Презентация по лабораторной работе №6 по предмету Компьютерный практикум по статистическому анализу данных

СТУДЕНТ ГРУППЫ НФИБД-01-20 ЕВДОКИМОВ МАКСИМ МИХАЙЛОВИЧ (1032203019)

Цель работы

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

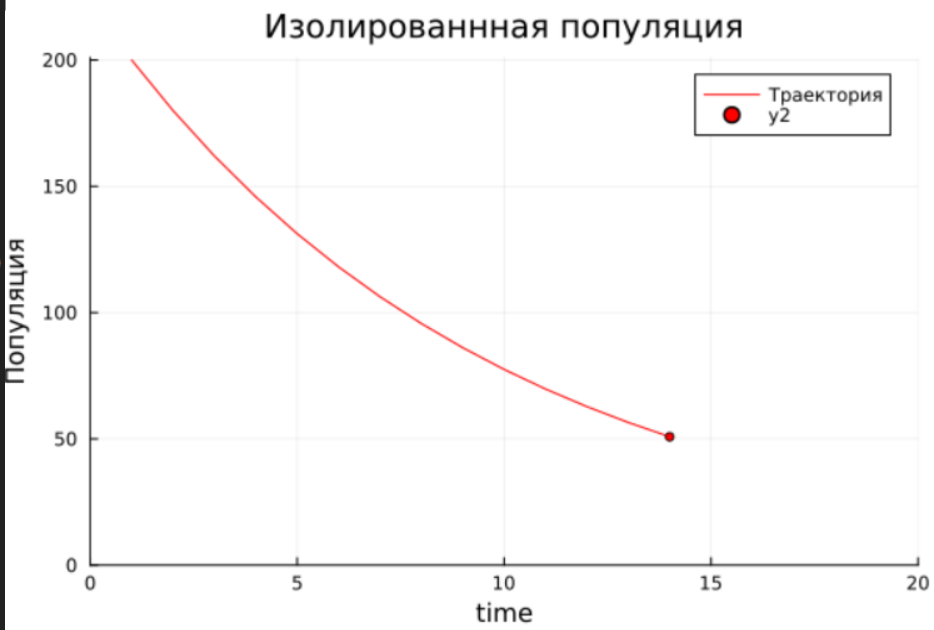
Задание 1

- Реализовать и проанализировать модель роста численности изолированной популяции (модель Мальтуса):

$$\dot{x} = ax, \quad a = b - c.$$

где x(t) — численность изолированной популяции в момент времени t, a — коэффициент роста популяции, b — коэффициент рождаемости, c — коэффициент смертности. Начальные данные и параметры задать самостоятельно и пояснить их выбор. Построить соответствующие графики (в том числе с анимацией).

```julia
# 1
function malthus_model(x0::Int64, b::Float64, c::Float64, t::Int64)
    x = Vector{Float64}([x0])
    a = b - c
    for i in 1:t
        x_new = a * x[i]
        push!(x, x_new)
    end
    return x
end

population = malthus_model(200, 1.4, 0.5, 15)

println("Численность популяции: ", population)
println("Численность популяции после промежуточного отрезка времени (половина): ", population[end÷2])
println("Соотношение начальным и финальным значением в процентах: $(round(population[end]/population[1]*100, digits=2))%")

fig = plt.figure(figsize=[6, 6], tight_layout=true)
ax = fig.add_subplot(title="Модель роста численности изолированной популяции 1", xlabel="time", ylabel="population")
ax.grid(color="k", visible=true, linestyle="--")
ax.plot([i for i in 1:length(population)], population, color="r", marker="o", label="График популяции")
ax.legend()
plt.show()
X = [i for i in 1:length(population)]

anim = @animate for i in 1:length(population)
    plot(X[1:i], population[1:i], c=:red, xlims=(0, 20), ylims=(0, maximum(population)+1),
         xlabel="time", ylabel="Популяция", title="Изолированнная популяция", label="Траектория")
    scatter!([X[i]], [population[i]], markersize=3, markershape=:circle, markercolor=:red)
end
gif(anim, "data//malthus_model.gif")
```




```
Численность популяции: [200.0, 179.99999999999997, 161.99999999999997, 145.79999999999995, 131.21999999999994, 118.09799999999994, 106.28819999
9993, 95.65937999999993, 86.09344199999992, 77.48409779999993, 69.73568801999993, 62.76211921799993, 56.48590729619993, 50.837316566579936, 45.
358490992194, 41.17822641892974]
Численность популяции после промежуточного отрезка времени (половина): 95.65937999999993
Соотношение начальным и финальным значением в процентах: 20.59%
```

Решение задания 1: код и результат

Описание.

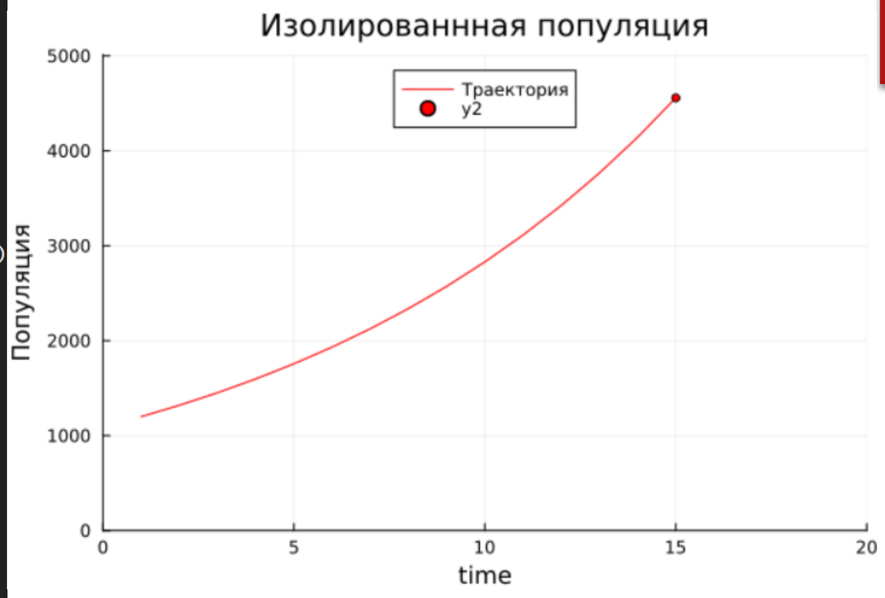
Задание 2

- Реализовать и проанализировать логистическую модель роста популяции, заданную уравнением:

