Development of Variable Stepsize Variable Order Block Method in Divided Difference Form for the Numerical Solution of Delay Differential Equations

Fuziyah Ishak, Mohamed B. Suleiman, Zanariah A. Majid and Khairil I. Othman

Abstract—This paper considers the development of a two-point predictor-corrector block method for solving delay differential equations. The formulae are represented in divided difference form and the algorithm is implemented in variable stepsize variable order technique. The block method produces two new values at a single integration step. Numerical results are compared with existing methods and it is evident that the block method performs very well. Stability regions of the block method are also investigated.

Keywords—block method, delay differential equations, predictor-corrector, stability region, variable stepsize variable order.

I. INTRODUCTION

In this paper we consider the development of a numerical method for solving systems of first order delay differential equations (DDEs) of the form:

\[
\begin{align*}
    y'(x) &= f(x, y(x), y(x - \tau_i)), \quad x \in [a, b], \tau_i > 0, \\
    y(x) &= \phi(x), \quad x \in [\xi, a],
\end{align*}
\]

(1)

where \(\phi(x)\) is the initial function and \(\tau_i\), for \(i = 1, 2, \ldots, n\), where \(n\) is an integer such that \(n \geq 1\) is the lag function and \(\xi = \min (x - \tau_i)\). The lag can be constant, time dependent where \(\tau_i = \tau(x)\) or state dependent, that is \(\tau_i = \tau_i(x, y(x))\). The expression \(y(x - \tau_i)\) is the solution of the delay term, or simply called the delay term. The function \(f\) where \(f : [a, b] \times C([\xi, \tau], R^n) \times C([\xi, \tau], R^n) \rightarrow R^n\) is continuous and satisfies a Lipschitz condition which guarantees the existence of a unique solution of (1). Here, \(C([\xi, \tau], R^n)\) denotes the space of continuous functions mapping \([\xi, \tau]\) into \(R^n\) for an integer \(m \geq 1\).

Most numerical methods for solving DDEs are adapted from that of numerical methods for ordinary differential equations (ODEs). For some of the earlier work, please refer to [1]-[6]. These methods approximate only one new solution at each integration step. Block methods, however, produce more than one approximate solution at every step. Greater efficiency is achieved because the total number of steps taken can be reduced. Block methods are also the natural candidates for parallel computations because these solutions can be computed simultaneously at every step. Block methods for solving ODEs have been proposed by [7]-[10].

The purpose of this paper is to develop a variable stepsize variable order (VSVO) two-point block method for solving (1). The method produces two new solutions at every step and varies the stepsize and order while achieving the desired accuracy as efficiently as possible. In variable stepsize, the integration coefficients need to be recalculated when the stepsize changes. The recalculation cost can be minimize by representing the formulae in divided difference form because the coefficients are calculated using a simple recursive formula. The performance of the method will be compared with the existing methods SDDDELM in [5] and S2PBBI in [11]. SDDDELM is an all purpose non-block VSVO algorithm where the formulae are represented in modified divided difference form. S2PBBI is a predictor-corrector two-point VSVO block method in divided difference form. The order of the corrector for S2PBBI differs from the order of the method developed here.

The organization of this paper is as follows. In section II, we describe the development of the VSVO predictor-corrector block method. Absolute stability for the two-point predictor-corrector block method is discussed in section III. Numerical results are presented in section IV. Finally section V is the conclusion.

II. METHOD DEVELOPMENT

For simplicity, we consider only single-delay scalar equation of (1). However, the results can easily be extended to systems of equations with multiple delays.

From (1), we let \(i = 1\) and \(\tau_i = \tau\). Consider a non-uniform grid given by \(\bar{\alpha} = a = x_0 < \cdots < x_n < x_{n+1} < \cdots \leq b\). We denote the approximation to \(y(x)\), the solution of (1) by \(y_h(x)\). The predicted value of \(y_h(x)\) is denoted as \(p_h(x)\). We also denote the expression \(f(x_n, y_h(x_n), y_h(\alpha))\), \(\alpha = x_n - \tau\) by \(f_n\). Assume that \(y_h(x)\) has been computed for \(x \in [\bar{\alpha}, x_n]\).

The immediate task is to evaluate \(y_h(x_{n+1}), y_h(x_{n+2})\), and their corresponding delay terms. The formulae are
implemented in PECE mode where P stands for an application of a predictor, \( E \) stands for an evaluation of a function \( f \), and \( C \) stands for an application of a corrector.

