

## EXPONENTIAL POLYNOMIAL BLOCK METHODS\*

TOMMASO BUVOLI†

**Abstract.** In this paper we extend the polynomial time integration framework to include exponential integration for both partitioned and unpartitioned initial value problems. We then demonstrate the utility of the exponential polynomial framework by constructing a new class of parallel exponential polynomial block methods (EPBMs) based on the Legendre points. These new integrators can be constructed at arbitrary orders of accuracy and have improved stability compared to existing exponential linear multistep methods. Moreover, if the ODE right-hand-side evaluations can be parallelized efficiently, then high-order EPBMs are significantly more efficient at obtaining highly accurate solutions than exponential linear multistep methods and exponential spectral deferred correction methods of equivalent order.

**Key words.** time-integration, polynomial interpolation, general linear methods, parallelism, high-order, exponential integrators

**AMS subject classifications.** 65L04, 65L05, 65L06, 65L07

**DOI.** 10.1137/20M1321346

Polynomial time integrators [4, 7] are a class of parametrized methods for solving first-order systems of ordinary differential equations (ODEs). The integrators are based on a new framework that combines ideas from complex analysis, approximation theory, and general linear methods. The framework encompasses all general linear methods that compute their outputs and stages using interpolating polynomials in the complex time plane. The use of polynomials enables one to trivially satisfy order conditions and easily construct a range of implicit or explicit integrators with properties such as parallelism and high-order of accuracy.

In order to extend the utility of the polynomial framework, we generalize it to include exponential integration. Exponential integrators are a general class of methods that incorporate exponential functions to provide increased stability and accuracy for solving stiff systems [24]. Continuing efforts to construct and analyze exponential integrators have already produced a wide range of methods [2, 12, 27, 29, 22, 23, 28, 24, 38] that can provide improved efficiency compared to fully implicit and semi-implicit integrators [19, 27, 31, 35].

Incorporating parallelism into exponential integrators remains an open question. Typically, parallelism is applied within an exponential scheme to speed up the estimation of exponential matrix functions and their products with vectors; examples include parallel Krylov projections [32] and parallel rational approximations of exponential functions [45, 20, 42, 43]. To the best of our knowledge there has only been limited research in developing exponential integrators that naturally incorporate parallelism in their stages and outputs. Exponential propagation iterative methods of Runge–Kutta (EPIRK) type methods with independent stages [40] and parallel exponential Rosenbrock methods [33] constitute exceptions to this assessment. However, both

---

\*Submitted to the journal's Methods and Algorithms for Scientific Computing section March 30, 2020; accepted for publication (in revised form) February 1, 2021; published electronically May 13, 2021.

<https://doi.org/10.1137/20M1321346>

**Funding:** This work was supported by the National Science Foundation through grants DMS-1216732 and DMS-2012875.

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

of these approaches are limited since they require a restricted integrator formulation that only permits a limited number of stages to be parallelized. Furthermore, it is difficult to derive arbitrary-order parallel schemes of this type.

The aim of this work is therefore twofold. First, we extend the polynomial framework to include exponential integration and introduce a general formulation that encompasses all families of exponential polynomial integrators. Second, we demonstrate the utility of the framework by presenting several method constructions for deriving both serial and parallel exponential polynomial block methods (EPBMs) with any order of accuracy. Unlike existing exponential integrators, the new parallel method constructions enable simultaneous computation of all their output values.

This paper is organized as follows. In sections 1 and 2 we provide a brief introduction to polynomial methods and exponential integrators. In section 3 we extend the polynomial framework to include exponential integration and introduce polynomial block methods (PBMs). In section 4 we propose several general strategies for constructing parallel or serial PBMs of any order. We also introduce a new class of PBMs based on the Legendre points. Next, in section 5, we analyze and compare the stability regions of the new methods to existing exponential Adams–Bashforth (EAB) and exponential Runge–Kutta methods (ERK). Finally, in section 6, we perform numerical experiments comparing EPBMs to a variety of other high-order exponential integrators.

**1. Polynomial time integrators.** Polynomial time integrators [4, 7] are a general class of parametrized methods constructed using interpolating polynomials in the complex time plane. The polynomial framework is based on the *ODE polynomial* and the *ODE dataset*. The former describes a range of polynomial-based approximations, and the latter contains the data values for constructing these interpolants. The primary distinguishing feature of a polynomial method is that each of its stages and outputs is computed by evaluating an ODE polynomial. Broadly speaking, the polynomial framework encapsulates the subset of all general linear methods with coefficients that correspond to those of ODE polynomials.

In the subsections that follow, we briefly review the ODE dataset, the ODE polynomial, and the notation used to describe polynomial time integrators.

**1.1. The ODE dataset.** The ODE dataset is an ordered set containing all possible data values for constructing the interpolating polynomials that are used to compute a method's outputs. At the  $n$ th timestep an ODE dataset contains a method's inputs, outputs, and stage values along with their derivatives and their temporal nodes. The data is represented in the local coordinates  $\tau$  where the global time is

$$(1.1) \quad t = r\tau + t_n,$$

and  $r$  is a scaling factor. An ODE dataset of size  $w$  is represented with the notation

$$(1.2) \quad D(r, t_n) = \{(\tau_j, y_j, rf_j)\}_{j=1}^w,$$

where  $y_j \approx y(t(\tau_j))$ , and  $f_j = F(t(\tau_j), y_j)$ .

**1.2. The ODE polynomial.** An ODE polynomial can be used to represent a wide variety of approximations for the Taylor series of a differential equation solution. In its most general form, an ODE polynomial of degree  $g$  with expansion point  $b$  is

$$(1.3) \quad p(\tau; b) = \sum_{j=0}^g \frac{a_j(b)(\tau - b)^j}{j!},$$

where each approximate derivative  $a_j(b)$  is computed by differentiating interpolating polynomials constructed from the values in an ODE dataset  $D(r, t_n)$ . For details regarding the most general formulations for the approximate derivatives  $a_j(b)$  we refer the reader to [4, 7].

*Adams ODE polynomials* are one special family of ODE polynomials that are related to this work. Every Adams ODE polynomial can be written as

$$(1.4) \quad p(\tau; b) = L_y(b) + \int_b^\tau L_F(\xi) d\xi,$$

where  $L_y(\tau)$  and  $L_F(\tau)$  are Lagrange interpolating polynomials that respectively approximate  $y(t(\tau))$  and its local derivative  $ry'(t(\tau))$ . These polynomials can be used to construct classical Adams–Bashforth methods and their generalized block counterparts from [7].

**1.3. Parameters and notation for polynomial methods.** During the timestep from  $t_n$  to  $t_{n+1} = t_n + h$ , a polynomial method accepts inputs  $y_j^{[n]}$  and produces outputs  $y_j^{[n+1]}$ , where  $j = 1, \dots, q$ . Every input and output of a polynomial method approximates the differential equation solution  $y(t)$  at a particular time. In local coordinates we represent these time points using the node set  $\{z_j\}_{j=1}^q$  such that

$$(1.5) \quad y_j^{[n]} \approx y(t_n + rz_j) \quad \text{and} \quad y_j^{[n+1]} \approx y(t_n + rz_j + h).$$

The input nodes of a polynomial method scale with respect to the *node radius*  $r$ , which is independent of the stepsize  $h = r\alpha$ . The parameter  $\alpha = h/r$  is known as the *extrapolation factor* and its value represents the number of times the node radius divides the timestep. For large  $\alpha$  the distances between a method's input nodes will be small relative to the stepsize, while the opposite is true if  $\alpha$  is small. In general, polynomial methods are described in terms of  $r$  and  $\alpha$  rather than  $r$  and  $h$  since these are natural variables for working with polynomials in local coordinates.

The polynomial framework encapsulates the subset of all general linear methods [3] that are constructed using ODE polynomials. The most general formulation for a polynomial method with  $s$  stages and  $q$  outputs is

$$(1.6) \quad \begin{aligned} Y_i &= p_j(c_j(\alpha), b_j(\alpha)), & j &= 1, \dots, s, \\ y_j^{[n+1]} &= p_{j+s}(z_j + \alpha; b_{j+s}(\alpha)), & j &= 1, \dots, q, \end{aligned}$$

where  $Y_i$  denote stage values,  $c_j(\alpha)$  are stage nodes, and  $p_j(\tau; b)$  are all ODE polynomials constructed from an ODE dataset containing the methods inputs, outputs, and stage values. From this general formulation it is possible to derive many different families of polynomial integrators, one example being PBMs from [7]. In this work we will generalize (1.6) for exponential integrators, and then explore the special subcase of exponential PBMs.

**1.3.1. Propagators and iterators.** All polynomial integrators are parametrized in terms of the extrapolation factor  $\alpha$  that scales the stepsize relative to the node radius  $r$ . A direct consequence of having two independent variables that control stepsize is that all polynomial integrators naturally fall into one of two classes. If  $\alpha > 0$ , then the method is a *propagator* that advances the solution forward in time. If  $\alpha$  is zero, then the stepsize  $h = r\alpha = 0$ , and the method reduces to an *iterator* that recomputes the solution at the current timestep.

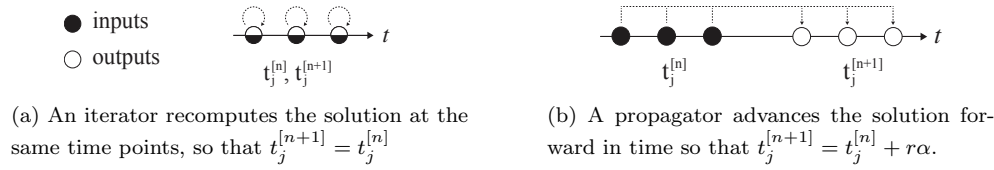


FIG. 1.1. *Iterator and propagator methods. Input times  $t_j^{[n]}$  and output times  $t_j^{[n+1]}$  are shown on the real  $t$  line for a method where the nodes  $z_j$  are three equispaced points. Iterators can be useful for correcting or updating an approximate solution, while propagators are traditional timestepping schemes.*

In Figure 1.1 we provide a visualization for these two types of methods. Iterators can serve several purposes, such as computing initial conditions, computing continuous output, or improving the accuracy of a candidate solution. The notion of using an iterator to improve the accuracy of a solution shares many commonalities with predictor corrector approaches for block methods [44] and spectral deferred correction methods [13, 11]. A detailed discussion of propagators and iterators is outside the scope of this paper. In this work, we will provide a first look at how iterators and propagators can be combined to create composite methods.

**2. Exponential integrators.** Exponential integrators are a class of numerical methods for solving stiff initial value problems [24]. They are derived from the prototypical semilinear equation

$$(2.1) \quad \mathbf{y}' = \mathbf{L}\mathbf{y} + N(t, \mathbf{y}), \quad \mathbf{y}(t_0) = \mathbf{y}_0,$$

where the autonomous linear operator  $\mathbf{L}$  and the nonlinear operator  $N(t, \mathbf{y})$  are chosen so that they approximate the original ODE problem. In practice, the linear operator  $\mathbf{L}$  is selected so that it captures the majority of the stiffness of the right-hand side.

There are two well-known classes of exponential integrators, namely *partitioned* exponential integrators [12, 2, 39] and *unpartitioned* integrators [40, 47, 48]. In the following two subsections, we briefly introduce each one.

**2.1. Partitioned exponential integrators.** If an initial value problem can be naturally partitioned into a semilinear equation of the form (2.1), then it can be solved using a partitioned exponential integrator. The exact solution to (2.1) is obtained by applying the discrete variation of constants formula to obtain the integral equation

$$(2.2) \quad \mathbf{y}(t) = e^{(t-t_n)\mathbf{L}}\mathbf{y}(t_n) + \int_{t_n}^t e^{(t-s)\mathbf{L}}N(s, \mathbf{y}(s))ds.$$

Partitioned exponential integrators treat the linear term  $\mathbf{L}$  exactly while approximating the nonlinearity  $N(s, \mathbf{y}(s))$  with a polynomial. For example, the second-order ERK, ETD RK2, from [12],

$$(2.3) \quad \begin{aligned} Y_1 &= e^{h\mathbf{L}}y_n + \int_0^1 e^{(1-s)h\mathbf{L}}N(t_n, y_n)ds, \\ y_{n+1} &= e^{h\mathbf{L}}y_n + \int_0^1 e^{(1-s)h\mathbf{L}}[N(t_n, y_n) + s(N(t_{n+1}, Y_1) - N(t_n, y_n))]ds, \end{aligned}$$



utilizes polynomials that are constructed using stage values. Similarly, the second-order EAB method [2, 12],

$$(2.4) \quad y_{n+1} = e^{h\mathbf{L}} y_n + h \int_0^1 e^{(1-s)h\mathbf{L}} [N(t_n, y_n) + s(N(t_n, y_n) - N(t_{n-1}, y_{n-1}))] ds,$$

utilizes a Lagrange interpolating polynomial constructed using previous solution values. Since exponential integrators involve weighted integrals of polynomials and exponentials, they are frequently expressed in terms of the  $\varphi$ -functions,

$$(2.5) \quad \varphi_n(z) = \begin{cases} e^z, & n = 0, \\ \frac{1}{(n-1)!} \int_0^1 e^{z(1-s)} s^{n-1} ds, & n > 0. \end{cases}$$

For example, using this notation the Adams–Bashforth method (2.4) is

$$y_{n+1} = \varphi_0(h\mathbf{L})y_n + \varphi_1(h\mathbf{L})hN(t_n, y_n) + \varphi_2(h\mathbf{L})h(N(t_n, y_n) - N(t_{n-1}, y_{n-1})).$$

**2.2. Unpartitioned exponential integrators.** Unpartitioned exponential integrators [40, 47, 48] can be used to solve the more general initial value problem

$$(2.6) \quad \mathbf{y}' = F(t, \mathbf{y}), \quad \mathbf{y}(t_0) = \mathbf{y}_0.$$

The key intuition is that one may obtain a localized semilinear problem of the form (2.1) at  $t = t_n$  by rewriting the system in its autonomous form  $\mathbf{y}' = F(\mathbf{y})$  and then rewriting  $F(\mathbf{y})$  as

$$(2.7) \quad \begin{aligned} F(\mathbf{y}) &= F(\mathbf{y}_n) + \mathbf{J}_n (\mathbf{y} - \mathbf{y}_n) + R(\mathbf{y}), \\ R(\mathbf{y}) &= F(\mathbf{y}) - [F(\mathbf{y}_n) + \mathbf{J}_n (\mathbf{y} - \mathbf{y}_n)], \end{aligned}$$

where  $\mathbf{y}_n = \mathbf{y}(t_n)$  and  $\mathbf{J}_n = \frac{\partial F}{\partial \mathbf{y}}(\mathbf{y}(t_n))$  is the Jacobian of  $\mathbf{y}(t)$  at  $t = t_n$ . The linear operator  $\mathbf{J}_n$  takes the place of  $\mathbf{L}$  in (2.1) and the remaining terms form the nonlinearity. Given the initial condition  $\mathbf{y}(t_n) = \mathbf{y}_n$ , the solution of (2.6) is

$$(2.8) \quad \mathbf{y}(t) = e^{(t-t_n)\mathbf{J}_n} \mathbf{y}_n + \int_{t_n}^t e^{(t-s)\mathbf{J}_n} [F(\mathbf{y}_n) - \mathbf{J}_n \mathbf{y}_n + R(\mathbf{y})] ds.$$

Depending on the approximation that is chosen for the remainder term  $R(\mathbf{y})$  one arrives at different families of unpartitioned exponential integrators [40, 47, 48].

