Отчёт по лабораторной работе

Лабораторная работа №4 (вариант 10)

Сергеев Тимофей Сергеевич

Содержание

# 1 Цель работы

Построить фазовый портрет гармонического осциллятора и решение уравнения гармонического осциллятора.

# 2 Задание

* Написать код на языке Julia для случаев:
  + без затуханий и без действий внешней силы;
  + c затуханием и без действий внешней силы;
  + c затуханием и под действием внешней силы;
* Написать код на языке Modelica для случаев:
  + без затуханий и без действий внешней силы;
  + c затуханием и без действий внешней силы;
  + c затуханием и под действием внешней силы;
* Составить отчёт на языке Markdown и сконвертировать его в docx и pdf.
* Подготовить презентацию на языке Markdown и защитить её.

# 3 Теоретическое введение

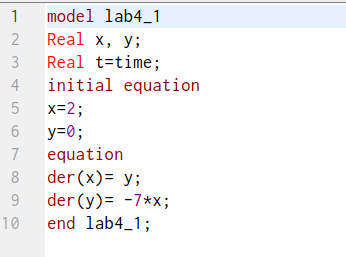
**Julia** – высокоуровневый высокопроизводительный свободный язык программирования с динамической типизацией, созданный для математических вычислений. Эффективен также и для написания программ общего назначения. Синтаксис языка схож с синтаксисом других математических языков (например, MATLAB и Octave), однако имеет некоторые существенные отличия. Julia написан на Си, C++ и Scheme. Имеет встроенную поддержку многопоточности и распределённых вычислений, реализованные в том числе в стандартных конструкциях. [1]

**OpenModelica** – свободное открытое программное обеспечение для моделирования, симуляции, оптимизации и анализа сложных динамических систем. Основано на языке Modelica. Активно развивается Open Source Modelica Consortium, некоммерческой неправительственной организацией. Open Source Modelica Consortium является совместным проектом RISE SICS East AB и Линчёпингского университета. [2]

**Гармоническим осциллятором** называется система, способная совершать гармонические колебания. В физике модель гармонического осциллятора играет важную роль, особенно при исследовании малых колебаний систем около положения устойчивого равновесия. Примером таких колебаний в квантовой механике являются колебания атомов в твердых телах, молекулах и т.д.[3]

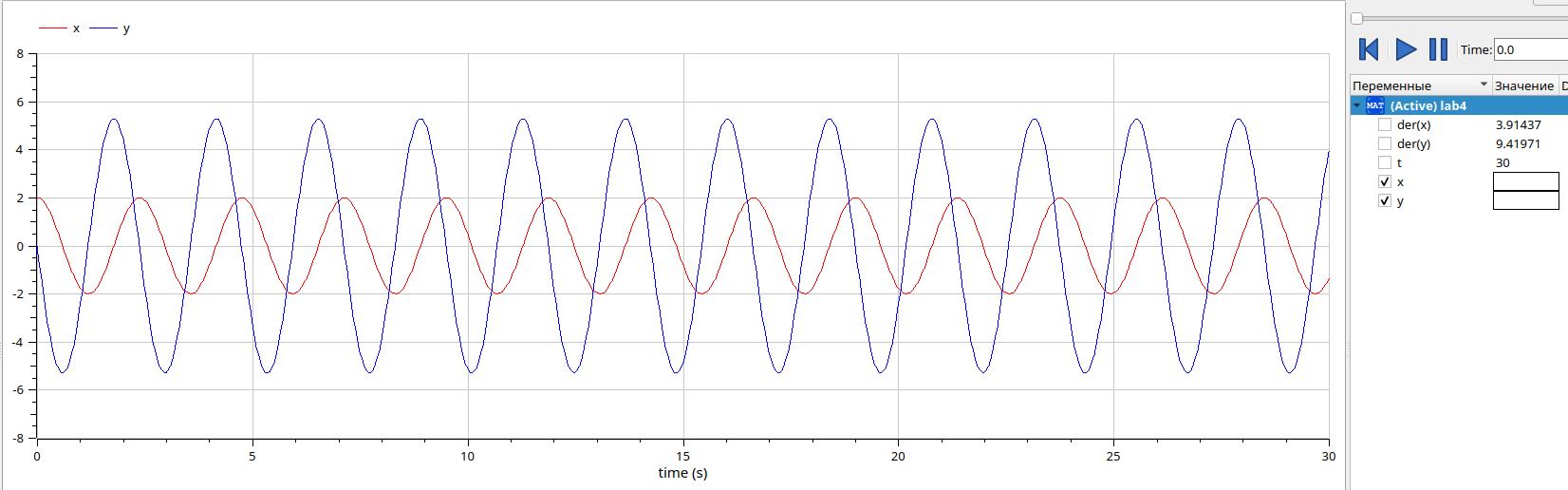
# 4 Выполнение лабораторной работы

1. Рассмотрим код на языке Modelica для первого случая. Объявим переменные типа Real (потому что это тип с плавающим знаком, наиболее подходящий для решения дифференциальных уравнений). Затем инициализурем х и у, подставив данные из условия. После этого пропишем решение наших дифференциальных уравнений (рис. ??).



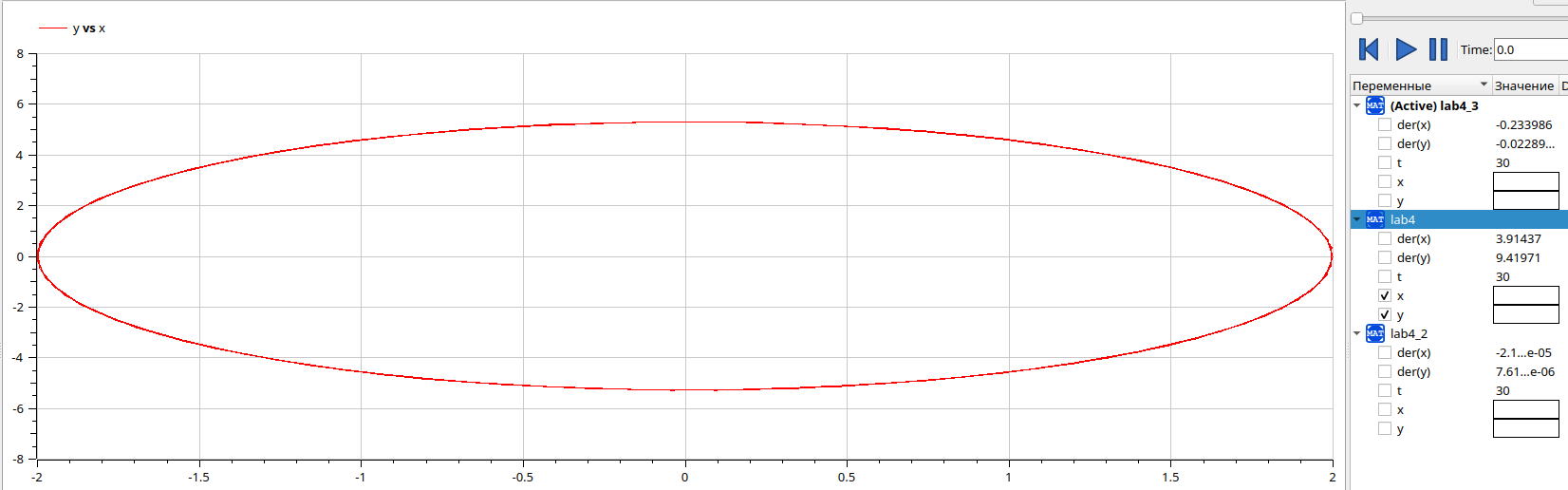
Реализация первого случая на языке Modelica

