Generalized Mono-Implicit Runge-Kutta Methods for Stiff Ordinary Differential Equations

By

Fatima Dow

A Thesis Submitted to Saint Marys University, Halifax, Nova Scotia in Partial Fulfillment of the Requirements for the Degree of Master of Science in Applied Science.



Examination Committee:

Approved: Dr. Paul Muir, Senior Supervisor

Department of Mathematics and Computing Science

Approved: Dr. David Iron, External Examiner

Department of Mathematics and Statistics, Dalhousie University

Approved: Dr. Walt Finden, Supervisory Committee Member

Department of Mathematics and Computing Science

Approved: Dr. Stavros Konstantinidis, Supervisory Committee Member

Department of Mathematics and Computing Science

Date: April 18, 2017

# Table of Contents

## List of Tables

## Acknowledgements

First of all, I would like to express my appreciation to my supervisor, Dr. Paul Muir, for his valuable guidance and encouragement during the work on this thesis. I would also like to extend my gratitude to the members of my supervisory committee, Dr. Walt Finden and Dr. Stavros Konstantinidis, and the external examiner, Dr. David Iron.

Finally, I would also like to thank my husband for supporting me throughout my research. My thanks, also go to my parents, my sisters, and my brothers, for their unwavering support throughout my studies.

## Abstract

Generalized Mono-Implicit Runge-Kutta Methods for Stiff Ordinary Differential Equations

by

Fatima Dow

Ordinary differential equations (ODEs) arise in many applications. Typically these ODEs are sufficiently complicated that they must be solved using numerical methods. One of the well-known classes of numerical methods for ODEs is the class of Mono-Implicit Runge-Kutta (MIRK) methods. An important property of a MIRK method is its order; a method is of order $p$ if its global error is $O(h^p)$. An issue with MIRK methods, when applied to certain ODEs, known as stiff ODEs, is that when they should be of order $p$, they perform as if their order is $q$, where $q < p$. This is called order reduction. This means that the MIRK methods will be inefficient when the ODE is stiff because the amount of computation that is performed is not consistent with the accuracy obtained.

In this thesis, we derive generalizations of MIRK methods that can avoid order reduction when the ODE is stiff.

Date: April 18, 2017

# Chapter 1

# Introduction

Mathematical models play an important role in the analysis of real world problems that arise in many branches of science, engineering, and economics; see, e.g., [5]. Often these mathematical models involve ODEs which describe how a system changes with time. These ODEs are typically too difficult to solve by hand. Therefore, it is common to obtain approximate solutions by means of numerical methods. In this thesis, we focus on a subclass of the well-known implicit-Runge-Kutta methods [6], called mono-implicit Runge-Kutta (MIRK) methods, [8], ]9], which have been used to compute numerical solutions of ODEs. During the process of solving the ODE these methods partition the problem domain using a step size sequence with steps of size $h$ for initial value ODEs (explained later in this chapter) or a mesh of subintervals each of size $h$ for boundary value ODEs (also explained later in this chapter).

An important property of a MIRK method is its order; a MIRK method is of order $p$ if its global error behaves like $\mathrm{O}(h^p)$. (The global error of a point $t$ is the difference between the numerical solution approximation at $t$ and the exact solution at $t$.) Unfortunately, when a MIRK method is applied to certain ODEs known as stiff ODEs, the method suffers from order reduction. Stiffness (explained later in

this chapter) is a property of some differential equations which causes the observed accuracy of some numerical methods to be unrelated to the theoretical order of the method [16], [22]. This means that a $p$th order MIRK method will perform as if it were a method of order $q$, where $q < p$. (In [16], it is shown that $q$ is the stage order of the method; stage order will be defined in Chapter 2.) This leads to these methods being inefficient for the numerical solution of stiff ODEs since the amount of computation performed is that of a $p$th order method, while the achieved accuracy is that of a lower order method. The lower order accuracy could be achieved using a method of order $q$ that requires less computational effort.

The main purpose of this thesis is to describe the development of a family of generalized MIRK methods for use with stiff ODEs. These new methods do not suffer from order reduction when applied to stiff ODEs.

This thesis is organized as follows. In this chapter, a summary of the types of differential equations we are interested in is provided; we also discuss stiffness. Chapter 2 first describes Runge-Kutta methods. This chapter also discusses MIRK methods. The order and stage order for a MIRK method are explained, and we give descriptions of a number of families of MIRK methods having orders 1 through 6. Next, in Chapter 3, numerical experiments on non-stiff ODEs and stiff ODEs are presented to demonstrate order reduction for standard MIRK methods. Chapter 4 considers our proposed solution to the issue of order reduction for MIRK methods, which leads us to introduce generalized MIRK methods; these are methods in which some of the

stages are generalized to allow for an increase in the stage order of the methods so that they will not suffer from order reduction when applied to stiff ODEs. In Chapter 5, we give numerical results from some computational experiments, where we compare the standard MIRK methods with the new generalized MIRK methods. These comparisons suggest that generalized MIRK methods are good candidates for use in the numerical solution of stiff ODEs since they do not show evidence of order reduction when applied to stiff ODEs . Finally, Chapter 6 gives the conclusions from this thesis and suggestions for future work.

## 1.1 Ordinary Differential Equations

An ordinary differential equation is an equation that includes a function of one independent variable $t$ (e.g., time or space) and derivatives of this function with respect to $t$ [2]. In this thesis, we assume a system of ODEs having the general form

$$y'(t) = f(t, y(t)), \quad y : R \to R^n, \quad f : R \times R^n \to R^n, \tag{1.1}$$

where $y$ and $f$ are vector functions of size $n$.

As mentioned earlier, ODEs occur in many applications. These include, for example, computational models of the human heart, numerical weather forecasting, and predicting the spread of viruses; see, e.g, [1].

### 1.1.1 Initial Value ODEs (IVODEs)

We look for a solution to (1.1) for $t \geq a$ with

$$y(a) = y_0, \tag{1.2}$$

where $a$ is the initial value of $t$ and $y_0 \in R^n$ is the known solution at $t = a$. The first order system of differential equations (1.1) with the initial condition described in (1.2) is known as an initial value ODE (IVODE); see, e.g, [4].

### 1.1.2 Boundary Value ODEs (BVODEs)

Boundary value ODEs are systems of differential equations for which the values of the solution components are specified at two distinct points [1]. We are interested in the BVODEs written in first order system form (1.1) with boundary conditions (BCs),

$$g(y(a), y(b)) = 0, \tag{1.3}$$

where $g : R^n \times R^n \to R^n$ is a vector function, and $0 \in R^n$ is the zero vector.

## 1.2 Stiff ODEs

There are various definitions of stiffness in the literature for ODEs. The earliest research on stiffness in differential equations was presented by the two chemists: Curtiss and Hirschfelder [7] in 1952. They gave a definition of stiffness as follows:

"Stiff equations are equations where certain implicit methods perform better, usually tremendously better, than explicit ones". The next important development was by Dahlquist [13] in 1963. Based on his work, he described stiff ODE systems as "Systems containing very fast components as well as slow components".

The chemical reaction of Robertson is one of the most well-known examples that involves stiff ODEs. The equations and initial values are given in [21] by

$$x_1' = -0.04x_1 + 10^4 x_2 x_3, \quad x_1(0) = 1, \tag{1.4}$$

$$x_2' = 0.04x_1 - 3.10^7 x_2^2 - 10^4 x_2 x_3, \quad x_2(0) = 0, \tag{1.5}$$

$$x_3' = 3.10^7 x_2^2, \quad x_3(0) = 0, \tag{1.6}$$

where the reaction rate 0.04 is relatively slow, while the reaction rate $3.10^7$ is very fast.

When an explicit method, such as the explicit Euler method (see (2.4)) is used to solve a stiff ODE, the issue is that the method is forced to take very small steps due to stability issues. This makes the method very inefficient (see, e.g., [4]).

There is a second issue associated with stiff ODEs. Previous work [16], [22] has shown that for stiff ODEs even when we use an implicit method that does not require a step size restriction due to stability issues, the accuracy of the numerical method appears to be unrelated to the classical order of the method. This phenomenon is

known as order reduction. In this thesis, we will study MIRK methods that suffer from order reduction when applied to stiff ODEs.

# Chapter 2

# Runge-Kutta Methods

Runge-Kutta (RK) methods are a popular class of methods for the numerical solution of ODEs [4]. RK methods can be divided into two major subclasses. The first is the Explicit Runge-Kutta (ERK) methods which were first developed for the solution of IVODEs by Runge, with further development by Heun and Kutta; see, e.g., [4]. The second major subclass is the Implicit Runge-Kutta (IRK) methods which were presented in [6] for the use in the numerical solution of IVODEs. These methods have also been described in [26] for the use in the numerical solution of BVODEs.

## 2.1 RK methods for IVODEs

A Runge-Kutta (RK) method can be used to obtain an approximation to the solution of an IVODE of the form

$$y'(t) = f(t, y(t)), \quad y(t_0) = y_0. \tag{2.1}$$

The RK method is used in a stepwise fashion, starting from the initial point, $t_0$, and passing through a sequence of points, $t_i$, with computed solution approximations, $y_i$, and ending at the required end point, $t_{end}$.

## 2.2 Explicit Runge-Kutta (ERK) methods

The family of $s$-stage Explicit Runge-Kutta (ERK) methods is defined as follows. Given a numerical solution approximation, $y_i$, at a point $t_i$, a numerical approximation $y_{i+1}$ at $t_{i+1}$ is given by

$$y_{i+1} = y_i + h \sum_{r=1}^{s} b_r k_r, \tag{2.2}$$

where

$$k_r = f\left(t_i + c_r h, y_i + h \sum_{j=1}^{s-1} a_{rj} k_j\right), \quad r = 1, 2, \ldots, s, \tag{2.3}$$

where $h = t_{i+1} - t_i$, and each stage $k_r$ depends only on previous stages, which makes the calculation of the stages a simple process. The coefficients $c_r, a_{rj}, b_r$ for an ERK method can be represented using a Butcher Tableau [4]:

$$\begin{array}{c|ccccc} 0 & 0 & 0 & \ldots & \ldots & 0 \\ c_2 & a_{21} & 0 & \ldots & \ldots & 0 \\ \vdots & \vdots & \ddots & & & \vdots \\ \vdots & \vdots & & \ddots & & \vdots \\ c_s & a_{s1} & a_{s2} & \ldots & a_{s,s-1} & 0 \\ \hline & b_1 & b_2 & \ldots & \ldots & b_s \end{array} .$$

The simplest ERK method is the (Explicit) Euler method. The formula for this method is

$$y_{i+1} = y_i + h f(t_i, y_i), \tag{2.4}$$

and the corresponding tableau is

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}.$$

The explicit Euler method is of first order accuracy; this means that its global error is $\mathrm{O}(h^1)$.

Another example of an ERK method is the classical 4-stage, 4th Order, stage order 1 Runge-Kutta method see, e.g., [4]. This has the form

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{2.5}$$

where

$$\begin{aligned} k_1 &= f(t_i, y_i), \\ k_2 &= f(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_1), \\ k_3 &= f(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_2), \\ k_4 &= f(t_i + h, y_i + hk_3). \end{aligned} \tag{2.6}$$

Its tableau is

$$\begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array} .$$

This method is fourth order; this means that its global error is $\mathrm{O}(h^4)$.

The coefficients of a RK method are chosen to satisfy certain conditions, known as order conditions, which determine the classical order of the method. We explain order conditions later in this chapter.

## 2.3 Implicit Runge-Kutta (IRK) methods

For an implicit RK (IRK) method, during the $i$th step, an approximation, $y_{i+1}$, to the exact solution, $y(t)$, evaluated at the point $t_{i+1} = t_i + h$, has the form

$$y_{i+1} = y_i + h \sum_{r=1}^{s} b_r k_r, \tag{2.7}$$

where

$$k_r = f\left(t_i + c_r h, y_i + h \sum_{j=1}^{s} a_{rj} k_j\right), \quad r = 1, 2, \ldots, s. \tag{2.8}$$

The coefficients of this method are given in a Butcher tableau of the form

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \dots & a_{1s} \\ c_2 & a_{21} & a_{22} & \dots & a_{2s} \\ \vdots & \vdots & \vdots & & \vdots \\ c_s & a_{s1} & a_{s2} & \dots & a_{ss} \\ \hline & b_1 & b_2 & \dots & b_s \end{array} .$$

The above tableau is sometimes condensed to:

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

where $c = (c_1, c_2, ..., c_s)^T$, $b = (b_1, b_2, ..., b_s)^T$, $A$ is the $s$ by $s$ matrix whose $(i\ ,\ j)$th component is $a_{ij}$. Also, the coefficients are usually required to satisfy $c = Ae$, where $e$ is the vector of l's of length $s$. This is equivalent to requiring $c_r = \sum_{j=1}^{s} a_{rj}$.

For IRK methods, each stage $k_r$ in (2.8) is implicitly defined in terms of itself and the other stages. Therefore, in order to compute values for the stages, it is necessary to solve a nonlinear system of size $n \times s$ where $n$ represents the number of differential equations and $s$ represents the number of stages of the method. The non-linear system for the determination of the stages is often solved using a Newton type iteration [4], which makes the calculation of the stages a computationally expensive process.

## 2.4 Mono-implicit Runge-Kutta methods

A number of subclasses of the IRK methods have been developed and investigated in the literature. One subclass which has been found to be efficient for the numerical solution of ODEs, is the class of mono-implicit Runge-Kutta (MIRK) methods [8], [9]. An important property of this class of methods is that for IVODEs, they are implicit only in the single unknown $y_{i+1}$, and for BVODEs they are explicit in $y_i$ and $y_{i+1}$.

Muir and Enright [19] introduced MIRK formulae as a particular class of parameterized implicit Runge-Kutta methods having the form

$$y_{i+1} = y_i + h \sum_{r=1}^{s} b_r k_r, \tag{2.9}$$

where

$$k_r = f\left(t_i + c_r h, (1 - v_r) y_i + v_r y_{i+1} + h \sum_{j=1}^{r-1} x_{rj} k_j\right), \quad r = 1, 2, \ldots, s. \tag{2.10}$$

The method is defined by the number of stages, $s$, the coefficients, $\{v_r\}_{r=1}^{s}$ and $\{x_{rj}\}_{j=1,r=1}^{r-1,s}$, and the weights $\{b_r\}_{r=1}^{s}$. The abscissa, $\{c_r\}_{r=1}^{s}$, are defined by $c_r = v_r + \sum_{j=1}^{r-1} x_{rj}$, which is equivalent to requiring $c = Xe + v$. The coefficients of a MIRK

method are often given in a tableau of the form,

$$\begin{array}{c|c|ccccc} c_1 & v_1 & 0 & 0 & \dots & \dots & 0 \\ c_2 & v_2 & x_{21} & 0 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & & & \vdots \\ \vdots & \vdots & \vdots & & \ddots & & \vdots \\ c_s & v_s & x_{s1} & x_{s2} & \dots & x_{s,s-1} & 0 \\ \hline & & b_1 & b_2 & \dots & \dots & b_s \end{array} .$$

which is sometimes condensed to the form

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

where $c = (c_1, c_2, ..., c_s)^T$, $v = (v_1, v_2, ..., v_s)^T$, $b = (b_1, b_2, ..., b_s)^T$, $X$ is the $s$ by $s$ matrix whose ($i$ , $j$)th component is $x_{ij}$. It can be shown that the MIRK method (2.9), (2.10) is equivalent to the general IRK method (2.7), (2.8), with $A$ defined as $A = X + vb^T$ [5].

