SIAM J. NUMER. ANAL.
Vol. 58, No. 6, pp. 3197–3225

TWO-DERIVATIVE ERROR INHIBITING SCHEMES AND ENHANCED ERROR INHIBITING SCHEMES*

ADI DITKOWSKI†, SIGAL GOTTLIEB‡, AND ZACHARY J. GRANT§

Abstract. High order methods are often desired for the evolution of ordinary differential equations, in particular those arising from the semidiscretization of partial differential equations. In prior work we investigated the interplay between the local truncation error and the global error to construct *error inhibiting* general linear methods (GLMs) that control the accumulation of the local truncation error over time. We defined sufficient conditions that allow us to postprocess the final solution and obtain a solution that is two orders of accuracy higher than expected from truncation error analysis alone. In this work we extend this theory to the class of two-derivative GLMs. We define sufficient conditions that control the growth of the error so that the solution is one order higher than expected from truncation error analysis, and furthermore, define the construction of a simple postprocessor that will extract an additional order of accuracy. Using these conditions as constraints, we develop an optimization code that enables us to find explicit two-derivative methods up to eighth order that have favorable stability regions, explicit strong stability preserving (SSP) methods up to seventh order, and A-stable implicit methods up to fifth order. We numerically verify the order of convergence of a selection of these methods, and the total variation diminishing performance of some of the SSP methods. We confirm that the methods found perform as predicted by the theory developed herein.

Key words. error inhibiting methods, two-derivative, time-stepping, postprocessing, superconvergent

AMS subject classifications. 65, 65L06, 65L20

DOI. 10.1137/19M1306129

1. Introduction.

In this paper we consider numerical solvers for ordinary differential equations (ODEs). Without loss of generality [19], we focus our attention on the autonomous ODE

$$u_t = F(u) \quad \text{for} \quad t \geq 0, \qquad u(t_0) = u_0. \tag{1.1}$$

The canonical numerical method for this problem is the forward Euler method

$$v_{n+1} = v_n + \Delta t F(v_n) \ ,$$

where v_n approximates the exact solution at time t_n and we let $v_0 = u_0$. The forward Euler method has local truncation error LTE^n and approximation error τ^n at any

*Received by the editors December 11, 2019; accepted for publication (in revised form) July 31, 2020; published electronically November 9, 2020.
https://doi.org/10.1137/19M1306129
Funding: The work of the authors was supported by the AFOSR grant FA9550-18-1-0383 and the ONR-DURIP grant N00014-18-1-2255. The work of the authors was partially supported by the U.S. Department of Energy, Office of Advanced Scientific Computing Research contract DE-AC05-00OR22725 and was performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

†School of Mathematical Sciences, Tel Aviv University, Tel Aviv 69978, Israel (adid@post.tau.ac.il).
‡Mathematics Department, University of Massachusetts Dartmouth, North Dartmouth, MA 02747 USA (sgottlieb@umassd.edu).
§Department of Computational and Applied Mathematics, Oak Ridge National Laboratory, Oak Ridge, TN 37830 USA (grantzj@ornl.gov).

given time t_n defined by

$$\tau^n = \Delta t LTE^n = u(t_{n-1}) + \Delta t F(u(t_{n-1})) - u(t_n) \approx O(\Delta t^2),$$

and it produces a global error which is first order accurate, $e^n = v_n - u(t_n) \approx O(\Delta t)$. We note the relationship between the local truncation error and the global error.

To develop methods that have higher order, we can add steps to define a linear multistep method [6], use multiple stages as in a Runge–Kutta method [6], or include higher derivatives as we do in Taylor series methods [6]. We can also combine the approaches above: the combination of multiple steps and stages is commonly seen, as in the general linear methods described in [5, 19], and multiple derivatives and stages have also been used [7, 8, 15, 28, 30, 34, 38]. In all of these cases the goal is to increase the order of the local truncation error and therefore of the global error. All of these approaches typically produce methods that have global errors that are of the same order as their local truncation errors.

The Dahlquist Equivalence theorem [45] states that any zero-stable, consistent linear multistep method with local truncation error $LTE^n = O(\Delta t^p)$ will have global error $O(\Delta t^p)$, provided that the solution has at least $(p + 1)$ smooth derivatives. This behavior is so expected for all one-step and multistep methods that we typically define the order of a stable numerical method solely by the order conditions derived by Taylor series analysis of the local truncation error. This relationship between the order of the local truncation error and global error is also seen in finite difference schemes for partial differential equations (PDEs) [17, 36]. Work over the past decade reminds us that, while for a stable scheme the global error must be *at least* of the same order as its local truncation error, it may in fact be of higher order.

There are several approaches to devise schemes that have global errors that are *higher order* than predicted by the local truncation errors. For both Runge–Kutta methods and general linear methods (GLMs) the idea of starting the time-stepping with a method $\mathcal{S}$ known as the starting method, then running the GLM (or Runge–Kutta) denoted $\mathcal{M}$ up to the final time and extracting a higher order solution using a finishing method $\mathcal{F}$, has shown that the overall scheme has global errors that are *higher order* than expected from the truncation error analysis of $\mathcal{M}$ alone [4, 44, 10, 31]. Understanding this phenomenon requires examining the underlying one-step method comprised of the composition of the three methods [44, 31]. This makes analysis of the methods simple, but it does not easily show how to construct such methods.

Another approach is the quasi-consistency or error inhibiting approach. This approach involves conditions on the coefficients that inhibit the growth of the truncation errors over time-evolution; under these conditions certain classes of methods can attain one order higher than expected from local truncation error analysis [24, 50, 11]. We call this approach the error inhibiting scheme (EIS) approach and review it in subsection 1.1. *The first aim of this paper is to extend the EIS approach to two-derivative peer methods, to develop efficient methods that provide a solution that is one order of accuracy higher than expected from truncation error analysis alone.*

An enhancement of the EIS approach, presented in [12], involves the addition of two conditions on the coefficients of the method. These conditions cause the accumulation of the errors to be controlled so that the leading error term at the end of the simulation can be written exactly. Such conditions that allow us to determine the exact form of the error term have been explored in [25, 49, 26] and leveraged for error estimation. The conditions we presented in [12] are less restrictive than those presented previously, and were used to design a postprocessor that extracts a numerical

solution that is *two orders higher* than expected from local truncation error analysis. This approach is called *EIS+ postprocessing* and is reviewed in subsection 1.2. *The second aim of this paper is to extend the EIS+ postprocessing approach to two-derivative peer methods, to develop efficient method/postprocessor pairs, that obtain an additional order of accuracy beyond that of EIS methods.*

To the best of our knowledge, none of these approaches have ever been developed or implemented in the context of multiderivative methods. *The innovation in this work is the extension of the theory described in* [12] *to two-derivative peer methods, and the development of novel EIS and EIS+postprocessing two-derivative methods (presented in section* 3). The stability and accuracy properties of the novel two-derivative methods are compared to existing methods, and are tested for confirmation of their accuracy properties.

1.1. Review of EIS peer methods. In 2009 there appeared two papers that showed how to use a property called quasi-consistency to design methods that have a solution of order $p+1$ although their truncation errors are of order p. Kulikov [24] studied Nordsiek methods, and Weiner et al. [50] studied explicit two-step peer methods, and followed along the lines of the quasi-consistency theory first introduced by Skeel in 1976 [40]. In 2017, Ditkowski and Gottlieb [11] used a different approach to derive similar sufficient conditions, for a family of peer methods of any number of steps and stages, under which one can control the accumulation of the local truncation error over time evolution and produce methods that are one order higher than expected from the truncation error alone. We named such methods *error inhibiting schemes* (EIS). In [12] we extended the EIS approach to a broader class of explicit and implicit peer methods, and also developed an enhancement of this approach which we review in subsection 1.2. In this subsection we review the EIS approach, using the notation in [12].

Consider a peer method written in the form

$$V^{n+1} = \mathbf{D}V^n + \Delta t\mathbf{A}F(V^n) + \Delta t\mathbf{R}F(V^{n+1}), \tag{1.2a}$$

where V^n is a vector of length s that contains the numerical solution at times $(t_n + c_j\Delta t)$ for $j = 1, \ldots, s$:

$$V^n = \left(v(t_n + c_1\Delta t), v(t_n + c_2\Delta t), \ldots, v(t_n + c_s\Delta t)\right)^T. \tag{1.2b}$$

The function $F(V^n)$ is defined as the componentwise function evaluation on the vector V^n:

$$F(V^n) = \left(F(v(t_n + c_1\Delta t)), F(v(t_n + c_2\Delta t)), \ldots, F(v(t_n + c_s\Delta t))\right)^T, \tag{1.2c}$$

where F is given in (1.1). The corresponding exact solution of the ODE (1.1) is given by

$$U^n = \left(u(t_n + c_1\Delta t), u(t_n + c_2\Delta t), \ldots, u(t_n + c_s\Delta t)\right)^T, \tag{1.2d}$$

and $F(U^n)$ is the componentwise function evaluation on the vector U^n.

As before, we define the normalized local truncation error LTE^n and approximation error $\boldsymbol{\tau}^n$ by

$$\Delta t\ LTE^n = \boldsymbol{\tau}^n = \left[\mathbf{D}U^{n-1} + \Delta t\mathbf{A}F(U^{n-1}) + \Delta t\mathbf{R}F(U^n)\right] - U^n, \tag{1.2e}$$

where

$$\boldsymbol{\tau}^n = \sum_{j=0}^{\infty} \boldsymbol{\tau}_j^n \Delta t^j = \sum_{j=0}^{\infty} \boldsymbol{\tau}_j \Delta t^j \left. \frac{d^j u}{dt^j} \right|_{t=t_n}, \tag{1.2f}$$

where $\left. \frac{d^j u}{dt^j} \right|_{t=t_n}$ is the jth derivative of the solution at time $t = t_n$, and Δt^j is the time-step raised to the power j. The vectors $\boldsymbol{\tau}_j$ are given by

$$\boldsymbol{\tau}_0 = (\mathbf{D} - \mathbf{I})\, \mathbb{1}, \tag{1.2g}$$

$$\boldsymbol{\tau}_j = \frac{1}{(j-1)!} \left(\frac{1}{j} \mathbf{D} (\mathbf{c} - \mathbb{1})^j + \mathbf{A} (\mathbf{c} - \mathbb{1})^{j-1} + \mathbf{R} \mathbf{c}^{j-1} - \frac{1}{j} \mathbf{c}^j \right) \quad \text{for } j = 1,2, \ldots. \tag{1.2h}$$

Here, $\mathbf{c}$ is the vector of abscissas $(c_1, c_2, \ldots, c_s)^T$ and $\mathbb{1} = (1, 1, \ldots, 1)^T$ is the vector of ones. We understand terms such as $\mathbf{c}^j$ as componentwise exponentiation: $\mathbf{c}^j = (c_1^j, c_2^j, \ldots, c_s^j)^T$.

If the truncation error vectors $\boldsymbol{\tau}_j$ satisfy the order conditions

$$\boldsymbol{\tau}_j = 0 \quad \text{for} \quad j = 0, \ldots, p, \tag{1.2i}$$

then we have local truncation error $LTE^n = O(\Delta t^p)$ and (correspondingly) approximation error $\boldsymbol{\tau}^n = O(\Delta t^{p+1})$. In such a case we typically observe that the global error at any time t^n is also of order $O(\Delta t^{p+1})$:

$$E^n = V^n - U^n = O(\Delta t^p).$$

However, in [24, 50, 11] it was shown that if the coefficients in $\mathbf{D}$, $\mathbf{A}$, and $\mathbf{R}$ satisfy the *additional condition*

$$\mathbf{D} \boldsymbol{\tau}_{p+1} = 0, \tag{1.2j}$$

then the method will have the global error

$$E^n = V^n - U^n = O(\Delta t^{p+1}).$$

This additional condition on the coefficients, (1.2j), allows us to design methods that have solution of order $p + 1$, although their truncation errors are of order p in (1.2i). Such methods were developed in [24, 50, 11, 12] for classes of explicit and implicit GLMs. These EIS methods were shown to have good stability properties, and be more efficient than typical (non-EIS) methods in their class. This is due to the fact that satisfying the order conditions (1.2i) up to order p, plus the EIS condition (1.2j), can be less restrictive than satisfying the order conditions (1.2i) up to order $p + 1$. We note that a similar theory was developed for implicit-explicit methods in [43].

1.2. Review of enhanced EIS peer-methods with postprocessing. The EIS methods described in subsection 1.1 allow us to inhibit the growth of the error over the simulation so that we obtain an order $p + 1$ solution even though truncation error analysis predicts order p. However, we wish to control the growth of the error even further, so that we can identify the *exact form* of the leading global error term at any given time. In this approach, we require the method (1.2a) to satisfy the following:

- The order conditions (1.2i) up to order p.
- The EIS condition (1.2j).
- The EIS+ conditions:

$$\mathbf{D}\boldsymbol{\tau}_{p+2} = 0, \tag{1.2k}$$

$$\mathbf{D}(\mathbf{A}+\mathbf{R})\boldsymbol{\tau}_{p+1} = 0. \tag{1.2l}$$

In [12] we showed under these conditions the solution produced by the method is of order $p+1$, and we can compute the *precise form* of the leading term in the global error. This allows an appropriately defined postprocessor to extract a numerical solution of order Δt^{p+2}.

