Computers and Mathematics with Applications 62 (2011) 770–775

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Runge–Kutta pairs of order 5(4) satisfying only the first column simplifying assumption

Ch. Tsitouras *

TEI of Chalkis, Department of Applied Sciences, GR34400, Psahna, Greece

ARTICLE INFO

Article history:
Received 24 January 2011
Received in revised form 15 April 2011
Accepted 1 June 2011

Keywords:
Runge–Kutta
Truncation error
Non-linear algebraic systems
Free parameters
Dense output

ABSTRACT

Among the most popular methods for the solution of the Initial Value Problem are the Runge–Kutta pairs of orders 5 and 4. These methods can be derived solving a system of nonlinear equations for its coefficients. To achieve this, we usually admit various simplifying assumptions. The most common of them are the so-called row simplifying assumptions. Here we neglect them and present an algorithm for the construction of Runge–Kutta pairs of orders 5 and 4 based only in the first column simplifying assumption. The result is a pair that outperforms other known pairs in the bibliography when tested to the standard set of problems of DETEST. A cost free fourth order formula is also derived for handling dense output.

1. Introduction

We consider the numerical solution of the non-stiff initial value problem,

$$y' = f(x, y), \qquad y(x_0) = y_0 \in \mathbb{R}^m, \quad x \in [x_0, x_f] \tag{1}$$

where the function $f : \mathbb{R}\times\mathbb{R}^m \to \mathbb{R}^m$ is assumed to be as smooth as necessary. Traditionally, explicit embedded Runge–Kutta methods produce an approximation to the solution of (1) only at the end of each step.

The general s-stage embedded Runge–Kutta pair of orders $p(p-1)$, for the approximate solution of the problem (1) can be defined by the following Butcher scheme [1,2]

$$\begin{array}{c|c} c & A \\ \hline & b \\ & \hat{b} \end{array}$$

where $A \in \mathbb{R}^{s\times s}$, is strictly lower triangular, $b^T, \hat{b}^T, c \in \mathbb{R}^s$ with

$$c = A \cdot e, \quad e = [1, 1, \ldots, 1]^T \in \mathbb{R}^s.$$

The vectors $\hat{b}$, b define the coefficients of the $(p-1)$-th and p-th order approximations respectively.

Starting with a given value $y(x_0) = y_0$, this method produces approximations at the mesh points $x_0 < x_1 < x_2 < \cdots < x_f$. Throughout this paper, we assume that local extrapolation is applied, hence the integration is advanced using the p-th order approximation. To estimate the error, two approximations are evaluated at each step x_n to $x_{n+1} = x_n + h_n$. These are:

$$\hat{y}_{n+1} = y_n + h_n \sum_{j=1}^{s} \hat{b}_j f_j \quad \text{and} \quad y_{n+1} = y_n + h_n \sum_{j=1}^{s} b_j f_j,$$

* Tel.: +30 2228099621.
E-mail address: tsitoura@teihal.gr.
URL: http://users.ntua.gr/tsitoura/.

0898-1221/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2011.06.002

where

$$f_i = f\left(x_n + c_i h_n, y_n + h_n \sum_{j=1}^{i-1} a_{ij} f_j\right), \quad i = 1, 2, \ldots, s.$$

The local error estimate $E_n = \|y_n - \hat{y}_n\|$ of the $(p-1)$-th order Runge–Kutta pair is used for the automatic selection of the step size. Given a Tolerance TOL $> E_n$, the algorithm

$$h_{n+1} = 0.9 \cdot h_n \cdot \left(\frac{\mathrm{TOL}}{E_n}\right)^{\frac{1}{p}}$$

furnishes the next step length. In case TOL $< E_n$ then we reject the current step and try again with the left side of the above formula being h_n.

In case that $c_s = 1, a_{s,j} = b_j$ for $j = 1, 2, \ldots, s-1$ and $b_s = 0 \neq \hat{b}_s$ then the First Stage of each step is the same As the Last one of the previous stage. This device was possibly first used in [3, pg. 22] and it is called FSAL. The pair shares effectively only $s-1$ stages per step then.

