A Generalized Runge Kutta Method of Order three

Per G. Thomsen *

June 4, 2002

Abstract

We present a numerical method for the solution of stiff systems of ODE's and index one DAE's. The type of method is a 4 stage Generalized Linear Method that is reformulated in a special Semi Implicit Runge Kutta Method of SDIRK type. Error estimation is by imbedding a method of order 4 based on the same stages as the method and the coefficients are selected for ease of implementation. The method has 4 stages and the stage order is 2. For purposes of generating dense output and for initializing the iteration in the internal stages a continuous extension is derived. The method is A-stable and we present the region of absolute stability and the order star of the order 3 method that is used for delivering the solution.

1 Introduction

The inspiration for the present study came from the results by Prothero and Robinson (1974) [7] where they discovered the order reduction of implicit Runge Kutta methods , when applied to stiff systems of differential equations. This observation led to new concepts of stability and eventually to a better understanding of the importance of stage order in connection with the overall properties of one-step methods.
Experiences from work on SDIRK-methods [8]on a three stage method of second order with stage order one, and following discussions with John C. Butcher, the idea of designing a generalized Runge Kutta method with stage order higher than one became interesting and this resulted in the method reported in this paper.

2 Derivation of the Generalized Runge Kutta Method

In the classical reference on Runge-Kutta and General Linear methods [5] Butcher introduces the generalized Runge Kutta Scheme.

*Informatics and Mathematical Modelling, DTU, Lyngby, Denmark

$$v_n = \tilde{A}u_n + h\tilde{B}f(v_n) \tag{1}$$
$$u_{n+1} = Au_n + hBf(v_n) \tag{2}$$

The scheme is characterized by the choice of the coefficient matrices and by chosing these we can obtain methods with the properties we are looking for. In schematic form our method can be expressed by the tableau

$\tilde{A}$	$\tilde{B}$
A	B

Schematic or matrix form of a general linear method.

Among the possible methods we chose one where the last function value from the previous step is used and three stages where the last stage is at the right endpoint of our step. This type of method is referred to as First Same As Last or FSAL. The choice will lead to the following form.

γ	0	0	1	a_{21}
a_{32}	γ	0	1	a_{31}
a_{42}	a_{43}	γ	1	a_{41}
0	0	1	1	b_1
0	0	0	0	0

General linear form of the method.

It is now possible to express the same scheme as a semi-implicit Runge Kutta scheme with an explicit first stage. In this form the method can be written as follows.

$$v_n = u_n + hAf(v_n) \tag{3}$$
$$u_{n+1} = u_n + hbf(v_n) \tag{4}$$

In this form each of the stages is a system of nonlinear equations that are solved stage by stage. The Butcher scheme for this is the following.

0	0			
c_2	a_{21}	γ		
c_3	a_{31}	a_{32}	γ	
1	b_1	b_2	b_3	γ
y_{n+1}	b_1	b_2	b_3	γ
e_{n+1}	d_1	d_2	d_3	d_4

The last line in the scheme contains the coefficients of the estimator for the local error obtained from imbedding an order 4 method in the scheme and computing the difference between the two solutions to obtain an estimate of the error for the method of order three.

2.1 Order conditions, stage order.

We want to satisfy the conditions for order 3 and stage order 2. The order conditions for the semi-implicit stages become:
Stage 2:

$$c_2 = a_{21} + \gamma = 2\gamma \Rightarrow a_{21} = \gamma; \tag{5}$$

Stage 3:

$$a_{32}c_2 + \gamma c_2 = \frac{c_3^2}{2}; \tag{6}$$

Stage 4:

$$b_2c_2 + b_3c_3 = \frac{1}{2} - \gamma; \quad b_2c_2^2 + b_3c_3^2 = \frac{1}{3} - \gamma; \tag{7}$$

2.2 Order conditions, method order.

The order condition for the imbedded error estimator, asking for the solution based on all the stages to be a fourth order solution leads to the equation.

$$d_2(\frac{c_2^3}{3} - \gamma c_2^2) + d_3(\frac{c_3^3}{3} - a_{32}c_2^2 - \gamma c_2^2) = 0 \tag{8}$$

This condition may be expressed by means of the errors from stages 2 and 3.

