Evolutionary Generation of Runge-Kutta Pairs

Y. G Petalas[1], **Ch. Tsitouras**[2], **G. S Androulakis**[3], and **M. N Vrahatis** *[1]

[1] Computational Intelligence Laboratory (CI Lab), Department of Mathematics, University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR–26110 Patras, Greece
[2] Department of Applied Sciences, TEI of Chalkis, GR–34400 Psahna, Greece
[3] Department of Business Administration, University of Patras, GR–26110 Patras, Greece

Received 10 July 2006, revised 20 July 2006, accepted 20 July 2006

Key words Evolutionary Algorithms, Initial Value Problem, Differential Evolution
Subject classification 65L05, 65K05

A new methodology for the generation of explicit or implicit Runge-Kutta (RK) methods is presented. In the proposed methodology, Evolutionary Algorithms will be used to generate RK methods. Candidate vectors of coefficients will be evolved in order to find RK coefficients which satisfy the RK algebraic system and simultaneously have as small as possible local truncation error.

1 Introduction

The problem of constructing Runge–Kutta (RK) methods has drawn a lot of attention and has evolved to a rather specialized branch of numerical mathematics [7, 10, 13, 16, 17]. Several authors have created such methods for the numerical solution of the Initial Value Problem (IVP) :

$$\begin{aligned} y'(x) &= f\Big(x, y(x)\Big), \\ y(x_0) &= y_0, \end{aligned} \tag{1}$$

with $x_0 \in \mathbb{R}$, $y_0 \in \mathbb{R}^n$, $y : \mathbb{R} \to \mathbb{R}^n$, $f : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$. Given a step–size $h_n = x_{n+1} - x_n$, an s–stage explicit RK pair is defined as follows :

$$y_{n+1} = y_n + h_n \sum_{j=1}^{s} b_i \, k_i, \qquad \hat{y}_{n+1} = y_n + h_n \sum_{j=1}^{s} \hat{b}_i \, k_i,$$

where

$$k_i = f\left(x_n + c_i \, h, y_n \, + \, h \sum_{j=1}^{i-1} a_{ij} \, k_j\right), \qquad i = 2, \ldots, s,$$

and it provides for each $n,\ n = 1, \ldots$, an approximation y_{n+1} to $y(x_{n+1})$ where $y(x)$ is the solution of IVP (1). The reference solution $\hat{y}_{n+1}$ is used for the control of the step–size. 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 $h_{n+1} = 0.9 \cdot h_n \cdot (TOL/E_n)^{1/p}$, furnishes the next step length. In case $TOL < E_n$ then we reject the current step and try again with the left side of above formula being h_n.

The number s of k_i's must be chosen large enough, and the parameters must be determined so that the Taylor series of $y(x_n + h)$ matches that of y_{n+1} through the terms involving h^p. These requirements generate a system of nonlinear algebraic equations in the parameters [3]. The number of equations grows rapidly with the size of the algebraic order p and it is in general smaller than the number of the parameters. Since there are free parameters there are infinitely many solutions of the above system.

* Corresponding author: e-mail: vrahatis@math.upatras.gr, Phone: +302610997374, Fax: +302610992965

Using Butcher trees [6] we are able to formulate RK algebraic system of equations i.e. the conditions which any s-stage RK method must satisfy, to be of any particular order.

In [1, 2] the solution of the RK algebraic system was transformed to the solution of unconstrained optimization problem. In this paper we propose a method that uses the Evolutionary Algorithms (EA) [8] to generate RK methods. EA are very effective when the objective function has many local optima (as in our case) and can be applied in situations where the objective function is not differentiable or /and discontinuous.

The paper is organized as follows. In Section 2 the algebraic system for the coefficients of a RK pair is presented. In Section 3 the Differential Evolution Algorithm (DE) is described. The proposed pair of orders $5(4)$ is exhibited in Section 4 while in Section 5 some preliminary results from the application of the proposed method are shown.

