AN ERROR CORRECTED EULER METHOD FOR SOLVING STIFF PROBLEMS BASED ON CHEBYSHEV COLLOCATION

PHILSU KIM†, XIANGFAN PIAO†, AND SANG DONG KIM†

Abstract. In this paper, we present error corrected Euler methods for solving stiff initial value problems, which not only avoid unnecessary iteration process that may be required in most implicit methods but also have such a good stability as all implicit methods possess. The proposed methods use a Chebyshev collocation technique as well as an asymptotical linear ordinary differential equation of first-order derived from the difference between the exact solution and the Euler’s polygon. In particular, it is proved that the proposed methods have a convergence order up to 4 regardless of the usage of the Jacobian. Numerical tests are given to support the theoretical analysis as evidences.

Key words. Chebyshev collocation method, Runge–Kutta method, stiff initial value problem, A-stability

AMS subject classifications. 65L04, 65L07, 65L20

DOI. 10.1137/100808691

1. Introduction. Most popular methods for solving stiff initial value problems \[ \phi'(t) = f(t,\phi(t)), \quad t > t_0; \quad \phi(t_0) = \phi_0 \] are types of implicit methods. For example, one can refer to many approaches based on Runge–Kutta or power series (see [2, 8, 9, 11, 19, 21, 37, 38], etc.) and also to various schemes based on modifications of the classical BDF (see [5, 12, 13, 15, 18, 20, 22, 25, 30, 35], etc.).

For implicit methods, one has to solve nonlinear systems for which one may use the most reliable and costly Newton’s method which requires the Jacobian matrix evaluated at each iteration (see [27], for example). To reduce its cost by an iteration method, numerous different approaches have been reported in [3, 4, 7, 10, 14, 23, 24]. One possibility to get a low complexity for a stiff solver is to avoid the iteration process. The Rosenbrock (or linearly implicit) method (see [33]), which has been favorably modified over many years (see [34, 36]), uses the Jacobian matrix directly in a numerical formula rather than within the iterations of Newton’s method. In general, the s-stage Rosenbrock methods require to solve s linear systems containing the Jacobian matrix at each integration step (see [20, p. 111]).

Another important issue for stiff problems is to check the stability of a numerical scheme. The A-stability for a numerical method is important because one can choose the step size based only on accuracy, without worrying about its stability constraints. It has been well known that an explicit k-step method cannot be A-stable and the order of an A-stable linear multistep method cannot exceed 2 (see [16, 6]). But, Ramos [31] recently suggested a nonstandard explicit numerical integration method which keeps A-stable with second-order accuracy for solving stiff initial value problems. Despite such developments on explicit methods, most implicit methods mentioned above have been used to take advantage of stability for solving stiff problems.

*Received by the editors September 15, 2010; accepted for publication (in revised form) July 29, 2011; published electronically November 1, 2011. This research 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 2010-0008317).
http://www.siam.org/journals/sinum/49-6/80869.html
†Department of Mathematics, Kyungpook National University, Daegu 702-701, Korea (kimps@knu.ac.kr, piaoxf76@hanmail.net, skim@knu.ac.kr).
The primary goal of this paper is to construct error corrected Euler’s methods which do not require such iteration steps for nonlinear discrete systems in most implicit methods but have both such good stability properties as implicit methods have and increase order of accuracy up to 4. These objects can be obtained by solving an asymptotical linear first-order ordinary differential equation (ODE) for the difference between the exact solution and the Euler’s polygon first and then adding its solution to the Euler’s approximation. More concisely, at each time step \([t_m, t_{m+1}]\), the algorithm we propose can be stated with the explicit form

\[
y_{m+1} = y_m + hf(t_m, y_m) + \beta_n
\]

where \(y_m\) is the approximation of \(\phi(t)\) at the time \(t_m\), \(h = t_{m+1} - t_m\) is the time step size and the correction term \(\beta_n\) is obtained by solving a linear system (see (3.19)) whose leading vector depends only on the Euler’s polygon defined by the previous step approximation \(y_m\). For the approximation of the asymptotical linear ODE, the Chebyshev collocation method (CCM) in [29, 30] is used which is an implicit type method and employs the Chebyshev interpolation polynomial. Two schemes of explicit types according to the usages of the Jacobian (i.e., the partial derivative \(f_\phi\)) are proposed. It is proved that both methods have a convergence order up to 4. Also, the stability analysis is performed and almost \(L\)-stability is shown, which coincides with the result in [30].

This paper is organized as follows. In section 2 we first review basic formulations and some properties for the Chebyshev interpolation polynomials required for our development. Section 3 is devoted to the derivation of the approximate scheme based on the CCM. In section 4, we analyze the convergence for the proposed schemes and present a geometric interpretation for the scheme. The stability properties for the developed schemes are discussed in section 5. Two test problems are performed in section 6 to give numerical evidences for the theoretical analysis in sections 4 and 5. Finally, we provide some comments and conclusions in section 7.

2. Review on Chebyshev interpolation polynomials. In this section, we will review the Chebyshev interpolation polynomial for a given function \(g \in C_k(-1,1)\), where \(C_k\) is the space of all functions whose \(k\) times derivatives are continuous on \((-1,1)\) for the rest of the paper. Most contents in this section can be found in [26].

Let \(T_k(s) = \cos(k \cos^{-1}(s))\) be the first kind Chebyshev polynomial of degree \(k\) and let \(\{s_j\}_{j=0}^{n}\) be the Chebyshev–Gauss–Lobatto (CGL) points such that

\[
s_j := \cos \left( \frac{(n-j)\pi}{n} \right), \quad j = 0, 1, \ldots, n.
\]

Let \(l_k(s)\) be the Lagrange basis defined by

\[
l_k(s) = \frac{\alpha_k}{n} \sum_{j=0}^{n} T_j(s_k)T_j(s), \quad \alpha_k = \begin{cases} 1 & \text{for } k = 0, n, \\ 2 & \text{for } k = 1, \ldots, n - 1, \end{cases}
\]

where the double prime indicates that both the first and last terms in the summation are to be halved. Then, the Chebyshev interpolation polynomial for a function \(g\) can be written as

\[
p_n(s) = \sum_{k=0}^{n} g(s_k)l_k(s).
\]
The convergence for the Chebyshev interpolation polynomial can be summarized as follows.

**Theorem 2.1.** Assume that \( g \in C^{n+2}(-1,1) \). For a nonnegative integer \( k \leq n+2 \), let

\[
M^k_g := \max_{s \in [-1,1]} |g^{(k)}(s)|.
\]

Then, the truncation \( \rho_n(s) := g(s) - p_n(s) \) and its derivative can be estimated by

\[
|\rho_n(s)| \leq \frac{M^{n+1}_g}{2^{n-1}(n+1)!}, \quad |\dot{\rho}_n(s)| \leq \frac{1}{2^{n-1}(n+1)!} \left( 2nM^{n+1}_g + \frac{M^{n+2}_g}{n+2} \right).
\]

