Отчёт по лабораторной работе №6
Компьютерный практикум по статистическому анализу данных

Решение моделей в непрерывном и дискретном времени

Выполнил: Махорин Иван Сергеевич,
НПИбд-02-21, 1032211221

Содержание

1 Цель работы **4**

2 Выполнение лабораторной работы **5**
2.1 Решение обыкновенных дифференциальных уравнений 5
2.2 Модель экспоненциального роста 5
2.3 Система Лоренца 7
2.4 Модель Лотки–Вольтерры 9
2.5 Самостоятельное выполнение 11

3 Вывод **16**

4 Список литературы. Библиография **17**

Список иллюстраций

2.1 График модели экспоненциального роста 6
2.2 График модели экспоненциального роста (задана точность решения) 7
2.3 Аттрактор Лоренца 8
2.4 Аттрактор Лоренца (интерполяция отключена) 9
2.5 Модель Лотки–Вольтерры: динамика изменения численности популяций 10
2.6 Модель Лотки–Вольтерры: фазовый портрет 11
2.7 Решение задания №1 12
2.8 Решение задания №2 12
2.9 Решение задания №3 13
2.10 Решение задания №4 13
2.11 Решение задания №5 14
2.12 Решение задания №6 14
2.13 Решение задания №7 15
2.14 Решение задания №8 15

1 Цель работы

Основной целью работы является освоение специализированных пакетов для решения задач в непрерывном и дискретном времени.

2 Выполнение лабораторной работы

2.1 Решение обыкновенных дифференциальных уравнений

Вспомним, что обыкновенное дифференциальное уравнение (ОДУ) описывает изменение некоторой переменной u.

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

2.2 Модель экспоненциального роста

Рассмотрим пример использования этого пакета для решение уравнения модели экспоненциального роста, описываемую уравнением, где a — коэффициент роста.

Численное решение в Julia будет иметь следующий вид, а также график, соответствующий полученному решению (рис. 2.1):

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),lw=3,ls=:dash,label="Аналитическое решение")
```

[2]:

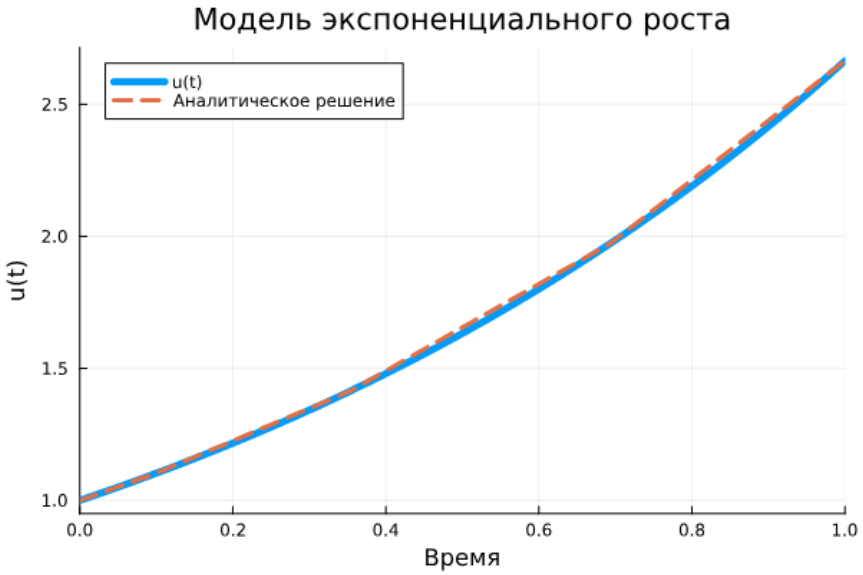

Рис. 2.1: График модели экспоненциального роста

При построении одного из графиков использовался вызов sol.t, чтобы захватить массив моментов времени. Массив решений можно получить, воспользовавшись sol.u.

Если требуется задать точность решения, то можно воспользоваться параметрами abstol (задаёт близость к нулю) и reltol (задаёт относительную точность). По умолчанию эти параметры имеют значение abstol = 1e-6 и reltol = 1e-3.

Для модели экспоненциального роста (рис. 2.2):

