uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Yuchuan Zhuang
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.Sc. (Systems Science)
GRADE / DEGREE

Systems Science
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Variable-Step Variable-Order 2-Stage Hermite-Birkhoff-Obrechkoff ODE Solver of Order 3 to 14 with a C Program

TITRE DE LA THÈSE / TITLE OF THESIS

Dr. Remi Vaillancourt
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

Dr. Tet Yeap
CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Dr. Mayer Alvo

Dr. Rafal Kulik

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

VARIABLE-STEP VARIABLE-ORDER 2-STAGE HERMITE–BIRKHOFF–OBRECHKOFF ODE SOLVER OF ORDER 3 TO 14 WITH A C PROGRAM

By
Yuchuan Zhuang, B.Sc.
September 2008

A Thesis
submitted to the Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the degree of
Master's of Science in Systems Science

Your file Votre référence
ISBN: 978-0-494-48524-8
Our file Notre référence
ISBN: 978-0-494-48524-8

Abstract

Variable-step variable-order 2-stage Hermite–Birkhoff–Obrechkoff (HBO) methods, HBO(p)2, of order $p = 3$ to 14, named HBO(3-14)2, are constructed for solving nonstiff first-order differential equations. Forcing an expansion of the numerical solution to agree with a Taylor expansion of the true solution leads to multistep and Runge–Kutta type order conditions which are reorganized into linear Vandermonde-type systems of HBO type. Fast algorithms are developed for solving these systems in O(p^2) operations to obtain Hermite–Birkhoff interpolation polynomials in terms of generalized Lagrange basis functions. The order and step size of these methods are controlled by four local error estimators. For numerical computation the lower order 3 is raised to 4 since HBO(4-14)2 produces better results. When programmed in Matlab, HBO(4-14)2 is superior to Matlab's `ode113` in solving several problems often used to test higher order ODE solvers on the basis of the number of steps, CPU time, and maximum global error. On the other hand, HBO(4-14)2 and the Dormand–Prince 13-stage nested Runge–Kutta pair DP(8,7)13M are programmed in C. In this case, DP(8,7) uses less CPU time, have smaller maximum global error but require a larger number of function evaluations than HBO(4-14)2 on nonexpensive problems. However, for expensive equations, such as the Cubicwave, HBO(4-14)2 is superior. Compared with previous results obtained by the 3-stage HBO(4-14)3 on Van der Pol equations with increasing value of ϵ, the new HBO(4-14)2 finally dominates.

Acknowledgements

I would like to express my deep appreciation to my supervisor, Dr. Rémi Vaillancourt and co-supervisor, Dr. Tet Yeap, for their academic and financial support, understanding and guidance during the completion of this thesis. Their constant encouragement, suggestions and ideas have been invaluable to my work.

I would also like to thank Dr. Truong Nguyen-Ba for his insightful suggestions and technical help which sharpened my thesis.

Dedication

To my family....

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Literature Review and Motivation

With the fast development of numerical methods for solving ordinary differential equations (ODEs), it has been realized that to improve the performance of predictor-corrector methods is to vary automatically both the step size and the order if necessary at each integration step. Methods with such a capacity are known as variable-step, variable-order (VSVO) methods.

A large variety of VSVO methods has been designed to solve nonstiff and stiff systems of first-order ODEs. The introduction here will only focus on several approaches and methods. Gear implemented quasi-constant step size methods in software called DIFSUB[23] which is one of the best general purpose differential equation solvers. It starts with a constant step size until changing step size is obviously needed. Considering that constant mesh spacing is very efficient and simple to implement, a continuous extension is used to get approximations to the solution at previous points in an equally spaced mesh to modify the step size. For some methods, the actual mesh size is chosen the same as done in Matlab's `ode113`. The `ode113` method is based on a Predictor-Evaluation-Corrector-Evaluation (PECE) Adams formula while the DIFSUB is based on Adams–Moulton formulas. In this thesis a fully variable step size method is implemented where the actual mesh is chosen by the code as for a PECE Adams formula.

How to choose and implement the formula is an essential concern in the implementation of a numerical method. A Lagrangian formulation is often used to get the coefficients of a method for efficiency. Advantage and disadvantage of using different formulations are discussed by Gear for the Nordsieck form [37], Krogh for modified divided differences [28], and Brayton *et al.* [6] for Lagrangian form. The Lagrangian form is known for simplicity. It should also be noticed that the number of iterations and the CPU time depend on the order of the formula. Krogh [28] dealt with the effects of roundoffs under stringent tolerances. It turns out that using divided differences is an efficient way to minimize roundoffs' effect. The Lagrangian form manipulates vectors of the same size as the number of equations. Sofroniou and Spaletta [44] have also manipulated vectors to develop extrapolation solvers which can be used to obtain remarkable accuracies in Mathematica.

The software DIFSUB implements Adams–Moulton formulas while the codes DVDQ [29] and `ode113` [42, 2] implement Adams–Bashforth–Moulton multistep formulas in PECE modes. Extrapolation is considered another way to obtain high accuracy. Deuflhard [15] took responsibility for drawing attention to the value of this method, while the codes of [25, Section II.9] are probably spreading its use. In fact, NDSolve of Mathematica is based on these codes. Hairer *et al.* [25] advocate extrapolation solving problems for its high accuracy, because it provides users with the accuracy they want. Extrapolation can be considered as a variable-order Runge–Kuttta method. Shampine and Baca [40] argued that RK of a fixed order (7,8) pair is more efficient than extrapolation at accuracies common in scientific computation after an important comparison of Enright and Hull [19] on fixed order Runge–Kutta codes and extrapolation codes had been made. In the end, Taylor series become a preferable choice in VSVO implementation such as in astronomical calculations [3]. One can refer to the work of Corliss and Chang [13] to get a general introduction. Besides, an interpolant [18, 12] for approximating the solution between mesh points needs more consideration depending on the global smoothness required. A fundamental element in developing the Matlab ODE Suite was to provide solvers with an event location facility. The occurrence of an event is when the value of a function of the dependent variables or their derivatives increases and decreases through a given

level. That is why the Suite does not contain an extrapolation code nor a high order Runge–Kutta pair. Reviewing the past, one has to admit that Fehlberg's (7,8) pair did draw attention to the value of high order RK pair.

The new VSVO HBO(3-14)2 methods [45] for solving nonstiff systems of first-order initial value problems of the form

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

can be seen as a combination of multistep methods and Runge–Kutta methods with backstep points. For general multistep methods, they use information prior to the last step, while for Runge–Kutta methods, they use derivative evaluations at points partway through the current step. The link between the two types of methods is that predicted values are obtained by means of predictors which use values at previous points. It is shown in [9] that in order to reduce the number of backsteps without lowering the order, general linear multistep methods merge function evaluations at off-step points.

Hermite–Birkhoff (HB) methods of orders 9, 10 and 11 have been studied in [32]. A 3-stage VSVO family HB(5-15)3 of order 5 to 15 and a 3-stage VSVO family HBO(4-14)3 of order 4 to 14 have been constructed in [35] and [36], respectively.

In order to solve relatively expensive nonstiff ordinary differential equations, we will formulate a new 2-stage variable-step variable-order (VSVO) general linear methods of order $p = 3, 4, \ldots, 14$ and program the algorithm in the interpreter language C. Since these methods use HB interpolation polynomials and first and second order derivatives of y at step points, they will be called Hermite–Birkhoff–Obrechkoff methods (HBO) and the family of such methods will be designated by HBO(3-14)2. We deal here only with algorithms for nonstiff initial value problems.

1.2 Thesis Objective

The objectives of this thesis are:

- To formulate algorithms for HBO(p)2 for non-stiff ordinary differential equations;

- To fully derive the order conditions for HBO(p)2 based on the formulation of the algorithm;
- To program the algorithm using the C programming language;
- To compare the performance of the proposed algorithm HBO(p)2 with DP(8,7)13M and HBO(4-14)3 in C.

1.3 Thesis Organization

In Chapter 2 we will present the background and review the literature on ODE solvers. In Chapter 3, we will introduce a new family of general HBO(p)2 methods of order $p = 3, 5, \ldots, 14$. Order conditions are listed in Section 3.3. In Section 3.4 general HBO(p)2 are represented in terms of Vandermonde-type systems. In Section 3.5 elementary matrices are constructed symbolically as functions of the parameters of the methods in view of factoring the coefficient matrices of Vandermonde-type systems. Fast solutions of Vandermonde-type systems are considered in Section 3.6. Section 3.7 deals with the step and order control. The implementation of HBO(4-14)2 and numerical results are found in Chapter 4. Section 3.8 considers the regions of absolute stability and principal local truncation coefficients of constant step HBO(3-14)2. Numerical testing of HBO(4-14)2 programmed in C is done in Sections 4.1–4.3. Section 4.4 briefly describes the C program used for the testing. Appendix A lists the algorithms. Appendix B describes the Matlab programming for Matlab users.

1.4 Thesis Contribution

The contribution of this thesis includes the following items:

- The derivation of the order conditions from the formulation of the algorithm for the HBO(3-14)2 method;
- The implementation of HBO(4-14)2 and DP(8,7) in C, including the regions of absolute stability, principal error terms, step control and order control;

- A comparison of the numerical results from HBO(4-14)2 with DP(8,7) and HBO(4-14)2 with HBO(4-14)3 in C;
- The writing up of a 4100-line C programming code for HBO(4-14)2 (available on demand).

Chapter 2

Background and Notation

2.1 ODE Systems

The core concepts for systems of ordinary differential equations will be introduced in this section. Numerical methods are implemented based on first-order systems since high-order systems can always be reduced to first-order systems. The general solution of linear systems with constant coefficients is also represented. In the end, two important interpolation formulas will also be discussed.

First-Order Systems

First-order system of ordinary differential equations in the form:

$$\begin{aligned}
{}^1y' &= {}^1f(x, {}^1y, {}^2y, \ldots, {}^my) \\
{}^2y' &= {}^2f(x, {}^1y, {}^2y, \ldots, {}^my) \\
&\vdots \quad \vdots \\
{}^my' &= {}^mf(x, {}^1y, {}^2y, \ldots, {}^my)
\end{aligned}$$

can be written in vector form as $y' = f(x, y)$, where $y = [{}^1y, {}^2y, \ldots, {}^my]^T$ and $f = [{}^1f, {}^2f, \ldots, {}^mf]^T$. If each ${}^tf$ $(t = 1, 2, \ldots, m)$ depends on ${}^1y, {}^2y, \ldots, {}^my$, the system is coupled. It is this coupling that is the essence of the system.

For a general m-dimensional system, if imposed by m side conditions that each ${}^ty, t = 1, 2, \ldots, m$ takes given value at the same initial point, such system is called

an initial value problem. Its vector form is denoted by $y' = f(x, y)$, $y(a) = \eta$, where $\eta = [\eta^1, \eta^2, \ldots, \eta^m]^T$. Not all initial value problems have a unique solution unless the following theorem is satisfied.

Let $f(x, y)$ be defined and continuous for all (x, y) in the region $\mathbb{D}$ where $a \leq x \leq b, -\infty <^t y < \infty, t = 1, 2, \ldots, m$. If there exists a constant L which satisfies $\|f(x, y) - f(x, y^*)\| \leq L\|y - y^*\|$ for all $(x, y), (x, y^*) \in \mathbb{D}$, then there exists a unique solution for the system. This requirement is known as a *Lipschitz condition* and L as a *Lipschitz constant.*

If $f(x, y)$ is differentiable with respect to y, then it is Lipschitz continuous with respect to y, since from the mean value theorem $f(x, y) - f(x, y^*) = J(x, \zeta)(y - y^*)$, where $J(x, \zeta)$ is the Jacobian of f with respect to y. ζ_i is on the line segment from (x, y) to (x, y^*) and the ith row of the Jacobian is evaluated at ζ_i (see [31, p. 6]).

Higher-order Systems

When it comes to the numerical solutions of higher-order systems, it is standard practice first to reduce them to first-order systems. According to Lambert's notation [31, p. 7], which is used in the program of this thesis, the qth-order m-dimensional system of ordinary differential equations in the form

$$y^{(q)} = \varphi(x, y^{(0)}, y^{(1)}, \ldots, y^{(q-1)}) \tag{2}$$

can be reduced to the first-order system of dimension qm by the following substitutions:

$$\begin{aligned} Y_1 &:= y && (\equiv y^{(0)}) \\ Y_2 &:= Y_1' && (\equiv y^{(1)}) \\ &\vdots && \vdots \\ Y_q &:= Y_{q-1}' && (\equiv y^{(q-1)}). \end{aligned}$$

which gives the system of first-order equations:

$$\begin{aligned} Y_1' &:= Y_2 \\ Y_2' &:= Y_3 \\ &\vdots \quad\quad \vdots \\ {Y_{q-1}}' &:= Y_q \\ Y_q' &= \varphi(x, Y_1, Y_2, \ldots, Y_q). \end{aligned}$$

The higher-order system (2) together with the initial conditions $y^{(r)}(a) = \eta_{r+1}, \quad r = 0, 1, \ldots, q-1$, can thus be written in the form

$$Y' = F(x, Y), \qquad Y(a) = \chi,$$

where $\chi := [\eta_1, \eta_2, \ldots, \eta_q]^T$.