In [12] we developed a number of implicit and explicit peer methods that satisfy the order conditions up to order p, and the conditions (1.2j), (1.2k), (1.2l), and we designed a postprocessor for each of these methods. We showed that such methods provide an efficient alternative to typical peer methods, particularly as the cost of postprocessing is negligible. Such methods, presented in [12], are denoted EIS+ methods, and the associated postprocessor is given with each method.

Note that in [25, 49, 26], conditions similar to (1.2k) and (1.2l) were presented that allow the leading error term to be leveraged for global error estimation. However, condition (1.2l) is less restrictive than the conditions used by [25, 49, 26].

Postprocessing has long been used to extract more accurate information from a numerical solution resulting from a variety of spatial discretizations. Some examples include Gegenbauer postprocessing for spectral methods [39], smoothness increasing accuracy conserving (SIAC) filtering for discontinuous Galerkin [37], postprocessing to enhance the solutions for the discontinuous Galerkin method [9], and a delta function approach for spectral element methods [20, 51]. This is usually accomplished by analyzing the existing numerical solution and recognizing that higher order information is buried in it, and designing sophisticated mathematical approaches to extract this higher order information.

The postprocessing approach we used for the peer methods in [12] is reminiscent of finishing methods commonly used for GLMs [4, 44]. However, there is a philosophical difference here: by design, a finishing method "is a L-Lipschitz map which undoes the work of a starting method $\mathcal{S}$, at least up to order $O(h^{p+1})$" [44]. Typically a k-step GLM is given an initial condition u_0 from which we need to build the initial vector of approximations V^0. A finishing method is one "which extracts the numerical approximation y_n from Y_n" [18], i.e., a finishing method extracts the numerical solution which approximates u^N at the final time t^N: this is often done trivially, by pulling out the correct element of V^n. Sometimes, this is done in a more sophisticated way which attains higher order by complementing the effect of the starting method $\mathcal{S}$. While we can write our postprocessor as a finishing method, the analysis of the underlying one-step method is not sufficient to obtain $O(\Delta t^{p+2})$ because it does not take into account the error inhibiting property; rather, we must examine the composite method over a large number of steps [2]. The postprocesser is only able to remove an $O(\Delta t^{p+1})$ error term because its precise form is known. In fact, the postprocessing method may be more appropriately thought of as related to "smoothing or to passive extrapolation" [3] that removes a known final-time error term of a particular form while reproducing a polynomial up to order $p+1$.

1.3. Two-derivative GLMs. Multiderivative Runge–Kutta methods, particularly two-derivative methods, were first considered in [32, 48, 46, 41, 42, 21, 22, 28, 34, 7], and later explored for use with PDEs [38, 47, 27, 35, 13]. Recent interest in ex-

ploring the strong stability properties of multiderivative Runge–Kutta methods, also known as multistage multiderivative methods, is evident in [8, 15, 30]. When using these multiderivative methods in the context of time evolution for PDEs, it would seem that the computation of the derivative term $\dot{F}$ should follow directly from the definition of F, and take the form $\dot{F} = F(u)_t = F_u u_t = F_u F$. However, obtaining F_u may be computationally prohibitive, or we may wish for other reasons to compute F_t using a different spatial discretization than the one used to compute F. Instead, information from the underlying PDE is typically used to replace the time derivatives by the spatial derivatives, and discretize these in space. The concern about the spatial errors that enter from this process and their impact on the time-error was addressed in [8] where it was shown both theoretically and numerically that it is not necessary to compute F_t exactly, and that replacing the temporal derivatives by spatial derivatives and discretizing each of these independently to high order accuracy, does not destroy the temporal accuracy in multiderivative methods.

More recently, the multiderivative approach was expanded to GLMs. A 2015 work by Okuonghae and Ikhile [33] discussed two- and three-derivative methods with multiple stages and steps. In 2019, Moradi, Farzi, and Abdi [29] presented explicit strong stability preserving (SSP) two-derivative GLMs. These two-derivative GLMs are similar to the explicit SSP ones in this work, but our methods have larger SSP coefficients. However, in those papers, the order of the two-derivative methods is as expected from truncation error analysis alone. In contrast, we extend the approach developed in [12] to two-derivative methods to obtain a numerical solution that is two orders higher than expected from the truncation error analysis.

In this work we consider two-derivative peer methods that extend the form (1.2a) above to include a second derivative term:

$$V^{n+1} = \mathbf{D}V^n + \Delta t \mathbf{A} F(V^n) + \Delta t \mathbf{R} F(V^{n+1}) + \Delta t^2 \hat{\mathbf{A}} \dot{F}(V^n) + \Delta t^2 \hat{\mathbf{R}} \dot{F}(V^{n+1}), \tag{1.3}$$

where most of the terms as defined above, and the function $\dot{F}(V^n)$ is

$$\dot{F}(V^n) = \left(\frac{dF}{dt}(v(t_n + c_1 \Delta t)), \frac{dF}{dt}(v(t_n + c_2 \Delta t)), \ldots, \frac{dF}{dt}(v(t_n + c_s \Delta t)) \right)^T. \tag{1.4}$$

For convenience, we select $c_1 = 0$ so that the first element in the vector V^n approximates the solution at time t_n, and the abscissas are nondecreasing $c_1 \le c_2 \le \cdots \le c_s$. To initialize these methods, we define the first element in the initial solution vector $V_1^0 = u(t_0)$ and the remaining elements $v(t_0 + c_j \Delta t)$ are computed using some highly accurate method and small time-step, so they approximate $u(t_0 + c_j \Delta t)$ to very high accuracy.

As before, we define the *global error* as the difference between the vectors of the exact and the numerical solutions at some time t_n,

$$E^n = V^n - U^n. \tag{1.5}$$

Methods of the form (1.3) have a local truncation error LTE^n at time t_n that is related to the approximation error $\boldsymbol{\tau}^n$ by

$$\Delta t \; LTE^n = \boldsymbol{\tau}^n,$$

where

$$\boldsymbol{\tau}^n = \Big[\mathbf{D}U^{n-1} + \Delta t \mathbf{A} F(U^{n-1}) + \Delta t \mathbf{R} F(U^n) + \Delta t^2 \hat{\mathbf{A}} \dot{F}(U^{n-1}) + \Delta t^2 \hat{\mathbf{R}} \dot{F}(U^n) \Big] - U^n. \tag{1.6}$$

Using Taylor series analysis, we see that

$$\boldsymbol{\tau}^n = \sum_{j=0}^{\infty} \boldsymbol{\tau}_j^n \Delta t^j = \sum_{j=0}^{\infty} \boldsymbol{\tau}_j \Delta t^j \left. \frac{d^j u}{dt^j} \right|_{t=t_n}, \tag{1.7}$$

where the truncation error vectors $\boldsymbol{\tau}_j$ have the form

$$\boldsymbol{\tau}_0 = (\mathbf{D} - \mathbf{I})\,\mathbb{1}, \tag{1.8a}$$

$$\boldsymbol{\tau}_j = \frac{1}{(j-1)!}\left(\frac{1}{j}\mathbf{D}(\mathbf{c}-\mathbb{1})^j + \mathbf{A}(\mathbf{c}-\mathbb{1})^{j-1} + (j-1)\hat{\mathbf{A}}(\mathbf{c}-\mathbb{1})^{j-2} + \mathbf{R}\mathbf{c}^{j-1} + (j-1)\hat{\mathbf{R}}\mathbf{c}^{j-2} - \frac{1}{j}\mathbf{c}^j\right) \quad \text{for } j = 1, 2, \ldots. \tag{1.8b}$$

Here, $\mathbf{c} = (c_1, c_2, \ldots, c_s)^T$ is the vector of abscissas and $\mathbb{1} = (1, 1, \ldots, 1)^T$ is the vector of ones, and the terms $\mathbf{c}^j$ are understood componentwise $\mathbf{c}^j = (c_1^j, c_2^j, \ldots, c_s^j)^T$. For a method to have order p it must satisfy the order conditions

$$\boldsymbol{\tau}_j = 0, \quad j \le p.$$

Note that the form (1.3) includes both explicit and implicit schemes, as V^{n+1} appears on both sides of the equation. However, if $\mathbf{R}$ and $\hat{\mathbf{R}}$ are both strictly lower triangular, the scheme is explicit. We are only interested in zero-stable methods. A sufficient condition for this is that the coefficient matrix $\mathbf{D}$ is a rank one matrix that has row sum one (i.e., satisfies the consistency condition). For simplicity, we assume this to be the case in the remainder of this work.

1.4. Overview of this paper. *The first result of this paper* is an extension of this EIS theory to two-derivative peer methods (1.3). Under a condition similar to (1.2j) we can obtain two-derivative peer methods that are one order higher than predicted by their local truncation error. In section 2.2 we extend the error inhibiting approach to explicit and implicit peer methods with two derivatives.

The second result of this paper is an extension of the EIS+ conditions needed to obtain the exact form of the leading error term, and the related postprocessor, to two-derivative peer methods. We show in section 2 that under conditions of the form (1.2j), (1.2k), (1.2l) we can compute the exact form of the Δt^{p+1} term in the global error of two-derivative peer methods, and in section 2.3 we show how to construct a simple postprocessor that extracts a solution that is two orders higher than predicted by truncation error alone.

These two results are facilitated by the fact that the second derivative terms all enter with Δt^2 terms multiplying them, and so the results in [12] carry over with few significant changes, aside from the definition of the truncation error. These results are a mathematically straightforward extension of the results in [12] to methods of the form (1.3), but they are extremely powerful as they show how to construct methods that produce a numerical solution one or two orders higher than expected by truncation error alone. *The central result in this paper* is the application of the EIS and EIS+ postprocessing theory to design novel, efficient methods that produce a numerical solution one or two orders higher than expected by truncation error alone. We present a selection of these methods in section 3, where we compare the SSP methods of this type to other SSP two-derivative GLM methods in [29]. In section 4 we test a selection of these methods on numerical examples to demonstrate their enhanced accuracy properties.

2. Designing multiderivative error inhibiting schemes and enhanced EIS. In this section we show how to construct two-derivative GLMs of the form (1.3) that are error inhibiting and so produce a solution of order $p+1$ even though one would only expect order p from their truncation errors. We further show that under additional conditions we can express the exact form of the error at any time-level and define an associated postprocessor that allow us to recover order $p+2$ from a scheme with truncation errors that are only pth order accurate.

Note that while we generally consider that problem (1.1) is a system, in this section we consider only the case where u, and therefore $F_u^n = \frac{\partial F}{\partial u}\big|_{u(t_n)}$, are scalars. While all the results in this section are obtained for the scalar case, they are also valid for systems. The extension to the vector case is mathematically trivial (though messy) and was shown in [12].

2.1. Preliminaries. In this subsection we make some observations that will be useful in the remainder of this paper. In the following we assume sufficient smoothness of all the quantities such as $F, \dot{F}$, etc. Furthermore, we assume that the order conditions $\boldsymbol{\tau}_j = 0$ hold for all $j = 0, \ldots, p$, and that we are in the asymptotic regime so that the errors are small and we can say that $\|E^n\| \leq O(\Delta t^p) \ll 1$. Our first observation is taken from [12].

OBSERVATION 1. *We observe that*

$$F(U^n + E^n) = F(U^n) + F_u^n E^n + O(\Delta t) E^n, \tag{2.1}$$

where $F_u^n = \frac{\partial F}{\partial u}\big|_{u(t_n)}$.

The proof of this is given in [12].

OBSERVATION 2. *Given the smoothness of* $\dot{F}$*, we observe that*

$$\dot{F}(U^n + E^n) = \dot{F}(U^n) + \dot{F}_u^n E^n + O(\Delta t) E^n, \tag{2.2}$$

where $\dot{F}_u^n = \frac{\partial \dot{F}}{\partial u}\Big|_{u(t_n))}$.

Proof. The proof of this observation is exactly the same as above, with the assumption that $\dot{F}$ is smooth enough to be expanded. ∎

We use these observations to develop an equation that describes the growth of the error.

