ON THE CONVERGENCE OF SPECTRAL DEFERRED CORRECTION METHODS

MATHEW F. CAUSLEY AND DAVID C. SEAL

ABSTRACT. In this work we analyze the convergence properties of the Spectral Deferred Correction (SDC) method originally proposed by Dutt et al. [BIT, 40 (2000), pp. 241–266]. The framework for this high-order ordinary differential equation (ODE) solver is typically described wherein a low order approximation (such as forward or backward Euler) is lifted to higher order accuracy by applying the low order method to an error equation and then adding in the resulting defect to the current solution. Our focus is not on solving the error equation to increase the order of accuracy, but on rewriting the solver as an iterative Picard integral equation solver. In doing so, our chief finding is that it is not the low order solver that picks up the order of accuracy with each correction, nor is it the expression of the error equation, but it is the underlying quadrature rule of the right hand side function that is solely responsible for picking up additional orders of accuracy. Our proofs point to a total of three sources of errors that SDC methods carry: the error at the current time point, the error from the previous iterate, and the numerical integration error that comes from the total number of quadrature nodes used for integration. We indicate that the difference between the current and previous iterate always gets multiplied by a multiple of the time step size, and therefore it has no bearing on whether or not the solver picks up an order of accuracy. From this vantage, we solidify the prospects of extending spectral deferred correction methods to a larger class of solvers to which we present some examples.

1. INTRODUCTION

The spectral deferred correction (SDC) method defines a large class of ordinary differential equation (ODE) solvers that were originally introduced in 2000 by Dutt, Greengard and Rokhlin [11]. These type of methods are typically introduced by defining an error equation, and then repeatedly applying the same low-order solver to the error equation and adding the solution back into the current approximation in order to pick up an order of accuracy. This idea can be traced back to the work of Zadunaisky in 1976 [24], where the author sought out high order solvers in order to reduce numerical roundoff errors for astronomical applications. Before introducing the classical SDC methods defined in [11], we stop here to point out some of the recent work that has been happening over the past two decades including [6, 8, 9, 18, 21, 23, 25]. We refer the interested reader to [27] for a nice list of references for the first of these last two decades. Here, we provide a sampling of some of the current topics of interest to the community.

Many variations of the original SDC method are being studied as part of an effort to expedite the convergence of the solver. The chief goal here is to reduce the total number of iterations required to obtain the same high-order accuracy of

Date: Last modified: June 21, 2017.

1
the original method. These methods include the option of using Krylov deferred correction methods \cite{16,17} as well as the multi-level SDC methods \cite{22,30}. The multi-level approach starts with a lower order interpolant and then successively increases the degree of the interpolant with each future sweep of the method. This has the primary advantage of decreasing the overall number of function evaluations that need to be conducted, but introduces additional complications involving the need to evaluate interpolating polynomials. In the same vein, higher order embedded integrators have been explored within the so-called integral deferred correction (IDC) framework \cite{8,9}, where a moderate order solver (such as second or fourth-order Runge-Kutta method) is embedded inside a very high order SDC solver. With this framework, each successive correction increases the order by the same amount as that of the base solver. In addition, parallel in time solvers \cite{4,7,12,26} are being investigated as a mechanism to address the needs of modern high performance computing architectures, and adaptive time stepping options have been more recently investigated in \cite{5}. This work is based upon the nice property that SDC methods naturally embed a lower order solver inside a higher order solver.

In addition to the above mentioned extensions, various semi-implicit formulations have, and are currently being explored. While the original solver was meant for classical non-linear ODEs, semi-implicit formulations have been conducted as early as 2003 \cite{25} and are still an ongoing topic of research \cite{6,24}. The effect of the choice of correctors including second order semi-implicit solvers for the error equation has been conducted in \cite{20}, and an investigation into the efficiency of semi-implicit and multi-implicit spectral deferred correction methods for problems with varying temporal scales have been conducted in \cite{21}. Related high-order operator splitting methods have been proposed in \cite{2,3,14}, where the focus is not on an implicit-explicit splitting, but rather on splitting the right hand side of the ODE into smaller systems that can be more readily inverted with each sweep of the solver.

Very recent work includes applications of the SDC framework to generate exponential integrators of arbitrary orders \cite{1}, exploring interesting LU decompositions of the implicit Butcher tableau on non-equispaced grids \cite{32}, further investigation into high order operator splitting \cite{10}, a comparison of essentially non-oscillatory (ENO) versus piecewise parabolic method (PPM) coupled with SDC time integrators \cite{19}, and additional implicit-explicit (IMEX) splittings for fast-wave slow-wave splitting constructed from within the SDC framework \cite{29}.

It is not our aim to conduct a comprehensive review and comparison of all of these methods, rather it is our goal to present rigorous analysis of the original method that can be extended to these more complicated solvers. With that in mind, we now turn to a brief introduction of the spectral deferred correction framework, and in the process of doing so, we seek to directly compare this method with that of the Picard integral formulation of a numerical ODE solver.

1.1. Picard Iteration and the SDC Framework. We begin by giving a brief description of classical SDC methods. In doing so, we explain the differences between SDC and that of Picard iteration, which defines the cornerstone of the present work.

SDC solvers are designed to solve initial value problems of the form

\[ y' = \frac{dy}{dt} = f(y), \quad t > 0, \quad y(0) = y_0, \]  \hspace{1cm} (1)
where $y$ can be taken to be a vector of unknowns. The solution $y(t)$ can be expressed as an integral through formal integration:

$$y(t) = y_0 + \int_0^t f(y(s))ds, \quad t > 0.$$  

(2)

In this work, we assume that $f$ is Lipshitz continuous. That is, we assume

$$|f(z) - f(w)| \leq L|z - w|,$$

(3)

for some constant $L \geq 0$ and all $z, w \in \mathbb{R}$. This is sufficient to guarantee existence and uniqueness for exact solutions of the ODE. The focus of this work is to explore convergence properties of the spectral deferred correction (SDC) method where we produce rigorous numerical error bounds. These error bounds will rely on Lipshitz continuity of the right hand side of the ODE.

Consider a set of $M$ quadrature points $0 \leq \xi_1 < \ldots < \xi_M \leq 1$ that partition the unit interval into a total of $N$ disjoint subintervals, defined by

$$N = \begin{cases} 
M - 1 & \text{if both endpoints are used}, \\
M & \text{if only one endpoint is used}, \\
M + 1 & \text{if neither endpoint is used}.
\end{cases}$$

We make this choice because a given quadrature rule may or may not include the endpoints of the interval, and this convention allows us to study Gaussian quadrature rules, uniformly spaced quadrature rules, right Radau quadrature rules and others all within the same context. With that in mind, we define the right endpoints $\xi_R^n$, for $n = 0, 1, \ldots, N - 1$, of each the $N$ subintervals as

$$\xi_R^n = \begin{cases} 
\xi_{n+1} & \text{if the left endpoint is included}, \\
\xi_n & \text{if the left endpoint not included},
\end{cases}$$

and $\xi_R^0 = 0, \xi_R^N = 1$ for the two boundary edge cases. Next, we define quadrature weights by

$$w_{n,m} = \frac{\xi^n_{m}}{\xi^n_{m-1}} \ell_m(x)dx, \quad n = 1, 2, \ldots, N, \quad m = 1, 2, \ldots, M,$$

