A CLASS OF SECOND DERIVATIVES LINEAR MULTI-STEP METHOD FOR INTEGRATION OF STIFF FIRST ORDER INITIAL VALUE PROBLEMS IN ORDINARY DIFFERENTIAL EQUATIONS

BY

ADETUNJI, ADERONKE OLAITAN

IMC/01/0995

THESIS IN THE DEPARTMENT OF MATHEMATICAL SCIENCES TO THE SCHOOL OF POST – GRADUATE STUDIES TOWARDS A PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF MASTERS OF TECHNOLOGY (M.TECH) IN MATHEMATICS OF FEDERAL UNIVERSITY OF TECHNOLOGY, AKURE, ONDO STATE, NIGERIA.

JANUARY, 2007.

CERTIFICATION

This is to certify that this project was carried out by Adetunji Aderonke Olaitan of the Department of Mathematical Sciences, Federal University of Technology, Akure.

..... 31-03-06

Dr. R.A. Ademiluyi
Major Supervisor

Date

..... 31/3/2006

Prof. D.O. Awoyemi,
Co-Supervisor

Date

DEDICATION

This work is specially dedicated to the Trinity and members of my family.

ACKNOWLEDGMENT

I sincerely express my gratitude to my supervisor Dr. R.A. Ademiluyi for his fatherly advice, contribution, patience, guidance, instruction at each stage of this research work. May God in his mercy reward him greatly. The contribution of Prof. D.O. Awoyemi, my co-supervisor cannot be over-emphasised. I appreciate him for his support in the area needed in the work.

My gratitude also goes to Prof. S.T. Oni, head of Department, Prof. J.K.Ogunmoyela, the former Head of Department, all members of staff and non teaching staff of Department of Mathematics, Prof. Ade Ajayi, Department of Materials and Metallurgical Engineering, Mr Kayode Ilori, Health Department, Brother Sola, Computer Resource Center, colleagues at the Federal University of Technology, Akure for their tremendous services, support and assistance rendered.

I express my warmest love and gratitude to the members of my family for their support morally and financially at various stages of my education career.

I cannot end this extension of gratitude without mentioning Mr O.O Obasa, Principal of Fatima College Ikire, Sister Julie and Funke, Mr and Mrs Oyeleke and all well wishers who have influenced the development of this work in one way or the other.

I acknowledge the contribution of the authors of the reference books and journals.

May God in his infinite mercy bless you all. Amen.

Adẹtúnji A.O.

TABLE OF CONTENT

CHAPTER TWO

CHAPTER THREE

CHAPTER FOUR

CHAPTER FIVE

Implementation and Numerical Results

CHAPTER SIX

LIST OF FIGURES

Figures

LIST OF TABLES

ABSTRACT

In this thesis, a class of second derivative Linear Multistep methods for integration of stiff initial value problem in ordinary differential equations is developed. The method is motivated by the second derivative method by Enright (1972). The basic properties of the method, such as the order, error term, consistency and stability were investigated. The methods are found to be consistent, stable and of order k +2. The developed formula was computerised and implemented on a digital computer to solve some sample initial value problems of Ordinary Differential Equations. Numerical results show that the new method is accurate and compares favourably with the existing second derivative method and Backward Difference Formula respectively.

CHAPTER ONE

INTRODUCTION

1.1 ORDINARY DIFFERENTIAL EQUATIONS

Some real life problems have to be translated from Physical problems to corresponding Mathematical problems (modeling) in order to achieve a desired solution. These Mathematical problems involve rate of change of one variable in relation to another. This rate of change is called a derivative.

A differential equation is a relation of the form,

$$y^{(n)} = f(x, y, y^{I}, y^{II}, - - - , y^{(n-1)}) \quad - \quad - \quad - \quad - \quad (1.1)$$

between the function y, its derivatives $y^{(n)}$, $n = 0(1)\ n - 1$ and the variable x upon which it depends.

The integer n denotes the order of the equation. If the dependent variable y and its derivative occurs to the first degree, then the differential equation is linear otherwise it is non-linear.

The concept of ordinary differential equation is of great importance in Science, Technology, Engineering, Physics, Economics and Management because some of their Mathematical formulation leads to ordinary differential equations. For example:

(a) The rate of growth of a certain population p can be determined from the mathematical equation.

$$\partial p/\partial t = kp \quad - \quad - \quad - \quad - \quad - \quad - \quad (1.2)$$

where k, is a real constant and p is the population respectively. This equation can help to determine the population growth of a giving inhabitance (Malthus, 1798).

(b) The flow of electric charge q across a section of a conductor can be described mathematically by equations

$$\partial q/\partial t = I, \quad - \quad - \quad - \quad - \quad (1.3)$$

(c) Newton's proposition for the motion described the motion of a particle of mass m falling under gravity in a medium in which the resistance to motion is proportional to the velocity (ds/dt) is describable in mathematical Language as

$$m\frac{d^2s}{dt^2} = mg - K\frac{ds}{dt} \quad - \quad - \quad - \quad - \quad (1.4)$$

Differential equations can be categorized into Ordinary Differential Equations (ODEs) and Partial Differential Equations (PDEs). Ordinary Differential Equations is a differential equation involving derivatives of a differentiable function of one independent variable while Partial Differential Equation is an equation involving a function of two or more variables and its partial derivates.

1.2 INITIAL VALUE PROBLEMS IN ORDINARY DIFFERENTIAL EQUATIONS

A solution of the differential equation (1.2) is a function $y = \phi(x)$ that is differentiable in a suitable number of times in some interval $I = [a, b]$ containing the independent variable x and which has the property that

$$f(x, \phi(x), \phi^{1}(x), \text{---}, \phi^{(n)}(x)) = 0$$

for all $x \in I$.

The general solution of the n^{th} order equation (1.1) contains n arbitrary constants. If in order to determine the n constants, n initial conditions are specified at a single point, then they are called Initial-conditions otherwise they are boundary conditions.

Initial value problems (I.V.Ps) is defined as the ordinary differential equations together with the sets of initial conditions. Thus an n^{th} order initial value problem of ODEs is of the form

$$\left.\begin{aligned} y^{(n)} &= f(x, y, y^{(1)}, \text{-----}, y^{(n-1)}) \\ y^{(r)}(xo) &= y_r^{(r)},\ r = 0\ (1)\ n\text{-}1 \end{aligned}\right\} \quad - \quad - \quad - \quad - \qquad (1.5)$$

1.3 PROPERTIES OF THE DIFFERENTIAL EQUATION

The properties of the differential equation (1.1) are important factors to consider especially when its numerical solutions are to be found. Since these properties determine whether the problems can have any solution or not. For

example, if a function f becomes infinite or undefined at some points in the domain of integration the partial derivatives of f do not exist or if they exist, they may be unbounded. As a result, some problems are to be modified in order to suite numerical integration or a new method may be developed to solve the problems.

Consequently, according to Coddington and Levingson (1955) a differential equation of typed (1.1.) has a unique solution, if the function f (x, y) is real valued and continuous at all points (x, y) in the region D defined by

$$D = \{(x, y)\ a \le x \le b, -\infty < y < \infty\}$$

where a and b are finite real number.

This means there exists a constant L such that for every $x \in D$, there exist y_1, y_2 such that (x, y_1) and (x, y_2) are both in D and

$$\left| f(x, y_1) - f(x, y_2) \right| \le L \left| y_1 - y_2 \right| \quad - \quad - \quad - \quad - \quad (1.6)$$

Then the equation has a unique solution. The requirement (1.6) is known as a lipschitz condition and the constant L as a lipschitz constant.

If the partial derivatives $\partial f/\partial y$ of f with respect to y are continuous and bounded, then the lipschitz constant L of the system may be taken to be

$$L = \sup_{(x,y) \in D} \left| \frac{\partial f}{\partial y}(x, y) \right| \quad - \quad - \quad - \quad - \quad - \quad (1.7)$$

where $L >> 1$

The system of initial value problems of ordinary equations with this property is called a stiff ordinary differential equations.

1.4 NATURE OF STIFF ORDINARY DIFFERENTIAL EQUATION

The early scientists and Mathematicians observed that the Mathematical formulation of physical situation in some field of applications such as chemical engineering, control theory yields initial value problems involving systems of ordinary differential equations which exhibit a phenomenon known as "stiffness".

Stiffness is a concept describing the nature of certain subset of ordinary differential equation whose solution contain component with fast and slow responses. The fast responding components are referred to as transient while the slow responding components are referred to as steady-state.

Stiffness may arise as a result of ill-conditioning of function f in (1.1), mixture of chemical species with different chemical reaction rates, motion of masses controlled by spring of varying stiffness (Cao et al 2002). The initial value problem of ordinary differential equation possessing these properties are said to be stiff ordinary differential equations.

Stiffness is quite related to the structure of the Jacobian matric J(x) of the differential equations with elements defined by

$J = (J_{ij})$

where $J_{ij} = \partial f_i / \partial y_j$ - - - - - - - - (1.8)

If λj , $j = 1(1)n$ are the eigen - value of $J(x)$ with the property $Re(\lambda_j) < 0$, but large in absolute sense, the system is stiff.

According to Ademiluyi (1987) a differential equation of the form

$$y^1 = f(x, y); \quad y(xo) = n \quad - \quad - \quad - \quad - \quad - \quad - \quad (1.9)$$

whose Jacobian J possesses eigen values

$$\lambda_j = u_j + iv_j; \quad j = 1(1)n \quad - \quad - \quad - \quad - \quad - \quad - \quad - \quad - \quad - \quad - \quad (1.10)$$

Satisfying

(a) $Re(\lambda_j) << 0, \quad j = 1(1)n$

(b) $\left(\underset{1 \le j \le n}{\text{Max}} \left|Re\,\lambda_j\right|\right) \Big/ \left(\underset{1 \le j \le n}{\text{Min}} \left|Re\,\lambda_j\right|\right) = S >> 1$

Where S is the stiffness ratio.

Condition (a) above shows that the system is stable while (b) indicates that the system possesses some component which decays very rapidly.

A linear system of equations

$$y^1 = Ay + \phi(x) \quad - \quad - \quad - \quad - \quad - \quad - \quad - \quad - \quad (1.11)$$

or a non-linear system

$$y^1 = f(x, y)$$

is said to be stiff in the interval I of x if.

- for $x \varepsilon I$ the eigen values $\lambda(x)$ of $A(\dot{x})$ or of the Jacobian $\partial f / \partial y$ satisfy a and b

for more details, consult Lambert (1980).

For clarity, consider the mxm in equation (1.11)

where A is an mxm matrix with distinct eigen values λ_j, $j = 1(1)m$ and corresponding eigen vectors C_j $j = 1(1)m$, with a general solution of the form

$$y(x) = \sum_{j=1}^{m} K_j \ell^{\lambda_j x} C_j + \psi(x) \quad - \quad - \quad (1.12)$$

assuming that $Re\lambda_j < 0$, $j = 1(1)m$ then the transient part

$$\sum_{j=1}^{m} K_j \ell^{\lambda_j x} C_j \longrightarrow 0 \text{ as } x \longrightarrow \infty$$

while steady-state component $\psi(x)$ settle down slowly.

Let $\lambda_j = u_j + iv_j$ be the eigen values of A such that

$$|Re(\lambda_j)| = |u_j| \geq |Im(\lambda_j)| = |V_j| \quad - \quad - \quad - \quad - \quad (1.13)$$

If the steady state solution $\psi(x)$ is to be found, the numerical solution (1.12) has to be pursed until the slowest decaying exponential in the transient solution $\ell^{\lambda_j x}$ is negligible (Butcher, 2001). So, the smaller the $Re(\lambda_j)$ the larger the range of integration because the eigen value of A far out to the left in the complex plane will force us to use excessive small step-lengths in order that $h\lambda_j$; $j = 1(1)m$ will lie within the region of absolute stability. The further out such eigen-values lie the more severe the restriction on step-length. Thus the basic difficulty identifiable with approximation of stiff initial value problem in Ordinary differential equations is the problem of satisfaction of absolute of stability. It necessitates the adoption of an

integration stepsize such that every one of the product $h\lambda_j$, $j = 1(1)m$ lies within the region of absolute stability of the numerical integrator.

1.5 THE EXISTING NUMERICAL METHODS

Ordinary differential equations can be solved by two basic methods namely analytical methods and numerical methods.

Analytical methods consist of expressing the dependent variable y as a function of x or in terms of elementary functions or some special functions such as the Bessel function. For more details consult Finizio and Ladas (1970). However, there are many differential equations that cannot be solved by the closed form method. This difficulty motivated scientists and mathematicians to develop computational formulae for the numerical approximation of the solution of such ordinary differential equations. This numerical method can also be categorized into one-step methods and multi-step methods respectively.

A one-step method is a numerical integration formula which requires one point information for the next approximate value of the solution. Examples of one-step methods are Euler method, Taylor series method, Runge-Kutta methods while Adams methods (Bashforth, Moulton) and Backward differentiation methods are examples of multi-step methods.

The general form of Linear Multi-step method is

$$\sum_{j=o}^{k} \alpha_j y_{n+j} = h \sum_{j=o}^{k} \beta_j f_{n+j} \quad - \quad - \quad - \quad (1.14)$$

Where α_j and β_j are real constants.

