

# Презентация по лабораторной работе №6

Компьютерный практикум по статистическому анализу данных

---

Еюбоглу Тимур

22 ноября 2025 г.

Российский университет дружбы народов, Москва, Россия

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

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

## Выполнение лабораторной работы

---

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

# Модель экспоненциального роста

## 1.1. Модель экспоненциального роста

```
using DifferentialEquations, Plots
```

```
* задаём описание модели с начальными условиями:
```

```
a = 0.08
```

```
f(u,p,t) = a*u
```

```
u0 = 1.0
```

```
* задаём интервал времени:
```

```
tspan = (0.0,1.0)
```

```
* решение:
```

```
prob = ODEProblem(f,u0,tspan)
```

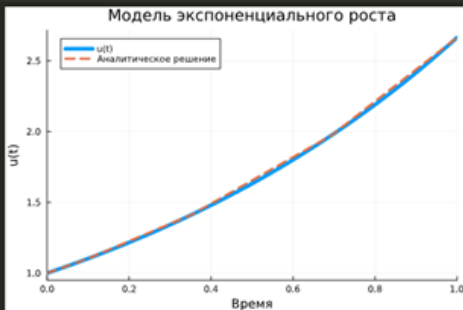
```
sol = solve(prob)
```

```
* строим график:
```

```
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")
```