**3. Exponential polynomial methods.** In this section we introduce the exponential equivalent of the generalized classical polynomial method (1.6), and then describe the family of EPBMs. To accomplish this, we first extend the definitions of the ODE dataset and the ODE polynomial. To be consistent with existing exponential integrators, the exponential extension of the ODE polynomial approximates the integral equation (2.2) by replacing the nonlinearity with a polynomial approximation.

**3.1. Exponential ODE dataset and ODE polynomial.** We first discuss the extension of the ODE dataset and ODE polynomial for the partitioned initial value problem (2.1). We can trivially extend the classical ODE dataset by adding the nonlinear derivative component  $N(t, \mathbf{y}(t))$  to each data element. Since the linear term is treated exactly by an exponential integrator, there is no need to include it.

DEFINITION 3.1 (partitioned exponential ODE dataset). *An exponential ODE dataset  $D_N(r, s)$  of size  $w$  is an ordered set of tuples*

$$(3.1) \quad D_N(r, s) = \{(\tau_j, \mathbf{y}_j, r\mathbf{f}_j, r\mathbf{N}_j)\}_{j=1}^w,$$

where  $t(\tau) = r\tau + s$ ,  $\mathbf{y}_j \approx \mathbf{y}(t(\tau_j))$ ,  $\mathbf{f}_j = \mathbf{L}\mathbf{y}_j + N(t(\tau_j), \mathbf{y}_j)$ , and  $\mathbf{N}_j = N(t(\tau_j), \mathbf{y}_j)$ .

To arrive at a generalization of the ODE polynomial (1.3), we first rewrite (2.1) in local coordinates  $\tau$ , where  $t(\tau) = r\tau + t_n$ , and then assume that the initial condition is provided at  $\tau = b$ . The corresponding integral equation for the solution is

$$(3.2) \quad \mathbf{y}(\tau) = e^{(\tau-b)r\mathbf{L}}\mathbf{y}(b) + \int_b^\tau e^{(\tau-s)r\mathbf{L}}r\mathbf{N}(t(s), \mathbf{y}(t(s)))ds.$$

By expanding the nonlinearity around  $\tau = b$ , we can express the exact solution as

$$(3.3) \quad \mathbf{y}(\tau) = e^{(\tau-b)r\mathbf{L}}\mathbf{y}(b) + \int_b^\tau e^{(\tau-s)r\mathbf{L}} \sum_{j=0}^{\infty} \frac{\mathbf{c}_{j+1}(b)(s-b)^j}{j!} ds,$$

$$\text{where } \mathbf{c}_j(b) = \left. \frac{d^{j-1}}{d\tau^{j-1}} r\mathbf{N}(t(\tau), \mathbf{y}(t(\tau))) \right|_{\tau=b}.$$

If we assume that the Taylor series converges uniformly within the domain of interest, then we can exchange the sum and the integral. Finally, using (2.5) we see that the solution is an infinite series of  $\varphi$ -functions,

$$(3.4) \quad \mathbf{y}(\tau) = \varphi_0((\tau-b)r\mathbf{L})\mathbf{y}(b) + \sum_{j=1}^{\infty} (\tau-b)^j \varphi_j((t-b)r\mathbf{L})\mathbf{c}_j(b).$$

Notice that this series is an exponential generalization of a Taylor series, since as  $\mathbf{L} \rightarrow \mathbf{0}$  we recover the Taylor series for the solution expanded around  $b$ .

To derive the classical ODE polynomial (1.3) one replaces the exact derivatives in a truncated Taylor series with polynomial approximations. Therefore, to derive its exponential equivalent, we will truncate the infinite series in (3.4) and replace the constants  $\mathbf{c}_j$  with a polynomial derivative approximation of the nonlinear term. We also allow the initial condition  $\mathbf{y}(b)$  to be replaced with a polynomial approximation of the solution at  $\tau = b$ . We formally define this type of approximant as follows.

DEFINITION 3.2 (partitioned polynomial  $\varphi$ -expansion). *A partitioned polynomial  $\varphi$ -expansion  $\psi(\tau; b)$  of degree  $g$  is a linear combination of  $\varphi$ -functions*

$$(3.5) \quad \psi(\tau; b) = \varphi_0((\tau-b)r\mathbf{L})\mathbf{a}_0(b) + \sum_{j=1}^g (\tau-b)^j \varphi_j((t-b)r\mathbf{L})\mathbf{a}_j(b),$$

where each approximate derivative  $\mathbf{a}_j(b)$  must be computed using the values from a partitioned exponential ODE dataset  $D_N(r, s) = \{(\tau_j, \mathbf{y}_j, r\mathbf{f}_j, r\mathbf{N}_j)\}_j^w$  as follows:

1. The zeroth approximate derivative  $\mathbf{a}_0$  is calculated in the same way as for a classical ODE polynomial. In other words,

$$(3.6) \quad \mathbf{a}_0 = h(b),$$

where  $h(\tau) \approx y(t(\tau))$  is a polynomial of lowest degree that interpolates at least one solution value in  $D_N(r, s)$  and whose derivative  $h'(\tau)$  interpolates any number of full derivative values so that

$$\begin{aligned} h(\tau_k) &= \mathbf{y}_k & \text{for} & & k \in \mathcal{A} \text{ and } \mathcal{A} \neq \emptyset, \\ h'(\tau_k) &= r\mathbf{f}_k & \text{for} & & k \in B, \end{aligned}$$

and the sets  $\mathcal{A}$  and  $\mathcal{B}$  contain unique indices ranging from 1 to  $w$ .

2. The remaining approximate derivatives  $\mathbf{a}_j(b)$ ,  $j > 0$ , are calculated by differentiating polynomial approximations of the nonlinear derivative component  $rN(\mathbf{y}(t(\tau)))$ . In particular,

$$(3.7) \quad \mathbf{a}_j(b) = \left. \frac{d^{j-1}}{d\tau^{j-1}} l_j(\tau) \right|_{\tau=b}, \quad j > 0,$$

where  $l_j(\tau)$  is a Lagrange interpolating polynomial that interpolates at least one nonlinear derivative component value in the ODE dataset  $D_N(r, s)$ . We may express the conditions on  $l_j(\tau)$  mathematically as

$$l_j(\tau_k) = r\mathbf{N}_k \quad \text{for} \quad k \in \mathcal{C}^j \text{ and } \mathcal{C}^j \neq \emptyset,$$

where  $\mathcal{C}^j$  is a set that contains unique indices ranging from 1 to  $w$ .

A polynomial  $\varphi$ -expansion generalizes a classical ODE polynomial in the sense that as  $\mathbf{L} \rightarrow 0$ , it reduces to (1.3). To further understand the properties of polynomial  $\varphi$ -expansions, it is convenient to introduce a new ODE polynomial that approximates the truncated Taylor series for the local solution  $rN(t(\tau), \mathbf{y}(\tau))$ , expanded around the point  $\tau = b$ .

**DEFINITION 3.3** (ODE derivative component polynomial). *An exponential ODE derivative component polynomial of degree  $g$  is a polynomial of the form*

$$(3.8) \quad p_N(\tau; b) = \sum_{j=0}^g \frac{\mathbf{a}_{j+1}(b)(\tau - b)^j}{j!},$$

where the approximate derivatives  $\mathbf{a}_j(b)$  are given by (3.7).

Using this definition we can rewrite a polynomial  $\varphi$ -expansion in two additional ways:

1. The integral formulation:

$$(3.9) \quad \psi(\tau; b) = e^{r\mathbf{L}(\tau-\tau_0)} \underbrace{\mathbf{a}_0(b)}_{\substack{\text{Interpolating polynomial approximating } y(t(\tau)) \text{ evaluated at } b.}} + \int_b^\tau e^{r\mathbf{L}(\tau-s)} \overbrace{p_N(s; b)}^{\substack{\text{ODE derivative component polynomial expanded at } b.}} ds.$$

This representation explicitly shows the polynomial approximations that have been substituted into the integral equation (3.2). To show equivalence with the  $\varphi$ -expansion (3.5), apply the change of variables  $s = (\tau - b)\nu + b$ , and simplify using (2.5).

2. The differential formulation:  $\psi(\tau; b)$  is the solution of the linear system

$$(3.10) \quad \mathbf{y}'(\tau) = r\mathbf{L}\mathbf{y} + p_N(\tau; b), \quad \mathbf{y}(b) = \mathbf{a}_0(b).$$

From this representation we see that all polynomial  $\varphi$ -expansions can be computed by solving a linear differential equation and that it is possible to use the value of a polynomial  $\varphi$ -expansion at time  $t_1$  as an initial condition for computing the polynomial  $\varphi$ -expansion at time  $t_2 > t_1$ . To show equivalence to (3.9) simply apply the integrating factor method with  $e^{\mathbf{L}\tau}$ .

As is the case for a classical ODE polynomial, there are many ways to select the approximate derivatives for a polynomial  $\varphi$ -expansion. To simplify method construction, we introduce a special family of polynomial  $\varphi$ -expansions that is related to the Adams ODE polynomial (1.4) for classical integrators.

**DEFINITION 3.4** (Adams  $\varphi$ -expansion). *The polynomial  $\varphi$ -expansion (3.9), (3.10), (3.5) is of Adams type if it satisfies the following two conditions:*

1. *The interpolating polynomial  $h(\tau)$ , used to set the zeroth approximate derivative  $\mathbf{a}_0(b) = h(b)$ , is a Lagrange interpolating polynomial  $L_y(\tau)$  that interpolates at least one solution value in an ODE dataset.*
2. *The ODE derivative component polynomial  $p_N(\tau; b) = L_N(\tau)$  where  $L_N(\tau)$  is a Lagrange interpolating polynomial that interpolates any number of derivative component values in an ODE dataset.*

The three equivalent representations for an Adams polynomial  $\varphi$ -expansion are

1. integral  $\psi(\tau; b) = e^{r\mathbf{L}(\tau-b)}L_y(b) + \int_b^\tau e^{r\mathbf{L}(\tau-s)}L_N(s)ds,$
2. differential  $\mathbf{y}'(\tau) = r\mathbf{L}\mathbf{y} + L_N(\tau), \quad \mathbf{y}(b) = L_y(b),$
3.  $\varphi$ -expansion  $\psi(\tau; b) = \varphi_0(r\mathbf{L}(\tau-b))L_y(b) + \sum_{k=0}^g (\tau-b)^{k+1} L_N^{(k)}(b) \varphi_{k+1}(r\mathbf{L}(\tau-b)).$

We name these simplified  $\varphi$ -expansions Adams  $\varphi$ -expansions for two reasons. First, they reduce to a classical Adams ODE polynomial (1.4) as  $\mathbf{L} \rightarrow \mathbf{0}$ . Second, we can derive EAB and Adams–Moulton methods [2] using an Adams  $\varphi$ -expansion constructed from an ODE dataset with equispaced nodes.

**3.1.1. Obtaining coefficients for polynomial  $\varphi$ -expansions.** From (3.5) we see that all polynomial  $\varphi$ -expansions are linear combinations of  $\varphi$ -functions where the weights are expressed in terms of the approximate derivatives  $\mathbf{a}_j(b)$ . All approximate derivatives can be written as linear combinations of the data elements in the corresponding exponential ODE dataset, and a detailed procedure for obtaining these weights is described in [4, Sect. 3.7]. For Adams  $\varphi$ -expansions, the procedure for obtaining coefficients is simpler and is described in Appendix A.

**3.1.2. Extension to unpartitioned problems.** We can also extend the ODE dataset and the ODE polynomial for the unpartitioned problem (2.6) using local linearization. To derive an unpartitioned polynomial  $\varphi$ -expansion we

1. rewrite (2.6) in autonomous form,
2. reexpress the equation in local coordinates  $\tau$ ,
3. locally linearize around  $\tau = \ell$  to obtain (2.7) with  $\mathbf{y}_n \rightarrow \mathbf{y}(\ell)$  and  $\mathbf{J}_n \rightarrow \frac{\partial F}{\partial \mathbf{y}}(\mathbf{y}_\ell)$ , and
4. assume that the initial condition is supplied at  $\tau = b$ .

Using the integrating factor method, the corresponding solution is

$$(3.11) \quad \mathbf{y}(\tau) = e^{(\tau-b)r\mathbf{J}(\mathbf{y}_\ell)}\mathbf{y}_0 + r \int_b^\tau e^{(\tau-s)r\mathbf{J}(\mathbf{y}_\ell)} [F(\mathbf{y}_\ell) - \mathbf{J}(\mathbf{y}_\ell) \mathbf{y}_\ell + R(\mathbf{y})] ds,$$

where  $\mathbf{y}_0 = \mathbf{y}(\tau_0)$ ,  $\mathbf{y}_\ell = \mathbf{y}(\ell)$ , and  $\mathbf{J}(y) = \frac{\partial F}{\partial \mathbf{y}}(y)$  is the Jacobian of the right hand side evaluated at  $y$ . Next we introduce the *unpartitioned ODE dataset*

$$(3.12) \quad D_R(r, s) = \{(\tau_j, \mathbf{y}_j, r\mathbf{f}_j, r\mathbf{R}_j)\}_{j=1}^w, \quad \mathbf{R}_j = R(\mathbf{y}_j),$$

and an associated ODE derivative component polynomial  $p_R(\tau; b)$  that is defined identically to the ODE derivative component polynomial in Definition 3.3, except that the approximate derivatives are now formed using  $\mathbf{R}_j$  instead of  $\mathbf{N}_j$ . To obtain an unpartitioned polynomial  $\varphi$ -expansion, we make the following replacements to (3.11):

$$(3.13) \quad \mathbf{y}_0 \rightarrow \mathbf{a}_0(b), \quad \mathbf{y}_l \rightarrow \hat{\mathbf{a}}_0(\ell), \quad \text{and} \quad R(\mathbf{y}(s)) \rightarrow p_R(s; b),$$

where  $\mathbf{a}_0$  and  $\hat{\mathbf{a}}_0$  are zeroth-order approximate derivatives defined in (3.6), and  $p_R(s; b)$  is an unpartitioned ODE derivative polynomial expanded at  $b$ .

