Метод Рунге — Кутты

Материал из Википедии — свободной энциклопедии

Ме́тоды Ру́нге — Ку́тты (в литературе встречается название **методы Рунге — Кутта**) — большой класс численных методов решения задачи Коши для обыкновенных дифференциальных уравнений и их систем. Первые методы данного класса были предложены около 1900 года немецкими математиками К. Рунге и М. В. Куттой.

К классу методов Рунге — Кутты относятся явный метод Эйлера и модифицированный метод Эйлера с пересчётом, которые представляют собой соответственно методы первого и второго порядка точности. Существуют стандартные явные методы третьего порядка точности, не получившие широкого распространения. Наиболее часто используется и реализован в различных математических пакетах (Maple, MathCAD, Maxima) *классический метод Рунге — Кутты*, имеющий четвёртый порядок точности. При выполнении расчётов с повышенной точностью всё чаще применяются методы пятого и шестого порядков точности[1][2]. Построение схем более высокого порядка сопряжено с большими вычислительными трудностями[3].

Методы седьмого порядка должны иметь по меньшей мере девять стадий, а методы восьмого порядка — не менее 11 стадий. Для методов девятого и более высоких порядков (не имеющих, впрочем, большой практической значимости) неизвестно, сколько стадий необходимо для достижения соответствующего порядка точности[3].

Содержание

Классический метод Рунге — Кутты четвёртого порядка

Метод Рунге — Кутты четвёртого порядка при вычислениях с постоянным шагом интегрирования столь широко распространён, что его часто называют просто методом Рунге — Кутты.

Рассмотрим задачу Коши для системы обыкновенных дифференциальных уравнений первого порядка (далее $\mathbf{y}, \mathbf{f}, \mathbf{k}_i \in \mathbb{R}^n$, а $x, h \in \mathbb{R}^1$).

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}), \quad \mathbf{y}(x_0) = \mathbf{y}_0.$$

Тогда приближенное значение в последующих точках вычисляется по итерационной формуле:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{h}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)$$

Вычисление нового значения проходит в четыре стадии:

$$\begin{aligned}
\mathbf{k}_1 &= \mathbf{f}\left(x_n, \mathbf{y}_n\right), \\
\mathbf{k}_2 &= \mathbf{f}\left(x_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_1\right), \\
\mathbf{k}_3 &= \mathbf{f}\left(x_n + \frac{h}{2}, \mathbf{y}_n + \frac{h}{2}\mathbf{k}_2\right), \\
\mathbf{k}_4 &= \mathbf{f}\left(x_n + h, \mathbf{y}_n + h\,\mathbf{k}_3\right).
\end{aligned}$$

где h — величина шага сетки по x.

Этот метод имеет четвёртый порядок точности. Это значит, что ошибка на одном шаге имеет порядок $O(h^5)$, а суммарная ошибка на конечном интервале интегрирования имеет порядок $O(h^4)$.

Явные методы Рунге — Кутты

Семейство явных методов Рунге — Кутты является обобщением как явного метода Эйлера, так и классического метода Рунге — Кутты четвёртого порядка. Оно задаётся формулами

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\sum_{i=1}^{n} b_i \mathbf{k}_i,$$

где h — величина шага сетки по x, а вычисление нового значения проходит в s этапов:

$$\begin{aligned}
\mathbf{k}_1 &= \quad \mathbf{f}(x_n, \mathbf{y}_n), \\
\mathbf{k}_2 &= \quad \mathbf{f}(x_n + c_2 h, \mathbf{y}_n + a_{21} h \mathbf{k}_1), \\
&\cdots \\
\mathbf{k}_s &= \quad \mathbf{f}(x_n + c_s h, \mathbf{y}_n + a_{s1} h \mathbf{k}_1 + a_{s2} h \mathbf{k}_2 + \cdots + a_{s,s-1} h \mathbf{k}_{s-1})
\end{aligned}$$

