KYUNGPOOK Math. J. 52(2012), 167-177
http://dx.doi.org/10.5666/KMJ.2012.52.2.167

Exponentially Fitted Error Correction Methods for Solving Initial Value Problems

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

Abstract. In this article, we propose exponentially fitted error correction methods(EECM) which originate from the error correction methods recently developed by the authors (see [10, 11] for examples) for solving nonlinear stiff initial value problems. We reduce the computational cost of the error correction method by making a local approximation of exponential type. This exponential local approximation yields an EECM that is exponentially fitted, A-stable and L-stable, independent of the approximation scheme for the error correction. In particular, the classical explicit Runge-Kutta method for the error correction not only saves the computational cost that the error correction method requires but also gives the same convergence order as the error correction method does. Numerical evidence is provided to support the theoretical results.

1. Introduction

In this paper we are concerned with numerical methods for solving stiff initial value problems (IVPs)

$$\frac{d\phi}{dt} = f(t, \phi(t)), \quad t \in (t_0, T]; \quad \phi(t_0) = \phi_0. \tag{1.1}$$

The main objectives of the numerical methods are not only to reduce the computational cost but also to enhance accuracy and stability. These objectives are related to the representative aspects of IVPs such as stiffness, nonlinearity and their correlation.

It is well known that the A-stability of numerical methods is a key issue for stiff systems because one can choose the step size based only on accuracy without

* Corresponding Author.
Received March 5, 2012; accepted June 22, 2012.
2010 Mathematics Subject Classification: 34A45, 65L20, 65L70.
Key words and phrases: Exponentially fitted, Error correction, Stiff initial-value problem, RK methods.
† 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).

worrying about its stability constraint. Using higher-order derivatives, the idea of exponentially fitted methods (EFM), which was originally proposed by Liniger and Willoughby [9], has received considerable attention [1, 5, 6, 8, 13]. The basic idea of exponential fitting is to derive integration formulae containing free parameters and then to choose these parameters so that the given exponential function satisfies the integration formula exactly. Explicit numerical methods are better for overcoming nonlinearity and simple to implement but poor for stability (see [4, 7] for example). There are only a few explicit numerical methods having a good stability. For example, Wu [14] developed a sixth-order A-stable explicit one-step method based on a Taylor's series method, which was proved to be an exponentially fitted L-stable method but requires higher-order derivatives for the function f. Recently, Ramos [12] suggested a non-standard explicit numerical integration method which is A-stable with second-order accuracy for solving stiff IVPs.

This article constructs a new type of EFM which only require just the first derivatives of f, and do not require such iteration steps for nonlinear discrete systems as most implicit methods do. The scheme we develop is based on the same procedures of the error correction methods (ECM) recently developed by the authors (see [10, 11] for examples) except for choosing the local approximation and solving the asymptotic equation for the difference between the true solution and the local approximation. We use a local approximation of the form

$$x(t) = a\exp(b(t-t_m)), \quad t \in [t_m, t_{m+1}].$$

This exponential local approximation yields an exponentially fitted error correction method (EECM) that is exponentially fitted A-stable and L-stable. That is, the proposed EECM has the exact solution for the Dahlquist's problem. In particular, we find that the error correction is identically zero for the Dahlquist's problem and hence the stability of the EECM doesn't depend on an approximation scheme for the error correction. The classical explicit Runge-Kutta (RK) method for approximating the error correction not only saves the computational cost that ECM requires but also has the same order of convergence as the ECM.

This paper is organized as follows. Section 2 derives an accurate L-stable EFM with the convergence order 4. In Section 3, we simulate several test problems to provide numerical evidences. Finally, we provide some comments and conclusions in Section 4.

2. Derivation of an approximation method

The goal of this section is to derive a novel algorithm to get next approximation y_{m+1} at time $t_{m+1} = t_m + h$ for a given approximation y_m at time t_m to the exact solution $\phi(t)$ for (1.1). The uniform time step size $h := t_{m+1} - t_m$, $m = 0, 1, \cdots$, will be used from now on. To the aim, we consider a local approximation $x(t)$ defined by

$$x(t) := y_m \exp\Big(\frac{f_m}{y_m}(t-t_m)\Big), \quad t \in [t_m, t_{m+1}], \tag{2.1}$$

