

Лабораторная работа №4

Модель гармонических колебаний

Кузнецов Юрий Владимирович

Содержание

Цель работы	1
Задачи.....	1
Среда	2
Теоретическое введение	2
Ход работы	3
Вывод.....	18
Ресурсы.....	18

Цель работы

Рассмотреть уравнение гармонических колебаний. Построить модель гармонических колебаний при помощи OpenModelica и Julia.

Задачи

Построить фазовый портрет гармонического осциллятора и решение уравнения гармонического осциллятора для следующих случаев

1. Колебания гармонического осциллятора без затуханий и без действий внешней силы $\ddot{x} + 3.3x = 0$.
2. Колебания гармонического осциллятора с затуханием и без действий внешней силы $\ddot{x} + 3\dot{x} + 0.3x = 0$.
3. Колебания гармонического осциллятора с затуханием и под действием внешней силы $\ddot{x} + 3.3\dot{x} + 0.3x = 3.3\sin(3t)$.

На интервале $t \in [0; 33]$ (шаг 0.05) с начальными условиями $x_0 = 1.3$, $y_0 = 0.3$.

Среда

- Julia – это открытый свободный высокопроизводительный динамический язык высокого уровня, созданный специально для технических (математических) вычислений. Его синтаксис близок к синтаксису других сред технических вычислений, таких как Matlab и Octave. [@unn-julia]
- OpenModelica — свободное открытое программное обеспечение для моделирования, симуляции, оптимизации и анализа сложных динамических систем. Основано на языке Modelica. [@wiki-om]

Теоретическое введение

Движение грузика на пружинке, маятника, заряда в электрическом контуре, а также эволюция во времени многих систем в физике, химии, биологии и других науках при определенных предположениях можно описать одним и тем же дифференциальным уравнением, которое в теории колебаний выступает в качестве основной модели. [@rudn-task]

Эта модель называется линейным гармоническим осциллятором. Уравнение свободных колебаний гармонического осциллятора имеет следующий вид:

$$\ddot{x} + 2\gamma\dot{x} + \omega_0^2 x = 0$$

где x – переменная, описывающая состояние системы (смещение грузика, заряд конденсатора и т.д.), γ – параметр, характеризующий потери энергии (трение в механической системе, сопротивление в контуре), ω_0 – собственная частота колебаний, t – время. (Обозначения $\ddot{x} = \frac{\partial^2 x}{\partial t^2}$, $\dot{x} = \frac{\partial x}{\partial t}$)

При отсутствии потерь в системе вместо вышеуказанного уравнения получаем уравнение консервативного осциллятора энергия колебания которого сохраняется во времени

$$\ddot{x} + \omega_0^2 x = 0$$

Для однозначной разрешимости уравнения второго порядка необходимо задать два начальных условия вида

$$\begin{cases} x(t_0) = x_0 \\ \dot{x}(t_0) = y_0 \end{cases}$$

Уравнение консервативного осциллятора энергия колебания которого сохраняется во времени можно представить в виде системы двух уравнений первого порядка:

$$\begin{cases} \dot{x} = y \\ \dot{y} = -\omega_0^2 x \end{cases}$$

Начальные условия для системы примут вид:

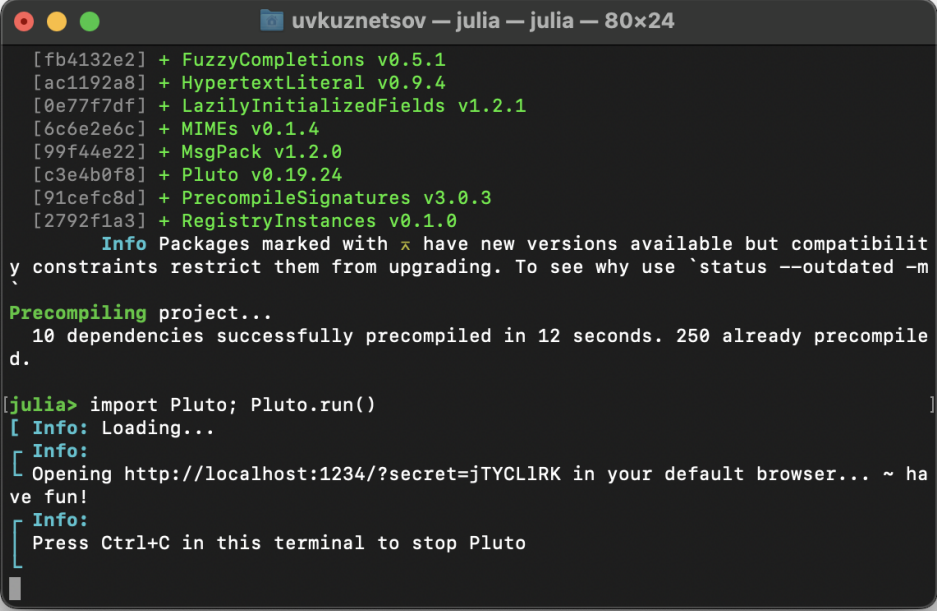
$$\begin{cases} x(t_0) = x_0 \\ y(t_0) = y_0 \end{cases}$$

Независимые переменные x, y определяют пространство, в котором «движется» решение. Это фазовое пространство системы, поскольку оно двумерно будем называть его фазовой плоскостью.

Значение фазовых координат x, y в любой момент времени полностью определяет состояние системы. Решению уравнения движения как функции времени отвечает гладкая кривая в фазовой плоскости. Она называется фазовой траекторией. Если множество различных решений (соответствующих различным начальным условиям) изобразить на одной фазовой плоскости, возникает общая картина поведения системы. Такую картину, образованную набором фазовых траекторий, называют фазовым портретом.

