

Лабораторная работа 6

Решение моделей в непрерывном и дискретном времени

Ланцова Я. И.

Российский университет дружбы народов, Москва, Россия

Информация

- Ланцова Яна Игоревна
- студентка
- Российский университет дружбы народов

Цель работы

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

Задание

1. Используя JupyterLab, повторить примеры. При этом дополнить графики обозначениями осей координат, легендой с названиями траекторий, названиями графиков и т.п.
2. Выполнить задания для самостоятельной работы.

Выполнение лабораторной работы

Выполнение лабораторной работы

```
Модель экспоненциального роста

[=]: # подключаем необходимые пакеты
import Pkg
Pkg.add("DifferentialEquations")
using DifferentialEquations

Updating registry at '~/.julia/registries/General.toml'
Resolving package versions...
Installed StatsFuns ━━━━━━━━ v1.5.2
Installed OrdinaryDiffEqRK45 ━━━━━━ v1.5.0
Installed OrdinaryDiffEqRosenbrock ━━━━━━ v1.10.1
Installed SciMLBase ━━━━━━ v1.0.8
Installed DifferentialEquationFunctions ━━━━ v0.1.20
Installed DifferentialEquations ━━━━ v7.37.0
Installed OrdinaryDiffEqStabilizedRK ━━━━ v1.4.0
Installed NonlinearSolve ━━━━ v4.12.0
Installed SciMLLogging ━━━━ v1.5.0
Installed Accessors ━━━━ v0.1.42
Installed AlmostBlockDiagonals ━━━━ v0.1.10
Installed BoundaryValueDiffEqFIRK ━━━━ v1.9.0
```

Рис. 1: Подключим основные пакеты

Выполнение лабораторной работы

```
[3]: # задаём описание модели с начальными условиями:  
# u(0,0) = 0.0  
# u(0,1) = 1.0  
# задаём интервал времени:  
tspan = [0,0,1,0]  
# решение:  
prob = ODEProblem(f,u0,tspan)  
sol = solve(prob)
```



```
[3]: retcode: Success  
Interpolation: 3rd order Hermite  
t: 5-element Vector{Float64}:  
0.0  
0.18842494449239292  
0.35218942027865868  
0.693443601221597829  
1.0  
u: 5-element Vector{Float64}:  
1.0  
1.434232847055465  
1.4121980713484919  
1.9738384457359198  
2.6644561424814266
```

Рис. 2: Модель экспоненциального роста

Выполнение лабораторной работы

```
[5]: # строка графика
plot(sol, linewidth=5,title="Модель экспоненциального роста", xaxis="Время",yaxis="u(t)",label="u(t)")
plot!(sol, t, t->exp(a*t),lw=5,lsm=dash,label="Аналитическое решение")
```

Модель экспоненциального роста

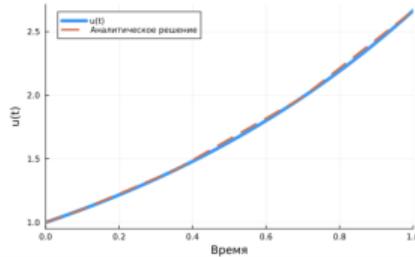


Рис. 3: График модели экспоненциального роста

Выполнение лабораторной работы

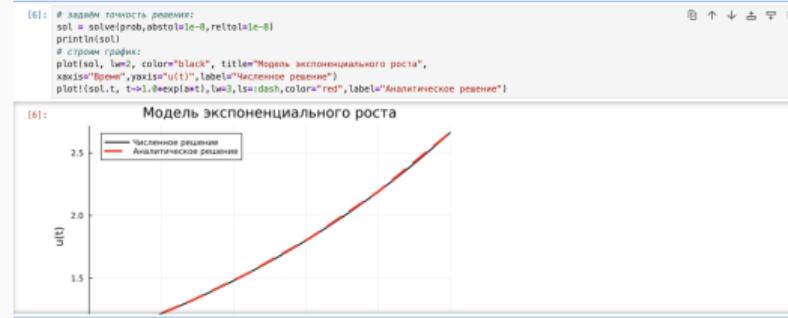


Рис. 4: График модели экспоненциального роста

Выполнение лабораторной работы

Система Лоренца

```
[8]: # задаем описание модели
function lorenz!(du,u,p,t)
    a,b,c = p
    du[1] = c*u[2]-u[3]
    du[2] = u[1]*c-p*u[3] - u[2]
    du[3] = u[1]*u[2] - b*u[1]
end

[8]: lorenz! (generic function with 1 method)

[9]: # зададим начальное условие:
u0 = [1.0,0.0,0.0]
# зададим значение параметров:
p = (10,28,8/3)
# зададим интервал времени:
tspan = (0.0,100.0)
# решим:
prob = ODEProblem(lorenz!,u0,tspan,p)
sol = solve(prob)

[9]: retcode: Success
Interpolation: 3rd order Hermite
t: 1292-element Vector{Float64}:
 0.0
-0.477099482393184e-5
 0.480392464653193254
 0.883262489731175873
 0.8995858760222868189
 0.81655647980176743
 0.82795598519429853
 0.8418562219428344
 0.8693498411385493296
 0.883685412104999294
 0.113364993361839311
 0.14862181393426532
```

