Вестник Ивановского государственного университета

Серия «Биология, Химия, Физика, Математика»

Вып. 3 / 2000 С. 155 – 164

С. И. Хашин

ЧИСЛЕННОЕ РЕШЕНИЕ УРАВНЕНИЙ БУТЧЕРА

В статье описывается программа, находящая решения системы уравнений Бутчера (нахождения методов Рунге – Кутта). Описываются полученные методы Рунге – Кутта, часть из них приведена полностью.

The program for numerical solution of Butcher equations (for obtaining Runge-Kutta methods) is described. Some received Runge-Kutta methods of high order are fulfilled.

УДК 513.6; ББК 22.161.61.

1. Введение

Методы Рунге-Кутты (РК), используемые для численного решения обыкновенных дифференциальных уравнение, характеризуются двумя основными параметрами (p, n) — порядком метода и количеством шагов соответственно. Естественно, что при фиксированном порядке (точности) метода, количество шагов хотелось бы минимизировать. Классические методы РК имеют параметры (4,4). К настоящему времени ([1], [2]) известны методы с параметрами (5,6), (6,7), (7,9) (8,13) и некоторые другие.

Каждый n-шаговый метод РК порядка p задается нижнетреугольной $n \times n$-матрицей $\tilde{A}$ вида:

$$\tilde{A} = \begin{pmatrix} 0 & 0 & \dots & 0 \\ a_{21} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & 0 \end{pmatrix},$$

и вектором $b = (b_1, \dots, b_n)$ длины n. Их можно объединить в одну расширенную матрицу A размера $(n+1) \times (n+1)$:

$$A = \begin{pmatrix} 0 & 0 & \dots & 0 \\ a_{21} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ b_1 & b_2 & \dots & 0 \end{pmatrix}$$

Классические методы порядка 4 (расширенные матрицы):

$$A_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1/2 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1/6 & 2/6 & 2/6 & 1/6 & 0 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 0 \\ -1/3 & 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 \\ 1/8 & 3/8 & 3/8 & 1/8 & 0 \end{pmatrix}.$$

Для нахождения n-шагового метода РК требуется найти $\mathbb{R}$-матрицу A, коэффициенты которой удовлетворят некоторой системе нелинейных (полиномиальных) уравнений. В практически важных случаях ($p = 7, \ldots, 10$) мы получаем систему от нескольких десятков (или даже сотен) переменных и еще большего количества уравнений. В настоящее время не существует общих способов ее решения. Нахождение каждого отдельного решения является большой проблемой.

Выписать получающуюся систему довольно трудно. Эффективный способ описания этих уравнений был предложен Джоном Бутчером [3, 4]. Он сопоставляет каждому отмеченному дереву одно и только одно уравнение на коэффициенты матрицы A.

Аналитическое решение уравнений Бутчера оказывается чрезвычайно сложным делом и в общем виде уравнения до сих пор не решены, известны лишь отдельные решения. Поэтому имеет смысл попытаться решить их численным методом (Ньютона).

Итак, мы имеем систему нелинейных (полиномиальных) уравнений

$$\begin{aligned} f_1(x_1, \ldots, x_p) &= 0, \\ \ldots&\ldots\ldots\ldots \\ f_q(x_1, \ldots, x_p) &= 0. \end{aligned}$$

Уравнения f_i можно выписать явно, но они оказываются черезчур громоздкими. На первый взгляд, при современных компьютерах и современных системах компьютерной алгебры это не составляет большой проблемы. Например, в систему "Mathematica" версии 2 включен в качестве стандартного специальный пакет для нахождения этих уравнений и некоторых их разновидностей. Но уже для методов порядка 6 точности размер системы уравнений (в текстовом виде) превышает мегабайт. Для более высоких порядков размер уравнений получится больше еще на много порядков. Ясно, что сколько-нибудь эффективная работа с такими уравнениями не возможна. Поэтому придется считать, что уравнения f_i заданы не аналитически, а лишь численно, т. е. у нас нет полной формулы, а только некоторая процедура (кстати, рекурсивная) нахождения значений функций f_i в заданной точке. Это означает, что частные производные (df_i/dx_j) придется находить численно, что так же вносит дополнительное усложнение в расчеты.