2 Order conditions for Runge–Kutta methods

The general $s-$stage embedded Runge-Kutta pair of orders $p(p-1)$, for the approximate solution of the IVP (1) can be defined by the following Butcher scheme [4, 5]

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

where $A \in \mathbb{R}^{s\times s}$, is strictly lower triangular, $b^\top,\ \hat{b}^\top,\ c \in \mathbb{R}^s$ with $c = A \cdot e$, $e = [1,\ 1, \ldots, 1]^\top \in \mathbb{R}^s$. The vectors $\hat{b},\ b$ define the coefficients of the $(p-1)-$th and $p-$th order approximations respectively.

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

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

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 $q = 1, 2, \cdots, p$ and $i = 1, 2, \cdots, \lambda_q$. 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 [3]

$$\lambda_1 = 1,\ \lambda_2 = 1,\ \lambda_3 = 2,\ \lambda_4 = 4,\ \lambda_5 = 9,\ \lambda_6 = 20,\ \lambda_7 = 48, \ldots,$$

The set $T^{(q)} = \{T_{q1},\ T_{q2}, \cdots, T_{q,\lambda_q}\}$ is formed by the $q-$th order truncation error coefficients. It is usual practice a $(q-1)-$th order method to have minimized $\|T^{(q)}\|_2 = \sqrt{\sum_{j=1}^{\lambda_q} T_{qj}^2}$.

In the present paper we are interested for deriving $5(4)$ pairs. For a fifth order method $\sum_{i=1}^{i=5} \lambda_i = 17$ equations have to be satisfied. For the companion fourth order formula another $\sum_{i=1}^{i=4} \lambda_i = 8$ equations have to be satisfied. So in total 25 constrains required. In addition $\|T^{(6)}\|_2 = \sqrt{\sum_{j=1}^{20} T_{6j}^2}$ is to be minimized.

The 17 equations of condition for orders 1 through 5 are given in Table–1. In this table operation "*" may understood as component-wise multiplication: $[u_1 \quad u_2 \cdots u_n]^\top * [v_1 \quad v_2 \cdots v_n]^\top = [u_1 v_1 \quad u_2 v_2 \cdots u_n v_n]^\top$. This operation has the less priority. Parentheses, powers and dot products are always evaluated before "*" Notice that $c^j = \underbrace{c * c * \cdots * c}_{j \text{ times}}$.

An explicit RK$5(4)$ pair uses 7 stages. Using FSAL property the last stage is reused as the first of the next step. So effectively only 6 stages needed per step. FSAL determines $c_7 = 1$ and $a_{7j} = b_j,\ j = 1, 2, \ldots, 6$. Finally 27 coefficients are available for satisfaction of the 25 constrains. Namely $c_2,\ c_3,\ c_4,\ c_5,\ c_6,$

Table 1 The truncation error coefficients for orders 1 through 5.

$T_{1,1} = be - 1$, $T_{2,1} = bc - \frac{1}{2}$, $T_{3,1} = \frac{1}{2}bc^2 - \frac{1}{6}$, $T_{3,2} = bAc - \frac{1}{6}$,
$T_{4,1} = \frac{1}{4}bc^3 - \frac{1}{24}$, $T_{4,2} = b(c * Ac) - \frac{1}{8}$, $T_{4,3} = \frac{1}{2}bAc^2 - \frac{1}{24}$, $T_{4,4} = bA^2c - \frac{1}{24}$,
$T_{5,1} = \frac{1}{24}bc^4 - \frac{1}{120}$, $T_{5,2} = \frac{1}{2}b(c^2 * Ac) - \frac{1}{20}$, $T_{5,3} = \frac{1}{2}b(Ac)^2 - \frac{1}{40}$, $T_{5,4} = \frac{1}{2}b(c * Ac^2) - \frac{1}{30}$,
$T_{5,5} = b(c * A^2c) - \frac{1}{30}$, $\widetilde{T}_{5,6} = \frac{1}{6}bAc^3 - \frac{1}{120}$, $\widetilde{T}_{5,7} = bA(c * Ac) - \frac{1}{40}$, $\widetilde{T}_{5,8} = \frac{1}{2}bA^2c^2 - \frac{1}{120}$,
$\widetilde{T}_{5,9} = bA^3c - \frac{1}{120}$.

