Abstract

The objective of this thesis studying and deriving with some modification as a new approach of Runge-Kutta method including explicit, semi-explicit and implicit methods as well as studying stability of convergence of these methods.

Also, one of most important themes of the thesis is to introduce variable step size and variable order methods using an extrapolation method which has the utility of controlling the local truncation error to be less than a prespecified tolerance error.

Acknowledgment

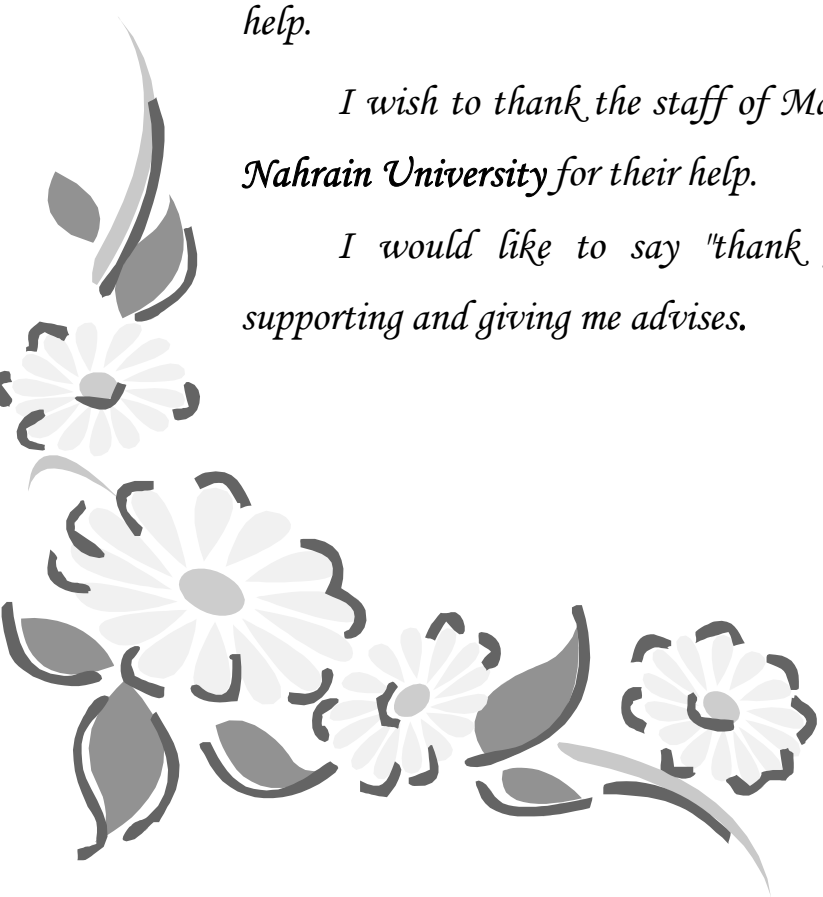
I would like to express my sincere appreciation to my research supervisor, ***Dr. Fadhel Subhi Fadhel,*** *for giving me the major steps to go on to explore the subject, shearing with me the ideas in my research "****Hand Identification using Fuzzy-Neural****" And perform the points that I felt were important.*

Also I wish to thank the Head of Department of Mathematics and my supervisor, ***Dr. Akram M. Al-Abood****, for his available advice and encouragement.*

I wish to say "thank you" to the ***Dr. Akram H. Al-Shather*** *for his help.*

I wish to thank the staff of Mathematics Department at the ***AL-Nahrain University*** *for their help.*

I would like to say "thank you" to my faithful friends for supporting and giving me advises.



APPENDIX A

Following the computer programs used in this thesis:

1. ERK (Explicit Runge-Kutta Program):

```
       {Computer Programming for Solving
           y'=-y+x+1, y(0)=1
           using explicit Runge-Kutta method]
program ar;
{uses crt;}
var x,y:array[0..12] of real;
    i,n:integer;
    aa,bb,k1,k2,h,a1,b1,error,ex:real;
    f:text;
begin
    x[0]:=0;y[0]:=1;h:=0.1;aa:=0;bb:=1;n:=round((bb-aa)/h);
    a1:=1;b1:=a1;
    assign(f,'d:\resultex.txt');
    rewrite(f);
    writeln(f,'x      y            exact         error');
    writeln(f,'---------------------------------------');
    for i:=0 to n do
     begin
      x[i]:=x[0]+i*h;
      k1:=-y[i]+x[i]+1;
      k2:=-y[i]-h*b1*k1+x[i]+a1*h+1;
      y[i+1]:=y[i]+0.5*h*(k1+k2);
      ex:=exp(-x[i])+x[i];
      error:=abs(y[i]-ex);
      writeln(f,x[i]:2:1,'  ',y[i]:10:10,'  ',ex:10:10,'
',error:10:10);
      writeln;
     end;
     close(f);
     readln;
end.
```

2. SERK (Semi Explicit Runge Kutta Program):

```
       {Computer Programming for Solving
           y'=-y+x+1, y(0)=1
           using semi- explicit Runge-Kutta method}
program ar;
var ex,x,y:array[0..16] of real;
    i,j,n:integer;

k1,k2,a1,a2,a3,a4,c1,c2,b11,b12,b21,b22,aa1,aa2,h,error:real;
    f:text;
begin
   n:=10;h:=0.1;
   aa1:=0.5-(1/(2*sqrt(3)));
   aa2:=0.5+(1/(2*sqrt(3)));
   b11:=0.5-(1/(2*sqrt(3)));
   b12:=0;
   b22:=0.25;
   b21:=0.25+(1/(2*sqrt(3)));
   a1:=1+b11*h;
   a2:=0;
   a3:=b21*h;
   a4:=1+b22*h;
   x[0]:=0;y[0]:=1;
   for i:=0 to n do
     x[i]:=x[0]+i*h;
   for i:=0 to n do
    begin
      c1:=-y[i]+x[i]+h*aa1+1;
      c2:=-y[i]+x[i]+h*aa2+1;
      k1:=c1/a1;
      k2:=(c2-a3*k1)/a4;
      y[i+1]:=y[i]+0.5*h*(k1+k2);
      ex[i]:=exp(-x[i])+x[i];
    end;
    assign(f,'d:\resultSM.txt');
    rewrite(f);
    writeln(f,'i   x      y         exact       error');
    writeln('_____________________________________________');
    for i:=0 to n do
     writeln(f,i,'  ',x[i]:3:1,'  ',y[i]:10:10,'
',ex[i]:10:10,'  ',abs(ex[i]-y[i]):10:10);
     close(f);
    readln;
end.
```

3. 2IRK (2-Stages Implicit Runge-Kutta Program):

```
       {Computer Programming for Solving
               y'=-y+x+1, y(0)=1
   using 2-stages implicit Runge-Kutta method}
program ar;
var ex,x,y:array[0..16] of real;
    i,n:integer;

k1,k2,a1,a2,a3,a4,c1,c2,b11,b12,b21,b22,aa1,aa2,h,error:real;
    f:text;
begin
   n:=10;h:=0.1;
   aa1:=0.5-(1/(2*sqrt(3)));
   aa2:=0.5+(1/(2*sqrt(3)));
   b11:=0.25;b22:=b11;
   b12:=0.25-(1/(2*sqrt(3)));
   b21:=0.25+(1/(2*sqrt(3)));
   a1:=1+b11*h;
   a2:=b12*h;
   a3:=b21*h;
   a4:=1+b22*h;
   x[0]:=0;y[0]:=1;
   for i:=0 to n do
     x[i]:=x[0]+i*h;
   for i:=0 to n do
    begin
      c1:=-y[i]+x[i]+h*aa1+1;
      c2:=-y[i]+x[i]+h*aa2+1;
      k2:=(c1*a3-c2*a1)/(a2*a3-a1*a4);
      k1:=(c1-a2*k2)/a1;
      y[i+1]:=y[i]+0.5*h*(k1+k2);
      ex[i]:=exp(-x[i])+x[i];
    end;
    assign(f,'d:\result2IM.txt');
    rewrite(f);
    writeln(f,'i   x      y         exact       error');

writeln(f,'_____________________________________________');
    for i:=0 to n do
     writeln(f,i,'  ',x[i]:2:1,'  ',y[i]:10:10,'
',ex[i]:10:10,'  ',abs(ex[i]-y[i]):10:10);
     close(f);
    readln;
end.
```

4. 3IRK (3-Stages Implicit Runge-Kutta Program):

```
      {Computer Programming for Solving
          y'=-y+x+1, y(0)=1
          using 3-stages implicit Runge-Kutta method}
program ar;
var ex,x,y:array[0..16] of real;
    i,j,n:integer;

k1,k2,k3,a1,a2,a3,a4,a5,a6,a8,a7,c3,c1,c2,b11,b12,b21,b22,b23
,b32,b33,b31,b13,aa1,aa2,h,aa3,a9:real;
    f:text;
begin
   n:=10;h:=0.1;
   aa1:=0.5;
   aa2:=0.5+(sqrt(15)/10);
   aa3:=0.5-(sqrt(15)/10);
   b11:=0.5-(sqrt(15)/5);b22:=b11;b33:=b11;
   b12:=sqrt(15)/5;b23:=b12;b13:=0;b31:=0;
   b21:=sqrt(15)/10;b32:=b21;
   a1:=1+b11*h;
   a2:=b12*h;
   a3:=0;
   a4:=b21*h;
   a5:=1+b22*h;
   a6:=b23*h;
   a7:=0;
   a8:=b32*h;
   a9:=1+b33*h;
   x[0]:=0;y[0]:=1;
   for i:=0 to n do
     x[i]:=x[0]+i*h;
   for i:=0 to n do
    begin
      c1:=-y[i]+x[i]+h*aa1+1;
      c2:=-y[i]+x[i]+h*aa2+1;
      c3:=-y[i]+x[i]+h*aa3+1;
      k2:=(a1*a9*c2-a9*a4*c1-a1*a6*c3)/(a1*a5*a9-a2*a9*a4-
a1*a6*a8);
      k1:=(c1-a2*k2)/a1;
      k3:=(c3-a8*k2)/a9;
      y[i+1]:=y[i]+(1/9)*h*(4*k1+2.5*k2+2.5*k3);
      ex[i]:=exp(-x[i])+x[i];
    end;
    assign(f,'d:\result3IMP.txt');
    rewrite(f);
    writeln(f,'i   x      y         exact       error');
    writeln(f,'_____________________________________________');
    for i:=0 to n do
```

```
     writeln(f,i,'  ',x[i]:2:1,'  ',y[i]:10:10,'
',ex[i]:10:10,'  ',abs(ex[i]-y[i]):10:10);
    close(f);
    readln;
end.
```

‘ROOT-RK Program
‘Evaluating the roots of Implicit Runge-Kutta method

```
CLS
OPEN "c:\arsh.dat" FOR OUTPUT AS #1
PRINT "     r          h"
PRINT "------------------------"
FOR h = -100 TO 100 STEP .001
r = 1 + (h - (.25 - SQR (3) / 6) * h ^ 2 + (SQR (3) / 24) * h ^ 3) / (2 - 3 * h / 2 + (15 / 36)*h
^ 2 + (3 / 72)* h ^ 3) + (h + (SQR (3) / 6)*h^2) / (2 - h + (1/ 6) * h ^ 2)
IF ABS(r) <= 1 THEN PRINT r, h: PRINT #1, r, h
NEXT h
CLOSE #1
```

الأهداء

الى من سهرت الليالي و رخصت لي الغوالي
الى من حفتني بالدعاء و عانت من أجلي الشقاء
والدتي الحنونة

الى من علمني الاخلاق فكان مثلي الاعلى
والدي العزيز

الى من يفرحون لفرحي و يحزنون لحزني
أخوتي و أخواتي

المستخلص

الهدف الرئيسي من هذة الاطروحة هو لدراسة و أشتقاق (مع بعض التطوير) لطرائق رانك كوتا(Runge-Kutta) و التي تتضمن الطرائق الصريحة(Explicit)، الطرائق شبة الصريحة(Semi-explicit) والضمنية(Implicit) بالأضافة الى دراسة أستقرارية و تقارب هذة الطرائق.

بالاضافة الى ذلك فأن من أهداف الأطروحة الأساسية هو أستخدام الطرائق ذات الخطوة المتغيرة(Variable Step Size) و طرائق الرتبة المتغيرة(Variable Order) بأستخدام طرائق الاستكمال الاضافي (Extrapolation Method) و التي أثبتت جدارتها بالسيطرة على حدود الخطأ بحيث تكون أقل من قيمة صغيرة.

CHAPTER ONE

GENERAL RUNGE-KUTTA METHODS

1.1 Introduction

After constructing a mathematical model for a certain real life problem as an ordinary differential equation, the next step is to find a solution. There are two approaches for evaluating the solution, "analytically" or "numerically". The analytic solution is usually obtained directly from the mathematical representation of the model formulate, while the numerical solution is generally an approximate obtained at certain node points. Most numerical methods are iterative, that is, the solution in a certain step uses the solution of the previous steps, such as Newton-Raphson method for approximating the roots of non-linear equation.

Error are an important aspect of computational life, they are every where and unavoidable. However, by careful analysis of the error in any numerical process, we can at least obtain bounds for these errors and therefore some measure of the accuracy of our final solution which must include the study of the sources an the propagation of the errors.

1.2 Basic Concepts

In this section, we shall present some of the basic concepts related to this work.

1.2.1 Finite Difference Equations [Bellman, 1963]:

Let $x_n = x_0 + nh$, $n = 0, 1, \ldots, k$; where h is the step length. A difference equation of order k is an equation involving the unknown y_n together with its differences up to order k, that is, difference equation has the form:

$$F(x_n; y_n, y_{n+1}, \ldots, y_{n+k}) = f(x_n) \quad \ldots\ldots (1.1)$$

Hence the order k of the difference equation is the difference between the highest and lowest indices of y.

We can classify difference equations according to several aspects, such as:

1. The finite difference equation is said to be linear if F appears linearly in y.
2. The finite difference equation is said to be homogeneous if $f(x_n) = 0$, otherwise it is non-homogenous.
3. The finite difference equation is said to be of constant coefficients if the coefficients of $y_n, y_{n+1}, \ldots, y_{n+k}$ are constants.

1.2.2 Solution of Linear Difference Equations with Constant Coefficients:

We shall occasionally need the general solution of the k-th order linear difference equation with constant coefficients [Lambert, 1973]:

$$a_k y_{n+k} + a_{k-1} y_{n+k-1} + \ldots + a_0 y_n = f_n \quad \ldots\ldots (1.2)$$

where $n = 1, 2, \ldots$, and a_j, $j = 0, 1, \ldots, k$, are constants independent of n, and $a_k \neq 0$, $a_0 \neq 0$.

A solution of such a difference equation will consist of a sequence y_1, y_2, …, which will be indicated by $\{y_n\}$.

Let $\{\hat{y}_n\}$ be the general solution of the corresponding homogeneous difference equation:

$$a_k y_{n+k} + a_{k-1} y_{n+k-1} + \ldots + a_0 y_n = 0 \quad \ldots\ldots (1.3)$$

$n = 1,2,\ldots$

If $\{\psi_n\}$ is any particular solution of eq.(1.2), then the general solution of eq.(1.2) is $\{y_n\}$; $y_n = \hat{y}_n + \psi_n$.

The solution of the difference equation can be evaluated easily by letting $y_n = \beta^n$, and considering the general solution of the difference equation depending on the roots of β whether it is a repeated or distinct real roots, or may be of complex roots, etc.