Ход работы

Запустим Pluto для выполнения первых задач.



```
uvkuznetsov — julia — julia — 80x24
[fb4132e2] + FuzzyCompletions v0.5.1
[ac1192a8] + HypertextLiteral v0.9.4
[0e77f7df] + LazilyInitializedFields v1.2.1
[6c6e2e6c] + MIMES v0.1.4
[99f44e22] + MsgPack v1.2.0
[c3e4b0f8] + Pluto v0.19.24
[91cefc8d] + PrecompileSignatures v3.0.3
[2792f1a3] + RegistryInstances v0.1.0
Info Packages marked with * have new versions available but compatibility constraints restrict them from upgrading. To see why use `status --outdated -m`
Precompiling project...
 10 dependencies successfully precompiled in 12 seconds. 250 already precompiled.
[julia> import Pluto; Pluto.run()]
[ Info: Loading...
[ Info:
Opening http://localhost:1234/?secret=jTYCLlRK in your default browser... ~ have fun!
[ Info:
Press Ctrl+C in this terminal to stop Pluto
```

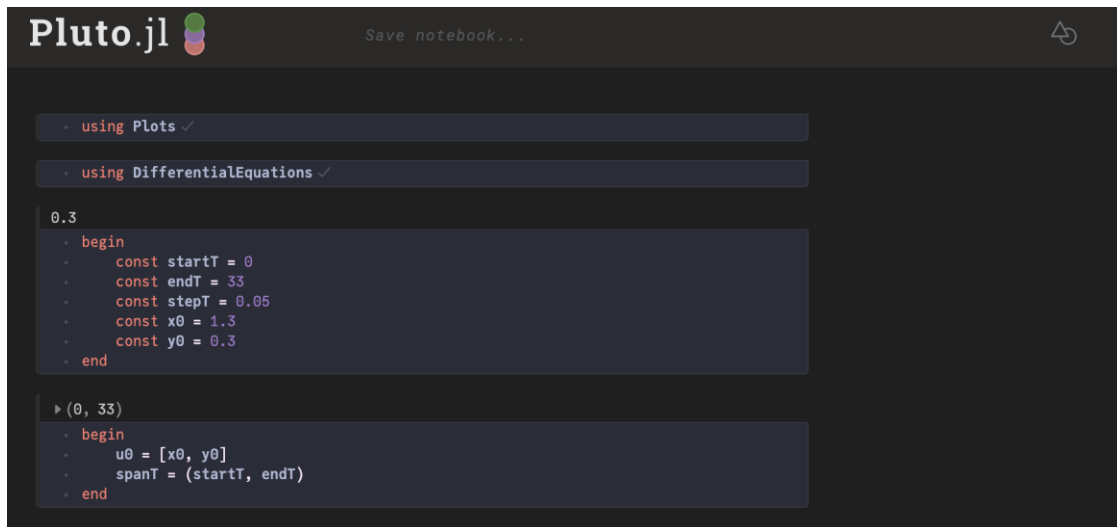
Julia. Заняк Pluto

Подключаем Plots и DifferentialEquations. Объявляем начальные данные при помощи констант. Устанавливаем начальное условие для системы ДУ и промежуток времени для визуализации.

```
using Plots
using DifferentialEquations
```

```
const startT = 0
const endT = 33
const stepT = 0.05
const x0 = 1.3
const y0 = 0.3
```

```
u0 = [x0, y0]
spanT = (startT, endT)
```



Julia. Начало написания скрипта для моделирование колебания гармонического осциллятора

В следующей ячейке Pluto построим фазовый портрет и решение уравнения гармонического осциллятора. Воспользуемся данным списком, чтобы построить фазовый портрет.

```
w = 3.3
```

```
function Fluctuations!(df, u, p, t)
  df[1] = u[2]
  df[2] = -w * u[1]
end
```

```
prob = ODEProblem(Fluctuations!, u0, spanT)
sol = solve(prob, dtmax=stepT)
```

```
X = [u[1] for u in sol.u]
```

```

Y = [u[2] for u in sol.u]

plt01 = plot(sol,
    dpi=500,
    xlabel="Время (s)",
    ylabel="x, y",
    legend=false)
savefig(plt01, "labart/result.png")

plt02 = plot(X, Y,
    dpi=500,
    xlabel="x",
    ylabel="y",
    legend=false)
savefig(plt02, "labart/result2.png")

println("Success!")

