Отчет по лабораторной работе №4

по дисциплине: Математическое моделирование

Ким Михаил Алексеевич

Содержание

Список иллюстраций

1 Цель работы

Продолжить знакомство с функционалом языка программирования Julia, дополнительных библиотек (DifferentialEquations, Plots), интерактивного блокнота Pluto, а также интерактивной командной строкой REPL. Продолжить ознакомление с языком моделирования Modelica и программным обеспечением OpenModelica. Используя эти средства, описать математическую модель гармонических колебаний.

2 Задание

Постройте фазовый портрет гармонического осциллятора и решение уравнения гармонического осциллятора для следующих случаев:

1. Колебания гармонического осциллятора без затуханий и без действий внешней силы $\ddot{x} + 2.5x = 0$.
2. Колебания гармонического осциллятора с затуханием и без действий внешней силы $\ddot{x} + 10\dot{x} + 11x = 0$.
3. Колебания гармонического осциллятора с затуханием и под действием внешней силы $\ddot{x} + \dot{x} + x = 3\sin(t)$.

На интервале $t \in [0;\ 65]$ (шаг 0.05) с начальными условиями $x_0 = -1,\ y_0 = 2$.

3 Теоретическое введение

3.1 Модель гармонических колебаний

Модель гармонических колебаний является одной из фундаментальных моделей в физике. Она описывает поведение объекта, который движется с постоянной частотой и амплитудой.

Представим, что у нас есть объект, который движется вокруг своей равновесной позиции. Это может быть, к примеру, маятник часов или колебательный контур в электрической цепи. Если мы будем наблюдать за движением объекта с течением времени, то заметим, что движение повторяется с постоянной частотой и амплитудой. Эти повторения и называются гармоническими колебаниями [1].

Модель гармонических колебаний находит применение во многих областях, включая физику, математику, инженерию, акустику и другие.

Гармонические колебания широко используются в радиофизике и телекоммуникациях для передачи информации. В этом случае частота колебаний используется для кодирования информации, которая затем может быть передана по радиоволнам или кабельным линиям.

Также гармонические колебания используются при изучении механики и электромагнетизма. Они позволяют предсказывать поведение системы в зависимости от начальных условий и параметров [2].

Система, способная совершать гармонические колебания, называется линейным гармоническим осциллятором.

Линейный гармонический осциллятор может быть использован для моделирования многих процессов, включая электрические контуры, оптические системы и колебания молекул.

Также линейный гармонический осциллятор является базовым элементом в физике твердого тела и квантовой механике [3].

В квантовой механике линейный гармонический осциллятор является важнейшей моделью для изучения колебаний атомов в молекулах и фотонах внутри оптических резонаторов.

Помимо этого, линейный гармонический осциллятор может быть использован для изучения резонанса и демпфирования, а также для моделирования волновых процессов в физике и инженерии [4].

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.

Для однозначной разрешимости ОДУ второго порядка необходимо задать два начальных условия вида:

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

Уравнение ОДУ второго порядка можно переписать в следующем виде как систему:

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

Начальные условия для такой системы примут вид:

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

Также в наше уравнение ОДУ второго порядка можно добавить правую часть. Тогда система примет вид:

$$
\begin{cases}
\dot{x} = y \\
\dot{y} = -2\gamma y - \omega_0^2 x + f(t)
\end{cases}
$$

Заметим, что ОДУ может иметь нулевой коэффицент γ — тогда система будет являться консервативной, и слагаемое из системы просто убирается.

При отображении зависимости координаты y от координаты x мы получим фазовый портрет, который также нужно отрисовать при выполнении лабораторной работы [5].

4 Выполнение лабораторной работы

4.1 Pluto.jl

4.1.1 Задание №1

1. Пишем программу, воспроизводящую модель на языке программирования Julia с использованием интерактивного блокнота Pluto (рис. 4.1, 4.2, 4.3, 4.4, 4.5).

```
begin
    import Pkg
    Pkg.activate()
    using DifferentialEquations
    using LaTeXStrings
    import Plots
end

begin
    const ⊠⊠² = 2.5

    "Начальные условия: u⊠[1] -- x⊠, u⊠[2] --   y⊠"
    u⊠ = [-1, 2]

    "Период времени"
    T = (0.0, 65.0)
```

```
end

"Правая часть нашей системы, p, t не   используются. u[1] -- x, u[2] -- y"
function F!(du, u, p, t)
    du[1] = u[2]
    du[2] = - ⊠⊠² * u[1]
end

prob = ODEProblem(F!, u⊠, T)

sol = solve(prob, saveat=0.05, abstol=1e-8, reltol=1e-8)

begin
    const xx = []
    const yy = []
    for u in sol.u
        x, y = u
        push!(xx, x)
        push!(yy, y)
    end
    Time = sol.t
    Time
end

    begin
    fig = Plots.plot(
        layout=(1, 2),
        dpi=150,
        grid=:xy,
        gridcolor=:black,
        gridwidth=1,
        # aspect_ratio=:equal,
        size=(800, 400),
```

```
        plot_title="Модель гармонических    колебаний"
    )

    Plots.plot!(
        fig[1],
        Time,
        [xx, yy],
        color=[:red :blue],
        xlabel="t",
        ylabel="x(t), y(t)=x'(t)",
        label=["x(t)" "y(t)=x'(t)"]
    )

    Plots.plot!(
        fig[2],
        xx,
        yy,
        color=[:gray],
        xlabel="x(t)",
        ylabel="y(t)=x'(t)",
        label="Фазовый портрет"
    )
end
```

```
begin
    import Pkg
    Pkg.activate()
    using DifferentialEquations
    using LaTeXStrings
    import Plots
end
```

```
Activating project at `~/.julia/environments/v1.8`
```

T

Период времени

```
begin
    const ω₀² = 2.5

    "Начальные условия: u₀[1] -- x₀, u₀[2] -- y₀"
    u₀ = [-1, 2]

    "Период времени"
    T = (0.0, 65.0)
end
```

Рис. 4.1: Импорт библиотек. Задание коэффициентов, начальных условий, периода времени

F!