LEMMA 1. *Given a zero-stable method of the form* (1.3) *which satisfies the order conditions*

$$\boldsymbol{\tau}_j = 0 \quad \textit{for } j = 0, \ldots, p$$

and where the functions F *and* $\dot{F}$ *are smooth, the evolution of the error can be described by*

$$\left(I - \Delta t \mathbf{R} F_u^n + O(\Delta t^2)\right) E^{n+1} = \left(\mathbf{D} + \Delta t \mathbf{A} F_u^n + O(\Delta t^2)\right) E^n + \boldsymbol{\tau}^{n+1}. \tag{2.3}$$

Proof. Recall that

$$\boldsymbol{\tau}^{n+1} = \mathbf{D}U^n + \Delta t \mathbf{A} F(U^n) + \Delta t^2 \hat{\mathbf{A}} \dot{F}(U^n) + \Delta t \mathbf{R} F(U^{n+1}) + \Delta t^2 \hat{\mathbf{R}} \dot{F}(U^{n+1}) - U^{n+1}$$

and use this equation to subtract U^{n+1} from

$$V^{n+1} = \mathbf{D}V^n + \Delta t \mathbf{A} F(V^n) + \Delta t^2 \hat{\mathbf{A}} \dot{F}(V^n) + \Delta t \mathbf{R} F(V^{n+1}) + \Delta t^2 \hat{\mathbf{R}} \dot{F}(V^{n+1})$$

obtaining

$$\begin{aligned} V^{n+1} - U^{n+1} = \mathbf{D}V^n - \mathbf{D}U^n + \Delta t\mathbf{A}\left(F(V^n) - F(U^n)\right) \\ +\Delta t^2\hat{\mathbf{A}}\left(\dot{F}(V^n) - \dot{F}(U^n)\right) + \Delta t\mathbf{R}\left(F(V^{n+1}) - F(U^{n+1})\right) \\ +\Delta t^2\hat{\mathbf{R}}\left(\dot{F}(V^{n+1}) - \dot{F}(U^{n+1})\right) + \boldsymbol{\tau}^{n+1}. \end{aligned}$$

Using Observations 1 and 2 we have

$$\begin{aligned} E^{n+1} &= \mathbf{D}(U^n + E^n) - \mathbf{D}U^n + \Delta t\mathbf{A}\left(F(U^n + E^n) - F(U^n)\right) \\ &\quad +\Delta t^2\hat{\mathbf{A}}\left(\dot{F}(U^n + E^n) - \dot{F}(U^n)\right) + \Delta t\mathbf{R}\left(F(U^{n+1} + E^{n+1}) - F(U^{n+1})\right) \\ &\quad +\Delta t^2\hat{\mathbf{R}}\left(\dot{F}(U^{n+1} + E^{n+1}) - \dot{F}(U^{n+1})\right) + \boldsymbol{\tau}^{n+1} \\ &= \mathbf{D}E^n + \Delta t\left(\mathbf{A}F_u^n + O(\Delta t)\right)E^n + \Delta t^2\left(\dot{F}_u^n\hat{\mathbf{A}} + O(\Delta t)\right)E^n \\ &\quad +\Delta t\left(\mathbf{R}F_u^{n+1} + O(\Delta t)\right)E^{n+1} + \Delta t^2\left(\dot{F}_u^{n+1}\hat{\mathbf{R}} + O(\Delta t)\right)E^{n+1} + \boldsymbol{\tau}^{n+1}, \end{aligned}$$

so that

$$\begin{aligned} E^{n+1} = \mathbf{D}E^n + \Delta t\mathbf{A}F_u^nE^n + \Delta t\mathbf{R}F_u^{n+1}E^{n+1} + \Delta t^2\hat{\mathbf{A}}\dot{F}_u^nE^n + \Delta t^2\hat{\mathbf{R}}\dot{F}_u^{n+1}E^{n+1} \\ +O(\Delta t^2)E^n + O(\Delta t^2)E^{n+1} + \boldsymbol{\tau}^{n+1}. \end{aligned}$$

Note that $F_u^n, \dot{F}_u^n, F_u^{n+1}$, and $\dot{F}_u^{n+1}$ are all scalars. Now recall that $O(\Delta t)O(\|E^n\|) < O(\Delta t^2)$ since $p \geq 1$, and that terms of the form $\hat{\mathbf{R}}F_u^{n+1}$ and $\hat{\mathbf{A}}F_u^n$ are all bounded, so we have

$$E^{n+1} = \mathbf{D}E^n + \Delta t\mathbf{A}F_u^nE^n + \Delta t\mathbf{R}F_u^{n+1}E^{n+1} + O(\Delta t^2)E^n + O(\Delta t^2)E^{n+1}.$$

Now move the E^{n+1} terms to the left side:

$$\left(I - \Delta t\mathbf{R}F_u^{n+1} + O(\Delta t^2)\right)E^{n+1} = \mathbf{D}E^n + \Delta t\mathbf{A}F_u^nE^n + O(\Delta t^2) + \boldsymbol{\tau}^{n+1}.$$

Noting that $F_u^{n+1} = F_u^n + O(\Delta t)$ we have the desired result

$$\left(I - \Delta t\mathbf{R}F_u^n + O(\Delta t^2)\right)E^{n+1} = \left(\mathbf{D} + \Delta t\mathbf{A}F_u^n + O(\Delta t^2)\right)E^n + \boldsymbol{\tau}^{n+1}. \qquad \square$$

We now verify the assumption in the first paragraph of this subsection, that the error E^n is small.

LEMMA 2. *Given a zero-stable scheme* (1.3)*, if the order conditions* $\boldsymbol{\tau}_j = 0$ *are satisfied for* $j = 0, \ldots, p$*, and* $\mathbf{R}$ *and* F_u^n *are bounded, then there is some time interval* $[0, T]$ *such that the error* E^n *(given by* (2.3)*) satisfies*

$$\|E^n\| \leq O(\Delta t^p) \ll 1.$$

The proof of this lemma is given in [12, Lemma 3]. Using the formula in the proof it is easy to see that this interval on which the solution is small is a nontrivial time interval. Within this interval the derivation above is valid.

2.2. Two-derivative EIS and EIS+ schemes. In this section we consider a multiderivative GLM of the form (1.3) where we assume that $\mathbf{D}$ is a rank one matrix that satisfies the consistency condition $\mathbf{D}\mathbb{1} = \mathbb{1}$ so that this scheme is zero-stable. Furthermore, we assume that the coefficient matrices $\mathbf{D}$, $\mathbf{A}$, $\mathbf{R}$, $\hat{\mathbf{A}}$, $\hat{\mathbf{R}}$ are such that the order conditions $\boldsymbol{\tau}_j = 0$ are satisfied for all $j \le p$, so that the method will give us a numerical solution that has error that is guaranteed of order p. In the following theorem we show that if the truncation error vector $\boldsymbol{\tau}_{p+1}$ lives in the null-space of the operator $\mathbf{D}$, then the global error is of order $p+1$. Finally, the theorem shows that under additional conditions on the coefficient matrices $\mathbf{D}$, $\mathbf{A}$, $\mathbf{R}$, $\hat{\mathbf{A}}$, $\hat{\mathbf{R}}$, we can determine precisely the leading term of the global error, and therefore remove it by postprocessing, as will be described in subsection 2.3.

THEOREM 2.1. *Given a zero-stable two-derivative general linear method of the form*

$$V^{n+1} = \mathbf{D}V^n + \Delta t\mathbf{A}F(V^n) + \Delta t\mathbf{R}F(V^{n+1}) + \Delta t^2\hat{\mathbf{A}}\dot{F}(V^n) + \Delta t^2\hat{\mathbf{R}}\dot{F}(V^{n+1}),$$

where $\mathbf{D}$ *is a rank one matrix that satisfies the consistency condition* $\mathbf{D}\mathbb{1} = \mathbb{1}$*, and the coefficient matrices* $\mathbf{D}$*,* $\mathbf{A}$*,* $\mathbf{R}$*,* $\hat{\mathbf{A}}$*,* $\hat{\mathbf{R}}$ *satisfy the order conditions*

$$\boldsymbol{\tau}_j = 0 \quad \text{for } j = 1, \dots, p,$$

where

$$\boldsymbol{\tau}_j = \frac{1}{(j-1)!}\left(\frac{1}{j}\mathbf{D}(\mathbf{c}-\mathbb{1})^j + \mathbf{A}(\mathbf{c}-\mathbb{1})^{j-1} + (j-1)\hat{\mathbf{A}}(\mathbf{c}-\mathbb{1})^{j-2} + \mathbf{R}\mathbf{c}^{j-1} + (j-1)\hat{\mathbf{R}}\mathbf{c}^{j-2} - \frac{1}{j}\mathbf{c}^j\right) \quad \text{for } j = 1, 2, \dots,$$

if the error inhibiting condition

$$\mathbf{D}\boldsymbol{\tau}_{p+1} = 0 \tag{2.4a}$$

is satisfied, then the numerical solution produced by this method will have error

$$E^n = O(\Delta t^{p+1}).$$

Furthermore, if the conditions

$$\mathbf{D}\boldsymbol{\tau}_{p+2} = 0, \tag{2.4b}$$

$$\mathbf{D}(\mathbf{A}+\mathbf{R})\boldsymbol{\tau}_{p+1} = 0 \tag{2.4c}$$

are also satisfied, then error vector will have the more precise form

$$E^n = \Delta t^{p+1}\boldsymbol{\tau}_{p+1}^n + O(\Delta t^{p+2}). \tag{2.5}$$

Proof. Using Lemma 1 we obtain the equation for the evolution of the error

$$E^{n+1} = \left(I - \Delta t\mathbf{R}F_u^n + O(\Delta t^2)\right)^{-1}\left[\left(\mathbf{D} + \Delta t\mathbf{A}F_u^n + O(\Delta t^2)\right)E^n + \boldsymbol{\tau}^{n+1}\right] \tag{2.6}$$

the fact that F is smooth assures us that $\|\Delta t\mathbf{R}F_u^n\| \ll O(1)$ so that we can expand the first term to obtain

$$\begin{aligned} E^{n+1} &= \left(I + \Delta t\mathbf{R}F_u^n + O(\Delta t^2)\right)\left[\left(\mathbf{D} + \Delta t\mathbf{A}F_u^n + O(\Delta t^2)\right)E^n + \boldsymbol{\tau}^{n+1}\right] \\ &= \left(\mathbf{D} + \Delta tF_u^n(\mathbf{R}\mathbf{D} + \mathbf{A}) + O(\Delta t^2)\right)E^n \\ &\quad + \Delta t\left(\Delta t^p\boldsymbol{\tau}_{p+1}^{n+1} + \Delta t^{p+1}(\boldsymbol{\tau}_{p+2}^{n+1} + F_u^n\mathbf{R}\boldsymbol{\tau}_{p+1}^{n+1}) + O(\Delta t^{p+2})\right) \\ &= Q^nE^n + \Delta tT_e^n. \end{aligned}$$

The discrete Duhamel principle (given as Lemma 5.1.1 in [17]) states that given an iterative process of the form

$$E^{n+1} \;=\; Q^n E^n + \Delta t T_e^n,$$

where Q^n is a linear operator, we have

$$E^n \;=\; \prod_{\mu=0}^{n-1} Q^\mu E^0 \;+\; \Delta t \sum_{\nu=0}^{n-1} \left(\prod_{\mu=\nu+1}^{n-1} Q^\mu \right) T_e^\nu \;. \tag{2.7}$$

To analyze the error, we separate it into four parts,

$$E^n \;=\; \underbrace{\prod_{\mu=0}^{n-1} Q^\mu E^0}_{I} + \underbrace{\Delta t T_e^{n-1}}_{II} + \underbrace{\Delta t Q^{n-1} T_e^{n-2}}_{III} + \underbrace{\Delta t \sum_{\nu=0}^{n-3} \left(\prod_{\mu=\nu+1}^{n-1} Q^\mu \right) T_e^\nu}_{IV},$$

and discuss each part separately:

I. The method is initialized so that the numerical solution vector V^0 is accurate enough to ensure that the initial error E^0 is negligible and we can ignore the first term.

