VARIABLE-STEP VARIABLE-ORDER 3-STAGE HERMITE–BIRKHOFF ODE SOLVER OF ORDER 5 TO 15

TRUONG NGUYEN-BA, HEMZA YAGOUB, YI LI, AND RÉMI VAILLANCOURT

To appear in the Canadian Applied Mathematics Quarterly

Abstract. Variable-step variable-order 3-stage Hermite–Birkhoff (HB) methods HB(p)3 of order $p = 5$ to 15 are constructed for solving non-stiff differential equations. Forcing a Taylor expansion of the numerical solution to agree with an expansion of the true solution leads to multistep and Runge–Kutta type order conditions which are reorganized into linear confluent Vandermonde-type systems of HB type. Fast algorithms are developed for solving these systems in O(p^2) operations to obtain HB interpolation polynomials in terms of generalized Lagrange basis functions. The stability regions of the HB methods have a remarkably good shape. The order and stepsize of these methods are controlled by four local error estimators. When programmed in C++, HB(p)3 uses less CPU time than Dormand–Prince DP(8,7)13M in solving costly problems at stringent tolerance.

Résumé. On construit un solveur d'Hermite–Birkhoff (HB) à 3 étages à pas et ordre variables, nommé HB(p)3, d'ordre $p = 5$ à 15, pour systèmes d'équations différentielles non raides $y' = f(x, y)$, $y(x_0) = y_0$. En identifiant les développements de Taylor tronqués des solutions exacte et numérique on obtient des conditions d'ordre du type Runge–Kutta qu'on réorganise en systèmes linéaires de Vandermonde confluents de type HB qu'on résout en O(p^2) opérations au moyen de nouveaux algorithmes rapides qui donnent lieu à des polynômes d'interpolation HB sur une base de fonctions de Lagrange généralisées. La forme des domaines de stabilité absolue des méthodes HB est remarquable. On contrôle l'ordre et le pas au moyen de 4 estimateurs de l'erreur locale. Programmé en C++, HB(p)3 résout des prolèmes coûteux à tolérance serrée plus rapidement que Dormand–Prince DP(8,7)13M.

1. Introduction

There is a large variety of variable step variable order (VSVO) methods designed to solve nonstiff and stiff systems of first-order differential equations (ODEs). This introduction intends to put some perspective among the several approaches and methods. Gear advocated a quasi-constant step size implementation in DIFSUB

1991 *Mathematics Subject Classification.* Primary: 65L06; Secondary: 65D05, 65D30.

Key words and phrases. general linear method for non-stiff ODE's, Hermite–Birkhoff method, maximum global error, number of function evaluations, non-stiff DETEST problems, confluent Vandermonde-type systems, C++.

This work was supported in part by the Natural Sciences ands Engineering Research Council of Canada and the Centre de recherches mathématiques of the Université de Montréal.

[19]. This software works with a constant step size until a change of step size is necessary or clearly advantageous. Then a continuous extension is used to get approximations to the solution at previous points in an equally spaced mesh. This was largely because constant mesh spacing is very helpful when solving stiff problems. Another possibility is fixed leading coefficient, which is seen in Petzold's popular code DASSL [28]. Finally, the actual mesh can be chosen by the code as done in Matlab's `ode113`. This is the equivalent of a PECE Adams formula in contrast with the Adams–Moulton formulas of DIFSUB and DASSL. In this paper a fully variable step size implementation is used with actual mesh chosen by the code equivalent of a PECE Adams formula.

A more basic point about the implementation of a method is the choice of the form. The present method uses a Lagrangian form and much of the paper is devoted to computing the coefficients efficiently. It might be acknowledged that there are pros and cons about the form; such matters are discussed by Gear with the Nordsieck form [27], Krogh with modified divided differences [23], and Brayton *et al.* [6] with Lagrangian form. The Lagrangian form has the virtue of simpilicity. It should be pointed out that the cost depends on the order of the formula. Remark 1 in a later section connects the computation of coefficients for various forms. Krogh [23] was concerned about the effects of roundoff at stringent tolerances, which is an implicit assumption of this paper, and he concluded that divided differences is a good way to minimize the effect of roundoff. Working with this form does involve manipulation of vectors of the length of the number of equations. These manipulations can be largely vectorized nowadays. Sofroniou and Spaletta [34] were even more concerned about this for extrapolation because the solvers might be used to achieve extraordinary accuracies in Mathematica.

The code DVDQ [22] and `ode113` [32, 2] implement Adams–Bashforth–Moulton multistep methods in PECE modes. On the other hand DIFSUB and DASSL implement Adams–Moulton formulas. Although these codes predict with Adams–Bashforth formulas, they iterate to completion so that, in principle, it does not matter what predictor was used. Extrapolation is another way of achieving high accuracy. Deuflhard [13] is the person most responsible for drawing attention to the value of this approach but the codes of [20, Section II.9] are probably the most visible. Indeed, NDSolve of Mathematica is based on those codes. Hairer *et al.* [20] consider extrapolation to be the best way of solving problems very accurately. That, in fact, is why it is used in Mathematica—they want to provide users with more-or-less whatever accuracy they require. Extrapolation can be viewed as a variable order Runge–Kuttta scheme. Following an important comparison of Enright and Hull [16] on fixed order Runge–Kutta codes and extrapolation codes, Shampine and Baca [30] argue that a fixed order (7,8) pair is more effective than extrapolation at accuracies common in scientific computation, but high order always wins if one asks for enough accuracy. Finally, Taylor series can be very attractive in VSVO implementation. It has been an excellent choice in astronomical calculations [3]. For general problems one can see the work of Corliss and Chang [12]. Lastly, an interpolant [15, 11] for approximating the solution between mesh

points is an important matter because two natural continuous extensions spring to mind, depending on the global smoothness desired. It is worth remarking that a principal reason for the Matlab ODE Suite was to provide solvers with an event location facility Indeed, this is the main reason why the Suite does not contain an extrapolation code nor a high order Runge–Kutta pair. In retrospect, one should acknowledge that Fehlberg's (7,8) pair is the one that drew attention to the value of high order RK pair.

General linear methods for solving nonstiff systems of first-order ordinary differential equations of the form

$$y' = f(x, y), \qquad y(x_0) = y_0, \tag{1}$$

can be thought of as multistep methods with off-step points or as Runge–Kutta methods with backstep points. Like multistep methods, they use information prior to the last step and, like Runge–Kutta methods, they use derivative evaluations at points partway through the current step. The link between the two types of methods is that values at off-step points are obtained by means of predictors which use values at previous points. It has been noted in [10] that general linear methods incorporate function evaluations at off-step points in order to reduce the number of backsteps without lowering the order.

In this paper, we construct new 3-stage VSVO general linear methods of order $p = 5, 6, \ldots, 15$ which use the values y_n, y_{n-1} and f_{n-j}, $j = 0, 1, \ldots, p-4$. Since these methods use Hermite–Birkhoff (HB) interpolation polynomials they will be called HB(p)3 methods and the family of such methods will be designated by HB(5-15)3.

It was found experimentally that increasing the number of backstep points is more efficient in increasing the accuracy of HB methods than increasing the number of off-step points. It was also found that increased speed is generally achieved by higher order HB methods.

The performance of HB(5-15)3 and DP(8,7)13M [29] was compared on several problems often used to test higher order ODE solvers. It was seen that HB(5-15)3 uses lower CPU time in solving costly equations.

Other HB methods of order 9, 10 and 11 have been studied in [25]. An efficient HB Obrechkoff 3-stage 6-step method of order 14 using $(d/dx)f(x, y(x))$ has been studied in [26].

In Section 2 we introduce a new family of general HB(p)3 methods of order $p = 5, 6, \ldots, 15$. Order conditions are listed in Section 3. In Section 4 general HB(p)3 are represented in terms of Vandermonde-type systems. In Section 5 symbolic elementary matrices are constructed as functions of the parameters of the methods in view of factoring the coefficient matrices of Vandermonde-type systems. In Section 6 a family of particular variable step HB(5-15)3 is defined by fixing the off-step points and is constructed in Section 7. Section 8 considers the regions of absolute stability and principal local truncation coefficients of constant step HB(5-15)3. Section 9 deals with the step and order control. In Section 10 three criteria are used to compare the numerical performance of the methods considered

in this paper. Appendix A lists the algorithms. Appendix B describes the Matlab programming for Matlab users.

2. General variable step HB(p)3 of order p

The following terminology will be useful. An HB(p)3 method is said to be a **general** variable-step HB method if its backstep and off-step points are variable parameters. If the off-step points are fixed, the method is said to be a **particular** variable-step method. If the stepsize is constant, and hence the backsteps and off-steps are fixed parameters, the method is said to be a **constant-step** method.

