MATEC Web of Conferences **41**, 05001 (2016)
DOI: 10.1051/matecconf/20164105001
© Owned by the authors, published by EDP Sciences, 2016

# On the modification of Differential Evolution strategy for the construction of Runge Kutta pairs

Ioannis Th. Famelis[1,a] and Ch. Tsitouras[2,b]

[1] *TEI of Athens, Dept. of Electronic Engineering, GR12210 Athens, Greece*
[2] *TEI of Sterea Hellas, Dept. of Automation Engineering, GR34400 Psahna Campus, Greece*

**Abstract.** Among the most popular methods for the solution of the Initial Value Problem are the Runge–Kutta (RK) pairs. These methods can be derived by solving a system of nonlinear equations after admitting various simplifying assumptions. The more simplifying assumptions we consider the more we narrow the solution space. In [1] Tsitouras presented an algorithm for the construction of Runge–Kutta pairs of orders 5 and 4 satisfying only the so called "first column simplifying assumption". In [2] Famelis and Tsitouras have studied the ability of Differential Evolution techniques to find solutions satisfying all the order conditions needed for the derivation of orders 5 and 4 pairs. In this work we propose an modification on the Differential Evolution strategy for the same problem. The current study will be a guide to the construction of other classes of RK that have not been presented in the literature.

## 1 Introduction

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

$$y' = f(x, y), \; y(x_0) = y_0 \in \mathbb{R}^m, \; x \geq x_0 \tag{1}$$

where the function $f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$ is assumed to be as smooth as necessary. For the approximate solution of the problem ( 1), the general $s-$stage embedded explicit Runge-Kutta pair of orders $p(p-1)$ advances the approximation of the solution from $x_n$ to $x_{n+1} = x_n + h_n$, $n = 0, 1, 2, \ldots$ using the formulae

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

where

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

As in both formulae $c_i, a_{ij}$ are common the coefficients $\hat{b}_j, \; b_j$ define the $(p-1)-$th and $p-$th order approximations respectively. The coefficients of such a pair of methods can be presented in a matrix-array form using the following Butcher Tableau [3, 4]

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

where $A \in \mathbb{R}^{s \times s}$ is strictly lower triangular, the column vectors $b^T$, $\hat{b}^T$, $\in \mathbb{R}^s$ and the line vector $c \in \mathbb{R}^s$ satisfies

$$c = Ae, \; e = [1, \; 1, \cdots, 1]^T \in \mathbb{R}^s.$$

Throughout this paper, we assume that local extrapolation is applied e.g. the $p-$th order approximation advances the integration. For the step size selection mechanism the local error estimate $E_n = \|y_n - \hat{y}_n\|$ is used. Given a tolerance parameter $TOL$, when $E_n \leq TOL$ the mechanism

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

furnishes the next step length. In case of $E_n > TOL$, the current step is rejected and a new approximation of $y_{n+1}$ is computed using as step size $h_n$ the outcome of the above formula.

In case that $c_s = 1$, $a_{s,j} = b_j$ for $j = 1, 2, \cdots, s-1$ and $b_s = 0 \neq \hat{b}_s$ the **F**irst **S**tage of each new step is the same **A**s the **L**ast one of the previous stage. This device called FSAL, possibly introduced in [5], effectively reduces the stages of the pair by one to $s-1$.

The Local Truncation Error (LTE) $e_{n+1}$ is the error of the integration when $y_n = y(x_n)$ e.g. the previous step value is exact. LTE associated with a $p-$th order RK method is

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

where

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

[a] e-mail: ifamelis@teiath.gr
[b] e-mail: tsitouras@mail.teiste.gr

Article available at http://www.matec-conferences.org or http://dx.doi.org/10.1051/matecconf/20164105001