Let \( P_{k,n} \) be the interpolating polynomial of degree \( k-1 \) interpolating \( f \) at points
\[
(x_0, f_0), (x_1, f_1), \ldots, (x_{n-k+1}, f_{n-k+1}).
\]
In divided difference form, the interpolating polynomial can be written as
\[
P_{k,n}(x) = f[x_n] + \sum_{i=0}^{k-1} f[x_{n-i,}] \prod_{j=0}^{i-1} (x-x_{n-j}),
\]
where
\[
f[x_{n-i,}] = \frac{f[x_0, x_1, \ldots, x_{n-i,}]}{x_n-x_{n-i+1}} - f[x_{n-i,}, x_{n-i+1,}, \ldots, x_{n-k+1}].
\]

Integrating (1) from \( x_0 \) to \( x_{n+d} \), \( d=1,2 \) and replacing \( f \) with \( P_{k,n} \) yields the following predicted values simultaneously at the grid points,
\[
p_h(x_{n+d}) = y_h(x_n) + \sum_{i=0}^{k-1} c_{n,i}(\xi)d \xi,
\]
where \( \xi \) is an independent dummy variable and
\[
c_{n,i}(x) = \left\{ \begin{array}{ll}
1, & i = 0, \\
(x-x_n)(x-x_{n-1})\cdots(x-x_{n-i+1}), & i \geq 1.
\end{array} \right.
\]
The predicted values of the derivative are given by
\[
p_h^i(x_{n+d}) = \sum_{i=0}^{k-1} c_{n,i}(x)
\]
Let the \( t \)-fold integral of \( c_{n,i}(x) \) be denoted as \( c_{n,i}^{(t)}(x) \). Now define \( g_{i,j}(x) \) as follows,
\[
g_{i,j}(x) = \left\{ \begin{array}{ll}
c_{n,i}(x), & t = 0, \\
c_{n,i}^{(t)}(x), & t \geq 1.
\end{array} \right.
\]
The formula for \( g_{i,j}(x) \) where \( t \geq 1, i \geq 1 \) can be obtained recursively by the relation
\[
g_{i,j}(x) = (x-x_{n-i+1})g_{i,j-1}(x) - g_{i-1,j}(x).
\]
The predicted values at the grid points are given by
\[
p_h(x_{n+d}) = y_h(x_n) + \sum_{i=0}^{k-1} g_{i,1}(x_{n+d})f[x_{n-i,}, x_{n-i+1,}, \ldots, x_{n-k+1}].
\]
and
\[
p_h^i(x_{n+d}) = \sum_{i=0}^{k-1} g_{i,0}(x_{n+d})f[x_{n-i,}, x_{n-i+1,}, \ldots, x_{n-k+1}].
\]
We now describe the corrector for the first point. Consider the interpolating polynomial \( P_{k+1,n+1} \) of degree \( k \) that interpolates \( f \) at points
\[
(x_{n+1,}, f_{n+1}^p), (x_n, f_n), (x_{n-1,}, f_{n-1}^p), \ldots, (x_{n-k+1,}, f_{n-k+1}^p),
\]
where
\[
f_{n+1}^p = f(x_{n+1}, p_h(x_{n+1}), p_h(x_{n+1}-\tau)).
\]
Clearly the interpolating polynomial \( P_{k+1,n+1} \) can be written as
\[
P_{k+1,n+1}(x) = P_{k,n}(x) + c_{k,1}(x)f^p[x_{n+1}, x_n, x_{n-1}, \ldots, x_{n-k+1}].
\]
