Journal of Scientific Computing manuscript No.
(will be inserted by the editor)

Deferred Correction Methods for Ordinary Differential Equations

Benjamin W. Ong · Raymond J. Spiteri

the date of receipt and acceptance should be inserted later

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 in a unified manner, specifically the classical algorithm of Zadunaisky/Stetter, the method of Dutt, Greengard, and Rokhlin, spectral deferred correction, and integral deferred correction. We highlight some nuances of their implementations, including the choice of quadrature nodes, 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. Lightweight and flexible Matlab software is provided for exploration with modern variants of deferred correction methods.

Keywords ordinary differential equations · initial-value problems · deferred correction · parallel computing · method of lines · multi-core computing

Mathematics Subject Classification (2010) 65B05 · 65L05 · 65L06 · 65M20 · 65Y05

1 Introduction

Differential equations have proven to be a powerful tool for mathematically describing the evolution of physical systems since the days of Newton and Leibniz in the late 1600s. Unbeknownst to many, the solutions to differential equations

B. W. Ong
Michigan Technological University, Department of Mathematical Sciences, Houghton, MI, 49931, USA. E-mail: ongbw@mtu.edu

R. J. Spiteri
University of Saskatchewan, Department of Computer Science, 176 Thorvaldson Building, 110 Science Place, Saskatoon, Saskatchewan, S7N 5C9, Canada. E-mail: spiteri@cs.usask.ca

significantly impact our daily lives, from the shapes of our planes, trains, and automobiles to how our financial markets behave to what we predict the weather will be. More specialized applications of differential equations range from modeling quantum mechanical systems to infectious disease spread to the universe at large. Many of these models take the form of partial differential equations (PDEs), and many others take the form of ordinary differential equations (ODEs). Because the nature of most differential equations is such that solutions can only be obtained numerically, the so-called method of lines, which semi-discretizes a PDE in all but one of its independent variables, is an abundant source of ODEs that arise from the process of numerically solving PDEs.

There is increasing demand for highly accurate solutions to differential equations, and with that comes a corresponding increasing demand for efficient numerical methods. One of the most efficient ways to generate highly accurate solutions to ODEs is through the use of high-order methods. Unfortunately, high-order methods can be difficult to derive and much more involved to implement than low-order ones. This has made researchers ponder whether there is a way to increase the accuracy of the results of their functioning low-order codes without the need of a new algorithm or an increase in resolution (and hence computational costs and times to solution).

Deferred correction (DC) methods are well-established methods for constructing 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) or boundary-value problem (BVP) for a system of ODEs is computed and then subsequently refined by solving a sequence of related problems. Under suitable assumptions, this process can be repeated to produce solutions with an arbitrarily high order of accuracy. We use the terminology "DC method" to generally refer to the process of refining the numerical solution to an ODE by iteration. This includes methods that have been referred to in other contexts as difference correction methods or defect correction methods; see also below. Classically, they are portrayed as *acceleration methods*, as in accelerating the convergence of the numerical method to the exact solution. Such acceleration methods include Richardson extrapolation. DC methods have the advantage, however, that they use only one mesh.

DC methods have been used simulate a wide range of systems, including plasmas [15], flames and other low Mach number reacting flows [52], high Reynolds number incompressible Navier–Stokes equations [1], and compressible flows associated with climate and numerical weather prediction [57]. Such methods have also been used in the optimal control of aircraft flight [8], the solution of forward-backward stochastic differential equations [68], and as geometric integrators [35].

DC methods were originally proposed by Fox in the 1940s [23] as *difference correction methods*. His influence seems to be the most prevalent in this first era of DC 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 followed by a correction term of higher-order differences that is then used to calculate a new solution. This correction turns out to be an estimate of the local discretization error [59]. 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 domain of integration, and high-order differences typically suffer from significant cancellation, leading to non-negligible round-off errors. In [23], deferred correction is applied to BVPs for ODEs and eigenvalue problems for ODEs and PDEs. The approach is applied to IVPs by Fox and Goodwin in [24].

Pereyra generalizes deferred correction to functional equations in [53] and considers specific examples in the solution of BVPs in [54]. Skeel in [59] credits Pereyra with the resurgence of a second era of DC methods in the 1960s. Pereyra offered a fresh perspective on DC methods in terms of local truncation error estimation and provided them with a relatively general and rigorous foundation.

Zadunaisky discusses in [75,76] a method for estimating the global error in a numerical solution to an IVP or BVP, an idea that now forms the basis for *defect correction*. Given a (continuous) 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 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 global error in $\widetilde{\mathbf{y}}(t)$, and hence $\widetilde{\mathbf{y}}(t)$ can be corrected. The method was different from [53] in that it did not involve calculating local truncation errors. The method requires the interpolation of the numerical solution, however, and is prone to problems when using large equi-spaced grids. Such problems are mitigated by dividing the domain of integration into small groups of intervals and performing the interpolation on them [76]. In [66], 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. This is the algorithm that we refer to here as *classical deferred correction* (CDC), with acknowledgments as well to important work by Frank and Ueberhuber right around that time. Stetter also contributed important ideas on global error estimation for IVPs in [66]. This seems to have been the starting point for Lindberg's work [41].

Besides reviewing the history of DC methods and related methods up to the time of its publication, Skeel in [59] provides general theoretical techniques for DC solutions to DEs that simplify and strengthen work initiated by Lindberg [41] on the numerical solution of operator equations. The major contribution seems to be a shift in emphasis from assuming the existence of asymptotic expansions of the global error in terms of the step size to its smoothness in terms of discrete Sobolev norms. This is a significant departure from Pereyra's approach of basing local error estimates on local error expansions.

DC methods have been extensively applied to IVPs [24,20,13,14,16,34] and BVPs [23,53,54] for ODEs, initial-boundary value problems for PDEs [23,37,56], differential-algebraic equations [56,36], and eigenvalue problems [24,19]. They have been used in conjunction with linear multistep methods [67], Krylov subspace methods [36,11,36], and splitting methods [30]. 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 [28]. Here, we also discuss the use of interpolants based on continuous extensions of numerical methods as well as splines.

In [20], Dutt, Greengard, and Rokhlin attempt to encapsulate the state of the art of DC methods at the time by proposing a "classical" deferred correction method, which is similar to (but not exactly) Zadunaisky's approach [76]. In view of this, Hansen and Strain [34] argue CDC is misleadingly named and call it

the DGR method after its proposers, and we follow suit. Despite its inauspicious introduction, we describe the DGR method in detail because it is a widely known method and it has the desirable feature of not requiring computation of the system Jacobian in its formulation. In [20], Dutt, Greengard, and Rokhlin also propose, however, an important variant of deferred correction, which they called *spectral deferred correction* (SDC), that bases the correction step on the Picard integral form of the solution to the IVP. Layton and Minion [39] argue SDC is also misleadingly named, citing the fact that although the quadrature rule over the entire interval is spectral, the intermediate quadratures over sub-intervals are not. Although in agreement with Theorem 4.1 of the original SDC paper [20], this argument was not completely rigorous. By viewing SDC in the framework of implicit RK methods, Hagstrom and Zou [30] show how full spectral accuracy can be achieved. The original motivation for using the Picard form is the increased stability. With this approach, however, it is possible to use locally non-uniformly spaced nodes in each integration subinterval, and convergence of the iteration is still guaranteed, in contrast to CDC, where convergence and order of accuracy typically break down in such cases [2]. Dutt, Greengard, and Rokhlin investigate convergence orders as high as 20 on test problems but limit the integrators to forward and backward Euler [20], a choice that turns out to be optimal in a sense described below. Indeed, [20] seems to have kicked off the third and most recent resurgence in research interest and developments in DC methods.

DC methods, and in particular SDC methods, can be designed to have some appealing properties. In [45], Liu *et al.* explore the strong-stability-preserving property in the context of SDC. In [38], a finite volume approach is used to discretize the compressible Navier–Stokes equations in order to produce a conservative method. Winkel *et al.* describe a high-order Boris integrator based on SDC for models in plasma physics in [73]. SDC was applied to fractional differential equations in [74].

Recent research has shown that under certain conditions DC methods [2], [3] and SDC methods [30] can be viewed as approximations to implicit Runge–Kutta (IRK) methods. Collocation methods are known to be equivalent to a special class of IRK methods (see [31, Chap. II Thm 7.7]), and by construction they have zero residual. Auzinger *et al.* in [2], [3] show that DC methods are iterative collocation solvers and, with certain choices of nodes, can exhibit superconvergence. In [30], 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. When solving IVPs by IRK methods, the iterative nature of DC methods allows them to offer convenient initial values for the solution of the stage / solution values at each step, hence promoting rapid convergence. The use of higher-order provisional solutions before the application of SDC iterations is explored in [40], where the backward Euler method is used to improve the order of accuracy by one at each SDC iteration. Other studies of the convergence of DC solutions to collocation solutions include [36], [71], [57], with subsequent insights into choosing quadrature methods presented in [55]. A more general 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 [39].

In [12], the convergence properties of SDC methods are analyzed by interpreting an SDC method as a Picard iteration. This differs from conventional approaches that view SDC methods as iteratively correcting provisional solutions by

solving an error equation. They show that the choice of the base solver need not be consistent with the underlying ODE to obtain a convergent method.

An important practical contribution to obtaining high-order numerical solutions of ODEs came from the work of Minion and co-workers, e.g., [9,46,47,38, 40]. The importance of these works was the practical demonstration of how high-order numerical solutions to N-additive problems with $N \geq 2$ could be obtained from low-order methods despite multiple splitting errors and without the need to solve order conditions. Analogous high-order methods were not readily available at the time, and this propelled the initial interest in SDC methods. SDC has been implemented in a semi-implicit manner for the solution of incompressible flows in [47], the Allen–Cahn and Cahn–Hilliard equations in [44], and the compressible Boussinesq equation in [57], where the splitting was based on wave speeds. SDC was applied in a multi-level framework for solving PDEs in [63], where spatial coarsening was used for generating predictors (initial approximations; *provisional* or *uncorrected* solutions). Multi-level SDC methods were combined with spherical harmonics to produce high-order implicit-explicit (IMEX) methods and used to solve wave propagation problems arising from solving the shallow water equations on a sphere [33]. Order reduction associated with semi-implicit SDC was observed in [46]. SDC was used in a multi-implicit manner in [9,38]. In [9], 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. Of particular note to parallel-in-time integrators, in [48] SDC was incorporated as the "fine propagator" in the parareal framework [43]. The move into parallel computing represents another significant development in the practical use of SDC methods and is elaborated upon below. SDC was also shown to be tolerant to soft hardware data faults in [27].

More recently, Christlieb *et al.* describe another variant of DC called *integral deferred correction* (IDC) that allows for higher-order single-step integrators to be used [13,14,18]. Operator splitting techniques (such as Lie–Trotter and Strang operator splitting), alternating direction implicit (ADI) methods, and IMEX methods were introduced in the IDC context in [18] and applied to the Vlasov–Poisson problem in [15]. IDC methods were applied to singular perturbation problems in [6]. Analysis of order reduction in IDC methods based on IMEX-RK methods and applied to singular perturbation problems was carried out in [7].

A particularly important recent advancement in the resurgence of DC methods has been with respect to their time parallelism. In this regard, there have been two main lines of research.

In the first line of research, Christlieb *et al.* demonstrate in [16] 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 [17].

In the second line of research, Minion and Williams [50] and Minion [48] proposed a method for parallelizing the numerical solution of ODEs based on using SDC within a parareal [42] framework. Emmett and Minion [22] then went on to combine the parallel (P) use of SDC within parareal algorithm with the full approximation scheme (FAS) from multi-grid, e.g., [29,10], space-time (ST) to construct parallel integrators for PDEs to form an algorithm known as PFASST. The central idea is that PFASST employs a FAS correction on coarse grids using information from fine grid SDC sweeps to efficiently enhance convergence. Coarsening is done in both space and time, and coarse and fine time integrations can be done in parallel, leading to a method that was shown to scale reasonably well on some benchmark problems in [22]. Since then, PFASST has become the hallmark parallel-in-time integration method, with numerous papers being published on the topic. A short list of such papers includes [64,49,62,4,5,26,61] and further references therein.

