Лабораторная работа № 6

Решение моделей в непрерывном и дискретном времени

Сунгурова М.М.

21 декабря 2024

Российский университет дружбы народов, Москва, Россия

Информация

Докладчик

- Сунгурова Мариян Мухсиновна
- Студентка группы НКНбд-01-21
- Российский университет дружбы народов

Вводная часть

Цель работы

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

Задание

1. Используя JupyterLab, повторите примерыи. При этом дополните графики обозначениями осей координат, легендой с названиями траекторий, названиями графиков и т.п.

2. Выполните задания для самостоятельной работы.

Julia – высокоуровневый свободный язык программирования с динамической типизацией, созданный для математических вычислений . Эффективен также и для написания программ общего назначения. Синтаксис языка схож с синтаксисом других математических языков, однако имеет некоторые существенные отличия.

Для выполнения заданий была использована официальная документация Julia .

Выполним примеры из лабораторной работы для знакомства с работой с различными моделями и способами их задания решения.

Модель экспоненциального роста

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

[1] ✓ 1m 48.8s

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

```
# 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)
```

[2] ✓ 1.8s

```
(0.0, 1.0)
```

```
prob = ODEProblem(f,u0,tspan)
sol = solve(prob)
```

```
# подключаем необходимые пакеты:
Pkg.add("Plots")
using Plots
```

✓ 27.1s

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

```
# 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")
```

✓ 8.9s

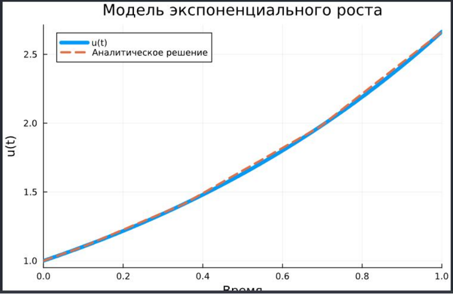


Рис. 2: Модель экспоненциального роста

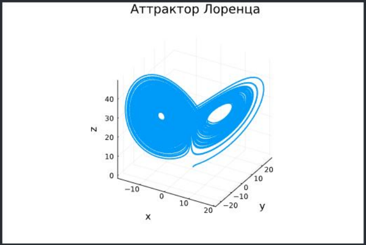
Рис. 3: Модель экспоненциального роста

Система Лоренца

```
# 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)
# строим график:
plot(sol, vars=(1,2,3), lw=2, title="Аттрактор Лоренца", xaxis="x",yaxis="y", zaxis="z",legend=false)
```

✓ 5.5s

```
┌ Warning: To maintain consistency with solution indexing, keyword argument vars will be removed in a future ver
│   caller = ip:0x0
└ @ Core :-1
```



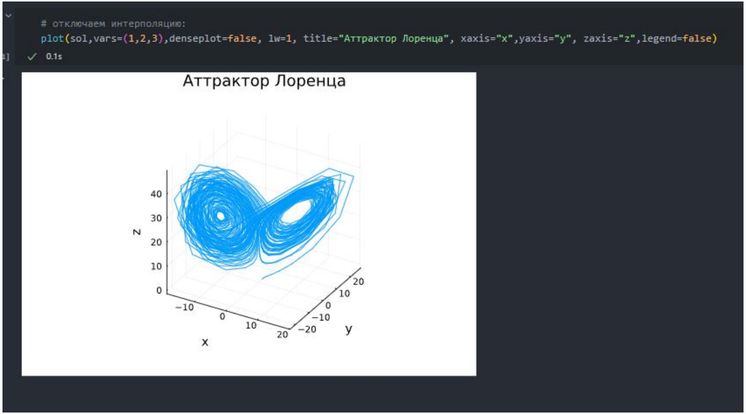
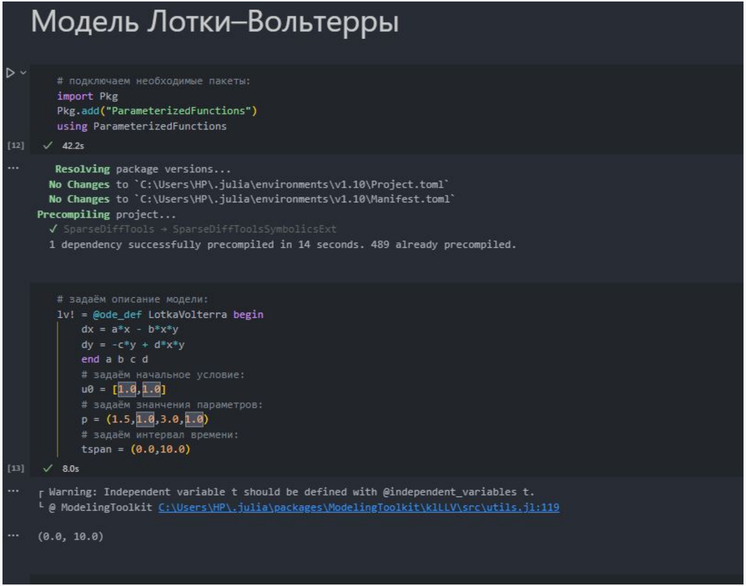


