Презентация по лабораторной работе №6

Компьютерный практикум по статистическому анализу данных

Еюбоглу Тимур

22 ноября 2025 г.

Российский университет дружбы народов, Москва, Россия

Докладчик

- Еюбоглу Тимур
- Студент группы НПИбд-01-22
- Студ. билет 1032224357
- Российский университет дружбы народов имени Патриса Лумумбы

Цель лабораторной работы

- Освоить специализированные пакеты для решения задач в непрерывном и дискретном времени.

Выполнение лабораторной работы

Решение обыкновенных дифференциальных уравнений

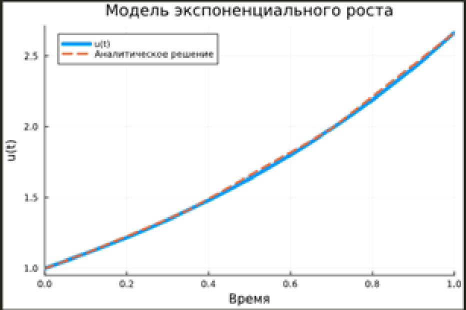
Для решения обыкновенных дифференциальных уравнений (ОДУ) в Julia можно использовать пакет diffrentialEquations.jl.

1.1. Модель экспоненциального роста

```julia
using DifferentialEquations, Plots

# 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)
# 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")
```

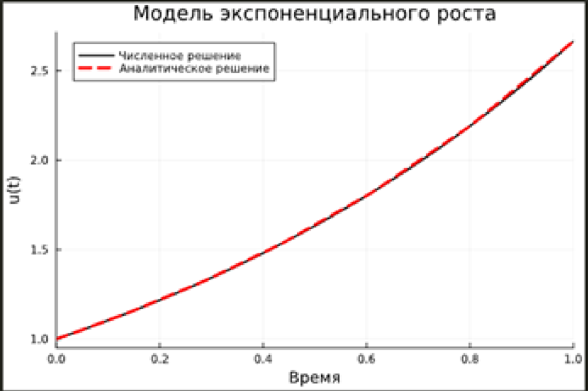
✓ 1.6s



Модель экспоненциального роста

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

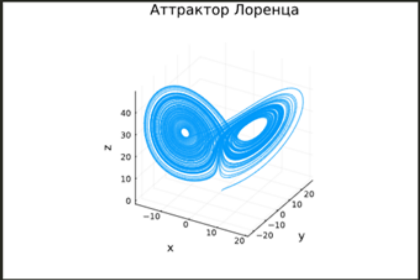
✓ 0.3s



1.2. Система Лоренца

```julia
# 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 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, Tsit5())
# Строим график
plot(sol, idxs=(1, 2, 3), lw=1, title="Аттрактор Лоренца", xaxis="x", yaxis="y", zaxis="z", legend=false)
```

3.3s



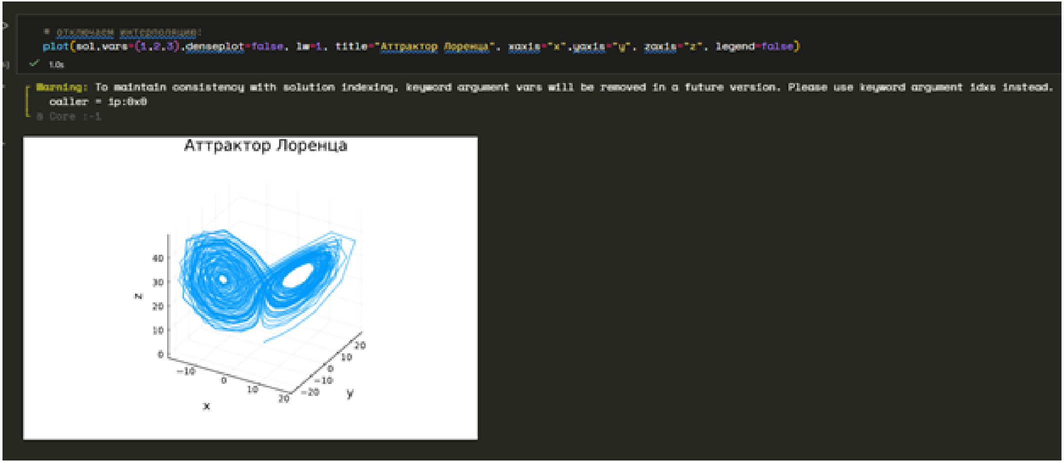
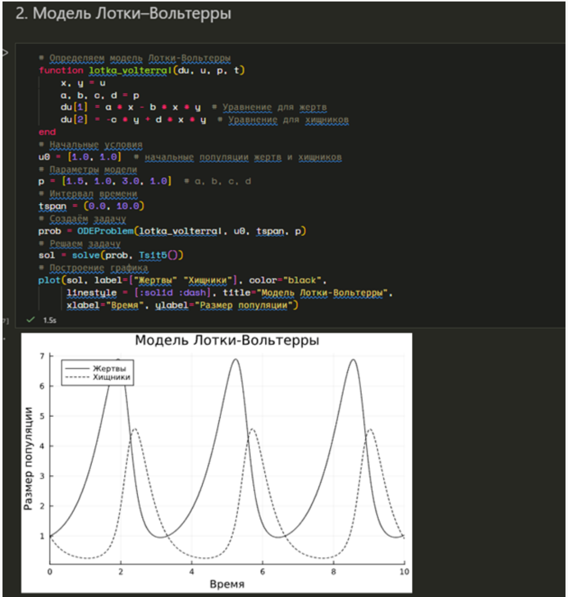