Конкретный метод определяется числом s и коэффициентами b_i, a_{ij} и c_i. Эти коэффициенты часто упорядочивают в таблицу (называемую *таблицей Бутчера*):

$$\begin{array}{c|ccccc}
0 & & & & & \\
c_2 & a_{21} & & & & \\
c_3 & a_{31} & a_{32} & & & \\
\vdots & \vdots & \vdots & \ddots & & \\
c_s & a_{s1} & a_{s2} & \ldots & a_{ss-1} & \\
\hline
 & b_1 & b_2 & \ldots & b_{s-1} & b_s
\end{array}$$

Для коэффициентов метода Рунге — Кутты должны быть выполнены условия $\sum_{j=1}^{i-1} a_{ij} = c_i$ для $i = 2, \ldots, s$. Если требуется, чтобы метод имел порядок p, то следует также обеспечить условие

$$\bar{\mathbf{y}}(h + x_0) - \mathbf{y}(h + x_0) = O(h^{p+1}),$$

где $\bar{\mathbf{y}}(h+x_0)$ — приближение, полученное по методу Рунге — Кутты. После многократного дифференцирования это условие преобразуется в систему полиномиальных уравнений относительно коэффициентов метода.

Неявные методы Рунге — Кутты

Все до сих пор упомянутые методы Рунге — Кутты являются явными методами. К сожалению, явные методы Рунге — Кутты, как правило, непригодны для решения жестких уравнений из-за малой области их абсолютной устойчивости[4]. Неустойчивость явных методов Рунге — Кутты создаёт весьма серьёзные проблемы и при численном решении дифференциальных уравнений в частных производных.

Неустойчивость явных методов Рунге — Кутты мотивировала развитие неявных методов. Неявный метод Рунге — Кутты имеет вид[5][6]:

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i,$$

где

$$k_i = f\Big(x_n + c_i h, y_n + h\sum_{j=1}^{s} a_{ij}k_j\Big), \quad i = 1, \ldots, s.$$

Явный метод характерен тем, что матрица коэффициентов a_{ij} для него имеет нижний треугольный вид (включая и нулевую главную диагональ) — в отличие от неявного метода, где матрица имеет произвольный вид. Это также видно по таблице Батчера.

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array} = \begin{array}{c|c} \mathbf{c} & A \\ \hline & \mathbf{b}^{\mathrm{T}} \end{array}$$

Следствием этого различия является необходимость на каждом шагу решать систему уравнений для $k_i, i = 1, 2, \ldots, s$, где s — число стадий. Это увеличивает вычислительные затраты, однако при достаточно малом h можно применить принцип сжимающих отображений и решать данную систему методом простой итерации[7]. В случае одной итерации это увеличивает вычислительные затраты всего лишь в два раза.

С другой стороны, Жан Кунцма́н (1961) и Джон Батчер (1964) показали, что при любом количестве стадий s существует неявный метод Рунге — Кутты с порядком точности $p = 2s$. Это значит, например, что для описанного выше явного четырёхстадийного метода четвёртого порядка существует неявный аналог с вдвое большим порядком точности.

Неявный метод Рунге — Кутты второго порядка

Простейшим неявным методом Рунге — Кутты является модифицированный метод Эйлера «с пересчётом». Он задаётся формулой:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\frac{\mathbf{f}(x_n, \mathbf{y}_n) + \mathbf{f}(x_{n+1}, \mathbf{y}_{n+1})}{2}.$$

Для его реализации на каждом шаге необходимы как минимум две итерации (и два вычисления функции).