1. Затем установим настройки симуляции (начальное (0) и конечное (30) время и шаг (0.05)) и запустим симуляцию. Получим следующий результат (рис. ??).



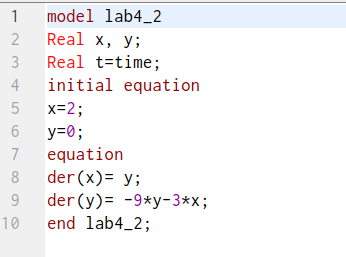
Результат первого случая на языке Modelica

1. Затем необходимо вывести фазовый портрет. В Open Modelica для этого необходимо октрыть новое окно параметрического вывода график, нажать Shift и, удерживая его, поставить галочку у х, после чего поставить галочку у у. В результате на экране появится фазовый портрет (рис. ??).



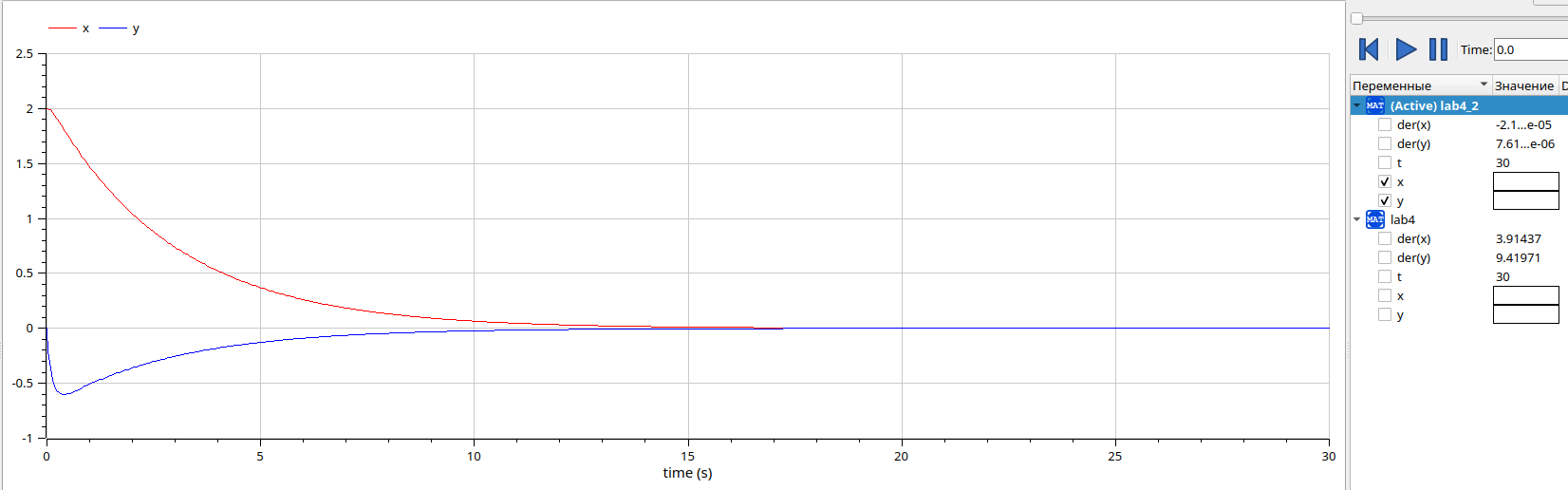
Фазовый портрет первого случая на языке Modelica

1. Аналогичным образом представим код для второго случая (рис. ??).



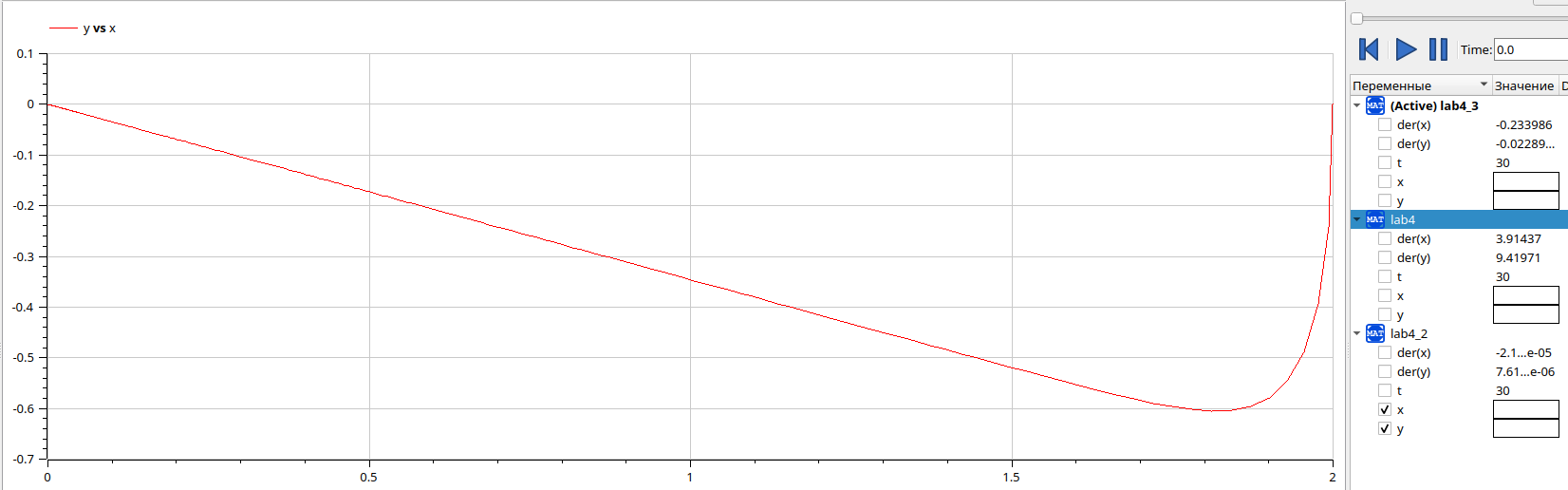
Реализация второго случая на языке Modelica

1. Затем установим настройки симуляции (начальное (0) и конечное (30) время и шаг (0.05)) и запустим симуляцию. Получим следующий результат (рис. ??).