Linear Systems with Constant Coefficients

If $f(x, y)$ of an m-dimensional first-order system $y' = f(x, y)$ has the form $f(x, y) = A(x)y + \varphi(x)$, where $A(x)$ is an $m \times m$ matrix and $\varphi(x) \in \mathbb{R}^m$, the system is said to be linear. Furthermore, if $A(x)$ is independent of x (denoted as A), it is called linear with constant coefficients and has the form of

$$y' = Ay + \varphi(x) \tag{3}$$

and its homogeneous form is

$$y' = Ay. \tag{4}$$

If A admits m linearly independent eigenvectors, the general solution of (4) is a linear combination of m linearly independent solutions $\{y_t(x) = e^{\lambda_t x} c_t, t = 1, 2, \ldots, m\}$, where λ_t is an eigenvalue of A and c_t the corresponding eigenvector. If $\psi(x)$ is one particular solution of (3), then the general solution of (3) is

$$y(x) = \sum_{t=1}^{m} \kappa_t \, e^{\lambda_t x} c_t + \psi(x),$$

where the κ_t are arbitrary constants.

The Newton-Gregory backward interpolation formula

A unique polynomial of degree at most q, $I_q(x)$, exists, which interpolates $q+1$ distinct data points (x_{n-j}, F_{n-j}), $j = 0, 1, \ldots, q$. Different representations of the polynomial $I_q(x)$ are created to compute $I_{q+1}(x)$ from $I_q(x)$ easily. When the data are evenly spaced, $I_q(x)$ can be written in terms of backward differences of F as

$$I_q(x) = P_q(r) = \sum_{i=0}^{q} (-1)^i \binom{-r}{i} \nabla^i F_n$$

where $x = x_n + rh$, and $\binom{-r}{i}$ is the binomial coefficient. In the case $q = 2$, we have

$$P_2(r) = F_n + r\nabla F_n + \frac{1}{2} r(r+1)\nabla^2 F_n$$

$$\begin{aligned} x = x_n &\Leftrightarrow r = 0 &&\Rightarrow P_2(r) = F_n \\ x = x_{n-1} &\Leftrightarrow r = -1 &&\Rightarrow P_2(r) = F_n - \nabla F_n = F_{n-1} \\ x = x_{n-2} &\Leftrightarrow r = -2 &&\Rightarrow P_2(r) = F_n - 2\nabla F_n + \nabla^2 F_n = F_{n-2} \end{aligned}$$

The Lagrange interpolation formula

When the data are unevenly spaced, an easy interpolation formula is based on Lagrange. Define

$$L_{q,j}(x) = \prod_{i=0, i\neq j}^{q} \frac{x - x_{n-i}}{x_{n-j} - x_{n-i}},$$

where $L_{q,j}(x)$ is a polynomial in x of degree q, such that

$$L_{q,j}(x_{n-i}) = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j, \end{cases} \qquad i = 1, 2, \ldots, q.$$

Thus the polynomial $I_q(x)$ can be written in the form

$$I_q(x) = \sum_{j=0}^{q} L_{q,j}(x) F_{n-j}.$$

A serious weak point for Lagrange formula is that when we need to add a further data point (x_{n-q-1}, F_{n-q-1}), with q replaced by $q+1$, we get an expression for $I_{q+1}(x)$, but there is no easy way we can generate $I_{q+1}(x)$ directly from $I_q(x)$.

2.2 Linear One-step and Multistep Methods

Consider a general m-dimensional first-order system

$$y' = f(x, y), y(a) = \eta,$$

where $y = [{}^1y, {}^2y, \ldots, {}^my]^T, f = [{}^1f, {}^2f, \ldots, {}^mf]^T$ and $\eta = [{}^1\eta, {}^2\eta, \ldots, {}^m\eta]^T$. The numerical solution is derived by implementing the idea of *discretization*. That is, instead of seeking an approximate solution on the continuous x of interval $[a, b]$, find an approximate solution y_n to the theoretical solution at x_n, that is, to $y(x_n)$, on discrete point set $\{x_n | n = 0, 1, \ldots, (b-a)/h\}$. A difference equation is generated to produce a sequence of values y_n which approximates the solution of the first-order system on the discrete point set $\{x_n\}$. Such a difference equation is a numerical method. Normally it involves a number of consecutive approximations $y_{n+j}, j = 0, 1, \ldots, k$, from which we can sequentially compute the sequence $y_n, n = 0, 1, 2, \ldots, N$. The integer k is called the stepnumber of the method. If $k = 1$, the method is called a *one-step* method. Otherwise, it is called a *multistep* or *k-step* method.

Linear k-step methods have the form of

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h \sum_{j=0}^{k} \beta_j f_{n+j},$$

where α_j and β_j are constants subject to the conditions $\alpha_k = 1$ and $|\alpha_0| + |\beta_0| \neq 0$. Adams multistep methods are of the form

$$y_{n+k} - y_{n+k-1} = h \sum_{j=0}^{k} \beta_j f_{n+j}.$$

One useful multistep method is Adams method. Its order of 1 to 13 or 14 are frequently used. It turns out that the region of absolute stability decreases as the order increases. The low order multistep methods are often replaced by predictor-corrector methods with larger regions of absolute stability due to limited step size at low order.

The multistep methods starts with the one-step pair for a given tolerance with no additional starting values required. The step size and order can be changed accordingly at each integration step. The error estimators are used to monitor the

step size and order of VSVO multistep methods. Assuming the order is accepted, let $E_{k,n+1}$ denote the norm of the local error estimate at x_{n+1}, where k is the order of the method and let τ be a user-defined tolerance. If $E_{k,n+1} \leq \tau$, the step is accepted. Otherwise the step will be redone with a smaller step size. Other predictors will be used to monitor the order changes.

2.3 Runge–Kutta Methods

Matching an expansion of the numerical solution y_{n+1} with the Taylor expansion of the exact solution $y(x_{n+1})$ is the core technique for deriving Runge–Kutta methods.

The 3-stage explicit Runge–Kutta method has the form:

$$\begin{aligned} y_{n+1} &= y_n + h(b_1k_1 + b_2k_2 + b_3k_3), \\ k_1 &= f(x_n, y_n), \\ k_2 &= f(x_n + hc_2, y_n + ha_{21}k_1), \\ k_3 &= f(x_n + hc_3, y_n + ha_{31}k_1 + ha_{32}k_2). \end{aligned} \tag{5}$$

where the simplifying assumptions $c_i = \sum_j a_{ij}$ are used. Thus $a_{31} = c_3 - a_{32}$. The Taylor expansion of $y(x_{n+1})$ is

$$y(x_{n+1}) = y(x_n) + hf + \frac{1}{2}h^2F + \frac{1}{6}h^3(Ff_y + G) + O(h^4), \tag{6}$$

where the elementary differentials F and G are defined as $F = f_x + k_1f_y$ and $G = f_{xx} + 2k_1f_{xy} + k_1^2f_{yy}$, also denoted by $\{f\}$ and $\{f^2\}$ in the context of Butcher's rooted trees [31, p. 153–154, 166].

Let $\widetilde{y}_{n+1}$ denote the value of y at x_{n+1} generated by the Runge–Kutta method. In order to use the local truncation error $T_{n+1} = y(x_{n+1}) - y_{n+1}$, we need an expansion for y_{n+1} similar to (6). Expanding the k_i given by (5), we have $k_1 = f$ and

$$k_2 = f + hc_2(f_x + k_1f_y) + \frac{1}{2}h^2{c_2}^2(f_{xx} + 2k_1f_{xy} + {k_1}^2f_{yy}) + O(h^3). \tag{7}$$

We can rewrite k_2 in the form

$$k_2 = f + hc_2F + \frac{1}{2}h^2{c_2}^2G + O(h^3). \tag{8}$$

If we substitute the expansion for k_1 and k_2 into k_3 and expand, we obtain

$$\begin{aligned} k_3 &= f + h\{c_3 f_x + [(c_3 - a_{32})k_1 + a_{32}k_2]f_y\} \\ &\quad + \frac{1}{2}h^2\{{c_3}^2 f_{xx} + 2c_3[(c_3 - a_{32})k_1 + a_{32}k_2]f_{xy} \\ &\quad + [(c_3 - a_{32})k_1 + a_{32}k_2]^2 f_{yy}\} + O(h^3) \\ &= f + hc_3 F + h^2(c_2 a_{32} F f_y + \frac{1}{2}{c_3}^2 G) + O(h^3). \end{aligned} \tag{9}$$

On substituting (8) and (9) into (5) and using the localizing assumption, which stipulates that past values are exact, we obtain the following expansion for y_{n+1}:

$$\begin{aligned} y_{n+1} &= y(x_n) + h(b_1 + b_2 + b_3)f + h^2(b_2 c_2 + b_3 c_3)F \\ &\quad + \frac{1}{2}h^3[2b_3 c_2 a_{32} F f_y + (b_2 {c_2}^2 + b_3 {c_3}^2)G] + O(h^4). \end{aligned} \tag{10}$$

Then we need to match the expansions (6) and (10).

One-Stage method.

If we set $b_2 = b_3 = 0$, (5) is a one-stage method. Then (10) reduces to

$$\widetilde{y}_{n+1} = y(x_n) + hb_1 f + O(h^4).$$

Matching with (6) yields $b_1 = 1$, then $T_{n+1} = O(h^2)$. There exists only one explicit one-stage Runge–Kutta method of order 1, namely Euler's Rule.

Two-Stage method.

If we set $b_3 = 0$, we get a two-stage method and (10) becomes

$$\widetilde{y}_{n+1} = y(x_n) + h(b_1 + b_2)f + h^2 b_2 c_2 F + \frac{1}{2}h^3 b_2 {c_2}^2 G + O(h^4).$$

If we match the coefficients of this expression with those of (6), we have $b_1 + b_2 = 1$ and $b_2 c_2 = 1/2$. The result is a Runge–Kutta method of order 2.

Because the two equations contains three unknowns, there exists a one-parameter family of solutions of explicit two-stage Runge–Kutta methods of order 2. No member of this family can achieve order higher than two.

Three-Stage method. In order to achieve order 3, we need to satisfy the following order conditions:

$$\begin{aligned} b_1 + b_2 + b_3 &= 1, \\ b_2 c_2 + b_3 c_3 &= \frac{1}{2}, \\ b_2 {c_2}^2 + b_3 {c_3}^2 &= \frac{1}{3}, \\ b_3 a_{32} c_2 &= \frac{1}{6}. \end{aligned} \tag{11}$$

That results from matching (6) and (10) up to $O(h^4)$. Since those four equations contains six unknowns, there exists a two-parameter family of solutions. No member of this family can achieve order higher than three.

The four elementary differentials of order four,

$$\{f^3\}, \{f\{f\}_2, \{{}_2f^2\}_2, \{{}_3f\}_3,$$

are used in the formulation of the principal error term in RK3.

2.4 Dormand–Prince Pair of Orders 8 and 7

DP(8,7)13M is the acronym for the 13-stage Runge–Kutta pair of Dormand–Prince. The coefficients set used to advance integration is of order 8 and the error estimation is of order 7. DP(8,7)13M is a `First Same As Last` method. The reason why it is called FSAL is that the value y at the end of each integration step is the same as the k_1 value at the next integration step. The Butcher tableau of a FSAL method is shown in Table 1. DP(8,7)13M was implemented in C and its performance was compared with HBO(4-14)2 also implemented in C.

2.5 ABM and PC Methods

Adams–Bashforth–Moulton (ABM) methods are also called Predictor-Corrector (PC) method where an explicit Adams-Bashforth method is used as predictor and an implicit Adams-Moulton method is used as corrector. The idea of a Predictor-Corrector

Table 1: The Butcher tableau of a FSAL method

0					
c_2	a_{21}				
c_3	a_{31}	a_{32}			
$\vdots$			$\ddots$		
c_{s-1}	$a_{s-1,1}$	$a_{s-1,2}$	$\cdots$	$a_{s-1,s-2}$	
1	b_1	b_2		b_{s-2}	b_{s-1}
	b_1	b_2		b_{s-2}	b_{s-1}

method is that a numerical method for an ODE generates an approximate solution step-by-step in discrete increments across the interval of integration. At each step, the predictor provides an initial guess for the next solution value. The corrector which is a more stable and more accurate method is then used to improve the initial guessed value, rather than solving the implicit equation by functional iteration [23].