Рис. 4: Аттрактор Лоренца (интерполяция отключена)

Модель Лотки–Вольтерры



Модель Лотки–Вольтерры

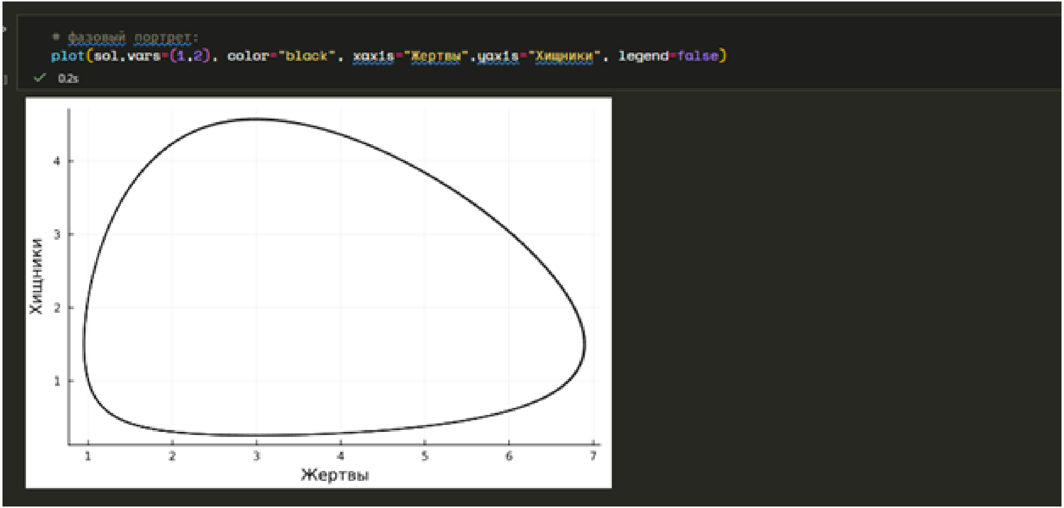


Рис. 6: Модель Лотки–Вольтерры: фазовый портрет

Самостоятельная работа

Самостоятельное выполнение

1) Реализовать и проанализировать модель роста численности изолированной популяции (модель Мальтуса). Начальные данные и параметры задать самостоятельно и пояснить их выбор. Построить соответствующие графики (в том числе с анимацией):

```julia
# Определение параметров
b = 1.0
c = 0.2
a = b - c
u0 = 1.0  # Начальная численность популяции
tspan = (0.0, 10.0)  # Интервал времени
# Описание модели
f(u, p, t) = a * u  # Уравнение роста популяции
# Задаём задачу
prob = ODEProblem(f, u0, tspan)
# Решение задачи
sol = solve(prob)
# Настройка анимации
anim = @animate for i in 1:length(sol.t)
    plot(sol.t[1:i], sol.u[1:i], linewidth=3, title="Модель Мальтуса", xlabel="Время", ylabel="Численность популяции", legend=false)
end
# Сохранение анимации в файл
gif(anim, "malthus_population_growth.gif", fps=15)
```