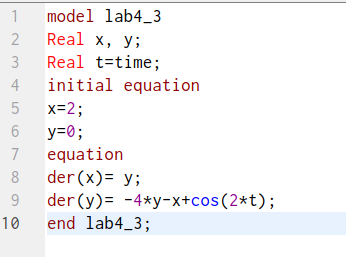
Результат второго случая на языке Modelica

1. Затем выведем фазовый портрет (рис. ??).



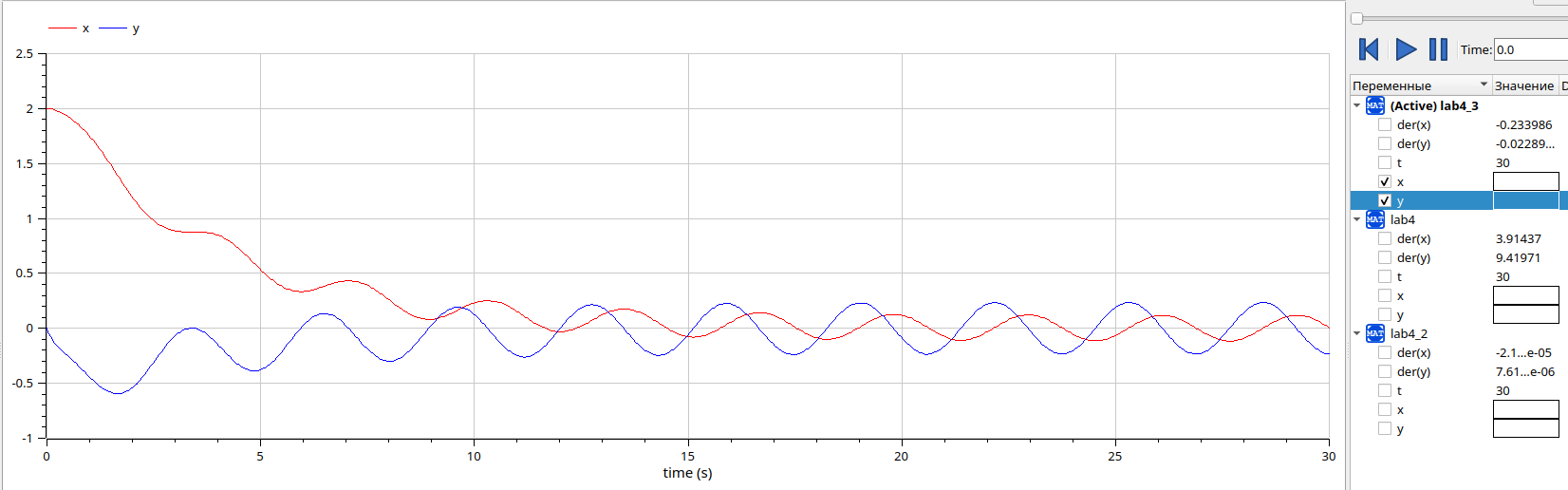
Фазовый портрет второго случая на языке Modelica

1. Аналогичным образом представим код для третьего случая (в отличие от первых двух случаев, добавляется тригонометрическая функция косинуса) (рис. ??).



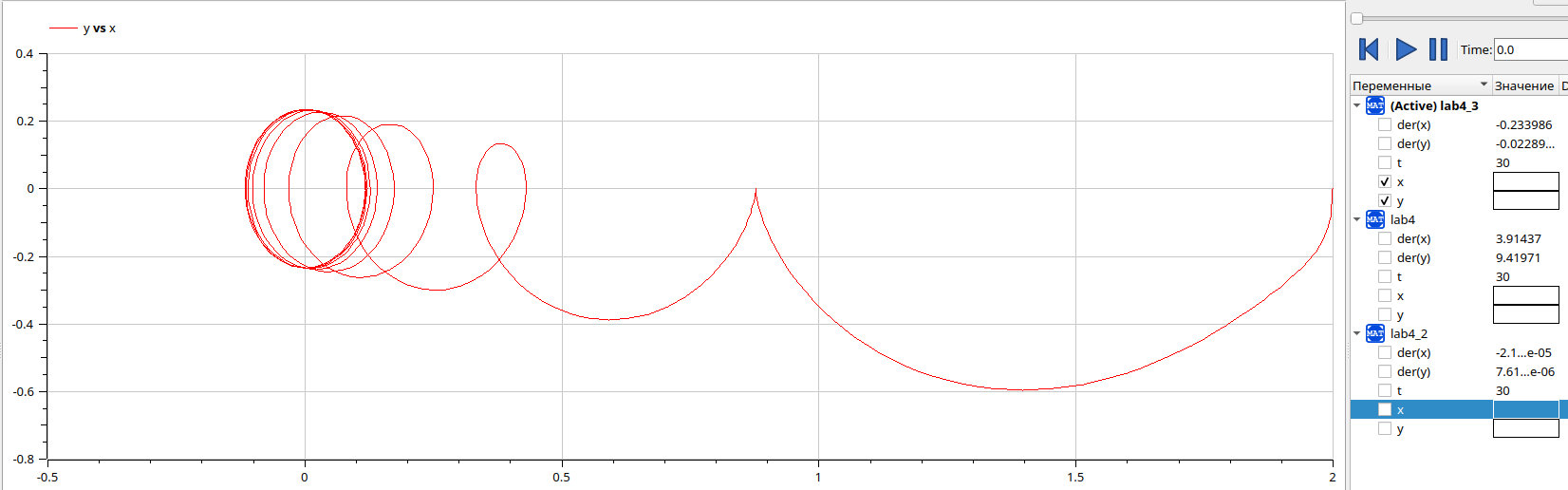
Реализация третьего случая на языке Modelica

1. Затем установим настройки симуляции (начальное (0) и конечное (30) время и шаг (0.05)) и запустим симуляцию. Получим следующий результат (рис. ??).



