РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра теории вероятностей и кибербезопасности

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №6

дисциплина: Компьютерный практикум по статистическому анализу данных

Студент: Быстров Глеб

Группа: НПИбд-01-20

МОСКВА

2023 г.

Цель работы

В данной лабораторной работе мне будет необходимо освоить специализированные пакеты для решения задач в непрерывном и дискретном времени.

Описание процесса выполнения работы

Модель экспоненциального роста

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

   $u'(t) = au(t), u(0) = u0.$ (6.1)

   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 (6.1) имеет вид:

   $u(t) = u0$

   $\exp(at)u(t)$.

   Численное решение в Julia будет иметь следующий вид:

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

   Построение графика (рис. 6.1), соответствующего полученному решению:

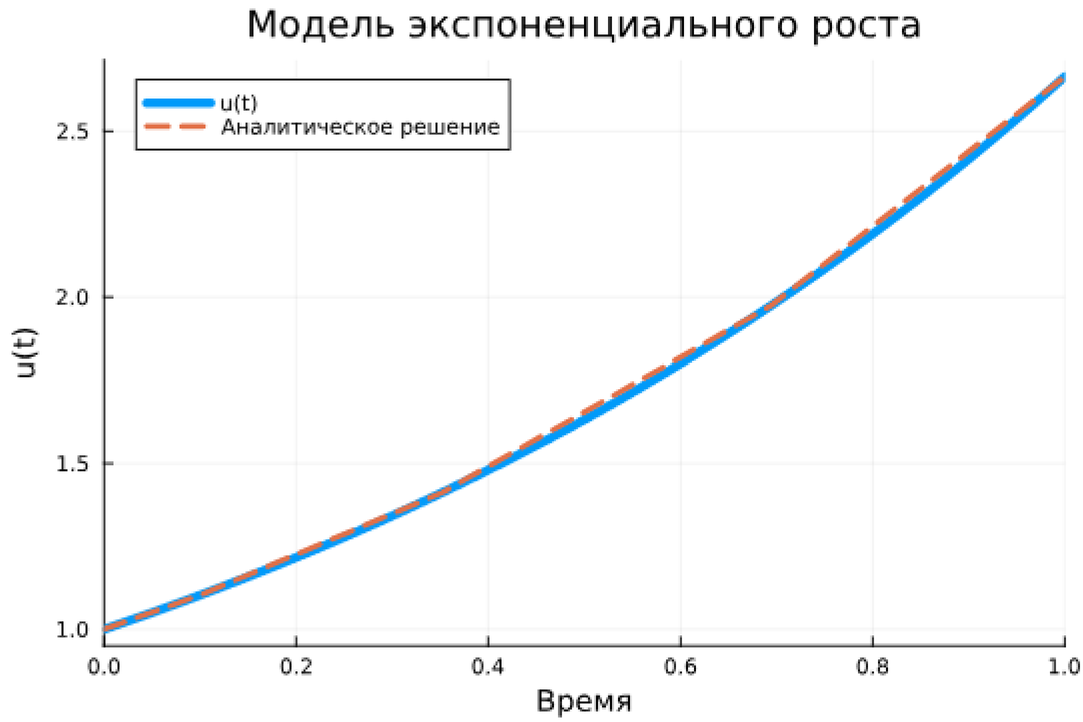
   ```
   # подключаем необходимые пакеты:
   Pkg.add("Plots")
   ```

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")



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 умолча нию эти параметры имеют значение abstol = 1e-6 и reltol = 1e-3.

