

UNIVERSITY OF BAHRAIN

College of Science

Department of Mathematics



Predictor-Corrector Methods for Solving Initial Value Problems

A Thesis Submitted in Partial Fulfillment of the Requirements
for the (Master) Degree in (Mathematics)

Submitted by

Noor A.Jalil Salman Alshakhoori

University number 20036092

Supervised by

Dr. Salman Hassan Abbas

(Associate Professor of Numerical Analysis)

University of Bahrain

Kingdom of Bahrain

January, 2015



University of Bahrain
Graduate Studies Council



Thesis Discussion Committee Decision

Student Name: <u>Noor A. Jali</u>	Student ID: <u>20036092</u>
Department: <u>Mathematics</u>	College: <u>Science</u>
Program: <u>MSc in Mathematics</u>	
Thesis Title: <u>Predictor - Corrector Methods for Solving Initial Value Problems</u>	
Supervisor Name: <u>Dr. Salman Abbas</u>	Student GPA: <u>3.37</u>

The Discussion Committee met on Sunday (day and date) 11/1/15 in Hall no. 541-200 (hall/room no.) 2000. Upon discussing the student whose details are stated above in public, the discussion committee recommends the following:

The student awarded grade A In the thesis subject MATH599

- ☐ Without making any changes to the thesis
☒ After making the modifications attached with this decision

Discussion Committee Members Approval:

Name

Signature

1. Dr. Salman Abbas (Advisor)
2. Prof. Nail Suleiman (Internal Examiner)
3. Prof. Christodoulos Sophocleous (External Examiner)

[Signature]
[Signature]





University of Bahrain
Graduate Studies Council

Thesis Modifications Approval Form

Student Name: <u>Noor A. Jalil</u>	Student ID: <u>20036092</u>
Department: <u>Mathematics</u>	College: <u>Science</u>
Program: <u>MSc in Mathematics</u>	
Thesis Title: <u>Predictor - Corrector Methods for Solving Initial value problem</u>	
Supervisor Name: <u>Dr Salman Abbas</u>	Student GPA: <u>3.37</u>

We would like to report that the student has carried out all the required modifications made by the discussion committee. Upon this, we recommend awarding the student grade A in thesis course number MATH599 based on the Discussion Committee recommendation.

Endorsement: ☒

Name	Signature	Date
1. <u>Dr Salman Abbas</u> (Advisor)		<u>11/1/2015</u>
2. <u>Prof Nail</u> (Internal Examiner)		<u>11/1/15</u>
3. <u>Dr Salman Abbas</u> (Department Coordinator)		<u>12/2/15</u>
4. <u>Dr Manai</u> (Department Chairman)		<u>18/2/2015</u>
5. <u>Dr Hashim</u> (College Dean)		<u>19-3-2015</u>



Copyright © by
Noor A.Jalil Salman Alshakhoori
University Of Bahrain
January, 2015
All Rights Reserved

ABSTRACT

Numerical solutions of ordinary differential equations (initial value problems) have been given considerable attention, concerning design and construction of algorithms for their applications in real life modelling, e.g., modeling of long-range transport of air pollutants in the atmosphere.

In this thesis a new block method of fifth order is derived for solving initial value problems of ordinary differential equations. Several problems from the literature were tested to demonstrate the method's accuracy using the software C++ language. We compare the accuracy of the new block method with known Runge-Kutta method of the same order.

To:

My great and whole-hearted mother and father

My fabulous and sweetheart husband and my handsome son, Ali

And all my family.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and gratitude to my supervisor Doctor Salman Abbas, for his invaluable help, suggestions and encouragement which made the task of completing this work possible.

Also, I would like to thank my University, University of Bahrain for giving me this opportunity to achieve, success and make my dream comes true when they open the registration in Master of Mathematics and for all facilities they provided. I am so grateful for Dr. Ahmed Saad Galal the Director of Graduate Programs for the Humanities for his valuable assistance, advice and encouragement. Many thanks and gratitude also to Mrs. Nawal Alorrayed for her help and for all who tried to help me especially in Information Technology College.

Finally, my deep and honest thanks and appreciation to all my family members and especially to the closest three in my life, my mother, my husband and my father for their true love, support and encouragement during studying and preparing my graduate degree, this meant so much to me and helped me during writing this thesis to put it in its final form.

My thesis is dedicated to my mother, my husband, my father and my son, Ali who had inspired me.

TABLE OF CONTENTS

SUBJECT	PAGE
THESIS DISCUSSION COMMITTEE DECISION	i
THESIS MODIFICATIONS APPROVAL FORM	ii
ABSTRACT	iv
DEDICATION	v
ACKNOWLEDGEMENTS	vi
TABLE OF CONTENTS	vii
LIST OF TABLES	x
LIST OF APPENDICES	xi
CHAPTER 1 INTRODUCTION	2
1.1 Statement of the problem and purpose	2
1.2 Background and history	3
1.2.1 Discrete variable methods	4
1.2.2 Runge-Kutta methods	6
1.2.3 Linear multistep methods	7
1.2.4 Predictor-corrector methods	9
1.3 Justification of the problem	11
1.4 Methods and procedures	12
1.5 Proposed chapters outline	14
CHAPTER 2 LITERATURE REVIEW	17

SUBJECT	PAGE
2.1 Block one-step methods	17
2.2 Predictor-corrector block methods	19
2.3 Multi-block methods	32
CHAPTER 3 METHODOLOGY: FIFTH ORDER BLOCK	36
3.1 Introduction	36
3.2 Derivation of a fifth order method	36
3.3 Principal local truncation error	42
3.4 Stability analysis	45
CHAPTER 4 NUMERICAL RESULTS	58
4.1 Description of the algorithm	58
4.2 Test problems	59
4.3 Computational results	62
4.4 Comparison with a Runge-Kutta method of the same order	64
CHAPTER 5 DISCUSSION	70
5.1 Summary and conclusion	70
5.2 Future work	71
BIBLIOGRAPHY	73
APPENDIX A COMPUTER LANGUAGE C++ FOR TEST PROBLEM (i)	

SUBJECT	PAGE
APPENDIX B COMPUTER LANGUAGE C++ FOR TEST PROBLEM (ii)	
APPENDIX C COMPUTER LANGUAGE C++ FOR TEST PROBLEM (iii)	
APPENDIX D COMPUTER LANGUAGE C++ FOR TEST PROBLEM (iv)	
SUMMARY IN ARABIC	

LIST OF TABLES

NUMBER	SUBJECT	PAGE
2.1.1	Formulas of the sequential two-point block scheme	18
4.3.1	Block method results using test problem (i)	62
4.3.2	Block method results using test problem (ii)	63
4.3.3	Block method results using test problem (iii)	63
4.3.4	Block method results using test problem (iv)	64
4.4.1	Comparison between Block method and Runge-Kutta method using test problem (i)	67
4.4.2	Comparison between Block method and Runge-Kutta method using test problem (ii)	67
4.4.3	Comparison between Block method and Runge-Kutta method using test problem (iii)	68
4.4.4	Comparison between Block method and Runge-Kutta method using test problem (iv)	68

LIST OF APPENDICES

	APPENDIX
A	COMPUTER LANGUAGE C++ FOR TEST PROBLEM (i)
B	COMPUTER LANGUAGE C++ FOR TEST PROBLEM (ii)
C	COMPUTER LANGUAGE C++ FOR TEST PROBLEM (iii)
D	COMPUTER LANGUAGE C++ FOR TEST PROBLEM (iv)

CHAPTER 1: INTRODUCTION

- 1.1 Statement of the problem and purpose
- 1.2 Background and history
 - 1.2.1 Discrete variable methods
 - 1.2.2 Runge-Kutta methods
 - 1.2.3 Linear multistep methods
 - 1.2.4 Predictor-corrector methods
- 1.3 Justification of the problem
- 1.4 Methods and procedures
- 1.5 Proposed chapters outline

CHAPTER 1

INTRODUCTION

1.1 Statement of the problem and purpose

The solution of initial value problems (IVPs) for ordinary differential equations using block methods has become an active area of research during the past few years. These differential equations arise in many applications in science, engineering and economics. Normally these equations are too complicated to solve analytically and we have to resort to numerical integration techniques to obtain an approximation to the solution.

The main objective in my thesis is to derive an efficient higher order method which can be used to solve initial value problems. So, the purpose of this thesis is to present a method to approximate the solution of initial value problems of ordinary differential equations using block method. A remarkable advantage of the block method is that it is easily implemented and gives a result with high accuracy.

The problem we will consider in this thesis is the numerical solution of the first order initial value problem

$$y' = f(x,y) \quad , \quad x_0 \leq x \leq x_n \quad (1.1.1)$$

$$y(x_0) = \beta \quad (1.1.2)$$

Here x and y are the independent and dependent variables, respectively, and y' denotes dy/dx is the 1st derivative of y with respect to x . In order

to define a unique solution to (1.1.1) it is necessary to give one initial condition as in (1.1.2). Sometimes, in many applications we will have a system of n simultaneous first order differential equations in n dependent variables $y_1, y_2, \dots, y_n$, which can be written as

$$\underline{y}' = \underline{f}(x, \underline{y}) \quad , \quad \underline{y}(x_0) = \underline{\beta} \quad (1.1.3)$$

where

$$\begin{aligned} \underline{y} &= (y_1, y_2, \dots, y_n)^T, \\ \underline{f} &= (f_1, f_2, \dots, f_n)^T, \\ \underline{\beta} &= (\beta_1, \beta_2, \dots, \beta_n)^T, \end{aligned}$$

where T denotes the transpose.

1.2 Background and history

Before we discuss the numerical methods which can be used to approximate the solution of initial value problem (1.1.3), we would like to know whether a unique solution exists. The existence of a unique solution is given by the following theorem, whose proof may be found in “Discrete variable methods in ordinary differential equations” by (Henrici, 1962).

Theorem 1.2

Let the function $\underline{f}(x, \underline{y})$ be continuous for all points in the region D defined by $a \leq x \leq b$, $-\infty < y_i < \infty$, $i = 1, \dots, n$. Suppose that there exist a constant $L < \infty$ such that

$$| \underline{f}(x, \underline{y}) - \underline{f}(x, \underline{y}^*) | \leq L | \underline{y} - \underline{y}^* | \quad (1.2.1)$$

for any $(x, \underline{y}), (x, \underline{y}^*) \in D$. Then for any initial condition $\underline{\beta}$, there exist a unique solution $\underline{y}(x)$ of the initial value problem (1.1.3), where $\underline{y}(x)$ is continuous and differentiable for all $(x, \underline{y}) \in D$. The requirement (1.2.1) is known as the Lipschitz condition, and the constant L is called Lipschitz constant, provided that $\underline{f}(x, \underline{y})$ is differentiable with respect to $\underline{y}$ for all $x \in [a, b]$ and all $\underline{y}$, it follows from the mean value theorem that (1.2.1) can be satisfied using

$$L = \sup_{(x, \underline{y}) \in D} \left| \frac{\partial \underline{f}}{\partial \underline{y}} \right|$$

Throughout this work we will assume that (1.1.3) satisfies this theorem and that (1.1.3) is well posed.

Having discussed the properties required of the differential equation itself we go on to discuss the numerical methods which are used to approximate the solution and examine which properties we require these numerical methods to have.

1.2.1 Discrete variable methods

Many basic numerical methods for the solution of

$$y' = f(x, y) \quad , \quad y(x_0) = \beta \quad , \quad x_0 \leq x \leq x_n \quad (1.2.2)$$

fall into the class known as discrete variable methods. These methods in their most basic form involve dividing the interval $[x_0, x_n]$ into a discrete mesh point x_i which is defined by $x_i = x_0 + ih, i = 0, 1, \dots, n$, where $n = \frac{x_n - x_0}{h}$ and $h > 0$ is the step size. In what follows we give some of the important theories associated with the numerical methods of (1.2.2).

we start off by introducing some terminology.

In particular let the exact solution of (1.2.2) at a point x_i by $y(x_i)$ and let the approximation at this point generated by the numerical method be y_i . Then we obtain the approximations $\{y_i\}$ sequentially by solving a finite difference equation, starting from x_0 and marching to x_i .

A general class of numerical methods for the integration of (1.2.2) and (1.1.3) is given by:

$$\sum_{j=0}^k \alpha_j y_{n+j} = h \phi_f(x_n; y_{n+k}, \dots, y_n; h), \quad n = 0, 1, \dots, N-k. \quad (1.2.3)$$

where α_j are constant, $\alpha_k \neq 0$ with given starting values $y_0, y_1, \dots, y_{k-1}$. If ϕ is independent of y_{n+k} then the method is explicit, otherwise the method is implicit. If $k = 1$, the method is known as one step method, while if $k > 1$, we have a multistep method (needs starting values) or k -step method.

Clearly what the user is interested in is to make sure that the numerical solution y_i is close to the true solution $y(x_i)$ for all i . Note that the quantity $\|y_n - y(x_n)\|$ is called the global truncation error or GTE at $x = x_n$, $0 \leq n \leq N$ where the global error is the error over the whole range.

The general class of methods contains the well-known methods which are Runge-Kutta methods and linear multistep methods, e.g.

$$\text{Rk: } k = 1, \phi = \sum_{r=0}^m c_r k_r$$

$$k_r = f(x_n + h a_r, y_n + \sum_{s=0}^m b_{rs} k_s)$$

$$\text{LMM: } \phi = \sum_{i=0}^k \beta_i f(x_{n+i}, y_{n+i})$$

where these methods are the most widely used for solving initial value problems such as (1.2.2) and (1.1.3).

1.2.2 Runge-Kutta methods

The general form of explicit Runge-Kutta methods is

$$y_{n+1} = y_n + h \phi(x_n, y_n, h) \quad (1.2.4)$$

$$\phi(x_n, y_n, h) = \sum_{r=1}^m b_r k_r$$

where

$$k_1 = f(x_n, y_n)$$

$$k_r = f(x_n + h c_r, y_n + h \sum_{s=1}^{r-1} a_{rs} k_s)$$

$$c_r = \sum_{s=1}^{r-1} a_{rs}, \quad r = 2, 3, \dots, m$$

where b_r , a_{rs} and c_r are real constants. The m -stage Runge-Kutta method involves m evaluations of f per step.

Implicit Runge-Kutta method is defined by

$$\begin{aligned}
 y_{n+1} &= y_n + h \phi(x_n, y_n, h) \\
 \phi(x_n, y_n, h) &= \sum_{r=1}^m b_r k_r \\
 k_r &= f(x_n + h c_r, y_n + h \sum_{s=1}^m a_{rs} k_s), \quad r = 1, 2, \dots, m
 \end{aligned} \tag{1.2.5}$$

The method (1.2.5) is fully implicit if $a_{rs} \neq 0$ for some $s > r$, the method is semi-implicit or semi-explicit if $a_{rs} = 0 \quad \forall s > r$, and the method is explicit if $a_{rs} = 0 \quad \forall s \geq r$.

1.2.3 Linear multistep methods

The general form of a linear k-step method is defined by

$$\sum_{j=0}^k \alpha_j y_{n+j} = h \sum_{j=0}^k \beta_j f_{n+j} \tag{1.2.6}$$

where $f_{n+j} = f(x_{n+j}, y_{n+j})$ and α_j and β_j are constants with $\alpha_k = 1$, $|\alpha_0| + |\beta_0| \neq 0$. The method is said to be linear because the expression in (1.2.6) is linear in f . We say that method (1.2.6) is explicit if $\beta_k = 0$ and implicit if $\beta_k \neq 0$. For this reason when using implicit methods we need a method to predict the starting values of the iteration. This method is called a predictor and the implicit method (1.2.6) is called a corrector.

The simplest example of linear multistep method is the explicit forward Euler method,

$$y_{n+1} = y_n + h f_n \tag{1.2.7}$$

and the implicit backward Euler method

$$y_{n+1} = y_n + h f_{n+1} \quad (1.2.8)$$

Formula (1.2.7) is used as a predictor and formula (1.2.8) as a corrector.

The local truncation error or LTE of (1.2.6) at $x_{n+k} \in [x_0, x_n]$ is given by

$$t_{n+k} = \sum_{j=0}^k \alpha_j y_{n+j} - h \phi_f(x_n; y_{n+k}, \dots, y_n; h)$$

The quantity t_{n+k} is the amount by which the true solution of (1.2.2) fails to satisfy (1.2.3).

To investigate the behavior of the local truncation error of the linear multistep method, we shall introduce the difference operator as follows:

$$L[y(x_n); h] = \sum_{j=0}^k [\alpha_j y(x_n + jh) - h \sum_{j=0}^k \beta_j y'(x_n + jh)] \quad (1.2.9)$$

where $y(x_n)$ is the exact solution of the initial value problem (1.2.2) and assuming that $y(x_n)$ is differentiable $p+1$ times then expanding (1.2.9) using a Taylor series, we obtain:

$$L[y(x_n); h] = c_0 y(x_n) + c_1 h y'(x_n) + \dots + c_q h^q y^{(q)}(x_n) + \dots \quad (1.2.10)$$

where

$$\begin{aligned}
c_0 &= \sum_{j=0}^k \alpha_j \\
c_1 &= \sum_{j=0}^k (j \alpha_j - \beta_j) \\
&\vdots \\
c_q &= \sum_{j=0}^k \left(\frac{j^q}{q!} \alpha_j - \frac{j^{q-1}}{(q-1)!} \beta_j \right), \quad q = 2, 3, 4, \dots
\end{aligned}$$

The linear multistep method (1.2.6) and the associated difference operator defined by (1.2.9) are said to be of order p , if in (1.2.10):

$$c_0 = c_1 = \dots = c_p = 0 \quad \text{and} \quad c_{p+1} \neq 0$$

The constant c_{p+1} is known as error constant and

$$c_{p+1} h^{p+1} y^{(p+1)}(x_n)$$

is called the principal local truncation error.

1.2.4 Predictor-corrector methods

Predictor-corrector methods have a long history in the numerical solution of IVPs. They have been proposed by (Moulton, 1926) in conjunction with Adams methods, by (Rosser, 1967) in conjunction with Runge-Kutta methods and by (Shampine and Watts, 1969) in terms of block methods. Also, two of the earliest papers on these methods for IVPs were by (Miranker and Liniger, 1967) and (Miranker, 1971), respectively, they point out that many standard predictor-corrector schemes

based on multistep formulas, such as

$$y_{n+1}^p = y_n^c + \frac{h}{2} (3f(x_n, y_n^c) - f(x_{n-1}, y_{n-1}^c)) \quad (1.2.11)$$

$$y_{n+1}^c = y_n^c + \frac{h}{2} (f(x_{n+1}, y_{n+1}^p) + f(x_n, y_n^c)) \quad (1.2.12)$$

are inherently sequential: y_n^c must be computed before y_{n+1}^p , which in turn must be computed before y_{n+1}^c , etc. However, the following pair of formulas

$$y_{n+1}^p = y_{n-1}^c + 2h f(x_n, y_n^c) \quad (1.2.13)$$

$$y_n^c = y_{n-1}^c + \frac{h}{2} (f(x_n, y_n^p) + f(x_{n-1}, y_{n-1}^c)) \quad (1.2.14)$$

both y_{n+1}^p and y_n^c from (1.2.13) and (1.2.14), respectively, can be computed simultaneously.

Let P denote an application of the predictor, C a single application of the corrector and E an evaluation of f in terms of known values of its argument.

So, a predictor-corrector method is an algorithm that proceeds in two steps. First, the prediction step calculates a rough approximation of the desired quantity. Second, the corrector step refines the initial approximation using another means.

We have two kinds of predictor-corrector formulas, the serial predictor-corrector formulas which has a sequence of computations or sequential calculations and a parallel predictor-corrector formulas where the corrector formula does not depend on the result of the corresponding predictor formula so, the sequence of computation is divided into two parts which may be executed simultaneously on separate processors.

