Journal of Mathematics and Statistics: 4(3): 174-180, 2008
ISSN: 1549-3644

Block Methods based on Newton Interpolations for Solving Special Second Order Ordinary Differential Equations Directly

[1]Yap Lee Ken, [2]Fudziah Ismail, [2]Mohamed Suleiman and [2]Suriah Md. Amin
[1]Institute for Mathematical Research, University Putra Malaysia,
43400, Serdang, Selangor, Malaysia
[2]Departmen of Mathematics, University Putra Malaysia, 43400, Serdang, Selangor, Malaysia

Abstract: This study focused mainly on the derivation of the 2 and 3-point block methods with constant coefficients for solving special second order ordinary differential equations directly based on Newton-Gregory backward interpolation formula. The performance of the new methods was compared with the conventional 1-point method using a standard set of test problems. Numerical results were presented to illustrate the effectiveness of the methods in terms of total number of steps taken, maximum error and execution time. The results suggested a significant improvement in efficiency of the r-point block method. AMS Subject Classification: 65L05.

Key word: Initial value problems, special second order ordinary differential equations, block method

INTRODUCTION

Special second order Ordinary Differential Equations (ODEs) arises naturally in describing mechanics and electrical systems, wave oscillations and a variety of other physical problems. Such equations can be written in the form:

$$\begin{aligned} & y'' = f(x, y), \qquad x \geq 0 \\ & y(x_0) = y_0, \quad y'(x_0) = y'_0 \end{aligned} \tag{1}$$

The easiest way to obtain the numerical solution of Eq. 1 is to reduce it to a system of first order ODEs twice the dimension. However some computational advantage can be gained if we can use methods specially designed to solve Eq. 1 directly, such methods can be seen in Dahlquist[1], Van Der Houwen[9] and Sommeijer[8] and El-Mikkawy and Rahmo[2]. These methods compute the numerical solution at one point at a time.

In this study we developed methods that can compute the numerical solution at more than one point at a time; such method is called block method. This method can be seen in Omar[6] whereby he developed multiblock methods based on the divided difference interpolation for the solution of general second order equation $y'' = f(x, y, y')$. Lee[4] proposed block methods based on backward difference interpolation for first order ODEs $y' = f(x, y)$ and Majid[5] developed the method based on Lagrange interpolation polynomial for general higher order ODEs $y^d = f(x, y, y', y'', \ldots, y^{d-1})$.

In this study, we are going to derive the block method for solving special second order ODEs directly based on Newton-Gregory backward interpolation formula.

MATERIALS AND METHODS

Derivation of explicit r-point block method: In r-point block method, the interval is divided into series of blocks with each block containing r points; r new values are obtained concurrently at each iteration of algorithm. Let $x_{n+i} = x_n + ih$, $i = 1, 2, \ldots$, $\forall n \in [a, b]$. Therefore:

$$\int_{x_n}^{x_{n+i}} \int_{x_n}^{x} y''(x)\, dx\, dx = \int_{x_n}^{x_{n+i}} \int_{x_n}^{x} f(x, y)\, dx\, dx$$

Integrating Eq. 1 twice, we obtain the formula:

$$\begin{aligned} & y(x_n + ih) - y(x_n) = ihy'(x_n) + \\ & \int_{x_n}^{x_{n+i}} (x_n + ih - x) f(x, y(x))\, dx \end{aligned} \tag{2}$$

In order to eliminate the first derivative of y(x), write the formula (2) with h replaced by -h and add the two expressions:

Corresponding Author: Fudziah Ismail, Department of Mathematics, University Putra Malaysia, 43400, Serdang,Malaysia

$$y(x_n+ih)-2y(x_n)+y(x_n-ih)=$$
$$\int_{x_n}^{x_{n+i}}(x_n+ih-x)(f(x)+f(2x_n-x))\,dx$$

which can be written as:

$$y(x_{n+i})-2y(x_n)+y(x_{n-i})= \int_{x_n}^{x_{n+i}}(x_{n+i}-x)(f(x)+f(2x_n-x))\,dx \qquad (3)$$

Define the interpolation polynomial $P_{k,n}(x)$ which interpolates $f(x,y)$ at the k back values as follows:

$$P_{k,n}(x_n+sh)=\sum_{q=0}^{k-1}(-1)^q\binom{-s}{q}\nabla^q f_n \qquad (4)$$

$$P_{k,n}(x_n-sh)=\sum_{q=0}^{k-1}(-1)^q\binom{s}{q}\nabla^q f_n \qquad (5)$$

where, $s=\frac{x-x_n}{h}$.

