New numerical strategies for initial value type ordinary differential equations

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Additional Information:

- A Doctoral Thesis. Submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of Loughborough University.

Metadata Record: [https://dspace.lboro.ac.uk/2134/14145](https://dspace.lboro.ac.uk/2134/14145)

Publisher: © Bahrom Bin Sanugi

Please cite the published version.
This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

For the full text of this licence, please go to: http://creativecommons.org/licenses/by-nc-nd/2.5/
This book was bound by

**Badminton Press**
18 Half Croft, Syston, Leicester, LE7 8LD
Telephone: Leicester (0533) 602918.
NEW NUMERICAL STRATEGIES FOR INITIAL VALUE TYPE ORDINARY DIFFERENTIAL EQUATIONS

BY

BAHROM BIN SANUGI, B.Sc., M.Sc.

A Doctoral Thesis
Submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of the Loughborough University of Technology August, 1986.

Supervisor: PROFESSOR D.J. EVANS, Ph.D., D.Sc., F.I.M.A., F.B.C.S.

Department of Computer Studies

© by Bahrom Bin Sanugi, 1986.
I declare that the following thesis is a record of research work carried out by me, and that the thesis is of my own composition. I also certify that neither this thesis nor the original work contained therein has been submitted to this or any other institution for a degree.

B.B. SANUGI.
DEDICATION

to

my mother, Supiah

and

to the memory of

my father, Allahyarham Sanugi.
ACKNOWLEDGEMENTS

First and foremost, I wish to give all the Praise to Almighty God for with His mercy He has given me the life and sustenance in this world, and giving me the strength and time to complete this research.

I would like to express my profound gratitude to my supervisor, Professor D.J. Evans, for giving me the opportunity in the first place; and subsequently, for his guidance and direction. This thesis owes its shape and content to him. He has directed my research since I came to Loughborough, and has had a great influence on what I have accomplished in these years and what I know today. His thoroughness and sincerity in doing this will be a model for me in the years to come. He also tried to teach me his high professional and personal standards; although I resisted as strongly as I could, there are still some traces of them in this thesis.

I am also very grateful to all the staff of the Computer Studies Department, most especially to Mr. S. Bedi, Mr. G.S. Samra, and Dr. W. Yousif; to the staff of the Computer Centre especially to Mr. Geoff Harris and Dr. Anne Mumford; and to my colleagues for the useful discussions I had with them at one time or the other during the course of the preparation of this thesis.

I extend my gratitude to the Government of Malaysia, and in particular the Universiti Teknologi Malaysia for granting me the leave of absence and the financial support to complete this work.
I wish to express my special indebtedness to my wife, Paimah, for more than words can express; and finally to my children Elinor, Eva-Marlina and Emilia for their presence which provided me with encouragement throughout the years of my present higher education.
ABSTRACT

This thesis is concerned with the development of new numerical techniques for solving initial value problems in ordinary differential equations (ODE).

The thesis begins with an introductory chapter on initial value type problems in ordinary differential equations followed by a chapter on basic mathematical concepts, which introduces and discusses, among others, the theory of Arithmetic and Geometric Means. This is followed, in Chapter 3, by a survey of the existing ODE solvers and their theoretical background. The advantages and disadvantages of some different strategies in terms of stability and truncation error are also considered.

The presentation of the elementary methods based on Arithmetic Mean (AM) and Geometric Mean (GM) formulae is done in Chapter 4, with emphasis on establishing the GM trapezoidal formula, and to the study of its stability and truncation error. Applications in the predictor-corrector and the extrapolation techniques are also considered. Special application in the solution of delay differential equations is also presented.

In Chapter 5, the application of the GM strategy in the Runge-Kutta type formulae is considered, producing a new class of methods called the GM-Runge-Kutta formulae which is found to be as competitive as the classical Runge-Kutta methods. Thereafter, a new strategy of error control called the Arithmeto-Geometric Mean (AGM) strategy is developed. Further application of the GM-Runge-Kutta in Fehlberg type formulae, and the GM-Iterative Multistep formulae are also considered.
Chapter 6 concerns with further applications of GM techniques in the development of generalised GM multistep and multiderivative methods, and for solving $y'=\lambda(x)y$. The general idea of the GM are also extended to other types of Means, such as Harmonic and Logarithmic Means.

In Chapter 7, some new formulae for solving problems with oscillatory and periodic solutions are considered.

Finally the thesis concludes with recommendations for further work.
## CONTENTS

### Chapter 1: INTRODUCTION

1.1 Origin of Problems
1.2 Classification of Ordinary Differential Equations (ODEs)
1.3 Initial Value and Boundary Value Problems
1.4 Numerical Solution of ODE's

### Chapter 2: BASIC MATHEMATICAL CONCEPTS

2.1 Means
2.2 Power Series
2.3 Fourier Series
2.4 Asymptotic Expansions
2.5 Symbolic Computation - An Introduction to REDUCE

### Chapter 3: SURVEY OF ODE SOLVERS

3.1 Introduction
3.2 Practical Numerical Methods
3.3 Classical Linear Multistep Methods
  3.3.1 The initial value problem
  3.3.2 Consistency, zero-stability and convergence
  3.3.3 Weak stability
3.4 Methods of Absolute Stability
  3.4.1 Stiff initial-value problems and stiffness ratio
  3.4.2 Stability for stiff problems
3.5 Runge-Kutta Methods and Rosenbrock Methods
  3.5.1 Runge-Kutta methods with R-levels
  3.5.2 Determination of the coefficients
  3.5.3 Semi-explicit processes and the methods of Rosenbrock
  3.5.4 Stability properties of the Runge-Kutta processes
### Chapter 4: ELEMENTARY INTEGRATION METHODS BASED ON ARITHMETIC MEAN (AM) AND GEOMETRIC MEAN (GM) FORMULAE

#### 4.1 The Weighted GM Formula

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.1 Introduction</td>
<td>70</td>
</tr>
<tr>
<td>4.1.2 Class of nonlinear methods based on Euler formulae</td>
<td>70</td>
</tr>
<tr>
<td>4.1.3 Accuracy and stability analysis of formula (4.1.9)</td>
<td>70</td>
</tr>
<tr>
<td>4.1.4 Stability consideration of the ( \theta )-method for stiff problems</td>
<td>71</td>
</tr>
<tr>
<td>4.1.5 Class of equations which the GM formula gives improved accuracy over the trapezoidal method</td>
<td>78</td>
</tr>
<tr>
<td>4.1.6 Application of the GM formula for a system of o.d.e.'s</td>
<td>83</td>
</tr>
<tr>
<td>4.1.7 Quadrature methods: GM-Romberg scheme</td>
<td>91</td>
</tr>
</tbody>
</table>

#### 4.2 Application of the GM Method in the Predictor-Corrector Manner

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.1 Euler's predictor formula</td>
<td>100</td>
</tr>
<tr>
<td>4.2.2 Newton Cotes formula</td>
<td>108</td>
</tr>
<tr>
<td>4.2.3 Hyperbolic p.d.e. example</td>
<td>113</td>
</tr>
</tbody>
</table>

#### 4.3 The GM Formula With Extrapolation

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.1 Introduction</td>
<td>118</td>
</tr>
<tr>
<td>4.3.2 Asymptotic expansion</td>
<td>121</td>
</tr>
<tr>
<td>4.3.3 Application of the extrapolation method to I.V.P.</td>
<td>131</td>
</tr>
<tr>
<td>4.3.4 Rational extrapolation</td>
<td>136</td>
</tr>
</tbody>
</table>

#### 4.4 GM Formula for Solving Delay Differential Equation

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.4.1 Introduction</td>
<td>142</td>
</tr>
<tr>
<td>4.4.2 Trapezoidal and GM methods for solving delay differential equations</td>
<td>144</td>
</tr>
</tbody>
</table>

### Chapter 5: APPLICATION OF THE GM STRATEGY IN THE RUNGE-KUTTA TYPE FORMULA

#### 5.1 GM Runge-Kutta Formulae

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.1 Introduction</td>
<td>156</td>
</tr>
<tr>
<td>5.1.2 Derivation of the 1st to 4th order GM Runge-Kutta formulae</td>
<td>157</td>
</tr>
<tr>
<td>5.1.3 Stability analysis for the GM Runge-Kutta methods</td>
<td>175</td>
</tr>
<tr>
<td>5.1.4 Numerical example</td>
<td>180</td>
</tr>
</tbody>
</table>
5.2 A New Strategy in the Use of Runge-Kutta Methods With Error Control for Solving $y' = Ay$

5.2.1 Arithmeto-Geometric Mean (AGM) strategy for error control 182
5.2.2 Derivation of the 4th order AGM formula 185
5.2.3 An error estimate for the 4th order AGM formula 188
5.2.4 Reliability test of the error estimate 189
5.2.5 Practical numerical example of stepsize control 190
5.2.6 AGM methods of lower order 191

5.3 GM Runge-Kutta Fehlberg Method 196

5.3.1 Runge-Kutta methods of Fehlberg variation 196
5.3.2 Derivation of GM RK3(4) formula 199
5.3.3 GM Runge-Kutta RK2(3) formula 205
5.3.4 Comments for further investigation 208

5.4 GM Version of the Iterative Multistep (IMS) Formula 210

5.4.1 Iterative multistep methods 210
5.4.2 IMS methods with GM base formula 212
5.4.3 Symbolic determination of the coefficients in the Hyman's IMS leap-frog method 218

5.5 Two-Stage Implicit GM Runge-Kutta Formula 227

5.5.1 Implicit Runge-Kutta methods 227
5.5.2 2-stage implicit GM Runge-Kutta method 228

Chapter 6: FURTHER APPLICATIONS OF GM TECHNIQUES

6.1 Generalised GM Multiderivative Methods 237

6.1.1 Approximation by rational functions and a survey on Pade' Table 237
6.1.2 Inclusion of the nonlinear methods in the Pade' Table - A preliminary approach 241
6.1.3 Implementation of formula (6.1.27) in PECE-mode of the predictor-corrector procedure 250
6.1.4 Comparison of the stability regions of the (0,2)-<2,2> and (0,2)-(2,2) combinations in PECE mode 253

6.2 GM Multistep Methods 256

6.2.1 GM 2-step formulae 256
6.2.2 Modification of formula (6.2.2) 258

6.3 Application of the GM formula to the Problems of the Form $y' = A(x)y$

6.3.1 Gourlay's method 264
6.3.2 GM alternative 265
6.4 Solution of the Goursat Problem in Hyperbolic Partial Differential Equations by the GM and the Modified GM Formulae

6.5 Methods Using Other Types of Means

6.5.1 Various types of means
6.5.2 Harmonic Mean (HM) formula
6.5.3 A second order one-step method based on a new mean
6.5.4 A method derived by using Logarithmic Mean

Chapter 7: OSCILLATORY AND PERIODIC PROBLEMS

7.1 The Numerical Solution of Oscillatory Problems

7.1.1 Gautschi's method
7.1.2 Geometric mean formula
7.1.3 Leap-frog method by using Gautshi's technique
7.1.4 Runge-Kutta formula for problems with oscillatory solutions
7.1.5 Higher order Runge-Kutta (oscillatory) formula

7.2 A Fourier Series Method for the Numerical Solution of a Certain Class of Nonlinear Parabolic PDE of the Form

\[ \frac{\partial u}{\partial t} = -\frac{1}{2} \frac{\partial^2 u}{\partial x^2} + \lambda \frac{\partial^2 u}{\partial x^2} + d(x,t) \]

7.2.1 Introduction
7.2.2 The method for nonlinear problem
7.2.3 Numerical example
7.2.4 A comparison between the Fourier series technique and the method of lines in terms of computational complexity and accuracy

7.3 The Fourier Series Solution of the Nonlinear Advection Problem

7.4 Further Application of Fourier Series Method

7.4.1 Fourier series solution of the first order hyperbolic systems in one space dimension
7.4.2 Linearised Burgers equation
7.4.3 Numerical solutions of the wave equation

7.5 Periodic Solution of Van Der Pol's Equation

7.5.1 Introduction
7.5.2 Numerical integration of the van der Pol equation
7.5.3 Greenspan's method for the determination of the period of the van der Pol oscillation
7.5.4 Determination of the period by autocorrelation methods
CHAPTER 1

INTRODUCTION
1.1 ORIGIN OF PROBLEMS

Many problems in science and engineering can be formulated in terms of differential equations. A large part of the motivation for building the early computers came from the need to compute ballistic trajectories accurately and quickly. Today's computers are used extensively to solve the equations of ballistic-missile and artificial-satellite theory, as well as those of electrical networks, bending of beams, stability of aircraft, vibration theory, all of which are representable in the form of differential equations.

As an example of how differential equations may arise in practical problems, let us consider a problem of the vertical motion of a rocket. This is a problem of interest where a rocket ascends in a straight line under the influence of gravity with a thrust that is constant both in magnitude and direction.

The thrust of the rocket is produced by the ejection of mass at a constant rate $f$ with exhaust velocity $V_e$, which when measured relative to the rocket is constant. The height $x$ above the Earth's surface at a time $t$ while the rocket is still firing is determined by the following non-linear differential equation

$$\frac{d^2x}{dt^2} + g \left( \frac{x}{R} \right)^{-2} = \frac{V_e f}{m_0 - ft}$$

(1.1.1)

where $g$ is the acceleration caused by gravity at the Earth's surface, $R$ is the radius of the Earth, and $m_0$ is the initial mass of the rocket. The initial conditions in this case are that $x=0$, $dx/dt=V$ when $t=0$, i.e., at the time of blast-off. A numerical method is thus necessary to compute the position of the rocket at time $t$ after the launching. The speed of the computation will depend on the machine being used, and almost always, such computations are carried out by on-board computers incorporated in the real-time mission-control systems.
1.2 CLASSIFICATION OF ORDINARY DIFFERENTIAL EQUATIONS

An ordinary differential equation is a relation connecting the function $y$ of an independent variable $x$ and its derivatives $y', y'', \ldots, y^{(n)}$ in the form of

$$f(x, y', y'', \ldots, y^{(n)}) = 0.$$  \hspace{1cm} (1.2.1)

Examples of ordinary differential equations are

$$\frac{dy}{dt} = -\lambda y$$  \hspace{1cm} (1.2.2)
$$\alpha \frac{d^2\theta}{dt^2} = -gsin\theta$$  \hspace{1cm} (1.2.3)
$$(y'')^3 + (y')^2 - y = e^x$$  \hspace{1cm} (1.2.4)
$$a_n(x)y^{(n)} + \ldots + a_1(x)y' + a_0(x)y = \psi(x)$$  \hspace{1cm} (1.2.5)

If the derivative of the highest order occurring in the equation is $y^{(n)} = \frac{d^n y}{dx^n}$, the equation is said to be of order $n$. The power to which the highest derivative occurs is called the degree of the equation. Therefore, equations (1.2.2) and (1.2.3) are of the first degree, but equation (1.2.4), however, is of third degree because the highest derivative occurs to the power 3. If the equation is of a form in which the function $y$ and its derivatives occur linearly and in which the coefficients $a_n(x), \ldots, a_0(x)$ depend only on $x$ and not on $y$ or of its derivatives, such as (1.2.5), the equation is said to be linear. If $\psi(x) = 0$, the equation is said to be homogeneous; and if $\psi(x)$ is not identically zero, it is said to be non-homogeneous. Equation (1.2.2) is linear and homogeneous, but equation (1.2.3) is nonlinear.

Historically, the study of differential equations is almost as old as that of the calculus itself. But the more recent work in ordinary differential equations is of a basic nature, concerned with the conditions that guarantee the existence of a solution of a given equation. The theory is more concerned with establishing that a solution exists rather than with trying to derive a closed form for it.
Such an attitude is essential when many of the practical problems involving differential equations are solved by the use of electronic computers. The validity of numerical processes must be thoroughly investigated. The study of differential equations continues to contribute to the solution of practical problems in control theory, in orbital mechanics, and in many other branches of science and technology, and also to ask challenging questions of pure mathematicians working in such apparently abstract subjects as functional analysis and the theory of differentiable manifolds.
1.3 INITIAL VALUE AND BOUNDARY VALUE PROBLEMS

The prototype problems associated with (1.2.1) are the initial value and boundary value problems. The former requires a function $y(x)$ to be determined for $x>x_0$ which satisfies an $n$th-order equation

$$y^{(n)}=f(x,y,y',...y^{(n-1)})$$

and takes the initial values

$$y(x_0)=A_0,...,y^{(n-1)}(x_0)=A_{n-1}.$$  

consisting of the function and its first $n-1$ derivatives at $x=x_0$. When $n \geq 2$, the single differential equation can be replaced by a system of first order equations. For example, if $n=2$, the second order equation becomes the first order system

$$y'=p, \quad p'=f(x,y,p)$$

with the initial conditions $y(x_0)=A_0$, $p(x_0)=A_1$. This can be written in vector form

$$Y'=F(x,Y), \quad Y=[y,p]^T, \quad F=[F_1,F_2]^T$$

in which the vector components are $F_1=p$ and $F_2=f(x,y,p)$, with initial condition is also expressible in vector form

$$Y(x_0)=[A_0,A_1]^T.$$  

A typical boundary value problem is given by a second order differential equation

$$y''=f(x,y,y')$$

together with the two boundary values at points a and b

$$y(a)=y_a, \quad y(b)=y_b.$$  

The solution is required for $a \leq x \leq b$.

In short, the initial value problems are those in which all boundary conditions are specified at one point, while the boundary value problems are those in which the boundary conditions are distributed between two or more points. A linear homogeneous boundary value problem is called the eigenvalue problem.
1.4 NUMERICAL SOLUTION OF ODE's

The numerical solution of ordinary differential equations becomes important because of the many physical problems which lead to ordinary differential equations that cannot be solved analytically.

In the vast majority of numerical methods for such equations the differential equation is replaced by a difference equation. A discrete set of equally placed arguments $x_k$ distance $h$ apart is chosen, and a sequence of corresponding values $y(x_k)$ is calculated from the difference formula. The mathematical preliminary that is essential in discussing these numerical methods will be given in the next chapter, while the details of the existing methods, together with its theory, will be included in Chapter 3.

In the rest of the thesis, attention will be directed only to the study and the development of new strategies in solving the initial value problems. In addition to the initial value problems, study is also carried out to apply initial value techniques to approximate periodic solutions of equations for which auxiliary conditions are not always given completely.

Further investigations of the GM technique which are related to the subject discussed in this thesis are published in Evans and Sanugi [1986a, 1986b] and are included in Appendix 8.
CHAPTER TWO

BASIC MATHEMATICAL CONCEPTS
2.1 MEANS

One of the most famous inequalities in the field of mathematical analysis concerns the arithmetic mean $A_n$ and the geometric mean $G_n$ of $n$ positive numbers $x_1, x_2, \ldots, x_n$. These means have been well known since Greek times. Indeed the first two sections of Book III of the Mathematical Collection of Pappus of Alexandria (a combined commentary and guidebook of Greek mathematics written towards the end of the third century AD) is devoted to the theory of music as well as arithmetic (see Colwell and Gillett [1985]). These means are defined by

\[
A_n = \frac{x_1 + x_2 + \ldots + x_n}{n} \quad (2.1.1)
\]

\[
G_n = \left( x_1 x_2 \ldots x_n \right)^{1/n} \quad (2.1.2)
\]

and satisfy the following inequality

\[
A_n \geq G_n,
\]

with equality occurring if and only if $x_1 = x_2 = \ldots = x_n$. Geometrically the inequality $A_n \geq G_n$ for two numbers can be proved by a geometric construction described in Figure 2.1. Representing numbers $x_1$ and $x_2$ by the straight line segments $AB$ and $BC$, we construct a circle, centre $0$, on $AC$ as diameter. Further, through $B$ draw a line perpendicular to $AC$ to cut the circle at $D$. Then it is easily proved that the arithmetic mean of the number $x_1$ and $x_2$ is given by $OD$ and the geometric mean by $BD$. Hence, since $OD \geq BD$, the arithmetic mean-geometric mean inequality follows. Incidentally, another mean, the harmonic mean

\[
H_2 = \frac{2x_1 x_2}{x_1 + x_2} \quad (2.1.3)
\]

is given by $FD$, where $F$ is the foot of the perpendicular from $B$ on to
OD, and it may be shown that

$$A_2 \geq G_2 \geq H_2.$$  

This result can also be extended to \( n \) positive numbers.

![Figure 2.1](image)

By using a more general mathematical statement, suppose that we are given an \( n \)-tuple \( x=(x_1,x_2,\ldots,x_n) \), \( x_i \geq 0 \). We set

$$M_r = M_r(x) = \left( \frac{1}{n} \sum_{i=1}^{n} x_i^r \right)^{1/r}.$$  \hspace{1cm} (2.1.4)

If at least one \( x_i \) is 0 and \( r < 0 \), we put \( M_r = 0 \). In particular, we put

$$A = M_1,$$  \hspace{1cm} (2.1.5)

$$G = \lim_{r \to 0^+} M_r = \left( \prod_{i=1}^{n} x_i \right)^{1/n},$$  \hspace{1cm} (2.1.6)

$$H = M_{-1}.$$  \hspace{1cm} (2.1.7)

These are the arithmetic mean, geometric mean, and harmonic mean of \( x_i \) (\( i=1,2,\ldots,n \)), respectively.

Except when either all \( x_i \) are identical, or some \( x_i \) is zero and \( r \leq 0 \), the function \( M_r \) increases strictly monotonically as \( t \) increases, and \( M_r \to \min x_i (r \to -\infty) \), \( M_r \to \max x_i (r \to +\infty) \). Therefore, we always have \( \min x_i \leq M_r \leq \max x_i \). In particular, we have \( H \leq G \leq A \) if the \( x_i \) are all positive and not all equal. In a wider sense equation
(2.1.4) is called the generalised mean.

In addition to these means we also have one which is called the logarithmic mean. For two positive numbers \(x_1\) and \(x_2\), the logarithmic mean is defined as

\[
M_{log} = \frac{x_2 - x_1}{\ln \left( \frac{x_2}{x_1} \right)}
\]

Weighted Means

Let us now extend the notion of means to the more general concept of weighted means. Suppose that \(x=[x_1, x_2, \ldots, x_n]\) and \(w=[w_1, w_2, \ldots, w_n]\) be two ordered set of values in \(R\). We now define the two types of weighted means as follows,

**Definition 2.1**

The weighted arithmetic mean, \(\bar{x}\) of \(x\) with weight \(w\) is defined by

\[
\bar{x} = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}
\]

**Definition 2.2**

The weighted geometric mean \(\hat{x}\) of \(x\) with weight \(w\) is defined by

\[
\hat{x} = \left( \prod_{i=1}^{n} x_i^{w_i} \right)^{\frac{1}{\sum_{i=1}^{n} w_i}}
\]

In most practical applications, the weights \(w\) are scaled so that \(\sum w_i = 1\).
Example

Let $x = [2, 3, 4, 5]$ and $w = [1, 2, 2, 1]$. Then we have, by Definition 2.1 and Definition 2.2,

$$
\bar{x} = \frac{1(2) + 2(3) + 2(4) + 1(5)}{1 + 2 + 2 + 1} = 3.50
$$

and

$$
\hat{x} = (2^{1}, 3^{2}, 4^{2}, 5^{1})^{1/6} = 3.36.
$$

As we may notice in this example, it is true that $\bar{x} > \hat{x}$. It is also noticed that the amount of work in computing the weighted arithmetic mean consists of $n$ multiplications and one division, whereas that of the geometric mean is $(n+1)$ powers. Unfortunately, therefore, the weighted geometric mean involves more computational work.
2.2 POWER SERIES

Let \( a \) and \( c_0, c_1, c_2, \ldots \) be elements of a field \( K \) and \( z \) be a variable. A series of the form

\[
P(z) = \sum_{n=0}^{\infty} c_n(z-a)^n
\]

is called a power series (in one variable). We assume that \( K \) is the field of complex numbers. For a given power series \( P \), we can determine a unique real number \( R \) (0 \( \leq R \leq \infty \)) such that \( P \) converges if \( |z-a| < R \) and diverges if \( R < |z-a| \). We call \( R \) the radius of convergence, and the circle \( |z-a| < R \) the circle of convergence of \( P \). The value of \( R \) is given by \( R = 1/\limsup_{n \to \infty} (|c_n|)^{1/n} \) (Cauchy-Hadamard formula).

A power series converges absolutely and uniformly in the wider sense in its circle of convergence, and defines there a single-valued complex function. Since the series is termwise differentiable, the function is actually a holomorphic function of a complex variable. Conversely, any function \( f(z) \) holomorphic in a domain can be represented by a power series in a neighbourhood of each point \( a \) of the domain, called the Taylor expansion of \( f(z) \) at \( a \) (or in the neighbourhood of \( a \)).

Besides the series (2.2.1), a series of the form

\[
Q(z) = \sum_{n=0}^{\infty} c_n z^{-n}
\]

is called a power series with centre at the point at infinity, and its value at \( \infty \) is defined to be \( c_0 \). By putting \( z = a + t \) when its centre \( a \) is a finite point and \( z = 1 + t \) when its centre is \( \infty \), every power series can be written in the form

\[
\sum_{n=0}^{\infty} c_n t^n,
\]

and such a \( t \) is called a canonical parameter.
When \( t \) is a local canonical parameter, a series of the form

\[
\sum_{n=-\infty}^{\infty} c_n t^n
\]

is called a Laurent series. Power series are also called Taylor series.

We shall now give in some detail the notion of the mean value theorem and the Taylor's series formula.

**The Mean Value Theorem and Taylor's Formula**

Let \( f(x) \) be a continuous function defined on \([a, b)\), and suppose that for every point \( x_0 \) on \((a, b)\) there exists a limit

\[
\lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h},
\]

which may be infinite. (These conditions are satisfied if \( f(x) \) is differentiable on \([a, b)\).) Then there exists a point \( \xi \) such that

\[
\frac{f(b) - f(a)}{b - a} = f'(\xi), \quad a < \xi < b.
\]

This proposition is called the *mean value theorem*. A special case of the theorem under the further condition that \( f(a) = f(b) \) is called Rolle's theorem. If we put \( b - a = h \), \( \xi = a + \theta h \), then the conclusion of the theorem may be written as

\[
f(a + h) = f(a) + hf'(a + \theta h), \quad 0 < \theta < 1.
\]

Further, suppose that \( f(x) \) is \( n \)-times differentiable on an open interval \( I \). For a fixed \( a \in I \) and an arbitrary \( x \in I \), we put

\[
f(x) = f(a) + \frac{f'(a)}{1!} (x - a) + \ldots + \frac{f^{(n-1)}(a)}{(n-1)!} (x - a)^{n-1} + R_n.
\]

Then

\[
R_n = \frac{f^{(n)}(\xi)}{n!} (x - a)^n \quad \text{for some} \ \xi \ \text{between} \ a \ \text{and} \ x.
\]

This is called
Taylor's formula, where \( R_n \) is the remainder of the nth order given by Lagrange.
2.3 FOURIER SERIES

The set of functions $\frac{1}{\sqrt{2\pi}}, \cos x/\sqrt{\pi}, \sin x/\sqrt{\pi}, \ldots, \cos kx/\sqrt{\pi}, \sin kx/\sqrt{\pi}, \ldots$, which is called the trigonometric system, is an orthogonal system in $(-\pi, \pi)$. Let $f(x)$ be an element of $L_1(-\pi, \pi)$, i.e., Lebesque integrable in $(-\pi, \pi)$. Throughout this section we assume that integrals are always Lebesque integrals. We put

$$
\begin{align*}
    a_k &= -\frac{1}{\pi} \int_{-\pi}^{\pi} f(t) \cos k t \, dt, \\
    b_k &= -\frac{1}{\pi} \int_{-\pi}^{\pi} f(t) \sin k t \, dt,
\end{align*}
$$

and call $a_k, b_k$ the Fourier coefficients of $f$. The formal series

$$
a_0/2 + \sum_{k=1}^{\infty} (a_k \cos kx + b_k \sin kx)
$$

is called the Fourier series of $f$ and is often denoted by $F(f)$. To indicate that a formal series $F(f)$, as above, is the Fourier series of a function $f$, we write

$$
f(x) \sim a_0/2 + \sum_{k=1}^{\infty} (a_k \cos kx + b_k \sin kx).
$$

The sign $\sim$ means that the numbers $a_k, b_k$ are connected with $f$ by the formula (2.3.1); it does not imply that the series is convergent, still less that it converges to $f$. Generally, trigonometric series are those of form (2.3.2), where $a_k, b_k$ are arbitrary real numbers. Since the trigonometric series have period $2\pi$, we assume that the function considered are extended for all real $x$ by the condition of periodicity $f(x+2\pi) = f(x)$. To study the properties of the series $F(f)$ and the representation of $f$ by $F(f)$ are major research topics in the theory of Fourier series. Since $e^{ix} = \cos x + i\sin x$, if we set $2c_k = a_k + ib_k, \quad c_k = \overline{c_k}$ ($k=0,1,2\ldots$), we have
\[ c_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(t)e^{-ikt} \, dt, \quad k=0,\pm 1, \ldots \]

Then \( F(f) \) is represented by the complex form \( \sum_{k=-\infty}^{\infty} c_k e^{ikx} \), and \( \{ e^{ikx} \} \) (\( k=0,\pm 1, \ldots \)) is an orthogonal system in \((-\pi, \pi)\). In this complex form we take symmetric partial sums such as \( \sum_{k=-n}^{n} c_k e^{ikx} \) (\( n=1,2,\ldots \)).

Consider the power series \( \frac{a_0}{2} + \sum_{k=1}^{\infty} (a_k - ib_k)z^k \) on the unit circle \( z=e^{ikx} \) in the complex plane. Its real part is the trigonometric series (2.3.2), and the imaginary part (with vanishing constant term) is
\[
\sum_{k=1}^{\infty} (a_k \sin kx - b_k \cos kx), \quad (2.3.3)
\]
which is called the conjugate series of \( f \) and is denoted by \( F^c(f) \). In complex form, the conjugate series is \(-i \sum_{k=-\infty}^{\infty} (\text{sgn } k)c_k e^{ikx}\).

If \( f \) and \( g \) belong to \( L_1(-\pi, \pi) \) and \( f(x) \sim \sum_{k=-\infty}^{\infty} c_k e^{ikx} \), \( g(x) \sim \sum_{k=-\infty}^{\infty} d_k e^{ikx} \), then
\[
\frac{1}{2\pi} \int_{0}^{2\pi} f(x-t)g(t) \, dt \sim \sum_{k=-\infty}^{\infty} c_k d_k e^{ikx}.
\]

The function
\[
f* g(x) = \frac{1}{2\pi} \int_{0}^{2\pi} f(x-t)g(t) \, dt \quad (2.3.4)
\]
is called the convolution of \( f \) and \( g \).

If \( f \) is absolutely continuous, then the derivative \( f'(x) \) satisfies
\[
f'(x) \sim i \sum_{k=-\infty}^{\infty} kc_k e^{ikx} = \sum_{k=1}^{\infty} k(-a_k \sin kx + b_k \cos kx).
\]

If \( f \) satisfies the Lipschitz condition of order \( \alpha \) \((0<\alpha<1)\), then \( c_n = O(n^{-\alpha}) \).
Convergence Tests

The nth partial sums \( s_n(x) = s_n(x; f) \) of the Fourier series \( F(f) \) can be written in the following form:

\[
s_n(x) = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x+t) D_n(t) \, dt,
\]

where

\[
D_n(t) = \frac{\sin(n+1/2)t}{2\sin(t/2)}.
\]

The function \( D_n(t) \) is called the Dirichlet kernel. For a fixed point \( x \) we set \( W_x(t) = f(x+t) + f(x-t) - 2f(x) \); then

\[
s_n(x) - f(x) = \frac{1}{\pi} \int_{0}^{\pi} W_x(t) D_n(t) \, dt.
\]

Hence if the integral on the right-hand side tends to zero as \( n \to \infty \),

\[
\lim_{n \to \infty} s_n(x) = f(x).
\]

If \( f \) vanishes in an interval \( I = (a, b) \), then \( F(f) \) converges uniformly in any interval \( I' = (a+e, b-e) \) interior to \( I \), and the sum of \( F(f) \) is 0. This is called the principle of localization.

There are a number of convergence tests, but here we give only three of them.

(1) The first test requires the notion of functions of bounded variation. Let \( f(x) \) be real bounded function defined on a closed interval \([a, b]\) in \( \mathbb{R} \). Given a subdivision of interval \( a=x_0<x_1<x_2< \ldots <x_n=b \), we denote the sum of positive differences \( f(x_i)-f(x_{i-1}) \) by \( P \) and the sum of negative differences \( f(x_i)-f(x_{i-1}) \) by \(-N\). Then we easily obtain

\[
P-N = f(b)-f(a), \quad \text{(2.3.6)}
\]

\[
P+N = \sum_{1}^{n} |f(x_i)-f(x_{i-1})| \quad \text{(2.3.7)}
\]

The suprema of \( P, N, \) and \( P+N \) are called the positive variation, the
negative variation, and the total variation of the function $f(x)$ in the
interval $[a,b]$, respectively. If any of these three values is finite,
then all three values are finite. In such a case, the function $f(x)$ is
called a function of bounded variation.

Now, if $f$ is of bounded variation, $F(f)$ converges at every point $x$
to the value $(f(x+0)+f(x-0))/2$. In addition, if $f$ is continuous at
every point of a closed interval $I$, $F(f)$ is uniformly convergent in $I$
(Jordan's test). As a special case of this test, bounded functions having
a finite number of maxima and minima and no more than a finite number
of points of discontinuity have convergent Fourier series (Dirichlet
test).

(2) If the integral

$$
\int_0^\pi \frac{|W_x(t)|}{t} \, dt
$$

is finite, then $F(f)$ converges at $x$ to $f(x)$ (Dini's test).

(3) If

$$
\int_0^h |W_x(t)| \, dt = o(h), \quad (2.3.8)
$$

$$
\lim_{\mu \to 0} \int_{t}^{t+\mu} \frac{|W_x(t) - W_x(t+\mu)|}{\mu} \, dt = 0 \quad (2.3.9)
$$

then $F(f)$ converges at $x$ to $f(x)$ (Lébesgue's test).
2.4 ASYMPTOTIC EXPANSIONS

Let $\alpha$ be a boundary point of an open connected domain $D$ in the complex $z$-plane or a Riemann surface. Assume that $Q_n(z)$ ($n=0, 1, 2, \ldots$) are holomorphic functions of $z$ in $D$ and that for every $n=0, 1, 2, \ldots$, as $z$ tends to $\alpha$ through $D$, $Q_{n-1}(z) = o(Q_n(z))$ holds. A function $f(z)$ is that holomorphic in $D \cup U$ for some neighbourhood $U$ of $\alpha$ will be said to have the asymptotic expansion in the form

$$f(z) \sim a_0 Q_0(z) + a_1 Q_1(z) + \ldots + a_n Q_n(z) + \ldots$$

as $z \to \alpha$ through $D$ if $f(z)$ satisfies

$$f(z)-a_0 Q_0(z)-a_1 Q_1(z)-\ldots-a_n Q_n(z) = o(Q_{n+1}(z))$$

for any integer $n > 0$ as $z \to \alpha$ through $D$. The coefficients $a_n$ ($n=0, 1, \ldots$) appearing in (2.4.1) are uniquely determined. This fact immediately follows from the formulae

$$a_0 = \lim_{z \to \alpha} \frac{f(z)}{Q_0(z)},$$

$$\ldots$$

$$a_n = \lim_{z \to \alpha} \frac{f(z)-a_0 Q_0(z)-\ldots-a_{n-1} Q_{n-1}(z)}{Q_n(z)}.$$

In most cases, the functions $Q_n(z)$ have the form $Q(z)^n W(z)$ ($n=0, 1, \ldots$), where $Q(z)$ and $W(z)$ are holomorphic functions of $z$ in $D$. For example, suppose that $f(z) = \Gamma(z)$, $D = \{z | 0 < |z| < \infty, \ argz \in \pi\}$, and $\alpha = \infty$. Then we have the following asymptotic expansion:

$$\log \Gamma(z) = \frac{B_2}{1.2z} - \frac{B_4}{3.4z^3} - \ldots + \frac{(-1)^{n-1}B_{2n}}{(2n-1)(2n)z^{2n-1}} + \ldots$$

(2.4.3)

where $B_{2n}$ is the Bernoulli number. Hence when $z$ tends to $\alpha = \infty$ through $D$, we have
If $f(z)$ is holomorphic at $z=\alpha$, its Taylor expansion

$$f(z) = a_0 + a_1(z-\alpha) + \ldots + a_n(z-\alpha)^n + \ldots$$

(2.4.4)
can be considered as an asymptotic expansion of $f(z)$ by taking $D = \{ z \mid 0<|z-\alpha|<r \}$, $Q_n(z) = (z-\alpha)^n$. Conversely, if $f(z)$ is a holomorphic and single-valued function of $z$ in $0<|z-\alpha|<r$ and, moreover, admits an asymptotic expansion of the form

$$f(z) \sim a_0 + a_1(z-\alpha) + \ldots + a_n(z-\alpha)^n + \ldots$$

(2.4.5)
then $\alpha$ is a removable singular point. Consequently, $f(z)$ admits Taylor expansion at $z=\alpha$. By virtue of the uniqueness of the coefficients in asymptotic expansions, the power series on the right-hand side of (2.4.5) converges uniformly for $|z-\alpha| < r$, and the sum coincides with $f(z)$. 

$$\Gamma(z) \sim \sqrt{2\pi} \ z^{z-\frac{1}{2}} e^{-z} \quad \text{(Stirling formula)}.$$
2.5 SYMBOLIC COMPUTATION - AN INTRODUCTION TO REDUCE

The purpose of this section is to introduce the new tool of symbolic algebraic computation (or symbolic computation, for short) and point out its potential in the numerical modelling and simulation of field problems.

Symbolic computation refers to the technique of manipulating on a computer, symbolic expressions that may not necessarily have numerical values. Therefore, techniques of symbolic computation can be used, among other things, to perform algebraic manipulations of mathematical formulae.Crudely, one can think of symbolic computation as a computerized version of the traditional "paper and pencil" manipulations of algebraic expressions commonly arising in Applied Mathematics. Therefore, symbolic computation can significantly reduce the tedium of analytic calculations and increase their reliability. This capability permits one to carry on the analytic calculations before numerical computations start.

A number of symbolic manipulation systems suitable for manipulating algebraic expressions have been developed over the past few years. A representative up to date sample of these are listed as follows:

- ALPAK
- FORMAC
- MATLAB
- SAC-1
- SMP
- ALTRAN
- MACSYMA
- muMath
- SCHOONSCHIP
- SYMBAL
- CAYLEY
- MAPLE
- REDUCE 3
- SCRATCHPAD II
- CONFORM

Jenson and Niordson [1977] have given a comparative study of some of these systems.
REDUCE is a system for carrying out algebraic operations accurately, on relatively complicated expressions. It can manipulate polynomials in a variety of forms, both expanding and factoring them, and extracting various parts of them as required. REDUCE can also perform differentiation and integration as well as the manipulation of arrays and in the operations of matrices. Its capability also includes the definition of procedures and operators and is backed by the availability of many options for varying computational procedures, output forms, and so on. As such it can usefully be applied in solving many topics of interest by the physicists, mathematicians and engineers.

REDUCE is also designed to be an interactive system, so that an algebraic expression can be input and its value inspected before moving on to the next calculation. However, REDUCE can also be used in a batch mode by inputting a sequence of calculations and obtaining results without any possibility of interaction during the calculations.

To show the interactive use of REDUCE, we shall give some examples which illustrates comprehensively the capabilities of the system. After a successful logging-in, we can run REDUCE on the VAX11 at Loughborough University by entering (in lower case)

```
* reduce
```

after which REDUCE will respond with a banner line and then prompt for the first line of input:

```
REDUCE 3.2, 15-Apr-85...
```

1:

where (1: ) is automatically assigned to the first command. We can now begin entering commands. If, for example, one wishes to work with the expression \((x-y)^4\), then after the appropriate logging-in procedure and when REDUCE is called, one proceeds by typing a FORTRAN-like expression, terminated by a semi-colon as follows:
1: \((x-y)^{**4}\);

The semi-colon indicates the end of the expression. By pressing the Return key, the system would then input the expression, evaluate it, and return the result in a form like:

\[x^4 - 4x^3y + 6x^2y^2 - 4xy^3 + y^4\]

2:

where (2:) is automatically assigned to the second command. Input may be in lower case or uppercase, but lower case is converted to upper case by the system, such that output is in upper case.

The results of a given calculation are saved in the variable WS (for workspace), so this can be used in the next calculation for further processing. For example, one could enter on line (2:) the expression

\[\text{df(ws,y);}\]

which calculates the derivative of the previous evaluation with respect to \(y\), and REDUCE responds with

\[4*(-x^3 + 3x^2y - 3xy^2 + y^3).\]

Alternatively,

\[\text{int(ws,x);}\]

would calculate the integral of the same expression with respect to \(x\) and REDUCE responds with

\[x*(-x^3 + 4x^2y - 6xy^2 + 4y^3).\]

In many cases, it is necessary to use the result of one calculation in succeeding calculations. One way to do this is via an assignment for a variable, such as,

\[v:=(x-y)^{**4};\]

If we now use \(v\) in later calculations, the value of the right hand side of the above will be used.

An important class of commands in REDUCE is that which defines
substitutions for variables and expressions to be made during the evaluation of expressions. Such substitutions use forms of the command LET.

LET rules stay in effect until replaced or CLEARed. For example, after assigning the expression \((x-y)^4\) to \(v\), we can give numerical values to \(x\) and \(y\) and hence find the numerical value of \(v\) by using the command LET as follows:

```plaintext
let x=3, y=1;
v;
```

REDUCE responds with

```
16
```

But if we want to have the value assigned to another variable \(u\) (say), we proceed as follows:

```plaintext
let x=3, y=1;
u:=v;
```

REDUCE then responds with

```
U:=16
```

Another command which is very useful for the purpose of substitution is the OPERATOR command. This is obtained by the use of the declaration OPERATOR.

```plaintext
e.g. operator e, s, arctan;
```

adds the prefix operators \(c\), \(s\) and \(arctan\) to the system. (There are a number of reserved operators from the three types of operators, i.e. infix operators, mathematical operators and prefix operators already supplied in the system).

This allows symbols like \(c(x,y)\), \(s(a+b,z)\), \(arctan(u/v)\) to be used in expressions. To give a meaning to an operator symbol, or express some of its properties, LET statement can be used. As an example if we
require to evaluate the average of any two numbers \( x, y \) we may proceed by defining the operator average as follow:

\[
\text{operator average;}
\]

\[
\text{for all } x, y \text{ let } \text{average}(x, y) = (x+y)/2;
\]

Thus, if we use for the next command:

\[
m := \text{average}(3, 7);
\]

REDUCE will respond with

\[
M = 5
\]

Another example for the usefulness of this facility is that we may use it as a tool to simplify complicated algebraic expressions such as the product of the cosine of two angles \( \cos(m) \cdot \cos(n) \) into a sum of two trigonometric values

\[
\frac{\cos(m-n) + \cos(m+n)}{2}.
\]

In REDUCE, this can be carried out as follows:

\[
\text{operator c;}
\]

\[
\text{for all } m, n \text{ let } c(m) \cdot c(n) = (c(m-n) + c(m+n))/2;
\]

Thereafter, any product of two trigonometric cosines will be simplified into the sum of two related trigonometric values.

A very powerful feature of the REDUCE system is the ease with which matrix calculations can be performed and its handling of symbolic matrices. For example,

\[
\text{matrix } m(3,3);
\]

declares \( m \) to be a (3x3) matrix, and

\[
m := \text{mat}((a,b,c),(d,e,f),(p,q,r));
\]
gives it specific element values. Expressions which include \( m \) and make algebraic sense may now be evaluated, such as \( 1/m \) to give the inverse, \( \text{det}(m) \) to give the determinant of \( m \) and \( 3*m+k*m**3 \) to give another combined matrix.
A REDUCE program consists of a set of functional commands which are evaluated sequentially by the computer. For example, consider the problem of finding the inverse matrix $Q$ (say) of a $(4,4)$ matrix $P$ given by

\[
P = \begin{bmatrix}
d & b & c & 0 \\
b & d & b & c \\
c & b & d & b \\
0 & c & b & d
\end{bmatrix}
\]

and then evaluate the matrix $S$ (say) which results from multiplying the matrix $Q$ by the matrix $R$, where $R$ is given by:

\[
R = \begin{bmatrix}
a & b \\
-b & a \\
b & -a \\
a & b
\end{bmatrix}
\]

The input to the system REDUCE can be written as follows:

- **The Input**
  
  matrix $p(4,4), q(4,4), r(4,2), s(4,2)$;
  
  $p := \text{mat}((d,b,c,0),(b,d,b,c), (c,b,d,b),(0,c,b,d));$
  
  $r := \text{mat}((a,b),(-b,a),(b,-a),(a,b));$
  
  $q := 1/p;$

- **Comments**
  
  Specify the dimensions of the matrices $P$, $Q$, $R$ and $S$.
  
  Setting the value of matrix $P$.
  
  Setting the value of matrix $R$.
  
  Find the inverse of matrix $P$ and assign it to the matrix $Q$. 
Evaluate the matrix product of \( Q \) and \( R \) and assign the result to \( S \). More examples in the use of REDUCE programmes can be found in Appendices 4 and 6.

In many applications, it is desirable to load previously prepared REDUCE files into the system, or to write output on other files. REDUCE offers some commands for this purpose, two of these commands are \texttt{IN} and \texttt{OUT}. The command \texttt{IN} takes a list of file names as argument and directs the system to input each file (which should contain REDUCE statements and commands) into the system. For example,

\begin{verbatim}
in probl,"prob2";
\end{verbatim}

will first load file probl, then prob2. Files to be read using \texttt{IN} should end with \texttt{;END;}.

The command \texttt{OUT} takes a single file name as argument, and directs the output to the file from then on, until another \texttt{OUT} changes the output file, or \texttt{SHUT} closes it. For example,

\begin{verbatim}
out file01;
\end{verbatim}

will direct output to the file file01.

To get out from the REDUCE system we use the command \texttt{BYE}. This command stops the execution of REDUCE and returns us to the computer system monitor program.

There are several reasons why symbolic computations are useful in the context of modelling and simulation of field problems. Some of the reasons cited by Brown and Hearn [1978] are listed below.

1. Sometimes it is prohibitively expensive, or even impossible, to solve an essentially numerical problem by purely numerical means because it involves too many variables, requires greater accuracy, or
is presented in an ill-conditioned or intractable form. However, a symbolic transformation may reduce the dimensionality, evade a large source of round-off error, finesse the ill conditioning, and otherwise change the problem into one that can be solved by standard numerical methods.

2. The algebraic result obtained via symbolic computation can be subsequently evaluated over a wide range of parameter values.

3. Symbolic computation provides an opportunity for realizing the vital computational symbiosis between numerical experiments and symbolic theories.

4. Symbolic computation can be used to generate a needed numerical subroutine.

5. Finally, in the realm of partial differential equations, Cloutman and Fullerton (1977) have used symbolic multidimensional Taylor series expansions, computed by the ALTRAN system, to analyse the discretization and round-off errors of various numerical methods and also, more importantly, to eliminate inaccurate or unstable methods prior to coding and testing, and to develop methods in which the lowest order errors cancel each other out. A similar application to the ordinary differential equations will eventually be considered in most of the chapters to follow in this thesis.
CHAPTER 3

SURVEY OF ODE SOLVERS
3.1 INTRODUCTION

In the historical origins of the numerical methods for solving ordinary differential equations, alternative methods have been developed such as using numerical quadrature formula which employ tabular differences of some derivatives of the dependent variable. It is assumed that such differences are available to enable us to carry out the integration. Such is the case if we are in the midst of a numerical solution. However, at the start of the solution the only known numerical data may be the initial conditions, so that no significant tabular differences are at our disposal.

Several analytical methods are already available to establish numerical values of the solution of a given differential equation in the neighbourhood of the origin. In this chapter we will consider two of them, namely, the method of Taylor series and Picard's approximations.

(i) Taylor series method

Suppose that we are concerned with investigating the solution of the initial value problem

\[ y' = f(x,y) \]
\[ y(x) = y(a) \text{ when } x = a \]

in the neighbourhood of \( x = a \). If the function \( y(x) \) satisfying this equation is analytic in \( x \), it should be possible to expand it in a Taylor series of the form

\[
y(x) = y(a) + (x-a)y'(a) + \frac{1}{2!}(x-a)^2y''(a) + ... \]
\[
+ \frac{1}{p!}-(x-a)^py^{(p)}(a) + ... \]

(3.1.3)

Suppose, moreover, that the values of successive higher derivatives \( y^{(j)}(x) \) \( (j=2,3,...) \) at \( x=a \) can be determined in terms of the given
initial condition from equations obtained by repeated differentiation of (3.1.1). If so, the series (3.1.3) within the radius of its convergence will represent the corresponding particular solution of our initial value problem; and a polynomial obtained by truncating the Taylor series after a certain number of terms will represent the corresponding approximation to \( y(x) \), whose truncation error becomes equal to the remainder term of the truncated Taylor expansion.

This method of solution can be taken as a basis for numerically solving the initial value problem (3.1.1) with initial condition (3.1.2). In particular, by taking \( x-a=h \), (3.1.3) can be rewritten as

\[
y(a+h) = y(a) + hy'(a) + \frac{1}{2!} h^2 y''(a) + \frac{1}{3!} h^3 y'''(a) + \ldots
\]

\[
+ \frac{1}{p!} h^p y(p)(a) + \frac{1}{(p+1)!} h^{p+1} y(p+1)(\xi),
\]

\( \xi \in (a, a+h) \) (3.1.4)

By discarding the remainder term in (3.1.4) we call

\[
y_1 = y(a) + hy'(a) + \frac{1}{2!} h^2 y''(a) + \frac{1}{3!} h^3 y'''(a) + \ldots
\]

\[
+ \frac{1}{p!} h^p y(p)(a),
\]

(3.1.5)

a \( p \)th order approximation to \( y(a+h) \).

In (3.1.5) the value of \( y(a) \) is taken from (3.1.2) while the values of \( y'(a), y''(a), \ldots, y^{(n)}(a) \) are obtained from (3.1.1) and from successive differentiation of (3.1.1). After obtaining \( y_1 \), one then generates in an analogous fashion \( y_2 \) from \( y_1 \) and \( y_3 \) from \( y_2 \), and so forth, until the procedure terminates. The \( p \)th order Taylor series method may be summarised, then, by the recursion formula

\[
y_{n+1} = y_n + hy'_n + \frac{1}{2!} h^2 y''_n + \ldots + \frac{1}{p!} h^p y^{(p)}_n
\]

(3.1.6)

\( n = 0, 1, \ldots \).
where derivatives of order one and higher are obtained from (3.1.1) and from successive differentiation of (3.1.1), and where each $y_n$ is assumed to be exact.

As an example, let us consider the solution of a non-linear differential equation

$$\frac{d^2 y}{dx^2} + \frac{2dy}{xdx} + y^n = 0,$$

(3.1.7)

$n$ being a fixed constant, which is subject to the initial conditions

$$y(0) = 1 \text{ and } y'(0) = 0$$

(3.1.8)

This is the Emden's celebrated equation governing the structure of self-gravitating gas spheres in polytropic equilibrium (see, for example, Kopal [1961], p. 144); and unless their polytropic index $n$ happens to be 0, 1, or 5, the solution of (3.1.7) cannot be expressed in a closed form in terms of known functions, so that we can resort to the numerical integrations approach.

By writing (3.1.7) as a system of first order equations using the additional variable $z = y'$ we obtain

$$\begin{align*}
y' &= z, \quad y(0) = 1 \\
z' &= -\frac{2}{x}z - y^n, \quad z(0) = 0
\end{align*}$$

(3.1.9)

Differentiating (3.1.9) and solving for the highest derivative we find that

$$\begin{align*}
y'' &= z' \\
z'' &= -\frac{2z'}{x} + \left(\frac{2}{x^2} - nyn^{-1}\right)z
\end{align*}$$

(3.1.10)

$$\begin{align*}
y''' &= z'' \\
z''' &= -\frac{2z''}{x} + \left(\frac{4}{x^2} - nyn^{-1}\right)z' - \left[\frac{4}{x^3} + n(n-1)yn^{-2}\right]z
\end{align*}$$

(3.1.11)

e tc., so that from the first of equations (3.1.10) we have

$$y'' = -2\left(\frac{z}{x}\right)' - y_0^n.$$ 

(3.1.12)
Since, from the initial conditions, \( z_0 = 0 \), the ratio \((z/x)_0\) becomes indeterminate and of the form \(0/0\), but its limiting value is found by l'Hospital rule to be equal to \( y_0'' \), which inserted in (3.1.12) yields \( y_0'' = -1/3 \).

The second of equations (3.1.10) can be written as

\[
y'' = \frac{2}{x} \left( \frac{y'}{x} - y'' \right) - ny^{n-1} y',
\]

but since at \( x=0 \), we just found that \((y'/x)_0 = y_0''\), it follows that \( y_0'' = 0 \). By further substitution and differentiation of (3.1.11) and repeated application of the l'Hospital rule where necessary, we establish similarly that

\[
\begin{align*}
y_0^{(i\nu)} & = n/5, \\
y_0^{(\nu)} & = 0, \\
y_0^{(\nu l)} & = (5n-8n^2)/21,
\end{align*}
\]

etc., so that the Taylor expansion (3.1.3) in the present case assumes the form

\[
y = 1 - \frac{1}{3!} x^2 + \frac{n}{5!} x^4 - \frac{8n^2-5n}{3(7!)} x^6 + \ldots
\]

and its first four terms would permit us, for example, to evaluate \( y(0.4) \) for \( n=3 \) correctly to four decimal figures.

Note that if the recursion formula (3.1.6) was to be carried out, the subsequent stages would find no difficulty since the indeterminate form becomes absent as \( x \) proceeds to non-zero values.

The Taylor series method is one of the older methods for approximating a solution of an initial value problem. In the past it has not been highly regarded because it requires extensive symbolic manipulation in the determination of high-order derivatives. However, since symbolic manipulation is now advancing as a computer science discipline, the Taylor series method is returning to a position of
stature (Greenspan [1974], p. 29).

(ii) Picard's Approximations

The method of Taylor series as outlined in the foregoing section lends itself readily for an extrapolation of a solution of our differential equation from a given initial point whenever the function defined by this equation turns out to be analytic at the initial point. The feasibility of the method depends, of course, on the ease with which a given equation can be repeatedly differentiated, as well as on the rapidity with which the successive terms of the corresponding Taylor series are diminishing. Should, however, the given equation be complicated and (or) the convergence of its Taylor expansion be slow, a method based on its use may prove too tedious to be really practicable.

In the present section we will outline an alternative process known as Picard's approximations which can also be invoked for the construction of a particular solution of a given differential equation and which, in one respect, becomes a converse of the method of Taylor series. That is to say that Picard's approximations will prove to be primarily applicable to equations whose highest derivatives are readily integrable.

The great advantage of this method is its applicability to equations whose solutions are not necessarily analytic, and also the fact that it yields a particular solution in the form of series which always converge within a certain range of $x$ which can be specified in terms of the given characteristics of the problem.

Let us consider again the task of constructing the particular solution of (3.1.1) which satisfies the initial condition (3.1.2). This particular solution may at once be expressed in the symbolic form

$$y = y(a) + \int_{a}^{x} f(x,y)\,dx$$  \hspace{1cm} (3.1.15)
Let, moreover, $y_n(x) = y_n$ represent an nth approximation to our solution (by whichever method it was obtained), and let an (n+1)st approximation be defined by

$$y_{n+1} = y(a) + \int_a^x f(x, y_n) \, dx$$

(3.1.16)

which, unlike (3.1.15), does not constitute an integral equation but a quadrature. The question is whether $y_{n+1}$ as defined by equation (3.1.16) represents a closer approximation to $y$ than $y_n$ would. To answer this question, we subtract (3.1.15) from (3.1.16) and use the Mean Value Theorem to obtain

$$y_{n+1} - y = \int_a^x [f(x, y_n) - f(x, y)] \, dx = \int_a^x (y_n - y)(f_y)' \, dx$$

(3.1.17)

where we have abbreviated

$$(f_y)' = \frac{\partial}{\partial y} f(x, y + \theta[y_n - y]), \quad 0 < \theta < 1.$$ 

By setting $y_n - y = e_n$, we may rewrite (3.1.17) as

$$e_{n+1} = \int_a^x (f_y)' e_n \, dx$$

and, therefore, assert that

$$|e_{n+1}| < \int_a^x |(f_y)'| |e_n| \, dx.$$ 

If we let $e_n'$ to denote the greatest value of $|e_n|$ in the range of integration, and $M_n$ be similarly the maximum value of $|f_y|$ within the same interval, then it follows that

$$|e_{n+1}| < \int_a^x e_n' M_n \, dx = e_n' M_n (x-a).$$

(3.1.18)

At some point of the range, $|e_{n+1}|$ will assume its greatest value $e_{n+1}'$; but even at this point the inequality (3.1.18) still holds, so that we can write
\[
\begin{align*}
\varepsilon_{n+1}' &< \varepsilon_{n}' M_n (x-a), \\
\varepsilon_{n}' &< \varepsilon_{n-1}' M_{n-1} (x-a), \\
& \quad \vdots \\
\varepsilon_{2}' &< \varepsilon_{1}' M_1 (x-a).
\end{align*}
\] (3.1.19)

Since all \( \varepsilon_n \)'s as well as the \( M_n \)'s are, by definition, positive, we can multiply the right-hand sides of all inequalities in (3.1.19) to find that

\[
\varepsilon_{n+1}' \varepsilon_{n}' \varepsilon_{n-1}' \cdots \varepsilon_2' \varepsilon_1' \varepsilon_{n-2}' \cdots \varepsilon_1' M_n M_{n-1}' \cdots M_1 (x-a)^n \] (3.1.20)

Both sides of this latter inequality contain the product

\[
\varepsilon_{n}' \varepsilon_{n-1}' \cdots \varepsilon_2'
\]

in common. Cancelling it we may rewrite (3.1.20) as

\[
\varepsilon_{n+1}' < \varepsilon_1' M^n (x-a)^n
\]

where \( M \) denotes the greatest of the numbers \( M_n, M_{n-1}, \ldots, M_1 \) (i.e., \( M \) is the maximum value of \( |f_y| \) encountered in the range from \( a \) to \( x \)). Therefore, if we choose \( x \) so that \( M(x-a) < 1 \), i.e., if the range of integration is chosen so that \( (x-a) < M^{-1} \), the maximum error of the \( n \)th approximation can be made less than any preassigned quantity, provided only that \( n \) has been taken sufficiently large. This is true no matter how large \( \varepsilon_1' \) may have been (i.e., no matter how unsuitable a function we may have adopted for our first approximation).

If we recall now that \( y \) is defined as a solution of the differential equation (3.1.1), it remains to be proved that, with an increasing value of \( n \), our approximations to this solution, as defined by (3.1.16), not only converge, but converge to the solution of (3.3.1). In order to do so, let us differentiate (3.1.16) to obtain

\[
\frac{dy_{n+1}}{dx} = f(x, y_n).
\]

Subtracting \( f(x, y_{n+1}) \) from both sides of this equation, we establish by
use of equations (3.1.17) and what follows,

\[ \frac{dy_{n+1}}{dx} = f(x, y_{n+1}) = f(x, y) - f(x, y_{n+1}) \]

\[ \leq \frac{|y_n - y_n|}{|f_y|} \theta \]

\[ \leq |(y_{n+1} - y) - (y_n - y)| |f_y| \theta \]

\[ \leq (e_{n+1} - e_n) |f_y| \theta. \]

As the right-hand side of the foregoing inequalities can (by choosing a sufficiently large value of n) be made to approach zero as closely as we please, it follows that

\[ \lim \frac{dy_{n+1}}{dx} = f(x, y_{n+1}) = 0; \]

and because of the uniform convergence of our process, the limit and the derivative can be interchanged to assert that

\[ \frac{d}{dx} (\lim y_{n+1}) = f(x, \lim y_{n+1}) = 0, \]

qed.

Example

Suppose we are required to expand in this way the particular solution of

\[ \frac{dy}{dx} = y + 1 \] \hspace{1cm} (3.1.21)

subject to the initial condition

y(0) = 0.

In accordance with (3.1.16), we rewrite (3.1.21) as

\[ y = y(0) + \int_0^x y'(x)dx = \int_0^x (y + 1)dx. \]

Putting \( y = y_0 = 0 \) in place of \( y \) behind the integral sign we obtain the first approximation

\[ y_1 = \int_0^x (0 + 1)dx = x. \]

The second approximation follows by evaluating the same integral while
setting $y = y_1$ behind the integral sign, which leads to

$$y_2 = \int_0^x (x+1)dx = x + \frac{x^2}{2}.$$

The third approximation follows likewise from

$$y_3 = \int_0^x (x + \frac{x^2}{2} + 1)dx = x + \frac{x^2}{2} + \frac{x^3}{6};$$

and repeating the same process we readily establish that

$$y = x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots = e^x - 1,$$

as could be verified by direct integration of (3.1.21).
3.2 PRACTICAL NUMERICAL METHODS

As shown earlier, very few ordinary differential equations can be solved as finite combinations of elementary functions (National Physical Laboratory [1961] and Henrici [1962]). There are two basic approaches to the numerical approximation of differential equations. One is to represent an approximate solution by the sum of a finite number of independent functions, for example, a truncated power series or the first few terms of an expansion in orthogonal functions. These methods are usually better suited to hand computations, although there has been a lot of work on the application of Chebyshev polynomials to ordinary differential equations (Gear [1971]).

The second approach is the difference method. In this approach, the solution is approximated by its value at a sequence of discrete points called the mesh points. In much of our discussion we will assume that these points are equally spaced and called $x_n = a + nh$, where $h$ is the spacing between adjacent points called the step size. The end point will usually be called $x_N = b$, so that $N = (b-a)/h$.

However, the step size $h$ will be seen to affect the error introduced, and what may be a good size of $h$ in one region of the interval will not be suitable elsewhere. Consequently, we may use a variable step size, in which case we will have

$$x_{n+1} = x_n + h_n, \quad x_0 = a.$$

To begin our survey, as well as to have a brief idea of the numerical methods for solving ODE we present now three crude approximations for numerically solving ordinary differential equations. These crude methods will however eventually lead to a good approximation of the true solution of the differential equation. For the numerical analysis we form a grid
\[ G_N = \{ x_n | x_n = a + nh, \ n = 0, 1, \ldots, N; \ h = (b-a)/N \} \] (3.2.1)

for an arbitrary positive integer \( N \). We denote the approximate value of \( y(x_n) \) by \( y_n \). These three numerical methods are Euler's method:

\[
y_0 = y(a) \text{ for an initial value and } \quad y_{n+1} = y_n + hf(x_n, y_n) \text{ for } n = 0, 1, \ldots, N; \quad (3.2.2)
\]

the backward Euler method:

\[
y_0 = y(a) \text{ for an initial value and } \quad y_{n+1} = y_n + hf(x_{n+1}, y_{n+1}) \text{ for } n = 0, 1, \ldots, N; \quad (3.2.3)
\]

and the trapezoidal method:

\[
y_0 = y(a) \text{ for an initial value and } \quad y_{n+1} = y_n + \frac{h}{2} [f(x_n, y_n) + f(x_{n+1}, y_{n+1})] \quad (3.2.4)
\]

for \( n = 0, 1, \ldots, N \).

Whereas Euler's method is a straightforward or explicit computation, the backward Euler and trapezoidal methods are implicit equations requiring some special process to solve them. All these three methods are called discrete variable methods because an approximate solution is attempted only on a discrete set of points.

In general, a discrete variable method is also called a step by step method, a difference method, or a marching process and provides a rule for computing the approximation at step \( n \) to \( y(x_n) \) in terms of the values of \( y \) at \( x_{n-1} \) and possibly preceding points. Ideally, the solution could be represented by its actual value at each mesh point, so that it could be approximated to high accuracy by interpolation between the mesh points. However, two problems interfere with this ideal (Gear [1971], p.8). First, the exact solution of the differential equation is not, in general, known and cannot be calculated, so the solution to a different problem which can be calculated is sought. The difference between these two solutions will be called truncation error;
and secondly, numbers cannot be represented exactly in the numerical processes involved. The change introduced by this mechanism will be called round-off error.

Consequently, the solution from a step-by-step method will be represented by a finite number of finite precision numbers containing two sources of error, round-off and truncation. The difference methods are generally more suited for the automatic computation of general non-linear problems than are series expansion methods which we have discussed more earlier, and are the methods most frequently used in common computer subroutine libraries such as NAG, IMSL, etc.

In the next section we will consider the classical linear multistep methods and the general consideration of the concept of accuracy and stability.
3.3 CLASSICAL LINEAR MULTISTEP METHODS

A well-known family of numerical methods for solving ordinary differential equations is the class of linear multistep methods. Since many basic ideas which we make use of such as precision, accuracy, stability and convergence for numerical methods are so well portrayed by the theory of linear multistep methods, we shall now discuss the relevant theory at this point.

Most of the established results are quoted without proof. For details and proofs concerning the theory to be reviewed here, we refer to Henrici (1962), Isaacson and Keller (1966), Dahlquist, Bjorck and Anderson (1974) or to any one of a number of other standard texts.

3.3.1 The initial Value Problem

Let us consider the non-linear initial value problem

\[ y' = f(x, y), \quad y(a) = \eta, \] (3.3.1)

where \(y, f\) and \(\eta \in \mathbb{C}^n\) (i.e., are n-tuples of complex numbers). We seek a solution to (3.3.1) on the interval \(I\);

\[ I = \{ x \mid a \leq x \leq b; -\infty < a < b < \infty \}. \]

The class of functions \(f\), for which (3.3.1) is a well-posed problem is specified in the following definition.

**Definition 3.1**

\(f\) is said to be an \(L\)-function (alternatively, \(f \in L\)) if for all \(x \in I\) and \(y, z \in \mathbb{C}^n\), there exists a constant \(L\) (the Lipschitz constant) such that

\[ \|f(x, y) - f(x, z)\| < L\|y - z\|. \]

We may now state the following existence and uniqueness theorem for the problem (3.3.1).
Theorem 3.1

If \( f \) is continuous in \( x \) for \( x \in I \) and if \( f \) is an L-function, the problem (3.3.1) has one and only one solution in \( I \).

The exact solution \( y(x) \) is approximated by \( y_n, \, n=0,1,\ldots,N \) at a set of points \( a=x_0 < x_1 < \ldots < x_N=b \), and the quantity \( e_n = y(x_n) - y_n \) is called the global discretization error.

The general \( k \)-step linear multistep method is defined by the equation

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

(3.3.2)

where \( \alpha_j \) and \( \beta_j \) are constants; we assume that \( \alpha_k \neq 0 \) and that not both \( \alpha_0 \) and \( \beta_0 \) are zero. Thus the problem of determining the solution \( y(x) \) of the initial value problem (3.3.1) is replaced by that of finding a sequence \( \{y_n\} \) which satisfies the difference equation (3.3.2) which is in general non-linear.

It is convenient to associate with (3.3.2) the first and second characteristic polynomials \( \rho(s) \) and \( \sigma(s) \), where

\[
\rho(s) = \sum_{j=0}^{k} \alpha_j s^j, \\
\sigma(s) = \sum_{j=0}^{k} \beta_j s^j,
\]

(3.3.3) (3.3.4)

which can be used interchangeably to define the multi-step method. The linear multistep method is said to be explicit if \( \beta_k = 0 \), in which case \( y_{n+k} \) is computed in terms of the values at the \( k \)-previous mesh points. When \( \beta_k \neq 0 \) the formula is said to be implicit, since \( y_{n+k} \) occurs as an argument in \( f_{n+k} = f(x_{n+k}, y_{n+k}) \) in the right hand side of (3.3.2). In this implicit case (3.3.2) represents a system of finite equations which must be solved by numerical procedures.

The linear multistep method allows the step by step determination of \( y_n \), provided that the values of \( y_0, \ldots, y_{k-1} \) are known. These
so-called starting values are determined by some independent procedure which may be called the starting procedure. The Taylor series, for example, could be implemented to provide such a procedure. As a notation for the starting procedure, we write

\[ y_m = S_m(h), \quad m = 0, 1, \ldots, k-1. \tag{3.3.5} \]

Basic properties of the starting procedure are given in the following two definitions.

**Definition 3.2**

The starting procedure is said to be bounded if there exists a positive constant \( M \) such that \( ||S_m(h)|| < M \) for all sufficiently small \( h \).

**Definition 3.3**

The starting procedure is said to be compatible if

\[ \lim_{h \to 0} S_m(h) = \eta, \quad m = 0, 1, \ldots, k-1. \]

Let

\[ h_0 = \alpha_k (\beta_k L)^{-1}. \]

The existence and uniqueness of the numerical procedure which we have just described is the subject of the following theorem, which we quote without proof.

**Theorem 3.2**

A linear multistep formula has one and only one solution \( y_n, \ n \in J_h \) for all starting procedures \( S_m(h) \) if \( 0 \leq h \leq h_0 \).

In practice it is required that the solution of the difference equation (3.3.2) should be a good approximation to the solution of the equation (3.3.1). Associated to the LMM (3.3.2) we now define the linear difference operator \( L \) as follows

\[ L[z(x);h] = \sum_{j=0}^{k} \alpha_j z(x+jh) - h \sum_{j=0}^{k} \beta_j z'(x+jh) \tag{3.3.6} \]

This operator may be applied to any difference function \( z(x) \). By
applying $L$ to $y(x)$, $L[y(x);h]$ is called the local truncation error (LTE) of the LLM which serves as a measure of the difference of the two sides of (3.3.2). Furthermore, if $y(x)$ is sufficiently differentiable the terms of $L[y(x);h]$ may be expanded in powers of $h$. Using Taylor's expansion we have,

$$L[y(x);h] = c_0y(x) + c_1y^{(1)}(x) + \ldots + c_qy^{(q)}(x) + \ldots,$$

where the coefficients $c_i$, $i=0,1,\ldots$ are constants which are independent of $h$ and the function $y(x)$.

From a simple calculation the $c_i$ are defined as:

$$c_0 = \alpha_0 + \alpha_1 + \ldots + \alpha_k$$
$$c_1 = \alpha_1 + 2\alpha_2 + \ldots + k\alpha_k - (\beta_0 + \beta_1 + \ldots + \beta_k)$$
$$\vdots$$
$$\vdots$$
$$c_q = \frac{1}{q!}(-\alpha_1 + 2\alpha_2 + \ldots + k\alpha_k) - \frac{1}{(q-1)!}(-\beta_1 + 2\beta_2 + \ldots + k\beta_k)$$

These formulae can be used to derive a LMM of a given structure and order.

Order of a LMM is given in the following definition.

**Definition 3.4**

The linear multistep method (3.3.2) is said to have degree of precision $p$ if the coefficients $\alpha_j$ and $\beta_j$ of $L$ are chosen so that $c_j = 0$, $j=0,1,\ldots,p$, and $c_{p+1} \neq 0$, i.e.,

$$L[y(x);h] = c_{p+1}h^{p+1}y^{(p+1)}(x) + O(h^{p+2}).$$

The term $c_{p+1}h^{p+1}y^{(p+1)}(x)$ is called the principal local truncation error.

**Examples of Linear Multistep Methods**

The following are some of the well-known linear multistep methods.

(1) Adams' method:

$$y_{n+k} - y_{n+k-1} - h \sum_{j=0}^{k} \beta_j f_{n+j} = 0.$$
\( B_k \neq 0 \): Adams-Moulton, \( k = 1 \): Trapezoidal formula.
\( B_k = 0 \): Adams-Bashforth, \( k = 1 \): Euler's formula.

(ii) Nystrom's method:

\[
y_{n+k} - y_{n+k-2} - h \sum_{j=0}^{k} B_j f_{n+j} = 0.
\]

\( k = 2 \): mid-point formula.

(iii) Method of Newton-Cotes:

\[
y_{n+k} - y_n - h \sum_{j=0}^{k} B_j f_{n+j} = 0.
\]

\( k = 2 \): Simpson's formula.

(iv) Backward differentiation formula:

\[
\sum_{j=0}^{k} \alpha_j y_{n+j} - h B_k f_{n+k} = 0.
\]

3.3.2 Consistency, Zero-stability and Convergence

Consistency of a linear multistep method is specified in the following definition.

Definition 3.5

A linear multistep method is said to be consistent if it has order at least one.

Alternatively, by applying the characteristic polynomials we incorporate in the following theorem the notion of consistency.

Theorem 3.3

A linear multistep method is consistent if and only if

\[ L(1) = \rho(1) = 0 \]

and

\[ L(x) = h(\rho'(1) - \sigma(1)) = 0. \]

Thus, for a consistent method, the first characteristic polynomial \( \rho(s) \) always has a root +1, this root will be called the principal root and labelled \( s_1 \).
The behaviour of the solution \( \{ y_n \} \), generated by the linear multistep method (3.3.2) as \( h \) tends to zero is of fundamental importance. It is necessary that the solution of (3.3.1) at any fixed point \( x \), tends to \( y(x) \) as \( h \to 0 \), if the starting values tend to their true values, i.e., obtained from a compatible starting procedures. This notion of convergence is characterized in the following definition.

**Definition 3.6**

A linear multistep method is convergent if, for all initial value problems, the following hold,

\[
\lim_{h \to 0} \frac{1}{nh} ||y_n - y(x_n)|| = 0 \text{ for all } x \in [a,b],
\]

and all the solution \( \{ y_n \} \) of the difference equation (3.3.2) satisfying the condition \( y_p = \eta(h) \), for which

\[
\lim_{h \to 0} ||\eta_p - \eta|| = 0, \quad n = 0, 1, \ldots, k-1.
\]

i.e., for all compatible starting procedures.

The zero-stability of a linear multistep method is characterized in the following definition.

**Definition 3.7**

The linear multistep method is said to be zero-stable if no root of the first characteristic polynomial \( \rho(s) \), has a modulus greater than 1, and every root with modulus 1 is simple.

Finally, we arrive at the following fundamental theorem of Dahlquist [1956, 1959] on linear multistep methods.

**Theorem 3.4**

The necessary and sufficient conditions for a linear multistep method to be convergent are that it is consistent and zero-stable.
3.3.3 Weak Stability

From the above although a L.M.M. used in the solution of the initial value problem can be convergent, this property gives no insight to the validity of the results obtained in practice.

An important property is the stability of the method at the stepsize used in computation. The essential idea is that the effects of perturbances should remain bounded for stable differential equations. In other words the stability properties of the algorithm at finite non-zero stepsize should be of the same qualitative kind as the stability properties of the differential equation.

Although this is a complex subject to analyse, the normal approach is to consider a simple test equation to arrive at some means of obtaining useful results about the property of a method.

Usually the equation,

\[ y' = \lambda y, \quad y(0) = 0, \lambda \text{ complex,} \]

is considered.

Employing a LMM, it can easily be shown that the behaviour of the solution depends on the roots of the stability polynomial:

\[ \pi(r, h') = \rho(r) - h' \sigma(r) = 0, \]

where \( h' = h\lambda \).

This can be formalised into the following definition.

**Definition 3.8**

A LMM is said to be absolutely stable in a region \( R \) of the complex plane if \( h\lambda \in R \Rightarrow |r_j| < 1, \ j=1,2,\ldots,k \), where \( r_j \) are the roots of (3.3.7). The region in the complex plane for which the method is absolutely stable is called the region of absolute stability.

The general methods for finding the regions of absolute stability is discussed, for example, in Lambert [1973].
3.4 METHOD OF ABSOLUTE STABILITY

In this section, we discuss the methods specially designed for solving the stiff problem. These proceed by imposing strong stability notions onto the traditional linear multistep methods.

We begin with the well-known notion of A-stability in Section 3.4.1. Then in Section 3.4.2, we discuss A(α)-stability and stiff stability, two of a large number of alternative stability notions associated with stiff problems. Finally in Section 3.4.3, we consider the problem of solving the equations which are generated by the numerical methods themselves.

3.4.1 Stiff Initial-Value Problems and Stiffness Ratio

"Stiffness" is a property of a mathematical problem (not of numerical solution method). Consider the system of m-coupled first-order ordinary differential equations

\[ y' = f(x, y). \]  

(3.4.1)

Let \( y(x), \ x \in [a, b] \), denote the exact solution to equation (3.4.1) which satisfies the given initial condition \( y(a) = \eta \).

To determine whether or not this initial-value problem is stiff, we need to know something about the nature of the solutions to equation (3.4.1) in the neighbourhood of the particular solution \( y(x) \). In such a neighbourhood, equation (3.4.1) may be closely approximated by the linearized, variational equations

\[ y' - J(x)[y-y(x)] - f(x,y(x)) = 0 \]  

(3.4.2)

where \( J(x) \) denotes the Jacobian matrix of partial derivatives \( \partial f/\partial y \), evaluated at \( (x,y(x)) \). If the variation of \( J(x) \) in an interval of \( x \) is sufficiently small, the localized eigensolutions of equations (3.4.2) are approximately the exponentials \( e^{\lambda_i x} \), where the \( \lambda_i = \lambda_i(x) \) are the local eigenvalues (assumed distinct) of the Jacobian matrix \( J(x) \).

Thus, the solution \( y \) to the equation (3.4.1) in a neighbourhood of
the exact solution $y(x)$ at $x$ are of the forms

$$y = y(x) + \sum_{i=1}^{n} c_i e^{\lambda_i x} z_i,$$

(3.4.3)

where the $c_i$ are constants and the $z_i$ are the eigenvalues of $J(x)$. The eigensolutions $e^{\lambda_i x}$ characterize the local response of the system to small changes or perturbation about $y(x)$. We shall assume that the system is locally stable, so that

$$Re(\lambda_i) < 0 \quad (i=1, 2, \ldots, m),$$

and the transient eigensolutions decay with increasing $x$ at rates which are proportional to

$$1/Re(-\lambda_i)$$

termed the local "time constants" of the system. It is the range in the local values of the "time constants" of a problem that provides a measure of stiffness.

**Definition 3.9** (cf. Lambert [1973])

The initial-value problem

$$y' = f(x, y), \quad y(a) = \eta, \quad x \in [a, b],$$

is said to be stiff in an interval $I \subset [a, b]$ if, for $x \in I$,

1. $Re(\lambda_i) < 0 \quad (i=1, 2, \ldots, m)$; and
2. $S(x) = \max_{i=1,m} Re(-\lambda_i) / \min_{i=1,m} Re(-\lambda_i) >> 0$,

where the $\lambda_i$ are the eigenvalues of $\partial f/\partial y$ evaluated on the solution $y(x)$ at $x$.

The ratio $S(x)$ may be termed the local "stiffness ratio" of the problem (Lambert [1973]). Problems may be considered to be marginally stiff if $S(x)$ is $O(10)$, while stiffness ratios up to $O(10^6)$ are not uncommon in practical problems arising in such fields as chemical kinetics, process control, electrical circuit theory and in many computer aided design techniques, particularly network analysis and simulation.
A stiff problem is often referred to in the literature as a problem with "widely differing time constants" or as a system "with a large Lipschitz constant".

3.4.2 Stability for Stiff Problems

Let us now consider the difficulties that arise in attempting to obtain a numerical approximation to the solution $y(x)$ of a stiff problem. The basic problem is that of numerical stability.

For the absolute stability of numerical solutions to the system of equation (3.4.1), it is necessary to use a step size $h$, such that every one of the (complex) values $h_1^i = h \lambda_1^i (i=1,2,\ldots,m)$, where the $\lambda_1^i$ are the eigenvalues of $J(x)$, lies within the region of absolute stability of the numerical method. For methods with finite absolute stability regions, the step size is thus restricted to the order of magnitude of the smallest time constant of the system and as the range of integration may well exceed the value of the largest time constant, the number of integration steps required may be comparable to the stiffness ratio of the system.

To overcome this stability limitation on the step size, numerical methods have been sought that possess regions of absolute stability that extend to infinity in the half-plane $\text{Re}(h\lambda) < 0$. Several definitions describing stability properties suitable for stiff systems have been proposed, i.e.,

Definition 3.10 (Dahlquist [1963])

A numerical method is said to be $A$-stable if its region of absolute stability contains the whole of the left-hand half-plane $\text{Re}(h\lambda) < 0$.

To determine which linear multistep methods are $A$-stable, we note that when the test equation $y' = \lambda y$ is inserted into the linear multistep formula, a linear difference equation results:
The characteristic equation corresponding to (3.4.4) is

\[ \pi(r; h') = \rho(r) - h'\sigma(r) = 0. \]  

(3.4.5)

\( \pi \) defines a \( k \)-valued mapping of \( h' \) into \( r \). The inverse of this mapping

\[ h'(r) = \rho(r)/\sigma(r), \]  

(3.4.6)

defines a single valued mapping of \( r \) into \( h' \).

Having made these observations, we may state the following proposition connecting A-stability and the mapping \( \pi \).

**Proposition 3.1**

Let \( r_i, i=1, \ldots, k \) be the roots of \( \pi(r; h')=0 \). Then the following three statements are equivalent.

(a) a linear multistep method is A-stable.

(b) \( \text{Re}(h') < 0 \) \( \Rightarrow \) \( |r_i| < 1, i=1, \ldots, k \).

(c) \( |r| \geq 1 \) \( \Rightarrow \text{Re}(h'(r)) \geq 0 \).

**Examples of A-stable Methods**

We now give several examples of A-stable methods.

(1) The trapezoidal formula:

\[ y_{n+1} - y_n - \frac{h}{2} (f_{n+1} + f_n) = 0. \]

\[ \rho(r) = r-1, \sigma(r) = \frac{r+1}{2} \]

\[ \text{Re}(h'(r)) = \frac{|r|^2 - 1}{|r+1|^2} \]

Thus, \( \text{Re} h'(r) > 0 \) for \( |r| > 1 \), and the root of \( \rho \) on \( |r| = 1 \) is simple.

(2) The backward Euler formula:

\[ y_{n+1} - y_n - h f_{n+1} = 0. \]

\[ \rho(r) = r-1, \sigma(r) = r. \]
The roots of \( \rho(r) \) are the \( k \)th roots of unity.

Note further that \( \rho(0) = 0, \rho'(0) = o(1) = k \), implying the consistency of this method. This example shows the existence of linear multistep methods which are consistent and \( A \)-stable for any \( k \) (i.e., for any number of steps).

The \( A \)-stability property was originally formulated for linear multistep methods, and it proves to be a very restrictive requirement for such methods. Dahlquist [1963] also proved that no explicit linear multistep method could be \( A \)-stable, and that the order of an implicit linear multistep method could not exceed two, the trapezoidal rule being the most accurate such method. In consequence, \( A \)-stability has been most widely used in studying implicit one-step methods (Ehle [1968], Axelsson [1969], Chipman [1971]) and various explicit generalizations of Runge-Kutta and linear multistep methods (Treanor [1966], Norsett [1969], Lambert and Sigurdsson [1972]).

As we have seen, the family of linear multistep methods is so desirable because of its simple form for both computation and analysis that the limitations imposed on this family by \( A \)-stability has made a great impact. In an attempt to preserve the family for the solution of stiff differential equations, a sequence of weakened forms of absolute stability were invented in stages. We will examine two of these. First we will consider the notion of \( A(\alpha) \)-stability, and following that, we will review the so-called stiffly stable methods.

Examination of the failure of the linear multistep methods to be
A-stable shows in many cases that failure occurs for values of $\lambda$ in the test equation which are nearly purely imaginary. It is then a simple step to abandon such values of $\lambda$ (i.e., highly oscillatory solutions) and to seek the analogue of A-stability corresponding to a subset of the left half complex plane which in particular excludes the imaginary axis.

Definition 3.11 (Widlund [1967])

A numerical method is said to be A($\alpha$)-stable, $\alpha \in (0, \pi/2)$, if its region of absolute stability contains the infinite wedge $|\arg(-\lambda)| < \alpha$ (Figure 3.1). A method is said to be A(0)-stable if it is A($\alpha$)-stable for some (sufficiently small) $\alpha \in (0, \pi/2)$.

Note that in the above definition a given eigenvalue $\lambda$ either lies inside or outside the wedge, regardless of the positive stepsize used. Widlund proved that no explicit linear multistep method could be A(0)-stable, and showed that implicit linear multistep methods of orders three and four existed that were A($\alpha$)-stable for any $\alpha < \pi/2$.

The above definitions are concerned only with stability. Gear [1971] defines a more complex property, involving both stability and accuracy of approximations to the exponential eigensolutions.

Definition 3.12 (Gear [1971])

A method is said to be stiffly stable if it is
(a) absolutely stable in the region $R_1$ ($\text{Re}(h\lambda) < D$) and
(b) accurate in the region $R_2$ ($D < \text{Re}(h\lambda) < \delta$, $|\text{Im}(h\lambda)| < \theta$); (see Figure 3.2).

The reasoning behind this definition is explained as follows. If the solution $y(x)$ contains transient eigensolutions $e^{\lambda x}$, these eigensolutions with small time constants decay rapidly and quickly become negligible, while the eigensolutions with larger time constants may contribute significantly to the solution. Using a stiffly stable method the stepsize $h$ may be chosen so that the negligible (sometimes
Figure 3.1: $A(\alpha)$-stability (Widlund [1967])

Figure 3.2: Stiff stability (Gear [1971])
called parasitic) eigensolutions are approximated stably with $h\lambda \in R_1$, while for the significant eigensolutions with larger time constants, $h\lambda \in R_2$ ensures an accurate approximation.

Among the methods which are stiffly stable are the so-called backward differentiation formulae (see Section 5.1.4 of Henrici [1962]). Low order formulae of this type provide the basis of the well known Gear's package for solving stiff differential equation. (See Hindmarsh [1974]). In particular, Gear [1969] shows that the $k$-step methods of order $k$ with second characteristic polynomial $\sigma(r) = r^k$ are stiffly stable for $k \leq 6$ for some $D$, $a$ and $\theta$.

The first three backward differentiation formulae are:

$k = 1$: 
$$y_{n+1} - y_n = hf_{n+1}$$

$k = 2$: 
$$y_{n+2} - \frac{4}{3}y_{n+1} + \frac{1}{3}y_n = \frac{2}{3}hf_{n+2}$$

$k = 3$: 
$$y_{n+3} - \frac{18}{11}y_{n+2} + \frac{9}{11}y_{n+1} - \frac{2}{11}y_n = \frac{6}{11}hf_{n+3}$$

In Table 3.1 we list the coefficients of $k$th order $k$-step Gear's method, scaled so that $\alpha_k = 1$.

<table>
<thead>
<tr>
<th>$k$</th>
<th>$\beta_k$</th>
<th>$\alpha_0$</th>
<th>$\alpha_1$</th>
<th>$\alpha_2$</th>
<th>$\alpha_3$</th>
<th>$\alpha_4$</th>
<th>$\alpha_5$</th>
<th>$\alpha_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$\frac{2}{3}$</td>
<td>1</td>
<td>-4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$\frac{6}{11}$</td>
<td>1</td>
<td>-$\frac{18}{11}$</td>
<td>$\frac{9}{11}$</td>
<td>-$\frac{2}{11}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$\frac{12}{25}$</td>
<td>1</td>
<td>-$\frac{48}{25}$</td>
<td>$\frac{36}{25}$</td>
<td>-$\frac{16}{25}$</td>
<td>$\frac{3}{25}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$\frac{60}{137}$</td>
<td>1</td>
<td>-$\frac{300}{137}$</td>
<td>$\frac{300}{137}$</td>
<td>-$\frac{200}{137}$</td>
<td>$\frac{75}{137}$</td>
<td>-$\frac{12}{137}$</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>$\frac{60}{147}$</td>
<td>1</td>
<td>-$\frac{360}{147}$</td>
<td>$\frac{450}{147}$</td>
<td>-$\frac{400}{147}$</td>
<td>$\frac{225}{147}$</td>
<td>-$\frac{72}{147}$</td>
<td>$\frac{10}{147}$</td>
</tr>
</tbody>
</table>

Table 3.1
In Figure 3.3, we plot the regions of stability for the first three methods. The methods are stable outside the curves indicated. In Figure 3.4, regions of stability are plotted for some stiffly stable methods of orders 4, 5 and 6 (see Gear [1971]).

In some respects the A-stability property is not stringent enough, and studies of A-stable one-step methods have produced a number of additional stability concepts. A one-step method applied to the equation \( y' = \lambda y \) gives an approximation

\[ y_{n+1} = Q(h\lambda) y_n \]

to the solution

\[ y(x_{n+1}) = e^{h\lambda} y(x_n), \]

and A-stability ensures that \( |Q(h\lambda)| < 1 \) for all \( \text{Re}(h\lambda) < 0 \). For many A-stable one-step methods, however, \( Q(h\lambda) \) is such that

\[ |Q(h\lambda)| \rightarrow 1 \text{ as } \text{Re}(h\lambda) \rightarrow -\infty, \]

so that numerical solution to the rapidly decaying eigensolutions with very small time constants may decay only very slowly. It is well-known that the trapezoidal rule has this unfortunate property (Rosenbrock [1963]). This leads us to define a further property, which has been variously termed L-stability (Axelsson [1969]), and strong A-stability (Chipman [1971], Axelsson [1972]).

**Definition 3.13**

A one-step method is said to be L-stable if it is A-stable, and when applied to the equation \( y' = \lambda y \) with \( \text{Re}(\lambda) < 0 \), it gives

\[ y_{n+1} = Q(h\lambda)y_n, \]

where \( |Q(h\lambda)| \rightarrow 0 \text{ as } \text{Re}(h\lambda) \rightarrow -\infty. \)

Gourlay [1970] has also noted that one-step methods with \( |Q(h\lambda)| \rightarrow 1 \text{ as } \text{Re}(h\lambda) \rightarrow -\infty \) may prove unstable when used to solve

\[ y' = \lambda(x) y, \] \hspace{1cm} (3.4.7)

a test equation appropriate to the variational equations (3.4.2), and he proposes a modification for the trapezoidal rule which is always stable for equation (3.4.7).
Figure 3.3: Regions of absolute stability for backward differentiation methods of step 1, 2 and 3.

Figure 3.4: Regions of absolute stability for the $k$-step backward differentiation methods, $k=4,5,6$. 
3.5 RUNGE-KUTTA METHODS AND ROSENBRICK METHODS

In this section we discuss the well-known class of Runge-Kutta methods, and show that in this class of methods we may also find A-stable methods of higher order. Then we consider a variant of these methods due to Rosenbrock which have desirable computational properties.

3.5.1 Runge-Kutta Methods with R-levels

The original Runge-Kutta methods have been better known in the present time as classical Runge-Kutta methods. By this we mean methods which were first derived in the pre-computer era. The choice of coefficients in these methods was largely motivated by the need to produce methods convenient for desk computation. Such methods are not necessarily the most suitable for use on an automatic computer.

The general R-stage Runge-Kutta method for solving $y' = f(x,y)$ is defined by

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

$$k_1 = f(x,y),$$

$$k_i = f(x+hb_i, y+hj\sum_{j=1}^{i-1} a_{ij} k_j), \quad i=2,3,...,R,$$

$$b_i = \sum_{j=1}^{i-1} a_{ij}, \quad i=2,3,...,R.$$ (3.5.1)

This is the representation of the formula which is widely used by most earlier authors. Eventually, there is a great deal of tedious manipulation involved in deriving Runge-Kutta methods of high order. Furthermore since this thesis will also mainly be concerned with the development of the non-linear Runge-Kutta formulae, we shall begin this earlier discussion by using the more brief representation of the formulae.
In this brief representation we will write the system to be solved as a family of autonomous equations

\[ y_i'(x) = f_i(y_1(x), y_2(x), \ldots, y_m(x)) \quad i=1,2,\ldots,m, \]  

(3.5.2)

which we will also write more compactly as

\[ y'(x) = f(y(x)) \]  

(3.5.3)

Note that there is no loss of generality in restricting ourselves to a system (3.5.2) or (3.5.3) in which \( x \) does not appear as one of the arguments of \( f_1, f_2, \ldots, f_m \) since, if necessary, an additional differential equation can be added to the family for which the solution is \( x \). An example is \( y' = 1 \) with initial condition \( y(0) = 0 \).

Thus, the general \( R \)-stage Runge-Kutta method is defined by the following relations,

\[ y_{n+1} = y_n + h \sum_{i=1}^{R} w_i k_i \]  

(3.5.4-i)

\[ k_i = f(y_n + h \sum_{j=1}^{i-1} a_{ij} k_j), \quad i=1,2,\ldots,R. \]  

(3.5.4-1i)

These relations are used to define an approximation, \( y_{n+1} \) to \( y(x_{n+1}) \) in terms of an approximation to \( y(x_n) \), denoted simply by \( y_n \), in (3.5.4) and in all future function evaluations. The coefficients \( w_i, a_{ij} \), \( i,j = 1,2,\ldots,R \) are to be determined by a procedure which we now describe.

### 3.5.2 Determination of the coefficients

By using (3.5.3), we may write the following list of formal relations,

\[
\begin{align*}
y' &= f, \\
y'' &= f f_y, \\
y''' &= f^2 f_{yy} + ff_y^2, \\
y^{(iv)} &= f^3 f_{yyy} + 4f^2 f_y f_{yy} + ff_y^3,
\end{align*}
\]

(3.5.5)
\[ y^{(v)} = f^4_{y y y y} + 7f^3_{y y y} f_y + 11f^2_{y y} f_y^2 + f^3_{y y} + ff^4_y, \]

\[ \ldots \]

\[ y^{(r)} = \sum_{r=1}^{p_r} a_{rs} f_{rs} \]

where \( f_y \) is the Jacobian, an array of order 2, \( f_{yy} \) is the Hessian, an array of order 3, ... The \( F_{rs} \), \( r=1,2,\ldots; s=1,2,\ldots, p_r \) are called the elementary differentials. For each index \( r \), there are \( p_r \) such differentials. For example, \( p_1 = 1, p_2 = 1, p_3 = 2, p_4 = 4, \ldots \), and

\[ F_{11} = f, F_{21} = ff_y, F_{31} = f^2_{yy}, F_{32} = ff_y. \]

Now let \( y \) and \( y_n \) denote the exact value of \( y \) at \( x_{n+1} \) and \( x_n \) respectively. By substituting the relations (3.5.5) into the formal statement

\[ y = y + \sum_{r=1}^{\infty} h^r y^{(r)}/r!, \]

of Taylor's theorem gives

\[ y = y + hf + \frac{1}{2} hf^2 + \frac{1}{6} (f^2_{yy} + ff_y^2) + \ldots \]

or

\[ y - y_n = \sum_{r=1}^{\infty} h^r \left( \sum_{s=1}^{p_r} \frac{a_{rs} f_{rs}}{r!} \right) \]

Now if we formally develop each \( k_i, i=1,\ldots, R \) as in equation (3.5.4-ii), in a series, we may write (3.5.4-i) as

\[ h \sum_{i=1}^{R} b_i k_i = \sum_{r=1}^{\infty} \frac{h^r}{(r-1)!} \left( \sum_{s=1}^{p_r} \frac{b_{rs} d_{rs} f_{rs}}{r!} \right) \]

Here the \( b_{rs} \) are numerical coefficients while \( d_{rs} \) are functions of the \( w_i \) and \( a_{ij} \).

For a Runge-Kutta process to be of order of precision \( p \), it is necessary that the formal series in (3.5.7) and (3.5.8) agree to \( p \) terms. Thus we find
as a set of \( \sum_{r=1}^{p} r \) equations for the determination of the \( R(R+1) \) coefficients \( a_{ij}, i,j=1,\ldots,R \). These equations are usually called the equations of conditions. Having solved these equations we normally display all the coefficients required to define the method as entries of a matrix of coefficients called the generating matrix. The following definition provides a convenient way of expressing a more general Runge-Kutta process.

**Definition 3.14**

Consider the problem of solving the initial value problem \( y' = f(y), y(a) = \eta \). Let \( A \) be an \((R+1)\times R\)-matrix with elements \( a_{ij}, i=1(1)R+1, j=1(1)R \). The one-step \((R+1)\)-stage forward step procedure defined by the relations,

\[
y_{n+1} = y_n + h \sum_{i=1}^{R} a_{R+1,i} k_1
\]

where

\[
k_1 = f(y_n + h \sum_{j=1}^{R} a_{ij} k_j), i=1,2,\ldots,R
\]

is called an \( R \)-stage Runge-Kutta procedure. The matrix \( A \) is called its symbol generating matrix.

In the following we define three classes of Runge-Kutta processes.

**Definition 3.15**

The symbol generating matrix \( A \) (and the related Runge-Kutta process) is said to be explicit if its elements \( a_{ij} \) satisfy

\[
a_{ij} = 0 \text{ for } j > i, i = 1(1)R.
\]

It is said to be semi-explicit if \( a_{ij} = 0, j > i \) and it is said to be implicit otherwise.