2.8s

```
[ Info: Saved animation to c:\Users\timur\OneDrive\Desktop\md\grafics\malthus_population_growth.gif
```

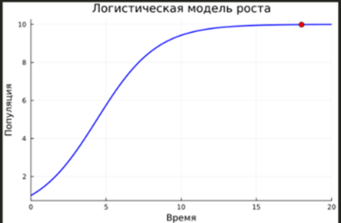
Самостоятельная работа

выбор. Построить соответствующие графики (в том числе с анимацией):

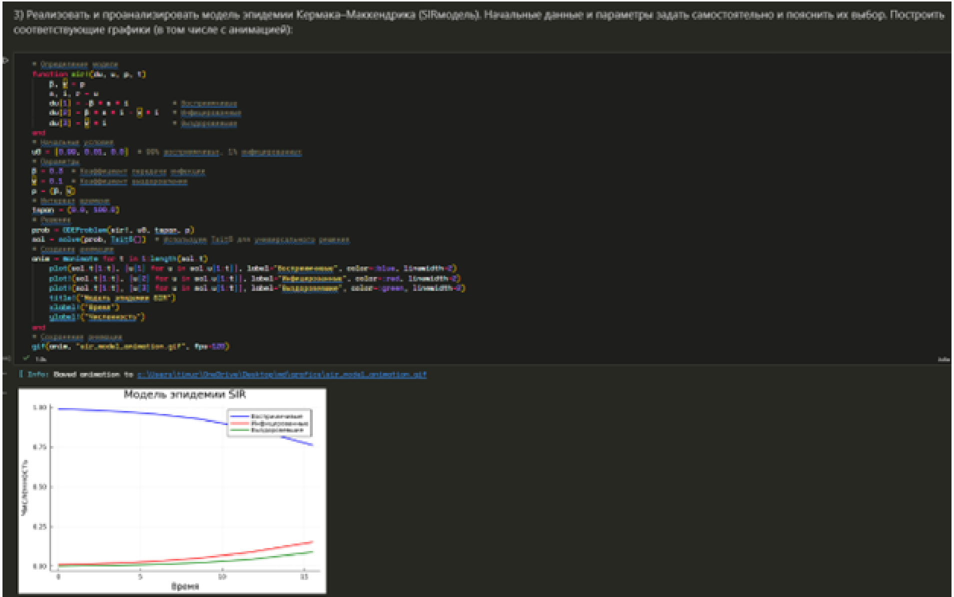
```julia
# Определение модели
function logistic!(du, u, p, t)
    r, k = p
    du[1] = r * u[1] * (1 - u[1] / k)
end
# Начальные условия
u0 = [1.0]  # Начальная численность популяции
# Параметры
r = 0.5  # Коэффициент роста
k = 10.0  # Емкость экосистемы
p = (r, k)
# Интервал времени
tspan = (0.0, 20.0)
# Решение
prob = ODEProblem(logistic!, u0, tspan, p)
sol = solve(prob, Tsit5())  # Используем Tsit5 для не-жёсткой задачи
# Создание объекта анимации
anim = Animation()
# Построение анимации
for t in 0:0.5:20
    plot(sol, vars=(0, 1), linewidth=2, color=:blue, title="Логистическая модель роста",
         xlabel="Время", ylabel="Популяция", legend=false)
    scatter!([t], [sol(t)[1]], color=:red, label="", markersize=5)  # Добавляем текущую точку
    frame(anim)  # Добавляем кадр в анимацию
end
# Сохранение анимации
gif(anim, "logistic_growth.gif", fps=10)  # Сохраняем анимацию в файл
```