$$\dot{x} = rx\left(1 - \frac{x}{k}\right), \quad r > 0, \quad k > 0$$

r — коэффициент роста популяции, k — потенциальная ёмкость экологической системы (предельное значение численности популяции). Начальные данные и параметры задать самостоятельно и пояснить их выбор. Построить соответствующие графики (в том числе с анимацией).

```julia
# 2
function malthus_model2(x0::Int64, r::Float64, k::Float64, t::Int64)
    x = Vector{Float64}([x0])
    for i in 1:t
        x_new = r * x[i] * (1 - x[i]/k)
        push!(x, x_new)
    end
    return x
end

population = malthus_model(1200, 1.9, 0.8, 15)

println("Численность популяции: ", population)
println("Численность популяции после промежуточного отрезка времени (половина): ", population[end÷2])
println("Соотношение начальным и финальным значением в процентах: $(round(population[end]/population[1]*100, digits=2))%")

fig = plt.figure(figsize=[6, 6], tight_layout=true)
ax = fig.add_subplot(title="Модель роста численности изолированной популяции 2", xlabel="time", ylabel="population")
ax.grid(color="k", visible=true, linestyle="--")
ax.plot([i for i in 1:length(population)], population, color="r", marker="o", label="График популяции")
ax.legend()
plt.show()

anim = @animate for i in 1:length(population)
    plot(X[1:i], population[1:i], c=:red, xlims=(0, 20), ylims=(0, maximum(population)+1), legend=:top,
         xlabel="time", ylabel="Популяция", title="Изолированнная популяция", label="Траектория")
    scatter!([X[i]], [population[i]], markersize=3, markershape=:circle, markercolor=:red)
end
gif(anim, "data//malthus_model2.gif")
```




```
Численность популяции: [1200.0, 1319.9999999999998, 1451.9999999999995, 1597.1999999999994, 1756.9199999999992, 1932.611999999999, 2125.8731999999986, 2338.4605199999983, 2572.3065719999977, 2829.537229199997, 3112.490952119996, 3423.7400473319954, 3766.1140520651943, 4142.725457271713, 4556.998002998884, 5012.697803298772]
Численность популяции после промежуточного отрезка времени (половина): 2338.4605199999983
Соотношение начальным и финальным значением в процентах: 417.72%
```

Решение задания 2: код и результат

Описание.

Задание 3

- Реализовать и проанализировать модель эпидемии Кермака–Маккендрика (SIRмодель):

$$\begin{cases} \dot{s} = -\beta i s, \\ \dot{i} = \beta i s - \nu i, \\ \dot{r} = \nu i, \end{cases}$$

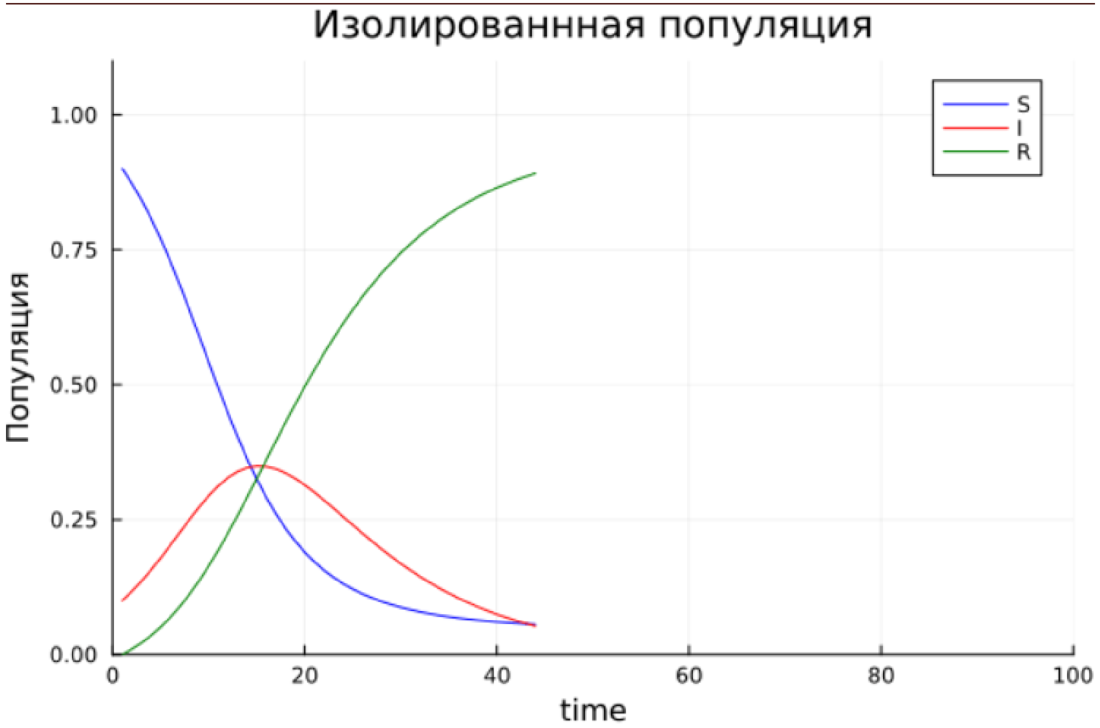
где s(t) — численность восприимчивых к болезни индивидов в момент времени е, i(t) — численность инфицированных индивидов в момент времени t, r(t) — численность переболевших индивидов в момент времени t, β — коэффициент интенсивности контактов индивидов с последующим инфицированием, v — коэффициент интенсивности выздоровления инфицированных индивидов. Численность популяции считается постоянной, т.е. $\dot{s} + \dot{i} + \dot{r} = 0$. Начальные данные и параметры задать самостоятельно и пояснить их выбор. Построить соответствующие графики (в том числе с анимацией).

```julia
# 3
function SIR_model(s0, i0, r0, B, v, t)
    s, i, r = [s0], [i0], [r0]
    for j in 1:t
        ds = -B * s[j] * i[j]
        di = B * s[j] * i[j] - v * i[j]
        dr = v * i[j]

        s_new = s[j] + ds
        i_new = i[j] + di
        r_new = r[j] + dr

        push!(s, s_new)
        push!(i, i_new)
        push!(r, r_new)
    end
    return s, i, r
end

S, I, R = SIR_model(0.9, 0.1, 0.0, 0.3, 0.1, 50)
percents = [[round(s, digits=3) for s in S.*100],
            [round(i, digits=3) for i in I.*100],
            [round(r, digits=3) for r in R.*100]]
X = [j for j in 1:51]
println("Восприимчевых: $(S)\nБольных: $(I)\nЗдоровых: $(R)\n")
println("Восприимчевых%: $(percents[1])\nБольных%: $(percents[2])\nЗдоровых%: $(percents[3])\n")
```

```julia
fig, axs = plt.subplots(ncols=2, nrows=1)
axs[1,1].set(title="SIR модель", xlabel="time", ylabel="population")
axs[1,1].grid(color="k", visible=true, linestyle="--")
axs[1,1].set_yticks([j for j in 0:0.1:1], ["$(i)" for i in 0:0.1:1])
axs[1,1].plot(X, S, color="r", marker="o", label="S")
axs[1,1].plot(X, I, color="g", marker="*", label="I")
axs[1,1].plot(X, R, color="b", marker="P", label="R")
axs[1,1].legend()

axs[2,1].set(title="SIR модель", xlabel="time", ylabel="%")
axs[2,1].grid(color="k", visible=true, linestyle="--")
axs[2,1].set_yticks([j for j in 0:5:100], ["$(i)%" for i in 0:5:100])
axs[2,1].plot(X, percents[1], color="r", marker="o", label="S")
axs[2,1].plot(X, percents[2], color="g", marker="*", label="I")
axs[2,1].plot(X, percents[3], color="b", marker="P", label="R")
axs[2,1].legend()
plt.show()

anim = @animate for i in 1:length(S)
    plot(X[1:i], S[1:i], c=:blue, xlims=(0, 100), ylims=(0, 1.1), legend=:topright,
        xlabel="time", ylabel="Популяция", title="Изолированнная популяция", label="S")
    plot!(X[1:i], I[1:i], c=:red, label="I")
    plot!(X[1:i], R[1:i], c=:green, label="R")
end
gif(anim, "data//SIR_model.gif")
```