$b_1,\ b_2,\ b_3,\ b_4,\ b_5,\ b_6$, $\hat{b}_1,\ \hat{b}_2,\ \hat{b}_3,\ \hat{b}_4,\ \hat{b}_5,\ \hat{b}_6,\ a_{32},\ a_{42}$, $a_{43},\ a_{52},\ a_{53},\ a_{54}$, $a_{62},\ a_{63},\ a_{64},$ and a_{65}. We set $\hat{b}_7 = 1/20 \neq b_7 = 0$ in order to avoid coincidence of vectors b and $\hat{b}$.

Traditionally simplifying assumptions are made in order to reduce the difficulty of the problem. These define various families of solutions. Especially $Ac = c^2/2$ and $b_2 = \hat{b}_2 = 0$, are common in all families ever appeared. Using for example the latter assumptions then $T_{3,2}$ coincides with $T_{3,1}$ and need not to be satisfied separately.

3 Differential Evolution

Differential Evolution (DE) [15] is a population – based algorithm for global optimization. It attains a set of possible solutions, for the problem under consideration, and evolves them using some operators in order to find the optimum one.

More specifically, in the DE algorithm terminology, each solution is called individual and the set of all the possible solutions is called population of individuals. Let u_i denote the ith individual of the population. Then, DE is described by the following four distinct steps:

Step 1. (Initialization step) Initialize randomly the individuals of the population. Set the mutation factor, *F*, and the crossover factor, *CR*, to fixed values within the interval $[0, 1]$ and choose an *objective function* for the problem under study.

Step 2. (Mutation step) Mutate each individual u_g^i (called the target individual) of the population to form a trial vector, v_{g+1}^i, by applying one of the following operators,

$$\begin{aligned}
v_{g+1}^i &= u_g^{r1} + F(u_g^{r2} - u_g^{r3}),\\
v_{g+1}^i &= u_g^{r1} + F(u_g^{r2} - u_g^{r3}) + F(u_g^{r4} - u_g^{r5}),\\
v_{g+1}^i &= u_g^{\text{best}} + F(u_g^{r1} - u_g^{r2}),\\
v_{g+1}^i &= u_g^i + F(u_g^i - u_g^{\text{best}}) + F(u_g^{r1} - u_g^{r2}),\\
v_{g+1}^i &= u_g^{\text{best}} + F(u_g^{r1} - u_g^{r2}) + F(u_g^{r3} - u_g^{r4}),
\end{aligned}$$

where $r1, r2, r3, r4$ are random integers such that, $r1 \neq r2 \neq r3 \neq r4 \neq i \neq best$. The index best is used to represent the individual with the optimum objective function value in the current population.

Step 3. (Recombination (Crossover) step) For each element of the trial vector, v_{g+1}^i, obtain a random value, $r \in [0, 1]$. If $r \leqslant CR$, set $u_{g+1}^i = v_{g+1}^i$, otherwise set $u_{g+1}^i = u_g^i$ [12].

Step 4. (Selection step) For each individual of the population u_{g+1}^i evaluate its value through the objective function. If this value is better than the one of the target individual u_g^i, then the individual u_{g+1}^i replaces the target individual in the next generation. Otherwise, the target individual is retained in the next generation of the DE algorithm. If the termination criterion is not satisfied, then go to the second step. As a termination criterion we can use a predefined number of generations or an error goal value of the objective function.

4 Proposed Method

From the Evolutionary Algorithms, here, we will use the Differential Evolution (DE) algorithm described in Section 3. The individuals of DE will be candidate vectors of RK coefficients. The objective function will be the

sum of the absolute values of the local truncation error equations. Using the ℓ_1 norm instead of the ℓ_2 norm (used in [1, 2]) we avoid the square increment of the error. Moreover, the usage of the ℓ_1 norm does not affect proposed method's operation. The individuals - RK coefficients will be evolved according to the DE steps in order to find the one with the smallest possible local truncation error. To assure that each individual satisfies the algebraic system of RK order conditions we shall make use once again of the DE algorithm. The aim of the second DE algorithm will be to solve the optimization problem that arises if we take as objective function the sum of the ℓ_1 norm of the RK order conditions. Thus for each individual (initial individual) of the first DE, we initialize in a neighborhood around this individual the population of the second DE. The output of the second DE will be an individual-RK coefficients that satisfy the RK order conditions within some precision. This output-individual will replace the initial individual in the first DE and then the new individual will be evaluated by computing the absolute values of the local truncation error equations.

