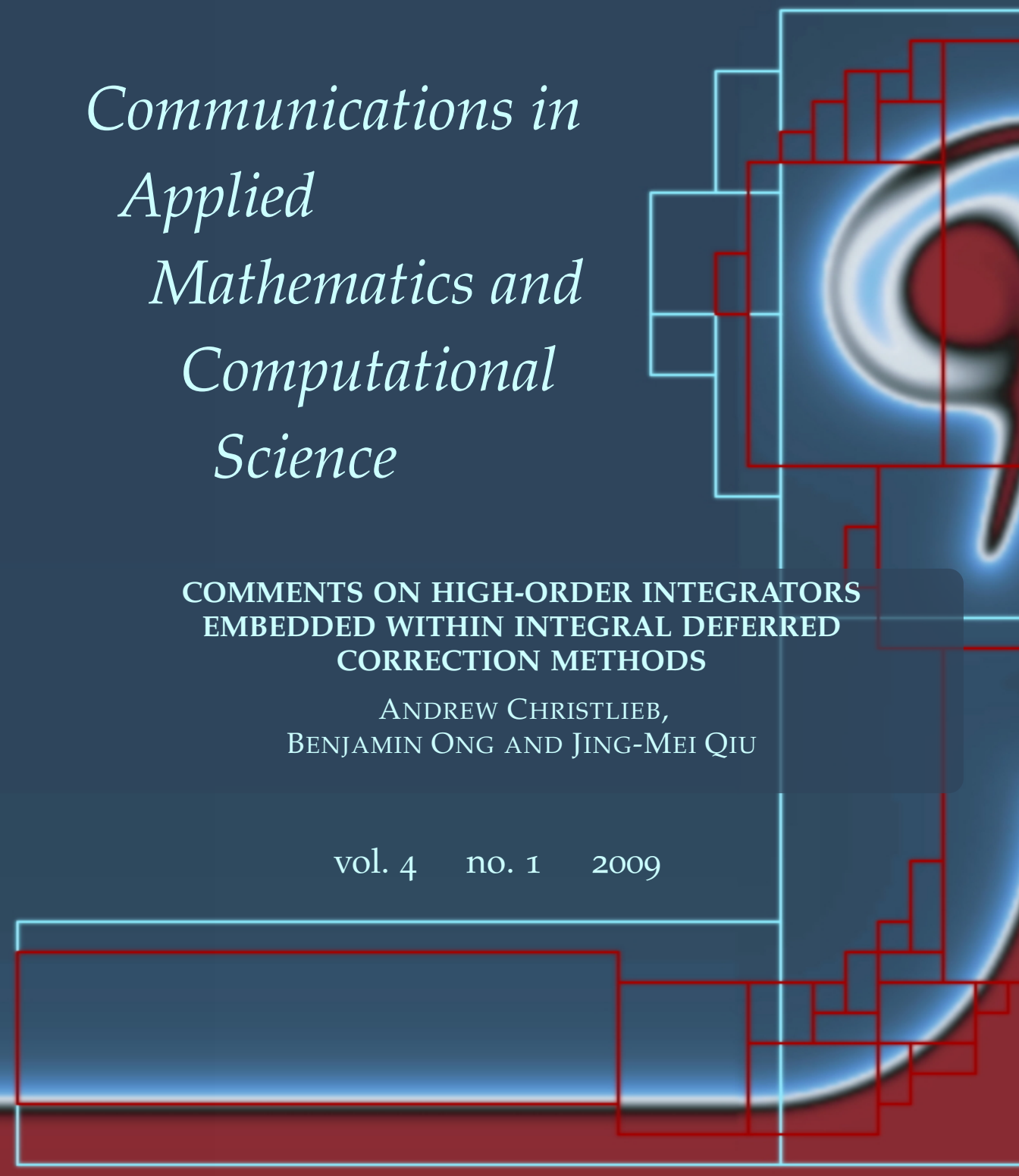
Communications in Applied Mathematics and Computational Science



COMMENTS ON HIGH-ORDER INTEGRATORS EMBEDDED WITHIN INTEGRAL DEFERRED CORRECTION METHODS

ANDREW CHRISTLIEB,
BENJAMIN ONG AND JING-MEI QIU

vol. 4 no. 1 2009

mathematical sciences publishers

COMM. APP. MATH. AND COMP. SCI.
Vol. 4, No. 1, 2009

COMMENTS ON HIGH-ORDER INTEGRATORS EMBEDDED WITHIN INTEGRAL DEFERRED CORRECTION METHODS

ANDREW CHRISTLIEB, BENJAMIN ONG AND JING-MEI QIU

A class of novel deferred correction methods, integral deferred correction (IDC) methods, is studied. This class of methods is an extension of ideas introduced by Dutt, Greengard and Rokhlin on spectral deferred correction (SDC) methods for solving ordinary differential equations (ODEs). The novel nature of this class of defect correction methods is that the correction of the defect is carried out using an accurate integral form of the residual instead of the more familiar differential form. As a family of methods, these schemes are capable of matching the efficiency of popular high-order RK methods.

The smoothness of the error vector associated with an IDC method is an important indicator of the order of convergence that can be expected from a scheme (Christlieb, Ong, and Qiu; Hansen and Strain; Skeel). It is demonstrated that embedding an r-th order integrator in the correction loop of an IDC method does not always result in an r-th order increase in accuracy. Examples include IDC methods constructed using non-self-starting multistep integrators, and IDC methods constructed using a nonuniform distribution of quadrature nodes.

Additionally, the integral deferred correction concept is reposed as a framework to generate high-order Runge–Kutta (RK) methods; specifically, we explain how the prediction and correction loops can be incorporated as stages of a high-order RK method. This alternate point of view allows us to utilize standard methods for quantifying the performance (efficiency, accuracy and stability) of integral deferred correction schemes. It is found that IDC schemes constructed using uniformly distributed nodes and high-order integrators are competitive in terms of efficiency with IDC schemes constructed using Gauss–Lobatto nodes and forward Euler integrators. With respect to regions of absolute stability, however, IDC methods constructed with uniformly distributed nodes and high-order integrators are far superior. It is observed that as the order of the embedded integrator increases, the stability region of the IDC method increases.

MSC2000: primary 65L05, 65L06, 65L20, 65L70; secondary 65B05.

Keywords: spectral deferred correction methods, integral deferred correction methods, Runge–Kutta methods, multistage methods, multistep methods, efficiency, stability, accuracy.

Research supported by Air Force Office of Scientific Research and Air Force Research Labs (Edward's and Kirtland). Grant nos. FA9550-07-1-0092 and FA9550-07-1-0144.

1. Introduction

In this paper, we construct and analyze a class of novel correction methods, integral deferred correction methods (IDC), which are constructed using high-order single-step and multistep integrators in the prediction and correction loops. IDC methods were first introduced in [4], and further developed and analyzed in [2; 5; 11; 18; 15]. Essentially, a deferred correction procedure is applied to an integral formulation of the error equation. This error equation is then solved by choosing a distribution of quadrature nodes and an integrator; these choices are crucial in determining the accuracy and stability of the scheme. For example, the selection of quadrature nodes is discussed in [15] and the selection of integrators for the prediction and correction loops are discussed in [16; 14; 18]. The authors in [11; 12] use Gaussian quadrature nodes and Krylov subspace methods to accelerate the convergence of the scheme. In [2], the advantages of using high-order RK integrators in SDC framework are shown analytically and numerically.

To study the properties of IDC schemes, the error arising from these schemes has to be analyzed. This error has two separate components: the first component is the error between the collocation solution on a given set of quadrature nodes and the exact solution [5; 11; 12]; this component limits the maximum achievable accuracy of IDC methods. The second component is the error that arises from using deferred correction iterations to approximate the collocation solution [2; 8; 9; 21]. In Section 3 of this paper, we focus on the second component of the abovementioned error. We will show that IDC methods constructed with p-th order multistage RK integrators (IDC-RK) and a uniform distribution of quadrature nodes give a p-th order increase in accuracy after each correction loop (under mild assumptions), whereas IDC-RK methods constructed with a nonuniform distribution of quadrature nodes do not give a p-th order increase ($p \geq 2$) after each correction loop. When multistep methods — for example, Adams–Bashforth (AB) methods — are used within an IDC method, the smoothness of the rescaled error vector prevents a high-order accuracy increase after each correction loop, regardless of the distribution of quadrature nodes.

In Section 4, we address a commonly perceived drawback of IDC methods: the additional computational overhead needed to implement these schemes. By formulating IDC-RK methods into an RK method, the local truncation error arising from IDC-RK schemes can be estimated. We show that a smaller truncation error offsets the computational overhead, making IDC methods constructed using uniformly distributed nodes and high-order integrators, as well as IDC methods constructed using Gaussian–Lobatto nodes and forward Euler integrators, competitive (in terms of efficiency) with known RK methods for eighth- and higher-order schemes. Additionally, the formulation of IDC-RK methods as an RK method gives a systematic way to generate arbitrary-order RK methods *without* solving

complicated order conditions; an added bonus is that the entries of the RK Butcher tableau [6] can be computed exactly using a symbolic manipulator. Accuracy plots are generated in Section 4.3, validating the error estimates while stability plots in Section 4.4 show that IDC methods offer a much larger stability region compared with known RK methods. In fact, as the order of the embedded integrator is increased, the stability region of an IDC method also increases. These superior stability regions are one of the promising features of IDC-RK methods.

This paper is organized into three main sections. In Section 2, a brief review of IDC methods is given. In Section 3, properties of IDC methods constructed using general high-order integrators and various distributions of quadrature nodes are given, along with some examples. IDC methods are then reformulated into high-order RK methods in Section 4, and a detailed comparison between IDC methods and RK methods is given. Section 5 contains the conclusion and closing remarks.

2. Review of IDC methods

This section is a review of IDC methods from [4]. Our discussion of these methods is based on notation introduced below. We consider an IVP consisting of a system of ODEs and initial conditions,

$$\begin{cases} y'(t) = f(t, y), & t \in [0, T], \\ y(0) = y_0. \end{cases} \tag{2-1}$$

The time domain, $[0, T]$, is discretized into intervals,

$$0 = t_1 < t_2 < \cdots < t_n < \cdots < t_N = T,$$

and each interval, $I_n = [t_n, t_{n+1}]$, is further discretized into subintervals,

$$t_n = t_{n,0} = t_{n,1} < \cdots < t_{n,m} < \cdots < t_{n,M} = t_{n+1}. \tag{2-2}$$

We refer to $t_{n,m}$ as quadrature nodes, whose index m runs from 0 to M.

An IDC method on each time interval $[t_n, t_{n+1}]$, described below, is iterated completely to define the starting value for the next interval, $[t_{n+1}, t_{n+2}]$. We drop the subscript n, so that $t_{n,m} =: t_m$ in (2-2), with the understanding that the IDC method is described for that one time interval.