(4)

where $\ell_m(x)$ is the Lagrange interpolating polynomial of degree at most $M - 1$ corresponding to the quadrature point $\xi_m$:

$$\ell_m(x) = \frac{1}{c_m} \prod_{k=1, k \neq m}^M (x - \xi_k), \quad c_m = \prod_{k=1, k \neq m}^M (\xi_m - \xi_k).$$

(5)

Once these weights are obtained, approximate integral solutions, say $\eta_n \approx y(\xi^n_R h)$ for $h > 0$, can be formed via

**Fully Implicit Collocation**: $\eta_n = \eta_{n-1} + h \sum_{m=1}^M w_{n,m} f(\eta_m), \quad n = 1, 2, \ldots, N,$

(6)

whereas the exact solution $y_m := y(\xi^n_R h)$ satisfies the exact integral

$$y_n = y_{n-1} + \int_{t_{n-1}}^{t_n} f(y(t)) dt, \quad t_n = \xi^n_R h, \quad n = 1, 2, \ldots, N.$$

(7)
By convention, $\eta_0 := y_0$ is known to high order (because it comes from the previous time step), and $\eta_N \approx y(h)$ constitutes one “full” time step. Since each substep uses information from all substeps to construct the right-hand side, the solution is higher order, but also requires the solution of a nonlinear system of $M$ unknowns (one for each quadrature point) at each time step. Although this integrator has some very nice properties such as it can be made to be symplectic and $L$-stable for suitably chosen quadrature points, it is not typically used in practice given the additional storage requirements and the larger matrices that need to be inverted for each time step. This is particularly relevant when it is used as the base solver for a partial differential equation, but even these bounds are being explored as a viable option for PDE solvers such as the discontinuous Galerkin method [28].

Picard iteration (with numerical quadrature) defines a solver by iterating on a current solution $\eta^{[p]}_n$, $p \in \mathbb{Z}_{\geq 0}$ and then creates a better approximation through

\[
\text{Picard Iteration} : \quad \eta^{[p+1]}_n = \eta^{[p]}_n + h \sum_{m=1}^{M} w_{n,m} f(\eta^{[p]}_m), \quad n = 1, 2, \ldots N. \tag{8}
\]

Note that the current value $\eta^{[p+1]}_0 := \eta_0 \approx y_0$ is a known value that is equal to the exact solution up to high order. While this solver picks up a single order of accuracy with each correction, it has the unfortunate consequence of having a finite region of absolute stability.

The spectral deferred correction framework modifies the region of absolute stability by updating the solution through

\[
\text{Exp. SDC} : \quad \eta^{[p+1]}_n = \eta^{[p]}_n + h_n \left[ f(\eta^{[p+1]}_{n-1}) - f(\eta^{[p]}_{n-1}) \right] + h \sum_{m=1}^{M} w_{n,m} f(\eta^{[p]}_m), \tag{9}
\]

where $h_n = (\xi^R_n - \xi^R_{n-1}) h$ is the length of the $n^{th}$ sub-interval.

\textbf{Remark 1.} Although traditional SDC methods were originally cast as a method that corrects a provisional solution by solving an error equation, some modern descriptions of the same solver identify Eqn. (9) as the base solver, which has the added benefit of pointing out a solid link between SDC methods and that of iterative Picard integral equation solvers.

In order to construct methods that have more favorable regions of absolute stability for stiff problems, the implicit SDC framework exacts multiple backward Euler time steps through each iteration with

\[
\text{Imp. SDC} : \quad \eta^{[p+1]}_n = \eta^{[p]}_{n-1} + h_n \left[ f(\eta^{[p+1]}_{n-1}) - f(\eta^{[p]}_{n-1}) \right] + h \sum_{m=1}^{M} w_{n,m} f(\eta^{[p]}_m). \tag{10}
\]

Note that this framework allows for implicit and high order solutions to be constructed with greater computational efficiency when compared to the fully implicit collocation solver defined in Eqn. (6) because smaller systems need to be inverted in order to take a single time step.

\textbf{Remark 2.} It has been noted that the scaling in front of the $h_n$ term does not have impact on the order of accuracy [23]. It is our aim to solidify that claim with rigorous numerical bounds, which we do for both the explicit and implicit solvers.

Before doing so, we point out an aside that is in common with all SDC solvers.
Remark 3. If \( \lim_{p \to \infty} \eta_n^p = \eta_n \) converges, then the solutions to Eqns. (8) and (9) converge to that of Eqn. (6).

While this is a convenient and interesting observation, unfortunately, or perhaps fortunately, there are few if any SDC solvers that attempt to iterate to convergence for each time step. This type of iteration would make the solver prohibitively expensive for large scale problems. In addition, proving that the solver convergences would require additional tools, but convergence (for the ODE) is easier to find by refining in \( h \), rather than in the number of quadrature points and the number of corrections. That is to say, it is our belief that the strength of an SDC solver is in the construction of a high-order ODE integrator, which relies on a fixed number of quadrature points and correction steps, that does not need to be iterated to convergence for each time step.

1.2. An Outline of the Present Work. Despite the increasing popularity of spectral deferred correction solvers, very little work has been performed on convergence results for this large class of methods. The results that are currently in the literature [8, 9, 14, 15, 31] typically proceed via induction on the current order of the approximate solution, and they all hinge on solving the error equation, wherein the same low order solver is applied and then a defect, or correction, is added back into the current solution in order to increase its overall order of accuracy. In this work, we do not require the use of the error equation, nor do we work with any sort of defect such as that defined in [34], rather we instead focus on the Picard integral underpinnings inherent to all SDC methods. Our work solely uses fundamental numerical analysis tools: numerical interpolation and integration. While these tools do rely on quadrature rules, our proofs are generic enough to accommodate any set of quadrature points, which are an ongoing discussion in terms of how to construct base solvers.

Many convergence results for the method rely on the existence of a Butcher tableau (in which case one can think of an SDC method as a special case of a Runge-Kutta method), wherein the order conditions for the Runge-Kutta method are verified on a case by case basis. Because the construction of arbitrary order RK methods requires solving non-linear algebraic equations with dramatically increasing complexity for the order conditions, we view SDC methods as a nice toolkit for constructing arbitrary order methods.

In this work, we prove rigorous error bounds for both implicit and explicit SDC methods, and in doing so, we expect the reader will find that these methods can be thought of as being built upon classical Picard iteration. Our results are applicable for general quadrature rules, but unlike the findings found in [31], where convergence is proven using the error equation, our work relies on the fundamental mechanics behind why the solver works. That is, we point out that the primary contributor to the order of accuracy of the solver lies within the integral of the residual, and not necessarily the application of any base solver to an error equation.

Indeed, our proofs follow in similar manner to that required to prove the Picard–Lindelöf theorem, but our proofs take into account numerical quadrature errors and do not rely on exact integration of the right hand side function \( f(y) \). The primary differences between our proofs and that of the Picard-Lindelöf theorem are the following:
Spectral deferred correction methods require the use of numerical quadrature to approximate the integrals presented in the Picard-Lindelöf theorem. Our error estimates take into account any errors resulting from quadrature rules.

Each correction step in the implicit scheme defined in (10) requires a nonlinear inversion, whereas the Picard-Lindelöf theorem is typically proven using exact integration.

There are two main theorems in this work, one for explicit SDC methods, and one for implicit SDC methods. Both of these theorems produce rigorous error bounds and are applicable for generic quadrature rules, and each of them point to a total of three sources of error that SDC methods carry.