Рис. 5: Система Лоренца

Выполнение лабораторной работы

```
[14]: [14.543594291970454, 0.633028572608300, 48.38844424656828]
[9.527702621885666, 0.887505733614436, 37.6471276883169]
[4.563755134814175, -2.4397213025654537, 31.159837751866518]
[1.2013491915335158, -0.429612308254818, 25.8593237347986]
[-4.15538509050905, -2.108807578180881, 28.48957895756388]
[-1.355432835252145, -2.573578051098236, 18.48952629832982]
[-2.161898772303465, -3.59578018801676297, 15.936724265473792]
[-3.4337834686573715, -5.786446127166832, 14.065127938866913]
[-5.971873646288483, -10.261846984477597, 14.466939896824572]
[-18.945388042598972, -17.31254205417734, 20.63949548858990]
[-14.714517151880372, -16.9683551893885, 33.06517212688602]
[-13.714517151880374, -4.323398384833889, 38.398231477265624]

[15]: # строка графики
plot(sol, vars=[1,2,3], lw=2, title="Аттрактор Лоренца", xaxis="x", yaxis="y", zaxis="z", legend=False)
[15]: Аттрактор Лоренца
```

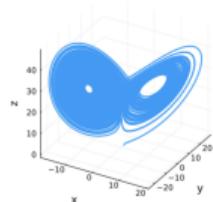


Рис. 6: График системы Лоренца

Выполнение лабораторной работы

Модель Лотки–Вольтерры

```
[2]: # подключаем необходимые пакеты:
import Pkg
Pkg.add!("ParameterizedFunctions")
using ParameterizedFunctions, DifferentialEquations, Plots

# Loading package versions...
Installed Unitful ━━━━━━ v1.25.1
Installed OffsetArrays ━━━━━━ v1.17.0
Installed Tricks ━━━━━━ v0.1.13
Installed FindFirstFunction ━━━━ v1.4.2
Installed Bijections ━━━━ v0.2.2
Installed DataAPI ━━━━ v0.7.16
Installed MutableArithmetics ━━━━ v1.1.7
Installed JuliaFormatter ━━━━ v2.2.8
Installed MultivariatePolynomials ━━ v0.5.13
Installed CompositeTypes ━━ v0.3.14
Installed Glob ━━ v1.3.1
Installed JuliaSyntax ━━ v0.4.18
Installed BiochemicalDoctor ━━ v0.0.26
Installed MSStyle ━━ v0.4.37
Installed Symbolics ━━ v6.37.0
Installed CommonMark ━━ v0.9.1
Installed DynamicPolynomials ━━ v0.6.4
Installed Combinatorics ━━ v1.6.2
Installed TaskLocalValues ━━ v0.1.3
Installed AbstractTrees ━━ v0.4.3
```

Рис. 7: Подключим основные пакеты

Выполнение лабораторной работы

```
[12]: # задание описание модели:
lvl = @ode_def LotkaVolterra begin
    dxdt = a*x - b*x*y
    dydt = -c*y + d*x*y
end a b c d

# задание начальное условие:
u0 = [1.0,1.0]

# задание значение параметров:
p = [1.5,1.0,3.0,1.0]

# задание интервал времени:
tspan = (0.0,10.0)

# Warning: Independent variable t should be defined with g independent_variables t.
# @ ModelingToolkit ~/julia/packages/ModelingToolkit/8TOKs/src/utils.jl:121
[12]: (0.0, 10.0)
```