1.2.3 Legendre Polynomials [Burden, 1985]:

One of the most common sets of orthogonal polynomials is the set of Legendre polynomials $\{p_n\}$, which are orthogonal on $[-1, 1]$ with respect to the weighted function $w(x) \equiv 1$. These polynomials are defined recursively by:

$$p_0(x) = 1,\ p_1(x) = x,\ p_k(x) = (x - B_k)p_{k-1}(x) - C_k p_{k-2}(x),\ k = 2, 3, \ldots$$

Where:

$$B_k = \frac{\int_{-1}^{1} x[p_{k-1}(x)]^2\,dx}{\int_{-1}^{1} [p_{k-1}(x)]^2\,dx}$$

and

$$C_k = \frac{\int_{-1}^{1} x p_{k-1}(x) p_{k-2}(x)\,dx}{\int_{-1}^{1} [p_{k-1}(x)]^2\,dx}$$

We do not need the explicit representation of the Legendre polynomials, but only the knowledge that the polynomials p_n, for each n, has n-distinct roots $x_1, x_2, \ldots, x_n$ all of which lies in $(-1, 1)$.

1.3 Runge-Kutta Methods [Butcher, 1987]

The idea of extending the Euler method by allowing for a multiplicity of evolutions of the function f within each step was originally proposed by Runge (1895). Further contributions were made by Heun (1900) and by Kutta (1901). The latter completely characterized the set of Runge-Kutta method of order 4 and proposed the first methods of order 5. Special methods for second-order differential equations were proposed by Nyström (1925) who also contributed to the development of methods for first-order equations.

Since the advent of digital computers, fresh interest had been focused on Runge-Kutta methods, and a large number of research workers have contributed to recent extensions to the theory and the development of

particular methods. Although, early studies were devoted entirely to explicit Runge-Kutta methods, interest has now extended to implicit methods, which are now recognized as appropriate for stiff differential equations.

1.3.1 Formulation of Runge-Kutta Methods [Lambert, 1973]:

The general form of an r-stages Runge-Kutta methods is given by:

$$y_{n+1} = y_n + h\sum_{i=1}^{r} c_i k_i$$

where

$$k_i = f\left(x_n + ha_i, y_n + h\sum_{j=1}^{r} b_{ij} k_j \right)$$

and

$$a_i = \sum_{j=1}^{r} b_{ij}$$

where c_i, a_i and b_{ij}, for all i, j = 1, 2, …, r; are constants to be determined.

For convenience, we design the process by an array of constants, as follows:

b_{11}	b_{12}	…	b_{1j}	a_1
b_{21}	b_{22}	…	b_{2j}	a_2
⋮	⋮	⋱	⋮	⋮
b_{i1}	b_{i2}	…	b_{ij}	a_i
c_1	c_2	…	c_j	

and it is easy to classify Runge-Kutta methods, as follows:

- If $b_{ij} = 0, \forall\ i < j$, then the method is called semi-explicit.
- If $b_{ij} = 0, \forall\ i \leq j$, then the method is called explicit.
- Otherwise it is called implicit.

1.3.2 Derivation of Some Runge-Kutta Methods [Lambert, 1973]:

We shall consider the derivation of some Runge-Kutta methods namely, 2-stages explicit, the 2-stages semi-explicit and the 2-stages implicit Runge-Kutta methods for the purpose of studying the accuracy of the methods.

1.3.2.1 Derivation of 2-Stages Explicit Runge-Kutta Method:

In order to derive two stages Runge-Kutta method; consider first the general form of two stages Runge-Kutta methods, which is given by:

$$y_{n+1} = y_n + h(c_1k_1 + c_2k_2)$$

where:

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

$$k_2 = f(x_n + a_2h, y_n + hb_{21}k_1)$$

and

$$a_2 = b_{21}$$

Hence, in this problem we have three unknown constants c_1, c_2, and a_2 which must be determined.

Now, recall the Taylor series expansion for a function $g(x + h, y + k)$ of two variables about (x, y), we have:

$$g(x + h, y + k) = g(x, y) + hg_x(x, y) + kg_y(x, y) + \frac{h^2}{2!} g_{xx}(x, y) + hkg_{xy}(x, y)$$

$$+ \frac{k^2}{2!} g_{yy}(x, y) + \ldots$$

So expanding k_2 using Taylor series about (x_n, y_n), we have:

$$k_2 = f(x_n + a_2h, y_n + hb_{21}k_1)$$

$$= f(x_n, y_n) + a_2hf_x + hb_{21}k_1f_y + \frac{a_2^2h^2}{2!} f_{xx} + a_2b_{21}h^2k_2f_{xy} + \frac{h^2b_{21}^2k_1^2}{2!} f_{yy} + \ldots$$

$$= f + a_2h(f_x + ff_y) + \frac{a_2^2h^2}{2!} (f_{xx} + 2ff_{xy} + f^2f_{yy}) + \ldots$$

So, the two stages Runge-Kutta method takes the form:

$$y_{n+1} = y_n + h(c_1k_1 + c_2k_2)$$

$$= y_n + h[c_1f + c_2(f + a_2h(f_x + ff_y) + \frac{a_2^2h^2}{2!} (f_{xx} + 2ff_{xy} + f^2f_{yy}) + \ldots)]$$

$$= y_n + h[c_1f + c_2f + c_2a_2h(f_x + ff_y) + \frac{c_2a_2^2h^2}{2!} (f_{xx} + 2ff_{xy} + f^2f_{yy}) + \ldots]$$

$$\square\ y_n + hf(c_1 + c_2) + h^2c_2a_2(f_x + ff_y) + \frac{c_2a_2^2h^3}{2!} (f_{xx} + 2ff_{xy} +$$

$$f^2f_{yy}) + O(h^4) \ldots\ldots (1.4)$$

Since Taylor method takes the form:

$$y_{n+1} \square\ y_n + h\,y_n' + \frac{h^2}{2!} y_n'' + \frac{h^3}{3!} y_n''' + O(h^4) \ldots\ldots (1.5)$$

and since $y' = f(x, y)$, then $y'' = f_x + ff_y$, and

$$y''' = f_{xx} + ff_{xy} + f_y(f_x + ff_y) + f(f_{xy} + ff_{yy})$$

$$= (f_{xx} + 2ff_{xy} + f^2f_{yy}) + (f_xf_y + ff_y^2)$$

Therefore, eq.(1.5) becomes:

$$y_{n+1} \approx y_n + hf + \frac{h^2}{2!}(f_x + ff_y) + \frac{h^3}{3!}[(f_{xx} + 2ff_{xy} + f^2f_{yy}) + (f_xf_y + ff_y^2)] + O(h^4) \quad \text{..... (1.6)}$$

Now, in order to get an agreement between Runge-Kutta method and Taylor's method (i.e., between eq.(1.4) and eq.(1.6)), we must have:

$$c_1 + c_2 = 1 \text{ and } c_2a_2 = \frac{1}{2}$$

with local truncation error of order h^3. Also, the order for Runge-Kutta method with r = 2 equals to p = 2.

The above two equations have an infinite number of solutions, e.g., we can take also $c_1 = c_2 = \frac{1}{2}$ and $a_2 = 1$.

1.3.2.2 Derivation of 2-Stages Semi-Explicit Runge-Kutta Method:

Since r = 2, then Runge-Kutta formula takes the form:

$$y_{n+1} = y_n + h(c_1k_1 + c_2k_2)$$

where:

$$k_1 = f(x_n + a_1h, y_n + ha_1k_1)$$

$$k_2 = f(x_n + a_2h, y_n + hb_{21}k_1 + hb_{22}k_2)$$

and $a_1 = b_{11}$, $a_2 = b_{21} + b_{22}$.

To find these constants, consider the following power series:

$$k_1 = A_1 + hB_1 + h^2C_1 + \dots \quad \text{..... (1.7)}$$

$$k_2 = A_2 + hB_2 + h^2C_2 + \dots \quad \text{..... (1.8)}$$

Since:

$$k_1 \sqcup f + h[a_1 f_x + a_1(A_1 + hB_1 + h^2C_1 + \ldots)f_y + \frac{h^2}{2}[a_1^2 f_{xx} + 2a_1(A_1 + hB_1 + h^2C_1)f_{xy} + a_1^2(A_1 + hB_1 + h_2C_1 + \ldots)^2 f_{yy}] + \frac{h^3}{6}[a_1^3 f_{xxx} + 3a_1^3(A_1 + hB_1 + h^2C_1 + \ldots)f_{xxy} + 3a_1^3(A_1 + hB_1 + h^2C_1 + \ldots)^2 f_{xyy} + a_1^3(A_1 + hB_1 + h^2C_1 + \ldots)^3 f_{yyy} + O(h^4)$$ ..... (1.9)

Comparing equations (1.7) and (1.9), we have:

$$A_1 = f$$

$$B_1 = a_1(f_x + f_y A_1) = a_1(f_x + ff_y) = a_1 F$$

$$C_1 = \frac{a_1^2}{2}(f_{xx} + 2A_1 f_{xy} + A_1^2 f_{yy}) + a_1 B_1 f_y$$

$$= \frac{a_1^2}{2}(f_{xx} + 2ff_{xy} + f^2 f_{yy}) + a_1^2(f_x + ff_y)f_y$$

$$= \frac{a_1^2}{2}G + a_1^2 F f_y$$

where:

$G = f_{xx} + 2ff_{xy} + f^2 f_{yy}$, and

$$D_1 = a_1 C_1 f_y + a_1^2 B_1 f_{xy} + a_1^2 A_1 B_1 f_{yy} + \frac{a_1^3}{6} f_{xxx} + \frac{a_1^3}{2} A_1 f_{xxy} + \frac{a_1^3}{2} A_1 f_{xxy} + \frac{a_1^3}{6} A_1 f_{yyy}$$

$$= a_1^3 F f_y^2 + a_1^3 F(f_{xy} + ff_{yy}) + a_1^3 G f_y + \frac{a_1^3}{6} H$$

where: $F = f_x + ff_y$

and $H = f_{xxx} + 3ff_{xxy} + 3f^2f_{xyy} + f^3f_{yyy}$

and similarly:

$$k_2 \sqcup f + h[a_2f_x + (b_{21}(A_1 + hB_1 + h^2C_1) + b_{22}(A_2 + hB_2 + h^2C_2))f_y] + \frac{h^2}{2}[a_2^2 f_{xx} + 2a_2(b_{21}(A_1 + hB_1) + b_{22}(A_2 + hB_2))f_{xy} + (b_{21}(A_1 + hB_1) + b_{22}(A_2 + hB_2))^2 f_{yy}] + \frac{h^3}{6}[a_2^3 f_{xxx} + 3a_2^2(b_{21}A_1 + b_{22}A_2)f_{xxy} + 3a_2(b_{21}A_1 + b_{22}A_2)^2 f_{xyy} + (b_{21}A_1 + b_{22}A_2)^3 f_{yyy}] + O(h^4) \quad \text{..... (1.10)}$$

Comparing equations (1.8) and (1.10), we have:

$A_2 = f$

$B_2 = a_2f_x + (b_{21}A_1 + b_{22}A_2)f_y$

$C_2 = (b_{21}B_1 + b_{22}B_2)f_y + \frac{a_2^2}{2} f_{xx} + a_2(b_{21}A_1 + b_{22}A_2)f_{xy} + \frac{1}{2}(b_{21}A_1 + b_{22}A_2)^2 f_{yy}$, and

$D_2 = (b_{21}C_1 + b_{22}C_2)f_y + a_2(b_{21}B_1 + b_{22}B_2) f_{xy} + (b_{21}A_1 + b_{22}A_2)(b_{21}B_1 + b_{22}B_2)f_{yy} + \frac{a_1^3}{6} f_{xxx} + \frac{a_1^2}{6}(b_{21}A_1 + b_{22}A_2)f_{xxy} + \frac{a_2}{2}(b_{21}A_1 + b_{22}A_2)^2 f_{xyy} + \frac{1}{6}(b_{21}A_1 + b_{22}A_2)^2 f_{yyy}$

Since $a_2 = b_{21} + b_{22}$, therefore, using a_2 to simplify and solve the last fore equations, we obtain the following solution:

$A_2 = f$

$B_2 = a_2f_x + (b_{21}f + b_{22}f)f_y$

$= a_2f_x + (b_{21} + b_{22})ff_y$

$= a_2f_x + a_2ff_y$

$= a_2(f_x + ff_y) = a_2F$

$C_2 = (b_{21}a_1 + b_{22}a_2)Ff_y + \frac{a_2}{2}G$

where, $G = f_{xx} + 2ff_{xy} + f^2f_{yy}$, an similarly:

$D_2 = [b_{21}(b_{11}a_1 + b_{12}a_2) + b_{22}(b_{21}a_1 + b_{22}a_2)]F f_y^2 + a_2(b_{21}a_1 + b_{22}a_2)F(f_{xy} +$

$ff_{yy}) + (b_{21}a_1^2 + b_{22}a_2^2)Gf_y + \frac{a_1^3}{6}H$

where $H = f_{xxx} + 3f_{xxy} + 3f^2f_{xyy} + f^3f_{yyy}$

Since $\phi(x, y, h) = \sum_{i=1}^{2} c_i k_i$. Hence:

$\phi = c_1k_1 + c_2k_2$

$\sqcup\ c_1(A_1 + hB_1 + h^2C_1 + h^3D_1) + c_2(A_2 + hB_2 + h^2C_2 + h^3D_2) + O(h^4)$

$\sqcup\ c_1A_1 + c_2A_2 + h(c_1B_1 + c_2B_2) + h^2(c_1C_1 + c_2C_2) + h^3(c_1D_1 + c_2D_2) +$
$O(h^4)$..... (1.11)

where the coefficients A_i, B_i, C_i and D_i, i = 1, 2, are given above.

Comparing with the total differential expansion of $\phi(x, y, h)$ of equation (1.11), we have:

$\phi(x, y, h) \sqcup f + \frac{1}{2}hF + \frac{1}{6}h^2(Ff_y + G) + \frac{1}{24}h^3[(3f_{xy} + 3ff_{yy} + f_y^2)F + Gf_y +$
$H] + O(h^4)$..... (1.12)

where:

$F = f_x + ff_y$

$G = f_{xx} + 2ff_{xy} + f^2f_{yy}$

$$H = f_{xxx} + 3ff_{xxy} + 3f^2f_{xyy} + f^3f_{yyy}$$

Comparing equations (1.11) and (1.12), we have the following case:

(i) Two stages semi-explicit Runge-Kutta method of order one, if:

$$c_1 + c_2 = 1$$

(ii) Two stages semi-explicit Runge-Kutta method of order two, if:

$$c_1 + c_2 = 1$$

$$c_1a_1 + c_2a_2 = 1/2$$

(iii) Two stages semi-explicit Runge-Kutta method of order three, if:

$$c_1 + c_2 = 1$$

$$c_1a_1 + c_2a_2 = 1/2$$

$$c_1 a_1^2 + c_2(b_{21}a_1 + b_{22}a_2) = 1/6$$

$$c_1 a_1^2 + c_2 a_2^2 = 1/3$$

(iv) Two stages semi-explicit Runge-Kutta method of order four, if:

$$c_1 + c_2 = 1$$

$$c_1a_1 + c_2a_2 = 1/2$$

$$c_1 a_1^2 + c_2(b_{21}a_1 + b_{22}a_2) = 1/6$$

$$c_1 a_1^2 + c_2 a_2^2 = 1/3$$

$$(c_1b_{11} + c_2b_{21})\ a_1^2 + c_2b_{22}(b_{21}a_1 + b_{22}a_2) = 1/24$$

$$c_1 a_1^3 c_2a_2(b_{21}a_1 + b_{22}a_2) = 1/8$$

$$c_1 a_1^3 + c_2(b_{21} a_1^2 + b_{22} a_2^2) = 1/12$$

$$c_1 a_1^3 + c_2 a_2^3 = 1/4$$

From the above results, one can see that more accurate methods (of higher order) could be used with small stages. For example, one of the solutions to the fourth order method is given by:

$$c_1 = c_2 = \frac{1}{2},\ a_{1,2} = \frac{1}{2} \mp \frac{\sqrt{3}}{6},\ b_{11} = \frac{1}{2} - \frac{\sqrt{3}}{6},\ b_{21} = a_2 - \frac{1}{4} = \frac{1}{4} + \frac{\sqrt{3}}{6}$$

and $b_{22} = \frac{1}{4}$

1.3.2.3 Derivation of 2-Stages Implicit Runge-Kutta Method:

Consider the 2-stages implicit Runge-Kutta method, which takes the form:

$$y_{n+1} = y_n + h(c_1k_1 + c_2k_2)$$

where:

$$k_1 = f(x_n + ha_1,\ y_n + b_{11}hk_1 + b_{12}hk_2)$$

$$k_2 = f(x_n + ha_2,\ y_n + b_{21}hk_1 + b_{22}hk_2)$$

and $a_1 = b_{11} + b_{12}$, $a_2 = b_{21} + b_{22}$.