The 4th order ABM method uses the formulas below to calculate y_{i+1}:

Predictor:

$$y_{i+1}^P = y_i^C + \frac{h}{24}\left(-9f_{i-3}^C + 37f_{i-2}^C - 59f_{i-1}^C + 55f_i^C\right).$$

Corrector:

$$y_{i+1}^C = y_i^C + \frac{h}{24}\left(f_{i-2}^C - 5f_{i-1}^C + 19f_i^C + 9f_{i+1}^P\right).$$

for $i = 3, 4, \ldots$, where the predicted values of y_i and f_i are denoted by y_i^P and f_i^P and the corrected values of y_i and f_i are denoted by y_i^C and f_i^C. The predictor uses the information at the points (x_{i-3}, f_{i-3}), (x_{i-2}, f_{i-2}), (x_{i-1}, f_{i-1}) and (x_i, f_i) to make the Lagrange polynomial approximation for $f(x, y(x))$. The information at the current point and the three previous points x_{i-1}, x_{i-2}, x_{i-3} is used to predict the value y_{i+1}^P at the next point, x_{i+1}. The corrector uses the value y_{i+1}^P from the predictor step. It can be seen that there are three separate processes involved in a Predictor-Corrector method. The first process is the predictor step denoted by P. The second process is the evaluation of the function $f_{i+1}^P = f(x_{i+1}, y_{i+1}^P)$ denoted by E. The third step is the corrector denoted by C. In this case, the Predictor-Corrector is said to be in the Predictor-Evaluation-Corrector (PEC) mode. Usually, a final evaluation yields

a better method than PEC. In this case, it is said to be a Predictor-Evaluation-Corrector-Evaluation (PECE) mode.

2.6 Introduction of HBO(4-14)3

Before constructing new HBO(3-14)2 method, similar HBO(4-14)3 [36] will be briefly introduced here. The variable-step variable-order method HBO(4-14)3 we developed to solve non-stiff systems of first-order differential equations. It was based on a 3-stage Runge–Kutta method of order 3 and a Hermite-Birkhoff interpolation polynomials. In order to perform the integration step of HBO(4-14)3, the predictor P_2,P_3,P_4 and the integration formula IF were computed as follows.

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

$$y_{n+c_2} = y_n + h_{n+1}\left(a_{21}f_{n+c_1} + \sum_{j=1}^{\lfloor (p-3)/2 \rfloor} \beta_{2j} f_{n-j}\right) + h_{n+1}^2\left(\sum_{j=0}^{\lfloor (p-4)/2 \rfloor} \gamma_{2j} f'_{n-j}\right). \quad (12)$$

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

$$y_{n+c_3} = y_n + h_{n+1}\left(\sum_{j=1}^{2} a_{3j} f_{n+c_j} + \sum_{j=1}^{\lfloor (p-3)/2 \rfloor} \beta_{3j} f_{n-j}\right) + h_{n+1}^2\left(\sum_{j=0}^{\lfloor (p-4)/2 \rfloor} \gamma_{3j} f'_{n-j}\right). \quad (13)$$

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

$$y_{n+1} = y_n + h_{n+1}\left(\sum_{j=1}^{3} b_{1j} f_{n+c_j} + \sum_{j=1}^{\lfloor (p-3)/2 \rfloor} \beta_{1j} f_{n-j}\right) + h_{n+1}^2\left(\sum_{j=0}^{\lfloor (p-4)/2 \rfloor} \gamma_{1j} f'_{n-j}\right). \quad (14)$$

(P_4) A Hermite–Birkhoff polynomial of degree p is used as step control predictor P_4 to obtain $\widetilde{y}_{n+1}$ to order $(p-2)$,

$$\widetilde{y}_{n+1} = y_n + h_{n+1}\left(\sum_{j=1}^{2} a_{4j} f_{n+c_j} + a_{43} f_{n+1} + \sum_{j=1}^{\lfloor (p-3)/2 \rfloor} \beta_{4j} f_{n-j}\right) + h_{n+1}^2 \left(\sum_{j=0}^{\lfloor (p-4)/2 \rfloor} \gamma_{4j} f'_{n-j}\right). \quad (15)$$

According to the numerical analysis in the reference [36], it was found that HBO(4-14)3 won over DP(8,7)13M for expensive problems such as: the Brusselator and Cubicwave problems on a long time period. For light problems such as D1 and Arenstorf, DP(8,7)13M is more efficient. For more details, one can refer to the paper mentioned above.

2.7 Definition of the colon ":" notation

The colon ":" notation is frequently used in this thesis to specify a column or row of a matrix. If $B \in \mathbb{R}^{m \times n}$, then $B(k,:)$ designates the kth row,

$$B(k,:) = [b_{k1}, b_{k2}, \ldots, b_{kn}].$$

The kth column is specified by

$$B(:,k) = \begin{bmatrix} b_{1k} \\ b_{2k} \\ \vdots \\ b_{mk} \end{bmatrix}.$$

If the integers p, q, and r satisfy $1 \le p \le q \le n$ and $1 \le r \le m$, then

$$B(r, p:q) = [b_{rp}, \ldots, b_{rq}] \in \mathbb{R}^{1 \times (q-p+1)}.$$

Similarly, if $1 \le p \le q \le m$ and $1 \le c \le n$, then

$$B(p:q, c) = \begin{bmatrix} b_{pc} \\ \vdots \\ b_{qc} \end{bmatrix} \in \mathbb{R}^{q-p+1}.$$

This notation is extended to designate a rectangular part of a matrix B as $B(p:q, r:s)$ from row p to row q and column r to column s.

The colon notation is extensively used in the numerical literature and in Matlab. See [24, pages 7–8 and 19].

Chapter 3

VSVO HBO(3-14)2

In this chapter, general variable-step HBO(p)2 of variable order p and fast algorithms for solving Vandermonde-type matrices will be constructed.

3.1 Goals and Improvements of HBO(3-14)2

The goals of this thesis are to show that the new HBO(4-14)2 has better performance than DP(8,7)13M for solving some expensive ODE problems. Besides, previous results of HBO(4-14)3 are also compared with the new HBO(4-14)2 on Van der Pol's equation E2 for different $\epsilon \in (1, 10)$. It turns out that HBO(4-14)3 is more efficient than HBO(4-14)2 when ϵ is small, but HBO(4-14)2 finally wins over HBO(4-14)3 as ϵ increases.

Contrary to HBO(4-14)3 which used off-step points and is based on RK 3, HBO(4-14)2 does not use off-step points and is based on RK2. HBO(4-14)3 has larger region of absolute stability and smaller error coefficients than HBO(4-14)2.

3.2 General Variable Step HBO(p)2 of Order p

An HBO(p)2 method is said to be a **general** variable-step HBO method if its backstep points are variable parameters. If the step size is constant, and hence the backsteps are fixed parameters, the method is said to be a **constant-step** method.

A general 2-stage HBO(p)2 of order $p = 3, 4, \ldots, 14$ requires the following three formulas to perform the integration step from x_n to x_{n+1}, where, for simplicity, the offstep point $c_1 = 0$ is used in the summations.

(P_2) A Hermite–Birkhoff polynomial of degree $(p-1)$ is used as predictor P_2 to obtain y_{n+c_2} to order $(p-1)$,

$$y_{n+c_2} = y_n + h_{n+1}\left(a_{21} f_{n+c_1} + \sum_{j=1}^{\mu-1} \beta_{2j} f_{n-j}\right) + h_{n+1}^2 \left(\sum_{j=0}^{\nu-1} \gamma_{2j} f'_{n-j}\right). \tag{16}$$

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

$$y_{n+1} = y_n + h_{n+1}\left(b_{11} f_{n+c_1} + b_{12} f_{n+c_2} + \sum_{j=1}^{\mu-1} \beta_{1j} f_{n-j}\right) + h_{n+1}^2 \left(\sum_{j=0}^{\nu-1} \gamma_{1j} f'_{n-j}\right). \tag{17}$$

(P_3) A Hermite–Birkhoff polynomial of degree p is used as step control predictor P_3 to obtain $\widetilde{y}_{n+1}$ to order $(p-2)$,

$$\widetilde{y}_{n+1} = y_n + h_{n+1}\left(a_{31} f_{n+c_1} + a_{32} f_{n+1} + \sum_{j=1}^{\mu-1} \beta_{3j} f_{n-j}\right) + h_{n+1}^2 \left(\sum_{j=0}^{\nu-1} \gamma_{3j} f'_{n-j}\right). \tag{18}$$

The "floor" of a real number q, denoted by $\lfloor q \rfloor$, is the largest integer smaller or equal to q. The number ν of f' in the above formulae is defined as the smallest integer $\nu = \lfloor \min\{p-1, 6\}/2 \rfloor$ and the number μ of steps of the method is $\mu = p - \nu - 1$. We note that f'_{n+1} is computed only once at x_{n+1}.

3.3 Order Conditions for HBO(p)2

As in similar search for ODE solvers, we impose the following multistep-type conditions on P_2:

$$k! B_2(k+1) = \frac{1}{k+1} c_2^{k+1}, \qquad k = 0, 1, 2, \ldots, p-2, \tag{19}$$

where

$$B_2(j) = \sum_{\ell=1}^{\mu-1} \left[\beta_{2\ell} \frac{\eta_{\ell+1}^{j-1}}{(j-1)!} \right] + \sum_{\ell=1}^{\nu-1} \left[\gamma_{2\ell} \frac{\eta_{\ell+1}^{j-2}}{(j-2)!} \right], \quad j = 0, 1, 2, \ldots, p, \tag{20}$$

with $\eta_{\ell+1}^{-2} = 0$ and $\eta_{\ell+1}^{-1} = 0$ by notation, and

$$\eta_j = -\frac{1}{h_{n+1}} (x_n - x_{n+1-j}) = -\frac{1}{h_{n+1}} \sum_{i=0}^{j-2} h_{n-i}, \qquad j = 2, 3, \ldots, 10. \tag{21}$$

Equation (21) will be frequently used in this thesis.

There remains the set of equations of the integration formula IF to be solved:

$$\sum_{i=1}^{2} b_{1i} c_i^k + k! B_1(k+1) = \frac{1}{k+1}, \qquad k = 0, 1, \ldots, p-1, \tag{22}$$

where

$$B_1(j) = \sum_{i=1}^{\mu-1} \beta_{1i} \frac{\eta_{i+1}^{j-1}}{(j-1)!} + \sum_{i=1}^{\nu-1} \gamma_{1i} \frac{\eta_{i+1}^{j-2}}{(j-2)!}, \qquad j = 1, 2, \ldots, p+1. \tag{23}$$

The numbers $B_1(k)$ and $B_2(k)$ are associated with IF and P_2, respectively.

3.4 Vandermonde-type Formulation of HBO(p)2

3.4.1 Integration formula IF

To find the IF coefficients which satisfy the order conditions, we consider the p-vector of reordered coefficients of the integration formula IF in (17),

$$\boldsymbol{u}^1 = [b_{11}, \gamma_{10}, b_{12}, \beta_{11}, \gamma_{11}, \beta_{12}, \gamma_{12}, \beta_{13}, \beta_{14}, \ldots, \beta_{1,\mu-1}]^T,$$

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

$$M^1 \boldsymbol{u}^1 = \boldsymbol{r}^1, \tag{24}$$

where

$$M^1 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 0 & \cdots & 1 \\ 0 & 1 & c_2 & \eta_2 & 1 & \eta_3 & 1 & \cdots & \eta_\mu \\ 0 & 0 & \frac{c_2^2}{2!} & \frac{\eta_2^2}{2!} & \eta_2 & \frac{\eta_3^2}{2!} & \eta_3 & \cdots & \frac{\eta_\mu^2}{2!} \\ \vdots & & & & & & & & \vdots \\ 0 & 0 & \frac{c_2^{p-1}}{(p-1)!} & \frac{\eta_2^{p-1}}{(p-1)!} & \frac{\eta_2^{p-2}}{(p-2)!} & \frac{\eta_3^{p-1}}{(p-1)!} & \frac{\eta_3^{p-2}}{(p-2)!} & \cdots & \frac{\eta_\mu^{p-1}}{(p-1)!} \end{bmatrix} \tag{25}$$

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

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

The leading error term of IF is

$$\left[b_{12} \frac{c_2^p}{p!} + \sum_{j=1}^{\mu-1} \beta_{1j} \frac{\eta_{j+1}^p}{p!} + \sum_{j=1}^{\nu-1} \gamma_{1j} \frac{\eta_{j+1}^{p-1}}{(p-1)!} - \frac{1}{(p+1)!} \right] h_{n+1}^{p+1} y_n^{(p+1)}.$$

The detailed structure of columns 4 to $\min\{p, 6\}$ of $M^1 \in \mathbb{R}^{p \times p}$ is as follows:

$$\begin{aligned} M^1(i,j) &= \eta_{\lfloor j/2 \rfloor}^{i-1}/(i-1)!, \\ M^1(i,j+1) &= \frac{d}{d\eta_{\lfloor j/2 \rfloor}} M^1(i,j), \end{aligned} \qquad \begin{cases} i = 1, 2, \ldots, p, \\ j = 4, 6, \ldots, \min\{p, 6\}, \end{cases}$$

with $j + 1 \leq p$ in the second equation.