Let $y_n(x)$ be the solution of the local initial value problem

$$y'(x) = f(x, y_n(x)), \quad x \geq x_n,\ y_n(x_n) = y_n.$$

Then E_{n+1} is an estimate of the error in the local solution $y_n(x)$ at $x = x_{n+1}$. The local truncation error t_{n+1} associated with the higher order method is

$$t_{n+1} = y_{n+1} - y_n(x_n + h_n) = \sum_{q=1}^{\infty} h_n^q \sum_{i=1}^{\lambda_q} T_{qi} P_{qi} = h_n^{p+1} \Phi(x_n, y_n) + O(h_n^{p+1})$$

where

$$T_{qi} = Q_{qi} - \xi_{qi}/q!$$

with Q_{qi} algebraic functions of A, b, c and ξ_{qi} positive integers. P_{qi} are differentials of f evaluated at (x_n, y_n) and $T_{qi} = 0$ for $q = 1, 2, \ldots, p$ and $i = 1, 2, \ldots, \lambda_q$. λ_q is the number of elementary differentials for each order and coincides with the number of rooted trees of order q. It is known that

$$\lambda_1 = 1, \lambda_2 = 1, \lambda_3 = 2, \lambda_4 = 4, \lambda_5 = 9, \lambda_6 = 20, \lambda_7 = 48, \ldots, \text{etc. [4]}.$$

The set $T^{(q)} = \{T_{q1}, T_{q2}, \ldots, T_{q,\lambda_q}\}$ is formed by the q-th order truncation error coefficients. It is the usual practice for a $(q-1)$-th order method to have minimized

$$\|T^{(q)}\|_2 = \sqrt{\sum_{j=1}^{\lambda_q} T_{qj}^2}.$$

2. Derivation of RK pairs of orders 5(4)

The construction of an effectively 6-stage FSAL Runge–Kutta pair of orders 5(4) requires the solution of a nonlinear system of 25 order conditions. $\lambda_1 + \cdots + \lambda_5 = 17$ equations for the higher order formula and $\lambda_1 + \cdots + \lambda_4 = 8$ equations for the lower order formula. There are 28 unknowns. Namely $c_2 - c_6$, $b_1 - b_6$, $\hat{b}_1 - \hat{b}_7$, a_{32}, a_{42}, a_{43}, a_{52}, a_{53}, a_{54} and $a_{62} - a_{65}$.

We proceed setting $c_6 = 1$ and an arbitrary value for $\hat{b}_7$. Then the only assumption we make is

$$b \cdot (A + C - I_s) = 0 \in \mathbb{R}^{1\times s} \tag{2}$$

with C =diag(c) and $I_s \in \mathbb{R}^{s\times s}$ the identity matrix. This is the minimal set of simplifying assumptions for pairs of orders 5(4). It is worth mentioning that in the family of methods introduced here

$$A \cdot c \neq \frac{c^2}{2}, \quad \text{and} \quad b_2 \neq 0,$$

contrary to the common practice of every 5(4) pair appearing until now [5,3,6]. Expression c^2 is to be understood as component-wise multiplication $c * c$.

The implicit algorithm that derives a pair of the new family follows. A different approach was given in [7].

The algorithm producing the coefficients of the new pair

Set $c_6 = 1$ and get an arbitrary $\hat{b}_7 \neq 0$. Select free parameters c_2, c_3, c_4 and $b_2 \neq 0$. Then

1. Solve $b \cdot e = 1$, $b \cdot c = \frac{1}{2}$, $b \cdot c^2 = \frac{1}{3}$, $b \cdot c^3 = \frac{1}{4}$, $b \cdot c^4 = \frac{1}{5}$ for b_1, b_3, b_4, b_5 and b_6.
2. Solve $\hat{b} \cdot e = 1$, $\hat{b} \cdot c = \frac{1}{2}$, $\hat{b} \cdot c^2 = \frac{1}{3}$, $\hat{b} \cdot c^3 = \frac{1}{4}$ for $\hat{b}_1, \hat{b}_3, \hat{b}_4$, and $\hat{b}_5$.

Table 1
The coefficients of the new pair.