Прогноз:

$$\tilde{\mathbf{y}}_{n+1} = \mathbf{y}_n + h\mathbf{f}(x_n, \mathbf{y}_n).$$

Коррекция:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\frac{\mathbf{f}(x_n, \mathbf{y}_n) + \mathbf{f}(x_{n+1}, \tilde{\mathbf{y}}_{n+1})}{2}.$$

Вторая формула — это простая итерация решения системы уравнений относительно $\mathbf{y}_{n+1}$, записанной в форме сжимающего отображения. Для повышения точности итерацию-коррекцию можно сделать несколько раз, подставляя $\tilde{\mathbf{y}}_{n+1} = \mathbf{y}_{n+1}$. Модифицированный метод Эйлера «с пересчётом» имеет второй порядок точности.

Устойчивость

Преимуществом неявных методов Рунге — Кутты в сравнении с явными является их бо́льшая устойчивость, что особенно важно при решении жестких уравнений. Рассмотрим в качестве примера линейное уравнение $y' = \lambda y$. Обычный метод Рунге — Кутты, применённый к этому уравнению, сведётся к итерации $y_{n+1} = r(h\lambda)\, y_n$, с r, равным

$$r(z) = 1 + zb^T(I - zA)^{-1}e = \frac{\det(I - zA + zeb^T)}{\det(I - zA)},$$

где e обозначает вектор-столбец из единиц[8]. Функция r называется *функцией устойчивости*[9]. Из формулы видно, что r является отношением двух полиномов степени s, если метод имеет s стадий. Явные методы имеют строго нижнюю треугольную матрицу A, откуда следует, что $\det(I - zA) = 1$, и что функция устойчивости является многочленом[10].

Численное решение данного примера сходится к нулю при условии $|r(z)| < 1$ с $z = h\lambda$. Множество таких z называется *областью абсолютной устойчивости*. В частности, метод называется *A-устойчивым*, если все z с $\mathrm{Re}(z) < 0$ находятся в области абсолютной устойчивости. Функция устойчивости явного метода Рунге — Кутты является многочленом, поэтому явные методы Рунге — Кутты в принципе не могут быть A-устойчивыми[10].

Если метод имеет порядок p, то функция устойчивости удовлетворяет условию $r(z) = \mathrm{e}^z + O(z^{p+1})$ при $z \to 0$. Таким образом, представляет интерес отношение многочленов данной степени, приближающее экспоненциальную функцию наилучшим образом. Эти отношения известны как аппроксимации Паде. Аппроксимация Паде с числителем степени m и знаменателем степени n A-устойчива тогда и только тогда, когда $m \leq n \leq m + 2$.[11]

s-стадийный метод Гаусса — Лежандра имеет порядок $2s$, поэтому его функция устойчивости является приближением Паде $m = n = s$. Отсюда следует, что метод является A-устойчивым[12]. Это показывает, что A-устойчивые методы Рунге — Кутты могут иметь сколь угодно высокий порядок. В отличие от этого, порядок A-устойчивости методов Адамса не может превышать два.

Произношение

Согласно грамматическим нормам русского языка, фамилия Ку́тта склоняется, поэтому говорят: «Метод Ру́нге — Ку́тты». Правила русской грамматики предписывают склонять все фамилии (в том числе и мужские), оканчивающиеся на -а, -я, которым предшествует согласный. Единственное исключение — фамилии французского происхождения с ударением на последнем слоге типа Дюма́, Золя́[13]. Однако иногда встречается несклоняемый вариант «Метод Ру́нге — Ку́тта» (например, в книге[14]).

Пример решения на алгоритмических языках программирования

$$y'' + 4y = \cos 3x, \quad y(0) = 0.8,\ y'(0) = 2,\ x \in [0, 1],\ h = 0.1$$

производя замену $y' = z$ и перенося $4y$ в правую часть, получаем систему:

$$\begin{cases} y' = z = g(x, y, z) \\ z' = \cos(3x) - 4y = f(x, y, z) \end{cases}$$

код на Java для решения системы дифференциальных уравнений методом Рунге-Кутты [показать]

