AN EFFICIENT IMPLEMENTATION OF RUNGE-KUTTA GAUSS METHODS USING VARIABLE STEPSIZE SETTING

SARA SYAHRUNNISAA BINTI MUSTAPHA

05-4506832 pustaka.upsi.edu.my Perpustakaan Tuanku Bainun Kampus Sultan Abdul Jalil Shah PustakaTBainun ptbupsi

UNIVERSITI PENDIDIKAN SULTAN IDRIS

2021

05-4506832 pustaka.upsi.edu.my Perpustakaan Tuanku Bainun Kampus Sultan Abdul Jalil Shah PustakaTBainun ptbupsi

AN EFFICIENT IMPLEMENTATION OF RUNGE-KUTTA GAUSS METHODS USING VARIABLE STEPSIZE SETTING

SARA SYAHRUNNISAA BINTI MUSTAPHA

05-4506832 pustaka.upsi.edu.my Perpustakaan Tuanku Bainun Kampus Sultan Abdul Jalil Shah PustakaTBainun ptbupsi

DISSERTATION SUBMITTED IN FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE (MATHEMATICS) (MODE BY RESEARCH)

FACULTY OF SCIENCE AND MATHEMATICS
UNIVERSITI PENDIDIKAN SULTAN IDRIS

2021

05-4506832 pustaka.upsi.edu.my Perpustakaan Tuanku Bainun Kampus Sultan Abdul Jalil Shah PustakaTBainun ptbupsi

UNIVERSITI PENDIDIKAN SULTAN IDRIS
SULTAN IDRIS EDUCATION UNIVERSITY

Please tick (√)

Project Paper	
Masters by Research	✓
Master by Mixed Mode	
PhD	

INSTITUTE OF GRADUATE STUDIES

DECLARATION OF ORIGINAL WORK

This declaration is made on the ……16……day of……02……20……21……

i. Student's Declaration:

I, SARA SYAHRUNNISAA BINTI MUSTAPHA, M20181001455, FACULTY OF SCIENCE AND MATHEMATICS (PLEASE INDICATE STUDENT'S NAME, MATRIC NO. AND FACULTY) hereby declare that the work entitled AN EFFICIENT IMPLEMENTATION OF RUNGE-KUTTA GAUSS METHODS USING VARIABLE STEPSIZE SETTING is my original work. I have not copied from any other students' work or from any other sources except where due reference or acknowledgement is made explicitly in the text, nor has any part been written for me by another person.

Signature of the student

ii. Supervisor's Declaration:

I ASSOC. PROF. DR. ANNIE GORGEY (SUPERVISOR'S NAME) hereby certifies that the work entitled AN EFFICIENT IMPLEMENTATION OF RUNGE-KUTTA GAUSS METHODS USING VARIABLE STEPSIZE SETTING (TITLE) was prepared by the above named student, and was submitted to the Institute of Graduate Studies as a * partial/full fulfillment for the conferment of DEGREE OF MASTER OF SCIENCE (MATHEMATICS) (PLEASE INDICATE THE DEGREE), and the aforementioned work, to the best of my knowledge, is the said student's work.

16/02/2021

Date

ASSOC. PROF DR. ANNIE GORGEY
LECTURER
MATHEMATICS DEPARTMENT
FACULTY OF SCIENCE & MATHEMATICS
SULTAN IDRIS EDUCATION UNIVERSITY

Signature of the Supervisor

INSTITUT PENGAJIAN SISWAZAH /
INSTITUTE OF GRADUATE STUDIES

BORANG PENGESAHAN PENYERAHAN TESIS/DISERTASI/LAPORAN KERTAS PROJEK
DECLARATION OF THESIS/DISSERTATION/PROJECT PAPER FORM

Tajuk / *Title*: AN EFFICIENT IMPLEMENTATION OF RUNGE-KUTTA GAUSS METHODS USING VARIABLE STEPSIZE SETTING

No. Matrik /*Matric's No.*: M20181001455

Saya / *I* : SARA SYAHRUNNISAA BINTI MUSTAPHA

(Nama pelajar / *Student's Name*)

mengaku membenarkan Tesis/Disertasi/Laporan Kertas Projek (Kedoktoran/Sarjana)* ini disimpan di Universiti Pendidikan Sultan Idris (Perpustakaan Tuanku Bainun) dengan syarat-syarat kegunaan seperti berikut:-
acknowledged that Universiti Pendidikan Sultan Idris (Tuanku Bainun Library) reserves the right as follows:-

1. Tesis/Disertasi/Laporan Kertas Projek ini adalah hak milik UPSI.
   The thesis is the property of Universiti Pendidikan Sultan Idris
2. Perpustakaan Tuanku Bainun dibenarkan membuat salinan untuk tujuan rujukan dan penyelidikan.
   Tuanku Bainun Library has the right to make copies for the purpose of reference and research.
3. Perpustakaan dibenarkan membuat salinan Tesis/Disertasi ini sebagai bahan pertukaran antara Institusi Pengajian Tinggi.
   The Library has the right to make copies of the thesis for academic exchange.
4. Sila tandakan (√) bagi pilihan kategori di bawah */ Please tick (√) for category below:-*

☐	**SULIT/*CONFIDENTIAL***	Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub dalam Akta Rahsia Rasmi 1972. / *Contains confidential information under the Official Secret Act 1972*
☐	**TERHAD/*RESTRICTED***	Mengandungi maklumat terhad yang telah ditentukan oleh organisasi/badan di mana penyelidikan ini dijalankan. / *Contains restircted information as specified by the organization where research was done.*
☑	**TIDAK TERHAD / *OPEN ACCESS***	

(Tandatangan Pelajar/ Signature)

ASSOC. PROF DR. ANNIE GORGEY
LECTURER
MATHEMATICS DEPARTMENT
FACULTY OF SCIENCE & MATHEMATICS
SULTAN IDRIS EDUCATION UNIVERSITY

(Tandatangan Penyelia / *Signature of Supervisor*)
& (Nama & Cop Rasmi / *Name & Official Stamp*)

Tarikh: 16/02/2021

Catatan: Jika Tesis/Disertasi ini **SULIT @ TERHAD**, sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali sebab dan tempoh laporan ini perlu dikelaskan sebagai **SULIT** dan **TERHAD**.

Notes: If the thesis is CONFIDENTAL or RESTRICTED, please attach with the letter from the organization with period and reasons for confidentiality or restriction.

ACKNOWLEDGEMENT

First and foremost, all praises to Allah S.W.T., the most Gracious and the most Merciful for the strengths and His blessing in completing this research. In the process of completing this research, I realized how true this gift of writing is for me. You give me the power to believe in my passion and I could never have done this without the faith I have in you, the Almighty.

Special appreciation goes to my supervisor, Assoc. Prof. Dr. Annie Gorgey as my supervisor for her patience, motivation and constant support in supervising this dissertation. Her invaluable guidance of constructive comments and suggestions throughout the experimental and dissertation works have contributed to the success of this research. This dissertation might not be able to complete on time without her continuous support. I am sincerely appreciates for all the supervision that I received from her and no other words to describe how grateful I am to have her as my supervisor. Also, I would like to thank Dr. Nor Azian Aini Mat as my co-supervisor for her valuable motivation and diligent support throughout my research journey.

Not forgotten, my deepest gratitude goes to my family members especially my mum for their endless love, prayers and encouragement. Last but not least, my sincere thanks goes to all my friends for the kindness and moral support throughout my Master studies. Their time spent with me really helped me to alleviate the stress and cherishes me whenever I encountered some problems. To those who indirectly contributed in this research, your kindness means a lot to me. Thank you very much.

ABSTRACT

The research is aimed to find the most efficient implementation strategies by Gauss numerical methods for solving stiff problems and the best error estimation in the variable stepsize setting. The numerical methods considered as a research methodology are the 2-stage (G2) and 3-stage (G3) implicit Runge-Kutta Gauss methods. Two strategies by Hairer and Wanner (HW) and Gonzalez-Pinto, Montijano and Randez (GMR) schemes were implemented. The variable stepsize setting employed the simplified Newton is modified to fit according to HW and GMR schemes in solving the nonlinear algebraic systems of the equations. The error estimation for the variable stepsize setting is computed using extrapolation technique with stepsizes h and $h/2$. HW and GMR schemes used the transformation matrix T to improve the efficiency of the methods and also compared with the modified Hairer and Wanner (MHW) scheme without using any transformation matrix T. Findings showed that G2 method using MHW scheme gave an efficient implementation in solving Kaps, Oreganator and HIRES problems while for G3 method, it was efficient in solving Kaps, Brusselator, Oreganator, Van der Pol and HIRES problems. In terms of error estimation, the G2 method gave the best error estimation for Brusselator, Oreganator, Van der Pol and HIRES problems, while for the G3 method it was efficient in solving Kaps, Brusselator, Oreganator, Van der Pol and HIRES problems, both by using HW scheme. In conclusion, the MHW scheme without any transformation matrix T can be as efficient as the HW and GMR schemes by using the variable stepsize setting and the MHW scheme is recommended in solving stiff problems. As for the implications, this research could be extended to other different types of problems such as delay and fuzzy differential equations.

KECEKAPAN PELAKSANAAN BAGI KAEDAH RUNGE-KUTTA GAUSS MENGGUNAKAN TETAPAN SAIZ LANGKAH BERUBAH-UBAH

ABSTRAK

Kajian ini bertujuan untuk mencari strategi pelaksanaan yang paling cekap dengan kaedah numerik Gauss untuk menyelesaikan masalah kaku dan anggaran ralat terbaik dalam tetapan saiz langkah berubah-ubah. Kaedah berangka yang dianggap sebagai metodologi kajian adalah kaedah Runge-Kutta Gauss tahap-2 (G2) dan tahap-3 (G3) tersirat. Dua strategi oleh Hairer dan Wanner (HW) dan Gonzalez-Pinto, Montijano dan Randez (GMR) dilaksanakan. Pengaturan saiz langkah berubah-ubah menggunakan Newton yang dipermudah diubah suai agar sesuai dengan skim HW dan GMR dalam menyelesaikan sistem persamaan algebra tidak linear. Anggaran ralat untuk tetapan saiz langkah berubah-ubah dikira menggunakan teknik ekstrapolasi dengan saiz langkah h dan $h/2$. Skim HW dan GMR menggunakan matriks transformasi T untuk meningkatkan kecekapan kaedah dan juga dibandingkan dengan skim Hairer dan Wanner yang diubah (MHW) tanpa menggunakan matriks transformasi T. Penemuan menunjukkan bahawa kaedah G2 menggunakan skim MHW memberikan pelaksanaan yang cekap dalam menyelesaikan masalah Kaps, Oreganator dan HIRES sedangkan untuk kaedah G3, ia berkesan dalam menyelesaikan masalah Kaps, Brusselator, Oreganator, Van der Pol dan HIRES. Dari segi anggaran ralat, kaedah G2 memberikan anggaran ralat terbaik untuk masalah Brusselator, Oreganator, Van der Pol dan HIRES, sementara untuk kaedah G3 ia berkesan dalam menyelesaikan masalah Kaps, Brusselator, Oreganator, Van der Pol dan HIRES, kedua-duanya dengan menggunakan skim HW. Kesimpulannya, skim MHW tanpa matriks transformasi T dapat menjadi secekap skim HW dan GMR dengan menggunakan pengaturan saiz langkah berubah-ubah dan skim MHW disarankan dalam menyelesaikan masalah kaku. Sebagai implikasi, kajian ini dapat diperluas ke berbagai jenis masalah lain seperti persamaan tunda jenis lewat dan persamaan pembezaan kabur.

CONTENTS

05-4506832 pustaka.upsi.edu.my Perpustakaan Tuanku Bainun Kampus Sultan Abdul Jalil Shah PustakaTBainun ptbupsi

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction to Numerical ODEs

Ordinary differential equations (ODEs) represent a mathematical model for many systems in various discipline of knowledge. Fatunla (2014) described that the numerical approximations are obtained at some specified points in the integration interval. The numerical method is said to be convergent if the method acquiring the properties of zero stability and consistency as mentioned by Lambert (1991). In numerical approximation, there exist a fact regarding conservation law. Shampine (2018) did mentioned that all linear conservation law are satisfying the numerical approximations of the standard methods. For nonlinear conservation law, the numerical methods basically do not produce a solution. Furthermore, the most well-known Adam-Bashforth is recognized to be a very efficient numerical method for the solution of linear and nonlinear differential equations including for the non-integer orders (Atangana & Araz, 2020). This is based on the Lagrange interpolation polynomial, however their accuracy is less than Newton interpolation's polynomial. Since many years before, numerical methods

for ODEs has been used in many discipline of research areas such as engineering, chemical, physics, biology, medical, astronomy and others due to its ability that provide the approximate solutions of nonlinear ODEs arising in those fields.

Nowadays, highly accurate solution for many kinds of complicated ODEs can be obtained by numerical approximation with the help of sophisticated software for computational mathematics. Development of computing power has revolutionized the utility of realistic mechanical and mathematical models in almost all fields as mentioned previously. Thus, a subtle numerical analysis is needed to implement these mathematical model that represents the real life problems such as given by Toufik and Atangana (2017), Owolabi (2019) and Araz (2020). Numerical method is said to be more advantages than analytical method because of the time consuming by the analytical method is much longer than numerical method when it comes to complex problems. The numerical methods are used when there is no solution for analytical methods. Even though the solution of analytical method is exact, however the analytical solution is sometimes unknown and in this case the numerical approach is required.

1.1.1 Ordinary Differential Equations

An equations that contained a derivative of one or more unknown functions (or dependent variable) with respect to one or more independent variable is called differential equations (DEs). DEs can be used to solve many system in real life problems including chemical, physical and biological processes. ODEs are parts of DEs that

consists only ordinary derivatives of one or more unknown functions with respect to only one independent variable.

First order ODEs can be written in the following form

$$y' = f(x, y), \quad y(x_0) = y_0, \quad f : [x_0, x_n] \times \mathbb{R}^n \to \mathbb{R}^n. \tag{1.1}$$

f is *autonomous* if it is a function of only y. But it is called *non autonomous* if f is explicitly depends on x. In equation (1.1), x is time variable or known as the independent variable and y is called the dependent variable, x_0 is the initial time and y_0 is the initial value. Function f is used to identify the unknown function y satisfying the ODEs.

For some equations that arising in physical modelling, Butcher (2016) mentioned that some of it are naturally expressed in one form or the other, but the emphasizing is always appropriate to write a non-autonomous equation in an corresponding autonomous form. There exists the coefficients $\mathbb{R}^n$ where it is referring to a set of real number while the coefficient N represents a set of positive integers. Equation (1.1) is known as the initial value problems (IVPs) if the value of x_0 and y_0 are given.

ODEs also can be solved analytically. However, analytical approach are difficult to solve stiff ODEs problem. This is causing by the most rational stiff systems that do not have analytical solutions, so the numerical methods is required to solve this kind of ODEs problem. A stiff ODEs is one of the fundamental of the solution that decays much

faster than the others (Lapidus & Schiesser, 1976). This behavior is sometimes troublesome even though it can be solved by numerical methods, because these systems are characterized by very high stability, which might turn into very high instability when approximated by standard numerical methods (Butcher, 2016). To overcome the instability problem, a few researcher in the past decade come out with an idea in developing many new sophisticated methods. Bjurel, Dahlquist, Lindberg, Linde, and Oden (1970) and Willoughby (1974) are the main literature survey that contributed to the finding of this methods. These methods consist of a wide variety of both explicit and implicit methods. Therefore, it is possible to perform the approximation of a solution when the exact solution of the ODEs problem is unknown.

There are three types of numerical methods that are popular among mathematicians in the solution of ODEs. These are Runge-Kutta (RK) methods, Linear Multistep methods and General Linear methods. Butcher (2016), mentioned the fact that Runge-Kutta methods only involve one step method. Fatunla (2014) give a brief explanations regarding one-step method where the consistency of this method ensures that the scheme is at least of order one. One of a simple RK method is the explicit Euler method. The explicit and implicit RK methods are able to produce a good approximate solution for certain problems depends on the nature of equations. The explicit and implicit RK methods are differ in term of the equations, coefficient and steps. Although explicit methods are easy to implement if compared to the implicit methods, the methods need more time to obtain the approximate solution (Cash, 1975). The implementation is not significant when the time taken by explicit methods are more than double the time consumed by the implicit methods. The difference of processing time occurs because of the internal stage equations of the explicit methods depends on

each other. The second stage equation need the value of the first stage equation and so on. On the other hand, for implicit methods every internal stage equations are independent which contribute to the shorter processing time. Besides, the explicit RK (ERK) is less stability compare to implicit RK (IRK) (Shampine, 1984). For these reason, this research is focusing on only IRK methods. A detailed introduction regarding RK methods is given in Subsection 1.1.2.

In application of mathematical modelling, there exists a special parameter that is called stiffness ratio and can be found in the ODEs system. A stiff equation is defined as a differential equation when the solution for solving the equations is numerically unstable for certain numerical methods, unless the appropriate stepsize selection is chosen to be extremely small (Liu, Zhang, & Zhang, 2019). Hairer and Wanner (1996) give few examples of stiff equations where it consists of a differential equations in chemical reactions, automatic control, electronic networks and biology. In obtaining a satisfactory results, it is not recommended to use a very small stepsize because this will lead to longer computational complication and is unfavorable to numerical approximation which can increase the round-off error. In the meantime, this will affect the accuracy of the simulation and the numerical results for stiff problems is not efficient, thus it is required to use a method with better stability to solve it (Liu et al., 2019). A problem is also called stiff by the fact that when the numerical solution of slow smooth movements is considerably perturbed by nearby rapid solutions (Hairer & Wanner, 1999). Simply said, a system is stiff when it involves different components that changing rapidly and slowly together.

To understand stiffness, consider the Prothero Robinson (PR) problem which is given in equation (1.2).

$$y' = \lambda\left(y - g\left(x\right)\right) + g'\left(x\right), \quad y\left(0\right) = g\left(0\right), \tag{1.2}$$

where $g\left(x\right) = \sin\left(x\right)$ with exact solution $y\left(x\right) = g\left(x\right)$ and λ is stiffness parameter. When λ become large negative number such -10000, PR problem is considered as a stiff problem resulting in using a much smaller stepsize to achieve convergence solution and in order to achieve stability as described by Gorgey (2012) and Butcher (2016). This implies high computational cost and so the search for methods with extended regions of stability is motivated (Dormand, 2018). A detailed explanation on stiff problems is given in Section 1.2.

When the numerical methods is applied throughout the investigation, there must be some errors that might spoil the solution, in other word it might cause less efficient and less accurate solution. Generally, common error is divided into two type namely local and global errors. Local error is a type of error that is produced by numerical method in an individual step where the value at the beginning of that step is assumed to be exact. When the local errors after *n* steps is accumulated, then this is where the global error will produced. In other words, the global error is another type of error that accumulated from the local error after *n* steps. Butcher (2016) had mentioned the fact that the accumulation is not necessarily causing by the summation of local errors at each *n* steps, on the other hand it is causing by the sum of the bounds on the local errors. Dormand (2018) described that the best process for global error computation is based on a parallel solution of a related system of differential equations. These are constructed

to have a solution satisfied by the actual global error of the main system of equations. Local errors, l_n can be defined by

$$l_n = u_n(x_n) - y_n, \tag{1.3}$$

where, u_n is the solution curve and y_n is called exact solution curve. The global error, ε_n is written in the following form

$$\varepsilon_n = y(x_n) - y_n, \tag{1.4}$$

where $y(x_n)$ is the solution curve at n steps. Equation (1.4) then can be written as

$$\varepsilon_n = y(x_n) - u_n(x_n) + l_n, \tag{1.5}$$

where ε_n is the actual error after n steps. Thus, there are two types of global error, one is related to the local errors at the present step and the other is related to the local errors at the previous steps.

Other than these errors, there is another error known as round-off error as mentioned before. These errors can destroy the numerical solutions if it is significant in numerical approximation. Detailed about round-off errors will be discussed on the next chapter. In the next section, a detail explanations regarding RK methods is discussed.

1.1.2 Introduction to Runge-Kutta Methods

Runge-Kutta (RK) methods have been popular among mathematicians for many year and are developed specialize in finding an approximate solution for ODEs. This methods are originally developed by Runge towards the end of the nineteenth century

and generalized by Kutta in the early twentieth century. These methods are easy to implement compared to Taylor polynomial scheme which requires the formation and evaluation of higher derivatives as described by Dormand (2018). Basically, an *s*-stage RK methods for the step $(x_{n-1}, y_{n-1}) \to (x_n, y_n)$ with stepsize h can be defined as

$$Y_i = y_{n-1} + h\sum_{j=1}^{s} a_{ij} f\left(x_{n-1} + c_j h, Y_j\right), \tag{1.6}$$

$$y_n = y_{n-1} + h\sum_{j=1}^{s} b_j f\left(x_{n-1} + c_j h, Y_j\right), \tag{1.7}$$

where $i, j = 1, 2, ..., s,$ s is the number of stage. Y_i represents the internal stage values for the i^{th} stage and y_n represents the update of y at the n^{th} step. The coefficient a is used to find the internal stages by using the linear combinations of the stage derivatives. The vector b represents the quadrature weights which indicates how the approximation to the solution depends on the derivatives of the internal stages. The coefficient c is the vector of abscissas which indicates the positions within the step of the stage values. A detailed explanation can be found in Butcher (2016).

It is called a one-step method and can be demonstrated schematically in the following diagram:

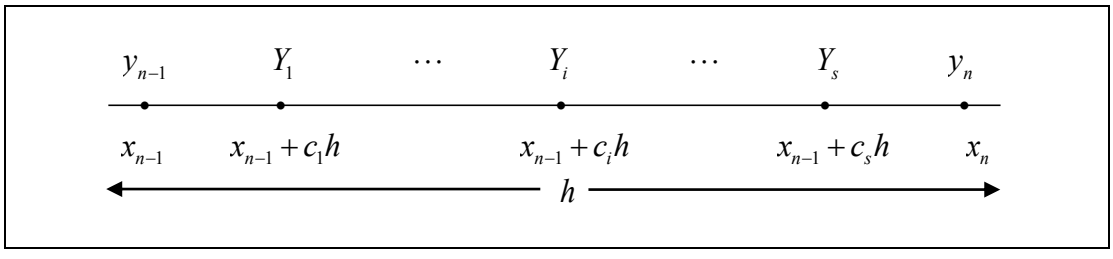


Figure 1.1. Diagram of One-Step Runge-Kutta Methods.

The coefficient a and c must hold the row-sum condition as given in the Table 1.1. The coefficients in the general equation (1.6) and (1.7) shall be represented by a partitioned tableau known as the Butcher tableau (Butcher, 2016) of the form

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

where A is a matrix that consist of the a values of RK methods and b^T is referring to vector b which is the quadrature weights.

RK methods are divided into two components, namely the implicit Runge-Kutta (IRK) and the explicit Runge-Kutta (ERK) methods. The ERK methods form a triangular matrix A of the coefficient a. One example of famous ERK methods is the classical RK method of order-4 (RK4). In IRK methods, the coefficient matrix A is not triangular that make a big difference with ERK methods. There are several types of implicit methods and it can be divided into few categories, the first one is known as fully-implicit if matrix A is not lower triangular and it is called semi-implicit if A is lower triangular with at least one non-zero diagonal element. Besides, the IRK methods is also known as diagonally-implicit if A is lower triangular with all the diagonal elements are equal and non-zero or simply called as diagonally implicit Runge-Kutta (DIRK) and singly implicit if A is matrix with a single non-zero eigenvalue singly implicit Runge-Kutta (SIRK). Table 1.1 describe these properties.

Table 1.1

Butcher Tableau of Explicit RK and Implicit RK Methods

Explicit RK	Implicit RK
$\begin{array}{c\|cccc} 0 & & & & \\ c_2 & a_{21} & & & \\ c_3 & a_{31} & a_{32} & & \\ \vdots & \vdots & \ddots & & \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s-1} \\ \hline & b_1 & b_2 & \cdots & b_{s-1} \quad b_s \end{array}$,	$\begin{array}{c\|cccc} c_1 & a_{11} & a_{12} & \cdots & a_{1s} \\ c_2 & a_{21} & a_{22} & \cdots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \cdots & a_{s,s} \\ \hline & b_1 & b_2 & \cdots & b_s \end{array}$.

Explicit methods are easy to implement as the internal stages can be calculated directly without depending on later stages as described on Subsection 1.1.1. However, explicit methods cannot be used to solve stiff problems since they have poor stability behavior (refer to Section 1.2). In other word, implicit methods are suitable for solving stiff problems however they are more costly to implement. The implementation of implicit methods is discussed in Chapter 3 on Section 3.4.

Some examples of ERK methods are the Euler method, explicit trapezoidal rule, explicit midpoint rule and other higher order explicit methods. The simplest ERK methods is the Euler method which of order-1. For the IRK methods, it consists of some methods such as the implicit Euler method, implicit midpoint rule, implicit trapezoidal rule, Gauss methods, Radau methods, Lobatto methods and other higher order implicit methods (Hairer & Wanner, 1996). For this research, it is only involving 2-stage (G2) and 3-stage (G3) Gauss methods. The Butcher tableau (Butcher, 2016) for the 2-stage and 3-stage Gauss methods are given in Table 1.2.

Table 1.2

Butcher Tableau of G2 and G3 Methods

2-stage Gauss method (G2)

$$\begin{array}{c|cc} \frac{1}{2}-\frac{\sqrt{3}}{6} & \frac{1}{4} & \frac{1}{4}-\frac{\sqrt{3}}{6} \\ \frac{1}{2}+\frac{\sqrt{3}}{6} & \frac{1}{4}+\frac{\sqrt{3}}{6} & \frac{1}{4} \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array},$$

3-stage Gauss method (G3)

$$\begin{array}{c|ccc} \frac{1}{2}-\frac{\sqrt{15}}{10} & \frac{5}{36} & \frac{2}{9}-\frac{\sqrt{15}}{15} & \frac{5}{36}-\frac{\sqrt{15}}{30} \\ \frac{1}{2} & \frac{5}{36}+\frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36}-\frac{\sqrt{15}}{24} \\ \frac{1}{2}-\frac{\sqrt{15}}{10} & \frac{5}{36}-\frac{\sqrt{15}}{30} & \frac{2}{9}-\frac{\sqrt{15}}{15} & \frac{5}{36} \\ \hline & \frac{5}{18} & \frac{4}{9} & \frac{5}{18} \end{array}.$$

G2 method is of order 4 whereas G3 method is of order 6. The defining equations as referring to equation (1.6) and (1.7) for the 2-stage Gauss method are given in equation (1.8), (1.9) and (1.10) while in equation (1.11), (1.12) and (1.13), it is referring to the defining equation for the 3-stage Gauss method.

The stage equations of 2-stage Gauss method are defined by

$$\begin{aligned} Y_1 &= y_{n-1} + h\left(\frac{1}{4}\right)F_1 + h\left(\frac{1}{4}-\frac{\sqrt{3}}{6}\right)F_2, \\ Y_2 &= y_{n-1} + h\left(\frac{1}{4}+\frac{\sqrt{3}}{6}\right)F_1 + h\left(\frac{1}{4}\right)F_2. \end{aligned} \tag{1.8}$$

The internal stage derivative equations of the 2-stage Gauss method are defined by

$$\begin{aligned} F_1 &= f\left(x_{n-1} + h\left(\frac{1}{2}-\frac{\sqrt{3}}{6}\right), Y_1\right), \\ F_2 &= f\left(x_{n-1} + h\left(\frac{1}{2}+\frac{\sqrt{3}}{6}\right), Y_2\right). \end{aligned} \tag{1.9}$$

The update equation of the 2-stage Gauss method is defined by

$$y_n = y_{n-1} + h\left(\frac{1}{2}\right)F_1 + h\left(\frac{1}{2}\right)F_2. \tag{1.10}$$

The equations (1.11) are the stage equations of the 3-stage Gauss method.

$$\begin{aligned} Y_1 &= y_{n-1} + h\left(\frac{5}{36}\right)F_1 + h\left(\frac{2}{9} - \frac{\sqrt{15}}{15}\right)F_2 + h\left(\frac{5}{36} - \frac{\sqrt{15}}{30}\right)F_3, \\ Y_2 &= y_{n-1} + h\left(\frac{5}{36} + \frac{\sqrt{15}}{24}\right)F_1 + h\left(\frac{2}{9}\right)F_2 + h\left(\frac{5}{36} - \frac{\sqrt{15}}{24}\right)F_3, \\ Y_3 &= y_{n-1} + h\left(\frac{5}{36} + \frac{\sqrt{15}}{30}\right)F_1 + h\left(\frac{2}{9} + \frac{\sqrt{15}}{15}\right)F_2 + h\left(\frac{5}{36}\right)F_3. \end{aligned} \tag{1.11}$$

The internal stage derivative equations of 3-stage Gauss method are defined by

$$\begin{aligned} F_1 &= f\left(x_{n-1} + h\left(\frac{1}{2} - \frac{\sqrt{5}}{10}\right), Y_1\right), \\ F_2 &= f\left(x_{n-1} + h\left(\frac{1}{2}\right), Y_2\right), \\ F_3 &= f\left(x_{n-1} + h\left(\frac{1}{2} + \frac{\sqrt{5}}{10}\right), Y_3\right). \end{aligned} \tag{1.12}$$

The update equation of 3-stage Gauss is defined by

$$y_n = y_{n-1} + h\left(\frac{5}{18}\right)F_1 + h\left(\frac{4}{9}\right)F_2 + h\left(\frac{5}{18}\right)F_3. \tag{1.13}$$

Since this research only focuses on IRK methods, thus several IRK methods will be used in solving ODEs problems. Some ODEs problems have an equation of exact solution. There exists an error of approximation where it is referring to the difference between the approximate solution and the exact solution. Normally, the efficiency of the methods can be represented by the graph of the error versus the tolerance and computational (CPU) time. In addition, the efficiency can be improved by proper

method of implementation by various researchers. Thus, several implementations were tested in solving several ODEs problems including chemistry and physics problems.