Note that the \( M \) equations in (3.5.9) are not independent, and so it is usually possible to satisfy them with a number \( N \) of coefficients considerably smaller than \( M \).
Example

Let us illustrate by deriving the 3-stage formulae of order 3 (i.e., $R=p=3$). For easy manipulation we use the variables $a_1, a_2$ and $a_3$ to represent $a_{11}, a_{21}$ and $a_{22}$ respectively. Our aim is to determine these parameters together with $w_1, w_2$ and $w_3$ so that the formula

$$y_{n+1} = y_n + h(w_1 k_1 + w_2 k_2 + w_3 k_3)$$  \tag{3.5.11}$$

where

$$k_1 = f(y)$$
$$k_2 = f(y + h a_1 k_1)$$
$$k_3 = f(y + h a_2 k_1 + h a_3 k_2)$$

will agree with the Taylor series (3.5.7) to the third degree.

Expanding the $k_i$'s in Taylor series about $y$ and writing $f(y)$ as $f$, $\frac{\partial f}{\partial y}$ as $f_y$, etc., we have

$$k_1 = f$$
$$k_2 = f + (h a_1 k_1) f_y + \frac{1}{2} (h a_1 k_1)^2 f_{yy} + ...$$
$$= f + h a_1 f f_y + \frac{1}{2} h^2 a_1^2 f^2 f_{yy} + ...$$
$$k_3 = f + (h a_2 k_1 + h a_3 k_2) f_y + \frac{1}{2} (h a_2 k_1 + h a_3 k_2)^2 f_{yy} + ...$$
$$= f + h (a_2 k_1 + a_3 k_2) f_y + \frac{1}{2} h^2 (a_2 k_1 + a_3 k_2)^2 f_{yy} + ...$$

but

$$(a_2 k_1 + a_3 k_2)^2 = a_2 f + a_3 (f + h a_1 f f_y + ...)$$
$$= (a_2 + a_3) f + h a_1 a_3 f f_y + h^2 a_1^2 a_3 f^2 f_{yy} + ...$$

so that

$$(a_2 k_1 + a_3 k_2)^2 = (a_2 + a_3)^2 f^2 + h(2a_1 a_3 (a_2 + a_3) f^2 f_y) + ...$$
$$(a_2 k_1 + a_3 k_2)^3 = (a_2 + a_3)^3 f^3 + ...$$

Therefore, after some algebraic manipulation, we have

$$k_3 = f + h (a_2 + a_3) f f_y + h^2 [a_1 a_3 f_y^2 + \frac{1}{2} (a_2 + a_3)^2 f^2 f_{yy}] + h^3 \left[ \frac{1}{2} a_1^2 a_3^2 f_{yy} f_{yy} + a_1 a_3 (a_2 + a_3) f^2 f_y f_{yy} + \frac{1}{6} (a_2 + a_3)^3 f_{yyy} + ... \right]$$

By substituting these into (3.5.11) we obtain
\[ y_{n+1} = y_n + h[w_1 f + w_2 (f + h a_1 f f_y) + \frac{1}{2} a_1^2 f^2 f_{yy} + \ldots) + w_3 (f + h (a_2 + a_3) f f_y) + h^2 [a_1 a_3 f f_y + \frac{1}{2} (a_2 + a_3)^2 f^2 f_{yy} + \ldots] \]
\[ = y_n + h(w_1 + w_2 + w_3) f + h^2 [w_2 a_1 + w_3 (a_2 + a_3)] f f_y + h^3 [\frac{1}{2} w_2 a_1^2 + \frac{1}{2} (a_2 + a_3)^2] f^2 f_{yy} + w_3 a_1 a_3 f f_y + \ldots \]

By comparing this expansion with the Taylor series expansion of \( y(x_{n+1}) \) about \( x_n \) we obtain the following equations of conditions:

\[ \begin{align*}
    h f: & \quad w_1 + w_2 + w_3 = 1 \\
    h^2 f f_y: & \quad w_2 a_1 + w_3 (a_2 + a_3) = \frac{1}{2} \\
    h^3 f f_y: & \quad w_3 a_1 a_3 = \frac{1}{6} \\
    h^3 f^2 f_{yy}: & \quad \frac{1}{2} w_2 a_1^2 + \frac{1}{2} w_3 (a_2 + a_3)^2 = \frac{1}{6}
\end{align*} \]

The problem remains to solve for the values of the parameters \( a_1, a_2, a_3, w_1, w_2 \) and \( w_3 \) from the above four equations. This is an underdetermined system of non-linear equations, and in general more than one solutions is possible. By introducing two more additional equations, we may employ some numerical process to obtain a set of the possible solutions. [Note that for a system of non-linear equations, the solution is not necessarily unique even though the number of equations equals the number of variables.] It is of great advantage if we could utilise this freedom to possibly obtain a formula that is convenient to compute, represent, or to have the least truncation error. In view of \( b_1 = \sum b_{ij} \) in (3.5.1) it is customary that we fix the values of \( b_2 \) and \( b_3 \). By fixing \( b_2 = a_1 = 1/2 \) and \( b_3 = a_2 + a_3 = 1 \) will effectively mean that we can compute the function at the middle and at the end of the interval \([x_n, x_{n+1}]\).

By solving these equations simultaneously we obtain

\[ a_1 = 1/2, a_2 = -1, a_3 = 2, w_1 = 1/6, w_2 = 2/3, w_3 = 1/6; \]

and the resulting method is given by
\[ y_{n+1} = y_n + h \left( k_1 + 4k_2 + k_3 \right) / 6, \]
\[ k_1 = f(x_n, y_n) \]
\[ k_2 = f(x_n + h/2, y_n + hk_1/2), \]
\[ k_3 = f(x_n + h, y_n - hk_1 + 2hk_2), \]

which is known as the Kutta's third-order rule. In a concise form this method is represented by the symbol generating matrix

\[
A = \begin{bmatrix}
0 & 0 & 0 \\
1/2 & 0 & 0 \\
-1 & 2 & 0 \\
1/6 & 2/3 & 1/6
\end{bmatrix}
\]

**Note:**

Another useful representation of the Runge-Kutta processes currently used by many authors is the array representation

\[
\begin{bmatrix}
\mathbf{b} \\
\mathbf{w}^T
\end{bmatrix} = \mathbf{A}' \mathbf{w}^T
\]

where \( \mathbf{A}' \) is the \( R \times R \) matrix whose elements are \( a_{ij} \), \( i, j = 1(1)R \), \( \mathbf{w}^T = [w_1, w_2, \ldots, w_R] \), and \( \mathbf{b} \) is the vector whose components are given by

\[
b_i = \sum_{j=1}^{R} a_{ij} w_j.
\]

With this representation, the above method is written as

\[
\begin{bmatrix}
0 & 0 & 0 & 0 \\
1/2 & 1/2 & 0 & 0 \\
1 & -1 & 2 & 0 \\
1/6 & 2/3 & 1/6
\end{bmatrix}
\]
3.5.3 Semi-explicit Processes and the Method of Rosenbrock

There are A-stable methods amongst the implicit and semi-explicit Runge-Kutta methods. For the purpose of solving stiff systems it looks worthwhile to consider them specifically. However, the implicit processes lead to methods which are difficult to apply in general, because at each step of the integration, the $k_i$, $i = 1, \ldots, R$ must be determined as the solution of the system of $R$ non-linear equations (3.4.10-ii). The semi-explicit processes, on the other hand, require the solution of a non-linear triangular system, the $j$-th equation of which contains only the unknown $k_i$, $i = 1, \ldots, j$. Thus, each equation in turn need only be solved for one unknown, i.e., the $i$-th equation for $k_i$, $i = 1, \ldots, R$.

In view of a simpler implementation of the semi-explicit process, let us consider the case in more great detail. The evaluation of the $k_i$, $i = 1, \ldots, R$ can be further simplified if we can replace the solution procedure by a single application of the Newton-Raphson iteration. From (3.5.10-ii) we have, for the semi-explicit case,

$$k_i = f(y_n + h \sum_{j=1}^{i-1} a_{ij}k_j)$$

$$= f(y_n + h \sum_{j=1}^{i-1} a_{ij}k_j + ha_{ii}k_i)$$

Now, we approximate the function by the first two terms in the Taylor series expansion of $f$ about $y_n + h \sum_{j=1}^{i-1} a_{ij}k_j$ to obtain

$$k_i = f(y_n + h \sum_{j=1}^{i-1} a_{ij}k_j) + ha_{ii}k_if_y(y_n + h \sum_{j=1}^{i-1} a_{ij}k_j)$$

Rearranging, we have

$$k_i(1 - ha_{ii}f_y(y_n + h \sum_{j=1}^{i-1} a_{ij}k_j)) = f(y_n + h \sum_{j=1}^{i-1} a_{ij}k_j)$$

or
Now, since the solution for $k_i$ is only an approximate one and that it depends on $a_{ij}$ which can be varied so that the resulting method may attain maximum possible accuracy (or to satisfy a certain stability requirement) we may therefore alter the $a_{ij}$ in the second summation by another coefficient $c_{ij}$ and then determine the values of these parameters. With this modification the resulting method becomes (in matrix form for a system),

$$k_i = (I - h a_{1i} f_y (y_n + h \sum_{j=1}^{i-1} a_{ij} k_j))^{-1} f(y_n + h \sum_{j=1}^{i-1} a_{ij} k_j)$$

for $i = 1, \ldots, R$

where $I$ is the $m \times m$ identity matrix and $m$ is the number of equations in the system. This is called the Rosenbrock linearised semi-explicit Runge-Kutta process. (See Rosenbrock [1962]).

**Example** Two-level third order Rosenbrock method.

In this case, the computations involved are

$$y_{n+1} = y_n + h (w_1 k_1 + w_2 k_2)$$

$$k_1 = [I - h a_{11} f_y (y_n)]^{-1} f(y_n)$$

$$k_2 = [I - h a_{22} f_y (y_n + h a_{21} k_1)]^{-1} f(y_n + h a_{21} k_1)$$

Expanding the function in the Taylor series about $y_n$ and making use of the usual short notations, we have

$$k_1 = (1 + h a_{11} f_y + h^2 a_{11} f_y^2 + h^3 a_{11} f_y^3 + \ldots) f$$

$$= f + h a_{11} f_y + h^2 a_{11} f_y^2 + h^3 a_{11} f_y^3 + \ldots$$

and
\[ k_2 = [1 - h a_{22}(f_y + h a_{21} k_1 f_{yy} + h^2 a_{21}^2 k_1^2 f_{yyy} + \ldots) \\
+ h^2 a_{22}^2 (f_y + h a_{21} k_1 f_{yy} + \ldots)^2 + \ldots f + h c_{21} k_1 f_y \\
+ \frac{1}{6} h^2 c_{21}^2 k_1^2 f_{yy} + \ldots) \\
= f + h(a_{22}+c_{21})f f_y + h^2[(a_{11} c_{21}+a_{22} c_{21}) f f_y \\
+ (a_{22} a_{21}+4c_{21}^2) f^2 f_{yy}] + O(h^3). \]

Therefore, after some algebraic manipulation (3.5.13) becomes

\[ y_{n+1} = y_n + hf(w_1+w_2) + h^2 f f_y (w_1 a_{11}+w_2 (a_{22}+c_{21})) \\
+ h^3\{[w_1 a_{11}^2+w_2 (a_{22} a_{11} c_{21}+a_{22} c_{21}^2)] f f_y \\
+ w_2 (a_{22} a_{21}+4c_{21}^2) f^2 f_{yy} \} + \ldots \]

This is then compared with the Taylor series expansion (3.5.7). If the two series must agree to the terms in \( h^3 \) (i.e. to obtain a method of third order) we must have

\[
\begin{align*}
&hf: \quad w_1 + w_2 = 1 \\
&h^2 f f_y: \quad w_1 a_{11} + w_2 (a_{22}+c_{21}) = 1/2 \\
&h^3 f f_y^2: \quad w_1 a_{11}^2 + w_2 (a_{22}^2+(a_{11}+a_{22}) c_{21}) = 1/6 \\
&h^3 f^2 f_{yy}: \quad w_2 (a_{22} a_{21}+4c_{21}^2) = 1/6 
\end{align*}
\]

Therefore we have four equations in six unknowns \( w_1, w_2, a_{11}, a_{21}, a_{22} \) and \( c_{21} \). The solution is not unique.

A particular solution due to Rosenbrock is given by

\[
\begin{align*}
a_{11} &= 1 + \frac{1}{\sqrt{6}} \\
a_{22} &= 1 - \frac{1}{\sqrt{6}} \\
a_{21} &= c_{21} = \frac{-6-\sqrt{6+\frac{58+20}{6}}}{6+2/6} \\
w_1 &= -0.413154 \\
w_2 &= 1.413154.
\end{align*}
\]

Since the solution is not unique we can also set \( a_{22} = a_{11} \) and \( a_{21} = 0 \). With this choice of constraints the two matrices in equation (3.5.13) which must be inverted become identical and thus considerably reduces the computation per step. The corresponding equations (3.5.14) then become
These equations have two solutions, one of which is given by Calahan as

\[ w_1 = 3/4, \quad w_2 = 1/4, \quad a_{11} = \frac{3+\sqrt{3}}{6}, \quad c_{21} = -\frac{2\sqrt{3}}{3}. \]

(See Calahan [1967]).

3.5.4 Stability Properties of the Runge-Kutta Processes

In this section we will discuss the absolute stability region of the Runge-Kutta methods. First we consider the stability regions for explicit Runge-Kutta methods, and later the absolute stability regions for the semi-explicit methods discussed in the preceding section.

We consider the equation \( y' = \lambda y \) for complex values of \( \lambda \). If we examine the \( R \)-stage Runge-Kutta process for \( y' = \lambda y \), we find

\[
\begin{align*}
k_1 &= f(y_n) \\
&= \lambda y_n \\
k_2 &= f(y_n + ha_{11}k_1) \\
&= \lambda(1 + a_{11}h\lambda)y_n \\
k_3 &= f(y_n + ha_{21}k_1 + ha_{22}k_2) \\
&= \lambda[1+a_{21}h\lambda + a_{22}h\lambda(1+a_{11}h\lambda)]y_n.
\end{align*}
\]

Substituting into equation (3.5.4-1) we obtain

\[
y_{n+1} = y_n + h[\lambda y_n + \lambda(1+a_{11}h\lambda)y_n + \\
\lambda[1+a_{21}h\lambda + a_{22}h\lambda(1+a_{11}h\lambda)]y_n + ...]
\]

By rearrangement of the right-hand side, this equation can be written in the form

\[
y_{n+1} = Q(h\lambda)y_n,
\]

where the polynomial \( Q(h\lambda) \) is determined by a proper substitution of \( a_{ij} \) depending on the formula being examined. The region of absolute stability is determined by first plotting the locus of \( Q(h\lambda) = 1 \) in the complex plane and then examine in which side of the curve the
polynomial $Q(h\lambda)$ satisfies

$$|Q(h\lambda)| < 1.$$  

In the following, we examine the stability of some selected explicit Runge-Kutta methods of order one through four which are represented by the given formulae. The corresponding polynomials $Q(h\lambda)$ are also given.

First order (Euler's rule):

$$k_1 = f(y_n)$$
$$y_{n+1} = y_n + h k_1$$
$$Q(h\lambda) = 1 + h\lambda$$

Second order (Improved Euler method):

$$k_1 = f(y_n)$$
$$k_2 = f(y_n + h k_1)$$
$$y_{n+1} = y_n + \frac{1}{2} h (k_1 + k_2)$$
$$Q(h\lambda) = 1 + h\lambda + \frac{1}{2} (h\lambda)^2$$

Third order (Heun's third-order formula):

$$k_1 = f(y_n)$$
$$k_2 = f(y_n + \frac{1}{3} h k_1)$$
$$k_3 = f(y_n + \frac{2}{3} h k_2)$$
$$y_{n+1} = y_n + \frac{h}{4} (k_1 + 3 k_3)$$
$$Q(h\lambda) = 1 + h\lambda + \frac{(h\lambda)^2}{2} + \frac{(h\lambda)^3}{6}.$$

Fourth order (fourth-order Runge-Kutta method):

$$k_1 = f(y_n)$$
$$k_2 = f(y_n + \frac{1}{2} h k_1)$$
$$k_3 = f(y_n + \frac{1}{2} h k_2)$$
$$k_4 = f(y_n + h k_3)$$
$$y_{n+1} = y_n + \frac{h}{6} (k_1 + 2 k_2 + 2 k_3 + k_4)$$
\[Q(h\lambda) = 1 + h\lambda + \frac{(h\lambda)^2}{2} + \frac{(h\lambda)^3}{6} + \frac{(h\lambda)^4}{24}\]

In general, the explicit Runge-Kutta method of order \(R\) has its polynomial \(Q(h\lambda)\) similar to that of the Taylor series expansion of \(e^{h\lambda}\).

The region of absolute stability for the above four methods are shown in Figure 3.5.

For the semi-explicit methods given by Rosenbrock and Calahan, the regions of absolute stability are obtained as follows. We apply the formula (3.5.12) to the test equation \(y' = \lambda y\) to obtain

\[k_1 = \frac{\lambda y_n}{1 - h\lambda a_{11}}\]

\[k_2 = \frac{\lambda (y_n + h\lambda c_{21}y_n / (1 - h\lambda a_{11}))}{1 - h\lambda a_{22}}\]

Therefore,

\[y_{n+1} = y_n + \left[\frac{h\lambda w_1}{1 - h\lambda a_{11}} + \frac{h\lambda w_2}{1 - h\lambda a_{22}} + \frac{h^2\lambda^2 w_{2c_{21}}}{(1 - h\lambda a_{11})(1 - h\lambda a_{22})}\right]y_n\]

or

\[Q(h\lambda) = 1 + \frac{h\lambda w_1}{1 - h\lambda a_{11}} + \frac{h\lambda w_2}{1 - h\lambda a_{22}} + \frac{h^2\lambda^2 w_{2c_{21}}}{(1 - h\lambda a_{11})(1 - h\lambda a_{22})}\]

By substituting the suitable parameters for the Rosenbrock and Calahan methods the regions of absolute stability can be obtained by using the procedure already explained earlier. These two regions are shown in Figure 3.6 where the methods are stable outside the closed contours. Clearly, since these stability regions contain the complex half-plane \(h\lambda < 0\), the two methods are \(A\)-stable.
Figure 3.5: Stability regions for the classical Runge-Kutta methods. The method of order \( k \) is stable inside the region indicated.

Figure 3.6: Stability regions for the methods of Rosenbrock and Calahan. The methods are stable outside the closed contours.
CHAPTER FOUR

ELEMENTARY INTEGRATION METHODS BASED ON ARITHMETIC MEAN (AM) AND GEOMETRIC MEAN (GM) FORMULAE
4.1 THE WEIGHTED GM FORMULA

4.1.1 Introduction

The class of linear one-step methods of order one is given by

\[ y_{n+1} = y_n + h\theta f_n + (1-\theta)f_{n+1}, \]

often referred to as the 'θ-method'. (See Lambert [1973], p.240). It can be shown that the method has a truncation error of (in terms of θ)

\[ T_{n+1} = (\theta-\frac{1}{2})h^2 y''_n + (\frac{\theta}{2} - \frac{1}{3})h^3 y'''_n + O(h^4). \]

This error is the smallest when \( \theta = \frac{1}{2} \), in which case the method has order 2, called the Trapezoidal rule whose truncation error is \( -\frac{1}{12} h^3 y'''_n \). Further, we can also show that the method \( (4.1.1) \) is A-stable if and only if \( \theta \leq \frac{1}{2} \).

In this section we will study the equivalent formulae in the geometric sense.

4.1.2 Class of nonlinear methods based on Euler formulae

The θ-method as described above can be obtained as follows. Consider the following two formulae of Euler

\[ y_{n+1} - y_n = hy'_n \]
\[ y_{n+1} - y_n = hy'_{n+1} \]

The first formula being the forward Euler and the second is the backward Euler formula. By taking a suitable linear combination of these two formulae, a class of consistent linear methods in the form of \( (4.1.1) \) is obtained. In particular by taking the arithmetic mean of the two formulae we have the well-known Trapezoidal rule

\[ y_{n+1} = y_n + \frac{h}{2} (y'_n + y'_{n+1}) \]

In general, if we multiply \( (4.1.2) \) by θ and \( (4.1.3) \) by \( (1-\theta) \) and add
together we obtain the generalised form of weighted Euler formula

\[ y_{n+1} = y_n + h[\theta y_n' + (1-\theta)y_{n+1}'] \]  

(4.1.5)

By putting \( \theta = 0 \) in (4.1.5) we obtain the special case of Euler's method, while putting \( \theta = 1 \) we obtain the backward Euler method.

In the following we will study the equivalent formulae in the geometric sense. A similar analysis as above is possible for the non-linear case if, instead of taking the arithmetic mean, we take the geometric means of the formulae. In particular, the nonlinear equivalent of the Trapezoidal formula would be of the form

\[ y_{n+1} = y_n + h\sqrt{y_n'y_{n+1}'} \]  

(4.1.6)

We shall refer to this formula as the GM formula and can be similarly generalised by taking the powers of equations (4.1.2) and (4.1.3), i.e.,

\[ (y_{n+1} - y_n)^\theta = (hy_n')^\theta \]  

(4.1.7)

\[ (y_{n+1} - y_n)^{1-\theta} = (hy_{n+1}')^{1-\theta} \]  

(4.1.8)

By multiplying (4.1.7) and (4.1.8) we obtain

\[ y_{n+1} - y_n = h(y_n')^\theta (y_{n+1}')^{1-\theta} \]  

(4.1.9)

Formula (4.1.9) is the equivalent nonlinear form of equation (4.1.5) and contains as special cases Euler (\( \theta = 1 \)), backward Euler (\( \theta = 0 \)) and formula (4.1.6) (\( \theta = \frac{1}{2} \)).

4.1.3 Accuracy and stability analysis of formula (4.1.9)

We note from the Taylor series expansion of \( y(x_{n+1}) \) about \( x_n \), an approximation of which is represented by \( y_{n+1}' \), the expansion for \( y'_{n+1} \) can be written as

\[ y'_{n+1} = y_n' + hy_n'' + \frac{h^2}{2} y_n''' + \ldots \]

\[ = y_n' [1 + \frac{h}{y_n'} + \frac{h^2}{2y_n''} + \ldots] \]
\[ y_n'[1 + \left( \frac{hy''_n}{y_n'} + \frac{h^2y''_n}{2y_n'} + \ldots \right)] \]

Therefore, we have (by putting \(1-\theta = \mu\) for a shorter notation)

\[(y_{n+1}')^\mu = (y_n')^\mu [1 + \left( \frac{hy''_n}{y_n'} + \frac{h^2y''_n}{2y_n'} + \ldots \right)]^\mu \]

(4.1.10)

The terms in the square bracket of (4.1.10) are in the form \((1+x)^\mu\), where \(x\) is the expressions contained in the curly bracket. We know, from the Binomial theorem, that

\[(1+x)^\mu = 1 + \mu x + \frac{\mu(\mu-1)x^2}{2} + \frac{\mu(\mu-1)(\mu-2)x^3}{6} + \ldots\]

Therefore (4.1.10) can be expanded to

\[(y_{n+1}')^\mu = (y_n')^\mu [1 + \mu(\frac{hy''_n}{y_n'} + \frac{h^2y''_n}{2y_n'} + \ldots) + \frac{\mu(\mu-1)h^2y''_n}{2y_n'} + \ldots] \]

(4.1.11)

By using this expansion, the product of \((y_n')^{1-\mu}(y_{n+1}')^\mu\) can be written as

\[(y_n')^{1-\mu}(y_{n+1}')^\mu = (y_n')^{1-\mu}(y_n')^\mu \{ \ldots \} \]

\[= y_n'[\ldots] \]

\[= y_n'[hy''_n + \frac{h^2y''_n}{2} + \mu(\mu-1)y''_n \frac{2}{2y_n'} + \ldots] \]

Substituting this into (4.1.9) we obtain
The Taylor series expansion for $y(x_{n+1})$ about $x_n$ is

$$y(x_{n+1}) = y_n + hy_n' + \frac{h^2y_n''}{2} + \frac{h^3y_n'''}{6} + \ldots$$

Hence,

$$y(x_{n+1}) - y_{n+1} = h^2(\frac{1}{4} - \mu)y_n'' + h^3[(\frac{1}{6} - \frac{\mu}{2})y_n''' - \frac{\mu(\mu-1)y_n''^2}{2y_n'}] + \ldots$$

$$= h^2(\theta - \frac{1}{2})y_n'' + h^3[(\frac{\theta}{3} - \frac{1}{3})y_n''' - \frac{\theta(\theta-1)y_n''^2}{2y_n'}] + \ldots$$

(after putting $\mu = 1-\theta$).

It can be clearly seen that the method agrees with the Taylor series expansion up to the first order terms unless $\theta=\frac{1}{2}$ in which case the second order terms also match and the principal error term becomes

$$T_{n+1} = h^3[(\frac{1}{4} - \frac{1}{3})y_n''' - \frac{\frac{1}{4}(\frac{1}{4}-1)y_n''^2}{2y_n'}] + \ldots$$

$$= h^3[\frac{-y_n''}{12} + \frac{y_n''^2}{8y_n'}] \quad (4.1.12)$$

Therefore the method (4.1.9) when used has least error with $\theta=\frac{1}{2}$. This result was also verified by experiments.

We may compare this formula with the slightly different version of the formula mentioned by Lambert and Shaw [1965], viz.,

$$y_{n+1} - y_n = \frac{h^2f_nf_{n+1}}{y_{n+1} - y_n} \quad (4.1.13)$$

which has a truncation error

$$T_{n+1} = h^3[\frac{-y_n''}{6} + \frac{y_n''^2}{4y_n'}]$$
Obviously the GM formula is more accurate in that its truncation error is half that of formula (4.1.13). The GM formula also appears to possess the advantage of less rounding errors in computing the increment function when the increment is small.

**Stability Analysis**

To study the stability property of method (4.1.9) we apply the formula to the test equation \( y' = \lambda y \). This will result in the following difference equation

\[
y_{n+1} = y_n + h(\lambda y_n)\theta(\lambda y_{n+1})^{(1-\theta)}
\]

(4.1.14)

Since this equation is non-linear and the analysis for general value of \( \theta \) rather difficult, we treat equation (4.1.14) separately for some individually given values of \( \theta \). In particular if \( \theta = \frac{1}{2} \), equation (4.1.14) becomes

\[
y_{n+1} = y_n + h\lambda y_n y_{n+1}
\]

or

\[
\frac{y_{n+1}}{y_n} = 1 + h\lambda \frac{y_{n+1}}{y_n}
\]

Writing \( y_{n+1}/y_n = Q_n \) we have

\[
Q_n = 1 + h\lambda \sqrt{Q_n}
\]

and further writing \( \sqrt{Q_n} = A_n \) we obtain the quadratic equation

\[
A_n^2 = 1 + h\lambda A_n
\]

or

\[
A_n^2 - h\lambda A_n - 1 = 0
\]

(4.1.15)

Absolute stability requires that

\[
|\frac{y_{n+1}}{y_n}| = |Q_n| = |A_n^2| < 1
\]

or, similarly,

\[
|A_n| < 1.
\]
From (4.1.15) we write the roots as

\[ A_n = \frac{h \lambda \pm \sqrt{h^2 \lambda^2 + 4}}{2} \]

Taking only the positive sign we have

\[ A_n = \frac{h \lambda + \sqrt{h^2 \lambda^2 + 4}}{2} \]

The condition \(|A_n| < 1\) implies

\[ |h \lambda + \sqrt{h^2 \lambda^2 + 4}| < 2 \]

To see for what values of \(h \lambda\) this inequality is valid, we examine the inequality from two angles.

(a) \(h \lambda\) is real

Letting \(h \lambda = x\) where \(x\) is real, the inequality becomes

\[ |x + \sqrt{x^2 + 4}| < 2. \]

By plotting the function \(f(x) = x + \sqrt{x^2 + 4}\) against \(x\) we see that \(|f(x)| < 2\) for all \(x < 0\) and \(f(x) \geq 2\) for all \(x > 0\).

(b) \(h \lambda\) is pure imaginary

Letting \(h \lambda = iy\) where \(y\) is real, the inequality becomes

\[ |iy + \sqrt{(iy)^2 + 4}| < 2. \]

\[ \Rightarrow |iy + \sqrt{4 + y^2}| < 2 \]

\[ \Rightarrow \sqrt{y^2 + (4-y^2)} < 2 \]

\[ \Rightarrow \sqrt{4} < 2 \]

or \(\pm 2 < 2\).

The last relationship will be true if we put the equality sign as well. This suggests that the imaginary axis of the complex plane is the boundary for the region of absolute stability of the method. The method is thus absolutely stable for \(h \lambda\) lying on the left half of the complex plane.
Fig. 4.1(a): Contours of $f(z) = \frac{z + \sqrt{z^2 + 4}}{2}$

Fig. 4.1(b): Contours of $f(z) = \frac{z - \sqrt{z^2 + 4}}{2}$
Figure 4.1(c): The surface defined by $\frac{z+\sqrt{z^2+4}}{2}$.
This result was also verified graphically by plotting the contour of
\[ f(z) = \left| \frac{z + \sqrt{z^2 + 4}}{2} \right|, \quad z \text{ complex} \quad (4.1.16) \]
using the GINOSURF library routine FUNCON which draws a number of equally spaced annotated contours conveniently covering the height range of the surface within the defined region. The contours of \( f(z) \) as given by equation (4.1.16) is shown in Figure 4.1(a) where the height, \( t \), satisfies the inequality \( 0 \leq t \leq 1 \) for \( \text{Re}(z) \leq 0 \). If we take the square root sign in (4.1.16) as negative, we obtain the contours as shown in Figure 4.1(b) which are approximately the mirror image of Figure 4.1(a) about the real axis. For clarity, the corresponding surface defined by \( |(z+\sqrt{z^2+4})/2| \) is drawn by using the GINOSURF routine FUNGRD and is shown in Figure 4.1(c). This suggests that when the value in the square root sign is negative we should use the negative sign of the evaluated square root, and obtain an equally stable process.

4.1.4 Stability consideration of the Q-method for stiff problems

With a numerical method, errors in the approximation are generated at each step and propagated from step to step. For the application to stiff problems we will now examine the errors with respect to the model stiff equation
\[ y' = \lambda(y-p(x)) + p'(x), \quad y(a) = \eta \quad (4.1.17) \]
with \( \eta \) and \( \lambda \) constants and \( \lambda < 0 \). The analytical solution is
\[ y(x) = (\eta-p(a))\exp[\lambda(x-a)] + p(x). \]
The first term goes to zero rapidly, so that the solution \( y(x) \) approaches \( p(x) \) as \( x-a \) increases.

Classical forward step numerical methods frequently give approximations that diverge from the stiff equation's true solution. We will illustrate this phenomena by the use of Euler's method. By defining the \textit{global error} as
\[ \delta_n = y_n - y(x_n) \quad (4.1.18) \]

we have for Euler's method

\[
\delta_{n+1} = y_{n+1} - y(x_{n+1}) \\
= y_n + hy_n' - y(x_{n+1}) \\
= y_n + h\lambda(y_n - p(x_n) + p'(x_n)) - y(x_{n+1}) \\
= y_n (1+h\lambda) + h[-\lambda p(x_n) + p'(x_n)] - y(x_{n+1}) \\
= y_n (1+h\lambda) + h[-\lambda y(x_n) + y'(x_n)] - y(x_{n+1}) \\
\text{(using (4.1.17) after rearrangement).}
\]

\[
= [y_n - y(x_n)](1+h\lambda) + [y(x_n) + hy'(x_n) - y(x_{n+1})] \\
= (1+h\lambda)\delta_n + \frac{h^2 y''(\xi)}{2} \leq \xi \leq x_{n+1}
\]

Thus, the global error at the \((n+1)\)st step comes from the error propagated from the \(n\)th step and from the local truncation error. The propagated error is characterised by the coefficient of \(\delta_n\) that relates to stability. Since absolute stability means that present errors are not magnified in subsequent steps, the propagated error is not magnified in Euler's method if \(|1+h\lambda| < 1\) or \(-2 < h\lambda < 0\) on the real axis.

The local truncation error, on the other hand, expresses the local accuracy in the amount the method misrepresents the differential equation. It is also a restriction on the size of \(h\). While accuracy dictates step size for most problems, stability dictates the step size for stiff problems. All general purpose methods solving stiff differential equations are necessarily implicit. The backward Euler method is the simplest of these. For this method we have (after some manipulation as in the case of Euler's method)

\[
\delta_{n+1} = \frac{\delta_n}{1-h\lambda} + \frac{h^2 y''(\xi)}{2(1-h\lambda)}
\]

In this case, the propagation error is not magnified if \(|1/(1-h\lambda)| < 1\), i.e., \(h\lambda < 0\) or \(h\lambda > 2\) on the real axis of \(h\).
In a similar way we can deduce that, for the Trapezoidal method,

\[
\delta_{n+1} = \left( \frac{1 + h\lambda/2}{1 - h\lambda/2} \right) \delta_n + \frac{h^3 y''''(\xi)}{12(1-h\lambda/2)}
\]

This means that the propagation of the errors is not amplified if

\[
\left| \frac{1 + h\lambda/2}{1 - h\lambda/2} \right| < 1,
\]

or equivalently if \( h\lambda < 0 \).

Also, for the stability analysis of the GM formula we can deduce that the global error at the \((n+1)\)st step can be written as

\[
\delta_{n+1} = \left( \frac{h\lambda + \sqrt{h^2 \lambda^2 + 4}}{2} \right)^2 \delta_n
\]

\[+ \psi(h\lambda) h^3 \left[ \frac{1}{12} y''_n + \frac{1}{6} \frac{y_{n+1}''}{y_n'} \right]
\]

In this case the propagation error is not amplified if

\[
\left| \frac{h\lambda + \sqrt{h^2 \lambda^2 + 4}}{2} \right| < 1,
\]

i.e. if \( h\lambda < 0 \) on the real axis. Thus the magnification factors for the four methods are given by

(1) Euler's method

\[
Q(h\lambda) = 1 + h\lambda
\]

(2) Backward Euler's method

\[
Q(h\lambda) = \frac{1}{1 - h\lambda}
\]

(3) Trapezoidal method

\[
Q(h\lambda) = \frac{1 + h\lambda/2}{1 - h\lambda/2}
\]
To compare the regions of the absolute stability in the complex plane for these four methods, we plot the curve for which $Q(h\lambda)$ has the value 1. The regions can be seen in Figure 4.2 where the stable regions are shown as indicated. In the case of the trapezoidal method the boundary is the imaginary axis, whereas in the case of the GM formula the boundary is the imaginary axis for $h\lambda$ near the pole. For $|\text{Im}(h\lambda)| > \delta$, the boundary is a vertical line whose equation is given by $\text{Re}(h\lambda) = \epsilon$, where $\delta$ and $\epsilon$ are positive fixed real numbers.

Thus, from the definition of A-stability, the backward Euler, the trapezoidal and the GM methods are A-stable.

However, this does not appear to be restrictive enough. For the stiff equation it is desired that $Q(h\lambda) \to 0$ for a very large step size, i.e. as $h\lambda \to -\infty$. This is the L-stability requirement (see Definition 3.13 of Chapter 3). For backward Euler's method $Q(h\lambda) \to 0$ as $h\lambda \to -\infty$. For the Trapezoidal method we have $Q(h\lambda) \to -1$ and for the the GM method $Q(h\lambda) \to 0$ as $h\lambda \to -\infty$. Since the latter is not very obvious, we may derive it as follows:

$$\lim_{h\lambda \to -\infty} \left[ \frac{h\lambda + \sqrt{h^2 \lambda^2 + 4}}{2} \right]^2 = \lim_{x \to \infty} \left[ \frac{-x + \sqrt{x^2 + 4}}{2} \right]^2 = \lim_{x \to \infty} \left[ \frac{-x + \sqrt{1 + \frac{4}{x^2}}}{2} \right]^2 = 0$$

Therefore, of the four methods we have just discussed, only the backward Euler and the GM methods are L-stable.

To conclude our findings, we list some of the major points relating to the stability of the methods for solving stiff differential equations as follows,

(a) Euler's method is the simplest to proceed with step by step because it is explicit. However, it has only a very small region where the error is not magnified. For stiff equations it has neither the desired
Figure 4.2: Stability regions for the Euler, Trapezoidal, GM and backward Euler methods.
A-stability nor the even more restrictive L-stability.

(b) The Trapezoidal method requires solving an implicit equation at each step and is therefore more difficult to use. However, it is more accurate with a higher order truncation error and it does not amplify the error for any negative $\lambda$. For the stiff equation, the trapezoidal method lacks the important L-stability condition.

(c) The backward Euler method lacks the ease of solving that the Euler method has and the higher order of accuracy that the Trapezoidal method has. However, it has the correct qualitative property for the stiff equation, i.e. the L-stability condition.

(d) Finally, the GM method lacks only the explicit form. This method has all the properties for a method that we are looking for, i.e. implicit, high order, A-stable and more importantly it is L-stable.

4.1.5 Class of equations in which the GM formula gives improved accuracy over the Trapezoidal method

As we have seen from the previous analysis, the principal error term for the GM formula is given by

$$E_G = -h^3 y_n^{'}/12 + h^3 y_n^{''}/(8y_n^{'})$$

as compared to the corresponding error term for the Trapezoidal method

$$E_T = -h^3 y_n^{''}/12.$$ 

We may now ask whether it is possible to find a class of differential equations the solution of which satisfies $|E_G| < |E_T|$, in which case we can expect a better result by using the GM method.

Substituting the related terms in the inequality, we have

$$\left| \frac{h^3 y_n^{'}}{12} - \frac{h^3 y_n^{''}}{8y_n^{'}} \right| < \left| \frac{h^3 y_n^{''}}{12} \right|$$

or,

$$-\left| \frac{h^3 y_n^{'}}{12} \right| < \left| \frac{h^3 y_n^{''}}{12} - \frac{h^3 y_n^{''}}{8y_n^{'}} \right| < \left| \frac{h^3 y_n^{''}}{12} \right|$$
If $y''_n > 0$ the inequalities become
\[
\frac{3y''_n}{h} < \frac{3y''_n}{12} - \frac{3y^n}{8y'_n} < \frac{3y''_n}{12},
\]
or
\[
\frac{-h^3y''_n}{6} < \frac{-h^3y''_n}{8y'_n} < 0.
\]
i.e., $y'_n > 0$ and $y'_n y''_n > 3(y''_n)^2/4$.

If $y''_n < 0$ we let $-y''_n = x > 0$ and the inequalities become
\[
\frac{3x}{12} < \frac{3y''_n}{12} - \frac{3y^n}{8y'_n} < \frac{3x}{12},
\]
or
\[
\frac{-3y^n}{8y'_n} < \frac{3x}{6}.
\]
The inequality on the left suggests that $y'_n < 0$. By letting $-y'_n = z > 0$, the inequality on the right becomes
\[
\frac{-3y^n}{8y'_n} < \frac{3x}{6}.
\]
i.e.,
\[
y'_n y''_n > \frac{3y^n}{4}
\]
as in the case of $y''_n > 0$.

Therefore the conditions for $|E_G| < |E_T|$ to be satisfied can be incorporated in the inequalities
\[
y'_n y''_n > 0 \quad \text{and} \quad y'_n y''_n > 3y^n^2/4 \quad \text{or more simply as,}
\]
\[
y'_n y''_n > \frac{3y^n}{4}. \quad \text{(4.1.19)}
\]
Failure to meet this condition would mean that $|E_G| \geq |E_T|$. Thus, we have established the necessary and sufficient condition for the GM formula to be more accurate than the Trapezoidal method.
In addition we have the situation \( E_G = 0 \) when \( y_n' y_n'' = 3y_n''^2 / 2 \) in which case the GM method is equivalent to a method of order 3 when used to solve the equation. Now, from the differential equation we have,

\[
y' = f(x, y) \\
y'' = f_x + ff_y \\
y''' = f_{xx} + 2ff_{xy} + f_x f_y + f^2 f_{yy} + ff_y^2.
\]

By performing this substitution into (4.1.17) we have

\[
ff_{xx} + 2f^2 f_{xy} + ff_x f_y + f^3 f_{yy} + f^2 f_y^2 > 3(f_x^2 + 2ff_x f_y + f^2 f_y^2)/4,
\]

or simplifying, we have

\[
ff_{xx} + 2f^2 f_{xy} - ff_x f_y/2 + f^3 f_{yy} + f^2 f_y^2/4 - 3f_x^2/4 > 0, \quad (4.1.20)
\]

i.e., the GM formula will be subjected to smaller error when the differential equation satisfies the condition (4.1.20). However we will not know whether this condition is satisfied throughout the region of integration until we have obtained the solution values throughout the interval. In practice we would therefore use this method if the inequality is satisfied at the initial point. Otherwise if the condition is not satisfied, the Trapezoidal method is normally preferred.

If in addition we have \( f(x, y) = f(y) \), a function of \( y \) only, then the inequality simplifies to

\[
f^3 f_{yy} + f^2 f_y^2/4 > 0
\]

or

\[
4ff_{yy} + f_y^2 > 0
\]

which is satisfied by most non-oscillatory problems and some oscillatory problems.

We have tested a few different functions \( y(x) \) to show that quite a number of differential equations do satisfy the above condition. The tests were carried out for the interval \( 0 \leq x \leq 20 \), the results of
which are tabulated in Table 4.2. The table shows how the methods compare in terms of the relative magnitudes of the truncation errors. We indicate with G where the GM formula produces smaller error. Otherwise we indicate with T for trapezoidal.

In performing the integration process we use the Picard iterations as described in Chapter 3, numerically. The Picard iterations produce approximations to the solution analytically within the range of integration as long as the integration is performed analytically. The numerical approach involving it is based on the principle of discretization in that no attempt is made to approximate the exact solution \( y(x) \) over continuous range of the independent variable. Rather, the solutions are approximated only on a set of discrete points \( x_0, x_1, x_2, \ldots \) and may be indicated by \( y_0, y_1, y_2, \ldots \) where each \( y_1 \) is now the limit of the Picard iteration within \( [x_{i-1}, x_i] \) with the integration performed, in our case, by using the Trapezoidal and GM formulae. Since the iterations could always be carried out to convergence, the error is therefore mainly due to the integration formula being used. This would enable us to compare qualitatively the accuracy of the two formulae under investigation.

In the following, we explain the technique by using one example. The others are only quoted in the results tabulated in Table 4.2.

**Example**

Consider the problem of solving

\[
y' = -y, \quad y(0) = 1
\]

whose exact solution is \( y(x) = \exp(-x) \).

By using the Picard iteration procedure, the application of the Trapezoidal and GM formulae results in the following recurrence equations
Trapezoidal: \[ y_{n+1} = y_n + \frac{h}{2} [f(x_n, y_n) + f(x_{n+1}, y_{n+1})] \]

GM: \[ y_{n+1} = y_n + h \sqrt{f(x_n, y_n) f(x_{n+1}, y_{n+1})} \]

where \( y_{n+1} \) on the right-hand side is the last calculated value of the left-hand side which initially may be taken arbitrarily. (Note that we are not yet considering the predictor-corrector technique, which will be discussed fully in Section 4.2). The computation is carried out until two successive values of \( y_{n+1} \) are acceptably close to each other.

\[ h = 0.10 \]

<table>
<thead>
<tr>
<th>( x )</th>
<th>Exact Solution</th>
<th>Numerical Solution</th>
<th>Error</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.100000E+01</td>
<td>0.904762E+00</td>
<td>-0.755099E-04</td>
<td>7</td>
</tr>
<tr>
<td>0.1</td>
<td>0.904837E+00</td>
<td>0.904875E+00</td>
<td>-0.376634E-04</td>
<td>7</td>
</tr>
<tr>
<td>0.2</td>
<td>0.818731E+00</td>
<td>0.818594E+00</td>
<td>-0.136652E-03</td>
<td>7</td>
</tr>
<tr>
<td>0.3</td>
<td>0.740818E+00</td>
<td>0.740633E+00</td>
<td>-0.185456E-03</td>
<td>7</td>
</tr>
<tr>
<td>0.4</td>
<td>0.670320E+00</td>
<td>0.670096E+00</td>
<td>-0.223743E-03</td>
<td>7</td>
</tr>
<tr>
<td>0.5</td>
<td>0.606531E+00</td>
<td>0.606278E+00</td>
<td>-0.253053E-03</td>
<td>7</td>
</tr>
<tr>
<td>0.6</td>
<td>0.548812E+00</td>
<td>0.548537E+00</td>
<td>-0.274756E-03</td>
<td>7</td>
</tr>
<tr>
<td>0.7</td>
<td>0.496585E+00</td>
<td>0.496295E+00</td>
<td>-0.299027E-03</td>
<td>7</td>
</tr>
<tr>
<td>0.8</td>
<td>0.449329E+00</td>
<td>0.449029E+00</td>
<td>-0.299905E-03</td>
<td>7</td>
</tr>
<tr>
<td>0.9</td>
<td>0.406570E+00</td>
<td>0.406264E+00</td>
<td>-0.305272E-03</td>
<td>7</td>
</tr>
<tr>
<td>1.0</td>
<td>0.367879E+00</td>
<td>0.367573E+00</td>
<td>-0.306903E-03</td>
<td>7</td>
</tr>
</tbody>
</table>

(a) Trapezoidal method

<table>
<thead>
<tr>
<th>( x )</th>
<th>Exact Solution</th>
<th>Numerical Solution</th>
<th>Error</th>
<th>No. of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.100000E+01</td>
<td>0.904875E+00</td>
<td>0.376634E-04</td>
<td>7</td>
</tr>
<tr>
<td>0.1</td>
<td>0.904837E+00</td>
<td>0.904875E+00</td>
<td>0.681470E-04</td>
<td>7</td>
</tr>
<tr>
<td>0.2</td>
<td>0.818731E+00</td>
<td>0.818594E+00</td>
<td>0.925064E-04</td>
<td>7</td>
</tr>
<tr>
<td>0.3</td>
<td>0.740818E+00</td>
<td>0.740633E+00</td>
<td>0.111594E-03</td>
<td>7</td>
</tr>
<tr>
<td>0.4</td>
<td>0.670320E+00</td>
<td>0.670432E+00</td>
<td>0.126225E-03</td>
<td>7</td>
</tr>
<tr>
<td>0.5</td>
<td>0.606531E+00</td>
<td>0.606657E+00</td>
<td>0.137062E-03</td>
<td>7</td>
</tr>
<tr>
<td>0.6</td>
<td>0.548812E+00</td>
<td>0.548949E+00</td>
<td>0.144696E-03</td>
<td>7</td>
</tr>
<tr>
<td>0.7</td>
<td>0.496385E+00</td>
<td>0.496730E+00</td>
<td>0.149632E-03</td>
<td>7</td>
</tr>
<tr>
<td>0.8</td>
<td>0.449329E+00</td>
<td>0.449479E+00</td>
<td>0.152320E-03</td>
<td>7</td>
</tr>
<tr>
<td>0.9</td>
<td>0.406570E+00</td>
<td>0.406722E+00</td>
<td>0.153138E-03</td>
<td>7</td>
</tr>
<tr>
<td>1.0</td>
<td>0.367879E+00</td>
<td>0.368033E+00</td>
<td>0.153138E-03</td>
<td>7</td>
</tr>
</tbody>
</table>

(b) GM method

Table 4.1
With stepsize \( h = 0.1 \), the results are shown in Table 4.1. The fact that the GM formula produces more accurate results in this example coincides with the fact that the inequality (4.1.19) is satisfied.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Solution</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y' = y, \ y(0) = 1 )</td>
<td>( y = \exp(x) )</td>
<td>G (all x)</td>
</tr>
<tr>
<td>( y' = -y, \ y(0) = 1 )</td>
<td>( y = \exp(-x) )</td>
<td>G (all x)</td>
</tr>
<tr>
<td>( y' = -2xy, \ y(0) = 1 )</td>
<td>( y = \exp(-x^2) )</td>
<td>G (x &gt; 1.8)</td>
</tr>
<tr>
<td>( y' = \exp(10(x-y)), \ y(0) = 0.1 )</td>
<td>( y = 0.1\ln[\exp(10x) + \exp(1) - 1] )</td>
<td>T (0 ≤ x ≤ 1.3) G (x &gt; 1.3)</td>
</tr>
<tr>
<td>( y' = (3x^2 - 2x)y, \ y(0) = 1 )</td>
<td>( y = \exp(x^3 - x^2) )</td>
<td>T (x &lt; 0.7) G (x &gt; 0.7)</td>
</tr>
<tr>
<td>( y' = - 5xy^2 + 5/x - 1/x^2, \ y(1) = 1 )</td>
<td>( y = 1/x )</td>
<td>G (all x)</td>
</tr>
<tr>
<td>( y' = 1 + y^2, \ y(0) = 1 )</td>
<td>( y = \tan(x + \pi/4) )</td>
<td>G (0 ≤ x ≤ 7/4)</td>
</tr>
</tbody>
</table>

Table 4.2: The choice between methods that produce smaller truncation errors

Computational Complexity

There are facilities in the Unix system on the Vax 750 at Loughborough University to obtain the percentage of time spent executing between an external symbol and the next, together with the number of times that the routine was called and the number of milliseconds per call. This is done by the use of the display profile data command, PROF(1).

A few problems are selected at random for the time comparison test. It is found that most time is spent for the execution of sqrt function and this function is found in the subroutine that accommodates
the GM formula. The results then show that the subroutine containing the GM formula take about 30 to 50 percent more time than the subroutine containing the Trapezoidal method depending on the problem being solved. This can be seen in Table 4.3.

<table>
<thead>
<tr>
<th>Problems</th>
<th>percentage time for evaluation of sqrt</th>
<th>milliseconds/call</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>GM</td>
</tr>
<tr>
<td>1. ( y' = -10y )</td>
<td></td>
<td>41.0</td>
</tr>
<tr>
<td>( y(0) = 1 ) ( 0 &lt; x &lt; 1 ) ( h = 0.001 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. ( y' = -2xy )</td>
<td></td>
<td>46.9</td>
</tr>
<tr>
<td>( y(0) = 1 ) ( 0 &lt; x &lt; 1 ) ( h = 0.001 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. ( y' = \exp(10(x-y)) ) ( y(0) = 0.1 ) ( 0 &lt; x &lt; 2 ) ( h = 0.01 &amp; 0.001 )</td>
<td></td>
<td>31.5</td>
</tr>
<tr>
<td>4. ( y' = -5xy^2 + 5/x - 1/x^2 )</td>
<td></td>
<td>40.3</td>
</tr>
<tr>
<td>( y(1) = 1 ) ( 1 &lt; x &lt; 2 ) ( h = 0.01 &amp; 0.001 )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.3: Comparison of the execution times for GM and Trapezoidal subroutines

Since computer time is expensive, every effort must be sought to utilise it as effectively as possible. We may argue that the extra time incurred by using the GM formula could have rather been used more effectively by the Trapezoidal formula to produce more accurate results, e.g., by reducing the steplength. However there are situations where the GM formula is more promising. For instance if the function contained squared terms initially and the square root could be avoided at source, e.g., \( \frac{dy}{dx} = \sin^4 x \cos^2 y \), etc.
Example

We solve the initial value problem

\[ y' = (\csc^2 x)y^2, \quad y(\pi/4) = 1 \]

whose exact solution is \( y = \tan x \).

In this example the Trapezoidal formula would effectively become

\[ y_{n+1} = y_n + \frac{h}{2} [f(x_n, y_n) + f(x_{n+1}, y_{n+1})] \]

where \( f(x, y) = (\csc^2 x)y^2 \),

while the GM formula would simply become

\[ y_{n+1} = y_n + hg(x_n, y_n)g(x_{n+1}, y_{n+1}) \]

where \( g(x, y) = (\csc x)y \).

Both algorithms are run for \( \pi/4 \leq x \leq 3\pi/8 \) by using steplength \( h=0.01 \).

The results are printed every ten steps as shown in Table 4.4.

Comparisons on the accuracy and the computing times show that the GM formula produces, in fact, in this example, a more accurate result with less time.

<table>
<thead>
<tr>
<th>x</th>
<th>Exact Solution</th>
<th>Error in Trapezoidal</th>
<th>Error in GM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7854</td>
<td>0.10000e+01</td>
<td>0.00000e-00</td>
<td>0.00000e-00</td>
</tr>
<tr>
<td>0.8854</td>
<td>0.12230e+01</td>
<td>0.21338e-04</td>
<td>0.45300e-05</td>
</tr>
<tr>
<td>0.9854</td>
<td>0.15085e+01</td>
<td>0.71049e-04</td>
<td>0.12875e-04</td>
</tr>
<tr>
<td>1.0854</td>
<td>0.18958e+01</td>
<td>0.19014e-03</td>
<td>0.28610e-04</td>
</tr>
<tr>
<td>1.1854</td>
<td>0.24650e+01</td>
<td>0.49872e-03</td>
<td>0.61982e-04</td>
</tr>
</tbody>
</table>

ms/call for Trapezoidal subroutine = 2.14

ms/call for GM subroutine = 1.71

Table 4.4
4.1.6 Application of the GM formula for a system of o.d.e.'s

Let us now consider a system of \( n \) first order (in general nonlinear) ordinary differential equations

\[
\begin{align*}
y_1' &= f_1(x, y_1, y_2, \ldots, y_n) \\
y_2' &= f_2(x, y_1, y_2, \ldots, y_n) \\
\vdots \\
y_n' &= f_n(x, y_1, y_2, \ldots, y_n)
\end{align*}
\]

Applying the GM formula produces the difference equations:

\[
\begin{align*}
y_i^{(m+1)} &= y_i^{(m)} + h \sqrt{f_i(x, y_1^{(m)}, y_2^{(m)}, \ldots, y_n^{(m)})} \\
&\quad \frac{\sqrt{f_i(x, y_1^{(m+1)}, y_2^{(m+1)}, \ldots, y_n^{(m+1)})}}{\sqrt{f_i(x, y_1^{(m+1)}, y_2^{(m+1)}, \ldots, y_n^{(m+1)})}} \\
\end{align*}
\]

(4.1.21)

where \( y_i^{(m)} \) is numerical solution at \( x = x_m \).

Equation (4.1.21) are generally nonlinear, not only because the differential equations are nonlinear, but also because the method used is nonlinear. The nonlinear algebraic equations (4.1.19) are normally solved by using a Picard's iteration procedure with initial guess \( (y_1^{(m+1)}, y_2^{(m+1)}, \ldots, y_n^{(m+1)})_0 \) computed in advance at each integration step by any explicit method (e.g. Euler's method). For a large system this procedure would probably be less effective and use much time since the convergence is only linear. Furthermore if the differential system is stiff this process might not be convergent. The sufficient conditions of convergence for the Picard's iteration method are given
as follows,

(1) \( F \), i.e. the right-hand side functions and their partial derivatives are continuous in a neighbourhood \( R \) of the root.

(2) \( |F| + |F| \leq K \) for all points in \( R \) and some \( K < 1 \).

(3) The initial approximation \( (x_0, y_0) \) is chosen in \( R \).

As we may expect, condition (2) is too difficult for a stiff system to satisfy. An answer to this difficulty is to use the Newton-Raphson procedure which is given by

\[
y^{(n+1)} = y^{(n)} - \frac{f(y^{(n)})}{f'(y^{(n)})}, \quad f'(y^{(n)}) \neq 0.
\]

The algebraic system (4.1.19) is written in the form

\[ g_1(y_{(m+1)}, y_{(m+1)}, \ldots, y_{(m+1)}) = 0, \quad i = 1, 2, \ldots, n \]

where the right-hand side is zero.

For example, for the first equation of (4.1.19) we have

\[ g_1(y_{(m+1)}, y_{(m+1)}, \ldots, y_{(m+1)}) = y_{(m)} + h\sqrt{f_1}\sqrt{f_1(x_{(m+1)}, y_{(m+1)}, \ldots, y_{(m+1)})} - y_{(m+1)}. \]

The problem is therefore to solve the system

\[ g_i = 0, \quad i = 1, 2, \ldots, n \]

for \( y_{(m+1)}, y_{(m+1)}, \ldots, y_{(m+1)} \).

To perform the Newton-Raphson iteration for a system, we have to supply the Jacobian of the system. In this case the Jacobian is the \( (n \times n) \)-matrix

\[
J(g) = \frac{\partial(g_1, g_2, \ldots, g_n)}{\partial(y_1, y_2, \ldots, y_n)} = \frac{\partial g_i}{\partial y_j}; \quad i, j = 1, 2, \ldots, n
\]

i.e., a matrix with the \((i, j)\)-th entry equal to \( \partial g_i / \partial y_j \).
Now,
\[ g_i(y) = y^{(m)}_1 + h \frac{f_1^{(m)}}{\sqrt{f_1^{(m+1)}}} - y^{(m+1)}_1 \]
where superscript \( m \) indicates that the function is evaluated at \((x_m, y^{(m)}_1, y^{(m)}_2, \ldots, y^{(m)}_n)\).

Differentiating w.r.t. \( y_j \) gives
\[ \frac{\partial g_i}{\partial y_j} = \frac{h f_1^{(m)}}{2 \sqrt{f_1^{(m+1)}}} \frac{\partial f_1}{\partial y_j}^{(m+1)} - \delta_{ij}. \]

Therefore, for a system, the Jacobian of the algebraic system is expressed in the Jacobian of the differential system by the relation (in matrix notation)
\[ \mathbf{J}(g) = k \mathbf{J}(y) \bigg|_{(m+1)} - \mathbf{I} \]
where
\[ k = \begin{bmatrix} h \frac{f_1^{(m)}}{\sqrt{f_1^{(m+1)}}} \\ \frac{h}{2} \frac{f_2^{(m)}}{\sqrt{f_2^{(m+1)}}} \\ \vdots \\ \frac{h}{2} \frac{f_n^{(m)}}{\sqrt{f_n^{(m+1)}}} \end{bmatrix} \]
and the result of the matrix operation \( k \odot \mathbf{J}(y) \) is a matrix \( \mathbf{B} \) defined by \( B[i,j] = k_{1} a_{ij} \).

Having found \( J \), we next solve the linear system \( \mathbf{Jd} = -g \) for \( d \). The improved solution is \( y^{(m+1)} + d \). (Note that the initial \( y^{(m+1)} \) is obtained by Euler's method). The process is repeated until convergent.

Newton's method converges quadratically under the following sufficient conditions (but not necessary):
g and all its derivatives through to second order are continuous and bounded in a region R containing the root.

(2) The Jacobian J(g) does not vanish in R.

(3) The initial approximation \( y_0 \) is chosen sufficiently close to the root.

These conditions are less strict if compared to the previous conditions for the Picard's iteration procedure.

The subroutine for solving the differential equation will basically consists of the following stages.

(1) Read values (supplied or computed) at present step.

(2) Guess the solution values at the next integration step by an explicit method.

(3) Compute J and g by using the values at present and the guessed next integration step.

(4) Solve \( J \Delta = -\bar{g} \) for \( \Delta \) by using any solution method, e.g., Gaussian elimination.

(5) Update the solution values at next integration step by adding \( \Delta \) to the guessed values.

(6) Return to step 3 with updated values in place of guessed values, until convergence.

(7) Return to the calling program.

In general practice, the Newton-Raphson procedure gives a satisfactory means of solving the nonlinear difference equations resulting from the application of the GM formula since the convergence is quadratic. For further enhanced convergence, the generalised Newton's formula may be used. The method is given by

\[
y^{(n+1)} = y^{(n)} - \frac{f(y^{(n)})}{f'(y^{(n)})},
\]

\[f'(y^{(n)}) \neq 0\] 

(4.1.23)
where \( w \) is the over-relaxation factor \( \neq 1 \).

### 4.1.7 Quadrature Methods: GM-Romberg Scheme

We have so far seen how the geometric mean approach has been used as a substitute to the more usually applied arithmetic mean in the generalised Euler's method for solving initial value problems. By following closely this technique we may now develop an integration formula which is based on the GM formula. In particular, from the Newton Cotes formula for integration we can establish the following non-linear formula obtained for quadrature

\[
\int_{x_0}^{x_1} f(x)\,dx = h \sqrt{f(x_0)f(x_1)}
\]  

(4.1.24)

where we have replaced the arithmetic mean in the Trapezoidal rule by a Geometric Mean.

We will now apply this formula to a Romberg integration in which the composite non-linear formula (4.1.24) is applied to give a preliminary approximation and then the Richardson extrapolation process is applied to obtain improvements of the approximation.

It is customary to use a basic stepsize \( h = b - a \) and a sequence of step sizes with ratio \( r = \frac{1}{2} \) when using Romberg integration to approximate

\[
I(f) = \int_{a}^{b} f(x)\,dx
\]

Thus we define

\[
G_0 = \frac{b-a}{2^m} [\sqrt{f_0 f_1} + \sqrt{f_1 f_2} + \ldots + \sqrt{f_{s-1} f_s}]
\]

(4.1.25)

\( m = 0, 1, 2, \ldots \)

where \( f_i = f(x_i) \), \( x_i = a + \frac{i(b-a)}{2^m} \),
and $s = 2^m$. This choice of $h$ and $r$ means that we can use the function values already available in the computation of $G_0^{(m+1)}$ and easily save half the work in evaluating $f(x)$ when forming $G_0^{(m+1)}$.

From the nature of the asymptotic expansion of the formula (4.1.24) we have seen that the error term initially contains only even powers of $h$; so the appropriate form of Richardson extrapolation to be used is given by

$$G_i^m = \frac{G_{i-1}^{(m+1)} - (\frac{1}{4})^{2i} G_{i-1}^{(m)}}{1 - (\frac{1}{4})^{2i}}, \quad i = 1, 2, \ldots, \quad m = 0, 1, \ldots$$

This equation can be simplified to the more convenient form

$$G_i = G_{i-1}^0 + \frac{G_{i-1}^{(m+1)} - G_{i-1}^{(m)}}{4^{-i} - 1}, \quad i = 1, 2, \ldots, \quad m = 0, 1, \ldots$$

(4.1.26)

Using (4.1.27) we have an easily programmed algorithm to construct Table 4.5, one column at a time, from left to right.

Table 4.5
Alternatively we may use another algorithm to construct Table 4.5, one row at a time, moving downwards. This algorithm enables us to compute only necessary entries before the process is terminated by an accuracy stopping criterion. The algorithm is given thus,

\[
g_0 = \frac{b-a}{2^m} \left( \sqrt{f_0 f_1} + \sqrt{f_1 f_2} + \ldots + \sqrt{f_{s-1} f_s} \right)
\]

\[
g_i = g_{i-1} + \frac{g_{i-1} - g_{i-2}}{4^i - 1}, \quad i=0,1,\ldots,m
\]

where \( m \) is the outer loop and \( i \) is the inner loop.

**Numerical Example**

Consider the problem of calculating

\[
\text{Si}(1) = \int_0^1 \frac{\sin t}{t} \, dt
\]

The exact value of \( \text{Si}(1) \) to eight places is \( \text{Si}(1) = 0.94608307 \). The result of the GM-Romberg integration is given in Table 4.6. The result obtained by using the Trapezoidal formula is also given for comparison. Both formulae are accurate to eight places after making use of four extrapolation points.

From this result we may conclude that the GM formula is equally competitive as the Trapezoidal formula when used in the Romberg integration scheme.
### (a) GM-Romberg

<table>
<thead>
<tr>
<th>$G_0^j$</th>
<th>$G_1^j$</th>
<th>$G_2^j$</th>
<th>$G_3^j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.91731728D+00</td>
<td>0.93872757D+00</td>
<td>0.94423392D+00</td>
<td>0.94562014D+00</td>
</tr>
<tr>
<td>0.94586433D+00</td>
<td>0.94606938D+00</td>
<td>0.94608305D+00</td>
<td>0.94608307D+00</td>
</tr>
</tbody>
</table>

### (b) Trapezoidal-Romberg

<table>
<thead>
<tr>
<th>$G_0^j$</th>
<th>$G_1^j$</th>
<th>$G_2^j$</th>
<th>$G_3^j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.92073549D+00</td>
<td>0.93979329D+00</td>
<td>0.94451352D+00</td>
<td>0.94569086D+00</td>
</tr>
<tr>
<td>0.94614588D+00</td>
<td>0.94608693D+00</td>
<td>0.94608301D+00</td>
<td>0.94608307D+00</td>
</tr>
<tr>
<td>0.94608331D+00</td>
<td>0.94608307D+00</td>
<td>0.94608307D+00</td>
<td>0.94608307D+00</td>
</tr>
</tbody>
</table>

Table 4.6: Romberg Integration by using GM and Trapezoidal formulae
4.2 APPLICATION OF THE GM METHOD IN THE PREDICTOR CORRECTOR MANNER

When \( y' = f(x,y) \) is nonlinear, implicit methods require the solution of a nonlinear equation. In the application of the GM method the resulting difference equation is always nonlinear regardless of the differential equation due to the presence of the square root in the formula. The solution of the nonlinear equation is usually done iteratively. Iterative equation solvers such as the Picard's and Newton's methods which we have discussed in the preceding section find successively more accurate solutions and usually a form of linearization is used. It does not make sense, however, to solve the difference equation, which is an approximation to the differential equation, to an accuracy greater than that of the approximation itself. Therefore solution by implicit methods are computed very accurately only when the equations are simple enough to solve directly or when stability is crucial.

We are thus led to consider a compromise between the simplicity of the explicit method (but lack of good stability) and the stability of the implicit process (with its more difficult computation). The predictor-corrector approach can be thought of as an approximation to solving the implicit equation. Basically it consists of the application of an explicit method to estimate (predict) the new value of the independent variable, and the subsequent application (or applications) of an implicit method to improve (correct) it. We will restrict our analysis only to the Trapezoidal and the GM formulae as the corrector formulae and examine the behaviour of the errors with successive applications of the corrector formulae.

Since there are a great many explicit methods which can be used as predictor, we will restrict ourself to three explicit formulae, namely the Euler formula, Newton Cotes open formula and the GM equivalent of the Newton Cotes open formula.
4.2.1 Euler's predictor formula

We shall now begin with the analysis of using Euler's method as a predictor formula followed by successive application of the GM formula in the P(EC)\^mE mode (c.f. notation used by Lambert [1973]). Thus, the step-by-step application of the formulae can be written as (P = Predict, C = Correct, E = Function evaluation),

\[
P: \quad y_{n+1}^{[0]} = y_n + hf_n^{[m]} \quad (4.2.1-i)\]

\[
EC^{\text{m}}: \quad y_{n+1}^{[s]} = f(x_{n+1}, y_{n+1}^{[s]}) \quad s=0,1, \ldots, m-1 \quad (4.2.1-ii)\]

\[
E: \quad y_{n+1}^{[s+1]} = y_n + h f_n^{[m]} f_{n+1}^{[s]} \quad (4.2.1-iii)\]

It is already known that the local truncation error for the Euler and the GM methods are respectively given by

\[
E_E = \frac{1}{2} h^2 y_n^{'''} + O(h^3) \]

\[
E_G = \left( - \frac{1}{12} y_n^{'''} + \frac{1}{6} y_n^{''} \right) h^3 + O(h^4).\]

Therefore from the predictor formula we produce a result which differs from the exact solution approximately by $E_E$, i.e.,

\[
y(x_{n+1}) - y_{n+1}^{[0]} = E_E \quad (4.2.2)\]

The subsequent application of the corrector formula will obey the following relations,

\[
y(x_{n+1}) - y(x_n) = h \sqrt{f_n(x_n, y_n)} f_n(x_{n+1}, y_{n+1}) + E_G \quad (4.2.3-i)\]

\[
y_{n+1}^{[s+1]} - y_{n+1}^{[m]} = h \sqrt{f_n^{[m]} f_{n+1}^{[s]}} \quad s=0,1, \ldots, m-1 \quad (4.2.3-ii)\]
On subtracting (4.2.3-ii) from (4.2.3-i) and by assuming that $y_n^{[m]}$ is free from error, we obtain

$$
y(x_{n+1}) - y_n^{[s+1]} = h \sqrt{f(x_n,y(x_n))} f(x_{n+1}, y(x_{n+1})) - h \sqrt{f(x_{n+1}, y(x_{n+1}))} - \sqrt{s} + E_G$$

$$= h \sqrt{f(x_n,y(x_n))} [\sqrt{f(x_{n+1}, y(x_{n+1}))} - \sqrt{s}] + E_G. \quad (4.2.4)$$

Letting $g = \sqrt{f}$, we have, from the Mean Value Theorem,

$$g(x_{n+1}, y(x_{n+1}))-g(x_n,y(x_n)) = \frac{\partial g}{\partial y}(x_n, z)[y(x_{n+1}) - y_n^{[s]}]$$

$$= \frac{1}{2} \sqrt{f} \frac{\partial f}{\partial y}(x_{n+1}, z)[y(x_{n+1}) - y_n^{[s]}]$$

for some $z$ within the interval whose end-points are $y_n^{[s]}$ and $y(x_{n+1})$.

By substituting this into (4.2.4) we obtain

$$y(x_{n+1}) - y_n^{[s+1]} = h \sqrt{f(x_n, y(x_n))} \frac{1}{2} \frac{\partial f}{\partial y}(x_{n+1}, z)[y(x_{n+1}) - y_n^{[s]}]$$

$$- y_n^{[s]} + E_G, \quad s = 0, 1, \ldots, m-1 \quad (4.2.5)$$

By putting successive values of $s$ we have,

$s = 0$,

$$y(x_{n+1}) - y_n^{[1]} = \frac{1}{2} h \sqrt{f} \frac{\partial f}{\partial y} [y(x_{n+1}) - y_n^{[0]}] + T_G$$

$$= \frac{1}{2} h \sqrt{f} \frac{\partial f}{\partial y} E + E_G,$$

$s = 1$,

$$y(x_{n+1}) - y_n^{[2]} = \frac{1}{2} h \sqrt{f} \frac{\partial f}{\partial y} [y(x_{n+1}) - y_n^{[1]}] + E_G$$

$$= \frac{1}{2} h \sqrt{f} \frac{\partial f}{\partial y} \left[ \frac{1}{2} \sqrt{f} \frac{\partial f}{\partial y} E + E_G \right] + E_G,$$

etc.
Since \( o(E_G) = o(hE_E) \), this predictor-corrector combination is of the same order as the corrector formula, but their local truncation errors are not identical when used with \( m=1 \). For \( m>1 \), the local truncation error of method (3.2.1) is identical to that of the GM formula. Therefore, with two applications of the corrector formula, we can expect to obtain a result that purely represents the numerical solution of the problem by using the GM formula, as though we have solved the nonlinear difference equation exactly.

For comparison we similarly examine the Euler-Trapezoidal combination formulae which is better known as Heun's second order method, i.e.

\[
P: \quad y(x_{n+1}) - y[0] = E_E
\]

(\text{EC})\text{M} : \[y(x_{n+1}) - y_n = \frac{h}{2}[f(x_n, y(x_n)) + f(x_{n+1}, y(x_{n+1}))] + E_T \]

\[y_{n+1} - y_n = \frac{h}{2}(f_n^m + f_{n+1}^m), \quad s=0,1,\ldots,m-1 \quad (4.2.7-i)\]

where \( E_T \) is the local truncation error of the Trapezoidal method.

By subtracting (4.2.7-ii) from (4.2.7-i) we obtain

\[
y(x_{n+1}) - y_{n+1}^{[s+1]} = \frac{h}{2}[f(x_n, y(x_n)) + f(x_{n+1}, y(x_{n+1}))] + E_T - \frac{h}{2}[f_n^m + f_{n+1}^m]
\]

\[= \frac{h}{2}[f(x_{n+1}, y(x_{n+1})) - f_{n+1}^m] + E_T
\]

\[= \frac{h\delta f}{\delta y}(x_{n+1}, z)[y(x_{n+1}) - y_{n+1}^s] + E_T
\]

\[s=0,1,\ldots,m-1.\]

Putting \( s=0 \), we have

\[
y(x_{n+1}) - y_{n+1}^{[1]} = \frac{h\delta f}{\delta y}(y(x_{n+1}) - y[0]) + E_T
\]

\[= \frac{h\delta f}{\delta y} E_E + E_T.
\]
Again we expect in practice to obtain a sufficiently close result to any other accurate method of solving the nonlinear difference equation by using two applications of the corrector, since the error is that of the corrector alone.

Thus, for the purpose of comparison, the accuracy of the methods will only depend on the LTE of the corrector, and the choice between the use of GM or trapezoidal formulae will again be based on the satisfaction of inequality (4.1.19) for the problem under consideration.

The above analysis was meant to show the stages in which the accuracy of the implicit methods may be achieved after a number of applications of the corrector formulae. In the following let us consider specifically the error terms involved in each stage by using the two formulae.

The application of the predictor formula would produce an approximation of the form \( y_{n+1}^{[0]} = y_n + h y'_n \). This is followed by the first application of the corrector which produces,

\[
y_{n+1}^{[1]} = y_n + h \sqrt{y_n^{[0]}} y'_n
\]

\[= y_n + h \sqrt{y_n^{[0]}} (y'_n + hy''_n)\]

\[= y_n + hy'_n \sqrt{1 + \frac{hy''_n}{y_n}}\]

\[= y_n + hy'_n (1 + \frac{hy''_n}{y_n} - \frac{1}{8} \frac{h^2 y''_n^2}{y_n^2} + \ldots )\]
where we have used the expansion

\[ \sqrt{1+x} = 1 + \frac{1}{2}x - \frac{1}{8}x^2 + \frac{1}{16}x^3 + \ldots \]  

(4.2.8)

with \( x = \frac{h^2}{y_n} \), and assuming \( y_n' \neq 0 \) throughout the analysis.

Therefore \( y_n^{[1]} \) is of the second order with local truncation error given by

\[ g_1 = \frac{1}{6} h^3 y_n'' + \frac{1}{8} h^3 \frac{y_n''}{y_n} \]

\[ = h^3 \left( \frac{y_n''}{6} + \frac{y_n''}{8y_n'} \right). \]  

(4.2.9)

The second application of the corrector formula would result in

\[ y_n^{[2]} = y_n + h \sqrt{y_n'} \left( y_n^{[1]} \right) \]

\[ = y_n + h \sqrt{y_n'} \left( y_n' + h y_n'' + \frac{1}{2} h^2 y_n''' - \ldots \right) \]

\[ = y_n + h y_n' \sqrt{1 + \frac{h^2}{y_n'} + \frac{h^2}{2} \left( \frac{y_n''}{y_n'} \right) - \ldots} \]

\[ = y_n + h y_n' + \frac{1}{2} h^2 y_n'' + \frac{1}{4} h^3 y_n''' - \frac{1}{8} h^3 \left( \frac{y_n''^2}{y_n'} \right) + \ldots \]

after making use of (4.2.8) with \( x = \frac{h^2}{y_n} + \frac{h^2}{2} \left[ \frac{y_n''}{y_n'} \right] - \ldots \)

Thus \( y_n^{[2]} \) is of second order with local truncation error given by

\[ g_2 = \frac{1}{6} h^3 y_n'' - \frac{1}{4} h^3 y_n''' + \frac{1}{8} h^3 \left( \frac{y_n''^2}{y_n'} \right) \]

\[ = h^3 \left( - \frac{y_n''}{12} + \frac{y_n''}{8y_n'} \right) \]  

(4.2.10)

Since only terms up to the second order from \( y_n^{[k]} \) will effectively be
used for the evaluation of $y_n^{[k+1]}$, we see that there will be no change in the principal error terms if we carry on the application of the corrector formula. Only higher order terms may change but this has no effect on the relative accuracy with the corrector formula.

The analysis for the trapezoidal formula may be carried out similarly. The corresponding errors for $y_n^{[1]}$ and $y_n^{[2]}$ by using the Trapezoidal method are given by $T_1$ and $T_2$ where

$$T_1 = \frac{1}{6} h^3 y_n^{''}$$

and

$$T_2 = -\frac{1}{12} h^3 y_n^{'''}.$$  

Thus for the GM predictor-corrector formula to be more accurate than Heun's 2nd order formula we must have $|G_1| < |T_1|$ for a single application of the corrector, or $|G_2| < |T_2|$ for two or more applications of the corrector.

**Stability**

Consider the step-by-step application of Heun's 2nd order formula

$$y_n^{[0]} = y_n + hf_n$$  

(4.2.13-i)

$$y_n^{[1]} = y_n + hf_n^{[0]} + \frac{h}{2} (f_n + f_{n+1}^{[0]})$$  

(4.2.13-ii)

$$y_n^{[k]} = y_n + hf_n^{[k-1]} + \frac{h}{2} (f_n + f_{n+1}^{[k-1]})$$  

(4.2.13-iii)

$k = 2, 3, \ldots$

to the test equation $y' = \lambda y$. This will effectively produce, in stages,

$$y_n^{[0]} = y_n + h\lambda y_n$$

$$= (1 + h\lambda) y_n$$

$$y_n^{[1]} = y_n + \frac{h}{2} [\lambda y_n + \lambda(1+h\lambda) y_n]$$
\[ y_{n+1}^{[2]} = y_n + \frac{h}{2}(\lambda y_n + \lambda(1+h\lambda+\frac{1}{2}h^2\lambda^2)y_n) 
= (1+h\lambda+\frac{1}{4}h^2\lambda^2)y_n \]

\[ y_{n+1}^{[3]} = y_n + \frac{h}{2} (\lambda y_n + \lambda(1+h\lambda+\frac{1}{2}h^2\lambda^2+\frac{1}{4}h^3\lambda^3)y_n) 
= (1+h\lambda+\frac{1}{6}h^2\lambda^2+\frac{1}{4}h^3\lambda^3+\frac{1}{8}h^4\lambda^4)y_n \]

\[ \vdots \]

etc.

A similar application of the test equation to the GM predictor-corrector formula

\[ y_{n+1}^{[0]} = y_n + hf_n \quad (4.2.14-i) \]

\[ y_{n+1}^{[1]} = y_n + h\sqrt{f_n \cdot f_0} \quad (4.2.14-ii) \]

\[ y_{n+1}^{[k+1]} = y_n + h\sqrt{f_n \cdot f_k} \quad (4.2.14-iii) \]

\( k=2,3, \ldots \)

produces

\[ y_{n+1}^{[0]} = (1+h\lambda)y_n \]
\[ y_{n+1}^{[1]} = (1+h\lambda \sqrt{1+h\lambda})y_n \]
\[ y_{n+1}^{[2]} = (1+h\lambda \sqrt{1+h\lambda \sqrt{1+h\lambda}})y_n \]
\[ y_{n+1}^{[3]} = (1+h\lambda \sqrt{1+h\lambda \sqrt{1+h\lambda \sqrt{1+h\lambda}}})y_n \]

\[ \vdots \]

etc.

These recurrence relations may be written as

\[ y_{n+1}^{[\mu]} = Q_\mu(h\lambda)y_n \quad (4.2.15) \]
Figure 4.3: Stability regions for the Heun 2nd order predictor-corrector formula.

Figure 4.4: Stability regions for the GM predictor-corrector formula.
where \( Q_\mu(h\lambda) \) is the \( \mu \)-th magnification factor of the correction process.

For absolute stability we require that \( |Q_\mu(h\lambda)| < 1 \). The regions of absolute stability for the Heun 2nd order formula is shown in Figure 4.3, while that of the GM predictor-corrector is shown in Figure 4.4. In each case the \( \mu \)-th correction is stable inside the closed curve indicated.

As can be seen from these two figures, the Heun formula has a bigger region of stability, probably due to the linearity and simplicity of the process. However, in practice, if the step size is properly chosen and especially for solving non-stiff problems, the GM predictor-corrector formula could become equally competitive to the Heun formula.

4.2.2 Newton Cotes Formula

The Newton Cotes open and closed formulae

\[
y_{n+1} = y_{n-2} + \frac{3h}{2} (y_{n-1}', + y_n'), \quad \text{LTE} = \frac{3h^3 y_n''}{4} \quad (4.2.16)
\]

\[
y_{n+1} = y_n + \frac{h}{2} (y_n' + y_{n+1}'), \quad \text{LTE} = -\frac{h^3 y_n''}{12} \quad (4.2.17)
\]

are often used as a predictor-corrector pair. In view of the resemblance we have already established in the earlier section, it seems possible to use their geometric mean equivalents also as a predictor-corrector pair, i.e.

\[
y_{n+1} = y_{n-2} + 3h \sqrt{y_{n-1}' y_n'}, \quad \text{LTE} = \frac{3}{4} h^3 y_n''' + \frac{3}{8} h^3 \frac{y_n''}{y_n'} \quad (4.2.18)
\]

\[
y_{n+1} = y_n + h \sqrt{y_n' y_{n+1}'}, \quad \text{LTE} = \frac{1}{12} h^3 y_n''' + \frac{1}{8} h^3 \frac{y_n''}{y_n'} \quad (4.2.19)
\]
The LTE in (4.2.18) was obtained as follows:-

Expanding the relevant terms in the Taylor series we have

\[ y_{n-1} = y_n - hy'_{n-2} + \frac{h^2}{2} y''_{n-2} - \frac{h^3}{6} y'''_{n-2} + \ldots \]  
(4.2.20-i)

\[ y'_{n-1} = y'_{n-2} - hy''_{n-2} + \frac{h^2}{2} y'''_{n-2} - \frac{h^3}{6} y'^{iv}_{n-2} + \ldots \]  
(4.2.20-ii)

\[ y_{n-2} = y_n - 2hy''_{n-2} + 2h^2 y'''_{n-2} - \frac{4}{3} h^3 y''''_{n-2} + \ldots \]  
(4.2.20-iii)

It follows that

\[ y_n y'_{n-1} = y_n^2 \left[ 1 + \left( \frac{-hy''_{n-2}}{y_{n-2}'} + \frac{h^2}{2} \frac{y'''_{n-2}}{y_{n-2}'} - \frac{h^3}{6} \frac{y'^{iv}_{n-2}}{y_{n-2}'} + \ldots \right) \right] \]

i.e.

\[ \sqrt{y_n y'_{n-1}} = y_n \left[ 1 + \frac{1}{2} \left\{ \frac{-hy''_{n-2}}{y_{n-2}'} + \frac{h^2}{2} \frac{y'''_{n-2}}{y_{n-2}'} - \frac{h^3}{6} \frac{y'^{iv}_{n-2}}{y_{n-2}'} + \ldots \right\} \right. \]

\[ \left. - \frac{1}{8} \left\{ h^2 \left( \frac{y''_{n-2}}{y_{n-2}'} \right)^2 - \frac{3}{4} \frac{y''''_{n-2}}{y_{n-2}'} + \ldots \right\} \right. \]

\[ + \left. \frac{1}{16} \left\{ -h^3 (y''_{n-2}/y_{n-2}')^3 + \ldots \right\} + \ldots \right] \]

\[ = y_{n-1} + h[\gamma_{n-1}^2] + h^2 \left[ \frac{1}{4} y''_{n-2} - \frac{1}{8} \frac{y''''_{n-2}}{y_{n-2}'} \right] + \ldots \]  
(4.2.21)

Substituting (4.2.20-iii) and (4.2.21) into (4.2.18) we obtain

\[ y_{n+1} = y_{n-2} + 3h \sqrt{y_{n-1} y'_{n-1}} \]

\[ = [y_n - 2hy'_{n-1} + 2h^2 y''_{n-1} - \frac{4}{3} h^3 y'''_{n-1} + \ldots] \]

\[ + 3h [y_{n-1} - \frac{hy''_{n-1}}{y_{n-1}'}] + h^2 \left( \frac{1}{4} y''_{n-1} - \frac{1}{8} \frac{y''''_{n-1}}{y_{n-1}'} \right) + \ldots \]
Equation (4.2.22) agrees with the Taylor series expansion of \( y(x_{n+1}) \) to the second order terms. The local truncation error is given by

\[
\text{LTE} = \frac{1}{6} y_n'''' - h^3\left[-\frac{7}{12} y_n''' - \frac{3y_n''^2}{8y_n''} \right] + O(h^4)
\]

which establishes the result in (4.2.18).

In the following we will consider the various combination of these formulae and compare their respective truncation errors.

(a) Formulae (4.2.16) and (4.2.17)

\[
y_{n+1}^{[0]} = y_{n-2} + \frac{3h}{2} [y_{n-1}' + y_n'] \\
y_{n+1}^{[1]} = y_n + \frac{h}{2} [y_n' + y_{n+1}^{[0]}]
\]

Using (4.2.20) we have

\[
y_{n+1}^{[0]} = y_n + hy_n' + \frac{h^2}{2} y_n'' - \frac{7}{12} h^3 y_n''' + ...
\]

so that

\[
y_{n+1}^{[1]} = y_n + \frac{h}{2} [y_n' + y_n' + hy_n'' + \frac{h^2}{2} y_n'' - ...]
\]

\[
= y_n + hy_n' + \frac{h^2}{2} y_n'' + \frac{h^2}{4} y_n''' + ...
\]

i.e. \( \text{LTE} = -\frac{1}{12} h^3 y_n'''' \) \hspace{1cm} (4.2.23)

(b) Formulae (4.2.18) and (4.2.19)

\[
y_{n+1}^{[0]} = y_{n-2} + 3h \sqrt{y_n'y_{n-1}} \\
y_{n+1}^{[1]} = y_n + h \sqrt{y_n'y_{n+1}^{[0]}}
\]

From (4.2.22) we have
\[ y_{n+1}^{[0]} = y_n + h y_n' + \frac{h^2}{2} y_n'' + h^3 \{ ... \} + ... . \]

Substituting into the corrector formula we obtain

\[ y_{n+1} = y_n + h \sqrt{y_n'[y_n + h y_n' + \frac{h^2}{2} y_n'' + ...]}. \]

\[ = y_n + h y_n' \sqrt{1 + \frac{h y_n''}{y_n' n} + \frac{h^2 y_n''}{2 y_n'} + ...} \]

\[ = y_n + h y_n' [1 + \frac{1}{8} \left( \frac{h y_n''}{y_n'} + \frac{h^2 y_n''}{2 y_n'} + ... \right) + \frac{1}{8} \frac{h^2 y_n''^2}{y_n'^2} + ...] \]

\[ = y_n + h y_n' + \frac{1}{8} h^2 y_n'' + h^3 \left( \frac{1}{4} y_n'' - \frac{1}{8} \frac{y_n'^2}{y_n''} \right) + ... \]

i.e. \[ \text{LTE} = h^3 \left[ -\frac{1}{12} y_n'' + \frac{y_n'^2}{8 y_n''} \right] \] (4.2.24)

(c) Formulae (4.2.16) and (4.2.19)

\[ y_{n+1}^{[0]} = y_{n-2} + \frac{3 h}{2} [y_n' + y_n''] \quad \text{Predictor} \]

\[ y_{n+1}^{[1]} = y_n + h \sqrt{y_n'[y_n^{[0]} + y_n''] \quad \text{Corrector} \]

Expanding the predictor formula, we have

\[ y_{n+1}^{[0]} = y_n - 2 h y_n' + \frac{4}{2} h^2 y_n'' - \frac{8}{6} h^3 y_n''' + ... \]

\[ + \frac{3 h}{2} [y_n' + y_n' - h y_n'' + \frac{1}{8} h^2 y_n'' - \frac{1}{6} h^3 y_n''' + ...] \]

so that

\[ y_{n+1}^{[1]} = y_n + h \sqrt{y_n''^2 \left( 1 + \frac{h y_n''}{y_n'} + \frac{h^2 y_n''}{2 y_n'} - ... \right)} \]

\[ = y_n + h y_n' + \frac{h^2}{2} y_n'' + h^3 \left( \frac{1}{4} y_n'' - \frac{y_n'^2}{8 y_n''} \right) + ... \]

i.e. \[ \text{LTE} = h^3 \left[ -\frac{1}{12} y_n'' + \frac{y_n'^2}{8 y_n''} \right] \] (4.2.25)
(d) Formula (4.2.18) and (4.2.17)

\[ y_{n+1}^{[0]} = y_{n-2} + 3h \sqrt{y_{n-1} y_n} \quad \text{Predictor} \]

\[ y_{n+1}^{[1]} = y_n + \frac{h}{2} (y_n' + y_{n+1}^{[0]}) \quad \text{Corrector} \]

Expanding, we have

\[ y_{n+1}^{[0]} = y_n + h y_n' + \frac{h^2}{2} y_n'' + h^3 \left[ -\frac{7}{12} y_n^{(3)} - \frac{3}{8} y_n'' \right] + \ldots \]

so that by substituting it into the corrector formula we obtain

\[ y_{n+1}^{[1]} = y_n + \frac{h}{2} [h y_n' + \frac{h^2}{2} y_n'' + \ldots + y_n'] \]

\[ = y_n + h y_n' + \frac{h^2}{2} y_n'' + \frac{1}{4} h^3 y_n'''' + \ldots \]

i.e.

\[ \text{LTE} = -\frac{1}{12} h^3 y_n^{(3)} \quad (4.2.26) \]

These results show that no matter which open formula is used as predictor, a single application of the corrector is sufficient to attain the full accuracy of the corrector formula. In particular we have the LTE in (a) is similar to the LTE in (d) and that the LTE in (b) is similar to the LTE in (c). Therefore, due to the simplicity of the linear open formula, we would suggest that formula (4.2.16) is used as the predictor, whereas the choice of the corrector is between the Trapezoidal or the GM formula.
4.2.3 Hyperbolic p.d.e example

Consider the problem of solving the hyperbolic problem

\[ \frac{\partial u}{\partial x} - \frac{\partial u}{\partial t} = K(x,t) \]

in the region \( R = \{ 0 \leq x \leq 1, \ t > 0 \} \) satisfying the initial condition

\[
\begin{align*}
    u(x,0) &= x(1-x) + 0.25 \\
    u(0,t) &= u(1,t) = 0.25e^{-t} \\
    K(x,t) &= 1 - 2x + 0.25e^{-t}.
\end{align*}
\]

The exact solution of the problem is given by

\[ u(x,t) = x(1-x) + 0.25e^{-t}. \]

Discretisation of the problem

We choose a mesh size \( \Delta x = h = 0.125, \Delta t = k = 0.125 \) on the domain as shown in Figure 4.5.

