Investigation on the Most Efficient Ways to Solve the Implicit Equations for Gauss Methods in the Constant Stepsize Setting

Mohd Hafizul Muhammad
Mathematics Department
Faculty of Science and Mathematics
Universiti Pendidikan Sultan Idris
Sultan Azlan Shah Campus, Proton City
Tanjong Malim 35900 Perak, Malaysia

Annie Gorgey
Mathematics Department
Faculty of Science and Mathematics
Universiti Pendidikan Sultan Idris
Sultan Azlan Shah Campus, Proton City
Tanjong Malim 35900 Perak, Malaysia

Copyright © 2018 Mohd Hafizul Muhammad and Annie Gorgey. This article is distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract

This article focuses on the implementation of the implicit Runge-Kutta (RK) Gauss methods. The Jacobian evaluations using full Newton (FN) is laborious, expensive and therefore suffer iteration errors that leads to less efficient implementation. The use of a very small stepsize can accumulate the round off errors and destroy the solutions. Therefore, compensated summation (CS) is required to minimize these round-off errors. This research investigate the best implementation technique for Gauss methods using simplified Newton-Raphson (SN) and FN together with compensated summation (CS). The investigation involved with four different implementation techniques that are the combinations of FN, SN with and without CS. The numerical results on Brusselator...
and Oregenator problems showed that the implementation of G2 and G3 using SN and FN are efficient with compensated summation especially when the stepsize used is very small. This article also showed that the most efficient ways to implement G2 and G3 is using SN for stiff problems and FN for nonstiff problems.

Keywords: Gauss methods, simplified Newton, implicit equations, compensated summation

1 Introduction

Runge-Kutta (RK) methods are one big family in numerical methods. RK methods can be categorized into two types that are explicit and implicit methods. The explicit and implicit RK methods are different in terms of the number of stages, stability, order and the style of the coefficient $A$ of the RK methods. If $A$ is lower triangular such as the explicit methods, then the implementation is cheaper. Else if $A$ is a fully block matrix, such as the Gauss method, the implementation will become costly. The implicit RK methods are better than the explicit RK methods because they are $A$ stable and therefore suitable in solving stiff problems [1]. However, one of the most important issue in the implementation of implicit RK method is solving the nonlinear equations which makes the implementation costly. In order to reveal the lower cost of implementation, several Newton based approach were studied since 1970 [4].

This article focuses more on the implementation strategy of the 2-stage Gauss (G2) and 3-stage Gauss (G3) methods in solving stiff problems.

Generally, the $s$-stage Runge-Kutta method is defined by the following equations:

$$
Y^{[n]} = e \otimes y_{n-1} + h(A \otimes I_N)F(x_{n-1} + ch, Y^{[n]}), \quad y_{n} = y_{n-1} + h(b^T \otimes I_N)F(x_{n-1} + ch, Y^{[n]}),
$$

where $\otimes$ refers to the Kronecker product, $e = (1, \ldots, 1)^T$ and $I_N$ an identity matrix of order $N$ for an initial value problem $y'(x) = f(x, y)$, $y(x_0) = y_0$.

The update is known as $y_{n}$ while $Y^{[n]}$ is the vector of internal stages with

$$
Y^{[n]} = \begin{bmatrix} Y_1^{[n]} \\ \vdots \\ Y_s^{[n]} \end{bmatrix}, \quad F = \begin{bmatrix} f(x_{n-1} + c_1 h, Y_1^{[n]}) \\ \vdots \\ f(x_{n-1} + c_s h, Y_s^{[n]}) \end{bmatrix}.
$$
The matrix $A$ is used to find the internal stages of RK methods. Let

$$G(Y) = Y - e \otimes y_{n-1} - h(A \otimes I_N)F(Y).$$

(3)

The solutions to the implicit equations (2) by the Newton method yields

$$\Delta Y = -G(Y)[DG(Y)]^{-1},$$

(4)

where $\Delta Y = (Y^{[k+1]} - Y^{[k]})$ and $DG(Y)$ contain the block diagonal matrix of the Jacobian such that

$$DG(Y) = (I_N - h(A \otimes J)).$$

(5)

The Jacobian matrix, $J \approx \frac{\partial f}{\partial y}(x_{n-1} + c_i h, Y)$ is given by

$$J = \begin{bmatrix}
J(x_{n-1} + c_1 h, Y) \\
\vdots \\
J(x_{n-1} + c_s h, Y)
\end{bmatrix}.$$

(6)

Equation (4) is a linear system of dimension $s \times N$ ($s$ stages and $N$ differential equations). Implicit RK methods are difficult to solve because one had to solve the nonlinear algebraic system of order $sN$.

