Hindawi Publishing Corporation
Discrete Dynamics in Nature and Society
Volume 2008, Article ID 790619, 38 pages
doi:10.1155/2008/790619

Research Article

Accelerated Runge-Kutta Methods

Firdaus E. Udwadia[1] and Artin Farahani[2]

[1] *Departments of Aerospace and Mechanical Engineering, Civil Engineering, Mathematics, and Information and Operations Management, 430K Olin Hall, University of Southern California, Los Angeles, CA 90089-1453, USA*
[2] *Department of Aerospace and Mechanical Engineering, University of Southern California, Los Angeles, CA 90089-1453, USA*

Correspondence should be addressed to Firdaus E. Udwadia, fudwadia@usc.edu

Received 31 December 2007; Revised 8 March 2008; Accepted 27 April 2008

Recommended by Leonid Berezansky

Standard Runge-Kutta methods are explicit, one-step, and generally constant step-size numerical integrators for the solution of initial value problems. Such integration schemes of orders 3, 4, and 5 require 3, 4, and 6 function evaluations per time step of integration, respectively. In this paper, we propose a set of simple, explicit, and constant step-size Accerelated-Runge-Kutta methods that are two-step in nature. For orders 3, 4, and 5, they require only 2, 3, and 5 function evaluations per time step, respectively. Therefore, they are more computationally efficient at achieving the same order of local accuracy. We present here the derivation and optimization of these accelerated integration methods. We include the proof of convergence and stability under certain conditions as well as stability regions for finite step sizes. Several numerical examples are provided to illustrate the accuracy, stability, and efficiency of the proposed methods in comparison with standard Runge-Kutta methods.

1. Introduction

Most of the ordinary differential equations (ODEs) that model systems in nature and society are nonlinear, and as a result are often extremely difficult, or sometimes impossible, to solve analytically with presently available mathematical methods. Therefore, numerical methods are often very useful for understanding the behavior of their solutions. A very important class of ODEs is the initial value problem (IVP):

$$\begin{aligned} \frac{d\tilde{\mathbf{y}}(t)}{dt} &= \tilde{\mathbf{f}}\left(t, \tilde{\mathbf{y}}(t)\right), \quad t_0 \le t \le t_N, \\ \tilde{\mathbf{y}}(t_0) &= \tilde{\mathbf{y}}_0, \end{aligned} \tag{1.1}$$

where $\tilde{\mathbf{y}}$, $\tilde{\mathbf{y}}_0$, and $\tilde{\mathbf{f}}$ are $(m-1)$-dimensional vectors, and t denotes the independent variable, time.

A constant step numerical integrator approximates the solution $\tilde{\mathbf{y}}(t_1)$ of the IVP defined by (1.1) at point $t_1 = t_0 + h$ by $\tilde{\mathbf{y}}_1$. This is referred to as "taking a step." Similarly, at step $n+1$, the numerical integrator approximates the solution $\tilde{\mathbf{y}}(t_{n+1})$ at point $t_{n+1} = t_n + h$ by $\tilde{\mathbf{y}}_{n+1}$. By taking steps successively, the approximate solutions $\tilde{\mathbf{y}}_2, \ldots, \tilde{\mathbf{y}}_N$ at points $t_2, \ldots, t_N$ are generated. The accuracy of a numerical integrator is primarily determined by the order of the local error it generates in taking a step. The local error at the end of step $n+1$ is the difference $\tilde{\mathbf{u}}(t_{n+1}) - \tilde{\mathbf{y}}_{n+1}$, where $\tilde{\mathbf{u}}(t)$ is the local solution that satisfies the local IVP:

$$\begin{aligned} \frac{d\tilde{\mathbf{u}}(t)}{dt} &= \tilde{\mathbf{f}}\big(t, \tilde{\mathbf{u}}(t)\big), \\ \tilde{\mathbf{u}}(t_n) &= \tilde{\mathbf{y}}_n. \end{aligned} \tag{1.2}$$

Standard Runge-Kutta (RK) methods are a class of *one-step* numerical integrators. That is, the approximate solution $\tilde{\mathbf{y}}_{n+1}$ is calculated using $\tilde{\mathbf{y}}_n$ and the function $\tilde{\mathbf{f}}$. An RK method that has a local error of $O(h^{p+1})$ is said to be of order p and is denoted by RKp. It has been established that RK3, RK4, and RK5 require 3, 4, and 6 function evaluations per time step of integration, respectively [1–3].

In this paper, we propose a new and simple set of numerical integrators named the accelerated-Runge-Kutta (ARK) methods. We derive these methods for the autonomous version of (1.1) given by

$$\begin{aligned} \frac{d\mathbf{y}(t)}{dt} &= \mathbf{f}\big(\mathbf{y}(t)\big), \quad t_0 \le t \le t_N, \\ \mathbf{y}(t_0) &= \mathbf{y}_0, \end{aligned} \tag{1.3}$$

where $\mathbf{y}$, $\mathbf{y}_0$, and $\mathbf{f}$ are m-dimensional vectors, and t denotes the independent variable, time. We assume that $\mathbf{f}$ is a (sufficiently smooth) Lipschitz continuous function, and hence the solution of the IVP given by (1.3) is unique.

Accelerated Runge-Kutta methods are inspired by standard Runge-Kutta methods but are *two-step* in nature. That is, the approximate solution $\mathbf{y}_{n+1}$ is calculated using $\mathbf{y}_n$, $\mathbf{y}_{n-1}$, and the function $\mathbf{f}$. ARKp denotes an ARK method whose local error is of $O(h^{p+1})$. We will see in the next section that ARK3, ARK4, and ARK5 require 2, 3, and 5 function evaluations per time step of integration, respectively. Since function evaluations are often the most computationally expensive part of numerically solving differential equations (see Section 4 for further details), ARK methods are expected to be more computationally efficient than standard RK methods.

Various authors have attempted to increase the efficiency of RK methods. As a result, numerous variations of two-step explicit RK methods exist today. Byrne and Lambert [4] proposed 3rd- and 4th-order two-step Pseudo RK methods. Byrne [5] later proposed a set of methods that minimize a conservative bound on the error, and Gruttke [6] proposed a 5th-order Pseudo RK method. Costabile [7] introduced the Pseudo RK methods of II species which involve many parameters to improve local accuracy. Jackiewicz et al. [8] considered the two-step Runge-Kutta (TSRK) methods up to order 4. Jackiewicz and Tracogna [9] studied a more general class of these TSRK mothods and derived order conditions up to order 5 using Albrecht's method. Recently, Wu [10] has proposed a set of two-step methods which make use of the derivative of the differential equation.

The accelerated-Runge-Kutta (ARK) methods presented here are along the lines first proposed in [11] and are simple and straightforward. They could be considered special cases of the more general TSRK methods. The simplicity of the ARK methods' construction not only reduces computational overhead cost, but also leads to a simpler and more elegant set of order equations that can be directly analyzed more precisely. The ARK methods also benefit from a more precise and effective optimization technique which transforms them into higher-order methods for some problems. In addition, we have presented here a complete analysis of all aspects of the ARK methods, including their motivation, derivation, accuracy, speedup, and stability.

In Section 2, we explain the central idea of ARK methods followed by a description of how to derive and optimize ARK3, ARK4, ARK4-4 (a more accurate ARK4), and ARK5 in Sections 2.1, 2.2, 2.3, and 2.4, respectively. In Section 3, we use seven standard initial value problems to compare the accuracy of ARK and RK methods. In Section 4, we present the computational speedup of ARK methods over the RK methods using some of the same standard problems. The hallmark of our ARK method is its simplicity. Tracogna and Welfert [12], based on methods developed in [9], use a more sophisticated approach using B-series to develop a 2-step TSRK algorithm for numerical integration. In Section 5, we compare the numerical performance of our method with that of [12] using the above-mentioned standard initial value problems. Section 6.1 deals with stability and convergence of ARK methods at small step sizes, and Section 6.2 compares the stability region of ARK and RK methods at finite step sizes. We end with our conclusions and findings, and include in the appendices, the description of the standard initial value problems used, and the Maple code that derives the ARK3 method.

2. Central idea of accelerated Runge-Kutta methods

The idea behind the ARK methods proposed herein is perhaps best conveyed by looking at RK3 and ARK3. An example of the RK3 method solving the IVP given by (1.1) has the form

$$\tilde{\mathbf{y}}_{n+1} = \tilde{\mathbf{y}}_n + \frac{1}{6}\tilde{\mathbf{k}}_1 + \frac{2}{3}\tilde{\mathbf{k}}_2 + \frac{1}{6}\tilde{\mathbf{k}}_3 \quad \text{for } 0 \le n \le N-1, \tag{2.1}$$

where

$$\begin{aligned}
\tilde{\mathbf{k}}_1 &= h\tilde{\mathbf{f}}\left(t_n, \tilde{\mathbf{y}}_n\right),\\
\tilde{\mathbf{k}}_2 &= h\tilde{\mathbf{f}}\left(t_n + \frac{1}{2}h, \tilde{\mathbf{y}}_n + \frac{1}{2}\tilde{\mathbf{k}}_1\right),\\
\tilde{\mathbf{k}}_3 &= h\tilde{\mathbf{f}}\left(t_n + h, \tilde{\mathbf{y}}_n - \tilde{\mathbf{k}}_1 + 2\tilde{\mathbf{k}}_2\right).
\end{aligned} \tag{2.2}$$

The approximate solution $\tilde{\mathbf{y}}_{n+1}$ at t_{n+1} is calculated based on the approximate solution $\tilde{\mathbf{y}}_n$ at t_n along with 3 function evaluations $\tilde{\mathbf{k}}_1$, $\tilde{\mathbf{k}}_2$, and $\tilde{\mathbf{k}}_3$ in $[t_n, t_{n+1}]$. Hence, RK3 is a one-step method with a computational cost of 3 function evaluations per time step.

An example of the *accelerated*-RK3 (ARK3) method solving the IVP given by (1.3) has the form

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \frac{1}{2}\mathbf{k}_1 + \frac{1}{2}\mathbf{k}_{-1} + \left(\mathbf{k}_2 - \mathbf{k}_{-2}\right) \quad \text{for } 1 \le n \le N-1, \tag{2.3}$$

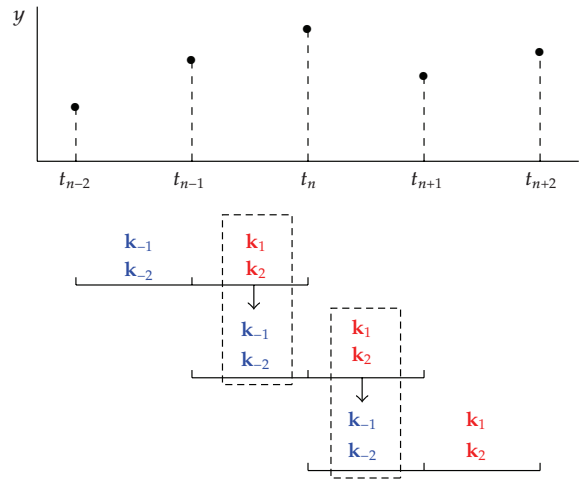


Figure 1: The central idea of the ARK3 method is illustrated here. Approximating the solution $\mathbf{y}_{n+1}$ at t_{n+1} requires 2 function evaluations $\mathbf{k}_1$ and $\mathbf{k}_2$ (shown in red) and two reused values $\mathbf{k}_{-1}$ and $\mathbf{k}_{-2}$ (shown in blue) from the previous steps $\mathbf{k}_1$ and $\mathbf{k}_2$. This process repeats for the next step (approximating the solution at t_{n+2}).

where

$$\begin{aligned} \mathbf{k}_1 &= h\mathbf{f}(\mathbf{y}_n), & \mathbf{k}_{-1} &= h\mathbf{f}(\mathbf{y}_{n-1}), \\ \mathbf{k}_2 &= h\mathbf{f}\left(\mathbf{y}_n + \frac{5}{12}\mathbf{k}_1\right), & \mathbf{k}_{-2} &= h\mathbf{f}\left(\mathbf{y}_{n-1} + \frac{5}{12}\mathbf{k}_{-1}\right). \end{aligned} \tag{2.4}$$

The approximate solution $\mathbf{y}_{n+1}$ at time t_{n+1} is calculated based on the approximate solutions $\mathbf{y}_n$ and $\mathbf{y}_{n-1}$ at times t_n and t_{n-1}, respectively, along with 2 function evaluations $\mathbf{k}_1$ and $\mathbf{k}_2$ in $[t_n, t_{n+1}]$ and 2 reused function evaluations $\mathbf{k}_{-1}$ and $\mathbf{k}_{-2}$ in $[t_{n-1}, t_n]$. In each step of integration from point t_n to point t_{n+1}, only $\mathbf{k}_1$ and $\mathbf{k}_2$ are calculated, while $\mathbf{k}_{-1}$ and $\mathbf{k}_{-2}$ are reused from the previous step's $\mathbf{k}_1$ and $\mathbf{k}_2$. Figure 1 illustrates this idea. The reuse of previously calculated data reduces the number of function evaluations at each time step of integration from 3 for RK3 to 2 for ARK3, making ARK3 more computationally efficient than RK3. This increased efficiency is shared by ARK4 and ARK5 when compared to RK4 and RK5, respectively, and is the central idea behind the ARK methods.

It is important to note that at the first step, there is no previous step. Therefore, ARK methods cannot be self-starting. A one-step method must supply the approximate solution $\mathbf{y}_1$ at the end of the first step t_1. The one-step method must be of sufficient order to ensure that the difference $\mathbf{y}_1 - \mathbf{y}(t_1)$ is of order p or higher. In this case, the ARKp method will be convergent of order p (see Section 6.1, Theorem 6.4). For example, the ARK3 method can be started by the RK3 or RK4 methods. This extra computation occurs only in the first step of integration; for all the subsequent steps, ARK methods bear less computational cost than their RK counterparts.

The general form of the ARK methods presented in this paper is

$$\mathbf{y}_{n+1} = c_0\mathbf{y}_n - c_{-0}\mathbf{y}_{n-1} + c_1\mathbf{k}_1 - c_{-1}\mathbf{k}_{-1} + \sum_{i=2}^{v} c_i(\mathbf{k}_i - \mathbf{k}_{-i}) \quad \text{for } 1 \le n \le N-1, \tag{2.5}$$

where

$$\begin{aligned}
\mathbf{k}_1 &= h\mathbf{f}(\mathbf{y}_n), & \mathbf{k}_{-1} &= h\mathbf{f}(\mathbf{y}_{n-1}),\\
\mathbf{k}_2 &= h\mathbf{f}(\mathbf{y}_n + a_1\mathbf{k}_1), & \mathbf{k}_{-2} &= h\mathbf{f}(\mathbf{y}_{n-1} + a_1\mathbf{k}_{-1}),\\
\mathbf{k}_3 &= h\mathbf{f}(\mathbf{y}_n + a_2\mathbf{k}_2), & \mathbf{k}_{-3} &= h\mathbf{f}(\mathbf{y}_{n-1} + a_2\mathbf{k}_{-2}),\\
\mathbf{k}_4 &= h\mathbf{f}(\mathbf{y}_n + a_3\mathbf{k}_3), & \mathbf{k}_{-4} &= h\mathbf{f}(\mathbf{y}_{n-1} + a_3\mathbf{k}_{-3}),\\
\mathbf{k}_5 &= h\mathbf{f}(\mathbf{y}_n + a_4\mathbf{k}_4), & \mathbf{k}_{-5} &= h\mathbf{f}(\mathbf{y}_{n-1} + a_4\mathbf{k}_{-4}).
\end{aligned} \tag{2.6}$$

Here, v denotes the number of function evaluations performed at each time step and increases with the order of local accuracy of the ARK method. In each step of integration, only $\mathbf{k}_1, \mathbf{k}_2, \ldots$ are evaluated, while $\mathbf{k}_{-1}, \mathbf{k}_{-2}, \ldots$ are reused from the previous step.

To determine the appropriate values of the parameters c's and a's, a three-part process is performed. (The first and second parts of this process are shown in detail for ARK3 in [11].) First, the ARK expression (2.5) is expanded using the Taylor series expansion. Second, after some algebraic manipulation, this expansion is equated to the local solution $\mathbf{u}(t_{n+1})$ at t_{n+1} given by the Taylor series expansion (see (1.2))

$$\begin{aligned}
\mathbf{u}(t_{n+1}) = \mathbf{u}(t_n + h) &= \mathbf{u}(t_n) + h\mathbf{u}'(t_n) + \frac{h^2}{2!}\mathbf{u}''(t_n) + \frac{h^3}{3!}\mathbf{u}'''(t_n) + \cdots\\
&= \mathbf{y}_n + h\mathbf{y}'_n + \frac{h^2}{2!}\mathbf{y}''_n + \frac{h^3}{3!}\mathbf{y}'''_n + \cdots .
\end{aligned} \tag{2.7}$$

This results in the system of nonlinear algebraic *order equations* (2.8)–(2.13). Third, we attempt to solve as many order equations as possible because the highest power of h for which all of the order equations are satisfied is the order of the resulting ARK method. The above process requires a great deal of algebraic and numeric calculations which were mainly performed using Maple (see Appendix B).

