Лабораторная работа №6.

Решение моделей в непрерывном и дискретном времени

Тазаева А. А.

Российский университет дружбы народов, Москва, Россия

Цели работы

Цели работы

Основная цель работы — освоение специализированных пакетов для решения задач в непрерывном и дискретном времени.

Задание

1. Используя Jupyter Lab, повторите примеры из раздела 6.2.
2. Выполните задания для самостоятельной работы (раздел 6.4).

Примеры из раздела 6.2

6.2.1. Решение обыкновенных дифференциальных уравнений

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

6.2.1.1. Модель экспоненциального роста

Рассмотрим пример использования этого пакета для решение уравнения модели экспоненциального роста, описываемую уравнением

$$u'(t) = au(t), u(0) = u_0.$$

где a - коэффициент роста. Предположим, что заданы следующие начальные данные $a = 0,98$, $u(0) = 1,0$, $t \in [0;1,0]$ Аналитическое решение модели имеет вид:

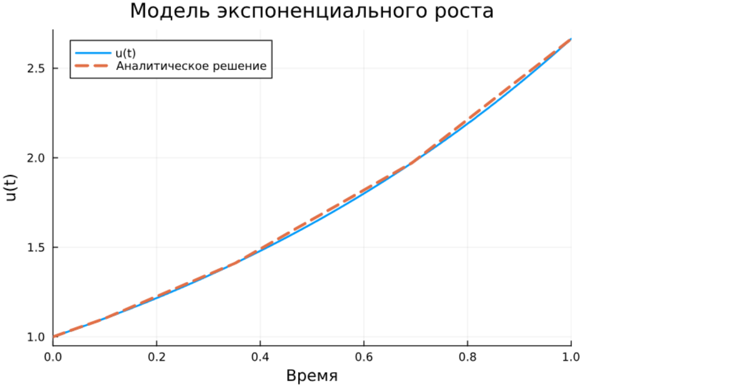
$$u(t) = u_0 exp(at)u(t).$$

```julia
# подключаем необходимые пакеты:
import Pkg
Pkg.add("DifferentialEquations")
using DifferentialEquations
# 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)
```

Рис. 1: Модель экпоненциального роста. Часть 1

```
# подключаем необходимые пакеты:
Pkg.add("Plots")
using Plots
# строим графики:
plot(sol, linewidth=2,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")
```

```
  Resolving package versions...
  No Changes to `C:\Users\noname\.julia\environments\v1.10\Project.toml`
  No Changes to `C:\Users\noname\.julia\environments\v1.10\Manifest.toml`
```



Самостоятельная работа

Задание 1

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

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

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).

```
#x'(t)=ax(t)
#a=b-c
# задаём описание модели с начальными условиями:
b = 1.36 # коэф.рождаемости
c = 0.98 # коэф.смертности
a = b - c
f(x,p,t) = a*x
x0 = 1.0
# задаём интервал времени:
tspan = (0.0,100.0)
# решение:
prob = ODEProblem(f,x0,tspan)
sol = solve(prob)
# строим графики:
plot(sol, linewidth=2,title="Модель Мальтуса", xaxis="Время",yaxis="x(t)",label="x(t)")
```

Рис. 3: Задание 1

Самостоятельная работа

```
import Pkg
Pkg.add("Distributions")
using Distributions
import Pkg
Pkg.add("Statistics")
```

```
   Resolving package versions...
  No Changes to `C:\Users\noname\.julia\environments\v1.10\Project.toml`
  No Changes to `C:\Users\noname\.julia\environments\v1.10\Manifest.toml`
   Resolving package versions...
  No Changes to `C:\Users\noname\.julia\environments\v1.10\Project.toml`
  No Changes to `C:\Users\noname\.julia\environments\v1.10\Manifest.toml`
```

```
task1_gif = @animate for i in 1:100
    tspan = (0,i)
    prob_1 = ODEProblem(f,x0,tspan)
    sol_1 = solve(prob_1)
    plot(xlim=[0,100], ylim=[0,1*10^16], xticks=(0:20:100), legend=false, title="Модель Мальтуса")
    plot!(sol_1)
end
gif(task1_gif, "Модель Мальтуса.gif", fps = 10)
```

```
[ Info: Saved animation to C:\Users\noname\Модель Мальтуса.gif
```