Для модели экспоненциального роста (рис. 6.2):

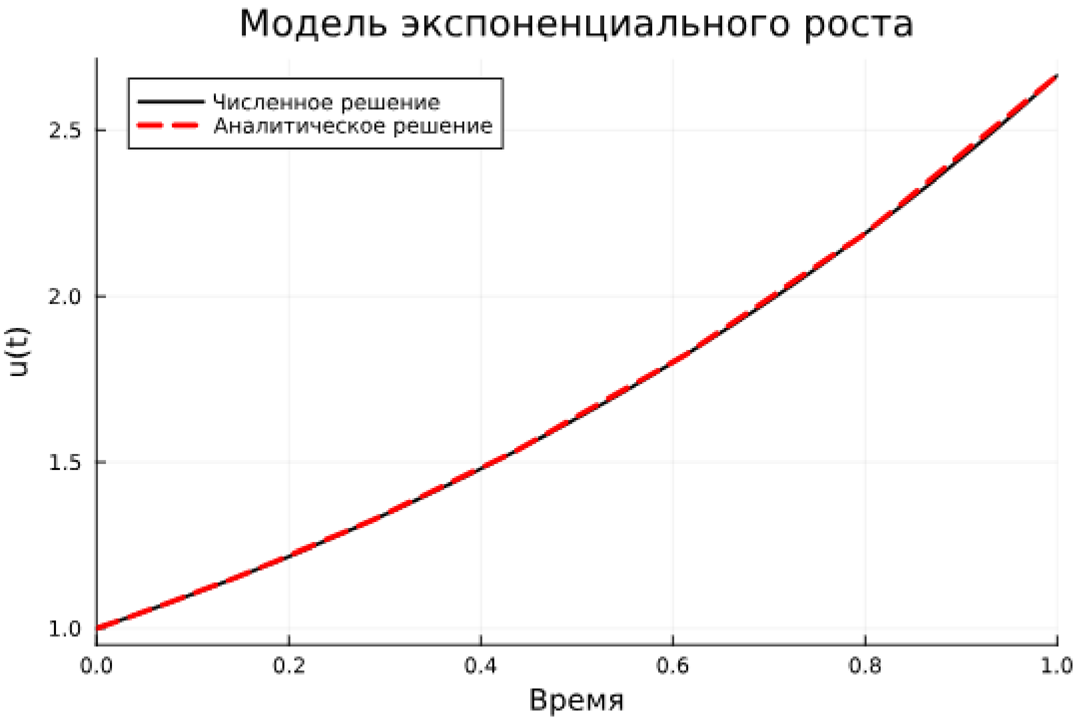
задаём точность решения:

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")
```



Система Лоренца

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

$$\begin{cases} \dot{x} = \sigma(y - x), \\ \dot{y} = \rho x - y - xz, \\ \dot{z} = xy - \beta z, \end{cases} \tag{6.2}$$

где σ, ρ и β — параметры системы (некоторые положительные числа, обычно указывают

$\sigma = 10$, $\rho = 28$ и $\beta = 8/3$).

Система (6.2) 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.

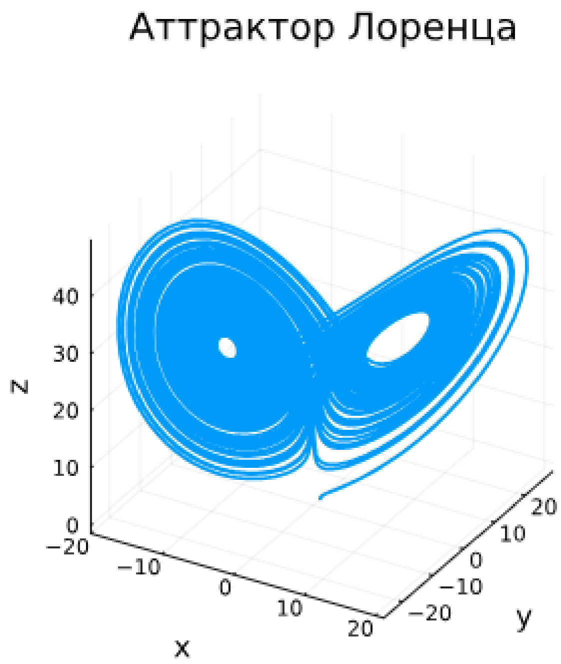
Численное решение в Julia будет иметь следующий вид:

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

Фазовый портрет (рис. 6.3):

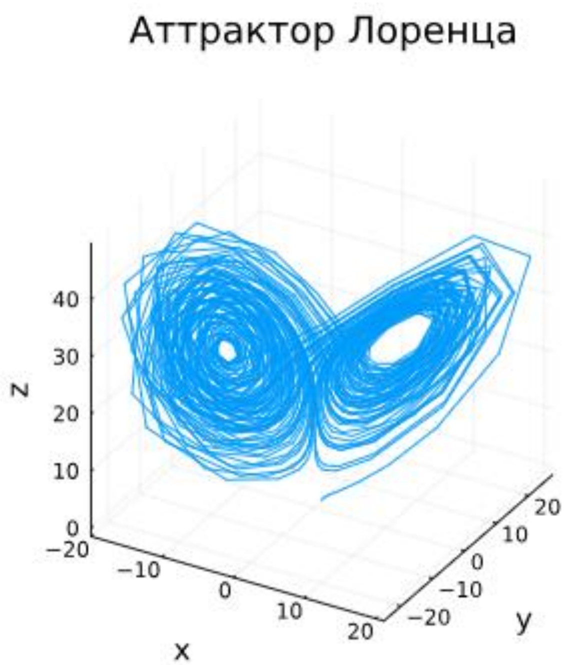
```
# подключаем необходимые пакеты:
Pkg.add("Plots")
```

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



Можно отключить интерполяцию (рис. 6.4):

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



Модель Лотки–Вольтерры

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

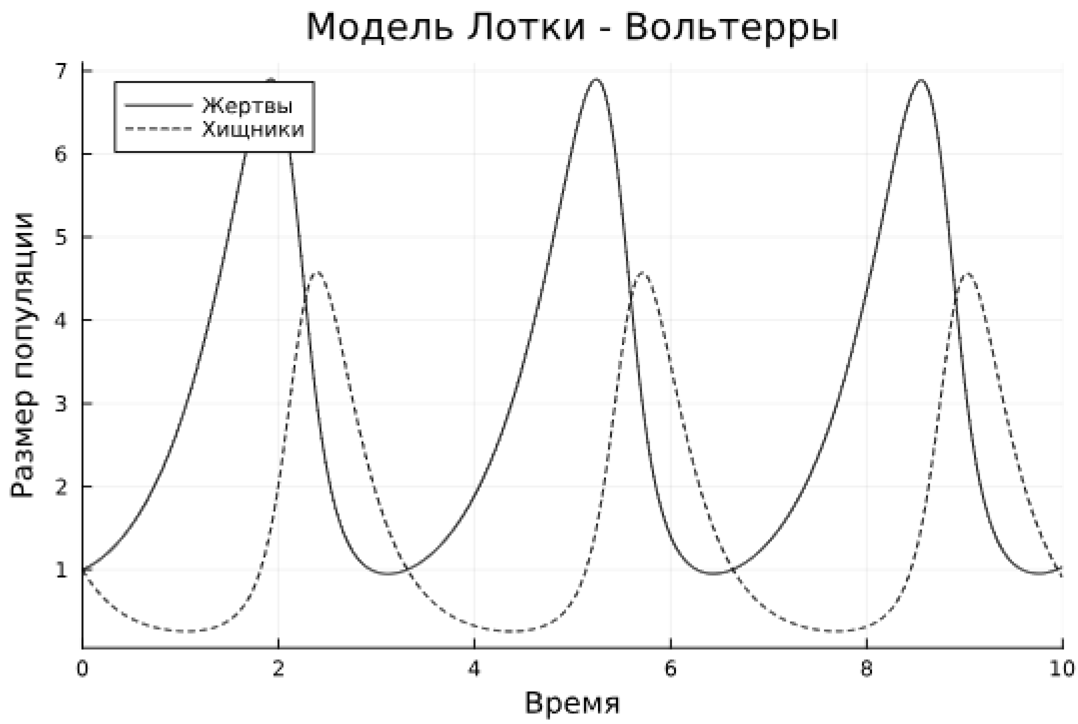
$$\begin{cases} \dot{x} = (\alpha - \beta y)x, \\ \dot{y} = (-\gamma + \delta x)y, \end{cases}$$

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 (рис. 6.5):

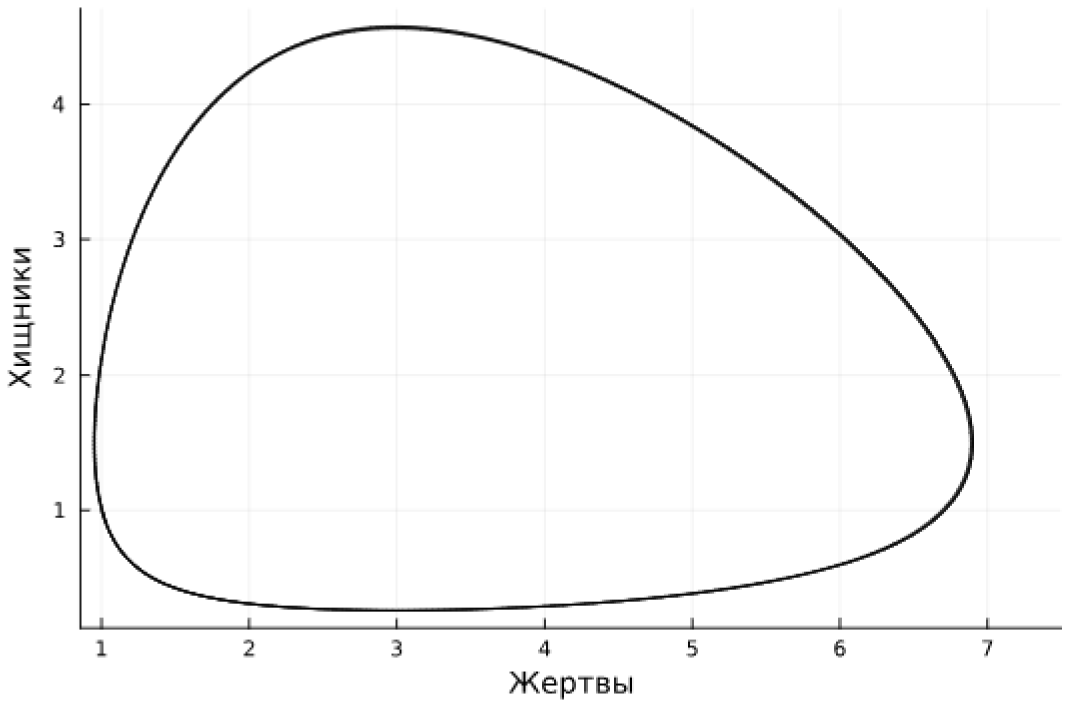
```
# подключаем необходимые пакеты:
import Pkg
Pkg.add("ParameterizedFunctions")
using ParameterizedFunctions, DifferentialEquations, Plots;
# задаём описание модели:
lv! = @ode_def LotkaVolterra begin
dx = a*x - b*x*y
dy = -c*y + d*x*y
end a b c d
# задаём начальное условие:
u0 = [1.0,1.0]
# задаём 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")
```



Фазовый портрет (рис. 6.6):

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



Задания для самостоятельного выполнения

4. 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

using ParameterizedFunctions, DifferentialEquations, Plots;

lv! = @ode_def Malthus begin
  dx = a*x
end a

u0 = [2]
b = 3.0
```

```
c = 1.0
p = (b - c)
tspan = (0.0, 3.0)

prob = ODEProblem(lv!, u0, tspan, p)
sol = solve(prob)

plot(sol, label = "Численность изолированной популяции x(t)", color="blue", ls=[:solid], title="Модель Мальтуса", xaxis="Время", yaxis="Размер изолированной популяции")

animate(sol, fps=7, "Malthus.gif", label = "Численность изолированной популяции x(t)", color="blue", ls=[:solid], title="Модель Мальтуса", xaxis="Время", yaxis="Размер изолированной популяции")
```