II. The final term in the summation is $\Delta t T_e^{n-1}$. Recall that

$$T_e^{n-1} = \left(\Delta t^p \boldsymbol{\tau}_{p+1}^n + \Delta t^{p+1} (\boldsymbol{\tau}_{p+2}^n + F_u^{n-1} \mathbf{R} \boldsymbol{\tau}_{p+1}^n) + O(\Delta t^{p+2}) \right)$$

so that this term contributes to the final-time error the term

$$\Delta t T_e^{n-1} = \Delta t^{p+1} \boldsymbol{\tau}_{p+1}^n + O(\Delta t^{p+2}).$$

III. The term $\Delta t Q^{n-1} T_e^{n-2}$ is a product of the operator

$$Q^{n-1} = \left(\mathbf{D} + \Delta t F_u^{n-1} (\mathbf{RD} + \mathbf{A}) + O(\Delta t^2) \right)$$

and the approximation error

$$T_e^{n-2} = \left(\Delta t^p \boldsymbol{\tau}_{p+1}^{n-1} + \Delta t^{p+1} (\boldsymbol{\tau}_{p+2}^{n-1} + F_u^{n-2} \mathbf{R} \boldsymbol{\tau}_{p+1}^{n-1}) + O(\Delta t^{p+2}) \right),$$

so we obtain

$$Q^{n-1} T_e^{n-2} = \Delta t^p \mathbf{D} \boldsymbol{\tau}_{p+1}^{n-1} + O(\Delta t^{p+1}) = O(\Delta t^{p+1}),$$

due to the condition (2.4a) that states that $\mathbf{D}\boldsymbol{\tau}_{p+1} = 0$. Then

$$\Delta t Q^{n-1} T_e^{n-2} = O(\Delta t^{p+2}).$$

IV. Finally, we look at the rest of the sum and use the boundedness of the operator Q^n to observe

$$\begin{aligned}
\left\| \Delta t \sum_{\nu=0}^{n-3} \left(\prod_{\mu=\nu+1}^{n-1} Q^\mu \right) T_e^\nu \right\| &= \left\| \Delta t \sum_{\nu=0}^{n-3} \left(\prod_{\mu=\nu+3}^{n-1} \hat{Q}^\mu \right) \left(\hat{Q}^{\nu+2} \hat{Q}^{\nu+1} T_e^\nu \right) \right\| \\
&\leq \Delta t \sum_{\nu=0}^{n-3} \left\| \prod_{\mu=\nu+3}^{n-1} Q^\mu \right\| \left\| Q^{\nu+2} Q^{\nu+1} T_e^\nu \right\| \\
&\leq \Delta t \sum_{\nu=0}^{n-3} (1 + c\,\Delta t)^{n-\nu-3} \left\| Q^{\nu+2} Q^{\nu+1} T_e^\nu \right\| \\
&= \frac{\exp(c t_n) - 1}{c} \max_{\nu=0,\ldots,n-3} \left\| Q^{\nu+2} Q^{\nu+1} T_e^\nu \right\|.
\end{aligned}$$

Clearly, the first term here is a constant that depends only on the final time, so it is the product $Q^{\nu+2}Q^{\nu+1}T_e^\nu$ we need to bound. Using the definition of the operators Q^μ

$$Q^{\nu+2}Q^{\nu+1} = \left[\mathbf{D}^2 + \Delta t\left(F_u^{\nu+2}\left(\mathbf{RD}+\mathbf{A}\right)\mathbf{D} + F_u^{\nu+1}\mathbf{D}\left(\mathbf{RD}+\mathbf{A}\right)\right) + O(\Delta t^2)\right]$$

and the approximation error

$$T_e^\nu = \left(\Delta t^p \boldsymbol{\tau}_{p+1}^{\nu+1} + \Delta t^{p+1}(\boldsymbol{\tau}_{p+2}^{\nu+1} + F_u^\nu \mathbf{R}\boldsymbol{\tau}_{p+1}^{\nu+1}) + O(\Delta t^{p+2})\right),$$

we have

$$\begin{aligned} Q^{\nu+2}Q^{\nu+1}T_e^\nu = {} & \Delta t^p\left(\mathbf{D} + \Delta t F_u^{\nu+2}\left(\mathbf{RD}+\mathbf{A}\right) + \Delta t F_u^{\nu+1}\mathbf{DR}\right)\mathbf{D}\boldsymbol{\tau}_{p+1}^{\nu+1} \\ & + \Delta t^{p+1}\left(F_u^{\nu+1}\mathbf{D}^2\mathbf{R} + F_u^{\nu+1}\mathbf{DA}\right)\boldsymbol{\tau}_{p+1}^{\nu+1} + \Delta t^{p+1}\mathbf{D}^2\boldsymbol{\tau}_{p+2}^{\nu+1} \\ & + O(\Delta t^{p+2}). \end{aligned}$$

Using the fact that in our case $\mathbf{D}^2 = \mathbf{D}$ and that $F_u^{\nu+2} = F_u^{\nu+1} + O(\Delta t)$, we obtain

$$\begin{aligned} Q^{\nu+2}Q^{\nu+1}T_e^\nu = {} & \Delta t^p\left(\mathbf{D} + \Delta t F_u^{\nu+2}\left(\mathbf{RD}+\mathbf{A}\right) + \Delta t F_u^{\nu+1}\mathbf{DR}\right)\mathbf{D}\boldsymbol{\tau}_{p+1}^{\nu+1} \\ & + \Delta t^{p+1}F_u^{\nu+1}\mathbf{D}\left(\mathbf{R}+\mathbf{A}\right)\boldsymbol{\tau}_{p+1}^{\nu+1} + \Delta t^{p+1}\mathbf{D}\boldsymbol{\tau}_{p+2}^{\nu+1} + O(\Delta t^{p+2}). \end{aligned}$$

The first and third terms disappear because of (2.4a) and (2.4b),

$$\mathbf{D}\boldsymbol{\tau}_{p+1} = 0 \quad \text{and} \quad \mathbf{D}\boldsymbol{\tau}_{p+2} = 0.$$

The second term is eliminated by (2.4c),

$$\mathbf{D}\left(\mathbf{R}+\mathbf{A}\right)\boldsymbol{\tau}_{p+1} = 0.$$

So we have

$$Q^{\nu+2}Q^{\nu+1}T_e^\nu = O(\Delta t^{p+2}).$$

All these parts together give us the result

$$\begin{aligned} E^n &= \underbrace{\prod_{\mu=0}^{n-1} Q^\mu E^0}_{I} + \underbrace{\Delta t T_e^{n-1}}_{II} + \underbrace{\Delta t Q^{n-1}T_e^{n-2}}_{III} + \underbrace{\Delta t \sum_{\nu=0}^{n-1}\left(\prod_{\mu=\nu+1}^{n-3} Q^\mu\right)T_e^\nu}_{IV} \\ &= 0 + \underbrace{\Delta t^{p+1}\boldsymbol{\tau}_{p+1}^n}_{II} + \underbrace{O(\Delta t^{p+2})}_{III} + \underbrace{O(\Delta t^{p+2})}_{IV} \\ &= \Delta t^{p+1}\boldsymbol{\tau}_{p+1}^n + O(\Delta t^{p+2}). \end{aligned} \tag{2.8}$$

∎

Remark 1. We note that generalizing this approach from our work in [12] to include a second derivative was quite simple. This is due to the fact that the second derivative appears with a Δt^{p+2} term multiplying it, so that the operator Q^n is in fact exactly the same for a two-derivative method of the form (1.3) as for the methods considered in [12]. For this reason, expanding this approach to higher derivatives will be simple: the only change would be in the definition of the order conditions (1.8).

2.3. Designing a postprocessor to recover $p+2$ order. If a method satisfies all the conditions of Theorem 2.1, including the order conditions, the EIS condition, and the EIS+ conditions, then at every time-step t_n the error vector E^n has the particular form (see (2.5))

$$E^n = \Delta t^{p+1} \boldsymbol{\tau}_{p+1}^n + O(\Delta t^{p+2}).$$

Thus, the leading order term $\Delta t^{p+1} \boldsymbol{\tau}_{p+1}^n$ can be removed by postprocessing at the end of the computation and a final-time solution of order $p+2$ can be obtained, as we showed in [12]. We notice that the form of the error for multiderivative method (1.3) is exactly the same as that of the error of the EIS+ methods presented in [12], the only difference being the definition of the truncation error vector. The theoretical discussion of this postprocessor is presented in [12], and we refer the reader to the presentation there. We review one such construction below:

1. Select the number of computation steps m that will be used, requiring that
$$ms \geq p+3,$$
where s is the number of steps and p the truncation-error order of the time-stepping method.
2. Define the time vector of all the abscissas in the last m computation steps:
$$\tilde{c} = \begin{pmatrix} \mathbf{c} - (m-1)\mathbb{1} \\ \mathbf{c} - (m-2)\mathbb{1} \\ \vdots \\ \mathbf{c} \end{pmatrix}.$$
3. Stack m copies of the final truncation error vector $\boldsymbol{\tau}_{p+1}$,
$$\tilde{\boldsymbol{\tau}} = \left(\underbrace{\boldsymbol{\tau}_{p+1}^T, \ldots, \boldsymbol{\tau}_{p+1}^T}_{m} \right)^T.$$
4. Define the vertically flipped Vandermonde interpolation matrix $\mathcal{T}$ on the vector $\tilde{\mathbf{c}}$, and replace the first column (the one corresponding to the highest polynomial term) by $\tilde{\boldsymbol{\tau}}$:
$$\mathcal{T} = \left(\tilde{\boldsymbol{\tau}}, \tilde{\mathbf{c}}^{ms-2}, \ldots, \tilde{\mathbf{c}}^2, \tilde{\mathbf{c}}, \mathbb{1} \right),$$
where terms of the form $\tilde{\mathbf{c}}^q$ are understood as componentwise exponentiation. Note that $\mathcal{T}$ is a square matrix of dimension $ms \times ms$.
5. Define the postprocessing filter
$$\Phi = \mathcal{T} \operatorname{diag} \left(0, \underbrace{1, \ldots, 1}_{ms-1}, \right) \mathcal{T}^{-1}.$$

Finally, left-multiply the solution vector

$$\tilde{V}^n = \left((V^{n-m+1})^T, \ldots, (V^n)^T \right)^T$$

by the postprocessing filter Φ to obtain the postprocessed solution

$$\hat{V}^n = \Phi \tilde{V}^n,$$

which will be of order $p+2$, as shown in [12].

As in [12] we remark that this process may break down if the matrix $\mathcal{T}$ is not invertible, and that numerical instabilities may result if $\|\Phi\|$ is large. This should be verified while building these matrices. Remedies to this are being investigated in the current work. However, for the methods reported in this paper this construction was appropriate.

Remark 2. Devising the postprocessor is completed *only once* for each method, and so does not add cost to the simulation. Furthermore, postprocessing is done once at the end of each simulation, and the cost of postprocessing is negligible as it involves only a linear combination of previously computed solutions. However, postprocessing does require the availability of m time-level solutions, and the storage may be prohibitive for extremely large problems. Another issue with postprocessing is the fact that step-size changes may affect the accumulation of the error and the variable time-step may not preserve the form of the error needed for postprocessing. However, it is important to recall that in the work of [25, 49, 26] on other methods, they provide error estimators through control of the growth of the errors through conditions similar to the EIS+ conditions (2.4b) and (2.4c), which suggests that some time-adaptivity will not destroy the form of the error. These issues are under active investigation and will be the subject of future work.

3. New error inhibiting GLM schemes. In this section we present the new general linear methods (GLMs) that satisfy the error inhibiting conditions (2.4a)–(2.4c). We divide these methods into three types: explicit methods that have good linear stability regions, explicit strong stability preserving methods, and implicit methods that are A-stable. The coefficients of all the methods mentioned in this section can be downloaded from our GitHub site [16].

3.1. Explicit EIS methods with favorable linear stability regions. We used the optimization procedure to produce explicit methods with relatively large regions of linear stability. As is usual in this field, the linear stability region is defined as the value of the complex number λ so that the method applied to the problem $y' = \lambda y$ (and so $y'' = \lambda y' = \lambda^2 y$) is stable.

We consider two types of explicit methods in this section, and denote them eEIS(s,P)$_2$ and eEIS+(s,P)$_2$. The eEIS(s,P)$_2$ methods have s stages and satisfy the order conditions to some order p and the EIS condition (2.4a) so that the overall order is $P = p+1$. The eEIS+(s,P)$_2$ methods also have s stages and satisfy the order conditions to some order p, but they satisfy all three error inhibiting conditions (2.4a)–(2.4c) so that the solution after postprocessing is accurate of order $P = p+2$. In this section we present and compare the linear stability regions of some of these methods. Due to space constraints we do not present the coefficients of all the methods. However, we present the coefficients of a selection of the methods that will be used in the numerical tests in section 4, and the coefficients of all methods mentioned in this section can be downloaded from our GitHub site [16].