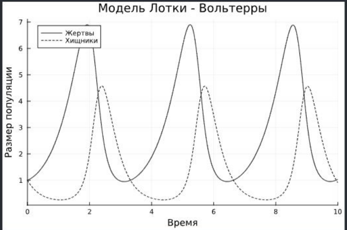
Рис. 5: Система Лоренца


Модель Лотки–Вольтерры
# подключаем необходимые пакеты:
import Pkg
Pkg.add("ParameterizedFunctions")
using ParameterizedFunctions
[12] ✓ 42.2s
Resolving package versions...
No Changes to `C:\Users\HP\.julia\environments\v1.10\Project.toml`
No Changes to `C:\Users\HP\.julia\environments\v1.10\Manifest.toml`
Precompiling project...
✓ SparseDiffTools → SparseDiffToolsSymbolicsExt
1 dependency successfully precompiled in 14 seconds. 489 already precompiled.
задаём описание модели:
lv! = @ode_def LotkaVolterra begin
dx = a*x - b*x*y
dy = -c*y + d*x*y
end a b c d
# задаём начальное условие:
u0 = [1.0,1.0]
# задаём знанчения параметров:
p = (1.5,1.0,3.0,1.0)
# задаём интервал времени:
tspan = (0.0,10.0)
[13] ✓ 8.0s
┌ Warning: Independent variable t should be defined with @independent_variables t.
└ @ ModelingToolkit C:\Users\HP\.julia\packages\ModelingToolkit\kILLV\src\utils.jl:119
(0.0, 10.0)

Выполнение лабораторной работы

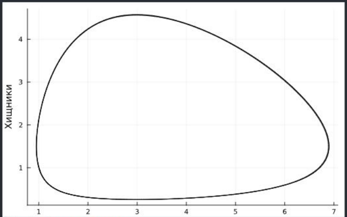
```
# решение:
prob = ODEProblem(lv!,u0,tspan,p)
sol = solve(prob)
plot(sol, label = ["Жертвы" "Хищники"], color="black", ls=[:solid :dash], title="Модель Лотки - Вольтерры",xaxis="Время",yaxis="Размер популяции")
```

[15] ✓ 4.8s




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

[16] ✓ 0.3s



Далее перейдем к заданиям для самостоятельного выполнения.

В первом задании реализуем и проанализируем 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. Построим соответствующие графики (в том числе с анимацией).