- (prediction step) Use an r_0-th order numerical method to obtain a numerical solution, $\vec{\eta}^{[0]} = (\eta_0^{[0]}, \eta_1^{[0]}, \ldots, \eta_m^{[0]}, \ldots, \eta_M^{[0]})$, which is an r_0-th order approximation to $\vec{y} = (y_0, y_1, \ldots, y_m, \ldots, y_M)$, where $y_m = y(t_m)$ is the exact solution at t_m.

- (correction loop) Use the error function to improve the accuracy of the scheme at each iteration.
  For $k = 1, \ldots, k_l$ (k_l is number of correction steps)
  (1) Denote the *error function* from the $(k-1)$-st loop as

$$e^{(k-1)}(t) = y(t) - \eta^{(k-1)}(t), \tag{2-3}$$

  where $y(t)$ is the exact solution and $\eta^{(k-1)}(t)$ is an M-th degree polynomial interpolating $\vec{\eta}^{[k-1]}$. Note that the error function, $e^{(k-1)}(t)$, is not a polynomial in general.
  (2) Compute the *residual function*, $\epsilon^{(k-1)}(t) = (\eta^{(k-1)})'(t) - f(t, \eta^{(k-1)}(t))$. In the literature, the residual function is often called the pointwise, or differential defect.
  (3) Compute the *numerical error vector*, $\vec{\delta}^{[k]} = (\delta_0^{[k]}, \ldots, \delta_m^{[k]}, \ldots, \delta_M^{[k]})$, using an r_k-th order numerical method to discretize the integral form of the *error equation*,

$$\begin{aligned}\left(e^{(k-1)} + \int_0^t \epsilon^{(k-1)}(\tau)\, d\tau\right)'(t) &= f(t, \eta^{(k-1)}(t) + e^{(k-1)}(t)) - f(t, \eta^{(k-1)}(t)) \\ &\doteq F(t, e^{(k-1)}(t)),\end{aligned} \tag{2-4}$$

  where $F(t, e(t)) = f(t, \eta(t) + e(t)) - f(t, \eta(t))$, $\vec{\delta}^{[k]}$ is an r_k-th order approximation to

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

  and $e_m^{[k-1]} = e^{(k-1)}(t_m)$ is the value of the exact error function at t_m.
  (4) Update the numerical solution $\vec{\eta}^{[k]} = \vec{\eta}^{[k-1]} + \vec{\delta}^{[k]}$.

Notationally, superscripts with a round bracket, for example (k), denote a function, while superscripts with a square bracket, $[k]$, denote a vector at the k-th correction step. English letters are reserved for functions or vectors in the exact solution space, for example an exact solution $y(t)$ and an exact error function $e(t)$, while Greek letters denote functions or vectors in the numerical solution space, for example a numerical solution $\eta(t)$ and a numerical error function $\delta(t)$.

A forward Euler discretization of the error (2-4) gives

$$\delta_{m+1}^{[k]} = \delta_m^{[k]} + h_m(f(t_m, \eta_m^{[k-1]} + \delta_m^{[k]}) - f(t_m, \eta_m^{[k-1]})) - \int_{t_m}^{t_{m+1}} \epsilon^{(k-1)}(t)\, dt, \tag{2-5}$$

where $h_m = t_{m+1} - t_m$. Expanding the integral in (2-5),

$$\int_{t_m}^{t_{m+1}} \epsilon^{(k-1)}(t)\, dt = \left[\eta^{(k-1)}(t_{m+1}) - \eta^{(k-1)}(t_m)\right] - \int_{t_m}^{t_{m+1}} f(t, \eta^{(k-1)}(t))\, dt, \tag{2-6}$$

and substituting (2-6) into (2-5) results in

$$\eta_{m+1}^{[k]} = \eta_m^{[k]} + h_m\big[f(t_m, \eta_m^{[k-1]} + \delta_m^{[k]}) - f(t_m, \eta_m^{[k-1]})\big] + \int_{t_m}^{t_{m+1}} f(t, \eta^{[k-1]}(t))\,dt. \tag{2-7}$$

The integral in (2-7) can be evaluated using a Lagrange interpolant constructed from the function values, $\int_{t_m}^{t_{m+1}} L_{(\vec{t},\vec{f})}(\tau)\,d\tau$, where

$$L_{(\vec{t},\vec{f})}(\tau) = \sum_{m=0}^{M} \alpha_m(\tau) f_m, \qquad \text{with } \alpha_m(\tau) = \prod_{n\neq m} \frac{\tau - t_n}{t_m - t_n} \text{ and } f_m = f(t_m, \eta^{[k-1]}(t_m)). \tag{2-8}$$

Hence, (2-7) can also be written as

$$\eta_{m+1}^{[k]} = \eta_m^{[k]} + h_m\left[f(t_m, \eta_m^{[k-1]} + \delta_m^{[k]}) - f(t_m, \eta_m^{[k-1]})\right] + \sum_{j=0}^{M} S_{mj} f(t_j, \eta_j^{[k-1]}),$$

where

$$S_{mj} = \int_{t_m}^{t_{m+1}} \alpha_j(\tau)\,d\tau$$

are the elements of the so-called integration matrix.

IDC methods constructed using s-stage RK integrators (IDC-RKs) are more involved. We provide the following details for discretizing the error (2-4) for uniformly distributed quadrature nodes; generalization to nonuniformly distributed quadrature nodes is straightforward. Denoting by h the interval size for the uniformly distributed nodes and implementing an s-stage RK integrator to discretize (2-4) gives

$$k_1 = F(t_m, \delta_m^{[k-1]}), \tag{2-9a}$$

$$k_2 = F\left(t_m + c_2 h, \delta_m^{[k-1]} + h a_{2,1} k_1 - \int_{t_m}^{t_m + c_2 h} \epsilon^{(k-1)}(\tau)\,d\tau\right), \tag{2-9b}$$

$$k_3 = F\left(t_m + c_3 h, \delta_m^{[k-1]} + h(a_{3,1}k_1 + a_{3,2}k_2) - \int_{t_m}^{t_m + c_3 h} \epsilon^{(k-1)}(\tau)\,d\tau\right),$$

$$\vdots$$

$$k_s = F\left(t_m + c_s h, \delta_m^{[k-1]} + h\sum_{l=1}^{s-1} a_{s,l} k_l - \int_{t_m}^{t_m + c_s h} \epsilon^{(k-1)}(\tau)\,d\tau\right), \tag{2-9c}$$

$$\delta_{m+1}^{[k-1]} = \delta_m^{[k-1]} + h\sum_{l=1}^{s} b_l k_l - \int_{t_m}^{t_m + h} \epsilon^{(k-1)}(\tau)\,d\tau, \tag{2-9d}$$

where $A, \vec{b}, \vec{c}$ are conventional Butcher table entries [7] for an s-stage RK integrator:

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

Each RK stage, for example (2-9b),

$$k_2 = f\Big(t_m + c_2 h,\, \eta^{(k-1)}(t_m + c_2 h) + \delta_m^{[k-1]} + h a_{2,1} k_1 - \int_{t_m}^{t_m + c_2 h} \epsilon^{(k-1)}(\tau)\, d\tau\Big) - f\big(t_m + c_2 h,\, \eta^{(k-1)}(t_m + c_2 h)\big), \quad (2\text{-}10)$$

involves the integral of the residual function,

$$\int_{t_m}^{t_m + c_2 h} \epsilon^{(k-1)}(\tau)\, d\tau = \big[\eta^{(k-1)}(t_m + c_2 h) - \eta^{(k-1)}(t_m)\big] - \int_{t_m}^{t_m + c_2 h} f(t, \eta^{(k-1)}(t))\, dt, \quad (2\text{-}11)$$

where the integral in (2-11)

$$\int_{t_m}^{t_m + c_2 h} f(t, \eta^{(k-1)}(t))\, dt = \sum_{j=0}^{M} S_{m\cdot s+2,j} f(t_j, \eta_j^{[k-1]}),$$

can be evaluated using the integration matrix, $S_{m\cdot s+2,j} = \int_{t_m}^{t_m + c_2 h} \alpha_j(t)\, dt$. For future reference, the general expression for the entries of this expanded integration matrix is

$$S_{m\cdot s+l,j} = \begin{cases} \int_{t_m}^{t_m + c_l h} \alpha_j(t)\, dt, & l = 2, \ldots, s,\ m = 0, \ldots, M-1,\ j = 0, \ldots, M, \\ \int_{t_m}^{t_{m+1}} \alpha_j(t)\, dt, & l = 1,\ m = 0, \ldots, M-1,\ j = 0, \ldots, M. \end{cases} \quad (2\text{-}12)$$

We choose this expanded definition of the integration matrix so that IDC methods constructed with single step integrators can be formulated as a high-order RK method in Section 4.1. Using this definition of the integration matrix, (2-10) can be expressed as

$$k_2 = f\Big(t_m, +c_2 h,\, \eta_m^{[k]} + h a_{2,1} k_1 - \sum_{j=0}^{M} S_{m\cdot s+2,j} f(t_j, \eta_j^{[k-1]})\Big) - f\big(t_m + c_2 h,\, \eta^{(k-1)}(t_m + c_2 h)\big). \quad (2\text{-}13)$$

The term $f\big(t_m + c_2 h, \eta^{(k-1)}(t_m + c_2 h)\big)$ in (2-13) can be computed by evaluating the Lagrangian interpolant at the intermediate stage, $L_{(\vec{t},\vec{f})}(t_m + c_2 h)$. This can also

be written as

$$L_{(\vec{t},\vec{f})}(t_m + c_2 h) = \sum_{j=0}^{M} L_{m\cdot s+2,j} f(t_j, \eta_j^{[k-1]}),$$

where the entries of the interpolation matrix are given by

$$L_{m\cdot s+l,j} = \alpha_j(t_m + c_l h), \quad l = 1, \ldots, s, \quad m = 0, \ldots, M-1. \tag{2-14}$$

The remaining stages and their combinations are evaluated in a similar fashion. In Section 4.1, we systematically formulate IDC methods constructed using RK integrators as high-order RK methods.

We omit details for constructing IDC methods using multistep integrators (such as IDC-AB) because such schemes are not self-starting. We show in Section 3 that the obvious approach of using a high-order RK integrator to compute the first few steps results in an error vector which lacks sufficient smoothness; this lack of smoothness results in a poorer than desired accuracy increase after each correction loop.

3. IDC methods constructed using high-order integrators

In this section, we discuss the accuracy of IDC methods constructed using high-order integrators and various distributions of quadrature nodes. Specifically, IDC methods constructed using multistage RK methods are discussed in Section 3.2 for uniformly spaced quadrature nodes, and Section 3.3 for a nonuniform distribution. IDC methods constructed using high-order multistep methods are given in Section 3.4. The smoothness of the rescaled error vector measured in a discrete Sobolev norm is a crucial tool for both discussions; we review this concept in Section 3.1.

