Commun. Math. Sci.
Vol. 9, No. 3, pp. 879–902

SEMI-IMPLICIT INTEGRAL DEFERRED CORRECTION CONSTRUCTED WITH ADDITIVE RUNGE-KUTTA METHODS[*]

ANDREW CHRISTLIEB[†], MAUREEN MORTON[‡], BENJAMIN ONG[§], AND JING-MEI QIU[¶]

Abstract. In this paper, we construct high order semi-implicit integrators using integral deferred correction (IDC) to solve stiff initial value problems. The general framework for the construction of these semi-implicit methods uses uniformly distributed nodes and additive Runge-Kutta (ARK) integrators as base schemes inside an IDC framework, which we refer to as IDC-ARK methods. We establish under mild assumptions that, when an r^{th} order ARK method is used to predict and correct the numerical solution, the order of accuracy of the IDC method increases by r for each IDC prediction and correction loop. Numerical experiments support the established theorems, and also indicate that higher order IDC-ARK methods present an efficiency advantage over existing implicit-explicit (IMEX) ARK schemes in some cases.

Key words. Defect correction methods, additive Runge-Kutta methods, semi-implicit methods, integral deferred correction methods, spectral deferred correction methods, implicit-explicit methods.

AMS subject classifications. 65B05, 65L05, 65L20.

1. Introduction

Many physical problems described by time dependent partial differential equations (PDEs) or ordinary differential equations (ODEs) involve multiple time scales that necessitate efficient and accurate numerical time integration. For example, chemical rate equations can have a dynamic range of behaviors which span disparate time scales. Such equations can be expressed in a simplified model as

$$\begin{aligned} y'(t) &= f_S(t,y) + f_N(t,y), \quad t \in [0,T], \\ y(0) &= y_0, \end{aligned} \tag{1.1}$$

where $f_S(t,y)$ contains stiff terms and $f_N(t,y)$ contains the nonstiff terms. The above system may also arise with a method of lines discretization of PDEs, such as hyperbolic systems with relaxtion terms, convection-diffusion systems, and convection-diffusion-reaction systems. Applying explicit numerical integrators to the above initial value problem (IVP) can result in impractical time step restrictions due to the stiff term, while fully implicit methods may involve costly Newton iterations.

A general approach to solving system (1.1) is to construct implicit-explicit (IMEX) methods, where the stiff term is handled implicitly, and the non-stiff term is handled explicitly, thereby gaining the benefit of the implicit method for the stiffness but potentially saving computational effort by handling some terms explicitly. In this paper, we construct a class of IMEX-RK methods, also known as additive Runge–Kutta (ARK) methods, using the defect correction framework. Currently, ARK methods above 5th order have not been constructed without the use of a defect correction

[*] Received: August 19, 2010; accepted (in revised version): January 25, 2011. Communicated by Lorenzo Pareschi.

[†] Department of Mathematics, Michigan State University, East Lansing, MI 48824, USA (andrewch@math.msu.edu).

[‡] Department of Mathematics, Michigan State University, East Lansing, MI 48824, USA (mortonm5@msu.edu).

[§] Department of Mathematics, Michigan State University, East Lansing, MI 48824, USA (bwo@math.msu.edu).

[¶] Department of Mathematics and Computer Sciences, Colorado School of Mines, Golden, CO 80401, USA (jingqiu@mines.edu).

framework [21, 22]. It is also possible to contruct IMEX methods using general linear multi-step schemes. Such methods however require multiple starting values, and their stabilities deteriorate at higher order [16, 3, 23].

In 2000, Dutt, Greengard, and Rokhlin introduced spectral deferred correction (SDC) methods [12]. SDC methods compute a provisional solution on a subgrid and then correct this provisional solution by approximating the error in subsequent correction loops over the subgrid. A unique feature of SDC is the formulation of the error equation in an integral form within the correction loops, which provides improved stability [12, 13]. Several investigations and modifications of SDC followed. For example, the order of accuracy for SDC methods constructed with Euler integrators was proved [28]. Krylov deferred correction, which uses SDC with an Euler integrator as a preconditioner for a Newton-Krylov method, was developed to handle differential algebraic equations [17, 18]. In [9], a variant of SDC, integral deferred correction (IDC), constructed using uniform nodes and high order explicit RK integrators in both the prediction and corrections, referred to as IDC-RK, was introduced. In [9], it was established that using an explicit RK method of order r in the correction results in r more degrees of accuracy with each successive correction. In [8] it was demonstrated that IDC-RK methods are more efficient than SDC methods (constructed with Euler integrators) of equivalent order. IDC-RK methods were also found to possess better stability properties than SDC [9] and can be written as RK methods in Butcher table form [8]. Semi-implicit IDC with backward and forward Euler was developed as an IMEX method to handle IVPs of the form (1.1) [25], while in [22], semi-implicit IDC methods using second order base schemes (BDF and RK) in both the prediction and correction loops were implemented.

In this paper, we provide a general framework for the construction of arbitrary order IMEX methods that use ARK integrators as base schemes inside the IDC framework. We will refer to this family as IDC-ARK methods. Our framework recovers the formulation in [22] for the second order ARK case. We do not study multistep methods embedded in the prediction and correction loops of IDC because multistep methods in general do not achieve high order accuracy increase after each correction since the error is not smooth [8]. In Section 3, we analyze the local truncation error of the IDC-ARK methods. It is found that, when an r^{th} order ARK method is used to predict and correct the numerical solution, the order of accuracy of the IDC method increases by r for each IDC prediction and correction loop. A corollary of the theorem provides the truncation error results for IDC methods constructed using implicit RK integrators in the prediction and correction loops. The analysis of the truncation error closely follows the analysis in [9], and, since only the essential differences are presented in this work, it is recommended that the interested reader first reads [9] before reading Section 3. Numerical results in Section 5 support the conclusion of the theorem. As experienced by many IMEX schemes, including ARK, some order reduction is apparent with increased stiffness. In future work, we plan to explore using asymptotic preserving ARK schemes (as in [27]) in the prediction and correction loops of IDC to potentially mitigate the issue of order reduction. In some scenarios, high order IDC-ARK presents an efficiency advantage over some existing IMEX ARK schemes.

The paper is organized as follows. An IMEX form of Runge-Kutta methods known as ARK is explained in Section 2.1. Our formulation of semi-implicit IDC-ARK schemes, based on [25] and [9], can be found in Section 2.2. Section 3 contains the analysis of the truncation error. Improved stability of the IDC-ARK over SDC-

FE/BE is shown in Section 4, and Section 5 summarizes the numerical experiments conducted, including order of accuracy and efficiency.

2. Formulation of IDC-ARK

In this section, we first present a brief description of ARK methods and their formulation within the framework of IDC. Then we provide both a general formulation that constructs arbitrary order semi-implicit IDC methods incorporating (arbitrary order) ARK integrators and an example involving a second order ARK integrator.

2.1. Additive Runge-Kutta. One method of splitting an ODE is to partition the right hand side of an IVP into Λ parts [10, 1, 21, 24]:

$$y'(t) = f(t,y) = \sum_{\nu=1}^{\Lambda} f_{[\nu]}(t,y), \quad y(0) = y_0, \quad t \in [0,T], \tag{2.1}$$

and to treat each $f_{[\nu]}$ with a separate numerical method. When different p-stage Runge-Kutta integrators are applied to each $f_{[\nu]}$, the entire numerical method is called an additive Runge-Kutta (ARK) method. If we define the numerical solution after one timestep h as η_1, which is an approximation to the exact solution $y(t_0+h)$, then one step of a p-stage ARK method is given by

$$\begin{aligned} \eta_1 &= y_0 + h \sum_{\nu=1}^{\Lambda} \sum_{i=1}^{p} b_i^{[\nu]} f_{[\nu]}(t_0 + c_i^{[\nu]} h, Y_i), \qquad (2.2)\\ \text{where} \quad Y_i &= y_0 + h \sum_{\nu=1}^{\Lambda} \sum_{j=1}^{p} a_{ij}^{[\nu]} f_{[\nu]}(t_0 + c_j^{[\nu]} h, Y_j). \end{aligned}$$

For the case $c_j^{[\nu]} = c_j$ for all j, ν, the ARK method has the Butcher table:

$$\begin{array}{c|cccc|c|cccc}
c_1 & a_{11}^{[1]} & a_{12}^{[1]} & \cdots & a_{1p}^{[1]} & & a_{11}^{[\Lambda]} & a_{12}^{[\Lambda]} & \ldots & a_{1p}^{[\Lambda]} \\
c_2 & a_{21}^{[1]} & a_{22}^{[1]} & \cdots & a_{2p}^{[1]} & & a_{21}^{[\Lambda]} & a_{22}^{[\Lambda]} & \ldots & a_{2p}^{[\Lambda]} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \cdots & \vdots & \vdots & \ddots & \vdots \\
c_p & a_{p1}^{[1]} & a_{p2}^{[1]} & \cdots & a_{pp}^{[1]} & & a_{p1}^{[\Lambda]} & a_{p2}^{[\Lambda]} & \cdots & a_{pp}^{[\Lambda]} \\
\hline
 & b_1^{[1]} & b_2^{[1]} & \cdots & b_p^{[1]} & \ldots & b_1^{[\Lambda]} & b_2^{[\Lambda]} & \ldots & b_p^{[\Lambda]}
\end{array}$$

Under certain conditions on the RK coefficients, an ARK method of desired order can be obtained without splitting error [11, 1, 24, 21].

Without loss of generality, we consider the case $\Lambda = 2$, where the IVP (2.1) simplifies to the IVP (1.1), where $f_S(t,y)$ is stiff and $f_N(t,y)$ is nonstiff. An IMEX version of ARK is then applied to (1.1), as shown for example in the Butcher table [21],

$$\begin{array}{c|cccc|ccccc}
0 & 0 & & & & 0 & & & & \\
c_2 & a_{21}^S & a_{22}^S & 0 & \ldots & a_{21}^N & 0 & & & \\
\vdots & \vdots & \vdots & \ddots & & \vdots & & & \ddots & \\
c_p & a_{p1}^S & a_{p2}^S & \cdots & a_{pp}^S & a_{p1}^N & a_{p2}^N & \cdots & a_{p,p-1}^N & 0 \\
\hline
 & b_1^S & b_2^S & \cdots & b_p^S & b_1^N & b_2^N & \cdots & b_{p-1}^N & b_p^N
\end{array}$$

where to reduce computational cost, an implicit DIRK method, with zero in the first diagonal entry, is applied to f_S, and an explicit method is applied to f_N. Note that, although one can formulate an ARK method with different stage nodes, say c_j^N and c_j^S, taking the same stage nodes, $c_j \doteq c_j^N = c_j^S$, simplifies the order conditions, as presented in [21].

In the literature, the definition of ARK for the case of $\Lambda = 2$ usually is presented in the form of (2.2) [1, 21, 10, 24]. We reformulate the definition of ARK ($\Lambda = 2$) below to suit our analysis. The equivalence of definitions can be seen by simple substitution. Our alternate definition can be given as follows.

Definition 2.1. *Let p denote the number of stages and a_{ij}^N, a_{ij}^S; b_j^N, b_j^S; c_j be real coefficients. Then the method*

$$k_i^\alpha = f_\alpha\left(t_0 + c_i h, y_0 + h\left(\sum_{j=1}^{i-1} a_{ij}^N k_j^N + \sum_{j=1}^{p} a_{ij}^S k_j^S\right)\right),$$

$$\eta_1 = y_0 + h\sum_{i=1}^{p} b_i^N k_i^N + b_i^S k_i^S, \tag{2.3}$$

for $\alpha = N, S$ and $i = 1, 2, \ldots, p$, is a p-stage ARK method for solving the IVP (1.1).