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

$$\frac{d}{dt}\mathbf{y}(t) = \mathbf{f}(t, \mathbf{y}(t)), \quad a < t < b, \tag{1.1a}$$

$$\mathbf{y}(a) = \mathbf{y}_a, \tag{1.1b}$$

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 focus of this study is the application of DC methods to problems that may best be described as *moderately stiff*. High-order semi-implicit SDC methods were reported to be more efficient than semi-implicit (additive or IMEX) Runge–Kutta methods in [47]. Highly stiff problems present a unique set of challenges. Boscarino and Qiu [6] perform an error analysis on IMEX-RK-IDC methods applied to (highly stiff) singular perturbation problems and analyze the phenomenon of order reduction. A full discussion of the behavior of DC methods applied to such problems is beyond the scope of this paper.

The remainder of this paper is structured as follows. In section 2, we review some of the theoretical background necessary for the understanding of deferred correction methods and establish a unified notation. In section 3, we review the classical defect correction method of Zadunaisky / Stetter, the DGR method, spectral deferred correction, integral deferred correction, and revisionist integral deferred correction. We offer some non-traditional ideas for implementation, including the choice of quadrature nodes, interpolants, and combinations of discretization methods, that illustrate the flexibility of deferred correction methods, with particular emphasis on the RIDC formulation for parallel architectures. In section 4, we discuss our numerical testing of these ideas and the results on some benchmark problems using lightweight and flexible Matlab software. In section 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 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 degree $(n-1)$ for the points $\mathbf{Y}$ at any point $t \in \mathbb{R}$ by the familiar formula

$$\mathbf{L}_n(t; \mathbf{Y}) = \sum_{i=1}^{n} \ell_i(t) \cdot \mathbf{y}_i, \tag{2.1}$$

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

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

This interpolant has approximation order n.

It is well known that approximations to 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 operator) 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

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

holds for all $\mathbf{Y}$, and is defined to be the *differentiation operator*.

Definition 2 (Integration operator) 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

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

holds for all $\mathbf{Y}$ and is defined to be the *integration operator*.

We have defined the lower bound of the integral in Definition 2 to be -1 in anticipation of using Gaussian quadrature nodes (e.g., 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

$$\bar{\tau}_i = \frac{b-a}{2} \cdot \tau_i + \frac{b+a}{2}, \quad i = 1, 2, \ldots, n. \tag{2.2}$$

If $\mathbf{D}_n$ and $\mathbf{Q}_n$ are computed using Gaussian nodes, they are called the *spectral* differentiation and integration operators, 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 with i. The Chebyshev nodes for the scaled and shifted problem on $[a,b] \subset \mathbb{R}$ are formed as in (2.2).

The effect of operators $\mathbf{D}_n$ and $\mathbf{Q}_n$ may of course be implemented as right multiplication by the appropriate matrices.

3 Classes of Deferred Correction Methods

We now describe some of the main classes of deferred correction methods, in particular some that have attracted the most recent attention from researchers. 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,

$$a = t_0 < t_1 < \cdots < t_j < \cdots < t_J = b. \tag{3.1}$$

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

$$t_j \le t_{j,1} < t_{j,2} < \cdots < t_{j,n} \le t_{j+1} \quad j = 0, 1, \ldots, J-1. \tag{3.2}$$

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 $[t_j, t_{j+1}]$, a method of order p_0 is used to determine a provisional solution for (1.1), $\mathbf{Y}_j^{[0]} = (\mathbf{y}_{j,1}^{[0]}, \ldots, \mathbf{y}_{j,n}^{[0]})$, where

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

where $\Delta t = t_{j+1} - t_j$ is the step size. This solution is then corrected in an iterative manner. Denoting the continuous approximation to $\mathbf{y}(t)$ based on interpolation of the discrete solution $\mathbf{Y}_j^{[k]}$ on interval $[t_j, t_{j+1}]$ and at iteration (correction level) k

by $\mathbf{Y}_j^{[k]}(t) = (\mathbf{y}_{j,1}^{[k]}(t), \ldots, \mathbf{y}_{j,n}^{[k]}(t))$, the *error function* at correction level k is defined by

$$\mathbf{e}_j^{[k]}(t, \mathbf{Y}_j^{[k]}(t)) = \mathbf{y}(t) - \mathbf{Y}_j^{[k]}(t), \quad t \in [t_j, t_{j+1}], \tag{3.3}$$

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

3.1 Classical Deferred Correction (CDC)

Although there is some room for debate, in this paper we refer to the global error estimation method using defect correction, as described by Zadunaisky in [76], combined with its generalization to iterative defect correction, as described generally by Stetter [66], as classical deferred correction (CDC).

CDC begins by finding a numerical solution to (1.1) and forming a continuous provisional solution $\mathbf{Y}^{[0]}(t)$. Then differentiating the error function (3.3), we form the error IVP

$$\begin{aligned}\frac{d}{dt}\mathbf{e}_j^{[k]}(t, \mathbf{Y}_j^{[k]}(t)) &= \frac{d}{dt}\mathbf{y}(t) - \frac{d}{dt}\mathbf{Y}_j^{[k]}(t) \\ &= \mathbf{f}\big(t, \mathbf{e}_j^{[k]}(t, \mathbf{Y}_j^{[k]}(t)) + \mathbf{Y}_j^{[k]}(t)\big) - \frac{d}{dt}\mathbf{Y}_j^{[k]}(t), \qquad (3.4a)\\ \mathbf{e}_j^{[k]}(t_j, \mathbf{Y}_j^{[k]}(t_j)) &= \mathbf{0}. \qquad (3.4b)\end{aligned}$$

The method proposed by Zadunaisky [76] linearizes (3.4) and performs one iteration of the system

$$\begin{aligned}\frac{d}{dt}\mathbf{e}_j^{[k]}(t, \mathbf{Y}_j^{[k]}(t)) &= \boldsymbol{J_f}\big(t, \mathbf{Y}_j^{[k]}(t)\big)\mathbf{e}_j^{[k]}(t, \mathbf{Y}_j^{[k]}(t)) - \boldsymbol{\delta}^{[k]}(t, \mathbf{Y}^{[k]}(t)), \qquad (3.5a)\\ \mathbf{e}_j^{[k]}(t_j, \mathbf{Y}_j^{[k]}(t_j)) &= \mathbf{0}. \qquad (3.5b)\end{aligned}$$

where

$$\boldsymbol{\delta}^{[k]}(t, \mathbf{Y}^{[k]}(t)) = \frac{d}{dt}\mathbf{Y}^{[k]}(t) - \mathbf{f}(t, \mathbf{Y}^{[k]}(t)) \tag{3.6}$$

is the *defect* function and

$$\boldsymbol{J_f}(t, \mathbf{y}) = \frac{\partial \mathbf{f}}{\partial \mathbf{y}}(t, \mathbf{y})$$

is the *Jacobian* of (1.1a). Stetter then proposed a general iteration of (3.5) over k.

We solve the error equation at correction level 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 then form the corrected solution

$$\mathbf{Y}_j^{[k]} = \mathbf{Y}_j^{[k-1]} + \mathbf{E}_j^{[k-1]}, \qquad k = 1, 2, \ldots, K,$$

extrapolating the interpolating polynomial to provide values at $t = t_{j+1}$, if desired.

The procedure for CDC is given in (3.1). It is understood that the algorithm is iterated completely on each group of nodes $[t_{j,1}, t_{j,n}]$, $j = 0, 1, \ldots, J-1$, before moving on to the next. If the order of the interpolant is sufficiently high and

uniformly spaced nodes are used within each interval, the solution after this procedure is accurate to order $O((\Delta t)^{P_K})$, where $P_K = \sum_{k=0}^{K} p_k$ [58], [2]. More specifically, however, if the order of accuracy of $\mathbf{Y}^{[k]}(t)$ is n (e.g., using Lagrange polynomial interpolation with n points), then the CDC solution has order of accuracy

$$O((\Delta t)^{\min(P_K, n-1)}) \tag{3.7}$$

because it uses the differentiated interpolant. In practice, the quality of the estimate is also influenced by the stability of the interpolant. For example, the "catastrophically bad" idea [69, p. 42] of high-order interpolation at equally spaced points generally gives poor results despite the formally high order of accuracy. Accordingly, the continuous solution $\mathbf{Y}^{[k]}(t)$ at level k is formed in practice as a piecewise polynomial (i.e., a *spline*) by concatenating interpolating polynomials every "few" steps (say no more than 10). In such cases, CDC can produce acceptable results. This is illustrated in section 4. Schild [58] proposed a modification of CDC to reach spectral accuracies in the context of BVPs. The idea is that the basic method is used to solve (1.1) on a uniform grid on each interval, whereas the defect is evaluated at spectral nodes, a notion reminiscent of SDC; see section 3.3. Related work can also be found in [2], [3].

Algorithm 3.1 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 (1.1) 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 (3.5) 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** $\mathbf{Y}_j^{[K]}(t_{j+1})$.

3.2 The Method of Dutt, Greengard, and Rokhlin (DGR)

As defined in [20], the DGR method uses a sequence of IVPs for the error based on (3.3) to continually improve a given approximate solution, $\mathbf{Y}_j^{[0]}(t)$, in each interval. Rather than linearizing (3.4) to obtain (3.5), the DGR method solves (3.4) directly. This is the key theoretical difference, leading to the high-level algorithmic difference with CDC, occuring in line 5 of Algorithm (3.1).

3.3 Spectral Deferred Correction (SDC)

Like CDC and DGR, SDC can be interpreted as using a sequence of IVPs to continually improve an initial approximate solution $\mathbf{Y}_j^{[0]}(t)$ in an interval $[t_j, t_{j+1}]$. The

equation used for the error, however, is based on the Picard integral formulation of the solution in each interval,

$$\mathbf{y}_j(t) = \mathbf{y}(t_j) + \int_{t_j}^{t} \mathbf{f}(t', \mathbf{y}(t'))\,\mathrm{d}t', \quad t \in [t_j, t_{j+1}]. \tag{3.8}$$

Given a continuous approximation $\mathbf{Y}_j^{[k]}(t)$ to the solution of (3.8), we define the *residual function*,

$$\mathbf{r}_j^{[k]}(t, \mathbf{Y}_j^{[k]}(t)) = \mathbf{Y}_j^{[k]}(t_j) + \int_{t_j}^{t} \mathbf{f}(t', \mathbf{Y}_j^{[k]}(t'))\,\mathrm{d}t' - \mathbf{Y}_j^{[k]}(t). \tag{3.9}$$

The error function can then be expressed as

$$\begin{aligned}\mathbf{e}_j^{[k]}(t, \mathbf{Y}_j^{[k]}(t)) &= \mathbf{y}_j(t) - \mathbf{Y}_j^{[k]}(t) \qquad (3.10)\\ &= \int_{t_j}^{t} \left[\mathbf{f}\big(t', \mathbf{Y}_j^{[k]}(t') + \mathbf{e}_j^{[k]}(t', \mathbf{Y}_j^{[k]}(t'))\big) - \mathbf{f}(t', \mathbf{Y}_j^{[k]}(t'))\right] \mathrm{d}t' + \mathbf{r}_j^{[k]}(t, \mathbf{Y}_j^{[k]}(t)).\end{aligned}$$

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

$$\boldsymbol{\Delta}\mathbf{f}^{[k]}(t, \mathbf{e}_j^{[k]}(t, \mathbf{Y}_j^{[k]}(t))) = \mathbf{f}\big(t, \mathbf{Y}_j^{[k]}(t) + \mathbf{e}_j^{[k]}(t, \mathbf{Y}_j^{[k]}(t))\big) - \mathbf{f}(t, \mathbf{Y}_j^{[k]}(t)). \tag{3.11}$$

