

Efficient time discretization for local discontinuous Galerkin methods

Yinhua Xia*, Yan Xu[†] and Chi-Wang Shu[‡]

Abstract

In this paper, we explore a few efficient time discretization techniques for the local discontinuous Galerkin (LDG) methods to solve partial differential equations (PDEs) with higher order spatial derivatives. The main difficulty is the stiffness of the LDG spatial discretization operator, which would require a unreasonably small time step for an explicit local time stepping method. We focus our discussion on the semi-implicit spectral deferred correction (SDC) method, and study its stability and accuracy when coupled with the LDG spatial discretization. We also discuss two other time discretization techniques, namely the additive Runge-Kutta (ARK) method and the exponential time differencing (ETD) method, coupled with the LDG spatial discretization. A comparison is made among these three time discretization techniques, to conclude that all three methods are efficient when coupled with the LDG spatial discretization for solving PDEs containing higher order spatial derivatives.

Key words: spectral deferred correction method; additive Runge-Kutta method; exponential time differencing method; local discontinuous Galerkin method; higher order spatial derivatives

AMS subject classification: 65L06, 65M60

1 Introduction

In this paper we are concerned with efficient time discretization of the local discontinuous Galerkin (LDG) method for solving time dependent partial differential equations containing higher order spatial derivatives.

*Department of Mathematics, University of Science and Technology of China, Hefei, Anhui 230026, P.R. China. E-mail: xiayh@mail.ustc.edu.cn

[†]Department of Applied Mathematics, University of Twente, P.O. Box 217, 7500 AE, Enschede, The Netherlands. E-mail: y.xu@math.utwente.nl

[‡]Division of Applied Mathematics, Brown University, Providence, RI 02912, USA. E-mail: shu@dam.brown.edu. Research supported in part by the Chinese Academy of Sciences while this author was visiting the University of Science and Technology of China (grant 2004-1-8) and the Institute of Computational Mathematics and Scientific/Engineering Computing. Additional support was provided by ARO grant W911NF-04-1-0291 and NSF grant DMS-0510345.

The discontinuous Galerkin (DG) method is a class of finite element methods, using discontinuous, piecewise polynomials as the solution and the test space. It was first designed as a method for solving hyperbolic conservation laws containing only first order spatial derivatives, e.g. Reed and Hill [23] for solving linear equations, and Cockburn et al. [8, 7, 6, 9] for solving nonlinear equations.

The LDG method is an extension of the DG method aimed at solving partial differential equations (PDEs) containing higher than first order spatial derivatives. The first LDG method was constructed by Cockburn and Shu in [10] for solving nonlinear convection diffusion equations containing second order spatial derivatives. Their work was motivated by the successful numerical experiments of Bassi and Rebay [4] for the compressible Navier-Stokes equations. Yan and Shu developed an LDG method for a general KdV type equation (containing third order spatial derivatives) in [29], and they generalized the LDG method to PDEs with fourth and fifth order spatial derivatives in [30]. Levy, Shu and Yan [21] developed LDG methods for nonlinear dispersive equations that have compactly supported traveling wave solutions, the so-called “compactons”. More recently, Xu and Shu [25, 26, 27, 28] further developed the LDG method to solve many nonlinear wave equations with higher order derivatives, including the general KdV-Burgers type equations, the general fifth-order KdV type equations, the fully nonlinear $K(n, n, n)$ equations, the generalized nonlinear Schrödinger equations, the coupled nonlinear Schrödinger equations, the Kuramoto-Sivashinsky equations, the Ito-type coupled KdV equations, the Kadomtsev-Petviashvili equation, and the Zakharov-Kuznetsov equation. A common feature of these LDG methods is that stability can be proved for quite general nonlinear cases. DG and LDG methods also have several attractive properties, such as their flexibility for arbitrary h and p adaptivity and their excellent parallel efficiency.

The application of the DG or LDG method discretizes the spatial variables and generates a large coupled system of ordinary differential equations (ODEs). One would then need to use a suitable ODE solver to discretize the time variable. For hyperbolic conservation laws and convection dominated convection diffusion equations, explicit and nonlinearly stable Runge-Kutta time discretizations [24, 16] are suitable choices. The resulting fully discretized scheme, termed Runge-Kutta DG (RKDG) method [8], are stable, efficient and accurate for solving such convection or convection dominated problems. However, for PDEs containing higher order spatial derivatives, especially when the coefficients in front of these higher order spatial derivative terms are not small, such explicit and local time discretization will suffer from extremely small time step restriction for stability, due to the stiffness of the spatial LDG operator which approximates both the lower and higher order spatial derivatives. Often, such a small time step is not needed for the purpose of accuracy and is purely an artifact of the explicit time discretization technique. It would therefore be desirable to use either nonlocal or implicit time discretization techniques to alleviate this problem. Implicit time discretization is already used in [30] for some of the numerical experiments. In [27, 28], the exponential time differencing (ETD) method [13, 18], which is an explicit but non-local time discretization method, is used for some of the test cases. A suitable time discretization technique for the LDG methods solving PDEs with higher order spatial derivatives should have good stiffness stability and should be easy to implement. Such methods are usually implicit [17].