For MIRK methods, all stages $k_r$, (2.10), are implicit only in $y_{i+1}$. The computation of $y_{i+1}$ can thus be reduced to the numerical solution of one system of only $n$ nonlinear equations. This means that the computation on each step, based on the use of MIRK method, is less costly than that based on IRK methods. The trade off, however, is that the maximum order of an implicit $s$-stage MIRK method is lower than that of

an $s$-stage IRK method for $s \geq 2$.

### 2.4.1 Order conditions for MIRK methods

As mentioned earlier, a MIRK method is of order $p$ if the numerical solution of the ODE, obtained by solving (2.9), satisfies $|y(t_{i+1}) - y_{i+1}| = \mathrm{O}(h^p)$, where $y(t_i)$ is the exact solution of (1.1) evaluated at $t_i$ and $y_{i+1}$ is the numerical solution approximation at $t_{i+1}$. MIRK methods that appear in the literature typically have $p$ in the range from 1 to 8. A family of MIRK methods of a particular order $p$ is derived by requiring its coefficients to satisfy a set of equations called order conditions. Since MIRK methods can be shown as IRK methods [19], the order conditions for MIRK methods are similar to those for IRK methods [4].

For example, an IRK method having order $p = 1$, with the tableau

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

must satisfy

$$b^T e = 1. \tag{2.11}$$

The order condition for second order is

$$b^T c = \frac{1}{2}. \tag{2.12}$$

An IRK method must satisfy (2.11) and (2.12) in order to be second order.

The order conditions for third order are

$$b^T c^2 = \frac{1}{3}, \quad b^T Ac = \frac{1}{6}, \tag{2.13}$$

where $c^j = (c_1^j, c_2^j, ..., c_s^j)^T$. An IRK method must satisfy (2.11), (2.12), and (2.13) in order to be third order.

The order conditions for fourth order are

$$b^T c^3 = \frac{1}{4}, \quad b^T cAc = \frac{1}{8}, \quad b^T Ac^2 = \frac{1}{12}, \quad b^T A^2 c = \frac{1}{24}. \tag{2.14}$$

So, if an IRK method has coefficients that satisfy all conditions (2.11), (2.12), (2.13), and (2.14), then it is a fourth order method.

For MIRK methods, the order conditions up to order 4 are $b^T e = 1$, $b^T c = \frac{1}{2}$, $b^T c^2 = \frac{1}{3}$, $b^T (Xc + \frac{v}{2}) = \frac{1}{6}$, $b^T c^3 = \frac{1}{4}$, $b^T c(Xc + \frac{v}{2}) = \frac{1}{8}$, $b^T (Xc^2 + \frac{v}{3}) = \frac{1}{12}$, $b^T (X(Xc + \frac{v}{2}) + \frac{v}{6}) = \frac{1}{24}$ [16].

Table 2.1: Number of order conditions for MIRK methods of orders $p = 1, ..., 8$.

| $p$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| number of order conditions | 1 | 2 | 4 | 8 | 17 | 37 | 85 | 200 |

Table (2.1) shows that the number of order conditions increases rapidly with increasing order.

### 2.4.2 Stage order conditions for MIRK methods

An additional set of conditions that can be imposed on the coefficients of a MIRK method are the stage order conditions. These are not required but it is often helpful to impose some of these conditions because it turns out that imposing stage order conditions will reduce the number of order conditions that must be satisfied [20]. A MIRK method is of stage order $q$ if its coefficients satisfy the stage order conditions up to stage order $q$

$$Xc^{j-1} + \frac{v}{j} = \frac{c^j}{j}, \quad j = 1, \ldots, q, \tag{2.15}$$

where $c^0 = e$ [20].

**Theorem** : (i) a MIRK method having stage order 2 must have $c_1 = 0$ or 1, (ii) a MIRK method, with at least 2 stages and having stage order 3, must have $x_{2,1} = 0$ and either $c_1 = 0; c_2 = 1$ or (equivalently) $c_1 = 1; c_2 = 0$, and (iii) the maximum stage-order of an $s$ stage MIRK method is $\min(s, 3)$.

A proof of this theorem can be found in [5]. According to this theorem, the maximum stage order for a $p$th order MIRK method is $q = 3$. This means, it is impossible to derive a MIRK method of stage order greater than 3.

In the following, we will show examples of MIRK methods of different orders.

## 2.5 A MIRK method of order 1

An example of a first order, 1-stage, stage order 1 MIRK method is the explicit Euler method (2.4). The general form of the tableau is given in (2.16); the explicit Euler method has $c_1 = v_1 = 0$ and $b_1 = 1$.

$$\begin{array}{c|c|c} c_1 & v_1 & 0 \\ \hline & & b_1 \end{array}. \tag{2.16}$$

This class of methods also includes the first order implicit Euler method, obtained by choosing $c_1 = v_1 = 1$ and $b_1 = 1$; this method has stage order 1.

## 2.6 A MIRK method of order 2

It is also possible to obtain a 1-stage, second order method by choosing $c_1$, $v_1$, $b_1$ appropriately. The only order 2 MIRK method with 1-stage is the mid-point method [20]. This method has stage order 1 and the tableau of this method is

$$\begin{array}{c|c|c} \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & & 1 \end{array}. \tag{2.17}$$

An example of a MIRK method of order 2 with 2 stages and having stage order 2 is

the trapezoidal method [5], which has the following tableau

$$\begin{array}{c|c|cc} 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ \hline & & \frac{1}{2} & \frac{1}{2} \end{array} . \tag{2.18}$$

## 2.7 A MIRK method of order 3

An example of a two stage, third order, stage order two MIRK method is given in [5]; its tableau is

$$\begin{array}{c|c|cc} 1 & 1 & 0 & 0 \\ \frac{1}{3} & \frac{5}{9} & \frac{-2}{9} & 0 \\ \hline & & \frac{1}{4} & \frac{3}{4} \end{array} . \tag{2.19}$$

## 2.8 A MIRK method of order 4

The unique three stage, fourth order, stage order three MIRK method is given in [5]; its tableau is

$$\begin{array}{c|c|ccc} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{8} & \frac{-1}{8} & 0 \\ \hline & & \frac{1}{6} & \frac{1}{6} & \frac{2}{3} \end{array} . \tag{2.20}$$

## 2.9 A MIRK method of order 5

An example of a four stage, fifth order, stage order three MIRK method is [5]

$$\begin{array}{c|c|cccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{5}{32} & \frac{9}{64} & \frac{-3}{64} & 0 & 0 \\ \frac{7}{10} & \frac{413}{1250} & \frac{-63}{5000} & \frac{-21}{1000} & \frac{252}{625} & 0 \\ \hline & & \frac{1}{14} & \frac{5}{54} & \frac{32}{81} & \frac{250}{567} \end{array} . \tag{2.21}$$

## 2.10 A MIRK method of order 6

An example of a five stage, sixth order MIRK method is given in [20], where the stage order has the maximum value of three:

$$\begin{array}{c|c|ccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2}-\frac{\sqrt{21}}{14} & \frac{1}{2}-\frac{9\sqrt{21}}{98} & \frac{1}{14}+\frac{\sqrt{21}}{98} & -\frac{1}{14}+\frac{\sqrt{21}}{98} & 0 & 0 & 0 \\ \frac{1}{2}+\frac{\sqrt{21}}{14} & \frac{1}{2}+\frac{9\sqrt{21}}{98} & \frac{1}{14}-\frac{\sqrt{21}}{98} & -\frac{1}{14}-\frac{\sqrt{21}}{98} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & -\frac{5}{128} & \frac{5}{128} & \frac{7\sqrt{21}}{128} & -\frac{7\sqrt{21}}{128} & 0 \\ \hline & & \frac{1}{20} & \frac{1}{20} & \frac{49}{180} & \frac{49}{180} & \frac{16}{45} \end{array} . \tag{2.22}$$

## 2.11 RK methods for BVODEs

In the BVODE context, given a mesh which subdivides the problem interval, the discrete system, $\Phi(Y) = 0$, consisting of the boundary conditions, $g(y_0, y_N) = 0$, and $n$ more equations per subinterval, can be solved with a Newton iteration to obtain a discrete numerical solution $Y$ having the form $Y = [y_0, y_1, ..., y_N]^T$, where $N$ is the number of subintervals. When the RK method is applied as the discretization method, the set of number $n$ equations associated with the $i$th subinterval has the form

$$\Phi_i(y_i, y_{i+1}) = y_{i+1} - y_i - h\sum_{r=1}^{s} b_r k_r = 0, \tag{2.23}$$

with the stages, $k_r$, defined as in (2.8).

The use of MIRK methods for the numerical solution of BVODEs has been described in [10,11,14,15,17]. When a MIRK method is used for the solution of a BVODE, the set of equations associated with the $ith$ subinterval has the same form as in (2.23), but the corresponding stages are of the mono-implicit type as in (2.10). Since approximations to both $y_i$ and $y_{i+1}$ can be obtained from the current Newton iterate, the stages of the MIRK method can be computed explicitly [14], which means these methods have approximately the same computation cost on each subinterval as the ERK methods do for each step when they are applied to an IVODE.

## 2.12 Order reduction

Order reduction arises when the stiffness of a problem causes a numerical method to suffer a decrease in its order, rather than having the expected classical order associated with the order conditions its coefficients satisfy. In 1974, Prothero and Robinson [22] showed that when certain numerical methods were applied to solve stiff problems, the order that these methods exhibited was not the expected order, but rather was equal to their stage order. Further analysis was provided in [16] where it was shown that the order of the method can be reduced to the stage order or the stage order plus 1, depending on certain characteristics of the problem. Moreover, it was noted by Voss and Muir [25] that the implicit Runge-Kutta methods are affected by the phenomena of order reduction when applied to stiff ODEs; regardless of its classical order, the method will behave as if its order is only its stage order.

Thus, it can be important for a method to have as high a stage order as possible. However, in [5] it is proved that while it is possible to derive a MIRK method of any classical order, the method can have at most stage order 3 (see Subsection 2.4.2). This means that for stiff ODEs, an order reduction phenomenon can cause a MIRK method having stage order 3 to behave if its order is only 3, independent of its classical order. This implies that the MIRK method will be inefficient for stiff ODEs because extra computation will be done without obtaining extra accuracy.

In 2014, Chen [12] has extended the tableau of different types of IRK methods to obtain methods with higher stage order. Such methods do not suffer from order

reduction when applied to stiff problems. This paper is relevant to this thesis because we will similarly extend the general form for a MIRK method to obtain a method with a higher stage order that will not suffer from order reduction when applied to a stiff ODE.

# Chapter 3

# Order Reduction Experiments for IVODEs and for BVODEs

## 3.1 Numerical Experiments for IVODEs

Numerical experiments are presented here to demonstrate order reduction for MIRK methods of different orders. For IVODEs, a MIRK method (2.9) with stages (2.10), computes approximate solution values $y_i \approx y(t_i)$ in a step-wise fashion (with step size $h$) starting with initial value $t_0 = a$ and the solution $y_0$ at $t_0 = a$. Then, using the MIRK method, the next approximation $y_{i+1}$ is computed at $t_{i+1} = t_i + h$, proceeding across the domain to the required final $t$ value. Since the stages are implicit in $y_{i+1}$, each step requires the solution of a system of $n$ non-linear equations at the end of each step. The absolute global error is computed by taking the difference between the known analytical solution and the approximate solution. The maximum error is the largest error over all steps. We then repeat the computation with the step size reduced by a half; if the method is of order $p$ the error will be reduced by $(1/2)^p$. Then, the observed order of the MIRK method is obtained by taking $log_2$ of the error ratio. For IVODEs, the results were obtained by using an implementation within the Scilab programming environment [27]. See Appendix A.

First, we consider the following simple non-stiff IVODE:

$$y'(t) = \lambda y(t), \quad \lambda = -1, \tag{3.1}$$

with initial condition

$$y(0) = 1.$$

Its exact solution is

$$y(t) = e^{-t}.$$

Second, we consider the following example of a stiff IVODE (see [22]). It is defined by the IVODE:

$$y'(t) = g'(t) + \lambda(y(t) - g(t)), \quad y(0) = g(0). \tag{3.2}$$

where

$$g(t) = 10 - (10 + t)e^{-t}, \quad \lambda \in R \quad and \quad \lambda < 0. \tag{3.3}$$

Its analytical solution is $g(t)$ (for any differentiable $g(t)$); this can be seen simply by substituting $y(t) = g(t)$ into (3.2).

### 3.1.1 Numerical Results

In this subsection, we will show numerical results for the non-stiff IVODE (3.1) and the stiff IVODE (3.2) obtained by using standard MIRK methods of various orders. For stiff ODEs, we need to choose $\lambda$ to be sufficiently large magnitude in order to observe order reduction.

Table 3.1: Numerical results for the non-stiff IVODE (3.1) with $\lambda = -1$ for the 2-stage, third order, stage order 2 MIRK method (2.19).

| $h$ | maxerr | error ratio | observed order |
|---|---|---|---|
| 0.1 | 0.0000050 | | |
| 0.05 | 0.0000006 | 7.8985544 | 2.9815886 |
| 0.025 | 7.931D-08 | 7.9479749 | 2.9905873 |

Table (3.1) shows that when we applied the 2-stage, 3rd order, stage order 2 MIRK method (2.19) to the non-stiff IVODE (3.1) with $\lambda = -1$, and reduced the step size by half, the error dropped by a factor of approximately 1/8. This means that the method has 3rd order, which means that there is no order reduction.

