SIAM J. SCI. COMPUT.
Vol. 42, No. 1, pp. A1–A27

A CLASS OF EXPONENTIAL INTEGRATORS BASED ON SPECTRAL DEFERRED CORRECTION*

TOMMASO BUVOLI†

Abstract. This paper introduces a new class of exponential integrators based on spectral deferred correction. These new methods are simple to implement at any order of accuracy and can be used to efficiently solve initial value problems when high precision is desired. We begin by deriving exponential spectral deferred correction (ESDC) methods for solving both partitioned and unpartitioned initial value problems. We then analyze the linear stability properties of these new integrators and show that they are comparable to those of existing semi-implicit spectral deferred correction schemes. Finally, we present five numerical experiments to demonstrate the improved efficiency of our new exponential integrator compared to semi-implicit spectral deferred correction schemes and existing fourth-order exponential Runge–Kutta methods.

Key words. spectral deferred correction, exponential integrators, semi-explicit, high-order, stiff systems, Fourier spectral methods

AMS subject classifications. 65L04, 65L05, 65L06, 65N35

DOI. 10.1137/19M1256166

1. Introduction.

Spectral deferred correction (SDC) methods, originally developed by Dutt, Greengard, and Rokhlin [13], are a class of time-integrators for solving ordinary differential equations (ODEs). All SDC methods iteratively improve the accuracy of a provisional solution by approximating a Picard integral equation that governs error. High-order SDC schemes offer improved stability in comparison to linear multistep methods and improved efficiency when compared with high-order extrapolation schemes. Since every SDC method can be written as either an explicit or a diagonally implicit Runge–Kutta method, we can view the SDC framework as an alternative approach to deriving Runge–Kutta methods of arbitrary order.

Over the past decade the SDC framework has continued to improve. Layton and Minion studied the effects of quadrature nodes [36], and Hansen and Strain produced a formal proof of convergence [19]. Christlieb, Ong, and Qiu generalized the framework by introducing integral deferred correction schemes [10], along with their parallel counterpart reductionist integral deferred correction [11]. For stiff partial differential equations, Minion and Christlieb respectively introduced semi-implicit SDC (SISDC) methods [40, 35] and semi-implicit integral deferred correction methods [9]. Both classes of integrators allow for efficient high-order integration and have proven effective on a range of different problems. The SDC framework has also proven invaluable for developing a host of parallel-in-time integrators [11, 14, 55, 62, 54].

In this work, we extend the SDC framework to include exponential integration. Exponential integrators [25] are a well-known class of time-stepping methods for solving stiff partial differential equations. For many application problems exponential integrators are more efficient than both fully implicit and semi-implicit integrators [18, 30, 37, 41]. Many different classes of exponential integrators have been previously

*Submitted to the journal's Methods and Algorithms for Scientific Computing section April 15, 2019; accepted for publication (in revised form) September 27, 2019; published electronically January 7, 2020.

https://doi.org/10.1137/19M1256166

Funding: This work was partially supported by the Applied Mathematics Department at the University of Washington and NSF grant DMS-1216732.

†Department of Applied Mathematics, University of California, Merced, Merced, CA 95343 (tbuvoli@ucmerced.edu).

studied, including exponential linear multistep methods [4], exponential Runge–Kutta methods [12, 30, 33, 23, 24, 31, 25], and exponential general linear methods [46]. The new class of exponential SDC (ESDC) methods introduced in this paper showcase an alternative construction strategy for deriving exponential integrators of arbitrary order without needing to explicitly solve order conditions and with improved stability compared to multistep-type schemes. This allows us to derive new exponential integrators of extremely high order (in this work we test integrators of order up to 32) that outperform existing state-of-the-art exponential Runge–Kutta schemes when high accuracy is required.

One particular area of interest is spectral methods, which have proven to be remarkably successful when applied to nonlinear wave equations [16, 59, 5]. When using high-order spectral methods to solve partial differential equations with smooth solutions, a traditional low-order time-integrator limits the overall order of accuracy. As we show in this paper, high-order ESDC integrators enable highly accurate numerical simulations at reduced computational cost and allow practitioners who apply high-order spatial discretizations to also integrate in time using a high-order method.

The paper is organized as follows. In sections 2 and 3 we provide introductions to SDC methods and exponential integrators. In section 4 we present the ESDC method along with a simple procedure for initializing its coefficients. Next, in section 5, we analyze and compare the stability regions of ESDC to those of SISDC methods. Finally, in section 6, we perform numerical experiments comparing ESDC to SISDC and the fourth-order exponential Runge–Kutta integrators ETDRK4 [12, 30] and EPIRK4s3 [38].

2. The spectral deferred correction framework. We first review SDC methods from [13] before extending the SDC framework to include exponential integration. For simplicity we consider the first-order scalar initial value problem

$$\begin{aligned} y'(t) &= F(t, y(t)), \\ y(t_0) &= y_0, \end{aligned} \tag{2.1}$$

though all the results in this paper are directly applicable to first-order multidimensional systems.

2.1. Euler-based SDC methods. Like all Runge–Kutta methods, an SDC method accepts the input y_n and produces the output y_{n+1} where $y_j \approx y(t_j)$. During the timestep from t_n to $t_{n+1} = t_n + h$, an SDC method subdivides the interval $[t_n, t_{n+1}]$ into p substeps

$$t_n \leq t_{n,1} < t_{n,2} < \cdots < t_{n,p} \leq t_{n+1}.$$

Using the initial condition y_n, the method computes a provisional solution at each substep $t_{n,j}$ using a low-order integrator. We will denote the provisional solution as

$$Y_{n,j}^{[1]} \approx y(t_{n,j}), \qquad\qquad j = 1, \ldots, p.$$

An SDC method then iteratively improves the accuracy of the provisional solution by approximating its error and performing an update of the form

$$Y_{n,j}^{[k+1]} = Y_{n,j}^{[k]} + E_{n,j}^{[k]}, \qquad\qquad j = 1, \ldots, p, \tag{2.2}$$

where each $E_{n,j}^{[k]}$ is an estimate of the error of the approximate solution $Y_{n,j}^{[k]}$, such that

$$E_{n,j}^{[k]} \approx y(t_{n,j}) - Y_{n,j}^{[k]}. \tag{2.3}$$

The update procedure (2.2) is repeated for $k = 1, \ldots, m$ using increasingly accurate error estimations. Once the approximate solution has reached the desired order of accuracy, an output value y_{n+1} is computed. If the final substep $t_{n,p}$ is equal to the output time t_{n+1}, then the solution value $Y_{n,p}^{[m]}$ is the output. If the final substep $t_{n,p}$ is smaller than t_{n+1}, then the method must also compute the final solution y_{n+1} at the next timestep. This is typically done by forming a final collocation solution but can also be achieved using interpolation.

The error at each substep is approximated by time-stepping an integral equation that governs its evolution. Different integral equations lead to different types of SDC methods. In the following two subsections we rederive the integral equation originally proposed by Dutt, Greengard, and Rokhlin and discuss their numerical procedure for approximating it. We then generalize this idea to derive a new error equation that leads to an ESDC method.

2.1.1. The classical integral equation for error. To derive the integral equation governing the error that was originally described in [13], we first consider the integral form of (2.1),

$$y(t) = y(a) + \int_a^t F(s, y(s))ds. \tag{2.4}$$

Next, let $\widetilde{y}(t)$ be an approximate solution to (2.4) where the error $E(t)$ is

$$E(t) = y(t) - \widetilde{y}(t).$$

Substituting $y(t) = \widetilde{y}(t) + E(t)$ into (2.4) produces the integral equation

$$E(t) = -\widetilde{y}(t) + \widetilde{y}(a) + E(a) + \int_a^t F(s, \widetilde{y}(s) + E(s))ds.$$

We can rewrite this integral equation as

$$E(t) = \underbrace{E(a) + \int_a^t G(s, E(s))ds}_{\text{Error Term}} + \underbrace{\left[\widetilde{y}(a) + \int_a^t F(s, \widetilde{y}(s))ds\right] - \widetilde{y}(t)}_{\text{Residual Term } R(t,a)}, \tag{2.5}$$

where the residual term $R(t, a)$ does not depend on the error $E(s)$ and

$$G(s, E(s)) = F(s, \widetilde{y}(s) + E(s)) - F(s, \widetilde{y}(s)). \tag{2.6}$$

2.1.2. Estimating the error of an approximate solution. We can now timestep the integral equation (2.5) to obtain error estimates for improving any approximate solution. Given a discrete approximate solution $Y_{n,j}^{[k]}$, $j = 1, \ldots, p$, we first write a continuous approximation using the Lagrange interpolating polynomial

$$L^{[k]}(t) = \sum_{j=1}^{p} Y_{n,j}^{[k]} \ell_j(t), \quad \text{where} \quad \ell_j(t) = \prod_{\substack{m=1 \\ m \neq k}}^{p} \frac{t - t_{n,m}}{t_{n,j} - t_{n,m}}. \tag{2.7}$$

To obtain the error estimates

$$E_{n,j}^{[k]} \approx y(t_{n,j}) - Y_{n,j}^{[k]},$$

an SDC method timesteps the integral equation (2.5) where $\tilde{y}(s)$ has now been replaced with $L^{[k]}(s)$. The time-stepping starts at $t = t_n$ and advances to each substep $t_{n,j}$. The method originally proposed by Dutt, Greengard, and Rokhlin for approximating error is an Euler-like scheme that can be derived by:

1. Approximating the integral in the residual term $R(t, a)$ using a quadrature formula with nodes at $t_{n,j}$, $j = 1, \ldots, p$. This choice of nodes allows us to use the derivative terms $F^{[k]}_{n,j} = F(t_{n,j}, Y^{[k]}_{n,j})$ as the function values for the quadrature.
2. Approximating the remaining integral term in (2.5) with an implicit or explicit Euler approximation.

These choices lead to the implicit ($\gamma = 1$) or explicit ($\gamma = 0$) time-stepping method

$$E^{[k]}_{n,j+1} = E^{[k]}_{n,j} + h_{n,j} G^{[k]}_{n,j+\gamma} + R^{[k]}_{n,j}, \qquad j = 0, \ldots, p-1, \tag{2.8}$$

where the substeps $h_{n,j}$ and the values $G^{[k]}_{n,j+\gamma}$ are

$$h_{n,j} = t_{n,j+1} - t_{n,j}, \tag{2.9}$$

$$\begin{aligned} G^{[k]}_{n,j+\gamma} &= G(t_{n,j+\gamma}, E^{[k]}_{n,j+\gamma}) \\ &= F\left(t_{n,j+\gamma}, Y^{[k]}_{n,j+\gamma} + E^{[k]}_{n,j+\gamma}\right) - F\left(t_{n,j+\gamma}, Y^{[k]}_{n,j+\gamma}\right), \end{aligned} \tag{2.10}$$

and the approximate residual $R^{[k]}_{n,j}$ is

$$R^{[k]}_{n,j} = Y^{[k]}_{n,j} - Y^{[k]}_{n,j+1} + \sum_{i=1}^{p} w_{ji} F(t_{n,i}, Y^{[k]}_{n,i}) \quad \text{for} \quad w_{ji} = \int_{t_{n,j}}^{t_{n,j+1}} \ell_i(s) ds.$$

The initial conditions for the iteration are always taken to be

$$t_{n,0} = t_n, \quad Y^{[k]}_{n,0} = y_n, \quad E^{[k]}_{n,0} = 0 \quad \forall k.$$

At first glance the time-stepping scheme (2.8) may appear to be only first-order accurate due to the Euler-like approximation. However, the accuracy of the method is limited by the accuracy of the quadrature approximation and the accuracy of the approximate solution. In general, if $F(t, y)$ is Lipschitz in its second argument and the discrete approximate solution values $Y^{[k]}_{n,j}$ are all order q accurate, then the error estimates will be at least order $\min(p + 1,\ q + 1)$. See the supplemental materials, section SM1, for more details.

2.1.3. The full SDC timestep. If we start with a first-order accurate provisional solution computed using Euler's method, then we can repeat the iteration (2.8) followed by (2.2) a total of m times to obtain an increasingly accurate sequence of approximations

$$\left\{Y^{[1]}_{n,j}\right\}_{j=1}^{p}, \ \left\{Y^{[2]}_{n,j}\right\}_{j=1}^{p}, \ldots, \ \left\{Y^{[m+1]}_{n,j}\right\}_{j=1}^{p}.$$

This full correction procedure constitutes a single timestep of an Euler-based SDC method. For a complete discussion of the error and convergence properties of SDC methods we refer the reader to the work of Hansen and Strain [19] or the work of Causley and Seal [8].