```

```

begin
    w = 3.3

    function Fluctuations!(df, u, p, t)
        df[1] = u[2]
        df[2] = -w * u[1]
    end

    prob = ODEProblem(Fluctuations!, u0, spanT)
    sol = solve(prob, dtmax=stepT)

    X = [u[1] for u in sol.u]
    Y = [u[2] for u in sol.u]

    plt01 = plot(sol,
        dpi=500,
        xlabel="Время (s)",
        ylabel="x, y",
        legend=false)
    savefig(plt01, "labart/result.png")

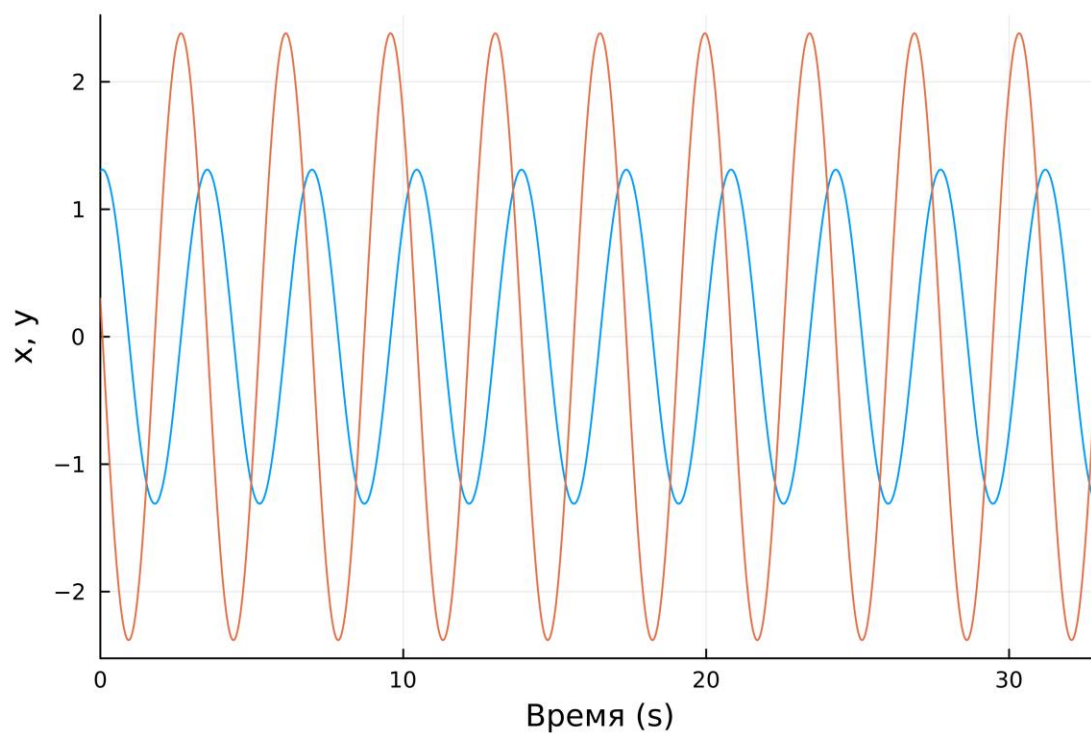
    plt02 = plot(X, Y,
        dpi=500,
        xlabel="x",
        ylabel="y",
        legend=false)
    savefig(plt02, "labart/result2.png")

    println("Success!")
end

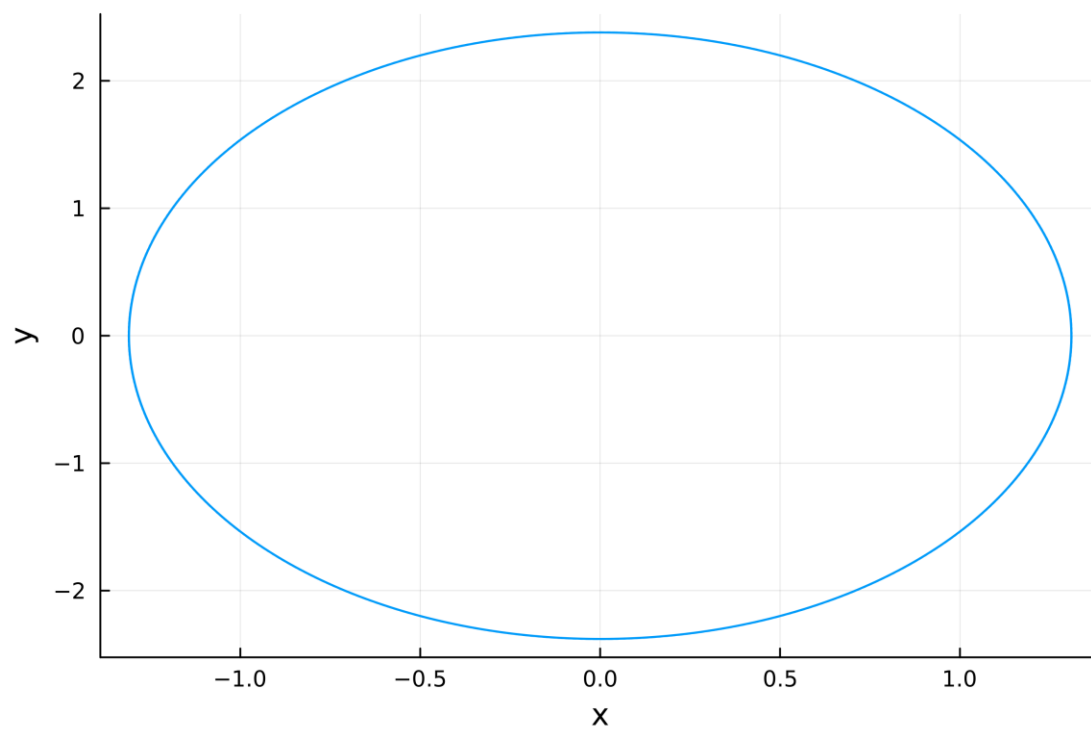
```

Success!

Julia. Скрипт. Колебания гармонического осциллятора без затуханий и без действий внешней силы



Julia. Модель. Решение уравнения гармонического осциллятора без затуханий и без действий внешней силы



Julia. Модель. Фазовый портрет осциллятора без затуханий и без действий внешней силы