Approximating f(x) and $f(2x_n-x)$ with (4) and (5), (3) is now used in the form:

$$y(x_{n+i})=2y(x_n)-y(x_{n-i})+ \int_{x_n}^{x_{n+i}}\sum_{q=0}^{k-1}(-1)^q(x_{n+i}-x)\left[\binom{-s}{q}+\binom{s}{q}\right]\nabla^q f_n\,dx \qquad (6)$$

Replacing $dx=h\,ds$ and changing the limit of integration in (6) gives:

$$y(x_{n+i})=2y(x_n)-y(x_{n-i})+ \int_0^i\sum_{q=0}^{k-1}(-1)^q(i-s)h\left[\binom{-s}{q}+\binom{s}{q}\right]\nabla^q f_n\,h\,ds$$

which leads to:

$$y_{n+i}=2y_n-y_{n-i}+h^2\sum_{q=0}^{k-1}\omega_{i,q}\nabla^q f_n$$

Where:

$$\omega_{i,q}=(-1)^q\int_0^i(i-s)\left[\binom{-s}{q}+\binom{s}{q}\right]ds$$

Let $V_i(t)$ be the generating function of the coefficients $\omega_{i,q}$ defined as follows:

$$V_i(t)=\sum_{q=0}^{\infty}\omega_{i,q}t^q$$
$$=\sum_{q=0}^{\infty}(-t)^q\int_0^i(i-s)\left[\binom{-s}{q}+\binom{s}{q}\right]ds$$
$$=\int_0^i(i-s)\sum_{q=0}^{\infty}\left((-t)^q\binom{-s}{q}+(-t)^q\binom{s}{q}\right)ds$$
$$=\int_0^i(i-s)\left((1-t)^{-s}+(1-t)^s\right)ds$$
$$=\frac{\left((1-t)^i-1\right)^2}{(1-t)^i\left[\ln(1-t)\right]^2}$$

Hence:

$$\left(\sum_{q=0}^{\infty}\omega_{i,q}t^q\right)\left[\ln(1-t)\right]^2=-2+\frac{1}{(1-t)^i}+(1-t)^i$$

Using the expansions, gives:

$$(\omega_{i,0}+\omega_{i,1}t+\omega_{i,2}t^2+\ldots)\left(t^2+\frac{2}{3}h_2t^3+\frac{2}{4}h_3t^4+\ldots\right)$$
$$=\left[\frac{1}{2}(i)(i+1)\,t^2+\frac{1}{6}(i)(i+1)(i+2)\,t^3+\ldots\right]+$$
$$\left[\frac{1}{2}(i-1)(i)\,t^2-\frac{1}{6}(i-2)(i-1)(i)\,t^3+\ldots\right]$$

By comparing coefficients yields:

$$\omega_{i,0}=i^2$$
$$\omega_{i,q}=\frac{(i+q+1)(i+q)\ldots(i+1)(i)}{(q+2)!}+$$
$$(-1)^q\frac{(i-q-1)(i-q)\ldots(i-1)(i)}{(q+2)!}-2\sum_{r=0}^{q-1}\frac{\omega_r h_{q+1-r}}{q+2-r}$$

with $q=1,2,\ldots$.

By using the integration coefficients and Newton-Gregory backward difference formula, we derive the explicit methods of order four as shown.

Explicit 2-point block method:

$$\begin{bmatrix}1&0\\0&1\end{bmatrix}\begin{bmatrix}y_{n+1}\\y_{n+2}\end{bmatrix}=\begin{bmatrix}-1&2\\0&2\end{bmatrix}\begin{bmatrix}y_{n-1}\\y_n\end{bmatrix}+\begin{bmatrix}0&0\\0&-1\end{bmatrix}\begin{bmatrix}y_{n-3}\\y_{n-2}\end{bmatrix}+$$
$$h^2\left(\begin{bmatrix}-\frac{5}{12}&\frac{7}{6}\\-\frac{20}{3}&\frac{20}{3}\end{bmatrix}\begin{bmatrix}f_{n-1}\\f_n\end{bmatrix}+\begin{bmatrix}-\frac{1}{12}&\frac{1}{3}\\-\frac{4}{3}&\frac{16}{3}\end{bmatrix}\begin{bmatrix}f_{n-3}\\f_{n-2}\end{bmatrix}\right)$$

Explicit 3-Point block method:

$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}\begin{bmatrix}y_{n+1}\\y_{n+2}\\y_{n+3}\end{bmatrix}=\begin{bmatrix}0&-1&2\\-1&0&2\\0&0&2\end{bmatrix}\begin{bmatrix}y_{n-2}\\y_{n-1}\\y_n\end{bmatrix}+\begin{bmatrix}0&0&0\\0&0&0\\0&0&-1\end{bmatrix}\begin{bmatrix}y_{n-5}\\y_{n-4}\\y_{n-3}\end{bmatrix}$$
$$+h^2\left(\begin{bmatrix}\frac{1}{3}&-\frac{5}{12}&\frac{7}{6}\\\frac{16}{3}&-\frac{20}{3}&\frac{20}{3}\\27&-\frac{135}{4}&\frac{45}{2}\end{bmatrix}\begin{bmatrix}f_{n-2}\\f_{n-1}\\f_n\end{bmatrix}+\begin{bmatrix}0&0&-\frac{1}{12}\\0&0&-\frac{4}{3}\\0&0&-\frac{27}{4}\end{bmatrix}\begin{bmatrix}f_{n-5}\\f_{n-4}\\f_{n-3}\end{bmatrix}\right)$$

Derivation of implicit R-Point block method: Let $x_{n+i}=x_n+ih$, $i=1,2,\ldots$, $\forall n\in[a,b]$. Therefore:

$$\int_{x_n}^{x_{n+i}}\int_{x_n}^{x} y''(x)\,dx\,dx=\int_{x_n}^{x_{n+i}}\int_{x_n}^{x} f(x,y)\,dx\,dx$$

Using the same approach as in the previous section, we obtain:

$$\begin{aligned}&y(x_{n+i})-2y(x_n)+y(x_{n-i})=\\&\int_{x_n}^{x_{n+i}}(x_{n+i}-x)\big(f(x)+f(2x_n-x)\big)dx\end{aligned}\qquad(7)$$

The interpolating polynomials which interpolate $f(x,y)$ at the set of points (x_{n+i-m},f_{n+i-m}) for $m=0,1,2,\ldots,k$ as follows:

$$P_{k,n+i}\big(x_n+(s+i)h\big)=\sum_{q=0}^{k}(-1)^m\binom{-s}{q}\nabla^m f_{n+i}\qquad(8)$$

and

$$P_{k,n+i}\big(x_n+(-s-i)h\big)=\sum_{q=0}^{k}(-1)^m\binom{s+2i}{q}\nabla^m f_{n+i}\qquad(9)$$

where $s=\dfrac{x-x_{n+i}}{h}$.

Approximating f(x) and $f(2x_n-x)$ with (8) and (9), (7) is now used in the form:

$$\begin{aligned}&y(x_{n+i})=2y(x_n)-y(x_{n-i})+\\&\int_{x_n}^{x_{n+i}}\sum_{q=0}^{k}(-1)^q(x_{n+i}-x)\left[\binom{-s}{q}+\binom{s+2i}{q}\right]\nabla^q f_{n+i}\,dx\end{aligned}\qquad(10)$$

Replacing dx = hds and changing the limit of integration in (10) gives:

$$\begin{aligned}&y(x_{n+i})=2y(x_n)-y(x_{n-i})+\\&\int_{-i}^{0}\sum_{q=0}^{k}(-1)^q(-s)h\left[\binom{-s}{q}+\binom{s+2i}{q}\right]\nabla^q f_{n+i}\,h\,ds\end{aligned}$$

which leads to:

$$y_{n+i}=2y_n-y_{n-i}+h^2\sum_{q=0}^{k}\upsilon_{i,q}\nabla^q f_{n+i}$$

where

$$\upsilon_{i,q}=(-1)^q\int_{-i}^{0}(-s)\left[\binom{-s}{q}+\binom{s+2i}{q}\right]ds$$

In order to obtain a useful recurrence relation for the coefficients $\upsilon_{i,q}$, the method of generating function is used. Let the generating function $L_i(t)$ be defined as follows:

$$\begin{aligned}L_i(t)&=\sum_{q=0}^{\infty}\upsilon_{i,q}t^q\\&=\sum_{q=0}^{\infty}(-t)^q\int_{-i}^{0}(-s)\left[\binom{-s}{q}+\binom{s+2i}{q}\right]ds\\&=\int_{-i}^{0}(-s)\sum_{q=0}^{\infty}\left((-t)^q\binom{-s}{q}+(-t)^q\binom{s+2i}{q}\right)ds\\&=\int_{-i}^{0}(-s)\left((1-t)^{-s}+(1-s)^{s+2i}\right)ds\\&=\frac{\left((1-t)^i-1\right)^2}{\left[\ln(1-t)\right]^2}\end{aligned}$$