Решение задания 3: код

Описание.




```
Восприимчевых: [0.9, 0.873, 0.8423577, 0.808004087051787, 0.7700194069707721, 0.7286656237385065, 0.6844061039198719, 0.6379046015548219, 0.58999
77540824755, 0.5416401497759483, 0.49382773262786744, 0.4475117360548944, 0.4035188241286599, 0.3624918878055019, 0.32486026113493577, 0.29084030
191362875, 0.26046039502726864, 0.233600667139365, 0.21003747219338265, 0.18948497862478775, 0.1716294019079922, 0.15615433270429474, 0.142757607
7968879, 0.13116120348125934, 0.12111593332564355, 0.11240260276485663, 0.10483095445801432, 0.09823738432299443, 0.09248209587502139, 0.08744611
685101729, 0.08302842695740134, 0.07914332742652624, 0.0757181077625832, 0.07269101984265305, 0.07000954436918538, 0.06762892225198602, 0.0655109
1881883349, 0.0636227885159209, 0.06193640987240178, 0.060427563691945727, 0.059075330953149874, 0.05786159033929945, 0.05677059847054216, 0.055
78863869334101, 0.05490372667684759, 0.05410536309350424, 0.0533843253585647, 0.0527324918129931, 0.052142692898235, 0.05160858483474857, 0.051
124542087126766]
Больных: [0.1, 0.117, 0.13594230000000002, 0.15670168294821302, 0.17901619473440655, 0.2024683584932315, 0.22648104246254297, 0.2503344405813387,
0.27320784399555126, 0.294244639025233, 0.31263261466035186, 0.3276853497672897, 0.33890972671679526, 0.346045690368273, 0.34907274800201243,
0.3481854324231182, 0.343747960671665, 0.3362318443483535, 0.3261718548595005, 0.31410716294214536, 0.30055202336472636, 0.2859718902319512, 0.2
707714261161629, 0.25529068782017517, 0.23980688919377344, 0.22453953083518302, 0.20965722605850703, 0.19528507358767622, 0.18151185467688163, 0.
16839664823319755, 0.15597467330349374, 0.14426230550401947, 0.1332612946175606, 0.12296225307573466, 0.11334750324162886, 0.1043933750346534,
0.09607204096435135, 0.08835296717082879, 0.08120404909726503, 0.07459249036799458, 0.06848547406999098, 0.0628506672768423, 0.0576565924179153,
0.05287289295332496, 0.04847051567448589, 0.04442182769038065, 0.04070068265628213, 0.03728244793622551, 0.03414400205677256, 0.031263709915170
24, 0.028621381671275023]
Здоровых: [0.0, 0.010000000000000002, 0.021700000000000004, 0.03529423000000001, 0.050964398294821314, 0.06886601776826197, 0.08911285361758511,
0.11176095786383941, 0.13679440192197329, 0.1641151863215284, 0.19353965271178075, 0.22480291417781595, 0.25757144915454494, 0.29146242182622445,
0.3260669908630518, 0.360974265663253, 0.39579280890556484, 0.4301674885122815, 0.46379067294711684, 0.4964078584330669, 0.5278185747272814, 0.55
78737770637541, 0.5864709660869492, 0.6135481086985655, 0.639077177480583, 0.6630578663999603, 0.6855118194834786, 0.7064775420893293, 0.72600604
9448097, 0.7441572349157851, 0.7609968997391049, 0.7765943670694543, 0.7910205976198562, 0.8043467270816123, 0.816642952389185, 0.82797770271334
86, 0.8384170402168151, 0.8480242443132502, 0.856859541030333, 0.8649799459400596, 0.8724391949768591, 0.8792877423838582, 0.8855728091115425, 0.
891338468353334, 0.8966257576486665, 0.9014728092161151, 0.9059149919851531, 0.9099850602507813, 0.9137133050444038, 0.9171277052500811, 0.920254
0762415982]

Восприимчевых%: [90.0, 87.3, 84.236, 80.8, 77.002, 72.867, 68.441, 63.79, 59.0, 54.164, 49.383, 44.751, 40.352, 36.249, 32.486, 29.084, 26.046, 2
3.36, 21.004, 18.948, 17.163, 15.615, 14.276, 13.116, 12.112, 11.24, 10.483, 9.824, 9.248, 8.745, 8.303, 7.914, 7.572, 7.269, 7.001, 6.763, 6.55
1, 6.362, 6.194, 6.043, 5.908, 5.786, 5.677, 5.579, 5.49, 5.411, 5.338, 5.273, 5.214, 5.161, 5.112]
Больных%: [10.0, 11.7, 13.594, 15.67, 17.902, 20.247, 22.648, 25.033, 27.321, 29.424, 31.263, 32.769, 33.891, 34.605, 34.907, 34.819, 34.375, 33.
623, 32.617, 31.411, 30.055, 28.597, 27.077, 25.529, 23.981, 22.454, 20.966, 19.529, 18.151, 16.84, 15.597, 14.426, 13.326, 12.296, 11.335, 10.43
9, 9.607, 8.835, 8.12, 7.459, 6.849, 6.285, 5.766, 5.287, 4.847, 4.442, 4.07, 3.728, 3.414, 3.126, 2.862]
Здоровых%: [0.0, 1.0, 2.17, 3.529, 5.096, 6.887, 8.911, 11.176, 13.679, 16.412, 19.354, 22.48, 25.757, 29.146, 32.607, 36.097, 39.579, 43.017, 4
6.379, 49.641, 52.782, 55.787, 58.647, 61.355, 63.908, 66.306, 68.551, 70.648, 72.601, 74.416, 76.1, 77.659, 79.102, 80.435, 81.664, 82.798, 83.8
42, 84.802, 85.686, 86.498, 87.244, 87.929, 88.557, 89.134, 89.663, 90.147, 90.591, 90.999, 91.371, 91.713, 92.025]
```