3.1. ***Mathematical preliminaries.*** Several analytical and numerical preliminaries are needed to analyze IDC methods. The smoothness of discrete data sets will be established, analog to the smoothness of functions; this idea of smoothness is used to analyze the error vectors. Let $f(t)$ be a function for $t \in [0, T]$, and denote the corresponding discrete data set,

$$(\vec{t}, \vec{f}) = \{(t_0, f_0), \ldots, (t_M, f_M)\}, \tag{3-1}$$

where

$$0 = t_0 < t_1 < t_2 < \ldots < t_M = H. \tag{3-2}$$

Definition 3.1 (Smoothness of a function). A function $f(t)$, $t \in [0, T]$, possesses S degrees of smoothness if $\|d^s f\|_\infty := \|\partial^s f/\partial t^s\|_\infty$ is bounded for $s = 0, 1, 2, \ldots, S$, where $\|f\|_\infty := \max_{t\in[0,T]} |f(t)|$.

Definition 3.2 (s-th degree spectral differentiation). Consider the discrete data set, $(\vec{t},\, \vec{f})$, defined in (3-1), and let $L_{(\vec{t},\vec{f})}(\tau)$ be the Lagrange interpolant described in (2-8). An s-th degree spectral differentiation is a linear mapping that maps $\vec{f}$ into

$$\overrightarrow{\hat{d}_s f}, \qquad \text{where } (\hat{d}_s f)_m = (\partial^s/\partial\tau^s) L_{(\vec{t},\vec{f})}(\tau)|_{\tau=t_m}.$$

This linear mapping can be represented by

$$\overrightarrow{\hat{d}_s f} = \hat{D}_s \cdot \vec{f},$$

where $\hat{D}_s \in \mathcal{R}^{(M+1)\times(M+1)}$ and $(\hat{D}_s)_{mn} = (\partial^s/\partial\tau^s) c_n(\tau)|_{\tau=t_m}$, $m, n = 0, \ldots, M$.

Remark 3.3. Given a distribution of quadrature nodes on $[0, 1]$, the spectral differentiation matrices, $\hat{D}_s^{[0,1]}$, $s = 1, \ldots, M$, have constant entries. If this distribution of quadrature nodes is rescaled from $[0, 1]$ to $[0, H]$, then the corresponding differentiation matrices are

$$\hat{D}_1 = \frac{1}{H}\hat{D}_1^{[0,1]} \quad \text{and} \quad \hat{D}_s = \left(\frac{1}{H}\right)^s \hat{D}_s^{[0,1]}.$$

Definition 3.4. The $(\hat{S}, \infty)$ Sobolev norm of a discrete data set $(\vec{t},\, \vec{f})$ is defined as

$$\|\vec{f}\|_{\hat{S},\infty} := \|\vec{f}\|_\infty + \sum_{s=1}^{S} \|\overrightarrow{\hat{d}_s f}\|_\infty = \|\vec{f}\|_\infty + \sum_{s=1}^{S} \|\hat{D}_s \cdot \vec{f}\|_\infty.$$

Definition 3.5 (Smoothness of a discrete date set). A discrete data set, (3-1), possesses $S \le M$ degrees of smoothness if $\|\vec{f}\|_{\hat{S},\infty}$ is bounded as $H \to 0$.

Remark 3.6. We emphasize that smoothness is a property of discrete data sets in the limit as $H \to 0$. We also impose $S \le M$, because

$$\overrightarrow{\hat{d}_S f} \equiv \vec{0},$$

for $S > M$. See [2] for a detailed discussion.

Example 3.7 (A discrete data set with only one degree of smoothness). Consider the discrete data set

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

The first derivative

$$\overrightarrow{\hat{d}_1 f} = \left(-\frac{4}{3}, \frac{10}{3}, 0, -\frac{10}{3}, \frac{4}{3}\right),$$

is bounded independent of H, while the second derivative

$$\overrightarrow{\hat{d}_2 f} = \left(\frac{272}{3H}, -\frac{16}{3H}, -\frac{112}{3H}, -\frac{16}{3H}, \frac{272}{3H}\right),$$

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

3.2. ***IDC methods constructed using RK integrators and uniformly spaced quadrature nodes.*** Integral deferred correction methods constructed using high-order RK integrators (IDC-RK) and uniformly spaced quadrature nodes boast superior accuracy and stability regions [2]. We restate the following theorem and lemmas from [2], which prove (under mild conditions) the accuracy of these IDC-RK methods. An example is provided to illustrate the main components of the theorem.

Theorem 3.8. *Let $y(t)$, the solution to the IVP* (2-1), *have at least $S \geq M+2$ degrees of smoothness in the continuous sense. Then, the local error for an IDC method constructed using $(M+1)$ uniformly distributed nodes, $(t_m = mh, m = 0, \ldots, M)$, an (r_0)-th order RK method in the prediction step and $(r_1, r_2, \ldots, r_{k_l})$-th order RK methods, is $\mathbb{O}(h^{(s_{k_l}+1)})$, where $s_{k_l} = \sum_{j=0}^{k_l} r_j \leq M+1$.*

The proof of Theorem 3.8 follows from the two lemmas below. Lemma 3.9 addresses the case $k = 0$, and Lemma 3.10 addresses the inductive argument. We emphasize that both lemmas not only bound the error vectors, but also guarantee sufficient smoothness in the prediction and correction steps.

Lemma 3.9. *(prediction step) Let $\vec{\eta}^{[0]} = (\eta_0^{[0]}, \ldots, \eta_m^{[0]}, \ldots, \eta_M^{[0]})$ be the numerical solution obtained after the prediction step. Then, the error vector $\vec{e}^{[0]} = \vec{y} - \vec{\eta}^{[0]}$ satisfies*

$$\|\vec{e}^{[0]}\|_\infty \sim \mathbb{O}(h^{r_0+1}),$$

and the rescaled error vector $\vec{\tilde{e}}^{[0]} = \frac{1}{h^{r_0}}\vec{e}^{[0]}$ has $\min(S-r_0, M)$ degrees of smoothness in the discrete sense.

Proof. We provide the following outline for a proof when a forward Euler integrator is used in the prediction step. Details for the more general case of using an RK integrator is provided in [2].

We drop the superscript [0] as there is no ambiguity. Since $\eta_{m+1} = \eta_m + hf(t_m, \eta_m)$, the error at t_{m+1}, $e_{m+1} = y_{m+1} - \eta_{m+1}$ satisfies

$$e_{m+1} = e_m + h(f(t_m, y_m) - f(t_m, \eta_m)) + \sum_{i=2}^{S-1} \frac{h^i}{i!} y^{(i)}(t_m) + \mathbb{O}(h^S),$$

where we have performed a Taylor expansion of y_{m+1} about $t = t_m$. Let $u_m = f(t_m, y_m) - f(t_m, \eta_m)$, and

$$r_m = \frac{h^2}{2!} y^{(2)}(t_m) + \cdots + \frac{h^{S-1}}{(S-1)!} y^{(S-1)}(t_m).$$

Notice that

$$u_m = e_m f_y(t_m, y_m) + \cdots + \frac{(-1)^{S-1}(e_m)^{S-2}}{(S-2)!} f_{y^{S-2}}(t_m, y_m) + \mathbb{O}((e_m)^{S-1}),$$

where we have performed a Taylor expansion of $f(t, \eta_m)$ about $y = y_m$. We are now ready to bound $\|\vec{e}^{[0]}\|_\infty$ by induction. By definition, $e_0 = 0$, so certainly, $e_0 \sim \mathbb{O}(h^2)$. Assume that $e_m \sim \mathbb{O}(h^2)$. Since $u_m \sim \mathbb{O}(e_m) \sim \mathbb{O}(h^2)$, we have

$$e_{m+1} = e_m + hu_m + r_m + \mathbb{O}(h^S) \sim \mathbb{O}(h^2),$$

which completes the inductive proof that $\|\vec{e}\|_\infty \sim \mathbb{O}(h^2)$. Note that the inductive proof was with respect to m, the index of the grid points.

To prove the smoothness of the rescaled error vector, we will again use an inductive approach, but this time with respect to s, the degree of smoothness. First, note that a discrete differentiation of the rescaled error vector gives

$$(d_1\tilde{e})_m = \frac{\tilde{e}_{m+1} - \tilde{e}_m}{h} = \tilde{u}_m + \frac{r_m}{h^2} + \mathbb{O}(h^{S-2}), \tag{3-3}$$

where

$$\tilde{u}_m = \frac{u_m}{h} = \sum_{i=1}^{S-2} (-1)^{i+1} \frac{h^{i-1}}{i!} f_{y^i}(t_m, y_m)(\tilde{e}_m)^i + \mathbb{O}(h^{2S-3}).$$

We are now ready to prove that $\vec{\tilde{e}}$ has M degrees of smoothness by induction. Since $\|\vec{\tilde{e}}\|_\infty \sim \mathbb{O}(h)$, $\vec{\tilde{e}}$ has at least zero degrees of smoothness in the discrete sense. Assume that $\vec{\tilde{e}}$ has $s \le M-1$ 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.

Since f_{y^i} has $(S-i-1)$ 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 $(S-i-1)$ degrees of smoothness in the discrete sense. Consequently, $h^{i-1}\overrightarrow{f_{y^i}}$ has $(S-2)$ degrees of smoothness, which implies that $\vec{\tilde{u}}$ has $\min(S-2, s)$ degrees of smoothness. Similarly, $\vec{r}/h^2$ has $(S-2)$ degrees of smoothness in the discrete sense. Hence $\overrightarrow{d_1\tilde{e}}$ has s degrees of smoothness $\implies$ $\vec{\tilde{e}}$ has $(s+1)$ degrees of smoothness. Since this argument holds for $S \ge M+2$, we can conclude that $\vec{\tilde{e}}$ has M degrees of smoothness. □

Lemma 3.10 (Correction step). *Suppose after the $(k-1)$-st correction loop the error vector satisfies $\vec{e}^{[k-1]} \sim \mathbb{O}(h^{s_{k-1}+1})$ and the rescaled error vector*

$$\vec{\tilde{e}}^{[k-1]} = \frac{1}{h^{s_{k-1}}} \vec{e}^{[k-1]}$$

has $M+1-s_{k-1}$ degrees of smoothness in the discrete sense. Then, after the k-th $(k < k_l)$ correction loop the updated error vector satisfies

$$\|\vec{e}^{[k]}\|_\infty \sim \mathbb{O}(h^{s_k+1}),$$

and the rescaled error vector

$$\vec{\tilde{e}}^{[k]} = \frac{1}{h^{s_k}}\vec{e}^{[k]}$$

has $M+1-s_k$ degrees of smoothness in the discrete sense.

The proof is similar in spirit to the proof of Lemma 3.9 and is omitted for brevity.