Рис. 8: Модель Лотки–Вольтерры

Выполнение лабораторной работы

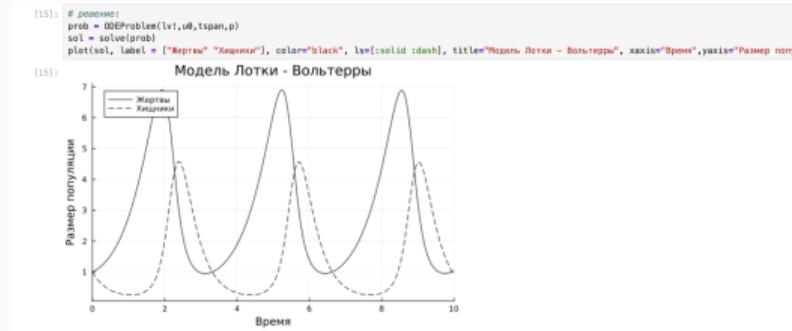


Рис. 9: График модели Лотки-Вольтерры

Выполнение лабораторной работы

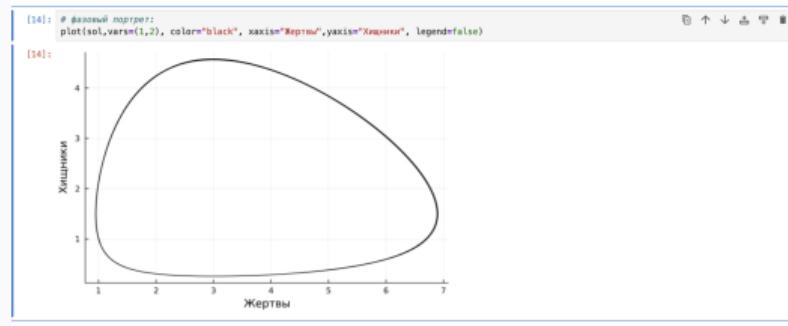


Рис. 10: Фазовый портрет Лотки–Вольтерры

Задание 1

Выполнение лабораторной работы

```
Задания для самостоятельного выполнения

Задание 1

Реализовать и проанализировать модель роста численности изолированной популяции (модель Мальтуса):  $x = ax$ ,  $a = b - c$ . Где  $x(t)$  — численность изолированной популяции в моменте времени  $t$ 

[16]: # Настройки графиков
grisize{300, 200}, legend:topleft}

[16]: Plots.GRBackend()

[17]: function malthus_model!(du, u, p, t)
    x = u[1]
    a = p[1]

    du[1] = a * x
end

[17]: malthus_model! (generic function with 1 method)

[18]: # Параметры для модели Мальтуса
a = 0.1 # коэффициент роста (рождаемость 0.15 – смертность 0.05)
x0_malthus = [10.0] # начальная численность популяции
tspan_malthus = (0.0, 50.0)

[18]: (0.0, 50.0)
```

Рис. 11: Задание №1

Выполнение лабораторной работы

```
[19]: # Решение ОДУ
prob_malthus = ODEProblem(malthus_model!, x0_malthus, tspan_malthus, [a])
sol_malthus = solve(prob_malthus, Tsit5())

[19]: retcode: Success
Interpolation: specialized 4th order "free" interpolation
t: 12-element Vector{Float64}:
 0.0
 0.0464748899371344
 1.502543026754578
 3.20215723104589
 7.81811804849577
12.8915574813694
17.123537618951876
22.8939253314886125
28.5920838815004
36.42928387952584
44.88208395722155
59.8

u: 12-element Vector{Vector{Float64}}:
 [18.0]
 [18.0]
 [15.59753144259899]
 [11.491235813782526]
 [15.417552627268108]
 [21.826895177972534]
 [33.596538742124775]
 [55.41985978867995]
 [98.68628116174498]
 [139.83116174498]
 [382.4384751808252]
 [821.1952074572912]
 [1484.0928812399486]
```

Рис. 12: Задание №1

Выполнение лабораторной работы

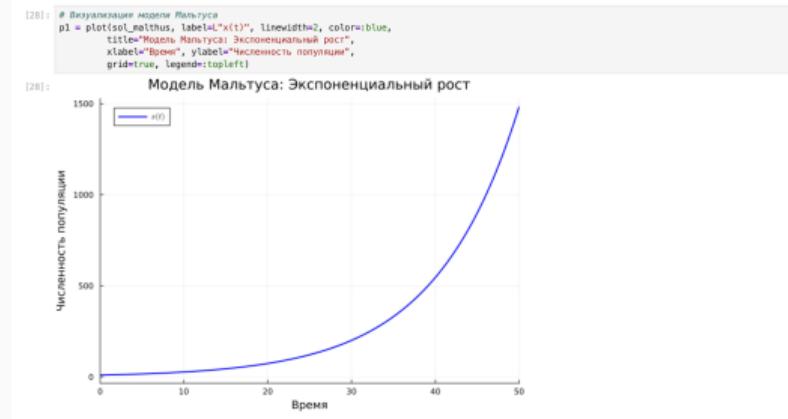


Рис. 13: График №1

Выполнение лабораторной работы

```
[29]: # Анимация для модели Мальтуса
anis_malthus = @animate for t in range(0, 50, length=200)
    plot(x0_malthus, label="x(t)", linewidth=2, color=blue,
        title="Модель Мальтуса: Экспоненциальный рост",
        xlabel="Время", ylabel="Численность популяции",
        xlims=(0, 50), ylims=(0, 1500))

    current_t = min(t, 50)
    current_x = x0_malthus[:] * exp(a * current_t)

    scatter!(current_t, [current_x], color=red, markersize=6,
            label="Текущее состояние")

    annotate!(40, 1200, text="t = $(round(current_t, digits=1))", 10)
    annotate!(40, 1000, text="x = $(round(Int, current_x))", 10)
end

gif(anis_malthus, "malthus_model.gif", fps=15)

[ Info: Saved animation to /Users/yana/malthus_model.gif
```