This formula (1.14) had been extensively studied and applied for numerical solution of several ODEs problems and have been found effective and efficient, however a sub-class of it in the form

$$\sum_{j=o}^{k} \alpha_j y_{n+j} = h\beta_k f_{n+k} \quad - \quad - \quad - \quad - \quad - \quad (1.15)$$

called Backward difference method was proposed by Gear (1971) for approximation of stiff ODES. This method was used for many years that it almost misled researchers to believe that there may be no need for further development of method for stiff equations. This popularity stemed from the code 'DIFSUB' and several variants of it amongst which are LISODE by Gear and Hindmarsh (1974) and EPISODE by Byrne and Hindmarsh (1975). It was later discovered that the method could not perform satisfactory on stiff ODEs with eigen-value closes to the imaginary axis. This class of equations are called stiff oscillatory ODEs.

This motivated Enright (1972) to suggest another method with more analytical properties of f and came up with second derivative formulas (SDFs)

$$y_{n+k} = y_{n+k-1} + h \sum_{j=0}^{k} \beta_j y'_{n+j} + h^2 \gamma_k y''_{n+k} \quad - \quad - \quad (1.16)$$

where β_j and γ_k are real constants. This formula has better stability property than BDFs codes but possess some difficulties which include the problems of solving system of algebraic equations. Enright, Sedgwick and Hull (1975) performed evaluation test on the method and indicated that there is only small difference in efficiency when compared with BDF (Ademiluyi, 1987). This lack of advantage of Enright method over Gears formula is one of the main motivations for this work.

1.6 MOTIVATION

As mentioned above, it has been found that the linear multi-step method (1.15) and (1.16) are suitable for both stiff and non-stiff equations but inadequate for equations whose stiffness are caused by eigen-value that are closed to the imaginary axis, (that is stiff oscillatory initial value problems of ordinary differential equations). The quest for better method which can solve such class of equation is another motivation for the present research work.

Consequently, the research work is directly at improving the Enright formula (1.16) so that it can be made suitable and efficient for integration of both non-stiff, stiff and stiff oscillatory ODEs. This modification adopts more analytical properties of the differential equation by way of involving more of the second derivatives of y into the numerical formula to obtain a new formula

$$y_{n+k} = y_{n+k-1} + h\,\beta_k\, y'_{n+k} + h^2 \sum_{j=o} \gamma_j\, y''_{n+j} \quad - \quad - \quad (1.17)$$

This method will have a better accuracy efficiency and stability properties than the Gear and Enright's methods.

1.7 OBJECTIVES

The objectives of this research work are to:

(i) Develop a new class of second derivative methods for integration of non-stiff, stiff and stiff oscillatory ODEs.

(ii) Determine its predictors

(iii) Analyse the basic properties of the methods

(iv) Determine the interval of absolute stability of the method

(v) Develop computer programme to implement the methods on a digital computer with sample problems.

(vi) Compare the accuracy of the results with those of Ademiluyi (1987) and Gear (1971).

1.8 METHODOLOGY

To achieve the above objectives, the Taylor series expansion process is adopted to generate the basic parameter of the method. The stability properties of the formula were analysed using the standard Dahlquist scalar test equation.

$$y^{1} = \lambda y, \quad y(x_0) = y_o, \quad a \leq x \leq b \qquad (1.18)$$

the non-linear equation

$$G(y_{n+k}) = 0 \qquad (1.19)$$

where $G(y^{(s)}_{n+k}) = y_{n+k-1} - h\ \beta_k\ y^{1}(y^{(s)}_{n+k}) - h^2 \sum_{j=0}^{k-1} \gamma_j\ y^{II}(y^{(s)}_{n+k}) - h^2\ \gamma_k\ y^{II}(y^{(s)}_{n+k})$

generated by the implicitness of the method were solved by the Newton's iterative scheme

$$w^{(s)}_{n+k}\ d^{(s+1)}_{n+k} = G(y^{(s)}_{n+k}) \qquad (1.20)$$

$$\text{where } d^{(s)}_{n+k} = y^{(s+1)}_{n+k} - y^{(s)}_{n+k} \qquad (1.21)$$

$$w^{(s)} = (I - h\beta_k J - h^2\ \gamma_k J^2) \qquad (1.22)$$

$$J = \partial f / \partial y\ (y^{(s)}_{n+k})$$

where w_{n+k} is called Newton iteration matrix and, J is the Jacobian of the differential systems. The iteration was implemented on a digital computer using Fortran programming language.

CHAPTER TWO

PRELIMINARY CONCEPTS AND PRINCIPLES

2.1 PRELIMINARY CONCEPTS

These are concepts which are relevant to the understanding of the new integration schemes. Some of the concepts are:

(a) Mesh point

The first step in the numerical solution of any ordinary differential equations is to divide the integration interval (a,b) into n-sub-intervals of step-length h, where $h = \frac{b - a}{N}$

$I_n = (x_{n+1}, x_n)$, and

$x_n = x_0 + nh, \quad n = 1, 2, 3 - - - - N$

The sequence of points

$a = x_0 < x_1 < x_2 < - - - < x_n = b$ are called mesh points

(ii) Explicit and Implicit Method

The general form of one-step methods are of the form

$$y_{n+1} = y_n + hf(x_n, y_n) \qquad (2.1)$$

where f is a function and h is the step size. Since y_{n+1} is not present on the right hand side of the equation (2.1), then the method is explicit, otherwise it is called implicit methods. Equation (1.17) is an implicit method because in

$f_{n+1} = f(x_{n+1}, y_{n+1})$, y_{n+1} appears in f. To use equation (1.16), we must predict or estimate a value of y_{n+1} in advance.

iii. Predictor Method

A predictor is another method that provides starting values at each step of the application of another method. This predictor must be of the same order of accuracy as the method it wants to serve to avoid data error and as well a better approximation (for more details see Fox and Mayer, 1987). The predictors used in the research work are Runge-Kutta methods of order three and four.

2.2 ITERATION SCHEME FOR IMPLICIT EQUATION

The numerical solution of the non-linear equation (1.17) generated by the implicit method will be done iteratively by constructing recursive relation of the form

$$y_{n+1}^{(s+1)} = f(y_{n+1}^{(s)}), \quad s = 0, (1) M \qquad (2.2)$$

The iteration is started with an approximation $y_{n+1}^{(0)}$ which has a better value than y_o and the computation is then repeated until convergence condition is attained.

2.3 BASIC PROPERTIES OF THE DIFFERENCE APPROXIMATIONS TECHNIQUE

The major concern of numerical computation is to make the global errors as small as possible. The difference between the numerical solution y_{n+1} at step x_{n+1} and the theoretical solution $y(x_{n+1})$ is called the global errors. It is defined as

$$e_{n+1} = y_{n+1} - y(x_{n+1}) \quad - \quad - \quad - \quad - \quad - \quad (2.3)$$

The error has to be made negligible by adjusting the step size h. The following definitions will make the concept clear.

Definition 2.1

The truncation error t_{n+k} associated with equation (1.14) is the amount by which the theoretical solution fails to satisfy the difference equation equivalent of the initial value problems. That is for system (1.2.) and method (1.14), the local truncation error t_{n+k} is as follows

$$t_{n+k} = y(x_{n+k}) - \sum_{j=0}^{k-1} \alpha_j\, y(x_{n+j}) - h\,\beta_k\, y^1(x_{n+k}) - h \sum_{j=0}^{k-1} \beta_j\, y^1(x_{n+j}) \quad - (2.4)$$

by subtracting equation (1.14) from equation (2.4) and simplifying

$$t_{n+k} = 1 - h\,\beta_k \frac{\partial f(\eta)}{\partial y} [\, y(x_{n+k}) - y_{n+k}\,]$$

where $\eta \in [\, y_{n+k},\ y(x_{n+k})\,]$, J, $\partial f/d\partial y$ = L lipschitz constant.

$$|t_{n+k}| \le |\, 1 - h\,\beta_k\, L\,| \cdot |y(x_{n+k}) \quad y_{n+1}|$$

setting $R = \left\| \dfrac{1}{(1 - h\,\beta_k\, L)} \right\|$

This makes the equation becomes

$$|y(x_{n+k}) - y_{n+k}| \le R\ t_{n+k} \quad - \quad - \quad - \quad - \quad - \quad (2.5)$$

This shows that the local truncation error is related to the global error per step, particularly when the derivation and computation of the local truncation error is rigorous and well calculated.

Definition 2.2

A method of the form (1.17) is said to be convergent if

$$\lim_{n \to \infty} y_n = y_{(x_n)}$$

where $y(x_n)$ is the theoretical solution of problem (1.1) at $x = x_n$ and y_n is the numerical approximation to it.

Definition 2.3

The method of class (1.17) is said to be consistent with the differential equation (1.1) if its order p is at least one. This means that $p \geq 1$

2.4 Instabilities of Numerical solution

A numerical solution to an initial value problem in ordinary differential equation is said to be accurate if the numerical results do not deviate significantly from the corresponding value of the exact solution otherwise it is in-accurate. The inaccuracy of solution can make the solution unstable.

There are two basic types of instability namely inherent and induced instability respectively. Inherent instability is due to error in mathematical

transformation of the real situation into differential equation while the induced instability is a characteristic of the numerical methods.

These concepts could be more understood by considering the scalar initial value problem (1.18). That is equation

$y^1 = \lambda y, \quad y(x_o) = y_o$

with the Re $(\lambda) < 0$ over a closed interval $a \leq x \leq b$

the theoretical solution of the equation (1.18)

$$y(x) = y_o e^{\lambda x} \quad - \quad - \quad - \quad - \quad - \quad - \quad - \quad (2.6)$$

if the initial condition in (1.18) is slightly modified by $\varepsilon > 0$, the initial value becomes $y(xo) = y_o + \varepsilon$ - - - - - (2.7)

the general solution of (1.18) together with equation (2.7) resulted to

$$y(x) = y_o e^{\lambda x} + \varepsilon e^{\lambda x} \quad - \quad - \quad - \quad - \quad - \quad - \quad - \quad (2.8)$$

for $\varepsilon > 0$, no matter how small, the second term $\varepsilon e^{\lambda x}$ will eventually grow exponentially, as the computation proceeds. Thus if Re $(\lambda) > 0$, $y(x) = y_o e^{\lambda x}$ will become unstable. This instability could be cured by reformulating the problem to reduce the spurious part of the solution.

At times, the differential equation may be stable yet the numerical method may produce unstable solution. This kind of instability is called induced instability. This may be as a result of adopting finite instead of infinite iteration process in computer implementation and truncation error of the scheme. The

instability show up in the spurious exponential as subjected to scalar test may be minimized by decreasing the step size.

Definition 2.4

A numerical method of the class of formula (1.17) is said to be stable if the difference $e_n = y_{(sn)} - y_n$ between the numerical solution and the theoretical solution (can be made as small as possible) is bounded. That is, if there exists e_0 such that

$|e_n| \leq k|e_n|$ where k is a real constant.

Definition 2.5

The numerical method (1.17) is said to be absolutely stable if for a given $h\lambda$ all the roots r_s of the stability polynomials

$\pi(r, h) = p(r) - h\delta(r)$ - - - - - - (2.9)

that is $|r_s| \leq |$, where r_s are the roots of $\pi(r, h) = 0$. In equation (2.9) $p(r)$ and $\delta(r)$ are first and second characteristic polynomial of the method.

Definition 2.6

A numerical method is A-stable if its region of absolute stability contains the whole of the left-hand of the complex $h\lambda$ plane $Re(h\lambda) < 0$.

This region is shown in the figure below.

