KYUNGPOOK Math. J. 56(2016), 311-327
http://dx.doi.org/10.5666/KMJ.2016.56.2.311
pISSN 1225-6951 eISSN 0454-8124

An Error Embedded Runge-Kutta Method for Initial Value Problems

Sunyoung Bu
Department of liberal arts, Hongik University, Sejong 30016, Korea
e-mail : syboo@hongik.ac.kr

WonKyu Jung
Research & Technology Division, Dongwoo Fine-Chem, PyeongTaek 451-822, Korea
e-mail : tailwind86@naver.com

Philsu Kim*
Department of Mathematics, Kyungpook National University, Daegu 702-701, Korea
e-mail : kimps@knu.ac.kr

Abstract. In this paper, we propose an error embedded Runge-Kutta method to improve the traditional embedded Runge-Kutta method. The proposed scheme can be applied into most explicit embedded Runge-Kutta methods. 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, the van der Pol equation and another one having a difficulty for the global

* Corresponding Author.
Received June 28, 2015; accepted October 20, 2015.
2010 Mathematics Subject Classification: 34A45, 65L04, 65L20, 65L70.
Key words and phrases: Error embedded Runge-Kutta method, Embedded Runge-Kutta method, Initial value problem.
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). The first author Bu was supported by the basic science research program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (grant number NRF-2013R1A1A2062783) and 2016 Hongik University Research Fund. Also, the corresponding author Kim was partially supported by the basic science research program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (grant number 2011-0009825).

error control are numerically solved. Finally, a two-body Kepler problem is also used to assess the efficiency of the proposed algorithm.

1. Introduction

There are many research topics [2, 4, 5, 8, 10, 11, 12, 15, 16, 17, 18] in developing numerical methods for solving initial value problems (IVPs) described by

$$\frac{d\phi}{dt} = f(t, \phi(t)), \qquad t \in [t_0, t_f]; \qquad \phi(t_0) = \phi_0, \tag{1.1}$$

where f has continuously bounded partial derivatives up to required order for the developed numerical method. In particular, A long time simulation, which is needed in many physical problems, is one of the most important topics in IVPs. The embedded Runge-Kutta (ERK) method is a popular strategy for the long time simulation. 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 is described by

$$\begin{cases} \phi_{m+1} = F(\phi_m), \\ e_{m+1} = G(\phi_m, \phi_{m+1}), \end{cases} \tag{1.2}$$

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 [6, 7, 9, 12, 13, 14, 21]).

Notice that the solution at time t_{m+1} evolves from the solution at the previous time, which is the most basic property for the solution $\phi(t)$ of (1.1). Due to this evolution property, the error is usually accumulated as the time is increasing even though the evolution property is not shown in the error formula described in (1.2). Hence, for a long time simulation, smaller integration step sizes are usually required to bound the truncation error as the time is going on. However, the usage of smaller integration step sizes can not fully resolve the error control to get a given tolerance. It is difficult to get reliable results at stringent tolerances (for example, see [19, 20]).

The subject of this paper is to improve the existing embedded integration scheme resolving the mentioned difficulties above without a considerable modification of ERK. Our motivation is on the well known fact that the addition of the estimated error e_{m+1} to the solution ϕ_{m+1} after the whole procedure gives more accurate solution. Based on this motivation, we propose a strategy to embed the

estimated error into the algorithm for calculating the solution as a remedy to avoid or reduce the accumulated error of e_m. That is, the proposed scheme is an explicit single step algorithm, so called an error embedded Runge-Kutta (EERK) method, of the form

$$\begin{cases} \phi_{m+1} = F(\phi_m, e_m), \\ e_{m+1} = G(\phi_m, e_m, \phi_{m+1}). \end{cases} \tag{1.3}$$

As in the given embedded RK$p(q)$, we use RKp and RKq to calculate the approximate solution ϕ_{m+1} and the estimated error e_{m+1} in (1.3), respectively. For an appropriate step size controller of (1.3), we exploit the same one used in the given RK$p(q)$. The proposed EERK controls both the error and the time step size simultaneously at each integration step, and it turns out that the proposed method possesses a good performance in the computational cost compared with the original one. For an assessment of the effectiveness of the proposed algorithm, the van der pol oscillator problem and another one having a difficulty for the global error control are numerically solved. Finally, a two-body Kepler problem is also used to assess the efficiency of this algorithm. Throughout these numerical tests, it is shown that the proposed method is quite efficient compared to several existing methods.

This paper is organized as follows. In Sec. 2, we describe the methodology to formulate and control the solution and error formulas based on ERK. Several numerical results are presented in Sec. 3 to give the numerical effectiveness of EERK. Finally, in Sec. 4, a summary for EERK and some discussion for further works are given.

2. Derivation of EERK

In this section, we derive a concrete algorithm of EERK based on a given ERK. Let us assume that a Butcher array

$$\begin{array}{c|c} \mathbf{c} & \mathcal{A} \\ \hline & \mathbf{b} \\ & \hat{\mathbf{b}} \end{array} \tag{2.1}$$

is given for a fixed embedded RK$p(q)$ method, where

$$\begin{aligned} &\mathbf{c} = [c_1, c_2, \cdots, c_n]^T, \quad \mathbf{b} = [b_1, b_2, \cdots, b_n], \\ &\hat{\mathbf{b}} = [\hat{b}_1, \hat{b}_2, \cdots, \hat{b}_n], \quad \mathcal{A} = (\alpha_{i,j})_{n\times n} \end{aligned} \tag{2.2}$$

