

Лабораторная работа №6

Решение моделей в непрерывном и дискретном времени

Кузнецова С.В.

Российский университет дружбы народов, Москва, Россия

Информация

Докладчик

- Кузнецова София Вадимовна
- Российский университет дружбы народов

Цель

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. Решение обыкновенных дифференциальных уравнений

1.1. Модель экспоненциального роста

```
[2]: # 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),lin=:dash,label="Аналитическое решение")
```

[2]:

Модель экспоненциального роста

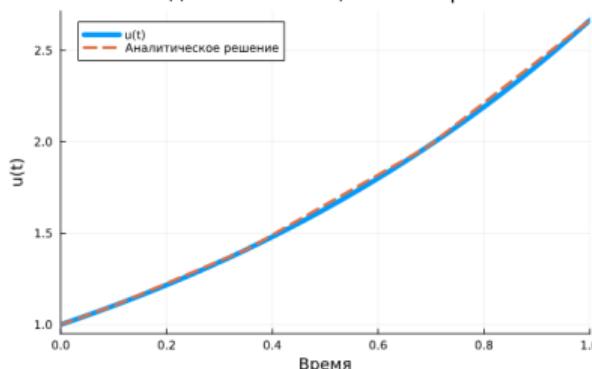


Рис. 1: График модели экспоненциального роста

Модель экспоненциального роста

```
[4]: # 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")
```

[4]: Модель экспоненциального роста

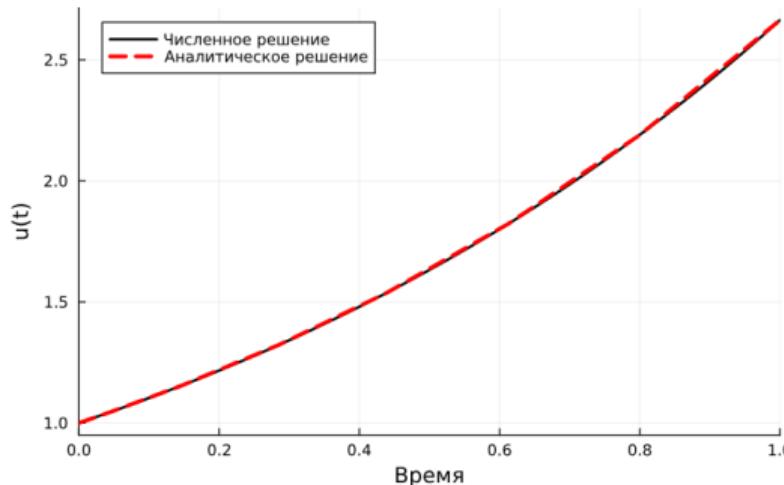


Рис. 2: График модели экспоненциального роста (задана точность решения)