Доработаем данный скрипт, чтобы построить решение уравнения и фазовый портрет гармонического осциллятора с затуханием и без действий внешней силы. Для этого нам необходимо добавить новый параметр - затухание. Также необходимо изменить функцию системы ДУ.

```
w = 0.3
g = 3

function Fluctuations!(df, u, p, t)
    df[1] = u[2]
    df[2] = -w * u[1] - g * u[2]
end

prob = ODEProblem(Fluctuations!, u0, spanT)
sol = solve(prob, dtmax=stepT)

X = [u[1] for u in sol.u]
Y = [u[2] for u in sol.u]

plt01 = plot(sol,
    dpi=500,
    xlabel="Время",
    ylabel="x, y",
    legend=false)
savefig(plt01, "labart/result_1.png")

plt02 = plot(X, Y,
    dpi=500,
    xlabel="x",
    ylabel="y",
    legend=false)
savefig(plt02, "labart/result_2.png")

println("Complete!")
```

```

begin
    w = 0.3
    g = 3

    function Fluctuations!(df, u, p, t)
        df[1] = u[2]
        df[2] = -w * u[1] - g * u[2]
    end

    prob = ODEProblem(Fluctuations!, u0, spanT)
    sol = solve(prob, dtmax=stepT)

    X = [u[1] for u in sol.u]
    Y = [u[2] for u in sol.u]

    plt01 = plot(sol,
        dpi=500,
        xlabel="Время",
        ylabel="x, y",
        legend=false)
    savefig(plt01, "labart/result_1.png")

    plt02 = plot(X, Y,
        dpi=500,
        xlabel="x",
        ylabel="y",
        legend=false)
    savefig(plt02, "labart/result_2.png")

    println("Complete!")
end

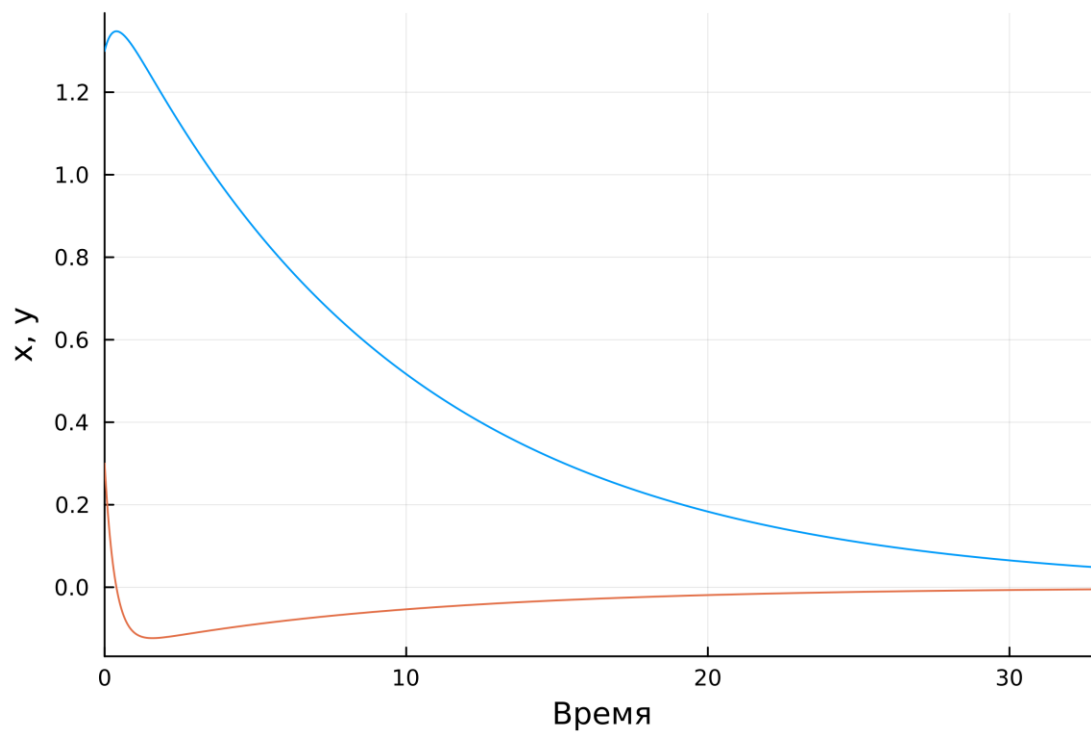
```



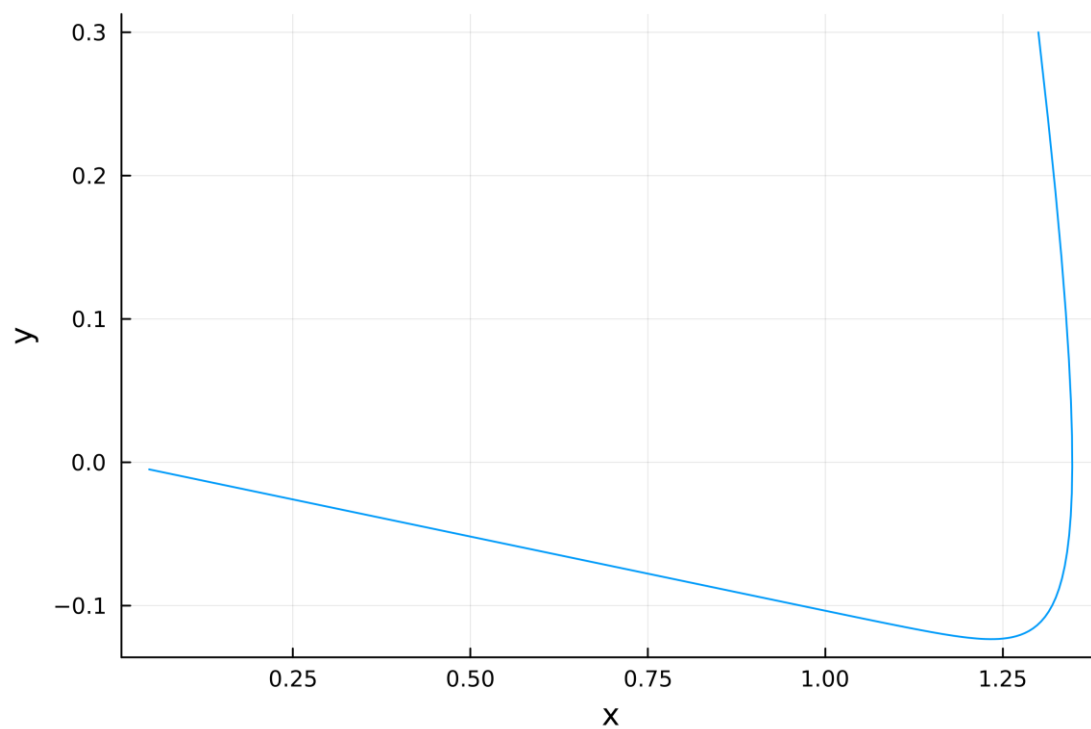
Complete!