Код на языке C# [скрыть]

```
using System;
using System.Collections.Generic;

namespace PRJ_RungeKutta
{
    /// <summary>
    /// Реализация метода Ру́нге — Ку́тты для обыкновенного дифференциального уравнения
    /// </summary>
    public abstract class RungeKutta
    {
        /// <summary>
        /// Текущее время
        /// </summary>
        public double t;
        /// <summary>
        /// Искомое решение Y[0] — само решение, Y[i] — i-я производная решения
        /// </summary>
        public double[] Y;
        /// <summary>
        /// Внутренние переменные
        /// </summary>
        double[] YY, Y1, Y2, Y3, Y4;
        protected double[] FY;
        /// <summary>
        /// Конструктор
        /// </summary>
        /// <param name="N">размерность системы</param>
        public RungeKutta(uint N)
        {
            Init(N);
        }
        /// <summary>
        /// Конструктор
        /// </summary>
        public RungeKutta(){}
        /// <summary>
        /// Выделение памяти под рабочие массивы
        /// </summary>
        /// <param name="N">Размерность массивов</param>
        protected void Init(uint N)
        {
            Y = new double[N];
            YY = new double[N];
            Y1 = new double[N];
            Y2 = new double[N];
            Y3 = new double[N];
            Y4 = new double[N];
            FY = new double[N];
        }
        /// <summary>
        /// Установка начальных условий
        /// </summary>
        /// <param name="t0">Начальное время</param>
        /// <param name="Y0">Начальное условие</param>
```

```
        public void SetInit(double t0, double[] Y0)
        {
            t = t0;
            if (Y == null)
                Init((uint)Y0.Length);
            for (int i = 0; i < Y.Length; i++)
                Y[i] = Y0[i];
        }
        /// <summary>
        /// Расчет правых частей системы
        /// </summary>
        /// <param name="t">текущее время</param>
        /// <param name="Y">вектор решения</param>
        /// <returns>правая часть</returns>
        abstract public double[] F(double t, double[] Y);
        /// <summary>
        /// Следующий шаг метода Рунге-Кутта
        /// </summary>
        /// <param name="dt">текущий шаг по времени (может быть переменным)</param>
        public void NextStep(double dt)
        {
            int i;

            if (dt < 0) return;

            // рассчитать Y1
            Y1 = F(t, Y);

            for (i = 0; i < Y.Length; i++)
                YY[i] = Y[i] + Y1[i] * (dt / 2.0);

            // рассчитать Y2
            Y2 = F(t + dt / 2.0, YY);

            for (i = 0; i < Y.Length; i++)
                YY[i] = Y[i] + Y2[i] * (dt / 2.0);

            // рассчитать Y3
            Y3 = F(t + dt / 2.0, YY);

            for (i = 0; i < Y.Length; i++)
                YY[i] = Y[i] + Y3[i] * dt;

            // рассчитать Y4
            Y4 = F(t + dt, YY);

            // рассчитать решение на новом шаге
            for (i = 0; i < Y.Length; i++)
                Y[i] = Y[i] + dt / 6.0 * (Y1[i] + 2.0 * Y2[i] + 2.0 * Y3[i] + Y4[i]);

            // рассчитать текущее время
            t = t + dt;
        }
    }
    class TMyRK : RungeKutta
    {
        public TMyRK(uint N) : base(N) { }

        /// <summary>
        /// пример математический маятник
        /// y''(t)+y(t)=0
        /// </summary>
        /// <param name="t">Время</param>
        /// <param name="Y">Решение</param>
        /// <returns>Правая часть</returns>
        public override double[] F(double t, double[] Y)
        {
            FY[0] =  Y[1];
            FY[1] = -Y[0];
            return FY;
        }
        /// <summary>
        /// Пример использования
        /// </summary>
        static public void Test()
        {
            // Шаг по времени
            double dt = 0.001;
            // Объект метода
            TMyRK task = new TMyRK(2);
            // Определим начальные условия y(0)=0, y'(0)=1 задачи
            double[] Y0 = { 0, 1 };
            // Установим начальные условия задачи
            task.SetInit(0, Y0);
```

```
            // решаем до 15 секунд
            while (task.t <= 15)
            {
                Console.WriteLine("Time = {0:F5}; Func = {1:F8};  d Func / d x = {2:F8}", task.t, task.Y[0], task.Y[1]);
// вывести t, y, y'
                // рассчитать на следующем шаге, шаг интегрирования
                task.NextStep(dt);
            }
            Console.ReadLine();
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            TMyRK.Test();
        }
    }
}
```