In the following, we will show what order of accuracy an ARK method can achieve with a given number of function evaluations v. For $v = 1$, a second-order method ARK2 is possible. It is, however, omitted here because such a low order of accuracy is of limited use in accurately solving differential equations; we will look at cases $v = 2, \ldots, 5$.

2.1. ARK3

For $v = 2$, from (2.5), we have the 6 parameters $c_0,\ c_{-0},\ c_1,\ c_{-1},\ c_2$, and a_1. The derived order equations up to $O(h^4)$ are

$$O(h^0) \quad c_0 - c_{-0} = 1, \tag{2.8a}$$

$$O(h^1) \quad c_{-0} + c_1 - c_{-1} = 1, \tag{2.8b}$$

$$O(h^2) \quad -\frac{1}{2}c_{-0} + c_{-1} + c_2 = \frac{1}{2}, \tag{2.8c}$$

$$O(h^3) \quad -\frac{1}{12}c_{-0} + c_2 a_1 = \frac{5}{12}, \tag{2.8d}$$

$$O(h^4) \quad c_2 a_1^2 = \frac{1}{3}, \tag{2.8e}$$

$$O(h^4) \quad 0 = \frac{1}{6}. \tag{2.8f}$$

Clearly, (2.8f) cannot be satisfied, so it is impossible to achieve a 4th-order method with 2 function evaluations per time step. It is, however, possible to achieve a 3rd-order method ARK3 with 2 function evaluations per time step by satisfying (2.8a)–(2.8d). We solve these equations in terms of c_0, c_{-0}, c_1, and c_{-1} and let the remaining two parameters c_2 and a_1 become free parameters.

We use sets of free parameters to optimize the accuracy of an ARK method by minimizing its local error. The local error is represented by higher order equations (order 4 and above in case of ARK3). The closer these order equations are to being satisfied, the smaller the local error is. Therefore, we try to minimize the difference between the LHS and RHS of these order equations. This can be done in several different ways. One popular technique is to minimize a general bound on the error [5]. A second technique is to minimize a norm of the error [13]. A third technique is to satisfy as many individual order equations as possible (hinted to by [4]). In general, there is no way of knowing a priori which technique yields the most accurate ARK method since the error ultimately depends on the problem at hand. We have chosen the last technique for two reasons. One, it is straightforward to implement exactly. Two, for some classes of problems, the differential term multiplying the unsolved higher order equations may vanish, which could effectively increases the order of the method. For instance, the differential term multiplying (2.8f) may vanish, and provided (2.8e) is already satisfied, the resulting ARK method (with 2 function evaluations) will become a 4th-order method.

Using the two free parameters c_2 and a_1, we can solve two higher-order equations: (2.8e) and any one of the two 5th-order equations,

$$O(h^5) \quad \frac{1}{120}c_{-0} + c_2 a_1^3 = \frac{31}{120}, \tag{2.9a}$$

$$O(h^5) \quad c_{-0} = 31. \tag{2.9b}$$

However, we cannot choose (2.9b) because to prove stability, we will need $-1 \leq c_{-0} < 1$ (see condition (6.16)). Therefore, we choose (2.8e) and (2.9a), which leads to Set 2 of Table 1. We also present a set in which $c_{-0} = 0$ because doing so can increase the stability (see Section 6) and speed (see Section 4) of an ARK method. Setting $c_{-0} = 0$ and solving (2.8a)–(2.8e) leads to Set 3 of Table 1. Set 1 of Table 1 is the only set of parameters here that is not found by solving all possible higher order equations. It is found by setting $c_{-0} = 0$, solving (2.8a)–(2.8d) for c_1, c_{-1}, and c_2, and searching for a value of a_1 that leads to a highly accurate ARK method in solving some standard IVPs (see Section 3 and Appendix A).

2.2. ARK4

For $v = 3$, we have the 8 parameters c_0, c_{-0}, c_1, c_{-1}, c_2, c_3, a_1, and a_2 along with order equations up to $O(h^5)$ which are

$$O(h^0) \quad c_0 - c_{-0} = 1, \tag{2.10a}$$

$$O(h^1) \quad c_{-0} + c_1 - c_{-1} = 1, \tag{2.10b}$$

$$O(h^2) \quad -\frac{1}{2}c_{-0} + c_{-1} + c_2 + c_3 = \frac{1}{2}, \tag{2.10c}$$

$$O(h^3) \quad -\frac{1}{12}c_{-0} + c_2 a_1 + c_3 a_2 = \frac{5}{12}, \tag{2.10d}$$

Table 1: Optimized ARK3 parameter sets.

	Set 1	Set 2	Set 3
c_0	1	$-4\dfrac{\sqrt{41}-11}{9+\sqrt{41}}$	1
c_{-0}	0	$-5\dfrac{\sqrt{41}-7}{9+\sqrt{41}}$	0
c_1	$\dfrac{1}{2}$	$\dfrac{16(6\sqrt{41}-1)}{3(9+\sqrt{41})^2}$	$\dfrac{47}{48}$
c_{-1}	$-\dfrac{1}{2}$	$\dfrac{4(3\sqrt{41}-13)}{3(9+\sqrt{41})^2}$	$-\dfrac{1}{48}$
c_2	1	$\dfrac{400}{3(9+\sqrt{41})^2}$	$\dfrac{25}{48}$
a_1	$\dfrac{5}{12}$	$\dfrac{9+\sqrt{41}}{20}$	$\dfrac{4}{5}$

$$O(h^4) \quad c_2a_1^2 + c_3a_2^2 = \frac{1}{3}, \tag{2.10e}$$

$$O(h^4) \quad c_3a_1a_2 = \frac{1}{6}, \tag{2.10f}$$

$$O(h^5) \quad \frac{1}{120}c_{-0} + c_2a_1^3 + c_3a_2^3 = \frac{31}{120}, \tag{2.10g}$$

$$O(h^5) \quad \frac{1}{240}c_{-0} + c_3a_1a_2^2 = \frac{31}{240}, \tag{2.10h}$$

$$O(h^5) \quad \frac{1}{360}c_{-0} + c_3a_1^2a_2 = \frac{31}{360}, \tag{2.10i}$$

$$O(h^5) \quad c_{-0} = 31. \tag{2.10j}$$

Since there are 10 order equations up to $O(h^5)$, we cannot achieve a 5th-order method. However, we can achieve a 4th-order method ARK4 by solving the first 6 order equations (2.10a)–(2.10f). Therefore, ARK4 requires 3 function evaluations per time step. We have chosen to solve these equations for the c's while keeping a_1 and a_2 as free parameters because all order equations are linear in terms of the c's which makes it straightforward to solve for them.

We use a_1 and a_2 to optimize ARK4's accuracy by solving two of the $O(h^5)$ equations. We choose (2.10g) and (2.10h) because they were used in deriving the other $O(h^5)$ equations. Set 2 and Set 3 of Table 2 are the two solutions. Set 1 of Table 2 is a set in which $c_{-0} = 0$ and is produced by solving (2.10b) to (2.10g). It is a set of approximate values and naturally contains some round-off error. The effects of this round-off error (in parameter values) can be significant if the error produced by the ARK methods is small enough to be of the same order. Therefore, to measure the accuracy of the ARK methods correctly (see Section 3), we have calculated the parameter values using 25 digits of computational precision in Maple. Note, however, that such high precision is not required for the ARK methods to be effective.

Table 2: Optimized ARK4 parameter sets.

	Set 1	Set 2	Set 3
c_0	1.0	$-4\dfrac{\sqrt{41}-11}{9+\sqrt{41}}$	$-4\dfrac{\sqrt{41}-11}{9+\sqrt{41}}$
c_{-0}	0.0	$-5\dfrac{\sqrt{41}-7}{9+\sqrt{41}}$	$-5\dfrac{\sqrt{41}-7}{9+\sqrt{41}}$
c_1	1.01762767320449524674963508	$\dfrac{16}{3}\dfrac{6\sqrt{41}-1}{(9+\sqrt{41})^2}$	$\dfrac{16}{3}\dfrac{6\sqrt{41}-1}{(9+\sqrt{41})^2}$
c_{-1}	0.01762767320449524674963508	$\dfrac{4}{3}\dfrac{3\sqrt{41}-13}{(9+\sqrt{41})^2}$	$\dfrac{4}{3}\dfrac{3\sqrt{41}-13}{(9+\sqrt{41})^2}$
c_2	$-$ 0.13300377780975252807712 93	0	$\dfrac{200}{3(9+\sqrt{41})^2}$
c_3	0.61537610460525728132749 42	$\dfrac{400}{3(9+\sqrt{41})^2}$	$\dfrac{200}{3(9+\sqrt{41})^2}$
a_1	0.35888611391988193765959 42	$\dfrac{9+\sqrt{41}}{40}$	$\dfrac{9+\sqrt{41}}{20}$
a_2	0.75466023484835962323552 57	$\dfrac{9+\sqrt{41}}{20}$	$\dfrac{9+\sqrt{41}}{20}$

2.3. ARK4-4

For $v = 4$, we have the 10 parameters c_0, c_{-0}, c_1, c_{-1}, c_2, c_3, c_4, a_1, a_2, and a_3. Order equations up to $O(h^5)$ are

$$O(h^0) \quad c_0 - c_{-0} = 1, \tag{2.11a}$$

$$O(h^1) \quad c_{-0} + c_1 - c_{-1} = 1, \tag{2.11b}$$

$$O(h^2) \quad -\frac{1}{2}c_{-0} + c_{-1} + c_2 + c_3 + c_4 = \frac{1}{2}, \tag{2.11c}$$

$$O(h^3) \quad -\frac{1}{12}c_{-0} + c_2a_1 + c_3a_2 + c_4a_3 = \frac{5}{12}, \tag{2.11d}$$

$$O(h^4) \quad c_2a_1^2 + c_3a_2^2 + c_4a_3^2 = \frac{1}{3}, \tag{2.11e}$$

$$O(h^4) \quad c_3a_1a_2 + c_4a_2a_3 = \frac{1}{6}, \tag{2.11f}$$

$$O(h^5) \quad \frac{1}{120}c_{-0} + c_2a_1^3 + c_3a_2^3 + c_4a_3^3 = \frac{31}{120}, \tag{2.11g}$$

$$O(h^5) \quad \frac{1}{240}c_{-0} + c_3a_1a_2^2 + c_4a_2a_3^2 = \frac{31}{240}, \tag{2.11h}$$

$$O(h^5) \quad \frac{1}{360}c_{-0} + c_3a_1^2a_2 + c_4a_2^2a_3 = \frac{31}{360}, \tag{2.11i}$$

$$O(h^5) \quad \frac{1}{720}c_{-0} + c_4a_1a_2a_3 = \frac{31}{720}. \tag{2.11j}$$

Table 3: Optimized ARK4-4 parameter sets.

	Set 1	Set 2	Set 3
c_0	1.0	1.0	1.0
c_{-0}	0.0	0.0	0.0
c_1	1.022831928839203211581411	0.959998362974052335776129 2	1.038087495003156301209584
c_{-1}	0.022831928839203211581410 16	−0.040001637025947664223868 92	0.038087495003156301209585 82
c_2	−0.045158301883180231641969 73	0.248334450574304939296430 5	−0.120695229675287590559474 7
c_3	−0.086187006135813174734622 00	−0.440029058805122729929279 1	0.430768853504061439164019 7
c_4	0.608513379179790194795185 5	0.731696245256765454856715 2	0.151838881168069850185868 1
a_1	0.246418984804535202766398 8	0.212807618423144803700727 5	0.234055561829377338659576 6
a_2	0.379427607085112010701626 9	0.380758689679147939139774 1	0.753248901556639066614579 1
a_3	0.756756177970740702853666 9	0.726208580354885731734735 2	0.793208497093576157136026 7

Since there are exactly 10 order equations up to $O(h^5)$, it would seem that we can achieve a 5th-order method. However, the only two solutions of the above order equations have $c_{-0} = 1$ and $c_{-0} = 25$, both of which produce unstable methods (see condition (6.16)).

Therefore, we set $c_{-0} = 0$ which leaves us with the 8 parameters c_1, c_{-1}, c_2, c_3, c_4, a_1, a_2, and a_3. We solve (2.11b)–(2.11f) for the c's and solve three of the four $O(h^5)$ order equations using the free parameters a_1, a_2, and a_3. The result is a 4th-order method called ARK4-4 that is more accurate than ARK4 for most of the IVPs that are investigated herein. The "-4" indicates that this 4th-order method requires 4 function evaluations, unlike ARK4 which requires only 3.

Set 1 of Table 3 satisfies the higher order equations (2.11g), (2.11h), and (2.11j). Set 3 of Table 3 satisfies the higher order equations (2.11g), (2.11h), and (2.11i). Set 2 of Table 3 satisfies the higher order equations (2.11g), (2.11j) and (2.12b). To understand where (2.12b) comes from, we first need to explain a unique property of the $O(h^5)$ order equations.

The $O(h^5)$ order equations are fundamentally different from lower order equations. If the order equations for ARK4-4 are derived with a two-dimensional coupled ODE system, we get the four $O(h^5)$ order equations presented above, that is, (2.11g)–(2.11j). However, if the derivation is done with a one-dimensional ODE or with a two-dimensional uncoupled ODE system, we get the three $O(h^5)$ equations:

$$O(h^5) \quad \frac{1}{120}c_{-0} + c_2a_1^3 + c_3a_2^3 + c_4a_3^3 = \frac{31}{120}, \tag{2.12a}$$

$$O(h^5) \quad \frac{1}{90}c_{-0} + c_3a_1a_2(a_1 + 2a_2) + c_4a_2a_3(a_2 + 2a_3) = \frac{31}{90}, \tag{2.12b}$$

$$O(h^5) \quad \frac{1}{720}c_{-0} + c_4a_1a_2a_3 = \frac{31}{720}. \tag{2.12c}$$

Hence, coupling of the ODE system affects the $O(h^5)$ order equations but not the $O(h^0), \ldots, O(h^4)$ order equations. We see that the difference between the two sets of order equations is that (2.11h) and (2.11i) are replaced by (2.12b) = 2 × (2.11h) + (2.11i). Therefore, a parameter set that satisfies (2.11h) and (2.11i) also automatically satisfies (2.12b), but not vice versa. To ensure that the order equations are valid for all ODE systems, we have considered a two-dimensional coupled ODE system in our derivations.

Although we have not derived the order equations based on a 3- or higher-dimensional coupled ODE system, we hypothesize that the order equations would be the same as

(2.11g)–(2.11j). We base this hypothesis on two observations. One, unlike moving from a one-dimensional system to a two-dimensional system where the coupling property emerges, no property emerges when we move from a two-dimensional system to a three-dimensional system, so there is no perceivable reason for the order equations to change. Two, using a two-dimensional coupled ODE system, we have derived the same standard RK order equations that others have reported [13].

With that explanation, it might seem counterintuitive to use a parameter set that satisfies (2.12b). After all, this equation only applies for uncoupled systems. However, the thinking is that for uncoupled systems, the resulting ARK4-4 becomes a 5th-order method. While for coupled systems, the resulting method yields a very accurate 4th-order method because it satisfies two of the four $O(h^5)$ equations as well as a weighted average of the other two order equations. This is confirmed by our experiments in solving standard problems (see Section 3) since the accuracy of Set 2 is comparable to that of Set 1 and Set 3.

