DEFERRED CORRECTION METHODS FOR ORDINARY DIFFERENTIAL EQUATIONS*

BENJAMIN W. ONG† AND RAYMOND J. SPITERI‡

Abstract. Deferred correction is a well-established method for incrementally increasing the order of accuracy of a numerical solution to a set of ordinary differential equations. Because implementations of deferred corrections can be pipelined, multi-core computing has increased the importance of deferred correction methods in practice, especially in the context of solving initial-value problems. In this paper, we review the theoretical underpinnings of deferred correction methods, specifically classical deferred correction, spectral deferred correction, and integral deferred correction. We highlight some non-traditional nuances of their implementations, including the choice of quadrature points, interpolants, and combinations of discretization methods, in a unified notation. We analyze how time-integration methods based on deferred correction can be effective solvers on modern computer architectures and demonstrate their performance on a diverse set of examples.

Key words. ordinary differential equations, initial-value problems, deferred correction, parallel computing, method of lines, multi-core computing

AMS subject classifications. 65B05, 65L05, 65L06, 65M20, 65Y05

1. Introduction.

Deferred correction (DC) methods are well-established methods for the construction of high-order approximations to the solution of differential equations based on lower-order numerical methods by a process of iterated corrections. The general idea is that a numerical solution to an initial-value problem (IVP) for a system of ordinary differential equations (ODEs) is computed and then subsequently refined by solving related IVPs. Under suitable assumptions, this process can be repeated to produce solutions with an arbitrarily high order of accuracy.

DC methods were originally proposed by Fox in [14] as *difference correction methods*. Fox notes that derivatives may be expressed as infinite series of differences, and thus the common finite-difference approximations to derivatives are truncated versions of these series. These difference correction methods compute a solution, then calculate a correction term of higher-order differences, which is then used to calculate a new solution. This correction turns out to be an estimate of the local discretization error [39]. These methods allow the improvement of accuracy of finite-difference solutions without increasing the complexity of the algebraic systems required for the solution. However, they require calculation of solution values outside the interval of integration, and high-order differences typically suffer from significant cancellation, leading to non-negligible round-off errors. In [14], deferred correction is applied to boundary-value problems (BVPs) for ODEs and eigenvalue problems for ODEs and partial differential equations (PDEs). The approach is applied to IVPs by Fox and Goodwin in [15]. Pereyra generalizes deferred correction to functional equations in [34] and considers specific examples in the solution of BVPs in [35].

Zadunaisky discusses in [46, 47] a method for estimating the error in a numerical solution to an IVP or BVP, an idea that now forms the basis for *defect correction*. Given a numerical solution $\widetilde{\mathbf{y}}(t)$ to an IVP, a neighbouring problem is constructed for which $\widetilde{\mathbf{y}}(t)$ is the exact solution. A numerical solution for the neighbouring problem

*This research was partially supported by the National Sciences and Engineering Research Council of Canada (NSERC).

†Michigan Technological University, Houghton, MI, 49931 (ongbw@mtu.edu)

‡University of Saskatchewan, Department of Computer Science, 176 Thorvaldson Building, 110 Science Place, Saskatoon, Saskatchewan, S7N 5C9, Canada. (spiteri@cs.usask.ca)

is then computed, and because the exact solution $\widetilde{\mathbf{y}}(t)$ to this problem is known by construction, its error can be calculated exactly. This error is then used as an estimate of the error in $\widetilde{\mathbf{y}}(t)$ and hence $\widetilde{\mathbf{y}}(t)$ can be corrected. The method requires the interpolation of the numerical solution and is prone to problems when using large equi-spaced grids. Such problems are mitigated by dividing the interval of integration into several pieces with small numbers of points in each [47]. In [42], Stetter generalizes this technique as the basis for an iterative defect correction method, in which the error approximation is added to the numerical solution to form a new solution, and the process of finding a neighbouring problem is repeated.

Skeel in [39] reviews the history of DC methods and related methods up to the time of its publication. DC methods have been extensively applied to IVPs [15, 13, 8, 9, 10, 20] and BVPs [14, 34, 35] for ODEs, initial-boundary value problems for PDEs [14, 23, 37], differential-algebraic equations [37, 22], and eigenvalue problems [15, 12]. They have been used in conjunction with linear multistep methods [43], Krylov subspace methods [21, 5, 22], and splitting methods [17]. The most popular construction for interpolation is via Lagrange polynomials. The use of rational functions for interpolation on equi-spaced grids for DC methods was studied in [16]. Here, we also discuss the use of interpolants based on continuous extensions of numerical methods as well as splines.

In [13], Dutt *et al.* propose a variant of deferred correction called *spectral deferred correction* (SDC) that bases the correction step on the Picard integral form of the solution to the IVP. The motivation for using the Picard form is the increased stability; however, the choice of Gauss–Legendre (spectral) nodes as abscissae for the associated interpolation also contributes to this. Dutt *et al.* investigate convergence orders as high as 20 on test problems but limit the integrators to forward and backward Euler.

DC and SDC methods have some appealing properties. In [29], Liu *et al.* explore the strong-stability-preserving property in the context of SDC. Recent research has shown that under certain conditions DC methods [1, 2] and SDC methods [17] can be viewed as approximations to implicit Runge–Kutta (IRK) methods. Collocation methods are known to be equivalent to IRK methods (see [18, Chap. II Thm 7.7]), and by construction they have zero residual. In [17], Hagstrom and Zhou show that by constructing residuals of sufficiently high order, an SDC method achieves the same order as an IRK method at the grid points. Similarly, Auzinger *et al.* in [1, 2] show that DC methods are iterative collocation solvers and, with certain choices of nodes, can exhibit superconvergence. A study of the convergence of DC solutions to collocation solutions and subsequent insights into choosing quadrature methods is presented in [36].

SDC was implemented in a semi-implicit manner for the solution of incompressible flows in [31], the Allen–Cahn and Cahn–Hilliard equations in [28], and the compressible Boussinesq equation in [38], where the splitting was based on wave speeds. SDC was applied in a multi-level framework for solving PDEs in [41], where spatial coarsening was used for generating predictors (initial approximations; *provisional* or *uncorrected* solutions). A study of the accuracy and stability of SDC methods for various choices of quadrature nodes (Gauss–Legendre, Gauss–Lobatto, Gauss–Radau, and uniformly spaced) was performed in [25]. The use of higher-order provisional solutions before the application of SDC iterations is explored in [26], where the backward Euler method is used to improve the order of accuracy by one at each SDC iteration. Order reduction associated with semi-implicit SDC was observed in [30]. SDC was used in a multi-implicit manner in [4] and [24]. In [4], advection-diffusion-reaction PDEs are solved using the method of lines. The advection term is integrated explicitly;

the diffusion and reaction terms are integrated implicitly, independently, and with potentially different time steps. In [24], a finite volume approach is used to discretize the compressible Navier–Stokes equations in order to produce a conservative method. SDC was applied to fractional differential equations in [45]. Of particular note to parallel-in-time integrators, in [32] SDC was incorporated as the "fine propagator" in the parareal framework [27].

More recently, Christlieb *et al.* describe another variant called *integral deferred correction* (IDC) that allows for higher-order single-step integrators to be used [8, 9, 7]. Operator splitting techniques (such as Lie–Trotter and Strang operator splitting), alternating direction implicit (ADI) methods, and implicit-explicit (IMEX) methods were introduced in the IDC context in [7] and applied to the Vlasov–Poisson problem in [6]. IDC methods were applied to singular perturbation problems in [3].

Of particular recent interest, in [10], Christlieb *et al.* demonstrate that it is possible to implement a parallel version of IDC, which they call *revisionist integral deferred correction* (RIDC), such that under mild assumptions it is possible to produce an arbitrarily high-order solution in essentially the same wall-clock time it takes to compute the provisional solution. In other words, it can exploit coarse-grained parallelism on a small to moderate number of compute cores with near perfect efficiency. More generally, this makes DC methods particularly attractive for use on modern computer architectures: a high-accuracy solution to an IVP can be obtained in approximately the same wall-clock time as a low-accuracy solution provided a small number of cores are available. Investigations into the use of adaptive step-size control within the RIDC framework were carried out in [11].

In this paper, we undertake a review of the different classes of DC methods used in the solution of IVPs. We assume that the IVP is in the standard form

$$\frac{d}{dt}\mathbf{y}(t) = \mathbf{f}(t, \mathbf{y}(t)), \quad a < t < b, \tag{1a}$$

$$\mathbf{y}(a) = \mathbf{y}_a, \tag{1b}$$

where $\mathbf{y}_a \in \mathbb{C}^m$, $\mathbf{y}(t) : \mathbb{R} \to \mathbb{C}^m$ and $\mathbf{f} : \mathbb{R} \times \mathbb{C}^m \to \mathbb{C}^m$. We assume that $\mathbf{f}$ is sufficiently smooth for existence and uniqueness of the solution and its corresponding series expansions when using high-order methods.

The remainder of this paper is structured as follows. In §2, we review some of the theoretical background necessary for the understanding of deferred correction methods and establish a unified notation. In §3, we review the different deferred correction methods in use, including (classical) deferred correction, spectral deferred correction, integral deferred correction, and revisionist integral deferred correction. We offer some ideas for implementation that illustrate the flexibility of deferred correction methods, with particular emphasis on the RIDC formulation for parallel architectures. In §4, we discuss our numerical testing of these ideas and the results on some benchmark problems. In §5, we give our conclusions.

2. Theoretical background.

In this section, we provide the essential theoretical background behind deferred correction for the solution of IVPs.

2.1. Accuracy, stability, and stiffness.

When solving an IVP (1) numerically, we are concerned with issues such as the order of accuracy and the stability of a numerical method and its effectiveness as a stiff solver.

If the numerical solution to (1) at the endpoint $t = b$ is $\widetilde{\mathbf{y}}(b)$, then the method used for its computation is said to be of *order of accuracy* (or simply *order*) p if for

sufficiently smooth $\mathbf{f}$ there exists a (real) constant $C > 0$ such that

$$\|\mathbf{y}(b) - \widetilde{\mathbf{y}}(b)\| < C \cdot (\Delta t)^p,$$

where Δt is the step size, assumed to be sufficiently small. The (linear) stability of a numerical method is analyzed by applying it to the scalar test equation

$$\begin{aligned} \frac{d}{dt}y(t) &= \lambda\, y(t), \quad 0 < t < b, \\ y(0) &= 1, \end{aligned}$$

where $\lambda \in \mathbb{C}$, and we are most interested in the case $\mathrm{Re}(\lambda) \leq 0$. This problem has the exact solution $y(t) = e^{\lambda t}$. The solution of the test equation with a numerical method can usually be written in terms of the method's *stability function* $R(z)$. After n steps of constant size Δt, the numerical solution is

$$y_n = R^n(z),$$

where $z = \lambda \Delta t$. Clearly if $|R(z)| \leq 1$, the numerical solution remains bounded as n increases, and the method is called *stable* for such values of z. If a method is stable for all z with $\mathrm{Re}(z) \leq 0$, i.e., for all z in the left half-plane, then the method is called *A-stable*. If a method is stable for all z with $\pi - \alpha \leq \arg(z) \leq \pi + \alpha$, then the method is *A(α)-stable*. It can be seen that A-stability is equivalent to $A(\alpha)$-stability with $\alpha = \frac{\pi}{2}$. A method is called *L-stable* if it is A-stable and in addition

$$\lim_{\mathrm{Re}(z)\to-\infty} R(z) = 0.$$

Notwithstanding some potential recent progress [40], there is arguably no universally accepted definition of stiffness in the numerical solution of differential equations. For the purposes of this discussion, it suffices to say that an IVP (1) is *stiff* when the restrictions imposed on the step size of a numerical method due to stability requirements are stricter than those due to accuracy requirements. Numerical methods that are A- or L-stable are generally considered to be effective in avoiding step-size restrictions due to stiffness.