TABLE 1

Pseudocode for a single timestep of Euler SDC methods where $t_{n,p} = t_n$. If $t_{n,1} = t_n$, then the pseudocode can be simplified by letting $Y^{[k]}_{n,1} = y_n$ and running all loops with j from $j = 1$ to $p-1$. Finally, if $t_{n,p} < t_{n+1}$, then the last line of code must be replaced with a procedure for computing y_{n+1} which may require additional right-hand side evaluations or implicit solves.

Implicit ($\gamma = 1$) or Explicit ($\gamma = 0$) $\mathbf{SDC^m_p}$

$E^{[k]}_0 = 0$, $Y^{[k]}_{n,0} = y_n$, $t_{n,0} = t_n$
for j = 0 to $p-1$
  $Y^{[1]}_{n,j+1} = Y^{[1]}_{n,j} + h_{n,j} F^{[1]}_{n,j+\gamma}$
for k = 1 to m
  for j = 0 to $p-1$
    $Y^{[k+1]}_{n,j+1} = Y^{[k+1]}_{n,j} + h_{n,j}\left[F^{[k+1]}_{n,j+\gamma} - F^{[k]}_{n,j+\gamma}\right] + \sum_{i=1}^{p} w_{ji} F^{[k]}_{n,i}$.
$y_{n+1} = Y^{[m+1]}_{n,p}$

For implementation purposes, it is convenient to rewrite the update procedure more compactly. Using (2.2), we can rewrite the constants $G^{[k]}_{n,j+\gamma}$ from (2.10) as

$$G^{[k]}_{n,j+\gamma} = F\left(t_{n,j}, Y^{[k+1]}_{n,j+\gamma}\right) - F\left(t_{n,j}, Y^{[k]}_{n,j+\gamma}\right).$$

Then by substituting the expression (2.8) into the update formula (2.2), we obtain

$$Y^{[k+1]}_{n,j+1} = Y^{[k+1]}_{n,j} + h_{n,j}\left[F^{[k+1]}_{n,j+\gamma} - F^{[k]}_{n,j+\gamma}\right] + \sum_{i=1}^{p} w_{ji} F^{[k]}_{n,i}, \tag{2.11}$$

where $F^{[k]}_{n,i} = F(t_{n,i}, Y^{[k]}_{n,i})$. In Table 1 we present the pseudocode for Euler-based SDC schemes using this compact update formula.

3. Exponential integrators. Exponential integrators are a class of numerical methods for solving stiff initial value problems [25]. They are derived from the prototypical semilinear equation

$$\mathbf{y}' = \mathbf{L}\mathbf{y} + N(t, \mathbf{y}), \tag{3.1}$$

where the linear operator $\mathbf{L}$ and the nonlinear operator $N(t, \mathbf{y})$ are chosen so that they approximate the original ODE problem. Typically the linear operator $\mathbf{L}$ captures the majority of the stiffness so that its spectral radius is significantly larger than that of the locally linearized nonlinearity.

Regardless of how $\mathbf{L}$ and $N(t, \mathbf{y})$ are chosen, exponential integrators are derived by applying the discrete variation of constants formula to (3.1) and then approximating the nonlinearity $N(t, \mathbf{y})$ with a polynomial

$$\mathbf{y}(t) = e^{(t-a)\mathbf{L}}\mathbf{y}(a) + \int_a^t e^{(t-s)\mathbf{L}} \underbrace{N(s, \mathbf{y}(s))ds}_{\text{Approximate with a polynomial.}}. \tag{3.2}$$

For example, by approximating the nonlinearity using the zeroth-order polynomial $p(s) = N(t_n, \mathbf{y}_n)$, where $\mathbf{y}_n = \mathbf{y}(t_n)$, we obtain the exponential Euler method [45, 4, 12, 39, 56, 25]

$$\mathbf{y}_{n+1} = e^{h\mathbf{L}}\mathbf{y}_n + \left(\mathbf{L}^{-1}\left(\exp(h\mathbf{L}) - \mathbf{I}\right)\right) N(t_n, \mathbf{y}_n). \tag{3.3}$$

More sophisticated exponential multistep methods can be derived by approximating the nonlinearity with a high-order polynomial that interpolates values of $N(s, \mathbf{y})$ computed at previous timesteps [4]. Similarly, exponential Runge–Kutta methods compute stage values and outputs by replacing the nonlinearity with linear combinations of $N(s, \mathbf{y})$ values evaluated at previous stages [12].

The coefficients of many exponential integrators are expressed in terms of φ-functions [25, 3, 39, 32], where the jth φ-function is defined as

$$\varphi_j(z) = \begin{cases} e^z, & j = 0, \\ \dfrac{1}{(n-1)!} \displaystyle\int_0^1 e^{(1-s)z} s^{j-1} ds, & j > 0, \end{cases} \tag{3.4}$$

or alternatively using the recursive formula

$$\varphi_{j+1}(z) = z^{-1}(\varphi_j(z) - 1/n), \quad \varphi_0(z) = e^z.$$

The first few φ-functions are given by

$$\varphi_0(z) = e^z, \quad \varphi_1(z) = \frac{e^z - 1}{z}, \quad \varphi_2(z) = \frac{e^z - 1 - z}{z^2}, \quad \varphi_3(z) = \frac{e^z - 1 - z - \frac{1}{2}z^2}{z^3}.$$

Using these definitions, we can write the exponential Euler method (3.3) as

$$y_{n+1} = \varphi_0(h\mathbf{L})y_n + h\varphi_1(h\mathbf{L})N(t_n, \mathbf{y}_n).$$

Every step of an exponential integrator requires the computation of matrix-vector products with φ-functions. These operations can be very expensive if computed naively. Furthermore, the explicit formulas for the φ-functions possess a removable discontinuity at $z = 0$ that can cause catastrophic numerical roundoff error for matrices with eigenvalues near the origin.

Fortunately, over the last two decades a number of different algorithms have been developed for efficiently computing either $\varphi_j(\mathbf{L})$ or the action of $\varphi_j(\mathbf{L})$ on a vector $\mathbf{b}$. These include scaling and squaring methods [32, 2], contour integration methods [30, 60], Krylov-subspace methods [22, 21, 43, 42, 17], parallel rational approximations [20, 51], and hybrid approaches that evaluate parallel rational functions arising from contour integrals [52].

3.1. Partitioned and unpartitioned integrators. Exponential integrators can be classified depending on the choice of linear operator $\mathbf{L}$ and the nonlinear operator $N(t, y)$. In this work, we distinguish between two particular cases: unpartitioned integrators (e.g., [48, 56, 57]) and partitioned integrators (e.g., [12, 4, 49]). Note that sometimes these two classes have also been referred to as unsplit and split.

A *partitioned exponential integrator* can only be applied to semilinear initial value problems of the form (3.1). Fortunately, systems arising from the spatial discretizations of many partial differential equations naturally lend themselves to this partitioning. A few canonical examples include Burgers, nonlinear Schrödinger, and Korteweg–de Vries; further examples can be found in [12, 30, 18]. In each case we may write a partitioned system where $\mathbf{L}$ is the sum of all the discretized linear derivative operators.

An *unpartitioned exponential integrator* can be used to solve the more general initial value problem $\mathbf{y}' = F(t, \mathbf{y})$. This is accomplished by rewriting the right-hand side in terms of its local linearization at each timestep. To obtain a localized semilinear problem at the timestep $t = t_n$, one rewrites the system in its autonomous

form

$$\mathbf{y}' = F(\mathbf{y}) \tag{3.5}$$

and then re-expresses $F(\mathbf{y})$ as

$$\begin{aligned} F(\mathbf{y}) &= F(\mathbf{y}_n) + \mathbf{J}_n(\mathbf{y} - \mathbf{y}_n) + R(\mathbf{y}), \\ R(\mathbf{y}) &= F(\mathbf{y}) - [F(\mathbf{y}_n) + \mathbf{J}_n(\mathbf{y} - \mathbf{y}_n)], \end{aligned} \tag{3.6}$$

where $\mathbf{y}_n = \mathbf{y}(t_n)$ and $\mathbf{J}_n = \frac{\partial F}{\partial y}(\mathbf{y}(t_n))$ is the Jacobian of $\mathbf{y}(t)$ at $t = t_n$. The linear operator $\mathbf{J}_n$ takes the place of $\mathbf{L}$ in (3.1), and the remaining terms form the nonlinearity. If we are solving the system (3.6) using this approach, then the exponential Euler method becomes

$$\mathbf{y}_{n+1} = e^{h\mathbf{J}_n}\mathbf{y}_n + \left(\frac{\exp(h\mathbf{J}_n) - \mathbf{I}}{h\mathbf{J}_n}\right) h\left[F(\mathbf{y}_n) - \mathbf{J}_n\mathbf{y}_\mathbf{n}\right], \tag{3.7}$$

which can be simplified as $\mathbf{y}_{n+1} = \mathbf{y}_n + h\varphi_1(h\mathbf{J}_n)F(\mathbf{y}_n)$.

Partitioned exponential integrators have certain computational advantages over unpartitioned methods. Since the linear operator $\mathbf{L}$ is constant across all timesteps, it is possible to either precompute matrix functions or apply fast exponentiation methods that exploit the structure of the linear operator. However, not all initial value problems are semilinear. Moreover, certain semilinear problems have stiff nonlinear terms that can cause partitioned integrators to become unstable. Since we are interested in creating new ESDC methods for solving the widest range of problems, we extend the SDC framework to encompass both partitioned and unpartitioned exponential integration.

4. Exponential spectral deferred correction methods. We now extend the SDC framework to encompass exponential integration. Like its classical counterpart, an ESDC method computes a low-order provisional solution

$$Y_{n,j}^{[1]} \approx y(t_{n,j}), \quad j = 1, \ldots, p.$$

However, instead of using implicit or explicit Euler, the provisional solution is now computed using the exponential Euler method. Next an ESDC method approximates the error of the provisional solution by substepping an exponential integral equation governing error, and then computes an improved solution using (2.2). This full correction procedure is again repeated a total of m times.

4.1. The exponential error integral equation. To derive the integral equation for ESDC methods we start with the integral equation for (3.1),

$$\mathbf{y}(t) = e^{\mathbf{L}(t-a)}\mathbf{y}(a) + \int_a^t e^{\mathbf{L}(t-s)}N(s, \mathbf{y}(s))ds. \tag{4.1}$$

As before, let $\widetilde{\mathbf{y}}(t)$ be an approximate solution to (4.1) where the error $E(t)$ is

$$E(t) = \mathbf{y}(t) - \widetilde{\mathbf{y}}(t).$$

Substituting $\mathbf{y}(t) = \widetilde{\mathbf{y}}(t) + E(t)$ into (4.1) produces the integral equation

$$E(t) = -\widetilde{\mathbf{y}}(t) + e^{\mathbf{L}(t-a)}\left[\widetilde{\mathbf{y}}(a) + E(a)\right] + \int_a^t e^{\mathbf{L}(t-s)}N(s, \tilde{\mathbf{y}}(s) + E(s))ds.$$

By introducing the exponential residual

$$\mathbf{r}(t,a) = \left[e^{\mathbf{L}(t-a)}\widetilde{\mathbf{y}}(a) + \int_a^t e^{\mathbf{L}(t-s)}N(s,\widetilde{\mathbf{y}}(s))ds\right] - \widetilde{\mathbf{y}}(t), \tag{4.2}$$

we can rewrite the integral equation as

$$E(t) = e^{\mathbf{L}(t-a)}E(a) + \int_a^t e^{\mathbf{L}(t-s)}H(E(s))ds + \mathbf{r}(t,a), \tag{4.3}$$

where

$$H(E(s)) = N(s,\widetilde{\mathbf{y}}(s) + E(s)) - N(s,\widetilde{\mathbf{y}}(s)). \tag{4.4}$$

4.2. Estimating the error of an approximate solution. We now derive an exponential method for solving the integral equation (4.3). Given the discrete approximate solution $Y_{n,j}^{[k]}$, $j = 1,\dots,p$, we can write a continuous approximation for $N(s,\mathbf{y})$ using the Lagrange interpolating polynomial

$$P^{[k]}(t) = \sum_{j=1}^{p} N_{n,j}^{[k]}\ell_j(t), \quad \text{where} \quad \ell_j(t) = \prod_{\substack{m=1\\m\neq j}}^{p} \frac{t - t_{n,m}}{t_{n,j} - t_{n,m}}, \tag{4.5}$$

and $N_{n,j}^{[k]} = N(t_{n,j}, Y_{n,j}^{[k]})$. To obtain the error estimates

$$E_{n,j}^{[k]} \approx \mathbf{y}(t_{n,j}) - Y_{n,j}^{[k]},$$

an ESDC method timesteps the integral equation (4.3) where $N(s,\tilde{y}(s))$ has been replaced with $P^{[k]}(s)$. We can derive an exponential counterpart to the method (2.8) by:

1. Approximating the integral in the residual term $\mathbf{r}(t,a)$ by replacing $N(s,\tilde{y}(s))$ with the polynomial $P^{[k]}(s)$. This amounts to applying a weighted quadrature formula with nodes $t_{n,j}$, $j = 1,\dots,p$, and weight $e^{\mathbf{L}(t-s)}$. In section 4.4 we describe how to express this quadrature formula in terms of the φ-functions.
2. Approximating the remaining integral term in (4.3) by replacing $H(E(s))$ with either the explicit approximation $H(E(s)) \approx H(E(a))$ or the implicit approximation $H(E(s)) \approx H(E(t))$.