Definition 2.2. *An ARK method has order r if for a sufficiently smooth IVP* (1.1),

$$\|y(t_0 + h) - \eta_1\| \leq K h^{r+1},$$

for some constant $K > 0$; i.e., the Taylor series for the exact solution $y(t_0 + h)$ and η_1 coincide up to and including the term h^r.

2.2. General Formulation of IDC-ARK. Generalizing the formulation from [9] for IDC-ARK, we obtain a general framework for semi-implicit IDC constructed using ARK integrators.

Consider the IVP (1.1), where the right hand side is split into stiff and nonstiff parts, $f_S(t, y)$ and $f_N(t, y)$, as in Section 2.1. The time interval $[0, T]$ is discretized into intervals $[t_n, t_{n+1}]$, $n = 0, 1, \ldots, N - 1$, such that

$$0 = t_0 < t_1 < t_2 < \cdots < t_n < \cdots < t_N = T, \tag{2.4}$$

with timestep $H_n = t_{n+1} - t_n$. Each interval $[t_n, t_{n+1}]$ is discretized again into M subintervals with quadrature nodes denoted by

$$t_{n,0} = t_n, \quad t_{n,m} = t_{n,m-1} + h_{n,m}, \quad m = 1, \ldots, M, \tag{2.5}$$

where $h_{n,m}$ is a step size that may vary with m, and $\sum_{m=1}^{M} h_{n,m} = h_n$ (so $t_{n,M} = t_{n+1}$). For IDC-ARK methods, we consider only the case of *uniform quadrature nodes*; i.e., $h_{n,m} = \frac{H_n}{M}$ for $m = 1, \ldots, M$. Although it is possible to use nonuniform quadrature nodes, we anticipate that nonuniform nodes will not consistently produce the desired order of accuracy results, based on the analysis in [8]. Note that the use of the word *uniform* does not preclude an adaptive timestepping implementation of the IDC-ARK methods since, for uniform quadrature nodes, we mean that the step H_n may be varied as desired, but within each step H_n, the smaller steps $h_{n,m}$ must remain equal for each m. Since we consider only uniform quadrature nodes, we drop the subscript m on $h_{n,m}$ in our analysis. Furthermore, since the focus of this paper is the construction and order of accuracy of general IDC-ARK methods (and not adaptive timestepping with such methods), and since the analysis is identical for any interval $[t_n, t_{n+1}]$, we subsequently drop the subscript n and apply the IDC method to the interval $[0, H]$. We also note that the numerical solution at the end of some interval $[t_n, t_{n+1}]$ is used as the initial condition for the subsequent interval $[t_{n+1}, t_{n+2}]$.

A summary of the IDC-ARK algorithm is as follows. First, in the prediction loop, we compute a provisional solution to the IVP (1.1) with an ${r_0}^{\text{th}}$ order ARK method. Then for $k = 1, \dots, K_{loop}$ correction loops, we compute an approximation to the error using an ${r_k}^{\text{th}}$ order ARK method and update the provisional solution with this approximate error. Thus an IDC method with order of accuracy $\min(r_0 + \cdots + r_k + \cdots + r_{K_{loop}}, M+1)$ is obtained.

Next we provide the details that are necessary for both the implementation of the IDC-ARK method and the proof of truncation error in Section 3.2.

- **Prediction loop:** Use an ${r_0}^{\text{th}}$ order numerical method to obtain a provisional solution to the IVP (1.1),

$$\eta^{[0]} = (\eta_0^{[0]}, \eta_1^{[0]}, \dots, \eta_m^{[0]}, \dots, \eta_M^{[0]}),$$

which is an ${r_0}^{\text{th}}$ order approximation to the exact solution

$$y = (y_0, y_1, \dots, y_m, \dots, y_M),$$

where $y_m = y(t_m)$ is the exact solution at t_m, for $m = 0, 1, \dots, M$. We apply a p_0-stage ${r_0}^{\text{th}}$ order ARK scheme (2.3) as follows: for $\alpha = N, S$; $i = 1, 2, \dots, p_0$ and $m = 0, 1, \dots, M-1$,

$$k_i^\alpha = f_\alpha\left(t_m + c_i h, \eta_m^{[0]} + h\left(\sum_{j=1}^{i-1} a_{ij}^N k_j^N + \sum_{j=1}^{p_0} a_{ij}^S k_j^S\right)\right),$$

$$\eta_{m+1}^{[0]} = \eta_m^{[0]} + h\sum_{i=1}^{p_0}(b_i^N k_i^N + b_i^S k_i^S).$$

- **Correction loop:** Use the error function to improve the order of accuracy of the numerical solution at each iteration.
For $k = 1$ to K_{loop} (K_{loop} is the number of correction loops):

1. Denote the *error function* from the previous step as

$$e^{(k-1)}(t) = y(t) - \eta^{(k-1)}(t), \tag{2.6}$$

where $y(t)$ is the exact solution and $\eta^{(k-1)}(t)$ is an M^{th} degree polynomial interpolating $\eta^{[k-1]}$. Note that the error function $e^{(k-1)}(t)$ is not a polynomial in general.

2. Denote the *residual function* as

$$\epsilon^{(k-1)}(t) \doteq (\eta^{(k-1)})'(t) - f_S(t, \eta^{(k-1)}(t)) - f_N(t, \eta^{(k-1)}(t)),$$

and compute the integral of the residual. For example,

$$\int_{t_0}^{t_{m+1}} \epsilon^{(k-1)}(\tau)\,d\tau$$
$$\approx \eta_{m+1}^{[k-1]} - y_0 - \sum_{j=0}^{M} \gamma_{m,j}(f_S(t_j, \eta_j^{[k-1]}) + f_N(t_j, \eta_j^{[k-1]})), \tag{2.7}$$

where $\gamma_{m,j}$ are the coefficients that result from approximating the integral by interpolatory quadrature formulas, as in [12], and $\eta_j^{[k-1]} = \eta^{(k-1)}(t_j)$.

3. Compute the *numerical error vector*, denoted by

$$\delta^{[k]} = (\delta_0^{[k]}, \dots, \delta_m^{[k]}, \dots, \delta_M^{[k]}), \tag{2.8}$$

which is an ${r_k}^{\text{th}}$ order approximation to the error

$$e^{[k-1]} = (e_0^{[k-1]}, \dots, e_m^{[k-1]}, \dots, e_M^{[k-1]}),$$

where $e_m^{[k-1]} = e^{(k-1)}(t_m)$ is the value of the exact error function (2.6) at t_m.

To compute $\delta^{[k]}$ by an ${r_k}^{\text{th}}$ order ARK method, we first discretize the integral form of the *error equation*, which can be found by differentiating the error,

$$\begin{aligned}(e^{(k-1)})'(t) = {} & f_S(t, \eta^{(k-1)}(t) + e^{(k-1)}(t)) - f_S(t, \eta^{(k-1)}(t)) \\ & + f_N(t, \eta^{(k-1)}(t) + e^{(k-1)}(t)) - f_N(t, \eta^{(k-1)}(t)) - \epsilon^{(k-1)}(t),\end{aligned}$$

letting

$$E^{(k-1)}(t) = \int_{t_0}^{t} \epsilon^{(k-1)}(\tau)\,d\tau,$$

bringing the residual to the left hand side,

$$\begin{aligned}(e^{(k-1)}(t) + E^{(k-1)}(t))' = {} & f_S(t, \eta^{(k-1)}(t) + e^{(k-1)}(t)) - f_S(t, \eta^{(k-1)}(t)) \\ & + f_N(t, \eta^{(k-1)}(t) + e^{(k-1)}(t)) - f_N(t, \eta^{(k-1)}(t)),\end{aligned}$$

and defining a differential equation in terms of a new variable Q, given by

$$Q^{(k-1)}(t) = e^{(k-1)}(t) + E^{(k-1)}(t), \tag{2.9}$$

so that the *error equation* whose integral form we discretize is given by

$$\begin{aligned}(Q^{(k-1)})'(t) &= f_S(t, \eta^{(k-1)}(t) + Q^{(k-1)}(t) - E^{(k-1)}(t)) - f_S(t, \eta^{(k-1)}(t)) \\ &\quad + f_N(t, \eta^{(k-1)}(t) + Q^{(k-1)}(t) - E^{(k-1)}(t)) - f_N(t, \eta^{(k-1)}(t)) \\ &\doteq F_S(t, Q^{(k-1)}(t) - E^{(k-1)}(t)) + F_N(t, Q^{(k-1)}(t) - E^{(k-1)}(t)) \\ &\doteq G_S(t, Q^{(k-1)}(t)) + G_N(t, Q^{(k-1)}(t)), \\ Q^{(k-1)}(0) &= 0, \end{aligned} \tag{2.10}$$

where

$$\begin{aligned} F_\alpha(t, e^{(k-1)}(t)) &= f_\alpha(t, \eta^{(k-1)}(t) + e^{(k-1)}(t)) - f_\alpha(t, \eta^{(k-1)}(t)), \\ G_\alpha(t, Q^{(k-1)}(t)) &= F_\alpha(t, e^{(k-1)}(t)), \quad \alpha = N, S. \end{aligned}$$

The f_α, $\alpha = N, S$ notation above (with Q and the approximation for E) is used in the numerical computation, while the G_α, $\alpha = N, S$ notation is employed in the theoretical explanations and proof in Section 3.2.
We apply a p_k-stage ${r_k}^{\text{th}}$ order ARK method to (2.10) and denote the ${r_k}^{\text{th}}$ order numerical approximation to $Q_m^{[k-1]} = Q^{(k-1)}(t_m)$ by $\Omega_m^{[k]}$. We therefore have, for $\alpha = N, S$, $i = 1, 2, \ldots, p_k$ and $m = 0, 1, \ldots, M-1$,

$$\begin{aligned} k_i^\alpha &= G_\alpha\left(t_m + c_i h, \Omega_m^{[k]} + h\left(\sum_{j=1}^{i-1} a_{ij}^N k_j^N + \sum_{j=1}^{p_k} a_{ij}^S k_j^S\right)\right) \\ &= F_\alpha(t_m + c_i h, \Omega_m^{[k]} + h\left(\sum_{j=1}^{i-1} a_{ij}^N k_j^N + \sum_{j=1}^{p_k} a_{ij}^S k_j^S\right) \qquad (2.11) \\ &\quad - E^{[k-1]}(t_m + c_j h)), \\ \Omega_{m+1}^{[k]} &= \Omega_m^{[k]} + h\sum_{i=1}^{p_k}(b_i^N k_i^N + b_i^S k_i^S). \qquad (2.12) \end{aligned}$$

From Equations (2.9), we compute our numerical error vector (2.8),

$$\delta^{[k]} = \Omega^{[k]} - E^{[k-1]},$$

where the components of $E^{[k-1]}$ are $E_m^{[k-1]} = E^{(k-1)}(t_m)$. In fact, we use

$$\begin{aligned} \delta_{m+1}^{[k]} &= \Omega_{m+1}^{[k]} - \int_{t_0}^{t_{m+1}} \epsilon^{(k-1)}(\tau)\, d\tau \\ &\overset{(2.7)}{\approx} \Omega_{m+1}^{[k]} - (\eta_{m+1}^{[k-1]} - y_0 - \sum_{j=0}^{M} \gamma_{m,j}(f_S(t_j, \eta_j^{[k-1]}) + f_N(t_j, \eta_j^{[k-1]}))), \end{aligned}$$

where we have approximated the integral by interpolatory quadrature formulas, as in [12].
In computing (2.12), one needs to evaluate $f_\alpha(t, \eta(t))$, $\alpha = N, S$, at some intermediate stage, e.g., at $t = t_0 + c_j h$. This results in additional function evaluations of $f_\alpha(t, \eta(t))$, which is usually the most expensive part of an algorithm. In our implementation, we avoid this extra cost by performing a polynomial interpolation. Specifically, given $f_\alpha = (f_\alpha(t_0, \eta_0), \ldots, f_\alpha(t_m, \eta_m), \ldots, f_\alpha(t_M, \eta_M))$, we construct an M^{th} degree Lagrange interpolant, $L^M(t, f_\alpha)$ and approximate $f_\alpha(t, \eta(t))|_{t=t_0+c_j h}$ using $L^M(t = t_0 + c_j h, f_\alpha)$ up to $\mathcal{O}(h^{M+1})$. Again note that f_α, F_α are more useful for understanding the numerical implementation, while G_α is employed in the theory in Section 3.2.