Задание 2

Реализовать и проанализировать логистическую модель роста популяции, заданную уравнением:

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

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).

```
# задаём описание модели с начальными условиями:
r = 1.12
k = 100
f(x,p,t) = r*x*(1-x/k)
x0 = 1.0
# задаём интервал времени:
tspan = (0.0,10.0)
# решение:
prob = ODEProblem(f,x0,tspan)
sol = solve(prob)
# строим графики:
plot(sol, linewidth=2,title="Логистическая модель роста попул", xaxis="Время",yaxis="x(t)",label="x(t)")
```

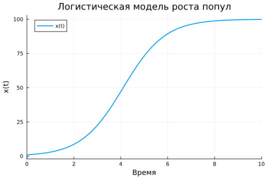
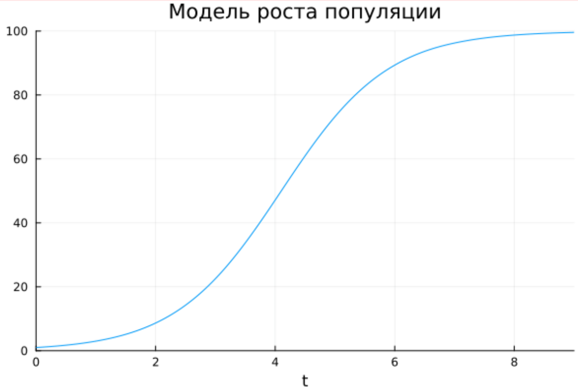


Рис. 5: Задание 2

Самостоятельная работа

```
task2_gif = @animate for i in 1:10
    tspan = (0,i)
    prob = ODEProblem(f,x0,tspan)
    sol = solve(prob)
    plot(xlim=[0,10], ylim=[0,100], xticks=(0:2:10), legend=false, title="Модель роста популяции")
    plot!(sol)
end
gif(task2_gif, "Задание 2.gif", fps = 10)
```

```
[ Info: Saved animation to C:\Users\noname\Задание 2.gif
```



Самостоятельная работа

Задание 3

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

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

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, 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).

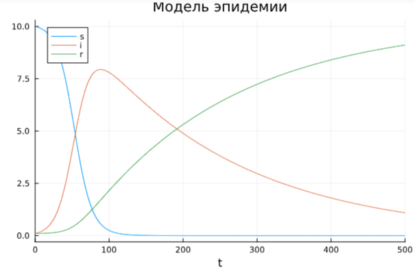
```
#using ParameterizedFunctions, DifferentialEquations, Plots;
task3! = @ode_def KerMak begin
    ds = -β*i*s
    di = β*i*s-υ*i
    dr = υ*i
end β υ
# задаём начальное условие:
u0 = [10.0,0.1,0.1]
# задаём значения параметров:
p = (0.009,0.005)
# задаём интервал времени:
tspan = (0.0,500)
# решение:
prob = ODEProblem(task3!,u0,tspan,p)
sol = solve(prob)
plot(sol, title = "Модель эпидемии" )
```

```
┌ Warning: Independent variable t should be defined with @independent_variables t.
└ @ ModelingToolkit C:\Users\noname\.julia\packages\ModelingToolkit\NQQXr\src\utils.jl:119
```