Рис. 14: Анимация №1

Выполнение лабораторной работы

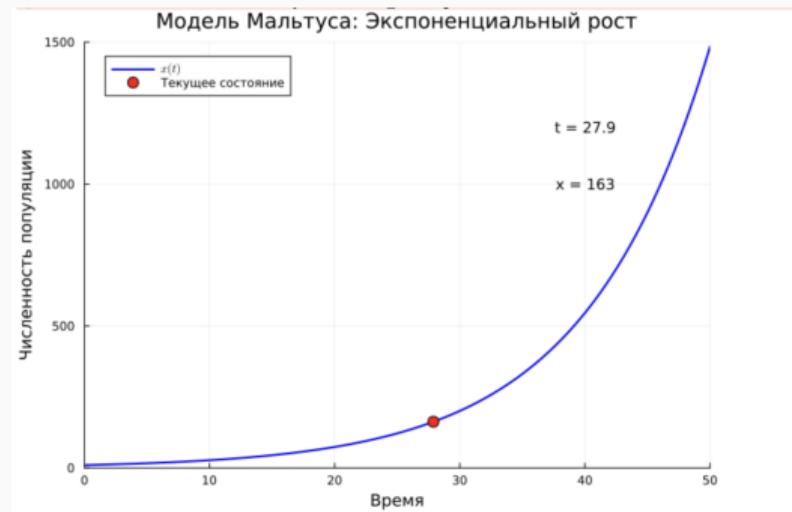


Рис. 15: Анимация №1

Задание 2

Выполнение лабораторной работы

Задание 2

Реализовать и проанализировать логистическую модель роста популяции, заданную уравнением, где r — коэффициент роста популяции, k — потенциальная ёмкость экологической системы (пределальное значение численности популяции). Начальные данные и параметры задать самостоятельно и пояснить их выбор

```
[38]: function logistic_model!(du, u, p, t)
    x = u[1]
    r, k = p

    du[1] = r * x * (1 - x/k)
end

[39]: logistic_model! (generic function with 1 method)

[31]: # Параметры для логистической модели
r = 0.2           # коэффициент роста
k = 1000.0         # ёмкость среды
x0_logistic = [10.0] # начальная численность
tspan_logistic = (0.0, 100.0)

prob_logistic = ODEProblem(logistic_model!, x0_logistic, tspan_logistic, [r, k])
sol_logistic = solve(prob_logistic, Tsit5());
```

```
[31]: retcode: Success
Interpolation: specialized 4th order "free" interpolation
ti: 21-element Vector{Float64}:
 0.0
 0.13825334480494908
 0.97748693998484323
 2.4987700000000003
 4.269828341498329
 6.5830514837796425
 9.128411459871941
12.169582447833334
15.62487825564484
```

Рис. 16: Задание №2

Выполнение лабораторной работы

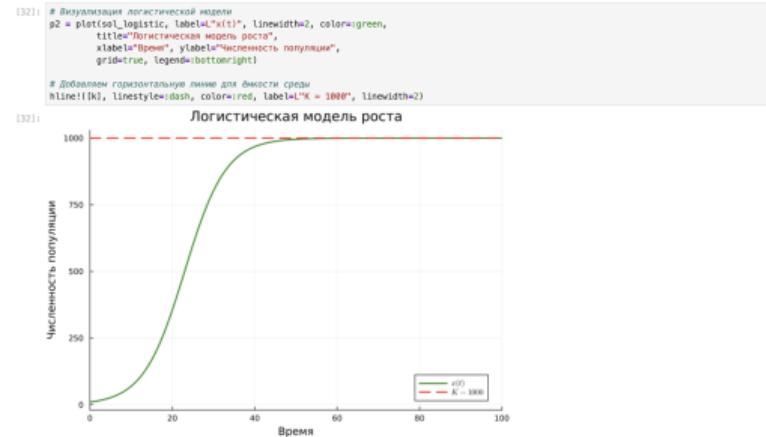


Рис. 17: График №2

Выполнение лабораторной работы

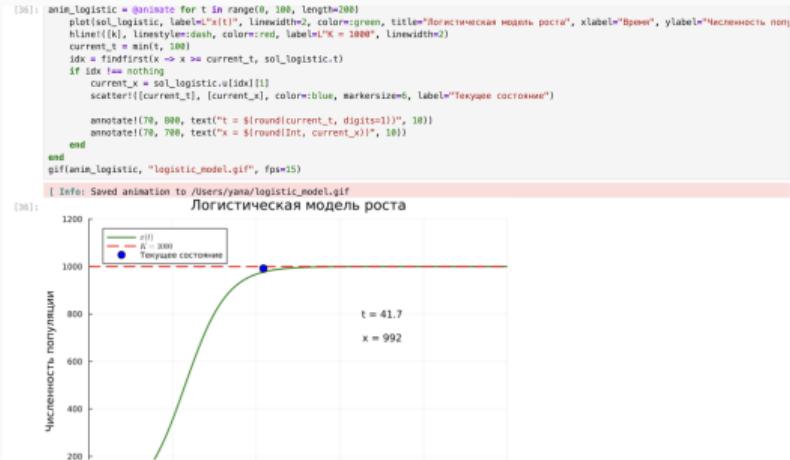


Рис. 18: Анимация №2

Задание 3

Выполнение лабораторной работы

Задание 3

Реализовать и проанализировать модель эпидемии Кермака–Маккендрика (SIRмодель), где $s(t)$ — численность восприимчивых к болезни индивидов в момент времени t , $i(t)$ — численность инфицированных индивидов в момент времени t , $r(t)$ — численность переболевших индивидов в момент времени t , β — коэффициент интенсивности контактов индивидов с последующим инфицированием, γ — коэффициент интенсивности выздоровления инфицированных индивидов. Численность популяции считается постоянной. Начальные данные и параметры задать самостоятельно и пояснить их выбор.

```
[37]: function sir_model!(du, u, p, t)
    s, i, r = u
    β, γ = p
    du[1] = -β * i * s      # ds/dt
    du[2] = β * i * s - γ * i # di/dt
    du[3] = γ * i            # dr/dt
end

[37]: sir_model! (generic function with 1 method)

[38]: # Параметры для SIR модели
       β = 0.3      # коэффициент заражения
       γ = 0.1      # коэффициент выздоровления
       N = 1000     # общая численность популяции

# Начальные условия: 1 зараженный, остальные восприимчивы
u0_sir = [s0, 1, 0]
tspan_sir = [0.0, 200.0]

prob_sir = ODEProblem(sir_model!, u0_sir, tspan_sir, [β, γ])
sol_sir = solve(prob_sir, Tsit5())
```