2.4. ARK5

For $v = 5$, we have the 12 parameters c_0, c_{-0}, c_1, c_{-1}, c_2, c_3, c_4, c_5, a_1, a_2, a_3, and a_4. Order equations up to $O(h^6)$ are

$$O(h^0) \quad c_0 - c_{-0} = 1, \tag{2.13a}$$

$$O(h^1) \quad c_{-0} + c_1 - c_{-1} = 1, \tag{2.13b}$$

$$O(h^2) \quad -\frac{1}{2}c_{-0} + c_{-1} + c_2 + c_3 + c_4 + c_5 = \frac{1}{2}, \tag{2.13c}$$

$$O(h^3) \quad -\frac{1}{12}c_{-0} + c_2a_1 + c_3a_2 + c_4a_3 + c_5a_4 = \frac{5}{12}, \tag{2.13d}$$

$$O(h^4) \quad c_2a_1^2 + c_3a_2^2 + c_4a_3^2 + c_5a_4^2 = \frac{1}{3}, \tag{2.13e}$$

$$O(h^4) \quad c_3a_1a_2 + c_4a_2a_3 + c_5a_3a_4 = \frac{1}{6}, \tag{2.13f}$$

$$O(h^5) \quad \frac{1}{120}c_{-0} + c_2a_1^3 + c_3a_2^3 + c_4a_3^3 + c_5a_4^3 = \frac{31}{120}, \tag{2.13g}$$

$$O(h^5) \quad \frac{1}{240}c_{-0} + c_3a_1a_2^2 + c_4a_2a_3^2 + c_5a_3a_4^2 = \frac{31}{240}, \tag{2.13h}$$

$$O(h^5) \quad \frac{1}{360}c_{-0} + c_3a_1^2a_2 + c_4a_2^2a_3 + c_5a_3^2a_4 = \frac{31}{360}, \tag{2.13i}$$

$$O(h^5) \quad \frac{1}{720}c_{-0} + c_4a_1a_2a_3 + c_5a_2a_3a_4 = \frac{31}{720}, \tag{2.13j}$$

$$O(h^6) \quad c_2a_1^4 + c_3a_2^4 + c_4a_3^4 + c_5a_4^4 = \frac{1}{5}, \tag{2.13k}$$

$$O(h^6) \quad c_3a_1a_2^3 + c_4a_2a_3^3 + c_5a_3a_4^3 = \frac{1}{10}, \tag{2.13l}$$

$$O(h^6) \quad c_3a_1^3a_2 + c_4a_2^3a_3 + c_5a_3^3a_4 = \frac{1}{20}, \tag{2.13m}$$

Table 4: Optimized ARK5 parameter sets.

	Set 1	Set 2	Set 3
c_0	1.0	1.0	1.87120458717158206517414 0
c_{-0}	0.0	0.0	0.87120458717158206517130 61
c_1	1.05556215137169893658899 6	0.84781861161579177688825 25	0.26964668866638216371280 20
c_{-1}	0.05556215137169893658900796	−0.15218138838420822311175 44	0.14085127583796422888743 80
c_2	−0.15507826549018113423494 42	0.63424822240505828729250 60	0.31587594655569976308087 50
c_3	0.42592470856062909111684 54	0.05195876382507141388229 794	0.32128307480494078660187 70
c_4	0.11030093105835812699349 50	−0.25919009955146520907640 61	0.15910610353930500045737 04
c_5	0.06329047449949497953556 305	0.22516450170554373101332 41	−0.00151410715211874643783 8297
a_1	0.21634433210095616972608 89	0.97101495143869389525856 86	0.50945869456439586647988 05
a_2	0.73554210891429434998013 71	−0.25561031463318690045865 66	0.51615884010011715740278 62
a_3	0.70463958528507163869393 35	1.09459954227069249019510 2	1.04169556610008939862512 0
a_4	0.93551217959468840143281 40	0.43431677438762241454203 28	2.13453867683349264069529 4

$$O(h^6) \quad c_3a_1^2a_2^2 + c_4a_2^2a_3^2 + c_5a_3^2a_4^2 = \frac{1}{15}, \tag{2.13n}$$

$$O(h^6) \quad c_4a_1a_2^2a_3 + c_5a_2a_3^2a_4 = \frac{1}{40}, \tag{2.13o}$$

$$O(h^6) \quad c_4a_1a_2a_3^2 + c_5a_2a_3a_4^2 = \frac{1}{60}, \tag{2.13p}$$

$$O(h^6) \quad c_4a_1^2a_2a_3 + c_5a_2^2a_3a_4 = \frac{1}{30}, \tag{2.13q}$$

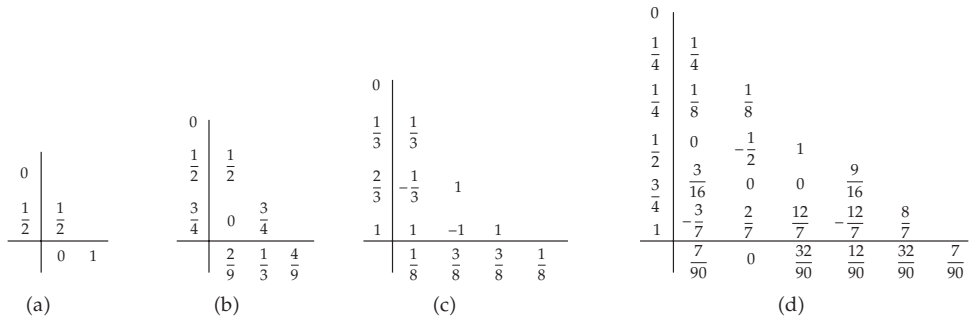
$$O(h^6) \quad c_5a_1a_2a_3a_4 = \frac{1}{120}. \tag{2.13r}$$

Solving the first 10 equations, (2.13a)–(2.13j), gives us a 5th-order method ARK5 which requires 5 function evaluations per time step. We solve the first 8 equations for the c's and solve equations (2.13i)–(2.13l) for $a_1, \ldots, a_4$. Among the multiple solutions of this system, we have found Set 3 in Table 4 to be the most accurate in solving the standard problems given in Appendix A. Setting $c_{-0} = 0$ results in 10 parameters and 9 equations of order up to $O(h^5)$. We use the one free parameter to solve (2.13k) which leads to numerous solutions. Among them, Set 1 and Set 2 in Table 4 are the most accurate in solving the standard initial value problems in Section 3.

3. Accuracy

We consider seven initial value problems (IVPs) to illustrate the accuracy of the ARK methods. These problems are taken from examples used by previous researchers [11, 14]. They are named IVP-1 through IVP-7, and are listed in Appendix A. Most of the problems model real-world phenomena (suggested by [14]) and have been used by other researchers to test numerical integrators. Specifically, IVP-3 models the Euler equations for a rigid body without external forces, IVP-4 and IVP-5 model a one-body gravitational problem in the plane, IVP-6 models a radioactive decay chain, and IVP-7 models the gravitational problem for five planets around the sun (and four inner planets). The first two example problems (IVP-1 and IVP-2) are

Table 5: RK parameters in Butcher's notation [15]: (a) RK2, (b) RK3, (c) RK4, and (d) RK5.



(a) (b) (c) (d)

one-dimensional ODEs—the first, autonomous, the second, nonautonomous. The dimension of the rest of the examples, which include linear and non-linear ODE systems, ranges from 3 to 30.

The approximate solution to each example ODE problem is computed for $0 \leq t \leq 15$ at the step sizes: 0.001, 0.0025, 0.005, 0.01, 0.025, 0.05, and 0.1 using ARK3, ARK4, ARK4-4, and ARK5. For comparison, we also show the corresponding results using the RK2, RK3, RK4, and RK5 methods. Each ARK method is started with an RK method of the same order, taking 10 steps in the first time step at 1/10th the value of step size. For ARK methods, Set 1 parameter values of Tables 1, 2, 3, and 4 are used, and for RK methods, the parameter values in Table 5 are used.

The accuracy plots for problems IVP-1 through IVP-7 are given in Figures 2, 3, 4, 5, 6, 7, and 8, respectively. Along the ordinate is plotted an average of the 2-norm of the global error, which is computed by taking the 2-norm of the difference between the solution $\mathbf{y}(t_n)$ and approximate solution $\mathbf{y}_n$, and then averaging this norm over the interval $10 \leq t \leq 15$. The exception is IVP-5 for which the average absolute value of global error in all four components of the solution are plotted separately. The results obtained when using RK methods are shown by lines with circles along them, while the results for ARK methods are shown by lines with squares along them. Lines with the same color indicate methods that have the same number of function evaluations.

Due to small step sizes, for example 10^{-3}, and the high order of local accuracy for some of the RK and ARK methods, the global error for some problems reaches 10^{-24}. This is beyond MATLAB's precision capabilities. MATLAB has a minimum round-off error of about `eps` $\approx 2.3\times10^{-16}$ (at magnitude 1) and a maximum of 17 significant digits of accuracy. Therefore, a proper analysis of the accuracy of the ARK and RK methods cannot be done in MATLAB. In fact, we found that for small step sizes, the higher order ARK and RK plot lines produced by MATLAB are erroneous; they are jagged lines with slopes different than the order of the method. That is why the PARI mathematics environment [16] which provides 28 digits of accuracy by default was used for all calculations in this section. This ensures that the computational round-off error is far smaller than the methods' global error. Note however, that using high precision computation is not necessary for the ARK methods to be effective. The ARK methods should retain their accuracy and efficiency advantages over the RK methods at any precision.

The slopes of the plot lines confirm the order of our ARK methods; the slopes of ARK3, ARK4, ARK4-4, and ARK5 lines (except when the time steps are large) are 3, 4, 4, and 5, respectively. The slope of the ARK5 lines for problems IVP-3 to IVP-7 (Figures 4 to 8) is

Table 6: Summary of order and number of function evaluations per time step for RK methods and ARK methods presented herein.

	RK2	ARK3	RK3	ARK4	RK4	ARK4-4	RK5	ARK5
Order of method	$O(h^2)$	$O(h^3)$	$O(h^3)$	$O(h^4)$	$O(h^4)$	$O(h^4)$	$O(h^5)$	$O(h^5)$
No. of function evaluations	2	2	3	3	4	4	6	5

especially important. We hypothesized in Section 2.3 that although our $O(h^5)$ order equations were derived based on a two-dimensional ODE system, they would also apply to higher-dimensional problems. The fact that problems IVP-3 to IVP-7 are 3-dimensional or higher and their ARK5 accuracy plot lines have slope 5 (for small step sizes) confirms our hypothesis.

Comparing ARK and RK methods, the plots show that generally speaking, the global error of the ARK methods is greater than that of the corresponding RK methods by about one order of magnitude when comparing methods of the same order, such as comparing ARK3 versus RK3. Although great effort was made to find the parameter sets that yield the most accurate ARK methods, it is possible that better sets can be found which would considerably improve the accuracy of the ARK methods. Set 1 of ARK4 exemplifies this claim. It is seen from the accuracy plots that for problems IVP-2, IVP-3, IVP-5, and IVP-7, ARK4 has virtually the same accuracy as RK4.

A more useful comparison of ARK and RK methods would be to compare those that have the same number of function evaluations per time step. Table 6 summarizes the order and number of function evaluations for the ARK and RK methods. In comparing ARK3 versus RK2, ARK4 versus RK3, and ARK4-4 versus RK4 in the accuracy plots, ARK methods prove to be more accurate. Because ARK3 and ARK4 are higher-order methods than RK2 and RK3, respectively, their gain in accuracy grows as the step size decreases. For example, for IVP-5 (see Figure 6) at step size 0.1, ARK3 and ARK4 are less than one order of magnitude more accurate than RK2 and RK3, respectively, but at step size 0.001, they are three and four orders of magnitude more accurate than RK2 and RK3, respectively. A similar gain of accuracy can occur even for ARK4-4 versus RK4 when for some problems (see Figures 2 and 7), ARK4-4 becomes a 5th-order method.

4. Speedup

As mentioned before, the ARK3, ARK4, and ARK5 methods require 2, 3, and 5 function evaluations per time step of integration, respectively, while the RK3, RK4, and RK5 methods require 3, 4, and 6 function evaluations per time step, respectively. It is commonly believed that function evaluations constitute the main computational cost of RK-type methods. Hence, we can expect that compared to the RK3, RK4, and RK5 methods, the ARK3, ARK4, and ARK5 methods have speedups of approximately $(3-2)/3 \approx 33\%$, $(4-3)/4 = 25\%$, and $(6-5)/6 \approx 17\%$, respectively.

To verify these speedups experimentally, we have implemented the methods in MATLAB. We have measured the time it takes to solve IVP-2 and IVP-4 for $0 \leq t \leq 1000$ (with such a large time span needed to get a reliable time measurement) at the 7 different step sizes used in Section 3. (We have used the MATLAB utility `TIC` and `TOC` for time measurements). We repeated this integration 100 times and took the average of integration times before calculating the speedup $(t_{\mathrm{RK}} - t_{\mathrm{ARK}})/t_{\mathrm{RK}}$. Figures 9 and 10 display the results. Speedup depends on the

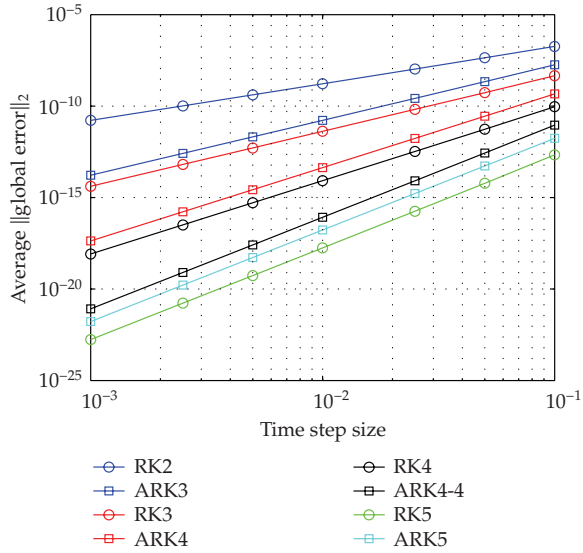


Figure 2: Accuracy plot for IVP-1 using log-log scale. The RK methods are shown by lines with circles. The ARK methods are shown by lines with squares. Lines of the same color correspond to methods with the same number of function evaluations per time step.

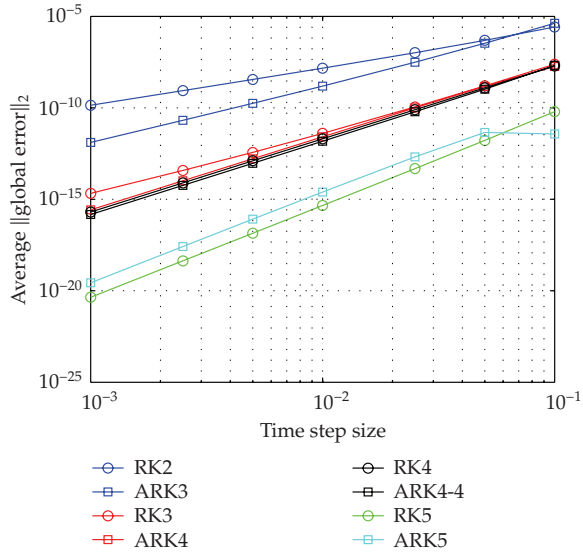


Figure 3: Accuracy plot for IVP-2 (see caption of Figure 2).

problem being solved, and among IVP-1 through IVP-7, IVP-2 leads to the lowest speedup, while IVP-1 leads to the highest. Speedup also depends on the parameter set used because it affects the computational overhead. The ARK implementations here use values of Set 1 in Tables 1, 2, and 4, and the RK implementations use parameter values in Table 5. The plots show a linear least square fit to the points. We expect this line to be fairly flat because the number

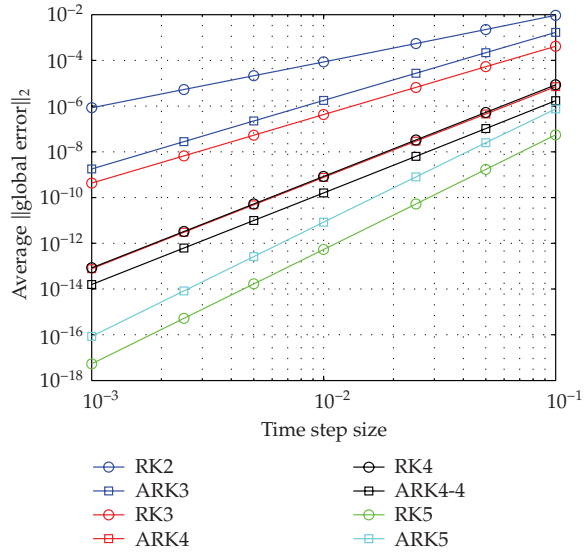


Figure 4: Accuracy plot for IVP-3 (see caption of Figure 2).