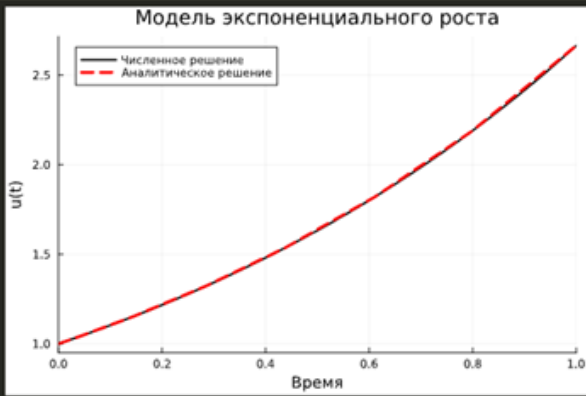
✓ 1.6s



## 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

```
* Задаём описание модели
function lorenzi(du, u, p, t)
    θ, ρ, β = p
    du[1] = θ * (u[2] - u[1])
    du[2] = u[1] * (ρ - 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)

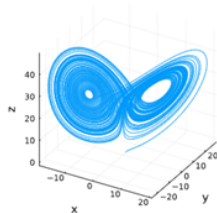
* Задаём интервал времени
tspan = (0.0, 100.0)

* Решаем
prob = ODEProblem(lorenzi, 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

Аттрактор Лоренца





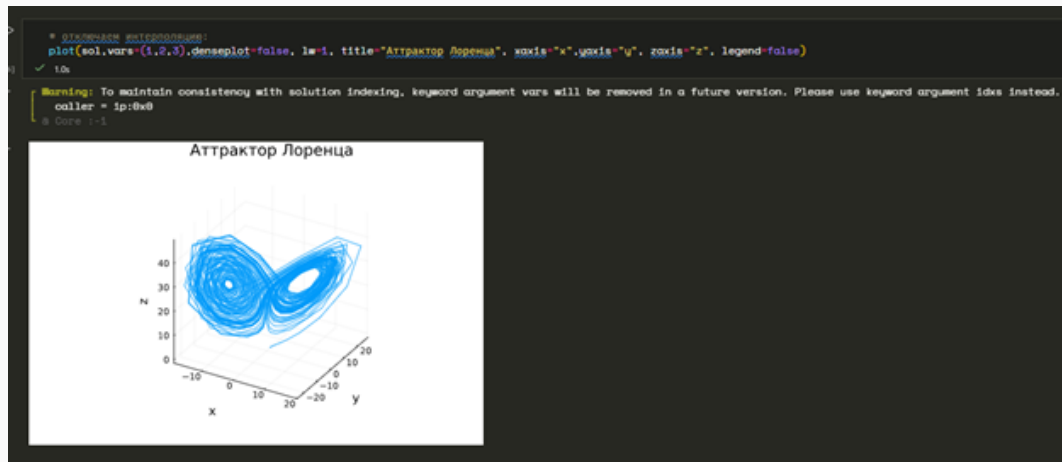


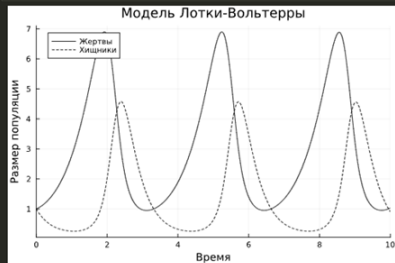
Рис. 4: Аттрактор Лоренца (интерполяция отключена)

## 2. Модель Лотки–Вольтерры

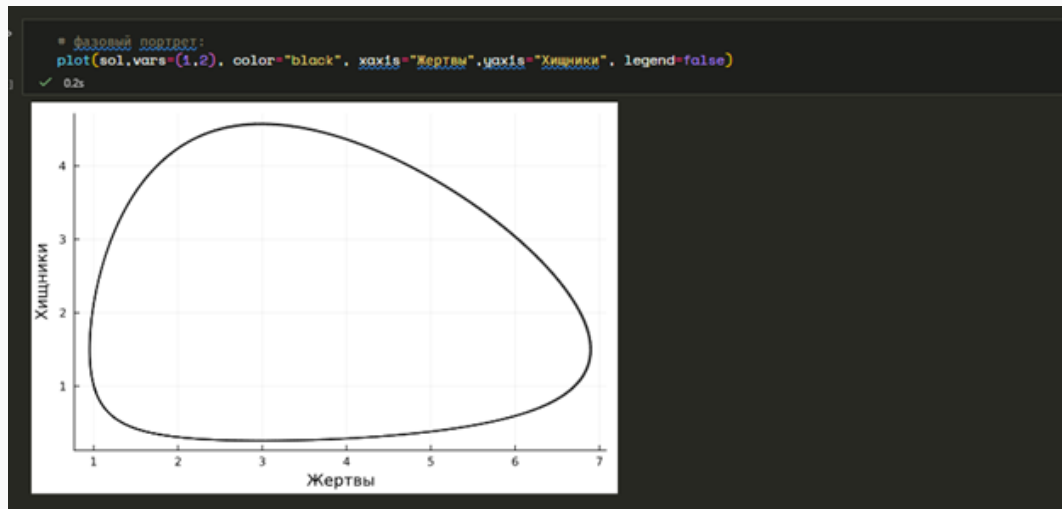
```
% Определяем модель Лотки-Вольтерры
function lotka_volterra(du, u, p, t)
    x, y = u
    a, b, c, d = p
    du[1] = a * x - b * x * y % Уравнение для жертв
    du[2] = -c * y + d * x * y % Уравнение для хищников
end

% Начальные условия
u0 = [1.0, 1.0] % начальные популяции жертв и хищников
% Параметры модели
p = [1.5, 1.0, 3.0, 1.0] % a, b, c, d
% Интервал времени
tspan = (0.0, 10.0)
% Создаем задачу
prob = ODEProblem(lotka_volterra, u0, tspan, p)
% Решаем задачу
sol = solve(prob, Tsit5())
% Построение графика
plot(sol, label=["Жертвы", "Хищники"], color="black",
      linestyle = [:solid :dash], title="Модель Лотки-Вольтерры",
      xlabel="Время", ylabel="Размер популяции")
```

✓ 1.5s



## Модель Лотки-Вольтерры



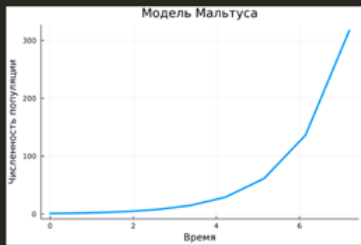
## Самостоятельное выполнение

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):

```
* Описание 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
* Задача задачи
prob = ODEProblem(f, u0, tspan)
* Решение задачи
sol = solve(prob)
* Настройка анимации
anim = animate for i in 1:length(sol.t)
| plot(sol.t[i:i], sol.u[i: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)
```

✓ 28s

[ Info: Saved animation to [file:///Users/timur10n@rive/Desktop/animatic/malthus\\_population\\_growth.gif](file:///Users/timur10n@rive/Desktop/animatic/malthus_population_growth.gif)

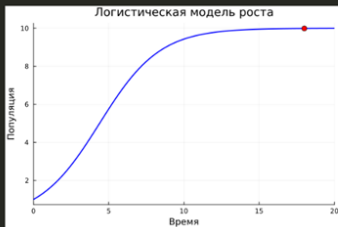


# Самостоятельная работа

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

