Efficient interpolators in implicit block method for solving delay differential equations

Fuziyah Ishak, Zanariah A. Majid, and Mohamed B. Suleiman

***Abstract*—**In this paper, the development of an implicit block method with variable stepsize variable order technique is described. The grid-point formulae for different number of blocks are derived. The block method produces two new approximations in a single integration step by using the same back values. In order to vary the stepsize and order as efficiently as possible, the coefficients of the method at grid points are calculated and stored in the program. Delay solutions are approximated by using Lagrange and Hermite interpolations. These interpolation techniques prove to be both efficient and reliable with the two-point implicit block method in solving a wide range of delay differential equations.

***Keywords*—** Block method, delay differential equations, interpolation technique, variable order, variable stepsize.

I. Introduction

DELAY differential equations (DDEs) have been used in modeling many real life phenomena, see for examples [1]-[4]. Among the phenomena described in these literatures are mixing problems, population models, economics models and biological models. At times when analytical solutions for functional differential equations are hard and almost impossible to find, scientists and engineers resort to numerical solutions that can be made as accurately as possible, [5]-[7]. Conventionally, numerical solutions for DDEs are adapted from the existing numerical solutions for ordinary differential equations (ODEs). In [8]-[10], among the popular methods are Runge-Kutta type of methods and multistep methods. All of these methods produce only one approximate solution in an integration step. Another approach that has gained interest recently is block methods. Block methods produce more than one approximate solution in a step, [11] and [12]. Greater efficiency is obtained since total number of steps taken can be reduced. The main differences between ODEs and DDEs are the existence of initial functions and the presence of delay terms. Initial function provides history of prior time for the solution of the derivative. Since the derivative of the unknown function also depends on the solution at prior time, numerical methods that provide discrete solutions at the grid points are not suitable for solving DDEs. Approximations at non-grid points are computed using appropriate interpolation techniques so that the accuracy and efficiency of numerical methods for DDEs are not compromised. Some interpolation techniques can be referred to [13] and [14].

In this paper we consider a two-point block method as a numerical solution for systems of first order DDEs of the form:

$$y'(x) = f(x, y(x), y(x-\tau_1), \ldots, y(x-\tau_n)), \; x \in [a,b],$$
$$y(x) = \varphi(x), \qquad x \in [\bar{a}, a], \qquad \bar{a} = \min_{x \in [a,b]} (x - \tau_i),$$

where φ is the initial function and $\tau_1, \tau_2, \ldots, \tau_n > 0$ are either constant, time dependent or state dependent lag functions. The function f is continuous and satisfies a Lipschitz condition which guarantees the existence of a unique solution. The two-point block method is implemented in variable stepsize variable order scheme. We develop an efficient technique by calculating the coefficients beforehand and storing them at the start of the code. This technique eliminates the computational cost of recalculating the integration coefficients whenever a stepsize changes. The delay terms are computed by using Lagrange and Hermite interpolations. We compare the results of these two interpolation techniques.

The organization of this paper is as follows. In section II, we describe the formulae derivation for two-point block method using different back values. The interpolation approaches and the method development together with the stepsize and order changing strategies are also discussed. Numerical results from some test examples are presented in section III and section IV is the conclusion.

II. Formulae Derivation and Method Development

A. Grid-point Formulae

For simplicity, we consider a single delay scalar equation of the form:

$$\begin{aligned} y'(x) &= f(x, y(x), y(x-\tau)), \quad x \in [a,b], \\ y(x) &= \varphi(x), \qquad\qquad\quad x \in [\bar{a}, a]. \end{aligned} \tag{1}$$

Extension to systems of DDEs with multiple delays is obvious. The interval $[a,b]$ is divided into series of blocks with non-

F. Ishak is with the Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, MALAYSIA (phone: +603-5543-5348; fax: +603-5543-5501; e-mail: fuziyah@tmsk.uitm.edu.my).

Z. A. Majid is with the Department of Mathematics, Universiti Putra Malaysia, 43400 Serdang, Selangor, MALAYSIA (e-mail: zanariah@math.upm.edu.my).

M. B. Suleiman is with the Department of Mathematics, Universiti Putra Malaysia, 43400 Serdang, Selangor, MALAYSIA (e-mail: mohamed@math.upm.edu.my).

uniform grids $a = x_0, x_1, \ldots, x_{n-1}, x_n, x_{n+1}, \ldots, x_N = b.$ Each block contains two grids of equal size, but the length of earlier blocks may not be of the same length as the current block. The points x_{n+1} and x_{n+2} are contained in the current block. We evaluate $y_h(x_{n+1})$ and $y_h(x_{n+2})$, together with their corresponding delay solutions. The notation $y_h(x_{n+1})$ refers to the approximation of $y(x_{n+1})$ where y is the solution of (1). The formulae for $y_h(x_{n+1})$ and $y_h(x_{n+2})$ use the same number of back values. We refer to the number of back values in terms of blocks. In this paper, the block method approximates two new values at x_{n+1} and x_{n+2} by using the previous one, two, or three blocks. Fig. 1 shows a two-point one-block method where the back values at x_{n-2}, x_{n-1} and x_n are used to evaluate $y_h(x_{n+1})$ and $y_h(x_{n+2})$. The length of the current block is $2h$ while the length of the previous block is $2rh$. Since we will consider either keeping the same stepsize, halving or doubling at each integration step, r takes the value of either 1, 2 or $\frac{1}{2}$.