**Proof.** First we rewrite the interpolation polynomial \( p_n(s) \) in terms of the Lagrangian form

\[
p_n(s) = \sum_{j=0}^{n} \frac{q_n(s)}{(s - \theta_j) \prod_{k=0}^{n} (\theta_j - \theta_k)} g(s_j),
\]

where

\[
q_n(s) = \prod_{j=0}^{n} (s - \theta_j) = 2^{-n+1}(s^2 - 1)U_{n-1}(s), \quad \theta_j = \cos s_j,
\]

where \( U_{n-1}(s) \) is the second kind Chebyshev polynomial of degree \( n - 1 \). Then, the truncation error \( \rho_n(s) \) becomes

\[
\rho_n(s) = \frac{q_n(s)g^{(n+1)}(\xi_s)}{(n+1)!}, \quad \xi_s \in (-1,1),
\]

where the mean-value point \( \xi_s \) depends continuously on \( s \) and

\[
\frac{d}{ds}g^{(n+1)}(\xi_s) = \frac{g^{(n+2)}(\chi_s)}{n+2}
\]

for some \( \chi_s \in (-1,1) \) (see [17, 32, (18)] and the references therein). Thanks to the change of variable \( s = \cos t \), it may be shown that the polynomial \( q_n(s) \) in (2.5) can be simplified to

\[
q_n(s) = q_n(\cos t) = -\frac{\sin t \sin nt}{2^{n-1}}.
\]

This equation yields

\[
|q_n(s)| \leq \frac{1}{2^{n-1}} \quad \text{and} \quad \left| \frac{d}{ds}q_n(s) \right| \leq \frac{2n}{2^{n-1}}.
\]

Thus, (2.6) and the first inequality of (2.8) show the first inequality of (2.4). Similarly, by differentiating both sides of (2.6), the second inequality of (2.4) can be obtained from (2.7) and the second inequality of (2.8). \( \square \)
Before closing this section, we observe basic properties for the basis function \( l_k(s) \) in (2.2). First, note that the polynomials \( l_k(s) \) satisfy

\[
l_k(s_j) = \delta_{jk}, \quad \sum_{k=0}^{n} l_k(s) = 1, \quad s \in [-1, 1],
\]

where \( \delta_{jk} \) denotes the Kronecker delta function and that \( \dot{T}_k = \frac{dT_k}{ds} \) satisfies the orthogonality property such that

\[
\int_{0}^{\pi} \dot{T}_k(\cos \theta)\dot{T}_l(\cos \theta) \sin^2 \theta \, d\theta = \frac{k \pi}{2} \delta_{kl}.
\]

Let \( \eta_j \) be arbitrary distinct points in \([-1, 1]\) arranged by \(-1 \leq \eta_1 < \cdots < \eta_n \leq 1\). Using these points \( \{\eta_j\} \) and functions \( l_k \), define the matrix \( \mathcal{L} \) as

\[
\mathcal{L} = \begin{pmatrix} L_{jk} \end{pmatrix}, \quad \text{where} \quad L_{jk} := \dot{l}_k(\eta_j) = \frac{dl_k}{ds}(\eta_j), \quad j, k = 1, \ldots, n.
\]

Then, \( \mathcal{L} \) is nonsingular, which is shown in the following lemma.

**Lemma 2.2.** The matrix \( \mathcal{L} \) is nonsingular.

**Proof.** For the vector \( \mathbf{x} = [x_1, \ldots, x_n]^T \) satisfying \( \mathcal{L} \mathbf{x} = 0 \), where the superscript \( T \) denotes the transpose, the polynomial \( p(s) = \sum_{k=1}^{n} x_k \dot{l}_k(s) \) should be identically zero because \( p(s) \) has \( n \) zeros \( \{\eta_j\}_{j=1}^{n} \), but the degree \( p(s) \) is \( n - 1 \). Hence, the definition of \( l_k(s) \) in (2.2) and the orthogonality (2.10) lead to the linear system

\[
\sum_{k=1}^{n} T_j(s_k) x_k \alpha_k = 0, \quad j = 1, \ldots, n.
\]

Thus, it follows that for each fixed \( l = 1, \ldots, n \),

\[
0 = \sum_{j=1}^{n} \left( \sum_{k=1}^{n} T_j(s_k) x_k \alpha_k \right) T_j(s_l) = \sum_{k=1}^{n} x_k \alpha_k \left( \sum_{j=1}^{n} T_j(s_k) T_j(s_l) - \frac{1}{2} \right),
\]

where the prime notation indicates that the last term in the summation is to be halved. Applying the first identity of (2.9) into the last summation above, we have

\[
0 = \sum_{k=1}^{n} x_k \alpha_k \left( \frac{n \delta_{lk} - \frac{1}{2}}{\alpha_k} \right) = n x_l - \frac{1}{2} \sum_{k=1}^{n} x_k \alpha_k, \quad l = 1, \ldots, n,
\]

which implies that all coefficients \( x_l \) are the same, that is, \( x_1 = x_2 = \cdots = x_n \equiv C \). If \( C \) is zero, then the proof will be done. Otherwise, from (2.11) we have \( \sum_{k=1}^{n} T_j(s_k) = 0 \), which is impossible. Hence \( C = 0 \). These arguments complete the proof. \( \square \)

Let us define the vector \( \mathbf{a} \) as

\[
\mathbf{a} := [\dot{l}_0(\eta_1), \ldots, \dot{l}_0(\eta_n)]^T.
\]

Then, the above lemma gives the following corollary.

**Corollary 2.3.** All components of the vector \( \mathcal{L}^{-1} \mathbf{a} \) should be exactly \(-1\).

**Proof.** By differentiating both sides of the second equation of (2.9) and evaluating the equation at \( \eta_j \), it is easy to show that

\[
-\dot{l}_0(\eta_j) = \sum_{k=1}^{n} \dot{l}_k(\eta_j), \quad j = 1, \ldots, n.
\]

This system can be written in the matrix form \( \mathcal{L} \mathbf{x} = -\mathbf{a}, \ \mathbf{x} = [1, \ldots, 1]^T \). Thus, the invertibility of \( \mathcal{L} \) leads to the conclusion. \( \square \)
3. Derivation of an approximation method. The target problem for developing an approximation method is a scalar stiff initial value problem of the form

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

In this initial problem (3.1), the function \(f\) is assumed to satisfy all the necessary requirements for the existence of a unique solution. In particular, the uniform boundedness for the partial derivative \(f_\phi\) in the range of the solution is assumed (that is, \(\|f_\phi\|_\infty < \infty\), where \(\|\cdot\|_\infty\) denotes the supremum norm).

The goal of this section is to derive an accurate algorithm to obtain the next approximation \(y_{m+1}\) at the point \(t_{m+1} = t_m + h\) for a given approximation \(y_m\) at the time \(t_m\) to the exact solution \(\phi(t)\) for (3.1). The uniform time step \(h := t_{m+1} - t_m, m = 0, 1, \ldots\), will be used from now on. As mentioned in the introduction, a standard approach to get the approximation \(y_{m+1}\) for the original equation (3.1) creates a nonlinear system of equations with the Jacobian for \(f\). Thus, in this section, we begin with a derivation of an asymptotic linear type ODE to the original problem (3.1).