Table 3.2: Numerical results for the stiff IVODE (3.2) with $\lambda = -150$ for the 2-stage, third order, stage order 2 MIRK method (2.19).

| $h$ | maxerr | error ratio | observed order |
|---|---|---|---|
| 0.2 | 0.0001645 | | |
| 0.1 | 0.0000381 | 4.3179637 | 2.1103511 |
| 0.05 | 0.0000077 | 4.9472846 | 2.3066369 |

Table (3.2) shows that when we applied the 2-stage, 3rd order, stage order 2 MIRK method (2.19) to the stiff IVODE (3.2) with $\lambda = -150$, the order of the 2-stage, 3rd order, stage order 2 MIRK method drops from its classical order, 3, to its stage order, 2. That is, the error ratio drops to approximately 1/4. This illustrates that the method suffers from order reduction.

Table 3.3: Numerical results for the non-stiff IVODE (3.1) with $\lambda = -1$ for the 3-stage, fourth order, stage order 3 MIRK method (2.20).

| $h$ | maxerr | error ratio | observed order |
|---|---|---|---|
| 0.2 | 0.0000008 | | |
| 0.1 | 5.112D-08 | 16.028571 | 4.0025739 |
| 0.05 | 3.194D-09 | 16.007143 | 4.0006439 |

Table (3.3) shows that the 3-stage, 4th order, stage order 3 MIRK method (2.20) does not suffer from order reduction when applied to the non-stiff IVODE (3.1) with $\lambda = -1$, because it has its classical order, which is 4.

Table 3.4: Numerical results for the stiff IVODE (3.2) with $\lambda = -5000$ for the 3-stage, fourth order, stage order 3 MIRK method (2.20).

| $h$ | maxerr | error ratio | observed order |
|---|---|---|---|
| 0.1 | 0.0000002 | | |
| 0.05 | 2.553D-08 | 7.0152226 | 2.8104889 |
| 0.025 | 2.660D-09 | 9.5955363 | 3.2623634 |

Table (3.4) shows that for the stiff IVODE (3.2) with $\lambda = -5000$, the 3 stage, 4th order, stage order 3 MIRK method (2.20) drops from its classical order 4, to its stage order 3. This shows that the method suffers from order reduction.

Table 3.5: Numerical results for the non-stiff IVODE (3.1) with $\lambda = -1$ for the 4-stage, fifth order, stage order 3 MIRK method (2.21).

| $h$ | maxerr | error ratio | observed order |
|---|---|---|---|
| 0.25 | 0.0000002 | | |
| 0.125 | 4.878D-09 | 33.460711 | 5.0643962 |
| 0.0625 | 1.492D-10 | 32.687957 | 5.0306873 |

Table (3.5) shows that the 4-stage, 5th order, stage order 3 MIRK method (2.21) does not suffer from order reduction when applied to the non-stiff IVODE (3.1) with $\lambda = -1$, because it retains its classical order, which is 5.

Table 3.6: Numerical results for the stiff IVODE (3.2) with $\lambda = -55$ for the 4-stage, fifth order, stage order 3 MIRK method (2.21).

| $h$ | maxerr | error ratio | observed order |
|---|---|---|---|
| 0.25 | 0.0001151 | | |
| 0.125 | 0.0000108 | 10.66354 | 3.4146146 |
| 0.0625 | 0.0000012 | 9.2841103 | 3.2147637 |

Table (3.6) shows that for the stiff IVODE (3.2) with $\lambda = -55$, the 4-stage, 5th order, stage order 3 MIRK method (2.21) drops from its classical order 5, to its stage order 3. This implies that the method suffers from order reduction.

Table 3.7: Numerical results for the non-stiff IVODE (3.1) with $\lambda = -1$ for the 5-stage, sixth order, stage order 3 MIRK method (2.22).

| $h$ | maxerr | error ratio | observed order |
|---|---|---|---|
| 0.1 | 3.651D-12 | | |
| 0.05 | 5.690D-14 | 64.167805 | 6.0037777 |

Table (3.7) shows that the 5-stage, 6th order, stage order 3 MIRK method (2.22) does not suffer from order reduction when applied to the non-stiff IVODE (3.1) with $\lambda = -1$, because it retains its classical order, which is 6.

Table 3.8: Numerical results for the stiff IVODE (3.2) with $\lambda = -5000$ for the 5-stage, sixth order, stage order 3 MIRK method (2.22).

| $h$ | maxerr | error ratio | observed order |
|---|---|---|---|
| 0.012 | 2.021D-14 | | |
| 0.006 | 2.665D-15 | 7.5833333 | 2.9228321 |

Table (3.8) shows that, for the stiff IVODE (3.2) with $\lambda = -5000$, the 5-stage, 6th order, stage order 3 MIRK method (2.22) drops from its classical order 6, to its stage order 3. This shows that the method suffers from order reduction.

### 3.2 Numerical experiments for BVODEs

For BVODEs, a MIRK method (2.23) with stages (2.10), computes approximate solution values $y_i$, $i = 0, 1, ..., N$, at the points of an $(N + 1)$-point mesh which subdivides the problem interval. Since the stages are explicit in $y_i$ and $y_{i+1}$ (assuming the availability of the approximations $y_i$, $i = 0, 1, ..., N$, from the current Newton iterate $Y^{(i)}$), this requires only the solution of a system of $n(N + 1)$ non-linear equations where $n$ is the number of ODEs and $N$ is the number of subintervals. The results were obtained by using an implementation within the Scilab programming environment [27]. See Appendix A.

We consider the following example of a BVODE [1]:

$$y_1'(t) = \lambda y_2(t), \tag{3.4}$$

$$y_2'(t) = \lambda y_1(t) + \lambda \cos(\pi t)^2 + 2/\lambda \pi^2 \cos(2t\pi),$$

where the boundary conditions are

$$y_1(0) = 0,$$

$$y_2(1) = 0.$$

Its exact solution is

$$y_1(t) = \frac{e^{\lambda(t-1)} + e^{-\lambda t}}{1 + e^{-\lambda}} - \cos^2(\pi t),$$

$$y_2(t) = \frac{e^{\lambda(t-1)} - e^{-\lambda t}}{1 + e^{-\lambda}} + \frac{\pi}{\lambda}\sin(2\pi t).$$

Note that a BVODE (3.4) is only stiff if we choose $\lambda$ to have a large enough negative value; e.g. $\lambda = -1$ will not give a stiff BVODE.

### 3.3 Numerical Results for BVODEs

In this section, we will show numerical results for the BVODE (3.4) obtained by using the 3-stage, 4th order, stage order 3 MIRK method (2.20) and the 5-stage, 6th order, stage order 3 MIRK method (2.22).

Table 3.9: Numerical results for the non-stiff BVODE (3.4) with $\lambda = -1$ for the 3-stage, 4th order, stage order 3 MIRK method (2.20).

| $N$ | $h$ | maxerr1 | maxerr2 | observed order1 | observed order2 |
|---|---|---|---|---|---|
| 52 | 1/52 | 0.0000002 | 0.0000003 | | |
| 104 | 1/104 | 1.223D-08 | 1.889D-08 | 4.0009056 | 3.9984742 |

For the non-stiff BVODE (3.4) with $\lambda = -1$, the 3-stage, 4th order, stage order 3 MIRK method (2.20) does not suffer from order reduction because it has its classical order of 4.

Table 3.10: Numerical results for the stiff BVODE (3.4) with $\lambda = -150$ for the 3-stage, 4th order, stage order 3 MIRK method (2.20).

| $N$ | $h$ | maxerr1 | maxerr2 | observed order1 | observed order2 |
|---|---|---|---|---|---|
| 52 | 1/52 | 0.0242038 | 0.0242039 | | |
| 104 | 1/104 | 0.0023085 | 0.0023085 | 3.3901937 | 3.3901973 |

For the stiff BVODE (3.4) with $\lambda = -150$, the 3-stage, 4th order, stage order 3 MIRK method (2.20) drops from its classical order 4, to its stage order 3. This shows

that the method suffers from order reduction.

Table 3.11: Numerical results for the stiff BVODE (3.4) with $\lambda = -1$ for the 5-stage, 6th order, stage order 3 MIRK method (2.22).

| $N$ | $h$ | maxerr1 | maxerr2 | observed order1 | observed order2 |
|---|---|---|---|---|---|
| 19 | 1/19 | 9.141D-10 | 5.989D-10 | | |
| 38 | 1/38 | 1.424D-11 | 9.445D-12 | 5.9867628 | 6.0043873 |

For the non-stiff BVODE (3.4) with $\lambda = -1$, the 5-stage, 6th order, stage order 3 MIRK method (2.22) does not suffer from order reduction because it retains its classical order of 6.

Table 3.12: Numerical results for the stiff BVODE (3.4) with $\lambda = -750$ for the 5-stage, 6th order, stage order 3 MIRK method (2.22).

| $N$ | $h$ | maxerr1 | maxerr2 | observed order1 | observed order2 |
|---|---|---|---|---|---|
| 19 | 1/19 | 0.2968541 | 0.2969199 | | |
| 38 | 1/38 | 0.0265662 | 0.0265662 | 3.4820928 | 3.4824125 |

For the stiff BVODE (3.4) with $\lambda = -750$, the 5-stage, 6th order, stage order 3 MIRK method (2.22) drops from its classical order 6, to its stage order 3. This shows that the method suffers from order reduction.

### 3.4 Summary

MIRK methods of different orders are applied to solve non-stiff and stiff IVODEs and BVODE. We experimentally observed that when a classical $p$th order MIRK method is applied to the non-stiff IVODE (3.1) with $\lambda = -1$ and the non-stiff BVODE (3.4) with $\lambda = -1$ the order of the MIRK method was as expected; that is there is no

order reduction. However, when we applied the same method to the stiff IVODE (3.2) and the stiff BVODE (3.4) with $\lambda << -1$, we found that the order of the method drops from its classical order to its stage order. When the order of a method drops from its classical order, there is wasted computation; it would be better to use a lower order method that does not suffer from order reduction.

Order reduction for RK methods applied to stiff IVODEs has been observed in the literature; see, e.g., [12, 16, 22, 25]. To our knowledge, the order reduction phenomenon for RK methods applied to stiff BVODEs has not been reported in the literature.

# Chapter 4

## Addressing the Order Reduction Issue for MIRK Methods

As mentioned earlier, although one can derive a MIRK method of any desired order, the resultant method can have at most stage order 3 [5]. Thus, when a MIRK method is applied to a stiff ODE, for example, even a 6th order method will behave like a 3rd order method due to the order reduction phenomenon.

In this chapter, we will start with the general form for a MIRK method; such a general form limits the method to having at most stage order 3. We will then extend this general form to allow us to derive generalizations of MIRK methods that have a higher stage order.

This extended general form will allow us to increase the number of coefficients associated with certain stages of the method that limit its stage order. We will first apply the appropriate stage order conditions in order to ensure that the method has the desired higher stage order. Then, we will derive a specific method by forcing the remaining free coefficients to satisfy the desired order conditions. As mentioned earlier, imposing high stage order leads to a reduction in the number of order conditions that must be satisfied in order to obtain a method of a desired classical order. This will determine specific values for the coefficients of the method allowing us to derive

a specific generalized MIRK method with higher stage order. We expect that such a method will not suffer from order reduction when applied to stiff ODEs.

For IVODEs, the higher stage order is obtained by allowing some of the stages to be implicitly defined in terms of themselves and some of the other stages. This is in addition to the dependence these stages already have on $y_{i+1}$, the unknown solution at the right hand end of the step. This means that in order to compute these implicit stages and $y_{i+1}$ in order to increase the stage order, the size of the non-linear system that must be solved, will increase. Instead of needing to solve a non-linear system of size $n$ where $n$ is the number of ODEs, the computation will require the solution of a non-linear system of size $n(l+1)$ where $l$ is the number of stages that are implicitly defined in terms of each other.

When we impose a stage order that is equal to the classical order of the method, these new methods will retain the desired classical order even for stiff IVODEs and we can then take larger steps to achieve a desired accuracy. But the amount of work per step will increase.

For BVODEs, when we introduce a number of implicit stages, in order to increase the stage order the size of the non-linear system, that must be solved, will increase. Rather then needing to solve a non-linear system of size $n(N+1)$, the computation will involve the solution of a non-linear system of size $n(N+1)+l\cdot n\cdot N$ where $n$ is the number of ODEs, $N$ is the number of subintervals, and $l$ is the number of stages that are implicitly defined in terms of each other.

Since all first order MIRK methods (2.16) have stage order 1, no order reduction is possible. It is possible for a second order MIRK method to have only stage order 1 (the midpoint method (2.17)) and in that case order reduction would be a concern. However, the standard Trapezoidal method (2.18) is of second order and has stage order two and will not suffer from order reduction. In the following, we will derive generalization of MIRK methods of orders 3, 4, 5, and 6. The Maple programming environment [28] is employed to derive generalized MIRK methods. See Appendix B.

## 4.1 A 2-stage, 3rd order, stage order 3 Generalized MIRK method is not possible

We first consider a generalization of a two stage, third order, stage order two MIRK method, given in (2.19), to a two stage, third order, stage order three generalized MIRK method. Its tableau is

$$\begin{array}{c|c|cc} 0 & 0 & 0 & 0 \\ c_2 & v_2 & x_{21} & x_{22} \\ \hline & & b_1 & b_2 \end{array} . \tag{4.1}$$

This implies that $k_2 = f\left(t_i + c_2 h, (1 - v_2)y_i + v_2 y_{i+1} + h(x_{21}k_1 + x_{22}k_2)\right)$. Note that $k_2$ now depends implicitly on itself and on $y_{i+1}$.

The stage order three conditions are

$$Xe + v = c, \quad Xc + \frac{v}{2} = \frac{c^2}{2}, \quad Xc^2 + \frac{v}{3} = \frac{c^3}{3}. \tag{4.2}$$

Applying these conditions to (4.1), we get $v_2$, $x_{21}$, $x_{22}$ in terms of $c_2$ with $c_2 \neq \frac{2}{3}$. Then, we apply the third order conditions, which are $b^T e = 1$, $b^T c = \frac{1}{2}$, $b^T c^2 = \frac{1}{3}$. However in attempting to satisfy these conditions, we find that $c_2$ must be $\frac{2}{3}$. This proves that we cannot have a method with the tableau (4.1) being third order, and having stage order three.

## 4.2 A 3-stage, 3rd order, stage order 3 MIRK method

Since, it is not possible to obtain a 2-stage, third order, stage order 3 generalized MIRK method, we turn to a 3-stage, third order, stage order 3 MIRK method, for which the tableau is

$$\begin{array}{c|c|ccc} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ c_3 & v_3 & x_{31} & x_{32} & 0 \\ \hline & & b_1 & b_2 & b_3 \end{array} . \tag{4.3}$$

The application of the stage order 3 conditions (4.2), and the third order conditions, which for a stage order 3 method are $b^T e = 1$, $b^T c = \frac{1}{2}$, $b^T c^2 = \frac{1}{3}$, leads to one free

parameter $c_3$ with $c_3 \neq 0, 1$. An example with $c_3 = \frac{1}{3}$, is the method whose tableau is

$$\begin{array}{c|c|ccc} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{7}{27} & \frac{4}{27} & \frac{-2}{27} & 0 \\ \hline & & 0 & \frac{1}{4} & \frac{3}{4} \end{array} . \tag{4.4}$$

Thus it is possible to obtain a 3-stage, 3rd order, stage order 3 MIRK method: there is no need to employ a generalized MIRK method.

