

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

Отчет по лабораторной работе № 6

Дисциплина: Практикум по компьютерному моделированию

Студент: Логинов Сергей Андреевич

Группа: НФИбд-01-18

МОСКВА 2021г

Ход работы

Решение обыкновенных дифференциальных уравнений

Модель экспоненциального роста

$$u'(t) = au(t), \quad u(0) = u_0$$

Аналитическое решение данной модели имеет следующий вид:

$$u(t) = u_0 \exp(at)u(t)$$

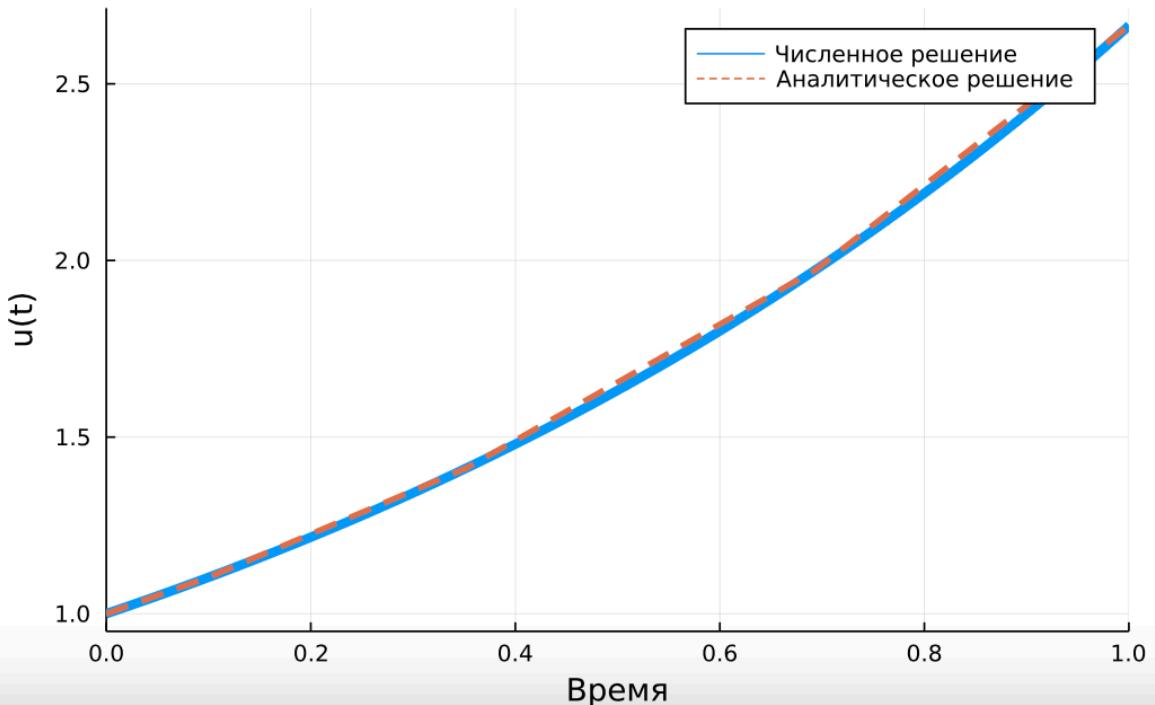
Численное решение в Julia (получаем решение и строим график):

```
In [70]: using DifferentialEquations  
a = 0.98  
f(u, p, t) = a * u  
u0 = 1.0  
  
tspan = (0.0, 1.0)  
  
prob = ODEProblem(f, u0, tspan)  
sol = solve(prob);
```

```
In [71]: using Plots  
  
plot(sol, linewidth = 5, 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,  
      label = "Аналитическое решение")
```

Out[71]:

Модель экспоненциального роста



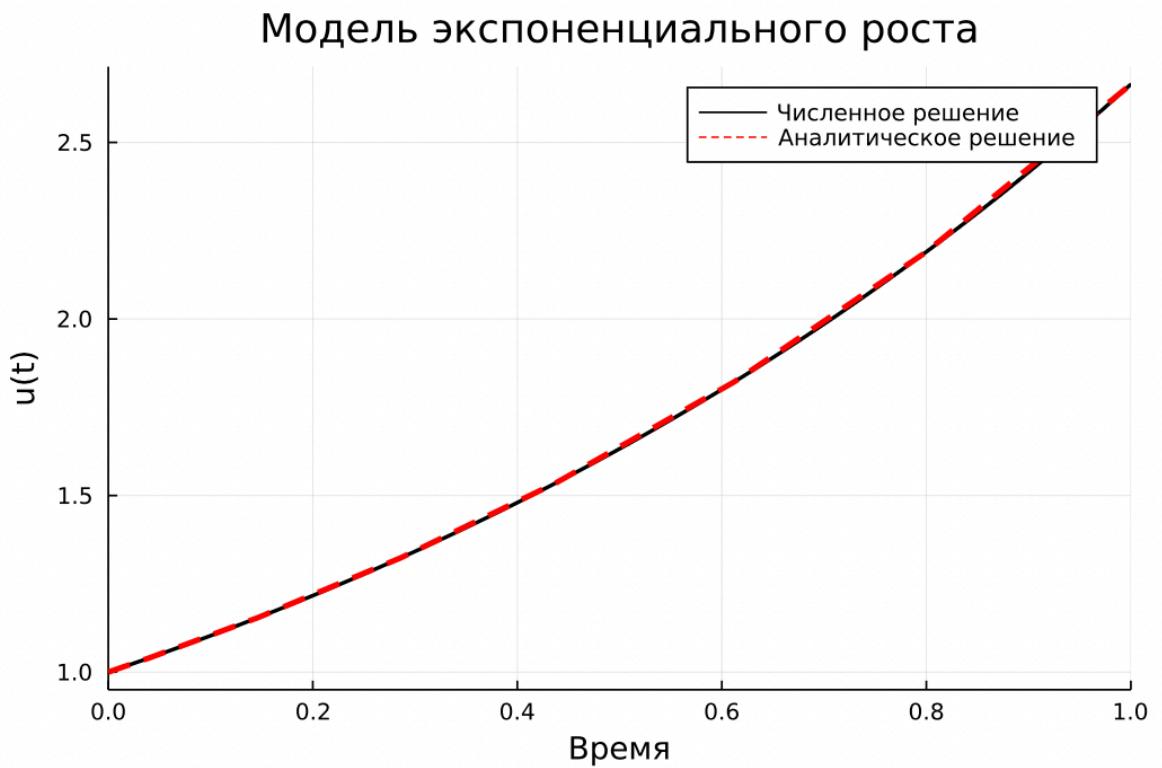
Для задания точности решения можно воспользоваться параметрами `abstol` (задает близость к нулю) и `reltol` (задает относительную точность). Повторим предыдущий шаг с использованием этих параметров:

```
In [72]: sol = solve(prob, abstol = 1e-8, reltol = 1e-8)

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")
```

