Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2011, Article ID 193691, 12 pages
doi:10.1155/2011/193691

Research Article

A Quantitative Comparison of Numerical Method for Solving Stiff Ordinary Differential Equations

S. A. M. Yatim,[1] Z. B. Ibrahim,[1] K. I. Othman,[2] and M. B. Suleiman[1]

[1] *Department of Mathematics, Faculty of Science, Universiti Putra Malaysia (UPM), Selangor, 43400 Serdang, Malaysia*
[2] *Department of Mathematics, Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Selangor, 40450 Shah Alam, Malaysia*

Correspondence should be addressed to Z. B. Ibrahim, zarinabb@science.upm.edu.my

Received 17 April 2011; Accepted 28 June 2011

Academic Editor: Claude Lamarque

We derive a variable step of the implicit block methods based on the backward differentiation formulae (BDF) for solving stiff initial value problems (IVPs). A simplified strategy in controlling the step size is proposed with the aim of optimizing the performance in terms of precision and computation time. The numerical results obtained support the enhancement of the method proposed as compared to MATLAB's suite of ordinary differential equations (ODEs) solvers, namely, ode15s and ode23s.

1. Introduction

Consider the first-order ordinary differential equations in the form of

$$y' = f(x, y) \tag{1.1}$$

with given initial values $y(a) = y_0$ in the given interval $x \in [a, b]$. The system of (1.1) is said to be stiff if the eigenvalues of the matrix $\partial f(x, y)/\partial y$ have negative real parts at every time x and varies greatly in magnitude. Solving stiff ODEs which aroused from mainly applied problems, for example, from physical, chemical, and biological phenomena is made easy with several methods of interest appeared in subroutine libraries [1]. Some examples can be seen in [2–4]. From the code pioneered by Gear called DIFSUB, the wide interest in stiff ODEs has caused many other codes evolved to meet the same objective of finding the most accurate

approximation for IVPs [5]. Some renowned codes are EPISODE and LSODE [5]. With the advancements of the existing methods of solving ODEs, many of these codes are preinstalled in MATLAB to work as numerical solvers. An overview of MATLAB's numerical solvers presented in [2] shows a wide choice of solvers according to the problem type, step, and order to deal with stiff and nonstiff problems. Since then, several measures have been brought up by many researchers to evaluate which method turns out to be the most efficient method [6]. Despite that there exist some limitations in each method, these methods proven to have their own strength in solving certain initial value problems [1].

The need to solve large systems of stiff IVPs has also influenced the development of other existing method. These include iterative linear equation solvers and sparse direct linear equation solvers. Further discussion and additional references on this method are discussed in more detail in [2]. The method that incorporated with BDF proposed by Gear has been expanded gradually to become an improved method [7]. The study on producing block approximations $y_{n+1}, y_{n+2}, y_{n+3}, \ldots, y_{n+k}$ also known as block backward differentiation formulae (BBDF) [7] was inspired by Gear's method. BBDF method in [7] has verified the competency of computing concurrent solution values at different points. Apart from that, block approximations have been used in different methods to improve the methods to give a better accuracy and computation time. Consequently, the study by Ibrahim et al. in [8] is extended in a way that the accuracy is improved with reduction of total steps and lesser computational time.

In the next section, we discuss the derivation of the 5th-order variable step block backward differentiation formulae (5oBBDF). The strategies and implementations are presented next, followed by the results and discussion. It is also one of the main aims of this paper to prove that the method proposed acts as an alternative way in solving IVPs which arise in engineering and applied sciences. We are interested to compare the numerical results obtained with stiff ODEs solvers provided by MATLAB, namely, ode15s and ode23s.

2. General 5oBBDF Formulation

The ratio distance between current (x_n) and previous step (x_{n-1}) is represented as q in Figure 1. In this paper, the step size is given selection to decrease to half of the previous steps or increase up to a factor of 1.9. For simplicity, q is assigned as $1, 2$ and $10/19$ for the case of constant, halving, and increasing the step size, respectively. The zero stability is achieved for each of these cases and explained in the next section.

We find approximating polynomials $P_k(x)$ by means of a k-degree polynomial interpolating the values of y at given points are $(x_{n-2}, y_{n-2}), (x_{n-1}, y_{n-1}), \ldots, (x_{n+2}, y_{n+2})$,

$$P_k = \sum_{j=0}^{k} y\left(x_{n+1-j}\right) \cdot L_{k,j}(x), \tag{2.1}$$

where

$$L_{k,j}(x) = \prod_{\substack{i=0 \\ i \neq j}}^{k} \frac{(x - x_{n+1-i})}{\left(x_{n+1-j} - x_{n+1-i}\right)} \quad \text{for each } j = 0, 1, \ldots, k. \tag{2.2}$$