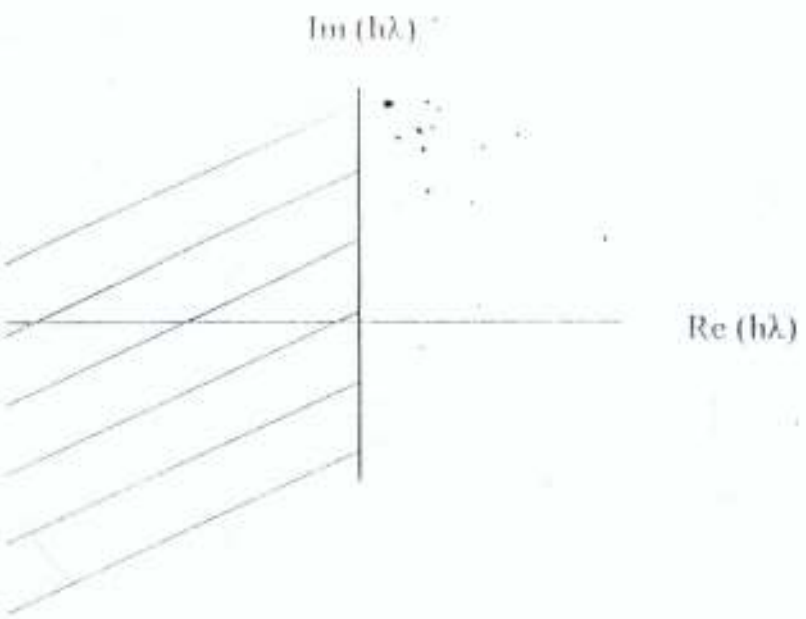


Figure 1- A-stability region for K-step method

Dalhquist (1963) explained in his proposed theorem that if an A-stable method is applied to a stiff system, no matter how large the max $\left|\mathrm{Re}\,(x_i)\right|$ no stability restriction on h can result

CHAPTER THREE

DERIVATION OF THE METHOD

In order to ensure high order of accuracy for the proposed method (1.17), the unknown coefficients β_k and γ_j, $J = 0\,(1)\,k$ are determined as to ensure that the resultant methods are:

(i) Accurate

(ii) Consistent

(iii) Zero-stable

(iv) Convergent and

(v) A-stable

Consequently, the methods are developed for the case $k = 1, 2, 3, 4, 5$ and 6. If T_{n+k} denotes the local truncation error associated with the method, we define T_{n+k} as

$$T_{n+k} = y(x_{n+k}) - y(x_{n+k-1}) - h[\beta_k y'(x_{n+k}, y_{n+k}) + h^2 \sum_{j=0}^{k} \gamma_j y''_{(x_{n+j}, y_{n+j})} \quad (3.1)$$

where $y_{(x_{n+k})}$ are theoretical solution of the equation at $x = x_n$

by taking the Taylor series expansion of $y_{(x_{n+k})}$, $y_{(x_{n+k})}$, $y'_{(x_{n+k})}$ and $y''_{(x_{n+j})}$ for $j = 0\,(1)\,k$ about x_n we have

$$y_{(x_n+k)} = \sum_{p=0}^{\infty} \frac{(kh)^p y^{(p)}_{(x_n)}}{p!} \quad (3.2)$$

$$y_{(x_n+k-1)} = \sum_{p=0}^{\infty} \frac{((k-1)h)^p y^{(p)}(x_n)}{p!} \quad - \quad - \quad - \quad (3.3)$$

$$y^1_{(x_n+k)} = \sum_{p=0}^{\infty} \frac{(kh)^p y^{(p+1)}(x_n)}{p!} \quad - \quad - \quad - \quad (3.4)$$

$$y^{11}(x_{n+j}) = \sum_{p=0}^{\infty} \frac{(jh)^p}{p+1} y^{(p+2)}(x_n) - \quad (3.5)$$

On substituting equations (3.2) – (3.5) into equation (3.1) and arranging terms in equal power of h, the equations become

$T_{n+k} = C_0 y_{(x_n)} + C_1 h y^1_{(x_n)} + C_2 h^2 y^{II}_{(x_n)} + \ldots + C_p h^p y^p_{(x_n)}$ where the coefficients C's are:

$C_0 = 0$

$C_1 = 1 - \beta_k = 0$

$C_2 = 1/2! (k^2 - (k-1)^2 - 2k\ \beta_k - (2)(1) \sum_{j=0}^{k} \gamma_j) = 0$

$C_3 = 1/3! (k^3 - (k-1)^3 - 3k^2 \beta_k - (3)(2) \sum_{j=0}^{k} j\ \gamma_j) = 0$

$C_4 = 1/4! (k^4 - (k-1)^4 - 4k^3 \beta_k - (1)(3) \sum_{j=0}^{k} j^2 \gamma_j) = 0$

$C_5 = 1/5! (k^5 - (k-1)^5 - 5k^4 \beta_k - (5)(1) \sum_{j=0}^{k} j^3 \gamma_j) = 0$

$C_p = 1/p! (k^p - (k-1)^p - pk^{p-1} \beta_k - p(p-1) \sum_{j=0}^{k} j^{p-2} \gamma_j) = 0$

$$T_{n+k} = \sum_{j=0}^{k} C_j h^j y^j_{(xn)} \quad - \quad - \quad - \quad - \quad - \quad (3.6)$$

Definition

According to Lambert (1973), a computational method such as (1.17) with truncation error T_{n+k} is said to be of order P if in equation (3.6)

$C_0 = C_1 = C_2 \quad - \quad - \quad - \quad - \quad - \quad = C_p = 0$, but $C_{p+1} \neq 0$

then the formula (1.17) is of order P and $T_{n+k} = C_{p+1} h^{p+1} y^{p+1}_{(xn)}$ is called the principal truncation error of the method. It can be seen from (3.6) that the Cjs depends on the parameter B_s, y_j, the values of these parameters are determined by imposing arbitrary order P of accuracy on T_{n+k} leading to the set of linear equations.

$$C_j = 0, \quad j = 0\,(1)\,P \quad - \quad - \quad - \quad - \quad - \quad - \quad (3.7)$$

These equations (3.7) were solved for the parameters by Gaussian elimination method and the results are as shown in the tale below

TABLE 1: Coefficient of the proposed SDFs for $1 \leq k \leq 6$

Step	Order of accuracy	Coefficient							Error constant
K	P	γ_0	γ_1	γ_2	γ_3	γ_4	γ_5	γ_6	Ecp
1	3	-1/6	-1/3						1/24
2	4	1/24	-1/4	-7/24					1/45
3	5	-1/45	13/120	-19/60	-97/360				7/480
4	6	7/480	-29/360	47/240	-3/8	-367/1440			107/10080
5	7	-1404413/60480	666201/45360	76297/15120	72271/15120	-18719/12096	-1231/5040		1477801/10886406
6	8	995/120960	-1209/20160	7703/40320	10621/20240	-17151/40320	-1925*/4032	-28549/120960	-7573841/25401600

However , we illustrate these with the cases k=1,2,3.

3.1 One-Step Method

By setting k = 1 in equation (3.1) the equation becomes one-step method

$$y_{n+1} = y_n + h\beta_1 y'_{n+1} + h^2(\gamma_0 y''_n + \gamma_1 y''_{n+1}) \quad - \quad - \quad - \quad - \quad (3.8)$$

A one-step application of the method for solution of (3.1) will lead to a truncation error T_{n+1} defined as

$$T_{n+1} = y_{(x_{n+1})} - y(x_n) - h\beta_1 y'_{(x_{n+1})} - h^2(\gamma_0 y''(x_{n+1}) + \gamma_1 y''_{(x_{n+1})} \quad - \quad (3.9)$$

By adopting Taylor series expansion of $y(x_{n+1})$, $y'(x_{n+1})$, $y''(x_{n+1})$ and substituting into the equation (3.1), collecting terms in equal power of h, we have

$$T_{n+1} = (1-1)\,yn + (1-(1-1)-\beta_1)\,hy^1(xn) + (\tfrac{1}{2}(1^2-(1-1)^2-2(1)\beta_1) - 2(2-1)\sum_{j=o}^{1}\gamma j)\,h^2 y^{II}(xn) + (1/6(1^3-(1-1)^3-3(1)^2\beta_1) - 3(2)\sum_{j=o}^{1} j\,\gamma j)\,h^3 y^{III}(xn) + \cdots + 1/p!\,(1^p-(1-1)^p - p(1)^{p-1}\beta_1 - p(p-1)\sum_{j=o}^{1} j^{p-2}\,\gamma j)\,h^p y$$

(3.10)

Imposing accuracy of order 3 on T_{n+1} we have

$$\left.\begin{aligned} Co &= 0 \\ C_1 &= 1-(1-1)-\beta_1 = 0 \\ C_2 &= \tfrac{1}{2}(1^2-(1-1)^2-2(1)\beta_1 - 2(2-1)\sum_{j=o}^{1}\gamma j = 0 \\ C_3 &= 1/6(1^3-(1-1)^3-3(1^2)\beta_1 - 3(2)\sum_{j=o}^{1} j\,\gamma j = 0 \end{aligned}\right\} \quad (3.11)$$

Solving the equations, we get

$$\left.\begin{aligned} B_1 &= 1 \\ \gamma_1 &= -1/6 \\ \gamma_2 &= -1/3 \end{aligned}\right\} \quad (3.12)$$

substituting the parameters (3.12) into equation (3.8) the one-step method is of the form

$$y_{n+1} = y_n + h\,y^1{}_{n+1} - 1/6h^2(y^{II}{}_n + 2\,y^{II}{}_{n+1})$$

is generated and its order is 3.

3.2 Two-step method

By setting $k = 2$ in equation (3.1) the general 2-step method is of the form:

$$y_{n+2} = y_{n+1} + h\,\beta_2 y^1{}_{n+2} + h^2(\gamma_0 y^{II}{}_n + \gamma_1 y^{II}{}_{n+1} + \gamma_2 y^{II}{}_{n+2} - \quad (3.13)$$

the steps adopted for the case k=1 are followed systematically. By adopting the Taylor series of $y(x_{n+1})$, $y(x_{n+2})$, $y^1(x_{n+2})$, $y^{11}(x_{n+1})$, $(y^{11}(x_{n+2})$ and (3.14) substituting into (3.13) and collecting terms in equal powers of h, we have

$$T_{n+2} = (1-1)\, y_{(xn)} + (1-(1-1)-\beta_2)\, h\, y^1_{(xn)} + (\tfrac{1}{2}(2^2-2\text{-}1)^2 - 2(2)\beta_2 - 2$$

$$(2\text{-}1)(\Sigma \gamma j)\, h^2 y^{11}(_{xn}) + 1/6\,(2^3\text{-}(2\text{-}1)^3 - 3(2^2)\beta_2 - 3(2)\sum_{j=0}^{2} j\gamma_j)\, h^3 y^{111(xn)} + 1/24\,(2^4 - (2\text{-}1)^4$$

$$-4(2^3)\beta_2 - 4(3)\sum_{j=0}^{2} j^{p-2}\gamma_j\, h^3 y^{IV}(xn) + \ldots + (1/P!)\,(2^p - (2\text{-}1)^p - (2)^{p-1}\beta_2 - p)p\text{-}1)$$

$$\sum_{j=0}^{2} j^{p-2}\gamma_j\, h^p y^p(xn) - - - - - - - - - - - - \quad (3.14)$$

imposing accuracy of order 4 on T_{n+2} we obtain

$$\left.\begin{aligned} C_0 &= 0 \\ C_1 &= 1-(1-1)-\beta_2 = 0 \\ C_2 &= \tfrac{1}{2}(2^2\text{-}(2\text{-}1)^2 - 2(2)\beta_2 - 2(2\text{-}1)\sum_{j=0}^{2}\gamma_j = 0 \\ C_3 &= 1/6\,(2^3 - (2\text{-}1)^3 - 3(2^2)\beta_2 - 3(2)\sum_{j=0}^{2} j^2\gamma_j = 0 \\ C_4 &= 1/24\,(2^4 - (2\text{-}1)^4 - 4(2^3)\beta_2 - 4(3)\sum_{j=0}^{2} j^{p-2}\gamma_j = 0 \end{aligned}\right\} \quad 3.15$$

Solving the equations

$$\left.\begin{aligned} \gamma_0 &= 1/24 \\ \gamma_1 &= -1/4 \\ \gamma_2 &= -7/24 \\ \beta_2 &= 1 \end{aligned}\right\} \qquad 3.16$$

Substituting the parameter in (3.16) into equation (3.13) the two-step method is of the form

$$y_{n+2} = y_{n+1} + hy^{I}_{n+2} + h^2/24(y^{II} - 6y^{II}_{n+1} - 7y^{II}_{n+2})$$

is generated and its order is 4

3.3 Three-Step Method

By setting k = 3 in equation (3.1) the general three-step method is of the form

$$y_{n+3} = y_{n+2} - h\ \beta_3 y^{I}(x_{n+3}) + h^2(\gamma_0 y^{II}n + \gamma_1 y^{II}_{n+1} + \gamma_2 y^{11}_{n+2} + \gamma_3 y^{II}_{n+3}) \qquad (3.17)$$

the steps adopted for the case K = 1 are followed systematically as well. By adopting the Taylor series of y (xn+2), y (xn+3), y^{I} (xn+3), y^{II} (xn+3) and substituting into the equation (3.1)

expanding and collecting in equal powers of h, we have

$$T_{n+3} = (1\text{-}1)\, y(xn) + (1 - (1\text{-}1) - \beta_3)\, hy^{I}(xn) + (1/2\,(3^2 - (3\text{-}1)^2 - 3(3)\beta_3 - 2(2\text{-}1)\sum_{j=o}^{3}\gamma_j\, h^2 y^{II}(xn) + (1/6\,(3\sum_{j=o}^{3} - (3\text{-}1)^3 - 3(3)^2\beta 3 - 3(2)\sum_{j=o}^{3} j\,\gamma_j)\, h^3 y^{III}(xn) + (1/24\,(3^4 - (3\text{-}1)^4 - 4(3)^3\beta_2 - 4(3)\sum_{j=o}^{3} j^2/2\,\gamma_j)\, h^4 y^{IV}(xn) + (1/120\,(3^5 - (3\text{-}1)^5 - 5(3)^4\beta_3 - 5(4)\sum_{j=o}^{3} j^3/6\,\gamma_j)\, h^5 y^{V}(xn) \qquad (3.18)$$

Imposing accuracy of order 5 on T_{n+1}, we get

$$
\left.
\begin{aligned}
C_0 &= 0 \\
C_1 &= 1 - (1-1) - \beta_3 = 0 \\
C_2 &= \tfrac{1}{2}(3^2 - (3-1)^2 - 3(3)\beta_3 - 2(2-1)\sum_{j=0}^{3}\gamma_j = 0 \\
C_3 &= 1/6\,(3^3 - (3-1)^3 - 3(3^2)\beta_3 - 3(2)\sum_{j=0}^{3} j\,\gamma_j = 0 \\
C_4 &= 1/24\,(3^4 - (3-1)^4 - 4(3^3)\beta_3 - 4(3)\sum_{j=0}^{3} j^2/2!\,\gamma_j = 0 \\
C_5 &= 1/120\,(3^5 - (3-1)^5 - 5(3^4)\beta_3 - 5(4\sum_{j=0}^{3} j^3/3!\,\gamma_j = 0
\end{aligned}
\right\} \qquad (3.19)
$$

Solving the equations, we get

$$
\left.
\begin{aligned}
\gamma_0 &= -1/45 \\
\gamma_1 &= 13/120 \\
\gamma_2 &= -19/60 \\
\gamma_3 &= -97/360 \\
\beta_3 &= 1
\end{aligned}
\right\} \qquad (3.20)
$$

By substituting the parameters in (3.20) into equation (3.17), the three-step method of the form

$$y_{n+3} = y_{n+2} + h\,y'_{n+3} - h^2/360\,(8y''_n - 39y''_{n+1} + 114y''_{n+2} + 97y''_{n+3})$$

is generated and its order is 5

From the above calculations, it can be seen that the order of accuracy of the methods are k+2, where k is the step number.