A general 3-stage HB(p)3 of order $p = 5, 6, \ldots, 15$ requires the following four formulae to perform the integration step from x_n to x_{n+1}, where, for simplicity, $c_1 = 0$ is used in the summations.

($\mathrm{P}_\mathbf{2}$) A Hermite–Birkhoff polynomial of degree $p-2$ is used as predictor P_2 to obtain y_{n+c_2} to order $p-2$,

$$y_{n+c_2} = \alpha_{20} y_n + \alpha_{21} y_{n-1} + h_{n+1} \left(a_{21} f_n + \sum_{j=1}^{p-4} \beta_{2j} f_{n-j} \right). \tag{2}$$

($\mathrm{P}_\mathbf{3}$) A Hermite–Birkhoff polynomial of degree $p-1$ is used as predictor P_3 to obtain y_{n+c_3} to order $p-2$,

$$y_{n+c_3} = \alpha_{30} y_n + \alpha_{31} y_{n-1} + h_{n+1} \left(a_{31} f_{n+c_1} + a_{32} f_{n+c_2} + \sum_{j=1}^{p-4} \beta_{3j} f_{n-j} \right). \tag{3}$$

(IF) A Hermite–Birkhoff polynomial of degree p is used as integration formula IF to obtain y_{n+1} to order p:

$$y_{n+1} = \alpha_{10} y_n + \alpha_{11} y_{n-1} + h_{n+1} \left(\sum_{j=1}^{2} b_{1j} f_{n+c_j} + b_{13} f_{n+c_3} + \sum_{j=1}^{p-4} \beta_{1j} f_{n-j} \right). \tag{4}$$

($\mathrm{P}_\mathbf{4}$) An Adams–Moulton corrector of order $p-2$ is used as P_4 to control the stepsize, h_{n+2}, and obtain $\widetilde{y}_{n+1}$ to order $p-2$,

$$\widetilde{y}_{n+1} = y_n + h_{n+1} \left(a_{41} f_n + a_{43} f_{n+1} + \sum_{j=1}^{p-4} \beta_{4j} f_{n-j} \right). \tag{5}$$

For the 3-stage $(p-3)$-step methods considered in this paper, the off-step points satisfy the following Runge–Kutta type simplifying conditions:

$$c_i = \sum_{j=1}^{i-1} a_{ij} + B_i(1), \qquad i = 2, 3, \tag{6}$$

where

$$B_i(j) = \alpha_{i1} \frac{\eta_2^j}{j!} + \sum_{\ell=1}^{p-4} \left[\beta_{i\ell} \frac{\eta_{\ell+1}^{j-1}}{(j-1)!} \right], \qquad j = 1, 2, \ldots, p, \quad i = 2, 3. \tag{7}$$

and

$$\eta_j = -\frac{1}{h_{n+1}}\,(x_n - x_{n+1-j}) = -\frac{1}{h_{n+1}}\sum_{i=0}^{j-1} h_{n-i}, \qquad j = 2,3,\ldots,p-3. \tag{8}$$

In the sequel, η_j will be frequently used without explicit reference to (8).

3. Order conditions of general HB(p)3

By forcing a Taylor expansion of the numerical solution produced by formulae HB(5-15)3 to agree with an expansion of the true solution we obtain multistep and Runge–Kutta type order conditions that must be satisfied by general HB(p)3 methods of order $p = 5,\ldots,15$.

As in similar search for ODE solvers, we impose the following simplifying assumptions:

$$\sum_{j=0}^{1} \alpha_{ij} = 1, \qquad i = 2,3, \tag{9}$$

$$\sum_{j=1}^{i-1} a_{ij}c_j^k + k!B_i(k+1) = \frac{1}{k+1}\,c_i^{k+1}, \qquad \begin{cases} i = 2,3, \\ k = 0,1,2,\ldots,p-3. \end{cases} \tag{10}$$

There remain three sets of equations to be solved:

$$\sum_{i=0}^{1} \alpha_{1i} = 1, \tag{11}$$

$$\sum_{i=1}^{3} b_{1i}c_i^k + k!B_1(k+1) = \frac{1}{k+1}, \qquad k = 0,1,\ldots,p-1, \tag{12}$$

$$\sum_{i=2}^{3} b_{1i}\left[\sum_{j=1}^{i-1} a_{ij}\frac{c_j^{p-2}}{(p-2)!} + B_i(p-1)\right] + B_1(p) = \frac{1}{p!}, \tag{13}$$

where the backstep parts, $B_1(j)$, are defined by

$$B_1(j) = \alpha_{11}\frac{\eta_2^j}{j!} + \sum_{i=1}^{p-4}\beta_i\frac{\eta_{i+1}^{j-1}}{(j-1)!}, \qquad j = 1,\ldots,p+1. \tag{14}$$

4. Vandermonde-type formulation of general HB(p)3

4.1. **Integration formula** IF**.** The $(p+1)$-vector of reordered coefficients of the integration formula IF in (4),

$$\boldsymbol{u}^1 = [\alpha_{10}, b_{11}, b_{12}, b_{13}, \beta_{11}, \beta_{12}, \ldots, \beta_{1,p-4}, \alpha_{11}]^T,$$

is the solution of the confluent Vandermonde-type system of order conditions

$$M^1\boldsymbol{u}^1 = \boldsymbol{r}^1, \tag{15}$$

where

$$M^1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & \cdots & 1 & \eta_2 \\ 0 & 0 & c_2 & c_3 & \eta_2 & \cdots & \eta_{p-3} & \frac{\eta_2^2}{2!} \\ 0 & 0 & \frac{c_2^2}{2!} & \frac{c_3^2}{2!} & \frac{\eta_2^2}{2!} & \cdots & \frac{\eta_{p-3}^2}{2!} & \frac{\eta_2^3}{3!} \\ \vdots & & & & & & & \vdots \\ 0 & 0 & \frac{c_2^{p-1}}{(p-1)!} & \frac{c_3^{p-1}}{(p-1)!} & \frac{\eta_2^{p-1}}{(p-1)!} & \cdots & \frac{\eta_{p-3}^{p-1}}{(p-1)!} & \frac{\eta_2^p}{p!} \end{bmatrix} \tag{16}$$

and $\boldsymbol{r}^1 = r_1(1:p+1)$ has components

$$r_1(i) = 1/(i-1)!, \qquad i = 1, 2, \ldots, p+1.$$

The leading error term of IF is

$$\left[\alpha_{11}\frac{\eta_2^{p+1}}{(p+1)!} + b_{12}\frac{c_2^p}{p!} + b_{13}\frac{c_3^p}{p!} + \sum_{j=1}^{p-4}\beta_{1j}\frac{\eta_{j+1}^p}{p!} - \frac{1}{(p+1)!}\right] h_{n+1}^{p+1} y_n^{p+1}.$$

4.2. **Predictor** P_2**.** The $(p-1)$-vector of reordered coefficients of predictor P_2 in (2),

$$\boldsymbol{u}^2 = [\alpha_{20}, a_{21}, \beta_{21}, \beta_{22}, \ldots, \beta_{2,p-4}, \alpha_{21}]^T,$$

is the solution of the system of order conditions

$$M^2\boldsymbol{u}^2 = \boldsymbol{r}^2, \tag{17}$$

where

$$M^2 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 1 & 1 & \cdots & 1 & \eta_2 \\ 0 & 0 & \eta_2 & \cdots & \eta_{p-3} & \frac{\eta_2^2}{2!} \\ 0 & 0 & \frac{\eta_2^2}{2!} & \cdots & \frac{\eta_{p-3}^2}{2!} & \frac{\eta_2^3}{3!} \\ \vdots & & & & & \vdots \\ 0 & 0 & \frac{\eta_2^{p-3}}{(p-3)!} & \cdots & \frac{\eta_{p-3}^{p-3}}{(p-3)!} & \frac{\eta_2^{p-2}}{(p-2)!} \end{bmatrix} \tag{18}$$

and $\boldsymbol{r}^2 = r_2(1:p-1)$ has components

$$r_2(i) = c_2^{i-1}/(i-1)!, \qquad i = 1, 2, \ldots, p-1.$$

A truncated Taylor expansion of the right-hand side of (2) about x_n gives

$$\sum_{j=0}^{p+1} S_2(j) h_{n+1}^j y_n^{(j)}$$

with coefficients

$$S_2(j) = M^2(j+1, 1:p-1)\boldsymbol{u}^2 = r_2(j+1) = \frac{c_2^j}{j!}, \qquad j = 0, 1, \ldots, p-2,$$

$$S_2(j) = \alpha_{21}\frac{\eta_2^j}{j!} + \sum_{i=1}^{p-4}\beta_{2i}\frac{\eta_{i+1}^{j-1}}{(j-1)!}, \qquad j = p-1, p, p+1.$$