are given coefficients for RK$p(q)$. Here, we assume that the arrays $\mathbf{b}$ and $\hat{\mathbf{b}}$ are coefficients for RKp and RKq, respectively. For a practice, we can consider RKF4(5), RKF7(8), DOP7(8), etc. 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 RKp and e_m is obtained by the difference between the solutions calculated by RKp and RKq, 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 $h = t_{m+1} - t_m$. For the given embedded RK method, it is true that $p < q$ and hence usually $E_m = \mathcal{O}(h^{p+1}) > \mathcal{O}(h^{q+1}) = E_m - e_m$. 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

$$\begin{cases} \psi'(t) = f(t, \psi(t)), & t \in [t_m, t_{m+1}], \\ \psi(t_m) = \phi_m + e_m \end{cases} \tag{2.3}$$

instead of

$$\begin{cases} \psi'(t) = f(t, \psi(t)), & t \in [t_m, t_{m+1}], \\ \psi(t_m) = \phi_m \end{cases} \tag{2.4}$$

to find the approximation ϕ_{m+1}, where $\psi(t)$ denotes a perturbed solution for $\phi(t)$ on $[t_m, t_{m+1}]$. Hence, as the embedded RK$p(q)$ with the Butcher array given by (2.1), we first solve the problem (2.3) with RKp to calculate the approximate solution ϕ_{m+1} and also solve (2.3) again with RKq to get the estimated error e_{m+1}. Summarizing the procedures, we get the following error embedded Runge-Kutta (EERK) method

$$\begin{cases} \phi_{m+1} = \phi_m + e_m + h \sum_{i=1}^{n} b_i k_i, \\ e_{m+1} = h \sum_{i=1}^{n} \Big(\hat{b}_i - b_i \Big) k_i, \end{cases} \tag{2.5}$$

where

$$k_i = f\Big(t_m + c_i h, \phi_m + e_m + h \sum_{j=1}^{i-1} \alpha_{i,j} k_j \Big), \quad i = 1, \cdots, n. \tag{2.6}$$

The difference of the geometric concepts between the algorithm (2.5) and ERK can be explained as in Fig. 1. Fig. 1 (a) and Fig. 1 (b) are descriptions for EERK and ERK, repsectively. The existing methods usually solve the perturbed IVP (2.4) at each integration step. Hence, after the first step, the estimated error e_m is accumulated as the time step is going on as shown in Fig. 1 (b). Therefore, we reduce this accumulation of the estimated errors by embedding them at each

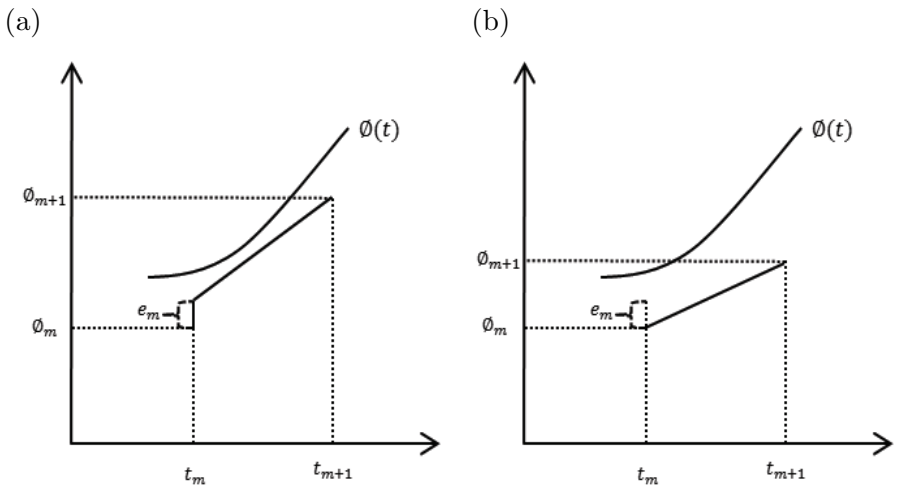


Figure 1: Geometric concepts of (a) EERK and (b) ERK

integration step, so that we can get a smaller global error. In other words, by giving the usage of estimated error, we can improve the capability of the existing methods, while existing methods use the estimated error only for the step-size selection. As one can see, it may be inevitable to give more accurate solution, since the EERK starts with possibly small perturbed initial value.

Remark 2.1. *Three main differences between EERK and ERK are summarized as follows.*

(a) *The traditional embedded RKp(q) uses only the approximate value ϕ_m at time t_m to calculate ϕ_{m+1}, whereas the algorithm* (2.5) *uses the value $\phi_m + e_m$ instead of ϕ_m, which is a remarkable difference compared to the embedded RKp(q).*

(b) *After the whole procedure of the embedded RK method, the estimated error e_{m+1} is frequently added to the solution ϕ_{m+1} for giving a more accurate result. Compared to it, the proposed scheme* (2.5) *uses more accurate intermediate values k_i defined by* (2.6) *by embedding the estimated error e_m. Hence, we would like to say the proposed algorithm as an error embedded Runge-Kutta method. Simply, we denote the scheme by EERKp(q).*

(c) *If we let $\tilde{\phi}_m := \phi_m + e_m$, then the above algorithm* (2.5) *gives the standard RKp(q) for $\tilde{\phi}_m$. That is, $\tilde{\phi}_m$ has the convergence order q and the approximate solution ϕ_m has the convergence order p.*

Using the formula (2.5), we present the following algorithm of EERK.
ALGORITHM EERK(f, ϕ_0, $[t_0, t_{final}]$, tol)

