## Методы Рунге-Кутты для численного решения обыкновенных дифференциальных уравнений: реализация и анализ в среде Julia

Аннотация

В данной работе рассматриваются классические методы Рунге-Кутты для численного решения задач Коши для обыкновенных дифференциальных уравнений (ОДУ). Особое внимание уделяется практической реализации методов на современном языке программирования Julia, сочетающем высокую производительность и простоту синтаксиса. Приводятся примеры кода и сравнительный анализ точности на тестовой задаче.

1. Введение

Часто в физике, биологии, экономике и других науках процессы описываются обыкновенными дифференциальными уравнениями (ОДУ). Уравнение вида `dy/dt = f(t, y)` с начальным условием `y(t₀) = y₀` называется задачей Коши. К сожалению, лишь для малой части таких уравнений удается найти аналитическое (точное) решение. На помощь приходят численные методы, которые позволяют найти приближенное решение в виде таблицы значений.

Методы Рунге-Кутты (Runge-Kutta methods, RK) — это одно из самых важных семейств численных методов для решения ОДУ. Их популярность обусловлена высокой точностью, устойчивостью и тем, что они являются явными (значение на следующем шаге вычисляется по известным значениям на предыдущих), что делает их реализацию относительно простой.

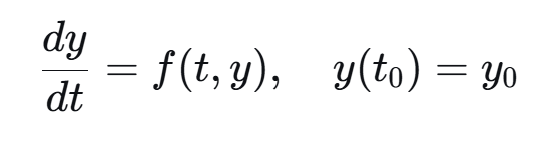
2. Теоретические основы методов Рунге-Кутты

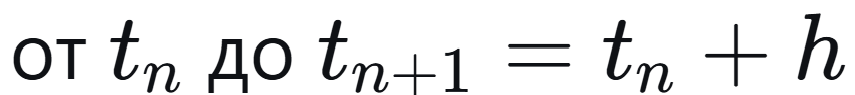
Идея методов Рунге-Кутты схожа с идеей метода Эйлера, но усовершенствована. Если в простейшем методе Эйлера мы делаем один "пробный шаг" в направлении производной, то в методах Рунге-Кутты делается несколько "пробных шагов" внутри одного шага интегрирования, и их результаты усредняются с определенными весами. Это позволяет значительно повысить точность.

Пусть мы находимся в точке `(t\_n, y\_n)` и хотим найти значение `y\_{n+1}` на шаге `h`.

2.1. Метод Рунге-Кутты 4-го порядка (RK4)

Для численного решения задачи Коши:

​

на одном шаге интегрирования  вычисления производятся по следующим формулам:

1. Первый пробный шаг

Изображение выглядит как Шрифт, рукописный текст, каллиграфия, типография

Контент, сгенерированный ИИ, может содержать ошибки.

2. Второй пробный шаг

Изображение выглядит как Шрифт, рукописный текст, линия, белый

Контент, сгенерированный ИИ, может содержать ошибки.

3. Третий пробный шаг

Изображение выглядит как Шрифт, рукописный текст, линия, белый

Контент, сгенерированный ИИ, может содержать ошибки.

4. Четвертый пробный шаг

Изображение выглядит как Шрифт, рукописный текст, текст, типография

Контент, сгенерированный ИИ, может содержать ошибки.

5. Усреднение и получение решения

Изображение выглядит как Шрифт, текст, рукописный текст, белый

Контент, сгенерированный ИИ, может содержать ошибки.

3. Реализация методов Рунге-Кутты в среде Julia

Ниже представлена реализация метода RK4.

Определение функции метода Рунге-Кутты 4-го порядка (RK4)

***function runge\_kutta\_4(f, y0, tspan, h)***

```julia

using Plots

"""

Параметры:

f - Правая часть ОДУ (функция f(t, y))

y0 - Начальное условие

tspan - Кортеж (tuple) с начальным и конечным временем (t\_start, t\_end)

h - Шаг интегрирования

Возвращает:

t - Вектор временных точек

y - Вектор численного решения

"""

t\_start, t\_end = tspan

t = collect(t\_start:h:t\_end) Создание массива времени

n = length(t)

y = similar(t, Float64)

y[1] = y0 Записываем начальное условие

for i in 1:n-1

k1 = h f(t[i], y[i])

k2 = h f(t[i] + h/2, y[i] + k1/2)

k3 = h f(t[i] + h/2, y[i] + k2/2)

k4 = h f(t[i] + h, y[i] + k3)

y[i+1] = y[i] + (k1 + 2k2 + 2k3 + k4) / 6

end

return t, y

end

#Пример: Решение уравнения y' = -2y, с точным решением y(t) = exp(-2t)

f(t, y) = -2 y

y0 = 1.0

tspan = (0.0, 3.0)

h = 0.1

#Вызов метода

t, y\_num = runge\_kutta\_4(f, y0, tspan, h)

#Вычисление точного решения для сравнения

y\_exact = exp.(-2 t)

#Построение графиков

plot(t, y\_exact, label="Точное решение", linewidth=2, legend=:topright)

scatter!(t, y\_num, label="Численное решение (RK4)", markersize=3)

xlabel!("Время, t")

ylabel!("y(t)")

title!("Сравнение численного и точного решения")

```

3.1. Использование встроенных средств Julia

Для практических расчетов удобнее использовать мощные встроенные солверы из пакета `DifferentialEquations.jl`.

```*julia* #Если пакет не установлен, выполним:  
 using Pkg; Pkg.add("DifferentialEquations")  
  
using DifferentialEquations  
  
#Определяем задачу в форме, понятной для солвера  
function ode\_func(du, u, p, t)  
 du[1] = -2 u[1]  
end  
  
u0 = [1.0] # Начальное условие  
tspan = (0.0, 3.0) #Интервал интегрирования  
prob = ODEProblem(ode\_func, u0, tspan) #Определение проблемы ОДУ  
  
# Решение с помощью встроенного солвера Tsit5 (метод типа Рунге-Кутты 5-го порядка)  
sol = solve(prob, Tsit5(), reltol=1e-8, abstol=1e-8)  
  
# Построение решения  
plot(sol, label="Решение Tsit5", linewidth=2)  
xlabel!("t")  
ylabel!("u(t)")  
```

4. Результаты

Реализованный метод RK4 и встроенный солвер `Tsit5` успешно решают тестовое уравнение `y' = -2y`. Визуальное сравнение с точным решением (рис. 1) показывает высокую точность метода RK4 даже при сравнительно большом шаге `h=0.1`. Использование специализированных библиотек, таких как `DifferentialEquations.jl`, позволяет легко менять методы, контролировать точность и решать более сложные задачи (системы уравнений, жесткие задачи).

5. Заключение

Методы Рунге-Кутты, в частности классический метод 4-го порядка, являются надежным и эффективным инструментом для численного интегрирования ОДУ. Их реализация на языке Julia оказывается интуитивно понятной и лаконичной. Сочетание простоты самостоятельной реализации для учебных целей и наличия высокопроизводительных профессиональных солверов делает экосистему Julia исключительно привлекательной для проведения научных вычислений в области математического моделирования.

Список литературы

1. Hairer, E., Nørsett, S. P., & Wanner, G. (1993). Solving Ordinary Differential Equations I: Nonstiff Problems.

2. Rackauckas, C., & Nie, Q. (2017). DifferentialEquations.jl – A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia.

3. Официальная документация по пакету DifferentialEquations.jl.