```
[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")
```

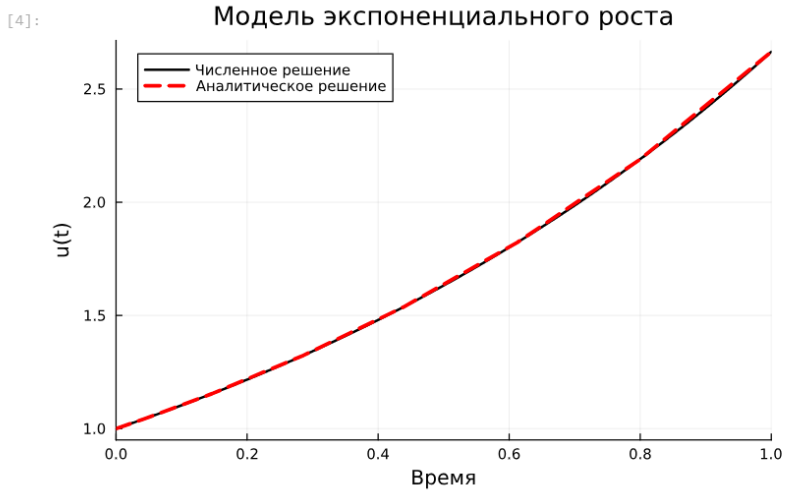

Рис. 2.2: График модели экспоненциального роста (задана точность решения)

2.3 Система Лоренца

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

Система получена из системы уравнений Навье–Стокса и описывает движение воздушных потоков в плоском слое жидкости постоянной толщины при разложении скорости течения и температуры в двойные ряды Фурье с последующем усечением до первых-вторых гармоник.

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

Численное решение в Julia будет иметь следующий вид (рис. 2.3):

1.2. Система Лоренца

[12]:
```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)
```

[12]:

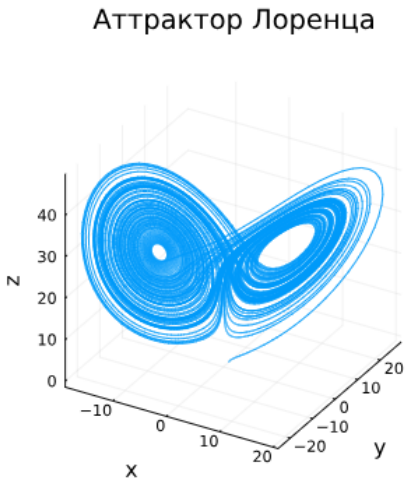

Рис. 2.3: Аттрактор Лоренца

Можно отключить интерполяцию (рис. 2.4):

```
[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]:

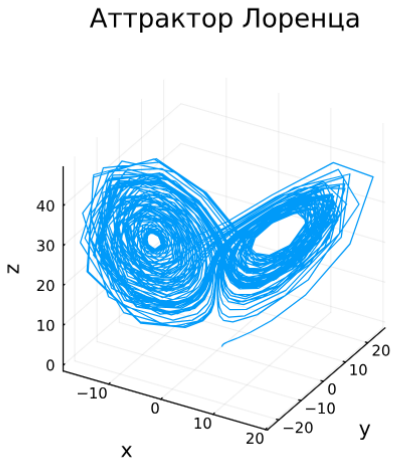

Рис. 2.4: Аттрактор Лоренца (интерполяция отключена)

2.4 Модель Лотки–Вольтерры

Модель Лотки–Вольтерры описывает взаимодействие двух видов типа «хищник – жертва».

Численное решение в Julia будет иметь следующий вид (рис. 2.5):

2. Модель Лотки–Вольтерры

[33]:
```
# 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)
# Создаём 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")
```

[33]:

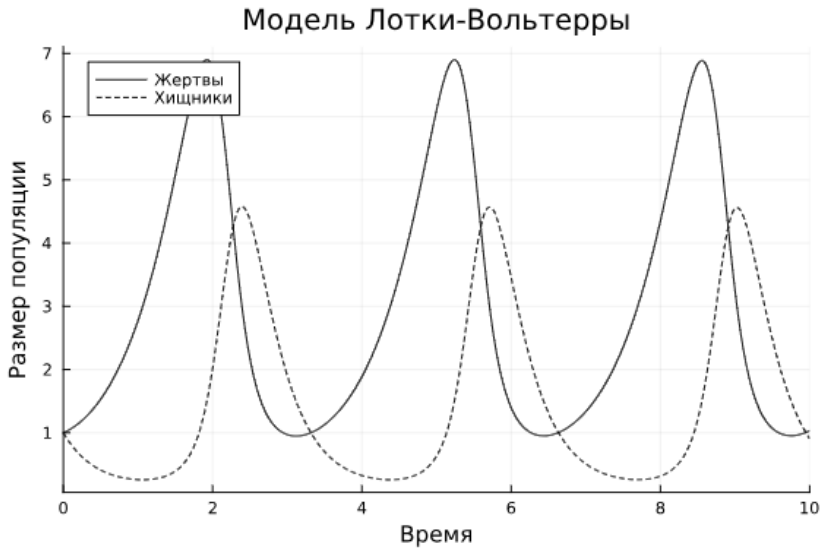

Рис. 2.5: Модель Лотки–Вольтерры: динамика изменения численности популяций

Фазовый портрет (рис. 2.6):