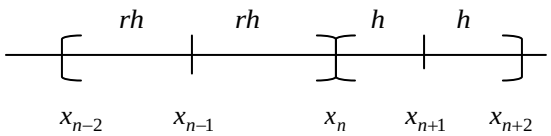

Fig. 1: Two-point one-block method

The grid-point formulae for two-point one-block method are derived by integrating equation (1) and replacing the function f by the polynomial $P_5(x)$ that interpolates f at the points $(x_{n+2-i}, f_{n+2-i}), i = 0,1,2,3,4.$ The notation f_{n+2} means $f(x_{n+2}, y_h(x_{n+2}), y_h(x_{n+2} - \tau))$. Thus,

$$y_h(x_{n+1}) = y_h(x_n) + \int_{x_n}^{x_{n+1}} P_5(x)dx$$

and

$$y_h(x_{n+2}) = y_h(x_n) + \int_{x_n}^{x_{n+2}} P_5(x)dx.$$

The polynomial $P_5(x)$ are written in Lagrange form, such as,

$$P_5(x) = \sum_{j=0}^{4} L_{4,j}(x) f_{n+2-j}$$

where

$$L_{\Phi,j}(x) = \prod_{\substack{i=0 \\ i \neq j}}^{\Phi} \frac{(x - x_{n+2-i})}{(x_{n+2-j} - x_{n+2-i})}, \text{ for } j = 0,1,\ldots,\Phi.$$

By letting $x = x_{n+2} + sh$, we have

$$y_h(x_{n+1}) = y_h(x_n) + h\int_{-2}^{-1} P_5(s)ds, \quad (2)$$

and

$$y_h(x_{n+2}) = y_h(x_n) + h\int_{-2}^{0} P_5(s)ds. \quad (3)$$

Solving the integrals in (2) and (3) gives the coefficients of the methods in terms of r. The results are as follows,

$$y_h(x_{n+1}) = y_h(x_n) + \frac{h}{240r^2(2r+1)(r+2)(r+1)}[$$
$$(r+2)(15r+7) f_{n-2} - 4(2r+1)(30r+7) f_{n-1} +$$
$$(2r+1)(r+2)(r+1)(100r^2+45r+7) f_n +$$
$$(4r^2)(r+2)(80r^2+75r+18) f_{n+1} -$$
$$r^2(2r+1)(20r^2+15r+3) f_{n+2}],$$

and

$$y_h(x_{n+2}) = y_h(x_n) + \frac{h}{15r^2(2r+1)(r+2)(r+1)}[$$
$$-(r+2) f_{n-2} - 4(2r+1) f_{n-1} +$$
$$(2r+1)(r+2)(r+1)(5r^2-1) f_n +$$
$$(4r^2)(r+2)(10r^2+15r+6) f_{n+1} -$$
$$r^2(2r+1)(5r^2+15r+9) f_{n+2}],$$

The coefficients are stored at the beginning of the code for $r = 1, 2,$ and $\frac{1}{2}$. The formulae for the first and second points can be written as

$$y_h(x_{n+1}) = y_h(x_n) + h\sum_{i=0}^{4} \beta_{1i} f_{n-2+i},$$

and

$$y_h(x_{n+2}) = y_h(x_n) + h\sum_{i=0}^{4} \beta_{2i} f_{n-2+i},$$

respectively. For $r = 1, 2,$ and $\frac{1}{2}$, the coefficients for the first and second points are given in Table I and Table II respectively.

TABLE I: THE FIRST POINT COEFFICIENTS FOR TWO-POINT ONE-BLOCK METHOD

r	β_{20}	β_{21}	β_{22}	β_{23}	β_{24}
1	$-\frac{74}{720}$	$\frac{11}{720}$	$\frac{456}{720}$	$\frac{346}{720}$	$-\frac{19}{720}$
2	$\frac{37}{14400}$	$-\frac{335}{14400}$	$\frac{7455}{14400}$	$\frac{7808}{14400}$	$-\frac{565}{14400}$
$\frac{1}{2}$	$\frac{145}{1800}$	$-\frac{704}{1800}$	$\frac{1635}{1800}$	$\frac{755}{1800}$	$-\frac{31}{1800}$

TABLE II: THE SECOND POINT COEFFICIENTS FOR TWO-POINT ONE-BLOCK METHOD

r	β_{20}	β_{21}	β_{22}	β_{23}	β_{24}
1	$-\frac{1}{90}$	$\frac{4}{90}$	$\frac{24}{90}$	$\frac{124}{90}$	$\frac{29}{90}$
2	$-\frac{1}{900}$	$\frac{5}{900}$	$\frac{285}{900}$	$\frac{1216}{900}$	$\frac{295}{900}$
$\frac{1}{2}$	$-\frac{20}{225}$	$\frac{64}{225}$	$\frac{15}{225}$	$\frac{320}{225}$	$\frac{71}{225}$

The derivation for two-point two-block method and two-point three-block method are done similarly. In two-point two-block method, two new values are obtained in the current block by using the back values of two previous blocks. Fig. 2 shows the two-point two-block method. Referring to Fig. 2, the length of the current block is $2h$, while the length of each of the previous two blocks is $2rh$ and $2qh$.

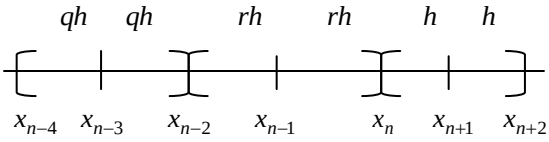

Fig. 2: Two-point two-block method

The function f will be replaced by the interpolating polynomial $P_7(x)$, such that

$$P_7(x) = \sum_{j=0}^{6} L_{6,j}(x) f_{n+2-j}.$$

Integrating (1) from x_n to x_{n+1} and changing the limit of integration yields the formula for the first point, such as

$$y_h(x_{n+1}) = y_h(x_n) + h\int_{-2}^{-1} P_7(s)ds. \qquad (4)$$

The integral in (4) is solved using MAPLE software where the resulting coefficients are in terms of q and r. For predetermined values of q and r, the following coefficients for the first point formula,

$$y_h(x_{n+1}) = y_h(x_n) + h\sum_{i=0}^{6} \beta_{1i}^{*} f_{n-4+i}$$

are stored in the code. The coefficients for $(r=1,q=1)$, $(r=1,q=2)$, $\left(r=1,q=\frac{1}{2}\right)$, $(r=2,q=2)$ and $\left(r=\frac{1}{2},q=\frac{1}{2}\right)$ are given in Table IIIa and Table IIIb.