Explicit two-stage third-order EIS method (eEIS(2,3)$_2$). We begin with an explicit EIS GLM that has $s = 2$ stages and satisfies the truncation error conditions to $p = 2$ and the EIS condition (2.4a) so that the overall order is $P = 3$, and refer to

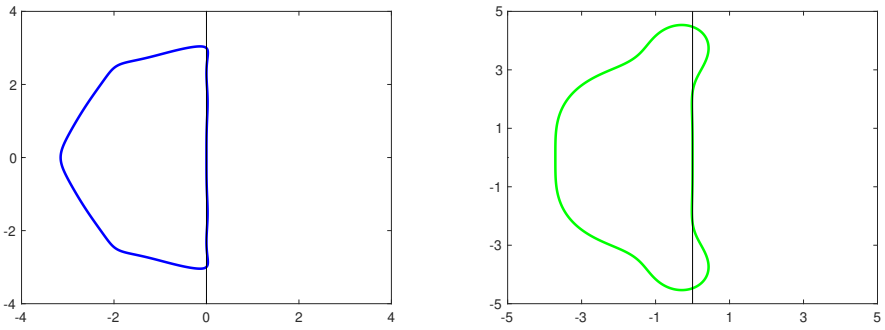

FIG. 3.1. *The linear stability region of the EIS method eEIS(2,3)$_2$ on left compared to that of the Kinnmark–Gray method eKG(5,3) on the right.*

this method as eEIS(2,3)$_2$:

$$\mathbf{D}_{1,i} = 1.347635863512091, \quad \mathbf{D}_{2,i} = -0.347635863512091,$$

$$\mathbf{A} = \begin{pmatrix} 1.110588320380528 & 0.206278390370703 \\ 1.160801319467423 & 0.191968442856969 \end{pmatrix},$$

$$\hat{\mathbf{A}} = \begin{pmatrix} 0.376508598017949 & 0.079881117612918 \\ 0.424704932282709 & 0.083778591655645 \end{pmatrix},$$

$$\mathbf{R} = \begin{pmatrix} 0 & 0 \\ 0.875587228946215 & 0 \end{pmatrix}, \quad \hat{\mathbf{R}} = \begin{pmatrix} 0 & 0 \\ 0.412259887079832 & 0 \end{pmatrix},$$

In Figure 3.1 we compare the stability region of this method to that of the five-stage third order Kinnmark–Gray method eKG(5,3) [23]. The linear stability region of the Kinnmark–Gray method eKG(5,3) method, which has five function evaluations, is shown with the y-axis going from $(-5, 5)$, while the eEIS(2,3)$_2$ method, which has four total function evaluations (two evaluations of F and two of $\dot{F}$), is shown with the y-axis going from $(-2s, 2s) = (-4, 4)$. Clearly, it seems that the Kinnmark–Gray method is more efficient, even considering the extra cost of computing five function evaluations over four for the eEIS(2,3)$_2$ method. However, the argument for using two-derivative methods [38] is that the second derivative is often needed for other parts of the simulation (e.g., the spatial derivative), and is computed whether or not it is used in the time-stepping. This, for example, is the case for hyperbolic PDEs in which the Jacobian F_u must be computed to determine the direction of the characteristics, and the term F is known, so that $\dot{F} = F_u F$ is available with no costly computations. Thus, we can consider the cost of the eEIS(2,3)$_2$ method to be that of two function evaluations rather than four. In this case, the eEIS(2,3)$_2$ has a *larger* effective linear stability region than the eKG(5,3) method. The coefficients of the well-known eKG(5,3) method can be found in [23] or downloaded from our GitHub site [16].

Next, we compare the linear stability regions of eEIS(s,P)$_2$ and eEIS+(s,P)$_2$ methods we found using our optimization method. In Figure 3.2, on the left we show the linear stability regions of the two-stage fifth order methods eEIS(2,5)$_2$ method (in blue) and the eEIS+(2,5)$_2$ method (in red). In Figure 3.2, on the right we show the linear stability regions of the three-stage seventh order methods eEIS(3,7)$_2$ (in blue)

and the eEIS+$(3,7)_2$ (in red). In both cases we can see that the linear stability region of the EIS+$(s,P)_2$ method is larger than that of the corresponding eEIS$(s,P)_2$ method. This suggests that the additional order conditions that the eEIS$(s,P)_2$ method must satisfy (the eEIS$(s,P)_2$ method must satisfy $P-1$ conditions instead of $P-2$ conditions for the eEIS+$(s,P)_2$) plays more of a role in constraining the linear stability region than the additional error inhibiting conditions (2.4b) and (2.4c) that the eEIS+$(s,P)_2$ conditions must satisfy. These results show that from a linear stability perspective, it is more efficient to use the explicit EIS+ methods than the corresponding EIS methods. This is because the cost of one-time postprocessing is negligible compared to the cost incurred by a restricted time-step which will require many more time-steps to reach the same final time.

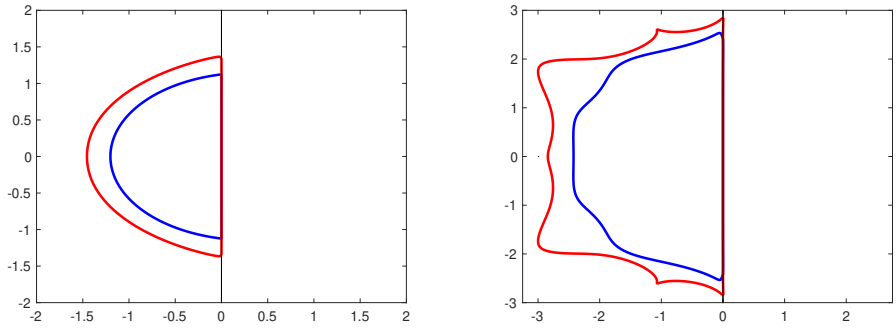

FIG. 3.2. *The linear stability region of the EIS$(s,P)_2$ compared to the EIS+$(s,P)_2$ methods. On the left: the stability region of the eEIS$(2,5)_2$ in blue and eEIS+$(2,5)_2$ in red. On the right: the stability region of the eEIS$(3,7)_2$ in blue and eEIS+$(3,7)_2$ in red. (Figure in color online.)*

We now focus on the linear stability regions of some explicit EIS+$(s,P)_2$ methods of up to eighth order.

Explicit two-stage fifth-order EIS+ method (eEIS+$(2,5)_2$).

$$\mathbf{D}_{1,i} = 0.500023658051142, \quad \mathbf{D}_{2,i} = 0.499976341948858,$$

$$\mathbf{A} = \begin{pmatrix} 0.627069692131650 & 0.151022064558538 \\ 0.709712162750524 & 0.848963643214302 \end{pmatrix},$$

$$\hat{\mathbf{A}} = \begin{pmatrix} 0.058142153689242 & 0.325582994094698 \\ 0.108273930132603 & 0.477624731406111 \end{pmatrix},$$

$$\mathbf{R} = \begin{pmatrix} 0 & 0 \\ -0.336746561995068 & 0 \end{pmatrix}, \quad \hat{\mathbf{R}} = \begin{pmatrix} 0 & 0 \\ 0.367133756538675 & 0 \end{pmatrix}.$$

The truncation error vector (needed for postprocessing) is

$$\boldsymbol{\tau}_4 = (-0.039533847641586, 0.039537588993770)^T.$$

Explicit two-stage sixth-order EIS+ method (eEIS+$(2,6)_2$).

$$\mathbf{D}_{1,i} = 0.193021555206000, \quad \mathbf{D}_{2,i} = 0.806978444794000,$$

$$\mathbf{A} = \begin{pmatrix} 1.089589263420254 & -0.469532861646008 \\ 1.011690204056872 & 1.112307786855907 \end{pmatrix},$$

$$\hat{\mathbf{A}} = \begin{pmatrix} 0.196914195858807 & 0.434709438834146 \\ 0.130811273979010 & 0.871687677021200 \end{pmatrix},$$

$$\mathbf{R} = \begin{pmatrix} 0 & 0 \\ -1.033119102271808 & 0 \end{pmatrix}, \quad \hat{\mathbf{R}} = \begin{pmatrix} 0 & 0 \\ 0.499137031946415 & 0 \end{pmatrix}.$$

The truncation error vector (needed for postprocessing) is

$$\boldsymbol{\tau}_5 = (-0.037857689452761, 0.009055198613815)^T.$$

Explicit three-stage seventh-order EIS+ method (eEIS+(3,7)$_2$).

$$\mathbf{D}_{1,i} = 1.581021525561460,\ \mathbf{D}_{2,i} = -0.598751979308602,\ \mathbf{D}_{3,i} = 0.017730453747142,$$

$$\mathbf{A} = \begin{pmatrix} 0.931591460185742 & 0.379244369981835 & -0.172141957956410 \\ 0.938547162180577 & 0.508131122095280 & -0.363857858559788 \\ 0.504648760586788 & 1.046850936001111 & -0.659275924405796 \end{pmatrix},$$

$$\hat{\mathbf{A}} = \begin{pmatrix} 0.057154143906362 & 0.302522642478094 & 0.175689200743141 \\ 0.045099335357263 & 0.359020777972142 & 0.164798140168151 \\ -0.060217523878309 & 0.456569929293375 & -0.005615338892051 \end{pmatrix},$$

$$\mathbf{R} = \begin{pmatrix} 0 & 0 & 0 \\ 0.307438691150295 & 0 & 0 \\ 1.789973573982305 & -0.870575633439973 & 0 \end{pmatrix},$$

$$\hat{\mathbf{R}} = \begin{pmatrix} 0 & 0 & 0 \\ 0.038804362951013 & 0 & 0 \\ 0.227157707727078 & 0.276283023303938 & 0 \end{pmatrix}.$$

The truncation error vector (needed for postprocessing) is

$$\boldsymbol{\tau}_6 = (-0.003599790543666, -0.012406980352919, -0.097987210664809)^T.$$

Explicit four-stage eighth-order EIS+ method (eEIS+(4,8)$_2$).

$$\mathbf{D}_{1,i} = 1.126765222628176, \quad \mathbf{D}_{2,i} = 0.808129178515260,$$

$$\mathbf{D}_{3,i} = -0.107647150078402, \quad \mathbf{D}_{4,i} = -0.827247251065033,$$

$$\mathbf{A} = \begin{pmatrix} 0.567574025309926 & 0.723999455772069 & 0.208196137734782 & 0.023532165559543 \\ 0.749691669482323 & 0.430151531239573 & 0.359568096205409 & -0.030974711893773 \\ 0.602555996794216 & 0.745759221902972 & 0.048559187429251 & -0.267889537378177 \\ 1.051588361923041 & -0.047355340428569 & 0.863960642835203 & 0.214102220881218 \end{pmatrix},$$

$$\hat{\mathbf{A}} = \begin{pmatrix} 0.041975696597772 & 0.205746598967380 & 0.137652258393657 & 0.039122406247340 \\ 0.064927843091523 & 0.213465637934016 & 0.160720650985361 & -0.047428374982532 \\ 0.056975020786010 & 0.171669459177575 & 0.226994033551341 & -0.021617692260293 \\ 0.095018403341495 & 0.263066907087928 & 0.147903147440657 & -0.036525606967693 \end{pmatrix},$$

$$\mathbf{R} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0.296825313241825 & 0 & 0 & 0 \\ 0.379857836431130 & 0.610459020171445 & 0 & 0 \\ 0.079086170545983 & 0.114409044614819 & 0.077980998192235 & 0 \end{pmatrix},$$

$$\hat{\mathbf{R}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0.095598816350501 & 0 & 0 & 0 \\ -0.143446089841412 & 0.076113483149991 & 0 & 0 \\ 0.309290513515929 & 0.063106409144583 & 0.076129207423402 & 0 \end{pmatrix}.$$

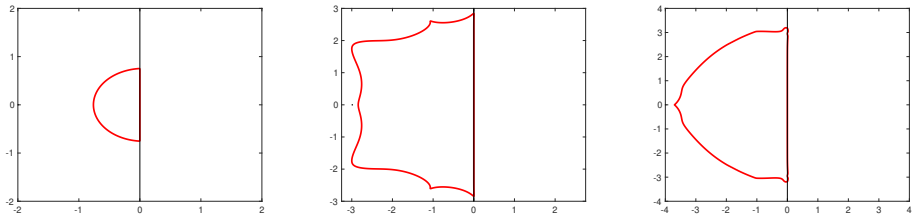

FIG. 3.3. *The linear stability regions of the eEIS+(2,6)$_2$ (left), eEIS+(3,7)$_2$ (middle), and eEIS+(4,8)$_2$ (right) methods. Note that both the x-axis and y-axis limits are proportional to the number of stages, so that the graph is on $(-s, s) \times (-s, s)$. This accounts for the increased cost of function evaluations that is proportional to the number of stages.*

The truncation error vector (needed for postprocessing) is

$$\boldsymbol{\tau}_7 = \begin{pmatrix} -0.000997109517747 \\ -0.006485724807936 \\ -0.023117224006582 \\ -0.004685791946531 \end{pmatrix}.$$

These methods are further explored in subsection 4.1 where these methods are tested for convergence on a nonlinear ODE. Figure 3.3 shows the linear stability regions of the eEIS+(2,6)$_2$ (left), eEIS+(3,7)$_2$ (middle), and eEIS+(4,8)$_2$ (right) methods. The axis limits are in all cases $(-s, s)$, to account for the increased cost of function evaluations that is proportional to the number of stages. The high order methods eEIS+(3,7)$_2$ and eEIS+(4,8)$_2$ clearly have very favorable regions of linear stability.

3.2. Explicit strong stability preserving EIS methods. The solution to a hyperbolic conservation law may develop sharp gradients or discontinuities, which requires careful handling to ensure that the numerical solution is stable. For this purpose, a spatial discretization is carefully designed to satisfy some nonlinear stability properties (such as total variation diminishing, maximum norm preserving, or positivity preserving properties) to ensure that it can handle the presence of a discontinuity. However, this spatial discretization method must be paired with an appropriate time-stepping method to ensure preservation of these nonlinear non–inner-product stability properties [14]. Such time-stepping methods are known as strong stability preserving (SSP) time discretizations [14].