The outline of this paper is as follows. In Section 2, we present some necessary lemmas concerning error estimates for integrals of interpolants as well as some error estimates for sequences of inequalities that show up in our proofs. In Section 3, we present a convergence proof for explicit SDC, and in Section 4, we present a convergence proof for the implicit SDC method. In Section 5, we present some numerical results, where we compare explicit SDC methods with Picard iterative methods, we investigate modified implicit SDC methods, and we experiment with different semi-implicit formulations of SDC methods. Error estimates for all of these variants come from direct extensions of the proofs found in this work. Finally, some conclusions and suggestions for future work are drawn up in Section 6.

2. Preliminaries

We now point out a couple of important tools that we use to show that SDC solvers converge. Our aim is to focus on a single time step. Without loss of generality, from here on out we will focus on constructing a solution over the interval $[0, h]$, where $h$ is the time step size and we will assume that $\eta_0 \approx y_0$ is a high-order approximation to the exact solution.

2.1. Error estimates for integrals of interpolants. If \( \eta = (\eta_1, \eta_2, \cdots, \eta_M) \) is a set of discrete values and \( t \in [0, h] \) is a time interval we are interested in studying, we define the interpolation operator \( I \) to be the projection onto the space of polynomials of degree at most \( M - 1 \) via

\[
I[f(\eta)](t) := \sum_{m=1}^{M} f(\eta_m)l_m(t/h), \quad f(\eta) := (f(\eta_1), f(\eta_2), \cdots, f(\eta_M)). \tag{11}
\]

Note that this produces the integration identity

\[
\int_{t_{n-1}}^{t_n} I[f(\eta)](t) \, dt = h \sum_{m=1}^{M} w_{n,m}f(\eta_m) \tag{12}
\]

after integrating (11) over a subinterval \([t_{n-1}, t_n] := [h\xi_{n-1}, h\xi_n]\), and the weights are defined as in Eqn. (4).

Convergence results for both the explicit and the implicit SDC method (as well as Picard iteration) require the use of the following lemma. It is this Lemma that serves as the underlying tool that allows all SDC methods to pick up an order of accuracy with each correction step.
Lemma 2.1. Suppose that \( f \circ y \in C^M([0, h]) \), \( \|f \circ y\|_\infty \leq F \), and that \( f \) is Lipschitz continuous with Lipshitz constant \( L \). Then we have the estimate
\[
\left| \int_{t_{n-1}}^{t_n} I[f(\eta)](t) - f(y(t)) \, dt \right| \leq h\|\eta - y\|W_n + \frac{F}{M!}h^{M+1}.
\]
where the discrete norm is defined by
\[
\|e\| := \max_{1 \leq n \leq M} |e_n|, \quad e = (e_1, e_2, \ldots, e_M),
\]
and the constant \( W_n \) is defined by
\[
W_n := \sum_{m=1}^{M} \int_{R_{\xi}} l_m(\xi) d\xi.
\]
For a fixed quadrature rule, this constant is finite and independent of the function.

Proof. Add and subtract the Lagrange interpolant \( I[f(y)](t) \) for \( f \circ y \) inside the left hand side of Eqn. (13) and apply the triangle inequality:
\[
\left| \int_{t_{n-1}}^{t_n} I[f(\eta)](t) - f(y(t)) \, dt \right| \leq \left| \int_{t_{n-1}}^{t_n} I[f(\eta)](t) - I[f(y)](t) \, dt \right| + \left| \int_{t_{n-1}}^{t_n} I[f(y)](t) - f(y(t)) \, dt \right|.
\]
An estimate for the first of these two terms follows by linearity of the interpolation operator:
\[
\left| \int_{t_{n-1}}^{t_n} I[f(\eta)](t) - I[f(y)](t) \, dt \right| = h \sum_{m=1}^{M} \omega_{n,m} \left| f(\eta_m) - f(y_m) \right|
\leq h \sum_{m=1}^{M} |\omega_{n,m}| \left| f(\eta_m) - f(y_m) \right|
\leq hL \sum_{m=1}^{M} |\omega_{n,m}| |\eta_m - y_m|
\leq hL\|\eta - y\| \sum_{m=1}^{M} |\omega_{n,m}|.
\]
The quadrature weights in this estimate can be bounded above by \( |\omega_{n,m}| \leq \int_{R_{\xi}} |\ell_m(\xi)| d\xi \)
and then summed over all \( m \) to produce the constant \( W_n \).

The second of the two integrals in (16) is a function solely of the smoothness of \( f \) and the choice of the quadrature rule. That is, classical interpolation error estimates result in a bound on the \( M^{th} \) derivative of \( f \circ y \) through a single point \( z(t) \in [0, h] \) that yields
\[
|I[f(y)](t) - f(y(t))| = \left| \frac{(f \circ y)^{(M)}(z(t))}{M!} \prod_{m=1}^{M} (t - t_m) dt \right| \leq \frac{F}{M!} \prod_{m=1}^{M} |t - t_m|.
\] Because \( |t - t_m| \leq h \) for each \( m \), the result follows after integration.
\( \square \)
2.2. Error estimates for sequences of inequalities. Finally, we require a second Lemma as well as a simple Corollary. Both of these are stated in [13] and their proofs are elementary.

**Lemma 2.2.** If \( \{a_n\}_{n \in \mathbb{Z}_0} \) is a sequence that satisfies \( |a_n| \leq A|a_{n-1}| + B \) with \( A \neq 1 \), then
\[
|a_n| \leq A^n|a_0| + \frac{A^n - 1}{A - 1}B.
\]

**Proof.** Recursively apply the inequality and sum the remaining finite geometric series. \( \square \)

**Corollary 2.3.** If \( A > 1 \) and \( \{a_n\}_{n \in \mathbb{Z}} \) is a sequence that satisfies \( |a_n| \leq A|a_{n-1}| + B \), then
\[
|a_n| \leq A^n|a_0| + nA^{n-1}B
\]
for every \( n \).

**Proof.** By Lemma 2.2, the sequence satisfies Eqn. (19). We estimate the (finite) geometric series by
\[
\frac{A^n - 1}{A - 1} = 1 + A + \cdots + A^{n-1} \leq nA^{n-1}
\]
because there are a total \( n \) terms and each \( A^l \leq A^{n-1} \) for \( l = 0, 1, \ldots, n - 1 \). \( \square \)

With these preliminaries out of the way, we are now ready to prove convergence for either the explicit or implicit SDC method. We begin with the former and then continue with the latter.

3. Convergence Results for the Explicit SDC Method

Recall that the iterative update for the explicit SDC method is defined by Eqn. (9), which is given by
\[
\eta^{[p+1]}_n = \eta^{[p]}_n - h_n f(\eta^{[p]}_{n-1}) + h \sum_{m=1}^M w_{n,m} f(\eta^{[p]}_m), \quad n = 1, 2, \ldots, N,
\]
where \( \eta^{[p]}_0 = \eta_0 \) is a known quantity. This value is typically taken to be the result from the previous time step, and it is common to assume that this is known to high-order. Our focus is on the local truncation error, in which case we assume that the error at time zero is non-zero. That is, we assume \( \epsilon_0 = \eta_0 - y_0 \neq 0 \). Once this is established, a global error can be directly found.