Now assuming the approximation \(y_m\) at \(t_m\), consider the Euler polygon \(y(t)\) (see [20]) on the interval \([t_m, t_{m+1}]\) such that

\[
y(t) := y_m + (t - t_m)f(t_m, y_m), \quad t \in [t_m, t_{m+1}].
\]

Let \(\psi(t)\) be the difference between the exact solution \(\phi(t)\) and the Euler polygon \(y(t)\), which is

\[
\psi(t) := \phi(t) - y(t).
\]

Then, it can be shown that \(\psi(t)\) satisfies an ODE

\[
\psi'(t) = g(t)\psi(t) + F(t), \quad t \in (t_m, t_{m+1}),
\]

where \(g(t)\) and \(F(t)\) are given by

\[
g(t) = \int_0^1 f_\phi(t, y(t) + \xi \psi(t))d\xi
\]

and

\[
F(t) = f(t, y(t)) - f(t_m, y_m), \quad t \in (t_m, t_{m+1}).
\]

Using (3.3), an approximation of the solution \(\psi(t)\) of (3.4) can be used to get the next approximation \(y_{m+1}\) of \(\phi(t_{m+1})\) for a given \(y_m\) at \(t_m\). Since the function \(g\) in (3.5) contains the unknown \(\psi(t)\) yet, its modification is required as follows. First, by the change of variable \(t = t_s = t_m + \frac{h}{2}(1 + s)\) from the computational region \([t_m, t_{m+1}]\) to the reference domain \([-1, 1]\), we consider the solution \(\tilde{\psi}(s)\) on \([-1, 1]\), which is

\[
\tilde{\psi}(s) := \psi(t) = \psi(t_s), \quad s \in [-1, 1].
\]

Then, instead of (3.4), one may have

\[
\tilde{\psi}'(s) = \frac{h}{2}\left(g(t_s)\tilde{\psi}(s) + F(t_s)\right), \quad s \in (-1, 1).
\]
Due to the mean value theorem, there is a function $\nu(t, \xi)$ between $y(t) + \xi \psi(t)$ and $y(t)$ so that

$$g(t) - f_\phi(t, y(t)) = \psi(t) \int_0^1 \xi f_\phi(t, \nu(t, \xi)) d\xi.$$\]

Hence, we arrive at

$$|g(t) - f_\phi(t, y(t))| \leq C|\psi(\xi)|,$$

where $C = \max |f_\phi(t, \chi)|$. Thus, it is possible to replace the function $g(\cdot)$ by $f_\phi(\cdot, y(\cdot))$ in (3.8) within a range of the difference function $\psi(\xi)$. Therefore, one has the following asymptotic first-order linear ODE in the reference domain $(-1, 1)$ instead of (3.8):

$$\bar{\psi}'(s) = \frac{\hbar}{2} \left( f_\phi(t_s, y(t_s)) \bar{\psi}(s) + F(t_s) \right) + O\left( h^2 \bar{\psi}(s)^2 \right).$$

Note that the original problem (3.1) is nonlinear in general, but (3.9) has a linearity with respect to $\bar{\psi}$ provided the asymptotic term $O(h^2 \bar{\psi}(s)^2)$ is ignored. This linearity in (3.9) could make it easier to develop an approximation scheme for $y_{m+1}$, which will be explored in this paper.

**Remark 3.1.** The calculation of the partial derivative $f_\phi$ in (3.9) may be costly. One way to avoid the calculation of the partial derivative $f_\phi$ is to approximate $f_\phi$ in (3.9) with the forward difference quotient

$$f_\phi(t_s, y(t_s)) \approx \frac{f(t_s, y(t_s) + h^\lambda) - f(t_s, y(t_s))}{h^\lambda}.$$\]

In this case, the asymptotic part in (3.9) will be changed with $O(h^2 \bar{\psi}(s)^2 + h^{\lambda+1} \bar{\psi}(s))$. Here, $\lambda$ is a parameter to be determined later.

Now, considering Remark 3.1, we will write the asymptotic linear ODEs for two cases in one form

$$\bar{\psi}'(s) = \frac{\hbar}{2} \left( \varphi(s) \bar{\psi}(s) + F(t_s) \right) + \omega(s),$$

where $\omega(s)$ is either

$$\omega(s) = O\left( h^2 \bar{\psi}(s)^2 \right) \quad \text{if } \varphi(s) = f_\phi(t_s, y(t_s))$$

or

$$\omega(s) = O\left( h^{\lambda+1} \bar{\psi}(s) + h^2 \bar{\psi}(s)^2 \right) \quad \text{if } \varphi(s) = \frac{f(t_s, y(t_s) + h^\lambda) - f(t_s, y(t_s))}{h^\lambda}.$$\]

**Remark 3.2.** It will be analyzed in the following section that the asymptotic part $\omega(s)$ defined in either (3.11) or (3.12) is so small that it can be ignored. Hence, by truncating the term $\omega(s)$ from (3.10), the equation will be completely linear and the approximation scheme for the solution will be an explicit type to solve quite easily, which is a remarkable fact.

The remainder of this section is devoted to deriving an approximation scheme for (3.10) based on the Chebyshev interpolation polynomial discussed in section 2 and to discuss the approximation of $y_{m+1}$ with the relations (3.2), (3.3), (3.7), and the Lagrange basis $l_k(s)$. With the CGL points $s_k \in [-1, 1]$ in (2.1) and the basis function

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
l_k(s) in (2.2), let us now approximate the solution \( \tilde{\psi}(s) \) of (3.10) by the Chebyshev interpolation polynomial \( p_n(s) \) such that

\[
(3.13) \quad p_n(s) = \sum_{k=0}^{n} \tilde{\psi}(s_k) l_k(s),
\]

whose error \( \rho_n(s) \) is given by the relation

\[
\tilde{\psi}(s) = p_n(s) + \rho_n(s).
\]

From now on, we assume that there is a fixed positive constant \( N_0 \) and we will use the degree \( n \) in the range \( 1 \leq n \leq N_0 \) for \( p_n(s) \). Then, combining (3.13) with (3.10) yields the way how the coefficients \( \tilde{\psi}(s_k), k = 1, \ldots, n \), are determined by collocating the residual \( r(s) \), which is

\[
\sum_{k=0}^{n} \tilde{\psi}(s_k) l_k(s) - \frac{h}{2} \left( \varphi(s) \sum_{k=0}^{n} \tilde{\psi}(s_k) l_k(s) + F(t_s) \right) = r(s).
\]

By collocating the residual \( r(s) \) at \( n \)-points \( s_j, j = 1, \ldots, n \), we have the discrete system

\[
(3.14) \quad \sum_{k=1}^{n} \tilde{\psi}(s_k) a_{jk} = \frac{h}{2} F(t_{s_j}) - \tilde{\psi}(s_0) a_{j0} + r_j, \quad 1 \leq j \leq n,
\]

where

\[
(3.15) \quad a_{jk} := \hat{l}_k(s_j) - \frac{h}{2} \varphi(s_j) \delta_{jk}, \quad r_j := \frac{h}{2} \varphi(s_j) \rho_n(s_j) - \rho_n(s_j) + \omega(s_j),
\]

where \( \omega(s) \) is defined in either (3.11) or (3.12). Here, we define vectors \( \mathbf{r} \) and \( \mathbf{b} \) as

\[
(3.16) \quad \mathbf{r} = [r_1, \ldots, r_n]^T, \quad \mathbf{b} = [a_{10}, \ldots, a_{n0}]^T.
\]

**Remark 3.3.** Two quantities \( \tilde{\psi}(s_0) \) and \( r_j \) in (3.14) can be neglected because (i) \( \tilde{\psi}(s_0) = \phi(t_m) - y_m \) is the error induced from the previous time step \([t_{m-1}, t_m]\) and is quite small (see Theorem 4.4), (ii) the first two terms of \( r_j \) are the main parts of \( r_j \), which are the errors induced from the truncation error \( \rho_n(s) \) for the interpolation polynomial and are quite small (see Corollary 4.1).