1. Remark: The algorithm EERK is based on the Butcher table given by (2.1) and a given function f. It calculates the approximate solution of (1.1). Further, $[t_0, t_{final}]$ is required integration interval and tol is given tolerance. We assume that the same technique of the step-size selection in ERKs is used.
2. Set N:=length(ϕ_0) and $err := \mathbf{0} \in \mathbb{R}^N$.
3. Initialize h and $told := t_0$.
4. Set $tnew$:=$told + h$. If $tnew > t_{final}$, then exit.
5. Calculate
$$\begin{cases} k_1 := f(told, \phi_0 + err), \\ k_i := f(told + c_i, \phi_0 + err + h\sum_{j=1}^{i-1} a_{i,j}k_j), \quad i = 2, \cdots, n. \end{cases}$$
6. Calculate $\phi_{new} := \phi_0 + err + h\sum_{i=1}^{n} b_i k_i$ and $err_{new} := h\sum_{i=1}^{n}\left(\hat{b}_i - b_i\right)k_i$.
7. If $||err_{new}|| < tol$, then take $\phi_0 := \phi_{new}$, $err := err_{new}$ and save them. After calculating new time step size h_{new} using err_{new} together with the given step size controller in RK$p(q)$ and setting $h = h_{new}$, go to step 4.
If $||err_{new}|| \geq tol$, then resize h using the given step size controller in RK$p(q)$ and go to step 5.

Example 2.2. In the following, we introduce three Butcher arrays corresponding to most popular embedded RK methods [6, 7, 12, 21], RKF4(5), RKF7(8) and DOP7(8) in Table 1, 2 and 3, respectively.

0	0	0	0	0	0	0
$\frac{1}{4}$	$\frac{1}{4}$	0	0	0	0	0
$\frac{3}{8}$	$\frac{3}{32}$	$\frac{9}{32}$	0	0	0	0
$\frac{12}{13}$	$\frac{1932}{2197}$	$-\frac{7200}{2197}$	$\frac{7296}{2197}$	0	0	0
1	$\frac{439}{216}$	-8	$\frac{3680}{513}$	$-\frac{845}{4104}$	0	0
$\frac{1}{2}$	$-\frac{8}{27}$	2	$-\frac{3544}{2565}$	$\frac{1859}{4104}$	$-\frac{11}{40}$	0
	$\frac{25}{216}$	0	$\frac{1408}{2565}$	$\frac{2197}{4104}$	$-\frac{1}{5}$	0
	$\frac{16}{135}$	0	$\frac{6656}{12825}$	$\frac{28561}{56430}$	$-\frac{9}{50}$	$\frac{2}{55}$

Table 1: Runge-Kutta-Fehlberg 4(5) pair

3. Numerical Results

0	0	0	0	0	0	0	0	0	0	0	0	0	0
$\frac{2}{27}$	$\frac{2}{27}$	0	0	0	0	0	0	0	0	0	0	0	0
$\frac{1}{9}$	$\frac{1}{36}$	$\frac{1}{12}$	0	0	0	0	0	0	0	0	0	0	0
$\frac{1}{6}$	$\frac{1}{24}$	0	$\frac{1}{8}$	0	0	0	0	0	0	0	0	0	0
$\frac{5}{12}$	$\frac{5}{12}$	0	$-\frac{25}{16}$	$\frac{25}{16}$	0	0	0	0	0	0	0	0	0
$\frac{1}{2}$	$\frac{1}{20}$	0	0	$\frac{1}{4}$	$\frac{1}{5}$	0	0	0	0	0	0	0	0
$\frac{5}{6}$	$-\frac{25}{108}$	0	0	$\frac{125}{108}$	$-\frac{65}{27}$	$\frac{125}{54}$	0	0	0	0	0	0	0
$\frac{1}{6}$	$\frac{31}{300}$	0	0	0	$\frac{61}{225}$	$-\frac{2}{9}$	$\frac{13}{900}$	0	0	0	0	0	0
$\frac{2}{3}$	2	0	0	$-\frac{53}{6}$	$\frac{704}{45}$	$-\frac{107}{9}$	$\frac{67}{90}$	3	0	0	0	0	0
$\frac{1}{3}$	$-\frac{91}{108}$	0	0	$\frac{23}{108}$	$-\frac{976}{135}$	$\frac{311}{54}$	$-\frac{19}{60}$	$\frac{17}{6}$	$-\frac{1}{12}$	0	0	0	0
1	$\frac{2383}{4100}$	0	0	$-\frac{341}{164}$	$\frac{4496}{1025}$	$-\frac{301}{82}$	$\frac{2133}{4100}$	$\frac{45}{82}$	$\frac{45}{164}$	$\frac{18}{41}$	0	0	0
0	$\frac{3}{205}$	0	0	0	0	$-\frac{6}{41}$	$-\frac{3}{205}$	$-\frac{3}{41}$	$\frac{3}{41}$	$\frac{6}{41}$	0	0	0
1	$-\frac{1777}{4100}$	0	0	$-\frac{341}{164}$	$\frac{4496}{1025}$	$-\frac{289}{82}$	$\frac{2193}{4100}$	$\frac{51}{82}$	$\frac{33}{164}$	$\frac{12}{41}$	0	1	0
	$\frac{41}{840}$	0	0	0	0	$\frac{34}{105}$	$\frac{9}{35}$	$\frac{9}{35}$	$\frac{9}{280}$	$\frac{9}{280}$	$\frac{41}{840}$	0	0
	0	0	0	0	0	$\frac{34}{105}$	$\frac{9}{35}$	$\frac{9}{35}$	$\frac{9}{280}$	$\frac{9}{280}$	0	$\frac{41}{840}$	$\frac{41}{840}$