4. Update the numerical solution $\eta^{[k]} = \eta^{[k-1]} + \delta^{[k]}$.

2.3. Example with second order ARK. Now we present an example where a second order ARK scheme is employed in the IDC framework. We consider the IMEX RK2 scheme from [2], and illustrate that our formulation reduces to Layton's formulation [22] for this specific IMEX RK2 scheme. Writing the ARK scheme (we denote it by ARK2ARS) in a Butcher table, we have

$$\begin{array}{c|ccc|ccc}
0 & 0 & & & & & \\
g & 0 & g & & g & & \\
1 & 0 & 1-g & g & d & 1-d & \\
\hline
 & 0 & 1-g & g & 0 & 1-g & g
\end{array}$$

where $g = 1 - \frac{\sqrt{2}}{2}$ and $d = -\frac{2\sqrt{2}}{3}$.

The prediction loop solving (1.1), for $\alpha = N, S$ and $m = 0, 1, \ldots, M-1$, is

$$\begin{aligned}
\eta_{m+1}^{[0]} &= \eta_m^{[0]} + h((1-g)(k_1^N + k_2^S) + g(k_3^N + k_3^S)), \text{ where} \\
k_1^\alpha &= f_\alpha(t_m, \eta_m^{[0]}), \\
k_2^\alpha &= f_\alpha(t_m + gh, \eta_m^{[0]} + hg(k_1^N + k_2^S)), \\
k_3^\alpha &= f_\alpha(t_{m+1}, \eta_m^{[0]} + h(dk_2^N + (1-d)k_2^N + (1-g)k_2^S + gk_3^S)),
\end{aligned}$$

The correction loop solving (2.10) is

$$\begin{aligned}
\delta_{m+1}^{[k]} &= \Omega_{m+1}^{[k]} - E^{[k-1]}(t_{m+1}), \text{ where} \\
\Omega_{m+1}^{[k]} &= \Omega_m^{[k]} + h((1-g)(k_2^N + k_2^S) + g(k_3^N + k_3^S)), \\
k_1^\alpha &= F_\alpha(t_m, \Omega_m^{[k]} - E^{[k-1]}(t_m)), \\
k_2^\alpha &= F_\alpha(t_m + gh, \Omega_m^{[k]} + hg(k_1^N + k_2^S) - E^{[k-1]}(t_m + gh)), \\
k_3^\alpha &= F_\alpha(t_{m+1}, \Omega_m^{[k]} + h(dk_1^N + (1-d)k_2^N + (1-g)k_2^S + gk_3^S) - E^{[k-1]}(t_{m+1})).
\end{aligned}$$

Letting $\eta_m^{[k]} = \delta_m^{[k]} + \eta_m^{[k-1]}$ and $\Omega_m^{[k]} = \delta_m^{[k]} + E_m^{[k-1]}$, and noting $\mathcal{Q}_m^{m+c}(f_S(t, \eta^{(k-1)}(t)) + f_N(t, \eta^{(k-1)}(t)))$ from [22] is an approximation of $\int_{t_m}^{t_m+ch} \epsilon^{(k-1)}(\tau)d\tau$, we see that our framework recovers the formulation of IDC constructed with ARK2ARS in [22].

$$\begin{aligned}
\phi_{m+g}^{(1)} &= \eta_{m+g}^{[k-1]} + \Omega_m^{[k]} + hg(k_1^N + k_2^S) - E^{[k-1]}(t_m + gh), \\
\phi_{m+1}^{(2)} &= \eta_{m+1}^{[k-1]} + \Omega_m^{[k]} + h(dk_1^N + (1-d)k_2^N + (1-g)k_2^S + gk_3^S) - E^{[k-1]}(t_{m+1}), \\
\eta_{m+1}^{[k]} &= \eta_{m+1}^{[k-1]} + \delta_{m+1}^{[k]},
\end{aligned}$$

where the notation on the left is as in [22], while the notation on the right is ours.

3. Theoretical analysis of IDC-ARK

Some mathematical preliminaries related to the smoothness of discrete data sets are introduced in Section 3.1. Section 3.2 then provides the local truncation error estimates of IDC-ARK along with a corollary for the local truncation error of IDC constructed with implicit RK integrators. This paper closely follows the formulation presented in [9]. Since much of the framework for analyzing the local truncation error remains unchanged, we only present the essential part of the analysis where it substantially differs from the construction in [9].

3.1. Mathematical preliminaries.

Several analytical and numerical preliminaries are needed to analyze IDC methods. The smoothness of discrete data sets will be established, analogously to the smoothness of functions; this idea of smoothness is used to analyze the error vectors. Let $\psi(t)$ be a function for $t \in [0,T]$, and without loss of generality, consider the first interval, $[0,H]$, of the grid (2.4). Denote the corresponding discrete data set as

$$(\vec{t},\vec{\psi}) = \{(t_0,\psi_0),\dots,(t_M,\psi_M)\}, \tag{3.1}$$

where t_m are the uniform quadrature nodes given by (2.5).

Definition 3.1 (smoothness of a function). *A function $\psi(t)$, $t \in [0,T]$, possesses σ degrees of smoothness if $\|d^s\psi\|_\infty := \left\|\frac{\partial^s}{\partial t^s}\psi\right\|_\infty$ is bounded for $s = 0,1,2,\dots,\sigma$, where $\|\psi\|_\infty := \max_{t\in[0,T]}|\psi(t)|$.*

Definition 3.2 (discrete differentiation). *Consider the discrete data set, $(\vec{t},\vec{\psi})$, defined in (3.1), and denote L^M as the usual the Lagrange interpolant, an M^{th} degree polynomial that interpolates (t,ψ).*

$$L^M(t,\psi) = \sum_{m=0}^{M} c_m(t)\psi_m, \quad \textit{where} \quad c_m(t) = \prod_{n\neq m}\frac{t-t_n}{t_m-t_n}.$$

An s^{th} degree discrete differentiation is a linear mapping that maps $\vec{\psi}$ into $\overrightarrow{\hat{d}_s\psi}$, where $(\hat{d}_s\psi)_m = \frac{\partial^s}{\partial t^s}L^M(t,\psi)|_{t=t_m}$. This linear mapping can be represented by a matrix multiplication $\overrightarrow{\hat{d}_s\psi} = \hat{D}_s\cdot\vec{\psi}$, where $\hat{D}_s \in \mathcal{R}^{(M+1)\times(M+1)}$ and $(\hat{D}_s)_{mn} = \frac{\partial^s}{\partial t^s}c_n(t)|_{t=t_m}$, $m,n = 0,\dots,M$.

Given a distribution of quadrature nodes on $[0,1]$, the differentiation matrices, $\hat{D}_s^{[0,1]}$, $s = 1,...,M$ have constant entries. If this distribution of quadrature nodes is rescaled from $[0,1]$ to $[0,H]$, then the corresponding differentiation matrices are $\hat{D}_1 = \frac{1}{H}\hat{D}_1^{[0,1]}$ and $\hat{D}_s = \left(\frac{1}{H}\right)^s\hat{D}_s^{[0,1]}$.

Definition 3.3. *The $(\hat{S},\infty)$ Sobolev norm of the discrete data set $(\vec{t},\vec{\psi})$ is defined as*

$$\left\|\vec{\psi}\right\|_{\hat{S},\infty} := \sum_{s=0}^{\sigma}\left\|\overrightarrow{\hat{d}_s\psi}\right\|_\infty = \sum_{s=0}^{\sigma}\left\|\hat{D}_s\cdot\vec{\psi}\right\|_\infty,$$

where $\overrightarrow{\hat{d}_s\psi} = Id\cdot\vec{\psi}$ is the identity matrix operating on $\vec{\psi}$.

Definition 3.4 (smoothness of a discrete date set). *A discrete data set,* (3.1), *possesses σ ($\sigma \le M$) degrees of smoothness if $\|\vec{\psi}\|_{\hat{S},\infty}$ is bounded as $h \to 0$.*

We emphasize that smoothness is a property of discrete data sets in the limit as $h \to 0$. We also impose $\sigma \le M$, because $\overrightarrow{\hat{d}_\sigma\psi} \equiv \vec{0}$, for $\sigma > M$. See [9] for a detailed discussion.

Example 3.1 (A discrete data set with only one degree of smoothness). *Consider the discrete data set*

$$(\vec{t},\vec{\psi}) = \left\{(0,0),\left(\frac{H}{4},\frac{H}{4}\right),\left(\frac{H}{2},\frac{H}{2}\right),\left(\frac{3H}{4},\frac{H}{4}\right),(H,0)\right\}.$$

The first derivative

$$\overrightarrow{\hat{d_1}\psi} = \left(-\frac{4}{3}, \frac{10}{3}, 0, -\frac{10}{3}, \frac{4}{3}\right),$$

is bounded independent of H*, while the second derivative*

$$\overrightarrow{\hat{d_2}\psi} = \left(\frac{272}{3H}, -\frac{16}{3H}, -\frac{112}{3H}, -\frac{16}{3H}, \frac{272}{3H}\right),$$

is unbounded as $H \to 0$*. Therefore,* $(\vec{t}, \vec{\psi})$ *has one, and only one degree of smoothness in the discrete sense.*

For further details, definitions, and properties of smoothness in the discrete and continuous sense, we refer the reader to [9].

3.2. Truncation error analysis of IDC-ARK. Here we provide the local truncation error estimates for IDC-ARK methods. The analysis easily extends to implicit-implicit or explicit-explicit IDC-ARK methods, as well as IDC methods constructed with ARK methods of the form (2.2), designed for differential equations whose right hand side can be partitioned into more than two parts. Since most IDC-ARK theoretical results are similar to those in [9], we only present the main theorems and lemmas, along with some helpful notations, and highlight the portions of the proof that differ from [9]. The implicit case is presented in Corollary 3.10.

Theorem 3.5. *Let* $y(t)$*, the solution to the IVP* (1.1)*, have at least* σ *(and* f_S *and* f_N *have at least* $\sigma - 1$*) degrees of smoothness in the continuous sense, where* $\sigma \geq M+2$*. Consider one time interval of an IDC method with* $t \in [0, H]$ *and* $M+1$ *uniformly distributed quadrature points (2.5). Suppose an* ${r_0}^{th}$ *order ARK method* (2.3) *is used in the prediction step and* $(r_1, r_2, \ldots, r_{K_{loop}})^{th}$ *order ARK methods are used in* K_{loop} *correction steps. For simplicity, assume that* $c_j \doteq c_j^N = c_j^S$ *and the number of stages for the implicit and explicit parts of each ARK method is the same. Let* $s_k = \sum_{j=0}^{k} r_j$*. If* $s_{K_{loop}} \leq M+1$*, then the local truncation error is of order* $\mathcal{O}(h^{s_{K_{loop}}+1})$*.*

The proof of Theorem 3.5 follows by induction from Lemmas 3.6 and 3.7, below, for the prediction and correction steps, respectively, which discuss the local truncation error and smoothness of the rescaled error when general high order ARK schemes are applied in the prediction and correction loops of IDC methods. Lemma 3.6 is the first case, and Lemma 3.7 is the induction step.