$c_2 = 0.161$	$c_3 = 0.327$
$c_4 = 0.9$	$c_5 = 0.9800255409045097$
$c_6 = c_7 = 1$	$b_1 = 0.09646076681806523$
$b_2 = 0.01$	$b_3 = 0.4798896504144996$
$b_4 = 1.379008574103742$	$b_5 = -3.290069515436081$
$b_6 = 2.324710524099774$	$\hat{b}_1 = 0.001780011052226$
$\hat{b}_2 = 0.000816434459657$	$\hat{b}_3 = -0.007880878010262$
$\hat{b}_4 = 0.144711007173263$	$\hat{b}_5 = -0.582357165452555$
$\hat{b}_6 = 0.458082105929187$	$\hat{b}_7 = \frac{1}{66},\ b_7 = 0$
$a_{32} = 0.3354806554923570$	$a_{42} = -6.359448489975075$
$a_{52} = -11.74888356406283$	$a_{43} = 4.362295432869581$
$a_{53} = 7.495539342889836$	$a_{54} = -0.09249506636175525$
$a_{62} = -12.92096931784711$	$a_{63} = 8.159367898576159$
$a_{64} = -0.07158497328140100$	$a_{65} = -0.02826905039406838$
$a_{i1} = c_i - \sum_{j=2}^{j=i-1} a_{ij},\ i \geq 2$	$a_{7i} = b_i,\ i = 1, 2, \ldots, 6$

3. Solve $b \cdot (A + C - I_s) = 0$ for $a_{62}, \ldots, a_{65}$.
4. Solve $b \cdot A^3 c = \frac{1}{120}$, $b \cdot A^2 \cdot c^2 = \frac{1}{60}$, $b \cdot C^2 Ac = \frac{1}{10}$, $b \cdot ACAc = \frac{1}{40}$, for a_{52}, a_{53}, a_{54} and a_{43}. Adjust the values of a_{62}, a_{63} and a_{64}
5. Solve $\hat{b} \cdot A^2 c = \frac{1}{24}$, $\hat{b} \cdot Ac^2 = \frac{1}{12}$, $\hat{b} \cdot Ac = \frac{1}{6}$ for $\hat{b}_2, \hat{b}_6$ and a_{42}. Adjust the values of a_{52}, a_{53}, a_{43} and a_{62}, a_{63}, a_{64}. Reevaluate all $\hat{b}$'s.
6. Solve $\hat{b} \cdot CAc = \frac{1}{8}$ for c_5. Reevaluate $a_{52}, a_{53}, a_{54}, a_{43}, \hat{b}_5, \hat{b}_6$ and a_{42}. Adjust all b's.
7. Solve $b \cdot (Ac)^2 = \frac{1}{20}$ for a_{32}. Compute the final values of the coefficients.
8. Compute explicitly $a_{21}, a_{31}, a_{41}, a_{51}, a_{61}$ from $A \cdot e = c$.

The equations 1–6 can be solved linearly for the coefficients. The seventh equation is a rational function over a_{32}. The numerator of that function is a polynomial of the sixth degree and may have some real solutions for a_{32}. It was not proven theoretically but almost 2000 random runs over accepted regions for the coefficients always gave some real roots.

In case that we choose $b_2 = 0$ as input then the equations in steps 6 and 7 become:

$$\hat{b} \cdot CAc - 1/8 = p(a_{32}) \cdot q_1(a_{32}, c_5) \quad \text{and} \quad b \cdot (Ac)^2 - 1/20 = p(a_{32}) \cdot q_2(a_{32}, c_5).$$

Thus we have the option to evaluate a_{32} from polynomial $p(a_{32})$ and satisfy both equations. c_5 remains free after using this alternative. Observe that the choice $b_2 = 0$ and $p(a_{32}) \neq 0$ give pairs with $A \cdot c \neq \frac{c^2}{2}$.

A Mathematica [8] implementation of the above algorithm requires reevaluation of the coefficients and the order conditions in every step of the algorithm above. On a small computer it needs about 2–3 s to derive the coefficients.