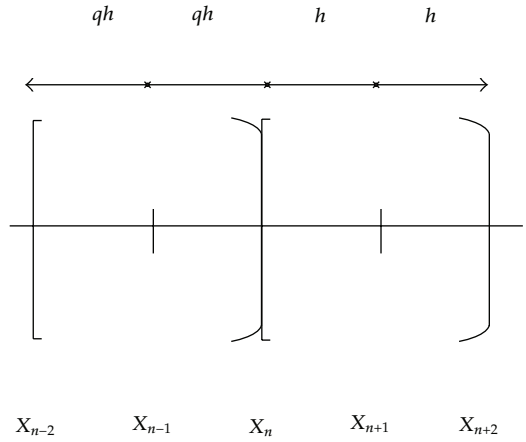

Figure 1: 5th order block method of variable step size.

Define $s = (x - x_{n+1})/h$ to find the 4th-order interpolating polynomial for (2.1),

$$\begin{aligned} P(x) &= P(x_{n+1} + sh) \\ &= \frac{(2q+1+s)(q+1+s)(1+s)(s)}{4(q+1)(q+2)} y_{n+2} + \frac{(2q+1+s)(q+1+s)(1+s)(s-1)}{-(q+1)(2q+1)} y_{n+1} \\ &\quad + \frac{(2q+1+s)(q+1+s)(s)(s-1)}{4q^2} y_n + \frac{(2q+1+s)(1+s)(s)(s-1)}{-q^2(q+1)(q+2)} y_{n-1} \\ &\quad + \frac{(q+1+s)(1+s)(s)(s-1)}{4q^2(2q+1)(q+1)} y_{n-2} \end{aligned} \tag{2.3}$$

s is equated to 0 and 1 after the polynomial is differentiated with respect to s, when $x = x_{n+1}$ and $x = x_{n+2}$, respectively,

$$\begin{aligned} hf_{n+1} &= \frac{1+2q^2+3q}{4(q+1)(q+2)} y_{n+2} + \frac{2+3q}{(q+1)(2q+1)} y_{n+1} + \frac{-1-2q^2-3q}{4q^2} y_n \\ &\quad + \frac{1+2q}{q^2(q+1)(q+2)} y_{n-1} + \frac{-1-q}{4q^2(2q+1)(q+1)} y_{n-2}, \\ hf_{n+2} &= \frac{10+3q^2+12q}{2(q+1)(q+2)} y_{n+2} + \frac{-4q^2-12q-8}{(q+1)(2q+1)} y_{n+1} + \frac{2+q^2+3q}{2q^2} y_n \\ &\quad + \frac{-4q-4}{q^2(q+1)(q+2)} y_{n-1} + \frac{q+2}{2q^2(2q+1)(q+1)} y_{n-2}. \end{aligned} \tag{2.4}$$

Upon substituting $q = 1, 2$ and $10/19$ into $P'(x_{n+1})$ and $P'(x_{n+2})$, we obtained the coefficients for points y_{n+1} and y_{n+2} as follows:

for $q = 1$,

$$y_{n+1} = \frac{6}{5}hf_{n+1} - \frac{3}{10}y_{n+2} + \frac{9}{5}y_n - \frac{3}{5}y_{n-1} + \frac{1}{10}y_{n-2},$$
$$y_{n+2} = \frac{12}{25}hf_{n+1} + \frac{48}{25}y_{n+2} - \frac{36}{25}y_n + \frac{16}{25}y_{n-1} - \frac{3}{25}y_{n-2},$$

for $q = 2$,

$$y_{n+1} = \frac{15}{8}hf_{n+1} - \frac{75}{128}y_{n+2} + \frac{225}{128}y_n - \frac{25}{128}y_{n-1} + \frac{3}{128}y_{n-2},$$
$$y_{n+2} = \frac{12}{23}hf_{n+2} + \frac{192}{115}y_{n+1} - \frac{18}{23}y_n + \frac{3}{23}y_{n-1} - \frac{2}{115}y_{n-2}, \tag{2.5}$$

for $q = 10/19$,

$$y_{n+1} = \frac{1131}{1292}hf_{n+1} - \frac{14703}{82688}y_{n+2} + \frac{1279161}{516800}y_n - \frac{183027}{108800}y_{n-1} + \frac{10469}{27200}y_{n-2},$$
$$y_{n+2} = \frac{1392}{3095}hf_{n+2} + \frac{89088}{40235}y_{n+1} - \frac{242208}{77375}y_n + \frac{198911}{77375}y_{n-1} - \frac{658464}{1005875}y_{n-2}.$$