Now, expanding k_i using Taylor series expansion about (x_n, y_n), we obtain that for i = 1, 2.

$$k_i \sqcup f(x_n, y_n) + h[a_i f_x + (b_{i1}k_1 + b_{i2}k_2)f_y] + \frac{h^2}{2}[a_i^2 f_{xx} + 2a_i(b_{i1}k_1 + b_{i2}k_2)f_{xy}$$

$$+ (b_{i1}k_1 + b_{i2}k_2)^2 f_{yy}] + \frac{h^3}{6}[a_i^3 f_{xxx} + 3a_i^2(b_{i1}k_1 + b_{i2}k_2)f_{xxy} + 3a_i(b_{i1}k_1$$

$$+ b_{i2}k_2)^2 f_{xyy} + (b_{i1}k_1 + b_{i2}k_2)^3 f_{yyy}] + O(h^4) \quad \text{..... (1.13)}$$

which will gives two implicit equations for i = 1, 2. Hence, we can no longer proceed as in previous derivations and Runge-Kutta methods. Therefore, as in the semi-explicit methods, suppose that:

$$k_i \sqcup A_i + hB_i + h^2C_i + h^3D_i + O(h^4),\ i = 1, 2 \quad \text{..... (1.14)}$$

Equating equations (1.13) and (1.14), we get:

$$A_i + hB_i + h^2C_i + h^3D_i \sqcup f + h[a_if_x + (b_{i1}(A_1 + hB_1 + h^2C_1) + b_{i2}(A_2 + hB_2 + h^2C_2))f_y] + \frac{h^2}{2}[a_i^2 f_{xx} + 2a_i(b_{i1}(A_1 + hB_1) + b_{i2}(A_2 + hB_2))f_{xy} + (b_{i1}(A_1 + hB_1) + b_{i2}(A_2 + hB_2))^2 f_{yy} + \frac{h^3}{6}[a_i^3 f_{xxx} + 3a_i^2(b_{i1}A_1 + b_{i2}A_2)f_{xxy} + 3a_i(b_{i1}A_1 + b_{i2}A_2)^2 f_{xyy} + (b_{i1}A_1 + b_{i2}A_2)^3 f_{yyy}] + O(h^4),\ i = 1, 2$$

Equating the coefficients of h^0, h^1, h^2 and h^3, we obtain that:

$$A_i = f$$

$$B_i = a_if_x + (b_{i1}A_1 + b_{i2}A_2)f_y$$

$$C_i = (b_{i1}B_1 + b_{i2}B_2)f_y + \frac{1}{2}a_i^2 f_{xx} + a_i(b_{i1}A_1 + b_{i2}A_2)f_{xy} + \frac{1}{2}(b_{i1}A_1 + b_{i2}A_2)^2 f_{yy}$$

$$D_i = (b_{i1}C_1 + b_{i2}C_2)f_y + a_i(b_{i1}B_1 + b_{i2}B_2)f_{xy} + (b_{i1}A_1 + b_{i2}A_2)(b_{21}B_1 + b_{i2}B_2)f_{yy} + \frac{1}{6}a_i^3 f_{xxx} + \frac{1}{2}a_i^2(b_{i1}A_1 + b_{i2}A_2)f_{xxy} + \frac{1}{2}a_i(b_{i1}A_1 + b_{i2}A_2)^2 f_{xyy} + \frac{1}{6}(b_{i1}A_1 + b_{i2}A_2)^3 f_{yyy},\ i = 1, 2.$$

Since $a_1 = b_{11} + b_{12}$ and $a_2 = b_{21} + b_{22}$, or in general notation $a_i = b_{i1} + b_{i2}$. Therefore, using the notation for a_1 and a_2 to simplify and solve the last form equations, we obtain the following solution:

$$A_i = f$$

$$B_i = a_i f_x + (b_{i1} f + b_{i2} f) f_y$$

$$= a_i f_x + (b_{i1} + b_{i2}) f f_y$$

$$= a_i f_x + a_i f f_y$$

$$= a_i (f_x + f f_y) = aF$$

$$C_i = (b_{i1} a_1 F + b_{i2} a_2 F) f_y + \frac{1}{2} a_i^2 f_{xx} + a_i (b_{i1} f + b_{i2} f) f_{xy} + \frac{1}{2} (b_{i1} f + b_{i2} f)^2 f_{yy}$$

$$= (b_{i1} a_1 + b_{i2} a_2) F f_y + \frac{1}{2} a_i^2 f_{xx} + a_i^2 f f_{xy} + \frac{1}{2} a_i^2 f^2 f_{yy}$$

$$= (b_{i1} a_1 + b_{i2} a_2) F f_y + \frac{1}{2} a_i^2 (f_{xx} + 2 f f_{xy} + f^2 f_{yy})$$

$$= (b_{i1} a_1 + b_{i2} a_2) F f_y + \frac{1}{2} a_i^2 G$$

where $G = f_{xx} + 2 f f_{xy} + f^2 f_{yy}$.

Similarly:

$$D_i = [b_{i1}(b_{11} a_1 + b_{12} a_2) + b_{i2}(b_{21} a_1 + b_{22} a_2)] F f_y^2 + a_i (b_{i1} a_1 + b_{i2} a_2) F (f_{xy} + f f_{yy}) + (b_{i1} a_1^2 + b_{i2} a_2^2) G f_y + \frac{1}{6} a_i^3 H$$

where $H = f_{xxx} + 3 f f_{xxy} + 3 f^2 f_{xyy} + f^3 f_{yyy}$.

Since:

$$\varphi(x, y, h) = \sum_{i=1}^{2} c_i k_i$$

Hence using eq.(1.14), one get:

$$\varphi = c_1 k_1 + c_2 k_2$$

$$\square\ c_1(A_1 + hB_1 + h^2C_1 + h^3D_1) + c_2(A_2 + hB_2 + h^2C_2 + h^3D_2) + O(h^4)$$

$$\square\ c_1A_1 + c_2A_2 + h(c_1B_1 + c_2B_2) + h^2(c_1C_1 + c_2C_2) + h^3(c_1D_1 + c_2D_2) + O(h^4) \quad \text{..... (1.15)}$$

where the coefficients A_i, B_i, C_i and D_i, $\forall$ i =1, 2, are given above.

Comparing with the total differential expansion of $\varphi(x, y, h)$ of eq.(1.15)

$$\varphi(x, y, h) \square\ f + \frac{1}{2}hF + \frac{1}{6}h^2(Ff_y + G) + \frac{1}{24}h^3[(3f_{xy} + 3ff_{yy} + f_y^2)F + Gf_y + H] + O(h^4) \quad \text{..... (1.16)}$$

where:

$$F = f_x + ff_y$$

$$G = f_{xx} + 2ff_{xy} + f^2f_{yy}$$

$$H = f_{xxx} + 3ff_{xxy} + 3f^2f_{xyy} + f^3f_{yyy}$$

Comparing (1.15) and (1.16), we have the following cases:

(i) Two stages implicit Runge-Kutta method of order one, if:

$$c_1 + c_2 = 1$$

(ii) Two stages implicit Runge-Kutta method of order two, if:

$$c_1 + c_2 = 1$$

$$c_1a_1 + c_2a_2 = 1/2$$

(iii) Two stages implicit Runge-Kutta method of order p = 3, if:

$$c_1 + c_2 = 1$$

$$c_1a_1 + c_2a_2 = 1/2$$

$$c_1(b_{11}a_1 + b_{12}a_2) + c_2(b_{21}a_1 + b_{22}a_2) = 1/6$$

$$c_1 a_1^2 + c_2 a_2^2 = 1/3$$

(iv) Two stages implicit Runge-Kutta method of order p = 4, if:

$$c_1 + c_2 = 1$$

$$c_1 a_1 + c_2 a_2 = 1/2$$

$$c_1(b_{11}a_1 + b_{12}a_2) + c_2(b_{21}a_1 + b_{22}a_2) = 1/6$$

$$c_1 a_1^2 + c_2 a_2^2 = 1/3$$

$$(c_1 b_{11} + c_2 b_{21})(b_{11}a_1 + b_{12}a_2) + (c_1 b_{12} + c_2 b_{22})(b_{21}a_1 + b_{22}a_2) = 1/24$$

$$c_1 a_1(b_{11}a_1 + b_{12}a_2) + c_2 a_2(b_{21}a_1 + b_{22}a_2) = 1/8$$

$$c_1(b_{11} a_1^2 + b_{12} a_2^2) + c_2(b_{21} a_1^2 + b_{22} a_2^2) = 1/12$$

$$c_1 a_1^3 + c_2 a_2^3 = 1/4$$

From the above results, one can see that more accurate methods (of higher order) could be obtained with small stages. For example, one of the solutions to the fourth order method is given by:

$$c_1 = c_2 = \frac{1}{2},\ a_{1,2} = \frac{1}{2} \mp \frac{\sqrt{3}}{6},$$

$$b_{11} = b_{22} = \frac{1}{4},\ b_{12} = a_1 - \frac{1}{4} = \frac{1}{4} - \frac{\sqrt{3}}{6},\ b_{21} = a_2 - \frac{1}{4} = \frac{1}{4} + \frac{\sqrt{3}}{6}$$

1.4 Stability and Convergence of Runge-Kutta Methods [Butcher, 1987], [Lambert, 1973]

Since the purpose of numerical analysis is to represent the solution to actual problems. It is important that what could be called qualitative properties of the numerical solution should resemble those of the true solution.

By stability analysis, we shall mean study of such qualitative properties as boundedness and convergence to zero of numerical solutions, when these properties are passed by the exact solution. Given a slightly different emphasis, this type of analysis is appropriate for studying the growth of numerical errors in a computed solution to a differential equation.

Remark [Atkinson, 1989], [Al-Kubeisy, 2004]:

Recalling the general form of Runge-Kutta method, which is:

$$y_{n+1} = y_n + h\varphi(x_n, y_n, h) \quad \text{..... (1.17)}$$

which could be considered as a special case of the general Linear Multistep Methods (for short LMM), given by:

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h \sum_{j=0}^{k} \beta_j f_{n+j} \quad \text{..... (1.18)}$$

(indeed, one step explicit LMM). Therefore, the stability of (1.17) is equivalent to the convergence of (1.17) (i.e., stability and convergence of Runge-Kutta are equivalent).

Theorem (1.1):

Assume the consistency condition and suppose that:

$$y_{n+1} - y_n = h\varphi(x_n, y_n, h) \quad \text{..... (1.19)}$$

which is a special case of (1.18) as one step method with $\alpha_1 = 1$, $\alpha_0 = -1$, $\beta_0 = 1$, $\beta_1 = 0$ and $f = \varphi$. Then (1.19) is converge if and only if it is zero stable.

Proof:

Suppose that the method is convergent and to prove that the method is zero stable

Consider for simplicity the problem, $y' = 0$, $y(0) = 0$, which has the exact solution $y(x) = 0$

Since $f = 0$, then for all i, we have $k_i = 0$ and hence $\varphi = 0$

Therefore, the method takes the form:

$$y_{n+1} - y_n = 0$$

Therefore, the first characteristic polynomial $\rho(r)$ is given by:

$$\rho(r) = r - 1 = 0$$

then $r = 1$.

and since $|r| \leq 1$, then the method is zero stable

Conversely, suppose that the Runge-Kutta method is zero stable and to prove that the method is convergent

Similarly, for simplicity purpose consider $y' = \lambda y$, $y(0) = 1$

To show that the term $[r_0(\lambda h)]^n$ in the general solution

$$y_0 = [r_0(\lambda h)]^n$$

will be converge to the exact solution $y(x) = e^{\lambda x}$ on [0, b], and then can be shown to be converge to the zero solution as $h \longrightarrow 0$

Expanding $r_0(\lambda h)$ using Taylor's theorem:

$$r_0(\lambda h) = r_0(0) + h\lambda r'_0(0) + O(h^2)$$

Hence:(using the consistency condition)

$$r'_0(0) = \frac{\sigma(r_0(0))}{\rho'(r_0(0))}$$

Since $r_0(0) = 1$. Then:

$$r'_0(0) = \frac{\sigma(1)}{\rho'(1)} = 1$$

Hence:

$$r_0(\lambda h) = 1 + \lambda h + O(h^2)$$

$$= e^{\lambda h} - O(h^2) + O(h^2)$$

$$\sqcup\ e^{\lambda h}$$

Therefore:

$$[r_0(\lambda h)]^n = [e^{\lambda h}]^n = e^{\lambda_n h} = e^{\lambda x_n}$$

Hence:

$$\underset{0 \le x_n \le b}{\text{Max}} |[r_0(\lambda h)]^n - e^{\lambda x_n}| \longrightarrow 0 \text{ as } h \longrightarrow 0. \quad \blacksquare$$

1.4.1 Stability of Explicit Runge-Kutta Method:

To obtain intervals of stability of 2-stages Runge-Kutta method applied to the test problem $y' = \lambda y, Re(\lambda) < 0$, we have:

$$y_{n+1} = y_n + h(c_1k_1 + c_2k_2)$$

where:

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

$$k_2 = f(x_n + a_2h, y_n + a_2hk_1)$$

and for $f(x, y) = \lambda y$, we have:

$$k_1 = \lambda y_n$$

$$k_2 = \lambda(y_n + a_2\lambda h y_n)$$

$$= \lambda y_n + a_2\lambda^2 h y_n$$

since:

$$y_{n+1} = y_n + h(c_1\lambda y_n + c_2\lambda y_n + c_2a_2h\lambda^2 y_n)$$

$$= y_n + (c_1 + c_2)\hbar\, y_n + c_2a_2\hbar^2 y_n,\ \hbar = \lambda h$$

$$= y_n[1 + (c_1 + c_2)\ \hbar\ + c_2a_2\hbar^2]$$

Hence, to find the roots of the first characteristic polynomial $\rho(r) = 0$, we have to letting $y_n = r^n$, so

$$r^{n+1} - r^n[1 + (c_1 + c_2)\hbar\ + c_2a_2\hbar^2] = 0$$

$$r^n\{r - [1 + (c_1 + c_2)\ \hbar\ + c_2a_2\hbar^2]\} = 0$$

and since $r^n \neq 0$, then:

$$r - [1 + (c_1 + c_2)\ \hbar\ + c_2a_2\hbar^2] = 0$$

i.e.,

$$r = 1 + (c_1 + c_2)\ \hbar\ + c_2a_2\hbar^2,\ |r| < 1$$

since for 2-stages Runge-Kutta method, $p = 2$, we have:

$$c_1 + c_2 = 1 \text{ and } c_2 a_2 = \frac{1}{2}$$

then:

$$r = 1 + \hbar + \frac{1}{2}\hbar^2$$

In order to get a method which is stable, we must have $|r| < 1$, so:

$$|1 + \hbar + \frac{1}{2}\hbar^2| < 1$$

Then

$$-1 < 1 + \hbar + \frac{1}{2}\hbar^2 < 1$$

Which implies that $-2 < \hbar < 0$. So all $r = p = 2$ Runge-Kutta methods have an interval of absolute stability to be $(-2, 0)$.

