International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:8, No:8, 2014

A Method for Improving the Embedded Runge Kutta Fehlberg 4(5)

Sunyoung Bu, Wonkyu Chung, Philsu Kim

Abstract—In this paper, we introduce a method for improving the embedded Runge-Kutta-Fehlberg4(5) method. At each integration step, the proposed method is comprised of two equations for the solution and the error, respectively. These solution and error are obtained by solving an initial value problem whose solution has the information of the error at each integration step. The constructed algorithm controls both the error and the time step size simultaneously and possesses a good performance in the computational cost compared to the original method. For the assessment of the effectiveness, EULR problem is numerically solved.

Keywords—Embedded Runge-Kutta-Fehlberg method, Initial value problem.

I. Introduction

THE embedded Runge-Kutta (ERK) method is a popular strategy for solving the initial value problem described by

$$\frac{d\phi}{dt} = f(t,\phi(t)), \qquad t \in [t_0, t_f]; \qquad \phi(t_0) = \phi_0, \qquad (1)$$

where f has continuously bounded partial derivatives up to required order for the developed numerical method. Most ERKs use two Runge-Kutta methods with different orders p and q, simply denoted by RK$p(q)$. In most cases, $q > p$ and the low order RKp method and the high order RKq method are applied to calculate the approximate solution ϕ_{m+1} and the local truncation error $E_{m+1} := \phi(t_{m+1}) - \phi_{m+1}$, respectively, at time t_{m+1} together with the information of ϕ_m at time t_m. Hence, the existing mechanism of ERK algorithm at each integration step may be described by

$$\begin{cases} \phi_{m+1} = F(\phi_m), \\ e_{m+1} = G(\phi_m, \phi_{m+1}), \end{cases} \qquad (2)$$

where F and G are functions derived from the numerical methods. Another important factor of ERK is to control the size of each integration step, for which an accurate and efficient scheme for calculating e_{m+1} is quite important, and RKq uses the same function values of RKp to reduce the computational cost. There are many research literatures concerning the technique selecting the time step size appropriately (for example, see [1], [2], [3], [5], [6], [7], [8]).

The aim of this paper is to introduce an improved algorithm of (2). In particular, we develop a method embedding the error in the algorithm (2). Practically, we consider the Butcher array for the Runge-Kutta Fehlbeg 4(5) pair given by

$$\begin{array}{c|c} \mathbf{c} & \mathcal{A} \\ \hline & \mathbf{b} \\ & \hat{\mathbf{b}}, \end{array} \qquad (3)$$

where

$$\mathbf{c} = [c_1, c_2, \cdots, c_6]^T = [0, \frac{1}{4}, \frac{3}{8}, \frac{12}{13}, 1, \frac{1}{2}]^T$$
$$\mathbf{b} = [b_1, b_2, \cdots, b_6] = [\frac{25}{216}, 0, \frac{1408}{2565}, \frac{2197}{4104}, -\frac{1}{5}, 0]$$
$$\hat{\mathbf{b}} = [\hat{b}_1, \hat{b}_2, \cdots, \hat{b}_6] = [\frac{16}{135}, 0, \frac{6656}{12825}, \frac{28561}{56430}, -\frac{9}{50}, \frac{2}{55}],$$
$$\mathcal{A} = (\alpha_{i,j})_{6\times 6} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{32} & \frac{9}{32} & 0 & 0 & 0 & 0 \\ \frac{1932}{2197} & -\frac{7200}{2197} & \frac{7296}{2197} & 0 & 0 & 0 \\ \frac{439}{216} & -8 & \frac{3680}{513} & -\frac{845}{4104} & 0 & 0 \\ -\frac{8}{27} & 2 & -\frac{3544}{2565} & \frac{1859}{4104} & -\frac{11}{40} & 0 \end{pmatrix}. \qquad (4)$$