Then, (3.10) can be rewritten in a Picard integral form like (3.8),

$$\mathbf{e}_j^{[k]}(t, \mathbf{Y}_j^{[k]}(t)) = \int_{t_j}^{t} \boldsymbol{\Delta}\mathbf{f}^{[k]}(t', \mathbf{e}_j^{[k]}(t'))\,\mathrm{d}t' + \mathbf{r}^{[k]}(t, \mathbf{Y}_j^{[k]}(t)). \tag{3.12}$$

To compute the error $\mathbf{E}_j^{[k-1]} = (\mathbf{e}_{j,1}^{[k-1]}, \ldots, \mathbf{e}_{j,n}^{[k-1]})$, we first compute the $m \times n$ residual matrix $\mathbf{R}_j(\mathbf{Y}^{[k]})$ by using quadrature to approximate the integral in (3.9),

$$\begin{aligned}\mathbf{R}_j^{[k]}(\mathbf{Y}_j^{[k]}) &= \big(\mathbf{r}_j^{[k]}(t_{j,1}, \mathbf{y}_{j,1}^{[k]}), \mathbf{r}^{[k]}(t_{j,2}, \mathbf{y}_{j,2}^{[k]}), \ldots, \mathbf{r}^{[k]}(t_{j,n}, \mathbf{y}_{j,n}^{[k]})\big) \qquad (3.13)\\ &\approx \tilde{\mathbf{Y}}_j^{[k]} + \mathbf{Q}_n \mathbf{F}_j^{[k]} - \mathbf{Y}_j^{[k]},\end{aligned}$$

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

$$\tilde{\mathbf{Y}}_j^{[k]} = \begin{cases} (\mathbf{y}_a, \mathbf{y}_a, \ldots, \mathbf{y}_a), & j = 1, \\ (\mathbf{Y}_{j-1}^{[k]}(t_j), \mathbf{Y}_{j-1}^{[k]}(t_j), \ldots, \mathbf{Y}_{j-1}^{[k]}(t_j)), & j > 1, \end{cases} \tag{3.14}$$

and the matrix $\mathbf{F}_j^{[k]}$ 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 error is then typically computed by applying the left-hand rule or right-hand rule quadrature rule to (3.12). The left-hand rule yields

$$\mathbf{e}_{j,i+1}^{[k]} = \mathbf{e}_{j,i}^{[k]} + \Delta t_i \boldsymbol{\Delta}\mathbf{f}(t_{j,i}, \mathbf{e}_{j,i}^{[k]}) + \left(\mathbf{R}_j^{[k]}[:, i+1] - \mathbf{R}_j^{[k]}[:, i]\right), \tag{3.15}$$

where $\mathbf{R}[:, i]$ refers to column i of matrix $\mathbf{R}$. Similarly, the right-hand rule yields

$$\mathbf{e}_{j,i+1}^{[k]} = \mathbf{e}_{j,i}^{[k]} + \Delta t_i \boldsymbol{\Delta}\mathbf{f}(t_{j,i+1}, \mathbf{e}_{j,i+1}^{[k]}) + \left(\mathbf{R}_j^{[k]}[:, i+1] - \mathbf{R}_j^{[k]}[:, i]\right). \tag{3.16}$$

The procedure for SDC is given in (3.2). As originally formulated, SDC uses spectral nodes $\bar{\tau}_i$, $i = 1, 2, \ldots, n$, e.g., the Gaussian nodes (2.2), as the places where $\mathbf{Y}_j^{[k]}$ is computed [20]. Although in principle it is not necessary to restrict the choice of integrators to forward or backward Euler, the general non-smoothness of the error vector as measured by a discrete Sobolev norm and associated with non-uniform spectral nodes only guarantees an increase in order of one per correction level [16]; hence the use of high-order integrators is generally not deemed worthwhile. Nonetheless, using spectral nodes, SDC generally gives superior results compared to CDC and DGR, especially for large n.

Algorithm 3.2 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 (1.1) at the spectral nodes $t_{j,i} \in [t_{j,1}, t_{j,n}]$, $i = 1, 2, \ldots, n$, using forward or backward Euler.
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 (3.13).
6. Compute the error, $\mathbf{E}_j^{[k-1]} = (\mathbf{e}_{j,1}^{[k-1]}, \ldots, \mathbf{e}_{j,n}^{[k-1]})$, e.g., using (3.15) or (3.16).
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** $\mathbf{Y}_j^{[K]}(t_{j+1})$. If the spectral nodes include the right endpoint, i.e., $t_{j,n} = t_{j+1}$, then $\mathbf{Y}_j^{[K]}(t_{j+1}) = \mathbf{Y}_n^{[K]}$. If the spectral nodes do not include the right endpoint, the collocation polynomial is formed using $\mathbf{Y}_j^{[K]}$ and evaluated at t_{j+1}.

For the purpose of comparing DC methods in section 3.5, we differentiate (3.12) to form the IVP that can be used to describe the SDC algorithm. One can view the SDC algorithm as being applied to each group of nodes (3.2) within $[t_j, t_{j+1}]$ using

$$\frac{d}{dt}\mathbf{e}_j^{[k]}(t, \mathbf{Y}_j^{[k]}(t)) = \boldsymbol{\Delta}\mathbf{f}^{[k]}(t, \mathbf{Y}_j^{[k]}(t)) + \frac{d}{dt}\mathbf{r}^{[k]}(t, \mathbf{Y}_j^{[k]}(t)), \tag{3.17a}$$

$$\mathbf{e}_j^{[k]}(t_j, \mathbf{Y}_j^{[k]}(t_j)) = \mathbf{0}. \tag{3.17b}$$

3.3.1 Collocation

Approximating the Picard integral formulation (3.8) using quadrature gives rise to

$$\mathbf{Y}_j^{[k]} = \tilde{\mathbf{Y}}_j^{[k]} + \mathbf{Q}_n \mathbf{F}_j^{[k]}, \tag{3.18}$$

where $\mathbf{Q}_n$ are the quadrature weights and $\tilde{\mathbf{Y}}_j^{[k]}$ is defined in equation (3.14). Equations (3.18) can be recognized as the standard conditions for collocation on the interval $[t_j, t_{j+1}]$; see, e.g., [31]. These conditions are also identical, however, to the vanishing of the residual $\mathbf{R}_j^{[k]}(\mathbf{Y}_j^{[k]})$ from (3.13); i.e., SDC converges to a collocation solution of (1.1).

The process by which this convergence occurs can be further understood by considering the direct solution of (3.18) by quasi-linearization (Newton's method). Let $\mathbf{f}(t,\mathbf{y}) = \mathbf{A}\mathbf{y}+\tilde{\mathbf{f}}(t)$ in (1.1), where $\mathbf{A}$ is a constant matrix. Then, the SDC form of the error equation can be written as

$$(\mathbf{I} - \mathbf{Q}_n\mathbf{A})\mathbf{E}_j^{[k]} = \mathbf{R}_j^{[k]}. \tag{3.19}$$

Let $\tilde{\mathbf{Q}}_n$ be a simpler quadrature rule, which we describe shortly. $\tilde{\mathbf{Q}}_n$ is used to precondition (3.19) as

$$(\mathbf{I} - \tilde{\mathbf{Q}}_n\mathbf{A})^{-1}(\mathbf{I} - \mathbf{Q}_n\mathbf{A})\mathbf{E}_j^{[k]} = (\mathbf{I} - \tilde{\mathbf{Q}}_n\mathbf{A})^{-1}\mathbf{R}_j^{[k]}, \tag{3.20}$$

which can further be written as

$$(\mathbf{I} - \boldsymbol{\Delta}\mathbf{Q}_n)\mathbf{E}_j^{[k]} = \mathbf{E}_j^{[k-1]},$$

where $\boldsymbol{\Delta}\mathbf{Q}_n = (\mathbf{I} - \tilde{\mathbf{Q}}_n\mathbf{A})^{-1}(\mathbf{Q}_n - \tilde{\mathbf{Q}}_n)\mathbf{A}$ [36,71,57]. Because $\tilde{\mathbf{Q}}_n \approx \mathbf{Q}_n$, we expect $\|\boldsymbol{\Delta}\mathbf{Q}_n\|$ to be small, especially as $\Delta t \to 0$. This interpretation has facilitated convergence analysis and links SDC methods to other iterative solvers and multigrid methods [55,60,71]. The operator $\tilde{\mathbf{Q}}_n$ is often chosen to be the lower-triangular matrix that arises from applying a right-hand quadrature rule to advance the solution at each node as in (3.16) [71]. Further, with this interpretation of SDC as a preconditioner for the collocation problem, this allows for incomplete solve of the linear systems arising in each correction sweep, referred to in the literature as inexact spectral deferred correction [65], [72].

3.4 Integral Deferred Correction (IDC)

The main idea behind integral deferred correction (IDC) [13], [14,16] is to solve an integral form of the error equation that also incorporates the defect of the solution. Using (3.6), the derivative of the error function given by (3.3) can be expressed as

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

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

$$\frac{d}{dt}\left[\mathbf{e}^{[k]}(t,\mathbf{Y}^{[k]}(t)) + \int_a^t \boldsymbol{\delta}^{[k]}(t',\mathbf{Y}^{[k]}(t'))\,\mathrm{d}t'\right] = \mathbf{f}\big(t,\mathbf{Y}^{[k]}(t) + \mathbf{e}^{[k]}(t,\mathbf{Y}^{[k]}(t))\big) - \mathbf{f}(t,\mathbf{Y}^{[k]}(t)), \tag{3.21}$$

or after using (3.6) and (3.11),

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

The IDC algorithm is applied to each group of nodes (3.2) within $[t_j, t_{j+1}]$ using

$$\frac{d}{dt}\left[\mathbf{e}_j^{[k]}(t, \mathbf{Y}_j^{[k]}(t)) + \int_{t_j}^{t} \boldsymbol{\delta}^{[k]}(t', \mathbf{Y}^{[k]}(t'))\,\mathrm{d}t'\right] = \boldsymbol{\Delta}\mathbf{f}^{[k]}(t, \mathbf{Y}_j^{[k]}(t)), \tag{3.22a}$$

$$\mathbf{e}_j^{[k]}(t_j, \mathbf{Y}_j^{[k]}(t_j)) = \mathbf{0}. \tag{3.22b}$$

Algorithm 3.3 Integral deferred correction

1. **for** $j = 1$ to J **do**
2. 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 $t_{j,l} \in [t_{j,1}, t_{j,n}]$, $l = 1, 2, \ldots, n$, 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. Compute the defect $\boldsymbol{\delta}_j^{[k-1]}(t, \mathbf{Y}_j^{[k-1]}(t))$ using (3.6).
6. Solve the IVP (3.22) 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 $t_{j,l} \in [t_{j,1}, t_{j,n}]$ $l = 1, 2, \ldots, n$.
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** $\mathbf{Y}_j^{[K]}(t_{j+1})$. If the nodes include the right endpoint, i.e., $t_{j,n} = t_{j+1}$, then $\mathbf{Y}_j^{[K]}(t_{j+1}) = \mathbf{Y}_n^{[K]}$. If the spectral nodes do not include the right endpoint, the collocation polynomial is formed using $\mathbf{Y}_j^{[K]}$ and evaluated at t_{j+1}.