2.2. Spectral integration and differentiation. Suppose that $\{t_1, t_2, \ldots, t_n\}$ is a strictly increasing sequence of points in $\mathbb{R}$ and that for each point t_i there is a corresponding solution value $\mathbf{y}_i$. Let $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n)$. Then we can define the *Lagrange form* of the interpolant of order n for the points $\mathbf{Y}$ at any point $t \in \mathbb{R}$ by the familiar formula

$$\mathbf{L}_n(t; \mathbf{Y}) = \sum_{i=1}^{n} \ell_i(t) \cdot \mathbf{y}_i,$$

where $\mathbf{L}_n : \mathbb{R} \times \mathbb{C}^{m\times n} \to \mathbb{C}^m$ and the basis functions $\ell_i(t)$ are given by the formula

$$\ell_i(t) = \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{t - t_j}{t_i - t_j}.$$

It is well known that the operations of differentiation and integration of polynomials can be conveniently viewed as matrix multiplication; we now define the differentiation and integration matrices to be used in this discussion.

DEFINITION 1 (Differentiation matrix). *Let* $\mathbf{y}(t) : \mathbb{R} \to \mathbb{C}^m$*, and let the matrix* $\mathbf{Y} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n)$ *be defined by*

$$\mathbf{y}_i = \mathbf{y}(t_i), \quad i = 1, 2, \ldots, n.$$

Then if $\mathbf{d} = (\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n)$ *is defined by*

$$\mathbf{d}_i = \frac{d}{dt}\Big|_{t=t_i} \mathbf{L}_n(t; \mathbf{Y}), \quad i = 1, 2, \ldots, n,$$

the linear mapping $\mathbf{D}_n : \mathbb{C}^{m\times n} \to \mathbb{C}^{m\times n}$ *for which*

$$\mathbf{d} = \mathbf{D}_n(\mathbf{Y})$$

is referred to as the differentiation matrix.

DEFINITION 2 (Integration matrix). *Similarly, if* $\mathbf{q} = (\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n)$ *is defined by*

$$\mathbf{q}_i = \int_{-1}^{t_i} \mathbf{L}_n(t; \mathbf{Y})\, \mathrm{d}t, \quad i = 1, 2, \ldots, n,$$

then the linear mapping $\mathbf{Q}_n : \mathbb{C}^{m\times n} \to \mathbb{C}^{m\times n}$ *for which*

$$\mathbf{q} = \mathbf{Q}_n(\mathbf{Y})$$

is referred to as the integration matrix.

We have defined the lower bound of the integral in Definition 2 to be -1 in anticipation of using Gaussian quadrature nodes (Gauss–Legendre or Gauss–Lobatto) for its evaluation in spectral deferred correction. Given a positive integer n, we denote the n Gaussian nodes on $[-1, 1]$ by $\tau_1, \tau_2, \ldots, \tau_n$. For the scaled and shifted problem on $[a, b] \subset \mathbb{R}$, we denote the n Gaussian nodes by $\bar{\tau}_1, \bar{\tau}_2, \ldots, \bar{\tau}_n$, given by

$$\bar{\tau}_i = \frac{b-a}{2} \cdot \tau_i + \frac{b+a}{2}, \quad i = 1, 2, \ldots, n. \tag{2}$$

If $\mathbf{D}_n$ and $\mathbf{Q}_n$ are computed using Gaussian nodes, they have been referred to as the *spectral* differentiation and integration matrices, respectively. However, this terminology generally applies to any choice of spectral nodes, including Chebyshev nodes. The n Chebyshev nodes on $[-1, 1]$ are

$$\theta_i = -\cos\Big(\frac{2i-1}{2n}\pi\Big), \quad i = 1, 2, \ldots, n,$$

where the negative sign is included so that the nodes are in increasing order. The Chebyshev nodes for the scaled and shifted problem on $[a, b] \subset \mathbb{R}$ are formed as in (2).

If the function $\mathbf{y}(t)$ is a polynomial of degree at most $n-1$ and the matrix $\mathbf{Y}$ is defined as above, then $\mathbf{y}(t) \equiv \mathbf{L}_n(t; \mathbf{Y})$, and the operators $\mathbf{D}_n$ and $\mathbf{Q}_n$ are exact.

3. Classes of Deferred Correction Methods. We now describe the main classes of deferred correction methods that have been proposed. On the continuous level, all the methods are equivalent. We distinguish between deferred correction methods based on the form of the error equation that is discretized and solved numerically. In practice, the stability of the implementation depends critically on how this is done.

We suppose that the time domain, $[a, b]$, is subdivided into J intervals,

$$a = t_0 < t_1 < \cdots < t_j < \cdots < t_J = b. \tag{3}$$

Each interval, $[t_j, t_{j+1}]$ is further subdivided using n nodes,

$$t_j \le t_{j,1} < t_{j,2} < \cdots < t_{j,n} \le t_{j+1} \quad j = 0, 1, \ldots, J-1. \tag{4}$$

We note that each group of nodes, $\{t_{j,k}\}_{k=1}^{n}$, may include both endpoints $\{t_j, t_{j+1}\}$ of each interval, as in the case of Gauss–Lobatto or uniformly spaced nodes, or they may not, as in the case of Gauss–Legendre, Gauss–Radau, or Chebyshev nodes.

The general idea for deferred correction methods is as follows. In interval $[t_j, t_{j+1}]$, one uses a method of order p_0 to determine a provisional solution for equation (1), $\mathbf{Y}_j^{[0]} = (\mathbf{y}_{j,1}^{[0]}, \ldots, \mathbf{y}_{j,n}^{[0]})$, where

$$\mathbf{y}_{j,i}^{[0]} = \mathbf{y}(t_{j,i}) + O((\Delta t)^{p_0}), \quad i = 1, 2, \ldots, n.$$

We then iterate with a deferred correction method. Denoting the continuous approximation to $\mathbf{y}(t)$ based on the points $\mathbf{Y}_j^{[k]}$ on interval $[t_j, t_{j+1}]$ and at iteration k by $\mathbf{Y}_j^{[k]}(t) = (\mathbf{y}_{j,1}^{[k]}(t), \ldots, \mathbf{y}_{j,n}^{[k]}(t))$, the *error function* at iteration k is defined by

$$\mathbf{e}_j^{[k]}(t, \mathbf{Y}_j^{[k]}(t)) = \mathbf{y}(t) - \mathbf{Y}_j^{[k]}(t), \quad t \in [t_j, t_{j+1}], \tag{5}$$

and can be constructed and returned as part of the output from each algorithm described below, if desired.

3.1. Classical Deferred Correction. Classical deferred correction (CDC) uses a sequence of IVPs for the error based on equation (5) to continually improve a given approximate solution, $\mathbf{Y}_j^{[0]}(t)$, in each interval. Differentiating the error function (5), we form the error IVP

$$\begin{aligned} \frac{d}{dt}\mathbf{e}_j^{[k]}(t, \mathbf{Y}_j^{[k]}(t)) &= \frac{d}{dt}\mathbf{y}(t) - \frac{d}{dt}\mathbf{Y}_j^{[k]}(t) \\ &= \mathbf{f}\big(t, \mathbf{e}_j^{[k]}(t, \mathbf{Y}_j^{[k]}(t)) + \mathbf{Y}_j^{[k]}(t)\big) - \frac{d}{dt}\mathbf{Y}_j^{[k]}(t), \end{aligned} \tag{6a}$$

$$\mathbf{e}_j^{[k]}(t_j, \mathbf{Y}^{[k]}(t_j)) = \mathbf{0}. \tag{6b}$$

We solve the error equation at iteration k with a method of order p_k to get an approximate solution $\mathbf{E}_j^{[k]} = (\mathbf{e}_{j,1}^{[k]}, \mathbf{e}_{j,2}^{[k]}, \ldots, \mathbf{e}_{j,n}^{[k]})$, where

$$\mathbf{e}_{j,i}^{[k]} = \mathbf{e}_j^{[k]}(t_{j,i}, \mathbf{y}_{j,i}^{[k]}) + O((\Delta t)^{p_k}), \quad i = 1, 2, \ldots, n.$$

We now form the corrected solution

$$\mathbf{Y}_j^{[k+1]} = \mathbf{Y}_j^{[k]} + \mathbf{E}_j^{[k]},$$

extrapolating to t_{j+1} if necessary.

The procedure for CDC is given in Algorithm 1. Provided the order of the interpolant is sufficiently high and uniformly spaced nodes are used, after this procedure the solution is accurate to order $O((\Delta t)^{P_K})$, where $P_K = \sum_{k=0}^{K} p_k$. More precisely

however, if the order of accuracy of $\mathbf{Y}^{[k]}(t)$ is n (e.g., using Lagrange polynomial interpolation with n points), then the solution has order of accuracy

$$O((\Delta t)^{\min(P,n-1)}) \tag{7}$$

because it uses the differentiated interpolant. In practice, the quality of the estimate is also influenced by the stability of interpolant. For example, the "catastrophically bad" idea [44, p. 42] of high-order interpolation at equally spaced points generally gives poor results despite the formally high order of accuracy. However, by keeping the order of the interpolant $\mathbf{Y}^{[k]}(t)$ low, e.g., by using splines or using Lagrange interpolation but on suitably small intervals $[t_j, t_{j+1}]$, CDC can produce acceptable results. This is illustrated in §4.

Algorithm 1 Classical deferred correction

1. **for** $j = 1$ to J **do**
2. Compute an initial approximation $\mathbf{Y}_j^{[0]} = (\mathbf{y}_{j,1}^{[0]}, \mathbf{y}_{j,2}^{[0]}, \ldots, \mathbf{y}_{j,n}^{[0]})$ to IVP (1) at the points $t_{j,i} \in [t_{j,1}, t_{j,n}]$, $i = 1, 2, \ldots, n$, using a method of order p_0.
3. **for** $k = 1$ to K **do**
4. Form the continuous solution $\mathbf{Y}_j^{[k-1]}(t)$.
5. Solve the IVP (6) using a method of order p_k to compute an approximation to the error $\mathbf{E}_j^{[k-1]} = (\mathbf{e}_{j,1}^{[k-1]}, \ldots, \mathbf{e}_{j,n}^{[k-1]})$.
6. Form the new approximate solution $\mathbf{Y}_j^{[k]} = \mathbf{Y}_j^{[k-1]} + \mathbf{E}_j^{[k-1]}$.
7. **end for**
8. **end for**
9. **return** $\mathbf{Y}_j^{[K]}(t_{j+1})$, using extrapolation if necessary.