These choices produce the implicit ($\gamma = 1$) or explicit ($\gamma = 0$) exponential time-stepping method

$$E_{n,j+1}^{[k]} = \varphi_0(h_{n,j}\mathbf{L})E_{n,j}^{[k]} + h_{n,j}\varphi_1(h_{n,j}\mathbf{L})H_{n,j+\gamma}^{[k]} + \tilde{R}_{n,j}^{[k]}, \quad j = 0,\dots,p-1, \tag{4.6}$$

where the substeps $h_{n,j}$ and the values $H_{n,j+\gamma}^{[k]}$ are

$$h_{n,j} = t_{n,j+1} - t_{n,j}, \tag{4.7}$$

$$\begin{aligned} H_{n,j+\gamma}^{[k]} &= H(t_{n,j+\gamma}, E_{n,j+\gamma}^{[k]}) \\ &= N\left(t_{n,j+\gamma}, Y_{n,j+\gamma}^{[k]} + E_{n,j+\gamma}^{[k]}\right) - N\left(t_{n,j+\gamma}, Y_{n,j+\gamma}^{[k]}\right), \end{aligned} \tag{4.8}$$

and the approximate exponential residual $\tilde{R}_{n,j}^{[k]}$ is

$$\tilde{R}_{n,j}^{[k]} = \left[\varphi_0(h_{n,j}\mathbf{L})Y_{n,j}^{[k]} + \int_{t_{n,j}}^{t_{n,j+1}} e^{\mathbf{L}(t_{n,j+1}-s)}P^{[k]}(s)ds\right] - Y_{n,j+1}^{[k]}. \tag{4.9}$$

Once again, the initial conditions for the iteration are always taken to be

$$t_{n,0} = t_n, \quad Y_{n,0}^{[k]} = y_n, \quad E_{n,0}^{[k]} = 0 \quad \forall k.$$

As was the case for the classical SDC, the error estimates will be order $\min(p+1, q+1)$ accurate if the discrete approximate solution is order q accurate; see supplemental materials section SM1 for more details.

4.3. The full ESDC timestep. The general procedure for computing a timestep of an ESDC method is identical to that of classical SDC. By repeating a set of nearly equivalent substitutions to those described in section 2.1.3, we can derive the compact update formula

$$Y_{n,j+1}^{[k+1]} = \varphi_0(h_{n,j}\mathbf{L})Y_{n,j}^{[k+1]} + h_{n,j}\varphi_1(h_{n,j}\mathbf{L})\left[N_{n,j+\gamma}^{[k+1]} - N_{n,j+\gamma}^{[k]}\right] + I_{n,j}^{[k]}, \tag{4.10}$$

$$I_{n,j}^{[k]} = \int_{t_{n,j}}^{t_{n,j+1}} e^{\mathbf{L}(t_{n,j+1}-s)}P^{[k]}(s)ds, \tag{4.11}$$

where $\gamma \in \{0, 1\}$. To obtain an ESDC method for solving the unpartitioned problem (3.6) we make the substitutions

$$\mathbf{L} = \mathbf{J}_n, \quad N(s, \mathbf{y}(s)) = F(\mathbf{y}(s)) - \mathbf{J}_n\mathbf{y}(s), \quad \text{and} \quad \varphi_0(\mathbf{A})\mathbf{y} = \mathbf{y} + \varphi_1(\mathbf{A})\mathbf{A}\mathbf{y}$$

and simplify to obtain

$$Y_{n,j+1}^{[k+1]} = Y_{n,j}^{[k+1]} + h_{n,j}\varphi_1(h_{n,j}\mathbf{J}_n)\left[F_{n,j+\gamma}^{[k+1]} - F_{n,j+\gamma}^{[k]}\right] + \hat{I}_{n,j}^{[k]}, \tag{4.12}$$

$$\hat{I}_{n,j}^{[k]} = \int_{t_{n,j}}^{t_{n,j+1}} e^{\mathbf{L}(t_{n,j+1}-s)}\left[Q^{[k]}(s) + \mathbf{J}_n\left(Y_{n,j}^{[k]} - y_n\right) + F(y_n)\right]ds, \tag{4.13}$$

where $Q^{[k]}(s)$ is a Lagrange polynomial $Q^{[k]}(t) = \sum_{j=1}^{p} R_{n,j}^{[k]}\ell_j(t)$ and $R_{n,j}^{[k]} = R(Y_{n,j}^{[k]})$. The key steps for the derivation of the unpartitioned ESDC are contained in Appendix A.

In Table 2 we present pseudocode for a partitioned and unpartitioned ESDC method. Both types of ESDC integrators are explicit if $\gamma = 0$ and implicit if $\gamma = 1$. For implicit ESDC methods it is necessary to implicitly solve nonlinear systems containing exponential matrix functions. Due to the increased computational cost of these operations, we do not discuss the implicit method any further. From here on we use ESDC_p^m to denote an explicit exponential spectral deferred correction method that performs m correction iterations on p quadrature points.

4.4. Rewriting the exponential integrals in terms of φ-functions. At each substep, partitioned ESDC and unpartitioned ESDC respectively require the exponential integral terms (4.11) and (4.13). In order to apply existing algorithms to compute these terms, it is convenient to re-express both integrals using the φ-functions defined in (3.4). Below we derive the formula for partitioned ESDC methods and then simply write the corresponding formula for unpartitioned ESDC.

TABLE 2

Pseudocode for a single timestep of a partitioned or unpartitioned ESDC method where $t_{n,p} = t_n$. The formula for the exponential integral terms $I_{n,j}^{[k]}$ and $\hat{I}_{n,j}^{[k]}$ is written in terms of φ-functions in section 4.4. If $t_{n,1} = t_n$, then the pseudocode can be simplified by taking $Y_{n,1}^{[k]} = y_n$ and running all loops with j from 1 to $p-1$. Finally, for methods where $t_{n,p} < t_{n+1}$, the last line of code must be replaced with an additional procedure for computing y_{n+1} which may require additional right-hand side evaluations and matrix-vector products.

Partitioned explicit ($\gamma = 0$) or implicit ($\gamma = 1$) ESDC$_{\mathbf{p}}^{\mathbf{m}}$

$Y_{n,0}^{[k]} = y_n, \;\; t_{n,0} = t_n, \;\; N_{n,j}^{[k]} = N(t_{n,j}, Y_{n,j}^{[k]})$
for j $= 0$ to $p-1$
  $Y_{n,j+1}^{[1]} = \varphi_0(h_{n,j}\mathbf{L})Y_{n,j}^{[1]} + h_{n,j}\varphi_1(h_{n,j}\mathbf{L})N_{n,j+\gamma}^{[1]}$
for k $= 1$ to m
  for j $= 0$ to $p-1$
    $Y_{n,j+1}^{[k+1]} = \varphi_0(h_{n,j}\mathbf{L})Y_{n,j}^{[k+1]} + h_{n,j}\varphi_1(h_{n,j}\mathbf{L})\left[N_{n,j+\gamma}^{[k+1]} - N_{n,j+\gamma}^{[k]}\right] + I_{n,j}^{[k]}$
$y_{n+1} = Y_{n,p}^{[m+1]}$

Unpartitioned explicit ($\gamma = 0$) or implicit ($\gamma = 1$) ESDC$_{\mathbf{p}}^{\mathbf{m}}$

$Y_{n,0}^{[k]} = y_n, \;\; t_{n,0} = t_n, \;\; \mathbf{J}_n = \frac{\partial F}{\partial y}(y_n), \;\; F_{n,j}^{[k]} = F(t_{n,j}, Y_{n,j}^{[k]})$
for j $= 0$ to $p-1$
  $Y_{n,j+1}^{[1]} = Y_{n,j}^{[1]} + h_{n,j}\varphi_1(h_{n,j}\mathbf{J}_n)F_{n,j+\gamma}^{[1]}$
for k $= 1$ to m
  for j $= 0$ to $p-1$
    $Y_{n,j+1}^{[k+1]} = Y_{n,j}^{[k+1]} + h_{n,j}\varphi_1(h_{n,j}\mathbf{J}_n)\left[F_{n,j+\gamma}^{[k+1]} - F_{n,j+\gamma}^{[k]}\right] + \hat{I}_{n,j}^{[k]}$
$y_{n+1} = Y_{n,p}^{[m+1]}$

1. *Partitioned ESDC.* A partitioned ESDC$_p^m$ method requires the weighted quadrature terms (4.11) where $P^{[k]}(s)$ is the Lagrange polynomial (4.5). By applying the change of variables

$$s = h_{n,j}\sigma + t_{n,j} \tag{4.14}$$

to (4.11) we obtain

$$I_{n,j}^{[k]} = h_{n,j}\int_0^1 e^{h_{n,j}\mathbf{L}(1-\sigma)}Q_j(\sigma)d\sigma, \tag{4.15}$$

where $Q_j(\sigma)$ is a Lagrange polynomial that interpolates the values $N_{n,i}$ at the nodes

$$\tau_{j,i} = \frac{t_{n,i} - t_{n,j}}{h_{n,j}}, \qquad i = 1, \ldots, p. \tag{4.16}$$

Note that the nodes $\tau_{j,i}$ are simply the substeps $t_{n,i}$ under the transformation (4.14). To express (4.15) using φ-functions we rewrite the polynomial $Q_j(\sigma)$ as the Taylor polynomial expanded at $\sigma = 0$,

$$Q_j(\sigma) = \sum_{\nu=0}^{p-1}\sigma^\nu \sum_{l=1}^{p} a_{j,l}^{(\nu)} N_{n,l}^{[k]},$$

where $a_{j,l}^{(\nu)}$ are the finite difference weights that approximate the νth derivative of a function $f(\sigma)$ at $\sigma = 0$ using the values $f(\tau_{j,i})$, $i = 1, \ldots, p$. Substituting the Taylor form of $Q_j(\sigma)$ into (4.15) and simplifying leads to

$$I_{n,j}^{[k]} = h_{n,j} \sum_{\nu=1}^{p} \varphi_\nu(h_{n,j}\mathbf{L})\mathbf{b}_\nu^{[k]}, \quad \text{where} \quad \mathbf{b}_\nu^{[k]} = \sum_{l=1}^{p} a_{j,l}^{(\nu)} N_{n,l}^{[k]}. \tag{4.17}$$

Alternatively we can reorder the terms and write the integral as

$$I_{n,j}^{[k]} = h_{n,j} \sum_{l=0}^{p} w_l(h_{n,j}\mathbf{L}) N_{n,l}^{[k]}, \quad \text{where} \quad w_l(\mathbf{A}) = \sum_{\nu=1}^{p} a_{j,l}^{(\nu)} \varphi_\nu(\mathbf{A}). \tag{4.18}$$

For ESDC implementations where the φ-functions are directly computed, it will be more efficient to store the functions $w_l(h_{n,j}\mathbf{L})$, rather than $\varphi_\nu(h_{n,j}\mathbf{L})$, and use (4.18).

2. *Unpartitioned ESDC.* by applying an identical set of substitutions, we obtain a similar expression for the exponential terms required for unpartitioned ESDC:

(4.19)

$$\hat{I}_{n,j}^{[k]} = h_{n,j}\varphi_1(h_{n,j}\mathbf{J}_n)\left[\mathbf{J}_n\left(Y_{n,j}^{[k]} - y_n\right) + F(y_n)\right] + h_{n,j}\sum_{\nu=1}^{p}\varphi_\nu(h_{n,j}\mathbf{J}_n)\mathbf{b}_\nu^{[k]},$$

$$\mathbf{b}_\nu^{[k]} = \sum_{l=1}^{p} a_{j,l}^{(\nu)} R(Y_{n,l}^{[k]}).$$

For both partitioned and unpartitioned ESDC, we can obtain the finite difference weights $a_{j,l}^{(\nu)}$ by inverting the $p \times p$ Vandermonde matrix $\mathbf{V}(j)_{c,d} = (\tau_{j,c})^{d-1}$ and letting

$$a_{j,l}^{(\nu)} = \nu! \mathbf{V}(j)_{\nu+1,l}^{-1}.$$

Unfortunately, Vandermonde matrices are notoriously ill-conditioned, and any coefficients obtained from direct inversion will be prone to numerical rounding errors even if p is small. To avoid this problem, we use the fast and stable, double-precision algorithm developed by Fornberg for computing finite difference weights [15].