3.4.2 Predictor P_2

The $(p-1)$-vector of reordered coefficients of predictor P_2 in (16),

$$\boldsymbol{u}^2 = [a_{21}, \gamma_{20}, \beta_{21}, \gamma_{21}, \beta_{22}, \gamma_{22}, \beta_{23}, \beta_{24}, \ldots, \beta_{2,\mu-1}]^T,$$

is the solution of the system of order conditions

$$M^2 \boldsymbol{u}^2 = \boldsymbol{r}^2, \tag{26}$$

where

$$M^2 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & \cdots & 1 \\ 0 & 1 & \eta_2 & 1 & \eta_3 & \cdots & \eta_\mu \\ 0 & 0 & \frac{\eta_2^2}{2!} & \eta_2 & \frac{\eta_3^2}{2!} & \cdots & \frac{\eta_\mu^2}{2!} \\ \vdots & & & & & & \vdots \\ 0 & 0 & \frac{\eta_2^{p-2}}{(p-2)!} & \frac{\eta_2^{p-3}}{(p-3)!} & \frac{\eta_3^{p-2}}{(p-2)!} & \cdots & \frac{\eta_\mu^{p-2}}{(p-2)!} \end{bmatrix} \tag{27}$$

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

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

The detailed structure of columns 3 to $\min\{p-1, 5\}$ of $M^2 \in \mathbb{R}^{(p-1)\times(p-1)}$ is as follows:

$$\begin{aligned} M^2(i,j) &= \eta_{\lfloor j/2 \rfloor}^{i-1}/(i-1)!, \\ M^2(i,j+1) &= \frac{d}{d\eta_{\lfloor j/2 \rfloor}} M^2(i,j), \end{aligned} \qquad \begin{cases} i = 1, 2, \ldots, p-1, \\ j = 3, 5, \ldots, \min\{p-1, 5\}, \end{cases}$$

with $j + 1 \leq p - 1$ in the second equation.

A truncated Taylor expansion of the right-hand side of (16) about x_n gives

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

where the coefficients are defined by

$$S_2(j) = M^2(j, 1:p-2)\boldsymbol{u}^2 = r_2(j) = \frac{c_2^j}{j!}, \qquad j = 1, 2, \ldots, p-1, \tag{28}$$

$$S_2(j) = \sum_{i=1}^{\mu-1} \beta_{2i} \frac{\eta_{i+1}^{j-1}}{(j-1)!} + \sum_{i=0}^{r_2-1} \gamma_{2i} \frac{\eta_{i+1}^{j-2}}{(j-2)!}, \qquad j = p-1, p, p+1.$$

We see by (28) that P_2 is of order $p-1$. The leading term of the error of P_2 is

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

3.4.3 Step control predictor P_3

The p-vector of reordered coefficients of P_3 in (18),

$$\widetilde{\boldsymbol{u}}^3 = [a_{31}, \gamma_{30}, a_{32}, \beta_{31}, \gamma_{31}, \beta_{32}, \gamma_{32}, \beta_{33}, \beta_{34}, \ldots, \beta_{3,\mu-1}]^T.$$

By setting $\gamma_{30} = \gamma_{10} + \omega_0$ and $a_{32} = b_{12} + \omega_2$, $\widetilde{\boldsymbol{u}}^3$ is reduced to the $(p-2)$-vector $\boldsymbol{u}^3$ which is the solution of the system of order conditions:

$$M^3 \boldsymbol{u}^3 = \boldsymbol{r}^3, \tag{29}$$

where

$$M^3 = \begin{bmatrix} 1 & 1 & 0 & \cdots & 1 \\ 0 & \eta_2 & 1 & \cdots & \eta_\mu \\ 0 & \frac{\eta_2^2}{2!} & \eta_2 & \cdots & \frac{\eta_\mu^2}{2!} \\ \vdots & & & & \vdots \\ 0 & \frac{\eta_2^{p-2}}{(p-2)!} & \frac{\eta_2^{p-3}}{(p-3)!} & \cdots & \frac{\eta_\mu^{p-2}}{(p-2)!} \end{bmatrix} \tag{30}$$

and the $(p-2)$ components of $\boldsymbol{r}^3 = r_3(1:p-2)$ are

$$r_3(i) = 1/i! - (b_{12} + \omega_2)\frac{1}{(i-1)!}, \qquad i = 1, 3, 4, \ldots, p-2,$$

$$r_3(2) = 1/2! - (b_{12} + \omega_2) - (\gamma_{10} + \omega_0), \qquad i = 2$$

where ω_0 and ω_2 are nonzero arbitrary numbers and P_3 yields $\widetilde{y}_{n+1}$ to order $(p-2)$. According to numerical experimentation, $\omega_2 = -0.025$ and $\omega_0 = 0.058$ are found to be a good choice.

3.5 Symbolic Construction of Elementary Matrices

Consider the matrices

$$M^\ell \in \mathbb{R}^{m_\ell \times m_\ell}, \qquad \ell = 1, 2, 3, \tag{31}$$

of the Vandermonde-type systems (24), (26), and (29), where

$$m_1 = p, \quad m_2 = p-1, \quad m_3 = p-2. \tag{32}$$

A fast solution of these systems in $O(m_\ell^2)$ operations will be achieved by decomposing $(M^\ell)^{-1}$ into the product of lower and upper bidiagonal matrices, one diagonal matrix and one upper tridiagonal matrix.

This section is to construct symbolically elementary lower and upper bidiagonal matrices as functions of the parameters of HBO(p)2. These functions will be used in Subsection 3.4.1 to diagonalize each M^ℓ, $\ell = 1, 2, 3$, and this can be easily done by means of the symbolic software.

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 thesis hold following the approach of Björck and Pereyra [5], Krogh [28], Galimberti and Pereyra [21] and Björck and Elfving [4]. Pivoting is not needed in this decomposition because of the special structure of Vandermonde-type matrices.

3.5.1 Construction of lower bidiagonal matrices for M^ℓ, $\ell = 1, 2, 3$

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

$$L_k = \begin{bmatrix} I_{k-1} & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & & 0 \\ 0 & 1 & -\tau_{k+1} & & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 1 & -\tau_m \end{bmatrix} \tag{33}$$

defined by the multipliers

$$\tau_i = \frac{x_{i-1}}{x_i} = -L_k(i,i), \qquad i = k+1, k+2, \ldots, m, \tag{34}$$

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 $k = k_0^\ell, k_0^\ell + 1, \ldots, m_\ell - 1$, left multiplying $T = L_{k-1}^\ell \cdots L_{k_0^\ell+1}^\ell L_{k_0\ell}^\ell M^\ell$ by L_k^ℓ zeros the last $(m_l - k)$ components of the kth column of T. Thus we obtain the upper triangular matrix

$$L^\ell M^\ell = L_{m_\ell-1}^\ell \cdots L_{k_0^\ell+1}^\ell L_{k_0^\ell}^\ell M^\ell \tag{35}$$

in $(m_\ell - k_0^\ell)$ matrix operations, where

$$k_0^1 = 3, \quad k_0^2 = 3, \quad k_0^3 = 2$$

We note that L^ℓ does not change the first two rows of M^ℓ.

Process 1 *At the kth step, starting with* $k = k_0$,

- $M^{\ell(k-1)} = L^{\ell}_{k-1} L^{\ell}_{k-2} \cdots L^{\ell}_{k_0^{\ell}} M^{\ell}$ *is an upper triangular matrix in columns* 1 *to* $k-1$.
- *The multipliers in* L^{ℓ}_k *are obtained from* $M^{\ell(k-1)}(k+1, \ldots, m_l, k)$ *since* $M^{\ell}(i,k) \neq 0$ *for* $i = k+1, k+2, \ldots, m_l$.

Algorithm 1 in Appendix A describes this process.

3.5.2 Symbolic construction of initializing upper tridiagonal matrices U^{ℓ}_1 for M^{ℓ}, $\ell = 1, 2, 3$

The second step in diagonalizing M^{ℓ} transforms the first two rows of $L^{\ell}M^{\ell}$ by right multiplication with an upper tridiagonal matrix U^{ℓ}_1 such that

$$L^{\ell}M^{\ell}U^{\ell}_1(1:2, 1:m_{\ell}) = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & \cdots & 1 \end{bmatrix}. \tag{36}$$

The action of U^{ℓ}_1 amounts to take the divided difference of the columns of $L^{\ell}M^{\ell}$ whose first component is 1 (cf. [4]).

3.5.3 Construction of upper bidiagonal matrices

We construct upper bidiagonal matrices U^{ℓ}_k, $k = 2, 3, \ldots, m_{\ell} - 1$. The right multiplication on $L^{\ell}M^{\ell}U^{\ell}_1$ amounts to take the kth order divided differences of the columns k to m_{ℓ} of the matrices on which they act.

Specifically, consider the two-row matrix:

$$\begin{aligned} E &\equiv L^{\ell}M^{\ell}U^{\ell}_1 \cdots U^{\ell}_{k-1}(k:k+1, 1:m_{\ell}) \\ &= \begin{bmatrix} y_{k1} & \cdots & y_{k,k-1} & 1 & 1 & \cdots & 1 & 1 \\ y_{k+1,1} & \cdots & y_{k+1,k-1} & y_{k+1,k} & y_{k+1,k+1} & \cdots & y_{k+1,m_{\ell}-1} & y_{k+1,m_{\ell}} \end{bmatrix} \end{aligned} \tag{37}$$

and define the upper bidiagonal matrix

$$U_k^\ell = \begin{bmatrix} I_{k-1} & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & -\sigma_{k+1} & 0 & \cdots & 0 & 0 \\ 0 & 0 & \sigma_{k+1} & -\sigma_{k+2} & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & 0 & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_{m_\ell-2} & -\sigma_{m_\ell-1} & 0 \\ 0 & 0 & 0 & \cdots & 0 & \sigma_{m_\ell-1} & -\sigma_{m_\ell} \\ 0 & 0 & 0 & \cdots & 0 & 0 & \sigma_{m_\ell} \end{bmatrix} \tag{38}$$

by means of the divisors

$$\sigma_i = \frac{1}{y_{2,i} - y_{2,i-1}} = U_k^\ell(i,i), \qquad i = k+1, k+2, \ldots, m_\ell. \tag{39}$$

Then, right multiplying (37) by U_k^ℓ returns E with $E_{1,j} = 0$ for $j = k+1, \ldots, m_\ell$ and $E_{2,j} = 1$ for $j = k+1, \ldots, m_\ell$. So

$$L^\ell M^\ell U_1^\ell \cdots U_{k-1}^\ell U_k^\ell (k : k+1, 1 : m_\ell) = \begin{bmatrix} y_{k1} & \cdots & y_{k,k-1} & 1 & 0 & \cdots & 0 & 0 \\ y_{k+1,1} & \cdots & y_{k+1,k-1} & y_{k+1,k} & 1 & \cdots & 1 & 1 \end{bmatrix}. \tag{40}$$

Applying U_k^ℓ, $k = 2, 3, \ldots, m_\ell - 1$, on the right of the upper triangular matrix $L^\ell M^\ell U_1^\ell$, yields the diagonal matrix

$$D^\ell = L^\ell M^\ell U^\ell = L_{m_\ell-1}^\ell \cdots L_{k_0^\ell+1}^\ell L_{k_0^\ell}^\ell M^\ell U_1^\ell U_2^\ell \cdots U_{m_\ell-1}^\ell \tag{41}$$

in $(m_\ell - 2)$ steps.

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

- $M^{\ell(k-1)} = (L^\ell M^\ell U_1^\ell) U_2^\ell \cdots U_{k-1}^\ell$ *is a diagonal 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* $M^{\ell(k-1)}(k+1, j) - M^{\ell(k-1)}(k+1, j-1) \neq 0$ *for* $j = k+1, k+2, \ldots, m_\ell$.