На каждом шаге метода Ньютона вектор смещения находится из некоторой системы линейных уравнений $Dx = r$ где $D = (df_i/dx_j)$. Количество уравнений может быть как больше, так и меньше количества переменных, эти числа почти никогда не совпадают. Поэтому будем искать такой вектор x, что бы отклонение $\|Dx - r\|$ было минимальным. Это сводится к решению другой системы линейных уравнений

$$(D^t D)(x) = D^t(r),$$

где символом D^t обозначена транспонированная матрица. Так как решений может быть бесконечно много, проще всего было бы потребовать в этом случае взять наименьшее по модулю решение. Но это оказывается вычислительно довольно сложно. Эффективной заменой этому требованию оказывается следующее. Выберем некоторое достаточно малое число ε (например, 10^{-12}) и потребуем минимизации величины

$$\|Dx - r\| + \varepsilon\|x\|.$$

Эта задача приводит к решению системы линейных уравнений

$$(D^t D + \varepsilon E)(x) = D^t(r).$$

Ввиду большого размера системы и плохой сходимости метода Ньютона, для достижения цели приходится использовать все возможности как-то сократить уравнения. Большую роль играют упрощающие предположения нескольких видов. Поэтому рассмотрим их более подробно. Пусть матрица A имеет вид:

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ a_{21} & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ a_{31} & a_{32} & 0 & 0 & \dots & 0 & 0 & 0 \\ a_{41} & a_{42} & a_{43} & 0 & \dots & 0 & 0 & 0 \\ \multicolumn{8}{c}{\dotfill} \\ a_{n,1} & a_{n,2} & a_{n,3} & & \dots & a_{n,n-1} & 0 & 0 \\ b_1 & b_2 & b_3 & b_4 & \dots & b_{n-1} & b_n & 0 \end{pmatrix}.$$

Классическое упрощающее предположение [2]: $B^2 d = (e - Ae) * Bd$:

$$\begin{array}{ccccc} (1) & b_n a_{n1} + b_{n-1}a_{n-1,1} + \dots + b_2 a_{21} & = & b_1 \\ (2) & b_n a_{n2} + b_{n-1}a_{n-1,2} + \dots + b_3 a_{32} & = & (1-c_2)b_2 \\ (3) & b_n a_{n3} + b_{n-1}a_{n-1,3} + \dots + b_4 a_{43} & = & (1-c_3)b_3 \\ (4) & b_n a_{n4} + b_{n-1}a_{n-1,4} + \dots + b_5 a_{54} & = & (1-c_4)b_4 \\ \multicolumn{4}{c}{\dotfill} \\ (n-2) & b_n a_{n,n-2} + b_{n-1}a_{n-1,n-2} & = & (1-c_{n-2})b_{n-2} \\ (n-1) & b_n a_{n,n-1} & = & (1-c_{n-1})b_{n-1} \\ (n) & 0 & = & (1-c_n)b_n \\ (n+1) & 0 & = & 0 \end{array}$$

Упрощающее предположение уровня 3: $(Ae * Ae - 2A^2e) * Bd = 0$. Это означает, что для каждого индекса соответствующая координата у одного из векторов $(Ae * Ae - 2A^2e)$ и Bd должна быть равна 0. Первая координата у первого вектора и последняя у второго всегда равны 0. Как показывают численные эксперименты, чаще всего у вектора Bd равны нулю координаты b_2, b_3 а все остальные координаты равны нулю у вектора $(Ae * Ae - 2A^2e)$. В программе, рассмотренной ниже, этот случай носит название "режим 3". С учетом этого упрощающее предположение уровня 3 выглядит так:

$$
\begin{array}{cccc}
(1) & 0 & = & 0 \\
(2) & b_2 & = & 0 \\
(3) & b_3 & = & 0 \\
(4) & c_4^2/2 & = & a_{42}c_2 + a_{43}c_3 \\
(5) & c_5^2/2 & = & a_{52}c_2 + a_{53}c_3 + a_{54}c_4 \\
\multicolumn{4}{c}{\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots} \\
(n-1) & c_{n-1}^2/2 & = & a_{n-1,2}c_2 + a_{n-1,3}c_3 + \dots + a_{n-1,n-2}c_{n-2} \\
(n) & c_n^2/2 & = & a_{n2}c_2 + a_{n3}c_3 + \dots + a_{n,n-1}c_{n-1} \\
(n+1) & c_{n+1}^2/2 & = & b_2c_2 + b_3c_3 + \dots + b_nc_n
\end{array}
$$

2. Описание программы

Все эти идеи были реализованы с помощью программы, описанной ниже. Она была написана на языке Паскаль для IBM PC (Turbo Pascal, TMT Pascal). По смыслу задачи, стандартной памяти 640K байт должно быть вполне достаточно. Поэтому ее можно написать так, что бы она работала на любом IBM-совместимом компьютере, вплоть до XT. Тем не менее, ввиду большого объема вычислений, все реальные результаты были получены на процессоре класса P333 Mhz.

Очень просто было определиться с типом действительных чисел. В Паскале их имеется несколько длиной от 4 до 10 байтов. Программе приходится иметь дело с сильно вырожденными матрицами. Точнее говоря, надо обращать матрицы, близкие к сильно вырожденным. При этом следует ожидать значительной потери точности. Поэтому действительные числа должны иметь максимальную длину, т. е. быть типа extended, длина 10 байтов, 64 двоичных знака после запятой, около 19 десятичных знаков после запятой. Кроме того, этот тип естественен для данного процессора, и обрабатывается им с максимальной скоростью. Таким образом, мы будем вести вычисления с 19 знаками после запятой. Метод будем считать найденным, если суммарная невязка для системы уравнений окажется меньше 10^{-19}.

Предлагается три режима работы — полный ($regim = 1$), с использованием упрощающего предположения ($regim = 2$) и двух упрощающих предположений ($regim = 3$). Допустимый порядок для искомого метода Рунге-Кутта — от 4 до 9. Количество уравнений в зависимости от режима и порядка указано в следующей таблице.

p	$r = 1$	$r = 2$	$r = 3$
4	8	3	–
5	17	8	6
6	37	19	13
7	85	47	32
8	200	114	79
9	486	285	202

Допустимое количество шагов — от 4 до 17. Количество переменных в зависимости от режима и порядка указано в следующей таблице.

n	$r=1$	$r=2$	$r=3$
4	10	5	–
5	15	9	6
6	21	14	10
7	28	20	15
8	36	27	21
9	45	35	28
10	55	44	36
11	66	54	45
12	78	65	55
13	91	77	66
14	105	90	78
15	120	104	91
16	136	119	105
17	153	135	120

Поскольку поведение метода Ньютона плохо предсказуемо, приходится отслеживать различные возможные причины завершения работы:
— исчерпан максимум шагов,
— одна из переменных стала черезчур велика,
— нажата клавиша Esc,
— черезчур большая производная,
— черезчур большая невязка,
— за много шагов невязка почти не уменьшилась.

Здесь стоит пояснить последнюю причину завершения работы. При поиске методов порядка 7 и выше, типичная ситуация такова: невязка быстро (за несколько десятков шагов) убывает до некоторого небольшого значения (порядка $10^{-7} - 10^{-9}$), после этого практически застывает на месте. Наиболее приемлемый выход из такой ситуации — начать вычисления заново, с новыми начальными (случайными) данными. Более точно, если за 50 шагов невязка не уменьшилась хотя бы на 5%, метод Ньютона завершается.

Начальное приближение может быть как случайными, так и взятым из файла. Результаты работы также записываются в текстовый файл в виде, пригодном для дальнейшей обработки: их можно использовать в качестве начальных данных для дальнейших расчетов.

Для 16-шаговых методов порядка 9 (119 переменных, 285 уравнений) время расчета на процессоре P150 составляет около 5 секунд на составление матрицы частных производных и около 15 секунд на решение системы линейных уравнений. Всего на один шаг метода Ньютона уходит около 20 секунд.