3.2. Spectral Deferred Correction. Like CDC, SDC uses a sequence of IVPs to continually improve an initial approximate solution $\mathbf{Y}^{[0]}(t)$. The equation used for the error is based on the Picard form of the IVP (1),

$$\mathbf{y}(t) = \mathbf{y}_a + \int_a^t \mathbf{f}(t', \mathbf{y}(t'))\,\mathrm{d}t'. \tag{8}$$

Given a continuous approximation $\mathbf{Y}^{[k]}(t)$ to the solution of (8), we define the *residual function*

$$\mathbf{r}^{[k]}(t, \mathbf{Y}^{[k]}(t)) = \mathbf{y}_a + \int_a^t \mathbf{f}(t', \mathbf{Y}^{[k]}(t'))\,\mathrm{d}t' - \mathbf{Y}^{[k]}(t)$$

and write

$$\mathbf{e}^{[k]}(t, \mathbf{Y}^{[k]}(t)) = \int_a^t \left[\mathbf{f}(t', \mathbf{Y}^{[k]}(t') + \mathbf{e}^{[k]}(t', \mathbf{Y}^{[k]}(t'))) - \mathbf{f}(t', \mathbf{Y}^{[k]}(t'))\right]\,\mathrm{d}t' + \mathbf{r}^{[k]}(t, \mathbf{Y}^{[k]}(t)). \tag{9}$$

We define the function $\boldsymbol{\Delta}\mathbf{f} : \mathbb{R} \times \mathbb{C}^m \to \mathbb{C}^m$ by

$$\boldsymbol{\Delta}\mathbf{f}^{[k]}(t, \mathbf{Y}^{[k]}(t)) = \mathbf{f}\big(t, \mathbf{Y}^{[k]}(t) + \mathbf{e}^{[k]}(t, \mathbf{Y}^{[k]}(t))\big) - \mathbf{f}(t, \mathbf{Y}^{[k]}(t)), \tag{10}$$

then rewrite (9) in a Picard integral form like (8),

$$\mathbf{e}^{[k]}(t, \mathbf{Y}^{[k]}(t)) = \int_a^t \boldsymbol{\Delta}\mathbf{f}^{[k]}(t', \mathbf{Y}^{[k]}(t'))\,\mathrm{d}t' + \mathbf{r}^{[k]}(t, \mathbf{Y}^{[k]}(t)). \tag{11}$$

We then differentiate (11) to form the IVP used for the SDC algorithm

$$\frac{d}{dt}\mathbf{e}^{[k]}(t,\mathbf{Y}^{[k]}(t)) = \boldsymbol{\Delta}\mathbf{f}^{[k]}(t,\mathbf{Y}^{[k]}(t)) + \frac{d}{dt}\mathbf{r}^{[k]}(t,\mathbf{Y}^{[k]}(t)),$$
$$\mathbf{e}^{[k]}(a,\mathbf{Y}^{[k]}(a)) = \mathbf{0}.$$

The SDC algorithm is applied independently to each group of nodes (4) within $[t_j, t_{j+1}]$ using

$$\frac{d}{dt}\mathbf{e}_j^{[k]}(t,\mathbf{Y}_j^{[k]}(t)) = \boldsymbol{\Delta}\mathbf{f}^{[k]}(t,\mathbf{Y}_j^{[k]}(t)) + \frac{d}{dt}\mathbf{r}^{[k]}(t,\mathbf{Y}_j^{[k]}(t)), \tag{12a}$$
$$\mathbf{e}_j^{[k]}(t_j,\mathbf{Y}_j^{[k]}(t_j)) = \mathbf{0}. \tag{12b}$$

The residual function within each group of nodes satisfies

$$\mathbf{r}_j^{[k]}(t,\mathbf{Y}_j^{[k]}(t)) = \mathbf{Y}_{j-1}^{[k]}(t_j) + \int_{t_j}^{t} \mathbf{f}(t',\mathbf{Y}_j^{[k]}(t'))\,\mathrm{d}t' - \mathbf{Y}_j^{[k]}(t). \tag{13}$$

We define the the $m \times n$ residual matrix $\mathbf{R}_j(\mathbf{Y}^{[k]})$ by using quadrature to approximate the integral in (13),

$$\begin{aligned}\mathbf{R}_j^{[k]}(\mathbf{Y}_j^{[k]}) &= \big(\mathbf{r}_j^{[k]}(t_{j.1},\mathbf{y}_{j,1}^{[k]}), \mathbf{r}^{[k]}(t_{j,2},\mathbf{y}_{j,2}^{[k]}), \ldots, \mathbf{r}^{[k]}(t_{j,n},\mathbf{y}_{j,n}^{[k]})\big), \\ &\approx \mathbf{A}_j^{[k]} + \mathbf{Q}_n(\mathbf{F}_j^{[k]}) - \mathbf{Y}_j^{[k]},\end{aligned} \tag{14}$$

where the matrix $\mathbf{A}_j^{[k]}$ is given by

$$\mathbf{A}_j^{[k]} = \begin{cases} (\mathbf{y}_a, \mathbf{y}_a, \ldots, \mathbf{y}_a), & j = 1, \\ (\mathbf{Y}_{j-1}^{[k]}(t_j), \mathbf{Y}_{j-1}^{[k]}(t_j), \ldots, \mathbf{Y}_{j-1}^{[k]}(t_j)), & j > 1, \end{cases}$$

and the matrix $\mathbf{F}^{[k]}$ is given by

$$\mathbf{F}_j^{[k]} = \big(\mathbf{f}(t_{j,1},\mathbf{y}_{j,1}^{[k]}), \mathbf{f}(t_{j,2},\mathbf{y}_{j,2}^{[k]}), \ldots, \mathbf{f}(t_{j,n},\mathbf{y}_{j,n}^{[k]})\big).$$

The procedure for SDC is given in Algorithm 2. SDC as originally formulated uses spectral nodes $\bar{\tau}_i$, $i = 1, 2, \ldots, n$, e.g., the Gaussian nodes (2), as the points at which $\mathbf{Y}_j^{[k]}$ is computed. Using spectral nodes, SDC generally gives superior results compared to CDC, especially for large n, because it does not differentiate the interpolating polynomial.

3.3. Integral Deferred Correction. The main idea behind integral deferred correction (IDC) is to solve an integral form of the error equation that also incorporates the defect of the solution. Given an approximate continuous solution $\mathbf{Y}^{[k]}(t)$ to (1), we define the *defect* function by

$$\boldsymbol{\delta}^{[k]}(t,\mathbf{Y}^{[k]}(t)) = \frac{d}{dt}\mathbf{Y}^{[k]}(t) - \mathbf{f}(t,\mathbf{Y}^{[k]}(t)). \tag{15}$$

Using (15), the derivative of the error function given by (5) can be expressed as

$$\begin{aligned}\frac{d}{dt}\mathbf{e}^{[k]}(t,\mathbf{Y}^{[k]}(t)) &= \frac{d}{dt}\mathbf{y}(t) - \frac{d}{dt}\mathbf{Y}^{[k]}(t) \\ &= \mathbf{f}(t,\mathbf{y}(t)) - \mathbf{f}(t,\mathbf{Y}^{[k]}(t)) - \boldsymbol{\delta}^{[k]}(t,\mathbf{Y}^{[k]}(t)) \\ &= \mathbf{f}\big(t,\mathbf{Y}^{[k]}(t) + \mathbf{e}^{[k]}(t,\mathbf{Y}^{[k]}(t))\big) - \mathbf{f}(t,\mathbf{Y}^{[k]}(t)) - \boldsymbol{\delta}^{[k]}(t,\mathbf{Y}^{[k]}(t)),\end{aligned}$$

Algorithm 2 Spectral deferred correction

1. **for** $j = 1$ to J **do**
2. Compute an approximation $\mathbf{Y}_j^{[0]} = (\mathbf{y}_{j,1}^{[0]}, \mathbf{y}_{j,2}^{[0]}, \ldots, \mathbf{y}_{j,n}^{[0]})$ to IVP (1) at the spectral nodes $t_{j,i} \in [t_{j,1}, t_{j,n}],\ i = 1, 2, \ldots, n$.
3. **for** $k = 1$ to K **do**
4. Form the continuous solution $\mathbf{Y}_j^{[k-1]}(t)$.
5. Compute the approximate residual $\mathbf{R}_j^{[k-1]}(\mathbf{Y}^{[k-1]})$ by (14).
6. Solve the IVP (12) using a method of order p_k to compute an approximation to the error $\mathbf{E}_j^{[k-1]} = (\mathbf{e}_{j,1}^{[k-1]}, \ldots, \mathbf{e}_{j,n}^{[k-1]})$.
7. Form the new approximate solution $\mathbf{Y}_j^{[k]} = \mathbf{Y}_j^{[k-1]} + \mathbf{E}_j^{[k-1]}$.
8. **end for**
9. **end for**
10. **return** $\mathbf{Y}_j^{[K]}(t_{j+1})$, using extrapolation if necessary.

and after rearranging

$$\frac{d}{dt}\mathbf{e}^{[k]}(t, \mathbf{Y}^{[k]}(t)) + \boldsymbol{\delta}^{[k]}(t, \mathbf{Y}^{[k]}(t)) = \mathbf{f}\big(t, \mathbf{Y}^{[k]}(t) + \mathbf{e}^{[k]}(t, \mathbf{Y}^{[k]}(t))\big) - \mathbf{f}(t, \mathbf{Y}^{[k]}(t)).$$

This can be expressed as

$$\frac{d}{dt}\left[\mathbf{e}^{[k]}(t, \mathbf{Y}^{[k]}(t)) + \int_a^t \boldsymbol{\delta}^{[k]}(t', \mathbf{Y}^{[k]}(t'))\,\mathrm{d}t'\right] = \mathbf{f}\big(t, \mathbf{Y}^{[k]}(t) + \mathbf{e}^{[k]}(t, \mathbf{Y}^{[k]}(t))\big) - \mathbf{f}(t, \mathbf{Y}^{[k]}(t)), \tag{16}$$

or after using (15) and (10),

$$\frac{d}{dt}\left[\mathbf{e}^{[k]}(t, \mathbf{Y}^{[k]}(t)) + \mathbf{Y}^{[k]}(t) - \int_a^t \mathbf{f}(t', \mathbf{Y}^{[k]}(t'))\,\mathrm{d}t'\right] = \boldsymbol{\Delta}\mathbf{f}^{[k]}(t, \mathbf{Y}^{[k]}(t)).$$

IDC has recently been presented as exclusively solving the error equation on (uniform) subdivisions (or groups of nodes) of the original mesh [9, 10]. This construction in itself improves the quality of the results, and it is also possible to contemplate adaptive time steps on the primary grid [8].

The IDC algorithm is applied independently to each group of nodes (4) within $[t_j, t_{j+1}]$ using

$$\frac{d}{dt}\left[\mathbf{e}_j(t, \mathbf{Y}_j^{[k-1]}(t)) + \int_{t_j}^t \boldsymbol{\delta}^{[k]}(t', \mathbf{Y}^{[k]}(t'))\,\mathrm{d}t'\right] = \boldsymbol{\Delta}\mathbf{f}^{[k-1]}(t, \mathbf{Y}_j^{[k-1]}(t)). \tag{17}$$

It is understood that the algorithm is iterated completely on each group of nodes before moving on to the next.