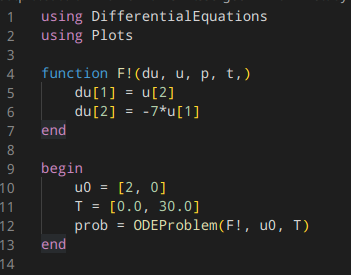
Результат третьего случая на языке Modelica

1. Затем выведем фазовый портрет (рис. ??).

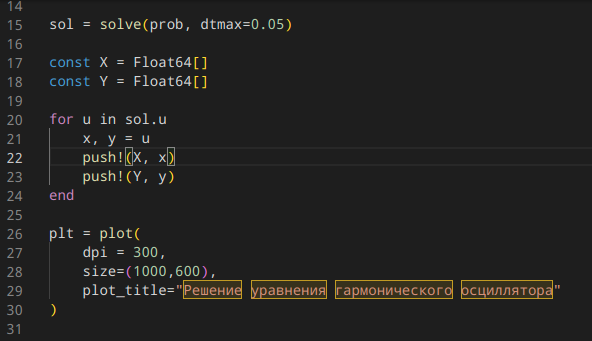


Фазовый портрет третьего случая на языке Modelica

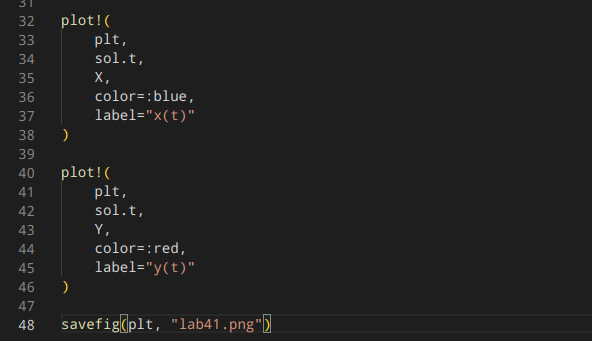
1. Теперь опишем эти случаи на языке Julia. Сперва опишем первый случай (рис. ??, ??, ??).



Подключаем библиотеки, задаём функцию, решающую дифференциальные уравнения. Затем зададим начальные условия.

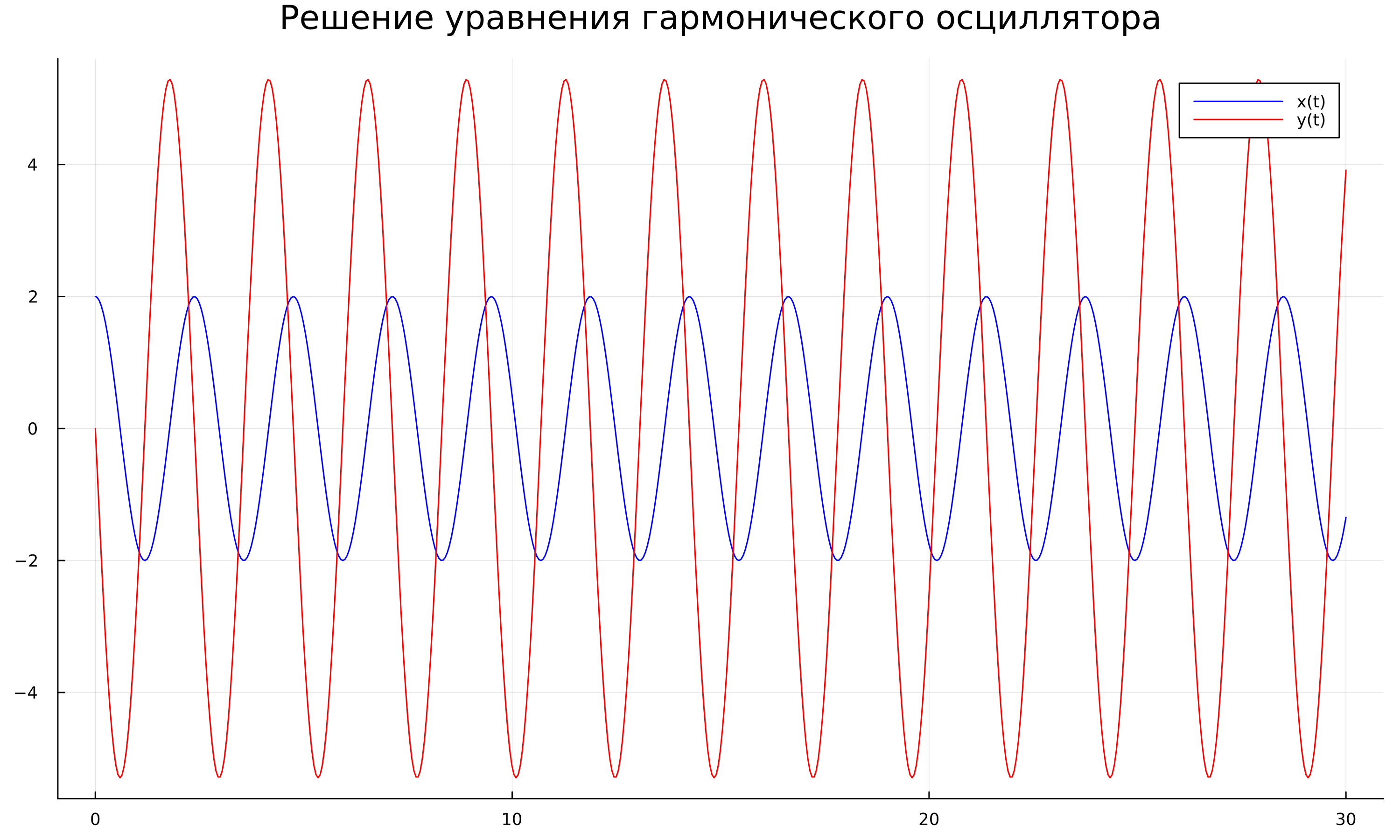


Выполним функцию с данными значениями. Создадим два пустых массива, в которые мы передадим полученные значения. Затем с помощью функционала библиотеки Plots создадим поле для вывода результата.