Задание 1

```
f(u, p, t) = a*u
b = 0.05
c = 0.02
a = b-c
tspan = (0.0, 100.0)
u0 = 1
prob = ODEProblem(f, u0, tspan)
solution = solve(prob)
```

[20] ✓ 5.3s

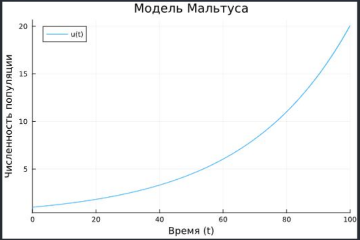
```
retcode: Success
Interpolation: 3rd order Hermite
t: 10-element Vector{Float64}:
   0.0
   0.20167987549068
   2.2184786303974797
   9.274487404424077
  19.850074620925533
  32.33554873981471
  47.755459484394564
  65.4998284624331
  85.71392255948412
 100.0
u: 10-element Vector{Float64}:
  1.0
  1.006068736882879
  1.068819062338894
  1.3207960488641517
  1.8139417179370756
  2.638119571822432
  4.189843442807894
  7.134866254850057
 13.084312263071691
 20.085458809094543
```

Коэффициенты модели Мальтуса обычно включают:

x – численность популяции;
t – время;
r = □ □ β, где □ – коэффициент рождаемости, β – коэффициент смертности;

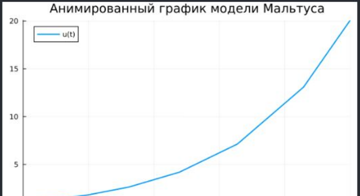
```
plot(solution, label="u(t)", xlabel="Время (t)", ylabel = "Численность популяции", title="Модель Мальтуса
```

[21] ✓ 0.3s




```
animated = @animate for t in 0.0:0.5:100.0
    plot(solution.t[solution.t .<=t], solution.u[solution.t .<= t],
    label="u(t)", title="Анимированный график модели Мальтуса", xlabel="Время",
    xlim=(0, 100), ylim=(0, maximum(solution.u)), lw=2)
end

gif(animated, "C:\\Uni\\StatsComp\\Lab2\\animation1.gif", fps=15)
```

[26] ✓ 7.0s



Далее во втором задании реализуем и проанализируем логистическую модель роста популяции:

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

где r – коэффициент роста популяции, k – потенциальная ёмкость экологической системы (предельное значение численности популяции). Построим соответствующие графики (в том числе с анимацией) (рис. (**fig:010?**)-(**fig:011?**)).

Задание 2

```
f(u,p,t)=r*u*(1-u/k)
r = 0.5
k = 1000
u0 = 150
tspan = (0.0, 100.0)
prob = ODEProblem(f, u0, tspan)
solution = solve(prob)
plot(solution, xlabel="Время (t)", ylabel = "Численность популяции", legend=false, title="Логистическая модель роста популяции")
```

[92] ✓ 2.6s

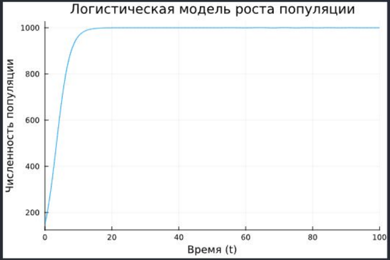


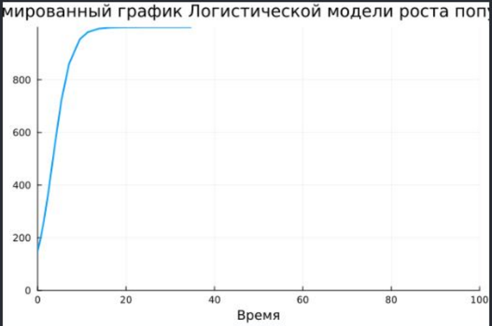
Рис. 10: Логистическая модель роста популяции

Анимация:

```
animated = @animate for t in 0.0:0.5:100.0
    plot(solution.t[solution.t .<=t], solution.u[solution.t .<= t],
    legend=false, title="Анимированный график Логистической модели роста популяции", xlabel="Время",
    xlim=(0, 100), ylim=(0, maximum(solution.u)), lw=2)
end

gif(animated, "C:\\Uni\\StatsComp\\Lab2\\animation2.gif", fps=15)
```