В программе на C# используется абстрактный класс RungeKutta, в котором следует переопределить абстрактный метод F, задающий правые части уравнений.

Пример решения в среде MATLAB

Решение систем дифференциальных уравнений методом Рунге — Кутты является одним из самых распространённых численных методов решений в технике. В среде MATLAB реализована его одна из разновидностей — метод Дормана — Принса. Для решения системы уравнений необходимо сначала записать функцию, вычисляющую производные, то есть функции $y = g(x,y,z)$ и $z = \cos(3x) - 4y = f(x,y,z)$, о чём сказано выше. В одном из каталогов, к которому имеется доступ из системы MATLAB, нужно создать текстовый файл с именем (например) *runge.m* со следующим содержимым (для MATLAB версии 5.3):

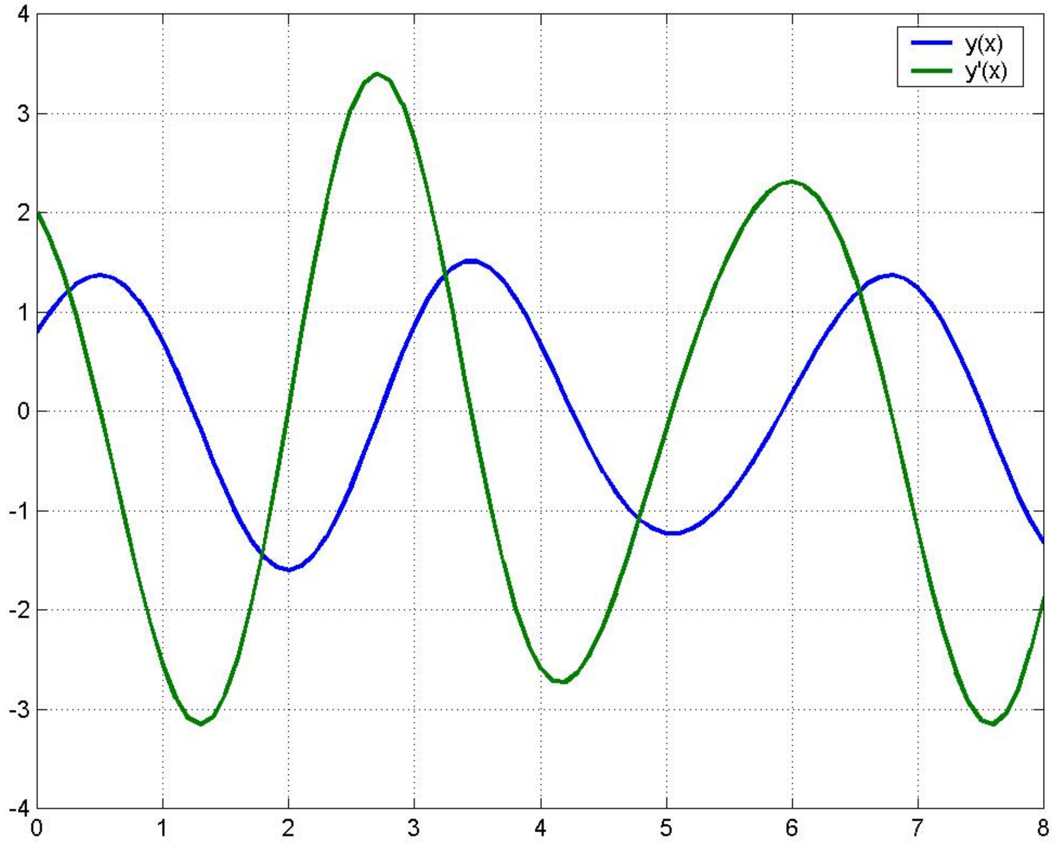
MATLAB, runge.m [показать]

Имя файла и имя функции должно совпадать, но оно может быть любым неиспользуемым ранее.