1.4.2 Stability of Semi-Explicit Runge-Kutta Method:

To obtain intervals of stability of 2-stages Runge-Kutta methods. Also, we consider the test problem $y' = \lambda y$, where $Re(\lambda) < 0$:

$$y_{n+1} = y_n + h(c_1 k_1 + c_2 k_2)$$

where:

$$k_1 = \lambda(y_n + h a_1 k_1)$$

$$k_2 = \lambda(y_n + h b_{21} k_1 + h b_{22} k_2)$$

so:

$$k_1 = \frac{\lambda y_n}{1 - \hbar a_1}, \quad 1 - \hbar a_1 \neq 0$$

$$k_2 = \frac{\lambda y_n[(1-\hbar a_1)+\hbar b_{21}]}{(1-\hbar b_{22})(1-\hbar a_1)}\ ,\ (1-\hbar\, b_{22})\ (1-\hbar\, a_1)\ \neq 0$$

so:

$$y_{n+1} = y_n + \frac{h}{2}\left[\frac{\lambda y_n}{1-\hbar a_1} + \frac{\lambda y_n[1-\hbar a_1+\hbar b_{21}]}{(1-\hbar b_{22})(1-\hbar a_1)}\right]$$

$$= y_n + \frac{\hbar}{2} y_n\left[\frac{1}{1-\hbar a_1} + \frac{1-\hbar a_1+\hbar b_{21}}{(1-\hbar b_{22})(1-\hbar a_1)}\right]$$

$$= y_n + \frac{\hbar}{2} y_n\left[\frac{(1-\hbar b_{22})+(1-\hbar a_1+\hbar b_{21})}{(1-\hbar b_{22})(1-\hbar a_1)}\right]$$

$$= y_n + \frac{\hbar}{2} y_n\left[\frac{2(1-\hbar b_{22})}{(1-\hbar b_{22})(1-\hbar a_1)}\right]$$

$$= y_n\left[1+\frac{\hbar}{1-\hbar a_1}\right]$$

So, the corresponding root is given by:

$$r = 1 + \frac{\hbar}{1-\hbar a_1} = \frac{1-\hbar a_1+\hbar}{1-\hbar a_1} = \frac{s+\hbar}{s},\ \hbar < 0$$

So $|r| = \left|\frac{s+\hbar}{s}\right| < 1$

Hence, the interval of stability is given by $(-\infty, 0)$.

1.4.3 Stability of Implicit Runge-Kutta Method:

To obtain intervals of stability of 2-stages, fourth order Runge-Kutta method given in section (1.3.2.3).

Consider the test problem $y' = \lambda y$, where $Re(\lambda) < 0$, hence:

$$y_{n+1} = y_n + h(c_1k_1 + c_2k_2)$$

where:

$$k_1 = f(x_n + ha_1, y_n + b_{11}hk_1 + b_{12}hk_2)$$

$$k_2 = f(x_n + ha_2, y_n + b_{21}hk_1 + b_{22}hk_2)$$

so for the test problem $y' = \lambda y$, we have:

$$k_1 = \lambda y_n + \frac{1}{4}\hbar k_1 + (\frac{1}{4} - \frac{\sqrt{3}}{6})\hbar k_2$$

$$k_2 = \lambda y_n + (\frac{1}{4} + \frac{\sqrt{3}}{6})\hbar k_1 + \frac{1}{4}\hbar k_2$$

Therefore, when $\hbar = \lambda h$, we have:

$$k_1 = \frac{\left[1-\left(\frac{1}{4}+\frac{\sqrt{3}}{6}\right)\hbar+\left(\frac{\sqrt{3}}{24}\right)\hbar^2\right]\lambda y_n}{1-\frac{3}{4}\hbar+\frac{15}{72}\hbar^2-\frac{3}{144}\hbar^3}\ ,\ 1-\frac{3}{4}\hbar+\frac{15}{72}\hbar^2-\frac{3}{144}\hbar^3 \neq 0$$

$$k_2 = \frac{\left[1+\frac{\sqrt{3}}{6}\hbar\right]\lambda y_n}{1-\frac{1}{2}\hbar+\frac{1}{12}\hbar^2}\ ,\ 1-\frac{1}{2}\hbar+\frac{1}{12}\hbar^2 \neq 0$$

and since:

$$y_{n+1} = y_n + h(c_1k_1 + c_2k_2)$$

Therefore:

$$y_{n+1} = y_n + \frac{\hbar y_n}{2}\left[\frac{1-\left(\frac{1}{4}+\frac{\sqrt{3}}{6}\right)\hbar+\left(\frac{\sqrt{3}}{24}\right)\hbar^2}{1-\frac{3}{4}\hbar+\frac{15}{72}\hbar^2-\frac{3}{144}\hbar^3}+\frac{\left(1+\frac{\sqrt{3}}{6}\hbar\right)}{1-\frac{1}{2}\hbar+\frac{1}{12}\hbar^2}\right]$$

$$= y_n\left[1+\frac{\hbar-\left(\frac{1}{4}+\frac{\sqrt{3}}{6}\right)\hbar^2+\frac{\sqrt{3}}{24}\hbar^3}{2-\frac{3}{2}\hbar+\frac{15}{36}\hbar^2-\frac{3}{72}\hbar^3}+\frac{\hbar+\frac{\sqrt{3}}{6}\hbar^2}{2-\hbar-\frac{1}{6}\hbar^2}\right]$$

So the corresponding characteristic root is given by:

$$r = 1+\frac{\hbar-\left(\frac{1}{4}+\frac{\sqrt{3}}{6}\right)\hbar^2+\frac{\sqrt{3}}{24}\hbar^3}{2-\frac{3}{2}\hbar+\frac{15}{36}\hbar^2-\frac{3}{72}\hbar^3}+\frac{\hbar+\frac{\sqrt{3}}{6}\hbar^2}{2-\hbar-\frac{1}{6}\hbar^2}$$

Therefore, upon applying computer programming **(ROOT-RK Program)**, the following interval of absolute stability is obtained, which is $\hbar \in (-1, -0.95) \cup (-0.64, 0)$.

CHAPTER TWO

MODIFIED RUNGE-KUTTA METHODS

2.1 Introduction

Derivations of explicit, semi-explicit and implicit Runge-Kutta methods are of great difficulties, especially when the stages of the method are increased. Therefore, the need for simple and efficient methods (with large stages) is necessary.

In this chapter, we will present some modified Runge-Kutta methods, which has its basis on tridiagonally implicit methods in which the diagonal elements has an equal values. This method has been proved to be stable and convergent.

2.2 Fundamental Results in Runge-Kutta Methods

In this section, fundamental theoretical results consuming Runge-Kutta methods will be given, as well as, their proofs.

In order to give some results, for simplicity an without loose of generality, the following ordinary differential equation will be considered:

$y' = f(y), y(x_0) = y_0$..... (2.1)

Therefore, one of the most important results (which are given in [Butcher, 1987], [Butcher, 1964], [Al-Exander, 1977] and [Bickart, 1977] without proof) in Runge-Kutta methods which has the utility of evaluating the parameter of Runge-Kutta methods, namely b_j's. This result will be stated in the next theorem:

Theorem (2.1):

Consider the system:

$y'(x) = f(y)$, $y = y_0$ at $x = x_0$ ..... (2.2)

then $\phi = \frac{1}{\gamma}$, where $\gamma = \frac{i\beta}{\alpha}$, $\forall$ i =1, 2, …, r and α, β are numerical coefficients independent of the form of f.

Proof:

Suppose that y, y* be the exact and approximate solutions of equation (2.2), respectively.

The equation defining y* for r-stages Runge-Kutta methods is:

$$y^* = y_0 + h\sum_{i=1}^{r} c_i k_i$$

where:

$$k_i = f\left(y_0 + h\sum_{j=1}^{r} b_{ij} k_j \right), \; i = 1, 2, \ldots, r$$

and b_{ij}, c_i, $\forall$ i, j = 1, 2, .., r; are constants to be determined.

The power series expansion of y and y* are respectively:

$$y = y_0 + \sum_{i=1}^{r} \alpha F \frac{h^i}{i!} \quad \text{where F for the function f.}$$

$$y^* = y_0 + \sum_{i=1}^{r} \beta \phi F \frac{h^i}{(i-1)!}$$

Since y and y* having the same order, then:

$$\alpha F \frac{h^i}{i!} = \beta \phi F \frac{h^i}{(i-1)!}$$

and hence:

$$\alpha F \frac{h^i}{i(i-1)!} = \beta \phi F \frac{h^i}{(i-1)!}$$

Therefore:

$$\phi = \frac{\alpha}{i\beta} = \frac{1}{\gamma}. \quad \blacksquare$$

The next lemma plays an important role in the basis of theorem (2.2).

Lemma (2.1) [Butcher, 1987]:

Let U and V be 3×3 matrices, such that:

$$UV = \begin{bmatrix} w_{11} & w_{12} & 0 \\ w_{21} & w_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Where $w_{11}w_{22} - w_{21}w_{12} \neq 0$. Then either the third row of U is the zero vector or the third column of V is the zero vector.

Remark:

Recalling the theory of Graphs and combining t is any tree with roots r_1, r_2, .., r_s, then we write this tree symbolically as t=[t_1, t_2, .., t_s] and we can write this conscience using the following notation which is the plans subscript on [ ] to indicate repetition.

For example the tree $[t_1t_1t_1t_2t_2]$ can be written as $[t_1^2t_2^2]$ or $[[[\tau]\tau]]$ or $[_2[\tau]\tau]_2$.

Theorem (2.2):

Consider the system:

$y'(x) = f(y)$, $y = y_0$ at $x = x_0$

If $\phi = \frac{1}{\gamma}$, $r \le \xi$, then $\sum_{j=1}^{r} c_j a_j^{k-1} = \frac{1}{k}$, for $k \le \xi$, where k = 1, 2, …, r, and r is the number of stages of Runge-Kutta method and ξ is the order of the considered method.

Proof:

In order to prove that $\sum_{j=1}^{r} c_j a_j^{k-1} = \frac{1}{k}$, for $k \le \xi$, we consider lemma (2.1) in mind with:

$$u_1^T = [c_2 \quad c_3 \quad c_4]$$

$$u_2^T = [c_2a_2 \quad c_3a_3 \quad c_4a_4]$$

$$u_3^T = \left[\sum_{i=1}^{r} c_i b_{i2} - c_2(1-a_2) \quad \sum_{i=1}^{r} c_i b_{i3} - c_3(1-a_3) \quad \sum_{i=1}^{r} c_i b_{i4} - c_4(1-a_4) \right]$$

$$v_1 = \begin{bmatrix} a_2 \\ a_3 \\ a_4 \end{bmatrix}, v_2 = \begin{bmatrix} a_2^2 \\ a_3^2 \\ a_4^2 \end{bmatrix}, v_3 = \begin{bmatrix} \sum_{j=1}^{r} b_{2j} a_j - \frac{1}{2} a_2^2 \\ \sum_{j=1}^{r} b_{3j} a_j - \frac{1}{2} a_3^2 \\ \sum_{j=1}^{r} b_{4j} a_j - \frac{1}{2} a_4^2 \end{bmatrix}$$

with $\gamma(x) = k$ in $\phi = \frac{1}{\gamma}$, $r \le \xi$, where $k \le 5$ [Butcher, 1987], we have:

$$u_1^T v_1 = \sum_k c_k a_k = \phi([\tau]) = \frac{1}{\gamma([\tau])} = \frac{1}{2}$$

$$u_1^T v_2 = u_2^T v_1 = \sum_k c_k a_k^2 = \phi([\tau^2]) = \frac{1}{\gamma([\tau^2])} = \frac{1}{3}$$

$$u_2^T v_2 = \sum_k c_k a_k^3 = \phi([\tau^3]) = \frac{1}{\gamma([\tau^3])} = \frac{1}{4}$$

hence we can generalize the result for $k > 3$ to obtain the formula:

$$\sum_{j=1}^{r} c_j a_j^{k-1} = \frac{1}{k}, \text{ for } k \le \xi$$

Remark [Butcher,1987]:

If the parameters a_2, a_3, a_4, …, c_4 are those of a four-stage fourth-order Runge-Kutta method it is possible to compute $u_i^T v_i$ for i, j = 1, 2, 3; since this will be a linear combination of the $\phi(x)$ for various x of order less than five and will thus be equal to a certain number formed from the corresponding $\gamma(x)$.

Hence the following formula $\sum_{j=1}^{r} b_{ij} a_j^{k-1} = \frac{a_j^k}{k}$ cold be obtained to evaluate a_j's for i = 1, 2, …, r and $k \le \xi$.

2.3 Derivation of Some Implicit Runge-Kutta Methods

In this section, we shall try to derive some implicit Runge-Kutta methods including some modification and new ideas for deriving.

2.3.1 Derivation of Two-Stages Implicit Runge-Kutta Method:

Consider the two-stage implicit Runge-Kutta method, with the following table of parameters:

b_{11}	b_{12}	a_1
b_{21}	b_{22}	a_2
c_1	c_2	

The Legendre polynomials will be used for deriving this method, where:

$p_0(x) = 1$, $p_1(x) = x$, $p_2(x) = (x - B_2)p_1(x) - C_2 p_0(x)$

and

$$B_2 = \frac{\int_{-1}^{1} x[p_1(x)]^2 \, dx}{\int_{-1}^{1} [p_1(x)]^2 \, dx} = \frac{\int_{-1}^{1} x^3 \, dx}{\int_{-1}^{1} x^2 \, dx} = 0$$

$$C_2 = \frac{\int_{-1}^{1} x p_1(x) p_0(x)\,dx}{\int_{-1}^{1} [p_0(x)]^2\,dx} = \frac{\int_{-1}^{1} x^2\,dx}{\int_{-1}^{1} dx} = \frac{1}{3}$$

Therefore:

$$p_2(x) = (x - 0)x - \frac{1}{3} = x^2 - \frac{1}{3}$$

hence, the roots of the second degree polynomial $p_2(x)$ at $x = 2a - 1$ are given by:

$$a_{1,2} = \frac{1}{2} \mp \frac{4}{8\sqrt{3}}$$

i.e., $a_1 = \frac{1}{2} - \frac{1}{2\sqrt{3}}$ and $a_2 = \frac{1}{2} + \frac{1}{2\sqrt{3}}$

To find c_1 and c_2, use is made as given in section (2.2), which is as follows:

$$\sum_{j=1}^{2} c_j a_j^{k-1} = \frac{1}{k}, \text{ for } k = 1, 2$$

hence for $k = 1, 2$, we have:

$$c_1 a_1^0 + c_2 a_2^0 = 1 \quad \text{..... (2.3)}$$

$$c_1 a_1^1 + c_2 a_2^1 = \frac{1}{2} \quad \text{..... (2.4)}$$

Solving eq.(2.3) and (2.4) for c_1 and c_2, gives $c_1 = c_2 = \frac{1}{2}$.

Finally, to find b_{11}, b_{12}, b_{21} and b_{22} use is made as given in section (2.2), which is as follows:

$$\sum_{j=1}^{2} b_{ij} a_j^{k-1} = \frac{a_i^k}{k}, \text{ for } i, k = 1, 2$$

Hence:

$$b_{11} a_1^0 + b_{12} a_2^0 = a_1, \; k = 1, \; i = 1$$

$$b_{21} a_1^0 + b_{22} a_2^0 = a_2^1, \; k = 1, \; i = 2$$

$$b_{11} a_1^1 + b_{12} a_2^1 = \frac{a_1^2}{2}, \; k = 2, \; i = 1$$

$$b_{21} a_1^1 + b_{22} a_2^1 = \frac{a_2^2}{2}, \; k = 2, \; i = 2$$

Solving these equations for b_{ij}'s, i, j = 1, 2, we have:

$$b_{11} = \frac{1}{4}, \; b_{12} = \frac{1}{4} - \frac{1}{2\sqrt{3}}, \; b_{21} = \frac{1}{4} + \frac{1}{2\sqrt{3}} \text{ and } b_{22} = \frac{1}{4}$$

Therefore, as a result, we have the following formula of Runge-Kutta method:

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

where:

$$k_1 = f(x_n + (\frac{1}{2} - \frac{1}{2\sqrt{3}})h, \; y_n + \frac{1}{4}hk_1 + (\frac{1}{4} - \frac{1}{2\sqrt{3}})k_2)$$

$$k_2 = f(x_n + (\frac{1}{2} + \frac{1}{2\sqrt{3}})h, \; y_n + (\frac{1}{4} + \frac{1}{2\sqrt{3}})hk_1 + \frac{1}{4}hk_2).$$

The stability of 2-stages implicit Runge-Kutta method had been discussed previously in section (1.4.3) of chapter one.

