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

Vladan Bozic

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

M.Sc. (Mathematics)

GRADE / DEGREE

Department of Mathematics and Statistics

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Three-Stage Hermite-Birkhoff-Taylor ODE Solver With A C++ Program

TITRE DE LA THÈSE / TITLE OF THESIS

R. Vaillancourt

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

T. Giordano

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

T. Yeap

D. Amundsen

Gary W. Slater

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

THREE-STAGE HERMITE–BIRKHOFF–TAYLOR ODE SOLVER WITH A C++ PROGRAM

By
Vladan Bozic, B.Sc.
November 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 Mathematics

Your file Votre référence
ISBN: 978-0-494-48588-0
Our file Notre référence
ISBN: 978-0-494-48588-0

Abstract

One-step 3-stage Hermite–Birkhoff–Taylor methods, denoted by HBT(p)3, are constructed for solving nonstiff systems of first-order differential equations of the form $y' = f(x, y)$, $y(x_0) = y_0$. The method uses derivatives y' to $y^{(p-2)}$ as in Taylor methods and is combined with a 3-stage Runge–Kutta method of order 3. Forcing a Taylor expansion of the numerical solution to agree with an expansion of the true solution leads to Taylor- and Runge–Kutta-type order conditions, which are then reorganized into Vandermonde-type linear systems whose solutions are the coefficients of the method. The new method yields impressive results with regards to intervals of absolute stability. A detailed formulation of variable step size (VS) fixed order HBT(p)3 is presented, as well as the formulation of variable-step variable-order (VSVO) HBT(p)3. Several problems often used to test high order ODE solvers on the basis the number of steps, CPU time, maximum global error, and maximum global energy error are considered. The results stress that both VS and VSVO HBT(p)3 methods are superior to Dormand–Prince DP(8,7)13M and Taylor method of order p, denoted by T(p). To obtain results at high precision, high order VS and VSVO HBT(p)3 methods have been implemented in multiple precision. These numerical results clearly show the benefit of formulating a method by adding high order derivatives to Runge–Kutta method.

Acknowledgements

I would like to thank my supervisors, Dr. Rémi Vaillancourt and Dr. Thierry Giordano, for invaluable academic and financial support, and a great amount of guidance and understanding during the completion of this thesis. Their encouragement, suggestions and ideas have helped me immensely during the preparation of this work.

I would also like to thank Dr. Truong Nguyen-Ba for his insightful suggestions and technical help that made a large part of this work feasible; Dr. Martín Lara and Dr. Fernando Blesa for sharing their `FORTRAN` programs with me; and Mr. Alexey Beshenov for support with his `MPFRCPP` `C++`library.

Dedication

A well known fact is that there is no single ordinary differential equation solver that is applicable to every problem with desired precision and accuracy. We, the applied mathematicians, must seek to increase the amount of available numerical solvers so the folks in the field have more choice in the matter. Consider this as my contribution to the pot.

Contents

Abstract ii

Acknowledgements iii

Dedication iv

1 Introduction 5

1.1 Introduction and Motivation 5

1.2 Thesis Objective 7

1.3 Thesis Contribution 7

1.4 Thesis Organization 8

2 Preliminaries 10

2.1 Systems of Differential Equations 10

2.1.1 First-Order Systems 11

2.1.2 Linear Systems with Constant Coefficients 12

2.2 Numerical Methods 14

2.2.1 Euler's Method 14

2.2.2 Heun's Method 15

2.2.3 Runge–Kutta's Method 17

2.2.4 Adams' Methods 20

2.2.5 Taylor Series 23

3 Three-stage Hermite-Birkhoff-Taylor Method **28**
3.1 One-step HBT(p)3 28
3.2 Off-step points of HBT(p)3 29
3.3 Order Conditions for HBT(p)3 31
3.4 Vandermonde-type Formulation of HBT(p)3 38
3.4.1 Integration Formula IF 38
3.4.2 Predictor P_2 39
3.4.3 Predictor P_3 40

4 HBT(p)3 Implementation **42**
4.1 Zero Stability 42
4.2 Regions of Absolute Stability 43
4.3 Region of Absolute Stability for High Orders 50
4.4 Principal Local Error Term 54
4.5 Variable Step Size Formulation 58
4.6 Variable Order Formulation 61

5 Numerical Results **64**
5.1 Single Equations 67
5.2 Small Systems 67
5.3 Orbit Equations 71
5.4 Higher Order Equations 80
5.5 Dynamical Systems 80
5.5.1 Arenstorf's Orbits 82
5.5.2 Hénon–Heiles Problem 83
5.5.3 The Equatorial Main Problem 88

6 Conclusion and Future Work **93**

Bibliography **94**

List of Figures

1 Regions of absolute stability for HBT(p)3 of orders 12, 16, and 20 with real values on horizontal axis and imaginary values on vertical axis. . 48

2 Regions of absolute stability for HBT(p)3 of orders 20, 40, and 60 with real values on horizontal axis and imaginary values on vertical axis. . 49

3 $x_{\min}$ and the error of the linear approximation. 50

4 CPU time (horizontal axis) versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for problem A1 using (a) double precision and (b) multiple precision. . . 68

5 CPU time (horizontal axis) versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for problem B1. 70

6 CPU time (horizontal axis) versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for problem B5. 70

7 CPU time (horizontal axis) versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for Kepler's problem with varying eccentricity. 73

8 Number of steps (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for Kepler's problem with varying eccentricity. 74

9 Number of steps (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for Kepler's problem with varying eccentricity using multiple precision. 77

10 CPU time (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for Kepler's problem of eccentricity $\varepsilon = 0.999$ using VS fixed order and VSVO HBT(p)3 implementation. 79

11 The evolution of MGEE at different precision levels; time (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for Kepler's problem with varying eccentricity using multiple precision. 81
12 CPU time (horizontal axis) versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for problem E2. 82
13 CPU time (horizontal axis) versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for Arenstorf's orbits problem. 83
14 CPU time (horizontal axis) versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for the Hénon–Heiles problem. 84
15 Number of steps (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for the Hénon–Heiles problem. 85
16 Number of steps (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for the Hénon–Heiles problem using multiple precision. 86
17 The evolution of MGEE at different precision levels; time (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for the Hénon-Heiles problem using multiple precision. 86
18 CPU time (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for the Hénon–Heiles problem using VS fixed order and VSVO HBT(p)3 implementation. 87
19 CPU time (horizontal axis) versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for the equatorial main problem. 89
20 Number of steps (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for the equatorial main problem. 89
21 Number of steps (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for the equatorial main problem using multiple precision. 90
22 CPU time (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for the equatorial main problem using VS fixed order and VSVO HBT(p)3 implementation. 91
23 The evolution of MGEE at different precision levels; time (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for the equatorial main problem using multiple precision. 92

List of Tables

1 Butcher tableau for a 3-stage Runge–Kutta method of order 3. 28

2 Elementary differentials for HBT(2-6)3 with predictors of order at least 1. 35

3 Elementary differentials for HBT(3-6)3 with predictors of order at least 2. 36

4 Elementary differentials for HBT(4-6)3 with predictors of order at least 3. 37

5 Elementary differentials for HBT(5-6)3 with predictors of order at least 4. 37

6 Intervals of absolute stability for HBT(p)3 up to order 60. 51

7 PLTC of HBT(p)3 and the scaled norms for HBT(p)3 and ABM($p, p-1$). 59

8 Number of steps, CPU time and relative error of HBT(40)3 and T(40) using multiple precision for problem A1. 68

9 Number of steps, CPU time and maximum global error of HBT(12)3, T(12), and DP(8, 7)13M for Kepler's problem using double precision . 72

10 CPU PEG of HBT(12)3 over T(12) and DP(8, 7)13M for Kepler's problem with varying eccentricity. 72

11 NS PEG of HBT(12)3 over T(12) for Kepler's problem with varying eccentricity. 75

12 Number of steps, CPU time and maximum global energy error of HBT(p)3 and T(p) of orders $p = 20$ and $p = 40$ for Kepler's problem using multiple precision 76
13 NS PEG of HBT(p)3 over T(p) of orders $p = 20$ and $p = 40$ for Kepler's problem with varying eccentricity using multiple precision. 78
14 CPU time and MGEE of VS fixed order and VSVO HBT(p)3 for Kepler's problem with $\varepsilon = 0.999$ at various precision levels. 78
15 Number of steps, CPU time and maximum global energy error of HBT(20)3 and T(20) for the Hénon-Heiles problem using multiple precision. 85
16 CPU time and MGEE of VS fixed order and VSVO HBT(p)3 for the Hénon–Heiles problem at various precision levels. 87
17 Number of steps, CPU time and maximum global error of HBT(40)3 and T(40) for the equatorial main problem using multiple precision. . 90
18 CPU time and MGEE of VS fixed order and VSVO HBT(p)3 using multiple precision for the equatorial main problem. 91

Chapter 1

Introduction

1.1 Introduction and Motivation

A Taylor method of order $p-2$, denoted by T$(p-2)$, and a 3-stage Runge–Kutta method of order 3 are cast into a one-step 3-stage Hermite–Birkhoff–Taylor method of order p, aptly named HBT$(p)3$ because it uses Hermite–Birkhoff interpolation polynomials and the derivatives y' to $y^{(p-2)}$ for solving $y' = f(x, y)$ at step points x_n, like in Taylor methods. The link between the two types of methods is that values at off-step points are obtained by means of predictors which use values of derivatives of different orders at the current step point. By construction, HBT$(p)3$ uses lower order derivatives than the traditional Taylor method of order p [22].

Taylor methods have been an excellent choice in sensitivity analysis of ordinary differential equations (ODE) and delayed algebraic equations (DAE) [4], astronomical calculations [5], solving general problems [9], and validating solutions of ODEs/DAEs by means of interval analysis [18, 24]. Deprit and Zahar [12] proved that recurrent power series in Taylor methods are very effective in achieving high accuracy, even with a small value of computing time and large step size. On the other hand, multi-derivative, multistep methods, which are a new form of the classical Adams–Cowell methods, were introduced by Huang and Innanen [19]. Some of these methods have larger stability interval and smaller truncation error than classical multistep methods. The above results prompted the effective addition of high order derivatives to

an ODE solver.

HBT(p)3 is designed for solving nonstiff systems of first-order initial value problems of the form

$$y' = f(x, y), \quad y(x_0) = y_0, \quad \text{where} \quad ' = \frac{d}{dx}. \tag{1.1.1}$$

The high order derivatives y'' to $y^{(p-2)}$ can be obtained by differentiating $f(x, y(x))$ in the right-hand side of equation (1.1.1), but this approach is useful only in theoretical studies because of the computational complexity of high order derivatives.

Higher efficiency is achieved by extending Newton's approach [34, 36], and the process involves recursive computation of Taylor coefficients. This procedure is also known as automatic differentiation (AD) technique, and it can be applied to compute sums, differences, products and powers of power series, to name but a few. Formulae for generating these high order derivatives may be found in textbooks, such as [15], and in summarized form in Section 2.2.5. A limitation of applying the AD technique is that the right-hand side of the differential system must be manipulated in order to suit the problem. Although there have been advances made to combine algebraic manipulators with programming languages to ease the task of manipulation [22], if the right-hand side of the differential system involves more than just a combination of elementary series, the problem becomes significantly more challenging. At that point, a possible approach would be to find the asymptotic behaviour of the functions, and form an estimate that would be valid over certain intervals.

Forcing a Taylor expansion of the numerical solution to agree with an expansion of the true solution leads to a combination of Taylor- and Runge–Kutta-type order conditions which are reorganized into linear Vandermonde-type systems. The coefficients of this one-step method are obtained by means of Gaussian elimination, although different ways are possible. Moreover, with HBT(p)3 there are no rejected steps because the step size is chosen in order to obtain the required precision level once the series is generated. The C++ performances of HBT(p)3, Dormand–Prince DP(8,7)13M [33], and T(p) were compared on several problems frequently used to test higher order ODE solvers. With application of multiple precision libraries, the results for some of the problems were computed at very high accuracy using variable

step, variable order (VSVO) HBT(p)3. From the results, it can be concluded that HBT(p)3 requires fewer steps, uses less CPU time, and has higher accuracy than DP(8,7)13M and T(p).

1.2 Thesis Objective

The objectives of this thesis are:

- Discuss formulation of the HBT(p)3 method;
- Discuss the optimal choice of off-step points;
- Derive order conditions for HBT(p)3;
- Discuss absolute stability for low and high orders;
- Formulate the principal local error term;
- Discuss and implement variable step size and variable order;
- Formulate algorithms for HBT(p)3 method for non-stiff ordinary differential equations in `C++` using various levels of precision.

1.3 Thesis Contribution

This thesis has made the following contributions to the original research:

- Discussion of the optimal choice of off-step points and the proof of it;
- Discussion of absolute stability for low and high orders, in particular as the order grows large;
- Formulation of the principal local error term by using Butcher theory;
- Discussion and implementation of variable step size and variable order procedures in `C++` using various levels of precision;

- Producing results in tabular and visual formats to describe the achievements of the new method.

1.4 Thesis Organization

A major component of this thesis is based on peer-reviewed and published articles [25, 26, 27] that deal with HBT(p)3 methods with 4, 7 and 9 stages. An article that will encompass the discussion of HBT(p)3 VSVO implementation in multiple precision presented in this thesis is being prepared.

In Chapter 2, the preliminary information necessary for understanding the HBT(p)3 method is presented, with emphasis on the evolution of the numerical methods in terms of complexity. For most of the topics presented, a brief overview of the formulation is given. An introduction to recursive formulae used in automatic differentiation is given in Section 2.2.5.

In Chapter 3, HBT(p)3 is developed, with discussions on optimal choice of off-step points and order conditions. We remark that by choosing two ending off-step points to agree with Runge–Kutta method, then utilizing Gauss' integration formulae, the third off-step point will be a function of the order of the method. For predictors and integration formula, the Vandermonde-type formulation is given with solutions for the coefficients of explicit 3-stage Runge–Kutta method of order 3 and Taylor series expansion.

Implementation of HBT(p)3 is covered in Chapter 4. The concept of regions of absolute stability is discussed and applied to HBT(p)3. With the help of graphs, the regions of absolute stability are presented for variable-step fixed order HBT(p)3, and important conclusions are drawn regarding the shape of these regions for arbitrarily high orders based on analytical reasoning. Similarly to Runge–Kutta method of order 3, we form the principal local error term of HBT(p)3 in terms of elementary differentials. A detailed explanation of variable-step and variable-step variable-order HBT(p)3 formulation is given, as well as the algorithm used in the `C++` procedure to implement variable-step variable-order.

The performance of HBT(p)3 in comparison to Dormand–Prince and Taylor methods is presented in Chapter 5. The three methods are implemented in `C++`, in both double and multiple precision. The performance was tested on a set of problems commonly used in numerical analysis community, with results presented in a variety of formats for easier interpretation.

Finally, in Chapter 6, we provide closing remarks and point out to some future work that could be done to further improve the method.

Chapter 2

Preliminaries

Numerical solutions of differential equations are not only sought by mathematicians, but scientists in many other fields. Differential equations can be found virtually everywhere. They describe problems ranging from minuscule organisms such as bacteria, to very large, intricately connected problems in astronomy and planetary motion. With the advances in computing technology, more difficult problems can be tackled and more precise answers obtained. For the application of differential equations to be complete, a strong theoretical knowledge is always necessary.

2.1 Systems of Differential Equations

A fundamental concept related to differential equations is the concept of first-order systems. These can be viewed as building blocks because larger, more complex high-order differential equations and systems can be reduced to systems of first-order. This is often applied in practice in order to ease the computational tasks, reduce the amount of operations necessary, and because most software are written for first-order systems.

Before attempting to numerically solve a problem, an important question must be posed: does the solution *exist*? More to the point, it must be known beforehand whether the differential equations describing the problems have a unique solution. The following definition and theorem lend a hand in answering this question.

Definition 1 *A function $f : [a,b] \times \mathbb{R}^n \to \mathbb{R}^n$ is said to be Lipschitz continuous if there exists a constant $M > 0$, called Lipschitz constant, such that*

$$\|f(z) - f(y)\|_{\ell_2} \leq M\|z - y\|_{\ell_2}, \quad \text{for all} \quad y, z \in [a,b]. \tag{2.1.1}$$

Note: Throughout the thesis, the ℓ_2 norm will be used, but the indicator will be omitted for clarity unless otherwise stated.

Theorem 1 (Existence and uniqueness theorem) *Let $f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$, and consider the initial value problem*

$$y' = f(x,y), \quad y(x_0) = y_0. \tag{2.1.2}$$

If f is continuous and bounded on the region $D = [a,b] \times \mathbb{R}^m$,

$$\|f(x,y)\| \leq K, \quad (x,y) \in D,$$

and Lipschitz continuous with respect to y for every $(x,y) \in D$, then (2.1.2) admits one and only one solution for any $(x_0, y_0) \in D$ and $y(x)$ is continuous and differentiable in D.

The proof of Theorem 1 can be found in many books on differential equations. Unless otherwise stated, all functions in this thesis are assumed to be Lipschitz continuous in y.

2.1.1 First-Order Systems

Consider an m-dimensional system of first-order differential equations defined in vector form by

$$y' = f(x,y), \qquad f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m, \tag{2.1.3}$$

with initial condition vector $y(a) = \eta$. In vector form, $y = [y^1, y^2, \ldots, y^m]^T$ and $f = [f^1, f^2, \ldots, f^m]^T$, with superscripts denoting the corresponding equation in the system. The initial condition vector η is similarly defined as $\eta = [\eta^1, \eta^2, \ldots, \eta^m]^T$. Furthermore, if f^t for $t = 1, 2, \ldots, m$ depends on $y^1, y^2, \ldots, y^m$, the system is said to be coupled.

Given that Theorem 1 holds, a general solution of such a system will contain m arbitrary constants, and the unique solution is obtained by imposing m initial conditions.

2.1.2 Linear Systems with Constant Coefficients

A first-order system (2.1.3) is said to be *linear* if

$$y' = f(x, y) = A(x)\, y + \varphi(x) \tag{2.1.4}$$

where $A(x)$ is an $m \times m$ matrix and $\varphi : \mathbb{R} \to \mathbb{R}^m$. If $A(x) = A$ is independent of x, the system (2.1.4) is said to be *linear with constant coefficients.* Thus, (2.1.4) turns into

$$y' = A\, y + \varphi(x) \tag{2.1.5}$$

with the homogeneous equation being

$$y' = A\, y. \tag{2.1.6}$$

Given m linearly independent solutions of the homogeneous equation (2.1.6), $y^1(x), y^2(x), \ldots, y^m(x)$, every solution $y(x)$ of (2.1.6) can be expressed in the form

$$y(x) = c_1\, y^1(x) + c_2\, y^2(x) + \cdots + c_m\, y^m(x),$$

where c_i for $i = 1, 2, \ldots, m$ are arbitrary constants. Let $Y(x)$ be a matrix whose columns are $y^1(x), y^2(x), \ldots, y^m(x)$, and c be a column vector of constants c_i. Then the general solution to (2.1.6) can be written as

$$y(x) = Y(x)\, c.$$

Furthermore, since the columns of $Y(x)$ are linearly independent, $Y(x)$ is called a *fundamental matrix solution* of (2.1.6).

If the general solution of the homogeneous system is known, by the method of variation of parameters, a particular solution of (2.1.5) can be assumed to be of the form

$$y(x) = u_1(x)\, y^1(x) + u_2(x)\, y^2(x) + \cdots + u_m(x)\, y^m(x).$$

Let $u(x)$ be a column vector of the functions $u_i(x)$ for $i = 1, 2, \ldots, m$. Then, the particular solution to (2.1.5) can be written as

$$y(x) = Y(x)\, u(x). \tag{2.1.7}$$

Substituting (2.1.7) in (2.1.5) yields

$$Y'(x)\, u(x) + Y(x)\, u'(x) = A\, Y(x)\, u(x) + \varphi(x). \tag{2.1.8}$$

Since the matrix $Y(x)$ is a fundamental solution, $Y^{-1}(x)$ exists and $Y'(x) = AY(x)$, which further simplifies (2.1.8) to

$$Y(x)\, u'(x) = \varphi(x).$$

Multiplying both sides by $Y^{-1}(x)$ yields

$$u'(x) = Y^{-1}(x)\, \varphi(x).$$

When the above equation is integrated between two points x_0 and x, it yields

$$u(x) = \int_{x_0}^{x} Y^{-1}(s)\, \varphi(s)\, ds. \tag{2.1.9}$$

Once (2.1.9) is substituted in (2.1.7), the solution to the linear system with constant coefficients is obtained in the form

$$y(x) = Y(x)\, u + Y(x) \int_{x_0}^{x} Y^{-1}(s)\, \varphi(s)\, ds. \tag{2.1.10}$$

The first term in (2.1.10) is a solution of the homogeneous system, the second term being a particular solution. By the properties of the fundamental solution $Y(x)$, certain simplifications can be made to the homogeneous and particular solutions. For the homogeneous solution, it is known that the columns of $Y(x)$ are of the form

$$y^i(x) = \zeta^{(i)}\, e^{\lambda_i\, x}, \qquad i = 1, 2, \ldots, m$$

where λ_i are the eigenvalues and $\zeta^{(i)}$ are the eigenvectors or general eigenvectors of A. For the particular solution, let $Y(x) = e^{Ax}$ and $Y^{-1}(s) = e^{-As}$. This yields the final form of (2.1.10)

$$y(x) = e^{Ax}\, c_i + \int_{x_0}^{x} e^{A(x-s)}\, \varphi(s)\, ds.$$

2.2 Numerical Methods

Numerical methods are used to solve differential equations numerically. These methods can be single-step methods, using knowledge at a certain point to evaluate the value at the next point; or multistep methods that use knowledge of several previous points to obtain an evaluation at the next point. Single-step methods are self-starting in the sense that they need no external method to begin evaluations, while the multistep methods need a starting method to provide values at the points necessary in the evaluation. There is a variety of numerical methods that can be used to start the evaluations, with Runge–Kutta methods being used most often.

An important theorem which is often used when deriving numerical methods is Taylor's Theorem, and its proof can be found in [1].

Theorem 2 *Let $a < x < b$ and $f : (a, b) \to \mathbb{R}$ has continuous $(n+1)st$ derivative in the interval (a, b). If $P_n(x)$ is the Taylor polynomial of degree n for $f(x)$ about $x = a$, that is*

$$P_n(x) = f(a) + f'(a)\,(x-a) + \frac{f''(a)}{2!}\,(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}\,(x-a)^n,$$

then the formula $f(x) = P_n(x) + E_n(x)$ holds where the error term $E_n(x)$ is given by

$$E_n(x) = \frac{f^{(n+1)}(c)}{(n+1)!}\,(x-a)^{n+1}$$

for some c between a and b.

2.2.1 Euler's Method

Euler's method is the simplest method for finding numerical solutions of differential equations with a given initial value. In this method the value of the dependent variable is calculated by a straight line extrapolation from a previous point. This is why Euler's method is sometimes referred to as the Tangent Line Method.

Let the differential equation and its initial value be

$$y' = f(x, y), \qquad y(x_0) = y_0, \qquad f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}.$$

For simplicity, assume that $x_0 = 0$. Given this information, $y'(0) = f(0, y_0)$ can be evaluated. By using the first two terms of Taylor's expansion of $y(x_0 + h)$ and the assumption that $x_0 = 0$, an approximation to $y(h)$ can be calculated

$$y(x_0 + h) = y(h) \approx y_0 + h\,y'(0). \tag{2.2.1}$$

Let $x_1 = x_0 + h$, define $y(x_1) = y(x_0 + h) = y(h) \equiv y_1$ and rewrite $y'(0) = f(x_0, y_0)$. Then equation (2.2.1) becomes

$$y_1 = y_0 + h\,f(x_0, y_0). \tag{2.2.2}$$

Generalizing (2.2.2), the next value is defined in terms of the current value of y_n, and x_n by

$$y_{n+1} = y_n + h\,f(x_n, y_n), \qquad n = 0, 1, \ldots, \tag{2.2.3}$$

which advances the solution from x_n to $x_{n+1} = x_n + h$. Since the advance is one step forward, Euler's method appropriately falls into the category of single-step methods.

The local truncation error of this method can be found by considering the difference between the exact solution and the approximation given by (2.2.3). To obtain the exact solution of the problem, $y(x_{n+1})$ is expanded in a Taylor series and Theorem 2 is applied to yield

$$y(x_{n+1}) = y_n + h\,f(x_n, y_n) + O(h^2) \tag{2.2.4}$$

where x_b is some value between x_n and x_{n+1}. In (2.2.4), the first two terms represent Euler's method. The difference between equations (2.2.4) and (2.2.3) yields the local truncation error $O(h^2)$.

Even though Euler's method has been largely superseded in practice due to its large error, its beauty lies in its simplicity. This makes it a rather attractive choice for illustration and an easy implementation of numerical solvers.