4.5. A remark regarding convergence. As the number of solution correction iterations tend to infinity, an ESDC method converges to a fully implicit exponential Runge–Kutta method with abscissa $c_j = (t_{n,j} - t_n)/h$ for $j = 1, \ldots, p$. To obtain the limiting method, we substitute $Y_{n,j}^{[k+1]} = Y_{n,j}^{[k]}$ into the ESDC method and rewrite the exponential integrals in local coordinates. For a partitioned ESDC with $t_{n,p} = t_n$, the underlying Runge–Kutta method is

$$\begin{aligned}
Y_0 &= y_n, \\
Y_{j+1} &= \varphi_0(\eta_j h\mathbf{L})Y_j + h\int_{c_j}^{c_{j+1}} e^{h\mathbf{L}(c_{j+1}-s)}P(s)ds, \quad j = 1, \ldots, p-2, \\
y_{n+1} &= \varphi_0(\eta_{p-1}h\mathbf{L})Y_{p-1} + h\int_{c_{p-1}}^{c_p} e^{h\mathbf{L}(c_p-s)}P(s)ds,
\end{aligned}$$

where $\eta_j = c_{j+1} - c_j$ and $P(s)$ is a Lagrange polynomial passing through the points $(c_j, N(t_{n,j}, Y_j))$ for $j = 1, \ldots, p$. Similarly, for an unpartitioned ESDC with $t_{n,p} = t_n$,

the underlying Runge–Kutta method is

$$
\begin{aligned}
Y_0 &= y_n, \\
Y_{j+1} &= Y_j + h \int_{c_j}^{c_{j+1}} e^{h\mathbf{J}_n(c_{j+1}-s)} \left[Q(s) + \mathbf{J}_n\left(Y_j - y_n\right) + F(y_n)\right] ds, \quad j = 1, \ldots, p-2, \\
y_{n+1} &= Y_{p-1} + h \int_{c_{p-1}}^{c_p} e^{h\mathbf{J}_n(c_p-s)} \left[Q(s) + \mathbf{J}_n\left(Y_j - y_n\right) + F(y_n)\right] ds,
\end{aligned}
$$

where $Q(s)$ is a Lagrange polynomial passing through the points $(c_j, R(t_{n,j}, Y_j))$ for $j = 1, \ldots, p$.

5. Linear stability. Now that we have introduced ESDC methods, we analyze their linear stability properties. To determine the linear stability regions for unpartitioned ESDC we consider the Dahlquist test problem

$$y' = \lambda y. \tag{5.1}$$

Unpartitioned integrators are constructed using the linearization (3.6) and integrate (5.1) exactly. Therefore, their linear stability regions are always equal to the region $\mathrm{Re}(h\lambda) \leq 0$.

To analyze linear stability for partitioned methods we consider the generalized Dahlquist test problem

$$y' = \lambda_1 y + \lambda_2 y. \tag{5.2}$$

The terms $\lambda_1 y$ and $\lambda_2 y$ are meant to respectively represent the linear and nonlinear terms. Equation (5.2) has been previously used to study linear stability for a variety of other integrators, including additive integrators [50], IMEX integrators [1, 28, 29, 7], and partitioned exponential integrators [12, 33, 18].

Applying a one-step partitioned method to (5.2) produces a recursion relation of the form $y_{n+1} = R(z_1, z_2)y_n$, where $z_1 = h\lambda_1$, $z_2 = h\lambda_2$, and h denotes the timestep. The region of absolute stability is then defined as

$$\mathcal{S} = \left\{(z_1, z_2) \in \mathbb{C}^2 : |R(z_1, z_2)| \leq 1\right\}.$$

The stability region $\mathcal{S}$ is four-dimensional and cannot be shown directly. Instead, we overlay two-dimensional slices of the stability regions formed by fixing z_1 and allowing z_2 to vary. (This is the same approach used in [4, 12, 33].) We pick a range of real, imaginary, and complex z_1 values to simulate initial value problems arising from the spatial discretization of partial differential equations with varying degrees of linear dispersion and dissipation. The specific values of z_1 we consider are

$$
\begin{aligned}
&\text{real-valued} && z_1 \in -1 \cdot [0, 30], \\
&\text{imaginary} && z_1 \in i \cdot [0, 30], \\
&\text{complex-valued} && z_1 \in \exp(3\pi i/4) \cdot [0, 30].
\end{aligned}
$$

We present the stability functions $R(z_1, z_2)$ for partitioned ESDC and SISDC from [40]. The stability function for both methods is defined recursively and written in terms of the normalized substeps $\eta_j = (t_{n,j} - t_n)/h$.

The formula for the stability function of an SDC_p^m method where $t_{n,p} = t_n$ is

$$R(z_1, z_2) = y_p^{[m+1]},$$

where $y_0^{[k]} = 1$ and $y_p^{[m+1]}$ is defined recursively via the following:

1. Partitioned exponential SDC_p^m:

$$\begin{aligned} y_{j+1}^{[1]} &= \varphi_0(z_1\eta_j)y_j^{[1]} + (\eta_j z_2)\varphi_1(z_1\eta_j)y_j^{[1]} \\ y_{j+1}^{[k+1]} &= \varphi_0(z_1\eta_j)y_j^{[k+1]} + (\eta_j z_2)\varphi_1(z_1\eta_j)(y_j^{[k+1]} - y_j^{[k]}) + I_j^{[k]} \\ I_j^{[k]} &= (z_2\eta_j)\sum_{\nu=1}^{p}\left[\varphi_\nu(z_1\eta_j)\sum_{l=1}^{p} a_{j,l}^{(\nu)}\, y_l^{[k]}\right]. \end{aligned}$$

2. SISDC_p^m:

$$\begin{aligned} y_{j+1}^{[1]} &= \left(\frac{1+z_2\eta_j}{1-z_1\eta_j}\right) y_j^{[1]}, \\ y_{j+1}^{[k+1]} &= \frac{y_j^{[k+1]} - (\eta_j z_1)y_{j+1}^{[k]} + (\eta_j z_2)(y_j^{[k+1]} - y_j^{[k]}) + I_j^{[k]}}{1 - r\eta_i}, \\ I_j^{[k]} &= (z_1+z_2)\sum_{\nu=1}^{p}\frac{(\eta_{j+1}-\eta_j)^\nu}{\nu!}\sum_{l=1}^{p} a_{j,l}^{(\nu)} y_l^{[k]}. \end{aligned}$$

The coefficients $a_{jl}^{(\nu)}$ are the finite difference weights described in subsection 4.4.

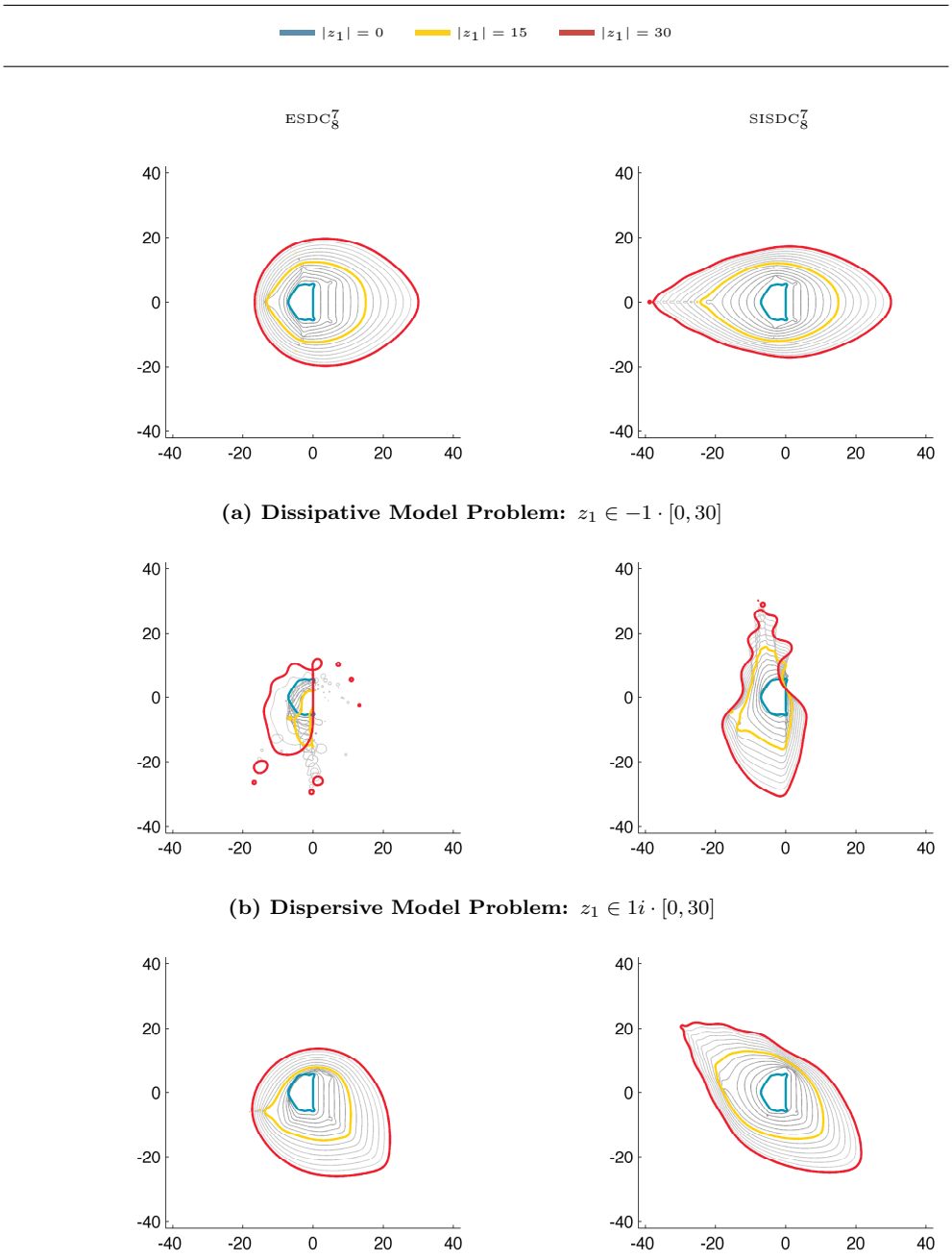
Since we are primarily interested in constructing high-order integrators, we only show stability regions for eighth-order methods and sixteenth-order methods with eight and sixteen nodes, respectively. We also compare the stability regions of ESDC to those of SISDC. In Figures 1 and 2 we respectively plot stability regions for eighth- and sixteenth-order ESDC and SISDC methods with the Chebyshev quadrature nodes

$$t_{n,j} = \frac{h}{2}\left(1 - \cos\left(\frac{\pi(j-1)}{p-1}\right)\right), \qquad j = 1, \ldots, p.$$

For all the z_1 we considered, the stability regions of all methods grow as $|z_1| \to \infty$. Interestingly, the stability regions for ESDC methods temporarily contract for imaginary z_1. Overall, both ESDC and SISDC methods exhibit satisfactory linear stability properties for all ranges of z_1; however, the stability regions for SISDC are marginally larger than those of ESDC.

6. Numerical experiments and discussion. We now present five numerical experiments to compare the efficiency of exponential SDC, SISDC, and two exponential Runge–Kutta integrators. We divide this section into two parts. In the first part we investigate partitioned integrators by solving three semilinear partial differential equations with stiff linear terms, while in the second part we compare unpartitioned integrators on a single partial differential equation with a stiff nonlinearity.

We solve all five partial differential equations using the method of lines on a fixed spatial grid. Moreover, since the problems are not analytically solvable, we compute reference solutions by averaging the outputs of at least two convergent integrators that were run using a small timestep. Finally, the relative error in all our results is always defined as $\|\mathbf{y}_{\text{ref}} - \mathbf{y}_{\text{method}}\|_\infty / \|\mathbf{y}_{\text{ref}}\|_\infty$, where $\mathbf{y}_{\text{ref}}$ is a vector containing the reference solution in physical space and $\mathbf{y}_{\text{method}}$ is a vector containing the output of a method.