For example one possible parallel method has been presented by (Franklin, 1978) and (Miranker and Linginer, 1967), is the fourth-order, two processors predictor-corrector method

$$\begin{aligned} y_{n+1}^p &= y_{n-1}^p + \frac{h}{3} (8f_n^p - 5f_{n-1}^c + 4f_{n-2}^c - f_{n-3}^c) \\ y_n^c &= y_{n-1}^c + \frac{h}{24} (9f_n^p + 19f_{n-1}^c - 5f_{n-2}^c + f_{n-3}^c) \end{aligned}$$

where the superscripts p and c refer to the predicted and corrected values and subscript n indicates the step number.

1.3 Justification of the problem

Solving initial value problems is considered to be arising concern and an active field of research in many applications, science and engineering, motivated by challenging real life problems, the last few years have seen the most widely used numerical methods for IVPs which were incorporated as basic tools including Runge-Kutta methods, multistep methods, different predictor-corrector schemes. This process has been encouraged to derive a block method with higher order. The method properties specifies the method order, the error constant and the method stability, in order to provide good solvers.

According to my knowledge there are some studies in this field are used to solve IVPs using block methods, so I will try to find my own way to solve these IVPs taking into account the reduction of the principal error. Therefore, this study attempts to fill the gap.

1.4 Methods and procedures

This section describes in some detail how we derive a new block method for the numerical solution of the first order IVP (1.2.2). In what follows, we give an overview of the block methods. The property of these methods is to produce simultaneously approximation to the solution of initial value problem at block points $x_{n+1}, x_{n+2}, \dots, x_{n+k}$ in each computational step. Only the solution at the last point in the block is used to start the next block. A block method yields a sequence of new estimates for y_i .

If $k \geq 1$ is the block size, then in simple case the value of x at which solutions are computed will be evenly separated. In other words, each basic cycle of the calculation has the potential to advance the solution by k new points in the x direction. Each such block can therefore be considered as a unit of calculation.

Let y_i denote the approximation solution to the exact solution $y(x_i)$ at $x = x_i$. Also f_i will denote the value of $f(x_i, y_i)$, the approximation for $y'(x_i)$. For $i = mk$, a block of solutions can be represented by the vector $Y_m = (y_{i+1}, y_{i+2}, \dots, y_{i+k})^T$ with y_{i+j} ($1 \leq j \leq k$) the generated solution at $x_{i+j} = x_i + jh$, where x_i the right hand end point of preceding block.

The formula of the block method can be expressed as:

$$Y_m = e y_i + h d f_i + h B F(Y_m) \quad (1.4.1)$$

where e and d are k -vectors, B is a $k \times k$ matrix, and F is a k -vector whose j^{th} entry is $f_{i+j} = f(x_{i+j}, y_{i+j})$, $1 \leq j \leq k$. As (1.4.1) is implicit in Y_m it has to be solved iteratively using, in the first instance, predicted solution values. A predictor equation for Y can be expressed in the form

$$Y_m^{(0)} = e y_i + h \tilde{d} f_i \quad (1.4.2)$$

where e and $\tilde{d}$ are k -vectors. Substitution of $Y_m^{(0)}$ into the right hand-side of (1.4.1) yields the block predictor-corrector (BPC) method

$$Y_m = e y_i + h d f_i + h B F(e y_i + h \tilde{d} f_i) \quad (1.4.3)$$

In accordance with the terminology used in the linear multistep case, this application is called PEC mode. Of course, one can continue this process by substituting the result of (1.4.3) into the right-hand side of (1.4.1) arriving at $P(EC)^v E^{1-\gamma}$ mode in which $\gamma = 0$ indicates that the final evaluation is done before proceeding to the next block.

The demand that the block method has at least principle error term of order p leads to p systems of equations of size $k+1$.

$$\sum_{j=0}^k \alpha_{r,j} = r, \quad \sum_{j=0}^k j \alpha_{r,j} = \frac{r^2}{2}, \quad \dots, \quad \sum_{j=0}^k j^{p-1} \alpha_{r,j} = \frac{r^p}{p} \quad (1.4.4)$$

$$\frac{r^{p+1}}{(p+1)!} - \sum_{j=1}^k \frac{j^p \alpha_{r,j}}{p!} = C_{p+1}^r, \quad r = 1, 2, \dots, k.$$

where C_{p+1}^r is the error constant of the method on the k^{th} processor and $\alpha_{r,j}$ is the coefficients of y'_{n+j} . The set of equations (1.4.4) represents the way in which the computation of the coefficients are found.

The block method can be written in the form

$$\begin{aligned} Y_{m+1}^{(0)} &= e \otimes y_i + h (A_p \otimes I) F(Y_m^{(v)}) \\ Y_{m+1}^{(l+1)} &= e \otimes y_i + h (A \otimes I) F(Y_{m+1}^{(l)}), \quad l = 0, \dots, v-1 \\ y_{m+1} &= e_{k+1}^T Y_{m+1}^{(v)} \end{aligned} \quad (1.4.5)$$

Apply (1.4.5) in $P(EC)^vE$ mode to the standard test equation

$$y' = \lambda y, \quad \lambda < 0$$

yields $Y_{m+1}^v = T^m(z)Y_0^v$. Consequently, the stability boundary is the largest number α such that $z \in (-\alpha, 0)$ when $\rho(T(z)) < 1$ where $\rho(T(z))$ denotes spectral radius of $T(z)$. Taking $v = p$, where p the order of the block method, then

$$\rho(T(z)) = \left| 1 + kz + \frac{(kz)^p}{p!} + \dots + \left(\frac{1}{(p+1)!} - \frac{C_{p+1}^k}{k^{p+1}} \right) (kz)^{p+1} \right|.$$

1.5 Proposed chapters outline

The thesis will contain five chapters:

Chapter One: (Introduction)

It includes:

- a. Statement of the problem and purpose
- b. Background and history
- c. Justification of the problem
- d. Methods and procedures with research overview

Chapter two: (Literature review)

This chapter is a survey of block methods, in this chapter, we include a review of the previous publications in the area of numerical solution of initial value problems.

Chapter three: (Methodology: Fifth order block method)

In this chapter, we show a derivation of a new block method for solving initial value problems, as well as we compute the local truncation error and give a stability analysis in details.

Chapter four: (Numerical results)

In this chapter, we first describe how to implement our algorithm, then we show our test problems and the results using the software C++ language, finally we present a comparison between the experimental results those obtained by our fifth order block method and Runge-Kutta method of the same order.

Chapter five: (Discussion)

In this chapter, we present the summary and conclusions and then recommendations and suggestions for future studies.

CHAPTER 2: LITERATURE REVIEW

2.1 Block one-step methods

2.2 Predictor-corrector block methods

2.3 Multi-block methods

CHAPTER 2

LITERATURE REVIEW

Block methods

The property of the block methods is to produce simultaneous approximation to the solution of initial value problem at block points $x_{n+1}, x_{n+2}, \dots, x_{n+k}$ in each computational step. Block method is introduced by (Worland, 1976) and may be applied either in one-step mode or in a predictor-corrector mode.

2.1 Block one-step methods

In one-step mode, a sequence of open and closed Newton-cotes formulas is applied to generate initial approximation to the solution $y(x)$ at each point in the current block. Then implicit formulas are applied iteratively until convergence is obtained to the maximum order of accuracy obtainable. Only the solution at the last point in the block is used to start the next block.

In one-step mode (Lambert, 1973) and (Rosser, 1967) described the sequence of the computations for this algorithm of formulas for the sequential two-point block scheme:

$$\begin{aligned}
y_{n+1,0} &= y_n + h y'_n \\
y_{n+1,1} &= y_n + \frac{h}{2} (y'_n + y'_{n+1,0}) \\
y_{n+2,1} &= y_n + 2h y'_{n+1,1} \\
y_{n+1,2} &= y_n + \frac{h}{12} (5y'_n + 8y'_{n+1,1} - y'_{n+2,1}) \\
y_{n+2,s+1} &= y_n + \frac{h}{3} (y'_n + 4y'_{n+1,2} + y'_{n+2,s}), \quad s = 1 \text{ or } 2.
\end{aligned} \tag{2.1.1}$$

The method (2.1.1) is used to produce the approximation y_{n+1} and y_{n+2} to $y(x_{n+1})$ and $y(x_{n+2})$ respectively. The order and the local truncation error for each formula of (2.1.1) are shown in the following table

Table (2.1.1)
Formulas of the sequential two-point block scheme

Formula number	Order	Local truncation error
1	1	$h^2/2 y_n^{(2)}(\xi)$
2	2	$-h^3/12 y_n^{(3)}(\xi)$
3	2	$h^3/3 y_n^{(3)}(\xi)$
4	3	$h^4/24 y_n^{(4)}(\xi)$
5	4	$-h^5/90 y_n^{(5)}(\xi)$

This procedure has five function evaluations for two-step block. However, in order to apply the algorithm at the next block to obtain y_{n+3} , y_{n+4} , the only starting value is y_{n+2} , and the loss of accuracy in y_{n+1} does not affect subsequent points and the algorithm remains of order four.

Also, we have the corresponding formulas for parallel two-point block scheme

$$\begin{aligned} y_{n+r,0} &= y_n + rh y'_n, \quad r = 1, 2 \\ y_{n+1,s+1} &= y_n + \frac{h}{12} (5y'_n + 8y'_{n+1,s} - y'_{n+2,s}) \\ y_{n+2,s+1} &= y_n + \frac{h}{3} (y'_n + 4y'_{n+1,s} + y'_{n+2,s}) \end{aligned} \quad (2.1.2)$$

where $s = 0, 1, 2, (3)$ as mentioned by (Worland, 1976).

The first formula is Euler's method which has order one, and the local truncation errors are

$$\frac{h^2}{2} y_n^{(2)}(\xi_{n+1}) \text{ and } 2h^2 y_n^{(2)}(\xi_{n+2}) \text{ for } r = 1 \text{ and } r = 2$$

respectively.

The local truncation errors for the second and third formulas of (2.1.2) are

$$\frac{h^4}{24} y_n^{(4)}(\xi_{n+1}) \text{ and } \frac{-h^5}{90} y_n^{(5)}(\xi_{n+2})$$

respectively. In this case, y_{n+1} and y_{n+2} are obtained simultaneously for each step.

2.2 Predictor-corrector block methods

Block schemes are readily adaptable to a predictor-corrector mode. The following algorithm is an example of this approach which is two-point, fourth-order block method of (Shampine and Watts, 1969). The predictor equations are

$$\begin{aligned} y_{n+1}^p &= \frac{1}{3} (y_{n-2}^c + y_{n-1}^c + y_n^c) + \frac{h}{6} (3f_{n-2}^c - 4f_{n-1}^c + 13f_n^c) \\ y_{n+2}^p &= \frac{1}{3} (y_{n-2}^c + y_{n-1}^c + y_n^c) + \frac{h}{12} (29f_{n-2}^c - 72f_{n-1}^c + 79f_n^c) \end{aligned} \quad (2.2.1)$$

The corrector equations are

$$\begin{aligned} y_{n+1}^c &= y_n^c + \frac{h}{12} (5f_n^c + 8f_{n+1}^p - f_{n+2}^p) \\ y_{n+2}^c &= y_n^c + \frac{h}{12} (f_n^c + 4f_{n+1}^p + f_{n+2}^p) \end{aligned} \quad (2.2.2)$$

in which the two predicted steps are computed in parallel, followed by the concurrent evaluation of the two corrections. Note that in this example the solution values are computed at equidistant points (which, in general, need not be the case). To implement the above algorithm, we need to compute the first two starting values y_{n-1} and y_{n-2} , after that the predictor equations (2.2.1) will be dependent on values taken from the previous block $(n, n-1, n-2)$. The corrector equations (2.2.2) need a single value from the previous block (n) and predicted values of current block $(n+1, n+2)$.

Also, one of the most interesting block predictor-corrector schemes for the parallel solution of ODEs, a block scheme based on the composite Trapezoidal rule by (Abbas and Delves, 1989) and its principal error term is linear in the block size.

Several authors have considered block methods for the solution of initial value problem (IVP)

$$y' = f(t, y), \quad y(t_0) = y_0, \quad (2.2.3)$$

where $y, f \in \mathbb{R}^s$. Let y_n denote the approximation to the exact solution $y(t_n)$ at $t = t_n$. Also, f_n will denote the value of $f(t_n, y_n)$, the approxima-

tion for $y'(t_n)$. For $n = mk$, a block of solutions can be represented by the vector $Y_m = (y_{n+1}, y_{n+2}, \dots, y_{n+k})^T$ with y_{n+j} ($1 \leq j \leq k$), the generated solution at $t_{n+j} = t_n + jh$, where t_n is the right-hand end point of the preceding block and h is the uniform spacing between solution values.

Adopting the notation of (Shampine and Watts, 1969), the formula for the block method can be expressed as

$$Y_m = ey_n + hdf_n + hBF(Y_m), \quad (2.2.4)$$

where e and d are k -vectors, B is a $k \times k$ matrix, and F is a k -vector whose j^{th} entry is $f_{n+j} = f(t_{n+j}, y_{n+j})$, $1 \leq j \leq k$. As (2.2.4) is implicit in Y_m , it has to be solved iteratively using, in the first instance, predicted solution values. A predictor equation for Y can be expressed in the form

$$Y_m^{(0)} = ey_n + h\tilde{d}f_n, \quad (2.2.5)$$

where e and $\tilde{d}$ are k -vectors. Substitution of $Y_m^{(0)}$ into the right-hand side of (2.2.4) yields the block predictor-corrector (BPC) method

$$Y_m = ey_n + hdf_n + hBF(ey_n + h\tilde{d}f_n). \quad (2.2.6)$$

Of course, one can continue this process by substituting the result of (2.2.6) into the right-hand side of (2.2.4) arriving at $P(EC)^v E^{1-\gamma}$ mode, in which $\gamma = 0$ indicates that a final evaluation is done before proceeding to the next block. (Abbas and Delves, 1989) considered this approach using an explicit Euler predictor and then corrected twice by a trapezoidal corrector applied in the composition case. This method can be computed in three steps for each equidistant step point $r = 1, \dots, k$ as

$$\begin{aligned}
y_{n+r}^{(0)} &= y_n + rh f(t_n, y_n), \\
y_{n+r}^{(1)} &= y_n + \frac{h}{2} f(t_n, y_n) + h \sum_{i=1}^{r-1} f(t_{n+i}, y_{n+i}^{(0)}) + \frac{h}{2} f(t_{n+r}, y_{n+r}^{(0)}), \\
y_{n+r}^{(2)} &= y_n + \frac{h}{2} f(t_n, y_n) + h \sum_{i=1}^{r-1} f(t_{n+i}, y_{n+i}^{(1)}) + \frac{h}{2} f(t_{n+r}, y_{n+r}^{(1)}).
\end{aligned} \tag{2.2.7}$$

With $Y_m^{(0)}$ given by (2.2.5), method (2.2.7) has $P(EC)^2$ form

$$\begin{aligned}
Y_m^{(0)} &= ey_n + h\tilde{d}f_n, \\
Y_m^{s+1} &= ey_n + h\tilde{d}f_n + hBF(Y_m^s), \quad s = 0, 1
\end{aligned} \tag{2.2.8}$$

where

$$e = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}, \quad \tilde{d} = \begin{bmatrix} 1 \\ 2 \\ \vdots \\ k \end{bmatrix}, \quad d = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \\ \vdots \\ \frac{1}{2} \end{bmatrix}, \quad B = \begin{bmatrix} \frac{1}{2} & 0 & \dots & \dots & \dots & 0 \\ 1 & \frac{1}{2} & 0 & \dots & \dots & 0 \\ & & \ddots & \ddots & & \vdots \\ 1 & \dots & \dots & 1 & \frac{1}{2} & 0 \\ 1 & \dots & \dots & 1 & 1 & \frac{1}{2} \end{bmatrix}.$$

While the local error of (2.2.8) has the form

$$-\frac{kh^3}{12} y^{(3)}(t_n) + O(h^4) \tag{2.2.9}$$

An other method, a fourth-order block method based on the composite Simpson rule by (Voss and Abbas, 1997), is developed for the solution of ordinary differential equations, like the block scheme based on the composite trapezoidal rule and its principal error term is linear in the block size. They consider a block method in which the order is fixed but the block length may be any value at the cost of increasing the truncation error.

The method developed using equation (2.2.5) to predict solutions to the problem and then applies a corrector in P(EC)^vE mode.

The method can be written as

$$\begin{aligned} Y_{m+1}^{(0)} &= e \otimes y_n + h(A_p \otimes I)F(Y_m^{(v)}), \\ Y_{m+1}^{(s+1)} &= e \otimes y_n + h(A \otimes I)F(Y_{m+1}^s), \quad s = 0, \dots, v-1, \\ y_{m+1} &= e_{k+1}^T Y_{m+1}^{(v)}, \end{aligned} \quad (2.2.10)$$

where

$$A_p = \begin{bmatrix} 0 & 0 & \dots & \dots & \dots & 0 \\ 0 & 0 & \dots & \dots & \dots & 1 \\ 0 & 0 & \dots & \dots & \dots & 2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \dots & \dots & k \end{bmatrix}$$

is the $(k+1) \times (k+1)$ prediction matrix. In connection with the block methods defined by (2.2.10), the matrix A of the method which was defined by (2.2.8), is given by $A = (\alpha_{i,j})$, $0 \leq i, j \leq k$ is the $(k+1) \times (k+1)$ Runge-Kutta matrix of coefficients, with

$$A = \begin{bmatrix} 0 & 0 & \dots & \dots & \dots & \dots & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & \dots & \dots & \dots & 0 \\ \frac{1}{2} & 1 & \frac{1}{2} & 0 & \dots & \dots & 0 \\ \frac{1}{2} & 1 & 1 & \frac{1}{2} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{2} & 1 & \dots & \dots & 1 & \frac{1}{2} & 0 \\ \frac{1}{2} & 1 & \dots & \dots & \dots & 1 & \frac{1}{2} \end{bmatrix}$$

The aim is to produce a fourth-order block scheme, as far as the order of accuracy is concerned, if we require a method of order p to be achieved then, as discussed by (Abbas, 1995), the local truncation error will normally be of order $(kh)^{p+1}$, a very high dependence upon the block length k . For this reason, they construct a fourth-order parallel block predictor-corrector method for the numerical solution for IVP (2.2.3) based on the composite Simpson rule and in order to attain the desired accuracy, the matrix of coefficients A in (2.2.10) must satisfy the following equations for $r = 1, 2, \dots, k$:

$$\sum_{j=0}^k \alpha_{r,j} = r \quad , \quad \sum_{j=0}^k j \alpha_{r,j} = \frac{r^2}{2} \quad , \quad \sum_{j=0}^k j^2 \alpha_{r,j} = \frac{r^3}{3} \quad , \quad \sum_{j=0}^k j^3 \alpha_{r,j} = \frac{r^4}{4} \quad (2.2.11)$$

For $k = 4$, a solution of the order equation (2.2.11) based on the composite Simpson rule has Runge-Kutta matrix

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ \frac{3}{8} & \frac{19}{24} & \frac{-5}{24} & \frac{1}{24} & 0 \\ \frac{1}{3} & \frac{4}{3} & \frac{1}{3} & 0 & 0 \\ \frac{3}{8} & \frac{9}{8} & \frac{9}{8} & \frac{3}{8} & 0 \\ \frac{1}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{1}{3} \end{bmatrix}$$

while with $k = 10$, the corresponding coefficients matrix becomes

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{8} & \frac{19}{24} & \frac{-5}{24} & \frac{1}{24} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{4}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{8} & \frac{9}{8} & \frac{9}{8} & \frac{3}{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{4}{3} & \frac{17}{24} & \frac{9}{8} & \frac{9}{8} & \frac{3}{8} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{17}{24} & \frac{9}{8} & \frac{9}{8} & \frac{3}{8} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{1}{3} & 0 & 0 \\ \frac{1}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{17}{24} & \frac{9}{8} & \frac{9}{8} & \frac{3}{8} & 0 \\ \frac{1}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{1}{3} \end{bmatrix}$$

Now, with four applications of the corrector, the principal local truncation error of SBPC is that of the corrector. The k^{th} component of the principal local truncation error is then given by

$$\varepsilon_k = y(t_{n+k}) - y(t_n) - h \sum_{j=0}^k \alpha_{k,j} y'_{n+j} \quad (2.2.12)$$

For k even, expansion of (2.2.12) yields

$$\begin{aligned} \varepsilon_k &= \left[\frac{k^5 h^5}{120} - \frac{4h^5}{72} (1^4 + 3^4 + 5^4 + 7^4 + \dots + (k-1)^4) \right. \\ &\quad \left. - \frac{2h^5}{72} (2^4 + 4^4 + 6^4 + \dots + (k-2)^4) - \frac{k^4 h^5}{72} \right] y^{(5)}(t_n), \quad (2.2.13) \\ &= \frac{-kh^5}{180} y^{(5)}(t_n), \end{aligned}$$

with the same result for k odd. From (2.2.13), the principal error is only linear in the block size k , and consequently, offers the potential for mas-

sive parallelism.