We note that P_2 is of order $p-2$ since it satisfies the order conditions

$$\sum_{i=0}^{1} \alpha_{2i} = 1,$$
$$S_2(j) = c_2^j/j!, \qquad j = 1, \ldots, p-2,$$

and its leading error term is

$$\left[S_2(p-1) - \frac{c_2^{p-1}}{(p-1)!} \right] h_{n+1}^{p-1} y_n^{(p-1)}.$$

4.3. **Predictor** P_3**.** The p-vector of reordered coefficients of predictor P_3 in (3),

$$\boldsymbol{u}^3 = [\alpha_{30}, a_{31}, a_{32}, \beta_{31}, \beta_{32}, \ldots, \beta_{3,p-4}, \alpha_{31}]^T,$$

is the solution of the system of order conditions

$$M^3 \boldsymbol{u}^3 = \boldsymbol{r}^3, \tag{19}$$

where

$$M^3 = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 1 & 1 & 1 & \cdots & 1 & \eta_2 \\ 0 & 0 & c_2 & \eta_2 & \cdots & \eta_{p-3} & \frac{\eta_2^2}{2!} \\ 0 & 0 & \frac{c_2^2}{2!} & \frac{\eta_2^2}{2!} & \cdots & \frac{\eta_{p-3}^2}{2!} & \frac{\eta_2^3}{3!} \\ \vdots & & & & & & \vdots \\ 0 & 0 & \frac{c_2^{p-2}}{(p-2)!} & \frac{\eta_2^{p-2}}{(p-2)!} & \cdots & \frac{\eta_{p-3}^{p-2}}{(p-2)!} & \frac{\eta_2^{p-1}}{(p-1)!} \end{bmatrix}. \tag{20}$$

The first $p-1$ components of $\boldsymbol{r}^3 = r_3(1:p)$ are

$$r_3(i) = c_3^{i-1}/(i-1)!, \qquad i = 1, 2, \ldots, p-1,$$

and the pth component is

$$r_3(p) = \frac{1}{b_{13}} \left[\frac{1}{p!} - b_{12} S_2(p-1) - B_1(p) \right],$$

which corresponds to the RK order conditions (13).

A truncated Taylor expansion of the right-hand side of (3) about x_n gives

$$\sum_{j=0}^{p+1} S_3(j) h_{n+1}^j y_n^{(j)}$$

with coefficients

$$S_3(j) = M^3(j+1, 1:p)\boldsymbol{u}^3 = r_3(j+1) = \frac{c_3^j}{j!}, \qquad j = 0, 1, \ldots, p-2,$$
$$S_3(j) = \alpha_{31} \frac{\eta_2^j}{j!} + a_{32} S_2(j-1) + \sum_{i=1}^{p-4} \beta_{3i} \frac{\eta_{i+1}^{j-1}}{(j-1)!}, \qquad j = p-1, p, p+1.$$

4.4. **Step control predictor** P_4**.** The $(p-2)$-vector of reordered coefficients of P_4 in (5),

$$\boldsymbol{u}^4 = [a_{41}, a_{43}, \beta_{41}, \beta_{42}, \ldots, \beta_{4,p-4}]^T,$$

is the solution of the system of order conditions:

$$M^4 \boldsymbol{u}^4 = \boldsymbol{r}^4, \tag{21}$$

where

$$M^4 = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & c_3 & \eta_2 & \cdots & \eta_{p-3} \\ 0 & \frac{c_3^2}{2!} & \frac{\eta_2^2}{2!} & \cdots & \frac{\eta_{p-3}^2}{2!} \\ \vdots & & & & \vdots \\ 0 & \frac{c_3^{p-3}}{(p-3)!} & \frac{\eta_2^{p-3}}{(p-3)!} & \cdots & \frac{\eta_{p-3}^{p-3}}{(p-3)!} \end{bmatrix} \tag{22}$$

and $\boldsymbol{r}^4 = r_4(1:p-2)$ has components

$$r_4(i) = 1/i!, \qquad i = 1, 2, \ldots, p-2.$$

The solutions $\boldsymbol{u}^\ell$, $\ell = 1, 2, 3, 4$, form generalized Lagrange basis functions for representing the HB interpolation polynomials.

5. Symbolic construction of elementary matrix functions

Consider the matrices

$$M^\ell \in \mathbb{R}^{m_\ell \times m_\ell}, \qquad \ell = 1, 2, 3, 4, \tag{23}$$

of the Vandermonde-type systems (15), (17), (19), and (21), where

$$m_1 = p+1, \quad m_2 = p-1, \quad m_3 = p, \quad m_4 = p-2, \tag{24}$$

and p is the order of the method.

The purpose of this section is to construct elementary lower and upper bidiagonal matrices as symbolic functions of the parameters of HP(p). These matrices are most easily constructed by means of a symbolic software. These functions will be used in Section 7 to factor each M^ℓ, $\ell = 1, 2, 3$, into a diagonal+last-column matrix, W^ℓ, which will be further diagonalized by a Gaussian elimination. This decomposition will lead to a fast solution of the systems $M^\ell \boldsymbol{u}^\ell = \boldsymbol{r}^\ell$ in $\mathrm{O}(p^2)$ operations.

Since the Vandermonde-type matrices M^ℓ can be decomposed into the product of a diagonal matrix containing reciprocals of factorials and a confluent Vandermonde matrix, the factorizations used in this paper hold following the approach of Björck and Pereyra [5], Krogh [23], Galimberti and Pereyra [17] and Björck and Elfving [4]. Pivoting is not needed in this decomposition because of the special structure of Vandermonde-type matrices.

5.1. **Symbolic construction of lower bidiagonal matrices.** We first describe the zeroing process of a general vector $\boldsymbol{x} = [x_1, x_2, \ldots, x_m]^T$ with no zero elements. The lower bidiagonal matrix

$$L_k = \begin{bmatrix} I_{k-1} & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & & 0 \\ 0 & 1 & -\tau_{k+1} & & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 1 & -\tau_m \end{bmatrix} \tag{25}$$

defined by the multipliers

$$\tau_i = \frac{x_{i-1}}{x_i} = -L_k(i,i), \qquad i = k+1, k+2, \ldots, m, \tag{26}$$

zeros the last $(m-k)$ components, $x_{k+1}, \ldots, x_m$, of $\boldsymbol{x}$. This zeroing process will be applied recursively on M^ℓ as follows.

For $k = 3, 4, \ldots, m_\ell - 1$, left multiplying $T_k^\ell = L_{k-1}^\ell \cdots L_4^\ell L_3^\ell M^\ell$ by L_k^ℓ zeros the last $(m_l - k)$ components of the kth column of T_k^ℓ. Thus we obtain the upper triangular matrix

$$L^\ell M^\ell = L_{m_\ell - 1}^\ell \cdots L_4^\ell L_3^\ell M^\ell \tag{27}$$

in $(m_\ell - 3)$ steps. We note that L^ℓ does not change the first two rows of M^ℓ.

Process 1. *At the kth step, starting with* $k = 3$*,*

- $M^{\ell(k-1)} = L_{k-1}^\ell L_{k-2}^\ell \cdots L_3^\ell M^\ell$ *is an upper triangular matrix in columns* 1 *to* $k-1$*.*
- *The multipliers in* L_k^ℓ *are obtained from* $M^{\ell(k-1)}(k{+}1 : m_l, k)$ *since* $M^\ell(i,k) \neq 0$ *for* $i = k+1, k+2, \ldots, m_l$*.*

Algorithm 1 in Appendix A describes this process.

5.2. **Symbolic construction of upper bidiagonal matrices.** For each matrix $L^\ell M^\ell$, $\ell = 1, 2, 3$, we construct recursively upper bidiagonal matrices $U_2^\ell, U_3^\ell \ldots, U_{m_\ell - 2}^\ell$ such that the upper triangular matrix $U^\ell = U_2^\ell U_3^\ell \cdots U_{m_\ell - 2}^\ell$ transforms $L^\ell M^\ell$ into a matrix $W^\ell = L^\ell M^\ell U^\ell$ with nonzero diagonal elements, $W^\ell(i,i) \neq 0$, $i = 1, 2, \ldots, m_\ell$, and nonzero $W^\ell(1 : m_\ell, m_\ell) \neq 0$, in the last column, and zero elsewhere. We call such a matrix a "diagonal+last-column" matrix.