A high level description of the proposed method is given by the following steps:

Step 1. Initialize randomly the population of the DE algorithm.

Step 2. For each individual of this population

Step 2a. Initialize a local population in a neighborhood around the individual.

Step 2b. The individuals of the local population are evolved according to the steps of the DE algorithm. The objective function will be the the sum of the absolute values of the RK order conditions. The DE algorithm will terminate when the RK order conditions are satisfied.

Step 2c. Replace the individual with the best individual found from the local population.

Step 3. Evaluate each individual using as objective function the sum of the absolute values of the RK local truncation error equations.

Step 4. Evolve the individuals according to the DE steps. If the new individual is not a RK method, apply Steps 2a–2c in order to satisfy the RK order conditions.

Applying the above described method to the set of equations for a RK pair of orders $5(4)$ we constructed a method presented in Table–2. This pair does not belong to any known family of solutions.

Table 2 The coefficients of the new pair.

$c_2 = 0.20427299377517$	$c_3 = 0.30783191706654$	$c_4 = 0.79929778653982$
$c_5 = 0.90075677981782$	$c_6 = c_7 = 1$	$b_1 = 0.09305053181522$
$b_2 = 0.00875587209919$	$b_3 = 0.44668491608246$	$b_4 = 0.57565550417358$
$b_5 = -0.24922802364186$	$b_6 = 0.12508119947141$	$\widehat{b}_1 = 0.09213862611890$
$\widehat{b}_2 = 0.00802050182998$	$\widehat{b}_3 = 0.45114229138473$	$\widehat{b}_4 = 0.54617230669534$
$\widehat{b}_5 = -0.20560640649727$	$\widehat{b}_6 = 0.08313268046832$	$\widehat{b}_7 = \frac{1}{40},\ b_7 = 0$
$a_{32} = 0.23518899782085$	$a_{42} = -3.41195602838532$	$a_{43} = 3.2962302133491$
$a_{52} = -13.10298053365142$	$a_{53} = 11.03548248550554$	$a_{54} = -0.40092201922022$
$a_{62} = -11.18958543937$	$a_{63} = 9.29028080486981$	$a_{64} = 0.12483355979476$
$a_{65} = -0.19774507863999$	$a_{i1} = c_i - \sum_{j=2}^{j=i-1} a_{ij},\ j = 1, 2, \cdots, 6$	$a_{7i} = b_i,\ i = 1, 2, \cdots, 6$

5 Numerical Results–Conclusions

We run the Runge-Kutta pair for the 25 DETEST [9] non–stiff problems and for tolerances $10^{-2}, 10^{-3}, \cdots, 10^{-7}$. DETEST was implemented through MATLAB–7 on a Pentium IV computer running Windows XP Professional at 3.4GHz. For comparison purposes the DP$5(4)$ pair [7] was also run for the same tolerances. We present the results in Table–3.

These results were developed according to the guidelines given in [14]. So, we notify the percentage difference in the number of function evaluations required for achieving a given end-point global error for each problem. Unity represents 10%. Numbers have been rounded to the nearest digit. Positive numbers mean that the first method is superior. The final row, gives the mean value of efficiency gain for all tolerances in a problem. The

Table 3 Efficiency gains of NEW$5(4)$ relative to DP$5(4)$, for the range of tolerances 10^{-2}, ..., 10^{-7}.