Julia. Скрипт. Колебания гармонического осциллятора с затуханием и без действий внешней силы



Julia. Модель. Решение уравнения гармонического осциллятора с затуханием и без действий внешней силы



Julia. Модель. Фазовый портрет осциллятора с затуханием и без действий внешней силы

Еще раз доработаем скрипт, чтобы построить решение уравнения и фазовый портрет гармонического осциллятора с затуханием и под действием внешней силы. Для этого нам необходимо добавить новый параметр - функция внешней силы. Также необходимо изменить функцию системы ДУ.

```
w = 3
g = 3.3
f(t) = 3.3 * sin.(3 * t)

function Fluctuations!(df, u, p, t)
    df[1] = u[2]
    df[2] = -w * u[1] - g * u[2] - f(t)
end

prob = ODEProblem(Fluctuations!, u0, spanT)
sol = solve(prob, dtmax=stepT)

X = [u[1] for u in sol.u]
Y = [u[2] for u in sol.u]

plt01 = plot(sol,
    dpi=500,
    xlabel="Время",
    ylabel="x, y",
    legend=false)
savefig(plt01, "labart/result_1_1.png")

plt02 = plot(X, Y,
    dpi=500,
    xlabel="x",
    ylabel="y",
    legend=false)
savefig(plt02, "labart/result_1_2.png")

println("OK!")
```

```

begin
    w = 3
    g = 3.3
    f(t) = 3.3 * sin.(3 * t)

    function Fluctuations!(df, u, p, t)
        df[1] = u[2]
        df[2] = -w * u[1] - g * u[2] - f(t)
    end

    prob = ODEProblem(Fluctuations!, u0, spanT)
    sol = solve(prob, dtmax=stepT)

    X = [u[1] for u in sol.u]
    Y = [u[2] for u in sol.u]

    plt01 = plot(sol,
        dpi=500,
        xlabel="Время",
        ylabel="x, y",
        legend=false)
    savefig(plt01, "labart/result_1_1.png")

    plt02 = plot(X, Y,
        dpi=500,
        xlabel="x",
        ylabel="y",
        legend=false)
    savefig(plt02, "labart/result_1_2.png")

    println("OK!")
end

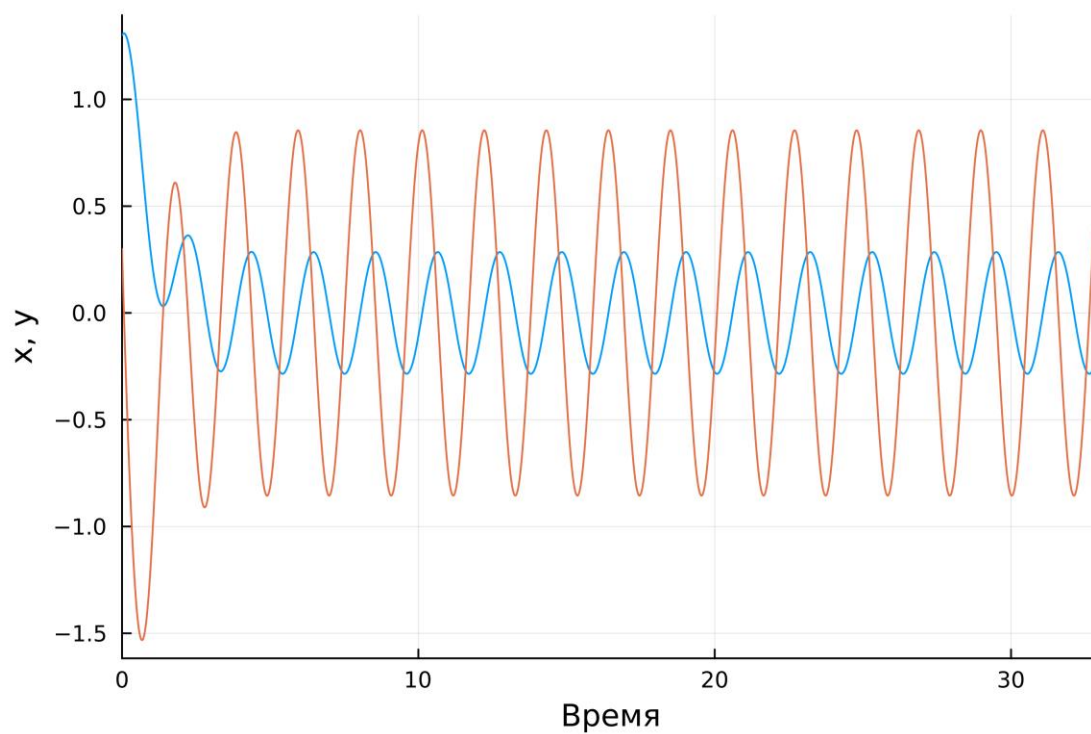
```