Table 2: Runge-Kutta-Fehlberg 7(8) pair

0	0	0	0	0	0	0	0	0	0	0	0	0	0
$\frac{1}{18}$	$\frac{1}{18}$	0	0	0	0	0	0	0	0	0	0	0	0
$\frac{1}{12}$	$\frac{1}{48}$	$\frac{1}{16}$	0	0	0	0	0	0	0	0	0	0	0
$\frac{1}{8}$	$\frac{1}{32}$	0	$\frac{3}{32}$	0	0	0	0	0	0	0	0	0	0
$\frac{5}{16}$	$\frac{5}{16}$	0	$-\frac{75}{64}$	$\frac{75}{64}$	0	0	0	0	0	0	0	0	0
$\frac{3}{8}$	$\frac{3}{80}$	0	0	$\frac{3}{16}$	$\frac{3}{20}$	0	0	0	0	0	0	0	0
$\frac{59}{400}$	$\frac{29443841}{614563906}$	0	0	$\frac{77736538}{692538347}$	$-\frac{28693883}{1125000000}$	$\frac{23124283}{1800000000}$	0	0	0	0	0	0	0
$\frac{93}{200}$	$\frac{16016141}{946692911}$	0	0	$\frac{61564180}{158732637}$	$\frac{22789713}{633445777}$	$\frac{545815736}{2771057229}$	$-\frac{180193667}{1043307555}$	0	0	0	0	0	0
$\frac{5490023248}{9719169821}$	$\frac{39632708}{573591083}$	0	0	$-\frac{433636366}{683701615}$	$-\frac{421739975}{2616292301}$	$\frac{100302831}{723423059}$	$\frac{790204164}{839813087}$	$\frac{800635310}{3783071287}$	0	0	0	0	0
$\frac{13}{20}$	$\frac{246121993}{1340847787}$	0	0	$-\frac{37695042795}{15268766246}$	$-\frac{309121744}{1061227803}$	$-\frac{12992083}{490766935}$	$\frac{6005943493}{2108947869}$	$\frac{393006217}{1396673457}$	$\frac{123872331}{1001029789}$	0	0	0	0
$\frac{1201146811}{1299019798}$	$-\frac{1028468189}{846180014}$	0	0	$\frac{8478235783}{508512852}$	$\frac{1311729495}{1432422823}$	$-\frac{10304129995}{1701304382}$	$-\frac{48777925059}{3047939560}$	$\frac{15336726248}{1032824649}$	$-\frac{45442868181}{3398467696}$	$\frac{3065993473}{597172653}$	0	0	0
1	$\frac{185892177}{718116043}$	0	0	$-\frac{3185094517}{667107341}$	$-\frac{477755414}{1098053517}$	$-\frac{703635378}{230739211}$	$\frac{5731566787}{1027545527}$	$\frac{5232866602}{850066563}$	$-\frac{4093664535}{808688257}$	$\frac{3962137247}{1805957418}$	$\frac{65686358}{487910083}$	0	0
1	$\frac{403863854}{491063109}$	0	0	$-\frac{5068492393}{434740067}$	$-\frac{411421997}{543043805}$	$\frac{652783627}{914296604}$	$\frac{11173962825}{925320556}$	$-\frac{13158990841}{6184727034}$	$\frac{3936647629}{1978049680}$	$-\frac{160528059}{685178525}$	$\frac{248638103}{1413531060}$	0	0
	$\frac{13451932}{455176623}$	0	0	0	0	$-\frac{808719846}{976000145}$	$\frac{1757004468}{5645159321}$	$\frac{656045339}{265891186}$	$-\frac{3867574721}{1518517206}$	$\frac{465885868}{322736535}$	$\frac{53011238}{667516719}$	$\frac{2}{45}$	0
	$\frac{14005451}{335480064}$	0	0	0	0	$-\frac{59238493}{1068277825}$	$\frac{181606767}{758867731}$	$\frac{561292985}{797845732}$	$-\frac{1041891430}{1371343529}$	$\frac{760417239}{1151165299}$	$\frac{118820643}{751138087}$	$-\frac{528747749}{2220607170}$	$\frac{1}{4}$

Table 3: Dormand and Prince 7(8) pair

Among many ERKs, we consider only the methods introduced in Example 2.2 and the corresponding EERK methods for them denoted by eeRK4(5), eeRKF7(8) and eeDOP7(8), respectively. To assess the improvement and effectiveness of the proposed scheme, we consider two well known problems, van der Pol oscillator and two-body Kepler problem. Also, we consider another one having a difficulty for the global error control. In the subsequent examples, the notations Rtol and Atol denote the relative and absolute tolerances, respectively. In all numerical results of the examples, we use the sum of the approximate solution ϕ_m and the estimated error e_m as the numerical solution to give more accurate results in each method. As a measure of the effectiveness for each method, we calculate the required number of function evaluations (nfeval) and the computational time (cputime) to solve each problem. For given tolerances Rtol and Atol, 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. In all numerical results, the y-axis represents the absolute errors and the x-axis represents either nfeval (for example, Fig. 2 (a)) or cputime (for example, Fig. 2 (b)). Also, all the marked points from left to right are corresponding to the given tolerances from large to small, respectively. All numerical simulation is executed with MATLAB 2011b(7.13.0.564) and Windows 7 O/S with Intel(R) Core(TM) i7-3770 @ 3.4GHz CPU.