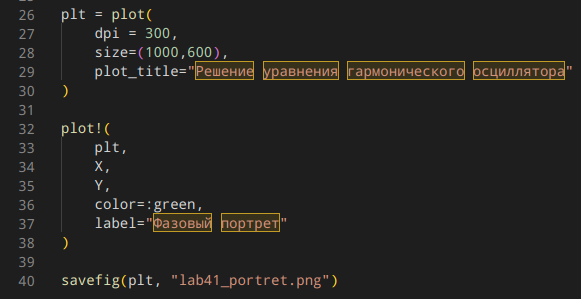
Выведем на экран полученные графы и сохраним результат в формате png

1. Получим следующий результат (рис. ??).



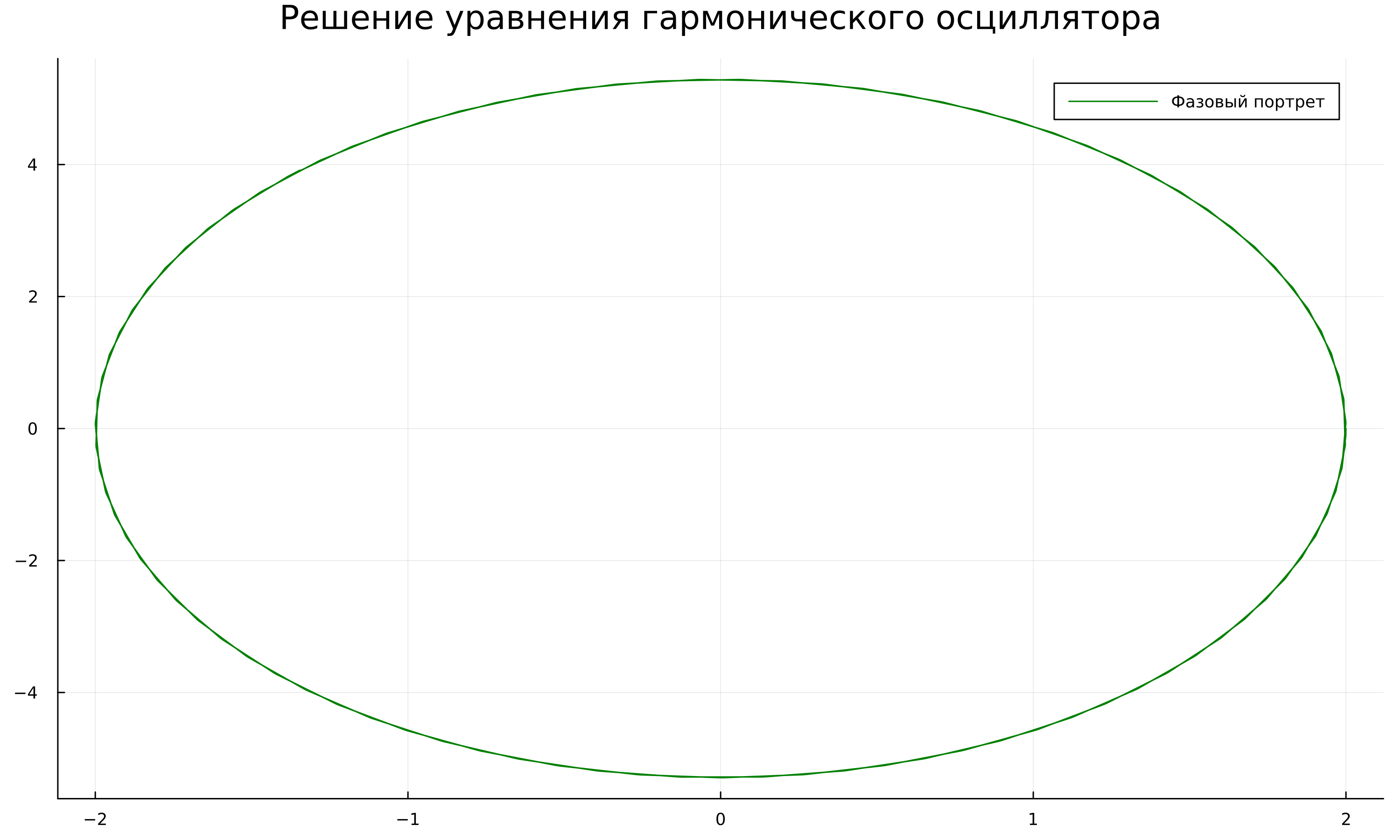
Результат работы программы для первого случая

1. Код для вывода на экран фазового портрета отличается лишь в том, что мы хотим вывести на экран (рис. ??).



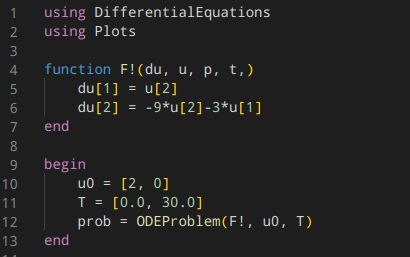
Меняем только последнюю часть кода

1. Получим следующий результат (рис. ??).

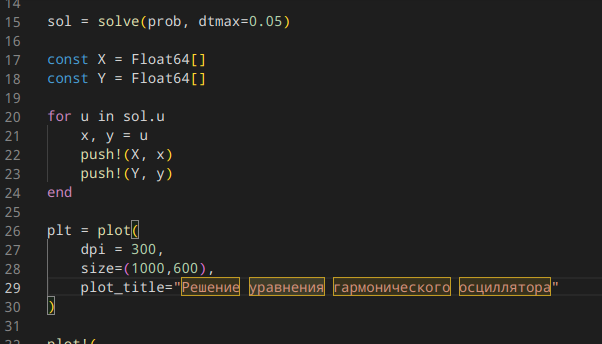


Фазовый портрет для первого случая

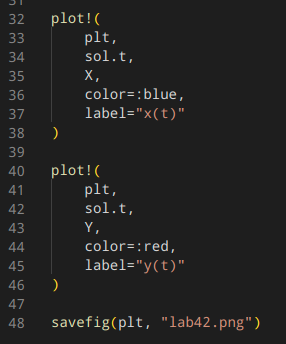
1. Теперь опишем второй случай (рис. ??, ??, ??).



Подключаем библиотеки, задаём функцию, решающую дифференциальные уравнения. Затем зададим начальные условия.

