

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

Цель лабораторной работы

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

Выполнение лабораторной работы

Решение обыкновенных дифференциальных уравнений

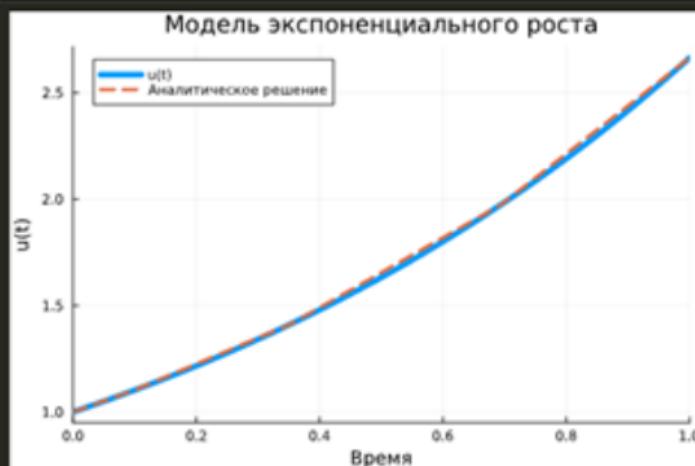
Для решения обыкновенных дифференциальных уравнений (ОДУ) в Julia можно использовать пакет `differentialEquations.jl`.

Модель экспоненциального роста

1.1. Модель экспоненциального роста

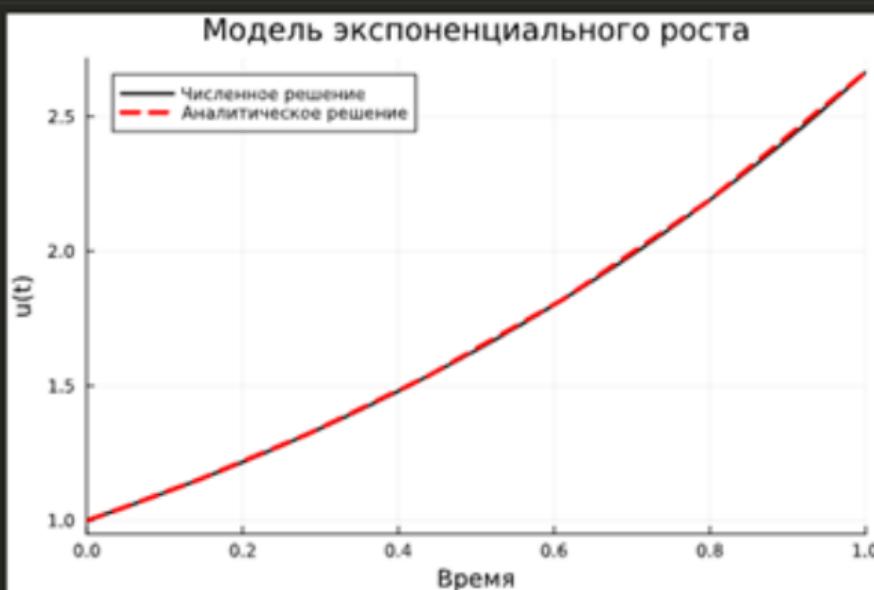
```
using DifferentialEquations, Plots

# 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", xlabel="Время", ylabel="u(t")",label="u(t")
plot!(sol.t, t->1.0*exp(a*t),lw=3,ls=:dash,label="Аналитическое решение")
```



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", xlabel="Время", ylabel="u(t)",label="Численное решение")  
plot(sol.t, t->1.0*exp(a*t),lw=3,ls=:dash,color="red",label="Аналитическое решение")
```



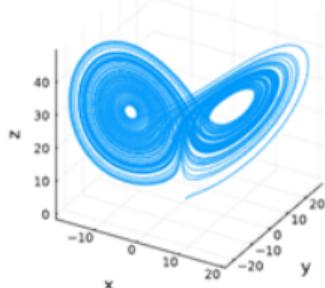
Система Лоренца

1.2. Система Лоренца