The linear stability properties of the block corrector formula (2.2.10) are determined through application to the test equation

$$y' = \lambda y, \quad \lambda < 0.$$

Letting $z = \lambda h$, in the case $k = 4$ (2.2.10) becomes

$$\begin{bmatrix} 1 - \frac{19}{24}z & \frac{5}{24}z & -\frac{1}{24}z & 0 \\ -\frac{4}{3}z & 1 - \frac{1}{3}z & 0 & 0 \\ -\frac{9}{8}z & -\frac{9}{8}z & 1 - \frac{3}{8}z & 0 \\ -\frac{4}{3}z & -\frac{2}{3}z & -\frac{4}{3}z & 1 - \frac{1}{3}z \end{bmatrix} \begin{bmatrix} y_{n+1} \\ y_{n+2} \\ y_{n+3} \\ y_{n+4} \end{bmatrix} = \begin{bmatrix} 1 + \frac{3}{8}z \\ 1 + \frac{1}{3}z \\ 1 + \frac{3}{8}z \\ 1 + \frac{1}{3}z \end{bmatrix} y_n.$$

With

$$Q = \begin{bmatrix} 1 - \frac{19}{24}z & \frac{5}{24}z & -\frac{1}{24}z & 0 \\ -\frac{4}{3}z & 1 - \frac{1}{3}z & 0 & 0 \\ -\frac{9}{8}z & -\frac{9}{8}z & 1 - \frac{3}{8}z & 0 \\ -\frac{4}{3}z & -\frac{2}{3}z & -\frac{4}{3}z & 1 - \frac{1}{3}z \end{bmatrix} \quad \text{and} \quad b = \begin{bmatrix} 1 + \frac{3}{8}z \\ 1 + \frac{1}{3}z \\ 1 + \frac{3}{8}z \\ 1 + \frac{1}{3}z \end{bmatrix},$$

and using Cramer's rule, we find that

$$y_{n+r} = \frac{D_r(z)}{D(z)} y_n, \quad r = 1, 2, 3, 4$$

where $D(z) = \det(Q)$ and $D_r(z) = \det(Q_r)$ and Q_r is obtained from Q by replacing its r^{th} column by the vector b . Absolute stability then requires

$$\left| \frac{D_k(z)}{D(z)} \right| < 1.$$

Consequently, the interval of stability of Trapezoidal block predictor-corrector approaches that of scaled explicit third-order Runge-Kutta methods which is approximately $(\frac{-2.5127}{k}, 0)$. Similarly, the interval of stability of Simpson block predictor-corrector approaches that of scaled explicit fifth-order Runge-Kutta methods which is approximately $(\frac{-3.2170}{k}, 0)$.

(Abbas, 1996) derived a new block methods of order two and three for the numerical solution of initial value problems. The coefficients matrix of these methods are chosen such that low powers of the block size appears in the principal local truncation errors.

Second order methods

Abbas gave two types of methods to solve (2.2.3), each of which is second order and applicable for arbitrary block size. For each method he computed the error constant C_3^k , the coefficients matrix A and interval of absolute stability $(-\alpha, 0)$, when the block size $k = 10$.

Type I

In this case $C_3^k = -\frac{k}{12}$ when k is even and $C_3^k = -\frac{1}{12}$ when k is odd.

The coefficients of the matrix A are given by

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{5}{6} & \frac{4}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 1 & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{5}{6} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 1 & 1 & 1 & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{5}{6} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{5}{6} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{1}{3} & 0 \\ \frac{1}{2} & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \frac{1}{2} \end{bmatrix}.$$

Note that, increasing the number of corrections is recommended to obtain good results and better accuracy. The interval of stability of this method is $(-0.2505, 0)$.

Type II

In this case $C_3^k = -\frac{1}{12k}$. The coefficients of the matrix A are given by

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{8} & \frac{5}{4} & \frac{3}{8} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{7}{9} & \frac{13}{9} & \frac{5}{18} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{11}{16} & \frac{26}{16} & \frac{11}{16} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{26}{30} & 1 & \frac{42}{30} & \frac{22}{30} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & \frac{37}{72} & \frac{71}{36} & \frac{37}{72} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & \frac{17}{21} & 1 & \frac{33}{21} & \frac{13}{21} & 1 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 1 & \frac{11}{32} & \frac{74}{32} & \frac{11}{32} & 1 & 1 & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 1 & 1 & \frac{1}{4} & \frac{5}{2} & \frac{1}{4} & 1 & 1 & 1 & \frac{1}{2} & 0 \\ \frac{1}{2} & 1 & 1 & 1 & \frac{7}{40} & \frac{106}{40} & \frac{7}{40} & 1 & 1 & 1 & \frac{1}{2} \end{bmatrix},$$

and the interval of stability of the method is $(-0.2512, 0)$. In general, for the k^{th} processor there are $k + 1$ coefficients; we choose $\alpha_{k,0} = \alpha_{k,k} = \frac{1}{2}$ and $(k - 4)$ coefficients to have value 1.

Third order methods

Type I

An interesting fact, associated with the principal error term $C_3^k = 0$, leads to a third order method. In the case of $p = 3$ and $k = 10$ with $C_4^k = \frac{k}{24}$ the matrix A is given by

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{5}{12} & \frac{2}{3} & \frac{-1}{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{4}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{3}{4} & \frac{3}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{2}{3} & \frac{5}{3} & \frac{2}{3} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & \frac{7}{12} & \frac{22}{12} & \frac{7}{12} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & \frac{1}{2} & 2 & \frac{1}{2} & 1 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & \frac{5}{12} & \frac{26}{12} & \frac{5}{12} & 1 & 1 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 1 & \frac{1}{3} & \frac{7}{3} & \frac{1}{3} & 1 & 1 & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 1 & 1 & \frac{1}{4} & \frac{5}{2} & \frac{1}{4} & 1 & 1 & 1 & \frac{1}{2} & 0 \\ \frac{1}{2} & 1 & 1 & 1 & \frac{1}{6} & \frac{8}{3} & \frac{1}{6} & 1 & 1 & 1 & \frac{1}{2} \end{bmatrix},$$

where the coefficients of the matrix A satisfy the first three order equations (1.4.4) and the interval of absolute stability is $(-0.2785, 0)$.

Type II

An interesting choice of finding the coefficients of the matrix A when $p = 3$ is $C_4^r = r \max(|C_4^1|, |C_4^2|) = \frac{r}{24}$ where $|C_4^1| = \frac{1}{24}$ and $|C_4^2| = 0$, which are the error terms for $r = 1$ and 2, respectively. The coefficients of the matrix A are given by

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{5}{12} & \frac{2}{3} & -\frac{1}{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{4}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{3}{4} & \frac{3}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{5}{6} & \frac{7}{6} & \frac{7}{6} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & \frac{7}{12} & \frac{22}{12} & \frac{7}{12} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & \frac{3}{4} & \frac{5}{4} & \frac{5}{4} & \frac{3}{4} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & \frac{5}{12} & \frac{26}{12} & \frac{5}{12} & 1 & 1 & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 1 & \frac{2}{3} & \frac{4}{3} & \frac{4}{3} & \frac{2}{3} & 1 & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & 1 & 1 & \frac{1}{4} & \frac{5}{2} & \frac{1}{4} & 1 & 1 & 1 & \frac{1}{2} & 0 \\ \frac{1}{2} & 1 & 1 & 1 & \frac{1}{6} & \frac{8}{3} & \frac{1}{6} & 1 & 1 & 1 & \frac{1}{2} \end{bmatrix}.$$

The error term constant $C_4^k = \frac{k}{24}$ and the interval of absolute stability is approximately $(-0.2785, 0)$.

So, as you seen, the author have investigated some second and third order block methods in this research, which can be utilize an arbitrary number of concurrent processors. It is shown that particular schemes can be derived whose principal error terms are relatively “small” functions of the block size while not deteriorating the interval of absolute stability. This allows a modest increase in parallelism over similar order methods.

Similar approaches have been considered by (Franklin, 1978), (Birta and Abou-Rabia, 1987), who have constructed block methods based on equidistant points, while (Mouney, Authi and Gayraud, 1991), have implemented an eight-point method of Birta and Abou-Rabia on a network of 32 transputers. Furthermore, (Chu and Hamilton, 1987), have constructed a two-block PECE method of order 4. However, its stability properties are inferior to those associated with the standard PECE approach based on the Adams methods of order 4 and 5. A different approach was adopted by (Miranker and Liniger, 1967), and later refined by Katz, (Franklin and Sen, 1977), in which predictions and higher corrections at past points move forward simultaneously in a diagonal wavefront.

So, the underlying feature of the traditional block method is that each application of the method generates a collection of equally spaced approximations to the solution. Block methods can be one-step as we have seen before in nature in which the values in a new block depend only on the last values in the preceding block or have a multi-block format in which the values depend on information from the previous blocks and thats what I am going to talk about.

2.3 Multi-block methods

A multi-block method is one in which several blocks of points are considered simultaneously. Methods of this type are considered by (Moody and Hans, 1987), and are described as k-block, r-points method. Time is divided into a series of blocks with each block containing r steps at which solutions to differential systems are to be found. The computation proceeds in blocks and is based on the computed values at the earlier

k blocks.

Let $x_k = x_0 + kh$ and y_k represents a numerical approximation to the exact solution $y(x_k)$ and $f_k = f(x_k, y_k)$. Denote the block number by $m = 0, 1, 2, \dots$ and r the block size. Let Y_m and F_m be vectors defined by

$$Y_m = [y_n, y_{n+1}, \dots, y_{n+r-1}]^T$$

$$F_m = [f_n, f_{n+1}, \dots, f_{n+r-1}]^T$$

then the general k-block, r-point method is a matrix finite difference equation of the form

$$Y_m = \sum_{i=1}^k A_i Y_{m-i} + h \sum_{i=0}^k B_i F_{m-i} \quad (2.3.1)$$

where the A's and B's are chosen to satisfy basic accuracy and the zero-stability. Note that when $r = 1$, then (2.3.1) is just the classical k-step method, and we say the method is explicit if $B_0 = 0$; otherwise it is implicit. The analogies of predictor-corrector modes for multi-block methods are given in "Parallel solution of ODEs by multi-block methods" by (Moody and Hans, 1987). Then the P(EC)^tE mode, for example, is given by

$$\begin{aligned} P: \quad Y_m^{(o)} &= \sum_{i=1}^k A_i^* Y_{m-i}^{(t)} + h \sum_{i=1}^k B_i^* F_{m-i}^{(t)}, \\ E: \quad F_m^{(s)} &= [f(x_n, y_n^{(s)}), \dots, f(x_{n+r-1}, y_{n+r-1}^{(s)})]^T, \\ C: \quad Y_m^{(s+1)} &= \sum_{i=1}^k A_i Y_{m-i}^{(t)} + h \sum_{i=1}^k B_i F_{m-i}^{(t)} + h B_0 F_m^{(s)} \text{ for } s = 0, \dots, t-1, \\ E: \quad F_m^{(t)} &= [f(x_n, y_n^{(t)}), \dots, f(x_{n+r-1}, y_{n+r-1}^{(t)})]^T, \end{aligned}$$

where the mark " $*$ " denotes to the predicted values.

It is noted that for economical purposes, one should use a corrector in which each component has equal or one order higher accuracy than the predictor. Here the local truncation error of PECE mode is equal to the order of the corrector.

Finally, after we have seen these different block methods in the literature I noticed that

- Previous studies provided criteria and guidelines both for solving initial value problems.
- Provided the knowledge level and informations to suggest a new method with higher order for solving IVP's.
- Most of these methods of order two, three and four.

I would like to say that I am going to work on my own fifth order block predictor-corrector method which is new method and I will show every thing about local truncation error, stability analysis and my results in details.

CHAPTER 3:
METHODOLOGY: FIFTH ORDER BLOCK METHOD

- 3.1 Introduction
- 3.2 Derivation of a fifth order method
- 3.3 Principal local truncation error
- 3.4 Stability analysis

CHAPTER 3

METHODOLOGY:

FIFTH ORDER BLOCK METHOD

3.1 Introduction

In this chapter, we describe a new fifth order method using a fixed step-size to solve ordinary differential equations defined by

$$y' = f(x, y), \quad y(x_0) = y_0, \quad x_0 \leq x \leq x_{\text{end}} \quad (3.1.1)$$

using one-step block methods, defined in general as

$$y_{n+r,0} = y_n + rh y'_n \quad (3.1.2)$$

$$y_{n+r,s+1} = y_n + h \sum_{j=0}^k \alpha_{rj} y'_{n+j,s}; \quad r = 1, 2, \dots, k; \quad s = 0, 1, 2, 3, 4. \quad (3.1.3)$$

We shall give a method to solve (3.1.1), which is fifth-order and applicable for arbitrary block length k . For our fifth-order method we compute the local truncation error and give a stability analysis.

3.2 Derivation of a fifth order method

The requirement that the method be fifth order for any block length k complicates the search for good method. Our aim was to produce fifth order method which might be very efficient to implement. As far as order of the accuracy and the stability are concerned, if we require a method of order S to be achieved then the local truncation error will normally be of order $k^{s+1}h^{s+1}$ which is a very high dependence upon the block length

k. We seek some methods where the local truncation errors have a lower power of k, as example, $k^s h^{s+1}$, but then over experience with order 2 methods suggests that the stability problem will be possibly serious.

Therefore, we consider the fifth order method

$$y_{n+r,s+1} = y_n + h (\alpha_{r,0} y'_n + \alpha_{r,1} y'_{n+1,s} + \alpha_{r,2} y'_{n+2,s} + \alpha_{r,3} y'_{n+3,s} + \alpha_{r,4} y'_{n+4,s});$$

$$r = 1, 2, \dots, k; \quad s = 0, 1, 2, 3, 4.$$

Note that the method will be fifth order if the coefficients of the method satisfy the five characteristic polynomials, which are given as:

$$\sum_{j=0}^k \alpha_{r,j} = r, \quad \sum_{j=0}^k j \alpha_{r,j} = \frac{r^2}{2}, \quad \sum_{j=0}^k j^2 \alpha_{r,j} = \frac{r^3}{3}$$

and $\sum_{j=0}^k j^3 \alpha_{r,j} = \frac{r^4}{4}, \quad \sum_{j=0}^k j^4 \alpha_{r,j} = \frac{r^5}{5}; \quad r = 1, 2, \dots, k.$ (3.2.2)

Therefore the coefficients of the method is given by

$$\begin{aligned} \alpha_{r,0} &= \frac{1}{120} r^5 - \frac{5}{48} r^4 + \frac{35}{72} r^3 - \frac{25}{24} r^2 + r \\ \alpha_{r,1} &= -\frac{1}{30} r^5 + \frac{3}{8} r^4 - \frac{13}{9} r^3 + 2r^2 \\ \alpha_{r,2} &= \frac{1}{20} r^5 - \frac{1}{2} r^4 + \frac{19}{12} r^3 - \frac{3}{2} r^2 \\ \alpha_{r,3} &= -\frac{1}{30} r^5 + \frac{7}{24} r^4 - \frac{7}{9} r^3 + \frac{2}{3} r^2 \\ \alpha_{r,4} &= \frac{1}{120} r^5 - \frac{1}{16} r^4 + \frac{11}{72} r^3 - \frac{1}{8} r^2; \quad r = 1, 2, \dots, k. \end{aligned}$$

After that, we shall give the values of the coefficients for $r = 1, 2, 3, \dots, 10$, as follows:

1. When $r = 1$:

$$\alpha_{1,0} = 251/720$$

$$\alpha_{1,1} = 323/360$$

$$\alpha_{1,2} = -11/30$$

$$\alpha_{1,3} = 53/360$$

$$\alpha_{1,4} = -19/720$$

2. When $r = 2$:

$$\alpha_{2,0} = 29/90$$

$$\alpha_{2,1} = 62/45$$

$$\alpha_{2,2} = 4/15$$

$$\alpha_{2,3} = 2/45$$

$$\alpha_{2,4} = -1/90$$

3. When $r = 3$:

$$\alpha_{3,0} = 27/80$$

$$\alpha_{3,1} = 51/40$$

$$\alpha_{3,2} = 9/10$$

$$\alpha_{3,3} = 21/40$$

$$\alpha_{3,4} = -3/80$$

4. When $r = 4$:

$$\alpha_{4,0} = 14/45$$

$$\alpha_{4,1} = 64/45$$

$$\alpha_{4,2} = 8/15$$

$$\alpha_{4,3} = 64/45$$

$$\alpha_{4,4} = 14/45$$

5. When $r = 5$:

$$\alpha_{5,0} = 95/144$$

$$\alpha_{5,1} = -25/72$$

$$\alpha_{5,2} = 25/6$$

$$\alpha_{5,3} = -175/72$$

$$\alpha_{5,4} = 425/144$$

6. When $r = 6$:

$$\alpha_{6,0} = 33/10$$

$$\alpha_{6,1} = -66/5$$

$$\alpha_{6,2} = 144/5$$

$$\alpha_{6,3} = -126/5$$

$$\alpha_{6,4} = 123/10$$

7. When $r = 7$:

$$\alpha_{7,0} = 9107/720$$

$$\alpha_{7,1} = -20629/360$$

$$\alpha_{7,2} = 3283/30$$

$$\alpha_{7,3} = -33859/360$$

$$\alpha_{7,4} = 26117/720$$

8. When $r = 8$:

$$\alpha_{8,0} = 1648/45$$

$$\alpha_{8,1} = -7552/45$$

$$\alpha_{8,2} = 4576/15$$

$$\alpha_{8,3} = -11392/45$$

$$\alpha_{8,4} = 3928/45$$

9. When $r = 9$:

$$\alpha_{9,0} = 7011/80$$

$$\alpha_{9,1} = -15957/40$$

$$\alpha_{9,2} = 7047/10$$

$$\alpha_{9,3} = -22707/40$$

$$\alpha_{9,4} = 14661/80$$

10. When $r = 10$:

$$\alpha_{10,0} = 3305/18$$

$$\alpha_{10,1} = -7450/9$$

$$\alpha_{10,2} = 4300/3$$

$$\alpha_{10,3} = -10150/9$$

$$\alpha_{10,4} = 6275/18$$

Hence,

$$y_{n+r,s+1} = y_n + h (\alpha_{r,0} y'_n + \alpha_{r,1} y'_{n+1,s} + \alpha_{r,2} y'_{n+2,s} + \alpha_{r,3} y'_{n+3,s} + \alpha_{r,4} y'_{n+4,s}); \quad r = 1, 2, \dots, 10; \quad s = 0, 1, 2, 3, 4.$$

is equivalent to

$$\begin{pmatrix} y_{n+1} \\ y_{n+2} \\ y_{n+3} \\ y_{n+4} \\ y_{n+5} \\ y_{n+6} \\ y_{n+7} \\ y_{n+8} \\ y_{n+9} \\ y_{n+10} \end{pmatrix} = \begin{pmatrix} y_n \\ y_n \\ y_n \\ y_n \\ y_n \\ y_n \\ y_n \\ y_n \\ y_n \\ y_n \end{pmatrix} + h \begin{pmatrix} 251/720 & 323/360 & -11/30 & 53/360 & -19/720 \\ 29/90 & 62/45 & 4/15 & 2/45 & -1/90 \\ 27/80 & 51/40 & 9/10 & 21/40 & -3/80 \\ 14/45 & 64/45 & 8/15 & 64/45 & 14/45 \\ 95/144 & -25/72 & 25/6 & -175/72 & 425/144 \\ 33/10 & -66/5 & 144/5 & -126/5 & 123/10 \\ 9107/720 & -20629/360 & 3283/30 & -33859/360 & 26117/720 \\ 1648/45 & -7552/45 & 4576/15 & -11392/45 & 3928/45 \\ 7011/80 & -15957/40 & 7047/10 & -22707/40 & 14661/80 \\ 3305/18 & -7450/9 & 4300/3 & -10150/9 & 6275/18 \end{pmatrix} \begin{pmatrix} y'_n \\ y'_{n+1} \\ y'_{n+2} \\ y'_{n+3} \\ y'_{n+4} \end{pmatrix}$$

Now, we would give the formulae of the method when $k = 5$ and 10.