We describe the zeroing process of the upper bidiagonal matrix U_k^ℓ on the two-row matrix

$$\begin{aligned} &L^\ell M^\ell U_2^\ell U_3^\ell \cdots U_{k-1}^\ell (k : k+1, 1 : m_\ell) \\ &\quad = \begin{bmatrix} y_{k1} & \cdots & y_{k,k-1} & 1 & 1 & \cdots & 1 & y_{k,m_\ell} \\ y_{k+1,1} & \cdots & y_{k+1,k-1} & y_{k+1,k} & y_{k+1,k+1} & \cdots & y_{k+1,m_\ell - 1} & y_{k+1,m_\ell} \end{bmatrix} \end{aligned} \tag{28}$$

The divisors

$$\sigma_i = \frac{1}{y_{2,i} - y_{2,i-1}} = U_k^\ell(i,i), \qquad i = k+1, k+2, \ldots, m_\ell - 1, \tag{29}$$

define the upper bidiagonal matrix

$$U_k^\ell = \begin{bmatrix} I_{k-1} & 0 & \cdots & 0 & \cdots & 0 & 0 \\ 0 & 1 & -\sigma_{k+1} & 0 & \cdots & 0 & 0 \\ 0 & 0 & \sigma_{k+1} & -\sigma_{k+2} & \cdots & 0 & 0 \\ \vdots & \vdots & & \ddots & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_{m_\ell-2} & -\sigma_{m_\ell-1} & 0 \\ 0 & 0 & 0 & \cdots & 0 & \sigma_{m_\ell-1} & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \tag{30}$$

Right-multiplying (28) by U_k^ℓ zeros the 1's in position $k, \ldots, m_\ell - 1$ in the first row and puts 1's in position $k+1, \ldots, m_\ell - 1$ in the second row:

$$L^\ell M^\ell U_2^\ell U_3^\ell \cdots U_{k-1}^\ell U_k^\ell (k : k+1, 1 : m_\ell) = \begin{bmatrix} y_{k1} & \cdots & y_{k,k-1} & 1 & 0 & \cdots & 0 & y_{k,m_\ell} \\ y_{k+1,1} & \cdots & y_{k+1,k-1} & y_{k+1,k} & 1 & \cdots & 1 & y_{k+1,m_\ell} \end{bmatrix}. \tag{31}$$

Thus, $U^\ell = U_2^\ell U_3^\ell \cdots U_{m_\ell-2}^\ell$ transforms the upper triangular matrix $L^\ell M^\ell$ into the diagonal+last-column matrix

$$W^\ell = L^\ell M^\ell U_2^\ell U_3^\ell \cdots U_{m_\ell-2}^\ell \tag{32}$$

in $(m_\ell - 3)$ steps.

Process 2. *At the kth step, starting with $k = 2$,*

- $M^{\ell(k-1)} = L^\ell M^\ell U_2^\ell U_3^\ell \cdots U_{k-1}^\ell$ *is a diagonal+last-column matrix in rows* 1 *to* $k-1$.
- *The divisors in* U_k^ℓ *are obtained from* $M^{\ell(k-1)}(k+1, k+1 : m_\ell)$ *since* $M^{\ell(k-1)}(k+1, j) - M^{\ell(k-1)}(k+1, j-1) \neq 0$, $j = k+1, k+2, \ldots, m_\ell - 1$.

Algorithm 2 in Appendix A describes this process.

6. Particular variable-step HB(p)3

The general HB(p)3 methods obtained in Section 3 contain one free coefficient, c_2, and depends on h_{n+1} and the previous nodes, $x_n, x_{n-1}, \ldots, x_{n-(p-4)}$, which determine $\eta_2, \ldots, \eta_{p-3}$ in (8).

For simplicity and to reduce the number of offstep points to one, thus reducing the cost per step in the implementation of a particular variable-step HB(p)3, for $p = 5, 6, \ldots, 15$, the following three coefficients were chosen

$$c_1 = 0, \quad c_2 = \frac{2}{3}, \quad c_3 = 1. \tag{33}$$

This choice turned out to be better than other neighboring choices on several problems. The remaining of this paper is concerned with the family of particular VSVO with coefficients c_j given in (33) again denoted by HB(5-15)3.

The procedure to advance integration from x_n to x_{n+1} is as follows.

(a) The order p is obtained by the procedure of Section 9. Then, the stepsize, h_{n+1}, is obtained by formula (38) of Section 9 with $\kappa = p - 1$.

(b) The numbers $\eta_2, \ldots, \eta_{p-3}$, defined in (8), are calculated.
(c) The coefficients of integration formula IF, predictors P_2, P_3 and step control predictor P_4 are obtained successively as solutions of systems (15), (17), (19) and (21).
(d) The values y_{n+c_2}, y_{n+c_3}, y_{n+1}, and $\widetilde{y}_{n+1}$ are obtained by formulae (2)–(5).
(e) The step is accepted if $|y_{n+1}-\widetilde{y}_{n+1}|$ is smaller than the chosen tolerance and the program goes to (a) with n replaced by $n+1$. Otherwise the program returns to (a) with the same order p and the smaller step $0.7\,h_{n+1}$.

7. Fast solution of Vandermonde-type systems for particular HB(p)3

Symbolic elementary matrix functions L_k^ℓ and U_k^ℓ, $\ell = 1, 2, 3$, are constructed once as functions of η_j, for $j = 2, 3, \ldots, p-3$ by Algorithms 1 and 2 in Appendix A to produce diagonal+last-column matrices, which, in turn, are diagonalized by a Gaussian elimination expressed as the product of two elementary matrix functions. These elementary matrix functions are used by fast Algorithms 3 and 4, in Appendix A, to solve systems (15), (17), (19) and (21) at each integration step.

7.1. Solution of $M^\ell \boldsymbol{u}^\ell = \boldsymbol{r}^\ell$, $\ell = 1, 2, 3$. We let

$$m_1 = p+1, \quad m_2 = p-1, \quad m_3 = p, \quad m_4 = p-2,$$

as defined in (24).

Firstly, the elimination procedure of subsection 5.1 is applied to M^ℓ to construct $m_\ell \times m_\ell$ lower bidiagonal matrices L_k^ℓ, $k = 3, \ldots, m_\ell - 1$, with multipliers

$$\tau_i = \frac{i+1-k}{M^\ell(3,k)} = -L_k^\ell(i,i), \qquad i = k+1, k+2, \ldots, m_\ell. \tag{34}$$

The matrix $L^\ell = L_{m_\ell-1}^\ell \cdots L_4^\ell L_3^\ell$ transforms the coefficient matrix M^ℓ into the upper triangular matrix $L^\ell M^\ell$ of the form (27).

Secondly, the elimination procedure of subsection 5.2 is used to construct $m_\ell \times m_\ell$ upper bidiagonal matrices U_k^ℓ, $k = 2, \ldots, m_\ell - 2$, with multipliers

$$\sigma_i = \frac{k-1}{M^\ell(3,i) - M^\ell(3,i-k+1)} = U_k^\ell(i,i), \qquad i = k+1, k+2, \ldots, m_\ell - 1. \tag{35}$$

The right-product of the U_k^ℓ will transform $L^\ell M^\ell$ into a diagonal+last-column matrix W^ℓ of the form (32).

Finally, a factored Gaussian elimination, $L_{m_\ell+1}^\ell L_{m_\ell}^\ell$, diagonalizes W^ℓ as follows. First, $W^\ell(m_\ell, m_\ell)$ is set to 1 by the diagonal matrix $L_{m_\ell}^\ell$:

$$L_{m_\ell}^\ell(i,i) = 1, \qquad i = 1, \ldots, m_\ell - 1,$$
$$L_{m_\ell}^\ell(m_\ell, m_\ell) = 1/W^\ell(m_\ell, m_\ell).$$

Then the non-diagonal entries in the last column of $L_{m_\ell}^\ell W^\ell$ are zeroed by the unit diagonal+last-column matrix $L_{m_\ell+1}^\ell$ whose last column has top $m_\ell - 1$ entries

$$L_{m_\ell+1}^\ell(1 : m_\ell - 1, m_\ell) = -(L_{m_\ell}^\ell W^\ell)(1 : m_\ell - 1, m_\ell).$$

This procedure transforms M^ℓ into the diagonal matrix

$$D^\ell = L^\ell_{m_\ell+1} L^\ell_{m_\ell} \cdots L^\ell_3 M^\ell U^\ell_2 U^\ell_3 \cdots U^\ell_{m_\ell-2},$$

where

$$D^\ell(i,i) = 1, \qquad i = 1, 2, 3, m_\ell,$$

and

$$D^\ell(i,i) = \frac{(i-2)!}{[-M^\ell(3,3)]\,[-M^\ell(3,4)]\cdots[-M^\ell(3,i-1)]}, \qquad i = 4, 5, \ldots, m_\ell - 1.$$

Thus we have the following factorization of M^ℓ into the product of elementary matrices:

$$M^\ell = \left(L^\ell_{m_\ell+1} L^\ell_{m_\ell} \cdots L^\ell_3\right)^{-1} D^\ell \left(U^\ell_2 U^\ell_3 \cdots U^\ell_{m_\ell-2}\right)^{-1},$$

and the solution is

$$\boldsymbol{u}^\ell = U^\ell_2 U^\ell_3 \cdots U^\ell_{m_\ell-2} (D^\ell)^{-1} L^\ell_{m_\ell+1} L^\ell_{m_\ell} \cdots L^\ell_3 \boldsymbol{r}^\ell, \tag{36}$$

where fast computation goes from right to left.