**Table 1.** The order conditions RK5(4).

| | | |
|---|---|---|
| $be = 1$ | $bc^4 = \frac{1}{5}$ | $\hat{b}e = 1$ |
| $bc = \frac{1}{2}$ | $b(c^2 * Ac) = \frac{1}{10}$ | $\hat{b}c = \frac{1}{2}$ |
| $bc^2 = \frac{1}{3}$ | $b(Ac)^2 = \frac{1}{20}$ | $\hat{b}c^2 = \frac{1}{3}$ |
| $bAc = \frac{1}{6}$ | $b(c * Ac^2) = \frac{1}{15}$ | $\hat{b}Ac = \frac{1}{6}$ |
| $bc^3 = \frac{1}{4}$ | $b(c * A^2c) = \frac{1}{30}$ | $\hat{b}c^3 = \frac{1}{4}$ |
| $b(c * Ac) = \frac{1}{8}$ | $bAc^3 = \frac{1}{20}$ | $\hat{b}(c * Ac) = \frac{1}{8}$ |
| $bAc^2 = \frac{1}{12}$ | $bA(c * Ac) = \frac{1}{40}$ | $\hat{b}Ac^2 = \frac{1}{12}$ |
| $bA^2c = \frac{1}{24}$ | $bA^2c^2 = \frac{1}{60}$ | $\hat{b}A^2c = \frac{1}{24}$ |
| | $bA^3c = \frac{1}{120}$ | |

with $Q_{qi}$ algebraic functions of $A,\ b,\ c$ and $\xi_{qi}$ 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 $i = 1, 2, \cdots, \lambda_q$. $\lambda_q$ is the number of elementary differentials for each order which coincides with the number of rooted trees of order $q$. It is known that $\lambda_1 = 1,\ \lambda_2 = 1,\ \lambda_3 = 2,\ \lambda_4 = 4,\ \lambda_5 = 9,\ \lambda_6 = 20,\ \lambda_7 = 48\cdots$, etc [6].

The construction of an effectively 6−stages FSAL Runge-Kutta pair of orders 5(4) (RK5(4)) requires the solution of a nonlinear system of 25 equations e.g. $\lambda_1 + \cdots + \lambda_5 = 17$ order conditions for the higher order formula and $\lambda_1 + \cdots + \lambda_4 = 8$ order conditions for the lower order formula. In Table 1 we present the 25 order conditions of a RK5(4) pair in matrix operation form. In these equations, the operation "*" is to be understood as component–wise multiplication and the power of an vector as a component–wise power, e.g. $c^2 = c * c$. The rest multiplications and the matrix powers are the known from linear algebra vector-matrix operations. So, our problem has a total of 28 parameters (unknowns), namely $c_2, \ldots, c_6$, $b_1, \ldots, b_6$, $\hat{b}_1, \ldots, \hat{b}_7$, $a_{32}, a_{42}, a_{43}, a_{52}, a_{53}, a_{54}$ and $a_{62}, \ldots, a_{65}$.

The exact solution of the resulted nonlinear system is out of question. Only after considering simplifying assumptions, we can use nonlinear optimization techniques to get accurate enough solutions. In practice nonlinear optimization techniques based on some kind estimation of derivatives such as conjugate gradient, back-propagation or other Newton-type methods methods are not easily applicable due to the nature of the problem. Instead nonlinear optimizers based to stochastic direct search seem to work very efficiently.

## 2 Differential Evolution

Optimization methods can be divided in two large classes. The former is continuous optimization where the search area and solutions are presumed to be situated in a certain continuous space with its metrics. The later is combinatorial optimization where the search area is limited by a finite number of feasible solutions. Depending on the nature of the objective function and the constrains, the former class is subdivided into linear programming , quadratic programming and nonlinear programming methods. The last subclass consists of local search methods and global methods.

The global methods can be either classical methods, where the global search is successfully realized as a sequence of solution of local optimization problems. Alternatively, we have metaheuristic methods which can be either population-based as Differential Evolution method, Particle Swarm method and genetic algorithms or neighborhood-based methods.

Combinatorial methods can either be exact methods where the enumeration of all sets of solutions results the global optimum. Due to computational cost such methods are appropriate for small scale problems. Alternatively we can consider approximate methods where a partial enumeration leads to a near to optimum solution with a bias. Approximate methods can be either heuristic which are designed for a certain problem and cannot usually used for other problems or metaheuristic algorithms. Such procedures can be used when we have mixed (continuous and discrete) parameters.

**Differential Evolution (DE)** [7–9] is a population based metaheuristic method which has become more and more popular for problems that either classical continuous and combinatorial methods fail to solve. Its virtues are that a) they do not require special conditions for the properties of the objective functions and the constrains, b) they can be applied in both continuous and combinatorial problems and c) they are extensible on multimodal and multiobjective optimization. Sensitivity of the process to the control parameters and the possible high computational cost are its drawbacks.