1. When $k = 5$, the formulae are

$$\begin{aligned}
y_{n+1} &= y_n + h \left[\frac{251}{720} y'_n + \frac{323}{360} y'_{n+1} - \frac{11}{30} y'_{n+2} + \frac{53}{360} y'_{n+3} - \frac{19}{720} y'_{n+4} \right] \\
y_{n+2} &= y_n + h \left[\frac{29}{90} y'_n + \frac{62}{45} y'_{n+1} + \frac{4}{15} y'_{n+2} + \frac{2}{45} y'_{n+3} - \frac{1}{90} y'_{n+4} \right] \\
y_{n+3} &= y_n + h \left[\frac{27}{80} y'_n + \frac{51}{40} y'_{n+1} + \frac{9}{10} y'_{n+2} + \frac{21}{40} y'_{n+3} - \frac{3}{80} y'_{n+4} \right] \\
y_{n+4} &= y_n + h \left[\frac{14}{45} y'_n + \frac{64}{45} y'_{n+1} + \frac{8}{15} y'_{n+2} + \frac{64}{45} y'_{n+3} + \frac{14}{45} y'_{n+4} \right] \\
y_{n+5} &= y_n + h \left[\frac{95}{144} y'_n - \frac{25}{72} y'_{n+1} + \frac{25}{6} y'_{n+2} - \frac{175}{72} y'_{n+3} + \frac{425}{144} y'_{n+4} \right]
\end{aligned} \tag{3.2.3}$$

2. When $k = 10$, the formulae are

$$\begin{aligned}
y_{n+1} &= y_n + h \left[\frac{251}{720} y'_n + \frac{323}{360} y'_{n+1} - \frac{11}{30} y'_{n+2} + \frac{53}{360} y'_{n+3} - \frac{19}{720} y'_{n+4} \right] \\
y_{n+2} &= y_n + h \left[\frac{29}{90} y'_n + \frac{62}{45} y'_{n+1} + \frac{4}{15} y'_{n+2} + \frac{2}{45} y'_{n+3} - \frac{1}{90} y'_{n+4} \right] \\
y_{n+3} &= y_n + h \left[\frac{27}{80} y'_n + \frac{51}{40} y'_{n+1} + \frac{9}{10} y'_{n+2} + \frac{21}{40} y'_{n+3} - \frac{3}{80} y'_{n+4} \right] \\
y_{n+4} &= y_n + h \left[\frac{14}{45} y'_n + \frac{64}{45} y'_{n+1} + \frac{8}{15} y'_{n+2} + \frac{64}{45} y'_{n+3} + \frac{14}{45} y'_{n+4} \right] \\
y_{n+5} &= y_n + h \left[\frac{95}{144} y'_n - \frac{25}{72} y'_{n+1} + \frac{25}{6} y'_{n+2} - \frac{175}{72} y'_{n+3} + \frac{425}{144} y'_{n+4} \right] \\
y_{n+6} &= y_n + h \left[\frac{33}{10} y'_n - \frac{66}{5} y'_{n+1} + \frac{144}{5} y'_{n+2} - \frac{126}{5} y'_{n+3} + \frac{123}{10} y'_{n+4} \right] \\
y_{n+7} &= y_n + h \left[\frac{9107}{720} y'_n - \frac{20629}{360} y'_{n+1} + \frac{3283}{30} y'_{n+2} - \frac{33859}{360} y'_{n+3} + \frac{26117}{720} y'_{n+4} \right] \\
y_{n+8} &= y_n + h \left[\frac{1648}{45} y'_n - \frac{7552}{45} y'_{n+1} + \frac{4576}{15} y'_{n+2} - \frac{11392}{45} y'_{n+3} + \frac{3928}{45} y'_{n+4} \right] \\
y_{n+9} &= y_n + h \left[\frac{7011}{80} y'_n - \frac{15957}{40} y'_{n+1} + \frac{7047}{10} y'_{n+2} - \frac{22707}{40} y'_{n+3} + \frac{14661}{80} y'_{n+4} \right] \\
y_{n+10} &= y_n + h \left[\frac{3305}{18} y'_n - \frac{7450}{9} y'_{n+1} + \frac{4300}{3} y'_{n+2} - \frac{10150}{9} y'_{n+3} + \frac{6275}{18} y'_{n+4} \right]
\end{aligned} \tag{3.2.4}$$

3.3 Principal local truncation error

The r^{th} component of local truncation error of (3.2.1) is given by

$$\begin{aligned}
\varepsilon_r &= y_{n+r,s+1} - y_n - h (\alpha_{r,0} y'_n + \alpha_{r,1} y'_{n+1,s} + \alpha_{r,2} y'_{n+2,s} \\
&\quad + \alpha_{r,3} y'_{n+3,s} + \alpha_{r,4} y'_{n+4,s})
\end{aligned} \tag{3.3.1}$$

where $y_{n+r} = y(x_{n+r})$ denotes the exact value of the solution at x_{n+r} .

ε_r can be expressed as a power series in h , and the remained term of Taylor's series expansion used to derive ε_r .

Therefore, the k^{th} component of local truncation error of (3.2.1) is given by:

$$\begin{aligned}\varepsilon_k = & y_{n+k,s+1} - y_n - h (\alpha_{k,0} y'_n + \alpha_{k,1} y'_{n+1,s} + \alpha_{k,2} y'_{n+2,s} \\ & + \alpha_{k,3} y'_{n+3,s} + \alpha_{k,4} y'_{n+4,s})\end{aligned}$$

where

$$\begin{aligned}\alpha_{k,0} &= \frac{1}{120} k^5 - \frac{5}{48} k^4 + \frac{35}{72} k^3 - \frac{25}{24} k^2 + k \\ \alpha_{k,1} &= -\frac{1}{30} k^5 + \frac{3}{8} k^4 - \frac{13}{9} k^3 + 2k^2 \\ \alpha_{k,2} &= \frac{1}{20} k^5 - \frac{1}{2} k^4 + \frac{19}{12} k^3 - \frac{3}{2} k^2 \\ \alpha_{k,3} &= -\frac{1}{30} k^5 + \frac{7}{24} k^4 - \frac{7}{9} k^3 + \frac{2}{3} k^2 \\ \alpha_{k,4} &= \frac{1}{120} k^5 - \frac{1}{16} k^4 + \frac{11}{72} k^3 - \frac{1}{8} k^2\end{aligned}$$

and using Taylor series:

$$\begin{aligned}y_{n+k} &= y(x_{n+k}) = y(x_n + kh) \\ &= y(x_n) + (kh) y'(x_n) + \frac{(kh)^2}{2!} y''(x_n) + \frac{(kh)^3}{3!} y'''(x_n) \\ &\quad + \frac{(kh)^4}{4!} y^{iv}(x_n) + \frac{(kh)^5}{5!} y^v(x_n) + \frac{(kh)^6}{6!} y^{vi}(x_n) + O(h^7) \\ y'_{n+1} &= y'(x_n + h) = y'(x_n) + h y''(x_n) + \frac{h^2}{2!} y'''(x_n) + \frac{h^3}{3!} y^{iv}(x_n) \\ &\quad + \frac{h^4}{4!} y^v(x_n) + \frac{h^5}{5!} y^{vi}(x_n) + O(h^6)\end{aligned}$$

$$\begin{aligned}
y'_{n+2} &= y'(x_n + 2h) := y'(x_n) + (2h)y''(x_n) + \frac{(2h)^2}{2!}y'''(x_n) \\
&\quad + \frac{(2h)^3}{3!}y^{iv}(x_n) + \frac{(2h)^4}{4!}y^v(x_n) + \frac{(2h)^5}{5!}y^{vi}(x_n) + O(h^6) \\
y'_{n+3} &= y'(x_n + 3h) = y'(x_n) + (3h)y''(x_n) + \frac{(3h)^2}{2!}y'''(x_n) \\
&\quad + \frac{(3h)^3}{3!}y^{iv}(x_n) + \frac{(3h)^4}{4!}y^v(x_n) + \frac{(3h)^5}{5!}y^{vi}(x_n) + O(h^6) \\
y'_{n+4} &= y'(x_n + 4h) = y'(x_n) + (4h)y''(x_n) + \frac{(4h)^2}{2!}y'''(x_n) \\
&\quad + \frac{(4h)^3}{3!}y^{iv}(x_n) + \frac{(4h)^4}{4!}y^v(x_n) + \frac{(4h)^5}{5!}y^{vi}(x_n) + O(h^6)
\end{aligned}$$

we get

$$\begin{aligned}
\varepsilon_k &= y(x_n) + (kh)y'(x_n) + \frac{(kh)^2}{2!}y''(x_n) + \frac{(kh)^3}{3!}y'''(x_n) + \frac{(kh)^4}{4!}y^{iv}(x_n) \\
&\quad + \frac{(kh)^5}{5!}y^v(x_n) + \frac{(kh)^6}{6!}y^{vi}(x_n) + O(h^7) - y(x_n) - h\alpha_{k,0}y'(x_n) \\
&\quad - h\alpha_{k,1}\left(y'(x_n) + hy''(x_n) + \frac{h^2}{2!}y'''(x_n) + \frac{h^3}{3!}y^{iv}(x_n) \right. \\
&\quad \left. + \frac{h^4}{4!}y^v(x_n) + \frac{h^5}{5!}y^{vi}(x_n) + O(h^6)\right) - h\alpha_{k,2}\left(y'(x_n) \right. \\
&\quad \left. + (2h)y''(x_n) + \frac{(2h)^2}{2!}y'''(x_n) + \frac{(2h)^3}{3!}y^{iv}(x_n) + \frac{(2h)^4}{4!}y^v(x_n) \right. \\
&\quad \left. + \frac{(2h)^5}{5!}y^{vi}(x_n) + O(h^6)\right) - h\alpha_{k,3}\left(y'(x_n) + (3h)y''(x_n) \right. \\
&\quad \left. + \frac{(3h)^2}{2!}y'''(x_n) + \frac{(3h)^3}{3!}y^{iv}(x_n) + \frac{(3h)^4}{4!}y^v(x_n) + \frac{(3h)^5}{5!}y^{vi}(x_n) \right. \\
&\quad \left. + O(h^6)\right) - h\alpha_{k,4}\left(y'(x_n) + (4h)y''(x_n) + \frac{(4h)^2}{2!}y'''(x_n) \right. \\
&\quad \left. + \frac{(4h)^3}{3!}y^{iv}(x_n) + \frac{(4h)^4}{4!}y^v(x_n) + \frac{(4h)^5}{5!}y^{vi}(x_n) + O(h^6)\right)
\end{aligned}$$

Therefore,

$$\begin{aligned}
\varepsilon_k = & \left(k - \alpha_{k,0} - \alpha_{k,1} - \alpha_{k,2} - \alpha_{k,3} - \alpha_{k,4} \right) h y'(x_n) \\
& + \left(\frac{k^2}{2!} - \alpha_{k,1} - 2\alpha_{k,2} - 3\alpha_{k,3} - 4\alpha_{k,4} \right) h^2 y''(x_n) \\
& + \left(\frac{k^3}{3!} - \frac{1}{2}\alpha_{k,1} - 2\alpha_{k,2} - \frac{9}{2}\alpha_{k,3} - 8\alpha_{k,4} \right) h^3 y'''(x_n) \\
& + \left(\frac{k^4}{4!} - \frac{1}{6}\alpha_{k,1} - \frac{4}{3}\alpha_{k,2} - \frac{9}{2}\alpha_{k,3} - \frac{32}{3}\alpha_{k,4} \right) h^4 y^{iv}(x_n) \\
& + \left(\frac{k^5}{5!} - \frac{1}{24}\alpha_{k,1} - \frac{2}{3}\alpha_{k,2} - \frac{27}{8}\alpha_{k,3} - \frac{32}{3}\alpha_{k,4} \right) h^5 y^v(x_n) \\
& + \left(\frac{k^6}{6!} - \frac{1}{120}\alpha_{k,1} - \frac{4}{15}\alpha_{k,2} - \frac{81}{40}\alpha_{k,3} - \frac{128}{15}\alpha_{k,4} \right) h^6 y^{vi}(x_n) \\
& + O(h^7) \\
\therefore \varepsilon_k = & k^2 \left(\frac{1}{720}k^4 - \frac{1}{60}k^3 + \frac{7}{96}k^2 - \frac{5}{36}k + \frac{1}{10} \right) h^6 y_n^{vi} = O(k^6 h^6)
\end{aligned}$$

Then we can express $\underline{\varepsilon}$ as:

$$\left[\frac{3}{160}h^6 y_n^{vi}, \frac{1}{90}h^6 y_n^{vi}, \frac{3}{160}h^6 y_n^{vi}, \dots, \frac{645}{2}h^6 y_n^{vi}, \dots, k^2 \left(\frac{1}{720}k^4 - \frac{1}{60}k^3 + \frac{7}{96}k^2 - \frac{5}{36}k + \frac{1}{10} \right) h^6 y_n^{vi} \right]^T$$

for $r = 1, 2, 3, \dots, k$. So, the local errors of our method (3.2.1) estimated depend on $k^6 h^6$. The global errors of this method is of order $k^5 h^5$.

3.4 Stability analysis

The following analysis is carried out according to the stability of a fixed step size studied in “A stable block implicit methods” by (Watts and Shampine, 1972). To discuss the stability of this block method “fifth order method”, we will check the cases when $k = 5$ and $k = 10$.

So, firstly we apply this method when $k = 5$;

$$\begin{aligned}
y_{n+r,s+1} = & y_n + h \left(\alpha_{r,0} y'_n + \alpha_{r,1} y'_{n+1,s} + \alpha_{r,2} y'_{n+2,s} + \alpha_{r,3} y'_{n+3,s} \right. \\
& \left. + \alpha_{r,4} y'_{n+4,s} \right); \quad r = 1, 2, \dots, 5; \quad s = 0, 1, 2, 3, 4.
\end{aligned}$$

Now, apply the method to the test equation $y' = \lambda y$, and let $H = \lambda h$, we get

$$y_{n+r} = y_n + H (\alpha_{r,0} y_n + \alpha_{r,1} y_{n+1} + \alpha_{r,2} y_{n+2} + \alpha_{r,3} y_{n+3} + \alpha_{r,4} y_{n+4});$$

$$r = 1, 2, \dots, 5.$$

Hence,

$$\begin{aligned} y_{n+1} &= y_n + H \left[\frac{251}{720} y_n + \frac{323}{360} y_{n+1} - \frac{11}{30} y_{n+2} + \frac{53}{360} y_{n+3} - \frac{19}{720} y_{n+4} \right] \\ y_{n+2} &= y_n + H \left[\frac{29}{90} y_n + \frac{62}{45} y_{n+1} + \frac{4}{15} y_{n+2} + \frac{2}{45} y_{n+3} - \frac{1}{90} y_{n+4} \right] \\ y_{n+3} &= y_n + H \left[\frac{27}{80} y_n + \frac{51}{40} y_{n+1} + \frac{9}{10} y_{n+2} + \frac{21}{40} y_{n+3} - \frac{3}{80} y_{n+4} \right] \\ y_{n+4} &= y_n + H \left[\frac{14}{45} y_n + \frac{64}{45} y_{n+1} + \frac{8}{15} y_{n+2} + \frac{64}{45} y_{n+3} + \frac{14}{45} y_{n+4} \right] \\ y_{n+5} &= y_n + H \left[\frac{95}{144} y_n - \frac{25}{72} y_{n+1} + \frac{25}{6} y_{n+2} - \frac{175}{72} y_{n+3} + \frac{425}{144} y_{n+4} \right] \end{aligned}$$

Then the general form when $k = 5$ of this method in this representation is given by:

$$\begin{pmatrix} 1 - \frac{323}{360}H & \frac{11}{30}H & -\frac{53}{360}H & \frac{19}{720}H & 0 \\ -\frac{62}{45}H & 1 - \frac{4}{15}H & -\frac{2}{45}H & \frac{1}{90}H & 0 \\ -\frac{51}{40}H & -\frac{9}{10}H & 1 - \frac{21}{40}H & \frac{3}{80}H & 0 \\ -\frac{64}{45}H & -\frac{8}{15}H & -\frac{64}{45}H & 1 - \frac{14}{45}H & 0 \\ \frac{25}{72}H & -\frac{25}{6}H & \frac{175}{72}H & -\frac{425}{144}H & 1 \end{pmatrix} \begin{pmatrix} y_{n+1} \\ y_{n+2} \\ y_{n+3} \\ y_{n+4} \\ y_{n+5} \end{pmatrix} = \begin{pmatrix} 1 + \frac{251}{720}H \\ 1 + \frac{29}{90}H \\ 1 + \frac{27}{80}H \\ 1 + \frac{14}{45}H \\ 1 + \frac{95}{144}H \end{pmatrix} y_n \quad (3.4.1)$$

Let

$$A = \begin{pmatrix} 1 - \frac{323}{360}H & \frac{11}{30}H & -\frac{53}{360}H & \frac{19}{720}H & 0 \\ -\frac{62}{45}H & 1 - \frac{4}{15}H & -\frac{2}{45}H & \frac{1}{90}H & 0 \\ -\frac{51}{40}H & -\frac{9}{10}H & 1 - \frac{21}{40}H & \frac{3}{80}H & 0 \\ -\frac{64}{45}H & -\frac{8}{15}H & -\frac{64}{45}H & 1 - \frac{14}{45}H & 0 \\ \frac{25}{72}H & -\frac{25}{6}H & \frac{175}{72}H & -\frac{425}{144}H & 1 \end{pmatrix} \text{ and } \underline{B} = \begin{pmatrix} 1 + \frac{251}{720}H \\ 1 + \frac{29}{90}H \\ 1 + \frac{27}{80}H \\ 1 + \frac{14}{45}H \\ 1 + \frac{95}{144}H \end{pmatrix}$$

Now, let us define $q(H)$, $q_k(H)$ by

$$q(H) = \det(A) \quad \text{and} \quad q_k(H) = \det(A_k)$$

where A_r is the matrix A with the r^{th} column replaced by the vector B .

Using the above definitions and Cramer's rule in (3.2.2), we find that

$$y_{n+r} = \frac{q_r(H)}{q(H)} y_n, \quad r = 1, 2, \dots, k;$$

where $q(H)$ and $q_r(H)$ are polynomials with degree k and it is necessary that A to be invertible for all H .