Hence:

$$\left(\sum_{q=0}^{\infty}\upsilon_{i,q}t^q\right)\left[\ln(1-t)\right]^2=1-2(1-t)^i+(1-t)^{2i}$$

Using the expansions, gives:

$$\begin{aligned}&\left(\upsilon_{i,0}+\upsilon_{i,1}t+\upsilon_{i,2}t^2+\ldots\right)\left(t^2+\frac{2}{3}h_2t^3+\frac{2}{4}h_3t^4+\ldots\right)\\&=\left[-(i)(i-1)t^2+\frac{1}{3}(i)(i-1)(i-2)t^3+\ldots\right]\\&\quad+\left[(i)(2i-1)t^2-\frac{1}{3}(i)(2i-1)(2i-2)t^3+\ldots\right]\end{aligned}$$

By comparing coefficients yields:

$$\upsilon_{i,0} = i^2$$

$$\upsilon_{i,q} = (-1)^{q+1}\frac{2(i-q-1)(i-q)\ldots(i-1)(i)}{(q+2)!} + (-1)^q\frac{2(2i-q-1)(2i-q)\ldots(2i-1)(i)}{(q+2)!} - 2\sum_{r=0}^{q-1}\frac{\upsilon_r h_{q+1-r}}{q+2-r}$$

with $q = 1, 2, \ldots$.

With the integration coefficients and Newton-Gregory backward difference formula, we derive the implicit block methods of order four.

Implicit 2-Point block method:

$$\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\begin{bmatrix}y_{n+1}\\y_{n+2}\end{bmatrix} = \begin{bmatrix}-1 & 2\\0 & 2\end{bmatrix}\begin{bmatrix}y_{n-1}\\y_n\end{bmatrix} + \begin{bmatrix}0 & 0\\0 & -1\end{bmatrix}\begin{bmatrix}y_{n-3}\\y_{n-2}\end{bmatrix} + h^2\left(\begin{bmatrix}\frac{19}{240} & 0\\\frac{16}{15} & \frac{1}{15}\end{bmatrix}\begin{bmatrix}f_{n+1}\\f_{n+2}\end{bmatrix} + \begin{bmatrix}\frac{7}{120} & \frac{17}{20}\\\frac{16}{15} & \frac{26}{15}\end{bmatrix}\begin{bmatrix}f_{n-1}\\f_n\end{bmatrix} + \begin{bmatrix}-\frac{1}{240} & \frac{1}{60}\\0 & \frac{1}{15}\end{bmatrix}\begin{bmatrix}f_{n-3}\\f_{n-2}\end{bmatrix}\right)$$

Implicit 3-Point block method:

$$\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}y_{n+1}\\y_{n+2}\\y_{n+3}\end{bmatrix} = \begin{bmatrix}0 & -1 & 2\\-1 & 0 & 2\\0 & 0 & 2\end{bmatrix}\begin{bmatrix}y_{n-2}\\y_{n-1}\\y_n\end{bmatrix} + \begin{bmatrix}0 & 0 & 0\\0 & 0 & 0\\0 & 0 & -1\end{bmatrix}\begin{bmatrix}y_{n-5}\\y_{n-4}\\y_{n-3}\end{bmatrix}$$
$$+ h^2\left(\begin{bmatrix}\frac{19}{240} & 0 & 0\\\frac{16}{15} & \frac{1}{15} & 0\\\frac{621}{40} & -\frac{117}{20} & \frac{117}{80}\end{bmatrix}\begin{bmatrix}f_{n+1}\\f_{n+2}\\f_{n+3}\end{bmatrix} + \begin{bmatrix}\frac{1}{60} & \frac{7}{120} & \frac{17}{20}\\\frac{1}{15} & \frac{16}{15} & \frac{26}{15}\\0 & \frac{657}{80} & -\frac{207}{20}\end{bmatrix}\begin{bmatrix}f_{n-2}\\f_{n-1}\\f_n\end{bmatrix} + \begin{bmatrix}0 & 0 & -\frac{1}{240}\\0 & 0 & 0\\0 & 0 & 0\end{bmatrix}\begin{bmatrix}f_{n-5}\\f_{n-4}\\f_{n-3}\end{bmatrix}\right)$$

Test problems: To illustrate the effectiveness of the method, the existing 1-point method which was also based on Newton interpolation[3], the 2-point and 3-point block methods with order k = 4 are used to solve the following problems numerically.