Решение задания 3: результат

Описание.

Задание 4

- Как расширение модели SIR (Susceptible-Infected-Removed) по результатом эпидемии испанки была предложена модель SEIR (Susceptible-Exposed-Infected-Removed):

$$\begin{cases} \dot{s}(t) = -\frac{\beta}{N} s(t) i(t), \\ \dot{e}(t) = \frac{\beta}{N} s(t) i(t) - \delta e(t), \\ \dot{i}(t) = \delta e(t) - \gamma i(t), \\ \dot{r}(t) = \gamma i(t). \end{cases} \qquad s(t) + e(t) + i(t) + r(t) = N$$

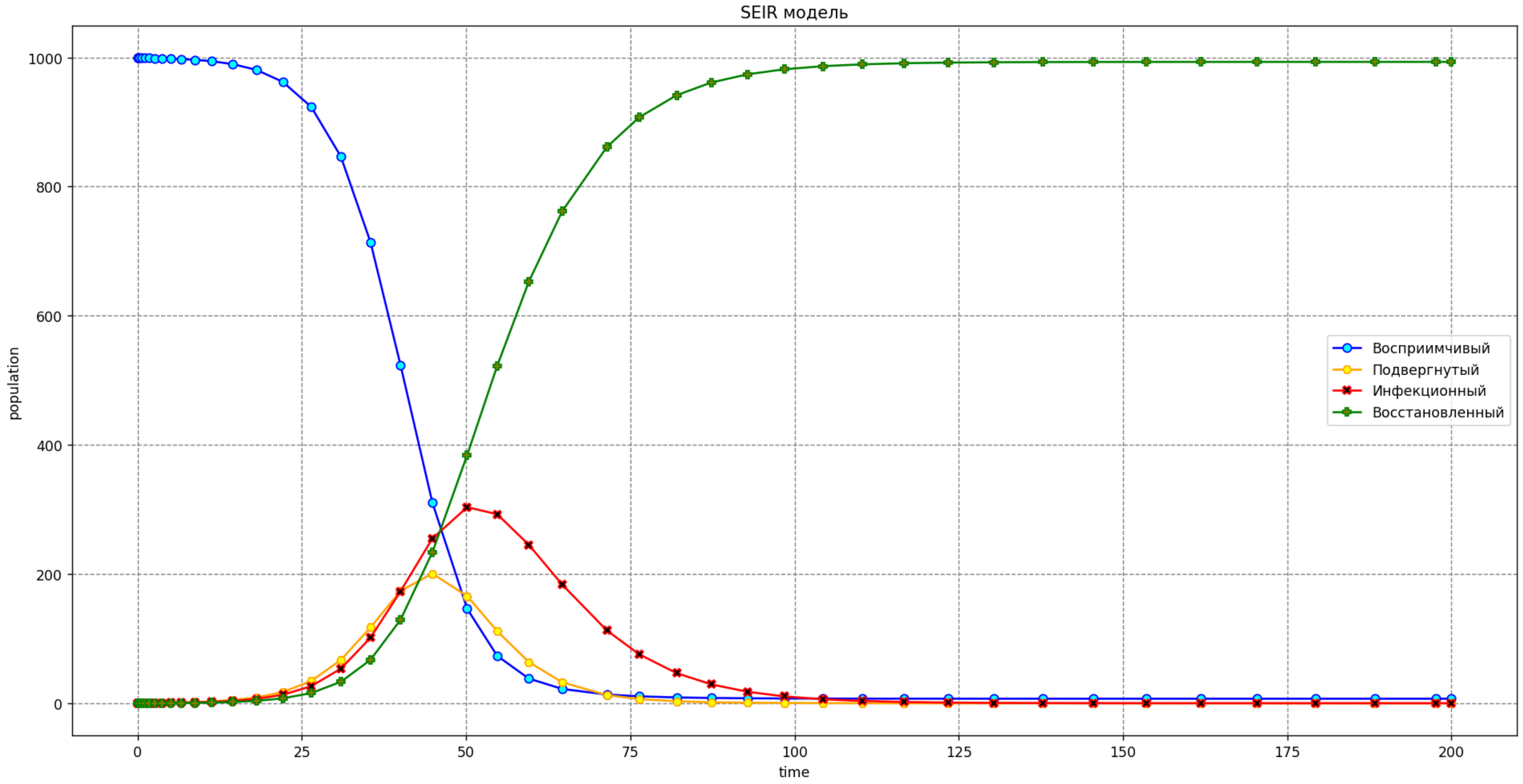
Размер популяции сохраняется, исследуйте, сравните с SIR.

```
# 4
function seir_model!(du, u, p, t)
    β, σ, γ, N = p
    s, e, i, r = u
    du[1] = -β * s * i / N
    du[2] = β * s * i / N - σ * e
    du[3] = σ * e - γ * i
    du[4] = γ * i
end

function simulate_seir_model(β, σ, γ, N, s0, e0, i0, r0, time)
    u0 = [s0, e0, i0, r0]
    p = [β, σ, γ, N]
    prob = ODEProblem(seir_model!, u0, time, p)
    sol = solve(prob, Tsit5())
    return sol
end

β, σ, γ, N = 0.5, 0.2, 0.1, 1000
s0, e0, i0, r0 = N - 1, 1, 0, 0
time = (0.0, 200.0)
```

```
sol = simulate_seir_model(β, σ, γ, N, s0, e0, i0, r0, time)
S, E, I, R = [], [], [], []
for j in sol.u push!(S, j[1]); push!(E, j[2]); push!(I, j[3]); push!(R, j[4]) end

fig = plt.figure(figsize=[6, 6], tight_layout=true)
ax = fig.add_subplot(title="SEIR модель", xlabel="time", ylabel="population")
ax.grid(color="gray", visible=true, linestyle="--")
ax.plot(sol.t, S, color="blue", marker="o", label="Восприимчивый", fillstyle="full", markerfacecolor="cyan")
ax.plot(sol.t, E, color="orange", marker="H", label="Подвергнутый", fillstyle="full", markerfacecolor="yellow")
ax.plot(sol.t, I, color="red", marker="X", label="Инфекционный", fillstyle="full", markerfacecolor="black")
ax.plot(sol.t, R, color="green", marker="P", label="Восстановленный", fillstyle="full", markerfacecolor="olive")
ax.legend()
plt.show()
```