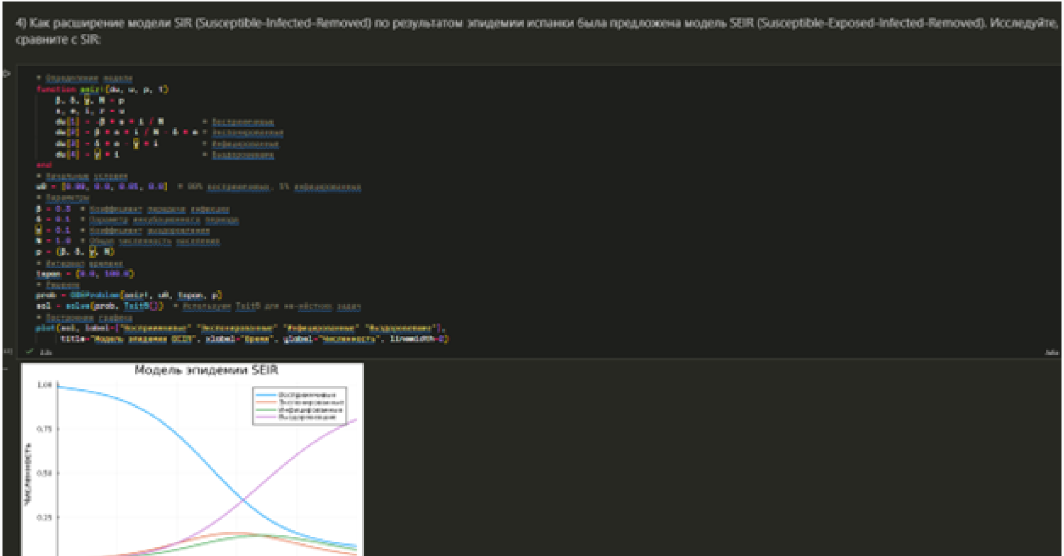
3.4s

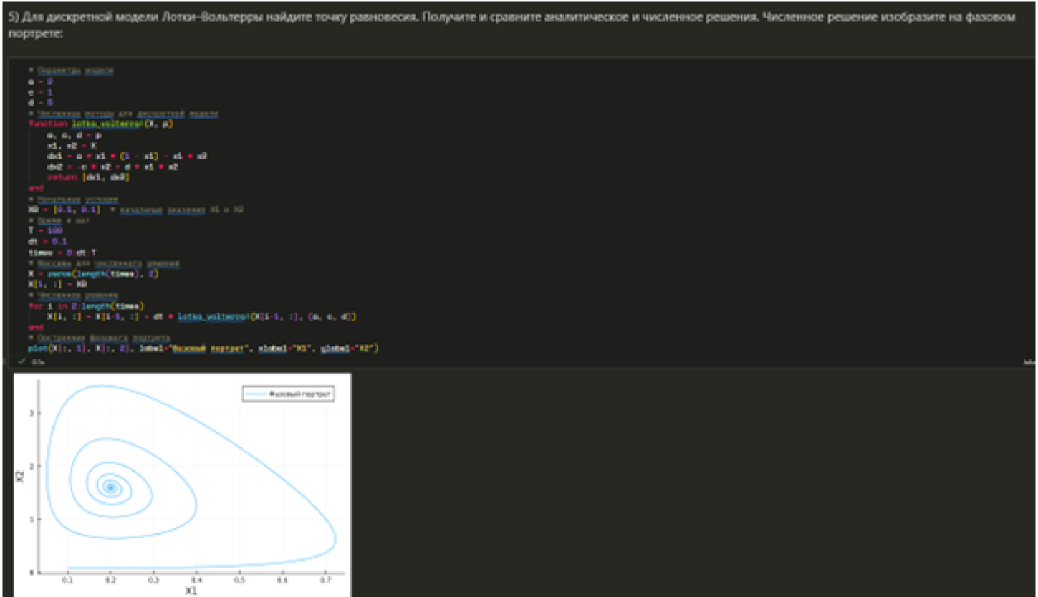
```
[ Info: Saved animation to c:\Users\timur\OneDrive\Desktop\md\grafics\logistic_growth.gif
```



Самостоятельная работа

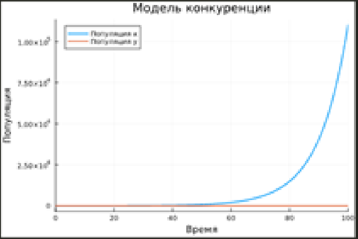



4) Как расширение модели SIR (Susceptible-Infected-Removed) по результатам эпидемии испанки была предложена модель SEIR (Susceptible-Exposed-Infected-Removed). Исследуйте, сравните с SIR.
Модель эпидемии SEIR


5) Для дискретной модели Лотки–Вольтерры найдите точку равновесия. Получите и сравните аналитическое и численное решения. Численное решение изобразите на фазовом портрете:
T = 100
dt = 0.1
times = 0:dt:T
Фазовый портрет
0.1
0.2
0.3
0.4
0.5
0.6
0.7
x1
x2