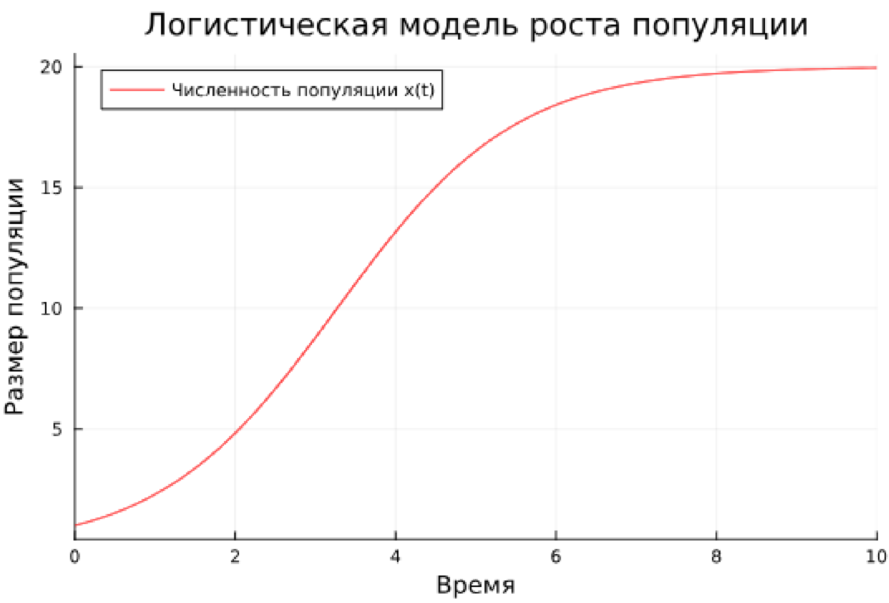
5. Реализовать и проанализировать логистическую модель роста популяции, заданную уравнением:

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

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).

#ЗАДАНИЕ №2

```
lv! = @ode_def Logistic_population begin
    dx = r*x*(1 - x/k)
    end r k

u0 = [1.0]
p = (0.9, 20)
tspan = (0.0, 10.0)

prob = ODEProblem(lv!, u0, tspan, p)
sol = solve(prob)

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

animate(sol, fps=7, "Logistic_population.gif", label = "Численность популяции x(t)", color="red", ls=[:solid], title="Логистическая модель роста популяции", xaxis="Время", yaxis="Размер популяции")
```

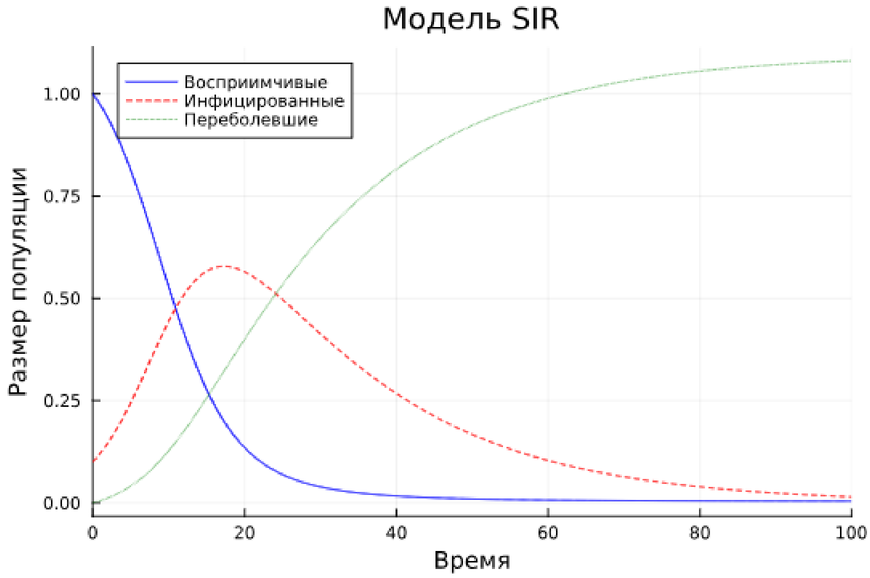


6. Реализовать и проанализировать модель эпидемии Кермака–Маккендрика (SIR-модель):