TABLE IIIa: THE FIRST POINT COEFFICIENTS FOR TWO-POINT TWO-BLOCK METHOD

(r,q)	β_{10}^{*}	β_{11}^{*}	β_{12}^{*}	β_{13}^{*}
(1,1)	$\frac{271}{60480}$	$\frac{-2088}{60480}$	$\frac{7299}{60480}$	$\frac{-16256}{60480}$
(1,2)	$\frac{261}{846720}$	$\frac{-3514}{846720}$	$\frac{47376}{846720}$	$\frac{-169120}{846720}$
$\left(1,\frac{1}{2}\right)$	$\frac{29295}{635040}$	$\frac{-138572}{635040}$	$\frac{204687}{635040}$	$\frac{-227598}{635040}$
(2,2)	$\frac{5285}{10160640}$	$\frac{5285}{10160640}$	$\frac{5285}{10160640}$	$\frac{5285}{10160640}$
$\left(\frac{1}{2},\frac{1}{2}\right)$	$\frac{4417}{105840}$	$\frac{-30144}{105840}$	$\frac{87402}{105840}$	$\frac{-139328}{105840}$

TABLE IIIb: THE FIRST POINT COEFFICIENTS FOR TWO-POINT TWO-BLOCK METHOD, CONTINUE

(r,q)	β_{14}^{*}	β_{15}^{*}	β_{16}^{*}
(1,1)	$\frac{46989}{60480}$	$\frac{25128}{60480}$	$\frac{-863}{60480}$
(1,2)	$\frac{1621306}{846720}$	$\frac{364320}{846720}$	$\frac{-13909}{846720}$
$\left(1,\frac{1}{2}\right)$	$\frac{518994}{635040}$	$\frac{256527}{635040}$	$\frac{-8113}{635040}$
(2,2)	$\frac{5285}{10160640}$	$\frac{5285}{10160640}$	$\frac{5285}{10160640}$
$\left(\frac{1}{2},\frac{1}{2}\right)$	$\frac{148512}{105840}$	$\frac{35686}{105840}$	$\frac{-705}{105840}$

For the second point, we integrate (1) from x_n to x_{n+2}. The function f is replaced with the interpolating polynomial $P_7(x)$. We solve the resulting integral to obtain the formula for the second point, that is,

$$y_h(x_{n+2}) = y_h(x_n) + h\sum_{i=0}^{6} \beta_{2i}^{*} f_{n-4+i}.$$

The integration coefficients for various values of q and r are stored in the code. The coefficients for various values of q and r such as $(r=1,q=1)$, $(r=1,q=2)$, $\left(r=1,q=\frac{1}{2}\right)$, $(r=2,q=2)$ and $\left(r=\frac{1}{2},q=\frac{1}{2}\right)$ are given in Table IVa and Table IVb.

TABLE IVa: THE SECOND POINT COEFFICIENTS FOR TWO-POINT TWO-BLOCK METHOD

(r,q)	β_{20}^{*}	β_{21}^{*}	β_{22}^{*}	β_{23}^{*}
(1,1)	$\frac{-37}{3780}$	$\frac{264}{3780}$	$\frac{-807}{3780}$	$\frac{1328}{3780}$
(1,2)	$\frac{-33}{3780}$	$\frac{406}{3780}$	$\frac{-4368}{3780}$	$\frac{11200}{3780}$
$\left(1,\frac{1}{2}\right)$	$\frac{-4221}{39690}$	$\frac{18944}{39690}$	$\frac{-25956}{39690}$	$\frac{21714}{39690}$

(2,2)	$\frac{-175}{317520}$	$\frac{1377}{317520}$	$\frac{-4914}{317520}$	$\frac{10542}{317520}$
$\left(\frac{1}{2},\frac{1}{2}\right)$	$\frac{-2387}{13230}$	$\frac{14976}{13230}$	$\frac{-38052}{13230}$	$\frac{47488}{13230}$

TABLE IVb: THE SECOND POINT COEFFICIENTS FOR TWO-POINT TWO-BLOCK METHOD, CONTINUE

(r,q)	β_{24}^{*}	β_{25}^{*}	β_{26}^{*}
(1,1)	$\frac{33}{3780}$	$\frac{5640}{3780}$	$\frac{1139}{3780}$
(1,2)	$\frac{5082}{3780}$	$\frac{77376}{3780}$	$\frac{16177}{3780}$
$\left(1,\frac{1}{2}\right)$	$\frac{-3150}{39690}$	$\frac{60219}{39690}$	$\frac{11830}{39690}$
(2,2)	$\frac{85617}{317520}$	$\frac{441856}{317520}$	$\frac{100737}{317520}$
$\left(\frac{1}{2},\frac{1}{2}\right)$	$\frac{-21672}{13230}$	$\frac{22372}{13230}$	$\frac{3735}{13230}$

Similarly, in two-point three-block method, two new values are obtained by using the back values of three previous blocks. The two-point three-block method is shown in Fig. 3. The length of the current block is $2h$, while the length of each of the previous three blocks is $2rh$, $2qh$ and $2ph$.

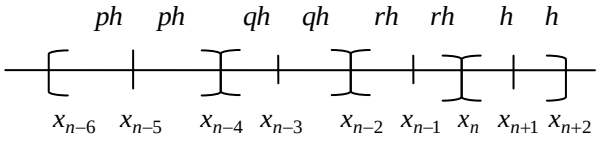

Fig. 3: Two-point three-block method

The derivation for two-point three-block formulae are carried out similarly with the earlier methods. We integrate (1) and replace f with the interpolating polynomial $P_9(x)$, where

$$P_9(x) = \sum_{j=0}^{8} L_{8,j}(x) f_{n+2-j}.$$

Integrating (1) from x_n to x_{n+1} and changing the limit of integration yields the formula for the first point, such as

$$y_h(x_{n+1}) = y_h(x_n) + h\int_{-2}^{-1} P_9(s)ds.$$

The integral is solved using MAPLE software where the resulting coefficients are in terms of p, q and r. For predetermined values of p, q and r, the following coefficients for the first point formula,