[38]: retcode: Success
Interpolation: specialized 4th order "free" interpolation

Рис. 19: Задание №3

Выполнение лабораторной работы

```
[44]: # Визуализация SIR модели
B3 = plot(sol_sir, labels=[r's(t)', r'i(t)', r'r(t)'], linewidth=2,
          palette=[purple, red, green],
          title="Модель эпидемии SIR",
          xlabel="Время", ylabel="Доля зараженных",
          grid=True, legend=True)
```

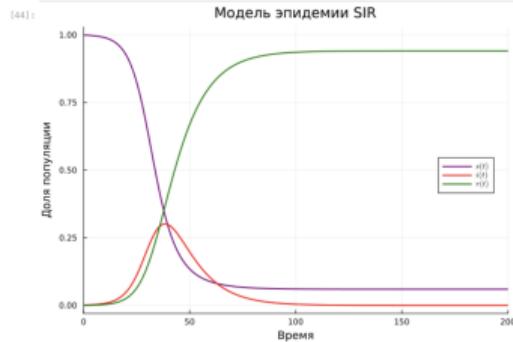


Рис. 20: График №3

Выполнение лабораторной работы

```
[46]: # Анимация для SIR модели
anim_sir = animate for t in range(0, 200, length=300)
plot(sol_sir, xlabel="t", ylabel="I(t)", linewidth=2, palette=[:blue, :red, :green], title="Модель эпидемии SIR",
      xlims=(0, 200), ylims=(0, 1),
      grid=True, legend=True)

current_t = mint, 200
idx = findfirst(x -> x >= current_t, sol_sir.t)

if idx == nothing
    current_S = sol_sir.u[idx][1]
    current_I = sol_sir.u[idx][2]
    current_R = sol_sir.u[idx][3]

    scatter!(current_t, current_I, [current_S, current_I, current_R], color=[:blue, :red, :green], markersize=6, label="")
    annotate!((150, 0.8, text("S = $(round(current_S, digits=3))"), 10, :blue))
    annotate!((150, 0.7, text("I = $(round(current_I, digits=3))"), 10, :red))
    annotate!((150, 0.6, text("R = $(round(current_R, digits=3))"), 10, :green))
end
end
```

Рис. 21: Анимация №3

Выполнение лабораторной работы

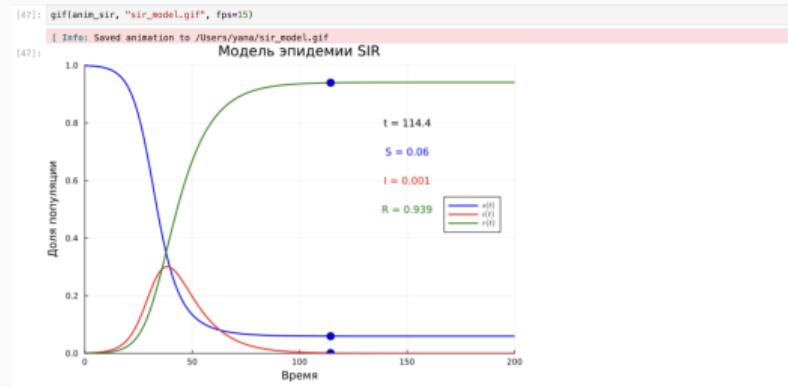


Рис. 22: Анимация №3

Задание 4

Выполнение лабораторной работы

Задание 4

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

```
[48]: function seir_model!(du, u, p, t)
    s, e, i, r = u
    β, γ, ρ = p

    du[1] = -β/N * s * i           # ds/dt
    du[2] = β/N * s * i - β * e   # de/dt
    du[3] = β * e - γ * i          # di/dt
    du[4] = γ * i                  # dr/dt
end

seir_model! (generic function with 1 method)

[49]: # Параметры для SEIR модели
β_seir = 0.4 # коэффициент заражения
δ_seir = 0.2 # скорость перехода из экспоненциальных в инфицированные (2/инкубационный период)
γ_seir = 0.1 # скорость выздоровления
N_seir = 1000 # общая численность популяции

# Начальные условия: I - экспоненциальный, остальные восприимчивые
s0_seir = (N_seir - 1) / N_seir
e0_seir = 0.0 / N_seir
i0_seir = 0.0
r0_seir = 0.0
r0_seir = [s0_seir, e0_seir, i0_seir, r0_seir]
tspan_seir = [0.0, 200.0]

prob_seir = ODEProblem(seir_model!, s0_seir, tspan_seir, [β_seir, δ_seir, γ_seir, N_seir])
sol_seir = solve(prob_seir, Tsit5());
```