Модель эпидемии

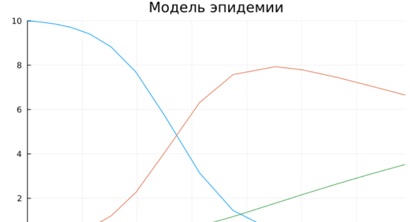
Рис. 7: Задание 3

Самостоятельная работа




```
task3_gif = @animate for i in 1:29
    plot(xlim=(0,500), ylim=(0,10), legend=false, title = "Модель эпидемии" )
    plot!(sol[1:i])
end
gif(task3_gif,"Задание 3.gif", fps=6)
```

```
[ Info: Saved animation to C:\Users\noname\Задание 3.gif
```



Задание 4

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

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

Размер популяции сохраняется:

$$s(t) + e(t) + i(t) + r(t) = N$$

Исследуйте, сравните с SIR.

```
N = 1000
p = [0.25,0.5,0.1]
u0 = [980,10,10,0.1]
tspan = (0.0,300.0)
task4 = @ode_def SEIR begin
    ds = -β/N*s*i
    de = β/N*s*i - δ*e  # болеют но не заразны, инкубационный период
    di = δ*e - γ*i
    dr = γ*i
end β δ γ
prob = ODEProblem(task4,u0,tspan,p)
sol = solve(prob)
plot(sol, title = "Модель эпидемии испанки" )
```

```
┌ Warning: Independent variable t should be defined with @independent_variables t.
└ @ ModelingToolkit C:\Users\noname\.julia\packages\ModelingToolkit\NQQXr\src\utils.jl:119
```

Рис. 9: Задание 4

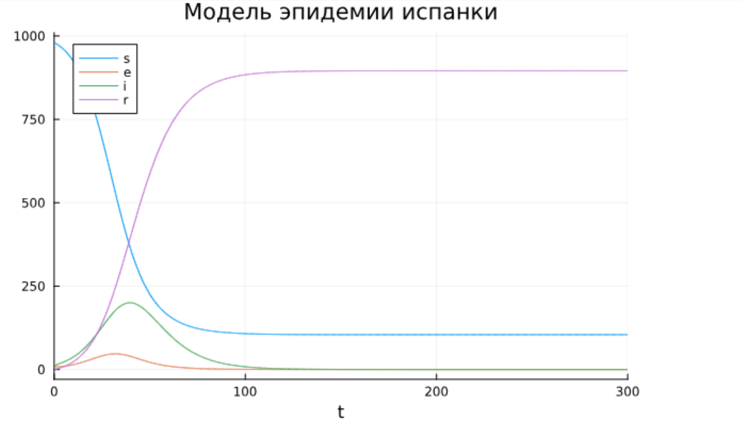


Рис. 10: Задание 4. Продолжение

Самостоятельная работа

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 $a = 2$, $c = 1$, $d = 5$ найдите точку равновесия. Получите и сравните аналитическое и численное решения. Численное решение изобразите на фазовом портрете

```
function task5(x0, p, t)
    a, b, c, d = p #parameters
    x1 = [x0[1]] #vector
    x2 = [x0[2]]
    for i=2:t
        res_x1 = a*x1[end]*(1-x1[end])-b*x1[end]*x2[end] #x1(t+1)
        append!(x1, res_x1)
        res_x2 = -c*x2[end]+d*x1[end-1]*x2[end] #x2(t+1)
        append!(x2, res_x2)
    end
    return [x1, x2]
end

#задаем параметры, по условию b=1, остальное на вкус
p = [2, 1, 1, 1.9]
x0 = [0.2, 2]
tspan = 5

sol = task5(x0, p, tspan)
plot(sol, title = "Модель Лотки - Вольтерра" )
```