Example 3.11. Consider the IVP

$$y'(t) = y(t); \quad y(0) = 1. \tag{3-4}$$

We solve IVP (3-4) with an IDC method constructed using six uniformly spaced quadrature nodes and the second-order trapezoidal RK method in the prediction and correction loops. Let H be the interval size and $h = \frac{H}{5}$ be the subinterval size. Computing the Taylor expansion of the numerical solution about $t = 0$ with $\mathbb{O}(h^7)$ truncation error, the rescaled error vectors are

$$\begin{aligned}\vec{\tilde{e}}^{[0]} = \Big\{&0, \frac{h}{750} + \frac{h^2}{15{,}000} + \frac{h^3}{375{,}000} + \frac{h^4}{11{,}250{,}000},\\ &\frac{h}{375} + \frac{h^2}{1{,}500} + \frac{4h^3}{46{,}875} + \frac{4h^4}{703{,}125}, \frac{h}{250} + \frac{9h^2}{5{,}000} + \frac{51h^3}{125{,}000} + \frac{71h^4}{1{,}250{,}000},\\ &\frac{2h}{375} + \frac{13h^2}{3{,}750} + \frac{53h^3}{46{,}875} + \frac{166h^4}{703{,}125}, \frac{h}{150} + \frac{17h^2}{3000} + \frac{181h^3}{75{,}000} + \frac{301h^4}{450{,}000}\Big\} + \mathbb{O}(h^5),\end{aligned}$$

$$\begin{aligned}\vec{\tilde{e}}^{[1]} = \Big\{&0, \frac{h}{225{,}000} - \frac{h^2}{4{,}500{,}000}, \frac{h}{112{,}500} + \frac{7h^2}{2{,}250{,}000},\\ &\frac{h}{75{,}000} + \frac{h^2}{100{,}000}, \frac{h}{56{,}250} + \frac{23h^2}{1{,}125{,}000}, \frac{h}{45{,}000} + \frac{86111h^2}{250{,}000}\Big\} + \mathbb{O}(h^3),\end{aligned}$$

$$\vec{\tilde{e}}^{[2]} = \mathbb{O}(h).$$

As postulated by the lemmas, the rescaled error vector $\vec{\tilde{e}}^{[0]}$ has five degrees of smoothness, $\vec{\tilde{e}}^{[1]}$ has three degrees of smoothness, and $\vec{\tilde{e}}^{[3]}$ has one degree of smoothness. Table 1 gives the error and order of the implemented IDC method after the prediction step and each correction loop. As expected, second-order convergence is observed after the prediction loop, fourth-order convergence is observed after one correction loop, and sixth-order convergence is observed after the second correction loop.

	1 loop of RK2		2 loops of RK2		3 loops of RK2	
steps	error	order	error	order	error	order
5	7.03E-4	–	1.06E-7	–	5.91E-11	–
10	1.79E-4	1.98	6.36E-9	4.07	9.55E-13	5.95
15	7.97E-5	1.99	1.24E-9	4.04	8.26E-14	6.04
20	4.50E-5	1.99	3.88E-10	4.03	1.20E-14	6.71
25	2.88E-5	1.99	1.59E-10	4.02	4.44E-16	14.77

Table 1. Example 3.11: IDC6-RK2, the sixth-order IDC method constructed using six uniformly distributed quadrature nodes and the trapezoidal RK2, is used to solve IVP (3-4). The error at $T = 1$ is measured after the prediction loop (1 loop of RK2), first correction loop (2 loops of RK2) and second correction loop (3 loops of RK2). The corresponding order of convergence is calculated.

3.3. ***IDC Methods constructed using RK integrators and nonuniform distributions of quadrature nodes.*** One might consider constructing an IDC method using a nonuniform distribution of quadrature nodes, such as Gaussian–Lobatto [1] (or Gaussian–Radau or Gaussian) nodes, because their collocation solution can achieve $2M$ ($(2M+1)$ or $(2M+2)$) orders of accuracy with $M+1$ nodes. Consequently, one would expect that the computational effort for an IDC scheme constructed with Gaussian–Lobatto points should be a fraction of that for an equivalent scheme using a uniform distribution of nodes.

However, when high-order integrators are applied to the error equation, the lack of smoothness of the rescaled error vector associated with such IDC methods destroys the high-order accuracy increase. Consequently, there is little advantage to constructing IDC-RK integrators using a nonuniform distribution of quadrature nodes. However, when considering IDC method with low-order integrators, such as forward/backward Euler, nonuniform quadrature points, such as Gaussian points, might be advantageous because of the reduced sensitivity to interpolation error, and a better conditioned interpolation/integration matrix. This is best illustrated by the following examples. In Example 3.12, we consider IDC methods constructed using the trapezoidal RK2 method and quadrature nodes with linearly increasing interval sizes. In Example 3.13, we consider an IDC method constructed using Gaussian–Lobatto quadrature nodes and the trapezoidal RK2 method.

Example 3.12 (Linearly increasing interval sizes). We solve IVP (3-4) numerically with an IDC method constructed using six quadrature nodes distributed smoothly,

though nonuniformly, with each interval satisfying

$$t_m - t_{m-1} = mh, \quad h = \frac{2H}{(M+1)M}, \quad m = 1, \ldots, M, \tag{3-5}$$

where H is the interval size. The trapezoidal RK2 integrator is applied in the prediction and correction loops. Computing the Taylor expansion of the numerical solution about $t = 0$ with $\mathbb{O}(h^7)$ truncation error, the rescaled error vector after the prediction step satisfies

$$\begin{aligned}\vec{\tilde{e}}^{[0]} = \Big\{ & 0, \frac{h}{20{,}250} + \frac{h^2}{1{,}215{,}000} + \frac{h^3}{91{,}125{,}000} + \frac{h^4}{8{,}201{,}250{,}000}, \\ & \frac{h}{2{,}250} + \frac{19h^2}{405{,}000} + \frac{h^3}{375{,}000} + \frac{h^4}{11{,}250{,}000}, \\ & \frac{2h}{1{,}125} + \frac{19h^2}{40{,}500} + \frac{161h^3}{2{,}531{,}250} + \frac{67h^4}{12{,}656{,}250}, \\ & \frac{2h}{405} + \frac{1469h^2}{607{,}500} + \frac{547h^3}{911{,}250} + \frac{31h^4}{328{,}050}, \\ & \frac{h}{90} + \frac{3521h^2}{405{,}000} + \frac{463h^3}{135{,}000} + \frac{707h^4}{810{,}000} \Big\} + \mathbb{O}(h^5).\end{aligned}$$

It can be checked by Definition 3.5 that $\vec{\tilde{e}}^{[0]}$ has one and only one degree of smoothness. Since $\vec{\tilde{e}}^{[0]}$ has only one degree of smoothness in the discrete sense, only one order increase in accuracy is guaranteed after the first correction loop, even when a high-order RK method is applied. By computing the rescaled error vector after subsequent correction loops, one can show that only one order increase in accuracy per loop can be guaranteed until the maximum order is achieved.

This is illustrated in Table 2, which gives the error and order of the IDC method using the quadrature nodes distributed according to (3-5). Second-order accuracy is

	RK2 pred.		1 corr. loop		2 corr. loops		3 corr. loops	
steps	error	order	error	order	error	order	error	order
5	1.16E-3	–	2.16E-6	–	2.84E-9	–	2.3 E-10	–
10	2.96E-4	1.97	3.03E-7	2.83	2.77E-10	3.36	4.02E-12	5.86
15	1.32E-4	1.98	9.29E-8	2.91	6.12E-11	3.73	3.75E-13	5.85
20	7.47E-5	1.99	3.99E-8	2.94	2.04E-11	3.82	7.01E-14	5.83
25	4.79E-5	1.99	2.06E-8	2.95	8.58E-12	3.87	1.82E-14	6.05

Table 2. Example 3.12: The error at $T = 1$ and the order of the implemented IDC-RK2 method are tabulated after the prediction loop, first correction loop, second correction loop, etc. The quadrature nodes are distributed as described in (3-5).

observed after the RK2 prediction loop. Then, only third- and fourth-order accuracy are observed after the first and second RK2 correction loop as per the discussion above. Sixth-order accuracy is observed after the third RK2 correction loop.

Example 3.13. (Gaussian–Lobatto quadrature nodes) IVP (3-4) is solved numerically with an IDC method constructed using six Gaussian–Lobatto quadrature nodes given by

$$\begin{aligned} t_0 &= 0, & t_3 &= \left(1+\sqrt{\tfrac{1}{21}(7-2\sqrt{7})}\right)\frac{H}{2}, \\ t_1 &= \left(1-\sqrt{\tfrac{1}{21}(7+2\sqrt{7})}\right)\frac{H}{2}, & t_4 &= \left(1+\sqrt{\tfrac{1}{21}(7+2\sqrt{7})}\right)\frac{H}{2}, \\ t_2 &= \left(1-\sqrt{\tfrac{1}{21}(7-2\sqrt{7})}\right)\frac{H}{2}, & t_5 &= H, \end{aligned} \tag{3-6}$$

where H is the interval size. RK2 is applied in the prediction and correction loops. Computing the Taylor expansion of the numerical solution about $t = 0$ with $\mathbb{O}(h^7)$ truncation error, the rescaled error vector after the prediction step satisfies

$$\begin{aligned} \vec{\tilde{e}}^{[0]} = \{&0,\ 0.00027018h + 0.00000793h^2 + 0.00000019h^3, \\ &0.00257165h + 0.00048115h^2 + 0.00004858h^3 + 0.00000289h^4, \\ &0.00643924h + 0.00287267h^2 + 0.00065172h^3 + 0.00008973h^4 \\ &0.00874071h + 0.00603452h^2 + 0.00209675h^3 + 0.00046357h^4, \\ &0.00901089h + 0.00730769h^2 + 0.00297836h^3 + 0.00078573h^4\} + \mathbb{O}(h^5). \end{aligned}$$

It can be checked by Definition 3.5 that $\vec{\tilde{e}}^{[0]}$ has one and only one degree of smoothness. Since $\vec{\tilde{e}}^{[0]}$ has only one degree of smoothness in the discrete sense, one order increase in accuracy is guaranteed after the first correction loop. Computing the rescaled error vectors after subsequent correction loops, one can show that the smoothness constraint guarantees only one order accuracy increase per loop until the maximum order is increased.