Algorithm 2 in Appendix A describes this process.

3.6 Fast Solution of Vandemonde-type Systems

For L_k^ℓ and U_k^ℓ, $\ell = 1, 2, 3$, these elementary matrix functions are constructed once as functions of $\eta_2, \ldots, \eta_{10}$. These matrices are used by the fast Algorithm 3 listed in Appendix A to solve $M^\ell \boldsymbol{u}^\ell = \boldsymbol{r}^\ell$ for $\ell = 1, 2, 3$, in (24), (26) and (29), respectively.

We recall, from (32), that

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

Firstly, the elimination procedure of subsection 3.4.1 is applied to M^ℓ to construct $m_\ell \times m_\ell$ lower bidiagonal matrices L_k^ℓ, $k = k_0^\ell, k_0^\ell + 1, \ldots, m_\ell - 1$, of the form (33) defined by the multipliers where

$$\tau_i = \frac{i + (k_0^\ell - 1) - k}{\mu_\ell(k)} = -L_k^\ell(i, i), \qquad i = k + 1, k + 2, \ldots, m_\ell, \tag{42}$$

and

$$\mu_\ell(k) = \begin{cases} M^\ell(2, k), & \text{if } M^\ell(1, k) = 1, \\ M^\ell(3, k), & \text{if } M^\ell(1, k) = 0, \end{cases} \qquad k = k_0^\ell, k_0^\ell + 1, \ldots, m_\ell - 1.$$

Left multiplying M^ℓ by L_k^ℓ, $k = k_0^\ell, k_0^\ell + 1, \ldots, m_\ell - 1$, produces the upper triangular matrix $L^\ell M^\ell = L_{m_\ell - 1}^\ell \cdots L_{k_0^\ell}^\ell M^\ell$ of the form (35).

Secondly, we construct the $m_\ell \times m_\ell$ upper tridiagonal matrix U_1^ℓ which transforms M^ℓ into the matrix of first order divided differences $M^\ell U_1^\ell$ of the columns of M^ℓ whose first component is 1 and where the divisors are taken from the second row of M^ℓ.

For given p and ℓ, Table 2 lists the diagonal elements of U_1^ℓ. The nonzero off-diagonal elements of U_1^ℓ satisfy the following three equations:

$$\begin{aligned} U_1^1(3, 4) &= -U_1^1(4, 4), \\ U_1^1(i - 2, i) &= -U_1^1(i, i), \qquad i = 3, 6, \ldots, \min\{m_1, 8\}, \\ U_1^1(i - 1, i) &= -U_1^1(i, i), \qquad i = 9, 10, \ldots, m_1, \end{aligned} \tag{43}$$

$$\begin{aligned} U_1^2(i - 2, i) &= -U_1^2(i, i), \qquad i = 3, 5, \min\{m_2, 7\}, \\ U_1^2(i - 1, i) &= -U_1^2(i, i), \qquad i = 8, 9, \ldots, m_2, \end{aligned} \tag{44}$$

Table 2: The diagonal entries of U_1^ℓ

i	$U_1^1(i,i)$	$U_1^2(i,i)$	$U_1^3(i,i)$
1	1	1	1
2	1	1	$1/\eta_2$
3	$1/c_2$	$1/\eta_2$	1
4	$1/(\eta_2 - c_2)$	1	$1/(\eta_3 - \eta_2)$
5	1	$1/(\eta_3 - \eta_2)$	1
6	$1/(\eta_3 - \eta_2)$	1	$1/(\eta_4 - \eta_3)$
7	1	$1/(\eta_4 - \eta_3)$	$1/(\eta_5 - \eta_4)$
8	$1/(\eta_4 - \eta_3)$	$1/(\eta_5 - \eta_4)$	$1/(\eta_6 - \eta_5)$
9	$1/(\eta_5 - \eta_4)$	$1/(\eta_6 - \eta_5)$	$1/(\eta_7 - \eta_6)$
10	$1/(\eta_6 - \eta_5)$	$1/(\eta_7 - \eta_6)$	$1/(\eta_8 - \eta_7)$
11	$1/(\eta_7 - \eta_6)$	$1/(\eta_8 - \eta_7)$	$1/(\eta_9 - \eta_8)$
12	$1/(\eta_8 - \eta_7)$	$1/(\eta_9 - \eta_8)$	$1/(\eta_{10} - \eta_9)$
13	$1/(\eta_9 - \eta_8)$	$1/(\eta_{10} - \eta_9)$	
14	$1/(\eta_{10} - \eta_9)$		

$$\begin{aligned} U_1^3(1,2) &= -U_1^3(2,2) \qquad if \quad m_3 \geq 2, \\ U_1^3(i-2,i) &= -U_1^3(i,i), \qquad i = 4,6,\ldots,\min\{m_3,6\}, \\ U_1^3(i-1,i) &= -U_1^3(i,i), \qquad i = 7,\ldots,m_3, \end{aligned} \tag{45}$$

The remaining elements of U_1^ℓ are zero. The first two rows of $M^\ell U_1^\ell$ are as in (36).

Thirdly, the elimination procedure of subsection 3.4.3 is used to construct $m_\ell \times m_\ell$ upper bidiagonal matrices U_k^ℓ, $k = 2,\ldots,m_\ell - 1$, of the form (38) where the divisors are given by

$$\sigma_i = \frac{k}{\mu_\ell(i) - \mu_\ell(i-k)} = U_k^\ell(i,i), \qquad i = k+1, k+2, \ldots, m_\ell. \tag{46}$$

Right multiplying $L^\ell M^\ell$ by U_k^ℓ, $k = 1,\ldots,m_\ell - 1$, produces the diagonal matrix

$$D^\ell = L_{m_\ell-1}^\ell L_{m_\ell-2}^\ell \cdots L_{k_0^\ell}^\ell M^\ell U_1^\ell U_2^\ell \cdots U_{m_\ell-1}^\ell,$$

where

$$D^\ell(i,i) = 1, \qquad i = 1,2,\ldots,k_0^\ell,$$

and

$$D^\ell(i,i) = \frac{k_0^\ell(k_0^\ell+1)\dots(i-1)}{[-\mu_\ell(k_0^\ell)]\,[-\mu_\ell(k_0^\ell+1)]\cdots[-\mu_\ell(i-1)]}, \qquad i = k_0^\ell+1, k_0^\ell+2, \dots, m_\ell.$$

Finally, M^ℓ is decomposed into the product of elementary matrices:

$$M^\ell = \left(L^\ell_{m_\ell-1}L^\ell_{m_\ell-2}\cdots L^\ell_{k_0^\ell}\right)^{-1} D^\ell \left(U_1^\ell U_2^\ell \cdots U^\ell_{m_\ell-1}\right)^{-1}$$

and the solution of $M^\ell \boldsymbol{u}^\ell = \boldsymbol{r}^\ell$ is

$$\boldsymbol{u}^\ell = U_1^\ell U_2^\ell \cdots U^\ell_{m_\ell-1}(D^\ell)^{-1}L^\ell_{m_\ell-1}L^\ell_{m_\ell-2}\cdots L^\ell_{k_0^\ell}\,\boldsymbol{r}^\ell, \tag{47}$$

where fast computation goes from right to left.

Process 3 *Procedure* (47) *is implemented in the following two steps:*

Step 1 Algorithm 3 in Appendix A overwrites $\boldsymbol{r}^\ell = r_\ell(1:m_\ell)$ *with* $U_2^\ell\cdots U^\ell_{m_\ell-1}(D^\ell)^{-1}L^\ell_{m_\ell-1}L^\ell_{m_\ell-2}\cdots L^\ell_{k_0^\ell}\,\boldsymbol{r}^\ell$ *in* $O(m_\ell^2)$ *operations. The input is* $M = M^\ell$; $m = m_\ell$; $\boldsymbol{r} = \boldsymbol{r}^\ell$; $L_k = L_k^\ell$, $k = k_0^\ell, k_0^\ell+1, \dots, m_\ell-1$; $U_k = U_k^\ell$, $k = 2, \dots, m_\ell - 1$; *and* $D = D^\ell$. *In Matlab,* $A(a:b)$ *is the vector which contains the elements between the a-th and b-th position of the array A.*

Step 2 For each value of ℓ, *one of the following three computations is performed:*

Case 1 ($\ell = 1$) *The following iteration overwrites* $\boldsymbol{r}^1 = r_1(1:m_1)$ *with* $U_1^1\boldsymbol{r}^1$:

$$\begin{aligned}
&r_1(3) = r_1(3)U_1^1(3,3),\\
&r_1(i) = r_1(i)U_1^1(i,i), && i = 4,6,\dots,\min\{m_1,8\}\\
&r_1(i) = r_1(i)U_1^1(i,i), && i = 9,10,\dots,m_1,\\
&r_1(1) = r_1(1) - r_1(3),\\
&r_1(3) = r_1(3) - r_1(4), && \textit{if}\quad m_1 \geq 4\\
&r_1(i) = r_1(i) - r_1(i+2), && i = 4,6,\dots,\min\{6, m_1-2\},\\
&r_1(i) = r_1(i) - r_1(i+1), && i = 8,9,\dots,m_1-1.
\end{aligned}$$

Case 2 ($\ell = 2$) The following iteration overwrites $\mathbf{r}^2 = r_2(1 : m_2)$ with $U_1^2\mathbf{r}^2$:

$$\begin{aligned}
r_2(i) &= r_2(i)U_1^2(i,i), && i = 3, 5, \ldots, \min\{7, m_2\},\\
r_2(i) &= r_2(i)U_1^2(i,i), && i = 8, 9, \ldots, m_2,\\
r_2(i) &= r_2(i) - r_2(i+2), && i = 1, 3, \ldots, \min\{5, m_2 - 2\},\\
r_2(i) &= r_2(i) - r_2(i+1), && i = 7, 8, \ldots, m_2 - 1,
\end{aligned}$$

Case 3 ($\ell = 3$) The following iteration overwrites $\mathbf{r}^3 = r_3(1 : m_3)$ with $U_1^3\mathbf{r}^3$:

$$\begin{aligned}
r_3(i) &= r_3(i)U_1^3(i,i), && i = 2, 4, \ldots, \min\{6, m_3\},\\
r_3(i) &= r_3(i)U_1^3(i,i), && i = 7, 8, \ldots, m_3,\\
r_3(1) &= r_3(1) - r_3(2), && \textit{if} \quad m_3 \geq 2.\\
r_3(i) &= r_3(i) - r_3(i+2) && i = 2, 4, \ldots, \min\{4, m_3 - 2\},\\
r_3(i) &= r_3(i) - r_3(i+1), && i = 6, 7, \ldots, m_3 - 1.
\end{aligned}$$

3.7 Controlling Stepsize and Order

A variant of the procedure described in [41] is used to control the step size and order,p, of our VSVO HBO(3-14)2 methods. For simplicity, the order of the step control predictor P_3 will be denoted by $q = p - 2$.

- The program computes the maximum norm

$$E_q = \|y_n - \widetilde{y}_{n,q}\|_\infty,$$

  where $\widetilde{y}_{n,q} := \widetilde{y}_n$ is the value obtained by the step control predictor P_3.

- The step size h_{n+1} is obtained by the formula (see [26]):

$$h_{n+1} = \min\left\{h_{\max}, \beta\, h_n \left(\frac{\text{tolerance}}{E_q}\right)^{1/\kappa}, 4\, h_n\right\}, \tag{48}$$

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

- The coefficients of the integration formula IF, predictors P_2 and step control predictor P_3 are obtained successively as solutions of the linear systems (24), (26) and (29).

- The step to x_{n+1} is accepted if $E_q \leq$ tolerance, else it is rejected and the program returns to the previous step with smaller step. The heuristic value $0.7\,h_{n+1}$ is often used.

- If the step to x_{n+1} is successful, besides P_3, three other step control predictors of order $\rho = q \pm 1$ and $q-2$,

$$\widetilde{y}_{n+1} = y_n + h_{n+1}\left[a_{31}f_{n+c_1} + a_{32}f_{n+1} + \sum_{j=1}^{\mu-1}\beta_{3j}f_{n-j}\right] + h_{n+1}^2\left[\sum_{j=0}^{\nu-1}\gamma_{3j}f'_{n-j}\right],$$

are used, where ν of f' in the formulae is $\nu = \lfloor \min\{\rho+1, 6\}/2 \rfloor$ and the number μ of steps of the predictor is $\mu = \rho + 2 - \nu - 1$.

