Лабораторная работа № 6

Покрас Илья Михайлович

2023, Москва

Цель работы

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

Ход работы

```julia
using Plots

function malthus(start, b, c, t)
    a = b - c
    x = Vector{Float64}([start])
    for t in 1:t
        push!(x, a * x[end])
    end
    return x
end

p = malthus(100, 1.4, 0.5, 20)

anim = @animate for i in 1:length(p)
    plot(1:i, p[1:i], markersize=3, markershape=:circle, xlims=(1, length(p)), ylims=(last(p), p[1]))
end

gif(anim, "malthus.gif")
```

```
[ Info: Saved animation to C:\Users\marin\Documents\UNI\DA\Lab6\malthus.gif
```

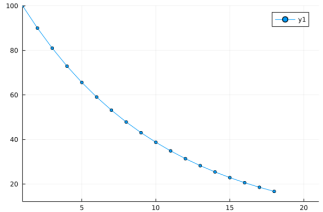


Рис. 1: Пункт 1

Ход работы

```
function malthus2(start, r, k, t, dt)
    x = Vector{Float64}([start])
    for t in 1:t
        deltax = r * x[end] * (1 - x[end] / k) * dt
        push!(x, x[end] + deltax)
    end
    return x
end

p2 = malthus2(20, 0.2, 85, 20, 0.1)

anim2 = @animate for i in 1:length(p2)
    plot(1:i, p2[1:i], markersize=3, markershape=:circle, xlims=(1, length(p2)), ylims=(p2[1], last(p2)))
end

gif(anim2, "malthus2.gif")
```

```
[ Info: Saved animation to C:\Users\marin\Documents\UNI\DA\Lab6\malthus2.gif
```

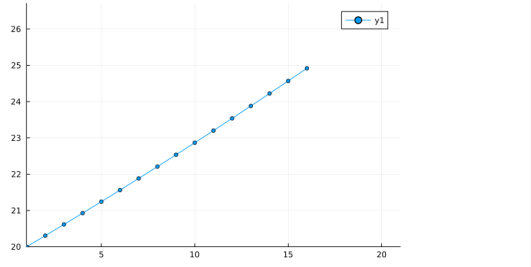


Рис. 2: Пункт 2

Ход работы

```julia
using Random

function kermack_mckendrick(N, I0, R0, β, γ, T)
    S0 = N - I0 - R0
    S = Vector{Float64}([S0])
    I = Vector{Float64}([I0])
    R = Vector{Float64}([R0])
    t = Vector{Float64}([0])
    dt = 0.1

    while t < T
        if t + dt > T
            dt = T - t
        end

        new_infections = β * S[end] * I[end] / N
        new_recoveries = γ * I[end]

        S_new = S[end] - new_infections
        I_new = I[end] + new_infections - new_recoveries
        R_new = R[end] + new_recoveries

        push!(S, S_new)
        push!(I, I_new)
        push!(R, R_new)

        t += dt
    end

    return S, I, R
end


N = 100
I0 = 1.0
R0 = 0.0
β = 0.2
γ = 0.1
T = 10
S, I, R = kermack_mckendrick(N, I0, R0, β, γ, T)

t = range(0, T, length=length(S))
plot(xlabel="Time", ylabel="Population")

anim = @animate for i = 1:length(S)
    plot!(t[1:i-1], [S[1:i-1], I[1:i-1]], color=:blue, ylims=(0, N), xlims=(0, last(t)))
    scatter!([t[i]], [S[i]], markersize=3, color=:blue, legend=false)

    plot!([t[1:i-1]], [I[1:i-1]], color=:red)
    scatter!([t[i]], [I[i]], markersize=3, color=:red, legend=false)

    plot!(t[1:i-1], R[1:i-1], color=:green)
    scatter!([t[i]], [R[i]], markersize=3, color=:green, legend=false)
end

gif(anim, "SIR1.gif")
```