$$d_2 e_2 + d_3 e_3 = 0 \tag{9}$$

where e_2 is the error from stage 2 and e_3 is the error from stage 3.

$$e_2 = (\frac{c_2^3}{3} - \gamma c_2^2) = \frac{c_2^3}{3} - \gamma(c_3^3 - c_2^2). \tag{10}$$

$$e_3 = (\frac{c_3^3}{3} - a_{32} c_2^2 - \gamma c_2^2) = -\frac{4}{3}\gamma^3. \tag{11}$$

3 Determination of the value γ

In order to define the method we must find the value of γ. This may be done by specifying the property that our method must be L-stable. This requires [2] that the following condition is satisfied:

$$\left[c_3(1 - 6\gamma + 12\gamma^2) - 2\gamma(2 - 9\gamma + 12\gamma^2)\right](c_3 - 2\gamma) = 0 \tag{12}$$

The choice $c_3 = 2\gamma = c_2$ is not desirable and we are left with the other condition

$$c_3(1 - 6\gamma + 12\gamma^2) - 2\gamma(2 - 9\gamma + 12\gamma^2) = 0 \tag{13}$$

From which we find $\gamma = \frac{5}{12}$ as a reasonable choice giving $c_3 = \frac{10}{21}$.
We now use the conditions (4) to (8) and the method is defined and may present the Butcher tableau:

0	0			
$\frac{5}{6}$	$\frac{5}{12}$	$\frac{5}{12}$		
$\frac{10}{21}$	$\frac{95}{588}$	$-\frac{5}{49}$	$\frac{5}{12}$	
1	$\frac{59}{600}$	$-\frac{31}{75}$	$\frac{539}{600}$	$\frac{5}{12}$
y_{n+1}	$\frac{59}{600}$	$-\frac{31}{75}$	$\frac{539}{600}$	$\frac{5}{12}$
e_{n+1}	$\frac{55}{600}$	$\frac{55}{75}$	$\frac{-245}{600}$	$\frac{-5}{12}$

Coefficients for the 4-stage GERK - method of order 3.

4 Stability Properties

In order to obtain suitable properties for the solution of index1 DAE's we must at least satisfy the conditions for A-stability. This can be found from considering the test equation and the resulting rational approximation to the exponential function. A straightforward calculation leads to the rational function.

$$\mathbf{R}(z) = \frac{P(z)}{Q(z)} \tag{14}$$

The rational fraction that has been obtained by the GERK-method is given by

$$\mathbf{R}(z) = \frac{1 - \frac{1}{4}z + \frac{11}{48}z^2 - \frac{17}{1728}z^3}{(1 - \frac{5}{12}z)^3} \tag{15}$$

This rational function is a third order approximation to $exp(z)$ and the stability properties of the GERK method is closely related to the acceptability of this approximation.

4.1 Stability region

The region of absolute stability is shown in the figure below.

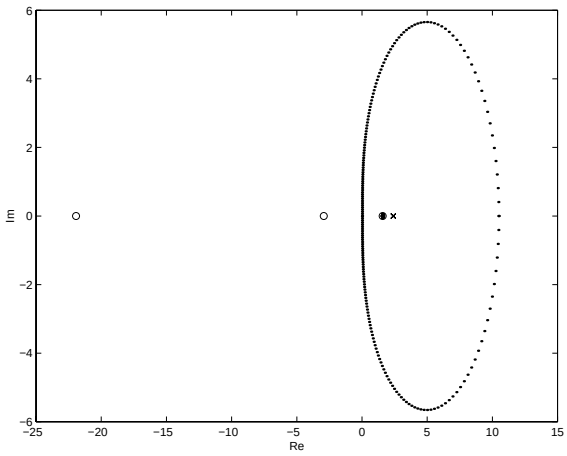

Figure 1: Stability region for Gerk(4)

4.2 Order Star

Following the analysis of [4] we apply the theory of order stars to verify the A-stability property of the GERK-scheme. The verification is equivalent to observing that no branch of the order star crosses the imaginary axis. For the

purpose of presenting the overall picture of the properties of the rational approximation we present the order star of type 1 in the figure below.
By inspecting the order star we see that no "finger" crosses the imaginary axis