Решение задания 4: код

Описание.



Решение задания 4: результат

Описание.

Задание 5

- Для дискретной модели Лотки–Вольтерры:

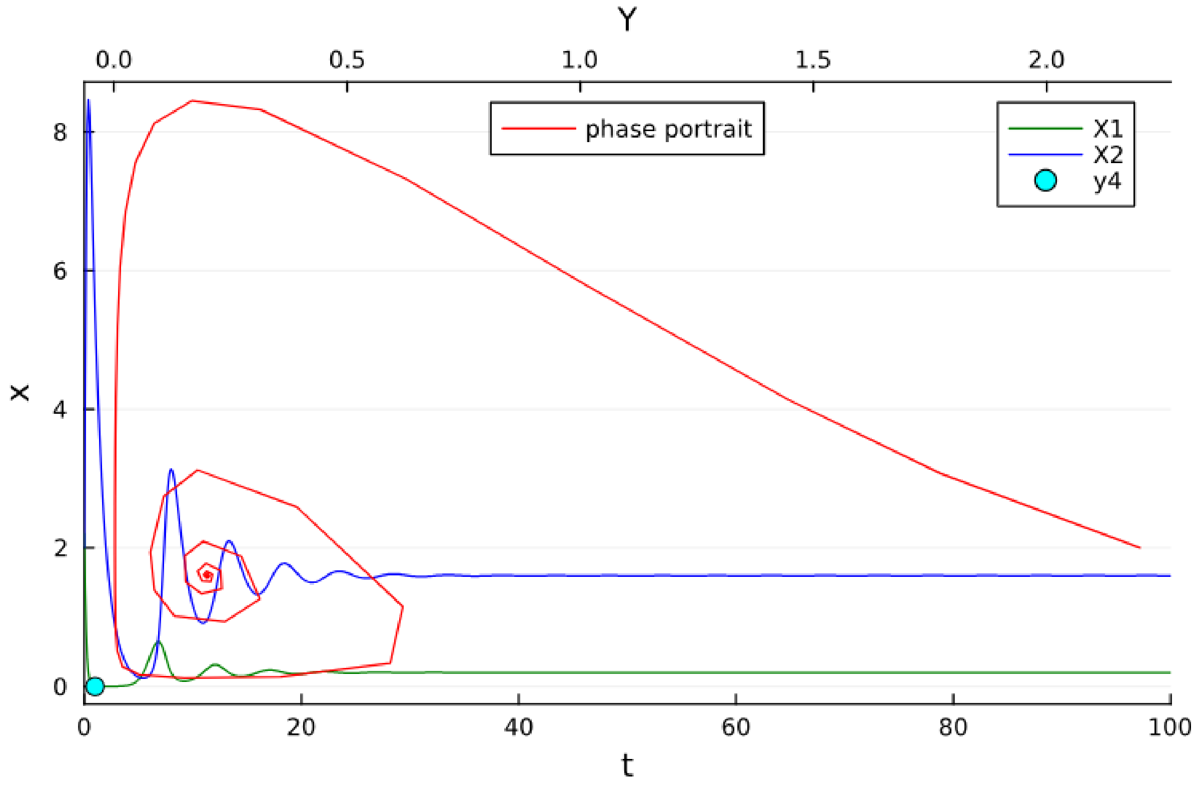
$$\begin{cases} X_1(t+1) = aX_1(t)(1 - X_1(t)) - X_1(t)X_2(t), \\ X_2(t+1) = -cX_2(t) + dX_1(t)X_2(t). \end{cases}$$

с начальными данными a = 2, c = 1, d = 5 (b = 1) найдите точку равновесия. Получите и сравните аналитическое и численное решения. Численное решение изобразите на фазовом портрете.

```julia
# 5
using NLsolve


function lotka_volterra!(du, u, p, t)
    du[1] = p[1]*u[1]*(1-u[1]) - p[2]*u[1]*u[2]
    du[2] = -p[3]*u[2] + p[4]*u[1]*u[2]
end


u0, X, Y = [2.2, 2.0], Vector([]), Vector([])
time = (0.0, 100.0)
p = [2.0, 1.0, 1.0, 5.0]
prob = ODEProblem(lotka_volterra!, u0, time, p)
sol = solve(prob, Tsit5())
for j in sol.u push!(X, j[1]); push!(Y, j[2]) end

f(x) = [2*x[1]*(1-x[1]) - 1*x[1]*x[2], -1*x[2] + 5*x[1]*x[2]]
equilibrium = nlsolve(f, [2.2, 2.0])
println("Equilibrium point: ", equilibrium)
z = equilibrium.zero

plot(sol, vars=(0, 1), color=:green, xlabel="t", ylabel="x", label="X1", )
plot!(sol, vars=(0, 2), color=:blue, xlabel="t", ylabel="x", label="X2")
plot!(twiny(), X, Y, color=:red, xlabel="Y", label="phase portrait", legend=:top)
scatter!([z[1]], [z[2]], color=:cyan, markersize=5)
```

Решение задания 5: код

Описание.




```
Equilibrium point: Results of Nonlinear Solver Algorithm
 * Algorithm: Trust-region with dogleg and autoscaling
 * Starting Point: [2.2, 2.0]
 * Zero: [0.9999999999751682, 4.693334918837513e-10]
 * Inf-norm of residuals: 0.000000
 * Iterations: 5
 * Convergence: true
   * |x - x'| < 0.0e+00: false
   * |f(x)| < 1.0e-08: true
 * Function Calls (f): 6
 * Jacobian Calls (df/dx): 6
```

Решение задания 5: результат

Описание.