2.2.2 Heun's Method

Heun's method is considered an improvement over Euler's method. This method is a single-step, two-stage method. Let the differential equation and its initial value be

represented as

$$y' = f(x, y), \qquad y(x_0) = y_0, \qquad f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}.$$

One way of obtaining the solution at the point (x_1, y_1) is to use the Fundamental Theorem of Calculus and integrate $y'(x)$ over the interval $[x_0, x_0 + h] = [x_0, x_1]$ to obtain

$$\int_{x_0}^{x_1} f(x, y(x))\, dx = \int_{x_0}^{x_1} y'(x)\, dx = y(x_1) - y(x_0).$$

From the above equation, we compute $y(x_1)$

$$y(x_1) = y(x_0) + \int_{x_0}^{x_1} f(x, y(x))\, dx.$$

If the integral is approximated by the trapezoidal rule, we obtain

$$y(x_1) \approx y(x_0) + \frac{1}{2}\left(\frac{x_1 - x_0}{b}\right)[f(x_0, y(x_0)) + f(x_1, y(x_1))] \tag{2.2.5}$$

where b is the number of area "strips." Let $b = 1$, and since $x_1 = x_0 + h$, (2.2.5) turns into

$$y(x_1) \approx y(x_0) + \frac{h}{2}[f(x_0, y(x_0)) + f(x_1, y(x_1))]. \tag{2.2.6}$$

The implicit equation (2.2.6) can be turned into an explicit equation by using Euler's method to evaluate $f(x_1, y(x_1))$:

$$y(x_1) \approx y(x_0) + \frac{h}{2}[f(x_0, y(x_0)) + f(x_0 + h, y(x_0) + h f(x_0, y_0))].$$

In general, improved Euler's method is represented by

$$y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y_n + h f(x_n, y_n))], \qquad n = 0, 1, \ldots. \tag{2.2.7}$$

The two aforementioned stages can be viewed as a predictor (Euler's method predicting the value at y_{n+1}) and corrector (Heun's method utilizing the trapezoidal rule to evaluate the y-value at the next step).

To analyze the local truncation error of Heun's method, consider the Taylor expansion of $y(x_n + h)$ with Theorem 2:

$$y(x_n + h) = y(x_n) + h y'(x_n) + \frac{h^2}{2} y''(x_n) + \frac{h^3}{6} y^{(3)}(\xi_1), \tag{2.2.8}$$

where $x_n \leq \xi_1 \leq x_n + h$.

Similarly, $y'(x_n + h)$ can be expressed as

$$y'(x_n + h) = y'(x_n) + h\,y''(x_n) + \frac{h^2}{2}y^{(3)}(\xi_2) \tag{2.2.9}$$

where $x_n \leq \xi_2 \leq x_n + h$. From equation (2.2.9)

$$y''(x_n) = \frac{y'(x_n + h) - y'(x_n)}{h} - \frac{h}{2}y^{(3)}(\xi_2). \tag{2.2.10}$$

If we substitute (2.2.10) into (2.2.8), we have:

$$y(x_n + h) = y(x_n) + h\,y'(x_n) + \frac{h^2}{2}\left[y'(x_n + h) - y'(x_n)\right] - h^3\left(\frac{y^{(3)}(\xi_2)}{4} - \frac{y^{(3)}(\xi_1)}{6}\right).$$

Simplifying the above equation yields

$$y_{n+1} = y_n + \frac{h}{2}\left[f(x_n, y_n) + f(x_n + h, y_n + h\,f(x_n, y_n))\right] - O(h^3). \tag{2.2.11}$$

Equation (2.2.11) matches with Heun's method (2.2.7) in the first two terms, thus the local truncation error is $O(h^3)$.

2.2.3 Runge–Kutta's Method

A general, s-stage Runge–Kutta method for the problem

$$y' = f(x, y), \qquad y(a) = \eta, \qquad f : \mathbb{R} \times \mathbb{R}^m \to \mathbb{R}^m$$

can be described by the following formula

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i\,k_i$$

where

$$k_i = f\left(x_n + c_i h, y_n + h\sum_{j=1}^{s} a_{ij}k_j\right).$$

The coefficients of a Runge–Kutta method can be summarized in a Butcher Tableau as follows:

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array}$$

where, for $i, j = 1, 2, \ldots, s$, c contains the c_i coefficients; A is an $s \times s$ matrix of a_{ij} coefficients; and b^T holds the weights b_i.

In the case of an explicit Runge–Kutta method, the matrix A is strictly lower triangular, i.e. $a_{ij} = 0$, $j \geq i$. An s-stage explicit Runge–Kutta method is described by the following Butcher tableau:

$$\begin{array}{c|ccccc} c_1 & & & & & \\ c_2 & a_{21} & & & & \\ c_3 & a_{31} & a_{32} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} & \\ \hline & b_1 & b_2 & \cdots & b_{s-1} & b_s \end{array}$$

Furthermore, for a Runge–Kutta method to be consistent, we shall assume that the following (the *row-sum*) condition holds [21]:

$$c_i = \sum_{j=1}^{i-1} a_{ij}, \qquad i = 2, 3, \ldots, s. \tag{2.2.12}$$

Three-stage explicit Runge–Kutta methods are described by the following formulae

$$\begin{aligned} y_{n+1} &= y_n + h(b_1 k_1 + b_2 k_2 + b_3 k_3), \\ k_1 &= f(x_n, y_n), \\ k_2 &= f(x_n + hc_2, y_n + ha_{21} k_1), \\ k_3 &= f(x_n + hc_3, y_n + h(a_{31} k_1 + a_{32} k_2)). \end{aligned} \tag{2.2.13}$$

Imposing condition (2.2.12) simplifies (2.2.13) to

$$\begin{aligned} k_1 &= f(x_n, y_n), \\ k_2 &= f(x_n + hc_2, y_n + hc_2 k_1), \\ k_3 &= f(x_n + hc_3, y_n + h(c_3 - a_{32}) k_1 + ha_{32} k_2). \end{aligned} \tag{2.2.14}$$

A Taylor expansion of the true solution $y(x_{n+1})$ about x_n yields

$$y(x_{n+1}) = y(x_n) + hf + \frac{1}{2} h^2 F + \frac{1}{6} h^3 (F f_y + G) + O(h^4), \tag{2.2.15}$$

where the elementary differentials F and G are defined as in [21]:

$$F = f_x + k_1 f_y, \qquad G = f_{xx} + 2k_1 f_{xy} + k_1^2 f_{yy}. \tag{2.2.16}$$

In order to find the local truncation error of a 3-stage Runge–Kutta method, the method needs to be expanded in Taylor series. Expanding the k_i terms yields

$$\begin{aligned}
k_1 &= f(x_n, y_n) = f, \\
k_2 &= f + hc_2(f_x + k_1 f_y) + \frac{1}{2} h^2 c_2^2 (f_{xx} + 2k_1 f_{xy} + k_1^2 f_{yy}) + O(h^3), \\
k_3 &= f + h\{c_3 f_x + [(c_3 - a_{32})k_1 + a_{32}k_2] f_y\} \\
&\quad + \frac{h^2}{2}\{c_3^2 f_{xx} + 2c_3[(c_3 - a_{32})k_1 + a_{32}k_2] f_{xy} + [(c_3 - a_{32})k_1 + a_{32}k_2]^2 f_{yy}\} + O(h^3).
\end{aligned}$$

If k_2 is rewritten using the elementary differentials defined in (2.2.16)

$$k_2 = f + hc_2 F + \frac{1}{2} h^2 c_2^2 G + O(h^3), \tag{2.2.17}$$

and substituted along with k_1 into the equation for k_3, the resulting equation is

$$k_3 = f + h\left(c_3 F + a_{32} h c_2 F f_y\right) + \frac{h^2}{2}\left(c_3^2 f_{xx} + 2c_3^2 k_1 f_{xy} + c_3^2 k_1^2 f_{yy}\right) + O(h^3).$$

Rewriting the equation for k_3 using elementary differentials defined in (2.2.16) and collecting the terms with like powers of h yields

$$k_3 = f + hc_3 F + \frac{h^2}{2}\left(2a_{32} c_2 F f_y + c_3^2 G\right) + O(h^3). \tag{2.2.18}$$

Substituting $k_1 = f$, (2.2.17) and (2.2.18) into (2.2.13) and making use of Theorem 2 yields the explicit 3-stage Runge–Kutta method is

$$\begin{aligned}
y_{n+1} &= y_n + h(b_1 + b_2 + b_3) f + h^2 (b_2 c_2 + b_3 c_3) F \\
&\quad + \frac{h^3}{2}\left[2a_{32} b_3 c_2 F f_y + \left(b_2 c_2^2 + b_3 c_3^2\right) G\right] + O(h^4).
\end{aligned} \tag{2.2.19}$$

Let $\widetilde{y}_{n+1}$ denote the value of y at x_{n+1} generated by the 3-stage Runge–Kutta method. To solve for the coefficient values, (2.2.15) and (2.2.19) can be matched.

For one-stage Runge–Kutta method $s = 1$, so $b_2 = b_3 = 0$. Thus, (2.2.19) reduces to

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

Matching the above equation with (2.2.15) yields $b_1 = 1$ and the local truncation error is $O(h^2)$. This is to be expected because the one-stage Runge–Kutta method is exactly Euler's method.

For two-stage Runge–Kutta methods $s = 2$, so $b_3 = 0$. Thus, (2.2.19) reduces to

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

Matching the above equation with (2.2.15) yields

$$b_1 + b_2 = 1, \qquad b_2 c_2 = 1/2.$$

Since there are two equations and three unknowns, there exists a one-parameter family of solutions. By equation (2.2.15), it can be concluded that no member of this family can achieve order higher than two.

For three-stage Runge–Kutta method $s = 3$, when (2.2.15) and (2.2.19) are matched, the following conditions must be satisfied:

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

Since there are four equations and six unknowns, there exists a two-parameter family of solutions. Again, by equation (2.2.15), it can be concluded that no member of this family can achieve order higher than three.

2.2.4 Adams' Methods

As seen previously, Euler's, Heun's and Runge–Kutta's methods are all single-step methods: y_n is needed to compute y_{n+1}. A multistep numerical methods utilizing predictor-corrector scheme, Adams' methods, which are sometimes referred to as Adams–Bashforth–Moulton, use an explicit Adams–Bashforth predictor and an implicit Adams–Moulton corrector. These methods use values of y_{n-j} at points x_{n-j} previous to x_n in the process of computing y_{n+1}. They are not self-starting as they

require a certain number of initial values which are usually found using a Runge–Kutta procedure. In order to distinguish between the explicit and implicit methods, let the superscript $*$ denote explicit methods.

To derive a k-step Adams–Bashforth predictor, let

$$y(x_{n+1}) - y(x_n) = \int_{x_n}^{x_{n+1}} y'(x)\,dx. \tag{2.2.20}$$

In this case, we seek an interpolating polynomial of the data

$$(x_n, f_n), (x_{n-1}, f_{n-1}), \ldots, (x_{n-k+1}, f_{n-k+1}).$$

By Newton–Gregory's backward interpolation formula, we have

$$I^*_{k-1}(x) \equiv P^*_{k-1}(r) = \sum_{i=0}^{k-1} (-1)^i \binom{-r}{i} \nabla^i f_n,$$

where

$$\binom{-r}{i} = (-1)^i \binom{r+k-1}{i},$$

and where r is defined by $x = x_n + rh$, and the backward differences is defined by the following formula

$$\nabla^m f_n = \sum_{i=0}^{m} (-1)^i \binom{m}{i} f_{n-i}. \tag{2.2.21}$$

The interpolating polynomial $y'(r)$ is generated at f_n

$$y'(x) \equiv y'(r) = f_n + r\nabla f_n + \frac{r(r+1)\nabla^2 f_n}{2} + \frac{r(r+1)(r+2)\nabla^3 f_n}{6} + \cdots.$$

Knowing that $x = x_n + rh$, r can be expressed as $r = (x - x_n)/h$; then $dx/dr = h$ and the limits of the integral in (2.2.20) can be changed to obtain

$$y_{n+1} - y_n = h \int_0^1 \left(f_n + r\nabla f_n + \frac{r(r+1)\nabla^2 f_n}{2} + \frac{r(r+1)(r+2)\nabla^3 f_n}{6} + \cdots \right) dr. \tag{2.2.22}$$

Integrating and evaluating (2.2.22) with respect to r yields the family of Adams–Bashforth methods

$$y_{n+1} - y_n = h \left(f_n + \frac{1}{2}\nabla f_n + \frac{5}{12}\nabla^2 f_n + \frac{3}{8}\nabla^3 f_n + \cdots \right). \tag{2.2.23}$$

By using equation (2.2.21) to generate the backward differences needed for (2.2.23), the Adams–Bashforth predictor is represented by

$$y_{n+1} = y_n + h\left(\frac{55}{24}f_n - \frac{59}{24}f_{n-1} + \frac{37}{24}f_{n-2} - \frac{3}{8}f_{n-3} + \cdots\right). \tag{2.2.24}$$

The Adams–Moulton corrector can be developed similarly. We start with equation (2.2.20), but seek an interpolating polynomial of the data

$$(x_{n+1}, f_{n+1}), (x_n, f_n), (x_{n-1}, f_{n-1}), \ldots, (x_{n-k+1}, f_{n-k+1}).$$

Now, instead of k data points, there are $k+1$, and the Newton–Gregory interpolation formula at x_{n+1} is

$$I_k(x) \equiv P_k(r) = \sum_{i=0}^{k}(-1)^i\binom{-r}{i}\nabla^i f_{n+1}.$$

The resulting interpolating polynomial is

$$y'(x) = y'(r) = f_{n+1} + r\nabla f_{n+1} + \frac{r(r+1)\nabla^2 f_{n+1}}{2} + \frac{r(r+1)(r+2)\nabla^3 f_{n+1}}{6} + \cdots.$$

Since f_{n+1} is used in the interpolating polynomial, the limits of the integral in (2.2.20) need to be updated to reflect the change. The interpolating polynomial substituted in (2.2.20) yields

$$y_{n+1} - y_n = h\int_{-1}^{0}\left(f_{n+1} + r\nabla f_{n+1} + \frac{r(r+1)\nabla^2 f_{n+1}}{2} + \frac{r(r+1)(r+2)\nabla^3 f_{n+1}}{6} + \cdots\right)dr. \tag{2.2.25}$$

Integrating and evaluating (2.2.25) with respect to r yields the family of Adams–Moulton methods

$$y_{n+1} - y_n = h\left(f_{n+1} - \frac{1}{2}\nabla f_{n+1} - \frac{1}{12}\nabla^2 f_{n+1} - \frac{1}{24}\nabla^3 f_{n+1} + \cdots\right). \tag{2.2.26}$$

By equation (2.2.21), the Adams–Moulton corrector is represented by

$$y_{n+1} = y_n + h\left(\frac{9}{24}f_{n+1} + \frac{19}{24}f_n - \frac{5}{24}f_{n-1} + \frac{1}{24}f_{n-2} + \cdots\right). \tag{2.2.27}$$

The approximation for the Adams–Bashforth predictor can be improved by adding the error term to (2.2.20) as per [13]:

$$y_{n+1} = y_n + \int_0^1 \sum_{i=0}^{k-1} (-1)^i \binom{-r}{i} \nabla^i f_n \, dr + (-1)^{k-1} h^{k+1} \int_0^1 \binom{-s}{k} f_n^{(k)} \, ds.$$

Based on the error term,

$$h^{k+1} \int_0^1 (-1)^{k-1} \binom{-s}{k} f_n^{(k)} \, ds = h^{k+1} \int_0^1 C_k^* f_n^{(k)} \, ds,$$

we can say that the k-step Adams–Bashforth predictor has order k, local truncation error $O(h^{k+1})$, and error constant C_k^*.

The same procedure can be repeated for the Adams–Moulton corrector to obtain:

$$y_{n+1} = y_n + \int_{-1}^0 \sum_{i=0}^{k} (-1)^i \binom{-r}{i} \nabla^i f_n \, dr + (-1)^k h^{k+1} \int_{-1}^0 \binom{-s+1}{k} f_n^{(k+1)} \, ds.$$

Based on the error term,

$$h^{k+1} \int_{-1}^0 (-1)^k \binom{-s+1}{k} f_n^{(k+1)} \, ds = h^{k+1} \int_{-1}^0 C_k f_n^{(k+1)} \, ds,$$

we can say that the k-step Adams–Moulton corrector has order $k+1$, local truncation error $O(h^{k+1})$, and error constant C_k.

2.2.5 Taylor Series

Taylor series are widely used in some fields, like astronomy, and the computation of its coefficients must be as economical as possible. Consider the solution of the initial value problem $y' = f(x, y)$, $y(x_0) = y_0$, where $f : [a, b] \times \mathbb{R}^n \to \mathbb{R}^n$ and $y : [a, b] \to \mathbb{R}^n$ in a Taylor series:

$$y(x_0 + h) = y(x_0) + y'(x_0)\, h + y''(x_0) \frac{h^2}{2!} + y'''(x_0) \frac{h^3}{3!} + \cdots . \tag{2.2.28}$$

By successive differentiation, the coefficients of (2.2.28) are found to be:

$$\begin{aligned} y' &= f, \\ y'' &= f_x + f_y y' = f_x + f_y f, \\ y''' &= f_{xx} + 2 f_{xy} f + f_{yy} f^2 + f_y (f_x + f_y f). \end{aligned}$$

For higher derivatives, the coefficients will become more complicated, thus not very economical. Higher efficiency is achieved by extending Newton's approach, which was rediscovered by Steffensen [36], and it involves recursive computation of Taylor coefficients.

Let

$$Y_i = \frac{1}{i!}\, y^{(i)}(x_0), \quad F_i = \frac{1}{i!}\, (f(x, y(x)))^{(i)} \Big|_{x=x_0} \tag{2.2.29}$$

be the Taylor coefficients of $y(x)$ and of $f(x, y(x))$ [15]. Then series (2.2.28) becomes

$$y(x_0 + h) = \sum_{i=0}^{\infty} h^i Y_i.$$

By $y' = f(x, y)$, $y(x_0) = y_0$ we can say that

$$Y_{i+1} = \frac{1}{i+1} F_i. \tag{2.2.30}$$

Suppose that $f(x, y)$ is the composition of a sequence of algebraic operations and elementary functions. With this, we can find formulae for generating the ith Taylor coefficient from the previous ones.

Before proceeding, let us consider a very useful concept. The term by term multiplication of two infinite series as follows

$$\begin{aligned}&(a_0 + a_1 x + a_2 x^2 + \cdots)(b_0 + b_1 x + b_2 x^2 + \cdots) \\ &\quad = a_0 b_0 + (a_0 b_1 + a_1 b_0)x + (a_0 b_2 + a_1 b_1 + a_2 b_0)x^2 + \cdots\end{aligned}$$

leads to the formula

$$\left(\sum_{n=0}^{\infty} a_n x^n\right)\left(\sum_{n=0}^{\infty} b_n x^n\right) = \sum_{n=0}^{\infty} c_n x^n \tag{2.2.31}$$

where

$$c_n = a_0 b_n + a_1 b_{n-1} + \cdots + a_n b_0 = \sum_{j=0}^{n} a_j b_{n-j}.$$

The resulting series in (2.2.31) is called the Cauchy product.

Recursive Formulae

As mentioned earlier, the application of automatic differentiation and recursive formulae to obtain high-order Taylor derivatives saves the amount of operations necessary

for the manipulation of the right-hand side of the differential system. The evident limitation of this procedure is that it is not applicable to every function, and in particular the functions that cannot be expressed or estimated in terms of the elementary functions.

If I is an open interval containing 0 and $g : I \to \mathbb{R}$ is analytic, then we denote by $\sum_{i\geq 0} g_i x^i$ the MacLaurin development of g in a neighbourhood of 0.

1. If $r(x) = p(x) \pm q(x)$, then

$$r_i = p_i \pm q_i, \quad i = 0, 1, \ldots \tag{2.2.32}$$

2. If $r(x) = p(x)\, q(x)$, then

$$r_i = \sum_{j=0}^{i} p_j\, q_{i-j}, \quad i = 0, 1, \ldots \tag{2.2.33}$$

The equation (2.2.33) is a result of direct application of the Cauchy product.

3. If $r(x) = p(x)/q(x)$, then

$$r_i = \frac{1}{q_0}\left(p_i - \sum_{j=0}^{i-1} r_j\, q_{i-j}\right), \quad i = 0, 1, \ldots \tag{2.2.34}$$

To obtain (2.2.34), first write $p(x) = r(x)\, q(x)$ and apply (2.2.33). Then, to solve for $r_i(x)$, evaluate the product at the last point in the summation to obtain

$$p_i = \sum_{j=0}^{i-1} r_j\, q_{i-j} + r_i\, q_0.$$

Solving the above equation for r_i yields (2.2.34).

4. If $r(x) = e^{p(x)}$, then

$$r_0 = e^{p_0}, \quad r_i = \frac{1}{i}\sum_{j=0}^{i-1}(i-j)\, r_j\, p_{i-j}, \quad i = 1, 2, \ldots \tag{2.2.35}$$

To obtain (2.2.35), take the derivative of $r(x) = e^{p(x)}$ to get $r'(x) = p'(x)\, r(x)$, then apply equations (2.2.30) and (2.2.33).

5. If $r(x) = \ln(p(x))$, then $r_0 = \ln(p_0)$ and

$$r_i = \frac{1}{p_0}\left(p_i - \frac{1}{i}\sum_{j=1}^{i-1}(i-j)\,p_j\,r_{i-j}\right), \quad i = 1, 2, \ldots \tag{2.2.36}$$

To obtain (2.2.36), rewrite $r(x) = \ln(p(x))$ as $p(x) = e^{r(x)}$ and rearrange (2.2.35) to solve for $p_i(x)$. Evaluate the summation at $j = 0$ to obtain the r_i term:

$$p_i = p_0\,r_i + \frac{1}{i}\sum_{j=1}^{i-1}(i-j)\,p_j\,r_{i-j}.$$

Solving the above equation for r_i yields (2.2.36).

6. If $r(x) = (p(x))^c$, where $c \neq 1$ is a constant, then $r_0 = (p_0)^c$ and

$$r_i = \frac{1}{i\,p_0}\left(\sum_{j=0}^{i-1}(ci - (c+1)j)\,r_j\,p_{i-j}\right), \quad i = 1, 2, \ldots \tag{2.2.37}$$

To obtain (2.2.37), take logarithm and derivative of $r(x) = (p(x))^c$ to obtain $p(x)r'(x) = cr(x)p'(x)$, then apply (2.2.33). Simplifying the result, and evaluating the summation at the point $j = i$ yields

$$\sum_{j=0}^{i-1}(j - (i-j)c)\,r_j\,p_{i-j} + i\,r_i\,p_0 = 0.$$

Solving the above equation for r_i yields (2.2.37).

7. If $r(x) = \cos(p(x))$, $s(x) = \sin(p(x))$ then for $i = 1, 2, \ldots$

$$\begin{aligned} r_0 &= \cos(p_0), \quad r_i = -\frac{1}{i}\sum_{j=0}^{i-1}(i-j)\,s_j\,p_{i-j} \\ s_0 &= \sin(p_0), \quad s_i = \frac{1}{i}\sum_{j=0}^{i-1}(i-j)\,r_j\,p_{i-j}. \end{aligned} \tag{2.2.38}$$

To obtain (2.2.38), we start with $r(x) = \cos(p(x))$ and $s(x) = \sin(p(x))$, take the derivatives to obtain

$$\begin{aligned} r'(x) &= -\sin(p(x))\,p'(x) = -s(x)\,p'(x) \\ s'(x) &= \cos(p(x))\,p'(x) = r(x)\,p'(x). \end{aligned}$$

With (2.2.33) applied to the above equations, the result is (2.2.38).

Computing Taylor coefficients this way is a very popular option when solving differential equations numerically. Many authors also provide automated procedures that will provide source code in `FORTRAN`[8, 9, 22] and `ANSI C` [23] for solving a first-order system of differential equations using Taylor's method.

Chapter 3

Three-stage Hermite-Birkhoff-Taylor Method

Following the procedure outlined in [26], in this chapter we shall construct one-step HBT(p)3, produce order conditions for HBT(p)3, and give Vandermonde-type formulation of HBT(p)3, where p is the order of the method.

3.1 One-step HBT(p)3

A general three-stage HBT(p) method to solve $y' = f(x, y)$ requires three formulae to perform integration from x_n to x_{n+1}: predictors P_2 and P_3, and integration formula IF. In the formulae, the constants a_{ij}, b_i and c_j are parameters of a 3-stage Runge–Kutta method of order 3 as listed in Table 1, also known as Butcher tableau; and γ_{lj} are the coefficients associated with the total derivatives of $f(x, y(x))$.

Table 1: Butcher tableau for a 3-stage Runge–Kutta method of order 3.

c_1			
c_2	a_{21}		
c_3	a_{31}	a_{32}	
	b_1	b_2	b_3

Hermite–Birkhoff polynomials of degree $l + (p-4)$ are used as predictors P_l to obtain y_{n+c_l} to orders $p-2$, with $p \geq 4$, for $l = 2, 3$, respectively,

$$y_{n+c_l} = y_n + h_{n+1} \sum_{j=1}^{l-1} a_{lj} f_{n+c_j} + \sum_{j=2}^{p-2} h_{n+1}^j \gamma_{lj} f_n^{(j-1)}, \quad l = 2, 3. \tag{3.1.1}$$

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

$$y_{n+1} = y_n + h_{n+1} \sum_{j=1}^{3} b_j f_{n+c_j} + \sum_{j=2}^{p-2} h_{n+1}^j \gamma_{1j} f_n^{(j-1)}. \tag{3.1.2}$$

By notation, y_{n+c_3} is different from y_{n+1}. Also, it is important to mention that derivatives f_n' to $f_n^{(p-3)}$ are computed only once per step at x_n. Numerical computations of higher derivatives are efficiently performed by applying the recurrent computation procedure outlined in Section 2.2.5.