Problem 1: $y'' = -\omega^2 y + (\omega^2 - 1)\sin x$, $y(0) = 1$, $y'(0) = 1 + \omega$, $\omega = 10$, $x \in [0, 2]$.
Exact solution: $y(x) = \cos(\omega x) + \sin(\omega x) + \sin x$ [8].

Problem 2: $y_1'' = -4x^2 y_1 - \frac{2y_2}{r}$, $y_1(x_0) = 0$, $y_1'(x_0) = -\sqrt{2\pi}$, $y_2'' = -4x^2 y_2 + \frac{2y_1}{r}$, $y_2(x_0) = 1$, $y_2'(x_0) = 0$, with $r^2 = y_1^2 + y_2^2$, $x \in \left[\sqrt{\frac{\pi}{2}}, 10\right]$.
Exact solution: $y_1(x) = \cos(x^2)$, $y_2(x) = \sin(x^2)$ [2].

Problem 3: $z'' + z = 0.001e^{ix}$, $z(0) = 1$, $z'(0) = 0.9995i$, $z \in C$, $z(x) = u(x) + iv(x)$, $u, v \in \Re$, $u(x) = \cos(x) + 0.0005x \sin(x)$, $v(x) = \sin(x) - 0.0005x \cos(x)$.

We choose to solve the equivalent real problems $u'' + u = 0.001\cos(x)$, $u(0) = 1$, $u'(0) = 0$, $v'' + v = 0.001\sin(x)$, $v(0) = 0$, $v'(0) = 0.9995$, $x \in [0, 40\pi]$ [7].

It is the "almost periodic" problem with the theoretical solution represents motion on a perturbed circular orbit in the complex plane.

RESULTS AND DISCUSSION

Table 1-12 show the performance comparison of new methods and the existing method in terms of the total number of steps taken, maximum error and execution time. The performance of the new code is also compared to code by Omar[6] as well. The new code is used to solve the special second order ODEs $y'' = f(x, y)$ directly whereas the code by Omar[6] is for the general second order ODEs $y'' = f(x, y, y')$. The notations used in the Table 1-12 are as follows:

h	=	Step size used
METHOD	=	Method employed
TSTEP	=	Total number of steps taken to obtain the solution
MAXERR	=	Magnitude of the maximum error of the computed solution
TIME	=	Execution time taken in microseconds (ms)
E1PN	=	Existing explicit 1-point method in[3]
E2PBN	=	The new explicit 2-point 1-block method
E3PBN	=	The new explicit 3-point 1-block method
I1PN	=	Existing implicit 1-point method in[3]
I2PBN	=	The new implicit 2-point 1-block method
I3PBN	=	The new implicit 3-point 1-block method
E1PO	=	Explicit 1-point method in Omar[6]
E2PBO	=	Explicit 2-point block method in Omar[6]
E3PBO	=	Explicit 3-point block method in Omar[6]
I1PO	=	Implicit 1-point method in Omar[6]
I2PBO	=	Implicit 2-point block method in Omar[6]
I3PBO	=	Implicit 3-point block method in Omar[6]

Table 1: Performance comparison between E1PN, E2PBN and E3PBN for solving Problem 1

h	METHOD	TSTEP	MAXERR	TIME
10^{-2}	E1PN	200	1.04867(-2)	690
	E2PBN	102	1.01941(-2)	670
	E3PBN	69	9.17238(-3)	656
10^{-3}	E1PN	2000	1.15576(-4)	4422
	E2PBN	1002	9.12913(-5)	4140
	E3PBN	669	6.68441(-5)	4015
10^{-4}	E1PN	20000	1.16663(-6)	43710
	E2PBN	10002	9.17045(-7)	41003
	E3PBN	6669	6.67303(-7)	40142
10^{-5}	E1PN	200000	1.19013(-8)	435537
	E2PBN	100002	9.16555(-9)	409839
	E3PBN	66669	6.70794(-9)	400231

Table 2: Performance comparison between E1PN, E2PBN and E3PBN for solving Problem 2

h	METHOD	TSTEP	MAXERR	TIME
10^{-2}	E1PN	875	2.28442(-3)	2790
	E2PBN	439	1.64625(1)	2394
	E3PBN	294	2.57005	2426
10^{-3}	E1PN	8747	6.11465(-6)	25036
	E2PBN	4375	4.80547(-6)	21490
	E3PBN	2918	4.40915(-6)	20988
10^{-4}	E1PN	87467	6.09090(-8)	249919
	E2PBN	43735	4.78556(-8)	212804
	E3PBN	29158	3.48057(-8)	208928
10^{-5}	E1PN	874669	2.53863(-9)	2498769
	E2PBN	437336	9.65937(-10)	2130344
	E3PBN	291559	8.22541(-10)	2090620