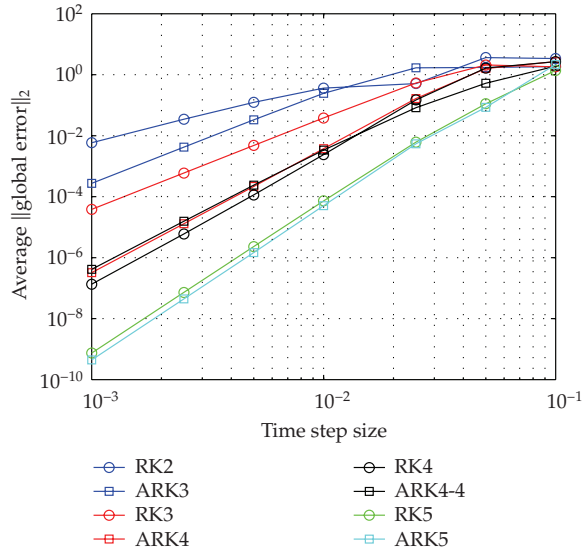


Figure 5: Accuracy plot for IVP-4 (see caption of Figure 2).

of steps for the ARK and RK methods are only slightly different (caused by the startup of the ARK methods at the first time step) and because speedup does not depend on the step size.

The speedups seen in Figures 9 and 10 are lower than expected. Moreover, we have found the time measurement and consequently the speedup to be machine dependent. Through measurements of the computation times at different points in our computer codes, we have found two main causes for the less-than-expected speedup of our ARK methods. First, the ARK methods require additional multiplication and addition operations when evaluating the

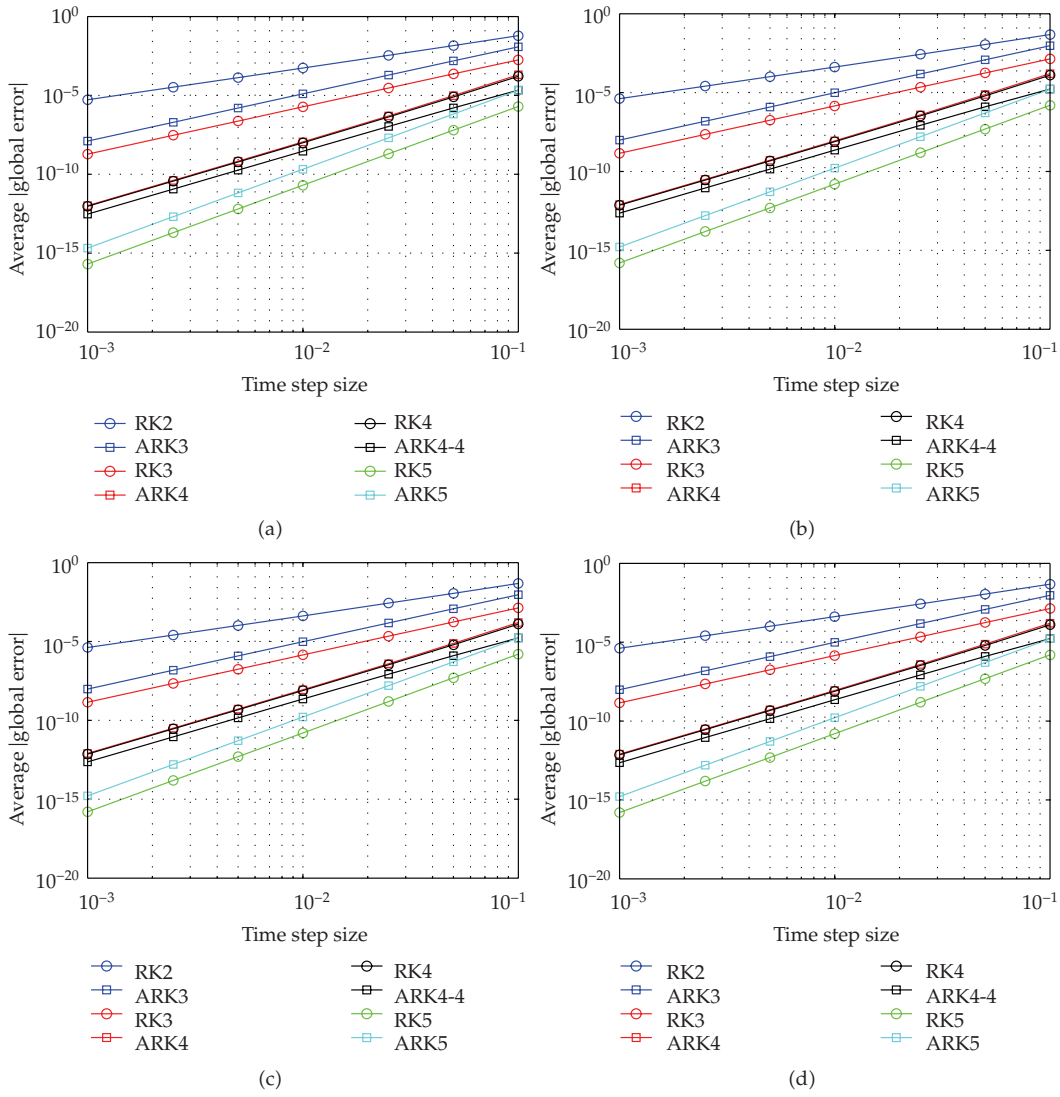


Figure 6: Accuracy plots for IVP-5: (a) y_1, (b) y_2, (c) y_3, and (d) y_4 (see caption of Figure 2).

expression for $\mathbf{y}_{n+1}$ than the RK methods require. This accounts for about half of the loss in the expected speedup. Second, the ARK methods require a number of assignment operations to set $\mathbf{y}_{n-1}, \mathbf{k}_{-1}, \ldots$ to $\mathbf{y}_n, \mathbf{k}_1, \ldots$ in every step of integration which the RK methods do not require. Somewhat surprisingly, this takes a significant amount of computation time in the MATLAB environment and is responsible for about the other half of the loss in the expected speedup.

5. Comparison with other two-step methods

We now compare the ARK4 method (with parameters of Set 1 of Table 2) to the more sophisticated TSRK4 method presented in [12]. Both are fourth-order methods with 3 function

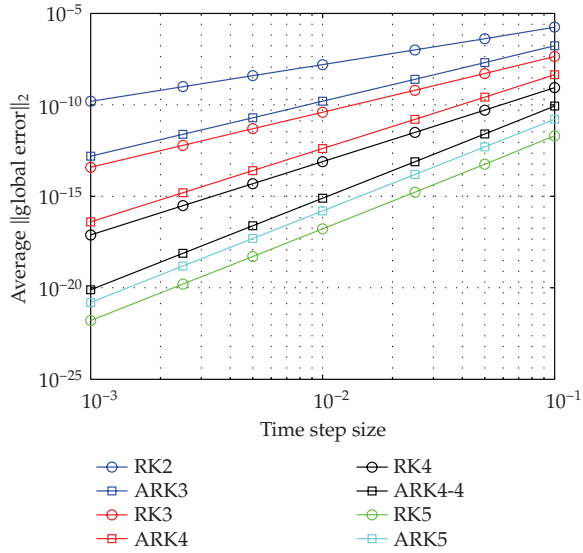


Figure 7: Accuracy plot for IVP-6 (see caption of Figure 2).

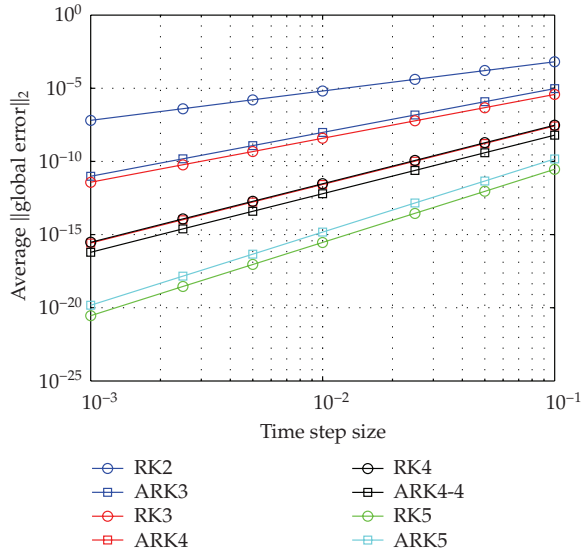


Figure 8: Accuracy plot for IVP-7 (see caption of Figure 2).

evaluations per time step. The accuracy of the two methods is calculated in the same way as in Section 3 but implemented in MATLAB. Because of MATLAB's lower precision, we have not calculated the accuracy at the two smallest step sizes. The average of the 2-norm of the global error for problems IVP-1, IVP-2, and IVP-4 is presented in Figures 11, 12, and 13. The TSRK4 method is more than one order of magnitude more accurate in solving IVP-1; the ARK4 method is less than one order of magnitude more accurate than the TSRK4 method in solving

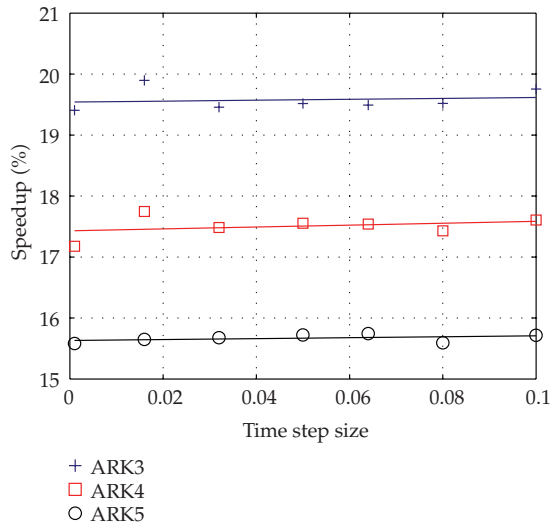


Figure 9: Speedup of ARK methods compared with RK methods for IVP-2.

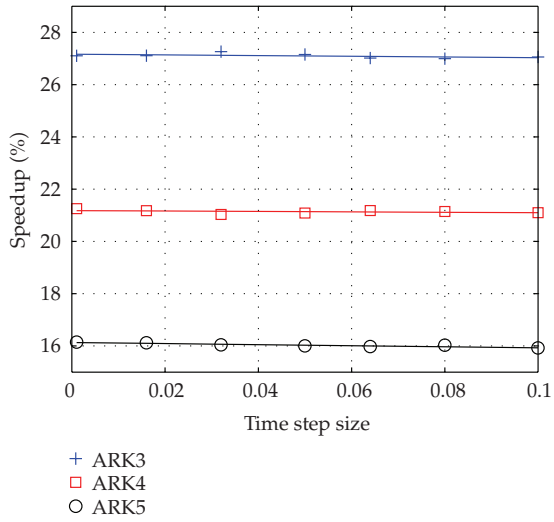


Figure 10: Speedup of ARK methods compared with RK methods for IVP-4.

IVP-2; and the TSRK4 method is slightly more accurate than ARK4 in solving IVP-4. The TSRK4 method is also more accurate in solving the other problems listed in Appendix A. This is because the TSRK4 method is more general and possesses more parameters, which have been effectively optimized. In Figure 14, we have plotted the relative change in computation time when solving IVP-1, IVP-2, and IVP-4 (similar to Section 4). The lower computation time (see Figure 14) required by the ARK4 method is due to its lower computational overhead, which is produced by its simpler form as given in (2.5).

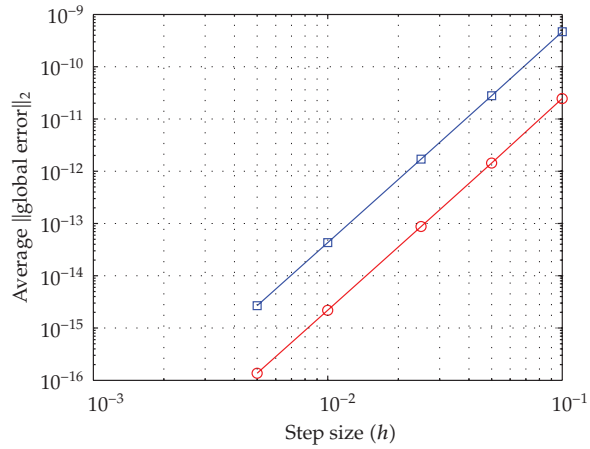


Figure 11: Comparison of the accuracy of the ARK4 (blue) and TSRK4 (red) methods in solving IVP-1.

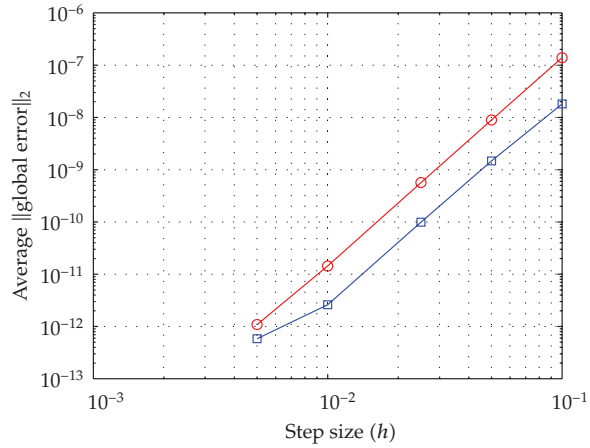


Figure 12: Comparison of the accuracy of the ARK4 (blue) and TSRK4 (red) methods in solving IVP-2.

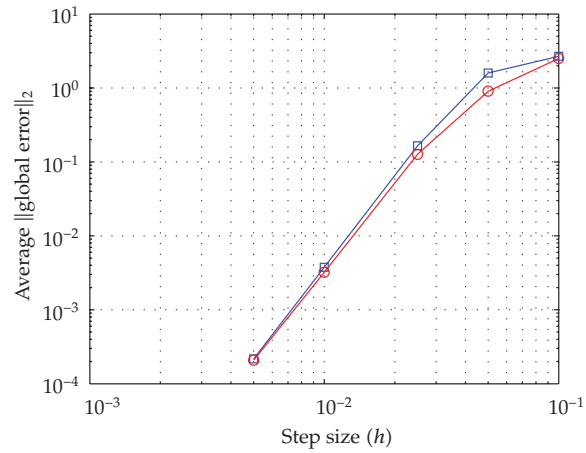


Figure 13: Comparison of the accuracy of the ARK4 (blue) and TSRK4 (red) methods in solving IVP-4.

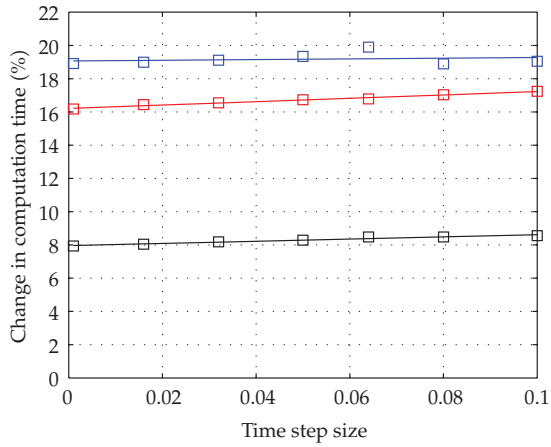


Figure 14: Relative change in computation time of the TSRK4 method compared with the ARK4 method in solving IVP-1 (blue), IVP-2 (red), and IVP-4 (black), showing that TSRK4 requires greater computation time.

6. Stability

We have already established that the error of an ARKp method over a single time step (local error) has the order $O(h^{p+1})$ (see Section 1). In this section, we will look at how this error accumulates over many time steps (global error) by looking at two cases, small step sizes (approaching zero, D-stability) and large step sizes. Similar results are provided in [12] in the form of (a) a bound on the global error for small step sizes and (b) a comparable stability region for finite step sizes.