Note that discretization of (17) also approximates the integral by a quadrature formula. For example, if the discretization is by the forward Euler method, then we have

$$\begin{aligned}\mathbf{e}_{j,i+1}^{[k]} = \mathbf{e}_{j,i}^{[k]} &- (\mathbf{y}_{j,i+1}^{[k]} - \mathbf{y}_{j,i}^{[k]}) \\ &+ \Delta t\Big(\mathbf{f}(t_{j,i}, \mathbf{y}_{j,i}^{[k]} + \mathbf{e}_{j,i}^{[k]}) - \mathbf{f}(t_{j,i}, \mathbf{y}_{j,i}^{[k]})\Big) + \int_{t_{j,i}}^{t_{j,i+1}} \mathbf{f}(t', \mathbf{Y}_j^{[k]}(t'))\,\mathrm{d}t',\end{aligned}$$

Algorithm 3 Integral deferred correction

for $j = 1$ to J **do**
  Compute the initial approximation $\mathbf{Y}_j^{[0]} = (\mathbf{y}_{j,1}^{[0]}, \mathbf{y}_{j,2}^{[0]}, \ldots, \mathbf{y}_{j,n}^{[0]})$ at the points $t_{j,l} \in [t_{j,1}, t_{j,n}]$, $l = 1, 2, \ldots, n$, using a method of order p_0.
  for $k = 1$ to K **do**
    Form the continuous solution $\mathbf{Y}_j^{[k-1]}(t)$.
    Compute the defect $\boldsymbol{\delta}_j^{[k]}(t, \mathbf{Y}_j^{[k-1]}(t))$ using (15).
    Solve the ODE (17), with initial condition $\mathbf{e}_j(t_j, \mathbf{Y}_j^{[k-1]}(t_j)) = \mathbf{0}$, using a method of order p_k to compute an approximation to the error $\mathbf{E}_j^{[k-1]} = (\mathbf{e}_{j,1}^{[k-1]}, \ldots, \mathbf{e}_{j,n}^{[k-1]})$ at the points $t_{j,l} \in [t_{j,1}, t_{j,n}]$ $l = 1, 2, \ldots, n$.
    Form the new approximate solution $\mathbf{Y}_j^{[k]} = \mathbf{Y}_j^{[k-1]} + \mathbf{E}_j^{[k-1]}$.
  end for
end for
return $\mathbf{Y}_j^{[K]}(t_{j+1})$, using extrapolation if necessary.

and replacing the integral with a quadrature formula

$$\mathbf{e}_{j,i+1}^{[k]} = \mathbf{e}_{j,i}^{[k]} - (\mathbf{y}_{j,i+1}^{[k]} - \mathbf{y}_{j,i}^{[k]}) + \Delta t\Big(\mathbf{f}(t_{j,i}, \mathbf{y}_{j,i}^{[k]} + \mathbf{e}_{j,i}^{[k]}) - \mathbf{f}(t_{j,i}, \mathbf{y}_{j,i}^{[k]})\Big) + \Delta t \sum_{i=1}^{n} S_{j,i} \mathbf{f}(t_{j,i}, \mathbf{y}_{j,i}^{[k]}), \tag{18}$$

where the quadrature weights $S_{j,i}$ can be defined, e.g., by integrating the basis functions of the Lagrange form of interpolating polynomial

$$S_{j,i} = \int_{t_{j,i}}^{t_{j,i+1}} \ell_{j,i}(t)\,\mathrm{d}t = \int_{t_{j,i}}^{t_{j,i+1}} \prod_{\substack{i'=1 \\ i' \neq i}}^{n} \frac{t - t_{j,i'}}{t_{j,i} - t_{j,i'}}\,\mathrm{d}t.$$

We observe that if SDC is formulated using uniform nodes, equation (18) is recovered. Further, an IDC method, constructed using $(n+1)$ quadrature nodes and $(K+1)$ prediction plus correction iterations of an s-stage explicit RK method, can be reformulated as a $((K+1)\,s\,n)$-stage RK method [8].

If the error equation (17) at iteration k is discretized using a method of order p_k, the corrected solution is of order $\sum_{j=0}^{K} p_j$ if the quadrature is sufficiently accurate and if uniform nodes are used. If non-uniform nodes are used, only order $(p_0 + K)$ can be guaranteed [9].

3.4. Parallel Deferred Correction. The deferred correction framework can allow for task-level parallelism, where multiple levels of correction can be computed simultaneously. This was recently observed, implemented, and discussed for RIDC methods [10], but the idea can be broadly applied to the classes of deferred correction equations previously discussed. Recall that the time domain is divided into J intervals, equation (3), and each subinterval is further subdivided using n nodes, equation (4). To describe the algorithm, it is convenient to consider nodes that include the endpoints of each interval,

$$t_j = t_{j,1} < t_{j,2} < \cdots < t_{j,n} = t_{j+1} \quad j = 0, 1, \ldots, J-1,$$

and relabel these nodes in a global order,

$$\tau_{m'} = t_{j,i}, \quad m' = 0, 1, \ldots, M, \tag{19}$$

where $\tau_0 = a$, $j = m' \div (n-1)$, $i = (m' \mod (n-1)) + 1$ and $M = J(n-1)$. Instead of iterating the CDC, SDC, or IDC algorithm completely on each group of nodes, one first observes that equations (6), (12), and (16), respectively, can be solved directly using a piecewise continuous approximation to $\mathbf{Y}^{[k-1]}(t)$. This allows us to change the order of the loops in Algorithms 1, 2, and 3. For example, a modified classical deferred correction algorithm is given in Algorithm 4.

Algorithm 4 Modified classical deferred correction

1. Compute an initial approximation $\mathbf{Y}^{[0]} = (\mathbf{y}_0^{[0]}, \mathbf{y}_1^{[0]}, \ldots, \mathbf{y}_M^{[0]})$ to IVP (1) using a method of order p_0.
2. **for** $k = 1$ to K **do**
3. Form the piecewise continuous solution $\mathbf{Y}^{[k-1]}(t)$.
4. Solve the IVP (6) using a method of order p_k to compute an approximation to the error $\mathbf{E}^{[k-1]} = (\mathbf{e}_0^{[k-1]}, \ldots, \mathbf{e}_M^{[k-1]})$.
5. Form the new approximate solution $\mathbf{Y}^{[k]} = \mathbf{Y}^{[k-1]} + \mathbf{E}^{[k-1]}$.
6. **end for**
7. **return** $\mathbf{y}_M^{[K]}$

It has been shown numerically that the modified methods are stable provided the predictor is stable [10]. Although the modified CDC, SDC, and IDC algorithms can be shown to have the same asymptotic convergence behavior as their CDC, SDC, and IDC counterparts, the final solution generated by the modified algorithms is generally less accurate because the correction equations are not iterated to completion (level K) on each group of nodes; i.e., the integrator at level k, $k < K$, proceeds using information from levels $k' < k$. The benefit of approximating the solution using the modified equations is the possibility for *pipeline parallelism*, where multiple correction levels can be simultaneously computed. Specifically, once a "piece" of the piecewise continuous solution $\mathbf{Y}^{[k-1]}(t)$ can be constructed, iterate k can be computed while iterate $(k-1)$ continues its computation. This idea is illustrated in Figure 1, where a first-order predictor is updating a provisional solution from t_{j-1} to t_j, while corrector k computes an approximation to the error at time t_{j-kn} in a pipeline-parallel fashion. For uniformly spaced nodes, a "sliding stencil" of minimal sizes is possible, as illustrated in Figure 2, reducing the memory footprint and the lag interval of a pipeline parallel implementation [10].

3.5. Analysis of Parallel Speedup and Efficiency. We present an analysis of the efficiency of a parallel implementation of CDC using a forward Euler integrator. In the notation introduced previously, we have J groups of n nodes (4), where for simplicity we assume each group contains both endpoints (and hence $n \geq 2$). We assume that function evaluations all require the same amount of computation time and that memory access/communication overhead is negligible compared to this computation time. Thus, (non-concurrent) function evaluations can be used as a proxy for computational time.

The forward Euler method requires one function evaluation per node for computing the provisional solution and one function evaluation per node for each correction step. Thus, the provisional solution requires $J(n-1)$ function evaluations and each

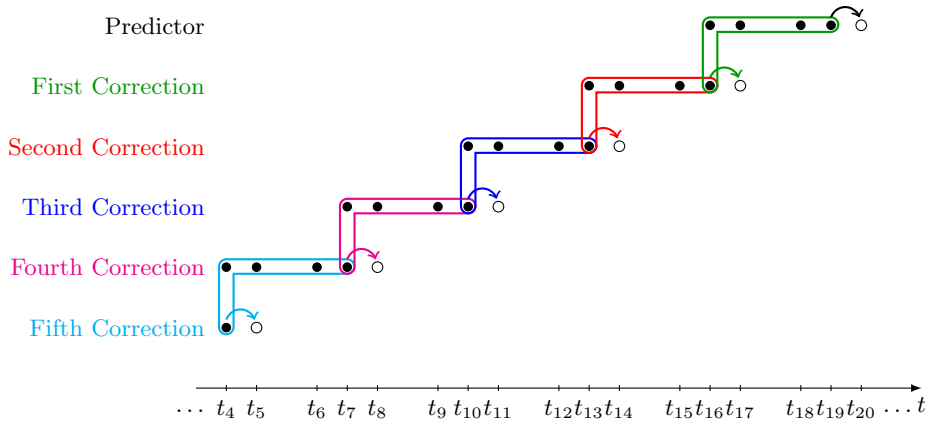


FIG. 1. *The deferred correction framework allows for task-level parallelism, where multiple levels of correction can be computed simultaneously. While the first-order predictor is updating a provisional solution from t_{j-1} to t_j, corrector k is able to compute an approximation to the error at time t_{j-kn} provided that a sufficiently accurate interpolant can be constructed. Here, each group of Gauss–Lobatto nodes generates a sixth-order piecewise interpolant that approximates $\mathbf{Y}^{[k]}(t)$.*

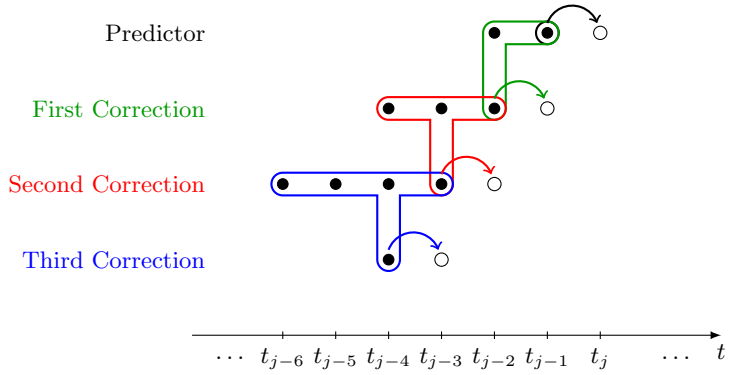


FIG. 2. *The deferred correction framework allows for task-level parallelism, where multiple levels of correction can be computed simultaneously. While the first-order predictor is updating a provisional solution from t_{j-1} to t_j, corrector k is able to compute an approximation to the error at time t_{j-k} provided that a sufficiently accurate interpolant can be constructed. The minimum stencil sizes required by each corrector, assuming uniform nodes, are shown for the respective levels.*

correction requires $J(n-1)$ function evaluations, giving a total of

$$\text{feval}_{\text{serial}} = J(n-1)(1+K),$$

where K is the number of corrections taken. This is also the time measure for a serial implementation. For a parallel implementation, we assume that the first correction step may begin once the initial solution has been computed for the first group of nodes, and similarly that the second correction step may begin once the first correction has been computed for the first group of nodes, and so on. For K corrections, this allows the utilization of $K+1$ processors for computation, provided there are sufficiently many groups (ideally $J \gg K$). We count only the number of non-concurrent function evaluations in our evaluation of the parallel implementation. For K corrections, the

number of non-concurrent function evaluations in a parallel implementation is

$$\text{feval}_{\text{parallel}} = J(n-1) + K(n-1).$$

The *speedup* of a parallel method is given by

$$S_{n_P} = \frac{T_1}{T_{n_P}},$$

where n_P is the number of processors and T_i is the execution time for a computation performed with i processors. A speedup greater than 1 indicates that a computation takes less time to run when additional processors are utilized. In the ideal case, a speedup of n_P is obtained when n_P processors are utilized, leading to a parallel efficiency $E_{n_P} = S_{n_P}/n_P$ of 1. The speedup for the parallel CDC as described above with $K+1$ processors is then

$$S_{K+1} = \frac{J(n-1)(1+K)}{(n-1)(J+K)} = \frac{J(1+K)}{J+K}.$$

In the case when $J = 1$, we obtain $S_{K+1} = 1$, independent of K; i.e., there is no opportunity for parallelization for the case of only one group, and the computation is necessarily serial. In the case when $J \gg K$, we obtain $S_{K+1} \approx 1 + K$, showing that the speedup improves as K increases. The parallel efficiency is

$$E_{K+1} = \frac{S_{K+1}}{K+1} = \frac{J}{J+K},$$

which approaches 1 for $J \gg K$.