where we assume that $y_m \neq 0$. Then, it can be shown that

$$G(t) := f(t, x(t)) - x'(t) = O(h), \quad t \in [t_m, t_{m+1}]. \tag{2.2}$$

Using the local approximation $x(t)$ defined by (2.1), we consider a perturbation $\psi(t)$ of the solution $\phi(t)$ on each time step $[t_m, t_{m+1}]$ defined by

$$\psi(t) := \phi(t) - x(t), \quad t \in [t_m, t_{m+1}], \tag{2.3}$$

which satisfies the asymptotic linear ODE (see [10] for example)

$$\psi'(t) = \varphi(t)\psi(t) + G(t) + O\Big(\psi(t)^2\Big), \quad \varphi(t) = f_\phi(t, x(t)), \quad t \in (t_m, t_{m+1}), \tag{2.4}$$

where the generic constant in $O(\cdot)$ depends on the bounds of the second order derivatives of f. By some manipulations with the integrating factor method for the equation (2.4), one may show that the solution $\phi(t_{m+1})$ can be expressed by

$$\phi(t_{m+1}) = x(t_{m+1}) + \mathcal{C}_m + \rho, \tag{2.5}$$

where

$$\mathcal{C}_m = \frac{h}{2}\int_{-1}^{1} \mathcal{E}(s)G(t(s))ds, \quad \mathcal{E}(s) = \exp\Big(\frac{h}{2}\int_{s}^{1} \varphi(t(\xi))d\xi\Big), \quad t(s) := t_m + \frac{h}{2}(1+s), \tag{2.6}$$

and

$$|\rho| \leq (1 + C_1 h)|E_m| + C_2 h^5, \quad E_m := \phi(t_m) - y_m \tag{2.7}$$

for some constants C_i independent of h and m. For the detailed proof, one refer to see the references [10, 11].

Theorem 2.1. *Let $\mathcal{A}_m$ be an arbitrary approximation for the correction term $\mathcal{C}_m$ with the error bound*

$$|\mathcal{C}_m - \mathcal{A}_m| \leq Ch^q, \quad m \geq 1 \tag{2.8}$$

for some constant C independent of h and m, where q is a positive number. Now, if one design an approximation scheme defined by

$$y_{m+1} = x(t_{m+1}) + \mathcal{A}_m, \quad m \geq 0; \quad y_0 = \phi_0, \tag{2.9}$$

then, the actual error $E_m := \phi(t_m) - y_m$ has the convergence result

$$|E_m| \leq D\Big(\exp(CT) - 1\Big)h^{\min(4, q-1)}, \quad m \geq 0 \tag{2.10}$$

for some constants C and D independent of h and m.

Proof. From (2.7) and (2.8), subtracting (2.9) from (2.5) yields

$$|E_{m+1}| \le (1+Ch)|E_m| + Dh^{\min(2p+3,q)}, \quad m \ge 0; \quad |E_0| = 0 \tag{2.11}$$

for some constants C and D independent of h and m. Thus, by induction, the difference equation (2.11) can be solved and gives the desired inequality (2.10). □

Now, for the approximation of the error correction term $\mathcal{C}_m$ defined by (2.6), we observe the following lemma.

Theorem 2.2. *Assume that $\vartheta(t)$ is the solution of the initial value problem*

$$\vartheta'(t) = \varphi(t)\vartheta(t) + G(t), \quad t \in (t_m, t_{m+1}]\ ; \quad \vartheta(t_m) = 0. \tag{2.12}$$

Then, one have

$$\mathcal{C}_m = \vartheta(t_{m+1}).$$

Proof. Using the change of variables $t = t(s)$ defined by (2.6), for the solution $\vartheta(t)$ of IPV (2.12) we define

$$\bar{\vartheta}(s) = \vartheta(t(s)), \quad s \in [-1, 1].$$

Then, one may check that $\bar{\vartheta}(s)$ is the solution of IVP

$$\bar{\vartheta}'(s) = \frac{h}{2}\varphi(t(s))\bar{\vartheta}(s) + \frac{h}{2}G(t(s)), \quad t \in (-1, 1]\ ; \quad \bar{\vartheta}(-1) = 0. \tag{2.13}$$