Table 3: Performance comparison between E1PN, E2PBN and E3PBN for solving Problem 3

h	METHOD	TSTEP	MAXERR	TIME
10^{-2}	E1PN	12567	1.16486(-4)	40318
	E2PBN	6285	9.25841(-5)	38059
	E3PBN	4191	6.80643(-5)	37577
10^{-3}	E1PN	125664	1.16492(-6)	399714
	E2PBN	62834	9.15580(-7)	377070
	3PBN	890	6.66222(-7)	372453
10^{-4}	E1PN	1256638	1.18440(-8)	4006061
	E2PBN	628321	9.36450(-9)	3767598
	3PBN	418882	6.70706(-9)	3729101
10^{-5}	E1PN	12566371	2.87949(-8)	40027952
	2PBN	6283187	1.69135(-8)	37641623
	3PBN	4188793	6.94793(-9)	37251793

Table 4: Performance comparison between I1PN, I2PBN and I3PBN for solving Problem 1

h	METHOD	TSTEP	MAXERR	TIME
10^{-2}	I1PN	200	9.16935(-3)	980
	I2PBN	102	7.24986(-3)	945
	I3PBN	69	5.36151(-3)	920
10^{-3}	I1PN	2000	9.93031(-5)	5942
	I2PBN	1002	7.48576(-5)	5672
	I3PBN	669	5.02747(-5)	5630
10^{-4}	I1PN	20000	1.00021(-6)	59156
	I2PBN	10002	7.50543(-7)	56323
	I3PBN	6669	5.00710(-7)	55158
10^{-5}	I1PN	200000	1.01023(-8)	591520
	I2PBN	100002	7.55203(-9)	561423
	I3PBN	66669	5.01627(-9)	549786

Table 5: Performance comparison between I1PN, I2PBN and I3PBN for solving Problem 2

h	METHOD	TSTEP	MAXERR	TIME
10^{-2}	I1PN	875	2.52652(-3)	3655
	I2PBN	439	2.48192(-3)	3469
	I3PBN	294	7.12478(-3)	3478
10^{-3}	I1PN	8747	2.39947(-5)	33067
	I2PBN	4375	3.10920(-5)	31730
	I3PBN	2918	5.98919(-6)	31220
10^{-4}	I1PN	87467	3.51616(-7)	329383
	I2PBN	43735	3.71797(-7)	315150
	I3PBN	29158	2.61012(-8)	310986
10^{-5}	I1PN	874669	2.52969(-9)	3296280
	I2PBN	437336	2.14713(-9)	3152669
	I3PBN	291559	7.74633(-10)	3100176

Table 6: Performance comparison between I1PN, I2PBN and I3PBN for solving Problem 3

h	METHOD	TSTEP	MAXERR	TIME
10^{-2}	I1PN	12567	1.16487(-4)	54832
	I2PBN	6285	7.51354(-5)	52640
	I3PBN	4191	5.06011(-5)	51687
10^{-3}	I1PN	125664	1.16492(-6)	544661
	I2PBN	62834	7.49128(-7)	520697
	I3PBN	41890	4.99765(-7)	510612
10^{-4}	I1PN	1256638	1.18440(-8)	5439182
	I2PBN	628321	7.56804(-9)	5198387
	I3PBN	418882	5.13160(-9)	5099863
10^{-5}	I1PN	12566371	2.87949(-8)	54426979
	I2PBN	6283187	1.69537(-8)	52040186
	I3PBN	4188793	7.02614(-9)	51013287

Table 7: Performance comparison between E2PBN, E3PBN and E2PBO, E3PBO for solving Problem 1

h	METHOD	TSTEP	MAXERR
10^{-2}	E2PBN	102	1.01941(-2)
	E2PBO	102	4.69426(-2)
	E3PBN	69	9.17238(-3)
	E3PBO	69	4.17711(-2)
10^{-3}	E2PBN	1002	9.12913(-5)
	E2PBO	1002	4.98878(-3)
	E3PBN	669	6.68441(-5)
	E3PBO	669	4.98097(-3)
10^{-4}	E2PBN	10002	9.17045(-7)
	E2PBO	10002	4.99914(-4)
	E3PBN	6669	6.67303(-7)
	E3PBO	6669	4.99906(-4)
10^{-5}	E2PBN	100002	9.16555(-9)
	E2PBO	100002	4.99992(-5)
	E3PBN	66669	6.70794(-9)
	E3PBO	66669	4.99992(-5)

