Saved animation to
│   fn = C:\Users\Admin\8.kolebOscil.gif
└ @ Plots C:\Users\Admin\.julia\packages\Plots\5ItHH\src\animation.jl:104
```



Вывод:

Получал навыки с пакетами для решения задач в непрерывном и дискретном времени.