For convenience, let us define matrices as

\[
(3.17) \quad \mathbf{A} = (a_{jk}), \quad \mathbf{L} = (L_{jk}), \quad \mathbf{J} = (J_{jk}), \quad 1 \leq j, k \leq n,
\]

where \( a_{jk} \) is defined in (3.15) and

\[
L_{jk} := \hat{l}_k(s_j) \quad \text{and} \quad J_{jk} := \varphi(s_j) \delta_{jk},
\]

and vectors as

\[
(3.18) \quad \mathbf{d} = [\beta_1, \ldots, \beta_n]^T, \quad \mathbf{f} = [F(t_{s_1}), \ldots, F(t_{s_n})]^T.
\]
Then, based on Remark 3.3, instead of solving (3.14) for \( \bar{\psi}(s_k) \), we will approximate the exact solution \( \bar{\psi}(s_k) \) by \( \beta_k \) which satisfies the discrete Chebyshev collocation system

\[
\mathcal{A}d = \left( \mathcal{L} - \frac{h}{2} \mathcal{J} \right) d = \frac{h}{2} f.
\]

On the other hand, the intrastep error \( t = [\epsilon_1, \ldots, \epsilon_n]^T \) satisfies the system

\[
\mathcal{A}t = r - \bar{\psi}(s_0)b,
\]

where \( r \) and \( b \) are defined in (3.16) and

\[
\epsilon_k := \bar{\psi}(s_k) - \beta_k \quad \text{for} \quad k = 1, 2, \ldots, n.
\]

The solvability of the linear system (3.19) is given in the following lemma.

**Lemma 3.4.** Assume that \( f_0 \) is uniformly bounded and that there is a positive integer \( N_0 \) such that \( 1 \leq n \leq N_0 \). Then for a sufficiently small time size \( h \), the matrices \( \mathcal{L} \) and \( \mathcal{J} \) in (3.17) are uniformly bounded and the matrix \( \mathcal{A} \) is invertible. Furthermore, the matrix \( \mathcal{A}^{-1} \) is uniformly bounded and

\[
\mathcal{A}^{-1} = \left( \mathcal{L} - \frac{h}{2} \mathcal{J} \right)^{-1} = \mathcal{L}^{-1} + O(h).
\]

**Proof.** Note that \( \mathcal{L} \) is nonsingular from Lemma 2.2 and \( \mathcal{L} \) depends only on the degree \( n \) of the Chebyshev interpolation polynomial. Thus, \( \|L\| \) and \( \|L^{-1}\| \) (here \( \| \cdot \| \) denotes a matrix-norm) are uniformly bounded independent of the step size \( h \). Also, the uniform boundedness of \( f_0 \) gives that for fixed \( n \), \( \|J\| \) is uniformly bounded independent of the step size \( h \). Hence, the invertibility of \( \mathcal{A} \) and (3.22) are followed by Lemma 2.2 and geometric series. Further, the inequality

\[
\|A^{-1}\| \leq \|L^{-1}\| \left\| \left( I - \frac{h}{2} \mathcal{L}^{-1} \mathcal{J} \right)^{-1} \right\| \leq \frac{\|L^{-1}\|}{1 - \frac{h}{2} \|L^{-1}\| \|J\|}
\]

shows the uniform boundedness for \( \mathcal{A}^{-1} \) provided the time step \( h \) is sufficiently small. \( \square \)

Solving the system (3.19), we obtain, in particular, the required approximation value at the final point on the interval, \( \psi(t_{m+1}) = \bar{\psi}(s_n) \approx \beta_n \). Also, the definitions of \( y(t) \), \( \psi(t) \), \( \psi(s) \) and \( \epsilon_n \) in (3.2), (3.3), (3.7), and (3.21), respectively, give

\[
\phi(t_{m+1}) = y(t_{m+1}) + \psi(t_{m+1})
= y(t_{m+1}) + \bar{\psi}(s_n)
= y_m + hf(t_m, y_m) + \beta_n + \epsilon_n,
\]

where \( \beta_n \) and \( \epsilon_n \) are the last components of the vectors \( d \) and \( t \) in (3.19) and (3.20), respectively. Therefore, one may define the approximation for \( \phi(t_{m+1}) \) by truncating the intrastep error \( \epsilon_n \) in (3.23) as follows:

\[
y_{m+1} = y_m + hf(t_m, y_m) + \beta_n.
\]

Here, one may say that the approximation procedure in (3.24) is an explicit type because the last term \( \beta_n \) is obtained by solving the linear system (3.19) and the right-hand side of (3.19) depends only on the previous value \( y_m \) from the definitions of \( F(t_s) \) and \( y(t) \) given in (3.6) and (3.2), respectively.
The explicit type algorithm for calculating $y_{m+1}$ in (3.24) can be summarized as follows:

**Algorithm 3.5. ECEM**($f, t_0, y_0, t_{end}, h$)

1. **Remark:** The problem being solved is $\phi' = f(t, \phi)$, $\phi(t_0) = y_0$, for $t_0 \leq t \leq t_{end}$, using the method described earlier in the section. The approximate solution values are printed at each node point.
2. Choose the step size $h$ and the degree $n$ of the interpolation polynomial.
3. Construct the matrix $L$.
4. Let $t_1 := t_0 + h$.
5. If $t_1 > t_{end}$, then exit.
7. Calculate the last component $\beta_n$ of $\frac{h}{2}A^{-1}f$.
8. Calculate $y_1 = y_0 + hf(t_0, y_0) + h\beta_n$.
9. Print $t_1, y_1$.
10. Set $t_0 := t_1$ and $y_0 := y_1$. Then go to step 4.

In the following section, we will provide the error analysis for $\phi(t_m) - y_m$ by analyzing the intrastep error $\epsilon_k = \psi(s_k) - \beta_k$.

**3.1. Simple extension to systems of ODEs.** In this subsection, we give a brief sketch on a direct extension of the proposed scheme (3.24) to a system of ODEs as follows. Consider

\begin{equation}
(3.25) \quad \Phi'(t) = F(t, \Phi(t)), \quad t \in (t_0, T]; \quad \Phi(t_0) = \Phi_0,
\end{equation}

where $\Phi = [\phi_1(t), \ldots, \phi_d(t)]^T$ and $F = [f_1(t, \Phi(t)), \ldots, f_d(t, \Phi(t))]^T$. Let us assume that $Y_m$ is a given approximation of $\Phi(t)$ at time $t = t_m$ and consider the Euler polygon defined by

\begin{equation}
Y(t) = Y_m + (t - t_m)F(t_m, Y_m), \quad t \in [t_m, t_{m+1}].
\end{equation}

Also, let $G(t) = [g_1(t), \ldots, g_d(t)]^T$ be the leading vector-valued function given by

\begin{equation}
G(t) = F(t, Y(t)) - F(t_m, Y_m).
\end{equation}

Now, by applying directly the procedures for deriving the system (3.19), it will become

\begin{equation}
(\mathbf{I}_d \otimes L - \frac{h}{2}J) \mathbf{c} = \frac{h}{2} \mathbf{g},
\end{equation}

where $\mathbf{I}_d \otimes L$ denotes the tensor product of the identity matrix $\mathbf{I}_d$ with size $d$ and $L$; the vector $\mathbf{g}$ is given by

\begin{equation}
\mathbf{g} = [g_1(t_{s_1}), \ldots, g_1(t_{s_n}), g_2(t_{s_1}), \ldots, g_d(t_{s_n})]^T,
\end{equation}

and the matrix $J = (J^{(\mu, \nu)})$, $1 \leq \mu, \nu \leq d$ is defined by

\begin{equation}
J^{(\mu, \nu)} = \left(\varphi_{\mu, \nu}(s_j) \delta_{jk}\right)_{n \times n},
\end{equation}

where $\varphi_{\mu, \nu}(s)$ is defined by either its forward approximation or

\begin{equation}
\varphi_{\mu, \nu}(s) = \frac{\partial}{\partial y_\nu}f_\mu(t_s, Y(t_s)).
\end{equation}
Note that for each \( i = 1, \ldots, d \), the \((n \times i)\)th component of \( c \) will be an approximation of the \( i \)th component of the vector \( \Phi(t_{m+1}) - Y(t_{m+1}) \). Hence after solving the system (3.26), we define a vector \( c^{(n)} := [c_n, c_{2n}, \ldots, c_{dn}]^T \), where \( c_j \) denotes the \( j \)th component of \( c \). Then one can approximate \( \Phi(t_{m+1}) \) with the formula

\[
\Phi(t_{m+1}) \approx Y(t_{m+1}) + c^{(n)} = Y_m + hf(t_m, Y_m) + c^{(n)}.
\]

4. Convergence analysis. In this section, we will analyze the actual error

\[
e_n = \phi(t_m) - y_m, \quad m = 1, 2, \ldots
\]

between the exact solution \( \phi \) of (3.1) and the approximate solution \( y_m \) obtained by Algorithm 3.5.