Out[72]:



Система Лоренца

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

$$\begin{cases} x' = \sigma(y - x) \\ y' = \rho x - y - xz \\ z' = xy - \beta z \end{cases}$$

Решение системы неустойчиво на аттракторе, что не позволяет применять классические численные методы на больших отрезках времени, требуется использовать высокоточные вычисления.

Численное решение в Julia:

```
In [73]: 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

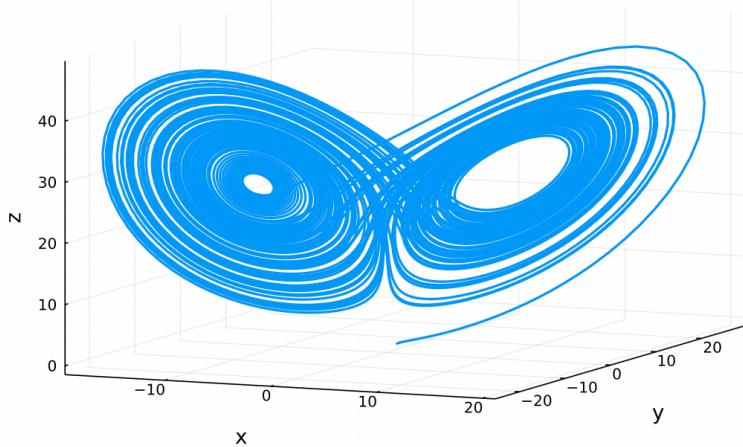
Out[73]: lorenz! (generic function with 1 method)

In [74]: u₀ = [1.0, 0.0, 0.0]
p = (10, 28, 8/3)
tspan = (0.0, 100.0)

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

In [75]: plot(sol, vars = (1, 2, 3), lw = 2, title = "Аттрактор Лоренца",
      xaxis = "x", yaxis = "y", zaxis = "z", legend = false)
```

Out[75]: Аттрактор Лоренца

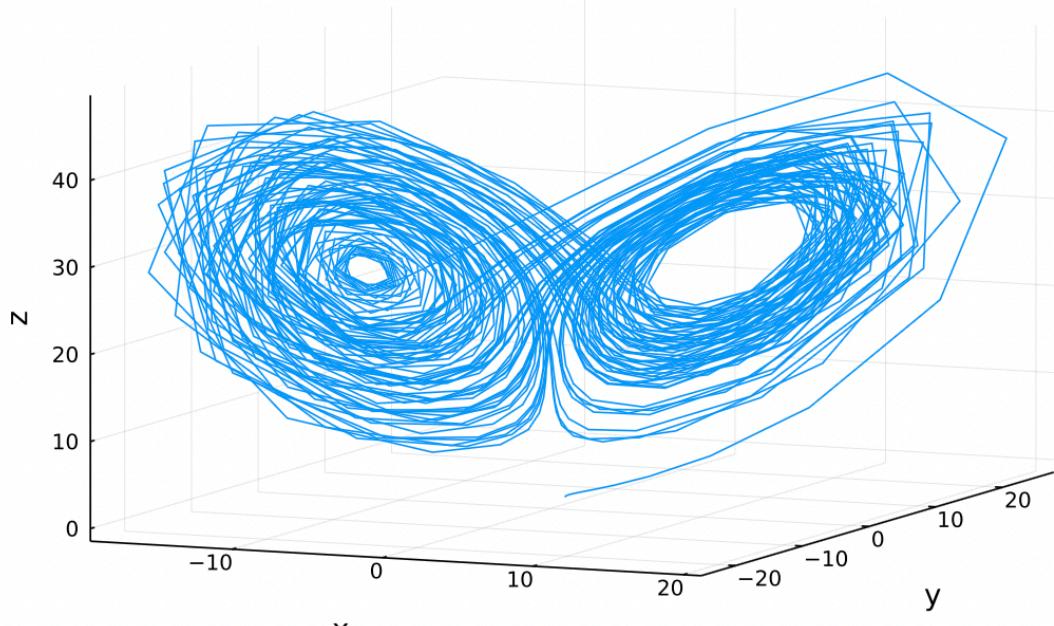


Можно отключить интерполяцию:

```
In [76]: plot(sol, vars = (1, 2, 3), denseplot=false, lw = 1,  
        title = "Аттрактор Лоренца", xaxis = "x", yaxis = "y", zaxis = "z",  
        legend = false)
```

Out[76]:

Аттрактор Лоренца



Модель Лотки-Вольтерры

Модель Лотки-Вольтерры описывает взаимодействие двух видов типа "хищник - жертва":

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

Численное решение в Julia имеет следующий вид:

In [3]:

```
using ParameterizedFunctions

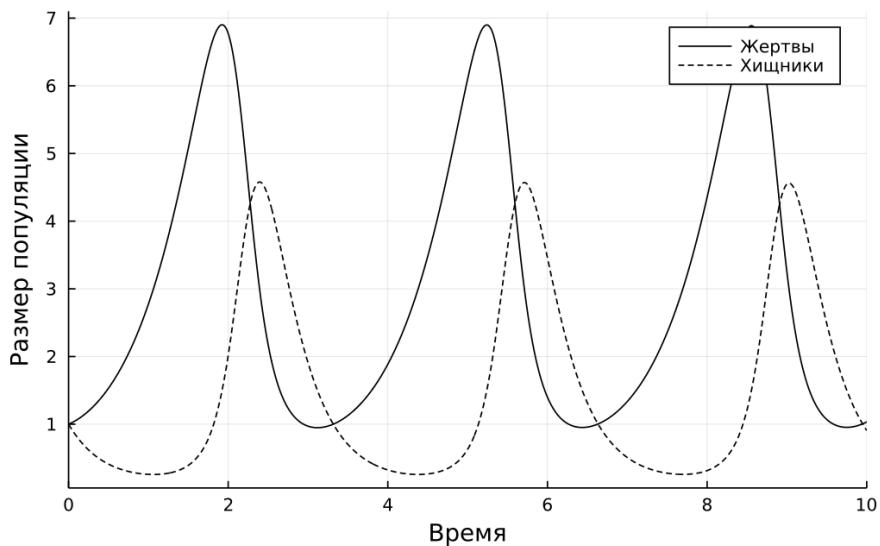
lv! = @ode_def LotkaVolterra begin
    dx = a * x - b * x * y
    dy = -c * y + d * x * y
end a b c d
|
u0 = [1.0, 1.0]
p = (1.5, 1.0, 3.0, 1.0)
tspan = (0.0, 10.0)

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 [3]:

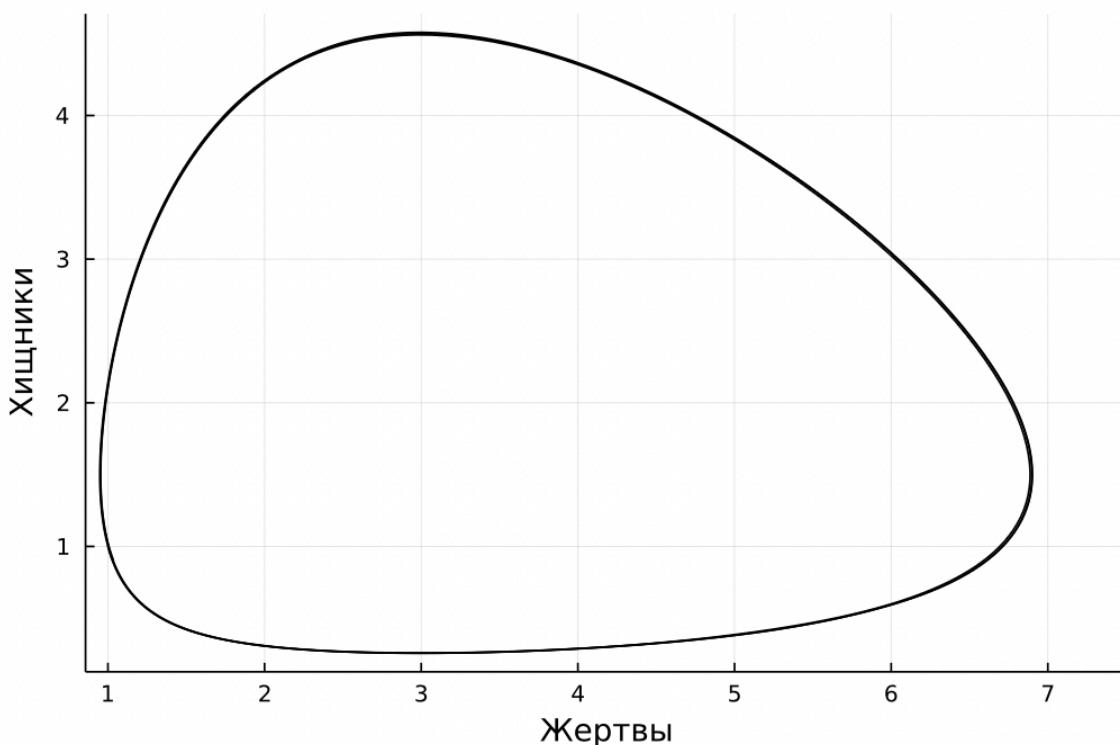
Модель Лотки – Вольтерры



Фазовый портрет системы:

```
In [78]: plot(sol, vars = (1,2), color = "black", xaxis = "Жертвы",
yaxis = "Хищники", legend = false)
```

Out[78]:



Задания для самостоятельного выполнения

1. Реализовать и проанализировать модель роста численности изолированной популяции (модель Мальтуса):

$$x' = ax, \quad a = b - c$$

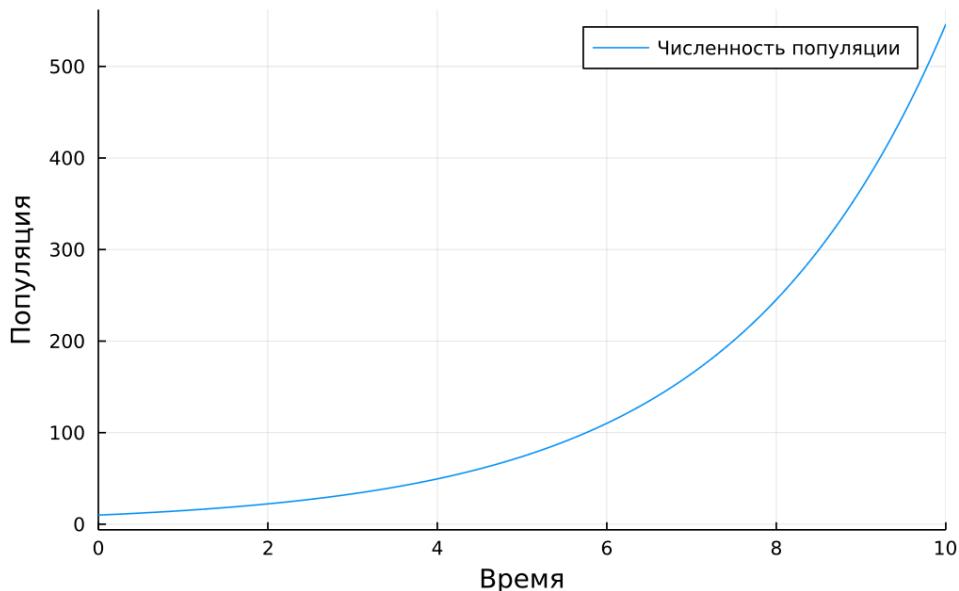
Реализация модели в Julia:

```
In [79]: f(u, a, t) = a * u
b = 1.7
c = 1.3
a = b - c
u0 = 10
tspan = (0.0, 10.0)

prob = ODEProblem(f, u0, tspan, a)
sol = solve(prob)
plot(sol, label = "Численность популяции", title = "Модель Мальтуса",
      xaxis = "Время", yaxis = "Популяция")
```

Out [79]:

Модель Мальтуса



```
In [15]: @gif for i in range(0.0, stop = 10, length = 200)
    prob = ODEProblem(f, u0, i, a)
    sol = solve(prob)
    plot(sol, label = "Численность популяции", title = "Модель Мальтуса",
          xaxis = "Время", yaxis = "Популяция")
end

Info: Saved animation to
  fn = /Users/sergejloginov/work/2021–2022/julia/lab6/tmp.gif
  @ Plots /Users/sergejloginov/.julia/packages/Plots/1RWGg/src/animation.jl:114
```

В данном случае имеем параметры a - коэффициент роста популяции, b - коэффициент рождаемости, c - коэффициент смертности. Следовательно, чем больше будет разность между b и c , тем быстрее функция будет возрастать (популяция увеличивается). И наоборот, чем больше разность между c и b , тем быстрее функция будет убывать (популяция уменьшается). Начальное значение 10 означает, что развитие популяции в нашей модели начинается с 10 особей.

Также после графика идет код для gif-анимации для данной функции.

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

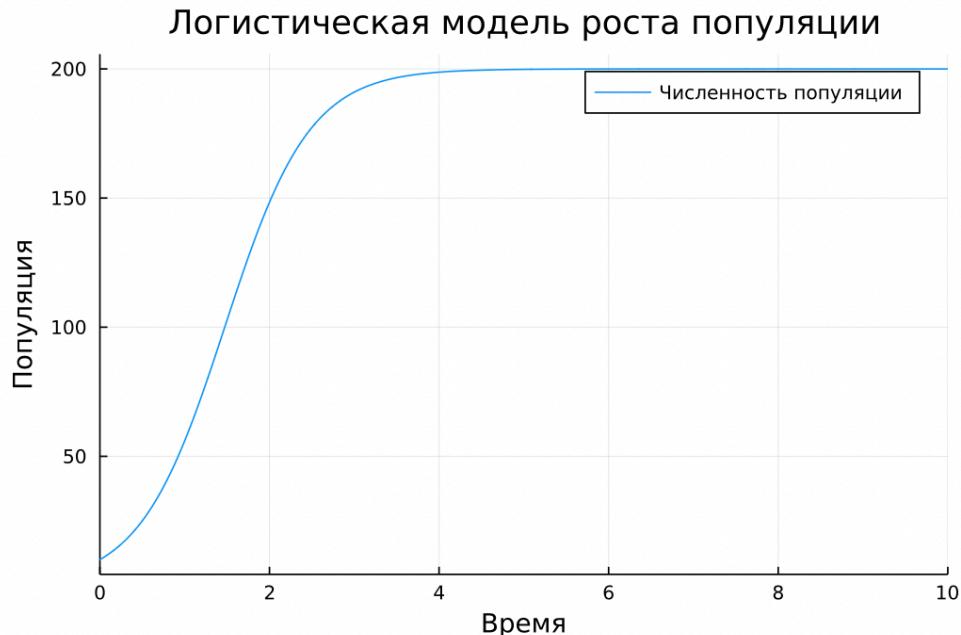
$$x' = rx\left(1 - \frac{x}{k}\right), \quad r > 0, \quad k > 0$$

Реализуем данную модель в Julia:

2

```
In [14]: r = 2
k = 200
logistic(u, p, t) = r * u * (1 - u / k)
u0 = 10
tspan = (0.0, 10.0)
prob = ODEProblem(logistic, u0, tspan)
sol = solve(prob)
plot(sol, label = "Численность популяции",
      title = "Логистическая модель роста популяции",
      xaxis = "Время", yaxis = "Популяция")
```

Out[14]:



```
In [15]: @gif for i in range(0.0, stop = 10, length = 200)
    prob = ODEProblem(logistic, u0, i)
    sol = solve(prob)
    plot(sol, label = "Численность популяции",
          title = "Логистическая модель роста популяции",
          xaxis = "Время", yaxis = "Популяция")
end
```

В данном случае мы имеем коэффициент r , отражающий скорость роста популяции, и k - потенциальная емкость популяции. Начальное значение - первичное количество особей. Были подобраны оптимальные параметры для отслеживания поведения нашей функции на графике.

Также была создана gif-анимация.

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

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

Реализация в Julia:

3

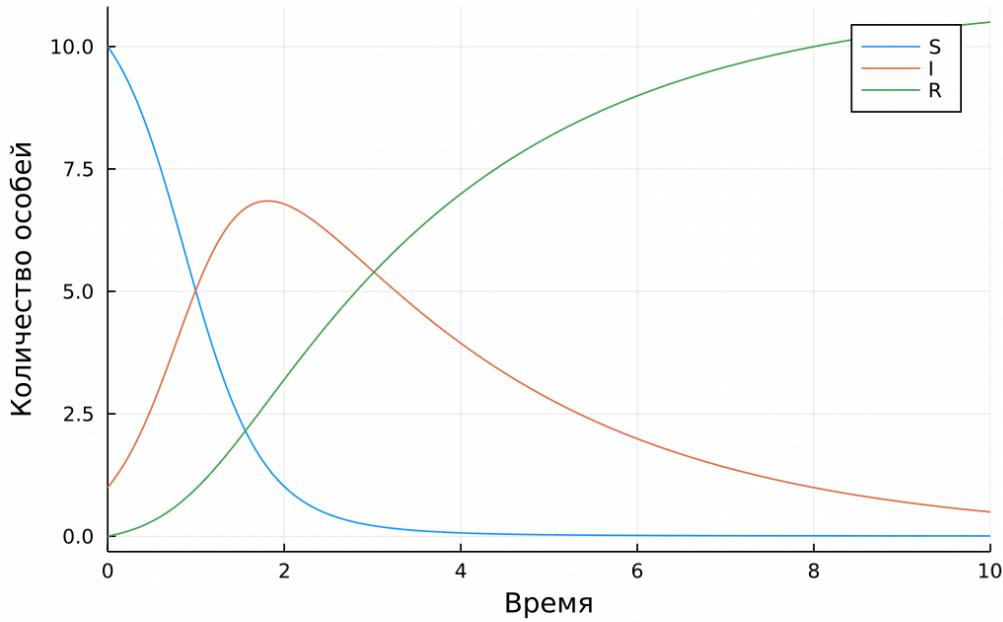
```
In [16]: function sir!(du, u, p, t)
    b, v = p
    du[1] = -b * u[2] * u[1]
    du[2] = b * u[2] * u[1] - v * u[2]
    du[3] = v * u[2]
end

u0 = [10, 1, 0]
p = (0.25, 0.35)
tspan = (0.0, 10.0)

prob = ODEProblem(sir!, u0, tspan, p)
sol = solve(prob)
plot(sol, label = ["S" "I" "R"], title = "Модель SIR",
      xaxis = "Время", yaxis = "Количество особей")
```

Out [16]:

Модель SIR



```
In [17]: @gif for i in range(0.0, stop = 10, length = 200)
    prob = ODEProblem(sir!, u0, i, p)
    sol = solve(prob)
    plot(sol, label = ["S" "I" "R"], title = "Модель SIR",
          xaxis = "Время", yaxis = "Количество особей")
end
```

В данной модели у нас есть три уравнения, представляющие три группы особей: s - восприимчивые к болезни особи, i - численность инфицированных, r - численность переболевших. Для модели возьмем 10 восприимчивых к болезни особей, 1 зараженную и 0 переболевших. Также возьмем коэффициент интенсивности контактов равный 0,25, чтобы график не был резким, и 0,35 коэффициент выздоровления по той же причине.

По графику видно, что число людей, еще не болевших, уменьшается до 0, количество переболевших растет до пика и далее уменьшается, количество переболевших растет.

Также была создана gif-анимация.

4. Реализовать модель SEIR:

$$\begin{cases} s' = -\frac{\beta}{N} si \\ e' = \frac{\beta}{N} si - \delta e \\ i = \delta e - \gamma i \\ r' = \gamma i \end{cases}$$

Реализация в Julia:

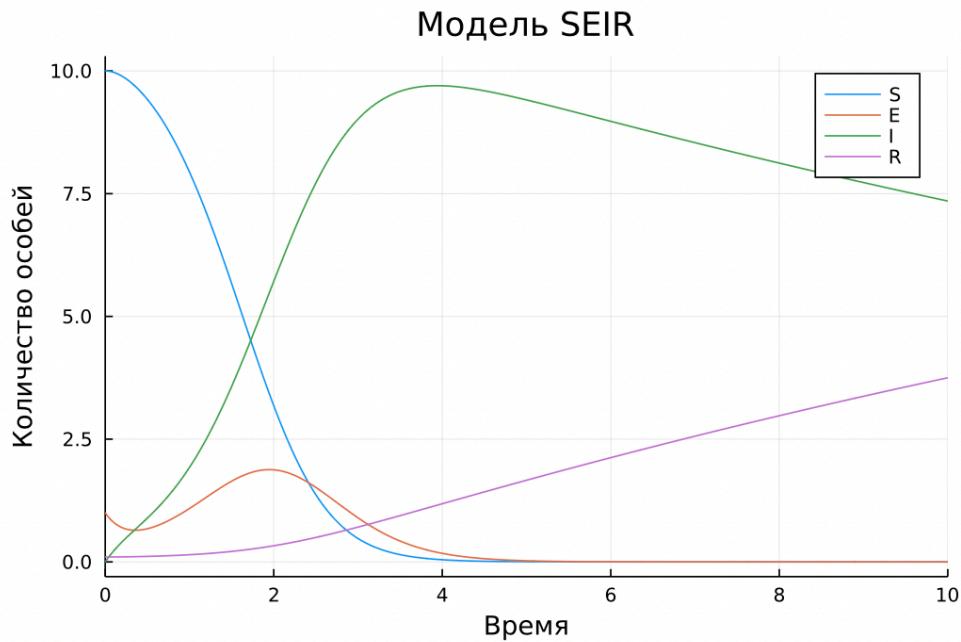
4