2.3.2 Derivation of Tridiagonals Three-Stages Implicit Runge-Kutta Method:

In this section, a modification is made in order to derive a new formula of triadiagonals implicit Runge-Kutta method with the property that the elements of each diagonal are equal, for simplicity, the parameters related to this method are presented in the following table:

ω	σ	0	a_1
δ	ω	σ	a_2
0	δ	ω	a_3
c_1	c_2	c_3	

Following similar approach as in section (2.3.1), one can find the values of a_1, a_2 and a_3 by solving the third degree Legendre polynomial, the obtained results are:

$$a_1 = \frac{1}{2},\ a_2 = \frac{1}{2} + \frac{\sqrt{15}}{10} \text{ and } a_3 = \frac{1}{2} - \frac{\sqrt{15}}{10}$$

Similarly, using theorem (2.2) in section (2.2), we can find c_1, c_2 and c_3, where:

$$\sum_{j=1}^{3} c_j a_j^{k-1} = \frac{1}{k}, \text{ for } k = 1, 2, 3$$

hence for k = 1, 2 and 3, we have:

$$c_1 + c_2 + c_3 = 1 \quad \text{..... (2.5)}$$

$$c_1a_1 + c_2a_2 + c_3a_3 = \frac{1}{2} \quad \text{..... (2.6)}$$

$$c_1 a_1^2 + c_2 a_2^2 + c_3 a_3^2 = \frac{1}{3} \quad \text{..... (2.7)}$$

Solving the above system for c_1, c_2 and c_3, we have:

$c_1 = 4/9$ and $c_2 = c_3 = 5/18$

Finally to find, ω, δ and σ use is made as given in section (2.2) in which the consistent equations are:

$$b_{11} a_1^{k-1} + b_{12} a_2^{k-1} + b_{13} a_3^{k-1} = \frac{a_1^k}{k}, \text{ for } i = 1$$

$$b_{21} a_1^{k-1} + b_{22} a_2^{k-1} + b_{23} a_3^{k-1} = \frac{a_2^k}{k}, \text{ for } i = 2$$

$$b_{31} a_1^{k-1} + b_{32} a_2^{k-1} + b_{33} a_3^{k-1} = \frac{a_3^k}{k}, \text{ for } i = 3$$

Hence for $k = 1$, we have:

$$\left.\begin{aligned} b_{11} + b_{12} + b_{13} &= a_1 \\ b_{21} + b_{22} + b_{23} &= a_2 \\ b_{31} + b_{32} + b_{33} &= a_3 \end{aligned}\right\} \quad \text{..... (2.8)}$$

For $k = 2$, we have:

$$\left.\begin{aligned} b_{11}a_1 + b_{12}a_2 + b_{13}a_3 &= \frac{a_1^2}{2} \\ b_{21}a_1 + b_{22}a_2 + b_{23}a_3 &= \frac{a_2^2}{2} \\ b_{31}a_1 + b_{32}a_2 + b_{33}a_3 &= \frac{a_3^2}{2} \end{aligned}\right\} \quad \text{..... (2.9)}$$

and for k = 3, we have:

$$\left.\begin{aligned} b_{11}a_1^2 + b_{12}a_2^2 + b_{13}a_3^2 &= \frac{a_1^3}{3} \\ b_{21}a_1^2 + b_{22}a_2^2 + b_{23}a_3^2 &= \frac{a_2^3}{3} \\ b_{31}a_1^2 + b_{32}a_2^2 + b_{33}a_3^2 &= \frac{a_3^3}{3} \end{aligned}\right\} \quad \text{..... (2.10)}$$

Since $b_{11} = b_{22} = b_{33} = \omega$, $b_{12} = b_{23} = \sigma$, $b_{21} = b_{32} = \delta$ and $b_{13} = b_{31} = 0$

From equations (2.8), we have:

$$\omega + \sigma + 0 = \frac{1}{2} \quad \text{..... (2.11)}$$

$$\delta + \omega + \sigma = \frac{1}{2} + \frac{\sqrt{15}}{10} \quad \text{..... (2.12)}$$

$$0 + \delta + \omega = \frac{1}{2} - \frac{\sqrt{15}}{10} \quad \text{..... (2.13)}$$

Solving equations (2.11), (2.12) and (2.13) for ω, δ and σ, we have:

$$\omega = \frac{1}{2} - \frac{\sqrt{15}}{5}, \ \sigma = \frac{\sqrt{15}}{5} \text{ and } \delta = \frac{\sqrt{15}}{10}$$

In summary, the results are given in the following table:

$\frac{1}{2}-\frac{\sqrt{15}}{5}$	$\frac{\sqrt{15}}{5}$	0	$\frac{1}{2}$
$\frac{\sqrt{15}}{10}$	$\frac{1}{2}-\frac{\sqrt{15}}{5}$	$\frac{\sqrt{15}}{5}$	$\frac{1}{2}+\frac{\sqrt{15}}{10}$
0	$\frac{\sqrt{15}}{10}$	$\frac{1}{2}-\frac{\sqrt{15}}{5}$	$\frac{1}{2}-\frac{\sqrt{15}}{10}$
$\frac{4}{9}$	$\frac{5}{18}$	$\frac{5}{18}$	

2.4 Stability of Tridiagonals Three-Stages Implicit Runge-Kutta Method:

To obtain intervals of stability of 3-stages Runge-Kutta method, we consider the test problem $y' = \lambda y$, where $Re(\lambda) < 0$. Recall the tridiagonals three steps implicit Runge-Kutta method:

$$y_{n+1} = y_n + h(c_1k_1 + c_2k_2 + c_3k_3)$$

where:

$$k_1 = \lambda(y_n + hb_{11}k_1 + hb_{12}k_2 + hb_{13}k_3)$$

$$k_2 = \lambda(y_n + hb_{21}k_1 + hb_{22}k_2 + hb_{23}k_3)$$

$$k_3 = \lambda(y_n + hb_{31}k_1 + hb_{32}k_2 + hb_{33}k_3)$$

Hence, we have:

$$k_1 = \lambda y_n + \hbar\left(\frac{1}{2} - \frac{\sqrt{15}}{5}\right)k_1 + \hbar\frac{\sqrt{15}}{5}k_2,\ \hbar = \lambda h \quad \text{..... (2.14)}$$

and

$$k_2 = \lambda y_n + \hbar \frac{\sqrt{15}}{10} k_1 + \hbar \left(\frac{1}{2} - \frac{\sqrt{15}}{5}\right) k_2 + \hbar \frac{\sqrt{15}}{5} k_3 \quad \text{..... (2.15)}$$

Also:

$$k_3 = \lambda y_n + \hbar \frac{\sqrt{15}}{10} k_2 + \hbar \left(\frac{1}{2} - \frac{\sqrt{15}}{5}\right) k_3 \quad \text{..... (2.16)}$$

Substituting equations (2.14) and (2.16) in equation (2.15), we get:

$$k_2 = \lambda y_n + \hbar \frac{\sqrt{15}}{10} \left[\frac{\lambda y_n + \hbar \frac{\sqrt{15}}{5} k_2}{1 - \left(\frac{1}{2} - \frac{\sqrt{15}}{5}\right)\hbar} \right] + \hbar \left(\frac{1}{2} - \frac{\sqrt{15}}{5}\right) k_2 +$$

$$\hbar \frac{\sqrt{15}}{5} \left[\frac{\lambda y_n + \hbar \frac{\sqrt{15}}{10} k_2}{1 - \left(\frac{1}{2} - \frac{\sqrt{15}}{5}\right)\hbar} \right]$$

Then after some simplifications, we have:

$$k_2 = \lambda y_n \left[\frac{1 - \left(\frac{1}{2} - \frac{\sqrt{15}}{2}\right)\hbar}{\left[1 - \left(\frac{1}{2} - \frac{\sqrt{15}}{5}\right)\hbar\right]^2 - \frac{3}{5}\hbar^2} \right], \left[1 - \left(\frac{1}{2} - \frac{\sqrt{15}}{5}\right)\hbar\right]^2 - \frac{3}{5}\hbar^2 \neq 0$$

Substituting k_2 in equation (2.14), yields:

$$k_1 = \frac{\lambda y_n + \lambda y_n \frac{\sqrt{15}}{5}\hbar\left[\frac{1-\left(\frac{1}{2}-\frac{\sqrt{15}}{2}\right)\hbar}{\left[1-\left(\frac{1}{2}-\frac{\sqrt{15}}{5}\right)\hbar\right]^2-\frac{3}{5}\hbar^2}\right]}{1-\left(\frac{1}{2}-\frac{\sqrt{15}}{5}\right)\hbar}$$

Then after some simplification, we get:

$$k_1 = \lambda y_n\left[\frac{1+\left(\frac{3\sqrt{15}}{5}-1\right)\hbar+\left(\frac{7}{4}-\frac{3\sqrt{15}}{10}\right)\hbar^2}{\left[1-\left(1-\frac{2\sqrt{15}}{5}\right)\hbar+\left(\frac{1}{4}-\frac{\sqrt{15}}{5}\right)\hbar^2\right]\left[1-\left(\frac{1}{2}-\frac{\sqrt{15}}{5}\right)\hbar\right]}\right],$$

$$\left[1-\left(1-\frac{2\sqrt{15}}{5}\right)\hbar+\left(\frac{1}{4}-\frac{\sqrt{15}}{5}\right)\hbar^2\right]\left[1-\left(\frac{1}{2}-\frac{\sqrt{15}}{5}\right)\hbar\right]\neq 0$$

substituting k_2 in equation (2.16)

$$k_3 = \frac{\lambda y_n + \lambda y_n \frac{\sqrt{15}}{10}\hbar\left[\frac{1-\left(\frac{1}{2}-\frac{\sqrt{15}}{2}\right)\hbar}{\left[1-\left(\frac{1}{2}-\frac{\sqrt{15}}{5}\right)\hbar\right]^2-\frac{3}{5}\hbar^2}\right]}{1-\left(\frac{1}{2}-\frac{\sqrt{15}}{5}\right)\hbar}$$

Hence, after some simplifications:

$$k_3 = \lambda y_n \left[\frac{1+\left(\frac{\sqrt{15}}{2}-1\right)\hbar+\left(1-\frac{\sqrt{15}}{4}\right)\hbar^2}{\left[1-\left(1-\frac{2\sqrt{15}}{5}\right)\hbar+\left(\frac{1}{4}-\frac{\sqrt{15}}{5}\right)\hbar^2\right]\left[1-\left(\frac{1}{2}-\frac{\sqrt{15}}{5}\right)\hbar\right]} \right],$$

$$\left[1-\left(1-\frac{2\sqrt{15}}{5}\right)\hbar+\left(\frac{1}{4}-\frac{\sqrt{15}}{5}\right)\hbar^2\right]\left[1-\left(\frac{1}{2}-\frac{\sqrt{15}}{5}\right)\hbar\right]$$

Therefore:

$$y_{n+1} = y_n + hy_n \left[\frac{4}{9} \frac{1+\left(\frac{3\sqrt{15}}{5}-1\right)\hbar+\left(\frac{7}{4}-\frac{3\sqrt{15}}{10}\right)\hbar^2}{\left[1-\left(1-\frac{2\sqrt{15}}{5}\right)\hbar+\left(\frac{1}{4}-\frac{\sqrt{15}}{5}\right)\hbar^2\right]\left[1-\left(\frac{1}{2}-\frac{\sqrt{15}}{5}\right)\hbar\right]} + \right.$$

$$\frac{5}{18} \frac{1-\left(\frac{1}{2}-\frac{\sqrt{15}}{2}\right)\hbar}{\left[1-\left(\frac{1}{2}-\frac{\sqrt{15}}{5}\right)\hbar\right]^2-\frac{3}{5}\hbar^2} +$$

$$\left. \frac{5}{18} \frac{1+\left(\frac{\sqrt{15}}{2}-1\right)\hbar+\left(1-\frac{\sqrt{15}}{4}\right)\hbar^2}{\left[1-\left(1-\frac{2\sqrt{15}}{5}\right)\hbar+\left(\frac{1}{4}-\frac{\sqrt{15}}{5}\right)\hbar^2\right]\left[1-\left(\frac{1}{2}-\frac{\sqrt{15}}{5}\right)\hbar\right]} \right]$$

Hence, as a result, we have:

$$y_{n+1} = y_n\left[1+\frac{13\hbar+\left(\frac{73\sqrt{15}}{10}-13\right)\hbar^2+\left(19-\frac{49\sqrt{15}}{20}\right)\hbar^3}{18\left[1-\left(\frac{1}{2}-\frac{2\sqrt{15}}{5}\right)\hbar+\left(\frac{1}{4}-\frac{\sqrt{15}}{5}\right)\hbar^2\right]\left[1-\left(\frac{1}{2}-\frac{\sqrt{15}}{5}\right)\hbar\right]}+\right.$$

$$\left.\frac{5\hbar+\left(\frac{5}{2}-\frac{5\sqrt{15}}{2}\right)\hbar^2}{18\left[1-\left(\frac{1}{2}-\frac{\sqrt{15}}{5}\right)\hbar\right]^2-\frac{54}{5}\hbar^2}\right],\ y_n = r^n$$

Hence the corresponding root is given by:

$$r = 1+\frac{13\hbar+\left(\frac{73\sqrt{15}}{10}-13\right)\hbar^2+\left(19-\frac{49\sqrt{15}}{20}\right)\hbar^3}{18\left[1-\left(\frac{1}{2}-\frac{2\sqrt{15}}{5}\right)\hbar+\left(\frac{1}{4}-\frac{\sqrt{15}}{5}\right)\hbar^2\right]\left[1-\left(\frac{1}{2}-\frac{\sqrt{15}}{5}\right)\hbar\right]}+$$

$$\frac{5\hbar+\left(\frac{5}{2}-\frac{5\sqrt{15}}{2}\right)\hbar^2}{18\left[1-\left(\frac{1}{2}-\frac{\sqrt{15}}{5}\right)\hbar\right]^2-\frac{54}{5}\hbar^2}$$

using computer facilities, one can find the values of $\hbar$, which satisfying $|r|<1$.

Therefore, upon applying computer programming **(ROOT-RK Program)**, the following interval of absolute stability is obtained, which is $\hbar \in (-9.5,-1) \cup (-0.637, 0)$.

CHAPTER THREE

VARIABLE STEP AND VARIABLE ORDER RUNGE-KUTTA METHODS

3.1 Introduction

Modifying the results obtained from numerical approaches is one of the fundamental aspects in numerical analysis in which the aim is to reduce the error terms imbedded in the methods, or the local trunction error.

Hence this chapter consist of introducing two fundamental approaches for reducing the error terms, which are the variable step size methods and variable order methods for the numerical solution of ODE using Runge-Kutta methods.