3. The new Runge–Kutta pair

At first we tried to solve all the required 25 equations using the Mathematica function `NMinimize`. The Differential Evolution technique [9] implemented in Mathematica was applied as an option. We found hundreds of solutions, all of them satisfying at least the Eq. (2) or $A \cdot c = \frac{c^2}{2}$. We avoided the latter as an assumption and proceeded using the algorithm given in the previous section which uses (2) as simplification. Again the function `NMinimize` was used with the same option and as objective function $\|T^{(6)}\|_2$ evaluated with the coefficients given from the given algorithm.

We got various interesting pairs and among the best is the one presented in Table 1. This method shares a rather large value of b_2 and it is clearly not included in any family of solutions appearing in the relevant literature until now. The Norm of the principal truncation error is $\|T^{(6)}\|_2 \approx 1.38 \cdot 10^{-4}$ while the corresponding value for the Dormand and Prince pair is $\|T^{(6)}\|_2 \approx 3.99 \cdot 10^{-4}$ [5].

4. Dense output

For a costless fourth order approximation at intermediate points $y(x_n + th_n)$ we use the same stages $f_j,\ 1 \leq j \leq 7$ and combine them to the formula:

$$y(x_n + th_n) + O(h^5) = \widetilde{y}_{n+t} = y_n + h_n \sum_{j=1}^{7} \widetilde{b}_j(t) f_j,$$

where $\widetilde{b}_j(t)$ are polynomials in t of the fourth degree. These polynomials form the vector $\widetilde{b}$ which satisfies the eight order conditions [10]:

$$\widetilde{b}(t)e = t, \quad \widetilde{b}(t)c = \frac{t^2}{2}, \quad \frac{1}{2}\widetilde{b}(t)c^2 = \frac{t^3}{6}, \quad \widetilde{b}(t)Ac = \frac{t^3}{6},$$
$$\frac{1}{6}\widetilde{b}(t)c^3 = \frac{t^4}{24}, \quad \frac{1}{2}\widetilde{b}(t)Ac^2 = \frac{t^4}{24}, \quad \widetilde{b}(t)\,(c*(Ac)) = \frac{t^4}{8}, \quad \widetilde{b}(t)A^2c = \frac{t^4}{24}.$$

These equations are linear in $\widetilde{b}$ and can be solved simultaneously leaving one polynomial as free parameter. This polynomial (say $\widetilde{b}_7$) has 5 coefficients. Only two of them are needed for satisfying C^0 continuity. For this property we ask:

$$\widetilde{b}_i(0) = 0, \quad i = 1, 2, \dots, 7, \quad \text{and} \quad \widetilde{b}_i(1) = b_i, \quad i = 1, 2, \dots, 7.$$

Then we proceed determining another two coefficients of the free polynomial for C^1 continuity. It holds

$$\left.\frac{\mathrm{d}\widetilde{b}_1(t)}{\mathrm{d}t}\right|_{t=0} = 1, \quad \left.\frac{\mathrm{d}\widetilde{b}_i(t)}{\mathrm{d}t}\right|_{t=0} = 0, \quad i = 2, 3, \dots, 7$$
$$\left.\frac{\mathrm{d}\widetilde{b}_7(t)}{\mathrm{d}t}\right|_{t=1} = 1, \quad \left.\frac{\mathrm{d}\widetilde{b}_i(t)}{\mathrm{d}t}\right|_{t=1} = 0.$$

Finally one coefficient remains for minimizing the truncation error coefficients of the fifth order. Since these terms depend on t, we integrate their Euclidean norm in the interval [0, 1]

$$\int_{t=0}^{t=1} \|\widetilde{T}^{(5)}\|_2 \mathrm{d}t = \int_{t=0}^{t=1} \left(\sqrt{\widetilde{T}_{5,1}^2(t) + \widetilde{T}_{5,2}^2(t) + \cdots \widetilde{T}_{5,9}^2(t)}\right) \mathrm{d}t.$$