[30] ✓ 5.7s

```
┌ Info: Saved animation to C:\Uni\StatsComp\Lab2\animation2.gif
└ @ Plots C:\Users\HP\.julia\packages\Plots\Ec1L1\src\animation.jl:156
```



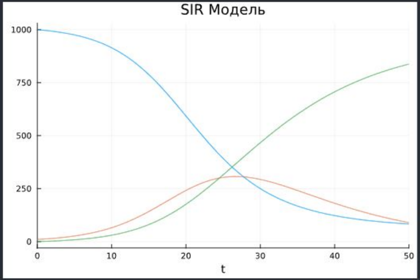
В заданиии номер 3 реализуем и проанализируем логистическую модель эпидемии Кермака–Маккендрика (SIR-модель):

$$\begin{cases} \dot{S} = -\beta I S, \\ \dot{I} = \beta I S - \gamma I, \\ \dot{R} = \gamma I, \end{cases}$$

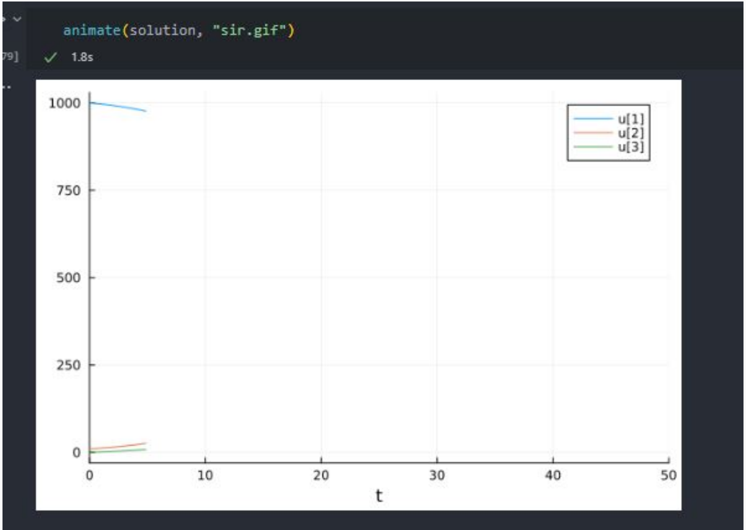
где S – численность восприимчивой популяции, I – численность инфицированных, R – численность удаленной популяции (в результате смерти или выздоровления), и N — это сумма этих трёх, а β и γ - это коэффициенты заболеваемости и выздоровления соответственно

```
# SIR -model
function SIRmodel(u, p, t)
    (S, I, R) = u
    (b, y) = p
    N = S + I + R
    dS = -(b*I*S)/N
    dI = (b*I*S)/N - y*I
    dR = y*I
    return [dS, dI, dR]
end

ddt = 0.1
tm = 50.0
tspan = (0.0, 50.0)
u0 = [1000.0, 10.0, 0.0]
p = [0.3, 0.1]
prob = ODEProblem(SIRmodel, u0, tspan, p)
solution = solve(prob, dt = ddt )

plot(solution, label = ["S", "I", "R"], legend=false, title="SIR Модель")
```

✓ 2.9s



Также была реализована и анимация



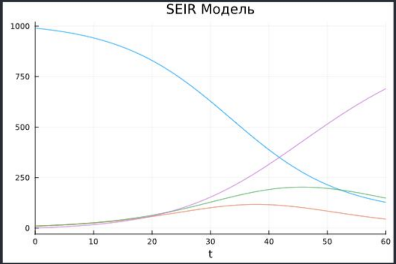
Далее в рамках четвертого задания как расширение модели SIR (Susceptible-Infected-Removed) по результатам эпидемии испанки была предложена модель SEIR (Susceptible-Exposed-Infected-Removed) (рис. (**fig:009?**)).

$$
\begin{cases}
\dot{S} = -\dfrac{\beta}{N} I S, \\
\dot{E} = \dfrac{\beta}{N} I S - \delta E, \\
\dot{I} = \delta E - \gamma I, \\
\dot{R} = \gamma I,
\end{cases}
$$