These three step control predictors are used to produce three values $\widetilde{y}_{n+1,\rho}$, respectively, to control the order and step size by means of the following three maximum norms,

$$E_{q\pm1} = \|y_{n+1} - \widetilde{y}_{n+1,q\pm1}\|_\infty \quad \text{and} \quad E_{q-2} = \|y_{n+1} - \widetilde{y}_{n+1,q-2}\|_\infty$$

which estimate the local error at x_{n+1} had the step to x_{n+1} been taken at orders $q \pm 1$ and $q-2$. These three quantities are combined with E to select the order and step size as follows. The lowest satisfactory order is used. Thus, the order is lowered if

$$E_{q-1} \leq \min\{E_q, E_{q+1}\} \quad \text{or} \quad E_q \geq \max\{E_{q-1}, E_{q-2}\}.$$

The order is raised only if the following stronger conditions,

$$E_{q+1} < E_q < \max\{E_{q-1}, E_{q-2}\},$$

are satisfied. When the order q of P_4 is 12, E_{q+1} is not available; Thus, the order is lowered if

$$E_q \geq \max\{E_{q-1}, E_{q-2}\}.$$

When $q = 2$, the order is raised only if

$$E_{q+1} < E_q.$$

- After selecting the order to be used, κ and E_q are reassigned according to the selected order. For example, if the order is to be lowered in the next step, $\kappa_{\text{new}} = \kappa_{\text{old}} - 1$ and $E_q = E_{q-1}$. The step size h_{n+1} is then controlled by formula (48).

3.8 Regions of Absolute Stability and Principal Error Term

Zero-stability is concerned with the stability of a method as the step size h tends to zero. The importance of zero-stability stems from Dalhquist's theorem with asserts that a method is convergent if and only if it is consistent, that is of order at least one, and zero stable [31, p. 35].

In practice, one applies a numerical method with $h > 0$. In this case one speaks of absolute stability. Depending on the method used to study absolute stability, one speaks of linear stability [31, pp. 68ff, 117ff and 198ff] and nonlinear stability [31, Chap. 7].

A numerical method is said to be absolutely stable if the error in the numerical solution of a differential equation remains bounded as the number of steps increases for a step size $h > 0$ which depends on the differential equation. Hence convergence is possible only with absolutely stable methods.

Absolute stability is ordinarily studied for linear equations $y' = \lambda y$ or linear systems $y' = Ay$, where the matrix A is diagonalizable with eigenvalues $\lambda_1, \lambda_2, \ldots$.

To obtain the regions of absolute stability, R, of HBO(p)2, we apply the predictor P_2 and integration formula IF of each method, with constant h, to the linear test

equation

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

This gives the difference equation

$$\sum_{j=0}^{\mu} \gamma_j(\lambda h) y_{n+j} = 0, \tag{49}$$

and, by the substitution $y_{n+j} = r^{n+j}$, we have the corresponding characteristic equation

$$\sum_{j=0}^{\mu} \gamma_j(\lambda h) r^j = 0, \tag{50}$$

where μ is the number of steps of the method. A complex number λh is in R if the μ roots of the characteristic equation (50) satisfy the root condition $|r_s| \leq 1$ and the multiple roots satisfy $|r_s| < 1$. In this case, the solution y_n remains bounded as $n \to \infty$.

The root condition is used to find the regions of absolute stability of HBO(3-14)2 in the complex plane $\mathbb{C}$. The region R is symmetric with respect to the real axis. Figure. 1 shows the part of R in the upper half-plane.

Let ABM$(p, p-1)$ denote the ABM method with predictor of order $p-1$ and corrector of order p in PECE mode [41, p. 135–140]. Table 3 lists the scaled abscissae of absolute stability, $\alpha/3$ and $\alpha/2$, of HBO(3-14)2 and ABM(3-13) [41, p. 135–140], respectively. It is seen that HBO$(p)2$ has a larger scaled interval of absolute stability than ABM$(p, p-1)$ for $p = 9, 10, \ldots, 13$.

The principal error term of HBO(3-14)2 is of the form

$$\left[\delta_1 \{f^p\} + \delta_2 \{{}_2f^{p-1}\}_2\right] h^{p+1},$$

where $\{f^p\}$ and $\{{}_2f^{p-1}\}_2$ are elementary differentials defined in [8], [30] and [25]. The principal local truncation coefficients (PLTC), δ_1, δ_2, of the principal error term are listed in Table 4. Also listed are the scaled norms $3 \times \|\text{PLTC}\|_2$ and $2 \times \|\text{PLTC}\|_2$ for HBO$(p)2$ and ABM$(p, p-1)$, respectively, the former being rapidly decreasing and being smaller that the latter.

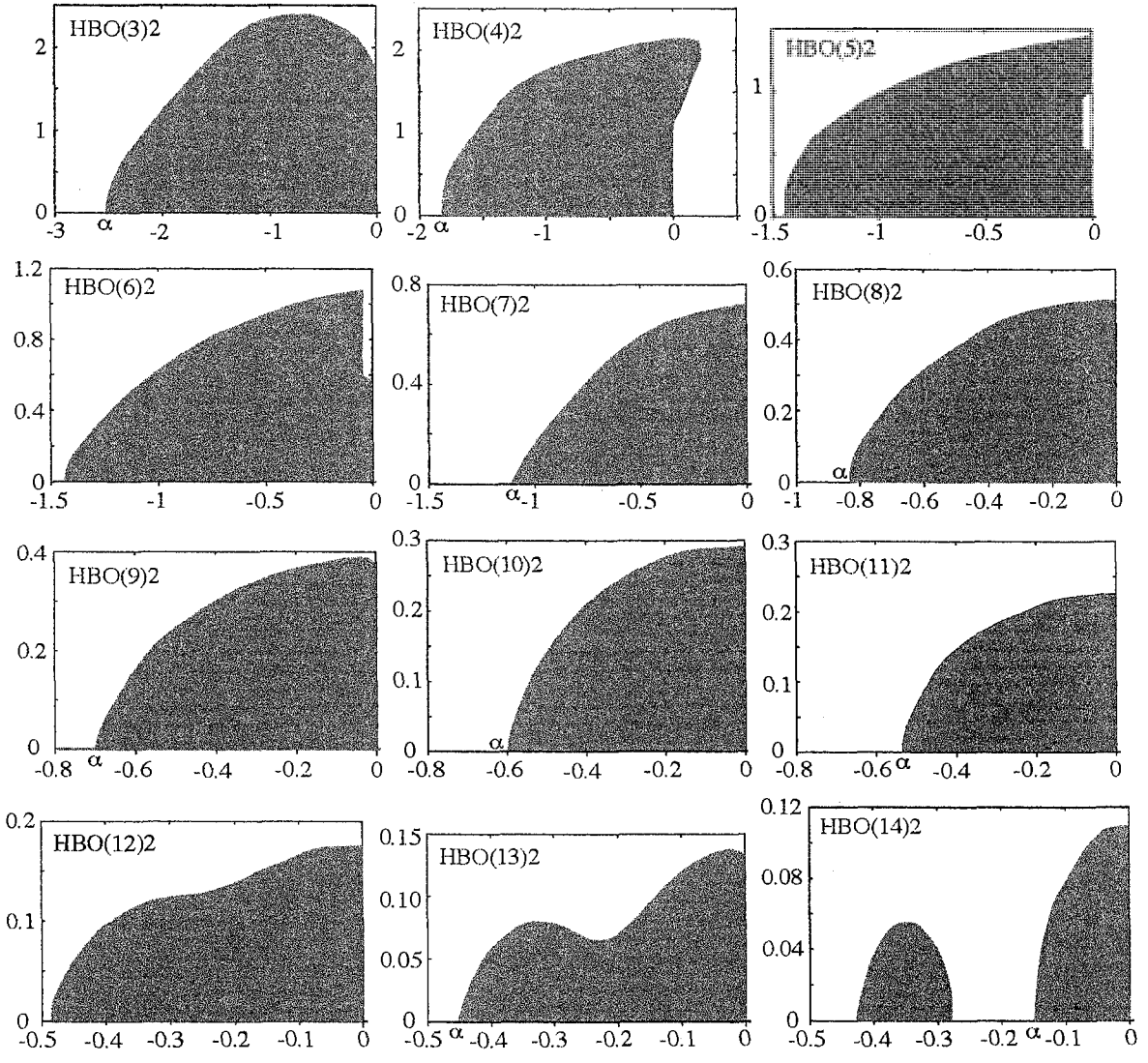


Figure 1: Upper part of regions of absolute stability, $R \in \mathbb{C}$, of HBO(3-14)2. Horizontal axis is real axis and vertical axis is imaginary axis.

Table 3: For order p, the table lists the scaled abscissae of absolute stability for HBO(p)2 and ABM($p, p-1$).

Order	$\alpha/3$	$\alpha/2$
p	HBO(p)2	ABM($p, p-1$)
3	-0.83	-1.20
4	-0.60	-0.97
5	-0.48	-0.70
6	-0.48	-0.52
7	-0.37	-0.39
8	-0.27	-0.30
9	-0.23	-0.22
10	-0.19	-0.17
11	-0.17	-0.13
12	-0.16	-0.11
13	-0.15	-0.03
14	-0.04	

Table 4: For given order p, the table lists the principal local truncation error coefficients (PLTC) of HBO(p)2 and the scaled norms $3 \times \|\text{PLTC}\|_2$ and $2 \times \|\text{PLTC}\|_2$ for HBO(p)2 and ABM($p, p-1$), respectively.

	PLTC of HBO(p)2		$3 \times \|\text{PLTC}\|_2$	$2 \times \|\text{PLTC}\|_2$
p	δ_1	δ_2	HBO(p)2	ABM($p, p-1$)
3	$-\frac{1}{72}$	$\frac{1}{24}$	13.17e-02	1.78e-01
4	$-\frac{1}{180}$	$\frac{197}{8640}$	7.05e-02	1.43e-01
5	$-\frac{13}{7200}$	$\frac{161}{17280}$	28.47e-03	1.22e-01
6	$-\frac{241}{302400}$	$\frac{3756}{754765}$	15.12e-03	1.09e-01
7	$-\frac{1283}{4233600}$	$\frac{3475}{1570026}$	6.69e-03	1.00e-01
8	$-\frac{221}{1587600}$	$\frac{899}{771028}$	3.51e-03	9.30e-02
9	$-\frac{91}{1256161}$	$\frac{2843}{4151123}$	20.67e-04	8.73e-02
10	$-\frac{333}{8077885}$	$\frac{8807}{20280549}$	13.08e-04	8.26e-02
11	$-\frac{763}{30405819}$	$\frac{7244}{24853301}$	8.76e-04	7.87e-02
12	$-\frac{530}{32913063}$	$\frac{1535}{7508037}$	6.15e-04	7.54e-02
13	$-\frac{567}{52587002}$	$\frac{607}{4086328}$	4.47e-04	7.25e-02
14	$-\frac{242}{32368071}$	$\frac{1207}{10866227}$	3.33e-04	

Chapter 4

Implementation and Numerical Results

4.1 Problems Used for Comparison

We list below standard test problems for ODE solvers. Many of these problems have been used to test HBO(4-14)2. In the next section, we report on the numerical performance of HBO(4-14)2 and DP(8,7) (both programmed in C) on all of the starred problems.

***CUBICWAVE** (CUBIC) Nonlinear wave equation in deep water [7].

***BRUSSELATOR** (BRUS) This reaction-diffusion partial differential equation is to be solved by the method of lines[1]. Thus the equation is transformed into a system of ordinary differential equations. [25, p. 248–249].

***ARENSTORF'S ORBITS** (AREN) Equations for the restricted three-body problem [25, p. 129–130].

[1]For a first-order partial differential equation (PDE), the method of lines discovers a line along which the PDE becomes an ordinary differential equation (ODE). Once the ODE is found, it can be solved along this line and transformed into a solution for the original PDE. If the line is a characteristic line or characteristic, the method is called method of characteristics.

THE PLEIADES (PLEI) A celestial mechanics problem for seven stars in the plane [25, p. 245–246].

RESTRICTED 3-BODY PROBLEM (R-3-BODY) Equations of motion of a restricted three-body problem [41, p. 246–247].

EULER'S EQUATION (EULER) Equations of motion for a rigid body without external force [41, p. 242–243].

The DETEST problems are divided into the following five classes, each containing five equations and are scaled so that $x_0 = 0$ and $x_f = 20$.

SINGLE EQUATIONS:

A3 $y' = -y\cos x, \quad y(0) = 1$ (An oscillatory problem).

SMALL SYSTEMS:

***B1** The growth of two conflicting populations [14, p. 102].

B4 The integral surface of a torus [16, p. 9].