4. Numerical Experiments. We now perform a number of numerical experiments to illustrate some of the behaviors of the deferred correction methods described by means of the following test problems:

1. A linear, non-autonomous system,

$$\frac{d}{dt}\mathbf{y}(t) = \begin{pmatrix} ty_2(t) + y_1(t) \\ -ty_1(t) + y_2(t) \end{pmatrix}, \quad 0 < t < 1,$$

$$\mathbf{y}(0) = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

with exact solution,
$$\mathbf{y}(t) = \begin{pmatrix} e^t\left(\cos(0.5t^2) + \sin(0.5t^2)\right) \\ e^t\left(\cos(0.5t^2) - \sin(0.5t^2)\right) \end{pmatrix}.$$

2. A system of ODEs arising from a methods-of-lines discretization of the Brusselator equation in $\mathbb{R}^1$

$$\begin{aligned} u_t &= A + u^2 v - (B+1)u + \alpha u_{xx}, \\ v_t &= Bu - u^2 v + \alpha v_{xx}, \end{aligned} \tag{20}$$

We discretize the spatial derivatives by centered differences and use the parameters $A = 1$, $B = 3$, and $\alpha = 0.02$, initial conditions

$$u(x,0) = 1 + \sin(2\pi x), \qquad v(x,0) = 3,$$

and boundary conditions

$$u(0,t) = u(1,t) = 1, \qquad v(0,t) = v(1,t) = 3.$$

3. A restricted three-body problem, known as the Arenstorf orbit problem [19], whose solution gives the orbit (y_1, y_2) of a light object, e.g., a satellite, moving under the influence of gravity of two heavy objects,

$$
\begin{aligned}
&\ddot{y}_1 = y_1 + 2\dot{y}_2 - \mu' \frac{y_1 + \mu}{D_1} - \mu \frac{y_1 - \mu'}{D_2}, \\
&\ddot{y}_2 = y_2 - 2\dot{y}_1 - \mu' \frac{y_2}{D_1} - \mu \frac{y_2}{D_2}, \\
&D_1 = \left((y_1 + \mu)^2 + y_2^2\right)^{3/2}, \quad D_2 = \left((y_1 - \mu')^2 + y_2^2\right)^{3/2}, \\
&\mu = 0.012277471, \quad \mu' = 1 - \mu.
\end{aligned} \tag{21}
$$

Choosing the initial conditions

$$y_1(0) = 0.994, \quad \dot{y}_1(0) = 0, \quad y_2(0) = 0, \quad \dot{y}_2(0) = -2.001585106379,$$

gives a closed periodic orbit with period 17.065216560159.

4.1. Order of Accuracy. We use test problem 1 to illustrate the expected order of accuracy that can be expected for deferred correction methods. Suppose that we have J groups of n nodes, as described in equation (4). If these nodes are uniformly distributed, we recall (7) that CDC methods can attain order

$$\min(P_K, n - 1),$$

where p_0 is the order of the integrator used to construct the provisional solution, p_k is the order of the integrator used to compute the numerical approximation at level k, K correction steps are taken, and $P_K = \sum_{k=0}^{K} p_k$. Figure 3 shows that the expected orders of accuracy are observed when forward Euler integrators are used with an $n = K + 1$ quadrature nodes for K corrections. Figure 4 shows that the expected orders of accuracy are observed when an RK3 integrator is used to generate the provisional solution and an RK2 corrector is used to solve the error equation (6).

The story is similar if SDC and IDC methods are constructed using uniform nodes. For J groups of n nodes, these methods can attain order

$$\min(P_K, n).$$

We note that the attainable order now depends on n instead of $n - 1$ because neither SDC nor IDC differentiates the interpolant. Figure 5 shows that the expected orders of accuracy are observed when forward Euler integrators are used. Figure 6 shows that the expected orders of accuracy are observed when an RK3 integrator is used to generate the provisional solution and an RK2 corrector is used to solve the error equation (17).

The story gets more interesting if non-uniform nodes are used. We consider CDC methods constructed with J groups of n nodes (4), where each group of n nodes are Gauss–Lobatto nodes, and forward Euler integrators are used to solve test problem 1 and the associated error equation (6). Figure 7 shows that after one correction, the accuracy and order of convergence improves, but with successive corrections, the accuracy improves while the order of accuracy remains stagnant at one. This has previously been observed in [20]. The observed behavior can be explained as follows: (i) the *discrete smoothness* of a discrete data set [9] precludes an increase in the order of accuracy for CDC methods constructed using general non-uniform

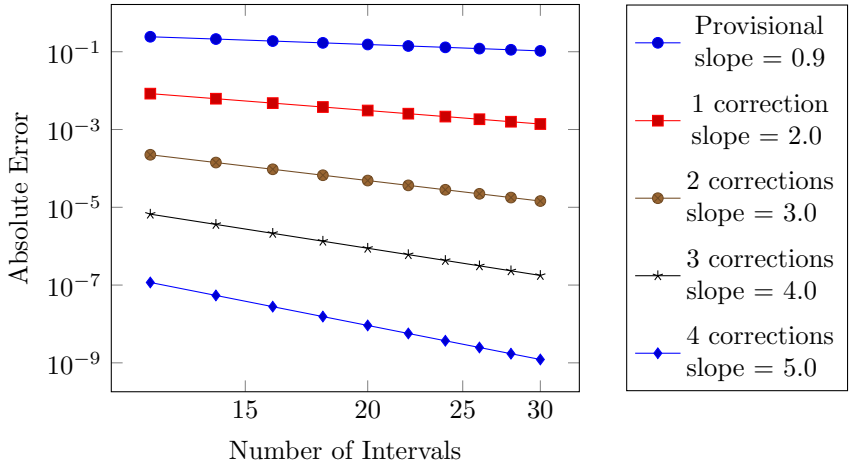


FIG. 3. *Error as a function of the number of intervals for various classical deferred correction methods. Forward Euler integrators are used to solve test problem 1 and the error equation (6). The method with K corrections uses $(K+1)$ uniformly spaced nodes per interval. The expected orders of accuracy are observed.*

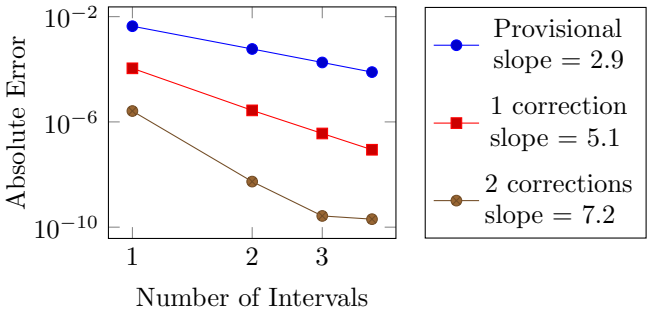


FIG. 4. *Error as a function of the number of intervals for various classical deferred correction methods. An RK3 integrator is used to solve test problem 1, and an RK2 integrator is used to solve the error equation (6). The method with one correction uses six uniformly spaced quadrature nodes in each interval. The method with two corrections uses eight uniformly spaced quadrature nodes in each interval. The expected orders of accuracy are observed. The error stagnates at 10^{-10}, likely due to quadrature error arising from the high-order interpolant.*

nodes, including Gauss–Lobatto, Gauss–Legendre, and Chebyshev nodes. For this example, the exception is (ii) the special case where three Gauss–Lobatto nodes are used because they are in fact three uniformly space nodes; thus, the method with one correction is able to attain second order. A similar behavior is observed if the midpoint method is used to solve test problem 1 and the associated error equation (6). Figure 8 shows that the accuracy improves with more corrections although the order of accuracy remains stagnant at two.

For SDC/IDC methods, the discrete smoothness of non-uniform nodes guarantees only one order of accuracy increase with each correction [9]. The maximum order of accuracy that can be attained depends on two factors: (i) the order of the quadrature formula, and (ii) whether the quadrature nodes (4) include the endpoints of the interval. If the quadrature nodes include the endpoints of the interval, the maximum

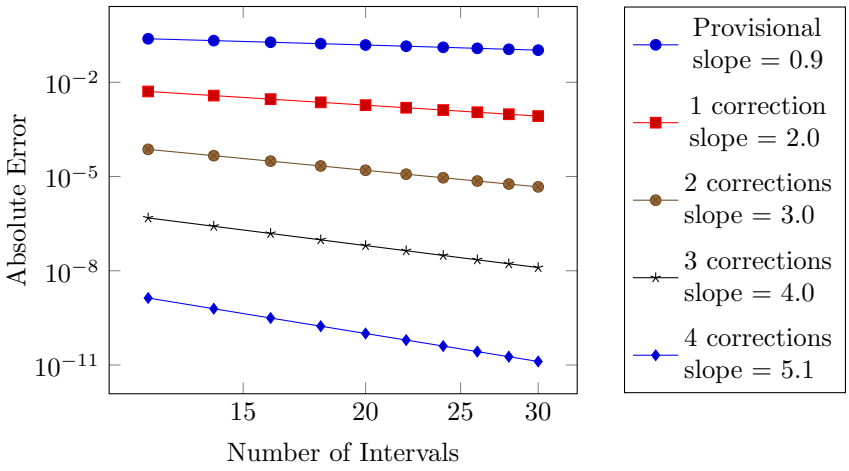


FIG. 5. *Error as a function of the number of intervals for various spectral deferred correction methods. Forward Euler integrators are used to solve test problem 1 and the error equation* (12). *The method with K corrections uses K uniformly spaced nodes per interval. The expected orders of accuracy are observed.*

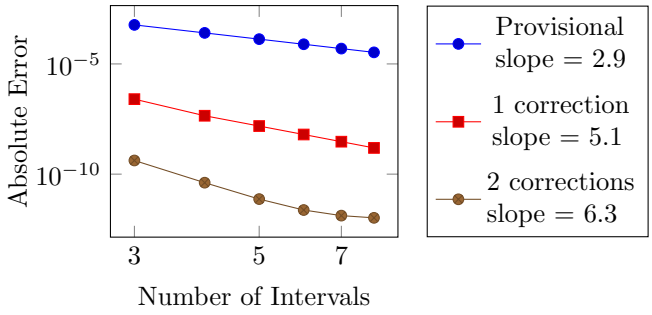


FIG. 6. *Error as a function of the number of intervals for various integral deferred correction methods. An RK3 integrator is used to solve test problem 1, and an RK2 integrator is used to solve the error equation* (17). *The method with one correction uses five uniformly spaced quadrature nodes in each interval. The method with two corrections uses seven uniformly spaced quadrature nodes in each interval. The expected orders of accuracy are observed for the provisional solution and the method with one correction. The error stagnates at 10^{-12}, likely due to quadrature error arising from the high-order interpolant.*

order that the IDC/SDC method can attain is precisely the order of the quadrature formula. The order of n-node Gauss–Lobatto quadrature is $(2n-2)$; the order of n-node Gauss–Legendre quadrature is $2n$; the order of n-node Chebyshev quadrature is $(n+1)$ if n is odd and $(n+2)$ if n is even. If the quadrature nodes do not include the endpoints of the interval, e.g., Gauss–Legendre and Chebyshev nodes, then the maximum order that can be attained is n because extrapolation is required. We highlight these behaviors in the following numerical experiments.