Multiplying an integrating factor $\exp\Big(-\frac{h}{2}\int_{-1}^{s}\varphi(t(\xi))d\xi\Big)$ to both sides of (2.13) leads to

$$\frac{d}{ds}\Big(\exp\Big(-\frac{h}{2}\int_{-1}^{s}\varphi(t(\xi))d\xi\Big)\bar{\vartheta}(s)\Big) = \frac{h}{2}\exp\Big(-\frac{h}{2}\int_{-1}^{s}\varphi(t(\xi))d\xi\Big)G(t(s).$$

Now, integrating it from -1 to 1 and using the initial condition of (2.13) yield

$$\exp\Big(-\frac{h}{2}\int_{-1}^{1}\varphi(t(\xi))d\xi\Big)\bar{\vartheta}(1) = \frac{h}{2}\int_{-1}^{1}\exp\Big(-\frac{h}{2}\int_{-1}^{s}\varphi(t(\xi))d\xi\Big)G(t(s))ds$$

or equivalently

$$\bar{\vartheta}(1) = \frac{h}{2}\int_{-1}^{1}\exp\Big(\frac{h}{2}\int_{s}^{1}\varphi(t(\xi))d\xi\Big)G(t(s))ds,$$

which is exactly same with the error correction $\mathcal{C}_m$ defined by (2.12). □

In the following Theorem 2.3, we will show that the stability constraint of the numerical method we want to develop depends only on the exponential local

approximation $x(t)$ defined by (2.1). That is, the stability is independent of the approximation scheme of the error correction $\mathcal{C}_m$ defined by (2.6). Notice that the error correction $\mathcal{C}_m$ defined by (2.6) contains two integrals which may require to evaluate an expensive exponential function for a matrix for the case of the system of ODE. Thus, instead of a direct use of the formula (2.6), we are going to use the solution of IVP (2.12) to optimize the computational cost for the calculation of $\mathcal{C}_m$. Based on Theorem 2.2, we will take the fourth-order explicit RK method, which has a fifth-order local truncation error, for solving IVP (2.12). By the facts $\vartheta(t_m) = 0$ and $G(t_m) = 0$, applying RK method to (2.12) yields

$$\mathcal{C}_m = \frac{h}{6}\Big[2V_1 + 2V_2 + V_3\Big] + O(h^5), \tag{2.14}$$

where

$$\begin{aligned} V_1 &= G(t_{m+\frac{1}{2}}), \\ V_2 &= \frac{h}{2}\varphi(t_{m+\frac{h}{2}})V_1 + G(t_{m+\frac{1}{2}}), \\ V_3 &= h\varphi(t_{m+1})V_2 + G(t_{m+1}), \end{aligned} \tag{2.15}$$

where $t_{m+\alpha} = t_m + \alpha h$. Substituting (2.14) into (2.15) leads to

$$\phi(t_{m+1}) = x(t_{m+1}) + \frac{h}{6}\Big[2V_1 + 2V_2 + V_3\Big] + \rho + O(h^5). \tag{2.16}$$

From the representation (2.16), one may define an explicit one step method as follows.

$$\begin{aligned} y_{m+1} &= x(t_{m+1}) + \frac{h}{6}\Big[2V_1 + 2V_2 + V_3\Big], \quad m \geq 0, \\ y_0 &= \phi_0. \end{aligned} \tag{2.17}$$

From Theorem 2.1, one may easily see that the algorithm (2.17) has the convergence order 4.

Now we close the section by analyzing the stability of the algorithm (2.17). To the aim, we apply the algorithm (2.17) with Dahlquist's probelm

$$\phi'(t) = \lambda\phi(t), \quad t > 0; \quad \phi(0) = \phi_0, \tag{2.18}$$

where λ is a complex number such that $Re(\lambda) < 0$. For the problem (2.18), the local approximation platform $x(t)$ defined in (2.1) becomes

$$x(t) = y_m \exp(\lambda(t - t_m)), \quad t \in [t_m, t_{m+1}] \tag{2.19}$$

and its residual error $G(t)$ defined in (2.2) becomes

$$G(t) = 0, \quad t \in [t_m, t_{m+1}]. \tag{2.20}$$

The identity (2.20) shows that the IVP (2.12) has a trivial solution. Hence, from the formula (2.19), the algorithm (2.17) can be simplified by

$$y_{m+1} = y_m \exp(\lambda h), \quad m \geq 0; \quad y_0 = \phi_0. \tag{2.21}$$

Hence, we have the following theorem.