## 4.3 A 3-stage, 4th order, stage order 4 Generalized MIRK method is not possible

We first consider a generalization of the three stage, fourth order, stage order three MIRK method given in (2.20), to a three stage, fourth order, stage order four generalized MIRK method. For this method, we have the tableau:

$$\begin{array}{c|c|ccc} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ c_3 & v_3 & x_{31} & x_{32} & x_{33} \\ \hline & & b_1 & b_2 & b_3 \end{array} . \tag{4.5}$$

We need to introduce $x_{33}$ in order to try to have $k_3$ have stage order four. Note that $k_3$ now depends implicitly on $y_{i+1}$ and on itself. That is

$$k_3 = f\left(t_i + c_3 h, (1 - v_3)y_i + v_3 y_{i+1} + h(x_{31}k_1 + x_{32}k_2 + x_{33}k_3)\right). \tag{4.6}$$

The stage order four conditions are

$$Xe + v = c, \quad Xc + \frac{v}{2} = \frac{c^2}{2}, \quad Xc^2 + \frac{v}{3} = \frac{c^3}{3}, \quad Xc^3 + \frac{v}{4} = \frac{c^4}{4}. \tag{4.7}$$

Applying these conditions to (4.5), we get $v_3$, $x_{31}$, $x_{32}$, and $x_{33}$ in terms of $c_3$ with $c_3 \neq 1, \frac{1}{2}$. Then, we apply the fourth order conditions, which for a stage order 4 method are $b^T e = 1$, $b^T c = \frac{1}{2}$, $b^T c^2 = \frac{1}{3}$, and $b^T c^3 = \frac{1}{4}$. Unfortunately, we find that the application of these conditions forces $c_3$ to be $\frac{1}{2}$. Thus, we cannot get a method with the tableau (4.5) being fourth order, and having stage order four.

## 4.4 A 4-stage, 4th order, stage order 4 Generalized MIRK method

Since, we cannot have a three stage, fourth order, stage order four Generalized MIRK method, we thus turn to using a four stage, fourth order, stage order four

Generalized MIRK method, for which the tableau is

$$\begin{array}{c|c|cccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ c_3 & v_3 & x_{31} & x_{32} & x_{33} & 0 \\ c_4 & v_4 & x_{41} & x_{42} & x_{43} & 0 \\ \hline & & b_1 & b_2 & b_3 & b_4 \end{array} . \tag{4.8}$$

Note that, the third stage, in addition to being implicit in $y_{i+1}$, is also implicit in itself. However, the fourth stage is implicit only in $y_{i+1}$.

The application of the stage order 4 conditions (4.7), and the fourth order conditions, which for a stage order 4 method are $b^T e = 1$, $b^T c = \frac{1}{2}$, $b^T c^2 = \frac{1}{3}$, and $b^T c^3 = \frac{1}{4}$, leads to a family of methods with two free parameters $c_3 \neq 0, 1, \frac{1}{2}$, or $c_4$, and $c_4 \neq 0, 1$, or $c_3$. An example, with $c_3 = \frac{1}{3}$, and $c_4 = \frac{2}{3}$, is the method whose tableau is

$$\begin{array}{c|c|cccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{-5}{27} & \frac{4}{27} & \frac{1}{27} & \frac{1}{3} & 0 \\ \frac{2}{3} & \frac{8}{27} & \frac{2}{27} & \frac{-1}{27} & \frac{1}{3} & 0 \\ \hline & & \frac{1}{8} & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} \end{array} . \tag{4.9}$$

### 4.5 A 4-stage, 5th order, stage order 4 Generalized MIRK method

In this section, we consider a generalization of the four stage, fifth order, stage order three MIRK method given in (2.21), to a four stage, fifth order, stage order four generalized MIRK method, for which the tableau is

$$\begin{array}{c|c|cccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ c_3 & v_3 & x_{31} & x_{32} & x_{33} & 0 \\ c_4 & v_4 & x_{41} & x_{42} & x_{43} & 0 \\ \hline & & b_1 & b_2 & b_3 & b_4 \end{array} . \tag{4.10}$$

Note that $k_3$ is implicit in itself and in $y_{i+1}$, while $k_4$ is implicit only in $y_{i+1}$.

We assume stage order four (4.7), which allow us to reduce the order conditions for fifth order to $b^T e = 1$, $b^T c = \frac{1}{2}$, $b^T c^2 = \frac{1}{3}$, $b^T c^3 = \frac{1}{4}$, and $b^T c^4 = \frac{1}{5}$. We apply the stage order 4 conditions (4.7) to (4.10) to ensure that the method has the desired higher stage order. We then apply the fifth order conditions. This gives a family of methods with 1 free parameter $c_3$ with the restriction that $c_3 \neq 0, 1$ or $c_4$.

By choosing $c_3 = \frac{1}{3}$, we obtain a generalized four stage, fifth order, stage order four MIRK method, which has the tableau

$$\begin{array}{c|cc|cccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{3} & \frac{-5}{27} & \frac{4}{27} & \frac{1}{27} & \frac{1}{3} & 0 & \cdot \\
\frac{4}{5} & \frac{416}{625} & \frac{4}{125} & \frac{-44}{625} & \frac{108}{625} & 0 & \\
\hline
 & & \frac{5}{48} & \frac{1}{24} & \frac{27}{56} & \frac{125}{336} &
\end{array} \tag{4.11}$$

We note that this method is of order 5 but has stage order four. When applied to a stiff ODE we would expect to see order reduction from order 5 to order 4. This is not as severe as in the MIRK method case where the order of the method would be reduced to order 3.

## 4.6 A 4-stage, 5th order, stage order 5 Generalized MIRK method is not possible

We cannot have a four stage, fifth order, stage order five generalized MIRK method by using the tableau (4.10) because there are not enough free coefficients to satisfy the stage order 5 conditions and the fifth order conditions. Therefore, we generalize (4.10) by introducing $x_{34}$ and $x_{44}$. The tableau is

$$\begin{array}{c|c|cccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ c_3 & v_3 & x_{31} & x_{32} & x_{33} & x_{34} \\ c_4 & v_4 & x_{41} & x_{42} & x_{43} & x_{44} \\ \hline & & b_1 & b_2 & b_3 & b_4 \end{array} . \tag{4.12}$$

Note that $k_3$ and $k_4$ are implicit in $y_{i+1}$, and each other.

Applying the stage order 5 conditions, which are $Xe + v = c$, $Xc + \frac{v}{2} = \frac{c^2}{2}$, $Xc^2 + \frac{v}{3} = \frac{c^3}{3}$, $Xc^3 + \frac{v}{4} = \frac{c^4}{4}$, and $Xc^4 + \frac{v}{5} = \frac{c^5}{5}$, to (4.12), we obtain $v_3$, $x_{31}$, $x_{32}$, $x_{33}$, $x_{34}$, $v_4$, $x_{41}$, $x_{42}$, $x_{43}$, $x_{44}$ in terms of $c_3 \neq 0, 1, c_4$, and $c_4 \neq 0, 1, c_3, \frac{7}{10}$. Then, we apply the order conditions for fifth order, which for stage order 5 methods are $b^T e = 1$, $b^T c = \frac{1}{2}$, $b^T c^2 = \frac{1}{3}$, $b^T c^3 = \frac{1}{4}$, and $b^T c^4 = \frac{1}{5}$. Unfortunately, we find that the application of these conditions forces $c_4$ to be $\frac{7}{10}$. Therefore, we cannot have a method with the tableau (4.12) being fifth order, and having stage order five.

## 4.7 A 5-stage, 5th order, stage order 5 Generalized MIRK method

Since it is not possible to get a 4-stage, 5th order generalized MIRK method with stage order 5, we therefore consider an extra stage. The tableau we consider is

$$\begin{array}{c|c|ccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ c_3 & v_3 & x_{31} & x_{32} & x_{33} & x_{34} & 0 \\ c_4 & v_4 & x_{41} & x_{42} & x_{43} & x_{44} & 0 \\ c_5 & v_5 & x_{51} & x_{52} & x_{53} & x_{54} & 0 \\ \hline & & b_1 & b_2 & b_3 & b_4 & b_5 \end{array} . \tag{4.13}$$

Note that the third and fourth stages are implicit in $y_{i+1}$ and each other.

We apply the stage order conditions up to stage order 5 which are $Xe + v = c$, $Xc + \frac{v}{2} = \frac{c^2}{2}$, $Xc^2 + \frac{v}{3} = \frac{c^3}{3}$, $Xc^3 + \frac{v}{4} = \frac{c^4}{4}$, and $Xc^4 + \frac{v}{5} = \frac{c^5}{5}$, and the fifth order conditions up to 5th order. These are again $b^T e = 1$, $b^T c = \frac{1}{2}$, $b^T c^2 = \frac{1}{3}$, $b^T c^3 = \frac{1}{4}$, and $b^T c^4 = \frac{1}{5}$. We find that it is possible to choose the coefficients of the method to satisfy these conditions. This gives a family of methods with free parameters $c_3$, $c_4$, and $c_5$. The restrictions on the parameters $c_3$, $c_4$, and $c_5$ are $c_3 \neq 0, 1, c_4, c_5$, $c_4 \neq 0, 1$, and $c_5 \neq 0, 1$. With a choice of $c_3 = \frac{1}{4}$, $c_4 = \frac{3}{4}$, and $c_5 = \frac{1}{2}$ the tableau of the resultant

method is

$$\begin{array}{c|c|ccccc|c} 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & \\ \frac{1}{4} & \frac{-11}{16} & \frac{9}{64} & \frac{3}{64} & \frac{15}{32} & \frac{9}{32} & 0 & \\ \frac{3}{4} & \frac{27}{16} & \frac{-3}{64} & \frac{-9}{64} & \frac{-9}{32} & \frac{-15}{32} & 0 & \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{24} & \frac{-1}{24} & \frac{1}{6} & \frac{-1}{6} & 0 & \\ \hline & & \frac{7}{90} & \frac{7}{90} & \frac{16}{45} & \frac{16}{45} & \frac{2}{15} & \end{array} . \tag{4.14}$$

## 4.8 A 5-stage, 6th order, stage order 4 Generalized MIRK method

We first consider a generalization of a five stage, sixth order, stage order three MIRK method given in (2.22), to a five stage, sixth order, stage order four generalized MIRK method. Its tableau is:

$$\begin{array}{c|c|ccccc|c} 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & \\ c_3 & v_3 & x_{31} & x_{32} & x_{33} & 0 & 0 & \\ c_4 & v_4 & x_{41} & x_{42} & x_{43} & 0 & 0 & \\ c_5 & v_5 & x_{51} & x_{52} & x_{53} & x_{54} & 0 & \\ \hline & & b_1 & b_2 & b_3 & b_4 & b_5 & \end{array} . \tag{4.15}$$

Note that $k_3$ is implicit in $y_{i+1}$ and itself.

Applying the stage order four conditions (4.7) allows us to reduce the number of order conditions, which is 37 for 6th order, to the seven conditions $b^T e = 1$, $b^T c = \frac{1}{2}$,

$b^T c^2 = \frac{1}{3}$, $b^T c^3 = \frac{1}{4}$, $b^T c^4 = \frac{1}{5}$, $b^T c^5 = \frac{1}{6}$, and $b^T(Xc^4 + \frac{v}{5}) = \frac{1}{30}$. The application of the stage order 4 conditions, and the order conditions up to order 6 leads to a family of methods with 2 free parameters $c_3$, and $c_4$, with $c_3 \neq 0, 1, c_4, c_5$, and $c_4 \neq 0, 1, c_5$. An example with $c_3 = \frac{1}{3}$, and $c_4 = \frac{2}{3}$, is the method whose tableau is

$$\begin{array}{c|c|ccccc|c}
0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & \\
\frac{1}{3} & \frac{-5}{27} & \frac{4}{27} & \frac{1}{27} & \frac{1}{3} & 0 & 0 & \\
\frac{2}{3} & \frac{8}{27} & \frac{2}{27} & \frac{-1}{27} & \frac{1}{3} & 0 & 0 & \\
\frac{1}{2} & \frac{-5}{8} & \frac{25}{128} & \frac{11}{128} & \frac{81}{128} & \frac{27}{128} & 0 & \\
\hline
 & & \frac{11}{120} & \frac{11}{120} & \frac{27}{40} & \frac{27}{40} & \frac{-8}{15} &
\end{array} . \tag{4.16}$$

This method improves upon the corresponding standard MIRK method in that it has stage order 4 but we would expect this method to exhibit some order reduction when applied to stiff ODEs since its stage order is two orders below its classical order.

## 4.9 A 5-stage, 6th order, stage order 5 Generalized MIRK method

Based on the results of the previous subsection, we next turn to a five stage, sixth order generalized MIRK with stage order five. The general form for the tableau of a

generalized, 5-stage, 6th order, stage order 5 MIRK method is

$$\begin{array}{c|c|ccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ c_3 & v_3 & x_{31} & x_{32} & x_{33} & x_{34} & 0 \\ c_4 & v_4 & x_{41} & x_{42} & x_{43} & x_{44} & 0 \\ c_5 & v_5 & x_{51} & x_{52} & x_{53} & x_{54} & 0 \\ \hline & & b_1 & b_2 & b_3 & b_4 & b_5 \end{array} . \tag{4.17}$$

Note that $k_3$, and $k_4$ now depend implicitly on $y_{i+1}$ and each other.