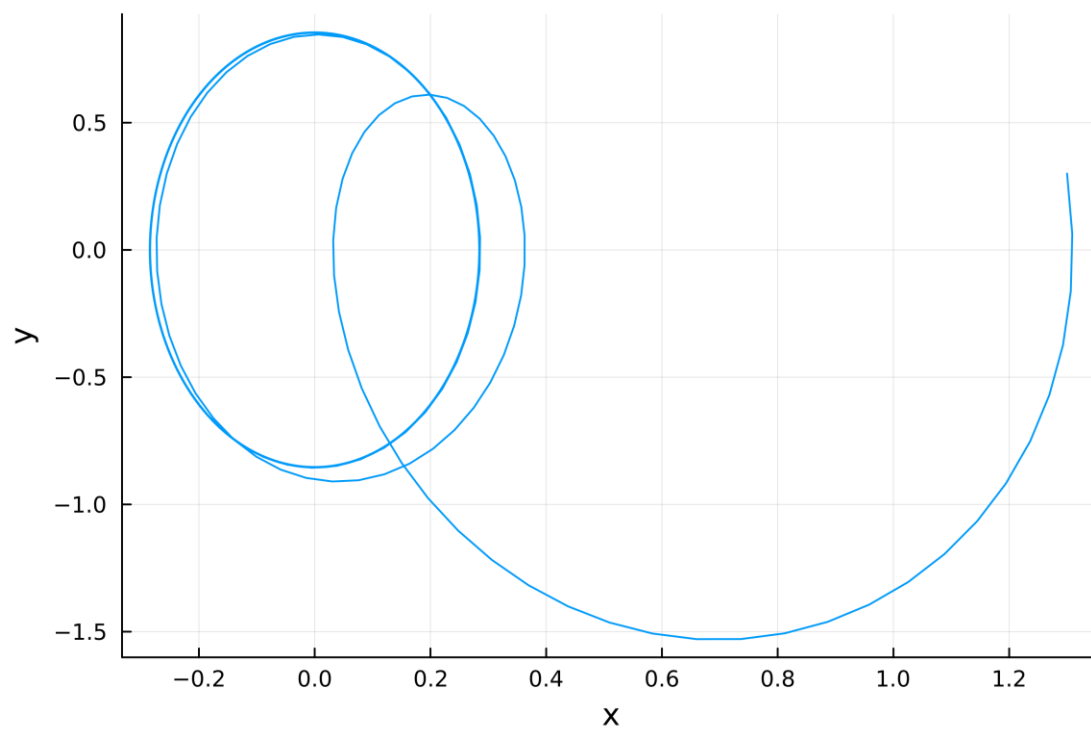
OK!



Julia. Скрипт. Колебания гармонического осциллятора с затуханием и под действием внешней силы



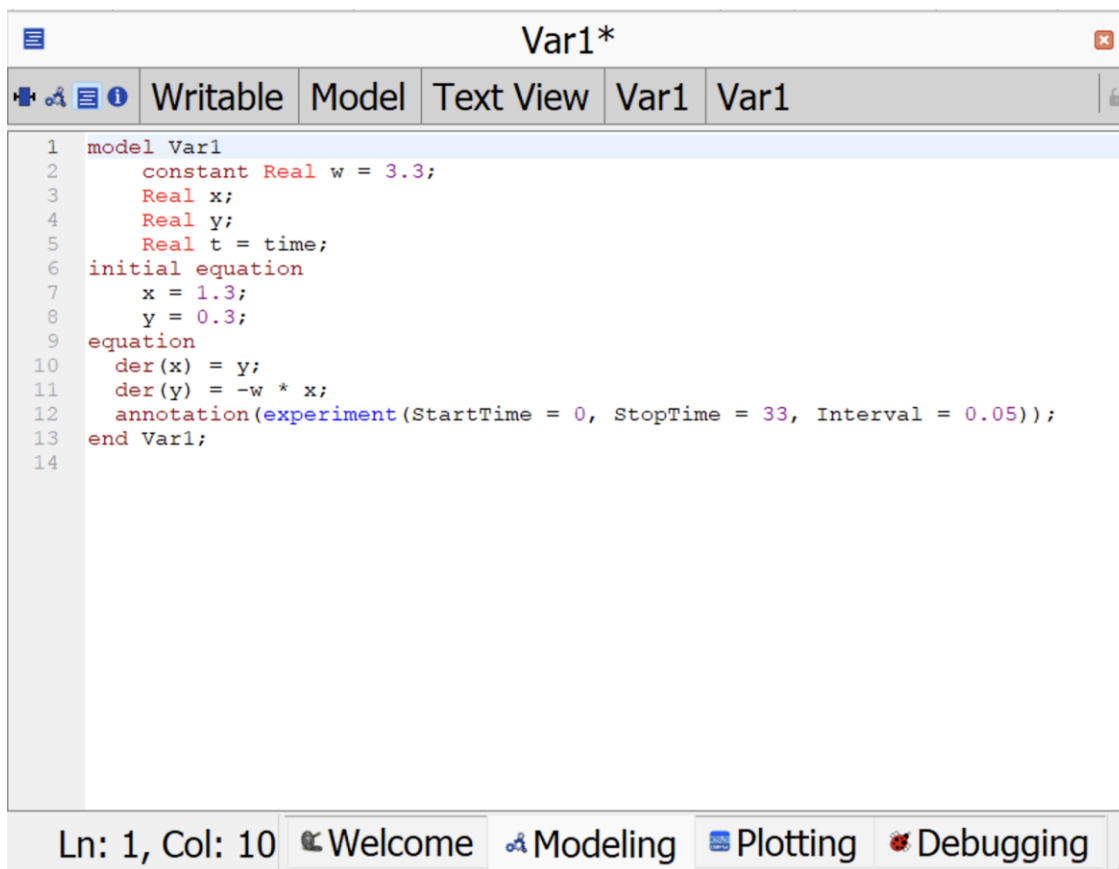
Julia. Модель. Решение уравнения гармонического осциллятора с затуханием и под действием внешней силы



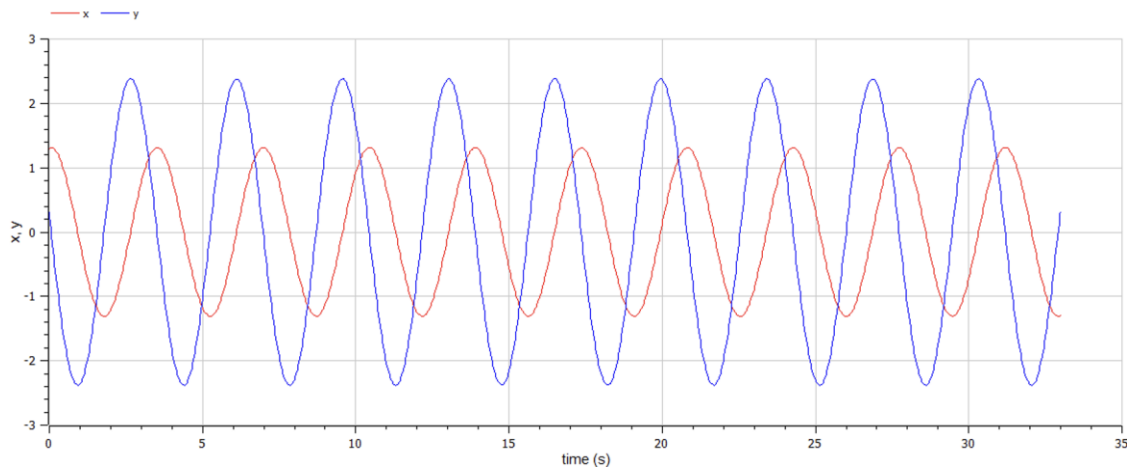
Julia. Модель. Фазовый портрет осциллятора с затуханием и под действием внешней силы

Построим модель колебания гармонического осциллятора без затуханий и без действий внешней силы на Modelica.