We begin this section with the estimation of the truncation error \( \rho_n(s) = \bar{\psi}(s) - p_n(s) \) for the Chebyshev interpolation \( p_n(s) \) defined in (3.13), which can be easily estimated by Theorem 2.1 as follows.

**Corollary 4.1** (local truncation error estimate). Assume that the solution \( \phi(t) \) for the problem (3.1) is in the space \( C^{n+2}([t_0, T]) \), \( n \geq 1 \), and for a nonnegative integer \( k \leq n + 2 \) and \( m \geq 0 \), let

\[
N_{\phi}^{k,m} := \max_{t \in [t_m, t_{m+1}]} |\phi^{(k)}(t)|.
\]

Then, the local truncation error \( \rho_n(s) = \bar{\psi}(s) - p_n(s) \) for the Chebyshev interpolation \( p_n(s) \) can be estimated by

\[
|\rho_n(s)| \leq \frac{N_{\phi}^{n+1,m}}{2^{n-1}(n+1)!} \left( \frac{h}{2} \right)^{n+1},
\]

\[
|\dot{\rho}_n(s)| \leq \left( \frac{h}{2} \right)^{n+1} \frac{1}{2^{n-1}(n+1)!} \left( 2nN_{\phi}^{n+1,m} + \frac{hN_{\phi}^{n+2,m}}{2(n+2)} \right).
\]

**Proof.** First note that \( \bar{\psi}(s) \in C^{n+2}[t_m, t_{m+1}] \) for each \( m = 0, 1, \ldots \). Hence, according to Theorem 2.1, it is enough to estimate \( M_{\psi}^k \) in (2.3) for the function \( \bar{\psi}(s) \) defined in (3.7). By the chain rule, (3.7) shows

\[
\bar{\psi}^{(k)}(s) = \left( \frac{h}{2} \right)^k \psi^{(k)} \left( t_m + \frac{h}{2}(1 + s) \right).
\]

Thus, it follows that

\[
M_{\psi}^k = \max_{s \in [-1, 1]} |\bar{\psi}^{(k)}(s)| = \left( \frac{h}{2} \right)^k \max_{t \in [t_m, t_{m+1}]} |\psi^{(k)}(t)| = \left( \frac{h}{2} \right)^k N_{\phi}^{k,m} = \left( \frac{h}{2} \right)^k N_{\phi}^{k,m},
\]

where the last equation holds for \( k > 1 \), which follows from (3.2) and (3.3). These arguments complete the proof. \( \square \)

From Corollary 4.1, each component \( r_j \) of the vector \( r \) in (3.16) can be estimated as follows.

**Lemma 4.2.** Assume that the parameter \( \lambda \) in Remark 3.1 is larger than 2 and the solution \( \phi(t) \) for the problem (3.1) is in the space \( C^{n+2}([t_0, T]) \), \( n \geq 1 \). Then, for each \( j = 1, \ldots, n \), the remainder \( r_j \) in (3.15) can be estimated by

\[
|r_j| \leq \left( \frac{h}{2} \right)^{n+1} E_{f,\phi,n,m} + C_j(h\epsilon_j^2 + h^3|\epsilon_j| + h^5),
\]
where $C_j$ is a positive constant independent of $h$, $\epsilon_j$ is the $j$th component of $\mathbf{t}$ defined in (3.21) and $E_{f,\phi,n,m}$ is the constant defined by

$$E_{f,\phi,n,m} := \frac{hN_{\phi}^{n+1,m} \|\mathbf{f}_\phi\|_\infty}{2^n(n+1)!} + \frac{1}{2^{n-1}(n+1)!} \left( 2nN_{\phi}^{n+1,m} + \frac{hN_{\phi}^{n+2,m}}{2(n+2)} \right).$$

**Proof.** Using Taylor’s expansion for the function $F(t)$ in (3.6) and the definition of the Euler polygon $y(t)$ in (3.2), it follows that $F(t) = O(h)$. Then from (3.18), (3.19), and Lemma 3.4 one may have

$$\beta_j = O\left( h^2 \right),$$

where $\beta_j$ is the $j$th component of the solution $\mathbf{t}$ of the system (3.19). Thus, for both functions $\omega(s)$ defined in either (3.11) or (3.12), the term $\omega(s_j)$ in (3.15) can be easily estimated as follows. For the case of $\omega(s)$ in (3.11), using (4.2) and (3.21), we have

$$|\omega(s_j)| = O\left( h\tilde{\omega}(s_j)^2 \right)$$
$$= O\left( h^2\tilde{\omega}(s_j)^2 + 2h(\tilde{\omega}(s_j) - \beta_j)^2 + h^2\beta_j^2 \right)$$
$$\leq C_j(h\epsilon_j^2 + h^3|\epsilon_j| + h^5),$$

where $C_j$ is a positive constant independent of $h$. For the case of $\omega(s)$ in (3.12), we also have a similar estimation if $\lambda \geq 2$ as follows:

$$|\omega(s_j)| = O\left( h\tilde{\omega}(s_j)^2 + h^{\lambda+1}\tilde{\omega}(s_j) \right)$$
$$= O\left( h^2\tilde{\omega}(s_j)^2 + (\tilde{\omega}(s_j) - \beta_j)(2\beta_j + h^\lambda + \beta_j^2) \right)$$
$$\leq C_j(h\epsilon_j^2 + h^3|\epsilon_j| + h^5),$$

where $C_j$ is also a positive constant independent of $h$.

Note that both functions $\varphi(s)$ defined in either (3.11) or (3.12) can be bounded by $\|f_\phi\|_\infty$. Thus, by applying Corollary 4.1 to

$$q_j := r_j - \omega(s_j) = \frac{h}{2}\varphi(s_j)\rho_n(s_j) - \rho_n(s_j),$$

one may get

$$|q_j| \leq \left( \frac{h}{2} \right)^{n+1} E_{f,\phi,n,m}, \quad j = 1, \ldots, n.$$

Combining three estimations (4.3), (4.4), and (4.5) leads to (4.1). \qed

From Lemma 4.2, we can estimate the last component $\epsilon_n$ of the intrastep error $\mathbf{t}$ in (3.20) as follows.