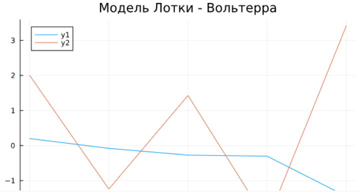


Рис. 11: Задание 5

Самостоятельная работа

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.

```
function task6(du, u, p, t)
    α,β = p
    du[1] = α*u[1]-β*u[1]*u[2]
    du[2] = α*u[2]-β*u[1]*u[2]
end
p = [0.1, 0.2]
u0 = [25, 50]
tspan = 5
prob = ODEProblem(task6, u0, tspan, p)
sol = solve(prob)
```

```
retcode: Success •••
```

```
plot(sol, title = "Модель отбора на основе конкурентных отношений" )
```

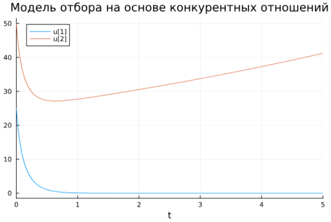


Рис. 12: Задание 6

Самостоятельная работа

Задание 7

Реализовать на языке Julia модель консервативного гармонического осциллятора

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

где ω_0 — циклическая частота. Начальные параметры подобрать самостоятельно, выбор пояснить. Построить соответствующие графики (в том числе с анимацией) и фазовый портрет

```
#ddx = -w^2*x
#dx = y
#x=x0
#we should use second order ode problem for ''
function task7(ddx, dx, x, ω, t)
    ddx .= - ω^2 * x
end
```

```
task7 (generic function with 1 method)
```

```
x0 = [0.0]
dx0 = [pi/4]
ω = 2.5
tspan = (0.0, 2pi)
prob = SecondOrderODEProblem(task7, dx0, x0, tspan, ω)
sol=solve(prob)
plot(sol, title="Модель консервативного гармонического осциллятора")
```

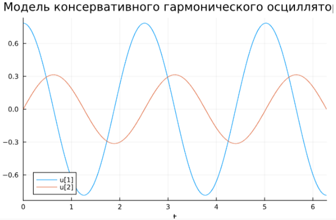


Рис. 13: Задание 7

Самостоятельная работа

Задание 8

Реализовать на языке Julia модель свободных колебаний гармонического осциллятора

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

где ω_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.

```
#ddx = -w^2*x
#dx = y
#x=x0
#we should use second order ode problem for ''
function task8(ddx, dx, x, p, t)
    γ, ω = p
    ddx .= - ω^2 * x .- 2 * γ *dx
end
```

```
task8 (generic function with 1 method)
```

```
x0 = [0.0]
dx0 = [pi/4]
ω = [1/3,10]
tspan = (0.0, 20)
prob = SecondOrderODEProblem(task8, dx0, x0, tspan, ω)
sol=solve(prob)
plot(sol, title="Свободные колебания гармонического осциллятора")
```

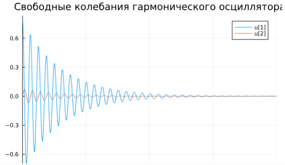


Рис. 14: Задание 8

```
plot(sol, vars=(1,2), title="Фазовый портрет модели свободного колебания" )
```

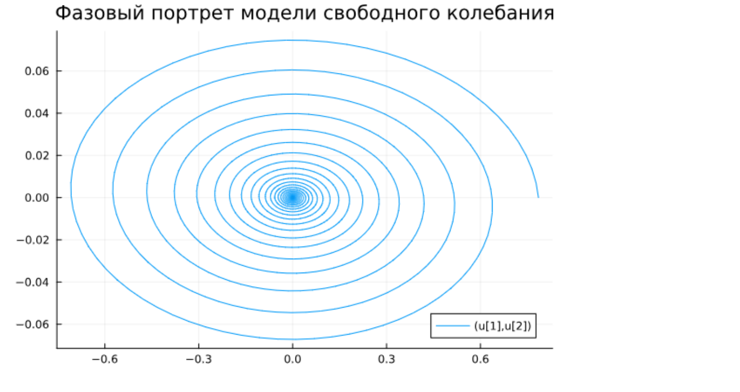


Рис. 15: Задание 8. Продолжение

Выводы по проделанной работе

Выводы по проделанной работе

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