```
* Описание модели
function logistic1(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 * Численность насыщения
p = (r, k)
* Интервал времени
tspan = (0.0, 20.0)
* Решение
prob = ODEProblem(logistic1, u0, tspan, p)
sol = solve(prob, Tsit5()) * Используем Tsit5 для не-жесткой 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=6) * Добавляем текущую точку
    frame(anim) * Добавляем кадр = анимация
end
* Сохранение анимации
gif(anim, "logistic_growth.gif", fps=10) * Сохранение анимации = файл
```

[ Info: Saved animation to [C:\Users\timur\OneDrive\Desktop\mclaratica\logistic\\_growth.gif](file:///C:/Users/timur/OneDrive/Desktop/mclaratica/logistic_growth.gif)



## Самостоятельная работа

3) Реализовать и проанализировать модель эпидемии Кермака–Маккендрика (SIR-модель). Начальные данные и параметры задать самостоятельно и пояснить их выбор. Построить соответствующие графики (в том числе с анимацией):

```

# Update the weights
function adapt(w, u, p, t)
   $\hat{p} = p$ 
   $a, i, r = u$ 
  do[1] =  $\hat{p} * a * i$  # Background
  do[2] =  $\hat{p} * a * i - \hat{p} * i$  # Background
  do[3] =  $\hat{p} * i$  # Background
end

# Generate points
w0 = [0.00, 0.00, 0.0] + 200 * randn(3, 1) # Background
# Generate
p = 0.5 # Background points
u = 0.5 # Background
p = [p, u]

# Weighted points
taper = [0.0, 100.0]

# Points
prob = GEPProblem(sir, w, taper, p)
sol = solve(prob, [t0, C]) # Background [t0, C] are parameters

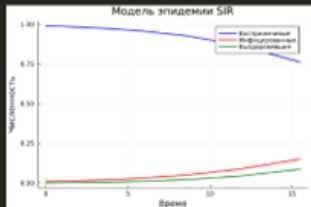
# Generate points
oria = Vector{Tuple{Int, Int, Int}}
plot!(t[1:t], [w[1] for w in sol.w[t]], label="Background", color=:blue, linewidth=2)
plot!(sol.w[t, 1], [w[2] for w in sol.w[t, 1]], label="Background", color=:red, linewidth=2)
plot!(sol.w[t, 2], [w[2] for w in sol.w[t, 2]], label="Background", color=:green, linewidth=2)
title("Weighted points GEP")
xlabel("t[year]")
ylabel("Background")

end

# Generate points
gif(oria, "sir_model_animation.gif", fps=500)

```

[ Info: Based animation to [g:\vegetation\02\Drive\Desktop\modelling\air\\_model\\_animation.tif](g:\vegetation\02\Drive\Desktop\modelling\air_model_animation.tif)



# 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:

```
# Расширенная модель
function seir(t0, m, p, t)
    S, E, I, R = p
    A, B, C, D = m
    dS(t) = -B * S * I / N      # Восприимчивость
    dE(t) = B * S * I / N - C * E # Экспонированность
    dI(t) = C * E - D * I      # Инфицированность
    dR(t) = D * I              # Выздоровевшие
end

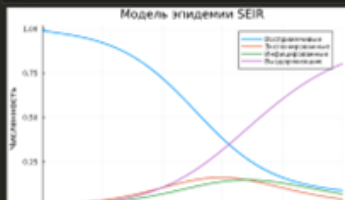
# Начальные условия
u0 = [0.99, 0.0, 0.01, 0.0] # 99% восприимчивых, 1% инфицированных

# Параметры
B = 0.5 # Коэффициент передачи инфекции
C = 0.1 # Коэффициент экспонирования
D = 0.1 # Коэффициент выздоровления
N = 1.0 # Общая численность населения
p = (S, E, I, R)

# Временной диапазон
tspan = (0.0, 100.0)

# Решение
prob = ODEProblem(seir, u0, tspan, p)
sol = solve(prob, Tsit5()) # Используем Tsit5 для не-жесткой задачи

# Построение графика
plot(sol, labels=["Восприимчивость", "Экспонированность", "Инфицированность", "Выздоровевшие"],
      title="Модель эпидемии SEIR", xlabel="Время", ylabel="Численность", linewidth=2)
```



## Самостоятельная работа

5) Для дискретной модели Лотки-Вольтерры найдите точку равновесия. Получите и сравните аналитическое и численное решения. Численное решение изобразите на фазовом портрете:

```
% Параметры модели
a = 2
p = 1
d = 0.5

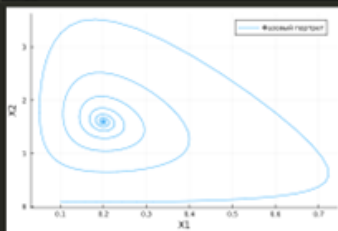
% Функция для решения ЛВ
function [dx1, dx2] = lotka_voltterra(X, p)
    a, a, d = p
    x1, x2 = X
    dx1 = a * x1 * (1 - x1) - x1 * x2
    dx2 = -p * x2 + d * x1 * x2
    return [dx1, dx2]
end

% Начальные условия
X0 = [0.5, 0.5] % начальные значения X1 и X2
% Время и шаг
T = 100
dt = 0.1
times = 0:dt:T

% Массив для хранения решения
X = zeros(length(times), 2)
X(1, :) = X0

% Численное решение
for i in 2:length(times)
    X(i, :) = X(i-1, :) + dt * lotka_voltterra(X(i-1, :), (a, p, d))
end

% Построение фазового портрета
plot(X(:, 1), X(:, 2), 'b'); hold on; title('Фазовый портрет'); xlabel('X1'); ylabel('X2')
```





## Самостоятельная работа

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.

```
# Модель конкуренции
function competition!(du, u, p, t)
    H, β = p
    x, y = u
    du[1] = H * x - β * x * x - y
    du[2] = -H * y - β * x * x - y
end

# Начальные условия
u0 = [10.0, 5.0] # Начальные популяции

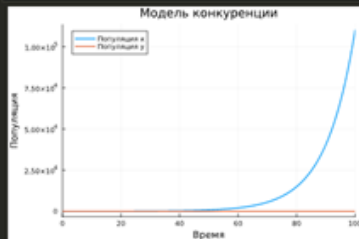
# Параметры
H = 0.1
β = 0.01
p = (H, β)

# Интервал времени
tspan = (0.0, 100.0)

# Решение
prob = ODEProblem(competition!, u0, tspan, p)
sol = solve(prob, Tsit5())

# Построение графика
plot(sol, label=["Популяция x" "Популяция y"], title="Модель конкуренции", xlabel="Время", ylabel="Популяция", linewidth=2)

✓ 7.5s
```



## 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:

```
* Модель гармонического осциллятора
function harmonic_oscillator!(du, u, p, t)
    u0 = p[1] * cos(2π * t)
    du[1] = u[2] * π * p[2]
    du[2] = -u0 * π * p[2] * π * p[2]
end

* Начальные условия
u0 = 1.0 * π * p[1]
u0 = 0.0 * π * p[2]
u0 = [u0, u0] * π * p[1]

* Параметры
u0 = 1.0 * π * p[1]
p = [u0] * π * p[2]

* Временная область
tspan = (0.0, 50.0)

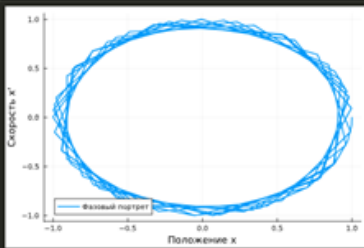
* Создание задачи для решения
prob = ODEProblem(harmonic_oscillator!, u0, tspan, p)

* Решение задачи
sol = solve(prob, Tsit5())

* Построение графиков (Положение x * Скорость x')
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)

0.3
```



## Самостоятельная работа

В) . 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:

```
# Модель свободных колебаний с потерями
function damped_oscillator!(du, u, p, t)
    @. u[1] = p
    du[1] = u[2]
    du[2] = -2 * u[2] - u[1]^2 * u[1]
end

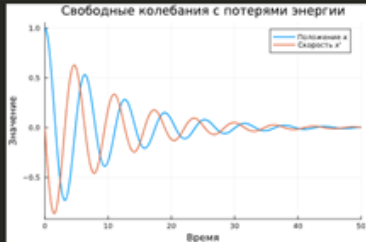
# Начальные условия
u0 = [1.0, 0.0] # Начальное положение = скорость
# Параметры
γ = 0.1
ω0 = 1.0
p = (γ, ω0)

# Интервал времени
tspan = (0.0, 50.0)

# Проблема
prob = ODEProblem{damped_oscillator!, u0, tspan, p}

# Используем более продвинутый алгоритм - калинин - Tsit5
sol = solve(prob, Tsit5())

# Построение графика
plot(sol, label=["Положение x" "Скорость x'"], title="Свободные колебания с потерями энергии", xlabel="Время", ylabel="Значение", linewidth=2)
```



## 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/>