In comparison to the partitioned polynomial  $\varphi$ -expansion we have an additional parameter  $\ell$  which specified the temporal location where the unpartitioned ODE was locally linearized. Once again, we can limit the number of free parameters by introducing a special family of Adams approximations.

**DEFINITION 3.5** (unpartitioned Adams  $\varphi$ -expansion). *Let  $L_y(\tau) \approx \mathbf{y}(t(\tau))$  and  $L_R(\tau) \approx rR(\mathbf{y}(t(\tau)))$  be two Lagrange interpolating polynomials constructed from the ODE dataset (3.12) such that  $L_y(\tau)$  interpolates at least one solution value, and  $L_R(\tau)$  interpolates any number of derivative component values;*

$$(3.14) \quad L_y(\tau_k) = \mathbf{y}_k, \text{ for } k \in \mathcal{A} \text{ where } |\mathcal{A}| \geq 1, \text{ and } L_R(\tau_j) = rR(\mathbf{y}_k), \text{ for } k \in \mathcal{B}.$$

An unpartitioned Adams  $\varphi$ -expansion is an approximation of the form (3.11), (3.13) where the expansion point  $\ell$  and left integration point  $b$  are equal such that  $\ell = b$ , and  $\mathbf{a}_0(b) = \hat{\mathbf{a}}_0(b) = L_y(b)$  and  $p_R(s; b) = L_R(s)$ .

If we let  $\mathbf{J}_b = \frac{\partial F}{\partial \mathbf{y}}(L_y(b))$ , then we can write the formula for an unpartitioned Adams polynomial  $\varphi$ -expansion in the usual three ways:

$$\text{differential} \quad \mathbf{y}'(\tau) = rF(L_y(b)) + r\mathbf{J}_b(\mathbf{y}(\tau) - L_y(b)) + L_R(\tau), \quad \mathbf{y}(b) = L_y(b),$$

$$\begin{aligned} \text{integral} \quad \hat{\psi}(\tau; b) &= e^{rL(\tau-b)} L_y(b) + \int_b^\tau e^{(\tau-s)r\mathbf{J}_b} [rF(L_y(b)) - r\mathbf{J}_b L_y(b) + L_R(s)] ds \\ &= L_y(b) + \int_b^\tau e^{(\tau-s)r\mathbf{J}_b} [rF(L_y(b)) + L_R(s)] ds, \end{aligned}$$

$$\begin{aligned} \varphi\text{-expansion} \quad \hat{\psi}(\tau; b) &= L_y(b) + (\tau - b)\varphi_1(r\mathbf{J}_b(\tau - b))rF(L_y(b)) \\ &\quad + \sum_{k=0}^g (\tau - b)^{k+1} L_R^{(k)}(b)\varphi_{k+1}(r\mathbf{J}_b(\tau - b)). \end{aligned}$$

The identity  $e^{h\mathbf{A}}\mathbf{x} = \mathbf{x} + h\varphi_1(h\mathbf{A})\mathbf{A}\mathbf{x}$  was used to write the  $\varphi$ -expansion and the second formula for the integral formulation.

**3.2. A general formulation for exponential polynomial methods.** The general formulation for a partitioned exponential polynomial integrator with  $s$  stages and  $q$  outputs is

$$(3.15) \quad \begin{aligned} Y_i &= \psi_j(c_j(\alpha); b_j(\alpha)), & j &= 1, \dots, s, \\ y_j^{[n+1]} &= \psi_{j+s}(z_j + \alpha; b_{j+s}(\alpha)), & j &= 1, \dots, q, \end{aligned}$$

where  $Y_i$  are stage values and  $\psi_j(\tau; b)$  are partitioned polynomial  $\varphi$ -expansions that are constructed from a partitioned exponential ODE dataset that contains the

method's inputs, outputs, and stage values. The definition for an unpartitioned exponential polynomial integrator is identical except we replace  $\psi_j$ ,  $j = 1, \dots, q + s$ , with unpartitioned polynomial  $\varphi$ -expansions  $\hat{\psi}_j$  that also depend on the parameters  $l_j(\alpha)$ .

Equation (3.15) encapsulates all families of polynomial exponential integrators and is the exponential generalization of (1.6). However, the space of such methods is too large to explore in this initial work. We therefore restrict ourselves to the family of EPBMs.

**3.3. Exponential polynomial block methods.** Classical block methods [44] are multivalued integrators that advance multiple solution values per timestep. Depending on the structure of their coefficient matrices they compute their outputs either in serial or in parallel [18, 16]. Block methods provide a good starting point for deriving high-order polynomial integrators, since they have no stage values and their multiple inputs can be used to easily construct high-order polynomial approximations. Parallel PBMs with nodes in the complex plane were introduced in [7].

As is the case for classical PBMs, EPBMs are a good starting point for deriving high-order exponential polynomial integrators. A partitioned EPBM depends on the parameters

$$\begin{aligned} q & \text{ number of inputs/outputs,} & \{z_j\}_{j=1}^q & \text{ nodes, } z_j \in \mathbb{C}, |z_j| \leq 1, \\ r & \text{ node radius, } r \geq 0, & \{b_j\}_{j=1}^q & \text{ expansion points,} \\ \alpha & \text{ extrapolation factor} \end{aligned}$$

and can be written as

$$(3.16) \quad y_j^{[n+1]} = \psi_j(z_j + \alpha; b_j(\alpha)), \quad j = 1, \dots, q,$$

where each  $\psi_j(\tau; b)$  is a polynomial  $\varphi$ -expansion built from the exponential ODE dataset

$$(3.17) \quad D_N(r, t_n) = \underbrace{\left\{ \left( z_j, \mathbf{y}_j^{[n]}, r\mathbf{N}_j^{[n]} \right) \right\}_{j=1}^q}_{\text{method inputs}} \cup \underbrace{\left\{ \left( z_j + \alpha, \mathbf{y}_j^{[n+1]}, r\mathbf{N}_j^{[n+1]} \right) \right\}_{j=1}^q}_{\text{method outputs}}.$$

We can also write any partitioned EPBM in coefficient form as

$$(3.18) \quad \begin{aligned} y_j^{[n+1]} &= \varphi_0(\eta_j \mathbf{L}) \sum_{k=1}^q \left( A_{jk} \mathbf{y}_k^{[n]} + C_{jk} \mathbf{y}_k^{[n+1]} \right) \\ &+ r \sum_{k=1}^q \varphi_k(\eta_j \mathbf{L}) \sum_{l=1}^q \left( B_{jkl} \mathbf{N}_l^{[n]} + D_{jkl} \mathbf{N}_l^{[n+1]} \right), \end{aligned}$$

where  $\eta_j = (z_j + \alpha - b_j)$  and  $j = 1, \dots, q$ . Similarly we can write an unpartitioned EPBM as

$$(3.19) \quad y_j^{[n+1]} = \hat{\psi}_j(z_n + \alpha; b_j(\alpha)), \quad j = 1, \dots, q,$$

or in its coefficient form

$$(3.20) \quad \begin{aligned} y_j^{[n+1]} &= \left( A_{jk} \mathbf{y}_k^{[n]} + C_{jk} \mathbf{y}_k^{[n+1]} \right) + r \varphi_1(\eta_j \mathbf{J}_{b_j}) F \left( \sum_{k=1}^q \left( A_{jk} \mathbf{y}_k^{[n]} + C_{jk} \mathbf{y}_k^{[n+1]} \right) \right) \\ &+ r \sum_{k=1}^q \varphi_k(\eta_j \mathbf{J}_{b_j}) \sum_{l=1}^q \left( B_{jkl} \mathbf{R}_l^{[n]} + D_{jkl} \mathbf{R}_l^{[n+1]} \right). \end{aligned}$$

**4. Constructing Adams exponential polynomial methods.** In this section we discuss several approaches for constructing EPBMs with Adams polynomial  $\varphi$ -expansions. In particular, we derive several new classes of both parallel and serial EPBMs, and also discuss a strategy for obtaining initial conditions by composing multiple iterator methods.

**4.1. Parameter selection.** To simplify the construction of Adams EPBMs it is convenient to rewrite them in the integral form

Partitioned:

$$(4.1) \quad y_j^{[n+1]} = e^{r\eta_j \mathbf{L}} L_y^{[j]}(b_j) + \int_{b_j}^{z_j+\alpha} e^{r\mathbf{L}(\tau-s)} L_N^{[j]}(s) ds,$$

Unpartitioned:

$$(4.2) \quad y_j^{[n+1]} = L_y^{[j]}(b_j) + r\eta_j \varphi_1(r\eta_j \mathbf{J}_{b_j}) F(L_y^{[j]}(b_j)) + \int_{b_j}^{z_j+\alpha} e^{r\mathbf{J}_{b_j}(\tau-s)} L_R^{[j]}(s) ds,$$

where  $\eta_j = z_j + \alpha - b_j$  and  $j = 1, \dots, q$ . To construct these types of methods we must select

1. a set of nodes  $\{z_j\}_{j=1}^q$  that determine the input and output times,
2. the Lagrange polynomials  $L_y^{[j]}$ ,  $L_N^{[j]}$  or  $L_y^{[j]}$ ,  $L_R^{[j]}$  that respectively interpolate values or derivatives at the input and output nodes, and
3. the integration endpoints  $b_j$ .

In the following subsections we describe multiple strategies for selecting each of these parameters. However, we note that the parameters for exponential methods can also be selected in entirely different ways and in a different order than what is shown in this paper.

**4.1.1. Node selection.** Polynomial methods can be constructed using either real-valued or complex-valued nodes  $\{z_j\}_{j=1}^q$ . In general, the node type has a non-trivial effect on the linear stability of a method. For example, for diagonally implicit polynomial methods, imaginary equispaced nodes offer improved stability compared to real-valued equispaced nodes [7]. For serial methods, the node ordering will also affect stability since it determines the output ordering.

With the exception of section 4.3.1, we only consider real-valued nodes that are normalized so they lie on the interval  $[-1, 1]$  and are ordered so that

$$z_1 < z_2 < \dots < z_q.$$

This is of course only one possible choice; however, a complete discussion on optimal node selection is nontrivial, and lies outside the scope of this paper.

**4.1.2. Selecting the polynomial  $L_y^{[j]}(\tau)$  and the endpoints  $b_j$ .** The polynomial  $L_y^{[j]}(\tau)$  approximates the initial condition for the integral equation (3.2) and (3.11). To construct high-order parallel methods, we choose the highest-order polynomial that does not lead to a method with coupled outputs. This property is obtained if we let  $L_y^{[j]}(\tau) = L_y(\tau)$ , where  $L_y(\tau)$  is a Lagrange interpolating polynomial of order  $q-1$  that interpolates through each of the methods inputs so that

$$(4.3) \quad L_y(\tau) = \sum_{j=1}^q \ell_j(\tau) y_j^{[n]}, \quad \text{where} \quad \ell_j(\tau) = \prod_{\substack{k=1 \\ k \neq j}}^q \frac{\tau - z_k}{z_j - z_k}.$$

To simplify things even further, we select the integration endpoints so that they are equal to any of the node values; in other words, there must exist an integer  $k(j)$  such that  $b_j = z_{k(j)}$  for  $j = 1, \dots, q$ . Under these assumptions, the formulas (4.1) and (4.2) for the  $j$ th output of an EPBM reduce to

partitioned:

$$(4.4) \quad y_j^{[n+1]} = e^{r\eta_j \mathbf{L}} y_{k(j)}^{[n]} + \int_{b_j}^{z_j + \alpha} e^{r\mathbf{L}(\tau-s)} L_N^{[j]}(s) ds,$$

unpartitioned:

$$(4.5) \quad y_j^{[n+1]} = y_{k(j)}^{[n]} + r\eta_j \varphi_1(r\eta_j \mathbf{J}_{b_j}) F(y_{k(j)}^{[n]}) + \int_{b_j}^{z_j + \alpha} e^{r\mathbf{J}_{b_j}(\tau-s)} L_R^{[j]}(s) ds.$$

**4.1.3. Selecting the polynomials  $L_N^{[j]}(\tau)$  and  $L_R^{[j]}(\tau)$ .** The Lagrange polynomials  $L_N^{[j]}(\tau)$  and  $L_R^{[j]}(\tau)$  respectively approximate the nonlinear terms in the integral equations (3.2) and (3.11). Here we propose three strategies for choosing these polynomials that lead to either parallel or serial methods. We will first describe the motivation behind our choices and then present formulas that determine the indices of the data used to form the Lagrange polynomials. In Figure 4.1 we also present several example stencils that graphically illustrate the temporal locations of the data. To appreciate the simple geometric construction behind each of the proposed polynomials, the descriptions and formulas presented below should be read in tandem with their corresponding diagrams in Figure 4.1.

Before describing our choices, we introduce the *input index set*  $I(j)$  and the *output index set*  $O(j)$  for the Lagrange polynomials  $L_N^{[j]}(\tau)$  and  $L_R^{[j]}(\tau)$ . These sets respectively contain all of the indices of the input data and the output data that is used to construct the polynomial. For example, if  $L_N^{[j]}(\tau)$  or  $L_R^{[j]}(\tau)$  has an interpolation node at an input node  $\tau = \tau_k$ , then  $k$  would be a member of  $I(j)$ . We formally define the sets as

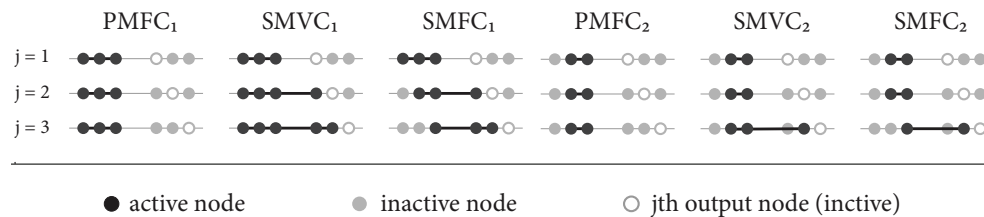


FIG. 4.1. A stencil-like diagram showcasing the three construction strategies for methods with three real-valued outputs ( $q = 3$ ) and equispaced nodes  $\{z_j\}_{j=1}^3 = \{-1, 0, 1\}$ . Each type of method must compute three outputs and therefore requires three polynomials. The stencils for each of these polynomials are stacked vertically. The horizontal gray lines represent the time axis that flows to the right. The leftmost group of three dots on each time-line represents the input nodes while the rightmost group of three dots represents the output nodes. The diagrams show the temporal locations of the data that is used to construct the Lagrange polynomials that approximate the nonlinear term. A node is colored black and labeled active if the polynomial  $L^{[j]}(\tau)$  interpolates data at this node. Conversely, the node is colored light gray and labeled inactive if the data at the node is unused. The inactive output node  $z_j + \alpha$ , which is the location where the polynomial  $\varphi$ -expansion is evaluated, is shown as a white circle with a gray border.