3. Implementation of 5oBBDF Method

It is commonly known that stiff ODEs codes must solve both nonlinear and linear systems at each step of differentiation. This is due to implicity of the formulae for solving stiff IVPs. Throughout this section, we illustrate the effect of Newton-type scheme to find the approximation solutions of y_{n+1} and y_{n+2} simultaneously in every step. The general forms of the 5th-order BBDF method are

$$y_{n+1} = \alpha_1 hf_{n+1} + \theta_1 y_{n+2} + \varphi_1,$$
$$y_{n+2} = \alpha_1 hf_{n+2} + \theta_1 y_{n+1} + \varphi_2 \tag{3.1}$$

with φ_1 and φ_2 are the back values. Equation (3.1) in matrix-vector form is equivalent to

$$(I - A)Y_{n+1,n+2} = hBF_{n+1,n+2} + \xi_{n+1,n+2}. \tag{3.2}$$

By setting $I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $Y_{n+1,n+2} = \begin{bmatrix} y_{n+1} \\ y_{n+2} \end{bmatrix}$, $A = \begin{bmatrix} 0 & \theta_1 \\ \theta_2 & 0 \end{bmatrix}$, $B = \begin{bmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{bmatrix}$, $F_{n+1,n+2} = \begin{bmatrix} f_{n+1} \\ f_{n+2} \end{bmatrix}$, and $\xi_{n+1,n+2} = \begin{bmatrix} \varphi_1 \\ \varphi_2 \end{bmatrix}$, (3.1) is simplified as

$$\widehat{f}_{n+1,n+2} = (I - A)Y_{n+1,n+2} - hBF_{n+1,n+2} - \xi_{n+1,n+2} = 0. \tag{3.3}$$

Newton iteration is performed to the system $\widehat{f}_{n+1,n+2} = 0$, by taking the analogous form

$$Y_{n+1,n+2}^{(i+1)} - Y_{n+1,n+2}^{(i)} = -\left[(I - A) - hB\frac{\partial F}{\partial Y}\left(Y_{n+1,n+2}^{(i)}\right)\right]^{-1}(I - A)Y_{n+1,n+2}^{(i)} - hBF\left(Y_{n+1,n+2}^{(i)}\right) - \xi_{n+1,n+2}, \tag{3.4}$$

where $J_{n+1,n+2} = (\partial F/\partial Y)(Y_{n+1,n+2}^{(i)})$ is the Jacobian matrix of F with respect to Y. Equation (3.4) is separated to three different matrices denoted as

$$E_{1,2}^{(i+1)} = Y_{n+1,n+2}^{(i+1)} - Y_{n+1,n+2}^{(i)}, \tag{3.5}$$

$$\widehat{A} = (I - A) - hB\frac{\partial F}{\partial Y}\left(Y_{n+1,n+2}^{(i)}\right), \tag{3.6}$$

$$\widehat{B} = (I - A)Y_{n+1,n+2}^{(i)} - hBF\left(Y_{n+1,n+2}^{(i)}\right) - \xi_{n+1,n+2}. \tag{3.7}$$

Two-stage Newton iteration is now introduced in order to find the approximate solution to (1.1). Thus, the corresponding linear system to be solved is $\widehat{A}E_{1,2}^{(i+1)} = \widehat{B}$. According to Jackson in [9], evaluating the Jacobian $J_{n+1,n+2}$ and LU factorization of $\widehat{A}$ require the most computation time during the operations. As a result, two strategies which are based on the step size are applied to ensure the efficiency of the method:

(i) no calculation of new matrices $\widehat{A}$ and $\widehat{B}$ if the step size h remains as previous step size. Hence, there is no new calculation of the Jacobian matrix $J_{n+1,n+2}$,

(ii) the matrices $\widehat{A}$ and $\widehat{B}$ are updated with new evaluation of the Jacobian matrix $J_{n+1,n+2}$ for every occurrence of changing step size.

As a result, this paper agrees with [10] that this method has a higher tendency in saving computational time. Unlike most modern solvers that incorporated full Newton iterations, 5oBBDF method offers refined strategy for reevaluating the Jacobian matrix $J_{n+1,n+2}$.

3.1. Stability Conditions for 5oBBDF

In this section, we provide the conditions for the stability of 5oBBDF method as in (2.5). To begin with, we include some definitions to support the practical criterion for a method to be useful in solving ODEs.

Definition 3.1. A method is said to be zero stable if the roots of the polynomial $\rho(z) = \pi(z, 0)$ satisfy the condition $z_1 = 1 \leq |z_2|,\ z_2 \neq 1$.

Definition 3.2. A method is said to be absolute stable in a region R for a given $h\lambda$ if for that $h\lambda$, all the roots r_s of the stability polynomial $\pi(r, h\lambda) = \rho(r) - h\lambda\sigma(r) = 0$ satisfy $|r_s| < 1$, $s = 1, 2, \ldots, k$.