Table 8: Performance comparison between E2PBN, E3PBN and E2PBO, E3PBO for solving Problem .2

h	METHOD	TSTEP	MAXERR
10^{-2}	E2PBN	439	1.64625(1)
	E2PBO	439	3.31501(-2)
	E3PBN	294	2.57005
	E3PBO	294	8.66280(-2)
10^{-3}	E2PBN	4375	4.80547(-6)
	E2PBO	4375	1.09973(-3)
	E3PBN	2918	4.40915(-6)
	E3PBO	2918	1.09963(-3)
10^{-4}	E2PBN	43735	4.78556(-8)
	E2PBO	43735	1.10003(-4)
	E3PBN	29158	3.48057(-8)
	E3PBO	29158	1.10003(-4)
10^{-5}	E2PBN	437336	9.65937(-10)
	E2PBO	437336	1.10003(-5)
	E3PBN	291559	8.22541(-10)
	E3PBO	291559	1.10003(-5)

Table 9: Performance comparison between E2PBN, E3PBN and E2PBO, E3PBO for solving Problem 3

h	METHOD	TSTEP	MAXERR
10^{-2}	E2PBN	6285	9.25841(-5)
	E2PBO	6285	4.99211(-3)
	E3PBN	4191	6.80643(-5)
	E3PBO	4191	4.98425(-3)
10^{-3}	E2PBN	62834	9.15580(-7)
	E2PBO	62834	4.99497(-4)
	E3PBN	41890	6.66222(-7)
	E3PBO	41890	4.99489(-4)
10^{-4}	E2PBN	628321	9.36450(-9)
	E2PBO	628321	4.99501(-5)
	E3PBN	418882	6.70706(-9)
	E3PBO	418882	4.99506(-5)
10^{-5}	E2PBN	6283187	1.69135(-8)
	E2PBO	6283187	4.99698(-6)
	E3PBN	4188793	6.94793(-9)
	E3PBO	4188793	5.00226(-6)

Table 10: Performance comparison between I2PBN, I3PBN and I2PBO, I3PBO for solving Problem 1

h	METHOD	TSTEP	MAXERR
10^{-2}	I2PBN	102	7.24986(-3)
	I2PBO	102	1.42942(-2)
	I3PBN	69	5.36151(-3)
	I3PBO	69	1.43029(-2)
10^{-3}	I2PBN	1002	7.48576(-5)
	I2PBO	1002	1.50554(-3)
	I3PBN	669	5.02747(-5)
	I3PBO	669	1.50553(-3)
10^{-4}	I2PBN	10002	7.50543(-7)
	I2PBO	10002	1.51305(-4)
	I3PBN	6669	5.00710(-7)
	I3PBO	6669	1.51305(-4)
10^{-5}	I2PBN	100002	7.55203(-9)
	I2PBO	100002	1.51381(-5)
	I3PBN	66669	5.01627(-9)
	I3PBO	66669	1.51381(-5)

Table 11: Performance comparison between I2PBN, I3PBN and I2PBO, I3PBO for solving Problem 2

h	METHOD	TSTEP	MAXERR
10^{-2}	I2PBN	439	2.48192(-3)
	I2PBO	439	3.28838(-3)
	I3PBN	294	7.12478(-3)
	I3PBO	294	3.28838(-3)
10^{-3}	I2PBN	4375	3.10920(-5)
	I2PBO	4375	3.32651(-4)
	I3PBN	2918	5.98919(-6)
	I3PBO	2918	3.32651(-4)
10^{-4}	I2PBN	43735	3.71797(-7)
	I2PBO	43735	3.33029(-5)
	I3PBN	29158	2.61012(-8)
	I3PBO	29158	3.33029(-5)
10^{-5}	I2PBN	437336	2.14713(-9)
	I2PBO	437336	3.33049(-6)
	I3PBN	291559	7.74633(-10)
	I3PBO	291559	3.33048(-6)

The maximum error is defined as $\text{MAXERR} = \max_{1 \le i \le \text{TSTEP}} \left(\left| y(x_i) - y_i \right| \right)$