Replacing the space derivative we obtain the set of o.d.e.'s

\[ \frac{du_i}{dt} = \frac{u_{i+1} - u_{i-1}}{2h} - K(x_i,t) \]

for \( i=1,2,3,4,5,6 \) and 7.
Applying the boundary conditions for \( i = 1 \) and \( 7 \) results in the following set of o.d.e.'s:

\[
\begin{bmatrix}
\frac{d u_1}{dt} \\
\frac{d u_2}{dt} \\
\vdots \\
\frac{d u_7}{dt}
\end{bmatrix} =
\begin{bmatrix}
0 & 1 \\
-1 & 0 & 1 \\
\vdots & \ddots & \ddots \\
-1 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
u_1 \\
u_2 \\
\vdots \\
u_7
\end{bmatrix}
\begin{bmatrix}
K(x_1, t) \\
K(x_2, t) \\
\vdots \\
K(x_7, t)
\end{bmatrix} =
\begin{bmatrix}
\frac{1}{2h}(0.25e^{-t}) \\
0 \\
\vdots \\
-\frac{1}{2h}(0.25e^{-t})
\end{bmatrix}
\]

In this example, the exact solution is only second order in \( x \) but not in \( t \). The usual finite difference approximation gives an exact representation in the \( x \)-direction leaving the \( t \)-direction to be contested between the methods used for integration. For the purpose of comparison we use the GM and the Trapezoidal methods with step size \( \Delta t = 0.125 \). The results are shown in Table 4.7 where the exact solutions, numerical solutions and the errors are given separately. In addition we have also computed the RMS error measure for the seven grid points at every time step to facilitate comparison. The results establish accuracy in favour of the GM formula. The overall errors in the Trapezoidal formula are about twice as much as in the GM formula. Unfortunately, as expected, the GM formula requires twice the computational work as the Trapezoidal method.
<table>
<thead>
<tr>
<th>$t$</th>
<th>$u_1$</th>
<th>$u_2$</th>
<th>$u_3$</th>
<th>$u_4$</th>
<th>$u_5$</th>
<th>$u_6$</th>
<th>$u_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.359375E+00</td>
<td>0.437500E+00</td>
<td>0.484375E+00</td>
<td>0.500000E+00</td>
<td>0.484375E+00</td>
<td>0.437500E+00</td>
<td>0.359375E+00</td>
</tr>
<tr>
<td>0.125</td>
<td>0.329999E+00</td>
<td>0.408124E+00</td>
<td>0.454999E+00</td>
<td>0.470624E+00</td>
<td>0.454999E+00</td>
<td>0.408124E+00</td>
<td>0.329999E+00</td>
</tr>
<tr>
<td>0.250</td>
<td>0.304075E+00</td>
<td>0.382200E+00</td>
<td>0.429075E+00</td>
<td>0.444700E+00</td>
<td>0.429075E+00</td>
<td>0.382200E+00</td>
<td>0.304075E+00</td>
</tr>
<tr>
<td>0.375</td>
<td>0.281197E+00</td>
<td>0.359322E+00</td>
<td>0.406197E+00</td>
<td>0.421822E+00</td>
<td>0.406197E+00</td>
<td>0.359322E+00</td>
<td>0.281197E+00</td>
</tr>
<tr>
<td>0.500</td>
<td>0.261008E+00</td>
<td>0.339133E+00</td>
<td>0.386008E+00</td>
<td>0.401633E+00</td>
<td>0.386008E+00</td>
<td>0.339133E+00</td>
<td>0.261008E+00</td>
</tr>
<tr>
<td>0.625</td>
<td>0.243190E+00</td>
<td>0.321315E+00</td>
<td>0.368190E+00</td>
<td>0.383815E+00</td>
<td>0.368190E+00</td>
<td>0.321315E+00</td>
<td>0.243190E+00</td>
</tr>
<tr>
<td>0.750</td>
<td>0.227467E+00</td>
<td>0.305592E+00</td>
<td>0.352467E+00</td>
<td>0.368092E+00</td>
<td>0.352467E+00</td>
<td>0.305592E+00</td>
<td>0.227467E+00</td>
</tr>
<tr>
<td>0.875</td>
<td>0.213591E+00</td>
<td>0.291716E+00</td>
<td>0.338591E+00</td>
<td>0.354216E+00</td>
<td>0.338591E+00</td>
<td>0.291716E+00</td>
<td>0.213591E+00</td>
</tr>
<tr>
<td>1.000</td>
<td>0.201345E+00</td>
<td>0.279470E+00</td>
<td>0.326345E+00</td>
<td>0.341970E+00</td>
<td>0.326345E+00</td>
<td>0.279470E+00</td>
<td>0.201345E+00</td>
</tr>
</tbody>
</table>

Table 4.7(a) Exact Solution
### (i) Numerical solution by using GM formula

<table>
<thead>
<tr>
<th>$t$</th>
<th>$u_1$</th>
<th>$u_2$</th>
<th>$u_3$</th>
<th>$u_4$</th>
<th>$u_5$</th>
<th>$u_6$</th>
<th>$u_7$</th>
<th>CPU time ($\mu$ sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.359375E+00</td>
<td>0.437500E+00</td>
<td>0.484375E+00</td>
<td>0.500000E+00</td>
<td>0.484375E+00</td>
<td>0.437500E+00</td>
<td>0.359375E+00</td>
<td>187271</td>
</tr>
<tr>
<td>0.125</td>
<td>0.330023E+00</td>
<td>0.408142E+00</td>
<td>0.470643E+00</td>
<td>0.453018E+00</td>
<td>0.408142E+00</td>
<td>0.330014E+00</td>
<td>0.304094E+00</td>
<td>188973</td>
</tr>
<tr>
<td>0.250</td>
<td>0.304128E+00</td>
<td>0.382230E+00</td>
<td>0.429113E+00</td>
<td>0.444735E+00</td>
<td>0.382230E+00</td>
<td>0.339164E+00</td>
<td>0.304094E+00</td>
<td>188973</td>
</tr>
<tr>
<td>0.375</td>
<td>0.281281E+00</td>
<td>0.359356E+00</td>
<td>0.406254E+00</td>
<td>0.421868E+00</td>
<td>0.359356E+00</td>
<td>0.304094E+00</td>
<td>0.281215E+00</td>
<td>188616</td>
</tr>
<tr>
<td>0.500</td>
<td>0.261121E+00</td>
<td>0.330853E+00</td>
<td>0.386085E+00</td>
<td>0.391614E+00</td>
<td>0.330853E+00</td>
<td>0.261022E+00</td>
<td>0.261022E+00</td>
<td>188395</td>
</tr>
<tr>
<td>0.625</td>
<td>0.243329E+00</td>
<td>0.313396E+00</td>
<td>0.368288E+00</td>
<td>0.354244E+00</td>
<td>0.313396E+00</td>
<td>0.243203E+00</td>
<td>0.243203E+00</td>
<td>189175</td>
</tr>
<tr>
<td>0.750</td>
<td>0.227624E+00</td>
<td>0.306061E+00</td>
<td>0.352522E+00</td>
<td>0.321399E+00</td>
<td>0.306061E+00</td>
<td>0.227481E+00</td>
<td>0.227481E+00</td>
<td>189175</td>
</tr>
<tr>
<td>0.875</td>
<td>0.213612E+00</td>
<td>0.291717E+00</td>
<td>0.338652E+00</td>
<td>0.291717E+00</td>
<td>0.291717E+00</td>
<td>0.213612E+00</td>
<td>0.213612E+00</td>
<td>186158</td>
</tr>
<tr>
<td>1.000</td>
<td>0.201520E+00</td>
<td>0.279458E+00</td>
<td>0.326417E+00</td>
<td>0.279458E+00</td>
<td>0.279458E+00</td>
<td>0.201520E+00</td>
<td>0.201520E+00</td>
<td>174626</td>
</tr>
</tbody>
</table>

### (ii) Errors by using GM formula

<table>
<thead>
<tr>
<th>$t$</th>
<th>Err $u_1$</th>
<th>Err $u_2$</th>
<th>Err $u_3$</th>
<th>Err $u_4$</th>
<th>Err $u_5$</th>
<th>Err $u_6$</th>
<th>Err $u_7$</th>
<th>RMS Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.125</td>
<td>0.238845E-04</td>
<td>0.179176E-04</td>
<td>0.193912E-04</td>
<td>0.179176E-04</td>
<td>0.179176E-04</td>
<td>0.143437E-04</td>
<td>0.179176E-04</td>
<td>0.000881E-04</td>
</tr>
<tr>
<td>0.250</td>
<td>0.528769E-04</td>
<td>0.297312E-04</td>
<td>0.378340E-04</td>
<td>0.341137E-04</td>
<td>0.297312E-04</td>
<td>0.190927E-04</td>
<td>0.297312E-04</td>
<td>0.000357E-04</td>
</tr>
<tr>
<td>0.375</td>
<td>0.837817E-04</td>
<td>0.339500E-04</td>
<td>0.491715E-04</td>
<td>0.447780E-04</td>
<td>0.359500E-04</td>
<td>0.179599E-03</td>
<td>0.339500E-04</td>
<td>0.000375E-03</td>
</tr>
<tr>
<td>0.500</td>
<td>0.113214E-03</td>
<td>0.312694E-04</td>
<td>0.548648E-04</td>
<td>0.505867E-04</td>
<td>0.312694E-04</td>
<td>0.148043E-04</td>
<td>0.312694E-04</td>
<td>0.000695E-03</td>
</tr>
<tr>
<td>0.625</td>
<td>0.138462E-03</td>
<td>0.236889E-04</td>
<td>0.597828E-04</td>
<td>0.553233E-04</td>
<td>0.236889E-04</td>
<td>0.127506E-03</td>
<td>0.236889E-04</td>
<td>0.001660E-03</td>
</tr>
<tr>
<td>0.750</td>
<td>0.157783E-03</td>
<td>0.132517E-04</td>
<td>0.615843E-04</td>
<td>0.558330E-04</td>
<td>0.132517E-04</td>
<td>0.103935E-03</td>
<td>0.132517E-04</td>
<td>0.002067E-03</td>
</tr>
<tr>
<td>0.875</td>
<td>0.170263E-03</td>
<td>0.125228E-05</td>
<td>0.612823E-05</td>
<td>0.561782E-05</td>
<td>0.125228E-05</td>
<td>0.125158E-03</td>
<td>0.125228E-05</td>
<td>0.002273E-03</td>
</tr>
<tr>
<td>1.000</td>
<td>0.175424E-03</td>
<td>0.116566E-04</td>
<td>0.610429E-04</td>
<td>0.571647E-04</td>
<td>0.116566E-04</td>
<td>0.116585E-04</td>
<td>0.116566E-04</td>
<td>0.002356E-03</td>
</tr>
</tbody>
</table>

Table 4.7(b)
### (i) Numerical solution by using Trapezoidal formula

<table>
<thead>
<tr>
<th>(t)</th>
<th>(u_1)</th>
<th>(u_2)</th>
<th>(u_3)</th>
<th>(u_4)</th>
<th>(u_5)</th>
<th>(u_6)</th>
<th>(u_7)</th>
<th>CPU time ((\mu) sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000</td>
<td>0.359375E+00</td>
<td>0.437500E+00</td>
<td>0.484375E+00</td>
<td>0.500000E+00</td>
<td>0.484375E+00</td>
<td>0.437500E+00</td>
<td>0.359375E+00</td>
<td>93379</td>
</tr>
<tr>
<td>0.125</td>
<td>0.329952E+00</td>
<td>0.408088E+00</td>
<td>0.454961E+00</td>
<td>0.470586E+00</td>
<td>0.454961E+00</td>
<td>0.408088E+00</td>
<td>0.329952E+00</td>
<td>94080</td>
</tr>
<tr>
<td>0.250</td>
<td>0.303970E+00</td>
<td>0.382140E+00</td>
<td>0.429007E+00</td>
<td>0.444630E+00</td>
<td>0.429007E+00</td>
<td>0.382140E+00</td>
<td>0.303970E+00</td>
<td>93597</td>
</tr>
<tr>
<td>0.375</td>
<td>0.281030E+00</td>
<td>0.359253E+00</td>
<td>0.406084E+00</td>
<td>0.421730E+00</td>
<td>0.406084E+00</td>
<td>0.359253E+00</td>
<td>0.281030E+00</td>
<td>93377</td>
</tr>
<tr>
<td>0.500</td>
<td>0.260781E+00</td>
<td>0.339069E+00</td>
<td>0.385833E+00</td>
<td>0.401533E+00</td>
<td>0.385833E+00</td>
<td>0.339069E+00</td>
<td>0.260781E+00</td>
<td>94019</td>
</tr>
<tr>
<td>0.625</td>
<td>0.242912E+00</td>
<td>0.321267E+00</td>
<td>0.368037E+00</td>
<td>0.383728E+00</td>
<td>0.368037E+00</td>
<td>0.321267E+00</td>
<td>0.242912E+00</td>
<td>93297</td>
</tr>
<tr>
<td>0.750</td>
<td>0.227149E+00</td>
<td>0.305564E+00</td>
<td>0.352366E+00</td>
<td>0.368084E+00</td>
<td>0.352366E+00</td>
<td>0.305564E+00</td>
<td>0.227149E+00</td>
<td>94419</td>
</tr>
<tr>
<td>0.875</td>
<td>0.213247E+00</td>
<td>0.291712E+00</td>
<td>0.338322E+00</td>
<td>0.354208E+00</td>
<td>0.338322E+00</td>
<td>0.291712E+00</td>
<td>0.213247E+00</td>
<td>94419</td>
</tr>
<tr>
<td>1.000</td>
<td>0.200991E+00</td>
<td>0.279493E+00</td>
<td>0.326074E+00</td>
<td>0.342014E+00</td>
<td>0.326074E+00</td>
<td>0.279493E+00</td>
<td>0.200991E+00</td>
<td>93686</td>
</tr>
</tbody>
</table>

### (ii) Errors by using Trapezoidal formula

<table>
<thead>
<tr>
<th>(t)</th>
<th>(Err\ u_1)</th>
<th>(Err\ u_2)</th>
<th>(Err\ u_3)</th>
<th>(Err\ u_4)</th>
<th>(Err\ u_5)</th>
<th>(Err\ u_6)</th>
<th>(Err\ u_7)</th>
<th>RMS Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.125</td>
<td>-0.472666E-04</td>
<td>-0.361052E-04</td>
<td>-0.387191E-04</td>
<td>-0.390055E-04</td>
<td>-0.377683E-04</td>
<td>-0.361060E-04</td>
<td>-0.292171E-04</td>
<td>0.100325E-03</td>
</tr>
<tr>
<td>0.250</td>
<td>-0.105107E-03</td>
<td>-0.602712E-04</td>
<td>-0.699406E-04</td>
<td>-0.696448E-04</td>
<td>-0.660269E-04</td>
<td>-0.602712E-04</td>
<td>-0.387023E-04</td>
<td>0.187318E-03</td>
</tr>
<tr>
<td>0.375</td>
<td>-0.167235E-03</td>
<td>-0.691166E-04</td>
<td>-0.921966E-04</td>
<td>-0.902624E-04</td>
<td>-0.691166E-04</td>
<td>-0.921966E-04</td>
<td>-0.360353E-04</td>
<td>0.261379E-03</td>
</tr>
<tr>
<td>0.500</td>
<td>-0.267342E-03</td>
<td>-0.637915E-04</td>
<td>-0.996131E-04</td>
<td>-0.918213E-04</td>
<td>-0.637915E-04</td>
<td>-0.996131E-04</td>
<td>-0.293621E-04</td>
<td>0.323277E-03</td>
</tr>
<tr>
<td>0.625</td>
<td>-0.277985E-03</td>
<td>-0.484090E-04</td>
<td>-0.196136E-04</td>
<td>-0.106352E-04</td>
<td>-0.484090E-04</td>
<td>-0.196136E-04</td>
<td>-0.245030E-04</td>
<td>0.374103E-03</td>
</tr>
<tr>
<td>0.750</td>
<td>-0.317391E-03</td>
<td>-0.273310E-04</td>
<td>-0.545643E-04</td>
<td>-0.110310E-04</td>
<td>-0.273310E-04</td>
<td>-0.545643E-04</td>
<td>-0.273310E-04</td>
<td>0.415211E-03</td>
</tr>
<tr>
<td>0.875</td>
<td>-0.343118E-03</td>
<td>-0.339070E-04</td>
<td>-0.732157E-03</td>
<td>-0.120567E-03</td>
<td>-0.339070E-04</td>
<td>-0.732157E-03</td>
<td>-0.339070E-04</td>
<td>0.448044E-03</td>
</tr>
<tr>
<td>1.000</td>
<td>-0.354216E-03</td>
<td>-0.226783E-04</td>
<td>-0.438131E-04</td>
<td>-0.140787E-03</td>
<td>-0.438131E-04</td>
<td>-0.140787E-03</td>
<td>-0.438131E-04</td>
<td>0.474088E-03</td>
</tr>
</tbody>
</table>

**Table 4.7(c)**
4.3 THE GM FORMULA WITH EXTRAPOLATION

4.3.1 Introduction

One popular scheme developed by Bulirsch and Stoer [1966] uses a second order method (usually leapfrog) and extrapolation (which may be Richardson but more commonly is polynomial or rational extrapolation) to achieve high order accuracy. The basic concept of extrapolation is best explained by an example.

A number, $T_0$, is required to be evaluated and, as is often the case in numerical analysis, an approximation, $T(h)$ is computed. $h$ is a positive discretization parameter, typically step length where $T(h) \rightarrow T_0$ as $h \rightarrow 0$.

Suppose that for every fixed $N$, $T(h)$ possesses an asymptotic expansion of the form,

$$T(h) \sim T_0 + T_1 h + T_2 h^2 + \ldots + T_N h^N + R_N(h),$$

where the coefficients $T_0, T_1, \ldots, T_N$ are independent of $h$. The above can be summarised to be

$$T(h) \sim T_0 + T_1 h + T_2 h^2 + \ldots$$

(4.3.1)

Assume two approximations are computed for $T_0$, say $T(h_0)$ and $T(h_0/2)$. Then

$$T(h_0) = T_0 + O(h_0) \text{ and }$$

$$T(h_0/2) = T_0 + O(h_0) \text{ as } h \rightarrow 0.$$  

Then, by using a linear combination of $T(h_0)$ and $T(h_0/2)$, a better approximation, that is a higher order approximation, can be obtained, namely

$$2T(h_0/2) - T(h_0) = T_0 + O(h_0^2).$$

(4.3.2)

The extrapolation formulae to be discussed will be based on repeated use of this technique (i.e. eliminating more and more leading terms by additional linear combinations). This repeated extrapolation is known as Richardson Extrapolation (Richardson [1927], pp.229-349) and is very often used to estimate principal local truncation errors.
This process can be continued by incorporating a \( T(h_0/4) \) term and forming a further linear combination which will eliminate another term of the expansion (4.3.1) to give \( T_0 \rightarrow O(h_0^3) \).

It is not necessary to consider a strict sequence of step sizes: \( h_0, h_0/2, h_0/4, \ldots \), as we have done in the example, in fact certain criteria will decide the actual sequence used. However, as we will see later on, certain condition must always be satisfied for a set of \( h_i \) to ensure convergence of the process. From the above, a linear combination can be found with the property:

\[
\sum_{i=0}^{m} c_{im} T(h_i) = T_0 + O(h_0^m) \text{ as } h \rightarrow 0,
\]

where the \( c_{im} \) are the coefficients obtained. The forming of such linear combinations is essentially equivalent to polynomial interpolation at \( h=0 \), of the data \( (h_i, T(h_i)), i=0,1,\ldots,m \) and since \( h_i > 0 \), \( i=0,1,\ldots,m \) the process is strictly one of extrapolation. An algorithm devised by Neville and Aitken makes it possible to avoid computing the coefficients \( c_{im} \) in (4.3.3) as the extrapolation is performed iteratively. The process can be summarised by the tableau shown in Table 4.8.

\begin{table}[h]
\begin{align*}
\begin{array}{ccccccc}
\text{ } & P_0(0) & P_0(1) & P_0(2) & \cdots & P_0(s) \\
P_0(0) & P_1(0) & P_1(1) & \cdots & & \\
P_1(0) & P_2(0) & \cdots & & & \\
\vdots & \vdots & \vdots & \ddots & \ddots & \ddots \\
P_s(0) & & & & P_s(1) & \cdots & P_s(s) \\
\end{array}
\end{align*}
\end{table}

Table 4.8
The entries in Table 4.8 are determined column by column such that

\[ p_i^{(0)} = T(h_i) \quad i=0,1,\ldots,s \]

and

\[ p_i^{(j)} = p_{i+1}^{(j-1)} + \frac{p_{i+1}^{(j-1)} - p_i^{(j-1)}}{h_i/h_{i+j}} \quad (4.3.4) \]

\[ j = 1,2,\ldots,s \]

\[ i = 0,1,\ldots,s-j. \]

If \( p_i^{(j)}(h) \) is defined by replacing \( h_i \) and \( h_{i+j} \) in (4.3.4) by \( h_i-h \) and \( h_{i+j}-h \), it is obvious that as \( p_i^{(0)}(h) \) is a constant, \( p_i^{(j)}(h) \) is a polynomial of degree 3 such that

\[ p_i^{(j)}(h_i) = p_i^{(j-1)}(h_i) = \cdots = p_i^{(0)}(h) = T(h_i), \]

that is \( p_i^{(j)}(h) \) is an interpolating polynomial. \( p_i^{(j)} \) is the interpolating polynomial defined by \( T(h_1)\ldots T(h_{i+j}) \), evaluated at \( h=0 \).

It follows directly that the error in the extrapolation algorithm (4.3.4) is given by

\[ T_0 = p_i^{(j)} + h_i h_{i+1} \cdots h_{i+j} E_1^{(j)}. \quad (4.3.5) \]

The coefficient \( E_1^{(j)} \) in the remainder term can be expressed in many ways as a divided difference (Davis [1963]), or as \( T^{(j+1)}(E_1^{(j)})/(j+1)! \)

where \( E_1^{(j)} \) is \( \min s_1^{(j)}, \max s_1^{(j)} \)

\[ s_1^{(j)} = \{0, h_1, \ldots, h_{i+j}\}. \]

The sequence \( \{h_1\} \) can have the following forms,

\[ \{h_0, h_0/2, h_0/4, h_0/8, h_0/16, \ldots\} \quad (4.3.6) \]

\[ \{h_0, h_0/2, h_0/3, h_0/4, h_0/5, \ldots\} \quad (4.3.7) \]

\[ \{h_0, h_0/2, h_0/3, h_0/4, h_0/6, h_0/8, h_0/12, \ldots\}, \quad (4.3.8) \]

thus the remainder in (4.3.5) is \( O(h_0^{(j+1)}) \). The sequence (4.3.6) yields accurate results but at the expense of doubling the number of operations. Equation (4.3.7) is cheaper to compute but leads to an unstable form of the extrapolation algorithm as the denominator.
Bulirsch and Stoer [1964] propose the sequence (4.3.8) which has the advantage of the other two since it leads to a stable algorithm but does not double the cost of the calculation for each additional row of the tableau. After stating the first three terms $h_0, h_0/2$ and $h_0/3$ the remaining terms of sequence (4.3.7) are formed by doubling the denominators of the last but one terms.

4.3.2 Asymptotic Expansion

Increasing the accuracy of Euler's method, backward Euler and the trapezoidal method by extrapolation depends upon their approximations

$$T(h) = y_n = y(x_n; h),$$

(4.3.9)

obtained by the application of the respective formulae. This in turn depends on the solution being a function that has an asymptotic expansion

$$y(x; h) = y(x) + \sum_{v=0}^{k} c_v(x) h^v + O(h^{k+1}).$$

(4.3.10)

Both $\gamma$ and $c_v(x)$ depend upon the numerical method used. Examples with $\gamma=1$ are Euler's method and the backward Euler method. The trapezoidal method is an example with $\gamma=2$. The $c_v(x)$ satisfy differential equations of the same type that $y(x)$ does (Steadman [1981]).

Gragg [1965] develops the asymptotic expansion for Euler's method with the prototype equation

$$y' = \lambda y, \quad y(a) = \eta$$

where $\lambda$ is a constant. The asymptotic expansion for the backward Euler and the trapezoidal method can be found, for example, in Steadman [1981]. The coefficients of these three expansions are also listed in Table 4.10 for the purpose of comparison with the asymptotic expansion of our GM formula which we will now consider.
Asymptotic expansion of the GM solution

We will now develop the asymptotic expansion for the solution by the GM method of the prototype equation

\[ y' = \lambda y, \quad y(a) = \eta \]  

(4.3.11)

This method gives us

\[ y_0 = \eta \text{ and} \]

\[ y_{n+1} = y_n + h \sqrt{y_n y_{n+1}} \]

which reduces to

\[ y_{n+1} = \left[ \frac{h \lambda + \sqrt{h^2 \lambda^2 + 4}}{4} \right]^2 y_n \]

\[ = \left[ \frac{h^2 \lambda^2 + h^2 \lambda^2 + 4 + 2h \lambda \sqrt{h^2 \lambda^2 + 4}}{4} \right] y_n \]

\[ = [1 + \frac{1}{4} (h^2 \lambda^2 + h \lambda \sqrt{h^2 \lambda^2 + 4})] y_n. \]

By successive substitutions we have

\[ y_n = [1 + \frac{1}{4} (h^2 \lambda^2 + h \lambda \sqrt{h^2 \lambda^2 + 4})]^n \eta. \]  

(4.3.12)

Since \( x = a + nh \), we substitute \( n = (x-a)/h \) in the above expression to obtain

\[ y(x; h) = [1 + \frac{1}{4} (h^2 \lambda^2 + h \lambda \sqrt{h^2 \lambda^2 + 4})]^{(x-a)/h} \eta \]

\[ = \exp \left[ \frac{(x-a)}{h} \right] \ln \left[ 1 + \frac{1}{4} (h^2 \lambda^2 + h \lambda \sqrt{h^2 \lambda^2 + 4}) \right] \eta \]

\[ = \exp \left[ \frac{\lambda (x-a)}{h \lambda} \right] \ln \left[ 1 + \frac{1}{4} (h^2 \lambda^2 + h \lambda \sqrt{h^2 \lambda^2 + 4}) \right] \eta \]  

(4.3.13)
\[
\exp \left\{ \frac{\lambda(x-a)}{\mu} \ln \left[ 1 + \frac{1}{4} \left( \mu^2 + \mu \sqrt{\mu^2 + 4} \right) \right] \right\} \eta
\]

where we have used \( h\lambda = \mu \) for simplicity in the algebra. Now, by using the binomial expansion we have

\[
\sqrt{4 + \mu^2} = 2 \sqrt{1 + \mu^2/4}
\]

\[
= 2 \left[ 1 + \frac{1}{4} \left( \mu^2/4 \right) - \frac{1}{8} \left( \mu^4/16 \right) + \frac{1}{16} (\mu^6/64) - \ldots \right]
\]

\[
= 2 + \mu^2/4 - \mu^4/64 + \mu^6/512 - \ldots
\]

Therefore the expression contained in the logarithm becomes

\[
\ln \left[ 1 + \frac{1}{4} \left( \mu^2 + 2\mu + \frac{\mu^3}{4} - \frac{\mu^5}{64} + \frac{\mu^7}{512} - \ldots \right) \right]
\]

\[
= \ln \left[ 1 + \left( \mu + \frac{3}{8} \mu^2 + \frac{\mu^3}{8} - \frac{\mu^5}{128} + \frac{\mu^7}{1024} - \ldots \right) \right]
\]

and using \( \ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots \), this becomes

\[
= \mu + \frac{3}{8} \mu^2 - \frac{\mu^5}{128} + \frac{\mu^7}{1024} + \ldots
\]

\[
- \frac{1}{4} \left( \mu^2 + \mu^3 + \frac{3\mu^4}{2} + \frac{\mu^5}{8} + 0\mu^6 - \frac{\mu^7}{128} + \ldots \right)
\]

\[
+ \frac{1}{3} \left( \mu^3 + \frac{3\mu^4}{2} + \frac{9\mu^5}{8} + \frac{3\mu^6}{8} + \frac{7\mu^7}{128} + \ldots \right) - \frac{1}{4} \left( \mu^4 + 2\mu^5 + 2\mu^6 + \frac{5\mu^7}{4} + \ldots \right) + \frac{1}{5} \left( \mu^5 + \frac{3\mu^6}{2} + \frac{25\mu^7}{8} + \ldots \right) - \frac{1}{6} (\mu^6 + 2\mu^7 + \ldots)
\]

\[
= \mu - \mu^3/24 + 3\mu^5/640 - \mu^6/5 + 3121\mu^7/21504 + \ldots
\]

Thus equation (4.3.13) becomes
\( y(x; h) = \eta \exp \left\{ \lambda(x-a)(1- \frac{h^2 \lambda^2}{24} + \frac{3h^4 \lambda^4}{640} - \frac{h^5 \lambda^5}{5} + \frac{3121h^6 \lambda^6}{21504} + \ldots ) \right\} \) \hspace{1cm} (4.3.14)

This is of the form

\[ y(x; h) = \sum_{v=0}^{k} c_v(x)h^v + R_k(x; h)h^{k+1} \] \hspace{1cm} (4.3.15)

To obtain \( c_v(x) \) we assume \( k = \infty \), differentiate (4.3.14) and (4.3.15), and equate them:

\[ \lambda \left[ 1 - \frac{h^2 \lambda^2}{24} + \frac{3h^4 \lambda^4}{640} - \frac{h^5 \lambda^5}{5} + \frac{3121h^6 \lambda^6}{21504} + \ldots \right] \sum_{v=0}^{\infty} c_v(x)h^v = \sum_{v=0}^{\infty} c_v'(x)h^v \] \hspace{1cm} (4.3.16)

Isolating equal powers of \( h \) produces,

\[ h^0: \quad c_0'(x) = \lambda c_0(x) \]
\[ h^1: \quad c_1'(x) = \lambda c_1(x) \]
\[ h^2: \quad c_2'(x) = \lambda c_2(x) - \frac{\lambda^3}{24} c_0(x) \]
\[ h^3: \quad c_3'(x) = \lambda c_3(x) - \frac{\lambda^3}{24} c_1(x) \]
\[ h^4: \quad c_4'(x) = \lambda c_4(x) - \frac{\lambda^3}{24} c_2(x) + \frac{3\lambda^5}{640} c_0(x) \]
\[ h^5: \quad c_5'(x) = \lambda c_5(x) + \frac{3\lambda^5}{640} c_1(x) - \frac{\lambda^6}{5} c_0(x) \]
\[ \vdots \]
The initial value conditions require that $c_0(a) = \eta$ and $c_v(a) = 0$ for $v \geq 1$.

Our next task is to obtain the solution $c_v(x)$, $v=0,1,2,...$ of the differential equations recursively.

The solution for the first equation is:

$$c_0(x) = \exp[\lambda(x-a)]\eta$$

and for the second equation is

$$c_1(x) = 0.$$ 

Substituting $c_0(x)$ into the third equation produces

$$c_2'(x) = \lambda c_2(x) - \frac{\lambda^3}{24} \exp[\lambda(x-a)]\eta, \ c_2(a) = 0,$$

the solution of which is

$$c_2(x) = -\frac{\lambda^3}{24} \eta(x-a) \exp[\lambda(x-a)].$$

Substituting $c_1=0$ in the fourth equation, we have

$$c_3'(x) = \lambda c_3(x), \ c_3(a) = 0.$$ 

The solution is $c_3(x) = 0$.

Substituting $c_2(x)$ and $c_0(x)$ in the fifth equation produces

$$c_4'(x) - \lambda c_4(x) = \left[\frac{3\lambda^5}{640} + \frac{\lambda^6}{376}(x-a)\right] \eta e^{\lambda(x-a)}, \ c_4(a) = 0.$$ 

The solution is

$$c_4(x) = \eta e^{\lambda(x-a)} \left[\frac{3\lambda^5}{640} (x-a) + \frac{\lambda^6}{1152} (x-a)^2\right].$$

The sixth equation now becomes

$$c_5'(x) - \lambda c_5(x) = -\frac{\lambda^6}{5} \eta e^{-\lambda(x-a)}, \ c_5(a) = 0,$$
the solution of which is

$$c_5(x) = -\frac{\lambda^6}{5} \eta(x-a)e^{\lambda(x-a)}.$$  

Higher degree coefficients can be obtained by continuing further the substitutions and solving the differential equations that result. The coefficients $c_v(x)$ for $v=0(1)5$ are tabulated in Table 4.9.

$$c_0(x) = \exp[\lambda(x-a)] \eta$$  

$c_1(x) = 0$

$$c_2(x) = -\left[\frac{\lambda^3}{24}(x-a)\right]c_0(x)$$  

$c_3(x) = 0$

$$c_4(x) = \left[\frac{3\lambda^5}{640}(x-a) + \frac{\lambda^6}{1152}(x-a)^2\right]c_0(x)$$  

$c_5(x) = -\left[\frac{\lambda^6}{5}(x-a)\right]c_0(x)$

| Table 4.9 |
|---|---|

For the purpose of comparison we tabulate in Table 4.10 the first six terms of the coefficients $c_v(x)$ of the asymptotic expansion of the four closely related numerical methods we have already discussed, i.e. Euler, backward Euler, Trapezoidal and GM methods.
<table>
<thead>
<tr>
<th></th>
<th>Euler</th>
<th>Backward Euler</th>
<th>Trapezoidal</th>
<th>GM</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_0$</td>
<td>$\exp[\lambda(x-a)]\eta$</td>
<td>$\exp[\lambda(x-a)]\eta$</td>
<td>$\exp[\lambda(x-a)]\eta$</td>
<td>$\exp[\lambda(x-a)]\eta$</td>
</tr>
<tr>
<td>$c_1$</td>
<td>$-\frac{\lambda^2(x-a)}{2}c_0$</td>
<td>$\frac{\lambda^2(x-a)}{2}c_0$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$c_2$</td>
<td>$\frac{\lambda^3(x-a)}{3}$</td>
<td>$\frac{\lambda^3(x-a)}{3}$</td>
<td>$\frac{\lambda^3(x-a)}{12}c_0$</td>
<td>$-\frac{\lambda^3(x-a)}{24}c_0$</td>
</tr>
<tr>
<td></td>
<td>$\frac{\lambda^4(x-a)^2}{8}c_0$</td>
<td>$\frac{\lambda^4(x-a)^2}{8}c_0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c_3$</td>
<td>$-\frac{\lambda^4(x-a)}{4}$</td>
<td>$\frac{\lambda^4(x-a)}{4}$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$\frac{\lambda^5(x-a)^2}{6}$</td>
<td>$\frac{\lambda^5(x-a)^2}{6}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\frac{\lambda^6(x-a)^3}{48}c_0$</td>
<td>$\frac{\lambda^6(x-a)^3}{48}c_0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c_4$</td>
<td>$\frac{\lambda^5(x-a)}{5}$</td>
<td>$\frac{\lambda^5(x-a)}{5}$</td>
<td>$\frac{\lambda^5(x-a)}{80}$</td>
<td>$\frac{3\lambda^5(x-a)}{640}$</td>
</tr>
<tr>
<td></td>
<td>$\frac{13\lambda^6(x-a)^2}{72}$</td>
<td>$\frac{13\lambda^6(x-a)^2}{72}$</td>
<td>$\frac{\lambda^6(x-a)^2}{288}$</td>
<td>$\frac{\lambda^6(x-a)^2}{1152}$</td>
</tr>
<tr>
<td></td>
<td>$\frac{\lambda^7(x-a)^3}{24}$</td>
<td>$\frac{\lambda^7(x-a)^3}{24}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\frac{\lambda^8(x-a)^4}{384}c_0$</td>
<td>$\frac{\lambda^8(x-a)^4}{384}c_0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c_5$</td>
<td>$-\frac{\lambda^6(x-a)}{6}$</td>
<td>$\frac{\lambda^6(x-a)}{6}$</td>
<td>0</td>
<td>$-\frac{\lambda^6(x-a)}{5}c_0$</td>
</tr>
<tr>
<td></td>
<td>$\frac{11\lambda^7(x-a)^2}{60}$</td>
<td>$\frac{11\lambda^7(x-a)^2}{60}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\frac{17\lambda^8(x-a)^3}{288}$</td>
<td>$\frac{17\lambda^8(x-a)^3}{288}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\frac{\lambda^9(x-a)^4}{144}c_0$</td>
<td>$\frac{\lambda^9(x-a)^4}{144}c_0$</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\frac{\lambda^{10}(x-a)^5}{3840}c_0$</td>
<td>$\frac{\lambda^{10}(x-a)^5}{3840}c_0$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 4.10: Coefficients for the four methods**

**solution of** $y' = \lambda y, \ y(a) = \eta$
From Table 4.10 we see that for the Euler and the backward Euler methods the coefficients $c_v(x)$ are all nonzero while for the Trapezoidal method these coefficients are non-zero only for $v = \text{even number}$. Therefore, the asymptotic expansion of the Euler and backward Euler methods contain all terms in (4.3.10) and correspondingly $\gamma = 1$. For the Trapezoidal method equation (4.3.10) contains only even powers of $h$ and therefore $\gamma = 2$.

However, for the GM method a different pattern of expansion emerges. The first two terms are of even power and the remainder contains every term in equation (4.3.10). Hence $\gamma = 2$ for $v=0,1$, and $\gamma = 1$ otherwise. (In practice, however, using $\gamma = 2$ for all $v$ results in more accurate numerical solution).

Thus, with two-point extrapolation for example, we have results with error terms of order 2 in the Euler and backward Euler methods and of order 4 in the Trapezoidal and GM methods. The differences in order are shown in Table 4.11.

<table>
<thead>
<tr>
<th>No. of Extrapolation points</th>
<th>Order of accuracy</th>
<th>Euler</th>
<th>Backward Euler</th>
<th>Trapezoidal</th>
<th>GM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 1</td>
<td>2 2</td>
<td>2 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2 2</td>
<td>4 4</td>
<td>4 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3 3</td>
<td>6 6</td>
<td>5 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4 4</td>
<td>8 8</td>
<td>6 6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>n n</td>
<td>2n n+2</td>
<td>n+2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 4.11: Order of accuracy of the four methods in the extrapolation process as functions of number of extrapolation points**
By looking at the magnitude of the coefficients of $h^2$ and $h^4$ in the Trapezoidal and the GM methods we find that the coefficients for the GM formula are smaller in magnitudes than the corresponding coefficients in the Trapezoidal method. Thus, we may say that the GM method is theoretically the best amongst these four methods when used with two-point extrapolation (i.e. with $s=1$). Theoretically, we have to limit $s$ to have the maximum value 1 and monitor the basic steplength to suit our needs in accuracy for the GM formula to be more effective than the Trapezoidal formula when used with extrapolation. In practice, however, even when applied with more extrapolation points the GM method still shows greater accuracy compared to the Trapezoidal method over a few steps of integration in some problems.

It has been a fact that the benefits of repeated extrapolation are greatly enhanced if it happens that the asymptotic expansion for $y(x; h)$ contains only even powers of $h$, but it is probably too much a requirement since in practice we normally use not more than 6 or 7 extrapolation points so that it would be sufficient to have a method with asymptotic expansion of even powers up to probably $h^{12}$ or $h^{14}$ only. As we have seen the GM method is an example of a method with this characteristic with even powers only up to $h^4$.

Having obtained the asymptotic expansion of the GM method we may now proceed with extrapolation. This begins with a sequence of positive integers $N_0, N_1, \ldots, N_s$ and their associated tabular intervals $h_j = (b-a)/N_j$, $j=0, 1, \ldots, s$. For numerical stability in extrapolation it is required that $h_0 > h_1 > \ldots > h_s > 0$ and $s = \text{sup}(h_{n+1}/h_n) < 1$ (Gragg [1965]).

The benefits of repeated extrapolation are clearly greatly enhanced if it happens that the asymptotic expansion for $T(h)$ in (4.3.1) contains only even powers of $h$ - which strengthens the argument for using the sequence (4.3.8), since it can be used without increasing the
sensitivity to round off. For example, corresponding to (4.3.2) the following will occur

\[ \frac{4}{3} T(h_0/2) - \frac{1}{3} T(h_0) = T_0 + O(h_0^4). \]

This is the situation that happens to the trapezoidal method, and as we have seen earlier, since the coefficients \( c_v(x) \) in the GM formula are zero for \( v \) low odd numbers, we expect a similar situation applies to the GM formula.

If the asymptotic expansion of \( T(h) \) has the form,

\[ T(h) = T_0 + T_2 h^2 + T_4 h^4 + \ldots, \tag{4.3.17} \]

then the process of repeated extrapolation produces, instead of (4.3.4) the following,

\[
\begin{align*}
\pi^{(0)}_i &= T(h_i), \quad i=0,1,\ldots,s \\
\pi^{(j)}_i &= \pi^{(j-1)}_i + \frac{(j-1) - \pi^{(j-1)}_{i+1}}{(h_i/h_{i+j})^2 - 1} \\
&\quad j=1,2,\ldots,s; \quad i=0,1,\ldots,s-j,
\end{align*}
\tag{4.3.18}
\]

giving \( \pi^{(j)}_i = T_0 + O(h_i^{(2j+2)}) \).

Gragg [1963,1965] analyses the algorithm defined by Table 4.8 and (4.3.18) where (4.3.17) is assumed. His findings are contained in the following theorem.

**Theorem 4.1**

Let \( y(b;h) \) have an asymptotic expansion

\[ y(x;h) = y(x) + \sum_{v=1}^{k} c_v(x) h^v + O(h^{k+1}) \]

and let \( \sup_{i \geq 0} \left( \frac{h_{i+1}}{h_i} \right) \leq \alpha < 1. \) Then as \( i \to \infty \)

\[
\pi^{(j)}_i = y(b) + (-1)^j c_{j+1}(b) (h_i \ldots h_{i+j})^\gamma + O((h_i \ldots h_{i+j})^\gamma). 
\]
Further results by Gragg show that each column of Table 4.8 converges to $T_0$ faster than the one to its left, and that if, in addition, $\inf(h_{n+1}/h_n) > 0$, the principal diagonal $p_0^{(0)}$, $p_0^{(1)}$, $p_0^{(2)}$, ..., converges to $T_0$ faster than any other column. Indeed under milder conditions on $T(h)$, $p_0^{(n)}$ converges to $T_0$ superlinearly in the sense that
\[ |p_0^{(n)} - T_0| \leq K_n \text{ and } \lim_{n \to \infty} (K_{n+1}/K_n) = 0. \]

By using the GM method to compute the approximate values $y(a;h_1)$, $y(a+ih_1;h_1)$, ..., $y(a+ih_1;h_1)$ for $i = 0, 1, ..., s$, we obtain the sequence $y(a+N_0 h_0; h_0)$, $y(a+N_1 h_1; h_1)$, ..., $y(a+N_s h_s; h_s)$. We now use the Neville algorithm to produce a better approximation of $y(a+N_1 h_1; h_1) = y(b)$, i.e. the value of $y$ at the next integration point.

### 4.3.3 Application of the extrapolation method to IVP

The application of the extrapolation algorithm by building up the table of values using the recursion relations given by equation (4.3.18) has the effect of computing values in Table 4.8 column by column starting from left to right. In this procedure the number of extrapolation is fixed, i.e. for a given value of $s$ we compute all the intermediate values in the table to obtain $p_0^{(s)}$ and then take this final value to represent $y(b)$. The order of computation is indicated by the arrows in Table 4.12. This procedure has a drawback in that a check of the convergence of the sequence $p_0^{(0)}$, $p_0^{(1)}$, $p_0^{(2)}$, ... becomes inapplicable. We tend to compute up to the very final value $p_0^{(s)}$ even though convergence to a required accuracy has been achieved at an earlier stage $p_0^{(r)}$, $r<s$. Otherwise, the availability of the later values $p_{r+1}^{(0)}$, ..., $p_s^{(0)}$ become redundant. In addition, if the accuracy
of the final value is not satisfactory we have to rebuild the tableau with a larger value of s.

![Diagram](image)

**Table 4.12**

An alternative to this procedure is to compute \( P_0^{(i)} \) as soon as \( P_j^{(0)} \) become available. Effectively this will involve the computation of tabular values in the order indicated by the arrows in Table 4.13.

![Diagram](image)

**Table 4.13**
The scheme is built up by generating, at the $n$-th stage, the upward sloping diagonal beginning with $p_n^{(0)}$. Thus, the recurrence relations (4.3.18) become

$$p_1^{(0)} = y(a + N_1 h_1; h_1) = y(b; h_1)$$

$$p_{1-j}^{(j)} = p_{1-j+1}^{(j-1)} + \frac{p_{1-j+1}^{(j-1)} - p_{1-j}^{(j-1)}}{h_{j-1}}$$

for $j=1,2,\ldots,i$

for $i=0,1,\ldots,k$

where $k$ is the smallest integer such that

$$\left| \frac{p_{(k)} - p_{(k-1)}}{p_{(k-1)}} \right| < \varepsilon$$

for some pre-set tolerance $\varepsilon$.

The routine should therefore stop from computing the next $y(b)$ with a smaller step size when the convergence criterion (4.3.20) has been satisfied. Some facilities to change the basic step size, $H$ should also be provided in case the number $k$ is not achieved after a certain number of tabular points have been computed.

**Numerical Example 1**

Consider the numerical solution of the initial value problem

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

whose exact solution is $y(x) = e^x$.

For the purpose of comparison we solve this problem by using the three methods, i.e. Euler, Trapezoidal and GM methods and we apply the
Table 4.14: Polynomial extrapolation methods
by using Euler, Trapezoidal and GM formulae

<table>
<thead>
<tr>
<th>h</th>
<th>1.0000000</th>
<th>0.5000000</th>
<th>0.2500000</th>
<th>0.1250000</th>
<th>0.0625000</th>
<th>0.0312500</th>
<th>0.0156250</th>
<th>0.0078125</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Euler's method (s = 7)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>1.0000000</td>
<td>0.30000000E+01</td>
<td>0.27037037E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5000000</td>
<td>0.27777778E+01</td>
<td>0.27184794E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.2500000</td>
<td>0.27326114E+01</td>
<td>0.27182342E+01</td>
<td>0.27182842E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1250000</td>
<td>0.27218319E+01</td>
<td>0.27182811E+01</td>
<td>0.27182811E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0625000</td>
<td>0.27191673E+01</td>
<td>0.27182815E+01</td>
<td>0.27182815E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(b) Trapezoidal method (s = 4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>1.0000000</td>
<td>0.26180340E+01</td>
<td>0.27151533E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5000000</td>
<td>0.26908734E+01</td>
<td>0.27182509E+01</td>
<td>0.27182817E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.2500000</td>
<td>0.27112614E+01</td>
<td>0.27182509E+01</td>
<td>0.27182817E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1250000</td>
<td>0.27165158E+01</td>
<td>0.27182817E+01</td>
<td>0.27182817E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0625000</td>
<td>0.27178395E+01</td>
<td>0.27182818E+01</td>
<td>0.27182818E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(c) GM method (s = 4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>1.0000000</td>
<td>0.26770833E+01</td>
<td>0.27138790E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.5000000</td>
<td>0.26328154E+01</td>
<td>0.27177040E+01</td>
<td>0.27182509E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.2500000</td>
<td>0.26240483E+01</td>
<td>0.27182357E+01</td>
<td>0.27182509E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.1250000</td>
<td>0.26184495E+01</td>
<td>0.27182357E+01</td>
<td>0.27182509E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0625000</td>
<td>0.26179285E+01</td>
<td>0.27182357E+01</td>
<td>0.27182509E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0312500</td>
<td>0.26160510E+01</td>
<td>0.27182357E+01</td>
<td>0.27182509E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0156250</td>
<td>0.26176998E+01</td>
<td>0.27182357E+01</td>
<td>0.27182509E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.0078125</td>
<td>0.26183315E+01</td>
<td>0.27182357E+01</td>
<td>0.27182509E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
polynomial extrapolation procedure for the sequence $N_s = \{1, 2, 4, 8, 16, \ldots \}$ for one basic step of length $h_0 = 1.0$.

For $x = h_0 = 1.0$ the numerical values obtained by these methods are given separately in Table 4.14. Convergence of the diagonal values for the Euler method correct to the eight significant figure value $e = 2.7182818$ is attained, as expected, only for $s = 7$, whereas the Trapezoidal and the GM methods give the corresponding accuracy for $s = 4$. However, the convergence of $p_0$ to $e$ is the fastest in the earlier stages in the case of the GM method as exhibited in Table 4.15. In this table we have actually compared the ratios

$$R_n = \frac{|(n+1)p_{n+1} - e|}{|np_n - e|}$$

for each method.

The computations were carried out on the Honeywell computer at Loughborough in double precision arithmetic.

<table>
<thead>
<tr>
<th>n</th>
<th>Euler</th>
<th>Trapezoidal</th>
<th>GM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.3039</td>
<td>0.0518</td>
<td>0.0312</td>
</tr>
<tr>
<td>1</td>
<td>0.1887</td>
<td>0.0136</td>
<td>0.0099</td>
</tr>
<tr>
<td>2</td>
<td>0.1069</td>
<td>0.0036</td>
<td>0.0032</td>
</tr>
<tr>
<td>3</td>
<td>0.0572</td>
<td>0.0074</td>
<td>0.0715</td>
</tr>
</tbody>
</table>

Table 4.15

The fact that the GM formula turned out to be the best of the three methods when used with extrapolation with $s$ taking a value around 4 lies in the fact that it has a smaller first five coefficients $c_v(x)$ of the asymptotic expansion as tabulated in Table 4.10. Although the order of accuracy of the GM and the Trapezoidal methods are the same, the difference in these magnitudes has greatly enhanced the speed of
convergence more in the GM than in the Trapezoidal methods. Thus, the earlier statement on the speed of convergence is numerically confirmed.

4.3.4 Rational Extrapolation

A better strategy than the linear extrapolation that we have already discussed is the rational extrapolation of Bulirsch and Stoer [1964], with

\[
\begin{align*}
\rho_1^{(-1)} &= 0 \\
\rho_1^{(0)} &= y(a+N_1 h, h) = y(b; h), \\
\rho_1^{(j)} &= \rho_1^{(j-1)} + \frac{\rho_1^{(j-1)} - \rho_1^{(j-2)}}{\left(\frac{h}{h+1}\right)^j - \left(\frac{h}{h+1}\right)^{j-1}} - 1 \\
&\quad \text{for } j \geq 1.
\end{align*}
\]

(4.3.21)

This formula was based on the rational interpolation of Stoer [1961]. Where in equation (4.3.19) the values of \( p_1^{(j)} \) is the \( j \)-th degree polynomial in \( h^Y \) that passes through \((h^i, y(b; h_i)), (h^i+1, y(b; h^i+1)), \ldots, (h^i+j, y(b; h^i+j))\) evaluated at \( h=0 \), in equation (4.3.21) the values of \( \rho_1^{(j)} \) is the \( j \)-th degree rational function in \( h^Y \) that passes through the same points and is evaluated at \( h=0 \). This rational function has a polynomial numerator in \( h^Y \) of degree \( k \) and a polynomial denominator in \( h^Y \) of degree \( k \) or \( k-1 \) when respectively \( j=2k \) or \( j=2k+1 \).

Formula (4.3.21) is calculated in the sequence as shown by the arrows in Table 4.12. An equivalent formulation for use as in Table 4.13 is given as follows,
\[ r_i^{-1} = 0 \text{ for all } i, \]

for \( i = 0, 1, 2, \ldots, k, \)

\[ r_i^{(0)} = y(a+N_i, h_i; h_k) = y(b; h_k) \]

\[ r_{i-j} = r_{i-j+1} + \frac{r_{i-j}^{(j-1)} - r_{i-j+1}^{(j-1)}}{\left( \frac{h_{i-j}}{h_i} \right)^{\gamma} \left[ 1 - \frac{r_{i-j+1}^{(j-1)} - r_{i-j+1}^{(j-2)}}{r_{i-j+1}^{(j-1)} - r_{i-j+1}^{(j-2)}} \right] - 1} \]

\( j = 1, 2, \ldots, i \)

**Definition 4.1.**

Given a sequence of numbers \( c_0, c_1, c_2, \ldots \) and \( c_v = 0 \) for \( v < 0, \) then a Hankel determinant is defined as

\[ H_{p, q} = \begin{bmatrix}
\begin{array}{cccc}
  c_{p-q+1} & c_{p-q+2} & \cdots & c_p \\
  c_{p-q+2} & c_{p-q+3} & \cdots & c_{p+1} \\
  \vdots & \vdots & \ddots & \vdots \\
  c_p & c_{p+1} & \cdots & c_{p+q-1}
\end{array}
\end{bmatrix} \]

e.g.

\[ H_{2, 3} = \begin{bmatrix}
  c_0 & c_1 & c_2 \\
  c_1 & c_2 & c_3 \\
  c_2 & c_3 & c_4
\end{bmatrix} \]

and
If the $c_v$ in $H_{p,q}$ are taken as the $c_v(b)$ of the asymptotic expansion then the error for rational extrapolation is given by Gragg in the following theorem.

**Theorem 4.2 (Gragg [1965])**

Let $y(b;h)$ satisfy the condition of theorem 4.1 and in addition let $H_{q-1,q}$ and $H_{q,q} \neq 0$ for $q=1,2,\ldots$. If $h_0$ is sufficiently small, the $j$-th column of Bulirsch-Stoer scheme (4.3.21) exists and

$$r_1^{(j)} = y(b) + (-1)^{j}c_{j+1}(b)(h_1\cdots h_{1+j})^Y$$

as $i\to\infty$ where

$$c_{2j} = \frac{H_{j,j+1}}{H_{j-1,j}} \quad \text{and} \quad c_{2j+1} = \frac{H_{j+1,j+1}}{H_{j,j}}.$$

We have seen in the previous example how the GM formula compares well with the Trapezoidal formula when applied with extrapolation. In the next example we will see how it can still be improved by using rational extrapolation.

**Example 2**

We reconsider the problem of solving the IVP

$$y' = y, \quad y(0) = 1, \quad 0 \leq x \leq 1,$$
### Table 4.16: Rational extrapolation methods by using Euler, Trapezoidal and GM formulae

<table>
<thead>
<tr>
<th>h</th>
<th>Euler's method (s=5)</th>
<th>Trapezoidal method (s=3)</th>
<th>GM method (s=3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0000000</td>
<td>0.20000000E+01</td>
<td>0.25714286E+01</td>
<td>0.2180340E+01</td>
</tr>
<tr>
<td>0.5000000</td>
<td>0.22500000E+01</td>
<td>0.26684061E+01</td>
<td>0.27160624E+01</td>
</tr>
<tr>
<td>0.2500000</td>
<td>0.24414063E+01</td>
<td>0.27035160E+01</td>
<td>0.27181262E+01</td>
</tr>
<tr>
<td>0.1250000</td>
<td>0.25657845E+01</td>
<td>0.27142469E+01</td>
<td>0.27182758E+01</td>
</tr>
<tr>
<td>0.0625000</td>
<td>0.26379285E+01</td>
<td>0.27172260E+01</td>
<td>0.27182817E+01</td>
</tr>
<tr>
<td>0.0312500</td>
<td>0.26769901E+01</td>
<td>0.27182714E+01</td>
<td>0.27182818E+01</td>
</tr>
</tbody>
</table>

(a) Euler's method (s=5)

(b) Trapezoidal method (s=3)

(c) GM method (s=3)
as given in Example 1. This time, the problem is solved by using the three methods but with rational extrapolation given by formula (4.3.22). We use the basic step size and the sequence of \((h_i)\) as given in Example 1.

For \(x = h_0 = 1.0\), the numerical values obtained by these methods are given in Table 4.16. Convergence of the diagonal values for the Euler method correct to the eight significant figure value \(e = 2.7182818\) is attained only for \(s = 5\) as compared to \(s = 7\) in Example 1. For the Trapezoidal and the GM methods similar accuracy is achieved only with \(s = 3\) as compared to \(s = 4\) in Example 1. The values of \(R_n\) for this computation are given in Table 4.17.

<table>
<thead>
<tr>
<th>(n)</th>
<th>(R_n) Euler</th>
<th>(R_n) Trapezoidal</th>
<th>(R_n) GM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.2044</td>
<td>0.0264</td>
<td>0.0221</td>
</tr>
<tr>
<td>1</td>
<td>0.0210</td>
<td>0.0021</td>
<td>0.0027</td>
</tr>
<tr>
<td>2</td>
<td>0.0409</td>
<td>0.0026</td>
<td>0.0034</td>
</tr>
</tbody>
</table>

In this table we observe that the rate of convergence of the diagonal values of Table 4.16 to the exact solution is fastest in the earlier stage by using the GM method. Although in the final stages the rates are slower than the Trapezoidal formula, the overall convergence is still faster in the case of the GM formula as the values in Tables 4.16(b) and 4.16(c) show. This is due to the fact that the first tabular value \(p_0^{(0)} = 2.6180340\) in the GM formula is already closer to the exact value \(e = 2.7182818\) than \(p_0^{(0)} = 3.0000000\) in the Trapezoidal method. Therefore, a drastic change is not expected to occur again for further improvements.

Furthermore, as indicated by Schryer [1974], for a given function and initial approximation \(p_1^{(0)}, \ldots, p_{i+n}^{(0)}\), the best approximation to \(y(b)\)
may not be the furthest column $p_n^{(1)}$. In fact he gives a method of testing for the best approximation in the tableau, but we will not discuss it here.
4.4 GM FORMULA FOR SOLVING DELAY DIFFERENTIAL EQUATIONS

4.4.1 Introduction

Many physical systems possess the feature of having a delay response to input conditions, so that the rate at which processes occur depends not only on the current state of the system but also on the past states. Mathematical models of such processes commonly result in differential equations with a term involving a time delay. Equations of this type are called Delay-Differential Equations (DDE).

A general form of a first order DDE is

\[
\frac{du(t)}{dt} = f(t, u(t), u(t-d(t, u(t))))
\]  

(4.4.1)

where \( d(t, u(t)) \geq 0 \) is referred to as the "Delay" and \( t-d(t, u(t)) \) is referred to as the "Lag". In general the delay is a function both of \( t \) and of the solution \( u(t) \).

DDE's arise in a wide variety of topics, from the growth of infection to a model of a heart-lung complex, reaction to X-ray treatment, and population growth. DDE's also arise in control theory, machine tooling, and analytic number theory. A more detailed study of the applications of DDE's can also be found in Schmitt [1972] and Halanay [1966].

The initial conditions needed to determine the solution of the DDE (4.4.1) for \( t > t_0 \), generally involve prescribed values of \( u(t) \) on an appropriate initial set, say an interval of the form \( [a, t_0] \). The precise nature of these initial conditions depend on the nature of the delay and the range of values of the argument \( t \). If the solution of the problem is defined by an initial value at a single point, we shall refer to the problem as an initial value DDE, otherwise we shall call
it an initial function DDE. In general if we seek the solution of (4.4.1) for $t_0 < t < T$, we require an initial function of the form

$$u(t) = \Psi(t), \quad \min(t_0, t^*) \leq t \leq t_0$$

where $t^* = \min[t-d(t,u(t))]$, $t \in [t_0,t]$. One then tries to extend $u(t)$ continuously so as to satisfy the differential equation for $t > t_0$.

A more general DDE than (4.4.1) is one with multiple delays of the form

$$\frac{d}{dt} u(t) = f(t, u(t), u(t-d_1(t,u(t))), \ldots, u(t-d_q(t,u(t)))) \quad (4.4.2)$$

where $d_i(t,u(t)) \geq 0$ for $i = 1,2,\ldots,q$.

If the derivative at the present time is a function of the solution at the present time and the solution at some future time, then we have an Advanced-Differential Equation of the form

$$\frac{d}{dt} u(t) = f(t, u(t), u(t+d(t,u(t)))), \quad (4.4.3)$$

$$d(t,u(t)) \geq 0.$$  

Also if the derivative depends on the solution at the present time and both past and future times then we have a Neutral-Differential Equation of the form

$$\frac{d}{dt} u(t) = f(t, u(t), u(t-d_1(t,u(t))), u(t+d_2(t,u(t)))) \quad (4.4.4)$$
where \( d_1(t,u(t)), d_2(t,u(t)) \geq 0 \).

As in ordinary differential equations, the DDE can of course be similarly extended to a system.

### 4.4.2 Trapezoidal and GM methods for solving Delay Differential Equations

Consider the DDE of the type,

\[
u'(t) = f[t, u(t), u(t-d(t, u(t)))); \quad t_0 < t \leq T.
\]

First, we consider the solution by using the Trapezoidal and the GM methods with the delay term being approximated by linear interpolation. Suppose the solution at \( t_n \) has been found. We want to obtain the solution at \( t_{n+1} = t_n + h \) by using the Trapezoidal and the GM formulae. The delay term \( u(t-d) \) has to be evaluated first at point \( t_{n+1} \). We consider two cases.

(a) Small delay

For small delay, i.e. \( d \leq h \), the interpolation process uses the values of \( u(t_n) \) and \( u(t_{n+1}) \). This is demonstrated in Figure 4.6. Let

\[ s = t - d, \quad s_n = t_n - d, \]

and the approximate value of \( u(s_n) \) found by using linear interpolation is given by \( z(s_n) \), then we have,

\[ \frac{u_{n+1} - u_n}{t_{n+1} - t_n} = \frac{z_{n+1} - u_n}{t_{n+1} - d - t_n} \]

i.e.

\[ z_{n+1} = (d/h)u_n + (1-d/h)u_{n+1}. \tag{4.4.6} \]

Given the initial values \( u_0 \) and \( z_0 \), the subsequent values are then calculated by the following predictor-corrector algorithm:

(i) Predict \( u_1 \) by using a predictor formula,

(ii) Compute \( z_1 \), i.e.

\[ z_1 = (d/h)u_0 + (1-d/h)u_1 \]
(iii) Improve $u_1$ by either

**Trapezoidal:** $u_1 = u_0 + \frac{1}{2}h[f(t_0, u_0, z_0) + f(t_1, u_1, z_1)]$, or

**GM:** $u_1 = u_0 + h \sqrt{f(t_0, u_0, z_0) f(t_1, u_1, z_1)}$

(iv) Repeat improvements in P(EC)$^m$E mode or test for convergence by using a preset error tolerance, $\varepsilon$, i.e.

$$|u_1^{(i+1)} - u_1^{(i)}| \geq \varepsilon, \text{ go to step (ii)}$$

$$|u_1^{(i+1)} - u_1^{(i)}| < \varepsilon, \text{ go to next step, i.e.}$$

$$u_0 \leftarrow u_1$$

$$z_0 \leftarrow z_1$$

(for constant delay).
Numerical Example

Consider the equation

\[ u' = au(t) + bu(t-d) \]

\[ a = -2.1, b = 1, d = \ln(1.1), \]

which has the exact solution

\[ u(t) = \exp(-t). \]

This is an example of a linear DDE with constant delay and has a smooth solution. The problem is solved numerically by using the Trapezoidal and GM integration formulae with \( h = 0.1 \). Since in this case \( d < h \) we have a problem with small delay. We use the "DDES" program and obtain the solutions for \( 0 \leq t \leq 10 \) as shown in Table 4.18. In this table the solutions are printed at every ten steps \( (k=10) \).

<table>
<thead>
<tr>
<th>t</th>
<th>Exact Solution</th>
<th>Numerical Solution</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.1000000E+01</td>
<td>0.367676E+00</td>
<td>0.203244E-03</td>
</tr>
<tr>
<td>1.0</td>
<td>0.367879E+00</td>
<td>0.367676E+00</td>
<td>0.147588E-03</td>
</tr>
<tr>
<td>2.0</td>
<td>0.135335E+00</td>
<td>0.135187E+00</td>
<td>0.810450E-04</td>
</tr>
<tr>
<td>3.0</td>
<td>0.497870E-01</td>
<td>0.497060E-01</td>
<td>0.396503E-04</td>
</tr>
<tr>
<td>4.0</td>
<td>0.183156E-01</td>
<td>0.182759E-01</td>
<td>0.182373E-04</td>
</tr>
<tr>
<td>5.0</td>
<td>0.673794E-02</td>
<td>0.671970E-02</td>
<td>0.805279E-05</td>
</tr>
<tr>
<td>6.0</td>
<td>0.247875E-02</td>
<td>0.24769E-02</td>
<td>0.342441E-05</td>
</tr>
<tr>
<td>7.0</td>
<td>0.911881E-03</td>
<td>0.908457E-03</td>
<td>0.142203E-05</td>
</tr>
<tr>
<td>8.0</td>
<td>0.335462E-03</td>
<td>0.334040E-03</td>
<td>0.611109E-06</td>
</tr>
<tr>
<td>9.0</td>
<td>0.123409E-04</td>
<td>0.122798E-03</td>
<td>0.272269E-06</td>
</tr>
<tr>
<td>10.0</td>
<td>0.453998E-04</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

continued...
(b) GM formula

<table>
<thead>
<tr>
<th>t</th>
<th>Exact Solution</th>
<th>Numerical Solution</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.100000E+01</td>
<td>0.368094E+00</td>
<td>-0.214967E-03</td>
</tr>
<tr>
<td>1.0</td>
<td>0.367879E+00</td>
<td>0.368094E+00</td>
<td>-0.214967E-03</td>
</tr>
<tr>
<td>2.0</td>
<td>0.135335E+00</td>
<td>0.135494E+00</td>
<td>-0.159278E-03</td>
</tr>
<tr>
<td>3.0</td>
<td>0.497870E-01</td>
<td>0.498752E-01</td>
<td>-0.881571E-04</td>
</tr>
<tr>
<td>4.0</td>
<td>0.183156E-01</td>
<td>0.183589E-01</td>
<td>-0.433125E-04</td>
</tr>
<tr>
<td>5.0</td>
<td>0.673794E-02</td>
<td>0.675785E-02</td>
<td>-0.190050E-04</td>
</tr>
<tr>
<td>6.0</td>
<td>0.347875E-02</td>
<td>0.248753E-02</td>
<td>-0.877899E-05</td>
</tr>
<tr>
<td>7.0</td>
<td>0.911881E-03</td>
<td>0.915679E-03</td>
<td>-0.379815E-05</td>
</tr>
<tr>
<td>8.0</td>
<td>0.335462E-03</td>
<td>0.337080E-03</td>
<td>-0.161816E-05</td>
</tr>
<tr>
<td>9.0</td>
<td>0.123409E-03</td>
<td>0.124061E-03</td>
<td>-0.651927E-06</td>
</tr>
<tr>
<td>10.0</td>
<td>0.453998E-04</td>
<td>0.456400E-04</td>
<td>-0.240195E-06</td>
</tr>
</tbody>
</table>

(b) Large delay

Unlike in the small delay case, the delay term has to be evaluated outside the interval of the present consideration \([t_n, t_{n+1}]\) when the delay is large, i.e. \(d > h\). This is demonstrated in Figure 4.7.

![Figure 4.7: Large Delay](attachment:figure.png)
In this case let the delay term be calculated at the point

\[ s = t_{n+1} - d \]

where \( t_j \leq s < t_{j+1} \) for some \( j < n \).

We may write

\[ s = t_j + rh_j, \quad h_j = t_{j+1} - t_j, \]

so that

\[ r = (s - t_j) / h_j. \]

A linear interpolation yields

\[ z(s) = (1-r)u_j + ru_{j+1}, \]

where the local truncation error of the approximation, \( e(s) \), is obtained as follows,

\[ e(s) = u(s) - z(s) = u(s) - (1-r)u_j - ru_{j+1}. \]

Expanding \( u_j \) and \( u_{j+1} \) about \( s \) and writing \( u(s) = u_s \) we have

\[
\begin{align*}
    r h & \quad (1-r)h \\
    t_j & \quad s \quad t_{j+1}
\end{align*}
\]

\[ u_j = u_s + (-rh)u_s' + \frac{1}{2} (rh)^2 u_s'' + \ldots \]

\[ u_{j+1} = u_s + (1-r)hu_s' + \frac{1}{2} (1-r)^2 h^2 u_s'' + \ldots . \]

Therefore,

\[
e(s) = u_s - (1-r)[u_s - rh u_s' + \frac{1}{2} h^2 u_s'' + \ldots ]
- r[u_s + (1-r)hu_s' + \frac{1}{2} (1-r)^2 h^2 u_s'' + \ldots ]
= \frac{1}{2} r(r-1)h^2 u_s'' + O(h^3)
= \frac{1}{2} r(r-1)h^2 u''(\xi_j), \quad t_j < \xi_j < t_{j+1}. \quad (4.4.7)
\]

Since \( s \) is no longer within \([t_n, t_{n+1}]\), it is therefore necessary to store all the previous solution values for the purpose of computing the delay terms, especially when the delay is arbitrary.
For a problem with constant delay we could immediately overwrite the last used values with new solution values at an expense of keeping up a record of the step difference for identification. For example, when \( n=5, j=0 \), we have to compute \( u_6 \) by using \( u_0 \) and \( u_1 \). Once \( u_6 \) is obtained, we can then store \( u_6 \) in \( u_0 \), etc. Alternatively we can compute \( z_6 \) immediately after we obtain \( u_1 \).

For a problem with arbitrary delay the computer program is complicated by the fact that we have to look for the interval over which the term \( s = t_{n+1} - d_{n+1} \) falls.

Another difficulty that we may encounter in the large delay case is the ability to supply the off-the-range values at the early stages which in many cases become excessive as in the case of using small step size. This is demonstrated in Figure 4.8 where \( z_0, z_1 \) and \( z_2 \) have to be supplied because \( s_0, s_1, s_2 < t_0 \).

![Figure 4.8](image_url)

In general, these values are determined by experiment. This is probably the problem with any numerical method for solving DDE's. If instead of evaluating the solution for \( u(t) \) we solve for \( u(t-d) \) we could probably avoid supplying the back stage values, but we are
confronted with the problem of implicitness over several steps, (i.e. similar to a problem with negative delay).

**Numerical Example**

Consider the equation

\[ u_t = au(t) + bu(t-d) \]

with \( a = -10, \ b = 1, \ d = \ln 9, \)

which has the exact solution

\[ u(t) = \exp(-t). \]

In this example we have to supply the values of \( z \) for \( t < \ln 9 \)

since for these values of \( t \) we have \( t-d < 0 \) which is outside the range of integration. We have in this case used the exact solution in place of the unknown values of \( z. \)

We use the computer program "DDEL" for solving this problem. The results show that the GM formula produces more accurate results at the earlier stage of integration, but as the integration goes further the Trapezoidal formula becomes more accurate. This example shows that despite its increased complexity in the computation, the GM formula can be as competitive as the Trapezoidal formula, and the condition under which it can perform more accurately than the Trapezoidal formula is given in the analysis presented in Section 4.1.5.

**Table 4.19 - Solutions for large delay**

\( h = 0.1, \ k = 10 \)

(a) Trapezoidal formula

<table>
<thead>
<tr>
<th>( t )</th>
<th>Exact Solution</th>
<th>Numerical Solution</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.100000E+01</td>
<td>0.367845E+00</td>
<td>0.340566E-04</td>
</tr>
<tr>
<td>1.0</td>
<td>0.367879E+00</td>
<td>0.135322E+00</td>
<td>0.124815E-04</td>
</tr>
<tr>
<td>2.0</td>
<td>0.135335E+00</td>
<td>0.497844E-01</td>
<td>0.265892E-05</td>
</tr>
<tr>
<td>3.0</td>
<td>0.497870E-01</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

continued...
4.4.3 AM and GM formulae with Hermite Interpolation

We will now use Hermite interpolation based on the values of the solution and its derivatives at two points $t_n$ and $t_{n+1}$. We will discuss only for the case of small delay since the discussion for large delay is similar.

**Small delay case**

Let $s = t_n + rh$, $0 < r < 1$, $t_n < s < t_{n+1}$.

Assume that the solution and its derivative are known at the points $t_n$ and $t_{n+1}$, then the Hermite interpolation yields

$$z(s) = p_1 u_n + p_2 u_{n+1} + p_3 h u'_n + p_4 h u'_{n+1}$$ (4.4.8)

where

$$p_1 = (1-r)^2(1+2r)$$
\[ p_2 = r^2(3-2r) \]
\[ p_3 = r(1-r)^2 \]
\[ p_4 = -r^2(1-r) \]

We use the trapezoidal formula to express \( u_{n+1} \) in terms of \( u_n \), \( u_{n+1} \) and \( u_n' \), i.e., from

\[ u_{n+1} = u_n + \frac{1}{2}h(u_n'u_n + u_n') \]

we have

\[ u_{n+1}' = -\frac{2}{h}u_n + \frac{2}{h}u_{n+1} - u_n' \quad (4.4.9) \]

Substituting (4.4.9) in (4.4.8) produces

\[ z(s) = (p_1 - 2p_4)u_n + (p_2 + 2p_4)u_{n+1} + h(p_3 - p_4)u_n' \quad (4.4.10) \]

The truncation error of this approximation is obtained in terms of \( u(s) \) as follows,

\[ \begin{array}{c}
\text{rh} \\
\hline
\hline
\text{tn} & s & t_{n+1} \\
\hline
\end{array} \]

\[ u_n = u_s - rhu_s' + \frac{1}{6}r^2h^2u_s'' - \frac{1}{6}r^3h^3u_s''' + \ldots \]
\[ u_n' = u_s' - rhu_s'' + \frac{1}{6}r^2h^2u_s''' + \ldots \]
\[ u_{n+1} = u_s + (1-r)hu_s' + \frac{1}{6}(1-r)^2h^2u_s'' + \frac{1}{6}(1-r)^3h^3u_s''' + \ldots \]

Substituting this into (4.4.10) and writing \( p_4 \) in terms of \( r \) we obtain
\[ z(s) = \left[ (1-r)^2(1+2r)+2r^2(1-r) \right] [u_s'\!-\!rhu_s + \frac{1}{6}r^2h^2u_s''] \\
- \frac{1}{6}r^2h^2u_s'' + \ldots \right] + \left[ r^2(3\!-\!2r\!-\!2r^2(1\!-\!r)) \right] [u_s + (1\!-\!r)hu_s' \\
+ \frac{1}{6}r^2h^2u_s'' + \frac{1}{6}h^3(1\!-\!r)^3u_s'' + \ldots ] \\
+ h [r(1\!-\!r)^2+r^2(1\!-\!r)] [u_s'\!-\!rhu_s + \frac{1}{6}r^2h^2u_s''] + \ldots \right] \\
= u_s + \frac{1}{6}r^2(1\!-\!r)h^3u_s'' + \ldots \quad (4.4.11) \]

Therefore, the truncation error \( e(s) \) is given by
\[
u(s) = z(s) - u(s) \\
= \frac{1}{6}r^2(1\!-\!r)h^3u''(z_n) \\
t_n < s < t_{n+1}
\]

**Computational Algorithm**

Given the initial values \( u_0 \) and \( z_0 \), the subsequent values are then calculated by the following steps,

(i) predict \( u_1 \) by using a predictor formula;

(ii) compute \( z_1 \) by 2-point Hermite interpolation at \( t_0 \) and \( t_1 \), i.e.,
\[
z_1 = (p_1-2p_4)u_0 + (p_2+2p_4)u_1 + h(p_3-p_4)u_0' \\
\]
where
\[
u_0' = f(t_0,u_0,z_0) \\
\]

(iii) improve \( u_1 \) by either
\[
\text{AM: } u_1 = u_0 + \frac{1}{2}h[f(t_0,u_0,z_0)+f(t_1,u_1,z_1)] \\
\text{or } \text{GM: } u_1 = u_0 + \frac{h}{2} \left[ f(t_0,u_0,z_0)f(t_1,u_1,z_1) \right] \\
\]

(iv) Test for convergence:

If \(| \text{difference} | > \varepsilon\), go to step (i)

If \(| \text{difference} | < \varepsilon\), \( u_0 = u_1 \)
\[
z_0 = z_1 \\
t_0 = t_1 \\
go to step (i)
Numerical Example

Consider the equation

\[ u' = au(t) + bu(t-d) \]

\[ a = -2.1, \ b = 1, \ d = \ln(1.1), \]

which has the exact solution

\[ u(t) = \exp(-t). \]

The numerical solutions obtained by using the AM and GM formulae are shown in Table 4.20. Note that the GM formula produces more accurate results at the early stage of the integration, in particular for \( x \leq 10 \). For \( x > 10 \), the AM formula seems to perform slightly better than the GM formula.

We also note that there is no significant improvement of accuracy in using a higher interpolating polynomial for approximating the delay term. This is due to the fact that the method used for obtaining the full-step solution values is only 2nd order accurate. Therefore in practice any 1st order interpolating polynomial will suffice to serve the purpose.

Table 4.20: Solutions of small delay problem with Hermite interpolation

(a) AM formula:

\( h = 0.1, \ k = 10. \)

<table>
<thead>
<tr>
<th>t</th>
<th>Exact Solution</th>
<th>Numerical Solution</th>
<th>Error</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.10000000E+00</td>
<td>0.36760150E+00</td>
<td>0.27794391E-03</td>
<td>0.75552988E-01</td>
</tr>
<tr>
<td>1.0</td>
<td>0.36787944E+00</td>
<td>0.13513140E+00</td>
<td>0.20388886E-03</td>
<td>0.15065462E+00</td>
</tr>
<tr>
<td>2.0</td>
<td>0.49787074E-01</td>
<td>0.18260593E-01</td>
<td>0.55049546E-04</td>
<td>0.30056028E+00</td>
</tr>
<tr>
<td>3.0</td>
<td>0.18315642E-01</td>
<td>0.67126162E-02</td>
<td>0.25330344E-04</td>
<td>0.37593566E+00</td>
</tr>
<tr>
<td>4.0</td>
<td>0.67379466E-02</td>
<td>0.24675628E-02</td>
<td>0.11886474E-04</td>
<td>0.45138239E+00</td>
</tr>
<tr>
<td>5.0</td>
<td>0.91188148E-03</td>
<td>0.33347653E-03</td>
<td>0.19858562E-05</td>
<td>0.59197582E+00</td>
</tr>
<tr>
<td>6.0</td>
<td>0.12340969E-03</td>
<td>0.67379466E-02</td>
<td>0.83833038E-06</td>
<td>0.67930678E+00</td>
</tr>
</tbody>
</table>

continued...
155

<table>
<thead>
<tr>
<th>( t )</th>
<th>( t ) Exact Solution</th>
<th>( t ) Numerical Solution</th>
<th>Error</th>
<th>% Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.10000000E+00</td>
<td>0.13848022E-03</td>
<td>-0.37642826E-01</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>0.36787944E+00</td>
<td>0.80921624E-04</td>
<td>-0.11266411E+00</td>
<td></td>
</tr>
<tr>
<td>2.0</td>
<td>0.13533529E+00</td>
<td>0.35471903E-03</td>
<td>-0.27518487E-03</td>
<td></td>
</tr>
<tr>
<td>3.0</td>
<td>0.49787074E+01</td>
<td>0.12618664E-04</td>
<td>-0.18727759E+00</td>
<td></td>
</tr>
<tr>
<td>4.0</td>
<td>0.18315642E-01</td>
<td>0.12618664E-04</td>
<td>-0.18727759E+00</td>
<td></td>
</tr>
<tr>
<td>5.0</td>
<td>0.67379466E-02</td>
<td>0.12618664E-04</td>
<td>-0.18727759E+00</td>
<td></td>
</tr>
<tr>
<td>6.0</td>
<td>0.24877571E-02</td>
<td>0.12618664E-04</td>
<td>-0.18727759E+00</td>
<td></td>
</tr>
<tr>
<td>7.0</td>
<td>0.91188148E-03</td>
<td>0.12618664E-04</td>
<td>-0.18727759E+00</td>
<td></td>
</tr>
<tr>
<td>8.0</td>
<td>0.33646239E-03</td>
<td>0.12618664E-04</td>
<td>-0.18727759E+00</td>
<td></td>
</tr>
<tr>
<td>9.0</td>
<td>0.12340969E-03</td>
<td>0.12618664E-04</td>
<td>-0.18727759E+00</td>
<td></td>
</tr>
<tr>
<td>10.0</td>
<td>0.45399875E-04</td>
<td>0.12618664E-04</td>
<td>-0.18727759E+00</td>
<td></td>
</tr>
<tr>
<td>11.0</td>
<td>0.16701677E-04</td>
<td>0.12618664E-04</td>
<td>-0.18727759E+00</td>
<td></td>
</tr>
<tr>
<td>12.0</td>
<td>0.61442021E-05</td>
<td>0.12618664E-04</td>
<td>-0.18727759E+00</td>
<td></td>
</tr>
<tr>
<td>13.0</td>
<td>0.22603251E-05</td>
<td>0.12618664E-04</td>
<td>-0.18727759E+00</td>
<td></td>
</tr>
<tr>
<td>14.0</td>
<td>0.83152693E-06</td>
<td>0.12618664E-04</td>
<td>-0.18727759E+00</td>
<td></td>
</tr>
<tr>
<td>15.0</td>
<td>0.30590159E-06</td>
<td>0.12618664E-04</td>
<td>-0.18727759E+00</td>
<td></td>
</tr>
<tr>
<td>16.0</td>
<td>0.11253488E-06</td>
<td>0.12618664E-04</td>
<td>-0.18727759E+00</td>
<td></td>
</tr>
<tr>
<td>17.0</td>
<td>0.41399308E-07</td>
<td>0.12618664E-04</td>
<td>-0.18727759E+00</td>
<td></td>
</tr>
<tr>
<td>18.0</td>
<td>0.15229969E-07</td>
<td>0.12618664E-04</td>
<td>-0.18727759E+00</td>
<td></td>
</tr>
<tr>
<td>19.0</td>
<td>0.56027978E-08</td>
<td>0.12618664E-04</td>
<td>-0.18727759E+00</td>
<td></td>
</tr>
<tr>
<td>20.0</td>
<td>0.20611561E-08</td>
<td>0.12618664E-04</td>
<td>-0.18727759E+00</td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER FIVE

APPLICATION OF THE GM STRATEGY IN THE RUNGE-KUTTA TYPE FORMULAE
5.1 GM RUNGE-KUTTA FORMULAE

5.1.1 Introduction

In view of the close relation between the weighted arithmetic mean and the weighted geometric mean we may extend the idea of the classical Runge-Kutta procedure to the equivalent GM formula. In particular we observe that the single application of the corrector in the Heun's second order formula is equivalent to the application of the second order classical Runge-Kutta method given by,

\[
\begin{align*}
    k_1 &= f(x_n, y_n) \\
    k_2 &= f(x_n + h, y_n + hk_1) \\
    y_{n+1} &= y_n + \frac{1}{2}h(k_1 + k_2)
\end{align*}
\]

(5.1.1)

Similarly as we have seen in Section 4.2.1, the single application of the corrector in the Euler-GM predictor-corrector pair could be regarded as forming a GM equivalent to formula (5.1.1) in the form,

\[
\begin{align*}
    k_1 &= f(x_n, y_n) \\
    k_2 &= f(x_n + h, y_n + hk_1) \\
    y_{n+1} &= y_n + h\sqrt{k_1 k_2}
\end{align*}
\]

(5.1.2)

which is of second order with local truncation error given by (4.2.9).

Generalising this idea to the GM formula of the Runge-Kutta type of any order, we may now adopt the equivalent definition of the symbol generating matrix in the Runge-Kutta process for use in the geometric sense. The applicability of this adoption will eventually be examined in the next section.
Definition 5.1

Let $A$ be a $(R+1) \times R$-matrix. The $R$-stage GM version of the Runge-Kutta procedure with generating matrix $A$ is the one-step $(R+1)$-stage forward step procedure defined by the relations,

\[(a)\]  
\[k_i = f(y_n + h \sum_{j=1}^{R} a_{ij} k_j),\]

\[i = 1, 2, \ldots, R\]

\[(b)\]  
\[y_{n+1} = y_n + h \prod_{i=1}^{R} k_i,\]

where \[w_i = \frac{a_{R+1,i}}{\sum_i a_{R+1,i}}\]

5.1.2 Derivation of the 1st to 4th order GM Runge-Kutta formulae

The consistency of the definition 5.1 to the development of the GM Runge-Kutta formula of various order may be first checked by developing the methods of lower order. A natural starting point is to produce a method of the first order. As in the case of the classical Runge-Kutta method, first order formula in the class should be able to be achieved by using only a single function evaluation. Thus, the formula is identical to the Euler's formula, i.e.

\[k_1 = f(x_n, y_n)\]
\[y_{n+1} = y_n + h k_1.\]  
\[ (5.1.4)\]

2-stage GM Runge-Kutta formulae

The general 2-stage GM Runge-Kutta formulae take the following form,
\[ \begin{align*}
k_1 &= f(y) \\
k_2 &= f(y + h a_1 k_1) \\
y_{n+1} &= y_n + h(k_1 w_1 k_2 w_2).
\end{align*} \] (5.1.5)

Since \( w_1 + w_2 = 1 \), we let for simplicity, \( w_2 = p \) and \( w_1 = 1 - p \). Expanding \( k_2 \) as a Taylor series about \( y \) we have,

\[ k_2 = f + h a_1 \frac{\partial f}{\partial y} + \frac{1}{2} h^2 a_1 \frac{\partial^2 f}{\partial y^2} + \ldots \]

Therefore,

\[ k_2^P = f^P \{ 1 + h p a_1 \frac{\partial f}{\partial y} + h^2 (4 p a_1 \frac{\partial^2 f}{\partial y^2} + \ldots \} \]

and

\[ k_1^{1-p} k_2^P = f^{1-p} f^P \{ 1 + h p a_1 \frac{\partial f}{\partial y} + h^2 (4 p a_1 \frac{\partial^2 f}{\partial y^2} + \ldots \} \]

\[ = f + h p a_1 \frac{\partial f}{\partial y} + h^2 (4 p a_1 \frac{\partial^2 f}{\partial y^2} + \ldots \} \]

Hence,

\[ y_{n+1} = y_n + h k_1^{1-p} k_2^P \]

\[ = y_n + h f + h^2 p a_1 \frac{\partial f}{\partial y} + h^3 (4 p a_1 \frac{\partial^2 f}{\partial y^2} + \ldots \} \]

Comparing term by term with the Taylor series expansion for \( y_{n+1} \) about \( y_n \), i.e.

\[ y_{n+1} = y_n + h \frac{\partial f}{\partial y} + h^2 \frac{\partial^2 f}{\partial y^2} + \frac{1}{6} h^3 \frac{\partial^3 f}{\partial y^3} + \ldots \] (5.1.6)

we have the following three equations of conditions,

\[ \frac{\partial^2 f}{\partial y^2} : \quad p a_1 = \frac{1}{2} \] (5.1.8-i)

\[ \frac{\partial^2 f}{\partial y^2} : \quad 4 p a_1 = 1/6 \] (5.1.8-ii)

\[ \frac{\partial^2 f}{\partial y^2} : \quad 4 p (p-1) a_1 = 1/6. \] (5.1.8-iii)
Now an order three formula cannot be achieved since we have three equations in two unknowns. To obtain a method of order 2 we must choose $p$ and $a_1$ to satisfy (5.1.8-i). Equations (5.1.8-ii) and (5.1.8-iii) then determine the principal term in the local truncation error of the method. Since there are infinitely many possible solutions for equation (5.1.8-i), we list in Table 5.1, three sets of the solution together with the local truncation errors.

**Table 5.1: Three set of parameters that produce 2nd order GM Runge-Kutta formulae**

<table>
<thead>
<tr>
<th></th>
<th>set 1</th>
<th>set 2</th>
<th>set 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>1</td>
<td>2/3</td>
<td>-1/6</td>
</tr>
<tr>
<td>$p$</td>
<td>1/2</td>
<td>3/4</td>
<td>-3</td>
</tr>
<tr>
<td>LTE</td>
<td>$-\frac{1}{12}h^3f_yf_{yy}$</td>
<td>$\frac{5}{24}h^3ff_y$</td>
<td>$\frac{5}{24}h^3f^2f_{yy}$</td>
</tr>
<tr>
<td></td>
<td>$+\frac{7}{24}h^3ff_y^2$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The first set of solution produces the following method,

\[
\begin{align*}
    k_1 &= f(x_n, y_n) \\
    k_2 &= f(x_n + h, y_n + hk_1) \\
    y_{n+1} &= y_n + h\sqrt{k_1k_2}
\end{align*}
\]

\hspace{8cm} (5.1.9)

i.e. the GM equivalent of the Heun's formula given by equations (5.1.2). We will use this method to represent the 2-stage GM Runge-Kutta methods in our future analysis (i.e. in the study of stability, etc.) because in practice this method performs better than the others due to the cancellation of some of the error terms in most problems.
The general 3-stage explicit GM Runge-Kutta method is given by the relations,

\[
\begin{align*}
  k_1 &= f(y_n) \\
  k_2 &= f(y_n + h a_1 k_1) \\
  k_3 &= f(y_n + h a_2 k_1 + h a_3 k_2) \\
  y_{n+1} &= y_n + h (w_1 k_1 + w_2 k_2 + w_3 k_3).
\end{align*}
\]

(5.1.10)

In this relation the parameters \(a_1, a_2, a_3\) and \(w_1, w_2, w_3\) are to be determined so that the method will have as high an order of accuracy as possible. We will expand the formula about \(y_n\) and then compare with the Taylor series expansion of \(y(x_{n+1})\) about \(x_n\).

\[
\begin{align*}
  k_1 &= f(y_n) = f \\
  k_2 &= f(y_n + h a_1 k_1) \\
  &= f(y_n) + h a_1 f'(y_n) + \frac{h^2}{2!} a_1^2 f''(y_n) + \frac{h^3}{3!} a_1^3 f'''(y_n) + \ldots \\
  &= f + h a_1 f + \frac{h^2}{2} a_1^2 f_y + \frac{h^3}{6} a_1^3 f_yy + \ldots \\
  k_3 &= f(y_n + h(a_2 k_1 + a_3 k_2)) \\
  &= f(y_n) + h(a_2 k_1 + a_3 k_2) f'(y_n) + \frac{h^2}{2!} (a_2 k_1 + a_3 k_2)^2 f''(y_n) + \frac{h^3}{3!} (a_2 k_1 + a_3 k_2)^3 f'''(y_n) + \ldots \\
  &= f + h a_2 f + \frac{h^2}{2} a_2^2 f_y + \frac{h^3}{6} a_2^3 f_yy + \ldots
\end{align*}
\]

(5.1.11)

But,

\[
\begin{align*}
  (a_2 k_1 + a_3 k_2) &= a_2 f + a_3 (f + h a_1 f_y + \frac{h^2}{2} a_1^2 f_{yy} + \frac{h^3}{6} a_1^3 f_{yyy} + \ldots) \\
  &= a_2 f + a_3 f + a_1 f_y + \frac{h^2}{2} a_1^2 f_{yy} + \frac{h^3}{6} a_1^3 f_{yyy} + \ldots \\
  &= (a_2 + a_3) f + h a_1 f_y + \frac{h^2}{2} a_1^2 f_{yy} + \frac{h^3}{6} a_1^3 f_{yyy} + \ldots
\end{align*}
\]

(5.1.12)
\[(a_2 k_1 + a_3 k_2)^2 = \frac{a_2^2 + a_3^2}{2} + h(2a_1 a_3 (a_2 + a_3) f_y^2) + h^2\left[\frac{a_1^2 a_3^2 f_y^2}{2} + (a_2 + a_3) a_1^2 a_3 f_{yy}^2\right] + \ldots
\]

\[(a_2 k_1 + a_3 k_2)^3 = \frac{a_2^3 + a_3^3}{3} + h[3(a_2 + a_3)^2 a_1 a_3 f_y^3] + \ldots
\]

and

\[(a_2 k_1 + a_3 k_2)^4 = \frac{a_2^4 + a_3^4}{4} + \ldots
\]

Therefore,

\[
k_3 = f + h[(a_2 + a_3)f + \frac{a_2^2 a_3^2}{2} f_{yy} + \frac{1}{6} a_1^2 a_3^2 f_y^3 + \ldots] + h^2(\frac{a_2^2 + a_3^2}{2} + h^2[\frac{a_2^2 + a_3^2}{2} f_{yy} + \ldots]) + \frac{1}{6} h^3[(a_2 + a_3)^3 + 2a_1 a_3 f_y^3 + \ldots]
\]

\[
+ h[3(a_2 + a_3) a_1 a_3 f_y^3 + \ldots] f_{yy} + \frac{1}{24} h^4(a_2 + a_3)^4 f_{yyy} + \ldots
\]

\[= f + h(a_2 + a_3)f + h^2(\frac{a_2^2 + a_3^2}{2} f_{yy} + \frac{1}{6} a_1^2 a_3^2 f_y^3 + \ldots)
\]

Now, since \(w_1 + w_2 + w_3 = 1\) we set, for our convenience, \(w_2 = p\), \(w_3 = q\) and \(w_1 = 1 - p - q\). Taking the powers of \(k_1 = 1, 2, 3\), we have

\[
k_2^p = f^p\left[1 + (h a_1 f_y + h^2 a_1^2 f_{yy} + \frac{1}{6} a_1^3 f_y^3 + \ldots)\right]^p
\]

\[= f^p \left[1 + p(h a_1 f_y + h^2 a_1^2 f_{yy} + \frac{1}{6} a_1^3 f_y^3 + \ldots) + \frac{1}{6} p(p-1)(p-2) h^3 a_1^3 f_y^3 + \ldots\right]
\]

\[= f^p \left[1 + h(p a_1 f_y) + h^2(\frac{1}{6} p a_1^3 f_y^3 + p(p-1) a_1^2 f_y^2) + h^3(\frac{1}{6} p a_1^3 f_y^3 + \frac{1}{6} p(p-1) a_1^3 f_y^3) + \ldots\right]
\]

\[= f^p \left[1 + h(p a_1 f_y) + h^2(\frac{1}{6} p a_1^3 f_y^3 + p(p-1) a_1^2 f_y^2) + h^3(\frac{1}{6} p a_1^3 f_y^3 + \frac{1}{6} p(p-1) a_1^3 f_y^3) + \ldots\right]
\]
Similarly,

\[
k_3^q = f^q \left\{ 1 + h(a_2 + a_3)f_y + \frac{1}{2}[(a_2 + a_3)^2 f_{yy}]ight. \\
+ h^2[(a_2)^2 (a_2 + a_3)^2 f_{yy} + \frac{1}{6} (a_2 + a_3)^3 f_{yyy}] + \ldots \right\} \\
= f^q \left\{ 1 + q[h(a_2 + a_3)f_y + h^2(a_2 + a_3)^2 f_{yy}]ight. \\
+ h^3[(a_2)^2 (a_2 + a_3)^2 f_{yy} f_y + \frac{1}{6} (a_2 + a_3)^3 f_{yyy}] + \ldots \right\} \\
+ \frac{1}{6} q(q-1)(a_2 + a_3)^3 f_y \left[ f_{yy} + \ldots \right] \\
= f^q \left\{ 1 + h[q(a_2 + a_3)f_y] + h^2[(a_2 + a_3)^2 f_{yy}]ight. \\
+ h^3[q(a_2 + a_3)^2 f_{yy}] + \frac{1}{6} q(a_2 + a_3)^3 f_{yy} + \frac{1}{6} q(a_2 + a_3)^3 f_{yy} + \ldots \right\} \\
+ \frac{1}{6} q(q-1)(a_2 + a_3)^3 f_y \left[ f_{yy} + \ldots \right] \\
= f + h[p_1 + q(a_2 + a_3)]f_y + h^2[p_1^2 + q(a_2 + a_3)^2]f_{yy} \\
+ \{ pqa_1(a_2 + a_3) + q(a_2 + a_3)^2 f_{yy} \} \\
+ \ldots \\
(5.1.16)
\]

Now,

\[
k_1^{1-p-q} k_2 p_k^q = f \left\{ 1 + hpa_1 f_y + \frac{1}{2}[(a_2 + a_3)^2 f_{yy}]ight. \\
+ \ldots \right\} \left\{ 1 + h(q(a_2 + a_3)f_y) \\
+ h^2[q(a_2 + a_3)^2 f_{yy}]ight. \\
+ h^3[q(a_2 + a_3)^2 f_{yy}] + \ldots \right\} \\
(\text{collecting terms up to the second order}) \\
= f + h[p_1 + q(a_2 + a_3)]f_y + h^2[p_1^2 + q(a_2 + a_3)^2]f_{yy} \\
+ \{ pqa_1(a_2 + a_3) + q(a_2 + a_3)^2 f_{yy} \} \\
+ \ldots \\
(5.1.16)
\]

Therefore,

\[
y_{n+1} = y_n + h f + h^2[p_1 + q(a_2 + a_3)]f_y \\
+ h^3[p_1^2 + q(a_2 + a_3)^2]f_{yy} + (pqa_1(a_2 + a_3) \\
+ q(q-1)(a_2 + a_3)^2 f_{yy} \} + \ldots \\
(5.1.17)
\]
This last series is now compared with the Taylor's series expansion of y(x_n+h) about x_n in terms of the RHS function f, viz.

\[ y(x_n + h) = y(x_n) + hf + \frac{h^2}{2} f_y + \frac{1}{6} h^3 (f^2 f_y + ff_y^2) + \frac{1}{24} h^4 (f^3 f_{yyy} + 4f^2 f_y f_{yy} + ff_y^3) + \ldots \]

Term by term comparison reveals

\[ h^f_y: \quad pa_1 + q(a_2 + a_3) = \frac{1}{6} \]  
\[ (5.1.18-i) \]

\[ h^3 f^2_{yy}: \quad \frac{1}{4}a_1^2 + \frac{1}{4}q(a_2 + a_3)^2 = 1/6 \]  
\[ (5.1.18-ii) \]

\[ h^3 f^2_y: \quad pqa_1(a_2 + a_3) + \frac{1}{2}p(p-1)a_1^2 + qa_1a_3 \]
\[ + \frac{1}{4}q(q-1)(a_2 + a_3)^2 = 1/6 \]  
\[ (5.1.18-iii) \]

Thus, we have three equations for the determination of the five unknowns a_1, a_2, a_3, p and q. If we evaluate k_2 and k_3 at the mid-point and at the end-point of the interval (x_n , x_{n+1}) respectively then we are led to using two other equations,

\[ a_1 = \frac{1}{6} \]  
\[ (5.1.19) \]

and

\[ a_2 + a_3 = 1 \]  
\[ (5.1.20) \]

Substituting (5.1.19) and (5.1.20) into equations (5.1.18) produce

\[ \begin{align*}
\frac{1}{6}p + q &= \frac{1}{6} \\
\frac{1}{8}p + \frac{1}{4}q &= 1/6 \\
\frac{1}{4}pq + \frac{1}{8}p(p-1) + \frac{1}{4}qa_3 + \frac{1}{4}q(q-1) &= 1/6.
\end{align*} \]  
\[ (5.1.21) \]

The solution for this set of equations is given by

\[ p=2/3, \quad q=1/6, \quad a_3=5/2, \quad a_2=-3/2, \quad a_1=1/2. \]

Hence, one of the 3-stage GM Runge-Kutta method of order 3 takes the following form

\[ \begin{align*}
k_1 &= f(x_n, y_n) \\
k_2 &= f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}hk_1) \\
k_3 &= f(x_n + h, y_n - \frac{3}{2}hk_1 + \frac{5}{2}hk_2) \\
y_{n+1} &= y_n + h(k_1/6 + k_2/2 + k_3/6).
\end{align*} \]  
\[ (5.1.22) \]
The geometric symbol generating matrix is therefore given by

\[
A = \begin{bmatrix}
0 & 0 & 0 \\
\frac{1}{6} & 0 & 0 \\
-\frac{3}{2} & \frac{5}{2} & 0 \\
\frac{1}{6} & \frac{2}{3} & \frac{1}{6}
\end{bmatrix}
\]

Different formulae may be obtained by varying the internal points in the x-axis where \( k_2 \) and \( k_3 \) should be evaluated.

It is interesting to note that the set of values of the parameters \( a_1, a_2, a_3, p \) and \( q \) which form the 3rd order Runge-Kutta formula in the arithmetic sense does not automatically produce a 3rd order formula in the geometric sense, and vice versa. For example, the symbol generating matrix

\[
A = \begin{bmatrix}
0 & 0 & 0 \\
1 & 0 & 0 \\
-1 & 2 & 0 \\
\frac{1}{6} & \frac{2}{3} & \frac{1}{6}
\end{bmatrix}
\]

will form the Kutta's third order rule in the arithmetic way but produces only a second order formula with principal error term \( \frac{h^3}{24} y^2 \) when used in the geometric way.

Another example of solution to equations (5.1.18) is given by

\[ a_1 = \frac{1}{3}, \ a_2 = -\frac{1}{6}, \ a_3 = \frac{5}{6}, \ p = 0, \ q = \frac{3}{4}. \]

The symbol generating matrix for the resulting method is
and the formula is given by

\[
\begin{align*}
    k_1 &= f(x_n, y_n) \\
    k_2 &= f(x_n + \frac{1}{3}h, y_n + \frac{1}{3}hk_1) \\
    k_3 &= f(x_n + \frac{2}{3}h, y_n - \frac{1}{3}hk_2 + \frac{5}{6}hk_1) \\
    y_{n+1} &= y_n + h(k_1/4, 3/4, k_3).
\end{align*}
\]

This formula is of the third order. However the same generating matrix will only produce a method of order 2 when used in the arithmetic sense, with principal error \(\frac{1}{24}h^3 f''y^2\).

**Other Solutions:**

Other sets of solutions to the system of equations (5.1.18) are listed in Table 5.2. Each of them forms a method of order 3 but the error terms differ from one to another.