In Table 3, we show the error and rate of convergence of up to nine loops of RK2, to demonstrate that the maximum $2M$ order can be achieved when using Gauss–Lobatto points. The expected second/fourth/sixth-order convergence is observed after one/three/five loops of RK2 steps, respectively. However, fourth/sixth-order accuracy is observed after two/four RK2 loops. This discrepancy can be explained by carefully studying the error vector after the first and third RK2 correction loops.

$$\begin{aligned} \vec{\tilde{e}}^{[1]} = \{&0,\ -0.00000716h^4 + \mathbb{O}(h^5),\ -0.00004306h^4 + \mathbb{O}(h^5), \\ &-0.00004306h^4 + \mathbb{O}(h^5),\ -0.00000716h^4 + \mathbb{O}(h^5),\ 0.00004950h^5 + \mathbb{O}(h^6)\}. \end{aligned}$$

	RK-2 prediction		1 RK-2 corr. loop		2 RK-2 corr. loop	
steps	error	order	error	order	error	order
5	8.28E-3	–	2.13E-5	–	1.23E-6	–
10	2.19E-3	1.92	1.43E-6	3.90	8.51E-8	3.85
15	9.92E-4	1.95	2.89E-7	3.94	1.73E-8	3.93
20	5.63E-4	1.98	9.25E-8	3.96	5.55E-9	3.95
25	3.63E-4	1.97	3.82E-8	3.97	2.29E-9	3.97

	3 RK-2 corr. loops		4 RK-2 corr. loops	
steps	error	order	error	order
5	1.42E-8	–	2.25E-9	–
10	2.49E-10	5.84	3.86E-11	5.87
15	2.27E-11	5.91	3.48E-12	5.93
20	4.11E-12	5.94	6.27E-13	5.96
25	1.09E-12	5.95	1.64E-13	6.01

	6 loops of RK-2		7 loops of RK-2		8 loops of RK-2	
steps	error	order	error	order	error	order
3	8.89E-7	–	4.27E-8	–	1.59E-9	–
6	5.20E-9	7.41	3.70E-11	10.17	4.61E-12	8.43
9	2.29E-10	7.71	2.14E-13	12.70	1.03E-13	9.38
12	2.42E-11	7.81	9.59E-14	2.79	8.88E-15	8.52

Table 3. Example 3.13: The error and order of an IDC method used for solving IVP (3-4) using Gaussian–Lobatto quadrature nodes are tabulated. The error is computed at $T = 1$ after the RK-2 prediction loop, first RK-2 correction loop, second RK-2 correction loop, etc. Note that constructing an IDC method with six Gaussian–Lobatto points allows for up to tenth-order accuracy. Coarse steps are taken for the IDC method constructed with five correction loops or more because of machine precision limitations.

$\mathbb{O}(h^5)$ is observed in the last element of $\vec{e}^{[1]}$, corresponding to the results in the second column of Table 3; third-order, not fourth, is actually consistently achieved at the interior nodes after the first correction loop. After the second correction loop, fourth-order convergence is consistently achieved everywhere. Similarly, the error vector after the third correction loop,

$$\vec{e}^{[3]} = \{0, -0.000000004h^6 + \mathbb{O}(h^7), 0.00000003h^6 + \mathbb{O}(h^7),$$
$$0.00000003h^6 + \mathbb{O}(h^7), -0.000000004h^6 + \mathbb{O}(h^7), -0.00000002h^7 + \mathbb{O}(h^8)\},$$

is not consistently sixth-order at the interior nodes either.

3.4. ***IDC methods constructed using high-order multistep methods.*** A general linear p-step multistep method for solving IVP (2-1) is of the form

$$\begin{aligned} y_{n+p} + a_{p-1}y_{n+p-1} + a_{p-2}y_{n+p-2} + \cdots + a_0 y_n \\ = h(b_p f(t_{n+p}, y_{n+p}) + b_{p-1} f(t_{n+p-1}, y_{n+p-1}) + \cdots + b_0 f(t_n, y_n)). \end{aligned}$$

Examples of popular multistep methods include the explicit Adams–Bashforth methods (AB), implicit Adams–Moulton methods (AM), and backward differential formulas (BDF). For example,

$$y_{n+1} = y_n + h\left(\tfrac{3}{2} f(t_n, y_n) - \tfrac{1}{2} f(t_{n-1}, y_{n-1})\right), \tag{AB}$$

$$y_{n+1} = y_n + h\left(\tfrac{1}{2} f(t_n, y_n) + \tfrac{1}{2} f(t_{n+1}, y_{n+1})\right), \tag{AM}$$

$$y_{n+1} = \tfrac{4}{3} y_n - \tfrac{1}{3} y_{n-1} + \tfrac{2}{3} h f(t_{n+1}, y_{n+1}). \tag{BDF}$$

Most multistep methods are not self-starting; typically, a high-order integrator, such as an RK integrator, is used to compute the first few steps. In the next example, we show that using an RK-2 integrator as a starter ruins the desired accuracy increase which is possible with a high-order multistep method. Similar comments are also made in [14], in which a variable starting technique is suggested in conjunction with the multistep methods. Although this technique showed some promise in test examples (see [14, Figure 2]), we did not observe high-order increase in the correction loops of our numerical experiments.

Example 3.14. Consider an IDC method that is constructed using six uniformly distributed quadrature nodes and three loops of a second-order AB method in the prediction and correction steps (an RK-2 method is used to start the multistep method as necessary). We use this method to solve IVP (3-4). Let H denote the interval size for a single step of the IDC method, and $h = \frac{H}{5}$ the subinterval size. The numerical results are in Table 4 show the inconsistent accuracy increase after the first correction loop. Computing the Taylor expansion of the numerical solution about $t = 0$ with $\mathbb{O}(h^7)$ truncation error, the rescaled error vector satisfies

$$\begin{aligned} \vec{\tilde{e}}^{[0]} = \Big\{ & 0, \frac{h}{750} + \frac{h^2}{15{,}000} + \frac{h^3}{375{,}000} + \frac{h^4}{11{,}250{,}000}, \\ & \frac{7h}{1500} + \frac{2h^2}{1{,}875} + \frac{4h^3}{46{,}875} + \frac{4h^4}{703{,}125}, \frac{h}{125} + \frac{9h^2}{2{,}500} + \frac{81h^3}{125{,}000} + \frac{81h^4}{1{,}250{,}000}, \\ & \frac{17h}{1500} + \frac{14h^2}{1{,}875} + \frac{1643h^3}{750{,}000} + \frac{256h^4}{703{,}125}, \frac{11h}{750} + \frac{19h^2}{1500} + \frac{191h^3}{37{,}500} + \frac{5521h^4}{4{,}500{,}000} \Big\} + \mathbb{O}(h^5). \end{aligned}$$

	AB-2 pred.		1 AB-2 corr. loop		2 AB-2 corr. loop	
steps	error	order	error	order	error	order
5	1.55E-3	–	4.41E-6	–	7.74E-9	–
10	3.93E-4	1.98	4.56E-7	3.27	1.80E-10	5.43
15	1.76E-4	1.99	1.26E-7	3.18	1.88E-11	5.57
20	9.90E-5	1.99	5.10E-8	3.14	3.50E-12	5.84
25	6.34E-5	1.99	2.55E-8	3.11	8.61E-13	6.28

Table 4. Error and order of convergence for an IDC method constructed using a 2-step multistep method (AB) and uniformly distributed quadrature nodes. The error and rate of convergence are computed at $T = 1$.

By Definition 3.5, $\vec{\tilde{e}}^{[0]}$ has only one degree of smoothness since the leading term in

$$\overrightarrow{\hat{d}_2 e^{[0]}},$$

by Definition 3.2, is $\mathbb{O}(\frac{1}{h})$. Since $\vec{\tilde{e}}^{[0]}$ has no more than one degree of smoothness in the discrete sense, this limits the increase in convergence rate for IDC methods, although a high-order method is applied in the correction steps.

4. Comparisons between IDC and RK methods

IDC methods constructed using single-step integrators can be formulated into arbitrary high-order RK methods. This is of particular interest because RK methods are traditionally constructed by satisfying order conditions [6]; the number of order conditions to be satisfied grows exponentially as the order increases, making it difficult, if not impossible, to solve for the nodes, weights, and stage weights exactly. Here, we address how IDC methods constructed with RK integrators and uniformly distributed nodes can be formulated as a high-order RK method whose nodes, weights, and stage weights are known exactly. In Section 4.1, we describe the Butcher tableau structure for IDC-FE methods formulated as a high-order RK method. Then, we bound the local truncation error arising from IDC methods formulated as a high-order RK method; this bound on the local truncation error can be used to give an estimate for the global error, in essence, proving the convergence of IDC methods. The efficiency of IDC methods is then compared with known RK methods in Section 4.2. In general, known RK methods are more efficient than IDC methods for low-order schemes. For high-order schemes, comparable efficiency is observed numerically; the accuracy regions in Section 4.3 agree qualitatively

with the efficiency comparisons. In Section 4.4, we show that IDC methods offer a much larger stability region compared with known RK methods. Additionally, as the order of the embedded integrator is increased, the stability region of an IDC method also increases.

4.1. ***Constructing RK methods using an IDC-FE scheme.*** The following two points of view are equivalent: RK methods can be constructed using IDC ideas, or an IDC method can be reformulated as an RK method. The node points c_j, weights b_k and stage weights a_{jk} are often conveniently expressed in a Butcher tableau format using matrix A, and vectors b and c.

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

Here, we illustrate the Butcher tableau structure of IDC methods constructed using forward Euler time integrators. The algorithm is easily generalized for generating IDC-RK Butcher tableaus. In this section, we adopt a Matlab-style notation in our algorithms, where $A(j,:)$ denotes the j-th row of matrix A, and $A(:,j)$ denotes the j-th column of matrix A.

Proposition 4.1. *An IDC method constructed using* $(M+1)$ *quadrature nodes and* (k_l+1) *prediction/correction iterations of an* s*-stage RK method, can be reformulated as an* $((k_l+1)\cdot s\cdot M)$*-stage RK method.*

For example, an IDC method constructed with four quadrature nodes, $(M=3)$, a forward Euler prediction $(s=1)$, and three correction loops $(k_l=3)$, can be reformulated as a 12-stage RK method. Let's examine the structure of this $((k_l+1)\cdot s\cdot M)$-stage RK method. Suppose $(M+1)$ quadrature nodes, notated as before in (3-2),

$$0=t_0<t_1<t_2<\ldots<t_M=H,$$

have subinterval sizes

$$h_m=t_m-t_{m-1},\quad m=1,\ldots,M.$$

Then the prediction step of the IDC method constructed using forward Euler updates can be formulated as an RK method with the following Butcher array format:

$$\begin{array}{c|cccc} t_0 & & & & \\ t_1 & h_1 & & & \\ t_2 & h_1 & h_2 & & \\ \vdots & \vdots & & \ddots & \\ t_{M-1} & h_1 & h_2 & \ldots & h_{M-1} \\ \hline & h_1 & h_2 & \ldots & h_{M-1} \quad h_M \end{array}$$