Let us assume that an approximate solution ϕ_m and an estimated error e_m for the actual error $E_m := \phi(t_m) - \phi_m$ at time t_m are already calculated. More precisely, we assume that ϕ_m is calculated by the RK4 and e_m is obtained by the difference between the solutions calculated by RKF4 and RKF5, which is the basic process of ERK. Now, we are ready to introduce a strategy to calculate the next step values ϕ_{m+1} and e_{m+1} at time t_{m+1}. Let $h = t_{m+1} - t_m$, then $E_m = \mathcal{O}(h^5) > \mathcal{O}(h^6) = E_m - e_m$. Thus, from the relation

$$\phi(t_m) = \phi_m + E_m = \phi_m + e_m + E_m - e_m,$$

we give a guess that $\phi_m + e_m$ is a more accurate approximation of $\phi(t_m)$ than ϕ_m. Therefore, at the integration step $[t_m, t_{m+1}]$, it is reasonable to solve the initial value problem

$$\begin{cases} \psi'(t) = f(t,\psi(t)), & t \in [t_m, t_{m+1}], \\ \psi(t_m) = \phi_m + e_m \end{cases} \qquad (5)$$

instead of

$$\begin{cases} \psi'(t) = f(t,\psi(t)), & t \in [t_m, t_{m+1}], \\ \psi(t_m) = \phi_m \end{cases} \qquad (6)$$

to find the approximation ϕ_{m+1}. Hence, as the embedded RKF4(5) with the Butcher array given by (3), we first solve the problem (5) with RKF4 to get the approximate solution ϕ_{m+1} and also solve (5) again with RKF5 to get the estimated

Sunyoung Bu, Wonkyu Chung, Philsu Kim are with the Department of Mathematics, Kyungpook National University, Daegu 720- 710, Korea (e-mail: syboo@knu.ac.kr, wkchung@knu.ac.kr, kimps@knu.ac.kr).

error e_{m+1}. Summarizing the procedures, we get the following error embedded Runge-Kutta (EERK) method

$$\begin{cases} \phi_{m+1} = \phi_m + e_m + h\sum_{i=1}^{6} b_i k_i, \\ e_{m+1} = h\sum_{i=1}^{6}\Big(\hat{b}_i - b_i\Big)k_i \end{cases} \tag{7}$$

where

$$k_i = f\Big(t_m + c_i h, \phi_m + e_m + h\sum_{j=1}^{i-1}\alpha_{i,j}k_j\Big), \quad i = 1, \cdots, 6. \tag{8}$$

So far, the existing method solves the perturbed IVP (6) at each integration step. That is, after the first step, the estimated error e_m is accumulated as the time step is going on. So, we reduce this accumulation of the estimated errors by embedding them at each integration step. Thus, we can get more smaller global error. In other words, by giving the usage of estimated error, we can improve the capability of the existing method, while existing method uses the estimated error only for the step-size selection.

II. Numerical Result

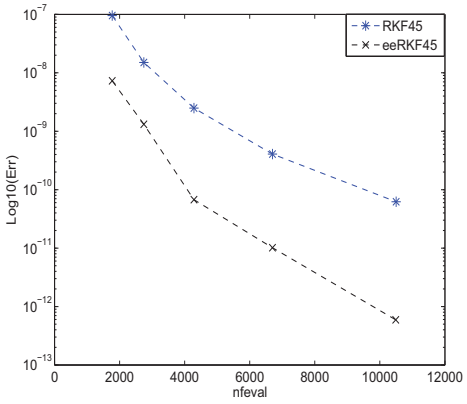
To assess the improvement and effectiveness of the proposed scheme, we consider the well known EULR problem ([4]), Euler's equation of rotation of a rigid body, given by

$$\begin{cases} I_1 y_1' &= (I_2 - I_3)y_2 y_3, \\ I_2 y_2' &= (I_3 - I_1)y_3 y_1, \\ I_3 y_3' &= (I_1 - I_2)y_1 y_2 + f(t), \end{cases} \tag{9}$$