Рис. 23: Задание №4

Выполнение лабораторной работы

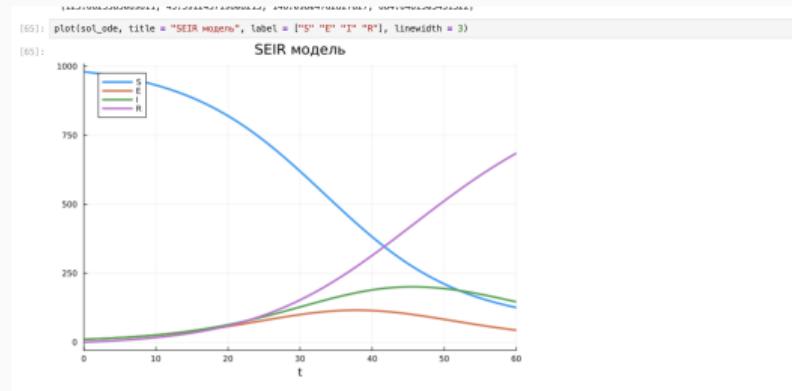


Рис. 24: График №4

Задание 5

Выполнение лабораторной работы

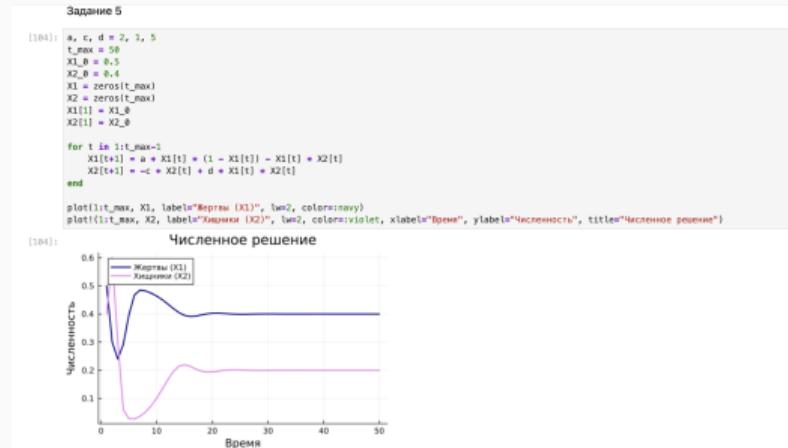


Рис. 25: Задание №5

Выполнение лабораторной работы

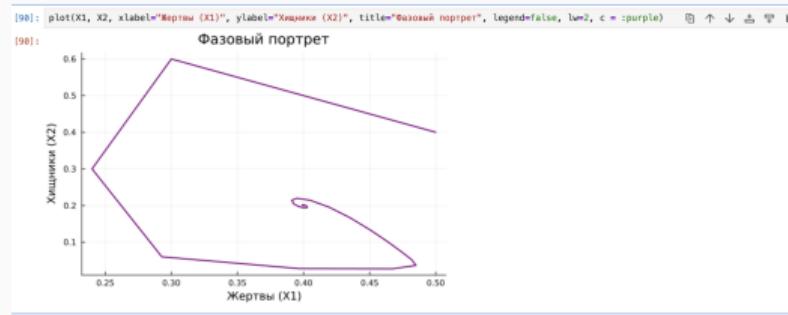


Рис. 26: График №5

Задание 6

Выполнение лабораторной работы

```
Задание 6

[128]: function competition(u,p,t)
    x, y = u
    dx = p[0]
    dy = p[1]
    dx = exx - beta*x
    dy = eyy - beta*yy
    return [dx, dy]
end

competition (generic function with 1 method)

[128]: tspan = [0.0, 50.0]
u0 = [0., 0.]
p = [1., 0.01]
prob = ODEProblem(competition, u0, tspan, p)
sol = solve(prob, Rodas5())

[128]: 
rectcode: Success
Interpolation: specialized 4th (Rodas5P = 5th) order "free" stiffness-aware interpolation
t: 132-element Vector{Float64}:
 0.0
 0.182695914446915
 0.364196311638577
 0.636392633279537
 1.00518466297680918
 1.4547655311856748
 1.9238539594998375
 2.3923485780748084
 2.859521864278532
 3.331126818642738
 3.75314023232488796
 4.289980875517596
 4.2495385853622125
 45.733325999666626
46.13215492387385
46.5389858448187
```