3.1 Example

As the first example, we consider the system of equations described by

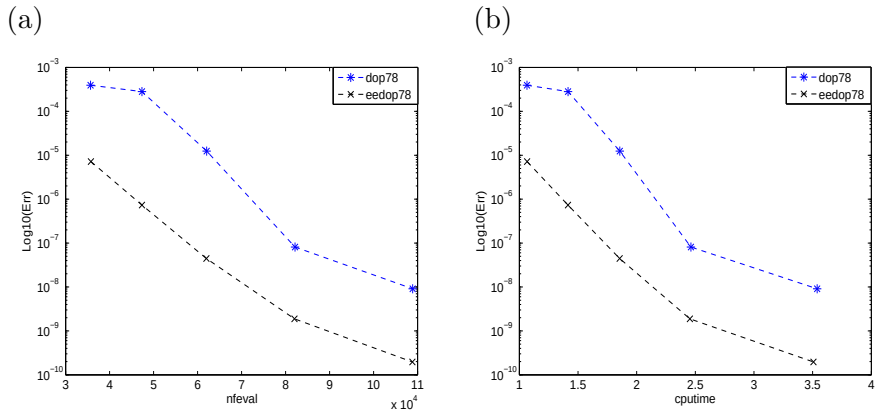
$$\begin{cases} {y_1}' = 2t{y_2}^{1/5}y_4, \\ {y_2}' = 10t\exp(5(y_3-1))y_4, \\ {y_3}' = 2ty_4, \\ {y_4}' = -2t\log(y_1) \end{cases} \tag{3.1}$$

defined on the interval $[0, 20]$ with the initial condition $[y_1(0), y_2(0), y_3(0), y_4(0)]^T = [1, 1, 1, 1]^T$. The analytic solution of the problem is given by

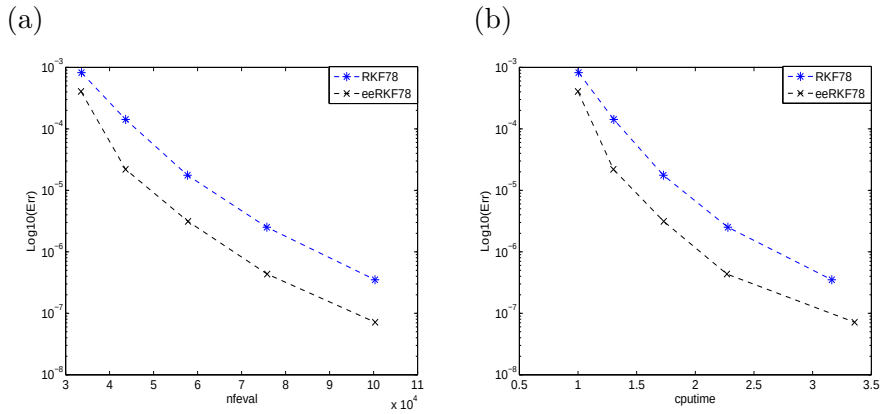
$$y_1(t) = \exp(\sin(t^2)), \quad y_2(t) = \exp(5\sin(t^2)), \quad y_3(t) = \sin(t^2) + 1, \quad y_4(t) = \cos(t^2). \tag{3.2}$$

This problem is known that the global error control task [19] is difficult. In each algorithm, the relative tolerance Rtol is varied from 1.0e-9 to 1.0e-13 and the absolute tolerance Atol from 1.0e-12 to 1.0e-16. In Fig. 2, we list the numerical results for each algorithm and compare them.

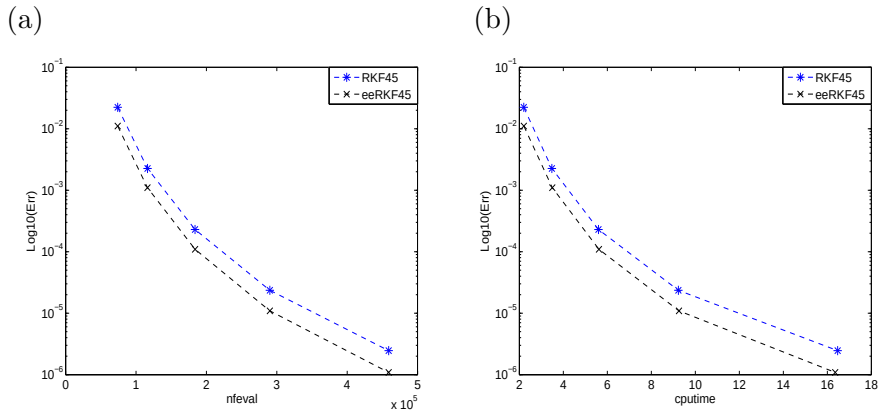
To check the efficiency of the EERK, let us observe the point in the right corner of Fig 2 A. (a) obtained by DOP7(8). DOP7(8) needs about $11 \cdot 10^{11}$ nfeval, but eeDOP7(8) needs only $7 \cdot 10^{11}$ approximately to get the same accuracy. In this sense, one may say that eeDOP7(8) improves DOP7(8) about 33 percentage(%). Likewise, from Fig. 2 B (a) and C (a), one can see that eeRKF7(8) and eeRKF4(5) improve RKF7(8) and RKF4(5) about 25% and 15%, respectively. Also, from the



A. The results of DOP7(8) and eeDOP7(8)



B. The results of RKF7(8) and eeRKF7(8)



C. The results of RKF4(5) and eeRKF4(5)

Figure 2: Comparison of (a) errors versus nfeval (b) errors versus cputime

right figures in Fig. 2, one can see that the speed of computation is understandably getting faster about 37%, 30% and 15% in order as displayed. Further, one can see that eeDOP7(8) get a more accurate numerical solution with 100 times smaller error than DOP7(8) under the same nfeval as shown in the right corner of Fig. 2 A (a). That is, the approximate solutions of DOP7(8) and eeDOP7(8) have the absolute errors about 1.0e-8 and 1.0e-9.8, respectively.

Likewise, the errors of eeRKF7(8) and eeRKF4(5) are about 10 times and 3 times smaller than those of RKF7(8) and RKF4(5), respectively. To conclude, eeDOP7(8) is most efficient among three improved methods for this example.