Затем необходимо создать главный файл с именем, например, *main.m*, который будет выполнять основные вычисления. Этот главный файл будет содержать следующий текст:

MATLAB, main.m [показать]

Так как MATLAB ориентирован на работу с матрицами, решение по методу Рунге — Кутты очень легко выполняется для целого ряда x, как, например, в приведённом примере программы. Здесь решение — график функции в пределах времён от *0* до *x_fin*.



Решение в среде MATLAB

Переменные *x* и *y*, полученные в результате работы функции *ODE45*, есть векторы значений. Очевидно, что решение конкретно заданного выше примера — второй элемент *x*, так как первое значение — 0, шаг интегрирования $h = 0{,}1$, а интересующее значение $x = 0{,}1$. Следующая запись в командном окне MATLAB даст искомое решение:

MATLAB, решение [показать]

Ответ: y1 = 0,98768

Примечания

1. *Бахвалов Н. С., Жидков Н. П., Кобельков Г. М.* . Численные методы. — М.: Лаборатория Базовых Знаний, 2001. — 630 с. — ISBN 5-93208-043-4. — С. 363—375.
2. *Ильина В. А., Силаев П. К.* . Численные методы для физиков-теоретиков. Ч. 2. — Москва-Ижевск: Институт компьютерных исследований, 2004. — 118 с. — ISBN 5-93972-320-9. — С. 16—30.
3. *Butcher J. C.* . Numerical Methods for Ordinary Differential Equations. — New York: John Wiley & Sons, 2008. — ISBN 978-0-470-72335-7.
4. Süli & Mayers, 2003, p. 349—351.
5. Iserles, 1996, p. 41.
6. Süli & Mayers, 2003, p. 351—352.
7. Неявные методы Рунге — Кутты (HTTP://WWW.ASTRO.TSU.RU/CHINTODY/TEXT/3_7.HTML) Архивная копия (https://web.archive.org/web/20190306230951/http://www.astro.tsu.ru/CHINTODY/TEXT/3_7.HTML) от 6 марта 2019 на Wayback Machine.
8. Hairer & Wanner, 1996, p. 40—41.
9. Hairer & Wanner, 1996, p. 40.
10. Iserles, 1996, p. 60.
11. Iserles, 1996, p. 62—63.
12. Iserles, 1996, p. 63.

13. Как склонять фамилии (трудные случаи) — «Грамота.ру» — справочно-информационный Интернет-портал «Русский язык» (http://new.gramota.ru/spravka/letters/71-rubric-482). Дата обращения: 5 июля 2016. Архивировано (https://web.archive.org/web/20180523112050/http://new.gramota.ru/spravka/letters/71-rubric-482) 23 мая 2018 года.
14. *Демидович Б. П., Марон И. А., Шувалова Э. З.* . Численные методы анализа. 3-е изд. — М.: Наука, 1967.

Литература

- *Hairer E., Wanner G.* . Solving ordinary differential equations II: Stiff and differential-algebraic problems. 2nd ed. — Berlin, New York: Springer-Verlag, 1996. — ISBN 978-3-540-60452-5.
- *Iserles A.* . A First Course in the Numerical Analysis of Differential Equations. — Cambridge: Cambridge University Press, 1996. — ISBN 978-0-521-55655-2.
- *Süli E., Mayers D.* . An Introduction to Numerical Analysis. — Cambridge: Cambridge University Press, 2003. — ISBN 0-521-00794-1.