6.1. Small step sizes

The ARK method given by expression (2.5) yields an approximate solution $\{\mathbf{y}_n\}$ $(2 \le n \le N)$ to the initial value problem (1.3) and can be written as

$$\begin{bmatrix} \mathbf{y}_{n+1} \\ \mathbf{y}_n \end{bmatrix} = \begin{bmatrix} c_0\mathbf{I} & -c_{-0}\mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{y}_n \\ \mathbf{y}_{n-1} \end{bmatrix} + \begin{bmatrix} c_1\mathbf{k}_1 - c_{-1}\mathbf{k}_{-1} + \sum_{i=2}^{v} c_i(\mathbf{k}_i - \mathbf{k}_{-i}) \\ \mathbf{0} \end{bmatrix} \tag{6.1}$$

for $1 \le n \le N-1$, where $\mathbf{I}$ is an m by m identity matrix. We can write (6.1) as

$$\overline{\mathbf{y}}_{n+1} = \mathbf{Q}\overline{\mathbf{y}}_n + \overline{\boldsymbol{\phi}}(\overline{\mathbf{y}}_n, \mathbf{f}, h) \quad \text{for } 1 \le n \le N-1, \tag{6.2}$$

where

$$\overline{\mathbf{y}}_{n+1} = \begin{bmatrix} \mathbf{y}_{n+1} \\ \mathbf{y}_n \end{bmatrix}, \qquad \overline{\mathbf{y}}_n = \begin{bmatrix} \mathbf{y}_n \\ \mathbf{y}_{n-1} \end{bmatrix}, \tag{6.3}$$

$\mathbf{Q}$ is the $2m$ by $2m$ block matrix given by

$$\mathbf{Q} = \begin{bmatrix} c_0\mathbf{I} & -c_{-0}\mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{bmatrix}, \tag{6.4}$$

and $\overline{\boldsymbol{\phi}}$ is the $2m$ by 1 vector defined as

$$\overline{\boldsymbol{\phi}} = \begin{bmatrix} \boldsymbol{\phi} \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} c_1\mathbf{k}_1 - c_{-1}\mathbf{k}_{-1} + \sum_{i=2}^{v} c_i(\mathbf{k}_i - \mathbf{k}_{-i}) \\ \mathbf{0} \end{bmatrix}. \tag{6.5}$$

With a slightly different initial condition $\mathbf{z}_0 = \mathbf{y}_0 + \boldsymbol{\Delta}_0$ at time t_0, a slightly different approximate solution $\mathbf{z}_1 = \mathbf{y}_1 + \boldsymbol{\Delta}_1$ at time t_1, and a slightly different recipe $\boldsymbol{\phi}(\overline{\mathbf{z}}_n, \mathbf{f}, h) + h\boldsymbol{\delta}_n$ $(1 \le n \le N-1)$, the ARK method yields a "perturbed" approximate solution $\{\mathbf{z}_n\}$ $(2 \le n \le N)$, and can be written as

$$\begin{bmatrix} \mathbf{z}_{n+1} \\ \mathbf{z}_n \end{bmatrix} = \begin{bmatrix} c_0\mathbf{I} & -c_{-0}\mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{z}_n \\ \mathbf{z}_{n-1} \end{bmatrix} + \begin{bmatrix} c_1\mathbf{k}_1 - c_{-1}\mathbf{k}_{-1} + \sum_{i=2}^{v} c_i(\mathbf{k}_i - \mathbf{k}_{-i}) \\ \mathbf{0} \end{bmatrix} + h \begin{bmatrix} \boldsymbol{\delta}_n \\ \mathbf{0} \end{bmatrix} \tag{6.6}$$

for $1 \le n \le N-1$, or can be written as

$$\overline{\mathbf{z}}_{n+1} = \mathbf{Q}\overline{\mathbf{z}}_n + \overline{\boldsymbol{\phi}}(\overline{\mathbf{z}}_n, \mathbf{f}, h) + h\overline{\boldsymbol{\delta}}_n \quad \text{for } 1 \le n \le N-1, \tag{6.7}$$

where

$$\overline{\boldsymbol{\delta}}_n = \begin{bmatrix} \boldsymbol{\delta}_n \\ \mathbf{0} \end{bmatrix}. \tag{6.8}$$

To prove stability, we will show that under some conditions, $\|\mathbf{z}_n - \mathbf{y}_n\|_\infty$ is bounded. To prove convergence, we will show that $\|\mathbf{y}(t_n) - \mathbf{y}_n\|_\infty \to 0$ as $h \to 0$, where $\mathbf{y}(t_n)$ is the exact solution of the IVP defined by (1.3). In order to do this, we will assume that the ARK method is stable and that as h tends to zero, $\|\mathbf{y}(t_1) - \mathbf{y}_1\|_\infty$ tends to zero, where t_0 is the initial time and $t_1 = t_0 + h$. The stability analysis comprises of two lemmas and two theorems. In Lemma 6.1, we will consider the eigenvalue problem for the matrix $\mathbf{Q}$. In Lemma 6.2, we will show that the function $\overline{\boldsymbol{\phi}}$ is Lipschitz continuous. In Theorem 6.3, we will establish stability of the ARK methods by looking at the difference of (6.2) and (6.7). In Theorem 6.4, we will prove convergence for ARK methods using the result of Theorem 6.3. This stability analysis is inspired by Shampine [17].

Lemma 6.1. *The eigenvalue problem for the matrix* $\mathbf{Q}$ *given in* (6.4) *can be written as*

$$\mathbf{QW} = \mathbf{WD}. \tag{6.9}$$

If $-1 \le c_{-0} < 1$, *the matrix* $\mathbf{W}$ *is nonsingular and satisfies*

$$\left\|\mathbf{W}^{-1}\mathbf{QW}\right\| = 1, \tag{6.10}$$

$$\|\mathbf{W}\| = 2, \tag{6.11}$$

$$w := \left\|\mathbf{W}^{-1}\right\| = \frac{2}{|c_{-0} - 1|}, \tag{6.12}$$

where here and throughout the stability analysis, all norms denote the infinity norm $\|\cdot\|_\infty$.

Proof. Taking

$$\mathbf{W} = \begin{bmatrix} \mathbf{I} & c_{-0}\mathbf{I} \\ \mathbf{I} & \mathbf{I} \end{bmatrix}, \qquad \mathbf{D} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & c_{-0}\mathbf{I} \end{bmatrix}, \tag{6.13}$$

where we have made use of the $O(h^0)$ order equation, $c_0 = 1 + c_{-0}$, we observe that (6.9) is satisfied. Noting that

$$\mathbf{W}^{-1} = \frac{1}{c_{-0} - 1} \begin{bmatrix} -\mathbf{I} & c_{-0}\mathbf{I} \\ \mathbf{I} & -\mathbf{I} \end{bmatrix}, \tag{6.14}$$

we require that $c_{-0} \neq 1$ for $\mathbf{W}^{-1}$ to exist. Recalling that $\|\cdot\| \equiv \|\cdot\|_\infty$, we can write

$$\left\|\mathbf{W}^{-1}\mathbf{Q}\mathbf{W}\right\| = \|\mathbf{D}\| = \max(\text{absolute row sum}) = \max\left(1, \left|c_{-0}\right|\right). \tag{6.15}$$

With the condition

$$-1 \leq c_{-0} < 1, \tag{6.16}$$

we can see from (6.15) that $\|\mathbf{W}^{-1}\mathbf{Q}\mathbf{W}\| = 1$. From condition (6.16) and expression (6.13), we find that

$$\|\mathbf{W}\| = \max\left(1 + \left|c_{-0}\right|, 2\right) = 2, \tag{6.17}$$

and from expression (6.14), we obtain

$$w := \left\|\mathbf{W}^{-1}\right\| = \frac{1}{\left|c_{-0} - 1\right|} \max\left(1 + \left|c_{-0}\right|, 2\right) = \frac{2}{\left|c_{-0} - 1\right|}. \tag{6.18}$$

Thus, Lemma 6.1 is proven. □

Lemma 6.2. *If* $\mathbf{f}$ *is a Lipschitz continuous function so that*

$$\left\|\mathbf{f}(\mathbf{z}_n) - \mathbf{f}(\mathbf{y}_n)\right\| \leq L\left\|\mathbf{z}_n - \mathbf{y}_n\right\| \quad \text{for a constant } L, \text{ and any } \mathbf{z}_n,\ \mathbf{y}_n\ (1 \leq n \leq N), \tag{6.19}$$

then, so is $\overline{\boldsymbol{\phi}}$*, that is,*

$$\left\|\overline{\boldsymbol{\phi}}(\overline{\mathbf{z}}_n) - \overline{\boldsymbol{\phi}}(\overline{\mathbf{y}}_n)\right\| \leq h\widehat{L}\left\|\overline{\mathbf{z}}_n - \overline{\mathbf{y}}_n\right\|, \quad \text{for a constant } \widehat{L}, \text{ and any } \overline{\mathbf{z}}_n,\ \overline{\mathbf{y}}_n\ (1 \leq n \leq N). \tag{6.20}$$

Proof. First, we will show by induction that

$$\left\|\mathbf{k}_i(\mathbf{z}_n) - \mathbf{k}_i(\mathbf{y}_n)\right\| \leq h\gamma_i\left\|\mathbf{z}_n - \mathbf{y}_n\right\| \quad \text{for a constant } \gamma_i\ (1 \leq i \leq v), \tag{6.21}$$

where we recall that $\|\cdot\| \equiv \|\cdot\|_\infty$, and that v is the number of function evaluations of the ARK method. Also, here and throughout this proof, $1 \leq n \leq N$. For $i = 1$, we have

$$\left\|\mathbf{k}_1(\mathbf{z}_n) - \mathbf{k}_1(\mathbf{y}_n)\right\| = \left\|h\mathbf{f}(\mathbf{z}_n) - h\mathbf{f}(\mathbf{y}_n)\right\| \leq hL\left\|\mathbf{z}_n - \mathbf{y}_n\right\| = h\gamma_1\left\|\mathbf{z}_n - \mathbf{y}_n\right\|, \tag{6.22}$$

where $\gamma_1 = L$. Assuming inequality (6.21) is true for $i = j$, we have

$$\begin{aligned}\left\|\mathbf{k}_{j+1}(\mathbf{z}_n) - \mathbf{k}_{j+1}(\mathbf{y}_n)\right\| &= \left\|h\mathbf{f}(\mathbf{z}_n + a_j\mathbf{k}_j(\mathbf{z}_n)) - h\mathbf{f}(\mathbf{y}_n + a_j\mathbf{k}_j(\mathbf{y}_n))\right\| \\ &\le hL\left\|\mathbf{z}_n - \mathbf{y}_n + a_j(\mathbf{k}_j(\mathbf{z}_n) - \mathbf{k}_j(\mathbf{y}_n))\right\| \\ &\le hL\left[\left\|\mathbf{z}_n - \mathbf{y}_n\right\| + |a_j|h\gamma_j\left\|\mathbf{z}_n - \mathbf{y}_n\right\|\right] \\ &= hL\left[1 + h\gamma_j|a_j|\right]\left\|\mathbf{z}_n - \mathbf{y}_n\right\| \\ &= h\gamma_{j+1}\left\|\mathbf{z}_n - \mathbf{y}_n\right\|,\end{aligned} \tag{6.23}$$

where $\gamma_{j+1} = L[1 + h\gamma_j|a_j|]$. Hence, inequality (6.21) holds for $j + 1$ and, therefore, it holds for $1 \le i \le v$. Similarly, we can show that

$$\left\|\mathbf{k}_{-i}(\mathbf{z}_{n-1}) - \mathbf{k}_{-i}(\mathbf{y}_{n-1})\right\| \le h\gamma_i\left\|\mathbf{z}_{n-1} - \mathbf{y}_{n-1}\right\| \quad \text{for } 1 \le i \le v. \tag{6.24}$$

Function $\boldsymbol{\phi}$ can be written as

$$\begin{aligned}\boldsymbol{\phi} &= c_1\mathbf{k}_1 - c_{-1}\mathbf{k}_{-1} + \sum_{i=2}^{v} c_i(\mathbf{k}_i - \mathbf{k}_{-i}) \\ &= \sum_{i=1}^{v} c_i\mathbf{k}_i - \sum_{i=1}^{v} c_{-i}\mathbf{k}_{-i}, \quad \text{where } c_{-i} = c_i \text{ for } 2 \le i \le v \\ &= \boldsymbol{\phi}_1 - \boldsymbol{\phi}_{-1}.\end{aligned} \tag{6.25}$$

Using inequality (6.21), we can write

$$\begin{aligned}\left\|\boldsymbol{\phi}_1(\mathbf{z}_n) - \boldsymbol{\phi}_1(\mathbf{y}_n)\right\| &= \left\|\sum_{i=1}^{v} c_i\mathbf{k}_i(\mathbf{z}_n) - \sum_{i=1}^{v} c_i\mathbf{k}_i(\mathbf{y}_n)\right\| \\ &\le \sum_{i=1}^{v}|c_i|\left\|\mathbf{k}_i(\mathbf{z}_n) - \mathbf{k}_i(\mathbf{y}_n)\right\| \\ &\le h\left[\sum_{i=1}^{v}(|c_i|\gamma_i)\right]\left\|\mathbf{z}_n - \mathbf{y}_n\right\|.\end{aligned} \tag{6.26}$$

Similarly, using inequality (6.24), we can write

$$\left\|\boldsymbol{\phi}_{-1}(\mathbf{z}_{n-1}) - \boldsymbol{\phi}_{-1}(\mathbf{y}_{n-1})\right\| \le h\left[\sum_{i=1}^{v}(|c_{-i}|\gamma_i)\right]\left\|\mathbf{z}_{n-1} - \mathbf{y}_{n-1}\right\|. \tag{6.27}$$

Since $\|\overline{\boldsymbol{\phi}}(\overline{\mathbf{z}}_n) - \overline{\boldsymbol{\phi}}(\overline{\mathbf{y}}_n)\| \equiv \|\overline{\boldsymbol{\phi}}(\overline{\mathbf{z}}_n) - \overline{\boldsymbol{\phi}}(\overline{\mathbf{y}}_n)\|_\infty = \max(\|\boldsymbol{\phi}(\overline{\mathbf{z}}_n) - \boldsymbol{\phi}(\overline{\mathbf{y}}_n)\|, 0)$, we have

$$\begin{aligned}\left\|\overline{\boldsymbol{\phi}}(\overline{\mathbf{z}}_n) - \overline{\boldsymbol{\phi}}(\overline{\mathbf{y}}_n)\right\| &= \left\|\boldsymbol{\phi}(\overline{\mathbf{z}}_n) - \boldsymbol{\phi}(\overline{\mathbf{y}}_n)\right\| \\ &= \left\|\boldsymbol{\phi}_1(\mathbf{z}_n) - \boldsymbol{\phi}_{-1}(\mathbf{z}_{n-1}) - (\boldsymbol{\phi}_1(\mathbf{y}_n) - \boldsymbol{\phi}_{-1}(\mathbf{y}_{n-1}))\right\| \\ &\le \left\|\boldsymbol{\phi}_1(\mathbf{z}_n) - \boldsymbol{\phi}_1(\mathbf{y}_n)\right\| + \left\|\boldsymbol{\phi}_{-1}(\mathbf{z}_{n-1}) - \boldsymbol{\phi}_{-1}(\mathbf{y}_{n-1})\right\| \\ &\le h\left[\sum_{i=1}^{v}(|c_i|\gamma_i)\right]\left\|\mathbf{z}_n - \mathbf{y}_n\right\| + h\left[\sum_{i=1}^{v}(|c_{-i}|\gamma_i)\right]\left\|\mathbf{z}_{n-1} - \mathbf{y}_{n-1}\right\|.\end{aligned} \tag{6.28}$$

Because $\|\overline{\mathbf{z}}_n - \overline{\mathbf{y}}_n\| = \max(\|\mathbf{z}_n - \mathbf{y}_n\|, \|\mathbf{z}_{n-1} - \mathbf{y}_{n-1}\|)$, we have

$$\|\mathbf{z}_n - \mathbf{y}_n\| \le \|\overline{\mathbf{z}}_n - \overline{\mathbf{y}}_n\|, \qquad \|\mathbf{z}_{n-1} - \mathbf{y}_{n-1}\| \le \|\overline{\mathbf{z}}_n - \overline{\mathbf{y}}_n\|. \tag{6.29}$$

Therefore,

$$\|\overline{\boldsymbol{\phi}}(\overline{\mathbf{z}}_n) - \overline{\boldsymbol{\phi}}(\overline{\mathbf{y}}_n)\| \le h\left[\sum_{i=1}^{v}|c_i|\gamma_i + \sum_{i=1}^{v}|c_{-i}|\gamma_i\right]\|\overline{\mathbf{z}}_n - \overline{\mathbf{y}}_n\| = h\widehat{L}\|\overline{\mathbf{z}}_n - \overline{\mathbf{y}}_n\|, \tag{6.30}$$

where $\widehat{L} = \sum_{i=1}^{v}(|c_i| + |c_{-i}|)\gamma_i$. Thus, Lemma 6.2 is proven. □

Theorem 6.3. *Suppose that an ARK method is used to solve the IVP given in* (1.3)*, and that* $\mathbf{f}$ *is a Lipschitz continuous function throughout the integration span. If* $-1 \le c_{-0} < 1$*, the ARKp method is stable for all sufficiently small step sizes, and*

$$\|\mathbf{z}_n - \mathbf{y}_n\| \le E\max\left(\max_{r=0,1}\|\mathbf{z}_r - \mathbf{y}_r\|, \max_{1\le m<n}\|\boldsymbol{\delta}_m\|\right) \quad \textit{for } 2 \le n \le N, \tag{6.31}$$

where

$$E = 4w\mathrm{e}^{2\widehat{L}w(t_n - t_1)}\max\left(t_n - t_1, 1\right). \tag{6.32}$$