Note that discretization of (3.22) also approximates the integral by a quadrature formula. For example, if the discretization is by the forward Euler method, then we have

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

and replacing the integral with a quadrature formula

$$\begin{aligned}\mathbf{e}_{j,i+1}^{[k]} = \mathbf{e}_{j,i}^{[k]} &- (\mathbf{y}_{j,i+1}^{[k]} - \mathbf{y}_{j,i}^{[k]}) \\ &+ \Delta t\Big(\mathbf{f}(t_{j,i}, \mathbf{y}_{j,i}^{[k]} + \mathbf{e}_{j,i}^{[k]}) - \mathbf{f}(t_{j,i}, \mathbf{y}_{j,i}^{[k]})\Big) + \Delta t \sum_{i=1}^{n} S_{j,i}\mathbf{f}(t_{j,i}, \mathbf{y}_{j,i}^{[k]}),\end{aligned} \tag{3.23}$$

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 (3.23) 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 [13].

If the error equation (3.22) at correction level k is discretized using a method of order p_k, the corrected solution is of order $P_K = \sum_{j=0}^{K} p_j$ if the quadrature is sufficiently accurate and uniform nodes are used. If non-uniform nodes are used, one is not guaranteed to obtain order p_k increase at correction k, even if the quadrature rule is sufficiently accurate. Consequently, there is little advantage to using a high-order RK method to solve error equation (3.22) if non-uniform nodes are used.

3.5 Examination of DC Methods

The four classes of DC methods discussed differ based on the specific form of the error equation and how it is discretized and solved numerically. For example, (3.5) is based on the linearization of (3.4). Similarly, (3.4) can be derived by differentiating (3.12), applying the fundamental theorem of calculus, and simplifying using (3.9), (3.11). Finally, (3.4) can be recovered by distributing the time derivative in (3.22), applying the fundamental theorem of calculus, and simplifying using (3.6).

Unlike the CDC and DGR methods, SDC and IDC methods discretize an integral form of the error equation. It is posited that discretizing an integral form of the error equation provides improved stability of the numerical method [20]. In terms of discretization, the main difference between SDC and IDC methods is the ability of the latter to use any single-step integrator to solve the error equation. If one discretizes (3.22) using explicit or implicit Euler integrators, one recovers (3.15) and (3.16), respectively. Another difference is that the SDC (Picard) formulation of the error equation (3.12) provides insight into how DC methods can be interpreted as an iterative approach to solve the collocation equations.

Section 4 illustrates, through numerical experiments, the formal orders of accuracy of the CDC, SDC, and IDC methods. We show that the choice of quadrature nodes affects the formal order of accuracy of the three classes of DC methods at a given stage k. We also show in sections 3.6 and 4.2 below that DC methods can be modified to allow for task-level parallelism; the parallel efficiency and the memory footprint used by the DC method also depend on the choice of quadrature nodes and the properties of the one-step integrator used.

So, how does one pick which DC method to use, how does one select the set of quadrature nodes, and in the case of IDC methods, how does one select which one-step integrator to use? Generally speaking, SDC and IDC methods have improved stability relative to CDC and DGR methods stemming from the discretization of an integral form of the error equation. If a DC method is desired and a forward or backward Euler integrator can be used with tolerable expense, then SDC methods with Gauss–Lobatto nodes (or Radau nodes for stiff problems) are likely to be a good choice. However, if the stepsize constraint imposed by a forward Euler integrator is too restrictive or an implicit integrator is undesirable, then IDC methods constructed with a more appropriate one-step integrator, e.g., having higher order or enhanced stability [70], may prove fruitful. The choice

of quadrature nodes depends largely on how much computational and memory resources are allocated or what solution accuracy is ultimately desired.

3.6 Parallel Deferred Correction

A common criticism of deferred correction methods is the computational cost (i.e., the number of right-hand side function evaluations) to obtain a comparable accuracy to a high-order RK or multi-step method. Although this criticism is generally justified, a key benefit of the deferred correction framework is the possibility for task-level parallelism, where the solution at multiple nodes (varying correction levels) can be computed simultaneously. This was recently observed, implemented, and discussed for SDC methods in [60] and for IDC methods in [16], but the idea can be broadly applied to the classes of deferred correction equations previously discussed. It was shown in [16] that the parallel IDC methods required fewer non-concurrent function evaluations to achieve similar accuracy to a comparable high-order RK method.

We recall that the time domain is divided into J intervals, (3.1), and each interval is further subdivided using n nodes, (3.2). To describe the parallel 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 with a global index,

$$t_{(m')} = t_{j,i}, \quad m' = 0, 1, \ldots, M, \tag{3.24}$$

where $t_{(0)} = a$, $j = m' \div (n-1)$, $i = (m' \mod (n-1)) + 1$, and $M = J(n-1)$. Instead of iterating the CDC, DGR, SDC, or IDC algorithm completely on each group of nodes within an interval, one first observes that (3.4), (3.5), (3.17), and (3.21) 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 3.1, 3.2, and 3.3. For example, a modified classical deferred correction algorithm is given in Algorithm 3.4.

Algorithm 3.4 Modified classical deferred correction

1. Compute an initial approximation $\mathbf{Y}^{[0]} = (\mathbf{y}^{[0]}_{(0)}, \mathbf{y}^{[0]}_{(1)}, \ldots, \mathbf{y}^{[0]}_{(M)})$ to IVP (1.1) 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 (3.5) using a method of order p_k to compute an approximation to the error $\mathbf{E}^{[k-1]} = (\mathbf{e}^{[k-1]}_{(0)}, \mathbf{e}^{[k-1]}_{(1)}, \ldots, \mathbf{e}^{[k-1]}_{(M)})$.
5. Form the new approximate solution $\mathbf{Y}^{[k]} = \mathbf{Y}^{[k-1]} + \mathbf{E}^{[k-1]}$.
6. **end for**
7. **return** $\mathbf{y}^{[K]}_{(M)}$

The modified algorithm (where the DC method runs to completion on a given level k) is sometimes called the global version, whereas in the other case it is called local version [25]. Both versions achieve the same order of convergence.

It has been observed numerically that the global versions of IDC methods are stable provided the predictor is stable [16]. Although the global CDC, DGR, SDC, and IDC methods can be shown to have the same asymptotic convergence behavior as their respective local counterparts, the final solution generated by the global versions of the method is generally less accurate because the correction equations are not iterated to completion (level K) on each group of nodes; i.e., in the global method, the integrator at level k, $k < K$, proceeds using information only from levels $k' < k$, whereas the local method generally uses information from level K.

Although Algorithm 3.4 is not explicitly given in parallel form, it provides the potential for *pipeline parallelism*, where multiple correction levels can be simultaneously computed [21]. 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 Fig. 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 Fig. 2, reducing the memory footprint and the lag interval of a pipeline parallel implementation [16].

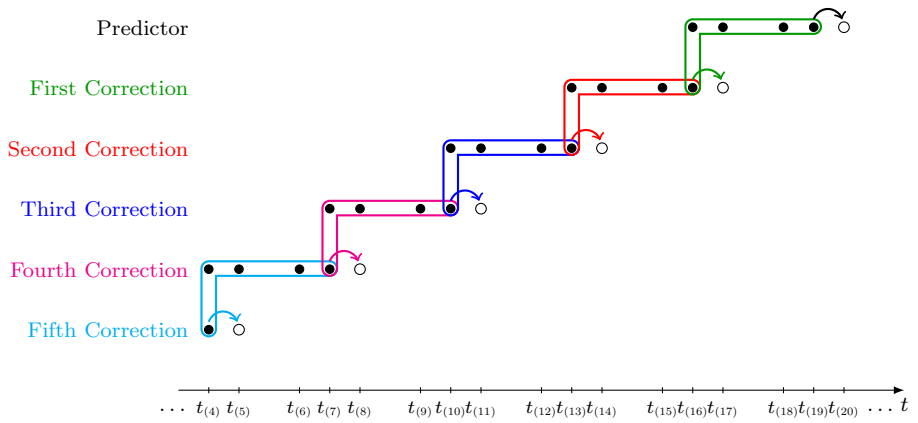


Fig. 1 The deferred correction framework allows for task-level parallelism, where multiple levels of correction can be computed simultaneously. While the first-order predictor is updating a provisional solution from t_{j-1} to t_j, corrector k is able to compute an approximation to the error at time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

3.7 Analysis of Parallel Speedup and Efficiency

We present an analysis of the efficiency of a parallel implementation of DC using forward Euler integrators. In the notation introduced previously, we have J groups of n nodes (3.2), where for simplicity we assume each group contains both endpoints (and hence $n \geq 2$). We assume that function evaluations dominate the

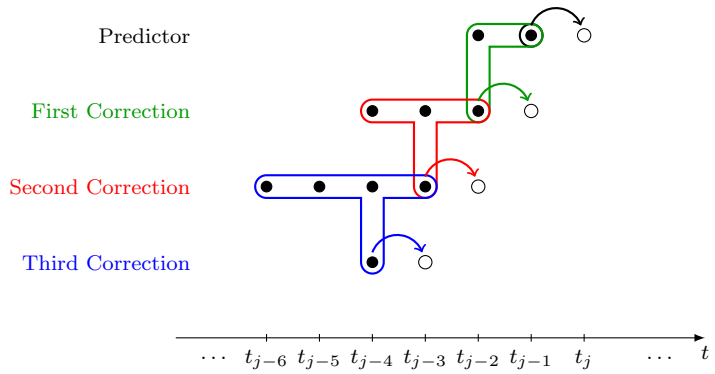


Fig. 2 The deferred correction framework allows for task-level parallelism, where multiple levels of correction can be computed simultaneously. While the first-order predictor is updating a provisional solution from t_{j-1} to t_j, corrector k is able to compute an approximation to the error at time 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.

computational time and that all function evaluations require the same amount of computation time; i.e., memory access and communication overhead are negligible compared to the computation time required for function evaluations. Thus, (non-concurrent) function evaluations can be used as a proxy for computational time. Although quite popular and valuable in practice, work-precision diagrams of error versus computational effort are decidedly difficult to reliably obtain in a parallel setting. Considerations such as memory footprint and communication costs render such results strongly system dependent and thus beyond the scope of this review.

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

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

where K_S is the number of corrections in serial mode. This is also the time required by 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_P corrections, this allows the utilization of K_P+1 processors for computation, provided there are sufficiently many groups (ideally $J \gg K_P$). We count only the number of non-concurrent function evaluations in our evaluation of the parallel implementation. For K_P corrections, the number of non-concurrent function evaluations in a parallel implementation is

$$\text{feval}_{\text{non-concurrent}} = (n-1)(J+K_P).$$

The *speedup* of a parallel method is generally given by

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

where T_i is the execution time for a computation performed with i processors and n_P is the number of 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 perfect parallel efficiency $E_{n_P} = S_{n_P}/n_P = 1$. The speedup for the parallel DC as described above with $K_P + 1$ processors is

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

In the case when $J = 1$, we have $K_S = K_P := K$, and hence $S_{K+1} = 1$, independent of K; i.e., there is no opportunity for parallelization for the case of only one group, and the computation is necessarily serial. It is possible that more RIDC iterations are required to converge to a similar level of error when J is large. In the case when $J \gg K_P$, we obtain $S_{K_P+1} \approx K_S + 1$, showing that the speedup improves as K_S increases. The parallel efficiency is

$$E_{K_P+1} = \frac{S_{K_P+1}}{K_P+1} = \frac{J(1+K_S)}{(K_P+1)(J+K_P)},$$

which approaches $(1 + K_S)/(1 + K_P)$ for $J \gg K_P$.

This result can be interpreted as follows. If $n_P = K_P + 1$ processors are used to implement a parallel DC method with K_P corrections and $J \gg K_P$, a high-order solution can be attained in virtually the same wall-clock time as a low-order provision solution computed with one processor.