Задание 4

```
#SEIR
function SEIRmodel(u,p,t)
    (S,E,I,R) = u
    (b, y, g) = p
    N = S + E + I + R
    dS = -(b*I*S)/N
    dE = (b*S*I)/N - g*E
    dI = g*E - y*I
    dR = y*I
    return [dS, dE, dI, dR]
end

ddt =0.1
tspan = (0.0, 60.0)
u0 = [990.0, 10.0, 10.0, 0.0]
p = [0.3, 0.1, 0.2]
prob = ODEProblem(SEIRmodel, u0, tspan, p)
solution = solve(prob, dt=ddt)
plot(solution, label = ["S","E",  "I", "R"], legend=false, title="SEIR Модель")
```

[76] ✓ 3.8s



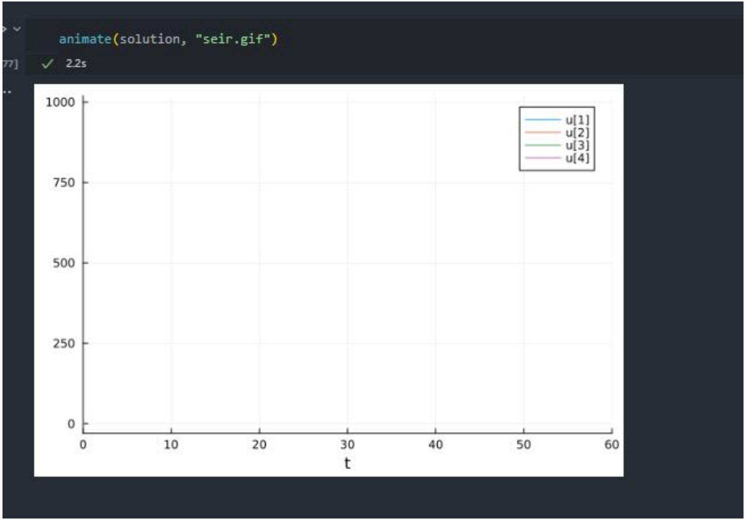


Рис. 15: SEIR-модель

В задании номер 5 для дискретной модели Лотки–Вольтерры:

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

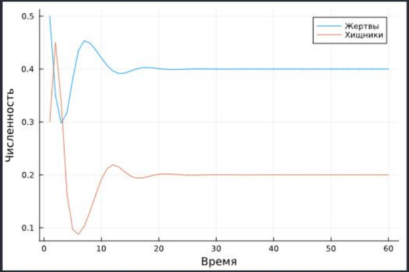
с начальными данными $a = 2, c = 1, d = 5$ найдем точку равновесия. Получим и сравним аналитическое и численное решения.

Выполнение лабораторной работы

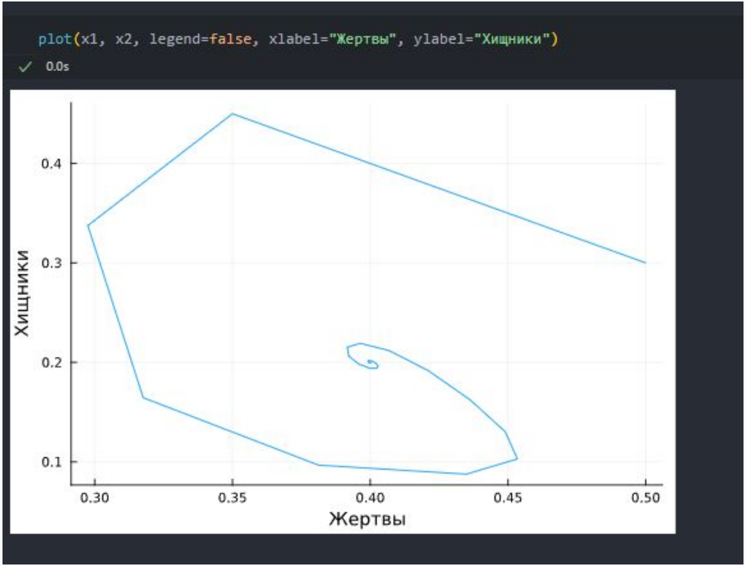
Дискретная модель Лотки-Вольтерры

```
a, c, d = 2, 1, 5
tm = 60

x10 = 0.5
x20 = 0.3

x1 = zeros(tm)
x2 = zeros(tm)
x1[1] = x10
x2[1] = x20

for t in 1:tm-1
    x1[t+1] = a*x1[t]*(1-x1[t]) - x1[t]*x2[t]
    x2[t+1] = -c*x2[t] + d*x1[t]*x2[t]
end

plot(1:tm, x1, label = "Жертвы")
plot!(1:tm, x2, label="Хищники", xlabel="Время", ylabel="Численность")
```