3.2 Off-step points of HBT(p)3

The defining formulae of HBT(p)3 depend on the Runge–Kutta parameters listed in Table 1 and on the Taylor expansion coefficients γ_{lj}. In this thesis, the three off-step points are

$$c_1 = 0, \quad c_2 = \frac{p-1}{p+1}, \quad c_3 = 1. \tag{3.2.1}$$

The choice of the off-step $c_1 = 0$ follows directly from the Runge–Kutta part of HBT(p)3 method. The choice of off-step point $c_3 = 1$ is based on extensive numerical experimentation whose goal was simplification of the computations subject to satisfying the conditions of stability.

Remark 1 *The choice of the off-step point c_2 depends on the order p of the method, and is obtained by applying Gauss' integration formulae to the interpolating polynomial of HBT(p)3.*

Let I represent the interpolating polynomial of HBT(p)3,

$$I(x) = x^{p-2}(x-1)(x+a_0), \quad p \geq 3. \tag{3.2.2}$$

The equation $I(x) = 0$ has a root at $x = 0$ with multiplicity $(p-2)$. The root $x = 1$ corresponds to the choice of the off-step point c_3. The root $x = -a_0$ will correspond to the choice of c_2.

Let $w(x) \geq 0$ represent a weight function. By [11], it is desired to find the $2n$ values $w_1, w_2, \ldots, w_n$, $x_1, x_2, \ldots, x_n$ such that

$$\int_a^b w(x)f(x)\,dx = \sum_{i=1}^{n} w_i f(x_i) \tag{3.2.3}$$

for $f(x) = 1, x, x^2, \ldots, x^{2n-1}$. We proceed by replacing f in (3.2.3) by the interpolating polynomial (3.2.2) of degree p. Furthermore, we let the integration interval $[a, b] = [0, 1]$ and the weight function w be the constant function 1. In addition, we denote the ith moment by

$$m_j = \int_0^1 x^j\,dx = \frac{1}{j+1}, \tag{3.2.4}$$

and define

$$I(x) = \sum_{k=0}^{p} q_k x^k, \quad q_p = 1. \tag{3.2.5}$$

After expanding (3.2.5)

$$x^p + x^{p-1}(a_0 - 1) - a_0 x^{p-2} = q_0 + q_1 x + \cdots + q_{p-3}x^{p-3} + q_{p-2}x^{p-2} + q_{p-1}x^{p-1} + q_p x^p$$

we obtain the values of the coefficients q_k:

$$q_0 = q_1 = \cdots = q_{p-3} = 0, \quad q_{p-2} = -a_0, \quad q_{p-1} = a_0 - 1. \tag{3.2.6}$$

It is obvious by (3.2.6) that a_0 is the only unknown, and it is found by solving the following equation (see [11, p. 86])

$$m_0 q_0 + m_1 q_1 + \cdots + m_{p-3}q_{p-3} + m_{p-2}q_{p-2} + m_{p-1}q_{p-1} = -m_p. \tag{3.2.7}$$

By (3.2.6) and (3.2.4), we obtain

$$-\frac{1}{p-1}\,a_0 + \frac{1}{p}\,(a_0 - 1) = -\frac{1}{p+1}, \tag{3.2.8}$$

and

$$a_0 = -\frac{p-1}{p+1}, \tag{3.2.9}$$

which is one of the roots of I, and also the value of the off-step point c_2 as stated in (3.2.1).

3.3 Order Conditions for HBT(p)3

In order to obtain the order conditions for HBT(p)3, we first consider Taylor's expansion of the true solution $y(x_0+h)$ in terms of elementary differentials as per [6] and the monograph [7, Chapter 3]:

$$y(x_0+h) = y_0 + \sum_{r=1}^{\infty} \frac{h^r}{r!} D^{r-1}\mathbf{f} = y_0 + \sum_{i=1}^{\infty} \frac{h^{r(i)}}{r(i)!}\alpha(i)\mathbf{F}(i) \tag{3.3.1}$$

where $\mathbf{F}(1)(=\mathbf{f}), \mathbf{F}(2), \mathbf{F}(3), \ldots$ are the complete set of elementary differentials with orders $r(1)(=1), r(2), r(3), \ldots$ in non-decreasing order and $\alpha(i)$ the coefficient with which $\mathbf{F}(i)$ occurs in $D^{r(i)-1}\mathbf{f}$. The Taylor expansion of $y(x_0+h)$ up to $O(h^p)$ can be easily found from (3.3.1)

$$y(x_0+h) = y_0 + h\mathbf{f} + \frac{h^2}{2!}\{\mathbf{f}\} + \frac{h^3}{3!}\left(\{{}_2\mathbf{f}\}_2 + \{\mathbf{f}^2\}\right) + O(h^p). \tag{3.3.2}$$

The definition and results of the elementary differentials are based on Butcher's theory of rooted trees (see [7, pp. 79–104]). This subject is too large to be included in this thesis.

Next, consider the Taylor expansion of the numerical solution y_1 produced by HBT(p)3 in terms of elementary differentials, given that the value y_0 is obtained by a method of order at least p:

$$\begin{aligned} y_1 = y_0 &+ h\left(\sum_{i=1}^{3} b_i\right)\mathbf{f} + h^2\left(\sum_{i=1}^{3} b_i c_i + \gamma_{12}\right)\{\mathbf{f}\} \\ &+ h^3\left\{\left[\sum_{i=2}^{3} b_i\left(\sum_{j=1}^{i-1} a_{ij}c_j + \gamma_{j2}\right) + \gamma_{13}\right]\{{}_2\mathbf{f}\}_2 + \left(\sum_{i=1}^{3} b_i\frac{c_i^2}{2!} + \gamma_{13}\right)\{\mathbf{f}^2\}\right\} + \cdots + O(h^p). \end{aligned} \tag{3.3.3}$$

Finally, consider the difference between equations (3.3.2) and (3.3.3)

$$\begin{aligned} y(x_0+h) - y_1 \;=\; & h\left(1 - \sum_{i=1}^{3} b_i\right)\mathbf{f} + h^2\left\{\frac{1}{2} - \left(\sum_{i=1}^{3} b_i c_i + \gamma_{12}\right)\right\}\{\mathbf{f}\} \\ & + h^3\left\{\frac{1}{6} - \left[\sum_{i=2}^{3} b_i\left(\sum_{j=1}^{i-1} a_{ij}c_j + \gamma_{j2}\right) + \gamma_{13}\right]\right\}\{{}_2\mathbf{f}\}_2 \\ & + h^3\left\{\frac{1}{6} - \left(\sum_{i=1}^{3} b_i\frac{c_i^2}{2} + \gamma_{13}\right)\right\}\{\mathbf{f}^2\} + O(h^4). \end{aligned} \tag{3.3.4}$$

By forcing an expansion of the numerical solution produced by the formulae of HBT(p)3 to agree with the Taylor expansion of the true solution to order p, we obtain Runge–Kutta-type order conditions that must be satisfied by a general HBT(p)3 method of order p. These order conditions can be obtained from (3.3.4), as in [6].

As in similar searches for ODE solvers [6, 28, 29, 30, 31, 32] to make the solution of the order conditions feasible, without serious loss to the class of methods, we impose the following simplifying assumptions on HBT(p)3, with $\gamma_{i,1} = 0$ and c_j as defined in (3.2.1):

$$\sum_{j=1}^{i-1} a_{ij} c_j^k + k! \gamma_{i,k+1} = \frac{1}{k+1} c_i^{k+1}, \qquad \begin{cases} i = 2, 3, \\ k = 0, 1, 2, \ldots, p-3. \end{cases} \tag{3.3.5}$$

Based on these simplifying assumptions, all Runge–Kutta-type order conditions associated with the elementary differentials as seen in (3.3.4) of order $r = 1, 2, \ldots, p-1$ are equivalent to the order conditions

$$\sum_{i=1}^{3} b_i c_i^{r-1} + (r-1)! \gamma_{1,r} = \frac{1}{r}. \tag{3.3.6}$$

These order conditions are associated with the elementary differential $\mathbf{f}$ for $r = 1$ and $\{\mathbf{f}^{r-1}\}$ otherwise. For (3.3.6) to hold, the predictors P_2 and P_3 must be of order at least $p-2$. Thus, we obtain the following order conditions that must be satisfied by an HBT(p)3 method:

$$\sum_{i=1}^{3} b_i c_i^k + k! \gamma_{1,k+1} = \frac{1}{k+1}, \quad k = 0, 1, \ldots, p-1, \tag{3.3.7}$$

$$\sum_{i=2}^{3} b_i \left[\sum_{j=1}^{i-1} a_{ij} \frac{c_j^{p-2}}{(p-2)!} \right] = \frac{1}{p!}. \tag{3.3.8}$$

Equation (3.3.8) is the result of simplifying the equation

$$\sum_{i=2}^{3} b_i \left[\sum_{j=1}^{i-1} a_{ij} \frac{c_j^{p-2}}{(p-2)!} + \gamma_{i,p-1} \right] + \gamma_{1,p} = \frac{1}{p!}, \tag{3.3.9}$$

since $\gamma_{1,p-1} = \gamma_{1,p} = 0$ and $\gamma_{i,p-1} = \gamma_{i,p} = 0$ because the maximum order of the derivative is $p-2$.

The order conditions (3.3.7) and (3.3.8) can be derived by induction on the order p. First, let $p = 3$ to obtain

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

which are the order conditions of three-stage Runge–Kutta methods as seen in Section 2.2.3. Thus, the order conditions hold for $p = 3$ since there is no Taylor coefficients γ_{ij} in this case.

Second, suppose that the order conditions of HBT(p)3 of order p are of the form (3.3.7) and (3.3.8). The four order conditions associated with the four elementary differentials of order $(p+1)$ are:

$$\sum_{i=2}^{s} b_i c_i^p = \frac{1}{p+1}, \tag{3.3.10}$$

$$\sum_{i=2}^{s} b_i \left[\sum_{j=1}^{i-1} a_{ij} \frac{c_j^{p-1}}{(p-1)!} \right] = \frac{1}{(p+1)!}, \tag{3.3.11}$$

$$\sum_{i=2}^{s} b_i \frac{c_i}{p} \left[\sum_{j=1}^{i-1} a_{ij} \frac{c_j^{p-2}}{(p-2)!} \right] = \frac{1}{(p+1)!}, \tag{3.3.12}$$

$$\sum_{i=2}^{s} b_i \left\{ \sum_{j=1}^{i-1} a_{ij} \left[\sum_{k=1}^{j-1} a_{jk} \frac{c_k^{p-2}}{(p-2)!} \right] \right\} = \frac{1}{(p+1)!}, \tag{3.3.13}$$

where s is the number of stages.

As mentioned previously, the predictors are of order at least $(p-2)$, hence a method of order $(p+1)$ cannot have more than four additional order conditions as listed in (3.3.10) to (3.3.13). Other order conditions associated with elementary differentials of order $(p+1)$ are equivalent to these four order conditions.

Consider the case where the order of the predictors is increased to at least $(p-1)$. Then equation (3.3.12) is equivalent to equation (3.3.10), and equation (3.3.13) is

equivalent to equation (3.3.11). Thus, if HBT$(p+1)$ satisfies (3.3.10) and (3.3.11), then (3.3.12) and (3.3.13) will follow, respectively. Hence, (3.3.7), (3.3.8) and (3.3.10) to (3.3.13) will be satisfied if the following order conditions hold:

$$\sum_{i=1}^{3} b_i c_i^k + k!\,\gamma_{1,k+1} = \frac{1}{k+1}, \quad k = 0, 1, \ldots, p-1, \tag{3.3.14}$$

$$\sum_{i=2}^{3} b_i \left[\sum_{j=1}^{i-1} a_{ij} \frac{c_j^{p-2}}{(p-2)!} \right] = \frac{1}{p!}. \tag{3.3.15}$$

Hence, the order conditions (3.3.7) and (3.3.8) will always hold.

Let us consider an example of deriving order conditions for HBT$(p)3$ of order 1 to 6. By forcing a Taylor expansion of the numerical solution produced by HBT$(p)3$ formulae of order 6 to agree with a Taylor expansion of the true solution, we obtain Runge–Kutta-type order conditions to be satisfied by HBT(6)3 with predictors of order at least 1. These order conditions correspond to the elementary differentials in Table 2. In Table 2, $r(i)$ is the order of the elementary differential $\mathbf{F}^{(i)}$. The coefficients $\alpha(i)$ are considered as different labellings of a tree, and it can be computed using the relationship from [21]

$$\alpha(i) = \frac{r(i)!}{\sigma(i)\,\gamma(i)},$$

where $\sigma(i)$ is the symmetry and $\gamma(i)$ is the density of a tree i. Furthermore $\alpha(i)$, $\beta(i)$, and $\gamma(i)$ follow the relationship

$$\frac{1}{r(i)} \cdot \frac{\alpha(i)}{\beta(i)} = \frac{1}{\gamma(i)},$$

as defined in [6]. The correspondence between elementary differentials and rooted trees lead to Butcher's fundamental theorem [21, Theorem 5.3, p. 169].

Theorem 3 *A Runge–Kutta method has order p if $\phi(i) = 1/\gamma(i)$ holds for all trees of order $r(i) \leq p$ and does not hold for some tree of order $p+1$.*

Let the order of predictors be at least 2. This implies that the order conditions corresponding to i = 3, 5, 7, 9, 11, 13, 15, 16, 18, 20, 22, 24, 25, 27, 29, 31, 32, 33,

Table 2: Elementary differentials for HBT(2-6)3 with predictors of order at least 1.

i	$r(i)$	$\mathbf{F}^{(i)}$	$\phi^0(i)$	$\alpha(i)$	$\beta(i)$	$\gamma(i)$
1	1	$\mathbf{f}$	$\sum_i b_i$	1	1	1
2	2	$\{\mathbf{f}\}$	$\sum_i b_i c_i$	1	1	2
3	3	$\{{}_2\mathbf{f}\}_2$	$\sum_{ij} b_i a_{ij} c_j$	1	2	6
4	3	$\{\mathbf{f}^2\}$	$\sum_i b_i c_i^2$	1	1	3
5	4	$\{{}_3\mathbf{f}\}_3$	$\sum_{ijk} b_i a_{ij} a_{jk} c_k$	1	6	24
6	4	$\{{}_2\mathbf{f}^2\}_2$	$\sum_{ij} b_i a_{ij} c_j^2$	1	3	12
7	4	$\{\{\mathbf{f}\}\mathbf{f}\}$	$\sum_{ij} b_i c_i a_{ij} c_j$	3	6	8
8	4	$\{\mathbf{f}^3\}$	$\sum_i b_i c_i^3$	1	1	4
9	5	$\{{}_4\mathbf{f}\}_4$	$\sum_{ijkl} b_i a_{ij} a_{jk} a_{kl} c_l$	1	24	120
10	5	$\{{}_3\mathbf{f}^2\}_3$	$\sum_{ijk} b_i a_{ij} a_{jk} c_k^2$	1	12	60
11	5	$\{{}_2\{\mathbf{f}\}\mathbf{f}\}_2$	$\sum_{ijk} b_i a_{ij} c_j a_{jk} c_k$	3	24	40
12	5	$\{{}_2\mathbf{f}^3\}_2$	$\sum_{ij} b_i a_{ij} c_j^3$	1	4	20
13	5	$\{\{{}_2\mathbf{f}\}_2\mathbf{f}\}$	$\sum_{ijk} b_i c_i a_{ij} a_{jk} c_k$	4	24	30
14	5	$\{\{\mathbf{f}^2\}\mathbf{f}\}$	$\sum_{ij} b_i c_i a_{ij} c_j^2$	4	12	15
15	5	$\{\{\mathbf{f}\}^2\}$	$\sum_i b_i (\sum_j a_{ij} c_j)^2$	3	12	20
16	5	$\{\{\mathbf{f}\}\mathbf{f}^2\}$	$\sum_{ij} b_i c_i^2 a_{ij} c_j$	6	12	10
17	5	$\{\mathbf{f}^4\}$	$\sum_i b_i c_i^4$	1	1	5
18	6	$\{{}_5\mathbf{f}\}_5$	$\sum_{ijklm} b_i a_{ij} a_{jk} a_{kl} a_{lm} c_m$	1	120	720
19	6	$\{{}_4\mathbf{f}^2\}_4$	$\sum_{ijkl} b_i a_{ij} a_{jk} a_{kl} c_l^2$	1	60	360
20	6	$\{{}_3\{\mathbf{f}\}\mathbf{f}\}_3$	$\sum_{ijkl} b_i a_{ij} a_{jk} c_k a_{kl} c_l$	3	120	240
21	6	$\{{}_3\mathbf{f}^3\}_3$	$\sum_{ijk} b_i a_{ij} a_{jk} c_k^3$	1	20	120
22	6	$\{{}_2\{{}_2\mathbf{f}\}_2\mathbf{f}\}_2$	$\sum_{ijkl} b_i a_{ij} c_j a_{jk} a_{kl} c_l$	4	120	180
23	6	$\{{}_2\{\mathbf{f}^2\}\mathbf{f}\}_2$	$\sum_{ijk} b_i a_{ij} c_j a_{jk} c_k^2$	4	60	90
24	6	$\{{}_2\{\mathbf{f}\}^2\}_2$	$\sum_i b_i a_{ij} (\sum_k a_{jk} c_k)^2$	3	60	120
25	6	$\{{}_2\{\mathbf{f}\}\mathbf{f}^2\}_2$	$\sum_{ijk} b_i a_{ij} c_j^2 a_{jk} c_k$	6	60	60
26	6	$\{{}_2\mathbf{f}^4\}_2$	$\sum_{ij} b_i a_{ij} c_j^4$	1	5	30
27	6	$\{\{{}_3\mathbf{f}\}_3\mathbf{f}\}$	$\sum_{ijkl} b_i c_i a_{ij} a_{jk} a_{kl} c_l$	5	120	144
28	6	$\{\{{}_2\mathbf{f}^2\}_2\mathbf{f}\}$	$\sum_{ijk} b_i c_i a_{ij} a_{jk} c_k^2$	5	60	72
29	6	$\{\{\{\mathbf{f}\}\mathbf{f}\}\mathbf{f}\}$	$\sum_{ijk} b_i c_i a_{ij} c_j a_{jk} c_k$	15	120	48
30	6	$\{\{\mathbf{f}^3\}\mathbf{f}\}$	$\sum_{ij} b_i c_i a_{ij} c_j^3$	5	20	24
31	6	$\{\{{}_2\mathbf{f}\}_2\{\mathbf{f}\}\}$	$\sum_i b_i (\sum_{jk} a_{ij} a_{jk} c_k)(\sum_j a_{ij} c_j)$	10	120	72
32	6	$\{\{\mathbf{f}^2\}\{\mathbf{f}\}\}$	$\sum_i b_i (\sum_j a_{ij} c_j^2)(\sum_j a_{ij} c_j)$	10	60	36
33	6	$\{\{{}_2\mathbf{f}\}_2\mathbf{f}^2\}$	$\sum_{ijk} b_i c_i^2 a_{ij} a_{jk} c_k$	10	60	36
34	6	$\{\{\mathbf{f}^2\}\mathbf{f}^2\}$	$\sum_{ij} b_i c_i^2 a_{ij} c_j^2$	10	30	18
35	6	$\{\{\mathbf{f}\}^2\mathbf{f}\}$	$\sum_i b_i c_i (\sum_j a_{ij} c_j)^2$	15	60	24
36	6	$\{\{\mathbf{f}\}\mathbf{f}^3\}$	$\sum_{ijk} b_i c_i^3 a_{ij} c_j$	10	20	12
37	6	$\{\mathbf{f}^5\}$	$\sum_i b_i c_i^5$	1	1	6

Table 3: Elementary differentials for HBT(3-6)3 with predictors of order at least 2.

i	$r(i)$	$\mathbf{F}^{(i)}$	$\phi^0(i)$	$\alpha(i)$	$\beta(i)$	$\gamma(i)$
1	1	$\mathbf{f}$	$\sum_i b_i$	1	1	1
2	2	$\{\mathbf{f}\}$	$\sum_i b_i c_i$	1	1	2
4	3	$\{\mathbf{f}^2\}$	$\sum_i b_i c_i^2$	1	1	3
6	4	$\{{}_2\mathbf{f}^2\}_2$	$\sum_{ij} b_i a_{ij} c_j^2$	1	3	12
8	4	$\{\mathbf{f}^3\}$	$\sum_i b_i c_i^3$	1	1	4
10	5	$\{{}_3\mathbf{f}^2\}_3$	$\sum_{ijk} b_i a_{ij} a_{jk} c_k^2$	1	12	60
12	5	$\{{}_2\mathbf{f}^3\}_2$	$\sum_{ij} b_i a_{ij} c_j^3$	1	4	20
14	5	$\{\{\mathbf{f}^2\}\mathbf{f}\}$	$\sum_{ij} b_i c_i a_{ij} c_j^2$	4	12	15
17	5	$\{\mathbf{f}^4\}$	$\sum_i b_i c_i^4$	1	1	5
19	6	$\{{}_4\mathbf{f}^2\}_4$	$\sum_{ijkl} b_i a_{ij} a_{jk} a_{kl} c_l^2$	1	60	360
21	6	$\{{}_3\mathbf{f}^3\}_3$	$\sum_{ijk} b_i a_{ij} a_{jk} c_k^3$	1	20	120
23	6	$\{{}_2\{\mathbf{f}^2\}\mathbf{f}\}_2$	$\sum_{ijk} b_i a_{ij} c_j a_{jk} c_k^2$	4	60	90
26	6	$\{{}_2\mathbf{f}^4\}_2$	$\sum_{ij} b_i a_{ij} c_j^4$	1	5	30
28	6	$\{\{{}_2\mathbf{f}^2\}_2\mathbf{f}\}$	$\sum_{ijk} b_i c_i a_{ij} a_{jk} c_k^2$	5	60	72
30	6	$\{\{\mathbf{f}^3\}\mathbf{f}\}$	$\sum_{ij} b_i c_i a_{ij} c_j^3$	5	20	24
34	6	$\{\{\mathbf{f}^2\}\mathbf{f}^2\}$	$\sum_{ij} b_i c_i^2 a_{ij} c_j^2$	10	30	18
37	6	$\{\mathbf{f}^5\}$	$\sum_i b_i c_i^5$	1	1	6

35, 36, are equivalent to the order conditions corresponding to i = 4, 6, 8, 10, 12, 14, 17, 17, 19, 21, 23, 26, 26, 28, 30, 34, 34, 34, 37, 37, respectively (the repetitions are intended), and can be discarded. For example, the order condition $\sum_{ij} b_i a_{ij} c_j$ is equivalent to order condition $\sum_i b_i c_i^2$ due to simplifying condition (3.3.5). The remaining elementary differentials of order 1 to 6 are listed in Table 3 for predictors of order at least 2. The conditions corresponding to $i = 1, 2, 4, 6, 8$ in Table 3 can be used to write down the order conditions for HBT(4)3.

Let the order of predictors be at least 3. This implies that some conditions in Table 3 are not independent. The order conditions corresponding to i = 6, 10, 14, 19, 23, 28, 34, are equivalent to the order conditions corresponding to i = 8, 12, 17, 21, 26, 30, 37, respectively, and can be discarded. The resulting elementary differentials of order 1 to 6 for HBT(p)3 predictors of order at least 3 are listed in Table 4. In this table, the elementary differentials corresponding to i = 1, 2, 4, 8, 12, 17 can be used

Table 4: Elementary differentials for HBT(4-6)3 with predictors of order at least 3.

i	$r(i)$	$\mathbf{F}^{(i)}$	$\phi^0(i)$	$\alpha(i)$	$\beta(i)$	$\gamma(i)$
1	1	$\mathbf{f}$	$\sum_i b_i$	1	1	1
2	2	$\{\mathbf{f}\}$	$\sum_i b_i c_i$	1	1	2
4	3	$\{\mathbf{f}^2\}$	$\sum_i b_i c_i^2$	1	1	3
8	4	$\{\mathbf{f}^3\}$	$\sum_i b_i c_i^3$	1	1	4
12	5	$\{{}_2\mathbf{f}^3\}_2$	$\sum_{ij} b_i a_{ij} c_j^3$	1	4	20
17	5	$\{\mathbf{f}^4\}$	$\sum_i b_i c_i^4$	1	1	5
21	6	$\{{}_3\mathbf{f}^3\}_3$	$\sum_{ijk} b_i a_{ij} a_{jk} c_k^3$	1	20	120
26	6	$\{{}_2\mathbf{f}^4\}_2$	$\sum_{ij} b_i a_{ij} c_j^4$	1	5	30
30	6	$\{\{\mathbf{f}^3\}\mathbf{f}\}$	$\sum_{ij} b_i c_i a_{ij} c_j^3$	5	20	24
37	6	$\{\mathbf{f}^5\}$	$\sum_i b_i c_i^5$	1	1	6

Table 5: Elementary differentials for HBT(5-6)3 with predictors of order at least 4.

i	$r(i)$	$\mathbf{F}^{(i)}$	$\phi^0(i)$	$\alpha(i)$	$\beta(i)$	$\gamma(i)$
1	1	$\mathbf{f}$	$\sum_i b_i$	1	1	1
2	2	$\{\mathbf{f}\}$	$\sum_i b_i c_i$	1	1	2
4	3	$\{\mathbf{f}^2\}$	$\sum_i b_i c_i^2$	1	1	3
8	4	$\{\mathbf{f}^3\}$	$\sum_i b_i c_i^3$	1	1	4
17	5	$\{\mathbf{f}^4\}$	$\sum_i b_i c_i^4$	1	1	5
26	6	$\{{}_2\mathbf{f}^4\}_2$	$\sum_{ij} b_i a_{ij} c_j^4$	1	5	30
37	6	$\{\mathbf{f}^5\}$	$\sum_i b_i c_i^5$	1	1	6

to write down the order conditions for HBT(5)3.