We begin with SDC methods constructed using Gauss–Lobatto nodes and forward Euler integrators applied to test problem 1. The method with one and two corrections used three Gauss–Lobatto nodes in each interval. The methods with three and four corrections used four Gauss–Lobatto nodes in each interval. The method with five corrections used five Gauss–Lobatto nodes. Figure 9 shows that methods with one

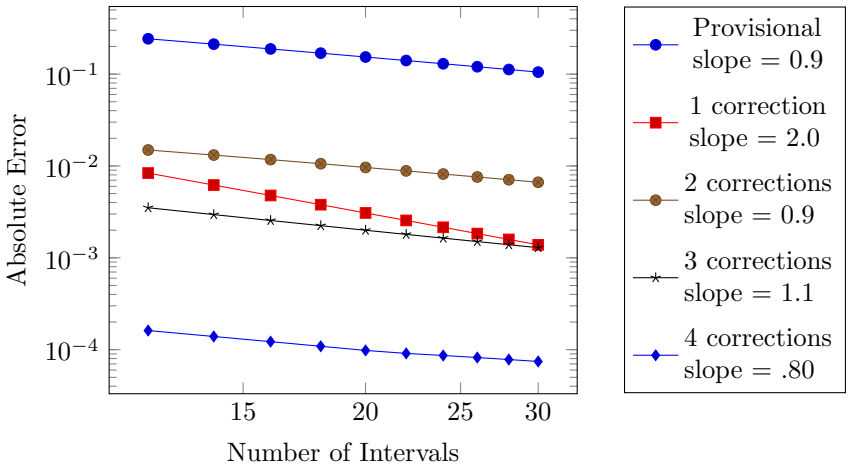


FIG. 7. *Error as a function of the number of intervals for various classical deferred correction methods. Forward Euler integrators are used to solve test problem 1 and the error equation (6). The method with K corrections uses $(K+1)$ Gauss–Lobatto nodes per interval. The order of accuracy is one, except for the special case where the method with one correction attains second order because three Gauss–Lobatto nodes are equivalent to three uniformly spaced nodes.*

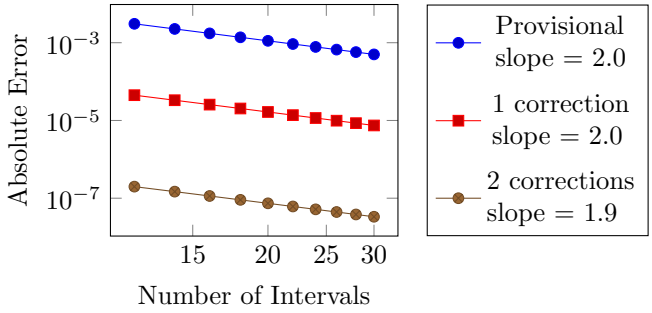


FIG. 8. *Error as a function of the number of intervals for various classical deferred correction methods. The midpoint method is used to solve test problem 1 and the associated error equation (6). The method with one correction uses five quadrature nodes in each interval. The method with two corrections uses seven quadrature nodes in each interval. The order of accuracy stagnates at two, the order of the underlying integrator.*

additional correction add one order of accuracy as expected.

We repeat the experiment constructing methods using Gauss–Legendre nodes and forward Euler integrators applied to test problem 1. The method with one and two corrections used two Gauss–Legendre nodes in each interval. The methods with three and four corrections used three Gauss–Legendre nodes in each interval. The method with five corrections used four Gauss–Legendre nodes. Figure 10 shows that the orders of accuracy of the methods are limited by the orders of the interpolating polynomials (i.e., one less than the number nodes) because extrapolation is required to construct the solution at the interval endpoint. By increasing the number of nodes (order of the interpolating polynomial) in each interval for SDC methods constructed with Gauss–Legendre nodes, we can still achieve one order of accuracy increase with each correction. In Figure 11, each method with K corrections is constructed with $(K+1)$ Gauss–Legendre nodes in each interval. A method with K corrections achieves

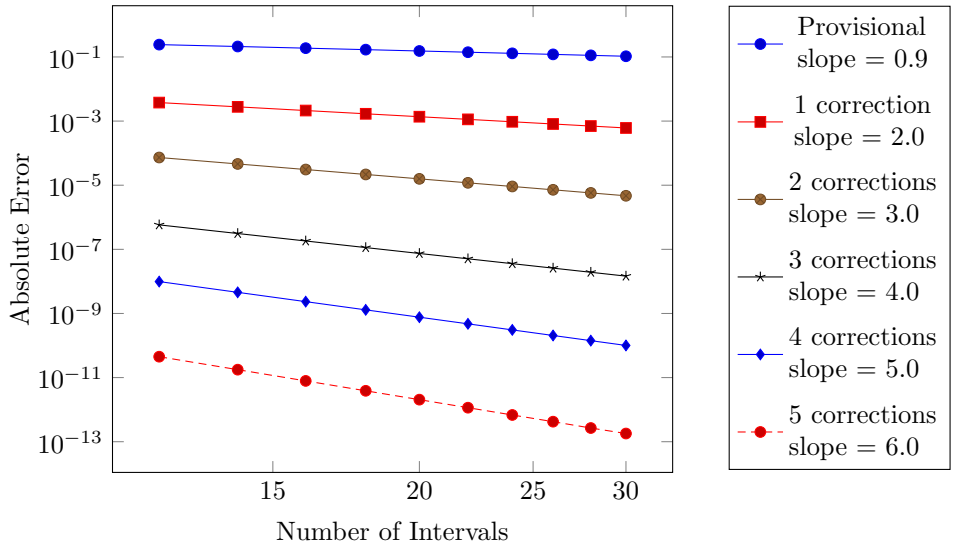


FIG. 9. *Error as a function of the number of intervals for various spectral deferred correction methods. Forward Euler integrators are used to solve test problem 1 and the error equation (12). The method with one and two corrections used three Gauss–Lobatto nodes in each interval. The methods with three and four corrections used four Gauss–Lobatto nodes in each interval. The method with five corrections used five Gauss–Lobatto nodes. The expected orders of accuracy are observed.*

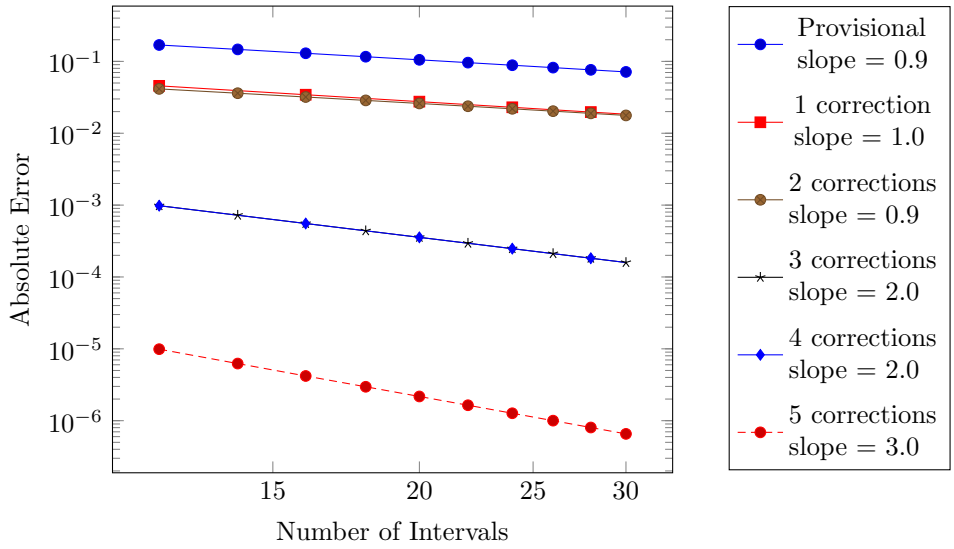


FIG. 10. *Error as a function of the number of intervals for various integral deferred correction methods. Forward Euler integrators are used to solve test problem 1 and the error equation (12). The method with one and two corrections used two Gauss–Legendre nodes in each interval. The methods with three and four corrections used three Gauss–Legendre nodes in each interval. The method with five corrections used four Gauss–Legendre nodes. The order of accuracy of each method is limited by the order of the interpolating polynomial used.*

order $(K+1)$ accuracy.

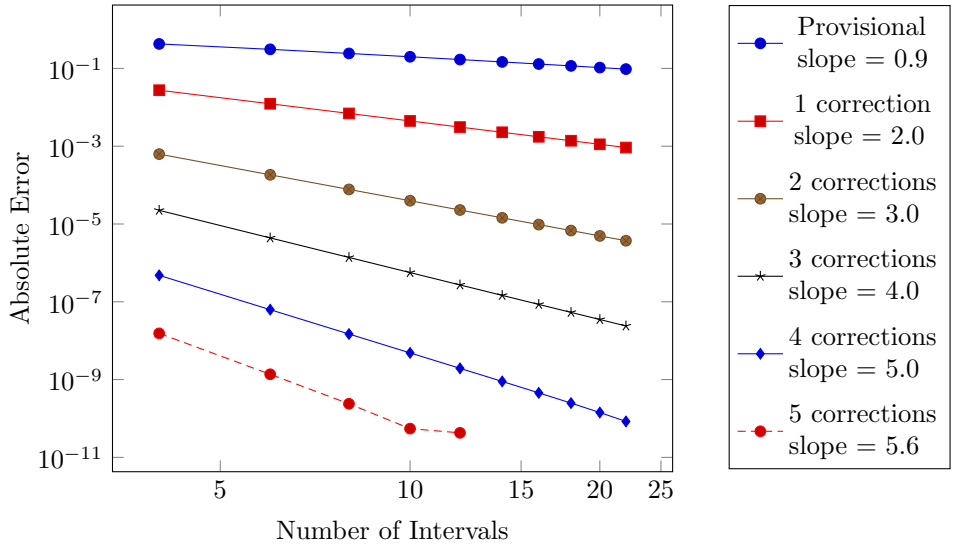


FIG. 11. *Error as a function of the number of intervals for various integral deferred correction methods. Forward Euler integrators are used to solve test problem 1 and the error equation* (12). *Each method with K corrections uses $(K+1)$ quadrature nodes and is able to achieve orders of accuracy $(K+1)$.*

If we instead construct IDC methods using Gauss–Lobatto nodes and higher-order integrators, e.g., the second-order midpoint integrator, we expect to only see a one order of accuracy increase for each correction due to the lack of discrete smoothness of the non-uniform grid. However, interesting behavior is seen when the IDC methods are applied to test problem 1. Similar to the experiment conducted to generate Figure 9, we use four Gauss–Lobatto nodes in each interval for the method with one correction (to allow for the possibility of fourth-order convergence) and five Gauss–Lobatto nodes in each interval for the method with two corrections (to allow for the possibility of sixth-order convergence). Figure 12 shows the convergence behavior of these IDC methods. The method with one correction attains better-than-expected convergence (fourth order instead of third order), the method with two corrections attains the expected order of accuracy (fourth order). The observed results are not in conflict with the analysis in Christlieb et. al [9] because the analysis guarantees only one order of increase per correction. More interestingly, consider IDC methods constructed using Gauss–Legendre and higher-order integrators, e.g., the second-order midpoint integrator. Similar to the experiment conducted to generate Figure 11, we use three Gauss–Lobatto nodes in each interval for the method with one correction (to allow for the possibility of fourth-order convergence) and four Gauss–Lobatto nodes in each interval for the method with two corrections (to allow for the possibility of sixth-order convergence). Figure 13 shows that these IDC methods converge with order P_K, which is higher than the guaranteed order of increase discussed in [9]. Further exploration and analysis into IDC methods constructed using Gauss–Legendre and Chebyshev nodes may be a fruitful research direction.