The stability polynomial, $R(t, \widehat{h})$, associated with the method of (2.5) is given by $\det(At^2 - Bt - C)$, while the absolute stability region of this method in the $h\lambda$ plane is

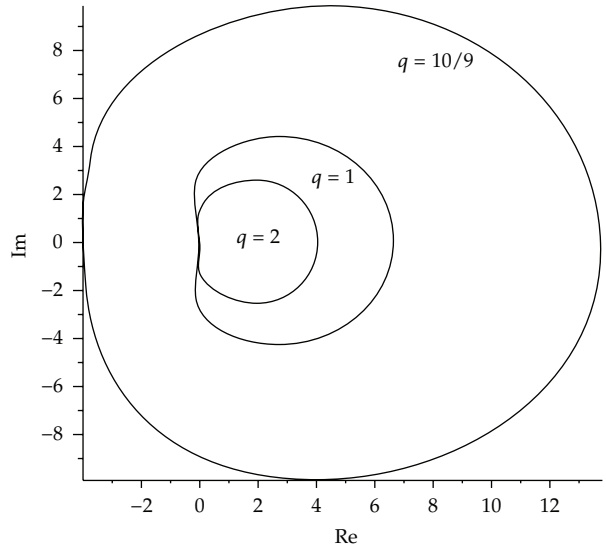

Figure 2: Stability regions when $q = 1$, $q = 2$, and $q = 10/19$.

determined by solving $\det(At^2 - Bt - C = 0)$. Below are the absolute stability regions in the method of (2.5) for different step size selections: $(q = 1)$ $(q = 2)$ and $(q = 10/19)$, respectively,

$$
\begin{aligned}
R\left(t,\widehat{h}\right) &= \frac{197}{125}t^4 - \frac{42}{25}t^4h - \frac{153}{125}t^3 - \frac{9}{25}t^2 + \frac{72}{125}t^4h^2 - \frac{252}{125}t^3h - \frac{18}{125}t^2h + \frac{1}{125}t = 0,\\
R\left(t,\widehat{h}\right) &= \frac{91}{46}t^4 - \frac{441}{184}t^4h - \frac{173}{92}t^3 - \frac{289}{2944}t^2 + \frac{45}{46}t^4h^2 - \frac{1155}{736}t^3h - \frac{3}{92}t^2h + \frac{1}{2944}t = 0,\\
R\left(t,\widehat{h}\right) &= \frac{1393273}{999685}t^4 - \frac{5298909}{3998740}t^4h - \frac{42149421}{199937000}t^3 - \frac{544951521}{420920000}t^2 + \frac{393588}{999685}t^4h^2\\
&\quad - \frac{1398292623}{399874000}t^3h - \frac{753768}{1315375}t^2h + \frac{47045881}{420920000}t = 0.
\end{aligned}
\tag{3.8}
$$

To determine for zero stable, we substitute $\widehat{h} = h\lambda = 0$ to (3.8). We will have the following equations for each step size selection mentioned previously:

$$
\begin{aligned}
\frac{197}{125}t^4 - \frac{153}{125}t^3 - \frac{9}{25}t^2 + \frac{1}{125}t &= 0,\\
\frac{91}{46}t^4 - \frac{173}{92}t^3 - \frac{289}{2944}t^2 + \frac{1}{2944}t &= 0,\\
\frac{1393273}{999685}t^4 - \frac{42149421}{199937000}t^3 - \frac{544951521}{420920000}t^2 + \frac{47045881}{420920000}t &= 0.
\end{aligned}
\tag{3.9}
$$

Hence, the roots for three different step size selections obtained by using Maple are listed below

(1) $t = -.24414201370,\ t = 0.02079175991,$ and $t = 1,$

(2) $t = -0.052708171410,\ t = 0.003257621961,$ and $t = 1,$

(3) $t = -0.93455113330,\ t = 0.08581158625,$ and $t = 1.$

Since all of the roots have modulus less than or equal to 1, the method (2.5) when $(q = 1)$ $(q = 2)$ and $(q = 10/19)$ is zero stable.

The stability region was given by the set of points determined by the boundary $t = e^{i\theta}$, $0 \leq \theta \leq 2\pi$. The stability region is obtained by finding the region for which $|t| < 1$. Figure 2 shows the stability for the cases $(q = 1)$ $(q = 2)$ and $(q = 10/19)$, respectively. The stability regions lie outside the closed region for each case.

Based on Figure 2, the 5th-order BBDF possesses the region absolute stability, which contains almost whole of the half-plane $\mathrm{Re}(h\lambda) < 0$.