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 $\mathbb{R}^1$

$$\begin{aligned} u_t &= A + u^2 v - (B+1)u + \alpha u_{xx}, \\ v_t &= Bu - u^2 v + \alpha v_{xx}, \end{aligned} \tag{4.1}$$

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 [32], 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,

$$\begin{aligned} &\ddot{y}_1 = y_1 + 2\dot{y}_2 - \mu' \frac{y_1 + \mu}{D_1} - \mu \frac{y_1 - \mu'}{D_2}, \\ &\ddot{y}_2 = y_2 - 2\dot{y}_1 - \mu' \frac{y_2}{D_1} - \mu \frac{y_2}{D_2}, \\ &D_1 = \left((y_1 + \mu)^2 + y_2^2\right)^{3/2}, \quad D_2 = \left((y_1 - \mu')^2 + y_2^2\right)^{3/2}, \\ &\mu = 0.012277471, \quad \mu' = 1 - \mu. \end{aligned} \tag{4.2}$$

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.
More specialized DC algorithms exist for these problems than those described in this paper; e.g., the Brusselator equation could be treated with a semi-implicit method [46,18] and the Arenstorf orbit problem could be treated directly as a second-order problem. The use of such specialized algorithms would undoubtedly affect performance in practice, but their implementation and assessment are beyond the scope of this review.

4.1 Order of Accuracy and Efficiency

We use test problem 1 to illustrate the expected order of accuracy that can be expected for deferred correction methods as well as to compare the computational effort for various methods. Suppose that we have J groups of n nodes, as described in (3.2). If these nodes are uniformly distributed, we recall (3.7) that DGR methods can attain order

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

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 $P_K = \sum_{k=0}^{K} p_k$. Suppose forward Euler integrators are used to generate the provisional solution and then used to solve the error equation (3.5). The standard convergence plot, error versus the number of intervals, is shown in Fig. 3; the methods achieve the designed orders of accuracy. To compare the efficiency of these integrators, we need to account for the varying amount of work performed by each integrator. One approach is to use

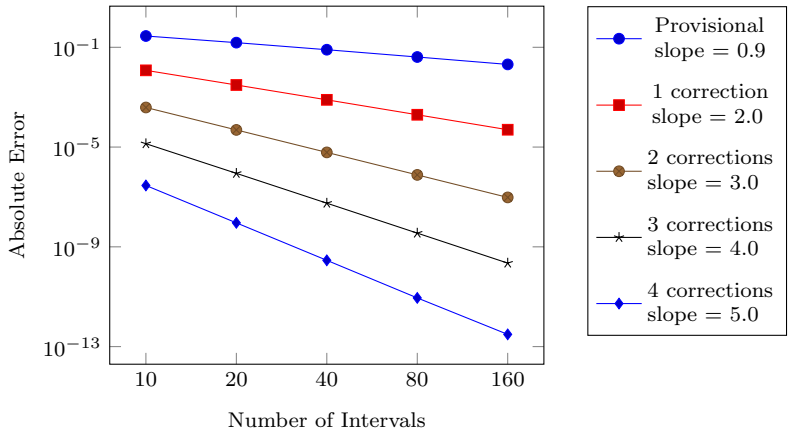


Fig. 3 Absolute error as a function of the number of intervals for various classical deferred correction methods. Forward Euler integrators are used to solve test problem 1 and the error equation (3.5). The method with K corrections uses $(K + 2)$ uniformly spaced quadrature nodes per interval. The expected orders of accuracy are observed.

the number of function evaluations as a proxy for the amount of work required by the integrator. Fig. 4 shows the error as a function of the number of function evaluations at a given level. The general expectation is that higher-order methods are more efficient than lower-order methods when solutions are sufficiently smooth and desired errors are small enough.

If one extrapolates the convergence curves in Fig. 4 for a small number of function evaluations, the convergence curves cross, showing that the high-order DGR methods have a larger error coefficient for the same amount of work. For a small enough time step (i.e., for a sufficiently large number of intervals / function evaluations), the rapid decrease of the error due to the high-order integrator results in better efficiency (smaller error for the same amount of work).

Higher-order integrators can be used to generate the provisional solution and to solve the error equation (3.5). In Fig. 5, we show the order of convergence and the relative work required when Kutta's three-stage, third-order explicit Runge–Kutta (RK3) integrator is used to generate the provisional solution and the two-stage, second-order explicit mid-point Runge–Kutta (RK2) integrator is used to solve the error equation (3.5). 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. We observe the expected orders of accuracy at each level. We also note that the specific choice of integrators has led to a significantly higher efficiency achieved for this problem for a given order.

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

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

We note that the attainable order is now n instead of $n - 1$ because neither SDC nor IDC differentiates the interpolant. Fig. 6 shows that the expected orders of

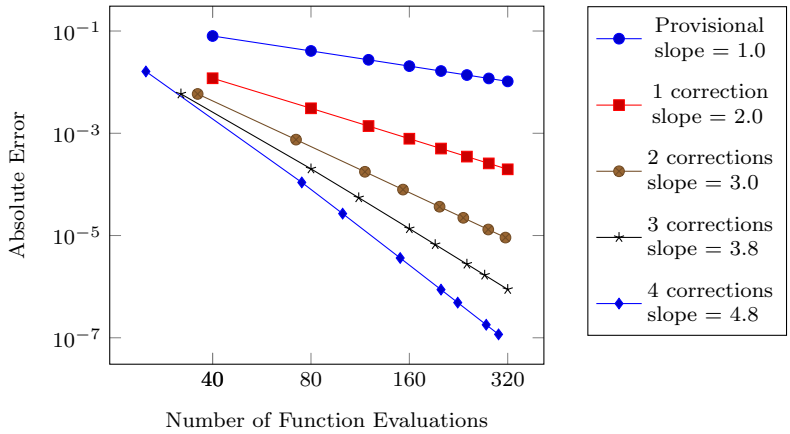


Fig. 4 Absolute error as a function of the number of function evaluations for various classical deferred correction methods. Forward Euler integrators are used to solve test problem 1 and the error equation (3.5). The method with K corrections uses $(K+2)$ uniformly spaced quadrature nodes per interval. The expected orders of accuracy are observed.

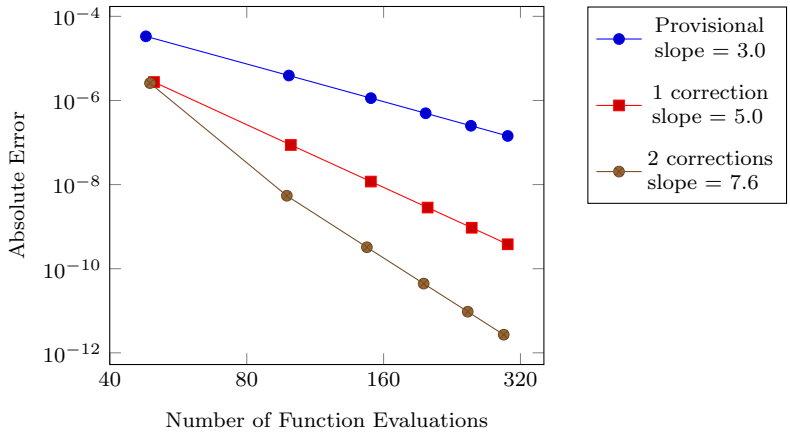


Fig. 5 Absolute error as a function of the number of function evaluations for various classical deferred correction methods. Kutta's RK3 integrator is used to generate the provisional solution for test problem 1; the explicit midpoint RK2 integrator is used to solve the error equation (3.5). The method 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.

accuracy are observed when IDC is used with forward Euler integrators and the method with K corrections uses $(K+1)$ uniformly spaced quadrature nodes.

Fig. 7 shows that the expected orders of accuracy are observed with IDC when the explicit midpoint RK2 integrator is used to generate the provisional solution and solve the error equation (3.22), and the method with one and two corrections uses five and seven uniformly spaced quadrature nodes in each interval, respectively. The efficiency of these two methods is comparable to the previous one.

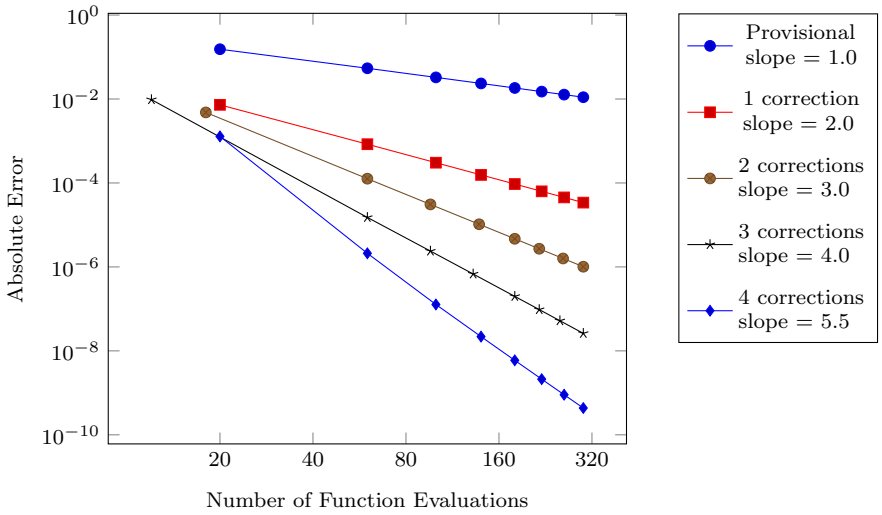


Fig. 6 Absolute error as a function of the number of function evaluations for various integral deferred correction methods. Forward Euler integrators are used to solve test problem 1 and the error equation (3.22). The method with K corrections uses $(K+1)$ uniformly spaced quadrature nodes per interval. The expected orders of accuracy are observed.

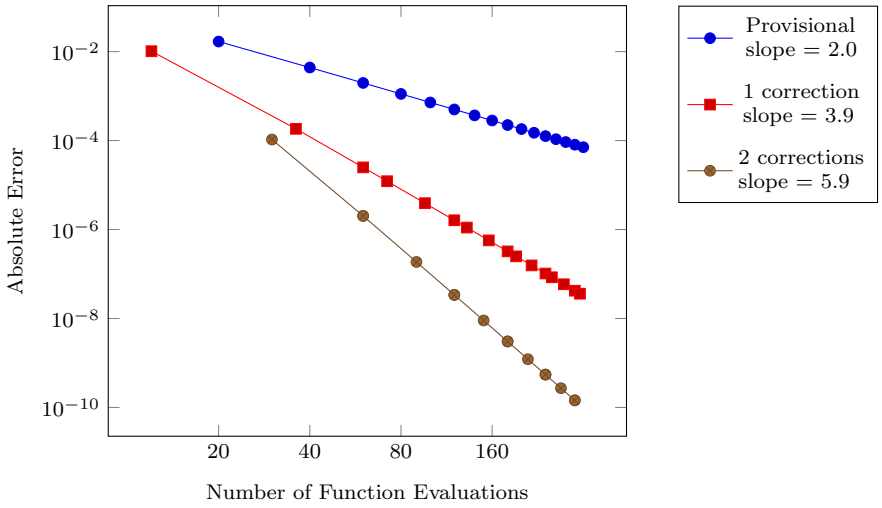


Fig. 7 Absolute error as a function of the number of function evaluations for various integral deferred correction methods. The explicit midpoint RK2 integrator is used to solve test problem 1 and the error equation (3.22). The method with one correction uses four uniformly spaced quadrature nodes in each interval. The method with two corrections uses six uniformly spaced quadrature nodes in each interval. The expected orders of accuracy are observed.

The rates of convergence are more subtle if non-uniform quadrature nodes are used. We consider DGR methods constructed with J groups of n nodes (3.2), where each group of n nodes is made up of $(K+2)$ Gauss–Lobatto nodes for a method with K corrections, and forward Euler integrators are used to solve test problem 1 and the associated error equation (3.5). Fig. 8 shows that for a method with one correction, the accuracy and order of convergence improve, but for methods with more corrections, the accuracy improves while the order of accuracy remains stagnant at one. This has previously been observed in [34] and is explained by the discrete smoothness of the underlying quadrature mesh [58]. Specifically, *the lack of discrete smoothness* of the mesh function of the approximate solution precludes an increase in the order of accuracy for DGR methods constructed using general non-uniform nodes, including Gauss–Lobatto, Gauss–Legendre, and Chebyshev nodes. For this example, the method that uses only one correction is an exception because three Gauss–Lobatto nodes are in fact uniformly spaced; thus, the limit on order increase imposed by a lack of discrete smoothness of the underlying mesh function does not apply, and the method with one correction is able to attain second order. In general, however, methods such as SDC or IDC that use spectral nodes at which to evaluate the defect do not suffer from this order barrier [58].