### Table 5.2: Some values of the parameters that give third order GM Runge-Kutta methods

<table>
<thead>
<tr>
<th>Set</th>
<th>(a_1)</th>
<th>(a_2)</th>
<th>(a_3)</th>
<th>(w_1)</th>
<th>(w_2)</th>
<th>(w_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/2</td>
<td>-3/2</td>
<td>5/2</td>
<td>1/6</td>
<td>2/3</td>
<td>1/6</td>
</tr>
<tr>
<td>2</td>
<td>1/3</td>
<td>-1/6</td>
<td>5/6</td>
<td>1/4</td>
<td>0</td>
<td>3/4</td>
</tr>
<tr>
<td>3</td>
<td>1/4</td>
<td>0</td>
<td>(\frac{1}{2})</td>
<td>2/3</td>
<td>-4/3</td>
<td>5/3</td>
</tr>
<tr>
<td>4</td>
<td>1/4</td>
<td>-3/4</td>
<td>3/2</td>
<td>1/9</td>
<td>1/3</td>
<td>5/9</td>
</tr>
</tbody>
</table>
Note that the 4th order term in the expansion of \( h(k_1^{1-p/q}k_2^{p/q}k_3^q) \) is given by

\[
\begin{align*}
\mathcal{h}^4 \{ f^3_{yyy} &+ \frac{1}{64}(a_2+a_3)^3 + \frac{1}{6}pq\frac{1}{3} + f^2_{y}f^2_{yy} [qa_1^2a_3 \\ &+ pq(a_2+a_3)+tq(q-1)(a_2+a_3)^3+pq\frac{1}{4}a_1^2a_3^2 \\ &+ pq^2(a_2+a_3)+tq(p-1)a_1^3 + ff^3_{y}a_1^3 + pq(q-1)a_1(a_2+a_3)^2 \\
&+ pq(q-1)(q-2)(a_2+a_3)^3 + pq(a_2+a_3)^3 + pq^2a_1^2a_3^2 + pq^2a_1^2a_3^2 \\ &+ pq^2(a_2+a_3) + \frac{1}{6}pq(p-1)(p-2)a_1^3] \}
\end{align*}
\]

The fourth order term in the Taylor series expansion of \( y(x_{n+1}) \) about \( x_n \) is given by

\[
\frac{1}{24} h^4 \{ f^3_{yyy} + 4f^2_{y}f_{yy} + ff^3_{y} \}
\]

The difference of these two quantities after a suitable substitution of the parameters will constitute the principal error term for the related method when it is third order accurate. For example, the LTE for the formula with the second set of parameters is given by

\[
\text{LTE} = \frac{h^3}{432} \{ 2f^3_{yyy} + 9f^2_{yy} + 28ff^3_{y} \}
\]

4-stage GM Runge-Kutta methods

The general 4-stage explicit GM Runge-Kutta formula is given by

\[
\begin{align*}
k_1 &= f(y_n) \\
k_2 &= f(y_n + ha_1k_1) \\
k_3 &= f(y_n + ha_2k_1 + ha_3k_2) \\
k_4 &= f(y_n + ha_4k_1 + ha_5k_2 + ha_6k_3) \\
y_{n+1} &= y_n + h(k_1^{w_1}k_2^{w_2}k_3^{w_3}k_4^{w_4})
\end{align*}
\]

(5.1.26)
For simplicity, since $w_1 + w_2 + w_3 + w_4 = 1$, we set $w_2 = p$, $w_3 = q$, $w_4 = r$ and so it follows that $w_1 = 1 - p - q - r$. The Taylor series expansion of $k_1$, $k_2$ and $k_3$ about $y_n$ have been given in the previous analysis for determining the 3rd order methods and are given by equations (5.1.11)-(5.1.13). We now expand $k_4$ as follows (working to the 3rd order terms in $h$).

$$k_4 = f(y_n + h a_4 k_1 + h a_5 k_2 + h a_6 k_3)$$

$$= f(y_n) + h (a_4 k_1 + a_5 k_2 + a_6 k_3) f_y + \frac{1}{2} h^2 (a_4 k_1 + a_5 k_2 + a_6 k_3)^2 f_{yy}$$

$$+ \frac{1}{6} h^3 (a_4 k_1 + a_5 k_2 + a_6 k_3)^3 f_{yyy} + ...$$

But,

$$a_4 k_1 + a_5 k_2 + a_6 k_3 = a_4 f + a_5 (f + h a_1 f_y) + \frac{1}{2} h^2 a_1^2 f_{yy}$$

$$+ \frac{1}{6} h^3 a_1^3 f_{yyy} + a_6 \left\{ f + h (a_2 + a_3) f_y + h^2 (a_2 + a_3)^2 f_{yy} + h^3 (a_2 + a_3)^3 f_{yyy} \right\} + ...$$

$$= (a_4 + a_5 + a_6) f + h (a_5 a_1 + a_6 a_2 + a_6 a_3) f_y + h^2 [(a_5 a_1$$

$$+ a_6 a_2 + a_6 a_3)^2 f_{yy} + h (a_5 a_1 + a_6 a_2 + a_6 a_3)^2 f_{yy}$$

$$+ a_6 a_3 f_{yy}] + \frac{1}{6} h^3 f_{yyy} [(a_5 a_1 + a_6 a_3)^3 f_{yyy}] + ...$$

Therefore,

$$k_4 = f + h f_y [(a_4 + a_5 + a_6) f + h (a_5 a_1 + a_6 a_2 + a_6 a_3) f_y$$

$$+ h^2 [(a_5 a_1 + a_6 a_2 + a_6 a_3)^2 f_{yy} + h (a_5 a_1 + a_6 a_2 + a_6 a_3)^2 f_{yy}$$

$$+ a_6 a_3 f_{yy}] + \frac{1}{6} h^3 f_{yyy} [(a_5 a_1 + a_6 a_3)^3 f_{yyy}] + ...$$

or,

$$k_4 = f + h [(a_4 + a_5 + a_6) f_y] + h^2 [(a_5 a_1 + a_6 a_2 + a_6 a_3) f_y$$

$$+ h [(a_5 a_1 + a_6 a_3)^2 f_{yy}] + h^3 (a_5 a_1 + a_6 a_3)^2 f_{yy}$$

$$+ (a_4 + a_5 + a_6) f_y f_{yy} + a_6 a_3 f_{yy}] + \frac{1}{6} h^3 (a_4 + a_5 + a_6)^3 f_{yyy}$$

$$+ h (a_4 + a_5 + a_6) f_y f_{yy} + a_6 a_3 f_{yy}] + \frac{1}{6} h^3 (a_4 + a_5 + a_6)^3 f_{yyy}] + ...$$  \hspace{1cm} (5.1.27)
Taking the rth power,

\[ k_4^r = r^r \{ 1 + rh(a_4+a_5+a_6)f_y + rh^2[(a_5a_1+a_6a_2+a_6a_3)f_y^2 + \frac{1}{2}(a_4+a_5+a_6)^2f_yyy] + rh^3[\{ \frac{1}{2}a_5a_1^2+a_6(a_2+a_3)^2 + (a_4+a_5+a_6)(a_5a_1+a_6a_2+a_6a_3)f_y + \frac{1}{6}(a_4+a_5+a_6)^3f_yyy \} + \frac{i}{r(r-1)}h^2(a_4+a_5+a_6)^2f_y^2 + 3r(r-1)h^3.2(a_4+a_5+a_6)f_y[(a_5a_1+a_6a_2+a_6a_3)f_y^2 + \frac{1}{4}(a_4+a_5+a_6)^2f_yyy] + \frac{1}{6}r(r-1)(r-2)h^3(a_4+a_5+a_6)^3f_y^3 + \ldots \} \]

\[ = f^r \{ 1 + rh[(a_4+a_5+a_6)f_y] + rh^2[(a_5a_1+a_6(a_2+a_3))f_y^2 + \frac{i}{r}(a_4+a_5+a_6)^2f_yyy + \frac{i}{r(r-1)}(a_4+a_5+a_6)^2f_y^2] + h^3(f_yf_yy \{ \frac{1}{2}a_5a_1^2+a_6(a_2+a_3)^2 \} + f_y^3 \{ ra_6a_1a_3 + r(r-1)(a_4+a_5+a_6)(a_5a_1+a_6(a_2+a_3)) + \frac{1}{6}r(r-1)(r-2)(a_4+a_5+a_6)^3f_yyy \} + \frac{1}{6}r(r-1)(r-2)h^4 \} \} \]

The next task now is to find expressions for \( k_1^{1-p-q}k_2p_kq_kk_4^r \). But we have already obtained the expansion for \( k_1^{1-p-q}k_2p_kq_k \) from the previous analysis of the 3-stage GM Runge-Kutta methods as given by equation (5.1.16) where the 4th order terms are given by equation (5.1.24). Writing \( a_2+a_3 = s_2 \) and \( a_4+a_5+a_6 = s_3 \) will further simplify the subsequent algebra.
Thus,

\[ k_1^{1-p-q-r}k_2p_k^r = f + hff_{y}rs_3 \]
\[ + h^2f[rf_y^2(a_5a_1+a_6s_2) + \frac{1}{4}rs_3^2ff_{yy} + \frac{1}{4}r(r-1)s_3^2f_y^2] \]
\[ + h^3f[\{a_5a_1+a_6s_2\}ff_yf_{yy} + rsa_1a_3f_y^2 \]
\[ + rs_3(a_5a_1+a_6s_2)ff_yf_{yy} + \frac{1}{6}rs_3^2f_{yy} \]
\[ + r(r-1)s_3^2f_y((a_5a_1+a_6s_2)f_y^2 + \frac{1}{6}rs_3^2f_{yy}) \]
\[ + \frac{1}{6}r(r-1)(r-2)s_3^2f_y^3 + \ldots + h(pa_1+qs_2)ff_y. \]
\[ + h^2 \left[ r(a_5a_1+a_6s_2)f_y^2 + \frac{1}{4}rs_3^2f_{yy} \right] \]
\[ + \frac{1}{6}h^2[(pa_1^2+qs_2^2)f_y^2f_{yy} + (pqa_1s_2^2+pq(p-1)a_1^3+a_3^3)ff_yf_{yy} + (q(q-1)a_1^3s_2) \]
\[ + \frac{1}{6}q(q-1)(q-2)s_2^3 + pqa_1^2a_3 + \frac{1}{6}pq(q-1)a_1^2s_2 \]
\[ + \frac{1}{6}p(p-1)qa_1^2s_2^2 + \frac{1}{6}p(p-1)(p-2)a_1^3ff_y^3 \] + 0(h^4). \hspace{1cm} (5.1.29)

Now, the Taylor series expansion for \( y(x+h) \) is given by

\[ y(x+h) = y_n + hf + \frac{1}{2}hf_y + \frac{1}{6}h^3(f_y^2f_{yy} + f_{yy}f_y) \]
\[ + \frac{1}{24}h^4(f_y^3f_{yyy} + 4f_yf_{yy}f_{yy} + f_{yy}^3) + \ldots \] \hspace{1cm} (5.1.29)

Equating the coefficients of similar terms in the expansion of (5.1.28) and (5.1.29) produces the following equations, (note that equation (5.1.28) is to be multiplied with \( h \) before equating),

\[ h^2f_{yy}: \quad pa_1 + qs_2 + rs_3 = \frac{1}{4} \hspace{1cm} (5.1.30-i) \]

\[ h^3f_{yy}^2: \quad r(a_5a_1+a_6s_2) + \frac{1}{4}r(r-1)s_3^2 + (pa_1+qs_2)rs_3 + pqa_1s_2 \]
\[ + \frac{1}{6}pq(p-1)a_1^2 + qa_1a_3 + \frac{1}{6}pq(q-1)s_2^2 = 1/6 \hspace{1cm} (5.1.30-ii) \]

\[ h^3f_{yy}^3: \quad \frac{1}{6}rs_3^2 + \frac{1}{6}pa_1^2 + \frac{1}{6}qs_2^2 = 1/6 \] \hspace{1cm} (5.1.30-iii)
These give six equations for the determination of the nine unknowns \( a_1, a_2, \ldots, a_6, p, q, r \). Three additional equations can be chosen at our disposal, but some might give rise to the non-existence of the solution to the nonlinear system. A logical choice for the additional equation is to set the following,

\[ a_1 = \alpha \]
\[ a_2 + a_3 = \beta \]
\[ a_4 + a_5 + a_6 = \gamma \]

This will mean that \( k_2, k_3 \) and \( k_4 \) are to be computed at the points \( x_n + \alpha h, x_n + \beta h, \) and \( x_n + \gamma h \) respectively during the nth integration step.

Now, we solve the system by using a NAG Library Routine C05NBVF in double precision arithmetic. This routine finds a zero of a system of \( n \) nonlinear functions in \( n \) variables by a modification of the Powell
hybrid method. It chooses the correction at each step as a convex combination of the Newton and scaled gradient directions. Under reasonable conditions this guarantees global convergence for starting points far from the solution and a fast rate of convergence.

In our case the functions are given as in equations (5.1.30) and (5.1.31) but with taking all terms to the LHS so that the system will have the form

\[ f_i(x_1, x_2, \ldots, x_9) = 0, \quad i = 1, 2, \ldots, 9, \]

where \( x_1 = a_1, x_2 = a_2, \ldots, x_6 = a_6, x_7 = p, x_8 = q \) and \( x_9 = r \).

Three sets of solutions corresponding to three different choices of \( \alpha, \beta \) and \( \gamma \) in equations (5.1.31) are given in Table 5.3. In this table the row values of 'res' indicate the residual of the system at the end of the iteration, and is given by

\[ \text{res} = \frac{1}{9} \sum_{i=1}^{9} |f_i|^2 \]

<table>
<thead>
<tr>
<th>Set 1</th>
<th>Set 2</th>
<th>Set 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3333333333D+00</td>
<td>0.4000000000D+00</td>
<td>0.2500000000D+00</td>
</tr>
<tr>
<td>-0.6141097397D+00</td>
<td>0.6000000000D+00</td>
<td>0.3333333333D+00</td>
</tr>
<tr>
<td>0.1280776406D+01</td>
<td>-0.3899161719D+01</td>
<td>0.7654671353D+00</td>
</tr>
<tr>
<td>0.1184658438D+01</td>
<td>0.1666666666D+00</td>
<td>0.2000000000D+00</td>
</tr>
<tr>
<td>-0.1526987657D+01</td>
<td>0.9938660721D+00</td>
<td>-0.9062867243D+00</td>
</tr>
<tr>
<td>0.1342329219D+01</td>
<td>-0.1562481137D+01</td>
<td>0.1412257479D+01</td>
</tr>
<tr>
<td>0.2750000000D+00</td>
<td>0.5489915610D+00</td>
<td>0.2298261312D+00</td>
</tr>
<tr>
<td>0.3750000000D+00</td>
<td>-0.6732251922D-01</td>
<td>0.3910179356D-01</td>
</tr>
<tr>
<td>0.1250000000D+00</td>
<td>0.3874927769D+00</td>
<td>0.2735828187D+00</td>
</tr>
<tr>
<td>0.5933996301D-18</td>
<td>0.1306189621D-18</td>
<td>0.3033032586D-17</td>
</tr>
</tbody>
</table>

The solutions are given in the real number form. Since the magnitudes of the residual are less than \( 10^{-10} \) we expect the results to be accurate around nine or ten decimal places so that in practice we
can use these parameters for computations up to a similar degree of accuracy, as in the case of single precision arithmetic.

For easier representation of the method, we may also approximate them by rational numbers of the form $M/N$ where the integers $M$ and $N$ are found to be of order $O(10^3)$ for correct representations to the seventh decimal place. As an example, the first set of parameters in Table 5.3 are represented, to seventh decimal place accuracy, by the following fractional values:

$$a_1 = 1/3, \quad a_2 = -2742/4465, \quad a_3 = 4817/3761,$$
$$a_4 = 5081/4289, \quad a_5 = -6818/4465, \quad a_6 = 5521/4113,$$
$$p = 3/8, \quad q = 3/8 \text{ and } r = 1/8.$$

A more accurate fractional form may be obtained by allowing $M$ and $N$ to assume larger values, but since the real values of the parameters are only approximations, the above fractional values are considered sufficient.

Since the first set of the parameters are distributed not far away from zero, this set gives a more uniform weight in the computation of the geometric mean in the formula. Also the parameters correspond to the evaluation of the function at simple intervals of $h/3$, i.e., $a=1/3$, $b=2/3$ and $y=1$. For this reason, we will use the related 4th order formula to represent the 4th order GM Runge-Kutta methods, and we may refer to this in the future analysis as the 4th order GM Runge-Kutta formula. The method is given by

\[
k_1 = f(x_n, y_n),
\]
\[
k_2 = f(x_n + \frac{1}{3}h, y_n + \frac{1}{3}hk_1)
\]
\[
k_3 = f(x_n + \frac{2}{3}h, y_n - \frac{2742}{4465}hk_1 + \frac{4817}{3761}hk_2)
\]
\[
k_4 = f(x_n + h, y_n + \frac{5081}{4289}hk_1 - \frac{6818}{4465}hk_2 + \frac{5521}{4113}hk_3)
\]
\[
y_{n+1} = y_n + h \left[ k_1 \frac{1}{8} + k_2 \frac{3}{8} + k_3 \frac{3}{8} + k_4 \frac{1}{8} \right].
\]
We have thus so far seen how the complexity in finding the GM Runge-Kutta formulae increases with the order of accuracy of the formula. In particular, the amount of algebra involved in producing the 4th order formula is tremendous but we will temporarily avoid finding the error terms.

An alternative way to producing the six equations of conditions as given by equations (5.1.30) is to use the REDUCE symbolic computation package. For this purpose we define the operators "power" and "taylor" which have the operational effects as follows:

(i) \[ \text{power}(x, k) = 1 + kx + \frac{k(k-1)x^2}{2} + \frac{k(k-1)(k-2)x^3}{6} + \ldots, \]

i.e. the Taylor series expansion of \((1+x)^k\).

(ii) \[ \text{taylor}(x) = f + xf_1 + \frac{x^2f_2}{2} + \frac{x^3f_3}{6} + \ldots, \]

i.e. the Taylor series expansion of \(f(x+y)\) about \(y\), where \(f\) and its derivatives \(f_1 = f', f_2 = f'', \ldots\) are evaluated at \(y\).

In this way the \(k_i\)'s as given in equations (5.1.26) are simply given by

\[ k_1 = f \]
\[ k_2 = \text{taylor}(ha_1 k_1) \]
\[ k_3 = \text{taylor}(ha_2 k_1 + ha_3 k_2) \]
\[ k_4 = \text{taylor}(ha_4 k_1 + ha_5 k_2 + ha_6 k_3) \]

which are expanded, in effect, by the operator "taylor" into the form

\[ k_1 = f + (\ldots)f_1 + (\ldots)f_2 + \ldots. \]

This can be rearranged into the following form,

\[ k_1 = f \left\{ 1 + (\ldots) \frac{f_1}{f} + (\ldots) \frac{f_2}{f} + \ldots \right\} = f \left\{ 1 + x_1 \right\} \]

\[ (5.1.35) \]

where \(x_1\) are obtained from the relation

\[ x_1 = \frac{k_i}{f} - 1, \; i = 1,2,3,4. \]

\[ (5.1.36) \]
In this way we can effectively use the operator "power" to obtain the expansion of \( k_2^p, k_3^q \) and \( k_4^r \) by using the following operations:

\[
\begin{align*}
k_2^p &= \text{power}(x_2, p) \\
k_3^q &= \text{power}(x_3, q) \\
k_4^r &= \text{power}(x_4, r)
\end{align*}
\]

Then, from the relations

\[
\frac{y_{n+1}}{\text{LHS}} = \frac{y_n + h(k_1^{1-p-q-r}k_2^p k_3^q k_4^r)}{\text{RHS}}
\]

the RHS is immediately obtained by multiplications, whereas the LHS is given by

\[
\begin{align*}
\text{LHS} &= y_n + hf + \frac{1}{6}h^2(ff_f + ff_l) + \frac{1}{24}h^4(f^3f_3 + 4f^2f_2ff_1 + f^2f_1^2) + \frac{1}{120}h^5(f^4f_4 + 7f^3f_3f_1^2 + 11f^2f_2f_1f_2 + ff_1^4) + \ldots \\
&= \text{(c.f. individual expressions for } y^{(n)} \text{ in equations (3.5.5)).}
\end{align*}
\]

The difference between the LHS and the RHS produces the error terms of the formula. For the method to have order 4, terms up to \( h^4 \) must vanish, and this results in the six equations of conditions given by equations (5.1.30). The computer program "GMRK" for this symbolic manipulation is provided in Appendix 3.

As we have mentioned earlier, the error analysis for the 4th order GM Runge-Kutta formula is highly complicated if it were to be examined manually. With the program "GMRK" we could now easily compute the error terms for the 4th order methods by substituting the corresponding parameters into the program. As an example, the method with the first set of parameters given in Table 5.3 has a local truncation error "approximately" given by
LTE = h^5 \left[ -1.5432 \times 10^{-4} f_y^4 + 1.6827 \times 10^{-3} f_y^3 \right. \\
\left. \quad - 2.4563 \times 10^{-2} f_y^2 - 2.9862 \times 10^{-2} f_y^2 \right. \\
\left. \quad + 4.6304 \times 10^{-2} \right].

The word "approximately" here refers to the fact that all the parameters are obtained by using an "approximate" iterative method and are therefore subject to errors, which in our case is very small for the purpose of single precision arithmetic computations.

5.1.3 Stability analysis for the GM Runge-Kutta methods

Let us now consider the absolute stability property of the GM Runge-Kutta methods which we developed in the preceding section. For this purpose we will determine the region of absolute stability of the methods of various order. For each order we will probably choose only one method to represent them, although as we will see later, different methods of similar order and stage number may have different regions of absolute stability.

Stability of the 1st to 4th order methods

The first order method in the family of the GM Runge-Kutta formulae is the Euler's method. Its stability property has been discussed in Section 3.5.4.

Let us now proceed with examining the stability of the 2nd order 2-stage formula. For this purpose we will use formula (5.1.9) as a representative. By applying formula (5.1.9) with R=2 to the test equation \( y' = \lambda y \) we obtain

\[
k_1 = f(x_n, y_n) = \lambda y_n \\
k_2 = f(x_n + h, y_n + h k_1) = \lambda (1 + h \lambda) y_n.
\]
Hence,

\[ y_{n+1} = y_n + h \sqrt{\lambda y_n \cdot \lambda (1+h\lambda)} y_n \]

\[ = y_n + h\lambda \sqrt{1+h\lambda} y_n \]

\[ = (1 + h\lambda \sqrt{1+h\lambda}) y_n. \]

Introducing the notation used in Section 3.5.4, we obtain, for this formula,

\[ \frac{y_{n+1}}{y_n} = Q(h\lambda) = 1 + h\lambda \sqrt{1+h\lambda}. \quad (5.1.38) \]

The set of values of \( h \) in the complex plane for which the polynomial \( Q(h\lambda) \) has magnitude less than unity determines the region of absolute stability of the method. This is shown in Figure 5.1 indicated by \( k=2 \).

In a similar manner we apply the third order GM Runge-Kutta formula (5.1.23) to the test equation \( y' = \lambda y \) to obtain

\[ k_1 = f(x_n, y_n) = \lambda y_n \]
\[ k_2 = f(x_n + \frac{1}{3}h, y_n + \frac{1}{3}hk_1) = \lambda (1+h\lambda/3)y_n \]
\[ k_3 = f(x_n + \frac{2}{3}h, y_n - \frac{1}{6}hk_1 + \frac{5}{6}hk_2) \]
\[ = \lambda [1 - \frac{1}{6}h\lambda + \frac{5}{6}h\lambda(1+h\lambda)] y_n \]

\[ y_{n+1} = y_n + h(k_1^{1/4}k_3^{3/4}) \]
\[ = y_n + h(\lambda y_n)^{1/4}(\lambda [1 - \frac{1}{6}h\lambda + \frac{5}{6}h\lambda(1+\frac{1}{3}h\lambda)]) y_n^{3/4} \]
\[ = y_n + h\lambda [1 - \frac{1}{6}h\lambda + \frac{5}{6}h\lambda(1+\frac{1}{3}h\lambda)]^{3/4} y_n \]
\[ = \{1 + h\lambda(1+\frac{2}{3}h\lambda + \frac{5}{18}h^2\lambda^2)^{3/4}\} y_n, \]
i.e. \[ Q(z) = 1 + z(1 + \frac{2}{3}z + \frac{5}{18}z^2)^{3/4} \] (5.1.39)

where we have denoted \( h\lambda \) by \( z \).

Finally we apply the general 4th order GM Runge-Kutta formula (5.1.26) to the test equation \( y' = \lambda y \). This will result in the following equations

\[
\begin{align*}
  k_1 &= f(x_n, y_n) = \lambda y_n \\
  k_2 &= f(x_n + \alpha h, y_n + h \lambda a_1 k_1) = \lambda (1 + h \lambda a_1) y_n \\
  k_3 &= f(x_n + \beta h, y_n + h \lambda a_2 k_1 + h \lambda a_3 k_2) \\
        &= \lambda (1 + h \lambda a_2 + h \lambda a_3 (1 + h \lambda a_1)) y_n \\
  k_4 &= f(x_n + \gamma h, y_n + h \lambda a_4 k_1 + h \lambda a_5 k_2 + h \lambda a_6 k_3) \\
        &= \lambda \left( 1 + h \lambda a_4 + h \lambda a_5 (1 + h \lambda a_1) + h \lambda a_6 [1 + h \lambda a_2 \\
        &+ h \lambda a_3 (1 + h \lambda a_1)] \right) y_n
\end{align*}
\]

\[
y_{n+1} = y_n + h \left( \lambda y_n \right)^{1-p-q-r} \left( \lambda (1 + h \lambda a_1) y_n \right)^p \left( \lambda (1 + h \lambda a_2 \\
&+ h \lambda a_3 (1 + h \lambda a_1)) y_n \right)^q \left( \lambda \left( 1 + h \lambda a_4 + h \lambda a_5 (1 + h \lambda a_1) \\
&+ h \lambda a_6 [1 + h \lambda a_2 + h \lambda a_3 (1 + h \lambda a_1)] \right) y_n \right)^r.
\]

Dividing the RHS by \( y_n \) produces the stability polynomial \( Q(h\lambda) \). By denoting \( h\lambda = z \), this polynomial is given by

\[
Q(z) = 1 + z(1 + za_1)^p (1 + za_2 + za_3 (1 + za_1))^q (1 + za_4 + za_5 (1 + za_1)) \\
+ za_6 [1 + za_2 + za_3 (1 + za_1)])^r
\] (5.1.40)

In particular, if we require the stability region for any particular 4th order 4-stage method we replace the corresponding values of the parameters in (5.1.40). As an example, by substituting the second set of parameters of Table 5.3 into (5.1.40) we obtain the stability polynomial for the method given by (5.1.32). The regions of absolute
Figure 5.1: Stability regions for GM Runge-Kutta formulae of order $k = 1, 2, 3, 4$.

Figure 5.2: Stability regions for the 3rd order formula with set 3 parameters, and 4th order formula with set 1 parameters.
stability for the four methods we consider here are shown in Figure 5.1. Inside the closed curves the magnitude of the respective stability polynomials $Q(z)$ are less than unity.

Note that these regions of absolute stability differ only very slightly from the stability regions of the corresponding classical Runge-Kutta methods given in Figure 3.5. We also note that the higher order methods do not have significantly larger regions of stability but only improved accuracy within the region of stability. As a general rule we may say that the GM Runge-Kutta formulae can successfully integrate equations of the form $y' = Ay$ only if the eigenvalues $\lambda_i$ of $A$ are such that $|h\lambda_i|$ lies between 1 and 2.5 approximately depending on the formula being used.

However, an important consideration in comparing the stability regions is that, for a given $R$, $R=1,2,3,4$, all $R$-stage classical Runge-Kutta methods of order $R$ have the same interval of absolute stability (see Lambert [1973], p.138). This is not the case with the GM formulae where the regions will depend on the particular choice for the free parameters. For example, for $R=4$, if we use the first set of parameters given in Table 5.3 we will obtain the smaller stability region shown in Figure 5.2. But, by using the third set parameters given in Table 5.2 for the third order GM formulae we obtain the larger region as shown in Figure 5.2. This opens the prospect of producing methods which possess reasonably large stability regions in the desired direction of the complex plane which makes them more applicable for a wider range of problems. We leave the subject for further research.
5.1.4 Numerical Example

As an example we solve the nonlinear initial value problem

\[ y' = 1 + y^2, \quad y(0) = 1 \quad \text{for} \quad 0 \leq x \leq 0.7. \]

The exact solution is given by \( y(x) = \tan(x + \pi/4) \). We compare the solutions obtained by using the 4th order classical Runge-Kutta method and the 4th order GM Runge-Kutta method we developed in Section 5.1.1 and by using the step size \( h = 0.01 \). The results, printed in every 10 steps are shown in Table 5.4. It is clear that the GM Runge-Kutta formula is capable of producing results that are as accurate as the ones obtained by using the classical formula. In this example, particularly, the results are more accurate at the earlier steps of integration, i.e., for \( 0 \leq x \leq 0.4 \). The only drawback in using the GM formula is, as we expect, the increase in the computational complexity of the calculation which in this case involves about 60 percent of extra work.

<table>
<thead>
<tr>
<th>( x )</th>
<th>( \text{Exact Solution} )</th>
<th>( \text{Numerical Solution} )</th>
<th>( \text{Error} )</th>
<th>( \text{cpu time} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>0.1223049E+01</td>
<td>0.1223049E+01</td>
<td>-0.4436639E-07</td>
<td>19995</td>
</tr>
<tr>
<td>0.20</td>
<td>0.1508498E+01</td>
<td>0.1508498E+01</td>
<td>-0.5124582E-07</td>
<td>19713</td>
</tr>
<tr>
<td>0.30</td>
<td>0.1895765E+01</td>
<td>0.1895765E+01</td>
<td>-0.7094620E-07</td>
<td>19699</td>
</tr>
<tr>
<td>0.40</td>
<td>0.2464963E+01</td>
<td>0.2464963E+01</td>
<td>-0.9026128E-07</td>
<td>19722</td>
</tr>
<tr>
<td>0.50</td>
<td>0.3408223E+01</td>
<td>0.3408223E+01</td>
<td>-0.1611525E-06</td>
<td>19714</td>
</tr>
<tr>
<td>0.60</td>
<td>0.5331855E+01</td>
<td>0.5331855E+01</td>
<td>0.2362229E-07</td>
<td>19775</td>
</tr>
<tr>
<td>0.70</td>
<td>0.1168137E+02</td>
<td>0.1168135E+02</td>
<td>0.2622175E-04</td>
<td>19715</td>
</tr>
</tbody>
</table>

(a) Results by using the classical Runge-Kutta formula:

\( h = 0.01, \quad k = 10 \).

continued...
(b) Results by using the GM Runge Kutta formula:

$h = 0.01, k = 10.$

<table>
<thead>
<tr>
<th>$x$</th>
<th>Exact Solution</th>
<th>Numerical Solution</th>
<th>Error</th>
<th>cpu time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>0.1223049E+01</td>
<td>0.1223049E+01</td>
<td>-0.4309193E-07</td>
<td>32370</td>
</tr>
<tr>
<td>0.20</td>
<td>0.1508498E+01</td>
<td>0.1508498E+01</td>
<td>-0.4541190E-07</td>
<td>31481</td>
</tr>
<tr>
<td>0.30</td>
<td>0.1895765E+01</td>
<td>0.1895765E+01</td>
<td>-0.4793913E-07</td>
<td>31476</td>
</tr>
<tr>
<td>0.40</td>
<td>0.2464963E+01</td>
<td>0.2464963E+01</td>
<td>0.8257042E-08</td>
<td>31544</td>
</tr>
<tr>
<td>0.50</td>
<td>0.3408223E+01</td>
<td>0.3408223E+01</td>
<td>0.3845414E-06</td>
<td>31492</td>
</tr>
<tr>
<td>0.60</td>
<td>0.5331855E+01</td>
<td>0.5331849E+01</td>
<td>0.5353902E-05</td>
<td>31495</td>
</tr>
<tr>
<td>0.70</td>
<td>0.1168137E+02</td>
<td>0.1168110E+02</td>
<td>0.2739411E-03</td>
<td>31551</td>
</tr>
</tbody>
</table>
5.2 A NEW STRATEGY IN THE USE OF RUNGE KUTTA METHODS WITH ERROR CONTROL FOR SOLVING \[ y' = Ay \]

Most modern computer subroutines for differential equation solving have built-in devices for the control of order and step size. The user need only state the error requirement and, of course, the information necessary to define the differential equation (or set of equations) to be solved. The program will then select the optimum order and step size to achieve the user's desires at lowest cost.

One problem with Runge-Kutta methods of the past was the difficulty of estimating the error in the results during the calculation, thus making automatic error control difficult. A number of methods have been developed that overcome this deficiency by including a means of calculating an estimate of the error in the results. This allows the routine to adjust the step size automatically to maintain the desired accuracy. Amongst these methods are the Merson [1957], Scraton [1964] and Fehlberg [1964] variations.

5.2.1 Arithmeto-Geometric Mean (AGM) strategy for error control

The Merson [1957] (arithmetic mean) variation of the Runge-Kutta method is well known and is given by

\[
\begin{align*}
    k_1 &= f(x_n, y_n), \\
    k_2 &= f(x_n + \frac{1}{3}h, y_n + \frac{1}{3}hk_1), \\
    k_3 &= f(x_n + \frac{1}{3}h, y_n + \frac{1}{6}hk_1 + \frac{1}{2}hk_2), \\
    k_4 &= f(x_n + \frac{2}{3}h, y_n + \frac{1}{3}hk_1 + \frac{2}{3}hk_2), \\
    k_5 &= f(x_n + h, y_n + \frac{1}{6}hk_1 + \frac{1}{2}hk_3 + 2hk_4), \\
    y_{n+1} &= y_n + \frac{h}{6} (k_1 + 4k_4 + k_5) .
\end{align*}
\] (5.2.1)
Alternatively this may be considered as being comprised from the following computational stages

\[ y_{n+1/3}^* = y_n + \frac{1}{3}h \cdot f(x_n, y_n) \]  
(Euler predictor - 1/3 step)  

\[ y_{n+1/3}^{**} = y_n + \frac{h}{6} \left[ f(x_n, y_n) + f(x_{n+1/3}, y_{n+1/3}^*) \right] \]  
(Trapezoidal corrector - 1/3 step)  

\[ y_{n+1/2}^* = y_n + \frac{h}{8} \left[ f(x_n, y_n) + 3f(x_{n+1/3}, y_{n+1/3}) \right] \]  
(Adams-Bashforth type predictor - half step)  

\[ y_{n+1}^* = y_n + \frac{h}{2} \left[ f(x_n, y_n) + 3f(x_{n+1/3}, y_{n+1/3}) \right] \]  
\[ + 4f(x_{n+1/2}, y_{n+1/2}) \]  
(Adams's Bashforth type predictor - full step)  

\[ y_{n+1}^{**} = y_n + \frac{h}{6} \left[ f(x_n, y_n) + 4f(x_{n+1/2}, y_{n+1/2}) \right] \]  
\[ + f(x_{n+1}, y_{n+1}) \]  
(Simpson's rule corrector - full step).

In this formula the accuracy of \( y_{n+1}^* \) and \( y_{n+1}^{**} \) are given by

\[ y_{n+1}^* = y(x_{n+1}) - \frac{h}{120} \cdot y^5 + O(h^6) \]  

\[ y_{n+1}^{**} = y(x_{n+1}) - \frac{1}{720} \cdot y^5 + O(h^6) \]

Subtracting,

\[ y_{n+1}^{**} - y_{n+1}^* = \frac{5}{720} \cdot h \cdot y^5 + O(h^6) \]

Therefore the error in the final result is given by

\[ y(x_{n+1}) - y_{n+1}^{**} = \frac{1}{720} \cdot h \cdot y^5 = \frac{1}{5} \left( y_{n+1}^{**} - y_{n+1}^* \right). \]  
(5.2.3)

Obviously, the error estimate is obtained for little additional computational effort.

For the same reason the GM Runge-Kutta formulae also suffer from a similar difficulty of error control which we mentioned earlier. In response to this, it seems reasonable to obtain, in a similar manner to the Merson's procedure, the error control formulation of the GM Runge-Kutta methods. At one stage we feel it is possible to derive a
method in which the intermediate values similar to the stages in (5.2.2) are also evaluated in the geometric mean sense. This will necessarily depend on our ability to show the existence of such a formula, and even if it does exist we feel that the complexity in the computation would be too great to make it competitive with the classical formulae.

In contemplating this difficulty, it is interesting to find that it would be more beneficial if we could incorporate the combination of the AM and GM Runge-Kutta formulae to produce a totally new way of estimating the error. Basically this idea will work as follows. Suppose it is desirable to produce a 4th order method with automatic error control by using this strategy. The intermediate values as exemplified by equations (5.2.2), or similarly the $k_i$'s in (5.2.1), remain to be computed in the AM sense. It is only in the last stage that the GM averaging takes place in addition to the AM. It is important that we adjust the values of the free parameters in the $k_i$'s so that they can be used to compute two 4th order approximations to the solution via AM and GM formulae. A correction may then be made by comparing the accuracy of these two final values. This sounds very likely a possibility, since in the derivation of the GM Runge-Kutta method in Section 5.1 we still have extra degree of freedom in the value of the variables. In particular, we have set $a_1 = \alpha$, $a_2 + a_3 = \beta$, $a_4 + a_5 + a_6 = \gamma$, where $\alpha, \beta, \gamma$ are then fixed. This freedom may usefully be utilised to adjust the weighting parameters in the AM formula, so that two formulae (AM and GM) are obtained that make use of the common values of $k_i$'s. Since this procedure involves computations by using the arithmetic and geometric means, we will call this strategy the Arithmeto-Geometric Mean strategy or in short, the AGM strategy. We will now consider the details of the derivation of the formulae.
5.2.2 Derivation of the 4th order AGM formula

In this section we will establish a 4th order Runge-Kutta formula by adjusting the parameters in the $k_i$'s so that they can be used to compute the 4th order approximations to the solution via the AM and GM formulae. An estimate of the local truncation error is then computed from the difference of the final results. In particular, we are looking for the following formula,

\[
\begin{align*}
    k_1 &= f(Y_n) \\
    k_2 &= f(Y_n + h a_1 k_1) \\
    k_3 &= f(Y_n + h a_2 k_1 + h a_3 k_2) \\
    k_4 &= f(Y_n + h a_4 k_1 + h a_5 k_2 + h a_6 k_3) \\
    y_{n+1}^* &= y_n + h k_1^p p_{k_1} + k_2 + k_3 + k_4 \\
    y_{n+1}^{**} &= y_n + h (w_1 k_1 + w_2 k_2 + w_3 k_3 + w_4 k_4)
\end{align*}
\]

where $w_1 + w_2 + w_3 + w_4 = 1$ \( (5.2.5) \)

In Section 5.1 we have successfully established a formula for $y_{n+1}^*$ by solving for the nine parameters $a_1, ..., a_6, p, q$ and $r$ from the six equations of condition (5.1.30). In achieving this, three additional equations had to be supplied at our discretion for the purpose of uniqueness in the solution. With the introduction of $y_{n+1}^{**}$ to the chain of this formula we expect to have six additional equations similar in form to that of the previous six. However three more variables are involved, namely $w_2, w_3$ and $w_4$, but not $w_1$ since it can be expressed as $w_1 = 1 - w_2 - w_3 - w_4$ from the relation (5.2.5). Therefore we will have in total the right number of 12 equations in the 12 parameters to be solved.

Recall that $k_1, k_2, k_3$ and $k_4$ are given by (5,1,11), (5,1,12), (5,1,13) and (5,1,27) respectively, we now have,
Comparing the RHS of \( y_{n+1} \) term by term with the Taylor series expansion of \( y(x_n+h) \) given by (5.1.29) we now obtain the following equations of condition for the AM formula,

\[
\begin{align*}
\text{Comparing the RHS of } y_{n+1} \text{ term by term with the Taylor series expansion of } y(x_n+h) \text{ given by (5.1.29) we now obtain the following equations of condition for the AM formula,}
\end{align*}
\]
unknown parameters. By using the NAG Library routine C05NBF in double precision arithmetic, and suitable starting values of the parameters, we obtain the following solution

\[
\begin{align*}
    a_1 &= 0.1012632771D+01 \\
    a_2 &= 0.4570328148D+00 \\
    a_3 &= 0.1483148374D+00 \\
    a_4 &= 0.9635748681D+00 \\
    a_5 &= -0.5855741721D-01 \\
    a_6 &= 0.1253292577D+00 \\
    p &= -0.3630955785D+01 \\
    q &= 0.9834843363D+00 \\
    r &= 0.3475990029D+01 \\
    w_2 &= -0.2275946443D+01 \\
    w_3 &= 0.8654853197D+00 \\
    w_4 &= 0.2213602886D+01 \\
    \text{res} &= 0.1805254356D-17
\end{align*}
\]  

(5.2.8)

These parameters can be approximated by rational fractions up to eight significant figures accuracy as follows:

\[
\begin{align*}
    a_2 &= 9173/19992, \quad a_3 = 2900/19553, \quad a_4 = 7404/7687, \\
    a_5 &= -1618/27631, \quad a_6 = 3854/30751, \quad p = -9829/2707, \\
    q &= 8575/8719, \quad r = 4126/1187, \quad w_2 = -1864/819, \\
    w_3 &= 9915/11456, \quad w_4 = 6021/2720, \quad a_1 = 3527/3483.
\end{align*}
\]

Thus the method is given by the following computational processes,
\[ \begin{align*}
\mathcal{K}_1 &= f(x_n, y_n) \\
\mathcal{K}_2 &= f(x_n + \frac{3527}{3483} h, y_n + \frac{3527}{3483} h \mathcal{K}_1) \\
\mathcal{K}_3 &= f(x_n + \frac{566}{935} h, y_n + \frac{9137}{19992} h \mathcal{K}_1 + \frac{2900}{19553} h \mathcal{K}_2) \\
\mathcal{K}_4 &= f(x_n + \frac{8590}{8337} h, y_n + \frac{7407}{7687} h \mathcal{K}_1 - \frac{1618}{27631} h \mathcal{K}_2 + \frac{3854}{30751} h \mathcal{K}_3) \\

y_{n+1}^* &= y_n + h \mathcal{K}_1 + \frac{165}{854} \mathcal{K}_2 - \frac{9829}{2707} \mathcal{K}_3 - \frac{8576}{8917} \mathcal{K}_4 \\

y_{n+1}^{**} &= y_n + h \left( \frac{1629}{8275} \mathcal{K}_1 - \frac{1864}{819} \mathcal{K}_2 + 9915 \mathcal{K}_3 + \frac{6021}{2720} \mathcal{K}_4 \right) \\

\end{align*} \]

(5.2.9)

\[ y_{n+1}^{**} = y_n + h \left( \frac{1629}{8275} \mathcal{K}_1 - \frac{1864}{819} \mathcal{K}_2 + 9915 \mathcal{K}_3 + \frac{6021}{2720} \mathcal{K}_4 \right) \\

\]

5.2.3 An Error estimate for the 4th order AGM formula

Since both \( y_{n+1}^* \) and \( y_{n+1}^{**} \) are 4th order accurate, we can write

\[ \begin{align*}
y_{n+1}^* &= y(x_{n+1}) + O(h^5) \\
y_{n+1}^{**} &= y(x_{n+1}) + O(h^5) \\
\end{align*} \]

In particular we may approximate them as follows,

\[ \begin{align*}
\text{GM:} & \quad y(x_{n+1}) = y_{n+1}^* + 0(h^5) \\
\text{AM:} & \quad y(x_{n+1}) = y_{n+1}^{**} + 0(h^5) \\
\end{align*} \]

(5.2.10)

(5.2.11)

where the quantity \( \phi \) is still to be determined.

By using the REDUCE program "GMRK", and bearing in mind that the differential equation is now only a linear one, so that the higher derivatives \( f_{yy}, f_{yyy}, \ldots \) are all zero, we can now substitute the parameters given by (5.2.8) to produce the numerical value for \( \phi \) in the local truncation error term. This is found to be

\[ \phi = -0.017462188 - \frac{1}{120} \]

By using (5.2.10) and (5.2.11) we have an estimate for the LTE of \( y_{n+1}^{**} \) given by

\[ \text{EST} = \frac{(y_{n+1}^{**} - y_{n+1}^*)}{(120 \times 0.017462188)} = \frac{(y_{n+1}^{**} - y_{n+1}^*)}{2.0954625}. \]

(5.2.12)
Note that the formula is 4th order accurate when used to any problem \( y' = f(x,y) \), although the error estimate is only correct when the function \( f(x,y) \) is linear in both \( x \) and \( y \), a fact which even the Merson formula must obey (see Lambert [1973], p.132).

The advantage of this formula over the Merson's type formulae is obvious since it requires only four function evaluations as compared to five in the Merson's formula (improvements of which are given by Scraton [1964], England [1969] and Shintani [1965, 1966]). Assuming that function evaluations are the principal time consumer, we could, by this strategy, save up to one fifth of the time used in the Merson's formula.

5.2.4 Reliability test of the error estimate

We apply this procedure to the initial value problem

\[ y' = -y, \quad y(0) = 1, \]

whose exact solution is \( y(x) = e^{-x} \). The exact truncation error, \( \text{ERR} = y_{n+1} - \exp(-x_{n+1}) \) and the estimated error, \( \text{EST} \), at each step are compared as shown on the last two columns of Table 5.5.

<table>
<thead>
<tr>
<th>( x )</th>
<th>Exact Solution</th>
<th>Numerical Solution (N)</th>
<th>Error</th>
<th>Estimated Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.10000000E+01</td>
<td>0.9048375E+00</td>
<td>0.8196404E-07</td>
<td>0.8037068E-07</td>
</tr>
<tr>
<td>0.10</td>
<td>0.9048374E+00</td>
<td>0.8187309E+00</td>
<td>0.1483283E-06</td>
<td>0.7272241E-07</td>
</tr>
<tr>
<td>0.20</td>
<td>0.8187308E+00</td>
<td>0.7408184E+00</td>
<td>0.2013195E-06</td>
<td>0.6580196E-07</td>
</tr>
<tr>
<td>0.30</td>
<td>0.7408182E+00</td>
<td>0.6703203E+00</td>
<td>0.2428819E-06</td>
<td>0.5954008E-07</td>
</tr>
<tr>
<td>0.40</td>
<td>0.6703200E+00</td>
<td>0.6065309E+00</td>
<td>0.2747108E-06</td>
<td>0.5387410E-07</td>
</tr>
<tr>
<td>0.50</td>
<td>0.6065307E+00</td>
<td>0.5488119E+00</td>
<td>0.2982823E-06</td>
<td>0.4874731E-07</td>
</tr>
<tr>
<td>0.60</td>
<td>0.5488116E+00</td>
<td>0.4965856E+00</td>
<td>0.3148798E-06</td>
<td>0.4410839E-07</td>
</tr>
<tr>
<td>0.70</td>
<td>0.4965853E+00</td>
<td>0.4493293E+00</td>
<td>0.3256172E-06</td>
<td>0.3991093E-07</td>
</tr>
<tr>
<td>0.80</td>
<td>0.4493290E+00</td>
<td>0.4065700E+00</td>
<td>0.3314595E-06</td>
<td>0.3611290E-07</td>
</tr>
<tr>
<td>0.90</td>
<td>0.4065697E+00</td>
<td>0.3678798E+00</td>
<td>0.332411E-06</td>
<td>0.3267631E-07</td>
</tr>
<tr>
<td>1.00</td>
<td>0.3678794E+00</td>
<td>0.3678798E+00</td>
<td>0.333241E-06</td>
<td>0.3267631E-07</td>
</tr>
</tbody>
</table>
The two values are reasonably close to each other at the early stages which indicates that the error estimate is good. After several steps they become more separated because the exact error represents the global error of the method whereas the estimated error represents the error at each step.

5.2.5 Practical numerical example of step size control

We solve the problem

\[ y' = -y, \ y(0) = 1 \]

by the AGM procedure. The computer program "AGM" implements the AGM method given by equations (5.2.9). This program, given in Appendix 5, tests whether the estimated truncation error EST given by (5.2.12) exceeds a certain pre-set tolerable error \( e \). If this happens the routine halves the step size, recompute EST and test again. On the other hand if EST is smaller than \( \left( \frac{1}{4} \right)e \), we can safely double the step size. In this example, we run the program with \( e = 0.1 \times 10^{-5} \) and with initial step size set at \( h = 1.0 \).

The results, given in Table 5.6 show that the magnitude of the errors at each step which appear on the error column are well controlled and appear to be in the desired range, i.e.

\[ \frac{e}{32} < \text{error} < e. \]

In order to achieve this, the routine halves the step size several times until the required tolerance is satisfied by using \( h = 0.125 \).

In this routine, provision has to be given to avoid looping in the case when doubling the step size would cause the error to become larger than \( e \) and hence re-halving the step size again. In a case like this which may happen as a result of rounding error, we shall settle at using the smaller step size.
Table 5.6

Error Tolerance, $e = 0.1 \times 10^{-5}$

<table>
<thead>
<tr>
<th>$h$</th>
<th>$0.0000$</th>
<th>$0.1250$</th>
<th>$0.2500$</th>
<th>$0.5000$</th>
<th>$0.6250$</th>
<th>$0.7500$</th>
<th>$0.8750$</th>
<th>$1.0000$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h$</td>
<td>$0.0000$</td>
<td>$0.1000000E+01$</td>
<td>$0.8824972E+00$</td>
<td>$0.7788008E+00$</td>
<td>$0.6872899E+00$</td>
<td>$0.6065313E+00$</td>
<td>$0.5352622E+00$</td>
<td>$0.4723666E+00$</td>
</tr>
<tr>
<td>$x$</td>
<td>$0.0000$</td>
<td>$0.8824969E+00$</td>
<td>$0.7788008E+00$</td>
<td>$0.6872899E+00$</td>
<td>$0.6065313E+00$</td>
<td>$0.5352622E+00$</td>
<td>$0.4723666E+00$</td>
<td>$0.4168620E+00$</td>
</tr>
<tr>
<td>$x$</td>
<td>$0.1250$</td>
<td>$0.8824972E+00$</td>
<td>$0.7788012E+00$</td>
<td>$0.6820176E+00$</td>
<td>$0.6065312E+00$</td>
<td>$0.5352622E+00$</td>
<td>$0.4723666E+00$</td>
<td>$0.4168628E+00$</td>
</tr>
<tr>
<td>$x$</td>
<td>$0.2500$</td>
<td>$0.8824972E+00$</td>
<td>$0.7788012E+00$</td>
<td>$0.6820176E+00$</td>
<td>$0.6065312E+00$</td>
<td>$0.5352622E+00$</td>
<td>$0.4723666E+00$</td>
<td>$0.4168620E+00$</td>
</tr>
<tr>
<td>$x$</td>
<td>$0.5000$</td>
<td>$0.8824972E+00$</td>
<td>$0.7788012E+00$</td>
<td>$0.6820176E+00$</td>
<td>$0.6065312E+00$</td>
<td>$0.5352622E+00$</td>
<td>$0.4723666E+00$</td>
<td>$0.4168620E+00$</td>
</tr>
<tr>
<td>$x$</td>
<td>$0.6250$</td>
<td>$0.8824972E+00$</td>
<td>$0.7788012E+00$</td>
<td>$0.6820176E+00$</td>
<td>$0.6065312E+00$</td>
<td>$0.5352622E+00$</td>
<td>$0.4723666E+00$</td>
<td>$0.4168620E+00$</td>
</tr>
<tr>
<td>$x$</td>
<td>$0.7500$</td>
<td>$0.8824972E+00$</td>
<td>$0.7788012E+00$</td>
<td>$0.6820176E+00$</td>
<td>$0.6065312E+00$</td>
<td>$0.5352622E+00$</td>
<td>$0.4723666E+00$</td>
<td>$0.4168620E+00$</td>
</tr>
<tr>
<td>$x$</td>
<td>$0.8750$</td>
<td>$0.8824972E+00$</td>
<td>$0.7788012E+00$</td>
<td>$0.6820176E+00$</td>
<td>$0.6065312E+00$</td>
<td>$0.5352622E+00$</td>
<td>$0.4723666E+00$</td>
<td>$0.4168620E+00$</td>
</tr>
<tr>
<td>$x$</td>
<td>$1.0000$</td>
<td>$0.3678794E+00$</td>
<td>$0.3678803E+00$</td>
<td>$0.8307514E+00$</td>
<td>$0.1031159E+00$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.2.6 AGM methods of lower order

The defect of the Merson's type formula as a general measure for error control is that they are applicable only to the 4th order Runge-Kutta process, or presumably of order not less than 4. This follows from the fact that the 4-stage formula can have an order four but the highest order that can be achieved by a 5-stage formula is also four. The difference in the truncation error is then used to determine the error estimate.

The AGM strategy, on the other hand, uses the same number of stages to produce two approximations by the AM and GM formulae with similar order of accuracy, the difference of which then serves as a measure for determining the error estimate. Obviously this is possible with methods of any order. Therefore, to complete our analysis we derive in this section the 2nd and 3rd order AGM formulae.
2nd order AGM formula

The 2-stage 2nd order AGM formula takes the form,

\[
\begin{align*}
  k_1 &= f(y_n) \\
  k_2 &= f(y_n + h a_1 k_1) \\
  y_{n+1}^* &= y_n + h k_1 (1-p_k) \\
  y_{n+1}^{**} &= y_n + h[(1-w_2)k_1 + w_2 k_2]
\end{align*}
\]  

(5.2.13)

where the parameters \(a_1, p\) and \(w_2\) are still to be determined.

The equations of condition (5.1.30) and (5.2.7) then simply reduce to

\[
\begin{align*}
  h^2 f_y (GM): & \quad p a_1 = \frac{1}{3} \\
  (AM): & \quad w_2 a_1 = \frac{1}{3}
\end{align*}
\]  

(5.2.14)  

(5.2.15)

There are more than one possible solution to (5.2.14) and (5.2.15), one of the simplest being \(a_1 = 1, p = w_2 = \frac{1}{3}\). With this choice of parameters the truncation errors are given as follows

\[
\begin{align*}
  \text{LTE}_{GM} &= y(x_{n+1}^*) - y_{n+1}^* \\
  &= -\frac{1}{12} h^3 f_y^2 y_{yy} + \frac{7}{24} h^3 f_y^2 \\
  \text{LTE}_{AM} &= y(x_{n+1}^{**}) - y_{n+1}^{**} \\
  &= -\frac{1}{12} h^3 f_y^2 y_{yy} + \frac{1}{6} h^3 f_y^2
\end{align*}
\]  

(5.2.16)  

(5.2.17)

Assuming \(f_y^2 = 0\) (achieved when \(f_y^2 = 0\), i.e. \(y' = Ay\)) we have from (5.2.16) and (5.2.17),

\[
\begin{align*}
  y(x_{n+1}) - y_{n+1}^* &= \frac{7}{24} h^3 f_y^2 \\
  y(x_{n+1}) - y_{n+1}^{**} &= \frac{1}{6} h^3 f_y^2
\end{align*}
\]  

(5.2.18)  

(5.2.19)

Subtracting (5.2.19) from (5.2.18) we obtain

\[
\begin{align*}
  y_{n+1}^{**} - y_{n+1}^* &= \frac{1}{8} h^3 f_y^2 \\
  h^3 f_y^2 &= 8(y_{n+1}^{**} - y_{n+1}^*)
\end{align*}
\]

or

Therefore,

\[
\begin{align*}
  \text{LTE}_{AM} &= \frac{1}{6} h^3 f_y^2 \\
  &= \frac{1}{6} \cdot \frac{8}{3} (y_{n+1}^{**} - y_{n+1}^*) \\
  &= \frac{4}{3} (y_{n+1}^{**} - y_{n+1}^*)
\end{align*}
\]  

(5.2.20)
Thus, one of the second order AGM formula is given by

\[
\begin{align*}
    k_1 &= f(x_n, y_n) \\
    k_2 &= f(x_n + h, y_n + h k_1) \\
    y_{n+1} &= y_n + \frac{1}{2} h (k_1 + k_2)
\end{align*}
\]

(5.2.21)

with the error estimate given by

\[
    \text{EST} = \frac{4}{3} h [\frac{1}{2} (k_1 + k_2) - \sqrt{k_1 k_2}]
\]

(5.2.22)

which can be computed with negligible extra effort.

3rd order AGM method

In a similar way we can derive the 3-stage 3rd order AGM formula which will take the form

\[
\begin{align*}
    k_1 &= f(y_n) \\
    k_2 &= f(y_n + h a_1 k_1) \\
    k_3 &= f(y_n + h a_2 k_1 + h a_3 k_2) \\
    y_{n+1} &= y_n + h (1 - p q) k_1 + p q k_2 + q a_1 k_3
\end{align*}
\]

(5.2.23)

by solving the following six equations of condition taken from (5.1.30) and (5.2.7) for the parameters \( a_1, a_2, a_3, p, q, w_2, w_3 \).

GM

\[
\begin{align*}
    h^2 f_{y}^2 : & \quad p a_1 + q s_2 = \frac{1}{6} \\
    h^3 f_{y}^2 : & \quad p q a_1 s_2 + \frac{1}{6} p (p-1) a_1^2 + q a_1 a_3 \\
                      & \quad + \frac{1}{6} q (q-1) s_2^2 = 1/6
\end{align*}
\]

(5.2.24-i, 5.2.24-ii)

AM

\[
\begin{align*}
    h^2 f_{y} : & \quad w_2 a_1 + w_3 s_2 = \frac{1}{6} \\
    h^3 f_{y}^2 : & \quad w_3 a_1 a_3 = 1/6 \\
    h^3 f_{yy}^2 : & \quad w_2 a_1^2 + w_3 s_2^2 = 1/6
\end{align*}
\]

(5.2.24-iv, 5.2.24-v, 5.2.24-vi)
If the solution to the system (5.2.24) exists, it may not be unique since the system is nonlinear and, in addition, there are more variables than equations. For example by introducing another constraint,

\[ a_1 = a_2 + a_3 \]  
(5.2.24-vii)

we obtain as one of the solutions, the following,

\[ a_1 = \frac{2}{3}, \quad a_2 = \frac{1}{3}, \quad a_3 = \frac{1}{3}, \quad p = -\frac{3}{16}, \]
\[ q = \frac{15}{16}, \quad w_2 = 0, \quad w_3 = \frac{3}{4}. \]

The error estimate for this formula is obtained as follows:

By substituting the values of the parameters into equation (5.1.30-v) we have

\[ \text{LTE}_{\text{GM}} = y(x_{n+1}) - y_{n+1} = 0.064814814h^4f_y^3 \]  
(5.2.25)

and by substituting into (5.2.7-v) we obtain

\[ \text{LTE}_{\text{AM}} = y(x_{n+1}) - y_{n+1} = \frac{1}{24} h^4f_y^3 \]  
(5.2.26)

where we have assumed, as before, \( f_{yy}, f_{yyy}, \ldots \) are all zero.

On subtracting (5.2.26) from (5.2.25) we obtain

\[ y_{n+1}^{**} - y_{n+1}^{*} = 0.023148143h^4f_y^3, \]

or

\[ h^4f_y^3 = (y_{n+1}^{**} - y_{n+1}^{*})/0.023148143. \]

Thus, from (5.2.24) we have

\[ \text{LTE}_{\text{AM}} = (y_{n+1}^{**} - y_{n+1}^{*})/(24 \times 0.023148143) \]
\[ = (y_{n+1}^{**} - y_{n+1}^{*})/0.55555544 \]
\[ = 1.8(y_{n+1}^{**} - y_{n+1}^{*}). \]  
(5.2.27)

Thus, one of the third order AGM formula is given by

\[
\begin{align*}
  k_1 &= f(x_n, y_n) \\
  k_2 &= f(x_n + \frac{2}{3} h, y_n + \frac{2}{3} h k_1) \\
  k_3 &= f(x_n + \frac{2}{3} h, y_n + \frac{1}{3} h k_1 + \frac{1}{3} h k_2) \\
  y_{n+1} &= y_n + \frac{h}{4} (k_1 + 3k_3)
\end{align*}
\]  
(5.2.28)
with the error estimate given by

\[ \text{EST} = 1.8(y_{n+1} - y_n - h k_1^{1/4} k_2^{-3/16} k_3^{15/16}) \]  \hspace{1cm} (5.2.29)
5.3 GM RUNGE-KUTTA FEHLBERG METHOD

5.3.1 Runge-Kutta methods of Fehlberg variation

One popular variable-step Runge-Kutta method is based on a pair of fourth- and fifth-order methods attributed to Fehlberg. Fehlberg [1969] derives the first to the fourth-order formulae with step size control for solving

\[ y' = f(x,y), \quad y(x_0) = y_0. \]  

(5.3.1)

Initially 4th order formulae are developed of the form,

\[
\begin{align*}
  k_1 &= f(x_0, y_0) \\
  k_i &= f(x_0 + h\alpha_i, y_0 + h\sum_{j=1}^{i-1} \beta_{ij} k_j), \\
  i &= 2, 3, 4, 5, 6
\end{align*}
\]

(5.3.2)

and

\[
\begin{align*}
  y &= y_0 + h \sum_{i=1}^{5} c_i k_i + O(h^5) \quad (5.3.3-i) \\
  y^* &= y_0 + h \sum_{i=1}^{6} c_i^* k_i + O(h^6) \quad (5.3.3-ii)
\end{align*}
\]

where \( h \) is the integration step size.

Equations (5.3.3) imply that we try to determine the coefficients \( \alpha_i, \beta_{ij}, c_i, c_i^* \) in such a way that the first formula (5.3.3-i) represents a 4th order, and the second formula (5.3.3-ii), a 5th order formula. The difference \( y - y^* \) then represents an approximation for the leading (5th order) truncation error term of the 4th order Runge-Kutta formula (5.3.3-i) and can be easily used for establishing a reliable step size control procedure for this formula.

In determining these coefficients, which have to satisfy certain equations of condition, Fehlberg made the attempt to reduce the truncation error of his 4th order formula by making a proper choice of
the available free (arbitrary) parameters. This truncation error
consists of nine sub-terms of the form,

$$T_1V_1 + T_2V_2 + \ldots + T_9V_9$$  \hspace{1cm} (5.3.4)

where $V_1, V_2, \ldots, V_9$ are certain expressions built up by the partial
derivatives of the RHS of the differential equation (5.3.1), and $T_1$
through $T_9$ are termed as numerical factors. Thus reducing the error is
equivalent to reducing these numerical factors to as small values as
possible, probably by a measure of norms.

He gives 4th order formulae in the paper, one of which, being the
more accurate, is given by the augmented symbol generating matrix $F_4$
(where the last two columns indicate the values of $c_i$ and $c_i^*$).

$$
F_4 = \begin{bmatrix}
0 & 25 & 16 \\
1/4 & 216 & 135 \\
3/32 & 0 & 0 \\
9/32 & 1408 & 6656 \\
1932 & 7200 & 7296 \\
2197 & 2197 & 2197 \\
439 & -8 & 3680 \\
216 & 7296 & 2197 \\
-8/27 & 2 & 3544 \\
& 2565 & 1859 \\
& 4104 & 11 \\
& 40 & 2 \\
& & 55
\end{bmatrix}$$  \hspace{1cm} (5.3.5)

Fehlberg's two formulae were found to be better in accuracy, and
hence operate more economically than the Sarafyan's RK4(5) formula
(Sarafyan [1966]), which is given by the following symbol generating
matrix.
Fehlberg further developed the 3rd order formulae, RK3(4), by using five function evaluations. (It is impossible to use only four function evaluations since that will lead to a contradiction in the equations of condition). However the fifth evaluation is chosen in such a way that this evaluation can be taken over as the first evaluation for the next step. Thereby the number of evaluations per step again will be reduced to four, except for the very first step, when the integration is started. One of his 3rd order formulae is given below by the augmented symbol generating matrix \( F_3 \).

\[
\begin{bmatrix}
0 & \frac{1}{6} & \frac{1}{24} \\
\frac{1}{2} & 0 & 0 \\
\frac{1}{4} & \frac{1}{4} & 2 \\
\frac{2}{27} & \frac{10}{27} & 0 & \frac{1}{27} & \frac{27}{56} \\
\frac{28}{625} & -\frac{1}{5} & \frac{546}{625} & \frac{54}{625} & -\frac{378}{625} & \frac{125}{336}
\end{bmatrix}
\]

\( S_4 = \)

\[
\begin{bmatrix}
0 & -1 & 2 & \frac{1}{6} & \frac{5}{48} \\
\frac{2}{27} & \frac{10}{27} & 0 & \frac{1}{27} & \frac{27}{56} \\
\frac{28}{625} & -\frac{1}{5} & \frac{546}{625} & \frac{54}{625} & -\frac{378}{625} & \frac{125}{336}
\end{bmatrix}
\]

(5.3.6)

\[
F_3 = \begin{bmatrix}
0 & 79/490 & 229/1470 \\
2/7 & 0 & 0 \\
77/900 & 343/900 & 2175/3626 & 1125/1813 \\
805/1444 & -77175/54872 & 97125/54872 & 2166/9065 & 13715/81585 \\
79/490 & 0 & 2175/3626 & 2166/9065 & 1/18
\end{bmatrix}
\]

(5.3.7)
Note that in this matrix the entries of the 5th column are identical to the first four entries of the last row, ensuring that \( k_1 \) of the next step is similar to \( k_5 \) of the present step. In addition, Fehlberg also derived the 2nd and 1st order formulae by using four and three function evaluations respectively.

5.3.2 Derivation of GM RK3(4) formula

We will now try to derive the GM RK3(4) formula by using only four function evaluations if possible. We will resort to using five evaluations only when any equation of condition is proved to contradict another. The proposed formula will be of the form,

\[
\begin{align*}
    k_1 &= f(y_n) \\
    k_2 &= f(y_n + ha_1k_1) \\
    k_3 &= f(y_n + ha_2k_1 + ha_3k_2) \\
    k_4 &= f(y_n + ha_4k_1 + ha_5k_2 + ha_6k_3)
\end{align*}
\]

\[
\begin{align*}
    y_{n+1} &= y_n + h(k_1 - \gamma - \delta k_2 + k_3) + O(h^4) \hspace{1cm} (5.3.9-i) \\
    y_{n+1} &= y_n + h(1 - p - q - r)k_1 + p k_2 + q k_3 + r k_4 + O(h^5) \hspace{1cm} (5.3.9-ii)
\end{align*}
\]

The coefficients \( a_1, a_2, a_3, a_4, a_5, a_6, p, q, r, \gamma \) and \( \delta \) have to satisfy the related equations of condition. The six equations of condition for the 4th order formula (5.3.9-ii) are already obtained from the derivation of the 4th order GM Runge-Kutta formulae given by equations (5.1.30) and are reproduced here in equations (5.3.10-i) through (5.3.10-vi). Three more equations of condition for the 3rd order formula (5.3.9-i) are similar to the equations given by (5.1.18) and are reproduced here in equations (5.3.10-vii) through (5.3.10-ix). These nine equations
have to be solved simultaneously for the eleven unknown parameters, thus giving us two degrees of freedom. The equations are given below, where \( s_2 = a_2 + a_3 \) and \( s_3 = a_4 + a_5 + a_6 \). The Roman numerals to the left of the equations indicate the order of the terms in the Taylor expansion.

\[
\begin{align*}
\text{II} & \quad p a_1 + q s_2 + r s_3 = \frac{1}{6} \quad (5.3.10-1) \\
\text{III} & \quad \frac{1}{6} p a_1^2 + \frac{1}{6} q s_2^2 + \frac{1}{6} r s_3^2 = \frac{1}{12} \quad (5.3.10-11) \\
\text{III} & \quad r(a_5 a_1 + a_6 s_2) + \frac{1}{6} r(r-1)s_3^2 + (p a_1 + q s_2) s_3 + p q a_1 s_2 \\
& \quad + \frac{1}{6} p(p-1)a_1^2 + q a_1 s_3 + \frac{1}{6} q(q-1)s_2^2 = \frac{1}{6} \quad (5.3.10-11) \\
\text{IV} & \quad \frac{1}{6} p a_1^3 + \frac{1}{6} q s_2^3 + \frac{1}{6} r s_3^3 = \frac{1}{24} \quad (5.3.10-11) \\
\text{IV} & \quad r a_6 a_3 + r(r-1)s_3(a_5 a_1 + a_6 s_2) \\
& \quad + \frac{1}{6} r(r-1)(r-2)s_3^2 + q(q-1)a_1 a_3 s_2 + \frac{1}{6} q(q-1)(q-2)s_2^3 \\
& \quad + p q a_1^2 a_3 + \frac{1}{6} p q(q-1)a_1 s_2^2 + \frac{1}{6} p(p-1)q a_1^2 s_2 \\
& \quad + \frac{1}{6} p(p-1)(p-2)a_1^3 + (p q a_1 s_3 + \frac{1}{6} p(p-1)a_1^2 s_2 \\
& \quad + qa_1 a_3 + \frac{1}{6} q(q-1)s_2^2) s_3 + [r(a_5 a_1 + a_6 s_2) \\
& \quad + \frac{1}{6} r(r-1)s_3^2](p a_1 + q s_2) = \frac{1}{24} \quad (5.3.10-v) \\
\text{IV} & \quad r(\frac{1}{6} a_5 a_1^2 + \frac{1}{6} a_6 s_2^2) + r s_3(a_5 a_1 + a_6 s_2) + \frac{1}{6} r(r-1)s_3^3 \\
& \quad + \frac{1}{6} r s_3^3(p a_1 + q s_2) + (\frac{1}{6} p a_1^2 + \frac{1}{6} q s_2^2) s_3 + \frac{1}{6} q a_1^2 a_3 \\
& \quad + qa_1 a_3 s_2 + \frac{1}{6} q(q-1)s_2^3 + \frac{1}{6} p q a_1 s_2^2 + \frac{1}{6} p q a_1^2 s_2 \\
& \quad + \frac{1}{6} p(p-1)a_1^3 = \frac{1}{6} \quad (5.3.10-vi) \\
\text{II} & \quad \gamma a_1 + \delta s_2 = \frac{1}{6} \quad (5.3.10-vii) \\
\text{III} & \quad \gamma a_1^2 + \delta s_2^2 = \frac{1}{6} \quad (5.3.10-viii) \\
\text{III} & \quad \gamma a_1 s_2 + \gamma (\gamma-1)a_1^2 + \delta a_1 a_3 + \frac{1}{6} (\delta-1)s_2^2 = \frac{1}{6} \quad (5.3.10-ix)
\end{align*}
\]

We observe from equations (5.3.10-vii) and (5.3.10-viii) that given the values of \( a_1 \) and \( s_2 \) (\( \neq a_1 \)) we can uniquely determine \( \gamma \) and \( \delta \) and then from equation (5.3.10-ix) we can obtain the value of \( a_3 \).
Having obtained the values of $a_1$, $a_2$, and $a_3$, the next step is to solve from the remaining six equations (5.3.10-i) through (5.3.10-vi) for the still unknown parameters $a_4$, $a_5$, $a_6$, $p$, $q$, and $r$. Note that $\gamma$ and $\delta$ are not related to these equations.

As suggested earlier, we try with some predetermined values of $a_1$ and $s_2$, and hence values of $a_1$, $a_2$, and $a_3$. The NAG routine C05NBF is used to solve these equations numerically, and it is found that some choice of $a_1$ and $s_2$ exists that lead to a convergent set of solutions. Therefore, unlike in the AM case, four function evaluations are sufficient to define the GM RK3(4) pair. One set of solution is obtained as follows.

Let us now set $a_1=1/2$, $s_2=1$. From equations (5.3.10-vii) and (5.3.10-viii) we obtain $\gamma=2/3, \delta=1/6$, and further from (5.3.10-ix) we have $a_3=5/2$. Substituting these values into equations (5.3.10-i) through (5.3.10-vi) and solve, we obtain the following set of solution.

\[
\begin{align*}
  a_1 &= 1/2 \\
  a_2 &= -3/2 \\
  a_3 &= 5/2 \\
  a_4 &= 0.5751743896D+00 = 9235/16056 \\
  a_5 &= -0.1314740754D+00 = -9994/76015 \\
  a_6 &= -0.3027237237D-02 = -341/112644 \\
  p &= 0.4258238789D+01 = 7882/1851 \\
  q &= -0.1290184351D-02 = -12/9301 \\
  r &= -0.3705672148D-02 = -8297/2239 \\
  \gamma &= 2/3 \\
  \delta &= 1/6 \\
  \text{res} &= 0.5836739756D-17.
\end{align*}
\]

The numerical factors $T_1$, $T_2$, and $T_3$ are obtained by evaluating the 4th order coefficients given by equations (5.3.10-iv, v, vi) with $p, q, r$ taking the values of $\gamma$, $\delta$ and $0$ respectively. They are given by $T_1=-0.77777778$, $T_2=-0.78549381$, $T_3=1.3298611$. 


Thus, one of the GM variation of the RK3(4) formulae of Fehlberg type is given by

\[ k_1 = f(x_n, y_n) \]
\[ k_2 = f(x_n + \frac{1}{2} h, y_n + \frac{1}{2} h k_1) \]
\[ k_3 = f(x_n + \frac{3}{2} h, y_n - \frac{3}{2} h k_1 + \frac{5}{2} h k_2) \]
\[ k_4 = f(x_n + \frac{255}{256} h, y_n + \frac{1065}{256} h k_1 - \frac{6988}{256} h k_2 + \frac{6872}{256} h k_3) \] (5.3.11)

\[ y_{n+1} = y_n + h \left( k_1 \frac{1}{6} + k_2 \frac{2}{3} + k_3 \frac{1}{6} \right) \]

** EST = y_{n+1}^* - y_{n+1}^**

By setting different values of \(a_1\) and \(a_2\) we can obtain a different set of solutions. As for example, by setting \(a_1 = \frac{1}{4}\) and \(a_2 = \frac{3}{4}\), we obtain the following set of solutions,

\[ a_1 = 1/2 \]
\[ a_2 = -3/16 \]
\[ a_3 = 15/16 \]
\[ a_4 = 0.4940747147D+00 = 9881/19999 \]
\[ a_5 = -0.2538711070D+00 = -4148/16339 \]
\[ a_6 = -0.3526533605D-01 = -3763/22009 \]
\[ p = 0.2078103207D+01 = 1490/717 \]
\[ q = -0.3481757436D+00 = -7663/22009 \]
\[ r = -0.3267410730D+01 = -7319/2240 \]
\[ y = 1/3 \]
\[ \delta = 4/9 \]
\[ \text{res} = 0.1846784363D-18 \]
\[ T_1 = -0.038628472 = -89/2304 \]
\[ T_2 = 0.042534722 = 49/1152 \]
\[ T_3 = -0.0201822920 = -155/768. \]
It is thought that it would be possible to obtain a different set of parameters that will minimise the root mean squares value of $T_1$, $T_2$ and $T_3$ which will in turn give rise to the most accurate method. However we will not investigate it in this thesis and leave it for future research.

Numerical Example

We use formula (5.3.11) to integrate $y' = -y$, $y(0) = 1$, in the interval $[0,1]$, and require that the truncation error should not exceed $10^{-4}$. The results are shown in Table 5.7. This formula chooses a stepsize of length $h=0.0625$ and takes about 0.083 sec. to perform the integration. The corresponding AM formula as given by (5.3.7) takes a stepsize of length $h = 0.25$ to satisfy the accuracy requirement and therefore takes shorter computing time, i.e. about 0.016 sec., to perform the integration. Therefore the GM formula is about five times as slow as the AM formula. We expect, this is due to the larger truncation error in the 3rd order GM formula. This suggests that a more accurate GM formula will make it more competitive.

It is interesting to note that although the Fehlberg 3(4) formula is a 5-stage formula, it is essentially a 4-stage formula from the computational point of view when considering $k_1$ is not to be evaluated except in the very first step. In contrast this cannot be the case with a GM formula of the same kind in that $k_1$ for the next step cannot simply be taken from the last evaluation of the previous step. Thus using more function evaluations than necessary is not an advantage, except for the purpose of reducing the error in which case it will become less favourable than the corresponding AM formula in terms of
Table 5.7

Error tolerance = 10^{-4}

(a) Results by using the GM RK3(4) formula (5.3.11)

\[ h = 0.50000 \]
\[ h = 0.25000 \]
\[ h = 0.12500 \]
\[ h = 0.06250 \]

<table>
<thead>
<tr>
<th>( x )</th>
<th>Exact Solution</th>
<th>Numerical Solution</th>
<th>Error</th>
<th>Estimated Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.06250</td>
<td>0.9394131E+00</td>
<td>0.9393837E+00</td>
<td>-0.2934280E-04</td>
<td>-0.2747496E-04</td>
</tr>
<tr>
<td>0.12500</td>
<td>0.8824969E+00</td>
<td>0.8824418E+00</td>
<td>-0.5512917E-04</td>
<td>-0.2580953E-04</td>
</tr>
<tr>
<td>0.18750</td>
<td>0.8290291E+00</td>
<td>0.829514E+00</td>
<td>-0.7768237E-04</td>
<td>-0.2424506E-04</td>
</tr>
<tr>
<td>0.25000</td>
<td>0.7788008E+00</td>
<td>0.7787035E+00</td>
<td>-0.9729960E-04</td>
<td>-0.2277541E-04</td>
</tr>
<tr>
<td>0.31250</td>
<td>0.7316156E+00</td>
<td>0.7315014E+00</td>
<td>-0.1142539E-03</td>
<td>-0.2139485E-04</td>
</tr>
<tr>
<td>0.37500</td>
<td>0.6872893E+00</td>
<td>0.6871605E+00</td>
<td>-0.1287959E-03</td>
<td>-0.2009797E-04</td>
</tr>
<tr>
<td>0.43750</td>
<td>0.6456485E+00</td>
<td>0.6455074E+00</td>
<td>-0.1411557E-03</td>
<td>-0.1887971E-04</td>
</tr>
<tr>
<td>0.50000</td>
<td>0.6065307E+00</td>
<td>0.6063791E+00</td>
<td>-0.1515445E-03</td>
<td>-0.1773529E-04</td>
</tr>
<tr>
<td>0.56250</td>
<td>0.5697828E+00</td>
<td>0.5696227E+00</td>
<td>-0.1601558E-03</td>
<td>-0.1666024E-04</td>
</tr>
<tr>
<td>0.62500</td>
<td>0.5352614E+00</td>
<td>0.5350943E+00</td>
<td>-0.1671668E-03</td>
<td>-0.1565036E-04</td>
</tr>
<tr>
<td>0.68750</td>
<td>0.5028316E+00</td>
<td>0.5026588E+00</td>
<td>-0.1727398E-03</td>
<td>-0.1470170E-04</td>
</tr>
<tr>
<td>0.75000</td>
<td>0.4723666E+00</td>
<td>0.4721895E+00</td>
<td>-0.1770234E-03</td>
<td>-0.1381053E-04</td>
</tr>
<tr>
<td>0.81250</td>
<td>0.4437473E+00</td>
<td>0.4435672E+00</td>
<td>-0.1801535E-03</td>
<td>-0.1297339E-04</td>
</tr>
<tr>
<td>0.87500</td>
<td>0.4168620E+00</td>
<td>0.4166798E+00</td>
<td>-0.1822541E-03</td>
<td>-0.1218699E-04</td>
</tr>
<tr>
<td>0.93750</td>
<td>0.3916056E+00</td>
<td>0.3914222E+00</td>
<td>-0.1834384E-03</td>
<td>-0.1144826E-04</td>
</tr>
<tr>
<td>1.00000</td>
<td>0.3678794E+00</td>
<td>0.3676956E+00</td>
<td>-0.1838099E-03</td>
<td>-0.1075431E-04</td>
</tr>
</tbody>
</table>

Total computing time = 83661 \mu-sec.

(b) Results by using the AM RK3(4) formula (5.3.7)

\[ h = 0.50000 \]
\[ h = 0.25000 \]

<table>
<thead>
<tr>
<th>( x )</th>
<th>Exact Solution</th>
<th>Numerical Solution</th>
<th>Error</th>
<th>Estimated Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25000</td>
<td>0.7788008E+00</td>
<td>0.7788257E+00</td>
<td>-0.2494336E-04</td>
<td>-0.1963119E-04</td>
</tr>
<tr>
<td>0.50000</td>
<td>0.6065307E+00</td>
<td>0.6065695E+00</td>
<td>-0.3885243E-04</td>
<td>-0.1528928E-04</td>
</tr>
<tr>
<td>0.75000</td>
<td>0.4723666E+00</td>
<td>0.4724119E+00</td>
<td>-0.4538818E-04</td>
<td>-0.1190768E-04</td>
</tr>
<tr>
<td>1.00000</td>
<td>0.3678794E+00</td>
<td>0.3679266E+00</td>
<td>-0.4713189E-04</td>
<td>-0.9274009E-05</td>
</tr>
</tbody>
</table>

Total computing time = 16543 \mu-sec.
computing time. Thus, future work on this area should aim at finding the parameters that will produce a method with the least possible error with minimum number of stages.

5.3.3 GM Runge-Kutta RK2(3) formula

We will try to use three evaluations per step for a GM RK2(3) formula of the form

\[
\begin{align*}
    k_1 &= f(y_n) \\
    k_2 &= f(y_n + ha_1 k_1) \\
    k_3 &= f(y_n + ha_2 k_1 + ha_3 k_2)
\end{align*}
\]

\[y_{n+1}^* = y_n + \frac{1}{2} k_2 + \frac{1}{2} k_3 + O(h^3)\]  

\[y_{n+1}^{**} = y_n + k_1 - y + \frac{1}{2} p q + O(h^4)\]

In a similar way as in equations (5.3.10) the equations of condition for the third order formula is given by, (where \(s_2 = a_2 + a_3\)),

\[
\begin{align*}
    \text{II} & \quad p a_1 + q s_2 = \frac{1}{6} \\
    \text{III} & \quad t p a_1^2 + t q s_2^2 = \frac{1}{6} \\
    \text{III} & \quad p q a_1 s_2 + t p (p-1) a_1^2 + q a_1 a_3 + t q (q-1) s_2^2 = \frac{1}{6}
\end{align*}
\]

In addition, the second order formula must satisfy

\[\text{II} \quad \gamma a_1 = \frac{1}{6}\]

The numerical factors \(T_1\) and \(T_2\) are obtained from equations (5.3.15-ii) and (5.3.15-iii) by putting \(p=\gamma\) and \(q=0\), i.e.

\[
\begin{align*}
    T_1 &= \frac{1}{6} \gamma a_1^2 - \frac{1}{6} \\
    T_2 &= \frac{1}{6} \gamma (\gamma-1) a_1^2 - \frac{1}{6}.
\end{align*}
\]
In solving the nonlinear system (5.3.15) let us find the solution that will minimise the L-2 norm of the numerical factors, i.e. solution that minimises

$$\phi = T_1^2 + T_2^2$$

with the constraint \( y_{a_1} = \frac{1}{4} \). Substituting the corresponding expressions for \( T_1 \) and \( T_2 \) into (5.3.16) we obtain

$$\phi = \left( \frac{1}{4} a_1 - \frac{1}{6} \right)^2 + \left( -\frac{1}{4} a_1 - \frac{1}{24} \right)^2$$

and $$\frac{\partial \phi}{\partial a_1} = \frac{1}{4} a_1 - \frac{1}{16}$$

Therefore \( \phi \) is a minimum when \( a_1 = 1/4 \). Next, we let \( s_2 = \frac{1}{4} \) (other values may also be used). Then from (5.3.15-i) and (5.3.15-ii) we have \( p = -4/3 \) and \( q = 5/3 \), and from (5.3.15-iii) we obtain \( a_3 = \frac{1}{4} \). Since \( s_2 = a_2 + a_3 \) we have \( a_2 = 0 \) and \( T_1 = T_2 = -5/48 \).

Thus, a GM RK2(3) formula with minimum truncation error (in the sense of the L-2 norm) is given by the symbol generating matrix

$$G_2 = \begin{bmatrix} 0 & 0 & -1 & 2/3 \\ 1/4 & 0 & 2 & -4/3 \\ 0 & 1/2 & 5/3 \end{bmatrix}$$

with \( T_1 = T_2 = -5/48 \), in which case

$$\phi_{GM} = T_1^2 + T_2^2 = 50/2304 = 25/1152.$$  

For the purpose of comparison, the AM RK2(3) formula which is based on three evaluations is given by the symbol generating matrix \( F_2 \), where

$$F_2 = \begin{bmatrix} 0 & 0 & \frac{1}{4} & \frac{1}{6} \\ 1 & 0 & \frac{1}{2} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{4} & \frac{2}{3} \end{bmatrix}$$
with $T_1 = -1/6$, $T_2 = 1/12$ (see also Fehlberg [1969], p.28). In this case,

$$\rho_{AM} = T_1^2 + T_2^2 = 5/144 = 25/720$$

which is obviously greater than $\rho_{GM}$. For this reason we may say that the GM RK2(3) formula is more accurate than the corresponding Fehlberg's RK2(3) formula.

Numerical Example

We solve the IVP $y' = -y$, $y(0) = 1$, for $0 \leq x \leq 1.0$ by using the GM and AM formulae represented respectively by $G_2$ and $F_2$, and require that the truncation error should not exceed $10^{-5}$. In this example, both methods chose a stepsize of length $h = 1/32$. The results are more accurate in the case of the GM formula. This agrees with the fact that $T_1$ and $T_2$ have smaller magnitudes for the GM formula than in the AM formula.

<table>
<thead>
<tr>
<th>$h$</th>
<th>$x$</th>
<th>$x$ Exact Solution</th>
<th>$x$ Numerical Solution</th>
<th>$x$ Error</th>
<th>$x$ Estimated Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.03125</td>
<td>0.03125</td>
<td>0.9692332E+00</td>
<td>0.9692364E+00</td>
<td>-0.3139425E-05</td>
<td>-0.3178980E-05</td>
</tr>
<tr>
<td>0.25000</td>
<td>0.25000</td>
<td>0.7788008E+00</td>
<td>0.7788210E+00</td>
<td>-0.2018102E-04</td>
<td>-0.2554440E-05</td>
</tr>
<tr>
<td>0.50000</td>
<td>0.50000</td>
<td>0.6065307E+00</td>
<td>0.6065621E+00</td>
<td>-0.3143440E-04</td>
<td>-0.1989452E-05</td>
</tr>
<tr>
<td>0.75000</td>
<td>0.75000</td>
<td>0.4723666E+00</td>
<td>0.4724033E+00</td>
<td>-0.3672217E-04</td>
<td>-0.1549427E-05</td>
</tr>
<tr>
<td>1.00000</td>
<td>1.00000</td>
<td>0.3678794E+00</td>
<td>0.3679176E+00</td>
<td>-0.3813284E-04</td>
<td>-0.1206726E-05</td>
</tr>
</tbody>
</table>

Total computing time = 131044 µ-sec.

continued...
(b) Results by using AM RK2(3) formula

\[ h = 0.50000 \]
\[ h = 0.25000 \]
\[ h = 0.12500 \]
\[ h = 0.06250 \]
\[ h = 0.03125 \]

<table>
<thead>
<tr>
<th>( h )</th>
<th>Exact Solution</th>
<th>Numerical Solution</th>
<th>Error</th>
<th>Estimated Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.03125</td>
<td>0.9692332E+00</td>
<td>0.9692383E+00</td>
<td>-0.5046774E-05</td>
<td>-0.5086263E-05</td>
</tr>
<tr>
<td>0.25000</td>
<td>0.7788008E+00</td>
<td>0.7788332E+00</td>
<td>-0.3244216E-04</td>
<td>-0.4087076E-05</td>
</tr>
<tr>
<td>0.50000</td>
<td>0.6065307E+00</td>
<td>0.6065812E+00</td>
<td>-0.5053302E-04</td>
<td>-0.3183151E-05</td>
</tr>
<tr>
<td>0.75000</td>
<td>0.4723666E+00</td>
<td>0.4724256E+00</td>
<td>-0.5903396E-04</td>
<td>-0.2479144E-05</td>
</tr>
<tr>
<td>1.00000</td>
<td>0.3678794E+00</td>
<td>0.3679407E+00</td>
<td>-0.5130220E-04</td>
<td>-0.1930839E-05</td>
</tr>
</tbody>
</table>

Total computing time = 109304 \( \mu \)-sec.

5.3.4 Comments For Further Investigation

The fourth order method derived by Fehlberg uses six function evaluations to produce \( y_{n+1} \) and an estimate of the local error. By contrast an unrelated pair of fourth- and fifth-order methods would usually require four evaluations for \( y_n^* \) and six more evaluations for \( y_{n+1}^* \). This makes it obvious how the choice of parameters in the evaluation of functions have contributed to the time saving in the computational process. There are, however, Runge-Kutta pairs that are similar to F4 but which have higher orders. For example, the subroutine DVERK in the IMSL library is a pair of the fifth- and sixth-order methods requiring only eight evaluations per step to produce \( y_{n+1}^* \) and \( y_{n+1}^{**} \) (see, for example, Johnson and Riess [1982], p.379). Thus, in producing a method that can be used to solve the problem as economically as possible, we are frequently confronted with two crucial opposing factors of accuracy and computational complexity.

The prospect of making the GM technique a competitive one is not impossible. The experience of using the GM formula as opposed to the
Trapezoidal formula, and later in the step size control as in the Merson's formula, the insight of how the GM approach can actually be made competitive was obtained. The GM2(3) formula has already shown that its performance is more accurate than the corresponding RK2(3) formula. It is thought possible that the corresponding GM4(5) formula which is equivalent in form to the AM formula developed by Fehlberg could be obtained. The derivation of such formula and of the higher order ones would depend on our ability to solve the highly nonlinear algebraic equations which would occur. This will eventually require us to use a more versatile general purpose routine. Our experience shows that the usage of the NAG library routine C05NBF frequently ends up with a non-convergent or non-existent solution.
5.4 GM VERSION OF THE ITERATIVE MULTISTEP (IMS) FORMULA

5.4.1 Iterative Multistep Methods

Hyman [1978] developed a class of new methods to solve multirate systems of ordinary differential equations. These methods, called iterative multistep (IMS) methods, are extensions of standard explicit Runge-Kutta and multistep predictor-corrector methods. They are A-stable when iterated to convergence and converge to the exact solution for linear autonomous systems of equations. One of the most common multirate systems, on which the development of IMS methods was based, arises in the method of lines (MOL) approximation of partial differential equations.

An example of a common iterative method is the improved Euler method with the

\[
\text{predictor: } y_{n+1}^{(1)} = y_n + hf_n \quad (5.4.1)
\]

and

\[
\text{corrector: } y_{n+1}^{(i)} = y_n + \frac{1}{2}h[f_n + f_{n+1}^{(i-1)}] \quad (5.4.2)
\]

for \(i=2,3,4,\ldots\). Here \(n+1\) refers to time \(t_{n+1}\) and \(i\) is the iteration index. As we have explained in Section 4.3, the stability of the improved Euler method increases for the first few iterations as seen in Figure 4.3. After three iterations, the stability stagnates to the restriction \(\lambda_{\text{max}}|h| \leq 2\). Also, even when the method does converge, it converges to a solution of the difference equation not the differential equation.

The IMS methods were developed on the premise that if we are willing to do extra work by iterating, then it is not unreasonable to
expect the stability and accuracy to improve on each and every iteration. These methods based on the recursion relation

\[ y_{n+1}^{(i)} = y_n + c_i h (f_{n+1}^{(i-1)} - f_{n+1}^{(i-2)}) \]  

(5.4.3)

for \( i = 3, 4, \ldots \). That is, after the corrector cycle \( i=2 \) a different corrector is used for each additional iteration. The constants \( c_i \) depend on the iteration count and the predictor-corrector method used to start the process. The \( c_i \) are chosen to increase the order of accuracy of the method for linear autonomous systems at each iteration. Hence, when iterated an infinite number of times (or to convergence) the method is of infinite order and converges to the exact solution.

The simplest IMS indicated by Hyman is based on the improved Euler formula. This is given by,

\[ y_{n+1}^{(1)} = y_n + h f_n \]  

(5.4.4-i)

\[ y_{n+1}^{(2)} = y_n + \frac{1}{2} h [f_n + f_n^{(1)}] \]  

(5.4.4-ii)

\[ y_{n+1}^{(i)} = y_{n+1}^{(i-1)} + c_i h [f_{n+1}^{(i-1)} - f_{n+1}^{(i-2)}] \]  

(5.4.4-iii)

where \( c_i = \frac{1}{i} \) for \( i = 3, 4, \ldots k \).

The stability of this method increases with each iteration. In fact, for the first four iterations these regions are equivalent to the stability regions of a Runge-Kutta method shown in Figure 3.5.

Other methods where the coefficients of the IMS methods have been derived are based on the Adams-Bashforth-Moulton predictor-corrector sequence, polynomial extrapolation with a backward difference corrector and a leap frog or centered difference predictor-corrector sequence. Each method has a unique corrector sequence and different stability
regions. For example the IMS method which is based on the second order leap-frog predictor and a third order corrector may take the following form:

\[ y_{n+1}^{(1)} = y_{n-1} + 2hf_n \]
\[ y_{n+1}^{(2)} = \frac{4}{5} y_n + \frac{1}{5} y_{n-1} + \frac{1}{5} h[4f_n + 2f_n^{(1)}] \]
\[ y_{n+1}^{(1)} = y_n + c_1 h[f_n^{(1)} - f_{n+1}^{(2)}] \]

where \( c_3 = 3/10, c_4 = 7/30, c_5 = 4/21 \), etc.

The stability regions for these formulae are found to be particularly good along the imaginary axis, and thus are more suitable for systems which normally arise in the MOL approximation of hyperbolic partial differential equations. (See Hyman [1979]).

As we have seen here, a different predictor-corrector pair lead to a different stability regions of the IMS formula. These stability regions can be used as a guide to choose a good method depending on the eigenvalues of the differential equations being solved. This is a fact which invited our participation in developing new methods especially ones which utilise the advantages possessed by the GM formulae.

### 5.4.2 IMS Methods with GM Base Formula

An obvious extension of the IMS formula given by equations (5.4.4) is to replace the arithmetic mean in (5.5.4-11) with the corresponding geometric mean. Thus a new IMS formula based on the GM predictor-corrector is given by,
where the coefficients \( c_i \) are still to be determined so as to make the order of accuracy of the method for linear autonomous systems increased on each iteration.

It is possible in this case to determine the coefficients \( c_i \) by applying the analysis used for determining the stability regions of the formulae. Hence we will, at the same time, obtain the value of \( c_i \) and the magnification factor \( Q_1(h\lambda) \) for each iteration count \( i \).

To proceed in this way, we apply the formulae (5.4.6) to the model problem \( y' = \lambda y \). Thus from (5.4.6-i) we obtain

\[
y^{(1)}_{n+1} = (1+h\lambda)y_n,
\]

i.e.

\[
Q_1(h\lambda) = 1 + h\lambda,
\]

or writing \( h\lambda = z \), we have

\[
Q_1(z) = 1 + z.
\] (5.4.7)

This is a first order approximation to \( \exp(z) \). In a similar way (5.4.6-ii) becomes

\[
y^{(2)}_{n+1} = y_n + h\sqrt{\lambda y_n^{(1)}\lambda y_n}
\]

\[
= y_n + h\lambda \sqrt{(1+h\lambda)y_n^2}
\]

\[
= (1+h\lambda \sqrt{1+h\lambda})y_n.
\]

i.e.

\[
Q_2(z) = 1 + z \sqrt{1+z}
\] (5.4.8)
which is a second order approximation to \( \exp(z) \), since by expanding the square root we have,
\[
\sqrt{1+z} = 1 + \frac{z}{2} - \frac{z^2}{8} + \frac{z^3}{16} - \frac{5z^4}{128} - \frac{7z^5}{256} - \ldots
\]
and hence (5.4.8) can be written as
\[
Q_2(z) = 1 + z + \frac{\sqrt{z}}{2} - \frac{z^2}{8} + \frac{z^3}{16} - \frac{5z^4}{128} - \ldots
\]
which match with the Taylor series expansion of \( \exp(z) \) up to the second order. Next, the application of (5.5.4-iii) produces
\[
y_n(3) = y_n(2) + c_3 h [\lambda y_n(2) - \lambda y_n(1)]
= [Q_2 + c_3 z(Q_2 - Q_1)] y_n
= Q_3(z) y_n
\]
where
\[
Q_3(z) = 1 + z + \frac{z^2}{2} + \left(\frac{1}{8} \cdot \frac{c_3}{1} \right) z^3 + \frac{8 - 16 c_3}{128} \frac{z^4}{z^4} + \frac{8 c_3 - 5}{128} z^5 + \ldots
\]
For \( Q_3(z) \) to be the third order approximation to \( \exp(z) \) we must have
\[
1 c_3 - 1/8 = 1/6
\]
or
\[
c_3 = 7/12.
\]
With this value of \( c_3 \), \( Q_3(z) \) now becomes
\[
Q_3(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} - \frac{z^4}{96} - \frac{z^5}{384} + \ldots
\]
Similarly we may obtain a fourth order \( y_n(4) \) by using the following procedure.
\[
y_n(4) = y_n(3) + c_4 h [\lambda y_n(3) - \lambda y_n(2)]
= [Q_3 + c_4 z(Q_3 - Q_2)] y_n
= Q_4(z) y_n
\]
where
\[
Q_4(z) = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \frac{7 c_4}{24} - \frac{1}{96} + \left(\frac{7 c_4}{24} - \frac{1}{96}\right) z^4
+ \left(\frac{7 c_4}{96} - \frac{1}{384}\right) z^5 + \ldots
\]
Now, for $Q_4(z)$ to be a fourth order approximation we must have

\[ \frac{7}{24} c_4 - \frac{1}{96} = \frac{1}{24} \]

or $c_4 = 5/28$.

With $c_4$ taking this value, the expansion of $Q_4(z)$ thus becomes

\[ Q_4(z) = 1 + z + \frac{z^2}{2} + \frac{3z^3}{6} + \frac{4z^4}{24} - \frac{5z^5}{64} + \ldots \]

In a similar manner $c_5$ is obtained from expanding

\[ y_{n+1}^{(5)} = [Q_4 + c_5 z (Q_4 - Q_3)] y_n = Q_5(z) y_n \]

where $Q_5(z)$ contains, in its fifth order term, the following expression,

\[ \frac{5}{96} c_5 - \frac{1}{64} = \frac{1}{120} \]

i.e. $c_5 = 23/50$.