Proof. Subtracting (6.7) from (6.2), we have

$$\overline{\mathbf{z}}_{n+1} - \overline{\mathbf{y}}_{n+1} = \mathbf{Q}\left(\overline{\mathbf{z}}_n - \overline{\mathbf{y}}_n\right) + \overline{\boldsymbol{\phi}}\left(\overline{\mathbf{z}}_n, \mathbf{f}, h\right) - \overline{\boldsymbol{\phi}}\left(\overline{\mathbf{y}}_n, \mathbf{f}, h\right) + h\overline{\boldsymbol{\delta}}_n \quad \text{for } 1 \le n \le N-1. \tag{6.33}$$

We multiply this equation by $\mathbf{W}^{-1}$ to get

$$\mathbf{W}^{-1}\left(\overline{\mathbf{z}}_{n+1} - \overline{\mathbf{y}}_{n+1}\right) = \mathbf{W}^{-1}\mathbf{Q}\left(\overline{\mathbf{z}}_n - \overline{\mathbf{y}}_n\right) + \mathbf{W}^{-1}\left[\overline{\boldsymbol{\phi}}\left(\overline{\mathbf{z}}_n, \mathbf{f}, h\right) - \overline{\boldsymbol{\phi}}\left(\overline{\mathbf{y}}_n, \mathbf{f}, h\right)\right] + h\mathbf{W}^{-1}\overline{\boldsymbol{\delta}}_n. \tag{6.34}$$

Let

$$\mathbf{d}_n = \mathbf{W}^{-1}\left(\overline{\mathbf{z}}_n - \overline{\mathbf{y}}_n\right) \quad \text{so that } \mathbf{W}\mathbf{d}_n = \left(\overline{\mathbf{z}}_n - \overline{\mathbf{y}}_n\right) \text{ for } 1 \le n \le N, \tag{6.35}$$

and we get

$$\mathbf{d}_{n+1} = \mathbf{W}^{-1}\mathbf{Q}\mathbf{W}\mathbf{d}_n + \mathbf{W}^{-1}\left[\overline{\boldsymbol{\phi}}\left(\overline{\mathbf{z}}_n, \mathbf{f}, h\right) - \overline{\boldsymbol{\phi}}\left(\overline{\mathbf{y}}_n, \mathbf{f}, h\right)\right] + h\mathbf{W}^{-1}\overline{\boldsymbol{\delta}}_n. \tag{6.36}$$

Taking the infinity norm and using the Lipschitz condition on $\overline{\boldsymbol{\phi}}$ (Lemma 6.2), we can write

$$\|\mathbf{d}_{n+1}\| \le \|\mathbf{W}^{-1}\mathbf{Q}\mathbf{W}\|\|\mathbf{d}_n\| + h\widehat{L}\|\mathbf{W}^{-1}\|\|\overline{\mathbf{z}}_n - \overline{\mathbf{y}}_n\| + h\|\mathbf{W}^{-1}\|\|\overline{\boldsymbol{\delta}}_n\| \quad \text{for } 1 \le n \le N-1. \tag{6.37}$$

Considering expressions (6.10) and (6.12), we have

$$\|\mathbf{d}_{n+1}\| \le \|\mathbf{d}_n\| + h\widehat{L}w\|\overline{\mathbf{z}}_n - \overline{\mathbf{y}}_n\| + hw\|\overline{\boldsymbol{\delta}}_n\|. \tag{6.38}$$

Also, using expressions (6.35) and (6.11), we have

$$\left\|\overline{\mathbf{z}}_n - \overline{\mathbf{y}}_n\right\| = \left\|\mathbf{W}\mathbf{d}_n\right\| \le \|\mathbf{W}\|\left\|\mathbf{d}_n\right\| = 2\left\|\mathbf{d}_n\right\|, \tag{6.39}$$

so inequality (6.38) becomes

$$\left\|\mathbf{d}_{n+1}\right\| \le \left(1 + 2h\widehat{L}w\right)\left\|\mathbf{d}_n\right\| + hw\left\|\overline{\boldsymbol{\delta}}_n\right\| \quad \text{for } 1 \le n \le N-1. \tag{6.40}$$

Relation (6.40) establishes a bound on $\|\mathbf{d}_{n+1}\|$ based on $\|\mathbf{d}_n\|$. This is not very useful because we do not know how large $\|\mathbf{d}_n\|$ is. Instead, we want to find a bound based on $\|\mathbf{d}_1\|$. We will now show by induction that this bound can be written as

$$\left\|\mathbf{d}_n\right\| \le \mathrm{e}^{2\widehat{L}w(t_n - t_1)}\left\|\mathbf{d}_1\right\| + hw\sum_{m=1}^{n-1}\mathrm{e}^{2\widehat{L}w(t_n - t_{m+1})}\left\|\overline{\boldsymbol{\delta}}_m\right\| \quad \text{for } 2 \le n \le N. \tag{6.41}$$

In the following, we will use the inequality

$$1 \le 1 + t \le \mathrm{e}^t \quad \text{for } t \ge 0, \tag{6.42}$$

which follows from the expression $\mathrm{e}^t = 1 + t + (1/2)t^2 + \cdots$. For $n = 2$, the right hand side of inequality (6.41) becomes

$$\mathrm{e}^{2\widehat{L}w(t_2 - t_1)}\left\|\mathbf{d}_1\right\| + hw\mathrm{e}^{2\widehat{L}w(t_2 - t_2)}\left\|\overline{\boldsymbol{\delta}}_1\right\| = \mathrm{e}^{2\widehat{L}wh}\left\|\mathbf{d}_1\right\| + hw\left\|\overline{\boldsymbol{\delta}}_1\right\|. \tag{6.43}$$

Inequality (6.40) can be written as

$$\left\|\mathbf{d}_n\right\| \le \left(1 + 2h\widehat{L}w\right)\left\|\mathbf{d}_{n-1}\right\| + hw\left\|\overline{\boldsymbol{\delta}}_{n-1}\right\| \quad \text{for } 2 \le n \le N, \tag{6.44}$$

and for $n = 2$ as

$$\left\|\mathbf{d}_2\right\| \le \left(1 + 2h\widehat{L}w\right)\left\|\mathbf{d}_1\right\| + hw\left\|\overline{\boldsymbol{\delta}}_1\right\| \le \mathrm{e}^{2\widehat{L}wh}\left\|\mathbf{d}_1\right\| + hw\left\|\overline{\boldsymbol{\delta}}_1\right\|. \tag{6.45}$$

Comparing expressions (6.45) and (6.43), we can see that inequality (6.41) holds for $n = 2$. Assuming it holds for $n = k$, we have from inequality (6.40) that

$$\begin{aligned}\left\|\mathbf{d}_{k+1}\right\| &\le \left(1 + 2h\widehat{L}w\right)\left\|\mathbf{d}_k\right\| + hw\left\|\overline{\boldsymbol{\delta}}_k\right\| \quad \text{for } 1 \le k \le N-1 \\ &\le \mathrm{e}^{2h\widehat{L}w}\left(\mathrm{e}^{2\widehat{L}w(t_k - t_1)}\left\|\mathbf{d}_1\right\| + hw\sum_{m=1}^{k-1}\mathrm{e}^{2\widehat{L}w(t_k - t_{m+1})}\left\|\overline{\boldsymbol{\delta}}_m\right\|\right) + hw\left\|\overline{\boldsymbol{\delta}}_k\right\|.\end{aligned} \tag{6.46}$$

Since for any r, $h + (t_k - t_r) = t_{k+1} - t_r$, we have that

$$\begin{aligned}\left\|\mathbf{d}_{k+1}\right\| &\le \mathrm{e}^{2\widehat{L}w(t_{k+1} - t_1)}\left\|\mathbf{d}_1\right\| + hw\sum_{m=1}^{k-1}\mathrm{e}^{2\widehat{L}w(t_{k+1} - t_{m+1})}\left\|\overline{\boldsymbol{\delta}}_m\right\| + hw\mathrm{e}^{2\widehat{L}w(t_{k+1} - t_{k+1})}\left\|\overline{\boldsymbol{\delta}}_k\right\| \\ &= \mathrm{e}^{2\widehat{L}w(t_{k+1} - t_1)}\left\|\mathbf{d}_1\right\| + hw\sum_{m=1}^{k}\mathrm{e}^{2\widehat{L}w(t_{k+1} - t_{m+1})}\left\|\overline{\boldsymbol{\delta}}_m\right\| \quad \text{for } 1 \le k \le N-1.\end{aligned} \tag{6.47}$$

Therefore, inequality (6.41) holds for $k + 1$ and by induction, it holds for $2 \le n \le N$.

We can simplify inequality (6.41) by writing

$$\begin{aligned}
\|\mathbf{d}_n\| &\le e^{2\hat{L}w(t_n-t_1)}\|\mathbf{d}_1\| + hw\sum_{m=1}^{n-1} e^{2\hat{L}w(t_n-t_{m+1})}\|\overline{\boldsymbol{\delta}}_m\| \quad \text{for } 2 \le n \le N \\
&\le e^{2\hat{L}w(t_n-t_1)}\|\mathbf{d}_1\| + hwe^{2\hat{L}w(t_n-t_2)}(n-1)\max_{1\le m<n}\|\overline{\boldsymbol{\delta}}_m\| \\
&\le e^{2\hat{L}w(t_n-t_1)}\|\mathbf{d}_1\| + hwe^{2\hat{L}w(t_n-t_1)}(n-1)\max_{1\le m<n}\|\overline{\boldsymbol{\delta}}_m\| \\
&= e^{2\hat{L}w(t_n-t_1)}\|\mathbf{W}^{-1}(\overline{\mathbf{z}}_1-\overline{\mathbf{y}}_1)\| + we^{2\hat{L}w(t_n-t_1)}(t_n-t_1)\max_{1\le m<n}\|\overline{\boldsymbol{\delta}}_m\| \\
&\le e^{2\hat{L}w(t_n-t_1)}w\left[\|\overline{\mathbf{z}}_1-\overline{\mathbf{y}}_1\| + (t_n-t_1)\max_{1\le m<n}\|\overline{\boldsymbol{\delta}}_m\|\right] \\
&\le e^{2\hat{L}w(t_n-t_1)}w\left[\max(t_n-t_1,1)\|\overline{\mathbf{z}}_1-\overline{\mathbf{y}}_1\| + \max(t_n-t_1,1)\max_{1\le m<n}\|\overline{\boldsymbol{\delta}}_m\|\right] \\
&\le 2we^{2\hat{L}w(t_n-t_1)}\max(t_n-t_1,1)\max\left(\|\overline{\mathbf{z}}_1-\overline{\mathbf{y}}_1\|, \max_{1\le m<n}\|\overline{\boldsymbol{\delta}}_m\|\right).
\end{aligned} \tag{6.48}$$

Using the inequality (6.39), we have

$$\|\overline{\mathbf{z}}_n-\overline{\mathbf{y}}_n\| \le 4we^{2\hat{L}w(t_n-t_1)}\max(t_n-t_1,1)\max\left(\|\overline{\mathbf{z}}_1-\overline{\mathbf{y}}_1\|, \max_{1\le m<n}\|\overline{\boldsymbol{\delta}}_m\|\right). \tag{6.49}$$

Since $\|\mathbf{z}_n-\mathbf{y}_n\| \le \|\overline{\mathbf{z}}_n-\overline{\mathbf{y}}_n\|$, and $\|\overline{\boldsymbol{\delta}}_m\| = \|\boldsymbol{\delta}_m\|$, we have

$$\|\mathbf{z}_n-\mathbf{y}_n\| \le E\max\left(\max_{r=0,1}\|\mathbf{z}_r-\mathbf{y}_r\|, \max_{1\le m<n}\|\boldsymbol{\delta}_m\|\right) \quad \text{for } 2 \le n \le N, \tag{6.50}$$

where

$$E = 4we^{2\hat{L}w(t_n-t_1)}\max(t_n-t_1,1). \tag{6.51}$$

Hence, stability of ARK methods is established. Recall that the ARK methods require to be provided with the initial condition $\mathbf{y}_0$ at the beginning of the first time step t_0 and the approximate solution $\mathbf{y}_1$ at the end of the first time step t_1. The $\max_{r=0,1}\|\mathbf{z}_r-\mathbf{y}_r\|$ term in expression (6.31) refers to the perturbations in these values. □

Theorem 6.4. *Consider an ARKp method that is used to solve the IVP given by* (1.3)*, where* $\mathbf{f}$ *is a sufficiently smooth Lipschitz continuous function throughout the integration span. If* $-1 \le c_{-0} < 1$ *and the approximate solution* $\mathbf{y}_1$ *at time* t_1 *is accurate to* $O(h^q)$*, the ARKp method is convergent to order* $\min(p,q)$*, and*

$$\|\mathbf{y}(t_n)-\mathbf{y}_n\| \le E\max(O(h^q),O(h^p)) \quad \textit{for } 2 \le n \le N, \tag{6.52}$$

where

$$E = 4we^{2\hat{L}w(t_n-t_1)}\max(t_n-t_1,1). \tag{6.53}$$

In particular, $\|\mathbf{y}(t_n)-\mathbf{y}_n\| \to 0$ *as* $h \to 0$.

Proof. Consider again the approximate solution $\{\mathbf{y}_n\}$ $(2 \le n \le N)$ and the perturbed solution $\{\mathbf{z}_n\}$ $(2 \le n \le N)$ of the ARKp method given by

$$\mathbf{y}_{n+1} = c_0\mathbf{y}_n - c_{-0}\mathbf{y}_{n-1} + \boldsymbol{\phi}(\mathbf{y}_n, \mathbf{y}_{n-1}, \mathbf{f}, h) \quad \text{for } 1 \le n \le N-1, \tag{6.54}$$

$$\mathbf{z}_{n+1} = c_0\mathbf{z}_n - c_{-0}\mathbf{z}_{n-1} + \boldsymbol{\phi}(\mathbf{z}_n, \mathbf{z}_{n-1}, \mathbf{f}, h) + h\boldsymbol{\delta}_n \quad \text{for } 1 \le n \le N-1, \tag{6.55}$$

where $\mathbf{z}_0 = \mathbf{y}_0 + \boldsymbol{\Delta}_0$, $\mathbf{z}_1 = \mathbf{y}_1 + \boldsymbol{\Delta}_1$, and $\mathbf{z}_n$'s recipe is perturbed by $h\boldsymbol{\delta}_n$. The truncation error $h\boldsymbol{\delta}_n^*$ is defined as the difference between the exact and approximate solutions at the end of the current step if the method is provided with the exact solution at the beginning of the current and previous steps. Therefore, the exact solution $\mathbf{y}(t_n)$ satisfies

$$\mathbf{y}(t_{n+1}) = c_0\mathbf{y}(t_n) - c_{-0}\mathbf{y}(t_{n-1}) + \boldsymbol{\phi}(\mathbf{y}(t_n), \mathbf{y}(t_{n-1}), \mathbf{f}, h) + h\boldsymbol{\delta}_n^*. \tag{6.56}$$

We will now show that $\{\mathbf{y}(t_n)\}$ $(2 \le n \le N)$ is a perturbed solution of the ARK method. For $n = 1$, we have

$$\mathbf{y}_2 = c_0\mathbf{y}_1 - c_{-0}\mathbf{y}_0 + \boldsymbol{\phi}(\mathbf{y}_1, \mathbf{y}_0, \mathbf{f}, h), \tag{6.57}$$

$$\mathbf{z}_2 = c_0\mathbf{z}_1 - c_{-0}\mathbf{z}_0 + \boldsymbol{\phi}(\mathbf{z}_1, \mathbf{z}_0, \mathbf{f}, h) + h\boldsymbol{\delta}_1, \tag{6.58}$$

$$\mathbf{y}(t_2) = c_0\mathbf{y}(t_1) - c_{-0}\mathbf{y}(t_0) + \boldsymbol{\phi}(\mathbf{y}(t_1), \mathbf{y}(t_0), \mathbf{f}, h) + h\boldsymbol{\delta}_1^*. \tag{6.59}$$

We set

$$\begin{aligned} \mathbf{z}_0 &= \mathbf{y}(t_0) = \mathbf{y}_0, \\ \mathbf{z}_1 &= \mathbf{y}(t_1) = \mathbf{y}_1 + \boldsymbol{\Delta}_1^*, \\ \boldsymbol{\delta}_1 &= \boldsymbol{\delta}_1^*, \end{aligned} \tag{6.60}$$

and from (6.58) and (6.59), we have $\mathbf{z}_2 = \mathbf{y}(t_2)$. Similarly, we set $\boldsymbol{\delta}_n = \boldsymbol{\delta}_n^*$ for $1 \le n \le N-1$, and from (6.55) and (6.56), we get $\mathbf{z}_n = \mathbf{y}(t_n)$ for $2 \le n \le N$.