Let the order of predictors be at least 4. This implies that some conditions in Table 4 are not independent. The order conditions corresponding to $i = 12, 21, 30$ are equivalent to conditions $i = 17, 26, 37$, respectively, and thus they can be discarded. The resulting elementary differentials of order 1 to 6 for HBT(p)3 with predictors of order at least 4 are listed in Table 5. These elementary differentials can be used to write down the order conditions for HBT(6)3.

3.4 Vandermonde-type Formulation of HBT(p)3

To efficiently find the values of the coefficients of the predictors and the integration formula (3.1.1) and (3.1.2), we shall rewrite them using Vandermonde-type notation. For a vector $x = [x_0, x_1, \ldots, x_n] \in \mathbb{R}^{n+1}$, a matrix $V \in \mathbb{R}^{(n+1)\times(n+1)}$ of the form

$$V \equiv V(x) = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_0 & x_1 & \cdots & x_n \\ \vdots & \vdots & & \vdots \\ x_0^n & x_1^n & \cdots & x_n^n \end{bmatrix},$$

is said to be a Vandermonde matrix [14]. The coefficients of the integration formula IF are organized in a Vandermonde-type matrix M^1; the coefficients of predictors P_2 and P_3 are organized in a Vandermonde-type matrices M^2 and M^3, respectively.

3.4.1 Integration Formula IF

To find the values of the coefficients of IF (3.1.2) that satisfy the order conditions (3.3.7), we consider the p-vector of the reordered coefficients of IF,

$$\boldsymbol{u}^1 = [b_3, b_2, b_1, \gamma_{12}, \gamma_{13}, \ldots, \gamma_{1,p-2}]^T,$$

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

$$M^1 \boldsymbol{u}^1 = \boldsymbol{r}^1, \tag{3.4.1}$$

where $M^1 \in \mathbb{R}^{p\times p}$, and for $s, t = 1, 2, \ldots, p$,

$$M^1 \equiv (m^1_{s,t}) = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & \cdots & 0 \\ c_3 & c_2 & 0 & 1 & 0 & & 0 \\ c_3^2/2! & c_2^2/2! & 0 & 0 & 1 & & 0 \\ \vdots & & & & & \ddots & \\ c_3^{p-3}/(p-3)! & c_2^{p-3}/(p-3)! & 0 & 0 & 0 & & 1 \\ \vdots & & & & & & \vdots \\ c_3^{p-1}/(p-1)! & c_2^{p-1}/(p-1)! & 0 & 0 & 0 & & 0 \end{bmatrix}. \tag{3.4.2}$$

The vector $\boldsymbol{r}^1 \in \mathbb{R}^p$ has components

$$r_1(i) = \frac{1}{i!}, \quad i = 1, 2, \ldots, p.$$

The solution of (3.4.1) is

$$\begin{aligned} b_3 &= \frac{1 - p\,(c_2 - 1)}{p\,(p-1)(c_2 - 1)}, \\ b_2 &= \frac{1 - b_3\,(p-1)}{(p-1)\,c_2^{p-2}}, \\ b_1 &= r_1(1) - \left(m^1_{1,1}\,b_3 + m^1_{1,2}\,b_2\right), \\ \gamma_{1i} &= r_1(i) - \left(m^1_{i,1}\,b_3 + m^1_{i,2}\,b_2\right), \quad i = 2, 3, \ldots, p-2. \end{aligned} \tag{3.4.3}$$

The leading error term of IF is of order $(p+2)$,

$$\left[b_3\,\frac{c_3^{p+1}}{(p+1)!} + b_2\,\frac{c_2^{p+1}}{(p+1)!} - \frac{1}{(p+2)!}\right] h_{n+1}^{p+2}\,y_n^{(p+2)}.$$

3.4.2 Predictor P_2

We expand $y(x_n + c_2 h_{n+1})$ in a truncated Taylor series,

$$y(x_n + c_2 h_{n+1}) = y_n + \sum_{j=1}^{p-2} \frac{c_2^j}{j!}\,h_{n+1}^j y_n^{(j)} + O(h_{n+1}^{p-1}),$$

and the right-hand side (3.1.1) with $\ell = 2$ in terms of $h_{n+1}^j y_n^{(j)}$. After replacing f_{n+c_1} by y_n' since $c_1 = 0$,

$$y_{n+c_2} = y_n + a_{21} h_{n+1} y_n' + \sum_{j=2}^{p-2} \gamma_{2j} h_{n+1}^j y_n^{(j)} \equiv \sum_{j=0}^{p-2} S_2(j) h_{n+1}^j y_n^{(j)},$$

where the convenient notation $S_2(j)$ is introduced for use below.

Equating coefficients of equal powers of h_{n+1} in the above sums, we have

$$S_2(j) = \begin{cases} 1, & j = 0, \\ a_{21} = c_2, & j = 1, \\ \gamma_{2j} = c_2^j/j!, & j = 2, \ldots, p-2. \end{cases} \tag{3.4.4}$$

Moreover, we shall take

$$S_2(j) = 0, \qquad j = p-1, p,$$

in subsequent formulae. The predictor P_2 is of order $p-2$ since $y(x_n+c_2h_{n+1})-y_{n+c_2} = O(h_{n+1}^{p-1})$.

3.4.3 Predictor P_3

We expand $y(x_n + c_3h_{n+1})$ in a truncated Taylor series,

$$y(x_n + c_3h_{n+1}) = y_n + c_3\, h_{n+1}y'_n + \sum_{j=2}^{p-2} \frac{c_3^j}{j!}\, h_{n+1}^j y_n^{(j)} + O(h_{n+1}^{p-1}),$$

and the right-hand side (3.1.1) with $\ell = 3$ in terms of $h_{n+1}^j y_n^{(j)}$. After replacing f_{n+c_1} by y'_n since $c_1 = 0$ and f_{n+c_2} by y'_{n+c_2},

$$y_{n+c_3} = y_n + h_{n+1}\left[a_{32}y'_{n+c_2} + a_{31}y'_n\right] + \sum_{j=2}^{p-1} \gamma_{3j}h_{n+1}^j y_n^{(j)} \equiv \sum_{j=0}^{p} S_3(j)h_{n+1}^j y_n^{(j)},$$

where the convenient notation $S_3(j)$ is introduced for use below.

Equating coefficients of equal powers of h_{n+1} in both expressions, we obtain the following linear system

$$M^3\boldsymbol{u}^3 = \boldsymbol{r}^3, \tag{3.4.5}$$

for the $(p-1)$ reordered coefficients of predictor P_3:

$$\boldsymbol{u}^3 = [a_{32}, a_{31}, \gamma_{32}, \gamma_{33}, \ldots, \gamma_{3,p-2}]^T,$$

where $M^3 \in \mathbb{R}^{(p-1)\times(p-1)}$,

$$M^3 \equiv (m_{s,t}^3) = \begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 \\ c_2 & 0 & 1 & 0 & & 0 \\ c_2^2/2! & 0 & 0 & 1 & & 0 \\ \vdots & & & & \ddots & \\ c_2^{p-3}/(p-3)! & 0 & 0 & 0 & & 1 \\ c_2^{p-2}/(p-2)! & 0 & 0 & 0 & \cdots & 0 \end{bmatrix}. \tag{3.4.6}$$

The vector $\boldsymbol{r}^3 \in \mathbb{R}^{p-1}$, has components

$$r_3(i) = \frac{c_3^i}{i!}, \quad i = 1, 2, \ldots, p-2,$$
$$r_3(p-1) = \frac{1}{b_3}\left[\frac{1}{p!} - b_2\, S_2(p-1)\right].$$

Thus, we have

$$S_3(j) = \begin{cases} 1, & j = 0, \\ a_{32} + a_{31} = c_3, & j = 1, \\ (M^3)_j^T\, \boldsymbol{u}^3 = \dfrac{c_3^j}{j!}, & j = 1, 2, \ldots, p-2, \\ a_{32} S_2(j-1), & j = p-1, p, p+1. \end{cases}$$

The predictor P_3 is of order $p-2$ since $y(x_n + c_3 h_{n+1}) - y_{n+c_3} = O(h_{n+1}^{p-1})$.

The solution of (3.4.5) is

$$\begin{aligned} a_{32} &= \frac{(p-2)!}{b_3\, c_2^{p-2}\, p!}, \\ a_{31} &= r_3(1) - m_{1,1}^3\, a_{32}, \\ \gamma_{3i} &= r_3(i) - m_{i,1}^3\, a_{32}, \quad i = 2, 3, \ldots, p-2. \end{aligned} \tag{3.4.7}$$

Chapter 4

HBT(p)3 Implementation

Let $y \in \mathbb{R}$ and $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. The multistep methods considered in this thesis for $y' = f(x, y)$ can be described by the general difference equation

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h \sum_{j=0}^{k} \beta_j f_{n+j}, \tag{4.0.1}$$

where the numerical approximations y_i and values f_i appear linearly; $\alpha_i, \beta_i \in \mathbb{R}$, h is the step size, and

$$f_i = f(x_i, y_i), \qquad x_i = x_0 + ih.$$

Furthermore, by [15] let $\alpha_k \neq 0$ and $|\alpha_0| + |\beta_0| > 0$. The linear k-step method (4.0.1) is said to be explicit if $\beta_k = 0$ and implicit if $\beta_k \neq 0$.

4.1 Zero Stability

In order to study the zero stability of method (4.0.1), we consider the behaviour of the solution of the simple linear differential equation with constant coefficient $y' = \lambda y$, $\lambda < 0$ as $n \to \infty$ with nh fixed. Then, as $h \to 0$ we obtain

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = 0. \tag{4.1.1}$$

Equation (4.1.1) can be interpreted as the numerical solution of (4.0.1) for the equation $y' = 0$. By Lagrange's method [15], we substitute $y_j = \zeta^j$ to obtain the polynomial equation

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = \alpha_0 \zeta^n + \alpha_1 \zeta^{n+1} + \cdots + \alpha_k \zeta^{n+k} = 0. \tag{4.1.2}$$

Equation (4.1.2) is satisfied if $\zeta = 0$ or if

$$\alpha_0 + \alpha_1 \zeta^1 + \cdots + \alpha_k \zeta^k = 0.$$

Thus, ζ must be a root of the first characteristic equation

$$\rho(\zeta) = \alpha_0 + \alpha_1 \zeta^1 + \cdots + \alpha_k \zeta^k = 0. \tag{4.1.3}$$

By [15], we assume that $y_n = n^{j-1}\zeta^n$, $j = 1, 2, \ldots, m$ are solutions of (4.1.1). Then, the following lemma and definition will hold.

Lemma 1 *Let $\zeta_1, \ldots, \zeta_l$ be the roots of* (4.1.3) *of respective multiplicity $m_1, \ldots, m_l$, where $\sum_{j=1}^{\ell} m_j = k$. Then the general solution of* (4.1.1) *is*

$$y_n = p_1(n)\zeta_1^n + \cdots + p_l(n)\zeta_l^n$$

where $p_j(n)$ are polynomials of degree $m_j - 1$.

Definition 2 (Zero stability) *The linear k-step method* (4.0.1) *is said to be zero stable if the first stability polynomial* (4.1.3) *satisfies the root condition, i.e.*

- *the roots of $\rho(\zeta)$ lie on or within the unit circle,*
- *the roots on the unit circle are simple.*

4.2 Regions of Absolute Stability

When dealing with numerical methods, there is often a need for a stability theory applicable when h takes a fixed non-zero value. We shall only consider linear stability for linear equations. The nonlinear stability theory [21, Chapter 7] is too difficult for our HBT(p)3 method and is beyond the scope of this thesis.

We consider a homogeneous linear system with real constant coefficients:

$$y' = Ay, \quad y : \mathbb{R} \to \mathbb{R}^m, \tag{4.2.1}$$

where the matrix $A \in \mathbb{R}^{m\times m}$ has distinct real or complex eigenvalues λ_i, $i = 1, 2, \ldots, m$, that satisfy

$$\Re\,\lambda_i < 0, \qquad i = 1, 2, \ldots, m,$$

and independent eigenvectors c_i, $i = 1, 2, \ldots, m$. The general solution of (4.2.1) is

$$y(x) = \sum_{i=1}^{m} \kappa_i\, e^{\lambda_i\, x}\, c_i,$$

where κ_i are arbitrary constants. Since $\Re\,\lambda_i < 0$,

$$\lim_{x\to\infty} y(x) = \lim_{x\to\infty} \sum_{i=1}^{m} \kappa_i\, e^{\lambda_i\, x}\, c_i \to 0,$$

thus, $\|y(x)\| \to 0$ as $x \to \infty$.

To find the conditions to be imposed on (4.2.1) so that $\|y_n\| \to 0$ as $n \to \infty$ and nh is fixed, we start with a linear k-step method defined in (4.0.1), and apply it to (4.2.1) to obtain

$$\sum_{j=0}^{k} (\alpha_j I - h\beta_j A)\, y_{n+j} = 0, \tag{4.2.2}$$

where I is the $m \times m$ identity matrix. By assumption, all λ_i eigenvalues are distinct, thus there exists a non-singular matrix Q such that

$$Q^{-1} A\, Q = \Lambda = \mathrm{diag}[\lambda_1, \lambda_2, \ldots, \lambda_m].$$

Multiply (4.2.2) by Q^{-1} and substitute $y_n = Qz_n$ to obtain

$$\sum_{j=0}^{k} (\alpha_j I - h\beta_j \Lambda)\, z_{n+j} = 0. \tag{4.2.3}$$

Since both I and Λ are diagonal matrices, the system (4.2.3) is decoupled and can be written as

$$\sum_{j=0}^{k} (\alpha_j - h\beta_j \lambda_i)^{\,i} z_{n+j} = 0, \qquad i = 1, 2, \ldots, m, \tag{4.2.4}$$

where $z_n = [{}^1z_n, {}^2z_n, \ldots, {}^mz_n]^T$.

Since $y_n = Qz_n$, $\|y_n\| \to 0$ as $n \to \infty$ only if $\|z_n\| \to 0$ as $n \to \infty$, which will hold only if all solutions $\{{}^iz_n\}$ satisfy

$$|{}^iz_n| \to 0, \quad \text{as} \quad n \to \infty,$$

for $i = 1, 2, \ldots, m$. By [21] we know that the general solution of each difference equation in (4.2.4) is of the form

$${}^iz_n = \sum_{s=1}^{k} \kappa_{is} r_s^n, \qquad i = 1, 2, \ldots, m,$$

where κ_{is} are arbitrary complex constants and r_s for $s = 1, 2, \ldots, k$ are the distinct roots of the characteristic polynomial

$$\sum_{j=0}^{k} (\alpha_j - h\beta_j\lambda_i)\, r^j = 0, \qquad i = 1, 2, \ldots, m. \tag{4.2.5}$$

Similarly to (4.1.3), we can define

$$\rho(r) = \sum_{j=0}^{k} \alpha_j\, r^j, \tag{4.2.6}$$

$$\sigma(r) = \sum_{j=0}^{k} \beta_j\, r^j, \tag{4.2.7}$$

The equations (4.2.6) and (4.2.7) are known as the first and second stability polynomials, respectively. The polynomial (4.2.5) can be written as

$$\pi(r, \hat{h}) = \rho(r) - \hat{h}\,\sigma(r), \qquad \hat{h} = \lambda h, \tag{4.2.8}$$

where λ represents any of the eigenvalues λ_i, $i = 1, 2, \ldots, m$, of the matrix A. The polynomial $\pi(r, \hat{h})$ is called the stability polynomial of the linear k-step method. The following definition ensures that condition $|z_n^i| \to 0$, as $n \to \infty$, as well as $|y_n| \to 0$ as $n \to \infty$ are satisfied.

Definition 3 *The linear k-step method* (4.0.1) *is said to be absolutely stable for given* $\hat{h}$*, if for that* $\hat{h}$ *all the roots of the stability polynomial* (4.2.8) *satisfy* $|r_s| < 1$, $s = 1, 2, \ldots k$, *and to be absolutely unstable for that* $\hat{h}$ *otherwise.*

The definition of the region of absolute stability follows.

Definition 4 *The linear k-step method* (4.0.1) *is said to have the region of absolute stability $\mathcal{R}_A$, where $\mathcal{R}_A$ is the region of the complex $\hat{h}$ plane for which the method is absolutely stable. The intersection of $\mathcal{R}_A$ with the real axis is called the interval of absolute stability.*

To obtain the region of absolute stability of HBT(p)3, we apply the predictors P_2 and P_3, and the integration formula IF with constant step size h to the linear test equation

$$y' = \lambda y, \qquad y_0 = 1. \tag{4.2.9}$$

Hence, in this case $f(x,y) = \lambda y$. Once the predictors P_2 and P_3 are applied to equation (4.2.9), we obtain

$$y_{n+c_l} = y_n + \lambda h_{n+1} \sum_{j=1}^{l-1} a_{lj} f_{n+c_j} + \sum_{j=2}^{p-2} (\lambda h_{n+1})^j \gamma_{lj} f_n^{(j-1)}, \quad l = 2, 3. \tag{4.2.10}$$

Similarly, the integration formula becomes

$$y_{n+1} = y_n + \lambda h_{n+1} \sum_{j=1}^{3} b_j y_{n+c_j} + \sum_{j=2}^{p-2} (\lambda h_{n+1})^j \gamma_{1j} y_n. \tag{4.2.11}$$

From (4.2.10) we can obtain y_{n+c_j} to be used in (4.2.11). Once the substitutions are made, and the terms in like powers of the product λh_{n+1} are collected, we obtain

$$\begin{aligned} y_{n+1} \;=\; & y_n + \lambda h_{n+1} y_n \left(\sum_{i=1}^{3} b_i \right) + (\lambda h_{n+1})^2 y_n \left(\sum_{i=2}^{3} b_i \sum_{j=1}^{i-1} a_{ij} \right) \\ & + (\lambda h_{n+1})^3 y_n \left[\sum_{i=2}^{3} b_i \left(\sum_{j=1}^{i-1} a_{ij} a_{i,j+1} + \gamma_{i2} \right) + \gamma_{13} \right] \\ & + y_n \sum_{k=4}^{p} (\lambda h_{n+1})^k \left[\sum_{i=2}^{3} b_i \left(\sum_{j=2}^{i-1} a_{ij} \gamma_{j,k-2} + \gamma_{i,k-1} \right) + \gamma_{1k} \right]. \end{aligned} \tag{4.2.12}$$

If we collect the coefficients of y_n in (4.2.12) and denote the result by r_s, we obtain a first-order difference equation

$$y_{n+1} - r_s y_n = 0, \tag{4.2.13}$$

where r_s is defined as

$$r_s = 1 + \sum_{j=1}^{p} s_j (\lambda h)^j. \tag{4.2.14}$$

The coefficients of (4.2.14) are defined as

$$\begin{aligned}
s_1 &= \sum_{i=1}^{3} b_i, \\
s_2 &= \sum_{i=2}^{3} b_i \sum_{j=1}^{i-1} a_{ij} + \gamma_{12}, \\
s_3 &= \sum_{i=2}^{3} b_i \left(\sum_{j=1}^{i-1} a_{i-1,j}\, a_{i,j+1} + \gamma_{i2} \right) + \gamma_{13}, \\
s_k &= \sum_{i=2}^{3} b_i \left(\sum_{j=2}^{i-1} a_{ij} \gamma_{j,k-2} + \gamma_{i,k-1} \right) + \gamma_{1k}, \qquad \text{for } k = 4, 5, \ldots, p.
\end{aligned} \tag{4.2.15}$$

The characteristic polynomial of the difference equation (4.2.13) is

$$\pi(r, \hat{h}) = r - r_s = 0, \tag{4.2.16}$$

with the root $r = r_s$. By Definition 4, a complex number λh is in the region of absolute stability $\mathcal{R}_A$ if the root r_s satisfies the criteria $|r_s| < 1$.

To obtain the regions of absolute stability of an arbitrary HBT(p)3 method, the root of (4.2.16) is plotted in the complex plane for all values of λh satisfying $|r_s| < 1$. A procedure has been written in C++ for creating a MATLAB code that plots values of λh, thus forming the region of absolute stability $\mathcal{R}_A$. Using MATLAB, the regions of absolute stability for HBT(p)3 of orders $p = 12, 16$, and 20 are shown in Figure 1. In Figure 2, the evolution of the size of $\mathcal{R}_A$ is shown for HBT(p)3 of orders $p = 20, 40$, and 60.

It can be noticed that the regions of absolute stability increase as the order increases, and that they tend to a quarter-circle. The radii of the quarter-circles is about the size of the calculated lower end of the interval of absolute stability $x_{\min}$, which is the complex number λh with smallest absolute real part. It could also be noted that the isolated regions of absolute stability, i.e. the "moons," also appear in Dormand–Prince's DP(5,4)7M [21, p. 205]. Since we are using a scanning method to

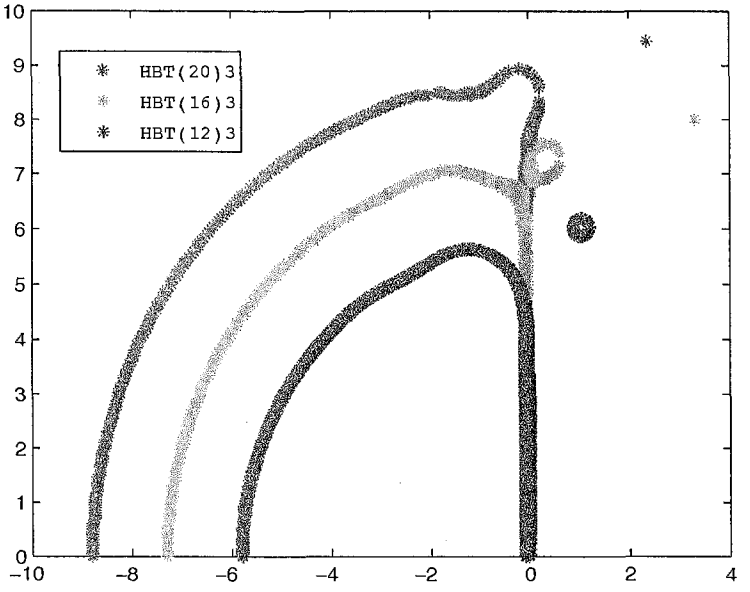


Figure 1: Regions of absolute stability for HBT(p)3 of orders 12, 16, and 20 with real values on horizontal axis and imaginary values on vertical axis.

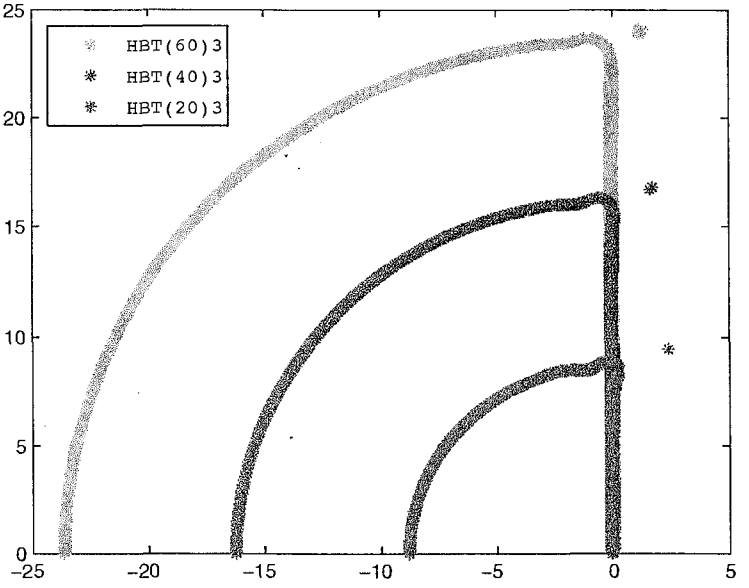


Figure 2: Regions of absolute stability for HBT(p)3 of orders 20, 40, and 60 with real values on horizontal axis and imaginary values on vertical axis.

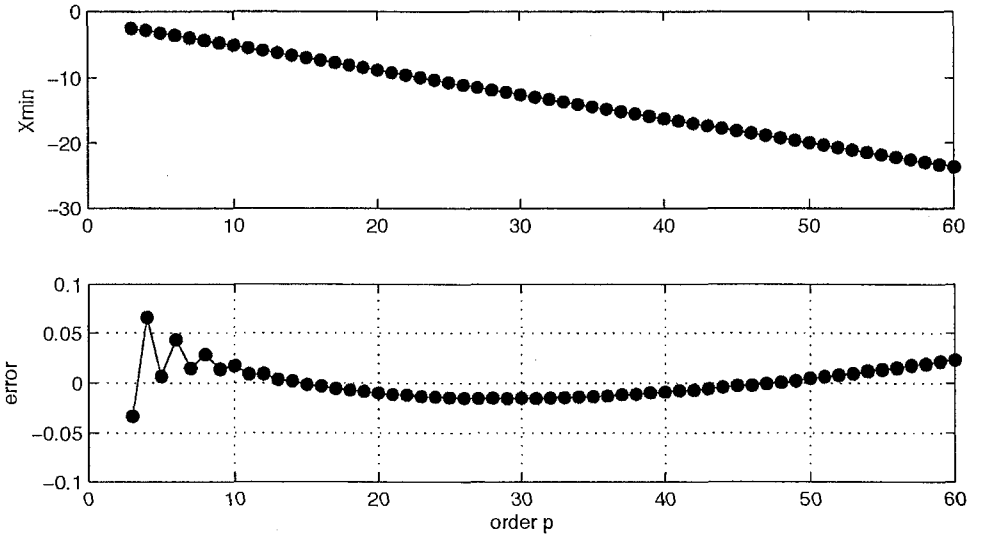


Figure 3: $x_{\min}$ and the error of the linear approximation.

find the regions of absolute stability, it is expected that the "moons" will appear even as the order increases. This is confirmed in Figure 2.