3.2 Variable Step Rung-Kutta Methods [James, 1992], [Jassim, 1999]

Error terms for members of the Runge-Kutta family are rather complicated than in LMMS. They become more so for higher-order methods such as classical Runge-Kutta which is locally $O(h^5)$. Fortunately it turns out that reasonably effective estimates of stepsize required to attain a specified local truncation error can be found that use only the order of the local truncation error and do not require further knowledge of the form of the error

term. The first variable-stepsize method, which will be consider here, is based upon comparison of the estimates for one and two steps for the value of y at some time obtained by a Runge-Kutta method with local truncation error term that is of the form Ch^n, where C is a constant to be determined. Suppose that one are already in position of an estimating y_x for y(x) and a candidate stepsize h_0. The Runge-Kutta method is used to calculate $y^{(1)}_{x+h_0}$ and $y^{(2)}_{x+h_0}$, approximations to $y(x+h_0)$ using stepsizes of h_0 and $h_0/2$, respectively. If $E_{est} := \left| y^{(1)}_{x+h_0} - y^{(2)}_{x+h_0} \right|$ is less than certain tolerance (tol), than the more accurate of the two approximations, $y^{(2)}_{x+h_0}$, is accepted as the approximation for $y(x + h_0)$. Whether or not the approximation is accepted, we need a new estimate h_{tol} of the stepsize that will produce an approximation within the specified tolerance. If the approximation was accepted, this value will be used as h_0 in the next step; if not, then it will be used as h_0 repeating the current step. To find h_{tol}, it is noticeable that

$$E_{est} = \left| y^{(1)}_{x+h_0} - y^{(2)}_{x+h_0} \right| \approx \left| Ch_0^n - C\left(\frac{h_0}{2}\right)^n \right| = (1-2^{-n})Ch_0^n \quad \text{..... (3.1)}$$

This gives the value of C, to be:

$$C \approx \frac{E_{est}}{(1-2^{-n})h_o^n} \quad \text{..... (3.2)}$$

Since h_{tol} is to satisfy $tol \approx Ch_{tol}^n$, then it is found that

$$h_{tol} = \left[\frac{(1-2^{-n})tol}{E_{est}}\right]^{1/n} h_0 \quad \text{..... (3.3)}$$

At the start of the application of the variable-stepsize process, we have $y_{x_0} = y_0$. Locking any better information, h_0 is taken to be x_f-x_0.

3.3 Variable Order Methods [James, 1992], [Jassim, 1999]

Let A(h) be a scheme for approximating some quantity A(0); that is, $\lim_{h\to 0} A(h) = A(0)$. The parameter h is typically the interval width. It is assumed that the error of approximation A(0) − A(h) has an expansion in powers of h whence.

$$A(0) = A(h) + a_1 h + a_2 h^2 + a_3 h^3 + a_4 h^4 + \ldots \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.4)$$

Recall that Richardson entrapolation entails using approximations $A(h_0)$, $A(h_1)$, $A(h_2)$,…with $h_0 > h_1 > h_2$… to successively eliminate the terms in the error expansion. Thereby producing approximations of higher and higher order. The sequence used was $h_j := h/2^j$, j = 0, 1,2, …, where h is some starting interval width; however, for our present purposes other sequences $\{h_j\}$ may be more advantageous. If a_1 in (3.4) is not zero, then the approximation scheme A(h) is only O(h). To obtain an $O(h^2)$ approximation we note that

$$\left.\begin{aligned} A(0) &= A(h_0) + a_1 h_0 + a_2 h_0^2 + a_3 h_0^3 + a_4 h_0^4 + \ldots \\ A(0) &= A(h_1) + a_1 h_1 + a_2 h_1^2 + a_3 h_1^3 + a_4 h_1^4 + \ldots \end{aligned}\right\} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.5)$$

Upon subtracting h_0 times the second equation from h_1 times the first and solving for A(0), one can obtain.

$$A(0) = \frac{h_1 A(h_0) - h_0 A(h_1)}{h_1 - h_0} - a_2 h_0 h_1 - a_3 h_0 h_1 (h_0 + h_1) - a_4 (h_0^2 + h_0 h_1 + h_1^2) - \ldots$$

$$= A(h_1) + \frac{A(h_1) - A(h_0)}{h_0/h_1 - 1} - a_2 h_0 h_1 - a_3 h_0 h_1 (h_0 + h_1) - a_4 (h_0^2 + h_0 h_1 + h_1^2) - \ldots \quad (3.6)$$

Thus:

$$A_1(h_0) := A(h_1) + \frac{A(h_1) - A(h_0)}{h_0/h_1 - 1} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.7)$$

Is an $O(h_0^2)$ approximation to A(0) since $h_1<h_0$. Since any pair h_j, h_{j+1} could be used in the elimination process above, thus it is easily seen that:

$$A_1(h_j) := A(h_{j+1}) + \frac{A(h_{j+1}) - A(h_j)}{h_j / h_{j+1} - 1} \quad \text{..... (3.8)}$$

is an $O(h_j^2)$ approximation to A(0). It is know have:

$$\left.\begin{aligned} A(0) &= A_1(h_0) - a_2 h_0 h_1 - a_3 h_0 h_2 (h_0 + h_1) - a_4 h_0 h_1 (h_0^2 + h_0 h_1 + h_1^2) - \ldots \\ A(0) &= A_1(h_1) - a_2 h_1 h_2 - a_3 h_1 h_2 (h_1 + h_2) - a_4 h_1 h_2 (h_1^2 + h_1 h_2 + h_2^2) - \ldots \end{aligned}\right\} \quad (3.9)$$

Upon eliminating the terms involving a_2,

$$A(0) = A_2(h_0) + a_3 h_0 h_1 h_2 + a_4 h_0 h_1 h_2 (h_0 + h_1 + h_2) + \ldots \text{..... (3.10)}$$

where:

$$A_2(h_0) := A_1(h_1) + \frac{A_1(h_1) - A_1(h_0)}{h_0 / h_2 - 1} \quad \text{..... (3.11)}$$

is an $O(h_0^3)$ approximation to A(0).

More generally:

$$A_2(h_j) := A_1(h_{j+1}) + \frac{A_1(h_{j+1}) - A_1(h_j)}{h_j / h_{j+2} - 1} \quad \text{..... (3.12)}$$

is an $O(h_j^3)$ approximation to A(0). Continuity in this manner, then recursively, a sequence can be defined by:

$$A_0(h_j) := A(h_j) \quad \text{..... (3.13)}$$

and

$$A_n(h_j) = A_{n-1}(h_{j+1}) + \frac{A_{n-1}(h_{j+1}) - A_{n-1}(h_j)}{h_j / h_{j+n} - 1}, \ n = 1,2,\ldots$$

is obtained.

On the basis of the results for $A(h_j)$ and $A_2(h_j)$, it seems that $A_n(h_j)$ provides an $O(h_j^{n+1})$ approximation to A(0). This may be verified directly by following the evaluation of the general term $a_n h^n$ in the error expansion but is perhaps obtained more easily by an alternative approach.

The recurrence relations, could be illustrated in table (3.1).

Table (3.1)

Level	$O(h_j)$	$O(h_j^2)$	$O(h_j^3)$	$O(h_j^4)$
0	$A_0(h_0)$			
1	$A_0(h_1)$	$A_1(h_0)$		
2	$A_0(h_2)$	$A_1(h_1)$	$A_2(h_0)$	
3	$A_0(h_3)$	$A_1(h_2)$	$A_2(h_1)$	$A_3(h_0)$

The Lagrange interpolating polynomial for the distinct points $(h_0,A(h_0))$, $(h_1,A(h_1)),\ldots,(h_n,A(h_n))$ is

$$P_n(h) := A[h_n] + A[h_n, h_{n-1}](h - h_n) + A[h_n, h_{n-1}, h_{n-2}](h - h_n)$$

$$(h - h_{n-1}) + \ldots.. + A[h_n, h_{n-1}, \ldots, h_0, h](h - h_n)(h - h_{n-1})\ldots$$

$$(h - h_1)(h - h_0) \qquad \text{...} \ (3.14)$$

It is easily seen that $P_n(0)=A_n(h_0)$, which is why the process is referred to as extrapolation. From the remainder term of the lagrange formula recalling that, the K-th divided difference of a function f(x) can be approximated by:

$$f[x_0, x_1, ..., x_k] = \frac{f^{(k+1)}(\zeta)}{(k+1)!}$$

for some point ζ in the interval spanned by x_0, x_1, …, x_k.

It follows that error made in approximating A(0) by $P_n(0)$ is $\frac{(-1)^{(n+1)} A^{(n+1)}(\eta)}{(n+1)!} h_0 h_1 ... h_n$ with $0 \le \eta \le h_0$ whence it is an $O(h_0^{n+1})$ approximation.

Table (3.1) shows that the extrapolation sequence generated by the formulas (3.13). Only the first column requires application of the method A(h). The higher-order refinements are generated by simple arithmetic computations and thus are inexpensive in terms of computing time.

3.3.1 Richardson Extrapolation

Let A(h) be a scheme for approximating numerically a quantity A(0) which depends upon a parameter h in such away that $\lim_{h \to 0} A(h) = A(0)$.

Suppose moreover that the error made in approximating A(0) by A(h) has for some N≥1 a power series expansion in h of the form

$$A(0) - A(h) = \sum_{j=1}^{N} a_{2j} h^{2j} + O(h^{2N+1}) \quad \text{..... (3.15)}$$

3.3.1.1 The Central Difference Formula

Consider the central difference formula for approximating the first derivative.

$$f'(x) \approx \nabla_{cent,h} f := \frac{f(x+h) - f(x-h)}{2h} \quad \text{..... (3.16)}$$

Let us show that the expansion of the error made by the central difference formula has the form (3.15). For $f \in C^{2N+1}$ we have from Taylor's theorem

$$\nabla_{cent,h} f = \frac{1}{2h}\left[\sum_{j=0}^{2N} \frac{f^{(j)}(x)h^j}{j!} + O(h^{2N+1}) - \sum_{j=0}^{2N} \frac{f^{(j)}(x)(-h)^j}{j!} + O(h^{2N+1})\right] \quad \text{..... (3.17)}$$

Observe that the even powers of h cancel where as the odd powers sum together, this gives:

$$f'(x) - \nabla_{cent,h} f = -\sum_{k=1}^{N-1} f^{(2k+1)}(x)h^{2k} + O(h^{2N}) \quad \text{..... (3.18)}$$

which is of the form (3.15).

The process of Richardson extrapolation consists of successively eliminating terms in the error expansion to produce approximations of higher order.

From (3.15), we have:

$$\begin{aligned} A(0) &= A(h) + \sum_{j=1}^{N} a_{2j} h^{2j} + O(h^{2N+1}) \\ A(0) &= A(\frac{h}{2}) + \sum_{j=1}^{N} a_{2j} (\frac{h}{2})^{2j} + O(h^{2N+1}) \end{aligned} \quad \text{..... (3.19)}$$

Multiplying the second equation in (3.19) by 4 and subtracting the first yields.

$$3A(0) = 4A\left(\frac{h}{2}\right) - A(h) + \sum_{j=2}^{N}\left(\frac{1}{2^{2j-2}} - 1\right)a_{2j}h^{2j} + O(h^{2N+1}) \text{ ..... (3.20)}$$

The multiplicative factor 4 was chosen to make the h^2 terms canceled.

Equation (3.20) shows that

$$A_1(h) := \frac{4A\left(\frac{h}{2}\right) - A(h)}{3} \text{ ..... (3.21)}$$

is an $O(h^4)$ approximation to A(0). Observe that we did not actually need to know the value of the coefficient a_2 but only that the error expansion had the form (3.15). In a similar manner, the process can be continued. From (3.20) it is know that:

$$A(0) = A_1(h) - \frac{3}{4}a_4h^4 + \ldots.$$

By eliminating the h^4 term, the order-six approximation will be obtained, given by:

$$A_2(h) = \frac{16A_1(h/2) - A_1(h)}{15} \text{ ..... (3.22)}$$

In general, one obtains recursively the $O(h^{2n+2})$ approximations

$A_0(h) := A(h)$

$$A_n(h) := \frac{4^n A_{n-1}(h/2) - A_1(h)}{4^n - 1} \text{ ..... (3.23)}$$

for the Richardson extrapolation process.

3.3.1.2 Extrapolation of The Central Difference Formula

Substituting the central difference approximation (see section 3.3.1.1), $A(h) = [f(x + h) - f(x - h)]/2h$ into (3.21), the fourth-order approximation, will be obtained as:

$$A_1(h) = \frac{f(x-h) - 8f(x-h/2) + 8f(x+h/2) - f(x+h)}{6h} \quad \text{..... (3.24)}$$

to the first derivative. In turn, substituting (3.24) into (3.22) yields the sixth-order approximation.

$$A_2(h) = \frac{1}{30h}\left[-f(x-h) + 16f\left(x-\frac{h}{2}\right) - 64f\left(x-\frac{h}{4}\right) + 64f\left(x+\frac{h}{4}\right) - 16f\left(x+\frac{h}{2}\right) + f(x+h)\right]$$

..... (3.25)

Clearly, using the extrapolation formula (3.23) to generate formulas for higher-order approximations becomes very cumbersome. Fortunately, it is not necessary to have explicit formulas available to calculate the numerical values of the higher-order approximations.

3.3.1.3 Euler-Maclaurian Summation Formula

Observe that while in (3.15),it is assumed that the error expansion had only even powers of h, this is not strictly necessary. The process could be carried out if the error expansion was of the form

$$A(0) - A(h) = \sum_{j=1}^{N} a_j h^j + O(h^{N+1}) \quad \text{..... (3.26)}$$

however, at each level of the extrapolation table the order of the approximation on the diagonal would be only one greater than on the succeeding level rather than two greater for the extrapolation table for (3.15).

For the reason extrapolation is more effective when the underlying method of approximation A(h) has an error expansion of the form (3.15).

The reason for the choice of the trapezoid rule as the base method of approximation is that its error expansion is of the form (3.15); that is, it has only even powers of h. This is the content of the Euler-Maclaurin Summation formula, which is used as well in order branches of mathematics such as number theory. The formula involves Bernoulli numbers which will be discussed shortly.

Before setting up Euler's theorem, first the following for constructing, the so called Bernoulli's numbers using recurrence relation will be introduced.

3.3.1.4 Bernoulli Numbers [Knuth, 1973]

The Bernoulli numbers are defined by

$$B_j := \lim_{t\to 0} \frac{d^j}{dt^j}\left(\frac{t}{e^t - 1}\right) \quad j = 0, 1, \ldots \quad \text{..... (3.27)}$$

From this definition they are easily seen to the numerators of the coefficients of the Maclaurin expansion

$$\frac{t}{e^t - 1} = \sum_{j=0}^{\infty} \frac{B_j}{j!} t^j \quad \text{..... (3.28)}$$

The Bernoulli numbers may be calculated from the following theorem.

Theorem (3.1)

The Bernoulli numbers satisfy the recurrence relation:

$$B_0 = 1,\ B_j = -\frac{1}{j+1}\sum_{k=0}^{j-1}\binom{j+1}{k}B_k,\ j = 1, 2,\ldots \quad \text{..... (3.29)}$$

Proof:

From (3.28), we have:

$$t = \left(\sum_{j=0}^{\infty}\frac{B_j}{j!}t^j\right)(e^t - 1)$$

$$= \left(\sum_{j=0}^{\infty}\frac{B_j}{j!}t^j\right)\left(\sum_{k=1}^{\infty}\frac{t^k}{k!}\right)$$

$$= \sum_{j=0}^{\infty}\sum_{k=0}^{\infty}\frac{B_k t^k}{k!}\frac{t^{j-k}}{(j-k)!}$$

$$= \sum_{j=1}^{\infty}t^j\sum_{k=0}^{j-1}\frac{B_k}{k!(j-k)!} \quad \text{..... (3.30)}$$

Equating like powers of t gives $B_0 = 1$ and

$$\sum_{k=0}^{j-1}\frac{B_k}{k!(j-k)!} = 0$$

for j = 1,2,3,…. Solving for B_{j-1} yields

$$B_{j-1} = -(j-1)!\sum_{k=0}^{j-2}\frac{B_k}{k!(j-k)!} = -\frac{1}{j}\sum_{k=0}^{j-2}\binom{j}{k}B_k \quad \text{..... (3.31)}$$

which is eq.(3.29).

From (3.29) it can be verified that $B_0 = 1$, $B_1 = -1/2$, $B_2 = 1/6$, $B_3 = 0$, and $B_4 = -1/30$. It turns out that $B_j = 0$ for odd $j \geq 3$. ▮ ▮

Theorem (3.2) (The Euler-Maclaurian summation formula):

For $f \in C^{2n+1}[a,b]$,

$$\int_a^b f(x)dx = h\sum_{i=0}^{N-1} f(x_i) + \frac{h}{2}[f(b)-f(a)] -$$

$$\sum_{j=1}^{n} \frac{B_{2j}h^{2j}}{(2j)!}[f^{(2j-1)}(b) - f^{(2j-1)}(a)] + O(h^{2n+1}) \quad \text{..... (3.32)}$$

where $x_i = a + ih$, $i = 0, 1\ldots, N$, with $h = (b - a)/N$, and the B_{2j} are Bernoulli numbers.