Система Лоренца

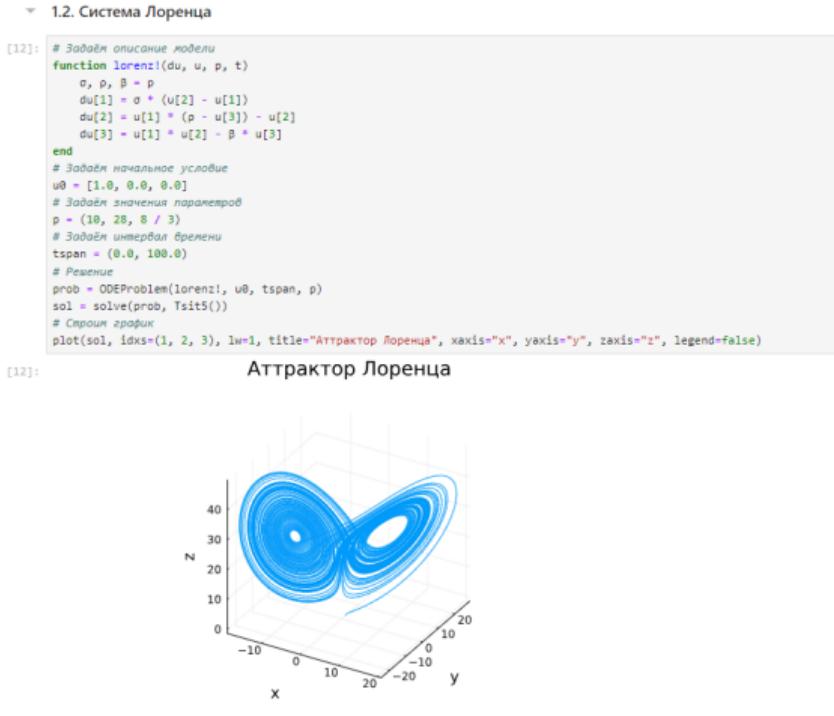


Рис. 3: Аттрактор Лоренца

Система Лоренца

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

[14]:

Аттрактор Лоренца

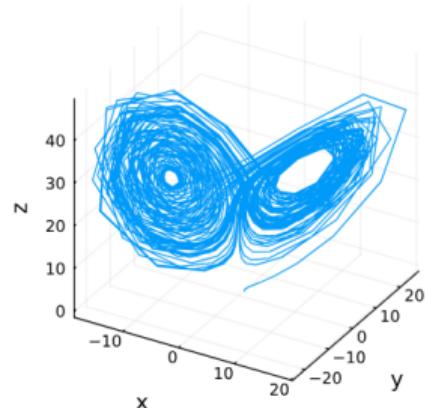


Рис. 4: Аттрактор Лоренца (интерполяция отключена)