3.2 Van der Pol oscillator

The Van der Pol (VDPOL) problem originates from electronics and it describes the behavior of nonlinear vaccum tube circuits [1]. The solution of the problem satisfies a second order differential equation and it can be rewritten to a first order form given by

$$\begin{cases} y_1' = y_2, \\ y_2' = \mu(1 - y_2^2)y_2 - y_1, \quad \mu > 0, \end{cases} \tag{3.3}$$

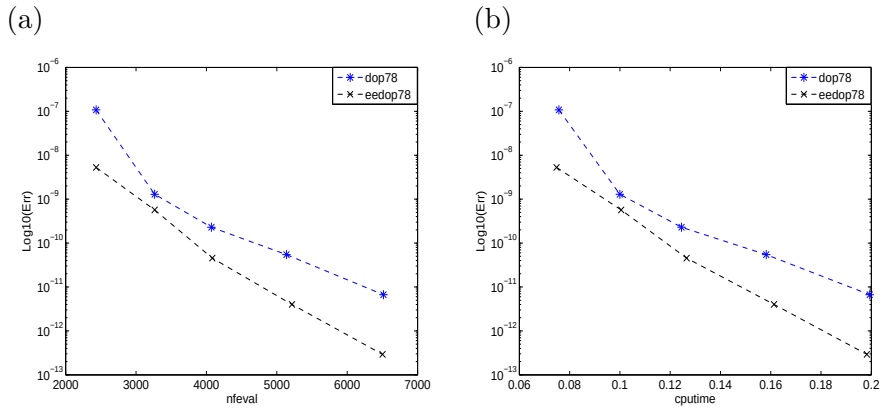
where the scalar parameter μ indicates the nonlinearity and the strength of the damping. According to the magnitude of μ, the stiffness of the described problem is determined. In this example, we consider a nonstiff case by taking $\mu = 5$. We solve the problem on the integration interval $[0, 20]$ with initial value $[y_1(0), y_2(0)]^T = [2, 0]^T$. It is well known that there are no analytic solution and hence we take a reference solution at final time calculated by RADAU5 with the tolerances $Atol = Rtol = eps$, where eps is the double precision of floating numbers in MATLAB. In each algorithm, the relative tolerance Rtol is used by varing from 1.0e-7 to 1.0e-11 and the absolute tolerance Atol from 1.0e-10 to 1.0e-14. In Fig. 3, we list the numerical results for each algorithm and compare them.

As the same ways of the previous example, we discuss the efficiency and the improvement of the proposed algorithms for the problem and summarize the results in Table 4 and 5.

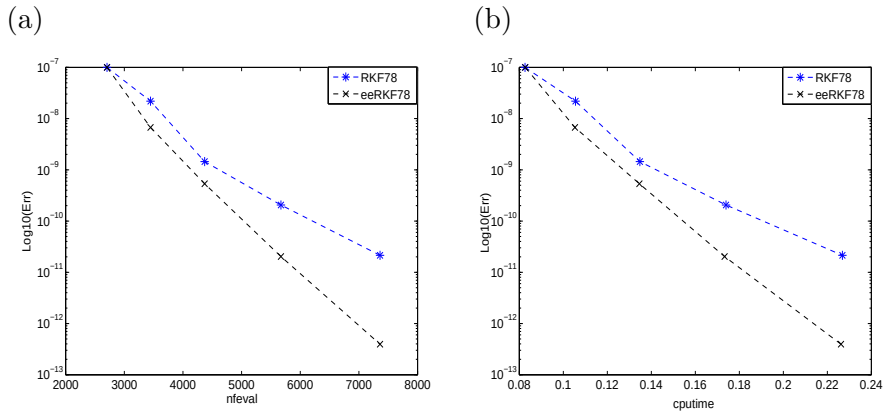
	nfeval	cputime
$eeDOP7(8)$	23%	25%
$eeRKF7(8)$	24%	26%
$eeRKF4(5)$	50%	52%

Table 4: Improvement ratio for VDPOL

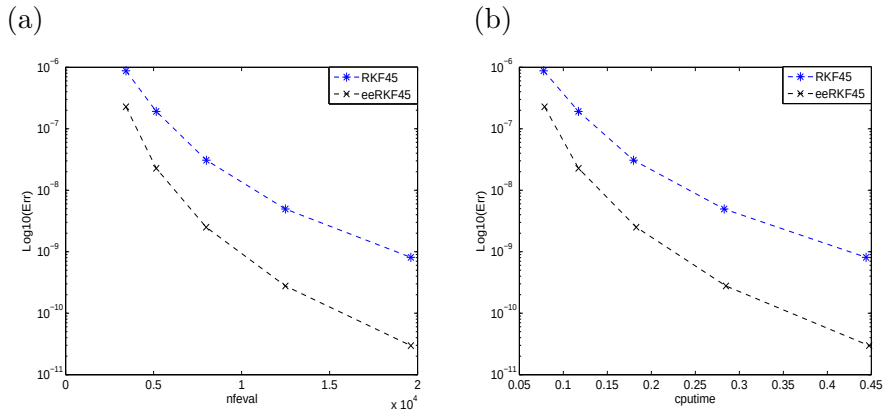
As seen in Table 4, eeRKF4(5) is the biggest among three improvements in the sense of both nfeval and cputime. To examine the accuracy of the EERK, we investigate the error at the final time with nfeval in the Table 5. One can see that the EERKs give more accurate result than existing algorithms. In particular, eeDOP7(8) requires the smallest nfeval with the smallest error