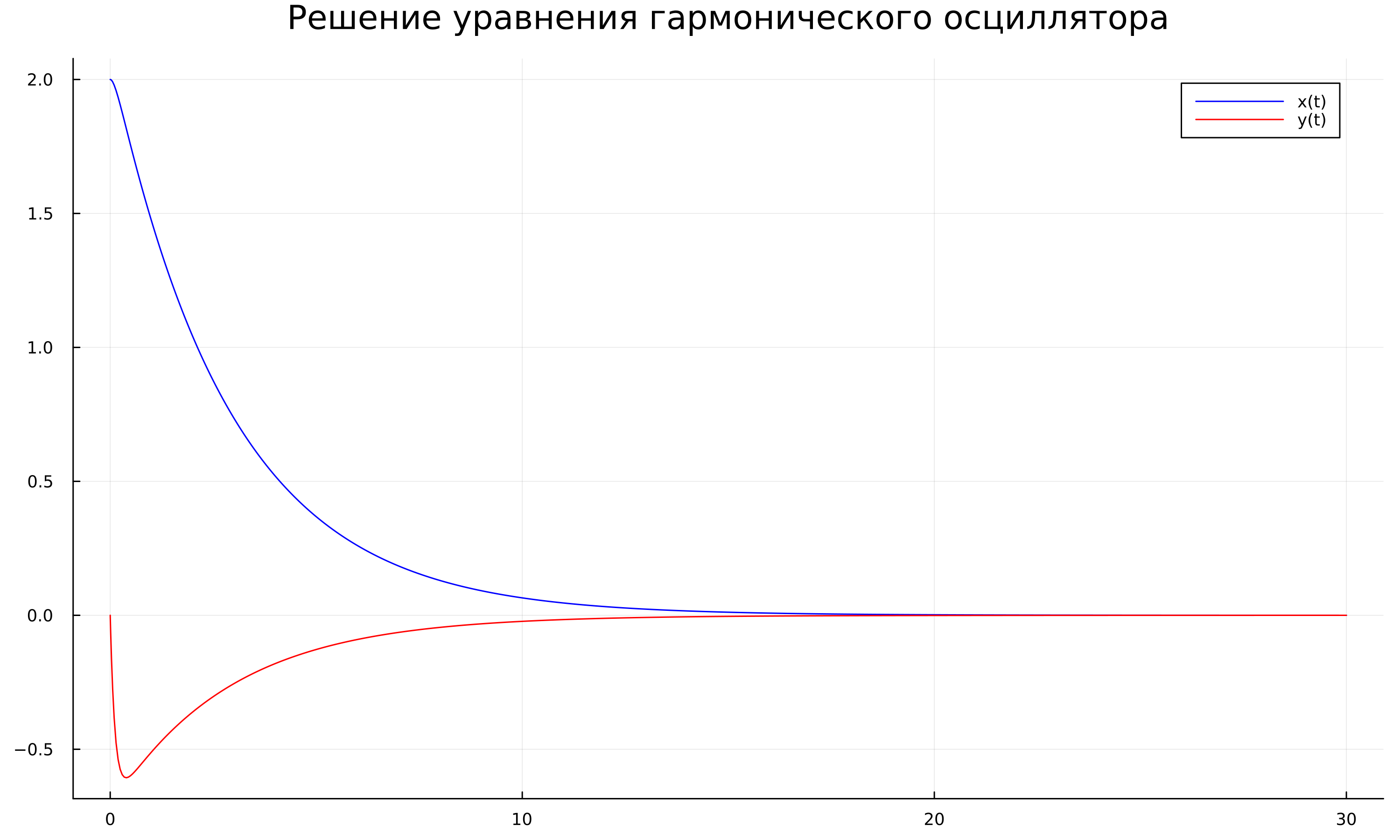


Выполним функцию с данными значениями. Создадим два пустых массива, в которые мы передадим полученные значения. Затем с помощью функционала библиотеки Plots создадим поле для вывода результата.



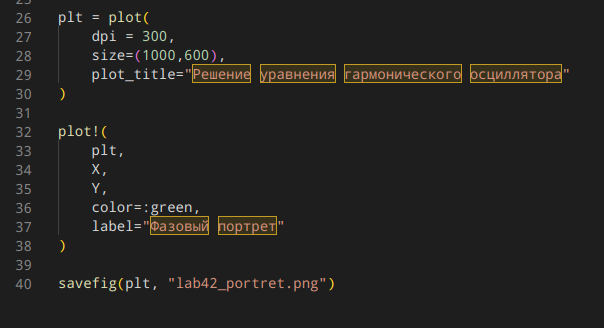
Выведем на экран полученные графы и сохраним результат в формате png

1. Получим следующий результат (рис. ??).



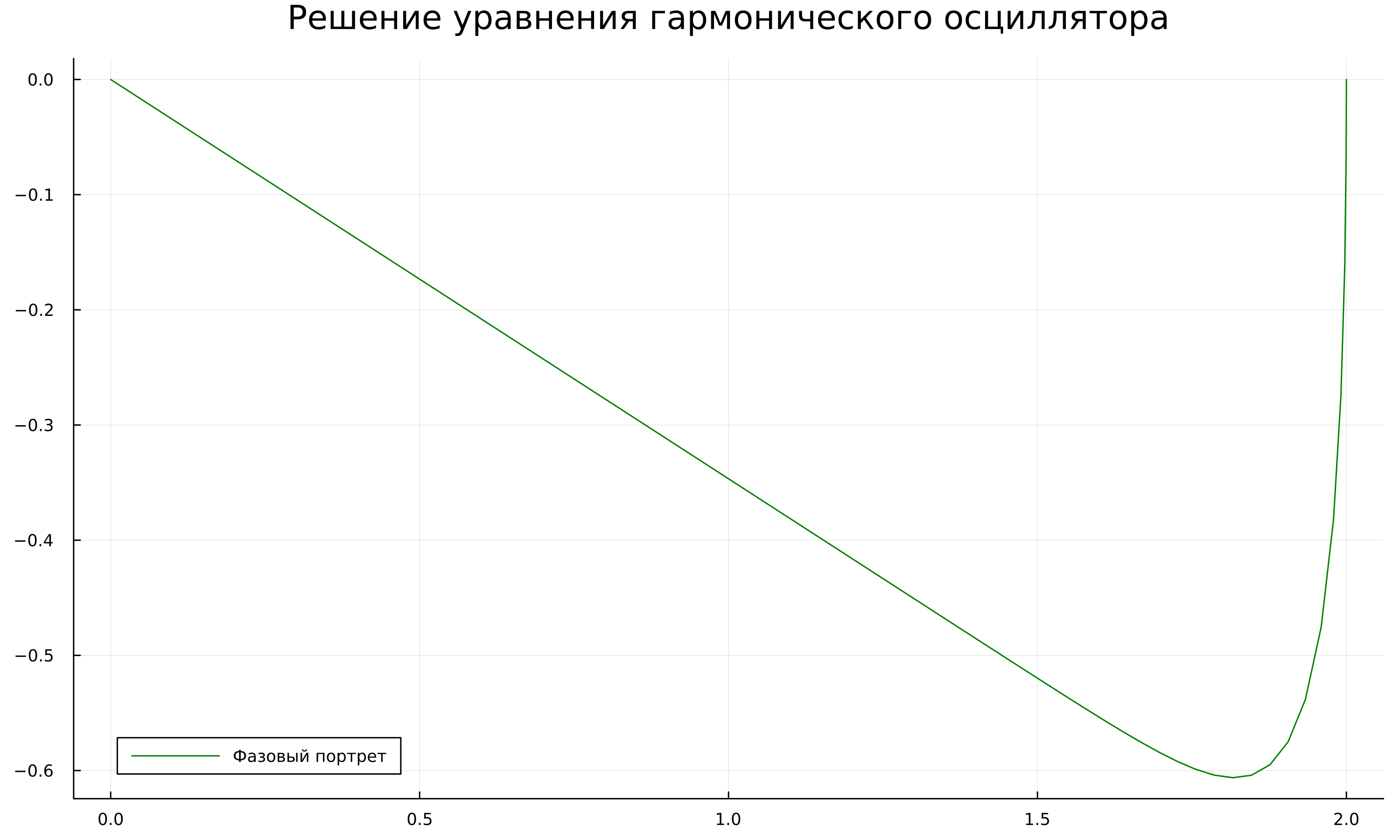
Результат работы программы для второго случая