In numerical methods, ODEs plays an important rule in solving a simple linear equation. Several analytical methods can be used to solve the equations such as separable variable, factorization, substitution and other methods. However, analytical solution for nonlinear equations are always hard to solve. On the other hand, several type of numerical methods mentioned earlier is also quite important since it can solve approximate solution of the nonlinear equations whenever the exact solution is unknown. In obtaining a good result of numerical solutions, the combination of a good implementation and very small error will lead to the closest exact solution. The IRK and ERK methods are able to produce a good approximate solution for certain problems depends on the nature of equations.

1.2 Problem Statement

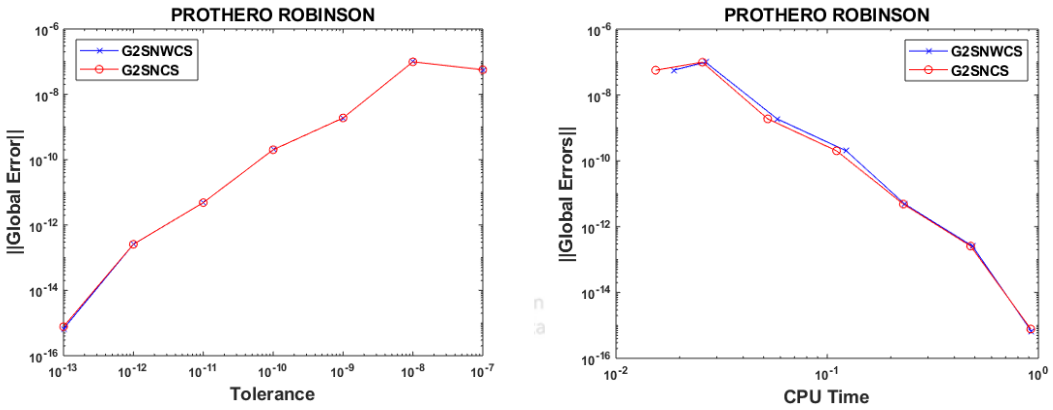
As ERK method is very easy to implement, so the internal stages can be calculated directly without depending on the later stages as mentioned on the previous section. Besides, this method also incurs cheap implementation cost. Even though ERK methods having this advantages, however the stability of the ERK methods is classified as not *A*-stable (Iserles, 2009). Thus, the ERK methods cannot be used to solve stiff problems compared to the IRK methods as they have poor stability (Sanderse & Koren, 2012). IRK methods not only possess strongest stability properties, thus it also satisfy the properties of *A*-contractivity (algebraic stability) even though it is difficult to implement

(Hairer & Wanner, 1981). Therefore they are suitable in solving stiff problems. The IRK methods not only important in solving stiff problems, furthermore it is beneficial to differential algebraic equations. Nevertheless, the IRK methods are expensive and difficult to implement due to the nonlinear equations involved when finding the internal stage derivatives Y_i and need to be replaced by an iterative computation which is known as Newton-Raphson iteration. Even though it is difficult to implement, the IRK methods gives a fewer stages for the same order and better stability if compared to the ERK methods. Due to this better stability, the implicit methods are widely used in the applications of physics, engineering, chemistry and medical problems.

There are two ways to implement Newton-Raphson iterations, which are full Newton and simplified Newton. Full Newton iteration is preferred for nonstiff problems as investigated by Muhammad and Gorgey (2018). However, to solve certain real-life stiff problems such as Van der Pol, Brusselator and Oregonator problems, small stepsize such as 0.001 is required if using constant stepsize setting. This not only takes longer computational time, round-off errors also can accumulate and destroy the solution. Therefore using constant stepsize is no longer appropriate. For this research, variable stepsize setting will be used to investigate the performances of three different implementation strategies.

At the beginning of the code, a technique known as compensated summation is introduced to make sure the round-off errors will not destroy the numerical solutions. The purpose of compensated summation is to minimize the effect of round-off errors and it is applied together with simplified Newton iteration. However, based on the numerical results for Prothero-Robinson test problem with $q = -10000$, there is no

effect in terms of accuracy on G2 method with simplified Newton and compensated summation (G2SNCS) (Refer Figure 2). The numerical results showed that G2 using simplified Newton without compensated summation (G2SNWCS) has similar results with G2SNCS. Therefore, no compensated summation is required to investigate the performance of G2 and G3 methods using implementation schemes by Hairer and Wanner (1999), González-Pinto, González-Concepción, and Montijano (1994) and González-Pinto, Montijano, and Randez (1995).



(a) (b)

Figure 1.2. The effect of round-off error on (a) Tolerance and (b) CPU Time between G2SNCS and G2SNWCS for Prothero Robinson problem using variable stepsize setting.

Although in Muhammad (2018) thesis, he studied of the implementation strategies by Hairer and Wanner (1999), Cooper and Butcher (1983) and González-Pinto et al. (1994, 1995) schemes, however he studied only for constant stepsize setting. In difficult nonlinear ODEs problems, constant stepsize setting will require more computational time to solve depends on the stiffness ratio. For example, consider the Robertson problem (Robertson, 1966) or it was known as ROBER problem. Hairer and Wanner (1996) are the one who gave the name ROBER. A detailed explanation regarding ROBER problem is given in Subsection 5.1.1.

Variable stepsize setting is very important to be implemented for the numerical approximations of RK Gauss methods. When the variable stepsize setting is used, the stepsize changing policy will automatically adjusted especially when the difficult nonlinear problems involved. Hence, in this research, a variable stepsize setting is investigated in detailed using the implementation schemes from Hairer and Wanner (1999) and González-Pinto et al. (1994, 1995).

1.3 Research Objectives

This thesis investigated two numerical methods for solving the ordinary differential equations (ODEs) such as 2-stage Gauss (G2) and 3-stage Gauss (G3) methods. The main objectives of this research are:

1. To study the implementation ideas for implicit Runge-Kutta methods recommended by Hairer and Wanner (1999) and González-Pinto et al. (1994, 1995) using variable stepsize setting.
2. To study the best error estimation for the variable stepsize setting in solving stiff problems.
3. To investigate the most efficient implementation strategy for Gauss methods in solving stiff problems using variable stepsize setting.

1.4 Research Questions

In this research, several questions of interest are wished to attempt. Some of these are:

1. How does the implementation schemes by Hairer and Wanner (1999) and González-Pinto et al. (1994, 1995) are implemented for implicit Runge-Kutta methods using variable stepsize setting?
2. Which scheme is giving the best error estimation for the variable stepsize setting in solving stiff problems?
3. Which implementation strategy is the most efficient for the Gauss methods in solving stiff problems by using variable stepsize setting?

1.5 Significant of Research

By the end of the research, the researchers are wish to obtain:

1. The computational cost for the implicit methods can be reduced by using the most efficient implementation strategy suggested.
2. The most efficient implementation strategy can be identified for the Gauss methods in solving stiff problems.
3. Researchers have broad knowledge regarding the idea of the implementation techniques for implicit Runge-Kutta methods.
4. Researchers can start using implicit Runge-Kutta methods which is proven to satisfy the efficiency properties and capable in solving real life problems especially for Robertson and Van der Pol problems.

5. The round-off errors can be reduced even without using compensated summation technique for the variable stepsize setting.

1.6 Scope of Study

This research focuses on the IRK methods. Only 2-stage (G2) and 3-stage (G3) Gauss methods that emphasized in this research. Preliminary study of this research is about understanding the ideas of implementation for IRK methods recommended by various researchers including the standard/common implementation methods. The first stage in this research is to perform test problems using Prothero Robinson (1974) problem to investigate the importance of using simplified Newton with compensated summation for variable stepsize setting. The implementation strategies by Hairer and Wanner (1999) and González-Pinto et al. (1994, 1995) are selected to solve real life problems such as Robertson, Van der Pol, Kaps and Oreganator problems using variable stepsize setting. All of these problems are given by Hairer and Wanner (1996). A detailed explanation regarding all of these problems can be found in Section 5.1. The construction of the G2 and G3 methods as well as the numerical experiments for all the problems involved are performed using MATLAB R2019a numerical software.

1.7 Thesis Outlines

There are 6 chapters in this thesis.

Chapter 1 is about the Introduction. This chapter divided into five main parts. The first part discussed about the background of this study which is the introduction to numerical ODEs including some basic knowledge regarding RK methods, problem statement, objectives, significant of research and scope of this research.

Chapter 2 discussed about the literature review. A brief explanation about history of implementation of RK methods, efficiency of Gauss methods and the implementation ideas by other researchers is discussed in this chapter.

In Chapter 3, the discussion regarding research methodology that consists of research design, the construction of 2-stage (G2) and 3-stage (G3) Gauss methods and the implementation of implicit Runge-Kutta methods based on the simplified Newton iteration.

Chapter 4 gives the construction of implementation strategies by Hairer and Wanner (1999) and González-Pinto et al. (1994, 1995) in solving some real life problems such as Robertson and Brusselator problems as given by Hairer and Wanner (1996). The implementation strategies will be used in solving real life problems in Chapter 5.

Chapter 5 gives the numerical experiments of this research. This chapter gives all the numerical results for real-life problems. The numerical results are given by the tolerance and computational (CPU) time diagrams. The tolerance diagram indicates the accuracy of the methods while the CPU time diagram is to measure the efficiency for certain stepsize. A detailed description of variable stepsize setting and explanation of error estimation also will be discussed in this chapter.

Lastly, Chapter 6 summarizes the numerical results and presents some conclusions and also the suggestions for future work.

CHAPTER 2

LITERATURE REVIEW

2.1 History of Runge-Kutta Methods

In numerical analysis, there exists a method that is called the Runge-Kutta (RK) methods. RK methods consists of implicit and explicit iterative methods, which includes the popular method which is known as Euler method. The Euler method is being used in temporal discretization for the approximate solutions of ordinary differential equations (ODEs) (DeVries & Hasbun, 2011). Carl Runge and Martin Wilhelm Kutta are the well-known German mathematicians that developed this methods around 1900 (Butcher, 1996).

Carl Runge was completing his famous paper one hundred years ago and this work was published in 1895. He extended the approximation of Euler method to a more elaborate scheme which useful in producing greater accuracy. A detailed explanations was mentioned by Butcher (1996). The basic idea of Euler method was to generate the solution of an initial value problems (IVPs) in more precise steps. At the beginning of

the step, the rate of change of the solution that evaluated from the derivative formula is treated as constant in each step. Rechenberg (2001) described the differential equations that occur in the atomic spectra research had led Carl Runge developed the numerical method to solve the equations. Hitchens (2015) mentioned the fact that Martin Kutta is the one who contributed to the numerical method for differential equation in the aerodynamics.

Development of RK methods are done since past decades by many researchers. The first published article is in 1895. They searched for greater order of explicit Runge-Kutta (ERK) throughout the years. Modern development of RK processes has occurred since 1960, mainly as a direct result of the advances due to Butcher in the development and simplification of RK error coefficients (Dormand, 2018). Butcher (1996) gave the chronology of the developed methods by the corresponding author based on the order hierarchy as given in the Table 2.1.

Before 1970, Kuntzmann (1961) and Butcher (1964) suggested that the IRK methods are based on Gauss quadrature formulae. To construct a good method in solving stiff problems, the criteria that need to be considered are high accuracy, good stability and low implementation cost. The Gauss methods are chosen because they are highly stable as well as high accuracy and possess higher order than explicit and other implicit methods where the order p is equal to $2s$, s is referring to the number of stages of the IRK methods. The higher the order of Legendre Polynomials, the more accurate the numerical approximation is (Cerrolaza, Shefelbine, & Garzón-Alvarado, 2018). Even though the Gauss methods possess a good stability and high accuracy properties, therefore they are expensive to implement because the methods have

different and complex eigenvalues which makes the implementation difficult. Nevertheless, the Gauss methods are categorized as a symmetric method. Gorgey and Muhammad (2017) mentioned further regarding this behaviour. It is an advantages for Gauss methods since it provide capability to give more accurate solution. This kind of property is an extra advantage that cannot be found in ERK methods.

Table 2.1

List of the Explicit Methods Based on the Order and Author (Butcher, 1996)

Order (p)	Stages (s)	Author	Year	Reference
2	2	Runge	1985	(Runge, 1895)
3	3	Heun	1900	(Huen, 1900)
4	4	Kutta	1901	(Kutta, 1901)
5	6	Kutta	1901	(Kutta, 1901)
5	6	Nystrom	1925	(Butcher, 1996)
6	8	Huta	1956	(Huta, 1965)
6	7	Butcher	1964	(Butcher, 1964b)
7	9	Butcher	1987	(Butcher, 1987)
8	11	Curtis	1970	(Curtis, 1970)
8	11	Cooper and Verner	1972	(Cooper and Verner, 1972)
10	18	Curtis	1975	(Curtis, 1975)
10	17	Hairer	1978	(Hairer, 1978)

The IRK methods consist of several types of components named with semi-implicit RK (SIRK) methods, semi-explicit RK (SERK) methods, diagonally-implicit RK (DIRK) methods and singly-diagonally-implicit RK (SDIRK) methods (Butcher,

1996). The IRK methods for which $a_{ij} = 0$, for $j > i$, are called semi-implicit formulae, the class of which the one-stage RK2 are members. These can be made it as *A*-stable. For practical purposes they are simpler to implement than the fully implicit formulae, since each stage consists of the determination of only a single f. When solving stiff problems, it was found out that the Gauss and Lobatto IIIA methods suffer from the order reduction phenomenon. This is one of the disadvantage of one step methods that can be found when solving stiff problems. For example, the numerical order of convergences for fully IRK methods such as Gauss-Legendre methods suffers from order reduction where their order decreases from $2s$ to s, coefficient s denoted the number of internal stages (Rang, 2016). To make sure the order reduction is reduced, a researcher came out with the study of stability and convergence and a new technique known as symmetrization has been introduced by Gorgey (2012).

The RK methods is called *A*-stable if the stability function $R(z) = 1 + zb^T (I - zA)^{-1} e$ where $e = (1,\ldots,1)^T \in \mathbb{R}^s$, $b = (b_1,\ldots b_s)^T$ and $A = \left(a_{ij}\right)_{i,j=1}^{s}$ satisfy the properties $|R(z)| \le 1$ for all $z \in \mathbb{C}^-$. Otherwise, if $R(\infty) < 1$, the RK methods is called strongly *A*-stable and *L*-stable if $R(\infty) = 0$ (Ehle, 1973). Rang (2016) did mentioned about *A*-stability property implies that RK method is dissipative for Dahlquist's problem. It is guarantee in getting stable numerical solution if the method satisfy the *A*-stability. RK method is called *B*-stable if they are algebraically stable and able to solve the nonlinear problems. Some of it are Gauss-Legendre, Radau IA, Radau IIA and Lobatto IIIC methods. Furthermore, other advantages of Gauss-Legendre

methods is also the behaviour that satisfy the simplifying conditions $B(1),\ldots,B(2p)$ and $C(1),\ldots,C(s)$.

Since many decades, many researchers studied this method for solving ODEs problems such as Chan (1990), Cong (1994), Calvo, Franco, Montijano and Randez (2009). Zhu, Hu, Tang, and Zang (2016) showed in their article that second order symmetric RK methods perform better than non-symmetric RK method in long-term integration and almost energy conservation. Several years before, Chan and Gorgey (2013) reported that symmetric RK methods with symmetrization techniques give more accuracy and efficiency for solving stiff linear problem.

2.2 Efficiency of Gauss methods

In numerical analysis, it is very important to choose a method that satisfy the good stability properties and having higher order of convergence rate. Since RK methods complies with these properties, thus a method such as Gauss methods are particularly being chosen because of their advantages that suitable in solving stiff systems. This is also due to sufficiently high stage and classical orders. The computational cost of these methods is relatively high because they are fully implicit and require at each step the evaluation of $ms \times ms$ system of equation (1.6). The coefficient m is refers to the dimension of the system and coefficient s is refers to the stage of derivatives. Even though the computational cost is relatively high, however the methods provide better solution of same accuracy as the order of the IRK methods. The

study showed that the methods numerically integrate various sorts of ODEs such as non-stiff and stiff problems, Hamiltonian systems and invertible equations. González-Pinto et al. (1994) investigated an experiment regarding linear stability of IRK methods. In their research, they proposed a method by Cooper and Butcher (1983) in determining the most efficient method in solving IRK methods. They concluded that the implementation by using Gauss method performs much better than DIRK method even though both of the methods are categorized as *A*-stable and have the same order 4. This is such a big difference that can be found during the investigation due mainly to the fact that the both methods having the same cost per step required on one side. Even though the Gauss methods having the handicap of solving the implicit system $Y = e \otimes y_n + h(A \otimes I\ )F(Y)$ (similarly, refer to equation 1.6) during the experiments, however their relatively high stages and good stability properties make them not only competitive but highly recommended to other methods like DIRK methods for the solution of nonlinear stiff problems when implemented using special iterative schemes.

Varah (1979) described the comparison of methods used in producing an efficient implementation of IRK methods. Since we concerned that Butcher methods have order s or $s+1$, while the Gauss methods have order $2s$ or $2s-1$, it can be seen that for the method that has the properties of same order method, Gauss methods require less work per step compared to Butcher methods. Moreover, it also turn out that the Butcher methods having an error constants larger than Gauss methods especially for the *A*-stable methods. This leads the methods to produce more steps for the same accuracy. In addition, this inefficient behaviour make it difficult to compare these methods with stiff multistep methods like those of Gear (1980). In Gauss-Legendre method, the operations involved are complex because of the complexities in eigenvalues. If it were

programmed directly in a language with complex type declarations, it may be requires much less work compared to complex multiplication that involving four real ones. For example, it is more practical to use a factor of two in Fortran on IBM 370 machines. González-Pinto et al. (1994) also mentioned that Gauss methods having of advantages of high order of convergence in comparison with the number of stages and good stability properties that make it suitable for solving stiff systems. Due to this, Gauss methods requires relatively high computational cost since they are fully implicit.

A research by Agam and Yahaya (2014), they have developed a more efficient and stable method of new 3-stage IRK methods using collocation method at pertubed Gaussian points. Basically, the method is different from the existing 3-stage Gauss in term of the equation of the internal stage value Y_i and the coefficient of b in the equation of the update solution y_{n-1}. The internal stage derivative $F(Y_i)$ in the equation Y_i of the existing 3-stage Gauss method was replaced with a new one that was formulated using collocation method. Besides, the coefficient of b in the equation (1.6) from the general form of RK was replaced with new coefficient b that was computed using new coefficient c. It is proved that this new method produced an efficient results than the existing 3-stage Gauss method in solving one dimensional of a linear and a nonlinear problems of first order ODEs.

Kulikov (2015) constructed nested Gauss and Lobatto methods for solving stiff differential problems using variable stepsize. The methods preserved the properties of IRK methods, such as *A*-stability, symmetry and symplecticity. Symplectic RK methods was systematically developed by Sanz-Serna (1988). Their idea is based on

algorithm of algebraic stability introduced that involving stiff systems studied by Burrage and Butcher (1979). Sanz-Serna (2016) did mentioned that these methods also has a wide range of applications not only in Hamiltonian problems but also beneficial in other applications that required the use of adjoint systems and optimal control problems. Gorgey and Mat (2018) have mentioned about the combination of two methods that can be shown to be symmetric and symplectic which is known as partitioned RK methods (PRK) that also advantageous in solving Hamiltonian system that is separable. A further explanations about PRK can be found in Abia and Sanz-Serna (1993) and Sun (2000).

Generally, Gauss method is also known as a collocation method that based on the Gaussian quadrature formulas. Since the algebraic accuracy of Gaussian quadrature formulas for point s is $2s-1$ while its truncation error is $l_n = y(x_{n-1}) - y_{n+1} = O\left(h^{2s+1}\right)$, hence it is satisfies the order conditions which is $2s$. The order of numerical methods is the crucial indicator in measuring the accuracy of the method. Basically, the higher accuracy is affected by the relatively higher order of the numerical method. For SDIRK and SIRK methods, the maximum attainable order for both methods are $s+1$. Here, it is clear that the Gauss method has higher order and higher accuracy, which is the main objective why this method is chosen as numerical approximation. However, it is doubtful when this method is applied to a large system simulation because it is fully implicit with a larger computational cost. This caused the computational cost to grow increasingly expensive for higher stages methods and higher dimensional system. Because of this reasons, it is necessarily to reduce its computational cost by using new method proposed by Liu et al. (2019). This method is known as banded IRK (BIRK) method and will be discussed further on the Section 2.3.

2.3 Implementation Ideas by Other Researchers

Butcher (1997) introduced a classic transformed method which is known as singly implicit Runge-Kutta (SIRK) method where the method has only one real s fold eigenvalue. Nevertheless, not all SIRK methods are categorized as A-stable that makes the maximum attainable order reduced. Thus, Liu et al. (2019) proposed a new method which is known as banded implicit Runge-Kutta (BIRK) method. The aim of this method is to reduce the computational cost by making a changes to the Jacobian matrix from a full coefficient matrix to a banded matrix while maintaining the high accuracy and good stability properties. The purpose of reducing the computational cost for IRK methods produced a singly diagonally implicit Runge-Kutta (SDIRK) method, where the coefficient matrix A is lower triangular with same diagonal elements λ rather than using a full coefficient matrix (Ababneh & Ahmad, 2009). Even though the SDIRK method has a straightforward computational advantages over the fully IRK method, however the method has some inconvenience components that makes their stability and accuracy affected by the simplification of the coefficient matrix A. The main advantages of BIRK methods is that the method reduced the computational complexity of the LU factorization and back substitution to the Newton update iteration which this behaviour did not obtained by Gauss-IRK method. In addition to that, the BIRK method maintained a good accuracy compared to the Gauss-IRK method. Hence, it can concluded that the BIRK method is easier to implement programmatically compared to the SDIRK and SIRK methods. This method of order $2s$ is also categorized as A-stable.

Berghe and Daele (2011) presented the development of symmetric and symplectic modified exponentially-fitted Runge-Kutta (EFRK) method. They derived

the EFRK methods as a 4-stage Gauss of eighth-order. The suitable frequency is needed for determining the order and accuracy. The numerical results shows that the solutions from classical and the 4-stage EFRK method are not largely different compared with the solutions of 2-stage and 3-stage methods. The construction procedure for the development of other EFRK method of different order also being recommended. In addition, the EFRK method preserved the symplectic properties. They also mentioned that the method gives same result as other exponentially fitted methods such as multistep methods. Furthermore, the method provide more accuracy than the classical method, hence their studies considered as a great achievement.

In Skvortsov and Kozlov (2014), an efficient implementation has been developed for three types of diagonally implicit Runge-Kutta (DIRK) methods. Four types of implementation scheme involved namely, trivial, modified trivial, standard and economical schemes. The trivial is about the use of trivial prediction which is the computed values at the initial point of approximation step that being used as initial values for iterations. For modified trivial, it is being used to modify stage equation of the corresponding methods. Different approach is applied to the standard implementation where the initial values for the iteration are provided as a linear combination of the previous stage values. For the economical scheme, it is about a prediction for estimating the initial values of internal stage derivative. Based on the numerical test problem, it is showing a result that economical scheme secures an acceptable convergence by a single calculation, but the standard scheme requires two computations. By right, the economical scheme saves one calculation at each implicit stage compared to the standard scheme. The schemes also has been tested to solve real life problems such as Van der Pol, Oregonator and HIRES by using three different

values of tolerance. The tested is to capture the size of error, number of internal stages derivative function and Jacobian.

Nazari, Mohammadian, Charron, and Zadra (2014) performed optimization on 3-stage DIRK methods in finding a scheme that can retains a good order accuracy. The numerical results shows that the new scheme giving highly accurate solution than their previous scheme for the problem that involving larger stepsize. Other than that, the new scheme gives better accuracy for low spatial resolutions with the same stepsize. The scheme that was developed is also categorized as *A*-stable which makes it suitable option for solving stiff problems. Even though the scheme gives better efficiency, the computation for diffusion coefficient is not really cheap. However the proposed scheme performs well with large stepsize for the problem involved.

An updated technique known as generalized summation-by-part (GSBP) was constructed by Boom and Zingg (2015) in solving IRK methods. The methods that was constructed are based on Lobatto IIIC and Radau IA/IIA discontinuos collocation, Gauss quadrature points and some algebraically stable and DIRK method. The numerical simulation shows that the GSBP methods are more competent compared to the classical summation-by-parts (SBP) methods. The comparison has been investigated between the Gauss and Radau IA methods when applying the GSBP methods. The numerical results obtained are both of the methods retains the same properties, however the Gauss method is more efficient in terms of stage error. For the non-SBP method with the same number of stages, it gives more efficient result however it is categorized as not L-stable. The study is extended to fifth-order explicit singly

DIRK (ESDIRK) method that beneficial in constructing higher order GSBP methods which are diagonally implicit (Boom & Zingg, 2015).

Zhang, Sandu, and Tranquilli (2015) discovered a new technique specialize to recover the order of corresponding IRK methods for their research. They introduced a refinement procedure to correct stage values that was motivated from the simplified Newton method. The procedure successfully recover the order of the methods in solving non-stiff, midly stiff and stiff problems. In some cases, the order is recovered by only small number of refinement iterations for non-stiff and midly stiff problems, however for stiff problems large number of refinement iterations is needed since the increasing stiffness deteriorates the convergence. Before refinement procedure been introduced, the approximate matric factorization to high order linearly IRK methods is unstable for stiff problems. After several test problems has been done, they concluded that the refinement procedure improves the efficiency and it validates the accuracy and stability based on theoretical findings.

Development of a new Runge-Kutta method has been developed by Ramos (2019). This development was known as a two-step hybrid block method that was specialized for numerically solving first-order IVPs. This method largely using the well-known schemes of RK and multistep methods. A new formula of this method is obtained by choosing two intermediate points of the interpolation derivation and collocation at different points through the optimization of the local truncation errors with continuous approximation. This method are practicing self-starting method where it does not provide any starting values when using other approaches. Even though this method might requires more computational cost, however the number of occurrences

of the source term f is reduced that resulting in the most competent formulation. For linear problems, it might be seen the both formulation are essentially the same, but when the problem where f is difficult, this reformulation will result in an outstanding saving on computational cost. The existing block methods are using this strategy in getting the best behaviour of the block formulation. Moreover, this method consists of a good characteristics which satisfy the convergence order and A-stability property that make it appropriate for solving stiff problems.

In Kennedy and Carpenter (2019), a general purpose of DIRK methods has been performed to first order ODEs that involving five types of explicit singly diagonally implicit Runge-Kutta (ESDIRK) and their implicit-explicit (IMEX) methods. The purpose of the methods is focusing on achieving a 2-stage order, stiff-accuracy, L-stability, internal L-stability, an embedded method with good quality, algebraic stability of matrix eigenvalues with a small magnitudes and a small values of a_{ii}. All of the mentioned characteristics are persistently important in maximizing the scheme efficiency. An embedded method is being used to facilitate the stepsize control through error estimation. As the stage order affected the order reduction, focusing on 2-stage order helps in determining the accuracy. The order reduction is depends on the problem that being tested. It is observed that the methods produced moderate order reduction for the Kaps problem while the Van de Pol problem is having a severe order reduction. For the problems that categorized as excessively stiff behaviour, the utility of the fifth and sixth order of ESDIRK methods that being used in this research is lower compared to those lower order methods.

In the same year of 2019, a researcher that was known as Zhang had discovered a new RK methods using unstructured numerical search. The emphasizing of the new methods is to exhibit a minimum number of stages in constructing RK methods in order to maximize their order. Nevertheless, higher order RK methods are challenging to be implemented since their parameter must comply with an exponentially large system of polynomial equations. In his research, he studied about the strategy in decreasing the number of stages for higher order method. The research that has been investigated proved that the 10^{th}-order RK method only required 16 stages. He was the first one that managed to break a 40 years standing record that proved giving a less number of stages in achieving the accuracy. The mechanism of techniques and theorems that empowering the discovery of this method is discussed further in his research (Zhang, 2019).