We apply the stage order five conditions $Xe + v = c$, $Xc + \frac{v}{2} = \frac{c^2}{2}$, $Xc^2 + \frac{v}{3} = \frac{c^3}{3}$, $Xc^3 + \frac{v}{4} = \frac{c^4}{4}$, and $Xc^4 + \frac{v}{5} = \frac{c^5}{5}$, and the order conditions up to order six, which for stage order five methods, are $b^T e = 1$, $b^T c = \frac{1}{2}$, $b^T c^2 = \frac{1}{3}$, $b^T c^3 = \frac{1}{4}$, $b^T c^4 = \frac{1}{5}$, and $b^T c^5 = \frac{1}{6}$. This gives a family of methods with $c_3$ and $c_4$ free parameters with the restrictions that $c_3 \neq 0, 1, c_4$, and $c_4 \neq 0, 1$. By choosing $c_3 = \frac{1}{5}$, and $c_4 = \frac{4}{5}$, we get a method with the tableau

$$\begin{array}{c|c|ccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{5} & \frac{-79}{625} & \frac{52}{625} & \frac{2}{625} & \frac{14}{75} & \frac{4}{75} & 0 \\ \frac{4}{5} & \frac{704}{625} & \frac{-2}{625} & \frac{-52}{625} & \frac{-4}{75} & \frac{-14}{75} & 0 \\ \frac{1}{2} & \frac{1}{2} & \frac{7}{256} & \frac{-7}{256} & \frac{125}{768} & \frac{-125}{768} & 0 \\ \hline & & \frac{1}{16} & \frac{1}{16} & \frac{125}{432} & \frac{125}{432} & \frac{8}{27} \end{array} . \tag{4.18}$$

This method improves upon the method of the previous section but still has stage order one below its classical order and it would therefore be expected to exhibit order reduction when applied to a stiff ODE.

## 4.10 A 5-stage, 6th order, stage order 6 Generalized MIRK method is not possible

We next consider trying to derive a 5-stage, 6th order generalized MIRK method with stage order 6. The tableau we consider is

$$\begin{array}{c|c|ccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ c_3 & v_3 & x_{31} & x_{32} & x_{33} & x_{34} & x_{35} \\ c_4 & v_4 & x_{41} & x_{42} & x_{43} & x_{44} & x_{45} \\ c_5 & v_5 & x_{51} & x_{52} & x_{53} & x_{54} & x_{55} \\ \hline & & b_1 & b_2 & b_3 & b_4 & b_5 \end{array} . \tag{4.19}$$

Note that $k_3$, $k_4$, and $k_5$ depends implicitly on $y_{i+1}$ and on each other.

Applying the six stage order conditions $Xe + v = c$, $Xc + \frac{v}{2} = \frac{c^2}{2}$, $Xc^2 + \frac{v}{3} = \frac{c^3}{3}$, $Xc^3 + \frac{v}{4} = \frac{c^4}{4}$, $Xc^4 + \frac{v}{5} = \frac{c^5}{5}$, and $Xc^5 + \frac{v}{6} = \frac{c^6}{6}$, we get $v_3$, $x_{31}$, $x_{32}$, $x_{33}$, $x_{34}$, $x_{35}$, $v_4$, $x_{41}$, $x_{42}$, $x_{43}$, $x_{44}$, $x_{45}$, $v_5$, $x_{51}$, $x_{52}$, $x_{53}$, $x_{54}$, and $x_{55}$ in terms of $c_3 \neq 0, 1, c_4, c_5$, $c_4 \neq 0, 1, c_3, c_5$, and $c_5 \neq 0, 1, \frac{1}{2}$. However, when we apply the order conditions for sixth order, which for stage order 6 methods are $b^T e = 1$, $b^T c = \frac{1}{2}$, $b^T c^2 = \frac{1}{3}$, $b^T c^3 = \frac{1}{4}$, $b^T c^4 = \frac{1}{5}$, and $b^T c^5 = \frac{1}{6}$, we find that $c_5$ must be $\frac{1}{2}$. Therefore, it is not possible to

obtain a 5-stage, 6th order, stage order 6 generalized MIRK method.

### 4.11 A 6-stage, 6th order, stage order 6 Generalized MIRK method

Since we cannot have a 5-stage, 6th order, stage order 6 generalized MIRK method, we therefore consider using an extra stage to increase the stage order. The tableau we consider is

$$\begin{array}{c|c|cccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_3 & v_3 & x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & 0 \\ c_4 & v_4 & x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & 0 \\ c_5 & v_5 & x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & 0 \\ c_6 & v_6 & x_{61} & x_{62} & x_{63} & x_{64} & x_{65} & 0 \\ \hline & & b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \end{array} . \tag{4.20}$$

Note that $k_3$, $k_4$, and $k_5$ now depend implicitly on $y_{i+1}$, and each other.

The application of the stage order 6 conditions $Xe+v=c$, $Xc+\frac{v}{2}=\frac{c^2}{2}$, $Xc^2+\frac{v}{3}=\frac{c^3}{3}$, $Xc^3+\frac{v}{4}=\frac{c^4}{4}$, $Xc^4+\frac{v}{5}=\frac{c^5}{5}$, and $Xc^5+\frac{v}{6}=\frac{c^6}{6}$, and the order conditions, which for a stage order 6 method are $b^Te=1$, $b^Tc=\frac{1}{2}$, $b^Tc^2=\frac{1}{3}$, $b^Tc^3=\frac{1}{4}$, $b^Tc^4=\frac{1}{5}$, and $b^Tc^5=\frac{1}{6}$, leads to four free parameters $c_3$, $c_4$, $c_5$, and $c_6$. The restrictions on the parameters $c_3$, $c_4$, $c_5$, and $c_6$ are $c_3 \neq 0,1,c_4,c_5,c_6$, $c_4 \neq 0,1$, and $c_5 \neq 0,1$, and $c_6 \neq 0,1$. With a choice of $c_3=\frac{1}{3}$, $c_4=\frac{2}{3}$, $c_5=\frac{1}{4}$, $c_6=\frac{3}{4}$ the tableau of the resultant

MIRK method is

$$\begin{array}{c|c|cccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{1}{3} & \frac{-23}{81} & \frac{23}{243} & \frac{20}{729} & \frac{-2}{9} & \frac{7}{45} & \frac{2048}{3645} & 0 \\
\frac{2}{3} & \frac{-56}{81} & \frac{32}{243} & \frac{47}{729} & \frac{1}{9} & \frac{22}{45} & \frac{2048}{3645} & 0 \\
\frac{1}{4} & \frac{-299}{1024} & \frac{783}{8192} & \frac{231}{8192} & \frac{-2187}{8192} & \frac{6561}{40960} & \frac{21}{40} & 0 \\
\frac{3}{4} & \frac{-567}{1024} & \frac{987}{8192} & \frac{435}{8192} & \frac{729}{8192} & \frac{21141}{40960} & \frac{21}{40} & 0 \\
\hline
 & & \frac{29}{360} & \frac{29}{360} & \frac{27}{200} & \frac{27}{200} & \frac{64}{225} & \frac{64}{225}
\end{array} . \tag{4.21}$$

# Chapter 5

# Experimental Results for Generalized MIRK Methods

## 5.1 Numerical Results For IVODEs

In this section, we show numerical results for the stiff IVODE (3.2) for generalized MIRK methods of various orders. The implementations are done in Scilab. See Appendix A.

Table 5.1: Numerical results for the stiff IVODE (3.2) with $\lambda = -150$ for the 3-stage, third order, stage order 3 MIRK method (4.4).

| $h$ | maxerr | error ratio | observed order |
|---|---|---|---|
| 0.25 | 0.0000832 | | |
| 0.125 | 0.0000103 | 8.0680902 | 3.0122272 |

Table (5.1) shows that for the stiff IVODE (3.2) with $\lambda = -150$, the 3-stage, 3rd order, stage order 3 MIRK method (4.4) does not suffer from order reduction because it still has its classical order of 3.

Table 5.2: Numerical results for the stiff IVODE (3.2) with $\lambda = -5000$ for the 4-stage, fourth order, stage order 4 generalized MIRK method (4.9).

| $h$ | maxerr | error ratio | observed order |
|---|---|---|---|
| 0.6 | 0.0000002 | | |
| 0.3 | 1.321D-08 | 14.254651 | 3.8333608 |
| 0.15 | 8.701D-10 | 15.181713 | 3.9242627 |

Table (5.2) shows that the 4-stage, 4th order, stage order 4 generalized MIRK method (4.9) does not suffer from order reduction when applied to the stiff IVODE (3.2) with $\lambda = -5000$ because it has its classical order of 4. This is in contrast to the standard 3-stage, 4th order, stage order 3 MIRK method (2.20), which we saw in Table 3.4 exhibits order reduction when applied to this problem.

Table 5.3: Numerical results for the stiff IVODE (3.2) with $\lambda = -55$ for the 4-stage, fifth order, stage order 4 generalized MIRK method (4.11).

| $h$ | maxerr | error ratio | observed order |
|---|---|---|---|
| 0.1 | 3.442D-10 | | |
| 0.05 | 2.607D-11 | 13.202391 | 3.7227273 |
| 0.025 | 1.644D-12 | 15.856718 | 3.9870223 |

Table (5.3) shows that for the stiff IVODE (3.2) with $\lambda = -55$, the order of the 4-stage, 5th order, stage order 4 generalized MIRK method (4.11) is reduced to 4, but this is better than the standard 4-stage, 5th order, stage order 3 MIRK method (2.21), whose order was reduced to 3. See Table 3.6.

Table 5.4: Numerical results for the stiff IVODE (3.2) with $\lambda = -55$ for the 5-stage, fifth order, stage order 5 generalized MIRK method (4.13).

| $h$ | maxerr | error ratio | observed order |
|---|---|---|---|
| 0.5 | 0.0000001 | | |
| 0.25 | 3.076D-09 | 32.509753 | 5.0228007 |

Table (5.4) shows that for the stiff IVODE (3.2) with $\lambda = -55$, the 5-stage, 5th order, stage order 5 generalized MIRK method (4.13) does not suffer from order reduction because it retains its classical order of 5.

Table 5.5: Numerical results for the stiff IVODE (3.2) with $\lambda = -5000$ for the 5-stage, sixth order, stage order 4 generalized MIRK method (4.16).

| $h$ | maxerr | error ratio | observed order |
|---|---|---|---|
| 0.2 | 2.703D-09 | | |
| 0.1 | 1.737D-10 | 15.55887 | 3.9596654 |
| 0.05 | 1.081D-11 | 16.06284 | 4.0056551 |

Table (5.5) shows that for the stiff IVODE (3.2) with $\lambda = -5000$, the 5-stage, 6th order, stage order 4 generalized MIRK method (4.16) has its order reduced to 4, but this is better than the standard 5-stage, 6th order, stage order 3 MIRK method (2.22), whose order was reduced to 3. See Table 3.8.

Table 5.6: Numerical results for the stiff IVODE (3.2) with $\lambda = -5000$ for the 5-stage, sixth order, stage order 5 generalized MIRK method (4.18).

| $h$ | maxerr | error ratio | observed order |
|---|---|---|---|
| 0.2 | 1.161D-10 | | |
| 0.1 | 4.322D-12 | 26.852446 | 4.7469816 |
| 0.05 | 1.181D-13 | 36.586466 | 5.1932382 |

Table (5.6) shows that for the stiff IVODE (3.2) with $\lambda = -5000$, the order of the 5-stage, 6th order, stage order 5 generalized MIRK method (4.18) reduced to 5. However, this is better than the standard 5-stage, 6th order, stage order 3 MIRK method (2.22), whose order was reduced to 3. See Table 3.8.

Table 5.7: Numerical results for the stiff IVODE (3.2) with $\lambda = -5000$ for the 6-stage, sixth order, stage order 6 generalized MIRK method (4.21).

| $h$ | maxerr | error ratio | observed order |
|---|---|---|---|
| 0.6 | 1.874D-10 | | |
| 0.3 | 3.222D-12 | 58.169515 | 5.8621914 |
| 0.15 | 5.218D-14 | 61.753191 | 5.9484418 |

Table (5.7) shows that for the stiff IVODE (3.2) with $\lambda = -5000$, the 6-stage, 6th order, stage order 6 generalized MIRK method (4.21) does not suffer from order reduction because it still has its classical order of 6.

## 5.2 Numerical Results For BVODEs

In this section, we show numerical results for the stiff BVODE (3.4) obtained by using the 4-stage, 4th order, stage order 4 generalized MIRK method (4.9) and the 6-stage, 6th order, stage order 6 generalized MIRK method (4.21). The implementations are done in Scilab. See Appendix A.

Table 5.8: Numerical results for the stiff BVODE (3.4) with $\lambda = -150$ for the 4-stage, fourth order, stage order 4 generalized MIRK method (4.9).

| $N$ | $h$ | maxerr1 | maxerr2 | order1 | order2 |
|---|---|---|---|---|---|
| 50 | 1/50 | 0.0043325 | 0.0043325 | | |
| 100 | 1/100 | 0.0003322 | 0.0003322 | 3.7049415 | 3.7049414 |

Table (5.8) shows that for the stiff BVODE (3.4) with $\lambda = -150$, the 4-stage, 4th order, stage order 4 generalized MIRK method (4.9) retains its classical order of 4. This implies that there is no order reduction.

Table 5.9: Numerical results for the stiff BVODE (3.4) with $\lambda = -750$ for the 6-stage, sixth order, stage order 6 generalized MIRK method (4.21)

| $N$ | $h$ | maxerr1 | maxerr2 | order1 | order2 |
|---|---|---|---|---|---|
| 20 | 1/20 | 0.1015255 | 0.1015255 | | |
| 40 | 1/40 | 0.0012637 | 0.0012637 | 6.3280603 | 6.3280603 |

Table (5.9) shows that for the stiff BVODE (3.4) with $\lambda = -750$, the 6-stage, 6th order, stage order 6 generalized MIRK method (4.21) does not suffer from order reduction because it still has its classical order of 6.

### 5.3 Timing Comparison

In this section, we show results to compare CPU times for standard MIRK methods with generalized MIRK methods when applied to stiff ODEs. We will require that the methods achieve the same error. We note that the standard MIRK methods will be more efficient per step but will have to take smaller steps due to order reduction. On the other hand, the generalized MIRK method will use more work per step but can take larger steps. The time is given in seconds as measured within the Scilab environment; version 5.5.1. The computer had an Intel(R) Core(TM) i3-2310M processor.

Table 5.10: Comparison of the 3-stage, fourth order, stage order 3 standard MIRK method (2.20) with the 4-stage, fourth order, stage order 4 generalized MIRK method (4.9) on the stiff IVODE (3.2) with $\lambda = -5000$.

| Method Type | $h$ | maxerr | time |
|---|---|---|---|
| The 4th order standard MIRK method | 0.1 | 0.0000002 | 4.2140315 |
| The 4th order generalized MIRK method | 0.6 | 0.0000002 | 2.1216136 |

We experimentally chose the step size for each method so that they obtain the same error. We see that the 4th order generalized MIRK method can take a step that is six times larger. Even though there is more work per step for the generalized MIRK method, the fact that it can take larger steps means that it overall requires less CPU time than the standard MIRK method.