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

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

lv! = @ode_def SIR begin
ds = - b*i*s
di = b*i*s - v*i
dr = v*i
end b v

u0 = [1.0, 0.1, 0]
p = (0.25, 0.05)
tspan = (0.0, 100.0)

prob = ODEProblem(lv!,u0,tspan,p)
sol = solve(prob)

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


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")
```



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

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

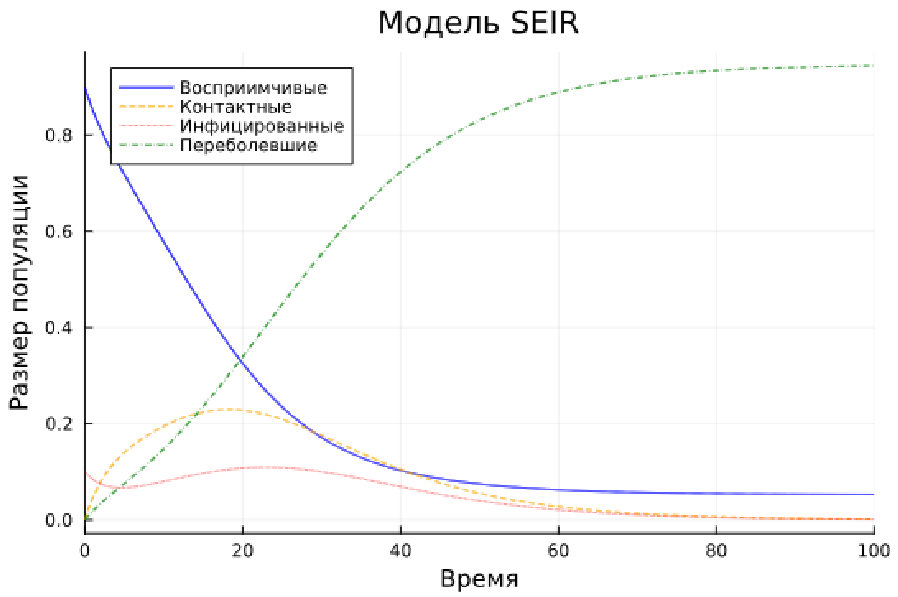
Размер популяции сохраняется:

$$s(t) + e(t) + i(t) + r(t) = N.$$

Исследуйте, сравните с SIR.

```
#ЗАДАНИЕ №4

M = 1.0

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
u0 = [(M - initialInfect), 0.0, initialInfect, 0.0]
p = (0.6, 0.2, 0.1)
tspan = (0.0, 100.0)

prob = ODEProblem(lv!,u0,tspan,p)
sol = solve(prob)

plot1 = plot(sol, label = ["Восприимчивые" "Контактные" "Инфицированные" "Переболевшие"], color=["blue" "orange" "red" "green"], ls=[:solid :dash :dot :dashdot], title="Модель SEIR", xaxis="Время",yaxis="Размер популяции")

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



8. Для дискретной модели Лотки–Вольтерры:

$$\begin{cases} X_1(t+1) = aX_1(t)(1 - X_1(t)) - X_1(t)X_2(t), \\ X_2(t+1) = -cX_2(t) + dX_1(t)X_2(t). \end{cases}$$

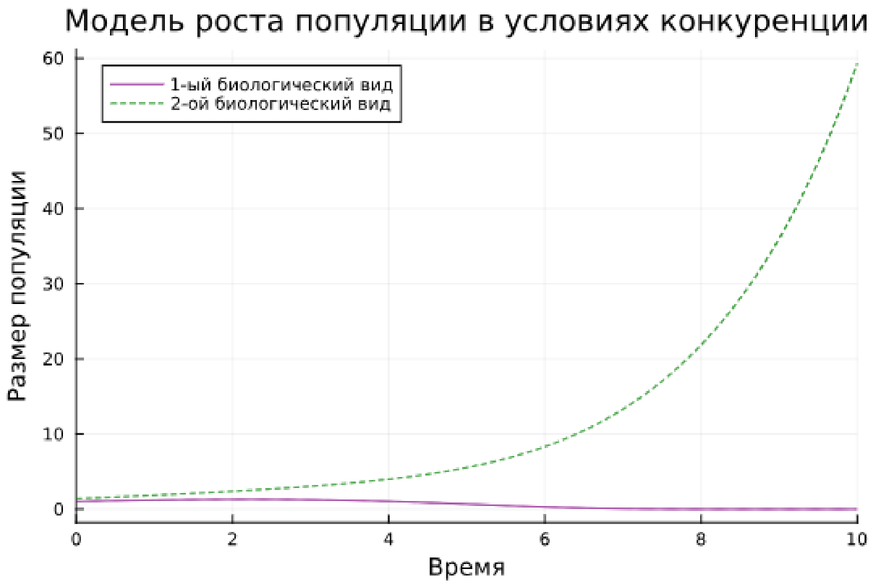
с начальными данными $a = 2$, $c = 1$, $d = 5$ найдите точку равновесия. Получите и сравните аналитическое и численное решения. Численное решение изобразите на фазовом портрете.