SSP time-stepping methods preserve the nonlinear non–inner-product stability properties of the spatial discretization when coupled with forward Euler, under some time-step restriction

$$\|u^n + \Delta t F(u^n)\| \le \|u^n\| \quad \text{for} \quad \Delta t \le \Delta t_{FE}. \tag{3.1}$$

For two-derivative methods, preserving the forward Euler condition is not enough: we need to augment this with a condition that includes the second derivative.

There are two approaches to deriving such a condition. The first is to consider a second derivative condition of the form

$$\|u^n + \Delta t^2 \dot{F}(u^n)\| \le \|u^n\| \quad \text{for} \quad \Delta t \le K \Delta t_{FE}. \tag{3.2}$$

Christlieb et al. [8] considered this type of condition and derived two-derivative Runge–Kutta methods which preserve the strong stability properties of forward Euler

(3.1) and the second derivative condition (3.2). In [29], Moradi, Farzi, and Abdi extended this SSP approach to design two-derivative GLMs. However, this approach has a significant flaw in that many common spatial discretizations that scientists wish to use in simulations do not satisfy the condition (3.2), and so the methods in [8, 29] are not useful in these cases. Due to the limitations imposed by two-derivative methods that preserve the strong stability properties of forward Euler (3.1) and the second derivative condition (3.2), Gottlieb, Grant, and Seal [15] considered an alternative condition which allows the development of two-derivative Runge–Kutta methods which are still SSP if the forward Euler and second derivative conditions (3.1) and (3.2) hold, but also apply to a broader set of spatial discretizations where the second derivative condition (3.2) does not hold.

Instead of relying on the second derivative condition (3.2), the approach in [15] preserves the SSP properties of the forward Euler method (3.1) and the Taylor series condition

$$\left\| u^n + \Delta t F(u^n) + \frac{1}{2}\Delta t^2 \dot{F}(u^n) \right\| \le \|u^n\| \text{ for } \Delta t \le K \Delta t_{FE}, \tag{3.3}$$

where K is some constant. It was shown in [15] that any spatial discretization that satisfies the forward Euler and second derivative conditions (3.1) and (3.2) will also satisfy the Taylor series condition (3.3). Thus, two-derivative time-stepping methods that are SSP in the sense that they preserve the forward Euler and Taylor series conditions (3.1) and (3.3) are applicable to problems with spatial discretization that satisfy the conditions (3.1) and (3.2). In this sense the time-stepping methods in [15] are a subset of the methods in [8].

In this work we consider, as we did in [15], the base conditions (3.1) and (3.3). The explicit SSP two-derivative GLM methods that we develop in this work can be decomposed into a convex combination of (3.1) and (3.3) with $K = 1$, and thus preserve their strong stability properties under the time-step restriction

$$\Delta t \le \mathcal{C} \Delta t_{FE}.$$

This makes the SSP methods we develop in this work more applicable to semidiscretizations of PDEs which arise from a broader class of spatial discretization than the two-derivative GLMs in [29].

Using the optimization code we found a number of two-derivative error inhibiting SSP methods with optimized SSP coefficient $\mathcal{C}$. We denote by eSSP-EIS(s,P)$_2$ the methods that satisfy the order conditions to order $p = P - 1$ and the error inhibiting condition (2.4a). The methods that satisfy the order conditions to order $p = P - 2$ and satisfy the error inhibiting conditions (2.4a)–(2.4c) are denoted eSSP-EIS+(s,P)$_2$. We present three SSP error inhibiting methods in this section. The other SSP method mentioned in this section can be downloaded from our GitHub repository [16].

Explicit strong stability preserving eSSP-EIS(2,3)$_2$ with $\mathcal{C} = 1.5$. This method satisfies the truncation error conditions up to order $p = 2$ and the EIS condition (2.4a) so that we obtain third order convergence ($P = 3$). The coefficients of this method are rational and given by

$$\mathbf{D} = \frac{1}{16}\begin{pmatrix} 7 & 9 \\ 7 & 9 \end{pmatrix}, \quad \mathbf{A} = \frac{1}{8}\begin{pmatrix} 2 & 3 \\ 2 & 3 \end{pmatrix}, \quad \hat{\mathbf{A}} = \frac{1}{8}\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}, \quad \mathbf{R}_{2,1} = \frac{2}{3}, \quad \hat{\mathbf{R}}_{2,1} = \frac{2}{9}.$$

The remaining coefficients of $\mathbf{R}$ and $\hat{\mathbf{R}}$ are zero, as this method is explicit. It is interesting to note that the optimization procedure converged to this EIS method

TABLE 3.1

SSP coefficients of the error inhibiting methods found and the comparable MSRK methods in [1]. *A dash denotes that no such method was obtained. The asterisk indicates that the code searching for a method of the type eSSP-EIS+(s,P)$_2$ actually converged to a method of the type eSSP-EIS(s,P)$_2$.*

s	P	eSSP-EIS(s,P)$_2$	eSSP-EIS+(s,P)$_2$	eSSP-MSRK(s,s,P)
2	3	1.500	*	0.732
2	4	0.990	1.000	–
2	5	–	0.555	–
3	4	1.811	1.896	1.163
3	5	1.369	1.548	0.638
3	6	0.546	1.078	0.029
3	7	–	0.129	–

even when we requested a method that satisfies the truncation error condition only up to order $p = 1$ and all the EIS+ conditions (2.4a)–(2.4c).

The SSP coefficient of this method is $\mathcal{C} = 1.5$. We compare this to the SSP coefficient of the standard third order GLM method with two steps and two stages given in [1]; we denote this one-derivative method by eSSP-MSRK(2,2,3), indicating that it has two steps, two stages, and is order three. The SSP coefficient of the eSSP-MSRK(2,2,3) method is $\mathcal{C} \approx 0.73$. Our two-derivative error inhibiting method eSSP-EIS(2,3)$_2$ (with $\mathcal{C} = 1.5$) has four function evaluations and the eSSP-MSRK(2,2,3) method has two function evaluations, so the two methods would seem to be equally efficient; however, if we consider applications in which the second derivative is available with no additional cost (such as in simulations of hyperbolic PDEs), then the two-step method eSSP-EIS(2,3)$_2$ has an *effective SSP time-step* that is twice as large.

We found a number of other SSP methods, and present their SSP coefficients in Table 3.1. The first column gives the number of stages s, the second column gives the overall order P that the solutions can be expected to be: for EIS methods this is $P = p + 1$ and for EIS+ methods this is $P = p + 2$, which is the order after postprocessing. We note that we did not find an eSSP-EIS+(2,3)$_2$ method because the optimization scheme converged to the eSSP-EIS(2,3)$_2$ method instead, and that we could not obtain an eSSP-EIS(2,5)$_2$ or an eSSP-EIS(3,7)$_2$ method and believe they do not exist.

In Table 3.1 we compare the SSP coefficients of the eSSP-EIS(s,P)$_2$ and eSSP-EIS+(s,P)$_2$ methods to those of the comparable SSP GLM methods of [1]. These methods are denoted eSSP-MSRK(s,s,P) to indicate that they have s steps and s stages and are of order P. This table shows that for a given s and P, the EIS+ methods have larger allowable SSP coefficients and so are more efficient, in cases where the derivative evaluation does not incur any additional cost, and the cost of postprocessing is negligible. Table 3.1 can be downloaded from our GitHub directory [16]. Below we list the coefficients of two featured methods.

Explicit strong stability preserving eSSP-EIS+(2,4)$_2$ with $\mathcal{C} = 1.0$. This method has the following coefficients:

$$\mathbf{D} = \begin{pmatrix} 0.435605756635718 & 0.564394243364282 \\ 0.435605756635718 & 0.564394243364282 \end{pmatrix},$$

$$\mathbf{A} = \begin{pmatrix} 0.232303428413552 & 0.564394243364282 \\ 0.216263460427852 & 0.564394243364282 \end{pmatrix},$$

$$\hat{\mathbf{A}} = \begin{pmatrix} 0.000000005124887 & 0.260081562620613 \\ 0.000000001928255 & 0.146835746492061 \end{pmatrix},$$

where the nonzero elements of $\mathbf{R}$ and $\hat{\mathbf{R}}$ are

$$\mathbf{R}_{2,1} = 0.376253295127924 \text{ and } \hat{\mathbf{R}}_{2,1} = 0.162082671864920.$$

The truncation error vector (needed for postprocessing) is

$$\tau_3 = \begin{pmatrix} -0.063938362828511 \\ 0.049348339827035 \end{pmatrix}.$$

Explicit strong stability preserving eSSP-EIS+$(3,6)_2$ with $\mathcal{C} = 1.0782$. This method has the following coefficients:

$$\mathbf{D}_{1,i} = 0.235787420033905, \ \mathbf{D}_{2,i} = 0.332249926343388, \ \mathbf{D}_{3,i} = 0.431962653622707,$$

$$\mathbf{A} = \begin{pmatrix} 0.179040619183497 & 0 & 0.400647796399945 \\ 0.147616987633695 & 0.118289307755180 & 0.400647796399945 \\ 0.194101834261448 & 0.212027154638658 & 0.400647796399945 \end{pmatrix},$$

$$\hat{\mathbf{A}} = \begin{pmatrix} 0.032860477842919 & 0 & 0.068024553668439 \\ 0.024965463148830 & 0.034155124171981 & 0.021087452933654 \\ 0.011487692416560 & 0.092903917927740 & 0.124915188800131 \end{pmatrix},$$

$$\mathbf{R} = \begin{pmatrix} 0 & 0 & 0 \\ 0.287524583705647 & 0 & 0 \\ 0.214948333287866 & 0.243023557774243 & 0 \end{pmatrix},$$

$$\hat{\mathbf{R}} = \begin{pmatrix} 0 & 0 & 0 \\ 0.133340336145235 & 0 & 0 \\ 0.050250968106130 & 0.112702859933545 & 0. \end{pmatrix},$$

The truncation error vector (needed for postprocessing) is

$$\tau_5 = \begin{pmatrix} -0.010752778908703 \\ -0.021534888908005 \\ 0.022433270953649 \end{pmatrix}.$$

We note that only the method itself is guaranteed to be SSP; we do not expect the postprocessed solution to be SSP as well. The postprocessor is only designed to extract a higher order solution but not to preserve the strong stability properties. This does not typically pose a problem because preserving the nonlinear non–inner-product stability properties is generally only important for the stability of the time evolution. Once we reach the final time, these properties of the solution are no longer needed. If we still desire some properties that are destroyed by the postprocessor, we can opt to use the non–post-processed solution instead.

3.2.1. Comparing the EIS and EIS+ SSP two-derivative GLMs to other SSP methods. In this work we follow the SSP approach in [15] and the EIS+ approach in [12] and extend it to two-derivative EIS peer methods. This approach differs from the approach in [29] in several ways: First, it exploits the EIS and EIS+ mechanism to obtain higher order methods that have larger effective SSP coefficients; second, the SSP approach adopted here is the one used in [15] which is applicable to a broader class of problems than those in [29]; third, we consider only methods that are in the form (1.3) which extends peer methods to two derivatives while the methods in [29] are a broader class of GLMs. The methods we consider *all* have higher stage

order and produce a vector of solutions where all the solution is approximated to high order at all time-levels.

In [8] the typical value of K was $K = \frac{\sqrt{2}}{2}$, because the second derivative condition (3.2) for the centered difference approximation of u_{xx} had this bound for total variation behavior. This value was also adopted in [29] as the typical value. The same centered difference approximation of u_{xx} gives a value of $K = \frac{1+\sqrt{5}}{2}$ when we consider the Taylor series condition (3.3). However, the Taylor series condition is also satisfied for the upwind approximation of u_{xx} with a value of $K = 1$, so we use as the typical value of K the stricter of the two, which is the value $K = 1$. Note that the second derivative condition (3.2) is not satisfied when we use the upwind approximation of u_{xx} and so time-stepping methods such as those in [8, 29] cannot be used for this discretization.

In Table 3.2 we compare the effective SSP coefficients of our methods to those in [29]. We see in this table that when we look at third order methods with $s = 2$ steps and stages, our eSSP-EIS$(2, 3)_2$ method has stage order $q = p = 2$ and its effective SSP coefficient is 37% higher than that of the SGLM$_{32}(P)$ method in [29]. In general, we observe that the improvement in the EIS methods compared to the SGLM$_{sq}$ methods in [29] ranges from 16% to 47%. The EIS+ postprocessing methods show improvement in the range of 15% to 25% compared to the SGLM$_{sq}(P)$ methods in [29]. In addition, there are many methods, such as the $s = 2$ fifth order and $s = 3$ sixth and seventh order methods, that are not obtained at all in [29]. This is especially true for the eSSP-EIS+$(s, P)_2$ methods.