Модель Лотки–Вольтерры

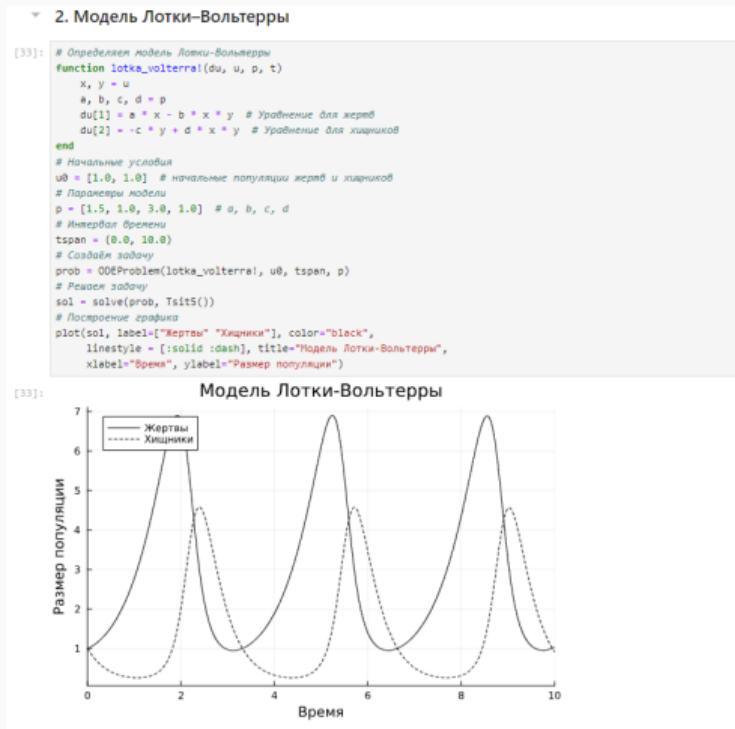


Рис. 5: Модель Лотки–Вольтерры: динамика изменения численности популяций

Модель Лотки–Вольтерры

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

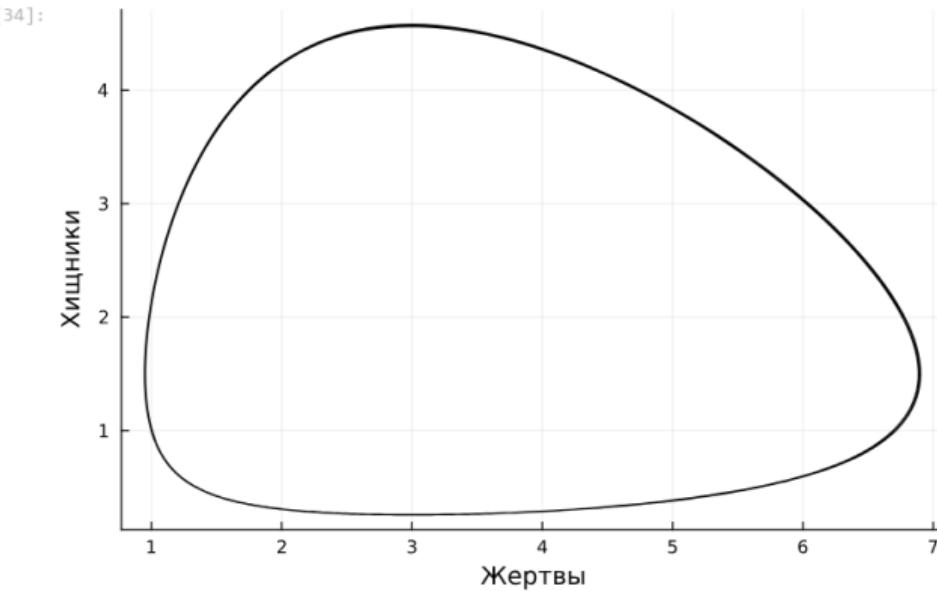


Рис. 6: Модель Лотки–Вольтерры: фазовый портрет

Самостоятельная работа

- ▼ 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):

```
[39]: # 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
# Задача 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)
```

```
[ Info: Saved animation to C:\Users\Ivan\Favorites\malthus_population_growth.gif
```

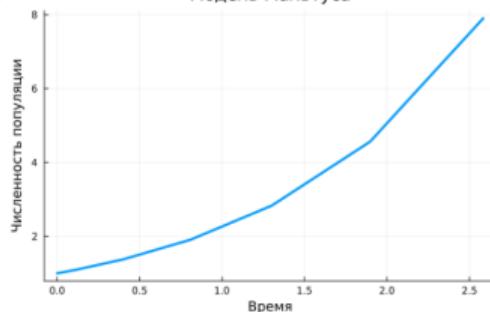


Рис. 7: Решение задания №1

Самостоятельная работа

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

```
[46]: # 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-линейской 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
```

