ФИЗИЧЕСКИЕ ВЫЧИСЛЕНИЯ НА ЯЗЫКЕ ПРОГРАММИРОВАНИЯ JULIA

ВОЛКОВ Евгений Валерьевич
старший преподаватель
Оренбургский государственный педагогический университет
г. Оренбург, Россия

В статье приведен опыт использования языка программирования Julia и программного блокнота Pluto для физических расчетов. На примере модели математического маятника сравниваются результаты расчетов для упрощенной модели и результаты численного дифференцирования. Наглядно на графиках демонстрируется различие результатов расчетов. Делается вывод о возможности использования языка Julia и блокнота Pluto в учебной практике.

Ключевые слова: математический маятник, дифференциальное уравнение, Julia, Pluto, расчет, моделирование.

Язык программирования Julia разрабатывается в Массачусетском технологическом университете с 2009 г., распространяется бесплатно и ориентирован в первую очередь на высокопроизводительные вычисления в научно-технических областях (https://docs.julialang.org/en/v1/manual/documentation/).

Блокнот Pluto – интерфейс для работы с Julia использующий веб-браузер (https://github.com/fonsp/Pluto.jl). Код в Pluto записывается в ячейки и выполняется в соответствии с графом зависимостей ячеек друг от друга. После выполнения кода в каждой ячейке результат его выполнения отображается над ячейкой. На страницах блокнота можно сочетать выражения Julia с пояснительным текстом, отформатированным с помощью Markdown или HTML получая, таким образом, документ, содержащий код, результаты его выполнения и их возможную графическую интерпретацию [3].

Для решения дифференциальных уравнений различных типов в Julia используется пакет DifferentialEquation. Далее рассмотрим его применение для решения дифференциального уравнения движения математического маятника. На рисунке 1 показаны длина нити L и начальный угол отклонения θ_0.

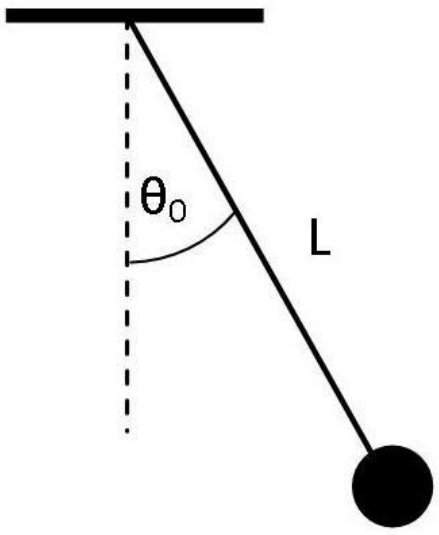

Рисунок 1. **Математический маятник**

Угол отклонения отсчитывается от вертикали против часовой стрелки, вектор ускорения свободного падения направлен вниз.

Дифференциальное уравнение движения маятника на основании второго закона Ньютона имеет вид

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L}\sin(\theta).$$

Данное дифференциальное уравнение второй степени не имеет решения в элементарных функциях. Обычно, ограничиваются малыми углами отклонения ($\theta_0 \leq 5°$), для которых $\sin(\theta) \approx \theta$, в этом случае решением дифференциального уравнения является гармоническая функция.

Для численного решения исходного дифференциального уравнения применим пакет DifferentialEquation, который работает с системами уравнений первой степени. Преобразуем исходное уравнение в систему уравнений первой степени

$$\frac{d\theta}{dt} = \omega$$
$$\frac{d\omega}{dt} = -\frac{g}{L}\sin(\theta).$$

Решениями будут две функции времени: угла отклонения θ(t) и угловой скорости ω(t).

Первый шаг – определение аргументов для DifferentialEquation:

du – массив производных функций решений;

u – массив функций решения;

p – массив параметров;

t – время.