In an optimization problem we have an objective function (usually called a **fitness function**) $f : B \subseteq \mathbb{R}^D \rightarrow \mathbb{R}$ for which we want to find a minimum point $X_{gmin} \in B$ where $f$ attains its global minimum subject to some inequality constrains. $B$ is the set of feasible points that satisfy the constrains. Our aim is to satisfy an optimization criterion which consists of the fitness function and the constrains. Usually nonlinear problems have many local minimum so the approximate problem solution is to find a $X_{appr} \in B$ which satisfies the constrains and $f(X_{appr})$ has a desirable precision $VTR$ (value-to-reach) e.g. $|f(X_{appr})| \leq VTR$.

DE is a very powerful tool for global optimization which the applies a procedure of evolution of a population of individuals in a intelligent manner. DE disposes three control parameters the **population size** $NP$, the **differentiation constant** $F$ and the **crossover constant** $Cr$. Even though DE is more robust regarding control parameters (compared to Particle Swarm optimization or evolutionary algorithms) the proper choice of the control parameter values improves the procedure's convergence.

DE is an iterative process where in each iteration, called **generation** $g$, we work with a "**population**" of individuals $\mathbb{P}^g = \{X^g_{cin} \in B,\ cin = 1, 2, \ldots, NP\}$. Every member of this population is a potentially optimal solution and it is a set of $D$ **gens** e.g. $X^g_{cin} = \{x^g_{cin,j}\ j = 1, 2, \ldots, D\}$. In the first step of the algorithm, the **initialization**, an initial population $\mathbb{P}^0$ is considered, usually randomly and the fitness function is evaluated on it. Then, in each iteration (generation) a reproduction scheme updates all the individuals of the $\mathbb{P}^g$ performing a sequential procedure with three phases: the **Differentiation**, the **Crossover** and the **Selection**.

In **Differentiation** phase three (up to five ) individuals $X_1^g, \ldots, X_5^g$ are chosen from the population. The Differentiation strategy is defined by the way of choice which can be based on random, directed, local or hybrid criteria. For each individual $X_{cin}^g \in \mathbf{P}^g$ the result of the Differentiation is the trial individual:

$$\omega_{cin}^g = \beta + F \cdot \delta,$$

where F is the Differentiation constant, $\beta$ is the base vector and $\delta$ is the difference vector. The base vector can be chosen either randomly, without taking any information about the values of the fitness function, or locally by choosing either $X_{cin}^g$ or the best $X_{best}^g$ individual of the generation. The difference vector can be formed either randomly or by using the values of the objective function to determine a direction which can be viewed as an imitation of the gradient function. Combination of these selections in the difference vector formation has been proposed too [9]. Thee most popular strategies are the following:

**Strategy 1** *best/rand2* where $\beta = X_{best}^g$ and $\delta = X_1^g - X_2^g$.

**Strategy 2** *rand1/rand2* where $\beta = X_1^g$ and $\delta = X_2^g - X_3^g$.

**Strategy 3** *local/rand2,best1,dir1* where $\beta = X_{cin}^g$ and $\delta = X_{best}^g - X_{cin}^g + X_1^g - X_2^g$.

**Strategy 4** *best/rand4* where $\beta = X_{best}^g$ and $\delta = X_1^g - X_2^g + X_3^g - X_4^g$.

**Strategy 5** *rand1/rand4* where $\beta = X_5^g$ and $\delta = X_1^g - X_2^g + X_3^g - X_4^g$.

**Strategy 6** *local/hybrid, linear crossover combination of local,dir1 and rand2* where

$$\beta = X_{cin}^g \text{ and } F \cdot \delta = F_{cr}(X_{best}^g - X_{cin}^g) + F(X_1^g - X_2^g).$$

In **Crossover** phase the trial individual $\omega_{cin}^g$ is recombined with $X_{cin}^g$ and a new trial individual $\omega_{v_{cin}}^g$ is formed by inheriting its gens by using the following probabilistic rule:

$$\omega_{v_{cin,j}}^g = \begin{cases} \omega_{cin,j}^g, & \text{if } rand_j \geq Cr \\ X_{cin,j}^g, & \text{otherwise} \end{cases}$$

where $j = 1, 2, \ldots, D$, $Cr \in [0, 1]$ the Crossover constant and $rand_j \in [0, 1)$ a random number.