CHAPTER FOUR

BASIC PROPERTIES OF THE NEW SCHEME

The basic properties of the new scheme are important aspect to consider in order to be sure that such methods are capable of solving non-stiff, stiff and stiff oscillatory ordinary differential equations for which it is intended. Some of the properties to be considered are: order of accuracy, consistency, zero-stability, convergence and A-stability.

ACCURACY

In numerical solution of ordinary differential equations it is necessary and important to estimate errors such that if the magnitude of the estimate is greater than some error tolerance then it can be known that the method is not accurate and the stepsize may be reduced for the iteration scheme to converge.

The errors can be as a result of the numerical approximation techniques or as a result of arithmetic operation adopted. They include truncation, discretization and round-off errors which arise from numerical approximation techniques and other computers devices.

Truncation error is defined as the amount by which the solution of the differential equation fails to satisfy the difference equation of the scheme. The truncation error T_{n+k} associated with the method is defined in equation (3.1)

$$T_{n+k} = y(x_{n+k}) - y(x_{n+k-1}) - h\beta_k y'(x_{n+k}) - h^2 \sum_{j=0}^{k} \gamma_j \, y''(x_{n+j})$$

By adopting the Taylor series expansion of $y(xn + k)$, $y^1(xn + k)$, $y^n(xn + j)$, $j = 0(1)k$ about X_n. Substituting these into error equation 3.1, the result obtained

$T_{n+k} = C_{p+1} h^{p+1} y^{p+1}(xn)$; $P \geq 2$

where $C_0 = C_1 = C_2 = \ldots = C_p = 0$

where P is the order of accuracy of the scheme.

While discretization errors are introduced as a result of the replacement of differential equation by its difference equivalence. The discretization error associated with the method (1.6.1) can be expresses as follows:

$$e_n = y(x_n) - y_n \quad \ldots \quad (4.1)$$

where the exact solution is $y(x_n)$ and the numerical approximation is y_n at point x_n

Round-off error is introduced as a result of arithmetic operations involving the computing devices. The magnitude of these errors depends on the way and manners machine operates or performed on the storage and manipulation of numbers. This is defined Mathematically by r_{n+k}

$$r_{n+k} = y_{n+k} - p_{n+k} \quad \ldots \quad (4.2)$$

where y_{n+k} is the exact solution, p_{n+k} is the value produced by the computer at the point $x = x_{n+k}$

4.1 ORDER OF ACCURACY OF THE METHOD AND ERROR CONSTANT

The order of accuracy of the method as explained in equation (3.7) is k + 2.

The error constant of the method is determined by considering the first of the non-vanishing coefficient C_ps of T_{n+k}. The C_{p+1} has absolute significance in the sense that its magnitude determines the degree of accuracy of the method. The smaller the magnitude of C_{p+1} the more accurate is the method. The error constant is often called the principal truncation error. The principal error is of the form

$$C_{p+1} = 1/p+1! \,(k^{p+1} - (k-1)^{p+1} - (p+1)\,\beta_k - p^{(p+1)} \sum_{j=0}^{k} j^{p-1}\gamma_j \quad - \qquad (4.3)$$

The error constant for the k-step method for k = 1 (1) 6 are shown in table 1.

4.2 CONSISTENCY

According to Gear (1972) and Lambert (1973) a linear multi-step method such as (1.17) is said to be consistent if it has order $P \geq 1$. Since the order P of accuracy of this method is

$P = k+2 > 1$ for all k $\quad 1 \leq k \leq 6$

then the method is consistent.

4.3 ZERO-STABILITY

In the spirit of Lambert (1991), the characteristics polynomial of the proposed method is

$$\pi(R, r) = r^k - r^{k-1} - \beta_k r^k - \sum_{j=o}^{k} \gamma_j r^j \quad - \quad - \quad - \quad - \quad (4.4)$$

where the first characteristic polynomial $\rho(r)$ of the method (1.17) is

$$\rho(r) = r^k - r^{k-1}$$

while the second characteristic of the method $\sigma(r)$ is defined as

$$\sigma(r) = \beta_k r^k - \sum_{j=o}^{k} \gamma_j r^j \quad - \quad - \quad - \quad - \quad - \quad (4.5)$$

Definition

The method (1.17) is said to be zero stable if no root of the first characteristic polynomial $\rho(r)$ has modulus greater than one and if every root with modulus one is simple. To investigate this for the method (1.17)

$$\rho(r) = r^k - r^{k-1}$$

The roots of $\rho(r) = 0$ can be obtained from

$$r^{k-1}(r-1) = 0$$

which simplifies into $r-1 = 0$ or $r^{k-1} = 0$ that is $r = 1$ or 0.

Since all the roots of $\rho(r) = 0$ is inside a unit circle then the method is zero-stable (Lambert, 1973).

4.4 CONVERGENCE

In Lambert (1973), a necessary and sufficient condition for a Linear multi-step methods to be convergent are that it must be consistent and zero stable. Since the method is consistent and zero-stable as proved above, then the method is convergent.

4.5 ABSOLUTE STABILITY

Even though Fatunla (1988),Fox and Mayers (1987) said that stability is a necessary and sufficient condition for the numerical solution to converge to the analytical solution, but if error is introduced at any stage of the computation it can produce unstable numerical results which can make the approximate solution deviate significantly from the true solution. This makes the analysis of the A-stability of the new method necessary. Hence the A-stability of the proposed formula is assessed by adopting the method to solve the scalar test, equation (1.18). Consequently, adopting (1.17) to solve (1.18), we have

$$y_{n+k} = y_{n+k-1} + h\beta_k \lambda y_{n+k} + h^2 \sum_{j=o}^{k} \lambda^2 \gamma_j y_{n+j}$$

$$y_{n+k}(1 - h\lambda\beta_k - h^2\lambda^2\gamma_k) = y_{n+k-1} + h^2\lambda^2 \sum_{j=o}^{k-1} \gamma_j y_{n+j} \qquad (4.6)$$

By setting

$$g(y_{n+k-1}) = y_{n+k-1} + h^2\lambda^2 \sum_{j=o}^{k-1} \gamma_j y_{n+j}$$

equation (4.6) becomes

$$y_{n+k} = \frac{1}{(1 - h\lambda\beta_k - h^2\lambda^2\gamma_k)} \; g(y_{n+k-1}) \qquad (4.7)$$

by setting $\bar{h} = h\lambda$ the k^{th} order difference equation (4.7) has a stable solution if the stability function

$$\mu(\bar{h}) = \frac{1}{1 - h\beta_k - \bar{h}^2\gamma_k} \quad - \quad - \quad - \quad - \quad - \quad (4.8)$$

Satisfies the inequality equation

$$-1 < \frac{1}{1 - h\beta_k - h^2\gamma k} < 1 \quad - \quad - \quad - \quad - \quad - \quad - \quad (4.5.4)$$

Simplifying we get

$$\bar{h} \in (-\infty, 0)$$

provided $\gamma k \neq 0$. That is, the method is A – stable (Lambert (1973)

CHAPTER FIVE

IMPLEMENTATION AND NUMERICAL RESULTS

5.1 IMPLEMENTATION

The non-linear equations

$$G(y_{n+k}) = 0 \qquad - - - - - - - - - - - - \quad (5.1)$$

generated by the application of (1.17) to solve eq. (1.1) will be solved at each step by an iterative scheme of the form (1.20). The iteration matrix $w_{n+k}^{(s)}$, the coefficient β_k, γ_k varies from iteration to iteration due to change in step size h. Different iteration schemes results from different choices of $W_{n+k}^{(s)}$. The cost of the linear algebra that will be associated with large systems is high and can be reduced if we use the iteration scheme which adopts the value of the Jacobian of G approximated at the previous iteratives, that is

$$W_{n+k}^{(S+1)} = G^{1}(y_{n+k}^{(S)}). \quad s = 0(1)m$$

where m is the number of iterations.

5.1.1 COMPUTER PROGRAM

The program that implements this iteraction is written in fortran programming language and implemented on a digital computer as discussed below. In order to achieve the above objective, we adopt the following steps:

i. re- write the formula in algorithm form

ii. translate algorithm into a flow chart

iii. translate the flow chart into computer code

iv. implement the code with simple problems on a digital computer

v. discuss the results

5.1.2 ALGORITHM

An algorithm is defined as a step by step procedure taken to obtain solution to a particular problem. The algorithm for implementing the method described in chapter three especially formula (3.8) is as follows:

Step 1: Start

2: Declare all necessary variables

3: Initialise variables ($x = 0$, $y = 1$, $h = 0.1$, $n = 0$)

4: Compute y^1

5: Compute k_1, k_2, k_3, using Runge-Kutta method of order three

6: Compute $y_{n+1} = y_n + \frac{h}{4}(k_1 + 3k_3)$

7: Compute $y_{n+1}^{(n+1)} = y_{n+1}^{(n)} - \frac{f(y_{n+1})_{ns}}{f^1(y_{n+1})}$

8: Compute $f(y_{n+1}^{(n)})$

9: Compute $y_{n+1}^{(n-1)}$

10: Test if $(Y_{n+1}^{(n)}) > 0.0001$ then go to step 6 otherwise 11

11: Display results

12: Increment $x_{n+1} = x + h$ assign $y_{n+1}^{(n-1)}$ to $y_{n+1}^{(n)}$

13: Test if $x < 10$ then go to step 4 else 14

14: Stop

15: End.

5.1.3 FLOW CHART

A Flow chart is a diagram that shows the sequences of events in a program. The flow chart for the above algorithm is shown in figure 2 below.

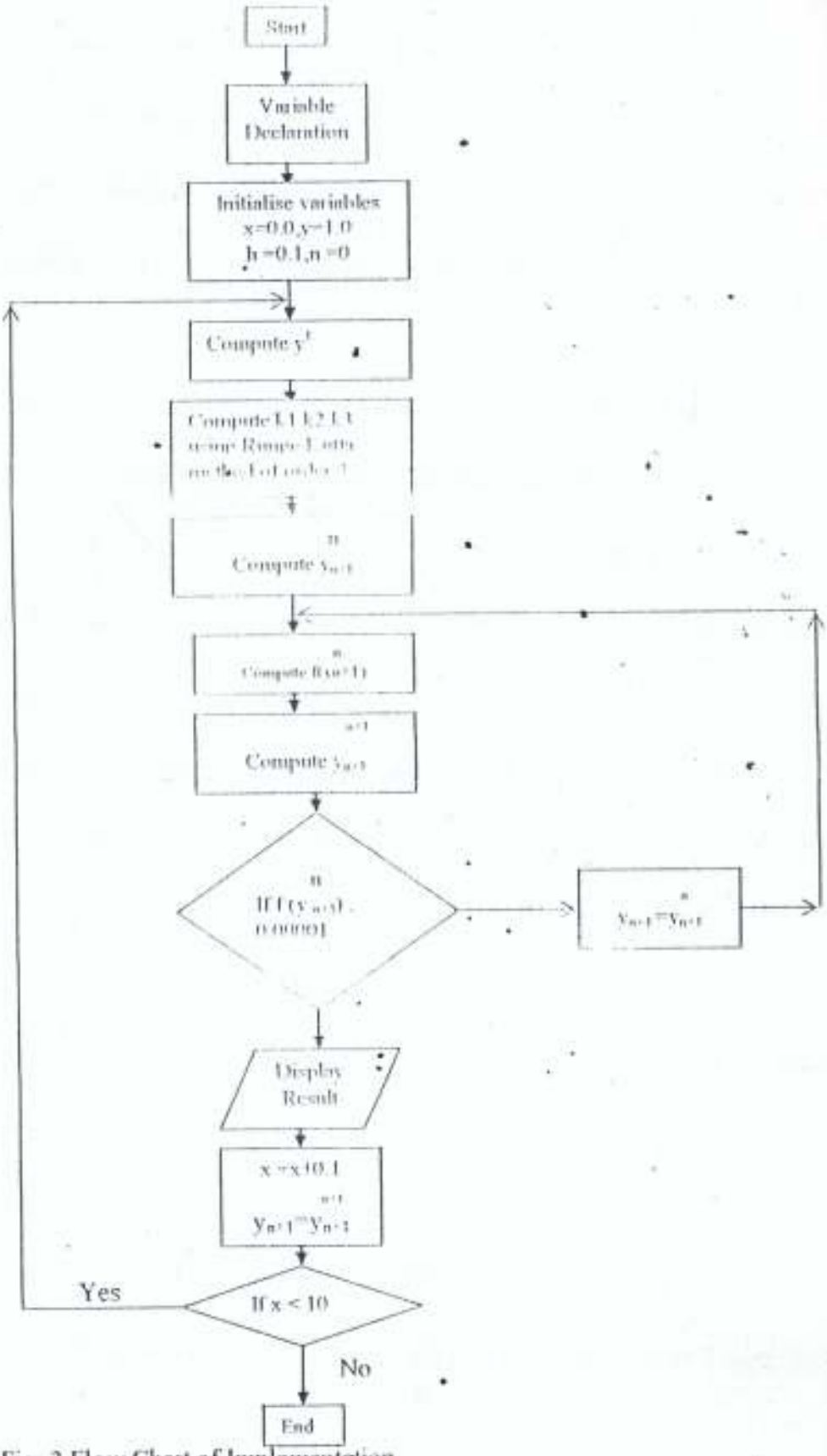


Fig. 2 Flow Chart of Implementation

5.2 The steps adopted for terminating the iteration scheme

The criteria for stopping the iteration is the closeness between the current and previous iterates. One of the methods discussed in Hall and Watt (1976) adopts the m^{th} iterate $(y_{n+k}^{(m)})$ as more accurate than $y_{n+k}^{(m-1)}$ the amount of correction