The resulting interpolant is:

$$\widetilde{b}_1 = -1.0530884977290216t(t - 1.3299890189751412)\left(t^2 - 1.4364028541716351t + 0.7139816917074209\right)$$
$$\widetilde{b}_2 = 0.1017t^2\left(t^2 - 2.1966568338249754t + 1.2949852507374631\right)$$
$$\widetilde{b}_3 = 2.490627285651252793t^2\left(t^2 - 2.38535645472061657t + 1.57803468208092486\right)$$
$$\widetilde{b}_4 = -16.54810288924490272(t - 1.21712927295533244)(t - 0.61620406037800089)t^2$$
$$\widetilde{b}_5 = 47.37952196281928122(t - 1.203071208372362603)(t - 0.658047292653547382)t^2$$
$$\widetilde{b}_6 = -34.87065786149660974(t - 1.2)(t - 0.666666666666666667)t^2$$
$$\widetilde{b}_7 = 2.5(t - 1)(t - 0.6)t^2.$$

We observed that

$$\max_t \|\widetilde{T}^{(5)}\|_2 \approx 7.78 \cdot 10^{-4}$$

for $t \approx 0.285$ and $\|\widetilde{T}^{(5)}\|_2$ is kept for every $t \in [0, 1]$ well under the corresponding value of the underlying 4-th order method, $\|\hat{T}^{(5)}\|_2 \approx 1.75 \cdot 10^{-3}$.

5. Numerical results

We run the Runge–Kutta pair for the 25 DETEST [11] non-stiff problems and for tolerances $10^{-3}, 10^{-4}, \dots, 10^{-7}$. For stringent tolerances it is preferred to use higher order pairs. DETEST was implemented through MATLAB2009a on a Pentium IV computer running Windows XP at 3.4 GHz. For comparison purposes the DP5(4) pair [5] was also run for the same tolerances. We present the results in Table 2.

These results were developed according to the guidelines given in [12,13]. Briefly, let us assume that the global error achieved at all grid points over the integration interval satisfies the relation $ge = C \cdot \mathrm{TOL}^E$ and that its value is known for several tolerances. The values of E and C can easily be found in the sense of a least squares approximation. These values are then used, with linear interpolation, in order to estimate the number of derivative evaluations required to achieve a prescribed ge. We present the efficiency gains of DP5(4) in relation to the new one, for the respective problems and the expected accuracies, counted in units of 10%, in Table 2. The numbers in these tables are the ratios in the function evaluation costs of the two pairs being tested. The larger value is always divided by the smaller value and the efficiency gain is formed by subtracting 1 from this ratio. Subsequently the result is multiplied by 10 and rounded to the nearest integer. Positive numbers mean that the first of the two pairs is superior. Zero entries indicate a difference less than ±5%. Unity entries indicate differences between 5%–15% and so on. The final row gives the mean value of efficiency gain for each tolerance and problem. The final row's first number is the average efficiency gain for all problems. Empty places in the tables are due to the unavailability of data for the respective tolerances. See [14] for more details.

The coefficient $\hat{b}_7$ does not affect $\|T^{(6)}\|_2$. It was chosen so comparable global errors were achieved by both methods for the same tolerances. Thus Table 2 is as full as possible. We finally observe that the new method is in average 10% more efficient than Dormand–Prince 5(4) for the DETEST problems. It is a remarkable improvement over pairs of the same order and origin.

Table 2
Efficiency gains of NEW5(4) relative to DP5(4), for the range of tolerances $10^{-3}, \ldots, 10^{-7}$.

g.e.	A_1	A_2	A_3	A_4	A_5	B_1	B_2	B_3	B_4	B_5	C_1	C_2	C_3	C_4	C_5	D_1	D_2	D_3	D_4	D_5	E_1	E_2	E_3	E_4	E_5
−1																1	1	2	1	0					
−2						−1			1							1	0	1	0	−1		0	1		
−3			0			0			1	0		0		0	1	1	0	0	−1		1	0	2		
−4	1	1	0	2	0	1	0	1	0	1	1	0	3	1	2	1	0	−1			1	0	2		
−5	1	1	−1	2	0	3	1	1	0	3	1	0		1	2	1					2	1	2	1	4
−6	1	1	−1	2	1		1	1			1	0		1	2						3			1	3
−7	1						1	1			2			2										1	1
10%	1	1	0	2	0	1	1	1	0	1	1	0	3	1	2	1	0	0	0	0	2	0	2	1	3