ORBIT EQUATIONS:

***D1** Two-body problem with eccentricity $\epsilon = 0.1$.

D2 As in D1 except with eccentricity $\epsilon = 0.3$.

D3 As in D1 except with eccentricity $\epsilon = 0.5$.

D4 As in D1 except with eccentricity $\epsilon = 0.7$.

D5 As in D1 except with eccentricity $\epsilon = 0.9$.

HIGHER ORDER EQUATIONS:

E1 Derived from Bessel's equation of order 1/2 with origin shifted one unit to the left [14, p. 4, 69].

***E2** Derived from Van der Pol's equation [14, p. 358, 531].

CPU time, the number of function evaluations and the maximum global error are compared on the above starred problems to test HBO(4-14)2 as a VSVO solver on problems which require many function evaluations. Usually large step sizes can be used to solve easier problems with fewer function evaluations. From the comparison graph, HBO(4-14)2 obviously wins on expensive problems.

Computations were performed on a System with a dual 2.0 GHz Laptop 704 MB DDR2 running under Windows XP Professional and Matlab Version 6.5. Algorithms 2 and 3 were written in C and made into system-dependent Matlab `mex` files for speed.

4.2 HBO(4-14)2 against DP(8,7) in C

In this section, the numerical performance of HBO(4-14)2 and DP(8,7) in C is compared on two harder problems: the Brusselator and the Cubicwave, and on other easier problems mentioned above.

4.2.1 CPU time against maximum global error

The maximum global error is taken to be

$$\text{MGE} = \max_n \{\|y_n - z_n\|_\infty\},$$

where y_n is the numerical value obtained by HBO(4-14)2 and z_n is the "exact solution" obtained by `DP(8,7)` with stringent tolerance 5×10^{-14}.

In Fig. 2, the CPU time in seconds (horizontal axis) is plotted against the common logarithm of the Maximum Global Error (MGE) (vertical axis),

$$\log_{10}(\text{MGE}), \tag{51}$$

for six problems. By comparing the two curves for the Cubicwave in Fig. 2, HBO(4-14)2 wins for such expensive problem since the curve for HBO(4-14)2 lies to the left of the one for DP(8,7). However, DP(8,7) works better than HBO(4-14)2 for easy problems such as D1 and E2, because in HBO(4-14)2 many coefficients are calculated

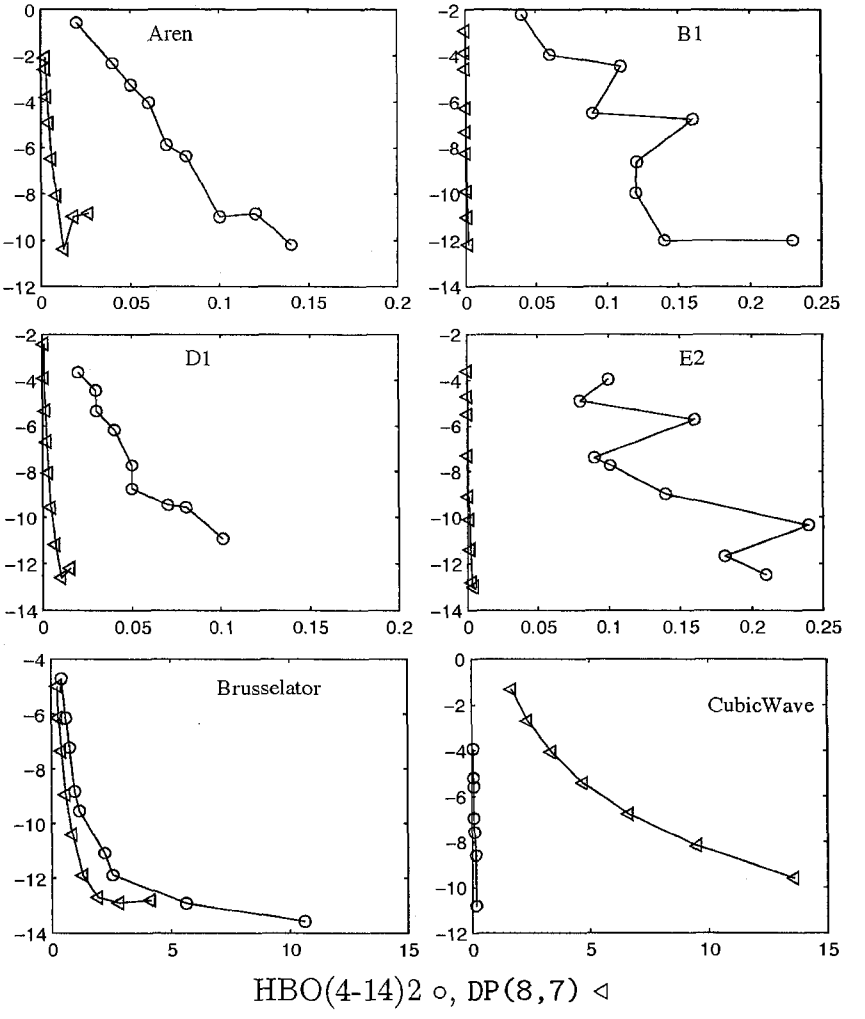


Figure 2: CPU time in seconds (horizontal axis) versus $\log_{10}$(MGE) (vertical axis) for Aren, B1, D1, E2, Brusselator and Cubicwave.

and the step sizes have to be changed. Thus, HBO(4-14)2 takes more CPU time than DP(8,7). Discarding CPU time, at small MGE, which corresponds to stringent tolerance, HBO(4-14)2 is better than DP(8,7) for the Brusselator. For the expensive CubicWave, HBO(4-14)2 wins over DP(8,7) at all tolerances.

4.2.2 NFE against maximum global error

We suppose that the costs of evaluating f and f' are the same. So, for HBO(4-14)2, NFE splits into 2/3 and 1/3 between f and f', respectively.

Figure 3 lists the NFE versus $\log_{10}$(MGE) for the six problems.

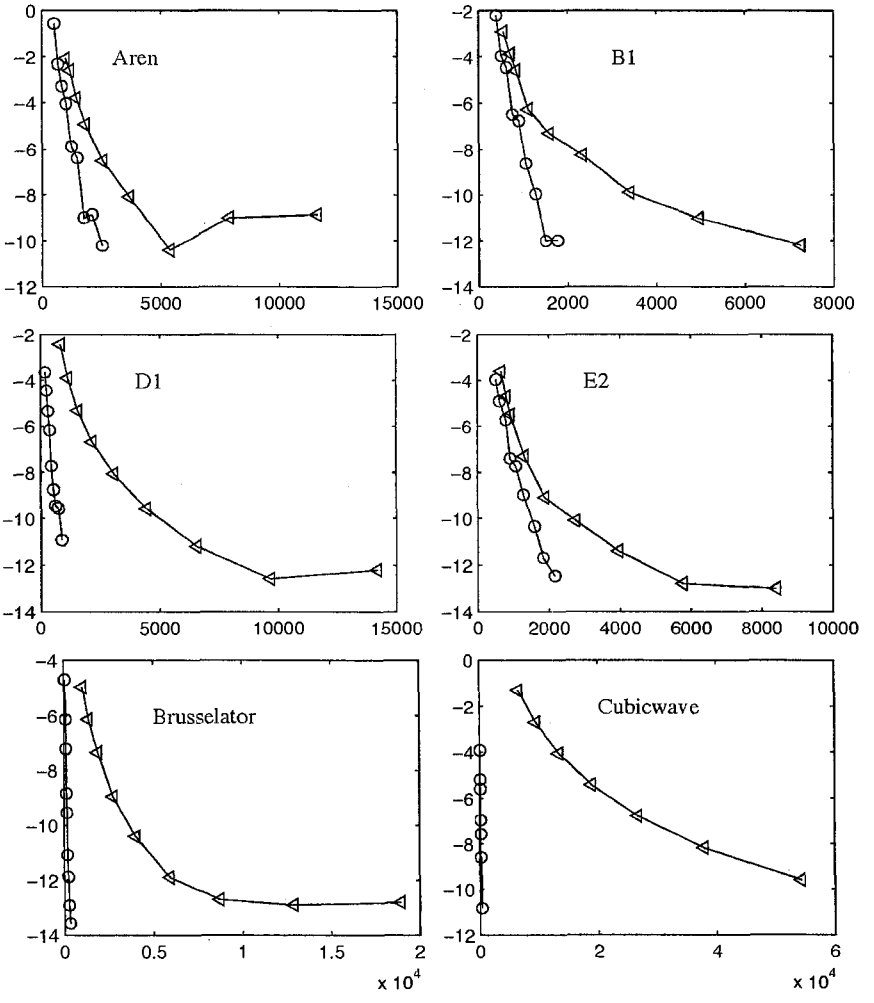
4.3 HBO(4-14)2 against HBO(4-14)3 in C

In this section, the 2-stage HBO(4-14)2 is compared with the 3-stage HBO(4-14)3. Figure 4 lists the NFE versus $\log_{10}$(MGE) for Van der Pol's equation for $\epsilon = 1, 3, 5, 7, 9$. As ϵ becomes bigger, HBO(4-14)2 gets better and finally wins over HBO(4-14)3.

4.4 The C Program

The structure of the C program for HBO(4-14)2 is briefly described in this section. The listing of my 4100-line C program is available on demand.

HBO(4-14)2 is implemented in C. The name is `HBO3_14_2Ed1_d2prgsim()`. The main subroutines of the calling hierarchy for this program are `HBO5d1d2initstepsprg()` which calls `HBO5d1d2varopstep()` to do initial HBO(5) steps. Then the program goes to the main loop and does the following three calls until the end (one integration step for each iteration): `HBO3_14_2Ed1d2corrs()` to calculate the IF coefficients; `HBO2_13_2Ed12prdc2sim()` to calculate the P2 coefficients; `HBO3_14_2Ed12_4SCcs()` to calculate the coefficients of the four local error estimators in order to control the step size and order. When a testing problem is selected, the main method of HBO(4-14)2 starts.



HBO(4-14)2 ○, DP(8,7) ◁

Figure 3: Number of function evaluation(NFE) (horizontal axis) versus $\log_{10}$(MGE) (vertical axis) for Aren, B1, D1, E2, Brusselator and Cubicwave.

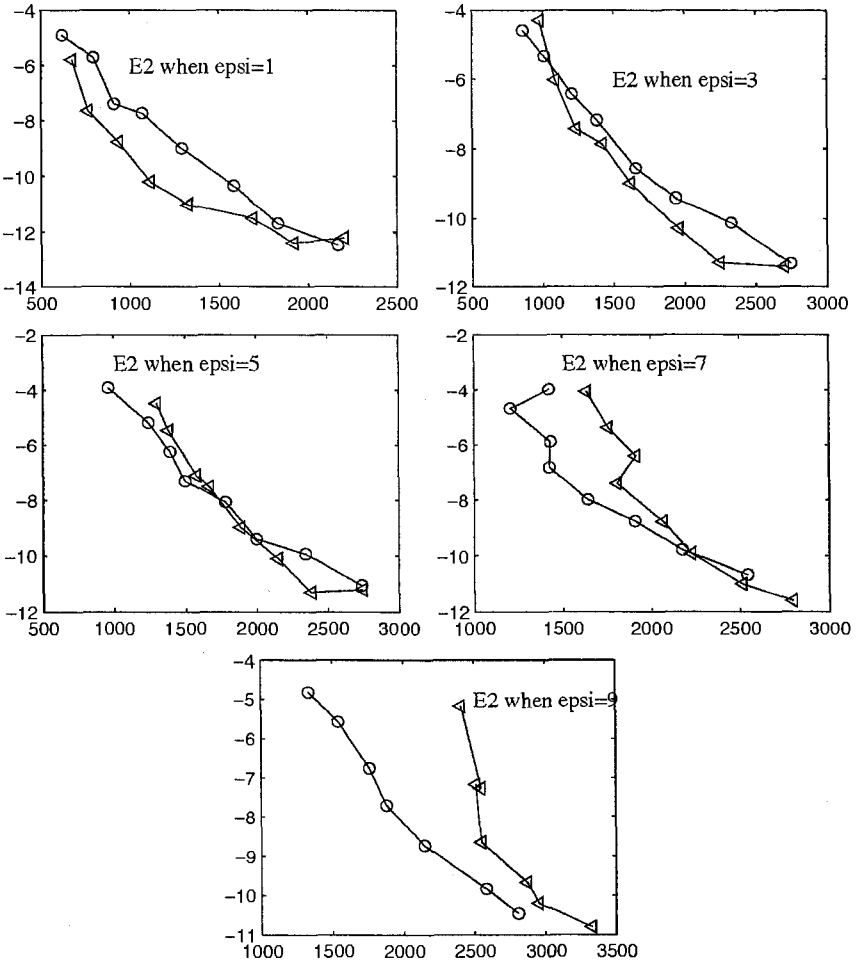


Figure 4: Number of function evaluation(NFE) (horizontal axis) versus $\log_{10}$(MGE) (vertical axis) for E2,where $\epsilon = 1, 3, 5, 7, 9$

The C program for HBO(4-14)2 uses the most fundamental and important characteristic of the C programming language and some C++ features, that is, procedure-oriented, pointers and generic templates. The GNU GCC compiler and advantages of C++ programming language are also discussed here.