Table 5.11: Comparison of the 5-stage, sixth order, stage order 3 standard MIRK method (2.22) with the 6-stage, sixth order, stage order 6 generalized MIRK method (4.21) on the stiff IVODE (3.2) with $\lambda = -5000$.

| Method Type | $h$ | maxerr | time |
|---|---|---|---|
| The 6th order standard MIRK method | 0.2 | $1.7\times10^{-9}$ | 6.9018476 |
| The 6th order generalized MIRK method | 0.88 | $1.6\times10^{-9}$ | 5.0388323 |

We experimentally select the step size for each method so that they achieve the same error. We notice that the 6th order generalized MIRK method can take a step that is 4.4 times larger. Even though the generalized MIRK method uses more work per step, the fact that it can take larger steps means that it requires less CPU time than the standard MIRK method.

If we were to apply a generalized MIRK method to a non-stiff ODE, this advantage would disappear because the standard MIRK method would perform at its classical order. This would mean that it could take steps of approximately the same size as this of the generalized MIRK method but would require less work per step. This points out that it is important to couple the use of generalized MIRK methods with a stiffness detection algorithm.

## 5.4 Summary

Based on the results presented in Tables 5.10 and 5.11, we observed that the generalized MIRK methods run faster than standard MIRK methods when both are required to achieve the same accuracy.

# Chapter 6

# Conclusion and Future Work

## 6.1 Conclusion

In this thesis, we discussed MIRK methods for solving IVODEs and BVODEs. We reviewed some of the background literature for MIRK methods. The phenomenon of order reduction for certain types of numerical methods when applied to stiff ODEs was described. Then, numerical experiments were presented to demonstrate order reduction for MIRK methods when they are applied to stiff ODEs. After that, generalized MIRK methods of various orders were developed by introducing implicit stages to allow the method to have a higher stage order. We then provided numerical experiments that show that the generalized MIRK methods do not suffer from order reduction when applied to some stiff problems. We also compared CPU times for standard MIRK methods and generalized MIRK methods, and found that even though the generalized MIRK methods require more work per step they are more efficient than the corresponding MIRK methods because they can take larger steps while achieving the same accuracy.

## 6.2 Future Work

The main direction of future work following from the work described in this thesis is to implement the generalized MIRK methods within the BVP SOLVER [23] software package. BVP SOLVER is a Fortran 90/95 based solver used to solve BVODEs. A second direction for future work is to look more closely at the solution of the non-linear equations associated with the implicit stages. There may be techniques for improving the efficiency of this computation. A third direction for future work is to develop continuous mono-implicit Runge-Kutta (CMIRK) methods to augment the generalized MIRK methods to provide an accurate continuous solution approximation over the entire domain. A fourth direction for future work is to develop methods for stiffness detection so that a numerical solver can decide whether the presence of order reduction might be a concern, allowing it to switch over to the more appropriate methods described in this thesis when necessary.

## Bibliography

1. U.M. Ascher, R.M.M. Mattheij, R.D. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Classics in Applied Mathematics Series, SIAM, Philadelphia, 1995.

2. U.M. Ascher and L. P. Petzold, *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*, SIAM, Philadelphia, 1998.

3. U. Ascher, *Collocation for two-point boundary value problems revisited*, SIAM

J. Numer. Anal. 23 (1986) 596-609.

4. J.C. Butcher, *The Numerical Analysis of Ordinary Differential Equations*, Wiley, Chichester, 1987.

5. K.Burrage, F.H. Chipman and P.H. Muir, *Order results for mono-implicit Runge-Kutta methods*, SIAM J. Numer. Anal. 31 (1994) 876-891.

6. J.C. Butcher, *Implicit Runge-Kutta processes*, Math. Comp. 18 (1964) 50-64.

7. C. F. Curtiss and J. O. Hirschfelder, *Integration of stiff equations.* Proc. Nat. Acad. Sci. 38 (1952) 235-243.

8. J.R. Cash, A. Singhal, *Mono-implicit Runge-Kutta formulae for the numerical integration of stii differential systems*, IMA J. Numer. Anal. 2 (1982) 211-217.

9. J. R. Cash and A. Singhal, *High order methods for the numerical solution of two-point boundary value problems*, BIT 22 (1982) 184-199.

10. J. R. Cash, *On the numerical integration of nonlinear two-point boundary value problems using iterated deferred corrections*, *Part 1 : A survey and comparison of some one-step formulae*, Comput. Math. Appl. 12 (1986) 1029-1048.

11. J. R. Cash and M. H. Wright, *A deferred correction method for nonlinear two-point boundary value problems: implementation and numerical evaluation*, SIAM J. Sci. Statist. Comput. 12 (1991) 971-989.

12. D. L. Chen, *The efficiency of Singly-implicit Runge-Kutta methods for stiff differential equations*, Numer. Algor. 65 (2014) 533-554.

13. G. Dahlquist, *A special stability problem for linear multi-step*. Numer. Math. 3 (1963) 27-43.

14. W. H. Enright and P. H. Muir, *Runge-Kutta software with defect control for boundary value ODEs*. SIAM J. Sci. Comput. 17 (1996) 479-497.

15. W. H. Enright and P. H. Muir, *Efficient classes of Runge-Kutta methods for two-point boundary value problems*, Comput. 37 (1986) 315-334.

16. R. Frank, J. Schneid and C.W. Ueberhuber, *Order results for implicit Runge-Kutta methods applied to stiff systems*, SIAM J. Numer. Anal. 22 (1985) 515-534.

17. S. Gupta, *An adaptive boundary value Runge-Kutta solver for first order boundary value problems*, SIAM J. Numer. Anal. 22 (1985) 114-126.

18. L. D. Lambert, *Numerical Methods for Ordinary Differential Equations*. Wiley, New York, 1991.

19. P.H. Muir and W.H. Enright, *Relationships among some classes of implicit Runge-Kutta methods and their stability functions*, BIT 27 (1987) 403-423.

20. P.H. Muir, *Optimal discrete and continuous mono-implicit Runge-Kutta schemes for BVODEs*, Adv. Comput. Math. 10 (1999) 135-167.

21. H. H. Robertson, *The solution of a set of reaction rate equations*, in Numerical Analysis: An Introduction, J. Walsh, ed., Academic Press, 1966, 178-182.

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

23. L. F. Shampine, P.H. Muir, and H. Xu, *A user-friendly Fortran BVP solver*, J. Numer. Anal. Ind. Appl. Math. 1 (2006) 201-217.

24. W.M.G. Van Bokhoven, *Efficient higher order implicit one-step methods for integration of stiff differential equations*, BIT 20 (1980) 34-43.

25. Voss and P.H. Muir, *Mono-implicit Runge-Kutta schemes for the parallel solution of initial value problems*, J. Comput. Appl. Math. 102 (1999) 235-252.

26. R. Weiss, *The application of implicit Runge-Kutta and collocation methods to boundary value problems*, Math. Comput. 28 (1974) 449-464.

27. The Scilab Consortium. Scilab. http://www.scilab.org.

28. http://www.maplesoft.com/standards/MathML/info.html.

# Appendix A

### Scilab Scripts

**Application of the 2-stage, 3rd order, stage order 2 MIRK for solving a stiff IVODE (3.2)**

```
h=0.1;
for j=1:3 do
function ydot=f(t,y)
ydot=exp(-t)*(9+t)+lambda*(y-10+(10+t)*exp(-t));
endfunction
function nonlin=g(z)
fff1=f(t+h,z);
fff2=f(t+(h/3),(4/9)*y+(5/9)*z+h*((-2/9)*fff1));
nonlin=z-(y+h*((1/4)*fff1+(3/4)*fff2));
endfunction
z0=0;
y=0;
t=0;
lambda=-150;
tfinal=1; n=1; tt(n)=t; yy(n)=y;
while t<tfinal
```

```
z=fsolve(z0,g);

y=z;

t=n*h;

n=n+1;

z0=z;

tt(n)=t; yy(n)=y;

end

tt,yy

e = abs(yy-(10-(10+tt).*exp(-tt)));

ee(j)=max(e);

h=h/2;

end

for j=1:2 do

order(j)=log2(ee(j)/ee(j+1));

end

disp(order)
```

**Application of the 3-stage, 4th order, stage order 3 MIRK for solving stiff IVODE (3.2)**

```
h=0.1;

for j=1:3 do

function ydot=f(t,y)

ydot=exp(-t)*(9+t)+lambda*(y-10+(10+t)*exp(-t));
```

```
endfunction

y=0;

t=0;

lambda=-5000;

function nonlin=g(z)

fff1=f(t,y);

fff2=f(t+h,z);

fff3=f(t+(h/2),(y+z)/2+(h/8)*(fff1-fff2));

nonlin=z-(y+h*((1/6)*fff1+(1/6)*fff2+(2/3)*fff3));

endfunction

z0=0; tfinal=12; n=1; tt(n)=t; yy(n)=y;

while t<tfinal

z=fsolve(z0,g);

y=z;

t=n*h;

n=n+1; z0=z;

tt(n)=t; yy(n)=y;

end

tt,yy

e=abs(yy-(10-(10+tt).*exp(-tt)));

ee(j)=max(e);

h=h/2;
```

```
end
for j=1:2 do
order(j)=log2(ee(j)/ee(j+1));
end
disp(order)
```

**Application of the 4-stage, 4th order, stage order 4 generalized MIRK for solving stiff IVODE (3.2)**

```
h=0.6;
for j=1:3 do
function ydot=f(t,y)
ydot=exp(-t)*(9+t)+lambda*(y-10+(10+t)*exp(-t));
endfunction
y=0;
t=0;
lambda=-5000;
function nonlin=g(z)
fff1=f(t,y);
fff2=f(t+h,z(1));
fff3=f(t+(1/3)*h,(32/27)*y+(-5/27)*z(1)+h*((4/27)*fff1+(1/27)*fff2+
(1/3)*z(2)));
fff4=f(t+(2/3)*h,(19/27)*y+(8/27)*z(1)+h*((2/27)*fff1-(1/27)*fff2+
(1/3)*z(2)));
```

```
nonlin(1)=z(1)-(y+h*((1/8)*fff1+(1/8)*fff2+(3/8)*z(2)+(3/8)*fff4));

nonlin(2)=z(2)-fff3;

endfunction

z0(1)=0;

z0(2)=0;

tfinal=12; n=1; tt(n)=t; yy(n)=y;

while t<tfinal

z =fsolve(z0,g);

y=z(1);

t=n*h;

n=n+1; z0=z;

tt(n)=t; yy(n)=y;

end

tt,yy

error = abs(yy-(10-(10+tt).*exp(-tt)));

ee(j)=max(error)

h=h/2;

end

for j=1:2 do

order(j)=log2(ee(j)/ee(j+1));

end

disp(order)
```

```
// Note that I compared the standard (3-4-3) MIRK
with the generalized (4-4-4) MIRK method using tfinal=600.
```

**Application of the 4-stage, 5th order, stage order 4 generalized MIRK for solving stiff IVODE (3.2)**

```
h=0.1;
for j=1:3 do
function ydot=f(t,y)
ydot=exp(-t)*(9+t)+lambda*(y-10+(10+t)*exp(-t));
endfunction
function nonlin=g(z)
fff1=f(t,y);
fff2=f(t+h,z(1));
fff3=f(t+(1/3)*h,(32/27)*y-(5/27)*z(1)+h*((4/27)*fff1+
(1/27)*fff2+(1/3)*z(2)));
fff4=f(t+(4/5)*h,(209/625)*y+(416/625)*z(1)+h*((4/125)*fff1-
(44/625)*fff2+(108/625)*z(2)));
nonlin(1)=z(1)-(y+h*((5/48)*fff1+(1/24)*fff2+(27/56)*z(2)+
(125/336)*fff4));
nonlin(2)=z(2)-fff3;
endfunction
z0(1)=0;
z0(2)=0;
```

```
y=0;
t=0;
lambda=-55;
tfinal=1; n=1; tt(n)=t; yy(n)=y;
while t<tfinal
z=fsolve(z0,g);
y=z(1);
t=n*h;
n=n+1;
z0=z;
tt(n)=t; yy(n)=y;
end
tt,yy
e = abs(yy-(10-(10+tt).*exp(-tt)));
ee(j)=max(e)
h=h/2;
end
for j=1:2 do
order(j)=log2(ee(j)/ee(j+1)); end
disp(order)
```

**Application of the 5-stage, 6th order, stage order 4 generalized MIRK for solving stiff IVODE (3.2)**

```
h=0.2;
for j=1:3 do
function ydot=f(t,y)
ydot=exp(-t)*(9+t)+lambda*(y-10+(10+t)*exp(-t));
endfunction
function nonlin=g(z)
fff1=f(t,y);
fff2=f(t+h,z(1));
fff3=f(t+(1/3)*h,(32/27)*y-(5/27)*z(1)+h*((4/27)*fff1+(1/27)*fff2
+(1/3)*z(2)));
fff4=f(t+(2/3)*h,(19/27)*y+(8/27)*z(1)+h*((2/27)*fff1-(1/27)*fff2
+(1/3)*z(2)));
fff5=f(t+h/2,(13/8)*y-(5/8)*z(1)+h*((25/128)*fff1+(11/128)*fff2
+(81/128)*z(2)+(27/128)*fff4));
nonlin(1)=z(1)-(y+h*((11/120)*fff1+(11/120)*fff2+(27/40)*z(2)+
(27/40)*fff4-(8/15)*fff5));
nonlin(2)=z(2)-fff3;
endfunction
z0(1)=0;
z0(2)=0;
y=0;
t=0;
```

```
lambda=-5000;
tfinal=12; n=1; tt(n)=t; yy(n)=y;
while t<tfinal
z=fsolve(z0,g);
y=z(1);
t=n*h;
n=n+1;
z0=z;
tt(n)=t; yy(n)=y;
end
tt,yy
e = abs(yy-(10-(10+tt).*exp(-tt)));
ee(j)=max(e)
h=h/2;
end
for j=1:2 do
order(j)=log2(ee(j)/ee(j+1));
end
disp(order)
```