For order p, Table 6 lists calculated abscissae of absolute stability $x_{\min}$, and the scaled intervals of absolute stability $x_{\min}/p$ of HBT(p)3 up to order 60. In Figure 3 we plot $x_{\min}$ values, and it appears that the curve is a linear function of the order of HBT(p)3 with linear ℓ_2 fit

$$x_{\min}(p) \approx x_{\text{approx}}(p) = -1.3611 - 0.3725p. \tag{4.2.17}$$

Also, the error of the approximation x_{approx} of (4.2.17) is shown in Figure 3.

4.3 Region of Absolute Stability for High Orders

An important extension of Section 4.1 is to study the size of the region of absolute stability as $p \to \infty$. We begin by defining Schur polynomials.

Definition 5 *A polynomial $\pi(r)$ of degree k is said to be Schur if its roots r_t satisfy $|r_t| < 1$, $t = 1, 2, \ldots, k$.*

Table 6: Intervals of absolute stability for HBT(p)3 up to order 60.

p	$x_{\min}$	$x_{\min}/p$	p	$x_{\min}$	$x_{\min}/p$
			31	−12.924	−0.417
			32	−13.296	−0.416
3	−2.512	−0.837	33	−13.668	−0.414
4	−2.785	−0.696	34	−14.040	−0.413
5	−3.217	−0.643	35	−14.412	−0.412
6	−3.553	−0.592	36	−14.784	−0.411
7	−3.954	−0.565	37	−15.155	−0.410
8	−4.313	−0.539	38	−15.527	−0.409
9	−4.700	−0.522	39	−15.898	−0.408
10	−5.069	−0.507	40	−16.270	−0.407
11	−5.450	−0.495	41	−16.641	−0.406
12	−5.822	−0.485	42	−17.013	−0.405
13	−6.200	−0.477	43	−17.384	−0.404
14	−6.574	−0.470	44	−17.755	−0.404
15	−6.950	−0.463	45	−18.126	−0.403
16	−7.324	−0.458	46	−18.498	−0.402
17	−7.699	−0.453	47	−18.869	−0.401
18	−8.073	−0.449	48	−19.240	−0.401
19	−8.447	−0.445	49	−19.611	−0.400
20	−8.821	−0.441	50	−19.981	−0.400
21	−9.195	−0.438	51	−20.352	−0.399
22	−9.568	−0.435	52	−20.723	−0.399
23	−9.942	−0.432	53	−21.094	−0.398
24	−10.315	−0.430	54	−21.464	−0.397
25	−10.688	−0.428	55	−21.835	−0.397
26	−11.061	−0.425	56	−22.206	−0.397
27	−11.434	−0.423	57	−22.576	−0.396
28	−11.806	−0.422	58	−22.947	−0.396
29	−12.179	−0.420	59	−23.317	−0.395
30	−12.551	−0.418	60	−23.688	−0.395

The Routh–Hurwitz criterion [21] produces a set of inequalities that must be satisfied by the coefficients of a Schur polynomial. This criterion is of particular interest as it is a criterion that the roots of a polynomial lie in the left half-plane. First, we will make the partial linear transformation $\mathbb{C} \to \mathbb{C}$, $r \mapsto z$:

$$z \mapsto \frac{r-1}{r+1}. \tag{4.3.1}$$

This transformation will map the boundary of the circle $|r| = 1$ onto the imaginary axis, and the interior of the disk $|r| \leq 1$ onto the left half-plane.

Applying transformation (4.3.1) to (4.2.16), we obtain

$$(1-z)\,\pi\left(\frac{1+z}{1-z}, \hat{h}\right) = 2z - (1-z)\sum_{j=1}^{p} s_j \hat{h}^j. \tag{4.3.2}$$

Furthermore, we can say that

$$(1-z)\,\pi\left(\frac{1+z}{1-z}, \hat{h}\right) = a_0 z + a_1, \tag{4.3.3}$$

where the coefficients a_0 and a_1 must satisfy the Routh–Hurwitz criterion, so the following inequalities must be satisfied for any HBT(p)3 method of arbitrary order p:

$$k = 1; \quad a_i > 0, \quad i = 0, 1. \tag{4.3.4}$$

From (4.3.2) and (4.3.3) we obtain the coefficients that must satisfy the conditions in (4.3.4):

$$a_0 > 0 \quad \Rightarrow \quad 2 + \sum_{j=1}^{p} s_j \hat{h}^j > 0, \tag{4.3.5}$$

$$a_1 > 0 \quad \Rightarrow \quad -\sum_{j=1}^{p} s_j \hat{h}^j > 0. \tag{4.3.6}$$

From conditions (4.3.5) and (4.3.6) it is evident that the inequality to be satisfied is

$$\left| 1 + \sum_{j=1}^{p} s_j \hat{h}^j \right| < 1, \tag{4.3.7}$$

or equivalently, $|r_s| < 1$.

We are interested in the behaviour of the root r_s of polynomial (4.2.16) as the order of the method p grows. The following remark will further simplify the analysis.

Lemma 2 *For HBT(p)3 method, the coefficients $s_j = 1/j!$, for $j = 1, 2, \ldots, p$, with $p \geq 5$.*

Proof: Recall the formulation of the s_j coefficients (4.2.15) and the off-step points for HBT(p)3 as defined in (3.2.1). Using this information let us show that Lemma 2 holds. For simplicity, we will consider each s_j separately.

- $j = 1$: We want to show that $s_1 = 1$ for $j = 1$. From the solution of the Vandermonde-type system for the integration formula IF (3.4.3), we know that $b_1 = 1 - (b_3 + b_2)$. Then,

$$b_1 + b_2 + b_3 = 1 - (b_3 + b_2) + b_2 + b_3 = 1,$$

  as it was to be shown.

- $j = 2$: We want to show that $s_2 = 1/2!$ for $j = 2$. From the solutions of the Vandermonde-type systems (3.4.3) and (3.4.4), we know that

$$a_{21} = c_2, \quad a_{31} = 1 - a_{32}, \quad \gamma_{12} = \frac{1}{2!} - b_3 - c_2 b_2.$$

  Substituting the above results into the formula for s_2 reduces it to $1/2!$ as it was to be shown.

- $j = 3$: We want to show that $s_2 = 1/3!$ for $j = 3$. From the solutions of the Vandermonde-type systems (3.4.3), (3.4.4) and (3.4.7), we know that

$$a_{21} = c_2, \quad \gamma_{13} = \frac{1}{3!} - \frac{b_3}{2} - \frac{c_2^2}{2} b_2, \quad \gamma_{22} = \frac{c_2^2}{2}, \quad \gamma_{32} = \frac{1}{2} - c_2 a_{32}.$$

  Substituting the above results into the formula for s_3 reduces it to $1/3!$ as it was to be shown.

- $j = p$: Finally, when $j = p$, we will have

$$s_p = \sum_{i=2}^{3} b_i \left(\sum_{j=2}^{i-1} a_{ij} \gamma_{j,p-2} + \gamma_{i,p-1} \right) + \gamma_{1p},$$

where $\gamma_{i,r} = 0$ for $i = 1, 2, 3$ and $r \geq p - 1$. Thus, the equation for s_p can be simplified to:

$$s_p = \sum_{i=2}^{3} b_i \left(\sum_{j=2}^{i-1} a_{ij} \gamma_{j,p-2} \right).$$

Once the summations are expanded, we obtain

$$s_p = b_3 a_{32} \gamma_{2,p-2}.$$

From the solution of the Vandermonde-type systems (3.4.3), (3.4.4) and (3.4.7), we know that

$$a_{32} = \frac{(p-2)!}{b_3 \, c_2^{p-2} \, p!},$$

which will reduce to the equation $s_p = 1/p!$, as required. □

Since we have shown that Lemma 2 holds for all values of $p \geq 5$, we note that equation (4.3.7) yields

$$\left| \sum_{j=0}^{p} \frac{\hat{h}^j}{j!} \right| < 1,$$

and, as $p \to \infty$, we obtain

$$\left| \exp(\hat{h}) \right| < 1. \tag{4.3.8}$$

Since all eigenvalues are assumed distinct and $\Re \lambda h < 0$, by the Routh–Hurwitz criterion it can be said that, as $p \to \infty$, the region of absolute stability will be the left half-plane.

4.4 Principal Local Error Term

The principal local error term (PLET) of HBT(p)3, similarly to that of Runge–Kutta method of order 3, is defined in terms of elementary differentials as:

$$[\delta_1 \{\{\mathbf{f}^{p-2}\}\mathbf{f}\} + \delta_2 \{{}_3\mathbf{f}^{p-2}\}_3 + \delta_3 \{\mathbf{f}^p\} + \delta_4 \{{}_2\mathbf{f}^{p-1}\}_2] \, h^{p+1}. \tag{4.4.1}$$

Due to the choice of the off-step point, $c_2 = (p-1)/(p+1)$, the coefficients δ_3 and δ_4 vanish, and the PLET of HBT(p)3 is defined as:

$$[\delta_1 \{\{\mathbf{f}^{p-2}\}\mathbf{f}\} + \delta_2 \{{}_3\mathbf{f}^{p-2}\}_3] \, h^{p+1}.$$

To solve for the principal local truncation error coefficients (PLTC), δ_1 and δ_2, we consider the difference of the true solution, as seen in Section 3.3, and the numerical solution, both in terms of elementary differentials. The numerical solution can be expressed as in [6]:

$$\hat{y} = y_0 + \sum_{i=1}^{\infty} \frac{h^{r(i)}}{(r(i)-1)!}\,\beta(i)\,\phi(i)\,\mathbf{F}(i). \tag{4.4.2}$$

The difference between the solutions expressed in (3.3.1) and (4.4.2) is

$$y - \hat{y} = \sum_{i=1}^{\infty} \frac{\alpha(i)}{r(i)!}\,[1 - \gamma(i)\,\phi(i)]\,h^{r(i)}\,\mathbf{F}(i). \tag{4.4.3}$$

To simplify the computational tasks, we will consider a pair of remarks that depend on a theorem from [6].

Theorem 4 *If $(h^r/r!)\alpha$ is the coefficient of $\boldsymbol{F} = \{\boldsymbol{F}_1^{\mu_1}\,\boldsymbol{F}_2^{\mu_2}\cdots\boldsymbol{F}_\sigma^{\mu_\sigma}\}$ in the expansion (3.3.1) and $\boldsymbol{F}_1^{\mu_1}, \boldsymbol{F}_2^{\mu_2}, \ldots, \boldsymbol{F}_\sigma^{\mu_\sigma}$ are all distinct then*

$$\alpha = (r-1)! \prod_{i=1}^{\sigma} \frac{1}{\mu_i!}\left(\frac{\alpha(i)}{r(i)!}\right)^{\mu_i}, \tag{4.4.4}$$

where $(h^{r(i)}/r(i)!)\,\alpha(i)$, $i = 1, 2, \ldots, \sigma$, is the coefficient of $\boldsymbol{F}(i)$ in (3.3.1).

Remark 2 *The value of α associated with the elementary differential $\{\{\boldsymbol{f}^{p-2}\}\boldsymbol{f}\}$ is equal to the order p of HBT(p)3.*

To show that Remark 2 holds, we consider the elementary differential for any p, $\{\{\mathbf{f}^{p-2}\}\mathbf{f}\}$. We note that the order of this elementary differential is $r(\{\{\mathbf{f}^{p-2}\}\mathbf{f}\}) = p+1$. Using Theorem 4 and the method outlined in [6], we separate the elementary differential into

$$\mathbf{F}_1 = \mathbf{f}, \qquad \mathbf{F}_2 = \{\mathbf{f}^{p-2}\}. \tag{4.4.5}$$

Since there are two unique elementary differentials, $\sigma(\{\{\mathbf{f}^{p-2}\}\mathbf{f}\}) = 2$. Furthermore, $\mu(\mathbf{F}_1) = 1$, and from Table 2 we know that $r(\mathbf{F}_1) = \alpha(\mathbf{F}_1) = 1$.

For $\mathbf{F}_2$, $\mu(\mathbf{F}_2) = 1$, the order is $r(\mathbf{F}_2) = p-1$ and we need to solve for $\alpha(\mathbf{F}_2)$. We approach the task of solving for $\alpha(\mathbf{F}_2)$ recursively by applying Theorem 4 $\{\mathbf{f}^{p-2}\}$ to obtain:

$$\alpha(\mathbf{F}_2) = (p-2)! \cdot \frac{1}{(p-2)!} \cdot \left(\frac{1}{1!}\right)^{p-2} = 1,$$

with $\sigma(\{\mathbf{f}^{p-2}\}) = 1$. We note that the order of the elementary differential $\{\mathbf{f}^{p-2}\}$ is $r(\{\mathbf{f}^{p-2}\}) = p-1$. Once all values are substituted into (4.4.4), we can evaluate $\alpha(\{\{\mathbf{f}^{p-2}\}\mathbf{f}\})$:

$$\alpha(\{\{\mathbf{f}^{p-2}\}\mathbf{f}\}) = p!\left[\left(\frac{1}{\mu(\mathbf{F}_1)!}\cdot\frac{\alpha(\mathbf{F}_1)}{r(\mathbf{F}_1)!}\right)\cdot\left(\frac{1}{\mu(\mathbf{F}_2)!}\cdot\frac{\alpha(\mathbf{F}_2)}{r(\mathbf{F}_2)!}\right)\right] = p.$$

to obtain $\alpha = p$. □

Remark 3 *The value of α associated with the elementary differential $\{{}_3\mathbf{f}^{p-2}\}_3$ is equal to 1 for arbitrary order p of HBT(p)3.*

The order of the elementary differential in Remark 3 is $r(\{{}_3\mathbf{f}^{p-2}\}_3) = p$, and we make use of Theorem 4 and the method from [6] to separate the elementary differential into

$$\mathbf{F}_1 = \{{}_2\mathbf{f}^{p-2}\}_2,$$

with $\sigma(\{{}_3\mathbf{f}^{p-2}\}_3) = 1$, $\mu(\mathbf{F}_1) = 1$, $r(\mathbf{F}_1) = p-1$, and $\alpha(\mathbf{F}_1)$ unknown. Recursively applying the same method as above, we define

$$\mathbf{F}_2 = \{\mathbf{f}^{p-2}\},$$

and by the discussion that follows Remark 2, we know that $\alpha(\{\mathbf{f}^{p-2}\}) = 1$. Once all values are substituted in equation (4.4.4), we obtain $\alpha = 1$. □

δ_1 coefficient

Further simplifications can be made to the term $1-\gamma(i)\,\phi(i)$ in (4.4.3) by realizing that it depends on the order condition associated with the elementary differential $\{\{\mathbf{f}^{p-2}\}\mathbf{f}\}$ in PLET. After some algebraic manipulation of general form of this order condition, we obtain that

$$1-\gamma(i)\,\phi(i) = 1-\frac{(p+1)!}{p\,(p-2)!}\sum_{ij} b_i c_i a_{ij} c_j^{p-2}. \tag{4.4.6}$$

Since in HBT(p)3 an explicit 3-stage Runge–Kutta method of order 3 is used, we can further simplify (4.4.6) by removing coefficients that are equal to zero in the summation. Thus, we obtain

$$1-\gamma(i)\,\phi(i) = 1-\frac{(p+1)!}{p\,(p-2)!}\left(b_3 c_3 a_{32} c_2^{p-2}\right). \tag{4.4.7}$$

Substituting the off-step point $c_3 = 1$ and a_{32} from the solution of the Vandermonde-type systems for P_3 in (3.4.7), we reduce (4.4.7) to

$$1 - \gamma(i)\,\phi(i) = \frac{1}{p}. \tag{4.4.8}$$

The error of an HBT(p)3 method is $O(h^{p+1})$, so $r(i) = p + 1$. By Remark 2, we know that $\alpha(i) = p$ for the elementary differential $\{\{\mathbf{f}^{p-2}\}\mathbf{f}\}$, so by (4.4.3) we solve for δ_1:

$$\delta_1 = -\frac{1}{(p+1)!}. \tag{4.4.9}$$

δ_2 coefficient

Similarly to above, further simplifications can be made to the term $1 - \gamma(i)\,\phi(i)$ in (4.4.3) by realizing that it depends on the order condition associated with the elementary differential $\{_3\mathbf{f}^{p-2}\}_3$ in PLET. After some algebraic manipulation of general form of this order condition, we obtain that

$$1 - \gamma(i)\,\phi(i) = 1 - \frac{(p+1)!}{(p-2)!}\sum_{ijk} b_i a_{ij} a_{jk} c_k^{p-2}. \tag{4.4.10}$$

Since in HBT(p)3, an explicit 3-stage Runge–Kutta method of order 3 is used, we can show that the summation in (4.4.10) equals to zero. Thus, we obtain

$$1 - \gamma(i)\,\phi(i) = 1. \tag{4.4.11}$$

The error of HBT(p)3 is $O(h^{p+1})$, so $r(i) = p + 1$. By Remark 3, we know that $\alpha(i) = 1$ for the elementary differential $\{_3\mathbf{f}^{p-2}\}_3$, so by (4.4.3) we solve for δ_2:

$$\delta_2 = \frac{1}{(p+1)!}. \tag{4.4.12}$$

Since the PLTC are found for an arbitrary HBT(p)3 method of order p, we wish to compare their scaled norm with a popular Predictor-Corrector (PC) method, the Adams Method. This method was discussed in detail in Section 2.2.4.

Let the method be denoted by ABM($p, p-1$), with Adams–Bashforth predictor of order $p-1$ and Adams–Moulton corrector of order p, in Predict-Evaluate-Correct-Evaluate (PECE) mode. The PLTC of ABM($p, p-1$) are $[\beta_k C_p^*, C_{p+1}]$ by [21, p. 107].

In Section 2.2.4, a general form is provided for the error coefficient of the Adams–Bashforth predictor of order $p-1$:

$$C_p^* = (-1)^{p-1} \int_0^1 \binom{-r}{p-1} dr, \quad p = 1, 2, \ldots. \tag{4.4.13}$$

In the same section, a general form is provided for the error coefficient of the Adams–Moulton corrector of order p:

$$C_{p+1} = (-1)^p \int_{-1}^0 \binom{-r}{p} dr, \quad p = 0, 1, 2, \ldots. \tag{4.4.14}$$

It is important to note that the definition of C_p^* and C_{p+1} in (4.4.13) and (4.4.14), respectively, is strictly for the case where the order of the corrector is greater than the order of the predictor by exactly 1.

The coefficient β_k comes from the Adams–Moulton corrector, as seen in Section 2.2.4. We note that each of the backward differences will contribute a weight to the coefficient β_k, so we can conclude that:

$$\beta_k = \sum_{i=0}^{p-1} (-1)^i \int_{-1}^0 \binom{-r}{i} dr, \quad p = 1, 2, \ldots. \tag{4.4.15}$$

At this point we have all the necessary information for comparing the PLTC norms of HBT(p)3 and ABM($p, p-1$), and we summarize these results in Table 7. The scaling of the norms comes from the number of evaluations per step for the methods. From Table 7 we observe that the scaled PLTC norm of HBT(p)3 is much smaller than the scaled PLTC norm of ABM($p, p-1$).

4.5 Variable Step Size Formulation

The step size h is accepted if the integration step to x_{n+1} produces error which is at most equal to the desired tolerance `tol`, otherwise it is rejected and a new step size must be chosen. Since HBT(p)3 is by construction very similar to Taylor method, the procedure for choosing the step size for HBT(p)3 will be similar as well.

For desired tolerance `tol`, the step size of Taylor method of order p is chosen in the following manner. First, we assume that the root criterion for convergence is

Table 7: PLTC of HBT(p)3 and the scaled norms for HBT(p)3 and ABM($p, p-1$).

	PLTC of HBT(p)3		$p \times \|\text{PLTC}\|_2$	$2 \times \|\text{PLTC}\|_2$
p	δ_1	δ_2	HBT(p)3	ABM($p, p-1$)
4	$-\frac{1}{120}$	$\frac{1}{120}$	$4.7140E-02$	$2.8616E-01$
5	$-\frac{1}{720}$	$\frac{1}{720}$	$9.8209E-02$	$2.4594E-01$
6	$-\frac{1}{5040}$	$\frac{1}{5040}$	$1.6836E-03$	$2.1948E-01$
7	$-\frac{1}{40320}$	$\frac{1}{40320}$	$2.4552E-04$	$2.0049E-01$
8	$-\frac{1}{362880}$	$\frac{1}{362880}$	$3.1178E-05$	$1.8605E-01$
9	$-\frac{1}{3628800}$	$\frac{1}{3628800}$	$3.5075E-06$	$1.7461E-01$
10	$-\frac{1}{39916800}$	$\frac{1}{39916800}$	$3.5429E-07$	$1.6527E-01$
11	$-\frac{1}{479001600}$	$\frac{1}{479001600}$	$3.2477E-08$	$1.5746E-01$
12	$-\frac{1}{6227020800}$	$\frac{1}{6227020800}$	$2.7253E-09$	$1.5081E-01$
13	$-\frac{1}{87178291200}$	$\frac{1}{87178291200}$	$2.1089E-10$	$1.4506E-01$
14	$-\frac{1}{1307674368000}$	$\frac{1}{1307674368000}$	$1.5141E-11$	
15	$-\frac{1}{20922789888000}$	$\frac{1}{20922789888000}$	$1.0139E-12$	
16	$-\frac{1}{355687428096000}$	$\frac{1}{355687428096000}$	$6.3616E-14$	
17	$-\frac{1}{6402373705728000}$	$\frac{1}{6402373705728000}$	$3.7551E-15$	
18	$-\frac{1}{121645100408832000}$	$\frac{1}{121645100408832000}$	$2.0926E-16$	
19	$-\frac{1}{2432902008176640000}$	$\frac{1}{2432902008176640000}$	$1.1044E-17$	
20	$-\frac{1}{51090942171709440000}$	$\frac{1}{51090942171709440000}$	$5.5361E-19$	
30	$-\frac{1}{31!}$	$\frac{1}{31!}$	$5.1596E-33$	
40	$-\frac{1}{41!}$	$\frac{1}{41!}$	$1.6910E-48$	
50	$-\frac{1}{51!}$	$\frac{1}{51!}$	$4.5587E-65$	
60	$-\frac{1}{61!}$	$\frac{1}{61!}$	$1.6717E-82$	

applicable to Taylor series

$$y(x_0 + h) = \sum_{i \geq 0} \mathbf{Y}_i \, h^i, \qquad \mathbf{Y}_i = \frac{1}{i!} y^{(i)}(x_0)$$

which by [5, 22] yields

$$(\|\mathbf{Y}_p\|_\infty \, h^p)^{1/p} = \|\mathbf{Y}_p\|_\infty \, h < k < 1, \quad \forall p > p_0. \tag{4.5.1}$$

Since tolerance `tol` is known, equation (4.5.1) can be solved for the coefficient k by truncating the series solution at order p, and forcing the local error E_p to be less than `tol`. Since the root criterion has been assumed, we have

$$|E_p| = k^{p+1} + k^{p+2} + k^{p+3} + \cdots = k^{p+1} \left(1 + k + k^2 + \cdots\right) = \frac{k^{p+1}}{1-k},$$

and furthermore, we impose that

$$\frac{k^{p+1}}{1-k} = \texttt{tol}. \tag{4.5.2}$$

In numerical computations, the value of the coefficient k needs to be computed only once for a given tolerance `tol` and the order of the method p. There are several ways to solve equation (4.5.2), and in this thesis solutions are obtained by means of the `fsolve()` function from MAPLE for fixed order methods, and by means of Newton's method in the case of variable order methods.

Solving $(\|\mathbf{Y}_p\|_\infty)^{1/p} = k(\texttt{tol}, p)$ yields

$$h = k(\texttt{tol}, p) \, \|\mathbf{Y}_p\|_\infty^{-1/p}. \tag{4.5.3}$$

The coefficient $\mathbf{Y}_p$ in (4.5.3) is the last non-zero coefficient in the series solution and it can be replaced by $y_n^{(p)}/p!$ to obtain

$$h = k(\texttt{tol}, p) \left\| \frac{y_n^{(p)}}{p!} \right\|_\infty^{-1/p}.$$

By [3, 5], a better estimator for the step size is obtained by taking the minimum value involving the last two coefficients in the series solution in order to avoid possible problems with odd or even functions:

$$h = \min \left\{ k(\texttt{tol}, p-1) \left\| \frac{y_n^{(p-1)}}{(p-1)!} \right\|_\infty^{-1/(p-1)}, \; k(\texttt{tol}, p) \left\| \frac{y_n^{(p)}}{p!} \right\|_\infty^{-1/p} \right\}.$$

The step size for HBT(p)3 is chosen in a similar way, but since there are no derivatives of orders $p-1$ and p, the step size is calculated as

$$h = \min\left\{ k(\texttt{tol}, p-3) \left\| \frac{y_n^{(p-3)}}{(p-3)!} \right\|_\infty^{-1/(p-3)}, k(\texttt{tol}, p-2) \left\| \frac{y_n^{(p-2)}}{(p-2)!} \right\|_\infty^{-1/(p-2)} \right\}. \tag{4.5.4}$$

It is important to note that in HBT(p)3, as well as in Taylor's method, there are no rejected steps in the integration procedure since the step size is chosen once the series solution is generated to produce the required precision level.