TABLE 4.1

Formulas describing the input index sets  $I(h)$  and the output index sets  $O(j)$  for three construction strategies.

	$I(j)$ , input index set	$O(j)$ , output index set
<b>PMFC</b> $_{\ell}$	$\{\ell, \ell + 1, \dots, q\}$	$\{\}$
<b>SMFC</b> $_{\ell}$	$\{\max(j, \ell), \max(j, \ell) + 1, \dots, q\}$	$\{\ell, \ell + 1, \dots, j - 1\}$
<b>SMVC</b> $_{\ell}$	$\{\ell, \ell + 1, \dots, q\}$	$\{\ell, \ell + 1, \dots, j - 1\}$

partitioned

$$k \in I(j) \iff L_N^{[j]}(z_k) = N_k^{[n]},$$

$$k \in O(j) \iff L_N^{[j]}(z_k + \alpha) = N_k^{[n+1]},$$

unpartitioned

$$k \in I(j) \iff L_R^{[j]}(z_k) = R_k^{[n]},$$

$$k \in O(j) \iff L_R^{[j]}(z_k + \alpha) = R_k^{[n+1]}.$$

The order of the polynomial  $L_N^{[j]}(z_k)$  or  $L_R^{[j]}(z_k)$  is therefore  $|I(j)| + |O(j)| - 1$ .

Our three proposed strategies for the construction  $L_N^{[j]}(\tau)$  and  $L_R^{[j]}(\tau)$  are as follows:

1. *Parallel maximal-fixed-cardinality- $\ell$*  (PMFC $_{\ell}$ ): The Lagrange polynomial interpolates the nonlinear derivatives at all input nodes with index greater than or equal to  $\ell$ . The set  $O(j)$  must be empty to retain parallelism.
2. *Serial maximal-variable-cardinality- $\ell$*  (SMVC $_{\ell}$ ): The Lagrange polynomial interpolates the nonlinear derivatives at all input nodes and previously computed outputs with index greater than or equal to  $\ell$ . The cardinality of set  $O(j)$  grows as we add new information.
3. *Serial maximal-fixed-cardinality- $\ell$*  (SMFC $_{\ell}$ ): The Lagrange polynomial interpolates the nonlinear derivatives at all previously computed outputs and some of the inputs with index greater than or equal to  $\ell$ . To keep the cardinality of  $I(j) \cup O(j)$  fixed for all  $j$ , we drop inputs in favor of more recently computed outputs.

The formulas for the three construction strategies are given in Table 4.1. Finally, implicit EPBMs can be created by taking  $O(j) \rightarrow O(j) \cup j$ ; however, a detailed discussion of these methods is outside the scope of this paper.

**4.2. Iterator methods for obtaining initial conditions.** EPBMs require multiple inputs at the first timestep. Since we are normally provided with only one initial value, an ERK scheme can be used to compute the solution at each of the starting nodes. However, this approach requires implementing a separate method. Moreover, it may not always be possible to match the order of a starting method with that of the EPBM. Here we present an alternative strategy for obtaining initial conditions by repeatedly applying an iterator method (i.e., a polynomial method with  $\alpha = 0$ , described in section 1.3.1).

If a set of input values contains at least one accurate initial condition, then it is possible to construct a polynomial method that iteratively improves the accuracy of the other solution values [4, Chap. 6.2]. This is accomplished by repeatedly applying a polynomial iterator method that uses the accurate value as the initial condition in a discrete Picard iteration. Each application of the method increases the order of accuracy of the inputs by one. The maximum achievable accuracy is bounded above by the order of the iterator or the order of the accurate initial condition. This idea is closely related to spectral deferred correction methods [13], except we do not use an error formulation and we do not discard any values after the iteration is complete.

Here we describe two exponential iterators that can be used to produce input values for any EPBM. They are both constructed using the strategies from section 4.1.

For simplicity we assume that the initial condition is given at the first node so that  $\mathbf{y}_1^{[0]} = y(rz_1 + t_0)$  (if this is not true, then simply redefine  $t_0$ ). We first obtain a coarse estimate for the remaining initial values by assuming that the solution is equal at all the nodes. We then repeatedly apply an iterator to improve the accuracy of this coarse estimate. We can abstractly write the formula for this iteration as

$$(4.6) \quad \mathbf{y}^{[0]} = M^k \mathbf{c}, \quad \mathbf{c}_j = \mathbf{y}_1^{[0]}, \quad j = 1, \dots, q,$$

where  $\mathbf{c}$  denotes the initial coarse approximation,  $M$  denotes the PBM iterator method, and the computation involves applying the method  $k$  times to the course initial condition  $\mathbf{c}$ .

To construct the method  $M$  we propose the following parameters:  $L_y^{[j]}(\tau) = L_y(\tau)$  from (4.3),  $b_j = z_1$ , and  $L_N^{[j]}(\tau)$  or  $L_R^{[j]}(\tau)$  should be constructed in accordance with the formulas for PMFC $_\ell$  or SMFC $_\ell$ . Selecting PMFC $_\ell$  leads to a parallel iterator while SMFC $_\ell$  leads to a serial iterator. The SMVC $_\ell$  construction cannot be used since the input and output nodes overlap when  $\alpha = 0$ . For typical node choices  $\{z_j\}_{j=1}^q$  it is sufficient to run the iteration  $q$  times; however, node sets that allow for higher accuracy can benefit from additional iterations. Finally, as  $k \rightarrow \infty$  the iteration converges to the stages and outputs of a fully implicit exponential collocation method.

Iterators can also be applied to construct composite EPBMs. An iterator and a propagator that share the same nodes can be combined to create a composite method whose computation involves first advancing the timestep with the propagator and then correcting the output  $\kappa$  times with the iterator. The composite method can be written abstractly as

$$(4.7) \quad \mathbf{y}^{[n+1]} = M^\kappa P \mathbf{y}^{[n]},$$

where  $P$  denotes a propagator and  $M$  the iterator. As we show in our numerical experiments, this method construction can lead to methods with improved stability properties.

**4.3. Parallel Adams PBMs with Legendre nodes.** The parameter choices proposed in section 4.1 can be used to construct families of partitioned or unpartitioned exponential methods. To obtain one single method it is necessary to select a set of nodes and the extrapolation parameter  $\alpha$ .

For the sake of brevity, we only analyze parallel polynomial exponential methods where  $L_N(\tau)$  is constructed in accordance to the PMFC strategy. Furthermore, we only consider real-valued nodes that are given by the union of negative one and the  $q - 1$  Legendre points so that

$$(4.8) \quad \{z_j\}_{j=1}^q = \{-1, x_1, \dots, x_{q-1}\},$$

where  $x_j$  is the  $j$ th zero of the  $q$ th Legendre polynomial  $P_q(x)$ . Since we are using Legendre points we choose the PMFC $_2$  construction strategy so that the polynomials for  $L_N^{[j]}(\tau)$  or  $L_R^{[j]}(\tau)$  are both constructed using only the Legendre nodes. We also select  $b_j = z_1$  so that the integration endpoint is negative one. We describe the method parameters in Table 4.2.

Like all polynomial methods, these Legendre EPBMs are parametrized in terms of the extrapolation factor  $\alpha$ ; we can write the method abstractly as  $M(\alpha)$ . Different choices will impact both the method's stability, accuracy, and susceptibility to round-off errors. We will primarily focus on Legendre propagators with  $\alpha = 2$ , and Legendre

TABLE 4.2

Parameters for Legendre EPBMs. The coefficients for methods with  $q = 2, 3, 4$  are shown in Appendix B.

$\{z_j\}_{j=1}^q$	:	$\{-1, x_{q-1}, \dots, x_{q-1}\}$ from Eqn (4.8)
$L_y^{[j]}(\tau)$	:	$\sum_{j=1}^q \ell_j(\tau) y_j^{[n]}$ from Eqn (4.3)
$L_N^{[j]}(\tau)$ or $L_R^{[j]}(\tau)$	:	PMFC <sub>2</sub>
$\{b_j\}_{j=1}^q$	:	$\{z_1\}_{j=1}^q$

TABLE 4.3

Pseudocode for a single timestep of a partitioned composite EPBM method (4.9) where the method  $M(\alpha)$  is constructed using the parameters from Table 4.2. All for loops that can be run in parallel have been explicitly annotated.

Partitioned Composite EPBM (4.9) with $M(\alpha)$ from Table 4.2	
# - Definitions	
$\eta_j(\alpha) = z_j + \alpha + 1$	
$\hat{\ell}_j(\tau) = \prod_{k=1, k \neq j}^{q-1} \frac{\tau - x_k}{x_j - x_k}$ # $x_j$ are the Legendre nodes; see Eqn. (4.8).	
$L_N(\tau) = \sum_{j=1}^{q-1} \hat{\ell}_j(\tau) \mathbf{N}_{j+1}$	
# - Propagator	
<b>for</b> $j = 2$ to $q$ # parallelizable loop	
$\mathbf{N}_j = N(y_j^{[n]})$	
<b>for</b> $j = 1$ to $q$ # parallelizable loop	
$y_j^{[n+1]} = \varphi_0(r\mathbf{L}\eta_j(\alpha))y_1^{[n]} + \sum_{k=0}^{q-1} (\eta_j(\alpha))^{k+1} \varphi_{k+1}(r\mathbf{L}\eta_j(\alpha))L_N^{(k)}(-1)$	
# - Iterator	
<b>for</b> $k = 1$ to $\kappa$	
<b>for</b> $j = 2$ to $q$ # parallelizable loop	
$\mathbf{N}_j = N(y_j^{[n]})$	
<b>for</b> $j = 1$ to $q$ # parallelizable loop	
$y_j^{[n+1]} = \varphi_0(r\mathbf{L}\eta_j(0))y_1^{[n+1]} + \sum_{k=0}^{q-1} (\eta_j(0))^{k+1} \varphi_{k+1}(r\mathbf{L}\eta_j(0))L_N^{(k)}(-1)$	

iterators with  $\alpha = 0$  for computing initial conditions. For dispersive equations we also briefly consider composite methods of the form

$$(4.9) \quad M_{\text{composite}}(\alpha) = M(0)^\kappa M(\alpha).$$

In Table 4.3 we show pseudocode for the composite method with Legendre nodes. If  $\kappa=0$ , then the method reduces to the standard Legendre EPBM.

Finally, though we have found that Legendre based nodes lead to efficient methods, they are by no means the optimal choice. However, optimal node selection is nontrivial and should be treated as a separate topic.

**4.3.1. Parallel Adams EPBMs with imaginary equispaced nodes.** We demonstrate the flexibility of the polynomial framework by reusing the PMFC strategy to construct two example parallel EPBM methods using the set of  $m$  imaginary equispaced nodes

$$(4.10) \quad \chi_{j,m} = i \left( 1 - \frac{2(j-1)}{m-1} \right), \quad j = 1, \dots, k.$$

The parameters for the two methods are described in Table 4.4. Method A is identical to the method from Table 4.2 except that the Legendre nodes have been replaced with imaginary equispaced nodes, while method B is the exponential equivalent of

TABLE 4.4

Parameters for EPBMs with imaginary equispaced nodes. The constants  $\chi_{i,j}$  represent the imaginary equispaced formula from (4.10).

	Method A	Method B
$\{z_j\}_{j=1}^q :$	$\{-1, \chi_{1,q-1}, \dots, \chi_{q-1,q-1}\}$	$\{\chi_{j,q}\}_{j=1}^q$
$L_y^{[j]}(\tau) :$	$L_y(\tau)$ from (4.3)	$L_y(\tau)$ from (4.3)
$L_N^{[j]}(\tau)$ or $L_R^{[j]}(\tau) :$	PMFC <sub>2</sub>	PMFC <sub>1</sub>
$\{b_j\}_{j=1}^q :$	$\{z_1\}_{j=1}^q$	$\{z_j\}_{j=1}^q$

the BAM method from [7]. Serial imaginary methods can also be constructed by using the SMFC and SMVC strategies for imaginary nodes that are proposed in [4].

As we have shown here, constructing methods with complex-valued nodes is no more difficult than constructing methods with real-valued nodes. However, extending into the complex plane introduces complexities, including (1) the solution must be sufficiently analytic off the real time-line, (2) matrix exponentials must also be efficient to compute at complex times, and (3) the code for the ODE right hand side must accept complex arguments.

For diagonally implicit PBMs from [7], extending into the complex plane provided significantly improved stability compared to real-valued nodes. However, for exponential PBMs we can obtain good stability with real-valued nodes while avoiding the aforementioned issues. With the exception of a single numerical experiment contained in the appendix, we therefore focus entirely on EPBMs with real-valued nodes.