Lemma 3.6 (Prediction step). *Let* $y(t)$*,* f_S*, and* f_N *satisfy the smoothness requirements in Theorem 3.5, and let* $\eta^{[0]} = (\eta_0^{[0]}, \eta_1^{[0]}, \ldots, \eta_m^{[0]}, \ldots, \eta_M^{[0]})$ *be the numerical solution on the uniformly distributed quadrature points* (2.5)*, obtained using an* ${r_0}^{th}$ *order ARK method (as described in Theorem 3.5) at the prediction step. Then:*

1. *The error vector* $e^{[0]} = y - \eta^{[0]}$ *satisfies* $\|e^{[0]}\|_\infty \sim \mathcal{O}(h^{r_0+1})$*.*
2. *The rescaled error vector* $\widetilde{e}^{[0]} = \frac{1}{h^{r_0}} e^{[0]}$ *has* $\min(\sigma - r_0, M)$ *degrees of smoothness in the discrete sense.*

Lemma 3.7 (Correction step). *Let* $y(t)$*,* f_S*, and* f_N *satisfy the smoothness requirements in Theorem 3.5. Suppose* $e^{[k-1]} \sim \mathcal{O}(h^{s_{k-1}+1})$ *and* $\widetilde{e}^{[k-1]} = \frac{1}{h^{s_{k-1}}} e^{[k-1]}$ *has* $M + 1 - s_{k-1}$ *degrees of smoothness in the discrete sense after the* $(k-1)$*st correction step. Then, after the* k^{th} *correction step, provided an* ${r_k}^{th}$ *order ARK method (as described in Theorem 3.5) is used and* $k \leq K_{loop}$*,*

1. *The error vector* $e^{[k]}$ *satisfies* $\|e^{[k]}\|_\infty \sim \mathcal{O}(h^{s_k+1})$.
2. *The rescaled error vector* $\widetilde{e}^{[k]} = \frac{1}{h^{s_k}} e^{[k]}$ *has* $M + 1 - s_k$ *degrees of smoothness in the discrete sense.*

Note that the smoothness results of the rescaled error vectors in both lemmas are essential for the proof of the correction loop in Lemma 3.7. The smoothness proofs of Lemmas 3.6 and 3.7 (analogous to those in [9]) are quite technical. We instead state and sketch a proof for the smoothness of the rescaled error vector following a forward-backward Euler (FBE) prediction step in Lemma 3.8. The smoothness proof for the more general case is similar in spirit. Once the smoothness of the rescaled error vector is proved, the magnitude of the error vector follows directly from Definition 2.2.

Lemma 3.8 (FBE prediction step). *Consider an IDC method constructed using* $M + 1$ *uniformly distributed nodes, and a FBE integrator for the prediction step. Let* $y(t)$, *the solution to the IVP* (1.1), *have at least* $\sigma \geq M + 2$ *degrees of smoothness, and let* $\vec{\eta}^{[0]} = (\eta_0^{[0]}, \ldots, \eta_m^{[0]}, \ldots, \eta_M^{[0]})$ *be the numerical solution computed after the prediction step. Then the rescaled error vector,* $\vec{\tilde{e}}^{[0]} = \frac{1}{h}\vec{e}^{[0]}$, *has* M *degrees of smoothness in the discrete sense.*

Proof. We drop the superscript [0] as there is no ambiguity. A FBE discretization gives

$$\eta_{m+1} = \eta_m + hf_N(t_m, \eta_m) + hf_S(t_{m+1}, \eta_{m+1}), \tag{3.2}$$

and the exact solution satisfies

$$\begin{aligned} y_{m+1} &= y_m + \int_{t_m}^{t_{m+1}} f_N(\tau, y(\tau))d\tau + \int_{t_m}^{t_{m+1}} f_S(\tau, y(\tau))d\tau, \\ &= y_m + hf_N(t_m, y_m) + \sum_{i=1}^{\sigma-2} \frac{h^{i+1}}{(i+1)!} \frac{d^i f_N}{dt^i}(t_m, y_m) \\ &\quad + hf_S(t_{m+1}, y_{m+1}) + \sum_{i=1}^{\sigma-2} \frac{h^{i+1}}{(i+1)!} \frac{d^i f_S}{dt^i}(t_{m+1}, y_{m+1}) + \mathcal{O}(h^\sigma). \end{aligned} \tag{3.3}$$

Subtracting Equation (3.2) from Equation (3.3) gives

$$\begin{aligned} e_{m+1} = e_m &+ h(f_N(t_m, y_m) - f_N(t_m, \eta_m)) + h(f_S(t_{m+1}, y_{m+1}) - f_S(t_{m+1}, \eta_{m+1})) \\ &+ \sum_{i=1}^{\sigma-2} \frac{h^{i+1}}{(i+1)!} \frac{d^i f_N}{dt^i}(t_m, y_m) + \sum_{i=1}^{\sigma-2} \frac{h^{i+1}}{(i+1)!} \frac{d^i f_S}{dt^i}(t_{m+1}, y_{m+1}) + \mathcal{O}(h^\sigma), \end{aligned} \tag{3.4}$$

where $e_{m+1} = y_{m+1} - \eta_{m+1}$ is the error at t_{m+1}. Let

$$u_m = (f_N(t_m, y_m) - f_N(t_m, \eta_m)) + (f_S(t_{m+1}, y_{m+1}) - f_S(t_{m+1}, \eta_{m+1})),$$

and

$$r_m = \sum_{i=1}^{\sigma-2} \frac{h^{i+1}}{(i+1)!} \left(\frac{d^i f_N}{dt^i}(t_m, y_m) + \frac{d^i f_S}{dt^i}(t_{m+1}, y_{m+1}) \right). \tag{3.5}$$

To prove the smoothness of the rescaled error vector, $\tilde{e} = e/h$, we will use an inductive approach with respect to s, the degree of smoothness. First, note that a discrete differentiation of the rescaled error vector gives,

$$(d_1\tilde{e})_m = \frac{\tilde{e}_{m+1} - \tilde{e}_m}{h} = \frac{u_m}{h} + \frac{r_m}{h^2} + \mathcal{O}(h^{\sigma-2}), \tag{3.6}$$

We are now ready to prove that $\vec{\tilde{e}}$ has M degrees of smoothness by induction. Since $\|\vec{\tilde{e}}\|_\infty \sim \mathcal{O}(h)$, $\vec{\tilde{e}}$ has at least zero degrees of smoothness in the discrete sense. Assume that $\vec{\tilde{e}}$ has $s \leq M-1$ degrees of smoothness. We will show that $\overrightarrow{d_1\tilde{e}}$ has s degrees of smoothness, from which we can conclude that $\vec{\tilde{e}}$ has $(s+1)$ degrees of smoothness.

First,

$$\begin{aligned} u_m = & \sum_{i=1}^{\sigma-2} \frac{1}{i!} e_m^i \frac{\partial^i f_N}{\partial y^i}(t_m, y_m) + \sum_{i=1}^{\sigma-2} \frac{1}{i!} e_{m+1}^i \frac{\partial^i f_S}{\partial y^i}(t_{m+1}, y_{m+1}) \\ & + \mathcal{O}((e_m)^{\sigma-1}) + \mathcal{O}((e_{m+1})^{\sigma-1}) \\ = & \sum_{i=1}^{\sigma-2} \frac{1}{i!} \tilde{e}_m^i h^i \frac{\partial^i f_N}{\partial y^i}(t_m, y_m) + \sum_{i=1}^{\sigma-2} \frac{1}{i!} \tilde{e}_{m+1}^i h^i \frac{\partial^i f_S}{\partial y^i}(t_{m+1}, y_{m+1}) \\ & + \mathcal{O}((h\tilde{e}_m)^{\sigma-1}) + \mathcal{O}((h\tilde{e}_{m+1})^{\sigma-1}), \end{aligned} \tag{3.7}$$

where we have performed a Taylor expansion of $f_N(t, \eta_m)$ about $y = y_m$ and of $f_S(t, \eta_{m+1})$ about $y = y_{m+1}$. Denote f_{y^i} to represent either $\frac{\partial^i f_N}{\partial y^i}$ or $\frac{\partial^i f_S}{\partial y^i}$. Since f_{y^i} has $(\sigma - i - 1)$ degrees of smoothness in the continuous sense, $\overrightarrow{f_{y^i}} = [f_{y^i}(t_0, y_0), \ldots, f_{y^i}(t_M, y_M)]$ has $(\sigma - i - 1)$ degrees of smoothness in the discrete sense. Consequently, $h^{i-1}\overrightarrow{f_{y^i}}$ has $(\sigma - 2)$ degrees of smoothness, which implies that $\frac{u_m}{h}$ has $\min(\sigma - 2, s)$ degrees of smoothness. Also $\frac{r_m}{h^2}$ has at least s degrees of smoothness from the smoothness property of the IVP (1.1). Hence the fact that $\overrightarrow{d_1\tilde{e}}$ has s degrees of smoothness implies that $\vec{\tilde{e}}$ has $(s+1)$ degrees of smoothness. Since this argument holds for $\sigma \geq M+2$, we can conclude that $\vec{\tilde{e}}$ has M degrees of smoothness. □

Note that most results for IDC-ARK methods stated in Lemmas 3.6, 3.7 and the proofs are similar to the analysis in [9]. However, there is a main difference between the proof of the truncation error for IDC-RK [9] and IDC-ARK methods within the proof of Lemma 3.7, described below within the proof of Proposition 3.9. For clarity, we outline the proofs of the lemmas and provide the essential details where the proofs differ.

Proof of Lemma 3.6.

Proof. We drop the superscript [0] since there is no ambiguity. First, we Taylor expand the error, $e_{m+1} = y_{m+1} - \eta_{m+1}$, where y_{m+1} is the exact solution and η_{m+1} is the numerical solution obtained using an r_0th order ARK method. In [9], for IDC-RK methods, the expansion is about $t = t_m$, whereas in this paper, for semi-implicit IDC-ARK methods, the expansion is analogous to (3.4), given in the proof of Lemma 3.8, above. That is, the explicit part is expanded about $t = t_m$, and the expansion for the implicit part is about $t = t_{m+1}$. Then the error can be written in the form

$$\begin{aligned} e_{m+1} = e_m &+ h\mathcal{U}_m\big(f_N(t_m, y_m), e_m, f_S(t_{m+1}, y_{m+1}), e_{m+1}\big) \\ &+ \mathcal{R}_m\big(h, f_N(t_m, y_m), f_S(t_{m+1}, y_{m+1})\big) + \mathcal{O}(h^\sigma), \end{aligned}$$

where $\mathcal{U}_m$ is analogous to (3.7) with $f(t_m, \eta_m)$ and $f(t_{m+1}, \eta_{m+1})$ expanded about y_m and y_{m+1}, respectively, $\mathcal{R}_m$ is analogous to (3.5), and $\mathcal{R}_m$ is $\mathcal{O}(h^{r_0+1})$. Now we may bound $\|e^{[0]}\|_\infty$ by induction. By definition, $e_0 = 0$, so certainly, $e_0 \sim \mathcal{O}(h^{r_0+1})$. Now assume $e_m \sim \mathcal{O}(h^{r_0+1})$. We already know that $\mathcal{R}_m \sim \mathcal{O}(h^{r_0+1})$. Since $\mathcal{U}_m$ is analogous to (3.7), we know that its terms consist of derivatives of f_N and f_S multiplied by powers of e_m and e_{m+1}. Trivially, any terms with e_m are at worst $\mathcal{O}(h^{r_0+1})$, which would complete the proof for IDC-RK methods, but not for IDC-ARK methods. For IDC-ARK methods, we are left with

$$e_{m+1} = \mathcal{O}(h^{r_0+1}) + ha_1 e_{m+1} + ha_2 e_{m+1}^2 + \cdots + ha_{\sigma-2}\, e_{m+1}^{\sigma-2} + \mathcal{O}(h\, e_{m+1}^{\sigma-1}),$$

where a_i are constants depending on derivatives of f_N and f_S, so clearly $e_{m+1} \sim \mathcal{O}(h^{r_0+1})$, which completes the inductive proof.