To computational cost to solve the system of nonlinear equations involves the following factors.

- Evaluation of the $F$ and $G$,
- Evaluation of the Jacobian, $J$,
- Evaluation of $DG$,
- LU factorization of the iteration matrix $DG$ and
- Back substitution to get the Newton increment vector, $\Delta Y$.

2 Implementation Technique

The computation cost in solving the nonlinear part of the internal stage of the implicit Runge-Kutta methods are expensive [3] because operations like LU factorization and backward substitutions are involved. Before 1967, fixed-point iteration was a command method when solving the nonlinear system of equations. However, this approach is no longer popular because the method
crush the stability of the implicit Runge-Kutta [5]. The fixed point iteration is only popular when solving nonstiff problems. In 1970, the advantage of Newton-Raphson iteration method revealed by Liniger and Willoughby [9]. To solve the nonlinear system of equations using Newton-Raphson approach, it involve the evaluation of the Jacobian matrix. Taking the inverse of the Jacobian is very expensive. If the Jacobian is evaluated at every iteration then this approach is also known as full Newton (FN). Evaluating Jacobian for a large scale ordinary differential equations needs high computational cost and the technique also contributes to the amplification of the iteration error [10]. In contrast to that, if the Jacobian is only evaluated once at the start of the iteration, then the approach is known as simplified Newton (SN). Hairer and Wanner had implemented the simplified Newton on Radau IIA method of order-5 and the approach has become popular ever since in solving stiff problems [5]. This also has opened the doors of many researchers to find the best implementation strategy that minimizes the computational cost in solving the nonlinear system of equations by the implicit methods [3, 7]. Although SN is efficient than FN, but for certain types of problems FN still give efficient results. Therefore, this paper is important because it give a detailed explanation on the implementation technique using simplified Newton and full Newton for the solution of stiff problems.

The Algorithm 1 and Algorithm 2 represent the pseudo code for full Newton method and simplified Newton method respectively that have been implemented in our code. The tolerance, $\tau = 1 \times 10^{-10}$ is chosen based on the problems that need to be solved and it is also depends on the desired overall accuracy. The starting value is the initial estimates given for the problem.

Algorithm 1 and Algorithm 2 are efficient enough to solve certain problems. The numerical experiments done for many types of problems concluded that Algorithm 1 is efficient to solve nonstiff problems. However for stiff problems, Algorithm 2 is advantageous over Algorithm 1. However, to solve stiff problems, one required to use a very small stepsize especially in the constant stepsize setting. Decreasing stepsize increases the round-off errors and thus destroy the solutions. In order to fix the round-off errors, compensated summation is introduced in the next section.

## 3 Enhancement by compensated summation

The solution produce by computer arithmetic is not exactly same as the exact arithmetic because of the rounding occurs in the computation [2]. The finite precision arithmetic on the computer is always rounding the value to certain decimal place. As a result of the rounding, the decimal place will become slightly shorter. The accumulation of round-off errors in the computation may
Algorithm 1 FULL NEWTON ITERATION

Set $Y = y$, $\tau = 1 \times 10^{-10}$, $N$ be any positive integers.

for $i = 1$ to $N$
    Evaluate $F$
    Evaluate $G$
    Evaluate $J$
    Evaluate $D_G$
    Evaluate $\Delta Y$
    if $(\|\Delta Y\|_\infty) < \tau \cdot \max(1, (\|Y\|_\infty))$
        $Y \leftarrow Y + \Delta Y$
        break

$y_{\text{new}} \leftarrow \text{update } y_n$
$x_{\text{new}} \leftarrow x + h$
return $(y_{\text{new}}=y)$
return $(x_{\text{new}}=x)$