**5. Linear stability.** We now briefly analyze the linear stability properties of partitioned and unpartitioned Legendre EPBMs from Table 4.2. Like all unpartitioned integrators, any unpartitioned EPBM will integrate the classical Dahlquist test problem  $y' = \lambda y$  exactly, and therefore its linear stability region is always equal to the left-half-plane  $\text{Re}(h\lambda) \leq 0$ . To analyze linear stability for partitioned methods we use the generalized Dahlquist test problem [1, 25, 26, 10, 12, 29, 19, 41]

$$(5.1) \quad y' = \lambda_1 y + \lambda_2 y,$$

where the terms  $\lambda_1 y$  and  $\lambda_2 y$  are meant to respectively represent the linear and nonlinear terms. When applied to (5.1), an EPBM with  $q$  inputs reduces to the matrix iteration

$$(5.2) \quad \mathbf{y}_{n+1} = \mathbf{M}(z_1, z_2, \alpha) \mathbf{y}_n,$$

where  $\mathbf{M}$  is a  $q \times q$  matrix,  $z_1 = h\lambda_1$ ,  $z_2 = h\lambda_2$ , and  $h$  denotes the timestep. The stability region  $S$  is the subset of  $\mathbb{C}^2 \times \mathbb{R}^+$  where  $\mathbf{M}(z_1, z_2, \alpha)$  is power bounded, so that

$$(5.3) \quad S = \left\{ z : \sup_{n \in \mathbb{N}} \|\mathbf{M}(z_1, z_2, \alpha)^n\| < \infty \right\}.$$

The matrix  $\mathbf{M}(z_1, z_2, \alpha)$  is power bounded if its eigenvalues lie inside the closed unit disk, and any eigenvalues of magnitude one are nondefective. The stability region  $S$  is five-dimensional and cannot be shown directly. Instead, we overlay two-dimensional slices of the stability regions formed by fixing  $z_1$  and  $\alpha$  while allowing  $z_2$  to vary. The choices for  $z_1$  are

1.  $z_1 \in -1 \cdot [0, 30]$ , negative real values mimic a problem with linear dissipation;
2.  $z_1 \in i \cdot [0, 30]$ , imaginary values mimic linear advection or dispersion;
3.  $z_1 \in \exp\left(\frac{3\pi i}{4}\right) \cdot [0, 30]$ , complex values mimic both dispersion and dissipation.

We compare the linear stability regions of Legendre EPBMs to those of EAB [2] and the fourth-order ERK method from [12]. It is interesting to compare EPBMs to EABs since both methods are constructed using high-order Lagrange polynomials. On the other hand, ERK methods serve as a good benchmark for gauging stability since they are stable on a variety of problems [35, 27, 12, 19].

In Figure 5.1 we show the stability regions for fourth- and eighth-order Legendre EPBMs with  $\alpha = 2$  compared against those of ERK4 and the fourth- and eighth-order

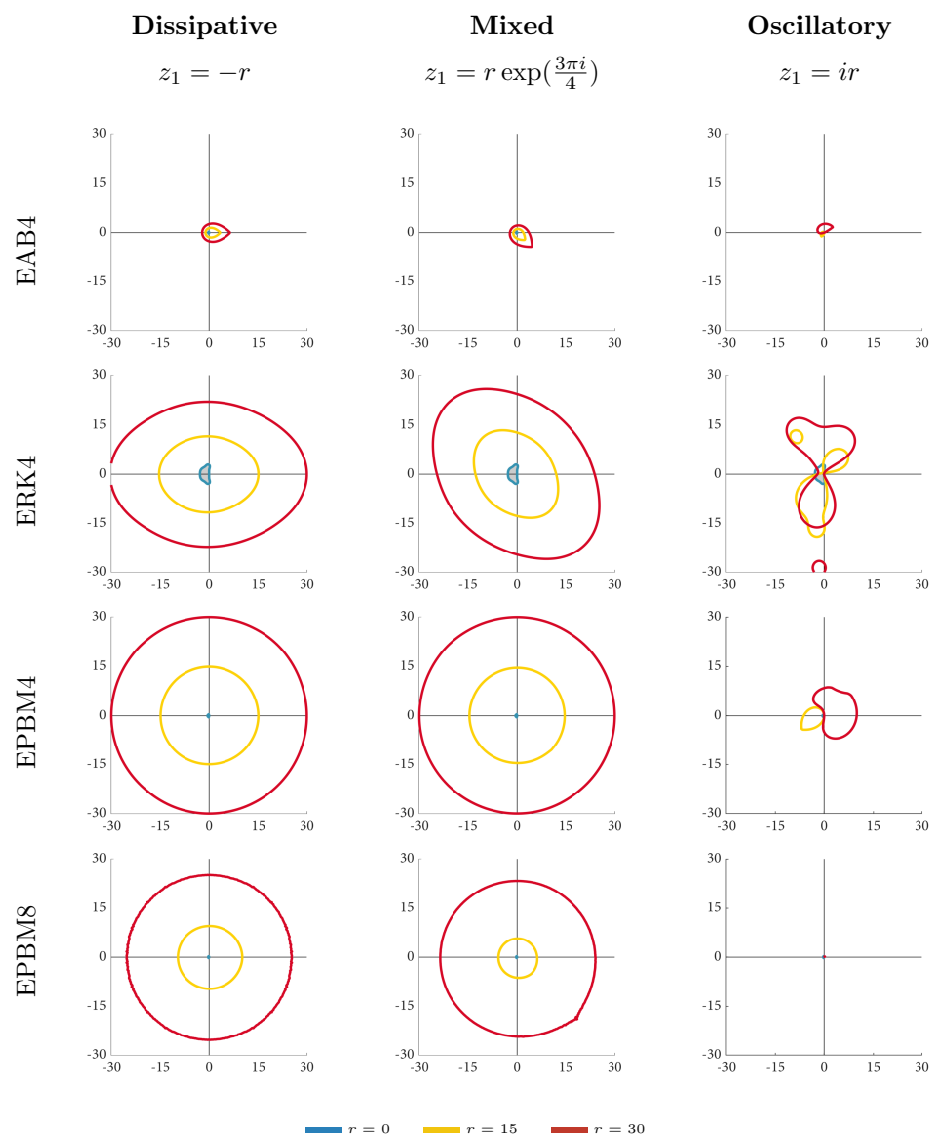


FIG. 5.1. Linear stability regions of the fourth-order EAB [2], the fourth-order ERK4 [12] and the fourth-order and eighth-order Legendre EPBMs from Table 4.2. Each row corresponds to a method and each column to one of the three choices for  $z_1$  described in section 5. The colored contours represent the stability regions for  $z_1$  with magnitudes of 0, 15, and 30. We did not include the stability regions for eighth-order EAB in this plot since they are too small to be visible at this scale.

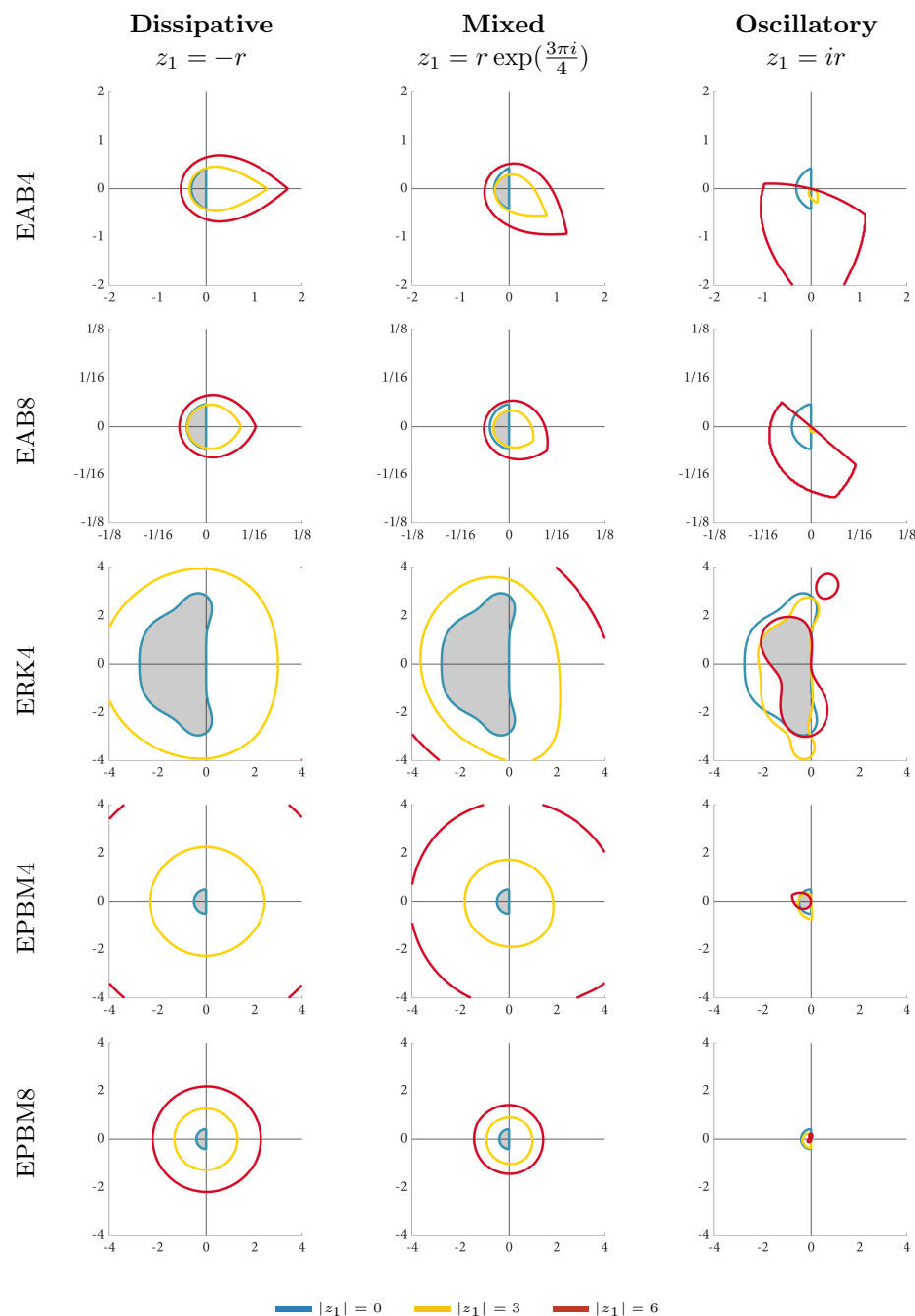


FIG. 5.2. Magnified stability regions from Figure 5.1. The axis used in the plots for EAB4 and EAB8 are magnified by a factor of 2 and 32, respectively, compared to those of ERK4 and the Legendre EPBMs. The colored contours now denote stability regions, where  $z_1$  has magnitude 0, 3, and 6.

EAB methods. In Figure 5.2 we also show magnified plots that highlight the stability properties near the origin.

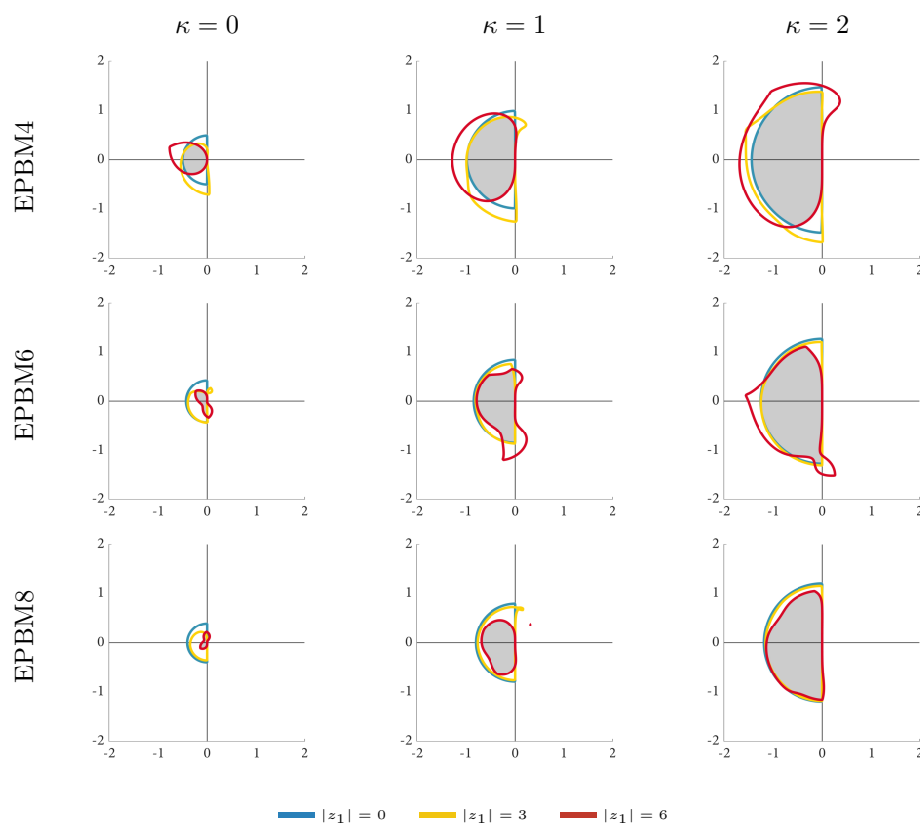


FIG. 5.3. Magnified stability regions for composite Legendre methods (4.9) on the oscillatory Dahlquist test problem where  $z_1 = ir$ . The composite method takes one propagation step with  $\alpha = 2$  and  $k$  iteration steps with  $\alpha = 0$ . The stability regions increase substantially even after a single iteration step.

When dissipation is present, the stability regions for EPBMs are significantly larger than those of EAB schemes. The difference between the two methods increases with order, with the eighth-order EPBM possessing a stability region that is approximately 32 times larger than the eighth-order EAB method for small  $|z_1|$ . For large values of  $|z_1|$ , the stability regions of EPBMs are comparable to those of the fourth-order Runge–Kutta; however, for smaller  $|z_1|$ , Runge–Kutta integrators display larger stability regions.

For the purely oscillatory Dahlquist test problem, all three families of integrators have small stability regions that contract as  $|z_1|$  increases. For small  $|z_1|$  ERK has the largest stability regions, followed by EPBM, and EAB last. These results suggest that high-order EPBMs will exhibit poor stability properties for purely advective or dispersive equations. However, we can remedy the situation by considering the composite method (4.9). In Figure 5.3 we plot the stability regions for the composite method with  $q = 4, 6, 8$  and  $\kappa = 0, 1, 2$ , where  $\kappa$  is the number of times the iterator method is applied. The computational cost of the composite method scales by a factor of  $(\kappa + 1)$  compared to the propagator; therefore it is critical that the stability regions increase by a similar factor or the composite method will not be as efficient as the propagator alone. Fortunately, by applying even a single iterator the linear stability

properties increase drastically, suggesting that composite schemes are suitable for efficiently solving advective or dispersive equations.

Finally, since the parameter choices and plot axes in Figure 5.1 are identical to those presented in [5] we can also compare the stability regions of EPBMs to those of high-order exponential spectral deferred correction (ESDC) methods constructed with Chebyshev nodes. Overall ESDC methods have significantly improved stability regions compared to all methods shown in this paper. Moreover, unlike EPBM or EAB, the stability regions of high-order ESDC methods grow with order.

**6. Numerical experiments and discussion.** We investigate the efficiency of the Legendre EPBMs from Table 4.2 by conducting six numerical experiments consisting of four partitioned systems and two unpartitioned systems. The systems all arise from solving partial differential equations using the method of lines. In each case, the reference solutions were computed numerically by averaging the outputs of multiple methods that were run using a small timestep; the averaging prevents any one method from appearing to converge past discretization error.