**Application of the 5-stage, 6th order, stage order 5 generalized MIRK for solving stiff IVODE (3.2)**

```
h=0.2;
```

```
for j=1:3 do

function ydot=f(t, y)

ydot=exp(-t)*(9+t)+lambda*(y-10+(10+t)*exp(-t));

endfunction

function nonlin=g(z)

fff1=f(t,y);

fff2=f(t+h,z(1));

fff3=f(t+(1/5)*h,(704/625)*y-(79/625)*z(1)+h*((52/625)*fff1+

(2/625)*fff2+(14/75)*z(2)+(4/75)*z(3)));

fff4=f(t+(4/5)*h,(-79/625)*y+(704/625)*z(1)+h*((-2/625)*fff1-

(52/625)*fff2-(4/75)*z(2)-(14/75)*z(3)));

fff5=f(t+(1/2)*h,(1/2)*y+(1/2)*z(1)+h*((7/256)*fff1-(7/256)*fff2+

(125/768)*z(2)-(125/768)*z(3)));

nonlin(1)=z(1)-(y+h*((1/16)*fff1+(1/16)*fff2+(125/432)*z(2)+

(125/432)*z(3)+(8/27)*fff5));

nonlin(2)=z(2)-fff3;

nonlin(3)=z(3)-fff4

endfunction

z0(1)=0;

z0(2)=0;

z0(3)=0;

y=0;
```

```
t=0;
lambda=-5000;
tfinal=12; n=1; tt(n)=t; yy(n)=y;
while t¡tfinal
z=fsolve(z0,g);
y=z(1);
t=n*h;
n=n+1;
z0=z;
tt(n)=t; yy(n)=y;
end
tt,yy
e = abs(yy-(10-(10+tt).*exp(-tt)));
ee(j)=max(e);
h=h/2;
end
for j=1:2 do
order(j)=log2(ee(j)/ee(j+1));
end
disp(order)
```

**Application of the 6-stage, 6th order, stage order 6 generalized MIRK for solving stiff IVODE (3.2)**

```
function ydot=f(t,y)
ydot=exp(-t)*(9+t)+lambda*(y-10+(10+t)*exp(-t));
endfunction
function nonlin=g(z)
fff1=f(t,y);
fff2=f(t+h,z(1));
fff3=f(t+(1/3)*h,(104/81)*y-(23/81)*z(1)+h*((23/243)*fff1+
(20/729)*fff2-(2/9)*z(2)+(7/45)*z(3)+(2048/3645)*z(4)));
fff4=f(t+(2/3)*h,(137/81)*y-(56/81)*z(1)+h*((32/243)*fff1+
(47/729)*fff2+(1/9)*z(2)+(22/45)*z(3)+(2048/3645)*z(4)));
fff5=f(t+(1/4)*h,(1323/1024)*y-(299/1024)*z(1)+h*((783/8192)*fff1+
(231/8192)*fff2-(2187/8192)*z(2)+(6561/40960)*z(3)+(21/40)*z(4)));
fff6=f(t+(3/4)*h,(1591/1024)*y-(567/1024)*z(1)+h*((987/8192)*fff1+
(435/8192)*fff2+(729/8192)*z(2)+(21141/40960)*z(3)+(21/40)*z(4)));
nonlin(1)=z(1)-(y+h*((29/360)*fff1+(29/360)*fff2+
(27/200)*z(2)+(27/200)*z(3)+(64/225)*z(4)+(64/225)*fff6));
nonlin(2)=z(2)-fff3;
nonlin(3)=z(3)-fff4;
nonlin(4)=z(4)-fff5;
endfunction
z0(1)=0;
z0(2)=0;
```

```
z0(3)=0;

z0(4)=0;

y=0;

t=0;

lambda=-5000;

h=0.6;

tfinal=12; n=1; tt(n)=t; yy(n)=y;

while t¡tfinal

z=fsolve(z0,g);

y=z(1);

t=n*h;

n=n+1;

z0=z;

tt(n)=t; yy(n)=y;

end

tt,yy

e = abs(yy-(10-(10+tt).*exp(-tt)));

ee=max(e)

// Note that I compared the standard (5-6-3) MIRK

with the generalized (6-6-6) MIRK method using tfinal=880.
```

**Application of the 3-stage, 4th order, stage order 3 MIRK for solving stiff BVODE (3.4)**

```
N=52; // Number of subintervals
function wdox=f(x, w)
wdox(1)=lambda*w(2);
wdox(2)=lambda*w(1)+lambda*cos(pi*x)^2+2/lambda*(pi^2)*cos(2*x*pi);
endfunction
lambda=-150;
a=0;
b=1;
h=(b-a)/N;
function mirk=g(y)
x=0;
mirk(1)=y(1);
for i=1:N
z(1)=y(2*i-1);
z(2)=y(2*i);
k1=f(x,z);
xx=x+h;
w(1)=y(2*i+1);
w(2)=y(2*i+2);
k2=f(xx,w);
xxx=x+(h/2);
u(1)=(z(1)+w(1))/2+(h/8)*(k1(1)-k2(1));
```

```
u(2)=(z(2)+w(2))/2+(h/8)*(k1(2)-k2(2));

k3=f(xxx,u);

mirk(2*i)=y(2*i+1)-y(2*i-1)-h*(k1(1)/6+k2(1)/6+2/3*k3(1));

mirk(2*i+1)=y(2*i+2)-y(2*i)-h*(k1(2)/6+k2(2)/6+2/3*k3(2));

x=x+h;

end

mirk(2*N+2)=y(2*N+1);

endfunction

y0(1)=0;

for i=2:(2*N)+2 do

y0(i)=1;

end;

s=fsolve(y0,g);

y1 = s(1:2:2*N+2);

y2 = s(2:2:2*N+2);

x=[0:h:1];

exactSol1 = (exp(lambda*(x-1))+exp(-lambda*x))/(1+exp(-lambda))

-(cos(pi*x)^2);

exactSol2 = ((exp(lambda*(x-1))-exp(-lambda*x))/(1+exp(-lambda)))

+((pi/lambda)*sin(2*pi*x));

error1(1) = abs(y1(1)-exactSol1(1));

error2(1) = abs(y2(1)-exactSol2(1));
```

for i=1:N do

error1(i+1) = abs(y1(i+1)-exactSol1(i+1));

error2(i+1) = abs(y2(i+1)-exactSol2(i+1));

end

max1 = max(error1);

max2 = max(error2);

**Application of the 4-stage, 4th order, stage order 4 generalized MIRK for solving stiff BVODE (3.4)**

N=50; // Number of subintervals

m=2; //Number of ODEs

s=2*m; //Shift=Storage of y plus storage of k3

function wdot=f(t, w)

wdot(1)=lambda*w(2);

wdot(2)=lambda*w(1)+lambda*cos(pi*t)$^2$+2/lambda*(pi$^2$)*cos(2*t*pi);

endfunction

t=0;

lambda=-150;

a=0;

b=1;

h=(b-a)/N;

function nonlin=g(y)

t=0;

```
nonlin(1)=y(1);
for i=1:N do
si=s*(i-1);
yi=y(si+1:si+2);
fff1=f(t,yi);
yip1=y(si+5:si+6);
fff2=f(t+h,yip1);
k3=y(si+3:si+4);
y3=(32/27)*yi-(5/27)*yip1+h*((4/27)*fff1+(1/27)*fff2+(1/3)*k3);
fff3=f(t+(1/3)*h,y3);
y4=(19/27)*yi+(8/27)*yip1+h*((2/27)*fff1-(1/27)*fff2+(1/3)*fff3);
fff4=f(t+(2/3)*h,y4);
nonlin(si+2:si+3)=yip1-yi-h*((1/8)*fff1+(1/8)*fff2+(3/8)*fff3+
(3/8)*fff4);
nonlin(si+4:si+5)=fff3-k3;
t=t+h;
end
nonlin(s*N+2)=y(s*N+1);
endfunction
y0(1)=0;
for i=2:(s*N)+2 do
y0(i)=1; end
```

```
sol=fsolve(y0,g);

y1=sol(1:s:s*N+2);

y2=sol(2:s:s*N+2);

t=[0:h:1];

exactSol1 = (exp(lambda*(t-1))+exp(-lambda*t))/(1+exp(-lambda))

-(cos(pi*t)^2);

exactSol2 = ((exp(lambda*(t-1))-exp(-lambda*t))/(1+exp(-lambda)))

+((pi/lambda)*sin(2*pi*t));

error1(1) = abs(y1(1)-exactSol1(1));

error2(1) = abs(y2(1)-exactSol2(1));

for i=1:N do

error1(i+1) = abs(y1(i+1)-exactSol1(i+1));

error2(i+1) = abs(y2(i+1)-exactSol2(i+1));

end

max1 = max(error1);

max2 = max(error2);
```

**Application of the 6-stage, 6th order, stage order 6 generalized MIRK for solving stiff BVODE (3.4)**

```
N=20; // Number of subintervals

m=2; // Number of ODEs

s=4*m; // Shift= Storage of y plus storage of k3,k4,k5

function wdox=f(x, w)
```

```
wdox(1)=lambda*w(2);

wdox(2)=lambda*w(1)+lambda*(cos(pi*x))^2+2/lambda*(pi^2)*cos(2*x*pi);

endfunction

lambda=-750;

a=0;

b=1;

h=(b-a)/N;

function nonlin=g(y)

x=0;

nonlin(1)=y(1);

for i=1:N

si=s*(i-1)

yi=y(si+1:si+2);

ff1=f(x,yi);

xx=x+h;

yip1=y(si+9:si+10);

ff2=f(xx,yip1);

k3=y(si+3:si+4);

k4=y(si+5:si+6);

k5=y(si+7:si+8);

y3=(104/81)*yi-(23/81)*yip1+h*((23/243)*ff1+(20/729)*ff2-

(2/9)*k3+(7/45)*k4+(2048/3645)*k5);
```

```
ff3=f(x+(1/3)*h,y3);
y4=(137/81)*yi-(56/81)*yip1+h*((32/243)*ff1+(47/729)*ff2+
(1/9)*ff3+(22/45)*k4+(2048/3645)*k5);
ff4=f(x+(2/3)*h,y4);
y5=(1323/1024)*yi-(299/1024)*yip1+h*((783/8192)*ff1+(231/8192)
*ff2-(2187/8192)*ff3+(6561/40960)*ff4+(21/40)*k5);
ff5=f(x+(h/4),y5);
y6=(1591/1024)*yi-(567/1024)*yip1+h*((987/8192)*ff1+(435/8192)
*ff2+(729/8192)*ff3+(21141/40960)*ff4+(21/40)*ff5);
ff6=f(x+(3/4)*h,y6);
nonlin(si+2:si+3)=yip1-yi-h*((29/360)*ff1+(29/360)*ff2+(27/200)
*ff3+(27/200)*ff4+(64/225)*ff5+(64/225)*ff6);
nonlin(si+4:si+5)=ff3-k3;
nonlin(si+6:si+7)=ff4-k4;
nonlin(si+8:si+9)=ff5-k5;
x=x+h;
end
nonlin(s*N+2)=y(s*N+1);
endfunction
y0(1)=0;
for i=2:(s*N)+2 do
y0(i)=1;
```

```
end;
sol=fsolve(y0,g);
y1 = sol(1:s:s*N+2);
y2 = sol(2:s:s*N+2);
x=[0:h:1];
exactSol1 = (exp(lambda*(x-1))+exp(-lambda*x))/(1+exp(-lambda))
-(cos(pi*x)^2);
exactSol2 = ((exp(lambda*(x-1))-exp(-lambda*x))/(1+exp(-lambda)))
+((pi/lambda)*sin(2*pi* x));
error1(1) = abs(y1(1)-exactSol1(1));
error2(1) = abs(y2(1)-exactSol2(1));
for i=1:N do
error1(i+1) = abs(y1(i+1)-exactSol1(i+1));
error2(i+1) = abs(y2(i+1)-exactSol2(i+1));
end
max1 = max(error1);
max2 = max(error2);
```

# Appendix B

## Maple Scripts

### Code for a 3-stage, 4th order, stage order 4 generalized MIRK method

```
s := 3;

one := array(1..s);

c := array(1..s);

c2 := array(1..s);

c3 := array(1..s);

c4 := array(1..s);

v := array(1..s);

b := array(1..s);

x := array(1..s,1..s);

C1 := array(1..s);

C2 := array(1..s);

C3 := array(1..s);

C4 := array(1..s);

M := array(1..3,1..s);

RHS := array(1..3);

// Definitions

for i from 1 to s do
```

one[i]:=1;

c2[i] := $c[i]^2$;

c3[i] := $c[i]^3$;

c4[i] := $c[i]^4$;

od;

// Fill in zeros in the 'x' matrix.

for i from 1 to 2 do

for j from i to s do

x[i,j] := 0

od;

od;

// The stage order conditions allow us to reduce the number of order

conditions to $b^T * e = 1, b^T * c = 1/2, b^T * c^2 = 1/3, b^T * c^3 = 1/4$

// Set up stage order conditions for C(4).

C1:= matadd(matadd(multiply(x,one),v),c,1,-1);

C2:= matadd(matadd(multiply(x,c),v,1,1/2),c2,1,-1/2);

C3:= matadd(matadd(multiply(x,c2),v,1,1/3),c3,1,-1/3);

C4:= matadd(matadd(multiply(x,c3),v,1,1/4),c4,1,-1/4);

// Impose C(4) conditions on the first stage.

// Q:=solve(C1[1],C2[1],C3[1],C4[1],c[1],v[1]);

// This gives two solutions: c[1]=v[1]=0 or c[1]=v[1]=1.

c[1] := 0; v[1] := c[1];

```
// Impose C(4) on the second stage.

Q:=solve(C1[2],C2[2],C3[2],C4[2],c[2],v[2],x[2,1]);

// This gives solutions: c[2] := 1; v[2] :=1; x[2,1]:=0;

// Impose C(4) on the third stage.

Q:=solve(C1[3],C2[3],C3[3],C4[3],v[3],x[3,1],x[3,2],

x[3,3]);

assign(Q);

// Setup the coefficient matrix,M, for the determination of the weights.

for i from 1 to s do

M[1,i] := 1; M[2,i] := c[i]; M[3,i] := c2[i];

RHS[i] := 1/i

od;

b:=linsolve(M,RHS);

solve (dotprod(b,c3)-1/4, c[3]);
```

**Code for a 4-stage, 4th order, stage order 4 generalized MIRK method.**

```
s := 4;

one := array(1..s);

c := array(1..s);

c2 := array(1..s);

c3 := array(1..s);

c4 := array(1..s);

v := array(1..s);
```

b := array(1..s);

x := array(1..s,1..s);

C1 := array(1..s);

C2 := array(1..s);

C3 := array(1..s);

C4 := array(1..s);

M := array(1..4,1..s);

RHS := array(1..4);

// Definitions

for i from 1 to s do

one[i]:=1;

c2[i] := $c[i]^2$;

c3[i] := $c[i]^3$;

c4[i] := $c[i]^4$;

od;

// Fill in zeros in the 'x' matrix.

for i from 1 to 2 do

for j from i to s do

x[i,j] := 0

od;

od;

// The stage order conditions allow us to reduce the number of order

conditions to $b^T * e = 1, b^T * c = 1/2, b^T * c^2 = 1/3, b^T * c^3 = 1/4$

// Set up stage order conditions for C(4).