Рис. 3: Пункт 3 - код

Ход работы

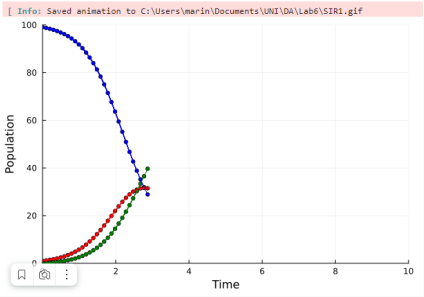


Рис. 4: Пункт 3 - визуализация

Ход работы

```
function seir_model(N, EB, IB, RB, β, σ, γ, T)
    SB = N - EB - IB - RB
    S = Vector{Float64}([SB])
    E = Vector{Float64}([EB])
    I = Vector{Float64}([IB])
    R = Vector{Float64}([RB])
    S = Vector{Float64}([SB])
    t = 0.0
    dt = 0.1

    while t < T
        if t + dt > T
            dt = T - t
        end

        new_exposed = β * S[end] * I[end] / N
        new_infections = σ * E[end]
        new_recoveries = γ * I[end]

        S_new = S[end] - new_exposed
        E_new = E[end] + new_exposed - new_infections
        I_new = I[end] + new_infections - new_recoveries
        R_new = R[end] + new_recoveries

        push!(S, S_new)
        push!(E, E_new)
        push!(I, I_new)
        push!(R, R_new)

        t += dt
    end

    return S, E, I, R
end

N = 100
EB = 1.0
IB = 1.0
RB = 0.0
β = 0.3
σ = 0.2
γ = 0.1
T = 10

S, E, I, R = seir_model(N, EB, IB, RB, β, σ, γ, T)

t = range(0, T, length=length(S))
plot(xlabel="Time", ylabel="Population")

anim4 = @animate for i = 1:length(S)
    plot!(t[1:i], [S[1:i], I[1:i]], color=:blue, ylims=(0, N), xlims=(0, last(t)))
    scatter!([t[i]], [S[i]], markersize=1, color=:blue, legend=false)

    plot!([t[1:i], I[1:i]], color=:orange)
    scatter!([t[i]], [I[i]], markersize=1, color=:orange, legend=false)

    plot!([t[1:i]], [I[1:i]], color=:red)
    scatter!([t[i]], [I[i]], markersize=2, color=:red, legend=false)

    plot!(t[1:i], R[1:i], color=:green)
```

Рис. 5: Пункт 4 - код

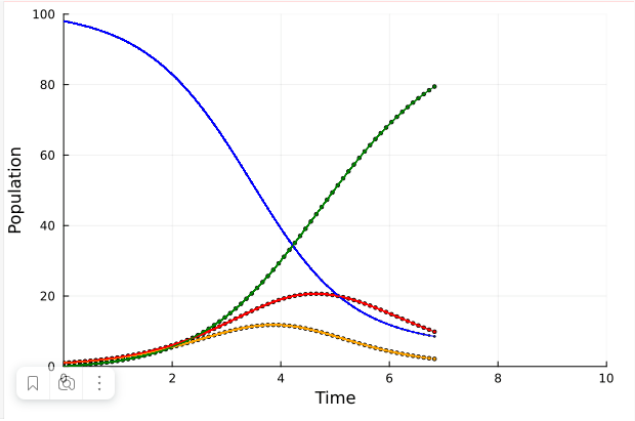
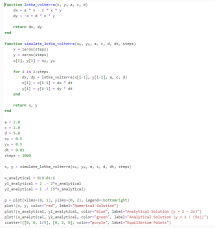


Рис. 6: Пункт 4 - визуализация

Ход работы



```
function lotka_volterra(x, y, a, b, c, d)
    dx = a * x - c * x * y
    dy = -b * d * x + y
    return dx, dy
end

function simulate_lotka_volterra(x0, y0, a, b, c, d, dt, steps)
    x = zeros(steps)
    y = zeros(steps)
    x[1], y[1] = x0, y0

    for i in 2:steps
        dx, dy = lotka_volterra(x[i-1], y[i-1], a, c, d)
        x[i] = x[i-1] + dx * dt
        y[i] = y[i-1] + dy * dt
    end

    return x, y
end

a = 2.0
b = 1.0
d = 5.0
c = 0.5
x0 = 0.5
dt = 0.01
steps = 2000

x, y = simulate_lotka_volterra(x0, y0, a, b, c, d, dt, steps)

x_analytical = 0:0.01:5
y1_analytical = 2 .- 2*x_analytical
y2_analytical = 1 .// (3*x_analytical)

p = plot(size=(10, 7), xlims=(0, 5), legend=:bottomright)
plot!(x, y, color="red", label="Numerical Solution")
plot!(x_analytical, y1_analytical, color="blue", label="Analytical Solution (y = 2 - 2x)")
plot!(x_analytical, y2_analytical, color="green", label="Analytical Solution (y = 1 / (3x))")
scatter!([0, 0, 1/3], [0, 2, 0], color="purple", label="Equilibrium Points")
```