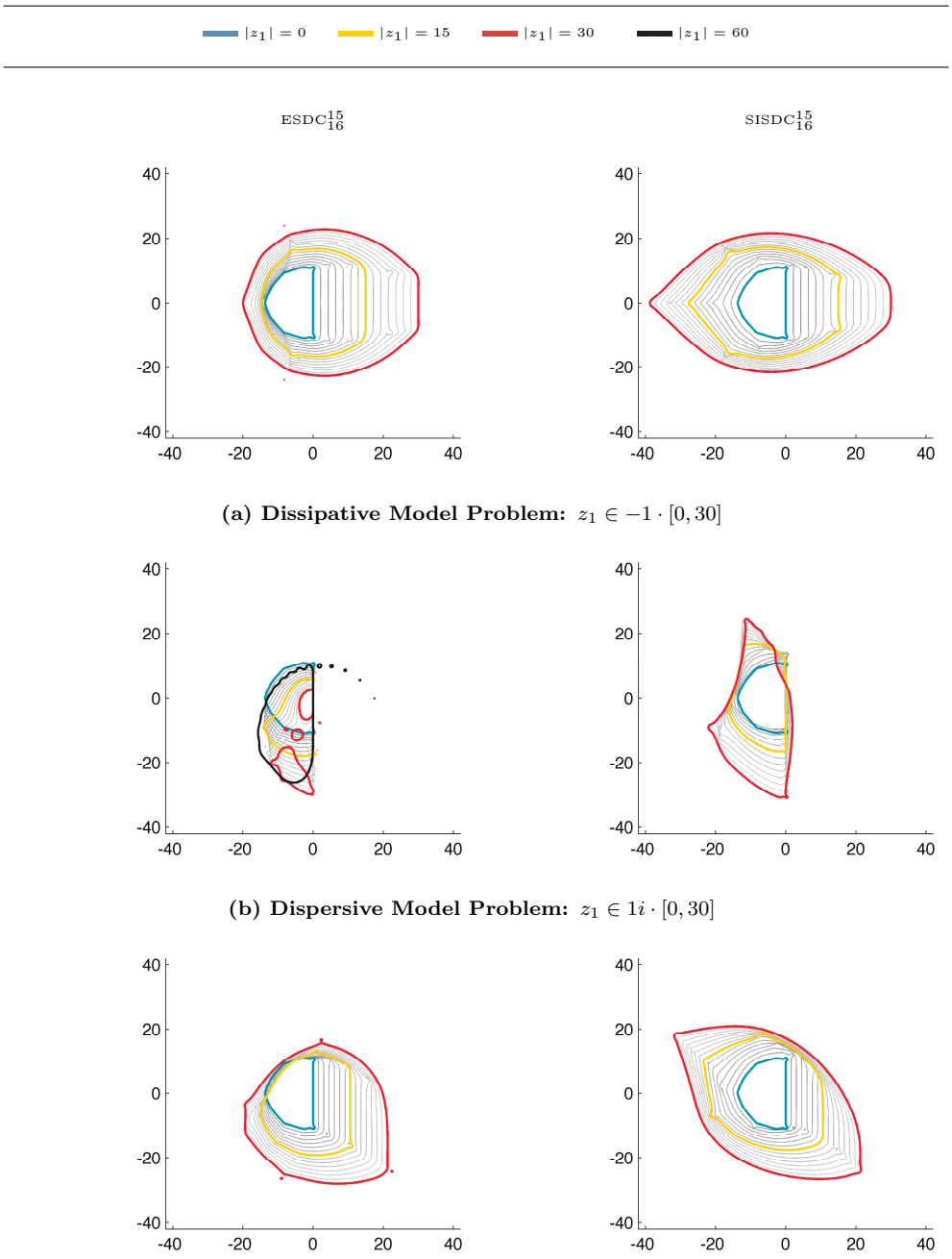


(a) Dissipative Model Problem: $z_1 \in -1 \cdot [0, 30]$

(b) Dispersive Model Problem: $z_1 \in 1i \cdot [0, 30]$

(c) Dissipative & Dispersive Model Problem: $z_1 \in \exp(3\pi i/4) \cdot [0, 30]$

FIG. 1. *Overlaid slices of the stability regions for 8th-order partitioned ESDC and SISDC methods with Chebyshev quadrature nodes. Each slice is formed by fixing z_1 and allowing z_2 to vary. Colored contours correspond to different values of $|z_1|$.*



(a) Dissipative Model Problem: $z_1 \in -1 \cdot [0, 30]$

(b) Dispersive Model Problem: $z_1 \in 1i \cdot [0, 30]$

(c) Dissipative & Dispersive Model Problem: $z_1 \in \exp(3\pi i/4) \cdot [0, 30]$

FIG. 2. *Overlaid slices of the stability regions for* 16*th-order partitioned ESDC and SISDC methods with Chebyshev quadrature nodes. Each slice is formed by fixing* z_1 *and allowing* z_2 *to vary. Colored contours correspond to different values of* $|z_1|$*. We plot an additional black contour for the* ESDC_{16}^{15} *method on the dispersive model problem to show that stability regions eventually grow for imaginary* z_1 *of sufficiently large magnitude.*

6.1. Partitioned numerical experiments. We first investigate the efficiency of partitioned methods for solving the initial value problem

$$\mathbf{y}' = \mathbf{L}\mathbf{y} + N(t, \mathbf{y}).$$

At each timestep, a partitioned exponential integrator requires matrix-vector products between the matrix functions $\varphi_j(h\mathbf{L})$ and linear combinations of the nonlinear derivative component. If the timestep h is constant, then it is possible to precompute and store the matrix functions before the first timestep. For large systems this operation may be costly; however, if $\mathbf{L}$ is diagonal, easily diagonalizable, or banded [27], then this approach can be very efficient.

We compare partitioned integrators by solving three semilinear partial differential equations with periodic boundary conditions and Fourier spectral spatial discretizations. We solve each equation using ESDC and SISDC methods of orders four, eight, sixteen, and thirty-two. We also compare the efficiency of these SDC methods against the partitioned exponential integrator ETDRK4 [12], which has been shown to outperform many other exponential and implicit-explicit methods on problems with diagonal linear operators [18, 30, 41].

For each problem, we present plots of relative error versus function evaluations, relative error versus stepsize, and relative error versus computational time. When determining computational time, we do not include the time for initializing the exponential matrices and the inverse matrices since this computation only has to be completed once before the first timestep and is insignificant when compared to the cost of integrating over many timesteps.

Below we describe the test problems, their initial conditions, and the corresponding numerical parameters. We base the first two numerical experiments on those from [30, 18] so that our results can be compared with a wide range of other implicit-explicit and exponential integrators. All of the following partial differential equations are equipped with periodic boundary conditions.

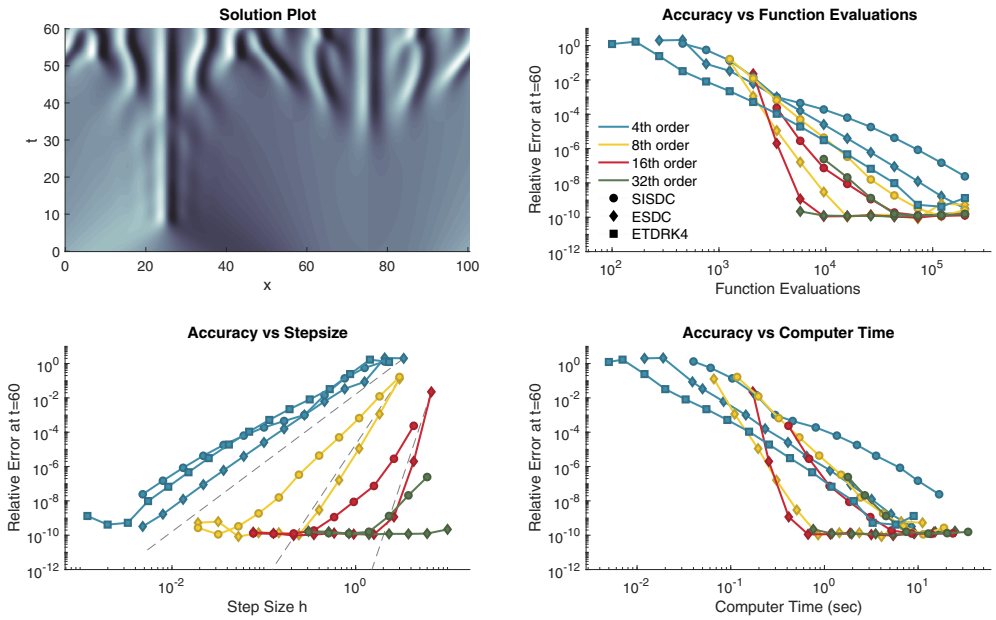
1. The *Kuramoto–Sivashinsky* (KS) equation models reaction-diffusion systems [34]. We consider the KS equation from [30]:

$$\begin{aligned} &\frac{\partial u}{\partial t} = -\frac{\partial^2 u}{\partial x^2} - \frac{\partial^4 u}{\partial x^4} - \frac{1}{2}\frac{\partial}{\partial x}\left(u^2\right), \\ &u(x, t=0) = \cos\left(\tfrac{x}{16}\right)\left(1 + \sin\left(\tfrac{x}{16}\right)\right), \quad x \in [0, 64\pi]. \end{aligned} \tag{6.1}$$

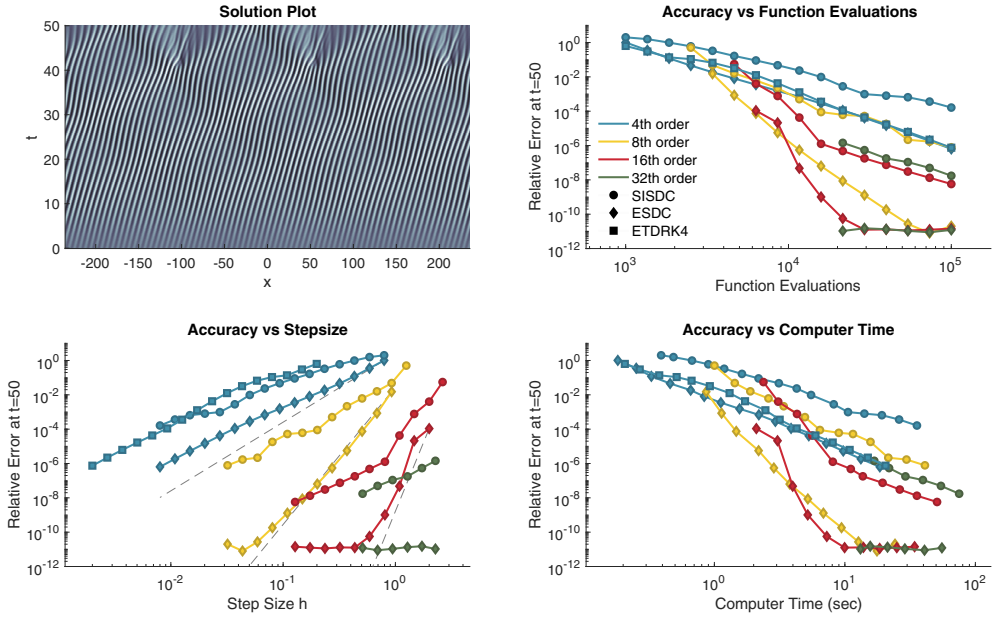
   We numerically integrate (6.1) using a 1024 point Fourier spectral discretization in x and run the simulation out to $t = 60$. The matrix $\mathbf{L}$ includes all the discretized linear derivative terms and has eigenvalues $\lambda(k) = k^2 - k^4$, where k denotes the discrete Fourier wavenumbers on the spatial grid. We present our numerical results in Figure 3.

2. The *Nikolaevskiy* equation was originally developed for studying seismic waves [44] and now serves as a model for pattern formation in a variety of systems [53]. We consider the Nikolaevskiy equation from [18]:

$$\begin{aligned} &\frac{\partial u}{\partial t} = \alpha\frac{\partial^3 u}{\partial x^3} + \beta\frac{\partial^5 u}{\partial x^5} - \frac{\partial^2}{\partial x^2}\left(r - \left(1 + \frac{\partial^2}{\partial x^2}\right)^2\right)u - \frac{1}{2}\frac{\partial}{\partial x}\left(u^2\right), \\ &u(x, t=0) = \sin(x) + \epsilon\sin\left(\tfrac{x}{25}\right), \quad x \in [-75\pi, 75\pi], \end{aligned} \tag{6.2}$$



(a) Kuramoto–Sivashinsky



(b) Nikolaevskiy

FIG. 3. (a) *Results for the Kuramoto–Sivashinsky equation.* (b) *Results for the Nikolaevskiy equation. In both plots, the thin and dashed gray lines in the accuracy versus stepsize plots correspond to fourth, eight, and sixteenth order convergence.*

where $r = 1/4$, $\alpha = 2.1$, $\beta = 0.77$, and $\epsilon = 1/10$. We solve the Nikolaevskiy equation using a 4096 point Fourier spectral discretization in x and run the simulation out to $t = 50$. The matrix $\mathbf{L}$ includes all the discretized linear derivative terms and has eigenvalues $\lambda(k) = k^2(r - (1-k^2)^2) - i\alpha k^3 + i\beta k^5$, where k denotes the discrete Fourier wavenumbers on the spatial grid. We present our numerical results in Figure 3.