2.4 Variable Stepsize Setting

Variable stepsize setting is crucial for the solution of nonlinear equations such as stochastic wave equations of Schrödinger because they are typically nonlinear which the error might propagate very rapidly and deteriorate the solutions. This was given by Wilkie and Çetinbaş (2005). In their research, they showed that by implementing the variable stepsize, the explicit 9th order RK method (with an embedded 8th order method) of ODEs yields an order 4.5 method for stochastic differential equations (SDEs) which is part of the stochastic differential systems. When the lower order methods is implemented with constant stepsize, the solutions produced is highly inaccurate that might destroy the solutions, hence they become relatively inefficient

because of their lower order. This clearly proved that the variable stepsize setting is suitable in solving higher order methods.

In difficult mathematical applications, it is unrealistic to use a constant stepsize setting. This is due to the research that was investigated by Chan and Razali (2014) regarding the two-step symmetrization in a constant stepsize setting. In order to achieve the convergence, a very small stepsize is required so that the approximate solution is closed to the exact solution. Razali, Nopiah and Othman (2018) did mentioned about the ways of computing the approximate solution so that it is close to exact solution. A specific tolerance and a right method selections is necessary so that the estimated error lies within the given tolerance at each step and the stepsize for the next step can be predicted which generate an error within the tolerance.

In 2015, an investigation regarding variable stepsize setting based on reference separation system for online blind source separation (BSS) was done by Xu, Yuan, Jian, and Zhao (2015). BSS is about extracting the latent unknown source signals from their observed mixtures by an array of sensors without highlighting the original source signals and the mixing coefficients. In order to improve the learning rate and stability performance, they proposed a new variable stepsize algorithms. During the iteration, there is increasingly in terms of the correlation between the estimated and original source signals. To overcome this, the reference separation system was introduced to approximately estimate the correlation in terms of mean square error (MSE). The MSE is important in updating the stepsize. In their simulations, they demonstrated that the proposed method exhibits good convergence rate and gives excellent performance than the constant stepsize setting for the noise-free case. Aside from that, their proposed

method is also converging faster than the classical variable stepsize setting in both stationary and nonstationary environments.

A research regarding the variable stepsize algorithm for solving the nonlinear Duffing oscillator (second order nonlinear initial-value ODEs) was investigated by Rasedee et al. (2017). A variable order stepsize (VOS) together with the backward difference formulation (BDF) was introduced to solve the numerical approximation of Duffing oscillator. BDF is functioning in overcome an uninteresting calculations of integration coefficients everytime the stepsize make a changes as required by the divided difference formulation that based on the Direct Integration (DI) method. A further work regarding the Duffing oscillator can be found in Branch and Manshahr (2016) and Najafi and Nemati (2017). The VOS with backward difference (VOSBD) method was tested on several nonlinear Duffing oscillators of different parameters. Their numerical approximations shows that the DI method gives an excellent behaviour for larger tolerances whereas the VOSBD is better with a stringent tolerance. It can be concluded that the VOS algorithm provides an efficient computational code without affecting its accuracy.

As we concern, the variable stepsize setting is very important to be implemented as many researchers are still finding the best way that suitable to solve certain problems either in mathematical, biological, chemical, physical, engineering or in any related fields. In 2017, a new generalized variable stepsize was investigated by Wang, Zhou, Wang, and Chen (2017) that involving the CQ algorithm for solving the split feasibility problem (SFP). The proposed technique consists of two algorithms, namely CQ algorithm with two simpler variable stepsizes and two general KM-CQ algorithms with

generalized variable stepsizes. Both of the general algorithms with the generalized variable stepsizes able to solve the SFP and solve some special variational inequality even better. The models that being used in this investigation are the compressed sensing and deconvolution models. The proposed stepsizes with the former ones are then compared with those models and the numerical results appear to give an excellent behaviour.

Apart of SDEs, there also exists a stochastic delay differential equations (SDDEs) and an equation known as stochastic pantograph differential equations (SPDEs) are parts of it. This was studied by Yang, Yang and Xiao (2020). The exact solution of nonlinear SPDEs was introduced by Guo and Li (2019) and established the Razumikhin-type theorems on the αth moment polynomial stability. Yang, Yang, Wang, and Han (2019) are the one that investigated the mean-square stability of nonlinear SPDEs. The numerical solutions of SPDEs by using constant stepsize are investigated by many researchers previously (Fan, Song & Liu, 2009). When the difficult problems is applied throughout the investigation, it leads to the limited computer memory that encourages the researchers to implement the variable stepsize setting and transformation approach for the deterministic pantograph equations to solve the storage problem. Yang et al. (2020) were originally investigated the asymptotical mean-square stability under variable stepsize for linear SPDEs by using linear θ-methods. Linear θ-methods is also categorized as A-stable as proved by Liu (1995). From the investigation by Yang et al. (2020), they proved that the stability region of linear θ-methods by using variable stepsize is the same as the deterministic problems where $\theta \in \left(\frac{1}{2}, 1\right]$.

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Introduction

In this research, only 2-stage and 3-stage Gauss methods will be focused. It will covers the construction of the 2-stage and 3-stage Gauss methods using MATLAB R2019a software. The construction are based on the implementation schemes by Hairer & Wanner (1999) and González-Pinto et al. (1994, 1995). Besides, this chapter also include the MATLAB code for the implementation methods. The implementation are done based on Newton-Raphson iteration. The Newton-Raphson iteration for $f(x) = 0$ where $f(x): \mathbb{R}^n \to \mathbb{R}^n$ is given by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},\ f'(x_n) \neq 0. \tag{3.1}$$

The simplified Newton method evaluates Jacobian once while the full Newton method evaluate Jacobian many times throughout the iterations (Hairer & Wanner, 1996). Only simplified Newton was used in solving test and real life problems. To complete this research, the efficiency of 2-stage and 3-stage Gauss methods has been compared based

on different schemes proposed by other researchers using different problems taken from Enright, Hull, and Lindberg (1975) and Gorgey (2012).

3.2 Research Design

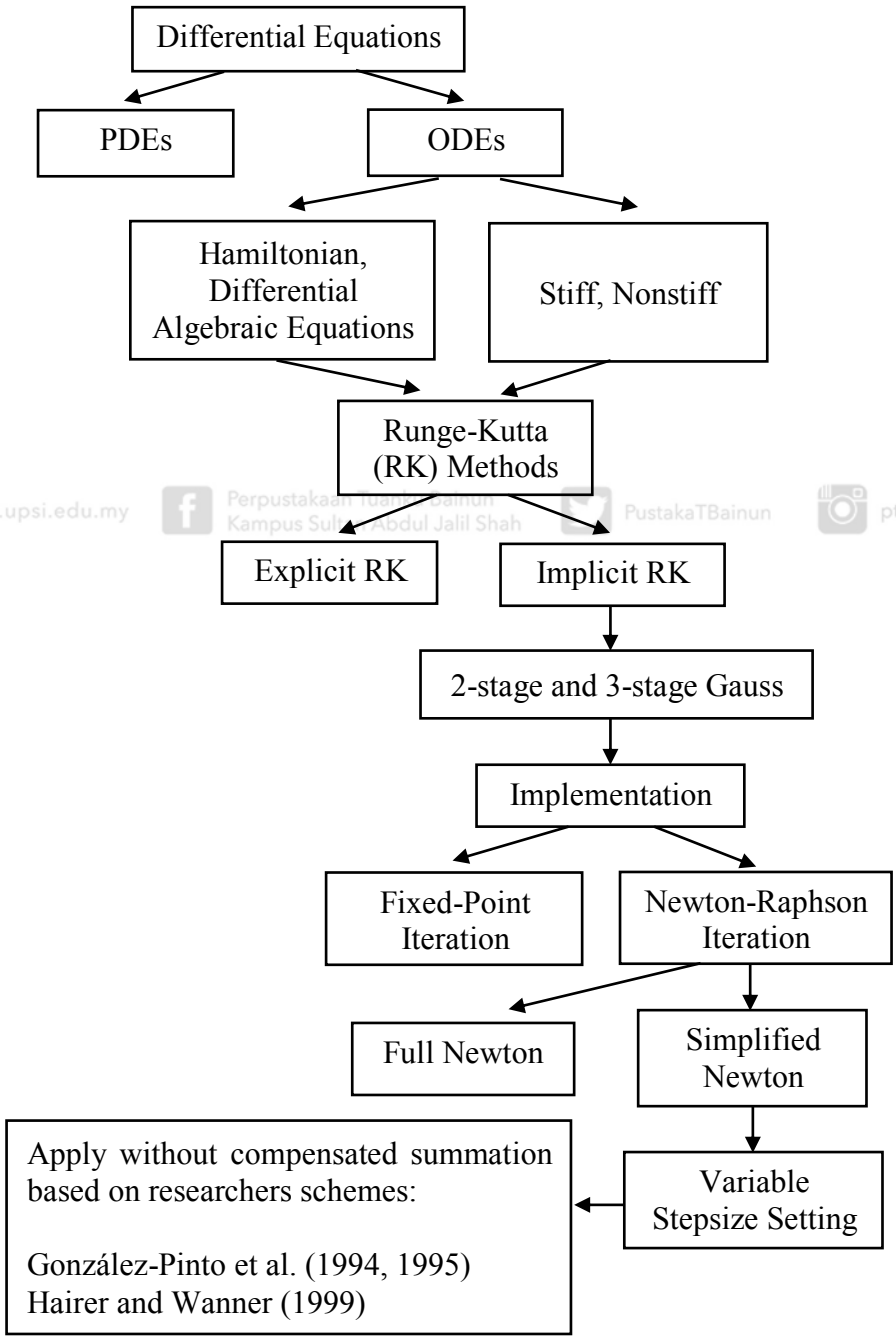


Figure 3.1: Research Design

Figure 3.1 represents a research design for the thesis. The thesis starts by choosing the proper type of differential equations. Since the most important purpose of this research is to find an efficient implementation technique in solving stiff problem, thus only ordinary differential equations (ODEs) are considered and it can be clarify by using Runge-Kutta (RK) methods. 2-stage (G2) and 3-stage (G3) Gauss methods are a family of implicit Runge-Kutta (IRK) methods and are chosen since they are efficient in solving stiff problems (Chan & Gorgey, 2013). In 1964, Butcher presented RK methods based on the Radau and Lobatto quadrature formulas. Since Hairer and Wanner (1999) implementation scheme was originally implemented to Radau IIA method, therefore this method of order-3 was also being tested in this research to clarify either we implemented the right implementation scheme for the G2 and G3 methods.

Although IRK methods are advantages in solving stiff problems, however they are difficult to implement if compared to explicit RK (ERK) methods as stated in section 1.2 under problem statement. The stage equation (1.8) need to be solved using Newton-Raphson iteration. Some computationally cheaper variants are often being used when at each Newton iteration, the methods required the s evaluations of the Jacobian matrix $\partial f / \partial y$ and a LU decomposition of a $sd \times sd$ matrix (Antoñana, Makazaga, & Murua, 2018). Since the problems that being investigated in this research is considered as a stiff problems, thus the fixed point iteration is no longer appropriate to be used and hence the Newton iteration is implemented to compute the stage vectors Y_i from equation (1.6). Even though the Newton iteration is suitable in solving stiff problems, it does not means that the non-stiff problems cannot implemented this iteration. The Newton iteration may be still be an attractive choice where in some cases, the quadruple precision or in arbitrary precision arithmetic calculations with high precision

computations is implemented with mixed-precision strategies in order to reduce the cost of the linear algebra and also the evaluation of the Jacobians that was performed in lower-precision arithmetic than the evaluations of the right-hand side of the system of ODEs (Baboulin et al., 2009).

Newton-Raphson iteration can be divided into two parts which are full Newton and simplified Newton. Full Newton iteration is preferred for non-stiff problems (Muhammad & Gorgey, 2018). Since the problems that are chosen for this research are categorized as stiff problems, therefore only simplified Newton was considered throughout the research. In difficult nonlinear ODEs problems, constant stepsize setting will require more computational time to solve depends on the stiffness ratio as mentioned previously in Section 1.2. In order to overcome this, a variable stepsize setting was investigated in detailed using implementation schemes from Hairer and Wanner (1999) and González-Pinto et al. (1994, 1995) to investigate their effectiveness and efficiencies in solving stiff real life problems.

In solving linear and nonlinear problems, it is preferable to choose a higher order methods such as Radau IIA method of order-5 and G3 method of order-6. This is because the higher order methods are having the tendency to give a greater accuracy than lower order methods (Ismail & Gorgey, 2015). Lower order methods require smaller stepsize than higher order methods. If the stepsize is chosen to be very small, then this can lead to round-off error where eventually will destroy the solution. Thus, a compensated summation technique is applied at the beginning of the code to minimize the effect of round-off errors. A smaller quantities $Z_i = Y_i - e \otimes y_0$ as suggested by Hairer and Wanner (1996) and Butcher (2016) being used together with the code in

order to reduce the influence of round-off errors. Chan and Gorgey (2013) and Gorgey and Chan (2015) had mentioned that compensated summation is very useful when solving a stiff problem that requires a very small stepsize and also when the accuracy of the numerical solutions need to be increased by extrapolation technique. A comparison between simplified Newton with compensated summation (SNCS) and without compensated summation (SNWCS) using variable stepsize setting by Hairer and Wanner (1999) and González-Pinto et al. (1994, 1995) implementation schemes has been investigated. However, the numerical results shows that there is no effect in terms of accuracy on G2 method with simplified Newton and compensated summation (G2SNCS), therefore no compensated summation is needed for this research as explain in Section 1.2 under problem statement. Since there is no effect on compensated summation using variable stepsize setting, the Matlab code are proceed without compensated summation. The numerical results will be discussed in Chapter 5.

3.3 Construction of G2 and G3 Methods

Implicit Runge-Kutta (IRK) methods are called A-stable if there are no stability constraints for $y' = \lambda y$, $\mathrm{Re}\,\lambda < 0$ and $h > 0$. Dahlquist (1963) introduced this concept for linear multistep methods, but this concept is also practiced to RK method processes. A further explanations can be found in Hairer and Wanner (1996). Since their stability properties has been proven, hence IRK methods are the main methods used in this research. G2 and G3 methods are chosen since these two methods are convenient for the solution of stiff differential equations.

An *s*-stage Gauss method satisfies $B(2s)$, $C(s)$ and is of classical order $2s$. The abscissas are the zeros of the shifted Legendre polynomial $P_s(2s-1)$, where $P_s(x)$ denoted the Legendre polynomial of degree s defined on the interval [1,1] (Williams, 2017). Few shifted Legendre polynomials are shown in Table 3.1.

Matrix A can be constructed using the equation $A = CSDS^{-1}$. The C is the root obtained from the Legendre equation, S is the Vandermonde matrix for C, and D is the diagonal matrix $\text{diag}\left(1, \frac{1}{2}, ..., \frac{1}{s}\right)$. The definition for the Vandermonde matrix is given in Definition 3.3.1 on page 44.

Table 3.1

The first few shifted Legendre polynomials

s	$P_s(x)$
0	1
1	$2x-1$
2	$6x^2-6x+1$
3	$20x^3-30x^2+12x-1$

Other investigation regarding shifted Legendre polynomials was experimented by Wang and Chen (2020). They mentioned that the shifted Legendre polynomials algorithm will increase the reliability on predicting the viscoelastic behaviors and dynamic properties regarding the pipes conveying fluid problem.

Definition 3.3.1 *Vandermonde matrix (V) can be defined as follows (Butcher, 2016)*

$$V = \begin{bmatrix} 1 & c_1 & c_1^2 & \cdots & c_1^{n-1} \\ 1 & c_2 & c_2^2 & \cdots & c_2^{n-1} \\ 1 & c_3 & c_3^2 & \cdots & c_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \cdots \\ 1 & c_m & c_m^2 & \cdots & c_m^{n-1} \end{bmatrix},$$

where the *c*'s is taken from the *C* roots. In linear algebra, Vandermonde matrix is a matrix with the terms of a geometric progression in each row which is written as $m \times n$ matrix that was named after Alexandre-Théophile Vandermonde (1735-1796). A thorough discussion on this can be found in Ycart (2012). Yaici and Hariche (2019) did mentioned that both the Vandermonde matrix and its inverse are often appointed in the control theory, derivation of numerical formulas and in the systems theory. It is also very important for the solution of polynomial interpolation. The discovery of the Vandermonde matrix was found around 1965 and even before, where many researchers deals with the study and its properties, inverse and its determinant (Rushanan, 1989). In Ye (2017), he proved that every generic $m \times n$ matrix is a product of Vandermonde matrix and its transpose. Kim and Kräuter (2018) also mentioned that Vandermonde matrix is decomposed in order to obtain the variants of the Lagrange interpolation polynomial.

The Legendre equation $P_2(x) = 6x^2 - 6x + 1$ (refer to Table 3.1) was used for 2-stage Gauss (G2) method and having the roots for the equation $\left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right)$ and $\left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right)$. These roots are the *c*'s as in the Butcher tableau (refer to Table 1.2). The construction of the G2 method is given next.

$$A = CSDS^{-1},$$

$$= \begin{bmatrix} \frac{1}{2}-\frac{\sqrt{3}}{6} & 0 \\ 0 & \frac{1}{2}+\frac{\sqrt{3}}{6} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2}-\frac{\sqrt{3}}{6} \\ 1 & \frac{1}{2}+\frac{\sqrt{3}}{6} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} \frac{\sqrt{3}+1}{2} & -\frac{\sqrt{3}-1}{2} \\ -\sqrt{3} & \sqrt{3} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{4} & \frac{1}{4}-\frac{\sqrt{3}}{6} \\ \frac{1}{4}+\frac{\sqrt{3}}{6} & \frac{1}{4} \end{bmatrix}. \qquad (3.2)$$

Similarly, the *c* value for the 3-stage Gauss method is obtained by solving the Legendre polynomial for P_3. The roots are given as $\left(\frac{1}{2}-\frac{\sqrt{15}}{10}\right), \left(\frac{1}{2}\right)$ and $\left(\frac{1}{2}+\frac{\sqrt{15}}{10}\right)$. The shifted Legendre polynomial for G3 is $P_3(x) = 20x^3 - 30x^2 + 12x - 1$.

$$A = CSDS^{-1},$$

$$\text{Solving for } DS^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{3} \end{bmatrix} \begin{bmatrix} \frac{5+\sqrt{15}}{6} & -\frac{2}{3} & \frac{5-\sqrt{15}}{6} \\ \frac{-\sqrt{15}-10}{3} & \frac{20}{3} & \frac{\sqrt{15}-10}{3} \\ \frac{10}{3} & -\frac{20}{3} & \frac{10}{3} \end{bmatrix},$$

$$= \begin{bmatrix} \frac{5+\sqrt{15}}{6} & -\frac{2}{3} & \frac{5-\sqrt{15}}{6} \\ \frac{-10-\sqrt{15}}{6} & \frac{10}{3} & \frac{\sqrt{15}-10}{6} \\ \frac{10}{9} & -\frac{20}{9} & \frac{10}{9} \end{bmatrix},$$

$$A = CSDS^{-1},$$

$$= \begin{bmatrix} \frac{1}{2}-\frac{\sqrt{15}}{10} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2}+\frac{\sqrt{15}}{10} \end{bmatrix} \begin{bmatrix} 1 & \frac{1}{2}-\frac{\sqrt{15}}{10} & \frac{2}{5}-\frac{\sqrt{15}}{10} \\ 1 & \frac{1}{2} & \frac{1}{4} \\ 1 & \frac{1}{2}+\frac{\sqrt{15}}{10} & \frac{2}{5}+\frac{\sqrt{15}}{10} \end{bmatrix} \begin{bmatrix} \frac{5+\sqrt{15}}{6} & -\frac{2}{3} & \frac{5-\sqrt{15}}{6} \\ \frac{-10-\sqrt{15}}{6} & \frac{10}{3} & \frac{\sqrt{15}-10}{6} \\ \frac{10}{9} & -\frac{20}{9} & \frac{10}{9} \end{bmatrix},$$

$$= \begin{bmatrix} \frac{5}{36} & \frac{2}{9}-\frac{\sqrt{15}}{15} & \frac{5}{36}-\frac{\sqrt{15}}{30} \\ \frac{5}{36}+\frac{\sqrt{15}}{24} & \frac{2}{9} & \frac{5}{36}-\frac{\sqrt{15}}{24} \\ \frac{5}{36}+\frac{\sqrt{15}}{30} & \frac{2}{9}+\frac{\sqrt{15}}{15} & \frac{5}{36} \end{bmatrix}. \tag{3.3}$$

Matrices (3.2) and (3.3) are obtained by using Maple 2019 mathematical software. A detailed explanations regarding Legendre polynomials can be found in Butcher (2016).

3.4 Implementation of Implicit Runge-Kutta Methods

The 2-stage (G2) and 3-stage (G3) Gauss method has been implemented by using simplified Newton of Newton-Raphson iteration. The implementation idea is taken from Hairer and Wanner (1999) and González-Pinto et al. (1994, 1995) which had been modified according to the Gaussian method. Previously, Hairer's code was done for the Radau IIA method of order-3 where the eigenvalues are real (Hairer & Wanner, 1999). However for this research, the Hairer's code was tested on G2 and G3 methods to investigate their accuracy and efficiency.

Consider applying IRK method for solving the initial value problem (IVP) of an ODEs with dimension *N* which is given by

$$y'(x) = f(x, y), \quad y(x_0) = y_0 \ . \tag{3.4}$$

Generally, the approximate solution obtained by an *s*-stage RK methods with stepsize h for the interval $[x_0, x_n]$ can be defined by the following equations (Butcher, 2016):

$$\begin{aligned} Y^{[n]} &= e \otimes y_{n-1} + h(A \otimes I_N) F\left(x_{n-1} + ch, Y^{[n]}\right), \\ y_n &= y_{n-1} + h\left(b^T \otimes I_N\right) F\left(x_{n-1} + ch, Y^{[n]}\right), \\ x_n &= x_0 + h, \end{aligned} \tag{3.5}$$

where $\otimes$ denotes the Kronecker product, $e = (1,...,1)^T$ and I_N is the $N \times N$ identity matrix and y_n is the update of the RK method. y_n will be updated until the approximate solution is obtained for each problem that being tested. Normally the numerical solution is approximated until the desired solution is obtained or until the approximate solution reached the target interval x_n.

In equation (3.5), it can be seen that the function hF is computed to find the interval stages as well as to find the update y_n. This can be a waste of computational time. Therefore, it is recommended by Hairer and Wanner (1996) to write the update of equation (3.5) as given by:

$$y_n = y_{n-1} + b^T A^{-1}\left(Y^{[n]} - e \otimes y_{n-1}\right). \tag{3.6}$$

In addition to that, to make sure that the influence of round-off errors is reduced, it is also suggested by Hairer and Wanner (1996) to use a smaller quantity such that

$$Z^{[n]} = Y^{[n]} - e \otimes y_{n-1} \ . \tag{3.7}$$

Equation (3.5) then can be written in the form

$$Z^{[n]} = h\left(A \otimes I_N\right) F\left(x_{n-1} + ch, Z^{[n]} + e \otimes y_{n-1}\right), \tag{3.8}$$

where $Z^{[n]}$ consist of $sN \times 1$ vector which is given by

$$Z^{[n]} = \begin{bmatrix} Z_1 \\ \vdots \\ Z_s \end{bmatrix}.$$

The RK method such as given in equation (3.8) is nonlinear because of the difficulties that occurs in solving for $Z^{[n]}$. However, this complexities can be figure out by implemented the Newton-Raphson iterative method for N dimensional system of equation such as

$$Z^{[n+1]} = Z^{[n]} - \frac{F\left(Z^{[n]}\right)}{J\left(Z^{[n]}\right)}, \tag{3.9}$$

where $F\left(Z^{[n]}\right)$ is given by $sN \times 1$ system of equation

$$F\left(Z^{[n]}\right) = \begin{bmatrix} f\left(x_{n-1} + c_1 h, z_1 + e \otimes y_{n-1}\right) \\ \vdots \\ f\left(x_{n-1} + c_s h, z_s + e \otimes y_{n-1}\right) \end{bmatrix},$$

and the Jacobian matrix, $J\left(Z^{[n]}\right) = \frac{\partial f}{\partial z}$ is computed such that

$$J\left(Z^{[n]}\right) = \begin{bmatrix} \frac{\partial f_1}{\partial z_1} & \frac{\partial f_1}{\partial z_2} & \cdots & \frac{\partial f_1}{\partial z_N} \\ \frac{\partial f_2}{\partial z_1} & \frac{\partial f_2}{\partial z_2} & \cdots & \frac{\partial f_2}{\partial z_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_N}{\partial z_1} & \frac{\partial f_N}{\partial z_2} & \cdots & \frac{\partial f_N}{\partial z_N} \end{bmatrix}. \tag{3.10}$$

For simplicity, consider writing equation (3.8) as

$$G\left(Z^{[n]}\right) = Z^{[n]} - h\left(A \otimes I_N\right) F\left(x_{n-1} + ch, Z^{[n]} + e \otimes y_{n-1}\right). \tag{3.11}$$

To solve for $Z^{[n]}$, we need to find the Jacobian matrix as given in equation (3.10). This can be obtained by taking the derivatives of equation (3.11) with respect to Z such that

$$D_G\left(Z^{[n]}\right) = I_N - h\left(A \otimes I_N\right) J\left(x_{n-1} + ch, Z^{[n]} + e \otimes y_{n-1}\right). \tag{3.12}$$

The values of $Z^{[n]}$ can be obtained by solving equations (3.11) and (3.12) using Newton-Raphson iteration method introduced in equation (3.9) such that

$$\Delta Z^{[n]} = -G\left(Z^{[n]}\right)\left(D_G\left(Z^{[n]}\right)\right)^{-1}, \tag{3.13}$$

where,

$$\Delta Z^{[n]} = Z^{[n+1]} - Z^{[n]}.$$

The update y_n as given in equation (3.6) is therefore given by

$$y_n = y_{n-1} + b^T A^{-1} Z^{[n]}. \tag{3.14}$$

All of this transformation was introduced in Hairer and Wanner (1996). For cheaper implementation cost, the coefficient matrix $D_G\left(Z^{[n]}\right)$ is only evaluated at the starting of the iteration. The rest of the computations used the same value of $D_G\left(Z^{[n]}\right)$ and this implementation is known as simplified Newton-Raphson method.

CHAPTER 4

IMPLEMENTATION OF G2 AND G3 METHODS

4.1 Implementation Issues

In solving ordinary differential equations (ODEs) problems, the main issues in the implementation method that need to be considered is the strategy to achieve high accuracy, high efficiency and low implementation cost. These criteria are the main objectives in getting a good numerical approximation because it is related to computational cost. There are several aspects that need to be considered in achieving the objectives. Some of them are the convergence, tolerance, initial values and round-off errors that will be discussed in the next subsection.

4.1.1 Convergence

In getting a good numerical approximation, the problems must satisfy the convergence test which the convergence rate is given by $\left\|\Delta Z^{k+1}\right\| \le \Theta \left\|\Delta Z^{k}\right\|$ where $\Theta < 1$ and $k \ge 1$

(Hairer & Wanner, 1996). It is considered satisfies the convergence test when the approximate solution from step to step are approaching towards the exact solution. Meaning to say, the convergence test is a relevant identification method in the iteration of the numerical approximation. Convergence is a type of numerical method related to truncation errors that provides the numerical solution to converge onto the exact solution and when the truncation error approach zero at all stepsize indices in the limit $\Delta h \to 0$. As the stepsize become smaller, the maximum absolute global truncation error between the analytical and numerical solutions is giving a smaller error. Atkinson, Han and Steward (2009) gave a brief explanation regarding this matter. The mechanism to measure the convergence test is referring to how stepsize, computational (CPU) time and tolerance behaves with the global error in the numerical approximations. The tolerance and stepsize are the identical criteria in deciding the converging test when approaching the exact solution. It is said approaching the convergence if the tolerance and stepsize are decreasing proportionally with global error. For the CPU time, it is approaching convergence if the global error decreases as the CPU time increases. The accuracy is determined by the tolerance and stepsize while the efficiency is shown by the CPU time graphs that obtained from the numerical approximation. In this scenario, the access of the accuracy is important before the efficiency because the accuracy can determine whether the implementation is correct or wrong based on the order of the IRK method. In other words, we can say that the accuracy is affected by the order of the method which of order-4 for G2 method and order-6 for G3 method.