$y_{n+k}^{(m)} - y_{n+k}^{(m-1)}$ is used as estimate of the local error in $y_{n+k}^{(m-1)}$. The iteration is stopped if the magnitude of the error is smaller than the error threshold, that is

$$\left| y_{n+k}^{(m)} - y_{n+k}^{(m-1)} \right| \leq \text{Tol} \quad - \quad - \quad - \quad - \quad - \quad - \quad (5.1)$$

The current iteration $y_{n+k}^{(m-1)}$ is considered accurate and $y_{n+k}^{(m)}$ is regarded to as the best approximations to the exact solution y_{n+k} of equation (1.1). It is considered to adopt an iteration stopping criterion that takes effects of errors in the derivatives into account. The reason is because the formula suitable for stiff systems always involve past and present values of the derivatives and the errors associated with these past values may affect the stability of the computed solution (Ademiluyi, 1987). To see this, consider the derivative error $y^1(y_{n+k}^{(m)}) - y^1(y_{n+k}^*)$. By the mean value theorem, we have

$$y^1(y_{n+k}^{(m)}) - y^1(y_{n+k}^*) = J(y_{n+k}^*)(y_{n+k}^{(m)} - y_{n+k}^*) \quad - \quad - \quad (5.2)$$

where $y^1(y_{n+k}^{(m)})$ and $y^1(y_{n+k}^*)$ are the derivatives of y and $J(y_{n+k}^*)$ is the Jacobian of the differential system of equation on the line between $y_{n+k}^{(m)}$ and y_{n+k}^*.

The accuracy in the derivative values was ignored in most early codes for stiff ordinary differential equations where the closeness of $y_{n+k}^{(m)}$ to the exact solution y_{n+k}^{*} was only considered and neglected the closeness of $y^1(y_{n+k}^{(m)})$ to the $y^1(y_{n+k}^{*})$. the closeness of $y_{n+k}^{(m)}$ to y_{n+k}^{*} does not indicate the closeness of their derivatives. This assumption may affect, the stability of the computed solution in stiff ODEs. Since the eigen values of J is expected to have large modulus and the small $\left|y_{n+k}^{(m)} - y_{n+k}^{*}\right|$ does not implies small

$$\left|y^1 y_{n+k}^{(m)}) - y^1(y_{n+k}^{*})\right|$$

The residual error test as proposed by Shampine (1975), takes care of this type of error estimate. The approach is independent of the form of iteration used and it involves the current iterate and the norm of the residual error $r^{(m)}$ is defined as

$$\left|r^{(m)}\right| = \left|y_{n+k}^{(m)} - h\beta_k y^1(y_{n+k}^{(m)}) - h^2\gamma_k y^1(y_{n+k}^{(m)}) J^{(m)} dy - g\right| \quad - \quad - \quad - \quad (5.3)$$

where g denotes the function that combines both past values of y and y^1.

If $r^{(m)}$ is sufficiently small, the current iterate can be accepted as the most accurate approximation to the exact solution (1.17). For clarity purpose, let $e_{n+k}^{(m)}$ be the difference between the most accurate and the exact solution that is,

$$e_{n+k}^{(m)} = y_{n+k}^{(m)} - y_{n+k}^{*}$$

If y_{n+k}^{*} is a solution of (1.17) and satisfies

$$y_{n+k}^{*} - h\beta_k y^1(y_{n+k}^{*}) - h^2\gamma k\ J(y_{n+k}^{*}) y^1(y_{n+k}^{*}) - g = 0 \quad (5.4)$$

likewise $y^{(m)}$ satisfies

$$y_{n+k} - h\beta_k y^1(y_{n+k}^{(m)}) - h^2\gamma_k J(y_{n+k}^{(m)}) - y^1(y_{n+k}^{(m)}) - g = 0 \qquad (5.5)$$

by subtracting (5.5) from (5.4) we have

$$r_{n+k}^{(m)} = y_{n+k}^{(m)} - y_{n+k}^{*} - h\beta_k (y^1(y_{n+k}^{(m)}) - y^1(y_{n+k}^{*}) - h^2\gamma_k J(y^1(y_{n+k}^{(m)}) - y^1(y_{n+k}^{*}) - \qquad (5.6)$$

where $J = \partial f/\partial y \ (y_{n+k}^{*})$

by applying the mean value theorem to equation (5.6)

$$r_{n+k}^{(m)} = e_{n+k}^{(m)} - h\beta_k Je_{n+k}^{(m)} - h^2\gamma_k J^2 e_{n+k}^{(m)}$$

$$r_{n+k}^{(m)} = (I - h\beta_k J - h^2\gamma_k J^2) e_{n+k}^{(m)} - \quad - \quad - \qquad (5.7)$$

where I is an m by m identity matrix.

For a stiff system of ordinary differential equation, $J(y_{n+k}^{*})$ is expected to have at least one eigen-value with large modulus. That is

$$\left| (I - h\beta_k J - h^2\gamma_k J^2) \right| \geq 1 \quad - \quad - \quad - \quad - \quad - \qquad (5.8)$$

Hence

$$\left| r_{n+k}^{(m)} \right| < \left| e_{n+k}^{(m)} \right|, \qquad m = 1, 2, \quad - \quad - \quad - \quad - \qquad (5.9)$$

In solving the iteration equation (1.20) for stiff system, the norm of the residual is expected to be larger than the norm of the error vector. If the norm of the residual is found to be small than error tolerance then error in the current iterate is small. Since the test involves both the current iterate and the corresponding values, the residual error test can serve as the error estimate.

5.3 Step-Size Control

Step size h can be seen as a catalyst for improving the efficiency of the scheme. So step size adjustment will play an important role in the integration of ODEs. Step size control involves the step by step computation of the local truncation error identified with the integrations scheme. The manner in which the integration formula is represented dictates the form of error estimates to use. There are many forms of error estimate. All these estimates are based on the idea that the current integration scheme can be represented by incorporating more analytical property of the equation into the formula will provide a better result (See Sedgwick (1973) for more detail).

One error estimate considered adequate for stiff equation explain in this work is based on the Richardson's extrapolation process.

The local error e_n for stepping from x_n to x_{n+1} is written as

$$y(x_{n+1}) - y_{n+1} = \phi(x_n, y(x_n))h^{p+1} + O(h^{p+2}) \qquad (5.9)$$

where p is the order and y_{n+1} is the numerical approximation to y at x_{n+1}.

By applying the proposed scheme with another step size h/2 and compute an approximation y_{n+1}^* to $y(x_{n+1})$ with the same local assumption can be defined as

$$y(x_{n+1}) - y_{n+1}^* = \phi(x(x_n, y(x_n))(h/2)^{p+1} + O(h^{p+2}) \qquad (5.10)$$

subtracting equation (5.10) from (5.9), we have

$$y_{n+1} - y_{n+1}^* = \phi(x_n, y(x_n))h^{p+1}(1 - (1/2)^{p+1}) \qquad (5.11)$$

Thus, by adopting the Richardson's extrapolation we compute two estimates over two steps using step size h/2 and another using step size h. The difference between the two approximations multiplied by $(2^{p+1}-1)^{-1}$ gives the error estimate for the scheme. This form of error estimate involves a considerable amount of computational efforts.

The numerical approximation to the solution from step x_n to step x_{n+1} is acceptable if the norm of the error estimate e_{n+1} is less than the error tolerance.

$$\left|e_{n+1}\right| \leq E_p s \qquad (5.12)$$

The estimate can now be used to compute the next integration step size.

The Ways of Varying The Step Size

Although, there are many ways of varying the step size, one of the method is described as follows, an initial step size h which is based on the characteristic behaviour of the function f is determined, since the function changes very rapidly for stiff ordinary differential equations the value of h to be chosen must be in such a way that the product $h\left|\partial f/\partial y\right|$ shall lie within the stability region of the method or ensure the value of h is

$$h = \min_{ij} \left|\frac{1}{\frac{\partial f_i}{\partial y_j}}\right| \quad i,j = 1(1)m \qquad (5.13)$$

If the starting step size h is sufficiently large, the past values are reliable and the past values are generated with minimum error. If the method fails to start, it

indicates that the error tolerance is too small and the program implementing the scheme will automatically adjust, h through the formula.

$$h_{new} = \left| \frac{E_1 s}{E_{n+1}} \right|^{1/p+1} \qquad (5.14)$$

The step size will be allowed to increase only if

$$|E_{n+1}| \leq |E_{n+1-1}| \qquad (5.15)$$

As soon as the program approaches the end point of the integration interval the step size will be chosen so that the last point coincides with the end point.

5.4 NUMERICAL EXAMPLES

In order to assess the suitability and accuracy of the method we adopt the following one-step formula.

$$y_{n+1} = y_n + h\,y'_{n+1} - 1/6h^2\,(y''_n + 2y''_{n+1})$$

and two-step formula

$$y_{n+2} = y_{n+1} + y'_{n+2} + h^2/24\,(y''_n - 6y''_{n+1} - 7y''_{n+2})$$

to solve the sample problems

Example 1

$y' = x + y \qquad y(o) = 1$

exact solution $\qquad y(x) = 2e^x - x - 1$

Example 2

$y' = -100y, \qquad y(o) = 1$

exact solution $= y(x) = e^{-100x}$

Example 3

$y^1 = 1 + y^2$; $y(o) = 1$

exact solution: $y(x) = \tan(x + \pi/4)$

Taking from Fatunla (1988), Lambert (1973) and Okosun (2002): The results are as shown in the tables 1 – 7

DISCUSSION OF THE RESULTS

It can be seen from the results in table 1-7 that the local error e_n of the methods are small. The comparison of the proposed method with the existing method and the graphical representation show that the proposed method gives better results. All the results proved that the new method has a high order of accuracy.

Table 2: Performance of One-step second derivative formula on problem 1

		Theoretical solution	Numerical solution	Error
n	x_n	$y(x_n)$	y_n	e_n
0	0.00	1.0000000000	1.0000000000	0.0000000000
1	.10	1.1103419289	1.1103321051	.0000098238
2	.20	1.2428055257	1.2427840078	.0000215179
3	.30	1.3997177556	1.3996819598	.0000357958
4	.40	1.5836494789	1.5835968564	.0000526224
5	.50	1.7974424288	1.7973699616	.0000724671
6	.60	2.0442377552	2.0441413453	.0000964099
7	.70	2.3275053725	2.3273813093	.0001240632
8	.80	2.6510821804	2.6509251103	.0001570702
9	.90	3.0192065462	3.0190113137	.0001952025
10	1.00	3.4365643486	3.4363213615	.0002429872