На рисунке 2 приведена запись системы уравнений движения маятника на языке Julia.

```
function mayatnik!(du, u, p, t)
    L, g = p
    θ, ω = u
    du[1] = ω
    du[2] = -g/L * sin(θ)
end
```

Рисунок 2. **Система уравнений движения маятника**

Восклицательный знак у имени функции указывает, что это изменяющаяся функция, так как по мере вычисления изменяются массивы du и u. Чтобы получить решение, необходимо передать параметры в функцию solve(). Ими являются: массив параметров, начальные условия, временной интервал, для которого необходимо решение, и функция, определяющая дифференциальные уравнения. В нашем случае это mayatnik!(). На рисунке 3 приведена запись решения системы дифференциальных уравнений.

```
using DifferentialEquations

p = [1.0, 9.8];
# L g - Параметры (длина нити, ускорение свободного падения)

u0 = [deg2rad(5), 0];
# θ ω - Начальные условия (угол отклонения, угловая скорость)

tspan = (0, 10);
# Прмежуток времени

prob = ODEProblem(mayatnik!, u0, tspan, p);

sol5d = solve(prob);
```

Рисунок 3. **Решение системы дифференциальных уравнений с использованием пакета DifferentialEquation**

Здесь используются две функции из пакета DifferentialEquation: ODEProblem() и solve(). В функцию ODEProblem() передаются четыре аргумента: функция, определяющая систему уравнений mayatnik!, массив начальных условий u0, временной интервал tspan и массив параметров p.

Результат sol5d, возвращаемый функцией ODEProblem(), содержит решения двух функций: угла отклонения θ(t) и угловой скорости ω(t).

Как уже указывалось, для малых углов отклонения аналитическое решение имеет вид гармонической функции:

$$\theta(t) = \theta_0 \sin\left(\sqrt{\frac{g}{L}}t\right),$$

где θ_0 – начальный угол отклонения.

Численное и аналитическое решение можно сверить графически. Для этого построим графики зависимостей (рисунок 4).

```
using Plots

plot(sol5d; idxs=1, lw=4, lc=:lightgrey, label="Численно",
     legend=:outerright, title="θ0 = 5°");

L, g = p;

plot!(t -> u0[1]*cos(sqrt(g/L)*t); xrange=(0, 10),lw=2,
      ls=:dash, lc=:black, label="Аналитически");
```

Рисунок 4. **Построение на одном графике двух решений**

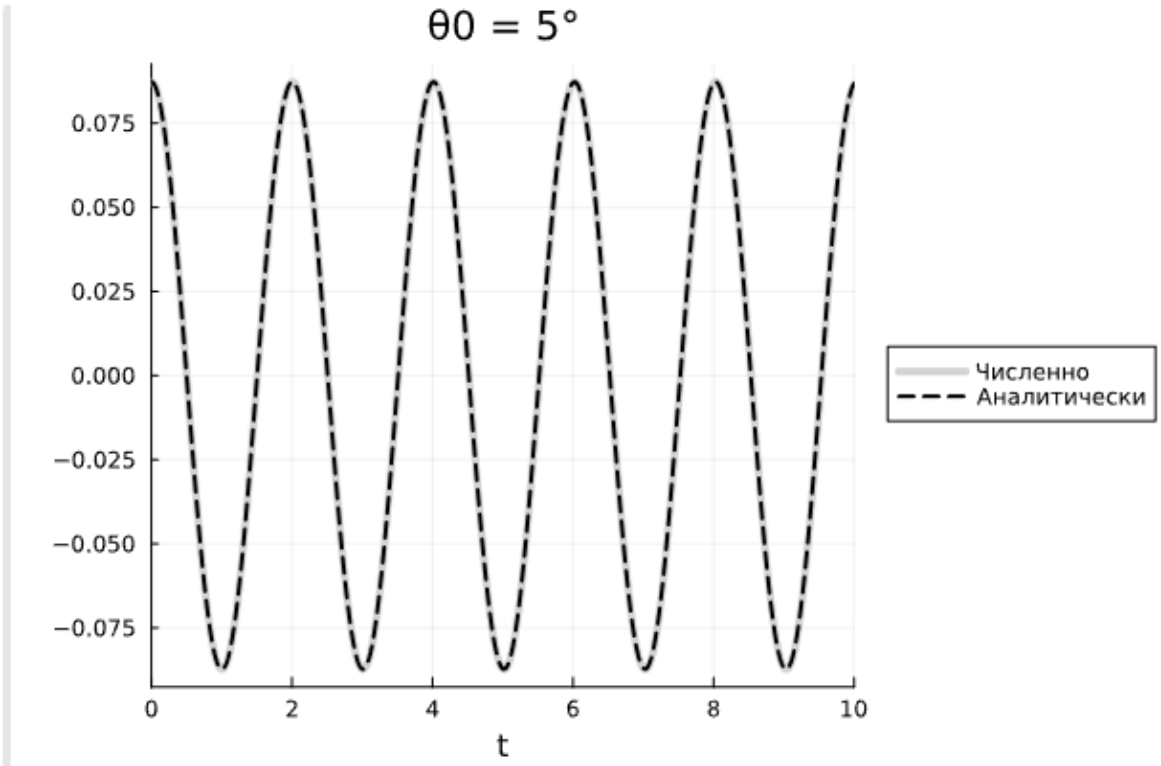

Рисунок 5. **Графики функций угла отклонения θ(t) для численного и аналитического решения при малых углах**