Higher order iteration formulae may be obtained if we start expanding $\sqrt{1+h\lambda}$ in the earlier stages to the required order. Therefore the GM version of the IMS formula is given by equations (5.4.6) with $c_i$ given by $c_3=7/12$, $c_4=5/28$, $c_5=23/50$, etc.

The stability functions for these formulae are given by

\[ Q_1(z) = 1 + z \] \hspace{1cm} (5.4.9-1)

\[ Q_2(z) = 1 + z \sqrt{1+z} \] \hspace{1cm} (5.4.9-ii)

\[ Q_i(z) = Q_{i-1}(z) + c_1 z (Q_{i-1} - Q_{i-2}) \] \hspace{1cm} (5.4.9-iii)

for $i = 3, 4, \ldots$

The regions in the complex $z$-plane where $|Q_i(z)| < 1$ determine the stability regions of the related formulae. The regions for $i=1(1)5$ are shown in Figure 5.4. By comparing this with the stability regions for the IMS formulae (5.4.4) as shown in Figure 5.3, we notice that these regions are smaller for each $i$. 
Figure 5.3: Stability regions for the IMS method for $i=1(1)5$.

Figure 5.4: Stability regions for the GM-IMS method for $i=1(1)5$. 
As we have indicated earlier, the purpose of examining different methods as the base for the IMS formulae is to obtain the most suitable methods for use with specific problems so as to achieve stability. At one stage, we may feel it possible to obtain a GM version of the IMS leap-frog formula. However, the following analysis will show that both formulae actually coincide.

We note that the leap-frog formula

\[ y_{n+1} = y_{n-1} + 2h f_n \]

can be considered as the limiting case of the trapezoidal formula over two steps of integration \([x_{n-1}, x_{n+1}]\). As the interval is shrunk to the middle, we may approximate the function values at both points to be equal to the value at the middle point, i.e.

Trapezoidal formula
(over 2-steps):

\[ y_{n+1} = y_{n-1} + 2h \frac{f_{n-1} + f_{n+1}}{2} \]

Taking the limit of the function evaluation we have

\[ \lim_{x_{n-1} \to x_n} \frac{f_{n-1} + f_{n+1}}{2} = \frac{f_n + f_n}{2} = f_n \]

and thus producing the leap-frog formula.

The same procedure may be applied to the GM formula over 2-steps.

GM formula
(over 2-steps):

\[ y_{n+1} = y_{n-1} + 2h \sqrt{f_{n-1} f_{n+1}} \]

\[ \lim_{x_{n-1} \to x_n} \sqrt{f_{n-1} f_{n+1}} = \sqrt{f_n f_n} = f_n \]

and again producing the leap-frog method. Therefore the corresponding GM formula for the leap-frog method is the leap-frog method itself and
we shall not investigate further. Alternatively we will look into the variable step size implementation of the Hyman's IMS leap-frog method and by using a symbolic manipulation package we will produce a list of the constants $c_i$. This consideration is the subject of the next section.

5.4.3 Symbolic Determination of the Coefficients in the Hyman's IMS Leap-Frog Method

Hyman's IMS leap-frog method consists of the second order leap-frog predictor given by

$$y_{n+1}^{(1)} = (1-r^2)y_n + r^2y_{n-1} + h_{n+1}(1+r)f_n$$

(5.4.10)

and the third order leap-frog corrector given by

$$y_{n+1}^{(2)} = [(2-r)(1+r)^2y_n + r^3y_{n-1} + h_{n+1}(1+r)^2f_n$$

$$+ h_{n+1}(1+r)f_n^{(1)})/(2+3r)$$

(5.4.11)

where $r = h_{n+1}/h_n$ is the ratio of the present and the previous stepsize, $h_n = t_n - t_{n-1}$.

Firstly we will examine the purpose of the parameter $r$ in the formula. Expanding formula (5.4.10) in a Taylor series about $x_n$ we obtain

$$y_{n+1}^{(1)} = (1-r^2)y_n + r^2(y_n - h_n y_n' + \frac{h_n^2}{2} y_n'' - \frac{1}{6} h_n^3 y_n''')$$

$$+ \frac{h_n^4}{24} y_n^{(iv)} + \ldots + h_{n+1}(1+r)y_n'$$

$$= y_n (1-r^2 + r^2) + y_n' (-r^2 h_n + (1+r) h_{n+1})$$

$$+ y_n'' (\frac{1}{2} r h_n^2) + y_n''' (-\frac{1}{6} r^2 h_n^3)$$

$$+ y_n^{(iv)} (\frac{1}{24} r^2 h_n^4) + \ldots$$

$$= y_n + h_{n+1} y_n' (-\frac{r}{r} +1+r) + h_{n+1}^2 y_n'' (-\frac{r^2}{r} + \frac{1}{6} r^2)$$

$$+ h_{n+1}^3 y_n''' (-\frac{r^3}{6}(1/r)^3) + h_{n+1}^4 y_n^{(iv)} (r^2/24)(1/r^4) + \ldots$$

$$= y_n + h_{n+1} y_n' + \frac{h_n^2}{2} y_n'' - \frac{1}{6} r h_{n+1}^3 y_n''''$$

$$+ \frac{h_{n+1}^4 (iv)}{24 r^2} + \ldots + (-1)^j \frac{j! h_{n+1}^j y_n^{(j)}}{j! r^{(j-2)}}$$

(5.4.12)
This formula is clearly of order 2.

Next,

\[ y_{n+1}^{(2)} = \frac{(2-r)(1-r)^2}{2+3r} y_n + \frac{r^3}{2+3r} y_{n-1} + h \frac{(1+r)^2}{2+3r} f_n \]

\[ + h \frac{1+r}{2+3r} f_{n+1} \]

\[ = \frac{(2-r)(1+r)^2}{(2+3r)} y_n + \frac{r^3}{(2+3r)} [y_n - h_y + \frac{1}{6}h^2 y'' + \ldots] \]

\[ + \frac{1}{6}h^3 y''' + \frac{1}{24}h^4 y^{(iv)} - \frac{1}{120}h^5 y^{(v)} + \ldots \]

\[ + \frac{h^2}{(2+3r)} y_n' + h \frac{(1+r)}{(2+3r)} [y_n' + h y_n'' + \ldots] \]

\[ + \frac{1}{6}h^3 y''' + \frac{1}{24}h^4 y^{(iv)} + \frac{1}{24} \frac{1+r}{r} h \frac{1}{2+3r} \frac{1}{2+3r} \]

\[ = y_n + h y_n' + \frac{1}{6}h^3 y''' + \frac{1}{24}h^4 y^{(iv)} + \frac{1}{120}h^5 y^{(v)} + \ldots \]

\[ + h \frac{5}{2+3r} y^{(v)} \left[ \frac{-1 + \frac{1+r}{24}}{r^2(2+3r)} \right] + \ldots + h \frac{1}{n+1} j \frac{1}{r}(j-3)(2+3r) \]

This formula is of order 3.

From equations (5.4.12) and (5.4.13) we may now define the general terms of the dummy variable \( i \) as \( A_{1j} \) and \( A_{2j} \) respectively as follows:

\[ A_{1j} = \left( \begin{array}{c} (-1)^j h y_n \hfill \frac{j!}{r(j-2)} \\
\end{array} \right) \]

\[ A_{2j} = \left( \begin{array}{c} h y_n \hfill (-1)^j \left[ \frac{1}{j!} - \frac{1+r}{(j-1)!} \right] \hfill \frac{r(j-3)(2+3r)}{r(j-3)(2+3r)} \\
\end{array} \right) \]

The purpose of \( r \) is therefore to give the facility to change the stepsize while maintaining the order of the predictor and corrector formulae. For example, if we want to double the stepsize we use \( r=2 \), and if we want to halve the stepsize we use \( r=4 \).
After the corrector cycle, additional iterations are based on the recurrence relation

\[ y_{n+1}^{(i)} = y_{n+1}^{(i-1)} + c_i h_{n+1}^{(i)} \left[ f_{n+1}^{(i-1)} - f_{n+1}^{(i-2)} \right]. \]

We will now determine the values of \( c_i \) as functions of \( r \) in general so that the need for a facility of changing stepsize is backed by the availability of the related \( c_i \). We write,

\[
\begin{align*}
    y_{n+1}^{(3)} &= y_{n+1}^{(2)} + c_3 h_{n+1}^{(2)} \left[ f_{n+1}^{(2)} - f_{n+1}^{(1)} \right] \\
    &= y_n + h \frac{1}{6} y'_n + \frac{1}{6} h y''_n + \frac{h^3}{6} y'''_n \\
    &\quad + h^4 \frac{1}{6} \left( \frac{1}{24} - \frac{1 + r}{r(2 + 3r)} \right) + c_3 \left( \frac{1}{6} + \frac{1}{6r} \right) \\
    &\quad + h^5 \frac{1}{6} \left[ \frac{-1}{120} + \frac{1 + r}{r(2 + 3r)} \right] + c_3 \left( \frac{1}{24} - \frac{1 + r}{r(2 + 3r)} - \frac{1}{24r^2} \right) \\
    &\quad + h^6 \left( A_{36} + h^7 (A_{37}) + \ldots + h^j (A_{3j}) + \ldots \right)
\end{align*}
\]

where

\[ A_{3j} = A_{2j} + c_3 \left( A_{2, j-1} + A_{1, j-1} \right). \]

For \( y_{n+1}^{(3)} \) to have order 4, we must have

\[
A_{34} = \frac{1}{24} - \frac{1 + r}{6r} + c_3 \left( \frac{1}{6} + \frac{1}{6r} \right) = \frac{1}{24}, \quad \text{i.e.} \quad c_3 = \frac{3(r+1)}{4(2+3r)}. \]

With the value of \( c_3 \) as given, we may now compute all \( A_{3j} \) for \( j=5,6,\ldots \).

The next stage is to determine \( c_4 \) in the evaluation of

\[ y_{n+1}^{(4)} = y_{n+1}^{(3)} + c_4 h_{n+1}^{(3)} \left[ f_{n+1}^{(3)} - f_{n+1}^{(2)} \right]. \]

The same procedure is employed. In general we have the following algorithm to determine \( c_i \), \( i=3,4,5,\ldots,n \) for any pre-assigned value of \( n \).
Algorithm

1. Compute $A_{1j} = \frac{(-1)^j}{j! \cdot r(j-2)}$, $j = 3, 4, \ldots$

2. Compute

$$A_{23} = \frac{1}{6}$$

$$A_{2j} = \frac{(-1)^j [1 - j(1+r)]}{j! \cdot r(j-3)(2 + 3r)}$$, $j = 4, 5, \ldots$

and

3. For $i=3, 4, \ldots, n$

Calculate

$$A_{1, i+1} = \frac{1}{(i+1)!}$$

$$c_i = \frac{A_{1, i+1} - A_{1-1, i+1}}{A_{1-1, i} - A_{1-2, i}}$$

$$A_{1, j} = A_{1-1, j} + c_i (A_{1-1, j-1} - A_{1-2, j-1})$$

$j = i+2, i+3, \ldots, n.$

To enable us to compute the $c_i$'s in terms of the variable $r$ we use a symbolic computation package on the computer. At Loughborough University, the REDUCE symbolic computation is installed on the DEC VAX 11 computer which runs Berkeley UNIX operating system.
The computer program and the results for general \( r \) are given in Appendix 6. Since the values of \( r=\frac{1}{2}, 1 \) and 2 correspond to halving the stepsize, no change in stepsize, and doubling the stepsize respectively, we also produce the results of the corresponding \( c_i \)'s which are given in Table 5.9 for \( n=15 \).

A similar procedure can also be used to evaluate the \( c_i \)'s in any other IMS formulae with suitable modification in the values of \( A_{1j} \) and \( A_{2j} \). As an example, we also produce the values of \( c_i \)'s in the GM-IMS formula for \( i=3 \) to \( i=15 \), the first three values of which have already been previously calculated by hand in Section 5.4.2. These are given in Table 5.10.
Table 5.9: Values of $c_3$ through $c_{15}$ in the
coefficients of the Hyman's IMS leap-frog formula

(a) with $r=\frac{1}{2}$

<table>
<thead>
<tr>
<th>$c_3$</th>
<th>9/28</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_4$</td>
<td>5/21</td>
</tr>
<tr>
<td>$c_5$</td>
<td>67/350</td>
</tr>
<tr>
<td>$c_6$</td>
<td>79/469</td>
</tr>
<tr>
<td>$c_7$</td>
<td>3127/22120</td>
</tr>
<tr>
<td>$c_8$</td>
<td>22013/197001</td>
</tr>
<tr>
<td>$c_9$</td>
<td>192917/1540910</td>
</tr>
<tr>
<td>$c_{10}$</td>
<td>1882735/14854609</td>
</tr>
<tr>
<td>$c_{11}$</td>
<td>186581/263582900</td>
</tr>
<tr>
<td>$c_{12}$</td>
<td>65672567/16978871</td>
</tr>
<tr>
<td>$c_{13}$</td>
<td>820163237/4597079690</td>
</tr>
<tr>
<td>$c_{14}$</td>
<td>-19124185413/15500000951793</td>
</tr>
<tr>
<td>$c_{15}$</td>
<td>26088680581037/107095063831280</td>
</tr>
</tbody>
</table>

(b) with $r=1$

<table>
<thead>
<tr>
<th>$c_3$</th>
<th>3/10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_4$</td>
<td>7/30</td>
</tr>
<tr>
<td>$c_5$</td>
<td>4/21</td>
</tr>
<tr>
<td>$c_6$</td>
<td>451/2800</td>
</tr>
<tr>
<td>$c_7$</td>
<td>314/2255</td>
</tr>
<tr>
<td>$c_8$</td>
<td>1153/9420</td>
</tr>
<tr>
<td>$c_9$</td>
<td>126/1153</td>
</tr>
<tr>
<td>$c_{10}$</td>
<td>21289/215600</td>
</tr>
<tr>
<td>$c_{11}$</td>
<td>774686/8622045</td>
</tr>
<tr>
<td>$c_{12}$</td>
<td>8302007/100709180</td>
</tr>
<tr>
<td>$c_{13}$</td>
<td>634850/8302007</td>
</tr>
<tr>
<td>$c_{14}$</td>
<td>94640561/1333185000</td>
</tr>
<tr>
<td>$c_{15}$</td>
<td>31088334/473202805</td>
</tr>
</tbody>
</table>

continued...
(c) with $r=2$

\begin{align*}
c_3 &= 9/32 \\
c_4 &= 107/480 \\
c_5 &= 631/3424 \\
c_6 &= 110841/706720 \\
c_7 &= 483581/3546912 \\
c_8 &= 9321429/77372960 \\
c_9 &= 11100563/58079673 \\
c_{10} &= 60916628047880642233201/237778817481522493800624 \\
c_{11} &= 5310547949027826910818461041275359317/ \\
& \quad 22657985139246566410725382134372040 \\
c_{12} &= 143999556141897529139568517077042239031428860306051/ \\
& \quad 646330757512623292374105080079400658097453367651191 \\
c_{13} &= 5693164999086214797282417743352368508734340977208741021258607/ \\
& \quad 255140893866648888137936607809903499556006424851100830852036032 \\
c_{14} &= 154928018883488170456514129011368349139574608414082661335547116376867264047/ \\
& \quad 6928979234953265451307461076439100282063554149386577535674089396934528878610 \\
c_{15} &= 643001667547798320038718046406178451370700059516244679260469139654648565263943115603385/ \\
& \quad 2881279596398405402241284852914277415222543368357424669931475812787864076606839404301494 \\
\end{align*}
Table 5.10: Values of $c_3$ through $c_{15}$  
in the coefficients of the GM-IMS formula.

\begin{align*}
c_3 &= \frac{7}{12} \\
c_4 &= \frac{5}{28} \\
c_5 &= \frac{23}{50} \\
c_6 &= -\frac{85}{552} \\
c_7 &= -\frac{1583}{1190} \\
c_8 &= -\frac{4405}{6332} \\
c_9 &= -\frac{64243}{79290} \\
c_{10} &= -\frac{4208363}{5139440} \\
c_{11} &= -\frac{77766745}{92583986} \\
c_{12} &= -\frac{797922407}{933200940} \\
c_{13} &= -\frac{18002502265}{20745982582} \\
c_{14} &= -\frac{885564602641}{1008140126840} \\
c_{15} &= -\frac{14193981871421}{26566938079230}
\end{align*}

**Numerical Example**

To see the competitiveness of the GM-IMS formula to the Hyman's IMS method we use both formulae to solve the IVP

\[ y' = -y, \quad y(0) = 1, \quad 0 \leq x \leq 1. \]

To facilitate the comparison we print out the errors in the first to the fifth order formulae at each step. The errors obtained by using $h=0.1$ are shown in Table 5.11. It is seen that both the IMS and the GM-IMS formulae produce an increasing degree of accuracy as the iteration count increases. This follows from the fact that the errors, where

\[ \text{error} = \text{exact solution} - \text{numerical solution}, \]

are decreasing as $i$ increases. The errors in the GM-IMS method is seen not significantly different from the error in the IMS formula, although it is slightly larger. Thus, we conclude that the GM-IMS formula is competitive for use as an alternative method for solving the initial value problems in ODEs.
Table 5.11

\( y' = -y, \ y(0)=1 \)

Errors in the 1th order formula
\( h = 0.1, \ k=1. \)

(a) QM-MS formula

<table>
<thead>
<tr>
<th>x</th>
<th>1=1</th>
<th>1=2</th>
<th>1=3</th>
<th>1=4</th>
<th>1=5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>0.483742E-02</td>
<td>-0.294252E-03</td>
<td>0.509527E-05</td>
<td>-0.250221E-06</td>
<td>-0.432821E-08</td>
</tr>
<tr>
<td>0.20</td>
<td>0.437685E-02</td>
<td>-0.266477E-03</td>
<td>0.439398E-05</td>
<td>-0.452818E-06</td>
<td>-0.230325E-06</td>
</tr>
<tr>
<td>0.30</td>
<td>0.396014E-02</td>
<td>-0.241323E-03</td>
<td>0.376193E-05</td>
<td>-0.614590E-06</td>
<td>-0.413271E-06</td>
</tr>
<tr>
<td>0.40</td>
<td>0.358309E-02</td>
<td>-0.218544E-03</td>
<td>0.321857E-05</td>
<td>-0.741473E-06</td>
<td>-0.559311E-06</td>
</tr>
<tr>
<td>0.50</td>
<td>0.324195E-02</td>
<td>-0.197914E-03</td>
<td>0.274455E-05</td>
<td>-0.836404E-06</td>
<td>-0.673813E-06</td>
</tr>
<tr>
<td>0.60</td>
<td>0.293329E-02</td>
<td>-0.179232E-03</td>
<td>0.233161E-05</td>
<td>-0.910600E-06</td>
<td>-0.761458E-06</td>
</tr>
<tr>
<td>0.70</td>
<td>0.265401E-02</td>
<td>-0.162313E-03</td>
<td>0.197240E-05</td>
<td>-0.961269E-06</td>
<td>-0.826320E-06</td>
</tr>
<tr>
<td>0.80</td>
<td>0.240133E-02</td>
<td>-0.146991E-03</td>
<td>0.166045E-05</td>
<td>-0.994048E-06</td>
<td>-0.871942E-06</td>
</tr>
<tr>
<td>0.90</td>
<td>0.217270E-02</td>
<td>-0.131166E-03</td>
<td>0.139000E-05</td>
<td>-0.101188E-05</td>
<td>-0.901397E-06</td>
</tr>
<tr>
<td>1.00</td>
<td>0.196584E-02</td>
<td>-0.120550E-03</td>
<td>0.115600E-05</td>
<td>-0.101732E-05</td>
<td>-0.917350E-06</td>
</tr>
</tbody>
</table>

(b) IMS formula

<table>
<thead>
<tr>
<th>x</th>
<th>1=1</th>
<th>1=2</th>
<th>1=3</th>
<th>1=4</th>
<th>1=5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>0.483742E-02</td>
<td>-0.162582E-03</td>
<td>0.406470E-05</td>
<td>-0.819640E-07</td>
<td>0.136925E-08</td>
</tr>
<tr>
<td>0.20</td>
<td>0.437700E-02</td>
<td>-0.147184E-03</td>
<td>0.362183E-05</td>
<td>-0.148328E-06</td>
<td>0.729251E-07</td>
</tr>
<tr>
<td>0.30</td>
<td>0.396041E-02</td>
<td>-0.133245E-03</td>
<td>0.321000E-05</td>
<td>-0.201319E-06</td>
<td>0.133092E-06</td>
</tr>
<tr>
<td>0.40</td>
<td>0.358347E-02</td>
<td>-0.120626E-03</td>
<td>0.284386E-05</td>
<td>-0.242882E-06</td>
<td>0.181474E-06</td>
</tr>
<tr>
<td>0.50</td>
<td>0.322420E-02</td>
<td>-0.109202E-03</td>
<td>0.251829E-05</td>
<td>-0.274711E-06</td>
<td>0.218851E-06</td>
</tr>
<tr>
<td>0.60</td>
<td>0.293390E-02</td>
<td>-0.988956E-04</td>
<td>0.222893E-05</td>
<td>-0.298228E-06</td>
<td>0.247738E-06</td>
</tr>
<tr>
<td>0.70</td>
<td>0.265456E-02</td>
<td>-0.894966E-04</td>
<td>0.197184E-05</td>
<td>-0.314804E-06</td>
<td>0.269145E-06</td>
</tr>
<tr>
<td>0.80</td>
<td>0.240191E-02</td>
<td>-0.810208E-04</td>
<td>0.174349E-05</td>
<td>-0.325617E-06</td>
<td>0.284235E-06</td>
</tr>
<tr>
<td>0.90</td>
<td>0.217330E-02</td>
<td>-0.733473E-04</td>
<td>0.154075E-05</td>
<td>-0.333241E-06</td>
<td>0.294015E-06</td>
</tr>
<tr>
<td>1.00</td>
<td>0.196645E-02</td>
<td>-0.664009E-04</td>
<td>0.136080E-05</td>
<td>-0.333241E-06</td>
<td>0.299360E-06</td>
</tr>
</tbody>
</table>
5.5 TWO-STAGE IMPLICIT CN RUNGE-KUTTA FORMULA

5.5.1 Implicit Runge-Kutta Methods

All methods discussed so far in this chapter have been explicit. In the search for reliable and efficient one-step methods for solving general ODE problems, explicit Runge-Kutta methods are sometimes avoided for two reasons. The first is that the computational cost, particularly as measured in terms of derivative evaluations increases rapidly as high order requirements are imposed. The second reason is specific to stiff problems and is concerned with the stability properties of these methods.

As pointed out by Butcher [1963], it is also possible to consider implicit Runge-Kutta methods. The general R-stage implicit Runge-Kutta methods is defined by

\[ y_{n+1} = y_n + h \sum_{r=1}^{R} c_r k_r \]  

(5.5.1-i)

\[ k_r = f(x_n + h b_r , y_n + h \sum_{s=1}^{R} a_{rs} k_s) , \quad r=1,2,\ldots,R \]  

(5.5.1-ii)

\[ b_r = \sum_{s=1}^{R} a_{rs} , \quad r=1,2,\ldots,R . \]  

(5.5.1-iii)

Unlike the explicit case, the functions \( k_r \) are no longer defined explicitly but by a set of \( R \) implicit equations which are in general non-linear. Butcher [1964] gives a general treatment for the methods. An example is the two-stage method obtained by setting \( R=2 \) of order 4 given by

\[ k_1 = f(x_n + (1+\sqrt{3}/6)h , y_n + \frac{1}{4}hk_1 + (\frac{1}{4}+\frac{\sqrt{3}}{6})hk_2) , \]  

(5.5.2)

\[ k_2 = f(x_n + (1-\sqrt{3}/6)h , y_n + (\frac{1}{4} - \frac{\sqrt{3}}{6})hk_1 + \frac{1}{4}hk_2) , \]  

(5.5.2)

\[ y_{n+1} = y_n + \frac{h}{2} (k_1 + k_2) , \]  

(5.5.2)
a method originally proposed by Hammer and Hollingsworth [1955]. (See, for example, Lambert [1973], p.153). The implicit Runge-Kutta methods can also offer substantially improved weak stability characteristics (Lambert [1973], p.155).

5.5.2 2-Stage Implicit GM Runge-Kutta Method

The development of the GM Runge-Kutta formulae as an extension of the classical Runge-Kutta formula by employing the concept of the geometric mean has made a breakthrough to the development of new formulae to complement the existing ones. It is possible, in addition, to extend the formulation to the implicit case. However, the derivation of such implicit methods is expected to be rather complicated. We consider here only the two-stage method given by the following relations:

\[ k_1 = f(y_n + h a_{11} k_1 + h a_{12} k_2) \]  
\[ k_2 = f(y_n + h a_{21} k_1 + h a_{22} k_2) \]  
\[ y_{n+1} = y_n + h k_1 p k_2 q \]

Expanding as a Taylor series about \( y_n \) we obtain

\[ k_1 = f + (h a_{11} k_1 + h a_{12} k_2) f_y + \frac{1}{2} (h a_{11} k_1 + h a_{12} k_2)^2 f_{yy} \]
\[ + \frac{1}{6} (h a_{11} k_1 + h a_{12} k_2)^3 f_{yyy} + \ldots \]
\[ = f + h(a_{11} k_1 + a_{12} k_2) f_y + \frac{1}{2} h^2 (a_{11} k_1 + a_{12} k_2)^2 f_{yy} \]
\[ + \frac{1}{6} h^3 (a_{11} k_1 + a_{12} k_2)^3 f_{yyy} + \ldots \]  

and similarly,

\[ k_2 = f + h(a_{21} k_1 + a_{22} k_2) f_y + \frac{1}{2} h^2 (a_{21} k_1 + a_{22} k_2)^2 f_{yy} \]
\[ + \frac{1}{6} h^3 (a_{21} k_1 + a_{22} k_2)^3 f_{yyy} + \ldots \]
Since these two equations are implicit, we can no longer proceed by successive substitution as in the explicit case. Let us assume, instead, that the solutions for $k_1$ and $k_2$ may be expressed in the form

$$k_i = A_i + hB_i + h^2C_i + h^3D_i + O(h^4)$$  \hspace{1cm} (5.5.6)\hspace{1cm} i = 1,2.

Substituting for $k_1$ and $k_2$ by (5.5.6) in (5.5.4) we have

$$A_1 + hB_1 + h^2C_1 + h^3D_1 + \ldots$$

$$= f + h(a_{11}A_1+hB_1+h^2C_1+h^3D_1+\ldots) + a_{12}(A_2+hB_2+h^2C_2+h^3D_2+\ldots)$$

$$+ a_{12}(A_2+hB_2+h^2C_2+h^3D_2+\ldots)f_{yy} + \frac{1}{6} h^3(a_{11}(A_1+hB_1) + h^2C_1+h^3D_1+\ldots) + a_{12}(A_2+hB_2+h^2C_2+h^3D_2+\ldots)$$

$$= f + h [(a_{11}A_1+a_{12}A_2) + h(a_{11}B_1+a_{12}B_2) + h^2 \{ (a_{11}A_1 + a_{12}A_2)^2$$

$$+ 2h(a_{11}A_1+a_{12}A_2)(a_{11}B_1+a_{12}B_2) + h^2 \{ (a_{11}B_1 + a_{12}B_2)^2 + 2(a_{11}A_1+a_{12}A_2)(a_{11}C_1+a_{12}C_2) \} f_{yy}$$

$$+ \frac{1}{6} h^3(a_{11}A_1+a_{12}A_2)^3 + h \{ (a_{11}B_1+a_{12}B_2)(a_{11}A_1+a_{12}A_2)^2$$

$$+ a_{12}A_2 \} + 2(a_{11}A_1+a_{12}A_2)(a_{11}B_1+a_{12}B_2)(a_{11}C_1+a_{12}C_2) \} f_{yyy}$$

On equating powers of $h$ we obtain,

<table>
<thead>
<tr>
<th>Constant</th>
<th>$A_1 = f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$h$</td>
<td>$B_1 = (a_{11}A_1+a_{12}A_2) + h(a_{11}B_1+a_{12}B_2)$</td>
</tr>
<tr>
<td>$h^2$</td>
<td>$C_1 = (a_{11}B_1+a_{12}B_2) + \frac{1}{6} h(a_{11}A_1+a_{12}A_2)^3 f_{yyy}$</td>
</tr>
</tbody>
</table>
| $h^3$    | $D_1 = (a_{11}C_1+a_{12}C_2) + \frac{1}{6} h(a_{11}A_1+a_{12}A_2)^3 + 2(a_{11}A_1+a_{12}A_2)^2 + a_{12}A_2 f_{yyy}$ | (5.5.8)
Similarly, on substituting $k_1$ and $k_2$ by (5.5.6) in (5.5.5) gives

\[
\begin{align*}
A_2 &= f \\
B_2 &= (a_{21}A_1 + a_{22}A_2)f_y \\
C_2 &= (a_{21}B_1 + a_{22}B_2)f_y + \frac{1}{2} \frac{d}{df}(a_{21}A_1 + a_{22}A_2)^2f_yy \\
D_2 &= (a_{21}C_1 + a_{22}C_2)f_y + \frac{1}{6} (a_{21}A_1 + a_{22}A_2)^3f_yyy.
\end{align*}
\] (5.5.9)

The set of equations (5.5.8) and (5.5.9) is seen to be explicit and can be solved by successive substitution. Hence, after simplifying,

\[
\begin{align*}
A_1 &= f \\
A_2 &= f \\
B_1 &= s_1ff_y, \quad \text{where } s_1 = a_{11} + a_{12} \\
B_2 &= s_2ff_y, \quad \text{where } s_2 = a_{21} + a_{22} \\
C_1 &= (a_{11} s_1 + a_{12} s_2)^2f_y^2 + i s_1^2 f_y^2f_yy \\
C_2 &= (a_{21} s_1 + a_{22} s_2)^2f_y^2 + i s_2^2 f_y^2f_yy \\
D_1 &= [a_{11}(a_{11} s_1 + a_{12} s_2) + a_{12}(a_{21} s_1 + a_{22} s_2)]ff_y^3 \\
&\quad + [i s_1^2 + i a_{12} s_2 + s_1(a_{11} s_1 + a_{12} s_2)]f_y^2f_yf_yy \\
&\quad + \frac{1}{6} s_1^3 f_y^3f_yyy \\
D_2 &= [a_{21}(a_{11} s_1 + a_{12} s_2) + a_{22}(a_{21} s_1 + a_{22} s_2)]ff_y^3 \\
&\quad + [i a_{21} s_1^2 + i a_{22} s_2^2 + s_2(a_{21} s_1 + a_{22} s_2)]f_y^2f_yf_yy \\
&\quad + \frac{1}{6} s_2^3 f_y^3f_yyy.
\end{align*}
\]

Therefore,

\[
\begin{align*}
k_1 &= f + h[s_1ff_y] + h^2[(a_{11} s_1 + a_{12} s_2)ff_y^2 + i s_1^2 f_y^2f_yy] \\
&\quad + h^3\left\{ a_{11}(a_{11} s_1 + a_{12} s_2)ff_y^3 + [i s_1^2 + i a_{12} s_2 + s_1(a_{11} s_1 + a_{12} s_2)]f_y^2f_yf_yy \\
&\quad + \frac{1}{6} s_1^3 f_y^3f_yyy \right\}, \quad (5.5.10)
\end{align*}
\]
\[ k_1^P = f^P \left\{ 1 + hps_1 f_y + h^2 \left[ p(a_{11}s_1 + a_{12}s_2)f_y + ps_1^2 f_{yy} + \frac{1}{6} ps_1 f_{yy} \right] + h^3 \left[ q[a_{11}s_1 + a_{12}s_2]f_y + ps_1 f_{yy} + \frac{1}{6} ps_1 f_{yy} \right] + (p-1)s_1 f_{yy} + \frac{1}{6} s_1^3 f_{yy} \right\}, \] (5.5.11)

and similarly,

\[ k_2^q = f^q \left\{ 1 + hqs_2 f_y + h^2 \left[ q(a_{21}s_1 + a_{22}s_2)f_y^2 + q^2 f_{yy} + \frac{1}{6} q^3 f_{yy} \right] + h^3 \left[ q[a_{21}s_1 + a_{22}s_2]f_y + q^2 f_{yy} + \frac{1}{6} q^3 f_{yy} \right] + \frac{1}{6} (q-1)q^3 f_{yy} \right\}. \] (5.5.12)

Thus,

\[ k_1 P k_2^q = f^{p+q} \left\{ 1 + h[(ps_1 + qs_2)f_y] + h^2 [pq s_1 s_2 f_y^2 + p(a_{11}s_1 + a_{12}s_2)f_y + ps_1^2 f_{yy} + q(a_{21}s_1 + a_{22}s_2)f_y + q^2 f_{yy} + \frac{1}{6} q^3 f_{yy} \right\}, \] (5.5.13)

\[ + h^3 \left[ q[a_{11}s_1 + a_{12}s_2]f_y + q^2 f_{yy} + \frac{1}{6} q^3 f_{yy} \right] + \frac{1}{6} (p-1)s_1 f_{yy} + \frac{1}{6} s_1^3 f_{yy} \right\} + p(p-1)(s_1^3 f_{yy} + \frac{1}{6} s_1^3 f_{yy} \right\}. \]
Comparing the expansion in (5.5.13) with the Taylor series expansion of 
y(x_{n+1}), term by term we obtain,

\[ hf: \quad p+q = 1 \]  
(5.5.14-1)

\[ h^2 f_{xy}: \quad ps_1 + qs_2 = \frac{1}{4} \]  
(5.5.14-i)

\[ h^3 f_{yy}: \quad pqs_1 s_2 + p(a_{11}s_1 + a_{12}s_2) + \frac{1}{p}(p-1)s_1^2 \]  
+ q(a_{21}s_1 + a_{22}s_2) + \frac{1}{q}(q-1)s_2^2 = \frac{1}{6} \]  
(5.5.14-iii)

\[ h^3 f_{yy}: \quad \frac{1}{6}ps_1^2 + \frac{1}{6}qs_2^2 = \frac{1}{6} \]  
(5.5.14-iv)

\[ h^4 f_{yy}: \quad qs_2(p(a_{11}s_1 + a_{12}s_2) + \frac{1}{p}(p-1)s_1^2) + ps_1(q(a_{21}s_1 + a_{22}s_2) + \frac{1}{q}(q-1)s_2^2) \]  
+ \frac{1}{6}p(p-1)(p-2)s_1^3 + q[a_{21}(a_{11}s_1 + a_{12}s_2) + a_{22}(a_{21}s_1 + a_{22}s_2) + q(q-1)s_2(a_{21}s_1 + a_{22}s_2)] \]  
+ \frac{1}{24}q(q-1)(q-2)s_2^3 = \frac{1}{24} \]  
(5.5.14-v)

\[ h^4 f_{yy}: \quad qs_2(\frac{1}{6}ps_1^2) + ps_1(\frac{1}{6}qs_2^2) + p[a_{11}s_2^2] \]  
+ \frac{1}{6}a_{12}s_2^2 + s_1(a_{11}s_1 + a_{12}s_2) + p(p-1)s_1(\frac{1}{6}s_1^2) \]  
+ q[a_{21}s_1^2 + a_{22}s_2^2 + s_2(a_{21}s_1 + a_{22}s_2)] \]  
+ q(q-1)s_2(\frac{1}{6}s_2^2) = \frac{1}{24} \]  
(5.5.14-vi)

\[ h^4 f_{yy}: \quad \frac{1}{6}ps_1^3 + \frac{1}{6}qs_2^3 = \frac{1}{24} \]  
(5.5.14-vii)

There are in effect only six coefficients, namely \( a_{11}, a_{12}, a_{21}, a_{22}, p \) and \( q \) to be determined. The form of expansion for \( k_1 pk_2^q \) holds out for the possibility of attaining order 4. This will entirely depend on the nature of the system since we need to satisfy the seven conditions (i.e. equations (5.5.14-1) through (5.5.14-vii)) with only six coefficients at our disposal. In solving this nonlinear system of equations we will certainly use the first four equations for an order of at least 3. The two other equations to be satisfied are chosen from among the last three equations.
If the solutions obtained are also satisfied by the unchosen equation then we have a method of order 4. Otherwise, the unchosen equation will provide an error term for the method, thus producing three different methods with three different error terms.

However, unlike the corresponding arithmetic mean formula which is more fortunate, one of the seven conditions cannot be satisfied with the GM formula. The three sets of parameters which give rise to the third order formulae are given in Table 5.12. The error terms for the methods are given as indicated.

Table 5.12: Three sets of parameters which give rise to the third order GM Runge-Kutta formulae

(a) Set 1 parameters
\[ a_{11} = 0.368596443918D+00 \]
\[ a_{12} = 0.61833368305D+00 \]
\[ a_{21} = -0.393859999940D-01 \]
\[ a_{22} = 0.368245660598D+00 \]
\[ p = 0.260063974909D+00 \]
\[ q = 0.739936025090D+00 \]
Error term = $0.438596489945D-02 \ h^3 f^{3} f \ y y y$

(b) Set 2 parameters
\[ a_{11} = 0.405502116925D+00 \]
\[ a_{12} = 0.383173017682D+00 \]
\[ a_{21} = -0.498396842665D-01 \]
\[ a_{22} = 0.261164549701D+00 \]
\[ p = 0.500000000000D+00 \]
\[ q = 0.500000000000D+00 \]
Error term = $0.138888911678D-01 \ h^3 f^{2} f \ y y y$

continued...
(c) Set 3 parameters
\[ a_{11} = 0.3221668783649 \times 10^0 \]
\[ a_{12} = 0.466506350958 \times 10^0 \]
\[ a_{21} = 0.334936489974 \times 10^{-1} \]
\[ a_{22} = 0.177831216438 \times 10^0 \]
\[ p = 0.500000000000 \times 10^0 \]
\[ q = 0.500000000000 \times 10^0 \]
Error term = 0.208333333333 \times 10^{-1} \cdot h^4 f_y y^3.

Numerical Example

Since the 2-stage implicit GM Runge-Kutta formulae are only of the third order, they are not really comparable in terms of the fourth order accuracy with the classical 2-stage implicit Runge-Kutta formula has for solving the general problem. However, since the error terms of the GM formulae have the property, for example the error terms will vanish if \( f_y y = 0 \) with the first set of parameters, or \( f_y = 0 \) with the second set of parameters as shown in Table 5.12, then these formulae would eventually become competitive for a certain class of problems.

We consider two different problems, i.e.,

(a) \( y' = -y \), \( y(0)=1 \), \( 0 \leq x \leq 1 \).

For this problem \( f_y y = f_y y = 0 \), and hence the formulae with the set 1 and/or set 2 parameters are reduced to fourth order as in the classical case. The results are shown in Table 5.13, where we have used \( h=0.1 \).

(b) \( y' = 1+y^2 \), \( y(0)=1 \), \( 0 \leq x \leq 1 \).

For this problem, \( f_y y = 0 \), but \( f_y y \neq 0 \). Thus, the method with the first set of parameters reduces to a fourth order method but this is not achieved with the second set of parameters. The results by using \( h=0.1 \) are shown in Table 5.14.
Table 5.13

Problem (a) \( y' = -y \), \( y(0)=1 \)

(i) Implicit GM Runge-Kutta with set 1 parameters \( h = 0.1 \)

<table>
<thead>
<tr>
<th>( x )</th>
<th>Exact Solution</th>
<th>Numerical Solution</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.10000000E+01</td>
<td>0.90483746E+00</td>
<td>-0.48742456E-07</td>
</tr>
<tr>
<td>0.1</td>
<td>0.90483741E+00</td>
<td>0.90483741E+00</td>
<td>-0.88207999E-07</td>
</tr>
<tr>
<td>0.2</td>
<td>0.81873075E+00</td>
<td>0.81873084E+00</td>
<td>-0.11972085E-06</td>
</tr>
<tr>
<td>0.3</td>
<td>0.74081822E+00</td>
<td>0.74081834E+00</td>
<td>-0.14443721E-06</td>
</tr>
<tr>
<td>0.4</td>
<td>0.67032004E+00</td>
<td>0.67032019E+00</td>
<td>-0.16333941E-06</td>
</tr>
<tr>
<td>0.5</td>
<td>0.60653066E+00</td>
<td>0.60653082E+00</td>
<td>-0.17713604E-06</td>
</tr>
<tr>
<td>0.6</td>
<td>0.54881163E+00</td>
<td>0.54881181E+00</td>
<td>-0.18718958E-06</td>
</tr>
<tr>
<td>0.7</td>
<td>0.49658530E+00</td>
<td>0.49658549E+00</td>
<td>-0.19356180E-06</td>
</tr>
<tr>
<td>0.8</td>
<td>0.44932896E+00</td>
<td>0.44932915E+00</td>
<td>-0.19702605E-06</td>
</tr>
<tr>
<td>0.9</td>
<td>0.40656966E+00</td>
<td>0.40656985E+00</td>
<td>-0.19807809E-06</td>
</tr>
<tr>
<td>1.0</td>
<td>0.36787944E+00</td>
<td>0.36787963E+00</td>
<td>-0.19807809E-06</td>
</tr>
</tbody>
</table>

(ii) Implicit GM Runge-Kutta with set 2 parameters

<table>
<thead>
<tr>
<th>( x )</th>
<th>Exact Solution</th>
<th>Numerical Solution</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.10000000E+01</td>
<td>0.90483744E+00</td>
<td>-0.30426891E-07</td>
</tr>
<tr>
<td>0.1</td>
<td>0.90483741E+00</td>
<td>0.90483744E+00</td>
<td>-0.55062780E-07</td>
</tr>
<tr>
<td>0.2</td>
<td>0.81873075E+00</td>
<td>0.81873080E+00</td>
<td>-0.74373429E-07</td>
</tr>
<tr>
<td>0.3</td>
<td>0.74081822E+00</td>
<td>0.74081829E+00</td>
<td>-0.90163186E-07</td>
</tr>
<tr>
<td>0.4</td>
<td>0.67032004E+00</td>
<td>0.67032013E+00</td>
<td>-0.10197829E-06</td>
</tr>
<tr>
<td>0.5</td>
<td>0.60653066E+00</td>
<td>0.60653076E+00</td>
<td>-0.11072906E-06</td>
</tr>
<tr>
<td>0.6</td>
<td>0.54881163E+00</td>
<td>0.54881174E+00</td>
<td>-0.11689043E-06</td>
</tr>
<tr>
<td>0.7</td>
<td>0.49658530E+00</td>
<td>0.49658542E+00</td>
<td>-0.12087639E-06</td>
</tr>
<tr>
<td>0.8</td>
<td>0.44932896E+00</td>
<td>0.44932908E+00</td>
<td>-0.12304616E-06</td>
</tr>
<tr>
<td>0.9</td>
<td>0.40656966E+00</td>
<td>0.40656978E+00</td>
<td>-0.12370526E-06</td>
</tr>
<tr>
<td>1.0</td>
<td>0.36787944E+00</td>
<td>0.36787956E+00</td>
<td>-0.12370526E-06</td>
</tr>
</tbody>
</table>

(iii) Classical implicit Runge-Kutta method

<table>
<thead>
<tr>
<th>( x )</th>
<th>Exact Solution</th>
<th>Numerical Solution</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.10000000E+01</td>
<td>0.90483743E+00</td>
<td>-0.12586608E-07</td>
</tr>
<tr>
<td>0.1</td>
<td>0.90483741E+00</td>
<td>0.90483743E+00</td>
<td>-0.22777668E-07</td>
</tr>
<tr>
<td>0.2</td>
<td>0.81873075E+00</td>
<td>0.81873077E+00</td>
<td>-0.30915130E-07</td>
</tr>
<tr>
<td>0.3</td>
<td>0.74081822E+00</td>
<td>0.74081825E+00</td>
<td>-0.37297556E-07</td>
</tr>
<tr>
<td>0.4</td>
<td>0.67032004E+00</td>
<td>0.67032008E+00</td>
<td>-0.42185280E-07</td>
</tr>
<tr>
<td>0.5</td>
<td>0.60653066E+00</td>
<td>0.60653070E+00</td>
<td>-0.45804984E-07</td>
</tr>
<tr>
<td>0.6</td>
<td>0.54881163E+00</td>
<td>0.54881168E+00</td>
<td>-0.48353741E-07</td>
</tr>
<tr>
<td>0.7</td>
<td>0.49658530E+00</td>
<td>0.49658535E+00</td>
<td>-0.50002600E-07</td>
</tr>
<tr>
<td>0.8</td>
<td>0.44932896E+00</td>
<td>0.44932901E+00</td>
<td>-0.50899752E-07</td>
</tr>
<tr>
<td>0.9</td>
<td>0.40656966E+00</td>
<td>0.40656971E+00</td>
<td>-0.51173333E-07</td>
</tr>
<tr>
<td>1.0</td>
<td>0.36787944E+00</td>
<td>0.36787949E+00</td>
<td>-0.51173333E-07</td>
</tr>
</tbody>
</table>
Table 5.14

Problem (b) \( y' = 1+y^2, \ y(0)=1 \)
\[ h = 0.1 \]

(i) Implicit GM Runge-Kutta with set 1 parameters

<table>
<thead>
<tr>
<th>( x )</th>
<th>Exact Solution</th>
<th>Numerical Solution</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.10000000E+01</td>
<td>0.12230483E+01</td>
<td>0.49892175E-06</td>
</tr>
<tr>
<td>0.1</td>
<td>0.12230488E+01</td>
<td>0.12230483E+01</td>
<td>0.49892175E-06</td>
</tr>
<tr>
<td>0.2</td>
<td>0.15084976E+01</td>
<td>0.15084933E+01</td>
<td>0.42682405E-05</td>
</tr>
<tr>
<td>0.3</td>
<td>0.18957651E+01</td>
<td>0.18657422E+01</td>
<td>0.22888741E-04</td>
</tr>
<tr>
<td>0.4</td>
<td>0.24649627E+01</td>
<td>0.24648435E+01</td>
<td>0.1197643E-03</td>
</tr>
<tr>
<td>0.5</td>
<td>0.34082234E+01</td>
<td>0.34074512E+01</td>
<td>0.77217186E-03</td>
</tr>
<tr>
<td>0.6</td>
<td>0.53318552E+01</td>
<td>0.53226922E+01</td>
<td>0.91629716E-02</td>
</tr>
</tbody>
</table>

(ii) Implicit GM Runge-Kutta with set 2 parameters

<table>
<thead>
<tr>
<th>( x )</th>
<th>Exact Solution</th>
<th>Numerical Solution</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.10000000E+01</td>
<td>0.12230846E+01</td>
<td>-0.35790336E-04</td>
</tr>
<tr>
<td>0.1</td>
<td>0.12230488E+01</td>
<td>0.12230846E+01</td>
<td>-0.35790336E-04</td>
</tr>
<tr>
<td>0.2</td>
<td>0.15084976E+01</td>
<td>0.15086180E+01</td>
<td>-0.12043595E-03</td>
</tr>
<tr>
<td>0.3</td>
<td>0.18957651E+01</td>
<td>0.18961071E+01</td>
<td>-0.34198753E-03</td>
</tr>
<tr>
<td>0.4</td>
<td>0.24649627E+01</td>
<td>0.24659883E+01</td>
<td>-0.10256155E-03</td>
</tr>
<tr>
<td>0.5</td>
<td>0.34082234E+01</td>
<td>0.34120380E+01</td>
<td>-0.38145947E-02</td>
</tr>
<tr>
<td>0.6</td>
<td>0.53318552E+01</td>
<td>0.53554452E+01</td>
<td>-0.23590057E-01</td>
</tr>
</tbody>
</table>

(iii) Classical implicit Runge-Kutta formula

<table>
<thead>
<tr>
<th>( x )</th>
<th>Exact Solution</th>
<th>Numerical Solution</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.10000000E+01</td>
<td>0.12230498E+01</td>
<td>-0.99542324E-06</td>
</tr>
<tr>
<td>0.1</td>
<td>0.12230488E+01</td>
<td>0.12230498E+01</td>
<td>-0.99542324E-06</td>
</tr>
<tr>
<td>0.2</td>
<td>0.15084976E+01</td>
<td>0.15085008E+01</td>
<td>-0.31865768E-05</td>
</tr>
<tr>
<td>0.3</td>
<td>0.18957651E+01</td>
<td>0.18957735E+01</td>
<td>-0.84381175E-05</td>
</tr>
<tr>
<td>0.4</td>
<td>0.24649627E+01</td>
<td>0.24649856E+01</td>
<td>-0.22881607E-04</td>
</tr>
<tr>
<td>0.5</td>
<td>0.34082234E+01</td>
<td>0.34082968E+01</td>
<td>-0.73393913E-04</td>
</tr>
<tr>
<td>0.6</td>
<td>0.53318552E+01</td>
<td>0.53322250E+01</td>
<td>-0.36982226E-03</td>
</tr>
</tbody>
</table>
CHAPTER SIX

FURTHER APPLICATIONS OF

GM TECHNIQUES
6.1 GENERALISED GM MULTIDERIVATIVE METHODS

6.1.1 Approximation By Rational Functions and A Survey on Pade' Table

A rational function $R(x)$ is defined to be a quotient of polynomials of the form

$$R(x) = \frac{P(x)}{Q(x)} \quad (6.1.1)$$

where we assume that $P(x)$ and $Q(x)$ are reduced to its lowest terms, i.e. $P(x)$ and $Q(x)$ have no common factors. We will now use $R(x)$ to approximate $f(x)$, a continuous function in $[a,b]$ so that we will naturally insist that $Q(x)$ does not vanish in $[a,b]$.

As pointed out in many references (e.g. Johnson and Reiss [1982], p.248), one reason for introducing rational functions as a means to approximate a function $f(x)$ is that ordinary polynomial approximation may not be practical. For example, in order to approximate $f(x)=\arccos(x)$ for $x$ in $[-1,1]$, with an accuracy of $10^{-8}$ it can be shown that a polynomial of degree $10^4$ would be necessary. Furthermore since graphs of rational functions can assume shapes that the graph of a polynomial cannot, rational functions could provide a useful class of approximations.

The most widely used approach to choosing a rational function approximation is that of the Pade' approximation. Suppose $f(x)$ has a Maclaurin's series expansion

$$f(x) = \sum_{i=0}^{\infty} a_i x^i. \quad (6.1.2)$$

In the Pade' approximation, we seek $P(x)$ and $Q(x)$ such that

$$f(x) - \frac{P(x)}{Q(x)} = \sum_{i=s}^{\infty} c_i x^i \quad (6.1.3)$$
where $s$ is as large as possible. It is also known that for a given $f$ and for a fixed order $m$ of $P(x)$ and $n$ of $Q(x)$ the rational approximation $R(x)$ which has the maximum order $m+n$ is unique. By taking $f(z)=e^z$, leads us to forming a table known as the Pade' Table for the exponential function.

This table provides different approximations to $e^z$ entered into a table according to the $(m,k)$th position. In particular the $(m,k)$th entry is the approximant to $e^z$ given by

$$R_{mk}(z) = \frac{P_k(z)}{Q_m(z)} \quad (6.1.4)$$

where $P_k$, $Q_m$ are polynomials defined by

$$P_k(z) = 1 + p_{1,k}z + p_{2,k}z^2 + \ldots + p_{k,k}z^k \quad (6.1.5)$$

and

$$Q_m(z) = 1 - q_{1,m}z + q_{2,m}z^2 - \ldots + (-1)^mq_{m,m}z^m \quad (6.1.6)$$

with

$$p_{1,k} > p_{2,k} > \ldots > p_{k,k} > 0$$

and

$$q_{1,m} > q_{2,m} > \ldots > q_{m,m} > 0.$$  

Since the Taylor series expansion for $e^z$ is given by

$$e^z = 1 + z + \frac{z^2}{2} + \frac{z^3}{6} + \ldots \quad (6.1.7)$$

it follows that

$$1+z, 1+z+z^2, 1+z+\frac{z^2}{2}+\frac{z^3}{6}, \ldots$$

are respectively the $(0,1)$, $(0,2)$, $(0,3)$, ... entries of the Pade' Table. For this example the first integer indicates the order of the polynomial $Q_0(z)=1$, and the second integer indicates the order of the polynomial $P_k(z)$. The $(m,k)$th entries for $m,k=0,1,2,3$ of the Pade' Table is given in Table 6.1.
Table 6.1: (m, k) entries of the Padé Table for m, k = 0, 1, 2, 3.

<table>
<thead>
<tr>
<th>1</th>
<th>1+z</th>
<th>1+z+z^2/2</th>
<th>1+z+z^2/2+z^3/6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-z</td>
<td>1+z/2</td>
<td>1+2z/3+z^2/6</td>
<td>1+3z/4+z^2/4+z^3/24</td>
</tr>
<tr>
<td>1-z+z^2/2</td>
<td>1+z/3</td>
<td>1+z/2+z^2/12</td>
<td>1+3z/5+3z^2/20+z^3/60</td>
</tr>
<tr>
<td>1-z+z^2/2-z^3/6</td>
<td>1+z/4</td>
<td>1+2z/5+z^2/20</td>
<td>1+z/2+z^2/10+z^3/120</td>
</tr>
<tr>
<td>1-z+z^2/2-z^3/6</td>
<td>1-3z/4+z^2/4-z^3/24</td>
<td>1-3z/5+3z^2/20-z^3/60</td>
<td>1-z/2+z^2/10-z^3/120</td>
</tr>
</tbody>
</table>
Now let us consider the problem of solving the initial value problem

\[ y' = f(x, y), \quad y(x_0) = y_0 \] (6.1.8)

by one-step multiderivative methods of the form

\[ y_{n+1} = y_n + \sum_{i=1}^{k} p_{i,k} h^i y_n^{(i)} + \sum_{j=1}^{m} (-1)^{j+1} q_{j,m} h^j y_{n+1}^{(j)} \] (6.1.9)

where the coefficients \( p_{i,k} \) and \( q_{j,m} \) are chosen so that the RHS of (6.1.9) matches with the Taylor series expansion of \( y(x+h) \) to as high an order as possible. It is noted that these coefficients actually coincide with that of the \((m,k)\)th entry of the Pade' Table. An example with \( m=2 \) and \( k=2 \), is a single step implicit double derivative formula given by

\[ y_{n+1} = y_n + \frac{h}{2} (y_n' + y_{n+1}') + \frac{h^2}{12} (y_n'' - y_{n+1}''). \] (6.1.10)

It can be easily deduced that this method is of order 4 with its local truncation error given by

\[ \text{LTE} = -\frac{1}{720} h^5 y_n''(v). \]

In addition, the method is A-stable with the stability polynomial when applied to the model equation \( y' = \lambda y \) given by

\[ R(h\lambda) = \frac{1+h\lambda/2+h^2\lambda^2/12}{1-h\lambda/2+h^2\lambda^2/12} \]

i.e. the \((2,2)\)-entry in the Pade' Table.

With this relation we may call the methods given by (6.1.9) as a class of multiderivative methods based on the \((m,k)\)th entry of the Pade' Table. When the interval of absolute stability is calculated for
the single test equation

\[ y' = \lambda y, \quad y(0) = y_0 \]  \hspace{1cm} (6.1.11)

it is found that we are approximating \( e^{\lambda h} \) in the recurrence relation

\[ y(x+h) = e^{\lambda h} y(x) \]

which the solution of (6.1.11) is seen to satisfy, by

\[ y_{n+1} = R_{m,k}^{(\lambda h)} y_n + O(h^{m+k+1}) \]  \hspace{1cm} (6.1.12)

where \( R_{m,k} \) is the \( (m,k) \) Pade' approximant to \( e^{\lambda h} \).

6.1.2 Inclusion of the nonlinear methods in the Pade' Table -

A Preliminary Approach

As we know all the \((m,k)\) combinations have already exhausted the Pade' Table. This means that other combinations of \( p_i,k \) and \( q_j,m \) in (6.1.9) will lead to a method of lower order than \((k+m)\). But this is not any more the case when the combination of \( y_{n(i)} \) and \( y_{n+1(j)} \) is done in a nonlinear way such as by using the GM approach. In this case the corresponding GM formula which is equivalent to (6.1.9) is given by,

\[
y_{n+1} = y_n + a_1 h y_{n+1}^{(1-p_1)} y_n^{p_1} + a_2 h y_n^{(1-p_2)} y_{n+1}^{p_2} + \ldots \\
+ a_k h y_n^{(1-p_k)} y_{n+1}^{p_k} + a_{k+1} h^{(k+1)} y_{n+1}^{(k+1)} + \ldots + a_m h^{m} y_{n+1}^{(m)} \]  \hspace{1cm} (6.1.13)

More precisely we will use the \( <m,k> \) notation to denote the highest order method obtained by using all the quantities \( y_n \), \( y_n^{(1)} \), \( y_n^{(2)} \), \ldots, \( y_n^{(m)} \) in a combination of the form

\[
y_{n+1} = y_n + \sum_{i=1}^{\min(m,k)} a_i h_i y_n^{(1-p_i)} y_n^{p_i} + \sum_{i=\max(m,k)}^{\min(m,k)} a_i h_i y_n^{(1-p_i)} y_n^{p_i} \]  \hspace{1cm} (6.1.14)
where
\[ s = \begin{cases} 
0 & \text{if } \min(m,k) = m \\
1 & \text{if } \min(m,k) = k.
\end{cases} \]

More specifically, we have the following formulae:

\(<0,1>: \quad y_{n+1} = y_n + a_1 h y_n' \]
\(<1,1>: \quad y_{n+1} = y_n + a_1 h y_n' (1-p_1)(y_{n+1})^{p_1} \]
\(<1,0>: \quad y_{n+1} = y_n + a_1 h y_n' \]
\(<0,2>: \quad y_{n+1} = y_n + a_1 h y_n' + a_2 h^2 y_n'' \]
\(<1,2>: \quad y_{n+1} = y_n + a_1 h y_n' (1-p_1)(y_{n+1})^{p_1} + a_2 h^2 y_n'' \]
\(<2,2>: \quad y_{n+1} = y_n + a_1 h y_n' (1-p_1)(y_{n+1})^{p_1} + a_2 h^2 y_n'' (1-p_2)(y_{n+1})^{p_2} \]

\[ \text{etc.} \]

In each formula we have to determine the parameters \( a_1, p_1 \) so that the formula is of the highest order. In \<(6.1.15-i)\> \( a_1 = 1 \). In \<(6.1.15-ii)\> \( a_1 = 1, p_1 = 1 \). In \<(6.1.15-iii)\> \( a_1 = 1, \quad a_2 = 1 \), etc. We notice that the formulae corresponding to \(<0,1>, \quad <1,0>, \quad <0,2> \) and \(<0,j> \) for all \( j > 2 \) coincide with the formula in the similar position of the Pade' Table. The \(<1,1> \) formula is the GM formula instead of the trapezoidal formula in \((1,1) \) position. This formula has no obvious disadvantages as far as the accuracy and stability are concerned.

As for the other formulae in the \(<m,k> \) class we are in a position to investigate as to whether we can attain the same accuracy as in the equivalent AM formula, and if not whether the stability region is increased, at least.

First, we develop the formula based on the \(<2,2> \) position

\[ y_{n+1} = y_n + a_1 h y_n' (1-p_1)(y_{n+1})^{p_1} + a_2 h^2 y_n'' (1-p_2)(y_{n+1})^{p_2}. \]

\[ (6.1.16) \]
We have
\[ y_{n+1} = y_n + hy_n^\prime + h^2 y_n^{\prime\prime} + \frac{1}{6} h^3 y_n^{(iv)} + \ldots \]

\[ y_{n+1} = y_n + p_1 (1 + \frac{h y_n^\prime}{y_n^\prime} + \frac{h^2 y_n^{\prime\prime}}{2 y_n} + \frac{h^3 y_n^{(iv)}}{6 y_n} + \ldots ) P_1 \]

\[
(1-p_1) \frac{p_1}{y_n^\prime} y_{n+1} = y_n \left[ 1 + \frac{h y_n^\prime}{y_n^\prime} + \frac{h^2 y_n^{\prime\prime}}{2 y_n} + \frac{h^3 y_n^{(iv)}}{6 y_n} + \ldots \right]
\]

\[
+ \frac{p_1 (p_1-1)}{2} \left( \frac{h^2 y_n^{\prime\prime}}{y_n^\prime} + \frac{h^3 y_n^{(v)}}{y_n^\prime} + \ldots \right)
\]

\[
+ \frac{p_1 (p_1-1)(p_1-2)}{6} \left( \frac{h^3 y_n^{(vi)}}{y_n^\prime} + \ldots \right) \] \hspace{1cm} (6.1.17)

Similarly,

\[
(1-p_2) \frac{p_2}{y_n^\prime} y_{n+1} = y_n \left[ 1 + \frac{h y_n^\prime}{y_n^\prime} + \frac{h^2 y_n^{\prime\prime}}{2 y_n} + \frac{h^3 y_n^{(iv)}}{6 y_n} + \frac{h^4 y_n^{(v)}}{24 y_n} + \frac{h^5 y_n^{(vi)}}{120 y_n} + \ldots \right]
\]

\[
+ \frac{p_2 (p_2-1)}{2} \left( \frac{h^2 y_n^{\prime\prime}}{y_n^\prime} + \frac{h^3 y_n^{(v)}}{y_n^\prime} + \frac{h^4 y_n^{(vi)}}{y_n^\prime} + \frac{h^5 y_n^{(vii)}}{y_n^\prime} + \ldots \right) \]

\[
+ \frac{p_2 (p_2-1)(p_2-2)}{6} \left( \frac{h^3 y_n^{(vi)}}{y_n^\prime} + \frac{h^4 y_n^{(vii)}}{y_n^\prime} + \frac{h^5 y_n^{(viii)}}{y_n^\prime} + \ldots \right) \] \hspace{1cm} (6.1.18)

By substituting (6.1.17) and (6.1.18) in (6.1.16) we obtain,

\[
y_{n+1} = y_n + a_1 h [y_n^\prime + \frac{h}{2} y_n^{\prime\prime} + \frac{h^2}{6} y_n^{(iv)} + \frac{h^3}{24} y_n^{(v)} + \frac{h^4}{120} y_n^{(vi)} + \ldots ] + a_2 h^2 [y_n^{\prime\prime} + \frac{h}{2} y_n^{(iv)} + \frac{h^2}{6} y_n^{(v)} + \frac{h^3}{24} y_n^{(vi)} + \ldots ]
\]

or

\[
y_{n+1} = y_n + a_1 [h y_n^\prime + \frac{h}{2} y_n^{\prime\prime} + \frac{h^2}{6} y_n^{(iv)} + \frac{h^3}{24} y_n^{(v)} + \frac{h^4}{120} y_n^{(vi)} + \ldots ]
\]

\[
+ a_2 [h^2 y_n^{\prime\prime} + \frac{h}{2} y_n^{(iv)} + \frac{h^2}{6} y_n^{(v)} + \frac{h^3}{24} y_n^{(vi)} + \ldots ] \] \hspace{1cm} (6.1.19)

By matching (6.1.19) with the Taylor series expansion for $y(x_n + h)$ we have,

\[ h: \quad a_1 = 1 \quad (6.1.20) \]
\[ h^2: \quad a_1 p_1 + a_2 = \frac{1}{1} \quad (6.1.21) \]
\[ h^3: \quad \frac{1}{4} a_1 p_1 + a_2 p_2 = \frac{1}{6} \quad (6.1.22-1) \]
\[ \frac{1}{4} a_1 p_1 (p_1 - 1) = 0 \quad (6.1.22-11) \]

This set of equations has two sets of solutions, namely,

**Set 1:** $a_1 = 1$, $a_2 = \frac{1}{4}$, $p_1 = 0$, $p_2 = \frac{1}{3}$,

**Set 2:** $a_1 = 1$, $a_2 = -\frac{1}{4}$, $p_1 = 1$, $p_2 = \frac{2}{3}$,

which give rise to two 3rd order formulae given by

\[ y_{n+1} = y_n + h y_n' + \frac{1}{4} h^2 y_n'' \left( \frac{2}{3} y_n'' \right)^{1/3} \quad (6.1.23) \]

with local truncation error $y(x_{n+1}) - y_{n+1}$ given by

\[ \text{LTE} = \left( -\frac{1}{24} y_n^{(iv)} + \frac{1}{18} \left( y_n'' \right)^2 \right) h^4 \quad (6.1.24) \]

and

\[ y_{n+1} = y_n + h y_n' - \frac{1}{4} h^2 y_n'' \left( \frac{1}{3} y_n'' \right)^{2/3} \quad (6.1.25) \]

with local truncation error given by,

\[ \text{LTE} = \left( \frac{1}{24} y_n^{(iv)} - \frac{1}{18} \left( y_n'' \right)^2 \right) h^4 \quad (6.1.26) \]

Thus, the derivation of the GM formula leads to the finding of two formulae of order 3 for $\langle 2, 2 \rangle$ against one formula of order 4 for $\langle 2, 2 \rangle$ of the AM schemes.

We also notice that both the above formulae have the same magnitude of truncation error but of opposite sign. Therefore, by adding and dividing by 2 we can produce a single formula of order four
given by
\[ y_{n+1} = y_n + \frac{1}{4} h^2 (y_n^{2/3} y_n^{1/3} - y_n^{1/3}) + o(h^5) \] (6.1.27)

Formula (6.1.27) can be viewed as a combination of the AM formula in the first derivatives and the GM formula in the second derivatives and therefore does not preserve the original form of the proposed formula of (6.1.14). In view of this property we shall now define the nonlinear multiderivative formulae accordingly to consider the involvement of the nonlinear combinations only in the highest derivatives, i.e. the 2nd derivative in the case of formula (6.1.27). This suggests that instead of looking for a formula of the form (6.1.16) we should have directly looked for a formula of the form
\[ y_{n+1} = y_n + h(\alpha_1 y_n' + \beta_1 y_{n+1}') + \alpha_2 y_n^{(1-p)} y_{n+1}^{p} \] (6.1.28)

Definition 6.1

The general single step implicit \(<m,k>\) nonlinear formula is defined by the nonlinear relationship involving \(y_n^{(j)}, y_{n+1}^{(j)}, j = 0, 1, \ldots\) in the form
\[ y_{n+1} = y_n + \sum_{j=1}^{\gamma-1} \frac{1}{j!} h^j \alpha_j y_n^{(j)} + \beta_j y_{n+1}^{(j)} \]
\[ + \sum_{m=1}^{\gamma} \delta_{m,k} h^m \alpha_m y_n^{(m)} + \beta_m y_{n+1}^{(m)} \]
\[ + \delta_{m,k} \alpha_m y_n^{(m)} + \beta_m y_{n+1}^{(m)} \] (6.1.29)

where \( \gamma = \max(m, k) \)
\[ \delta_{m,k} = \begin{cases} 1 & \text{if } m=k \\ 0 & \text{if } m \not= k \end{cases} \]
and \( \alpha_j, \beta_j \) and \( c \) are constants.

This definition has a new implication in that the \((m,k)\) and \(<m,k>\) formulae are necessarily the same formula if \( m \not= k \). For \( m=k \), the \((m,m)\) and \(<m,m>\) formulae differ only in the highest order derivatives. Thus, the \(<m,k>\)th entry of the nonlinear case is essentially similar to the
(m,k)th entry of the Pade' Table. The \( <m,m> \) entries will be obtained from the magnification factor in the stability analysis of the related formula.

Stability of formula (6.1.27)

Applying formula (6.1.27) to the test equation \( y'=\lambda y \) produces

\[
y_{n+1} = y_n + \frac{i}{2}h(\lambda y_n + \lambda y_{n+1}) + \frac{1}{4} h^2 \left( \frac{\lambda^2 y_n}{y_n} \right)^{2/3} \left( \frac{\lambda^2 y_{n+1}}{y_{n+1}} \right)^{1/3} - \left( \frac{\lambda^2 y_n}{y_n} \right)^{1/3} \left( \frac{\lambda^2 y_{n+1}}{y_{n+1}} \right)^{2/3} \]

\[
y_{n+1} = y_n + \frac{i}{2}h\lambda (y_n + y_{n+1}) + \frac{1}{4} h^2 \lambda^2 (y_n^{2/3} y_{n+1}^{1/3} - y_n^{1/3} y_{n+1}^{2/3})
\]

\[
y_{n+1}(1-\frac{i}{2}h\lambda) = y_n(1+i^2h\lambda) + \frac{1}{4} h^2 \lambda^2 (y_n^{2/3} y_{n+1}^{1/3} - y_n^{1/3} y_{n+1}^{2/3})
\]

Now, dividing through by \( y_n \), we obtain

\[
\frac{y_{n+1}}{y_n} = (1-\frac{h\lambda}{2}) + \frac{1}{4} h^2 \lambda^2 \left( \frac{y_{n+1}}{y_n} \right)^{1/3} - \left( \frac{y_{n+1}}{y_n} \right)^{2/3}
\]

(6.1.30)

Letting \( \frac{y_{n+1}}{y_n} = \phi \) and \( h\lambda = z \), (6.1.30) now becomes

\[
(1-\frac{z}{2})\phi = (1 + \frac{z}{2}) + \frac{z}{4} (\phi - \phi^{1/3} - \phi^{2/3})
\]

and taking \( \phi^{1/3} = t \) we have

\[
(1-iz)t^3 = (1+iz) + \frac{1}{4} z^2 (t-t^2)
\]

or,

\[
(1-iz)t^3 + \frac{1}{4} z^2 t^2 - \frac{1}{4} z^2 t - (1+iz) = 0
\]

(6.1.31)

In (6.1.31), it looks rather complicated to express \( t \) explicitly in terms of \( z \), so that we may now avoid finding the stability region directly from the relation \( |t(z)|<1 \). However, since equation (6.1.31) is only quadratic in \( z \), we may conveniently express \( z \) in terms of \( t \), i.e.
We shall now use the boundary locus method to determine the stability region of the formula. First, we determine the boundary $\partial R$ of the stability region $R$ on which $t$ has modulus unity (i.e. $t$ lies on the unit circle of the complex plane). The locus $\partial R$ is determined by setting $t = \cos \theta + i \sin \theta$ for $0 < \theta < 2\pi$ and is shown in Figure 6.1 as the straight line which coincides with the imaginary axis of the complex plane. To examine which side of the complex plane belongs to $R$ we take, for simplicity, $z=2$ in equation (6.1.31) to obtain

$$t^2 - t - 2 = 0$$

i.e. $t = 2, -1$.

Therefore for $z$ lying on the right half-plane, the modulus of $t = \left(\frac{y_{n+1}}{y_n}\right)^{1/3}$ would have a value greater than unity. This means that the required region $R$ is the whole left half-plane, and thus the method is A-stable.

$$z^2 \left(\frac{1}{4} t^2 - \frac{1}{4} t\right) + z(-it^3 - \frac{1}{4}) + (t^3 - 1) = 0$$

from which we obtain

$$z = \frac{1}{4}(t^3+1) \pm \frac{\sqrt{t^6/4-t^5+t^4+t^3/2+t^2-t+1/4}}{4(t^2-1)}$$  \hspace{1cm} \text{(6.1.32)}$$

Figure 6.1: Stability region for $<2,2>$ formula.
As we have mentioned earlier, it was rather complicated to express the magnification factor explicitly in terms of $z$. Thus, while the $<1,1>$ position of the Padé Table is filled with the GM stability polynomial

$$R_{1,1} = \left[ \frac{z + \sqrt{z^2 + 4}}{2} \right]^2$$

(see Section 4.1.4 in Chapter 4), the $R_{22}$ entry is still undetermined explicitly.

Thus up to this stage we have made available the first 3x3 entries of the GM Padé Table as given in Table 6.2.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>1+z</th>
<th>1+z+z^2/2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-z</td>
<td>\left[ \frac{z + \sqrt{z^2 + 4}}{2} \right]^2</td>
<td>\frac{1+2z/3+z^2/6}{1-z/3}</td>
<td></td>
</tr>
<tr>
<td>\frac{1}{1-z+z^2/2}</td>
<td>\frac{1+z/3}{1-2z/3+z^2/6}</td>
<td>to be determined later</td>
<td></td>
</tr>
</tbody>
</table>

**Table 6.2: GM Padé Table**

---

**Example 6.1**

(From Khaliq and Twizell [1985])

\[ y_1' = 0.01 - (0.01 + y_1 + y_2)(y_1^2 + 1001y_1 + 1001), \]

\[ y_2' = 0.01 - (0.01 + y_1 + y_2)(1 + y_2^2), \]

\[ y(0) = (0,0)^T. \]
This problem arises in reactor kinetics. The Jacobian matrix $\frac{\partial f}{\partial y}$ has eigenvalues $-1012$ and $-0.01$ at $x=0$; it thus has an initial stiffness ratio $\approx 10^5$ and may be classified initially as being very stiff.

In this example the second derivative is obtained from the relation

$$y'' = \frac{df}{dx} = \frac{\partial f}{\partial x} \frac{dy_1}{dx} + \frac{\partial f}{\partial y_1} \frac{dy_1}{dx} + \frac{\partial f}{\partial y_2} \frac{dy_2}{dx}$$

i.e.

$$\begin{bmatrix} y_1'' \\ y_2'' \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial y_1} + f_2 \frac{\partial f_1}{\partial y_2} \\ f_1 \frac{\partial f_2}{\partial y_1} + f_2 \frac{\partial f_2}{\partial y_2} \end{bmatrix}$$

or

$$y_1'' = -f_1(0.01+y_1+y_2)(2y_1+1001)+y_1^2+1001y_1+1001$$

and

$$y_2'' = -f_2((0.01+y_1+y_2)(2y_2)+1+y_2^2).$$

The theoretical solution of this problem is not known and was found approximately using the fourth order Runge-Kutta process with small stepsize. (See also Lambert [1973], p.248). The results obtained by using the $<2,2>$ and $(2,2)$ formulae with $h=0.001$ and iterating to machine accuracy are shown in Table 6.3. There is no significant difference between the two results and both formulae appear to perform equally well. Therefore, the nonlinear formula is equally competitive as the linear formula both in terms of accuracy and stability.
Table 6.3:

Example 6.1 (Khaliq & Twizell [1985]).
h=0.001, k=10 (result after 10 steps)
Using iteration to convergence.

(a) by using $<2,2>$ formula

\[ x = 0.01 \]

<table>
<thead>
<tr>
<th></th>
<th>$y_1$</th>
<th>$y_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact Solution</td>
<td>-0.1006914E-01</td>
<td>0.8978912E-04</td>
</tr>
<tr>
<td>Numerical Solution</td>
<td>-0.1006915E-01</td>
<td>0.8978918E-04</td>
</tr>
<tr>
<td>Error</td>
<td>-0.8440052E-08</td>
<td>0.5764432E-10</td>
</tr>
</tbody>
</table>

Computing time = 0.977 sec.

(b) by using $(2,2)$ formula

\[ x = 0.01 \]

<table>
<thead>
<tr>
<th></th>
<th>$y_1$</th>
<th>$y_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact Solution</td>
<td>-0.1006914E-01</td>
<td>0.8978912E-04</td>
</tr>
<tr>
<td>Numerical Solution</td>
<td>-0.1006913E-01</td>
<td>0.8978815E-04</td>
</tr>
<tr>
<td>Error</td>
<td>0.8070134E-08</td>
<td>-0.9697208E-09</td>
</tr>
</tbody>
</table>

Computing time = 0.656 sec.

6.1.3 Implementation of formula (6.1.27) in PECE-mode of the predictor-corrector procedure

Solving an IVP $y' = f(x,y)$, $y(x_0) = y_0$ by using formula (6.1.27) will result in solving a nonlinear algebraic equation of the form

\[ y_{n+1} = g(y_{n+1}) \]  

(6.1.33)

which in general cannot be solved exactly. Iterative methods will normally give satisfactory results if the condition for convergence is
satisfied in the equation
\[ y_{n+1}^{(k+1)} = g(y_{n+1}^{(k)}), \quad k=0,1,\ldots. \]

However if the starting value \( y_{n+1}^{(0)} \) is chosen reasonably accurately, the convergence is greatly enhanced, and in most cases a single iteration would be sufficient.

Since the values of \( y_n, y_n', \) and \( y_n'' \) are to be made available for use in formula (6.1.27) we might as well use these values to compute a reasonably close initial guess \( y_{n+1}^{(0)} \) to the solution of (6.1.33). The most accurate explicit formula which uses only these values is the 2nd order truncated Taylor series formula
\[ y_{n+1} = y_n + h y_n' + \frac{1}{2} h^2 y_n'' \]
(6.1.34)
i.e. the formula in the (0,2) entry of the Pade' Table.

The PECE-mode of the predictor-corrector combination for formula (6.1.27) is therefore given by

\[ f_n = f(x_n, y_n), \quad f_n' = f'(x_n, y_n) \]
\[ P: \quad y_{n+1}^{(0)} = y_n + hf_n + \frac{1}{2} h^2 f_n' \]
(6.1.35-a)
\[ E: \quad f_{n+1} = f(x_{n+1}, y_{n+1}^{(0)}), \quad f_{n+1}' = f'(x_{n+1}, y_{n+1}^{(0)}) \]
(6.1.35-b)
\[ C: \quad y_{n+1}^{(1)} = y_n + \frac{1}{2} h(f_n + f_{n+1}') + \frac{1}{4} h^2 (f_n' + \frac{1}{3} f_{n+1}' + \frac{1}{3} f_n) \]
(6.1.35-c)
\[ E: \quad f_{n+1} = f(x_{n+1}, y_{n+1}^{(1)}), \quad f_{n+1}' = f'(x_{n+1}, y_{n+1}^{(1)}) \]
(6.1.35-d)
\[ y_{n+1} = y_n + 1/3 \left( f_{n+1} + f_{n+1}' \right) \]

Proceed to the next integration step.
We will identify this process as the \((0,2)-\{2,2\}\) PECE combination. The equivalent linear formula is denoted by the \((0,2)-(2,2)\) combination and was studied by Khaliq and Twizell [1985].

**Example 6.2**

We repeat example 6.1 by using PECE-mode, i.e. allowing only one iteration at each step. The results are shown in Table 6.4, and again both formulae appear to perform reasonably well in terms of accuracy. In addition, the computing times now compare better than in the case of iterating to convergence.

<table>
<thead>
<tr>
<th>Table 6.4: Results for Example 6.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>h=0.001, k=10 (result after 10 steps)</td>
</tr>
<tr>
<td>Using predictor corrector method in PECE-mode</td>
</tr>
<tr>
<td>(a) by using ((0,2)-{2,2}) combination</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Solution at (x = 0.01)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(y_1)</td>
</tr>
<tr>
<td>Exact Solution</td>
</tr>
<tr>
<td>Numerical Solution</td>
</tr>
<tr>
<td>Error</td>
</tr>
<tr>
<td>Computing time = 0.042sec.</td>
</tr>
</tbody>
</table>

(b) by using \((0,2)-(2,2)\) combination

<table>
<thead>
<tr>
<th>Solution at (x = 0.01)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(y_1)</td>
</tr>
<tr>
<td>Exact Solution</td>
</tr>
<tr>
<td>Numerical Solution</td>
</tr>
<tr>
<td>Error</td>
</tr>
</tbody>
</table>

Computing time = 0.031sec.
6.1.4 Comparison of the stability regions of the (0,2)-<2,2> and (0,2)-(2,2) combinations in PECE mode

(a) (0,2)-<2,2> combination

Applying formulae (5.1.35) to the test equation \( y' = \lambda y \) produces the following recurrence,

\[
P: \quad y_{n+1}^* = (1 + h \lambda + \frac{h^2 \lambda^2}{2}) y_n
\]

\[
C: \quad y_{n+1} = y_n + \frac{h}{4} (y_n + y_{n+1}^* + 1 \cdot \frac{(y_n + y_{n+1})}{1/3} - y_n y_{n+1})
\]

or

\[
\frac{y_{n+1}}{y_n} = 1 + h \lambda + \frac{h^2 \lambda^2}{2} + \frac{h^3 \lambda^3}{4}
\]

\[
+ \frac{h^2}{4} \left[ \lambda^2 (1 + h \lambda + \frac{h^2 \lambda^2}{2})^{1/3} - \lambda^2 (1 + h \lambda + \frac{h^2 \lambda^2}{2})^{2/3} \right]
\]

Putting \( h \lambda = z \) we have

\[
\frac{y_{n+1}}{y_n} = 1 + z + \frac{z^2}{2} + \frac{z^3}{4}
\]

\[
+ \frac{z^2}{4} \left[ (1 + z + \frac{z^2}{2})^{1/3} - (1 + z + \frac{z^2}{2})^{2/3} \right]
\]

The set of values of \( z \) for which \( |y_{n+1}/y_n| < 1 \) gives the stability region of the formula and is shown in Figure 6.2 as the region bounded by the inner closed curve.
Figure 6.2: Stability regions for $(0,2)-(2,2)$ and $(0,2)-\langle 2,2 \rangle$ combinations in the PECE mode.
(b) (0,2)-(2,2) combination

P: \[ y_{n+1}^* = (1+h\lambda + \frac{1}{2}h^2\lambda^2)y_n \]

C: \[ y_{n+1} = y_n + \frac{1}{2}h(y_n^* + y_{n+1}^*) + \frac{1}{12} h^2(y_n y_{n+1} - y_{n+1}) \]

\[ y_{n+1} = y_n + \frac{1}{2}h[\lambda y_n + \lambda(1+h\lambda + \frac{1}{2}h^2\lambda^2)y_n] \]

\[ + \frac{1}{12} h^2[\lambda^2 y_n - \lambda^2(1+h\lambda + \frac{1}{2}h^2\lambda^2)y_n] \]

\[ \frac{y_{n+1}}{y_n} = 1 + h\lambda + \frac{1}{2}h^2\lambda^2 + \frac{1}{4} h^3\lambda^3 - \frac{1}{12} h^3\lambda^3 - \frac{1}{24} h^4\lambda^4 \]

The set of values of $h\lambda = z$ for which $|y_{n+1}/y_n| < 1$, gives the stability region of the related formula. The region is shown in Figure 6.2 bounded by the outer closed curve. In comparison, the stability region for the linear formula is larger than its nonlinear counterpart. However, they are similar on the real line, i.e. within the interval [-2,0].
6.2 GM MULTISTEP METHODS

6.2.1 GM 2-step formulae

Due to the complexity in the algebra we will restrict our discussion in this section to developing the 2-step formulae in the GM sense only.

To begin with we consider the application of the GM formula over an interval of length 2h. This will result in the following recurrence:

\[ y_{n+1} = y_{n-1} + 2h \sqrt{f_{n-1}f_{n+1}} \]  \hspace{1cm} (6.2.1)

We may, by experiment, replace \(2 \sqrt{f_{n-1}f_{n+1}}\) by \(\sqrt{f_{n-1}f_n} + \sqrt{f_n f_{n+1}}\) to produce the following new formula:

\[ y_{n+1} = y_{n-1} + h \left( \sqrt{f_{n-1}f_n} + \sqrt{f_n f_{n+1}} \right) \]  \hspace{1cm} (6.2.2)

In the following we will determine the order of this formula and see if any modification can be done to increase its accuracy.

Error Analysis of formula (6.2.2)

We shall now express the RHS of equation (6.2.2) as a Taylor series expansion about \(y\). By Taylor series expansion we have

\[
\sqrt{y_{n-1}} = \sqrt{y_n - hy_n'' + \frac{1}{2} h^2 y_n^{'''} - \frac{1}{6} h^3 y_n^{(iv)}}
\]

\[
= \sqrt{y_n'} \sqrt{1 + \left( -\frac{hy_n''}{y_n'} + \frac{h^2 y_n^{'''}}{2y_n'} - \frac{h^3 y_n^{(iv)}}{6y_n'} + \ldots \right)}
\]

\[
= \sqrt{y_n'} \left[ 1 + \frac{-hy_n''}{y_n'} + \frac{h^2 y_n^{'''}}{2y_n'} - \frac{h^3 y_n^{(iv)}}{6y_n'} + \ldots \right]
\]

\[ - \frac{1}{8} \left( -\frac{hy_n''}{y_n'} + \frac{h^2 y_n^{'''}}{2y_n'} - \frac{h^3 y_n^{(iv)}}{6y_n'} + \ldots \right)^2
\]

\[ + \frac{1}{16} \left( -\frac{hy_n''}{y_n'} + \ldots \right)^3 + \ldots \]
\[
\sqrt{y_n'} = [1 - \frac{1}{4} \frac{h y''}{y'} + h^2 \left( \frac{y'''}{4y'} - \frac{y_n^2}{8y_n^2} \right) + \ldots]
\]

\[
+ h^3 \left[ -\frac{y_n^2}{12y_n} - \frac{1}{8} \left( -\frac{y'' y'''}{y_n'} - \frac{y_n^3}{16 y_n'^3} \right) + \ldots \right]
\]

Therefore,

\[
\sqrt{y_n'y_{n-1}} = y_n' - \frac{1}{2} h y'' + h^2 \left[ \frac{1}{2} y''' - \frac{1}{8} \frac{y_n^2}{y_n'} \right]
\]

\[
+ h^3 \left[ -\frac{1}{12} y_n^{(iv)} + \frac{1}{8} \frac{y_n y''''}{y_n'} - \frac{1}{16} \frac{y_n^3}{y_n'^3} \right] + \ldots
\]

(6.2.3)

From Chapter 4, we already have

\[
\sqrt{y_n'y_{n+1}} = y_n' + \frac{1}{2} h y'' + h^2 \left( \frac{1}{2} y''' - \frac{y_n^2}{8y_n'} \right)
\]

\[
+ h^3 \left( \frac{1}{12} y_n^{(iv)} - \frac{y_n y''''}{8y_n'} + \frac{y_n^3}{16 y_n'^3} \right) + \ldots
\]

(6.2.4)

Adding (6.2.4) and (6.2.5)

\[
\sqrt{y_n'y_{n+1}} + \sqrt{y_n'y_{n+1}} = 2y_n' + 2h^2 \left( \frac{1}{2} y''' - \frac{y_n^2}{8y_n'} \right) + O(h^4)
\]

(6.2.6)

Equation (6.2.2) then becomes

\[
y_{n+1} = y_{n-1} + 2h y_n' + 2h^3 \left( \frac{1}{2} y''' - \frac{y_n^2}{8y_n'} \right) + \ldots
\]

(6.2.7)

Now, the Taylor series expansion of \( y(x_{n+1}) \) about \( x_{n-1} \) is given by

\[
y(x_{n+1}) = y_{n-1} + 2h y_n' + 2h^2 y_n'' + \frac{4}{3} h^3 y_n''' + \ldots
\]

or

\[
y(x_{n+1}) = y_{n-1} + 2h y_n' + 0 h^2 y_n'' + \frac{1}{3} h^3 y_n''' + \ldots
\]

(6.2.8)
Therefore, by comparing (6.2.7) and (6.2.8) we see that formula (6.2.2) is of second order with its local truncation error given by

\[
\text{LTE} = \frac{1}{3} h^3 y_n''' - 2h^3 \left( \frac{1}{4} y_n'' + \frac{y_n'}{y_n'} \right)
\]

\[
= h^3 \left[ -\frac{1}{6} y_n'' + \frac{1}{4} \frac{y_n'}{y_n'} \right].
\]

(6.2.9)

6.2.2 Modification of formula (6.2.2)

From the previous analysis it is interesting to note that the expressions \( \sqrt{y_{n-1} y_{n+1}} \) and \( \sqrt{y_n y_{n+1}} \) have opposite signs in their terms involving odd powers of \( h \), so that their sum cancels these terms out leaving only even powers in \( h \) as shown in equation (6.2.6). Consequently formula (6.2.2) involves only odd powers in \( h \). In such a case, then provided we have sufficient information, we might be able to adjust the 3rd order term to match the Taylor series expansion (6.2.8).

We try to put in additional information by evaluating \( \sqrt{y_{n-1} y_{n+1}} \).

Expanding, we have

\[
\sqrt{y_{n-1} y_{n+1}} = (y_n + hy_n'' + \frac{1}{2} h^2 y_n''' - \frac{1}{6} h^3 y_n^{(iv)} + \ldots)(y_n + hy_n'' + \frac{1}{2} h^2 y_n''' + \frac{1}{6} h^3 (y_n^{(iv)} + \ldots))^\frac{1}{2}
\]

\[
= [y_n + hy_n'' + \frac{1}{2} h^2 y_n''' + \frac{1}{6} h^3 (y_n^{(iv)} + \ldots)]^\frac{1}{2}
\]

\[
= y_n \left[ 1 + h^2 \left( -\frac{y_n''}{y_n'} + \frac{y_n'}{y_n'} \right) + h^4 \left( \frac{1}{4} \frac{y_n''}{y_n'}^2 \right) + \ldots \right]^{\frac{1}{2}}
\]

\[
= y_n \left[ 1 + \frac{y_n''}{y_n'} \left( \frac{y_n''}{y_n'} + \frac{y_n'}{y_n'} \right) + \frac{1}{4} \frac{y_n''}{y_n'}^2 + \ldots \right]^{\frac{1}{2}}
\]

\[
= y_n \left[ 1 + \left( h^2 \left( -\frac{y_n''}{y_n'} + \frac{y_n'}{y_n'} \right) + h^4 \left( \frac{1}{4} \frac{y_n''}{y_n'}^2 \right) + \ldots \right) \right]^{\frac{1}{2}}
\]

\[
= y_n \left[ 1 + \frac{y_n''}{y_n'} \left( \frac{y_n''}{y_n'} + \frac{y_n'}{y_n'} \right) + \frac{1}{4} \frac{y_n''}{y_n'}^2 + \ldots \right]^{\frac{1}{2}}
\]
At this stage let us recapitulate all the expansions that we have already had:

\[ A(1) \quad h y'_{n-1} y_n = h y'_{n} - \frac{1}{2} h^2 y''_{n} + \frac{1}{4} h^3 (\frac{1}{8} y''_{n}) + \ldots \]

\[ A(2) \quad h y'_{n} y'_{n+1} = h y'_{n} + \frac{1}{2} h^2 y''_{n} + \frac{1}{4} h^3 (\frac{1}{8} y''_{n}) - \frac{1}{8} y''_{n} + \ldots \]

\[ A(3) \quad h y'_{n-1} y'_{n+1} = h y'_{n} + 0 + \frac{1}{2} h^3 (\frac{1}{2} y'''_{n}) - \frac{1}{2} \frac{1}{2} y'''_{n} + \ldots \]

Also it is necessary that we have a fourth equation to enable us to make adjustment in the four terms. Thus we may include the trapezoidal formula as well, i.e.

\[ A(4) \quad \frac{1}{2} h (y'_{n} + y'_{n+1}) = h y'_{n} + \frac{1}{2} h^2 y''_{n} + \frac{1}{6} h^3 y'''_{n}. \]

Now, our aim is to make a linear combination of A(1), A(2), A(3) and A(4) to match the Taylor series expansion of \( y(x_n+h) \) in the multistep form

\[ y_{n+1} = y_n + h[\alpha \sqrt{y'_{n-1} - y'_n} + \beta \sqrt{y'_{n} y'_{n+1}} + \gamma \sqrt{y'_{n-1} y'_n} + \frac{\delta}{2} (y'_{n} + y'_{n+1}) \] (6.2.11)

The matching leads to the following system of linear equation.

\[ h y'_{n}: \quad \alpha + \beta + \gamma + \delta = 1 \] (6.2.12-I)

\[ h^2 y'_{n}: \quad -\frac{\alpha}{2} + \frac{\beta}{2} + 0x\gamma + \delta = \frac{1}{4} \] (6.2.12-II)

\[ h^3 y'_{n}: \quad \frac{\alpha}{4} + \frac{\beta}{4} + \gamma + \frac{\delta}{4} = \frac{1}{6} \] (6.2.12-III)

\[ h^3 \frac{y''_{n}}{y'_{n}}: \quad -\frac{\alpha}{8} = \frac{\beta}{8} - \frac{\gamma}{2} + 0x\delta = 0 \] (6.2.12-IV)
The solution of this system is

\[ \alpha = \frac{1}{6}, \beta = \frac{7}{6}, \gamma = -\frac{1}{3}, \delta = 0. \]

This means that the fourth equation was not necessary to obtain the third order matching of the error terms. The modified version of the GM 2-step formula is therefore given by

\[
y_{n+1} = y_n + \frac{1}{6} h \left[ \sqrt{y_{n-1}' y_n'} + 7 \sqrt{y_n' y_{n+1}'} \right]
\]

\[
- 2 \sqrt{y_{n-1}' y_{n+1}'} \] (6.2.13)

which is of order 3. We also note that this formula does not need any extra function evaluations had the formula taken the form of equation (6.2.2).

Numerical Example

We use formula (6.2.13) to solve the initial value problem

\[ y' = -y, \quad y(0) = 1, \]

and compare its result with the one obtained by using the trapezoidal and the GM formulae. The results given in Table 6.5 show that our GM 2-step formula is more accurate, thus confirming its 3rd order accuracy.

In term of computing time we have adopted a time-saving procedure in all the three methods in which values already available are saved until not needed and will not be recomputed. The results show that only a slight increase in computing time is needed by formula (6.2.13) to give a favourable result.
Table 6.4

\[ y' = -y, \ y(0) = 1 \]
\[ h = 0.1, \ k = 1 \]

<table>
<thead>
<tr>
<th>x</th>
<th>Exact Solution</th>
<th>Error (2-step GM)</th>
<th>Error (Trapezoidal)</th>
<th>Error (GM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>0.9048374E+00</td>
<td>-0.1958088E-05</td>
<td>-0.7551930E-04</td>
<td>0.3765396E-04</td>
</tr>
<tr>
<td>0.20</td>
<td>0.8187308E+00</td>
<td>-0.3523090E-05</td>
<td>-0.1366540E-03</td>
<td>0.6814848E-04</td>
</tr>
<tr>
<td>0.30</td>
<td>0.7408182E+00</td>
<td>-0.4773654E-05</td>
<td>-0.1854653E-03</td>
<td>0.9249836E-04</td>
</tr>
<tr>
<td>0.40</td>
<td>0.6703201E+00</td>
<td>-0.5752588E-05</td>
<td>-0.2237426E-03</td>
<td>0.1115996E-03</td>
</tr>
<tr>
<td>0.50</td>
<td>0.6065307E+00</td>
<td>-0.6499358E-05</td>
<td>-0.2530484E-03</td>
<td>0.1262314E-03</td>
</tr>
<tr>
<td>0.60</td>
<td>0.5488116E+00</td>
<td>-0.7056531E-05</td>
<td>-0.2747512E-03</td>
<td>0.1370640E-03</td>
</tr>
<tr>
<td>0.70</td>
<td>0.4965853E+00</td>
<td>-0.7447586E-05</td>
<td>-0.2900272E-03</td>
<td>0.1446975E-03</td>
</tr>
<tr>
<td>0.80</td>
<td>0.4493290E+00</td>
<td>-0.7700625E-05</td>
<td>-0.2999048E-03</td>
<td>0.1496308E-03</td>
</tr>
<tr>
<td>0.90</td>
<td>0.4065697E+00</td>
<td>-0.7835465E-05</td>
<td>-0.3052705E-03</td>
<td>0.1523211E-03</td>
</tr>
<tr>
<td>1.00</td>
<td>0.3678794E+00</td>
<td>-0.7878570E-05</td>
<td>-0.3069003E-03</td>
<td>0.1531413E-03</td>
</tr>
</tbody>
</table>

Average computing time per step:

- 2-step GM formula: 9469 \( \mu \)-sec.
- Trapezoidal formula: 8377 \( \mu \)-sec.
- GM formula: 9393 \( \mu \)-sec.

Stability of formula (6.2.13)

We apply the formula to the test equation \( y' = \lambda y \) to produce

\[
y_{n+1} = y_n + \frac{1}{6} h\lambda \left[ \sqrt{y_{n-1}y_n} + 7 \sqrt{y_ny_{n+1}} - 2 \sqrt{y_{n-1}y_{n+1}} \right]
\]

(6.2.14)

Let the solution of this difference equation be \( y_n = \mu^n \). Substituting this into (6.2.14) we obtain

\[
\mu^{n+1} = \mu^n + \frac{1}{6} h\lambda \left[ \sqrt{\mu^{n-1}\mu} + 7 \sqrt{\mu\mu^{n+1}} - 2 \sqrt{\mu^{n-1}\mu^{n+1}} \right]
\]

\[
= \mu^n + \frac{1}{6} h\lambda \left[ \mu^{n+\frac{1}{2} + 7\mu^{n+\frac{1}{2}} - 2\mu^n} \right]
\]

\[
\mu^{n+1} - \mu^n - \frac{1}{6} h\lambda \left[ \mu^{n+\frac{1}{2} + 7\mu^{n+\frac{1}{2}} - 2\mu^n} \right] = 0
\]

\[
\mu^n \left[ \mu - 1 - \frac{1}{6} h\lambda \left[ \mu^{n+\frac{1}{2}} + 7\mu^{n+\frac{1}{2}} - 2 \right] \right] = 0,
\]

i.e. \( \mu = 0 \) is a root with multiplicity \( n \).
Next, we want to determine the roots of
\[
\mu - 1 - \frac{1}{6} h\lambda [\mu^{-1/2} + 7\mu^{-1} - 2] = 0 \tag{6.2.15}
\]

We observe that \(\mu = 0\) is not a root of this equation. Let \(\mu^2 = v\).