We divide our numerical experiments and the subsequent discussion into a partitioned and an unpartitioned section. In both cases we compare Legendre EPBMs against EAB [2, 47], ESDC [5], and the fourth-order ERK methods from [12] and [34]. Each of these methods pertains to a different method family: EAB methods are linear multistep methods, ESDC are one-step methods, and EPBM are block methods. We selected EAB and ESDC schemes because both families can be constructed at arbitrarily high orders of accuracy for both partitioned and unpartitioned problems. This allows a fair comparison across a range of orders. As a reference, we also include fourth-order ERK schemes since they have proven very efficient compared to a host of other methods [34, 40, 35].

We provide the code that was used to run the experiments in [6]. All the timing results presented in this paper were produced on a 12 core, 2.2 GHz Intel Xeon E5-2650 v4 processor with hyper-threading enabled. Finally, since the equations and spatial discretizations are identical to those presented in [5], we only list the critical information for each problem below.

**6.1. Partitioned numerical experiments.** We consider the following equations that are each equipped with periodic boundary conditions:

1. The Kuramoto–Sivashinsky (KS) equation from [30, 27],

$$(6.1) \quad \frac{\partial u}{\partial t} = -\frac{\partial^2 u}{\partial x^2} - \frac{\partial^4 u}{\partial x^4} - \frac{1}{2} \frac{\partial}{\partial x} (u^2),$$

$$u(x, t = 0) = \cos\left(\frac{x}{16}\right) \left(1 + \sin\left(\frac{x}{16}\right)\right), \quad x \in [0, 64\pi],$$

integrated to time  $t = 60$  using a 1024 point Fourier spectral discretization in  $x$ .

2. The Nikolaevskiy equation from [46, 19],

$$(6.2) \quad \frac{\partial u}{\partial t} = \alpha \frac{\partial^3 u}{\partial x^3} + \beta \frac{\partial^5 u}{\partial x^5} - \frac{\partial^2}{\partial x^2} \left( r - \left(1 + \frac{\partial^2}{\partial x^2}\right)^2 \right) u - \frac{1}{2} \frac{\partial}{\partial x} (u^2),$$

$$u(x, t = 0) = \sin(x) + \epsilon \sin\left(\frac{x}{25}\right), \quad x \in [-75\pi, 75\pi],$$

where  $r = 1/4$ ,  $\alpha = 2.1$ ,  $\beta = 0.77$ , and  $\epsilon = 1/10$ . The equation is integrated to time  $t = 50$  using a 4096 point Fourier spectral discretization.



3. The Korteweg–deVries (KdV) equation from [49],

$$(6.3) \quad \frac{\partial u}{\partial t} = - \left[ \delta \frac{\partial^3 u}{\partial x^3} + \frac{1}{2} \frac{\partial}{\partial x} (u^2) \right],$$

$$u(x, t = 0) = \cos(\pi x), \quad x \in [0, 2],$$

where  $\delta = 0.022$ . This equation is integrated to time  $t = 3.6/\pi$  using a 512 point Fourier spectral discretization.

4. The barotropic quasigeostrophic equation on a  $\beta$ -plane with linear Ekman drag and hyperviscous diffusion of momentum from [19]:

$$(6.4) \quad \partial_t \nabla^2 \psi = - [\beta \partial_x \psi + \epsilon \nabla^2 \psi + \nu \nabla^{10} \psi + \mathbf{u} \cdot \nabla (\nabla^2 \psi)],$$

$$\psi(x, y, t = 0) = \frac{1}{8} \exp \left( -8 (2y^2 + x^2/2 - \pi/4)^2 \right), \quad (x, y) \in [-\pi, \pi],$$

where  $\beta = 10$ ,  $\epsilon = 1/100$ ,  $\nu = 10^{-14}$ . The equation is integrated to time  $t = 5$  using a  $512 \times 512$  point Fourier discretization.

Each of these four equations is solved in Fourier space. Since linear derivative operators diagonalize in Fourier space, each initial value problem has the form  $\mathbf{y}' = \mathbf{L}\mathbf{y} + N(t, \mathbf{y})$ , where  $\mathbf{L}$  is a diagonal operator that includes all the discretized linear derivative terms. For example, for the KdV equation  $\mathbf{L} = \text{diag}(i\delta \mathbf{k}^3)$ , where  $\mathbf{k}$  is a vector of Fourier wave numbers. Computing matrix-vector products with  $\varphi$ -functions of  $\mathbf{L}$  now amounts to multiplication with a diagonal matrix, or equivalently elementwise vector multiplication. Moreover, since our experiments only consider constant stepsizes, we can efficiently precompute and store the diagonal entries of  $\varphi_j(\mathbf{L})$  as a vector. For small wave numbers with magnitude less than one we compute the diagonal entries of  $\varphi_j(\mathbf{L})$  using the contour integral method [27], while for larger wave numbers we simply use the recurrence relation  $\varphi_{j+1}(z) = z^{-1}(\varphi_j(z) - 1/j!)$ ,  $\varphi_0(z) = e^z$ . Last, for antialiasing we apply the standard two-thirds rule to zero out the top third of the spectrum.

One of the key features of Legendre EPBMs is their inherent parallelism that allows each of the method's outputs to be computed simultaneously. In order to investigate the potential advantages of this approach we implemented all the partitioned integrators in Fortran and used OpenMP to parallelize the EPBM timestep computation. The code was therefore run using a single thread for serial methods and using  $q$  threads for a Legendre method with  $q$  nodes.

Finally, in Appendix C we also briefly compare EPBMs with imaginary equispaced nodes to the EPBMs with Legendre nodes on the Kuramoto equation.

**6.2. Discussion of partitioned numerical experiments.** In Figure 6.1 we present plots of relative error versus stepsize and relative error versus computational time for each of the four partitioned problems. In Figure 6.2 we also investigate the efficiency of EPBMs when they are run in serial on a single thread by repeating our Kuramoto experiment. For the dispersive KdV equation we conduct a separate test in Figure 6.3 using the composite Legendre methods (4.9) in order to validate the results of the linear stability analysis. Finally, in Figure 6.4 we investigate the effects of varying  $\alpha$  on the Nikolaevskiy equation where high-order EPBMs showed order reduction.

Stability, accuracy, and computational cost per timestep are the three factors that determine a method's overall efficiency. An ideal method should have a large stability region that allows for coarse stepsizes, a small error constant to outcompete other

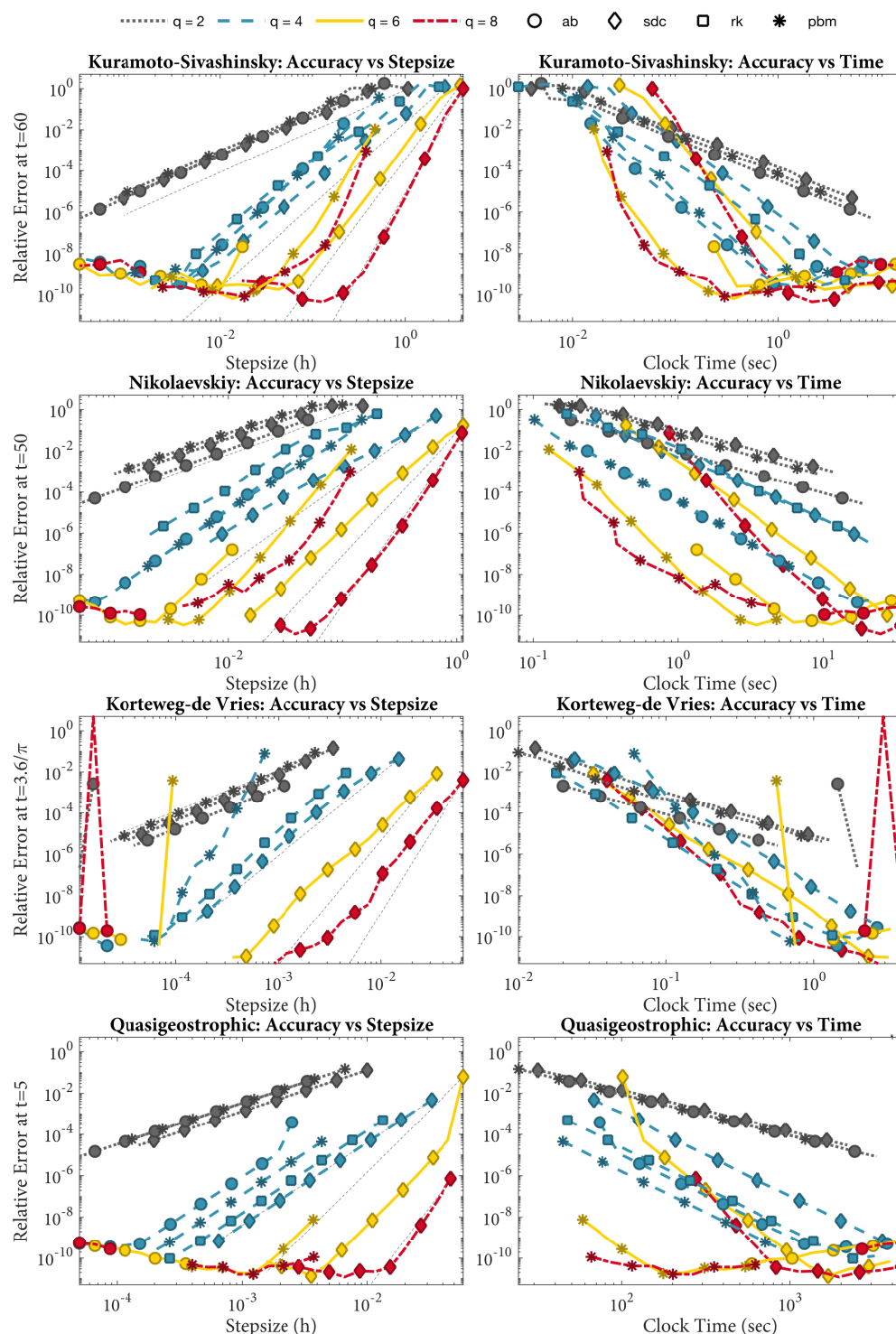


FIG. 6.1. Plots of accuracy versus stepsize and accuracy versus clock time for the four partitioned equations. All the methods shown in this figure are exponential integrators. Color represents order of accuracy and the markers denote the method type. In the legend Pbm stands for polynomial block methods, Sdc for spectral deferred correction, rk for Runge-Kutta, and ab for Adams-Bashforth. The thin dashed gray lines in the accuracy versus stepsize plots correspond to second-, fourth-, sixth-, and eighth-order convergence.

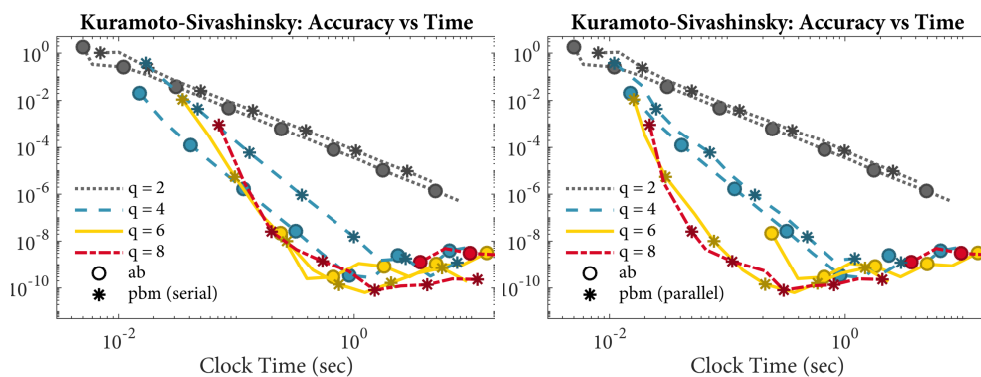


FIG. 6.2. Quantifying the effects of parallelism for EPBMs on the KS equation. The two plots show the accuracy versus clock time for EAB methods and EPBMs that are implemented in serial (left) and in parallel (right). The right plot is identical to the one shown in Figure 6.1 except that the ESDC and ERK methods have been removed.

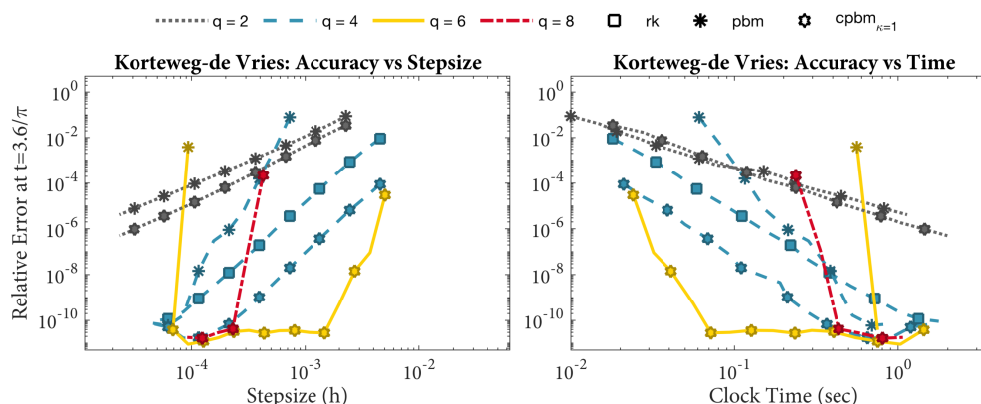


FIG. 6.3. Composite methods for solving KdV. We compare Legendre EPBMs to the composite Legendre methods (4.9) with  $k = 1$ . We also include ERK4, which was one of the most efficient methods for solving KdV in Figure 6.1. As suggested by the linear stability results, composing a PBM with a single iterator leads to a composite method with drastically improved stability on dispersive equations.