Smoothness results are proved analogously to Lemma 3.8. □

Before proving the IDC-ARK correction loop, Lemma 3.7, we first introduce some notational details related to the rescaling of the error equation, and refer back to the error Equation (2.10). The analysis for the correction steps in this section will rely on this form of the error equation.

Assume that after the $(k-1)^{\text{th}}$ correction, the method is $\mathcal{O}(h^{s_{k-1}})$. We rescale the error $e^{(k-1)}$ (2.6), and $Q^{(k-1)}$ and G_α from the error Equation (2.10), by $h^{s_{k-1}}$. Define the rescaled quantities as

$$\widetilde{e}^{(k-1)}(t) = \frac{1}{h^{s_{k-1}}} e^{(k-1)}(t), \tag{3.8a}$$

$$\widetilde{Q}^{(k-1)}(t) = \frac{1}{h^{s_{k-1}}} Q^{(k-1)}(t), \tag{3.8b}$$

$$\widetilde{G}_\alpha^{(k-1)}(t, \widetilde{Q}^{(k-1)}(t)) = \frac{1}{h^{s_{k-1}}} G_\alpha(t, h^{s_{k-1}} \widetilde{Q}^{(k-1)}(t)), \quad \alpha = N, S. \tag{3.8c}$$

Then the rescaled error equation is given by

$$\begin{aligned} (\widetilde{Q}^{(k-1)})'(t) &= \widetilde{G}_N^{(k-1)}(t, \widetilde{Q}^{(k-1)}(t)) + \widetilde{G}_S^{(k-1)}(t, \widetilde{Q}^{(k-1)}(t)), \\ \widetilde{Q}^{(k-1)}(t) &= 0. \end{aligned} \tag{3.9}$$

The rescaled Equation (3.8) and (3.9) are $\mathcal{O}(1)$ (see [9]).

A p-stage, r^{th} order ARK method (2.3) applied to (3.9) gives, for $i = 1, 2, \ldots, p$ and $\alpha = N, S$,

$$\begin{aligned} \widetilde{k}_i^\alpha &= \widetilde{G}_\alpha^{(k-1)} \left(t_0 + c_i h, \widetilde{Q}_0^{[k-1]} + h \left(\sum_{j=1}^{i-1} a_{ij}^N \widetilde{k}_j^N + \sum_{j=1}^{p} a_{ij}^S \widetilde{k}_j^S \right) \right), \\ \widetilde{\Omega}_1^{[k]} &= \widetilde{Q}_0^{[k-1]} + h \sum_{i=1}^{p} (b_i^N \widetilde{k}_i^N + b_i^S \widetilde{k}_i^S), \end{aligned} \tag{3.10}$$

where the Greek letter $\widetilde{\Omega}$ denotes the rescaled solution in the numerical space, and $\widetilde{Q}_0^{[k-1]} = \widetilde{Q}^{(k-1)}(t_0)$. In the actual implementation, we discretize (2.10) as in Equations (2.12) with F_N, F_S. We use (2.12) with G_N, G_S instead of F_N, F_S for the analysis. **Proof of Lemma 3.7.**

Proof. The smoothness results are analogous to, but more tedious than, the proof of Lemma 3.8. Thus they will not be presented in this paper.

The outline of the proof of the truncation error is:

1. Subtract the numerical error vector from the integrated error equation,

$$e^{[k]}_{m+1} = e^{[k-1]}_{m+1} - \delta^{[k]}_{m+1},$$

   and make appropriate substitutions and expansions using the rescaled equations. This is quite similar to the proof in [9].
2. Bounding $\|e^{[k]}\|_\infty$ using an inductive argument, similarly to the proof of Lemma 3.6, and similarly to the proof in [9].

The details of bounding $\|e^{[k]}\|_\infty$ require several minor propositions, which can be found in [9], and one quite important proposition, which we state and prove below, that claims the equivalence of the rescaled and unscaled error vectors and shows the truncation error of the rescaled error vector. Employing the rescaled Equations (3.9), (3.8), and (3.10) greatly simplifies the proof of the truncation error. ∎

The proof of the following proposition is where the essential difference between the proof of the truncation error for IDC-RK [9] and IDC-ARK methods lies.

Proposition 3.9. *The Taylor series for the rescaled exact error,* $\widetilde{e}^{(k-1)}(t_0+h) = \frac{1}{h^{s_{k-1}}} e^{(k-1)}(t_0+h)$*, and for the rescaled numerical error,* $\frac{1}{h^{s_{k-1}}}\delta^{[k]}_1$ *in* (2.12)*, coincide up to and including the term* h^r *for a sufficiently smooth error function* $\widetilde{e}^{(k-1)}(t)$*, when the exact solution* $y(t)$ *has* σ *and* f_N*,* f_S *have* $\sigma-1$ *degrees of smoothness.*

Proof. It suffices to prove the following claim:

$$h^{s_{k-1}}\widetilde{k}^\alpha_i = k^\alpha_i + \mathcal{O}(h^{\sigma-1}), \quad \forall\, i = 1,2,\ldots,p, \quad \alpha = N, S, \tag{3.11}$$

where k^α_i and $\widetilde{k}^\alpha_i$ are as in Equations (2.12) and (3.10). Note in [9], the $\mathcal{O}(h^{\sigma-1})$ term is not required due to the explicit nature of IDC-RK methods, whereas here for semi-implicit IDC-ARK methods, although the term may not be necessary, it allows for the structure of the proof that we provide. The claim (3.11) states the equivalence of the rescaled and unscaled error vectors. More specifically, it asserts that the rescaled error vectors (rescaled by $h^{s_{k-1}}$), obtained from formally applying ARK methods to a rescaled error equation, are equivalent (up to order $\mathcal{O}(h^{\sigma-1})$) to the error resulting from the numerical formulation that we implement. Thus, since an ${r_k}^{\text{th}}$ order ARK method applied to the rescaled error equations achieves ${r_k}^{\text{th}}$ order, the (unscaled) numerical implementation is $\mathcal{O}(h^{s_{k-1}+r_k})$, provided the exact solution $y(t)$ and the functions f_N, f_S are sufficiently smooth.

Before proving (3.11), we set

$$A_i \doteq h\left(\sum_{j=1}^{i-1} a^N_{ij} h^{s_{k-1}}\widetilde{k}^N_j + \sum_{j=1}^{p} a^S_{ij} h^{s_{k-1}}\widetilde{k}^S_j\right) = h\sum_{j=1}^{p}(a^N_{ij} h^{s_{k-1}}\widetilde{k}^N_j + a^S_{ij} h^{s_{k-1}}\widetilde{k}^S_j), \tag{3.12}$$

$$B_i \doteq h\left(\sum_{j=1}^{i-1} a^N_{ij} k^N_j + \sum_{j=1}^{p} a^S_{ij} k^S_j\right) = h\sum_{j=1}^{p}(a^N_{ij} k^N_j + a^S_{ij} k^S_j), \tag{3.13}$$

$$C^\alpha_{lj} \doteq \frac{1}{l!}\frac{\partial^l}{\partial y^l} G_\alpha(t_0 + c_j h, Q^{(k-1)}_0)\sum_{v=1}^{l} A^{l-v}_j B^{v-1}_j,$$

where $\alpha = N, S$, and the equalities in (3.12) and (3.13) hold since $a_{ij}^N = 0$ for $j \geq i$. Now, to prove claim (3.11), we prove that $h^{s_{k-1}}\widetilde{k}_i^\alpha - k_i^\alpha = \mathcal{O}(h^n)$ for $n = 1, 2, \ldots, \sigma - 1$, for $i = 1, 2, \ldots, p$, and $\alpha = N, S$ using induction with respect to n; i.e., we show that

$$\begin{aligned} &h^{s_{k-1}}\widetilde{k}_i^\alpha - k_i^\alpha \\ &\quad = h^n \sum_{l,n} \sum_{j,n} (\beta_{lj,n}^{\alpha_N}(h^{s_{k-1}}\widetilde{k}_{j_n}^N - k_{j_n}^N) + \beta_{lj,n}^{\alpha_S}(h^{s_{k-1}}\widetilde{k}_{j_n}^S - k_{j_n}^S)) + \mathcal{O}(h^{\sigma-1}), \end{aligned} \tag{3.14}$$

where the notation is as follows.

$$\sum_{l,n} = \sum_{l_1}^{\sigma-2} \sum_{l_2}^{\sigma-2} \cdots \sum_{l_n}^{\sigma-2}, \quad \sum_{j,n} = \sum_{j_1}^{p} \sum_{j_2}^{p} \cdots \sum_{j_n}^{p},$$
$$\beta_{lj,n}^{\alpha_N} = \sum_{1}^{2^{n-1}} \prod_{1}^{n} Ca, \quad \beta_{lj,n}^{\alpha_S} = \sum_{1}^{2^{n-1}} \prod_{1}^{n} Ca,$$

where for the β coefficients, C represents some C_{lj}^N or C_{lj}^S, a represents some a_{ij}^N or a_{ij}^S, and ${}^{\alpha_N}$ and ${}^{\alpha_S}$ denote that the C, a coefficients that appear in the sums depend on α and N or α and S, respectively. For example, if $n = 2$, then

$$\begin{aligned} \beta_{lj,2}^{\alpha_N} &= C_{l_1 i}^{\alpha} a_{ij_1}^N C_{l_2 j_1}^N a_{j_1 j_2}^N + C_{l_1 i}^{\alpha} a_{ij_1}^S C_{l_2 j_1}^S a_{j_1 j_2}^N, \\ \beta_{lj,2}^{\alpha_S} &= C_{l_1 i}^{\alpha} a_{ij_1}^N C_{l_2 j_1}^N a_{j_1 j_2}^S + C_{l_1 i}^{\alpha} a_{ij_1}^S C_{l_2 j_1}^S a_{j_1 j_2}^S, \end{aligned}$$

which is clarified in the expansions below.

We prove (3.14), and hence (3.11), using induction with respect to n, the power of h in equation (3.14).