3. The *Korteweg–de Vries* (KDV) equation describes weakly nonlinear shallow water waves. In 1965 Kruskal and Zabusky observed that smooth initial conditions could give rise to soliton solutions [61]. As in their original numerical experiment, we consider the KDV equation on a periodic domain

$$\begin{aligned} &\frac{\partial u}{\partial t} = -\left[\delta \frac{\partial^3 u}{\partial x^3} + \frac{1}{2}\frac{\partial}{\partial x}(u^2)\right], \\ &u(x, t=0) = \cos(\pi x), \quad x \in [0, 2], \end{aligned} \tag{6.3}$$

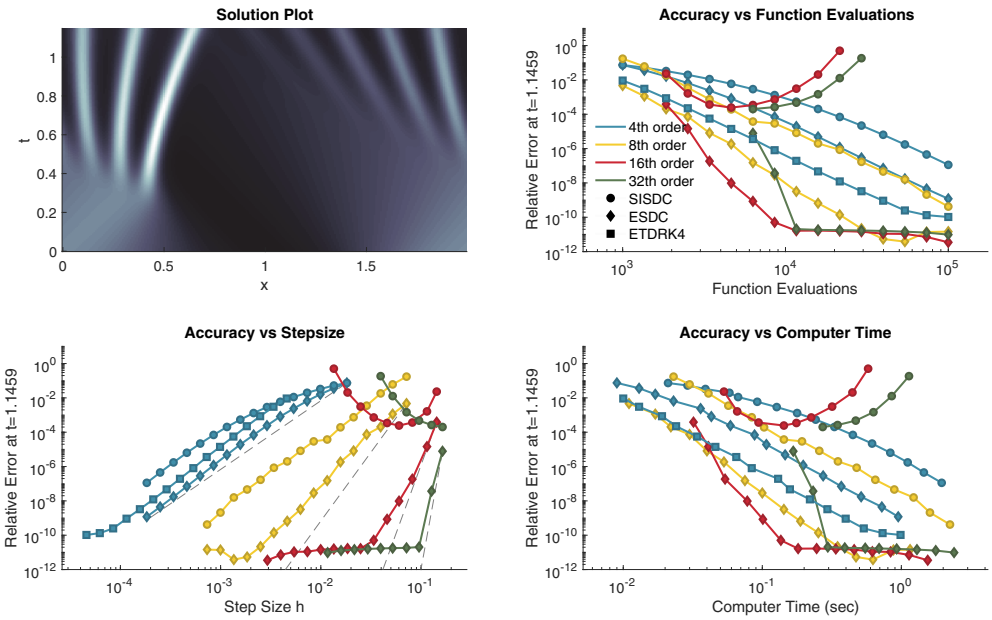
where $\delta = 0.022$. We solve the KDV equation using a 256 point Fourier spectral discretization and run the simulation out to time $t = 3.6/\pi$. The matrix $\mathbf{L}$ includes all the discretized linear derivative terms and has eigenvalues $\lambda(k) = \delta i k^3$, where k denotes the discrete Fourier wavenumbers on the spatial grid. We present our numerical results for the KDV equation in Figure 4.

4. The *quasigeostrophic* (QG) equations model a variety of atmospheric and oceanic phenomena [47]. We consider the barotropic QG equation on a β-plane with linear Ekman drag and hyperviscous diffusion of momentum from [18]:

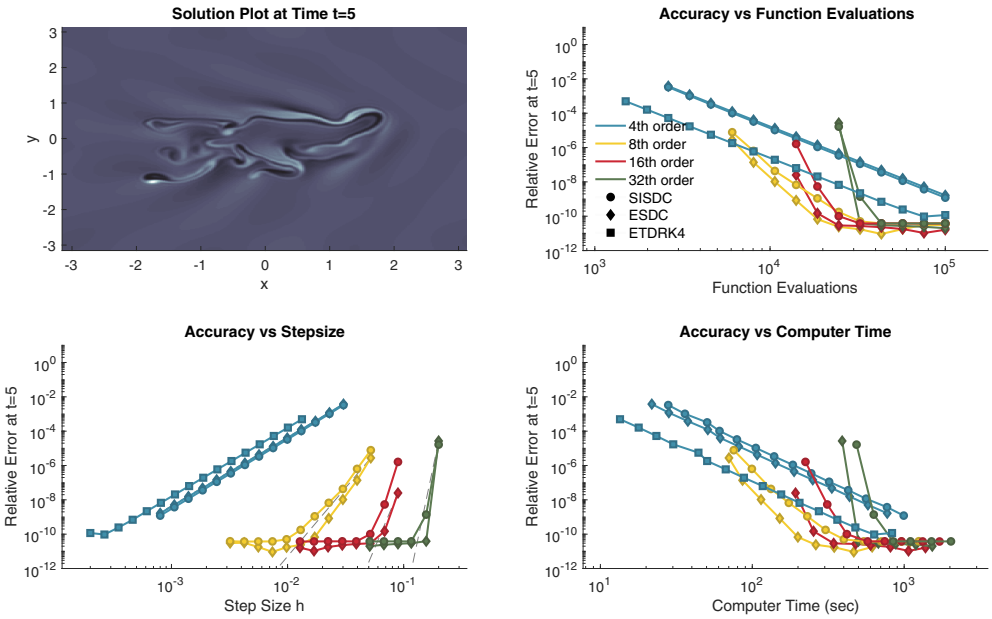
$$\begin{aligned} &\partial_t \nabla^2 \psi = -\left[\beta \partial_x \psi + \epsilon \nabla^2 \psi + \nu \nabla^{10} \psi + \mathbf{u} \cdot \nabla(\nabla^2 \psi)\right], \\ &\psi(x, y, t=0) = \frac{1}{8} \exp\left(-8\left(2y^2 + x^2/2 - \pi/4\right)^2\right), \\ &(x, y) \in [-\pi, \pi], \end{aligned} \tag{6.4}$$

where $\beta = 10$, $\epsilon = 1/100$, $\nu = 10^{-14}$, and $\psi(x, y)$ is the stream function for two-dimensional velocity field $\mathbf{u} = (-\partial_y \psi, \partial_x \psi)$. We run the simulation to time $t = 5$ using a 256×256 point Fourier discretization. The matrix $\mathbf{L}$ includes all the linear derivative terms and has eigenvalues $\lambda(k, l) = \frac{-ik - \epsilon k^2}{k^2 + l^2} - \nu(k^8 + l^8)$, where k and l denote the discrete Fourier wavenumbers on the spatial grid.
We consider a different initial condition than the one presented in [18], since $\nabla^2 \psi(x, y)$ was originally chosen to be discontinuous at the point $(0, 0)$. We also note that (6.4) describes the change in the vorticity $\omega = \nabla^2 \psi$ in terms of the stream function ψ. In order to obtain ψ at each timestep, it is necessary to solve Poisson's equation $\nabla^2 \psi = \omega$ which we do in Fourier space. We present our numerical results in Figure 4.

6.1.1. Fourier discretization. We solve each equation in Fourier space where the operator $\mathbf{L}$ is a diagonal matrix, and we do not apply any form of antialiasing. Computing the exponential or inverse of a diagonal matrix is equivalent to computing the exponential or inverse of multiple scalar terms. Solving in Fourier space therefore enables us to efficiently precompute and store all the φ-functions for exponential integrators and the matrix inverses for semi-implicit integrators. To compute the φ-functions for wavenumbers with magnitude smaller than one, we apply the scalar form of the Cauchy integral formula described in [30], while for large wavenumbers we use the recurrence relation (3).



(a) KDV



(b) Quasigeostrophic

FIG. 4. (a) *Results for the KDV equation.* (b) *Results for the quasigeostrophic equation. In both plots, the thin and dashed gray lines in the accuracy versus stepsize plots correspond to fourth, eight, and sixteenth order convergence.*

Computing matrix-vector products with φ-functions or matrix inverses now amounts to multiplication with a precomputed diagonal matrix. Therefore, the majority of the computational work for both exponential and semi-implicit integrators is due to nonlinear function evaluations. For example, the nonlinear function associated with all one-dimensional equations is

$$N(t, \hat{\mathbf{y}}) = iK\mathcal{F}\left(\left(\mathcal{F}^{-1}\left(\hat{\mathbf{y}}/2\right)\right)^2\right),$$

where $\mathcal{F}$ denotes the forward discrete Fourier transform and K is a diagonal matrix of Fourier wave numbers. Though the fast Fourier transform is extremely efficient, it is still significantly more expensive than multiplication of a diagonal matrix with a vector.

6.2. Discussion of partitioned numerical experiments. The precision diagrams and the function evaluation diagrams for each partitioned numerical experiment are remarkably similar. This demonstrates that nonlinear function evaluations are the primary computational cost for both exponential and semi-implicit partitioned integrators. This occurs because the computational cost of the Fourier transform greatly exceeds that of the diagonal matrix-vector products corresponding to the exponential or implicit terms. In short, the most efficient integrators are those that required the smallest number of function evaluations to achieve a desired level of accuracy.

The experiments can also help us determine whether high-order ESDC methods should be considered in the context of double-precision. At sufficiently low error tolerances, a high-order method will always become more efficient than a lower-order method. Conversely, high-order methods are generally more expensive per timestep and will be less efficient when only a few digits of precision are required. The results demonstrate that high-order ESDC methods can achieve high accuracy using the fewest function evaluations on our test problems. Partitioned ESDC methods of order eight and sixteen were able to reach machine precision ten times faster than fourth-order ETDRK4. However, for large error tolerances, ETDRK4 is still the fastest method, outperforming all ESDC or SISDC methods. Partitioned ESDC methods of order 32 were generally less competitive than those of 8th or 16th order, and should only be considered in situations where higher-precision arithmetic allows for relative errors below 1×10^{-10}.

It is important to note that our claims regarding efficiency and high-order methods are only true for problems that are similar to those tested in this paper. In a more general context with nondiagonalizable linear operators the efficiency will depend on additional factors such as the existence of efficient preconditioners for implicit terms and efficient techniques for computing matrix exponentials.

Next we discuss stability. The partitioned experiments each mimic one of the three model problems studied in section 5: the KS equation has purely dissipative linear derivatives, the KDV equation has purely dispersive linear terms, while the Nikolaevskiy and quasigeostrophic equations have linear terms with both dispersion and dissipation. Moreover, all the equations possess stiff linear terms and nonstiff nonlinearities (see Table 3, where we present the spectral radii of the Jacobians for the linear and nonlinear operators). For all equations with dissipation, both exponential and semi-implicit integrators were convergent. However, for the purely dispersive KDV equation, SISDC methods of order 16 and 32 diverged, while all ESDC methods remained stable. Overall, the numerical experiments suggest that ESDC methods can be used to solve initial value problems with stiff linear terms that possess any degree of dissipation or dispersion.

TABLE 3

Spectral radii of the Jacobians for the linear operator $\mathbf{L}$ *and the nonlinearity operator* $N(t, y)$ *for the partitioned equations. All numbers have been rounded to three digits, and the nonlinear Jacobian has been evaluated at the initial condition.*

	$\rho(\mathbf{L})$	$\rho\left(\frac{\partial N}{\partial y}(y_0)\right)$
Kuramoto–Sivashinsky	6.528×10^4	1.960×10^1
Nikolaevskiy	4.136×10^8	2.460×10^1
KDV	6.502×10^7	3.731×10^2

Finally, we address order reduction. High-order partitioned ESDC methods consistently achieved better accuracy per timestep than SISDC methods and did not exhibit order reduction on any problem that we tested. On the other hand, high-order SIDC methods suffered from significant order reduction. This phenomenon is known to affect SISDC methods [36, 26], and its presence greatly reduced the efficiency of the integrators in our tests. Therefore, our results suggest that high-order partitioned ESDC methods will offer significant improvement over high-order SISDC on any problem where the cost of computing the exponential integrals is similar to that of inverting the implicit component.

6.3. Unpartitioned initial value problems. Next we investigate the efficiency of unpartitioned exponential integrators for solving stiff systems (3.5). Unpartitioned integrators provide improved stability compared to partitioned integrators for problems with stiff nonlinearities. However, at each timestep an unpartitioned exponential integrator requires matrix-vector products between φ-functions of the Jacobian and the remainder term $R(y)$ from (3.6). Since the Jacobian varies in time, it is no longer possible to precompute the matrix functions. Furthermore, unless the dimension of the initial value problem is small, it is often more efficient to compute the exponential matrix-vector products using algorithms that avoid forming the matrix functions explicitly.