Theorem 2.3. *The algorithm* (2.17) *gives an exact solution to the Dahlquist's problem* (2.18) *and is A-stable and L-stable.*

Proof. By induction, the difference relation (2.21) can be solved by

$$y_m = y_0 \exp(\lambda m h) = y_0 \exp(\lambda t_m), \quad t_m = mh \in [0, T]. \tag{2.22}$$

Thus, for any $t = t_m \in [0, T]$, we have

$$y_m = \phi(t).$$

This means that the algorithm (2.17) gives an exact solution to the Dahlquist's problem (2.18).

For the proof of the stability, we let $\lambda = x + iy$ with $x < 0$. Then, the first equation of (2.22) shows

$$y_m = y_0 \exp(mhx) \exp(imhy), \quad m \geq 0. \tag{2.23}$$

Since $x < 0$, the equation (2.23) shows that the sequence $\{y_m\}$ converges to 0 whenever $m \to \infty$. Consequently, the algorithm (2.17) is A-stable. It is easy to see from (2.21) that the algorithm (2.17) is also L-stable. In fact, we have $Q(h\lambda) = \exp(h\lambda)$ and $|Q(h\lambda)| \to 0$ as $Re(h\lambda) \to -\infty$. □

Remark 2.4. Some comparisons between the algorithm (2.17) and the ECM in [10] can be summarized as follows.

(a) Theorem 2.2 says that the algorithm (2.17) is an explicit one-step method and exponentially fitted L-stable with the convergence of accuracy 4, which is a considerable improvement compared with the ECM (see [10] for example) in the sense of the stability.

(b) From the fact (2.20), the correction term $\mathcal{C}_m$ for the Dahlquist's problem is identically zero. It means that the stability of the algorithm (2.17) depends only on the exponential local approximation $x(t)$ defined by (2.1). Hence, one can choose an approximation scheme for the error correction term based only on the accuracy without worrying about its stability constraint, but that of the algorithm ECM depends on the approximation scheme for the correction term. Thus the classical RK method for the correction term completely reduces the computational cost that the ECM requires.

3. Numerical experiments

In this section, three test problems are taken to give numerical evidences of the theoretical results for the proposed methods. For the comparison of the numerical results, the maximum error $Err(h)$ and convergence rates are employed, which are defined by

$$Err(h) = \max_{1\le j\le n} \|\phi(t_j) - y_j\|_\infty, \quad rate = \frac{\log(Err(h_1)/Err(h_2))}{\log(h_1/h_2)}, \tag{3.1}$$

respectively, where $\|\cdot\|_\infty$ denotes the maximum norm and h_j, $j = 1, 2$ are given two time step sizes. All numerical computations in the following are performed with Matlab R2007a(32bit) in the personal computer having the processor Intel(R) Core2Quad(TM) Q8200 2.33GHz. All existing methods we take for the numerical comparison use a time step control process to optimize the computational cost. Also, the order of convergence for the existing methods are different from each other. Thus, a direct comparison between the present algorithm EECM and the existing methods seems to be unfair and so we will compare the numerical results of the existing methods and EECM just in the sense of the computation time cost "cpu" and the maximum error.

Example 3.1. Consider the following nonlinear problem (see [3])

$$\frac{d\phi}{dt} = \frac{\lambda\phi(t)(1-\phi(t))}{2\phi(t)-1}, \quad t > 0; \quad \phi(0) = \frac{5}{6}, \tag{3.2}$$

whose solution is $\phi(t) = \frac{1}{2} + \sqrt{\frac{1}{4} - \frac{5}{36}e^{-\lambda t}}$. To investigate the stiffness, the problem (3.2) is solved on the interval $[0, 2]$ with the parameter $\lambda = 30$ which gives a mild stiffness around initial time. For the numerical convergence, we solve the problem with different step sizes $h = 2^{-n}$, $n = 4, \cdots, 10$. One can see that from the numerical results reported in Table 1, the numerical convergence order converges to 4 as the step size is decreasing. To compare a numerical efficiency, the problem (3.2) is solved by the EECM and two matlab built-in functions ODE15s and ODE23s on the interval $[0, 0.4]$ containing the stiff region. The uniform step size $h = 2^{-10}$ is chosen for the algorithm EECM. Also, the error tolerances RelTol $= 10^{-9} =$ AbsTol are taken for the matlab built-in functions ODE15s and ODE23s, which are known as solvers for stiff problems. The numerical results of the maximum error and the CPU time for each method are reported in Table 2 and one can see that the algorithm EECM is superior to two existing methods in the sense of the maximum error and CPU time.