In deciding the most efficient implementation among researchers, it is important to make a comparison which scheme is having the least error and the least CPU time taken in solving the ODEs problems. The values selection for κ and the number of

iteration are also plays an important role in improving the convergence test. It is suggested in Hairer and Wanner (1996) that the most efficient values of κ is around 10^{-1} or 10^{-2} and this values was tested for the code RADAU5. They also mentioned that the code becomes efficient with the use of relatively high number of iteration of 7 or 10. During this iteration processes, it helps the computations to restart the iteration with a smaller stepsize $(h/2)$ in condition where $\Theta \geq 1$. If this case happened, the computations is interrupted that lead the iteration to become diverge. We can conclude that the convergence test is a crucial issue to help the researchers to do the troubleshooting and avoid unnecessary computations in case the approximate solution goes wrong and not converging, furthermore gives beneficial to them in saving time.

4.1.2 Tolerance

Investigation regarding tolerance value selection was done by Hairer and Wanner (1996). From the investigation, it shows that the code RODAS which is referring to Rosenbrock's codes of order 4 with an embedded order 3 error estimator is considered giving best behaviour for low tolerances whereas the code RADAU5 which refers to Radau IIA method with $s = 3$ of order 5 is recommended for high precision. As the tolerances become smaller, the more precise the numerical approximation is for the longer CPU time. The code was tested using Van der Pol, Robertson and Oreganator problems together with different methods such as RODAS, LSODE, SEULEX and RADAU5. SEULEX is an extrapolation code which implement the stiff linearly implicit Euler extrapolation method. For LSODE, Hindmarsh (1980) was the first to

implement this code that refers to backward difference formula (BDF) which is the model for a class of multistep methods.

From the numerical approximation, the Rosenbrock's code RODAS is giving the best behaviour for low tolerances between 10^{-2} to 10^{-5}, while the extrapolation code SEULEX is superior for stringent tolerances. The more stringent the value of tolerances, the easier the method to solve the problems. However, it is preferred to use not a very stringent tolerance as suggested by Muhammad (2018). Due to the cheapness of the function evaluations by multistep code LSODE, more computing time is required in general compared to one-step codes does. The code RADAU5 gives the most definite result for the code where the tolerance value is $\mathrm{Tol} = 10^{-5}$ followed by RODAS, SEULEX and LSODE. Furthermore, it has been proven that using smaller tolerances gives the precise solution. Since RADAU5 gives the precisest result among the others and are part of a family of IRK methods, thus we are interested in investigating the numerical approximation using different IRK methods such as G2 and G3 methods with the use of the same tolerance value, $\mathrm{Tol} = 10^{-5}$ or using tolerance value which is smaller than that.

4.1.3 Initial Value

An initial value problems (IVPs) is an ordinary differential equations (ODEs) together with an initial condition or best known as initial value which specifies the value of the unknown function at a given point in the domain. Initial value are frequently needed values in solving the IVPs that involving a modelling system in mathematics, physics

and other sciences. In other words, the differential initial value is referring to an equation which specifies how the system evolves with time given the initial conditions. A proper selection of the initial value for ODEs is very important because it can be an alternative approach to achieve converging solution. By selecting the proper initial value, it can avoid the codes fail if the initial value inappropriate. Basically, most of the ODEs have the initial value. It is belongs to the variables in the ODEs. The initial value is generally assigned as x_0 and y_0 in the ODEs problem. The number of initial value is the same as the dimension of the ODEs. Practically, the initial value can be changed accordingly in the implementation in order to achieve converging solution. To get the best initial value, basically the researchers will carry out a task for try and error until the solution converge and achieving their accuracy and efficiency.

4.1.4 Round-off Errors

Round-off errors is an error created due to approximate representation of number (Butcher, 2016). This happened when the stepsize chosen is very small that can destroy the solution. Thus, it is suggested to use not a slightly small stepsize to avoid the round-off error from accumulate at the numerical approximation. When this is happened, it will affect the accuracy of the iteration and thus cannot represent the order of the IRK methods. However, some stiff ODEs problems are demanding of using relatively smaller stepsize in order to achieve convergence. To reduce the effect of round-off error when smaller stepsize is applied, it is suggested to apply a technique known as compensated summation. Compensated summation is a technique used to minimize the effect of round-off error and therefore beneficial in improving the accuracy and

efficiency. Higham (1993) explained further about compensated summation in their research. He did mentioned about the instability sometimes is not caused by the accumulation of millions of rounding errors, but by the dangerous growth of just a few rounding errors. The compensated summation works as capturing the round-off error at each individual step where the round-off error is gathered for y-values. Thus, the compensated summation is very important to be implemented in getting better numerical approximation especially when extrapolation is applied together. Detailed investigation on a numerical results regarding the use of other compensated summation is also given in Antoñana et al., (2018).

Even though compensated summation technique gives a lot of advantages, however this technique is not applied to this research. This is regarding the numerical approximation that has been tested on the Prothero-Robinson problem. The numerical analysis shows no requirement in using compensated summation with simplified Newton for variable stepsize setting. This behaviour was explained in details in Chapter 1 under problem statement section. Hence, the Gauss methods of order 2 (G2) and of order 3 (G3) were implemented only with simplified Newton to study the behaviour in achieving the convergence.

4.2 Variable Stepsize Setting

In achieving the convergence faster, one of the strategy that can be followed for the implementations of IRK methods is by employing the variable stepsize instead of constant stepsize. Variable stepsize are very useful in getting excellent performance for

IRK methods. A stepsize control formula that was originally proposed by Gustafsson, (1994) based on the two-step estimator, which in combination with the standard one-step estimator, proved that the code RADAU5 fails less steps in the integration of some stiff problems (Hairer & Wanner, 1996).

A research regarding the variable stepsize control for Radau IIA methods has been done by González-Pinto, Hernandez-Abreau, and Montijano (2019). In their research, they proposed a new strategy in pursuing a variable stepsize setting nevertheless not an extension of Gustafsson (1994). They mentioned that the two-step estimator does not needed any additional evaluation of the derivative function unlike the one-step estimator does. This gives a briefly explanation that when the variable stepsize setting is used, no filtering is needed for that estimator and thus gives beneficial to the codes which save some extra solutions of real linear systems that are required by the one-step estimator. From the numerical results obtained, it shows that the code takes slightly smaller number of steps which is 4897 steps for the two-step estimator, while one-step estimator gives a value of 4923 steps for the same tolerance $(\mathrm{Tol} = 10^{-12})$ and this results has been tested for Van der Pol problem. Furthermore, stepsize control for tolerance proportionality has also been considered giving nice global errors with the supplied tolerances. However, one-step estimator gives fewer Jacobian evaluations than two-step estimator but the error produced much bigger than two-step estimator. We can summarized here that two-step estimator or variable stepsize setting gives an efficient results in solving stiff problems.

4.2.1 Error Estimation

In solving stiff ODEs problems of IRK methods, extrapolation technique has been introduced as an alternative for local error estimation and is applied together with G2 and G3 methods. Bader and Deuflhard (1983) introduced a METANI code, where this code was known as a first successful extrapolation code for stiff differential equations which implements the linearly implicit midpoint rule. Extrapolation is a technique to enhance the stability and efficiency of a method. The general equation of extrapolation is given by

$$\bar{y} = \frac{2^p (y_2) - y_1}{2^p - 1}, \tag{4.1}$$

where p is the order of the RK methods and y_2 and y_1 are the solutions attained by using stepsizes, h and $h/2$ respectively. The difference between y_1 and y_2 gives the local error estimation. This step halving or best known as step doubling in obtaining the local error estimation was introduced by Shampine (1985). This technique is also known as Richardson extrapolation. Extrapolation can be found in two difference modes such as active and passive modes. Active extrapolation happened when the value of extrapolation is used to capture the next computation while passive extrapolation occurs when there is no need in using the extrapolated value for any subsequent computations (Ismail & Gorgey, 2015).

Since extrapolation can increase accuracy and efficiency, many researchers are still finding the best ways to apply extrapolation. Gorgey (2012) showed that passive extrapolation of the G2 method is more competent than the active extrapolation for linear problems that using constant stepsize setting. In addition to that, Faragó, Havasi,

and Zlatev (2013) found out that computational time by using Richardson extrapolation for both active and passive modes are more than ten times smaller than the corresponding computing time by the backward Euler formula. Thus, they concluded that extrapolation is an impressive technique for increasing the accuracy and efficiency with taking into account the computational cost especially when the accuracy condition is not too low. Another approaches was investigated in Gorgey and Mat (2018) regarding the efficiency of IRK methods in solving simple harmonic oscillators. After a very short period of time, they concluded that passive extrapolation is observed to produce quadratic error growth while for active extrapolation, a linear error growth is obtained for a much longer period of time. It can be summarized here that the numerical results for active extrapolation is observed to give the lowest error if compared with passive extrapolation. Therefore, there is only one mode that can be applied in the variable stepsize setting which is the active mode.

Another approach to estimate the local error was given by Gorgey (2012). In her thesis, symmetrizer is used to estimate the local error for G2 method. Although the error estimation by using symmetrizer is efficient, the computational time between this approaches with the traditional error estimation by the extrapolation is not much different. Therefore, local error for the variable stepsize in this article is estimated using extrapolation technique.

The variable code that estimate the local error started by setting the coefficient $x = x_0$, $p = 4$ which is the order of G2 method and $p = 6$ which is the order of G3 method. We then set up the minimum and maximum h values that is required for the problem by setting

$$h_{\max} = (x_n - x)/16 \ , \ h_{\min} = (x_n - x)/2\times 10^8 \ . \tag{4.2}$$

To make sure that we are choosing the correct value of h for each problem, set

$$h = \max\left(\left[h_0, (x_n - x)/10^7\right]\right). \tag{4.3}$$

If $x + h > x_n$, then the first value of the update for stepsize h which is y_1 is computed. Then, another two steps of the update is computed for $h/2$ which is denoted by y_2. Upon obtaining the values of y_1 and y_2, the local error is estimated such that

$$\varepsilon = y_2 - y_1, \tag{4.4}$$

where $\delta = \|\varepsilon\|_\infty$ and $\tau \leftarrow \text{Tol.}\max\left(\|y_1\|_\infty, 1.0\right)$.

If $\delta \le \tau$, thus the new value of x and the improved result, $y = y_2 + \varepsilon$ is computed. A sophisticated stepsize strategy has been used in deciding the stepsize selection. It leads to the formula

$$h \leftarrow \min\left(\left[h_{\max}, 4h, 0.9h\left(\tau/\delta\right)^{\frac{1}{p+1}}\right]\right), \tag{4.5}$$

If condition (4.5) is satisfied, then the h value is accepted. Otherwise we reject it and the h formula is recomputed by using the following condition

$$h \leftarrow h.\max\left(\left[0.25, 0.9\left(\tau/\delta\right)^{\frac{1}{p+1}}\right]\right). \tag{4.6}$$

This variable code implementation is introduced in Hairer and Wanner (1996).

4.3 Implementation Strategies

The strategies that being used for the implementation of the RK methods in this research is simplified Newton of Newton-Raphson iteration. As mentioned in the previous

chapter, no compensated summation is needed for the iteration of variable stepsize setting. Newton-Raphson is chosen because of the capability in solving the IRK methods and gives better solution compared to fixed-point iteration. Thus, it gives efficient implementation for the numerical approximation. Hairer and Wanner (1996) were the first to implement the simplified Newton technique and Radau IIA method of order 5 is chosen as a method that being tested. Since then, it became popular for the solution of stiff problems. Since the Jacobian is only evaluated once before Newton iteration, therefore this strategy provide less computational cost that gives beneficial in solving stiff ODEs problems.

In Antoñana et al. (2018), they also implemented the simplified Newton iterations in getting efficient implementation for the symplectic IRK schemes. Based on their investigation, when the value of the stiffness constant is increasing the simplified Newton iteration that being implemented requires more iterations per step. This observation motivated them to make a modification to the original simplified Newton iteration to produce new algorithm which is known as Kahan's compensated summation. This algorithm requires an evaluations of the Jacobian matrix which at each step s additional evaluations is required. In their numerical experiments using stiff pendulum problem, the use of that algorithm does improve efficiency which is reduce the number of iterations also shows a robustness. However, there is no interest of using this algorithm for this research, thus only standard simplified Newton is implemented throughout the research.

In our MATLAB implementation of variable stepsize, there are three script files of pseudo code for simplified Newton method respectively. The first script file solves

the nonlinear part of the method (see Algorithm 4.3.1), the second script file computes the n steps of the base method (see Algorithm 4.3.2) and the third script file is the variable code that estimates the local error (see Algorithm 4.3.3).

Algorithm 4.3.1: Newton Iteration $(\Delta Z = 0)$

Set trace = 0, $Y = e \otimes y$ and $\Delta Z = \Delta G / -G$.
Evaluate Jacobian.
Evaluate $\sigma = \|\Delta Z\|_\infty$.
Evaluate $\eta = \Theta / (1 - \Theta)$.
if $\eta.\sigma \le \kappa.\text{Tol}$, where $\kappa = 10^{-1}$
$\begin{cases} YY \leftarrow Y + \Delta Z \\ \text{else } Z \leftarrow Z + \Delta Z \end{cases}$
for $i \leftarrow 1$ to 10
do $\begin{cases} \text{Evaluate } F(Z) \text{ using the same Jacobian.} \\ \text{Recalculate } \beta = \|\Delta Z\|_\infty \text{ and } \Theta = \beta / \sigma. \\ \text{if } \Theta \ge 1 \\ \quad \begin{cases} \text{trace} = 1 \\ \eta = 1.0 \end{cases} \\ \text{else if } \Theta^{(10-i)} / (1 - \Theta).\beta > \kappa.\text{Tol} \\ \{\text{trace} = 1 \\ \text{Evaluate } \eta \leftarrow \Theta / (1 - \Theta). \\ \text{Evaluate } \Theta \leftarrow \max(10^{-16}, \Theta)^{0.8}. \\ \text{if } \eta\beta \le \kappa.\text{Tol} \\ \{\sigma \leftarrow \beta \\ Z \leftarrow Z + \Delta Z \end{cases}$
$YY \leftarrow Z$
return (trace)

Algorithm 4.3.2: Constant Stepsize (y)

Set TRACE = 1 and hout=h.
while TRACE
{ TRACE = 0
$x = x_0$
$y = y_0$
$\Theta = 0.5$
for $i \leftarrow 1$ to n
do { if TRACE = 1
{ hout $\leftarrow$ hout / 2
Store the value of Y_1 at the $n+1$-th step
if hout < h
{ TRACE = 1
return (y)
return (TRACE)

In the computational processes, there exists a numerical error where the error is measured by the difference between two components, which are the numerical solution and the exact solution. The efficiency graph that was obtained from the numerical approximation could resolve whether the significant of round-off errors could affected the implementation of the IRK method. For the case when the round-off errors is increasing, the efficiency graph will generate a slope where it will change from negative to positive slope. For some cases, the slope is zero which means the numerical errors is at the same value and it should decreasing along numerical approximation. However, round-off errors is not the only components that could significantly affected the computations. The significant round-off error could be affected if the stepsize used is very small that might destruct the computation (Butcher, 2016).

Algorithm 4.3.3: Variable Stepsize (h)

while $(x < x_n)$ and $(h \geq h_{\min})$

{ if $(x + h > x_n)$

{ $h \leftarrow x_n - x$

if TRACE $h = h_{out}$

{ Estimate the error, $\varepsilon = y_{out1} - y_{out}$

$\delta \leftarrow \|\varepsilon\|_\infty$

$\mathrm{T} \leftarrow \tau.\max\left(\|y\|_\infty, 1.0\right)$

if $(\delta \leq \mathrm{T})$

{ $x \leftarrow x + h$

$y \leftarrow y_{out1} + \varepsilon$

if $(\delta = 0)$

$h \leftarrow \min\left(\left[h_{\max}, 4.h, 0.9.h\left(\mathrm{T}/\delta\right)^{p+1}\right]\right)$

Accept *h*.

else

$h \leftarrow \min\left(\left[h_{\max}, 4.h\right]\right)$.

else

$h \leftarrow h.\max\left(\left[0.25, 0.9.\left(\mathrm{T}/\delta\right)^{p+1}\right]\right)$

4.4 Implementation Scheme by González-Pinto et al. (1994, 1995)

The methods introduced in this research retains the coefficients b_i, c_i and a_{ij} of the 2-stage (G2) and 3-stage (G3) Gauss methods. Implementation schemes by González-Pinto et al. (1994, 1995) is a modification from Cooper and Butcher (1983) where the scheme consists of modified variables such as $S = I$, $L = T^{-1}P$ and $B = I - L$. B and S are $s \times s$ nonsingular real matrices while L is a strictly lower triangular $s \times s$ matrix. Originally, the variable B, S, L and λ (real positive number) in Cooper and Butcher

(1983) were chosen so that the spectral radius of $M(z)$ which is denoted by $\rho[M(z)]$ is minimum for $\mathrm{Re}(z) \le 0$. Their scheme were selected because it was classified as the most efficient implementation for the integration of stiff problems (Peat & Thomas, 1989). Even though the numerical results satisfies the convergence and efficiency behaviour, however the numerical analysis that has been done before is specialize for linear and constant coefficient problems only. Since the convergence for nonlinear stiff problems has not been explored in details, therefore González-Pinto et al. (1994, 1995) iterative schemes were introduced in solving nonlinear stiff problems for G2 and G3 methods.

The general equation of the iterative scheme given by González-Pinto et al. (1994, 1995) are modified based on equation 3.5 in previous chapter with some modification from Cooper and Butcher (1983). The derivation of the iterative scheme can be found in González-Pinto et al. (1994) and the general equation are given as follows:

$$\begin{aligned} \left[I_N - h\left(T \otimes J\right)\right] E^{[n]} &= Y^{[n]} - e \otimes y_{n-1} + h\left(A \otimes I_N\right) F\left(Y^{[n]}\right), \\ Y^{[n+1]} &= Y^{[n]} + E^{[n]}, \end{aligned} \tag{4.7}$$

where $n = 1, 2, \ldots, s$. In González-Pinto et al. (1994, 1995), the coefficient k is used instead of n. In this thesis, we changed into coefficient n because we want to use the same coefficient as the general equations of Runge-Kutta methods introduced by Butcher (2016). Smaller quantities $Z^{[n]} = Y^{[n]} - e \otimes y_{n-1}$ is applied to equation (4.7) and the new equation of the iteration are given by

$$\begin{aligned} \left[I_N - h\left(T \otimes J\right)\right] E^{[n]} &= Z^{[n]} + h\left(A \otimes I_N\right) F\left(Z^{[n]} + e \otimes y_{n-1}\right), \\ Z^{[n+1]} &= Z^{[n]} + E^{[n]}. \end{aligned} \tag{4.8}$$

There exists matrix T such that T is a real nonsingular constant matrix of dimension s and it contained a unique eigenvalue $\lambda > 0$. This matrix T could advantages in reducing the additional cost that was involved in the implementation.

The matrix T of G2 method is given by

$$T = \begin{bmatrix} \frac{\sqrt{3}}{6} & 0 \\ \frac{\sqrt{3}}{3} & \frac{\sqrt{3}}{6} \end{bmatrix},$$

while the value of matrix T for G3 method is given by

$$T = \gamma S\left(I - L\right)^{-1} S^{-1},\ \omega = 1 - 0.0371745516 \text{ and } \gamma = \sqrt[3]{\frac{1}{120}},$$

$$L = \begin{bmatrix} 0 & 0 & 0 \\ 2\omega & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},\ S = \begin{bmatrix} -0.0455241821 & 0.0441943589 & 0.0721518521 \\ -0.140048242 & 0.139620426 & 0.118832579 \\ 1 & -0.244595668 & 1 \end{bmatrix},$$

$$T = \begin{bmatrix} 0.1190762649202001 & -0.01352480890549548 & 0.002955703944789629 \\ 0.2567321613764653 & 0.2864264722250291 & -0.008257284502425157 \\ 0.2617169889707876 & 0.5210947821158048 & 0.2027174624121108 \end{bmatrix}.$$

The matrix T for G2 method is given by González-Pinto et al. (1994) whereas the matrix T for G3 method is given by González-Pinto et al. (1995).

4.5 Implementation Scheme by Hairer and Wanner (1999)

For implementation scheme by Hairer and Wanner (1999), a new transformation has been introduced and this changes are done to equations (3.11) – (3.13). Firstly, pre-multiply (3.11) by $\left(hA\right)^{-1} \otimes I_N$. This gives

$$G\left(Z^{[n]}\right)=\left((hA)^{-1}\otimes I_N\right)Z^{[n]}-F\left(x_{n-1}+ch,Z^{[n]}+e\otimes y_{n-1}\right). \tag{4.9}$$

Similarly, equation (3.12) becomes

$$D_G\left(Z^{[n]}\right)=\left((hA)^{-1}\otimes I_N\right)-J\left(x_{n-1}+ch,Z^{[n]}+e\otimes y_{n-1}\right). \tag{4.10}$$

The reason for multiplying the stage derivatives by $(hA)^{-1}\otimes I_N$ is to transform matrix T so that $S=T^{-1}A^{-1}T$ and $W^{[n]}=\left(T^{-1}\otimes I_N\right)Z^{[n]}$ can be introduced where S is the Jordan canonical form of A that has the same diagonal elements.

Since $Z^{[n]}=\left(T\otimes I_N\right)W^{[n]}$, the stage equation (4.9) becomes

$$G\left(W^{[n]}\right)=h^{-1}\left(S\otimes I_N\right)W^{[n]}-\left(T^{-1}\otimes I_N\right)F\left(x_{n-1}+ch,\left(T\otimes I_N\right)W^{[n]}+e\otimes y_{n-1}\right). \tag{4.11}$$

To solve for $W^{[n]}$, we need to find the Jacobian so that Newton-Raphson can be applied. Equation (4.10) becomes

$$D_G\left(W^{[n]}\right)=h^{-1}\left(S\otimes I_N\right)-\left(T^{-1}\otimes I_N\right)J\left(x_{n-1}+ch,\left(T\otimes I_N\right)W^{[n]}+e\otimes y_{n-1}\right). \tag{4.12}$$

Finally, solving for $W^{[n]}$ by using Newton-Raphson iteration yields

$$\Delta W^{[n]}=\left(-G\left(W^{[n]}\right)\right)\left(D_G\left(W^{[n]}\right)\right)^{-1}, \tag{4.13}$$

where,

$$\Delta W^{[n]}=W^{[n+1]}-W^{[n]}.$$

The update y_n as given in (3.6) is therefore given by

$$y_n=y_{n-1}+b^TA^{-1}\left(T\otimes I_N\right)W^{[n]}. \tag{4.14}$$

CHAPTER 5

NUMERICAL EXPERIMENTS

In this chapter, the numerical results on the efficiency of 2-stage (G2) and 3-stage (G3) Gauss methods using implementation strategies by Hairer and Wanner (1999) and González-Pinto et al. (1994, 1995) were discussed for solving real life stiff problems. The schemes were implemented by using variable stepsize setting with simplified Newton iteration. All of these numerical results are very important in determining the convergence test and to identify which implementation scheme gives efficient behaviour.

The numerical experiments has been done by using MATLAB R2019a mathematical software on HP with 2.3GHz Intel ® Core i3-7020U with RAM 8GB. All of the problems that have been tested are categorized as nonlinear problems. For each problem, the results are tested in terms of tolerance and computational (CPU) time plots. The tolerance graph is referring to how the tolerance behaves on a certain given value Tol and how does it affect the error. It also determined the accuracy of the methods for the given problems based on the researcher's implementation schemes. The tolerance

used in this numerical experiments is $\mathrm{Tol} = 10^{-7}$. The efficiency of G2 and G3 methods is measured in terms of CPU time (seconds) using *tic and toc* build-in function in MATLAB. Implementation scheme by Hairer and Wanner (1999) is denoted by HW scheme while González-Pinto et al. (1994, 1995) is denoted by GMR scheme.

5.1 Real Life Problems

There are six problems that has been investigated such as Robertson, Kaps, Brusselator, Oreganator, Van der Pol and HIRES problems. All of the problems are classified as stiff nonlinear problems, thus all of these problems consumed more time for the computations. The schemes is compared based on three different implementations, which denoted by GMR scheme for González-Pinto et al.(1994,1995), HW scheme for Hairer and Wanner (1999) and the last one denoted by MHW scheme which refer to modified HW scheme. The difference between HW and MHW scheme is that no transformation such that $(hA)^{-1} \otimes I_N$, $S = T^{-1}A^{-1}T$ and $W^{[n]} = \left(T^{-1} \otimes I_N\right) Z^{[n]}$ is applied to MHW scheme. The HW scheme is specially designed for the 3-stage RADAU method and this scheme has been proven to give a robust implementation. As mentioned in the previous chapter, the GMR scheme is a modification from Cooper and Butcher (1983) implementation scheme. Their scheme is proven to give a convergent behaviour for linear and constant coefficient problems and also very efficient for general problems. Since the nonlinear stiff problems has been not investigated in details, thus the GMR scheme is implemented in solving the nonlinear stiff problems for G2 and G3 methods.

For MHW scheme, it is quite similar with HW scheme, however the scheme is not involving the coefficient matrix T for the implementation.

5.1.1 Robertson Problem

The Robertson problem is a chemical reaction problem proposed by Robertson (1966) that describes the kinetics of an autocatalytic reaction. It was known as ROBER problem and consists of a stiff system of three nonlinear ODEs (Hairer & Wanner, 1996). The problem can be written in the following form

$$\frac{dy}{dt} = f(y), \quad y(0) = y_0,$$

with

$$y \in \mathbb{R}^3, \quad t \in [0, T].$$

The function f can also be written in a system as given by

$$\begin{aligned} y_1' &= -0.04y_1 + 10^4 y_2 y_3 & y_1(0) &= 1, \\ y_2' &= 0.04y_1 - 10^4 y_2 y_3 - 3 \cdot 10^7 {y_2}^2 & y_2(0) &= 0, \\ y_3' &= 3 \cdot 10^7 {y_2}^2 & y_3(0) &= 0. \end{aligned} \tag{5.1}$$

Table 5.1 shows the structure of the reactions, where k_1, k_2, k_3 are the rate constants and A, B and C are referring to the chemical species involved.

Table 5.1

Reaction scheme for problem ROBER

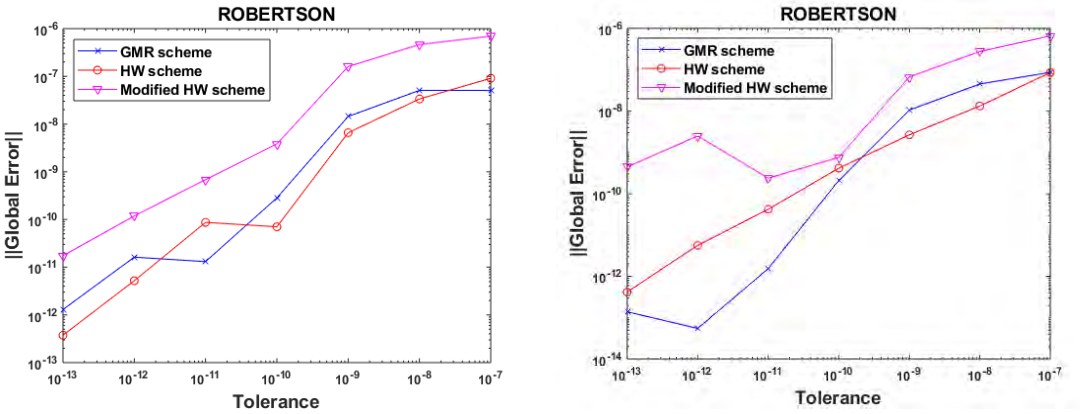
1.	A	$\xrightarrow{k_1}$	B
2.	$B+B$	$\xrightarrow{k_2}$	$C+B$
3.	$B+C$	$\xrightarrow{k_3}$	$A+C$

Aiken (1985) describes some idealized conditions and the expectation that it is involving rate functions and the mass action law is applied to it. The mathematical odel of ROBER problem consists of a set of three ODEs and can be shown by

$$\begin{pmatrix} y_1' \\ y_2' \\ y_3' \end{pmatrix} = \begin{pmatrix} -k_1 y_1 + k_3 y_2 y_3 \\ k_1 y_1 - k_2 y_2^{\,2} - k_3 y_2 y_3 \\ k_2 y_2^{\,2} \end{pmatrix}, \tag{5.2}$$

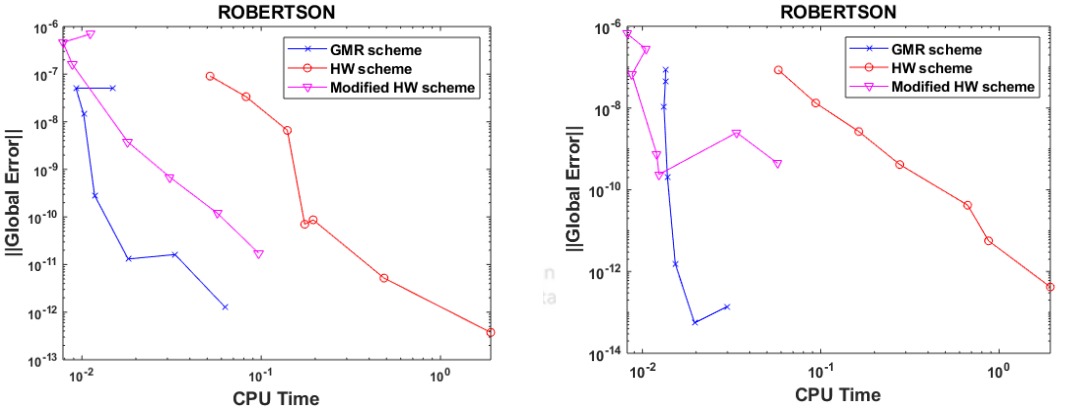
with $\left(y_1(0), y_2(0), y_3(0)\right)^T = \left(y_{01}, y_{02}, y_{03}\right)^T$ where the coefficients y_1, y_2, y_3 are the concentrations of A, B and C respectively, while y_{01}, y_{02}, y_{03} are the concentrations for which the time $t = 0$. Since past decade, the ROBER problem became very popular among mathematicians for the numerical studies and is favorable to be used as a test problem for the solution of stiff systems. Originally, the problem was posed on the time interval $0 \le t \le 40$, but it is reasonable to integrate on much longer intervals in determining their stability and efficiency. However, Hindmarsh (1980) discovered that many codes fail if the problem is integrated at a longer computational time t.