$$y_h(x_{n+1}) = y_h(x_n) + h\sum_{i=0}^{8} \beta_{1i}^{**} f_{n-6+i}$$

are stored in the code. Similarly, integrating (1) from x_n to x_{n+2} and changing the limit of integration yields the formula for the second point, such as

$$y_h(x_{n+2}) = y_h(x_n) + h\int_{-2}^{0} P_9(s)ds.$$

The formula for the second point is given by

$$y_h(x_{n+2}) = y_h(x_n) + h\sum_{i=0}^{8} \beta_{2i}^{**} f_{n-6+i}.$$

The coefficients are in terms of p, q and r. The values for p, q and r are predetermined based on the stepsize changing strategy. The coefficients for these values are stored in the code and used according to the changing stepsize. This practice eliminates the cost of recalculating the coefficients when the stepsize changes. Based on empirical studies, we store the coefficients for different values of p, q and r such as $(r=1,q=1,p=1)$, $\left(r=\frac{1}{2},q=\frac{1}{2},p=\frac{1}{2}\right)$, $\left(r=1,q=\frac{1}{2},p=\frac{1}{2}\right)$, $\left(r=1,q=1,p=\frac{1}{2}\right)$, $\left(r=\frac{1}{2},q=\frac{1}{2},p=1\right)$, $\left(r=\frac{1}{2},q=\frac{1}{2},p=\frac{1}{4}\right)$, $(r=2,q=2,p=2)$, $(r=2,q=2,p=1)$, $(r=1,q=2,p=2)$ and $(r=1,q=1,p=2)$.

B. Delay Solution

In advancing a step, we require the solutions of the delay terms $y_h(\alpha_1)$ and $y_h(\alpha_2)$, where $\alpha_1 = x_{n+1} - \tau$ and $\alpha_2 = x_{n+2} - \tau$. For $d = 1,2$, if $\alpha_d \le a$, then $y_h(\alpha_d) = \varphi(\alpha_d)$. Otherwise, we interpolate the delay solutions using the approximate solutions at grid points. In this algorithm, the delay solutions are obtained by using Lagrange and Hermite interpolating polynomials.

For Lagrange polynomial, the number of interpolation points depend upon the interval where α_d lies. If $x_j < \alpha_d \le x_{j+1}$, and t is the number of interpolation points, then t is either:

- six, if $y_h(x_{j+1})$ is obtained by the two-point one-block method, or
- eight, if $y_h(x_{j+1})$ is obtained by the two-point two-block method, or
- ten, if $y_h(x_{j+1})$ is obtained by the two-point three-block method.

Thus the delay solution $y_h(\alpha_d)$ is given by,

$$y_h(\alpha_d) = \sum_{i=j-t+2}^{j+1} y_h(x_i) L_{t-1,i}(\alpha_d),$$

where

$$L_{t-1,i}(\alpha_d) = \prod_{\substack{s=j-t+2 \\ s\neq i}}^{j+1} \frac{(\alpha_d - x_s)}{(x_i - x_s)}.$$

The number of interpolation points used in Lagrange polynomial is always one point higher than the number of interpolation points used for the solution at grid points. This practice preserves the order of the method at the particular interval.

For Hermite interpolation, the numbers of interpolation points used are at most four. If $x_j < \alpha_d \le x_{j+1}$, and the number of interpolation points is three, then $y_h(\alpha_d)$ that is approximated by the Hermite polynomial of degree at most five is given by

$$y_h(\alpha_d) = \sum_{i=0}^{2} y_h(x_{j+i}) H_{2,j+i}(\alpha_d) + \sum_{i=0}^{2} f_{j+i} \hat{H}_{2,j+i}(\alpha_d),$$

where

$$H_{2,j+i}(\alpha_d) = \left[1 - 2(\alpha_d - x_{j+i}) L'_{2,j+i}(x_{j+i})\right] L^2_{2,j+i}(\alpha_d),$$

$$\hat{H}_{2,j+i}(\alpha_d) = (\alpha_d - x_{j+i}) L^2_{2,j+i}(\alpha_d),$$

and

$$L_{2,j+i}(\alpha_d) = \prod_{\substack{s=0 \\ s\neq i}}^{2} \frac{(\alpha_d - x_{j+s})}{(x_{j+i} - x_{j+s})}.$$

The advantage of using Hermite interpolating polynomial lies in the fact that the number of interpolation points required is less than the number of interpolation points in Lagrange polynomial in order to have the same degree. Moreover, Hermite polynomial conveniently uses the derivative of the approximated solutions at grid points.

C. Error Estimation, Stepsize and Order Strategy

We implement the algorithm in the predictor-corrector scheme where the corrector is iterated until convergence. Empirical results show that a reliable way to control the error at the grid points is to control the local error at the second point. The local error $E_{2,k}$ at the second point of order k is estimated by comparing the formulae of different orders. Thus,

$$E_{2,k} = \hat{y}_h(x_{n+2}) - y_h(x_{n+2}),$$

where $\hat{y}_h(x_{n+2})$ is the estimation for $y(x_{n+2})$ using order $k+1$. In this algorithm k takes the values of 4, 6 and 8, depending upon the number of blocks used.

At each step, the result is accepted if $E_{2,k}$ satisfies the user specified tolerance. Otherwise, the step fails and the result is rejected. After each successful step, we consider changing the stepsize and order of the method. For reason of stability, we raise the order only if we have been using a constant stepsize and the same order had been repeated at least twice. The order is kept the same or reduced depending whether the estimated stepsize for the next step using the new order is the maximum.

As for the new stepsize, we first consider h_k, the estimated stepsize for each possible order as follows,

$$h_k = h\left(\frac{\mathrm{TOL}}{|E_{2,k}|}\right)^{\frac{1}{k+1}},$$

where h is the current stepsize and TOL is the user specified tolerance. After a successful step, the new stepsize is either doubled or kept the same. We double the stepsize if the maximum stepsize, $h_{\max} = \max_{k\in\{4,6,8\}} h_k$ is such that $0.8h_{\max} \ge 2h$. The new stepsize is equal to h if $0.8h_{\max} \le 2h$. The value 0.8 is taken as a safety factor to avoid having too many rejected steps. For a step failure, we do not consider raising the order. The new stepsize is reduced by half. For a repeated failure, we consider a restart where the stepsize is reduced to a minimum, together with method of order one. In this code we store the coefficients of the formulae that are varied by constant, half or double the stepsize only. This practice proves to be efficient since the choice for the most optimal stepsize is either kept constant, halved or doubled.