The first two terms on the right-hand side of (3.32) together constitute the composite trapezoid rule. Hence the Euler-Maclaurin formula status that the error expansion for the trapezoid rule approximation to a definite integral has the form (3.15)

$$E_{Ctrap}(f) = -\sum_{j=1}^{n} \frac{B_{2j}h^{2j}}{(2j)!}[f^{(2j-1)}(b) - f^{(2j-1)}(a)] + O(h^{2n+1}) \quad \text{..... (3.33)}$$

Proof:

Let:

$$I(f) := \int_a^b f(x)dx, \quad R_N(f) := h\sum_{i=0}^{N-1} f(x_i), \quad E_N(f) := I(f) - R_N(f)$$

Observe that $R_N(f)$ is simply the Riemann sum approximation to I(f) for N subintervals with left endpoint evaluation and that $E_N(f)$ is the error made by this approximation. To show that for $f \in C^{m+1}[a, b]$,

$$E_N(f) = -\sum_{j=1}^{m} \frac{B_j h^j}{j!} I(f^{(j)}) + O(h^{m+1}) \quad \text{..... (3.34)}$$

Upon setting m = 2n and noting that $B_1 = -1/2$, $B_{2j+1} = 0$ for $j \geq 1$, and $I(f^{(j)}) = f^{(j-1)}(b) - f^{(j-1)}(a)$, this becomes the Euler-Maclaurin formula (3.32).

Note that:

$$E_N(f) = \sum_{i=0}^{N-1} \left[\int_{x_i}^{x_i+h} f(x)dx - hf(x_i) \right]$$

$$=: \sum_{i=0}^{N-1} e_i(h) \quad \text{..... (3.35)}$$

Let us first find the Maclaurin series expansion for the error $e_i(h)$ on the ith subinterval with respect to the subinterval width h. Let $f \in C^{m+1}[a, b]$. Immediately $e_i(0) = 0$, and since $e_i'(h) = f(x_i + h) - f(x_i)$, $e_i'(0) = 0$ also. From then on $e_i^{(j)}(0) = f^{(j-1)}(x_i)$. This gives the Maclaurin expansion:

$$e_i(h) = \sum_{j=2}^{m+1} \frac{h^j}{j!} f^{(j-1)}(x_i) + \frac{h^{m+2}}{(m+2)!} f^{(m+1)}(\xi_i) \quad \text{..... (3.36)}$$

where $\zeta_i \in (x_i, x_{i+1})$. Substituting (3.36) into (3.35);

$$E_N(f) = \sum_{i=0}^{N-1} \sum_{j=2}^{m+1} \frac{h^j}{j!} f^{(j-1)}(x_i) + \sum_{i=0}^{N-1} \frac{h^{m+2}}{(m+2)!} f^{(m+1)}(\xi_i) \quad \text{..... (3.37)}$$

Now:

$$\left| \sum_{i=0}^{N-1} \frac{h^{m+2}}{(m+2)!} f^{(m+1)}(\xi_i) \right| \leq \frac{h^{m+1}}{(m+2)!} \frac{(b-a)}{N} \sum_{i=0}^{N-1} \left| f^{(m+1)}(\xi_i) \right|$$

$$\leq \frac{h^{m+1}(b-a)}{(m+1)!} \max_{a \leq x \leq b} \left| f^{(m+1)}(x) \right| \quad \text{..... (3.38)}$$

Thus the remainder term in (3.37) is $O(h^{m+1})$. Upon interchanging the order of summation (3.37), one can:

$$E_N = \sum_{j=2}^{m+1} \frac{h^j}{j!} \sum_{i=0}^{N-1} f^{(j-1)}(x_i) + O(h^{m+1})$$

$$= \sum_{j=2}^{m+1} \frac{h^{j-1}}{j!} R_N(f^{(j-1)}) + O(h^{m+1}) \quad \text{..... (3.39)}$$

After a shift of the index j this gives:

$$I(f) = R_N(f) + \sum_{j=1}^{m} \frac{h^j}{(j+1)!} R_N(f^{(j)}) + O(h^{m+1}) \quad \text{..... (3.40)}$$

Substituting I_N-E_N for R_N yields finally:

$$E_N(f) = \sum_{j=1}^{m} \frac{h^j}{(j+1)!} \left[I(f^{(j)}) - E_N(f^{(j)}) \right] + O(h^{m+1}) \quad \text{..... (3.41)}$$

Note that a particular consequence of (3.36) is that if $f \in C^1[a, b]$, then $E_N(f)$ is O(h). Thus if $f \in C^2[a, b]$, then $f' \in C^1[a, b]$ whence $E_N(f')$ is O(h), and again from (3.41), it follows that:

$$E_N(f) = \frac{h}{2} I(f') + O(h^2)$$

$$= -B_1 h + O(h^2) \quad \text{..... (3.42)}$$

which is (3.34) for m = 1.

We now establish (3.34) in general by induction on the degree of differentiability m+1. Assume that for L < m, $f \in C^{L+1}[a, b]$ implies that:

$$E_N(f) = -\sum_{k=1}^{L} \frac{B_k h^k}{k!} I(f^{(k)}) + O(h^{L+1}) \quad \text{..... (3.43)}$$

Let us show that (3.43) holds for L = m. If $f \in C^{m+1}[a,b]$, then $f^{(j)} \in C^{m-j+1}[a,b]$ whence (3.43) holds with L = m − j + 1. Substituting this into (3.41), gives:

$$E_N(f) = \sum_{j=1}^{m} \frac{h^j}{(j+1)!}\left[I(f^{(j)}) + \sum_{k=1}^{m-j} \frac{B_k h^k}{k!} I(f^{(j+k)}) \right] + O(h^{m+1})$$

$$= \sum_{j=1}^{m} \frac{h^j}{(j+1)!} I(f^{(j)}) + \sum_{j=1}^{m} \sum_{k=1}^{m-j} \frac{B_k h^{j+k}}{k!(j+1)!} I(f^{(j+k)}) + O(h^{m+1}) \quad .. (3.44)$$

Rearrangement of the double sum in (3.44) yields in turn

$$E_N(f) = \sum_{j=1}^{m} \frac{h^j}{(j+1)!} I(f^{(j)}) + \sum_{j=1}^{m} h^j I(f^{(j)}) \sum_{k=1}^{j-1} \frac{B_k}{k!(j-k+1)!} + O(h^{m+1})$$

$$= \sum_{j=1}^{m} \frac{h^j I(f^{(j)})}{(j+1)!}\left[B_0 + \sum_{k=1}^{j-1} \binom{j+1}{k} B_k \right] + O(h^{m+1})$$

$$= \sum_{j=1}^{m} \frac{h^j I(f^{(j)})}{(j+1)!}(-j!B_j) + O(h^{m+1}) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (3.45)$$

from theorem (3.1). This demonstrates (3.34) and hence the Euler-Maclaurin formula. ▮ ▮

CHAPTER FOUR

NUMERICAL AND COMPARISON RESULTS

4.1 Introduction

This chapter, is devoted for illustrating the numerical Runge-Kutta methods derived and discussed in chapter one and two, this is done by solving examples using these methods, then comparing the results with the exact solution which given have for comparison propose.

The results are presented and tabulated in a table which consists also the error terms.

In addition, numerical examples illustrating variable order and variable stepsize methods discussed in chapter three are present, with its comparison with the exact solution.

4.2 Numerical Examples

Example (4.1):

Consider the first ordering differential equation:

$$y' = -y + x + 1$$

with initial solution $y(0) = 1$.

In order to give a comparison and describe the precision of the previously derived methods of Runge-Kutta, we can easily find the exact solution:

$$y(x) = e^{-x} + x$$

Therefore using the 2-stage explicit, 2-stage semi-explicit, 2-stage implicit and tri-diagonal implicit Runge-Kutta methods, and upon exciting the computer programs (ERK), (SERK), (2IRK) and (3IRK), we get the results presented in tables (4.1) and (4.2) with step lengths $h = 0.1$ and $h = 0.01$.

One can see from error estimation of the results that (2-stage implicit) is the more accurate. Also three stages implicit tri-diagonal gives reasonable agreement exact solution.

Table (4.1) Numerical results of example (4.1) with step length h =0.1.

x_i	*Exact*	*Explicit*		*Semi-explicit*		*Two stages implicit*		*Three stages implicit*	
		Numeric solution	*Error*	*Numerical solution*	*Error*	*Numerical solution*	*Error*	*Numerical solution*	*Error*
0.0	1.00000000	1.00000000	0.00000000	1.00000000	0.00000000	1.00000000	0.00000000	1.00000000	0.00000000
0.1	1.00483741	1.00500000	0.00016258	1.00482757	0.00000984	1.00483743	0.00000001	1.00466161	0.00017580
0.2	1.01873075	1.01902500	0.00029424	1.01871293	0.00001781	1.01873077	0.00000002	1.01841263	0.00031811
0.3	1.04081822	1.04121762	0.00039940	1.04079403	0.00002418	1.04081825	0.00000003	1.04038650	0.00043171
0.4	1.07032004	1.07080195	0.00048190	1.07029087	0.00002917	1.07032008	0.00000003	1.06979924	0.00052079
0.5	1.10653065	1.10707576	0.00054510	1.10649766	0.00003299	1.10653070	0.00000004	1.10594167	0.00058898
0.6	1.14881163	1.14940356	0.00059193	1.14877580	0.00003582	1.14881168	0.00000004	1.14817217	0.00063946
0.7	1.19658530	1.19721022	0.00062492	1.19654748	0.00003782	1.19658535	0.00000004	1.19591032	0.00067498
0.8	1.24932896	1.24997525	0.00064629	1.24928985	0.00003911	1.24932901	0.00000005	1.24863103	0.00069793
0.9	1.30656965	1.30722760	0.00065794	1.30652984	0.00003981	1.30656971	0.00000005	1.30585927	0.00071038
1.0	1.36787944	1.36854098	0.00066154	1.36783941	0.00004002	1.36787949	0.00000005	1.36716530	0.00071413

Table (4.2) Numerical results of example (4.1) with step length h =0.01.

x_i	*Exact*	*Explicit*		*Semi-explicit*		*Two stages implicit*		*Three stages implicit*	
		Numeric solution	*Error*	*Numerical solution*	*Error*	*Numerical solution*	*Error*	*Numerical solution*	*Error*
0.0	1.00000000	1.00000000	0.00000000	1.00000000	0.00000000	1.00000000	0.00000000	1.00000000	0.00000000
0.1	1.00483741	1.00483893	0.00000151	1.00483731	0.00000010	1.00483741	0.00000000	1.00483581	0.00000159
0.2	1.01873075	1.01873350	0.00000274	1.01873057	0.00000018	1.01873075	0.00000000	1.01872785	0.00000289
0.3	1.04081822	1.04082195	0.00000373	1.04081797	0.00000024	1.04081822	0.00000000	1.04081429	0.00000392
0.4	1.07032004	1.07032454	0.00000450	1.07031974	0.00000029	1.07032004	0.00000000	1.07031530	0.00000473
0.5	1.10653065	1.10653575	0.00000509	1.10653032	0.00000033	1.10653065	0.00000000	1.10652530	0.00000535
0.6	1.14881163	1.14881716	0.00000552	1.14881126	0.00000036	1.14881163	0.00000000	1.14880581	0.00000581
0.7	1.19658530	1.19659114	0.00000583	1.19658491	0.00000038	1.19658530	0.00000000	1.19657916	0.00000614
0.8	1.24932896	1.24933500	0.00000603	1.24932856	0.00000040	1.24932896	0.00000000	1.24932261	0.00000635
0.9	1.30656965	1.30657580	0.00000614	1.30656925	0.00000040	1.30656965	0.00000000	1.30656319	0.00000646
1.0	1.36787944	1.36788561	0.00000617	1.36787903	0.00000040	1.36787944	0.00000000	1.36787294	0.00000650

Example (4.2):

Consider the first ordering differential equation:

$$y' = -2y+2x^2+2x$$

with initial solution $y(0) = 1$.

In order to give a comparison and describe the precision of the previously derived methods of Runge-Kutta, we can easily find the exact solution

$$y(x) = e^{2x}+x^2$$

Therefore using the 2-stage explicit, 2-stage semi-explicit, 2-stage implicit and tri-diagonals implicit Runge-Kutta methods, and upon exciting the computer programs (ERK), (SERK), (2IRK) and (3IRK), we get the results presented in tables (4.3) and (4.4) with step lengths $h = 0.1$ and $h = 0.01$.

One can see from error estimation of the results that (2-stage implicit) is the more accurate. Also three stages implicit tri-diagonal gives reasonable agreement exact solution.

Table (4.3) Numerical results of example (4.2) with step length h =0. 1.

x_i	*Exact*	*Explicit*		*Semi-explicit*		*Two stages implicit*		*Three stages implicit*	
		Numeric solution	*Error*	*Numerical solution*	*Error*	*Numerical solution*	*Error*	*Numerical solution*	*Error*
0.0	1.00000000	1.00000000	0.00000000	1.00000000	0.00000000	1.00000000	0.00000000	1.00000000	0.00000000
0.1	0.82873075	0.83100000	0.00226925	0.82862297	0.00010779	0.82873112	0.00000036	0.82647915	0.00225160
0.2	0.71032005	0.71422000	0.00389995	0.71013659	0.00018346	0.71032064	0.00000060	0.70648957	0.00383048
0.3	0.63881164	0.64386040	0.00504876	0.63857652	0.00023512	0.63881237	0.00000073	0.63390472	0.00490692
0.4	0.60932896	0.61516553	0.00583656	0.60905998	0.00026898	0.60932976	0.00000080	0.60371746	0.00561151
0.5	0.61787944	0.62423573	0.00635629	0.61758963	0.00028981	0.61788026	0.00000082	0.61183550	0.00604394
0.6	0.66119421	0.66787330	0.00667909	0.66089300	0.00030122	0.66119502	0.00000081	0.65491428	0.00627993
0.7	0.73659696	0.74345611	0.00685914	0.73629104	0.00030593	0.73659773	0.00000077	0.73022029	0.00637667
0.8	0.84189652	0.84883401	0.00693749	0.84159052	0.00030600	0.84189724	0.00000072	0.83551949	0.00637703
0.9	0.97529889	0.98224389	0.00694500	0.97499593	0.00030296	0.97529955	0.00000066	0.96898602	0.00631287
1.0	1.13533528	1.14223999	0.00690470	1.13503735;	0.00029793	1.13533589	0.00000060	1.12912761	0.00620768

Table (4.4) Numerical results of example (4.2) with step length h =0.01.

x_i	*Exact*	*Explicit*		*Semi-explicit*		*Two stages implicit*		*Three stages implicit*	
		Numeric solution	*Error*	*Numerical solution*	*Error*	*Numerical solution*	*Error*	*Numerical solution*	*Error*
0.0	1.00000000	1.00000000	0.00000000	1.00000000	0.00000000	1.00000000	0.00000000	1.00000000	0.00000000
0.1	0.82873075	0.82875099	0.00002024	0.82872962	0.00000113	0.82873075	0.00000000	0.82871258	0.00001817
0.2	0.71032005	0.71035484	0.00003480	0.71031812	0.00000192	0.71032005	0.00000000	0.71028912	0.00003093
0.3	0.63881164	0.63885671	0.00004507	0.63880917	0.00000247	0.63881164	0.00000000	0.63877200	0.00003964
0.4	0.60932896	0.60938110	0.00005214	0.60932614	0.00000282	0.60932896	0.00000000	0.60928362	0.00004535
0.5	0.61787944	0.61793626	0.00005682	0.61787640	0.00000304	0.61787944	0.00000000	0.61783058	0.00004886
0.6	0.66119421	0.66125396	0.00005975	0.66119105	0.00000316	0.66119421	0.00000000	0.66114343	0.00005078
0.7	0.73659696	0.73665838	0.00006141	0.73659376	0.00000321	0.73659696	0.00000000	0.73654538	0.00005158
0.8	0.84189652	0.84195869	0.00006217	0.84189331	0.00000321	0.84189652	0.00000000	0.84184493	0.00005159
0.9	0.97529889	0.97536118	0.00006229	0.97529571	0.00000318	0.97529889	0.00000000	0.97524781	0.00005108
1.0	1.13533528	1.13539727	0.00006199	1.13533216	0.00000312	1.13533528	0.00000000	1.13528505	0.00005024

Example (4.3):

Suppose one have to approximate the solution of y(0.3) to within an accuracy of tol = 0.05 for the initial value problem $y' = 5(x - 1)y$, $y(0) = 5$. The Runge-Kutta method which will be used, and hence n = 2 in (3.3).