Рис. 7: Пункт 5 - код

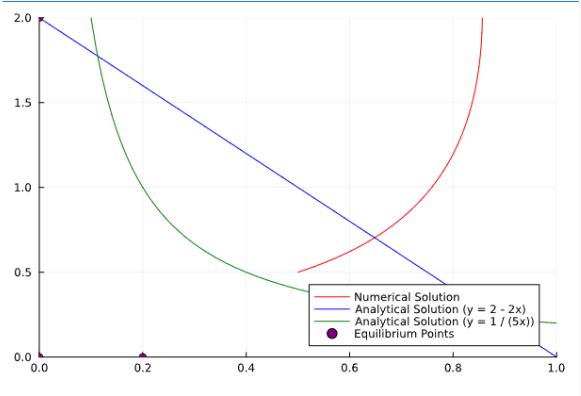


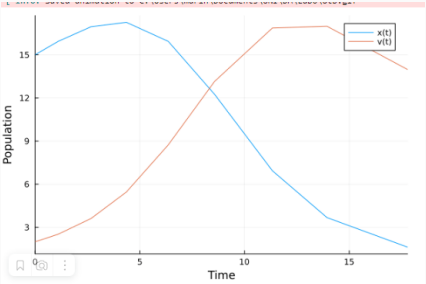
Рис. 8: Пункт 5 - визуализация

Ход работы

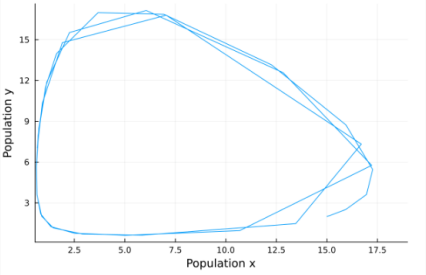
```julia
using DifferentialEquations

function competitive_selection!(du, u, p, t)
    alfa, betta = p
    du[1] = alfa * u[1] - betta * u[1] * u[2]
    du[2] = -alfa * u[2] + betta * u[1] * u[2]
end

alfa = 0.1
betta = 0.02
x0 = 15.0
y0 = 2.0


u0 = [x0, y0]
tspan = (0.0, 200.0)
p = [alfa, betta]

prob = ODEProblem(competitive_selection!, u0, tspan, p)
sol = solve(prob)

anim5 = @animate for i in 1:length(sol)
    plot(sol[1:i], label=["x(t)" "v(t)"], xlabel="Time", ylabel="Population")
end

anim6 = @animate for i in 1:length(sol)
    plot(sol[1:i], vars=(1,2), xlabel="Population x", ylabel="Population y", label="")
end

display(gif(anim5, "otb.gif"))

gif(anim6, "phase1.gif")
```

Рис. 9: Пункт 6 - код

Ход работы



[Info: Saved animation to C:\Users\marin\Documents\UNI\DA\Lab6\phase1.gif



Ход работы

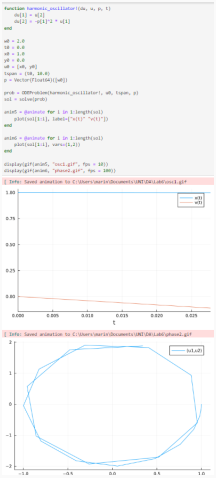


Рис. 11: Пункт 7

Ход работы

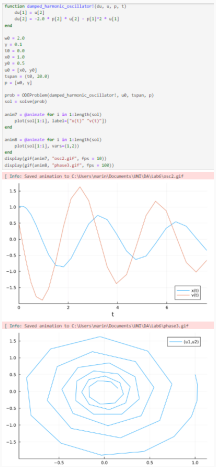


Рис. 12: Пункт 8

Выводы

В ход выполнения работы Я освоил специализированные пакеты для решения задач в непрерывном и дискретном времени.