```
> % Задаем описание модели
function lorenzi(du, u, p, t)
    g, β = p
    du[1] = g * (u[2] - u[1])
    du[2] = u[1] * (β - u[3]) - u[2]
    du[3] = u[1] * u[2] - β * u[3]
end
% Задаем начальное условие
u0 = [1.0, 0.0, 0.0]
% Задаем значения параметров
p = (10, 28, 8 / 3)
% Задаем интервал времени
tspan = (0.0, 100.0)
% Решение
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



Система Лоренца

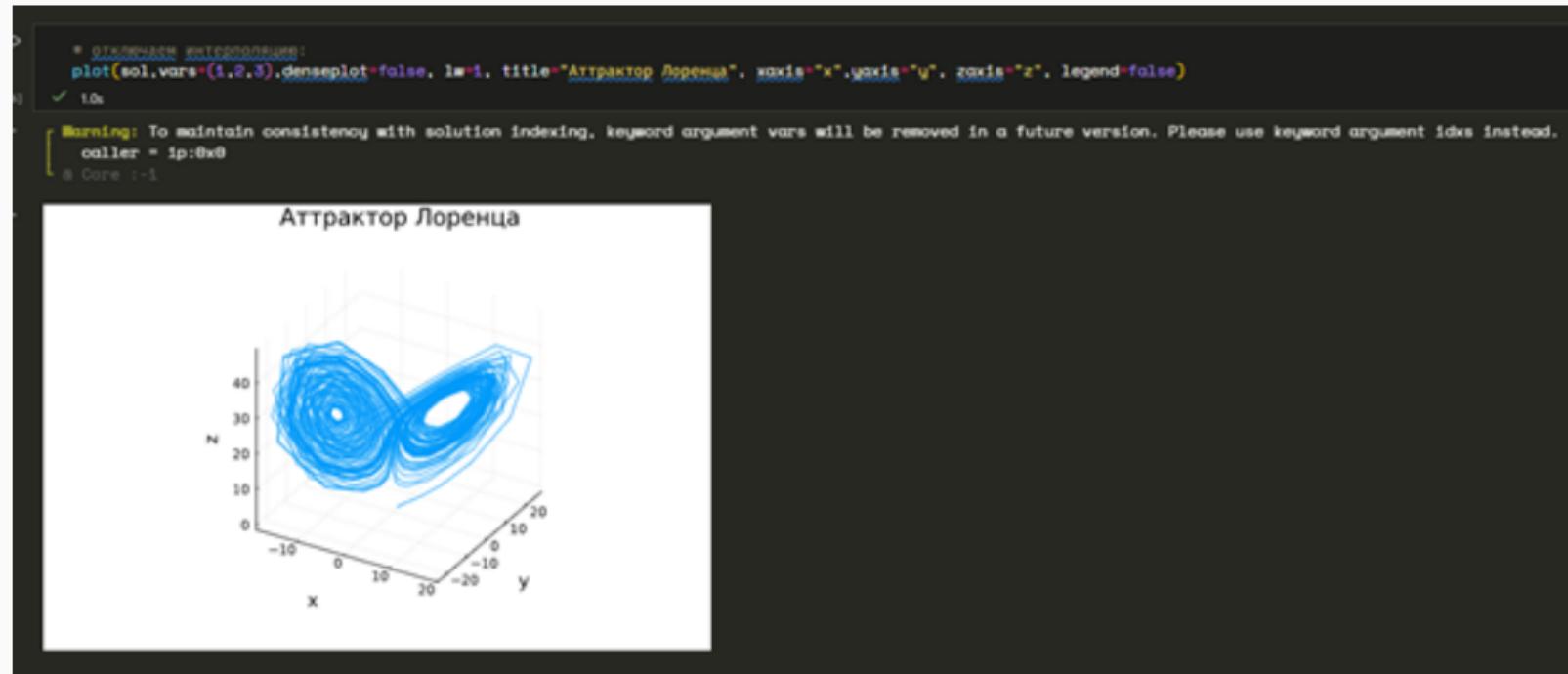
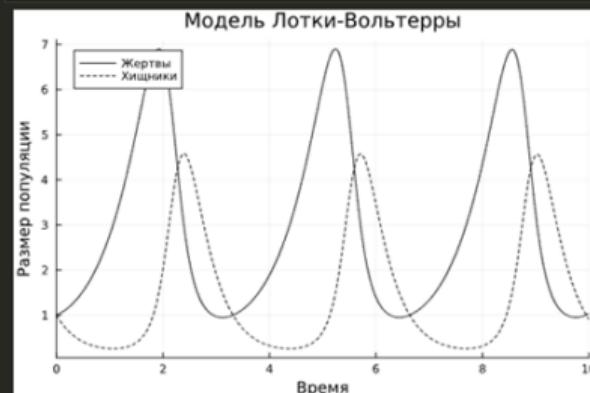