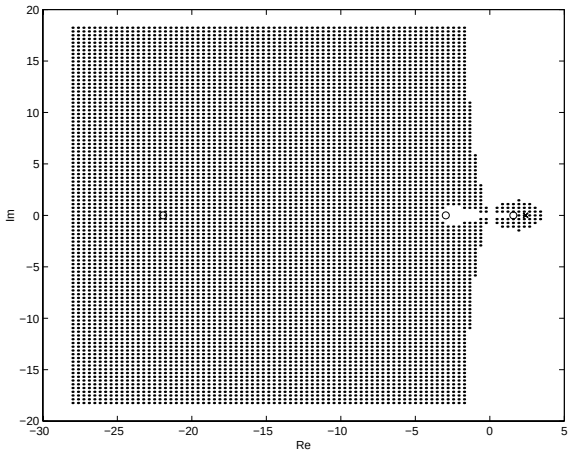

Figure 2: Order star for Gerk(4)

and the method is indeed A-stable.

5 Continuous Extension

In the GERK-method every stage except the first involves the solution of a system of nonlinear equations. For this system some kind of iterative solver is applied and in order to get the iterations started we need a starting guess. For that purpose and also to generate dense output of the solution we supply a continuous extension or interpolation formula of the form

$$u(t_n + \theta h) = u_n + h \sum_{s=1}^{r} d_s(\theta) f(v_s) \tag{16}$$

The derivation of this formula for the interpolation follows the same theory as was developed in [13] where the methods were all explicit. The basic ideas however are very similar and have been used in [11], here we give the resulting coefficient matrix.

$$d_s(\theta) = d_{s,1}\theta + d_{s,2}\theta^2 + d_{s,3}\theta^3 \tag{17}$$

here the coefficients are given in the form of three vectors.

d_1	$\frac{29}{244}$	$\frac{-1620}{671}$	$\frac{145}{44}$
d_2	$\frac{-141}{244}$	$\frac{5832}{671}$	$\frac{-357}{44}$
d_3	$\frac{216}{244}$	$\frac{-3888}{671}$	$\frac{216}{44}$

Polynomial coefficients $d_s(\theta)$ for the continuous extension.

6 Implementation

For testing the GERK method we have developed an Object-Oriented C++ program package SDIRK [10] that is intended for solution of stiff ODE's. In this section we describe how to use this implementation and some of the strategies that have been used for stepsize and convergence control. The basic ideas are similar to those in Gustavsson [6] and have been developed in the GODESS-project [3]. This implementation is still a research code mainly for use in research and has proven helpful also in teaching. The basic ideas in the Object Oriented software development that are applied has been helpful to illustrate the basic ideas in ODE-solvers.
A third implementation in Matlab has been used for comparing control strategies. This code includes facilities for switching between different methods and stepsize control strategies and is made available to be copied from my homepage, *www.imm.dtu.dk/~pgt/GERK*. The code is mainly intended for testing but may be used for the solution of stiff ODE's in general since it is indeed very efficient and flexible. A documentation is found at the same website.

6.1 PI-control of stepsize.

In order to optimize the stepsize-control strategy a Matlab implementation of the method has been implemented under the name gerk.m. As a special feature the gerk code has the possibility to choose from a number of stepsize-control strategies. These have all been forged in the same template that is derived from the PI-controllers that have been developed by Søderlind [9]. The basic formula for estimating the stepsize for the current step from data gathered in previous steps is the following.

$$h_{n+1} = h_n(\frac{\tau}{e_n})^{\beta_1}(\frac{\tau}{e_{n-1}})^{\beta_2}(\frac{h_n}{h_{n-1}})^{-\alpha_2} \tag{18}$$

For the purpose of relating this control to the traditional step-control strategy and others from the litterature we give a couple of examples on typical choices of parameters.

- ordinary step control
  $\alpha_2 = 0, \beta_1 = 1/3, \beta_2 = 0.$

- Watts step control
  $\alpha_2 = 0, \beta_1 = 1/3, \beta_2 = 1/3.$
- Choice used by Gustavsson
  $\alpha_2 = 1, \beta_1 = 0.3/3, \beta_2 = 0.4/3.$
- Second order PI-control
  $\alpha_2 = 1/2, \beta_1 = 1/6, \beta_2 = 1/6.$

For further discussion on the properties of the PI-control strategies we refer to [9]. Results from testing the four strategies is found in the example-section below.