3.2. Choosing Step Size

The importance in choosing the step size is to achieve reduction in computation time and number of iterations. Therefore, this paper proposed three basic strategies for the step size selections. For each successful step, the step size remains constant $(q = 1)$ or increased by a factor of 1.9 $(q = 10/19)$. Inversely, when a fail step occurs, the next step size will be halved of the previous step size $(q = 2)$. The user initially will have to provide an error tolerance limit, TOL on any given step, and obtain the local truncation error (LTE) for each iteration. The LTE is obtained from

$$\mathrm{LTE} = y_{n+2}^{(k+1)} - y_{n+2}^{(k)}, \quad k = 4, \tag{3.10}$$

where $y_{n+2}^{(k+1)}$ is the $(k+1)$th order method, and $y_{n+2}^{(k)}$ is the kth order method.

The successful step is dependent on the condition LTE < TOL. If this condition fails, the values of y_{n+1}, y_{n+2} are rejected, and the current step is reiterated with step size selection $(q = 2)$. On the contrary, the step size increment for each successful step is defined as

$$h_{\mathrm{new}} = c \times h_{\mathrm{old}} \times \left(\frac{\mathrm{TOL}}{\mathrm{LTE}}\right)^{1/p}. \tag{3.11}$$

And if $h_{\mathrm{new}} > 1.9 \times h_{\mathrm{old}}$, then $h_{\mathrm{new}} = 1.9 \times h_{\mathrm{old}}$. Where c is the safety factor, p is the order of the method, while h_{old} and h_{new} are the step size from previous and current blocks, respectively. In this paper, c is set to be 0.8.

4. MATLAB's Stiff Numerical Solvers

MATLAB offers several numerical solvers to solve either stiff or nonstiff ordinary differential equations. These built-in numerical solvers are capable in approximating solutions to almost any system of differential equations [2]. As far as this paper is concerned, we are interested in solving stiff ODEs. For comparison purposes, this paper considers only ode15s and

ode23s. This is because both of the methods deal with stiff differential equations based on backward differentiation formulas (BDFs) and on a modified Rosenbrock formula of order 2, respectively. Therefore, a fair comparison is obtained, and the discussions are made easy.

5. Numerical Results

We carry out numerical experiments to compare the performance of the 5th-order BBDF method with stiff ODE solvers in MATLAB mentioned above. Three sets of stiff test problems aroused from problems in physics are tested for the purpose of elucidating the difference in numerical results obtained. These test problems are performed under different conditions of error tolerances—(a) 10^{-2}, (b) 10^{-4}, and (c) 10^{-6}, and the error trend is defined by

$$\text{Error} = \left| y_{\text{approximate}} - y_{\text{actual}} \right|. \tag{5.1}$$

The test problems and solution are listed below

(1) basic circuit problem is

$$y' = -20y + 24, \quad y(0) = 0, \quad 0 \le x \le 10, \tag{5.2}$$

with solution $y(x) = 6/5 - (6/5)e^{-20x}$,

(2) oscillatory problems in physics: Torsion spring oscillator with dry and viscous friction [7]

$$\begin{aligned} y_1' &= 998y_1 + 1998y_2, \quad y_1(0) = 1, \quad 0 \le x \le 10, \\ y_2' &= -999y_1 - 1999y_2, \quad y_2(0) = 2, \end{aligned} \tag{5.3}$$

with solution

$$\begin{aligned} y_1(x) &= 2e^{-x} - e^{-1000x}, \\ y_2(x) &= -e^{-x} + e^{-1000x}, \end{aligned} \tag{5.4}$$

eigenvalues, $\lambda = -1,\ -1000$,

(3) Physical problem [11] is

$$\begin{aligned} y_1' &= -20y_1 - 0.25y_2 - 19.75y_3, \quad y_1(0) = 1, \quad 0 \le x \le 10, \\ y_2' &= 20y_1 - 20.25y_2 + 0.25y_3, \quad y_2(0) = 0, \\ y_3' &= 20y_1 - 19.75y_2 - 0.25y_3, \quad y_3(0) = -1, \end{aligned} \tag{5.5}$$