The iterative process starts with a **provisional solution** \( \eta^{[0]}_n \) that approximates the exact solution \( \eta^{[0]}_n \approx y_n := y(\xi_R h) \). For explicit SDC, this is typically constructed through forward Euler time stepping:

**FE:** \( \eta^{[0]}_n = \eta^{[0]}_{n-1} + h_n f(\eta^{[0]}_{n-1}), \quad h_n = (\xi_R - \xi_{R-1})h, \quad n = 1, 2, \ldots, N \),

but our numerical (and analytical) results indicate the “predictor” step has little bearing on the overall order of accuracy of the solver. For example, it is possible to hold the solution constant for the initial iteration and still obtain high order accuracy, albeit with one additional iteration.
3.1. Convergence properties of the explicit SDC method. Our aim is to study the error defined by \( e_n^{[p]} := y_n^{[p]} - y_n \) where \( y_n := y(\xi_n^k h) \) is the exact solution of (1). With Lemmas 2.1 and 2.2 we now have the required tools to prove the first of two main results in this work: each correction step improves the order of accuracy of the method by one degree.

**Theorem 3.1.** The errors for a single step of the explicit SDC method satisfy

\[
|e_n^{[p+1]}| \leq e^{NhL}|e_0| + C_1h\|e^{[p]}\| + C_2h^{M+1},
\]

where \( N \) is the number of intervals under consideration, \( L \) is the Lipschitz constant of \( f \), and \( C_1 \) and \( C_2 \) are constants that depend only on \( f \), the exact solution \( y \), and the selection of quadrature points.

Before proving this theorem, we stop to point out an important observation.

**Remark 4.** The statement of this theorem highlights that there are a total of three sources of error that (explicit) SDC methods admit, which are ordered by appearance in the right hand side of (23):

1. the error at the current time step: \( e_0 = \eta_0 - y_0 \).
2. the error from the previous iterate (or predictor): \( e^{[p]} = \eta^{[p]} - y \), and
3. the number of quadrature points, \( M \).

The most important take-away is that because the error from the previous iterate, \( \|e^{[p]}\| \), gets multiplied by a factor of \( h \), gets improved by one order of accuracy with each correction. Of course this order reaches a maximum order based upon the number of the quadrature points chosen, which can be seen in the third source of error. This can be improved by selecting quadrature points with superconvergence properties such as the Gaussian or Gauss-Lobatto quadrature points.

**Proof.** We begin by subtracting the exact solution in (9) from the approximate solution in (8) to reach a discrete error evolution equation:

\[
e_n^{[p+1]} = e_{n-1}^{[p+1]} + h_n \left[ f(\eta^{[p+1]}_{n-1}) - f(\eta^{[p]}_{n-1}) \right] + \int_{t_{n-1}}^{t_n} I[f(\eta^{[p]}_n)](t) - f(y(t)) \, dt.
\]

Next, we take absolute values of (24) and start estimating each term through

\[
|e_n^{[p+1]}| \leq |e_{n-1}^{[p+1]}| + h_n \left| f(\eta^{[p+1]}_{n-1}) - f(\eta^{[p]}_{n-1}) \right| + |I_n| \tag{25a}
\]

\[
\leq |e_{n-1}^{[p+1]}| + h_n L \left| \eta^{[p+1]}_{n-1} - \eta^{[p]}_{n-1} \right| + |I_n|, \tag{25b}
\]

where

\[
I_n := \int_{t_{n-1}}^{t_n} I[f(\eta^{[p]}_n)](t) - f(y(t)) \, dt. \tag{26}
\]

First, we add and subtract \( y_{n-1} \) inside the second term and observe

\[
|\eta^{[p+1]}_{n-1} - \eta^{[p]}_{n-1}| = |\eta^{[p+1]}_{n-1} - y_{n-1} + y_{n-1} - \eta^{[p]}_{n-1}| \leq |e^{[p+1]}_{n-1}| + |e^{[p]}_{n-1}|. \tag{27}
\]

Next, we appeal to Lemma 2.1 to estimate \( I_n \) with

\[
|I_n| = \left| \int_{t_{n-1}}^{t_n} I[f(\eta^{[p]}_n)](t) - f(y(t)) \, dt \right| \leq h\|e^{[p]}\|W_n L + \frac{F}{M!} h^{M+1}. \tag{28}
\]
Putting this all together, we have
\[
|e^{[p+1]}_n| \leq (1 + h_n L) |e^{[p+1]}_{n-1}| + h (1 + W_n L) \|e^{[p]}\| + \frac{F}{M!} h^{M+1}
\]
\[
\leq (1 + hL) |e^{[p+1]}_n| + hW \|e^{[p]}\| + \frac{F}{M!} h^{M+1},
\]
where \( W := \max_{1 \leq n \leq N} (1 + W_n L) \). Next, we appeal to Corollary 2.3 and use \( A = (1 + hL) > 1 \) and \( B = hW \|e^{[p]}\| + \frac{F}{M!} h^{M+1} \) to observe
\[
|e^{[p+1]}_n| \leq (1 + hL)^n |e_0| + n(1 + hL)^{n-1} \left( hW \|e^{[p]}\| + \frac{F}{M!} h^{M+1} \right). \tag{30}
\]
Since \( 1 + hL \leq e^{hL} \) and \( n \leq N \), we have
\[
|e^{[p+1]}_n| \leq e^{nhL} |e_0| + ne^{(n-1)hL} \left( hW \|e^{[p]}\| + \frac{F}{M!} h^{M+1} \right).
\tag{31a}
\]
\[
\leq e^{NhL} |e_0| + N e^{NhL} \left( hW \|e^{[p]}\| + \frac{F}{M!} h^{M+1} \right). \tag{31b}
\]
With \( C_1 = N e^{NhL} W \) and \( C_2 = N e^{NhL} F/M! \), we conclude the desired result because these constants depend only on \( f \), the exact solution \( y \), and the number of quadrature points.

Before moving on to implicit solvers, we point out that we do not attempt a complete analysis of the stability properties for any of the solvers presented in this work. Changing the quadrature rules, the order of the predictors, and other options certainly changes the regions of absolute stability [23,24].

4. Convergence Results for the Implicit SDC Method

The implicit SDC method is traditionally based upon the backward Euler (BE) discretization, given by
\[
\text{BE} : \quad y(\xi h) \approx y_0 + \xi hf(y(\xi h)). \tag{32}
\]
This method serves as the base solver for the predictor, as well as each correction in a classical SDC method. That is, the provisional solution, or initial guess \( y^{[0]}_n \approx y(\xi_n h) \), is typically defined with
\[
y^{[0]}_n = y^{[0]}_{n-1} + h_n f(y^{[0]}_n), \quad n = 1, 2, \ldots N, \tag{33}
\]
where \( h_n = (\xi_n^R - \xi_{n-1}^R) h \). Recall that further corrections are defined by the implicit SDC method defined in Eqn. (10), which means that for each sub-interval \([t_{n-1}, t_n]\), we solve
\[
y^{[p+1]}_n = y^{[p+1]}_{n-1} + h_n \left[ f(y^{[p+1]}_n) - f(y^{[p]}_n) \right] + h \sum_{m=1}^{M} w_{n,m} f(y^{[p]}_m), \quad n = 1, 2, \ldots N, \tag{34}
\]
for each \( y^{[p+1]}_n \) in correction loop.

We repeat that the collocation method defined in (6) requires simultaneously solving for each \( \eta_n \) and is clearly more expensive than multiple applications of the backward Euler method. We also repeat that provided that if \( \eta^{[p]} \) converge as \( p \to \infty \), then Eqn. (10) defines a solution to Eqn. (6). Proving which initial guesses converge to the fully implicit solver is beyond the scope of this work. Currently,
our aim is to show that each correction step in the SDC framework picks up at least a single order of accuracy.