Правая часть нашей системы, p, t не используются. u[1] – x, u[2] – y

```
"Правая часть нашей системы, p, t не используются. u[1] -- x, u[2] -- y"
function F!(du, u, p, t)
    du[1] = u[2]
    du[2] = - ω₀² * u[1]
end
```

```
prob = ODEProblem with uType Vector{Int64} and tType Float64. In-place: true
       timespan: (0.0, 65.0)
       u0: 2-element Vector{Int64}:
        -1
         2
```

```
prob = ODEProblem(F!, u₀, T)
```

Рис. 4.2: Запись систему уравнений в виде функции. Постановка задачи sol =

	timestamp	value1	value2
1	0.0	-1.0	2.0
2	0.05	-0.896981	2.11862
3	0.1	-0.788358	2.22401
4	0.15	-0.674811	2.31551
5	0.2	-0.557049	2.39254
6	0.25	-0.435806	2.45462
7	0.3	-0.311842	2.50138
8	0.35	-0.185929	2.5325
9	0.4	-0.0588552	2.54781
10	0.45	0.0685865	2.5472
⋮ more			

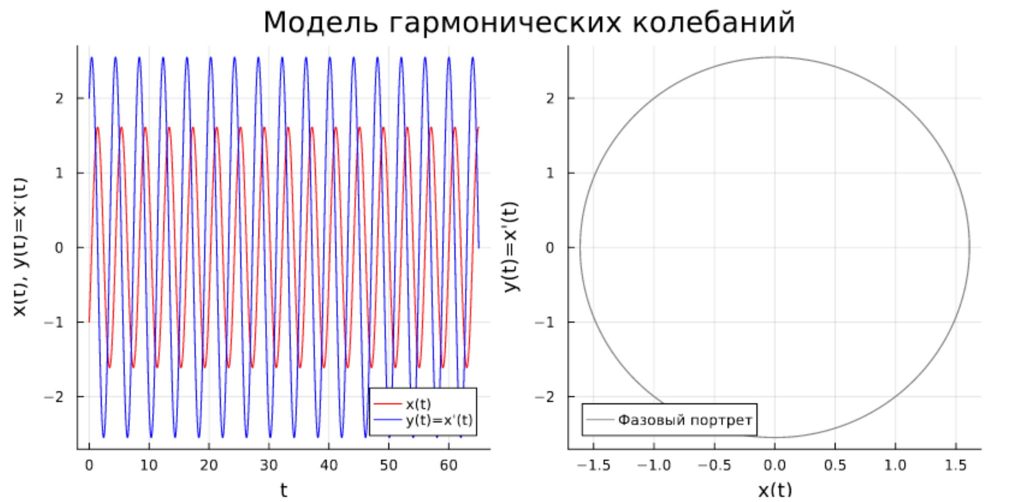
```
sol = solve(prob, saveat=0.05, abstol=1e-8, reltol=1e-8)
```

Рис. 4.3: Решение задачи

```
▸[0.0, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.
```

```
begin
    const xx = []
    const yy = []
    for u in sol.u
        x, y = u
        push!(xx, x)
        push!(yy, y)
    end
    Time = sol.t
    Time
end
```

Рис. 4.4: Формирование трех массивов, содержащих значения x, y, t




```
begin
    fig = Plots.plot(
        layout=(1, 2),
        dpi=150,
        grid=:xy,
        gridcolor=:black,
        gridwidth=1,
        # aspect_ratio=:equal,
        size=(800, 400),
        plot_title="Модель гармонических колебаний"
    )

    Plots.plot!(
        fig[1],
        Time,
        [xx, yy],
        color=[:red :blue],
        xlabel="t",
        ylabel="x(t), y(t)=x'(t)",
        label=["x(t)" "y(t)=x'(t)"]
    )

    Plots.plot!(
        fig[2],
        xx,
        yy,
        color=[:gray],
        xlabel="x(t)",
        ylabel="y(t)=x'(t)",
        label="Фазовый портрет"
    )
end
```

Рис. 4.5: Отрисовка графиков

4.1.2 Задание №2

1. Помимо коэффициента, представляющего собой квадрат собственной частоты колебаний (ω_0^2), добавляем коэффициент, представляющий собой потерю энергии в системе, умноженную на два (2γ). Остальные блоки кода оставляем без изменений. Любуемся результатом (рис. 4.6, 4.7).

```
begin
    const ⊠⊠² = 11.0
    const ⊠⊠2 = 10.0

    "Начальные условия: u⊠[1] -- x⊠, u⊠[2] -- y⊠"
    u⊠ = [-1, 2]

    "Период времени"
    T = (0.0, 65.0)
end

"Правая часть нашей системы, p, t не используются. u[1] -- x, u[2] -- y"
function F!(du, u, p, t)
    du[1] = u[2]
    du[2] = - ⊠⊠2 * u[2] - ⊠⊠² * u[1]
end
```

T

Период времени

```
begin
    const ω₀² = 11.0
    const γ·2 = 10.0

    "Начальные условия: u₀[1] -- x₀, u₀[2] -- y₀"
    u₀ = [-1, 2]

    "Период времени"
    T = (0.0, 65.0)
end
```

F!