For $n = mk$, $m = 0, 1, 2, \dots$,
we have

$$y_{n+r} = \left[\frac{q_r(H)}{q(H)} \right] \left[\frac{q_k(H)}{q(H)} \right]^m, \quad r = 1, 2, \dots, k.$$

Hence, the block method is said to be A-stable if and only if

$$\left| \frac{q_k(H)}{q(H)} \right| < 1 \text{ for all } H, \operatorname{Re}(H) < 0.$$

Therefore, in the case when $k = 5$, $q(H)$ for the above scheme (3.4.1) is given by

$$q(H) = \begin{vmatrix} 1 - \frac{323}{360}H & \frac{11}{30}H & -\frac{53}{360}H & \frac{19}{720}H & 0 \\ -\frac{62}{45}H & 1 - \frac{4}{15}H & -\frac{2}{45}H & \frac{1}{90}H & 0 \\ -\frac{51}{40}H & -\frac{9}{10}H & 1 - \frac{21}{40}H & \frac{3}{80}H & 0 \\ -\frac{64}{45}H & -\frac{8}{15}H & -\frac{64}{45}H & 1 - \frac{14}{45}H & 0 \\ \frac{25}{72}H & -\frac{25}{6}H & \frac{175}{72}H & -\frac{425}{144}H & 1 \end{vmatrix}$$

$$q(H) = \begin{vmatrix} 1 - \frac{323}{360}H & \frac{11}{30}H & -\frac{53}{360}H & \frac{19}{720}H \\ -\frac{62}{45}H & 1 - \frac{4}{15}H & -\frac{2}{45}H & \frac{1}{90}H \\ -\frac{51}{40}H & -\frac{9}{10}H & 1 - \frac{21}{40}H & \frac{3}{80}H \\ -\frac{64}{45}H & -\frac{8}{15}H & -\frac{64}{45}H & 1 - \frac{14}{45}H \end{vmatrix}$$

The four cofactors for $q(H)$ are:

$$\begin{aligned} q^{(1)}(H) &= \left(1 - \frac{323}{360}H \right) \begin{vmatrix} 1 - \frac{4}{15}H & -\frac{2}{45}H & \frac{1}{90}H \\ -\frac{9}{10}H & 1 - \frac{21}{40}H & \frac{3}{80}H \\ -\frac{8}{15}H & -\frac{64}{45}H & 1 - \frac{14}{45}H \end{vmatrix} \\ &= \left(1 - \frac{323}{360}H \right) \left(1 - \frac{397}{360}H + \frac{73}{180}H^2 - \frac{1}{30}H^3 \right) \\ &= 1 - 2H + \left(\frac{73}{180} + \frac{323}{360} \cdot \frac{397}{360} \right) H^2 - \left(\frac{1}{30} + \frac{323}{360} \cdot \frac{73}{180} \right) H^3 \\ &\quad + \frac{323}{10800} H^4 \end{aligned}$$

$$\begin{aligned} q^{(2)}(H) &= \left(-\frac{11}{30}H \right) \begin{vmatrix} -\frac{62}{45}H & -\frac{2}{45}H & \frac{1}{90}H \\ -\frac{51}{40}H & 1 - \frac{21}{40}H & \frac{3}{80}H \\ -\frac{64}{45}H & -\frac{64}{45}H & 1 - \frac{14}{45}H \end{vmatrix} \\ &= \left(-\frac{11}{30}H \right) \left(-\frac{62}{45}H + \frac{10}{9}H^2 - \frac{4}{15}H^3 \right) \\ &= \frac{341}{675}H^2 - \frac{11}{27}H^3 + \frac{22}{225}H^4 \end{aligned}$$

$$\begin{aligned} q^{(3)}(H) &= \left(-\frac{53}{360}H \right) \begin{vmatrix} -\frac{62}{45}H & 1 - \frac{4}{15}H & \frac{1}{90}H \\ -\frac{51}{40}H & -\frac{9}{10}H & \frac{3}{80}H \\ -\frac{64}{45}H & -\frac{8}{15}H & 1 - \frac{14}{45}H \end{vmatrix} \\ &= \left(-\frac{53}{360}H \right) \left(\frac{9}{20}H^2 - \frac{3}{10}H^3 + \frac{51}{40}H \right) \\ &= -\frac{901}{4800}H^2 - \frac{53}{800}H^3 + \frac{53}{1200}H^4 \end{aligned}$$

$$\begin{aligned}
q^{(4)}(H) &= \left(-\frac{19}{720}H \right) \begin{vmatrix} -\frac{62}{45}H & 1 - \frac{4}{15}H & -\frac{2}{45}H \\ -\frac{51}{40}H & -\frac{9}{10}H & 1 - \frac{21}{40}H \\ -\frac{64}{45}H & -\frac{8}{15}H & -\frac{64}{45}H \end{vmatrix} \\
&= \left(-\frac{19}{720}H \right) \left(-\frac{16}{15}H^3 - \frac{64}{45}H^2 - \frac{64}{45}H \right) \\
&= \frac{76}{2025}H^2 + \frac{76}{2025}H^3 + \frac{19}{675}H^4
\end{aligned}$$

Also the $q_5(H)$ for the above scheme (3.4.1) is given by

$$q_5(H) = \begin{vmatrix} 1 - \frac{323}{360}H & \frac{11}{30}H & -\frac{53}{360}H & \frac{19}{720}H & 1 + \frac{251}{720}H \\ -\frac{62}{45}H & 1 - \frac{4}{15}H & -\frac{2}{45}H & \frac{1}{90}H & 1 + \frac{29}{90}H \\ -\frac{51}{40}H & -\frac{9}{10}H & 1 - \frac{21}{40}H & \frac{3}{80}H & 1 + \frac{27}{80}H \\ -\frac{64}{45}H & -\frac{8}{15}H & -\frac{64}{45}H & 1 - \frac{14}{45}H & 1 + \frac{14}{45}H \\ \frac{25}{72}H & -\frac{25}{6}H & \frac{175}{72}H & -\frac{425}{144}H & 1 + \frac{95}{144}H \end{vmatrix}$$

The five cofactors for $q_5(H)$ are:

$$\begin{aligned}
q_5^{(1)}(H) &= \left(1 - \frac{323}{360}H \right) \begin{vmatrix} 1 - \frac{4}{15}H & -\frac{2}{45}H & \frac{1}{90}H & 1 + \frac{29}{90}H \\ -\frac{9}{10}H & 1 - \frac{21}{40}H & \frac{3}{80}H & 1 + \frac{27}{80}H \\ -\frac{8}{15}H & -\frac{64}{45}H & 1 - \frac{14}{45}H & 1 + \frac{14}{45}H \\ -\frac{25}{6}H & \frac{175}{72}H & -\frac{425}{144}H & 1 + \frac{95}{144}H \end{vmatrix} \\
&= \left(1 - \frac{323}{360}H \right) \left(1 + \frac{191}{45}H + \frac{31}{60}H^2 + \frac{659}{180}H^3 + \frac{13}{12}H^4 \right) \\
&= 1 + \frac{241}{72}H + \left(\frac{31}{60} - \frac{323}{360} \cdot \frac{191}{45} \right)H^2 + \left(\frac{659}{180} - \frac{323}{360} \cdot \frac{31}{60} \right)H^3 \\
&\quad + \left(\frac{13}{12} - \frac{323}{360} \cdot \frac{659}{180} \right)H^4 - \frac{4199}{4320}H^5
\end{aligned}$$

$$\begin{aligned}
q_5^{(2)}(H) &= \left(-\frac{11}{30}H \right) \begin{vmatrix} -\frac{62}{45}H & -\frac{2}{45}H & \frac{1}{90}H & 1 + \frac{29}{90}H \\ -\frac{51}{40}H & 1 - \frac{21}{40}H & \frac{3}{80}H & 1 + \frac{27}{80}H \\ -\frac{64}{45}H & -\frac{64}{45}H & 1 - \frac{14}{45}H & 1 + \frac{14}{45}H \\ \frac{25}{72}H & \frac{175}{72}H & -\frac{425}{144}H & 1 + \frac{95}{144}H \end{vmatrix} \\
&= \left(-\frac{11}{30}H \right) \left(-\frac{69}{40}H + \frac{3}{4}H^2 + \frac{3}{20}H^3 - \frac{1}{2}H^4 \right) \\
&= \frac{253}{400}H^2 - \frac{11}{40}H^3 - \frac{11}{200}H^4 + \frac{11}{60}H^5
\end{aligned}$$

$$\begin{aligned}
q_5^{(3)}(H) &= \left(-\frac{53}{360}H \right) \begin{vmatrix} -\frac{62}{45}H & 1 - \frac{4}{15}H & \frac{1}{90}H & 1 + \frac{29}{90}H \\ -\frac{51}{40}H & -\frac{9}{10}H & \frac{3}{80}H & 1 + \frac{27}{80}H \\ -\frac{64}{45}H & -\frac{8}{15}H & 1 - \frac{14}{45}H & 1 + \frac{14}{45}H \\ \frac{25}{72}H & -\frac{25}{6}H & -\frac{425}{144}H & 1 + \frac{95}{144}H \end{vmatrix} \\
&= \left(-\frac{53}{360}H \right) \left(\frac{29}{45}H^2 - \frac{11}{20}H^3 - \frac{1}{4}H^4 + \frac{73}{45}H \right) \\
&= -\frac{3869}{16200}H^2 - \frac{1537}{16200}H^3 + \frac{583}{7200}H^4 + \frac{53}{1440}H^5
\end{aligned}$$

$$\begin{aligned}
q_5^{(4)}(H) &= \left(-\frac{19}{720}H \right) \begin{vmatrix} -\frac{62}{45}H & 1 - \frac{4}{15}H & -\frac{2}{45}H & 1 + \frac{29}{90}H \\ -\frac{51}{40}H & -\frac{9}{10}H & 1 - \frac{21}{40}H & 1 + \frac{27}{80}H \\ -\frac{64}{45}H & -\frac{8}{15}H & -\frac{64}{45}H & 1 + \frac{14}{45}H \\ \frac{25}{72}H & -\frac{25}{6}H & \frac{175}{72}H & 1 + \frac{95}{144}H \end{vmatrix} \\
&= \left(-\frac{19}{720}H \right) \left(-\frac{68}{45}H^3 - \frac{1}{3}H^4 - \frac{27}{10}H^2 - \frac{637}{360}H \right) \\
&= \left(\frac{19}{720} \cdot \frac{637}{360} \right) H^2 + \frac{57}{800}H^3 + \frac{323}{8100}H^4 + \frac{19}{2160}H^5
\end{aligned}$$

$$\begin{aligned}
q_5^{(5)}(H) &= \left(1 + \frac{251}{720} H \right) \begin{vmatrix} -\frac{62}{45}H & 1 - \frac{4}{15}H & -\frac{2}{45}H & \frac{1}{90}H \\ -\frac{51}{40}H & -\frac{9}{10}H & 1 - \frac{21}{40}H & \frac{3}{80}H \\ -\frac{64}{45}H & -\frac{8}{15}H & -\frac{64}{45}H & 1 - \frac{14}{45}H \\ \frac{25}{72}H & -\frac{25}{6}H & \frac{175}{72}H & -\frac{425}{144}H \end{vmatrix} \\
&= \left(1 + \frac{251}{720} H \right) \left(5H^4 - \frac{5}{3}H^3 + \frac{65}{9}H^2 - \frac{25}{72}H \right) \\
&= -\frac{25}{72}H + \left(\frac{65}{9} - \frac{1255}{10368} \right)H^2 + \frac{1103}{1296}H^3 + \frac{1909}{432}H^4 \\
&\quad + \frac{251}{144}H^5
\end{aligned}$$

Therefore, the polynomials $q(H)$ and $q_{(5)}(H)$ are

$$\begin{aligned}
q(H) &= 1 - 2H + \frac{7}{4}H^2 - \frac{5}{6}H^3 + \frac{1}{5}H^4 \\
q_5(H) &= 1 + 3H + \frac{17}{4}H^2 + \frac{15}{4}H^3 + \frac{137}{60}H^4 + H^5
\end{aligned}$$

Hence, the amplification factor is

$$\left| \frac{q_5(H)}{q(H)} \right| = \left| \frac{1 + 3H + \frac{17}{4}H^2 + \frac{15}{4}H^3 + \frac{137}{60}H^4 + H^5}{1 - 2H + \frac{7}{4}H^2 - \frac{5}{6}H^3 + \frac{1}{5}H^4} \right|$$

which is less than one for all $H \in (-2.32688, 0)$. So, the method is A-stable for all $H \in (-2.32688, 0)$. Then the method is A-stable when $k = 5$.

Similarly, apply the system (3.2.4) when $k = 10$ to the test equation, we get

$$\begin{pmatrix} 1 - \frac{323}{360}H & \frac{11}{30}H & -\frac{53}{360}H & \frac{19}{720}H & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{62}{45}H & 1 - \frac{4}{15}H & -\frac{2}{45}H & \frac{1}{90}H & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{51}{40}H & -\frac{9}{10}H & 1 - \frac{21}{40}H & \frac{3}{80}H & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{64}{45}H & -\frac{8}{15}H & -\frac{64}{45}H & 1 - \frac{14}{45}H & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{25}{72}H & -\frac{25}{6}H & \frac{175}{72}H & -\frac{425}{144}H & 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{66}{5}H & -\frac{144}{5}H & \frac{126}{5}H & -\frac{123}{10}H & 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{20629}{360}H & -\frac{3283}{30}H & \frac{33859}{360}H & -\frac{26117}{720}H & 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{7552}{45}H & -\frac{4576}{15}H & \frac{11392}{45}H & -\frac{3928}{45}H & 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{15457}{40}H & -\frac{7047}{10}H & \frac{22707}{40}H & -\frac{14661}{80}H & 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{7450}{9}H & -\frac{4300}{3}H & \frac{10150}{9}H & -\frac{6275}{18}H & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} y_{n+1} \\ y_{n+2} \\ y_{n+3} \\ y_{n+4} \\ y_{n+5} \\ y_{n+6} \\ y_{n+7} \\ y_{n+8} \\ y_{n+9} \\ y_{n+10} \end{pmatrix} = \begin{pmatrix} 1 + \frac{251}{720}H \\ 1 + \frac{29}{90}H \\ 1 + \frac{27}{80}H \\ 1 + \frac{14}{45}H \\ 1 + \frac{95}{144}H \\ 1 + \frac{33}{10}H \\ 1 + \frac{9107}{720}H \\ 1 + \frac{1648}{45}H \\ 1 + \frac{7011}{80}H \\ 1 + \frac{3305}{18}H \end{pmatrix} y_n \quad (3.4.2)$$

Therefore, $q(H)$ is given by

$$q(H) = \begin{vmatrix} 1 - \frac{323}{360}H & \frac{11}{30}H & -\frac{53}{360}H & \frac{19}{720}H & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{62}{45}H & 1 - \frac{4}{15}H & -\frac{2}{45}H & \frac{1}{90}H & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{51}{40}H & -\frac{9}{10}H & 1 - \frac{21}{40}H & \frac{3}{80}H & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{64}{45}H & -\frac{8}{15}H & -\frac{64}{45}H & 1 - \frac{14}{45}H & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{25}{72}H & -\frac{25}{6}H & \frac{175}{72}H & -\frac{425}{144}H & 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{66}{5}H & -\frac{144}{5}H & \frac{126}{5}H & -\frac{123}{10}H & 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{20629}{360}H & -\frac{3283}{30}H & \frac{33859}{360}H & -\frac{26117}{720}H & 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{7552}{45}H & -\frac{4576}{15}H & \frac{11392}{45}H & -\frac{3928}{45}H & 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{15457}{40}H & -\frac{7047}{10}H & \frac{22707}{40}H & -\frac{14661}{80}H & 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{7450}{9}H & -\frac{4300}{3}H & \frac{10150}{9}H & -\frac{6275}{18}H & 0 & 0 & 0 & 0 & 0 & 1 \end{vmatrix}$$

$$q(H) = \begin{vmatrix} 1 - \frac{323}{360}H & \frac{11}{30}H & -\frac{53}{360}H & \frac{19}{720}H \\ -\frac{62}{45}H & 1 - \frac{4}{15}H & -\frac{2}{45}H & \frac{1}{90}H \\ -\frac{51}{40}H & -\frac{9}{10}H & 1 - \frac{21}{40}H & \frac{3}{80}H \\ -\frac{64}{45}H & -\frac{8}{15}H & -\frac{64}{45}H & 1 - \frac{14}{45}H \end{vmatrix}$$

$$\begin{aligned}
q(H) = & (1 - \frac{323}{360}H)(1 - \frac{397}{360}H + \frac{73}{180}H^2 - \frac{1}{30}H^3) \\
& + (-\frac{11}{30}H)(-\frac{62}{45}H + \frac{10}{9}H^2 - \frac{4}{15}H^3) \\
& + (-\frac{53}{360}H)(\frac{9}{20}H^2 - \frac{3}{10}H^3 + \frac{51}{40}H) \\
& + (-\frac{19}{720}H)(-\frac{16}{15}H^3 - \frac{64}{45}H^2 - \frac{64}{45}H)
\end{aligned}$$

$$q(H) = 1 - 2H + \frac{7}{4}H^2 - \frac{5}{6}H^3 + \frac{1}{5}H^4$$

Also, $q_{10}(H)$ is given by

$$\begin{aligned}
q_{10}(H) = & \begin{vmatrix} 1 - \frac{323}{360}H & \frac{11}{30}H & -\frac{53}{360}H & \frac{19}{720}H & 0 & 0 & 0 & 0 & 0 & 1 + \frac{251}{720}H \\ -\frac{62}{45}H & 1 - \frac{4}{15}H & -\frac{2}{45}H & \frac{1}{90}H & 0 & 0 & 0 & 0 & 0 & 1 + \frac{29}{90}H \\ -\frac{51}{40}H & -\frac{9}{10}H & 1 - \frac{21}{40}H & \frac{3}{80}H & 0 & 0 & 0 & 0 & 0 & 1 + \frac{27}{80}H \\ -\frac{64}{45}H & -\frac{8}{15}H & -\frac{64}{45}H & 1 - \frac{14}{45}H & 0 & 0 & 0 & 0 & 0 & 1 + \frac{14}{45}H \\ \frac{25}{72}H & -\frac{25}{6}H & \frac{175}{72}H & -\frac{425}{144}H & 1 & 0 & 0 & 0 & 0 & 1 + \frac{95}{144}H \\ \frac{66}{5}H & -\frac{144}{5}H & \frac{126}{5}H & -\frac{123}{10}H & 0 & 1 & 0 & 0 & 0 & 1 + \frac{33}{10}H \\ \frac{20629}{360}H & -\frac{3283}{30}H & \frac{33859}{360}H & -\frac{26117}{720}H & 0 & 0 & 1 & 0 & 0 & 1 + \frac{9107}{720}H \\ \frac{7552}{45}H & -\frac{4576}{15}H & \frac{11392}{45}H & -\frac{3928}{45}H & 0 & 0 & 0 & 1 & 0 & 1 + \frac{1648}{45}H \\ \frac{15457}{40}H & -\frac{7047}{10}H & \frac{22707}{40}H & -\frac{14661}{80}H & 0 & 0 & 0 & 0 & 1 & 1 + \frac{7011}{80}H \\ \frac{7450}{9}H & -\frac{4300}{3}H & \frac{10150}{9}H & -\frac{6275}{18}H & 0 & 0 & 0 & 0 & 0 & 1 + \frac{3305}{18}H \end{vmatrix} \\
q_{10}(H) = & \begin{vmatrix} 1 - \frac{323}{360}H & \frac{11}{30}H & -\frac{53}{360}H & \frac{19}{720}H & 1 + \frac{251}{720}H \\ -\frac{62}{45}H & 1 - \frac{4}{15}H & -\frac{2}{45}H & \frac{1}{90}H & 1 + \frac{29}{90}H \\ -\frac{51}{40}H & -\frac{9}{10}H & 1 - \frac{21}{40}H & \frac{3}{80}H & 1 + \frac{27}{80}H \\ -\frac{64}{45}H & -\frac{8}{15}H & -\frac{64}{45}H & 1 - \frac{14}{45}H & 1 + \frac{14}{45}H \\ \frac{7450}{9}H & -\frac{4300}{3}H & \frac{10150}{9}H & -\frac{6275}{18}H & 1 + \frac{3305}{18}H \end{vmatrix}
\end{aligned}$$