- $n = 1$: Using the rescaled quantities (3.8) and Taylor expanding $G_\alpha(t, y)$ about $Q_0^{(k-1)}$ in y, we obtain for $\alpha = N, S$,

$$\begin{aligned} &h^{s_{k-1}}\widetilde{k}_i^\alpha - k_i^\alpha \\ &= h^{s_{k-1}}\widetilde{G}_\alpha^{(k-1)}(t_0 + c_i h, \widetilde{Q}^{(k-1)}(t_0) + h\sum_{j=1}^{p}(a_{ij}^N \widetilde{k}_j^N + a_{ij}^S \widetilde{k}_j^S)) \\ &\quad - G_\alpha(t_0 + c_i h, Q^{(k-1)}(t_0) + h\sum_{j=1}^{p}(a_{ij}^N k_j^N + a_{ij}^S k_j^S)) \\ &= G_\alpha(t_0 + c_i h, h^{s_{k-1}}(\widetilde{Q}^{(k-1)}(t_0) + h\sum_{j=1}^{p}(a_{ij}^N \widetilde{k}_j^N + a_{ij}^S \widetilde{k}_j^S))) \\ &\quad - G_\alpha(t_0 + c_i h, Q^{(k-1)}(t_0) + B_i) \\ &= G_\alpha(t_0 + c_i h, Q^{(k-1)}(t_0) + A_i) - G_\alpha(t_0 + c_i h, Q_0^{(k-1)} + B_i) \\ &= \sum_{l_1=1}^{\sigma-2} \frac{1}{l_1!} \frac{\partial^{l_1}}{\partial y^{l_1}} G_\alpha(t_0 + c_i h, Q^{(k-1)}(t_0))((A_i)^{l_1} - (B_i)^{l_1}) + \mathcal{O}(h^{\sigma-1}). \end{aligned} \tag{3.15}$$

Factoring $(A_i)^{l_1} - (B_i)^{l_1} = (A_i - B_i)\sum_{v=1}^{l_1} A_i^{l_1-v} B_i^{v-1}$, $\forall i = 1, 2, \ldots, p$, then

$$\begin{aligned}
&h^{s_{k-1}}\widetilde{k}_i^{\alpha} - k_i^{\alpha} \qquad (3.16)\\
&= \sum_{l_1=1}^{\sigma-2} C_{l_1 i}^{\alpha}(A_i - B_i) + \mathcal{O}(h^{\sigma-1})\\
&= h\sum_{l_1=1}^{\sigma-2} C_{l_1 i}^{\alpha} \sum_{j_1=1}^{p} (a_{ij_1}^N(h^{s_{k-1}}\widetilde{k}_{j_1}^N - k_{j_1}^N) + a_{ij_1}^S(h^{s_{k-1}}\widetilde{k}_{j_1}^S - k_{j_1}^S)) + \mathcal{O}(h^{\sigma-1})\\
&= h\sum_{l_1=1}^{\sigma-2}\sum_{j_1=1}^{p} (C_{l_1 i}^{\alpha} a_{ij_1}^N(h^{s_{k-1}}\widetilde{k}_{j_1}^N - k_{j_1}^N) + C_{l_1 i}^{\alpha} a_{ij_1}^S(h^{s_{k-1}}\widetilde{k}_{j_1}^S - k_{j_1}^S)) + \mathcal{O}(h^{\sigma-1})\\
&= h^1 \sum_{l,1}\sum_{j,1} (\beta_{lj,1}^{\alpha_N}(h^{s_{k-1}}\widetilde{k}_{j_1}^N - k_{j_1}^N) + \beta_{lj,1}^{\alpha_S}(h^{s_{k-1}}\widetilde{k}_{j_1}^S - k_{j_1}^S)) + \mathcal{O}(h^{\sigma-1}).
\end{aligned}$$

Note here that $\beta_{lj,1}^{\alpha_N} = C_{l_1 i}^{\alpha} a_{ij_1}^N$, and $\beta_{lj,1}^{\alpha_S} = C_{l_1 i}^{\alpha} a_{ij_1}^S$.

- $n+1$: Assume (3.14) holds for $n < \sigma - 1$ (for both $\alpha = N$ and S). Following the same process of Taylor expanding as in Equations (3.15) and (3.16) in the $n = 1$ case , we see that for $\alpha = N, S$:

$$\begin{aligned}
&h^{s_{k-1}}\widetilde{k}_i^{\alpha} - k_i^{\alpha}\\
&= h^n \sum_{l,n}\sum_{j,n} (\beta_{lj,n}^{\alpha_N}(h^{s_{k-1}}\widetilde{k}_{j_n}^N - k_{j_n}^N) + \beta_{lj,n}^{\alpha_S}(h^{s_{k-1}}\widetilde{k}_{j_n}^S - k_{j_n}^S)) + \mathcal{O}(h^{\sigma-1})\\
&\overset{(3.16)}{=} h^n \sum_{l,n}\sum_{j,n} (\beta_{lj,n}^{\alpha_N}(h \sum_{l_{n+1}=1}^{\sigma-2}\sum_{j_{n+1}=1}^{p} (C_{l_{n+1}j_n}^N a_{j_n j_{n+1}}^N (h^{s_{k-1}}\widetilde{k}_{j_{n+1}}^N - k_{j_{n+1}}^N)\\
&\quad + C_{l_{n+1}j_n}^N a_{j_n j_{n+1}}^S (h^{s_{k-1}}\widetilde{k}_{j_{n+1}}^S - k_{j_{n+1}}^S)) + \mathcal{O}(h^{\sigma-1}))\\
&\quad + \beta_{lj,n}^{\alpha_S}(h \sum_{l_{n+1}=1}^{\sigma-2}\sum_{j_{n+1}=1}^{p} (C_{l_{n+1}j_n}^S a_{j_n j_{n+1}}^N (h^{s_{k-1}}\widetilde{k}_{j_1}^N - k_{j_1}^N)\\
&\quad + C_{l_{n+1}j_n}^S a_{j_n j_{n+1}}^S (h^{s_{k-1}}\widetilde{k}_{j_{n+1}}^S - k_{j_{n+1}}^S)) + \mathcal{O}(h^{\sigma-1}))) + \mathcal{O}(h^{\sigma-1})\\
&= h^{n+1} \sum_{l,n+1}\sum_{j,n+1} ((\beta_{lj,n}^{\alpha_N} C_{l_{n+1}j_n}^N a_{j_n j_{n+1}}^N + \beta_{lj,n}^{\alpha_S} C_{l_{n+1}j_n}^S a_{j_n j_{n+1}}^N)(h^{s_{k-1}}\widetilde{k}_{j_1}^N - k_{j_1}^N)\\
&\quad + (\beta_{lj,n}^{\alpha_N} C_{l_{n+1}j_n}^N a_{j_n j_{n+1}}^S + \beta_{lj,n}^{\alpha_S} C_{l_{n+1}j_n}^S a_{j_n j_{n+1}}^S)(h^{s_{k-1}}\widetilde{k}_{j_{n+1}}^S - k_{j_{n+1}}^S)) + \mathcal{O}(h^{\sigma-1})\\
&= h^{n+1} \sum_{l,n+1}\sum_{j,n+1} (\beta_{lj,n+1}^{\alpha_N}(h^{s_{k-1}}\widetilde{k}_{j_{n+1}}^N - k_{j_{n+1}}^N) + \beta_{lj,n+1}^{\alpha_S}(h^{s_{k-1}}\widetilde{k}_{j_{n+1}}^S - k_{j_{n+1}}^S))\\
&\quad + \mathcal{O}(h^{\sigma-1}), \quad \forall i = 1, 2, \ldots, p,
\end{aligned}$$

where

$$\begin{aligned}
\beta_{lj,n+1}^{\alpha_N} &= \beta_{lj,n}^{\alpha_N} C_{l_{n+1}j_n}^N a_{j_n j_{n+1}}^N + \beta_{lj,n}^{\alpha_S} C_{l_{n+1}j_n}^S a_{j_n j_{n+1}}^N,\\
\beta_{lj,n+1}^{\alpha_S} &= \beta_{lj,n}^{\alpha_N} C_{l_{n+1}j_n}^N a_{j_n j_{n+1}}^S + \beta_{lj,n}^{\alpha_S} C_{l_{n+1}j_n}^S a_{j_n j_{n+1}}^S,
\end{aligned}$$

which completes the inductive proof, so it follows that (3.11) holds.

The rest of the proof of Proposition 3.9 is similar to the IDC-RK analysis in [9].

In particular,

$$\begin{aligned}
\delta^{[k-1]} &\doteq h^{s_{k-1}}\widetilde{\Omega}_1^{[k-1]} - \int_{t_0}^{t_1} \epsilon^{(k-1)}(\tau)d\tau \\
&= h^{s_{k-1}}\,\widetilde{Q}^{(k-1)}(t_0) + h\sum_{j=1}^{p} b_j^N\, h^{s_{k-1}}\widetilde{k}_j^N + h\sum_{j=1}^{p} b_j^S\, h^{s_{k-1}}\widetilde{k}_j^S - \int_{t_0}^{t_1} \epsilon^{(k-1)}(\tau)d\tau \\
&= Q^{(k-1)}(t_0) - \int_{t_0}^{t_1} \epsilon^{(k-1)}(\tau)d\tau + h\sum_{j=1}^{p} b_j^N\,(k_j^N + \mathcal{O}(h^{\sigma-1})) + h\sum_{j=1}^{p} b_j^S\,(k_j^S + \mathcal{O}(h^{\sigma-1})) \\
&= e_0^{[k-1]} - \int_{t_0}^{t_0+h} \epsilon^{(k-1)}(\tau)d\tau + h\sum_{j=1}^{p} b_j^N\,k_j^N + h\sum_{j=1}^{p} b_j^S\,k_j^S + \mathcal{O}(h^{\sigma}).
\end{aligned}$$

Since the Taylor series for $\widetilde{Q}^{(k-1)}(t_0+h)$ and $\widetilde{\Omega}_1^{[k]}$ coincide up to and including the term h^r (since $\widetilde{\Omega}_1^{[k]}$ is an r^{th} order ARK approximation to $\widetilde{Q}^{(k-1)}(t_0+h)$), then the Taylor series for $\frac{1}{h^{s_{k-1}}}e^{(k-1)}(t_0+h)$ and $\frac{1}{h^{s_{k-1}}}\delta_1^{[k]}$ also coincide up to and including the term h^r. ∎

Corollary 3.10. *Let $y(t)$ be the solution to the IVP*

$$\begin{aligned}
y'(t) &= f(t,y), \quad t \in [0,T], \\
y(0) &= y_0,
\end{aligned}$$

and suppose $y(t)$ has at least σ ($\sigma \geq M+2$) and f has at least $\sigma - 1$ degrees of smoothness in the continuous sense. Consider one time interval of an IDC method with $t \in [0,H]$ and uniformly distributed quadrature points (2.5)*. Suppose an ${r_0}^{th}$ order implicit RK method is used in the prediction step and $(r_1, r_2, \ldots, r_{K_{loop}})^{th}$ order implicit RK methods are used in K_{loop} correction steps. Let $s_k = \sum_{j=0}^{k} r_j$. If $s_{K_{loop}} \leq M+1$, then the local truncation error is of order $\mathcal{O}(h^{s_{K_{loop}}+1})$.*

Proof. The result follows from Theorem 3.5, if we set $f_N = 0$. ∎

4. Stability

It is well accepted that the linear stability of an explicit or implicit scheme can be measured by the stability region [15]. However, the stability measure for a mixed scheme, such as ARK, is much more complicated. In the literature, different measures for stability have been proposed for semi-implicit (including additive Runge-Kutta) methods [3, 2, 26, 24]. We adopt the stability measure from [3, 2], which is explained clearly in [25] via the following definitions.

Definition 4.1. *The* amplification factor*, $Am(\lambda)$ with $\lambda = \alpha + i\beta$, for a semi-implicit method, is interpreted as the numerical solution to*

$$y'(t) = \alpha y(t) + i\beta y(t), \quad y(0) = 1, \tag{4.1}$$

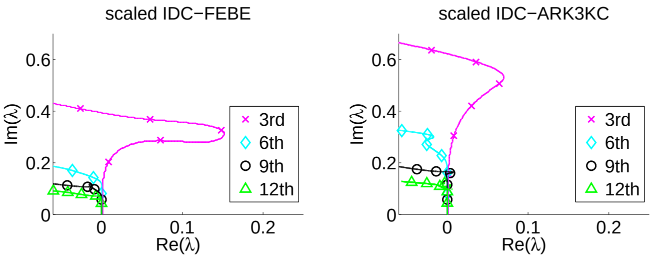
where the explicit component of the semi-implicit method is applied to the imaginary part $i\beta y(t)$, and the implicit component of the semi-implicit method is applied to the real part $\alpha y(t)$, after a time step of size 1 *for $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{R}$, i.e., $Am(\lambda) = y(1)$.*

Definition 4.2. *The* stability region*, S, for a semi-implicit method, is the subset of the complex plane $\mathbb{C}$ consisting of all λ such that $Am(\lambda) \leq 1$ for ODE 4.1, i.e., $S = \{\lambda : Am(\lambda) \leq 1\}$.*

Definition 4.3. *The* scaled stability region, $\widetilde{S}$, *for a semi-implicit method, is the stability region S divided by the number of implicit function evaluations.*

For IDC methods, the number of implicit function evaluations is $sM(K_{loop}+1)$, where s is the number of stages of the implicit RK method embedded inside IDC, and M is the number of intervals. For example, an $(M+1)^{st}$ order IDC-FBE method requires $M(M+1)$ implicit function evaluations since BE involves only one stage, and an $(M+1)^{st}$ order IDC-ARK3KC method requires $4M((M+1)/3)$ function evaluations since ARK3KC has 4 stages and $(M+1)/3$ loops are required to achieve $(M+1)^{st}$ order accuracy.