Правая часть нашей системы, p, t не используются. u[1] – x, u[2] – y

```
"Правая часть нашей системы, p, t не используются. u[1] -- x, u[2] -- y"
function F!(du, u, p, t)
    du[1] = u[2]
    du[2] = - γ·2 * u[2] - ω₀² * u[1]
end
```

Рис. 4.6: Добавление необходимых коэффициентов и изменение системы уравнений

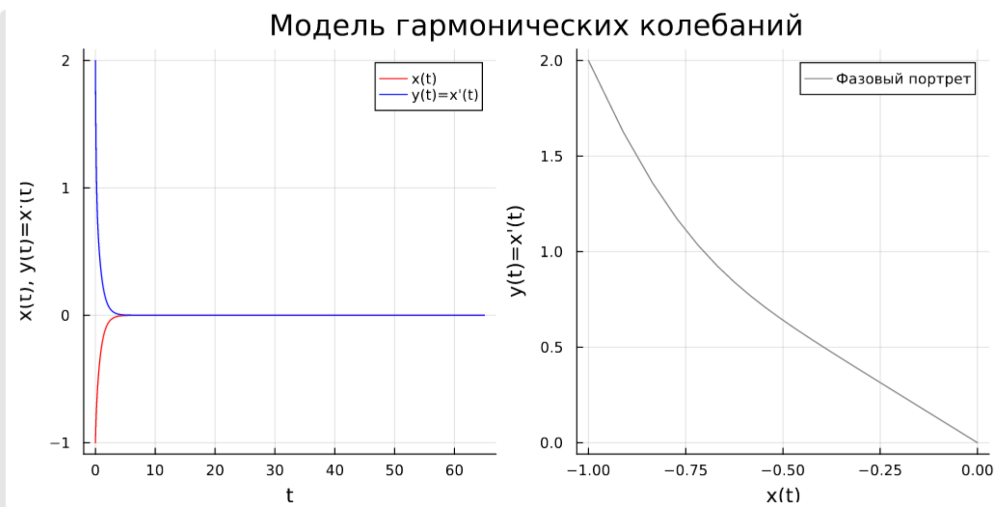


Рис. 4.7: Результат в виде графиков

4.1.3 Задание №3

1. Добавляем правую часть ОДУ ($f(t) = 3\sin(t)$) в систему. Остальные блоки кода оставляем без изменений. Любуемся результатом (рис. 4.8, 4.9).

```
begin
    const ⊠⊠² = 1.0
    const ⊠⊠2 = 1.0

    function f(t)
        return 3 * sin(t)
    end

    "Начальные условия: u⊠[1] -- x⊠, u⊠[2] -- y⊠"
    u⊠ = [-1, 2]

    "Период времени"
    T = (0.0, 65.0)
end

"Правая часть нашей системы, p, t не используются. u[1] -- x, u[2] -- y"
function F!(du, u, p, t)
    du[1] = u[2]
    du[2] = - ⊠⊠2* u[2] - ⊠⊠² * u[1] + f(t)
end
```

T

Период времени

```
begin
    const ω₀² = 1.0
    const γ·2 = 1.0

    function f(t)
        return 3 * sin(t)
    end

    "Начальные условия: u₀[1] -- x₀, u₀[2] -- y₀"
    u₀ = [-1, 2]

    "Период времени"
    T = (0.0, 65.0)
end
```

F!

Правая часть нашей системы, p, t не используются. u[1] – x, u[2] – y

```
"Правая часть нашей системы, p, t не используются. u[1] -- x, u[2] -- y"
function F!(du, u, p, t)
    du[1] = u[2]
    du[2] = - γ·2* u[2] - ω₀² * u[1] + f(t)
end
```