Задание 6

- Реализовать на языке Julia модель отбора на основе конкурентных отношений:

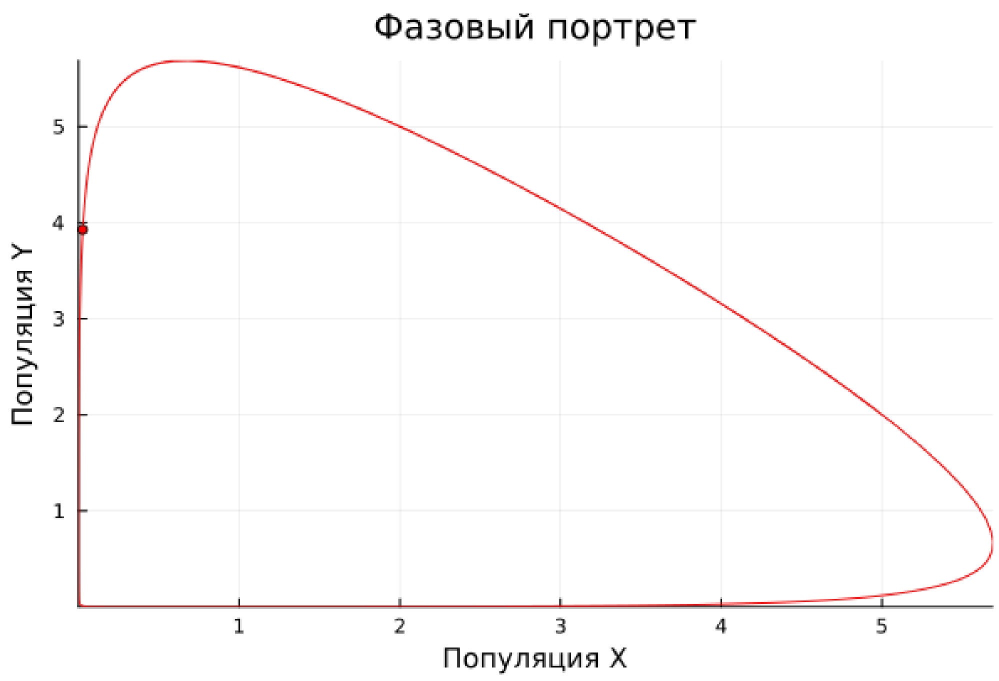
$$\begin{cases} \dot{x} = \alpha x - \beta x y, \\ \dot{y} = \alpha y - \beta x y. \end{cases}$$

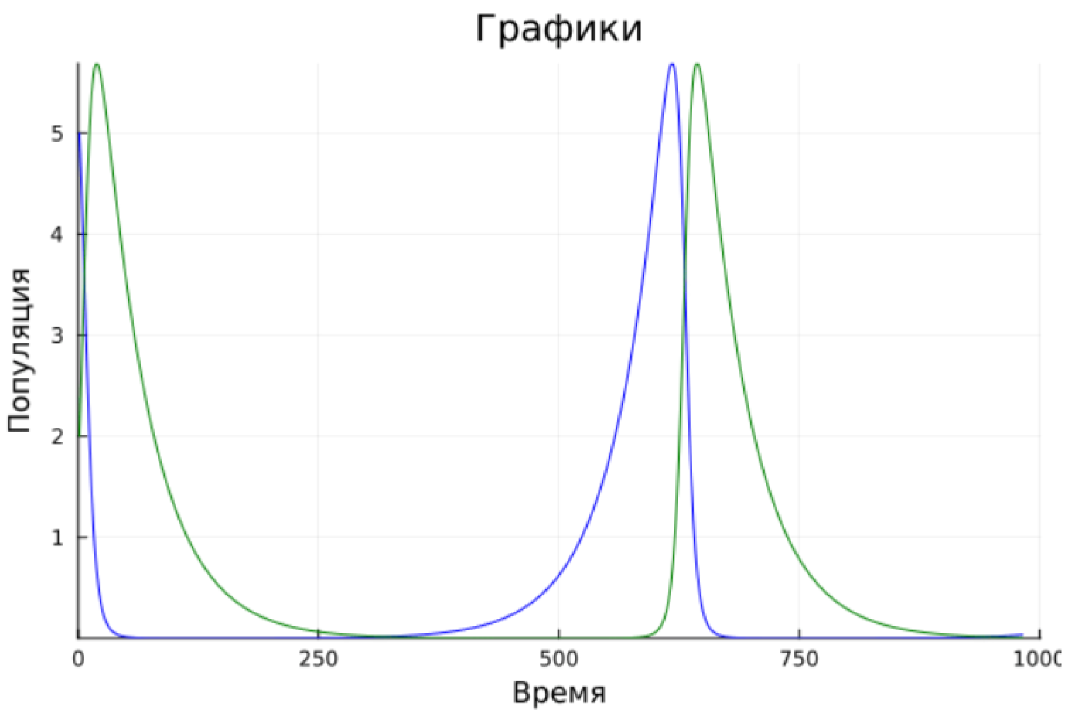
Начальные данные и параметры задать самостоятельно и пояснить их выбор. Построить соответствующие графики (в том числе с анимацией) и фазовый портрет.

```
# 6
gr()
function relationship!(du, u, p, t)
    du[1] = p[1]*u[1] - p[2]*u[1]*u[2]
    du[2] = -p[1]*u[2] + p[2]*u[1]*u[2]
end

u0, p = [5, 2], [0.2, 0.3]
time, X, Y = 100, [], []
prob = ODEProblem(relationship!, u0, time, p)
sol = solve(prob, Tsit5(), saveat=0.1)

for j in sol.u push!(X, j[1]); push!(Y, j[2]) end
N = Vector(1:size(X)[1])

anim1 = @animate for i in 1:N[end]
    plot(X[1:i], Y[1:i], c=:red, xlims=(minimum(X), maximum(X)), ylims=(minimum(Y), maximum(Y)), legend=false,
         xlabel="Популяция X", ylabel="Популяция Y", title="Фазовый портрет", label="Траектория")
    scatter!([X[i]], [Y[i]], markersize=3, markershape=:circle, markercolor=:red)
end; gif(anim1, "data//competitive_relationshipF.gif")

anim2 = @animate for i in 1:N[end]
    plot(N[1:i], X[1:i], c=:blue, xlims=(0, N[end]), ylims=(minimum(X), maximum(X)), legend=false,
         xlabel="Время", ylabel="Популяция", title="Графики", label="Траектория")
    plot!(N[1:i], Y[1:i], c=:green)
end; gif(anim2, "data//competitive_relationshipT.gif")
```

Решение задания 6: код

Описание.