For this numerical experiments, the problem is integrated to $x_n = 10$ with stepsize $h = 0.01$. The numerical result for G2 and G3 methods using Robertson problem is given in Figures 5.1 – 5.3.



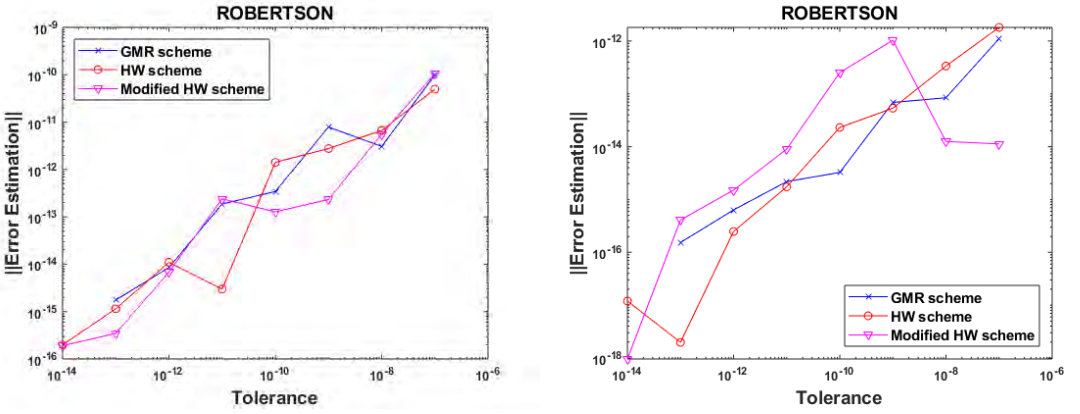
a) G2 method b) G3 method

Figure 5.1. Global error versus tolerance of (a) G2 and (b) G3 methods for Robertson problem.



a) G2 method b) G3 method

Figure 5.2. Global error versus CPU time of (a) G2 and (b) G3 methods for Robertson problem.



a) G2 method b) G3 method

Figure 5.3. Error estimation by using extrapolation versus tolerance of (a) G2 and (b) G3 methods for Robertson problem.

Figure 5.1 and Figure 5.2 showed two plots which are the loglog absolute global error versus loglog tolerance plot and loglog global error versus CPU time plot. In Figure 5.1 (a), as the tolerance gets stringent, we can see that the global error for HW scheme of G2 method decreases and thus gives the smallest errors. However, the HW scheme requires more computational time than GMR and the MHW schemes. Furthermore, the GMR scheme as shown in Figure 5.2 (a) is being chosen as the most efficient scheme in solving Robertson problem for G2 method. For the G3 method, it has been proven that GMR scheme gives the smallest error among the others as the tolerances get stringent (refer Figure 5.1(b)). The scheme also very efficient and takes shorter computational time compared to HW and MHW schemes as shown in Figure 5.2 (b).

Figure 5.3 shows the error estimation by using extrapolation versus tolerance for G2 and G3 methods. For G2 method, it shows that the MHW and HW schemes collide to each other at the last iteration and hence give the smallest error value. However we intended to choose the MHW scheme as the scheme that gives the best error estimation. Same goes to G3 method as shown Figure 5.3 (b), the MHW scheme is proven to give the best error estimation as the tolerance become stringent. For both G2 and G3 methods, the error estimation obtained is not that significant. This behaviour shows than the local extrapolation does not effected the implementation scheme.

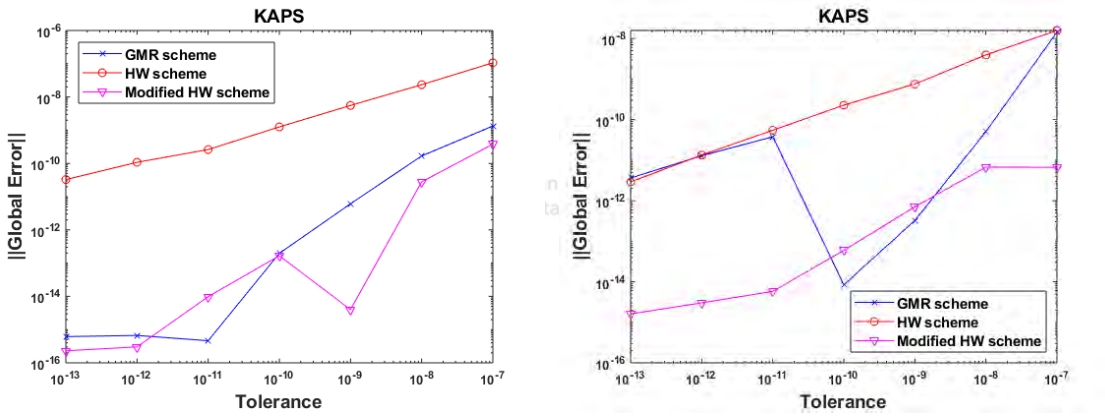
5.1.2 Kaps Problem

The Kaps problem is used to investigate the decreased order phenomenon (Dekker & Verwer, 1984). The exact solution of this problem is $y_1(x) = e^{-2x}$ and $y_2(x) = e^{-x}$. In

an article written by Kennedy and Carpenter (2019), the exact solution of y_1 has been modified which is $y_1 = y_2^2$ where it is referring to emergent (algebraic variable). This problem consist of stiffness parameter q and this is a two-dimensional nonlinear test problem which given by

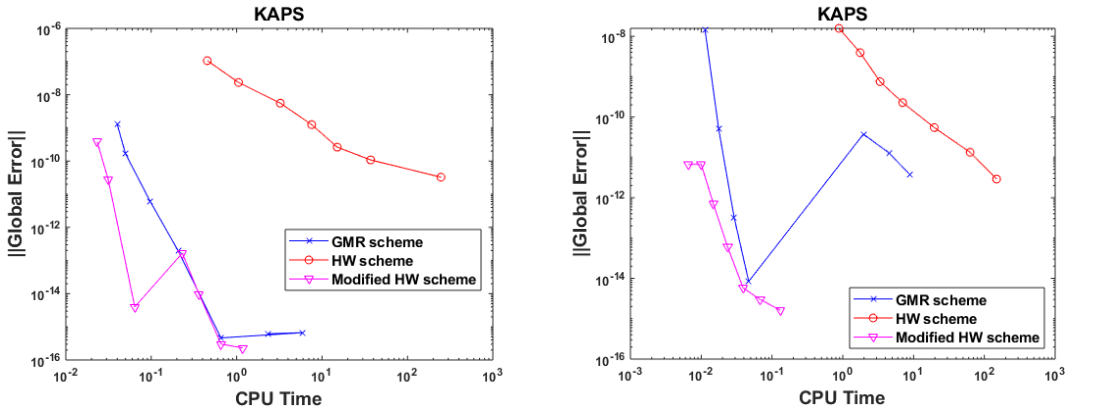
$$\begin{aligned} y_1' &= (q-2)y_1 - q y_2^2, & y_1(0) &= 1, \\ y_2' &= y_1 - y_2 - y_2^2, & y_2(0) &= 1. \end{aligned} \tag{5.2}$$

The problem is integrated to $x_n = 5$, stepsize $h = 0.01$ and constant stiff value $q = -10000$. The numerical result for Kaps problem is given in Figures 5.4 – 5.6.



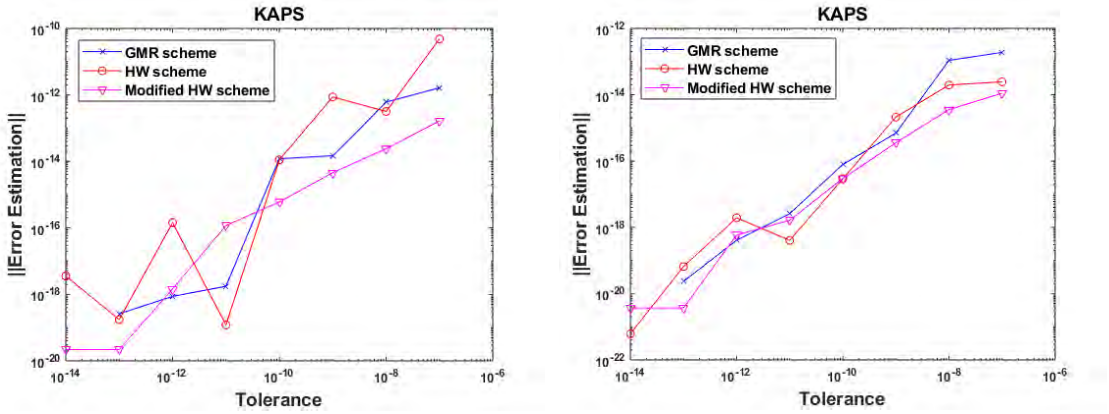
a) G2 method b) G3 method

Figure 5.4. Global error versus tolerance of (a) G2 and (b) G3 methods for Kaps problem.



a) G2 method b) G3 method

Figure 5.5. Global error versus CPU time of (a) G2 and (b) G3 methods for Kaps problem.



a) G2 method b) G3 method

Figure 5.6. Error estimation by using extrapolation versus tolerance of (a) G2 and (b) G3 methods for Kaps problem.

For the Kaps problem as given in Figure 5.4, the error for the MHW scheme for G2 and G3 methods gives the smallest error as the tolerance gets stringent. The MHW scheme also giving the most efficiency behaviour in solving this two dimensional nonlinear test problem as shown in Figure 5.5 (a) and (b) and it takes shorter computational time compared to GMR and HW schemes. For the approximate tolerance value of $\mathrm{Tol} = 10^{-10}$ as shown in Figure 5.4 (b), the GMR scheme giving the least global error than MHW scheme. However, the solution of GMR scheme is fluctuated significantly as the tolerances become stringent meanwhile the global error of MHW scheme is decreasing as the tolerance get stringent. Thus it can be concluded that the MHW scheme is the most efficient implementation strategies in solving the Kaps problem for G2 and G3 methods.

In Figure 5.6 (a), the MHW scheme also gives an efficient results for G2 method where the error estimation by using extrapolation is smaller than the others. However, for G3 method as shown in Figure 5.6 (b), it can be seen that all of the schemes is giving almost similar solutions as the tolerance get stringent. Furthermore, it can be concluded

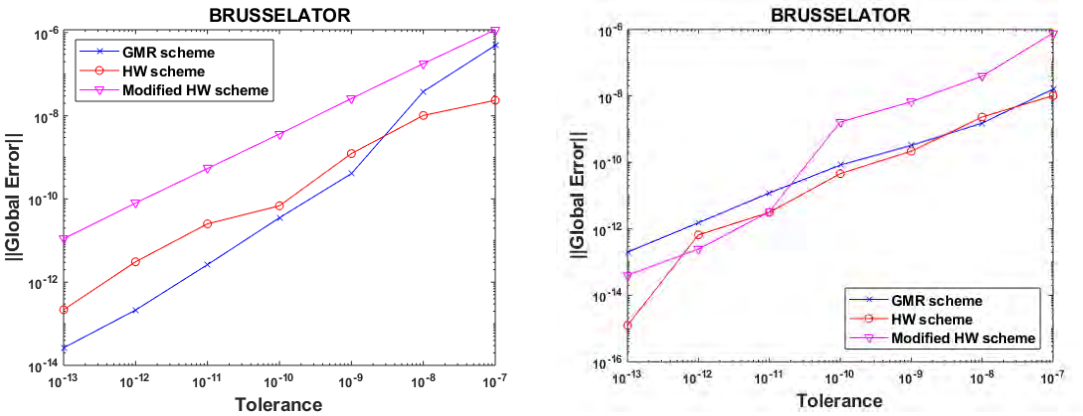
that the error estimation by using extrapolation does not effected the implementation strategies.

5.1.3 Brusselator Problem

The Brusselator is a theoretical model of a single chemical reaction or it is known as autocatalytic reaction that was proposed by physical chemist, Ilya Prigogine and his collaborators at the Free University of Brussels (Hairer & Wanner, 1996). The problem is defined by the following equations:

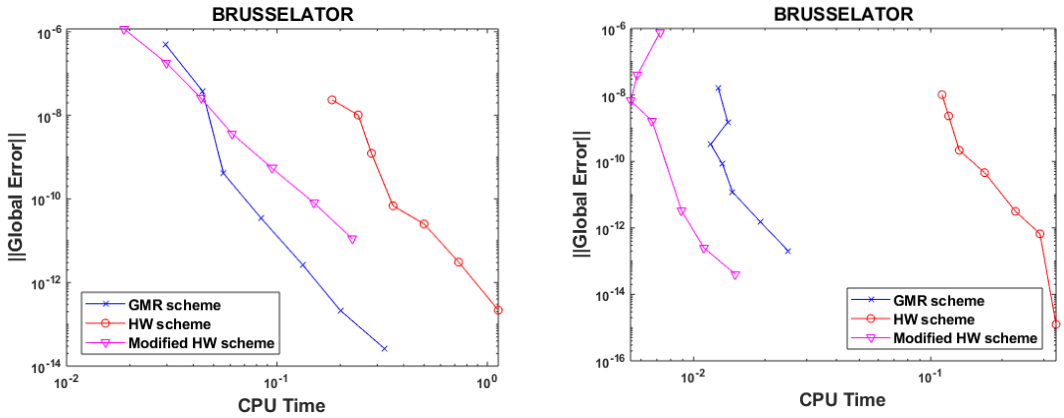
$$\begin{aligned} y_1' &= 1 + y_1^2 y_2 - 4y_1, & y_1(0) &= 1.5, \\ y_2' &= 3y_1 - y_1^2 y_2, & y_2(0) &= 3. \end{aligned} \tag{5.3}$$

The problem is integrated to $x_n = 10$ and stepsize $h = 0.01$. The numerical result for Brusselator problem is given in Figures 5.7 – 5.9.



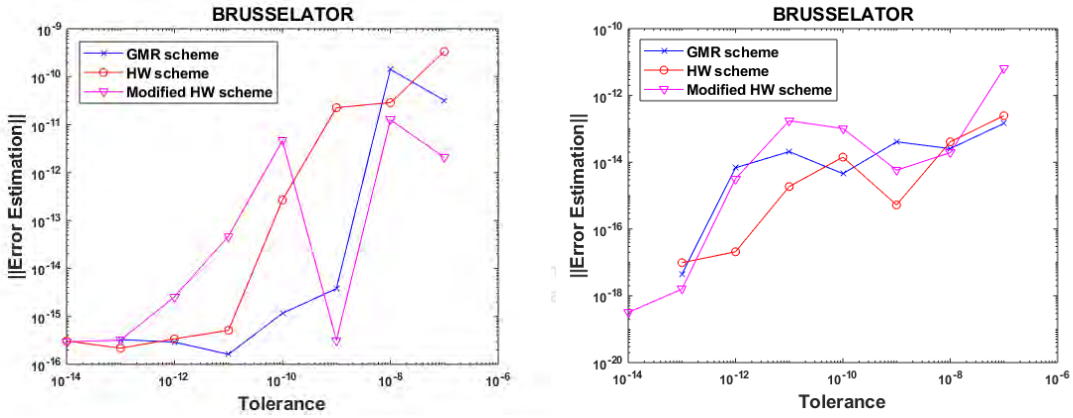
a) G2 method b) G3 method

Figure 5.7. Global error versus tolerance of (a) G2 and (b) G3 methods for Brusselator problem.



a) G2 method b) G3 method

Figure 5.8. Global error versus CPU time of (a) G2 and (b) G3 methods for Brusselator problem.



a) G2 method b) G3 method

Figure 5.9. Error estimation by using extrapolation versus tolerance of (a) G2 and (b) G3 methods for Brusselator problem.

Referring to Figure 5.7 (a), it can be shown that the scheme that gives the least error in solving Brusselator problem for G2 method is the GMR scheme followed by HW and MHW schemes. GMR scheme is said to give efficient numerical results among the others as given in Figure 5.8 (a), therefore suitable in solving stiff problem. For the 3-stage Gauss method as shown in Figure 5.7 (b), the HW scheme is giving the least global error as the tolerance get stringent, however the scheme need more computational time to solve the Brusselator problem as shown in Figure 5.8 (b). Hence, the most efficient implementation scheme is proved by the MHW scheme as shown in

Figure 5.8 (b) where it takes shorter computational time since there is a need in numerical analysis to choose the implementation strategies that gives shorter computational time in obtaining good numerical results.

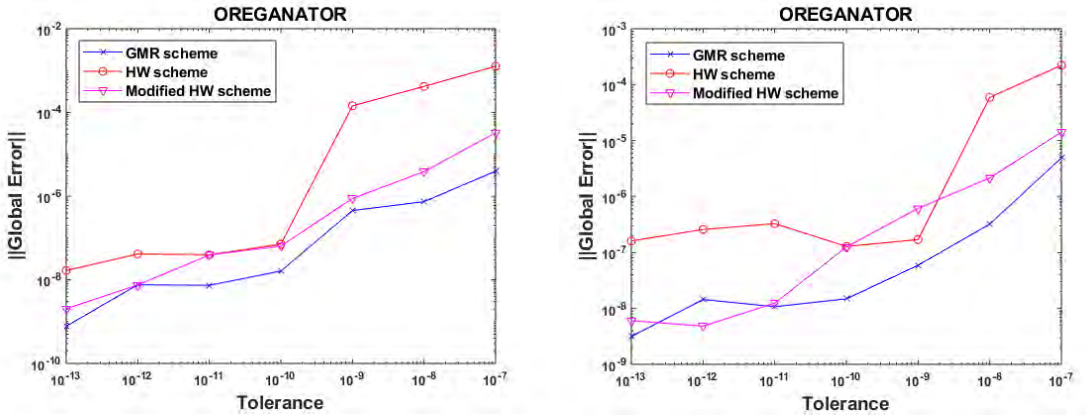
Figure 5.9 shows the numerical results for G2 and G3 methods regarding the error estimation that implemented using extrapolation technique. For G2 and G3 methods, it turns out that all schemes giving almost similar error at the last iteration. Thus, it is difficult to choose which implementation schemes is giving the best error estimation. However, it can be observed that the MHW scheme in Figure 5.9 (a) is destroyed by the round-off error and hence, it is not recommended to be used in solving Brusselator problem for G2 method.

5.1.4 Oreganator Problem

The Oreganator is one of the famous model with a periodic solution that was proposed for the Belusov-Zhabotinskii reaction. It is one of the example of non-equilibrium thermodynamics that categorized as nonlinear chemical oscillator of stiff problem with three dimensions (Hairer & Wanner, 1996). The equations of the problem is given by

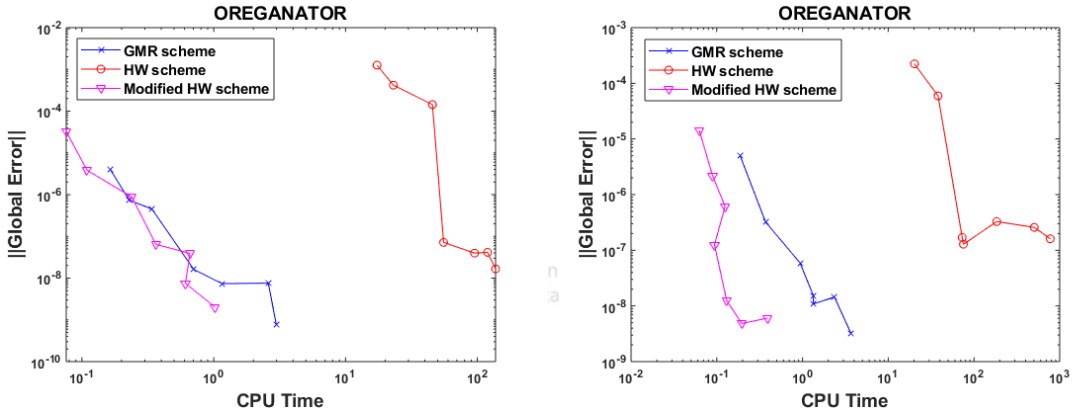
$$\begin{aligned} y_1' &= 77.27\left(y_2 + y_1\left(1 - 8.375\times10^{-6} y_1 - y_2\right)\right) \\ y_2' &= \frac{1}{77.27}\left(y_3 - \left(1 + y_1\right) y_2\right) \\ y_3' &= 0.161\left(y_1 - y_3\right) \end{aligned} \qquad \begin{aligned} y_1(0) &= 1, \\ y_2(0) &= 2, \\ y_3(0) &= 3. \end{aligned} \tag{5.4}$$

The problem is integrated to $x_n = 30$ and stepsize $h = 0.01$. The numerical result for Oreganator problem are given in Figures 5.10 – 5.12.



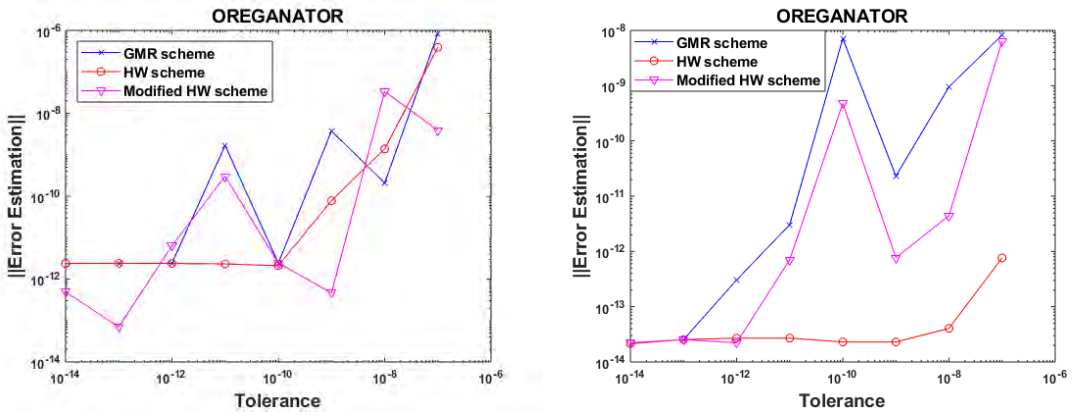
a) G2 method b) G3 method

Figure 5.10. Global error versus tolerance of (a) G2 and (b) G3 methods for Oreganator problem.



a) G2 method b) G3 method

Figure 5.11. Global error versus CPU time of (a) G2 and (b) G3 methods for Oreganator problem.



a) G2 method b) G3 method

Figure 5.12. Error estimation by using extrapolation versus tolerance of (a) G2 and (b) G3 methods for Oreganator problem.

Figures 5.10 – 5.12 shows the numerical results obtained for Oreganator problem. From the numerical results obtained, the G2 and G3 methods as shown in Figure 5.10 is giving the least global error as the tolerance get stringent which implemented by GMR scheme followed by MHW and HW schemes. However, the MHW scheme as shown in Figure 5.11 gives the most efficient implementation strategies by G2 and G3 methods and thus suitable in solving Oreganator problem.

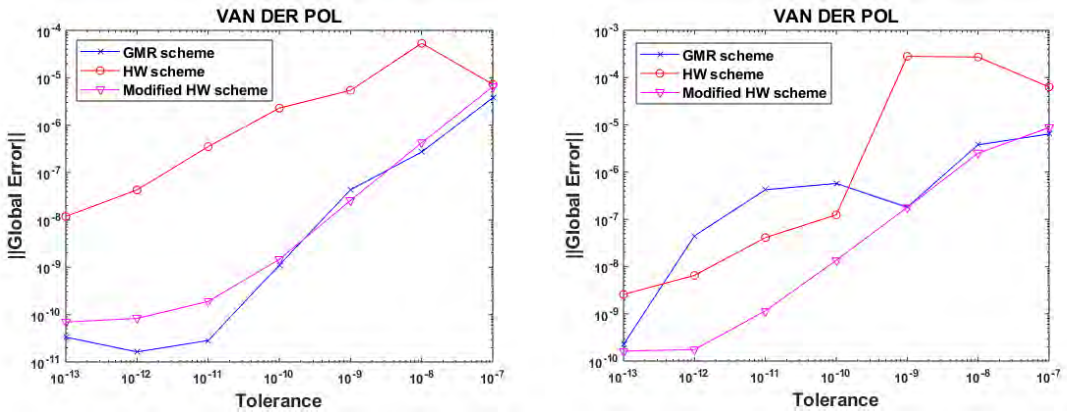
In Figure 5.12, the error estimation that being investigated are not giving an excellent behaviour. This is because the results obtained shows that the schemes is fluctuated significantly and destroyed by the round-off errors for both G2 and G3 methods especially for GMR and MHW schemes. However, it can be seen that the HW scheme is giving the best error estimation for both G2 and G3 methods as the tolerance get stringent.

5.1.5 Van der Pol Problem

The Van der Pol oscillator was originally proposed by the well-known physicist, Balthasar Van der Pol while he was working at Philips, Amsterdam which is one of the largest electronics companies in the world. He found a stable oscillations or it is called as relaxation-oscillations which are now known as a type of limit cycle in electrical circuits employing vacuum tubes. The Van der Pol equation has being used in both the physical and biological sciences. The equation consists of stiffness parameter ε. The problem is defined by

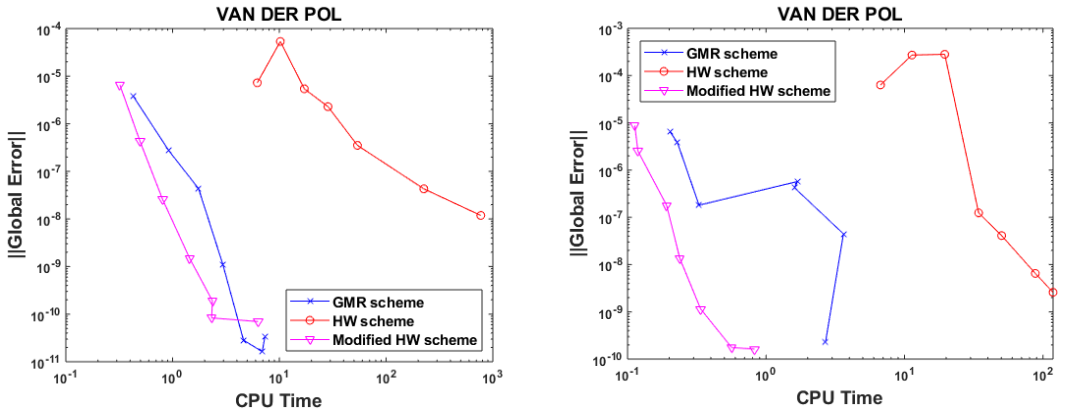
$$\begin{aligned} y_1' &= y_2 & y_1(0) &= 2, \\ y_2' &= \frac{1}{\varepsilon}\left(\left(1-y_1^{\,2}\right)y_2 - y_1\right) & y_2(0) &= 0. \end{aligned} \tag{5.5}$$

The problem is integrated to $x_n = 5$, stepsize $h = 0.01$ and $\varepsilon = 10^{-3}$. The numerical results for Van der Pol problem are given in Figures 5.13 – 5.15.