Основная проблема состоит в том, что система уравнений сильно вырождена. Результат этого можно увидеть уже на примере поведения метода Ньютона для простой системы уравнений

$$f_1(x,y) = x^2 - y^3 = 0,$$
$$f_2(x,y) = x^3 - y^4 = 0,$$

в окрестности решения $(x,y) = (0,0)$. Нетрудно проверить, что сходимость метода к 0 будет крайне медленной.

Аналогичные проблемы возникают и при рассмотрении системы уравнений Бутчера. Для нахождения метода четвертого порядка требуется 4-8 ньютоновских итераций, для метода порядка 5 — от 10 до 100. Для методов порядков 7, 8 и 9 требуемое количество итераций оказывается равным многим тысячам. При этом, далеко не каждое начальное случайное приближение приводит к результату. Тем не менее, на процессоре P333 в течении суток можно найти несколько методов порядка 8.

Следует отметить, что основным параметром компьютера, влияющим на время расчета, является скорость работы процессора. Обращения к диску происходят достаточно редко. Для работы программы памяти 640K байт оказывается вполне достаточно, по крайней мере для методов порядка не более 9.

3. Полученные результаты

1. Для контроля были найдены несколько методов порядка 4. Все они с хорошей точностью (10^{-18}) ложатся на известное двумерное многообразие решений [2].

2. Было найдено 16 шестишаговых методов порядка 5. Если вести вычисления в режиме 1, т. е. без использования упрощающих предположений, мы получаем систему 17 уравнений от 21 переменной. В режиме 2 — 8 уравнений от 14 переменных.

$a(2,1) = 0.296802166060682649;$
$a(3,1) = 0.035185538759093851;$
$a(3,2) = 0.406011868192878156;$
$a(4,1) = -0.032247544960809447;$
$a(4,2) = 0.131853864155749628;$
$a(4,3) = 0.404464154665595844;$
$a(5,1) = 0.151457352147073521;$
$a(5,2) = 0.263132523242443512;$
$a(5,3) = 0.194146025110258676;$
$a(5,4) = 0.362098291831739548;$
$a(6,1) = 0.163552899044345702;$
$a(6,2) = 0.208281681993355479;$
$a(6,3) = 0.322638585809491891;$
$a(6,4) = 0.347221372596309083;$
$a(6,5) = -0.041694539443502156;$
$b(1) = 0.110460204922505662;$
$b(2) = 0.265380007426228863;$
$b(3) = 0.111427106454495828;$
$b(4) = 0.227876997528701984;$
$b(5) = 0.947975162345393397;$
$b(6) = -0.663119478677325735.$

3. Метод Ньютона для нахождения 7-шаговых методов порядка 6 сходится сравнительно быстро. Поэтому без особого труда удалось найти несколько десятков методов этого типа. В режиме 1, мы имеем систему 37 уравнений от 28 переменных. В режиме 2 — 19 уравнений от 20

переменных. Таким образом, заранее видно, что получающаяся система уравнений будет сильно вырождена.

7-шаговый метод порядка 6

$a(2,1) = 0.032950806900670306;$	$a(6,5) = 0.202062918189024132;$
$a(3,1) = -0.411584693622404125;$	$a(7,1) = 0.414539595247245122;$
$a(3,2) = 0.612493385723656891;$	$a(7,2) = -0.551470347425987166;$
$a(4,1) = 0.563477177063214707;$	$a(7,3) = 0.804062355993411875;$
$a(4,2) = -0.608900484970971998;$	$a(7,4) = -0.258966183098249996;$
$a(4,3) = 0.382304985278695170;$	$a(7,5) = -0.348186057334349970;$
$a(5,1) = 0.567785851219958206;$	$a(7,6) = 0.940020636617930134;$
$a(5,2) = -0.085484529451268134;$	$b(1) = 0.075453719927331703;$
$a(5,3) = -1.338471156993922230;$	$b(2) = 0.000000000000000000;$
$a(5,4) = 1.576314578099124990;$	$b(3) = 0.169935185641897970;$
$a(6,1) = -0.362575243900385067;$	$b(4) = 0.275351642622906629;$
$a(6,2) = 0.404785391464724318;$	$b(5) = 0.109077541050097745;$
$a(6,3) = 0.385980085612906542;$	$b(6) = 0.289719945682532925;$
$a(6,4) = 0.108681261812069064;$	$b(7) = 0.080461965075233028.$