Example 3.2. Consider the following linear stiff problem taken from [2]

$$\frac{d\phi}{dt} = -100\phi(t) + 99\exp(2t) + 100, \quad t > 0; \quad \phi(0) = 1, \tag{3.3}$$

Table 1. Results for Example 3.1 on $[0, 2]$ with step size $h = 2^{-n}$, $n = 4, \cdots, 10$

n	$Err(h)$	rate
4	4.05×10^{-2}	$--$
5	3.73×10^{-3}	3.44
6	2.57×10^{-4}	3.86
7	1.45×10^{-5}	4.15
8	8.34×10^{-7}	4.12
9	4.99×10^{-8}	4.06
10	3.03×10^{-9}	4.04

Table 2. Numerical comparison for Example 3.1, using EECM, ODE15s and ODE23s

EECM		ODE15s		ODE23s	
$Err(h)$	CPU	$Err(h)$	CPU	$Err(h)$	CPU
3.03×10^{-9}	0.02	3.80×10^{-9}	0.05	6.57×10^{-8}	0.22

whose solution is $\phi(t) = \frac{33}{34}(\exp(2t) - \exp(-100t)) + 1$.

Note that the stiffness of the solution occurs around the zero and the solution is dominated by $\frac{33}{34}\exp(2t)$ far away from zero. For the numerical convergence test, the problem (3.3) is solved on the interval $[0, 5]$ with different step sizes $h = 2^{-n}$, $n = 6, \cdots, 11$. One can see that from the numerical results reported in Table 3, the numerical convergence order approaches to 4 as the step size is decreasing. For a comparison of numerical efficiency, the uniform step size $h = 2^{-11}$ for the EECM and the error tolerances RelTol $= 10^{-12}$, AbsTol $= 10^{-14}$ are used for the matlab built-in functions ODE15s and ODE23s. The numerical results of the maximum error and the CPU time for each method are reported in Table 4 and one can see that the EECM is superior to two existing methods in the sense of cpu time.

Also, from the figure for the absolute error given in Fig. 1, one can see that the EECM has the maximum error around the stiff region, but two existing methods ode15s and ode23s have their maximum errors at the end of the time. That is, one may assert that the algorithm EECM works well to a solution with the exponential growth.

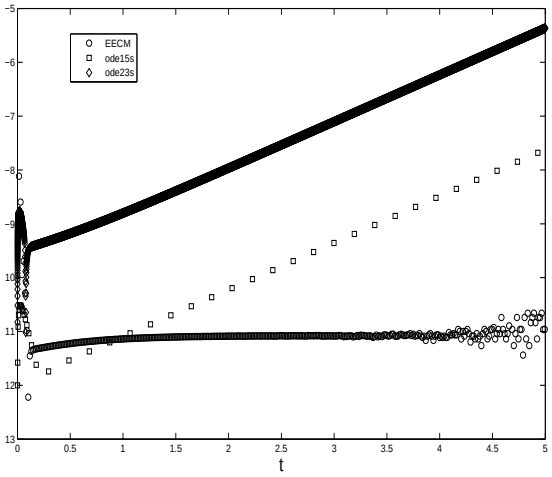

Fig. 1. Numerical results obtained by EECM, ODE15s and ODE23s for the problem (3.3) on the time $[0, 5]$; $h = 2^{-11}$ for EECM.

Example 3.3. As a system example, consider the following nonlinear problem

$$\frac{d\Phi}{dt} = \mathfrak{F}(\Phi), \quad t > 0; \quad \Phi(0) = \Phi_0, \tag{3.4}$$

where $\Phi = [\phi_1, \phi_2]^T \in \Phi \in \mathbb{R}^2$, Φ_0 and $\mathfrak{F}$ is defined by

$$\Phi_0 = [1, 1]^T, \quad \mathfrak{F}(\Phi) = \begin{bmatrix} -(\lambda + 2)\phi_1(t) + \lambda\phi_2^2(t) \\ \phi_1(t) - \phi_2(t)(1 + \phi_2(t)) \end{bmatrix},$$

whose solutions are given by

$$\phi_1(t) = \exp(-2t), \quad \phi_2(t) = \exp(-t), \quad t > 0. \tag{3.5}$$

Note that the larger the parameter λ is, the stronger the stiffness becomes around the initial time. In this numerical test, we take $\lambda = 80$ giving a mild stiffness at the initial time. The problem (3.4) is solved on the time interval $[0, 2]$ by using three methods EECM, ode15s, and ode23s. We use the uniform time step size $h = 2^{-5}$ for the EECM and the error tolerances RelTol $= 10^{-13}$ and AbsTol $= 10^{-15}$ for the matlab built-in functions ODE15s and ODE23s. The numerical results of the maximum error and the CPU time for each method are compared in Table 5.