The behaviors described in this section can be observed for different test problems or other integrators within the CDC/IDC/SDC methods, some of which are described subsequently. The reader is able to numerically explore properties of de-

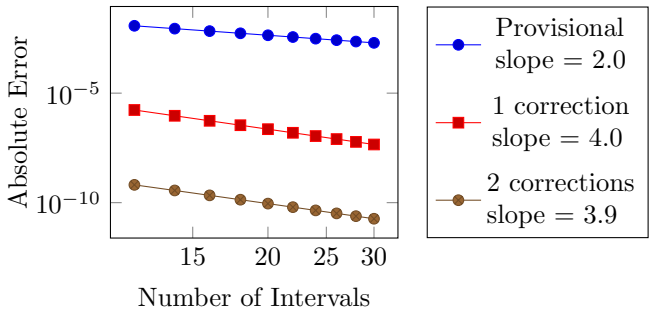


FIG. 12. *Error as a function of the number of intervals for various integral deferred correction methods. The midpoint method is used to solve test problem 1 and the error equation (17). The method with one correction uses four Gauss–Lobatto nodes in each interval; the method with two corrections uses five Gauss–Lobatto nodes in each interval.*

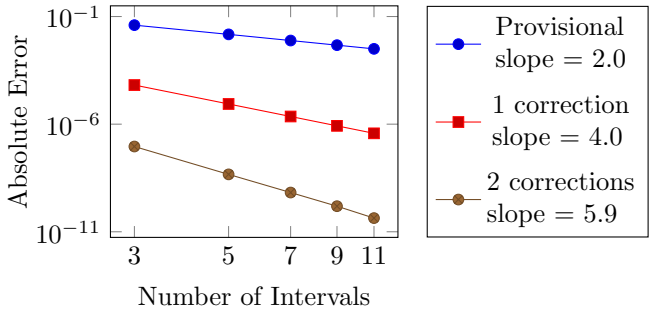


FIG. 13. *Error as a function of the number of intervals for various integral deferred correction methods. The midpoint method is used to solve test problem 1 and the error equation (17). The method with one correction uses three Gauss–Legendre nodes in each interval; the method with two corrections uses four Gauss–Lobatto nodes in each interval.*

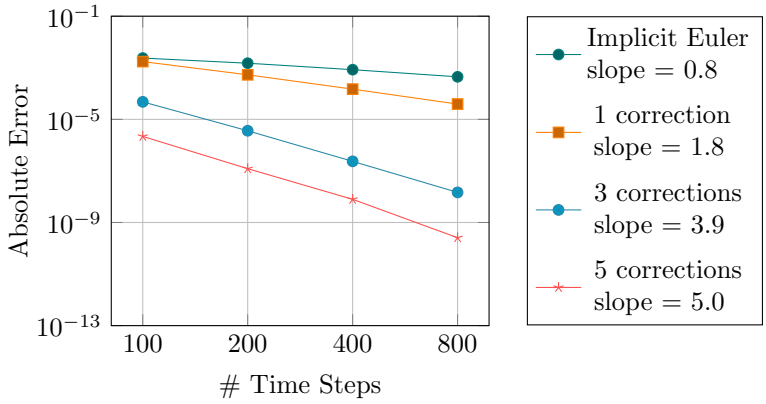
ferred correction by downloading the MATLAB source code used to generate results in this section.[1]

4.2. Pipeline Parallelism. To observe parallel speedup in DC methods, the computational overhead of solving the error equations (i.e., computing the quadrature approximation or interpolant in the respective error equations) must be inexpensive compared to the cost of advancing the provisional solution from time t to $t + \Delta t$. Parallel speedup can be expected when evaluation of the ODE right-hand side is expensive, as in case of N-body problems [10], or when solutions of nonlinear system of equations arise when implicit integrators are used for the predictors or correctors.

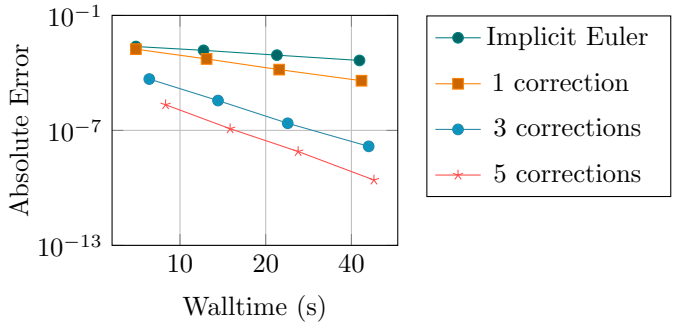
The following numerical experiment demonstrates parallel speedup for the latter case, where a nonlinear system of equations arises when the backward Euler integrator is used to solve the Brusselator equation in $\mathbb{R}^1$, equation (20), and the IDC correction equation, equation (17). The nonlinear system of equations is solved using a Newton iteration. The RIDC software [33] and the GNU Scientific Library (GSL) are used to

[1] The software used to generate numerical results in this manuscript is hosted tempoarily at http://mathgeek.us/code/dcrev/. The final version of the MATLAB scripts/C++ files will be archived on Zenodo with DOI entry upon acceptance of this manuscript, unless the journal has its own mechanism for hosting source code affiliated with a manuscript.

generate the timing results. The scaling studies were performed on a single computational node consisting of a dual socket Intel E5-2680v4 chipset. The interval $[0, 10]$ is used as a time domain, and the spatial domain is divided into 400 subintervals. Figure 14(a) shows a standard convergence study of RIDC. The order increases as more correctors are used. We note that the method with five corrections only achieves fifth order (instead of the anticipated sixth order). Figure 14(b) shows the results from the same numerical experiment but with error plotted against walltime. Here, $(p+1)$ processors are used for RIDC integrators with p correctors. If the data markers line up vertically, then the RIDC method would scale perfectly with no computational overhead. The slight offset in data points can be interpreted as the overhead of the RIDC method: there is a startup and shutdown phase where not all the processors are marching with a full pipeline, as well as the communication/memory access overhead, and the extra flops needed to compute the quadrature and interpolant. In this numerical example, RIDC methods achieve over 90% efficiency when 800 time steps are computed.



(a) Standard convergence study: Error versus number of time steps

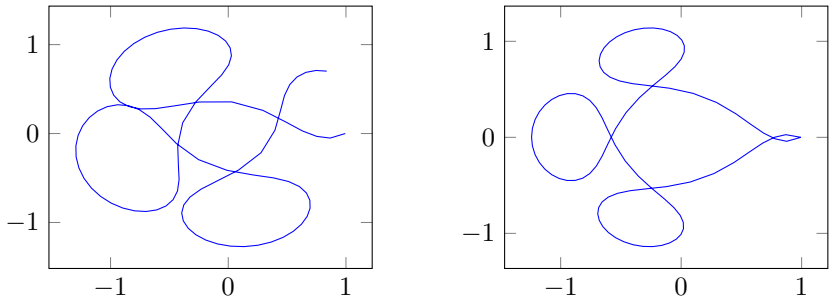


(b) Parallel convergence study: Error versus walltime

FIG. 14. *RIDC integrators (constructed using backward Euler integrators) used to solve the Brusselator equation in* $\mathbb{R}^1$.

4.3. Choice of Interpolant. Lastly, we observe that in most of the deferred correction literature, including the numerical experiments above, the quadrature formulas are obtained by constructing a Lagrange interpolating polynomial based on the

quadrature nodes and then computing the corresponding quadrature weights using the Lagrange polynomial. In some cases, however, it might be useful to utilize other interpolants such as continuous extensions to numerical methods or cubic or Hermite splines. This is potentially useful for pipeline implementations of deferred correction methods. In this numerical experiment, we first solve the Arenstorf orbit problem using classical deferred correction, where uniformly distributed nodes are chosen, and a cubic spline interpolant is constructed, evaluated, and differentiated when solving the error equation, equation (6). Figure 15 shows that although 10^5 time steps are used for the predictor, the orbit is not closed. The CDC corrector (utilizing spline interpolants) is able to correct the orbit to obtain a closer approximation to the expected periodic orbit. Special care must be taken to ensure that the splines provide a sufficiently accurate approximation to the derivative (for classical deferred correction) or the integral (for integral deferred correction).



(a) Predictor (b) 1 Correction

FIG. 15. *Solutions to the Arenstorf orbit problem. A uniform mesh with* 10^5 *time steps is used. The predictor is the forward Euler integrator. The corrector uses the forward Euler intgrator and cubic splines to solve the error equation* (6).

5. Conclusion. In this manuscript, a survey is given of the main classes of deferred correction methods for solving initial-value problems: classical deferred correction, spectral deferred correction, and integral deferred correction. A unified notation is used to review the theoretical underpinnings, and an analysis of task parallelization is presented to demonstrate how integration methods based on deferred correction can be effective solvers on modern computer architectures. Numerical experiments illustrate the order of accuracy that can be expected from deferred correction methods, and a pipeline parallel implementation is reviewed for the integral deferred correction framework. We also highlight that a wider choice of interpolant is possible than is typically discussed in the literature and demonstrate a classical deferred correction method constructed using spline interpolants.

REFERENCES

[1] W. AUZINGER, H. HOFSTÄTTER, W. KREUZER, AND E. WEINMÜLLER, *Modified defect correction algorithms for ODEs. I. General theory*, Numer. Algorithms, 36 (2004), pp. 135–155, https://doi.org/10.1023/B:NUMA.0000033129.73715.7f, http://dx.doi.org/10.1023/B:NUMA.0000033129.73715.7f.

[2] W. AUZINGER, H. HOFSTÄTTER, W. KREUZER, AND E. WEINMÜLLER, *Modified defect correction algorithms for ODEs. II. Stiff initial value problems*, Numer. Algorithms, 40

(2005), pp. 285–303, https://doi.org/10.1007/s11075-005-5327-4, http://dx.doi.org/10.1007/s11075-005-5327-4.

[3] S. Boscarino and J.-M. Qiu, *Error estimates of the integral deferred correction method for stiff problems*, ESAIM Math. Model. Numer. Anal., 50 (2016), pp. 1137–1166, https://doi.org/10.1051/m2an/2015072, http://dx.doi.org/10.1051/m2an/2015072.

[4] A. Bourlioux, A. T. Layton, and M. L. Minion, *High-order multi-implicit spectral deferred correction methods for problems of reactive flow*, J. Comput. Phys., 189 (2003), pp. 651–675, https://doi.org/10.1016/S0021-9991(03)00251-1, http://dx.doi.org/10.1016/S0021-9991(03)00251-1.