The five cofactors for $q_{10}(H)$ are:

$$\begin{aligned}
q_{10}^{(1)}(H) &= \left(1 - \frac{323}{360} H \right) \begin{vmatrix} 1 - \frac{4}{15} H & -\frac{2}{45} H & \frac{1}{90} H & 1 + \frac{29}{90} H \\ -\frac{9}{10} H & 1 - \frac{21}{40} H & \frac{3}{80} H & 1 + \frac{27}{80} H \\ -\frac{8}{15} H & -\frac{64}{45} H & 1 - \frac{14}{45} H & 1 + \frac{14}{45} H \\ -\frac{4300}{3} H & \frac{10150}{9} H & -\frac{6275}{18} H & 1 + \frac{3305}{18} H \end{vmatrix} \\
&= \left(1 - \frac{323}{360} H \right) \left(1 + \frac{33467}{40} H - \frac{5392}{5} H^2 + \frac{31433}{60} H^3 + 238 H^4 \right) \\
&= 1 + \frac{7522}{9} H - \left(\frac{5392}{5} + \frac{323}{360} \cdot \frac{33467}{40} \right) H^2 + \left(\frac{31433}{60} + \frac{323}{360} \cdot \frac{5392}{5} \right) H^3 \\
&\quad + \left(238 - \frac{323}{360} \cdot \frac{31433}{60} \right) H^4 - \frac{38437}{180} H^5
\end{aligned}$$

$$\begin{aligned}
q_{10}^{(2)}(H) &= \left(-\frac{11}{30} H \right) \begin{vmatrix} -\frac{62}{45} H & -\frac{2}{45} H & \frac{1}{90} H & 1 + \frac{29}{90} H \\ -\frac{51}{40} H & 1 - \frac{21}{40} H & \frac{3}{80} H & 1 + \frac{27}{80} H \\ -\frac{64}{45} H & -\frac{64}{45} H & 1 - \frac{14}{45} H & 1 + \frac{14}{45} H \\ \frac{7450}{9} H & \frac{10150}{9} H & -\frac{6275}{18} H & 1 + \frac{3305}{18} H \end{vmatrix} \\
&= \left(-\frac{11}{30} H \right) \left(-\frac{37312}{45} H + \frac{832}{3} H^2 + \frac{10688}{45} H^3 - \frac{448}{3} H^4 \right) \\
&= \frac{205216}{675} H^2 - \frac{4576}{45} H^3 - \frac{58784}{675} H^4 + \frac{2464}{45} H^5
\end{aligned}$$

$$\begin{aligned}
q_{10}^{(3)}(H) &= \left(-\frac{53}{360} H \right) \begin{vmatrix} -\frac{62}{45} H & 1 - \frac{4}{15} H & \frac{1}{90} H & 1 + \frac{29}{90} H \\ -\frac{51}{40} H & -\frac{9}{10} H & \frac{3}{80} H & 1 + \frac{27}{80} H \\ -\frac{64}{45} H & -\frac{8}{15} H & 1 - \frac{14}{45} H & 1 + \frac{14}{45} H \\ \frac{7450}{9} H & -\frac{4300}{3} H & -\frac{6275}{18} H & 1 + \frac{3305}{18} H \end{vmatrix} \\
&= \left(-\frac{53}{360} H \right) \left(\frac{24794}{45} H^2 - \frac{6223}{60} H^3 - 98 H^4 + \frac{298459}{360} H \right) \\
&= - \left(\frac{53}{360} \cdot \frac{298459}{360} \right) H^2 - \left(\frac{53}{360} \cdot \frac{24794}{45} \right) H^3 \\
&\quad + \left(\frac{53}{360} \cdot \frac{6223}{60} \right) H^4 + \frac{2597}{180} H^5
\end{aligned}$$

$$\begin{aligned}
q_{10}^{(4)}(H) &= \left(-\frac{19}{720}H \right) \begin{vmatrix} -\frac{62}{45}H & 1 - \frac{4}{15}H & -\frac{2}{45}H & 1 + \frac{29}{90}H \\ -\frac{51}{40}H & -\frac{9}{10}H & 1 - \frac{21}{40}H & 1 + \frac{27}{80}H \\ -\frac{64}{45}H & -\frac{8}{15}H & -\frac{64}{45}H & 1 + \frac{14}{45}H \\ \frac{7450}{9}H & -\frac{4300}{3}H & \frac{10150}{9}H & 1 + \frac{3305}{18}H \end{vmatrix} \\
&= \left(-\frac{19}{720}H \right) \left(-\frac{4332}{5}H^3 - 224H^4 - \frac{6906}{5}H^2 - \frac{4146}{5}H \right) \\
&= \frac{13129}{600}H^2 + \frac{21869}{600}H^3 + \frac{6859}{300}H^4 + \frac{266}{45}H^5
\end{aligned}$$

$$\begin{aligned}
q_{10}^{(5)}(H) &= \left(1 + \frac{251}{720}H \right) \begin{vmatrix} -\frac{62}{45}H & 1 - \frac{4}{15}H & -\frac{2}{45}H & \frac{1}{90}H \\ -\frac{51}{40}H & -\frac{9}{10}H & 1 - \frac{21}{40}H & \frac{3}{80}H \\ -\frac{64}{45}H & -\frac{8}{15}H & -\frac{64}{45}H & 1 - \frac{14}{45}H \\ \frac{7450}{9}H & -\frac{4300}{3}H & \frac{10150}{9}H & -\frac{6275}{18}H \end{vmatrix} \\
&= \left(1 + \frac{251}{720}H \right) \left(1120H^4 - 1940H^3 + \frac{17510}{9}H^2 - \frac{7450}{9}H \right) \\
&= -\frac{7450}{9}H + \left(\frac{17510}{9} - \frac{251}{720} \cdot \frac{7450}{9} \right) H^2 \\
&\quad - \left(1940 - \frac{251}{720} \cdot \frac{17510}{9} \right) H^3 + \frac{15973}{36}H^4 + \frac{3514}{9}H^5
\end{aligned}$$

Therefore,

$$q_{10}(H) = 1 + 8H + \frac{127}{4}H^2 + \frac{250}{3}H^3 + \frac{1627}{10}H^4 + 252H^5$$

Hence, the amplification factor is

$$\left| \frac{q_{10}(H)}{q(H)} \right| = \left| \frac{1 + 8H + \frac{127}{4}H^2 + \frac{250}{3}H^3 + \frac{1627}{10}H^4 + 252H^5}{1 - 2H + \frac{7}{4}H^2 - \frac{5}{6}H^3 + \frac{1}{5}H^4} \right| < 1$$

for all $H \in (-0.47767, 0)$.

So, the method is A-stable for all $H \in (-0.47767, 0)$, which means that the method is A-stable when $k = 10$. In general, as the block length increases, the stability bounds decrease. Note that the stability analysis depends only on the Corrector formula.

CHAPTER 4: NUMERICAL RESULTS

- 4.1 Description of the algorithm
- 4.2 Test problems
- 4.3 Computational results
- 4.4 Comparison with a Runge-Kutta method of the same order

CHAPTER 4

NUMERICAL RESULTS

4.1 Description of the algorithm

In this chapter we present the experimental results of our block method which is mentioned in chapter 3. We shall first present the implementation of the block method, using the Euler's formula as a predictor and block method where the block length is equal five as a corrector formula. This method can be computed in three steps for each equidistant step point $r = 1, 2, 3, 4, 5$ and the mode $P(EC)^5$ which will be used is described as follows:

$$P: \quad y_{n+r,0} = y_n + rh \, y'_n$$

For $i = 1 \longrightarrow 5$

$$E: \quad y'_{n+r,s} = f(x_{n+r}, y_{n+r,s})$$

$$\begin{aligned} C: \quad y_{n+1,s+1} &= y_n + h \left(\frac{251}{720} y'_n + \frac{323}{360} y'_{n+1,s} - \frac{11}{30} y'_{n+2,s} + \frac{53}{360} y'_{n+3,s} - \frac{19}{720} y'_{n+4,s} \right) \\ y_{n+2,s+1} &= y_n + h \left(\frac{29}{90} y'_n + \frac{62}{45} y'_{n+1,s} + \frac{4}{15} y'_{n+2,s} + \frac{2}{45} y'_{n+3,s} - \frac{1}{90} y'_{n+4,s} \right) \\ y_{n+3,s+1} &= y_n + h \left(\frac{27}{80} y'_n + \frac{51}{40} y'_{n+1,s} + \frac{9}{10} y'_{n+2,s} + \frac{21}{40} y'_{n+3,s} - \frac{3}{80} y'_{n+4,s} \right) \\ y_{n+4,s+1} &= y_n + h \left(\frac{14}{45} y'_n + \frac{64}{45} y'_{n+1,s} + \frac{8}{15} y'_{n+2,s} + \frac{64}{45} y'_{n+3,s} + \frac{14}{45} y'_{n+4,s} \right) \\ y_{n+5,s+1} &= y_n + h \left(\frac{95}{144} y'_n - \frac{25}{72} y'_{n+1,s} + \frac{25}{6} y'_{n+2,s} - \frac{175}{72} y'_{n+3,s} + \frac{425}{144} y'_{n+4,s} \right) \end{aligned}$$

$s = 0, 1, 2, 3, 4.$

End loop,

where n initially takes the value zero, then increases by the value of 5 until the end of the number of steps.

First, the above algorithm computes the predicted values y_{n+1} ,

y_{n+2} , y_{n+3} , y_{n+4} and y_{n+5} . Note that the algorithm is implemented using computer language C++. Second, the algorithm computes the function evaluations, $f(x_{n+r}, y_{n+r})$, $r = 1, 2, 3, 4$. Finally, the algorithm computes the corrected values y_{n+1}^c , y_{n+2}^c , $\dots$, y_{n+5}^c . The algorithm is written in computer language C++ using the following test problems.

4.2 Test problems

In this section we present four select test problems given by:

Problem (i):

$$y' = y, \quad y(0) = 1, \quad 0 \leq x \leq 1$$

The analytic solution is $y = e^x$.

Problem (ii):

We use the following problem in “Two fortran packages for assessing initial value methods” by (Enright and Pryce, 1987),

$$y' = -y^3/2, \quad y(0) = 1, \quad 0 \leq x \leq 1$$

The analytic solution is $y = \frac{1}{\sqrt{x+1}}$.

Problem (iii):

$$y' = xy, \quad y(0) = 1, \quad 0 \leq x \leq 1$$

The analytic solution is $y = e^{x^2/2}$.

Problem (iv):

$$y' = x + y, \quad y(0) = 1, \quad 0 \leq x \leq 1$$

The analytic solution is $y = -x - 1 + 2e^x$.

Using the above first test problem with $h = 0.01$, the above algorithm is written in computer language C++ as follows:

```
#include<iostream>
#include<cmath>
#include<iomanip>
using namespace std;
int main ( )
{
double h=0.01, x[101], yp[101], yc[101], d[101];
x[0]=0;
yp[0]=1.0;
yc[0]=1.0;
d[0]=1.0;
cout<<"all valuable of x from 0 to 100"<<endl;
for (int i=0; i<101; i++)
{
x[i]=i*h;
cout<<"x"<<i<<"="<<x[i]<<endl;    // compute  $x_0, x_1, x_2, \dots, x_{100}$ 
}
for (int n=0; n<96; n=n+5)
{
for (int r=1; r<6; r++)
yp[n+r]=yc[n]+r*h*d[n];    // compute the predicted values
for (int j=1; j<=5; j++)
{
for (int i=1; i<6; i++)
```

```

{
if (j == 1) d[n+i]=yp[n+i];      // compute the function evaluation
else d[n+i]=yc[n+i];           // in case  $y' = y$ 
}
yc[n+1]=yc[n]+h*((251.0/720)*d[n]+(323.0/360)*d[n+1]
-(11.0/30)*d[n+2]+(53.0/360)*d[n+3]-(19.0/720)*d[n+4]);
yc[n+2]=yc[n]+h*((29.0/90)*d[n]+(62.0/45)*d[n+1]
+(4.0/15)*d[n+2]+(2.0/45)*d[n+3]-(1.0/90)*d[n+4]);
yc[n+3]=yc[n]+h*((27.0/80)*d[n]+(51.0/40)*d[n+1]
+(9.0/10)*d[n+2]+(21.0/40)*d[n+3]-(3.0/80)*d[n+4]);
yc[n+4]=yc[n]+h*((14.0/45)*d[n]+(64.0/45)*d[n+1]
+(8.0/15)*d[n+2]+(64.0/45)*d[n+3]+(14.0/45)*d[n+4]);
yc[n+5]=yc[n]+h*((95.0/144)*d[n]-(25.0/72)*d[n+1]
+(25.0/6)*d[n+2]-(175.0/72)*d[n+3]+(425.0/144)*d[n+4]);
// compute the corrected values
if (j == 5) d[n+1]=yc[n+1];
if (j == 5) d[n+2]=yc[n+2];
if (j == 5) d[n+3]=yc[n+3];
if (j == 5) d[n+4]=yc[n+4];
if (j == 5) d[n+5]=yc[n+5];
}
}
cout<<"all valuable of y from 0 to 100"<<endl;
for (int r=0; r<101; r++)
{
cout<<fixed<<showpoint<<setprecision(16);

```

```

cout<<"y"<<r<<"="<<yc[r]<<endl;    // compute  $y_0, y_1, y_2, \dots, y_{100}$ 
}
cout<<"The exact solution = 2.7182818284590451"<<endl;
cout<<"Error = "<<2.7182818284590451-yc[100]<<endl;
system("PAUSE");
return 0;
}

```

For the remaining programming of the other test problems with each step-size h (see appendices A – D).

4.3 Computational results

The following tables show the experimental results for the above test problems and the accuracy of the solution which is measured by computing the differences between the approximation solution and the exact solution at the end of the steps.

Table (4.3.1)
Block method results using test problem (i)

Step-size h	Computed solution	Accuracy
0.01	2.7182818284339207	2.51×10^{-11}
0.001	2.7182818284590464	1.30×10^{-15}
0.0001	2.7182818284590429	2.20×10^{-15}

The exact solution is $y = e^x$

The exact solution at $x = 1$:

$$y(1) = e^1 = 2.7182818284590451$$

Table (4.3.2)
Block method results using test problem (ii)

Step-size h	Computed solution	Accuracy
0.01	0.70710678109597458	9.1×10^{-11}
0.001	0.70710678118654646	1.0×10^{-15}
0.0001	0.70710678118654713	3.3×10^{-16}

The exact solution is $y = 1/\sqrt{x+1}$

The exact solution at $x = 1$:

$$y(1) = 1/\sqrt{2} = 0.70710678118654746$$

Table (4.3.3)
Block method results using test problem (iii)

Step-size h	Computed solution	Accuracy
0.01	1.6487212703448912	3.55×10^{-10}
0.001	1.6487212707001244	3.80×10^{-15}
0.0001	1.6487212707001289	7.00×10^{-16}

The exact solution is $y = e^{x^2/2}$

The exact solution at $x = 1$:

$$y(1) = e^{1/2} = 1.6487212707001282$$

Table (4.3.4)
Block method results using test problem (iv)

Step-size h	Computed solution	Accuracy
0.01	3.4365636568678393	5.0×10^{-11}
0.001	3.4365636569180902	4.0×10^{-16}
0.0001	3.4365636569180955	4.9×10^{-15}

The exact solution is $y = -x - 1 + 2e^x$

The exact solution at $x = 1$:

$$y(1) = -1 - 1 + 2e^1 = 3.4365636569180906$$

4.4 Comparison with a Runge-Kutta method of the same order

In this section we compare the experimental results which was obtained by our block method (with block length = 5) and the following fifth order Runge-Kutta method given by (Luther and Konen, 1965),

$$y_{n+1} = y_n + (23k_1 + 125k_2 - 81k_5 + 125k_6) / 192$$

$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf(x_n + h/3, y_n + k_1/3)$$

$$k_3 = hf(x_n + 2h/5, y_n + (4k_1 + 6k_2)/25)$$

$$k_4 = hf(x_n + h, y_n + (k_1 - 12k_2 + 15k_3)/4)$$

$$k_5 = hf(x_n + 2h/3, y_n + (6k_1 + 90k_2 - 50k_3 + 8k_4)/81)$$

$$k_6 = hf(x_n + 4h/5, y_n + (6k_1 + 36k_2 + 10k_3 + 8k_4)/75)$$

The results from Runge-Kutta method using the above four test problems obtained from the following C++ programme:

```
#include<iostream>
#include<cmath>
#include<iomanip>
using namespace std;
double F(double x, double y, int TestProblem) // Test differential equations
{
    if (TestProblem == 1)
        return y;
    if (TestProblem == 2)
        return -1*pow(y,3.0)/2;
    if (TestProblem == 3)
        return x*y;
    if (TestProblem == 4)
        return x+y;
}
int main( )
{
    double K1,K2,K3,K4,K5,K6;
    for (int i=1; i<=4; i++)      // i is the number of DE
    {
        cout<<"solution of equation "<<i<<" ="<<endl;
        for (int j=2; j<=4; j++)    // to manage the step size
        {
            double h=pow(0.1,j);
```

```

int n=0;
double x[10001], y[10001];
x[0]=0, y[0]=1;
for (n=0; n<1/h; n++)
{
K1=h*F(x[n],y[n],i);
K2=h*F(x[n]+h/3.0,y[n]+K1/3.0,i);
K3=h*F(x[n]+(2.0*h/5),y[n]+(4.0*K1+6.0*K2)/25,i);
K4=h*F(x[n]+h,y[n]+(K1-12.0*K2+15.0*K3)/4,i);
K5=h*F(x[n]+(2.0*h/3),y[n]+(6.0*K1+90.0*K2-50.0*K3+8.0*K4)/81,i);
K6=h*F(x[n]+(4.0*h/5),y[n]+(6.0*K1+36.0*K2+10.0*K3+8.0*K4)/75,i);
y[n+1]=y[n]+(23.0*K1+125.0*K2-81.0*K5+125.0*K6)/192;
x[n+1]=x[n]+h;
}
cout<<fixed<<showpoint<<setprecision(6);
cout<<y[n]<<" if h = "<<h<<endl;
}
cout<<endl;
}
system("PAUSE");
return 0;
}

```

Table (4.4.1)
Comparison between Block method and Runge-Kutta method
using test problem (i)

Step-size h	Accuracy	
	Fifth order block method	Runge-Kutta method
0.01	3.18×10^{-11}	1.18×10^{-3}
0.001	1.30×10^{-15}	1.18×10^{-4}
0.0001	2.20×10^{-15}	1.20×10^{-5}

Table (4.4.2)
Comparison between Block method and Runge-Kutta method
using test problem (ii)

Step-size h	Accuracy	
	Fifth order block method	Runge-Kutta method
0.01	8.9×10^{-11}	8×10^{-5}
0.001	1.0×10^{-15}	8×10^{-6}
0.0001	3.3×10^{-16}	1×10^{-6}

Table (4.4.3)
Comparison between Block method and Runge-Kutta method
using test problem (iii)

Step-size h	Accuracy	
	Fifth order block method	Runge-Kutta method
0.01	3.56×10^{-10}	9.54×10^{-4}
0.001	3.80×10^{-15}	9.50×10^{-5}
0.0001	7.00×10^{-16}	9.00×10^{-6}

Table (4.4.4)
Comparison between Block method and Runge-Kutta method
using test problem (iv)

Step-size h	Accuracy	
	Fifth order block method	Runge-Kutta method
0.01	6.4×10^{-11}	2.36×10^{-3}
0.001	4.0×10^{-16}	2.36×10^{-4}
0.0001	4.9×10^{-15}	2.40×10^{-5}

From the above tables we notice that the accuracy is satisfactory and much better than the 5th order Runge-Kutta method.

CHAPTER 5: DISCUSSION

5.1 Summary and conclusion

5.2 Future work

CHAPTER 5

DISCUSSION

This is the final chapter that summaries the main findings of the study. The chapter is divided into two sections, the first section presents the conclusion of the study and the the second section presents managerial recommendations and suggestions for future studies.

5.1 Summary and conclusion

The study is aimed to provide a new block method of higher order which can be implemented as a corrected formula of any block size together with Euler's method as a predicted formula. However, this research is concerned with solving initial value problems using block predictor-corrector method of 5th order. The study highlights the importance of solving initial value problems and their application in the real life problems.

Many physical situations can be modelled by initial value problems. There are a variety of traditional number of methods for solving these problems, nearly all of which follow the basic patter. Because of that practical importance, we only give a flavor here.

The method we have described in this thesis is a new method in which we are most interested throughout this research. The Euler method, the explicit, is used as predictor formula, and our new block method, implicit, is used as corrector formula. The best way of proceeding is to imply an explicit formula (method) to predict the values and the implicit

(method) to (refine) correct. It should be noted that the final evaluation in $P(EC)^5$ (predict, evaluate, correct) mode and the number of corrections can have a marked influence on the accuracy of the calculated values.