TABLE 3.2

Effective SSP coefficients of the eSSP-EIS$(s, P)_2$ and eSSP-EIS+$(s, P)_2$ methods, compared to the SGLM$_{sq}(P)$ methods in [29]. *The value of K used is $1/\sqrt{2}$ for the methods from* [29] *and $K = 1$ for our methods. The value s is the number of steps and stages in the method, q is the stage order, and P is the overall order of the solution. Dashes indicate that these methods are not present in* [29], *and therefore, there is no percent improvement in the SSP coefficient listed.*

$s =$	$q =$	$P =$	eSSP-EIS$(s, P)_2$	SGLM$_{sq}(P)$	Improvement
2	2	3	0.75	0.547	37%
2	3	4	0.495	0.427	16%
3	3	4	0.603	0.428	40%
3	4	5	0.456	0.310	47%
3	5	6	0.182	–	–
$s =$	$q =$	$P =$	eSSP-EIS+$(s, P)_2$	SGLM$_{sq}(P)$	Improvement
2	2	4	0.5	0.436	15%
2	3	5	0.2775	–	–
3	2	4	0.632	0.504	25%
3	3	5	0.516	0.427	21%
3	4	6	0.359	–	–
3	5	7	0.043	–	–

This is an interesting result because the class of methods we explore here can be seen as a subset of the methods considered in [29] in two senses:

(1) The methods we considered here use an SSP definition that relies on preserving (3.1) and (3.3), which is a subset of the methods that preserve (3.1) and (3.2) (as in [29]) in the sense that any spatial discretization that preserves (3.1) and (3.2) will also preserve (3.1) and (3.3), but our methods also apply to spatial discretizations that don't have the property (3.2). For this reason, there are more SSP methods in the sense of [29], and the SSP methods in this work are a subset of the class in [29].

(2) In [29] the more general class of two-derivative GLMs is used, whereas we require the two-derivative peer method form (1.3). These two factors would make us

expect that our methods would have smaller effective SSP coefficients than those in [29]. In fact, as we showed in Table 3.2, our methods typically have larger effective SSP coefficients. We see that replacing order conditions with the EIS and EIS+ conditions allows for a larger SSP coefficient; this is the power of the error inhibiting paradigm. In summary, the EIS and EIS+ explicit two-derivative SSP methods we produced are more efficient in the sense of larger allowable effective time-step, and applicable to a broader class of spatial discretizations.

3.3. Implicit A-stable EIS methods. The implicit A-stable methods that we found treat both F and $\dot{F}$ implicitly. We tried to find methods that treat F implicitly and $\dot{F}$ explicitly, but were not able to find A-stable methods of this type. In this section we present two A-stable implicit iEIS+(s,P)$_2$ methods. Both $\mathbf{R}$ and $\hat{\mathbf{R}}$ are diagonal matrices and so the method can be implemented efficiently in parallel, as each stage is computed independently. The coefficients of the methods as well as the $\boldsymbol{\tau}_{P-1}$ vector needed for postprocessing are given below.

A-stable parallel-efficient iEIS+(2,4)$_2$.

$$\mathbf{D}_{1,i} = 0.594710614896760, \quad \mathbf{D}_{2,i} = 0.405289385103240,$$

$$\mathbf{A} = \begin{pmatrix} -2.187376304427630 & -0.964459220078949 \\ -1.117865907067007 & 2.067845436796621 \end{pmatrix},$$

$$\hat{\mathbf{A}} = \begin{pmatrix} 0.778080609332642 & -1.088765766927099 \\ -2.898999040140121 & 1.440243113199464 \end{pmatrix},$$

$$\mathbf{R} = \begin{pmatrix} 3.949190831954959 & 0 \\ 0 & 0.347375777718766 \end{pmatrix},$$

$$\hat{\mathbf{R}} = \begin{pmatrix} -2.706937237458932 & 0 \\ 0 & 0.978108368826293 \end{pmatrix}.$$

The truncation error vector (needed for postprocessing) is

$$\tau_3 = \begin{pmatrix} -3.111010490530440 \\ 4.565012136457357 \end{pmatrix}.$$

A-stable parallel-efficient iEIS+(3,5)$_2$.

$$\mathbf{D} = \begin{pmatrix} 0.439087264857344 & 0.700945256500558 & -0.140032521357901 \\ 0.439087264857344 & 0.700945256500558 & -0.140032521357901 \\ 0.439087264857344 & 0.700945256500558 & -0.140032521357901 \end{pmatrix},$$

$$\mathbf{A} = \begin{pmatrix} 2.507826539020301 & 3.279683213077780 & -1.170881137598611 \\ -0.334032190141782 & -4.031402321497854 & 0.685583668720811 \\ -1.750770284075905 & -4.999999998880823 & 3.295317723260540 \end{pmatrix},$$

$$\hat{\mathbf{A}} = \begin{pmatrix} 2.333968082671988 & 0.419378200972933 & -2.408406401605122 \\ -2.145600247202041 & 0.897829295036851 & -0.721006948644857 \\ -4.988816152192916 & 3.020756581381562 & -1.533772624102988 \end{pmatrix},$$

$$\mathbf{R} = \begin{pmatrix} -3.756922019094389 & 0 & 0 \\ 0 & 4.872890771657239 & 0 \\ 0 & 0 & 4.981825821767937 \end{pmatrix},$$

$$\hat{\mathbf{R}} = \begin{pmatrix} 3.591518759368352 & 0 & 0 \\ 0 & -2.760598976218027 & 0 \\ 0 & 0 & -3.950356833416136 \end{pmatrix}.$$

The truncation error vector (needed for postprocessing) is

$$\tau_4 = \begin{pmatrix} 3.466008686399261 \\ -4.575755330149971 \\ -12.036302018622621 \end{pmatrix}.$$

4. Numerical results. In this section we focus on testing the numerical methods found and presented in section 3. We first verify the convergence properties of the methods presented in sections 3.1 and 3.3, on a system of nonlinear ODEs. We find that these methods achieve the desired convergence rates. Next, we explore the behavior of the SSP methods presented in section 3.2 in terms of preserving the total variation diminishing properties of spatial discretizations, and show that these methods behave as expected on a standard test-case (see [15]).

4.1. Convergence of high order explicit and implicit schemes on a nonlinear test-case. We focus on verifying the convergence of the explicit high order methods eEIS+$(2,6)_2$, eEIS+$(3,7)_2$), and eEIS+$(4,8)_2$ presented in section 3.1 and the implicit high order methods iEIS+$(2,4)_2$ and iEIS+$(3,5)_2$ presented in section 3.3. We show that these methods show the predicted order before and after postprocessing on a nonlinear system of ODEs.

Nonstiff Van der Pol oscillator. The nonlinear system of ODEs is given by

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}' = \begin{pmatrix} y_2 \\ a(1-y_1^2)y_2 - y_1 \end{pmatrix},$$

with $a = 2$ and initial condition $\mathbf{y}(0) = (2,0)^T$. We use the explicit methods eEIS+$(2,6)_2$, eEIS+$(3,7)_2$, eEIS+$(4,8)_2$ to evolve this problem to the final time $T_f = 3.0$ and postprocess the solution at the final time as described in section 2.3. We use three repeats of the truncation error vector ($m = 3$) for all methods except the eEIS$_2$+(2,6) method for which we used $m = 4$. (We note that the iEIS+$(3,5)_2$ method only requires $m = 2$, but $m = 3$ works well so we use it for consistency with the other methods.)

In Figure 4.1 we show the square-root of the sum of squares of the errors in each component for different values of Δt. The slopes of these lines are computed using MATLAB's `polyfit` function and are as follows:

	eEIS+(s,P)$_2$ =			iEIS+(s,P)$_2$ =	
	(2,6)	(3,7)	(4,8)	(2,4)	(3,5)
Without postprocessing	4.7	5.8	7.0	3.0	3.9
After postprocessing	5.8	6.6	7.7	4.0	5.0

Clearly, the orders without postprocessing are of order $P-1$ and after postprocessing are of order P for the eEIS+(s,P)$_2$ and iEIS+(s,P)$_2$ methods found in section 3.1 and 3.3, respectively. This example verifies that numerical solutions from the explicit and implicit error inhibiting methods attain the expected orders of convergence with and without postprocessing. These examples confirm that the mathematical conditions and the methods we found work as expected in practice.

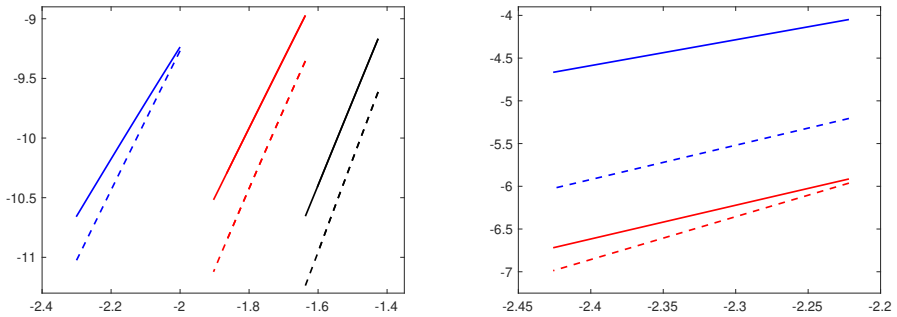

FIG. 4.1. *Convergence of the solution (in solid line) and postprocessed solution (dashed line) from evolving the Van der Pol system to time $T_f = 3.0$ for the nonstiff Van der Pol system. On the y-axis is the $\log_{10}$ of the square-root of the sum of squares of the errors; on the x-axis is $\log_{10}(\Delta t)$. Left: The explicit methods eEIS$_2$+(2,6) (in blue), eEIS$_2$+(3,7) (in red), and eEIS$_2$+(4,8) (in black). Right: The implicit methods eEIS$_2$+(2,4) (in blue), eEIS$_2$+(3,5) (in red).*

4.2. Comparison of total variation behavior of explicit SSP schemes. SSP time-stepping methods are useful for evolving in time hyperbolic PDEs with discontinuities. We focused on finding SSP methods of the form (1.3) that preserve the nonlinear non–inner-product stability properties of the spatial discretization methods when coupled with forward Euler and the Taylor series methods, as in [15]. The eSSP-EIS(s,P)$_2$ and eSSP-EIS+(s,P)$_2$ methods we presented in section 3.2 are tested here on a problem where the spatial discretization is total variation diminishing when coupled with forward Euler time-stepping and when coupled with a Taylor series method. To verify that the methods satisfy the SSP property we examine the maximal rise in total variation when this problem is evolved forward with the eSSP-EIS+(s,p)$_2$ scheme. We further investigate the effect of postprocessing on the total variation of this problem.

We consider the linear advection equation with step function initial conditions:

$$u_t + u_x = 0, \qquad u(0,x) = \begin{cases} 0 & \text{if } 0 \le x \le 1/2, \\ 1 & \text{if } x > 1/2 \end{cases}$$

on the domain $[-1,1]$ with periodic boundary conditions. We use the fact that $u_{tt} = -u_{xt} = -u_{tx} = u_{xx}$ to approximate $\dot{F}$. For the spatial discretization we use a first order forward difference for the first and second derivatives with 200 spatial points:

$$F(u^n)_j := -\frac{u_j^n - u_{j-1}^n}{\Delta x} \approx -U_x(x_j),$$

$$\dot{F}(u^n)_j := \frac{u_j^n - 2u_{j-1}^n + u_{j-2}^n}{\Delta x^2} \approx U_{xx}(x_j).$$

These spatial discretizations satisfy the following.

Forward Euler condition.

$$u_j^{n+1} = u_j^n + \frac{\Delta t}{\Delta x}\left(u_{j-1}^n - u_j^n\right),$$

which is TVD for $\Delta t \le \Delta x$.

Taylor series condition.

$$u_j^{n+1} = u_j^n + \frac{\Delta t}{\Delta x}\left(u_{j-1}^n - u_j^n\right) + \frac{1}{2}\left(\frac{\Delta t}{\Delta x}\right)^2\left(u_{j-2}^n - 2u_{j-1}^n + u_j^n\right),$$

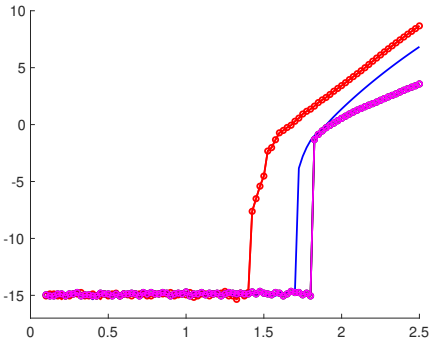

FIG. 4.2. *SSP study: the maximal rise in total variation of the explicit SSP methods SSP-EIS+(2,4)$_2$ (red line), eSSP-EIS(2,3)$_2$ (blue line), and eSSP-EIS+(3,6)$_2$ (magenta line) when used to evolve forward the solution in time of the linear advection equation with a step function initial conditions. On the y-axis is $\log_{10}$ of the maximal rise in total variation, and on the x-axis the CFL number λ. The circles mark the difference between the total variation of the unprocessed solution and that of the postprocessed solution. (Figure in color online.)*

which is TVD for $\Delta t \leq \Delta x$.
Thus $\Delta t_{\text{FE}} = \Delta x$ and in this case we have $K = 1$ in (3.3).