Решение задания 6: результат

Описание.

Задание 7

- Реализовать на языке Julia модель консервативного гармонического осциллятора:

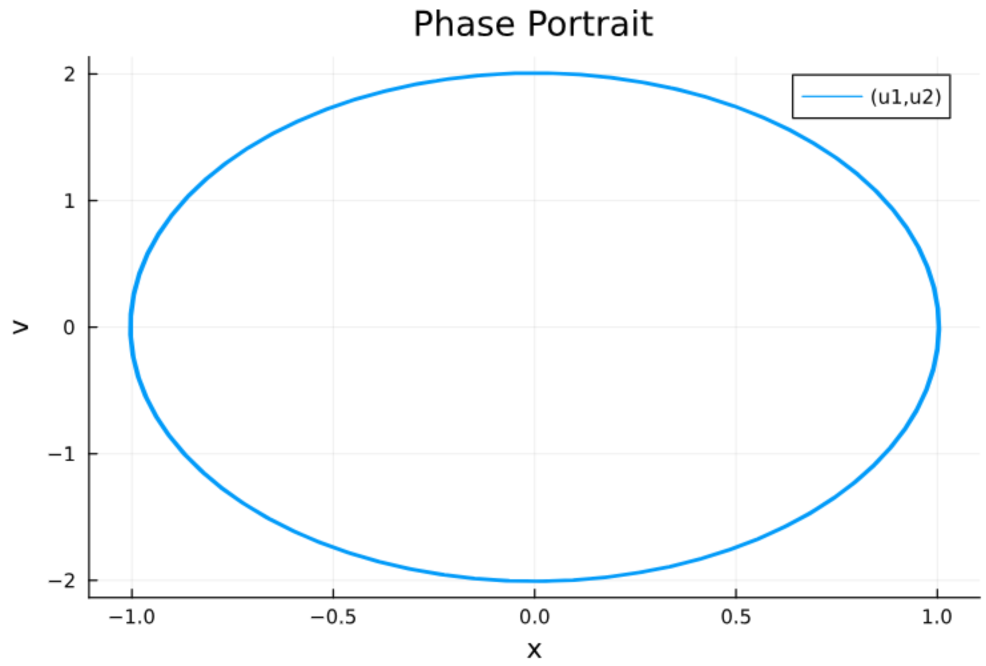
$$\ddot{x} + \omega_0^2 x = 0, \quad x(t_0) = x_0, \quad \dot{x}(t_0) = y_0$$

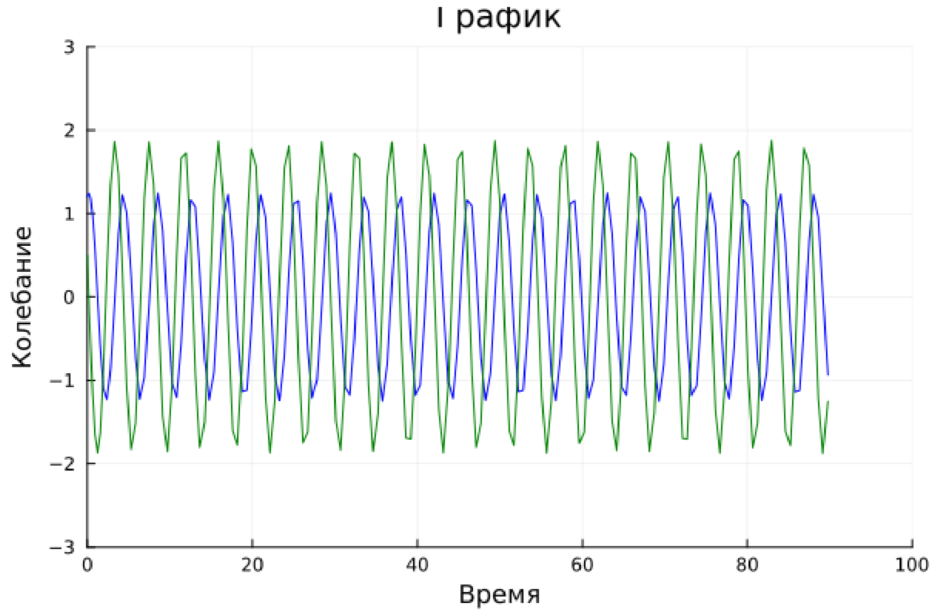
где $\omega 0$ — циклическая частота. Начальные параметры подобрать самостоятельно, выбор пояснить. Построить соответствующие графики (в том числе с анимацией) и фазовый портрет.

```julia
# 7
function harmonic_oscillator!(du, u, p, t)
    du[1] = u[2]
    du[2] = -p^2 * u[1]
end

x0, v0, w0 = 1.2, 0.5, 1.5
u0, X, Y = [x0, v0], [], []
tspan = (0.0, 100.0)
probh = ODEProblem(harmonic_oscillator!, u0, tspan, w0)
solh = solve(probh)
for j in solh.u push!(X, j[1]); push!(Y, j[2]) end
N = solh.t

anim1 = @animate for i in 1:length(N)
    plot(X[1:i], Y[1:i], c=:red, xlims=(-3, 3), ylims=(-3, 3), legend=false,
         xlabel="X", ylabel="Y", title="Фазовый портрет")
    scatter!([X[i]], [Y[i]], markersize=3, markershape=:circle, markercolor=:red)
end; gif(anim1, "data//harmonic_oscillatorF.gif")

anim2 = @animate for i in 1:length(N)
    plot(N[1:i], X[1:i], c=:blue, xlims=(0, N[end]), ylims=(-3, 3), legend=false,
         xlabel="Время", ylabel="Колебание", title="График")
    plot!(N[1:i], Y[1:i], c=:green)
end; gif(anim2, "data//harmonic_oscillatorT.gif")

plot(sol, vars=(1,2), xlabel="x", ylabel="v", title="Phase Portrait")
```

Решение задания 7: код

Описание.





Решение задания 7: результат

Описание.