Also, it is worth while looking at the way Runge-Kutta method of 5th order is implemented in general-purpose software for the solution of IVPs. As the integration proceeds the exact error is computed. If the error is considered to be too large then the step-size is reduced.

Finally, the result indicates that the block method which was derived in the study could be considered as a good method on the basis of the accuracy and the comparison with a Runge-Kutta method of the same order. Also, we noticed that the accuracy is satisfactory and much better than the 5th order Runge-Kutta method. Thus, for small h we can say that the algorithm may give a better accuracy. Therefore, the results are accepted.

5.2 Future work

It is important to emphasis here that the previous literatures have explained that the block methods can be implemented in parallel computers.

The first need that has led to the development of parallel computers was the desire to be able to solve problems that require more data storage and take too long to solve on a sequential computers.

The second need that has led to the development of parallel computers is the requirement to be able to solve more and more problems in real time. If these computers are available in our university I am sure that our study will be a very powerful research.

This study focused on solving initial value problems, therefore further studies can solve IVPs using block predictor-corrector method on multiprocessor computers, which is consist of a number of processors each of which is capable of performing complex tasks independently of the others. Another motivating possibility for the future research, both, speed-up is given by $S_p = \frac{t_1}{t_p}$, where t_1 is the time for the serial algorithm on a single processor, and, t_p , to be the execution time for a parallel algorithm on p processors, and the efficiency is given by $E_p = \frac{S_p}{p}$ are measured.

BIBLIOGRAPHY

- Abbas, S. (1995). A fourth order parallel block predictor-corrector method for IVPs. *J. Kuwait University*, (22), No. 2, 5-11.
- Abbas, S. (1997). Derivation of new block methods for the numerical solution of first-order IVPs. *Intern. J. Computer Math.*, (64), 235-244.
- Abbas, S. & Delves, L.M. (1989). Parallel solution of ODEs by one-step block methods. *Report CSMR*, University of Liverpool.
- Birta, L.G. & Abou-Rabia, O. (1987). Parallel block predictor-corrector methods for ODEs. *IEEE Trans. Comput.*, (C-36), 299-311.
- Chu, M.T. & Hamilton, H. (1987). Parallel solution of ODEs by multi-block methods. *SIAM J. Sci. Stat. Comput.*, (8), 342-352.
- Enright, W.H. & Pryce, J.D. (1987). Two fortran packages for assessing initial value methods. *ACM Transcation on Mathematical Software*, (13), No. 1, 1-27.
- Franklin, M.A. (1978). Parallel solution of ordinary differential equations. *IEEE Trans. Computers*, (C-27), 413-420.
- Henrici, P. (1962). *Discrete variable methods in ordinary differential equations*. New York: John Wiley and Sons.
- Katz, I.N., Franklin, M.A. & Sen, A. (1977). Optimally stable parallel predictors for Adams-Moulton correctors. *Comp. and Maths. with Appls.*, (3), 217-233.
- Lambert, D. (1973). *Computational methods in ordinary differential equations*. New York: Wiley.

- Luther H.A. & Konen H.P. (1965). Some fifth-order classical runge-kutta formulas. *SIAM Review*, (7), No. 4, 551-558.
- Miranker, W.L. (1971). A survey of parallelism in numerical analysis. *SIAM Review*, (13), 524-547.
- Miranker, W.L. & Liniger, W. (1967). Parallel methods for the numerical integration of ordinary differential equations. *Math. Comp.*, (21), 303-320.
- Moody, T.C. & Hans, H. (1987). Parallel solution of ODEs by multi-block methods. *SIAM, J. Sci., STAT. Comput.*, (8), 342-353.
- Moulton, F.R. (1926). New methods in exterior ballistics. University of Chicago Press, Chicago.
- Mouney, G., Authie, G. & Gayraud, T. (1991). Parallel solution of ODEs. Implementation of block methods on transputer networks, *Contributed paper at ICIAM 91*, Washington D.C.
- Rosser, J.B. (1967). A Runge-Kutta for all seasons. *SIAM Review*, (9), 417-452.
- Shampine, L.F. & Watts, H.A. (1969). Block implicit one-step methods. *Math. Comp.*, (23), 731-740.
- Voss, D. & Abbas, S. (1997). Block predictor-corrector schemes for the parallel solution of ODEs. *Computers Math Applic.*, (33), No. 6, 65-72.
- Watts, H.A. & Shampine, L.F. (1972). A stable block implicit methods. *BIT*, (12), 252-266.
- Worland, P.B. (1976). Parallel methods for the numerical solution of ordinary differential equations. *IEEE Trans. Computers*, (C-25), 1045-1048.

APPENDIX A

COMPUTER LANGUAGE C++ FOR TEST PROBLEM (i)

This appendix is to show our algorithm (refer to section 4.1 in chapter 4) written in computer language C++ using the first test problem with $h = 0.001$ & $h = 0.0001$.

$$\bullet \quad y' = y, \quad y(0) = 1, \quad 0 \leq x \leq 1, \quad h = 0.001$$

```
#include<iostream>
```

```
#include<cmath>
```

```
#include<iomanip>
```

```
using namespace std;
```

```
int main ( )
```

```
{
```

```
double h=0.001, x[1001], yp[1001], yc[1001], d[1001];
```

```
x[0]=0;
```

```
yp[0]=1.0;
```

```
yc[0]=1.0;
```

```
d[0]=1.0;
```

```
cout<<"all valuable of x from 0 to 1000"<<endl;
```

```
for (int i=0; i<1001; i++)
```

```
{
```

```
x[i]=i*h;
```

```
cout<<"x"<<i<<"="<<x[i]<<endl;    // compute  $x_0, x_1, x_2, \dots, x_{1000}$ 
```

```
}
```

```

for (int n=0; n<996; n=n+5)
{
for (int r=1; r<6; r++)
yp[n+r]=yc[n]+r*h*d[n];      // compute the predicted values
for (int j=1; j<=5; j++)
{
for (int i=1; i<6; i++)
{
if (j == 1) d[n+i]=yp[n+i];      // compute the function evaluation
else d[n+i]=yc[n+i];            // in case  $y' = y$ 
}
yc[n+1]=yc[n]+h*((251.0/720)*d[n]+(323.0/360)*d[n+1]
-(11.0/30)*d[n+2]+(53.0/360)*d[n+3]-(19.0/720)*d[n+4]);
yc[n+2]=yc[n]+h*((29.0/90)*d[n]+(62.0/45)*d[n+1]
+(4.0/15)*d[n+2]+(2.0/45)*d[n+3]-(1.0/90)*d[n+4]);
yc[n+3]=yc[n]+h*((27.0/80)*d[n]+(51.0/40)*d[n+1]
+(9.0/10)*d[n+2]+(21.0/40)*d[n+3]-(3.0/80)*d[n+4]);
yc[n+4]=yc[n]+h*((14.0/45)*d[n]+(64.0/45)*d[n+1]
+(8.0/15)*d[n+2]+(64.0/45)*d[n+3]+(14.0/45)*d[n+4]);
yc[n+5]=yc[n]+h*((95.0/144)*d[n]-(25.0/72)*d[n+1]
+(25.0/6)*d[n+2]-(175.0/72)*d[n+3]+(425.0/144)*d[n+4]);
// compute the corrected values
if (j == 5) d[n+1]=yc[n+1];
if (j == 5) d[n+2]=yc[n+2];
if (j == 5) d[n+3]=yc[n+3];
if (j == 5) d[n+4]=yc[n+4];
}
}

```

```

    if (j == 5) d[n+5]=yc[n+5];
    }
}

cout<<"all valuable of y from 0 to 1000"<<endl;
for (int r=0; r<1001; r++)
{
    cout<<fixed<<showpoint<<setprecision(16);
    cout<<"y"<<r<<"="<<yc[r]<<endl;    // compute  $y_0, y_1, y_2, \dots, y_{1000}$ 
}
cout<<"The exact solution = 2.7182818284590451"<<endl;
cout<<"Error = "<<yc[1000]-2.7182818284590451<<endl;
system("PAUSE");
return 0;
}

```

- $y' = y, \quad y(0) = 1, \quad 0 \leq x \leq 1, \quad h = 0.0001$

```

#include<iostream>
#include<cmath>
#include<iomanip>
using namespace std;
int main ( )
{
    double h=0.0001, x[10001], yp[10001], yc[10001], d[10001];
    x[0]=0;
    yp[0]=1.0;
    yc[0]=1.0;
    d[0]=1.0;

```

```

cout<<"all valuable of x from 0 to 10000"<<endl;
for (int i=0; i<10001; i++)
{
x[i]=i*h;
cout<<"x"<<i<<"="<<x[i]<<endl;    // compute  $x_0, x_1, x_2, \dots, x_{10000}$ 
}
for (int n=0; n<9996; n=n+5)
{
for (int r=1; r<6; r++)
yp[n+r]=yc[n]+r*h*d[n];    // compute the predicted values
for (int j=1; j<=5; j++)
{
for (int i=1; i<6; i++)
{
if (j == 1) d[n+i]=yp[n+i];    // compute the function evaluation
else d[n+i]=yc[n+i];    // in case  $y' = y$ 
}
yc[n+1]=yc[n]+h*((251.0/720)*d[n]+(323.0/360)*d[n+1]
-(11.0/30)*d[n+2]+(53.0/360)*d[n+3]-(19.0/720)*d[n+4]);
yc[n+2]=yc[n]+h*((29.0/90)*d[n]+(62.0/45)*d[n+1]
+(4.0/15)*d[n+2]+(2.0/45)*d[n+3]-(1.0/90)*d[n+4]);
yc[n+3]=yc[n]+h*((27.0/80)*d[n]+(51.0/40)*d[n+1]
+(9.0/10)*d[n+2]+(21.0/40)*d[n+3]-(3.0/80)*d[n+4]);
yc[n+4]=yc[n]+h*((14.0/45)*d[n]+(64.0/45)*d[n+1]
+(8.0/15)*d[n+2]+(64.0/45)*d[n+3]+(14.0/45)*d[n+4]);
yc[n+5]=yc[n]+h*((95.0/144)*d[n]-(25.0/72)*d[n+1]

```

```

+(25.0/6)*d[n+2]-(175.0/72)*d[n+3]+(425.0/144)*d[n+4]);
// compute the corrected values
if (j == 5) d[n+1]=yc[n+1];
if (j == 5) d[n+2]=yc[n+2];
if (j == 5) d[n+3]=yc[n+3];
if (j == 5) d[n+4]=yc[n+4];
if (j == 5) d[n+5]=yc[n+5];
}
}
cout<<"all valuable of y from 0 to 10000"<<endl;
for (int r=0; r<10001; r++)
{
cout<<fixed<<showpoint<<setprecision(16);
cout<<"y"<<r<<"="<<yc[r]<<endl; // compute  $y_0, y_1, y_2, \dots, y_{10000}$ 
}
cout<<"The exact solution = 2.7182818284590451"<<endl;
cout<<"Error = "<<2.7182818284590451-yc[10000]<<endl;
system("PAUSE");
return 0;
}

```

APPENDIX B

COMPUTER LANGUAGE C++ FOR TEST PROBLEM (ii)

We present here the computer language C++ for our algorithm (refer to section 4.1 in chapter 4) using the second test problem with $h = 0.01$, $h = 0.001$ & $h = 0.0001$

$$\bullet \quad y' = -y^3/2, \quad y(0) = 1, \quad 0 \leq x \leq 1, \quad h = 0.01$$

```
#include<iostream>
```

```
#include<cmath>
```

```
#include<iomanip>
```

```
using namespace std;
```

```
int main ( )
```

```
{
```

```
double h=0.01, x[101], yp[101], yc[101], d[101];
```

```
x[0]=0;
```

```
yp[0]=1.0;
```

```
yc[0]=1.0;
```

```
d[0]=-1.0*(pow(yp[0],3)/2);
```

```
cout<<"all valuable of x from 0 to 100"<<endl;
```

```
for (int i=0; i<101; i++)
```

```
{
```

```
x[i]=i*h;
```

```
cout<<"x"<<i<<"="<<x[i]<<endl;    // compute  $x_0, x_1, x_2, \dots, x_{100}$ 
```

```
}
```

```

for (int n=0; n<96; n=n+5)
{
for (int r=1; r<6; r++)
yp[n+r]=yc[n]+r*h*d[n];    // compute the predicted values
for (int j=1; j<=5; j++)
{
for (int i=1; i<6; i++)
{
// compute the function
if (j == 1) d[n+i]=-1.0*(pow(yp[n+i],3)/2); // evaluation in case
else d[n+i]=-1.0*(pow(yc[n+i],3)/2);      //  $y' = -y^3/2$ 
}
yc[n+1]=yc[n]+h*((251.0/720)*d[n]+(323.0/360)*d[n+1]
-(11.0/30)*d[n+2]+(53.0/360)*d[n+3]-(19.0/720)*d[n+4]);
yc[n+2]=yc[n]+h*((29.0/90)*d[n]+(62.0/45)*d[n+1]
+(4.0/15)*d[n+2]+(2.0/45)*d[n+3]-(1.0/90)*d[n+4]);
yc[n+3]=yc[n]+h*((27.0/80)*d[n]+(51.0/40)*d[n+1]
+(9.0/10)*d[n+2]+(21.0/40)*d[n+3]-(3.0/80)*d[n+4]);
yc[n+4]=yc[n]+h*((14.0/45)*d[n]+(64.0/45)*d[n+1]
+(8.0/15)*d[n+2]+(64.0/45)*d[n+3]+(14.0/45)*d[n+4]);
yc[n+5]=yc[n]+h*((95.0/144)*d[n]-(25.0/72)*d[n+1]
+(25.0/6)*d[n+2]-(175.0/72)*d[n+3]+(425.0/144)*d[n+4]);
// compute the corrected values
if (j == 5) d[n+1]=-1.0*(pow(yc[n+1],3)/2);
if (j == 5) d[n+2]=-1.0*(pow(yc[n+2],3)/2);
if (j == 5) d[n+3]=-1.0*(pow(yc[n+3],3)/2);
if (j == 5) d[n+4]=-1.0*(pow(yc[n+4],3)/2);
}
}
}

```

```

if (j == 5) d[n+5]=-1.0*(pow(yc[n+5],3)/2);
}
}
cout<<"all valuable of y from 0 to 100"<<endl;
for (int r=0; r<101; r++)
{
cout<<fixed<<showpoint<<setprecision(17);
cout<<"y"<<r<<"="<<yc[r]<<endl;    // compute  $y_0, y_1, y_2, \dots, y_{100}$ 
}
cout<<"The exact solution = 0.70710678118654746"<<endl;
cout<<"Error = "<<0.70710678118654746-yc[100]<<endl;
system("PAUSE");
return 0;
}

```

- $y' = -y^3/2, \quad y(0) = 1, \quad 0 \leq x \leq 1, \quad h = 0.001$

```

#include<iostream>
#include<cmath>
#include<iomanip>
using namespace std;
int main ( )
{
double h=0.001, x[1001], yp[1001], yc[1001], d[1001];
x[0]=0;
yp[0]=1.0;
yc[0]=1.0;
d[0]=-1.0*(pow(yp[0],3)/2);

```

```

cout<<"all valuable of x from 0 to 1000"<<endl;
for (int i=0; i<1001; i++)
{
x[i]=i*h;
cout<<"x"<<i<<"="<<x[i]<<endl;    // compute  $x_0, x_1, x_2, \dots, x_{1000}$ 
}
for (int n=0; n<996; n=n+5)
{
for (int r=1; r<6; r++)
yp[n+r]=yc[n]+r*h*d[n];    // compute the predicted values
for (int j=1; j<=5; j++)
{
for (int i=1; i<6; i++)
{
// compute the function
if (j == 1) d[n+i]=-1.0*(pow(yp[n+i],3)/2); // evaluation in case
else d[n+i]=-1.0*(pow(yc[n+i],3)/2);      //  $y' = -y^3/2$ 
}
yc[n+1]=yc[n]+h*((251.0/720)*d[n]+(323.0/360)*d[n+1]
-(11.0/30)*d[n+2]+(53.0/360)*d[n+3]-(19.0/720)*d[n+4]);
yc[n+2]=yc[n]+h*((29.0/90)*d[n]+(62.0/45)*d[n+1]
+(4.0/15)*d[n+2]+(2.0/45)*d[n+3]-(1.0/90)*d[n+4]);
yc[n+3]=yc[n]+h*((27.0/80)*d[n]+(51.0/40)*d[n+1]
+(9.0/10)*d[n+2]+(21.0/40)*d[n+3]-(3.0/80)*d[n+4]);
yc[n+4]=yc[n]+h*((14.0/45)*d[n]+(64.0/45)*d[n+1]
+(8.0/15)*d[n+2]+(64.0/45)*d[n+3]+(14.0/45)*d[n+4]);
yc[n+5]=yc[n]+h*((95.0/144)*d[n]-(25.0/72)*d[n+1]

```

```

+(25.0/6)*d[n+2]-(175.0/72)*d[n+3]+(425.0/144)*d[n+4]);
// compute the corrected values
if (j == 5) d[n+1]=-1.0*(pow(yc[n+1],3)/2);
if (j == 5) d[n+2]=-1.0*(pow(yc[n+2],3)/2);
if (j == 5) d[n+3]=-1.0*(pow(yc[n+3],3)/2);
if (j == 5) d[n+4]=-1.0*(pow(yc[n+4],3)/2);
if (j == 5) d[n+5]=-1.0*(pow(yc[n+5],3)/2);
}
}
cout<<"all valuable of y from 0 to 1000"<<endl;
for (int r=0; r<1001; r++)
{
cout<<fixed<<showpoint<<setprecision(17);
cout<<"y"<<r<<"="<<yc[r]<<endl;  // compute y0,y1,y2,...,y1000
}
cout<<"The exact solution = 0.70710678118654746"<<endl;
cout<<"Error = "<<0.70710678118654746-yc[1000]<<endl;
system("PAUSE");
return 0;
}

```

• $y' = -y^3/2, \quad y(0) = 1, \quad 0 \leq x \leq 1, \quad h = 0.0001$

```

#include<iostream>
#include<cmath>
#include<iomanip>
using namespace std;
int main ( )

```

```

{
double h=0.0001, x[10001], yp[10001], yc[10001], d[10001];
x[0]=0;
yp[0]=1.0;
yc[0]=1.0;
d[0]=-1.0*(pow(yp[0],3)/2);
cout<<"all valuable of x from 0 to 10000"<<endl;
for (int i=0; i<10001; i++)
{
x[i]=i*h;
cout<<"x"<<i<<"="<<x[i]<<endl;    // compute  $x_0, x_1, x_2, \dots, x_{10000}$ 
}
for (int n=0; n<9996; n=n+5)
{
for (int r=1; r<6; r++)
yp[n+r]=yc[n]+r*h*d[n];    // compute the predicted values
for (int j=1; j<=5; j++)
{
for (int i=1; i<6; i++)
{
// compute the function
if (j == 1) d[n+i]=-1.0*(pow(yp[n+i],3)/2);    // evaluation in case
else d[n+i]=-1.0*(pow(yc[n+i],3)/2);    //  $y' = -y^3/2$ 
}
}
yc[n+1]=yc[n]+h*((251.0/720)*d[n]+(323.0/360)*d[n+1]
-(11.0/30)*d[n+2]+(53.0/360)*d[n+3]-(19.0/720)*d[n+4]);
yc[n+2]=yc[n]+h*((29.0/90)*d[n]+(62.0/45)*d[n+1]