4.6 Variable Order Formulation

Since the optimal order for a particular precision level is not known *a priori*, it is important to consider a variable order implementation of HBT(p)3. To define a variable order HBT(p)3 method, we closely follow the formulation of a variable order Taylor method as per [3, 5]. The order increment q is chosen beforehand, and if the method was at order p_i, the possible orders it can attain at the next integration step are $p_i - q$ or $p_i + q$. Once the next order is chosen, the step size has to be estimated. In HBT(p)3, if the order has been lowered to $p_i - q$, we estimate the step size by

$$h_{\text{low.est.}} = k(\texttt{tol}, p_i - 2 - q) \cdot \left\| \frac{y_n^{(p_i-2-q)}}{(p_i-2-q)!} \right\|_\infty^{-1/(p_i-2-q)}.$$

In the above formula, $p_i - 2$ represents the number of derivatives of HBT(p)3 at order p_i. If the order is increased, the procedure for estimating the step size is slightly more involved because we have not evaluated the coefficients up to order $p_i + q$.

By [23], for a function f that is analytic on a disk of radius ρ, there exist constants A_1 and A_2 such that

$$\frac{A_1}{\rho^{p_i-2}} \leq \left\| \frac{y_n^{(p_i-2)}}{(p_i-2)!} \right\|_\infty \leq \frac{A_2}{\rho^{p_i-2}}.$$

The value of the radius ρ can be estimated by considering the last several terms of

the series expansion at previous integration step. Thus,

$$\rho \approx \rho_{\text{est.}} = \min \left\{ \begin{array}{l} \left\| \left[\dfrac{y_n^{(p_i-3)}}{(p_i-3)!} \right] \Big/ \left[\dfrac{y_n^{(p_i-2)}}{(p_i-2)!} \right] \right\|_\infty, \\ \left\| \left[\dfrac{y_n^{(p_i-4)}}{(p_i-4)!} \right] \Big/ \left[\dfrac{y_n^{(p_i-2)}}{(p_i-2)!} \right] \right\|_\infty^{1/2}, \\ \left\| \left[\dfrac{y_n^{(p_i-5)}}{(p_i-5)!} \right] \Big/ \left[\dfrac{y_n^{(p_i-3)}}{(p_i-3)!} \right] \right\|_\infty^{1/2}. \end{array} \right.$$

By using the ratio of several last terms of the series expansion, possible problems with odd or even functions are avoided. Once ρ is estimated, the step size at order $p_i + q$ is estimated by

$$h_{\text{high.est.}} = k(\texttt{tol}, p_i - 2 + q) \cdot \left[\left\| \frac{y_n^{(p_i-2)}}{(p_i-2)!} \right\|_\infty \cdot \frac{1}{\rho_{\text{est}}^q} \right]^{-1/(p_i-2+q)}.$$

By [23], we know that the computational time for a Taylor method of order p_i is $O(p_i^2)$. Due to similarities between HBT(p)3 and Taylor's method, it can be concluded that the computational time for HBT(p)3 will be at most $O(p_i^2)$. As per [5], two cases should be considered at each integration step in order to decide whether the order should be lowered or increased:

$$\left[\frac{(p_i-2)+q+1}{(p_i-2)+1} \right]^2 < \frac{h_{\text{high.est.}}}{h_i} \quad \text{or} \quad \left[\frac{(p_i-2)-q+1}{(p_i-2)+1} \right]^2 < \frac{h_{\text{low.est.}}}{h_i},$$

where h_i is the step size at the last step in the integration procedure. The procedure for a variable order implementation if the order is to be changed after every M steps is presented in Algorithm 1.

The constants fac1 and fac2 are the control factors in the variable order algorithm [5]. Making these constants relatively small will results in increased difficulty while trying to change the order. Based on the numerical experiments performed, these factors have been set to fac1 = 1.0 and fac2 = 0.95 for all precision levels. The starting order p_0 is chosen similarly to [5, 23] and it equals

$$p_0 = -\frac{1}{2}\ln(\texttt{tol}),$$

where `tol` is the desired precision. It is important to note that the numerical results indicate that there is a close relation between the requested tolerance level and the resulting order of HBT(p)3 in VSVO implementation.

1. **if** *step i is a multiple of M* **then**
2. $\quad p_{i+1} = p_i$
3. $\quad h_{\max} = \max\{h_{i-M}, \ldots, h_{i-1}\}$
4. $\quad h_{\min} = \min\{h_{i-M}, \ldots, h_{i-1}\}$
5. $\quad$ **if** $((h_{i-M} < h_{\min})$ or $(h_{i-M} = h_{\min}$ and $p_{i-1} > p_i))$ **then**
6. $\quad\quad h_{\text{low.est.}} = k(\texttt{tol}, p_i - 2 - q) \cdot \left\| y_n^{(p_i-2-q)}/(p_i-2-q)! \right\|_\infty^{-1/(p_i-2-q)}$
7. $\quad\quad$ **if** $\left[\frac{(p_i-2)-q+1}{(p_i-2)+1}\right]^2 < \text{facl} \cdot \frac{h_{\text{low.est.}}}{h_i}$ **then**
8. $\quad\quad\quad p_{i+1} = p_i - q$ /* decrease order */
9. $\quad\quad$ **end**
10. $\quad$ **else if** $((h_{i-M} > h_{\max})$ or $(h_{i-M} = h_{\max}$ and $p_{i-1} < p_i))$ **then**
11. $\quad\quad \rho_{\text{est.}} =$
$$\min\left\{ \left\| \left[\frac{y_n^{(p_i-3)}}{(p_i-3)!}\right] \Big/ \left[\frac{y_n^{(p_i-2)}}{(p_i-2)!}\right] \right\|_\infty, \left\| \left[\frac{y_n^{(p_i-4)}}{(p_i-4)!}\right] \Big/ \left[\frac{y_n^{(p_i-2)}}{(p_i-2)!}\right] \right\|_\infty^{\frac{1}{2}}, \left\| \left[\frac{y_n^{(p_i-5)}}{(p_i-5)!}\right] \Big/ \left[\frac{y_n^{(p_i-3)}}{(p_i-3)!}\right] \right\|_\infty^{\frac{1}{2}} \right\}$$
12. $\quad\quad h_{\text{high.est.}} = k(\texttt{tol}, p_i - 2 + q) \cdot \left[\left\| \frac{y_n^{(p_i-2)}}{(p_i-2)!} \right\|_\infty \cdot \frac{1}{\rho_{\text{est}}^q} \right]^{-1/(p_i-2+q)}$
13. $\quad\quad$ **if** $\left[\frac{(p_i-2)+q+1}{(p_i-2)+1}\right]^2 < \text{fac2} \cdot \frac{h_{\text{high.est.}}}{h_i}$ **then**
14. $\quad\quad\quad p_{i+1} = p_i + q$ /* increase order */
15. $\quad\quad$ **end**
16. $\quad$ **end**
17. **else**
18. $\quad p_{i+1} = p_i$
19. **end**

Algorithm 1: Variable order implementation of HBT(p)3.

Chapter 5

Numerical Results

The numerical performance of HBT(p)3 is compared with the well-established Taylor method T(p) [3, 5, 22] and Dormand–Prince DP(8,7)13M method on problems commonly used to test the numerical solvers for ordinary differential equations. The full extent of flexibility and adaptability of HBT(p)3 and T(p) is visible here as it is possible to set the tolerance `tol` to be arbitrarily stringent. All three methods will be compared for tolerances up to 10^{-15}. For stringent tolerances, up to at most 10^{-50}, HBT(p)3 and T(p) of orders $p = 20$ and $p = 40$ will be compared using multiple precision.

It is important to note that higher order methods can achieve better results than lower order methods on tolerances that are not very stringent, but at the expense of a larger number of function evaluations for the derivatives involved. Also, it is not known *a priori* what is the best order of the method to solve a problem for a given tolerance. This issue is the main reason for seeking variable-step variable-order (VSVO) implementation of HBT(p)3 method. In this section we will see how VSVO HBT(p)3 compares to fixed order variable-step HBT(p)3 for several problems, and the close relationship between the desired tolerance and the average order of VSVO HBT(p)3 will be evident.

Numerical simulations were performed on 32-bit x86 architecture under Ubuntu Linux 8.04 using the GNU Compiler Collection (`GCC`) and `g++` as front end. To allow for arbitrary precision, the following libraries have been used:

- GNU Multiple Precision (GMP) library in C (http://gmplib.org/),
- Multiple-Precision Floating-point with correct Rounding (MPFR) library in C (http://www.mpfr.org/),
- Multiple-Precision Floating-point Reliable Library (MPFRCPP) C++ interface (http://beshe

HBT(p)3 procedures in double precision should be compiled with the following command:

```
g++ <procedure-name>.cpp.
```

HBT(p)3 procedures in multiple precision should be compiled with the following command:

```
g++ <procedure-name>.cpp -lgmp -lmpfr -lmpfrcpp.
```

On 32-bit processor architecture, the double precision is 64 bits long which provides approximately 16 significant digits. For multiple precision simulations, the precision of 256 bits is used, providing approximately 77 significant digits. In order to retain the correct number of significant digits during calculations, various parameters and constants have been evaluated to 500-bit precision, where applicable.

HBT(p)3, T(p) and DP(8, 7)13M are compared based on the number of steps and central processing unit (CPU) time in seconds used to perform the integration from t_0 to t_f. To obtain a better estimate of CPU time data, the numerical tests have been repeated 1000 times for each problem, unless otherwise specified.

The maximum global error (MGE) is calculated from the uniform norm $\|y_{n+1} - y(t_{n+1})\|_\infty$ of the difference between the numerical solution y_{n+1} and the analytic solution $y(t_{n+1})$ at every integration step. MGE has been calculated for all methods considered, then graphically displayed in relationship to CPU time on the horizontal axis with $\log_{10}(|\text{MGE}|)$ on the vertical axis. Each point on the graph corresponds to a different tolerance level used in calculations. The lines connecting the points are provided to enhance the visualization of method performance.

The CPU percentage efficiency gain (CPU PEG) is defined by the formula [35]

$$(\text{CPU PEG})_i = 100 \left[\frac{\sum_j \text{CPU}_{2,ij}}{\sum_j \text{CPU}_{1,ij}} - 1 \right]$$

where $\text{CPU}_{1,ij}$ and $\text{CPU}_{2,ij}$ are the CPU times of methods 1 and 2, respectively, associated with problem i, and $j = -\log_{10}(|\text{MGE}|)$. The CPU time is obtained from the curves which fit, in a least-squares sense, the data $(\log_{10}|\text{MGE}|, \log_{10}(\text{CPU}))$ by means of MATLAB's `polyfit` function.

The maximum global energy error (MGEE) is obtained from the maximum of the absolute value of the relative error $\mathcal{H}/\mathcal{H}_0 - 1$ at every integration step [26]. $\mathcal{H}$ is the value of the Hamiltonian at t_{n+1} and $\mathcal{H}_0$ is the value of the Hamiltonian at t_0. MGEE is applicable to Orbit Equations in Section 5.3 and Dynamical Systems in Section 5.4. Furthermore, MGEE has been graphically displayed as a function of number of steps (NS) used to obtain the solution by the method time on the horizontal axis with $\log_{10}(|\text{MGEE}|)$ on the vertical axis. Similarly to MGE calculations, each point on the graph corresponds to a different tolerance level used in calculations and the lines connecting the points are provided to enhance the visualization of method performance.

The number of step percentage gain (NS PEG) is defined by the formula [26]

$$(\text{NS PEG})_i = 100\left[\frac{\sum_j \text{NS}_{2,ij}}{\sum_j \text{NS}_{1,ij}} - 1\right]$$

where $\text{NS}_{1,ij}$ and $\text{NS}_{2,ij}$ are the number of steps of methods 1 and 2, respectively, associated with problem i, and $j = -\log_{10}(|\text{MGEE}|)$.

For several problems, we also show the evolution of MGEE with respect to each time point within the interval of integration. This idea follows from [37] where it is shown that the global error of an ODE solver is expected to be proportional to the tolerance `tol`. We will compare the evolution of the MGEE magnitude in case of HBT(p)3 and T(p), at various orders and precision levels.

Higher derivatives for HBT(p)3 and T(p) are calculated at each integration step using the recurrence formulae from Section 2.2.5. This implementation significantly reduces the time necessary to solve a problem by applying recurrence, and thus updating values for the higher derivatives from previous ones as opposed to calculating them individually.

5.1 Single Equations

As a first single-equation problem, we take the problem of DETEST [20] A1:

$$\text{A1:} \quad y' = -y, \quad y(0) = 1.$$

The analytic solution is

$$y(x) = \exp(-x).$$

The interval of integration for was set between $t_0 = 0$ and $t_f = 10$, and to obtain a better estimate of CPU time, the test has been repeated 3000 times. In Figure 4a, CPU time (horizontal axis) is plotted versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for HBT(12)3 and T(12), and DP(8, 7)13M using double precision. Furthermore, we have:

- CPU PEG of HBT(12)3 over T(12) is 87%;
- CPU PEG of HBT(12)3 over DP(8, 7)13M is 201%.

CPU PEG for problem A1 shows that HBT(12)3 performs better than T(12) and DP(8, 7)13M. In Figure 4b, CPU time (horizontal axis) is plotted versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for HBT(40)3 and T(40) using multiple precision. We compute CPU PEG of HBT(40)3 over T(40) to be 124%. It should be noted that, as the order increases from $p = 12$ to $p = 40$, HBT(p)3 retains its superiority over T(p).

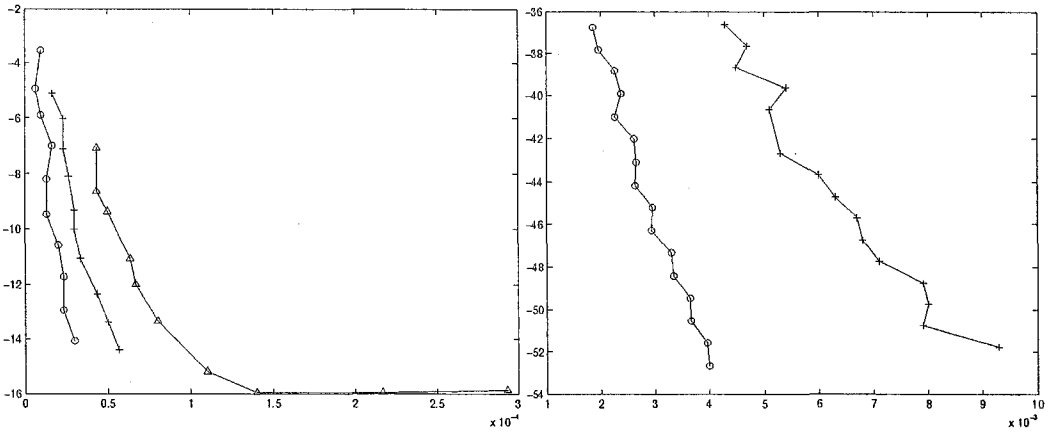
For this problem we also compare the performance of HBT(40)3 and T(40) using multiple precision when calculating the relative error. We define the relative error as the absolute value of the quotient

$$\left| \frac{\text{actual solution} - \text{calculated solution}}{\text{actual solution}} \right|,$$

and summarize the results in Table 8. It is evident that HBT(p)3 compares very favourably to T(p), and is capable of obtaining precise solutions at stringent tolerances, with improvements in the number of steps and CPU time.

5.2 Small Systems

In this section we compare the performance of the methods on two problems: the growth of two conflicting populations (B1) and the Euler equations of motion for a



(a) HBT(12)3 ∘, T(12) × and DP(8,7)13M ▷ (b) HBT(40)3 ∘, T(40) ×

Figure 4: CPU time (horizontal axis) versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for problem A1 using (a) double precision and (b) multiple precision.

Table 8: Number of steps, CPU time and relative error of HBT(40)3 and T(40) using multiple precision for problem A1.

A1	HBT(40)3			T(40)		
Tol	NS	CPU	RelErr	NS	CPU	RelErr
10^{-35}	4	$1.86E-03$	$1.0453E-33$	6	$4.76E-03$	$1.7445E-33$
10^{-40}	6	$2.62E-03$	$1.8991E-38$	7	$5.14E-03$	$8.7887E-39$
10^{-45}	8	$3.30E-03$	$8.5195E-44$	9	$6.92E-03$	$1.3271E-43$
10^{-50}	10	$5.10E-03$	$2.2286E-49$	12	$9.05E-03$	$2.6448E-48$

rigid body without external forces (B5) from DETEST [20]. We define the problems as

$$\begin{aligned} \text{B1:} \quad & y_1' = 2(y_1 - y_1 y_2), & y_1(0) &= 1 \\ & y_2' = -(y_2 - y_1 y_2), & y_2(0) &= 3 \end{aligned}$$

$$\begin{aligned} \text{B5:} \quad & y_1' = y_2 y_3, & y_1(0) &= 0 \\ & y_2' = -y_1 y_3, & y_2(0) &= 1 \\ & y_3' = -0.51 y_1 y_2, & y_3(0) &= 1. \end{aligned}$$

The interval of integration for problem B1 was set between $t_0 = 0$ and $t_f = 20$. In Figure 5, CPU time (horizontal axis) is plotted versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for HBT(12)3 and T(12), and DP(8, 7)13M. Furthermore, we have:

- CPU PEG of HBT(12)3 over T(12) is 37%;
- CPU PEG of HBT(12)3 over DP(8, 7)13M is 80%.

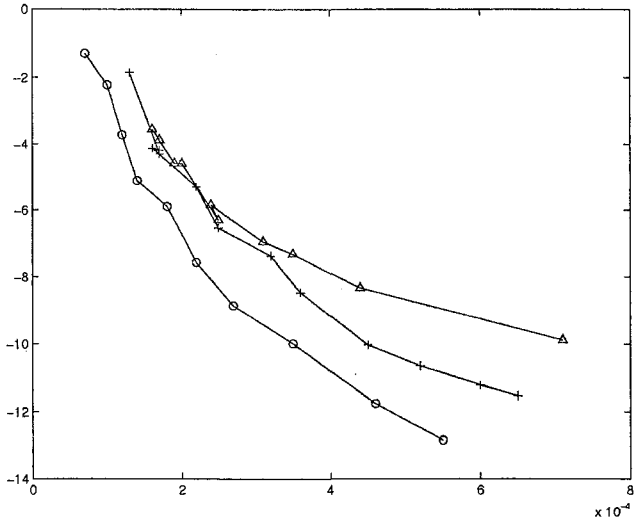
CPU PEG for problem B1 shows that HBT(12)3 performs slightly better than T(12) and DP(8, 7)13M.

The interval of integration for problem B5 was set between $t_0 = 0$ and $t_f \approx 52.15$. In Figure 6, CPU time (horizontal axis) is plotted versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for HBT(12)3, T(12), and DP(8, 7)13M. Furthermore, we have:

- CPU PEG of HBT(12)3 over T(12) is 199%;
- CPU PEG of HBT(12)3 over DP(8, 7)13M is 191%.

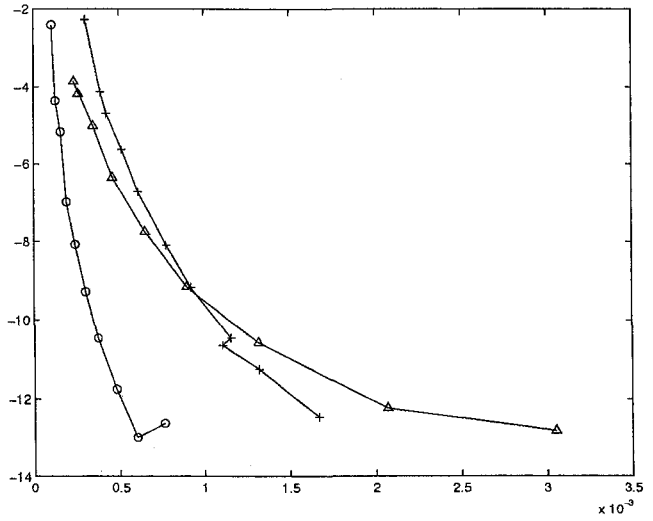
CPU PEG for problem B5 shows that HBT(12)3 performs better than T(12) and DP(8, 7)13M.

As the performance of HBT(p)3 is compared to T(p) and DP(8, 7)13M for problems B1 and B5, we note that HBT(p)3 performed better on a more complicated problem. As it will be shown in the sections to come, this is the true power of HBT(p)3: its performance on complicated problems with a large number of computations is very respectable.



HBT(12)3 $\circ$, T(12) $\times$ and DP(8, 7)13M $\triangleright$

Figure 5: CPU time (horizontal axis) versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for problem B1.



HBT(12)3 $\circ$, T(12) $\times$ and DP(8, 7)13M $\triangleright$

Figure 6: CPU time (horizontal axis) versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for problem B5.

5.3 Orbit Equations

The performance of the methods on two-body Kepler's problem with various eccentricity ε is shown here. The first-order differential equations that describe this problem are:

$$\begin{aligned} y_1' &= y_3, & y_1(0) &= 1-\varepsilon \\ y_2' &= y_4, & y_2(0) &= 0 \\ y_3' &= -y_1/(y_1^2+y_2^2)^{3/2}, & y_3(0) &= 0 \\ y_3' &= -y_2/(y_1^2+y_2^2)^{3/2}, & y_4(0) &= \sqrt{(1+\varepsilon)/(1-\varepsilon)}. \end{aligned}$$

The above equations are derived from the orbit equations

$$\begin{aligned} x'' &= -x/r^3, \quad x(0) = 1-\varepsilon, \quad x'(0) = 0 \\ y'' &= -y/r^3, \quad x(0) = 0, \qquad y'(0) = \sqrt{(1+\varepsilon)/(1-\varepsilon)}, \end{aligned}$$

with solutions

$$x = \cos u - \varepsilon, \quad x' = -\frac{\sin u}{1-\varepsilon\cos u},$$

$$y = \sin u\sqrt{1-\varepsilon^2}, \quad y' = -\frac{\cos u\sqrt{1-\varepsilon^2}}{1-\varepsilon\cos u},$$

where $u - \varepsilon\sin u - t = 0$ [20]. The Hamiltonian,

$$\mathcal{H}_{\text{Kepler}} = \frac{1}{2}\left(y_3^2+y_4^2\right) - \frac{1}{\sqrt{y_1^2+y_2^2}},$$

is used in the calculation of MGEE.

During numerical simulations, the interval of integration used was $t_0 = 0$ and $t_f = 16\pi$, and in Table 9 we summarize the results for double precision using various eccentricities. In Figure 7, CPU time (horizontal axis) is plotted versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for HBT(12)3, T(12), and DP(8,7)13M. Based on this data, CPU PEG results for the methods are in Table 10.

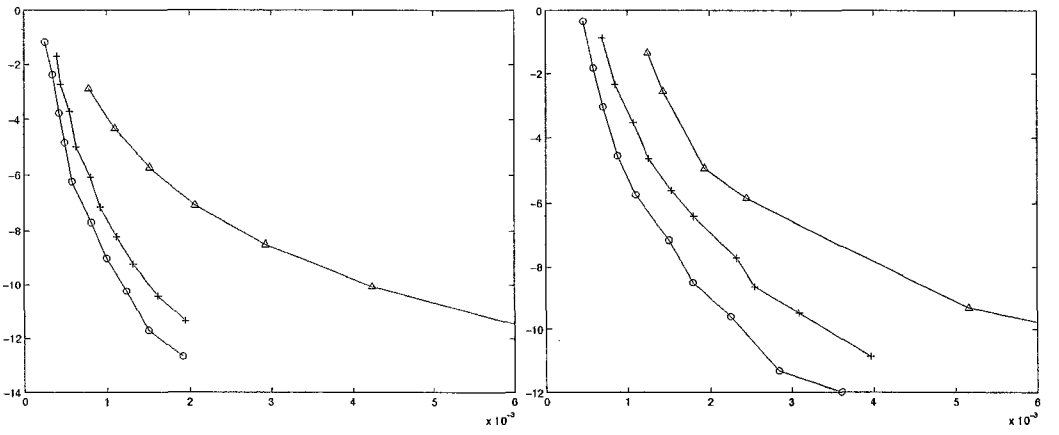
In Figure 8, number of steps (horizontal axis) is plotted versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for HBT(12)3 and T(12). Based on this data, NS PEG results for the methods are in Table 11. NS PEG results indicate that, as the problems become computationally difficult, HBT(p)3 performs favourably over T(p) of the same order.