Рис. 27: Задание №6

Выполнение лабораторной работы

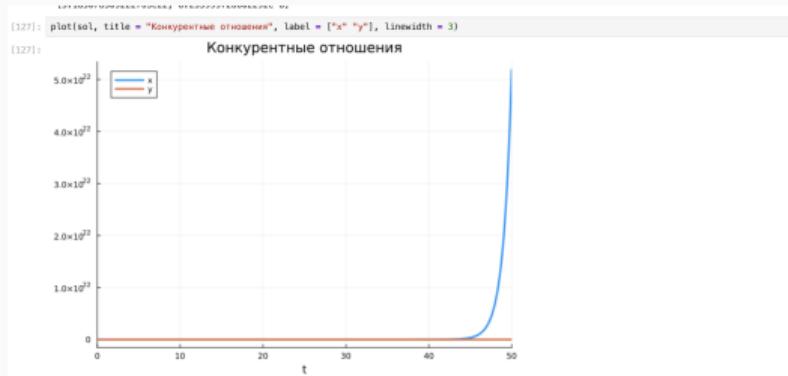


Рис. 28: График №6

Выполнение лабораторной работы

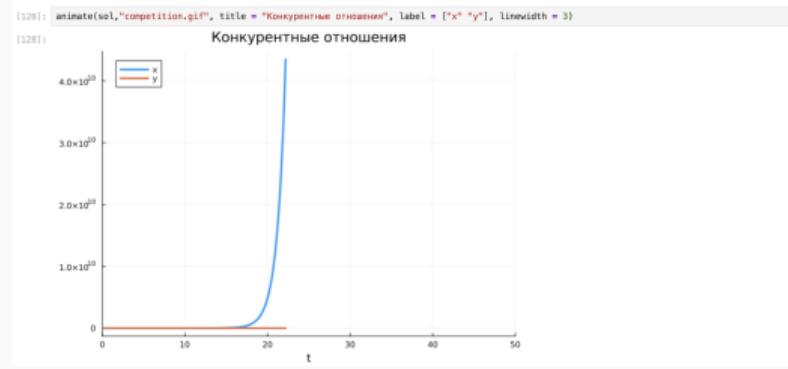


Рис. 29: Анимация №6

Задание 7

Выполнение лабораторной работы

```
Задание 7

29): function harmonic_oscillator!(du, u, p, t)
    x, y = u # y = dx/dt
    ux = p[1]

    du[1] = y # dx/dt = y
    du[2] = -ux^2 * x # dy/dt = -ux^2 * x
end

30): harmonic_oscillator! (generic function with 1 method)

31): # Параметры консервативного осциллятора
u_cons = 2.0 # циклическая частота

# Начальные условия: различные амплитуды и фазы
initial_conditions_osc = [
    [1.0, 0.0], # Начальное отклонение, нулевая скорость
    [2.0, 0.0], # Большое отклонение
    [0.0, 3.0], # Нулевое отклонение, начальная скорость
    [1.5, 2.0] # Комбинированные условия
]
tspan_osc = (0.0, 10.0);
```

Рис. 30: Задание №7

Выполнение лабораторной работы

```
[141]: prob_cons = ODEProblem(harmonic_oscillator!, u0, tspan_osc, [w_cons])
sol_cons = solve(prob_cons, Tsit5i())
plot(sol_cons, title = "Гармонического осциллятор", label = ["x" "y"], linewidth = 3)
```

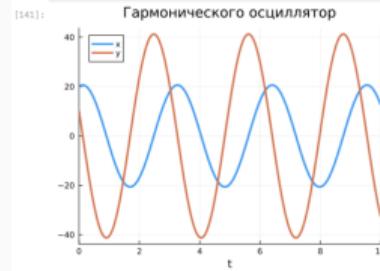


Рис. 31: График №7

Выполнение лабораторной работы

```
[142]: p_cons_time = plot(layout=[2,2], size=[1000, 600])
p_cons_phase = plot(size=[600, 600], legend="bottomleft")

for (idx, u0) in enumerate(initial_conditions_osc)
    prob_cons = ODEProblem(harmonic_oscillator, u0, tspan_osc, [u=u_cons])
    sol_cons = solve(prob_cons, Tsit5f)

    t_points = 0:0.01:10 # временные шаги (положение и скорость)
    x_sol = [sol_cons[t][1] for t in t_points]
    y_dot = [sol_cons[t][2] for t in t_points]

    plot!(p_cons_time[idx], t_points, x_sol, label="x(t)", linewidth=2, color=:blue)
    plot!(p_cons_time[idx], t_points, y_dot, label="y'(t) = \dot{x}(x(t))", linewidth=2, color=:red)
    plot!(p_cons_time[idx], title="Маc. y(u): ($u0[1]), $u0[2])", xlabel="Время", ylabel="Значение", grid=true, legend=:right)

    # фазовый портрет
    plot!(p_cons_phase, x_sol, y_dot, label="Траектория $(u0)", linewidth=2)
    scatter!(p_cons_phase, [u0[1]], [u0[2]], markersize=4, marker=:square, label="")

end

plot!(p_cons_phase, title="Фазовый портрет консервативного осциллятора",
      xlabel="x", ylabel="\dot{x}", grid=true)
```