Substituting into (6.2.15) we obtain
\[
v^2 - 1 - \frac{1}{6} h\lambda [v^{-1/2} + 7v^{-1} - 2] = 0
\]

Multiplying by \(v\) throughout we obtain
\[
v^3 - v - \frac{1}{6} h\lambda [1 + 7v^2 - 2v] = 0 \tag{6.2.16}
\]

or
\[
v^3 - \frac{7}{6} h\lambda v^2 + (\frac{1}{3} h\lambda - 1)v - \frac{1}{6} h\lambda = 0 \tag{6.2.17}
\]

Now, we shall find for what values of \(h\lambda\) will this equation have roots satisfying \(|v| < 1\).

From (6.2.16) we have
\[
v^3 - v = h\lambda [\frac{1 + 7v^2 - 2v}{6}]
\]

By the boundary locus method, the boundary of this region, \(R\), is determined by
\[
h(\theta) = \frac{\rho}{\sigma} = \frac{6[\exp(3i\theta) - \exp(i\theta)]}{1 + 7\exp(2i\theta) - 2\exp(i\theta)}
\]

\[0 \leq \theta \leq 2\pi\]

This region is enclosed in the curve shown in Figure 6.3. From this region, it is seen that the method is suitable for solving problems with complex eigenvalues such that \(h\lambda\) lies in \(R\).
Figure 6.3: Stability region of formula (6.2.13)
6.3 APPLICATION OF THE GM FORMULA TO THE PROBLEMS OF THE FORM
\[ y' = \lambda(x)y \]

6.3.1 Gourlay's Method

Despite its low order accuracy the Trapezoidal method has been proved to perform well when applied to most moderately stiff problems. This is due to the fact that it possesses the important A-stability property. Unfortunately, however, as in many other A-stable one-step methods, the magnification factor \( Q(h\lambda) \) of the Trapezoidal method is such that

\[ Q(h\lambda) \to 1 \text{ as } \text{Re}(h\lambda) \to -\infty, \]

so that the numerical approximations to the rapidly decaying eigensolutions with very small time constants may decay only very slowly.

Gourlay [1970] shows that the application of the Trapezoidal formula

\[ y_{n+1} = y_n + \frac{1}{2}h[f(x_n, y_n) + f(x_{n+1}, y_{n+1})] \quad (6.3.1) \]

for solving certain types of problems, namely \( y' = \lambda(x)y \), leads to an undesirable property in the results. In particular, he shows that for certain functions \( \lambda(x) \), the stability requirement imposes a restriction on the mesh length \( h \) to satisfy

\[ h[\lambda(x_n) - \lambda(x_{n+1})] \leq 4 \quad (6.3.2) \]

Condition (6.3.2) is certainly satisfied if \( \lambda(x_n) \leq \lambda(x_{n+1}) \) but, if \( \lambda(x_n) > \lambda(x_{n+1}) \), then (6.3.2) restricts the mesh size to lie in the interval

\[ 0 < h \leq 4[\lambda(x_n) - \lambda(x_{n+1})]^{-1} \quad (6.3.3) \]
An alternative strategy to remove this restriction was provided by replacing (6.3.1) by

\[ y_{n+1} = y_n + hf(x_{n+1}, y_n + y_{n+1}) \]  

(6.3.4)

where the two function values are evaluated at the same point \( x = x_{n+1} = x_n + h/2 \). Method (6.3.4) requires twice as many function evaluations of the Trapezoidal method to carry out the integration of the differential equation over the interval \( x_0 \leq x \leq X \). To remove this latter defect Gourlay suggests that (6.3.1) be replaced by the method

\[ y_{n+1} = y_n + hf(x_{n+1}, y_n + y_{n+1}) \]  

(6.3.5)

which has the same order of accuracy as the normal Trapezoidal rule. Unlike the Trapezoidal formula given by (6.3.1), Gourlay's method given by (6.3.5) is always stable for equation \( y' = \lambda(x)y \).

6.3.2 GM Alternative

In a similar way to the Gourlay's approach, we shall now demonstrate (if any) the defect of the GM formula, i.e.

\[ y_{n+1} = y_n + h(\lambda(x) \frac{y_n + y_{n+1}}{2}) \]  

(6.3.6)

by applying it to the equation

\[ y' = \lambda(x)y, \ \lambda(x) \leq 0. \]  

(6.3.7)

For simplicity we denote \( \lambda(x_n) \) by \( \lambda_n \) and by applying (6.3.6) to (6.3.7) we obtain

\[ y_{n+1} = y_n + h(\lambda_n \frac{y_n + y_{n+1}}{2}) \]

\[ = y_n + h \frac{\lambda_n \lambda_n + y_{n+1}}{2y_n y_{n+1}}. \]

Dividing both sides of this equation by \( y_n \) (assuming \( y_n \neq 0 \)), we obtain
\[
\frac{y_{n+1}}{y_n} = 1 + h \sqrt{\frac{\lambda_n}{\lambda_{n+1}}} \sqrt{\frac{y_{n+1}}{y_n}}
\]

i.e.

\[
\left( \frac{y_{n+1}}{y_n} - 1 \right)^2 = h^2 \frac{\lambda_n}{\lambda_{n+1}} \left( \frac{y_{n+1}}{y_n} \right)
\]

or

\[
\left( \frac{y_{n+1}}{y_n} \right)^2 - (2 + h^2 \frac{\lambda_n}{\lambda_{n+1}}) \frac{y_{n+1}}{y_n} + 1 = 0 \quad (6.3.8)
\]

This is a quadratic equation in \( \frac{y_{n+1}}{y_n} \) and can be solved to obtain

\[
\frac{y_{n+1}}{y_n} = \frac{2 + h^2 \frac{\lambda_n}{\lambda_{n+1}} \pm \sqrt{(2 + h^2 \frac{\lambda_n}{\lambda_{n+1}})^2 - 4}}{2}
\]

\[
= 1 + \frac{1}{2} \left[ h^2 \frac{\lambda_n}{\lambda_{n+1}} \pm \sqrt{h^2 \frac{\lambda_n}{\lambda_{n+1}} (4 + h^2 \frac{\lambda_n}{\lambda_{n+1}})} \right] \quad (6.3.9)
\]

Now, for recurrence (6.3.9) to be acceptable, we require \( \frac{y_{n+1}}{y_n} \leq 1 \), i.e.

\[
1 + \frac{1}{2} \left[ h^2 \frac{\lambda_n}{\lambda_{n+1}} \pm \sqrt{h^2 \frac{\lambda_n}{\lambda_{n+1}} (4 + h^2 \frac{\lambda_n}{\lambda_{n+1}})} \right] \leq 1 \quad (6.3.10)
\]

Inequality (6.3.10) is satisfied if \( h \sqrt{\frac{\lambda_n}{\lambda_{n+1}}} < 0 \), i.e. by taking the negative sign in the square root of (6.3.10). This condition is satisfied if \( \lambda(x) \) is a negative function, which is in fact the case.

Therefore, there is no restriction on the stepsize as far as the solution of \( y' = \lambda(x)y \) is concerned, by using the GM formula. Unlike the Trapezoidal formula, a modification in the evaluation of the function should not be necessary. This property follows immediately from the fact that the GM formula is L-stable as opposed to the Trapezoidal formula which is only A-stable. (We have established this in Section 4.1.4). However, due to the simplicity in the form of the modified formula as well as its better accuracy when applied to certain problems, as we will see later, we also give the modified formula in the form,
\[ y_{n+1} = y_n + hf(x_{n+1}, \sqrt{y_n y_{n+1}}) \quad (6.3.11) \]

which has the same order of accuracy as formula (6.3.6).

**Numerical Example**

We use the example given by Gourlay [1970] to demonstrate the defect of the Trapezoidal formula as well as to compare the performance of our formula with Gourlay's formula. The problem is given by

\[ y' = \lambda(x)y, \quad y(0) = 1 \quad (6.3.12) \]

where

\[ \lambda(x) = \begin{cases} \alpha^2(x-\beta) & \text{for } 0 \leq x \leq \beta \\ 0, & x \geq \beta \end{cases} \quad (6.3.13) \]

The solution to this problem is given by

\[ y(x) = \begin{cases} \exp\{-\alpha^2x(\beta - x/2)\}, & 0 \leq x \leq \beta \\ \exp\{-\alpha^2\beta^2/2\}, & x \geq \beta. \end{cases} \quad (6.3.14) \]

The stability condition in this case requires \( h \leq 2/\alpha \) for the Trapezoidal formula, but there is no restriction on \( h \) if the GM or the Gourlay's formulae are to be used.

In this example we compute the solution for the choice of constants

\[ \alpha = 10, \quad \beta = 1, \]

and for the values of \( h=0.1, 0.01 \). The stepsize requirement for the Trapezoidal method is \( h \leq 0.2 \) for \( 0 \leq x \leq 1 \). However, in practice, as our computation shows, even with \( h=0.1 \) the results demonstrate nondamping
of the solution of all the formulae used. The numerical results obtained by using all the four methods are shown in Table 6.6. As can be seen, the worse affected is the Trapezoidal formula, followed by Gourlay's formula and then the GH formula and finally our new formula given by (6.3.11). This is judged from the speed at which the errors grow. For smaller h, however, the accuracy of the results obtained by these four methods are about the same.

Table 6.6

(a) h = 0.1, k=1

<table>
<thead>
<tr>
<th>x</th>
<th>Exact Solution</th>
<th>Error (Trapezoidal)</th>
<th>Error (Gourlay)</th>
<th>Error (GH)</th>
<th>Error (Modified GH)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>0.7485183E-04</td>
<td>0.8507095E+37</td>
<td>0.1701412E+38</td>
<td>-0.8798871E+02</td>
<td>-0.8823874E+02</td>
</tr>
<tr>
<td>0.20</td>
<td>0.1522996E-07</td>
<td>-0.7025615E+18</td>
<td>-0.4683744E+18</td>
<td>0.6157947E+04</td>
<td>0.6197510E+04</td>
</tr>
<tr>
<td>0.30</td>
<td>0.8423464E-11</td>
<td>-0.8507059E+37</td>
<td>-0.1701412E+38</td>
<td>-0.3324151E+06</td>
<td>-0.3361006E+06</td>
</tr>
<tr>
<td>0.40</td>
<td>0.1266417E-13</td>
<td>-0.4636708E+18</td>
<td>0.4593672E+18</td>
<td>0.1328829E+08</td>
<td>0.1351970E+08</td>
</tr>
<tr>
<td>0.50</td>
<td>0.5175558E-16</td>
<td>0.8507059E+37</td>
<td>0.1701412E+38</td>
<td>-0.3719696E+09</td>
<td>-0.3814522E+09</td>
</tr>
<tr>
<td>0.60</td>
<td>0.5749522E-18</td>
<td>0.1170936E+19</td>
<td>-0.4683744E+18</td>
<td>0.6668035E+10</td>
<td>0.6940538E+10</td>
</tr>
<tr>
<td>0.70</td>
<td>0.1736205E-19</td>
<td>-0.8507059E+37</td>
<td>-0.1701412E+38</td>
<td>-0.600674E+11</td>
<td>-0.7045682E+11</td>
</tr>
<tr>
<td>0.80</td>
<td>0.1425164E-20</td>
<td>overflow</td>
<td>0.4593672E+18</td>
<td>0.2463405E+12</td>
<td>0.2818273E+12</td>
</tr>
<tr>
<td>0.90</td>
<td>0.3179971E-21</td>
<td>...</td>
<td>0.6562388E+17</td>
<td>-0.3041013E+11</td>
<td>0.1072543E+12</td>
</tr>
<tr>
<td>1.00</td>
<td>0.1928750E-21</td>
<td>...</td>
<td>0.3937433E+17</td>
<td>-0.3041013E+11</td>
<td>0.6538349E+11</td>
</tr>
</tbody>
</table>

(b) h = 0.01, k=10

<table>
<thead>
<tr>
<th>x</th>
<th>Exact Solution</th>
<th>Error (Trapezoidal)</th>
<th>Error (Gourlay)</th>
<th>Error (GH)</th>
<th>Error (Modified GH)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>0.7485183E-04</td>
<td>-0.4331087E-04</td>
<td>-0.4232418E-04</td>
<td>0.2885305E-04</td>
<td>0.2884071E-04</td>
</tr>
<tr>
<td>0.20</td>
<td>0.1522996E-07</td>
<td>-0.1172277E-07</td>
<td>-0.1151764E-07</td>
<td>0.1153792E-07</td>
<td>0.1153110E-07</td>
</tr>
<tr>
<td>0.30</td>
<td>0.8423464E-11</td>
<td>-0.7134286E-11</td>
<td>-0.7028668E-11</td>
<td>0.9060035E-11</td>
<td>0.9052854E-11</td>
</tr>
<tr>
<td>0.40</td>
<td>0.1266417E-13</td>
<td>-0.1117637E-13</td>
<td>-0.1102485E-13</td>
<td>0.1667959E-13</td>
<td>0.1666217E-13</td>
</tr>
<tr>
<td>0.50</td>
<td>0.5175558E-16</td>
<td>-0.4658230E-16</td>
<td>-0.4596976E-16</td>
<td>0.7657740E-16</td>
<td>0.7647300E-16</td>
</tr>
<tr>
<td>0.60</td>
<td>0.5749522E-18</td>
<td>-0.5225496E-18</td>
<td>-0.5155816E-18</td>
<td>0.9056664E-19</td>
<td>0.9040538E-19</td>
</tr>
<tr>
<td>0.70</td>
<td>0.1736205E-19</td>
<td>-0.1584953E-19</td>
<td>-0.1563463E-19</td>
<td>0.2817549E-19</td>
<td>0.2810993E-19</td>
</tr>
<tr>
<td>0.80</td>
<td>0.1425164E-20</td>
<td>-0.1303473E-20</td>
<td>-0.1285295E-20</td>
<td>0.2336870E-20</td>
<td>0.2320314E-19</td>
</tr>
<tr>
<td>0.90</td>
<td>0.3179971E-21</td>
<td>-0.2910312E-21</td>
<td>-0.2868857E-21</td>
<td>0.1845557E-20</td>
<td>0.3391030E-19</td>
</tr>
<tr>
<td>1.00</td>
<td>0.1928750E-21</td>
<td>-0.1765433E-21</td>
<td>-0.1740089E-21</td>
<td>0.1970679E-20</td>
<td>0.3403343E-19</td>
</tr>
</tbody>
</table>
6.4 SOLUTION OF THE GOURSAT PROBLEM IN HYPERBOLIC PARTIAL DIFFERENTIAL EQUATIONS BY THE GM AND THE MODIFIED GM FORMULAE

We will now consider the solution of the Goursat problem,

\[
\begin{align*}
    u_{xy} &= f(x,y,u,u_x,u_y), \\
    u(x,0) &= \sigma(x), \quad u(0,y) = \tau(y), \quad \sigma(0) = \tau(0) \\
    0 \leq x \leq a, & \quad 0 \leq y \leq b.
\end{align*}
\]  

(6.4.1)

by using the GM formula.

The AM formula for solving this problem has been discussed by several authors including Day [1966], Jain and Sharma [1968], Stetter and Torning [1963]. The Trapezoidal rule for solving (6.4.1) is given by

\[
u_{i+1,j+1} + u_{ij} - u_{i+1,j} = \frac{1}{4} h^2 [f_{i+1,j+1} + f_{ij} + f_{i,j+1} + f_{i+1,j}]
\]

(6.4.2)

where \( f_{ij} = f(x_i,y_j,u_{ij},u_{x ij},u_{y ij}) \), with difference replacements for \( (u_x)_{ij} \) and \( (u_y)_{ij} \) given by

\[
(u_x)_{ij} = (u_{i+1,j} - u_{ij})/h \text{ or } (u_{ij} - u_{i-1,j})/h
\]

and

\[
(u_y)_{ij} = (u_{i,j+1} - u_{ij})/h \text{ or } (u_{ij} - u_{i,j-1})/h.
\]

The RHS of (6.4.2) can be viewed as the arithmetic mean of the four function values at the four edges of the square region currently being considered given by \( x_i < x < x_{i+1}, \quad y_j < y < y_{j+1} \). The formula is implicit and is solved at each point by an iterative process which uses the starting values.
Note that this formula requires the storage of the function values at all the edges of the square other than at point \((x_{i+1}, y_{j+1})\) to avoid re-evaluation. Evaluations are started at the initial lines and subsequent evaluations are carried out while evaluating \(u_{i+1,j+1}\).

Gourlay [1970] also proposes the analogue of his modified Trapezoidal formula given by (6.3.5) in the preceding section for the solution of this problem. He does not mention any defect of formula (6.3.2) but rather stresses that his modified formula is as competitive, if not more accurate than the original one, and also with added simplicity. This formula is given by

\[
 u_{i+1,j+1} = \frac{1}{4} (u_{i+1,j+1}, j+1) + u_{i,j+1} \cdot (u_{i+1,j+1}, y)
\]

(6.4.3)

\[
 (u_x)_{i+1,j+1} = \frac{1}{h} \left( u_{i+1,j+1}, u_{i,j+1} \right)
\]

\[
 (u_y)_{i+1,j+1} = \frac{1}{h} \left( u_{i,j+1}, u_{i+1,j+1} \right).
\]

(6.4.4)

Since this formula is implicit, the iterative process with starting values given by (6.4.3) may be used to iterate with

\[
 u^{(m+1)}_{i+1,j+1} = \frac{1}{4} (u^{(m+1)}_{i+1,j+1}, j+1) + u_{i,j+1} \cdot (u^{(m+1)}_{i+1,j+1}, y)
\]

(6.4.5)

\[
 (u_x^{(m+1)})_{i+1,j+1} = \frac{1}{h} \left( u^{(m+1)}_{i+1,j+1}, u^{(m+1)}_{i,j+1} \right)
\]

\[
 (u_y^{(m+1)})_{i+1,j+1} = \frac{1}{h} \left( u^{(m+1)}_{i,j+1}, u^{(m+1)}_{i+1,j+1} \right).
\]

(6.4.6)

where

\[
 u^* = \frac{1}{4} \left( u_{i+1,j+1}, j+1, u_{i,j+1}, j+1 \right)
\]

(6.4.5-1)

\[
 p^* = \frac{1}{2h} \left( u_{i+1,j+1}, u_{i+1,j+1}, u_{i+1,j+1}, u_{i+1,j} \right)
\]

(6.4.5-11)

\[
 q^* = \frac{1}{2h} \left( u_{i+1,j+1}, u_{i+1,j+1}, u_{i+1,j}, u_{i+1,j+1} \right)
\]

(6.4.5-11)

(6.4.7)

and the truncation error is of order \(h^4\).

etc.
By using the GM technique we may modify formula (6.4.2) into the following form,

\[ u_{i+1,j+1} + u_{i,j} - u_{i+1,j} = \pm h^2 \left| f_{i+1,j+1} f_{i,j} f_{i,j+1} f_{i+1,j+1} \right|^{1/4} \]  \hspace{1cm} (6.4.8)

with the method of implementation as given earlier. The only problem is to determine the sign of the quantity on the RHS. If we take the sign to be 
\[ \text{sign} = \text{sign}(f_{i+1,j+1} + f_{i,j} + f_{i,j+1} + f_{i+1,j}) \],

then formula (6.4.8) is similar to formula (6.4.2) but the numerical value of the RHS. An error analysis will show whether this quantity is closer to the exact value than in formula (6.4.2).

The only drawback, as always with all the GM formula, is the extra time needed for the determination of the sign and in taking the roots. Following the stability argument for solving \( y' = \lambda(x)y \) using the GM formula, we may now claim that this formula would be as competitive as the Gourlay formula for solving problems with varying eigenvalues.

However, for the sake of simplicity in the computation, we may introduce further Gourlay's version of this formula which is given by

\[ u_{i+1,j+1} + u_{i,j} - u_{i+1,j} = h^2 f(x_{i+\frac{1}{2}}, y_{j+\frac{1}{2}}, u^{*}, p^{*}, q^{*}) \]  \hspace{1cm} (6.4.9)

where

\[ u^{*} = \left| u_{i+1,j+1} + u_{i,j} - u_{i+1,j} \right|^{1/4}, \]  \hspace{1cm} (6.4.10-i)

\[ p^{*} = \frac{1}{2h} \left( u_{i+1,j+1} + u_{i,j} - u_{i+1,j} \right), \]  \hspace{1cm} (6.4.10-ii)

\[ q^{*} = \frac{1}{2h} \left( u_{i+1,j+1} + u_{i,j} - u_{i+1,j} \right). \]  \hspace{1cm} (6.4.10-iii)

Alternatively, we may substitute equations (6.4.10-ii) and (6.4.10-iii) with
\[ p^* = \frac{1}{h} \sqrt{(u_{i+1,j+1} - u_{i,j+1})(u_{i+1,j} - u_{i,j})} \]

and

\[ q^* = \frac{1}{h} \sqrt{(u_{i+1,j+1} - u_{i+1,j})(u_{i,j+1} - u_{i,j})} \]

to obtain a formula which is completely based on the geometric mean.

**Numerical Example**

We will consider the three examples discussed by Day [1966] and Gourlay [1970] in which the error is taken to be the relative error, i.e.

\[ \text{error} = \frac{\text{true value} - \text{approximate value}}{\text{true value}}. \]

**Example 1**

\[ u_{xy} = e^{2u}, \]
\[ u(x,0) = x/2 - \log(1+e^x), \]
\[ u(0,y) = y/2 - \log(1+e^y). \]

The solution of this problem is

\[ u(x,y) = (x+y)/2 - \log(e^x+e^y). \]

By taking \( h=0.05 \), errors for \( u \) were obtained as shown in Table 6.7 where the four formulae as indicated have been used.

**Example 2**

\[ u_{xy} = u_x u_y/u \]
\[ u(x,1) = e^{(x+1)} \sin(1) \]
\[ u(1,y) = e^{(1+y)} \sin(y) \]

The solution of the problem is

\[ u(x,y) = e^{(x+y)} \sin y. \]

Taking \( h=0.05 \) the errors shown in Table 6.8 were obtained.
Example 3

\[ u_{xy} = \frac{u_x + u_y + u}{3} \]

\[ u(x,0) = e^x \]

\[ u(0,y) = e^y. \]

The solution of this problem is

\[ u(x,y) = e^{(x+y)}. \]

By taking \( h = 0.05 \) the errors shown in Table 6.9 were obtained.

Thus the GM formula gives satisfactory results in the three examples discussed here. In particular, for examples 1 and 3 the GM formula and our GM version of Gourlay's formula perform better than the corresponding AM and the Gourlay formula. For example 2, the performance of AM and GM formulae are comparable. The performance of our modified GM formulae are also very good especially with the geometric mean being used in all the averaging processes. But for this particular example Gourlay's method also performs at its best.

In addition to the advantage of being easy to program, Gourlay's version of the Trapezoidal method and the version of the GM formula presented here requires less computer space and a smaller amount of computational work so that the average computing time is reduced by a factor of 0.5 to 0.7 of the time needed by the usual formulae depending on the given problems.
Table 6.7
Example 1
stepsize, h=0.05
print results every 20 steps in both x and y-directions.

<table>
<thead>
<tr>
<th>AM</th>
<th>cpu time</th>
<th>x</th>
<th>1.0</th>
<th>2.0</th>
<th>3.0</th>
<th>4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5589822</td>
<td>1.0</td>
<td>-0.7288797E-04</td>
<td>-0.9707535E-04</td>
<td>-0.6498244E-04</td>
<td>-0.4038224E-04</td>
</tr>
<tr>
<td></td>
<td>5872723</td>
<td>2.0</td>
<td>-0.9707535E-04</td>
<td>-0.2946705E-03</td>
<td>-0.3412809E-03</td>
<td>-0.2589145E-03</td>
</tr>
<tr>
<td></td>
<td>6056869</td>
<td>3.0</td>
<td>-0.6498244E-04</td>
<td>-0.3412809E-03</td>
<td>-0.8042395E-03</td>
<td>-0.9513218E-03</td>
</tr>
<tr>
<td></td>
<td>5525608</td>
<td>4.0</td>
<td>-0.4038224E-04</td>
<td>-0.2589145E-03</td>
<td>-0.9513218E-03</td>
<td>-0.2114966E-02</td>
</tr>
</tbody>
</table>

Courlay's formula

<table>
<thead>
<tr>
<th>Courlay's formula</th>
<th>cpu time</th>
<th>x</th>
<th>1.0</th>
<th>2.0</th>
<th>3.0</th>
<th>4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2893686</td>
<td>1.0</td>
<td>-0.8159337E-04</td>
<td>-0.1390855E-03</td>
<td>-0.1329020E-03</td>
<td>-0.1080514E-03</td>
</tr>
<tr>
<td></td>
<td>2960503</td>
<td>2.0</td>
<td>-0.1390855E-03</td>
<td>-0.4149637E-03</td>
<td>-0.5682329E-03</td>
<td>-0.6104686E-03</td>
</tr>
<tr>
<td></td>
<td>2673663</td>
<td>3.0</td>
<td>-0.9707390E-04</td>
<td>-0.3412809E-03</td>
<td>-0.1361797E-02</td>
<td>-0.1752821E-02</td>
</tr>
<tr>
<td></td>
<td>2628100</td>
<td>4.0</td>
<td>-0.1080514E-03</td>
<td>-0.5104086E-03</td>
<td>-0.1752821E-02</td>
<td>-0.3947489E-02</td>
</tr>
</tbody>
</table>

GM

<table>
<thead>
<tr>
<th>GM</th>
<th>cpu time</th>
<th>x</th>
<th>1.0</th>
<th>2.0</th>
<th>3.0</th>
<th>4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5685541</td>
<td>1.0</td>
<td>-0.8159337E-04</td>
<td>-0.1390855E-03</td>
<td>-0.1329020E-03</td>
<td>-0.1080514E-03</td>
</tr>
<tr>
<td></td>
<td>5329904</td>
<td>2.0</td>
<td>-0.1390855E-03</td>
<td>-0.4149637E-03</td>
<td>-0.5682329E-03</td>
<td>-0.6104686E-03</td>
</tr>
<tr>
<td></td>
<td>5193777</td>
<td>3.0</td>
<td>-0.1329020E-03</td>
<td>-0.5682329E-03</td>
<td>-0.1361797E-02</td>
<td>-0.1752821E-02</td>
</tr>
<tr>
<td></td>
<td>5212143</td>
<td>4.0</td>
<td>-0.1080514E-03</td>
<td>-0.5104086E-03</td>
<td>-0.1752821E-02</td>
<td>-0.3947489E-02</td>
</tr>
</tbody>
</table>

Modified GM formula
(with AM in the finite difference)

<table>
<thead>
<tr>
<th>Modified GM formula</th>
<th>cpu time</th>
<th>x</th>
<th>1.0</th>
<th>2.0</th>
<th>3.0</th>
<th>4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3078299</td>
<td>1.0</td>
<td>-0.7569184E-04</td>
<td>-0.1137578E-03</td>
<td>-0.9708461E-04</td>
<td>-0.7535388E-04</td>
</tr>
<tr>
<td></td>
<td>2968988</td>
<td>2.0</td>
<td>-0.1137578E-03</td>
<td>-0.3419431E-03</td>
<td>-0.4396740E-03</td>
<td>-0.3777090E-03</td>
</tr>
<tr>
<td></td>
<td>2806066</td>
<td>3.0</td>
<td>-0.9708461E-04</td>
<td>-0.4396740E-03</td>
<td>-0.1048606E-02</td>
<td>-0.1316556E-02</td>
</tr>
<tr>
<td></td>
<td>2979586</td>
<td>4.0</td>
<td>-0.7535388E-04</td>
<td>-0.3777080E-03</td>
<td>-0.1316556E-02</td>
<td>-0.2957145E-02</td>
</tr>
</tbody>
</table>

Modified GM formula
(GM throughout)

<table>
<thead>
<tr>
<th>Modified GM formula</th>
<th>cpu time</th>
<th>x</th>
<th>1.0</th>
<th>2.0</th>
<th>3.0</th>
<th>4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3294676</td>
<td>1.0</td>
<td>-0.7569184E-04</td>
<td>-0.1137578E-03</td>
<td>-0.9707390E-04</td>
<td>-0.7535715E-04</td>
</tr>
<tr>
<td></td>
<td>3243977</td>
<td>2.0</td>
<td>-0.1137578E-03</td>
<td>-0.3419431E-03</td>
<td>-0.4396751E-03</td>
<td>-0.3776973E-03</td>
</tr>
<tr>
<td></td>
<td>3197455</td>
<td>3.0</td>
<td>-0.9707390E-04</td>
<td>-0.4396751E-03</td>
<td>-0.1048606E-02</td>
<td>-0.1316556E-02</td>
</tr>
<tr>
<td></td>
<td>2970767</td>
<td>4.0</td>
<td>-0.7535715E-04</td>
<td>-0.3776973E-03</td>
<td>-0.1316556E-02</td>
<td>-0.2957145E-02</td>
</tr>
</tbody>
</table>
Table 6.8
Example 2
stepsize, h=0.05
print results in every 4 steps in both x and y-directions.

AM

<table>
<thead>
<tr>
<th>cpu time</th>
<th>x</th>
<th>1.2</th>
<th>1.4</th>
<th>1.6</th>
<th>1.8</th>
<th>2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1834964</td>
<td>1.2</td>
<td>0.2757659E-01</td>
<td>0.5168154E-01</td>
<td>0.7293269E-01</td>
<td>0.9178728E-01</td>
<td>0.1086176E00</td>
</tr>
<tr>
<td>1733905</td>
<td>1.4</td>
<td>0.4724581E-01</td>
<td>0.8857117E-01</td>
<td>0.1245482E00</td>
<td>0.1562293E00</td>
<td>0.1843072E00</td>
</tr>
<tr>
<td>1704966</td>
<td>1.6</td>
<td>0.6189012E00</td>
<td>0.1152868E00</td>
<td>0.1617207E00</td>
<td>0.2023903E00</td>
<td>0.2382447E00</td>
</tr>
<tr>
<td>1750053</td>
<td>1.8</td>
<td>0.7229600E-01</td>
<td>0.1344530E00</td>
<td>0.1883254E00</td>
<td>0.2353384E00</td>
<td>0.2766370E00</td>
</tr>
<tr>
<td>1734877</td>
<td>2.0</td>
<td>0.7944317E-01</td>
<td>0.1476440E00</td>
<td>0.2066316E00</td>
<td>0.2580183E00</td>
<td>0.3030682E00</td>
</tr>
</tbody>
</table>

Couray's method

<table>
<thead>
<tr>
<th>cpu time</th>
<th>x</th>
<th>1.2</th>
<th>1.4</th>
<th>1.6</th>
<th>1.8</th>
<th>2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1295047</td>
<td>1.2</td>
<td>0.7333081E-01</td>
<td>0.9792227E-01</td>
<td>0.9620187E-01</td>
<td>0.1276293E-07</td>
<td>0.1722631E-07</td>
</tr>
<tr>
<td>1289666</td>
<td>1.4</td>
<td>0.7436767E-01</td>
<td>0.1490948E-01</td>
<td>0.1920293E-01</td>
<td>0.1869222E-07</td>
<td>0.2408820E-07</td>
</tr>
<tr>
<td>1395101</td>
<td>1.8</td>
<td>0.1659126E-01</td>
<td>0.1975484E-01</td>
<td>0.3466835E-01</td>
<td>0.3296052E-01</td>
<td>0.4194062E-07</td>
</tr>
<tr>
<td>1197773</td>
<td>2.0</td>
<td>0.1451502E-01</td>
<td>0.2400493E-01</td>
<td>0.3735923E-01</td>
<td>0.3581683E-01</td>
<td>0.4907081E-07</td>
</tr>
</tbody>
</table>

GM

<table>
<thead>
<tr>
<th>cpu time</th>
<th>x</th>
<th>1.2</th>
<th>1.4</th>
<th>1.6</th>
<th>1.8</th>
<th>2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1752147</td>
<td>1.2</td>
<td>0.2796781E-01</td>
<td>0.5519039E-01</td>
<td>0.8183513E-01</td>
<td>0.1086239E00</td>
<td>0.1317634E00</td>
</tr>
<tr>
<td>1661722</td>
<td>1.4</td>
<td>0.4864127E-01</td>
<td>0.9463283E-01</td>
<td>0.1388056E00</td>
<td>0.1817814E00</td>
<td>0.2201553E00</td>
</tr>
<tr>
<td>1624222</td>
<td>1.6</td>
<td>0.6413268E-01</td>
<td>0.1242405E00</td>
<td>0.1807841E00</td>
<td>0.2346784E00</td>
<td>0.2816219E00</td>
</tr>
<tr>
<td>1522694</td>
<td>1.8</td>
<td>0.7654914E-01</td>
<td>0.1741394E00</td>
<td>0.2133540E00</td>
<td>0.2750911E00</td>
<td>0.3281214E00</td>
</tr>
<tr>
<td>1507697</td>
<td>2.0</td>
<td>0.8961901E-01</td>
<td>0.1661689E00</td>
<td>0.2379680E00</td>
<td>0.3032189E00</td>
<td>0.3584001E00</td>
</tr>
</tbody>
</table>

Modified GM formula
(with An in the finite difference)

<table>
<thead>
<tr>
<th>cpu time</th>
<th>x</th>
<th>1.2</th>
<th>1.4</th>
<th>1.6</th>
<th>1.8</th>
<th>2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1315245</td>
<td>1.2</td>
<td>-0.6243630E-04</td>
<td>-0.1249126E-03</td>
<td>-0.1874264E-03</td>
<td>-0.2499720E-03</td>
<td>-0.3125514E-03</td>
</tr>
<tr>
<td>1375060</td>
<td>1.4</td>
<td>-0.1048241E-03</td>
<td>-0.2097061E-03</td>
<td>-0.3146493E-03</td>
<td>-0.4196545E-03</td>
<td>-0.5247136E-03</td>
</tr>
<tr>
<td>1306001</td>
<td>1.6</td>
<td>-0.1337606E-03</td>
<td>-0.2676321E-03</td>
<td>-0.4015520E-03</td>
<td>-0.5355517E-03</td>
<td>-0.6696215E-03</td>
</tr>
<tr>
<td>1273692</td>
<td>1.8</td>
<td>-0.1530167E-03</td>
<td>-0.3061291E-03</td>
<td>-0.4593160E-03</td>
<td>-0.6125912E-03</td>
<td>-0.7659395E-03</td>
</tr>
<tr>
<td>1087980</td>
<td>2.0</td>
<td>-0.1649089E-03</td>
<td>-0.3299110E-03</td>
<td>-0.4949973E-03</td>
<td>-0.6601643E-03</td>
<td>-0.8254298E-03</td>
</tr>
</tbody>
</table>

Modified GM
(GM throughout)

<table>
<thead>
<tr>
<th>cpu time</th>
<th>x</th>
<th>1.2</th>
<th>1.4</th>
<th>1.6</th>
<th>1.8</th>
<th>2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1547169</td>
<td>1.2</td>
<td>0.6917179E-08</td>
<td>0.8860423E-08</td>
<td>0.8222481E-08</td>
<td>0.1089932E-07</td>
<td>0.1489680E-07</td>
</tr>
<tr>
<td>1611844</td>
<td>1.4</td>
<td>0.6771706E-08</td>
<td>0.1361973E-07</td>
<td>0.1720775E-07</td>
<td>0.1603198E-07</td>
<td>0.2076290E-07</td>
</tr>
<tr>
<td>1691901</td>
<td>1.6</td>
<td>-0.2439091E-08</td>
<td>0.8015852E-08</td>
<td>0.1776402E-07</td>
<td>0.2079505E-07</td>
<td>0.2708680E-07</td>
</tr>
<tr>
<td>1361708</td>
<td>1.8</td>
<td>0.1518752E-07</td>
<td>0.1722640E-07</td>
<td>0.2737482E-07</td>
<td>0.3165717E-07</td>
<td>0.4067990E-07</td>
</tr>
<tr>
<td>1435433</td>
<td>2.0</td>
<td>0.1308783E-07</td>
<td>0.2117054E-07</td>
<td>0.3310765E-07</td>
<td>0.5264750E-07</td>
<td>0.6198485E-07</td>
</tr>
</tbody>
</table>
Example 3
stepsize, h=0.05
print results in every 20 steps in both x and y-directions.

Table 6.9

<table>
<thead>
<tr>
<th>AM</th>
<th>cpu time</th>
<th>x</th>
<th>y</th>
<th>1.0</th>
<th>2.0</th>
<th>3.0</th>
<th>4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8293763</td>
<td>1.0</td>
<td></td>
<td>0.9062287E-01</td>
<td>0.141065E+00</td>
<td>0.1689464E+00</td>
<td>0.1824632E+00</td>
</tr>
<tr>
<td></td>
<td>8515727</td>
<td>2.0</td>
<td></td>
<td>0.1410650E+00</td>
<td>0.2291956E+00</td>
<td>0.2833794E+00</td>
<td>0.3162438E+00</td>
</tr>
<tr>
<td></td>
<td>9389718</td>
<td>3.0</td>
<td></td>
<td>0.1689464E+00</td>
<td>0.2833794E+00</td>
<td>0.3592560E+00</td>
<td>0.4086788E+00</td>
</tr>
<tr>
<td></td>
<td>8506406</td>
<td>4.0</td>
<td></td>
<td>0.1842632E+00</td>
<td>0.3152438E+00</td>
<td>0.4086788E+00</td>
<td>0.4721758E+00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Courlay's formula</th>
<th>cpu time</th>
<th>x</th>
<th>y</th>
<th>1.0</th>
<th>2.0</th>
<th>3.0</th>
<th>4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5311999</td>
<td>1.0</td>
<td></td>
<td>-0.1616372E-03</td>
<td>-0.2617730E-03</td>
<td>-0.3232517E-03</td>
<td>-0.3607133E-03</td>
</tr>
<tr>
<td></td>
<td>5173223</td>
<td>2.0</td>
<td></td>
<td>-0.2617730E-03</td>
<td>-0.4474020E-03</td>
<td>-0.5762395E-03</td>
<td>-0.6640808E-03</td>
</tr>
<tr>
<td></td>
<td>5045828</td>
<td>3.0</td>
<td></td>
<td>-0.3232517E-03</td>
<td>-0.5762395E-03</td>
<td>-0.7685356E-03</td>
<td>-0.9107679E-03</td>
</tr>
<tr>
<td></td>
<td>5117670</td>
<td>4.0</td>
<td></td>
<td>-0.3607133E-03</td>
<td>-0.6640808E-03</td>
<td>-0.9107679E-03</td>
<td>-0.1105883E-02</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GM</th>
<th>cpu time</th>
<th>x</th>
<th>y</th>
<th>1.0</th>
<th>2.0</th>
<th>3.0</th>
<th>4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8104933</td>
<td>1.0</td>
<td></td>
<td>0.9901132E-01</td>
<td>0.1534548E+00</td>
<td>0.1831854E+00</td>
<td>0.1993219E+00</td>
</tr>
<tr>
<td></td>
<td>8290152</td>
<td>2.0</td>
<td></td>
<td>0.1534548E+00</td>
<td>0.2481023E+00</td>
<td>0.3055599E+00</td>
<td>0.3399858E+00</td>
</tr>
<tr>
<td></td>
<td>8291805</td>
<td>3.0</td>
<td></td>
<td>0.1831854E+00</td>
<td>0.3055599E+00</td>
<td>0.3856237E+00</td>
<td>0.4370937E+00</td>
</tr>
<tr>
<td></td>
<td>8993316</td>
<td>4.0</td>
<td></td>
<td>0.1993219E+00</td>
<td>0.3399858E+00</td>
<td>0.4370937E+00</td>
<td>0.5028901E+00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Modified GM (with AM in the finite difference)</th>
<th>cpu time</th>
<th>x</th>
<th>y</th>
<th>1.0</th>
<th>2.0</th>
<th>3.0</th>
<th>4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5390273</td>
<td>1.0</td>
<td></td>
<td>-0.400403E-04</td>
<td>-0.6542676E-04</td>
<td>-0.8078643E-04</td>
<td>-0.9015153E-04</td>
</tr>
<tr>
<td></td>
<td>5508038</td>
<td>2.0</td>
<td></td>
<td>-0.6542676E-04</td>
<td>-0.1118130E-03</td>
<td>-0.1440095E-03</td>
<td>-0.1659465E-03</td>
</tr>
<tr>
<td></td>
<td>5381091</td>
<td>3.0</td>
<td></td>
<td>-0.8078643E-04</td>
<td>-0.1440095E-03</td>
<td>-0.1900063E-03</td>
<td>-0.2275722E-03</td>
</tr>
<tr>
<td></td>
<td>5620968</td>
<td>4.0</td>
<td></td>
<td>-0.9015153E-04</td>
<td>-0.1659465E-03</td>
<td>-0.2275722E-03</td>
<td>-0.2763116E-03</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Modified GM (GM throughout)</th>
<th>cpu time</th>
<th>x</th>
<th>y</th>
<th>1.0</th>
<th>2.0</th>
<th>3.0</th>
<th>4.0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5264850</td>
<td>1.0</td>
<td></td>
<td>0.8075509E-04</td>
<td>0.1307758E-03</td>
<td>0.1614780E-03</td>
<td>0.1801658E-03</td>
</tr>
<tr>
<td></td>
<td>5072050</td>
<td>2.0</td>
<td></td>
<td>0.1307758E-03</td>
<td>0.2234830E-03</td>
<td>0.1614780E-03</td>
<td>0.1801658E-03</td>
</tr>
<tr>
<td></td>
<td>5068938</td>
<td>3.0</td>
<td></td>
<td>0.1614780E-03</td>
<td>0.2877914E-03</td>
<td>0.3837009E-03</td>
<td>0.4547653E-03</td>
</tr>
<tr>
<td></td>
<td>4897810</td>
<td>4.0</td>
<td></td>
<td>0.1801658E-03</td>
<td>0.3316439E-03</td>
<td>0.4547653E-03</td>
<td>0.5520982E-03</td>
</tr>
</tbody>
</table>
6.5 METHODS USING OTHER TYPES OF MEANS

6.5.1 Various Types of Means

In the development of methods for solving ordinary differential equations it is not clear that the arithmetic mean is always the best choice. Naturally the arithmetic mean formulae are the most convenient to use but are not necessarily the most accurate formulae to use for all types of problems. This has been shown in our experience of using the geometric mean in these formulae. There are, however, several other means which will also produce consistent approximations. In the following we list some of the means for the two positive quantities $x_i$ and $x_{i+1}$, i.e.

1. arithmetic mean

$$x_{i+1}^{AM} = \frac{x_i + x_{i+1}}{2} \quad (6.5.1)$$

2. geometric mean

$$x_{i+1}^{GM} = (x_i x_{i+1})^{\frac{1}{2}} \quad (6.5.2)$$

3. harmonic mean

$$x_{i+1}^{HM} = \frac{2x_i x_{i+1}}{x_i + x_{i+1}} \quad (6.5.3)$$

4. weighted geometric mean

$$x_{i+1}^{GM} = \left( \frac{w_1}{w_1 + w_2} x_i + \frac{w_2}{w_1 + w_2} x_{i+1} \right) \quad (6.5.4)$$

5. weighted harmonic mean

$$x_{i+1}^{HM} = \frac{w_1 x_i + w_2 x_{i+1}}{w_1 + w_2} \quad (6.5.5)$$

6. logarithmic mean

$$x_{i+1}^{LM} = \frac{x_{i+1} - x_i}{\ln \left( \frac{x_{i+1}}{x_i} \right)} \quad (6.5.6)$$
In Chapter 2 we have shown how the first three means are related to each other by the inequality

\[
\frac{\text{AM}}{x_{1+\frac{1}{2}}} \geq \frac{\text{GM}}{x_{1+\frac{1}{2}}} \geq \frac{\text{HM}}{x_{1+\frac{1}{2}}}
\]  

(6.5.7)

It is natural to ask whether there exists another mean which has a numerical value greater than the arithmetic mean. If it does, we could probably establish a formula which has a truncation error located away from the GM and presumably the HM formulae on the other side of the AM formula. We could then incorporate these formulae to form a more suitable general purpose method if the stability properties are good. Our investigation reveals that the following mean satisfy this inequality.

7. a new mean

\[
x_{1+\frac{1}{2}}^* = \frac{x_1^2 + x_{1+1}^2}{x_1 + x_{1+1}}
\]  

(6.5.8)

Note that this relation qualifies the requirement for a mean since \(x_1 \leq x_{1+\frac{1}{2}} \leq x_{1+1}\), and when \(x_1 = x_{1+1}\) we have \(x_1 = x_{1+\frac{1}{2}} = x_{1+1}\). The condition being \(x_1 > 0, x_{1+1} > 0\).

To show that \(x_{1+\frac{1}{2}}^*\) as given by equation (6.5.8) is numerically greater than \(\frac{\text{AM}}{x_{1+1}}\) it is sufficient to prove the following theorem.

**Theorem 6.1**

Any two numbers \(a\) and \(b\) where \(a \geq 0\) and \(b \geq 0\) will always satisfy the following inequality,

\[
\frac{a^2 + b^2}{a + b} \geq \frac{a + b}{2}.
\]
Proof

For any two real numbers \(a\) and \(b\) we have,

\[(a-b)^2 \geq 0\]

\[\Rightarrow a^2 + b^2 - 2ab \geq 0\]

\[\Rightarrow a^2 + b^2 \geq 2ab\]

\[\Rightarrow 2(a^2 + b^2) \geq (a^2 + b^2) + 2ab\]

\[\Rightarrow 2(a^2 + b^2) \geq (a+b)^2.\]

On dividing both sides by \(a+b \geq 0\), we have

\[\frac{2(a^2 + b^2)}{a+b} \geq a+b\]

or

\[\frac{a^2 + b^2}{a+b} \geq \frac{a+b}{2}\]

q.e.d.

Thus, the inequalities (6.5.7) may be extended into the following inequalities,

\[x_{i+\frac{1}{2}}^* \geq x_{i+\frac{1}{2}}^{AM} \geq x_{i+\frac{1}{2}}^{GM} \geq x_{i+\frac{1}{2}}^{HM}\]

(6.5.9)

In addition, we can easily deduce that \(x_{i+\frac{1}{2}}^{AM}\) is numerically located in between \(x_{i+\frac{1}{2}}^*\) and \(x_{i+\frac{1}{2}}^{HM}\) since

\[x_{i+\frac{1}{2}}^{AM} = \frac{x_{i+\frac{1}{2}}^* + x_{i+\frac{1}{2}}^{HM}}{2}\]

(6.5.10)

In the succeeding sections we shall analyse the accuracy and the stability property of the single step implicit integration formulae deduced from the Trapezoidal method by substituting the arithmetic mean in the increment function with the other types of means. More specifically we will consider the integration formula of the form

\[y_{n+1} = y_n + h \psi\]

(6.5.11)

where \(\psi\) is the increment function computed by using \(f_{n+\frac{1}{2}}^M\) where \(M\) can be either any of the means indicated \(AM, GM, HM,\) etc.
6.5.2 Harmonic Mean (HM) Formula

The harmonic mean formula is obtained by replacing $\psi$ in (6.5.11) with the related harmonic mean (6.5.3) for the function $f$. The formula is given by

$$y_{n+1} = y_n + h \left( \frac{2f_n}{f_n + f_{n+1}} \right)$$  \hfill (6.5.12)

Truncation Error

We will establish the truncation error by expanding the RHS of (6.5.12) as a Taylor series expansion about $x_n$. By putting

$$f_{n+1} = y'_{n+1}$$

$$= y' + h y'' + \frac{1}{2} h^2 y''' + \frac{1}{6} h^3 y^{(iv)} + ...$$  \hfill (6.5.13)

in the formula, we have

$$y_{n+1} = y_n + h \left( \frac{2y'_n y'_{n+1}}{y'_n + y'_{n+1}} \right)$$

$$= y_n + h \left[ \frac{2y'^2_n + 2hy'_ny''_n + h^2 y'^2_2}_n + ...}{2y'_n + hy''_n + \frac{1}{2} h^2 y'''_n + ...} \right]$$  \hfill (6.5.14)

The quotient in the square brackets can be performed and simplified into

$$y'_n + \frac{h}{2} y''_n + \frac{h^2}{4} \left( \frac{y''_n y'_n - y''^2_n}{y'_n} \right) + ...$$

by direct division. Substituting this result in (6.5.14) we obtain

$$y_{n+1} = y_n + h \left[ y'_n \right. + \frac{h}{2} y''_n + \frac{h^2}{4} \left( \frac{y''_n y'_n - y''^2_n}{y'_n} \right) + ... \left. \right]$$

$$= y_n + hy'_n + \frac{h^2}{2} y''_n + \frac{h^3}{4} \left( y''_n - \frac{y''^2_n}{y'_n} \right) + ...$$  \hfill (6.5.15)
The Taylor series expansion of $y(x_{n+1})$ about $x_n$ is given by

$$y(x_{n+1}) = y_n + hy_n^\prime + \frac{1}{2} h^2 y_n^\prime\prime + \frac{1}{6} h^3 y_n^\prime\prime\prime + \ldots$$  \hspace{1cm} (6.5.16)

Subtracting (6.5.15) from (6.5.16) we obtain

$$y(x_{n+1}) - y_{n+1} = h^3 \left( \frac{y_n^\prime\prime\prime}{6} - \frac{y_n^\prime\prime}{4} + \frac{y_n^\prime\prime\prime}{4y_n^\prime} \right) + O(h^4)$$

$$= h^3 \left( - \frac{y_n^\prime\prime}{12} + \frac{y_n^\prime\prime\prime}{4y_n^\prime} \right) + O(h^4).$$

Therefore method (6.5.12) has an accuracy of order 2 and the principal term of the local truncation error is given by

$$LTE = \left( - \frac{y_n^\prime\prime}{12} + \frac{y_n^\prime\prime\prime}{4y_n^\prime} \right) h^3.$$ \hspace{1cm} (6.5.17)

A preliminary test with formula (6.5.12) reveals that a problem of division by zero arises in evaluating the mean

$$\frac{2f_n f_{n+1}}{f_n + f_{n+1}}$$

when $f$ becomes zero or very near to zero. A special treatment to resolve this difficulty is considered here.

**Applicability of the HM formula**

Our experiment shows that the HM formula performs reasonably well as long as the denominator of the increment function $\Psi$ does not become zero or very near to zero as the integration proceeds, and as long as the function values do not change sign. In particular the method is easily applied if $f > 0$, which is the original requirement for the harmonic mean. However we may also conveniently use the formula for general values of $f$ if we make some modification to the formula.
In the case when $f_n + f_{n+1} = 0$ (or $\approx 0$) the computation will involve a division by zero. There are two possibilities when this can occur, and so we must seek ways to avoid them.

(i) when both $f_n = 0$ and $f_{n+1} = 0$. In this case $\psi = 0$ since

\[
\psi = \lim_{f_n \to 0, f_{n+1} \to 0} \frac{2f_n f_{n+1}}{f_n + f_{n+1}} = \lim_{f_n \to 0} 2f_n = 0,
\]

(by applying L'Hopital rule on $f_{n+1}$).

(ii) when $f_n = -f_{n+1}$. Mathematically

\[
\lim_{f_n + f_{n+1} \to 0} \psi = \infty.
\]

Geometrically we can say that the solution curve has a turning point within $[x_n, x_{n+1}]$. Therefore, we can take $\psi = 0$, i.e. $y_{n+1} = y_n$ (assuming the curve is symmetrical about the turning point).

The method is now temporarily given by

\[ y_{n+1} = y_n + h\psi, \]

where

\[
\psi = \begin{cases} 
0, & \text{if } f_n + f_{n+1} = 0 \\
\frac{2f_n f_{n+1}}{f_n + f_{n+1}}, & \text{otherwise.}
\end{cases}
\]
Supposing that \( f \) changes sign within \([x_n, x_{n+1}]\) and that \( f_n + f_{n+1} \neq 0 \). In this case the HM of \( f_n \) and \( f_{n+1} \) is not located in between \( f_n \) and \( f_{n+1} \) unless we change the sign of the HM. As an example if \( a=10, \ b=-1 \), then the harmonic mean of \( a \) and \( b \) is given by

\[
HM = \frac{2(10)(-1)}{10-1} = \frac{-20}{9} = -2.22
\]

On multiplying by \(-1\), we have \( HM=2.22 \) which is now located inside \([-1,10]\). Similarly, if \( a=-10, \ b=1 \), we have

\[
HM = \frac{2(-10)(1)}{-10+1} = \frac{-20}{-9} = 2.22
\]

and changing the sign produces \( HM=-2.22 \) which is seen to be more realistic than the positive value. The more general formula would now be in the form,

\[
\psi = \begin{cases} 
0 & \text{if } f_n + f_{n+1} = 0 \\
\frac{2f_n f_{n+1}}{f_n + f_{n+1}} & \text{if } \text{sgn}(f_n) \neq \text{sgn}(f_{n+1}) \\
\frac{2f_n f_{n+1}}{f_n + f_{n+1}} & \text{otherwise.}
\end{cases}
\] (6.5.19)

Stability of HM formula

Applying the method to the test equation \( y' = \lambda y \) produces

\[
y_{n+1} = y_n + 2h \frac{\lambda y_n + \lambda y_{n+1}}{\lambda y_n + \lambda y_{n+1}}
\]

\[
y_{n+1} = y_n + 2h \lambda \frac{y_n y_{n+1}}{y_n + y_{n+1}}
\]

Multiplying both sides with \( (y_n y_{n+1}) \) we obtain

\[
y_n y_{n+1} + y_{n+1}^2 = y_n^2 + y_n y_{n+1} + 2h\lambda y_n y_{n+1}.
\]

By cancelling the \( y_n y_{n+1} \) terms out and dividing through by \( y_n^2 \) we obtain
\[
\left( \frac{Y_{n+1}}{Y_n} \right)^2 = 1 + 2h \lambda \left( \frac{Y_{n+1}}{Y_n} \right).
\]

Writing \( \frac{Y_{n+1}}{Y_n} = Q \) and \( h \lambda = z \) we have

\[
Q^2 - 2Qz - 1 = 0
\]

\[
Q = \frac{2z \pm \sqrt{4z^2 + 4}}{2} = z \pm \sqrt{z^2 + 1}
\]

The stability region is determined by the values of \( z \) such that

\[
\left| \frac{Y_{n+1}}{Y_n} \right| < 1, \text{ i.e.}
\]

\[
|z \pm \sqrt{z^2 + 1}| < 1 \quad (6.5.20)
\]

The surface whose height is defined by \( |z \pm \sqrt{z^2 + 1}| \) is drawn by using the GINOSURF routine FUNGRD as can be seen in Figure 6.4. This surface shows that inequality (6.5.20) is achieved for all \( z \) with \( \text{Re}(z) < 0 \). The corresponding region of absolute stability is shown in Figure 6.5. It is seen that this region contains the whole left-hand side of the complex plane, and hence the method is A-stable.

### 6.5.3 A second order one-step method based on a new mean

By substituting the mean (6.5.8) into the increment function, we obtain a new formula given by the recurrence relation,

\[
y_{n+1} = y_n + h \left( \frac{f_n^2 + f_{n+1}^2}{f_n + f_{n+1}} \right) \quad (6.5.21)
\]
Figure 6.4: The surface defined by $|z + \sqrt{z^2 + 1}|$
Figure 6.5: Stability region for the HM formula.
The Truncation Error

From (6.5.13) we have

$$y_{n+1} = y_n' + hy_n'' + \frac{h^2}{2} y_n''' + \ldots$$

or

$$(y_{n+1}')^2 = y_n^2 + 2hy_n' y_n'' + h^2 (y_n'')^2 + y_n'y_n''' + \ldots$$

Therefore, the increment function in (6.5.21) can be expressed as

$$\frac{f^2_n + f^2_{n+1}}{f_n + f_{n+1}} = \frac{2y_n^2 + 2hy_n' y_n'' + h^2 (y_n'')^2 + y_n'y_n'''} + \ldots$$

By direct division the quotient on the RHS of this equation can be simplified into the form,

$$y' + \frac{h}{2y_n'} \left[ \frac{1}{2} (y_n'')^2 + \frac{h^2}{2} y_n''' + \ldots \right]$$

Thus (6.5.21) becomes

$$y_{n+1} = y_n + h[y_n' + \frac{h}{2} y_n'' + \frac{h^2}{2y_n'} (\frac{1}{2} (y_n'')^2 + \frac{h^2}{2} y_n''' + \ldots)]$$

$$= y_n + hy_n' + \frac{h^2}{2} y_n'' + h^3 \left( \frac{y_n'''}{4} + \frac{(y_n'')^2}{4y_n'} \right) + \ldots \quad (6.5.22)$$

But

$$y(x_{n+1}) = y_n + hy_n' + \frac{h^2}{2} y_n'' + \frac{h^3}{6} y_n''' + \ldots$$

Thus, the local truncation error is given by

$$LTE = y(x_{n+1}) - y_{n+1} = h^3 \left( - \frac{y_n''}{12} - \frac{(y_n'')^2}{4y_n'} \right) \quad (6.5.23)$$
Note that the local truncation error (6.5.23) is deviated by a similar amount as the local truncation error (6.5.17) from \(-\frac{1}{12} h^3 y_n''\), i.e. the local truncation error of the trapezoidal method, but in the opposite direction. Thus, theoretically the average of the results obtained by using formula (6.5.12) and formula (6.5.21) is similar to the one obtained by using the trapezoidal rule.

In addition, this also implies that, for some problems, when the GM and the HM formulae do not improve the accuracy of the trapezoidal formula, the method given by (6.5.21) may prove to be more accurate by bringing the error down at the other direction.

**Stability Analysis**

By applying the model equation

\[ y' = \lambda y, \; y(0) = 1 \]

to equation (6.5.21) we obtain the following recurrence,

\[ y_{n+1} = y_n + h \left( \frac{\lambda^2 y_n^2 + \lambda^2 y_{n+1}^2}{\lambda y_n + \lambda y_{n+1}} \right) \]

or,

\[ y_{n+1} = y_n + h \lambda \left( \frac{y_n^2 + y_{n+1}^2}{y_n + y_{n+1}} \right) \]

On multiplying both sides by \(y_n + y_{n+1}\) we obtain

\[ y_{n+1}(y_n + y_{n+1}) = y_n(y_n + y_{n+1}) + h\lambda(y_n^2 + y_{n+1}^2) \]

or,

\[ y_{n+1}^2 = y_n^2 + h\lambda(y_n^2 + y_{n+1}^2) \]
Dividing throughout by $y_n^2$,

$$\left(\frac{y_{n+1}}{y_n}\right)^2 = 1 + h\lambda \left[ 1 + \left(\frac{y_{n+1}}{y_n}\right)^2 \right]$$

$$(1-h\lambda)\left(\frac{y_{n+1}}{y_n}\right)^2 = 1 + h\lambda$$

$$\frac{y_{n+1}}{y_n} = \sqrt{\frac{1+h\lambda}{1-h\lambda}}$$

(6.5.24)

The stability region of the formula is determined by the values of $h\lambda$ in the complex plane such that $y_{n+1}/y_n$ as given by (6.5.24) satisfies the inequality,

$$\left| \frac{y_{n+1}}{y_n} \right| < 1$$

i.e. $$\left| \frac{1+h\lambda}{1-h\lambda} \right| < 1$$

or, $$\left| \frac{1+h\lambda}{1-h\lambda} \right| < 1$$

(6.5.25)

The surface defined by $|(1+z)/(1-z)|$ where $z=h\lambda$ is shown in Figure 6.6. The region for which inequality (6.5.25) is satisfied is shown in Figure 6.7 which coincides with the left half plane. Thus, method (6.5.21) is A-stable.
Figure 6.6: The surface defined by $\left| \frac{1+z}{1-z} \right|$.
Figure 6.7: Stability region for method (6.5.21)
6.5.4 A method derived by using the Logarithmic Mean

In conjunction with the logarithmic mean given by (6.5.6) we may now form the corresponding integration formula given by,

\[ y_{n+1} = y_n + h \left[ \frac{(y_{n+1} - y_n')}{\ln \left( \frac{y_{n+1}}{y_n'} \right)} \right] \quad (6.5.26) \]

The Truncation error

As before, we have

\[ y_{n+1}' = y_n' + hy_n'' + \frac{h^2}{2} y_n''' + \frac{h^3}{6} (y_n'^{(iv)}) + ... \]

so that

\[ \ln \left( \frac{y_{n+1}'}{y_n'} \right) = \ln \left[ 1 + \left( \frac{hy_n''}{y_n'} + \frac{h^2 y_n'''}{2y_n'} + \frac{h^3 (y_n'^{(iv)})}{6y_n'} + ... \right) \right] \]

By using the Taylor series expansion of \( \ln(1+x) \), i.e.

\[ \ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + ... \quad -1 \leq x \leq 1 \]

we have,

\[
\ln \left( \frac{y_{n+1}'}{y_n'} \right) = \left[ \frac{hy_n''}{y_n'} + \frac{h^2 y_n'''}{2y_n'} + \frac{h^3 (y_n'^{(iv)})}{6y_n'} + ... \right]
- \frac{1}{2} \left[ \frac{h^2 (y_n'')^2}{(y_n')^2} + \frac{h^3 y_n'' y_n'''}{(y_n')^2} \right] + \frac{1}{3} \left[ \frac{h^3 (y_n'')^3}{(y_n')^3} + ... \right]
= h \frac{y''}{y_n'} + \frac{h^2}{2} \left[ \frac{y_n'''}{y_n'} - \frac{(y_n'')^2}{(y_n')^2} \right]
+ h^3 \left[ \frac{1}{6} \frac{y_n'^{(iv)}}{y_n'} - \frac{1}{4} \frac{y_n y_n'''}{(y_n')^2} + \frac{1}{3} \frac{(y_n'')^3}{(y_n')^2} \right]
+ ... \quad (6.5.27)

Therefore,

\[
\psi = \frac{(y_{n+1}' - y_n')}{\ln \left( \frac{y_{n+1}'}{y_n'} \right)}
\]

\[
\psi = \frac{h^{2}}{2} y_n'' + \frac{h^{3}}{6} y_n^{(iv)} + \ldots
\]

\[
\frac{h y_n''}{y_n'} + \frac{h^{2}}{6} \left[ \frac{y_n'' - \left( y_n' \right)^{2}}{y_n'} \right] + h^{3} \left[ \frac{y_n^{(iv)} - y_n y_n'''}{6y_n'} - \frac{y_n'''}{2(y_n')^2} + \frac{(y_n'')^3}{3(y_n')^3} \right] + \ldots
\]

\[
y_n' + \frac{h}{2} y_n'' + h^{2} \left[ \frac{1}{4} y_n'' - \frac{1}{12} \frac{(y_n'')^2}{y_n'} \right] + \ldots
\]

and thus (6.5.26) becomes,

\[
y_{n+1} = y_n + hy_n' + \frac{h^2}{2} y_n'' + h^3 \left[ \frac{1}{4} y_n'' - \frac{1}{12} \frac{(y_n'')^2}{y_n'} \right] + \ldots
\]

(6.5.28)

Comparing (6.5.28) with the Taylor series expansion for \( y(x_{n+1}) \), i.e.

\[
y(x_{n+1}) = y_n + hy_n' + \frac{h^2}{2} y_n'' + \frac{h^3}{6} y_n''' + \ldots
\]

we see that the method is of 2nd order with truncation error given by

\[
LTE = y(x_{n+1}) - y_{n+1} = h^3 \left[ - \frac{1}{12} y_n''' + \frac{1}{12} \frac{(y_n'')^2}{y_n'} \right]
\]

Let us now summarise the results of our analysis by rewriting the local truncation error of each method we have discussed.

**Trapezoidal:**

\[
LTE = - \frac{1}{12} h^3 y_n''
\]

**Geometric Mean:**

\[
LTE = h^3 \left[ - \frac{1}{12} y_n''' + \frac{1}{8} \frac{(y_n'')^2}{y_n'} \right]
\]

**Harmonic Mean:**

\[
LTE = h^3 \left[ - \frac{1}{12} y_n''' + \frac{1}{4} \frac{(y_n'')^2}{y_n'} \right]
\]

**Method (6.5.21):**

\[
LTE = h^3 \left[ - \frac{1}{12} y_n''' - \frac{1}{4} \frac{(y_n'')^2}{y_n'} \right]
\]

**Logarithmic Mean:**

\[
LTE = h^3 \left[ - \frac{1}{12} y_n''' + \frac{1}{12} \frac{(y_n'')^2}{y_n'} \right]
\]
As we have suggested earlier, it is not clear that any of these methods are better than the other in terms of accuracy. Apparently the degree of their accuracy will depend on how small their error terms are and incidentally this will depend also on the problem being solved.

Numerical Example

We solve the initial value problem

\[ y' = -y, \quad y(0) = 1 \]

by using all the five methods we have studied in this section.

For this particular problem we have

\[ y_n' = -y_n \]
\[ y_n'' = y_n \]
\[ y_n''' = -y_n' \]

This causes the local truncation error in the logarithmic mean formula to vanish but is nonzero in the other means. Therefore, we expect the logarithmic mean to produce the most accurate results for this particular problem, i.e. equivalent to a method of order at least 3. This is in fact confirmed by our results shown in Table 6.9 where we have used the stepsize \( h = 0.1 \) for all the five methods.
Table 6.9: Solution of \( y' = -y \), \( y(0) = 1 \).

\[ h = 0.1 \]

<table>
<thead>
<tr>
<th>( x )</th>
<th>Exact Solution</th>
<th>Error (Trapezoidal)</th>
<th>Error (GI)</th>
<th>Error (IM)</th>
<th>Error (6.5.21)</th>
<th>Error (log)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.9048374E+00</td>
<td>-0.7550586E-04</td>
<td>0.3766340E-04</td>
<td>0.1501475E-03</td>
<td>-0.3034222E-03</td>
<td>0.3413405E-08</td>
</tr>
<tr>
<td>0.2</td>
<td>0.8187308E+00</td>
<td>-0.136523E-03</td>
<td>0.6815019E-04</td>
<td>0.2717309E-03</td>
<td>-0.5489023E-03</td>
<td>-0.3562900E-08</td>
</tr>
<tr>
<td>0.3</td>
<td>0.7408182E+00</td>
<td>-0.1854563E-03</td>
<td>0.9250731E-04</td>
<td>0.3684725E-03</td>
<td>-0.7449265E-03</td>
<td>-0.3284374E-08</td>
</tr>
<tr>
<td>0.4</td>
<td>0.6703200E+00</td>
<td>-0.2237413E-03</td>
<td>0.1116009E-03</td>
<td>0.4450257E-03</td>
<td>-0.8986554E-03</td>
<td>-0.2893355E-08</td>
</tr>
<tr>
<td>0.5</td>
<td>0.6065307E+00</td>
<td>-0.2530472E-03</td>
<td>0.1262326E-03</td>
<td>0.5033908E-03</td>
<td>-0.1016442E-02</td>
<td>0.8110662E-09</td>
</tr>
<tr>
<td>0.6</td>
<td>0.5488116E+00</td>
<td>-0.2747501E-03</td>
<td>0.1370651E-03</td>
<td>0.5466279E-03</td>
<td>-0.1103781E-02</td>
<td>-0.7720132E-09</td>
</tr>
<tr>
<td>0.7</td>
<td>0.4965853E+00</td>
<td>-0.2900236E-03</td>
<td>0.1446975E-03</td>
<td>0.5770947E-03</td>
<td>-0.1164907E-02</td>
<td>0.1735577E-08</td>
</tr>
<tr>
<td>0.8</td>
<td>0.4493290E+00</td>
<td>-0.2990126E-03</td>
<td>0.1496344E-03</td>
<td>0.5968227E-03</td>
<td>-0.1204599E-02</td>
<td>0.1402758E-08</td>
</tr>
<tr>
<td>0.9</td>
<td>0.4065697E+00</td>
<td>-0.3052707E-03</td>
<td>0.1523209E-03</td>
<td>0.6075799E-03</td>
<td>-0.1225779E-02</td>
<td>0.2966792E-10</td>
</tr>
<tr>
<td>1.0</td>
<td>0.3678794E+00</td>
<td>-0.3069009E-03</td>
<td>0.1531411E-03</td>
<td>0.6108946E-03</td>
<td>-0.1232774E-02</td>
<td>-0.1700181E-08</td>
</tr>
</tbody>
</table>
CHAPTER SEVEN

OSCILLATORY AND PERIODIC PROBLEMS
7.1 THE NUMERICAL SOLUTION OF OSCILLATORY PROBLEMS

7.1.1 Gautschi's Method

An interesting and important class of initial value problems which can arise in practice consists of differential equations whose solutions are known to be periodic, or to oscillate with a known frequency. Examples of such problems can be found in the field of ecology, medical sciences and oscillatory motion in a nonlinear force field.

Gautschi [1961] developed a class of methods based on trigonometrical polynomials which is appropriate for use on such problems if the frequency, or some suitable substitute, can be estimated in advance. His methods integrate exactly appropriate trigonometric polynomials of given order, just as classical methods integrate exactly algebraic polynomials of given degree.

We recall that the order p of a linear multistep method may be defined by the requirement

\[
\begin{align*}
L[x^r;h] &= 0, \quad r=0,1,\ldots,p; \\
L[x^{p+1};h] &\neq 0,
\end{align*}
\]

(7.1.1)

where \( L \) is the associated linear difference operator defined by Eq. (3.3.6) of Chapter 3. In other words, the linear difference operator \( L \), of order \( p \), annihilates all algebraic polynomials of order \( \leq p \). The methods developed by Gautschi similarly annihilate trigonometric polynomials up to a certain degree. Let the known or estimated period of the oscillation be \( T \), and define the frequency \( \omega = 2\pi/T \). Then the method

\[
\sum_{j=0}^{k} \alpha_j y_{n+j} = h \sum_{j=0}^{k} \beta_j(v)f_{n+j},
\]

(7.1.2)

\[ v = wh, \quad \alpha_k = +1. \]
is said to be of \textit{trigonometric order q relative to the frequency w} if the associated difference operator

\[ L_w[y(x);h] = \sum_{j=0}^{k} [\alpha_j y(x+jh) - h^2 \beta_j(v)y'(x+jh)] \]  

(7.1.3)

satisfies

\[ L_w[1;h] = 0 \]

\[ L_w[\cos(rwx);h] = L_w[\sin(rwx);h] = 0 \]

\[ r = 1, 2, \ldots, q, \]  

(7.1.4)

\[ L_w[\cos((q+1)wx);h] \text{ and } L_w[\sin((q+1)wx);h] \]

not both identically zero.

Let us now consider an implicit one-step method of class (7.1.2), i.e. with \( k = 1 \). If we require it to be of trigonometric order 1, then the three unspecified coefficients \( \alpha_0, \beta_1(v) \) and \( \beta_0(v) \) must satisfy the conditions

\[ L_w[1;h] = 1 + \alpha_0 = 0 \]

\[ L_w[\cos(wx);h] = \cos[w(x+h)] + \alpha_0 \cos(wx) \]

\[ + hw \{ \beta_1 \sin[w(x+h)] + \beta_0 \sin(wx) \} = 0 \]

\[ L_w[\sin(wx);h] = \sin[w(x+h)] + \alpha_0 \sin(wx) \]

\[ - hw \{ \beta_1 \cos[w(x+h)] + \beta_0 \cos(wx) \} = 0. \]

Substituting \( \alpha_0 = -1 \) into the second and third equations we obtain the following system of algebraic equations,

\[
\begin{pmatrix}
\sin(wx) & \sin[w(x+h)] \\
-v\cos(wx) & -v\cos[w(x+h)]
\end{pmatrix}
\begin{pmatrix}
\beta_0 \\
\beta_1
\end{pmatrix}
= \begin{pmatrix}
-cos[w(x+h)]+cos(wx) \\
-sin[w(x+h)]+sin(wx)
\end{pmatrix}
\]
The solution of this set of equations is

\[ a_0 = -1, \quad \beta_0 = \beta_1 = \frac{\tan(v/2)}{v}, \]

where \( v = wh \), yielding the trigonometric equivalent of the trapezoidal rule

\[ y_{n+1} = y_n + \frac{htan(v/2)}{v} (f_n + f_{n+1}) \quad (7.1.5) \]

Now, \( \tan(v/2) \) can be expanded in power series as

\[
\tan(v/2) = \frac{v}{2} + \frac{(v/2)^3}{3} + \frac{2(v/2)^5}{15} + \frac{17(v/2)^7}{315} + \ldots
\]

\[
= \frac{v}{2} + \frac{v^3}{24} + \frac{v^5}{240} + \frac{17v^7}{40320} + \ldots
\]

Substituting into (7.1.5) we have,

\[
y_{n+1} = y_n + h\left[ \frac{v^2}{24} + \frac{v^4}{240} + \ldots \right] (f_n + f_{n+1})
\]

Expanding the RHS about \( y_n \) we obtain

\[
y_{n+1} = y_n + h \left[ \frac{v^2}{24} + \frac{v^4}{240} + \ldots \right] (y'_n + y''_n)
\]

\[
+ hy''_n + \frac{h^2}{2} y'''_n + \frac{h^3}{6} y(iv)_n + \ldots
\]

\[ = y_n + h y'_n + \frac{h^2 y''_n}{2} + h^3 \left( \frac{y'''_n}{4} + \frac{w y''_n}{12} \right) + \ldots
\]

This expansion is of order 2 and the local truncation error is given by
\[ \text{LTE} = y(x_{n+1}) - y_{n+1} \]

\[ = h^3 \left[ \frac{y''_{n}}{6} - \frac{y'''}{4} - \frac{w^2 y'_{n}}{12} \right] + o(h^4) \]

\[ = - \frac{h^3}{12} (y''_{n} + w^2 y'_{n}) + o(h^2). \]  

(7.1.6)

Thus, the order of (7.1.5) in a sense analogous to that of a linear multistep method, is two. This is referred to by Gautschi as the algebraic order.

On the trigonometric order attainable by a method of class (7.1.2) of given stepnumber, Gautschi [1961] proves the following result (see, for example, Lambert [1973], p.207). Let \( v = wh \) be given, and let \( k \) be a given even (or odd) integer. Then, for any given set of coefficients \( \alpha_j, j=0,1,\ldots,k \), subject to \( \sum_{j=0}^{k} \alpha_j = 0 \), there exists a unique explicit (or implicit) method of class (7.1.2) whose trigonometric order is \( q=\frac{j}{2}k \) (or \( q=\frac{j}{2}(k+1) \)). In both cases, when \( w = 0 \) the method will reduce to a linear multistep method of algebraic order \( 2q \). Coefficients of such methods, when the left-hand side is specified to be of Adams type are derived by Gautschi for \( k=1,2,\ldots,6 \).

More recently Neta [1985] derived a special variation of the Gautschi's formula in the form,

\[ \sum_{j=0}^{k} \alpha_j y_{n+j} = h^8 \_k (v)f_{n+k} \]

(7.1.7)

\( v=wh, \alpha_k = +1 \).

(7.1.7) is referred to by Neta as families of backward differentiation methods based on trigonometric polynomial. One advantage of these formulae mentioned by Neta is that it does not suffer from the sensitivity to changes in the frequency \( w \) as the other Gautschi formula do. He derived the formulae for \( k=1,2,3,4 \).
7.1.2 Geometric Mean Formula

It looks straightforward that formula (7.1.5) may be modified for use in the geometric sense if we substitute the arithmetic mean by the corresponding geometric mean, i.e.,

\[ y_{n+1} = y_n + \frac{h \tan(v/2)}{v} \cdot 2 \sqrt{f_{n+1}} \quad (7.1.8) \]

Unfortunately this formula does not have a trigonometric order 1 possessed by formula (7.1.5), since by substituting \( y = \cos(wx) \) and bringing all the terms into the LHS we obtain

\[
\cos(x+h) - \cos(wx) - h \frac{\tan(wh/2)}{wh} \cdot 2 [(-\sin wx) (-\sin w(x+h))]^{\frac{1}{2}},
\]

which does not vanish identically.

However, as we shall now show, its algebraic order is 2. By rewriting (7.1.8) in a series form we obtain

\[
y_{n+1} = y_n + 2h \left[ \frac{1}{4} + \frac{y^2}{24} + \frac{y^4}{240} + \ldots \right] \left[ h y_n' + \frac{h^2}{2} y_n'' \right. \\
\left. + h^3 \left( \frac{y_n'''}{4} - \frac{(y_n'')^2}{8y_n''} \right) + \ldots \right]
\]

\[ = y_n + h y_n' + \frac{h^2}{2} y_n'' + h^3 \left[ \frac{y_n'''}{12} + \frac{y_n'}{12} + \frac{y_n''}{8y_n''} \right] + \ldots \]

The local truncation error of (7.1.8) is thus given by

\[
\text{LTE} = h^3 \left[ - \frac{y_n''}{12} - \frac{w^2 y_n'}{12} + \frac{y_n'^2}{8y_n'} \right] \quad (7.1.9)
\]

which confirms the 2nd order accuracy of the formula.
It is clear that the existence of the square root signs or non-integer powers in the GM formula makes it impossible to match the formula with the trigonometric sine and cosine functions to obtain a suitable trigonometric order. We shall, therefore, conclude that there does not exist GM formula equivalent to formula (7.1.2) which has trigonometric order $\geq 1$. However, although it may be possible for such a formula to have an algebraic order as high as we may require, their suitability for application to problems with oscillatory solutions is no more superior than the ordinary methods for general applications such as the Trapezoidal or Adams formulae because they are equally of zero order in the trigonometric sense.

7.1.3 Leap-frog method by using Gautschi's technique

It is well known that the leap-frog method (or sometimes known as the mid-point rule) is an explicit two-step method which is convenient to use, despite, of course, its poor stability property. The method is given by the following relation,

$$y_{n+2} = y_n + 2hf_{n+1}. \quad (7.1.10)$$

From our observation on how the linear multistep methods are related to Gautschi's formulae, i.e. the coefficients of the trigonometric methods are $O(v^2)$ perturbations of those of the corresponding linear multistep methods of Adam's type, where $v=wh$, it makes sense that we extend (7.1.10) correspondingly. The proposed formula will be of the form,

$$y_{n+2} = y_n + h\beta_1(v)f_{n+1} \quad (7.1.11)$$

where the coefficient $\beta_1(v)$ is to be determined so that the formula will have a certain trigonometric order.
Now, if we require (7.1.11) to be of trigonometric order 1, then the following three conditions must be satisfied.

\[ L_w[1;x] = 1 - 1 = 0 \quad (7.1.12) \]
\[ L_w[\cos(wx);h] \equiv \cos(w(x+2h)) - \cos(wx) + hw_1 \sin(w(x+h)) = 0 \quad (7.1.13) \]
\[ L_w[\sin(wx);h] \equiv \sin(w(x+2h)) - \sin(wx) - hw_1 \cos(w(x+h)) = 0 \quad (7.1.14) \]

Now, we see that Eq. (7.1.12) is identically satisfied. For (7.1.13), we use the identity

\[ \cos A - \cos B = -2 \sin \left( \frac{A+B}{2} \right) \sin \left( \frac{A-B}{2} \right) \]

to obtain

\[-2 \sin(w(x+h)) \sin(wh) + wh_1 \sin(w(x+h)) = 0,\]

whence

\[ \beta_1 = \frac{2 \sin(w(x+h)) \sin(wh)}{wh \sin(w(x+h))} \]

\[ = \frac{2 \sin(wh)}{wh} \cdot \]

For (7.1.14) we use the identity

\[ \sin A - \sin B = 2 \cos \left( \frac{A+B}{2} \right) \sin \left( \frac{A-B}{2} \right) \]

to obtain

\[ 2 \cos(w(x+h)) \sin(wh) - wh_1 \cos(w(x+h)) = 0.\]

Thus,

\[ \beta_1 = \frac{2 \cos(w(x+h)) \sin(wh)}{wh \cos(w(x+h))} \]

\[ = \frac{2 \sin(wh)}{wh}, \text{ as before.} \]
Therefore the conditions are just right for us to obtain a consistent solution, and thus produce a formula of trigonometric order 1, i.e.

\[ y_{n+2} = y_n + \frac{2h \sin(v)}{v} f_{n+1} \]  

(7.1.15)

where \( v = \omega h \).

Next, the algebraic order of this formula can be obtained as follows:—  Expanding \( \sin v \) as a Taylor series we have

\[ \sin v = v - \frac{v^3}{6} + \frac{v^5}{120} - \ldots \]  

(7.1.16)

By substituting (7.1.16) into (7.1.15) we have

\[
\begin{align*}
y_{n+2} &= y_n + 2h[1 - \frac{v^2}{6} + \frac{v^4}{120} - \ldots] y'_{n+1} \\
&= y_n + 2h[1 - \frac{\omega^2 h^2}{6} + \frac{\omega^4 h^4}{120} - \ldots] [y'_{n} + hy''_{n} + \frac{h^2}{2} y'''_{n} + \ldots] \\
&= y_n + 2h y'_{n} + 2h^2 y''_{n} + h^3 (y'''_{n} - \frac{\omega^2 y'_{n}}{3} ) + \ldots
\end{align*}
\]

(7.1.17)

By comparing (7.1.17) with the Taylor series expansion of \( y(x_n+2h) \), i.e.

\[ y(x_n+2h) = y_n + (2h)y'_{n} + \frac{(2h)^2}{2} y''_{n} + \frac{(2h)^3}{6} y'''_{n} + \ldots \]

we obtain the local truncation error of the formula which is given by

\[
\text{LTE} = y(x_n+2h) - y_{n+2} \\
= h^3 [\frac{8}{6} y''_{n} - y'''_{n} + \frac{\omega^2 y'_{n}}{3} ] \\
= \frac{1}{3} h^3 (y'''_{n} + \omega^2 y'_{n} ) + O(h^4)
\]

(7.1.18)

Therefore formula (7.1.15) is of algebraic order 2. We note that this formula is explicit and therefore the resulting difference equation can be solved conveniently.
Numerical Example

We solve the problem

$$y' = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} y, \quad y(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

whose exact solution is

$$y = \begin{pmatrix} \cos x \\ \sin x \end{pmatrix}$$

By using formula (7.1.15) for $0 \leq x \leq 10$ the numerical solution is obtained as shown in Table 7.1. It is seen that the solution is exact to the machine accuracy confirming the first order accuracy of the formula trigonometrically.

Table 7.1

<table>
<thead>
<tr>
<th>x</th>
<th>Exact $y_1$</th>
<th>Exact $y_2$</th>
<th>Numerical $y_1$</th>
<th>Numerical $y_2$</th>
<th>Error $y_1$</th>
<th>Error $y_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>0.5403023E+00</td>
<td>0.8414710E+00</td>
<td>0.5403023E+00</td>
<td>0.8414710E+00</td>
<td>-0.9286994E-09</td>
<td>-0.4406342E-08</td>
</tr>
<tr>
<td>2.0</td>
<td>-0.4161468E+00</td>
<td>0.9092974E+00</td>
<td>-0.4161468E+00</td>
<td>0.9092974E+00</td>
<td>0.2136747E-09</td>
<td>-0.5019381E-08</td>
</tr>
<tr>
<td>3.0</td>
<td>-0.9899252E+00</td>
<td>0.1411200E+00</td>
<td>-0.9899252E+00</td>
<td>0.1411200E+00</td>
<td>0.2686793E-08</td>
<td>0.3605956E-09</td>
</tr>
<tr>
<td>4.0</td>
<td>-0.6536436E+00</td>
<td>-0.7568025E+00</td>
<td>-0.6536436E+00</td>
<td>-0.7568025E+00</td>
<td>0.2475443E-08</td>
<td>-0.1117252E-08</td>
</tr>
<tr>
<td>5.0</td>
<td>0.2836622E+00</td>
<td>-0.9589243E+00</td>
<td>0.2836622E+00</td>
<td>-0.9589243E+00</td>
<td>0.8270016E-09</td>
<td>-0.5230228E-09</td>
</tr>
<tr>
<td>6.0</td>
<td>0.9601703E+00</td>
<td>-0.2794155E+00</td>
<td>0.9601703E+00</td>
<td>-0.2794155E+00</td>
<td>0.2863994E-08</td>
<td>0.2086564E-08</td>
</tr>
<tr>
<td>7.0</td>
<td>0.7539023E+00</td>
<td>0.6569866E+00</td>
<td>0.7539023E+00</td>
<td>0.6569866E+00</td>
<td>-0.2444674E-08</td>
<td>-0.4969356E-08</td>
</tr>
<tr>
<td>8.0</td>
<td>0.3455000E+00</td>
<td>0.9893582E+00</td>
<td>0.3455000E+00</td>
<td>0.9893582E+00</td>
<td>-0.1652902E-09</td>
<td>0.2771237E-08</td>
</tr>
<tr>
<td>9.0</td>
<td>-0.9111303E+00</td>
<td>0.4121185E+00</td>
<td>-0.9111303E+00</td>
<td>0.4121185E+00</td>
<td>0.2329089E-08</td>
<td>-0.3096365E-08</td>
</tr>
<tr>
<td>10.0</td>
<td>-0.8390715E+00</td>
<td>-0.5440211E+00</td>
<td>-0.8390715E+00</td>
<td>-0.5440211E+00</td>
<td>-0.1705265E-08</td>
<td>-0.1945853E-08</td>
</tr>
</tbody>
</table>
7.1.4 Runge-Kutta formula for problems with oscillatory solutions

A classical Runge-Kutta method can be regarded as a particular case of the general explicit one-step method,

\[ y_{n+1} = y_n + h \phi (x_n, y_n, h). \]  \hspace{1cm} (7.1.19)

Examples of how these two classes of methods are related are given by the methods stated below, where they are presented in the Runge-Kutta formulation.

1) \( R=1 \), Euler Method

\[ y_{n+1} = y_n + hf(x_n, y_n) \]  \hspace{1cm} (7.1.20)

2) \( R=2 \)

(a) Modified Euler Method

\[
\begin{align*}
k_1 &= f(x_n, y_n) \\
k_2 &= f(x_n + h/2, y_n + \frac{1}{2}hk_1) \\
y_{n+1} &= y_n + hk_2
\end{align*}
\]  \hspace{1cm} (7.1.21)

(b) Improved Euler Method

\[
\begin{align*}
k_1 &= f(x_n, y_n) \\
k_2 &= f(x_n + h, y_n + hk_1) \\
y_{n+1} &= y_n + \frac{1}{2}h(k_1 + k_2)
\end{align*}
\]  \hspace{1cm} (7.1.22)

3) \( R=4 \)

\[
\begin{align*}
k_1 &= f(x_n, y_n) \\
k_2 &= f(x_n + h/2, y_n + \frac{1}{2}hk_1) \\
k_3 &= f(x_n + h/2, y_n + \frac{1}{2}hk_2) \\
k_4 &= f(x_n + h, y_n + hk_3) \\
y_{n+1} &= y_n + \frac{h}{6} (k_1 + 2k_2 + 2k_3 + k_4)
\end{align*}
\]  \hspace{1cm} (7.1.23)
It seems reasonable to assume that there might exist the corresponding Runge-Kutta formula suitable for use in problems with oscillatory solutions. As for the improved Euler method we may consider it as the trapezoidal rule applied in a PECE mode with the Euler method serving as predictor. Similarly for oscillatory problems we might be able to use formula (7.1.5) in a predictor-corrector technique to obtain the 2-stage Runge-Kutta (oscillatory) method,

\[
\begin{align*}
    k_1 &= f(x_n, y_n) \\
    k_2 &= f(x_n + h, y_n + hk_1) \\
    y_{n+1} &= y_n + h \frac{\tan(v/2)}{v} (k_1 + k_2)
\end{align*}
\]

(7.1.24)

We do not expect this method to have trigonometric order one which is the trigonometric order of the base formula (7.1.5). However we may expect that the numerical results would approximate trigonometric sine and cosine functions better than the classical Runge-Kutta formulae of the same order do. Algebraically, formula (7.1.24) is of the second order. This can be established as follows:

By writing

\[
\begin{align*}
    k_1 &= f, \\
    k_2 &= f + h f_y + \frac{1}{2} h f_{yy} + \ldots
\end{align*}
\]

(7.1.25)

in (7.1.25) we have,

\[
\begin{align*}
    y_{n+1} &= y_n + h \left[ \frac{1}{2} + \frac{w^2}{24} + \frac{w^4 h^4}{240} + \ldots \right] \left[ 2f + h f_y + \frac{1}{2} h^2 f_{yy} + \ldots \right] \\
    &= y_n + h \left[ f + \frac{1}{2} h f_y + h^2 \left( \frac{w^2}{12} + \frac{1}{4} f f_{yy} \right) + \ldots \right] \\
    &= y_n + hf + \frac{1}{2} h^2 f_y + h^3 \left[ \frac{w^2}{12} + \frac{1}{4} f f_{yy} \right] + \ldots
\end{align*}
\]

(7.1.26)
By comparing (7.1.26) with the corresponding Taylor series expansion of $y(x_{n+1})$ in terms of $f$ and its partial derivatives we obtain the local truncation error which is given by

$$
LTE = h^3 \left( \frac{1}{6} f_y f_y^2 + \frac{1}{6} f_y f_{yy}^2 - \frac{1}{12} w^2 - \frac{1}{4} f_{yy}^2 \right)
$$

This error term confirms the second order accuracy of the formula.

7.1.5 Higher order Runge-Kutta (oscillatory) formula

We notice that the trigonometric order of a multistep (Gautschi) formula is maximal, in the sense that the parameters solve all the equations of conditions, if the step number is odd and in which case the formula is implicit. This suggests that any Runge-Kutta (oscillatory) formula which we are looking for, should use an implicit $k$-step method with $k = \text{odd number}$ as the base formula. In addition, for this purpose, the first characteristic polynomial must be of the form

$$
\rho(\xi) = \xi^k - 1,
$$

and the second characteristic polynomial is strictly of degree $k$ (i.e. an implicit formula).

A new formula

By taking $k = 3$ we shall first obtain a base formula of the form

$$
y_{n+3} = y_n + h[\beta_3 f_{n+3} + \beta_2 f_{n+2} + \beta_1 f_{n+1} + \beta_0 f_n]
$$

which will have an algebraic order $p=4$ and a trigonometric order $q=2$. The equation of conditions that has to be satisfied is given by
\[
\begin{bmatrix}
-\sin(wx) & -\sin(wx+v) & -\sin(wx+2v) & -\sin(wx+3v) \\
\cos(wx) & \cos(wx+v) & \cos(wx+2v) & \cos(wx+3v) \\
-2\sin(2wx) & -2\sin(2wx+2v) & -2\sin(2wx+4v) & -2\sin(2wx+6v) \\
2\cos(2wx) & 2\cos(2wx+2v) & 2\cos(2wx+4v) & 2\cos(2wx+6v)
\end{bmatrix}
\begin{bmatrix}
\beta_0 \\
\beta_1 \\
\beta_2 \\
\beta_3
\end{bmatrix}
= \begin{bmatrix}
\cos(wx+3v) - \cos(wx) \\
\sin(wx+3v) - \sin(wx) \\
\cos(2wx+6v) - \cos(2wx) \\
\sin(2wx+6v) - \sin(2wx)
\end{bmatrix}
\]

Since this equation is true for any value of \(wx\), then for the sake of simplicity in the algebra involved we may replace \(wx=0\) in the solution procedure. The solution of the system is obtained by using the REDUCE symbolic computation package and is given as follows,

\[
\beta_0 = (3*(412992\cdot v^{16}-4280400\cdot v^{14}-781610\cdot v^{12}-12319825\cdot v^{10} +451324475\cdot v^{8}-270016250\cdot v^{6}+29473500\cdot v^{4}-416500\cdot v^{2} +2940000))/(140\cdot(2016\cdot v^{12}-104600\cdot v^{10}+946545\cdot v^{8} -1341000\cdot v^{6}+207900\cdot v^{4}-280000\cdot v^{2}+168000)),
\]

\[
\beta_1 = (27*(-62208\cdot v^{12}+750600\cdot v^{10}-1780875\cdot v^{8}+830425\cdot v^{6} -107800\cdot v^{4}-24500\cdot v^{2}+14000))/(2*(2016\cdot v^{12}-104600\cdot v^{10}+946545\cdot v^{8} -1341000\cdot v^{6}+207900\cdot v^{4}-280000\cdot v^{2} +168000)),
\]

\[
\beta_2 = (27*(8262\cdot v^{12}-287685\cdot v^{10}+2144295\cdot v^{8}-1570450\cdot v^{6} +294700\cdot v^{4}-49000\cdot v^{2}+28000))/(4*(2016\cdot v^{12}-104600\cdot v^{10}+946545\cdot v^{8} -1341000\cdot v^{6}+207900\cdot v^{4}-280000\cdot v^{2} +168000)),
\]

\[
\beta_3 = (15*(-384\cdot v^{12}+14536\cdot v^{10}-118043\cdot v^{8}+149315\cdot v^{6}-68880\cdot v^{4}-11900\cdot v^{2}+8400))/(2*(2016\cdot v^{12}-104600\cdot v^{10}+946545\cdot v^{8} -1341000\cdot v^{6}+207900\cdot v^{4}-280000\cdot v^{2}+168000)).
\]
As we can see, the solutions are given as rational functions. To obtain a power series form of the solution we perform the division as follows:

Let

\[ p(x) = p_0 + p_2 x^2 + p_4 x^4 + \ldots, \]
\[ q(x) = q_0 + q_2 x^2 + q_4 x^4 + \ldots, \]

and

\[ h(x) = h_0 + h_2 x^2 + h_4 x^4 + \ldots, \]

where \( h(x) = p(x)/q(x) \) is still to be determined. Writing

\[ h(x)q(x) = p(x) \]

and expanding we obtain

\[ h_0 q_0 + h_0 q_2 + h_2 q_0 + h_2 q_2 + h_4 q_0 x^2 + h_4 q_4 x^4 + \ldots = p_0 + p_2 x^2 + p_4 x^4 + \ldots \]

Equating similar coefficients produces

\[ h_0 q_0 = p_0 \implies h_0 = p_0/q_0 \]
\[ h_0 q_2 + h_2 q_0 = p_2 \implies h_2 = (p_2 - h_0 q_2)/q_0 \]
\[ h_0 q_4 + h_2 q_2 + h_4 q_0 = p_4 \implies h_4 = (p_4 - h_0 q_4 - h_2 q_2)/q_0 \]
\[ h_0 q_6 + h_2 q_4 + h_4 q_2 + h_6 q_0 = p_6 \implies h_6 = (p_6 - h_0 q_6 - h_2 q_4 - h_4 q_2)/q_0 \]
\[ h_0 q_8 + h_2 q_6 + h_4 q_4 + h_6 q_2 + h_8 q_0 = p_8 \implies h_8 = (p_8 - h_0 q_8 - h_2 q_6 - h_4 q_4 - h_6 q_2)/q_0 \]
\[ h_0 q_{10} + h_2 q_8 + h_4 q_6 + h_6 q_4 + h_8 q_2 + h_{10} q_0 = p_{10} \implies \]
\[ h_{10} = (p_{10} - h_0 q_{10} - h_2 q_8 - h_4 q_6 - h_6 q_4 - h_8 q_2)/q_0 \]

By applying this procedure to the solution functions given by \( \beta_0, \beta_1, \beta_2, \) and \( \beta_3 \) we obtain

\[ \beta_0 = \frac{3}{8} \left( 1 + \frac{1}{4} v^2 + \frac{2209}{240} v^4 - \frac{1387597}{20160} v^6 + \frac{28749227}{1209600} v^8 \right. \]
\[ + \left. \frac{2805517639}{14515200} v^{10} + \ldots \right), \]
The method is thus given by,

\[ y_{n+3} = y_n + 3h (\beta_0 f_n + \beta_1 f_{n+1} + \beta_2 f_{n+2} + \beta_3 f_{n+3}), \]  

(7.1.29)

where

\[ \beta_0 = \frac{1}{8} (1 + \frac{1}{4} v^2 + \frac{2209}{240} v^4 + \ldots ) \]

\[ \beta_1 = \frac{3}{8} (1 - \frac{1}{12} v^2 - \frac{1307}{720} v^4 + \ldots ) \]

\[ \beta_2 = \frac{3}{8} (1 - \frac{1}{12} v^2 + \frac{6587}{720} v^4 + \ldots ) \]

\[ \beta_3 = \frac{1}{8} (1 + \frac{1}{4} v^2 - \frac{433}{48} v^4 + \ldots ) \]

This is a new formula which has an algebraic order 4 and a trigonometric order 2 and could be suitable for use as a basis for developing a 4-stage Runge-Kutta formula for oscillatory problems, if such a method exists?