Table 9: Number of steps, CPU time and maximum global error of HBT(12)3, T(12), and DP(8, 7)13M for Kepler's problem using double precision

Kepler	HBT(12)3			T(12)			DP(8, 7)13M		
tol	NS	CPU	MGE	NS	CPU	MGE	NS	CPU	MGE
$\varepsilon = 0.1$									
10^{-5}	41	$2.50E-03$	$6.54E-02$	54	$3.80E-04$	$1.87E-03$	86	$1.06E-03$	$1.25E-04$
10^{-10}	126	$9.90E-04$	$9.72E-10$	139	$9.20E-04$	$1.16E-08$	504	$5.96E-03$	$6.71E-12$
10^{-15}	397	$2.95E-03$	$7.02E-14$	361	$2.36E-03$	$6.94E-14$	3452	$4.06E-02$	$2.01E-13$
$\varepsilon = 0.5$									
10^{-5}	75	$5.80E-04$	$1.52E-02$	104	$7.00E-04$	$4.64E-03$	115	$1.51E-03$	$2.80E-03$
10^{-10}	235	$1.79E-03$	$3.27E-09$	268	$1.77E-03$	$3.93E-08$	690	$7.99E-03$	$2.05E-11$
10^{-15}	741	$5.54E-03$	$1.96E-13$	695	$4.56E-03$	$1.60E-12$	4693	$5.44E-02$	$4.77E-12$
$\varepsilon = 0.9$									
10^{-5}	146	$9.00E-04$	$1.00E-02$	204	$1.35E-03$	$3.49E-01$	245	$3.80E-03$	$2.59E-01$
10^{-10}	460	$3.41E-03$	$1.57E-07$	526	$3.43E-03$	$3.22E-07$	1164	$1.38E-02$	$1.63E-08$
10^{-15}	1449	$1.07E-02$	$8.67E-11$	1363	$8.96E-03$	$1.34E-10$	7982	$9.40E-02$	$8.76E-11$
$\varepsilon = 0.99$									
10^{-5}	260	$1.54E-03$	$2.44E-01$	358	$2.35E-03$	$1.40E+01$	438	$6.96E-03$	$5.41E+00$
10^{-10}	789	$5.81E-03$	$5.45E-05$	894	$5.85E-03$	$2.44E-04$	1948	$2.24E-02$	$4.70E-06$
10^{-15}	2484	$1.90E-02$	$2.60E-08$	2319	$1.56E-02$	$1.45E-07$	16513	$2.43E-01$	$6.16E-07$

Table 10: CPU PEG of HBT(12)3 over T(12) and DP(8, 7)13M for Kepler's problem with varying eccentricity.

Kepler	CPU PEG of HBT(12)3 over:	
ε	T(12)	DP(8, 7)13M
0.1	29%	306%
0.5	40%	233%
0.9	41%	358%
0.99	7%	319%

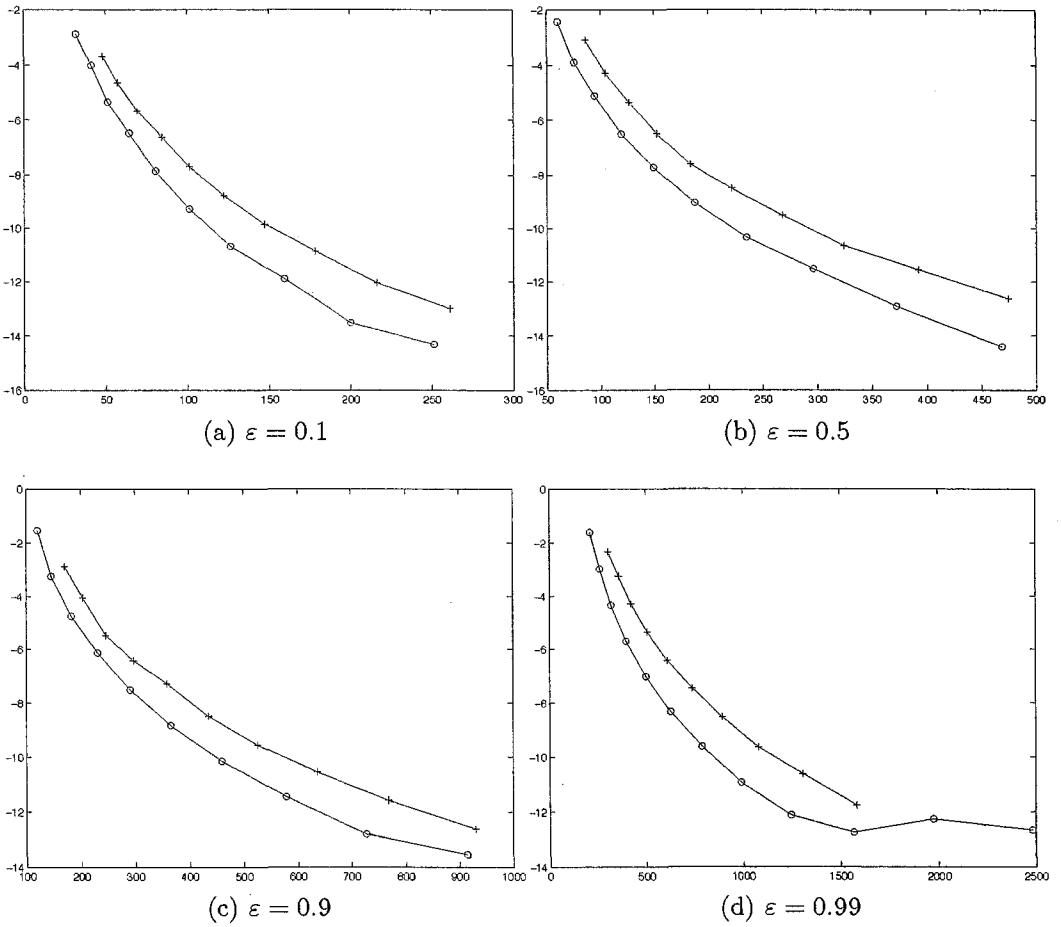


(a) $\varepsilon = 0.1$ (b) $\varepsilon = 0.5$

(c) $\varepsilon = 0.9$ (d) $\varepsilon = 0.99$

HBT(12)3 $\circ$, T(12) $\times$ and DP$(8,7)$13M $\triangleright$

Figure 7: CPU time (horizontal axis) versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for Kepler's problem with varying eccentricity.



(a) $\varepsilon = 0.1$ (b) $\varepsilon = 0.5$

(c) $\varepsilon = 0.9$ (d) $\varepsilon = 0.99$

HBT(12)3 ∘, T(12) ×

Figure 8: Number of steps (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for Kepler's problem with varying eccentricity.

Table 11: NS PEG of HBT(12)3 over T(12) for Kepler's problem with varying eccentricity.

Kepler ε	NS PEG over T(12)
0.1	32%
0.5	31%
0.9	24%
0.99	21%

In Table 12, number of steps, CPU time and MGEE are summarized for HBT(p)3 and T(p) of orders $p = 20$ and $p = 40$ at stringent tolerances using multiple precision. In this case, the eccentricity varies and approaches 1, thus making calculations more difficult. As the demand for increased precision grows, larger CPU times are to be expected and we note that HBT(p)3 method yields solutions with high precision.

In Figure 9, number of steps (horizontal axis) is plotted versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for HBT(40)3 and T(40) for various eccentricities. Based on this data, NS PEG results for the methods are in Table 13. NS PEG results confirm that, as the problems become computationally difficult, HBT(p)3 performs favourably over T(p) of the same order.

In Table 14, a comparison is made between VS fixed order and VSVO HBT(p)3 at given tolerances using multiple precision at eccentricity of $\varepsilon = 0.999$. The last row in the table shows the average order of VSVO HBT(p)3. VSVO HBT(p)3 compares favourably to VS HBT(p)3 with improvements in CPU times as well as MGEE. For HBT(12)3 it was not feasible to compute CPU times and MGEE at stringent tolerances, thus they have been omitted. The results from Table 14 are plotted in Figure 10 with CPU time on horizontal axis and $\log_{10}(|\text{MGEE}|)$ on vertical axis.

The claim that HBT(p)3 is a good choice for stringent tolerances is confirmed by considering the evolution of MGEE for different precision levels in Figure 11. The magnitude of MGEE has been noted across the integration interval from t_0 to t_f,

Table 12: Number of steps, CPU time and maximum global energy error of HBT(p)3 and T(p) of orders $p = 20$ and $p = 40$ for Kepler's problem using multiple precision

Kepler	HBT(p)3			T(p)		
Tol	NS	CPU	MGEE	NS	CPU	MGEE
$p = 20, \varepsilon = 0.9$						
10^{-20}	563	3.66	$1.5651E-19$	730	3.39	$1.6692E-19$
10^{-25}	1067	6.84	$2.5093E-25$	1404	6.28	$2.7059E-25$
10^{-30}	2021	12.91	$4.4134E-31$	2494	11.76	$2.1409E-30$
$p = 20, \varepsilon = 0.99$						
10^{-20}	1066	6.93	$8.8789E-20$	1224	5.64	$1.0691E-18$
10^{-25}	2018	13.19	$1.4209E-25$	2356	10.62	$1.2046E-24$
10^{-30}	3823	24.99	$2.2083E-31$	4185	18.83	$9.6898E-30$
$p = 20, \varepsilon = 0.999$						
10^{-20}	1525	9.82	$2.6600E-19$	1746	8.03	$3.5174E-18$
10^{-25}	2887	18.45	$3.6971E-25$	3360	15.16	$4.9394E-24$
10^{-30}	5469	35.42	$6.2052E-31$	5968	26.19	$3.2661E-29$
$p = 40, \varepsilon = 0.999999$						
10^{-20}	1024	10.15	$6.2314E-18$	1157	16.16	$4.7229E-17$
10^{-25}	1389	13.53	$2.1440E-23$	1544	21.60	$7.1428E-22$
10^{-30}	1882	18.43	$1.5390E-28$	1978	27.84	$2.8671E-26$

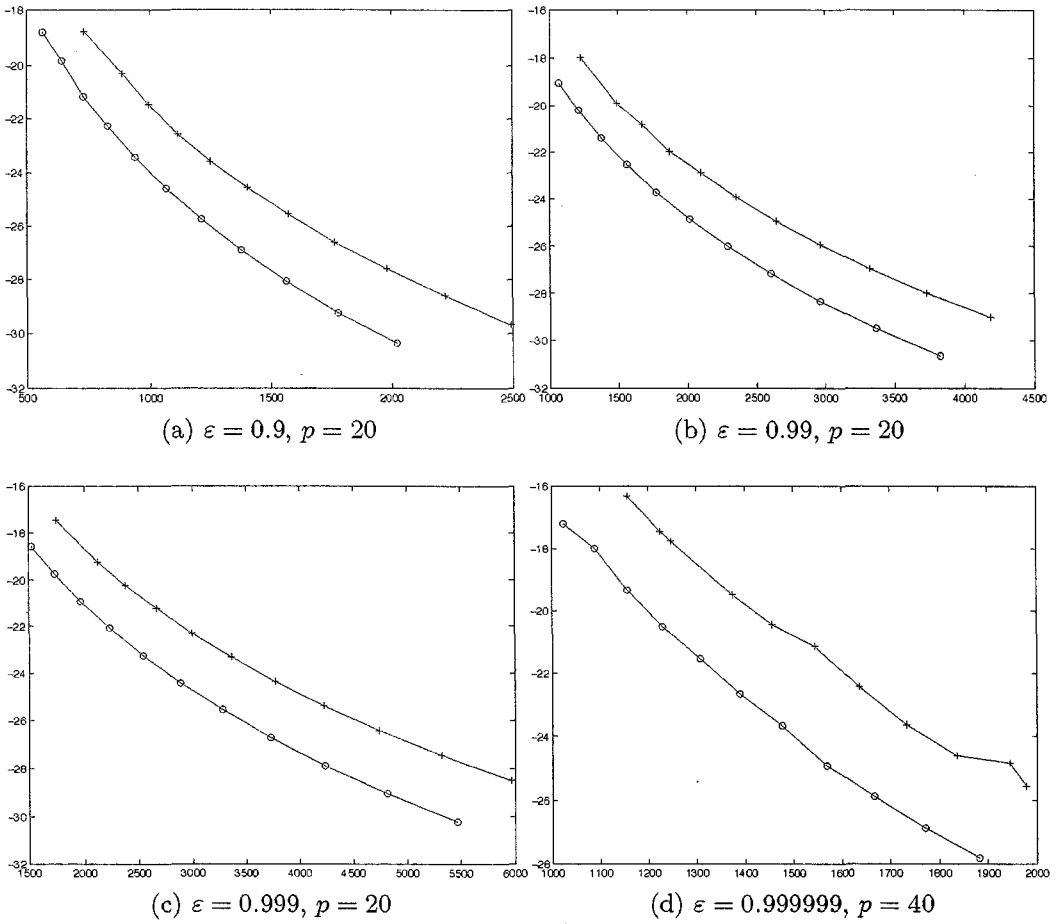


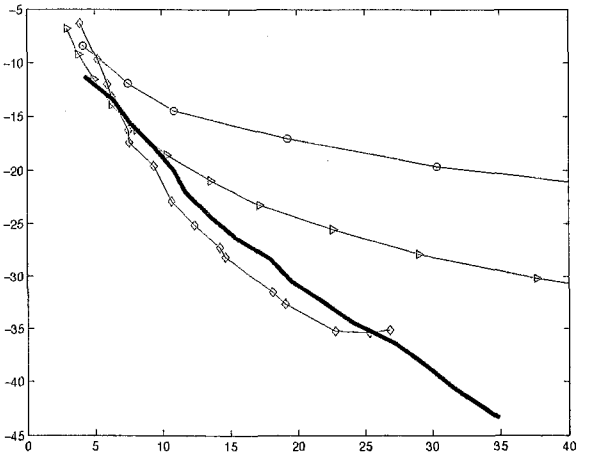
Figure 9: Number of steps (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for Kepler's problem with varying eccentricity using multiple precision.

Table 13: NS PEG of HBT(p)3 over T(p) of orders $p = 20$ and $p = 40$ for Kepler's problem with varying eccentricity using multiple precision.

Kepler ε	Order p	NS PEG of HBT(p)3 over T(p)
0.9	20	28%
0.99	20	25%
0.999	20	28%
0.999999	40	15%

Table 14: CPU time and MGEE of VS fixed order and VSVO HBT(p)3 for Kepler's problem with $\varepsilon = 0.999$ at various precision levels.

Kepler	tol=10^{-10}		tol=10^{-20}		tol=10^{-30}		tol=10^{-40}	
Order	CPU	MGEE	CPU	MGEE	CPU	MGEE	CPU	MGEE
12	4.17	$4.1210E-09$	40.91	$6.3569E-22$	n/a		n/a	
20	2.95	$1.6441E-07$	9.82	$2.6600E-19$	35.42	$6.2052E-31$	129.39	$3.0861E-42$
40	3.94	$4.9841E-07$	7.54	$4.0000E-18$	14.68	$7.1092E-29$	26.82	$7.7858E-36$
VO	4.60	$5.0344E-12$	12.10	$1.0558E-22$	21.72	$4.0785E-33$	33.82	$4.5630E-44$
Avg. Order	12		23		32		41	



HBT(12)3 ○, HBT(20)3 ▷, HBT(40)3 ◇, and VSVO HBT(p)3 –

Figure 10: CPU time (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for Kepler's problem of eccentricity $\varepsilon = 0.999$ using VS fixed order and VSVO HBT(p)3 implementation.

then plotted using $\log_{10}(|\text{MGEE}|)$ versus time values for both HBT(p)3 and T(p) at various orders and eccentricities.

- In Figure 11a, for both HBT(20)3 and T(20), MGEE gradually increases over the integration interval with noticeable changes to the magnitude at intervals of 2π. Within the integration interval, MGEE of HBT(20)3 grows within one order of the tolerance, while MGEE of T(20) grows within two orders of tolerance.

- In Figures 11b and 11c, we compare the performance of HBT(p)3 and T(p) at the same tolerance level but varying the order from $p = 20$ to $p = 40$. With $p = 20$, the methods settle on the magnitude of MGEE rather quickly, with HBT(20)3 performing better than T(20). With the same requirements and order $p = 40$, HBT(40)3 performs better than T(40) while showing slightly more variation in the magnitude of MGEE.

- In Figure 11d, the methods behave similarly as in the previous cases, and it appears that MGEE of HBT(40)3 experiences less changes near the end of the integration interval as oppose to T(40).

Overall, it can be concluded that HBT(p)3 performs better than T(p) method when stringent tolerances are imposed, and when calculations become more complex.

5.4 Higher Order Equations

We consider the performance of the methods on a problem derived from Van der Pol's equation [10, p. 358, 531], and denoted by E2

$$\text{E2}: \quad y'' - (1 - y^2)\, y' + y = 0. \tag{5.4.1}$$

The system of first-order differential equations that corresponds to (5.4.1) is

$$\begin{aligned} y_1' &= y_2, & y_1(0) &= 2, \\ y_2' &= (1 - y_1^2)\, y_2 - y_1, & y_2(0) &= 0. \end{aligned}$$

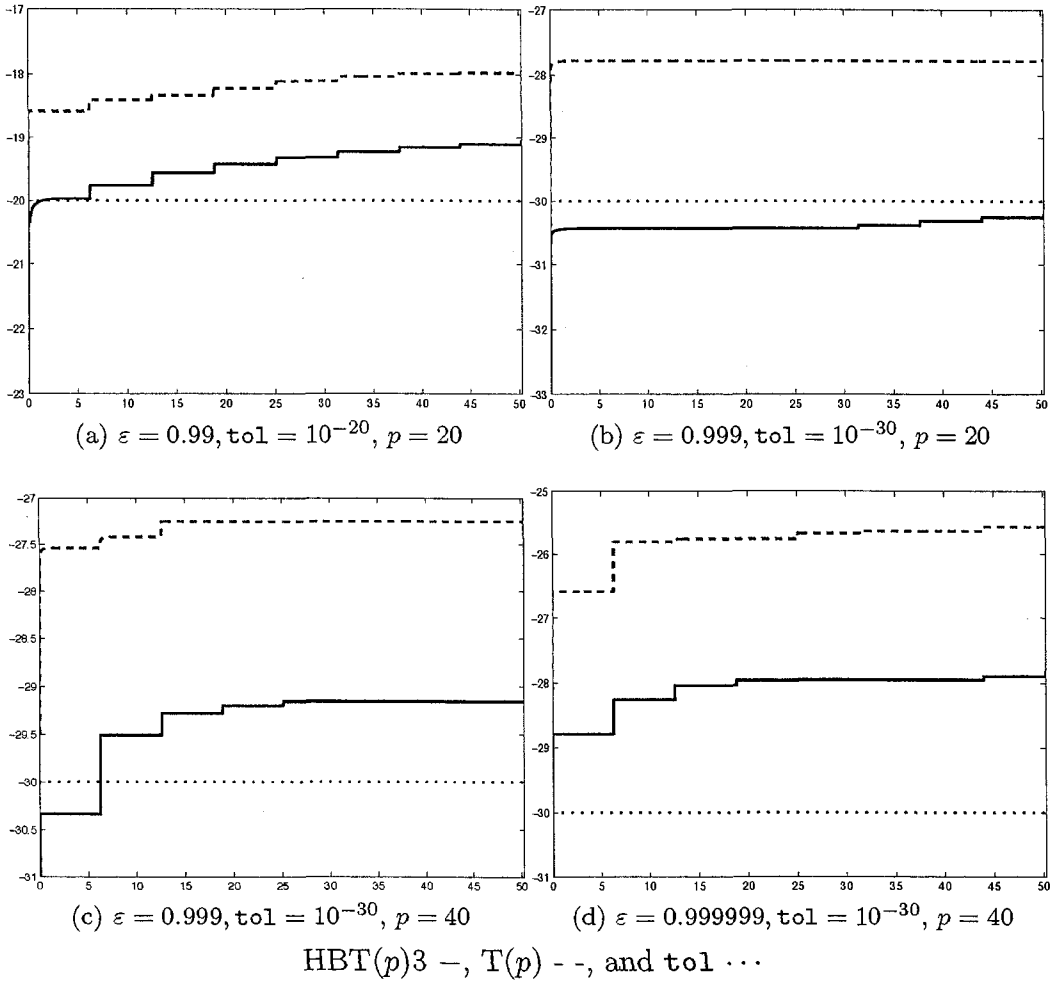
The interval of integration for problem E2 was set between $t_0 = 0$ and $t_f = 20$. In Figure 12, CPU time (horizontal axis) is plotted versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for HBT(12)3, T(12), and DP(8, 7)13M. Furthermore, we have:

- CPU PEG of HBT(12)3 over T(12) is 132%;
- CPU PEG of HBT(12)3 over DP(8, 7)13M is 117%.

CPU PEG for problem E2 shows that HBT(12)3 performs better than T(12) and DP(8, 7)13M.

5.5 Dynamical Systems

This problem section was inspired by various dynamical systems literature, and we consider the performance of the methods on Arenstorf's orbits, Hénon–Heiles problem and the equatorial main problem.

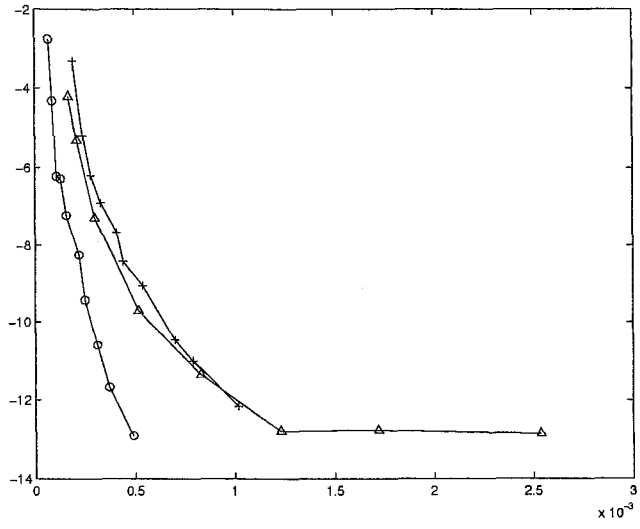


(a) $\varepsilon = 0.99, \texttt{tol} = 10^{-20}$, $p = 20$ (b) $\varepsilon = 0.999, \texttt{tol} = 10^{-30}$, $p = 20$

(c) $\varepsilon = 0.999, \texttt{tol} = 10^{-30}$, $p = 40$ (d) $\varepsilon = 0.999999, \texttt{tol} = 10^{-30}$, $p = 40$

HBT(p)3 $-$, T(p) - -, and `tol` $\cdots$

Figure 11: The evolution of MGEE at different precision levels; time (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for Kepler's problem with varying eccentricity using multiple precision.



HBT(12)3 ∘, T(12) × and DP(8,7)13M ▷

Figure 12: CPU time (horizontal axis) versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for problem E2.

5.5.1 Arenstorf's Orbits

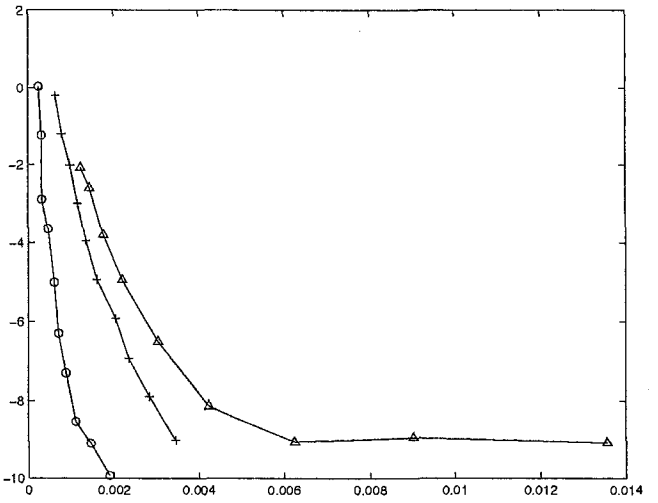
This problem is a specific case of the restricted three-body problem with small mass ratio that has one-parameter families of closed solution curves [2]. The equations describing this problem are [15, 38]

$$\begin{aligned} x'' &= x + 2y' - \mu' \frac{x+\mu}{D_1} - \mu \frac{x-\mu'}{D_2}, \\ y'' &= y - 2x' - \mu' \frac{y}{D_1} - \mu \frac{y}{D_2}, \end{aligned} \tag{5.5.1}$$

where

$$\begin{aligned} D_1 &= \left((x+\mu)^2 + y^2\right)^{3/2}, \\ D_2 &= \left((x-\mu')^2 + y^2\right)^{3/2}, \\ x(0) &= 0.994, \quad x'(0) = 0, \\ y(0) &= 0, \quad y'(0) = -2.00158510637908252240537862224, \\ \mu &= 0.012277471, \quad \mu' = 1 - \mu. \end{aligned}$$

The interval of integration is between $t_0 = 0$ and $t_f \approx 17.1$, and in Figure 13, CPU



HBT(12)3 ∘, T(12) × and DP(8, 7)13M ▷

Figure 13: CPU time (horizontal axis) versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for Arenstorf's orbits problem.

time (horizontal axis) is plotted versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for HBT(12)3, T(12), and DP(8, 7)13M. Furthermore, for this problem we find that

- CPU PEG of HBT(12)3 over T(12) is 168%;
- CPU PEG of HBT(12)3 over DP(8, 7)13M is 358%.

The above results indicate that HBT(12)3 wins over T(12) and DP(8, 7)13M by a large margin.