[74] ✓ 0.0s



Выполнение лабораторной работы

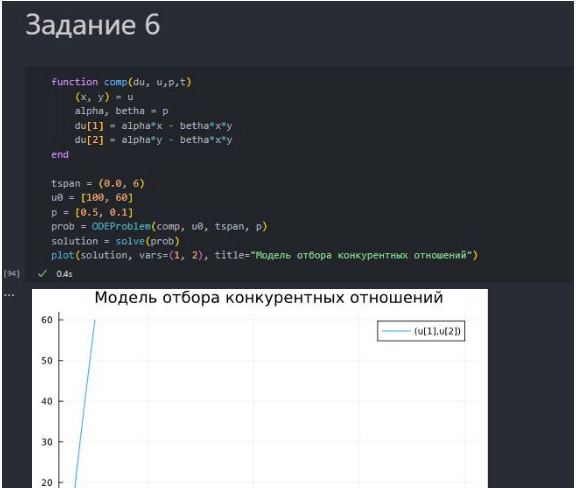


В задании 6 Реализуем на языке Julia модель отбора на основе конкурентных отношений:

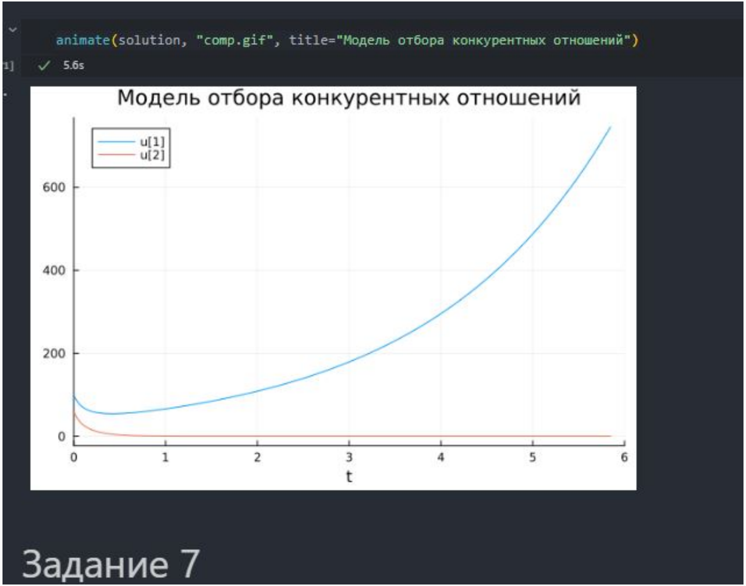
$$\begin{cases} \dot{x} = \alpha x - \beta x y, \\ \dot{y} = \alpha y - \beta x y, \end{cases}$$

Выполнение лабораторной работы

Построим соответствующие графики (в том числе с анимацией) и фазовый портрет (рис. (fig:018?)-(fig:019?)).