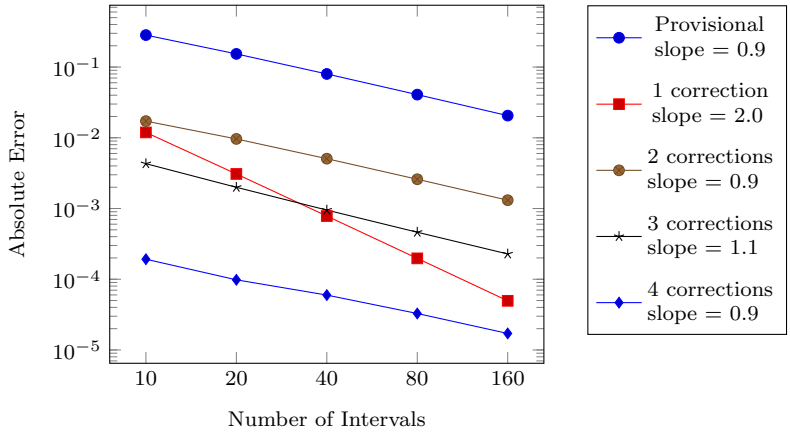


Fig. 8 Absolute error as a function of the number of intervals for various classical deferred correction methods. Forward Euler integrators are used to solve test problem 1 and the error equation (3.5). The method with K corrections uses $(K+2)$ Gauss–Lobatto nodes per interval. The order of accuracy for all but one methods is one. The exception is for the method with one correction. It attains second order because it uses three Gauss–Lobatto nodes, which are in fact uniformly spaced and hence allow for an additional order of accuracy.

A similar behavior is observed if the RK2 method is used to solve test problem 1 and the associated error equation (3.5). Fig. 9 shows that the accuracy improves for methods that utilize more corrections; the order of accuracy, however, remains stagnant at two. The method with one correction uses five Gauss–Lobatto quadrature nodes in each interval. The method with two corrections uses seven Gauss–Lobatto quadrature nodes in each interval.

For SDC/IDC methods, the discrete smoothness of non-uniform nodes guarantees only one order of accuracy increase with each correction [2], [14]. The subse-

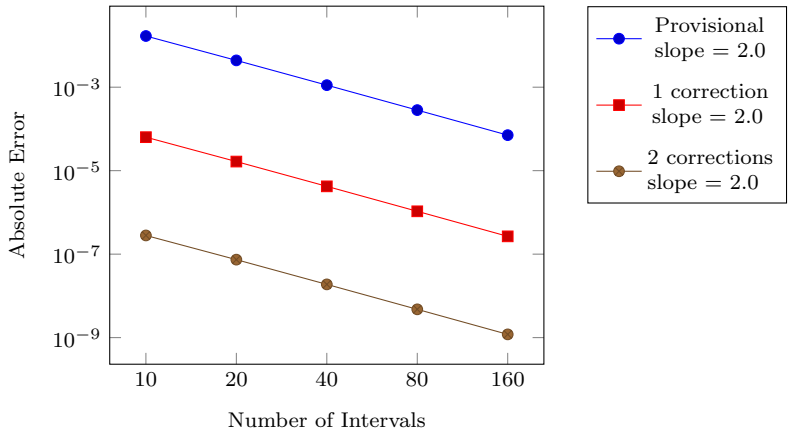


Fig. 9 Absolute error as a function of the number of intervals for various classical deferred correction methods. The explicit midpoint method is used to solve test problem 1 and the associated error equation (3.5). The method with one correction uses five Gauss–Lobatto quadrature nodes in each interval. The method with two corrections uses seven Gauss–Lobatto quadrature nodes in each interval. The order of accuracy stagnates at two, the order of the underlying integrator.

quent maximum order of accuracy that can be attained depends on the accuracy of the quadrature formula. The order of n-node Gauss–Legendre quadrature is $2n$; the order of n-node Gauss–Lobatto quadrature is $(2n-2)$; the order of n-node Chebyshev quadrature is $(n+1)$ if n is odd and $(n+2)$ if n is even.

We begin with SDC methods constructed using Gauss–Lobatto quadrature nodes and forward Euler integrators applied to test problem 1. The method with one correction uses two Gauss–Lobatto nodes in each interval. The methods with two and three corrections use three Gauss–Lobatto nodes in each interval. The methods with four and five corrections use five Gauss–Lobatto nodes. Fig. 10 shows that methods with one additional correction add one order of accuracy, as expected.

Lastly, we construct SDC methods constructed using Gauss–Legendre nodes and forward Euler integrators applied to test problem 1. Using n quadrature nodes, we construct a method with $2n-1$ corrections to show that the order-$2n$ collocation solution can be obtained. The efficiency for each method is shown in Fig. 11. For these last two cases, we again observe comparable efficiencies with the second case for a given order and number of function evaluations.

The behaviors described in this section can be observed for different test problems or other integrators within the DC methods, some of which are described subsequently. The reader is able to numerically explore properties of deferred correction by downloading the MATLAB source code used to generate results in this section.[1]

[1] The software used to generate numerical results in this manuscript is hosted temporarily 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.

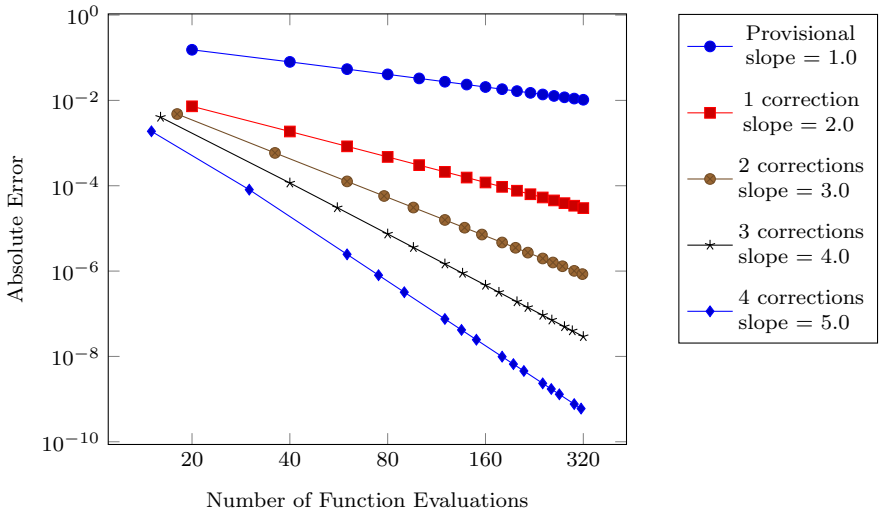


Fig. 10 Absolute error as a function of the number of function evaluations for various spectral deferred correction methods. Forward Euler integrators are used to solve test problem 1 and the error equation (3.17). The method with one correction uses two Gauss–Lobatto quadrature nodes in each interval, the methods with two and three corrections use three Gauss–Lobatto quadrature nodes in each interval, and the method with four corrections uses four Gauss–Lobatto quadrature nodes. The expected orders of accuracy are observed.

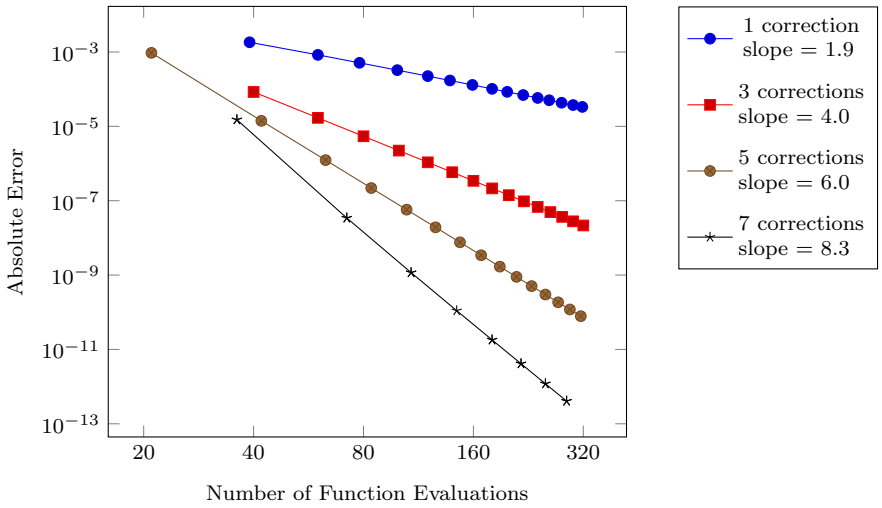


Fig. 11 Absolute error as a function of the number of function evaluations for various spectral deferred correction methods. Forward Euler integrators are used to solve test problem 1 and the error equation (3.17). The method with $2n-1$ corrections uses n Gauss–Legendre quadrature nodes. The expected orders of accuracy of the collocation solutions are achieved.

4.2 Pipeline Parallelism

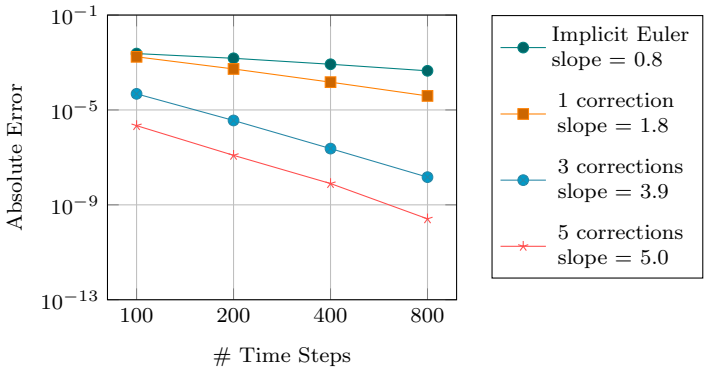
To observe parallel speedup in DC methods, the computational overhead of solving the error equations (i.e., computing the quadrature approximation or interpolant in the respective error equations) must be inexpensive compared to the cost of advancing the provisional solution from time t to $t + \Delta t$. Parallel speedup can be expected when evaluation of the ODE right-hand side is expensive, as in the case of N-body problems [16], or when solutions of nonlinear systems 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 system (4.1) in $\mathbb{R}^1$ and the IDC correction equation (3.22). The nonlinear system of equations is solved using a Newton iteration. The RIDC software [51] and the GNU Scientific Library (GSL) are used to generate the timing results. The scaling studies were performed on a single computational node consisting of a dual socket Intel E5-2680v4 chipset. The domain of integration is $[0, 10]$, and the spatial domain $[0, 1]$ is divided into 400 intervals. Fig. 12(a) shows a standard convergence study of RIDC. The order increases as more correctors are used. We note that the RIDC method with five corrections exhibits order reduction, achieving only fifth order (instead of the anticipated sixth order). Order reduction has been previously observed when high-order methods are applied to solve similar non-linear oscillatory problems [30], and we posit that a similar behavior is being observed when high-order RIDC is applied to solve the Brusselator system. Fig. 12(b) shows the results from the same numerical experiment but with error plotted against walltime. Here, $(K+1)$ processors are used for RIDC integrators with K correctors. Data markers that line up vertically indicate that the RIDC method scales perfectly. The offset in data points can be interpreted as the overhead of the RIDC method: the startup and shutdown phases in which the pipeline is not full, the communication/memory access overhead, and the extra flops needed to compute the quadrature and interpolant. In this example, RIDC methods achieve over 90% efficiency when 800 time steps are computed.