Задание 8

- Реализовать на языке Julia модель свободных колебаний гармонического осциллятора:

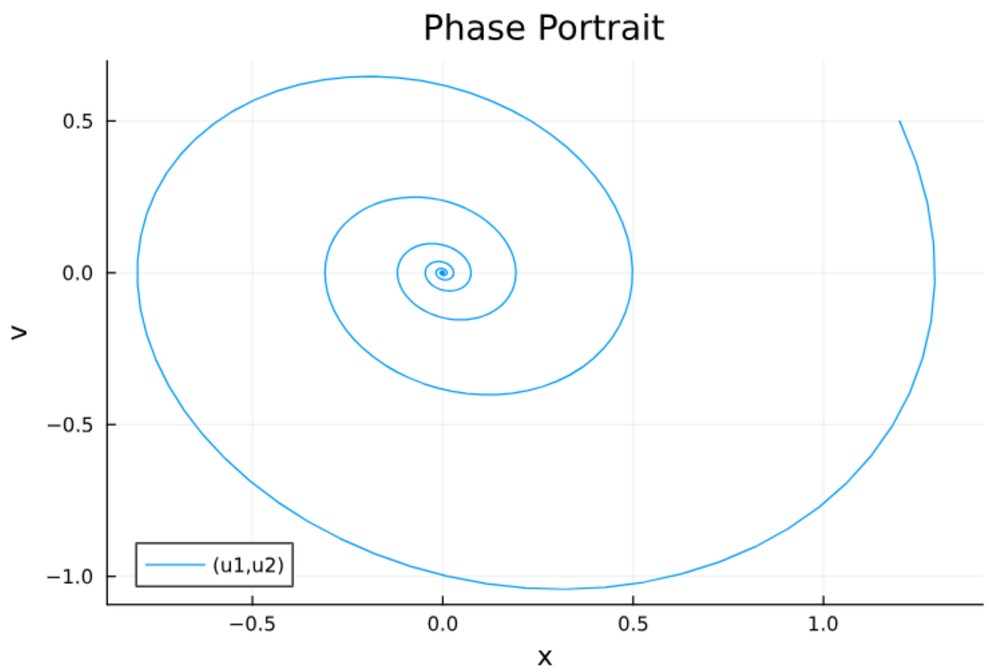
$$\ddot{x} + 2\gamma\dot{x} + \omega_0^2 x = 0, \quad x(t_0) = x_0, \quad \dot{x}(t_0) = y_0$$

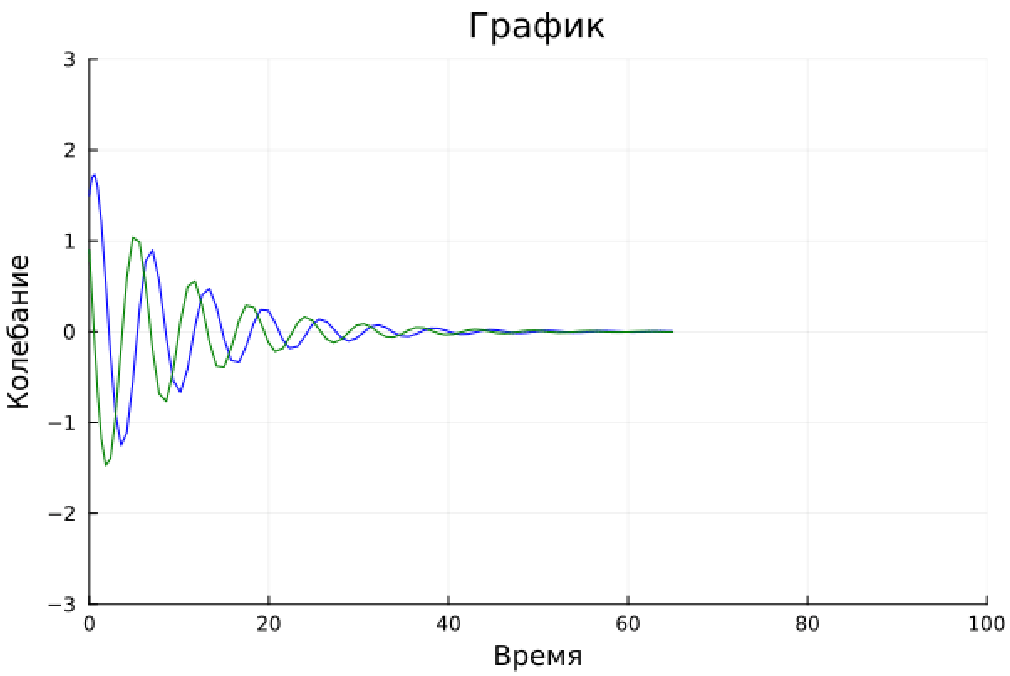
где $\omega 0$ — циклическая частота, γ — параметр, характеризующий потери энергии. Начальные параметры подобрать самостоятельно, выбор пояснить. Построить соответствующие графики (в том числе с анимацией) и фазовый портрет.

```
# 8
function fading_harmonic_oscillator!(du, u, p, t)
    w0, γ = p
    du[1] = u[2]
    du[2] = -γ * u[2] - w0^2 * u[1]
end

p, u0 = [1.0, 0.2], [1.5, 0.9]
tspan = (0.0, 100.0)
X, Y = [], []
prob = ODEProblem(fading_harmonic_oscillator!, u0, tspan, p)
solf = solve(prob, Tsit5())
for j in solf.u push!(X, j[1]); push!(Y, j[2]) end
N = solf.t

anim1 = @animate for i in 1:length(N)
    plot(X[1:i], Y[1:i], c=:red, xlims=(-2, 2), ylims=(-2, 2), legend=false,
         xlabel="X", ylabel="Y", title="Фазовый портрет")
    scatter!([X[i]], [Y[i]], markersize=3, markershape=:circle, markercolor=:red)
end; gif(anim1, "data//fading_harmonic_oscillatorF.gif")

anim2 = @animate for i in 1:length(N)
    plot(N[1:i], X[1:i], c=:blue, xlims=(0, N[end]), ylims=(-3, 3), legend=false,
         xlabel="Время", ylabel="Колебание", title="График")
    plot!(N[1:i], Y[1:i], c=:green)
end; gif(anim2, "data//fading_harmonic_oscillatoTr.gif")


plot(sol, vars=(1,2), xlabel="x", ylabel="v", title="Phase Portrait")
```

Решение задания 8: код

Описание.





Решение задания 8: результат

Описание.

Заключение

- В ходе выполнения лабораторной работы были изучены и построены различные непрерывные, линейные и дискретные модели времени. Повторены методы построения различных графиков, фазовых портретов и анимации.