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

[34]:

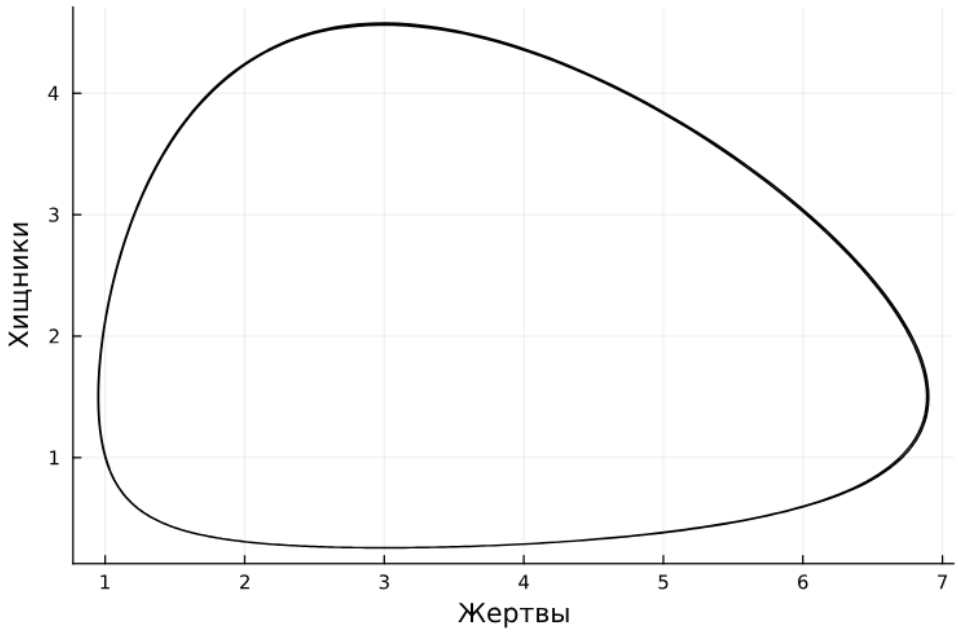

Рис. 2.6: Модель Лотки–Вольтерры: фазовый портрет

2.5 Самостоятельное выполнение

Выполнение задания №1 (рис. 2.7):

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
# 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)
```

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

[39]:

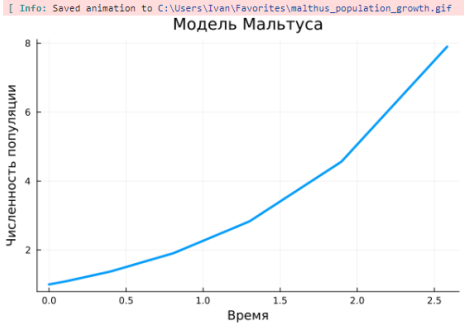

Рис. 2.7: Решение задания №1

Выполнение задания №2 (рис. 2.8):

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  # Ёмкость 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
```

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

[46]:

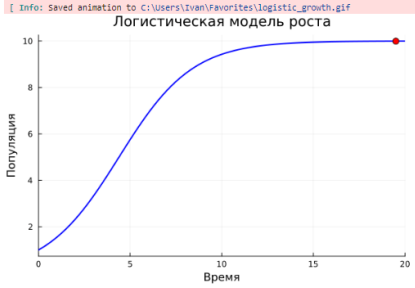

Рис. 2.8: Решение задания №2

Выполнение задания №3 (рис. 2.9):

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

```
[53]: # Определение модели
function sir!(du, u, p, t)
    β, γ = p
    s, i, r = u
    du[1] = -β * s * i          # Восприимчивые
    du[2] = β * s * i - γ * i   # Инфицированные
    du[3] = γ * i               # Выздоровевшие
end
# Начальные условия
u0 = [0.99, 0.01, 0.0]  # 99% восприимчивых, 1% инфицированных
# Параметры
β = 0.3  # Коэффициент передачи инфекции
γ = 0.1  # Коэффициент выздоровления
p = (β, γ)
# Интервал времени
tspan = (0.0, 100.0)
# Решение
prob = ODEProblem(sir!, u0, tspan, p)
sol = solve(prob, Tsit5())  # Используем Tsit5 для универсального решения
# Создание анимации
anim = @animate for t in 1:length(sol.t)
    plot(sol.t[1:t], [u[1] for u in sol.u[1:t]], label="Восприимчивые", color=:blue, linewidth=2)
    plot!(sol.t[1:t], [u[2] for u in sol.u[1:t]], label="Инфицированные", color=:red, linewidth=2)
    plot!(sol.t[1:t], [u[3] for u in sol.u[1:t]], label="Выздоровевшие", color=:green, linewidth=2)
    title!("Модель эпидемии SIR")
    xlabel!("Время")
    ylabel!("Численность")
end
# Сохранение анимации
gif(anim, "sir_model_animation.gif", fps=120)
```

```
[ Info: Saved animation to C:\Users\ivan\Favorites\sir_model_animation.gif
```