We label components of the above Butcher tableau conventionally:

$$\begin{array}{c|c} \vec{c}_1 & A_1 \\ \hline & \vec{b}_1^T \end{array}$$

The first correction loop can now be included (2-7). The updated Butcher tableau takes the form

$$\begin{array}{c|cc} \vec{c}_1 & A_1 & Z \\ \vec{c}_2 & P_1 & A_2 \\ \hline & \vec{d}_1^T & \vec{b}_2^T \end{array},$$

where Z is a $M \times M$ matrix of zeros,

$$\vec{c}_2 = \left[t_M, t_1, t_2, \ldots, t_{M-1}\right]^T,$$

$$P_1 = \begin{bmatrix} h_1 & h_2 & h_3 & \ldots & h_M \\ \tilde{S}_{10} & \tilde{S}_{11} & \tilde{S}_{12} & \ldots & \tilde{S}_{1,M-1} \\ \tilde{S}_{20} & (\tilde{S}_{21} - h_2) & \tilde{S}_{22} & \ldots & \tilde{S}_{2,M-1} \\ \tilde{S}_{30} & (\tilde{S}_{31} - h_2) & (\tilde{S}_{32} - h_3) & \ldots & \tilde{S}_{3,M-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \tilde{S}_{M-1,0} & (\tilde{S}_{M-1,1} - h_2) & (\tilde{S}_{M-1,2} - h_3) & \ldots & \tilde{S}_{M-1,M-1} \end{bmatrix},$$

where the terms

$$\tilde{S}_{ij} = \begin{cases} S_{ij} & i = 1, \quad j = 0, \ldots, M, \\ S_{ij} + S_{i-1,j} & i = 2, \ldots, M, \quad j = 0, \ldots, M, \end{cases}$$

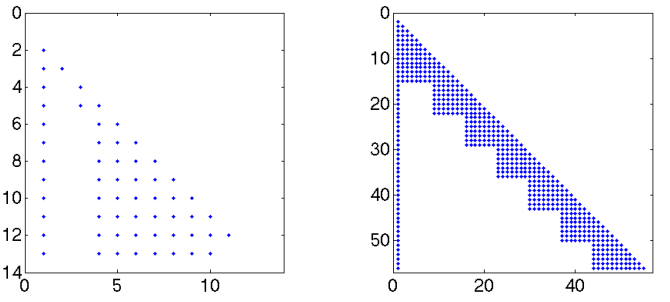
are the sums of the integration matrix defined in (2-12),

$$A_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & \ldots & 0 \\ \tilde{S}_{1M} & 0 & 0 & 0 & \ldots & 0 \\ \tilde{S}_{2M} & h_2 & 0 & 0 & \ldots & 0 \\ \tilde{S}_{3M} & h_2 & h_3 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & 0 \\ \tilde{S}_{M-1,M} & h_2 & h_3 & \ldots & h_{M-1} & 0 \end{bmatrix},$$

and

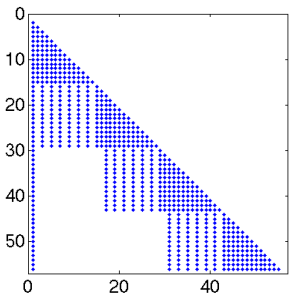
$$\vec{d}_1 = \left[\tilde{S}_{M0}, (\tilde{S}_{M1} - h_2), (\tilde{S}_{M2} - h_3), \ldots, (\tilde{S}_{M,M-1} - h_M)\right]^T,$$
$$\vec{b}_2 = \left[\tilde{S}_{MM}, h_2, h_3, \ldots, h_M\right]^T.$$

Subsequent correction steps can be added into a Butcher tableau format in a similar fashion. This results in a distinct block structure, since at the k-th correction loop,

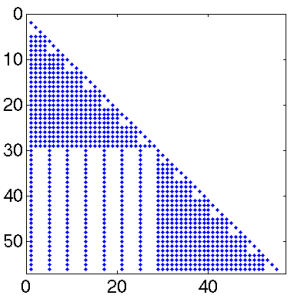


(a) RK8 [20]

(b) IDC8-FE



(c) IDC8-RK2



(d) IDC8-RK4

Figure 1. The sparsity structure of an RK8 [20] Butcher tableau and various IDC8 Butcher tableaus. The block structure for the IDC weights are evident; for example, IDC8-RK2 shows the prediction loop, and the subsequent three correction loops.

only the initial value $f(t, y_0)$, the previous approximations $f(t, \eta^{[k-1]})$, and the current iterates $f(t, \eta^{[k]})$ are used.

This block structure is more easily seen in Figure 1, where uniformly spaced quadrature nodes are used to construct various IDC schemes. For future reference, we adopt the following notation to denote our IDC schemes: IDCn-RKp denotes an n-th order IDC scheme constructed using p-th order RK integrators. To construct an n-th order IDC scheme, either $(n+1)$ uniformly distributed quadrature nodes, or $(\lceil \frac{n}{2} \rceil + 1)$ Gauss–Lobatto nodes are used. The only time we distinguish between using uniformly distributed and Gauss–Lobatto nodes is when forward Euler integrators are used to construct the IDC scheme; in all other cases, we use uniform nodes to achieve the order of accuracy desired. In Figure 1, IDC8-FE denotes an eighth-order IDC scheme constructed using forward Euler updates, IDC8-RK2 denotes an eighth-order IDC scheme constructed using a trapezoidal RK2 scheme for the prediction and correction steps, and IDC8-RK4 denotes an eighth-order

IDC scheme constructed using an RK4 predicition and correction loop. Several observations can be made presently. The number of stages for the IDC methods are consistent with Proposition 4.1. Specifically, IDC8-FE has $M = 7, s = 2, k_l = 7$, resulting in 56 overall stages. IDC8-RK2 has $M = 7, s = 2, k_l = 3$, and IDC8-RK4 has $M = 7, s = 4, k_l = 1$, both resulting in 56 overall stages. The sparsity structure of the Butcher tableau which arises from the prediction and correction steps is also evident. For example, IDC8-RK2 shows the prediction step and three subsequent correction steps.

It is important to note that the stage weights, a_{ij}, can be computed *exactly* using a symbolic manipulator such as Maple or Mathematica. This contrasts with most other optimization schemes for generating RK methods, where the coefficients have to be approximated numerically. For eighth- or lower-order schemes, computing the stage weights to double precision is sufficient. From numerical experiments, it seems that ninth-order IDC schemes (or higher) require quad precision or better.

4.2. ***Efficiency comparison.*** In order to compare how various p-th order RK methods stack up against each other, a quantitative measure is the so-called efficiency: how much computational effort is required to obtain a certain error tolerance. To make this measurement, we need to review the computational effort of IDC/RK methods, as well as bound the local truncation error (LTE).

In solving IVP (2-1), the evaluation of $f(t, y)$ is usually the most computationally expensive component. Hence, we will use the number of function evaluations, n_{fe}, (or equivalently, the number of stages of an RK method), as a measure of the computational effort. Recall that an IDC method constructed using $(M + 1)$ quadrature nodes, k_l correction loops, and an s-stage RK integrator requires $((M-1)\cdot(k_l+1)\cdot s)$ function evaluations (stages). For a p-th order IDC method, this corresponds to at least $p\,(p-1)$ function evaluations when p uniformly spaced quadrature nodes are used, and at least $(\frac{p}{2}-1)\,p$ function evaluations when $\frac{p}{2}$ Gaussian nodes are used. Compared to classically known p-th order RK methods which involve s_p stages, IDC methods require significantly more function evaluations per iteration. This is offset, however, by the smaller LTE that arises from IDC methods.

The LTE which arises from solving $y' = f(t, y)$ can be computed by taking the appropriate Taylor expansions of the scheme. For a p-th order method [10] the LTE can be expressed as

$$\text{LTE} = \sum_{i=p+1}^{\infty} h^i \left(\sum_{j=1}^{\lambda_i} \alpha_{ij} D_{ij} \right).$$

Here, h is the interval size, D_{ij} are the elementary differentials (sums of products of partial derivatives of $f(t, y)$), α_{ij} are the truncation error coefficients, and λ_i denotes the number of elementary differentials of order $\mathbb{O}(h^i)$. Consequently, a very

Method	# Stages	LTE	Efficiency
RK4 (classical)	4	1.0417E-2	1
IDC4-FE (unif)	12	7.716 E-4	1.78
IDC4-FE (gauss)	8	3.906 E-3	1.64
IDC4-RK2 (unif)	12	5.144 E-4	1.64
SDC4-RK2 (gauss)	8	2.6042E-3	1.52
RK6 [20]	9	8.4369E-7	1
RK6 [17]	7	1.455 E-2	3.13
IDC6-FE (unif)	30	1.0000E-6	3.42
IDC6-FE (gauss)	18	5.5014E-5	3.63
IDC6-RK2 (unif)	30	8.8889E-7	3.36
IDC6-RK3 (unif)	30	4.4444E-7	3.04
RK8 [20]	13	3.8872E-6	1
RK8 [3]	11	2.1957E-5	1.03
IDC8-FE (unif)	56	6.776 E-10	1.65
IDC8-FE (gauss)	32	1.181 E-7	1.67
IDC8-RK2 (unif)	56	5.0193E-10	1.59
IDC8-RK4 (unif)	56	3.0689E-11	1.17

Table 5. A comparison of classical RK methods and IDC methods. The second column lists the effective number of stages, the third column lists a bound on the LTE coefficients, and the last column is the computed efficiency between the respective orders. Eighth-order IDC methods are almost as efficient as an RK8 method. The LTE for twelfth-order methods is not presented due to machine precision restrictions. Also, since three Gauss–Lobatto nodes are in fact uniformly spaced, $x = \{0, 0.5, 1\}$, we are able to generate a fourth-order IDC scheme using three Gauss–Lobatto nodes and RK2 integrators for the prediction and correction loops. Note than an efficiency close to 1 is optimal.

crude bound for the LTE, if h is sufficiently small, is

$$\text{LTE} \leq h^{p+1} \cdot \lambda_{p+1} \cdot \|\alpha_{p+1,j}\|_\infty \cdot \|D_{p+1,j}\|_\infty.$$

We note that this local error estimate gives a bound on the global error [6], proving the convergence of IDC-RK methods.

Now, consider the LTE for two p-th order RK methods,

$$(\text{LTE})_1 = c_1 h^{p+1} \cdot (\lambda_{p+1} \|D_{p+1,j}\|_\infty),$$
$$(\text{LTE})_2 = c_2 h^{p+1} \cdot (\lambda_{p+1} \|D_{p+1,j}\|_\infty).$$

If both LTEs are bounded by the same tolerance ϵ, then the largest step size that will satisfy this tolerance for both methods is

$$h_1 = \left(\frac{\epsilon}{\beta c_1}\right)^{1/(p+1)}, \quad h_2 = \left(\frac{\epsilon}{\beta c_2}\right)^{1/(p+1)},$$

respectively, where $\beta = (\lambda_{p+1}\|D_{p+1,j}\|_\infty)$. If method one is computed in s_1 stages and method two is computed in s_2 stages, then the total amount of work done by each methods is s_i/h_i, since $1/h_i$ is the number of iterations required, and s_i is the cost per iteration. A measure of efficiency is then given by the ratio of the amount of work done:

$$\text{efficiency} = \frac{s_2/h_2}{s_1/h_1} = \frac{s_2}{s_1}\left(\frac{c_2}{c_1}\right)^{1/(p+1)}. \tag{4-1}$$

Using (4-1), the efficiencies for various IDC methods are computed and compared to classically known RK methods. In Table 5, we list the number of stages for each method, a bound on the LTE (using a code provided in [10]), and the computed efficiency. An efficiency close to 1 is optimal while an efficiency of 1.5 means it takes 50% more work to achieve the same error tolerance. We compute the LTEs for eighth- and lower-order schemes to avoid machine precision issues. (As mentioned in the previous section, the accuracy increase is lost when the nodes are nonuniformly spaced; thus, IDC schemes constructed using Gaussian nodes and high-order integrators are in general less efficient than IDC schemes constructed using uniformly spaced nodes and high-order integrators. Consequently, the efficiency analysis for other IDC schemes using Gaussian nodes is not presented.)