[46]: Saved animation to C:\Users\Ivan\Favorites\logistic_growth.gif

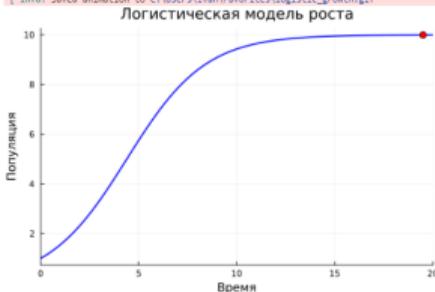


Рис. 8: Решение задания №2

Самостоятельная работа

в) 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):

```
[51]: # Определение модели
function sir!(du, u, p, t)
    i, l, r = u
    du[1] = -p * i * l           # Вспышка
    du[2] = p * i * l - v * l   # Инфицирование
    du[3] = v * l               # Выздоровление
end
# Начальные условия
ub = [0.99, 0.01, 0.0] # 99% восприимчивых, 1% инфицированных
# Параметры
p = 0.3 # Коэффициент передачи инфекции
v = 0.1 # Коэффициент выздоровления
r = (l, v)
# Инициализация браузера
tspan = (0.0, 100.0)
# Решение
prob = ODEProblem(sir!, ub, tspan, p)
sol = solve(prob, Tsit5()) # Используя Tsit5 для универсального решения
# Создание анимации
anim = Animation(tspan, sol)
for t in 1:tspan[2]:1:length(sol)
    plot(sol[1:t], [u[1] for u in sol[1:t]], label="Восприимчивые", color=:blue, linewidth=2)
    plot(sol[1:t], [u[2] for u in sol[1:t]], label="Инфицированные", color=:red, linewidth=2)
    plot(sol[1:t], [u[3] for u in sol[1:t]], label="Выздоровевшие", color=:green, linewidth=2)
    title!("Модель эпидемии SIR")
    xlabel!("Время")
    ylabel!("Численность")
end
# Создание анимации
gif(anim, "sir_model_animation.gif", fps=120)
```

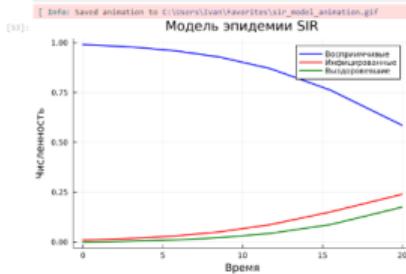


Рис. 9: Решение задания №3

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:

```
[62]: # Определение модели
function seir!(du, u, p, t)
    δ, β, γ, I₀ = p
    S, E, I, R = u
    du[1] = -β * S * I / N      # восприимчивые
    du[2] = β * S * I / N + δ * E # экспонируемые
    du[3] = δ * E - γ * I      # инфицированные
    du[4] = γ * I               # выздоровевшие
end
# начальные условия
u0 = [0.99, 0.0, 0.01, 0.0] # 99% восприимчивых, 1% инфицированных
# параметры
β = 0.3 # коэффициент передачи инфекции
δ = 0.1 # параметр инкубационного периода
γ = 0.1 # коэффициент выздоровления
N = 1.0 # общая численность населения
p = (β, δ, γ, N)
# интервал времени
tspan = (0.0, 100.0)
# Решение
prob = ODEProblem(seir!, u0, tspan, p)
sol = solve(prob, Tsit5()) # Используем Tsit5 для не-линейных задач
# построение графика
plot(sol, label:=[["Восприимчивые", "Экспонируемые", "Инфицированные", "Выздоровевшие"], title:"Модель эпидемии SEIR", xlabel:"время", ylabel:"численность", linewidth=2)
```

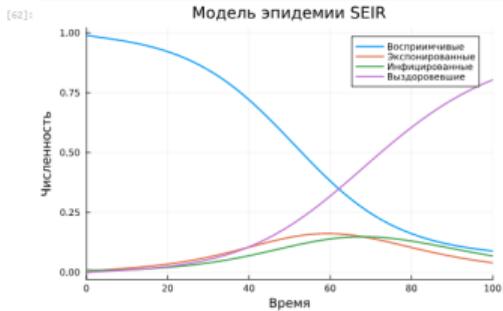


Рис. 10: Решение задания №4

Самостоятельная работа

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

```
[67]: # Параметры модели
a = 2
c = 1
d = 5
# Численные методы 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 = [0.1, 0.1] # начальные значения X1 и X2
# Время и шаг
T = 100
dt = 0.1
times = 0:dt:T
# Массив для численного решения
X = zeros(length(times), 2)
X[1, :] = X0
# Численное решение
for i in 2:length(times)
    X[i, :] = X[i-1, :] + dt * lotka_volterra!(X[i-1, :], (a, c, d))
end
# Построение фазового портрета
plot(X[:, 1], X[:, 2], xlabel="x1", ylabel="x2")
```