Помимо 7-шаговых методов порядка 6 были найдены так же 8- и 9-шаговые того же порядка. Но они вряд ли представляют особый интерес с практической точки зрения: 7-шаговые методы предпочтительнее.

4. Для методов Рунге-Кутта порядка 7 наименьшее количество шагов равно 9. В режиме 1, мы имеем систему 85 уравнений от 45 переменных. В режиме 2 — 47 уравнений от 35 переменных. В режиме 3 — 32 уравнения от 28 переменных. Из-за сильной вырожденности системы уравнений сходимость очень плохая — каждый раз для нахождения решения требовалось несколько тысяч шагов. Тем не менее было найдено 13 различных решений этой системы уравнений.

Для методов Рунге-Кутта порядка 8 наименьшее количество шагов равно 11. В режиме 1 мы имеем систему 200 уравнений от 66 переменных. В режиме 2 — 114 уравнений от 54 переменных. В режиме 3 — 79 уравнений от 45 переменных. Было найдено 23 различных решения этой системы уравнений. В одних случаях оказалось, что $b_2 = b_3 = 0$, в других

$$a_{4,2} = a_{6,2} = a_{7,2} = a_{8,2} = a_{9,2} = \\ = a_{10,2} = a_{11,2} = b_2 = b_3 = b_5 = 0.$$

Ниже приведено, для примера, одно из решений.

9-шаговый метод порядка 7

$a(2,1) = 0.113179927667738406;$	$a(4,1) = 0.102728593619062964;$
$a(3,1) = 0.049219881180079666;$	$a(4,2) = -0.359436122584395178;$
$a(3,2) = 0.104804762710473952;$	$a(4,3) = 0.557157080849307528;$

$a(5,1) = 0.199092550677384833;$
$a(5,2) = -0.099832752020211442;$
$a(5,3) = -0.119739083688801858;$
$a(5,4) = 0.528515566967551287;$
$a(6,1) = -0.350439143765170241;$
$a(6,2) = 0.542745999087193074;$
$a(6,3) = 0.629694222104261786;$
$a(6,4) = -0.656361934003744630;$
$a(6,5) = 0.452963585569137644;$
$a(7,1) = -0.662848861781147710;$
$a(7,2) = 0.515350973635847237;$
$a(7,3) = 1.440293782910771610;$
$a(7,4) = -1.038011560503347910;$
$a(7,5) = -0.151292199327624093;$
$a(7,6) = 0.708760997375124620;$
$a(8,1) = -0.369243133102607611;$
$a(8,2) = 0.236265744606767408;$
$a(8,3) = 1.001431226232166850;$
$a(8,4) = -0.517352367865280345;$
$a(8,5) = -0.144917751418109253;$
$a(8,6) = 0.589394066933993918;$
$a(8,7) = 0.036254454758449798;$
$a(9,1) = 0.036142823278044161;$
$a(9,2) = 0.354975145480235400;$
$a(9,3) = -0.149878015794613351;$
$a(9,4) = -0.023606430077946213;$
$a(9,5) = 0.464755777386500985;$
$a(9,6) = 0.293579023909080741;$
$a(9,7) = -2.670677502282863890;$
$a(9,8) = 2.694709178101562160;$
$b(1) = 0.049135665354465524;$
$b(2) = 0.000000000000000000;$
$b(3) = 0.205305274193170839;$
$b(4) = 0.131782263475263043;$
$b(5) = 0.169894715914204381;$
$b(6) = 0.200344075933446291;$
$b(7) = -0.459966443357690592;$
$b(8) = 0.662180015158790240;$
$b(9) = 0.041324433328350272.$