The notation \( f^p[x_{n+1}, x_n, x_{n-1}, \ldots, x_{n-k+1}] \) is referred to the divided difference that uses \( f^p \) for the points described above. Replacing \( f \) in (1) with the polynomial \( P_{k+1,n+1} \) and integrating from \( x_n \) to \( x_{n+1} \), the first point corrector formula is obtained as,
\[
y_h(x_{n+1}) = p_h(x_{n+1}) + g_{k,1}(x_{n+1})f^p[x_{n+1}, x_n, x_{n-1}, \ldots, x_{n-k+1}].
\]
The corrected value of the derivative is given by
\[
y'_{n+1}(x_{n+1}) = y_h'(x_{n+1}) = p_h'(x_{n+1}) + g_{k,0}(x_{n+1})f^p[x_{n+1}, x_n, x_{n-1}, \ldots, x_{n-k+1}].
\]
For the second point corrector, consider the interpolating polynomial \( P_{k+2,n+2} \) of degree \( k+1 \) that interpolates \( f \) at points
\[
(x_{n+2,}, f_{n+2}^p), (x_{n+1,}, f_{n+1}^p), (x_n, f_n), (x_{n-1,}, f_{n-1}^p), \ldots, (x_{n-k+1,}, f_{n-k+1}^p),
\]
where
\[
f_{n+2}^p = f(x_{n+2}, p_h(x_{n+2}), p_h(x_{n+2}-\tau)) \quad \text{and} \quad f_{n+1}^p = f(x_{n+1}, p_h(x_{n+1}), p_h(x_{n+1}-\tau)).
\]
The interpolating polynomial \( P_{k+2,n+2} \) can be written as
\[
P_{k+2,n+2}(x) = P_{k,n}(x) + c_{k,1}(x)f^p[x_{n+2}, x_{n+1}, x_n, x_{n-1}, \ldots, x_{n-k+1}]
\]
\[+ c_{k+1,1}(x)f^p[x_{n+1}, x_n, x_{n-1}, \ldots, x_{n-k+1}].
\]
The notation \( f^p[x_{n+2}, x_{n+1}, x_n, x_{n-1}, \ldots, x_{n-k+1}] \) is referred to the divided difference with the interpolations points \( f_{n+1}^p \) and \( f_{n+2}^p \). Replacing \( f \) in (1) with the polynomial \( P_{k+2,n+2} \) and integrating from \( x_n \) to \( x_{n+2} \), the formulae for the second point corrector are obtained as
\[
y_h(x_{n+2}) = p_h(x_{n+2}) + g_{k,1}(x_{n+2})f^p[x_{n+2}, x_{n+1}, x_n, x_{n-1}, \ldots, x_{n-k+1}]
\]
\[+ (h_{k,1}(x_{n+2}) - g_{k,2}(x_{n+2}))f^p[x_{n+1}, x_n, x_{n-1}, \ldots, x_{n-k+1}],
\]
and
\[
y'_{n+2}(x_{n+2}) = y_h'(x_{n+2}) + g_{k,0}(x_{n+2})f^p[x_{n+2}, x_{n+1}, x_n, x_{n-1}, \ldots, x_{n-k+1}]
\]
\[+ h_{k,0}(x_{n+2})f^p[x_{n+1}, x_n, x_{n-1}, \ldots, x_{n-k+1}].
\]
For computational purposes, we let \( e_d \), \( d=1,2 \) be defined as
\[
ed = y_h'(x_{n+d}) - p_h'(x_{n+d}).
\]
Thus
\[
f^p[x_{n+1}, x_n, x_{n-1}, \ldots, x_{n-k+1}] = \frac{e_d}{g_{k,0}(x_{n+1})},
\]
\[ f^P[x_{n+2}, x_{n+1}, x_n, x_{n-1}, \ldots, x_{n-k+1}] = \frac{1}{h} \left( e_2 \frac{g_{k,0}(x_{n+2})}{g_{k,0}(x_{n+1})} - e_1 \right) \]