A. The results of DOP7(8) and eeDOP7(8)



B. The results of RKF7(8) and eeRKF7(8)



C. The results of RKF4(5) and eeRKF4(5)

Figure 3: Comparison of (a) errors versus nfeval (b) errors versus cputime

	Error at the final-time	nfeval
$eeDOP7(8)$	$2.927e-13$	6502
$DOP7(8)$	$6.685e-12$	6515
$eeRKF7(8)$	$3.942e-13$	7360
$RKF7(8)$	$2.143e-11$	7360
$eeRKF4(5)$	$2.967e-11$	$1.962e+4$
$RKF4(5)$	$8.806e-10$	$1.962e+4$

Table 5: Error and its demanded nfeval for the point corresponding to the smallest tolerance in Fig. 3

3.3 Kepler problem

In astronomy's problems such as Kepler problem, a long-term simulation is an indispensable factor. Hence, we solve a two-body Kepler's problem subject to Newton's law of gravitation revolving around their center of mass, placed at the origin, in elliptic orbits in the (q_1, q_2)-plane [3]. Assuming unitary masses and gravitational constant, the dynamics of the two-body is described by the Hamiltonian function H given by

$$H(p_1, p_2, q_1, q_2) = \frac{1}{2}(p_1^2 + p_2^2) - \frac{1}{\sqrt{q_1^2 + q_2^2}} \tag{3.4}$$

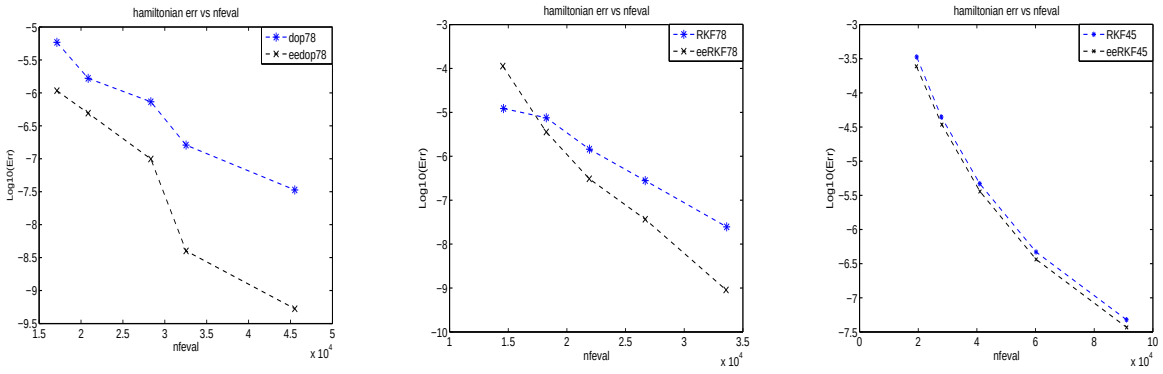
together with the angular momentum L, which is another invariant of the system, described by

$$L(p_1, p_2, q_1, q_2) = q_1 p_2 - q_2 p_1, \tag{3.5}$$

whose components p_i, q_i $(i = 1, 2)$ satisfy the following IVP

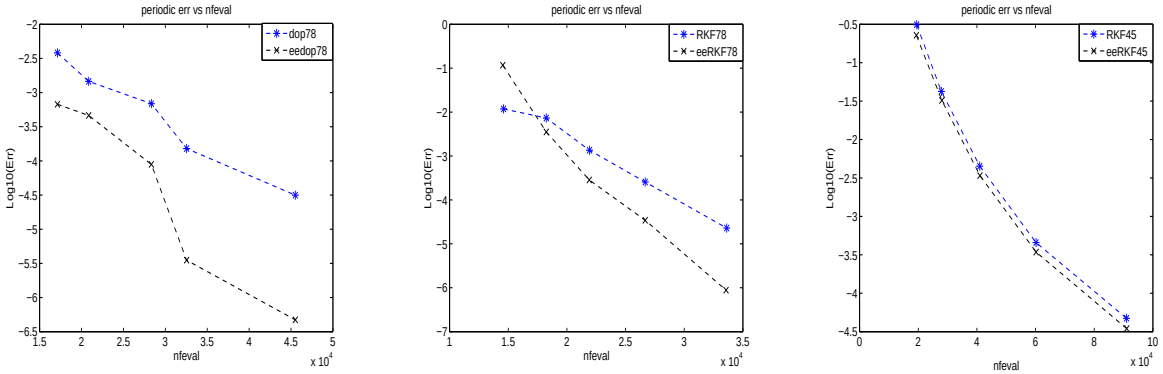
$$\begin{cases} {p_1}'(t) = -q_1({q_1}^2 + {q_2}^2)^{(-3/2)}, \\ {p_2}'(t) = -q_2({q_1}^2 + {q_2}^2)^{(-3/2)}, \\ {q_1}'(t) = p_1, \\ {q_2}'(t) = p_2. \end{cases} \tag{3.6}$$

We solve the system (3.6) with the initial conditions $p_1(0) = 0$, $p_2(0) = 2$, $q_1(0) = 0.4$, $q_2(0) = 0$ on the interval $[0, 100\pi]$ by varying the Atol and Rtol from 1.0e-6 to 1.0e-10. It is well known that the true solution is periodic with periodicity 2π [3]. To examine how the considered algorithms satisfy the conservation property, we calculate the errors for the total energy H defined by (3.4) and plot the error versus nfeval in the Fig. 4 to compare each algorithm. Also, to examine the error



(a) (b) (c)

Figure 4: Comparison of Hamiltonian errors versus number of function-evaluations for given tolerances for (a) DOP7(8) (b) RKF7(8) (c) RKF4(5)



(a) (b) (c)

Figure 5: Comparison of errors of periodic solution versus number of function-evaluations for given tolerances for (a)dop78 (b) RKF78 (c) RKF45

behavior at the periodic point, the errors between the starting point $(q_1(0), q_2(0)) = (0.4, 0)$ and the numerical solutions at final time $t = 100\pi$ obtained by EERKs and ERKs are presented in Fig. 5.