Рис. 4.8: Добавление правой части ОДУ и изменение системы уравнений

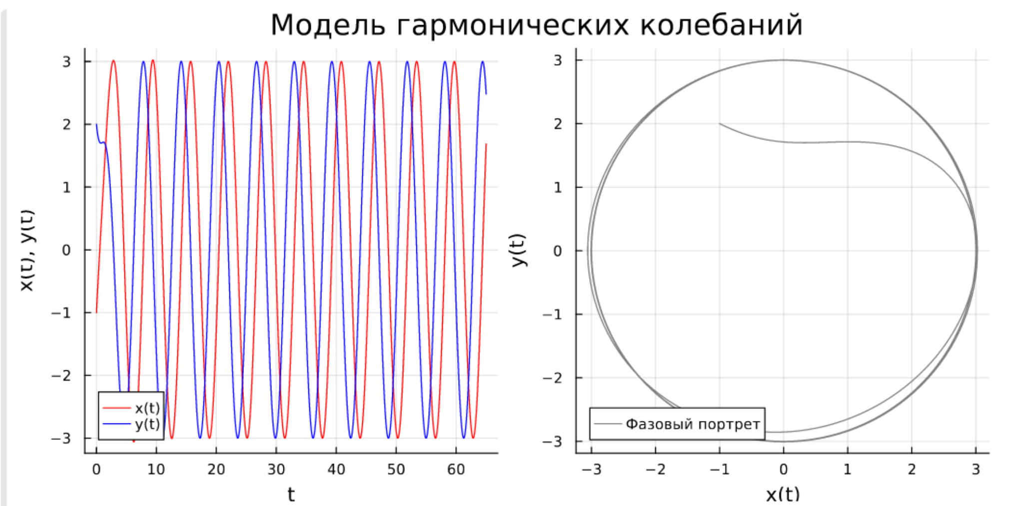


Рис. 4.9: Результат в виде графиков

4.2 Julia

4.2.1 Задание №1

1. Код на Julia в файле аналогичен тому же, написанному с использованием Pluto. Единственные различия: блоки перенесены в файл в виде построчного алгоритма без повторяющихся ‘begin’ и ‘end’, отличающийся синтаксис подключения библиотек, выгрузка графиков в виде изображений при помощи метода в последней строчке кода (рис. 4.10, 4.11).

```
using DifferentialEquations
using Plots

const ⊠⊠² = 2.5

"Начальные условия: u⊠[1] -- x⊠, u⊠[2] -- y⊠"
u⊠ = [-1, 2]

"Период времени"
T = (0.0, 65.0)

"Правая часть нашей системы, p, t не используются. u[1] -- x, u[2] -- y"
function F!(du, u, p, t)
    du[1] = u[2]
    du[2] = - ⊠⊠² * u[1]
end


prob = ODEProblem(F!, u⊠, T)
sol = solve(prob, saveat=0.05, abstol=1e-8, reltol=1e-8)
```

```
const xx = []
const yy = []
for u in sol.u
    x, y = u
    push!(xx, x)
    push!(yy, y)
end
Time = sol.t

fig = Plots.plot(
        layout=(1, 2),
        dpi=150,
        grid=:xy,
        gridcolor=:black,
        gridwidth=1,
        # aspect_ratio=:equal,
        size=(800, 400),
        plot_title="Модель гармонических колебаний"
    )

Plots.plot!(
    fig[1],
    Time,
    [xx, yy],
    color=[:red :blue],
    xlabel="t",
    ylabel="x(t), y(t)=x'(t)",
    label=["x(t)" "y(t)=x'(t)"]
)
```

```
Plots.plot!(
    fig[2],
    xx,
    yy,
    color=[:gray],
    xlabel="x(t)",
    ylabel="y(t)=x'(t)",
    label="Фазовый портрет"
)


savefig(fig, "../lab4_1")
```

```julia
using DifferentialEquations
using Plots

const ω₀² = 2.5

"Начальные условия: u₀[1] -- x₀, u₀[2] -- y₀"
u₀ = [-1, 2]

"Период времени"
T = (0.0, 65.0)

"Правая часть нашей системы, p, t не используются. u[1] -- x, u[2] -- y"
function F!(du, u, p, t)
    du[1] = u[2]
    du[2] = - ω₀² * u[1]
end


prob = ODEProblem(F!, u₀, T)
sol = solve(prob, saveat=0.05, abstol=1e-8, reltol=1e-8)

const xx = []
const yy = []
for u in sol.u
    x, y = u
    push!(xx, x)
    push!(yy, y)
end
Time = sol.t

fig = Plots.plot(
        layout=(1, 2),
        dpi=150,
        grid=:xy,
        gridcolor=:black,
        gridwidth=1,
        # aspect_ratio=:equal,
        size=(800, 400),
        plot_title="Модель гармонических колебаний"
    )

Plots.plot!(
    fig[1],
    Time,
    [xx, yy],
    color=[:red :blue],
    xlabel="t",
    ylabel="x(t), y(t)=x'(t)",
    label=["x(t)" "y(t)=x'(t)"]
)

Plots.plot!(
    fig[2],
    xx,
    yy,
    color=[:gray],
    xlabel="x(t)",
    ylabel="y(t)=x'(t)",
    label="Фазовый портрет"
)


savefig(fig, "../lab4_1")
```

Рис. 4.10: Код программы на Julia. Аналогичен коду задания для Pluto.jl

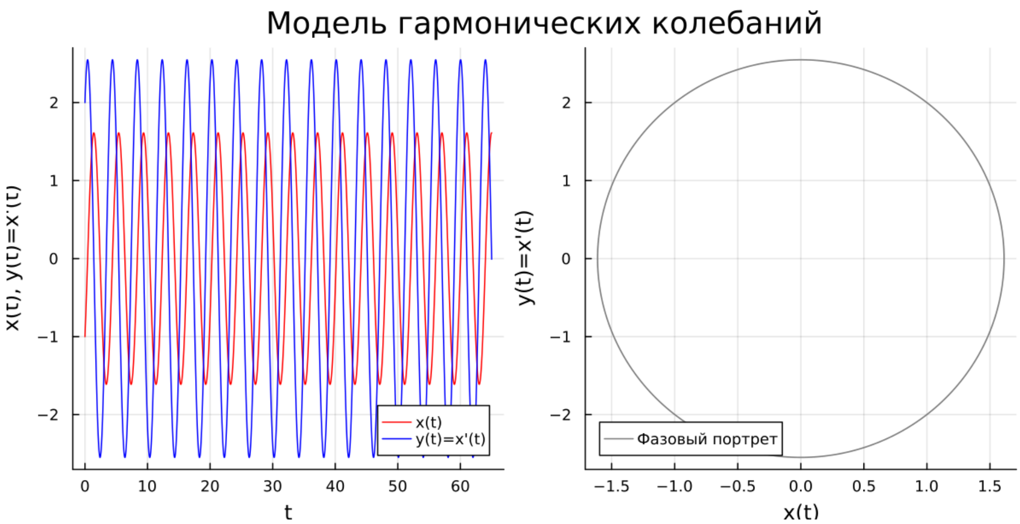


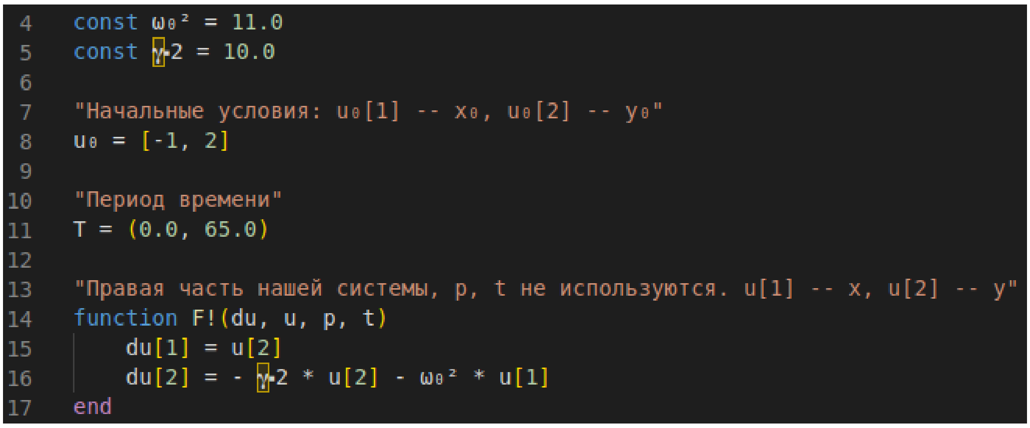
Рис. 4.11: Результат в виде графиков

4.2.2 Задание №2

1. Изменяем необходимые строчки и любуемся результатом (подробное объяснение давалось в предыдущей главе) (рис. 4.12, 4.13).

```
const ⊠⊠² = 11.0
const ⊠⊠2 = 10.0


function F!(du, u, p, t)
    du[1] = u[2]
    du[2] = - ⊠⊠2 * u[2] - ⊠⊠² * u[1]
end
```



```
const ω₀² = 11.0
const γ2 = 10.0

"Начальные условия: u₀[1] -- x₀, u₀[2] -- y₀"
u₀ = [-1, 2]

"Период времени"
T = (0.0, 65.0)

"Правая часть нашей системы, p, t не используются. u[1] -- x, u[2] -- y"
function F!(du, u, p, t)
    du[1] = u[2]
    du[2] = - γ2 * u[2] - ω₀² * u[1]
end
```