**Lemma 4.3.** Assume that the same assumptions of Lemma 4.2 holds and the quantity $\epsilon_m = \phi(t_m) - y_m = \psi(t_m) - \psi(s_0)$ is sufficiently small. Then, for a fixed $n \geq 1$ and a sufficiently small step size $h$, the last component $\epsilon_n$ of the vector $\mathbf{t}$ can be estimated by

$$|\epsilon_n| \leq (1 + Ch)^2(1 + Ch|\psi(s_0)|)|\psi(s_0)| + Dh^{\min\{n+1,5\}},$$

where $C$ and $D$ are some constants independent of time step size $h$. 

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited.
Proof. Before starting the proof, let $\bar{a}_{ij}$ be the $(i,j)$ component of the matrix $A^{-1}$ for simplicity. Note that the vector $b$ in (3.16) is exactly the same with the vector $a$ in (2.12) because (3.15) shows $a_{j0} = \dot{\bar{l}}_0(s_j)$. Hence, using Lemma 3.4, $A^{-1}b$ can be written as

$$A^{-1}b = A^{-1}a = \mathcal{L}^{-1}a + O(h)$$

and hence from Corollary 2.3, its $j$th component $(A^{-1}b)_j$ can be estimated by

$$(4.6) \quad |(A^{-1}b)_j| \leq 1 + O(h).$$

Also, if $\bar{r}_j$ is the $j$th component of the vector $A^{-1}r$ for the vector $r$ in (3.16), then Lemmas 3.4 and 4.2 show that

$$(4.7) \quad |\bar{r}_j| \leq \sum_{k=1}^{n} |\bar{a}_{jk}|\varsigma_k + O\left(h^{\min\{n+1.5\}}\right),$$

where $\bar{a}_{jk}$ is the $(j,k)$ component of $A^{-1}$ and each $\varsigma_k$ is defined by

$$(4.8) \quad \varsigma_k = C_j \left(h\epsilon_k^2 + h^3|\epsilon_k|\right), \quad k = 1, \ldots, n.$$}

Thus, using (4.6) and (4.7), the $j$th component $\epsilon_j$ of the intrastep error $t = A^{-1}(r - \bar{\psi}(s_0)b)$ of (3.20) can be estimated by

$$|\epsilon_j| \leq |\bar{r}_j| + |\bar{\psi}(s_0)||A^{-1}b_j| \leq O(h^{\min\{n+1.5\}}) + \sum_{k=1}^{n} |\bar{a}_{jk}|\varsigma_k + (1 + O(h))|\bar{\psi}(s_0)|.$$

Hence, if we let $|\epsilon_{j_0}|$ be the maximum among $|\epsilon_j|$, $j = 1, \ldots, n$, then (4.8) and (4.10) show that for each $j = 1, \ldots, n,$

$$|\epsilon_j| \leq (1 + c_1h)|\bar{\psi}(s_0)| + c_2h^{\min\{n+1.5\}} + c_3\sum_{k=1}^{n} h|\epsilon_k| + h^2$$

$$\leq (1 + c_1h)|\bar{\psi}(s_0)| + c_2h^{\min\{n+1.5\}} + nc_3h|\epsilon_{j_0}|(1 + h^2)$$

$$\leq (1 + c_4h)|\bar{\psi}(s_0)| + c_2h^{\min\{n+1.5\}} + c_4h|\epsilon_{j_0}|(1 + h^2)$$

for some positive constants $c_i$ $(i = 1, 2, 3)$ independent of $h$ and $c_4 = \max\{c_1, nc_3\}$. Note that

$$\delta(h) := (1 + c_4h)|\bar{\psi}(s_0)| + c_2h^{\min\{n+1.5\}},$$

is sufficiently small. Hence the inequality (4.9) gives

$$(4.10) \quad c_4h|\epsilon_{j_0}|^2 - (1 - c_4h^3)|\epsilon_{j_0}| + \delta(h) \geq 0.$$}

Then, the inequality (4.10) implies that $|\epsilon_{j_0}|$ should be small because $\delta(h)$ is sufficiently small by the assumptions for $h$ and $\bar{\psi}(s_0)$. Further, since the discriminant of the quadratic function in the left-hand side of (4.10) is positive for sufficiently small $h$ and $\delta(h)$, the inequality can be solved as follows:

$$|\epsilon_{j_0}| \leq \frac{1 - c_4h^3 - g(h)}{2c_4h}, \quad g(h) = \sqrt{(1 - c_4h^3)^2 - 4c_4h\delta(h)}.$$
ERROR CORRECTION METHODS FOR SOLVING STIFF IVPS

From Taylor series for $g(h)$, we can have

$$|\epsilon_{f_0}| = (1 + c_5 h)|\tilde{\psi}(s_0)| + O(h^2)$$

for some constant $c_5$. By substituting it into (4.9) and taking $j = n$, we see that

$$|\epsilon_n| \leq (1 + c_4 h)|\tilde{\psi}(s_0)| + c_2 h^{\min\{n+1, 5\}} + c_4 h((1 + c_5 h)|\tilde{\psi}(s_0)| + c_6 h^2)^2$$

for some positive constant $c_6$ independent of $h$. Finally, taking

$$C = \max\{c_4, c_5, 2c_6\} \quad \text{and} \quad D = 2\max\{c_2, c_4 c_6^2\},$$

one may have

$$|\epsilon_n| \leq Dh^{\min\{n+1, 5\}} + (1 + Ch)|\tilde{\psi}(s_0)|\left(1 + c_4 h(1 + Ch)|\tilde{\psi}(s_0)| + Ch^2\right)$$

$$\leq (1 + Ch)^2(1 + Ch|\tilde{\psi}(s_0)|)|\tilde{\psi}(s_0)| + Dh^{\min\{n+1, 5\}},$$

which completes the proof.

Equation (3.23) and Lemma 4.3 say that the actual error $e_{m+1}$ satisfies the difference equation:

$$|e_{m+1}| \leq (1 + Ch)^2(1 + Ch|e_m|)|e_m| + Dh^{\min\{n+1, 5\}}, \quad m \geq 0; \quad e_0 = 0.$$

This difference relation gives the geometric meaning as shown in Figure 4.1 for the Algorithm 3.5. That is, the algorithm consists of two steps: (1) first $y_{m+1}$ is predicted with the Euler’s polygon $y(t)$ defined in (3.2), and (2) $y_{m+1}$ is corrected with the approximated value $\beta_n$ for the solution $\psi(t)$ of the ODE (3.4). In this sense, one may assert that the algorithm is a type of predictor-corrector method. As predictor-corrector methods, Adams–Bashforth and Adams–Moulton methods are well known and these are perfectly different with Algorithm 3.5. It is remarkable that Algorithm 3.5 is an explicit one. Also, if the term $\beta_n$ in Algorithm 3.5 is removed, then the algorithm coincides with Euler’s method. That is, one may say that the term $\beta_n$ is an error collected term for Euler’s method.
From the difference equation (4.11), we can estimate the actual error $e_m$ as follows.

**Theorem 4.4 (convergence).** For sufficiently small $h$, the actual error $e_m$ satisfies the inequality

$$|e_m| \leq \frac{D}{(1 + Ch)^3 - 1}(\exp(3CT) - 1)h^{\min\{n+1,5\}}, \quad m \geq 0,$$

where $C$ and $D$ are positive constants independent of $h$ and $m$ and $T$ is the end of the time in (3.1).

**Proof.** By mathematical induction, it is easy to show that the difference equation (4.11) can be solved by

$$|e_m| \leq \frac{(1 + Ch)^{3m} - 1}{(1 + Ch)^3 - 1}Dh^{\min\{n+1,5\}}, \quad m \geq 0.$$  

If $mh \leq T$, then $1 + Ch \leq \exp(Ch)$ and $(1 + Ch)^m \leq \exp(3mCh) \leq \exp(3CT)$. Thus, the inequality (4.13) shows that the actual error $e_m$ has the bound (4.12). 