**Theorem 4.1.** The errors for a single step of the implicit SDC method satisfy

\[ |e_n^{[p+1]}| \leq e^{2NhL}|e_0| + C_1 h^p \|e^p\| + C_2 h^{M+1} \]  

provided \( h < 1/(2L) \). The constants \( C_1 \) and \( C_2 \) depend only on the smoothness of \( f \), the exact solution \( y \), and the choice of quadrature points.

This error estimate is very similar to that shown in Theorem 3.1 however this time the error constant in front of \( |e_0| \) is larger. In addition, we have an extra caveat regarding the maximum time step size in order to find this error involving the reciprocal of \( L \). These are typical assumptions when going about finding error constants for implicit methods.

**Proof.** We again construct a discrete error evolution equation by subtracting the exact solution defined in (7) from the iterative corrections in (10):

\[ e_n^{[p+1]} = e_{n-1}^{[p+1]} + h_n \left[ f(\eta_n^{[p+1]}) - f(\eta_n^p) \right] + \int_{t_{n-1}}^{t_n} I[f(\eta^p)](t) - f(y(t)) \, dt. \]

Identical to the explicit case, we estimate the third term with

\[ |I_n| := \left| \int_{t_{n-1}}^{t_n} I[f(\eta^p)](t) - f(y(t)) \, dt \right| \leq h^p \|e^p\| W_n L + \frac{F}{M!} h^{M+1}. \]

To estimate the other terms, we begin by taking absolute values of (36) and start estimating them with

\[ |e_n^{[p+1]}| \leq |e_{n-1}^{[p+1]}| + h_n \left| f(\eta_n^{[p+1]}) - f(\eta_n^p) \right| + |I_n| \]

\[ \leq |e_{n-1}^{[p+1]}| + h_n L \left| \eta_n^{[p+1]} - \eta_n^p \right| + |I_n| \]

\[ \leq |e_{n-1}^{[p+1]}| + h_n L \left( |e_n^{[p+1]}| + |e_n^p| \right) + |I_n|. \]

The third line follows by adding and subtracting \( y_n \) to the inside of the second term of the right hand side and then applying the triangle inequality.

We continue by subtracting \( h_n L |e_n^{[p+1]}| \) from both sides, dividing by \( 1 - h_n L > 0 \), recognizing that \( h_n < h \), and using Lemma 2.1 to estimate \( |I_n| \) to lead us to

\[ |e_n^{[p+1]}| \leq \frac{1}{1 - h_n L} \left( |e_{n-1}^{[p+1]}| + h_n L |e_n^p| + |I_n| \right) \]

\[ \leq \frac{1}{1 - hL} \left( |e_{n-1}^{[p+1]}| + (h + hW_n L) |e_n^p| + \frac{F}{M!} h^{M+1} \right) \]

\[ = \frac{1}{1 - hL} |e_{n-1}^{[p+1]}| + \frac{1}{1 - hL} \left( hW |e_n^p| + \frac{F}{M!} h^{M+1} \right), \]

where we again define \( W := \max_{1 \leq n \leq N} (1 + W_n L) \).

We make use of two separate estimates for \( 1/(1 - hL) \) to estimate the two terms found in the right hand side of (39). For the first term, we expand the geometric series and keep the first two terms:

\[ \frac{1}{1 - hL} = 1 + (hL) + (hL)^2 + \cdots = 1 + (hL) + (hL)^2 \frac{1}{1 - hL}. \]
This is valid because $hL < 1$. Moreover, because $hL < 1/2$, we have

$$hL < 1 - hL \implies \frac{(hL)^2}{1 - hL} < hL,$$

and therefore

$$\frac{1}{1 - hL} \leq 1 + 2hL \leq e^{2hL}.$$  \hspace{1cm} (42)

For the second term, we have $1/(1 - hL) \leq 2$ for all $hL \in [0, 1/2]$. This leads us to observe that

$$|e_n^{p+1}| \leq e^{2hL} |e_{n-1}^{p+1}| + 2 \left( hW \|e^p\| + \frac{F}{M!} h^{M+1} \right).$$ \hspace{1cm} (43)

Finally, we appeal to Corollary 2.3 with $A = e^{2hL} > 1$ and $B = 2 \left( hW \|e^p\| + \frac{F}{M!} h^{M+1} \right)$ to conclude that

$$|e_n^{p+1}| \leq e^{2hL} |e_0| + 2ne^{2(n-1)hL} \left( hW \|e_{n-1}^p\| + \frac{F}{M!} h^{M+1} \right).$$ \hspace{1cm} (44)

After recognizing $n \leq N$, we have the desired estimate with

$$C_1 = 2Ne^{2NhL}W \quad \text{and} \quad C_2 = 2Ne^{2NhL} \frac{F}{M!}.$$  \hspace{1cm} \square

It is worth noting that the error estimate provided here is an asymptotic error estimate. That is, one key assumption that we have to make is that $h < 1/(2L)$, which we did not have to make for the explicit case. Unfortunately, one key benefit of implicit solvers is that large time steps can be taken, in which case it is certainly possible that the solver does not obey this assumption. For these cases, a rigorous error estimate and analysis when $h > 1/(2L)$ would make for an interesting result, which would be especially important for multiscale problems that contain large time separations. This observation is beyond the scope of the present work.

5. Numerical results

The primary contribution of this work is to construct rigorous error estimates for classical SDC methods, and therefore we only include a couple of numerical results because an abundance of SDC examples applied to ordinary and partial differential equations can be found in the literature. With that being said, one of our key goals is to promulgate the primary source of high-order accuracy inherent in all SDC methods without relying on any error equations.

With this newfound knowledge, our goal with this section in particular is to demonstrate how classical SDC methods can extended to produce related high-order solvers. First, we introduce a comparison of errors (and stability regions) for explicit SDC vs. Picard iteration, second we explore modifications of the constant in front of an implicit SDC method, and finally we compare semi-implicit SDC and modified semi-implicit SDC solvers. For the sake of brevity the proposed modifications to SDC methods are not formally analyzed but straightforward extensions of the theorems presented in this work can be constructed to present formal error bounds for these methods. The numerical evidence presented here supports this claim.
5.1. A comparison of explicit SDC and Picard iteration. In this numerical example, we compare the errors and stability regions by applying the Picard iterative method defined in (8) to that of the explicit SDC defined in (9). In order to present an equal comparison of these two solvers, we consider identical initial guesses, or provisional solutions \( \eta^{[0]} \) based upon forward Euler time stepping.

5.1.1. Errors for a linear test case. In Table 1, we compare errors for the linear equation
\[
y' = \lambda y, \quad y(0) = 1
\]
at a final time of \( T = 10 \) with \( \lambda = -2 \). For the sake of brevity, we only compare fourth-order methods based upon a total of four equispaced quadrature points. In both methods, we apply a total of three corrections so the overall accuracy is fourth-order. Other values of \( \lambda \) show similar results where we observe slightly smaller error constants when using the Picard iterative method compared to the equivalent SDC method. This is consistent with the findings of Theorem 3.1, because the first error estimate
\[
|e^{[p+1]}_{n}| \leq |e^{[p+1]}_{n-1}| + h_{n} \left| f(\eta^{[p+1]}_{n-1}) - f(\eta^{[p]}_{n-1}) \right| + |I_{n}|
\]
could be tightened up to read
\[
|e^{[p+1]}_{n}| \leq |e^{[p+1]}_{n-1}| + |I_{n}|
\]
which produces a smaller (provable) overall error for the Picard method when compared to the SDC method.