Appendix

The Mathematica code for the algorithm described in Section 2. We insert three distinct c_2, c_3, c_4 and an arbitrary b_2 as input with $c_2c_3c_4 \neq 0$. The package returns the sets of coefficients depending on the number of real roots of the equation $b \cdot (Ac)^2 = 1/20$ with respect to a_{32}. i.e. two, four or six sets of coefficients.

```
BeginPackage[ "csa‘"];
Clear[ "csa‘*" ]

TEST::usage = "  TEST[x1,x2,x3,x4]  of a RK5(4) with b.(a+c-i)=0 "

Begin["‘Private‘"];
Clear[ "csa‘Private‘*" ];

TEST[cc2_?NumericQ, cc3_?NumericQ, cc4_?NumericQ, bc2_?NumericQ] :=
    Module[{a32, a42, a43, a52, a53, a54, a62, a63, a64, a65, bb1, bb2,
    bb3, bb4, bb5, bb6, bb7, b1, b2, b3, b4, b5, b6, c2, c3, c4, c5, so,
    so1, a, b, bb, c, ba, equ, eequ, temp, j1},

c2 = Rationalize[cc2, 10^-6]; c3 = Rationalize[cc3, 10^-6];
c4 = Rationalize[cc4, 10^-6];b2 = Rationalize[bc2, 10^-6];

equ = {-(1/120) + b.a.a.a.c, -(1/60) + b.a.a.c^{2}, -(1/40) + b.a.(c*a.c),
       -(1/20) + b.a.c^{3}, -(1/30) + b.(c*a.a.c), -(1/15) +  b.(c*a.c^{2}),
       -(1/20) + b.(a.c)^{2}, -(1/10) + b.(c^{2}*a.c), -(1/5) +  b.c^{4},
       -(1/24) + b.a.a.c, -(1/12) + b.a.c^{2}, -(1/8) + b.(c*a.c),
       -(1/4) + b.c^{3}, -(1/6) + b.a.c, -(1/3) + b.c^{2}, -(1/2) + b.c,
       -1 + b.{1, 1, 1, 1, 1, 1, 1}};

eequ = {-(1/24) + bb.a.a.c, -(1/12) + bb.a.c^{2}, -(1/8) +  bb.(c*a.c),
        -(1/4) + bb.c^{3}, -(1/6) + bb.a.c, -(1/3) + bb.c^{2}, -(1/2) + bb.c,
        -1 + bb.{1, 1, 1, 1, 1, 1, 1}};

c = {0, c2, c3, c4, c5, 1, 1};

b = {b1, b2, b3, b4, b5, b6,  0};

bb = {bb1, bb2, bb3, bb4, bb5, bb6, 1/40};

a = {{0, 0, 0, 0, 0, 0, 0},
     {c2, 0, 0, 0, 0, 0, 0},
     {c3-a32, a32, 0, 0, 0, 0, 0},
     {c4 - a42 - a43, a42, a43, 0,  0, 0, 0},
     {c5 - a52 - a53 - a54, a52, a53, a54, 0, 0,  0},
     {1 - a62 - a63 - a64 - a65, a62, a63, a64, a65, 0, 0}, b};

so = Solve[{equ[[17]] == 0, equ[[16]] == 0, equ[[15]] == 0,
           equ[[13]] == 0, equ[[9]] == 0}, {b1, b3, b4, b5, b6}];
```

```
{b1, b3, b4, b5, b6} = Simplify[so[[1, 1 ;; 5, 2]]];
so = Solve[{eequ[[4]] == 0, eequ[[6]] == 0, eequ[[7]] == 0, eequ[[8]] == 0},
                                                         {bb1, bb3, bb4, bb5}];

{bb1, bb3, bb4, bb5} = Simplify[so[[1, 1 ;; 4, 2]]];

ba = Simplify[b.(a + DiagonalMatrix[c] - IdentityMatrix[7])];

so = Solve[{ba[[2 ;; 5]] == {0, 0, 0, 0}}, {a62, a63, a64, a65}];

{a62, a63, a64, a65} = Simplify[so[[1, 1 ;; 4, 2]]];

so = Solve[{equ[[2]] == 0, equ[[1]] == 0, equ[[3]] == 0, equ[[8]] == 0},
                                   {a43, a52, a53, a54}, Sort -> False];

{a43, a52, a53, a54} = Simplify[so[[1, 1 ;; 4, 2]]];

{a62, a63, a64, a65} = Simplify[{a62, a63, a64, a65}];

so = Solve[{eequ[[1]] == 0, eequ[[2]] == 0}, {bb2, bb6}, Sort->False];

{bb2, bb6} = Simplify[so[[1, 1 ;; 2, 2]]];

so=Solve[{eequ[[5]]==0},{a42}];a42=Simplify[so[[1,1,2]]];

{a43, a52, a53, a54, a62, a63, a64, bb1, bb3, bb4, bb5} =
             Simplify[{a43, a52, a53, a54, a62, a63, a64, bb1, bb3, bb4, bb5}];

so= Solve[{eequ[[3]] == 0}, {c5}];
c5 = Simplify[so[[1, 1, 2]]];

{a52, a53, a54, a43, bb5, bb6, a42, b1, b2, b3, b4, b5, b6} =
         Simplify[{a52, a53, a54, a43, bb5, bb6, a42, b1, b2, b3, b4, b5, b6}];

{bb1, bb2, bb3, bb4, bb5, bb6} = Simplify[{bb1, bb2, bb3, bb4, bb5 ,bb6}];

so = Union[Solve[{Numerator[Simplify[equ[[7]]]] == 0}, {a32}]];

temp = N[Select[so, Im[#[[1, 2]]] == 0 &],25];

Return[Table[{a, b, bb, c} /. temp[[j1]], {j1, 1, Length[temp]}]]
];

End[ ];
EndPackage[ ];
```