In **Selection** phase the new trial individual replaces $X_{cin}^g$ if it attains a smaller fitness value. The algorithm iteration terminates when a stopping criterion, such as a maximum number of generations is reached or $VTR$ criterion is satisfied [9].

The numerical experiments presented in [2] revealed that the trial individual the $\omega_{cin}^g$ should be considered having a linear combination of both $X_{cin}^g$ and $X_{best}^g$. The bigger the participation of the best individual is, the less generations we need to have an acceptable solution but we loose on the percentage of successes. Whereas, the Strategy 6, with a randomly in [0, 1] (as suggested in the literature) choice of the contribution of $X_{cin}^g$ and $X_{best}^g$ in the linear combination, proves to be very efficient.

**Table 2.** Numerical Experiments Results

| Strategy | % success | Mean of Gens |
|---|---|---|
| 1 or 6 with $F_{cr} = 1$ | 24 | 21584 |
| 3 or 6 with $F_{cr} = 0.8$ | 47 | 20268 |
| 6 with $F_{cr} = 0.2$ | 53 | 47333 |
| 6 with $F_{cr} = 0.5$ | 55 | 23999 |
| 6, random $F_{cr} \in [0, 1]$ | 49 | 24607 |
| 6, random $F_{cr} \in [0.5, 1]$ | 49 | 20158 |
| 6, random $F_{cr} \in [0, 0.5]$ | 56 | 40496 |
| 6, random $F_{cr} \in [0.3, 0.7]$ | 54 | 24972 |

## 3 Numerical Experiments and Conclusions

For the construction of the RK5(4) pair considering the Table 1 order conditions we have a fitness function

$$f(x) = \sqrt{\sum_{q=1}^{5} \sum_{i=1}^{\lambda_q} (Q_{qi} - \xi_{qi}/q!)^2 + \sum_{q=1}^{4} \sum_{i=1}^{\lambda_q} (\hat{Q}_{qi} - \xi_{qi}/q!)^2}$$

subject to linear constrains which keep the coefficients of the method in appropriate limits. In order to construct effective methods of the desired order as $VTR$ we consider computer arithmetic epsilon. We set an arbitrary value for $\hat{b}_7$ and so the dimension of the real precision parameters $D = 27$. I We set the population size of the population $NP = 270$ and the maximum number of generations equal to 100000. We experiment with various modifications of the random choice in Strategy 6. We apply the DE procedure for 500 times and we record the average number of convergence of the procedure having reached the desired $VTR$ accuracy and the average number of generations needed. Our numerical results (See Table 2) reveal that for both robustness and fast convergence we should modify Strategy 6 and choose the $F_{cr}$ randomly in [0.5, 1].

## Acknowledgements

This research has been co-funded by the European Union (European Social Fund) and Greek national resources through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program : ARCHIMEDES III. Investing in knowledge society through the European Social Fund (grand no.383779/2).

## References

[1] Ch. Tsitouras, Runge–Kutta pairs of orders 5(4) satisfyning only the first column simplifyiyng assumption, Computers and Mathematics with Applications, 62 (2011), 770-775.

[2] I. Th. Famelis and Ch. Tsitouras, Differential Evolution for the derivation of Runge Kutta pairs., (ICNAAM 2014) AIP Conference Proceedings 1648, 740004 (2015); doi: 10.1063/1.4912959

[3] J. C. Butcher, Implicit Runge-Kutta processes, Math. Comput., 18 (1964), 50-64.

[4] J. C. Butcher, On Runge-Kutta processes of high order, J. Austral. Math. Soc., 4 (1964), 179-194.

[5] E. Fehlberg, Low order classical Runge-Kutta formulas with stepsize control and their application to some heat-transfer problems, TR R-287, NASA, (1969)

[6] J. C. Butcher, Coefficients for the study of Runge-Kutta integration processes, J. Austr. Math. Soc., 3 (1963), 185-201.

[7] R. M. Storn and K. V. Price, Differential Evlution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces, Journal of Global Optimization 11 (1997), 341-359.

[8] K. V. Price, R. M. Storn and J. A. Lampinen, Differential Evolution: A practical approach to global optimization, Springer (2005).

[9] V. Feoktistov, Differential Evolution, In Search of Solutions, Springer (2006).