For our test problem, we consider the two-dimensional advection-diffusion-reaction (ADR) equation with homogeneous Neumann boundary conditions from [49],

$$\begin{aligned} &u_t = \epsilon\,(u_{xx} + u_{yy}) + \delta\,(u_x + u_y) + \gamma u(u - 1/2)(1 - u), \\ &u(x, t = 0) = 256(xy(1 - x)(1 - y))^2 + 0.3, \\ &x, y \in [0, 1]. \end{aligned} \tag{6.5}$$

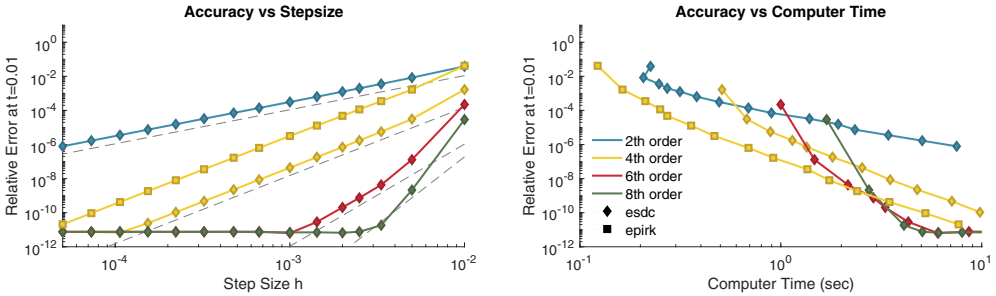
We discretize in space using a 200×200 point grid, apply standard second-order accurate finite differences to approximate the derivatives, and run the simulation out to $t = 1/100$. We consider two different choices for the parameters ϵ, δ, and γ: the first produces an equation that is dominated by linear effects, while the second emphasizes the nonlinearity. The parameters are shown in Table 4, along with the corresponding spectral radii of the Jacobians for the linear and nonlinear terms of the ADR equation.

In our numerical tests, we use the exact Jacobian and apply the Krylov-based KIOPS (Krylov with incomplete orthogonalization procedure solver) algorithm [17] for computing matrix-vector products with φ-functions. We compare ESDC against the stiffly accurate fourth-order EPIRK4s3 method from [38]. EPIRK4s3 has been specially constructed to minimize the number of Krylov projections required by KIOPs at each timestep. Moreover, since it is stiffly accurate, the method satisfies additional stiff order conditions [23] that are obtained by expanding the local error in terms of

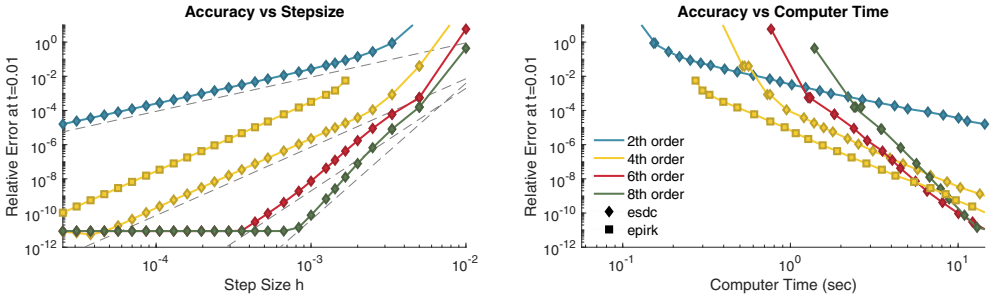
TABLE 4

Parameter choices for the 2D ADR equation and the corresponding spectral radius ρ for the linear and nonlinear terms. The nonlinearity $N(u) = \gamma u(u - 1/2)(1 - u)$ was linearized about the initial condition, and all the spectral radii have been rounded to the nearest integer.

	ϵ	δ	γ	$\rho(\mathbf{L})$	$\rho\left(\frac{\partial N}{\partial y}\right)$
stiff linearity	1/100	-10	100	4288	167
stiff nonlinearity	1/10000	$-1/10$	1000	43	1670



(a) ADR - stiff linearity



(b) ADR - stiff nonlinearity

FIG. 5. (a) *Results for the ADR equation with stiff linear terms.* (b) *Results for the ADR equation with a stiff nonlinearity. The dashed lines in the accuracy diagrams correspond to second, fourth, sixth, and eight order convergence. We do not include ESDC methods of order higher than eight because they are not competitive on this problem.*

bounded functions of the Jacobian.

For each numerical experiment we present plots of relative error versus stepsize, and relative error versus computational time. However, we do not include a graph of relative error versus function evaluations since the cost of evaluating the remainder term is insignificant compared to the cost of computing the matrix-vector products with φ-functions. In fact, the total time spent evaluating the remainder term $R(y)$ only amounted to approximately five percent of the total time spent by KIOPs computing exponential terms.

In Figure 5 we present results for ESDC methods of orders two, four, six, and eight. Though it is possible to use higher-order unpartitioned ESDC schemes, we do not include them in our numerical experiment because they are not competitive with state-of-the-art exponential Runge–Kutta integrators on this problem.

6.3.1. Discussion of unpartitioned numerical experiments. For unpartitioned problems, the exponential matrix functions cannot be precomputed since they depend on the local Jacobian of the right-hand side. Therefore, the majority of the computational cost comes from the evaluation of exponential terms. For ESDC, every correction iteration requires $(p+1)^2$ exponential matrix-vector products for computing the exponential integrals $\hat{I}_{n,j}^{[k]}$ from (4.19).

In general, the work required to compute these terms will depend on the choice of algorithm. When using KIOPs [17], each $\hat{I}_{n,j}^{[k]}$ can be computed using one Krylov projection from $t = t_{n,j}$ to $t = t_{n,j+1}$. In practice this implies that the computational cost of a full correction iteration is approximately equal to that of a single step of EPIRK4s3. Therefore, a pth-order ESDC method requiring p correction iterations is approximately p times more expensive per timestep than EPIRK4s3.

On the test problems both sixth- and eighth-order unpartitioned ESDC were more efficient than EPIRK4s3 at extremely low error tolerances, but the performance gains are less significant than those for partitioned ESDC. Moreover, the high cost of repeated corrections renders unpartitioned ESDC methods of order higher than eight too costly to be competitive. Fourth-order methods were more accurate per timestep but less efficient overall. Finally, our results are also consistent with linear stability analysis and demonstrate that unpartitioned SDC can be used to solve problems with both stiff linear terms or stiff nonlinear terms.

6.4. Implementation. For partitioned schemes, we provide both MATLAB and Fortran implementations of the ESDC, SISDC, and partitioned exponential Runge–Kutta method ETDRK4 [12], along with the code for reproducing our numerical experiments in [6]. All the timing results for partitioned methods were obtained using our Fortran implementation. For unpartitioned schemes, our numerical experiments were implemented and run using the MATLAB version of the EPIC integrator framework [58]. Finally, all the timing results contained in this paper were produced on a computer with a 3.5 Ghz Intel i7-4771 processor.

7. Summary and conclusions. In this work we extend the SDC framework to include exponential integration. We accomplish this by writing an exponential version of the classical SDC error equation and introducing a correction iteration based on exponential Euler. We introduce both a partitioned ESDC integrator for solving semilinear systems and an unpartitioned ESDC method for solving more general stiff systems. As a whole, the ESDC approach enables us to construct stable exponential methods of arbitrary order that can be used to solve initial value problems characterized by both dissipation and dispersion.

In the numerical experiments for partitioned integrators we demonstrate the efficiency of high-order partitioned ESDC for integrating Fourier pseudospectral discretizations of partial differential equations. Eighth- and sixteenth-order ESDC methods were consistently more efficient than the fourth-order ETDRK4 method if high precision is required. This is surprising since ETDRK4 is currently one of the fastest methods for solving partial differential equations on periodic domains with Fourier discretizations where the φ-functions can be precomputed and stored as diagonal matrices [41, 18]. ESDC methods also appear to be more robust to order reduction than SISDC methods and are stable on a wider range of problems, opening the possibility of using extremely high-order integration methods. In short, provided that the cost of computing φ-functions is similar to that of inverting the implicit component, our results suggest that partitioned ESDC should be considered over SISDC.

For more general initial value problems, the unpartitioned ESDC algorithm is not competitive with current state-of-the art unpartitioned exponential integrators unless extremely accurate solutions are needed. Nevertheless our results demonstrate that exponential SDC can be used to solve a variety of problems, including those with stiff nonlinear terms.

The ESDC integrators presented in this work represent an entire class of methods. Concrete schemes can be constructed by selecting an order of accuracy and an algorithm for computing the exponential terms. These two choices will determine the efficiency and computational complexity of the resulting method. Unfortunately there is no simple way to determine the most efficient choice since it will depend on the properties of the underlying ODE, the desired accuracy, and the computer architecture (parallel versus serial) where the numerical simulations are run.

Finally, if more advanced features from the SDC framework were to be incorporated into ESDC, then we suspect there would be further efficiency gains. In particular, in a follow-up paper we plan to extend this work to incorporate time-parallelism. It would also be worth comparing the efficiency of ESDC against eighth- or tenth-order-accurate exponential Runge–Kutta methods that are constructed using order conditions; however, to the best of our knowledge such methods have yet to be derived. Finally, it would be interesting to investigate whether ESDC methods satisfy stiff order conditions for exponential Runge–Kutta integrators [23].

Appendix A. Equations for unpartitioned system. We list the relevant equations for deriving an unpartitioned ESDC method.

- The integral equation for the system (3.5)–(3.6) is

$$\mathbf{y}(t) = \mathbf{y}(a) + \int_a^t e^{\mathbf{J}_n(t-s)} \left[F(\mathbf{y}_n) + \mathbf{J}_n(\mathbf{y}(a) - \mathbf{y}_n) + R(\mathbf{y}(s))\right] ds.$$

  To obtain this equation apply the integrating factor $\exp(-\mathbf{J}_n t)$, and then substitute

$$\exp((t-a)\mathbf{J}_n)\mathbf{y}(a) = \mathbf{y}(a) + (t-a)\varphi_1((t-a)\mathbf{J}_n)\mathbf{J}_n\mathbf{y}(a) = y(a) + \int_a^t e^{\mathbf{J}_n(t-s)}\mathbf{J}_n ds.$$

- The integral equation governing the error of an approximate solution $\tilde{\mathbf{y}}(t)$ is

$$\begin{aligned}\mathbf{y}(t) &= E(a) + \int_a^t e^{\mathbf{J}_n(t-s)} \left[F(s, \mathbf{y} + E(s)) - F(s, \mathbf{y})\right] ds + \mathbf{r}(t,a),\\ \mathbf{r}(t,a) &= \Bigg[\tilde{\mathbf{y}}(a) + (t-a)\varphi_1((t-a)\mathbf{J}_n)\left[F(\mathbf{y}_n) + \mathbf{J}_n\left(\tilde{\mathbf{y}}(a) - \mathbf{y}_n\right)\right]\\ &\qquad + \int_a^t e^{\mathbf{J}_n(t-s)} R(\tilde{\mathbf{y}}(s)) ds\Bigg] - \tilde{\mathbf{y}}(t),\end{aligned}$$

  where $R(\mathbf{y}) = F(\mathbf{y}) - [F(\mathbf{y}_n) + \mathbf{J}_n(\mathbf{y} - \mathbf{y}_n)]$. To obtain this result make the substitution $\mathbf{y}(t) = E(t) + \tilde{\mathbf{y}}(t)$, add and subtract the exponential residual, and then simplify.