C1:= matadd(matadd(multiply(x,one),v),c,1,-1);

C2:= matadd(matadd(multiply(x,c),v,1,1/2),c2,1,-1/2);

C3:= matadd(matadd(multiply(x,c2),v,1,1/3),c3,1,-1/3);

C4:= matadd(matadd(multiply(x,c3),v,1,1/4),c4,1,-1/4);

// Impose C(4) conditions on the first stage.

// Q:=solve(C1[1],C2[1],C3[1],C4[1],c[1],v[1]);

// This gives two solutions: c[1]=v[1]=0 or c[1]=v[1]=1.

c[1] := 0; v[1] := c[1];

// Impose C(4) on the second stage.

Q:=solve(C1[2],C2[2],C3[2],C4[2],c[2],v[2],x[2,1]);

// This gives solutions: c[2] := 1; v[2] :=1; x[2,1]:=0;

// Impose C(4) on the third stage.

Q:=solve(C1[3],C2[3],C3[3],C4[3],c[3],v[3],x[3,1],x[3,2],x[3,3]);

assign(Q);

// Impose C(4) on the fourth stage.

Q:=solve(C1[4],C2[4],C3[4],C4[4],v[4],x[4,1],x[4,2],x[4,3]);

assign(Q);

// Setup the coefficient matrix,M, for the determination of the weights.

for i from 1 to s do

M[1,i] := 1; M[2,i] := c[i]; M[3,i] := c2[i];

```
M[4,i] := c3[i];

RHS[i] := 1/i

od;

b:=linsolve(M,RHS);
```

**Code for a 4-stage, 5th order, stage order 4 generalized MIRK method.**

```
s := 4;

one := array(1..s);

c := array(1..s);

c2 := array(1..s);

c3 := array(1..s);

c4 := array(1..s);

v := array(1..s);

b := array(1..s);

x := array(1..s,1..s);

C1 := array(1..s);

C2 := array(1..s);

C3 := array(1..s);

C4 := array(1..s);

M := array(1..4,1..s);

RHS := array(1..4);

// Definitions

for i from 1 to s do
```

one[i]:=1;

c2[i] := $c[i]^2$;

c3[i] := $c[i]^3$;

c4[i] := $c[i]^4$;

od;

// Fill in zeros in the 'x' matrix.

for i from 1 to 2 do

for j from i to s do

x[i,j] := 0

od;

od;

// The stage order conditions allow us to reduce the number of order conditions to $b^T * e = 1, b^T * c = 1/2, b^T * c^2 = 1/3, b^T * c^3 = 1/4, b^T * c^4 = 1/5$

// Set up stage order conditions for C(4).

C1:= matadd(matadd(multiply(x,one),v),c,1,-1);

C2:= matadd(matadd(multiply(x,c),v,1,1/2),c2,1,-1/2);

C3:= matadd(matadd(multiply(x,c2),v,1,1/3),c3,1,-1/3);

C4:= matadd(matadd(multiply(x,c3),v,1,1/4),c4,1,-1/4);

// Impose C(4) conditions on the first stage.

// Q:=solve(C1[1],C2[1],C3[1],C4[1],c[1],v[1]);

// This gives two solutions: c[1]=v[1]=0 or c[1]=v[1]=1.

c[1] := 0; v[1] := c[1];

```
// Impose C(4) on the second stage.

Q:=solve(C1[2],C2[2],C3[2],C4[2],c[2],v[2],x[2,1]);

// This gives solutions:

c[2] := 1; v[2] :=1; x[2,1]:=0;

// Impose C(4) on the third stage.

Q:=solve(C1[3],C2[3],C3[3],C4[3],v[3],x[3,1],x[3,2],x[3,3]);

assign(Q);

// Impose C(4) on the fourth stage.

Q:=solve(C1[4],C2[4],C3[4],C4[4],v[4],x[4,1],x[4,2],x[4,3]);

assign(Q);

// Setup the coefficient matrix,M, for the determination of the weights.

for i from 1 to s do

M[1,i] := 1; M[2,i] := c[i]; M[3,i] := c2[i]; M[4,i] := c3[i];

RHS[i] := 1/i

od;

b:=linsolve(M,RHS);

solve (dotprod(b,c4)-1/5, c[4]);
```

**Code for a 5-stage, 6th order, stage order 4 generalized MIRK method.**

```
s := 5;

one := array(1..s);

c := array(1..s);

c2 := array(1..s);
```

c3 := array(1..s);

c4 := array(1..s);

c5 := array(1..s);

v := array(1..s);

b := array(1..s);

x := array(1..s,1..s);

C1 := array(1..s);

C2 := array(1..s);

C3 := array(1..s);

C4 := array(1..s);

M := array(1..5,1..s);

RHS := array(1..5);

// Definitions

for i from 1 to s do

one[i]:=1;

c2[i] := $c[i]^2$;

c3[i] := $c[i]^3$;

c4[i] := $c[i]^4$;

c5[i] := $c[i]^5$;

od;

// Fill in zeros in the 'x' matrix.

for i from 1 to 2 do

for j from i to s do

x[i,j] := 0

od;

od;

// The stage order four conditions allow us to reduce the number of order conditions

to the six conditions $b^T * e = 1, b^T * c = 1/2, b^T * c^2 = 1/3, b^T * c^3 = 1/4, b^T * c^4 = 1/5, b^T * c^5 = 1/6$, plus the condition $b^T(X * c^4 + v/5) = 1/30$.

// The four stage order conditions are $Xe + v = c, Xc + v/2 = c^2/2, Xc^2 + v/3 = c^3/3, Xc^3 + v/4 = c^4/4$.

// Set up stage order conditions for C(4).

C1:= matadd(matadd(multiply(x,one),v),c,1,-1);

C2:= matadd(matadd(multiply(x,c),v,1,1/2),c2,1,-1/2);

C3:= matadd(matadd(multiply(x,c2),v,1,1/3),c3,1,-1/3);

C4:= matadd(matadd(multiply(x,c3),v,1,1/4),c4,1,-1/4);

// Impose C(4) conditions on the first stage.

// Q:=solve(C1[1],C2[1],C3[1],C4[1],c[1],v[1]);

// This gives two solutions: c[1]=v[1]=0 or c[1]=v[1]=1.

c[1] := 0; v[1] := c[1];

// Impose C(4) on the second stage.

Q:=solve(C1[2],C2[2],C3[2],C4[2],c[2],v[2],x[2,1]);

// This gives solutions:

```
c[2] := 1; v[2] :=1; x[2,1]:=0;

// Impose C(4) on the third stage.

Q:=solve(C1[3],C2[3],C3[3],C4[3],v[3],x[3,1],x[3,2],x[3,3]);

assign(Q);

// Impose C(4) on the fourth stage.

Q:=solve(C1[4],C2[4],C3[4],C4[4],v[4],x[4,1],x[4,2],x[4,3]);

assign(Q);

// Impose C(4) on the fifth stage.

Q:=solve(C1[5],C2[5],C3[5],C4[5], x[5,1],x[5,2],x[5,3], x[5,4]);

assign(Q);

// Setup the coefficient matrix,M, for the determination of the weights.

for i from 1 to s do

M[1,i] := 1; M[2,i] := c[i]; M[3,i] := c2[i]; M[4,i] := c3[i]; M[5,i] := c4[i];

RHS[i] := 1/i

od;

b:=linsolve(M,RHS);

Cond[1]:=dotprod(b,c5)-1/6;

Q:=solve(Cond[1],c[5]);

assign(Q);
```

**Code for a 5-stage, 6th order, stage order 5 generalized MIRK method.**

```
s := 5;

one := array(1..s);
```

c := array(1..s);

c2 := array(1..s);

c3 := array(1..s);

c4 := array(1..s);

c5 := array(1..s);

v := array(1..s);

b := array(1..s);

x := array(1..s,1..s);

C1 := array(1..s);

C2 := array(1..s);

C3 := array(1..s);

C4 := array(1..s);

C5 := array(1..s);

M := array(1..5,1..s);

RHS := array(1..5);

// Definitions

for i from 1 to s do

one[i]:=1;

c2[i] := $c[i]^2$;

c3[i] := $c[i]^3$;

c4[i] := $c[i]^4$;

c5[i] := $c[i]^5$;

od;

// Fill in zeros in the 'x' matrix.

for i from 1 to 2 do

for j from i to s do

x[i,j] := 0

od;

od;

//The stage order four conditions allow us to reduce the number of order conditions to the

six conditions $b^T * e = 1, b^T * c = 1/2, b^T * c^2 = 1/3, b^T * c^3 = 1/4, b^T * c^4 = 1/5, b^T * c^5 = 1/6,$

The five stage order conditions are $Xe + v = c, Xc + v/2 = c^2/2, Xc^2 + v/3 = c^3/3, Xc^3 + v/4 = c^4/4, Xc^4 + v/5 = c^5/5.$

// Set up stage order conditions for C(5).

C1:= matadd(matadd(multiply(x,one),v),c,1,-1);

C2:= matadd(matadd(multiply(x,c),v,1,1/2),c2,1,-1/2);

C3:= matadd(matadd(multiply(x,c2),v,1,1/3),c3,1,-1/3);

C4:= matadd(matadd(multiply(x,c3),v,1,1/4),c4,1,-1/4);

C5:= matadd(matadd(multiply(x,c4),v,1,1/5),c5,1,-1/5);

// Impose C(5) conditions on the first stage.

// Q:=solve(C1[1],C2[1],C3[1],C4[1], C5[1],c[1],v[1]);

//This gives two solutions: c[1]=v[1]=0 or c[1]=v[1]=1.

```
c[1] := 0; v[1] := c[1];

//Impose C(5) on the second stage.

Q:=solve(C1[2],C2[2],C3[2],C4[2],C5[2],c[2],v[2],x[2,1]); // This gives solutions:

c[2] := 1; v[2] :=1; x[2,1]:=0;

// Impose C(5) on the third stage.

Q:=solve(C1[3],C2[3],C3[3],C4[3],C5[3],v[3],x[3,1],x[3,2],x[3,3],x[3,4]);

assign(Q);

// Impose C(5) on the fourth stage.

Q:=solve(C1[4],C2[4],C3[4],C4[4],C5[4],v[4],x[4,1],x[4,2],x[4,3],x[4,4]);

assign(Q);

// Impose C(5) on the fifth stage.

Q:=solve(C1[5],C2[5],C3[5],C4[5],C5[5],v[5],x[5,1],x[5,2],x[5,3],x[5,4]);

assign(Q);

//Setup the coefficient matrix,M, for the determination of the weights.

for i from 1 to s do

M[1,i] := 1; M[2,i] := c[i]; M[3,i] := c2[i]; M[4,i] := c3[i]; M[5,i] := c4[i];

RHS[i] := 1/i

od;

b:=linsolve(M,RHS);

solve(dotprod(b,c5)-1/6,c[5]);
```

**Code for a 6-stage, 6th order, stage order 6 generalized MIRK method.**

```
s := 6;

one := array(1..s);

c := array(1..s);

c2 := array(1..s);

c3 := array(1..s);

c4 := array(1..s);

c5 := array(1..s);

c6 := array(1..s);

v := array(1..s);

b := array(1..s);

x := array(1..s,1..s);

C1 := array(1..s);

C2 := array(1..s);

C3 := array(1..s);

C4 := array(1..s);

C5 := array(1..s);

C6 := array(1..s);

M := array(1..6,1..s);

RHS := array(1..6);

// Definitions

for i from 1 to s do

one[i]:=1;
```

c2[i] := $c[i]^2$;

c3[i] := $c[i]^3$;

c4[i] := $c[i]^4$;

c5[i] := $c[i]^5$;

c6[i] := $c[i]^6$;

od;

// Fill in zeros in the 'x' matrix.

for i from 1 to 2 do

for j from i to s do

x[i,j] := 0

od;

od;

//The stage order four conditions allow us to reduce the number of order conditions to the six conditions $b^T * e = 1, b^T * c = 1/2, b^T * c^2 = 1/3, b^T * c^3 = 1/4, b^T * c^4 = 1/5, b^T * c^5 = 1/6$, The six stage order conditions are $Xe + v = c, Xc + v/2 = c^2/2, Xc^2 + v/3 = c^3/3, Xc^3 + v/4 = c^4/4, Xc^4 + v/5 = c^5/5, Xc^5 + v/6 = c^6/6$.

// Set up stage order conditions for C(6).

C1:= matadd(matadd(multiply(x,one),v),c,1,-1);

C2:= matadd(matadd(multiply(x,c),v,1,1/2),c2,1,-1/2);

C3:= matadd(matadd(multiply(x,c2),v,1,1/3),c3,1,-1/3);

C4:= matadd(matadd(multiply(x,c3),v,1,1/4),c4,1,-1/4);

C5:= matadd(matadd(multiply(x,c4),v,1,1/5),c5,1,-1/5);

```
C6:= matadd(matadd(multiply(x,c5),v,1,1/6),c6,1,-1/6);

//Impose C(6) conditions on the first stage.

// Q:=solve(C1[1],C2[1],C3[1],C4[1], C5[1],C6[1],c[1],v[1]);

// This gives two solutions: c[1]=v[1]=0 or c[1]=v[1]=1.

c[1] := 0; v[1] := c[1];

// Impose C(6) on the second stage.

Q:=solve(C1[2],C2[2],C3[2],C4[2],C5[2],C6[2],c[2],v[2],x[2,1]);

// This gives solutions:

c[2] := 1; v[2] :=1; x[2,1]:=0;

// Impose C(6) on the third stage.

Q:=solve(C1[3],C2[3],C3[3],C4[3],C5[3],C6[3],v[3],x[3,1],x[3,2],x[3,3],x[3,4], x[3,5]);

assign(Q);

// Impose C(6) on the fourth stage.

Q:=solve(C1[4],C2[4],C3[4],C4[4],C5[4],C6[4],v[4],x[4,1],x[4,2],x[4,3],x[4,4],x[4,5]);

assign(Q);

//Impose C(6) on the fifth stage.

Q:=solve(C1[5],C2[5],C3[5],C4[5],C5[5],C6[5],v[5],x[5,1],x[5,2],x[5,3],x[5,4],x[5,5]);

assign(Q);

//Impose C(6) on the sixth stage.

Q:=solve(C1[6],C2[6],C3[6],C4[6],C5[6],C6[6],v[6],x[6,1],x[6,2],x[6,3],x[6,4],x[6,5]);

assign(Q);

// Setup the coefficient matrix,M, for the determination of the weights.
```

```
for i from 1 to s do

M[1,i] := 1; M[2,i] := c[i]; M[3,i] := c2[i]; M[4,i] := c3[i]; M[5,i] := c4[i];

M[6,i] := c5[i];

RHS[i] := 1/i

od;

b:=linsolve(M,RHS);
```