In Figure 4.1a, the scaled stability regions for 3rd, 6th, 9th, and 12th order IDC-FBE (IDC constructed using forward and backward Euler, or SISDC with uniform quadrature nodes) with 2, 5, 8, and 11 correction loops, respectively, are computed numerically and plotted. The regions are similar to the stability regions of the SISDC methods that use Gauss-Lobatto quadrature nodes [25]. Figure 4.1b shows 3rd, 6th, 9th, and 12th order IDC-ARK3KC (IDC constructed with 3rd order ARK3KC) with 0, 1, 2, and 3 correction loops, respectively. The scaled stability regions of IDC-ARK schemes are significantly larger than those of IDC-FBE schemes. This correlates with results in [9], where IDC-RK methods were found to possess better stability properties than SDC methods [9]. Such a stability measure has implications on the stability property of the IDC-ARK method, especially when applied to the convection-diffusion problem in Section 5.1, where the convection term is treated explicitly and diffusion term implicitly. We remark that the stability property of the semi-implicit scheme greatly depends on the splitting of Equation (1.1), therefore, the stability analysis we presented has to be adapted for specific problems.



(a) IDC-FBE (b) IDC-ARK3KC

Fig. 4.1: *Scaled stability regions for 3rd, 6th, 9th, and 12th order IDC constructed using (a) forward and backward Euler and (b) 3rd order ARK3KC. The inside of each stability region is the region to the left and below the portion of its boundary that is shown. IDC-ARK3KC regions are significantly larger than IDC-FBE regions.*

5. Numerical results

We now summarize numerical experiments applying IDC-ARK methods to a linear advection-diffusion equation and a nonlinear Brusselator problem. Accuracy results that support Theorem 3.5 and efficiency results are presented. Several different ARK methods of various orders were tested within the IDC prediction and correction loops: second order ARK2A1 [24] and ARK2ARS [2], third order ARK3KC [21] and ARK3BHR (Appendix 1. in [7]), and fourth order ARK4A2 [24] and ARK4KC [21]. For brevity, we select only a few of these to present. For the convergence studies, error is plotted versus the time step size H. In each implementation, we anticipate that *# loops * ARK order = expected IDC order.* To compare efficiency, the IDC-ARK methods were compared with their constituent ARK methods and with IDC-FBE methods (which are SISDC methods with uniform quadrature nodes [25]).

We further comment that the focus of our study is not the choice of f_N, f_S, and thus we make naive choices of f_N and f_S in each example below, acknowledging that better choices are likely to exist (in particular, there is much study on the choice of f_N, f_S for PDEs, such as in [19, 20, 14]). Note also that, in most of the literature, the examples of semi-implicit methods used to solve problems with stiff regions only test convergence and efficiency up to the time just *before* the stiff layer, at which point they claim that adaptive steps should be taken [21, 7, 2]. In this work, we also present most tests in the standard way, computing solutions up to the time just before the stiff layer, at which point we assume adaptive steps should be taken if we were to continue the computation.

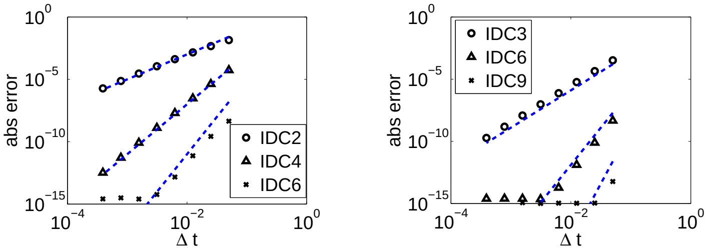
5.1. Advection-diffusion example. Consider the advection-diffusion equation

$$u_t = -u_x + u_{xx}, \quad x \in [0, \pi/2], \quad t \in [0, 0.1], \tag{5.1a}$$

$$u(x, 0) = 2 + \sin(4x), \tag{5.1b}$$

with periodic boundary conditions. We solve (5.1) via the method of lines with fast Fourier transform (FFT) for the spatial derivatives and time-stepping with semi-implicit IDC-ARK, where the advection term is treated explicitly ($f_N = -u_x$) and the diffusion term is treated implicitly ($f_S = u_{xx}$). This choice of methods ensures that the error is dominated by the time-stepping. The numerical solution is compared to a reference solution, computed with a ninth order IDC-FBE method with $\Delta x = \pi/780$, $\Delta t = 1/24800$. Error is calculated as the l^1 spatial norm of the absolute error at the final time $T = 0.1$.

In Figure 5.1, we see the expected theoretical orders or accuracy; that is, orders of convergence increase by two and three for IDC constructed with second and third order ARK integrators, respectively. Here $\Delta x = \pi/260 \approx 10^{-2}$. Notice that $\Delta t \gg \Delta x^2$, a key reason for employing semi-implicit methods.

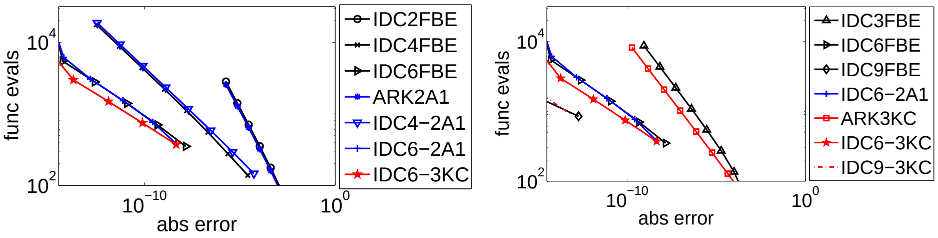


(a) IDC-ARK2A1 (b) IDC-ARK3KC

Fig. 5.1: *Convergence study of various IDC schemes applied to the advection-diffusion equation at* $T = 0.1$. *In (a), we have 2nd, 4th, and 6th order IDC constructed using 2nd order ARK2A1 and (b), 3rd, 6th, and 9th order IDC constructed using 3rd order ARK3KC. The numerical results agree with the theoretical order of accuracy (dotted lines).*

A reasonable measure of efficiency is the number of implicit function evaluations versus the error (as in [25]), since the implicit calls will often dominate due to the

Newton iterations. It is not surprising that higher order IDC-ARK methods require fewer function evaluations for accurate solutions, as shown in Figure 5.2. IDC6-ARK3KC is more efficient than IDC6-FBE and IDC6-ARK2A1.



(a) Various IDC2, IDC4 and IDC6 schemes

(b) Various IDC3, IDC6 and IDC9 schemes

Fig. 5.2: *Efficiency study of various IDC schemes applied to the advection diffusion equation. The number of implicit function evaluations is plotted as a function of the error at* $T = 0.1$*. High order IDC-ARK methods require fewer function evaluations to obtain accurate solutions.*

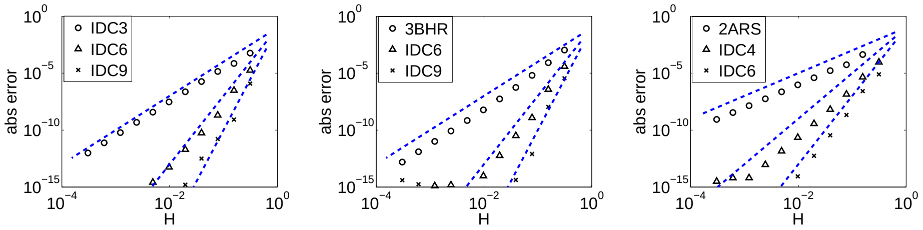
5.2. Brusselator example. We now consider the Brusselator problem with diffusion,

$$\begin{aligned} u_t &= A + u^2 v - (B+1)u + \alpha u_{xx}, \quad x \in [0,1], \quad t \in [0,10], \\ v_t &= Bu - u^2 v + \alpha v_{xx}, \quad A = 1, \quad B = 3, \quad \alpha = 0.02, \\ u(0,t) &= u(1,t) = 1, \quad v(0,t) = v(1,t) = 3, \\ u(x,0) &= 1 + \sin(2\pi x), \quad v(x,0) = 3. \end{aligned} \tag{5.2}$$

We solve (5.2) via the method of lines with second order centered differencing for the spatial derivatives and time-stepping with semi-implicit IDC-ARK, where the nonlinear term, $f_N = [A + u^2 v - (B+1)u, Bu - u^2 v]^T$, is treated explicitly and the diffusion term, $f_S = [\alpha u_{xx}, \alpha v_{xx}]^T$ is treated implicitly. Error is computed by taking the maximum of the l^1 spatial norm of u and v, as compared with successive solutions at the final time $T = 10$.

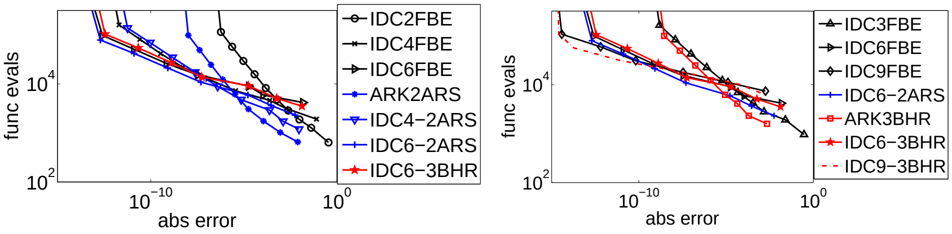
Figures 5.3 also shows that our IDC-ARK methods attain the expected theoretical orders of accuracy; that is, orders of convergence increase by two and three for IDC constructed with second and third order ARK integrators. For this problem, IDC-ARK methods have a smaller error coefficient. Again, these computations are stable ($\Delta t \gg \Delta x^2 \sim 10^{-4}$), the key motivation behind implementing semi-implicit methods.

In Figure 5.4, it is again observed that higher order IDC-ARK methods require fewer function evaluations for accurate solutions, as shown in Figure 5.2. The ARK2ARS and ARK3BHR1 methods are more efficient than the IDC2-FBE and IDC3FBE. IDC6-ARK2ARS is more efficient than IDC6-FBE, and IDC9-ARK3BHR1 is more efficient than IDC9-FBE. It would seem that the efficiency is dependent on the embedded ARK scheme.