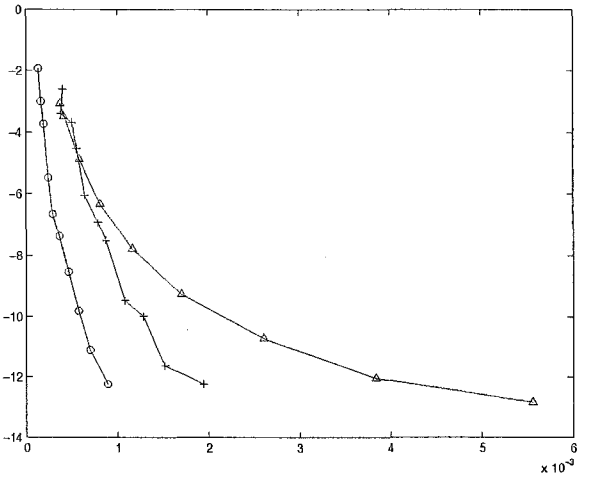
5.5.2 Hénon–Heiles Problem

This problem is described in [17] and given by the Hamiltonian

$$\mathcal{H}_{\text{Hénon-Heiles}} = \frac{1}{2}\left(X^2 + Y^2\right) + \frac{1}{2}\left(x^2 + y^2\right) + \varepsilon y\left(x^2 - \frac{1}{3}y^2\right).$$

The initial values for this problem are

$$x(0) = 0, \quad y(0) = 0.52, \quad X(0) = 0.371956090598519, \quad Y(0) = 0, \quad \varepsilon = 1.$$



HBT(12)3 $\circ$, T(12) $\times$ and DP(8, 7)13M $\triangleright$

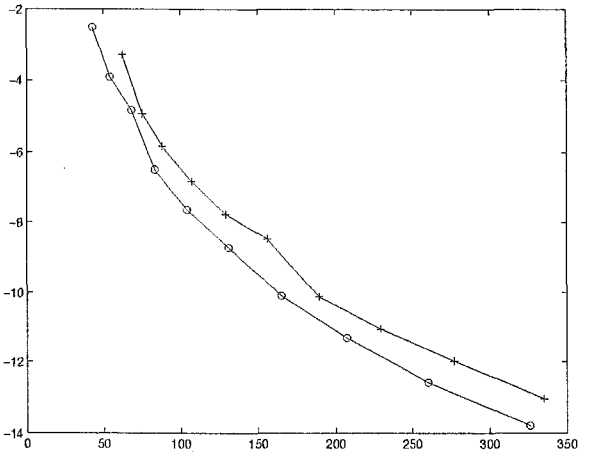
Figure 14: CPU time (horizontal axis) versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for the Hénon–Heiles problem.

During the numerical simulations, the interval of integration used was $t_0 = 0$ and $t_f = 70$, and in Figure 14, CPU time (horizontal axis) is plotted versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for HBT(12)3, T(12) and DP(8, 7)13M. Based on this data, CPU PEG results are:

- CPU PEG of HBT(12)3 over T(12) is 127%;
- CPU PEG of HBT(12)3 over DP(8, 7)13M is 240%.

In Figure 15, the number of steps (horizontal axis) is plotted versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for HBT(12)3 and T(12) of order $p = 12$. Based on this data, NS PEG of HBT(12)3 over T(12) is 18%. NS PEG result indicates that HBT(p)3 performs favourably over T(p) of the same order.

In Figure 16, the number of steps (horizontal axis) is plotted versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for HBT(20)3 and T(20). Based on this data, NS PEG of HBT(20)3 over T(20) is 17%. The NS PEG result indicates that HBT(p)3 performs favourably over T(p) of the same order at stringent tolerances. It should be noted that NS PEG of HBT(20)3 over T(20) is only slightly less than NS PEG of HBT(12)3 over T(12), thus showing the capability of HBT(p)3 to retain its excellent performance in more



HBT(12)3 ∘, T(12) ×

Figure 15: Number of steps (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for the Hénon–Heiles problem.

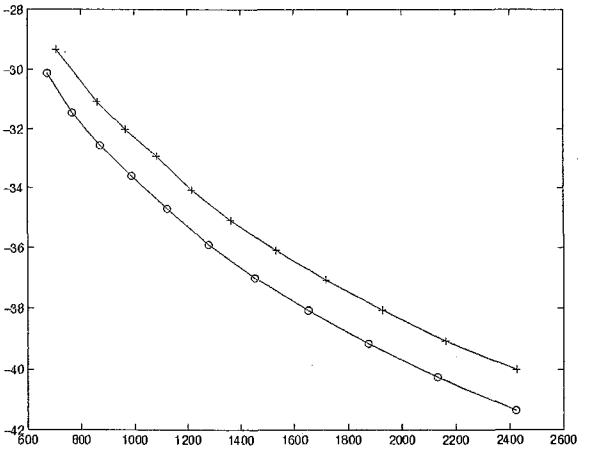
Table 15: Number of steps, CPU time and maximum global energy error of HBT(20)3 and T(20) for the Hénon-Heiles problem using multiple precision.

HH	HBT(20)3			T(20)		
tol	NS	CPU	MGEE	NS	CPU	MGEE
10^{-30}	675	2.97	$7.4039E-31$	708	2.10	$4.6323E-30$
10^{-35}	1279	6.07	$1.2975E-36$	1364	3.99	$8.1966E-36$
10^{-40}	2423	11.52	$4.3542E-42$	2425	7.30	$9.9317E-41$

demanding scenarios.

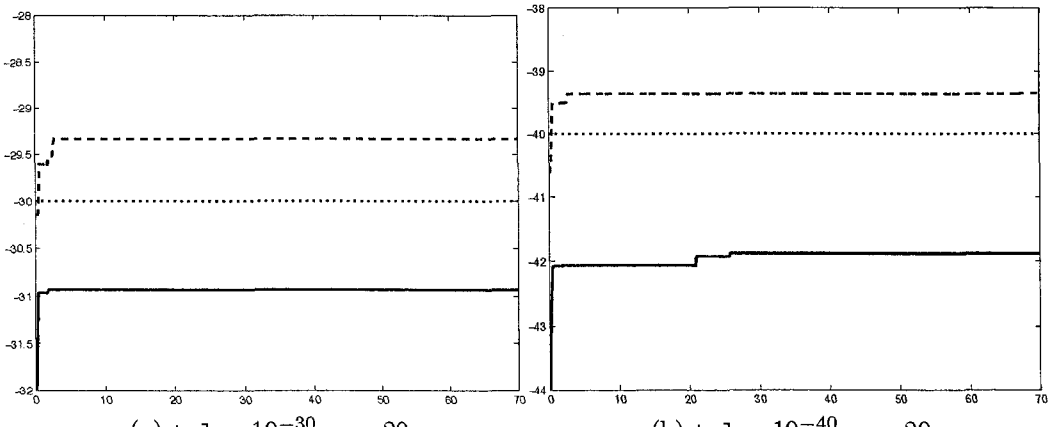
In Figure 17, the evolution of MGEE is shown at different precision levels and a comparison is made between HBT(20)3 and T(20) at tolerances 10^{-30} and 10^{-40}. It is evident that HBT(20)3 outperforms T(20) in terms of MGEE at the precision levels considered. In Table 15, the number of steps, CPU time and MGEE are summarized for HBT(20)3 and T(20) at stringent tolerances using multiple precision. We note that HBT(20)3 is slightly slower than T(20), but it compensates by exceeding the performance of T(20) in terms of MGEE.

In Table 16, a comparison is made between VS fixed order and VSVO HBT(p)3



HBT(20)3 $\circ$, T(20) $\times$

Figure 16: Number of steps (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for the Hénon–Heiles problem using multiple precision.



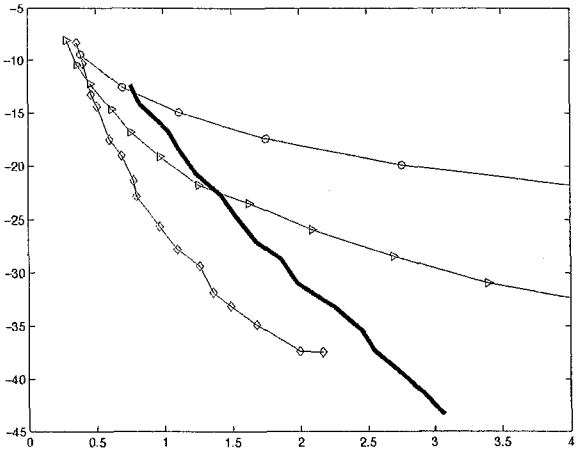
(a) tol $= 10^{-30}$, $p = 20$ (b) tol $= 10^{-40}$, $p = 20$

HBT(p)3 $-$, T(p) - -, and tol $\cdots$

Figure 17: The evolution of MGEE at different precision levels; time (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for the Hénon-Heiles problem using multiple precision.

Table 16: CPU time and MGEE of VS fixed order and VSVO HBT(p)3 for the Hénon–Heiles problem at various precision levels.

HH	tol=10^{-10}		tol=10^{-20}		tol=10^{-30}		tol=10^{-40}	
Order	CPU	MGEE	CPU	MGEE	CPU	MGEE	CPU	MGEE
12	0.41	$3.8673E-10$	4.43	$4.5892E-23$	43.82	$2.6540E-34$	n/a	
20	0.29	$8.8049E-09$	0.97	$7.2776E-20$	2.97	$7.4039E-31$	11.52	$4.3542E-42$
40	0.36	$5.0942E-09$	0.69	$9.0632E-20$	1.26	$4.4292E-30$	2.17	$4.1097E-38$
VO	0.76	$4.3139E-13$	1.35	$2.0268E-23$	2.24	$6.5352E-34$	3.07	$4.4028E-44$
Avg. Order	12		24		31		43	



HBT(12)3 ∘, HBT(20)3 ▷, HBT(40)3 ◇, and VSVO HBT(p)3 –

Figure 18: CPU time (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for the Hénon–Heiles problem using VS fixed order and VSVO HBT(p)3 implementation.

at given tolerances using multiple precision. The last row in the table shows the average order of VSVO HBT(p)3. Again, VSVO HBT(p)3 compares favourably to VS HBT(p)3 with improvements in CPU times as well as MGEE. For HBT(12)3 it was not feasible to compute CPU times and MGEE at tolerance 10^{-40}, thus it has been omitted. The results from Table 16 are plotted in Figure 18 with CPU time on horizontal axis and $\log_{10}(|\text{MGEE}|)$ on vertical axis.

5.5.3 The Equatorial Main Problem

This problem accepts the polar component Λ of the angular momentum as an integral and it describes artificial satellite theory [38]. Other parameters used to describe this problem are: the gravitational constant of the planet μ, oblateness coefficient J_2, and scaling factor α. The Hamiltonian in cylindrical coordinates for this problem is

$$\mathcal{H}_{\text{eq. main prob.}} = \frac{1}{2}\left(P^2 + \frac{\Lambda^2}{\rho^2} + Z^2\right) + \frac{\mu}{r} + \frac{\alpha^2 J_2\, \mu\, P_2(u)}{r^3},$$

where $u = z/r$, $r = \sqrt{\rho^2 + z^2}$ and $P_2(x) = (3x^2 - 1)/2$ is the Legendre polynomial of degree 2. The initial values used in numerical simulations are [5]

$$\rho(0) = 0.3, \quad z(0) = 2, \quad P(0) = 0, \quad Z(0) = -1.$$

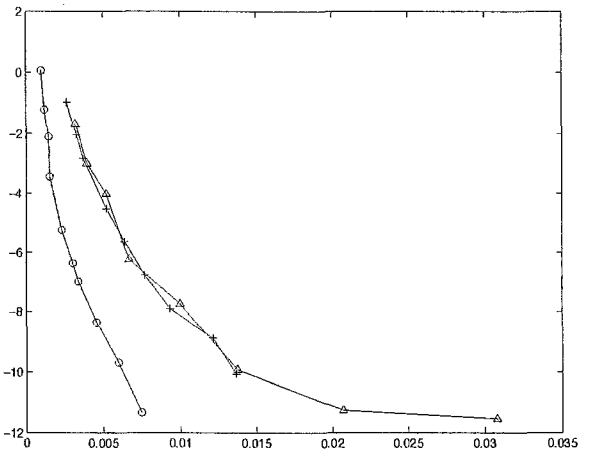
During the numerical simulations, the interval of integration was set between $t_0 = 0$ and $t_f = 70$. In Figure 19, CPU time (horizontal axis) is plotted versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for HBT(12)3, T(12), and DP(8, 7)13M. Based on this data, CPU PEG results are:

- CPU PEG of HBT(12)3 over T(12) is 137%;
- CPU PEG of HBT(12)3 over DP(8, 7)13M is 167%.

In Figure 20, the number of steps (horizontal axis) is plotted versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for HBT(12)3 and T(12). Based on this data, NS PEG of HBT(12)3 over T(12) is 27%. NS PEG result indicates that HBT(p)3 performs favourably over T(p) of the same order.

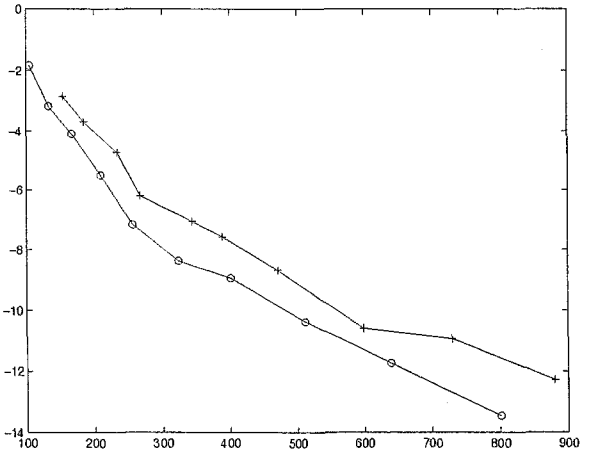
In Figure 21, the number of steps (horizontal axis) is plotted versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for HBT(40)3 and T(40). Based on this data, NS PEG of HBT(40)3 over T(40) is 12%. The NS PEG result confirms that HBT(p)3 performs favourably over T(p) of the same order at stringent tolerances.

In Table 18, a comparison is made between VS fixed order and VSVO HBT(p)3 at given tolerances using multiple precision. The last row in the table shows the average order of VSVO HBT(p)3. Once again, VSVO HBT(p)3 compares favourably to VS HBT(p)3 with improvements in CPU times as well as MGEE. For HBT(12)3 it



HBT(12)3 $\circ$, T(12) $\times$ and DP(8, 7)13M $\triangleright$

Figure 19: CPU time (horizontal axis) versus $\log_{10}(|\text{MGE}|)$ (vertical axis) for the equatorial main problem.

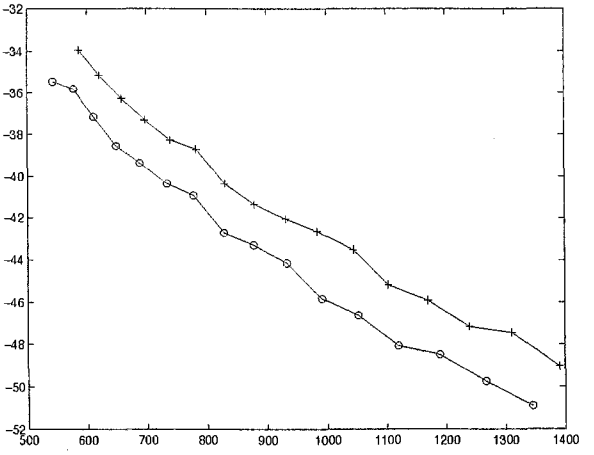


HBT(12)3 $\circ$, T(12) $\times$

Figure 20: Number of steps (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for the equatorial main problem.

Table 17: Number of steps, CPU time and maximum global error of HBT(40)3 and T(40) for the equatorial main problem using multiple precision.

EqMP	HBT(40)3			T(40)		
Tol	NS	CPU	MGEE	NS	CPU	MGEE
10^{-35}	543	19.73	$3.4739E-36$	587	24.47	$1.1516E-34$
10^{-40}	734	26.77	$4.3099E-41$	783	33.01	$1.8821E-39$
10^{-45}	992	35.99	$1.5094E-46$	1044	42.20	$2.9639E-44$
10^{-50}	1345	49.27	$1.3180E-51$	1391	58.52	$9.1661E-50$

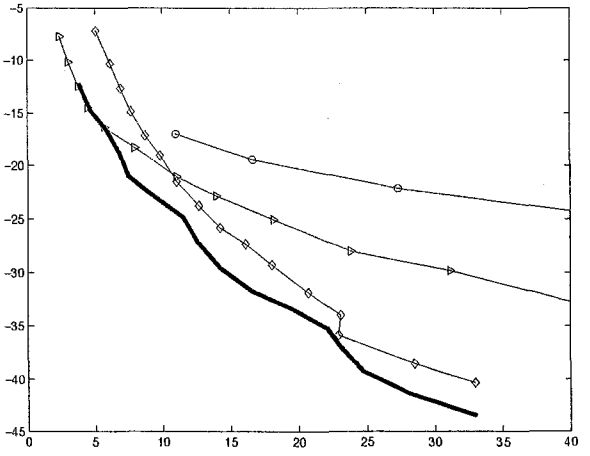


HBT(40)3 $\circ$, T(40) $\times$

Figure 21: Number of steps (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for the equatorial main problem using multiple precision.

Table 18: CPU time and MGEE of VS fixed order and VSVO HBT(p)3 using multiple precision for the equatorial main problem.

EqMP	tol=10^{-10}		tol=10^{-20}		tol=10^{-30}		tol=10^{-40}	
Order	CPU	MGEE	CPU	MGEE	CPU	MGEE	CPU	MGEE
12	2.31	$5.2175E-10$	27.32	$6.9241E-23$	n/a		n/a	
20	2.36	$1.9710E-08$	7.97	$4.7979E-19$	31.77	$1.6248E-30$	107.89	$7.7238E-43$
40	5.09	$6.7086E-08$	9.85	$8.7174E-20$	18.06	$5.7204E-30$	26.77	$4.3099E-41$
VO	3.79	$5.9640E-13$	9.06	$3.4240E-23$	19.24	$5.3048E-34$	34.03	$3.2769E-44$
Avg. Order	10		24		28		35	

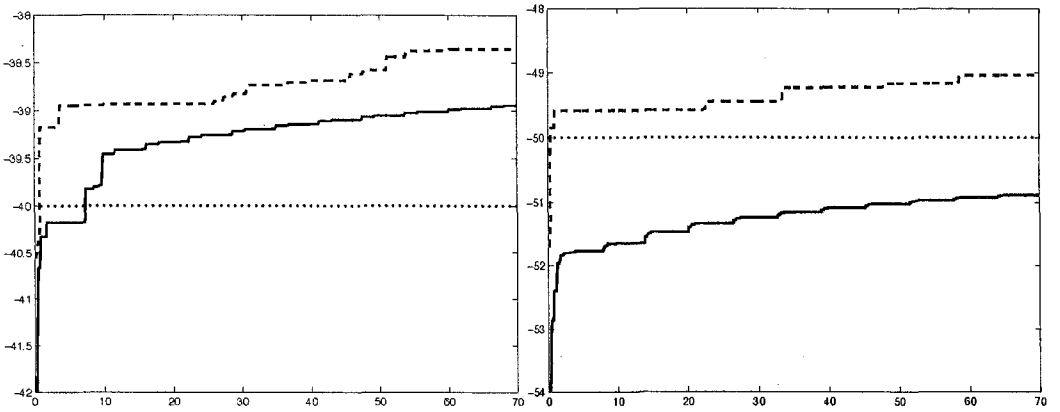


HBT(12)3 ○, HBT(20)3 ▷, HBT(40)3 ◇, and VSVO HBT(p)3 –

Figure 22: CPU time (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for the equatorial main problem using VS fixed order and VSVO HBT(p)3 implementation.

was not feasible to compute CPU times and MGEE at stringent tolerances, thus the results have been omitted. The results from Table 18 are plotted in Figure 22 with CPU time on the horizontal axis and $\log_{10}(|\text{MGEE}|)$ on the vertical axis.

In Figure 23, the evolution of MGEE is shown at different tolerance levels and a comparison is made between HBT(40)3 and T(40) at tolerances 10^{-40} and 10^{-50}. It is evident that HBT(40)3 performs better than T(40) at stringent tolerances. The results shown in Table 17 confirm that this is the case for tolerances up to 10^{-50}.



(a) `tol` $= 10^{-40}$ (b) `tol` $= 10^{-50}$

HBT(40)3 $-$, T(40) - -, and `tol` $\cdots$

Figure 23: The evolution of MGEE at different precision levels; time (horizontal axis) versus $\log_{10}(|\text{MGEE}|)$ (vertical axis) for the equatorial main problem using multiple precision.

Chapter 6

Conclusion and Future Work

A variable-step and variable-step variable-order one-step, 3-stage Hermite–Birkhoff–Taylor method of order p (HBT(p)3), was constructed by solving Vandermonde-type systems satisfying Taylor- and Runge–Kutta-type order conditions. By construction, HBT(p)3 uses less derivatives than a traditional Taylor method of the same order. The stability region of HBT(p)3 has a remarkably good shape. On the basis of CPU time versus the maximum global error, and the number of steps versus the maximum global energy error, HBT(p)3 wins over Taylor method of the same order and Dormand–Prince DP(8, 7)13M, on a set of problems commonly used by numerical analysts for testing the performance of ordinary differential equation solvers. The benefits of variable-step variable-order and multiple precision implementation of HBT(p)3 are noted throughout.

A follow up to the work done here would be to provide an automated procedure capable of taking a differential equation or a system of equations and producing necessary HBT(p)3 procedure in C++, similar to [8, 9, 22, 23]. This would allow a wider use of the method as it would not be necessary to possess high programming skills in order to produce necessary procedures. Another point that should be addressed is the limitation of the automatic differentiation procedure in terms of non-elementary or special functions.

Also, following tests on a wide variety of problems and peer-reviewing of the source

code, the method could be made available as part of a larger open-source mathematical software, providing a competitive alternative where high-precision numerical solutions of ordinary differential equations are sought.

Bibliography

[1] R. A. Adams, *Calculus: A complete course*, Addison-Wesley, Don Mills, 1999.

[2] 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.

[3] R. Barrio, *Performance of the Taylor series method for ODEs/DAEs*, Appl. Math. Comput., **163** (2005), pp. 525–545

[4] R. Barrio, *Sensitivity analysis of ODEs/DAEs using the Taylor series method*, SIAM J. Sc. Comp., **27**(6) (2006), pp. 1929–1947.

[5] 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.

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

[7] J. C. Butcher, *Numerical analysis of ordinary differential equations*, Wiley, Chichester, 1987.

[8] Y. F. Chang and G. F. Corliss, *ATOMFT: Solving ODEs and DAEs using Taylor series*, Computers Math. Applic., **28**(10–12) (1994), pp. 209–233.

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

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

[11] P. J. Davis and P. Rabinowitz, *Numerical integration*, Academic Press, New York, 1975.

[12] A. Deprit and R. M. W. Zahar, *Numerical integration of an orbit and its concomitant variations*, Z. Angew. Math. Phys., **17** (1966), pp. 425–430.

[13] W. C. Gear, *Numerical initial value problems in ordinary differential equations*, Prentice–Hall, Inc., Englewood Cliffs, 1971.

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

[15] E. Hairer, S. P. Nørsett and G. Wanner, *Solving ordinary differential equations I. Nonstiff problems*, Springer-Verlag. Berlin, 1993.

[16] E. Hairer and G. Wanner, *Solving ordinary differential equations II. Stiff and differential-algebraic problems*, Springer-Verlag. Berlin, 1993.

[17] M. Hénon and C. Heiles, *The applicability of the third integral of motion: Some numerical examples*, Astron. J., **69**(1964), pp. 73–79.

[18] J. Hoefkens, M. Berz, and K. Makino, *Computing validated solutions of implicit differential equations*, Adv. Comput. Math., **19** (2003), pp. 231–253.

[19] T. Y. Huang and K. Innanen, *A survey of multiderivative multistep integrators*, Astronomical J., **112**(3) (1996), pp. 1254–1262.

[20] 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.

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

[22] M. Lara, A. Elipe and M. Palacios, *Automatic programming of recurrent power series*, Math. Comput. Simul., **49**(1999), pp. 351–362.

[23] A. Jorba and M. Zou, *A software package for the numerical integration of ODE by means of high-order Taylor methods*, Experimental Mathematics, **14**(1)(2005), pp. 99–117.

[24] N. S. Nedialkov, K. R. Jackson, and G. F. Corliss, *Validated solutions of initial value problems for ordinary differential equations*, Appl. Math. Comput., **105** (1999), pp. 21–68.

[25] T. Nguyen-Ba, V. Bozic, E. Kengne and R. Vaillancourt, *One-step 4-stage Hermite–Birkhoff–Taylor ODE Solver of order 14*, Scientific Proceedings of Riga Technical University, **33**, Boundary Field Problems and Computer Simulation, 49th issue, (2007) 26–41. Also as technical report, **CRM–3251**, 2007.

[26] T. Nguyen-Ba, V. Bozic and R. Vaillancourt, *One-step 7-stage Hermite–Birkhoff–Taylor ODE solver of order 13*, International J. Pure Appl. Math., **43**(4) (2008) 569–592.

[27] T. Nguyen-Ba, V. Bozic, A. Przybylo and R. Vaillancourt, *One-step 9-stage Hermite–Birkhoff–Taylor ODE Solver of order 11*, University Scientific J., Telecommunications and Electronics Series, University of Technology and Life Sciences (UTP), Bydgoszcz, Poland. In press.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

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

[34] E. Rabe, *Determination and survey of periodic Trojan orbits in the restricted problem of three bodies*, Astronomical J., **66**(9) (1961), pp. 500–513.

[35] P. W. Sharp, *Numerical comparison of explicit Runge–Kutta pairs of orders four through eight*, Trans. on Mathematical Software, **17** (1991), pp. 500–513.

[36] J. F. Steffensen, *On the restricted problem of three bodies*, Danske Vid. Selsk., Mat.-fys. Medd., **30**(18) (1956), 17p.

[37] H. J. Stetter, *Global error estimation in ODE-solvers*, Lecture Notes in Mathematics **630**(1978), pp. 179–189.

[38] V. Szebehely, *Theory of orbits: The restricted problem of three bodies*, Academic Press, New York, 1967.