Since we have made the same assumptions as Theorem 6.3, relations (6.31) and (6.32) hold, and we have

$$\|\mathbf{y}(t_n) - \mathbf{y}_n\| \le E \max\left(\|\mathbf{y}(t_1) - \mathbf{y}_1\|, \max_{1\le m<n} \|\boldsymbol{\delta}_m^*\|\right) \quad \text{for } 2 \le n \le N. \tag{6.61}$$

For ARKp, $\boldsymbol{\delta}_m^* = O(h^p)$. The approximate solution $\mathbf{y}_1$ at time t_1 which is required by the ARKp method is supplied by a one-step method such as RKp. If $\mathbf{y}_1$ is accurate of order q, that is, $\|\mathbf{y}(t_1) - \mathbf{y}_1\| = O(h^q)$, the above inequality shows that the ARKp method is convergent of order $\min(p, q)$, that is,

$$\|\mathbf{y}(t_n) - \mathbf{y}_n\| \le E \max\left(O(h^q), O(h^p)\right) \quad \text{for } 2 \le n \le N. \tag{6.62}$$

In particular, $\|\mathbf{y}(t_n) - \mathbf{y}_n\| \to 0$ as $h \to 0$, and Theorem 6.4 is proven. □

6.2. Large step sizes

To produce the stability region of ARK methods for finite step sizes, we look at the approximate solution of the linear one-dimensional problem $y' = \lambda y$, where λ is a complex number. After some simplification using the order equations, the ARK method's approximate solution of this problem becomes

$$\begin{bmatrix} y_{n+1} \\ y_n \end{bmatrix} = \begin{bmatrix} 1 + c_{-0} + \frac{1}{2}(3 - c_{-0})(\lambda h) + S & -c_{-0} - \frac{1}{2}(1 + c_{-0})(\lambda h) - S \\ 1 & 0 \end{bmatrix} \begin{bmatrix} y_n \\ y_{n-1} \end{bmatrix}, \tag{6.63}$$

where for ARK3,

$$S = \frac{1}{12}(5 + c_{-0})(\lambda h)^2, \tag{6.64}$$

for ARK4,

$$S = \frac{1}{12}(5 + c_{-0})(\lambda h)^2 + \frac{1}{6}(\lambda h)^3, \tag{6.65}$$

for ARK4-4,

$$S = \frac{1}{12}(5 + c_{-0})(\lambda h)^2 + \frac{1}{6}(\lambda h)^3 + c_4 a_1 a_2 a_3 (\lambda h)^4, \tag{6.66}$$

and for ARK5,

$$S = \frac{1}{12}(5 + c_{-0})(\lambda h)^2 + \frac{1}{6}(\lambda h)^3 + \frac{1}{720}(31 - c_{-0})(\lambda h)^4 + c_5 a_1 a_2 a_3 a_4 (\lambda h)^5. \tag{6.67}$$

For stability, we require that the modulus of the eigenvalues of the matrix in (6.63) be smaller than 1. Figures 15, 16, 17, and 18 show the stability regions of ARK and RK methods produced in this way. The curves represent the points at which the computed maximum modulus of the eigenvalues equals 1. The region of stability is the area enclosed by the curves and the real axis. It is clear from (6.64) and (6.65) that the stability region of ARK3 and ARK4 depends only on c_{-0}, but the stability of ARK4-4 and ARK5 depends also on the products $c_4 a_1 a_2 a_3$ and $c_5 a_1 a_2 a_3 a_4$, respectively.

7. Conclusion

In this paper, we have developed a simple set of constant step size, explicit, accelerated Runge-Kutta (ARK) methods for the numerical integration of initial value problems. They rely on the general form posited in (2.5). The simplicity of this form facilitates their detailed treatment, an aspect missing in previous work on similar multistep methods. Specifically, we have provided a study of their motivation, derivation, optimization, accuracy, speedup, stability, and convergence.

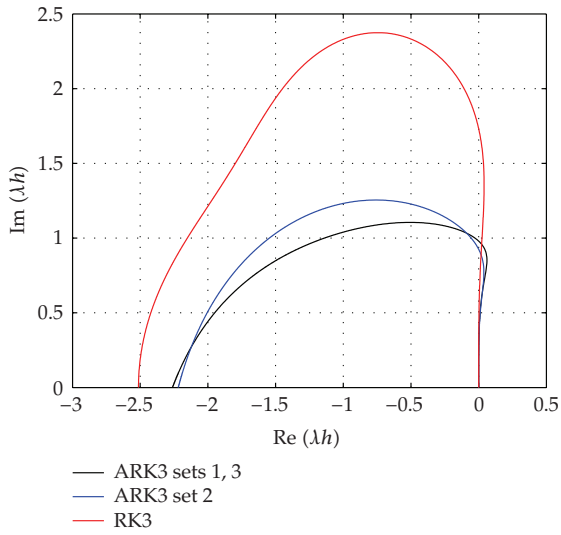


Figure 15: Stability region of ARK3 sets of Table 1 and RK3 set of Table 5.

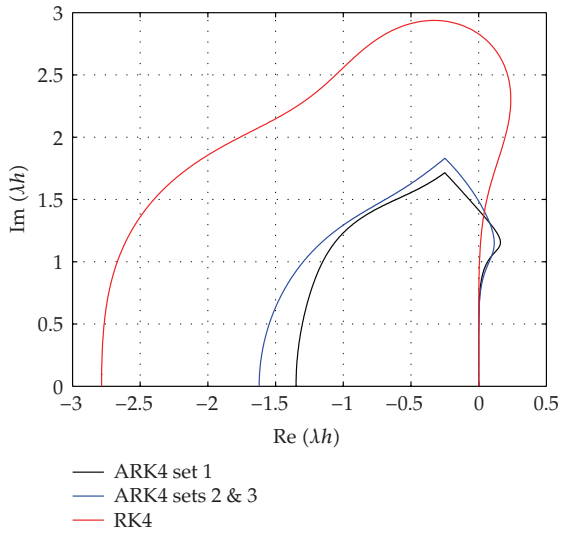


Figure 16: Stability region of ARK4 sets of Table 2 and RK4 set of Table 5.

Our main conclusions are as follows.

(1) The ARK methods developed in this paper are simple to analyze and implement and they lead to an elegant set of order equations. Three accuracy-optimized parameter sets for ARK3, ARK4, ARK4-4, and ARK5 have been provided. The accuracy of the methods obtained from these sets is confirmed by numerically solving a standard set of seven initial value problems.

(2) The ARK methods are more computationally efficient than RK methods at providing the same order of local accuracy because they reuse function evaluations from

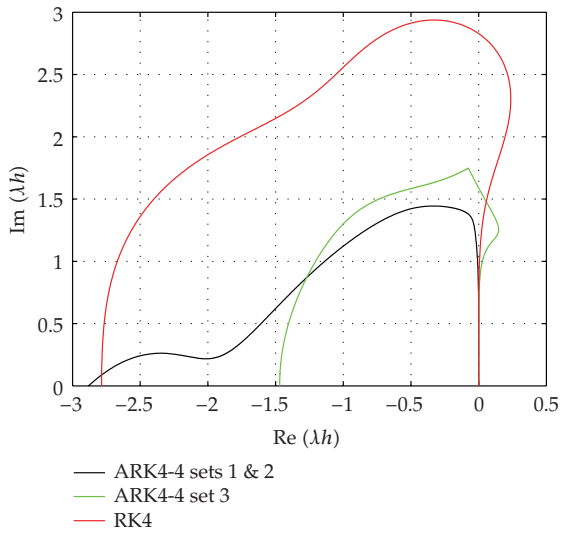


Figure 17: Stability region of ARK4-4 sets of Table 3 and RK4 set of Table 5.

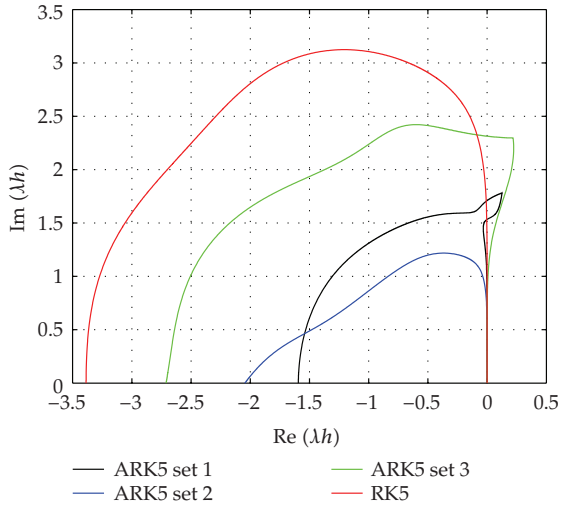


Figure 18: Stability region of ARK5 sets of Table 4 and RK5 set of Table 5.

previous steps of integration. ARK3, ARK4, and ARK5 require one less function evaluation per time step of integration than RK3, RK4, and RK5, respectively.

(3) Numerical examples show that the accuracy-optimized ARK3, ARK4, ARK4-4, and ARK5 methods presented here are superior to RK methods that computationally cost the same in terms of the number of function evaluations. Also, the ARK methods are comparable to RK methods that have the same order of local accuracy.

(4) ARK methods require to be initially started by a one-step method of equal or higher order such as an RK method; however, this extra computation occurs only at the first

step of integration and is insignificant compared to the computational savings of ARK methods over the subsequent steps.

(5) In solving the standard initial value problems considered here, the ARK3, ARK4, and ARK5 methods exhibit minimum speedups of 19%, 17%, and 15% compared to RK3, RK4, and RK5 methods, respectively, on our computer. However, the theoretical speedups of the ARK methods are approximately 33%, 25%, and 17%, respectively, based on the smaller number of function evaluations required. This reduction in speedup is found to be caused by the higher computational overheads in the ARK methods, when compared to the RK methods.

(6) Convergence and stability for small step sizes are proven for ARK methods with some conditions. The stability regions of ARK methods for large step sizes are shown to be generally smaller than, but comparable to, those of RK methods.

Appendices

A. Standard initial value problems

The following are seven initial value problems (IVPs) that we have chosen from the literature for numerical experiments. Most of the problems model real-world phenomena (suggested by [14]) and have been used by other researchers to test numerical integrators. They help to illustrate the accuracy, speedup, and stability of ARK methods in comparison with RK methods. These problems are solved for $0 \le t \le 15$.

The first problem is a simple IVP which has the form

IVP-1 [14]

$$y' = -y, \quad y(t_0) = 1, \quad \text{with solution } y(t) = e^{-t}. \tag{A.1}$$

The second problem is an example of a simple nonautonomous ODE. We use it to illustrate that such problems can be readily converted to autonomous form and solved by ARK methods without difficulty. The problem is written as

IVP-2 [11]

$$y' = -\frac{ty}{1+t^2}, \qquad y(t_0) = 1, \quad \text{with solution } y(t) = \frac{1}{\sqrt{1+t^2}}. \tag{A.2}$$

The third problem is the Euler equations of motion for a rigid body without external forces. This problem has no closed form solution, so its exact solution is approximated by RK6 with a step size of 10^{-4}. We have

IVP-3 [14]

$$\begin{aligned} y_1' &= y_2 y_3, & y_1(t_0) &= 0, \\ y_2' &= -y_1 y_3, & y_2(t_0) &= 1, \\ y_3' &= -0.51 y_1 y_2, & y_3(t_0) &= 1. \end{aligned} \tag{A.3}$$

The fourth problem is a 1-body gravitational problem with eccentricity $e = 0.8$. The solution, which is the orbit of a body revolving around a center body, has regions of high

stability and low stability as pointed out by the high and low values of the spectral norm of the Jacobian. We use this problem to show the effects of such a stability behavior on the ARK methods' performance. The ARK and RK methods fail to maintain their order of accuracy for step sizes larger than about 0.01. This is due to the instability of the solution particulary in the region where the revolving body is passing close to the center body. This problem is commonly solved by employing a variable-step numerical method which reduces the step size in this region. The problem is written as

IVP-4 [14]

$$\begin{aligned}
&y_1' = y_3, && y_1(t_0) = 1 - e, && y_1(t) = \cos(u) - e,\\
&y_2' = y_4, && y_2(t_0) = 0, && y_2(t) = \sqrt{1-e^2}\,\sin(u),\\
&y_3' = -\frac{y_1}{\left(y_1^2 + y_2^2\right)^{3/2}}, && y_3(t_0) = 0, \quad \text{with solution } && y_3(t) = \frac{-\sin(u)}{1 - e\cos(u)},\\
&y_4' = -\frac{y_2}{\left(y_1^2 + y_2^2\right)^{3/2}}, && y_4(t_0) = \sqrt{\frac{1+e}{1-e}}, && y_4(t) = \frac{\sqrt{1-e^2}\cos(u)}{1 - e\cos(u)},
\end{aligned} \tag{A.4}$$

where u is the solution of the Kepler equation $u = t + e\sin(u)$.

The fifth problem is similar to the above problem but with $e = 0$. With consistent stability behavior, it is a good example to compare with the previous problem. It has the form

IVP-5 [14]

$$\begin{aligned}
&y_1' = y_3, && y_1(t_0) = 1, && y_1(t) = \cos(t),\\
&y_2' = y_4, && y_2(t_0) = 0, && y_2(t) = \sin(t),\\
&y_3' = -\frac{y_1}{\left(y_1^2 + y_2^2\right)^{3/2}}, && y_3(t_0) = 0, \quad \text{with solution } && y_3(t) = -\sin(t),\\
&y_4' = -\frac{y_2}{\left(y_1^2 + y_2^2\right)^{3/2}}, && y_4(t_0) = 1, && y_4(t) = \cos(t).
\end{aligned} \tag{A.5}$$

The sixth problem is a linear 10th-order system that models a radioactive decay chain. This problem and the next are used to measure the performance of the ARK methods when solving larger systems of ODEs. Although this problem has a closed form solution, because of numerical errors in computing it, it was approximated by RK6 with a step size of 10^{-4}. It takes the form

IVP-6 [14]

$$\begin{bmatrix} y_1' \\ y_2' \\ \vdots \\ y_{10}' \end{bmatrix} = \begin{bmatrix} -1 & & & & & & & & & \\ 1 & -2 & & & 0 & & & & & \\ & 2 & -3 & & & & & & & \\ & & \cdot & \cdot & & & & & & \\ & & & \cdot & \cdot & & & & & \\ & & & & \cdot & \cdot & & & & \\ & 0 & & & & 8 & -9 & & & \\ & & & & & & 9 & 0 & & \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{10} \end{bmatrix}, \qquad y(t_0) = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{A.6}$$

The seventh problem is a 5-body gravitational problem for the orbit of 5 outer planets around the sun and 4 inner planets. The orbit eccentricity of the planets 1 through 5 are 0.31, 0.33, 0.30, 0.13, and 0.63, respectively. The subscripts p and c denote the planet and coordinate, respectively. The exact solution of this problem was also approximated by RK6 with a step size of 10^{-4}. It is a 30-dimensional system of first-order ODEs, and can be written as

IVP-7 [14]

$$y''_{pc} = G\left[-(m_0+m_p)\frac{y_{pc}}{r_p^3} + \sum_{\substack{k=1\\k\neq p}}^{5} m_k\left(\frac{y_{kc}-y_{pc}}{d_{pk}^3} - \frac{y_{kc}}{r_k^3}\right)\right], \quad p = 1,\ldots,5,\ c = 1,\ldots,3, \tag{A.7}$$

where

$$r_p^2 = \sum_{c=1}^{3} y_{pc}^2, \qquad d_{kp}^2 = \sum_{c=1}^{3}\left(y_{kc}-y_{pc}\right)^2, \quad k = 1,\ldots,5. \tag{A.8}$$

The gravity constant and planet masses are

$$\begin{aligned}
&G = 2.95912208286, && m_3 = 0.0000437273164546 \text{ (Uranus)},\\
&m_0 = 1.00000597682 \text{ (sun \& 4 inner planets)}, && m_4 = 0.0000517759138449 \text{ (Neptune)},\\
&m_1 = 0.000954786104043 \text{ (Jupiter)}, && m_5 = 0.00000277777777778 \text{ (Pluto)}.\\
&m_2 = 0.000285583733151 \text{ (Saturn)},
\end{aligned} \tag{A.9}$$