First we consider the case when \( v=0 \), and instead of using the 3-step formula we use a single step formula with the functions evaluated at the intermediate points. Formula (7.1.29) is transformed into
\[ y_{n+1} = y_n + h \left( w_1 f_n + w_2 f_{n+1/3} + w_3 f_{n+2/3} + w_4 f_{n+1} \right) \]  
(7.1.30)

where temporarily we regard \( w_1 = \frac{1}{8}, w_2 = \frac{3}{8}, w_3 = \frac{3}{8}, w_4 = \frac{1}{8} \). We will now replace \( k_1 = f_n, k_2 = f_{n+1/3}, k_3 = f_{n+2/3}, k_4 = f_{n+1} \) in the form

\[
\begin{align*}
  k_1 &= f(y_n) \\
  k_2 &= f(y_n + ha_1 k_1) \\
  k_3 &= f(y_n + h(a_2 k_1 + a_3 k_2)) \\
  k_4 &= f(y_n + h(a_4 k_1 + a_5 k_2 + a_6 k_3))
\end{align*}
\]  
(7.1.31)

where \( a_1 = \frac{1}{3}, a_2 + a_3 = \frac{2}{3}, a_4 + a_5 + a_6 = 1 \)  
(7.1.32)

The reason for taking \( v = 0 \) is obvious in that we do not intend to relate \( a_i \) with \( v \). But a 4-stage Runge-Kutta formula must satisfy the six equations of conditions, viz.

\[ w_2 a_1 + w_3 a_2 + w_4 a_3 = \frac{1}{6} \]
(7.1.33)

\[ \left( \frac{3}{8} \right) \left( \frac{1}{3} \right) + \left( \frac{3}{8} \right) \left( \frac{2}{3} \right) + \left( \frac{1}{8} \right) (1) = \frac{1}{2} \implies \frac{1}{2} = \frac{1}{2} \]  
(satisfied)

\[ w_2 a_1 + w_3 a_2 = \frac{1}{6} \]
(7.1.34)

\[ \left( \frac{1}{2} \right) \left( \frac{3}{8} \right) \left( \frac{1}{9} \right) + \left( \frac{1}{2} \right) \left( \frac{3}{8} \right) \left( \frac{4}{9} \right) + \left( \frac{1}{2} \right) \left( \frac{1}{8} \right) (1) = \frac{1}{6} \]

\[ \implies \frac{1}{6} = \frac{1}{6} \]  
(satisfied)

\[ w_2 a_1 + w_3 a_2 = \frac{1}{24} \]
(7.1.35)

\[ \left( \frac{1}{6} \right) \left( \frac{3}{8} \right) \left( \frac{1}{27} \right) + \left( \frac{1}{6} \right) \left( \frac{3}{8} \right) \left( \frac{8}{27} \right) + \left( \frac{1}{6} \right) \left( \frac{1}{8} \right) (1) = \frac{1}{24} \]

\[ \implies \frac{1}{24} = \frac{1}{24} \]  
(satisfied)
We must now solve equations (7.1.33), (7.1.34), (7.1.35) for the three unknown parameters \( a_3 \), \( a_5 \) and \( a_6 \). There are two sets of solutions, i.e.

Set 1: \( a_3 = 1, \ a_5 = -1, \ a_6 = 1 \),

Set 2: \( a_3 = -\frac{1}{3}, \ a_5 = 11, \ a_6 = -3 \).

The other parameters are obtained from the relations (7.1.32). The values of the parameters \( a_i \) (\( i=1,2,\ldots,6 \)) for the two sets of solutions are given in Table 7.2.

**Table 7.2**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Set 1</th>
<th>Set 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1 )</td>
<td>( \frac{1}{3} )</td>
<td>( \frac{1}{3} )</td>
</tr>
<tr>
<td>( a_2 )</td>
<td>(-\frac{1}{3})</td>
<td>1</td>
</tr>
<tr>
<td>( a_3 )</td>
<td>1</td>
<td>(-\frac{1}{3})</td>
</tr>
<tr>
<td>( a_4 )</td>
<td>1</td>
<td>-7</td>
</tr>
<tr>
<td>( a_5 )</td>
<td>-1</td>
<td>11</td>
</tr>
<tr>
<td>( a_6 )</td>
<td>1</td>
<td>-3</td>
</tr>
</tbody>
</table>
Accordingly, we have two sets of Runge-Kutta formulae based on the multistep formula (7.1.29). They are given as follows,

\[(i)\]
\[
\begin{align*}
k_1 &= f(x_n, y_n) \\
k_2 &= f(x_n + \frac{h}{3}, y_n + \frac{1}{3}hk_1) \\
k_3 &= f(x_n + \frac{2}{3}h, y_n + h(-\frac{1}{3}k_1+k_2)) \\
k_4 &= f(x_n+h, y_n + h(k_1-k_2+k_3))
\end{align*}
\]
\[
\begin{align*}
\text{and } y_{n+1} &= y_n + h(w_1k_1 + w_2k_2 + w_3k_3 + w_4k_4)
\end{align*}
\]

\[(ii)\]
\[
\begin{align*}
k_1 &= f(x_n, y_n) \\
k_2 &= f(x_n + \frac{1}{3}h, y_n + \frac{1}{3}hk_1) \\
k_3 &= f(x_n + \frac{2}{3}h, y_n + h(k_1-\frac{1}{3}k_2)) \\
k_4 &= f(x_n+h, y_n + h(-7k_1+11k_2-3k_3))
\end{align*}
\]
\[
\begin{align*}
\text{and } y_{n+1} &= y_n + h(w_1k_1 + w_2k_2 + w_3k_3 + w_4k_4)
\end{align*}
\]

where in both cases,

\[
\begin{align*}
w_1 &= \frac{1}{8} \left(1 + \frac{1}{4} v^2 + \frac{2209}{240} v^4 + \ldots\right), \\
w_2 &= \frac{3}{8} \left(1 - \frac{1}{12} v^2 - \frac{1307}{144} v^4 + \ldots\right), \\
w_3 &= \frac{3}{8} \left(1 - \frac{1}{12} v^2 + \frac{6587}{720} v^4 + \ldots\right), \\
w_4 &= \frac{1}{8} \left(1 + \frac{1}{4} v^2 - \frac{433}{48} v^4 + \ldots\right).
\end{align*}
\]

These two formulae are of order 4 (algebraically) when used for solving a general problem of the form \(y' = f(x, y)\) with \(v=wh=0\). In other words, when \(v=0\), these two formulae are exactly the classical Runge-Kutta formula with \(k_i\) (\(i=1,2,3,4\)) evaluated at the points \(x_n\), \(x_n+h/3\), \(x_n+2h/3\) and \(x_n+h\) respectively.

As in the case of the second order formula (7.1.24) we expect this formula will perform better than the classical Runge-Kutta formula when applied to oscillatory problems with a known frequency. However it is
still premature to prove or disprove this expectation at this stage. A numerical experiment would probably verify our finding but a more rigorous error analysis would be necessary to enable any firm conclusion.

Numerical Experiment

We consider the problem of solving the system

\[
\frac{dX}{dx} = \begin{bmatrix}
0 & -1 \\
1 & 0
\end{bmatrix} X, \quad X(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}
\]

The exact solution for this system is given by

\[
X = \begin{bmatrix}
\cos(x) \\
\sin(x)
\end{bmatrix}
\]

which are oscillatory functions with the frequency given by \(w=1\).

In this experiment we solve this problem by using formula (7.1.29), and the Runge-Kutta type formula given by (7.1.36) with \(w=1\) and \(w=0\). Note that by putting \(w=0\) is equivalent to solving by using a 4th order classical Runge-Kutta formula (which has an algebraic order 4). Putting \(w=1\) in (7.1.36) is equivalent to using the new Runge-Kutta formula for oscillatory problems which has an algebraic order 4 and hopefully a trigonometric order 2. (Note that since this is essentially equivalent to a predictor-corrector process, we do not expect to obtain a fully second order method trigonometrically but merely a close approximation to that order asymptotically. The results are given in Table 7.3 where we show only the errors in \(y_1\) and \(y_2\) for the three methods used.
As expected, formula (7.1.29) gives the exact solution because the problem is only first order trigonometrically. However, contrary to expectation by using \( w=0 \) in formula (7.1.36) a more accurate result than that by using \( w=1 \) was given, which means that the classical formula is more accurate than the proposed formula for oscillatory problems.

Since by using the original formula (7.1.29) we obtain the exact solution, this shows that the Runge-Kutta version of the Gautschi formula does not automatically inherit the trigonometric property of the base formula.

This fact is also confirmed by using the proposed simpler 2-stage Runge-Kutta formula (7.1.24) derived from the Gautschi's version of the trapezoidal formula, where using \( w=0 \) gives a more accurate result than using the exact value \( w=1 \). The result is shown in Table 7.4. The reason why these formulae are not as accurate as the corresponding Gautschi formula may be due to the fact that our intermediate values are computed only approximately as opposed to the Gautschi formula. Thus we suspect that the parameters \( a_i \) in the function evaluations are to be related to \( v \) in the same way the parameters \( w_1, w_2, \ldots \) are.

Due to space and time constraints in the preparation of this thesis, this subject will be investigated further at a later date.
Table 7.3

<table>
<thead>
<tr>
<th>$x$</th>
<th>Formula (7.1.29)</th>
<th>Formula (7.1.36) $w = 0$</th>
<th>Formula (7.1.36) $w = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error $y_1$</td>
<td>Error $y_2$</td>
<td>Error $y_1$</td>
</tr>
<tr>
<td>1.0</td>
<td>$-0.6761133E-07$</td>
<td>$0.5065267E-07$</td>
<td>$-0.6607032E-06$</td>
</tr>
<tr>
<td>2.0</td>
<td>$-0.141743E-06$</td>
<td>$-0.6695504E-07$</td>
<td>$-0.156841E-05$</td>
</tr>
<tr>
<td>3.0</td>
<td>$-0.3619075E-07$</td>
<td>$-0.2326451E-06$</td>
<td>$-0.5602855E-06$</td>
</tr>
<tr>
<td>4.0</td>
<td>$0.2288152E-06$</td>
<td>$-0.1910078E-06$</td>
<td>$0.2330195E-05$</td>
</tr>
<tr>
<td>5.0</td>
<td>$0.3357922E-06$</td>
<td>$0.1067563E-06$</td>
<td>$0.4079239E-05$</td>
</tr>
<tr>
<td>6.0</td>
<td>$0.1111806E-06$</td>
<td>$0.4203519E-06$</td>
<td>$0.1788378E-05$</td>
</tr>
<tr>
<td>7.0</td>
<td>$-0.3410894E-06$</td>
<td>$0.3909999E-06$</td>
<td>$0.3450561E-05$</td>
</tr>
<tr>
<td>8.0</td>
<td>$-0.5870722E-06$</td>
<td>$-0.942763E-07$</td>
<td>$-0.6653198E-05$</td>
</tr>
<tr>
<td>9.0</td>
<td>$-0.2770928E-06$</td>
<td>$-0.6157895E-06$</td>
<td>$-0.3651142E-05$</td>
</tr>
<tr>
<td>10.0</td>
<td>$0.4113427E-06$</td>
<td>$-0.6196621E-06$</td>
<td>$0.3937290E-05$</td>
</tr>
</tbody>
</table>

Table 7.4

<table>
<thead>
<tr>
<th>$x$</th>
<th>Formula (7.1.25) $w = 0$</th>
<th>Formula (7.1.25) $w = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error $y_1$</td>
<td>Error $y_2$</td>
</tr>
<tr>
<td>1.0</td>
<td>$0.1331609E-02$</td>
<td>$-0.1001930E-02$</td>
</tr>
<tr>
<td>2.0</td>
<td>$0.3124365E-02$</td>
<td>$0.1160998E-02$</td>
</tr>
<tr>
<td>3.0</td>
<td>$0.1062741E-02$</td>
<td>$0.4885714E-02$</td>
</tr>
<tr>
<td>4.0</td>
<td>$-0.4720187E-02$</td>
<td>$0.4708415E-02$</td>
</tr>
<tr>
<td>5.0</td>
<td>$-0.8139413E-02$</td>
<td>$-0.1791786E-02$</td>
</tr>
<tr>
<td>6.0</td>
<td>$-0.3460423E-02$</td>
<td>$-0.9384079E-02$</td>
</tr>
<tr>
<td>7.0</td>
<td>$0.7039384E-02$</td>
<td>$-0.9307148E-02$</td>
</tr>
<tr>
<td>8.0</td>
<td>$0.1329720E-01$</td>
<td>$0.1033697E-02$</td>
</tr>
<tr>
<td>9.0</td>
<td>$0.7093458E-02$</td>
<td>$0.1322289E-01$</td>
</tr>
<tr>
<td>10.0</td>
<td>$-0.8117106E-02$</td>
<td>$0.1456447E-01$</td>
</tr>
</tbody>
</table>
7.2 A FOURIER SERIES METHOD FOR THE NUMERICAL SOLUTION OF A CLASS OF NONLINEAR PARABOLIC P.D.E. OF THE FORM

\[ \frac{\partial u}{\partial t} = - \frac{1}{2} \left( \frac{\partial u}{\partial x} \right)^2 + \lambda \frac{\partial^2 u}{\partial x^2} + d(x,t) \]

7.2.1 Introduction

So far our attention has been predominantly focussed on the numerical solution of ordinary differential equations. Now, in discussing partial differential equations we shall necessarily be concerned with problems in which the dependent variable is a function of two independent variables. Extension to three or more variables can also be made, details of which are discussed in, for example, Forsythe and Wasow [1960].

Three different classes of differential equations are distinguishable and usually different numerical approaches are used for the solution of each class. A class of differential equation may be made to assume a simple form (usually by a change of variable) called the canonical form for the given class. The various forms and their associated classifications are

\[ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \psi \quad \text{(elliptic)} \]  \hspace{1cm} (7.2.1)

\[ \frac{\partial^2 u}{\partial x^2} = \psi \quad \text{(parabolic)} \]  \hspace{1cm} (7.2.2)

\[ \frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = \psi \quad \text{(hyperbolic)} \]  \hspace{1cm} (7.2.3)

where

\[ \psi = \psi(x,y, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}). \]
The most commonly encountered elliptic equation in physical application is
\[
\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \quad \text{(Laplace's equation)} \quad (7.2.4)
\]
and
\[
\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = k \text{ (constant) \quad (Poisson's equation)} \quad (7.2.5)
\]

For higher order equations, the most common one include amongst others, the fourth order biharmonic expression
\[
\frac{\partial^4 u}{\partial x^4} + \frac{2\partial^4 u}{\partial x^2 \partial y^2} + \frac{\partial^4 u}{\partial y^4} = k. \quad (7.2.6)
\]

The most commonly used parabolic equation has the form
\[
\alpha \frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial y}, \quad (\alpha = \text{constant}) \quad (7.2.7)
\]
This is called the one dimensional transient diffusion or heat equation with \( y \) representing time.

Hyperbolic equations are often used in the form
\[
\frac{\partial^2 u}{\partial x^2} = \alpha \frac{\partial^2 u}{\partial y^2} \quad (\alpha = \text{constant}) \quad (7.2.8)
\]
This is called the wave equation.

For the solutions of elliptic type equations to exist, there is a necessity that boundary conditions be specified at every point of a well defined (closed) boundary. Any local variation in the boundary conditions suffices to alter the solution over the entire region of the problem. Usually the physical application problems dictate the appropriate boundary conditions. The number and type of boundary conditions which completely define a problem makes a unique solution dependent on the nature and order of the equation of the problem.
For parabolic and hyperbolic equations it is not in general necessary to have boundary conditions specified at every point of the boundary. Specification of conditions at only a segment of a boundary is often sufficient to determine the solution within a certain portion of the problem region. The boundary itself is not closed.

As we have explained earlier, there are different approaches used for the solution of each class. In fact there are also various methods that can be employed for each class. Forrington [1963] studied the numerical solution of certain parabolic differential equations in one space variable, the class of which is characterised by having a solution which is periodic in the space variable rather than the more usual case of having prescribed conditions on two boundaries.

In particular he considered the linear parabolic p.d.e. of the form

$$u_t = au_{xx} + bu_x + cu + d$$

(7.2.9)

with the properties:

(1) $a, b, c$ independent of $x$,

(2) $d$ is representable in the form

$$d(x,t) = \frac{1}{2} p_0(t) + \sum_{r=1}^{N} [p_r(t)\cos(\frac{2\pi rx}{L}) + q_r(t)\sin(\frac{2\pi rx}{L})]$$

(7.2.10)

(3) At $t=0$, $u(x,t)$ is representable as

$$u(x,0) = \frac{1}{2} k_0 + \sum_{r=1}^{M} [k_r\cos(\frac{2\pi rx}{L}) + m_r\sin(\frac{2\pi rx}{L})]$$

(7.2.11)

where $k_r, m_r$ are constants.

(4) $u(x,t)$ is periodic, of period $L$, in the space variable.
The method that Forrington developed was based on a Fourier series technique to reduce the problem to a system of o.d.e's with one point boundary conditions which may then be solved by a standard technique. This can lead to a considerable saving in computing time over conventional finite difference methods as well as free from discretization error in the space variable.

7.2.2 The method for nonlinear problems

In this section we shall adopt the Fourier series method for solving the nonlinear parabolic p.d.e. of the form,

\[
\frac{\partial u}{\partial t} = \frac{1}{2} \left( \frac{\partial u}{\partial x} \right)^2 + \lambda \frac{\partial^2 u}{\partial x^2} + d(x,t)
\] (7.2.12)

in the region \(-\infty < x < \infty, \ t > 0\), with the following properties:

1. \(\lambda\) is positive and is independent of \(x\),
2. \(d\) is representable in the form

\[
d(x,t) = \frac{1}{2} p_0(t) + \sum_{r=1}^{N} \left[ p_r(t) \cos\left(\frac{2\pi r x}{L}\right) + q_r(t) \sin\left(\frac{2\pi r x}{L}\right) \right]
\] (7.2.13)

3. At \(t=0\), \(u(x,t)\) is representable as

\[
u(x,0) = \frac{1}{2} k_0 + \sum_{r=1}^{M} \left[ k_r \cos\left(\frac{2\pi r x}{L}\right) + m_r \sin\left(\frac{2\pi r x}{L}\right) \right]
\] (7.2.14)

where \(k_r, m_r\) are constants.

4. \(u(x+L,t) = u(x,t)\),

i.e. \(u(x,t)\) is periodic, of period \(L\), in the space variable.

Reduction to o.d.e. form

We assume there exists a solution of the form:

\[
u(x,t) = \frac{1}{2} f_0(t) + \sum_{r=1}^{\infty} \left[ f_r(t) \cos\left(\frac{2\pi r x}{L}\right) + g_r(t) \sin\left(\frac{2\pi r x}{L}\right) \right]
\] (7.2.15)
which is periodic and of period $L$, and which satisfies the initial conditions (7.2.14) provided

$$
\begin{align*}
    f_0(0) &= k_0 \\
    f_r(0) &= k_r & r = 1, 2, \ldots, M \\
    g_r(0) &= m_r \\
    f_r(0) &= 0 & r > M \\
    g_r(0) &= 0
\end{align*}
$$

(7.2.17)

To enable us to substitute (7.2.16) and (7.2.13) into (7.2.12), we compute the necessary derivatives first, i.e.

$$
\begin{align*}
    u_t &= \frac{1}{2} f_0(t) + \sum_{r=1}^{\infty} [f_r'(t)\cos\left(\frac{2\pi r x}{L}\right) + g_r'(t)\sin\left(\frac{2\pi r x}{L}\right)] \\
    u_x &= \sum_{r=1}^{\infty} [-(\frac{2\pi r}{L}) f_r(t)\sin\left(\frac{2\pi r x}{L}\right) + (\frac{2\pi r}{L}) g_r(t)\cos\left(\frac{2\pi r x}{L}\right)] \\
    u_{xx} &= \sum_{r=1}^{\infty} [-(\frac{2\pi r}{L})^2 f_r(t)\cos\left(\frac{2\pi r x}{L}\right) - (\frac{2\pi r}{L})^2 g_r(t)\sin\left(\frac{2\pi r x}{L}\right)].
\end{align*}
$$

In addition we have to compute $u_x^2$ and express it in the form of

$$
(u_x^2) = \frac{1}{2} a_0 + \sum_{r=1}^{\infty} [a_r\cos\left(\frac{2\pi r x}{L}\right) + b_r\sin\left(\frac{2\pi r x}{L}\right)],
$$

where the $a_r$ and $b_r$ are related to $f_r$ and $g_r$ in some form. Generally we shall say that this is the most laborious part in the application of the technique especially if a conventional computing procedure is to be used. However, this difficulty is greatly reduced by making use of the availability of the symbolic computation packages to obtain an explicit expression of $u_x^2$ in terms of the coefficients $f_r(t)$ and $g_r(t)$. But still, the degree of complexity is grossly magnified with an increase in the number of $f_r$ and $g_r$ terms that are going to be nonzero, i.e.
with the increase of $\gamma = \max(M,N)$. Also, as a general rule, $u_x^2$ contains twice as many terms as $u_x$. This explains the increase in the complexity which we mentioned earlier.

To obtain the relation between $a_r$, $b_r$, $f_r$, $g_r$ we proceed as follows:

Define the operators $c$, $s$, $f$, $g$ to mean

$$c(r) = \cos\left(\frac{2\pi r x}{L}\right),$$

$$s(r) = \sin\left(\frac{2\pi r x}{L}\right),$$

$$f(r) = \left(\frac{2\pi r}{L}\right)f_r(t),$$

and

$$g(r) = \left(\frac{2\pi r}{L}\right)g_r(t).$$

By using the properties of the trigonometric functions we also define the following properties of the operators $c$ and $s$ to enable us to convert the trigonometric products into trigonometric basis functions, viz.

$$s(m)s(n) = \frac{c(m-n)-c(m+n)}{2},$$

$$c(m)c(n) = \frac{c(m-n)+c(m+n)}{2},$$

$$c(-m) = c(m),$$

$$s(-m) = -s(m),$$

$$c(0) = 1,$$

$$s(0) = 0.$$

Now, writing $u_x$ as

$$u_x = \sum_{r=1}^{\infty} \left[ -f(r)s(r) + g(r)c(r) \right]$$

we obtain

$$(u_x^2) = \left( \sum_{r=1}^{\infty} \left[ -f(r)s(r) + g(r)c(r) \right] \right)^2$$

(7.2.18)
which is understandably complicated to write in the expanded form and as a linear combination of the trigonometric basis functions.

By limiting the number of terms in (7.2.18) so that \( r \) runs from 1 to \( \gamma \) say, we obtain a rather convenient expression in the square. For example with \( \gamma = 6 \), the following expression is obtained.

\[
(\mathbf{r}_x) = \frac{1}{2} a_0 + \sum_{r=1}^{12} \left[ a_r \cos\left(\frac{2\pi r x}{L}\right) + b_r \sin\left(\frac{2\pi r x}{L}\right) \right]
\]

(7.2.20)

where

\[
a_0 = f(1)^2 + g(1)^2 + \ldots + f(6)^2 + g(6)^2
\]

\[
= \sum_{r=1}^{6} [f(r)^2 + g(r)^2]
\]

\[
a_1 = f(1)f(2) + g(1)g(2) + \ldots + f(5)f(6) + g(5)g(6)
\]

\[
= \sum_{r=1}^{5} [f(r)f(r+1) + g(r)g(r+1)]
\]

\[
b_1 = f(1)g(2) + g(1)f(2) + \ldots + f(5)g(6) + g(5)f(6)
\]

\[
= \sum_{r=1}^{5} [f(r)g(r+1) - g(r)f(r+1)]
\]

etc.

where

\[
f(r) = \left(\frac{2\pi r}{L}\right)f_r(t) \quad \text{and} \quad g(r) = \left(\frac{2\pi r}{L}\right)g_r(t).
\]

The complete relation obtained by using the REDUCE symbolic computation package can be seen in Appendix 7.

Substituting these expressions into (7.2.9) we obtain

\[
\frac{1}{2} f_0^t + \sum_{r=1}^{t} \left\{ f_r^t \cos\left(\frac{2\pi r x}{L}\right) + g_r^t \sin\left(\frac{2\pi r x}{L}\right) \right\}
\]

\[
= -\frac{1}{2} \left[ \frac{1}{2} a_0 + \sum_{r=1}^{t} \left\{ a_r \cos\left(\frac{2\pi r x}{L}\right) + b_r \sin\left(\frac{2\pi r x}{L}\right) \right\} \right]
\]

\[
+ \lambda \sum_{r=1}^{t} \left\{ -\left(\frac{2\pi r}{L}\right)^2 f_r \cos\left(\frac{2\pi r x}{L}\right) - \left(\frac{2\pi r}{L}\right)^2 g_r \sin\left(\frac{2\pi r x}{L}\right) \right\}
\]

\[
+ \frac{1}{2} p_0(t) + \sum_{r=1}^{N} \left\{ p_r \cos\left(\frac{2\pi r x}{L}\right) + q_r \sin\left(\frac{2\pi r x}{L}\right) \right\}
\]
By equating the coefficients of sine and cosine on the LHS and RHS we obtain the following relations:-

Terms
independent of $x$:

$$\frac{1}{2} f_0' = -\frac{1}{4} a_0 + \frac{1}{2} p_0$$  \hspace{1cm} (7.2.22a)$$

$$\cos\left(\frac{2\pi r x}{L}\right): \quad f_r' = -\frac{1}{2} a_r - \lambda \left(\frac{2\pi r}{L}\right)^2 f_r + p_r$$  \hspace{1cm} (7.2.22b)$$

$$\sin\left(\frac{2\pi r x}{L}\right): \quad g_r' = -\frac{1}{2} b_r - \lambda \left(\frac{2\pi r}{L}\right)^2 g_r + q_r,$$  \hspace{1cm} (7.2.22c)$$

where $a_r$ and $b_r$ are functions of $f_r$ and $g_r$, and $p_r$ and $q_r$ are zero for $r > N$.

It is interesting to note that for any $r > \gamma$, $a_r$ and $b_r$ are dependent on $f_i$ and $g_i$, $i=1,2,\ldots,\gamma$. This means that for $r > \gamma = \max(M,N)$ the differential equations (7.2.22b) and (7.2.22c) with the related zero initial conditions will still have a solution that is not identically zero. Naturally one will ask whether this will lead to an infinite sequence of $\{f_r\}$ and $\{g_r\}$? Luckily this is not the case, since for $j > 2\gamma$ each factor in $a_j$ and $b_j$ must appear as a multiple of $f_r$ and $g_r$ with $r > \gamma$. This leads to Eqs. (7.2.22b) and (7.2.22c) to have a trivial solution for $r > 2\gamma$.

Thus we have replaced the original p.d.e. by a finite system of o.d.e's, i.e.,

$$\frac{df_0}{dt} = -\frac{1}{2} a_0 + p_0$$

$$\frac{df_r}{dt} = -\frac{1}{2} a_r - \lambda \left(\frac{2\pi r}{L}\right)^2 f_r + p_r$$ \hspace{1cm} (r=1,2,\ldots,2\gamma)$$

$$\frac{dg_r}{dt} = -\frac{1}{2} b_r - \lambda \left(\frac{2\pi r}{L}\right)^2 g_r + q_r$$
with one point boundary conditions

\[
\begin{align*}
    f_0(0) &= k_0 \\
f_r(0) &= k_r & r &= 1, 2, \ldots, M \\
g_r(0) &= m_r \\
f_r(0) &= 0 & r &= M+1, \ldots, 2M \\
g_r(0) &= 0
\end{align*}
\]

and

We note that so far there has been no discretisation of the problem and hence no error has been introduced. This system of equations may then be solved by standard techniques.

While the Fourier series method for linear problems results in solving \(2M+1\) equations, this method results in solving \(4M+1\) equations. This is the cost that we have to pay in dealing with the more difficult nonlinear problem. However if the convergence of the Fourier series representing the solution is rapid, we may choose a fewer number of equations to be solved. This will amount to testing the convergence of the series at each step of the integration.

7.2.3 Numerical Example

The burning of a gas in a rocket is described by Forsythe and Wasow [1960, p.141] in terms of the equation

\[
\frac{\partial u}{\partial t} = -i\left(\frac{\partial u}{\partial x}\right)^2 + \lambda \frac{\partial^2 u}{\partial x^2} + d(x,t) \tag{7.2.23}
\]

for \(-\infty < x < \infty, t > 0\), where \(\lambda\) is a positive constant and \(d(x,t)\) is periodic of period \(2\pi\) in \(x\). The auxiliary conditions are

\[
u(x,0) = \phi(x), \quad u(x+2\pi,t) = u(x,t)
\]

where \(\phi(x)\) is periodic of period \(2\pi\).
In this example we will solve this problem with $\lambda = 2$ and

$$d(x,t) = \left[ 2t + \frac{t^2}{4} + \frac{1}{4(1+t)^2} \right] - (1+2t)\cos x$$

$$+ \left[ \frac{2}{1+t} - \frac{1}{(1+t)^2} \right] \sin x$$

$$+ \left[ \frac{1}{4(1+t)^2} - \frac{t^2}{4} \right] \cos 2x + \frac{t}{2(1+t)} \sin 2x.$$ 

with the boundary conditions at $t=0$ given by $u(x,0) = 1 + \sin x$.

The exact solution to this problem is given by

$$u(x,t) = (t^2+1) - t\cos x + \frac{1}{1+t} \sin x,$$

which is periodic with period $2\pi$. (Note: This problem may not represent a real physical phenomenon, but the nature of the equation and its known solution may prove useful to show the credibility of the method).

With $L=2\pi$, the Fourier series coefficients for $d(x,t)$ are given by

$$p_0 = 4t + \frac{1}{2}t^2 + \frac{1}{2(1+t)^2}.$$ 

$$p_1 = -(1+2t)$$

$$q_1 = \frac{2}{1+t} - \frac{1}{(1+t)^2}.$$ 

$$p_2 = \frac{1}{4(1+t)^2} - \frac{t^2}{4}.$$ 

$$q_2 = \frac{t}{2(1+t)}.$$ 

(i.e. $N=2$)

Also,

$k_0=2$, $k_1=0$, $m_1=1$ (i.e. $N=1$).
Note that with $L=2\pi$,

$$f(r) = rf_r \text{ and } g(r) = rg_r, \ r \geq 1.$$ 

Thus we have, with $\gamma = \max(M,N)=2$,

\[
\begin{align*}
a_0 &= f_1^2 + g_1^2 + 4(f_2^2 + g_1^2) + 9(f_3^2 + g_2^2) + 16(f_4^2 + g_3^2) \\
a_1 &= 2(f_1 f_2 + g_1 g_2) + 6(f_2 f_3 + g_2 g_3) + 12(f_3 f_4 + g_3 g_4) \\
b_1 &= 2(f_1 g_2 - f_2 g_1) + 6(f_2 g_3 - f_3 g_2) + 12(f_3 g_4 - f_4 g_3) \\
a_2 &= \frac{1}{4}(-f_1^2 + g_1^2) + 3(f_1 f_3 + g_1 g_3) + 8(f_2 f_4 + g_2 g_4) \\
b_2 &= -f_1 g_1 + 3(f_1 g_3 - f_3 g_1) + 8(f_2 g_4 - f_4 g_2) \\
a_3 &= 2(-f_1 f_2 + g_1 g_2) + 4(f_1 f_4 + g_1 g_4) \\
b_3 &= 2(-f_1 g_2 - f_2 g_1) + 4(f_1 g_4 - f_4 g_1) \\
a_4 &= 2(-f_2^2 + g_2^2) + 3(-f_1 f_3 + g_1 g_3) \\
b_4 &= 3(-f_1 g_3 - f_3 g_1) - 4f_2 g_2.
\end{align*}
\]

With all the terms as given above, we now have to solve the following system of equations:

\[
\begin{align*}
\frac{df_0}{dt} &= -\frac{1}{2}a_0 + p_0 \\
\frac{df_r}{dt} &= -\frac{1}{2}a_r - 2r^2 f_r + p_r \\
\frac{dg_r}{dt} &= -\frac{1}{2}b_r - 2r^2 g_r + q_r
\end{align*}
\]

In this example we use the NAG Fortran Library Routine D02BBF to solve the resulting system of o.d.e's. The routine integrates the system

$$\frac{dy_i}{dt} = f_i(t, y_1, y_2, \ldots, y_n)$$

from $t=t_0$ to $t=t_{\text{end}}$ using the Merson form of the Runge-Kutta method. The solution at the points specified by the user is obtained by quintic Hermite interpolation on the solution values produced by the Runge-Kutta method. The accuracy of the integration and, indirectly,
the interpolation is controlled by the parameters tol and irelab.

In the following we give the numerical solutions which are presented in three cases:-

**Case I:** We use all the coefficients \( a_r \) and \( b_r \) as given by Eq. (7.2.24) and solve all the nine equations (7.2.25) where \( \gamma = 2 \). The errors in the solution for \( 0 \leq x \leq 2\pi \) and \( 0 \leq t \leq 10 \) are given in Table 7.5.

**Case II:** We use all the coefficients \( a_r \) and \( b_r \) in (7.2.24) putting \( f_3 = g_3 = f_4 = g_4 = 0 \) in (7.2.25) and solve the nine equations (\( \gamma = 2 \)). The errors in the solution are given in Table 7.6.

**Case III:** Using the first three simple equations where we assume \( f_1 = g_1 = 0 \) for \( i > 1 \) in all computations. The errors in the numerical result obtained are given in Table 7.7.

In all cases we use \( \text{tol} = 10^{-5} \) and the results obtained are found to be as accurate as desired, and in addition the computation time is reduced by using a fewer number of equations in the system of o.d.e's. This shows that the maximum number of equations should not necessarily be used. In practice, a proper examination on the convergence of the Fourier series (solution) might be able to help us to determine how many equations should be used.

Note that in a general purpose implementation, the quantities \( a_i \) and \( b_i \) can be conveniently written in the computer program to the maximum number we desire, and we can use them whether in full or in part as this example shows.
Table 7.5
Errors in the solutions in Case I
neq=9, tol=0.1E-06, irelab=0,
dx = 0.4π

<table>
<thead>
<tr>
<th>X</th>
<th>0.0</th>
<th>0.4π</th>
<th>0.8π</th>
<th>1.2π</th>
<th>1.6π</th>
<th>2.0π</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
<td>0.14901E-07</td>
<td>0.37253E-08</td>
<td>0.46566E-09</td>
<td>0.14901E-07</td>
</tr>
<tr>
<td>0.2</td>
<td>0.74506E-08</td>
<td>-0.29684E-07</td>
<td>-0.14828E-07</td>
<td>0.14828E-07</td>
<td>0.36071E-08</td>
<td>0.74506E-08</td>
</tr>
<tr>
<td>0.4</td>
<td>0.0000E+00</td>
<td>-0.44509E-07</td>
<td>-0.14781E-07</td>
<td>0.29682E-07</td>
<td>0.35304E-08</td>
<td>0.0000E+00</td>
</tr>
<tr>
<td>0.6</td>
<td>0.74506E-08</td>
<td>-0.58822E-07</td>
<td>-0.29319E-07</td>
<td>0.44220E-07</td>
<td>-0.78248E-09</td>
<td>0.74506E-08</td>
</tr>
<tr>
<td>0.8</td>
<td>0.22352E-07</td>
<td>-0.74244E-07</td>
<td>-0.29641E-07</td>
<td>0.44542E-07</td>
<td>-0.77120E-08</td>
<td>0.22352E-07</td>
</tr>
<tr>
<td>1.0</td>
<td>0.21759E-07</td>
<td>-0.59649E-07</td>
<td>-0.29830E-07</td>
<td>0.59632E-07</td>
<td>-0.14857E-07</td>
<td>0.21759E-07</td>
</tr>
</tbody>
</table>

Table 7.6
Errors in the solutions in Case II
neq=9, tol=0.1E-06, irelab=0,
dx = 0.4π

<table>
<thead>
<tr>
<th>X</th>
<th>0.0</th>
<th>0.4π</th>
<th>0.8π</th>
<th>1.2π</th>
<th>1.6π</th>
<th>2.0π</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
<td>0.14901E-07</td>
<td>0.37253E-08</td>
<td>0.46566E-09</td>
<td>0.14901E-07</td>
</tr>
<tr>
<td>0.2</td>
<td>0.74506E-08</td>
<td>-0.29802E-07</td>
<td>-0.14901E-07</td>
<td>0.14901E-07</td>
<td>0.37253E-08</td>
<td>0.74506E-08</td>
</tr>
<tr>
<td>0.4</td>
<td>0.0000E+00</td>
<td>-0.44703E-07</td>
<td>-0.14901E-07</td>
<td>0.29802E-07</td>
<td>0.37253E-08</td>
<td>0.0000E+00</td>
</tr>
<tr>
<td>0.6</td>
<td>0.74506E-08</td>
<td>-0.59605E-07</td>
<td>-0.29802E-07</td>
<td>0.44703E-07</td>
<td>-0.13272E-13</td>
<td>0.74506E-08</td>
</tr>
<tr>
<td>0.8</td>
<td>0.22352E-07</td>
<td>-0.74506E-07</td>
<td>-0.29802E-07</td>
<td>0.44703E-07</td>
<td>-0.74506E-08</td>
<td>0.22352E-07</td>
</tr>
<tr>
<td>1.0</td>
<td>0.22243E-07</td>
<td>-0.59605E-07</td>
<td>-0.29802E-07</td>
<td>0.59605E-07</td>
<td>-0.14901E-07</td>
<td>0.22243E-07</td>
</tr>
</tbody>
</table>
Table 7.7

Errors in the solutions in Case III
neq=3, tol=0.1E-06, irelab=0,
dx = 0.4π

<table>
<thead>
<tr>
<th>t</th>
<th>0.0</th>
<th>0.4π</th>
<th>0.8π</th>
<th>1.2π</th>
<th>1.6π</th>
<th>2.0π</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.00000E+00</td>
<td>-0.60851E-08</td>
<td>0.65967E-08</td>
<td>0.85386E-09</td>
<td>0.31520E-10</td>
<td>0.86736E-18</td>
</tr>
<tr>
<td>0.2</td>
<td>0.00000E+00</td>
<td>-0.19661E-07</td>
<td>-0.91259E-08</td>
<td>0.52609E-08</td>
<td>0.11611E-07</td>
<td>0.75894E-18</td>
</tr>
<tr>
<td>0.4</td>
<td>-0.14901E-07</td>
<td>-0.51373E-07</td>
<td>-0.39564E-07</td>
<td>0.97614E-08</td>
<td>0.15058E-07</td>
<td>-0.14901E-07</td>
</tr>
<tr>
<td>0.6</td>
<td>-0.29802E-07</td>
<td>-0.92979E-07</td>
<td>-0.51591E-07</td>
<td>0.33891E-07</td>
<td>0.46241E-07</td>
<td>-0.29802E-07</td>
</tr>
<tr>
<td>0.8</td>
<td>-0.29802E-07</td>
<td>-0.12644E-06</td>
<td>-0.98432E-07</td>
<td>0.34118E-07</td>
<td>0.67479E-07</td>
<td>-0.29802E-07</td>
</tr>
<tr>
<td>1.0</td>
<td>-0.44703E-07</td>
<td>-0.14800E-06</td>
<td>-0.11035E-06</td>
<td>0.50748E-07</td>
<td>0.81953E-07</td>
<td>-0.44703E-07</td>
</tr>
</tbody>
</table>
7.2.4 A comparison between the Fourier series technique and the method of lines in terms of computational complexity and accuracy.

In a preceding subsection we have confirmed that the Fourier series technique is applicable to use to solve non-linear problems of the form,

\[ \frac{\partial u}{\partial t} = -i\left(\frac{\partial u}{\partial x}\right)^2 + \lambda \frac{\partial^2 u}{\partial x^2} + d(x,t). \]

We have also assumed that the solutions obtained are more accurate than any other solutions obtained by discrete methods due to the fact that no discretisation error is made in the space variable.

In the present subsection we shall compare how much advantage we gain by using the technique as compared to using the method of lines.

For this purpose we shall also solve the example presented in subsection 7.2.3 by using a method of lines. We also solve it by making use of routine D03PAF of the NAG library. This routine integrates a single linear or non-linear parabolic p.d.e. in one space variable, using the method of lines and Gear's method. In this routine, the parabolic equation is approximated by a system of o.d.e's, obtained by replacing the space derivatives by finite differences. This system is then integrated forwards in time using Gear's method. The approximation uses a uniform mesh in the space direction; in the time direction the interval is chosen by the routine to maintain the local accuracy specified by the user for the time integration. The routine, however, does not check the accuracy of the finite-difference approximation in the space dimension.

In our comparison we list the results at fixed equidistant points \( x = 0(0.4\pi)2\pi \) and \( t = 0(1.0)10.0 \). We also compute the actual errors \( e_{ij} \) to
the solutions at these 66 points and represent them in the form of the 
$L_2$-error norm,

$$\|e\| = \sqrt{\sum_{i} \sum_{j} e_{ij}^2}.$$ 

Therefore, $\|e\|$ represents the overall errors within the region under 
consideration. Our results reveal the following:-

(R1) With 6 grid points in the x-direction, the Fourier series method 
produces

$$T_{\text{Fourier}} \approx 0.31 \text{ sec.}$$
$$\|e\|_{\text{Fourier}} = 0.6 \times 10^{-5} \quad \text{(Table 7.8)}$$

(R2) With 11 grid points in the x-direction, the method of lines 
produces

$$T_{\text{line}} \approx 8 \text{ sec.}$$
$$\|e\|_{\text{line}} = 3.6 \quad \text{(Table 7.9)}$$

(R3) By doubling the number of grid points in the x-direction, the 
accuracy in the method of lines is improved, but the computing time is 
doubled. With 21 points we obtained

$$T_{\text{line}} \approx 16.6 \text{ sec.}$$
$$\|e\|_{\text{line}} = 0.88 \quad \text{(Table 7.10)}$$

For higher accuracy, even more time is needed in the method of lines. 
The fact is that given a fixed stepsize in the t direction, the 
application of the method of lines with arbitrarily small stepsize in 
the x-direction would not produce as accurate a result as the Fourier 
series method would. By taking this advantage we would be able, 
therefore, to increase the stepsize in the t-direction of the Fourier 
series method to obtain a result that is as competitive as the one
obtained by using small discretization in the x-direction in the method of lines but with using only a small fraction of the computing time needed in the method of lines.

Further investigation (Table 7.11) shows that by setting error tolerance $= 10^{-2}$ in the Fourier series method we are able to reach $\|e\|_{\text{Fourier}} = 0.88$, i.e. of the same magnitude as in (R3). But in this case the computing time is 0.154 sec. as compared to 16.5 sec. in (R3). We therefore conclude that with the Fourier series method we can obtain the same degree of accuracy as the method of lines but approximately $16.5/0.154 = 100$ times faster.
Table 7.8
Solution by using Fourier series technique
tol = $0.100E-06$

<table>
<thead>
<tr>
<th>$t$</th>
<th>0.0</th>
<th>0.4$\pi$</th>
<th>0.8$\pi$</th>
<th>1.2$\pi$</th>
<th>1.6$\pi$</th>
<th>2.0$\pi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.10000E+01</td>
<td>0.19511E+01</td>
<td>0.15878E+01</td>
<td>0.41221E+00</td>
<td>0.48943E-01</td>
<td>0.10000E+01</td>
</tr>
<tr>
<td>1.0</td>
<td>0.10000E+01</td>
<td>0.21665E+01</td>
<td>0.31029E+01</td>
<td>0.25151E+01</td>
<td>0.12155E+01</td>
<td>0.10000E+01</td>
</tr>
<tr>
<td>2.0</td>
<td>0.30000E+01</td>
<td>0.46990E+01</td>
<td>0.68140E+01</td>
<td>0.64221E+01</td>
<td>0.40649E+01</td>
<td>0.30000E+01</td>
</tr>
<tr>
<td>3.0</td>
<td>0.70000E+01</td>
<td>0.93107E+01</td>
<td>0.12574E+02</td>
<td>0.12280E+02</td>
<td>0.88352E+01</td>
<td>0.70000E+01</td>
</tr>
<tr>
<td>4.0</td>
<td>0.13000E+02</td>
<td>0.15954E+02</td>
<td>0.20354E+02</td>
<td>0.20119E+02</td>
<td>0.15574E+02</td>
<td>0.13000E+02</td>
</tr>
<tr>
<td>5.0</td>
<td>0.21000E+02</td>
<td>0.24613E+02</td>
<td>0.30143E+02</td>
<td>0.29947E+02</td>
<td>0.24296E+02</td>
<td>0.21000E+02</td>
</tr>
<tr>
<td>6.0</td>
<td>0.31000E+02</td>
<td>0.35282E+02</td>
<td>0.41938E+02</td>
<td>0.41770E+02</td>
<td>0.35010E+02</td>
<td>0.31000E+02</td>
</tr>
<tr>
<td>7.0</td>
<td>0.43000E+02</td>
<td>0.47956E+02</td>
<td>0.55737E+02</td>
<td>0.55590E+02</td>
<td>0.47718E+02</td>
<td>0.43000E+02</td>
</tr>
<tr>
<td>8.0</td>
<td>0.57000E+02</td>
<td>0.62634E+02</td>
<td>0.71537E+02</td>
<td>0.71407E+02</td>
<td>0.62422E+02</td>
<td>0.57000E+02</td>
</tr>
<tr>
<td>9.0</td>
<td>0.73000E+02</td>
<td>0.79314E+02</td>
<td>0.89340E+02</td>
<td>0.89222E+02</td>
<td>0.79124E+02</td>
<td>0.73000E+02</td>
</tr>
<tr>
<td>10.0</td>
<td>0.91000E+02</td>
<td>0.97996E+02</td>
<td>1.0914E+03</td>
<td>1.0904E+03</td>
<td>0.97823E+02</td>
<td>0.91000E+02</td>
</tr>
</tbody>
</table>
Table 7.9
Solution by using NAG routine D03PAF
(method of lines)
tol = 0.100E-06, dx = 0.2\pi (11 points)

<table>
<thead>
<tr>
<th>x</th>
<th>0.0</th>
<th>0.4\pi</th>
<th>0.8\pi</th>
<th>1.2\pi</th>
<th>1.6\pi</th>
<th>2.0\pi</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.10000E+01</td>
<td>0.19511E+01</td>
<td>0.15878E+01</td>
<td>0.41221E+00</td>
<td>0.48943E-01</td>
<td>0.10000E+01</td>
</tr>
<tr>
<td>1.0</td>
<td>0.10000E+01</td>
<td>0.21993E+01</td>
<td>0.31443E+01</td>
<td>0.25400E+01</td>
<td>0.12100E+01</td>
<td>0.10000E+01</td>
</tr>
<tr>
<td>2.0</td>
<td>0.30000E+01</td>
<td>0.47735E+01</td>
<td>0.69299E+01</td>
<td>0.65277E+01</td>
<td>0.41084E+01</td>
<td>0.30000E+01</td>
</tr>
<tr>
<td>3.0</td>
<td>0.70000E+01</td>
<td>0.94527E+01</td>
<td>0.12818E+02</td>
<td>0.12518E+02</td>
<td>0.89534E+01</td>
<td>0.70000E+01</td>
</tr>
<tr>
<td>4.0</td>
<td>0.13000E+02</td>
<td>0.16168E+02</td>
<td>0.20749E+02</td>
<td>0.20510E+02</td>
<td>0.15769E+02</td>
<td>0.13000E+02</td>
</tr>
<tr>
<td>5.0</td>
<td>0.21000E+02</td>
<td>0.24893E+02</td>
<td>0.30690E+02</td>
<td>0.30491E+02</td>
<td>0.24560E+02</td>
<td>0.21000E+02</td>
</tr>
<tr>
<td>6.0</td>
<td>0.31000E+02</td>
<td>0.35622E+02</td>
<td>0.42629E+02</td>
<td>0.42459E+02</td>
<td>0.35336E+02</td>
<td>0.31000E+02</td>
</tr>
<tr>
<td>7.0</td>
<td>0.43000E+02</td>
<td>0.48351E+02</td>
<td>0.56567E+02</td>
<td>0.56418E+02</td>
<td>0.48101E+02</td>
<td>0.43000E+02</td>
</tr>
<tr>
<td>8.0</td>
<td>0.57000E+02</td>
<td>0.63083E+02</td>
<td>0.72504E+02</td>
<td>0.72372E+02</td>
<td>0.62860E+02</td>
<td>0.57000E+02</td>
</tr>
<tr>
<td>9.0</td>
<td>0.73000E+02</td>
<td>0.79815E+02</td>
<td>0.90441E+02</td>
<td>0.90322E+02</td>
<td>0.79614E+02</td>
<td>0.73000E+02</td>
</tr>
<tr>
<td>10.0</td>
<td>0.91000E+02</td>
<td>0.98548E+02</td>
<td>0.11038E+03</td>
<td>0.11027E+03</td>
<td>0.98365E+02</td>
<td>0.91000E+02</td>
</tr>
</tbody>
</table>

Computing Time = 8039153 \mu-sec
Error Norm = 0.36037E+01
ifail = 0
Table 7.10
Solution by using NAG routine D03PAF
(method of lines)
tol = 0.100E-06, dx = 0.1π (21 points)

<table>
<thead>
<tr>
<th></th>
<th>0.0</th>
<th>0.4π</th>
<th>0.8π</th>
<th>1.2π</th>
<th>1.6π</th>
<th>2.0π</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.10000E+01</td>
<td>0.19511E+01</td>
<td>0.15878E+01</td>
<td>0.41221E+00</td>
<td>0.48943E-01</td>
<td>0.10000E+01</td>
</tr>
<tr>
<td>1.0</td>
<td>0.10000E+01</td>
<td>0.21747E+01</td>
<td>0.31134E+01</td>
<td>0.25214E+01</td>
<td>0.12142E+01</td>
<td>0.10000E+01</td>
</tr>
<tr>
<td>2.0</td>
<td>0.30000E+01</td>
<td>0.47174E+01</td>
<td>0.68433E+01</td>
<td>0.64487E+01</td>
<td>0.40758E+01</td>
<td>0.30000E+01</td>
</tr>
<tr>
<td>3.0</td>
<td>0.70000E+01</td>
<td>0.93455E+01</td>
<td>0.12635E+02</td>
<td>0.12340E+02</td>
<td>0.88643E+01</td>
<td>0.70000E+01</td>
</tr>
<tr>
<td>4.0</td>
<td>0.13000E+02</td>
<td>0.16006E+02</td>
<td>0.20452E+02</td>
<td>0.20216E+02</td>
<td>0.15621E+02</td>
<td>0.13000E+02</td>
</tr>
<tr>
<td>5.0</td>
<td>0.21000E+02</td>
<td>0.24681E+02</td>
<td>0.30278E+02</td>
<td>0.30081E+02</td>
<td>0.24361E+02</td>
<td>0.21000E+02</td>
</tr>
<tr>
<td>6.0</td>
<td>0.31000E+02</td>
<td>0.33648E+02</td>
<td>0.42108E+02</td>
<td>0.41939E+02</td>
<td>0.35089E+02</td>
<td>0.31000E+02</td>
</tr>
<tr>
<td>7.0</td>
<td>0.43000E+02</td>
<td>0.48052E+02</td>
<td>0.55940E+02</td>
<td>0.55792E+02</td>
<td>0.47811E+02</td>
<td>0.43000E+02</td>
</tr>
<tr>
<td>8.0</td>
<td>0.57000E+02</td>
<td>0.62743E+02</td>
<td>0.71774E+02</td>
<td>0.71642E+02</td>
<td>0.62529E+02</td>
<td>0.57000E+02</td>
</tr>
<tr>
<td>9.0</td>
<td>0.73000E+02</td>
<td>0.79436E+02</td>
<td>0.89609E+02</td>
<td>0.89491E+02</td>
<td>0.79244E+02</td>
<td>0.73000E+02</td>
</tr>
<tr>
<td>10.0</td>
<td>0.91000E+02</td>
<td>0.98131E+02</td>
<td>1.0944E+03</td>
<td>0.10934E+03</td>
<td>0.97956E+02</td>
<td>0.91000E+02</td>
</tr>
</tbody>
</table>

Computing Time = 16532952 μ-sec
Error Norm = 0.88108E+00
ifail = 0
Table 7.11
Solution by using Fourier series technique
tol = 0.100E-01

<table>
<thead>
<tr>
<th>x</th>
<th>0.0 π</th>
<th>0.4 π</th>
<th>0.8 π</th>
<th>1.2 π</th>
<th>1.6 π</th>
<th>2.0 π</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.1000E+01</td>
<td>0.19511E+01</td>
<td>0.15878E+01</td>
<td>0.41221E+00</td>
<td>0.48943E-01</td>
<td>0.10000E+01</td>
</tr>
<tr>
<td>1.0</td>
<td>0.10006E+01</td>
<td>0.21637E+01</td>
<td>0.31027E+01</td>
<td>0.25166E+01</td>
<td>0.12175E+01</td>
<td>0.10006E+01</td>
</tr>
<tr>
<td>2.0</td>
<td>0.30006E+01</td>
<td>0.46964E+01</td>
<td>0.68126E+01</td>
<td>0.64247E+01</td>
<td>0.40687E+01</td>
<td>0.30006E+01</td>
</tr>
<tr>
<td>3.0</td>
<td>0.70002E+01</td>
<td>0.92969E+01</td>
<td>0.12566E+02</td>
<td>0.12289E+02</td>
<td>0.88493E+01</td>
<td>0.70002E+01</td>
</tr>
<tr>
<td>4.0</td>
<td>0.13001E+02</td>
<td>0.15957E+02</td>
<td>0.20356E+02</td>
<td>0.20118E+02</td>
<td>0.15572E+02</td>
<td>0.13001E+02</td>
</tr>
<tr>
<td>5.0</td>
<td>0.21005E+02</td>
<td>0.24779E+02</td>
<td>0.30248E+02</td>
<td>0.29853E+02</td>
<td>0.24141E+02</td>
<td>0.21005E+02</td>
</tr>
<tr>
<td>6.0</td>
<td>0.31003E+02</td>
<td>0.35570E+02</td>
<td>0.42117E+02</td>
<td>0.41597E+02</td>
<td>0.34729E+02</td>
<td>0.31003E+02</td>
</tr>
<tr>
<td>7.0</td>
<td>0.42979E+02</td>
<td>0.47886E+02</td>
<td>0.55686E+02</td>
<td>0.55599E+02</td>
<td>0.47746E+02</td>
<td>0.42979E+02</td>
</tr>
<tr>
<td>8.0</td>
<td>0.56953E+02</td>
<td>0.62319E+02</td>
<td>0.71325E+02</td>
<td>0.71524E+02</td>
<td>0.62642E+02</td>
<td>0.56953E+02</td>
</tr>
<tr>
<td>9.0</td>
<td>0.72950E+02</td>
<td>0.79441E+02</td>
<td>0.89399E+02</td>
<td>0.89062E+02</td>
<td>0.78896E+02</td>
<td>0.72950E+02</td>
</tr>
<tr>
<td>10.0</td>
<td>0.90927E+02</td>
<td>0.98137E+02</td>
<td>0.10920E+03</td>
<td>0.10883E+03</td>
<td>0.97536E+02</td>
<td>0.90927E+02</td>
</tr>
</tbody>
</table>

Computing Time = 154595 μ-sec
Error Norm = 0.88130E-00
ifail = 0
7.3 THE FOURIER SERIES SOLUTION OF THE NONLINEAR ADVECTION PROBLEM

We shall now consider the application of the Fourier series method to the non-linear advection problem of the form,

\[ \frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = f(x,t) \]  

(7.3.1)

with properties:

1. \( f(x,t) \) is representable in the form of

\[ f(x,t) = \sum_{r=1}^{N} \left\{ p_r(t)\cos\left(\frac{2\pi rx}{L}\right) + q_r(t)\sin\left(\frac{2\pi rx}{L}\right) \right\} \]  

(7.3.2)

2. At \( t=0 \), \( u(x,t) \) is representable as

\[ u(x,0) = \sum_{r=1}^{M} \left\{ k_r\cos\left(\frac{2\pi rx}{L}\right) + m_r\sin\left(\frac{2\pi rx}{L}\right) \right\} \]  

(7.3.3)

where \( k_r, m_r \) are constants.

3. \( u(x,t) \) is periodic, of period \( L \), in the space variable.

This last condition implies that we need only determine the solution for \( 0 \leq x \leq L, \ t > 0 \).

Reduction to ODE Form

We assume a solution to (7.3.1) of the form

\[ u(x,t) = \sum_{r=1}^{\infty} \left\{ f_r(t)\cos\left(\frac{2\pi rx}{L}\right) + g_r(t)\sin\left(\frac{2\pi rx}{L}\right) \right\} \]  

(7.3.4)

This will satisfy the initial conditions (7.3.3) provided that

\[ \begin{align*}
    f_0(0) &= k_0 \\
    f_r(0) &= k_r & r &= 1, 2, \ldots, M \\
    g_r(0) &= m_r \\
    f_r(0) &= 0 & r &= M \\
    g_r(0) &= 0 & r &= M 
\end{align*} \]  

(7.3.5)
By differentiating (7.3.4) both w.r.t. t and x we have

\[ \frac{\partial u}{\partial t} = \int f_0(t) + \sum_{r=1}^{\infty} \left\{ f_r'(t) \cos\left(\frac{2\pi r x}{L}\right) + g_r'(t) \sin\left(\frac{2\pi r x}{L}\right) \right\} \]

\[ \frac{\partial u}{\partial x} = \sum_{r=1}^{\infty} \left\{ \left(\frac{2\pi r}{L}\right) f_r(t) \sin\left(\frac{2\pi r x}{L}\right) + \left(\frac{2\pi r}{L}\right) g_r(t) \cos\left(\frac{2\pi r x}{L}\right) \right\} . \]

Let the product \( (u)(\frac{\partial u}{\partial x}) \) be written in Taylor series form as

\[ u(x) = a_0 + \sum_{r=1}^{\infty} \left( a_r \cos\left(\frac{2\pi r x}{L}\right) + b_r \sin\left(\frac{2\pi r x}{L}\right) \right) \]

where the \( a_r \) and \( b_r \) are related to \( f_r \) and \( g_r \) in some form. In order to obtain this relation we use the technique developed in Section 7.2.

First, we define the following operators:

- \( c(r) = \cos\left(\frac{2\pi r x}{L}\right) \)
- \( s(r) = \sin\left(\frac{2\pi r x}{L}\right) \)
- \( f(r) = f_r(t) \)
- \( g(r) = g_r(t) \)
- \( z(r) = \frac{2\pi r}{L} \).

Then \( u \) and \( u_x \) can be written as

\[ u = \int f_0 + \sum_{r=1}^{\infty} \{ f(r)c(r) + g(r)s(r) \} \]

\[ u_x = \sum_{r=1}^{\infty} \{ -z(r)f(r)s(r) + z(r)g(r)c(r) \} . \]

Multiplying we have,

\[ (u)(u_x) = [\int f_0 + \sum_{r=1}^{\infty} \{ f(r)c(r) + g(r)s(r) \}] \sum_{r=1}^{\infty} \{ -z(r)f(r)s(r) + z(r)g(r)c(r) \} \]

\[ = a_0 + \sum_{r=1}^{\infty} \{ a_r \cos\left(\frac{2\pi r x}{L}\right) + b_r \sin\left(\frac{2\pi r x}{L}\right) \} . \]

This multiplication is carried out by using the REDUCE symbolic computation package and we obtain the final result in the form

\[ \int a_0 + \sum_{r=1}^{\infty} \{ a_r \cos\left(\frac{2\pi r x}{L}\right) + b_r \sin\left(\frac{2\pi r x}{L}\right) \} . \]

An example of the result by using \( f_r \) and \( g_r \) for \( r \) up to 4 is given as follows:
\[ a_0 = 0 \]
\[ a_1 = \frac{1}{2}(f_0 g_1 + f_1 g_2 - f_2 g_1 + f_2 g_3 - f_3 g_2 + f_3 g_4 - f_4 g_3) \]
\[ a_2 = \frac{1}{2}(f_0 g_2 + 2f_1 g_1 + 2f_1 g_3 + f_2 g_4 - 2f_3 g_1 - 2f_4 g_2) \]
\[ a_3 = \frac{1}{2}(f_0 g_3 + 3f_1 g_2 + 3f_1 g_4 + 3f_2 g_1 - 3f_4 g_1) \]
\[ a_4 = \frac{1}{2}(f_0 g_4 + 4f_1 g_3 + 4f_2 g_2 + 4f_3 g_1) \]
\[ a_5 = \frac{1}{2}(f_1 g_4 + 5f_2 g_3 + 5f_3 g_2 + 5f_4 g_1) \]

By substituting the suitable expressions into (7.3.1) we have

\[
\begin{align*}
\mathbf{f}_0(t) + \sum_{r=1}^{\infty} \{ f_r(t) \cos(\frac{2\pi r x}{L}) + g_r(t) \sin(\frac{2\pi r x}{L}) \} \\
+ \mathbf{a}_0(t) + \sum_{r=1}^{\infty} \{ a_r(t) \cos(\frac{2\pi r x}{L}) + b_r(t) \sin(\frac{2\pi r x}{L}) \} \\
= \mathbf{p}_0(t) + \sum_{r=1}^{N} \{ p_r(t) \cos(\frac{2\pi r x}{L}) + q_r(t) \sin(\frac{2\pi r x}{L}) \} \\
\end{align*}
\]

(7.3.7)

Equating coefficients we have,

\[
\begin{align*}
\mathbf{f}_0(t) + \mathbf{a}_0(t) = \mathbf{p}_0(t) \\
f_r'(t) + a_r(t) = p_r(t) \\
g_r'(t) + b_r(t) = q_r(t) \\
\end{align*}
\]

(7.3.8)
Unlike the strategy employed by Forrington [1963], this set of o.d.e. is not sufficient to represent the p.d.e. This is because the \(a_r\) and \(b_r\) are also functions of \(f_i\) and \(g_i\) with \(i<r\). But for \(r>2M\), the quantities \(a_r\) and \(b_r\) must contain \(f_i\) and \(g_i\) with \(i>M\) as factors. Therefore, considering the initial conditions are all zero for \(r>M\), the following additional equations will be sufficient to constitute the system of o.d.e's to represent (7.3.1). These are

\[
\begin{align*}
  f_r'(t) + a_r(t) &= 0 \\
  g_r'(t) + b_r(t) &= 0
\end{align*}
\]

For \(r>2M\) the differential equations are of the form,

\[
\begin{align*}
  \frac{df_r}{dt} &= 0, \quad f_r(0)=0 \\
  \frac{dg_r}{dt} &= 0, \quad g_r(0)=0
\end{align*}
\]

whose solutions are \(f_r(t) = 0\), \(g_r(t) = 0\).

Therefore, the original p.d.e. is transformed into a finite number of o.d.e's (7.3.8) and (7.3.9) with initial conditions (7.3.5).

**Numerical Example**

We solve the problem,

\[
\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = \cos(x+t) + \frac{1}{2}\sin(2(x+t))
\]

(7.3.10)

with initial conditions \(u(x,0) = \sin x\).

The exact solution of this problem is given by \(u(x,t) = \sin(x+t)\).

Note that in this example the RHS function of (7.3.1) is provided by

\[
f(x,t) = \cos(x+t) + \frac{1}{2}\sin(2(x+t))
\]

\[
= \cos x \cos t - \sin x \sin t + \frac{1}{2}(\sin 2x \cos 2t + \cos 2x \sin 2t)
\]

\[
= \sum_{r=1}^{\infty} \{ p_r \cos rx + q_r \sin rx \}
\]

(7.3.11)
where
\[
\begin{align*}
p_0 &= 0, \\
p_1 &= \cos t, \quad q_1 = -\sin t, \\
p_2 &= \frac{1}{2}\sin 2t, \quad q_2 = \frac{1}{2}\cos 2t, \\
\text{remainder} &= 0, \text{i.e. } N = 2.
\end{align*}
\]

At \( t=0, \) \( u(x,0) = \sin x. \) Therefore, \( k_0 = 0, \ k_1 = 0, \ k_2 = 0, \ldots \) and \( m_1 = 1, \ m_2 = 0, \ldots, \) i.e. \( M = 1. \) It follows that \( \gamma = \max(M,N) = \max(1,2) = 2 \) and \( 2\gamma = 2. \) Therefore the set of o.d.e's which represent the original problem is given by

\[
\begin{align*}
f_0'(t) + a_0(t) &= p_0(t), \quad f_0(0) = 0 \\
f_1'(t) + a_1(t) &= p_1(t), \quad f_1(0) = 0 \\
g_1'(t) + b_1(t) &= q_1(t), \quad g_1(0) = 1 \\
f_2'(t) + a_2(t) &= p_2(t), \quad f_2(0) = 0 \\
g_2'(t) + b_2(t) &= q_2(t), \quad g_2(0) = 0
\end{align*}
\]

where
\[
\begin{align*}
a_0 &= 0, \\
a_1 &= \frac{i}{2}(f_0g_1 + f_1g_2 - f_2g_1), \\
a_2 &= f_0g_2 + f_1g_1, \\
b_1 &= \frac{i}{2}(-f_0f_1 - f_1f_2 + g_1g_2) \\
b_2 &= \frac{i}{2}(-2f_0f_1 + f_1^2 + g_1^2).
\end{align*}
\]

Since \( f_0(t) = 0 \) from the first equation, the set of o.d.e's reduces to

\[
\begin{align*}
y_1' &= \cos t - a_1, \quad y_1(0) = 0, \\
y_2' &= -\sin t - b_1, \quad y_2(0) = 1 \\
y_3' &= \frac{1}{2}\sin 2t - a_2, \quad y_3(0) = 0 \\
y_4' &= \frac{1}{2}\cos 2t - b_2, \quad y_4(0) = 0
\end{align*}
\]
where

\[ \begin{align*}
    a_1 &= \frac{1}{2}(y_1 y_4 - y_3 y_2) \\
    a_2 &= y_1 y_2 \\
    b_1 &= \frac{1}{2}(-y_1 y_3 - y_2 y_4) \\
    b_2 &= \frac{1}{2}(-y_1^2 + y_2^2)
\end{align*} \]

and the independent variables \(y_1, y_2, y_3, y_4\) are used in place of \(f_1, f_2, g_1, g_2\) respectively.

System (7.3.13) is then solved by using the NAG routine D02BBF (Runge-Kutta Merson method) within the domain \(0 \leq x \leq 2\pi, 0 \leq t \leq 10\). The errors in the solutions are shown in Table 7.12. It is noticed that these errors lie within the accuracy requirement, \(\text{tol} = 10^{-7}\). This confirms that the errors are only due to the discretisation in the \(t\) direction and that it is within our control. The whole computation for \(0 \leq t \leq 10.0\) takes approximately 0.8 sec.

### Table 7.12

Errors in the numerical solution of the nonlinear advection problem

<table>
<thead>
<tr>
<th>(x)</th>
<th>0.0</th>
<th>0.5\pi</th>
<th>\pi</th>
<th>1.5\pi</th>
<th>2\pi</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
</tr>
<tr>
<td>1.0</td>
<td>-0.63744E-07</td>
<td>0.36855E-07</td>
<td>-0.38206E-07</td>
<td>0.65095E-07</td>
<td>-0.3744E-07</td>
</tr>
<tr>
<td>2.0</td>
<td>-0.46184E-07</td>
<td>0.10912E-07</td>
<td>-0.34426E-07</td>
<td>0.69698E-07</td>
<td>-0.46184E-07</td>
</tr>
<tr>
<td>3.0</td>
<td>0.45528E-07</td>
<td>-0.65725E-07</td>
<td>0.44726E-07</td>
<td>-0.24529E-07</td>
<td>0.45528E-07</td>
</tr>
<tr>
<td>4.0</td>
<td>-0.19816E-07</td>
<td>0.80702E-08</td>
<td>0.73192E-08</td>
<td>0.44271E-08</td>
<td>-0.19816E-07</td>
</tr>
<tr>
<td>5.0</td>
<td>-0.67706E-07</td>
<td>0.46209E-07</td>
<td>-0.17294E-08</td>
<td>0.23226E-07</td>
<td>-0.67706E-07</td>
</tr>
<tr>
<td>6.0</td>
<td>0.37288E-07</td>
<td>-0.61191E-07</td>
<td>0.96530E-07</td>
<td>-0.72628E-07</td>
<td>0.37288E-07</td>
</tr>
<tr>
<td>7.0</td>
<td>-0.49582E-07</td>
<td>-0.47741E-07</td>
<td>0.58700E-07</td>
<td>-0.60541E-07</td>
<td>0.49582E-07</td>
</tr>
<tr>
<td>8.0</td>
<td>0.56140E-08</td>
<td>0.25797E-07</td>
<td>-0.18287E-07</td>
<td>-0.13124E-07</td>
<td>0.56140E-08</td>
</tr>
<tr>
<td>9.0</td>
<td>0.51461E-07</td>
<td>-0.22000E-07</td>
<td>0.44861E-07</td>
<td>-0.74322E-07</td>
<td>0.51461E-07</td>
</tr>
<tr>
<td>10.0</td>
<td>0.28262E-07</td>
<td>-0.22647E-07</td>
<td>0.44694E-07</td>
<td>-0.50309E-07</td>
<td>0.28262E-07</td>
</tr>
</tbody>
</table>

Computing Time = 885365 \(\mu\)-sec

ifail = 0
7.4 FURTHER APPLICATION OF FOURIER SERIES METHOD

7.4.1 Fourier Series Solution of the First Order Hyperbolic Systems in One Space Dimension

We will now consider the first-order system of equations

\[ \frac{\partial u}{\partial t} + A \frac{\partial u}{\partial x} = h(x,t) \] (7.4.1)

where \( A \) is an \( nxn \) real matrix, and \( u \) is an \( n \)-component column vector. Initially \( A \) is assumed to be constant. We consider the case where \( A \) has all real eigenvalues and \( n \) linearly independent eigenvectors, so that the system is hyperbolic. \( A \) is not necessarily a symmetric matrix. (See Mitchell and Griffiths [1980], p.172).

We will now apply the Fourier series technique discussed in the preceding section for solving the parabolic p.d.e. For this purpose we set \( n=1 \) to concentrate on the method and not on the difficult computations. Thus the class of problems considered here is

\[ \frac{\partial u}{\partial t} + a \frac{\partial u}{\partial x} = h(x,t) \] (7.4.2)

with properties

1. \( a \) is independent of \( x \)
2. \( h \) is representable in the form
   \[ h(x,t) = \hat{p}_0(t) + \sum_{r=1}^{N} \{ p_r(t) \cos(\frac{2\pi rx}{L}) + q_r(t) \sin(\frac{2\pi rx}{L}) \} \] (7.4.3)
3. At \( t=0 \), \( u(x,t) \) is representable as
   \[ u(x,0) = \hat{k}_0 + \sum_{r=1}^{M} \{ k_r \cos(\frac{2\pi rx}{L}) + m_r \sin(\frac{2\pi rx}{L}) \} \] (7.4.4)
4. \( u(x,t) \) is periodic, of period \( L \), in the space variable.
Reduction to o.d.e. form

We assume that a solution exists in the form

\[ u(x,t) = \xi f_0(t) + \sum_{r=1}^{\infty} \left\{ f_r(t) \cos \left( \frac{2\pi rx}{L} \right) + g_r(t) \sin \left( \frac{2\pi r x}{L} \right) \right\} \]  \hspace{1cm} (7.4.5)

This solution will satisfy the initial conditions if

\[
\begin{align*}
    f_0(0) &= k_0 \\
    f_r(0) &= k_r & r &= 1, 2, \ldots, M \\
    g_r(0) &= m_r & r &= 1, 2, \ldots, M \\
    f_r(0) &= 0 & r &> M \\
    g_r(0) &= 0 \\
\end{align*}
\]  \hspace{1cm} (7.4.6)

Substituting Eqs. (7.4.5) and (7.4.3) into (7.4.2) we obtain

\[
\begin{align*}
    \frac{\partial^2}{\partial x^2} \left( \xi f_0(t) \right) + \sum_{r=1}^{\infty} \left\{ \frac{2\pi r}{L} f_r(t) \cos \left( \frac{2\pi r x}{L} \right) + \frac{2\pi r}{L} g_r(t) \sin \left( \frac{2\pi r x}{L} \right) \right\} \\
    + a \sum_{r=1}^{\infty} \left\{ \frac{2\pi r}{L} f_r(t) \sin \left( \frac{2\pi r x}{L} \right) + \frac{2\pi r}{L} g_r(t) \cos \left( \frac{2\pi r x}{L} \right) \right\} \\
    = \xi p_0(t) + \sum_{r=1}^{N} \left\{ p_r(t) \cos \left( \frac{2\pi r x}{L} \right) + q_r(t) \sin \left( \frac{2\pi r x}{L} \right) \right\}
\end{align*}
\]  \hspace{1cm} (7.4.7)

By collecting coefficients of like terms we obtain the relations,

\[
\begin{align*}
    \text{Independent Term:} & \quad \xi f'_0(t) = \xi p_0(t) \\
    \cos \left( \frac{2\pi rx}{L} \right): & \quad f'_r(t) + \frac{2\pi ra}{L} g_r(t) = p_r(t) \quad r = 1, 2, \ldots, N \\
    \sin \left( \frac{2\pi rx}{L} \right): & \quad g'_r(t) - \frac{2\pi ra}{L} f_r(t) = q_r(t) \quad r = 1, 2, \ldots, N
\end{align*}
\]  \hspace{1cm} (7.4.8)

Therefore we have transformed the p.d.e. (7.4.2) into a system of o.d.e's (7.4.8) with initial conditions given by (7.4.6).

Numerical Example

We solve the hyperbolic equation

\[
\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = \cos(x+t) \]  \hspace{1cm} (7.4.9)

with \( u(x,0) = \sin x \), \( u(0,t) = u(2\pi,t) \).
The exact solution of this problem is given by
\[ u(x,t) = \sin x \cos t. \]

In this example we have
\[ a = 1, \]
\[ h(x,t) = \cos(x+t) \]
\[ = \cos x \cos t - \sin x \sin t \]
\[ = p_1(t) \cos x + q_1(t) \sin x \]
where
\[ p_1(t) = \cos t, \]
\[ q_1(t) = -\sin t, \]
\[ (\text{other } p_i, q_i \text{ are zero}) \]
and
\[ u(x,0) = \sin x \]
\[ = k_0 + k_1(t) \cos x + m_1(t) \sin x \]
where \[ m_1(t) = 1, \] (other \( k_i, m_i \) are zero).

The system of o.d.e's (7.4.7) now become, \((L=2n)\)
\[
\begin{align*}
\frac{df_0}{dt} &= 0 \quad f_0(0) = 0 \\
\frac{df_1}{dt} &= g_1(t) + \cos t \quad f_1(0) = 0 \\
\frac{dg_1}{dt} &= f_1(t) - \sin t, \quad g_1(0) = 1
\end{align*}
\]
(7.4.10)

We use the NAG routine D02BBF (Runge-Kutta Merson method) to solve this system with \( TOL=10^{-7} \). The results obtained are correct to at least to seven decimal places as required in \( TOL \). The errors in the solutions are shown in Table 7.13 for \( t=0.0(1.0)10.0 \). Thus the Fourier series method is applicable for use in solving the hyperbolic equation of the form
\[
\frac{\partial u}{\partial x} + \frac{\partial u}{\partial t} = f(x,t). \]
Table 7.13
Errors in the solutions
tol = 0.100E-06, dx = 0.5π

<table>
<thead>
<tr>
<th>t</th>
<th>x = 0.0</th>
<th>x = 0.5π</th>
<th>x = π</th>
<th>x = 1.5π</th>
<th>x = 2.0π</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.000000E+00</td>
<td>0.000000E+00</td>
<td>0.000000E+00</td>
<td>0.000000E+00</td>
<td>0.000000E+00</td>
</tr>
<tr>
<td>1.0</td>
<td>-0.01657E-07</td>
<td>-0.27664E-07</td>
<td>0.61657E-07</td>
<td>0.27664E-07</td>
<td>-0.61657E-07</td>
</tr>
<tr>
<td>2.0</td>
<td>-0.20114E-07</td>
<td>-0.63248E-07</td>
<td>0.20114E-07</td>
<td>0.63248E-07</td>
<td>-0.61657E-07</td>
</tr>
<tr>
<td>3.0</td>
<td>0.97582E-07</td>
<td>-0.11000E-07</td>
<td>0.97582E-07</td>
<td>0.11000E-07</td>
<td>0.97582E-07</td>
</tr>
<tr>
<td>4.0</td>
<td>0.12505E-06</td>
<td>0.10682E-06</td>
<td>-0.12505E-06</td>
<td>-0.10682E-06</td>
<td>0.12505E-06</td>
</tr>
<tr>
<td>5.0</td>
<td>-0.24798E-08</td>
<td>0.16539E-06</td>
<td>-0.24798E-08</td>
<td>0.16539E-06</td>
<td>-0.24798E-08</td>
</tr>
<tr>
<td>6.0</td>
<td>-0.18890E-06</td>
<td>0.51319E-07</td>
<td>-0.18890E-06</td>
<td>0.51319E-07</td>
<td>-0.18890E-06</td>
</tr>
<tr>
<td>7.0</td>
<td>-0.20823E-06</td>
<td>-0.16574E-06</td>
<td>-0.20823E-06</td>
<td>0.16574E-06</td>
<td>-0.20823E-06</td>
</tr>
<tr>
<td>8.0</td>
<td>-0.32987E-08</td>
<td>0.27407E-06</td>
<td>-0.32987E-08</td>
<td>0.27407E-06</td>
<td>-0.32987E-08</td>
</tr>
<tr>
<td>9.0</td>
<td>0.26935E-06</td>
<td>-0.11944E-06</td>
<td>0.26935E-06</td>
<td>-0.11944E-06</td>
<td>0.26935E-06</td>
</tr>
<tr>
<td>10.0</td>
<td>0.30731E-06</td>
<td>0.19961E-06</td>
<td>0.30731E-06</td>
<td>-0.19961E-06</td>
<td>0.30731E-06</td>
</tr>
</tbody>
</table>

Computing Time = 620673 µ-sec
ifail = 0

7.4.2 Linearised Burgers Equation

Let us now consider the problem of solving the linearised Burgers equation given by

\[
\frac{\partial u}{\partial t} + c \frac{\partial u}{\partial x} = \eta \frac{\partial^2 u}{\partial x^2} \tag{7.4.11a}
\]

\[
u(x,0) = \sin(kx) \tag{7.4.11b}
\]

\[
u(x,t) = u(x + \frac{2\pi}{k}, t) \tag{7.4.11c}
\]

As before, we will apply the Fourier series technique to solve this problem. By substituting the Fourier series form of the solution into (7.4.11) we obtain

\[
\hat{f}_0(t) + \sum_{r=1}^{\infty} \left[ f_r(t) \cos\left(\frac{2\pi rx}{L}\right) + g_r(t) \sin\left(\frac{2\pi rx}{L}\right) \right] + c \left[ \sum_{r=1}^{\infty} \left\{ -\frac{2\pi r}{L} f_r(t) \sin\left(\frac{2\pi rx}{L}\right) + \frac{2\pi r}{L} g_r(t) \cos\left(\frac{2\pi rx}{L}\right) \right\} \right]
\]

\[
= \eta \left\{ \sum_{r=1}^{\infty} \left[ -\frac{2\pi r}{L} f_r(t) \cos\left(\frac{2\pi rx}{L}\right) - \frac{2\pi r}{L} g_r(t) \sin\left(\frac{2\pi rx}{L}\right) \right] \right\}
\]

where \( L = 2\pi/k \).

As before, collecting the coefficients of similar terms we obtain,
Independent Term: \( f'_0(t) = 0 \),

\[
\begin{align*}
\cos(\frac{2\pi r}{L}) & : 
 f'_r(t) + c(\frac{2\pi r}{L}) g_r(t) = -\eta(\frac{2\pi}{L})^2 f_r(t) \\
\sin(\frac{2\pi r}{L}) & : 
 g'_r(t) - c(\frac{2\pi r}{L}) f_r(t) = -\eta(\frac{2\pi}{L})^2 g_r(t)
\end{align*}
\]

In our example,

\[ u(x,0) = \sin kx \]

\[ = \frac{1}{2} k_0 + k_1 \cos k_0 x + m_1 \sin k_0 x, \]

where, by virtue of (7.4.11b), \( k_0 = k_1 = 0 \) and \( m_1 = 1 \). Therefore, we have reduced (7.4.11a) into the following system of o.d.e's:

\[
\begin{align*}
f'_0(t) &= 0, \\
f'_1(t) &= -ckg_1(t) - nk^2 f_1(t), \\
g'_1(t) &= ckf_1(t) - nk^2 f_1(t)
\end{align*}
\]

(7.4.12)

We solve this system by using the NAG routine D02BBF (Runge-Kutta Merson method) and use different combinations of \( k, c, \) and \( \eta \). In each case we use \( TOL=10^{-7} \) and the results from our experiments show that they are accurate to at least the seventh decimal place, except when the value of \( c \) is large relative to \( \eta \). For example, with \( k=\frac{1}{2}, c=10, \eta = 1 \) the results are accurate only to six decimal places. Two sets of results are given here. Table 4.14 gives the errors in the solution when \( k=2, c=2, \eta=2 \), while Table 7.15 gives the corresponding errors in the solution when \( k=\frac{1}{2}, c=10, \eta=1 \).

Thus the Fourier series method is also suitable for use in solving the linearised Burgers equation.
Table 7.14
Errors in the solution of the linearised Burgers equation
\( k=2.0, \ c=2.0, \ \eta=2.0 \)
tol = 0.100E-06, dx=0.2\pi

<table>
<thead>
<tr>
<th>( x )</th>
<th>0.0</th>
<th>0.25\pi</th>
<th>0.5\pi</th>
<th>0.75\pi</th>
<th>\pi</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
</tr>
<tr>
<td>1.0</td>
<td>0.60325E-07</td>
<td>0.12415E-07</td>
<td>-0.60325E-07</td>
<td>-0.12415E-07</td>
<td>0.60325E-07</td>
</tr>
<tr>
<td>2.0</td>
<td>0.78815E-08</td>
<td>-0.96910E-08</td>
<td>-0.78815E-08</td>
<td>0.96910E-08</td>
<td>0.78815E-08</td>
</tr>
<tr>
<td>3.0</td>
<td>0.11416E-07</td>
<td>0.89625E-07</td>
<td>-0.11416E-07</td>
<td>-0.89625E-07</td>
<td>0.11416E-07</td>
</tr>
<tr>
<td>4.0</td>
<td>-0.24913E-07</td>
<td>-0.50134E-08</td>
<td>0.24913E-07</td>
<td>0.50134E-07</td>
<td>-0.24913E-07</td>
</tr>
<tr>
<td>5.0</td>
<td>-0.86766E-07</td>
<td>-0.14748E-07</td>
<td>0.86766E-07</td>
<td>0.14748E-07</td>
<td>-0.86766E-07</td>
</tr>
<tr>
<td>6.0</td>
<td>0.14788E-07</td>
<td>-0.30510E-07</td>
<td>0.14788E-07</td>
<td>0.30510E-07</td>
<td>-0.14788E-07</td>
</tr>
<tr>
<td>7.0</td>
<td>0.24807E-07</td>
<td>-0.73615E-07</td>
<td>0.24807E-07</td>
<td>0.73615E-07</td>
<td>-0.24807E-07</td>
</tr>
<tr>
<td>8.0</td>
<td>0.29709E-07</td>
<td>-0.19340E-07</td>
<td>0.29709E-07</td>
<td>0.19340E-07</td>
<td>-0.29709E-07</td>
</tr>
<tr>
<td>9.0</td>
<td>0.52348E-07</td>
<td>0.51796E-07</td>
<td>-0.52348E-07</td>
<td>-0.51796E-07</td>
<td>0.52348E-07</td>
</tr>
<tr>
<td>10.0</td>
<td>0.22096E-08</td>
<td>0.12699E-07</td>
<td>0.22096E-08</td>
<td>0.12699E-07</td>
<td>-0.22096E-08</td>
</tr>
</tbody>
</table>

Computing Time = 337803 \( \mu \)-sec
ifail = 0

Table 7.15
Errors in the solution of the linearised Burgers equation
\( k=0.5, \ c=10.0, \ \eta=1.0 \)
tol = 0.100E-06, dx=\pi

<table>
<thead>
<tr>
<th>( x )</th>
<th>0.0</th>
<th>( \pi )</th>
<th>2( \pi )</th>
<th>3( \pi )</th>
<th>4( \pi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
<td>0.00000E+00</td>
</tr>
<tr>
<td>1.0</td>
<td>0.79066E-06</td>
<td>0.13230E-05</td>
<td>0.79066E-06</td>
<td>-0.13230E-05</td>
<td>-0.79066E-06</td>
</tr>
<tr>
<td>2.0</td>
<td>0.18132E-05</td>
<td>0.19662E-05</td>
<td>-0.18132E-05</td>
<td>-0.19662E-05</td>
<td>0.18132E-05</td>
</tr>
<tr>
<td>3.0</td>
<td>0.31498E-05</td>
<td>-0.15523E-05</td>
<td>0.31498E-05</td>
<td>0.15523E-05</td>
<td>-0.31498E-05</td>
</tr>
<tr>
<td>4.0</td>
<td>-0.68991E-06</td>
<td>-0.40181E-05</td>
<td>0.68991E-06</td>
<td>0.40181E-05</td>
<td>-0.68991E-06</td>
</tr>
<tr>
<td>5.0</td>
<td>-0.44287E-05</td>
<td>0.52127E-06</td>
<td>0.44287E-05</td>
<td>0.52127E-06</td>
<td>-0.44287E-05</td>
</tr>
<tr>
<td>6.0</td>
<td>-0.18445E-05</td>
<td>0.43058E-05</td>
<td>0.18445E-05</td>
<td>0.43058E-05</td>
<td>-0.18445E-05</td>
</tr>
<tr>
<td>7.0</td>
<td>0.36904E-05</td>
<td>-0.30639E-05</td>
<td>0.36904E-05</td>
<td>0.30639E-05</td>
<td>-0.36904E-05</td>
</tr>
<tr>
<td>8.0</td>
<td>0.40262E-05</td>
<td>-0.26982E-05</td>
<td>0.40262E-05</td>
<td>0.26982E-05</td>
<td>-0.40262E-05</td>
</tr>
<tr>
<td>9.0</td>
<td>-0.14383E-05</td>
<td>-0.46078E-05</td>
<td>0.14383E-05</td>
<td>0.46078E-05</td>
<td>-0.14383E-05</td>
</tr>
<tr>
<td>10.0</td>
<td>-0.47605E-05</td>
<td>0.75213E-07</td>
<td>0.47605E-05</td>
<td>0.75213E-07</td>
<td>-0.47605E-05</td>
</tr>
</tbody>
</table>

Computing Time = 1172856 \( \mu \)-sec
ifail = 0
7.4.3 Numerical Solutions of the Wave Equation

We consider now the following system of p.d.e's

\[ \frac{\partial^2 u_1}{\partial x^2} + c^2 \frac{\partial^2 u_2}{\partial t^2} = 0 \]  
(7.4.13)

where

\[ u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \quad \text{and} \quad A = \begin{bmatrix} 0 & c \\ c & 0 \end{bmatrix}. \]

In expanded form this system can be written as

\[ \frac{\partial u_1}{\partial t} + c \frac{\partial u_2}{\partial x} = 0 \]  
(7.4.14a)  
\[ \frac{\partial u_2}{\partial t} + c \frac{\partial u_1}{\partial x} = 0 \]  
(7.4.14b)

Differentiating (7.4.14a) w.r.t. \(x\) and (7.4.14b) w.r.t. \(t\) we obtain

\[ \frac{\partial^2 u_1}{\partial x^2} + c^2 \frac{\partial^2 u_2}{\partial t^2} = 0 \]  
(7.4.15a)  
\[ \frac{\partial^2 u_2}{\partial t^2} + c^2 \frac{\partial^2 u_1}{\partial x^2} = 0 \]  
(7.4.15b)

Now multiplying (7.4.15a) by \(c\) and subtracting (7.4.15b) from the result we obtain

\[ c^2 \frac{\partial^2 u_2}{\partial x^2} - \frac{\partial^2 u_2}{\partial t^2} = 0 \]  
(7.4.16)

Equation (7.4.16) is known as the wave equation. Since (7.4.16) can be obtained from Eqs. (7.4.14), then solving (7.4.16) is equivalent to solving the set of p.d.e's (7.4.14).

Suppose now the initial conditions of the system can be expressed as

\[ u_1(x,0) = \sum_{r=1}^{M} k_r \frac{\cos(\frac{2\pi r x}{L}) + \sin(\frac{2\pi r x}{L})}{L} \]  
(7.4.17)  
\[ u_2(x,0) = \sum_{r=1}^{N} k_r \frac{\cos(\frac{2\pi r x}{L}) + \sin(\frac{2\pi r x}{L})}{L} \]
and that the solutions are periodic with period $L$, so that we may write

$$
\begin{align*}
  u_1(x,t) &= \sum_{r=1}^{\infty} \left[ f_r(t)\cos\left(\frac{2\pi rx}{L}\right) + g_r(t)\sin\left(\frac{2\pi rx}{L}\right) \right] \\
  u_2(x,t) &= \sum_{r=1}^{\infty} \left[ \hat{f}_r(t)\cos\left(\frac{2\pi rx}{L}\right) + \hat{g}_r(t)\sin\left(\frac{2\pi rx}{L}\right) \right]
\end{align*}
$$

(7.4.18)

Differentiating w.r.t. $t$ and $x$ we have,

$$
\begin{align*}
  \frac{\partial u_1}{\partial t} &= \sum_{r=1}^{\infty} \left[ f_r'(t)\cos\left(\frac{2\pi rx}{L}\right) + g_r'(t)\sin\left(\frac{2\pi rx}{L}\right) \right] \\
  \frac{\partial u_1}{\partial x} &= \sum_{r=1}^{\infty} \left[ -(\frac{2\pi rx}{L})f_r(t)\sin\left(\frac{2\pi rx}{L}\right) + (\frac{2\pi rx}{L})g_r(t)\cos\left(\frac{2\pi rx}{L}\right) \right]
\end{align*}
$$

(7.4.19)

and with similar expressions for $u_2$.

Now, substituting these into Eqs. (7.4.14) we obtain

$$
\begin{align*}
  \frac{1}{2}f_0'(t) + \sum_{r=1}^{\infty} \left[ f_r'(t)\cos\left(\frac{2\pi rx}{L}\right) + g_r'(t)\sin\left(\frac{2\pi rx}{L}\right) \right] \\
  + c \sum_{r=1}^{\infty} \left[ -(\frac{2\pi r}{L})f_r(t)\sin\left(\frac{2\pi rx}{L}\right) + (\frac{2\pi r}{L})g_r(t)\cos\left(\frac{2\pi rx}{L}\right) \right] &= 0 \\
  (7.4.20a)
\end{align*}
$$

$$
\begin{align*}
  \frac{1}{2}\hat{f}_0'(t) + \sum_{r=1}^{\infty} \left[ \hat{f}_r'(t)\cos\left(\frac{2\pi rx}{L}\right) + \hat{g}_r'(t)\sin\left(\frac{2\pi rx}{L}\right) \right] \\
  + c \sum_{r=1}^{\infty} \left[ -(\frac{2\pi r}{L})f_r(t)\sin\left(\frac{2\pi rx}{L}\right) + (\frac{2\pi r}{L})g_r(t)\cos\left(\frac{2\pi rx}{L}\right) \right] &= 0 \\
  (7.4.20b)
\end{align*}
$$

By equating coefficients of similar terms we obtain the following system of o.d.e's

$$
\begin{align*}
  \frac{1}{2}f_0'(t) &= 0 \\
  f_r'(t) + c(\frac{2\pi r}{L})g_r(t) &= 0 \\
  g_r'(t) - c(\frac{2\pi r}{L})f_r(t) &= 0 \\
  \hat{f}_0'(t) &= 0 \\
  \hat{f}_r'(t) + c(\frac{2\pi r}{L})g_r(t) &= 0 \\
  \hat{g}_r'(t) - c(\frac{2\pi r}{L})f_r(t) &= 0.
\end{align*}
$$

(7.4.21)
and from (7.4.17) and (7.4.18), the initial conditions are given by

\[
\begin{align*}
    f_0(0) &= k_0 \\
    f_r(0) &= k_r & (r=1,2,\ldots,M) \\
    g_r(0) &= m_r \\
    f_r(0) &= g_r(0) = 0, \ r>M \\
    \hat{f}_0(0) &= 0 \\
    \hat{f}_r(0) &= \hat{k}_r & (r=1,2,\ldots,N) \\
    \hat{g}_r(0) &= \hat{m}_r \\
    \hat{f}_r(0) &= \hat{g}_r(0) = 0, \ r>N.
\end{align*}
\]  

Therefore the initial problem of solving (7.4.14) with initial conditions (7.4.17) is transformed into solving (7.4.21) with initial conditions (7.4.22) and with subscript \( r \) to run from 1 to \( \gamma = \max(M,N) \). Unlike in the nonlinear case, this transformation is straightforward. The number of equations in (7.4.21) is dependent only on \( M \) and \( N \).

**Numerical Example**

We solve the wave equation

\[
\begin{align*}
    \frac{\partial u_1}{\partial t} + \frac{\partial u_2}{\partial x} &= 0 \\
    \frac{\partial u_2}{\partial t} + \frac{\partial u_1}{\partial x} &= 0 \quad \text{(i.e. } c = 1) 
\end{align*}
\]  

(7.4.23)

with the initial conditions

\[
\begin{align*}
    u_1(x,0) &= 2 - \sin x + \cos x \\
    u_2(x,0) &= 3 - \sin x + \cos x.
\end{align*}
\]

The exact solution is given by

\[
\begin{align*}
    u_1(x,t) &= 2 + \sin(t-x) + \cos(t-x) \\
    u_2(x,t) &= 3 + \sin(t-x) + \cos(t-x).
\end{align*}
\]

In this example we have,
\[ \hat{k}_0 = 2, \hat{k}_1 = 1, \hat{m}_1 = -1, \text{ other } k_i, m_i \text{ are zero, i.e. } M=1; \]
\[ \hat{k}_0 = 3, \hat{k}_1 = 1, \hat{m}_1 = -1, \text{ other } \hat{k}_i, \hat{m}_i \text{ are zero, i.e. } N=1; \]

Therefore, the corresponding set of o.d.e's for this problem is

\begin{align*}
    f_0'(t) &= 0, \quad f_0(0)=4 \quad (7.4.24a) \\
    f_1'(t)+g_1(t) &= 0, \quad f_1(0)=1 \quad (7.4.24b) \\
    g_1'(t)-f_1(t) &= 0, \quad g_1(0)=-1 \quad (7.4.24c) \\
    \hat{f}_0'(t) &= 0, \quad \hat{f}_0(0)=6 \quad (7.4.24d) \\
    \hat{f}_1'(t)+g_1(t) &= 0, \quad \hat{f}_1(0)=1 \quad (7.4.24e) \\
    \hat{g}_1'(t)-f_1(t) &= 0, \quad \hat{g}_1(0)=-1 \quad (7.4.24f)
\end{align*}

We solve system (7.4.24) by using NAG routine D02BBF for \( 0 \leq t \leq 10 \) and evaluate the solution of (7.4.23) from the relations,

\[
    u_1 = f_0(t)+f_1(t)\cos x+g_1(t)\sin x
\]
\[
    u_2 = \hat{f}_0(t)+\hat{f}_1(t)\cos x+\hat{g}_1(t)\sin x.
\]

The errors in the results are shown in Table 7.16 where we have solved the problem in the domain \( 0 \leq x \leq 2\pi \), \( 0 \leq t \leq 10 \) with the accuracy requirement given by \( \text{tol}=10^{-7} \). Although the errors shown are slightly larger than the given tolerance value, the corresponding results are acceptable since in this example we have used the exact errors obtained from the known exact solution rather than using the approximate errors that would be provided by the Runge-Kutta Merson process.