| Mesh | SDC error  | Order | Picard error | Order |
|------|------------|-------|--------------|-------|
| 16   | 8.30 × 10^{-02} | —     | 2.47 × 10^{-02} | —     |
| 32   | 3.63 × 10^{-04} | 4.52  | 1.26 × 10^{-05} | 4.29  |
| 64   | 1.95 × 10^{-04} | 4.22  | 7.17 × 10^{-06} | 4.14  |
| 128  | 1.13 × 10^{-05} | 4.11  | 4.26 × 10^{-06} | 4.07  |
| 256  | 6.81 × 10^{-06} | 4.05  | 2.60 × 10^{-07} | 4.04  |
| 512  | 4.17 × 10^{-08} | 4.03  | 1.60 × 10^{-08} | 4.02  |
| 1024 | 2.58 × 10^{-09} | 4.01  | 9.95 × 10^{-10} | 4.01  |

5.1.2. A comparison of regions of absolute stability for explicit methods. Next, we seek to compare regions of absolute stability for explicit SDC methods and their Picard iterative cousins. Here we observe that the stability regions are slightly improved when the “Euler term” in the time stepping is dropped from the SDC method. That is to say, we find that the Picard iterative methods generally have larger regions of absolute stability when compared to their SDC counterparts.

In order to demonstrate this, in Figure 1, we include a comparison of plots of the regions of absolute stability, defined by
\[
\mathcal{D} := \{ z \in \mathbb{C} : |\rho(z)| < 1 \}
\]
where $\rho(z)$ is the amplification factor for various quadrature rules for both of these methods and $z := \lambda h$ and $\lambda$ is defined as in Eqn. [45]. There, we compare methods of orders two through ten, all based on equispaced quadrature points, forward Euler time stepping for the provisional solution, and the minimum number of corrections required to reach the desired order of accuracy. (For example, the third order method uses two corrections and the fifth order method uses four corrections.) Much like most Runge-Kutta methods, we find that the regions of absolute stability increase as the order is increased, but there are also more function evaluations per time step.

5.2. Implicit SDC methods with modified backward Euler time steps. Here, we consider implicit SDC methods with a variable constant in front of the
vanishing term:

\[ \eta_{n}^{[p+1]} = \eta_{n-1}^{[p+1]} + \theta h \left[ f(\eta_{n}^{[p+1]}) - f(\eta_{n}^{[p]}) \right] + h \sum_{m=1}^{M} w_{n,m} f(\eta_{m}^{[p]}), \quad n = 1, 2, \ldots, N. \] (47)

This same scaling has already been explored in [33] for SDC methods, but there the authors only consider the case where \(1/2 \leq \theta \leq 1\). With \(\theta = 0\), we have (explicit) Picard iteration (provided the provisional solution is modified), with \(\theta = 1\), we have the classical implicit SDC method, and with negative values of \(\theta\) we have backward Euler solves on negative time steps, but none of these changes effect the overall order of accuracy, only the size of the error constant and the regions of absolute stability. That is, a straightforward extension of Theorem 4.1 that includes values of \(\theta \neq 1\) would verify the high-order accuracy of this solver, but it would have a modified error constant. With \(\theta \gg 1\) we have a method that is heavy handed on multiple backward Euler solves, and therefore it has a very large region of absolute stability, however this method unfortunately introduces a larger error constant. Small values of \(\theta\) decrease this error constant, but they modify the regions of absolute stability to the point where they become finite and therefore undesirable because this is an implicit method. We first numerically validate the current claim that this method is genuinely high-order for various values of \(\theta\) before presenting results for regions of absolute stability.

5.2.1. A verification of high-order accuracy: The nonlinear pendulum problem. As a verification of the high order accuracy of the solver, we consider the nonlinear pendulum equation:

\[ x''(t) + \sin(x(t)) = 0 \] (48)

with appropriate initial conditions. If we perform the change of variables \(y_1(t) = x(t)\) and \(y_2(t) = x'(t)\), we end up with the following first-order nonlinear system of equations that is equivalent to Eqn. (48):

\[ (y_1, y_2)' = (y_2, -\sin(y_1)) . \] (49)

We consider initial conditions defined by \((y_1(0), y_2(0)) = (0, 1)\) and we integrate this problem to a final time of \(T = 10\). To compute an exact solution, we use MATLAB’s built-in ode45 with a relative tolerance of \(10^{-12}\) and an absolute tolerance of \(10^{-14}\). Convergence results are presented in Table 2 that indicate each method is indeed high order. For the sake of brevity, we only present results for the fourth-order solver that makes use of a total of three corrections. Tests of other methods indicate similar results.

5.2.2. A parameter study of regions of absolute stability for implicit methods. Given the results of the previous section, it would be tempting to want to set \(\theta = 0\) in order to reduce the total error. What is missing from this observation is an understanding of the regions of absolute stability to which we now turn. Our findings indicate that small values of \(\theta\) produce finite regions of absolute stability, and that large values of \(\theta\) increase the regions of absolute stability (when compared to classical SDC methods) but they also increase the stiffness of each implicit solve. With that being said, larger time steps should be able to be taken, but as is pointed out in the previous section, larger errors are introduced so there is the usual tradeoff between large time steps with large errors or smaller time steps with an increased computational cost.
Table 2. Nonlinear pendulum problem. Here, we compare SDC methods with different scalings on the backward Euler term as defined in Eqn. (47). The case with $\theta = 1$ is classical implicit SDC. We observe the expected result that all methods have high-order accuracy and that $\theta > 1$ produces larger error constants. The case with $\theta = -0.1$ is not a useful method because we will see that it has a finite region of absolute stability.

| Mesh | $\eta = -0.1$ | Order | $\eta = 0.5$ | Order | $\eta = 1.0$ | Order | $\eta = 1.5$ | Order |
|------|---------------|-------|--------------|-------|--------------|-------|--------------|-------|
| 40   | $2.46 \times 10^{-04}$ | 3.95  | $1.41 \times 10^{-05}$ | 4.17  | $3.27 \times 10^{-06}$ | 4.08  | $1.35 \times 10^{-07}$ | 2.42  |
| 80   | $9.54 \times 10^{-05}$ | 3.98  | $7.69 \times 10^{-06}$ | 4.11  | $1.84 \times 10^{-07}$ | 4.15  | $9.55 \times 10^{-09}$ | 3.83  |
| 160  | $9.01 \times 10^{-06}$ | 3.99  | $4.61 \times 10^{-07}$ | 4.06  | $1.08 \times 10^{-08}$ | 4.10  | $5.44 \times 10^{-10}$ | 4.13  |
| 320  | $3.77 \times 10^{-07}$ | 3.99  | $2.82 \times 10^{-08}$ | 4.03  | $6.49 \times 10^{-10}$ | 4.05  | $3.12 \times 10^{-11}$ | 4.12  |
| 640  | $3.70 \times 10^{-08}$ | 4.00  | $1.74 \times 10^{-09}$ | 4.01  | $3.98 \times 10^{-10}$ | 4.03  | $1.85 \times 10^{-11}$ | 4.07  |
| 1280 | $1.45 \times 10^{-09}$ | 4.03  | $1.11 \times 10^{-10}$ | 3.97  | $2.47 \times 10^{-11}$ | 4.01  | $1.13 \times 10^{-12}$ | 4.04  |
| 2560 | $7.59 \times 10^{-11}$ | 4.26  | $1.05 \times 10^{-12}$ | 4.40  | $1.56 \times 10^{-13}$ | 3.98  | $6.96 \times 10^{-14}$ | 4.02  |

In Figure 2, we present results for a third order method with various values of $\theta$, where we plot the modulus of the amplification factor $|\rho(z)|$ in place of the boundary defined by $|\rho(z)| = 1$ because if we were to plot the boundary then it would not be clear what parts are stable. In this sequence of images, we present results for various values of $\theta \in [-0.4, 5]$. Larger values of $\theta$ such as $\theta = 100$ look very similar to that of $\theta = 5$. Tests on methods of other orders produce similar results involving transitions between finite and infinite regions of absolute stability as $\theta$ increases from 0. The tradeoff between the size and shape of the stability regions for various quadrature rules is left for future work.