6.2 The structure of the SDIRK code

A general ODE-solver has been implemented i C++, named SDIRK. The Object Oriented implementation makes use of the high degree of structuring that is offered by the C++ environment.
For details we refer the reader to the users manual of SDIRK [10] The structure of the program and the connectivity is shown in the figures below.

6.3 The implementation in GODESS.

Until recently, the testing of ODE/DAE solvers has been limited to comparing software. The complex process of developing software from a mathematically specified method entails constructing control structures and objectives, selecting termination criteria for iterative methods, choosing norms and many more decisions. Most software constructors have taken a heuristic approach to these design choices, and as a consequence two different implementations of the same method may show significant differences in performance. Yet it is common to try to deduce from software comparisons that one *method* is better than another. Such conclusions are not warranted, however, unless the testing is carried out under true *ceteris paribus* conditions. Moreover, testing is an empirical science and as such requires a formal *test protocol*; without it conclusions are questionable, invalid or even false. We argue that ODE/DAE software can be constructed and analyzed by proven, "standard" scientific techniques instead of heuristics, and that each solver should have a complete specification of its algorithmic content. Further, we indicate that a test protocol can be devised such that firm conclusions may be drawn from careful testing. The goal is reproducibility as well as improved software quality.

6.3.1 Results from testing in GODESS

The implementation that uses Krylov subspace techniques for the solution of the linear subproblems that arise is developed with the intention to obtain knowledge about the performance of different iterative algorithms and different types of preconditioning. Godess proved to be an efficient platform for making such

comparisons. Details can be found in [12] here we bring a comparison of the GERK-method and seven other methods for stiff systems.
The testproblem is a 2D model of the production of Ozone in the stratosphere [1]. The model consists of two coupled PDE's

$$\begin{aligned}
\frac{\partial c^i}{\partial t} &= K_h \frac{\partial^2 c^i}{\partial x^2} + \frac{\partial}{\partial z}\left(K_v(z)\frac{\partial c^i}{\partial z}\right) \\
&\quad + V\frac{\partial c^i}{\partial x} + R^i(c^1, c^2, t) \qquad (i = 1, 2) \qquad (19) \\
K_h &= 4 * 10^{-6}, \quad K_v(z) = 10^{-8} e^{z/5}, \quad V = 0.01 \\
R^1(c^1, c^2, t) &= -k_1 c^1 - k_2 c^1 c^2 + k_3(t) * 7.4 * 10^{16} + k_4(t) c^2 \\
R^2(c^1, c^2, t) &= k_1 c^2 - K_2 c^1 c^2 - k_4(t) c^2 \\
k_1 &= 6.031, \quad k_2 = 4.66 * 10^{-16} \\
k_3 &= \begin{cases} exp(-22.62/sin(\pi t/43200)) & , \quad t < 43200 \\ 0 & , \quad otherwise \end{cases} \\
k_4 &= \begin{cases} exp(-7.601/sin(\pi t/43200)) & , \quad t < 43200 \\ 0 & , \quad otherwise \end{cases}
\end{aligned}$$

Here the concentrations of oxygen c^1 and Ozone c^2 are the variables and the equations represent reaction-transport with horizontal diffusion and advection. The boundaries are $0 \leq x \leq 20$, $30 \leq z \leq 50$, $0 \leq t \leq 86400$. The Jacobian has a banded structure reflecting the discretization used for the derivatives on a uniform rectangular grid.The system is stiff and the spectrum of the Jacobian matrix can be found in [1]. The absolute tolerance is 10^-3 and the relative tolerance is 10^-5. For the Krylov method we have selected an Arnoldi type type of INcomplete Orthogonalization in order to compare with the results from [1]. The preconditioning is a simple diagonal preconditioning for this test. Other preconditionings are available in the GODESS implementation. From the results

Method	Steps	GKD	GNI	AMV	ALA
ESDIRK23a	1052	2.56	2.04	16505	55.7
ESDIRK23b	865	2.75	2.10	15015	54.5
ESDIRK45a	257	3.62	2.53	11731	57.0
ESDIRK45b	239	3.62	2.74	11800	57.7
Hairwann	1430	2.49	2.77	34500	123.2
sd34var	414	3.02	2.66	14800	57.7
GERK	515	3.16	2.55	12466	51.1
BDF	610	2.61	2.18	3476	10.5