Therefore, the corrector formulae for the first point can be written as

\[
y_h(x_{n+1}) = p_h(x_{n+1}) + \frac{g_{k,0}(x_{n+1})}{g_{k,0}(x_{n+1})} e_1,
\]

\[
y'_h(x_{n+1}) = p'_h(x_{n+1}) + e_1.
\]

The corrector formulae for the second point can be written as

\[
y_h(x_{n+2}) = p_h(x_{n+2}) + \frac{g_{k,2}(x_{n+2})}{g_{k,0}(x_{n+1})} e_1
\]

\[
+ \left( \frac{g_{k,1}(x_{n+1}) - g_{k,2}(x_{n+2})}{g_{k,0}(x_{n+1})} \right) e_2,
\]

\[
y'_h(x_{n+2}) = p'_h(x_{n+2}) + e_2.
\]

Approximation formulae for the delay terms \( y_h(x_{n+1} - \tau) \) and \( y_h(x_{n+2} - \tau) \) are derived following the earlier discussions for formulae at grid points. For \( d = 1,2 \), let \( \alpha_d = x_{n+d} - \tau \) be the delay argument. If \( \alpha_d < a \), the predictor and the corrector are evaluated by using the initial function, that is, \( p_h(\alpha_d) = y_h(\alpha_d) = \phi(\alpha_d) \). If \( a < \alpha_d \leq x_n \), we determine the exact location of \( \alpha_d \) by finding the value of \( j \) such that \( x_j < \alpha_d \leq x_{j+1} \). The predictor is given by

\[
p_h(\alpha_d) = y_h(x_j) + \sum_{i=0}^{k-1} g_{i,j}(\alpha_d) f[x_j, x_{j-1}, \ldots, x_j - j],
\]

and the corrector is given by

\[
y_h(\alpha_d) = p_h(\alpha_d) + g_{k,j}(\alpha_d) f^P[x_j, x_j-1, \ldots, x_j - j+1].
\]

If \( x_n < \alpha_d \leq x_{n+1} \), the predictor is given by

\[
p_h(\alpha_d) = y_h(x_n) + \sum_{i=0}^{k-1} g_{i,0}(\alpha_d) f[x_n, x_{n-1}, \ldots, x_n - i],
\]

and the corrector is given by

\[
y_h(\alpha_d) = p_h(\alpha_d) + g_{k,0}(\alpha_d) e_1.
\]

If \( x_{n+1} < \alpha_d \leq x_{n+2} \),

\[
y_h(\alpha_d) = p_h(\alpha_d) + \frac{g_{k,1}(\alpha_d)}{g_{k,0}(x_{n+1})} e_1
\]

\[
+ \left( \frac{(\alpha_d - x_{n+1})g_{k,1}(\alpha_d) - g_{k,2}(\alpha_d)}{g_{k,0}(x_{n+1})} \right) e_2
\]

\[
\times \frac{e_2}{g_{k,0}(x_{n+2})} - \frac{e_1}{g_{k,0}(x_{n+1})}.
\]

Local errors at the grid points are estimated by comparing the formulae of different orders. The error magnitude of the first point is presumably higher than that of the second point because the approximation at the first point is one order less than the approximation at the second point. By controlling the error at the first point, the method produces better result. We simply refer to the estimated local error at the first point as the local error. From [11], the local error when applying the formulae of order \( k \), denoted by \( E_k \) is given by

\[
E_k = \frac{\text{error}}{g_{k,0}(x_{n+1})} e_1.
\]

We also calculate local errors for formulae of orders \( k - 2, k - 1, k + 1 \) because these errors will facilitate in the stepsize and order changing strategy. Local errors of adjacent orders \( E_{k-2}, E_{k-1}, \) and \( E_{k+1} \) can be calculated directly from (2) by exchanging \( k \) with \( k - 2, k - 1, \) and \( k + 1 \) respectively.

This VSVO algorithm is self-starting. A method of order one with a suitable initial stepsize starts the integration. The algorithm varies the order while taking the most optimal stepsize to achieve the desired accuracy as efficiently as possible.

A step is accepted if

\[
E_k < \text{TOL} \times (A + B \times p_h(x_{n+1})),
\]

where \( \text{TOL} \) is the user specified tolerance and the values of \( A \) and \( B \) are either 1 or 0 depending upon whether we use relative, absolute or mixed error test. Whenever a step fails, the new stepsize is reduced by half. New order is selected before the integration step is repeated. When a step is accepted, we select a new order \( k_{\text{new}} \) based on the following strategy. The order is reduced by one, that is, \( k_{\text{new}} = k - 1 \) if we encounter the following criteria,

- for \( k > 2 \), \( \max[F_{k-2}, F_{k-1}] \leq E_k \),
- for \( k = 2 \), \( |E_{k-1}| \leq 0.5|E_k| \),
- if \( |E_{k+1}| \) is available, \( k > 1 \) and \( |E_{k+1}| \leq \min(|E_k|, |E_{k-1}|) \).

We consider raising the order by one only after \( k + 1 \) successful steps at constant stepsize. The new order \( k_{\text{new}} = k + 1 \) if

- for \( k = 1 \), \( |E_{k+1}| < 0.5|E_k| \),
- for \( k > 1 \), \( |E_{k+1}| < |E_k| \leq |E_{k-1}| \).

If none of the above criteria is satisfied, \( k_{\text{new}} = k \).