We evolve the solution 10 time-steps using the SSP methods eSSP-EIS(2,3)$_2$, eSSP-EIS+(2,4)$_2$, and eSSP-EIS+(3, 6)$_2$ for different values of Δt. At each time-level y^n we compute the total variation of the solution at time u^n,

$$\|u^n\|_{TV} = \sum_j \left|u^n_{j+1} - u^n_j\right|.$$

The maximal rise in total variation (solid line) is graphed against $\lambda = \frac{\Delta t}{\Delta x}$ in Figure 4.2. As we can see, the value of λ for which the total variation begins to rise is always greater than the guaranteed SSP value.

We then postprocess the solution obtained from the methods eSSP-EIS+(2,4)$_2$ and eSSP-EIS+(3,6)$_2$ at the final time for all values of Δt, and compute the difference between the total variation of the solution at the final time and the postprocessed solution

$$\|u^n\|_{TV} - \|\widehat{u}^n\|_{TV}.$$

We see in Figure 4.2 that the difference between the total variation of the solution at the final time and the postprocessed solution (circle markers) is minimal for all the values of Δt for which the maximal rise in total variation remains bounded. In particular, for values of λ for which the total variation has not started to rise, we observe that the maximal difference between the total variation of the solution and the postprocessed solution remains very small ($\approx 10^{-15}$). Although the postprocessor is only designed to extract a higher order solution but not to preserve the strong stability properties, it is reassuring that the total variation is not adversely impacted by the postprocessor in this case.

5. Conclusions. In this work we extended the theory developed in [12] to the case of two-derivative GLMs. In [12] we showed that if a GLM of the peer-method form satisfies the truncation error conditions to order p and three additional error inhibiting conditions, the error term is one order higher than expected by the truncation error

analysis, and has a particular form. This enabled us to design a postprocessor that eliminates this term in the error, thus extracting an additional order of accuracy. In this work we extend this theory to two-derivative GLMs of the form (1.3) and prove that under the sufficient conditions (2.4a)–(2.4c) we have an error of the form (2.5). This form is exactly the same as the form in [12], except that the value of the truncation error vector is different for the two-derivative GLMs than for the one-derivative GLMs. For this reason, the construction of a postprocessor (in section 2.3) follows immediately from the theory in [12].

Using this theory, we programmed an optimization code in MATLAB that uses the order conditions and error inhibiting conditions as constraints and aims to maximize some stability properties. We generated a number of methods that satisfy the order conditions and the error inhibiting condition (2.4a) or all the error inhibiting conditions (2.4a)–(2.4c). The coefficients of all the methods can be downloaded on our GitHub repository [16]. The explicit methods found by the optimizer either have favorable regions of linear stability or are SSP in the sense of [15]. The implicit methods are A-stable. We test a selection of these methods for convergence, and show that they demonstrate the predicted order before and after postprocessing on a nonlinear system of ODEs. We test the SSP methods for the preservation of the total variation diminishing properties on a PDE with discontinuities, and show that they produce total variation diminishing results beyond the guaranteed SSP time-step, and demonstrated that postprocessing does not adversely affect the total variation. These results validate the methods we found, as well as the theory that we developed.

Acknowledgment. The authors wish to thank Prof. John C. Butcher for many illuminating discussions and for his ongoing advice.

REFERENCES

[1] C. Bresten, S. Gottlieb, Z. Grant, D. Higgs, D. I. Ketcheson, and A. Nemeth, *Explicit strong stability preserving multistep Runge-Kutta methods*, Math. Comp., 86 (2017), pp. 747–769.

[2] J. C. Butcher, S. Gottlieb, and Z. J. Grant, *A B-series analysis of error inhibiting GLMs*, in preparation.

[3] J. C. Butcher, *private communication*, 2020.

[4] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations: Runge–Kutta and General Linear Methods*, John Wiley & Sons, Chichester, UK, 1987.

[5] J. C. Butcher, *Diagonally-implicit multi-stage integration method*, Appl. Numer. Math., 11 (1993), pp. 347–363.

[6] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations*, John Wiley & Sons, Chichester, UK, 2008.

[7] R. P. K. Chan and A. Y. J. Tsai, *On explicit two-derivative Runge-Kutta methods*, Numer. Algorithms, 53 (2010), pp. 171–194.

[8] A. Christlieb, S. Gottlieb, Z. Grant, and D. C. Seal, *Explicit strong stability preserving multistage two-derivative time-stepping schemes*, J. Sci. Comput., 68 (2016), pp. 914–942.

[9] B. Cockburn, M. Luskin, C. W. Shu, and E. Suli, *Enhanced accuracy by post-processing for finite element methods for hyperbolic equations*, Math. Comp., 72 (2002), pp. 577–606.

[10] E. M. Constantinescu, *On the order of general linear methods*, Appl. Math. Lett., 22 (2009), pp. 1425–1428.

[11] A. Ditkowski and S. Gottlieb, *Error inhibiting block one-step schemes for ordinary differential equations*, J. Sci. Comput., 73 (2017), pp. 691–711.

[12] A. Ditkowski, S. Gottlieb, and Z. J. Grant, *Explicit and implicit error inhibiting schemes with post-processing*, Comput. & Fluids, 208 (2020), 104534, https://doi.org/10.1016/j.compfluid.2020.104534.

[13] Z. Du and J. Li, *A Hermite WENO reconstruction for fourth order temporal accurate schemes based on the GRP solver for hyperbolic conservation laws*, J. Comput. Phys., 355 (2018), pp. 385–396.

[14] S. Gottlieb, D. I. Ketcheson, and C.-W. Shu, *Strong Stability Preserving Runge–Kutta and Multistep Time Discretizations*, World Scientific Press, Hackensack, NJ, 2011.
[15] S. Gottlieb, Z. Grant, and D. C. Seal, *Explicit strong stability preserving multistage two-derivative time-stepping schemes*, J. Sci. Comput., 68 (2016), pp. 914–942.
[16] Z. Grant, S. Gottlieb, and A. Ditkowski, *Explicit and Implicit Two-derivative EIS Methods with Post-processing*, GitHub repository, 2019, https://github.com/EISmethods/EIS_2derivative.
[17] B. Gustafsson, H.-O. Kreiss, and J. Oliger, *Time Dependent Problems and Difference Methods*, Pure Appl. Math. (N. Y.) 24, John Wiley & Sons, New York, 1995.
[18] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, Springer Ser. Comput. Math. 31, Springer-Verlag, Berlin, Heidelberg, 2006.
[19] Z. Jackiewicz, *General Linear Methods for Ordinary Differential Equations*, John Wiley & Sons, Hoboken, NJ, 2009.
[20] J.-P. Suarez and G. B. Jacobs, *Regularization of singularities in the weighted summation of Dirac-delta functions for the spectral solution of hyperbolic conservation laws*, J. Sci. Comput., 72 (2017), pp. 1080–1092.
[21] K. Kastlunger and G. Wanner, *On Turan type implicit Runge-Kutta methods*, Computing (Arch. Elektron. Rechnen), 9 (1972), pp. 317–325.
[22] K. H. Kastlunger and G. Wanner, *Runge Kutta processes with multiple nodes*, Computing (Arch. Elektron. Rechnen), 9 (1972), pp. 9–24.
[23] I. P. E. Kinnmark and W. G. Gray, *One step integration methods with maximum stability regions*, Math. Comput. Simulation, 26 (1984), pp. 87–92.
[24] G. Yu. Kulikov, *On quasi-consistent integration by Nordsieck methods*, J. Comput. Appl. Math., 225 (2009), pp. 268–287.
[25] G. Yu. Kulikov and R. Weiner, *Variable-stepsize interpolating explicit parallel peer methods with inherent global error control*, SIAM J. Sci. Comput., 32 (2010), pp. 1695–1723, https://doi.org/10.1137/090764840.
[26] G. Yu. Kulikov and R. Weiner, *Doubly quasi-consistent fixed-stepsize numerical integration of stiff ordinary differential equations with implicit two-step peer methods*, J. Comput. Appl. Math., 340 (2018), pp. 256–275.
[27] J. Li and Z. Du, *A two-stage fourth order time-accurate discretization for Lax–Wendroff type flow solvers* I. *Hyperbolic conservation laws*, SIAM J. Sci. Comput., 38 (2016), pp. A3046–A3069, https://doi.org/10.1137/15M1052512.
[28] T. Mitsui, *Runge-Kutta type integration formulas including the evaluation of the second derivative.* I, Publ. Res. Inst. Math. Sci., 18 (1982), pp. 325–364.
[29] A. Moradi, J. Farzi, and A. Abdi, *Strong Stability Preserving Second Derivative General Linear Methods*, J. Sci. Comput., 81 (2019), pp. 392–435.
[30] T. Nguyen-Ba, H. Nguyen-Thu, T. Giordano, and R. Vaillancourt, *One-step strong-stability-preserving Hermite-Birkhoff-Taylor methods*, Scientific Journal of Riga Technical University, 45 (2010), pp. 95–104.
[31] T. J. T. Norton, *Structure-preserving General Linear Methods*, Doctoral Thesis, University of Bath, Bath, UK, 2015.
[32] N. Obreschkoff, *Neue Quadraturformeln*, Abh. Preuss. Akad. Wiss. Math.-Nat. Kl., 1940 (1940), 4.
[33] R. I. Okuonghae and M. N. O. Ikhile, *L(α)-stable multi-derivative GLM*, J. Algorithms Comput. Technol., 9 (2015), pp. 339–376.
[34] H. Ono and T. Yoshida, *Two-stage explicit Runge-Kutta type methods using derivatives*, Japan J. Indust. Appl. Math., 21 (2004), pp. 361–374.
[35] L. Pan, K. Xu, Q. Li, and J. Li, *An efficient and accurate two-stage fourth-order gas-kinetic scheme for the Euler and Navier–Stokes equations*, J. Comput. Phys., 326 (2016), pp. 197–221.
[36] A. Quarteroni, R. Sacco, and F. Saleri, *Numerical Mathematics*, Texts Appl. Math. 37, Springer-Verlag, New York, 2010.
[37] H. Mirzaee, J. K. Ryan, and R. M. Kirby, *Efficient implementation of smoothness-increasing accuracy-conserving (SIAC) filters for discontinuous Galerkin solutions*, J. Sci. Comput., 52 (2012), pp. 85–112.
[38] D. C. Seal, Y. Guclu, and A. J. Christlieb, *High-order multiderivative time integrators for hyperbolic conservation laws*, J. Sci. Comput., 60 (2014), pp. 101–140.
[39] C. W. Shu and P. S. Wong, *A note on the accuracy of spectral method applied to nonlinear conservation laws*, J. Sci. Comput., 10 (1995), pp. 357–369.
[40] R. D. Skeel, *Analysis of fixed-stepsize methods*, SIAM J. Numer. Anal., 13 (1976), pp. 664–685,

https://doi.org/10.1137/0713055.
[41] H. Shintani, *On one-step methods utilizing the second derivative*, Hiroshima Math. J., 1 (1971), pp. 349–372.
[42] H. Shintani, *On explicit one-step methods utilizing the second derivative*, Hiroshima Math. J., 2 (1972), pp. 353–368.
[43] B. Soleimani and R. Weiner, *A class of implicit peer methods for stiff systems*, J. Comput. Appl. Math., 316 (2017), pp. 358–368.
[44] D. Stoffer, *General linear methods: Connection to one step methods and invariant curves*, Numer. Math., 64 (1993), pp. 395–407.
[45] E. Süli and D. F. Mayers, *An Introduction to Numerical Analysis*, Cambridge University Press, Cambridge, 2003.
[46] D. D. Stancu and A. H. Stroud, *Quadrature formulas with simple Gaussian nodes and multiple fixed nodes*, Math. Comp., 17 (1963), pp. 384–394.
[47] A. Y. J. Tsai, R. P. K. Chan, and S. Wang, *Two-derivative Runge–Kutta methods for PDEs using a novel discretization approach*, Numer. Algorithms, 65 (2014), pp. 687–703.
[48] P. Turán, *On the theory of the mechanical quadrature*, Acta Sci. Math. (Szeged), 12 (1950), pp. 30–37.
[49] R. Weiner, G. Yu. Kulikov, and H. Podhaisky, *Variable-stepsize doubly quasi-consistent parallel explicit peer methods with global error control*, Appl. Numer. Math., 62 (2012), pp. 1591–1603.
[50] R. Weiner, B. A. Schmitt, H. Podhaisky, and S. Jebensc, *Superconvergent explicit two-step peer methods*, J. Comput. Appl. Math., 223 (2009), pp. 753–764.
[51] B. W. Wissink, G. B. Jacobs, J. K. Ryan, W. S. Don, and E. T. A. van der Weide, *Shock regularization with smoothness-increasing accuracy-conserving Dirac-delta polynomial kernels*, J. Sci. Comput., 77 (2018), pp. 579–596.