Рис. 4.12: Измененная часть кода

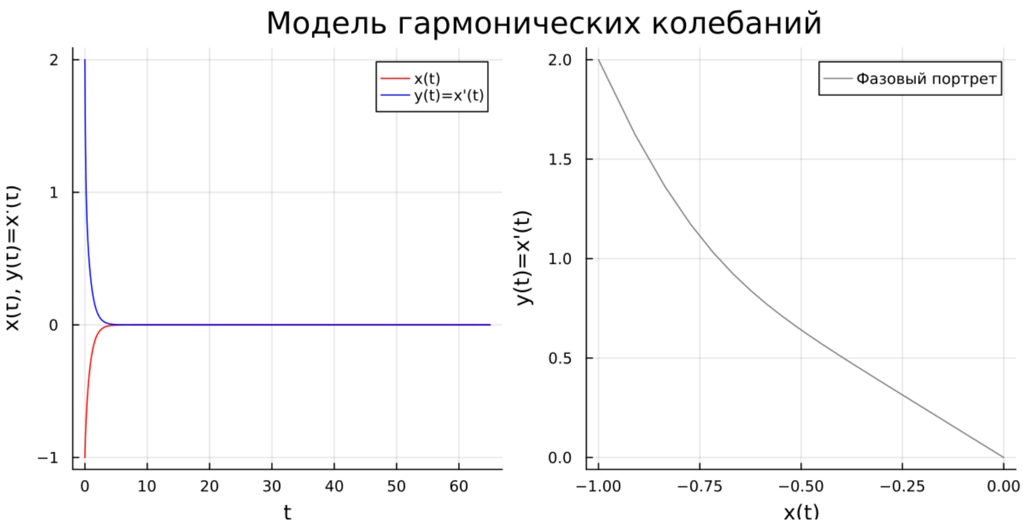


Рис. 4.13: Результат в виде графиков

4.2.3 Задание №3

1. Изменяем необходимые строчки и любуемся результатом (подробное объяснение давалось в предыдущей главе) (рис. 4.14, 4.15).

```
const ⊠⊠² = 1.0
const ⊠⊠2 = 1.0
function f(t)
```

```
    return 3 * sin(t)
end


function F!(du, u, p, t)
    du[1] = u[2]
    du[2] = - ⊠⊠2* u[2] - ⊠⊠² * u[1] + f(t)
end
```

```
const ω₀² = 1.0
const γ2 = 1.0
function f(t)
    return 3 * sin(t)
end

"Начальные условия: u₀[1] -- x₀, u₀[2] -- y₀"
u₀ = [-1, 2]

"Период времени"
T = (0.0, 65.0)

"Правая часть нашей системы, p, t не используются. u[1] -- x, u[2] -- y"
function F!(du, u, p, t)
    du[1] = u[2]
    du[2] = - γ2* u[2] - ω₀² * u[1] + f(t)
end
```

Рис. 4.14: Измененная часть кода

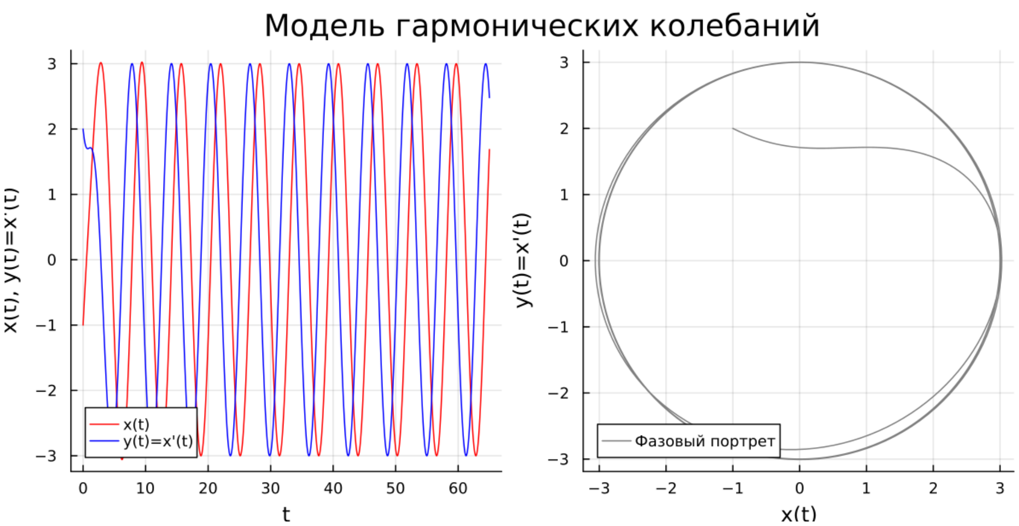


Рис. 4.15: Результат в виде графиков

4.3 Modelica

4.3.1 Задание №1

1. По аналогии с Julia пишем программу, воспроизводящую модель гармонических колебаний на языке моделирования Modelica с использованием ПО OpenModelica. Любуемся результатами (рис. 4.16, 4.17, 4.18).

```
model lab4_1
  constant Real omega_0_square = 2.5;
  Real t = time;
  Real x;
  Real y;
initial equation
  x = -1;
  y = 2;
equation
  der(x) = y;
  der(y) = - omega_0_square * x;
  annotation(experiment(StartTime=0, StopTime=65, Interval = 0.05));
end lab4_1;
```

```
model lab4_1
  constant Real omega_0_square = 2.5;
  Real t = time;
  Real x;
  Real y;
initial equation
  x = -1;
  y = 2;
equation
  der(x) = y;
  der(y) = - omega_0_square * x;
  annotation(experiment(StartTime=0, StopTime=65, Interval = 0.05));
end lab4_1;
```