Algorithm 2 SIMPLIFIED NEWTON ITERATION

Set $Y = y$, $\tau = 1 \times 10^{-10}$, $N$ be any positive integers.
Evaluate the $J$
Evaluate $D_G$

for $i = 1$ to $N$
    Evaluate $F$
    Evaluate $G$
    Evaluate $\Delta Y$
    if $(\|\Delta Y\|_\infty) < \tau \cdot \max(1, (\|Y\|_\infty))$
        $Y \leftarrow Y + \Delta Y$
        break

$y_{\text{new}} \leftarrow \text{update } y_n$
$x_{\text{new}} \leftarrow x + h$
return $(y_{\text{new}}=y)$
return $(x_{\text{new}}=x)$
produce solution which is far different from the exact solution. In the computational practice, the numerical error is computed by the difference between the numerical solution and the exact solution. The presentation of the efficiency graph could determine that whether the implementation of the IRK method is infected by the significant round-off errors. The round-off errors occurred will cause a big change in the slope of the efficiency graph, normally the slope will change from negative slope to positive slope or zero slope. However, not all computation significantly infected by the round off error. The use of incredibly small stepsize is the factor that cause the significant round off error and destruct the computation. The small stepsizes is needed in obtaining highly accurate numerical solution for some complicated problems especially in the constant stepsize setting. However, it is found that the implementation method become less efficient when stepsize used is very small. In 1951, Gill-Moller algorithm had been introduced to the computational analysis [6]. The effort aimed to reduce the numerical error growth which is contributed by the round off error. In later year, the term compensated summation was used by N. Higham [8]. Butcher had given a detailed explanation about the mechanism of this approach [2] for implicit Euler method. Therefore, compensated summation is introduced to the algorithm to see the results of the solutions.

The modification done to Algorithm 1 and Algorithm 2 are by introducing the terms \( r, rx \) and \( z \) where \( Z = Y - y \). It is suggested in [5] as well as in [2] that the influence of round-off errors can be reduced by using smaller quantities \( z \). The aim of compensated summation is to capture the round-off errors in each individual step. The \( r \) quantity is where the round-off errors are stored for the \( y \)-values and \( r_x \) are the round-off errors stored for the \( x \)-values.

The accumulation of this errors can be reduced by working with smaller quantities:

\[
Z = Y - e \otimes y_{n-1}.
\]

The stage equation (1) becomes

\[
Z = h(A \otimes I_N)F(Z + e \otimes y_{n-1}).
\]

The update \( y_n \) then becomes

\[
y_n = y_{n-1} + (b^TA^{-1} \otimes I_N)Z.
\]

The solution to the implicit equations by the Newton methods with compensated summation now yields

\[
D_G(Z)\Delta Z = -G(Z),
\]  

(7)
where $D_G(Z) = (I_N \otimes I_S) - h(A \otimes J)$, and $\Delta Z = (Z^{[k+1]} - Z^{[k]})$, and

$$G(Z) = Z - e \otimes y_{n-1} - h(A \otimes I_N)F(Z).$$

Several test to the Jacobian has been done, and it is proven that for simplified Newton the choice of the Jacobian with $J \approx \frac{\partial f}{\partial y}(x_0, y_0)$ gives the best approximations. Since the Jacobian is only evaluated once at the beginning of the iteration, therefore the increment on $Z$ does not help much in the computational.

The Algorithm 3 represent the pseudo code for the simplified Newton applied together with compensated summation.

### Algorithm 3 SIMPLIFIED NEWTON WITH COMPENSATED SUMMATION

Set $Z = 0, r = 0, rx = 0, Z = Y - y, \tau = 1 \times 10^{-10}$ and $N$ be any positive integers. Evaluate the Jacobian $J$

Evaluate $D_G$

for $i = 1$ to $N$

Evaluate $F$

Evaluate $G$

Evaluate $\Delta Z$

if $(\|\Delta Z\|_{\infty}) < \tau \cdot \max(1, (\|Z\|_{\infty}))$

$Z \leftarrow Z + \Delta Z$

break

ynew $\leftarrow$ update $y_n$

xnew $\leftarrow x + h$

return (ynew=$y$)

return (xnew=$x$)

4 Numerical experiments

The numerical experiments are done for the 2-stage Gauss (G2) and 3-stage Gauss (G3) using full Newton (FN) and simplified Newton (SN) with and without compensated summation (CS) as given in Table 1. The efficiency graphs measuring the computational time (CPU) are shown in all the plots.
The efficiency of SN and FN with and without CS are obtained by plotting the loglog plot of errors versus cputime. The stepsize is decreased by half after every iterations and the round-off errors are minimized using compensated summation technique.