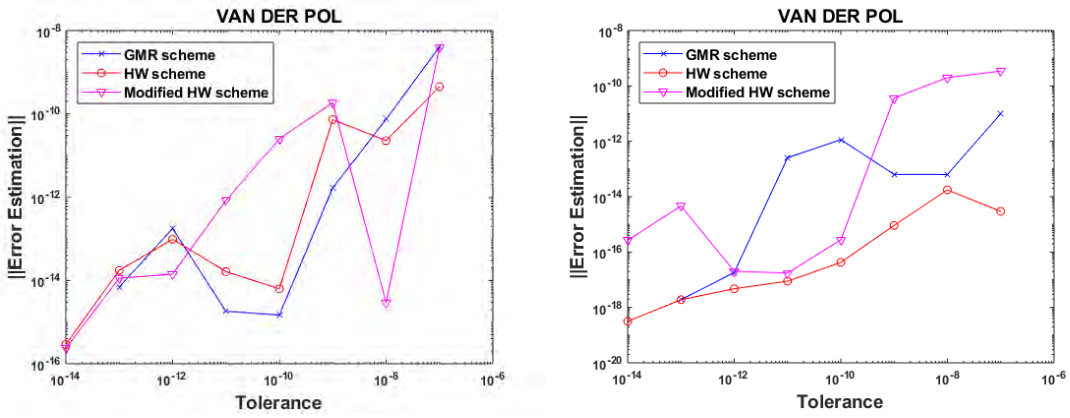
a) G2 method b) G3 method

Figure 5.13. Global error versus tolerance of (a) G2 and (b) G3 methods for Van der Pol problem.



a) G2 method b) G3 method

Figure 5.14. Global error versus CPU time of (a) G2 and (b) G3 methods for Van der Pol problem.



a) G2 method b) G3 method

Figure 5.15. Error estimation by using extrapolation versus tolerance of (a) G2 and (b) G3 methods for Van der Pol problem.

Figures 5.13 – 5.15 shows the numerical results of G2 and G3 methods for Van der Pol problem in terms of tolerance, CPU time and error estimation by extrapolation. In Figure 5.13, it can be observed that GMR scheme for G2 method gives the least global error followed by MHW and HW schemes, therefore it can be concluded that the scheme gives better accuracy for the solution of nonlinear stiff problem. The GMR scheme also giving the most efficient implementation strategies in solving Van der Pol problem for the solutions of G2 method since it takes shorter computational time compared to the others as shown in Figure 5.14 (a). For G3 method as shown in Figure 5.13 (b), as the tolerances become stringent the MHW and GMR schemes are observed to give almost similar error where both of it are satisfying the efficiency behaviour. Nevertheless, among these two schemes we intended to conclude that the MHW scheme is the most efficient implementation strategies in solving Van der Pol problem by the G3 method as it gives shorter computation time (refer Figure 5.14 (b)).

In Figure 5.15, both figures (a) and (b) shows the error estimation that estimated using extrapolation technique. It can be observed that the error estimation obtained by

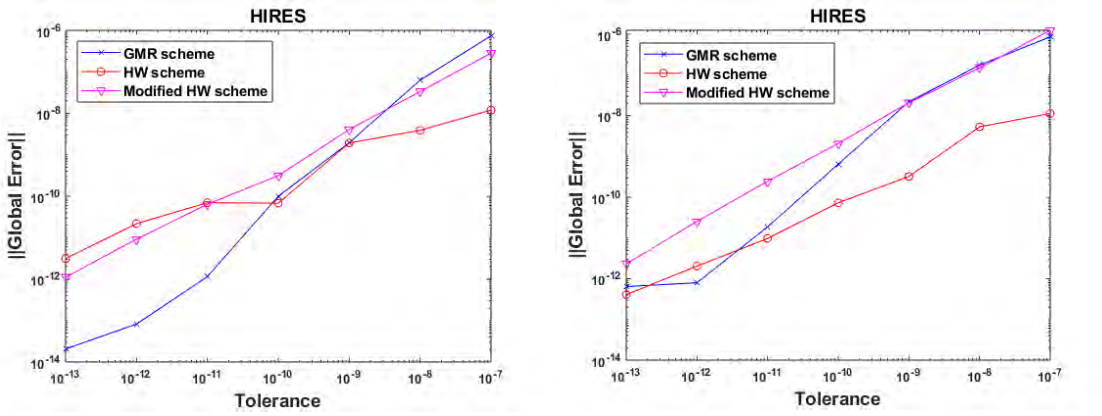
G2 method is giving the best error estimation for GMR and HW schemes and both of it are suitable for the solution of stiff problems using variable stepsize setting where the schemes satisfy the convergence property. The MHW scheme as shown in Figure 5.15 (a) is fluctuated significantly and got destroyed by the round-off errors along the iteration. However, the HW scheme is shown to give the best error estimation for G3 method where it generated the least error estimation as the tolerance become stringent.

5.1.6 HIRES Problem

Schäfer (1975) proposed a HIRES problem and defined it as a reaction of 8 reactants. The studied is about the photomorphogenesis of a plant that used a high-frequency-controlled light source to grow a plant. The word HIRES was originally stand for 'High Irradiance RESponse' where the mathematical model of ODEs was given by Hairer and Wanner (1996). A further explanation of HIRES can be found in Swart, Jacques and Lioen (1998). HIRES is a nonlinear system of 8 dimensions and categorized as a moderately stiff problem. The problem is of the following form

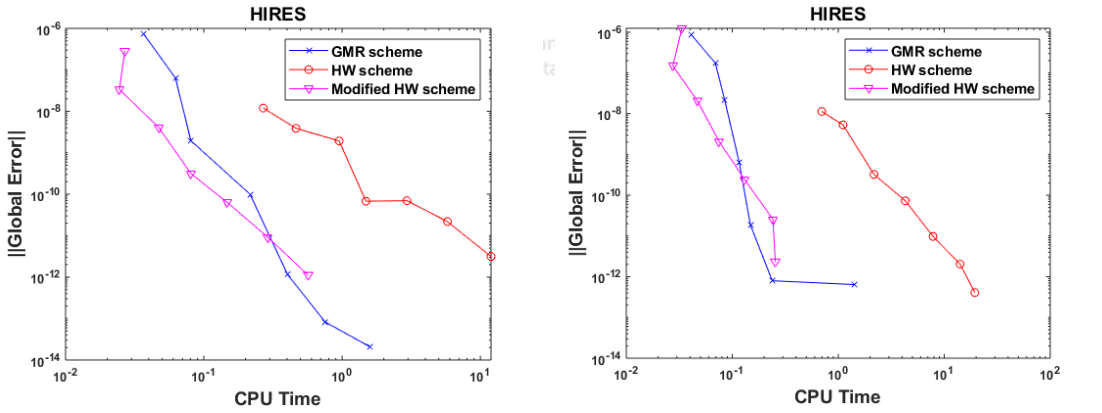
$$
\begin{aligned}
y_1' &= -1.71y_1 + 0.43y_2 + 8.32y_3 + 0.0007, & y_1(0) &= 1,\\
y_2' &= 1.71y_1 - 8.75y_2, & y_2(0) &= 0,\\
y_3' &= -10.03y_3 + 0.43y_4 + 0.035y_5, & y_3(0) &= 0,\\
y_4' &= 8.32y_2 + 1.71y_3 - 1.12y_4, & y_4(0) &= 0,\\
y_5' &= -1.745y_5 + 0.43y_6 + 0.43y_7, & y_5(0) &= 0,\\
y_6' &= -280y_6y_8 + 0.69y_4 + 1.71y_5 - 0.43y_6 + 0.69y_7, & y_6(0) &= 0,\\
y_7' &= 280y_6y_8 - 1.81y_7, & y_7(0) &= 0,\\
y_8' &= -280y_6y_8 + 1.87y_7, & y_8(0) &= 0.0057.
\end{aligned}
\tag{5.6}
$$

The problem is integrated to $x_n = 321.8122$ and stepsize $h = 0.01$. The numerical results for HIRES problem are given in Figures 5.16 – 5.18.



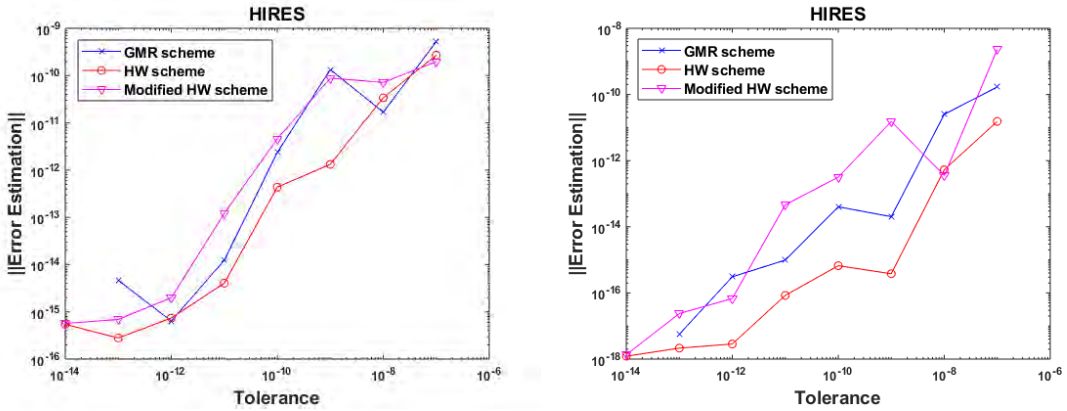
a) G2 method b) G3 method

Figure 5.16. Global error versus tolerance of (a) G2 and (b) G3 methods for HIRES problem.



a) G2 method b) G3 method

Figure 5.17. Global error versus CPU time of (a) G2 and (b) G3 methods for HIRES problem.



a) G2 method b) G3 method

Figure 5.18. Error estimation by using extrapolation versus tolerance of (a) G2 and (b) G3 methods for HIRES problem.

Figure 5.16 shows the numerical results for HIRES problem by G2 and G3 methods. It can be observed that the GMR scheme gives the lowest global error as the tolerance become stringent for G2 method, whereas the HW scheme is giving the lowest global error for G3 method. In terms of CPU time as shown in Figure 5.17, the MHW scheme is the most efficient implementation strategies for both G2 and G3 methods where the scheme is giving less computational time among the others.

Lastly, the numerical results for G2 and G3 methods are presented in terms of error estimation that computed by using extrapolation technique (refer Figure 5.18). The HW and MHW schemes are observed to give almost similar error estimation for G2 and G3 methods towards the end of the iteration. However, we intended to choose the HW scheme as the scheme that gives the best error estimation for both G2 and G3 methods in solving HIRES problem using variable stepsize setting.

5.2 Summary on Numerical Results

Based on all the numerical results, it can be summarized that different problems gives different numerical approximations depends on their stiffness ratio using different implementation strategies. Apparently, the scheme proposed by Hairer and Wanner (1999) or denoted as HW scheme gives the weaker yet less efficiency among the others for the solution of stiff problems using variable stepsize setting. It can be concluded that HW scheme is not recommended to be used with variable stepsize setting for Gauss methods. This is due to the implementation that was investigated using 2-stage and 3-stage Gauss methods. As it is concerned that even though the HW scheme is very efficient in solving Radau IIA method, thus their efficiency is now proven to give an efficient implementation limited to this method only, and thus not efficient when implemented with other IRK methods such as 2-stage and 3-stage Gauss methods. It might be a reason of the differences in eigenvalues where Radau IIA method only have a single eigenvalue whereas the Gauss methods have a complex eigenvalue. Other than that, Radau IIA method also satisfy the properties of A-stable and L-stable whereas Gauss method only satisfied the A-stable property. The L-stable property only can be found in Radau IIA method which contributes to the extra advantages. Furthermore, it can be summarized that the MHW scheme is as efficient as GMR and HW schemes when implemented with extrapolation technique even though the scheme is without any transformation matrix T. Table 5.2 shows the summary of the numerical results that was obtained for the solution of G2 and G3 methods.

Table 5.2

The most efficient implementation of real life stiff problems

Problems	Global error versus tolerance		Global error versus CPU time		Error estimation (extrapolation) versus tolerance	
	G2	G3	G2	G3	G2	G3
Robertson	HW	GMR	GMR	GMR	MHW	MHW
Kaps	MHW	MHW	MHW	MHW	MHW	GMR HW MHW
Brusselator	GMR	HW	GMR	MHW	GMR HW	GMR HW MHW
Oreganator	GMR	GMR	MHW	MHW	HW	HW
Van der Pol	GMR	MHW	GMR	MHW	GMR HW	HW
HIRES	GMR	HW	MHW	MHW	HW	HW

The summary on numerical results that was presented in Table 5.2 can also be illustrated in terms of the error values for each scheme as shown in Table 5.3 and Table 5.4.

Table 5.3

Error values for each scheme in terms of global error versus tolerance $\left(\text{Tol} = 10^{-13}\right)$

Problems	Error values for each scheme (G2 method)		
	GMR	HW	MHW
Robertson	1.29314433252579e-12	3.74256612837401e-13	1.7390072683974e-11
Kaps	6.15770636140043e-16	3.2662438383076e-11	2.30607329869179e-16
Brusselator	2.63775533058874e-14	2.17300796281834e-13	1.10792004254562e-11
Oreganator	7.75043188081224e-10	1.6635073305158e-08	2.03212543216419e-09
Van der Pol	3.33713731538484e-11	1.18690109433547e-08	6.97397387928324e-11
HIRES	2.05399986098905e-14	3.12527885208792e-12	1.10873918922055e-12

Problems	Error values for each scheme (G3 method)		
	GMR	HW	MHW
Robertson	1.39703239165766e-13	4.19109717497298e-13	4.44148181970094e-10
Kaps	3.63261375116538e-12	2.90037660036354e-12	1.61425130908426e-15
Brusselator	2.0222790867847e-13	1.25607396694702e-15	4.01943669423046e-14
Oreganator	3.14428090124009e-09	1.60922280152946e-07	6.04281640138125e-09
Van der Pol	2.25200245459964e-10	2.58545366667904e-09	1.62577421763854e-10
HIRES	6.42259813206665e-13	4.07593549004841e-13	2.29913913254419e-12

Table 5.4

Error values for each scheme in terms of error estimation by using extrapolation versus tolerance $\left(\text{Tol} = 10^{-13}\right)$

Problems	Error estimation values for each scheme (G2 method)		
	GMR	HW	MHW
Robertson	1.75203266981614e-15	1.14736773620886e-15	3.4656060859046e-16
Kaps	2.61682076447296e-19	1.74454717631531e-19	2.18068397039413e-20
Brusselator	3.2937050688833e-16	2.1771948760415e-16	3.20996680442016e-16
Oreganator	2.40844495663904e-12	2.40844495663904e-12	7.00129347860186e-14
Van der Pol	6.97818870526122e-15	1.74957147218309e-14	1.13716563140937e-14
HIRES	4.64167305834273e-15	2.7952007132512e-16	6.92498001638361e-16

Problems	Error estimation values for each scheme (G3 method)		
	GMR	HW	MHW
Robertson	1.53367054264227e-16	1.97431921063568e-18	4.10416148056316e-16
Kaps	2.34880926516375e-20	6.55194163440414e-20	3.70864620815329e-21
Brusselator	4.43059600334046e-18	9.81060543596816e-18	1.58235571547874e-18
Oreganator	2.54015182897119e-14	2.54015182897119e-14	2.54015182897119e-14
Van der Pol	1.89882685857448e-18	1.89882685857448e-18	4.61984574691171e-15
HIRES	5.63961466719843e-18	2.14606993911804e-18	2.38595753801542e-17

The error values that highlighted in red colour as shown in Table 5.3 and Table 5.4 indicates the lowest error obtained among the schemes for certain given tolerance, $\text{Tol} = 10^{-13}$. For the problem that highlighted more than one red colour indicates the same error values obtained by the schemes.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

The main objectives of this thesis is to study the implementation schemes by González-Pinto et al. (1994, 1995) and Hairer and Wanner (1999) by using variable stepsize setting that involving 2-stage (G2) and 3-stage (G3) Gauss methods. To know either the implementation is correct or wrong, the first stage is to implement the schemes to Radau IIA method of order 3. This method is chosen because it is proven to give a robust implementation when implemented with Hairer and Wanner (1999) implementation scheme using variable stepsize setting as mentioned previously. It is considered giving a correct implementation if the numerical approximations satisfies the convergence test and efficiency behaviour. Thus, the Matlab code is then being implemented using G2 and G3 Gauss methods.

Literature review suggested GMR scheme is constructed for the families of Gauss methods while HW scheme is constructed for Radau IIA method. However,

based on this research, it is shown that the standard implementation scheme with some tuning using HW scheme known as modified HW (MHW) scheme that does not involve any transformation matrix T can be as efficient as the HW and GMR schemes. Therefore, this thesis proved that the transformation matrix T is not necessary and although they can cause cheaper implementation, however the computational time is marginally adjustable. Hence, the conclusions of the research are done by answering the three objectives of this research.

6.1.1 Implementation Ideas

As mentioned previously, Hairer and Wanner (1999) implementation scheme is specially designed for the 3-stage Radau method and it has been proven to give robustness and satisfy the efficiency properties. For this research, an investigation regarding researcher's scheme has been investigated by using different implicit Runge-Kutta (IRK) methods which are 2-stage (G2) and 3-stage (G3) Gauss methods. In the research that was investigated by González-Pinto et al. (1994, 1995), it has been proven that the Gauss methods gives the least error than the diagonally-implicit Runge-Kutta (DIRK) methods. In addition to that, they also found out that the performance of 3-stage Radau method is poorer than the Gauss methods probably due to the one order less. In other words, Gauss methods are particularly suitable for solving stiff systems because they have higher order of convergence and good stability properties. This properties has also been proven in the numerical results that was obtained for this research. The real life problems that were investigated are the Robertson, Kaps, Brusselator, Oreganator, Van der Pol and HIRES problems as explained in detailed in previous chapter. Based

on the numerical results obtained, it can be summarized that all the scheme involved which denote by GMR scheme for González-Pinto et al. (1994, 1995), HW scheme for Hairer and Wanner (1999) and the last one denote by modified HW (MHW) scheme are suitable in solving certain real life problems by the G2 and G3 methods.

6.1.2 Best Error Estimation

The extrapolation technique has been implemented throughout the research to estimate the best error estimation among the researcher's schemes using six real life stiff problems as described previously. From the numerical approximations obtained, it is proven that the GMR scheme by using G2 method is giving the best error estimation for Brusselator and Van der Pol problems, whereas for G3 method it is giving the best error estimation for Kaps and Brusselator problems. For HW scheme, the G2 method is giving the best error estimation for Brusselator, Oreganator, Van der Pol and HIRES problems. The HW scheme by using G3 method is also giving the best error estimation for all the problems except for the Robertson problem. The comparison also being compared with the so-called MHW scheme without using any transformation matrix T and it has been proven that the scheme is giving the best error estimation in solving Robertson and Kaps problems by using G2 method meanwhile for the G3 method, the Robertson, Kaps and Brusselator problems also giving the best error estimation. From the numerical results obtained, it can be summarized that among these three schemes, the HW scheme is giving the best error estimation because the scheme can solve almost all real life stiff problems involved in this research.

6.1.3 Most Efficient Implementation Strategies for Gauss Methods

In deciding the most efficient implementation strategy for Gauss methods, a comparison has been made between the schemes by González-Pinto et al. (1994, 1995) which is denoted by GMR scheme, and the two types of Hairer and Wanner (1999) implementation schemes which are denoted by HW and MHW schemes. The difference between HW and MHW schemes are mentioned in previous chapter.

Based on all the numerical results obtained, it can be summarized that the GMR scheme gives efficient implementation in solving Robertson, Brusselator and Van der Pol problems using variable stepsize setting by the G2 method whereas for G3 method, only Robertson problem gives the most efficiency behaviour. GMR scheme not only satisfies the requirement of high accuracy and high efficiency in solving stiff problems but also has lower computational cost. This clearly proved that the Robertson problem is the most efficient stiff problem which implemented with GMR scheme because the problem is satisfy the efficiency properties for both of G2 and G3 methods.

As mentioned previously, the HW scheme is specially designed for 3-stage Radau IIA method and it has been proven to give a robust implementation. However, for this research which implemented by using G2 and G3 methods, the scheme is not giving good efficiency behaviour among the problems involved. None of the problems involved are showing good efficiency behaviour. The reason might be because of the differences in the eigenvalue involved since the 3-stage Radau IIA method is having a single eigenvalue while the family of Gauss methods is consisting of complex eigenvalue. Besides, the difference in stability behaviour might affected the numerical

analysis. Generally, the Radau IIA method is satisfy the property of *A*-stable and *L*-stable while for the family of Gauss methods, it is only satisfy the property of *A*-stable. *L*-stable is an extra advantages that only can be found in the family of Radau methods.

The scheme is also being compared with the so called MHW scheme. Even though the MHW scheme is without using any transformation matrix *T*, however the scheme is proven to give efficient implementation for the solution of G2 and G3 methods and thus suitable in solving stiff problems using variable stepsize setting. For the solution of G2 method, the scheme is efficient in solving Kaps, Oreganator and HIRES problems while for the G3 method, it is efficient in solving all real life problems except for the Robertson. This behaviour obviously shows that even though the MHW scheme is without using any transformation matrix *T*, the scheme is as efficient as GMR and HW schemes. This research therefore recommended the use of MHW scheme in solving stiff problems by the implicit Gauss methods as it is shown from all the numerical experiments that MHW although requires a little computational time, the scheme is considered to give the most stable behaviour and works as efficient as the other two schemes.

6.2 Future Work

In this thesis, we have shown that the standard compensated summation is not a crucial components that need to be implemented when the variable stepsize setting is used even though it plays an important role in reducing the round-off errors. However, it will be

an interest to investigate in detailed the use of other compensated summation such as Kahan's compensated summation with implementation using other IRK methods. Besides, the researchers also interested to conduct the implementation schemes using higher order stages which implemented with different approach of Runge-Kutta methods as introduced by various researchers in Section 2.3. This kind of approach is then can be tested on more real life problems such as linear and nonlinear problems with real and complex eigenvalues and also a few problems with nonlinear coupling which implemented using variable stepsize setting.

Furthermore, the researcher also could extend the research based on other error estimation such as symmetrization instead of extrapolation as described by Gorgey (2015) in order to determine the error estimation by using variable stepsize setting. Symmetrization is a technique that is use to dampens the oscillator behaviour caused by the 2-stage (G2) and 3-stage (G3) Gauss methods. Symmetrizers can be used to determine the error estimations for the Gauss method instead of extrapolation as it is proven in Gorgey (2015) that, symmetrizers give less error estimation than by the local extrapolation. Other than symmetrization and extrapolation, the error estimation also can be determined by using any embedded method such as splitting and composition methods as described by Blanes, Casas and Thalhammer (2019). This is the new error estimator that proposed by them. In addition to these, this research can also explore different types of problems such as delay differential equations as described by Roussel (2019) as well as in Holder and Eichholz (2019). Lastly, the research also could be extended to fuzzy differential equations as mentioned by Yu and Jafari (2019) and Hussain and Abdul-Abbas (2019).

LIST OF PUBLICATION AND CONFERENCE

Publications

Mustapha, S. S., Gorgey, A. & Imran, G. (2021). Error Estimation by using Symmetrization and Efficient Implementation Scheme for 3-stage Gauss Method. *Journal of Science and Mathematics Letters, 9,* 36-44.

Conference

International Conference on Education, Mathematics and Science 2020 (ICEMS2020) in conjunction with 8th International Postgraduate Conference on Science and Mathematics 2020 (IPCSM2020).

REFERENCES

Ababneh, O. Y. & Ahmad, R. (2009). Construction of third-order diagonal implicit Runge-Kutta methods for stiff problems. *Chinese Physics Letters, 26*(8), 080503.

Abia, L. & Sanz-Serna, J. M. (1993). Partitioned Runge-Kutta methods for separable Hamiltonian problems. *Mathematics of Computation, 60*(202), 617-634.

Agam, S. A. & Yahaya, Y. A. (2014). A highly efficient implicit Runge-Kutta method for first order ordinary differential equations. *African Journal of Mathematics and Computer Science Research, 7*(5), 55-60.

Aiken, R. C. (1985). *Stiff Computation*. Oxford: Oxford University Press.

Antoñana, M., Makazaga, J. & Murua, A. (2018). Efficient implementation of symplectic implicit Runge-Kutta schemes with simplified Newton iterations. *Numerical Algorithms, 78*(1), 63-86.

Araz, S. İ. (2020). Numerical analysis of a new Volterra integro-differential equation involving fractal-fractional operators. *Chaos, Solitons & Fractals, 130,* 109396.

Atangana, A. & Araz, S. İ. (2020). New numerical method for ordinary differential equations: Newton polynomial. *Journal of Computational and Applied Mathematics, 372,* 112622.

Atkinson, K., Han, W. & Stewart, D. E. (2009). *Numerical Solution of Ordinary Differential Equations*. New Jersey: John Wiley & Sons, Inc.

Baboulin, M., Buttari, A., Dongarra, J., Kurzak, J., Langou, J., Langou, J., et al. (2009). Accelerating scientific computations with mixed precision algorithms. *Computer Physics Communications, 180*(12), 2526-2533.

Bader, G. & Deuflhard, P. (1983). A semi-implicit mid-point rule for stiff systems of ordinary differential equations. *Numerische Mathematik, 41*(3), 373-398.

Berghe, G. V. & Daele, M. V. (2011). Symplectic exponentially-fitted four-stage Runge-Kutta methods of the Gauss type. *Numerical Algorithms, 56*(4), 591-608.

Bjurel, G., Dahlquist, G., Lindberg, B., Linde, S. & Oden, L. (1970). *Survey of stiff ordinary differential equations.* Department of Information Processing, Royal Institute of Technology, Stockholm.

Blanes, S., Casas, F. & Thalhammer, M. (2019). Splitting and composition methods with embedded error estimators. *Applied Numerical Mathematics. 146* (2019), 400-415.

Boom, P. D. & Zingg, D. W. (2015). Investigation of efficient high-order implicit Runge-Kutta methods based on generalized summation-by-parts operators. *22nd AIAA Computational Fluid Dynamics Conference, 2757,* 1-15.

Branch, M. & Mahshahr, I. R. I. (2016). Computing simulation of the generalized duffing oscillator basedon EBM and MHPM. *Mechanics and Mechanical Engineering, 20*(4), 595-604.

Burrage, K. & Butcher, J. C. (1979). Stability criteria for implicit Runge-Kutta methods. *SIAM Journal on Numerical Analysis, 16*(1), 46-57.

Butcher, J. C. (1964). Implicit Runge-Kutta processes. *Mathematics of Computation, 18*(85), 50-64.

Butcher, J. C. (1996). A history of Runge-Kutta methods. *Applied Numerical Mathematics, 20*(3), 247-260.

Butcher, J. C. (1997). An introduction to "Almost Runge-Kutta" methods. *Applied Numerical Mathematics, 24*(2-3), 331-342.

Butcher, J. C. (2016). *Numerical Methods for Ordinary Differential Equations*. United Kingdom: John Wiley & Sons.

Calvo, M., Franco, J. M., Montijano, J. I. & Rández, L. (2009). Sixth-order symmetric and symplectic exponentially fitted Runge-Kutta methods of the Gauss type. *Journal of Computational and Applied Mathematics, 223*(1), 387–398.

Cash, J. R. (1975). A class of implicit Runge-Kutta methods for the numerical integration of stiff ordinary differential equations. *Journal of the ACM (JACM), 22*(4), 504-511.

Cerrolaza, M., Shefelbine, S. & Garzón-Alvarado, D. (Eds.). (2018). *Numerical Methods and Advanced Simulation in Biomechanics and Biological Processes.* London, UK: Academic Press.

Chan, R. P. K. (1990). On symmetric Runge-Kutta methods of high order. *Computing*, *45*(4), 301–309.

Chan, R. P. K. & Gorgey, A. (2013). Active and passive symmetrization of Runge-Kutta Gauss methods. *Applied Numerical Mathematics, 67,* 64–77.

Chan, R. P. K. & Razali, N. (2014). Smoothing effects on the IMR and ITR. *Numerical Algorithms, 65*(3), 401-420.

Cong, N. H. (1994). Parallel iteration of symmetric Runge-Kutta methods for nonstiff initial-value problems. *Journal of Computational and Applied Mathematics, 51*(1), 117–125.

Cooper, G. J. & Butcher, J. C. (1983). An iteration scheme for implicit Runge-Kutta methods. *IMA Journal of Numerical Analysis, 3*(2), 127-140.

Dahlquist, G. G. (1963). A special stability problem for linear multistep methods. *BIT Numerical Mathematics, 3*(1), 27-43.

Dekker, K. & Verwer J.G. (1984). *Stability of Runge-Kutta methods for stiff nonlinear differential equations.* North-Holland: CWI Monographs.

DeVries, P. & Hasbun, J. (2011). *A First Course in Computational Physics.* Sudbury, MA: Jones and Bartlett Publishers.

Dormand, J. R. (2018). *Numerical Methods for Differential Equations: A Computational Approach*. Boca Raton, FL: CRC Press.

Ehle, B. L. (1973). A-stable methods and Padé approximations to the exponential. *SIAM Journal on Mathematical Analysis, 4*(4), 671-680.

Enright, W. H., Hull, T. E. & Lindberg, B. (1975). Comparing numerical methods for stiff systems of ODEs. *BIT Numerical Mathematics, 15*(1), 10-48.

Fan, Z., Song, M. & Liu, M. (2009). The αth moment stability for the stochastic pantograph equation. *Journal of Computational and Applied Mathematics, 233*(2), 109-120.