**Remark 4.5.** Theorem 4.4 shows that one can make a scheme of the convergence order up to 4 apart from the choice of the function $\varphi$ in (3.11) or (3.12) if one uses the Chebyshev interpolation polynomial of degree $n$ with $n \leq 4$.

## 5. Stability analysis

For the stability analysis of Algorithm 3.5, we will try Dahlquist’s test problem, $y' = \lambda y$.

Let $L = (L_{jk})$, $A = (a_{jk}(\lambda h))$, and $b = (b_1, b_2, \ldots, b_n)^T$ be matrices and vector, whose entries are defined by

$$L_{jk} := \hat{l}_k(s_j), \quad a_{jk}(z) := L_{jk} - \frac{z}{2}\delta_{jk}, \quad b_j := 1 + s_j.$$  

Then, we have the following proposition.

**Proposition 5.1.** Algorithm 3.5 with the discrete Chebyshev collocation system (3.19) applied to $y' = \lambda y$ yields

$$y_{m+1} = S_m(\lambda h)y_m, \quad m \geq 0$$  

with the stability function $S_m(z)$ defined by

$$S_m(z) = 1 + z + \left(\frac{z}{2}\right)^2 \mu_m(z),$$  

where $\mu_m(z)$ is the last component of the vector $(L - \frac{\lambda h}{2}I)^{-1}b$. Here, $I$ denotes the $n \times n$ identity matrix.

**Proof.** For Dahlquist’s test problem $y' = \lambda y$, $f(t, y) = \lambda y$ is a linear function and hence the diagonal matrix $J$ in (3.19) becomes $J = \lambda I$. Further, the vector $f$ in the right-hand side of (3.19) becomes

$$f = \frac{h\lambda^2}{2}y_m b$$

by the definitions of $F$ and Euler’s polygon $y(t)$ in (3.6) and (3.2), respectively. Hence, the discrete Chebyshev collocation system (3.19) applied to Dahlquist’s test problem $y' = \lambda y$ becomes

$$A d = \left(L - \frac{\lambda h}{2}I\right) d = \left(\frac{h\lambda}{2}\right)^2 y_m b$$
with the unknown \( d = [\beta_1, \ldots, \beta_n]^T \). Thus, the formula (3.24) becomes

\[
y_{m+1} = (1 + \lambda h) y_m + \beta_n,
\]

where \( \beta_n \) is the solution of the system (5.3). Therefore, solving (5.3) and inserting the solution into (5.4) leads to (5.1).

Another useful formula for \( S_n(z) \) is the following.

**Proposition 5.2.** The stability function \( S_n(z) \) of (5.2) satisfies

\[
S_n(z) = \frac{\det \begin{pmatrix} L - \frac{z}{2} I & \left( \frac{z}{2} \right)^2 b \\ r & 1 + z \end{pmatrix}}{\det(L - \frac{z}{2} I)},
\]

where \( r = [0, 0, \ldots, 0, -1] \) is the \( 1 \times n \) vector.

**Proof.** The two equations (5.3) and (5.4) can be written as

\[
\begin{pmatrix} L - \frac{z}{2} I & 0 \\ r & 1 \end{pmatrix} \begin{pmatrix} d \\ y_{m+1} \end{pmatrix} = y_m \begin{pmatrix} \left( \frac{z}{2} \right)^2 b \\ 1 + z \end{pmatrix},
\]

where \( z = \lambda h \). Thus, Cramer’s rule implies that the denominator of \( S_n(z) \) is \( \det(L - \frac{z}{2} I) \) and its numerator is

\[
det \begin{pmatrix} L - \frac{z}{2} I & \left( \frac{z}{2} \right)^2 b \\ r & 1 + z \end{pmatrix}.
\]

For the function \( S_n(z) \) defined in (5.2) (or (5.5)), the stability domain of the method is defined as (see [20, p.16])

\[
\Gamma_n := \{ z \in \mathbb{C} : |S_n(z)| < 1 \}.
\]

When the left-half complex plane is contained in \( \Gamma_n \), the method is called \( A \)-stable [20, p. 42]. Also, a method is said to be \( A(\alpha) \)-stable if the sector

\[
\{ z : |\text{arg}(-z)| < \alpha, \ z \neq 0 \}
\]

is contained in the stability region \( \Gamma_n \) [20, p. 45]. Further, a method is called \( L \)-stable [20, p. 44] if it is \( A \)-stable and if, in addition,

\[
\lim_{z \to \infty} S_n(z) = 0.
\]

Using the symbolic calculation with Mathematica, the explicit formula for the stability function \( S_n(z) \) given in (5.5) can be obtained up to the order as possible as the memory of the Mathematica program is allowed. Here, we list the explicit formula and draw the corresponding stability domain \( \Gamma_n \).

The stability functions \( S_n(z) \), \( n = 1, \ldots, 4 \) can be obtained as follows:

\[
S_1(z) = \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}, \quad S_2(z) = \frac{4, 1}{4, -3, 1},
\]

\[
S_3(z) = \frac{(96, 32, 3)}{(96, -64, 19, -3)}, \quad S_4(z) = \frac{(384, 144, 20, 1)}{(384, -240, 68, -11, 1)}
\]
where the notation \((a_0, \ldots, a_k)\) means the polynomial \(\sum_{j=0}^{k} a_j z^j\) with respect to \(z\).

Figure 5.1 shows the stability domains for \(S_n\), \(n = 2, 3, 4\) and it can be observed that \(A\)-stability holds only for \(n = 2\). But, it can be noted that all methods up to the convergence order 4 are \(A(\alpha)\)-stable with a large \(\alpha\), even though the magnitude of \(\alpha\) goes to small when the convergence orders are increasing. Further (5.6) is satisfied for all cases. Thus, one may say that the proposed scheme is almost \(L\)-stable (see [30]).

6. Numerical test. In this section, we test three examples to give numerical evidence for the theoretical results of the proposed method.

Example 1. As the first example, we test the Prothero–Robinson equation, which is a particular case of the family of scalar equations proposed by Prothero and Robinson in [28] and constitutes a stiff problem,

\[ \phi'(t) = \nu (\phi(t) - g(t)) + g'(t), \quad t \in (0, 10]; \quad \phi(0) = 1, \]

where the eigenvalue \(\nu\) is \(\nu = -10^6\) and \(g(t) = \sin(t)\). The exact solution is given by \(\phi(t) = \sin(t)\). In Table 6.1, for different step sizes \(h = 2^{-n}\), the results obtained from the proposed fourth-order method and the fourth-order CCM [30] are compared,
Table 6.1
Results for Example 1 using the fourth-order CCM [30] and the proposed fourth-order method.