Before continuing, we stop to point out that diagonally implicit Runge-Kutta methods (on nonequispaced points) can be constructed from this very same framework. The point here is that because the term involving the difference $f(\eta_n^{[p+1]}) - f(\eta_n^{[p]})$ in Eqn. (47) does not contribute to the overall order of accuracy; the scaling in front of this term can be modified so that each implicit solve uses the exact same time step. This could be advantageous for easing the implementation of SDC methods in large scale code bases. In such a case the provisional solution would have to be modified in order to retain constant time steps for each stage in the solver. This would mean an extra correction or a (low order) polynomial interpolation step would be necessary to not lose the first order accuracy provisional solution, which would again modify the regions of absolute stability for solvers of various orders.

5.3. Modified semi-implicit SDC methods. Semi-implicit SDC methods (SISDC) have been used to generate arbitrary order implicit-explicit (IMEX) solvers for ordinary and partial differential equations [25]. These solvers typically begin with a partition of the right hand side into two new functions $f_I$ and $f_E$ via

$$y' = f(y), \quad f(y) = f_I(y) + f_E(y), \quad y(0) = y_0,$$

(50)
and then they apply a forward Euler/backward Euler (FE/BE) pair to the right hand side:

$$\eta_n^{[p+1]} = \eta_n^{[p+1]} + h_n \left[ f_I(\eta_n^{[p+1]}) - f_I(\eta_n^{[p]}) \right]$$

$$+ h_n \left[ f_E(\eta_n^{[p+1]}) - f_E(\eta_n^{[p]}) \right] + h \sum_{m=1}^{M} w_{n,m} f(\eta_m^{[p]}).$$  \(51\)

With a straightforward extension the results from the present work point out that high order accuracy can be achieved where there are no forward Euler time
steps on the explicit term \( f_E(y) \). That is, we propose examining the following modified SISDC method:

\[
\eta_n^{[p+1]} = \eta_n^{[p]} + h_n \left[ f_I(\eta_m^{[p+1]}) - f_I(\eta_m^{[p]}) \right] + h \sum_{m=1}^{M} w_{n,m} f_m(\eta_m^{[p]}). \tag{52}
\]

This serves as another example of how the findings from this work permit modifications to classical SDC methods in order to produce nearby variations. As an additional benefit, this reduces the computational coding complexity by asking the user to only define \( f \) and \( f_I \) as opposed to \( f, f_I \) and \( f_E \). This type of modification can only be found after understanding the source of high-order accuracy inherent to the SDC framework.

Given that the iterative Picard methods demonstrate smaller errors by dropping the forward Euler terms in the right hand of the iterations from the SDC solver, one might expect that this method has better accuracy than its SISDC parent. In the event when \( f_I = 0 \) (or is small), this would be true because in that case we would be comparing explicit SDC to Picard iteration, and we have already shown that those errors are smaller for some problems. However, we will shortly see that this is not necessarily the case. Even though this modification does not affect the overall order of the solver, we will show that for the following test case it does not improve the total overall error of the solver. With that being said, we believe it is still important to understand the source of the overall order of the SDC solvers, because only then can new methods be developed from the existing framework.

\subsection{Van der Pol’s equation.}

As a prototypical IMEX example, we include results for Van der Pol’s equation:

\[
x''(t) = -x(t) + \mu (1 - x(t))^2 x'(t)
\]

with appropriate initial conditions. After making the usual transformation of \( y_1(t) = x(t), \ y_2(t) = \mu x'(t), \) and rescaling time through \( t \to t/\mu \), we have the following system of differential equations \cite{20, 25}:

\[
y_1' = y_2, \quad y_2' = \frac{-y_1 + (1 - y_1^2)y_2}{\epsilon}, \quad \epsilon = \frac{1}{\mu^2}.
\tag{54}
\]

In an IMEX setting, this problem is typically split into \( f_E(y) = (y_2, 0) \) and \( f_I = (0, (-y_1 + (1 - y_1^2)y_2)/\epsilon) \) as an effort to account for the stiffness as \( \epsilon \to 0 \). For this problem we only seek to verify the high-order accuracy of the classical semi-implicit SDC method defined in (51), denoted by SISDC, as well as the modified solver defined in (52), denoted by “modified SISDC.” With this aim in mind, we set \( \epsilon = 1 \) so that the equations remain non-stiff, and we integrate to a moderate final time of \( T = 4 \). The initial conditions are the same as those found in an example in \cite{20}, which are \( y_1(0) = 2 \) and \( y_2(0) = -0.666666654321 \). In Table 3 we compare a convergence study for the fourth-order versions of these two methods where we use a total of four equispaced quadrature points, a provisional solution defined by the split forward/backward Euler method, as well as three corrections in the solver. For this problem, we find that the SISDC method has smaller errors, despite what the theory might otherwise predict we could observe. For problems where \( f_I \) is negligible or small, the modified method should outperform the SISDC solver.
Table 3. Van der Pol oscillator. Here we present numerical results where we compare the implicit classical method defined in (51) as well as the modified semi-implicit SDC method defined in (52) against each other. Despite the fact that theory can show that the errors could be smaller for the modified method that relies solely on backward Euler time stepping embedded within Picard iteration, in this case the classical SISDC method based forward/backward Euler time stepping outperforms the other solver with its smaller error constants.

| Mesh | SISDC | Order | Modified SISDC | Order |
|------|-------|-------|----------------|-------|
| 4    | 2.24 \times 10^{-02} | —     | 6.45 \times 10^{-02} | —     |
| 8    | 6.06 \times 10^{-04} | 5.21  | 2.84 \times 10^{-03} | 4.51  |
| 16   | 4.11 \times 10^{-05} | 3.88  | 1.91 \times 10^{-04} | 3.89  |
| 32   | 3.44 \times 10^{-06} | 3.58  | 1.46 \times 10^{-05} | 3.71  |
| 64   | 2.56 \times 10^{-07} | 3.75  | 1.01 \times 10^{-06} | 3.85  |
| 128  | 1.78 \times 10^{-08} | 3.85  | 6.68 \times 10^{-08} | 3.92  |
| 256  | 1.17 \times 10^{-09} | 3.93  | 4.29 \times 10^{-09} | 3.96  |
| 512  | 7.26 \times 10^{-11} | 4.01  | 2.69 \times 10^{-10} | 3.99  |

6. Conclusions

In this work we present rigorous error bounds for both explicit and implicit spectral deferred correction methods. Unlike most presentations that introduce SDC methods as a method that iteratively corrects provisional solutions by solving an error equation, our work hinges on the fact that the basic solver can be recast as a variation on Picard iteration. This observation allows new SDC methods to be developed through modifications of the (forward or backward) Euler part of the iterative procedure. In the numerical results section we present some sample variations that serve to indicate that the choice of the base solver is not as important as one might expect with an SDC method. This is because SDC methods are typically presented as a defect correction method that picks up the orders of accuracy by solving an error equation. Our findings indicate that it is not important to use the same low order solver for each correction step because all of the desired high-order accuracy can be found in the integral of the residual. Future work involves further analysis of embedded high-order base solvers as well as exploring further modifications of the solver to modify regions of absolute stability of existing solvers for explicit, implicit, and semi-implicit SDC methods.