Рис. 4.16: Определяем коэффициенты, изменяемые переменные, начальные условия, систему уравнений, а также начальное/конечное время и частоту разбиения при симуляции

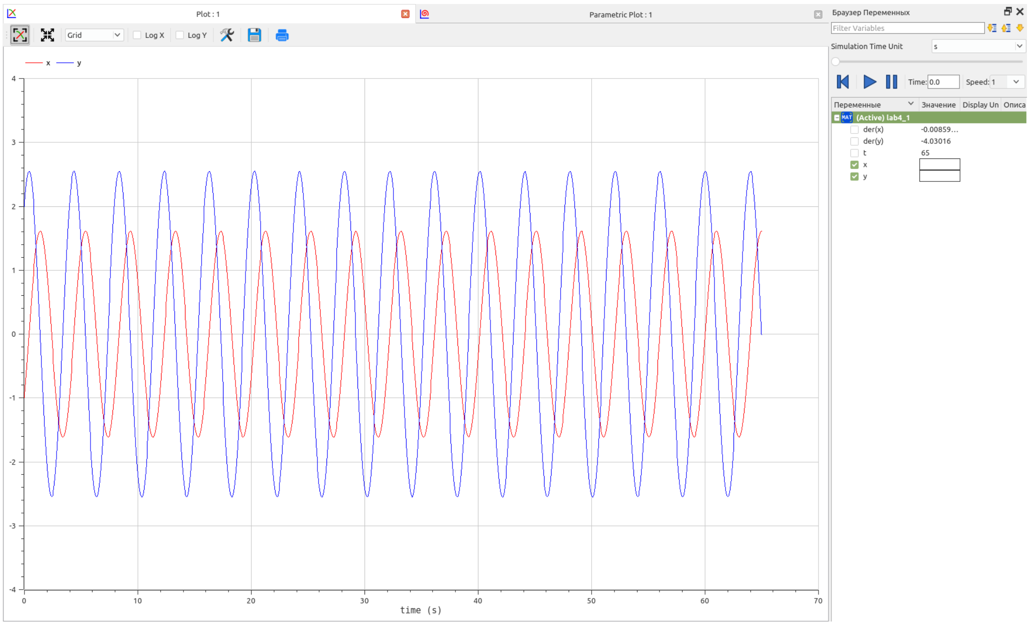


Рис. 4.17: Результат в виде графика зависимости x и y от t

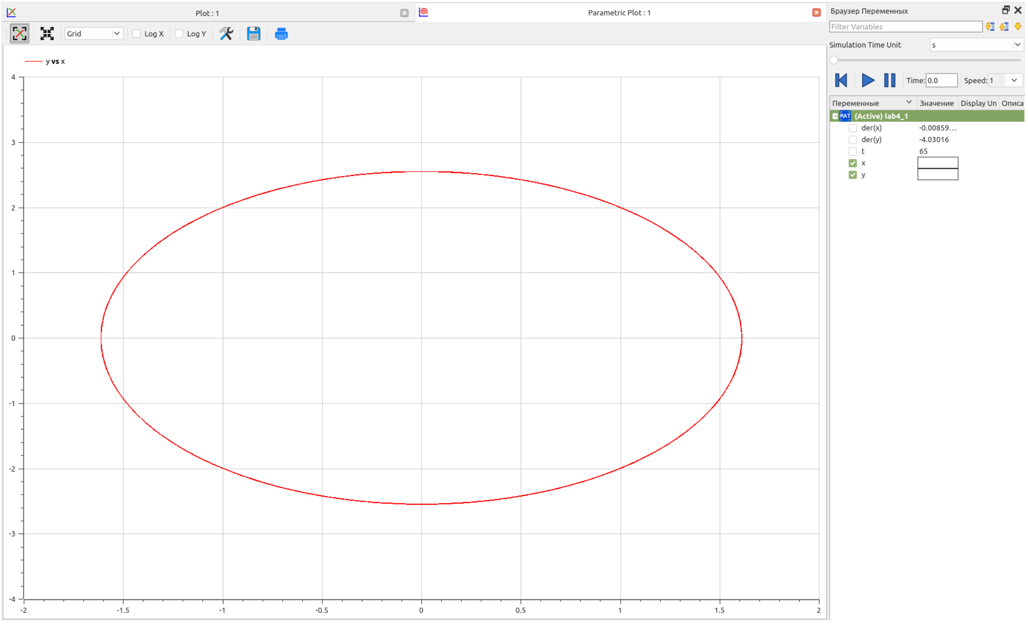


Рис. 4.18: Результат в виде графика зависимости y от x

4.3.2 Задание №2

1. По аналогии с Julia пишем программу для второго случая. Любуемся результатами (рис. 4.19, 4.20, 4.21).

```
model lab4_2
  constant Real omega_0_square = 11.0;
  constant Real gamma_2 = 10.0;
  Real t = time;
  Real x;
  Real y;
initial equation
  x = -1;
  y = 2;
equation
  der(x) = y;
```

```
  der(y) = - gamma_2*y - omega_0_square*x;
  annotation(experiment(StartTime=0, StopTime=65, Interval = 0.05));
end lab4_2;
```

```
model lab4_2
  constant Real omega_0_square = 11.0;
  constant Real gamma_2 = 10.0;
  Real t = time;
  Real x;
  Real y;
initial equation
  x = -1;
  y = 2;
equation
  der(x) = y;
  der(y) = - gamma_2*y - omega_0_square*x;
  annotation(experiment(StartTime=0, StopTime=65, Interval = 0.05));
end lab4_2;
```

Рис. 4.19: По сравнению с прошлым случаем добавляется коэффициент и изменятся система