References

[1] J.C. Butcher, Implicit Runge–Kutta processes, Math. Comp. 18 (1964) 50–64.
[2] J.C. Butcher, On Runge–Kutta processes of high order, J. Aust. Math. Soc. 4 (1964) 179–194.
[3] E. Fehlberg, Low order classical Runge–Kutta formulas with stepsize control and their application to some heat-transfer problems, TR R-287, NASA, 1969.
[4] J.C. Butcher, Coefficients for the study of Runge–Kutta integration processes, J. Aust. Math. Soc. 3 (1963) 185–201.
[5] J.R. Dormand, P.J. Prince, A family of embedded Runge–Kutta formulae, J. Comput. Appl. Math. 6 (1980) 19–26.
[6] S.N. Papakostas, G. Papageorgiou, A family of fifth order Runge–Kutta pairs, Math. Comp. 65 (1996) 1165–1181.
[7] Ch. Tsitouras, Runge–Kutta pairs of orders 5(4) using the minimal set of simplifying assumptions, AIP Conf. Proc. 1168 (2009) 69–72.
[8] Wolfram Research, Inc., Mathematica, Version 6.0, Champaign, IL, 2007.
[9] K.V. Price, R.M. Storn, J.A. Lampinen, Differential Evolution: A Practical Approach to Global Optimization, Springer, Berlin, 2005.
[10] Ch. Tsitouras, Runge–Kutta interpolants for high precision computations, Numer. Algorithms 44 (2007) 291–307.
[11] W.H. Enright, J.D. Pryce, Two FORTRAN packages for assessing initial value methods, ACM Trans. Math. Softw. 13 (1987) 1–27.
[12] P.W. Sharp, Numerical comparisons of some explicit Runge–Kutta pairs, ACM Trans. Math. Softw. 17 (1991) 387–409.
[13] Ch. Tsitouras, S.N. Papakostas, Cheap error estimation for Runge–Kutta pairs, SIAM J. Sci. Comput. 20 (1999) 2067–2088.
[14] S.N. Papakostas, Ch. Tsitouras, G. Papageorgiou, A general family of Runge–Kutta pairs of orders 6(5), SIAM J. Numer. Anal. 33 (1996) 917–926.