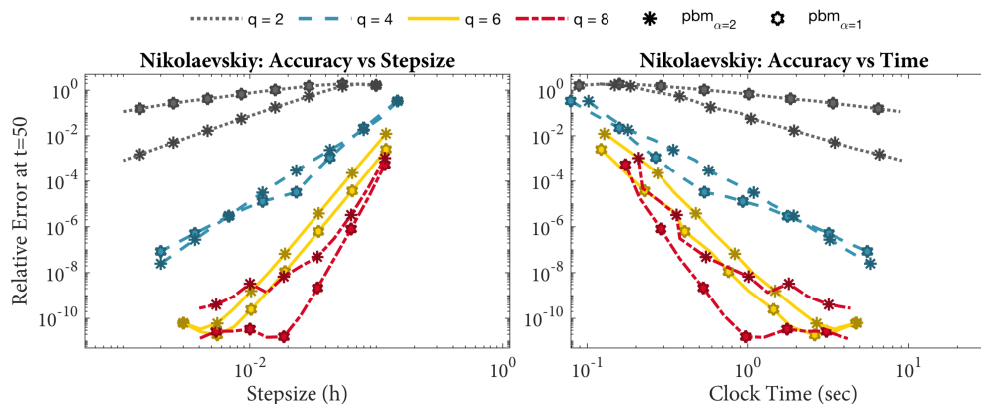


FIG. 6.4. Quantifying the effects of the extrapolation parameter  $\alpha$ . Accuracy versus stepsize and accuracy versus clock time for the Nikolaevskiy equation using Legendre EPBMs with  $\alpha = 2$  and  $\alpha = 1$ . High-order methods benefit from a smaller  $\alpha$  to avoid problems with round-off errors; however, low-order methods have better error constants with larger  $\alpha$ .

methods of the same order, and a low computational cost per timestep. Each of the three method families presented in our numerical experiments has different strengths and weaknesses. EAB methods have the lowest cost per timestep, but they suffer from limited stability and poor error constants at high orders. EPBMs have improved error constants and stability compared to EAB and nearly the same timestep cost when implemented in parallel. ESDC methods have excellent stability and accuracy compared to other methods; however, their computational cost is significantly larger.

The stability region and error constant are fixed properties of a method; however, the cost per timestep is highly problem dependent. For the partitioned experiments we precompute the matrix exponential functions, and the cost of computing a matrix-vector product  $\varphi_j(\mathbf{L})\mathbf{x}$  is equivalent to that of an elementwise vector multiplication between  $\mathbf{x}$  and a precomputed vector. Therefore, the majority of the computational work for the partitioned experiments is due to nonlinear function evaluations, and the cost of a method is determined by the number of function evaluations required per timestep.

EAB methods only require a single nonlinear function evaluation per timestep and serve as a benchmark for optimal compute cost. In contrast, Legendre EPBMs require one function evaluation per node. However, since the outputs are independent, each function evaluation can be computed in parallel. If we neglect parallel overhead, then EPBMs have the same cost per timestep as an EAB method. On the other hand an ESDC method with  $q$  nodes requires  $q(q-1)$  serial function evaluations per timestep, making the cost significantly higher than that of the other methods. Finally the fourth-order Runge–Kutta method requires four serial function evaluations per timestep.

Among the second-order methods, EAB is consistently most efficient due to its superior accuracy compared to the other integrators. At fourth order, no family of integrator proved more efficient for all the test problems. On the Kuramoto and Nikolaevskiy equations, EAB was the most efficient at fine error tolerances, while at coarse tolerances EPBM was more efficient since EAB was no longer stable. For KdV the fourth-order ETDRK4 was the most efficient except near discretization error where EPBM is stable. At orders six and eight, the Legendre EPBMs were able to consistently achieve the best accuracy in the smallest amount of computational time. This was due to their improved accuracy and use of parallelism. For one-dimensional problems with dissipation, EPBMs were significantly more efficient than existing methods. However, when solving Kuramoto and Nikolaevskiy, EPBM8 showed order reduction at small timesteps. This phenomenon is due to rounding error and is present in other high-order polynomial integrators [7, 4]. As is the case for classical integrators, the issue can be resolved by simply selecting a smaller extrapolation factor. As shown in Figure 6.4, reducing  $\alpha$  to one makes EPBM8 the most efficient method for obtaining highly accurate solutions.

When EPBMs were run in serial on the Kuramoto equation (Figure 6.2), their computational advantages decreased significantly. Fourth-order EPBM is now less efficient than the fourth-order EAB, while the sixth- and eighth-order EPBMs only marginally outperformed EAB schemes at the lowest error tolerances. This behavior is expected because a serial implementation of a Legendre method with  $q$  nodes requires  $q-1$  function evaluations per timestep, compared to the single function evaluation for an EAB scheme. In short, Figure 6.2 shows that the large computational gains for EPBM schemes are possible only if the methods can be parallelized efficiently.

Finally, as predicted by linear stability analysis, EAB methods and Legendre EPBMs demonstrated poor stability on the dissipative KdV equation and only converged at smaller timesteps. However, the composite Legendre method with  $k=1$

shown in Figure 6.3 was able to retain stability across a much wider range of timesteps. Moreover, though its cost per timestep is twice that of a Legendre EPBM, the improved stability and accuracy made both the fourth-order and the sixth-order EBPM significantly more efficient than ETD4.

When comparing methods across different orders of accuracy we see that high-order EPBMs are frequently able to obtain extremely accurate solutions in the same amount of time that it takes a second-order integrator to achieve only a few digits of accuracy. Moreover, if we fix the timestep, we see that all stable EPBMs run in nearly the same computational time. The equations presented here are simple and the domains are conveniently periodic so that we may diagonalize the linear operators. Nevertheless our results suggest that for similar equations, or more broadly for problems where efficient parallelism is possible, low-order EPBMs should be considered only if one wants the solution in the fastest possible time and requires very little accuracy; if more accuracy is required, then, barring other computational restrictions such as limited memory, a high-order EPBM can obtain a highly accurate solution at little to no extra cost.

**6.3. Unpartitioned initial value problems.** We consider the two-dimensional advection-diffusion-reaction (ADR) equation with homogeneous Neumann boundary conditions from [40],

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

The equation is integrated to time  $t = 1/100$  and the spatial discretization employs standard second-order accurate finite differences on a  $200 \times 200$  point grid. We solve this equation using the two parameter sets described in Table 6.1 that lead to either a stiff linearity or a stiff nonlinearity.

To compute matrix-vector products with  $\varphi$ -functions we apply the Krylov-based KIOPs algorithm [17] using the exact Jacobian. In Figure 6.5 we show results for Legendre EPBMs of orders two, four, six, and eight compared against unpartitioned Adams–Bashforth methods [47], unpartitioned ESDC methods [5], and the fourth-order unpartitioned ERK method EPIRK43s [34].

**6.3.1. Discussion of unpartitioned numerical experiments.** All unpartitioned exponential integrators require matrix-vector products with  $\varphi$ -functions of the Jacobian. Since the Jacobian varies in time, we are no longer able to store the exponential matrix functions at the first time step. Moreover, the Jacobian matrices are nondiagonal, therefore the majority of the computational effort at each time step is now due to the vector products with  $\varphi$ -functions.

Krylov methods like KIOPS [17] and PHIPM [37, 36] approximate matrix-vector products of  $\varphi$ -functions within a Krylov subspace. The operation to build a Krylov

TABLE 6.1

*Parameters for the two-dimensional ADR equation. We also show the spectral radius  $\rho$  for the linear and nonlinear terms.*

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

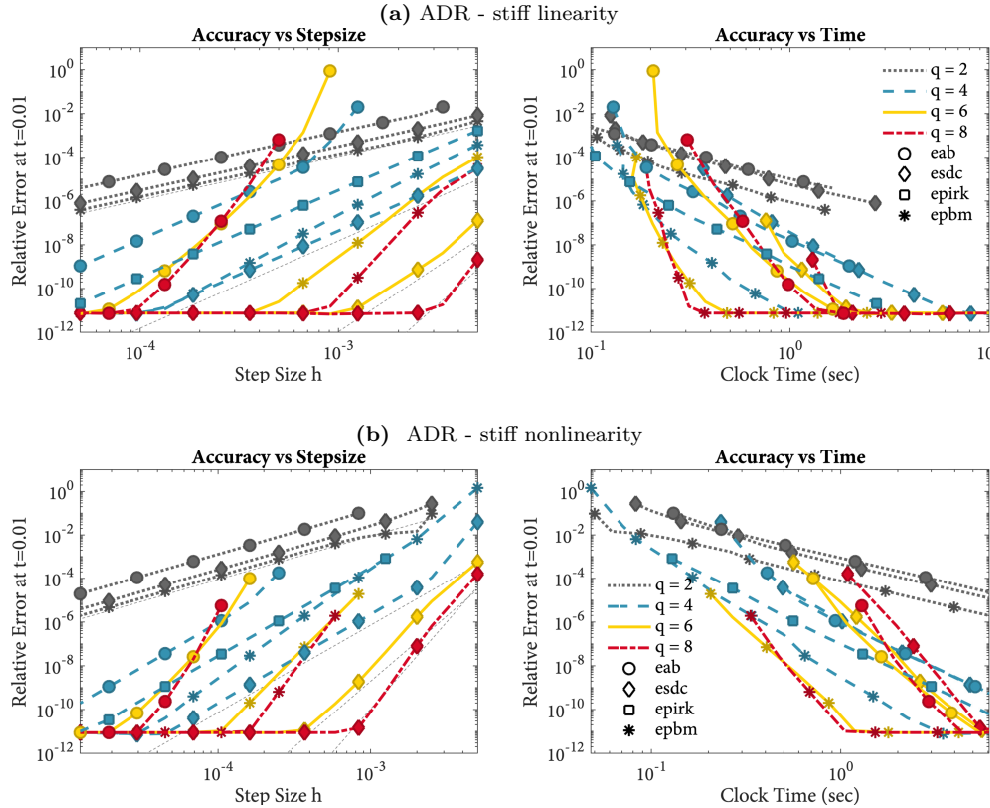


FIG. 6.5. (a) Diagrams for the ADR equation with a stiff linearity. (b) Diagrams for the ADR equation with a stiff nonlinearity. The dashed lines in the accuracy diagrams correspond to second-, fourth-, sixth-, and eighth-order convergence.

space and approximate a  $\varphi$ -function is called a *projection* and its cost will depend on the spectrum of  $\mathbf{A}$ . The cost of a time integrator is therefore dependent on the total number of projections per time step. In general, every  $\varphi$ -function in an arbitrary linear combination must be computed using a separate projection. However, expressions of the form

$$(6.6) \quad \varphi_0(\tau\mathbf{A})\mathbf{x}_0 + \tau\varphi_1(\tau\mathbf{A})\mathbf{x}_1 + \tau^2\varphi_2(\tau\mathbf{A})\mathbf{x}_2 + \cdots + \tau^p\varphi_p(\tau\mathbf{A})\mathbf{x}_p$$

can be evaluated at multiple  $\tau$  values in a single projection. Furthermore, this computation is made cheaper if  $\tau$  is small. This fact can be used to construct highly efficient integrators such as EPIRK methods [47, 48, 40, 34] whose coefficients are specifically chosen to minimize the total number of projections required at each timestep.

It is important to state that Krylov methods are not the only way to compute the expression (6.6). Leja interpolation [9] and scaling-and-squaring based algorithms [21] are two such alternatives; a detailed comparison of these methods is presented in [8], and [21] contains a survey of existing codes.

In our experiment we used KIOPs to compute the exponential terms. EAB methods once again serve as a benchmark for optimal cost per timestep as they require only one projection where  $\tau = h$ . In contrast, EPIRK43s requires two serial projections: the first has  $\tau = h$  and the second has  $\tau = h/9$ . For polynomial integrators, all

polynomial  $\varphi$ -expansions can be written in the form (6.6) and therefore each output of any EPBM requires one projection to compute. However, since Legendre EPBMs are constructed using a single Adams polynomial  $\varphi$ -expansion, all the outputs can be computed in one projection where  $\tau = 2h$ . Finally, ESDC methods are the most expensive, requiring one projection per correction iteration, meaning that a method with  $q$  nodes requires  $q$  projections to  $\tau = h$ . Under these conditions, the cost per timestep of an unpartitioned Legendre EPBM can be up to roughly twice as expensive as EPIRK43s and EAB methods and a factor of  $q/2$  times less expensive than an ESDC method with  $q$  nodes. Finally, we also note that for unpartitioned Legendre EPBMs it is possible to parallelize the right-hand-side evaluations. However, for our test problems the cost of right-hand-side evaluations of  $R(\mathbf{y})$  is negligible compared to that of the Krylov projections, hence we always evaluate the right hand side in serial.

Similar to our partitioned experiments we see that high-order ESDC methods have the best stability and error constants; however, their high computational cost ultimately renders them less efficient. On the other hand, EPBMs possess improved stability and accuracy compared to EAB schemes and were overall the most efficient methods at each order of accuracy. Among, the nonpolynomial integrators, the fourth-order EPIRK method outperformed all other EAB and ESDC integrators except at very high precision, where eighth-order EAB becomes more efficient.

Unlike for the partitioned experiments, we see that higher-order unpartitioned EPBMs are more efficient than low-order EPBMs only if high accuracy is required. In particular it would be wasteful to always use a high-order integrator since it will require a substantially finer stepsize (and therefore more computational work) than a low-order method to remain stable.

Finally we note that it is trivial to construct methods that only require a projection with  $\tau = h$  per timestep. For example, we obtain this property by choosing Chebyshev nodes and PMFC<sub>1</sub> with endpoints  $b_j = 1$ . It is also simple to apply parallelism to speed up the computation of methods that use different endpoints  $b_j$  or polynomials  $L_R^{[j]}(\tau)$  for computing each output. Though such methods will require  $q$  projections per timestep, each projection can now be computed in parallel to nullify the additional cost.

**7. Summary and conclusions.** We presented an extension of the polynomial framework that incorporates exponential integration. To achieve this we combined the classical ODE polynomial with the integrating factor method to introduce the polynomial  $\varphi$ -expansions that form the basis of the new methods. By utilizing polynomial  $\varphi$ -expansions it is possible to construct many families of new exponential integrators including those with desirable properties such as parallelism and high order.

After introducing the framework we demonstrated its potential by presenting several general construction strategies for EPBMs and by deriving a new class of parallel EPBMs that use Legendre points. Our numerical experiments demonstrate the potential of these new parallel methods for both partitioned and unpartitioned problems. Based on our results it appears that Legendre EPBMs can provide significant computational savings compared to current state-of-the-art methods if they can be efficiently parallelized.

The generality of the exponential polynomial framework creates many opportunities for additional developments that we plan to address in future work. In particular we will investigate the construction of exponential polynomial general linear methods that can exploit parallelism on a larger scale.