III. Numerical Results

In this section, we describe the numerical results for the block method derived earlier. Two interpolation types, which are Lagrange and Hermite interpolations, are used to approximate the delay solutions. A wide range of DDEs with exact solutions are used in order to test the efficiency and the accuracy of the block method using these two types of interpolating polynomials. Six test equations from existing literatures are taken as examples here. Example 1 is taken from [15] while Example 2 – Example 6 are taken from [16]. Numerical results are tabulated and the accuracy of the method is justified by evaluating the maximum and average errors.

Example 1:

$$y'(x) = \cos(x)y(y(x)-2), \quad 0 \le x \le 50,$$
$$y(x) = 1, \quad x \le 0.$$

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

Example 2:

$$y'(x) = y\left(\frac{x}{(1+2x)^2}\right)^{(1+2x)^2}, \quad 0 \le x \le 1,$$
$$y(0) = 1.$$

The exact solution is $y(x) = e^x$.

Example 3:

$$y'(x) = 1 - y\left(\exp\left(1 - \frac{1}{x}\right)\right), \quad 2 \le x \le 100,$$
$$y(x) = \ln(x), \qquad 0 \le x \le 2.$$

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

Example 4:

$$y_1'(x) = y_5(x-1) + y_3(x-1), \qquad 0 \le x \le 1,$$
$$y_2'(x) = y_1(x-1) + y_2(x-0.5), \quad 0 \le x \le 1,$$
$$y_3'(x) = y_3(x-1) + y_1(x-0.5), \quad 0 \le x \le 1,$$
$$y_4'(x) = y_5(x-1) \times y_4(x-1), \quad 0 \le x \le 1,$$
$$y_5'(x) = y_1(x-1), \qquad 0 \le x \le 1,$$

with the initial condition

$$y_1(x) = y_4(x) = y_5(x) = \exp(x+1), \quad -1 \le x \le 0,$$
$$y_2(x) = \exp(x+0.5), \qquad -1 \le x \le 0,$$
$$y_3(x) = \sin(x+1), \qquad -1 \le x \le 0.$$

The exact solution is

$$y_1(x) = e^x - \cos x + e, \qquad 0 \le x \le 1,$$
$$y_2(x) = \begin{cases} 2e^x + e^{0.5} - 2, & 0 \le x \le 0.5, \\ e^x + 2e^{x-0.5} + xe^{0.5} - 2x & \\ \quad + 1.5e^{0.5} - 3, & 0.5 \le x \le 1, \end{cases}$$
$$y_3(x) = \begin{cases} e^{x+0.5} - \cos x + 1 - e^{0.5} & \\ \quad + \sin 1, & 0 \le x \le 0.5, \\ -\cos x + e^{x-0.5} - \sin(x-0.5) & \\ \quad + (x+0.5)e - e^{0.5} + \sin 1, & 0.5 \le x \le 1, \end{cases}$$
$$y_4(x) = 0.5e^{2x} - 0.5 + e, \qquad 0 \le x \le 1,$$
$$y_5(x) = e^x + e - 1, \qquad 0 \le x \le 1.$$

Example 5:

$$y_1'(x) = y_3(x), \qquad x \ge 0,$$
$$y_2'(x) = y_4(x), \qquad x \ge 0,$$
$$y_3'(x) = -2my_2(x) + (1+m^2)(-1)^m y_1(x-\pi), \qquad x \ge 0,$$
$$y_4'(x) = -2my_1(x) + (1+m^2)(-1)^m y_2(x-\pi), \qquad x \ge 0,$$

with the initial condition

$$y_1(x) = \sin(x)\cos(mx), \qquad x \le 0,$$
$$y_2(x) = \cos(x)\sin(mx), \qquad x \le 0,$$
$$y_3(x) = \cos(x)\cos(mx) - m\sin(x)\sin(mx), \qquad x \le 0,$$
$$y_4(x) = m\cos(x)\cos(mx) - \sin(x)\sin(mx), \qquad x \le 0.$$

The exact solution is

$$y_1(x) = \sin(x)\cos(mx), \qquad x \ge 0,$$
$$y_2(x) = \cos(x)\sin(mx), \qquad x \ge 0,$$
$$y_3(x) = \cos(x)\cos(mx) - m\sin(x)\sin(mx), \qquad x \ge 0,$$
$$y_4(x) = m\cos(x)\cos(mx) - \sin(x)\sin(mx), \qquad x \ge 0.$$

When $m = 1$, the system is solved for $0 \le x \le 5$.

Example 6:

$$y_1'(x) = y_2(x), \qquad 0 \le x \le 5,$$
$$y_2'(x) = 2y_1\left(x - \tfrac{\pi}{2}\right) + \exp(\sin(x))\left(\cos(x)^2 - \sin(x)\right) - 2\exp(-\cos(x)), \qquad 0 \le x \le 5,$$

with the initial condition

$$y_1(x) = \exp(\sin(x)), \qquad x \le 0,$$
$$y(x) = \cos(x)\exp(\sin(x)), \qquad x \le 0.$$

The exact solution is

$$y_1(x) = \exp(\sin(x)), \qquad x \ge 0,$$
$$y(x) = \cos(x)\exp(\sin(x)), \qquad x \ge 0.$$

The error at the grid point for each component, err_i is defined as

$$(err_i)_t = \left| \frac{(y_h(x_i))_t - (y_h(x_i))_t}{A + B(y(x_i))_t} \right|,$$

where $(y)_t$ is the t-th component of y and $y(x_i)$ is the exact solution at x_i. A and B may take the values of either 1 or 0 depending upon the type of error test chosen. In this case, we use mix error test where $A = 1$ and $B = 1$ as opposed to absolute error test and relative error test. For absolute error test, $A = 1$ and $B = 0$, while for relative error test, $A = 0$ and $B = 1$. The maximum error, MAXE and the average error, AVERR are defined as follows,

$$\text{MAXE} = \max_{1 \le i \le \text{SSTEP}} \left(\max_{1 \le t \le \text{N}} (err_i)_t \right),$$

$$\text{AVERR} = \frac{\sum_{i=1}^{\text{SSTEP}} \sum_{t=1}^{\text{N}} (err_i)_t}{(\text{N})(\text{SSTEP})},$$

where N is the number of equations in the system and SSTEP is the total number of successful steps.