Рис. 4: Аттрактор Лоренца (интерполяция отключена)

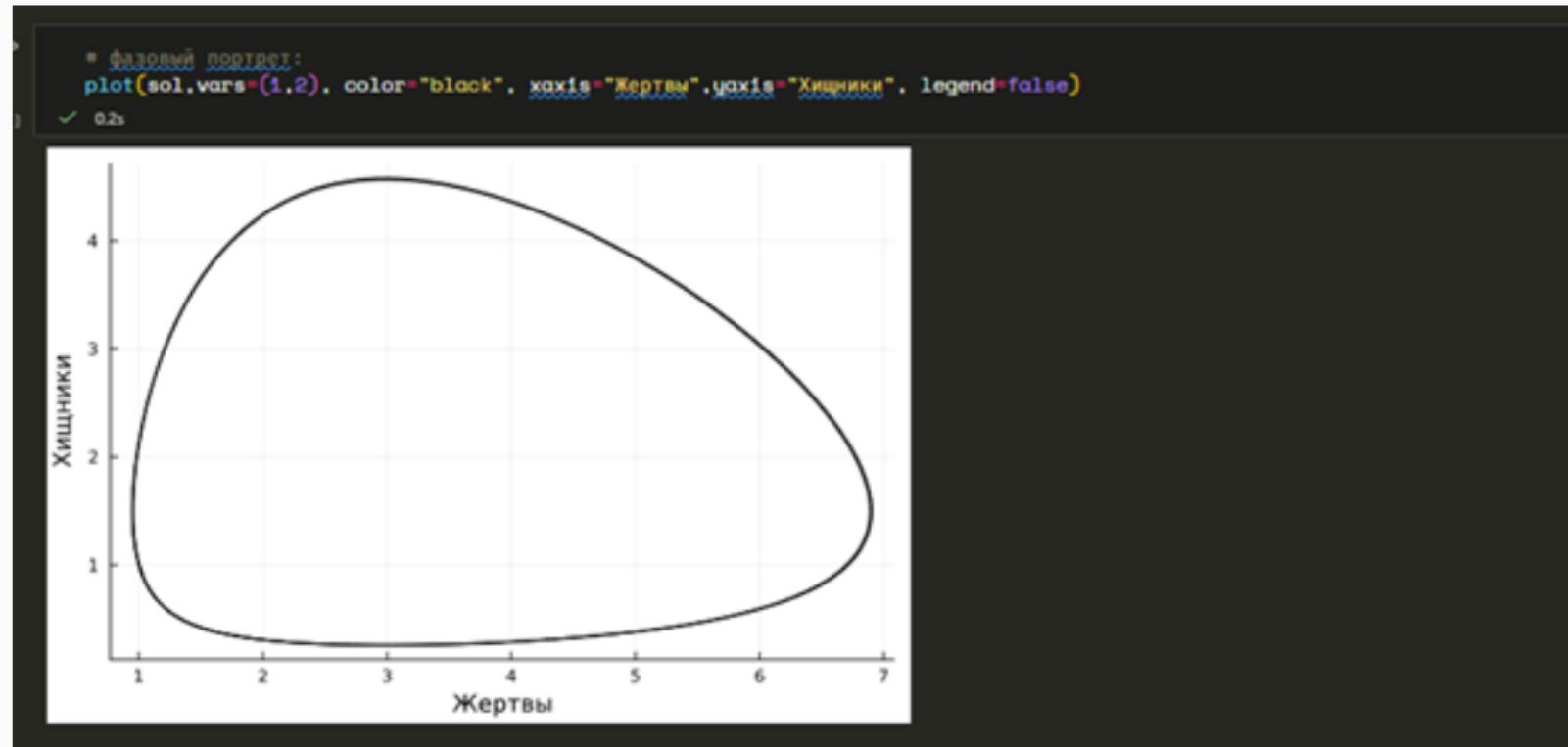
Модель Лотки–Вольтерры

2. Модель Лотки–Вольтерры

```
< определяем модель Лотки–Вольтерры
function lotka_volterra1(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 подудания 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_volterra1, 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")
```



Модель Лотки–Вольтерры

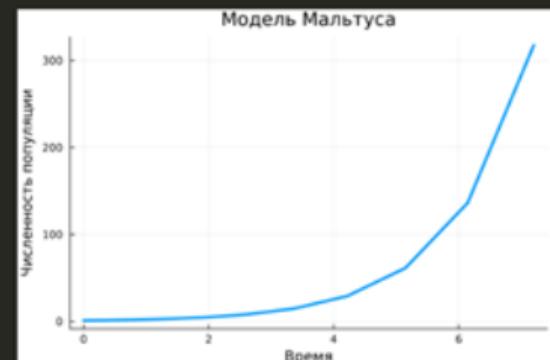


Самостоятельная работа

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):

```
* 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
# Задачи 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[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)
```

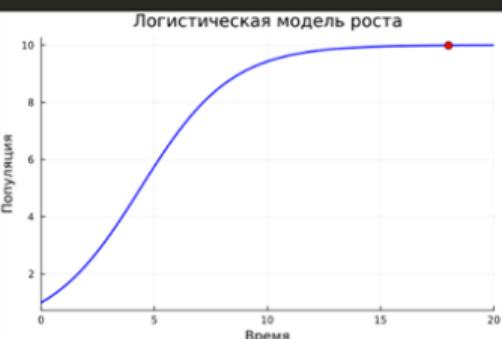


Самостоятельная работа

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

```
> % Определение 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 % Емкость экосистемы
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-линейной 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

```



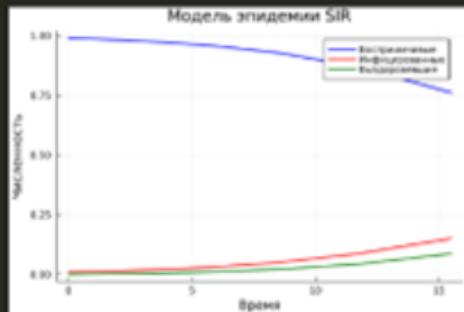