1. Код для вывода на экран фазового портрета отличается лишь в том, что мы хотим вывести на экран (рис. ??).



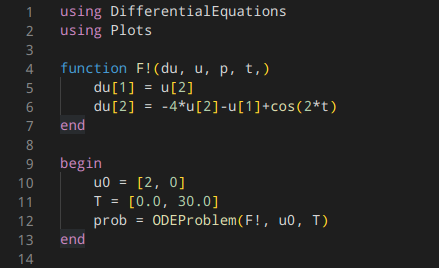
Меняем только последнюю часть кода

1. Получим следующий результат (рис. ??).

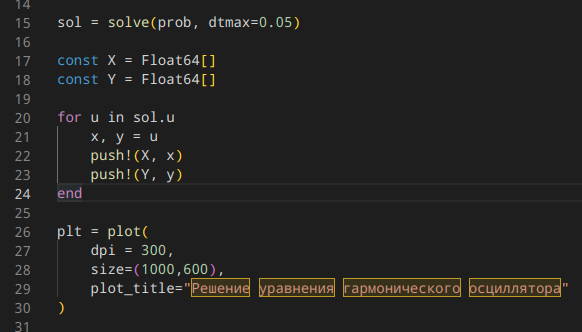


Фазовый портрет для второго случая

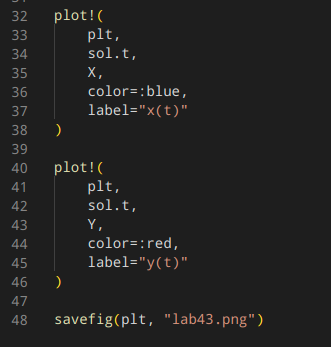
1. Теперь опишем третий случай (рис. ??, ??, ??).



Подключаем библиотеки, задаём функцию, решающую дифференциальные уравнения. Затем зададим начальные условия.

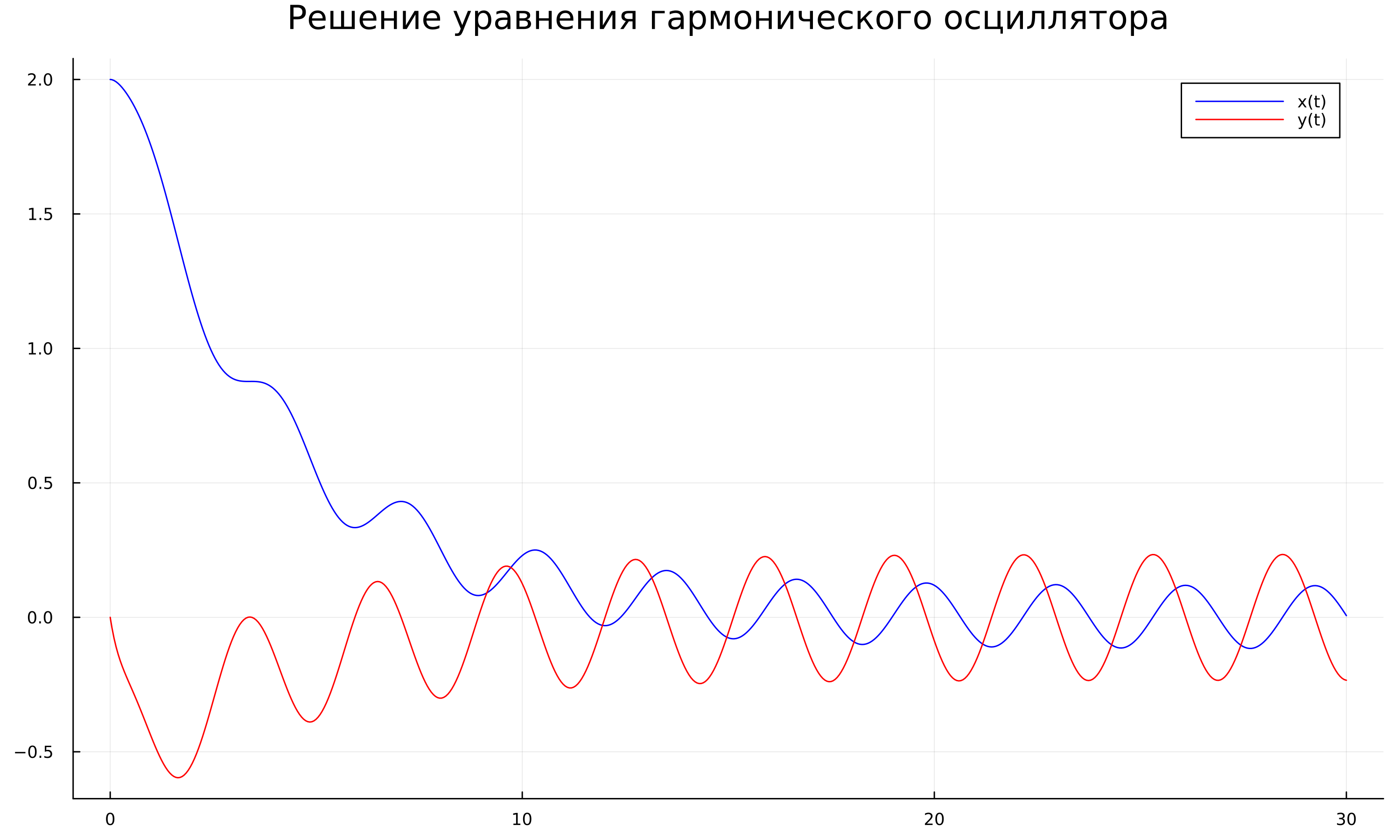


Выполним функцию с данными значениями. Создадим два пустых массива, в которые мы передадим полученные значения. Затем с помощью функционала библиотеки Plots создадим поле для вывода результата.