Table 1: Comparison of different methods and the dimension of the Krylov subspace used in the iterations. Number of time steps (Steps), Average Dimension of subspace (GKD) ,Average Number of Iterations (GNI), Number of Matrix-vector Operations (AMV) , Other Linear algebra operations (ALA) in millions.

we see, that the GERK method is the most efficient of the one-step methods while the BDF-method is the overall winner since it does less work pr. step although it takes more steps than GERK. The general impression is that the dimension of the subspace is indeed very small compared to the size of the system on a 10x10 grid this is 100 and the dimension of the Krylov subspace is near 3. This shows that in general for this type of systems the Krylov type of linear solver is very efficient, it needs very few iterations to converge to the level of accuracy that is prescribed.

6.4 Using the SDIRK - solver

We will illustrate the general usage of the SDIRK package by applying the code on the Van Der Pol equation. Two tests have been carried out in order to check the performance for different types of problems.

6.4.1 Example: the Van Der Pol equation

The Van Der Pol problem is a well known testexample that may be used to illustrate the performance of a piece of code for solving stiff and non-stiff ODE's. The Van Der Pol system describes a model of an electronic oscillator with a nonlinear element and has one parameter μ that may be varied or even made dependent on the integration time.

$$y'' - \mu(1 - y^2)y' + y = 0 \tag{20}$$

This is a second order ODE but may be transformed into two coupled first order ODE's which then becomes partly stiff when the parameter μ is large. In the first graph is shown how the total amount of work depends on the error tolerances that are prescribed. Tolerances are varied between 0.1 and 10^{-10} the results show that the work is roughly proportional to the required accuracy. The parameter $\mu = 200$ has been chosen to make the problem moderately stiff.
In the second test the parameter μ in the Van Der Pol equation is varied over the values $0 < \mu < 10000$ the work is measured as the number of steps needed for the solution over a fixed interval $[0, 1000]$, this will in all cases include the initial transient. The relative error tolerance is 10^{-6} while the absolute tolerance is 10^{-8} this is a quite severe test ranging over non-stiff to very stiff.
We see that the number of steps varies a lot with the μ-value and since the stiffness of the problem is roughly equivalent to the size of μ it is obvious that the method is coping very well with stiffness but at the same time will be relatively expensive for non-stiff problems. This is due to the fact that the order of the method is relatively low.
A third test is performed for the Van Der Pol equation with $\mu = \mu_o exp(t - t_o)$ This gives a severe test for the solver since the problem gradually becomes more and more stiff and harder to solve when time increases. The solution may be pictured in the Phase-plane and is shown below as it was generated by the solver. The value of μ where the solver gave up was close to 35000 (this was

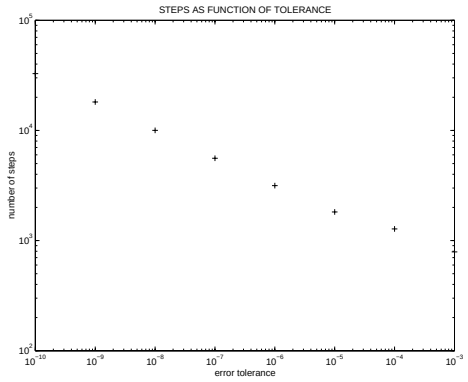

Figure 3: Steps versus tolerance test, log-log plot.

enforced when the stepsize became smaller than 10^{-8}).

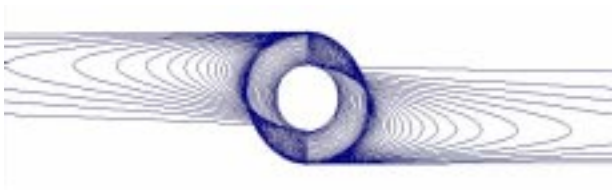

Figure 4: Van Der Pol solutions for variable μ.

6.4.2 Testing the GERK - solver

Using the above test example we can illustrate the performance of different choices of stepsize strategies. The table shows the number of accepted steps (steps), the number of failed steps (FSTP) and the percentage of failed steps (PFSTP) for the four different control strategies and for two different values of the relative error tolerance.
We see from these results that the number of failed steps are very important for the performance of the strategies and the assymptotic is best in both the stiff and the non-stiff case when the tolerance is moderate (reps= 10^{-4}) while the second order control is better for tighter tolerance (reps = 10^{-6}). However an inspection of the stepsize sequence gives a different picture altogether since the Gustavsson and second order strategies lead to a much smoother stepsize sequence than the assymptotic and this in most cases is what is wanted to make the stepsize control robust.