Numerical results for Example 1 – Example 6 are given in Table V – Table X respectively. The following abbreviations are used in the tables: TOL – the user specified tolerance, INT – interpolation types, LGR – Lagrange, HMT – Hermite, STEP – the total number of steps, FS – the number of failed steps, AVERR – the average error and MAXE – the maximum error. The notation 3.60652E-01 means 3.90952×10^{-1}.

TABLE V: NUMERICAL RESULTS FOR EXAMPLE 1

TOL	INT	STEP	FS	AVERR	MAXE
10^{-2}	LGR	67	6	5.82248E-02	2.66059E-01
	HMT	67	6	5.82432E-02	2.66135E-01
10^{-4}	LGR	94	3	8.71758E-06	4.84840E-05
	HMT	94	3	8.71758E-06	4.84840E-05
10^{-6}	LGR	109	0	3.67883E-08	1.46129E-07
	HMT	109	0	3.67883E-08	1.46129E-07
10^{-8}	LGR	226	1	1.53328E-09	3.25542E-09
	HMT	226	1	1.53328E-09	3.25542E-09
10^{-10}	LGR	281	0	2.38966E-12	7.36582E-12
	HMT	281	0	2.38985E-12	7.36644E-12

TABLE VI: NUMERICAL RESULTS FOR EXAMPLE 2

TOL	INT	STEP	FS	AVERR	MAXE
10^{-2}	LGR	22	0	5.82936E-03	5.82897E-02
	HMT	24	0	2.18544E-04	1.20016E-03
10^{-4}	LGR	35	0	7.57052E-05	3.96581E-04
	HMT	35	0	7.86277E-05	4.30244E-04
10^{-6}	LGR	43	0	5.45077E-07	1.01137E-05
	HMT	61	2	1.24757E-06	1.01137E-05
10^{-8}	LGR	53	0	3.42637E-08	2.30481E-07
	HMT	50	0	2.83289E-08	2.25246E-07
10^{-10}	LGR	69	0	7.62940E-11	4.58373E-10
	HMT	65	0	6.39557E-11	3.65983E-10

TABLE VII: NUMERICAL RESULTS FOR EXAMPLE 3

TOL	INT	STEP	FS	AVERR	MAXE
10^{-2}	LGR	36	0	2.36935E-05	1.22157E-04
	HMT	36	0	2.35219E-05	1.19211E-04
10^{-4}	LGR	51	0	3.15039E-06	1.07738E-05
	HMT	51	0	3.10917E-06	1.06626E-05
10^{-6}	LGR	70	0	7.79491E-09	2.73688E-08
	HMT	70	0	7.63518E-09	2.42246E-08
10^{-8}	LGR	97	0	3.08642E-10	8.59520E-10
	HMT	97	0	2.27569E-10	5.00718E-10
10^{-10}	LGR	138	0	1.93015E-12	5.39480E-12
	HMT	138	0	1.58373E-12	7.51754E-12

TABLE VIII: NUMERICAL RESULTS FOR EXAMPLE 4

TOL	INT	STEP	FS	AVERR	MAXE
10^{-2}	LGR	21	0	8.34773E-06	6.50548E-04
	HMT	21	0	8.34873E-06	6.50548E-04
10^{-4}	LGR	31	1	1.28298E-05	5.36563E-04
	HMT	31	1	1.28298E-05	5.36563E-04
10^{-6}	LGR	57	2	1.29884E-07	3.46274E-06
	HMT	57	2	1.30167E-07	3.46274E-06
10^{-8}	LGR	89	8	4.84291E-09	5.51480E-07
	HMT	85	6	6.14893E-10	1.03694E-08
10^{-10}	LGR	113	6	6.31359E-11	1.66644E-09
	HMT	113	6	6.31375E-11	1.66644E-09

TABLE IX: NUMERICAL RESULTS FOR EXAMPLE 5

TOL	INT	STEP	FS	AVERR	MAXE
10^{-2}	LGR	28	0	6.36851E-05	5.84259E-04
	HMT	28	0	6.31184E-05	5.84259E-04
10^{-4}	LGR	37	0	5.60661E-07	5.07276E-06
	HMT	37	0	5.81504E-07	5.06844E-06
10^{-6}	LGR	46	0	3.59609E-08	4.77830E-07
	HMT	46	0	3.75275E-08	4.78263E-07
10^{-8}	LGR	73	0	1.13111E-10	5.34061E-10
	HMT	73	0	1.13328E-10	5.36978E-10
10^{-10}	LGR	88	0	3.38100E-12	2.55492E-11
	HMT	88	0	3.36892E-12	2.54521E-11

TABLE X: NUMERICAL RESULTS FOR EXAMPLE 6

TOL	INT	STEP	FS	AVERR	MAXE
10^{-2}	LGR	26	0	2.07283E-04	5.43466E-03
	HMT	26	0	2.26111E-04	5.10223E-03
10^{-4}	LGR	37	0	5.37637E-06	6.43235E-05
	HMT	37	0	4.95181E-06	5.78668E-05
10^{-6}	LGR	59	0	7.60003E-09	6.17010E-08
	HMT	58	0	5.39406E-09	4.69602E-08
10^{-8}	LGR	73	0	4.87166E-10	3.74785E-09
	HMT	73	0	4.88089E-10	3.75407E-09
10^{-10}	LGR	114	1	1.31627E-10	1.15901E-09
	HMT	114	1	1.29950E-10	1.14370E-09

The plots of TOL vs. MAXE for the two-point block method with both Lagrange and Hermite interpolations for Example 1 – Example 6 are shown in Fig. 4 – Fig. 9. The plots for each example provide a crude measure of the tolerance proportionality that is achieved by the block method using both interpolation types.