<table>
<thead>
<tr>
<th>n</th>
<th>Feval</th>
<th>$Err(h)$</th>
<th>$R(h)$</th>
<th>Feval</th>
<th>$Err(h)$</th>
<th>$R(h)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>40</td>
<td>7.6985 × 10^{-9}</td>
<td></td>
<td>15</td>
<td>7.9576 × 10^{-9}</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>80</td>
<td>5.0844 × 10^{-10}</td>
<td>3.92</td>
<td>30</td>
<td>5.0844 × 10^{-10}</td>
<td>3.97</td>
</tr>
<tr>
<td>1</td>
<td>160</td>
<td>3.0297 × 10^{-11}</td>
<td>4.07</td>
<td>60</td>
<td>3.2457 × 10^{-11}</td>
<td>3.97</td>
</tr>
<tr>
<td>2</td>
<td>320</td>
<td>1.8114 × 10^{-12}</td>
<td>4.06</td>
<td>120</td>
<td>2.0329 × 10^{-12}</td>
<td>4.00</td>
</tr>
<tr>
<td>3</td>
<td>640</td>
<td>1.1013 × 10^{-13}</td>
<td>4.04</td>
<td>240</td>
<td>1.2708 × 10^{-13}</td>
<td>4.00</td>
</tr>
<tr>
<td>4</td>
<td>1280</td>
<td>6.7723 × 10^{-15}</td>
<td>4.02</td>
<td>480</td>
<td>1.2708 × 10^{-15}</td>
<td>4.00</td>
</tr>
</tbody>
</table>

Table 6.2
Results for Example 2 with $\kappa = 1$ using three methods, JECEM, ECEM, and CCM [30].

<table>
<thead>
<tr>
<th>n</th>
<th>FEval</th>
<th>$Err(h)$</th>
<th>$R(h)$</th>
<th>FEval</th>
<th>$Err(h)$</th>
<th>$R(h)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>60</td>
<td>1.62 × 10^{-4}</td>
<td></td>
<td>240</td>
<td>5.63 × 10^{-6}</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>120</td>
<td>9.20 × 10^{-6}</td>
<td>4.14</td>
<td>4.62</td>
<td>4.77 × 10^{-7}</td>
<td>3.56</td>
</tr>
<tr>
<td>3</td>
<td>240</td>
<td>5.52 × 10^{-7}</td>
<td>4.06</td>
<td>4.73</td>
<td>3.58 × 10^{-8}</td>
<td>3.74</td>
</tr>
<tr>
<td>4</td>
<td>480</td>
<td>3.29 × 10^{-8}</td>
<td>4.07</td>
<td>4.58</td>
<td>2.39 × 10^{-9}</td>
<td>3.90</td>
</tr>
<tr>
<td>5</td>
<td>960</td>
<td>2.00 × 10^{-9}</td>
<td>4.04</td>
<td>4.40</td>
<td>1.54 × 10^{-10}</td>
<td>3.95</td>
</tr>
</tbody>
</table>

where the column $Err(h)$ shows the maximum of the absolute error over the integration interval given by $Err(h) = \max_{j} \{|\phi_{\text{exact}}(t_j) - \phi_{\text{computed}}(t_j)|\}$. In the same table, “Feval” expresses the number of evaluations of the vector $f$ at step 6 of Algorithm 3.5 for the function in the right-hand side of (3.1) and the column $R(h)$ means the convergence order defined by $R(h) = \log(Err(h)/Err(h/2))/\log 2$. The results show that both CCM [30] and the proposed method have similar absolute errors. These results also reveal that the convergence order of the proposed method is 4 as expected by our theoretical analysis. However, with respect to function evaluations, the present method is superior to CCM [30] because of its explicit representation.

Example 2. Consider the nonlinear initial value problem in [1]

$$\frac{d\phi}{dt} = \frac{\kappa \phi(t)(1 - \phi(t))}{2\phi(t) - 1}, \quad (0, 10]; \quad \phi(0) = \frac{5}{6},$$

whose solution is $\phi(t) = \frac{1}{2} + \sqrt{\frac{1}{4} - \frac{5}{30}e^{-\kappa t}}$ with a parameter $\kappa$. To investigate the effects of the usage of the partial derivative $f_\phi$, we denote the method using $f_\phi$ by JECEM and the method using the forward difference approximation with $\lambda = 3$ in (3.12) for $f_\phi$ by ECEM. For small parameter $k = 1$, we solved the problem by the fourth-order schemes for various step sizes $h = 2^{-n}$, the results of three methods JECEM, ECEM, and CCM [30] are compared in Table 6.2. The numerical results, as expected in the theoretical analysis, show that both JECEM and ECEM have the same convergence order 4 with the same cost for the function evaluations. Even though CCM in [30] comparing to JECEM and ECEM has the similar convergence order 4 and a better quality of the maximum error, its cost for the function evaluations are increasing rapidly as the step size $h$ is decreasing. In this point of view, one may conclude that both JECEM and ECEM are superior to CCM. Also, if the use of partial derivative $f_\phi$ is quite costly, one may say that ECEM is more superior to the other two methods. For the quite large parameter $k = 50$, we solved the problem by two third-order methods ECEM and CCM. The numerical results in Table 6.3 show.
that quite less CPU time of ECEM compared to CCM is required to get a similar maximum error.

**Example 3.** Consider the stiff system of initial value problems taken from [31]

\[
\begin{align*}
\phi_1'(t) &= -1002\phi_1(t) + 1000\phi_2(t), \quad \phi_1(0) = 1, \\
\phi_2'(t) &= \phi_1(t) - \phi_2(t)(1 + \phi_2(t)), \quad \phi_2(0) = 1,
\end{align*}
\]

whose solution is \(\phi_1(t) = e^{-2t}\) and \(\phi_2(t) = e^{-t}\). The problem is solved with two third-order methods ECEM and CCM. According to Table 6.4, ECEM requires meaningfully less CPU time than CCM to get similar maximum errors.

**7. Conclusion.** Error correction methods for solving stiff initial value problems are developed using a Chebyshev interpolation polynomial. These methods are involved by employing either the partial derivative \(f_\phi\) or its forward difference approximation. Also, a unified convergence analysis is provided for both methods. The stability properties are analyzed and it is shown that the proposed methods are almost L-stable. It is remarkable phenomena that the good stability and convergence are obtained even though the scheme is a type of explicit method and is not a required evaluation of the partial derivatives. But one may note that the solution of the linear system (3.19) requires a similar cost to a Newton iteration in a standard implicit Runge–Kutta method and the computations in (3.9) can be regarded as counterparts of computations which occur in traditional implementations of implicit methods. Thus, we will discuss the reduction of computational costs under the same accuracy and stability in a forthcoming paper to employ the full advantage aspects of explicitness in (3.9). For example, one may consider another extension based on Gauss–Seidel iteration technique as follows: for \(t \in [t_m, t_{m+1}]\), consider the componentwise formula for \(i = 1, \ldots, d\)

\[
\frac{d\phi^{(i)}_j}{dt} = f_i(t, \phi^{(i)}_1(t), \ldots, \phi^{(i)}_{j-1}(t), \phi^{(i)}_j(t), \phi^{(i+1)}_j(t), \ldots, \phi^{(i-1)}_d(t)),
\]

\[
(7.1)
\]
where $\phi_i^{(j)}(t)$ denotes the $j$th iteration for $\phi_i(t)$ and one can choose the initial guess $\phi_i^{(0)}(t)$, $i = 2, \ldots, d$ by the Euler polygon

$$\phi_i^{(0)} = Y_{mi} + (t - t_m)f_i(t_m, Y_m), \quad t \in [t_m, t_{m+1}].$$

Here, $Y_{mi}$ denotes the $i$th component of the vector $Y_m$. Note that each componentwise formula (7.1) can be regarded with the scalar equation as we discussed in this paper. Hence, the proposed scheme can be applied directly to find an approximation for the $j$th iteration $\phi_i^{(j)}(t)$ for each $i = 1, \ldots, d$.

In a forthcoming work, we will also deal with these topics with applications to several time dependent partial differential equations.

**Acknowledgment.** The authors would like to thank the anonymous referees who suggested many helpful comments and corrections which improved this paper.

**REFERENCES**