Самостоятельная работа

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

```
/* Основные настройки */
function sir(Nt, u0, p, t)
    P, S = p
    S, I, R = u0
    dS[1] = -P * S * I
    dI[1] = P * S * I
    dR[1] = P * S * I
    end

/* Начальные условия */
u0 = [0.99, 0.01, 0.0]
    * 99% здоровых, 1% инфицированных
/* Параметры */
P = 0.5    * Коэффициент передачи инфекции
Y = 0.1    * Коэффициент выживания
r = (P, Y)

/* Инициал. значение */
tstart = 0 (0, 0, 1000, 0)
/* Решение */
prob = ODEProblem(sir, u0, tstart, r)
sol = solve(prob, TsitC3)    * Используем TsitC3 для универсального решения
/* График решения */
while true for t in 1:length(sol.t)
    plot(sol.t[1:t], [u1] for u in sol.u[t]), label="Здоровые", color=:blue, linewidth=2
    plot(sol.t[1:t], [u2] for u in sol.u[t]), label="Инфицированные", color=:red, linewidth=2
    plot(sol.t[1:t], [u3] for u in sol.u[t]), label="Умершие", color=:green, linewidth=2
    title("Модель эпидемии SIR")
    xlabel("Время")
    ylabel("Численность")
end
/* Сохранение решения */
gtf(sol, "sir_model_animation.gif", frame=100)
```