Once the new order is selected, it is renamed \( k \) and the corresponding \( E_k \) is used in selecting the new stepsize. For a successful step, the new stepsize is given by

\[
h_{\text{new}} = \begin{cases} 2h, & \text{if } \hat{R} \geq 2, \\ \hat{R}h, & \text{if } 1.6 < \hat{R} < 2 \text{ or } 0.5 \leq \hat{R} < 0.9, \\ 0.5h, & \text{if } \hat{R} < 0.5, \\ h, & \text{otherwise}, \end{cases}
\]

where

\[
R = 0.8R, \quad \theta_{n+1} = \left( \text{TOL} \frac{1}{|E_k|} \right)^{1/k_{\text{new}}}
\]

and

\[
\frac{1}{|E_k|} = A + B \times |y_h(x_{n+1})|,
\]
We multiply $R$ with a safety factor of 0.8 in order to give a more conservative estimate for the new stepsize. This practice reduces the number of steps being rejected.

Derivative discontinuities in the solutions of DDEs can be detected whenever the magnitude of the local error is too big as compared with a given tolerance. The algorithm reduces the stepsize and order of the method accordingly in order to reduce the magnitude of local error. The process is repeated until the discontinuity point is included in the grid.

III. ABSOLUTE STABILITY

It is of practical importance to study the behavior of the global error where the numerical solution is expected to behave as the exact solution does when the independent variable $x$ approaches infinity. Many authors including [12]-[15] studied various concepts of stability based on test equations with known stability regions. Common test equations are

$$y'(x) = Ay(x) + \mu y(x-r), \quad x \geq x_0, \quad (3)$$

$$y(x) = \phi(x), \quad -\tau \leq x \leq x_0,$$

and

$$y'(x) = \mu y(x-r), \quad x \geq x_0,$$

$$y(x) = \phi(x), \quad -\tau \leq x \leq x_0,$$

where $A$ and $\mu$ are complex numbers.

We refer to [14] for the definitions of P- and Q-stability regions. Let $h$ be a fixed stepsize such that $x_n = x_0 + nh$ and $mh = r$, $m \in \mathbb{N}$. Let $H_1 = hA$ and $H_2 = h\mu$.

Definition 1: For a fixed stepsize $h$ and $A, \mu \in R$ in (4), the region $R_p$ in the $H_1 - H_2$ plane is called the P-stability region if for any $(H_1, H_2) \in R_p$, the numerical solution of (3) vanishes as $x_n \to \infty$.

Definition 2: For a fixed stepsize $h$, $\mu \in C$ in (4), the region $R_q$ in the complex $H_2$-plane is called the Q-stability region if for any $H_2 \in R_Q$, the numerical solution of (4) vanishes as $x_n \to \infty$.

For notational purposes, let $n = 2N$. We define the matrices $Y_n, Y_0, F_n$ and $F_0$ as the following,

$$Y_n^p = [y_{n+1}, y_{n+2}, \ldots, y_{n+k-1}, y_{n+k-2}, y_{n+k-1}, y_{n+k}]^T,$$

$$Y_n^c = [y_{n+1}, y_{n+2}, \ldots, y_{n+k-1}, y_{n+k-2}, y_{n+k-1}, y_{n+k}]^T,$$

$$F_n^p = [f_{n+1}, f_{n+2}, \ldots, f_{n+k-1}, f_{n+k-2}, f_{n+k-1}, f_{n+k}]^T,$$

$$F_n^c = [f_{n+1}, f_{n+2}, \ldots, f_{n+k-1}, f_{n+k-2}, f_{n+k-1}, f_{n+k}]^T.$$

For simplicity, we denote the predicted value of $y_h(x_{n+k})$ by $y_{n+k}^p$ and the corrected value by $y_{n+k}^c$. The predictor-corrector formulae for the two-point block method are given by

$$Y_{n+1}^p = CY_n^c + hDF_n^c,$$

$$Y_{n+1}^c = SY_n^p + hTF_n^c.$$  

Here, $C, D, S$ and $T$ are $(k+2) \times (k+2)$ square matrices given by,

$$C = \begin{bmatrix}
0 & 0 & I_{k+1}
\vdots & \vdots & \vdots 
0 & 0 & 0 & 0 & 1
\end{bmatrix},$$

$$D = \begin{bmatrix}
0 & \cdots & \cdots & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots 
0 & d_{10} & d_{11} & \cdots & d_{1k-2} & d_{1k-1} \\
0 & d_{20} & d_{21} & \cdots & d_{2k-2} & d_{2k-1}
\end{bmatrix},$$