Two observations are in order: first, that the efficiency of IDC schemes improves as the order of the embedded integrator is increased; and second, that IDC8 schemes are comparable, in terms of efficiency, to RK8 schemes. Although we are unable to accurately compute the LTE for higher than eighth-order schemes, we show that twelfth-order IDC schemes are comparable in terms of efficiency to known RK-12 schemes by generating their accuracy regions, defined in Section 4.3.

4.3. ***Accuracy region.*** A more visual way to compare these IDC methods is to plot the accuracy region for each method. Specifically, the following IVP,

$$y'(t) = \lambda y(t), \quad y(0) = 1, \tag{4-2}$$

is solved for various λ's in the complex plane. A contour plot of the resulting error at $T = 1$ is called the accuracy region. Figures 2–5 show the accuracy regions for classical RK and IDC methods. Consistent with the efficiency analysis, the IDC4 and IDC6 schemes perform poorly in contrast with classical RK methods. IDC8-RK4 has a comparable accuracy region with RK8. The accuracy regions for IDC12 methods are plotted, even though the efficiency is not computed in

the previous section. It appears that IDC12-RK3 and IDC12-RK4 might be more efficient than classically known RK12 schemes. One should also note that the accuracy regions for IDC methods increase in area as the order of the embedded integrator is increased.

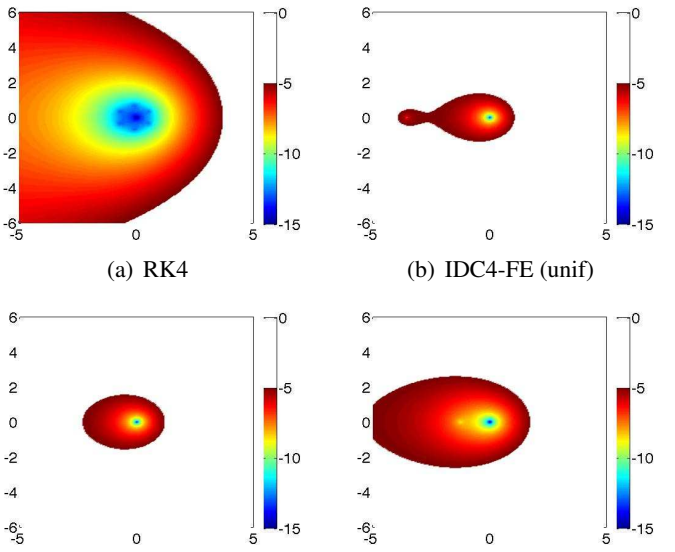
4.4. ***Stability region.*** Another way to quantitatively compare RK and IDC methods is to perform a linear stability analysis of the methods, and identify restrictions on the possible time steps. The linear stability region, S, is the subset of the complex plane, $\mathbb{C}$, satisfying

$$S = \{\lambda : \mathrm{Am}(\lambda) \leq 1\},$$

where $\mathrm{Am}(\lambda)$, the amplification factor for a numerical method, can be interpreted as the numerical solution of IVP (4-2)

$$y'(t) = \lambda y(t), \quad y(0) = 1,$$

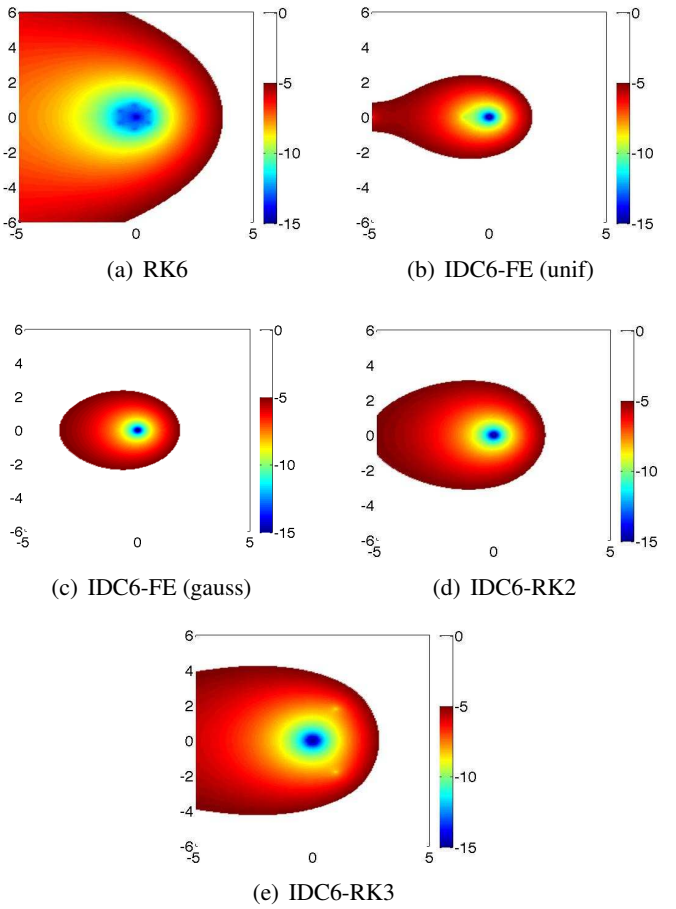
after a time step of size one. To quantify the size of these linear stability regions, we measure the *linear stability radius*, the real interval $\{z : \mathrm{Re}(z) \in S\}$, and the maximum imaginary value, $\sup |\mathrm{Im}(z)|$, $z \in S$.



(a) RK4 (b) IDC4-FE (unif)

(c) IDC4-FE (gauss) (d) IDC4-RK2

Figure 2. Accuracy plots for various fourth-order RK and IDC methods. Each plot was generated after 48 function evaluations. The RK4 method is vastly superior to the IDC methods.



(a) RK6 (b) IDC6-FE (unif)

(c) IDC6-FE (gauss) (d) IDC6-RK2

(e) IDC6-RK3

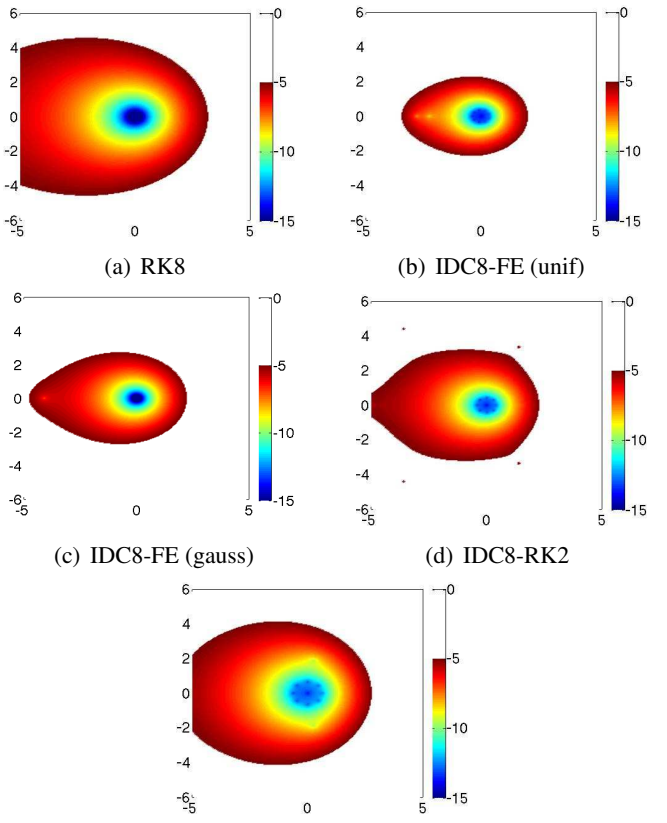
Figure 3. Accuracy plots for various sixth-order RK and IDC methods are generated using ≈ 60 function evaluations. The RK6 method has a larger accuracy region. Also, observe that the accuracy regions for the IDC methods increase with the order of the embedded integrator.

Definition 4.2. The linear stability radius is defined to be the radius of the largest disc that can fit inside the stability region,

$$\rho = \sup \{r : D(r) \in S\},$$

where $D(r)$ is the disc $D(r) = \{z \in \mathbb{C} : |z + r| \leq r\}$.

This measure of the stability region is argued to be a good compromise between stretching the stability region in the real and in the imaginary directions [13; 19].



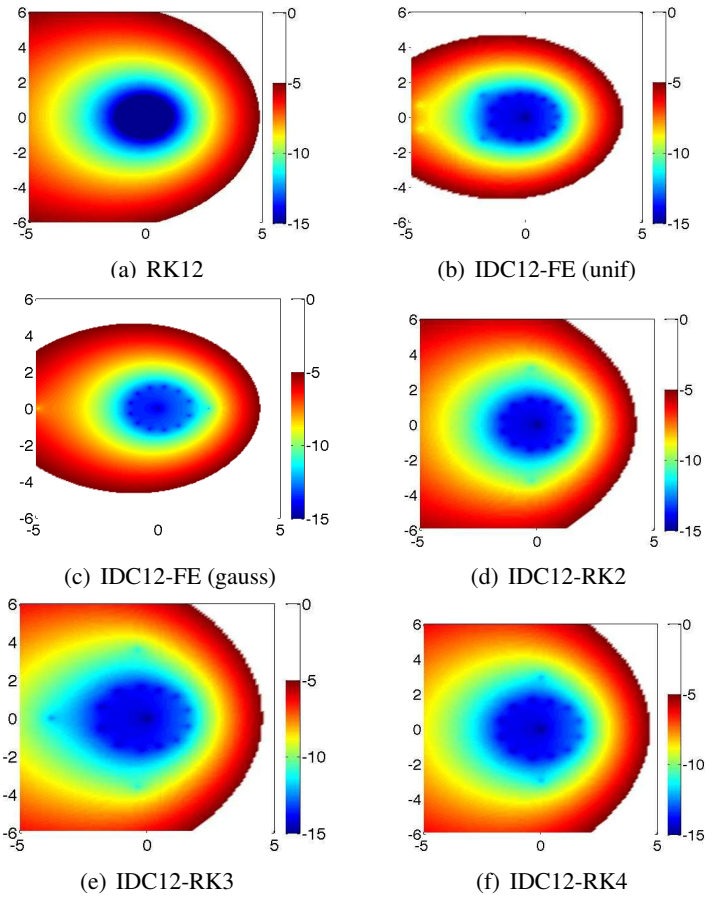
(a) RK8 (b) IDC8-FE (unif)

(c) IDC8-FE (gauss) (d) IDC8-RK2

(e) IDC8-RK4

Figure 4. The accuracy plots for various eighth-order RK and IDC methods are generated after ≈ 56 function evaluations. Notice that accuracy regions for IDC methods get larger as the order of the low-order integrator is increased. The accuracy region for IDC8-RK4 is comparable to the accuracy region for RK8, which is consistent with the efficiency analysis.