Procedure (36) is implemented in Algorithm 3 in Appendix A in $O(m_\ell^2)$ operations. The input is $M = M^\ell$; $m = m_\ell$; $\boldsymbol{r} = \boldsymbol{r}^\ell$; $L_k = L^\ell_k$, $k = 3, 4, \ldots, m_\ell - 1$; $U_k = U^\ell_k$, $k = 2, 3, \ldots, m_\ell - 2$; and $D = D^\ell$. The output is $\boldsymbol{u} = \boldsymbol{u}^\ell$;

7.2. **Solution of $M^4\boldsymbol{u}^4 = \boldsymbol{r}^4$.** The algorithm to solve the system $M^4\boldsymbol{u}^4 = \boldsymbol{r}^4$ in $O(m_4^2)$ operations is similar to the algorithm for the primal system of [5, p. 896] and is described in Algorithm 4 in Appendix A. The input is $M = M^4$; $m = m_4$; $\boldsymbol{r} = \boldsymbol{r}^4$ and the output is $\boldsymbol{u} = \boldsymbol{u}^4$.

Remark 1. *Formulae* (2)–(5) *can be put in matrix form. For instance,* (4) *can be written as*

$$y_{n+1} = F^1 \boldsymbol{u}^1.$$

where

$$F^1 = \Big[y_n, h_{n+1}f_n, h_{n+1}f_{n+c_2}, h_{n+1}f_{n+c_3}, h_{n+1}f_{n-1}, h_{n+1}f_{n-2}, \ldots, h_{n+1}f_{n-(p-4)}, y_{n-1}\Big],$$

and

$$\boldsymbol{u}^1 = [\alpha_{10}, b_{11}, b_{12}, b_{13}, \beta_{11}, \beta_{12}, \ldots, \beta_{1,p-4}, \alpha_{11}]^T,$$

It is interesting to note the three decomposition forms of the system $F\boldsymbol{u}$:

$$\begin{aligned} &F(UD^{-1}L\boldsymbol{r}) && (\textit{generalized Lagrange interpolation}),\\ &(FUD^{-1})L\boldsymbol{r} && (\textit{Krogh's modified divided differences}),\\ &(FUD^{-1}L)\boldsymbol{r} && (\textit{Nordsieck's formulation}). \end{aligned}$$

The first form is used in this paper, the second form for Vandermonde systems is found in [23]*, and the third form is found in* [27].

8. Regions of absolute stability and principal error term

The region of absolute stability, R, of HB(p)3 is obtained by applying the predictors $\mathrm{P}_2, \mathrm{P}_3$ and the integration formula IF with constant h to the linear test equation

$$y' = \lambda y, \qquad y_0 = 1.$$

This gives the following difference equation and corresponding characteristic equation

$$\sum_{j=0}^{p-3} \gamma_j y_{n+j} = 0, \qquad \sum_{j=0}^{p-3} \gamma_j r^j = 0, \tag{37}$$

respectively, where $p-3$ is the number of steps of the method. A complex number λh is in R if the $p-3$ roots of the characteristic equation satisfy the root condition $|r_s| \leq 1$ and the multiple roots satisfy $|r_s| < 1$. The method used to find R is similar to the one used for k-step multistep methods (see [20, pp. 256–257]).

Let ABM($p, p-1$) denote the ABM method with predictor of order $p-1$ and corrector of order p in PECE mode [31, p. 135–140].

To have a fair comparison of the performance of HB(5-15)3 and ABM($p, p-1$) their regions of absolute stability should be scaled by 1/3 and 1/2, respectively, to take the number of function evaluations into account at each step.

The upper part of the unscaled regions of absolute stability, R, of HB(5-15)3 are shown in grey in Fig. 1. The region R is symmetric with respect to the real axis. The good shape of the stability regions is remarkable.

The scaled intervals of absolute stability $(\alpha/3, 0)$ of HB(p)3 and $(\alpha/2, 0)$ of ABM($p, p-1$) are listed in the left part of Table 1. It is seen that HB methods have larger scaled intervals of absolute stability than ABM methods of comparable order for $p > 7$.

The principal error term of HB(5-15)3 is of the form

$$\left[\delta_1 \{f^p\} + \delta_2 (p)\{\{f^{p-2}\}f\} + \delta_3 \{{}_2f^{p-1}\}_2 + \delta_4 \{{}_3f^{p-2}\}_3\right] h^{p+1}.$$

where $\{f^p\}, \{\{f^{p-2}\}f\}, \{{}_2f^{p-1}\}_2, \{{}_3f^{p-2}\}_3$ are elementary differentials defined in [9, 20]. The principal local truncation coefficients (PLTC), $\delta_1, \delta_2, \delta_3$ and δ_4, of the principal error term are listed in Table 2.

The PLTC of ABM($p, p-1$) are $[\beta_k C_p *, C_{p+1}]$ [24, p. 107].

The scaled norms $3 \times \| \text{ PLTC } \|_2$ of HB(5-15)3 and $2 \times \| \text{ PLTC } \|_2$ of ABM($p, p-1$) of order $p = 5, \ldots, 13$ are listed in the right part of Table 1. It is observed that the scaled norm of HB(p) is smaller than the scaled norm of ABM($p, p-1$).

9. Controlling stepsize and order

A variant of the procedure described in [31] is used to control the stepsize and order of our VSVO HB methods.