```

```

+(4.0/15)*d[n+2]+(2.0/45)*d[n+3]-(1.0/90)*d[n+4]);
yc[n+3]=yc[n]+h*((27.0/80)*d[n]+(51.0/40)*d[n+1]
+(9.0/10)*d[n+2]+(21.0/40)*d[n+3]-(3.0/80)*d[n+4]);
yc[n+4]=yc[n]+h*((14.0/45)*d[n]+(64.0/45)*d[n+1]
+(8.0/15)*d[n+2]+(64.0/45)*d[n+3]+(14.0/45)*d[n+4]);
yc[n+5]=yc[n]+h*((95.0/144)*d[n]-(25.0/72)*d[n+1]
+(25.0/6)*d[n+2]-(175.0/72)*d[n+3]+(425.0/144)*d[n+4]);
// compute the corrected values
if (j == 5) d[n+1]=-1.0*(pow(yc[n+1],3)/2);
if (j == 5) d[n+2]=-1.0*(pow(yc[n+2],3)/2);
if (j == 5) d[n+3]=-1.0*(pow(yc[n+3],3)/2);
if (j == 5) d[n+4]=-1.0*(pow(yc[n+4],3)/2);
if (j == 5) d[n+5]=-1.0*(pow(yc[n+5],3)/2);
}
}
cout<<"all valuable of y from 0 to 10000"<<endl;
for (int r=0; r<10001; r++)
{
cout<<fixed<<showpoint<<setprecision(17);
cout<<"y"<<r<<"="<<yc[r]<<endl; // compute  $y_0, y_1, y_2, \dots, y_{10000}$ 
}
cout<<"The exact solution = 0.70710678118654746"<<endl;
cout<<"Error = "<<0.70710678118654746-yc[10000]<<endl;
system("PAUSE");
return 0;
}

```

APPENDIX C

COMPUTER LANGUAGE C++ FOR TEST PROBLEM (iii)

In this appendix we show our algorithm (refer to section 4.1 in chapter 4) written in computer language C++ using the third test problem with $h = 0.01$, $h = 0.001$ & $h = 0.0001$

$$\bullet \quad y' = xy, \quad y(0) = 1, \quad 0 \leq x \leq 1, \quad h = 0.01$$

```
#include<iostream>
#include<cmath>
#include<iomanip>
using namespace std;
int main ( )
{
double h=0.01, x[101], yp[101], yc[101], d[101];
x[0]=0;
yp[0]=1.0;
yc[0]=1.0;
d[0]=0.0;
cout<<"all valuable of x from 0 to 100"<<endl;
for (int i=0; i<101; i++)
{
x[i]=i*h;
cout<<"x"<<i<<"="<<x[i]<<endl;    // compute  $x_0, x_1, x_2, \dots, x_{100}$ 
}
```

```

for (int n=0; n<96; n=n+5)
{
for (int r=1; r<6; r++)
yp[n+r]=yc[n]+r*h*d[n];      // compute the predicted values
for (int j=1; j<=5; j++)
{
for (int i=1; i<6; i++)
{
if (j == 1) d[n+i]=x[n+i]*yp[n+i];      // compute the function
else d[n+i]=x[n+i]*yc[n+i];             // evaluation in case  $y' = xy$ 
}
yc[n+1]=yc[n]+h*((251.0/720)*d[n]+(323.0/360)*d[n+1]
-(11.0/30)*d[n+2]+(53.0/360)*d[n+3]-(19.0/720)*d[n+4]);
yc[n+2]=yc[n]+h*((29.0/90)*d[n]+(62.0/45)*d[n+1]
+(4.0/15)*d[n+2]+(2.0/45)*d[n+3]-(1.0/90)*d[n+4]);
yc[n+3]=yc[n]+h*((27.0/80)*d[n]+(51.0/40)*d[n+1]
+(9.0/10)*d[n+2]+(21.0/40)*d[n+3]-(3.0/80)*d[n+4]);
yc[n+4]=yc[n]+h*((14.0/45)*d[n]+(64.0/45)*d[n+1]
+(8.0/15)*d[n+2]+(64.0/45)*d[n+3]+(14.0/45)*d[n+4]);
yc[n+5]=yc[n]+h*((95.0/144)*d[n]-(25.0/72)*d[n+1]
+(25.0/6)*d[n+2]-(175.0/72)*d[n+3]+(425.0/144)*d[n+4]);
// compute the corrected values
if (j == 5) d[n+1]=x[n+1]*yc[n+1];
if (j == 5) d[n+2]=x[n+2]*yc[n+2];
if (j == 5) d[n+3]=x[n+3]*yc[n+3];
if (j == 5) d[n+4]=x[n+4]*yc[n+4];
}
}
}

```

```

if (j == 5) d[n+5]=x[n+5]*yc[n+5];
}
}
cout<<"all valuable of y from 0 to 100"<<endl;
for (int r=0; r<101; r++)
{
cout<<fixed<<showpoint<<setprecision(16);
cout<<"y"<<r<<"="<<yc[r]<<endl;    // compute  $y_0, y_1, y_2, \dots, y_{100}$ 
}
cout<<"The exact solution = 1.6487212707001282"<<endl;
cout<<"Error = "<<1.6487212707001282-yc[100]<<endl;
system("PAUSE");
return 0;
}

```

- $y' = xy, \quad y(0) = 1, \quad 0 \leq x \leq 1, \quad h = 0.001$

```

#include<iostream>
#include<cmath>
#include<iomanip>
using namespace std;
int main ( )
{
double h=0.001, x[1001], yp[1001], yc[1001], d[1001];
x[0]=0;
yp[0]=1.0;
yc[0]=1.0;
d[0]=0.0;

```

```

cout<<"all valuable of x from 0 to 1000"<<endl;
for (int i=0; i<1001; i++)
{
x[i]=i*h;
cout<<"x"<<i<<"="<<x[i]<<endl;    // compute  $x_0, x_1, x_2, \dots, x_{1000}$ 
}
for (int n=0; n<996; n=n+5)
{
for (int r=1; r<6; r++)
yp[n+r]=yc[n]+r*h*d[n];    // compute the predicted values
for (int j=1; j<=5; j++)
{
for (int i=1; i<6; i++)
{
if (j == 1) d[n+i]=x[n+i]*yp[n+i];    // compute the function
else d[n+i]=x[n+i]*yc[n+i];    // evaluation in case  $y' = xy$ 
}
yc[n+1]=yc[n]+h*((251.0/720)*d[n]+(323.0/360)*d[n+1]
-(11.0/30)*d[n+2]+(53.0/360)*d[n+3]-(19.0/720)*d[n+4]);
yc[n+2]=yc[n]+h*((29.0/90)*d[n]+(62.0/45)*d[n+1]
+(4.0/15)*d[n+2]+(2.0/45)*d[n+3]-(1.0/90)*d[n+4]);
yc[n+3]=yc[n]+h*((27.0/80)*d[n]+(51.0/40)*d[n+1]
+(9.0/10)*d[n+2]+(21.0/40)*d[n+3]-(3.0/80)*d[n+4]);
yc[n+4]=yc[n]+h*((14.0/45)*d[n]+(64.0/45)*d[n+1]
+(8.0/15)*d[n+2]+(64.0/45)*d[n+3]+(14.0/45)*d[n+4]);
yc[n+5]=yc[n]+h*((95.0/144)*d[n]-(25.0/72)*d[n+1]

```

```

+(25.0/6)*d[n+2]-(175.0/72)*d[n+3]+(425.0/144)*d[n+4]);
// compute the corrected values
if (j == 5) d[n+1]=x[n+1]*yc[n+1];
if (j == 5) d[n+2]=x[n+2]*yc[n+2];
if (j == 5) d[n+3]=x[n+3]*yc[n+3];
if (j == 5) d[n+4]=x[n+4]*yc[n+4];
if (j == 5) d[n+5]=x[n+5]*yc[n+5];
}
}
cout<<"all valuable of y from 0 to 1000"<<endl;
for (int r=0; r<1001; r++)
{
cout<<fixed<<showpoint<<setprecision(16);
cout<<"y"<<r<<"="<<yc[r]<<endl;  // compute  $y_0, y_1, y_2, \dots, y_{1000}$ 
}
cout<<"The exact solution = 1.6487212707001282"<<endl;
cout<<"Error = "<<1.6487212707001282-yc[1000]<<endl;
system("PAUSE");
return 0;
}

```

• $y' = xy, \quad y(0) = 1, \quad 0 \leq x \leq 1, \quad h = 0.0001$

```

#include<iostream>
#include<cmath>
#include<iomanip>
using namespace std;
int main ( )

```

```

{
double h=0.0001, x[10001], yp[10001], yc[10001], d[10001];
x[0]=0;
yp[0]=1.0;
yc[0]=1.0;
d[0]=0.0;
cout<<"all valuable of x from 0 to 10000"<<endl;
for (int i=0; i<10001; i++)
{
x[i]=i*h;
cout<<"x"<<i<<"="<<x[i]<<endl;    // compute  $x_0, x_1, x_2, \dots, x_{10000}$ 
}
for (int n=0; n<9996; n=n+5)
{
for (int r=1; r<6; r++)
yp[n+r]=yc[n]+r*h*d[n];    // compute the predicted values
for (int j=1; j<=5; j++)
{
for (int i=1; i<6; i++)
{
if (j == 1) d[n+i]=x[n+i]*yp[n+i];    // compute the function
else d[n+i]=x[n+i]*yc[n+i];    // evaluation in case  $y' = xy$ 
}
}
yc[n+1]=yc[n]+h*((251.0/720)*d[n]+(323.0/360)*d[n+1]
-(11.0/30)*d[n+2]+(53.0/360)*d[n+3]-(19.0/720)*d[n+4]);
yc[n+2]=yc[n]+h*((29.0/90)*d[n]+(62.0/45)*d[n+1]

```

```

+(4.0/15)*d[n+2]+(2.0/45)*d[n+3]-(1.0/90)*d[n+4]);
yc[n+3]=yc[n]+h*((27.0/80)*d[n]+(51.0/40)*d[n+1]
+(9.0/10)*d[n+2]+(21.0/40)*d[n+3]-(3.0/80)*d[n+4]);
yc[n+4]=yc[n]+h*((14.0/45)*d[n]+(64.0/45)*d[n+1]
+(8.0/15)*d[n+2]+(64.0/45)*d[n+3]+(14.0/45)*d[n+4]);
yc[n+5]=yc[n]+h*((95.0/144)*d[n]-(25.0/72)*d[n+1]
+(25.0/6)*d[n+2]-(175.0/72)*d[n+3]+(425.0/144)*d[n+4]);
// compute the corrected values
if (j == 5) d[n+1]=x[n+1]*yc[n+1];
if (j == 5) d[n+2]=x[n+2]*yc[n+2];
if (j == 5) d[n+3]=x[n+3]*yc[n+3];
if (j == 5) d[n+4]=x[n+4]*yc[n+4];
if (j == 5) d[n+5]=x[n+5]*yc[n+5];
}
}
cout<<"all valuable of y from 0 to 10000"<<endl;
for (int r=0; r<10001; r++)
{
cout<<fixed<<showpoint<<setprecision(16);
cout<<"y"<<r<<"="<<yc[r]<<endl; // compute  $y_0, y_1, y_2, \dots, y_{10000}$ 
}
cout<<"The exact solution = 1.6487212707001282"<<endl;
cout<<"Error = "<<yc[10000]-1.6487212707001282<<endl;
system("PAUSE");
return 0;
}

```

APPENDIX D

COMPUTER LANGUAGE C++ FOR TEST PROBLEM (iv)

This appendix is to show our algorithm (refer to section 4.1 in chapter 4) written in computer language C++ using the fourth test problem with $h = 0.01$, $h = 0.001$ & $h = 0.0001$

- $y' = x + y, \quad y(0) = 1, \quad 0 \leq x \leq 1, \quad h = 0.01$

```
#include<iostream>
#include<cmath>
#include<iomanip>
using namespace std;
int main ( )
{
double h=0.01, x[101], yp[101], yc[101], d[101];
x[0]=0;
yp[0]=1.0;
yc[0]=1.0;
d[0]=1.0;
cout<<"all valuable of x from 0 to 100"<<endl;
for (int i=0; i<101; i++)
{
x[i]=i*h;
cout<<"x"<<i<<"="<<x[i]<<endl;    // compute  $x_0, x_1, x_2, \dots, x_{100}$ 
}
```

```

for (int n=0; n<96; n=n+5)
{
for (int r=1; r<6; r++)
yp[n+r]=yc[n]+r*h*d[n];      // compute the predicted values
for (int j=1; j<=5; j++)
{
for (int i=1; i<6; i++)
{
if (j == 1) d[n+i]=x[n+i]+yp[n+i];      // compute the function
else d[n+i]=x[n+i]+yc[n+i];             // evaluation in case  $y' = x + y$ 
}
yc[n+1]=yc[n]+h*((251.0/720)*d[n]+(323.0/360)*d[n+1]
-(11.0/30)*d[n+2]+(53.0/360)*d[n+3]-(19.0/720)*d[n+4]);
yc[n+2]=yc[n]+h*((29.0/90)*d[n]+(62.0/45)*d[n+1]
+(4.0/15)*d[n+2]+(2.0/45)*d[n+3]-(1.0/90)*d[n+4]);
yc[n+3]=yc[n]+h*((27.0/80)*d[n]+(51.0/40)*d[n+1]
+(9.0/10)*d[n+2]+(21.0/40)*d[n+3]-(3.0/80)*d[n+4]);
yc[n+4]=yc[n]+h*((14.0/45)*d[n]+(64.0/45)*d[n+1]
+(8.0/15)*d[n+2]+(64.0/45)*d[n+3]+(14.0/45)*d[n+4]);
yc[n+5]=yc[n]+h*((95.0/144)*d[n]-(25.0/72)*d[n+1]
+(25.0/6)*d[n+2]-(175.0/72)*d[n+3]+(425.0/144)*d[n+4]);
// compute the corrected values
if (j == 5) d[n+1]=x[n+1]+yc[n+1];
if (j == 5) d[n+2]=x[n+2]+yc[n+2];
if (j == 5) d[n+3]=x[n+3]+yc[n+3];
if (j == 5) d[n+4]=x[n+4]+yc[n+4];
}
}
}

```

```

if (j == 5) d[n+5]=x[n+5]+yc[n+5];
}
}
cout<<"all valuable of y from 0 to 100"<<endl;
for (int r=0; r<101; r++)
{
cout<<fixed<<showpoint<<setprecision(16);
cout<<"y"<<r<<"="<<yc[r]<<endl;    // compute  $y_0, y_1, y_2, \dots, y_{100}$ 
}
cout<<"The exact solution = 3.4365636569180906"<<endl;
cout<<"Error = "<<3.4365636569180906-yc[100]<<endl;
system("PAUSE");
return 0;
}

```

- $y' = x + y, \quad y(0) = 1, \quad 0 \leq x \leq 1, \quad h = 0.001$

```

#include<iostream>
#include<cmath>
#include<iomanip>
using namespace std;
int main ( )
{
double h=0.001, x[1001], yp[1001], yc[1001], d[1001];
x[0]=0;
yp[0]=1.0;
yc[0]=1.0;
d[0]=1.0;

```

```

cout<<"all valuable of x from 0 to 1000"<<endl;
for (int i=0; i<1001; i++)
{
x[i]=i*h;
cout<<"x"<<i<<"="<<x[i]<<endl;    // compute  $x_0, x_1, x_2, \dots, x_{1000}$ 
}
for (int n=0; n<996; n=n+5)
{
for (int r=1; r<6; r++)
yp[n+r]=yc[n]+r*h*d[n];    // compute the predicted values
for (int j=1; j<=5; j++)
{
for (int i=1; i<6; i++)
{
if (j == 1) d[n+i]=x[n+i]+yp[n+i];    // compute the function
else d[n+i]=x[n+i]+yc[n+i];    // evaluation in case  $y' = x + y$ 
}
}
yc[n+1]=yc[n]+h*((251.0/720)*d[n]+(323.0/360)*d[n+1]
-(11.0/30)*d[n+2]+(53.0/360)*d[n+3]-(19.0/720)*d[n+4]);
yc[n+2]=yc[n]+h*((29.0/90)*d[n]+(62.0/45)*d[n+1]
+(4.0/15)*d[n+2]+(2.0/45)*d[n+3]-(1.0/90)*d[n+4]);
yc[n+3]=yc[n]+h*((27.0/80)*d[n]+(51.0/40)*d[n+1]
+(9.0/10)*d[n+2]+(21.0/40)*d[n+3]-(3.0/80)*d[n+4]);
yc[n+4]=yc[n]+h*((14.0/45)*d[n]+(64.0/45)*d[n+1]
+(8.0/15)*d[n+2]+(64.0/45)*d[n+3]+(14.0/45)*d[n+4]);
yc[n+5]=yc[n]+h*((95.0/144)*d[n]-(25.0/72)*d[n+1]

```

```

+(25.0/6)*d[n+2]-(175.0/72)*d[n+3]+(425.0/144)*d[n+4]);
// compute the corrected values
if (j == 5) d[n+1]=x[n+1]+yc[n+1];
if (j == 5) d[n+2]=x[n+2]+yc[n+2];
if (j == 5) d[n+3]=x[n+3]+yc[n+3];
if (j == 5) d[n+4]=x[n+4]+yc[n+4];
if (j == 5) d[n+5]=x[n+5]+yc[n+5];
}
}
cout<<"all valuable of y from 0 to 1000"<<endl;
for (int r=0; r<1001; r++)
{
cout<<fixed<<showpoint<<setprecision(16);
cout<<"y"<<r<<"="<<yc[r]<<endl;  // compute  $y_0, y_1, y_2, \dots, y_{1000}$ 
}
cout<<"The exact solution = 3.4365636569180906"<<endl;
cout<<"Error = "<<3.4365636569180906-yc[1000]<<endl;
system("PAUSE");
return 0;
}

```

• $y' = x + y, \quad y(0) = 1, \quad 0 \leq x \leq 1, \quad h = 0.0001$

```

#include<iostream>
#include<cmath>
#include<iomanip>
using namespace std;
int main ( )

```

```

{
double h=0.0001, x[10001], yp[10001], yc[10001], d[10001];
x[0]=0;
yp[0]=1.0;
yc[0]=1.0;
d[0]=1.0;
cout<<"all valuable of x from 0 to 10000"<<endl;
for (int i=0; i<10001; i++)
{
x[i]=i*h;
cout<<"x"<<i<<"="<<x[i]<<endl;    // compute  $x_0, x_1, x_2, \dots, x_{10000}$ 
}
for (int n=0; n<9996; n=n+5)
{
for (int r=1; r<6; r++)
yp[n+r]=yc[n]+r*h*d[n];    // compute the predicted values
for (int j=1; j<=5; j++)
{
for (int i=1; i<6; i++)
{
if (j == 1) d[n+i]=x[n+i]+yp[n+i];    // compute the function
else d[n+i]=x[n+i]+yc[n+i];    // evaluation in case  $y' = x + y$ 
}
}
yc[n+1]=yc[n]+h*((251.0/720)*d[n]+(323.0/360)*d[n+1]
-(11.0/30)*d[n+2]+(53.0/360)*d[n+3]-(19.0/720)*d[n+4]);
yc[n+2]=yc[n]+h*((29.0/90)*d[n]+(62.0/45)*d[n+1]

```



الملخص

لقد أعطيت الحلول العددية للمعادلات التفاضلية (مسائل القيم الأولية) اهتماماً كبيراً بشأن تصميم وبناء الخوارزميات نظراً لتطبيقاتها الكثيرة مثل نمذجة النقل البعيد المدى لملوثات الهواء في الغلاف الجوي.

في هذا البحث تم ابتكار طريقة جديدة من الدرجة الخامسة لحل مسائل القيم الأولية للمعادلات التفاضلية وقد تم اختبارها على عدة مسائل وكتابتها بلغة ++C للتأكد من دقة النتائج ومن ثم مقارنتها مع إحدى طرق رنك-كوتا من نفس الدرجة.

جامعة البحرين

قسم الرياضيات

كلية العلوم



الحلول العددية للمعادلات التفاضلية باستخدام طرق التنبأ والتصحيح

أطروحة مقدمة كجزء من متطلبات الحصول على درجة (الماجستير) في (الرياضيات)

إعداد

نور عبدالجليل سلمان الشاخوري

الرقم الجامعي ٢٠٠٣٦٠٩٢

إشراف

د. سلمان حسن عباس

(أستاذ مشارك)

جامعة البحرين

مملكة البحرين

يناير ٢٠١٥