The initial values are

$$\begin{aligned}
&y_{11}(t_0) = 3.42947415189, && y'_{11}(t_0) = -0.557160570446,\\
&y_{12}(t_0) = 3.35386959711, && y'_{12}(t_0) = 0.505696783289,\\
&y_{13}(t_0) = 1.35494901715, && y'_{13}(t_0) = 0.230578543901,\\
&y_{21}(t_0) = 6.64145542550, && y'_{21}(t_0) = -0.415570776342,\\
&y_{22}(t_0) = 5.97156957878, && y'_{22}(t_0) = 0.365682722812,\\
&y_{23}(t_0) = 2.18231499728, && y'_{23}(t_0) = 0.169143213293,\\
&y_{31}(t_0) = 11.2630437207, && y'_{31}(t_0) = -0.325325669158,\\
&y_{32}(t_0) = 14.6952576794, && y'_{32}(t_0) = 0.189706021964,\\
&y_{33}(t_0) = 6.27960525067, && y'_{33}(t_0) = 0.0877265322780,\\
&y_{41}(t_0) = -30.1552268759, && y'_{41}(t_0) = -0.0240476254170,\\
&y_{42}(t_0) = 1.65699966404, && y'_{42}(t_0) = -0.287659532608,\\
&y_{43}(t_0) = 1.43785752721, && y'_{43}(t_0) = -0.117219543175,\\
&y_{51}(t_0) = -21.1238353380, && y'_{51}(t_0) = -0.176860753121,\\
&y_{52}(t_0) = 28.4465098142, && y'_{52}(t_0) = -0.216393453025,\\
&y_{53}(t_0) = 15.3882659679, && y'_{53}(t_0) = -0.0148647893090.
\end{aligned} \tag{A.10}$$

B. Maple code for ARK3

The following is the Maple code that generates the ARK3 order equations of up to $O(h^4)$. ARK4, ARK4-4, and ARK5 codes are extensions to this code. They were not included for brevity.

```
restart:
tay_o := 4:
yp[1] := D(y[1])(t) = f[1](y[1](t),y[2](t)):
yp[2] := D(y[2])(t) = f[2](y[1](t),y[2](t)):
y2p[1] := (D@@2)(y[1])(t) = convert(diff(f[1](y[1](t),y[2](t)), t), D):
y2p[1] := subs({yp[1], yp[2]}, y2p[1]):
y2p[2] := (D@@2)(y[2])(t) = convert(diff(f[2](y[1](t),y[2](t)), t), D):
y2p[2] := subs({yp[1], yp[2]}, y2p[2]):
y3p[1] := (D@@3)(y[1])(t) = expand( convert(diff( f[1](y[1](t),
y[2](t)), t$2), D) ):
y3p[1] := subs({y2p[1], y2p[2]}, y3p[1]):
y3p[1] := subs({yp[1], yp[2]}, y3p[1]):
y3p[1] := expand(y3p[1]):
y3p[2] := (D@@3)(y[2])(t) = expand( convert(diff( f[2](y[1](t),y[2](t)),
 t$2), D) ):
y3p[2] := subs({y2p[1], y2p[2]}, y3p[2]):
y3p[2] := subs({yp[1], yp[2]}, y3p[2]):
y3p[2] := expand(y3p[2]):
y4p[1] := (D@@4)(y[1])(t) = expand( convert(diff( f[1](y[1](t),y[2](t)),
 t$3), D) ):
y4p[1] := subs({y3p[1],y3p[2]}, y4p[1]):
y4p[1] := subs({y2p[1],y2p[2]}, y4p[1]):
y4p[1] := subs({yp[1],yp[2]}, y4p[1]):
y4p[1] := expand(y4p[1]):
y4p[2] := (D@@4)(y[2])(t) = expand( convert(diff( f[2](y[1](t),y[2](t)),
 t$3), D) ):
y4p[2] := subs({y3p[1],y3p[2]}, y4p[2]):
y4p[2] := subs({y2p[1],y2p[2]}, y4p[2]):
y4p[2] := subs({yp[1],yp[2]}, y4p[2]):
y4p[2] := expand(y4p[2]):
ynm1[1] := convert( taylor(y[1](t-h), h=0, tay_o+1), polynom):
k1[1] := h*f[1](y[1](t),y[2](t)):
k1[2] := h*f[2](y[1](t),y[2](t)):
km1[1] := collect(h*convert(taylor(f[1](y[1](t-h), y[2](t-h)), h=0,
tay_o), polynom), h):
km1[1] := sort(km1[1], h, ascending):
km1[2] := collect(h*convert(taylor(f[2](y[1](t-h), y[2](t-h)), h=0,
tay_o),
polynom), h):
km1[2] := sort(km1[2], h, ascending):
k2[1] := expand( h * convert( taylor( f[1](y[1](t)+a1*k1[1],
 y[2](t)+a1*k1[2]),
```

```
h=0, tay_o), polynom) ):
k2[2] := expand( h * convert( taylor( f[2](y[1](t)+a1*k1[1],
 y[2](t)+a1*k1[2]),
h=0, tay_o), polynom) ):
km2[1] := collect(h*convert(taylor(f[1](y[1](t-h)+a1*km1[1],
 y[2](t-h)+a1*km1[2]),
h=0, tay_o), polynom), h):
km2[1] := sort(km2[1], h, ascending):
km2[2] := collect(h*convert(taylor(f[2](y[1](t-h)+a1*km1[1],
 y[2](t-h)+a1*km1[2]),
h=0, tay_o), polynom), h):
km2[2] := sort(km2[2], h, ascending):
ynp1[1] := c0*y[1](t) - cm0*ynm1[1] + c1*k1[1]
- cm1*km1[1] + c2*(k2[1]-km2[1]):
ynp1[1] := subs({yp[1], yp[2], y2p[1], y2p[2], y3p[1], y3p[2], y4p[1], y4p[2]},
ynp1[1]):
ynp1[1] := collect(expand(ynp1[1]), h):
ynp1_tay[1] := convert( taylor(y[1](t+h), h=0, tay_o+1), polynom ):
h0term0[1] := coeff(ynp1[1], h, 0):
h0term[1] := collect(h0term0[1], y[1](t)):
eqn0 := coeff(h0term[1], y[1](t)) = coeff( coeff(ynp1_tay[1], h, 0),
y[1](t) ):
h1term0[1] := coeff(ynp1[1], h, 1):
h1term[1] := applyrule( rhs(yp[1])=lhs(yp[1]), h1term0[1]):
h1term[1] := collect(h1term[1], D(y[1])(t)):
eqn1 := coeff(h1term[1], D(y[1])(t)) = coeff( coeff(ynp1_tay[1], h, 1),
D(y[1])(t)):
h2term0[1] := coeff(ynp1[1], h, 2):
( D[1](f[1])(y[1](t),y[2](t)) = solve(y2p[1], D[1](f[1])(y[1](t),
y[2](t))) ) * f[1](y[1](t),y[2](t)):
h2term[1] := applyrule(%, h2term0[1]):
h2term[1] := collect(h2term[1], (D@@2)(y[1])(t)):
eqn2 := coeff(h2term[1], (D@@2)(y[1])(t)) = coeff( coeff(ynp1_tay[1], h, 2),
(D@@2)(y[1])(t)):
h3term0[1] := coeff(ynp1[1], h, 3):
( D[1,1](f[1])(y[1](t),y[2](t)) = solve(y3p[1], D[1,1](f[1])(y[1](t),
y[2](t))) ) * f[1](y[1](t),y[2](t))^2:
h3term[1] := applyrule(%, h3term0[1]):
h3term[1] := collect(expand(h3term[1]), (D@@3)(y[1])(t)):
eqn3 := coeff(h3term[1], (D@@3)(y[1])(t)) = coeff( coeff(ynp1_tay[1], h, 3),
(D@@3)(y[1])(t)):
fsubs := {
f[1](y[1](t),y[2](t)) = x[1],
f[2](y[1](t),y[2](t)) = x[2],
D[1](f[1])(y[1](t),y[2](t)) = x[3],
D[1](f[2])(y[1](t),y[2](t)) = x[4],
```

```
D[2](f[1])(y[1](t),y[2](t)) = x[5],
D[2](f[2])(y[1](t),y[2](t)) = x[6],
D[1,1](f[1])(y[1](t),y[2](t)) = x[7],
D[1,1](f[2])(y[1](t),y[2](t)) = x[8],
D[1,2](f[1])(y[1](t),y[2](t)) = x[9],
D[1,2](f[2])(y[1](t),y[2](t)) = x[10],
D[2,2](f[1])(y[1](t),y[2](t)) = x[11],
D[2,2](f[2])(y[1](t),y[2](t)) = x[12],
D[1,1,1](f[1])(y[1](t),y[2](t)) = x[13],
D[1,1,1](f[2])(y[1](t),y[2](t)) = x[14],
D[1,1,2](f[1])(y[1](t),y[2](t)) = x[15],
D[1,1,2](f[2])(y[1](t),y[2](t)) = x[16],
D[1,2,2](f[1])(y[1](t),y[2](t)) = x[17],
D[1,2,2](f[2])(y[1](t),y[2](t)) = x[18],
D[2,2,2](f[1])(y[1](t),y[2](t)) = x[19],
D[2,2,2](f[2])(y[1](t),y[2](t)) = x[20]}:
psubs :={c0=1, cm0=2, c1=3, cm1=4, c2=5, a1=6}:
qsubs :={c0=100, cm0=99, c1=98, cm1=97, c2=96, a1=95}:
rsubs :={c0=7, cm0=24, c1=947, cm1=14, c2=85, a1=13}:
subs( psubs, subs(fsubs, ynp1[1]) ):
nops(expand(%)):
subs( qsubs, subs(fsubs, ynp1[1]) ):
nops(expand(%)):
subs( rsubs, subs(fsubs, ynp1[1]) ):
nops(expand(%)):
h4term0[1] := coeff(ynp1[1], h, 4):
( D[1,1,1](f[1])(y[1](t),y[2](t)) = solve(y4p[1],
D[1,1,1](f[1])(y[1](t),y[2](t))) ) * f[1](y[1](t),y[2](t))^3:
h4term[1] := applyrule(%, h4term0[1]):
h4term[1] := collect(expand(h4term[1]), (D@@4)(y[1])(t)):
h4termx[1] := subs(fsubs, h4term[1]):
collect( subs(psubs, h4termx[1]), [(D@@4)(y[1])(t), x[1], x[2], x[3],
 x[4], x[5]]):
eqn4_1 := coeff(h4termx[1], (D@@4)(y[1])(t)) = coeff (coeff(ynp1_tay[1], h, 4),
(D@@4)(y[1])(t)):
eqn4_2 := coeff( coeff( coeff(h4termx[1], x[1], 2), x[9]), x[4]) = 0:
eqn4_3 := coeff( coeff( coeff(h4termx[1], x[1], 2), x[7]), x[3]) = 0:
eqn4_4 := coeff( coeff( coeff(h4termx[1], x[1], 2), x[5]), x[8]) = 0:
eqn4_5 := coeff( coeff( coeff( coeff(h4termx[1], x[1]), x[2]), x[9]),
 x[3]) = 0:
eqn4_6 := coeff( coeff( coeff( coeff(h4termx[1], x[1]), x[2]), x[11]),
 x[4]) = 0:
eqn4_7 := coeff( coeff( coeff( coeff(h4termx[1], x[1]), x[2]), x[5]),
 x[7]) = 0:
eqn4_8 := coeff( coeff( coeff( coeff(h4termx[1], x[1]), x[2]), x[5]),
 x[10]) = 0:
```

```
eqn4_9 := coeff( coeff( coeff( coeff(h4termx[1], x[1]), x[2]), x[9]),
 x[6]) = 0:
eqn4_10 := coeff( coeff( coeff( coeff(h4termx[1], x[1]), x[5]), x[6]),
 x[4]) = 0:
eqn4_11 := coeff( coeff( coeff( coeff(h4termx[1], x[1]), x[3]), x[5]),
 x[4]) = 0:
eqn4_12 := coeff( coeff(h4termx[1], x[1]), x[3], 3) = 0:
eqn4_13 := coeff( coeff( coeff(h4termx[1], x[2], 2), x[3]), x[11]) = 0:
eqn4_14 := coeff( coeff( coeff(h4termx[1], x[2], 2), x[9]), x[5]) = 0:
eqn4_15 := coeff( coeff( coeff(h4termx[1], x[2], 2), x[12]), x[5]) = 0:
eqn4_16 := coeff( coeff( coeff(h4termx[1], x[2], 2), x[11]), x[6]) = 0:
eqn4_17 := coeff( coeff( coeff(h4termx[1], x[2]), x[4]), x[5], 2) = 0:
eqn4_18 := coeff( coeff( coeff(h4termx[1], x[2]), x[3], 2), x[5]) = 0:
eqn4_19 := coeff( coeff( coeff(h4termx[1], x[2]), x[5]), x[6], 2) = 0:
eqn4_20 := coeff( coeff( coeff( coeff(h4termx[1], x[2]), x[3]),
 x[5]), x[6]) = 0:
eqn0s := eqn0;
eqn1s := eqn1;
eqn2s := eqn2;
eqn3s := eqn3 + 1/2*eqn2;
eqn4_1s := 2*( eqn4_1 - 1/6*eqn2s + 1/2*eqn3s );
eqn4_2s := eqn4_2 + 1/2*eqn4_1s;
```

Acknowledgment

The authors would like to thank an anonymous reviewer for directing them to [12].

References

[1] C. Runge, "Ueber die numerische auflösung von differentialgleichungen," *Mathematische Annalen*, vol. 46, no. 2, pp. 167–178, 1895.

[2] W. Kutta, "Beitrag zur näherungsweisen integration totaler differentialgleichungen," *Zeitschrift für Mathematische Physik*, vol. 46, pp. 435–453, 1901.

[3] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*, John Wiley & Sons, Chichester, UK, 2003.

[4] G. D. Byrne and R. J. Lambert, "Pseudo-Runge-Kutta methods involving two points," *Journal of the Association for Computing Machinery*, vol. 13, no. 1, pp. 114–123, 1966.

[5] G. D. Byrne, "Parameters for pseudo Runge-Kutta methods," *Communications of the ACM*, vol. 10, no. 2, pp. 102–104, 1967.

[6] W. B. Gruttke, "Pseudo-Runge-Kutta methods of the fifth order," *Journal of the Association for Computing Machinery*, vol. 17, no. 4, pp. 613–628, 1970.

[7] F. Costabile, "Metodi pseudo Runge-Kutta di seconda specie," *Calcolo*, vol. 7, no. 3-4, pp. 305–322, 1970.

[8] Z. Jackiewicz, R. Renaut, and A. Feldstein, "Two-step Runge-Kutta methods," *SIAM Journal on Numerical Analysis*, vol. 28, no. 4, pp. 1165–1182, 1991.

[9] Z. Jackiewicz and S. Tracogna, "A general class of two-step Runge-Kutta methods for ordinary differential equations," *SIAM Journal on Numerical Analysis*, vol. 32, no. 5, pp. 1390–1427, 1995.

[10] X. Wu, "A class of Runge-Kutta formulae of order three and four with reduced evaluations of function," *Applied Mathematics and Computation*, vol. 146, no. 2-3, pp. 417–432, 2003.

[11] P. Phohomsiri and F. E. Udwadia, "Acceleration of Runge-Kutta integration schemes," *Discrete Dynamics in Nature and Society*, vol. 2004, no. 2, pp. 307–314, 2004.
[12] S. Tracogna and B. Welfert, "Two-step Runge-Kutta: theory and practice," *BIT Numerical Mathematics*, vol. 40, no. 4, pp. 775–799, 2000.
[13] J. R. Dormand and P. J. Prince, "A family of embedded Runge-Kutta formulae," *Journal of Computational and Applied Mathematics*, vol. 6, no. 1, pp. 19–26, 1980.
[14] T. E. Hull, W. H. Enright, B. M. Fellen, and A. E. Sedgwick, "Comparing numerical methods for ordinary differential equations," *SIAM Journal on Numerical Analysis*, vol. 9, no. 4, pp. 603–637, 1972.
[15] J. C. Butcher, "On Runge-Kutta processes of high order," *Journal of the Australian Mathematical Society*, vol. 4, pp. 179–194, 1964.
[16] The PARI Group, Bordeaux, France, version 2.3.2, 2007, http://pari.math.u-bordeaux.fr.
[17] L. F. Shampine, *Numerical Solution of Ordinary Differential Equations*, Chapman & Hall, New York, NY, USA, 1994.