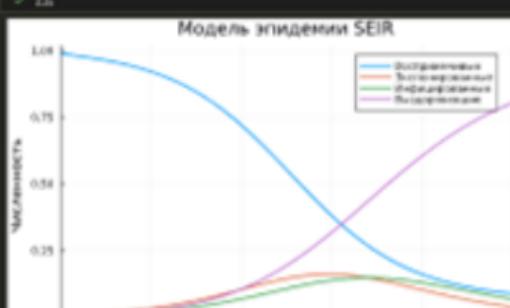


Самостоятельная работа

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

```
# (расширение модели)
function model(t)
    S, E, I, R = p
    S, E, I, R = S, E, I, R
    dS(t) = -beta * S * I / N          # Выздоровление
    dE(t) = beta * S * I / N - delta * E # Инфицирование
    dI(t) = delta * E - gamma * I      # Реконвалесценция
    dR(t) = gamma * I                 # Выздоровление
end

# Параметры
phi = [0.99, 0.0, 0.0, 0.0] # 100% выживаемость, 0% инфицирования
# Текущие
beta = 0.5 # Коэффициент передачи инфекции
delta = 0.1 # Вероятность исчезновения зараженного
gamma = 0.1 # Коэффициент выздоровления
N = 1.0 # Общая численность населения
p = (S, E, I, R)
# Инициализация
tspan = (0, 1000)
# График
prob = DifferentialEquation(model, phi, tspan, p)
sol = solve(prob, Tsit0) # Используем Tsit0 для начального задания
# Визуализация
plot(sol, labels=["Неспримечательные", "Заспороженные", "Инфицированные", "Выздоровевшие"],
    title="Модель эпидемии SEIR", xlabel="Время", ylabel="Численность", linewidth=2)
```

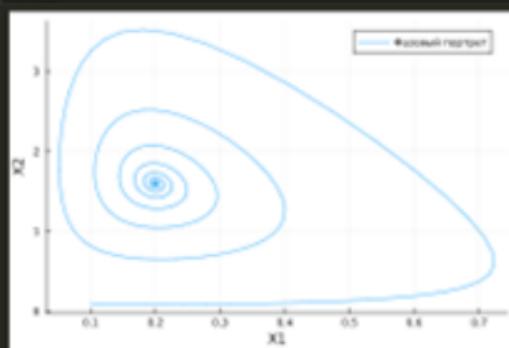


Самостоятельная работа

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

```
/* Определяем модель */
a = 2;
c = 1;
d = 0;
/* Написание метода для дискретной модели */
function lotka_volterra(x, p)
    a, c, d = p;
    x1, x2 = x;
    dx1 = a * x1 * (1 - x1) - x1 * x2;
    dx2 = -c * x2 + d * x1 * x2;
    return [dx1, dx2];
end;
/* Начальные условия */
X0 = [0.5, 0.1]; /* начальное значение x1 и x2 */
/* Шаги по времени */
T = 100;
dt = 0.1;
times = 0 : dt : T;
/* Высчитываем значения */
X = zeros(length(times), 2);
X[1, :] = X0;
/* Написание цикла */
for i in 1:length(times)
    X[i, :] = X[i-1, :] + dt * lotka_volterra(X[i-1, :], [a, c, d]);
end;
/* Выводим полученные данные */
plot(X[:, 1], X[:, 2], xlabel="Фазовый портрет", ylabel="X1", xlabel="X2");

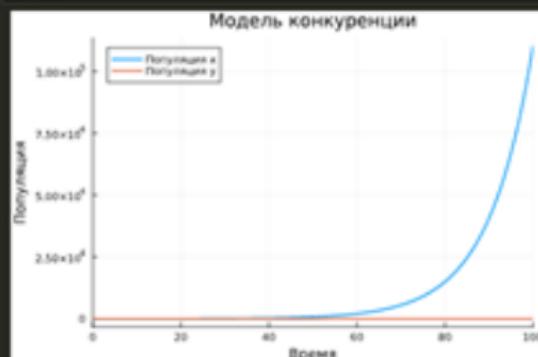
```