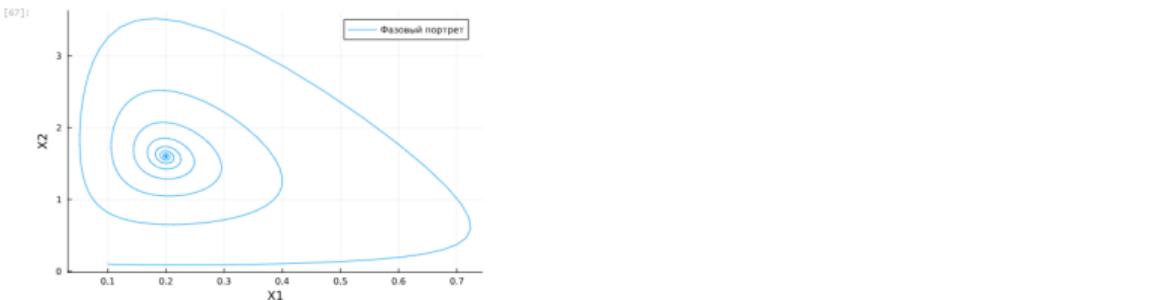


Рис. 11: Решение задания №5

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. 1

```
[72]: # 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 = 0.05
β = 0.02
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)
```

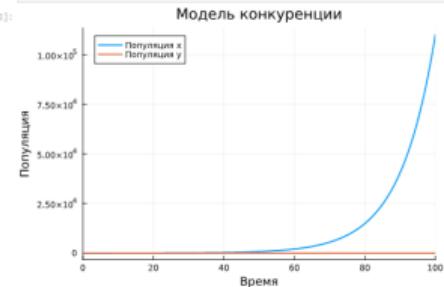


Рис. 12: Решение задания №6

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:

```
[54]: # модель гармонического осциллятора
function harmonic_oscillator!(du, u, p, t)
    u̇ = p[1] # циклическая частота
    du[1] = u[2] # x' = y
    du[2] = -ω² * u[1] # y' = -ω² * x
end
# начальные условия
x₀ = 1.0 # Начальное положение (x)
y₀ = 0.0 # начальная скорость (x/c)
u₀ = [x₀, y₀] # Начальные значения: x₀ и y₀
# Параметры
ω = 1.0 # циклическая частота (рад/с)
p = [ω] # 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)
```

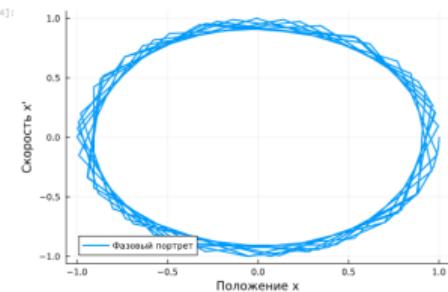


Рис. 13: Решение задания №7

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:

```
[87]: # Модель свободных колебаний с потерями
function damped_oscillator!(du, u, p, t)
    y, w = p
    du[1] = u[2]
    du[2] = -2 * y * u[2] - w^2 * u[1]
end
# начальные условия
w0 = [1.0, 0.0] # Начальное положение и скорость
# гармоники
y = 0.1
w0 = 1.0
p = (y, w0)
# интервал времени
tspan = (0.0, 50.0)
# решение
prob = ODEProblem(damped_oscillator!, w0, tspan, p)
# Используем более подобный алгоритм, например, Tsit5
sol = solve(prob, Tsit5())
# визуализация
plot(sol, label=["Положение x" "Скорость x"], title="Свободные колебания с потерями энергии", xlabel="Время", ylabel="Значение", linewidth=2)
```

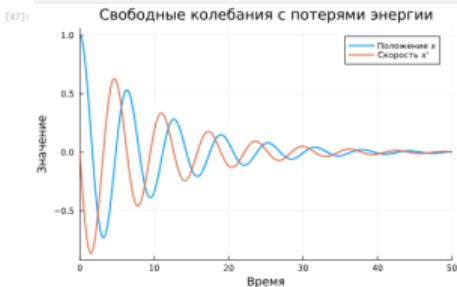


Рис. 14: Решение задания №8

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/>

Спасибо за внимание!