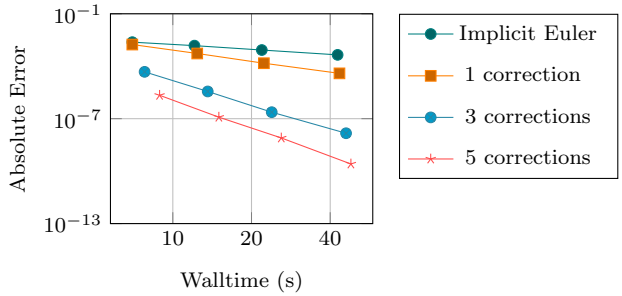
4.3 Choice of Interpolant

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. There is nothing sacrosanct about such a choice, however, and other interpolants, such as continuous extensions to numerical methods or cubic or Hermite splines, are valid as well.

In a final experiment, we first solve the Arenstorf orbit problem using classical deferred correction with forward Euler predictor and correctors, where uniformly distributed nodes are chosen, and a cubic spline interpolant is constructed, evaluated, and differentiated when solving the error equation (3.5). Fig. 13 shows that, although 10^5 time steps are used for the predictor, the orbit is not closed. The DGR corrector (utilizing spline interpolants) is able to correct the orbit to obtain a much closer approximation to the expected periodic orbit.

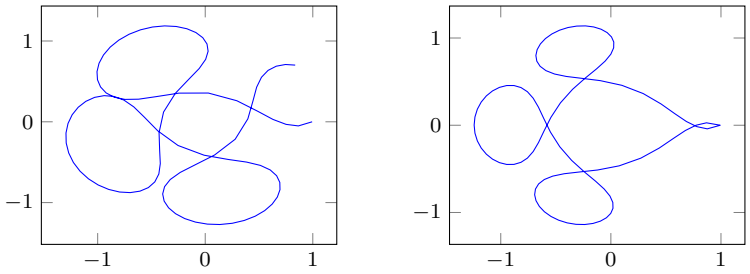


(a) Standard convergence study: Absolute error versus number of time steps



(b) Parallel convergence study: Absolute error versus walltime

Fig. 12 RIDC integrators (constructed using backward Euler integrators) used to solve the Brusselator system (4.1) in $\mathbb{R}^1$.



(a) Predictor (b) 1 Correction

Fig. 13 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 integrator and cubic splines to solve the error equation (3.5).

Of course, many questions on the use of spline interpolants in the context of deferred correction remain open. These include questions about the effects on stability, accuracy, and the conditions under which the use of spline interpolation is advantageous. A more comprehensive study of how splines perform relative to the Lagrange approach would likely be a welcome addition to the deferred correction literature.

5 Conclusion

In this paper, a survey is given of four popular classes of deferred correction methods for solving initial-value problems: the classical deferred correction methods of Zadunaisky / Stetter and Dutt, Greengard, and Rokhlin, spectral deferred correction, and integral deferred correction. A unified notation is used to review the theoretical underpinnings, including the choice of quadrature nodes, interpolants, and combinations of discretization methods, 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 on a diverse set of examples illustrate the order of accuracy and effectiveness of deferred correction methods in various situations. Lightweight and flexible Matlab software is available for the reader interested in experimenting with these aspects or solving other problems.

References

1. Aggul, M., Labovsky, A.: A high accuracy minimally invasive regularization technique for Navier-–Stokes equations at high Reynolds number. Numerical Methods for Partial Differential Equations **33**(3), 814–839 (2017). DOI 10.1002/num.22124. URL `http://dx.doi.org/10.1002/num.22124`
2. Auzinger, W., Hofstätter, H., Kreuzer, W., Weinmüller, E.: Modified defect correction algorithms for ODEs. I. General theory. Numer. Algorithms **36**(2), 135–155 (2004). DOI 10.1023/B:NUMA.0000033129.73715.7f. URL `http://dx.doi.org/10.1023/B:NUMA.0000033129.73715.7f`
3. Auzinger, W., Hofstätter, H., Kreuzer, W., Weinmüller, E.: Modified defect correction algorithms for ODEs. II. Stiff initial value problems. Numer. Algorithms **40**(3), 285–303 (2005). DOI 10.1007/s11075-005-5327-4. URL `http://dx.doi.org/10.1007/s11075-005-5327-4`
4. Bolten, M., Moser, D., Speck, R.: A multigrid perspective on the parallel full approximation scheme in space and time. Numer. Linear Algebra Appl. **24**(6), e2110, 24 (2017). DOI 10.1002/nla.2110. URL `https://doi-org.cyber.usask.ca/10.1002/nla.2110`
5. Bolten, M., Moser, D., Speck, R.: Asymptotic convergence of the parallel full approximation scheme in space and time for linear problems. Numer. Linear Algebra Appl. **25**(6), e2208, 22 (2018). DOI 10.1002/nla.2208. URL `https://doi-org.cyber.usask.ca/10.1002/nla.2208`
6. Boscarino, S., Qiu, J.M.: Error estimates of the integral deferred correction method for stiff problems. ESAIM Math. Model. Numer. Anal. **50**(4), 1137–1166 (2016). DOI 10.1051/m2an/2015072. URL `http://dx.doi.org/10.1051/m2an/2015072`
7. Boscarino, S., Qiu, J.M., Russo, G.: Implicit-explicit integral deferred correction methods for stiff problems. SIAM J. Sci. Comput. **40**(2), A787–A816 (2018). DOI 10.1137/16M1105232. URL `https://doi-org.cyber.usask.ca/10.1137/16M1105232`
8. Bottasso, C.L., Ragazzi, A.: Deferred-correction optimal control with applications to inverse problems in flight mechanics. Journal of Guidance, Control, and Dynamics **24**(1), 101–108 (2001). DOI 10.2514/2.4681. URL `https://doi.org/10.2514/2.4681`

9. Bourlioux, A., Layton, A.T., Minion, M.L.: High-order multi-implicit spectral deferred correction methods for problems of reactive flow. J. Comput. Phys. **189**(2), 651–675 (2003). DOI 10.1016/S0021-9991(03)00251-1. URL http://dx.doi.org/10.1016/S0021-9991(03)00251-1
10. Briggs, W.L., Henson, V.E., McCormick, S.F.: A multigrid tutorial, second edn. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA (2000). DOI 10.1137/1.9780898719505. URL https://doi.org/10.1137/1.9780898719505
11. Bu, S., Huang, J., Minion, M.L.: Semi-implicit Krylov deferred correction methods for differential algebraic equations. Math. Comp. **81**(280), 2127–2157 (2012). DOI 10.1090/S0025-5718-2012-02564-6. URL http://dx.doi.org/10.1090/S0025-5718-2012-02564-6
12. Causley, M.F., Seal, D.C.: On the convergence of spectral deferred correction methods. Commun. Appl. Math. Comput. Sci. **14**(1), 33–64 (2019). DOI 10.2140/camcos.2019.14.33. URL https://doi.org/10.2140/camcos.2019.14.33
13. Christlieb, A., Ong, B., Qiu, J.M.: Comments on high-order integrators embedded within integral deferred correction methods. Commun. Appl. Math. Comput. Sci. **4**, 27–56 (2009). DOI 10.2140/camcos.2009.4.27. URL http://dx.doi.org/10.2140/camcos.2009.4.27
14. Christlieb, A., Ong, B., Qiu, J.M.: Integral deferred correction methods constructed with high order Runge–Kutta integrators. Math. Comp. **79**(270), 761–783 (2010). DOI 10.1090/S0025-5718-09-02276-5. URL http://dx.doi.org/10.1090/S0025-5718-09-02276-5
15. Christlieb, A.J., Guo, W., Morton, M., Qiu, J.M.: A high order time splitting method based on integral deferred correction for semi-Lagrangian Vlasov simulations. J. Comput. Phys. **267**, 7–27 (2014). DOI 10.1016/j.jcp.2014.02.012. URL http://dx.doi.org/10.1016/j.jcp.2014.02.012
16. Christlieb, A.J., Macdonald, C.B., Ong, B.W.: Parallel high-order integrators. SIAM J. Sci. Comput. **32**(2), 818–835 (2010). DOI 10.1137/09075740X. URL http://dx.doi.org/10.1137/09075740X
17. Christlieb, A.J., Macdonald, C.B., Ong, B.W., Spiteri, R.J.: Revisionist integral deferred correction with adaptive step-size control. Commun. Appl. Math. Comput. Sci. **10**(1), 1–25 (2015). DOI 10.2140/camcos.2015.10.1. URL http://dx.doi.org/10.2140/camcos.2015.10.1
18. Christlieb, A.J., Morton, M., Ong, B., Qiu, J.M.: Semi-implicit integral deferred correction constructed with additive Runge–Kutta methods. Commun. Math. Sci. **9**(3), 879–902 (2011). DOI 10.4310/CMS.2011.v9.n3.a10. URL http://dx.doi.org/10.4310/CMS.2011.v9.n3.a10
19. Chu, K.W., Spence, A.: Deferred correction for the integral equation eigenvalue problem. J. Austral. Math. Soc. Ser. B **22**(4), 474–487 (1980/81). DOI 10.1017/S0334270000002812. URL http://dx.doi.org/10.1017/S0334270000002812
20. Dutt, A., Greengard, L., Rokhlin, V.: Spectral deferred correction methods for ordinary differential equations. BIT **40**(2), 241–266 (2000). DOI 10.1023/A:1022338906936. URL http://dx.doi.org/10.1023/A:1022338906936
21. Eijkhout, V.: Introduction to High Performance Scientific Computing. Lulu.com (2012)
22. Emmett, M., Minion, M.: Toward an efficient parallel in time method for partial differential equations. Commun. Appl. Math. Comput. Sci. **7**(1), 105–132 (2012). DOI 10.2140/camcos.2012.7.105. URL https://doi.org/10.2140/camcos.2012.7.105
23. Fox, L.: Some improvements in the use of relaxation methods for the solution of ordinary and partial differential equations. Proc. Roy. Soc. London. Ser. A. **190**, 31–59 (1947)
24. Fox, L., Goodwin, E.T.: Some new methods for the numerical integration of ordinary differential equations. Proc. Cambridge Philos. Soc. **45**, 373–388 (1949)
25. Frank, R., Ueberhuber, C.W.: Iterated defect correction for the efficient solution of stiff systems of ordinary differential equations. BIT Numerical Mathematics **17**(2), 146–159 (1977). DOI 10.1007/BF01932286. URL https://doi.org/10.1007/BF01932286
26. Götschel, S., Minion, M.L.: Parallel-in-time for parabolic optimal control problems using PFASST. In: Domain decomposition methods in science and engineering XXIV, *Lect. Notes Comput. Sci. Eng.*, vol. 125, pp. 363–371. Springer, Cham (2018)
27. Grout, R., Kolla, H., Minion, M., Bell, J.: Achieving algorithmic resilience for temporal integration through spectral deferred corrections. Commun. Appl. Math. Comput. Sci. **12**(1), 25–50 (2017). DOI 10.2140/camcos.2017.12.25. URL https://doi.org/10.2140/camcos.2017.12.25
28. Güttel, S., Klein, G.: Efficient high-order rational integration and deferred correction with equispaced data. Electron. Trans. Numer. Anal. **41**, 443–464 (2014)