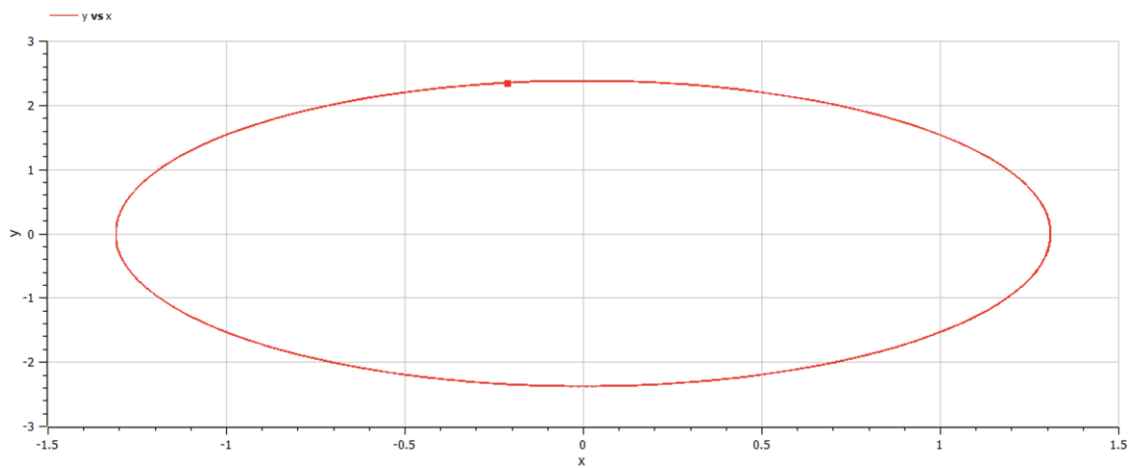
```
model Var1
  constant Real w = 3.3;
  Real x;
  Real y;
  Real t = time;
initial equation
  x = 1.3;
  y = 0.3;
equation
  der(x) = y;
  der(y) = -w * x;
  annotation(experiment(StartTime = 0, StopTime = 33, Interval = 0.05));
end Var1;
```



Modelica. Скрипт. Колебания гармонического осциллятора без затуханий и без действий внешней силы



Modelica. Модель. Решение уравнения гармонического осциллятора без затуханий и без действий внешней силы



Modelica. Модель. Фазовый портрет осциллятора без затуханий и без действий внешней силы

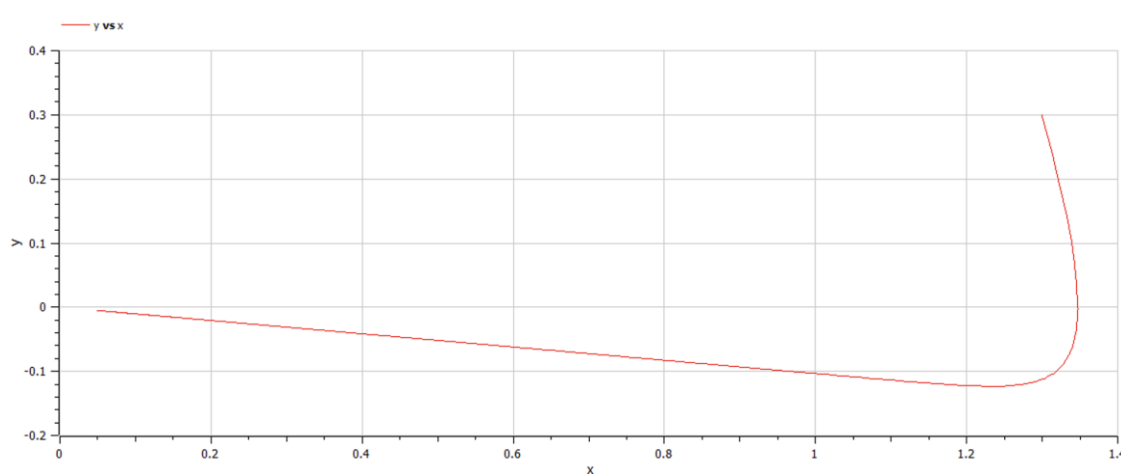
Построим модель колебания гармонического осциллятора с затуханием и без действий внешней силы на Modelica.

```
model Var2
  constant Real w = 0.3;
  constant Real g = 3;
  Real x;
  Real y;
  Real t = time;
initial equation
  x = 1.3;
  y = 0.3;
equation
  der(x) = y;
  der(y) = -w * x - g * y;
```

```
1  model Var2
2      constant Real w = 0.3;
3      constant Real g = 3;
4      Real x;
5      Real y;
6      Real t = time;
7  initial equation
8      x = 1.3;
9      y = 0.3;
10 equation
11     der(x) = y;
12     der(y) = -w * x - g * y;
13     annotation(experiment(StartTime = 0, StopTime = 33, Interval = 0.05));
14 end Var2;
```

Figure 1 is a line graph showing the time evolution of the expectation values of the Pauli matrices, x (red line) and y (blue line), as a function of time in seconds. The x-axis represents time (s) from 0 to 35, and the y-axis represents the expectation values x, y from -0.5 to 1.5. The red line starts at approximately 1.35, peaks slightly at 0.1s, and then decays towards 0. The blue line starts at approximately 0.3, drops to a minimum of about -0.1 at 1s, and then slowly rises back towards 0.