**Appendix A. Coefficients for Adams polynomial  $\varphi$ -expansion.** Here we describe how to rewrite the Adams  $\varphi$ -expansion from Definition 3.4 in terms of the values in an exponential ODE dataset. In short, one must rewrite  $L_y(b)$  and  $L_N^{(k)}(b)$  in terms of the values and derivatives that were used to construct the Lagrange interpolating polynomials  $L_y(\tau)$  and  $L_N(\tau)$ . In other words, we must find the coefficients  $a_j$  and  $c_{\nu,j}$  so that

$$(A.1) \quad L_y(b) = \sum_{j=1}^w a_j \mathbf{y}_j \quad \text{and} \quad L_N^{(\nu)}(b) = \sum_{j=1}^w c_{\nu,j} r \mathbf{N}_j.$$

If the solution value  $\mathbf{y}_k$  is not used to form  $L_y(\tau)$ , then  $a_k$  is zero. All the nonzero  $a_j$  are finite difference weights for computing the zeroth derivative at  $\tau = b$ , using data at the nodes of the Lagrange polynomial  $L_y(\tau)$ .

Similarly, if the derivative component value  $r \mathbf{N}_k$  is not used to form  $L_N(\tau)$ , then  $c_{\nu,k} = 0$ , and all the nonzero  $c_{\nu,j}$  are finite difference weights for computing the  $\nu$ th derivative at  $\tau = b$ , using data at the nodes of the Lagrange polynomials  $L_N(\tau)$ .

The direct procedure for computing the finite difference weights is as follows. Suppose that  $L(x)$  is a Lagrange polynomial with nodes  $x_j$  and data  $\mathbf{g}_j$  for  $j = 1, \dots, l$ . The finite difference weights  $w_{\nu,j}$  for computing the  $\nu$ th derivative of  $L(x)$  at  $x = x_0$  (i.e.,  $L^{(\nu)}(x_0) = \sum_{j=1}^l w_{\nu,j} \mathbf{g}_j$ ) are  $d_{\nu,j} = \nu! \mathbf{V}_{\nu+1,j}^{-1}$ , where  $\mathbf{V}$  is the  $l \times l$  Vandermonde matrix with entries  $\mathbf{V}_{i,j} = (x_i - x_0)^{j-1}$ . This direct procedure can be used to obtain the weights; however, since Vandermonde matrices can be ill-conditioned it is advantageous to use the fast, stable algorithm developed by Fornberg for computing finite difference weights [14, 15].

**Appendix B. Method coefficients.** The general form for a Legendre method from Table 4.2 with  $q$  nodes is

$$(B.1) \quad y_j^{[n+1]} = \varphi_0(r\eta_j \mathbf{L}) y_1^{[n]} + r \sum_{k=1}^{q-1} \eta_j^k \varphi_k(r\eta_j \mathbf{L}) \mathbf{v}_j, \quad j = 1, \dots, q,$$

where  $\eta_j = z_j + \alpha + 1$  and  $\mathbf{N}_j = N(z_j, \mathbf{y}(z_j))$ . The vectors  $\mathbf{v}_j$  can be derived by computing the derivatives of the Lagrange polynomial

$$(B.2) \quad L(\tau) = \sum_{j=1}^{q-1} \ell_j(\tau) \mathbf{N}_j, \quad \ell_j(\tau) = \prod_{\substack{l=1 \\ l \neq j}}^{q-1} \frac{\tau - x_l}{x_j - x_l},$$

at the point  $\tau = -1$ , where  $x_j$  is the  $j$ th zero of the  $q$ th Legendre polynomial  $P_q(x)$ . Below we provide several formulas for the  $\mathbf{v}_j$  if  $q = 2, 3, 4$ .

Legendre method:  $q = 2$ ,  $z_j = \{-1, 0\}$ ,

$$\mathbf{v}_1 = \mathbf{N}_2.$$



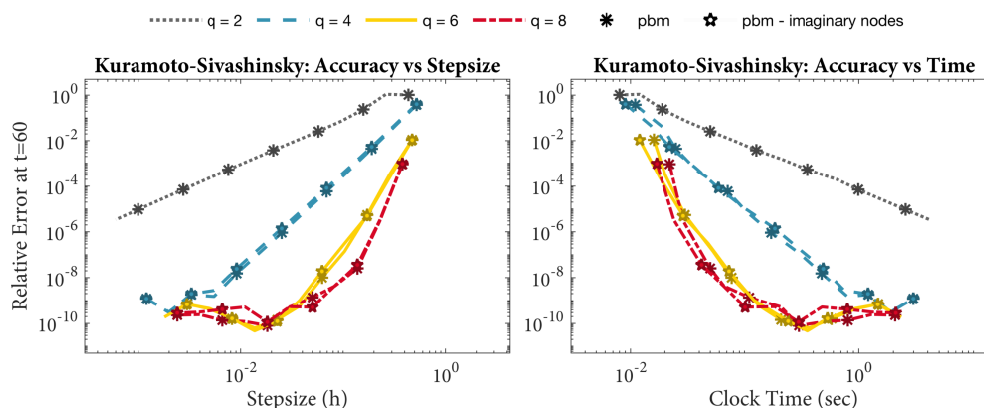


FIG. C.1. Comparison between methods with real-valued Legendre nodes and imaginary equispaced nodes on the KS equation.

Legendre method:  $q = 3$ ,  $z_j = \left\{-1, -\sqrt{\frac{1}{3}}, \sqrt{\frac{1}{3}}\right\}$ ,

$$\mathbf{v}_1 = \frac{(1 + \sqrt{3})\mathbf{N}_2}{2} + \frac{(1 - \sqrt{3})\mathbf{N}_3}{2},$$

$$\mathbf{v}_2 = \frac{\sqrt{3}\mathbf{N}_3}{2} - \frac{\sqrt{3}\mathbf{N}_2}{2}.$$

Legendre method:  $q = 4$ ,  $\{z_j\} = \left\{-1, -\sqrt{\frac{3}{5}}, 0, \sqrt{\frac{3}{5}}\right\}$ ,

$$\mathbf{v}_1 = \frac{(5 + \sqrt{15})\mathbf{N}_2}{6} - \frac{2\mathbf{N}_3}{3} - \frac{(\sqrt{15} - 5)\mathbf{N}_4}{6},$$

$$\mathbf{v}_2 = \frac{(-10 - \sqrt{15})\mathbf{N}_2}{6} + \frac{10\mathbf{N}_2}{3} + \frac{(\sqrt{15} - 10)\mathbf{N}_4}{6},$$

$$\mathbf{v}_2 = \frac{5\mathbf{N}_2}{3} - \frac{10\mathbf{N}_3}{3} + \frac{5\mathbf{N}_4}{3}.$$

### Appendix C. Numerical experiment for EPBMs with complex nodes.

In Figure C.1, we briefly demonstrate the ability to use EPBMs with complex nodes by solving the Kuramoto equation (6.1) using the EPBMs with Legendre nodes from Table 4.2 and the EPBMs with imaginary nodes labeled as “Method A” from Table 4.4. Both methods are run with  $\alpha = 2$ . Though they offer no computational advantage, the methods with imaginary nodes retain stability and have similar error properties to the real-valued methods on this equation.

**Acknowledgments.** I would like to thank Randall LeVeque for encouraging me to develop polynomial integrators during my graduate studies and for his comments and guidance through the course of this project. I would also like to thank Mayya Tokman for her helpful comments on drafts of this work.

### REFERENCES

- [1] U. M. ASCHER, S. J. RUUTH, AND B. T. WETTON, *Implicit-explicit methods for time-dependent partial differential equations*, SIAM J. Numer. Anal., 32 (1995), pp. 797–823.
- [2] 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.

- [3] J. C. BUTCHER, *General linear methods*, Acta Numer., 15 (2006), pp. 157–256.
- [4] T. BUVOLI, *Polynomial-Based Methods for Time-Integration*, Ph.D. thesis, University of Washington, 2018.
- [5] T. BUVOLI, *A class of exponential integrators based on spectral deferred correction*, SIAM J. Sci. Comput., 42 (2020), pp. A1–A27.
- [6] T. BUVOLI, *Codebase for Exponential Block Methods*, <http://doi.org/10.5281/zenodo.4500185>, (2021).
- [7] T. BUVOLI AND M. TOKMAN, *Constructing new time integrators using interpolating polynomials*, SIAM J. Sci. Comput., 41 (2019), pp. A2911–A2937.
- [8] M. CALIARI, P. KANDOLF, A. OSTERMANN, AND S. RAINER, *Comparison of software for computing the action of the matrix exponential*, BIT, 54 (2014), pp. 113–128.
- [9] M. CALIARI, P. KANDOLF, A. OSTERMANN, AND S. RAINER, *The Leja method revisited: Backward error analysis for the matrix exponential*, SIAM J. Sci. Comput., 38 (2016), pp. A1639–A1661.
- [10] A. CARDONE, Z. JACKIEWICZ, A. SANDU, AND H. ZHANG, *Construction of highly stable implicit-explicit general linear methods*, in Proceedings of the 10th AIMS Conference on Dynamical Systems, Differential Equations and Applications, Madrid, 2015.
- [11] A. J. CHRISTLIEB, C. B. MACDONALD, AND B. W. ONG, *Parallel high-order integrators*, SIAM J. Sci. Comput., 32 (2010), pp. 818–835.
- [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] B. FORNBERG, *Generation of finite difference formulas on arbitrarily spaced grids*, Math. Comput., 51 (1988), pp. 699–706.
- [15] B. FORNBERG, *Calculation of weights in finite difference formulas*, SIAM Rev., 40 (1998), pp. 685–691.
- [16] M. J. GANDER, *50 years of time parallel time integration*, in Multiple Shooting and Time Domain Decomposition Methods, Springer, New York, 2015, pp. 69–113.
- [17] S. GAUDREAU, G. RAINWATER, AND M. TOKMAN, *KIOPS: A fast adaptive Krylov subspace solver for exponential integrators*, J. Comput. Phys., 372 (2018), pp. 236–255.
- [18] C. W. GEAR, *Parallel methods for ordinary differential equations*, Calcolo, 25 (1988), pp. 1–20.
- [19] I. GROOMS AND K. JULIEN, *Linearly implicit methods for nonlinear PDEs with linear dispersion and dissipation*, J. Comput. Phys., 230 (2011), pp. 3630–3650.
- [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] N. J. HIGHAM AND E. HOPKINS, *A Catalogue of Software for Matrix Functions. Version 3.0*, 2020, <http://eprints.maths.manchester.ac.uk/2754/>.
- [22] M. HOCHBRUCK AND A. OSTERMANN, *Explicit exponential Runge–Kutta methods for semilinear parabolic problems*, SIAM J. Numer. Anal., 43 (2005), pp. 1069–1090.
- [23] M. HOCHBRUCK AND A. OSTERMANN, *Exponential Runge–Kutta methods for parabolic problems*, Appl. Numer. Math., 53 (2005), pp. 323–339.
- [24] M. HOCHBRUCK AND A. OSTERMANN, *Exponential integrators*, Acta Numer., 19 (2010), pp. 209–286.
- [25] G. IZZO AND Z. JACKIEWICZ, *Highly stable implicit–explicit Runge–Kutta methods*, Appl. Numer. Math., 113 (2017), pp. 71–92.
- [26] Z. JACKIEWICZ AND H. MITTELMANN, *Construction of IMEX DIMSIMs of high order and stage order*, Appl. Numer. Math., 121 (2017), pp. 234–248.
- [27] A. KASSAM AND L. TREFETHEN, *Fourth-order time stepping for stiff PDEs*, SIAM J. Sci. Comput., 26 (2005), pp. 1214–1233.
- [28] S. KOIKARI, *Rooted tree analysis of Runge–Kutta methods with exact treatment of linear terms*, J. Comput. Appl. Math., 177 (2005), pp. 427–453.
- [29] S. KROGSTAD, *Generalized integrating factor methods for stiff PDEs*, J. Comput. Phys., 203 (2005), pp. 72–88.
- [30] 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.
- [31] 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.
- [32] J. LOFFELD AND M. TOKMAN, *Implementation of parallel adaptive-Krylov exponential solvers for stiff problems*, SIAM J. Sci. Comput., 36 (2014), pp. C591–C616.
- [33] V. T. LUAN AND A. OSTERMANN, *Parallel exponential Rosenbrock methods*, Comput. Math. Appl., 71 (2016), pp. 1137–1150.

- [34] D. L. MICHELS, V. T. LUAN, AND M. TOKMAN, *A stiffly accurate integrator for elastodynamic problems*, ACM Trans. Graphics, 36 (2017), p. 116.
- [35] H. MONTANELLI AND N. BOOTLAND, *Solving Periodic Semilinear Stiff PDEs in 1D, 2D and 3D with Exponential Integrators*, preprint, arXiv:1604.08900, 2016.
- [36] J. NIESEN AND W. M. WRIGHT, *A Krylov Subspace Method for Option Pricing*, <http://dx.doi.org/10.2139/ssrn.1799124>, 2011.
- [37] J. NIESEN AND W. M. WRIGHT, *Algorithm 919: A Krylov subspace algorithm for evaluating the  $\varphi$ -functions appearing in exponential integrators*, ACM Trans. Math. Software, 38 (2012), pp. 22:1–22:19.
- [38] A. OSTERMANN, M. THALHAMMER, AND W. M. WRIGHT, *A class of explicit exponential general linear methods*, BIT, 46 (2006), pp. 409–431.
- [39] 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.
- [40] G. RAINWATER AND M. TOKMAN, *A new approach to constructing efficient stiffly accurate EPIRK methods*, J. Comput. Phys., 323 (2016), pp. 283–309.
- [41] A. SANDU AND M. GÜNTHER, *A generalized-structure approach to additive Runge–Kutta methods*, SIAM J. Numer. Anal., 53 (2015), pp. 17–42.
- [42] 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), p. e2220.
- [43] 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.
- [44] L. F. SHAMPINE AND H. WATTS, *Block implicit one-step methods*, Math. Comput., 23 (1969), pp. 731–740.
- [45] D. SHEEN, I. SLOAN, AND V. THOMÉE, *A parallel method for time-discretization of parabolic problems based on contour integral representation and quadrature*, Math. Comput., 69 (2000), pp. 177–195.
- [46] E. SIMBAWA, P. C. MATTHEWS, AND S. M. COX, *Nikolaevskiy equation with dispersion*, Phys. Rev. E, 81 (2010), 036220.
- [47] M. TOKMAN, *Efficient integration of large stiff systems of ODEs with exponential propagation iterative (EPI) methods*, J. Comput. Phys., 213 (2006), pp. 748–776.
- [48] M. TOKMAN, *A new class of exponential propagation iterative methods of Runge–Kutta type (EPIRK)*, J. Comput. Phys., 230 (2011), pp. 8762–8778.
- [49] 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.