Faragó, I., Havasi, Á. & Zlatev, Z. (2013). The convergence of diagonally implicit Runge-Kutta methods combined with Richardson extrapolation. *Computers and Mathematics with Applications, 65*(3), 395–401.

Fatunla, S. O. (2014). *Numerical Methods for Initial Value Problems in Ordinary Differential Equations*. San Diego, CA: Academic Press, Inc.

Gear, C. W. (1980). Runge-Kutta starters for multistep methods. *ACM Transactions on Mathematical Software (TOMS)*, *6*(3), 263-279.

González-Pinto, S., González-Concepción, C. & Montijano, J. I. (1994). Iterative schemes for Gauss methods. *Computers and Mathematics with Applications, 27*(7), 67-81.

González-Pinto, S., Hernández-Abreu, D. & Montijano, J. I. (2019). Variable step-size control based on two-steps for Radau IIA methods. Preprint: https://www.researchgate.net/publication/331687918_Variable_step-size_control_based_on_two-steps_for_Radau_IIA_methods

González-Pinto, S., Montijano, J. I. & Rández, L. (1995). Iterative schemes for three-stage implicit Runge-Kutta methods. *Applied Numerical Mathematics, 17*(4), 363–382.

Gorgey, A. (2012). *Extrapolation of symmetrized Runge-Kutta methods.* PhD thesis, ResearchSpace @ University of Auckland.

Gorgey, A. (2015). Extrapolation of symmetrized Runge-Kutta methods in the variable stepsize setting. *International Journal of Applied Mathematics and Statistics*, *55*(2), 14-22.

Gorgey, A. & Chan, R. P. K. (2015). Choice of strategies for extrapolation with symmetrization in the constant stepsize setting. *Applied Numerical Mathematics, 87,* 31-37.

Gorgey, A. & Mat, N. A. A. (2018). Efficiency of Runge-Kutta methods in solving simple harmonic oscillators. *Matematika, 34*(1), 1-12.

Gorgey, A. & Muhammad, H. (2017). Efficiency of Runge-Kutta methods in solving Kepler problem. *AIP Conference Proceedings. 1847*(1), 020016.

Guo, P. & Li, C. J. (2019). Razumikhin-type technique on stability of exact and numerical solutions for the nonlinear stochastic pantograph differential equations. *BIT Numerical Mathematics, 59*(1), 77-96.

Gustafsson, K. (1994). Control-theoretic techniques for stepsize selection in implicit Runge-Kutta methods. *ACM Transactions on Mathematical Software (TOMS), 20*(4), 496-517.

Hairer, E. & Wanner, G. (1981). Algebraically stable and implementable Runge-Kutta methods of high order. *SIAM Journal on Numerical Analysis, 18*(6), 1098-1108.

Hairer, E. & Wanner, G. (1996). *Solving Ordinary Differential Equations II*. London: Springer-Verlag Berlin Heidelberg.

Hairer, E. & Wanner, G. (1999). Stiff differential equations solved by Radau methods. *Journal of Computational and Applied Mathematics, 111*(1-2), 93-111.

Higham, N. J. (1993). The accuracy of floating point summation. *SIAM Journal on Scientific Computing, 14*(4), 783-799.

Hindmarsh, A. C. (1980). LSODE and LSODI, two new initial value ordinary differential equation solvers. *ACM Signum Newsletter, 15*(4), 10-11.

Hitchens, F. (2015). *Propeller Aerodynamics: The History, Aerodynamics & Operation of Aircraft Propellers.* Wellington, NZ: Andrews UK Limited.

Holder, A. & Eichholz, J. (2019). Modeling with delay differential equations. In *An Introduction to Computational Science*. Switzerland: Springer, Cham. 377-387.

Hussain, E. A. & Abdul-Abbass, Y. M. (2019). On Fuzzy differential equation. *Journal of Al-Qadisiyah for Computer Science and Mathematics*, *11*(2), 1-9.

Iserles, A. (2009). *A first course in the numerical analysis of differential equations*. New York, US: Cambridge University Press.

Ismail, A. & Gorgey, A. (2015). Behaviour of the extrapolated implicit midpoint and implicit trapezoidal rules with and without compensated summation. *Matematika, 31*(1), 47–57.

Kennedy, C. A. & Carpenter, M. H. (2019). Diagonally implicit Runge-Kutta methods for stiff ODEs. *Applied Numerical Mathematics, 146*(6), 221-244.

Kim, I. P. & Kräuter, A. R. (2018). Decompositions of a matrix by means of its dual matrices with applications. *Linear Algebra and its Applications, 537,* 100-117.

Kulikov, G. Y. (2015). Embedded symmetric nested implicit Runge-Kutta methods of Gauss and Lobatto types for solving stiff ordinary differential equations and Hamiltonian systems. *Computational Mathematics and Mathematical Physics, 55*(6), 983–1003.

Kuntzmann, J. (1961). Neuere entwicklungen der methode von runge und kutta. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik, 41*(1), 28-31.

Lambert, J. D. (1991). *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. New York, US: John Wiley & Sons, Inc.

Lapidus, L. & Schiesser, W. E. (1976). *Numerical Methods for Differential Systems: Recent Developments in Algorithms, Software, and Applications*. United Kingdom: Academic Press Inc.

Liu, Y. (1995). Stability analysis of θ-methods for neutral functional-differential equations. *Numerische Mathematik, 70*(4), 473-485.

Liu, M. Y., Zhang, L. & Zhang, C. F. (2019). Study on banded implicit Runge-Kutta methods for solving stiff differential equations. *Mathematical Problems in Engineering, 2019,* 4850872.

Muhammad, M. H. (2018). *An Efficient Implementation Technique for Implicit Runge-Kutta Methods in Solving the Stiff Problems*. Universiti Pendidikan Sultan Idris.

Muhammad, M. H. & Gorgey, A. (2018). Investigation on the most efficient ways to solve the implicit equations for Gauss methods in the constant stepsize setting. *Applied Mathematical Sciences, 12*(2), 93-103.

Najafi, R. & Nemati, S. B. (2017). Numerical solution of the forced Duffing equations using Legendre multiwavelets. *Computational Methods for Differential Equations, 5*(1), 43-55.

Nazari, F., Mohammadian, A., Charron, M. & Zadra, A. (2014). Optimal high-order diagonally-implicit Runge-Kutta schemes for nonlinear diffusive systems on atmospheric boundary layer. *Journal of Computational Physics, 271,* 118-130.

Owolabi, K. M. (2019). Mathematical modelling and analysis of love dynamics: A fractional approach. *Physica A: Statistical Mechanics and its Applications, 525,* 849-865.

Peat, K. D. & Thomas, R. M. (1989). *Implementation of iteration schemes for implicit Runge-Kutta methods.* University of Manchester. Department of Mathematics.

Prothero, A., & Robinson, A. (1974). On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. *Mathematics of Computation*, *28*(125), 145-162.

Ramos, H. (2019). Development of a new Runge-Kutta method and its economical implementation. *Computational and Mathematical Methods, 1*(2), e1016.

Rang, J. (2016). The Prothero and Robinson example: Convergence studies for Runge-Kutta and Rosenbrock-Wanner methods. *Applied Numerical Mathematics, 108,* 37-56.

Rasedee, A. F. N., Ishak, N., Hamzah, S. R., Ijam, H. M., Suleiman, M., Ibrahim, Z. B., et al. (2017). Variable order variable stepsize algorithm for solving nonlinear Duffng oscillator. *Journal of Physics: Conference Series*, *890,* 012045.

Razali, N., Nopiah, Z. M. & Othman, H. (2018). Comparison of one-step and two-step symmetrization in the variable stepsize setting. *Sains Malaysiana, 47*(11), 2927-2932.

Rechenberg, H. (2001). *The Historical Development of Quantum Theory,* Volume 1. New York, US: Springer Science & Business Media.

Robertson, H. H. (1966). *The solution of a set of reaction rate equations*. Cambridge, Massachusetts: Academic Press. 178-182.

Roussel, M. R. (2019). *Nonlinear Dynamics: A hands-on introductory survey*. Bristol, UK: Morgan & Claypool Publishers.

Rushanan, J. J. (1989). On the Vandermonde matrix. *The American Mathematical Monthly, 96*(10), 921-924.

Sanderse, B. & Koren, B. (2012). Accuracy analysis of explicit Runge-Kutta methods applied to the incompressible Navier-Stokes equations. *Journal of Computational Physics, 231*(8), 3041– 3063.

Sanz-Serna, J. (1988). Runge-Kutta schemes for Hamiltonian systems. *BIT Numerical Mathematics, 28*(4), 877-883.

Sanz-Serna, J. M. (2016). Symplectic Runge-Kutta schemes for adjoint equations, automatic differentiation, optimal control, and more. *SIAM Review, 58*(1), 3-33.

Schäfer, E. (1975). A new approach to explain the "high irradiance responses" of photomorphogenesis on the basis of phytochrome. *Journal of Mathematical Biology*, *2*(1), 41-56.

Shampine, L. F. (1984). Stability of explicit Runge-Kutta methods. *Computers & Mathematics with Applications, 10*(6), 419-432.

Shampine, L. F. (1985). Local error estimation by doubling. *Computing, 34*(2), 179-190.

Shampine, L. F. (2018). *Numerical Solution of Ordinary Differential Equations*. Boca Raton, FL: Routledge.

Skvortsov, L. M. & Kozlov, O. S. (2014). Efficient implementation of diagonally implicit Runge-Kutta methods. *Mathematical Models and Computer Simulations, 6*(4), 415-424.

Sun, G. A. (2000). Simple way constructing symplectic Runge-Kutta methods. *Journal of Computational Mathematics, 18*(1), 61–68.

Swart, D., Jacques, J. B. & Lioen, W. M. (1998). Collecting real-life problems to test solvers for implicit differential equations. *CWI Quarterly, 11*(1), 83-100.

Toufik, M. & Atangana, A. (2017). New numerical approximation of fractional derivative with non-local and non-singular kernel: application to chaotic models. *The European Physical Journal Plus, 132*(10), 444.

Varah, J. M. (1979). On the efficient implementation of implicit Runge-Kutta methods. *Mathematics of Computation, 33*(146), 557.

Wang, P., Zhou, J., Wang, R. & Chen, J. (2017). New generalized variable stepsizes of the CQ algorithm for solving the split feasibility problem. *Journal of Inequalities and Applications, 2017*(1), 135.

Wang, Y. & Chen, Y. (2020). Shifted Legendre Polynomials algorithm used for the dynamic analysis of viscoelastic pipes conveying fluid with variable fractional order model. *Applied Mathematical Modelling, 81*, 159-176.

Wilkie, J. & Çetinbaş, M. (2005). Variable-stepsize Runge-Kutta methods for stochastic Schrödinger equations. *Physics Letters A, 337*(3), 166-182.

Williams, G. (2017). *Linear Algebra with Applications.* United State of America: Jones & Barlett Learning.

Willoughby, R. A. (1974). Stiff Differential Systems. *International Symposium on Stiff Differential Systems.* Boston, MA: Springer. 1-19.

Xu, P., Yuan, Z., Jian, W. & Zhao, W. (2015). Variable step-size method based on a reference separation system for source separation. *Journal of Sensors, 2015*. 964098.

Yaici, M. & Hariche, K. (2019). A particular block Vandermonde matrix. *ITM Web of Conferences, 24.* 01008.

Yang, H., Yang, Z., Wang, P. & Han, D. (2019). Mean-square stability analysis for nonlinear stochastic pantograph equations by transformation approach. *Journal of Mathematical Analysis and Applications, 479*(1), 977-986.

Yang, X., Yang, Z. & Xiao, Y. (2020). *Asymptotical mean-square stability of linear θ-methods for stochastic pantograph differential equations: variable stepsize and transformation approach.* Unpublished manuscript. DOI: 10.22541/au.159023888.86381071

Ycart, B. (2012). A Case of Mathematical Eponymy: The Vandermonde Determinant. *arXiv preprint arXiv:1204.4716*.

Ye, K. (2017). New classes of matrix decompositions. *Linear Algebra and its Applications, 514*, 47-81.

Yu, W. & Jafari, R. (2019). Fuzzy Differential Equations. In *Modeling and Control of Uncertain Nonlinear Systems with Fuzzy Equations and Z-Number,* Piscataway, NJ: Wiley-IEEE Press. 21-37.

Zhang, D. K. (2019). Discovering New Runge-Kutta Methods using Unstructured Numerical Search. *arXiv preprint arXiv:1911.00318*.

Zhang, H., Sandu, A. & Tranquilli, P. (2015). Application of approximate matrix factorization to high order linearly implicit Runge-Kutta methods. *Journal of Computational and Applied Mathematics, 286,* 196–210.

Zhu, B., Hu, Z., Tang, Y. & Zhang, R. (2016). Symmetric and symplectic methods for gyrocenter dynamics in time-independent magnetic fields. *International Journal of Modeling, Simulation, and Scientific Computing, 7*(02), 1650008.

APPENDIX A

The code below shows the MATLAB code for the implementation of 2-stage (G2) Gauss method. The first part contains the main function of G2 method followed by the nstep file which functioning as computing the updated y_n and variable file which compute the variable stepsize where the extrapolation technique is implemented. These three parts is required to all schemes which denote as G2SNGMR (G2 with simplified Newton for GMR), G2SNHW (G2 with simplified Newton for HW) and G2SNMHW (G2 with simplified Newton for modified HW) schemes. The code for real life problems is computed in different file. Lastly, the order plot file is to run the numerical approximation.

1. MATLAB code for González-Pinto et al. (1994, 1995)

Step 1 (G2SNGMR_fix.m)

```
function [YY,trace,theta]=G2SNGMR_fix(f,J,tol,x,y,h,theta)
maxit = 10;
a1 = 1/4;
a2 = 1/2;
b = sqrt(3)/6;
A = [a1,a1-b; a1+b,a1];
C = [a2-b;a2+b];
m = length(y);
s = length(c);
e = ones(s,1);
z = zeros(m,1);
Z = kron(e,z);
trace = 0;
Y = kron(e,y);
kappa = 1.e-1;
Im = eye(m);

T = [sqrt(3)/6,0; sqrt(3)/3,sqrt(3)/6];

F1 = f(x+c(1)*h,y);
F2 = f(x+c(2)*h,y);

J1 = J(x+c(1)*h,y);
J2 = J(x+c(2)*h,y);

DG = [Im - T(1,1)*h*J1,-T(1,2)*h*J2 ; -T(2,1)*h*J1,Im - T(2,2)*h*J2];

G = [-A(1,1)*h*F1 - A(1,2)*h*F2 ; -A(2,1)*h*F1 - A(2,2)*h*F2];

DZ = DG\(-G);
sigma = norm(DZ,'inf');
eta = theta/(1.-theta);
if (eta*sigma <= kappa*tol)
    YY = Y + DZ;
```

```
    return;
end
Z = Z + DZ;

for i = 1:maxit
    Z1 = Z(1:m);
    Z2 = Z(m+1:2*m);
    F1 = f(x+c(1)*h,Z1+y);
    F2 = f(x+c(2)*h,Z2+y);
    G = [Z1 - A(1,1)*h*F1 - A(1,2)*h*F2 ; Z2 - A(2,1)*h*F1 -
         A(2,2)*h*F2];
    DZ = DG\(-G);
    beta = norm(DZ,'inf');
    theta = beta/sigma;

    if (theta >= 1)
        trace = 1;
        eta = 1.0;
        break
    end

    if ((theta^(10-i)/(1-theta))*beta > kappa*tol)
        trace = 1;
        break;
    end

    eta = theta/(1-theta);
    theta = (max(1.e-14,theta))^(0.8);
    Z = Z + DZ;
    if (eta*beta)<=(kappa*tol)
        break;
    end
    sigma = beta;
end
YY=[Z(1:m)+y ; Z(m+1:2*m)+y];
```

Step 2 (nstep_fixG2SNGMR.m)

```
function [y,trace,hout]=nstep_fixG2SNGMR(f,J,tol,x0,y0,h0,n)
theta = 0.8;
m = length(y0(:));
hout = h0;
trace = 1;

while trace
    trace = 0;
    x = x0;
    y = y0(:);

    for i=1:n
        [Y,tr,theta] = G2SNGMR_fix(f,J,tol,x,y,hout,theta);
        if tr
            trace = 1;
            hout = hout/2;
            break
        end
        y = y-sqrt(3)*Y(1:m)+sqrt(3)*Y(m+1:2*m);
        x = x+hout;
```

```
    end
end
if (hout < h0)
    trace= 1;
end
```

Step 3 (variable_fixG2SNGMR)

```
function[xout,yout,h,errout]=variable_fixG2SNGMR(f,J,tol,x0,xn,y0,h0)

p = 4;

if nargin<6
    tol = 1.e-6;
end

x = x0;
y = y0(:);
err = (y-y0)/(2^p-1);
xout = x;
yout = y';
errout = err;
hmax = (xn-x)/16;
hmin = (xn-x)/(2.e8);
h = max([h0,(xn-x)/1.e7]);

while (x < xn) && (h >= hmin)
    if (x + h > xn)
        h = xn - x ;
    end
    [y1,trace,hout] = nstep_fixG2SNGMR(f,J,tol,x,y,h,1);
    if trace
        h = hout;
    end
    [y2,~,~] = nstep_fixG2SNGMR(f,J,tol,x,y,h/2,2);
    err =(y2 - y1)/15;
    delta = norm(err,'inf');
    tau = tol*max(norm(y1,'inf'),1.0);
    if delta <= tau
        x = x+h;
        y = err+y2;
        xout = [xout;x];
        yout = [yout;y'];
        errout = [errout;err];
        if (delta ~= 0.0)
            h = min ([hmax,4*h,0.9*h*(tau/delta)^(1/(p+1))]);
        else
            h = min([hmax,4*h]);
        end
    else
       h = h*max([0.25,0.9*(tau/delta)^(1/(p+1))]);
    end
end
if (x < xn)
    disp('SINGULARITY LIKELY G2.')
end
```

2. Matlab code for Hairer and Wanner (1999)

Step 1 (G2SNHW_fix.m)

```
function [YY,trace,theta]=G2SNHW_fix(f,J,tol,x,y,h,theta)
maxit = 10;
a1 = 1/4;
a2 = 1/2;
b = sqrt(3)/6;
A = [a1,a1-b;a1+b,a1];
c = [a2-b ; a2+b];
m = length(y);
s = length(c);
e = ones(s,1);
z = zeros(m,1);
Z = kron(e,z);
trace = 0;
kappa = 1.e-1;
Im = eye(m);

T = [sqrt(3)/6,0 ; sqrt(3)/3,sqrt(3)/6];
Tinv = inv(T);
Ainv = inv(A);
S = Tinv*Ainv*T;
W = kron(Tinv,Im)*Z;

F1 = f(x+c(1)*h,y);
F2 = f(x+c(2)*h,y);
F = [F1;F2];

J1 = J(x+c(1)*h,y);
J2 = J(x+c(2)*h,y);

DG = [(1/h)*S(1,1)*Im - Tinv(1,1)*J1,(1/h)*S(1,2)*Im - Tinv(1,2)*J2;
      (1/h)*S(2,1)*Im - Tinv(2,1)*J1,(1/h)*S(2,2)*Im - Tinv(2,2)*J2];

G = (1/h)*kron(S,Im)*W - kron(Tinv,Im)*F;

DW = DG\(-G);

sigma = norm(DW,'inf');
eta = theta/(1.-theta);
if (eta*sigma <= kappa*tol)
   YY =  Y + DW;
    return;
end
W = W+ DW;


for i = 1:maxit
    W1 = W(1:m);
    W2 = W(m+1:2*m);
    W = [W1;W2];

    TI = kron(T,Im)*W;
```

```
    F1 = f(x+c(1)*h,TI(1:m)+y);
    F2 = f(x+c(2)*h,TI(m+1:2*m)+y);
    F = [F1;F2];
    G = (1/h)*kron(S,Im)*W-kron(Tinv,Im)*F;

    DW = DG\(-G);
    beta = norm(DW,'inf');
    theta = beta/sigma;

    if (theta >= 1)
        trace = 1;
        eta = 1.0;
        break
    end
    if ((theta^(10-i)/(1-theta))*beta > kappa*tol)
        trace = 1;
        break;
    end

    eta = theta/(1-theta);
    theta = (max(1.e-16,theta))^(0.8);
    W = W + DW;
    if (eta*beta)<=(kappa*tol)
        break;
    end
    sigma = beta;
end
TI = kron(T,Im)*W;
YY = [TI(1:m)+y;TI(m+1:2*m)+y];
```


Step 2 (nstep_fixG2SNHW.m)

```
function [y,trace,hout]=nstep_fixG2SNHW(f,J,tol,x0,y0,h0,n)

theta = 0.8;
m = length(y0(:));
hout = h0;
trace = 1;

while trace
    trace = 0;
    x = x0;
    y = y0(:);

    for i=1:n
        [Y,tr,theta] = G2SNHW_fix(f,J,tol,x,y,hout,theta);
        if tr
            trace = 1;
            hout = hout/2;
            break
        end
        y = y-sqrt(3)*Y(1:m)+sqrt(3)*Y(m+1:2*m);
        x = x+hout;
    end
end
if (hout < h0)
    trace = 1;
end
```

Step 3 (variable_fixG2SNHW.m)

```
function [xout,yout,h,errout]=variable_fixG2SNHW(f,J,tol,x0,xn,y0,h0)

p = 4;

if nargin<6
   tol = 1.e-6;
end

x = x0;
y = y0(:);
err = (y-y0)/(2^p-1);
xout = x;
yout = y';
errout = err;
hmax = (xn-x)/16;
hmin =(xn-x)/(2.e8);
h = max([h0,(xn-x)/1.e7]);

while (x < xn) && (h >= hmin)
    if (x + h > xn)
        h = xn - x ;
    end
    [y1,trace,hout]=nstep_fixG2SNHW(f,J,tol,x,y,h,1);
    if trace
        h = hout;
    end
    [y2,~,~]=nstep_fixG2SNHW(f,J,tol,x,y,h/2,2);
    err =(y2 - y1)/15;
    delta = norm(err,'inf');
    tau = tol*max(norm(y1,'inf'),1.0);
    if delta <= tau
        x = x+h;
        y = err+y2;
        xout = [xout;x];
        yout = [yout;y'];
        errout = [errout;err];
        if (delta ~= 0.0)
            h = min ([hmax,4*h,0.9*h*(tau/delta)^(1/(p+1))]);
        else
            h = min([hmax,4*h]);
        end
    else
       h = h*max([0.25,0.9*(tau/delta)^(1/(p+1))]);
    end
end
if (x < xn)
    disp('SINGULARITY LIKELY G2.')
end
```

3. Matlab code for modified Hairer and Wanner (1999)

Step 1 (G2SNMHW_fix.m)

```
function [YY,trace,theta] = G2SNMHW(f,J,tol,x,y,h,theta)

a1 = 1/4;
a2 = 1/2;
b = sqrt(3)/6;
A = [a1,a1-b;a1+b,a1];
c = [a2-b;a2+b];

m = length(y);
s = length(c);
e = ones(s,1);
z = zeros(m,1);
Z = kron(e,z);
trace = 0;
Y = kron(e,y);
Im = eye(m);

F1 = f(x+c(1)*h,y);
F2 = f(x+c(2)*h,y);

J1 = J(x+c(1)*h,y);
J2 = J(x+c(2)*h,y);

Minv = inv([Im-h*A(1,1)*J1,-h*A(1,2)*J2 ;-h*A(2,1)*J1,...
       Im-h*A(2,2)*J2]);

G1 = h*A(1,1)*F1+h*A(1,2)*F2;
G2 = h*A(2,1)*F1+h*A(2,2)*F2;
G = [G1;G2];
DZ = Minv*G;

temp = norm(DZ,'inf');
eta = theta/(1-theta);
if (eta*temp <= 1.e-1*tol)
    YY = Y+DZ;
    return;
end
Z = Z+DZ;

maxit=10;
for i = 1:maxit
    z1 = Z(1:m);
    z2 = Z(m+1:2*m);

    F1 = f(x+c(1)*h,z1+y);
    F2 = f(x+c(2)*h,z2+y);

    G1 = h*A(1,1)*F1+h*A(1,2)*F2-z1;
    G2 = h*A(2,1)*F1+h*A(2,2)*F2-z2;
    G = [G1;G2];
    DZ = Minv*G;
```

```
    delta = norm(DZ,'inf');
    theta = delta/temp;
    if (theta >= 1)
        trace = 1;
        eta=1.0;
        break;
    end

    if ((theta^(10-i)/(1-theta))*delta > 1.e-1*tol)
        trace = 1;
        break;
    end
    eta = theta/(1-theta);
    theta = (max(1.0e-16,theta))^(0.8);
    Z = Z+DZ;
    if (eta*delta <= 1.e-1*tol)
        break;
    end
    temp = delta;
end
YY = [y+Z(1:m);y+Z(m+1:2*m)];
```

Step 2 (nstep_fixG2SNMHW.m)

```
function [y,trace,hout] = nstep_fixG2SNMHW(f,J,tol,x0,y0,h0,m)
n = length(y0(:));
hout = h0;
trace = 1;
theta = 0.8;

while trace
    trace = 0;
    x = x0;
    y = y0(:);
    for i = 1:m
        [Y,tr,theta] = G2SNMHW(f,J,tol,x,y,hout,theta);
        if tr
            trace = 1;
            hout = hout/2;
            break;
        end
        y = y+sqrt(3)*(Y(n+1:2*n)-Y(1:n));
        x = x+hout;
    end
end

if (hout < h0)
    trace = 1;
end
```

Step 3 (variable_fixG2SNMHW.m)

```
Function[xout,yout,h,errout]=variable_fixG2SNMHW(f,J,tol,x0,xf,y0,h0)

p = 4;
pow = 1/(p+1);
```

```
if nargin < 6
    tol = 1.e-6;
end

x = x0;
y = y0(:);
err = (y-y0)/(2^p-1);
hmax = (xf-x)/16;
hmin = (xf-x)/2.e8;
h = max([h0,(xf-x)/1.e7]);
xout =x;
yout = y';
errout = err;

while ((x < xf) && (h >= hmin))
  if x+h > xf, h = xf-x; end
    [y1,trace,hout] = nstep_fixG2SNMHW(f,J,tol,x,y,h,1);
    if trace,[h,hout]; h= hout; end
    [y2,~,~] = nstep_fixG2SNMHW(f,J,tol,x,y,h/2,2);
  err = (y2-y1)/15;
  delta = norm(err,'inf');
  tau = tol*max([norm(y,'inf'),1.0]);
  if (delta <= tau)
    x = x+h;
    y = y2+err;
    xout = [xout;x];
    yout = [yout;y'];
    errout = [errout;err];
    if (delta ~= 0.0)
      h = min([hmax,4*h,0.9*h*(tau/delta)^pow]);
    else
      h = min([hmax,4*h]);
    end
  else
    h = h*max([0.25,0.9*(tau/delta)^pow]);
  end
end

if (x < xf)
  disp('SINGULARITY LIKELY  G2.')
end
```