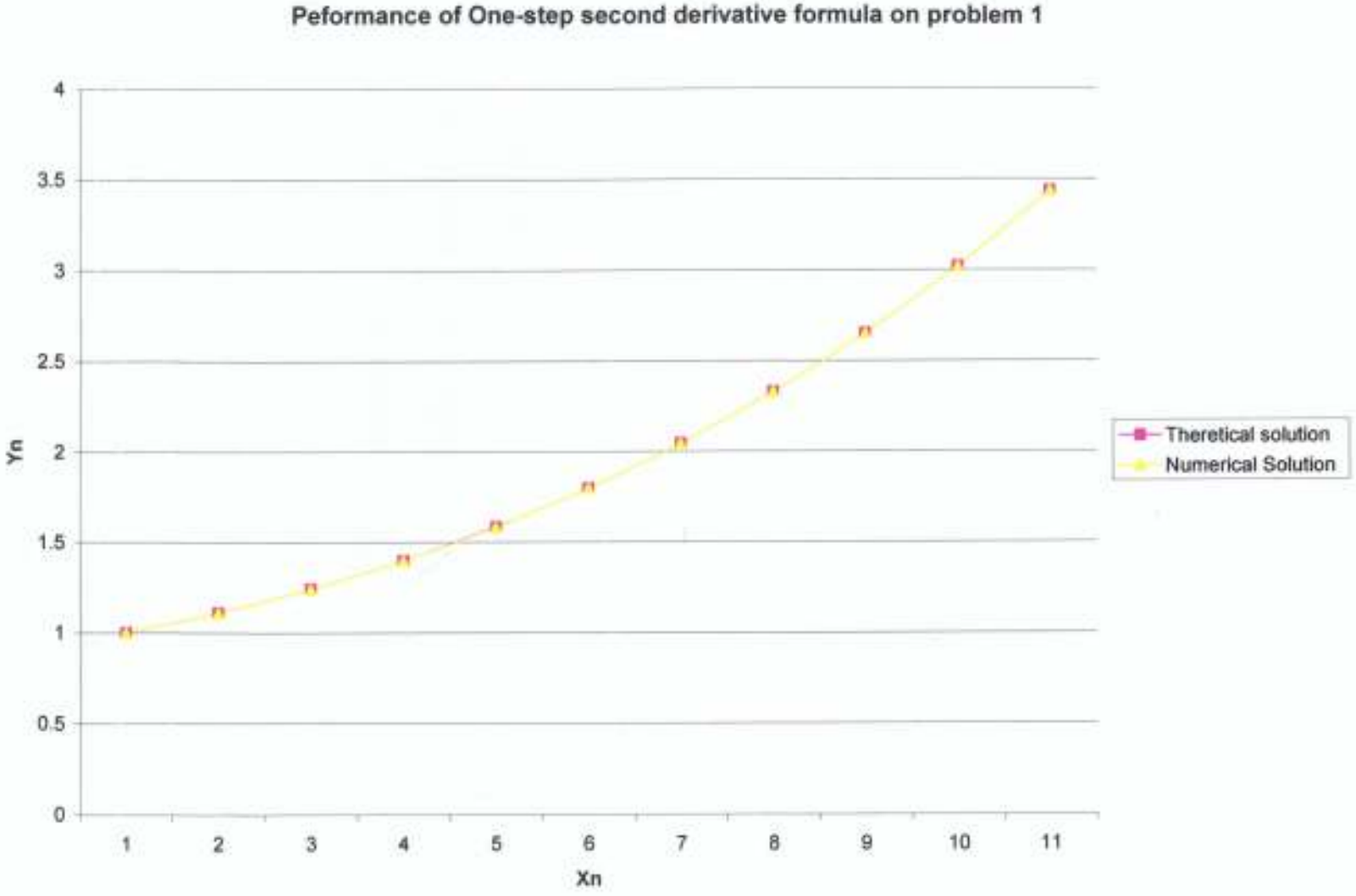

Peformance of One-step second derivative formula on problem 1
4
3.5
3
2.5
2
1.5
1
0.5
0
Yn
1
2
3
4
5
6
7
8
9
10
11
Xn
Theretical solution
Numerical Solution

Table 3: Performance of 2-step SDF on Problem 1

		Theoretical solution	Numerical solution	Error
n	x_n	$y(x_n)$	y_n	e_n
0	0.00	1.0000000000	1.0000000000	0.0000000000
1	.10	1.1103419289	1.1103416685	.0000002604
2	.20	1.2428055257	1.2428047805	.0000007452
3	.30	1.3997177556	1.3997161940	.0000015616
4	.40	1.5836494789	1.5836471538	.0000023251
5	.50	1.7974424288	1.7974393222	.0000031066
6	.60	2.0442377552	2.0442332224	.0000045328
7	.70	2.3275053725	2.3274996719	.0000057006
8	.80	2.6510821804	2.6510745130	.0000076674
9	.90	3.0192065462	3.0191970064	.0000095399
10	1.00	3.4365643486	3.4365522583	.0000120903

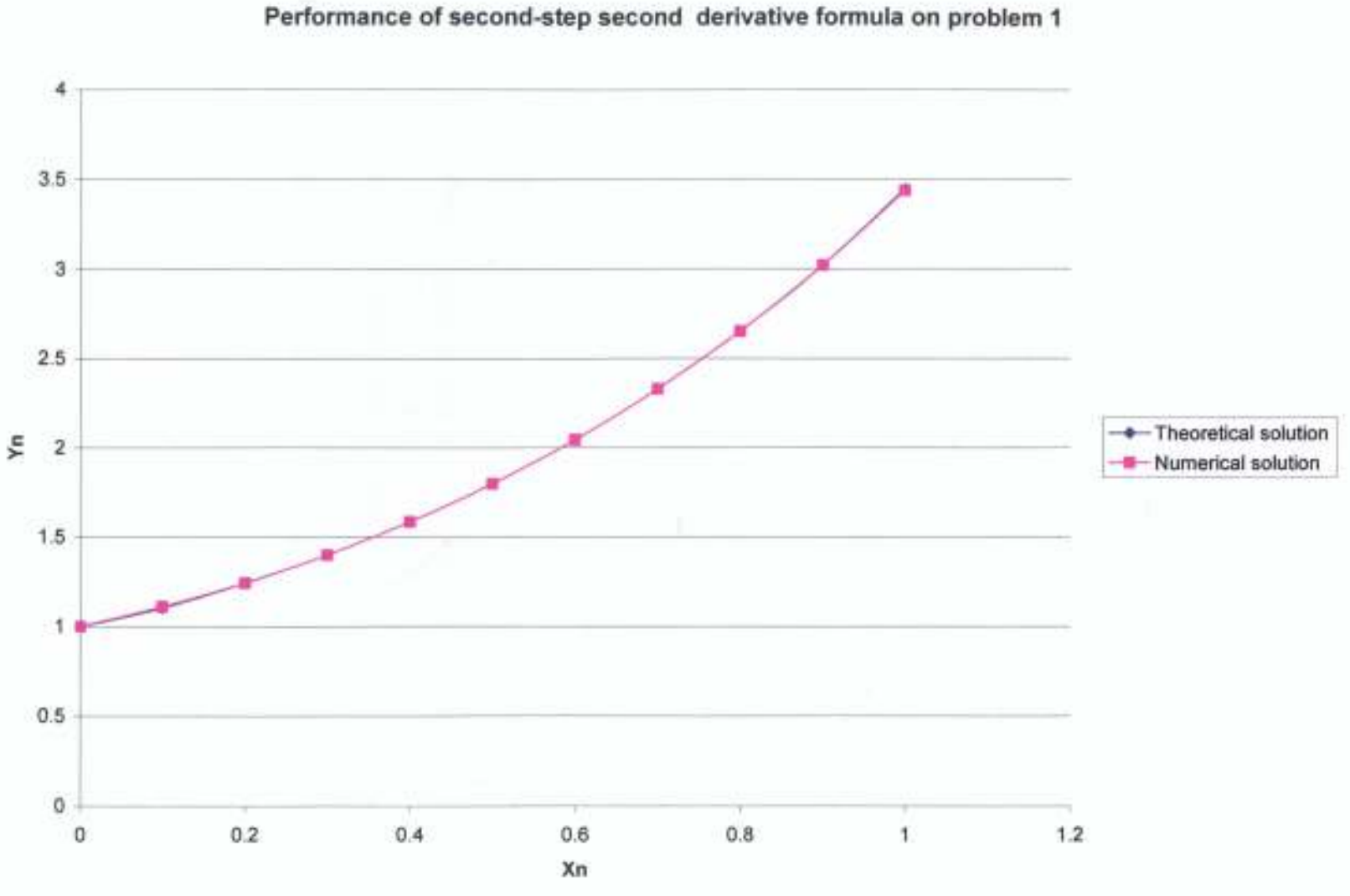

Performance of second-step second derivative formula on problem 1
4
3.5
3
2.5
2
1.5
1
0.5
0
Yn
0
0.2
0.4
0.6
0.8
1
1.2
Xn
Theoretical solution
Numerical solution

Table 4: Performance of one-step Second Derivative Formula and Backward Difference Formula on problem 1

	Theoretical solution	BDF solution	SDF solution	SDF error	BDF error
x_n	$y(x_n)$	yB_n	yS_n	eS_n	eB_n
0.00	1.0000000000	1.0000000000	1.0000000000	0.000000000	0.000000000
.10	1.1103419289	1.1222222247	1.1103321051	.0000098238	.0118802958
.20	1.2428055257	1.2559245643	1.2427840078	.0000215179	.0131190387
.30	1.3997177556	1.4142044568	1.3996819598	.0000357958	.0144867012
.40	1.5836494789	1.5996466272	1.5835968564	.0000526224	.0159971483
.50	1.7974424288	1.8151076234	1.7973699616	.0000724671	.0176651946
.60	2.0442377552	2.0637444069	2.0441413453	.0000964099	.0195066516
.70	2.3275053725	2.3490459476	2.3273813093	.0001240632	.0215405752
.80	2.6510821804	2.6748681333	2.6509251103	.0001570702	.0237859528
.90	3.0192065462	3.0454723601	3.0190113437	.0002952025	.0262658138
1.00	3.4365643486	3.4655681786	3.4363243615	.0002399872	.0290038299

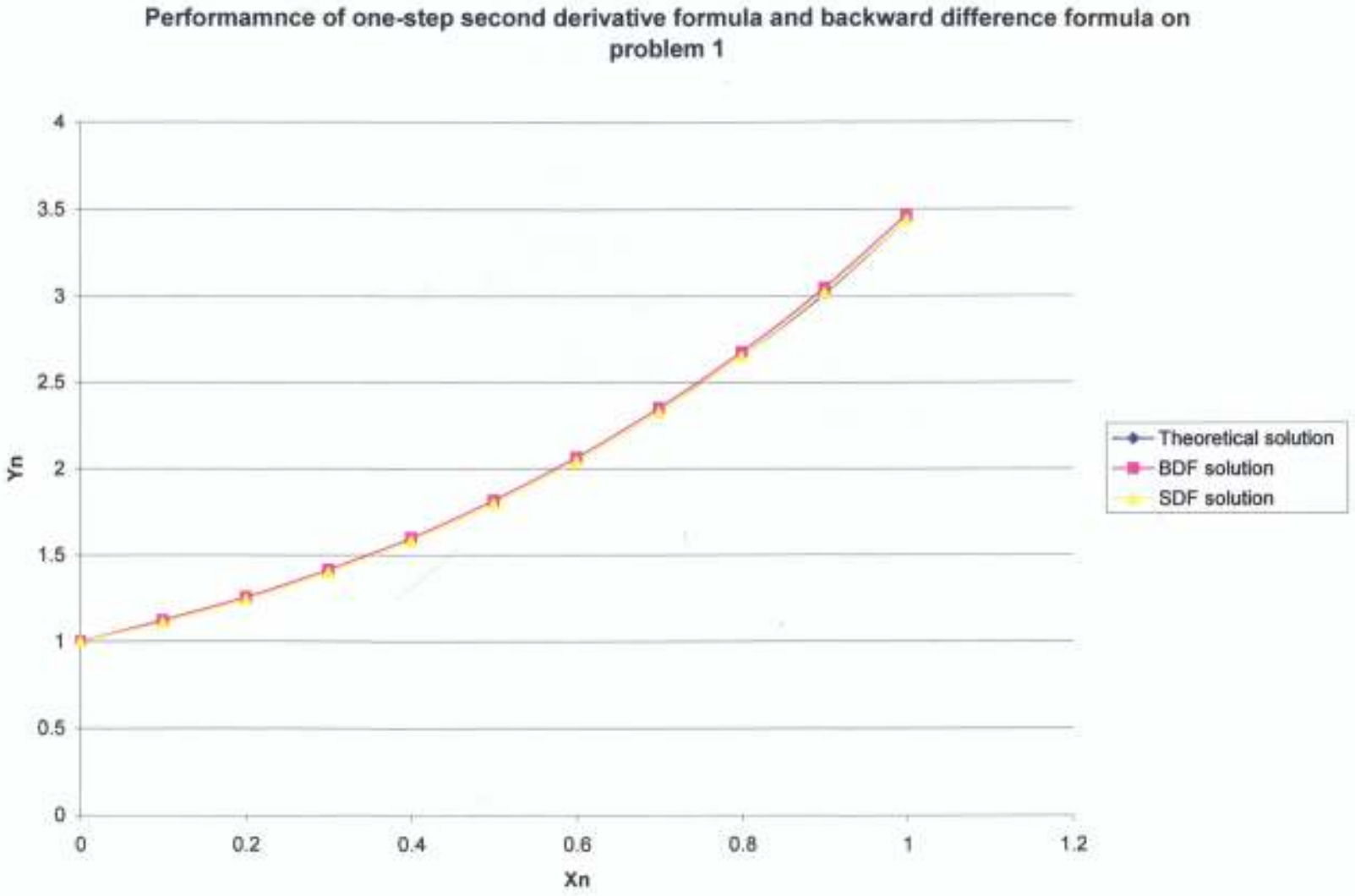

Performamnce of one-step second derivative formula and backward difference formula on problem 1
4
3.5
3
2.5
2
1.5
1
0.5
0
Yn
0
0.2
0.4
0.6
0.8
1
1.2
Xn
Theoretical solution
BDF solution
SDF solution

Table 5: Performance Of Two-Step Second Derivative Formula And Backward Difference Formula On Problem 1.