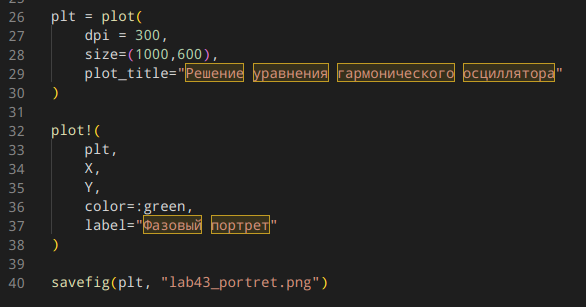
Выведем на экран полученные графы и сохраним результат в формате png

1. Получим следующий результат (рис. ??).



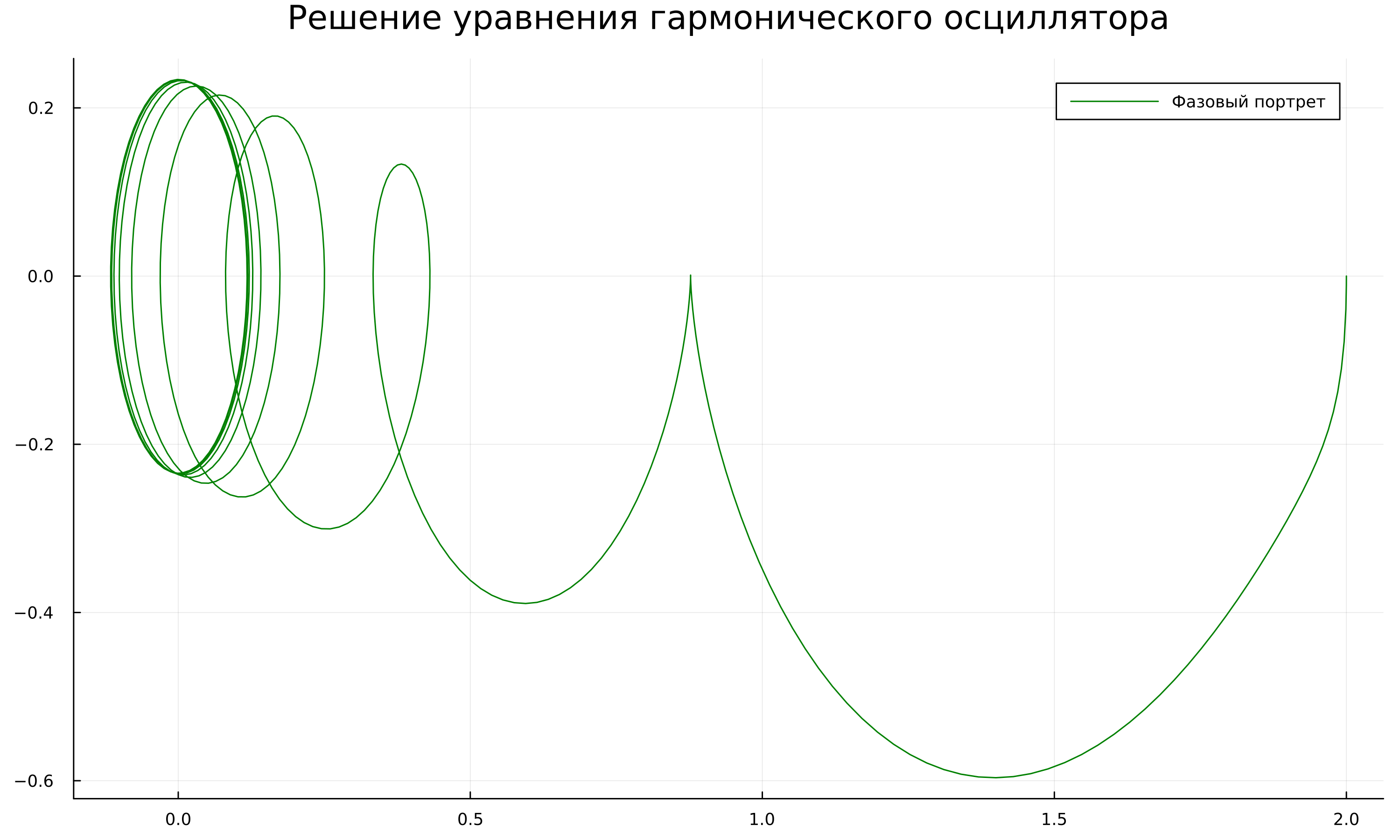
Результат работы программы для третьего случая

1. Код для вывода на экран фазового портрета отличается лишь в том, что мы хотим вывести на экран (рис. ??).



Меняем только последнюю часть кода

1. Получим следующий результат (рис. ??).



Фазовый портрет для третьего случая

# 5 Выводы

Выполнив данную лабораторную работу, мы продолжили знакомство с языками программирования Julia и Modelica. Сравнивая реализацию одной программы на этих двух языках, можно заметить, что реализация на языке Modelica заметно проще и более точно показывает результат, поскольку можно отследить значения переменных с максимальной точностью на любом отрезке графика.

# Список литературы

1. Julia [Электронный ресурс]. Free Software Foundation, 2023. URL: <https://ru.wikipedia.org/wiki/Julia_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)>.

2. OpenModelica [Электронный ресурс]. Free Software Foundation, 2023. URL: <https://ru.wikipedia.org/wiki/OpenModelica>.

3. Гармонический осциллятор [Электронный ресурс]. Free Software Foundation, 2023. URL: <http://fn.bmstu.ru/data-physics/library/physbook/tom5/ch4/texthtml/ch4_5.htm>.