- The program computes the maximum norm

$$\mathrm{E} = \|y_{n+1} - \widetilde{y}_{n+1,q}\|_\infty,$$

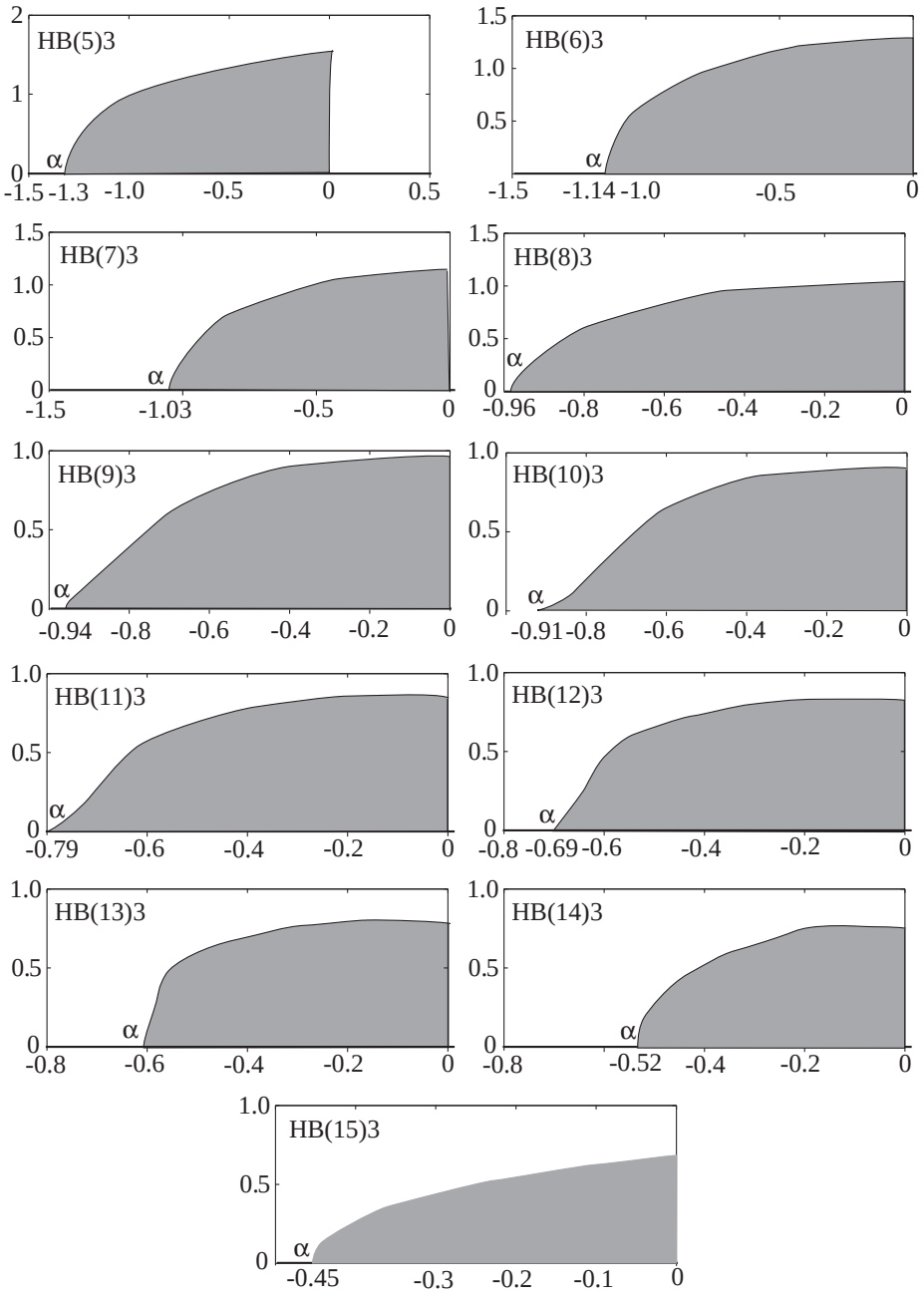

FIGURE 1. Unscaled regions of absolute stability, R, of HB(5-15)3.

where $\widetilde{y}_{n+1,q} := \widetilde{y}_{n+1}$ is the value obtained by the step control predictor P_4 of order $q = p - 2$.

- The stepsize h_{n+1} is obtained by the formula (see [21]):

$$h_{n+1} = \min\left\{h_{\max}, \beta\, h_n \left(\frac{\text{tolerance}}{\mathrm{E}}\right)^{1/\kappa}, 4\, h_n\right\}, \tag{38}$$

TABLE 1. For given order p, the table lists the scaled abscissa of absolute stability, α, and the scaled norm $3\| \mathrm{PLTC} \|_2$ for HB(p)3 and $2\| \mathrm{PLTC} \|_2$ ABM($p, p-1$), respectively.

	$\alpha/3$	$\alpha/2$	$3 \times \| \mathrm{PLTC} \|_2$	$2 \times \| \mathrm{PLTC} \|_2$
p	HB(p)3	ABM($p, p-1$)	HB(p)3	ABM($p, p-1$)
5	−0.43	−0.70	3.93e-02	2.44e-01
6	−0.38	−0.52	2.36e-02	2.18e-01
7	−0.34	−0.39	1.61e-02	2.00e-01
8	−0.32	−0.30	1.19e-02	1.86e-01
9	−0.31	−0.22	9.27e-03	1.75e-01
10	−0.30	−0.17	7.47e-03	1.65e-01
11	−0.26	−0.13	6.18e-03	1.57e-01
12	−0.23	−0.11	5.25e-03	1.51e-01
13	−0.20	−0.03	4.50e-03	1.45e-01
14	−0.17		3.93e-03	
15	−0.15		3.48e-03	

TABLE 2. For each order p, the table lists the principal local truncation coefficients for HB(5-15)3.

p	δ_1	δ_2	δ_3	δ_4
5	$\frac{122167958642}{593736279000119}$	$-\frac{1759218604442}{949978046398679}$	$-\frac{63281244764}{61509369910607}$	$\frac{23456248059221}{2533274790395869}$
6	$\frac{59}{873180}$	$-\frac{3229}{3492720}$	$-\frac{4688352}{11418240917}$	$\frac{8854480}{1597004107}$
7	$\frac{85471927}{3076068117697}$	$-\frac{707023}{1303219439}$	$-\frac{468962}{2354249605}$	$\frac{3854824}{1015639297}$
8	$\frac{2354423}{179914245043}$	$-\frac{587270}{1673247147}$	$-\frac{189737}{1753207927}$	$\frac{3686879}{1313789481}$
9	$\frac{336355}{49883522711}$	$-\frac{68883}{283875427}$	$-\frac{948607}{15000368616}$	$\frac{2708477}{1240785068}$
10	$\frac{179086}{48448766529}$	$-\frac{1873166}{10637173139}$	$-\frac{232193}{5984151746}$	$\frac{1604661}{911571649}$
11	$\frac{116718}{55190680283}$	$-\frac{172071}{1297436831}$	$-\frac{244227}{9931242032}$	$\frac{1821263}{1248757818}$
12	$\frac{62473}{50220577404}$	$-\frac{136176}{1323940543}$	$-\frac{235381}{14807436217}$	$\frac{4230511}{3428202290}$
13	$\frac{170880}{230209966259}$	$-\frac{686019}{8397578576}$	$-\frac{172405}{16640440694}$	$\frac{5242881}{4937467859}$
14	$\frac{123146}{278016455347}$	$-\frac{613889}{9276690181}$	$-\frac{300125}{44620746959}$	$\frac{2397797}{2588362967}$
15	$\frac{53849}{207525400761}$	$-\frac{448503}{8228865958}$	$-\frac{82398}{19247139919}$	$\frac{778718}{952540877}$

with $\kappa = p - 1$ and safety factor $\beta = 0.81$.

- The coefficients of integration formula IF, predictors P_2, P_3 and step control predictor P_4 are obtained successively as solutions of the linear systems (15), (17), (19) and (21).

- The step to x_{n+1} is accepted if $\mathrm{E} \leq$ tolerance, else it is rejected and the program returns to the previous step with smaller step $0.7\,h_{n+1}$.
- If the step to x_{n+1} is successful, besides P_4, three other Adams–Moulton step control predictors,

$$\widetilde{y}_{n+1,\rho} = y_n + h_{n+1}\left(a_{41}f_n + a_{43}f_{n+c_3} + \sum_{j=1}^{\rho-2}\beta_{4j}f_{n-j}\right), \tag{39}$$

of order $\rho = q \pm 1$ and $q-2$ are used to produce the three values $\widetilde{y}_{n+1,\rho}$, respectively, to control the order and stepsize by means of the following three maximum norms,

$$\mathrm{E}_{\pm 1} = \|y_{n+1} - \widetilde{y}_{n+1,q\pm 1}\|_\infty, \quad \mathrm{E}_{-2} = \|y_{n+1} - \widetilde{y}_{n+1,q-2}\|_\infty,$$

which estimate the local error at x_{n+1} had the step to x_{n+1} been taken at orders $q \pm 1$ and $q-2$, respectively. These three quantities are formed with E so that much of the order and step size selection can be done by using the following rules. The lowest satisfactory order is used. Thus, the order is lowered if

$$\mathrm{E}_{-1} \leq \min\{\mathrm{E}, \mathrm{E}_{+1}\} \quad \text{or} \quad \mathrm{E} \geq \max\{\mathrm{E}_{-1}, \mathrm{E}_{-2}\}.$$

The order is raised only if the following stronger conditions,

$$\mathrm{E}_{+1} < \mathrm{E} < \max\{\mathrm{E}_{-1}, \mathrm{E}_{-2}\},$$

are satisfied.

- When the order q of P_4 is 13, E_{+1} is not available; Thus, the order is lowered if

$$\mathrm{E} \geq \max\{\mathrm{E}_{-1}, \mathrm{E}_{-2}\}.$$

- When $q = 3$, the order is raised only if

$$\mathrm{E}_{+1} < \mathrm{E}.$$

- After selecting the order to be used, κ and E are reassigned according to the selected order. For example, if the order is to be lowered in the next step, $\kappa_{n+1} = \kappa_n - 1$ and $\mathrm{E} = \mathrm{E}_{-1}$. The stepsize h_{n+1} is then controlled by formula (38).