29. Hackbusch, W.: Multigrid methods and applications, *Springer Series in Computational Mathematics*, vol. 4. Springer-Verlag, Berlin (1985). DOI 10.1007/978-3-662-02427-0. URL https://doi.org/10.1007/978-3-662-02427-0
30. Hagstrom, T., Zhou, R.: On the spectral deferred correction of splitting methods for initial value problems. Commun. Appl. Math. Comput. Sci. **1**, 169–205 (2006). DOI 10.2140/camcos.2006.1.169. URL http://dx.doi.org/10.2140/camcos.2006.1.169
31. Hairer, E., Nørsett, S.P., Wanner, G.: Solving ordinary differential equations. I, *Springer Series in Computational Mathematics*, vol. 8, second edn. Springer-Verlag, Berlin (1993). Nonstiff problems
32. Hairer, E., Wanner, G.: Solving ordinary differential equations. II, *Springer Series in Computational Mathematics*, vol. 14, second edn. Springer-Verlag, Berlin (1996). DOI 10.1007/978-3-642-05221-7. URL http://dx.doi.org/10.1007/978-3-642-05221-7. Stiff and differential-algebraic problems
33. Hamon, F.P., Schreiber, M., Minion, M.L.: Multi-level spectral deferred corrections scheme for the shallow water equations on the rotating sphere. Journal of Computational Physics **376**, 435 – 454 (2019). DOI https://doi.org/10.1016/j.jcp.2018.09.042. URL http://www.sciencedirect.com/science/article/pii/S0021999118306442
34. Hansen, A.C., Strain, J.: On the order of deferred correction. Appl. Numer. Math. **61**(8), 961–973 (2011). DOI 10.1016/j.apnum.2011.04.001. URL http://dx.doi.org/10.1016/j.apnum.2011.04.001
35. Hofstätter, H., Koch, O.: Analysis of a defect correction method for geometric integrators. Numerical Algorithms **41**(2), 103–126 (2006). DOI 10.1007/s11075-005-9001-7. URL https://doi.org/10.1007/s11075-005-9001-7
36. Huang, J., Jia, J., Minion, M.: Accelerating the convergence of spectral deferred correction methods. J. Comput. Phys. **214**(2), 633–656 (2006). DOI 10.1016/j.jcp.2005.10.004. URL http://dx.doi.org/10.1016/j.jcp.2005.10.004
37. Kress, W., Gustafsson, B.: 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, pp. 241–251 (2002). DOI 10.1023/A:1015113017248. URL http://dx.doi.org/10.1023/A:1015113017248
38. Layton, A.T., Minion, M.L.: Conservative multi-implicit spectral deferred correction methods for reacting gas dynamics. J. Comput. Phys. **194**(2), 697–715 (2004). DOI 10.1016/j.jcp.2003.09.010. URL http://dx.doi.org/10.1016/j.jcp.2003.09.010
39. Layton, A.T., Minion, M.L.: Implications of the choice of quadrature nodes for Picard integral deferred corrections methods for ordinary differential equations. BIT **45**(2), 341–373 (2005). DOI 10.1007/s10543-005-0016-1. URL http://dx.doi.org/10.1007/s10543-005-0016-1
40. Layton, A.T., Minion, M.L.: Implications of the choice of predictors for semi-implicit Picard integral deferred correction methods. Commun. Appl. Math. Comput. Sci. **2**, 1–34 (2007). DOI 10.2140/camcos.2007.2.1. URL http://dx.doi.org/10.2140/camcos.2007.2.1
41. Lindberg, B.: Error estimation and iterative improvement for the numerical solution of operator equations. Tech. rep., Illinois University at Urbana–Champaign Department of Computer Science (1976)
42. Lions, J., Maday, Y., Turinici, G.: A "parareal" in time discretization of PDEs. Comptes Rendus de l'Academie des Sciences Series I Mathematics **332**(7), 661–668 (2001)
43. Lions, J.L., Maday, Y., Turinici, G.: Résolution d'EDP par un schéma en temps "pararéel". C. R. Acad. Sci. Paris Sér. I Math. **332**(7), 661–668 (2001). DOI 10.1016/S0764-4442(00)01793-6. URL http://dx.doi.org/10.1016/S0764-4442(00)01793-6
44. Liu, F., Shen, J.: Stabilized semi-implicit spectral deferred correction methods for Allen-Cahn and Cahn-Hilliard equations. Math. Methods Appl. Sci. **38**(18), 4564–4575 (2015). DOI 10.1002/mma.2869. URL http://dx.doi.org/10.1002/mma.2869
45. Liu, Y., Shu, C.W., Zhang, M.: Strong stability preserving property of the deferred correction time discretization. J. Comput. Math. **26**(5), 633–656 (2008)
46. Minion, M.L.: Semi-implicit spectral deferred correction methods for ordinary differential equations. Commun. Math. Sci. **1**(3), 471–500 (2003). URL http://projecteuclid.org/euclid.cms/1250880097
47. Minion, M.L.: Semi-implicit projection methods for incompressible flow based on spectral deferred corrections. Appl. Numer. Math. **48**(3-4), 369–387 (2004). DOI 10.1016/j.apnum.2003.11.005. URL http://dx.doi.org/10.1016/j.apnum.2003.11.005. Workshop on Innovative Time Integrators for PDEs

48. Minion, M.L.: A hybrid parareal spectral deferred corrections method. Commun. Appl. Math. Comput. Sci. **5**(2), 265–301 (2010)
49. Minion, M.L., Speck, R., Bolten, M., Emmett, M., Ruprecht, D.: Interweaving PFASST and parallel multigrid. SIAM J. Sci. Comput. **37**(5), S244–S263 (2015). DOI 10.1137/14097536X. URL https://doi-org.cyber.usask.ca/10.1137/14097536X
50. Minion, M.L., Williams, S.A.: Parareal and spectral deferred corrections. AIP Conference Proceedings **1048**(1), 388–391 (2008). DOI 10.1063/1.2990941. URL https://aip.scitation.org/doi/abs/10.1063/1.2990941
51. Ong, B.W., Haynes, R.D., Ladd, K.: Algorithm 965: RIDC methods: A family of parallel time integrators. ACM Trans. Math. Softw. **43**(1), 8:1–8:13 (2016). DOI 10.1145/2964377. URL http://doi.acm.org/10.1145/2964377
52. Pazner, W.E., Nonaka, A., Bell, J.B., Day, M.S., Minion, M.L.: A high-order spectral deferred correction strategy for low mach number flow with complex chemistry. Combustion Theory and Modelling **20**(3), 521–547 (2016). DOI 10.1080/13647830.2016.1150519
53. Pereyra, V.: On improving an approximate solution of a functional equation by deferred corrections. Numer. Math. **8**, 376–391 (1966)
54. Pereyra, V.: Iterated deferred corrections for nonlinear boundary value problems. Numer. Math. **11**, 111–125 (1968)
55. Qu, W., Brandon, N., Chen, D., Huang, J., Kress, T.: A numerical framework for integrating deferred correction methods to solve high order collocation formulations of ODEs. J. Sci. Comput. **68**(2), 484–520 (2016). DOI 10.1007/s10915-015-0146-9. URL http://dx.doi.org/10.1007/s10915-015-0146-9
56. Rangan, A.V.: Adaptive solvers for partial differential and differential-algebraic equations. Ph.D. thesis, University of California, Berkeley (2003). URL 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
57. Ruprecht, D., Speck, R.: Spectral deferred corrections with fast-wave slow-wave splitting. SIAM J. Sci. Comput. **38**(4), A2535–A2557 (2016). DOI 10.1137/16M1060078. URL http://dx.doi.org/10.1137/16M1060078
58. Schild, K.H.: Gaussian collocation via defect correction. Numer. Math. **58**(4), 369–386 (1990). URL https://doi.org/10.1007/BF01385631
59. Skeel, R.D.: A theoretical framework for proving accuracy results for deferred corrections. SIAM J. Numer. Anal. **19**(1), 171–196 (1982). DOI 10.1137/0719009. URL http://dx.doi.org/10.1137/0719009
60. Speck, R.: Parallelizing spectral deferred corrections across the method. ArXiv e-prints (2017)
61. Speck, R.: Algorithm 997: pySDC—prototyping spectral deferred corrections. ACM Trans. Math. Software **45**(3), Art. 35, 23 (2019). DOI 10.1145/3310410. URL https://doi-org.cyber.usask.ca/10.1145/3310410
62. Speck, R., Ruprecht, D.: Toward fault-tolerant parallel-in-time integration with PFASST. Parallel Comput. **62**, 20–37 (2017). DOI 10.1016/j.parco.2016.12.001. URL https://doi-org.cyber.usask.ca/10.1016/j.parco.2016.12.001
63. Speck, R., Ruprecht, D., Emmett, M., Minion, M., Bolten, M., Krause, R.: A multi-level spectral deferred correction method. BIT **55**(3), 843–867 (2015). DOI 10.1007/s10543-014-0517-x. URL http://dx.doi.org/10.1007/s10543-014-0517-x
64. Speck, R., Ruprecht, D., Krause, R., Emmett, M., Minion, M., Winkel, M., Gibbon, P.: Integrating an N-body problem with SDC and PFASST. In: Domain decomposition methods in science and engineering XXI, *Lect. Notes Comput. Sci. Eng.*, vol. 98, pp. 637–645. Springer, Cham (2014)
65. Speck, R., Ruprecht, D., Minion, M., Emmett, M., Krause, R.: Inexact spectral deferred corrections. In: T. Dickopf, M.J. Gander, L. Halpern, R. Krause, L.F. Pavarino (eds.) Domain Decomposition Methods in Science and Engineering XXII, pp. 389–396. Springer International Publishing, Cham (2016)
66. Stetter, H.J.: Economical Global Error Estimation, pp. 245–258. Springer US, Boston, MA (1974). DOI 10.1007/978-1-4684-2100-2_19. URL https://doi.org/10.1007/978-1-4684-2100-2_19
67. Sun, G.: The deferred correction procedure for linear multistep formulas. J. Comput. Math. **3**(1), 41–49 (1985)
68. Tang, T., Zhao, W., Zhou, T.: Deferred correction methods for forward backward stochastic differential equations. Numerical Mathematics: Theory, Methods and Applications **10**(2), 222—242 (2017). DOI 10.4208/nmtma.2017.s02

69. Trefethen, L.N.: Spectral methods in MATLAB, *Software, Environments, and Tools*, vol. 10. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA (2000). DOI 10.1137/1.9780898719598. URL http://dx.doi.org/10.1137/1.9780898719598
70. Verwer, J.G., Sommeijer, B.P.: An implicit-explicit Runge–Kutta–Chebyshev scheme for diffusion-reaction equations. SIAM J. Sci. Comput. **25**(5), 1824—-1835 (elec) (2004). DOI 10.1137/S1064827503429168. URL http://dx.doi.org/10.1137/S1064827503429168
71. Weiser, M.: Faster SDC convergence on non-equidistant grids by DIRK sweeps. BIT Numerical Mathematics **55**(4), 1219–1241 (2015). DOI 10.1007/s10543-014-0540-y. URL https://doi.org/10.1007/s10543-014-0540-y
72. Weiser, M., Ghosh, S.: Theoretically optimal inexact spectral deferred correction methods. Commun. Appl. Math. Comput. Sci. **13**(1), 53–86 (2018). DOI 10.2140/camcos.2018.13.53. URL https://doi.org/10.2140/camcos.2018.13.53
73. Winkel, M., Speck, R., Ruprecht, D.: A high-order Boris integrator. Journal of Computational Physics **295**, 456–474 (2015). DOI https://doi.org/10.1016/j.jcp.2015.04.022. URL http://www.sciencedirect.com/science/article/pii/S0021999115002685
74. Xin, J., Huang, J., Zhao, W., Zhu, J.: A spectral deferred correction method for fractional differential equations. Abstr. Appl. Anal. pp. Art. ID 139530, 6 (2013)
75. Zadunaisky, P.E.: A method for the estimation of errors propagated in the numerical solution of a system of ordinary differential equations. In: G.I. Kontopoulos (ed.) The Theory of Orbits in the Solar System and in Stellar Systems, *IAU Symposium*, vol. 25, pp. 281–287 (1966)
76. Zadunaisky, P.E.: On the estimation of errors propagated in the numerical integration of ordinary differential equations. Numer. Math. **27**(1), 21–39 (1976/77)