<table>
<thead>
<tr>
<th>FNWCS</th>
<th>full Newton and without compensated summation</th>
</tr>
</thead>
<tbody>
<tr>
<td>FNCS</td>
<td>full Newton and with compensated summation</td>
</tr>
<tr>
<td>SNWCS</td>
<td>simplified Newton and without compensated summation</td>
</tr>
<tr>
<td>SNCS</td>
<td>simplified Newton and with compensated summation</td>
</tr>
</tbody>
</table>

Table 1: Implementations tactics

Although only constant stepsize is considered, it is important to see the behaviour of G2 and G3 using SN and FN with and without CS. This result can provide the best choice of implementation technique to used in the future when using variable stepsize. Compensated summation may not be required for variable stepsize since the stepsize will be automatically adjusted. However, it is not known yet how bad the solutions can be effected and we wish to extend the results for variable stepsize setting in the future.

The numerical results are given for the following problems taken in [5].

**Oreganator problem**

Oregonator model is proposed for the Belousov-Zhabotinskii (BZ) reaction. The BZ reaction is one of a class of reactions that serve as a classical example of non-equilibrium thermodynamics, resulting in the establishment of a nonlinear chemical oscillator. OREGO is a nonlinear stiff system of 3 dimensions [5].

\[ y_1' = 77.27(y_2 + y_1(1 - 8.375 \times 10^{-6}y_1 - y_2)), \]
\[ y_2' = \frac{1}{77.27}(y_3 - (1 + y_1)y_2), \]
\[ y_3' = 0.161(y_1 - y_3), \quad x \in [0, 3600], \]

with \( y_1(0) = 1, y_2(0) = 2 \) and \( y_3(0) = 3 \). The problem is integrated to \( x_n = 6 \) using initial stepsize \( h = 0.03 \) for G2 and G3.

**Brusselator problem**

Brusselator is a problem in a chemical reaction which is oscillating catalyst in automatic system modelled by Ilya Prigogine and his collaborators at the Free University of Brussels [5]. The problem is defined by two ODEs

\[ y_1' = 1 + y_1^2y_2 - 4y_1, \]
\[ y_2' = 3y_1 - y_1^2y_2, \]
with \( y_1(0) = 1.5 \) and \( y_2(0) = 3 \). The problem is integrated to \( x_n = 5 \) using \( h = 0.05 \) for G2 and G3.

<table>
<thead>
<tr>
<th>Problems/Methods</th>
<th>G2</th>
<th>G3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oreganator</td>
<td>SNCS</td>
<td>SNCS</td>
</tr>
<tr>
<td>Brusselator</td>
<td>SNCS</td>
<td>FNCS</td>
</tr>
</tbody>
</table>

Table 2: Efficiency of G2 and G3 for the Oreganator and Brusselator problems

![Figure 1: The efficiency of G2 and G3 applied to the Oreganator problem.](image1)

![Figure 2: The efficiency of G2 and G3 applied to the Brusselator problem.](image2)

5 Conclusion

In conclusion, based on the numerical results and some other problems that are not mentioned in this article, SNCS is important to be implemented when
solving stiff problems especially when a very small stepsize is required. However, FNCS is enough to give efficient results in solving nonstiff and midly stiff problems. In both cases, compensated summation is a must and needed to be implemented in all the methods so that round-off errors would not destroy the solutions.

**Acknowledgments.** The authors would like to thank the Ministry of Higher Education in Malaysia for providing the funding. This research was supported by the research grant FRGS, Vote No: 2015-0158-104-02.

**References**


Received: December 1, 2017; Published: January 17, 2018