(a) IDC-FBE (b) IDC-ARK3BHR (c) IDC-ARK2ARS

Fig. 5.3: *Convergence study of various IDC schemes applied to the Brusselator with* $\Delta x = 1/50$ *at* $T = 10$*. (a) shows the convergence of IDC constructed with forward-backward Euler, (b) shows the convergence of IDC constructed with ARK3BHR, (c) shows the convergence of IDC constructed with ARK2ARS. The numerical results agree with the theoretical order of accuracy (dotted lines).*



(a) Various IDC2, IDC4 and IDC6 Schemes (b) Various IDC3, IDC6 and IDC9 Schemes

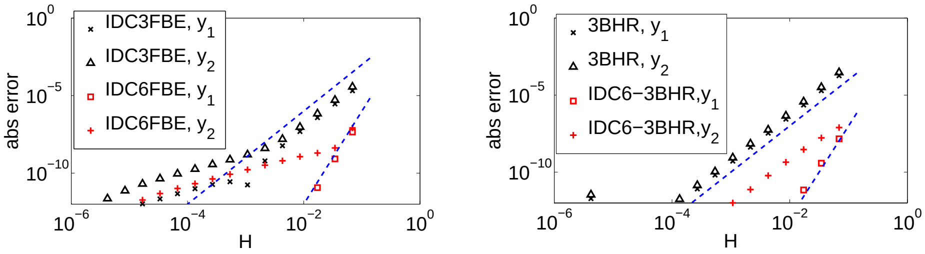
Fig. 5.4: *Efficiency studies of various IDC schemes applied to the Brusselator. The number of implicit function evaluations is plotted as a function of error at* $T = 10$*. High order IDC-ARK methods require fewer function evaluations to obtain accurate solutions.*

5.3. Van der Pol's example. Finally, we examine a standard nonlinear oscillatory test problem, Van der Pol's equation [7],

$$\begin{aligned} &y_1'(t) = y_2(t), \quad y_2'(t) = \frac{1}{\epsilon}(-y_1(t) + (1 - y_1(t)^2)y_2(t)), \quad t \in [0, 0.55139], \qquad (5.3)\\ &y_1(0) = 2, \quad y_2(0) = -\frac{2}{3} + \frac{10}{81}\epsilon - \frac{292}{2187}\epsilon^2 - \frac{1814}{19683}\epsilon^3, \end{aligned}$$

and we choose $f_N = [y_2, 0]^T$, $f_S = [0, (-y_1 + (1 - y_1^2)y_2)/\epsilon]^T$, and $\epsilon = 10^{-6}$. Error is calculated by taking the absolute error between successive solutions at the final time. We comment that semi-implicit ARK methods (and hence IDC-ARK methods) in general were not designed to handle problems similar to Van der Pol's oscillator, but we include this example since it appears frequently in the literature. We refer to y_1 as the differential variable and y_2 as the algebraic variable, since, in the limit as $\epsilon \to 0$, (5.3) becomes a differential-algebraic equation that involves only the derivative of y_1.

Figure 5.5 shows the convergence of various IDC schemes for the stiff oscillator ($\epsilon = 10^{-6}$). Despite the stiffness, the differential variable, y_1, attains the full theoretical order of accuracy. The algebraic variable, y_2, on the other hand, displays order reduction that appears to depend on the order of the base scheme. That is, although the error still decreases with successive corrections, IDC-FBE methods degrade to first order convergence in y_2, while the IDC-ARK3BHR1 methods degrade to third order convergence in y_2. IDC-ARK3BHR methods have smaller error coefficients compared to IDC-FBE methods. It is not unusual for certain implicit RK and semi-implicit methods, to exhibit order reduction for very stiff problems (small ϵ) [16, 21, 25]. In some cases, additional conditions on the structure of an RK or ARK method may mitigate or remove order reduction [16, 6, 4, 5]. Since IDC methods have a similar structure to RK methods [8], it is reasonable to look for IDC methods that maintain full order of accuracy in stiff regimes. Furthermore, the third order IDC-FBE method displays no order reduction for large timesteps (Figure 5.5a), and interestingly, appears to be more accurate than the ARK3BHR1 (no IDC) method in this timestep range. This surprising result is more obvious in the efficiency study in Figure 5.6, and provides impetus for investigating whether any other IDC methods that maintain full theoretical order of accuracy in stiff situations may be constructed.



(a) VdP (5.3), $\epsilon = 10^{-6}$ (stiff)

(b) VdP (5.3), $\epsilon = 10^{-6}$ (stiff)

Fig. 5.5: *Convergence study of various IDC schemes applied to Van der Pol's oscillator with $\epsilon = 10^{-6}$, $T = 0.55139$ vs H. The differential variables, y_1 attain their theoretical order of accuracy, while the differential variable suffers from order reduction (Dotted lines indicate theoretical order of accuracy).*

The efficiency plot, Figure 5.6, is particularly interesting. It appears for the Van der Pol's oscillator, that IDC-FBE methods are more efficient at attaining solutions that are accurate to 10^{-8}, before the onset of order reduction. Once order reduction sets in, IDC-ARK methods are more efficient.

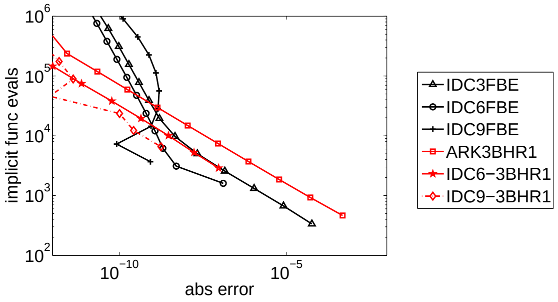


Fig. 5.6: *Efficiency studies of various IDC schemes applied to the stiff Van der Pol's oscillator ($\epsilon = 10^{-6}$). IDC-FBE methods are more efficient at attaining solutions that are accurate to 10^{-8}, before the onset of order reduction. Once order reduction sets in, IDC-ARK methods are more efficient.*

6. Concluding remarks

We have provided a general framework for the simple construction of arbitrary order IMEX methods. These semi-implicit IDC-ARK methods use ARK integrators as base schemes inside the IDC framework and can thus be constructed to higher order easily, with no need to consider the complicated order conditions that are typically required for the construction of ARK methods. High order IDC-ARK methods can be used to solve multiscale differential equations that involve disparate time scales more efficiently than popular ARK schemes. We have performed an analysis of the local truncation error of IDC-ARK methods, shown improved stability, and conducted numerical results showing order of convergence and improved efficiency. We are presently examining the use of asymptotic preserving ARK schemes (as in [27]) in the prediction and correction loops of IDC as a potential way to mitigate the issue of order reduction that arises with increased stiffness. We also have begun an investigation of embedded IDC methods, including IDC-ARK, and their implementation in an adaptive setting. We anticipate that adaptivity will provide additional efficiency gains over IDC-ARK.

Acknowledgements. This work was supported by IPAM, AFOSR grant number FA9550-07-1-0092 and NSF grant numbers DMS-0934568 and DMS-0914852.

REFERENCES

[1] A.L. Araújo, A. Murua, and J.M. Sanz-Serna, *Symplectic methods based on decompositions*, SIAM J. Numer. Anal., 34(5), 1926–1947, 1997.

[2] U.M. Ascher, S.J. Ruuth, and R.J. Spiteri, *Implicit-explicit Runge–Kutta methods for time-dependent partial differential equations*, Appl. Numer. Math., 25(2-3), 151–167, 1997. Special Issue on Time Integration (Amsterdam, 1996).

[3] U.M. Ascher, S.J. Ruuth, and B.T.R. Wetton, *Implicit-explicit methods for time-dependent partial differential equations*, SIAM J. Numer. Anal., 32(3), 797–823, 1995.

[4] S. Boscarino and G. Russo, *On the Uniform Accuracy of IMEX Runge-Kutta Schemes and Applications to Hyperbolic Systems with Relaxation*, Commun. to SIMAI Congress, 0(2), 2007.

[5] S. Boscarino and G. Russo, *On a class of uniformly accurate IMEX Runge–Kutta schemes and applications to hyperbolic systems with relaxation*, SIAM J. Sci. Comput., 31, 1926, 2009.

[6] S. Boscarino, *Error analysis of IMEX Runge-Kutta methods derived from differential-algebraic systems*, SIAM J. Numer. Anal., 45(4), 1600–1621 (electronic), 2007.

[7] S. Boscarino, *On an accurate third order implicit-explicit Runge–Kutta method for stiff problems*, Appl. Numer. Math., 59(7), 1515–1528, 2009.

[8] A. Christlieb, B. Ong, and J.M. Qiu, *Comments on high order integrators embedded within integral deferred correction methods*, Commun. Appl. Math. Comput. Sci., 4(1), 27–56, 2009.

[9] A. Christlieb, B. Ong, and J.M. Qiu, *Integral deferred correction methods constructed with high order Runge–Kutta integrators*, Math. Comput., 79, 761–783, 2010.

[10] G.J. Cooper and A. Sayfy, *Additive methods for the numerical solution of ordinary differential equations*, Math. Comput., 35(152), 1159–1172, 1980.

[11] G.J. Cooper and A. Sayfy, *Additive Runge–Kutta methods for stiff ordinary differential equations*, Math. Comput., 40(161), 207–218, 1983.

[12] A. Dutt, L. Greengard, and V. Rokhlin, *Spectral deferred correction methods for ordinary differential equations*, BIT, 40(2), 241–266, 2000.

[13] L. Greengard, *Spectral integration and two-point boundary value problems*, SIAM J. Numer. Anal., 28(4), 1071–1080, 1991.

[14] J. Haack and C. Hauck, *Oscillatory behavior of asymptotic-preserving splitting methods for a linear model of diffusive relaxation*, Los Alamos Report LA-UR-08-0571, 2008.

[15] E. Hairer, S.P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations. I*, Springer Series in Computational Mathematics, Springer-Verlag, Berlin, second edition, 8, 1993.

[16] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations. II*, Springer Series in Computational Mathematics, Springer-Verlag, Berlin, 14, 1991.

[17] J. Huang, J. Jia, and M. Minion, *Accelerating the convergence of spectral deferred correction methods*, J. Comput. Phys., 214(2), 633–656, 2006.

[18] J. Huang, J. Jia, and M. Minion, *Arbitrary order Krylov deferred correction methods for differential algebraic equations*, J. Comput. Phys., 221(2), 739–760, 2007.

[19] S. Jin, *Efficient asymptotic-preserving (AP) schemes for some multiscale kinetic equations*, SIAM J. Sci. Comput., 21(2), 441–454 (electronic), 1999.

[20] S. Jin, L. Pareschi, and G. Toscani, *Uniformly accurate diffusive relaxation schemes for multiscale transport equations*, SIAM J. Numer. Anal., 38(3), 913–936 (electronic), 2000.

[21] C.A. Kennedy and M.H. Carpenter, *Additive Runge–Kutta schemes for convection-diffusion-reaction equations*, Appl. Numer. Math., 44(1-2), 139–181, 2003.

[22] A.T. Layton, *On the choice of correctors for semi-implicit Picard deferred correction methods*, Appl. Numer. Math., 58(6), 845–858, 2008.

[23] A.T. Layton and M.L. Minion, *Implications of the choice of predictors for semi-implicit Picard integral deferred correction methods*, Commun. Appl. Math. Comput. Sci., 2, 1–34 (electronic), 2007.

[24] H. Liu and J. Zou, *Some new additive Runge–Kutta methods and their applications*, J. Comput. Appl. Math., 190(1-2), 74–98, 2006.

[25] M.L. Minion, *Semi-implicit spectral deferred correction methods for ordinary differential equations*, Commun. Math. Sci., 1(3), 471–500, 2003.

[26] L. Pareschi and G. Russo, *Implicit-explicit Runge–Kutta schemes for stiff systems of differential equations*, Recent trends in numerical analysis, Adv. Theory Comput. Math., Nova Sci. Publ., Huntington, NY, 3, 269–288, 2001.

[27] L. Pareschi and G. Russo, *Implicit-Explicit Runge–Kutta schemes and applications to hyperbolic systems with relaxation*, J. Sci. Comput., 25(1-2), 129–155, 2005.

[28] Y. Xia, Y. Xu, and C.W. Shu, *Efficient time discretization for local discontinuous Galerkin methods*, Discrete Contin. Dyn. Syst. Ser. B, 8(3), 677–693, 2007.