[5] S. Bu, J. Huang, and M. L. Minion, *Semi-implicit Krylov deferred correction methods for differential algebraic equations*, Math. Comp., 81 (2012), pp. 2127–2157, https://doi.org/10.1090/S0025-5718-2012-02564-6, http://dx.doi.org/10.1090/S0025-5718-2012-02564-6.

[6] A. Christlieb, W. Guo, M. Morton, and J.-M. Qiu, *A high order time splitting method based on integral deferred correction for semi-Lagrangian Vlasov simulations*, J. Comput. Phys., 267 (2014), pp. 7–27, https://doi.org/10.1016/j.jcp.2014.02.012, http://dx.doi.org/10.1016/j.jcp.2014.02.012.

[7] A. Christlieb, M. Morton, B. Ong, and J.-M. Qiu, *Semi-implicit integral deferred correction constructed with additive Runge-Kutta methods*, Commun. Math. Sci., 9 (2011), pp. 879–902, https://doi.org/10.4310/CMS.2011.v9.n3.a10, http://dx.doi.org/10.4310/CMS.2011.v9.n3.a10.

[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 (2009), pp. 27–56, https://doi.org/10.2140/camcos.2009.4.27, http://dx.doi.org/10.2140/camcos.2009.4.27.

[9] A. Christlieb, B. Ong, and J.-M. Qiu, *Integral deferred correction methods constructed with high order Runge-Kutta integrators*, Math. Comp., 79 (2010), pp. 761–783, https://doi.org/10.1090/S0025-5718-09-02276-5, http://dx.doi.org/10.1090/S0025-5718-09-02276-5.

[10] A. J. Christlieb, C. B. Macdonald, and B. W. Ong, *Parallel high-order integrators*, SIAM J. Sci. Comput., 32 (2010), pp. 818–835, https://doi.org/10.1137/09075740X, http://dx.doi.org/10.1137/09075740X.

[11] A. J. Christlieb, C. B. Macdonald, B. W. Ong, and R. J. Spiteri, *Revisionist integral deferred correction with adaptive step-size control*, Commun. Appl. Math. Comput. Sci., 10 (2015), pp. 1–25, https://doi.org/10.2140/camcos.2015.10.1, http://dx.doi.org/10.2140/camcos.2015.10.1.

[12] K. W. Chu and A. Spence, *Deferred correction for the integral equation eigenvalue problem*, J. Austral. Math. Soc. Ser. B, 22 (1980/81), pp. 474–487, https://doi.org/10.1017/S0334270000002812, http://dx.doi.org/10.1017/S0334270000002812.

[13] A. Dutt, L. Greengard, and V. Rokhlin, *Spectral deferred correction methods for ordinary differential equations*, BIT, 40 (2000), pp. 241–266, https://doi.org/10.1023/A:1022338906936, http://dx.doi.org/10.1023/A:1022338906936.

[14] L. Fox, *Some improvements in the use of relaxation methods for the solution of ordinary and partial differential equations*, Proc. Roy. Soc. London. Ser. A., 190 (1947), pp. 31–59.

[15] L. Fox and E. T. Goodwin, *Some new methods for the numerical integration of ordinary differential equations*, Proc. Cambridge Philos. Soc., 45 (1949), pp. 373–388.

[16] S. Güttel and G. Klein, *Efficient high-order rational integration and deferred correction with equispaced data*, Electron. Trans. Numer. Anal., 41 (2014), pp. 443–464.

[17] T. Hagstrom and R. Zhou, *On the spectral deferred correction of splitting methods for initial value problems*, Commun. Appl. Math. Comput. Sci., 1 (2006), pp. 169–205, https://doi.org/10.2140/camcos.2006.1.169, http://dx.doi.org/10.2140/camcos.2006.1.169.

[18] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving ordinary differential equations. I*, vol. 8 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, second ed., 1993. Nonstiff problems.

[19] E. Hairer and G. Wanner, *Solving ordinary differential equations. II*, vol. 14 of Springer Series in Computational Mathematics, Springer-Verlag, Berlin, second ed., 1996, https://doi.org/10.1007/978-3-642-05221-7, http://dx.doi.org/10.1007/978-3-642-05221-7. Stiff and differential-algebraic problems.

[20] A. C. Hansen and J. Strain, *On the order of deferred correction*, Appl. Numer. Math., 61 (2011), pp. 961–973, https://doi.org/10.1016/j.apnum.2011.04.001, http://dx.doi.org/10.1016/j.apnum.2011.04.001.

[21] J. Huang, J. Jia, and M. Minion, *Accelerating the convergence of spectral deferred correction methods*, J. Comput. Phys., 214 (2006), pp. 633–656, https://doi.org/10.1016/j.jcp.2005.10.004, http://dx.doi.org/10.1016/j.jcp.2005.10.004.

[22] J. Huang, J. Jia, and M. Minion, *Arbitrary order Krylov deferred correction methods for differential algebraic equations*, J. Comput. Phys., 221 (2007), pp. 739–760, https://doi.org/10.1016/j.jcp.2006.06.040, http://dx.doi.org/10.1016/j.jcp.2006.06.040.

[23] W. Kress and B. Gustafsson, *Deferred correction methods for initial boundary value problems*, in Proceedings of the Fifth International Conference on Spectral and High Order Methods (ICOSAHOM-01) (Uppsala), vol. 17, 2002, pp. 241–251, https://doi.org/10.1023/A:1015113017248, http://dx.doi.org/10.1023/A:1015113017248.

[24] A. T. Layton and M. L. Minion, *Conservative multi-implicit spectral deferred correction methods for reacting gas dynamics*, J. Comput. Phys., 194 (2004), pp. 697–715, https://doi.org/10.1016/j.jcp.2003.09.010, http://dx.doi.org/10.1016/j.jcp.2003.09.010.

[25] A. T. Layton and M. L. Minion, *Implications of the choice of quadrature nodes for Picard integral deferred corrections methods for ordinary differential equations*, BIT, 45 (2005), pp. 341–373, https://doi.org/10.1007/s10543-005-0016-1, http://dx.doi.org/10.1007/s10543-005-0016-1.

[26] 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 (2007), pp. 1–34, https://doi.org/10.2140/camcos.2007.2.1, http://dx.doi.org/10.2140/camcos.2007.2.1.

[27] J.-L. Lions, Y. Maday, and G. Turinici, *Résolution d'EDP par un schéma en temps "pararéel"*, C. R. Acad. Sci. Paris Sér. I Math., 332 (2001), pp. 661–668, https://doi.org/10.1016/S0764-4442(00)01793-6, http://dx.doi.org/10.1016/S0764-4442(00)01793-6.

[28] F. Liu and J. Shen, *Stabilized semi-implicit spectral deferred correction methods for Allen-Cahn and Cahn-Hilliard equations*, Math. Methods Appl. Sci., 38 (2015), pp. 4564–4575, https://doi.org/10.1002/mma.2869, http://dx.doi.org/10.1002/mma.2869.

[29] Y. Liu, C.-W. Shu, and M. Zhang, *Strong stability preserving property of the deferred correction time discretization*, J. Comput. Math., 26 (2008), pp. 633–656.

[30] M. L. Minion, *Semi-implicit spectral deferred correction methods for ordinary differential equations*, Commun. Math. Sci., 1 (2003), pp. 471–500, http://projecteuclid.org/euclid.cms/1250880097.

[31] M. L. Minion, *Semi-implicit projection methods for incompressible flow based on spectral deferred corrections*, Appl. Numer. Math., 48 (2004), pp. 369–387, https://doi.org/10.1016/j.apnum.2003.11.005, http://dx.doi.org/10.1016/j.apnum.2003.11.005. Workshop on Innovative Time Integrators for PDEs.

[32] M. L. Minion, *A hybrid parareal spectral deferred corrections method*, Commun. Appl. Math. Comput. Sci., 5 (2010), pp. 265–301.

[33] B. W. Ong, R. D. Haynes, and K. Ladd, *Algorithm 965: RIDC methods: A family of parallel time integrators*, ACM Trans. Math. Softw., 43 (2016), pp. 8:1–8:13, https://doi.org/10.1145/2964377, http://doi.acm.org/10.1145/2964377.

[34] V. Pereyra, *On improving an approximate solution of a functional equation by deferred corrections*, Numer. Math., 8 (1966), pp. 376–391.

[35] V. Pereyra, *Iterated deferred corrections for nonlinear boundary value problems*, Numer. Math., 11 (1968), pp. 111–125.

[36] W. Qu, N. Brandon, D. Chen, J. Huang, and T. Kress, *A numerical framework for integrating deferred correction methods to solve high order collocation formulations of ODEs*, J. Sci. Comput., 68 (2016), pp. 484–520, https://doi.org/10.1007/s10915-015-0146-9, http://dx.doi.org/10.1007/s10915-015-0146-9.

[37] A. V. Rangan, *Adaptive solvers for partial differential and differential-algebraic equations*, PhD thesis, University of California, Berkeley, 2003, http://gateway.proquest.com/openurl?url_ver=Z39.88-2004&rft_val_fmt=info:ofi/fmt:kev:mtx:dissertation&res_dat=xri:pqdiss&rft_dat=xri:pqdiss:3121657. Thesis (Ph.D.)–University of California, Berkeley.

[38] D. Ruprecht and R. Speck, *Spectral deferred corrections with fast-wave slow-wave splitting*, SIAM J. Sci. Comput., 38 (2016), pp. A2535–A2557, https://doi.org/10.1137/16M1060078, http://dx.doi.org/10.1137/16M1060078.

[39] R. D. Skeel, *A theoretical framework for proving accuracy results for deferred corrections*, SIAM J. Numer. Anal., 19 (1982), pp. 171–196, https://doi.org/10.1137/0719009, http://dx.doi.org/10.1137/0719009.

[40] G. Söderlind, L. Jay, and M. Calvo, *Stiffness 1952–2012: sixty years in search of a definition*, BIT, 55 (2015), pp. 531–558, https://doi.org/10.1007/s10543-014-0503-3, http://dx.doi.org/10.1007/s10543-014-0503-3.

[41] R. Speck, D. Ruprecht, M. Emmett, M. Minion, M. Bolten, and R. Krause, *A multi-level spectral deferred correction method*, BIT, 55 (2015), pp. 843–867, https://doi.org/10.1007/s10543-014-0517-x, http://dx.doi.org/10.1007/s10543-014-0517-x.

[42] H. J. Stetter, *The defect correction principle and discretization methods*, Numer. Math., 29 (1977/78), pp. 425–443.

[43] G. Sun, *The deferred correction procedure for linear multistep formulas*, J. Comput. Math., 3 (1985), pp. 41–49.

[44] L. N. Trefethen, *Spectral methods in MATLAB*, vol. 10 of Software, Environments, and Tools, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000, https://doi.org/10.1137/1.9780898719598, http://dx.doi.org/10.1137/1.9780898719598.

[45] J. Xin, J. Huang, W. Zhao, and J. Zhu, *A spectral deferred correction method for fractional differential equations*, Abstr. Appl. Anal., (2013), pp. Art. ID 139530, 6.

[46] P. E. Zadunaisky, *A method for the estimation of errors propagated in the numerical solution of a system of ordinary differential equations*, in The Theory of Orbits in the Solar System and in Stellar Systems, G. I. Kontopoulos, ed., vol. 25 of IAU Symposium, 1966, pp. 281–287.

[47] P. E. Zadunaisky, *On the estimation of errors propagated in the numerical integration of ordinary differential equations*, Numer. Math., 27 (1976/77), pp. 21–39.