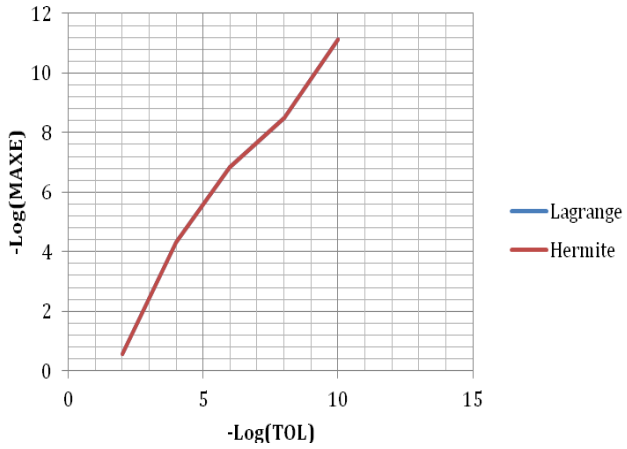

Fig. 4: TOL vs. MAXE graphs using Lagrange and Hermite interpolations for Example 1

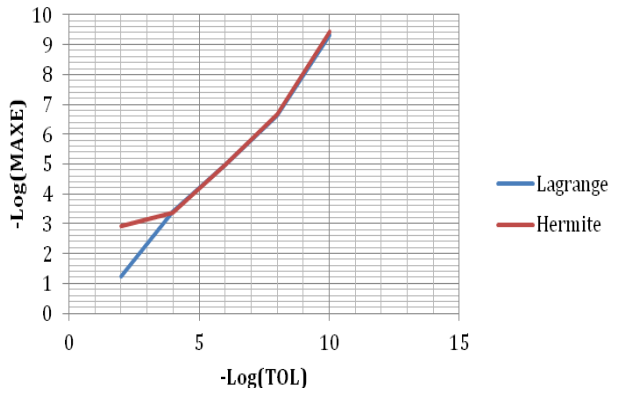

Fig. 5: TOL vs. MAXE graphs using Lagrange and Hermite interpolations for Example 2

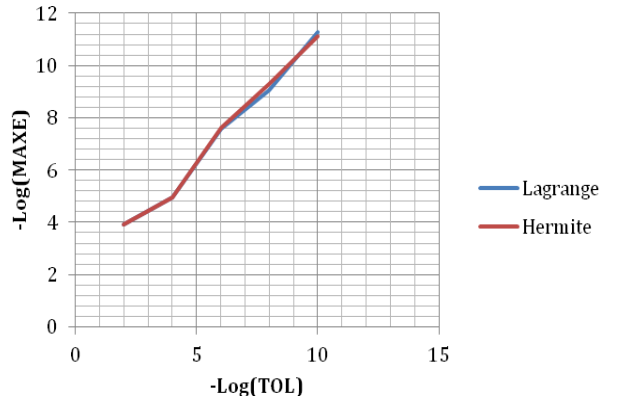

Fig. 6: TOL vs. MAXE graphs using Lagrange and Hermite interpolations for Example 3

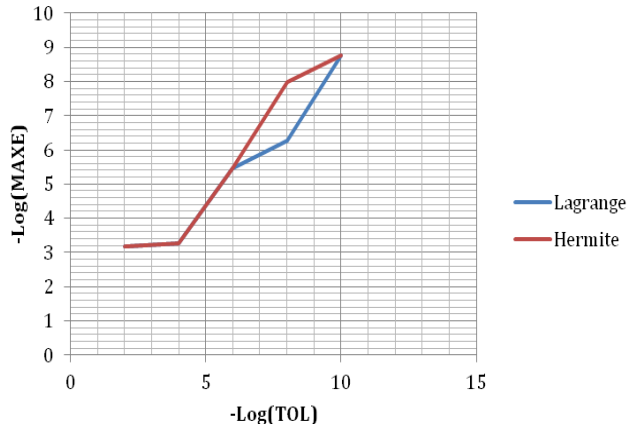

Fig. 7: TOL vs. MAXE graphs using Lagrange and Hermite interpolations for Example 4

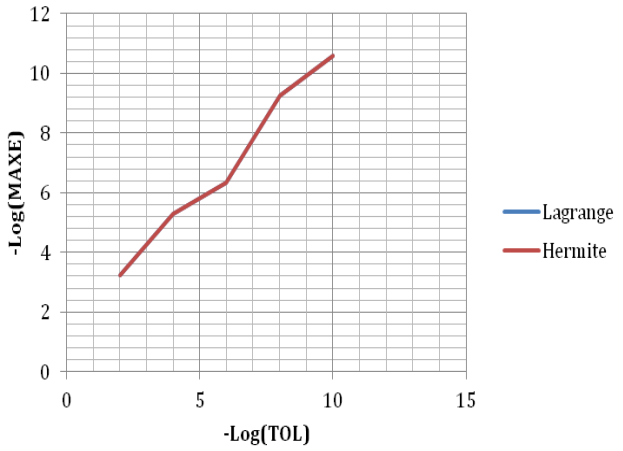

Fig. 8: TOL vs. MAXE graphs using Lagrange and Hermite interpolations for Example 5