```
In [17]: seir! = @ode_def SEIR_Model begin
    ds = -b / n * s * i
    de = b / n * s * i - sig * e
    di = sig * e - y * i
    dr = y * i
end b n sig y

u0 = [10, 1, 0, 0.1]
p = (0.25, 1, 2.5, 0.05)
tspan = (0.0, 10.0)

prob = ODEProblem(seir!, u0, tspan, p)
sol = solve(prob)
plot(sol,label = ["S" "E" "I" "R"], title = "Модель SEIR",
      xaxis = "Время", yaxis = "Количество особей" )
```

Out[17]:



```
In [19]: @gif for i in range(0.0, stop = 10, length = 200)
    prob = ODEProblem(seir!, u0, i, p)
    sol = solve(prob)
    plot(sol,label = ["S" "E" "I" "R"], title = "Модель SEIR",
          xaxis = "Время", yaxis = "Количество особей")
end
```

Данная модель расширяет модель SIR и вводит учет инкубационного периода (второе уравнение). На основе этого можно сделать вывод, что модель SEIR производит более качественный анализ и прогноз эпидемий, тк большинство болезней имеет инкубационный период, который необходимо учитывать при моделировании.

5. Дискретная модель Лотки-Вольтерры:

Выполнить этот пункт не удалось.

6. Реализовать модель отбора на основе конкурентных отношений:

$$\begin{cases} x' = \alpha x - \beta xy \\ y' = \alpha y - \beta xy \end{cases}$$

Реализация модели в Julia:

6

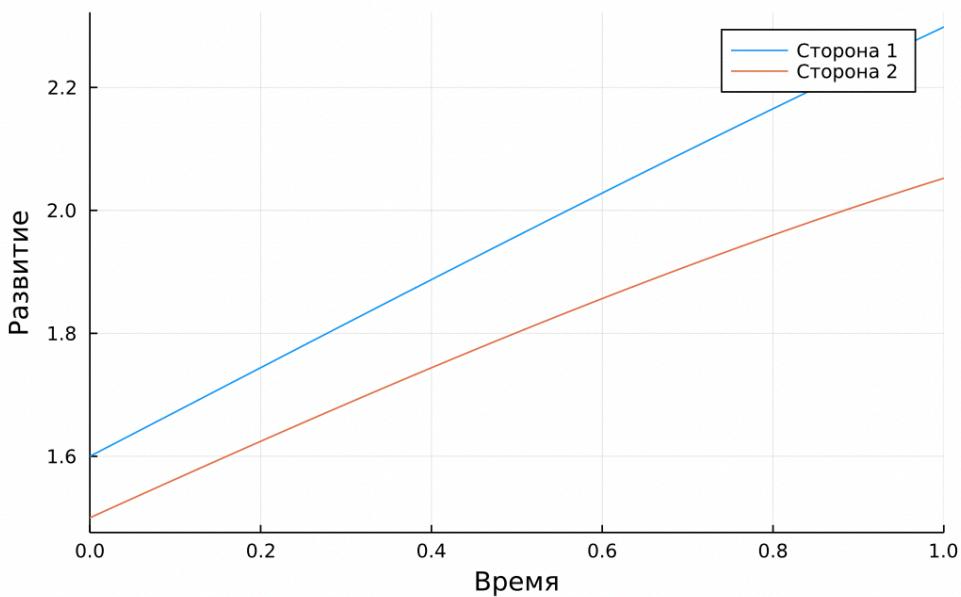
```
In [18]: conc! = @ode_def ConcRel begin
    dx = a * x - b * x * y
    dy = a * y - b * x * y
end a b

u0 = [1.6, 1.5]
p = (0.9, 0.3)
tspan = (0.0, 1.0)

prob = ODEProblem(conc!, u0, tspan, p)
sol = solve(prob)
plot(sol,label = ["Сторона 1" "Сторона 2"],
      title = "Модель конкурентных отношений",
      xaxis = "Время", yaxis = "Развитие")
```

Out[18]:

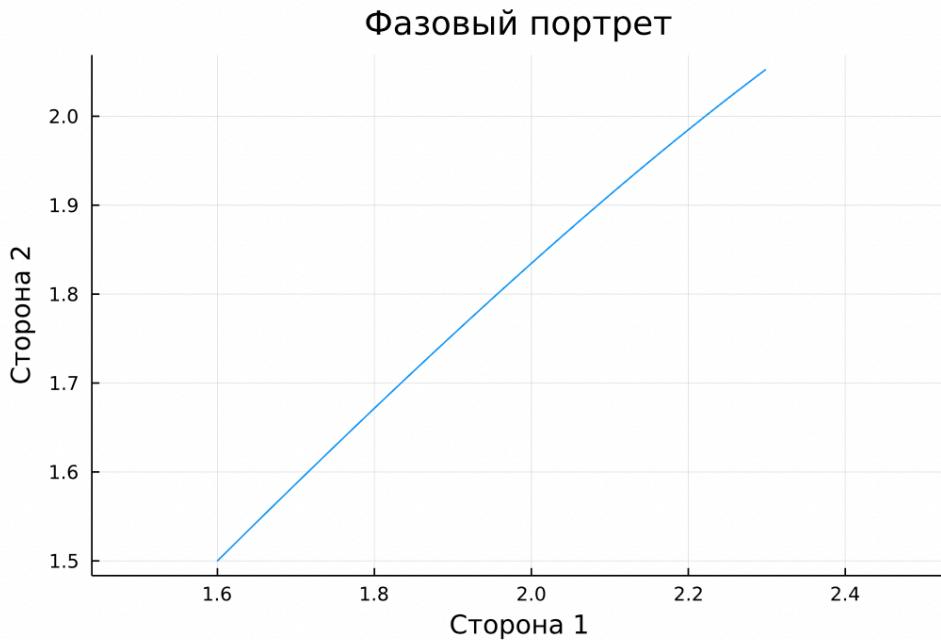
Модель конкурентных отношений



Фазовый портрет:

```
In [20]: plot(sol, vars = (1, 2), legend = false, title = "Фазовый портрет",
         xaxis = "Сторона 1", yaxis = "Сторона 2")
```

Out[20]:



В данной модели у нас есть два параметра: α - сила/масштаб конкуренции и β - количество конкурентных столкновений/встреч. При решении использованы значения параметров меньше единицы для более наглядного графика.

7. Реализовать модель консервативного гармонического осциллятора:

$$x'' + \omega_0^2 = 0, \quad x(t_0) = x_0, \quad x'(t_0) = y_0$$

Реализация в Julia:

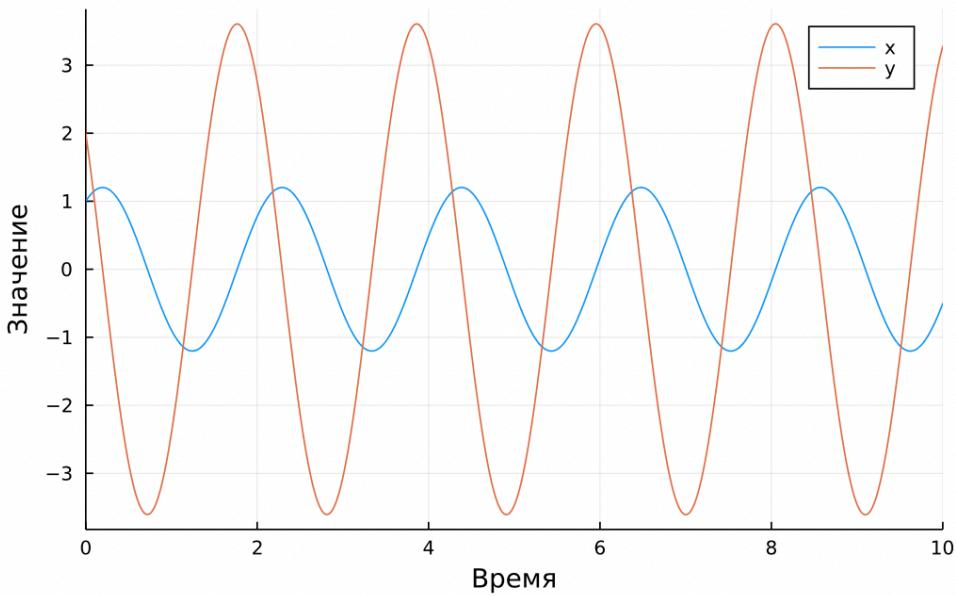
```
In [21]: garm_osc! = @ode_def Garm_Osc begin
    dx = y
    dy = -w^2 * x
end w

u0 = [1.0, 2.0]
p = 3.0
tspan = (0.0, 10.0)

prob = ODEProblem(garm_osc!, u0, tspan, p)
sol = solve(prob)
plot(sol,
    title = "Модель гармонического осциллятора",
    xaxis = "Время", yaxis = "Значение")
```

Out[21]:

Модель гармонического осциллятора

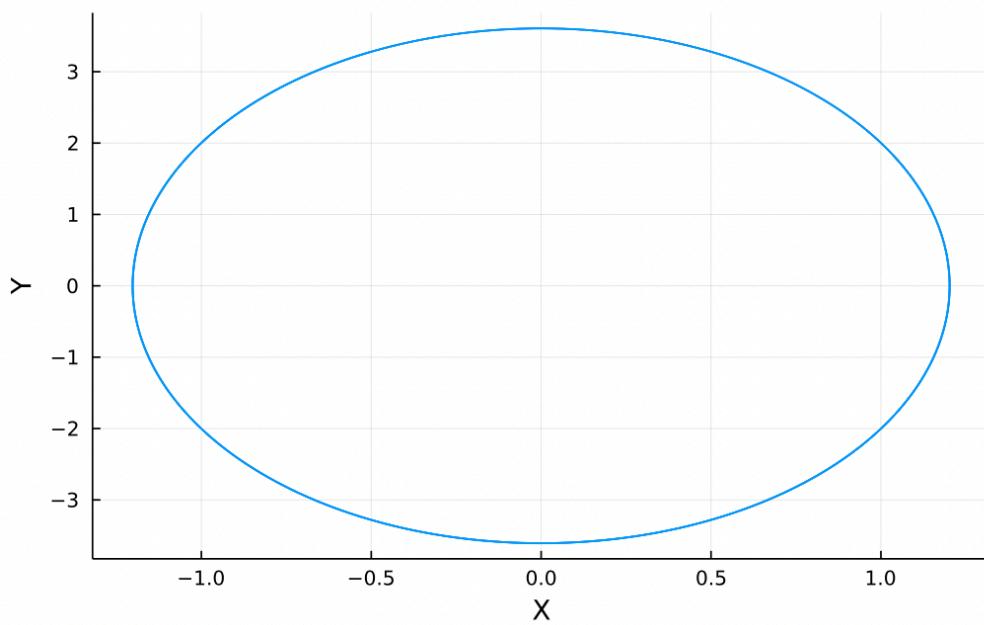


В данной модели имеем параметр ω , описывающий циклическую частоту. Чем он больше, тем чаще происходят колебания.

Фазовый портрет:

Out[22]:

Фазовый портрет



8. Реализовать модель свободных колебаний гармонического осциллятора

$$x'' + 2\gamma x' + \omega_0^2 x = 0, \quad x(t_0) = x_0, \quad x'(t_0) = y_0$$

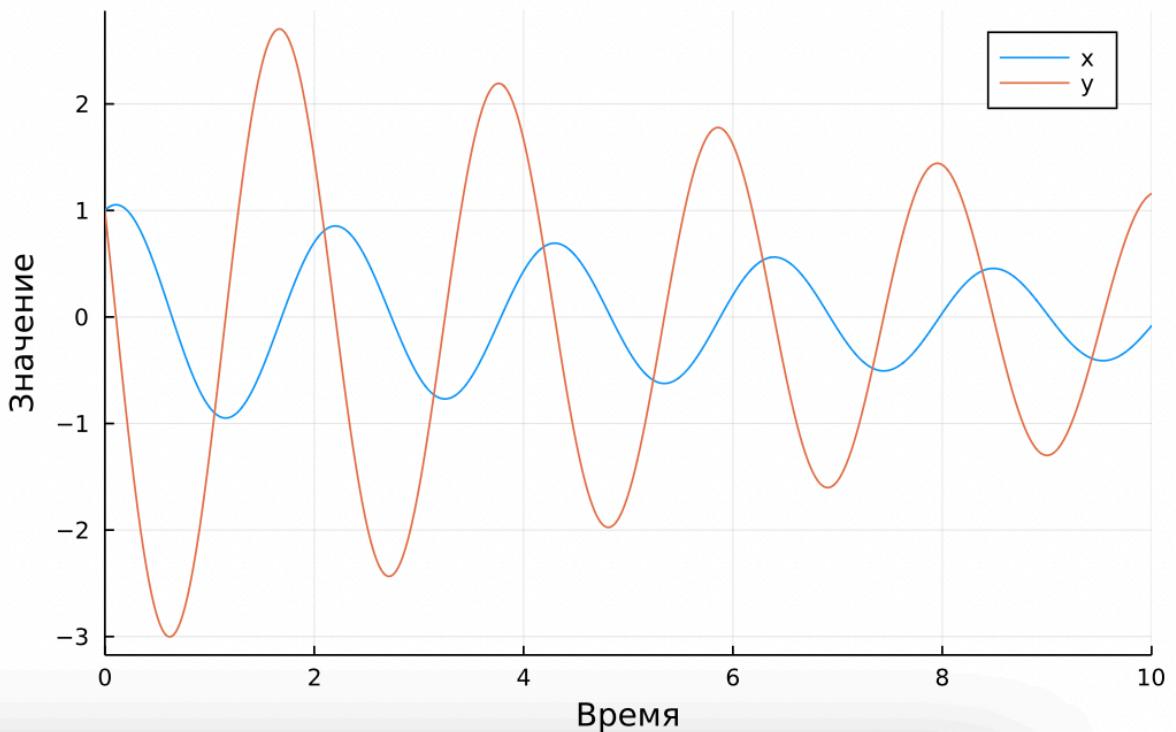
Реализация в Julia:

```
In [12]: garm_osc_mov! = @ode_def Garm_osc_m begin
    dx = y
    dy = -2 * gamma * y - w^2 * x
end gamma w

u0 = [1.0, 1.0]
p = (0.1, 3.0)
tspan = (0.0, 10.0)

prob = ODEProblem(garm_osc_mov!, u0, tspan, p)
sol = solve(prob)
plot(sol,
    title = "Модель свободных колебаний гармонического осциллятора",
    titlefont = font(10, "Times New Roman"),
    xaxis = "Время", yaxis = "Значение")
```

Out[12]: Модель свободных колебаний гармонического осциллятора

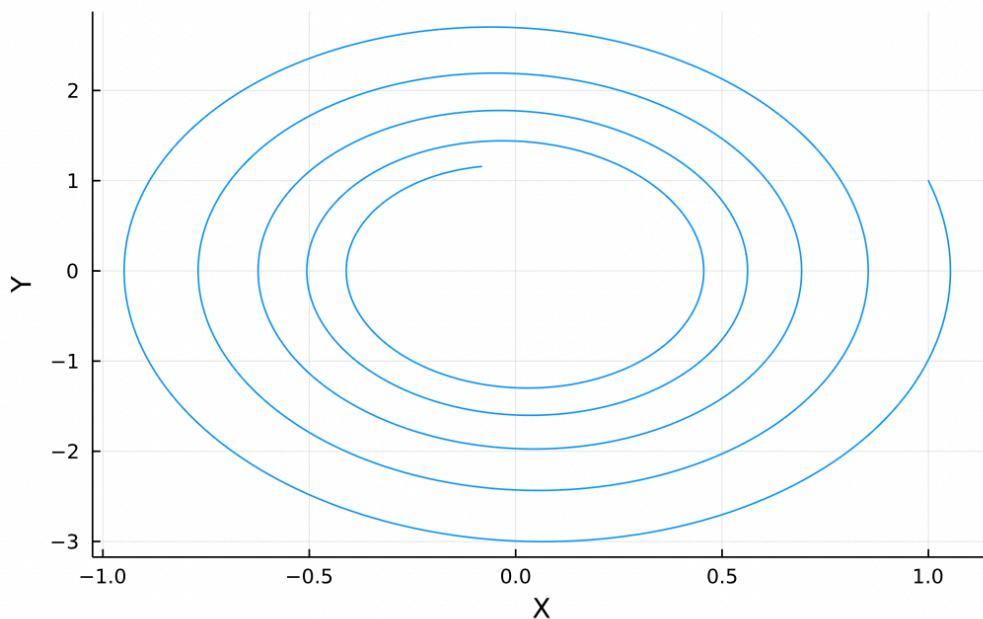


Фазовый портрет:

```
In [24]: plot(sol, vars = (1, 2), title = "Фазовый портрет",
yaxis = "Y", xaxis = "X", legend = false)
```

Out [24]:

Фазовый портрет



Данная модель расширяет предыдущую и делает ее реалистичной благодаря учету потерь системы (затухание). Данная величина описывается параметром γ , чем она больше, тем большие потери происходят и тем быстрее происходит полное затухание.

Вывод

В данной лабораторной работе были освоены пакеты для решения различных задач в непрерывном и дискретном времени