With the same way to check the efficiency as the previous examples, we summarize the results in Table 6 and 7. As shown in the Table 6, eeDOP7(8) achieves the largest improvement about 33% in both Hamiltonian and periodicity and eeRKF4(5) has a relatively small improvement.

	nfeval for Hamiltonian	nfeval for Periodic solution
$eeDOP7(8)$	33%	33%
$eeRKF7(8)$	20%	23%
$eeRKF4(5)$	5%	5%

Table 6: Improvement ratio for Kepler problem

	Hamiltonian Error	Periodic solution Error	nfeval(Hamiltonian)
$eeDOP7(8)$	$1.0e-9.276$	$1.0e-6.327$	$4.552e+4$
$DOP7(8)$	$1.0e-7.473$	$1.0e-4.502$	$4.552e+4$
$eeRKF7(8)$	$1.0e-9.044$	$1.0e-6.055$	$3.357e+4$
$RKF7(8)$	$1.0e-7.607$	$1.0e-4.641$	$3.362e+4$
$eeRKF4(5)$	$1.0e-7.434$	$1.0e-4.46$	$9.108e+4$
$RKF4(5)$	$1.0e-7.322$	$1.0e-4.325$	$9.102e+4$

Table 7: Error and its demanded nfeval and cputime for the point corresponding to the smallest tolerance in Fig. 4 and Fig. 5

Also, from Table 7, it can be seen that eeRKF7(8) has a similar conservation properties in Hamiltonian and the periodicity of the solution compared to eeDOP7(8) with the smaller nfeval. One may summarize that the proposed EERK scheme improves the existing ERK.

4. Conclusion and Further Discussion

In summary, an error embedding strategy for improving the embedded RK (ERK) method is newly introduced. Unlike the traditional way to approximate solutions in ERK, we suggest a methodology that contains itself the estimated error at each integration step. Throughout several numerical results, it is shown that the proposed scheme can be applied most existing ERKs. Especially, eeDOP7(8) gives a striking improvement in the discussed problems. In order to fully explore the efficiency of EERK, several extended issues are currently being pursued. One of

them is to investigate strategies for selecting time-integration step to reduce time-cost and to get bounded error behavior within given tolerances. The proposed method is developed only for explicit embedded Runge-Kutta methods. Hence, the other challenge is to extend the idea of proposed method into implicit method to solve stiff-problem more efficiently. Additionally, the generalization of the proposed idea will be applied to many physical problems expressed by partial differential equations (PDEs). Results along these directions will be reported in the future.

References

[1] `https://www.dm.uniba.it/testset/testsetivpsolvers`

[2] K. E. Atkinson, *An introduction to numerical analysis*, John Wiley & Sons, Inc., 1989.

[3] L. Brugnano, F. Iavernaro and D. Trigiante, *Energy- and quadratic invariants-preserving integrators based upon Gauss collocation formulae*, SIAM J. Numer. Anal., **50**(2012), 2897–2916.

[4] S. Bu, J. Huang and M. L. Minion, *Semi-implicit Krylov deferred correction methods for differential algebraic equations*, Math. Comput., **81(280)**(2012), 2127–2157.

[5] M. P. Calvo and E. Hairer, *Accurate long-term integration of dynamical systems*, Appl. Numer. Math., **18**(1995), 95-105.

[6] J. R. Dormand and P. J. Prince, *High order embedded Runge-Kutta formulae*, J. Comput. Appl. Math., **7(1)**(1981), 67–75.

[7] 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.

[8] C. W. Gear, *Numerical initial value problems in ordinary differential equations*, Prentice-Hall, 1971.

[9] K. Gustafsson, *Control-theoretic techniques for stepsize selection in implicit Runge-Kutta methods*, ACM Trans. Math. Softw., **20(4)**(1994), 496–517.

[10] E. Hairer, *Long-time integration of non-stiff and oscillatory Hamiltonian systems*, AIP Conf. Proc., **1168(1)**(2009), 3–6.

[11] E. Hairer, S. P. Norsett and G. Wanner, *Solving ordinary differential equations. I nonstiff*, Springer Series in Computational Mathematics, Springer, 1993.

[12] E. Hairer and G. Wanner, *Solving ordinary differential equations. II stiff and differential-algebraic problems*, Springer Series in Computational Mathematics, Springer, 1996.

[13] 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), 908–926.

[14] 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*, Adv. Water Resour., **24**(2001), 595–605.

[15] P. Kim, X. Piao and S. D. Kim, *An error corrected Euler method for solving stiff problems based on Chebyshev collocation*, SIAM J. Numer. Anal. **49**(2011), 2211–2230.

[16] S. D. Kim, X. Piao, D. H. Kim and P. Kim, *Convergence on error correction methods for solving initial value problems*, J. Comput. Appl. Math., **236(17)**(2012), 4448–4461.

[17] P. Kim, E. Lee and S. D. Kim, *Simple ECEM algorithms using function values only*, Kyungpook Math. J., **53**(2013), 573–591.

[18] P. Kim and S. Bu, *Error Control Strategy in Error Correction Methods*, Kyungpook Math. J., **55**(2015), 301–311.

[19] G. Y. Kulikov and R. Weiner, *Global error estimation and control in linearly-implicit parallel two-step peer W-methods*, J. Comput. Appl. Math., **236**(2011), 1226–1239.

[20] L. F. Shampine, *Error estimation and control for ODEs*, J. Sci. Comput., **25(1)**(2005), 3–16.

[21] L. F. Shampine, *Vectorized solution of ODEs in MATLAB, Scalable Comput.*, Pract. Exp., **10**(2010), 337–345.