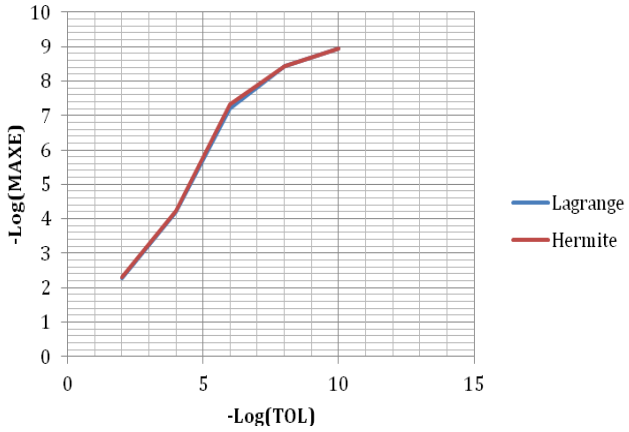

Fig. 9: TOL vs. MAXE graphs using Lagrange and Hermite interpolations for Example 6

It is clearly seen from the tables that for a given tolerance, the average and the maximum errors for all examples using Lagrange and Hermite interpolations are within acceptable range. As the tolerance gets smaller, it is expected that the total number of steps increases. In order to achieve the desired accuracy and to avoid failed steps, smaller stepsizes are taken. The implicit block method varies the stepsize and/or order of the method while obtaining two new values in a single integration step. The method is efficient because the most optimal stepsize and the correct order are chosen to obtain the desired accuracy. Local error control at the second point also proves to be reliable in obtaining the desired accuracy as well as keeping the number of rejected steps at a minimum.

The block method with Lagrange and Hermite interpolations for approximating the delay solution achieves the desired accuracy. The performance of the method in terms of the proportionality of the maximum error to tolerance shows that the method achieves the desired accuracy as indicated by the

graphs of TOL vs. MAXE where the slope of these graphs are almost equal to one.

Hermite interpolation is simpler in terms of number of interpolation points. With fewer points than that of Lagrange interpolation, Hermite interpolation approximates the delay solutions to the desired order of accuracy.

It can be concluded that from the numerical results, the implicit two-point block method in variable stepsize and variable order is efficient in solving DDEs to the desired accuracy. Both interpolation types can be used to solve the delay solutions, although it is recommended that Hermite interpolation is superior in terms of using less number of interpolation points.

IV. Conclusion

In this paper, we have presented the development of a two-point implicit block method using variable stepsize variable order technique. The order of the method is varied by taking the back values consisting of one, two or three previous blocks. The coefficients of the method are calculated beforehand, thus avoiding the recalculation of the coefficients when the stepsize changes. The delay solutions are solved by using Lagrange and Hermite interpolating polynomials.

The numerical results indicate that the block method with both interpolation types achieves the desired accuracy as efficiently as possible. The two-point implicit block method can be used to solve a wide range of DDEs.

Acknowledgment

The authors would like to acknowledge the financial support received from Universiti Teknologi MARA and the Ministry of Higher Education of Malaysia (MOHE) under the Fundamental Research Grant Scheme (FRGS) 600-RMI/ST/FRGS 5/3/Fst (20/2011).

References

[1] R. D. Driver, *Ordinary and Delay Differential Equations*, Springer-Verlag New York Inc.,New York, 1977, pp. 225–240.

[2] Y. Kuang, *Delay Differential Equations: with Applications in Population Dynamics*, Academic Press Inc., San Diego, 1993.

[3] W. Panitsupakamon, and C. Rattanakul, "A delay differential equations model of bone formation and resorption: effect of calcitonin," in *Proc. 12th WSEAS Int. Conf. on Applied Comp. Science*, Singapore, 2012, pp. 58–63.

[4] A. Keller, "Generalized delay differential equations to economic dynamics and control," in *Proc. Of the American Conference on Applied Mathematics (AMERICAN_MATH `10)*, Cambridge, 2010, pp. 278–286.

[5] A. Ochoche, and P. Ndajah, "Almost Runge-Kutta methods of orders up to five," *WSEAS Transactions on Mathematics*, vol. 10, issue 5, pp. 159–168, May 2011.

[6] Y. Zhang, "Spectrum of a class of delay differential equations and its solution expansion," *WSEAS Transactions on Mathematics*, vol. 10, issue 5, pp. 169–180, May 2011.

[7] N. E. Mastorakis, "Numerical solution of non-linear ordinary differential equations via collocation method (finite elements) and genetics algorithms," in *Proc. 6th WSEAS Int. Conf. on Evolutionary Computing*, Lisbon, 2005, pp. 36–42.

[8] M. B. Suleiman, and F. Ishak, "Numerical solution and stability of multistep method for solving delay differential equations," *Japan J. Indust. Appl. Math.*, vol. 27, pp. 395–410, 2010.

[9] W. H. Enright, and H. Hayashi, "A delay differential equation solver based on a continuous Runge-Kutta method with defect control," *Numer. Algorithms*, vol. 16, pp. 349–364, 1997.

[10] Z. Jackiewicz, and E. Lo, "Numerical solution of neutral functional differential equations by Adams method in divided difference form," *Journal of Comp. and Appl. Math.*, vol. 189, pp. 592–605, 2006.

[11] F. Ishak, M. Suleiman, and Z. Omar, "Two-point predictor-corrector block method for solving delay differential equations," *Matematika*, vol. 24, no. 2, pp. 131–140, 2008.

[12] F. Ishak, Z. A. Majid, and M. Suleiman, "Two-point block method in variable stepsize technique for solving delay differential equations," *Journal of Materials Sc. and Eng.*, vol. 4, no. 12, pp. 86–90, 2010.

[13] C. M. Niculae, and M. Niculae. (2007). "Simple interpolator for delayed differential equations," *Computational Physics.* http://anale.fizica.unibuc.ro/archiva/2007/An_Fiz_2007_1.pdf

[14] F. Ismail, R.A. Al-Khasawneh, A. S Lwin, and M. Suleiman, "Numerical treatment of delay differential equations by Runge-Kutta method using Hermite interpolation," *Matematika*, vol. 18, no. 2, pp. 79–90, 2002.

[15] A. N. Al-Mutib, "Numerical methods for solving delay differential equations," Ph.D. thesis, University of Manchester, 1997.

[16] C. A. H. Paul, "A test set of functional differential equations," University of Manchester, Numerical Analysis Report no. 243, 1994.

.