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)
    β = p
    K, u₀ = u
    du[1] = K * x - β * x * y
    du[2] = K * y - β * x * y
end
# Начальные условия
u₀ = [10.0, 5.0] # Население популяции
# Параметры
K = 0.1
β = 0.01
p = (K, β)
# Интервал времени
tspan = (0.0, 100.0)
# Симуляция
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 плюснить. Построить соответствующие графики (в том числе с анимацией) и фазовый портрет:

```
# Модель гармонического осциллятора
function harmonic_oscillator!(du, u, p, t)
    ω₀ = p[1] # циклическая частота
    du[1] = u[2] = 0 # y
    du[2] = -ω₀² * u[1] * u[2] = -ω₀² * x
end

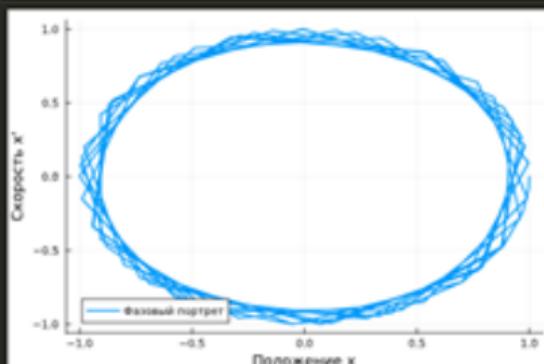
# Начальные условия
x₀ = 1.0 # Начальное положение (x)
y₀ = 0.0 # Начальная скорость (y/c)
u₀ = [x₀, y₀] # Начальные значения: x₀ и y₀

# Параметры
ω₀ = 1.0 # Циклическая частота (рад/с)
p = [ω₀] # Параметры системы

# Интервал времени
tspan = (0.0, 50.0)

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

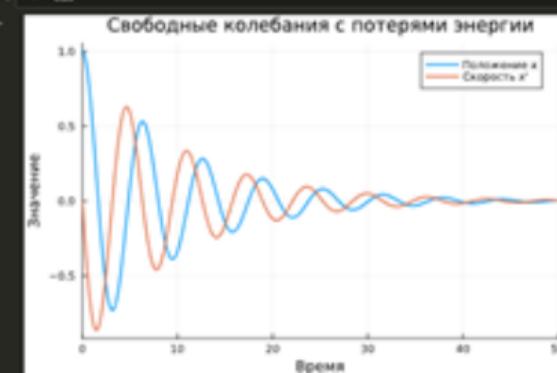
# Решение задачи
sol = solve(prob, Tsit5())
# Построение графиков (положение и скорость от времени)
plot(sol, label=["Положение x" "Скорость x"], title="Гармонический осциллятор", xlabel="Время (t)", ylabel="Значение", linewidth=2)
# Построение фазового портрета (x, x')
plot(sol[:, 1], sol[:, 2], label="Фазовый портрет", xlabel="Положение x", ylabel="Скорость x", linewidth=2)
```



Самостоятельная работа

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:

```
# Модель свободных колебаний с потерями
function damped_oscillator!(du, u, p, t)
    # w0 = p
    du[1] = u[2]
    du[2] = -2 * p * u[2] - u[2]^2 * u[1]
end
# Начальные условия
u0 = [1.0, 0.0] # Начальное положение + скорость
# Параметры
p = 0.1
w0 = 1.0
r = (p, w0)
# Интегрирование
tspan = (0.0, 50.0)
# Решение
prob = ODEProblem(damped_oscillator!, u0, tspan, r)
# 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)
```



Вывод

Вывод

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

Список литературы. Библиография

Список литературы. Библиография

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