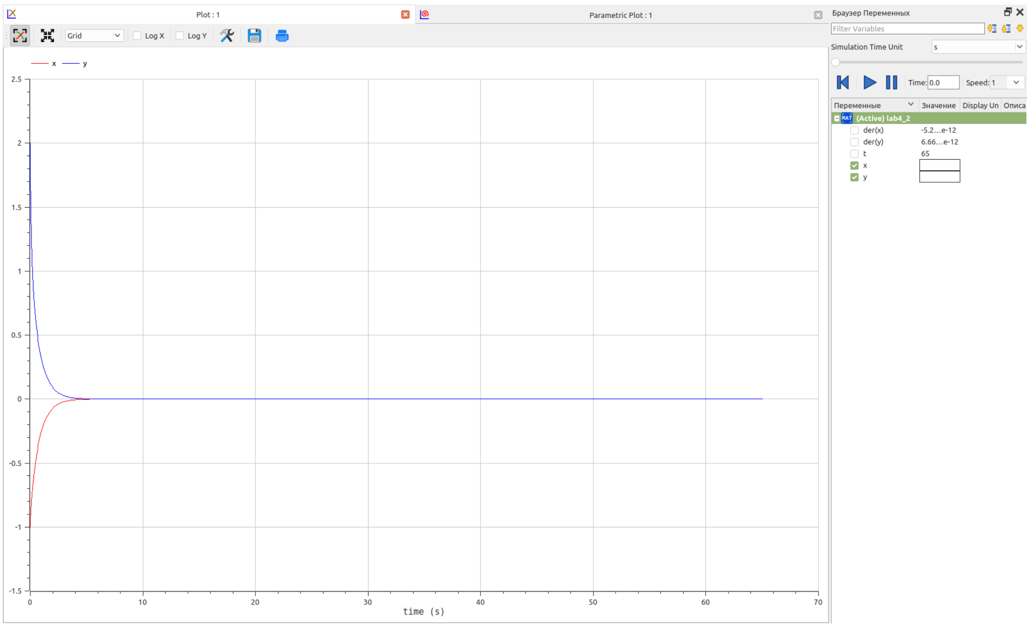


Рис. 4.20: Результат в виде графика зависимости x и y от t

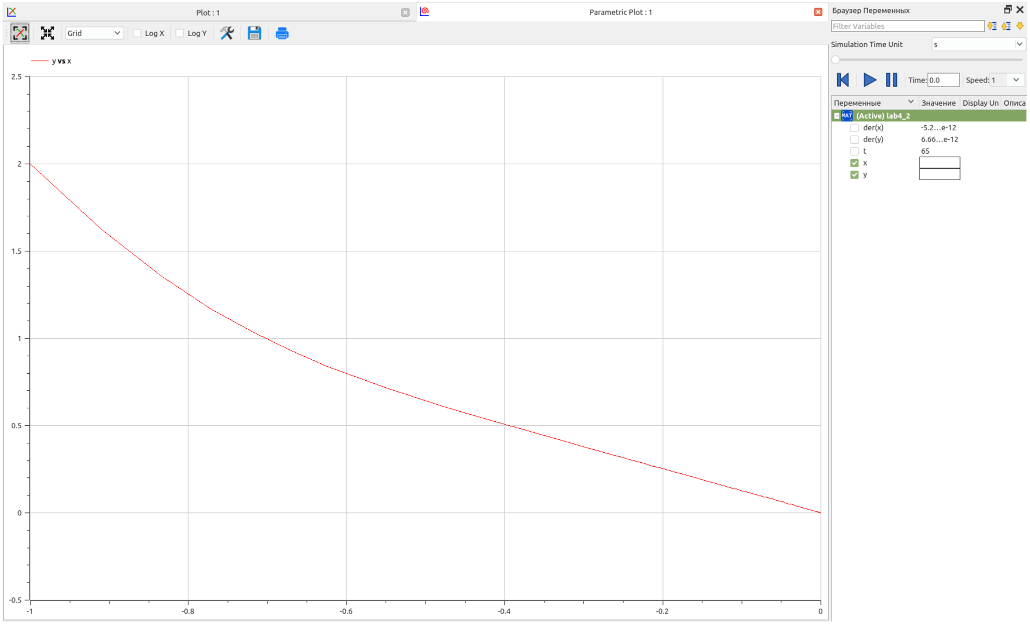


Рис. 4.21: Результат в виде графика зависимости y от x

4.3.3 Задание №3

1. По аналогии с Julia пишем программу для третьего случая. Любуемся результатами (рис. 4.22, 4.23, 4.24).

```
model lab4_3
  constant Real omega_0_square = 1.0;
  constant Real gamma_2 = 1.0;
  Real t = time;
  Real x;
  Real y;
initial equation
  x = -1;
  y = 2;
equation
  der(x) = y;
```

```
  der(y) = - gamma_2 * y - omega_0_square * x + 3 * sin(t);
  annotation(experiment(StartTime=0, StopTime=65, Interval = 0.05));
end lab4_3;
```

```
model lab4_3
  constant Real omega_0_square = 1.0;
  constant Real gamma_2 = 1.0;
  Real t = time;
  Real x;
  Real y;
initial equation
  x = -1;
  y = 2;
equation
  der(x) = y;
  der(y) = - gamma_2 * y - omega_0_square * x + 3 * sin(t);
  annotation(experiment(StartTime=0, StopTime=65, Interval = 0.05));
end lab4_3;
```

Рис. 4.22: По сравнению с прошлым случаем добавляется правая часть ОДУ и изменятся система