- The exponential time-stepping method for solving the error equation is

$$E^{[k]}_{n,j+1} = E^{[k]}_{n,j} + h_{n,j}\varphi_1(h_{n,j}\mathbf{J}_n)H^{[k]}_{n,j+\gamma} + \tilde{R}^{[k]}_{n,j}, \qquad j = 0, \ldots, p-1, \tag{A.1}$$

  where the substeps $h_{n,j} = t_{n,j+1} - t_{n,j}$, the values $H^{[k]}_{n,j+\gamma}$ are

$$H^{[k]}_{n,j+\gamma} = F\left(t_{n,j+\gamma}, Y^{[k]}_{n,j+\gamma} + E^{[k]}_{n,j+\gamma}\right) - F\left(t_{n,j+\gamma}, Y^{[k]}_{n,j+\gamma}\right),$$

and the approximate exponential residual $\tilde{R}_{n,j}^{[k]}$ is

$$\tilde{R}_{n,j}^{[k]} = \left[Y_{n,j}^{[k]} + h_{n,j}\varphi_1(h_{n,j}\mathbf{J}_n) \left[\mathbf{J}_n \left(Y_{n,j}^{[k]} - y_n \right) + F(y_n) \right] + \int_{t_{n,j}}^{t_{n,j+1}} e^{\mathbf{J}_n(t_{n,j+1}-s)} Q^{[k]}(s) ds \right] - Y_{n,j+1}^{[k]},$$

where $Q^{[k]}(s)$ is the Lagrange polynomial interpolating the values

$$\left(t_{n,j},\ R\left(Y_{n,j}^{[k]} \right) \right) \quad \text{for} \quad j = 1, \ldots, p.$$

- A direct update formula can be obtained by rewriting $H_{n,j+\gamma}^{[k]}$ as

$$H_{n,j+\gamma}^{[k]} = F\left(t_{n,j+\gamma}, Y_{n,j+\gamma}^{[k+1]} \right) - F\left(t_{n,j+\gamma}, Y_{n,j+\gamma}^{[k]} \right).$$

  Then by substituting the expression (A.1) into the update formula (2.2) and simplifying, we obtain the direct update formulas (4.12) and (4.13).

Acknowledgments. I would like to thank Randall J. LeVeque for our many useful discussions over the course of this project. I would also like to thank Mayya Tokman for sharing her many insights about exponential integrators and for her comments on this work.

REFERENCES

[1] U. M. Ascher, S. J. Ruuth, and B. T. R. Wetton, *Implicit-explicit methods for time-dependent partial differential equations*, SIAM J. Numer. Anal., 32 (1995), pp. 797–823, https://doi.org/10.1137/0732037.

[2] H. A. Ashi, L. J. Cummings, and P. C. Matthews, *Comparison of methods for evaluating functions of a matrix exponential*, Appl. Numer. Math., 59 (2009), pp. 468–486.

[3] H. Berland, B. Skaflestad, and W. M. Wright, *EXPINT—a MATLAB package for exponential integrators*, ACM Trans. Math. Softw., 33 (2007), p. 4.

[4] G. Beylkin, J. M. Keiser, and L. Vozovoi, *A new class of time discretization schemes for the solution of nonlinear PDEs*, J. Comput. Phys., 147 (1998), pp. 362–387.

[5] J. P. Boyd, *Chebyshev and Fourier Spectral Methods*, Dover, New York, 2013.

[6] T. Buvoli, *Codebase for ETD Spectral Deferred Correction Methods*, 2014, https://doi.org/10.5281/zenodo.12619.

[7] A. Cardone, Z. Jackiewicz, A. Sandu, and H. Zhang, *Construction of highly stable implicit-explicit general linear methods*, in Discrete Contin. Dynam. Syst. 2015, 10th AIMS Conference on Dynamical Systems, Differential Equations and Applications, Supplement, 2015, pp. 185–194.

[8] M. F. Causley and D. C. Seal, *On the Convergence of Spectral Deferred Correction Methods*, preprint, https://arxiv.org/abs/1706.06245, 2017.

[9] A. Christlieb, M. Morton, B. Ong, and J. Qiu, *Semi-implicit integral deferred correction constructed with additive Runge–Kutta methods*, Commun. Math. Sci., 9 (2011), pp. 879–902.

[10] A. Christlieb, B. Ong, and J. Qiu, *Integral deferred correction methods constructed with high order Runge-Kutta integrators*, Math. Comp., 79 (2010), pp. 761–783.

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

[12] S. M. Cox and P. C. Matthews, *Exponential time differencing for stiff systems*, J. Comput. Phys., 176 (2002), pp. 430–455.

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

[14] M. Emmett and M. Minion, *Toward an efficient parallel in time method for partial differential equations*, Commun. Appl. Math. Comput. Sci., 7 (2012), pp. 105–132.

[15] B. Fornberg, *Generation of finite difference formulas on arbitrarily spaced grids*, Math. Comp., 51 (1988), pp. 699–706.
[16] B. Fornberg, *A Practical Guide to Pseudospectral Methods*, Cambridge Monogr. Appl. Comput. Math. 1, Cambridge University Press, Cambridge, UK, 1996.
[17] S. Gaudreault, G. Rainwater, and M. Tokman, *KIOPS: A fast adaptive Krylov subspace solver for exponential integrators*, J. Comput. Phys., 372 (2018), pp. 236–255.
[18] I. Grooms and K. Julien, *Linearly implicit methods for nonlinear PDEs with linear dispersion and dissipation*, J. Comput. Phys., 230 (2011), pp. 3630–3650.
[19] A. C. Hansen and J. Strain, *On the order of deferred correction*, Appl. Numer. Math., 61 (2011), pp. 961–973.
[20] T. Haut, T. Babb, P. Martinsson, and B. Wingate, *A high-order time-parallel scheme for solving wave propagation problems via the direct construction of an approximate time-evolution operator*, IMA J. Numer. Anal., 36 (2015), pp. 688–716.
[21] M. Hochbruck and C. Lubich, *On Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal., 34 (1997), pp. 1911–1925, https://doi.org/10.1137/S0036142995280572.
[22] M. Hochbruck, C. Lubich, and H. Selhofer, *Exponential integrators for large systems of differential equations*, SIAM J. Sci. Comput., 19 (1998), pp. 1552–1574, https://doi.org/10.1137/S1064827595295337.
[23] M. Hochbruck and A. Ostermann, *Explicit exponential Runge–Kutta methods for semilinear parabolic problems*, SIAM J. Numer. Anal., 43 (2005), pp. 1069–1090, https://doi.org/10.1137/040611434.
[24] M. Hochbruck and A. Ostermann, *Exponential Runge–Kutta methods for parabolic problems*, Applied Numerical Mathematics, 53 (2005), pp. 323–339.
[25] M. Hochbruck and A. Ostermann, *Exponential integrators*, Acta Numerica, 19 (2010), pp. 209–286.
[26] J. Huang, J. Jia, and M. Minion, *Accelerating the convergence of spectral deferred correction methods*, Journal of Computational Physics, 214 (2006), pp. 633–656.
[27] A. Iserles, *How large is the exponential of a banded matrix*, New Zealand J. Math, 29 (2000), p. 56.
[28] G. Izzo and Z. Jackiewicz, *Highly stable implicit–explicit Runge–Kutta methods*, Applied Numerical Mathematics, 113 (2017), pp. 71–92.
[29] Z. Jackiewicz and H. Mittelmann, *Construction of IMEX DIMSIMs of high order and stage order*, Applied Numerical Mathematics, 121 (2017), pp. 234–248.
[30] A.-K. Kassam and L. N. Trefethen, *Fourth-order time-stepping for stiff PDEs*, SIAM J. Sci. Comput., 26 (2005), pp. 1214–1233, https://doi.org/10.1137/S1064827502410633.
[31] S. Koikari, *Rooted tree analysis of Runge–Kutta methods with exact treatment of linear terms*, J. Comput. Appl. Math., 177 (2005), pp. 427–453.
[32] S. Koikari, *An error analysis of the modified scaling and squaring method*, Comput. Math. Appl., 53 (2007), pp. 1293–1305.
[33] S. Krogstad, *Generalized integrating factor methods for stiff PDEs*, J. Comput. Phys., 203 (2005), pp. 72–88.
[34] Y. Kuramoto and T. Tsuzuki, *Persistent propagation of concentration waves in dissipative media far from thermal equilibrium*, Progr. Theoret. Phys., 55 (1976), pp. 356–369.
[35] A. Layton and M. Minion, *Implications of the choice of predictors for semi-implicit Picard integral deferred correction methods*, Commun. Appl. Math. Comput. Sci., 2 (2007), pp. 1–34.
[36] A. T. Layton and M. Minion, *Implications of the choice of quadrature nodes for Picard integral deferred corrections methods for ordinary differential equations*, BIT, 45 (2005), pp. 341–373.
[37] J. Loffeld and M. Tokman, *Comparative performance of exponential, implicit, and explicit integrators for stiff systems of ODEs*, J. Comput. Appl. Math., 241 (2013), pp. 45–67.
[38] D. L. Michels, V. T. Luan, and M. Tokman, *A stiffly accurate integrator for elastodynamic problems*, ACM Trans. Graphics, 36 (2017), p. 116.
[39] B. V. Minchev and W. M. Wright, *A Review of Exponential Integrators for First Order Semi-Linear Problems*, preprint, NTNU Trondheim, Trondheim, Norway, 2005.
[40] M. Minion, *Semi-implicit spectral deferred correction methods for ordinary differential equations*, Commun. Math. Sci., 1 (2003), pp. 471–500.
[41] H. Montanelli and N. Bootland, *Solving Periodic Semilinear Stiff PDEs in 1D, 2D and 3D with Exponential Integrators*, preprint, https://arxiv.org/abs/1604.08900, 2016.
[42] J. Niesen and W. M. Wright, *A Krylov Subspace Method for Option Pricing*, preprint, 2011, https://doi.org/10.2139/ssrn.1799124.

[43] J. Niesen and W. M. Wright, *Algorithm* 919: *A Krylov subspace algorithm for evaluating the φ-functions appearing in exponential integrators*, ACM Trans. Math. Softw., 38 (2012), 22.
[44] V. N. Nikolaevskiy, *Dynamics of viscoelastic media with internal oscillators*, in Recent Advances in Engineering Science, Lecture Notes in Engrg. 39, Springer-Verlag, Berlin, 1989, pp. 210–221.
[45] S. P. Norsett, *An A-stable modification of the Adams-Bashforth methods*, in Conference on the Numerical Solution of Differential Equations, Springer, New York, 1969, pp. 214–219.
[46] A. Ostermann, M. Thalhammer, and W. M. Wright, *A class of explicit exponential general linear methods*, BIT, 46 (2006), pp. 409–431.
[47] J. Pedlosky, *Geophysical Fluid Dynamics*, Springer-Verlag, New York, 1987.
[48] G. Rainwater and M. Tokman, *A new class of split exponential propagation iterative methods of Runge–Kutta type (sEPIRK) for semilinear systems of ODEs*, J. Comput. Phys., 269 (2014), pp. 40–60.
[49] G. Rainwater and M. Tokman, *A new approach to constructing efficient stiffly accurate EPIRK methods*, J. Comput. Phys., 323 (2016), pp. 283–309.
[50] A. Sandu and M. Günther, *A generalized-structure approach to additive Runge–Kutta methods*, SIAM J. Numer. Anal., 53 (2015), pp. 17–42, https://doi.org/10.1137/130943224.
[51] M. Schreiber and R. Loft, *A parallel time integrator for solving the linearized shallow water equations on the rotating sphere*, Numer. Linear Algebra Appl., 26 (2019), e2220.
[52] M. Schreiber, N. Schaeffer, and R. Loft, *Exponential integrators with parallel-in-time rational approximations for the shallow-water equations on the rotating sphere*, Parallel Comput., 85 (2019), pp. 56–65.
[53] E. Simbawa, P. C. Matthews, and S. M. Cox, *Nikolaevskiy equation with dispersion*, Phys. Rev. E, 81 (2010), 036220.
[54] R. Speck, *Parallelizing spectral deferred corrections across the method*, Comput. Vis. Sci., 19 (2018), pp. 75–83.
[55] R. Speck, D. Ruprecht, M. Emmett, M. Minion, M. Bolten, and R. Krause, *A multi-level spectral deferred correction method*, BIT, 55 (2015), pp. 843–867.
[56] M. Tokman, *Efficient integration of large stiff systems of ODEs with exponential propagation iterative (EPI) methods*, J. Comput. Phys., 213 (2006), pp. 748–776.
[57] M. Tokman, *A new class of exponential propagation iterative methods of Runge–Kutta type (EPIRK)*, J. Comput. Phys., 230 (2011), pp. 8762–8778.
[58] M. Tokman, *EPIC (Exponential Propagation Integrators Collection)*, 2016, https://faculty.ucmerced.edu/mtokman/#software.
[59] L. N. Trefethen, *Spectral Methods in MATLAB*, Software Environ. Tools 10, SIAM, Philadelphia, 2000, https://doi.org/10.1137/1.9780898719598.
[60] L. N. Trefethen, *Evaluating matrix functions for exponential integrators via Carathéodory–Fejér approximation and contour integrals*, Electron. Trans. Numer. Anal., 29 (2007), pp. 1–18.
[61] N. J. Zabusky and M. D. Kruskal, *Interaction of solitons in a collisionless plasma and the recurrence of initial states*, Phys. Rev. Lett., 15 (1965), pp. 240–243.
[62] S. Zhu and S. Weng, *A parallel spectral deferred correction method for first-order evolution problems*, BIT, 58 (2018), pp. 807–834.