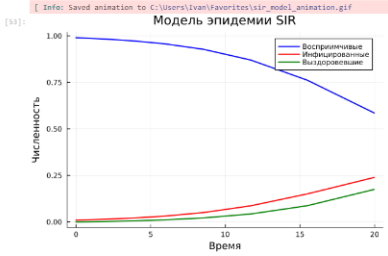

Рис. 2.9: Решение задания №3

Выполнение задания №4 (рис. 2.10):

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)
    β, δ, γ, N = p
    s, e, i, r = u
    du[1] = -β * s * i / N        # Восприимчивые
    du[2] = β * s * i / N - δ * e # Экспонированные
    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-жёстких задач
# Построение графика
plot(sol, label=["Восприимчивые" "Экспонированные" "Инфицированные" "Выздоровевшие"],
     title="Модель эпидемии SEIR", xlabel="Время", ylabel="Численность", linewidth=2)
```

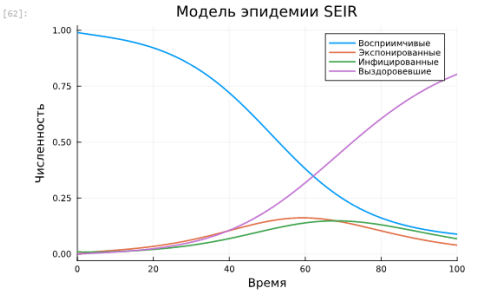

Рис. 2.10: Решение задания №4

Выполнение задания №5 (рис. 2.11):

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
# Массивы 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], label="Фазовый портрет", xlabel="X1", ylabel="X2")
```

[67]:

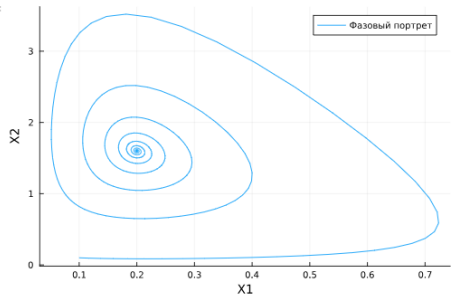

Рис. 2.11: Решение задания №5

Выполнение задания №6 (рис. 2.12):

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: ¶

[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 = 0.1
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)
```

[72]:

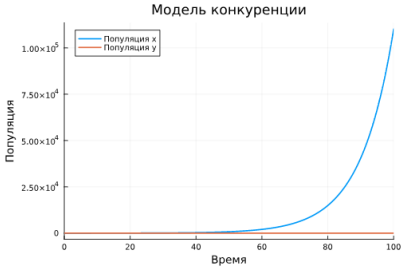

Рис. 2.12: Решение задания №6

Выполнение задания №7 (рис. 2.13):

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:

[84]:
```
# 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, 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)
```

[84]:

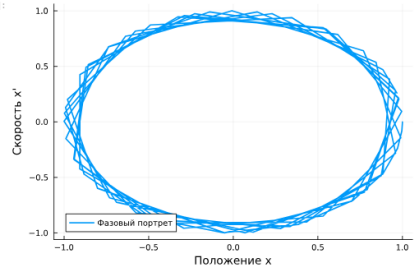

Рис. 2.13: Решение задания №7

Выполнение задания №8 (рис. 2.14):

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:

[87]:
```
# 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)
```

[87]:

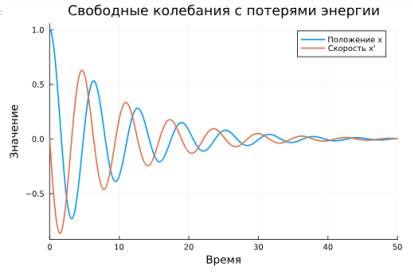

Рис. 2.14: Решение задания №8

3 Вывод

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

4 Список литературы. Библиография

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