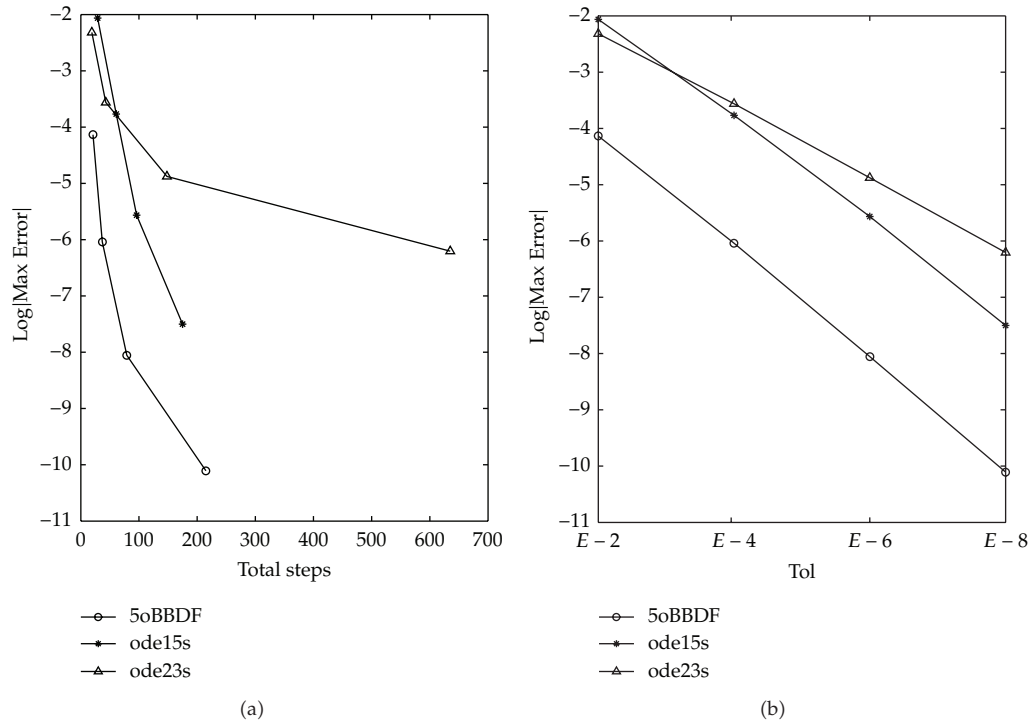

Figure 3: Efficiency curves for problem (1.1).

with solution

$$y_1(x) = 0.5\left[e^{-0.5x} + e^{-20x}(\cos 20x + \sin 20x)\right],$$
$$y_2(x) = 0.5\left[e^{-0.5x} - e^{-20x}(\cos 20x - \sin 20x)\right], \tag{5.6}$$
$$y_3(x) = -0.5\left[e^{-0.5x} + e^{-20x}(\cos 20x - \sin 20x)\right],$$

eigenvalues, $\lambda = -0.5,\ -20 \pm 20i$.

From Table 1, among the three methods tested, our method, 5oBBDF, requires the shortest execution time for each given tolerance level. On top of that, this method also gives the smallest maximum error and average error. From Figure 3, we can see that 5oBBDF gives the lowest maximum error for every tolerance level. When we look at the convergence of the methods compared in Figure 3, our method comparably converges faster with minimum number of steps taken.

From Table 2, once again, 5oBBDF requires the shortest execution time for each given tolerance level. From the error trends described in Figure 4, we can see that the maximum errors for 5oBBDF are the smallest for each TOL as compared to ode15s and ode23s. So the convergence of the method, 5oBBDF is found to converge fastest among the methods tested.

From Table 3, similarly, 5oBBDF gives the shortest execution time for each given tolerance level. This method is also completed with lesser total steps except when TOL is 10^{-2}.

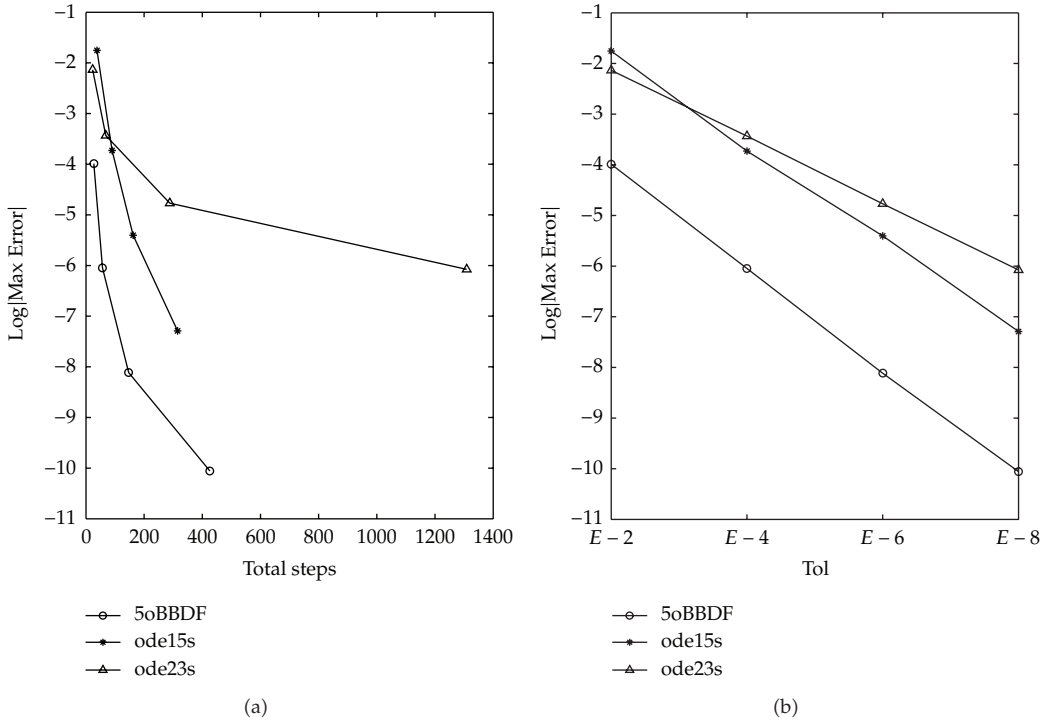

Figure 4: Efficiency curves for problem (2.1).