6) Реализовать на языке Julia модель отбора на основе конкурентных отношений. Начальные данные и параметры задать самостоятельно и пояснить их выбор. Построить соответствующие графики (в том числе с анимацией) и фазовый портрет:

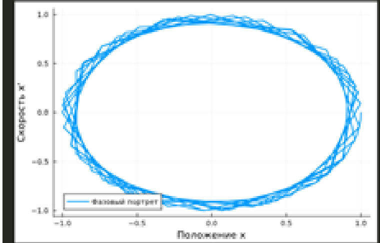
```julia
# Модель конкуренции
function competition!(du, u, p, t)
    α, β = p
    x, y = u
    du[1] = α * x - β * x * y
    du[2] = α * y - β * x * y
end
# Начальные условия
u0 = [10.0, 5.0]  # Начальные популяции
# Параметры
α = 0.1
β = 0.01
p = (α, β)
# Интервал времени
tspan = (0.0, 100.0)
# Решение
prob = ODEProblem(competition!, u0, tspan, p)
sol = solve(prob, Tsit5())
# Построение графиков
plot(sol, label=["Популяция x" "Популяция y"], title="Модель конкуренции", xlabel="Время", ylabel="Популяция", linewidth=2)
```



Самостоятельная работа

7) Реализовать на языке Julia модель консервативного гармонического осциллятора. Начальные параметры подобрать самостоятельно, выбор пояснить. Построить соответствующие графики (в том числе с анимацией) и фазовый портрет:

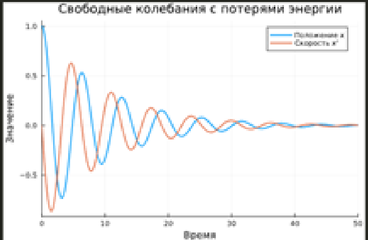
```julia
# Модель гармонического осциллятора
function harmonic_oscillator!(du, u, p, t)
    ω0 = p[1]  # Циклическая частота
    du[1] = u[2]  # x' = y
    du[2] = -ω0^2 * u[1]  # y' = -ω0^2 * x
end
# Начальные условия
x0 = 1.0    # Начальное положение (м)
y0 = 0.0    # Начальная скорость (м/с)
u0 = [x0, y0]  # Начальные значения: x0 и y0
# Параметры
ω0 = 1.0    # Циклическая частота (рад/с)
p = [ω0]    # Параметры системы
# Интервал времени
tspan = (0.0, 60.0)
# Создание задачи для решения
prob = ODEProblem(harmonic_oscillator!, u0, tspan, p)
# Решение задачи
sol = solve(prob, Tsit5())
# Построение графиков (положение и скорость от времени)
plot(sol, label=["Положение x" "Скорость x'"], title="Гармонический осциллятор", xlabel="Время (t)", ylabel="Значение", linewidth=2)
# Построение фазового портрета (x, x')
plot(sol[1, :], sol[2, :], label="Фазовый портрет", xlabel="Положение x", ylabel="Скорость x'", linewidth=2)
```



8) . Реализовать на языке Julia модель свободных колебаний гармонического осциллятора. Начальные параметры подобрать самостоятельно, выбор пояснить. Построить соответствующие графики (в том числе с анимацией) и фазовый портрет:

```julia
# Модель свободных колебаний с потерями
function damped_oscillator!(du, u, p, t)
    γ, ω0 = p
    du[1] = u[2]
    du[2] = -2 * γ * u[2] - ω0^2 * u[1]
end
# Начальные условия
u0 = [1.0, 0.0]  # Начальное положение и скорость
# Параметры
γ = 0.1
ω0 = 1.0
p = (γ, ω0)
# Интервал времени
tspan = (0.0, 50.0)
# Решение
prob = ODEProblem(damped_oscillator!, u0, tspan, p)
# Используем более подходящий алгоритм, например, Tsit5
sol = solve(prob, Tsit5())
# Построение графиков
plot(sol, label=["Положение x" "Скорость x'"], title="Свободные колебания с потерями энергии", xlabel="Время", ylabel="Значение", linewidth=2)
```

✓ 0.2s



Вывод

Вывод

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

Список литературы. Библиография

[1] Julia Documentation: https://docs.julialang.org/en/v1/