Modelica. Модель. Решение уравнения гармонического осциллятора с затуханием и без действий внешней силы



Modelica. Модель. Фазовый портрет осциллятора с затуханием и без действий внешней силы

Построим модель колебания гармонического осциллятора с затуханием и под действием внешней силы на Modelica.

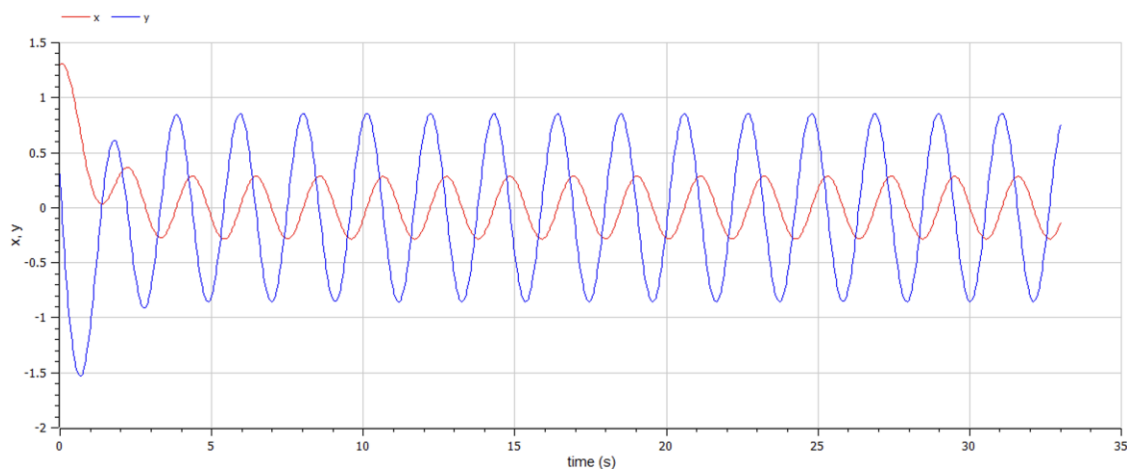
```
model Var3
  constant Real w = 0.3;
  constant Real g = 3;
  Real x;
  Real y;
  Real t = time;
initial equation
  x = 1.3;
  y = 0.3;
equation
  der(x) = y;
  der(y) = -w * x - g * y;
  annotation(experiment(StartTime = 0, StopTime = 33, Interval = 0.05));
end Var3;
```



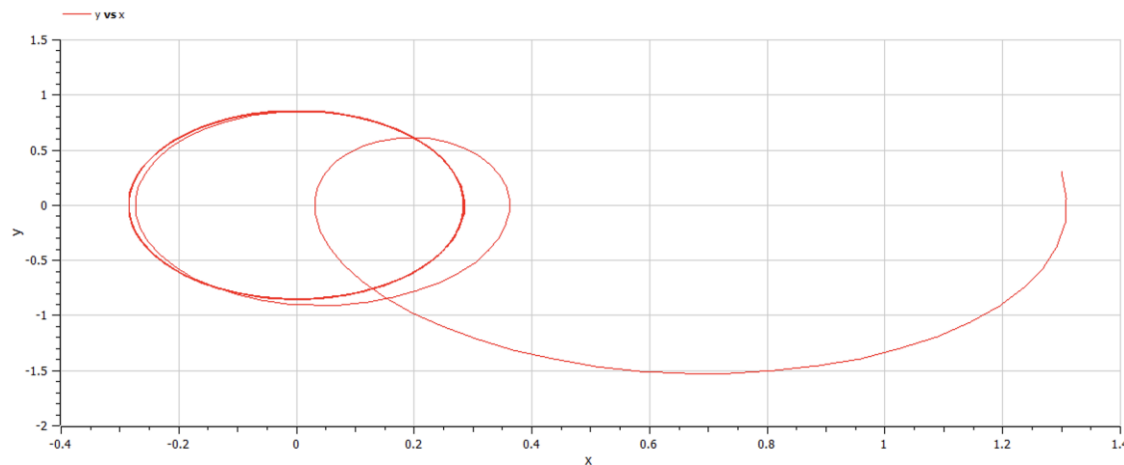
```
1 model Var3
2   constant Real w = 0.3;
3   constant Real g = 3;
4   Real x;
5   Real y;
6   Real t = time;
7   initial equation
8     x = 1.3;
9     y = 0.3;
10  equation
11    der(x) = y;
12    der(y) = -w * x - g * y;
13    annotation(experiment(StartTime = 0, StopTime = 33, Interval = 0.05));
14  end Var3;
15
16
```

Ln: 15, Col: 0 Welcome Modeling Plotting Debugging

Modelica. Скрипт. Колебания гармонического осциллятора с затуханием и под действием внешней силы



Modelica. Модель. Решение уравнения гармонического осциллятора с затуханием и под действием внешней силы



Modelica. Модель. Фазовый портрет осциллятора с затуханием и без под действием внешней силы

Вывод

Повысили навыки в написании программ на Julia. Улучшили понимание моделирования на OpenModelica. Познакомились с Pluto. Повысили навыки в решении ДУ.

Ресурсы

- Julia. URL: http://www.unn.ru/books/met_files/JULIA_tutorial.pdf.
- OpenModelica. URL: <https://ru.wikipedia.org/wiki/OpenModelica>.
- Модель гармонических колебаний. RUDN. URL: <https://esystem.rudn.ru/mod/resource/view.php?id=967241>.
- Pluto. URL: <https://plutojl.org/>.
- Plots in Julia. URL: <https://docs.juliaplots.org/latest/tutorial/>.
- Differential Equations in Julia. URL: https://docs.sciml.ai/DiffEqDocs/stable/getting_started/.