Method	Steps	FSTP	PFSTP	Steps	FSTP	PFST
$\mu = 20$, nonstiff	10^{-4}			10^{-6}		
assymptotic	880	146	14.23	2634	267	9.20
Watts	1077	280	20.63	3525	936	20.98
Gustavsson	1052	221	17.36	2985	372	11.08
Second order	919	167	15.38	2752	413	13.05
$\mu = 200$, stiff	10^{-4}			10^{-6}		
assymptotic	1615	286	15.04	4982	673	11.91
Watts	2011	535	21.01	6646	1877	22.02
Gustavsson	1818	397	17.92	5125	716	12.26
Second order	1626	336	17.13	4779	534	10.05

Table 2: Comparison of stepsize-strategies on the VanDerPol equation

When integrating large systems of ODEs or DAEs where Jacobian matrices are expensive to calculate and decompose the reduction in the number of needed Jacobian matrices will translate into a reduction of the CPU-time needed to perform the integration. This effect is not tested in the example from table 2. but included in the data in figure3.

7 Conclusion

The present work introduces a Generalised Runge Kutta solver for ODE's and DAE's of third order with four stages and stage order two. The method has been implemented and tested in two different implementations. The GODESS package is a test-environment for ODE-solvers and the properties of the GERK-method are demonstrated to be very promising especially for stiff systems. Qualitatively the tolerance proportionality in the implementation turns out to be very good.
The second implementation SDIRK is a general purpose ODE-solver where different stepsize strategies are available and the tests have shown that PI-control strategies are marginally better that the traditional stepsize control based purely on the assymptotic error behaviour. This is demonstrated on several tests on the VanDerPol equation.
Some comparisons using a Matlab implementation have led to the conclusion that a second order PI-control is a very good choice for a general ODE-solver that may be used for stiff as well as nonstiff problems for reasonably strict error tolerances.
The use of the GERK method in connection with retarded ODE's is presented in [11]. The use of the GERK method for general DAE's is referred to a later paper.

References

[1] Peter N. Brown and Alan C. Hindmarsh. Matrix-free methods for stiff systems of ODE's. *SIAM J. Numer. Anal.*, 23(3):610–638, 1986.

[2] E.Hairer and G.Wanner. *Solving Ordinary Differential Equations II*, volume 14 of *Springer series in computational Mathematics*. Springer Verlag, 1991.

[3] H.Olsson. Runge kutta solution of initial value problems. *LTH*, 1:1–178, 1998.

[4] A. Iserles and S.P. Nørsett. *Order Stars*, volume 2 of *Applied Mathematics and Mathematical Computation*. Chapman & Hall, 1991.

[5] J.C.Butcher. *The numerical analysis of ordinary differential equations. Runge Kutta and general linear methods.* J.Wiley & Sons, 1987.

[6] K.Gustafsson. Control of error and convergence in ode-solvers. *LTH*, 1, 1992.

[7] A. Prothero and A. Robinson. On the stability and accuracy of one-step methods for solving stiff systems of ordinary differential equations. *Math Comp*, 28:145–162, 1974.

[8] S.P.Nørsett and P.G.Thomsen. Imbedded sdirk-methods of basic order three. *BIT*, 24:634–646, 1984.

[9] G. Soderlind. The automatic control of numerical integration. *CWI Quaterly*, 11(1):55–74, 1998.

[10] E. Østergård. Documentation for the sdirk c++ solver. *IMM*, pages 1–16, 1998.

[11] P.G. Thomsen. Numerical solution of retarded differential equations. *IMM-REP-99-16*, 99(16), 1999.

[12] P.G. Thomsen and N.H.Bjurstrøm. Krylov subspace methods for the solution of large systems of ode's. *NATO science series*, 57(16):325–338, 1998.

[13] S.P.Nø rsett W.A.Enright, K.R.Jackson and P.G.Thomsen. Interpolants for runge-kutta formulas. *ACM TOMS*, 12(3):193–218, 1986.