$$S = \begin{bmatrix}
I_{k+1} & \cdots & \cdots & \cdots & 0 \\
0 & \cdots & 0 & 1 & 0 \\
0 & \cdots & 0 & 1 & 0
\end{bmatrix},$$

$$T = \begin{bmatrix}
0 & \cdots & 0 \\
\vdots & \vdots & \vdots 
I_{10} & t_{11} & \cdots & t_{1k-1} & t_{1k+1} \\
I_{20} & t_{21} & \cdots & t_{2k-1} & t_{2k+1}
\end{bmatrix},$$

where $I_{k+1}$ is the $k \times k$ identity matrix and $d_{ij}$ and $t_{ij}$ are the integration coefficients. The application of (5) to (3) yields

$$Y_{n+1}^p = (C + H_1)Y_n^c + H_2DY_n^c,$$

$$Y_{n+1}^c = (S + H_1T)Y_n^p + H_2TY_n^p + C.$$  

Consequently, the P-stability polynomial of the two-point predictor-corrector block method, $\pi_{2PECE,m}(H_1, H_2; \zeta)$ is given by

$$\det[I_{2(k+1)}^2 - ((S + H_1T)(C + H_1D)) \zeta^{2m} - ((H_2S + H_2T)(C + H_1D)) \zeta^{m} - H_2^2TD],$$

and the Q-stability polynomial, $\psi_{2PECE,m}(H_1; \zeta)$ is given by

$$\det[I_{2(k+1)}^2 - S \zeta^{2m} - H_2(SD + TC) \zeta^{m} - H_2^2TD],$$

where $\det[\cdot]$ is the determinant of a square matrix.

The P- and Q-stability regions are obtained by using boundary locus technique. Inside the regions of absolute stability, the magnitudes of all the roots of the P- and Q-stability polynomials are less than one. As an illustration, the P- and Q-stability regions are shown in Fig. 1 and Fig. 2 respectively. The Q-stability regions are symmetric about the real axis and Fig.2 represents the regions in the plane where $\Im(H_2) > 0$. 

IV. NUMERICAL RESULTS

We test the performance of the block method on various examples from [1]. The performance of the block method is compared with the methods S2PBDI from [11] and SNDDELM from [5]. The examples used are as follows,

Example 1:

\[ y'(x) = \cos(x)y(y(x) - 2), \quad 0 \leq x \leq 50, \]
\[ y(x) = 1, \quad x \leq 0. \]

The exact solution is \( y(x) = \sin(x) + 1 \).

Example 2:

\[ y'(x) = -y(x - 1 + \exp(-x)) + \sin(x - 1 + \exp(-x)) + \cos(x), \quad 0 \leq x \leq 10, \]
\[ y(x) = \sin(x), \quad x \leq 0. \]

The exact solution is \( y(x) = \sin(x) \).

Example 3:

\[ y'_1(x) = -y_1 \left( x - \frac{\pi}{2} \right), \quad \frac{\pi}{2} \leq x \leq 10, \]
\[ y'_2(x) = -y_2 \left( x - \frac{\pi}{2} \right), \quad \frac{\pi}{2} \leq x \leq 10, \]
\[ y_1(x) = \sin(x), \quad x \leq \frac{\pi}{2}, \]
\[ y_2(x) = \cos(x), \quad x \leq \frac{\pi}{2}. \]

The exact solution is \( y_1(x) = \sin(x) \) and \( y_2(x) = \cos(x) \).

Example 4:

\[ y'(x) = 1 - y \left( \exp \left( 1 - \frac{1}{x} \right) \right), \quad 2 \leq x \leq 100, \]
\[ y(x) = \ln(x), \quad 0 \leq x \leq 2. \]

The exact solution is \( y(x) = \ln(x), x > 0 \).

The error \( err_i \) at the grid point for each component is defined as

\[ (err_i)_t = \frac{\left( y_h(x_i) \right)_t - \left( y(x_i) \right)_t}{A + B_i(x_i)_t}, \]

where \( (y)_t \) is the \( t \)-th component of \( y \) and \( y(x)_t \) is the exact solution at \( x_t \). \( A \) and \( B \) may take the values of either 1 or 0 depending upon the kind of error test chosen. In this case, we used mixed error test where \( A = 1 \) and \( B = 1 \) as opposed to absolute error test (\( A = 1 \) and \( B = 0 \)) and relative error test (\( A = 0 \) and \( B = 1 \)). The maximum error is defined as follows,

\[ \text{MAXE} = \max_{i \leq \text{STEP} \leq N} \left( \max_i (err_i)_t \right), \]

where \( N \) is the number of equations in the system and \( \text{STEP} \) is the total number of successful steps. The experiment was executed on Sun Fire V1280 by using one processor. The machine is located at the Institute for Mathematical Research (INSPEM), Universiti Putra Malaysia.