Table 1: Numerical results for problem (1.1).

TOL	Method	TS	AVEE	MAXE	Time
	5oBBDF	21	1.09720e − 005	7.3154e − 005	0.01067
10^{-2}	ode15s	29	1.400e − 003	8.700e − 003	0.0313
	ode23s	19	1.200e − 003	4.800e − 003	0.2813
	5oBBDF	37	1.88948e − 007	9.1173e − 007	0.02234
10^{-4}	ode15s	61	3.4444e − 005	1.7046e − 004	0.0313
	ode23s	43	1.2509e − 004	2.7400e − 004	0.0156
	5oBBDF	79	1.72338e − 009	8.8279e − 009	0.009139
10^{-6}	ode15s	96	9.0543e − 007	2.7175e − 006	0.0469
	ode23s	148	8.0523e − 006	1.3309e − 005	0.0313

Table 2: Numerical results for problem (2.1).

TOL	Method	STs	AVEE	MAXE	Time
	5oBBDF	27	3.02605e − 005	1.0244e − 004	0.011902
10^{-2}	ode15s	38	3.100e − 003	1.760e − 002	0.0469
	ode23s	23	2.300e − 003	7.300e − 003	0.1406
	5oBBDF	57	3.15965e − 007	1.0632e − 006	0.064613
10^{-4}	ode15s	90	4.3445e − 005	1.8659e − 004	0.0156
	ode23s	68	1.9392e − 004	3.6837e − 004	0.0625
	5oBBDF	147	2.10931e − 009	1.0440e − 008	0.009559
10^{-6}	ode15s	162	8.3966e − 007	3.9569e − 006	0.0625
	ode23s	288	9.9147e − 006	1.7039e − 005	0.0781

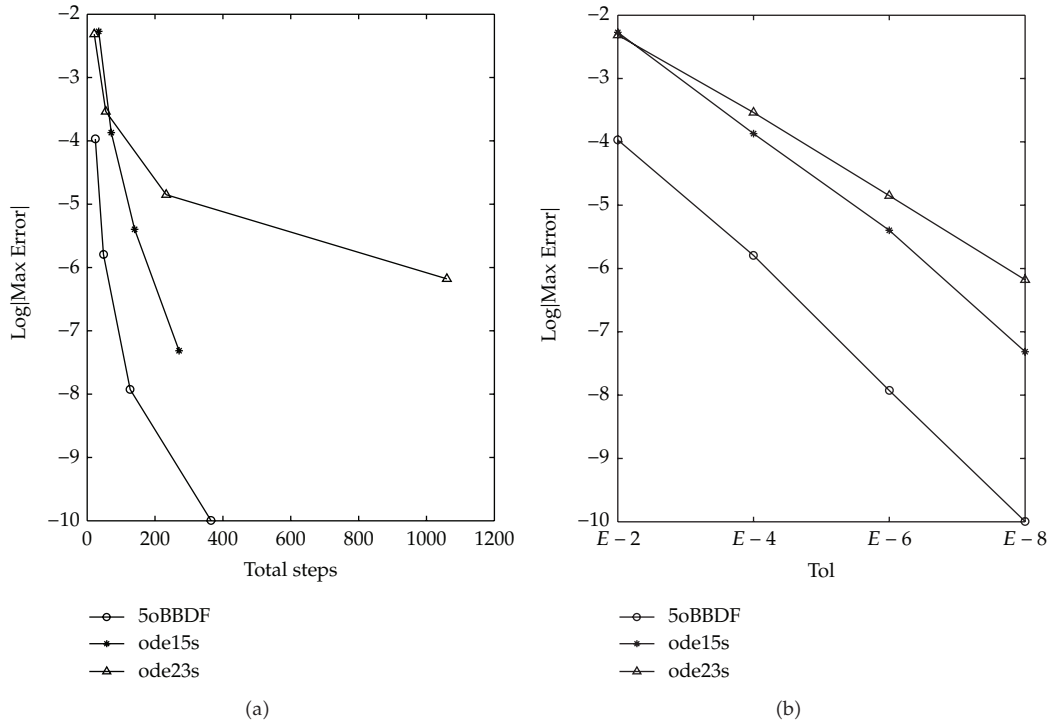

Figure 5: Efficiency curves for problem (2.2).