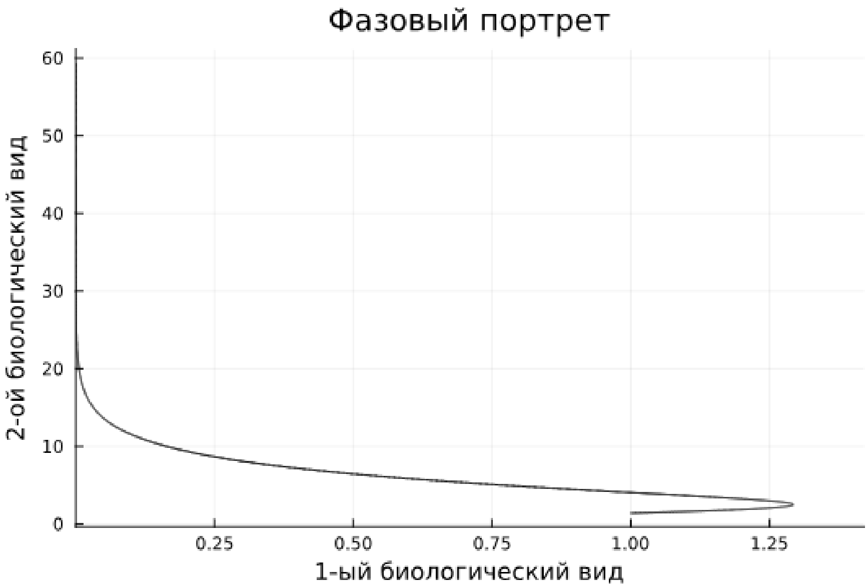
9. Реализовать на языке Julia модель отбора на основе конкурентных отношений:

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

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

```
#ЗАДАНИЕ №6

lv! = @ode_def CompetitiveSelectionModel begin
dx = a*x - b*x*y
dy = a*y - b*x*y
   end a b

u0 = [1.0, 1.4]
p = (0.5, 0.2)
tspan = (0.0, 10.0)

prob = ODEProblem(lv!,u0,tspan,p)
sol = solve(prob)


plot(sol, label = ["1-ый биологический вид" "2-ой биологический вид"], color=["purple" "green"], ls=[:solid :dash], title="Модель роста популяции в условиях конкуренции", xaxis="Время",yaxis="Размер популяции")

# фазовый портрет:
plot(sol, vars=(1,2), color="black", title="Фазовый портрет", xaxis="1-ый биологический вид", yaxis="2-ой биологический вид", legend=false)


animate(sol, fps=7, "CompetitiveSelectionModel.gif", label = ["1-ый биологический вид" "2-ой биологический вид"], color=["purple" "green"], ls=[:solid :dash],  title="Модель роста популяции в условиях конкуренции", xaxis="Время",yaxis="Размер популяции")
```


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


Фазовый портрет
2-ой биологический вид
1-ый биологический вид
0
10
20
30
40
50
60
0.25
0.50
0.75
1.00
1.25

Вывод

В данной лабораторной работе мне успешно удалось освоить специализированные пакеты для решения задач в непрерывном и дискретном времени.