g.e.	A1	A2	A3	A4	A5	B1	B2	B3	B4	B5	C1	C2	C3	C4	C5	D1	D2	D3	D4	D5	E1	E2	E3	E4	E5
-1			0			-2	0		-6	-1					0	0	0	0	0	-1	0	0	-1		
-2	0	0	0	0	0	-1	0	0	-4	0	0	0	0	0	0	0	0	0	-1	-2	0	0	-1		
-3	0	0	-1	0	0	-1	0	0	-3	0	0	0	1	0	0	0	-1	-1	-1	-2	0	0	-1	0	
-4	0	0	-1	0	0	0	0	0	-1	0	0	0	1	0	0	0	-1	-2	-2		1	0	0	0	-1
-5	0	0	-2	0	0	0	0	0	0	0	0	0	2	0	0	0	-2				1	0	0	0	-2
-6	0	0	-2	0	0		0	0	1	1	0	0		0	0						1	-1	0	0	-2
-7	0	0		0	0		0	0			0	0		0	0						1			0	-2
-8	0	0					1	0			0			0										0	-2
-3%	0	0	-1	0	0	-1	0	0	-2	0	0	0	1	0	0	0	-1	-1	-1	-2	1	0	-1	0	-2

left most lower 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 [11] for more details.

We finally observe that the new method is in average 3% less efficient than Dormand–Prince $5(4)$ for the DETEST problems. It is remarkable that a pair not belonging to any known family of solutions gives so good results.

Acknowledgment

This work was supported by the "Pythagoras I" research grant co funded by the European Social Fund and National Resources.

References

[1] G. S. Androulakis and T. N. Grapsa and M. N. Vrahatis, Generating optimal Runge–Kutta methods, VSP International Science Publishers, Zeist, The Netherlands, pp 1–7 (1996)
[2] G. S. Androulakis and M. N. Vrahatis, A generator of optimal Runge–Kutta methods, Science Culture Technology Publishing, Oxford Graphic Printers, pp 3–12 (1995)
[3] J. C. Butcher, Coefficients for the study of Runge-Kutta integration processes, J. Austr. Math. Soc., **3**, 185–201 (1963)
[4] J. C. Butcher, Implicit Runge-Kutta processes, Math. Comput., 18, 50-64 (1964)
[5] J. C. Butcher, On Runge-Kutta processes of high order, J. Austral. Math. Soc., 4, 179-194 (1964)
[6] J. C. Butcher, The Numerical Analysis of Ordinary Differential Equations. Runge-Kutta & General Linear methods, (John Wiley & sons, Chichester, 1987)
[7] J. R. Dormand and P. J. Prince, A family of embedded Runge-Kutta formulae, J. Comput. Appl. Math., **6**, 19–26 (1980)
[8] A. P. Engelbrecht, Computational intelligence: An introduction, John Wiley and Sons (2002)
[9] W. H. Enright and J. D. Pryce, Two Fortran Packages for Assessing Initial Value Methods, ACM TOMS, 13, 1-27 (1987)
[10] E. Fehlberg, Low order classical Runge-Kutta formulas with stepsize control and their application to some heat-transfer problems, TR R-287, NASA, (1969)
[11] S. N. Papakostas, Ch. Tsitouras and G. Papageorgiou, A general family of Runge-Kutta pairs of orders 6(5), SIAM J. Numer. Anal., **33**, 917–926 (1996)
[12] V. P. Plagianakos and M.N Vrahatis, Parallel evolutionary training algorithms for "hardware friendly" neural networks, Natural Computing, **1**, 307–322 (2002)
[13] P. J. Prince and J. R. Dormand, High order embedded Runge-Kutta formulae, J. Comput. Appl Math., **7**, 67–75 (1981)
[14] P. Sharp, Numerical comparisons of some explicit Runge–Kutta pairs, ACM TOMS, **17**, 387-409 (1991)
[15] R. Storn and K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Glob. Optim. **11**, 341–359 (1997)
[16] Ch. Tsitouras, Optimal Runge–Kutta pairs of orders 9(8), Appl. Numer. Math., **38**, 123–134 (2001)
[17] J. H. Verner, Explicit Runge-Kutta methods with estimates of the local truncation error, SIAM J. Numer. Anal., **15**, 772–790 (1978)