Table 3: Numerical results for problem (2.2).

TOL	Method	TSs	Average error	MAXE	Time
10^{-2}	5oBBDF	24	$3.16762e-005$	$2.1568e-004$	0.01150
	ode15s	34	$1.090e-002$	$1.090e-002$	0.0313
	ode23s	21	$1.004e-003$	$6.400e-003$	0.0313
10^{-4}	5oBBDF	48	$4.39634e-007$	$1.8652e-006$	0.033346
	ode15s	71	$5.7501e-005$	$2.0274e-004$	0.0469
	ode23s	55	$1.3953e-004$	$3.5375e-004$	0.0156
10^{-6}	5oBBDF	127	$1.41711e-008$	$2.0629e-009$	0.009460
	ode15s	140	$8.3688e-007$	$4.0059e-006$	0.0469
	ode23s	233	$7.2824e-006$	$1.7023e-005$	0.1250

The error trends for all of the methods tested are depicted in Figure 5. The error trend shows that 5oBBDF converged fastest with smallest maximum error for each TOL as compared to ode15s and ode23s.

6. Conclusion

For all problems tested, it is proven that, the 5th-order variable step BBDF has outperformed the ode15s and ode23s in terms of average errors as well as maximum errors. It also managed to reduce the number of total steps taken in most of the cases especially when TOL is less

than 10^{-2}. When we tested the convergence of the methods compared in this paper, the error trend worked as the identifier for the fastest error of the method approaches zero. Figures 3, 4, and 5 illustrate how fast the error of 5th-order variable step BBDF converged as compared to the other two methods used. As a result, we found that this method converges fastest for all problems tested. In terms of computation timewise, it gave lesser values for all of the test problems. Therefore, we can conclude that the 5th-order variable step BBDF is one compatible alternative solver for solving stiff ordinary differential equations which mostly arise in engineering and applied sciences.

Abbreviations

TS:	The total number of steps taken
TOL:	The initial value for the local error estimate
MAXE:	The maximum error
AVEE:	The average error
METHOD:	The method used
TIME:	The total execution time (seconds)
5oBBDF:	5th-order variable step BBDF
e:	Error for each point of approximations.

References

[1] G. D. Byrne and A. C. Hindmarsh, "Stiff ODE solvers: a review of current and coming attractions," *Journal of Computational Physics*, vol. 70, no. 1, pp. 1–62, 1987.

[2] C. F. Quo and M. D. Wang, "Quantitative comparison of numerical solvers for models of oscillatory biochemical systems," in *Proceedings of the 2nd International Multi-Symposiums on Computer and Computational Sciences (IMSCCS '07)*, pp. 44–51, University of Iowa, Iowa City, IA, USA, August 2007.

[3] C. W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1971.

[4] J. G. Verwer and D. Simpson, "Explicit methods for stiff ODEs from atmospheric chemistry," *Applied Numerical Mathematics*, vol. 18, no. 1–3, pp. 413–430, 1995.

[5] G. D. Byrne, A. C. Hindmarsh, K. R. Jackson, and H. G. Brown, "A comparison of two ODE codes: gear and EPISODE," *Computers and Chemical Engineering*, vol. 1, no. 2, pp. 125–131, 1977.

[6] W. H. Enright, T. E. Hull, and B. Lindberg, "Comparing numerical methods for stiff systems of ODEs," *BIT Numerical Mathematics*, vol. 15, no. 1, pp. 10–48, 1975.

[7] Z. B. Ibrahim, M. B. Suleiman, and K. I. Othman, "Fixed coefficients block backward differentiation formulas for the numerical solution of stiff ordinary differential equations," *European Journal of Scientific Research*, vol. 21, no. 3, pp. 508–520, 2008.

[8] Z. B. Ibrahim, K. I. Othman, and M. B. Suleiman, "Variable step size block backward differentiation formula for solving stiff odes," in *Proceedings of the World Congress on Engineering*, vol. 2, pp. 785–789, London, UK, 2007.

[9] K. R. Jackson, "The numerical solution of large systems of stiff IVPs for ODEs," *Applied Numerical Mathematics*, vol. 20, no. 1-2, pp. 5–20, 1996.

[10] A. S. Mahmood, L. Casasús, and W. Al-Hayani, "The decomposition method for stiff systems of ordinary differential equations," *Applied Mathematics and Computation*, vol. 167, no. 2, pp. 964–975, 2005.

[11] J. C. Polking and D. Arnold, *Ordinary Differential Equation Using MATLAB*, Upper Saddle River, NJ, USA, Prentice Hall, 2nd edition, 1999.