10. Numerical results

10.1. **Test problems.** The numerical performance of HB(5-15)3 and DP(8,7)13M has been compared on the following problems: Arenstorf's orbits [1], the Brusselator and the Pleiades [20, pp. 244–249], Euler's equation and the restricted three-body problem [31, pp. 232–259], the cubic wave equation [8] (pointed out to the authors by Philip W. Sharp), and the following nonstiff DETEST problems [21]: the growth problem B1 of two conflicting populations, two-body problems D1–D5, Van der Pol's equation E2, and three easier problems: the oscillatory problem A3, the integral surface of a torus B4, and Bessel's equation of order 1/2 with the origin shifted one unit to the left E1. We report on the performance of the present

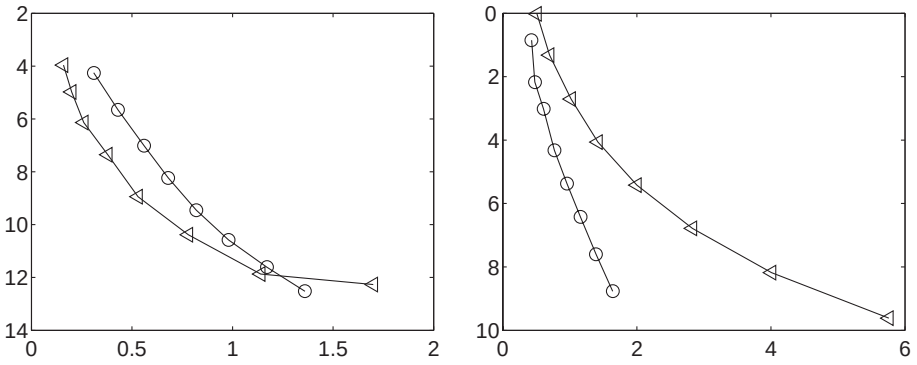

FIGURE 2. CPU (horizontal axis) versus $\log_{10}(|\mathrm{MGE}|)$ (vertical axis) for the Brusselator (left) and the cubic wave (right). VSVO HB(5-15)3 $\circ$ and DP(8,7) $\triangleright$.

methods only on the Brusselator and the cubic wave equation since DP(8,7)13M wins on the other problems.

10.2. **Starting procedure.** The necessary three starting values for HB(5-15)3 were obtained by DP(5,4)7FM (see [14]) with initial step size, h_1, chosen by a method similar to steps (a) and (b) of [20, p. 169].

10.3. **CPU against maximum global error.** CPU time (CPU) has been plotted in Fig. 2 versus the Maximum Global Error (MGE) in HB(5-15)3 and DP(8,7)13M for the Brusselator and the cubic wave. The horizontal axis is CPU for a given tolerance and the vertical axis is the common logarithm of MGE:

$$\log_{10}(|\mathrm{MGE}|)\,. \tag{40}$$

10.4. **CPU percentage efficiency gain.** The CPU percentage efficiency gain (CPU PEG) is defined by formula (cf. Sharp [33]),

$$(\mathrm{CPU\ PEG})_i = 100\left(\frac{\sum_j \mathrm{CPU}_{2,ij}}{\sum_j \mathrm{CPU}_{1,ij}} - 1\right), \tag{41}$$

where $\mathrm{CPU}_{1,ij}$ and $\mathrm{CPU}_{2,ij}$ are the CPU of methods 1 and 2, respectively, associated with problem i, and $j = -\log_{10}(|\mathrm{MGE}|)$.

The CPU PEG for the the Brusselator and the cubic wave is listed in Table 3.

TABLE 3. CPU percentage efficiency gain, CPU PEG, of HB(5-15)3 over DP(8,7)13M for the listed problems.

Problems	CPU PEG
Brusselator	-32%
Cubic Wave	151%

Similar to test results in [16], it is seen from Fig. 2 and Table 3 that for the cubic wave problem whose derivative evaluations are relatively expensive, the new VSVO HB(5-15)3 wins over DP(8,7)13M.

Computations were performed on a Mac with a dual 2.5 GHz PowerPC G5 and 4 GB DDR SSRAM running under Mac OS X Version 10.4.7 and Matlab Version 7.0.4.352 (R14) Service Pack 2.

11. Conclusion and Future Work

A family of variable-step variable-order 3-stage Hermite–Birkhoff (HB) methods of orders 5 to 15 was constructed by solving generalized confluent Vandermonde systems containing Runge–Kutta type order conditions. The stability regions of the HB methods have a remarkably good shape. The order and stepsize of these methods are controlled by four local error estimators.

These methods, in their vectorized Lagrange form, were tested on several problems and were found to have larger scaled regions of absolute stability at higher order and lower scaled error norm than multistep methods. When programmed in C++, they use less CPU time and require fewer function evaluations than DP(8,7)13M also programmed in C++ for costly problems at stringent tolerance.

Future work includes extrapolation, iteration to completion, and continuous extension for approximating the solution between mesh points. A desirable goal is to have the code in Fortran. It is expected that the shallow water problem in [7], which was pointed out to the authors by Philip W. Sharp, will be expensive to solve and prove to be an ideal problem for HB(5-15)3.

Acknowledgment

The anonymous referee is deeply thanked for pointing out important modifications to this paper, communicating invaluable insights for future work and suggesting to use a compiler language. Philip W. Sharp is heartily thanked for his suggestions which considerably improved the content and format of a preliminary version of this paper.

Appendix A. Algorithms

Algorithm 1. ***This algorithm constructs lower bidiagonal matrices*** L_k ***(applied to*** IF, P_2 ***and*** P_3***) as functions of*** c_2, c_3 ***and*** η_j, $j = 2, 3, \ldots, p-3$.

For $k = 3 : m-1$, do the following iteration:
  For $i = m : -1 : k+1$, do the following two steps:
    Step (1) $L_k(i,i) = -M^{\ell}(i-1,k)/M^{\ell}(i,k)$.
    Step (2) For $j = k : m$, compute:

$$M^{\ell}(i,j) = M^{\ell}(i-1,j) + M^{\ell}(i,j)L_k(i,i).$$

Algorithm 2. ***This algorithm constructs upper bidiagonal matrices*** U_k ***(applied to*** IF, P_2 ***and*** P_3***) as functions of*** c_2, c_3 ***and*** η_j, $j = 2, 3, \ldots, p-3$.

For $k = 2 : m-2$, do the following iteration:

For $j = m-1 : -1 : k+1$, do the following two steps:
Step (1) $U_k(j,j) = 1/[M^\ell(k+1,j) - M^\ell(k+1,j-1)]$.
Step (2) for $i = k : j$, compute

$$M^\ell(i,j) = (M^\ell(i,j) - M^\ell(i,j-1))U_k(j,j).$$

Algorithm 3. ***This algorithm solves the systems for*** IF, P_2 ***and*** P_3 ***in*** $O(m^2)$ ***operations***

Given $[\eta_2, \eta_3, \ldots, \eta_{p-3}]$ and $\boldsymbol{r} = r(1:m)$, the following algorithm overwrites $\boldsymbol{r}$ with the solution $\boldsymbol{u} = u(1:m)$ of the system $M\boldsymbol{u} = \boldsymbol{r}$.

Step (1) The following iteration overwrites $\boldsymbol{r} = r(1:m)$ with $L_{m-1}L_{m-2}\cdots L_3\boldsymbol{r}$:
for $k = 3, 4, \ldots, m-1$, compute

$$r(i) = r(i-1) + r(i)L_k(i,i), \qquad i = m, m-1, \ldots, k+1.$$

Step (2) First put

$$G(1:m) = M(1:m, m).$$

We obtain the coefficients of the last two row transformations, L_m and L_{m+1}, by means of the recursion:
for $k = 3, 4, \ldots, m-1$, compute

$$G(i) = G(i-1) + G(i)L_k(i,i), \qquad i = m, m-1, \ldots, k+1.$$

Step (3) The following computation overwrites the newly obtained $\boldsymbol{r}$ with $L_{m+1}L_m\boldsymbol{r}$:

$$r(m) = r(m)/G(m),$$

and for $k = m-1, m-2, \ldots, 1$, compute

$$r(k) = r(k) - G(k)r(m).$$

Step (4) The following iteration overwrites $\boldsymbol{r} = r(1:m)$ with $U_2U_3\cdots U_{m-2}D^{-1}\boldsymbol{r}$:

$$r(i) = r(i)/D(i,i), \qquad i = 1, 2, \ldots, m.$$

For $k = m-2, m-3, \ldots, 2$, compute

$$r(i) = r(i)U_k(i,i), \qquad i = k+1, k+2, \ldots, m-1,$$
$$r(i) = r(i) - r(i+1), \qquad i = k, k+1, \ldots, m-2.$$

Algorithm 4. ***This algorithm solves the system for the step control predictor*** P_4 ***in*** $O(m^2)$ ***operations***

Given $[\eta_2, \eta_3, \ldots, \eta_{p-3}]$ and $\boldsymbol{r} = r(1:m)$, the following algorithm overwrites $\boldsymbol{r}$ with the solution $\boldsymbol{u} = u(1:m)$ of the system $M\boldsymbol{u} = \boldsymbol{r}$.