Xn	Theoretical solution	BDF solution	SDF solution	SDF error	BDF Error
Xn	y(Xn)	yBn	ySn	eSn	eBn
0.00	1.0000000000	1.0000000000	1.0000000000	0.000000000	0.000000000
.10	1.1103419289	1.1103416685	1.1103416685	.0000002605	.0000002604
.20	1.2428055257	1.2433452424	1.2428047805	.0000007452	.0005397168
.30	1.3997177556	1.4010854668	1.3997161940	.0000015616	.0013677112
.40	1.5836494789	1.5862632484	1.5836471538	.0000023251	.0026137695
.50	1.7974424288	1.8019060028	1.7974393222	.0000031066	.0044635740
.60	2.0442377552	2.0514203091	2.0442332224	.0000045328	.0071825538
.70	2.3275053725	2.3386578315	2.3274996719	.0000057006	.0111524590
.80	2.6510821804	2.6679993298	2.6510745130	.0000076674	.0169171493
.90	3.0192065462	3.0444634547	3.0191970064	.0000095399	.0252569085
1.00	3.4365643486	3.4738497635	3.4365522583	.0000120903	.0372854149

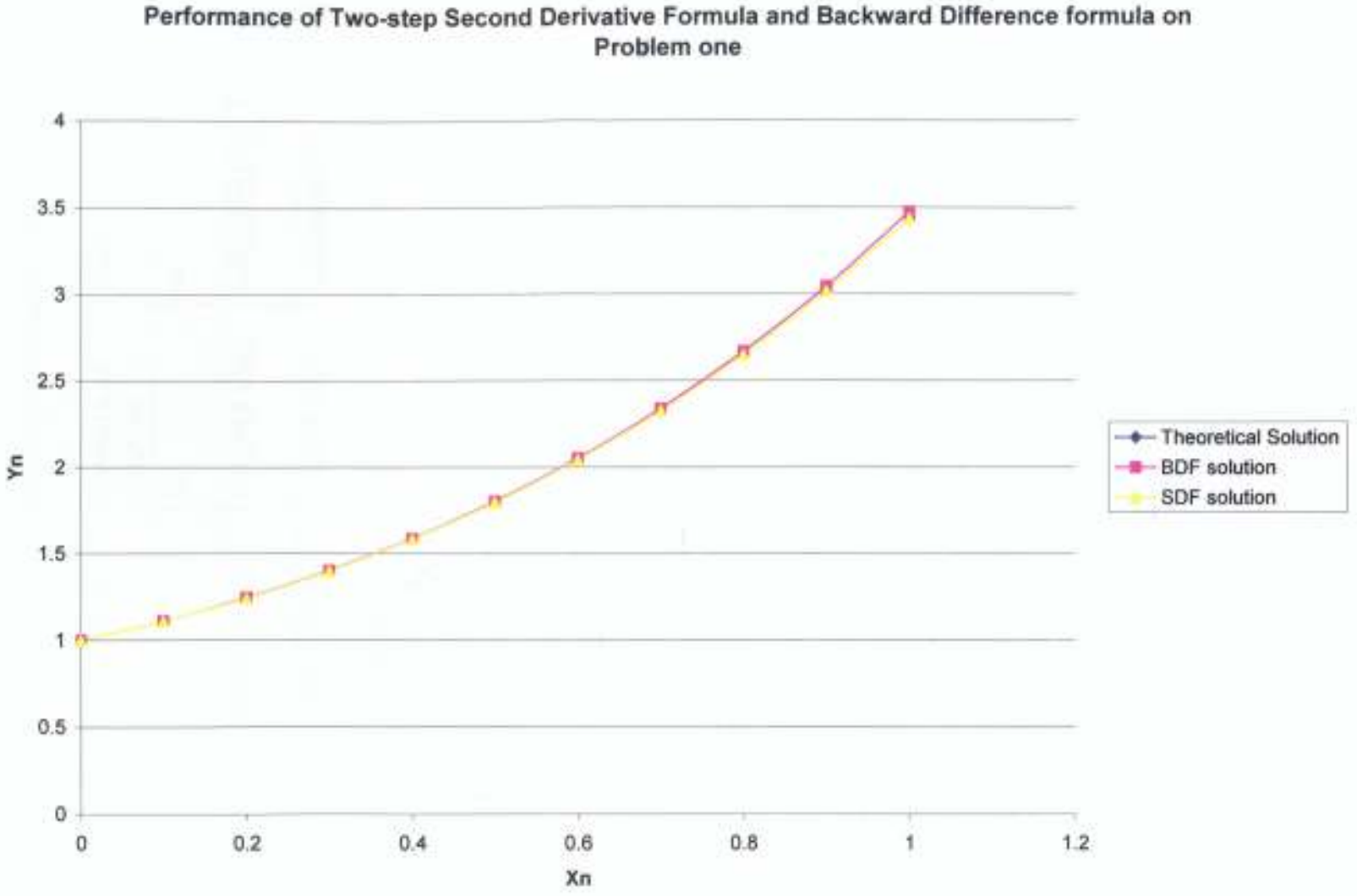
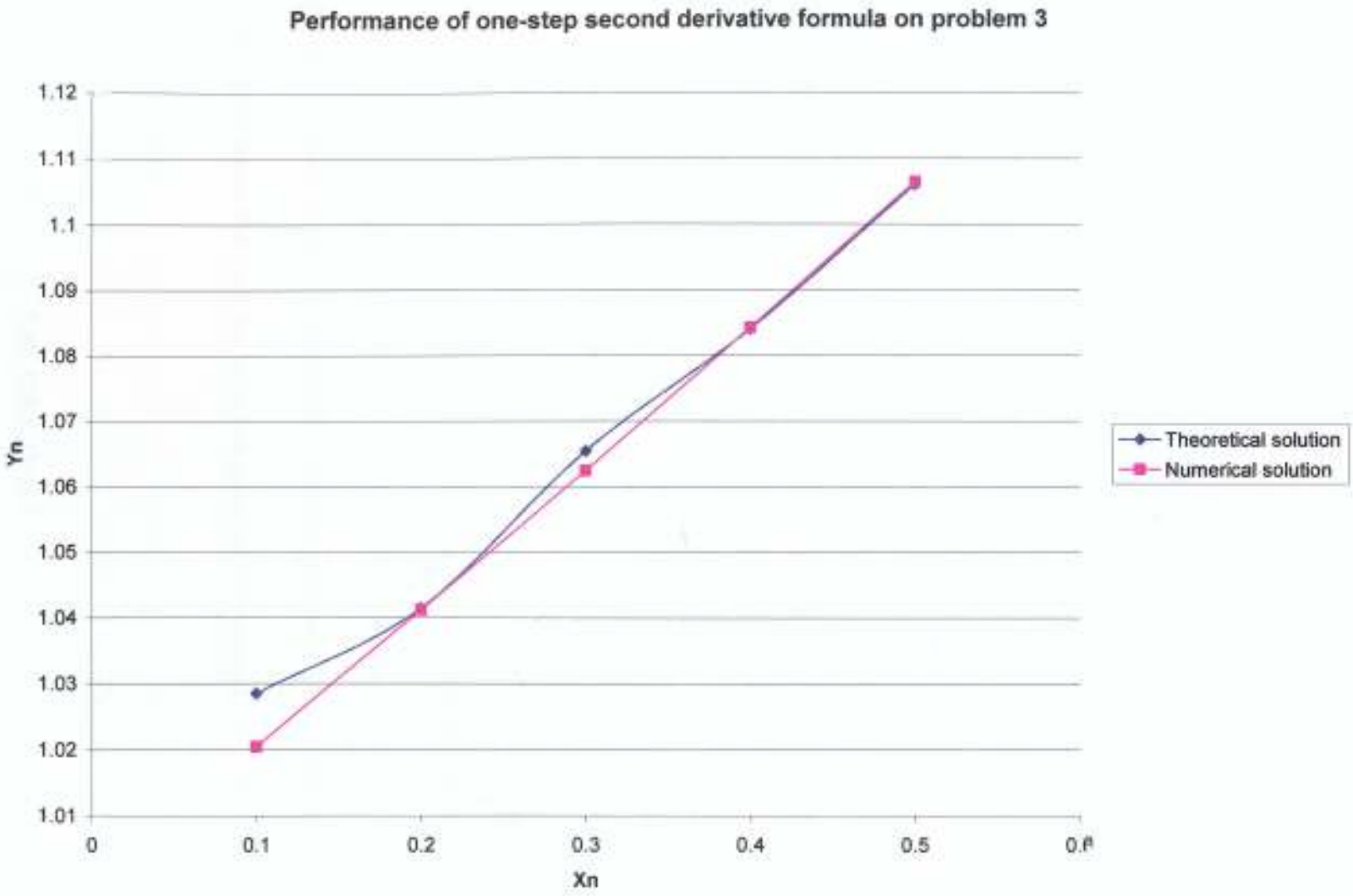

Performance of Two-step Second Derivative Formula and Backward Difference formula on Problem one
Yn
Xn
0
0.5
1
1.5
2
2.5
3
3.5
4
0
0.2
0.4
0.6
0.8
1
1.2
Theoretical Solution
BDF solution
SDF solution

Table 6: Performance of one-step second derivative formula on problem 2

	Theoretical solution	Numerical solution	Error
x_n	$y(x_n)$	y_n	e_n
0.000625	0.939412485	0.939413062	0.000000577
0.00125	0.88248848	0.882496902	0.000008422
0.0025	0.778688525	0.778800783	0.000112258
0.005	0.605263158	0.60530659	0.0012675501
0.01	0.357142857	0.637879441	0.010736584

Table 7: Performance of one - step second derivative formula on problem 3

	Theoretical solution	Numerical solution	Error
x_n	$y(x_n)$	y_n	e_n
0.10	1.0208480201	1.0204081628	0.0004398573
0.20	1.0414806419	1.0412371554	0.0002434864
0.30	1.065475709	1.0625045999	0.0000429710
0.40	1.0840670855	1.0842290474	0.0001620320
0.50	1.1060581556	1.1064300457	0.0003718901
0.60	1.1285412225	1.1291282059	0.0005869833
0.70	1.1515375604	1.1523452801	0.0008077198
0.80	1.1750697089	1.1761042410	0.0001034521
0.90	1.1991614904	1.2004293688	0.0012678784
1.00	1.2238381030	1.2253463537	0.0015082507


Performance of one-step second derivative formula on problem 3
1.12
1.11
1.1
1.09
1.08
1.07
1.06
1.05
1.04
1.03
1.02
1.01
Yn
0
0.1
0.2
0.3
0.4
0.5
Xn
Theoretical solution
Numerical solution

CHAPTER SIX

CONCLUSION

6.1 SUMMARY

A class of numerical method suitable for solving both stiff, non-stiff and stiff oscillatory ordinary differential equations is developed. The method is motivated by the second derivative method by Enright (1972). The basic properties of the method were investigated and found to be accurate, consistent, zero-stable, convergent, stable and of order k+2. Computer programme was developed and implemented with sample problems. The results show that the proposed method is accurate and compares favourably with the existing method.

6.2 Limitation

The method developed is an implicit method so there is a need for predictor to provide the starting values at each step of the application of method. The predictor must be of the same order with the method to avoid data error and as well has a better approximation. The scheme involves linear algebra, which provide analytical Jacobian matrix, and these requirements is expensive in terms of man-hour and large storage facilities.

6.3 RECOMMENDATION

The limitations of the method are overcomed with the appropriate use of the competent predictor. Better accuracy can be achieved by the use of double precision arithmetic to minimize the effects of round-off errors. If all the necessary precautions are taken into consideration the methods are recommended for solving non-stiff, stiff and stiff /.oscillatory initial value problems in ordinary differential equations.

6.4 Contribution to knowledge

The new schemes will be adequate for solution of non-stiff, stiff and stiff oscillatory equations.

REFERENCES

1. Ademiluyi, R.A (1987): " New hybrid methods for system of stiff ordinary differential equation" Ph.D Thesis University of Benin, Benin City Nigeria.
2. Butcher, J.C (2001): "General linear methods for stiff differential equations" BIT 41, no. 2,240 – 264, MathSciNet.
3. Byne G.D and Hindmarsh, A.C., (1975) "Episode. An Experimental package for the integration of the systems of ODEs" Report UCID – 30112, Lawrence Livermore lab, University of California Livermore.
4. Cao, Xue-nian; Li, Shou-fu; Liu, De-gui (2002): "modified parallel Rosenbrock methods for stiff differential equations" J.Comput. Math. 20, no.1,23—34, MathSciNet
5. Coddington and Levinson (1955): "Theory of Ordinary different equation" McGraw-Hill New York
6. Dahlquist, G (1969): "A Numerical Method for some ODEs with large Lipschitz & Constants." Information processing 68, North Holland Publishing Company, Amsterdan
7. Enright, W.H. (1972): "Studies in the Numerical solutions of stiff ODEs" Ph. D. Thesis, University of Toronto, Computer Science Report 42.
8. Fatunla S.O. (1988): "Numerical methods for Initial value problem in Ordinary differential equations" Academic Press Inc. Harcourt Brace, Jovanvioh, Publishers London, New York.
9. Finizio N. and Ladas, G. (1970): "Ordinary Differential Equations with modern applications" Wadsworth Publishing Company, Inc. Balmont, California.