Выполняя вычисления для начального угла отклонения $\theta_0 = 90°$ и строя графики (рисунок 6) видим, что найденная функция приближается к гармонической, но имеет частоту примерно на 25 % меньше, что соответствует экспериментальным данным.

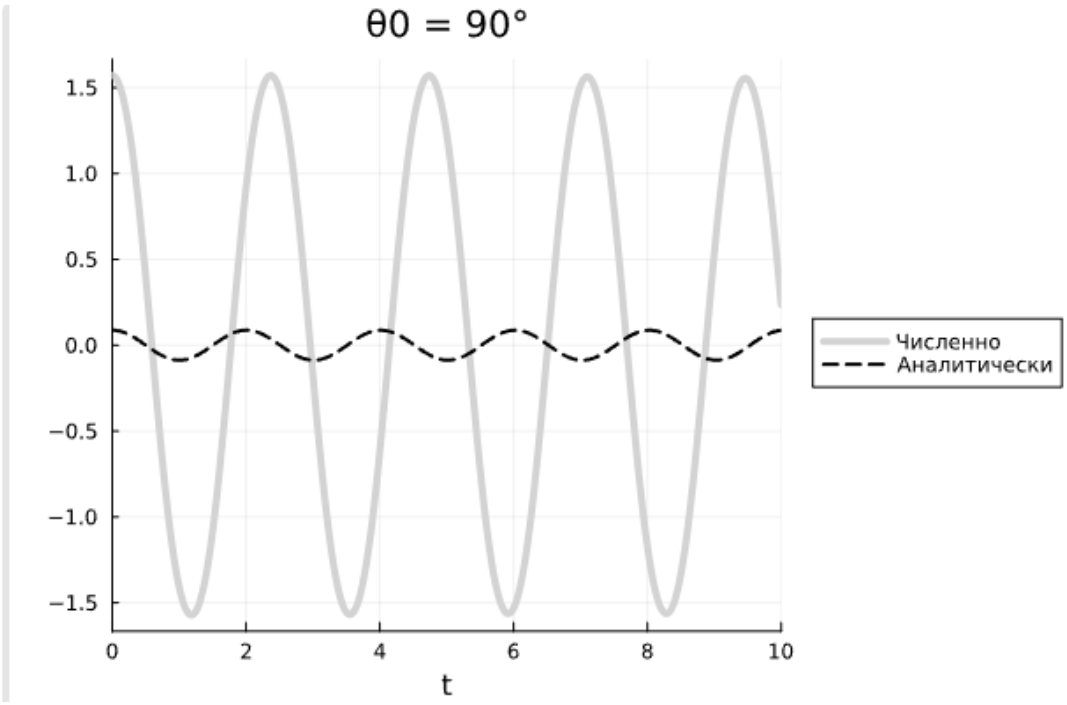

Рисунок 6. **Численное и аналитическое решение при $\theta_0 = 90°$**

На основании проведенного моделирования можно сделать выводы, что, в целом, совместное использование языка программирования Julia и блокнота Pluto позволяет при минимальных навыках программирования получить наглядные документы содержащие исходные данные, программу для расчетов и графический материал.

ЛИТЕРАТУРА

1. *Phillips L.* Practical Julia. A hands-on introduction for scientific minds. No Starch Press 2024. 680 p.

PHYSICAL CALCULATIONS IN THE JULIA PROGRAMMING LANGUAGE

VOLKOV Evgeny Valerievich
Senior lecturer
Orenburg State Pedagogical University
Orenburg, Russia

The article presents the experience of using the Julia programming language and the Pluto software notebook for physical calculations. Using the example of a mathematical pendulum model, the results of calculations for a simplified model and the results of numerical differentiation are compared. The graphs clearly demonstrate the difference in the calculation results. The conclusion is made about the possibility of using the Julia language and the Pluto notebook in educational practice.

Keywords: mathematical pendulum, differential equation, Julia, Pluto, calculation, modeling.