The numerical results are given in Table I – Table IV. The following abbreviations are used to describe the numerical solutions:

- \( \text{TOL} \) the chosen tolerance,
- \( \text{MTD} \) method employed,
- \( \text{STP} \) total number of steps,
- \( \text{FS} \) total number of failed steps,
- \( \text{MAXE} \) maximum error,
- \( \text{TIME} \) time taken in microseconds,
- \( \text{S2PBTI} \) the algorithm described in the paper,
- \( \text{S2PBDI} \) the algorithm in [11],
- \( \text{SNDDELM} \) the algorithm in [5],

2.47518E-02 is equivalent to 0.000247518.

<table>
<thead>
<tr>
<th>TOL</th>
<th>MTD</th>
<th>STP</th>
<th>FS</th>
<th>MAXE</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>10^{-2}</td>
<td>S2PBTI</td>
<td>55</td>
<td>7</td>
<td>4.86496E-01</td>
<td>1725</td>
</tr>
<tr>
<td>10^{-2}</td>
<td>S2PBDI</td>
<td>62</td>
<td>4</td>
<td>2.54222E-02</td>
<td>2858</td>
</tr>
<tr>
<td>10^{-2}</td>
<td>SNDDELM</td>
<td>81</td>
<td>3</td>
<td>2.44098E-03</td>
<td>9026</td>
</tr>
<tr>
<td>10^{-4}</td>
<td>S2PBTI</td>
<td>76</td>
<td>5</td>
<td>1.77338E-04</td>
<td>2374</td>
</tr>
<tr>
<td>10^{-4}</td>
<td>S2PBDI</td>
<td>88</td>
<td>3</td>
<td>1.81656E-04</td>
<td>4017</td>
</tr>
<tr>
<td>10^{-4}</td>
<td>SNDDELM</td>
<td>120</td>
<td>6</td>
<td>3.10620E-05</td>
<td>2374</td>
</tr>
<tr>
<td>10^{-6}</td>
<td>S2PBTI</td>
<td>126</td>
<td>5</td>
<td>4.18052E-06</td>
<td>5588</td>
</tr>
<tr>
<td>10^{-6}</td>
<td>S2PBDI</td>
<td>136</td>
<td>3</td>
<td>5.12003E-06</td>
<td>16118</td>
</tr>
<tr>
<td>10^{-6}</td>
<td>SNDDELM</td>
<td>179</td>
<td>13</td>
<td>5.12003E-06</td>
<td>16118</td>
</tr>
<tr>
<td>10^{-10}</td>
<td>S2PBTI</td>
<td>171</td>
<td>1</td>
<td>1.08162E-11</td>
<td>9841</td>
</tr>
<tr>
<td>10^{-10}</td>
<td>S2PBDI</td>
<td>214</td>
<td>7</td>
<td>5.99198E-11</td>
<td>10923</td>
</tr>
<tr>
<td>10^{-10}</td>
<td>SNDDELM</td>
<td>282</td>
<td>7</td>
<td>6.55478E-09</td>
<td>35516</td>
</tr>
</tbody>
</table>
From the numerical results, it is evident that S2PBTI achieves the desired accuracy and performs very well when compared to S2PBDI and SNDDELM. Using corrector of one order higher than S2PBDI results in reduction number of total steps and time taken. The predictor-corrector formulae in divided difference form as opposed to the modified divided difference form in SNDDELM do not reduce the performance of the method. Overall results show that total steps taken in S2PBTI are reduced. The simpler formulation together with the reduction in total steps reduces the integration time when compared with S2PBDI and SNDDELM.

V. CONCLUSION

We have described the development of a two-point predictor-corrector block method for solving DDEs. The formulae for the VSVO algorithm in PECE mode are represented in divided difference form. The corrector for the second point uses one extra point resulting from the prediction process. The P- and Q-stability polynomials of the predictor-corrector block method are determined and the stability regions indicate that the method is suitable for solving nonstiff DDEs. From the numerical results we conclude that the two-point block method is efficient, accurate and reliable for solving DDEs.

REFERENCES