10. Fox, I. and Mayers, D.F (1987): "Numerical Solution of Ordinary Differential Equations." Published in the USA by Oxford University Chapman and Hall West 35th Street New York NY10001.

11. Gear, C.W (1971): "Numerical Initial Value Problems In Ordinary Differential Equations." Prentice Hull, New Jersey.

12. Hall, G. and Watt, J.M.(1976): "Modern Numerical Methods for ordinary differential equations" Clarendon Press Oxford.

13. Hindmarsh, A.C.(1974): "Gear :ODEs system Solver" Revision 3, Report UCID –30001,Lawrence Livermore Laboratory, University of California, Livermore.

14. Hull ,T.E. Enright, W.H, Fellen, B.M. and Segwick, A.E. (1972): "Comparing numerical methods for ODEs" SIAMJ Numer. Anal. 9,603-607

15. Lambert J.D. (1973), "Computation Method for Ordinary Differential Equations." John Willey and Sons Inc. New York

16. Lambert, J.D.(1980): "Stiffness". Proceedings Computational Techniques for ordinary differential equations (Gladwell I and Sayers D.K. ed.) 19-46; Academic press a subsidiary of Harcourt Brace Jovanovich Publishers London , New York Toronto Sydney San Francisco.

17. Lambert,J.D. (1991): " Numerical methods for ordinary differential system of initial value problems". John Willey and Sons Inc. New York.

18. Sedgwick , A.E.(1973): "An effective variable order variable stepsize Adams methods" , Ph.D Dissertation Technical Report 53, Department of Computer Science , University of Toronto, Canada.

19. Shampine,L.F.(1975): local error control in codes for ordinary differential equations"APP1.Math.Compt. 3 . 189-210

20. Okosun, K.O. (2002), "K^{th} order inverse polynomial method for the integration of Ordinary Differential Equations with singularities." M.Tech.

```
double precision k1,k2,k3,y,h
      double precision yprime,ynplus1,fprimey,dprimey,dprmprmy
      double precision c,cc,ccc,fy,rnewy,g
      double precision ynpr me

      OPEN(UNIT=3,FILE='RESULT1.DAT',STATUS='NEW')
      OPEN(UNIT=4,FILE='RESULT.DAT',STATUS='NEW')

      WRITE (3,12)
12    FORMAT (2x,'X',8x,'Exact Sol.',3x,'Numerical Sol',10x,'Error')

      WRITE (3,13)
13    FORMAT (10x)
      WRITE (3,13)

      x=0.0
      y=1.0
      h=0.1
      kk=1

      fprimey=0.90333333333

5     yprime=x+y

      k1=x+y
      k2=(x+(1.0/3.0*h)+y+(1.0/3.0*h*k1))
      k3=(x+(2.0/3.0*h)+y+(2.0/3.0*h*k2))

      k=1

      Ynplus1=y+h/4.0*(k1+3*k3)

15    dprimey=Ynplus1+(x+0.1)
      dprmprmy=dprimey+1
      yprime=1+(x+y)
      c=Ynplus1-y
      cc=(0.1*(dprimey))

      ccc=((h*h)/6.0)*((yprime)+2*(dprmprmy))
      fy=c-cc+ccc

      rNewy=ynplus1-(fy/fprimey)
       nn=x+0.1

      g=2*exp(x+0.1)-(x+0.1)-1

      if (fy.gt.0.00001) then
      Ynplus1=rnewy
      goto 15
      end if

*     write(4,11)Ynplus1,y,dprimey,yprime,dprmprmy
11    format(f14.10,2X,f14.10,2X,f14.10,2X,f14.10,2X,f14.10)

10    format(f4.2,5x,f14.10,5x,f14.10,5x,f14.10)
```

```
write(3,10) x+0.1,g, rnewy,g-rnewy

WRITE (3,13)
WRITE (3,13)

x=x+0.1
y=rnewy

if (x.lt.1.0) then
kk=kk+1
goto 5
end if

stop
end
```

Program of one- step second derivative formula on problem 1

```
double precision k1,k2,k3,y,h
      double precision yprime,ynplus1,fprimey,dprimey,dprmprmy
      double precision c,cc,ccc,fy,rnewy,r
      double precision ynprime
      double precision secfprimey
      double precision secfy
      double precision secrnewy

      OPEN(UNIT=3,FILE='RESULT1b.DAT',STATUS='NEW')
      OPEN(UNIT=4,FILE='RESULTb.DAT',STATUS='NEW')

      WRITE (3,12)
 12   FORMAT (2x,'X',8x,'Exact Sol.',10x,'BDF',15x,'Proposed Sol',8x,
     1 'Proposed Error',5x,'BDF Error')

      WRITE (3,13)
 13   format(2x)
      WRITE (3,13)

      x=0.0
      y=1.0
      h=0.1
      kk=1

      fprimey=0.90333333333
      secfprimey=0.9
  5   yprime=x+y

      k1=x+y
      k2=(x+(1.0/3.0*h)+y+(1.0/3.0*h*k1))
      k3=(x+(2.0/3.0*h)+y+(2.0/3.0*h*k2))

      k=1

      Ynplus1=y+h/4.0*(k1+3*k3)


 15   dprimey=Ynplus1+(x+0.1)
      dprmprmy=dprimey+1
      ynprime=1+(x+y)
      c=Ynplus1-y
      cc=(0.1*(dprimey))

      ccc=(((h*h)/6.0)*((ynprime)+2*(dprmprmy)))

      secfy=c-cc

      fy=c-cc+ccc

      secrnewy=ynplus1-(secfy/secfprimey)

      rNewy=ynplus1-(fy/fprimey)
      xm=x+0.1

      q=2*exp(x+0.1)-(x+0.1)-1
```

```
      if (fy.gt.0.00001) then
      Ynplus1=rnewy
      goto 15
      end if

      write(4,11)Ynplus1,y,dprimey,ynprime,dprmprmy
11    format(f14.10,2X,f14.10,2X,f14.10,2X,f14.10,2X,f14.10)

10    format(f4.2,5x,f14.10,5x,f14.10,5x,f14.10,5x,f14.10,5x,f14.10)
      write(3,10) x+0.1,g,secrnewy,rnewy,g-rnewy,secrnewy-g

      WRITE (3,13)
      WRITE (3,13)
      x=x+0.1
      y=rnewy

      if (x.lt.1.00) then
      kk=kk+1
      goto 5
      end if

      stop
      end
```

Program of one -step Second derivative formula and Backward difference formula on problem 1

```
double precision k1,k2,k3,k4,y,h
      double precision yprime,ynplus1,ynn,fprimey,dprimey,dprmprmy
      double precision c,cc,ccc,fy,rnewy,g,ynprime
      double precision z
      double precisionqdprmprmy
      double precision xprev,yprev

      OPEN(UNIT=3,FILE='RESULT2.DAT',STATUS='NEW')
      OPEN(UNIT=4,FILE='RESULT.DAT',STATUS='NEW')

      WRITE (3,12)
12    FORMAT (3x,'X',11x,'Exact Sol.',11x,'Numerical Sol.',10x,'Error')

      WRITE (3,13)
13    FORMAT(2X)
      WRITE (3,13)
      x=0.0

      y=1.0
      h=0.1
      kk=1

      fprimey=0.90291e-06

      yprime=x+y

      k1=x+y
      k2=(x+(1.0/2.0*h)+y+(1.0/2.0*h*k1))
      k3=(x+(1.0/2.0*h)+y+(1.0/2.0*h*k2))
      k4=(x+h)+y+(h*k3)

      k=4

      Ynplus1=y+h/6.0*(k1+2*k2+2*k3+k4)
      rnewy=Ynplus1
      q=2*exp(x+0.1)-(x+0.1)-1

10    format(f6.2,5x,f16.10,5x,f16.10,5x,f16.10)
      write(3,10) x+0.1,q, rnewy,q-rnewy

      WRITE (3,13)
      WRITE (3,13)

      xprev=x
      yprev=y

      y=Ynplus1
      x=x+0.1

 5    yprime=x+y

      k1=x+y
      k2=(x+(1.0/2.0*h)+y+(1.0/2.0*h*k1))
      k3=(x+(1.0/2.0*h)+y+(1.0/2.0*h*k2))
      k4=(x+h)+y+(h*k3)
```

```
      ynn=y+h/6.0*(k1+2*k2+2*k3+k4)

c     ynn represent ynplus2
c     s represent ynplus2 prime

      s=ynn+x+0.1

15    dprimey=y'(x)

      dprmprmy=dprimey+1
      ynprime=1+(xprev+yprev)
      c=ynn-y
      s=ynn+x+0.1
      pdprmprmy=s+1
      cc=(0.1*(s))
      ccc=((h*h)/24.0)*(ynprime-6*(dprmprmy)-7*(pdprmprmy))
      fy=c-cc-ccc

      rNewy=ynn-(fy/fprimey)

      g=2*exp(x+0.1)-(x+0.1)-1
      if (fy.gt.0.00001) then
      Ynn=rnewy
      goto 15
      end if

      write(4,20) ynn
20    format('yplus2=',f16.10)

      write(3,10)x+0.1,g, rnewy,g-rnewy

      WRITE (3,13)
      WRITE (3,13)

      xprev=x
      yprev=y

      x=x+0.1
      y=rnewy

      write(4,21) y
21    format('y=',f16.10)
      write(4,22) x
22    format('x=',f16.10)

      if (x.lt.1) then
      kk=kk+1
      goto 5
      end if
      stop
      end
```

Program of two-step second derivative formula of problem one

```
      double precision k1,k2,k3,k4,y,h
      double precision yprime,yplus1,ynm,fprimey,dprimey,dprmprmy
      double precision c,cc,ccc,fy,rnewy,j,ynprime
      double precision b,bb,bbb
      double precision s
      double precision pdprmprmy
      double precision secfprimey
      double precision xprev,yprev
      double precision secyplus1
      double precision myprey
      double precision tprey
      double precision secrnewy
      double precision secy
      double precision myotemp


      OPEN(UNIT=3,FILE='RESULT2B.DAT',STATUS='NEW')
      OPEN(UNIT=4,FILE='RESULT2.DAT',STATUS='NEW')

      WRITE (3,12)
12    FORMAT(1x,'X',11x,'Exact Sol.',11x,'ODE',10x,'Proposed Sol.',
     1x,'Proposed Error',5x,'ODE Error')

      WRITE (3,13)
13    format(2x)
      WRITE (3,13)

      x=0.0
      y=1.0
      h=0.1
      kk=1

      fprimey=0.902916666
      secfprimey=0.93333333333

      yprime=x+y
      k1=x+y
      k2=(x+(1.0/2.0*h))+y+(1.0/2.0*h*k1))
      k3=(x+(1.0/2.0*h))+y+(1.0/2.0*h*k2))
      k4=(x+h)+y+(h*k3)

      k=1

      yplus1=y+h/6.0*(k1+2.0*k2+2.0*k3+k4)

      secyplus1=yplus1

      rnewy=Yplus1

      g=2.0*exp(x+0.1)-(x+0.1)-1

10    format(f6.2,5x,f16.10,5x,f16.10,5x,f16.10,5x,f16.10,5x,f16.10)
      write(3,10) x+0.1,g,secyplus1,rnewy,g-rnewy,secyplus1-g

      WRITE (3,13)
      WRITE (3,13)
```

```
      xprev=x

      yprev=y

      y=Ynplus1
      x=x+0.1

      secy=ynplus1

5     yprime=x+y

      k1=x+y
      k2=(x+(1.0/2.0*h)+y+(1.0/2.0*h*k1))
      k3=(x+(1.0/2.0*h)+y+(1.0/2.0*h*k2))
      k4=(x+h)+y+(h*k3)

      ynn=y+h/6.0*(k1+2.0*k2+2.0*k3+k4)

c     ynn represent ynplus2
c     n represent ynplus2 prime


      n=ynn+x+0.1

      dprimey=y+(x)

      dprmprmy=dprimey+1
      yprime=1+(xprev+yprev)

      mysecy=yprev

      c=ynn-y
      s=ynn+x+0.1
      pdprmprmy=n+1
      cc=(0.1*(s))

      ccc=((h*h)/24.0)*(yprime-6.0*(dprimey)+2.0*(dprmprmy))

      fy=c-cc-ccc


c      b=mysecy
      myotemp =1/3.0*(mysecy-4.0*secy-2.0*h*n)


      tsecfy=ynn+ myotemp


      secnewy=ynn-(tsecfy/secfprimey)

      rNewy=ynn-(fy/fprimey)


     g=2.0*exp(x+0.1)-(x+0.1)-1
```

```
      if (y.gt.0.000001) then
      Ynn=rnewy
      goto 15
      end if

      write(4,20) ynn
20    format('yplus2=',f16.10)


      write(3,10)x+0.1,q,secrnewy,rnewy,q-rnewy,secrnewy-q

      WRITE (3,13)
      WRITE (3,13)

      xprev=x
      yprev=y

      y=secrnewy

      x=x+0.1
      y=rnewy

      write(4,21) y
21    format('y=',f16.10)
      write(4,22) x
22    format('x=',f16.10)


      if (x.lt.1) then
      kk=kk+1
      goto 5
      end if

      stop
      end
```

Program of two-step second derivate formula and backward difference on problem 1