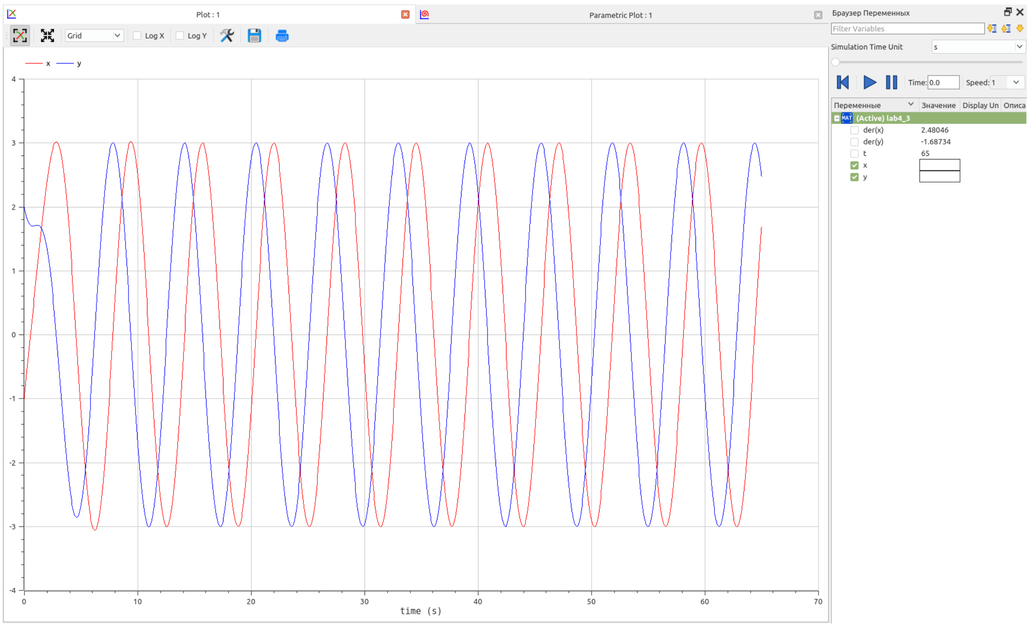


Рис. 4.23: Результат в виде графика зависимости x и y от t

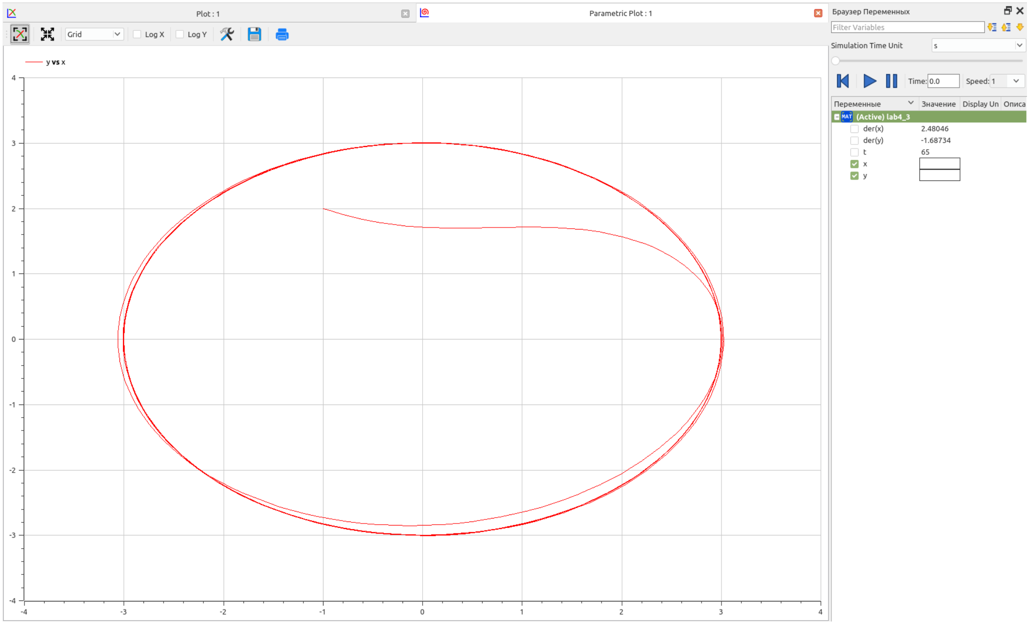


Рис. 4.24: Результат в виде графика зависимости y от x

5 Анализ результатов

На текущем примере построения математической модели гармонических колебаний мы можем продолжить сравнивать язык программирования Julia и язык моделирования Modelica. По сравнению с анализом результатов предыдущей лабораторной работы хотелось бы отметить, что определенные недостатки Julia по сравнению с Modelica (медленная скорость выполнения, объем и читабельность кода) лично для меня сглаживаются, т.к. в первую очередь, я теперь использую Pluto. Скорость отрисовки графиков после изменения кода в интерактивном блокноте в разы быстрее по сравнению с со скоростью сохранения графиков в файл при запуске *.jl. Также частая работа с библиотекой DifferentialEquations ведет к более легкому пониманию кода, пусть и более объемного, нежели на Modelica. Также гибкость настройки точности численного метода решения ОДУ в данный момент на стороне Julia.

Однако, с другой стороны, OpenModelica все еще предоставлят больше возможностей для настройки отображения графиков.

6 Выводы

Продолжил знакомство с функционалом языка программирования Julia, дополнительных библиотек (DifferentialEquations, Plots), интерактивного блокнота Pluto, а также интерактивной командной строкой REPL. Продолжил ознакомление с языком моделирования Modelica и программным обеспечением OpenModelica. Используя эти средства, описал математическую модель гармонических колебаний.

Список литературы

1. Simple harmonic motion [Электронный ресурс]. Wikimedia Foundation, Inc., 2023. URL: https://en.wikipedia.org/wiki/Simple_harmonic_motion.
2. Simple Harmonic Motion (SHM) [Электронный ресурс]. BYJU'S. URL: https://byjus.com/jee/simple-harmonic-motion-shm/.
3. Harmonic oscillator [Электронный ресурс]. Wikimedia Foundation, Inc., 2022. URL: https://en.wikipedia.org/wiki/Harmonic_oscillator.
4. The Harmonic Oscillator [Электронный ресурс]. California Institute of Technology, 2013. URL: https://www.feynmanlectures.caltech.edu/I_21.html.
5. Модель гармонических колебаний [Электронный ресурс]. RUDN, 2023. URL: https://esystem.rudn.ru/mod/resource/view.php?id=967241.