We plot the stability regions for various IDC and RK methods in Figure 6. In all cases, p-th order IDC methods offer a larger stability region in contrast with classically derived p-th order RK methods. Additionally, the stability regions of IDC methods increase with the order of the integrator used to construct the scheme; for example, IDC8-RK4 has a larger stability region than IDC8-RK2. Quantitative comparison of the stability regions are given in Table 6.



(a) RK12 (b) IDC12-FE (unif)

(c) IDC12-FE (gauss) (d) IDC12-RK2

(e) IDC12-RK3 (f) IDC12-RK4

Figure 5. The accuracy plots for various twelfth-order RK and IDC methods are generated after ≈ 132 function evaluations. The accuracy region for IDC12-RK3 and IDC12-RK4 appear larger than the classically known RK12 scheme.

5. Conclusion

In this paper, we studied a class of novel correction methods, IDC methods, constructed using high-order integrators within the prediction and correction loops. It was also shown that the accuracy of an IDC method is closely related to the smoothness of its error vector. Unlike IDC methods constructed with uniform quadrature points, the order of accuracy for IDC methods constructed with a general nonuniform distribution of quadrature nodes does not increase by r orders if an r-th order RK correction step is applied; for multistep methods, the accuracy of an IDC method depends heavily on the starting method. Finally, IDC methods are

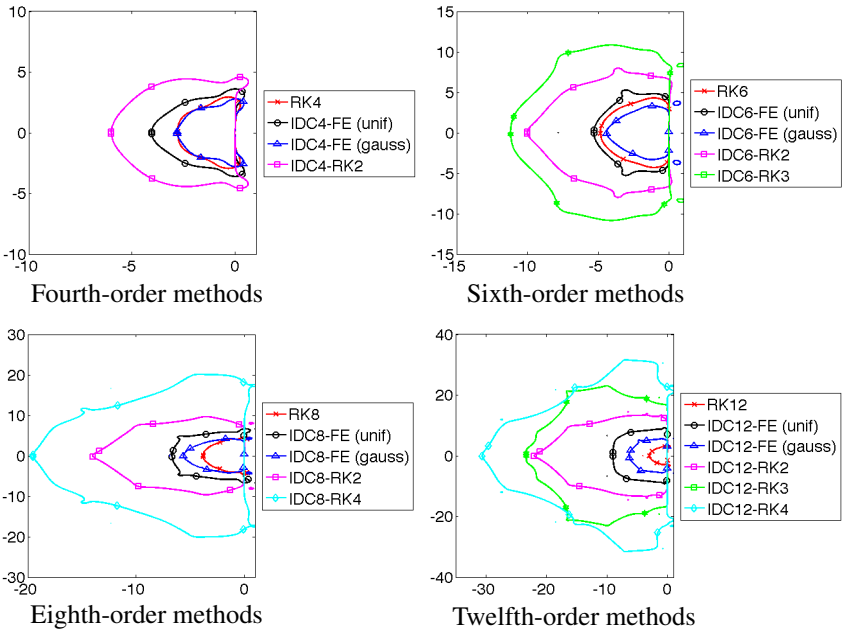


Figure 6. Stability regions for fourth-, sixth-, eighth- and twelfth-order IDC methods. The stability regions of IDC methods are larger than that of the RK method. Additionally, the stability region of the IDC methods increase with the order of the integrator used to construct the scheme.

viewed as a means for generating high-order RK methods. The efficiency, stability, and accuracy of IDC methods are compared with RK methods. As a family of methods, these IDC schemes are capable of matching the efficiency of optimized high-order RK methods. Additionally, superior regions of absolute stability are observed for IDC methods constructed using high order integrators.

Present studies and analyses are being conducted on IDC methods constructed using diagonally implicit Runge–Kutta integrators and IDC methods constructed using additive Runge–Kutta integrators.

References

[1] Claudio Canuto, M. Yousuff Hussaini, Alfio Quarteroni, and Thomas A. Zang, *Spectral methods in fluid dynamics*, Springer Series in Computational Physics, Springer, New York, 1988. MR 89m:76004 Zbl 0658.76001

[2] Andrew Christlieb, Benjamin Ong, and Jing-Mei Qiu, *Integral deferred correction methods constructed with high order Runge–Kutta integrators*, Math. Comp., accepted.

[3] A. R. Curtis, *An eighth order Runge–Kutta process with eleven function evaluations per step*, Numer. Math. **16** (1970), 268–277. MR 42 #5444 Zbl 0194.18902

Method	$\bar{\rho}$	Real Interval	Max Im
RK4 (classical)	1.39	$[-2.78, 0.24]$	2.93
IDC4-FE (unif)	2.00	$[-4.05, 0.43]$	3.60
IDC4-FE (gauss)	1.40	$[-2.81, 0.41]$	2.79
IDC4-RK2 (unif)	3.00	$[-6.00, 0.63]$	4.57
RK6 [20]	2.43	$[-4.85, 0.00]$	4.30
IDC6-FE (unif)	2.66	$[-5.32, 0.01]$	5.27
IDC6-FE (gauss)	2.14	$[-4.42, 0.00]$	3.32
IDC6-RK2 (unif)	4.76	$[-10.00, 0.17]$	7.98
IDC6-RK3 (unif)	5.59	$[-11.18, 0.16]$	10.84
RK8 [20]	1.90	$[-3.80, 0.33]$	4.66
IDC8-FE (unif)	3.33	$[-6.65, 0.54]$	6.66
IDC8-FE (gauss)	2.78	$[-6.92, 0.00]$	4.23
IDC8-RK2 (unif)	6.58	$[-14.0, \ 0.83]$	9.64
IDC8-RK4 (unif)	9.61	$[-19.49, 1.14]$	20.09
RK12	1.51	$[-3.00, 0.00]$	2.75
IDC12-FE (unif)	4.60	$[-9.01, 0.00]$	9.09
IDC12-FE (gauss)	3.34	$[-7.20, 0.25]$	4.97
IDC12-RK2 (unif)	9.94	$[-23.00, 0.00]$	13.47
IDC12-RK3 (unif)	11.45	$[-23.25, 0.00]$	23.02
IDC12-RK4 (unif)	14.92	$[-30.63, 1.05]$	31.61

Table 6. A table quantifying the stability regions of RK and IDC methods. For completeness, we list the linear stability radius ρ, the real interval, and the maximum imaginary value.

[4] Alok Dutt, Leslie Greengard, and Vladimir Rokhlin, *Spectral deferred correction methods for ordinary differential equations*, BIT **40** (2000), 241–266. MR 2001e:65104 Zbl 0959.65084

[5] Thomas Hagstrom and Ruhai Zhou, *On the spectral deferred correction of splitting methods for initial value problems*, Commun. Appl. Math. Comput. Sci. **1** (2006), 169–205. MR2008k: 65131

[6] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving ordinary differential equations, i: nonstiff problems*, 2nd ed., Springer Series in Computational Mathematics, no. 8, Springer, Berlin, 1993. MR 94c:65005 Zbl 0789.65048

[7] E. Hairer and G. Wanner, *Solving ordinary differential equations, ii: stiff and differential-algebraic problems*, 2nd ed., Springer Series in Computational Mathematics, no. 14, Springer, Berlin, 1996. MR 97m:65007 Zbl 0859.65067

[8] A. Hansen and J. Strain, *On the order of deferred correction*, preprint.

[9] ———, *Convergence theory for spectral deferred correction*, preprint, 2005.

[10] M. E. Hosea, *A new recurrence for computing Runge–Kutta truncation error coefficients*, SIAM J. Numer. Anal. **32** (1995), no. 6, 1989–2001. MR 96m:65073 Zbl 0841.65071

[11] Jingfang Huang, Jun Jia, and Michael Minion, *Accelerating the convergence of spectral deferred correction methods*, J. Comput. Phys. **214** (2006), no. 2, 633–656. MR 2006k:65173 Zbl 1094.65066

[12] ———, *Arbitrary order Krylov deferred correction methods for differential algebraic equations*, J. Comput. Phys. **221** (2007), no. 2, 739–760. MR 2008a:65134 Zbl 1110.65076

[13] Rolf Jeltsch and Olavi Nevanlinna, *Largest disk of stability of explicit Runge–Kutta methods*, BIT **18** (1978), no. 4, 500–502. MR 80b:65099 Zbl 0399.65051

[14] Anita T. Layton, *On the choice of correctors for semi-implicit Picard deferred correction methods*, Appl. Numer. Math. **58** (2008), no. 6, 845–858. MR 2420621 Zbl 1143.65057

[15] Anita T. Layton and Michael L. Minion, *Implications of the choice of quadrature nodes for Picard integral deferred corrections methods for ordinary differential equations*, BIT **45** (2005), no. 2, 341–373. MR 2006h:65087 Zbl 1078.65552

[16] ———, *Implications of the choice of predictors for semi-implicit Picard integral deferred correction methods*, Commun. Appl. Math. Comput. Sci. **2** (2007), 1–34. MR 2008e:65252 Zbl 1131.65059

[17] H. A. Luther, *An explicit sixth-order Runge–Kutta formula*, Math. Comp. **22** (1968), no. 102, 434–436. Zbl 0155.20402

[18] Michael L. Minion, *Semi-implicit spectral deferred correction methods for ordinary differential equations*, Commun. Math. Sci. **1** (2003), no. 3, 471–500. MR 2005f:65085 Zbl 1088.65556

[19] Brynjulf Owren and Kristian Seip, *Some stability results for explicit Runge–Kutta methods*, BIT **30** (1990), no. 4, 700–706. MR 91m:65203 Zbl 0718.65061

[20] James Verner, *High order Runge–Kutta methods*.

[21] Yinhua Xia, Yan Xu, and Chi-Wang Shu, *Efficient time discretization for local discontinuous Galerkin methods*, Discrete Contin. Dyn. Syst. Ser. B **8** (2007), 677–693. MR 2008e:65307 Zbl 1141.65076

Received November 13, 2008. Revised January 20, 2009.

ANDREW CHRISTLIEB: christlieb@math.msu.edu
Michigan State University, Department of Mathematics, D304 Wells Hall, East Lansing, MI 48824-1027, United States

BENJAMIN ONG: bwo@math.msu.edu
Michigan State University, Department of Mathematics, D304 Wells Hall, East Lansing, MI 48824-1027, United States

JING-MEI QIU: jingqiu@mines.edu
Mathematical and Computer Science, Colorado School of Mines, Golden, CO, 80401, United States