11-шаговый метод порядка 8

$a(2,1) = 0.059319505883829309;$
$a(3,1) = 0.020471297201502607;$
$a(3,2) = 0.071509381498310484;$
$a(4,1) = 0.034354808684220139;$
$a(4,2) = 0.000681870892768358;$
$a(4,3) = 0.102725508591067932;$
$a(5,1) = 0.112421163611666098;$
$a(5,2) = -0.001957750607276183;$
$a(5,3) = -0.294939891057742932;$
$a(5,4) = 0.445998024435520471;$
$a(6,1) = 0.334031032742736459;$
$a(6,2) = -0.002610998081461815;$
$a(6,3) = -0.393353211998318554;$
$a(6,4) = -0.097402323693928965;$
$a(6,5) = 0.649384028162091799;$
$a(7,1) = -0.499263628714436439;$
$a(7,2) = 0.005609245763983472;$
$a(7,3) = 0.845046518363683797;$
$a(7,4) = 0.420526210600834890;$
$a(7,5) = -0.536225020514290275;$
$a(7,6) = 0.280104641069148574;$
$a(8,1) = 0.148495575086211397;$
$a(8,2) = -0.000092637694304033;$
$a(8,3) = -0.013956093995142749;$
$a(8,4) = 0.070413852020828480;$
$a(8,5) = 0.050088898610570998;$
$a(8,6) = -0.192164445345554290;$
$a(8,7) = 0.689571149219483224;$
$a(9,1) = 0.101182567568445031;$
$a(9,2) = -0.000963250930202439;$
$a(9,3) = -0.145116095657098366;$
$a(9,4) = 0.056957148905869887;$

$a(9,5) = 0.511249132625016142;$
$a(9,6) = 0.199389732138940969;$
$a(9,7) = -0.171126575739931252;$
$a(9,8) = 0.306886695111395092;$
$a(10,1) = 0.025614259766717709;$
$a(10,2) = -0.001386774065325391;$
$a(10,3) = -0.208920886142416848;$
$a(10,4) = 0.444635040596401608;$
$a(10,5) = 0.119119650792147949;$
$a(10,6) = 0.703713851934866143;$
$a(10,7) = -0.469848966704835262;$
$a(10,8) = 0.168190479821966755;$
$a(10,9) = 0.089392334179579942;$
$a(11,1) = -0.007846116527054911;$
$a(11,2) = 0.002550436323561601;$
$a(11,3) = 0.384229435860082674;$
$a(11,4) = -0.167700285482564580;$
$a(11,5) = 0.115160074203313317;$
$a(11,6) = -0.439275091920598837;$
$a(11,7) = 0.806181104890393275;$
$a(11,8) = 0.108205167504093542;$
$a(11,9) = -0.226705528902226075;$
$a(11,10) = 0.425200804050999993;$
$b(1) = 0.043789881258183398;$
$b(2) = 0.000000000000000000;$
$b(3) = 0.000000000000000000;$
$b(4) = 0.181211816894774095;$
$b(5) = 0.121639796567780699;$
$b(6) = 0.182534013409433159;$
$b(7) = 0.126541500800936805;$
$b(8) = 0.139818916037593476;$
$b(9) = 0.020298355811190280;$
$b(10) = 0.141172827744498263;$
$b(11) = 0.042992891475609825.$

Программу, описанную в данной статье и файлы со всеми найденными методами можно получить у автора (Хашин Сергей Иванович) по адресу khash@interline.ru.

Библиографический список

1. *Березин И. С., Жидков Н. П.* Методы вычислений. М., 1962. Т.2.
2. *Хайрер Э., Нерсетт С., Ваннер Г.* Решение обыкновенных дифференциальных уравнений. Нежесткие задачи. М., 1990.
3. *Butcher J. C.* Coefficients for the study of Runge-Kutta Iteration Processes // J. of the Australian Math. Soc. 1963. Vol. 3.
4. *Butcher J.C.* On Runge-Kutta processes of high order // Ibid. 1964. Vol. 4.