Table 12: Performance comparison between I2PBN, I3PBN and I2PBO, I3PBO for solving Problem 3

h	METHOD	TSTEP	MAXERR
10^{-2}	I2PBN	6285	7.51354(-5)
	I2PBO	6285	1.51236(-3)
	I3PBN	4191	5.06011(-5)
	I3PBO	4191	1.51236(-3)
10^{-3}	I2PBN	62834	7.49128(-7)
	I2PBO	62834	1.51238(-4)
	I3PBN	41890	4.99765(-7)
	I3PBO	41890	1.51237(-4)
10^{-4}	I2PBN	628321	7.56804(-9)
	I2PBO	628321	1.51239(-5)
	I3PBN	418882	5.13160(-9)
	I3PBO	418882	1.51243(-5)
10^{-5}	I2PBN	6283187	1.69537(-8)
	I2PBO	6283187	1.51435(-6)
	I3PBN	4188793	7.02614(-9)
	I3PBO	4188793	1.51964(-6)

CONCLUSION

The block and non-block methods based on Newton-Gregory backward interpolation formula are compared in terms of three parameters namely the total number of steps, the accuracy and the execution time. As the step size decreases, 2-point block and 3-point block methods reduce the total number of steps taken to almost one half and one third compared to 1-point method. These results are expected since the r-point block methods calculate the values of y at r point simultaneously compared to non-block methods.

The maximum error for explicit 3-point block method is slightly smaller compared to explicit 2-point block method which in turn smaller compared to explicit 1-point method for various values of h. In general, the implicit methods are more accurate than the explicit counterparts. The implicit block methods have better accuracy than the implicit non-block methods.

Both the explicit and implicit block methods seem to be superior to the non-block counterparts in term of the execution time taken to obtain the solution. The implicit methods require more time to generate the solution since it involved extra computations. As the step size becomes finer, the advantage of using block methods is more obvious.

Table 7-12 show the advantage of using the new codes over codes by Omar[6] in term of accuracy. The increase in the accuracy is more obvious as the step size decreases. Thus, it can be concluded that the performance of both the explicit and implicit block methods based on Newton-Gregory backward interpolation formula is better in terms of the total number of steps taken, accuracy and execution time compared to the non-block methods and is more accurate compared to the existing block methods.

REFERENCES

1. Dahlquist, 1978. On accuracy and unconditional stability of linear multistep methods for second order differential equations. BIT Numeric. Math., 18: 133-136. DOI: 10.1007/BF01931689
2. El-Mikkawy, M. and El-D. Rahmo, 2003. A new optimized non-FSAL embedded Runge-Kutta-Nystrom algorithm of orders 6 and 4 in 6 stages. Applied Math. Comput., 145: 33-43. DOI: 10.1016/S0096-3003(02)00436-8
3. Hairer, E., S.P. Norsett and G. Wanner, 1993. Solving Ordinary Differential Equations I (Nonstiff Problems). 2nd Edn., Springer-Verlag, New York, pp: 528. ISBN: 3540566708.
4. Lee, L.S., M. Suleiman, Z. Omar and F. Ismail, 2000. Two and three-point explicit block methods for first order ordinary differential equations. Proceedings of the International Conference on Mathematics and its Applications in the New Millenium, July 18-19, Renaissance Palm Garden Hotel Putra Jaya, Malaysia, pp: 481-490.
5. Majid, Z. and M. Suleiman, 2007. Implementation of four-point fully implicit block methods for solving ordinary differential equations. Applied Math. Comput., 184: 514-522. http://cat.inist.fr/?aModele=afficheN&cpsidt=18519329.
6. Omar, Z., M. Suleiman, M.Y. Saman and D.J. Evans, 2002. Parallel r-point explicit block method for solving second order ODEs directly. Int. J. Comput. Math., 79: 289-298. http://cat.inist.fr/?aModele=afficheN&cpsidt=13652392.
7. Papageorgiou, G., I.Th. Famwlis and Ch. Tsitouras, 1998. A P-stable singly diagonally implicit runge-kutta-nystrom method. Numeric. Algorithm, 17: 345-353. DOI: 10.1023/A:1016644726305.
8. Sallam, S. and A.A. Karaballi, 1999. A continuous implicit nystrom method for solving ordinary second order initial value problems. Int. J. Comput. Math., 72: 189-198.
9. Van Der Houwen, P.J. and B.P. Sommeijer, 1987. Predictor-corrector for periodic second order initial value problems. IMA J. Numeric. Anal., 7: 407-422. http://cat.inist.fr/?aModele=afficheN&cpsidt=7432630.