4. Matlab code for real life problems (problem.m)

```
function [f,J,tol,x0,xn,y0,h0] = problem(problem)

tol=1.e-7;

switch (problem)

    case 'PR'   %prothero robinson
        q=-10000;
        f=@(x,y)(q*y+cos(x)-q*sin(x));
        J=@(x,y)(q);
        x0=0;
        xn=5;
        h0=0.001;
        y0=0;
```

```
case 'VDP'   %van de pol
    eps=1.e-3;
    f=@(x,y)([y(2);((1-y(1)^2)*y(2)-y(1))/eps]);
    J=@(x,y)([0,1;(-2*y(1)*y(2)-1)/eps,(1-y(1)^2)/eps]);
    x0=0;
    xn=5;
    h0=0.01;
    y0=[2;0];

case 'ROBER'   %robertson
    f=@(x,y)([-0.04*y(1)+10^4*y(2)*y(3);...
        0.04*y(1)-10^4*y(2)*y(3)-(3.e7)*y(2)^2;...
        (3.e7)*y(2)^2]);
    J=@(x,y)([-0.04,10^4*y(3),10^4*y(2);...
        0.04,(-10^4)*y(3)-(6.e7)*y(2),-10^4*y(2);...
        0,(6.e7)*y(2),0]);
    x0=0;
    xn=10;
    h0=0.01;
    y0=[1;0;0];

case 'HIRES'
    f=@(x,y)([-1.71*y(1)+0.43*y(2)+8.32*y(3)+0.0007;...
        1.71*y(1)-8.75*y(2);...
        -10.03*y(3)+0.43*y(4)+0.035*y(5);...
        8.32*y(2)+1.71*y(3)-1.12*y(4);...
        -1.745*y(5)+0.43*y(6)+0.43*y(7);...
        -280*y(6)*y(8)+0.69*y(4)+1.71*y(5)-
        0.43*y(6)+0.69*y(7);...
        280*y(6)*y(8)-1.81*y(7);...
        -280*y(6)*y(8)+1.81*y(7)]);
    J=@(x,y)([-1.71,0.43,8.32,0,0,0,0,0;...
        1.71,-8.75,0,0,0,0,0,0;...
        0,0,-10.03,0.43,0.035,0,0,0;...
        0,8.32,1.71,-1.12,0,0,0,0;...
        0,0,0,0,-1.745,0.43,0.43,0;...
        0,0,0,0.69,1.71,-280*y(8)-0.43,0.69,-280*y(6);...
        0,0,0,0,0,280*y(8),-1.81,280*y(6);...
        0,0,0,0,0,-280*y(8),1.81,-280*y(6)]);
    x0=0;
    xn=321.8122;
    h0=0.01;
    y0=[1;0;0;0;0;0;0;0.0057];

case 'KAPS'
    q = -10000;
    f=@(x,y)([(q-2)*y(1) - q*y(2)^2; y(1) - y(2) - y(2)^2]);
    J=@(x,y)([(q-2),-2*q*y(2); 1,-1 - 2*y(2)]);
    y0 = [1;1];
    x0 = 0;
    xn = 5;
    h0 = 0.01;

case 'BRUS'    %brusselator
    f=@(x,y)([1 +(y(1)^2)*y(2) - 4*y(1);3*y(1) - (y(1)^2)*y(2)]);
    J=@(x,y)([2*y(1)*y(2)-4,y(1)^2;3 - 2*y(1)*y(2),-y(1)^2]);
    x0=0;
    xn=10;
```

```
        y0=[1.5;3];
        h0=0.01;
        % n=100;

    case 'OREG'    %oreganator
        f=@(x,y)([77.27*(y(2)+y(1)*(1-y(1)*8.375*10^(-6) - y(2)));
                  (1/77.27)*(y(3)-(1+y(1))*y(2)); 0.161*(y(1)-y(3))]);

        J=@(x,y)([77.27-2*77.27*y(1)*8.375*10^(-6) - 77.27*y(2),
                  77.27 - 77.27*y(1),0 ; -y(2)/77.27,-1/77.27 -
                  y(1)/77.27,1/77.27 ; 0.161,0,-0.161]);
        x0 = 0;
        y0 = [1;2;3];
        xn = 30;
        h0 = 0.01;
```

5. Order plot to run the data (order_testproblem.m)

```
clearvars
clc

[f,J,tol,x0,xn,y0,h0] = problem('ROBER');

n = (xn-x0)/h0;
y = y0;
nit = 8;
m = length(y);
Tol = zeros(nit,1);

Y1 = zeros(nit,m);
C1 = zeros(nit,1);
LE1 = zeros(nit,1);
err1 = zeros(nit,m);
Lerrout1 = zeros(nit,1);

Y2 = zeros(nit,m);
C2 = zeros(nit,1);
LE2 = zeros(nit,1);
err2 = zeros(nit,m);
Lerrout2 = zeros(nit,1);

Y3 = zeros(nit,m);
C3 = zeros(nit,1);
LE3 = zeros(nit,1);
err3 = zeros(nit,m);
Lerrout3 = zeros(nit,1);

H = zeros(nit,1);

rep = 30;

for i=1:nit
    tic
    for j=1:rep

[xout1,yout1,hout1,errout1]=variable_fixG2SNGMR(f,J,tol,x0,xn,y0,h0);
    end
```

```
    cp1=toc;

    tic
    for j=1:rep

[xout2,yout2,hout2,errout2]=variable_fixG2SNHW(f,J,tol,x0,xn,y0,h0);
    end
    cp2=toc;

    tic
    for j=1:rep

[xout3,yout3,hout3,errout3]=variable_fixG2SNMHW(f,J,tol,x0,xn,y0,h0);
    end
    cp3=toc;

    Y1(i,:)=yout1(end);
    C1(i)=cp1;

    Y2(i,:)=yout2(end);
    C2(i)=cp2;

    Y3(i,:)=yout3(end);
    C3(i)=cp3;

    err1(i,:)=errout1(end);
    err2(i,:)=errout2(end);
    err3(i,:)=errout3(end);
    Tol(i) = tol;
    tol = tol/10;

    H(i)=h0;
    h0=h0/2;
    n=2*n;

    if i==1
        disp('~Iteration 1~');
    elseif i==2
        disp('~Iteration 2~');
    elseif i==3
        disp('~Iteration 3~');
    elseif i==4
        disp('~Iteration 4~');
    elseif i==5
        disp('~Iteration 5~');
    elseif i==6
        disp('~Iteration 6~');
    elseif i==7
        disp('~Iteration 7~');
    elseif i==8
        disp('~Iteration 8~');
    end

end


yexact=Y1(nit,:);
```

```
yexact1=Y2(nit,:);
yexact2=Y3(nit,:);
E1=abs(Y1-kron(ones(nit,1),yexact));
E2=abs(Y2-kron(ones(nit,1),yexact1));
E3=abs(Y3-kron(ones(nit,1),yexact2));

for i=1:nit
    LE1(i)=norm(E1(i,:));
    LE2(i)=norm(E2(i,:));
    LE3(i)=norm(E3(i,:));
end

LTol=Tol;

LC1=(C1/rep);
LC2=(C2/rep);
LC3=(C3/rep);

yexact=err1(nit,:);
yexact1=err2(nit,:);
yexact2=err3(nit,:);

errout1=abs(err1-kron(ones(nit,1),yexact))/15;
errout2=abs(err2-kron(ones(nit,1),yexact))/15;
errout3=abs(err3-kron(ones(nit,1),yexact))/15;

for i=1:nit
    Lerrout1(i)=norm(errout1(i,:));
    Lerrout2(i)=norm(errout2(i,:));
    Lerrout3(i)=norm(errout3(i,:));
end


figure(1)
loglog(LTol,LE1,'bx-');
hold on
loglog(LTol,LE2,'ro-');
hold on
loglog(LTol,LE3,'mv-');
legend('GMR scheme','HW scheme','Modified HW scheme');
xlabel('\fontsize{14}Tolerance');
ylabel('\fontsize{14}||Global Error||');
title('\fontsize{14}ROBERTSON')
grid on

figure(2)
loglog(LC1,LE1,'bx-');
hold on
loglog(LC2,LE2,'ro-');
hold on
loglog(LC3,LE3,'mv-');
legend('GMR scheme','HW scheme','Modified HW scheme');
xlabel('\fontsize{14}CPU Time');
ylabel('\fontsize{14}||Global Error||');
title('\fontsize{14}ROBERTSON')
grid on
figure(3)
loglog(LTol,Lerrout1,'bx-');
```

```
hold on
loglog(LTol,Lerrout2,'ro-');
hold on
loglog(LTol,Lerrout3,'mv-');
legend('GMR scheme','HW scheme','Modified HW scheme');
xlabel('\fontsize{14}Tolerance');
ylabel('\fontsize{14}||Error Estimation||');
title('\fontsize{14}ROBERTSON')
grid on
```

APPENDIX B

The code below shows the MATLAB code for the implementation of 3-stage (G3) Gauss method. The first part contains the main function of G3 method followed by the nstep file which functioning as computing the updated y_n and variable file which compute the variable stepsize where the extrapolation technique is implemented. These three parts is required to all schemes which denote as G3SNGMR (G3 with simplified Newton for GMR), G3SNHW (G3 with simplified Newton for HW) and G3SNMHW (G3 with simplified Newton for modified HW) schemes. The code for real life problems is computed in different file. Lastly, the order plot file is to run the numerical approximation.

1. MATLAB code for González-Pinto et al. (1994, 1995)

Step 1 (G3SNGMR_fix)

```
function [YY,trace,theta] = G3SNGMR_fix(f,J,tol,x,y,h,theta)

A = [5/36, 2/9-sqrt(15)/15, 5/36-sqrt(15)/30;...
     5/36+sqrt(15)/24, 2/9, 5/36-sqrt(15)/24;...
     5/36+sqrt(15)/30, 2/9+sqrt(15)/15, 5/36];
c = [0.5-sqrt(15)/10;1/2;0.5+sqrt(15)/10];
trace = 0;
m = length(y);
s = length(c);
e = ones(s,1);
z = zeros(m,1);
Z = kron(e,z);
Y = kron(e,y);

T = [0.1190762649202001,-0.01352480890549548,0.002955703944789629;...
    0.2567321613764653,0.2864264722250291,- 0.008257284502425157;...
    0.2617169889707876,0.5210947821158048,0.2027174624121108];

J1=J(x+c(1)*h,y);
J2=J(x+c(2)*h,y);
J3=J(x+c(3)*h,y);

F1=f(x+c(1)*h,y);
F2=f(x+c(2)*h,y);
F3=f(x+c(3)*h,y);

DG = [eye(m)-h*T(1,1)*J1,-h*T(1,2)*J2,-h*T(1,3)*J3;...
     -h*T(2,1)*J1,eye(m)-h*T(2,2)*J2,-h*T(2,3)*J3;...
     -h*T(3,1)*J1,-h*T(3,2)*J2,eye(m)-h*T(3,3)*J3];

G1 = - h*(A(1,1)*F1+A(1,2)*F2+A(1,3)*F3);
G2 = - h*(A(2,1)*F1+A(2,2)*F2+A(2,3)*F3);
G3 = - h*(A(3,1)*F1+A(3,2)*F2+A(3,3)*F3);
```

```
G = [G1;G2;G3];
DZ = DG\(-G);

temp = norm(DZ,'inf');
eta = theta/(1.-theta);
if (eta*temp <= 1.e-1*tol)
    YY = Y+DZ;
    return;
end
Z = Z+DZ;

maxit=10;

for i = 1:maxit
    z1 = Z(1:m);
    z2 = Z(m+1:2*m);
    z3 = Z(2*m+1:3*m);

    F1 = f(x+c(1)*h,z1+y);
    F2 = f(x+c(2)*h,z2+y);
    F3 = f(x+c(3)*h,z3+y);

    G1 = z1-( h*(A(1,1)*F1+A(1,2)*F2+A(1,3)*F3));
    G2 = z2-(h*(A(2,1)*F1+A(2,2)*F2+A(2,3)*F3));
    G3 = z3-(h*(A(3,1)*F1+A(3,2)*F2+A(3,3)*F3));
    G = [G1;G2;G3];
    DZ = DG\(-G);

    delta = norm(DZ,'inf');
    theta = delta/temp;
    if (theta >= 1)
        trace = 1;
        eta=1.0;
        break;
    end

    if ((theta^(10-i)/(1-theta))*delta > 1.e-1*tol)
        trace = 1;
        break;
    end
    eta = theta/(1-theta);
    theta = (max(1.0e-16,theta))^(0.8);
    Z = Z+DZ;
    if (eta*delta <= 1.e-1*tol)
        break;
    end
    temp = delta;
end
YY = [y+Z(1:m);y+Z(m+1:2*m);y+Z(2*m+1:3*m)];
```

Step 2 (nstep_fixG3SNGMR.m)

```
function [y,trace,hout]=nstep_fixG3SNGMR(f,J,tol,x0,y0,h0,n)

m = length(y0(:));
hout = h0;
trace = 1;
theta = 0.8;
```

```
while trace
    trace = 0;
    x = x0;
    y = y0(:);

    for i=1:n
        [Y,tr,theta] = G3SNGMR_fix(f,J,tol,x,y,hout,theta);
        if tr
            trace = 1;
            hout = hout/2;
            break
        end
        y = -y+(1/3)*(5*Y(1:m)-4*Y(m+1:2*m)+5*Y(2*m+1:3*m));
        x = x+hout;
    end
end
if (hout < h0)
    trace = 1;
end
```

Step 3 (variable_fixG3SNGMR.m)

```
function
[xout,yout,h,errout]=variable_fixG3SNGMR(f,J,tol,x0,xn,y0,h0)

p = 6;
pow = 1/(p+1);
if nargin < 6
    tol = 1.e-6;
end

x = x0;
y = y0(:);
err = (y-y0)/(2^p-1);
hmax = (xn-x)/16;
hmin = (xn-x)/2.e8;
h = max([h0,(xn-x)/1.e7]);
xout = x;
yout = y';
errout = err;

while ((x < xn) && (h >= hmin))
  if x+h > xn, h = xn-x; end
    [y1,trace,hout] = nstep_fixG3SNGMR(f,J,tol,x,y,h,1);
    if trace,[h,hout]; h= hout; end
    [y2,~,~] =nstep_fixG3SNGMR(f,J,tol,x,y,h/2,2);
  err = (y2-y1)/63;
  delta = norm(err,'inf');
  tau = tol*max([norm(y,'inf'),1.0]);
  if (delta <= tau)
    x = x+h;
    y = y2+err;
    xout = [xout;x];
    yout = [yout;y'];
    errout = [errout;err];
    if (delta ~= 0.0)
      h = min([hmax,4*h,0.9*h*(tau/delta)^pow]);
```

```
    else
      h = min([hmax,4*h]);
    end
  else
    h = h*max([0.25,0.9*(tau/delta)^pow]);
  end
end

if (x < xn)
  disp('SINGULARITY LIKELY  G3.')
end
```

2. Matlab code for Hairer and Wanner (1999)

Step 1 (G3SNHW_fix.m)

```
function [YY,trace,theta]=G3SNHW_fix(f,J,tol,x,y,h,theta)

maxit = 10;
a1 = sqrt(15)/15;
a2 = sqrt(15)/30;
a3 = sqrt(15)/24;
b = sqrt(15)/10;
A = [5/36 , 2/9 - a1 , 5/36 - a2 ; 5/36 + a3 , 2/9 , 5/36 - a3 ;...
    5/36 + a2 , 2/9 + a1 , 5/36];
C = [1/2 - b ; 1/2 ; 1/2 + b];
m = length(y);
s=length(c);
e = ones(s,1);
z = zeros(m,1);
Z = kron(e,z);
Y = kron(e,y);
trace = 0;
kappa = 1.e-1;
Im = eye(m);

T = [0.1190762649202001,-0.01352480890549548,0.002955703944789629;...
     0.2567321613764653,0.2864264722250291,-0.008257284502425157;...
     0.2617169889707876,0.5210947821158048,0.2027174624121108];

Tinv = inv(T);
Ainv = inv(A);
S = Tinv*Ainv*T;
W = kron(Tinv,Im)*Z;


F1 = f(x+c(1)*h,y);
F2 = f(x+c(2)*h,y);
F3 = f(x+c(3)*h,y);
F = [F1;F2;F3];

J1 = J(x+c(1)*h,y);
J2 = J(x+c(2)*h,y);
J3 = J(x+c(3)*h,y);
```

```
DG = [(1/h)*S(1,1)*Im-Tinv(1,1)*J1,(1/h)*S(1,2)*Im-
      Tinv(1,2)*J2,(1/h)*S(1,3)*Im-Tinv(1,3)*J3;...
      (1/h)*S(2,1)*Im-Tinv(2,1)*J1,(1/h)*S(2,2)*Im-
      Tinv(2,2)*J2,(1/h)*S(2,3)*Im-Tinv(2,3)*J3;...
      (1/h)*S(3,1)*Im-Tinv(3,1)*J1,(1/h)*S(3,2)*Im-
      Tinv(3,2)*J2,(1/h)*S(3,3)*Im-Tinv(3,3)*J3];

G = (1/h)*kron(S,Im)*W-kron(Tinv,Im)*F;

DW = DG\(-G);

sigma = norm(DW,'inf');
eta = theta/(1.-theta);
if (eta*sigma <= kappa*tol)
   YY =  Y + DW;
    return;
end
W = W+ DW;

for i = 1:maxit
    W1 = W(1:m);
    W2 = W(m+1:2*m);
    W3 = W(2*m+1:3*m);
    W = [W1;W2;W3];

    TI = kron(T,Im)*W;

    F1 = f(x+c(1)*h,TI(1:m)+y);
    F2 = f(x+c(2)*h,TI(m+1:2*m)+y);
    F3 = f(x+c(3)*h,TI(2*m+1:3*m)+y);
    F = [F1;F2;F3];

    G = (1/h)*kron(S,Im)*W-kron(Tinv,Im)*F;

    DW = DG\(-G);
    beta = norm(DW,'inf');
    theta = beta/sigma;

    if (theta >= 1)
        trace = 1;
        eta = 1.0;
        break
    end
    if ((theta^(10-i)/(1-theta))*beta > kappa*tol)
        trace = 1;
        break;
    end

    eta = theta/(1-theta);
    theta = (max(1.e-16,theta))^(0.8);
    W = W + DW;
    if (eta*beta)<=(kappa*tol)
        break;
    end
    sigma = beta;
end
TI=kron(T,Im)*W;
YY=[TI(1:m)+y;TI(m+1:2*m)+y;TI(2*m+1:3*m)+y];
```

Step 2 (nstep_fixG3SNHW.m)

```
function [y,trace,hout]=nstep_fixG3SNHW(f,J,tol,x0,y0,h0,n)

theta = 0.8;
m = length(y0(:));
hout = h0;
trace = 1;

while trace
    trace = 0;
    x = x0;
    y = y0(:);

    for i = 1:n
        [Y,tr,theta] = G3SNHW_fix(f,J,tol,x,y,hout,theta);
        if tr
            trace = 1;
            hout = hout/2;
            break
        end
        y = -y+(5/3)*Y(1:m)-(4/3)*Y(m+1:2*m)+(5/3)*Y(2*m+1:3*m);
        x = x+hout;
    end
end
if (hout < h0)
    trace = 1;
end
```


Step 3 (variable_fixG3SNHW.m)

```
function [xout,yout,h,errout]=variable_fixG3SNHW(f,J,tol,x0,xn,y0,h0)

p = 6;

if nargin<6
   tol = 1.e-6;
end

x = x0;
y = y0(:);
err = (y-y0)/(2^p-1);
hmax = (xn-x)/16;
hmin = (xn-x)/2.e8;
h = max([h0,(xn-x)/1.e7]);
xout =x;
yout = y';
errout = err;

while (x < xn) && (h >= hmin)
    if (x + h > xn)
        h = xn - x ;
    end
    [y1,trace,hout] = nstep_fixG3SNHW(f,J,tol,x,y,h,1);
    if trace
        h = hout;
    end
```

```
    [y2,~,~] = nstep_fixG3SNHW(f,J,tol,x,y,h/2,2);
    err =(y2 - y1)/63;
    delta = norm(err,'inf');
    tau = tol*max(norm(y1,'inf'),1.0);
    if delta <= tau
        x = x+h;
        y = err+y2;
        xout = [xout;x];
        yout = [yout;y'];
        errout = [errout;err];
        if (delta ~= 0.0)
                h = min ([hmax,4*h,0.9*h*(tau/delta)^(1/(p+1))]);
        else
            h = min([hmax,4*h]);
        end
    else
       h = h*max([0.25,0.9*(tau/delta)^(1/(p+1))]);
    end
end
if (x < xn)
    disp('SINGULARITY LIKELY G3.')
end
```

3. Matlab code for modified Hairer and Wanner (1999)

Step 1 (G3SNMHW_fix.m)


```
function [YY,trace,theta] = G3SNMHW_fix(f,J,tol,x,y,h,theta)

a1 = sqrt(15)/15;
a2 = sqrt(15)/30;
a3 = sqrt(15)/24;
b = sqrt(15)/10;
A = [5/36 , 2/9 - a1 , 5/36 - a2 ; 5/36 + a3 , 2/9 , 5/36 - a3 ;...
    5/36 + a2 , 2/9 + a1 , 5/36];
c = [1/2 - b ; 1/2 ; 1/2 + b];

m = length(y);
s = length(c);
e = ones(s,1);
z = zeros(m,1);
Z = kron(e,z);
trace = 0;
Y = kron(e,y);
Im = eye(m);
F1 = f(x+c(1)*h,y);
F2 = f(x+c(2)*h,y);
F3 = f(x+c(3)*h,y);

J1 = J(x+c(1)*h,y);
J2 = J(x+c(2)*h,y);
J3 = J(x+c(3)*h,y);

Minv = inv([Im-h*A(1,1)*J1 , -h*A(1,2)*J2 , -h*A(1,3)*J3 ;...
       -h*A(2,1)*J1 , Im-h*A(2,2)*J2 , -h*A(2,3)*J3 ;...
       -h*A(3,1)*J1 , -h*A(3,2)*J2 , Im-h*A(3,3)*J3]);
```

```
G1 = h*A(1,1)*F1+h*A(1,2)*F2+h*A(1,3)*F3;
G2 = h*A(2,1)*F1+h*A(2,2)*F2+h*A(2,3)*F3;
G3 = h*A(3,1)*F1+h*A(3,2)*F2+h*A(3,3)*F3;
G = [G1;G2;G3];

DZ = Minv*G;

temp = norm(DZ,'inf');
eta = theta/(1-theta);
if (eta*temp <= 1.e-1*tol)
    YY = Y+DZ;
    return;
end
Z = Z+DZ;

maxit = 10;
for i = 1:maxit
    z1 = Z(1:m);
    z2 = Z(m+1:2*m);
    z3 = Z(2*m+1:3*m);

    F1 = f(x+c(1)*h,z1+y);
    F2 = f(x+c(2)*h,z2+y);
    F3 = f(x+c(3)*h,z3+y);

    G1 = h*A(1,1)*F1+h*A(1,2)*F2+h*A(1,3)*F3-z1;
    G2 = h*A(2,1)*F1+h*A(2,2)*F2+h*A(2,3)*F3-z2;
    G3 = h*A(3,1)*F1+h*A(3,2)*F2+h*A(3,3)*F3-z3;
    G = [G1;G2;G3];

    DZ = Minv*G;

    delta = norm(DZ,'inf');
    theta = delta/temp;
    if (theta >= 1)
        trace = 1;
        eta=1.0;
        break;
    end
    if ((theta^(10-i)/(1-theta))*delta > 1.e-1*tol)
        trace = 1;
        break;
    end
    eta = theta/(1-theta);
    theta = (max(1.0e-16,theta))^(0.8);
    Z = Z+DZ;
    if (eta*delta <= 1.e-1*tol)
        break;
    end
    temp = delta;
end
YY = [Z(1:m)+y;Z(m+1:2*m)+y;Z(2*m+1:3*m)+y];
```

Step 2 (nstep_fixG3SNMHW.m)

```
function [y,trace,hout] = nstep_fixG3SNMHW(f,J,tol,x0,y0,h0,m)

n = length(y0(:));
hout = h0;
trace = 1;
theta=0.8;

while trace
    trace = 0;
    x = x0;
    y = y0(:);
    for i = 1:m
        [Y,tr,theta] = G3SNMHW_fix(f,J,tol,x,y,hout,theta);
        if tr
            trace = 1;
            hout = hout/2;
            break;
        end
        y = -y+(5/3)*Y(1:n)-(4/3)*Y(n+1:2*n)+(5/3)*Y(2*n+1:3*n);
        x = x+hout;
    end
end

if (hout < h0)
    trace = 1;
end
```


Step 3 (variable_fixG3SNMHW.m)

```
function
[xout,yout,h,errout] = variable_fixG3SNMHW(f,J,tol,x0,xn,y0,h0)

p=6;
pow = 1/(p+1);
if nargin < 6
    tol = 1.e-6;
end

x = x0;
y = y0(:);
err = (y-y0)/(2^p-1);
hmax = (xn-x)/16;
hmin = (xn-x)/2.e8;
h = max([h0,(xn-x)/1.e7]);
xout = x;
yout = y';
errout = err;
while (x < xn) && (h >= hmin)
  if (x+h > xn)
      h = xn-x;
  end
    [y1,trace,hout] = nstep_fixG3SNMHW(f,J,tol,x,y,h,1);
    if trace
        h = hout;
    end
    [y2,~,~] = nstep_fixG3SNMHW(f,J,tol,x,y,h/2,2);
```

```
  err = (y2-y1)/63;
  delta = norm(err,'inf');
  tau = tol*max(norm(y,'inf'),1.0);
  if (delta <= tau)
    x = x+h;
    y = y2+err;
    xout = [xout;x];
    yout = [yout;y'];
    errout = [errout;err];
    if (delta ~= 0.0)
      h = min([hmax,4*h,0.9*h*(tau/delta)^pow]);
    else
      h = min([hmax,4*h]);
    end
  else
    h = h*max([0.25,0.9*(tau/delta)^pow]);
  end
end

if (x < xn)
  disp('SINGULARITY LIKELY  G3.')
end
```

4. Matlab code for real life problems (problem.m) is just similar to the one used for G2 method.

5. Order plot to run the data (order_testproblem.m)

```
clearvars
clc

[f,J,tol,x0,xn,y0,h0] = problem('ROBER');

n = (xn-x0)/h0;
y = y0;
nit = 8;
m = length(y);
Tol = zeros(nit,1);

Y1=zeros(nit,m);
C1=zeros(nit,1);
LE1=zeros(nit,1);
err1=zeros(nit,m);
Lerrout1=zeros(nit,1);

Y2=zeros(nit,m);
C2=zeros(nit,1);
LE2=zeros(nit,1);
err2=zeros(nit,m);
Lerrout2=zeros(nit,1);

Y3=zeros(nit,m);
C3=zeros(nit,1);
LE3=zeros(nit,1);
err3=zeros(nit,m);
Lerrout3=zeros(nit,1);
```

```
H=zeros(nit,1);

rep=1;

for i=1:nit
    tic
    for j=1:rep

[xout1,yout1,hout1,errout1]=variable_fixG3SNGMR(f,J,tol,x0,xn,y0,h0);
    end
    cp1=toc;

    tic
    for j=1:rep

[xout2,yout2,hout2,errout2]=variable_fixG3SNHW(f,J,tol,x0,xn,y0,h0);
    end
    cp2=toc;

    tic
    for j=1:rep

[xout3,yout3,hout3,errout3]=variable_fixG3SNMHW(f,J,tol,x0,xn,y0,h0);
    end
    cp3=toc;

    Y1(i,:)=yout1(end);
    C1(i)=cp1;
    Y2(i,:)=yout2(end);
    C2(i)=cp2;

    Y3(i,:)=yout3(end);
    C3(i)=cp3;

    err1(i,:)=errout1(end);
    err2(i,:)=errout2(end);
    err3(i,:)=errout3(end);

    Tol(i) = tol;
    tol = tol/10;

    H(i)=h0;
    h0=h0/2;
    n=2*n;


if i==1
        disp('~Iteration 1~');
    elseif i==2
        disp('~Iteration 2~');
    elseif i==3
        disp('~Iteration 3~');
    elseif i==4
        disp('~Iteration 4~');
    elseif i==5
        disp('~Iteration 5~');
    elseif i==6
```

```
        disp('~Iteration 6~');
    elseif i==7
        disp('~Iteration 7~');
    elseif i==8
        disp('~Iteration 8~');
    end

end

yexact=Y1(nit,:);
yexact1=Y2(nit,:);
yexact2=Y3(nit,:);

E1=abs(Y1-kron(ones(nit,1),yexact));
E2=abs(Y2-kron(ones(nit,1),yexact1));
E3=abs(Y3-kron(ones(nit,1),yexact2));

for i=1:nit
    LE1(i)=norm(E1(i,:));
    LE2(i)=norm(E2(i,:));
    LE3(i)=norm(E3(i,:));
end

LTol=Tol;

LC1=(C1/rep);
LC2=(C2/rep);
LC3=(C3/rep);
```



```
yexact=err1(nit,:);
yexact1=err2(nit,:);
yexact2=err3(nit,:);

errout1=abs(err1-kron(ones(nit,1),yexact))/63;
errout2=abs(err2-kron(ones(nit,1),yexact))/63;
errout3=abs(err3-kron(ones(nit,1),yexact))/63;

for i=1:nit
    Lerrout1(i)=norm(errout1(i,:));
    Lerrout2(i)=norm(errout2(i,:));
    Lerrout3(i)=norm(errout3(i,:));
end

figure(1)
loglog(LTol,LE1,'bx-');
hold on
loglog(LTol,LE2,'ro-');
hold on
loglog(LTol,LE3,'mv-');
legend('GMR scheme','HW scheme','Modified HW scheme');
xlabel('\fontsize{14}Tolerance');
ylabel('\fontsize{14}||Global Error||');
title('\fontsize{14}ROBERTSON')
grid on

figure(2)
loglog(LC1,LE1,'bx-');
hold on
```

```
loglog(LC2,LE2,'ro-');
hold on
loglog(LC3,LE3,'mv-');
legend('GMR scheme','HW scheme','Modified HW scheme');
xlabel('\fontsize{14}CPU Time');
ylabel('\fontsize{14}||Global Error||');
title('\fontsize{14}ROBERTSON')
grid on

figure(3)
loglog(LTol,Lerrout1,'bx-');
hold on
loglog(LTol,Lerrout2,'ro-');
hold on
loglog(LTol,Lerrout3,'mv-');
legend('GMR scheme','HW scheme','Modified HW scheme');
xlabel('\fontsize{14}Tolerance');
ylabel('\fontsize{14}||Error Estimation||');
title('\fontsize{14}ROBERTSON')
grid on
```