Step (1) for $k = 2, 3, \ldots, m-1$, compute

$$r(i) = r(i-1) - r(i)\frac{i+1-k}{M^4(2,k)}, \qquad i = m, m-1, \ldots, k+1.$$

Step (2) compute

$$r(i) = r(i), \qquad i = 1, 2.$$

$$r(i) = r(i)\frac{[-M^4(2,2)]\,[-M^4(2,3)]\cdots[-M^4(2,i-1)]}{(i-1)!}, \qquad i = 3, 4, \ldots, m.$$

For $k = m-1, m-2, \ldots, 1$, compute

$$r(i) = r(i)\frac{k}{M^4(2,i+1) - M^4(2,i-k+1)}, \qquad i = k+1, k+2, \ldots, m,$$

$$r(i) = r(i) - r(i+1), \qquad i = k, k+1, \ldots, m-1.$$

ABM$(p, p-1)$

Appendix B. Matlab programming

This appendix is included for the benefit of Matlab users. When programmed in Matlab, HB(5-15)3 turned out to be superior to Matlab's `ode113` on all the problems listed in subsection 10.1.

Algorithm 3 which solves systems IF, P_2 and P_3 were programmed as subroutines in C, e.g., `IFsub`, `P2sub` and, `P3sub`

Algorithm 4 which solves the P_4 system was programmed in C as subroutines in C, e.g., `P4sub`.

A calling program in C, e.g., `IFP` which calls `IFsub`, `P2sub`, `P3sub` and, `P4sub` was compiled together with the above four subroutines by the Matlab `mex` command into `mex` files, e.g., `IFP.macmex`.

At runtime, the data of differential equations were input. Then, `IFP.macmex` was called and run to calculate the values of the coefficients of IF, P_2, P_3 and P_4 at each integration step until completion of the integration.

As an option, CPU time and NFE of function $f(x, y)$ in (1) at the runtime of Algorithms 3 and 4 can be recorded.

As another option, MGE can also be run. Matlab's `ode113` can be run with appropriate tolerance for comparison with HB(p)3.

The elementary matrices L_k^ℓ and U_k^ℓ, $\ell = 1, 2, 3, 4$, are constructed by Algorithms 1 and 2 as functions of η_j, for $j = 2, 3, \ldots, p-3$. These algorithms are not needed at runtime since these matrix functions are already implemented in the four subroutines `IFsub`, `P2sub`, `P3sub` and, `P4sub` which are compiled together with the calling program into Matlab `mex` file `IFP.macmex`.

References

[1] R. F. Arenstorf, *Periodic solutions of the restricted three-body problem representing analytic continuations of Keplerian elliptic motions*, Amer. J. Math., **LXXXV** (1963), pp. 27–35.

[2] R. Ashino, M. Nagase, and R. Vaillancourt, *Behind and beyond the Matlab ODE Suite*, Comput. & Math. with Applics., **40** (2000) pp. 491–512.

[3] R. Barrio, F. Blesa and M. Lara, *VSVO formulation of the Taylor method for the numerical solution of ODEs*, Comput. Math. Applic., **50**, pp. 93–111.

[4] A. Björck and T. Elfving, *Algorithms for confluent Vandermonde systems*, Numer. Math., **21** (1973), pp. 130–137.
[5] A. Björck and V. Pereyra, *Solution of Vandermonde systems of equations*, Math. Comp., **24** (1970), pp. 893–903.
[6] R. K. Brayton, F. G. Gustavson and G.D. Hachtel, *A new efficient algorithm for solving differential-algebraic systems using implicit backward differentiation formulas*, Proc. IEEE, **60** (1972), pp. 98–108.
[7] P. J. Bryant, *Periodic waves in shallow water*, J. Fluid Mech, **59**. part 4, (1973), 625–644.
[8] P. J. Bryant, *Nonlinear wave groups in deep water*, manuscript.
[9] J. C. Butcher, *Coefficients for the study of Runge–Kutta integration processes*, J. Aust. Math. Soc., **3** (1963), pp. 185–201.
[10] J. C. Butcher, *A modified multistep method for the numerical integration of ordinary differential equations*, J. Assoc. Comput. Mach., **12** (1965), pp. 124–135.
[11] M. Calvé and R. Vaillancourt, *Interpolants for Runge–Kutta pairs of order four and five*, Computing, **45** (1990), pp. 383–388.
[12] G. F. Corliss and Y. F. Chang, *Solving ordinary differential equations using Taylor series*, ACM Trans. Math. Software, **8**(2) (1982), pp. 114–144.
[13] P. Deuflhard, *Recent progress in extrapolation methods for ordinary differential equations*, SIAM Rev., **27** (1985), pp. 505–535.
[14] J. R. Dormand and P. J. Prince, *A reconsideration of some embedded Runge–Kutta formulae*, J. Comput. Appl. Math., **15** (1986), pp. 203–211.
[15] W. H. Enright, *Continuous numerical methods for ODEs with defect control*, J. Comput. Appl. Math., **125** (2000), pp. 159-170.
[16] W. H. Enright and T. E. Hull, *The test results on initial value methods for non-stiff ordinary differential equations*, SIAM J. Numr. Anal., **13** (1976) pp. 944–961.
[17] G. Galimberti and V. Pereyra, *Solving confluent Vandermonde systems of Hermite type*, Numer. Math, **18** (1971), pp. 44–60.
[18] C. W. Gear, *The numerical integration of ordinary differential equations*, Math. Comp., **21** (1967), pp. 146–156.
[19] C. W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, NJ, 1971.
[20] E. Hairer, S. P. Nørsett and G. Wanner, *Solving Ordinary Differential Equations I. Nonstiff Problems*, Section III.8, Springer-Verlag. Berlin, 1987.
[21] T. E. Hull, W. H. Enright, B. M. Fellen, and A. E. Sedgwick, *Comparing numerical methods for ordinary differential equations*, SIAM J. Numer. Anal., **9** (1972), pp. 603–637.
[22] F. T. Krogh, *VODQ/SVDQ/DVDQ-variable order integrators for the numerical solution of ordinary differential equations*, TU Doc. No. CP-2308, NPO-11643, May 1969, Jet Propulsion Laboratory, Pasadena, CA.
[23] F. T. Krogh, *Changing stepsize in the integration of differential equations using modified divided differences*, in Proc. Conf. on the Numerical Solution of Ordinary Differential Equations, University of Texas at Austin 1972 (Ed. D.G. Bettis), Lecture Notes in Mathematics No. 362, Springer-Verlag, Berlin, 22–71, (1974)
[24] J. D. Lambert, *Numerical Methods for Ordinary Differential Systems*, Wiley, Chichester UK, 1991.
[25] T. Nguyen-Ba and R. Vaillancourt, *Hermite–Birkhoff differential equation solvers*, Scientific Proceedings of Riga Technical University, 5-th series: Computer Science, 46-th thematic issue, **21** (2004), pp. 47–64.
[26] T. Nguyen-Ba and R. Vaillancourt, *Hermite–Birkhoff–Obrechkoff 3-stage 6-step ODE solver of order 14*, Can. Appl. Math. Quarterly, **13** (Summer 2005), pp. 151–181.
[27] A. Nordsieck, *On numerical integration of ordinary differential equations*, Math. Comp., **16** (1962), 22–49.

[28] L. R. Petzold, *A description of DASSL: A differential/algebraic system solver*, in Proceedings of IMACS World Congress, Montréal, Canada, 1982.
[29] P. J. Prince and J. R. Dormand, *High order embedded Runge–Kutta formulae*, J. Comput. Appl. Math., **7**(1) (1981), pp. 67–75.
[30] L. F. Shampine and L. S. Baca, *Fixed versus variable order Runge–Kutta*, ACM Trans. Math. Software, **12**(1) (1986), pp. 1–23.
[31] L. F. Shampine and M. K. Gordon, *Computer Solution of Ordinary Differential Equations: The Initial Value Problem*, Freeman, San Francisco, 1975.
[32] L. F. Shampine and M. W. Reichelt, *The Matlab ODE suite*, SIAM J. Sc. Comp., **18**(1) (1997), pp. 1–22.
[33] P. W. Sharp, *Numerical comparison of explicit Runge–Kutta pairs of orders four through eight*, Trans. on Mathematical Software, **17** (1991), pp. 387–409.
[34] M. Sofroniou and G. Spaletta, *Precise numerical computation*, J. Log. Algebr. Program, **64**(1) (2005), pp. 113-134.

Department of Mathematics and Statistics, University of Ottawa, Ottawa, Ontario, Canada K1N 6N5.

E-mail address: tnguyen@mathstat.uottawa.ca, hy.emails@gmail.com, yli028@uottawa.ca, remi@uottawa.ca