Firstly, matrices are created by using a two-dimensional array as data structure. In order to make the two-dimensional matrices operations applicable to arbitrary number of rows and columns, the C++ operators `new` and `delete` are used with `pointers` in the program for allocating and de-allocating memory of matrices to avoid consuming vast amounts of virtual memory and destroying performance through page swapping.

Secondly, generic templates which can be used with any arbitrary type in C++ is used when implementing complex number operations in the program. C++ templates provide a way to re-use source code as opposed to inheritance which provide a way to re-use object code. In this C program, most data types are "double" (precision) in the matrix. However, for the sake of re-usability, flexibility and compatibility, the utilization of the template is necessary.

Thirdly, the program is compiled by `GNU GCC compiler` which is a free distributable C++ compiler. Since GNU GCC is supported by most operating systems (Windows/Linux/Unix/MacOS), the program should be recompilable and executed on all these platforms without changing the code.

The C++ programming language is object-oriented which is different from C. The object-oriented approach defines the status and behavior of the one type object which can be easily derived to represent one specific object. Implementing objects would possibly make matrix management and code re-utilization easier.

Chapter 5

Conclusion

A fast, variable-step, variable-order, 2-stage, Hermite–Birkhoff–Obrechkoff method of order 3 to 14 was constructed by solving Vandermonde-type systems satisfying multistep and Runge–Kutta type order conditions. The step size and order are controlled by four local error estimators. This method, in its vectorized Lagrange form, was tested on the Brusselator, Euler's equation, Arenstorf's orbits, the restricted three-body problem, the Pleiades, and the nonstiff DETEST problems: two-body problems of class D, the growth problem of two conflicting populations of class B and Van der Pol's equation of class E.

The new method was found to have larger scaled regions of absolute stability at higher order and lower scaled error norm than multistep methods.

Programmed in C, HBO(4-14)2 wins over DP(8,7)13M for expensive problems such as: the Brusselator and Cubicwave problems on a long period of time. However, DP(8,7)13M uses less CPU time than HBO(4-14)2 for simple problems such as D1–D5 and Arenstorf. Comparing with previous work of HBO(4-14)3 on Van der Pol's equation for different ϵ, it is found that HBO(4-14)3 is better than HBO(4-14)2 for smaller value of ϵ. As ϵ becomes bigger, HBO(4-14)2 gets better and finally wins over HBO(4-14)3.

The C program may be improved so that the CPU time could be reduced.

Appendix A

Algorithms

Algorithm 1 ***This algorithm constructs lower bidiagonal matrices*** L_k ***(applied to*** IF ***and*** P_2***) as functions of*** c_2 ***and*** η_j***,*** $j = 2 : 10$.

For $k = k_0^\ell : m-1$, 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 ***and*** P_2***) as functions of*** c_2 ***and*** η_j***,*** $j = 2 : 10$.

For $k = 2 : m-1$, do the following iteration:

For $j = m : -1 : k+1$, 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 overwrites*** $r = r(1:m)$ ***with*** $U_2 \cdots U_{m-1} D^{-1} L_{m-1} L_{m-2} \cdots L_{k_0^\ell} r$ ***in*** $O(m^2)$ ***operations for IF,*** P_2 ***and*** P_3.

Given $[\eta_2, \eta_3, \ldots, \eta_{10}]$ and $r = r(1:m)$, the following algorithm overwrites r with $U_2 \cdots U_{m-1} D^{-1} L_{m-1} L_{m-2} \cdots L_{k_0^\ell} r$.

Step (1) The following iteration overwrites $r = r(1:m)$ with $L_{m-1} L_{m-2} \cdots L_{k_0^\ell} r$:

for $k = k_0^\ell : m-1$, compute

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

Step (2) The following iteration overwrites $r = r(1:m)$ with $U_2 U_3 \cdots U_{m-1} D^{-1} r$:

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

For $k = m-1 : -1 : 2$, compute

$$\begin{aligned} r(i) &= r(i) U_k(i,i), && i = k+1 : m, \\ r(i) &= r(i) - r(i+1), && i = k : m-1. \end{aligned}$$

Algorithm 3 uses minimum storage since the solution is obtained by successively transforming the right-hand side into the solution vector. This is an advantage compared to generating $m \times m$ triangular matrices as an intermediate result at each integration step.

Appendix B

Matlab Programming

MATLAB is a high-level computer language for scientific computation and data visualization built around an interactive programming environment. It has several advantages over other programming languages:

- It contains a large number of functions that access proven numerical libraries, such as `LINPACK` and `EISPACK`. This means many common tasks can be achieved with a single function call.
- There is entensive graphics support that allows the results of computations to be plotted with a few statements.
- All numerical objects are treated as double-precision arrays. So there is no need to declare data types and carry out type conversions.

Algorithm 3 which solves systems IF and P_2 was programmed in C and compiled by the MATLAB `mex` command into `mex` files.

Algorithm 3 which solves the P_3 system and three additional similar systems to produce four local error estimators $\widetilde{y}_{n+1,p-j}$, $j = 1, 4$ of order $p - j$ was programmed in C and compiled by the MATLAB `mex` command into a `mex` file.

At runtime, the data of differential equations were the input. Then, at each integration step until completion of the integration, the previous `mex` files were called and run to calculate the values of the coefficients of IF, P_2, P_3 and three other step

control predictors. CPU time and the number of evaluations of $f(x,y)$ and $f'(x,y)$ for the runtime of Algorithm 3 were recorded.

MATLAB's `ode113` can be run with appropriate tolerance for comparison with HBO(4-14)2.

The elementary matrix functions L_k^ℓ and U_k^ℓ of η_j, for $j = 2, 3, \ldots, 10$ and $\ell = 1, 2, 3$, are constructed with Algorithms 1 and 2. These algorithms are not needed at runtime since these matrix functions are already implemented in the MATLAB `mex` files mentioned above.

Bibliography

[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. Applic., **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** (2005), 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, *Nonlinear wave groups in deep water*, manuscript.

[8] J. C. Butcher, *Coefficients for the study of Runge–Kutta integration processes*, J. Aust. Math. Soc., **3** (1963), pp. 185–201.

[9] J. C. Butcher, *A modified multistep method for the numerical integration of ordinary differential equations*, J. Assoc. Comput. Mach., **12** (1965), pp. 124–135.

[10] J. C. Butcher, *A multistep generalization of Runge–Kutta methods with four or five stages*, J. Assoc. Comput. Mach., **14** (1967), pp. 84–99.

[11] J. C. Butcher, *The numerical analysis of ordinary differential equations*, Chapter 4, Wiley, Chichester, 1987.

[12] M. Calvé and R. Vaillancourt, *Interpolants for Runge–Kutta pairs of order four and five*, Computing, **45** (1990), pp. 383–388.

[13] G. F. Corliss and Y. F. Chang, *Solving ordinary differential equations using Taylor series*, ACM Trans. Math. Software, 8(2) (1982), pp. 114–144.

[14] H. T. Davi, *Introduction to nonlinear differential and integral equations*, Dover, New York, 1962.

[15] P. Deuflhard, *Recent progress in extrapolation methods for ordinary differential equations*, SIAM Rev., **27** (1985), pp. 505–535.

[16] J. W. Daniel and R. E. Moore, *Computation and theory in ordinary differential equations*, Freeman, San Francisco, 1970.

[17] J. R. Dormand and P. J. Prince, *A reconsideration of some embedded Runge–Kutta formulae*, J. Comput. Appl. Math., **15** (1986), pp. 203–211.

[18] W. H. Enright, *Continuous numerical methods for ODEs with defect control*, J. Comput. Appl. Math., **125** (2000), pp. 159-170.

[19] W. H. Enright and T. E. Hull, *The test results on initial value methods for non-stiff ordinary differential equations*, SIAM J. Numer. Anal., **13** (1976), pp. 944–961.

[20] A. A. Frost and R. G. Pearson, *Kinetics and mechanism*, rev. ed., Wiley, New York, 1961.

[21] G. Galimberti and V. Pereyra, *Solving confluent Vandermonde systems of Hermite type*, Numer. Math, **18** (1971), pp. 44–60.

[22] C. W. Gear, *The numerical integration of ordinary differential equations*, Math. Comp., **21** (1967), pp. 146–156.

[23] C. W. Gear, *Numerical initial value problems in ordinary differential equations*, Prentice-Hall, Englewood Cliffs, NJ, 1971.

[24] G. H. Golub and C.F. Van Loan, *Matrix computations*, 3rd edition, The Johns Hopkins University Press, Baltimore, MD, 1996.

[25] E. Hairer, S. P. Nørsett and G. Wanner, *Solving ordinary differential equations I. Nonstiff problems*, Section III.8, Springer-Verlag. Berlin, 1987.

[26] 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.

[27] 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.

[28] 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.

[29] F. T. Krogh, *Variable order integrators for the numerical solution of ordinary differential equations.* Jet Propulsion Laboratory Tech. Memo., California Institute of Technology, Pasadena, 1969.

[30] J. D. Lambert, *Computational methods in ordinary differential equations*, Ch. 5, London, Wiley, 1973.

[31] J. D. Lambert, *Numerical methods for ordinary differential systems*, London, Wiley, 1991.

[32] 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.

[33] T. Nguyen-Ba and R. Vaillancourt, *Hermite–Birkhoff–Obrechkoff 3-stage 6-step ODE solver of order 14*, Can. Appl. Math. Quarterly, **13**(2) (2005), pp. 151–181.

[34] T. Nguyen-Ba, H.Yagoub, S. J. Desjardins and R. Vaillancourt, *Variable-step variable-order 4-stage Hermite-Birkhoff-Obrechkoff ODE solver of order 5 to 14*, Scientific Proceedings of Riga Technical University, in series "Computer Science" **30** (48) (2006), pp. 53–80.

[35] T. Nguyen-Ba, H. Yagoub, Y. Li and R. Vaillancourt, *Variable-step variable-order 3-stage Hermite-Birkhoff ODE solver of order 5 to 15*, Can. Appl. Math. Quarterly, 14(1) (2006), pp. 43–69.

[36] T. Nguyen-Ba, H.Yagoub, Y. Zhang and R. Vaillancourt, *Variable-step variable-order 3-stage Hermite-Birkhoff-Obrechkoff ODE solver of order 4 to 14*, Can. Appl. Math. Quarterly, 14(4) (2006), pp. 413–437.

[37] A. Nordsieck, *On numerical integration of ordinary differential equations*, Math. Comp., **16** (1962), pp. 22–49.

[38] L. R. Petzold, *A description of DASSL: A differential/algebraic system solver*, in Proceedings of IMACS World Congress, Montréal, Canada, 1982.

[39] P. J. Prince and J. R. Dormand, *High order embedded Runge–Kutta formulae*, J. Comput. Appl. Math., **7**(1) (1981), pp. 67–75.

[40] L. F. Shampine and L. S. Baca, *Fixed versus variable order Runge–Kutta*, ACM Trans. Math. Software, **12**(1) (1986), pp. 1–23.

[41] L. F. Shampine and M. K. Gordon, *Computer Solution of Ordinary Differential Equations: The Initial Value Problem*, Freeman, San Francisco, 1975.

[42] L. F. Shampine and M. W. Reichelt, *The Matlab ODE suite*, SIAM J. Sc. Comp., **18**(1) (1997), pp. 1–22.

[43] P. W. Sharp, *Numerical comparison of explicit Runge–Kutta pairs of orders four through eight*, Trans. on Mathematical Software, **17** (1991), pp. 387–409.

[44] M. Sofroniou and G. Spaletta, *Precise numerical computation*, J. Log. Algebr. Program, **64**(1) (2005), pp. 113–134.

[45] Y. Zhuang, *Variable-step variable-order 2-stage Hermite–Birkhoff–Obrechkoff ODE Solver of order 3 to 14*, Scientific Proceedings of Riga Technical University, **37**(50), (2008). In press.

[46] J. A. Zonneveld, *Automatic numerical integration*, Mathematical Centre Tracts No. 8, Mathematisch Centrum, Amsterdam (1964).