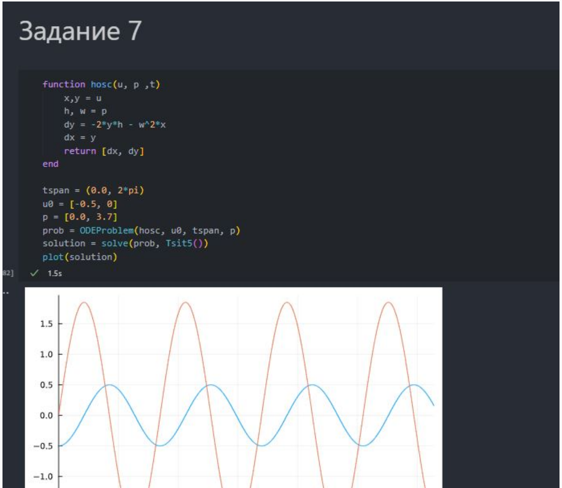
Выполнение лабораторной работы



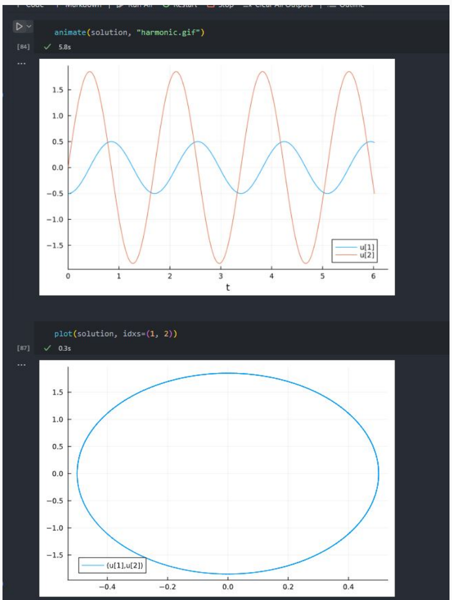
Реализуем на языке Julia модель консервативного гармонического осциллятора:

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

В задании 7 построим соответствующие графики (в том числе с анимацией) и фазовый портрет (рис. (fig:013?)-(fig:014?)).



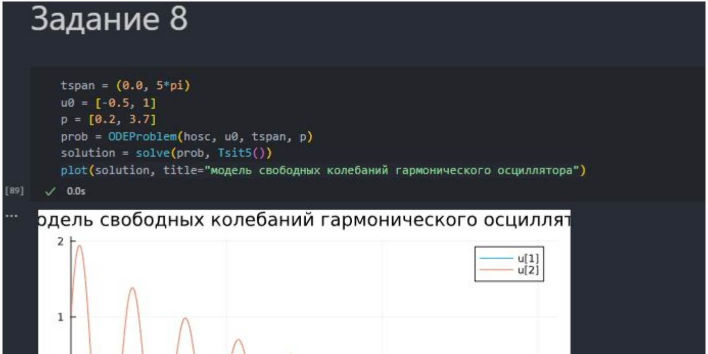
Выполнение лабораторной работы



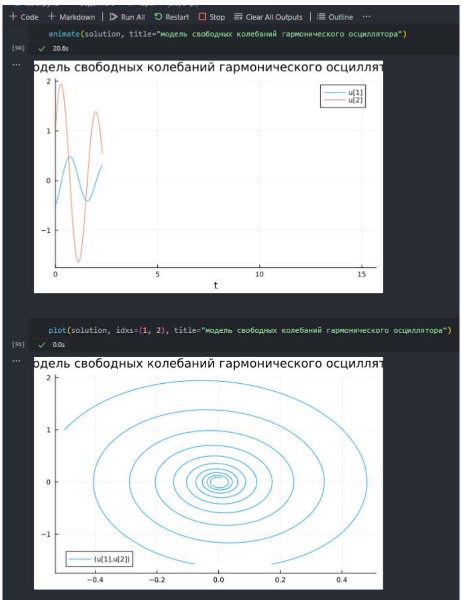
Реализуем на языке Julia модель свободных колебаний гармонического осциллятора:

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

Построим соответствующие графики (в том числе с анимацией) и фазовый портрет .



Выполнение лабораторной работы



Выводы

Выводы

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