Thus in all the three types of problems discussed in this section, the Fourier series method is clearly applicable. This also suggests that the method can be used in a wide variety of problems provided we can resolve the original p.d.e. into a set of o.d.e's which at the present time is facilitated by the availability of symbolic computation packages.
Table 7.16
Numerical solution of the wave equation
$tol = 0.1000E-06$
$dx = 0.5\pi$

<table>
<thead>
<tr>
<th>$t$</th>
<th>$x$</th>
<th>0.0</th>
<th>0.5$\pi$</th>
<th>$\pi$</th>
<th>1.5$\pi$</th>
<th>2$\pi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>$u_1$</td>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
</tr>
<tr>
<td></td>
<td>$u_2$</td>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
<td>0.0000E+00</td>
</tr>
<tr>
<td>1.0</td>
<td>$u_1$</td>
<td>-0.1000E-06</td>
<td>0.45950E-06</td>
<td>-0.1000E-06</td>
<td>0.45950E-06</td>
<td>-0.1000E-06</td>
</tr>
<tr>
<td></td>
<td>$u_2$</td>
<td>-0.1000E-06</td>
<td>0.45950E-06</td>
<td>-0.1000E-06</td>
<td>0.45950E-06</td>
<td>-0.1000E-06</td>
</tr>
<tr>
<td>2.0</td>
<td>$u_1$</td>
<td>-0.87286E-06</td>
<td>0.32476E-06</td>
<td>-0.87286E-06</td>
<td>0.32476E-06</td>
<td>-0.87286E-06</td>
</tr>
<tr>
<td></td>
<td>$u_2$</td>
<td>-0.87286E-06</td>
<td>0.32476E-06</td>
<td>-0.87286E-06</td>
<td>0.32476E-06</td>
<td>-0.87286E-06</td>
</tr>
<tr>
<td>3.0</td>
<td>$u_1$</td>
<td>-0.11172E-05</td>
<td>-0.83857E-06</td>
<td>-0.11172E-05</td>
<td>-0.83857E-06</td>
<td>-0.11172E-05</td>
</tr>
<tr>
<td></td>
<td>$u_2$</td>
<td>-0.11172E-05</td>
<td>-0.83857E-06</td>
<td>-0.11172E-05</td>
<td>-0.83857E-06</td>
<td>-0.11172E-05</td>
</tr>
<tr>
<td>4.0</td>
<td>$u_1$</td>
<td>0.13671E-06</td>
<td>-0.18678E-05</td>
<td>0.13671E-06</td>
<td>-0.18678E-05</td>
<td>0.13671E-06</td>
</tr>
<tr>
<td></td>
<td>$u_2$</td>
<td>0.13671E-06</td>
<td>-0.18678E-05</td>
<td>0.13671E-06</td>
<td>-0.18678E-05</td>
<td>0.13671E-06</td>
</tr>
<tr>
<td>5.0</td>
<td>$u_1$</td>
<td>0.20487E-05</td>
<td>-0.11125E-05</td>
<td>0.20487E-05</td>
<td>-0.11125E-05</td>
<td>0.20487E-05</td>
</tr>
<tr>
<td></td>
<td>$u_2$</td>
<td>0.20487E-05</td>
<td>-0.11125E-05</td>
<td>0.20487E-05</td>
<td>-0.11125E-05</td>
<td>0.20487E-05</td>
</tr>
<tr>
<td>6.0</td>
<td>$u_1$</td>
<td>0.24601E-05</td>
<td>0.13516E-05</td>
<td>0.24601E-05</td>
<td>0.13516E-05</td>
<td>0.24601E-05</td>
</tr>
<tr>
<td></td>
<td>$u_2$</td>
<td>0.24601E-05</td>
<td>0.13516E-05</td>
<td>0.24601E-05</td>
<td>0.13516E-05</td>
<td>0.24601E-05</td>
</tr>
<tr>
<td>7.0</td>
<td>$u_1$</td>
<td>0.22398E-06</td>
<td>0.32716E-05</td>
<td>0.22398E-06</td>
<td>0.32716E-05</td>
<td>0.22398E-06</td>
</tr>
<tr>
<td></td>
<td>$u_2$</td>
<td>0.22398E-06</td>
<td>0.32716E-05</td>
<td>0.22398E-06</td>
<td>0.32716E-05</td>
<td>0.22398E-06</td>
</tr>
<tr>
<td>8.0</td>
<td>$u_1$</td>
<td>-0.29995E-05</td>
<td>0.22289E-05</td>
<td>-0.29995E-05</td>
<td>0.22289E-05</td>
<td>-0.29995E-05</td>
</tr>
<tr>
<td></td>
<td>$u_2$</td>
<td>-0.29995E-05</td>
<td>0.22289E-05</td>
<td>-0.29995E-05</td>
<td>0.22289E-05</td>
<td>-0.29995E-05</td>
</tr>
<tr>
<td>9.0</td>
<td>$u_1$</td>
<td>-0.39484E-05</td>
<td>-0.14898E-05</td>
<td>-0.39484E-05</td>
<td>-0.14898E-05</td>
<td>-0.39484E-05</td>
</tr>
<tr>
<td></td>
<td>$u_2$</td>
<td>-0.39484E-05</td>
<td>-0.14898E-05</td>
<td>-0.39484E-05</td>
<td>-0.14898E-05</td>
<td>-0.39484E-05</td>
</tr>
<tr>
<td>10.0</td>
<td>$u_1$</td>
<td>-0.97667E-06</td>
<td>-0.45694E-05</td>
<td>-0.97667E-06</td>
<td>-0.45694E-05</td>
<td>-0.97667E-06</td>
</tr>
<tr>
<td></td>
<td>$u_2$</td>
<td>-0.97667E-06</td>
<td>-0.45694E-05</td>
<td>-0.97667E-06</td>
<td>-0.45694E-05</td>
<td>-0.97667E-06</td>
</tr>
</tbody>
</table>

Computing Time = 535558 µ-sec
ifail = 0
7.5 PERIODIC SOLUTION OF VAN DER POL'S EQUATION

7.5.1 Introduction

There are physical systems such that for small oscillations, energy is fed into the system, whereas for large oscillations, energy is taken from the system. In other words, large oscillations will be damped, whereas for small oscillations there is "negative damping" (the feeding of energy into the system). For physical reasons we expect such a system to approach a periodic behaviour, which will thus appear as a closed curve in the phase plane, called a limit cycle. A differential equation describing such vibrations is the famous van der Pol equation

\[ \ddot{y} - \lambda(1-y^2)\dot{y} + y = 0, \quad \lambda > 0. \quad (7.5.1) \]

This equation occurs in the study of electrical circuits with vacuum tubes. For \( \lambda = 0 \) this is \( \ddot{y} + y = 0 \) and we obtain harmonic oscillations. Let \( \lambda > 0 \). The damping term has the coefficient \(-\lambda(1-y^2)\). This is negative for small oscillations, namely when \( y^2 < 1 \), so that we have "negative damping;" is zero for \( y^2 = 1 \) (no damping) and is positive if \( y^2 > 1 \) (positive damping, loss of energy). If \( \lambda \) is small, we expect a limit cycle which is almost a circle because then our equation differs only a little from \( \ddot{y} + y = 0 \). If \( \lambda \) is large, the limit cycle will probably look very different.

7.5.2 Numerical Integration of the van der Pol Equation

Eq. (7.5.1) can be reduced into a system of two first order equations by the use of a substitution of variables. For example, if we let \( \dot{y} = z \), eq. (7.5.1) is equivalently written as

\[ \dot{y} = z \quad (7.5.2a) \]
\[ \dot{z} = \lambda(1-y^2)z - y \quad (7.5.2b) \]
In performing the integration for $0 \leq t \leq t_{\text{end}}$ we may use any method or any o.d.e. software package and use any arbitrary initial conditions $y(0)=\alpha, z(0)=\beta$ with $t_{\text{end}}$ large enough to see any sign of periodicity.

By performing this integration procedure one will always observe the following characteristics in the results:

1. With $y(0)$ and $z(0)$ arbitrarily chosen, the solution curve approaches a periodic behaviour as the integration proceeds.

2. Large initial oscillations are damped whereas for small initial oscillations there is a "negative damping".

3. If the solution curve $y(t)$ is plotted against the time, $t$, the curve is seen to repeat its periodic shape more persistently as $t$ increases.

4. If $\dot{y}(t)$ is plotted against $y(t)$ the curve appears to form a closed curve as $t$ increases. This curve is called the limit cycle on the phase plane $y(t)-\dot{y}(t)$. An example of the limit cycle for $\lambda=1$ and two solution curves approaching it are shown in Figure 7.1.
Figure 7.1: The van der Pol equation with $\lambda = 1$ in the phase plane showing the limit cycle and two solution curves approaching it.

5. The closer we choose the initial values $y(0)=a$ and $\dot{y}(0)=\beta$ to the limit cycle, the more rapid the solution curve settles into a periodic form.

6. With any $\lambda$ and any initial value the solution curve is oscillating in between the maximum value $= 2.0$ and the minimum value $= -2.0$ (i.e. $y = +2$ when $\dot{y}=0$). This suggests that for any $\lambda$ the integration may be started more efficiently by using the initial conditions $y(0)=2$, $\dot{y}(0)=0$.

By using this technique we are able to determine the unknown period. By integrating for a number of cycles we could measure the length of the time, $\tau$, to complete the number of cycles, $n$. The period is then obtained from

$$T = \tau/n \quad (7.5.3)$$
For example by using the Runge-Kutta Merson formula with $\lambda = 0.1$ and integrating to $t_{\text{end}} = 50$ we find that the time is $T = 44$ to complete 7 cycles. Thus, the period is given by

$$T = \frac{44}{7} = 6.28.$$ 

For various $\lambda$ we can obtain the corresponding period by following the same procedure.

7.5.3 Greenspan's method for the determination of the period of the van der Pol oscillation

Greenspan [1974] presented a strategy to approximate periodic solutions of the van der Pol equation by applying a numerical integration method. Since it is well known that eq. (7.5.1) has a unique periodic solution for each $\lambda$, we consider the problem as that of determining the constant $\alpha$ for which the solution of the I.V.P defined by (7.5.1) and

$$y(0) = \alpha, \quad y(0) = 0$$

is periodic. This can be explained as follows:

Consider the periodic curve of period $T$ as shown in Figure 7.2.

![Figure 7.2](image-url)
Since the curve is periodic, it is sufficient to determine the curve only within an interval of length $T$ and its periodic extension can be obtained easily outside this interval. In this method, it is more convenient to determine the curve for $\tau < t < \tau + T$ rather than the original interval $0 < t < T$ provided that at $t = \tau$ and $t = \tau + T$ the curve has maximum values, $\alpha$, say [i.e. $y(\tau) = \alpha$, $y'(\tau) = 0$]. Since eq. (7.5.1) does not depend explicitly on $t$, we could easily shift the curve to the left by an amount $\tau$, so that the initial condition becomes $y(0) = \alpha$, $y'(0) = 0$ as given by eq. (7.5.4).

Furthermore, particular to van der Pol equation (see Greenspan [1974], p.53), at the middle point between the two maxima, the solution curve has a unique local minimum where its function value equals $-\alpha$, i.e. $y(T/2) = -\alpha$, $y'(T/2) = 0$. In addition the curve possesses the property that $y(t + T/2) = -y(t)$. Any step-by-step method is therefore sufficient to determine $T$ if used to integrate from $t=0$ to a point, $t_m$, where the function value stops decreasing. $t_m$ is then taken as $T/2$. Greenspan [1974] suggests that the application of these methods should proceed as follows:

Let $y_0^{(n)} = n+1$, $n=0,1,\ldots,10$ be various approximations for $\alpha$. For each $n$ generate, in order, the sequence $y_{k+1}^{(n)}$, $k=0,1,2,\ldots$ by using the chosen integration formula. This sequence will initially be a decreasing one but after reaching its minimum value it will start to increase. We terminate the iteration when

$$y_{K+1}^{(n)} \geq y_K^{(n)} \quad \text{for some } K,$$

and record

$$S_n = y_0^{(n)} + y_K^{(n)} \quad \text{(7.5.5)}$$
The finite sequence \( S_n \), \( n=0,1,\ldots,10 \) will be an increasing sequence which initially is negative. From the first condition of (7.5.4) we seek to find a negative \( y_k^{(n)} \) so that \( S_n \) is zero. With this in mind, let \( n=\mu \) be the first value of \( n \) for which
\[
S_\mu S_{\mu+1} \leq 0. \tag{7.5.6}
\]
Then set \( \alpha=y_0^{(\mu)} \) and \( T/2=K\Delta t \). Thus, \( y_0^{(\mu)} \) is an integer which approximates \( \alpha \). To compute a one decimal place refinement of this approximation, set \( y_0^{(0)} = y_0^{(\mu)} + 0.0 \), \( y_0^{(1)} = y_0^{(\mu)} + 0.1 \), \ldots, \( y_0^{(10)} = y_0^{(\mu)} + 1.0 \), and recycle. In general, a \( j \)-decimal place refinement can be constructed in a similar fashion.

In practice, this method is time consuming to apply. In the following we give an alternative method to determine \( T \) reasonably quickly and yet acceptably accurate. The method is based on the statistical concept of autocorrelation.

7.5.4 Determination of the period by autocorrelation methods

Autocorrelation

Given the values \( u_1, u_2,\ldots,u_n \), the \((n-1)\) pairs \((u_1,u_2), (u_2,u_3),\ldots,(u_{n-1},u_n)\) constitute a set of bivariate values which have a correlation coefficient of the standard kind. Likewise for the \((n-2)\) pairs \((u_1,u_3), (u_2,u_4), (u_3,u_5)\) and so on. We call the coefficient \((k-1)\) terms apart, i.e. of \( u_t \) and \( u_{t+k} \), the "serial correlation of order \( k \)" and denote it by \( r_k \). In full generality the \( r_k \) is given by

\[
r_k = \frac{\frac{1}{n-k} \sum_{i=1}^{n-k} (u_i - \frac{1}{n-k} \sum_{i=1}^{n-k} u_i) (u_{i+k} - \frac{1}{n-k} \sum_{i=1}^{n-k} u_{i+k})}{\left( \frac{1}{n-k} \sum_{i=1}^{n-k} (u_i - \frac{1}{n-k} \sum_{i=1}^{n-k} u_i)^2 \right)^{\frac{1}{2}} \left( \frac{1}{n-k} \sum_{i=1}^{n-k} (u_{i+k} - \frac{1}{n-k} \sum_{i=1}^{n-k} u_{i+k})^2 \right)^{\frac{1}{2}}} \tag{7.5.7}
\]
This can be modified so as to measure all the variables about the mean of the whole series of \( u_1, u_2, \ldots, u_n \), and also replace the variance terms in the denominator by the variance of the whole series. The expression then simplifies greatly to

\[
 r_k = \frac{1}{n-k} \sum_{i=1}^{n-k} (u_i - \bar{u}) (u_{i+k} - \bar{u}) \overline{\sum_{i=1}^{n} (u_i - \bar{u})^2}
\]

(7.5.8)

where \( \bar{u} \) is the mean given by

\[
 \bar{u} = \frac{1}{n} \sum_{i=1}^{n} u_i
\]

We will use this regression analysis to relate observations to their predecessors. In particular, for a random distribution whose frequency is \( f(t) \) we could choose a set of values \( f(t_i) \) where \( \{t_i\}_{i=0}^{n} \) is the subdivision points of \( t \) in \([a,b]\) with \( t_i = a + ih \), \( h = (b-a)/n \). From (7.5.8) it follows that \( r_0 = 1 \).

If \( f(t) \) is periodic with period \( T \), then for a sufficiently large interval \([a,b]\) there exists an integer \( k = K \) such that \( r_k = 1 \). The smallest such \( K \) will give a span of one period. The period is then given by the relation

\[
 T = Kh
\]

(7.5.9)

**Determination of the period to the solution of the van der Pol's equation**

We know that the solution of the van der Pol's equation (7.5.1) is periodic for any \( \lambda \geq 0 \). We have also seen that the amplitude of the
solution is approximately $= 2.0$. The method that we are going to use comprises of two parts:

(i) The first part is to solve eq. (7.5.1) by using the initial conditions $y(0)=2$, $\dot{y}(0)=0$ within a sufficiently large interval, say $[0,50]$. As eq. (7.5.1) is stable, the solution curve is self-correcting to the limit-cycle, i.e., any cycle nearer to the right end of the interval represents a better approximation to the actual oscillation. Therefore we will pick only those solutions near the right end of the interval, say within $[20,50]$, and forget those solutions on the left portion. For our convenience we indicate $u_1 = y(20)$, $u_2 = y(20+h)$, ..., $u_n = y(50)$.

(ii) The second part is to apply the autocorrelation procedure to see whether any data given in (i) are related. In particular, we are looking for any sign of periodicity in those data. By applying formula (7.5.8) we may compute $r_k$ for $k=0,1,2,\ldots,n-1$ with $r_0=1$. We are particularly interested in finding the next integer $k=K$ for which $r_k$ has value 1 for the second time after $r_0$. To save time we may only compute $r_k$ at an initial coarse interval, say 20, to obtain $r_0$, $r_{20}$, $r_{40}$, etc. After a certain value of $k$ these values will become sufficiently close to 1. For example, this can be achieved when $k=M$ characterised by $|r_{M-1}| < \epsilon$ for a suitably chosen value of $\epsilon$. From now on we will compute $r_k$ at a finer interval of 1 starting back from $k=M-20$, (say), to determine the more accurate location of $k$ for which $r_k=1$. This location is spotted when $r_k$ starts to decrease again at $k=K$. The value $r_K=1$ implies that every item of data separated $K$ steps apart are similar, and therefore periodicity is achieved. This period is then given by

$$T = Kh.$$
Computer Program

The computer program consists essentially of two parts:

1. To obtain the solution values at the RHS region of the interval. For this purpose we use Runge-Kutta Merson method in NAG routine D02BBF.
2. To obtain the period T by using the autocorrelation technique.

For the purpose of comparing the timings for this method with that of the other methods, the time spent in Stage (1) is added with the time spent in Stage (2). Basically, in practice, the time consuming part is the evaluation of the solution, especially over a long range, required by large values of \( \lambda \). The time taken by the autocorrelation process depends on the number of points used, and the number of coefficients \( a_k \) evaluated.

Numerical Example

In this example we compare the timing taken by the method of Greenspan and the autocorrelation method which we have just discussed to the problem of finding the period, T, in the solution of the van der Pol equation for \( \lambda=0(5)20 \). The results are shown in Table 7.17. In the method of autocorrelation, the calculation is rounded to the nearest 0.05. If higher accuracy is required, this can be achieved by taking a smaller stepsize, h, and probably we will have to reduce the interval of integration to maintain the speed.
Our results show that the method of autocorrelation is very much favoured in terms of speed. In terms of accuracy nothing much can be said since we do not have the exact solution to refer to. Basically the two results are acceptably close to each other.
CHAPTER 8

CONCLUSIONS AND RECOMMENDATIONS

FOR FURTHER WORK
In this thesis our study has been devoted to three major areas of research, namely, (i) the development of the GM formula and the methods which are based on the concept of geometric mean; (ii) the extension of the concepts and techniques of the geometric mean to the Runge-Kutta, IMS and the multiderivative methods; and (iii) the development of new methods suitable for oscillatory problems.

Our detailed study commenced in Chapter 4 by introducing the GM formula and the analysis of its accuracy and its stability properties. It was established that the GM method is of second order, similar to the trapezoidal method but with slightly different error terms which makes it more accurate for use in some problems and hence could serve as a complement to the trapezoidal formula. In terms of stability, it was found that the method possesses the important L-stability property which makes it more suitable for use for a wider range of problems than any formula in the so-called $\Theta$-class formulae. Our numerical examples showed that the GM formula is behind only in terms of the computing time involved, but in some class of problems it can be made competitively fast or even faster than the trapezoidal rule.

The application of the GM method in the predictor corrector manner was studied in the second section of the chapter. With Euler's method used as the predictor, two applications of the GM (corrector) formula was found necessary to obtain full accuracy of the GM formula, but if the Newton Cotes open formula is used, only a single application of the corrector formula would be necessary. The application of the GM formula with extrapolation was more interesting in that the method could give a better convergence rate when used with a small number of extrapolation points. This is due to the fact that the GM method has a smaller first five coefficients $c_v(x)$ of the asymptotic expansion for
the solution of \( y' = \lambda y \) than the corresponding expansion for the trapezoidal formula solution. Again, the method could serve as a complement to the trapezoidal or the backward Euler formulae when used with extrapolation.

The applicability of the GM formula was further explored for the use of solving delay-differential equations in which its performance was found to be as good as the trapezoidal formula.

In Chapter 5, we examined the applicability of the GM strategy in the Runge-Kutta type formulae, where instead of the normal AM weighting, we use the weighted GM averaging on the evaluated functional values in the Runge-Kutta formulae. GM Runge-Kutta formulae of order 1 to 4 have already been derived. Unlike the AM Runge-Kutta formulae, the stability regions of the GM Runge-Kutta formula can be varied by using different sets of parameters which produce the same degree of accuracy. Thus, a different set of similar order formulae may be used depending on the stability requirement imposed by the problems to be solved. It is thought that higher order GM Runge-Kutta formulae are possible by employing the technique used in the development of the lower order formulae.

One important characteristic of the AM and GM Runge-Kutta formulae is that while their order can be made similar by using the similar function evaluations, their truncation errors are generally different. By using the error terms in both computations it was possible to estimate the magnitude of the truncation error of one of the formulae, and this was found to be superior, as against the older methods such as of Merson [1957] and Scraton [1964], without any additional function evaluation. This is the area where the GM strategy has made a great contribution to the numerical solution of ordinary differential
equations. This has essentially become the basis of the Arithmeto-Geometric Mean (AGM) methods of which we have developed the second, third and fourth order formulae.

Closely related to this approach of error estimation, we have extended the idea of the GM technique to the development of GM formulae equivalent to the Fehlberg [1969] strategy of error control. In particular, we have derived the GM-RK2(3) formula with minimum truncation error which uses only three function evaluations. We have also shown that in terms of accuracy, this formula has a smaller truncation error than the most accurate AM-RK2(3) formula which also uses three function evaluations. This idea was also used to develop third order GM-RK Fehlberg formula. It was shown that unlike the Fehlberg variation which requires five function evaluations, the GM-RK3(4) formula requires only four function evaluations. The only drawback of the GM variant is that the last function evaluation from the previous step cannot be used as the first evaluation for the current step. However, it is thought to be possible to find the most accurate GM-RK3(4) formula which, if found to be more accurate than the corresponding AM formula, would make it more competitive. This could be a promising research area awaiting further investigation.

The GM concept was further extended to the Iterated Multistep (IMS) formulae where instead of AM formulae, GM formulae were used as the base formulae. With the help of the REDUCE symbolic computation package, we have also determined the general coefficients in the Hyman's leap-frog formula which is suitable for variable stepsize implementation. Incidentally the REDUCE computer program could be used to generate the coefficients of the IMS formula which uses any other formula as the base. We concluded the chapter with the derivation of
the two-stage implicit GM Runge-Kutta formula, which unfortunately is only of third order as compared to the fourth order accuracy attained by the corresponding AM formula.

In Chapter 6, further applications of the GM techniques were considered. In the first section of the chapter, generalised GM multi-derivative methods were developed. We have established the equivalence of the Pade' Table in the geometric sense where the off-diagonal entries are essentially similar to the AM case, and the diagonal entries differ only in the terms that involve the highest order derivatives. In particular, we have developed the \( <2,2> \) formula of order 4 which is A-stable.

In the second section, we developed GM multistep formulae, namely the GM 2-step formula and its modification which is 3rd order accurate. The application of the GM formula to the problems of the form \( y' = \lambda(x)y \) was considered in the third section. The L-stability property of the method proved useful for solving the equation stably. A modification of the formula which is equivalent to Gourlay's version of the trapezoidal formula shows an even better performance in practice. This was verified by a numerical example and in the solution of the Goursat problem in hyperbolic partial differential equation.

In the development of methods for solving o.d.e's it was not clear that the arithmetic mean is always the best choice. In Section 6.5 we showed that the harmonic mean, the logarithmic mean and a new mean we developed could equally provide consistent approximations. In particular when used in a similar fashion to the trapezoidal and the GM formulae these means would all give second order approximations but with different error terms, all of them deviating from \( -\frac{1}{12}h^3 y_n'' \) by a certain amount. Thus, the truncation error of the trapezoidal formula
could be considered as the centre amongst these errors and the most accurate formula among them for a certain problem would be the one which will give the least truncation error. A numerical example given in this section has shown how these other means could produce more accurate results than the trapezoidal or the GM formulae. The HM formula was found to be L-stable (see Fig. 6.4) while the method based on our new mean is A-stable (see Fig. 6.6). Further research is thought to be possible to extend the application of these means to the Runge-Kutta, IMS and multiderivative formulae in a similar way as we did to the GM formula.

In Chapter 7 we considered the numerical solution of oscillatory problems. With the help of the REDUCE symbolic computation we explored further, in Section 7.1, Gautschi's method for oscillatory problems by developing more formulae which include, amongst other, a GM version of the formula. The development of the Runge-Kutta type formula for oscillatory problems was attempted but at that stage it was unsuccessful. A more comprehensive analysis is thought necessary to relate the parameters in the function evaluations with the frequency of the oscillation. The subsequent three sections of the chapter dealt with the solution of partial differential equations by Fourier series method. Again, with the help of the REDUCE symbolic computation package we have successfully extended the application of the Fourier series method to some nonlinear parabolic and hyperbolic equations. Several classes of problems were considered and numerical examples were provided.

The research work was concluded by the examination of the periodic solutions of the van der Pol equation. A new technique of determining the period of the solution by using the statistical idea of
autocorrelation was presented. It was shown that the technique can produce an accurate result in a shorter time than the method offered by Greenspan.

Finally, it may be mentioned here that in this thesis we have not considered exclusively the extension of the GM formula to systems of differential equations. Nevertheless, the applicability of the method to deal with such situations has been practically proven. An example of the application of the GM formula for a system was shown in Subsection 4.2.3. Generally we would say that the application to linear systems is quite easy, i.e.,

\[ y' = Ay + f \]

Assume that A is symmetric or has distinct eigenvalues. Then the eigenvalue-eigenvector relation gives us

\[ AU = UA \quad \text{or} \quad A = UAU^{-1} \]

where \( U \) = matrix of eigenvectors and \( A \) = diagonal matrix.

Then substituting in the differential system we have,

\[ U^{-1}y' = AU^{-1}y + U^{-1}f, \]

which becomes,

\[ z' = \Lambda z + f \quad \text{where} \quad z = U^{-1}y. \]

The system is now separable and each equation could be treated individually.

For arbitrary systems the situation is more complex, i.e.,

\[ y' = f(y), \quad y = (y_1, y_2, \ldots, y_n). \]
Then the GM formula becomes,

\[ y_{n+1} = y_n + h \sqrt{y_n'y_n'} \]

and a definition for the last term is required. In our practical usage of the GM formula we interpreted this as, i.e.

\[ \sqrt{y_n'y_n'} = \text{the vector of the square root} \]

\[ \text{of element-element multiplication} \]

\[ \text{of the two vectors } y_n' \text{ and } y_{n+1}' \]

which has proved to give satisfactory consistent approximations.


Cohen, A.M. [1985], "A bounding approach to \(a^{1/p}\)," *Computing* 34, pp.87-89.


Ehle, B.L. [1968], "High-order A-stable methods for the numerical solution of systems of differential equations, BIT, 8, pp.276-278.


Marchuk, G.I., and V.V. Shaidurov [1983], *Difference Methods and Their Applications*, Springer-Verlag, New York.


National Physical Laboratory [1961], *Modern Computing Methods*, Her Majesty's Stationary Office.


Vu, T.V. [1981], "Modified Runge-Kutta Methods for Solving ODEs," Report UIUCDSC-R-81-1069, Department of Computer Science, University of Illinois at Urbana-Champaign.


Appendix 1

program extrap

c

-----This program solves a system of neq ordinary
c-----differential equations by using the GM, Euler or
c-----the Trapezoidal formulae with extrapolation.
c-----A subprogram to evaluate the function is given

-----separately.

implicit real*8 (a-h,o-r,u-z)
real t
real*8 stepsz
integer s
character*11 met(3)
dimension y(10),p(0:15,0:15),nstep(0:15),h(0:15),
r1(0:15),r2(0:15)
external f, soln

-----first, collect the data needed.
data met/'GM', 'EULER', 'TRAPEZOIDAL'/

print*, 'select method: 1 for GM, 2 for Euler,
~ 3 for Trapezoidal'
read(5,*) isel
print *, 'give epsilon, the error tolerance used as
~ stopping criteria'
read(5,*) eps
print *, 'give initial value of the independent
~ variable, x0'
read(5,*) x0
print *, 'give the final value, xf'
read(5,*) xf
print *, 'give the initial value of the dependent
~ variable(s)' read(5,*) y(1)

continue

print *, 'give the basic steplength'
read(5,*) dx
print*, 'give the maximum number of extrapolation
~ points allowed'
read(5,*) npt

-----write the heading
write(1,170) met(isel),dx,npt,eps

format('Method with Polynomial
~ Extrapolation'/17x, 'basic steplength =', f5.3/7x,
~'max no of extrapolation points =',i2/7x,'epsilon =',e6.1)
s=npt-1
n=idint((xf-x0)/dx)
ytemp=y(1)
write(1,189)

format('/12x,1hx,8x,8hex' sol')
write(1,175)x0,y(1)

format(f15.4,e16.8/)
do 2 j=0,npt
   nstep(j)=2**j
   h(j)=dx/nstep(j)
2   continue

c-----the main loop
   do 1 icount=1,n
      do 500 j=0,s
         c-----set the parameters for the subroutine call
         neq=1
         x=x0+(icount-1)*dx
         y(1)=ytemp
         ns=nstep(j)
         stepsz=h(j)

         c-----to avoid repeating a non-convergent iteration, limit
         c-----the maximum iteration allowed by using isafe = 1;
         c-----if isafe .ne. 1 then stop the iteration procedure.
         isafe=1
         if (isel.eq.1) call gm(neq,x,y,ns,stepsz,f,isafe)
         if (isel.eq.2) call euler(neq,x,y,ns,stepsz,f)
         if (isel.eq.3) call trapez(neq,x,y,ns,stepsz,f,isafe)
         if(isafe.ne.1) go to 99
         p(0,j)=y(1)
         if(j.eq.0) then
            write(1,31)h(j),y(1)
         go to 500
         end if
         if(isel.eq.1) gamma=2.0
         if(isel.eq.2) gamma=1.0
         if(isel.eq.3) gamma=2.0
         do 400 m=1,j
            p(m,j-m)=p(m-1,j-m+1)+(p(m-1,j-m+1)-p(m-1,j-m))/{
            ((h(j-m)/h(j))**gamma-1.)
        400 continue
         write(1,31)h(j),y(1), (p(m,j-m),m=1,j)
31 format(2x,'h =',f10.8,1h: ,15d15.8)
         if(abs((p(j,0)-p(j-1,0))/p(j-1,0)).lt.eps) go to 700
      500 continue
      j=j-1
   700 y(1)=p(j,0)
ytemp=p(j,0)
exact=soln(x)
err1=y(1)-exact
errpc1=100.0*err1/exact
   write(1,97)
97 format(/12x,1hx,8x,8hex' sol',5x,9hnum' sol',7x,5herror,
   10x,6h2error,10x,1hs)
   write(1,191)x,exact,y(1),err1,errpc1,j
191 format(f15.4,1x,4d15.8,16/)
c-----the ratios that determine the rate of convergence
   c-----are next evaluated, i.e. the error with n+1 points
   c-----extrapolation is compared with the previous n
   c-----points extrapolation.
   do 77 k=0,j-1
r1(k)=abs((p(0,k+1)-exact)/(p(0,k)-exact))
r2(k)=abs((p(k+1,0)-exact)/(p(k,0)-exact))
continue
write(1,192) (r1(k),k=0,j-1)
192 format(2x,'ratios of method convergence, Rn are:'/10f9.5)
write(1,193) (r2(k),k=0,j-1)
193 format(2x,'ratios of extrapolated convergence, Rn are:'
~'/10f9.5)
write(1,92)
92 format(/)
continue
print*, 'if you want to change stepsize type 1,
~if not type any other number'
read(5,*) iff
if(iff.eq.1) go to 10
99 print*, 'error tolerance not satisfied'
98 stop
end

!-----defining the function:
function f(i,x,y)
imPLICIT REAL*8 (A-H,O-Z)
dimension y(10)
f=y(1)
return
end

!-----defining the solution:
function soln(x)
real*8 x
soln=dexp(x)
return
end

!-----Now, the subroutines which define the formulae

! subroutine gm (n,x,y,k,h,f,isafe)
!-----this subroutine advances the system of n ode's
!-----y(i)'=f(i,x,y1,y2,...,yn) by k steps of size h
!-----using the nonlinear Euler method.
!-----The other parameters are:
!-----x=initial value of the independent variable;
!-----it is increased by the subroutine
!-----y=initial value of the dependent variable(s);
!-----the routine overwrites this with the new value(s).
implicit real*8 (a-h,o-z)
external f
dimension y(10),yiter(10),yprev(10),error(10)
!-----The main loop
do 1 j=1,kiter=0
do 2 i=1,n
c---- A temporary array is needed in order to avoid using
c---- the updated values of yi
  yiter(i)=y(i)+h*f(i, x, y)
2  continue
c
c---- enorm = RMS norm of the improvement error,
c---- snorm = RMS norm of the improved solution values.
5  errsum=0.0
  solsum=0.0
  do 4 i=1,n
    yprev(i)=yiter(i)
    if(f(i, x, y).ne.0) then
      plsmin=sign(1., f(i, x, y))
    else
      plsmin=sign(1., f(i, x+h, yiter))
    end if
    yiter(i)=y(i)+plsmin*h*sqrt(abs(f(i, x, y)*f(i, x+h, yiter)))
  iter=iter+1
  if(iter.gt.100) go to 9
  error(i)=yiter(i)-yprev(i)
  errsum=errsum+error(i)*error(i)
  solsum=solsum+yiter(i)*yiter(i)
4  continue
  enorm=sqrt(errsum)
  snorm=sqrt(solsum)
  errtol=.00000001
  if(enorm/snorm.gt.errtol) go to 5
end
x=x+h
  do 6 i=1,n
    y(i)=yiter(i)
6  continue
1  continue
  go to 11
9  isafe=2
11  return
end

c

subroutine euler(n, x, y, k, h, f)
c---- comments as in the 'subroutine gm'
implicit real*8 (a-h, o-z)
external f
dimension y(10)
do 10 j=1,k
do 20 i=1,n
  y(i)=y(i)+h*f(i, x, y)
20  continue
x=x+h
10  continue
return
end
subroutine trapez (n,x,y,k,h,f,isafe)

subroutine as in the 'subroutine gm'

implicit real*6 (a-h,o-z)
ex

dimension y(10),yiter(10),yprev(10),error(10)
do 1 j=1,k
iter=0
2 continue

5 errsum=0.0
solsum=0.0
do 4 i=1,n
yprev(i)=yiter(i)
yiter(i)=y(i)+h/2*(f(i,x,y)+f(i,x+h,yiter))
iter=iter+1
if(iter.gt.100) go to 9
error(i)=yiter(i)-yprev(i)
errsum=errsum+error(i)*error(i)
solsum=solsum+yiter(i)*yiter(i)
4 continue
enorm=sqrt(errsum)
snorm=sqrt(solsum)
errtol=.00000001
if(enorm/snorm.gt.errtol) go to 5
x=x+h
6 continue
1 continue
11 return
end
Appendix 2

program ddes
c-----DELAY DIFFERENTIAL EQUATION
c-----SOLUTION BY AM AND GM FORMULA WITH LINEAR INTERPOLATION
c----- (small delay problem)
common a,b
       print *, 'give method, 1 for AM, 2 for GM'
       read(5,*) method
       a=-2.1
       b=1.0
       d=log(1.1)
       print*, 'give h and k'
       read(5,*) h,k
       u0=1.0
       z0=1.1
       errtol=.0000001
       x0=0.0
       xend=40.0
       if(d.gt.h) then
           print *, 'Large delay, use large delay program ddel'
           stop
       end if
       write(1,10) x0,u0
       10 format(f7.3,5e16.6)
       n=ifix((xend-x0)/h)
       do 50 i=1,n
          x1=x0+h
          u1=u0+h*f(x0,u0,z0)
          temp=u1
          z1=d/h*u0+(1-d/h)*u1
          if(method.eq.1) u1=u0+h/2*(f(x0,u0,z0)+f(x1,u1,z1))
          if(method.eq.2) u1=u0+h*sign(1.,f(x0,u0,z0))*sqrt( 
          "abs(f(x0,u0,z0)*f(x1,u1,z1)))
          if(abs(u1-temp).gt.errtol) go to 30
       c
       c next step
       c
          x0=x1
          u0=u1
          z0=z1
          if(i/k.ne.float(i)/k) go to 50
       exact=soln(x1)
       error=exact-u1
       errpc=100.*error/exact
       write(1,10)x1,exact,u1,error,errpc
       50 continue
       stop
       end
function f(x,u,z)
common a,b
f=a*u+b*z
return
end

function soln(x)
soln=exp(-x)
return
end

program ddel

!----(large delay problem)
dimension u(0:1000)
common a,b
print *, 'give method, 1 for AM, 2 for GM'
read(5,*) method
a=-10.0
b=1.0
d=log(9.0)
print *, 'give h and k'
read(5,*) h,k
u(0)=1.0
z0=9.0
errtol=.0000001
xinit=0.0
x0=xinit
xend=20.0
if(d.lt.h) then
   print *, 'Small delay, use small delay program ddes'
   stop
endif
write(1,35) x0,u(0)
n=ifix((xend-x0)/h)
do 50 i=1,n
x1=x0+h
u(i)=u(i-1)+h*f(x0,u(i-1),z0)
s=x1-d
do 10 j=0,i-1
if(s.ge.xinit+j*h.and.s.lt.xinit+(j+1)*h) then
  r=(s-(xinit+j*h))/h
  z1=(1.-r)*u(j)+r*u(j+1)
go to 20
end if

10 continue
print *, ' s=', s,' supply z(s)'
z1=exp(-s)
20 do 30 iter=1,30
  temp=u(i)
  if(method.eq.1) u(i)=u(i-1)+h/2*(f(x0,u(i-1),z0)
  ~+f(x1,u(i),z1))
    if(method.eq.2) u(i)=u(i-1)+h*sign(1.,f(x0,u(i-1),z0))*
      ~sqrt(abs(f(x0,u(i-1),z0)*f(x1,u(i),z1)))
    if(abs(u(i)-temp).lt.errtol) go to 31
30 continue
31 x0=x1
  z0=z1
  if(i/k.ne.float(i)/k) go to 50
  exact=soln(x1)
  error=exact-u(i)
  write(1,35) x1,exact,u(i),error
35 format(f6.3,5e16.6)
50 continue
stop
end

function f(x,u,z)
  common a,b
  f=a*u+b*z
return
end

function soln(x)
  soln=exp(-x)
return
end
Appendix 3

comment: "GMRK"

This REDUCE program is used to obtain the equations of conditions for the derivation of the 4-stage, 4th order Geometric-Mean Runge-Kutta formula. The values of a1, s2 and s3 given here are only examples and can be varied.

% %
operator power, taylor;
for all x let taylor(x)=f+x*f1+x**2*f2/2+x**3*f3/6
+x**4*f4/24+x**5*f5/120;
for all x, k let power(x,k)=1+k*x+k*(k-1)*x**2/2
+k*(k-1)*(k-2)*x**3/6+k*(k-1)*(k-2)*(k-3)*x**4/24;
let h**5=0;
a1=-1/3;
s2=-2/3;
s3=-1;
let a2+a3=s2;
let a4+a5+a6=s3;
k1=f;
k2=taylor(h*a1*k1);
k3=taylor(h*a2*k1+h*a3*k2);
k4=taylor(h*a4*k1+h*a5*k2+h*a6*k3);
% %
x2=k2/f-1;
x3=k3/f-1;
x4=k4/f-1;
k2p=power(x2,p);
k3q=power(x3,q);
k4r=power(x4,r);
rhs=h*f*k2p*k3q*k4r;
lhs=h*f+h**2*f*f1/2+h**3*(f*f1**2/6+f**2*f2/6)
+h**4*(f**3*f3/24+f**2*f1**3/24+f**2*f1*f2/6)
+h**5*(f**4*f4+7*f**3*f1**3+f3+11*f**2*f1**2)
+f**3*f2**2+f*f1**4)/120;
err:=lhs-rhs;
end;
This REDUCE program is used to obtain the equations of conditions for the derivation of the 4-stage, 4th order Geometric-Mean Runge-Kutta formula. The values of $a_1$, $s_2$ and $s_3$ given here are only examples and can be varied;

%  
% operator power, taylor;

for all $x$ let taylor($x$) = $f + x*f_1 + x^2*f_2/2 + x^3*f_3/6 + x^4*f_4/24 + x^5*f_5/120$;

for all $x, k$ let power($x, k$) = $1 + k*x + k*(k-1)*x^2/2 + k*(k-1)*(k-2)*x^3/6 + k*(k-1)*(k-2)*(k-3)*x^4/24$;

let $h^5 = 0$;

$a_1 := 1/3$;

$A_1 := 1/3$

$s_2 := 2/3$;

$s_2 := 2/3$

$s_3 := 1$;

$s_3 := 1$

let $a_2 + a_3 = s_2$;

let $a_4 + a_5 + a_6 = s_3$;

$k_1 := f$;

$K_1 := F$

$k_2 := \text{taylor}(h*a_1*k_1)$;

$k_2 :=$

$$3 \begin{array}{ccc} 4 & 2 & 3 \end{array} 2 \quad (F*(F*H*F4 + 12*F*H*F3 + 108*F*H*F2 + 648*H*F1 + 1944))/1944$$

1944
k3:=taylor(h*a2*k1+h*a3*k2);

K3 :=


k4:=taylor(h*a4*k1+h*a5*k2+h*a6*k3);

K4 :=


x2:=k2/f-1;

X2 :=

\[ (H * (F * H * F4 + 12F * H * F3 + 108F * H * F2 + 648F * F1)) / 1944 \]

x3:=k3/f-1;

X3 :=

\[ x_4 = k_4/f - 1; \]

\[ X_4 = \]
\[
\begin{array}{cccccccc}
3 & 3 & 2 & 3 & 2 & 3 & 2 & 3 \\
F & H & F & 2 & A & 5 & 2 & 2 \\
\end{array}
\]
\[
\]

\[ k_2P = \text{power}(x_2, p); \]

\[ K_2P = \]
\[
\begin{array}{cccccccc}
3 & 4 & 2 & 4 & 2 & 2 & 4 & 2 \\
F & H & P & F & 4 & 4F & H & P \\
\end{array}
\]
\[
(F * H * P * F4 + 4F * H * P * F1F3 + 3F * H * P * F2 - 4F * H * P * F1* F3 - 3F * H * P * F2 + 12F * H * P * F3 + 6F * H * P * F1 * F2 - 18F * \\
4F * P * F1 * F2 + 12F * H * P * F1 * F2 + 36F * H * P * F1 * F2 - 36F * H * P \\
4F * P * F1 * F2 + 108F * H * P * F2 + H * P * F1 - 6F * H * P * F1 + 11F * H * P * \\
4F * F1 - 6F * H * P * F1 + 12F * H * P * F1 - 36F * H * P * F1 + 24F * H * P * F1 \\
+ 108F * H * P * F1 - 108F * H * P * F1 + 648H * P * F1 + 1944)/1944
\]

\[ k_3Q = \text{power}(x_3, q); \]

\[ K_3Q = \]
\[
\begin{array}{cccccccc}
3 & 4 & 2 & 4 & 2 & 2 & 4 & 2 \\
Q * F & 1 * F & 3 * A & 3 - 16F * H * Q * F1F3 + 12F * H * Q * F2 + 39F * H * \\
Q * F1F3A3 - 16F * H * Q * F1F3 + 18F * H * Q * F2 * A3 - 12F * H * Q \\
2 & 2 & 3 & 4 & 3 & 2 & 4 & 2 \\
F2 + 24F * H * Q * F3 + 24F * H * Q * F1 * F2 + 126F * H * Q * F1 * F2* \\
A3 - 72F * H * Q * F1 * F2 + 27F * H * Q * F1 * F2A3 - 126F * H * Q * F1
\]
4 2
3
F2*A3 + 48*F*H *Q*F1 *F2 + 72*F*H *Q *F1*F2 + 135*F*H *Q*F1 *
3 2
2
F2*A3 - 72*F*H *Q*F1*F2 + 108*F*H *Q*F2 + 4*H *Q *F1 + 36*H *
3 4
4
3 4
4 2 4 2 4 2 4 4 2 4
Q *F1 *A3 - 24*H *Q *F1 + 27*H *Q *F1 *A3 - 108*H *Q *F1 *A3
4 2 4
4 4 2 2 4 2 2 4 2 4
+ 44*H *Q *F1 - 27*H *Q*F1 *A3 + 72*H *Q*F1 *A3 - 24*H *Q*
4 3 3 3 3 2 3 3 2 3 3
F1 + 24*H *Q *F1 + 108*H *Q *F1 *A3 - 72*H *Q *F1 - 108*H *
3 2 2 2 2 2 2 2 2 2 2
Q*F1 *A3 + 48*H *Q*F1 + 108*H *Q *F1 + 162*H *Q*F1 *A3 - 108
2 2
H *Q*F1 + 324*H*Q*F1 + 486)/486
k4r:=power(x4,r);

K4R :=
3 4
2 2 4 2 2 4 2 2 2 2 2
(27*F *H *R*F4 + 108*F *H *R *F1*F3 + 81*F *H *R *F2 + 112*F *
4 2
2 4
H *R*F1*F3*A5 + 248*F *H *R*F1*F3*A6 - 108*F *H *R*F1*F3 + 36*
2 4
2 2 4 2 2 4 2 2 4 2 2
F *H *R*F2 *A5 + 144*F *H *R*F2 *A6 - 81*F *H *R*F2 + 108*F *
3 4 3 2 4 2 2 2 2 2 2 2 2
H *R*F3 + 162*F*H *R *F1 *F2 + 360*F*H *R *F1 *F2*A5 + 792*F*
4 2 2
4 2 2
4 2 2
2 4 2 4 2 4 2 4 2 4 2 4
*F1 *F2*A5 + 144*F*H *R*F1 *F2*A6 - 792*F*H *R*F1 *F2*A6 +
4 2 2
3 2 3 2 3 2 3 2 3 2 3 2 3 2
3 2 2 2 2 2 2 2 2 2 2 2 2 2 2
4 4 4 3 4 4 3 4 4 3 4 4 3 4
*R *F1 + 108*H *R *F1 *A5 + 216*H *R *F1 *A6 - 162*H *R *F1
4 2 4
4 2 4 2 4 2 4 2 4 2 4
+ 216*H *R *F1 *A3*A6 + 36*H *R *F1 *A5 + 144*H *R *F1 *A5*
4 2 4
4 2 4 2 4 2 4 2 4 2 4
A6 - 324*H *R *F1 *A5 + 144*H *R *F1 *A6 - 648*H *R *F1 *A6
\[
\begin{align*}
4^2 & 4^4 & 4^4 & 4^2 \\
+ 297^H \ & *R \ & *F1 \ & - 216^H \ & *R\ & *F1 \ & *A3 \ & *A6 \ & - 36^H \ & *R\ & *F1 \ & *A5 \ & - 144^* & \\
4^4 & 4^4 & 4^4 & 4^2 & 4^2 & 4^2 \\
H \ & *R\ & *F1 \ & *A5 \ & *A6 \ & + 216^H \ & *R\ & *F1 \ & *A5 \ & - 144^H \ & *R\ & *F1 \ & *A6 \ & + 432^H \ & *R \\
4^4 & 4^4 & 3 & 3 & 3 & 3 & 3 & 2 & 3 \\
*F1 \ & *A6 \ & - 162^H \ & *R\ & *F1 \ & + 108^H \ & *R\ & *F1 \ & + 216^H \ & *R\ & *F1 \ & *A5 \\
3 & 2 & 3 & 3 & 2 & 3 & 3 & 3 & 3 \\
432^H \ & *R\ & *F1 \ & *A6 \ & - 324^H \ & *R\ & *F1 \ & + 216^H \ & *R\ & *F1 \ & *A3 \ & *A6 \ & - 216^H \\
3 & 3 & 3 & 3 & 3 & 3 & 3 & 2 & 2 & 2 \\
*R\ & *F1 \ & *A5 \ & - 432^H \ & *R\ & *F1 \ & *A6 \ & + 216^H \ & *R\ & *F1 \ & + 324^H \ & *R\ & *F1 \\
2 & 2 & 2 & 2 & 2 & 2 \\
216^H \ & *R\ & *F1 \ & *A5 \ & + 324^H \ & *R\ & *F1 \ & *A6 \ & - 324^H \ & *R\ & *F1 \ & + 648^H \ & *R\ & *F1 \\
+ 648)/648 \\
\text{rhs} := h*f*k2p*k3q*k4r; \\
\text{RHS} := \\
2 & 3 & 2 & 3 & 2 & 3 & 3 & 3 & 3 \\
(1944^F\ & *H\ & *(F\ & *H\ & *P\ & *F3 \ & + 8^F\ & *H\ & *Q\ & *F3 \ & + 27^F\ & *H\ & *R\ & *F3 \ & + 3^F\ & *F\ & *H\ & * \\
2 & 3 & 3 & 3 & 3 & 2 & 2 & 2 & 2 \\
P\ & *F1\ & *F2 \ & + 18^F\ & *H\ & *P\ & *Q\ & *F1\ & *F2 \ & + 36^F\ & *H\ & *P\ & *R\ & *F1\ & *F2 \ & - 3^F\ & *H\ & *P \\
3 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\
*F1\ & *F2 \ & + 24^F\ & *H\ & *Q\ & *F1\ & *F2 \ & + 90^F\ & *H\ & *Q\ & *R\ & *F1\ & *F2 \ & + 45^F\ & *H\ & *Q \\
3 & 3 & 2 & 3 & 3 & 2 & 3 & 3 & 3 \\
F1\ & *F2\ & *A3 \ & - 24^F\ & *H\ & *Q\ & *F1\ & *F2 \ & + 81^F\ & *H\ & *R\ & *F1\ & *F2 \ & + 63^F\ & *H\ & *R \\
3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
F1\ & *F2\ & *A5 \ & + 144^F\ & *H\ & *R\ & *F1\ & *F2\ & *A6 \ & - 81^F\ & *H\ & *R\ & *F1\ & *F2 \ & + 9^F\ & *F\ & *H \\
2 & 2 & 2 & 3 & 3 & 3 & 3 & 3 & 3 \\
*F2 \ & + 36^F\ & *H\ & *Q\ & *F2 \ & + 81^F\ & *H\ & *R\ & *F2 \ & + \ & H\ & *P\ & *F1 \ & + 6^H\ & *P\ & *Q \\
3 & 3 & 2 & 3 & 3 & 2 & 3 & 3 & 3 \\
F1 \ & + 9^H\ & *P\ & *R\ & *F1 \ & - 3^H\ & *P\ & *F1 \ & + 12^H\ & *P\ & *Q\ & *F1 \ & + 36^H\ & *P \\
3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
P\ & *Q\ & *R\ & *F1 \ & + 18^H\ & *P\ & *Q\ & *F1 \ & *A3 \ & - 18^H\ & *P\ & *Q\ & *F1 \ & + 27^H\ & *P\ & *R \\
3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
F1 \ & + 18^H\ & *P\ & *R\ & *F1 \ & *A5 \ & + 36^H\ & *P\ & *R\ & *F1 \ & *A6 \ & - 36^H\ & *P\ & *R\ & *F1 \\
3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
+ 2^H\ & *P\ & *F1 \ & + 8^H\ & *Q\ & *F1 \ & + 36^H\ & *Q\ & *R\ & *F1 \ & + 36^H\ & *Q\ & *F1 \\
3 & 3 & 2 & 3 & 3 & 2 & 3 & 3 & 3 \\
*A3 \ & - 24^H\ & *Q\ & *F1 \ & + 54^H\ & *Q\ & *R\ & *F1 \ & + 54^H\ & *Q\ & *R\ & *F1 \ & *A3 \ & + 36 \\
3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \\
*H\ & *Q\ & *R\ & *F1 \ & *A5 \ & + 72^H\ & *Q\ & *R\ & *F1 \ & *A6 \ & - 90^H\ & *Q\ & *R\ & *F1 \ & - 36^H\ & *Q}
\[3 F1 \ast A3 + 16H \ast Q \ast F1 + 27H \ast R \ast F1 + 54H \ast R \ast F1 \ast A5 +
3 2 3 3 3 3 3 3 2 3
108H \ast R \ast F1 \ast A6 - 81H \ast R \ast F1 + 54H \ast R \ast F1 \ast A3 \ast A6 - 54H \ast A5
3 2 2 3 3 2 3 3 3 3 2 2 2
\ast R \ast F1 \ast A5 - 108H \ast R \ast F1 \ast A6 + 54H \ast R \ast F1 + 9H \ast P \ast F1 +
2 2 2 2 2 2 2 2 2 2 2 2 2
36H \ast P \ast Q \ast F1 + 54H \ast P \ast R \ast F1 - 9H \ast P \ast F1 + 36H \ast Q \ast F1
2 2 2 2 2 2 2 2 2 2 2 2 2
- 108H \ast Q \ast R \ast F1 + 54H \ast Q \ast F1 \ast A3 - 36H \ast Q \ast F1 + 81H \ast R \ast F1
2 2 2 2 2 2 2 2 2 2 2 2 2
\ast F1 + 54H \ast R \ast F1 \ast A5 + 108H \ast R \ast F1 \ast A6 - 81H \ast R \ast F1 + 54H \ast P \ast F1
108H \ast Q \ast F1 + 162H \ast R \ast F1 + 162)}/314928

lhs:=h\ast f+h\ast h*2\ast f\ast f1/2+h\ast h*3*(f\ast f1*2/6+f\ast h*2*2\ast f2/6)+h\ast h*4*(f\ast h*3\ast f2/24+f\ast f1*3/24+f\ast h*2\ast f1*2\ast f2/6)+h\ast h*5*(f\ast h*3\ast f4*7+f\ast h*3\ast f1\ast f3+11\ast f\ast h*2\ast f1*2\ast f2+f\ast h*3*f2*2+f\ast f1*4)/120;

LHS :=
\[2 3 3 3 2 3 3 3 2 2
(F \ast H \ast (F \ast H \ast F3 + 4 \ast F \ast H \ast F1 \ast F2 + 4 \ast F \ast H \ast F2 + H \ast F1 + 4 \ast H \ast F1
+ 12 \ast H \ast F1 + 24))/24
\]

err:=lhs-rhs;

ERR :=
\[2 2 2 2 2 2 2 2 2
(81 \ast F \ast H * ( - 4 \ast F \ast H \ast P \ast F3 - 32 \ast F \ast H \ast Q \ast F3 - 108 \ast F \ast H \ast R \ast F3 + 27
2 2 2 2 2 2 2 2 2
\ast F \ast H \ast F3 - 12 \ast F \ast H \ast P \ast F1 \ast F2 - 72 \ast F \ast H \ast P \ast Q \ast F1 \ast F2 -
2 2 2 2 2 2 2 2 2
144 \ast F \ast H \ast P \ast R \ast F1 \ast F2 + 12 \ast F \ast H \ast P \ast F1 \ast F2 - 96 \ast F \ast H \ast Q \ast F1*
2 2 2 2 2 2 2 2 2
\ast F2 - 360 \ast F \ast H \ast Q \ast R \ast F1 \ast F2 - 180 \ast F \ast H \ast Q \ast F1 \ast F2 \ast A3 + 96 \ast F*
2 2 2 2 2 2 2 2 2
H \ast Q \ast F1 \ast F2 - 324 \ast F \ast H \ast R \ast F1 \ast F2 - 252 \ast F \ast H \ast R \ast F1 \ast F2 \ast A5
2 2 2 2 2 2 2 2 2
- 576 \ast F \ast H \ast R \ast F1 \ast F2 \ast A6 + 324 \ast F \ast H \ast R \ast F1 \ast F2 + 108 \ast F \ast H *
F1 \ast F2 - 36 \ast F \ast H \ast P \ast F2 - 144 \ast F \ast H \ast Q \ast F2 - 324 \ast F \ast H \ast R \ast F2 +
2 3 3 2 2 3 2 2 2
108 \ast F \ast H \ast F2 - 4 \ast H \ast P \ast F1 - 24 \ast H \ast P \ast Q \ast F1 - 36 \ast H \ast F \ast
\[ \begin{align*}
&3 \quad 2 \quad 2 \quad 3 \quad 2 \quad 2 \quad 3 \quad 2 \\
R*F1 + 12*H*P*F1 - 48*H*P*Q*F1 - 144*H*P*Q*R* \\
&3 \quad 2 \quad 3 \quad 2 \quad 3 \\
F1 - 2*H*P*Q*F1 *A3 + 72*H*P*Q*F1 - 108*H*P*R* \\
&3 \quad 2 \quad 3 \quad 2 \quad 3 \\
F1 - 72*H*P*R*F1 *A5 - 144*H*P*R*F1 *A6 + 144*H*P \\
&3 \quad 2 \quad 3 \quad 2 \quad 3 \quad 2 \quad 3 \\
*P*F1 - 8*H*P*F1 - 32*H*Q*F1 - 144*H*Q*R*F1 \\
&2 \quad 2 \quad 3 \\
*F1 *A3 + 96*H*Q*F1 - 216*H*Q*R*F1 \\
&2 \quad 3 \\
- 144*H*Q*R*F1 *A3 + 96*H*Q*F1 - 144*H*Q*R*F1 *A3 - 64*H*Q* \\
&2 \quad 3 \\
F1 *A6 + 360*H*Q*R*F1 + 144*H*Q*R*F1 *A3 + 144*H*Q* \\
&3 \quad 2 \quad 2 \quad 3 \\
F1 *A6 + 324*H*P*F1 - 216*H*P*F1 *A3*A6 + 216*H* \\
&3 \quad 2 \quad 3 \quad 2 \quad 3 \quad 2 \quad 3 \\
*P*F1 *A5 + 432*H*P*R*F1 *A6 - 216*H*P*R*F1 + 27*H*F1 \\
&2 \quad 2 \quad 3 \\
- 36*H*P*F1 - 144*H*P*Q*F1 - 216*H*P*R*F1 + 36*H \\
&2 \quad 2 \quad 2 \quad 2 \\
*P*F1 - 144*H*P*Q*F1 - 432*H*P*R*F1 - 216*H*Q*R*F1 \\
&2 \quad 3 \quad 2 \\
A3 + 144*H*Q*R*F1 - 324*H*P*F1 - 216*H*P*R*F1 *A5 - \\
&2 \quad 2 \quad 2 \\
432*H*P*R*F1 *A6 + 324*H*P*R*F1 + 108*H*P*F1 - 216*P*F1 \\
&2 \quad 2 \\
- 432*Q*F1 - 648*R*F1 + 324*F1) / 52488
\end{align*} \]
Appendix 4

program stabgmrk

!-This program is used to plot the stability
!-regions of the GM-Runge-Kutta formulae
!-of order \(k=1,2,3,4\). The boundaries of the
!-regions are plotted by using a number of
!-subroutines from the GINOSURF library.

external f1,f2,f3,f4

print *, 'select device:
  1 for t4010, 2 for c1051n, 3 for se281'
read(5,*) isel
if(isel.eq.1) call t4010
if(isel.eq.2) call c1051n
if(isel.eq.3) call se281

!-piccle
!-window(2)
!-levels(0.0,1.0)
!-pensel (1,0,3,3)
!
!funcon (-4.0,1.0,-3.5,3.5,f1,2,1)
!funcon (-4.0,1.0,-3.5,3.5,f2,2,1)
!funcon (-4.0,1.0,-3.5,3.5,f3,2,1)
!funcon (-4.0,1.0,-3.5,3.5,f4,2,1)
!!devend
!!stop
!!end

!-Stability functions, \(f1,f2,f3,f4\).

function f1(x,y)
complex z
z=cmplx(x,y)
f1=abs(1.+z)
return
end

function f2(x,y)
complex z
z=cmplx(x,y)
f2=abs(1.+z*sqrt(1.+z))
return
end

function f3(x,y)
complex z,fz
z=cmplx(x,y)
fz=1.+z*(1.+2*z/3.+5*z**2/18.)**0.75
f3=abs(fz)
return
function f4(x,y)
complex z,fz,t1,t2,t3
z=cmplx(x,y)
a1=2/5.
a2=3/5.
a3=-15815/4056.
a4=1/6.
a5=5833/5869.
a6=-2567/16557.
p=3838/6991.
q=-1266/18805.
r=2683/6924.
t1=1.+z*a1
t2=1.+z*a2+z*a3*t1
t3=1.+z*a4+z*a5*t1+z*a6*t2
fz=1.+z*t1**p*t2**q*t3**r
f4=abs(fz)
return
end
Appendix 5

program agm

c-----This program solves a single O.D.E. by using the

c-----Arithmeto-Geometric Mean (AGM) formula

c-----with error control

implicit real*8 (a-h,o-z)
integer 1(2)
external f,soln

c-----The following parameters are used to define the formula.
al=0.101263277151d+01
a2=0.457032814888d+00
a3=0.148314837406d+00
a4=0.963574868163d+00
a5=-0.585574172123d-01
a6=0.125329257761d+00
p=-0.363095578546d+01
q=0.983484336345d+00
r=0.347599002931d+01
w2=-0.227594644336d+01
w3=0.865485319704d+00
w4=0.221360288639d+01

c-----Defining the problem:
c-----errtol = error tolerance (preset accuracy requirement);
c-----xo = initial value of x; xend = last value of x;
c-----yo = initial value of y at xo;
c-----h = initial stepsize; k = printing stepnumber.
errtol=0.00001
xo=0.0
xend=1.0
yo=1.0
h=1.0
k=1

c-----Beginning of the computational work.
x=xo
y=yo
icount=0
write(1,19)errtol
19 format(5x,'Error tolerance =',d6.111)
write(1,21)x,y
21 format(f19.5,d15.7)
continue
call cpu_time_and_paging_(m,l,n)
itime1=1(2)
do 30 i=1,k

ak1=f(x,y)
ak2=f(x+a1*h,y+h*a1*ak1)
ak3=f(x+(a2+a3)*h,y+h*a2*ak1+h*a3*ak2)

10 continue
ak4=f(x+(a4+a5+a6)*h,y+h*(a4*ak1+a5*ak2+a6*ak3))

c

c----- yam = solution obtained by using AM formula;
c----- ygm = solution obtained by using GM formula;
c----- errest = estimated error in yam:
yam=y+h*((1.-w2-w3-w4)*ak1+w2*ak2+w3*ak3+w4*ak4)
ygm=y+sign(1.,f(x,y))*h*abs(ak1)**(1.-p-q-r)*abs(ak2)**p
    *abs(ak3)**q*abs(ak4)**r
errest=(yam-ygm)/2.09546256

c

c----- halve the stepsize if error is too large
    if(abs(errest).gt.errtol) then
      h=h/2.
      write(1,99)h
    end if

99  format(/1x,'h=',f8.5/) go to 11 end if

c

c----- or safely double the stepsize if error is too small,
c----- and avoid repetition of changing stepsize locally.
    if(abs(errest).lt.errtol/32.) then
      icount=icount+1
      if(icount.ge.2) go to 22
      h=2.*h
      write(1,99)h
    end if
    go to 11

22  icount=0
    y=yam
    x=x+h
30  continue
    call cpu_time_and_paging_(m,l,n)
    itime2=1(2)
    itime=itime2-itime1
    exact=soln(x)
    err=exact-y
    write(1,20)itime,x,exact,y,err,errest
20  format(i10,f9.5,4d15.7)
    if(x.lt.xend) go to 10
    stop
end

c

c----- The function which defines the problem:
    function f(x,y)
      implicit real*8 (a-h,o-z)
      f=-2*y
    return
end

c

c----- The function which defines the exact solution:
    function soln(x)
      implicit real*8 (a-h,o-z)
      soln=exp(-2*x)
    return
end
Appendix 6

comment:
This program is used to determine the coefficients \( c(i) \)
in the Hyman's leap-frog IMS formula with variable
stepsize (characterised by \( r=\frac{h(n+1)}{h(n)} \)). Any order
can, therefore, be achieved by using the related formula.
The program is written for use in REDUCE system installed
on the VAX 11/750 C.P.U. computer at Loughborough
University of Technology;

\begin{verbatim}
\%------------------------------------------------------------------------
\% n:=8;
N := 8
% [set maximum value of i]
array a(n,n+1),c(n);
on factor a,c;
% [ask the system to factor expressions
% into factors with integer coefficients].
% In the following, jfac is to be meant j!
for j:=3:n do << jfac:=for m:=1:j product m;
a(1,j):=(-1)**j/(jfac*r**(j-2)) >>;

for j:=4:n do << jfac:=for m:=1:j product m;
a(2,j):=(-1)**j*(1-j*(1+r))/(jfac*r**(j-3)*(2+3*r)) >>;

a(2,3):=1/6;
A(2,3) := 1/6
for i:=3:n do begin
a(i,i+1):=1/(for m:=1:i+1 product m);
c(i):=(a(i,i+1)-a(i-1,i+1))/(a(i-1,i)-a(i-2,i));
write "  c("i",")=",c(i);
for j:=i+2:n do << a(i,j):=a(i-1,j)+c(i)*(a(i-1,j-1)-a(i-2,j-1)) >>;
end;

2
\text{c(3)}=(3*(R + 1))/(4*(3*R + 2))
\text{c(4)}=(18*R + 18*R -1)/(30*R*(3*R + 2))
\text{c(5)}=(54*R + 216*R + 315*R + 190*R + 30*R - 6*R + 1)/(6*R
\text{*(54*R + 198*R + 267*R + 154*R + 29*R - 2))}
\text{c(6)}=(29160*R + 233280*R + 805140*R + 1555416*R +
1817100*R + 1281570*R + 500634*R + 80644*R + 2873*R
\end{verbatim}
\[ \begin{align*}
3 & + 6238R + 2092R - 148R + 4R = \frac{11}{(70R^2 + 2916R + 2092R - 148R + 4R)} \\
10 & + 22356R + 74196R + 138618R + 158175R + 41R - 2) \\
6 & + 111012R + 44839R + 8096R - 167R - 80R + 41R - 2) \\
4 & + \frac{6238R + 2092R - 148R + 4R}{70R + 2916R + 10987R + 22356R + 74196R + 138618R + 158175R + 6238R + 2092R - 148R + 4R} \\
24 & + \frac{111012R + 44839R + 8096R - 167R - 80R + 41R - 2)}{70R + 2916R + 10987R + 22356R + 74196R + 138618R + 158175R + 6238R + 2092R - 148R + 4R} \\
\end{align*} \]
10 9 8
5350051458*R + 2502026712*R + 834616050*R +
7 6 5 4
182177480*R + 20823398*R + 260343*R + 6586*R +
3 2 24
60376*R + 3504*R - 976*R + 32)/(9*(21257640*R +
23 22 21
340122240*R + 2533202100*R + 11641943376*R +
20 19
36885430242*R + 85227809904*R + 148090827825*
18 17 16
R + 196539481836*R + 200154747870*R +
15 14
155683959822*R + 91264133619*R + 39572810370*
13 12 11
R + 12629329921*R + 3220230328*R + 806759983*
10 9 8 7
R + 153981016*R - 28274505*R - 28564386*R -
6 5 4 3
5477655*R + 248898*R + 62661*R - 23540*R + 304*
2
R + 136*R - 5))
end;
Appendix 7

comment: This is a REDUCE computer program used to expand the square, ux**2, of a Fourier series representation of the function ux which contains 6 sine and cosine terms;

operator c,s,f,g;

ux:= for k:=1 step 1 until 6 sum g(k)*c(k)-f(k)*s(k);

UX :=
C(6)*G(6) + C(5)*G(5) + C(4)*G(4) + C(3)*G(3) + C(2)*G(2) + C(1)
*G(1) - F(6)*S(6) - F(5)*S(5) - F(4)*S(4) - F(3)*S(3) - F(2)*S(2)
- F(1)*S(1)
for all m,n let s(m)*s(n)=(c(m-n)-c(m+n))/2;

for all m,n let c(m)*c(n)=(c(m-n)+c(m+n))/2;

for all m,n let s(m)*c(n)=(s(m-n)+s(m+n))/2;

for all m let c(-m)=c(m);

for all m let s(-m)=-s(m);

for all m let c(m)*c(m)=(1+c(2*m))/2;

for all m let s(m)*s(m)=(1-c(2*m))/2;

let c(0)=1;

let s(0)=0;

let f(1)=f1;

let f(2)=2*f2;

let f(3)=3*f3;

let f(4)=4*f4;

let f(5)=5*f5;
let $f(6)=6*f_6$;

% let $g(1)=g_1$;

let $g(2)=2*g_2$;

let $g(3)=3*g_3$;

let $g(4)=4*g_4$;

let $g(5)=5*g_5$;

let $g(6)=6*g_6$;

$UXSQ:=ux**2$;

$UXSQ :=$

\[
\]
\[ \begin{align*}
g2g5 + 36c(3)g3g6 - c(2)f1 + 6c(2)f1f3 + 16c(2)f2 + 2 \\
&f4 + 30c(2)f3f5 + 48c(2)f4f6 + c(2)g1 + 6c(2)g1g3 + 16c(2)g2g4 + 30c(2)g3g5 + 48c(2)g4g6 + 4c(1)f1 + \\
f2 + 12c(1)f2f3 + 24c(1)f3f4 + 40c(1)f4f5 + 60c(1)f5f6 + 4c(1)g1 + 12c(1)g1g2 + 24c(1)g2g3 + 40c(1)g3g4 + 40c(1)g4g5 + 60c(1)g5g6 - 72s(12)f6g6 - 60s(11)f5g6 - \\
&60s(11)f6g5 - 48s(10)f4g6 - 50s(10)f5g5 - 48s(10)f6g4 - 36s(9)f3g6 - 40s(9)f4g5 - 40s(9)f5g4 - 36s(9)f6g3 - 24s(8)f2g6 - 30s(8)f3g5 - 32s(8)f4g4 - 30 \\
s(8)f5g3 - 24s(8)f6g2 - 12s(7)f1g6 - 20s(7)f2g5 - 24s(7)f3g4 - 24s(7)f4g3 - 20s(7)f5g2 - 12s(7)f6g1 - 10s(6)f1g5 - 16s(6)f2g4 - 18s(6)f3g3 - 16s(6)f4g2 - 10s(6)f5g1 - 8s(5)f1g4 + 12s(5)f1g6 - 12s(5) \\
f2g3 - 12s(5)f3g2 - 8s(5)f4g1 - 12s(5)f6g1 - 6s(4)f1g3 + 10s(4)f1g5 - 8s(4)f2g2 + 24s(4)f2g6 - 6s(4)f3g1 - 10s(4)f5g1 - 24s(4)f6g2 - 4s(3)f1g2 + 8 \\
s(3)f1g4 - 4s(3)f2g1 + 20s(3)f2g5 + 36s(3)f3g6 - 8s(3)f4g1 - 20s(3)f5g2 - 36s(3)f6g3 - 2s(2)f1g1 + 6s(2)f1g3 + 16s(2)f2g4 - 6s(2)f3g1 + 30s(2)f3g5 - 16s(2)f4g2 + 48s(2)f4g6 - 30s(2)f5g3 - 48s(2)f6g4 + 4s(1)f1g2 - 4s(1)f2g1 + 12s(1)f2g3 - 12s(1) \\
f3g2 + 24s(1)f3g4 - 24s(1)f4g3 + 40s(1)f4g5 - 40s \\
(1)f5g4 + 60s(1)f5g6 - 60s(1)f6g5 + f1 + 4f2 + 9* \\
&f3 + 16f4 + 25f5 + 36f6 + g1 + 4g2 + 9g3 + 16g4 + 2 \\
&25g5 + 36g6 )/2 \end{align*} \]
A NEW THIRD ORDER KUTTA METHOD FOR
SOLVING ORDINARY DIFFERENTIAL EQUATIONS

D.J. Evans & B.B. Sanugi
Department of Computer Studies
Loughborough University of Technology
Loughborough, Leicestershire, U.K.

COMPUTER STUDIES 281
May, 1986.
ABSTRACT

A new third order Runge-Kutta method based on the principle of geometric means instead of the usual arithmetic means is presented and the numerical results given confirm the accuracy obtained.

1. INTRODUCTION

The third order classical Runge-Kutta formula is well known and given by,

\[
\begin{aligned}
    k_1 &= f(y_n) \\
    k_2 &= f(y_n + ha_1k_1) \\
    k_3 &= f(y_n + ha_2k_1 + ha_3k_2) \\
    y_{n+1} &= y_n + h(\omega_1k_1 + \omega_2k_2 + \omega_3k_3)
\end{aligned}
\]

(1)

Kutta's third order rule, for example, uses 
\(a_1=1/2, \ a_2=-1, \ a_3=2, \ \omega_1=1/6, \ \omega_2=2/3, \ \omega_3=1/6,\)
i.e. equation (2) is written as,

\[
y_{n+1} = y_n + \frac{h}{6}(k_1 + 4k_2 + k_3)
\]

(3)

It is also possible to obtain a formula of the form

\[
y_{n+1} = y_n + \frac{h}{4}(k_1 + 2k_2 + k_3)
\]

(4)

and make an adjustment of the parameters to attain third order accuracy. Then Equation (4) can be rewritten as,

\[
y_{n+1} = y_n + \frac{h}{2}(\frac{k_1 + k_2}{2} + \frac{k_2 + k_3}{2})
\]

(5)

2. DERIVATION OF NEW METHOD

Now by substituting the arithmetic means of \(k_i, \ i=1,2,3\) in (5) with their geometric mean we obtain a new formula of the form,

\[
y_{n+1} = y_n + \frac{h}{2} (\sqrt[k_1]{k_2} + \sqrt[k_2]{k_3})
\]

(6)

and adjust the parameters \(a_i, \ i=1,2,3\) so that equation (6) will have the highest accuracy possible.

Using the REDUCE symbolic computer program as before [1], we obtain the following three equations of condition,
Solving these three equations simultaneously we obtain the values,
\[ a_1 = \frac{2}{3}, \quad a_2 = -\frac{1}{2}, \quad a_3 = \frac{7}{6} \]

Thus, the new method can be written as follows,
\[
\begin{align*}
 k_1 &= f(x_n, y_n) \\
 k_2 &= f(x_n + \frac{2h}{3}, y_n + \frac{2}{3}h k_1) \\
 k_3 &= f(x_n + \frac{2h}{3}, y_n - \frac{1}{2}h k_1 + \frac{7}{6}h k_2)
\end{align*}
\]

and
\[
y_{n+1} = y_n + \frac{h}{2}(\sqrt{k_1 k_2} + \sqrt{k_2 k_3})
\]

3. **ERROR ANALYSIS**

By substituting the values of \(a_1, a_2, a_3\) again in the REDUCE symbolic computer program, and this time showing all the terms up to \(h^4\) we obtain the truncation error in the formula to be,
\[
LTE = \frac{3h^4}{640} \left[ f_{yy} f_{yy} - 2 f_{fyy} + 7 f_{f}^3 \right],
\]
thus confirming the third order accuracy of the method.

4. **NUMERICAL EXAMPLE**

Solving \(y' = -y, \ y(0) = 1\) in \(0 \leq x \leq 1\), using \(h = 0.1\), we obtain the following results:

<table>
<thead>
<tr>
<th>(x)</th>
<th>(y_{Exact})</th>
<th>(y_{Num.})</th>
<th>(Error)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.10</td>
<td>0.90483748E+00</td>
<td>0.90483478E+00</td>
<td>0.2711474E-05</td>
</tr>
<tr>
<td>0.20</td>
<td>0.8187308E+00</td>
<td>0.8187258E+00</td>
<td>0.4916619E-05</td>
</tr>
<tr>
<td>0.30</td>
<td>0.7408182E+00</td>
<td>0.7408116E+00</td>
<td>0.666492E-05</td>
</tr>
<tr>
<td>0.40</td>
<td>0.6703200E+00</td>
<td>0.6703120E+00</td>
<td>0.8047910E-05</td>
</tr>
<tr>
<td>0.50</td>
<td>0.6065307E+00</td>
<td>0.6065216E+00</td>
<td>0.9098352E-05</td>
</tr>
<tr>
<td>0.60</td>
<td>0.5488116E+00</td>
<td>0.5488018E+00</td>
<td>0.9880673E-05</td>
</tr>
<tr>
<td>0.70</td>
<td>0.4965853E+00</td>
<td>0.4965749E+00</td>
<td>0.1042790E-04</td>
</tr>
<tr>
<td>0.80</td>
<td>0.4493290E+00</td>
<td>0.4493182E+00</td>
<td>0.1078387E-04</td>
</tr>
<tr>
<td>0.90</td>
<td>0.4065697E+00</td>
<td>0.4065587E+00</td>
<td>0.1097874E-04</td>
</tr>
<tr>
<td>1.00</td>
<td>0.367879E+00</td>
<td>0.3678684E+00</td>
<td>0.1103946E-04</td>
</tr>
</tbody>
</table>
5. **REFERENCE**

A NEW 4TH ORDER RUNGE KUTTA METHOD
FOR INITIAL VALUE PROBLEMS

D.J. Evans & B.B. Sanugi
Department of Computer Studies
Loughborough University of Technology
Loughborough, Leicestershire, U.K.

ABSTRACT
A new 4th order Runge Kutta method for solving initial value problems is derived by replacing the arithmetic means in the formula \( y_{n+1} = y_n + \frac{h}{6} (k_1 + 2k_2 + 2k_3 + k_4) \)
where \( k_i = f(y_n + h \cdot a_i \cdot k_{ij}) \) etc, by their Geometric means i.e. \( (k_1 k_2)/2 = \sqrt{k_1 k_2} \) etc. to yield initially a low order accuracy formula. However by re-comparing the Taylor series expansions of \( k_1, k_2, k_3 \) and \( k_4 \) in terms of the functional derivatives and the \( a_{ij} \) parameters, a fourth order accuracy formula is obtained which is confirmed by numerical experimentation.

INTRODUCTION
We will now denote \( a_{11} = a_1, a_{21} = a_2, a_{22} = a_3, \ldots \), etc. for simplicity so that the well known 4th order R.K. formula is given by

\[
\begin{align*}
  k_1 &= f(y_n) \\
  k_2 &= f(y_n + h \cdot a_1 \cdot k_{11}) \\
  k_3 &= f(y_n + h \cdot a_2 \cdot k_{12} + h \cdot a_3 \cdot k_{22}) \\
  k_4 &= f(y_n + h \cdot a_4 \cdot k_{14} + h \cdot a_2 \cdot k_{32} + h \cdot a_6 \cdot k_{36}) \\
  y_{n+1} &= y_n + \frac{h}{6} (k_1 + 2k_2 + 2k_3 + k_4)
\end{align*}
\]  
(1)

where 4th order accuracy is obtained by choosing, e.g.,

\[
\begin{align*}
  a_1 &= \frac{1}{3}, a_2 = 0, a_3 = \frac{1}{3}, a_4 = 0, a_5 = 0, a_6 = 1.
\end{align*}
\]  
(3)

Now equation (2) can be written as follows:

\[
y_{n+1} = y_n + \frac{h}{3} \left( \frac{k_1 + k_2}{2} + \frac{k_2 + k_3}{2} + \frac{k_3 + k_4}{2} \right).
\]

and substituting the arithmetic means of \( k_i \), \( i = 1, 2, 3, 4 \) with their geometric means we obtain a new formula similar to equation (2) of the form,

\[
y_{n+1} = y_n + \frac{h}{3} (\sqrt{k_1 k_2} + \sqrt{k_2 k_3} + \sqrt{k_3 k_4})
\]
(4)

By experiment, using (4) with the set of parameters (3) leads to a low accuracy formula, i.e. of order 2. Therefore these parameters are not suitable for use with formula (4).
2. A FOURTH ORDER RUNGE-KUTTA FORMULA

Now we will find the values of the parameters \( a_i, i = 1, \ldots, 6 \) that will cause formula (4) to have an accuracy of order 4.

The Taylor series expansion of \( k_1, k_2, k_3, \) and \( k_4 \) in terms of the functional derivatives and the six parameters \( a_1, a_2, \ldots, a_6 \) are given fully in Appendix 1 where expansions up to the fourth order are included.

To evaluate the RHS of equation (4) we use a binomial expansion strategy with the help of the REDUCE formula manipulation package. We note that the binomial expansion of \((1+x)^\frac{1}{2}\) is given by,

\[
(1+x)^\frac{1}{2} = 1 + \frac{1}{2}x - \frac{1}{8}x^2 + \frac{1}{16}x^3 + 
\]

(5)

First, we rewrite \( \sqrt{k_1k_2} \) in the form of

\[
f(1+x)^\frac{1}{2}
\]

(6)

where \( x \) is given by

\[
x = \frac{k_1k_2}{f^2} - 1
\]

(7)

Evaluating (6) with \( x \) as given by (7) and by using (5), we obtain \( \sqrt{k_1k_2} \) given in ascending powers of \( h \). In this case, we include \( h \) only to the 4th power. A similar technique is used to obtain \( \sqrt{k_2k_3} \) and \( \sqrt{k_3k_4} \).

By comparing the RHS of (4) with the Taylor series expansion for \( y(x_{n+1}) \) we obtain the following six equations of conditions, where we have set,

\[ a_2 + a_3 = \frac{1}{3} \text{ and } a_4 + a_5 + a_6 = 1. \]

(Note: we may choose other values if required).

These are:

\[
h^2 f_{yy} : -192a_1 + 96 = 0 \quad (8a)
\]

\[
h^3 f_{y}^2 : 108 - 48a_6 - 24a_1 - 96a_1a_5 - 192a_1a_3 + 48a_1^2 = 0 \quad (8b)
\]

\[
h^3 f_{yy}^2 : 24 - 96a_1^2 = 0 \quad (8c)
\]

\[
h^4 f_{y}^3 : 18 + 12a_6 + 3a_1 + 24a_5 - 96a_1a_3 + 6a_6^2 - 48a_1a_3^2 - 24a_1^3 = 0 \quad (8d)
\]

\[
h^4 f_{yy}^2 : 108 - 60a_6 - 6a_1 - 96a_1a_5 - 96a_1a_3 - 12a_1^2 - 48a_1a_5^2 - 96a_1a_3^2 + 48a_1^3 = 0 \quad (8e)
\]

and

\[
h^4 f_{yyy} : 4 - 32a_1^3 = 0 \quad (8f)
\]

Now from (8a) we have the value,
\[ a_1 = \frac{96}{192} = \frac{1}{4} \]

whilst from (8f) we have,
\[ 32a_1^3 = 4 \implies a_1^3 = \frac{4}{32} = \frac{1}{8} \]
to give \( a_1 = \frac{1}{2} \) (which is consistent)

Also from (8c) we have
\[ 96a_1^2 = 24 \implies a_1^2 = \frac{24}{96} = \frac{1}{4} \]
to give \( a_1 = \frac{1}{2} \) (which is consistent)

while from (8b) we have
\[ 108 - 48a_6 - 24(\frac{1}{4}) - 96(\frac{1}{4})a_5 - 192(\frac{1}{4})a_3 + 48(\frac{1}{4})^2 = 0 \]
or the relation
\[ 8a_3 + 4a_5 + 4a_6 = 9 \] (9)
and from (8e) we have
\[ 108 - 60a_6 - 6(\frac{1}{4}) - 96(\frac{1}{4})a_5 - 96(\frac{1}{4})a_3 - 12(\frac{1}{4})^2 - \\
48(\frac{1}{4})^2 a_5 - 96(\frac{1}{4})^2 a_3 + 48(\frac{1}{4})^3 = 0 \]
or the relation
\[ 6a_3 + 5a_5 + 5a_6 = 9 \] (10)
and finally from (8d) we have
\[ 18 + 12a_6 + 3(\frac{1}{4}) + 24(\frac{1}{4})a_5 - 96(\frac{1}{4})a_3 a_6 + 6(\frac{1}{4})^2 - \\
48(\frac{1}{4})^2 a_3 - 24(\frac{1}{4})^3 = 0 \]
or the relation
\[ 2a_3 - 2a_5 - 2a_6 + 8a_3 a_6 = 3 \] (11)

Solving (9), (10) and (11) simultaneously we obtain the values \( a_3 = 9/16, \]
\( a_5 = 5/24, a_6 = 11/12. \)

Since initially we choose,
\[ a_2 + a_3 = \frac{1}{2}, \text{ we have } a_2 = - \frac{1}{16} \]
and
\[ a_4 + a_5 + a_6 = 1, \text{ we have } a_4 = - \frac{1}{6}. \]

Therefore the six parameters are given by,
\[
\begin{align*}
  a_1 &= \frac{1}{4} \\
  a_2 &= -\frac{1}{16} \\
  a_3 &= \frac{9}{16} \\
  a_4 &= -\frac{1}{8} \\
  a_5 &= \frac{5}{24} \\
  a_6 &= \frac{11}{12}
\end{align*}
\] (12)

Thus, we predict that these parameters will result in formula (4) to have
an accuracy of order 4.
Thus, this new method can be written as follows

\[
\begin{align*}
k_1 &= f(x_n, y_n) \\
k_2 &= f(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1) \\
k_3 &= f(x_n + h, y_n + \frac{h}{16}(-k_1 + 9k_2)) \\
k_4 &= f(x_n + h, y_n + \frac{h}{24}(-3k_1 + 5k_2 + 22k_3)) \\
y_{n+1} &= y_n + \frac{h}{3}(k_1 + 2k_2 + k_3)
\end{align*}
\]

(13)

3. ERROR ANALYSIS

We also use the same REDUCE program to compute the error terms of the resulting formula. This time we use up to the 5th order terms in \( h \) in the error terms and make a substitution for the values of \( a_1, a_2, \ldots, a_6 \) as given by (12). Since these parameters cause the error terms up to order 4 to vanish, we have remaining only the 5th order term to represent the LTE of the formula. This is given by,

\[
LTE = \frac{h^5}{184320} \left[ -64f_{yy}^4f_{yyy} + 208f_{yy}^2f_{yyyy} - 374f_{yy}^3f_{yy}^2 - 1974f_{yy}^2f_{yy}^2 + 3561ff_{y}^4 \right]
\]

(14)

thus confirming the 4th order accuracy of the method.

4. NUMERICAL EXAMPLE

We solve the initial value problem

\[
y' = -y, \quad y(0) = 1
\]

by using the two different versions of the 4th order RK formula, i.e. the classical 4th order formula, and the new derived formula, and compare the results obtained in Table 1.
### TABLE 1

#### Standard formula equations (1)–(3)

<table>
<thead>
<tr>
<th>cpu t</th>
<th>x</th>
<th>Exact Solution</th>
<th>Numerical Solution</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>2768</td>
<td>0.10</td>
<td>0.9048374E+00</td>
<td>0.9048375E+00</td>
<td>-0.8537769E-07</td>
</tr>
<tr>
<td>2571</td>
<td>0.20</td>
<td>0.8187308E+00</td>
<td>0.8187309E+00</td>
<td>-0.1447658E-06</td>
</tr>
<tr>
<td>2473</td>
<td>0.30</td>
<td>0.7408182E+00</td>
<td>0.7408184E+00</td>
<td>-0.2046044E-06</td>
</tr>
<tr>
<td>2594</td>
<td>0.40</td>
<td>0.6703200E+00</td>
<td>0.6703203E+00</td>
<td>-0.2399892E-06</td>
</tr>
<tr>
<td>2123</td>
<td>0.50</td>
<td>0.6065307E+00</td>
<td>0.6065309E+00</td>
<td>-0.2755226E-06</td>
</tr>
<tr>
<td>2392</td>
<td>0.60</td>
<td>0.5488116E+00</td>
<td>0.5488119E+00</td>
<td>-0.2975112E-06</td>
</tr>
<tr>
<td>2901</td>
<td>0.70</td>
<td>0.4965858E+00</td>
<td>0.4965856E+00</td>
<td>-0.3166163E-06</td>
</tr>
<tr>
<td>2532</td>
<td>0.80</td>
<td>0.4493290E+00</td>
<td>0.4493293E+00</td>
<td>-0.3270209E-06</td>
</tr>
<tr>
<td>2183</td>
<td>0.90</td>
<td>0.4065697E+00</td>
<td>0.4065700E+00</td>
<td>-0.3314901E-06</td>
</tr>
<tr>
<td>2220</td>
<td>1.00</td>
<td>0.3678794E+00</td>
<td>0.3678799E+00</td>
<td>-0.3315419E-06</td>
</tr>
</tbody>
</table>

#### RK-new formula equation (13)

<table>
<thead>
<tr>
<th>cpu t</th>
<th>x</th>
<th>Exact Solution</th>
<th>Numerical Solution</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>4327</td>
<td>0.10</td>
<td>0.9048374E+00</td>
<td>0.9048376E+00</td>
<td>-0.1993220E-06</td>
</tr>
<tr>
<td>3651</td>
<td>0.20</td>
<td>0.8187308E+00</td>
<td>0.8187311E+00</td>
<td>-0.3509680E-06</td>
</tr>
<tr>
<td>3758</td>
<td>0.30</td>
<td>0.7408182E+00</td>
<td>0.7408187E+00</td>
<td>-0.4844737E-06</td>
</tr>
<tr>
<td>3321</td>
<td>0.40</td>
<td>0.6703200E+00</td>
<td>0.6703206E+00</td>
<td>-0.5776375E-06</td>
</tr>
<tr>
<td>3361</td>
<td>0.50</td>
<td>0.6065307E+00</td>
<td>0.6065313E+00</td>
<td>-0.6574187E-06</td>
</tr>
<tr>
<td>3095</td>
<td>0.60</td>
<td>0.5488116E+00</td>
<td>0.5488123E+00</td>
<td>-0.7121759E-06</td>
</tr>
<tr>
<td>2756</td>
<td>0.70</td>
<td>0.4965833E+00</td>
<td>0.4965861E+00</td>
<td>-0.7543546E-06</td>
</tr>
<tr>
<td>2725</td>
<td>0.80</td>
<td>0.4493290E+00</td>
<td>0.4493297E+00</td>
<td>-0.7796861E-06</td>
</tr>
<tr>
<td>2746</td>
<td>0.90</td>
<td>0.4065697E+00</td>
<td>0.4065705E+00</td>
<td>-0.7922771E-06</td>
</tr>
<tr>
<td>2825</td>
<td>1.00</td>
<td>0.3678794E+00</td>
<td>0.3678802E+00</td>
<td>-0.7948056E-06</td>
</tr>
</tbody>
</table>

### REFERENCES

The expansion of \( k_i \), \( i=1,2,3,4 \) in terms of the functional derivatives and the parameters \( a_i \), \( i=1,2,3,4,5,6 \) is obtained as follows:

By right, \( f \) is a function of \( x \) and \( y \) so that

\[
k_1 = f(x_n, y_n)
\]

\[
k_2 = f(x_n + \beta_1 h, y_n + \alpha_1 k_1)
\]

\[
k_3 = f(x_n + \beta_2 h, y_n + \alpha_2 k_1 + \alpha_3 k_2)
\]

\[
\vdots
\]

etc.

where \( \beta_1 = a_1 \)

\( \beta_2 = a_2 + a_3 \)

\( \vdots \)

etc.

so that it is not necessary to find \( \beta_1, \beta_2, \beta_3 \) initially but they can be found once the \( a_1, a_2, a_3 \ldots \) are determined. Therefore, it is sufficient to consider \( f \) as a function of \( y \) only as it considerably simplifies the accompanying algebra.

Now the expression of the Taylor's series of \( k_1, k_2, k_3 \) and \( k_4 \) in terms of the functional derivatives is as follows:

\[
k_1 = f(y_n) = f
\]

\[
k_2 = f(y_n + \alpha_1 k_1)
= f + \frac{h^2}{2} a_1 f_{yy} + \frac{h^3}{6} a_1^2 f_{yyy} + \frac{h^4}{24} a_1^3 f_{yyyy} + \ldots
\]  

\[
k_3 = f(y_n + \alpha_2 k_1 + \alpha_3 k_2)
= f + (a_2 + a_3) f + \frac{h^2}{2} a_1 a_2 f_{yy} + \frac{h^3}{6} a_1^2 a_2 f_{yyy} + \frac{h^4}{24} a_1^3 a_2 f_{yyyy}
\]

\[
k_4 = f(y_n + \alpha_4 k_1 + \alpha_5 k_2 + \alpha_6 k_3)
= f + h(a_4 + a_5 + a_6) f + \frac{h^2}{2} (a_5 a_1 + a_6 a_2 + a_6 a_3) f + \frac{h^3}{6} (a_5^2 + a_5 a_6 + a_5 a_6 + a_6 a_3) f + \frac{h^4}{24} (a_5^3 + a_5 a_6 + a_5 a_6 + a_6 a_3) f_{yyyy}
\]
By writing \( k_2 \) in the form,
\[
k_2 = f[1+(h^2 a_1^2 f_{yy} + \ldots)]
\]  
we can derive,
\[
k_1 k_2 = f^2[1+(h a_1 f_y + h^2 a_1^2 f_{yy} + \ldots)].
\]  
Then,
\[
\sqrt{k_1 k_2} = f[1+(h a_1 f_y + h^2 a_1^2 f_{yy} + \ldots)]^{1/2}
\]  
or
\[
\sqrt{k_1 k_2} = f[(1+x)^{1/2}],
\]  
where \( x = h a_1 f_y + h^2 a_1^2 f_{yy} + \ldots \)

which is a suitable form for hand computation but tedious.

Equivalently we can obtain \( x \) by rearranging (Al.9), i.e.,
\[
x = (k_1 k_2 / f^2 - 1),
\]  
and with the aid of the REDUCE symbol manipulation package since \( k_1 \) and \( k_2 \) are given a similar answer to (Al.10) is obtained.

By using the expansion,
\[
(1+x)^{1/2} = 1 + \frac{1}{2} x - \frac{1}{8} x^2 + \ldots
\]  
we can finally obtain,
\[
f(1+x)^{1/2} = \sqrt{k_1 k_2}
\]  
in terms of \( f, f_y, f_{yy}, \ldots \).

Similarly for the terms \( \sqrt{k_2 k_3} \) and \( \sqrt{k_3 k_4} \).

Substituting into the equation
\[
y_{n+1} = y_n + \frac{h}{3} (\sqrt{k_1 k_2} + \frac{h}{2} a_1^2 f_{yy} + \sqrt{k_2 k_3} + \sqrt{k_3 k_4})
\]  
we obtain an equation of the form,
\[
y_{n+1} = y_n + h f + h^2 f_y [a_1] + h^3 f f_y [8 a_6 + 3 a_1 + 12 a_1 a_3 + 24 a_1 a_3 - 8 a_1^2]
\]  
\[+ h^3 f f_{yy} [4 a_1^2] + h^4 f f_{yy} [\ldots]
\]  
\[+ h^4 f^2 f_{yy} [\ldots] + h^4 f f_y [\ldots]
\]  
where the terms in the square brackets are some functions of the parameters \( a_1, a_2, \ldots \).

Now the Taylor series expansion for \( y(x_{n+1}) \) where \( y' = f(y) \) and independent of \( x \), by successive differentiations can be obtained in the form,
\[ y^i = f \]
\[ y^{ii} = ff_y \]
\[ y^{iii} = ff_y^2 + f^2f_{yy} \]
\[ y^{iv} = f^3f_{yyy} + 4f^2f_{y.yy} + ff^3_y \]
\[ y^v = f^4f_{yyy} + 7f^3f_{y.yy} + 11f^2f^2_{y.yy} + f^3f_{yy} + ff^4_y \]

where all the functions are evaluated at \( x_n \).

Thus,
\[ y(x_{n+1}) = y + hy^i + \frac{h^2}{2} y^{ii} + \frac{h^3}{6} y^{iii} + \frac{h^4}{24} y^{iv} + \ldots \]

or
\[ y(x_{n+1}) = y + hf + \frac{h^2}{2} ff_y + \frac{h^3}{6} (ff_y^2 + f^2f_{yy}) + \frac{h^4}{24} (f^3_{yyy} + 4f^2f_{y.yy} + ff^3_y) + \ldots \]

Finally, by comparing equations (A1.15) and (A1.17) the first two terms are immediately satisfied and the remaining terms yield the relations given by equation (8), i.e.,

\[-192a_1 + 96 = 0\]
\[108 - 48a_6 - 24a_1 - 96a_1a_5 - 192a_1a_3 + 48a_1^2 = 0\]
\[24 - 96a_1^2 = 0\]
\[18 + 12a_6 + 3a_1 + 24a_1a_5 - 96a_1a_3a_6 + 6a_1^2 - 48a_1^2a_3 - 24a_1^3 = 0\]
\[108 - 60a_6 - 6a_1 - 96a_1a_5 - 96a_1a_3 - 12a_1^2 - 48a_1^2a_5 + 48a_1^2a_3 + 48a_1^3 = 0\]

and
\[4 - 32a_1^3 = 0\]

which have to be solved to determine the parameters \( a_i \), \( i = 1,2,3,4,5,6 \).