References

[1] T. Buvoli, A class of exponential integrators based on spectral deferred correction, arXiv preprint arXiv:1504.05543 (2015).
[2] A. J. Christlieb, W. Guo, M. Morton, and J.-M. Qiu, A high order time splitting method based on integral deferred correction for semi-Lagrangian Vlasov simulations, J. Comput. Phys. 267 (2014), 7–27. MR 3183279
[3] A. J. Christlieb, Y. Liu, and Z. Xu, High order operator splitting methods based on an integral deferred correction framework, J. Comput. Phys. 294 (2015), 224–242. MR 3343724
[4] A. J. Christlieb, C. B. Macdonald, and B. W. Ong, *Parallel high-order integrators*, SIAM J. Sci. Comput. **32** (2010), no. 2, 818–835. MR 2609341

[5] A. J. Christlieb, C. B. Macdonald, B. W. Ong, and R. J. Spiteri, *Revisionist integral deferred correction with adaptive step-size control*, Commun. Appl. Math. Comput. Sci. **10** (2015), no. 1, 1–25. MR 3327725

[6] A. J. Christlieb, M. Morton, B. Ong, and J.-M. Qiu, *Semi-implicit integral deferred correction constructed with additive Runge-Kutta methods*, Commun. Math. Sci. **9** (2011), no. 3, 879–902. MR 2865808

[7] A. J. Christlieb and B. Ong, *Implicit parallel time integrators*, J. Sci. Comput. **49** (2011), no. 2, 167–179. MR 2837106

[8] A. J. Christlieb, B. Ong, and J.-M. Qiu, *Comments on high-order integrators embedded within integral deferred correction methods*, Commun. Appl. Math. Comput. Sci. **4** (2009), 27–56. MR 2516213 (2010e:65094)

[9] A. J. Christlieb, M. Morton, B. Ong, and J.-M. Qiu, *Integral deferred correction methods constructed with high order Runge-Kutta integrators*, Math. Comp. **79** (2010), no. 270, 761–783. MR 2600542 (2011c:65122)

[10] M. Duarte and M. Emmett, *High order schemes based on operator splitting and deferred corrections for stiff time dependent PDEs*, arXiv preprint arXiv:1407.0195 (2016).

[11] A. Dutt, L. Greengard, and V. Rokhlin, *Spectral deferred correction methods for ordinary differential equations*, BIT **40** (2000), no. 2, 241–266. MR 1765736 (2001c:65104)

[12] M. Emmett and M. L. Minion, *Toward an efficient parallel in time method for partial differential equations*, Commun. Appl. Math. Comput. Sci. **7** (2012), no. 1, 105–132. MR 2979518

[13] I. Faragó, *Note on the convergence of the implicit Euler method*, Numerical analysis and its applications, Lecture Notes in Comput. Sci., vol. 8236, Springer, Heidelberg, 2013, pp. 1–11. MR 3149968

[14] T. Hagstrom and R. Zhou, *On the spectral deferred correction of splitting methods for initial value problems*, Commun. Appl. Math. Comput. Sci. **1** (2006), 169–205. MR 2299441

[15] A. C. Hansen and J. Strain, *On the order of deferred correction*, Appl. Numer. Math. **61** (2011), no. 8, 961–973. MR 2802688

[16] J. Huang, J. Jia, and M. L. Minion, *Accelerating the convergence of spectral deferred correction methods*, J. Comput. Phys. **214** (2006), no. 2, 633–656. MR 2216607

[17] J. Huang, J. Jia, and M. L. Minion, *Arbitrary order Krylov deferred correction methods for differential algebraic equations*, J. Comput. Phys. **221** (2007), no. 2, 739–760. MR 2293148

[18] J. Jia, J. C. Hill, K. J. Evans, G. I. Fann, and M. A. Taylor, *A spectral deferred correction method applied to the shallow water equations on a sphere*, Monthly Weather Review **141** (2013), no. 10, 3435–3449.

[19] S. Y. Kadioglu and V. Colak, *An essentially non-oscillatory spectral deferred correction method for conservation laws*, International Journal of Computational Methods **0** (2016), no. 0, 1650027.

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

[21] A. T. Layton, *On the efficiency of spectral deferred correction methods for time-dependent partial differential equations*, Appl. Numer. Math. **59** (2009), no. 7,
1629–1643. MR 2512282

[22] A. T. Layton and M. L. Minion, Conservative multi-implicit spectral deferred correction methods for reacting gas dynamics, J. Comput. Phys. 194 (2004), no. 2, 697–715. MR 2034861

[23] , Implications of the choice of quadrature nodes for Picard integral deferred corrections methods for ordinary differential equations, BIT 45 (2005), no. 2, 341–373. MR 2176198 (2006h:65087)

[24] , Implications of the choice of predictors for semi-implicit Picard integral deferred correction methods, Commun. Appl. Math. Comput. Sci. 2 (2007), 1–34. MR 2327081 (2008e:65252)

[25] M. L. Minion, Semi-implicit spectral deferred correction methods for ordinary differential equations, Commun. Math. Sci. 1 (2003), no. 3, 471–500. MR 2069941 (2005f:65085)

[26] , A hybrid parareal spectral deferred corrections method, Commun. Appl. Math. Comput. Sci. 5 (2010), no. 2, 265–301. MR 2765386

[27] M. M. Morton, Integral Deferred Correction methods for scientific computing, ProQuest LLC, Ann Arbor, MI, 2010, Thesis (Ph.D.)–Michigan State University. MR 2801737

[28] W. Pazner and P.-O. Persson, Stage-parallel fully implicit Runge-Kutta solvers for discontinuous Galerkin fluid simulations, J. Comput. Phys. 335 (2017), 700–717. MR 3612518

[29] D. Ruprecht and R. Speck, Spectral deferred corrections with fast-wave slow-wave splitting, SIAM J. Sci. Comput. 38 (2016), no. 4, A2535–A2557. MR 3537015

[30] R. Speck, D. Ruprecht, M. Emmett, M. L. Minion, M. Bolten, and R. Krause, A multi-level spectral deferred correction method, BIT 55 (2015), no. 3, 843–867. MR 3401816

[31] T. Tang, H. Xie, and X. Yin, High-order convergence of spectral deferred correction methods on general quadrature nodes, J. Sci. Comput. 56 (2013), no. 1, 1–13. MR 3049939

[32] M. Weiser, Faster SDC convergence on non-equidistant grids by DIRK sweeps, BIT 55 (2015), no. 4, 1219–1241. MR 3434037

[33] Y. Xia, Y. Xu, and C.-W. Shu, Efficient time discretization for local discontinuous Galerkin methods, Discrete Contin. Dyn. Syst. Ser. B 8 (2007), no. 3, 677–693. MR 2328730

[34] P. E. Zadunaisky, On the estimation of errors propagated in the numerical integration of ordinary differential equations, Numer. Math. 27 (1976/77), no. 1, 21–39. MR 0431696

Kettering University, Department of Mathematics, 1700 University Ave. Flint, MI 48504, USA

E-mail address: mcausley@kettering.edu

U.S. Naval Academy, Department of Mathematics, 572C Holloway Road, Annapolis, MD 21402, USA

E-mail address: seal@usna.edu