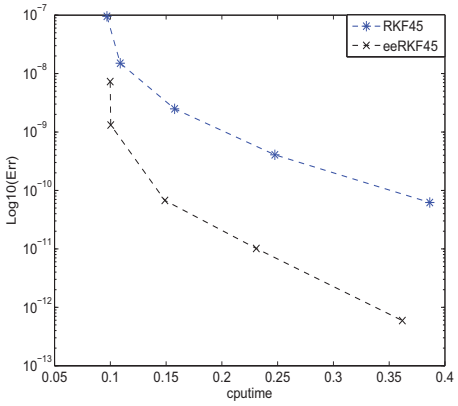
where y_1, y_2, y_3 are the coordinates of $\overrightarrow{\omega}$, the rotation vector, and I_1, I_2, I_3 are the principal moments of inertia. The third coordinates has an additional exterior force

$$f(x) = \begin{cases} 0.25\sin^2 t, & t \in [3\pi, 4\pi] \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

which is discontinuous in its second derivative. We choose the constants and initial values as $I_1 = 0.5,\ I_2 = 2,\ I_3 = 3,\ y_1(0) = 1,\ y_2(0) = 0,\ y_3(0) = 0.9$. We solve the problem on the time interval $[0, 10]$ and use the numerical solutions as the sum of the approximate solution ϕ_m and the estimated error e_m to give more accurate results in each method. As a measure of the effectiveness, we calculate the required number of function evaluations (nfeval) and the computational time (cputime) to solve the problem. For given relative and absolute tolerances, we calculate the L_2 norm for the absolute error in log-scale at the final time for each problem and also the required nfeval and cputime. We solve the problem with the improved RKF4(5) (eeRKF45) and the embedded RKF4(5) (RKF45) by varying the relative tolerance from 1.0e-9 to 1.0e-13 and absolute tolerance from 1.0e-11 to 1.0e-15. The reference solution is calculated by DOP853 with the tolerances $Atol = Rtol = eps$, where eps is the double precision of floating numbers in MATLAB. The numerical results are displayed on Fig.1 (a) and (b), where y and x axes represent the absolute errors and either nfeval or cputime, respectively. Also, all the marked points from left to right are corresponding to the given tolerances from large to small, respectively. One can see that the proposed scheme is more efficient than the original scheme. For example, let us consider the point in the right corner. In Fig.1 (a), the point for RKF4(5) evaluate 1.05e+004 number of function with the error 6.204e-11. However, eeRKF4(5) needs 1.049e+004 nfeval only with the error 5.919e-013. Also, to get the error about 1.0e-10, RKF4(5) needs 1.05+004 nfeval, while eeRKF4(5) only needs 4286 nfeval. This is surprising improvement without any complex derivation.



(a)



(b)

Fig. 1: Comparison of errors versus number of function-evaluations (a) and spent cputime (b)

III. Conclusion

In summary, an error embedding strategy for improving the embedded RK4(5) method is newly introduced. Unlike the traditional way to approximate solutions in an explicit RKF4(5), we suggest a methodology that contains itself the estimated error at each integration step.

Acknowledgment

This work was supported by basic science research program through the National Research Foundation of Korea (NRF)

funded by the Ministry of Education, Science and Technology (grant number 2011-0029013)

References

[1] J. R. Dormand, and P. J. Prince, *High order embedded Runge-Kutta formulae*, J. Comp. and Applied Math., 7(1) (1981) pp. 67–75.
[2] E. Fehlberg, *Classical fifth-, sixth-, seventh-, and eighth-order Runge-Kutta formulas with stepsize control*, NASA; for sale by the Clearinghouse for Federal Scientific and Technical Information, Springfield, VA, 1968.
[3] K. Gustafsson, *Control-theoretic techniques for stepsize selection in implicit Runge-Kutta methods*, ACM Trans. Math. Software, 20(4) (1994), pp. 496–517.
[4] E. Hairer, S. P. Norsett, and G. Wanner, *Solving ordinary differential equations. I Nonstiff*, Springer Series in Computational Mathematics, Springer, 1993.
[5] E. Hairer, and G. Wanner, *Solving ordinary differential equations. II Stiff and Differential-Algebraic Problems*, Springer Series in Computational Mathematics, Springer, 1996.
[6] C. Johnson, *Error estimates and adaptive time-step control for a class of one-step methods for stiff ordinary differential equations*, SIAM J. Numer. Anal., 25(4) (1988), pp. 908–926.
[7] D. Kavetski, P. Binning, and S. W. Sloan, *Adaptive time stepping and error control in a mass conservative numerical solution of the mixed form of Richards equation*, Advances in Water Resources, 24 (2001), pp. 595–605.
[8] L. F. Shampine, *Vectorized solution of ODEs in MATLAB*, Scalable Comput.: Pract. Experience, 10. (2010), pp. 337–345.