Рис. 32: График №7

Выполнение лабораторной работы

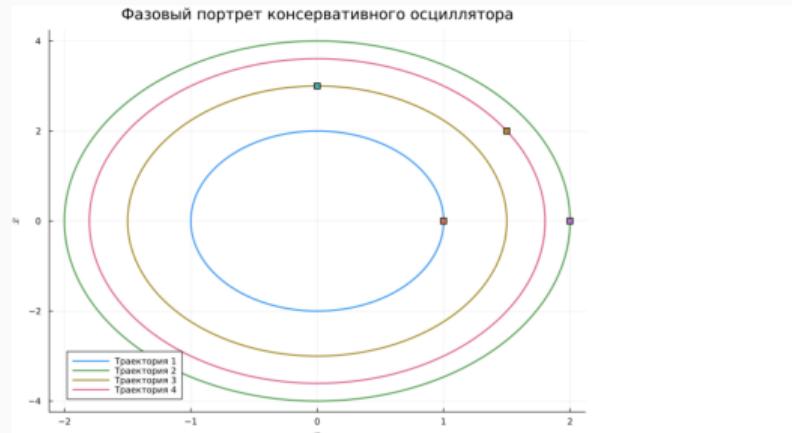


Рис. 33: График №7

Выполнение лабораторной работы

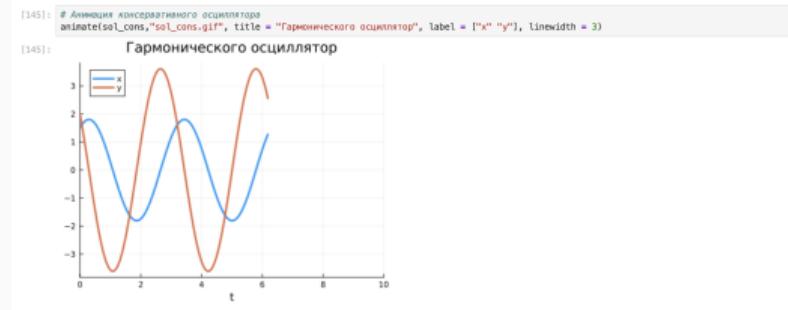


Рис. 34: Анимация №7

Задание 8

Выполнение лабораторной работы

Задание 8

```
[158]: function damped_oscillator!(du, u, p, t)
    x, y = u # y = dx/dt
    ux, vy = p

    du[1] = y # dx/dt = y
    du[2] = -2y * y - ux^2 * x # dy/dt = -2yy - ux^2 x
end
parameters_damped = [(2.0, 0.1), (2.0, 0.5), (2.0, 1.0), (2.0, 1.5) I
u0_damped = [1.0, 0.0] # начальное положение, начальная скорость
tspan_damped = [0.0, 10.0];
```



```
[159]: prob_damped = ODEProblem(damped_oscillator, u0_damped, tspan_damped, [2.0, 0.5])
sol_damped = solve(prob_damped, Tsit5())
plot(sol_damped, title = "Гармонического осциллятор", label = ["x" "y"], linewidth = 3)
```

Гармонического осциллятор

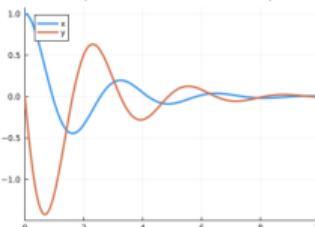


Рис. 35: Задание №8

Выполнение лабораторной работы

```
[152]: # Графики для затухающего осциллятора
p_damped_time = plot(layout=(2,2), size=(1000, 600))
p_damped_phase = plot(size=(800, 600), legend=topright)

regime_names = ["Слабое затухание", "Среднее затухание", "Критическое затухание", "Сильное затухание"]

for (idx, (u0, y)) in enumerate(parameters_damped):
    prob_damped = ODEProblem(damped_oscillator, u0_damped, tspan_damped, (u0, y))
    sol_damped = solve(prob_damped, Tsit5())

    # Временные ряды
    t_points = 0:0.01:10
    x_sol = [sol_damped(t)[1] for t in t_points]
    y_sol = [sol_damped(t)[2] for t in t_points]

    plot(p_damped_time[idx], t_points, x_sol, label="x(t)", linewidth=2, color=blue)
    plot(p_damped_time[idx], t_points, y_sol, label="|dot(x)|", linewidth=2, color=red)
    plot(p_damped_time[idx], title=f"${{\\text{\\textit{{regime}}}_{{\\text{\\textit{{names}}}}}[idx]}}$\\n$u_0 = $u_0$, y = $y$",
         xlabel="t", ylabel="Значение", grid=true, legend=topright)
    # Фазовый портрет
    plot(p_damped_phase, x_sol, y_sol, label=regime_names[idx], linewidth=2)
end

plot!(p_damped_phase, title="Фазовые портреты затухающего осциллятора",
      xlabel=L"x", ylabel=L"\dot{x}", grid=true)
```

Рис. 36: График №8

Выполнение лабораторной работы

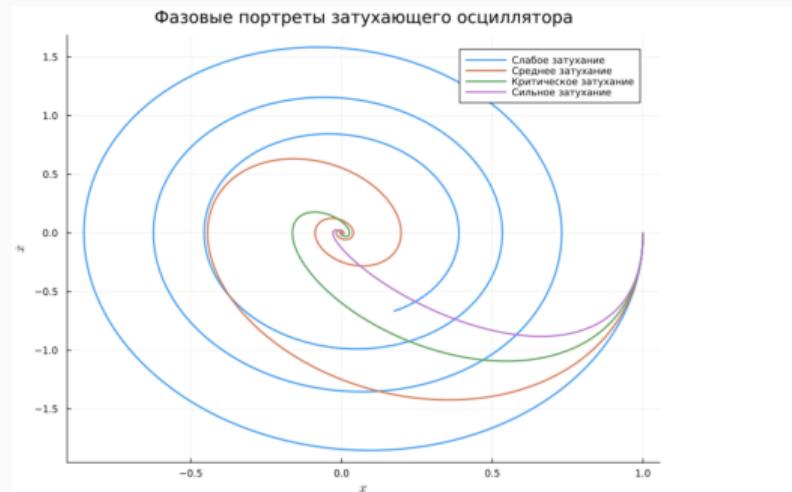


Рис. 37: График №8

Выполнение лабораторной работы

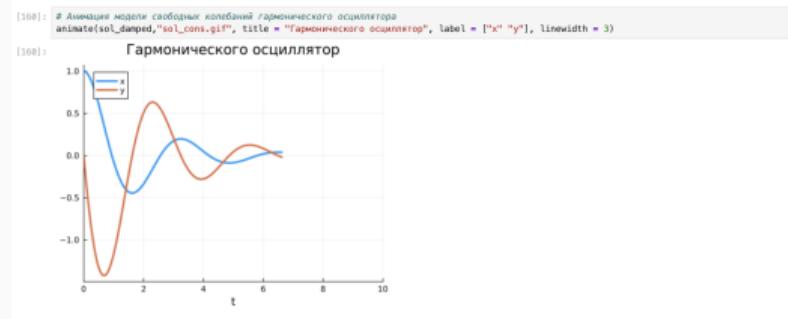


Рис. 38: Анимация №8

Выводы

Выводы

В результате выполнения данной лабораторной работы я освоила специализированные пакеты для решения задач в непрерывном и дискретном времени.