The numerical results show that the algorithm EECM is superior to two existing methods in the sense of both maximum error and cpu time.

Table 3. Results for Example 3.2 on $[0, 5]$ with step size $h = 2^{-n}$, $n = 6, \cdots, 11$

n	$Err(h)$	rate
6	2.68×10^{-1}	–
7	7.47×10^{-3}	5.16
8	2.39×10^{-4}	4.97
9	1.09×10^{-5}	4.46
10	5.84×10^{-7}	4.22
11	3.39×10^{-8}	4.11

Table 4. Numerical comparison for Example 3.2, using EECM, ODE15s and ODE23s

EECM		ODE15s		ODE23s	
$Err(h)$	CPU	$Err(h)$	CPU	$Err(h)$	CPU
3.39×10^{-8}	0.05	2.38×10^{-8}	0.42	4.31×10^{-6}	65.07

Table 5. Results for Example 3.3, using EECM, ODE15s and ODE23s

EECM		ODE15s		ODE23s	
$Err(h)$	CPU	$Err(h)$	CPU	$Err(h)$	CPU
6.53×10^{-14}	0.001	2.16×10^{-13}	0.23	3.63×10^{-11}	51.43

4. Conclusion

Explicit one-step exponentially fitted error correction methods with convergence order 4 for solving initial value problems are developed using an exponential local approximation platform. The stability properties are analyzed and it is shown that the proposed methods are L-stable. It is remarkable that the *good stability* and fourth-order convergence are obtained even though the scheme is an one-step explicit type and do not any linear matrix solver.

References

[1] C. E. Abhulimen, G. E. Omeike, *A sixth-order exponentially fitted shceme for the numerical solution of systems of ordinary differential equations*, J. Appl. Math. & Bioinf., **1**(2011), 175-186.

[2] R. R. Ahmad, N. Yaacob, A. H. Mohd Murid, *Explicit methods in solving stiff ordinary differential equations*, Int. J. Comput. Maht., **81**(2004), 1407-1415.

[3] J. Alverez, J. Rojo, *An improved class of generalized Runge-Kutta methods for stiff problems. Part I: The scalar case*, Appl. Math. Comput., **130**(2002), 537-560.

[4] L. Brugnano and C. Magherini, *Blended implementation of block implicit methods for ODEs*, Appl. Numer. Math., **42**(2002), 29-45.

[5] J. R. Cash, *On the exponential fitting of composite, multiderivative linear multistep methods*, SIAM J. Numer. Anal., **18**(1981), 808-821.

[6] M. V. Daele and G. V. Berghe, *Extended one-step methods: an exponential fitting approach*, Appl. Num. Anal. Comp. Math., **1**(2004), 353-362.

[7] G. Dahlquist, *A special stability problem for linear multistep methods*, BIT, **3**(1963), 27-43.

[8] L. W. Jackson and S. K. Kenue, *A fourth order exponetially fitted method*, SIAM J. Numer. Anal., **11**(1974), 965-978.

[9] W. Liniger and R. A. Willoughby, *Efficient integration methods for stiff systems of ordinary differential equations*, SIAM J. Numer. Anal., **7**(1970), 47-65.

[10] 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.

[11] S. D. Kim, X. Piao and P. Kim, *Convergence on Error correction methods for initial value problems*, J. Comp. Appl. Math., to appear.

[12] H. Ramos, *A non-standard explicit integration scheme for initial-value problems*, Appl. Math. Comp., **189**(1)(2007), 710-718.

[13] D. Voss, *A fifth-order exponentially fitted formula*, SIAM J. Numer. Anal., **25**(1988), 670-678.

[14] X. Y. Wu, *A sixth-order A-stable explicit one-step method for stiff systems*, Comput. Math. Appl., **35**(1998), 59-64.