The process is started with $y_0 = 5$, $h_0 = 0.3$. Thus we have:

Step1: First try:

$x = 0$, $h_0 = 0.3$, $x + h_0 = 0.3$

Applying the Runge-Kutta method with one and than two steps gives

$$y_{0.3}^{(1)} = 2.5626,\ y_{0.3}^{(2)} = 1.598,\ E_{est} = 0.9645$$

which is still slightly above the specified tolerance On the basis of the new value of E_{est}, formula (3.3) now predicts that $h_{tol} = 0.0592$ will suffice. This gives:

Second try:

$x = 0$, $h_0 = 0.0592$, $x + h_0 = 0.0592$

The Runge-Kutta method approximations are

$$y_{0.0592}^{(1)} = 3.7699,\ y_{0.0592}^{(2)} = 3.7558,\ E_{est} = 0.0141$$

and thus the estimated error is now within the given tolerance. Hence we have the approximation $y(0.0592) \approx 3.7558$. In fact, from the actual solution $y(0.0592) = 3.7517$, and thus the true error, 0.004, is smaller than the estimated error E_{est}. On the basis of the new value of E_{est}, formula (3.3) predicts that a stepsize of $h_{tol} = 0.0965$ is required to attain the error tolerance. We use this as the initial h_0 in the next step.

Step.2: First try:

$x = 0.0592$, $h_0 = 0.0965$, $x + h_0 = 0.1557$

The Runge-Kutta approximations are

$$y_{0.1557}^{(1)} = 2.4856,\ y_{0.1557}^{(2)} = 2.45,\ E_{est} = 0.0356$$

and thus the tolerance is achieved the first try. Hence y(0.1557) □ 2.45, and formula (3.3) gives the estimate $h_{tol} = 0.0991$ for use in the next step. The actual value of the solution is y(0.1557) = 2.4389, and thus at this point we are in error by 0.011.

Step.3: First try:

$$x = 0.1557,\ h_0 = 0.0991,\ x + h_0 = 0.2548$$

The Runge-Kutta approximations are

$$y_{0.2548}^{(1)} = 1.6744,\ y_{0.2548}^{(2)} = 1.6567,\ E_{est} = 0.0177$$

and the error estimate is again within tolerance. Formula (3.3) suggests a stepsize of 0.1442; however, the distance to $x_f = 0.3$ is only 0.0452.

Thus we use the stepsize $h_0 = 0.0452$.

Step.4: First try:

$$x = 0.2548,\ h_0 = 0.0452,\ x + h_0 = 0.3$$

The Runge-Kutta approximations are

$$y_{0.3}^{(1)} = 1.4082,\ y_{0.3}^{(2)} = 1.4073,\ E_{est} = 0.0009$$

and we obtain the approximation y(0.3)□ 1.4073, which is in error by 0.01.

When it is specified that an appropriate solution is desired with an error no more than 0.05, this is, of course, a statement concerning the global truncation error, while the choice (3.3) of h_{tol} is made to control the local truncation error at each step.

Thus it is fat from certain that we will obtain the specified accuracy. However, note that while h_{tol} is chosen so that the one-step approximation

$y^{(1)}_{x+h_0}$ is accurate to within the given tolerance, the two-step approximation $y^{(2)}_{x+h_0}$ which is more accurate by a factor of 2^n, is actually used. This help to compensate for the fact that we are controlling local, rather than global, error.

Example (4.4):

Suppose one have to approximate the solution of y(0.5) to within an accuracy of tol = 0.05 for the initial value problem $y' = 5x - 2y$, $y(0) = 1$. The Runge-Kutta method which will be used, and hence n = 2 in (3.3).

The process is started with $y_0 = 1$, $h_0 = 0.5$. Thus we have

Step.1: First try:

$x = 0, h_0 = 0.5, x + h_0 = 0.5$

Applying the Runge-Kutta method with one and than two steps gives:

$$y^{(1)}_{0.5} = 1.125,\ y^{(2)}_{0.5} = 0.8789,\ E_{est} = 0.2461$$

Which is still slightly above the specified tolerance On the basis of the new value of E_{est}, formula (3.3) now predicts that $h_{tol} = 0.1952$ will suffice. This gives

Second try:

$x = 0, h_0 = 0.1952, x+h_0 = 0.1952$

The Runge-Kutta method approximations are

$$y^{(1)}_{0.1952} = 0.7811,\ y^{(2)}_{0.1952} = 0.7652,\ E_{est} = 0.0159$$

and thus the estimated error is now within the given tolerance. Hence we have the approximation $y(0.1952) \approx 0.7652$. In fact, from the actual

solution. y(0.1952) = 0.7608, and thus the true error, 0.004, is smaller than the estimated error E_{est}. On the basis of the new value of E_{est}, formula (3.3) predicts that a stepsize of $h_{tol} = 0.2998$ is required to attain the error tolerance. We use this as the initial h_0 in the next step.

Step.2: First try:

$$x = 0.1952,\ h_0 = 0.2998,\ x + h_0 = 0.495$$

The Runge-Kutta approximations are

$$y^{(1)}_{0.495} = 0.8735,\ y^{(2)}_{0.495} = 0.8355,\ E_{est} = 0.038$$

and the error estimate is again within tolerance. Formula (3.3) suggests a stepsize of 0.2978, however, the distance to $x_f = 0.5$ is only 0.005.

Thus we use the stepsize $h_0 = 0.005$.

Step.3: First try:

$$x = 0.495,\ h_0 = 0.005,\ x + h_0 = 0.5$$

The Runge-Kutta approximations are

$$y^{(1)}_{0.5} = 0.8396,\ y^{(2)}_{0.5} = 0.8395,\ E_{est} = 0.0001$$

and we obtain the approximation y(0.5) □ 0.8395, which is in error by 0.012.

Example (4.5):

Consider the following differential equation $y' = f(x, y)$, $y(0) = 5$ where $f(x, y) = 5(x - 1)y$. The process is started with $y_0 = 5$, $h_0 = 0.1$. Thus we have

$$x = 0,\ h_0 = 0.1,\ x + h_0 = 0.1$$

Applying the Runge-Kutta method with one and then two steps gives:

$$y_{0.1}^{(1)} = 3.1875,\ y_{0.1}^{(2)} = 3.1262$$

Substituting $y_{0.1}^{(1)}$ and $y_{0.1}^{(2)}$ into eq.(3.7), we have:

$$y(x_n) = 3.064$$

with estimated error equals to 0.045.

Example (4.6):

Consider the following differential equation $y' = f(x, y)$, $y(0) = 1$ where $f(x, y) = 5x - 2y$. The process is started with $y_0 = 1$, $h_0 = 0.3$. Thus, we have:

$$x = 0,\ h_0 = 0.3,\ x + h_0 = 0.3$$

Applying the Runge-Kutta method with one and then two steps gives

$$y_{0.3}^{(1)} = 0.805,\ y_{0.3}^{(2)} = 0.7488$$

Substituting $y_{0.3}^{(1)}$ and $y_{0.3}^{(2)}$ into eq.(3.7), we have:

$$y(x_n) = 0.692$$

with estimated error equals to 0.043.

Conclusions and Recommendation

From the present study of this thesis we conclude and recommend the following

1. The 2-stages implicit Runge-Kutta method is the most accurate method than other Runge-Kutta method.
2. The improved tridiagonal method is so easy to drive which are indeed implicit method and therefore to drive improved method with five diagonal and proving its stability.
3. Variable step size and order methods are the most accurate methods which had reduce the error bounds.
4. Comparing between variable order and variable step size Runge-Kutta methods.
5. Using Runge-Kutta method for solving delay differential equations.

Contents

الأسم: أرشد أدهم أحمد خليل الدليمي.
Arshed Adham Ahmed khalil Al-Dulaimee
العنوان:ديالى-الخالص-هبهب-حي الأملاك
محلة:٢٠، زقاق:٦، دار:٤٢
موعد المناقشة: ٢٠٠٥/٥/٣
أسم الأطروحة:
(A Novel Approach for Deriving Some Runge-Kutta Methods)
Emil: ars-4280@ yahoo.com

Introduction

There is no general agreement on how the phrase "numerical analysis" should be interpreted. Some see “analysis” as the key word and wish to embed the subject entirely in rigorous modern analysis, others suggests that the "numerical" is the vital word and the algorithm is the only respectable yield. Numerical methods usually produces errors and we say that any numerical technique is a good one if the error approach quickly or rapidly to zero and the method requires a minimum computer capacity and less time consuming as possible.

In the eighteenth century, mathematicians encountered difference differential equations because they were trying to extend their knowledge of the mechanics of discrete particles to the mechanics of the continuum, which later came to be studied in terms of partial differential equations.

On the other hand, many complicated physical problems describable in terms of partial differential equations can be approximated by much simpler problems describable in terms of difference differential equations, [Piney, 1959].

This thesis, consist of four chapter. In chapter one, we introduce general Runge-Kutta methods for solving ordinary differential equations which consist of three types explicit, semi-explicit and implicit as well as there are mathematical derivation.

Also in this chapter we stay the stability and converges of the prove of proving the equivalence between stability of Runge-Kutta methods.

In chapter two, we present some fundamental results concerning with Runge-Kutta methods (see Theorem (1.1), Lemma (2.1) and Theorem (2.2)) which are needed in the derivation of Runge-Kutta methods deterministic. Also, in this chapter we present the derivation of a modified Runge-Kutta method which is three steps Runge-Kutta methods using tridiagonal systems as well as studying the stability of the modified method.

Chapter three consist of variable step size and variable order Runge-Kutta methods, which has the utility of reducing local truncation error at the node points.

Chapter four presents numerical examples, which had been solved using explicit, semi-explicit and implicit and in proved Runge-Kutta method as well as there are comparisons with the exact solution in order describe the accuracy of the methods.

It’s important, to note that the results are given in tabulated form and the programs (given in PASCAL language) are executed in personal computer Pentium IV.

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

وَمَا تَوْفِيقِيٓ إِلَّا بِٱللَّهِ ۚ عَلَيْهِ تَوَكَّلْتُ وَإِلَيْهِ أُنِيبُ ﴿٨٨﴾

صدق الله العظيم

هود-٨٨

الأسم: أرشد أدهم أحمد خليل الدليمي.
Arshed Adham Ahmed khalil Al-Dulaimee
العنوان:ديالى-الخالص-هبهب-حي الأملاك
موعد المناقشة: ٢٠٠٥/٥/٣
أسم الأطروحة:
(A Novel Approach for Deriving Some Runge-Kutta Methods)
Emil: ars-4280@ yahoo.com

References

- Butcher J.C., "The Numerical Analysis of Ordinarily Differential Equations", John Wiley, and Sons, Ltd., 1987.
- James L. Buchanan and Peter R. Turner," Numerical Methods and Analysis", McGraw-Hill, Inc., 1992.
- Lambert J.D.," Computational Methods in Ordinary Differential Equations", John Wiley and Son, Ltd., 1973.
- Burden R.L and Faires J.D., "Numerical Analysis", 3rd Edition, PWS, 1985.
- Atkinson K.E., "An Introduction to Numerical Analysis", John Wiley and Sons, Inc., 1989.
- Bellman, R. and Cooke, K.L., "Differential Equations", Academic press Inc., New York, 1963.
- Brauer, F. and Nohel, J.A., "Ordinary Differential Equations", W.A. Benjamin, Inc., 1973.
- Henrici R.P., "Discrete Variable Methods in Ordinary Differential Equations", John Wiley and Sons Inc., 1962.
- Jassim, F.A., "Variable Order and Variable step-size Methods for Initial Value Problem", M.Sc. Thesis, Department of Mathematics, Al-Nahrain University, Baghdad, Iraq, 1999.
- Isaacson, E. and H.B. Keller, "Analysis of Numerical Methods", John Wiley and Sons Inc., 1966.
- Pinney, E., "Ordinary Differential Equation", University of California, 1959.

- Butcher J.C., "Implicit Runge-Kutta Processes", J. of Math. Comp., V.18, pp. 50-64, 1964.
- AL-Exander. R. "Diagonally Implicit Runge-Kutta Methods for Stiff Ordinary Differential Equations", Siam. J. Numer Anal. V.14, No.6, pp. 1006-1021, 1977.
- AL-Kubeisy S.W., "Numerical Solution of Delay Differential Equations Using Linear Multistep Methods", M.Sc. Thesis, Department of Mathematics, Al-Nahrain University, Baghdad, Iraq, 2004.
- Smith G.D., "Numerical Solution of Partial Differential Equations", Larendon Press. Oxford, 1978.
- Ames W.F., "Numerical Methods for Partial Differential Equations", Academic Press, New York San Francisco, 1977.
- Gear C. W.,"Numerical Initial Value Problems in Ordinary Differential Equations",Prentice-Hall,Inc. Englewood Cliffs, New Jersey,1971.
- Stetter H. J. ,"Analysis of Discretization Methods for Ordinary Differential Equations",Springer-Verlag Berlin Heidelberg New York,1973.
- Rainville D.Earl,"Elementary Differential Equations", Macmillan Publishing Company, A Division of Maemillan, Inc., 1989.
- Knuth D. E.,"The Art of Computer Programming", Addison-Wesley Publishing Company, Inc., 1973.

Supervisor Certification

I certify that this thesis was prepared under our supervision at the Department of Mathematics and Computer Application, Collage of Science / Al-Nahrain University as partial fulfillment of the requirements of the degree of Master of Science in Applied Mathematics.

Signature:
Name: **Dr. Fadhel Subhi Fadhel**
Date: **/ /2005**

Signature:
Name: **Dr. Akram M. Al-Abood**
Date**: / /2005**

In view of the available recommendation, I forward this thesis for debate by the examination committee.

Signature:
Name: **Assist. Prof. Dr. Akram M. Al-Abood**
Title: **Chairman of Mathematics Department.**
Date: **/ /2005**

Ministry of Higher Education
and Scientific Research
Al-Nahrain University
College of Science

A Novel Approach for Deriving Some Runge-Kutta Methods

A Thesis

Submitted to the Collage of Science, Al-Nahrain University as a Partial Fulfillment of the Requirements for the Degree of Master of Science in Mathematics and Computer Applications

By

Arshed Adham Ahmed

(B.Sc. 2002)

Supervised by

Dr. Fadhel S. Fadhel and Dr. Akram M. Al-Abood

March 2005

وزارة التعليم العالي والبحث العلمي
جامعـــة النهرين
كلية العلوم

أسلوب مطور لأشتقاق بعض طرائق رانك كوتا

رسالة
مقدمة إلى كلية العلوم في جامعة النهرين
وهي جزء من متطلبات نيل درجة ماجستير علوم
في علوم الرياضيات وتطبيقات الحاسوب

من قبل
أرشد أدهم أحمد
(بكالوريوس علوم، ٢٠٠٢)

بإشراف
د. فاضل صبحي فاضل د.أكرم محمد العبود

آذار ٢٠٠٥