The ODE system obtained from the LDG spatial discretization for a PDE containing

higher order spatial derivatives often includes both stiff and non-stiff terms due to a disparity in time scales. An efficient time discretization technique is a semi-implicit method treating the non-stiff terms (often low-order but nonlinear) explicitly and the stiff terms (often higher order but linear) implicitly. This is much easier to implement and much less in computational cost than the fully implicit time discretization methods. If only the convection terms are nonlinear, the semi-implicit methods are as effective as the fully implicit methods in allowing large time steps.

The main time discretization method explored in this paper is the spectral deferred correction (SDC) method constructed by Dutt, Greengard and Rokhlin [15]. An advantage of this method is that it is a one step method (namely, to march to time level $n+1$ one would only need to store the value of the solution at time level n) and can be constructed easily and systematically for any order of accuracy. This is in contrast to Runge-Kutta methods which are more difficult to construct for higher order of accuracy. The SDC method relies on the Picard integral equation and is driven iteratively by the base Euler forward (for non-stiff problems) or Euler backward (for stiff problems) method. It has been shown in [15, 22] that the implicit SDC method driven by the Euler backward method is A -stable or $A(\alpha)$ -stable, with α close to 90° . In [15] an L -stable implicit SDC method is also constructed, which is a combination of two different SDC methods and hence doubles the cost of the usual SDC method. We present in this paper a different way to construct an L -stable implicit SDC method with a smaller computational cost than that in [15].

Two other techniques for time discretization of the LDG methods are also explored in this paper. One is the additive Runge-Kutta method given in [19], which has nice stiffness stability. The other is the exponential time differencing (ETD) method given in [13, 18], which is an explicit but nonlocal time discretization. We perform numerical experiments to compare these three different techniques of time discretization, and conclude that they are all competitive when solving ODEs from the LDG discretization of PDEs containing higher order spatial derivatives.

We remark that, for stiff problems, implicit Runge-Kutta methods possess excellent stability properties. But they are very expensive when high order accuracy is required except for the diagonally implicit Runge-Kutta methods [1, 2, 20, 17]. The additive Runge-Kutta method [19] that we explore in this paper belongs to this class. Implicit multi-step algorithms can be easily constructed for very high order of accuracy, but they have poor stability properties.

The organization of the paper is as follows. In Section 2, we give a description of the spectral deferred correction (SDC) method. In Section 3, we give a brief introduction to the additive Runge-Kutta (ARK) and exponential time differencing (ETD) methods. Section 4 is devoted to a concise description of the local discontinuous Galerkin (LDG) spatial discretization via an example. Numerical examples are presented in Section 5, testing the performance of the three different time discretization techniques with the LDG spatial discretization for a series of PDEs with higher order spatial derivatives including the Korteweg-de Vries (KdV) equation, the Kuramoto-Sivashinsky equation and the Kawahara equation. Finally, we give concluding remarks in Section 6.

2 The spectral deferred correction method

Dutt, Greengard and Rokhlin presented a new variation of the classical method of deferred correction, called the spectral deferred correction (SDC) method, in [15]. It is based on low order time integration methods, which are corrected iteratively, with the order of accuracy increased by one for each additional iteration. Attractively, we can split stiff and non-stiff terms as needed and treat them differently (implicitly for the stiff terms and explicitly for the non-stiff terms). Minion presented the semi-implicit SDC (SISDC) method in [22]. In the following we will introduce these methods.

Consider the ODE system

$$\begin{cases} u_t = F(t, u(t)), & t \in [0, T], \\ u(0) = u_0, \end{cases} \quad (2.1)$$

where $u_0, u(t) \in \mathbb{C}^n$ and $F : \mathbb{R} \times \mathbb{C}^n \rightarrow \mathbb{C}^n$. $F \in C^1(\mathbb{R} \times \mathbb{C}^n)$, which is sufficient to guarantee local existence and uniqueness of the solution to (2.1).

Suppose now the time interval $[0, T]$ is divided into N intervals by the partition $0 = t_0 < t_1 < \dots < t_n < \dots < t_N = T$. Let $\Delta t_n \equiv t_{n+1} - t_n$ and u_n denotes the numerical approximation of $u(t_n)$, with $u_0 = u(0)$.

Rewrite (2.1) into an integral form in the subinterval $[t_n, t_{n+1}]$:

$$u(t_{n+1}) = u(t_n) + \int_{t_n}^{t_{n+1}} F(\tau, u(\tau)) d\tau. \quad (2.2)$$

Then divide the time interval $[t_n, t_{n+1}]$ into P subintervals by choosing the points $t_{n,m}$ for $m = 0, 1, \dots, P$ such that $t_n = t_{n,0} < t_{n,1} < \dots < t_{n,m} < \dots < t_{n,P} = t_{n+1}$. Let $\Delta t_{n,m} = t_{n,m+1} - t_{n,m}$ and $u_{n,m}^k$ denotes the k^{th} order approximation to $u(t_{n,m})$. To avoid the instability of approximation at equispaced nodes for high order accuracy, the points $\{t_{n,m}\}_{m=0}^P$ are chosen to be the Gauss-Lobatto nodes on $[t_n, t_{n+1}]$. We can also use the Gauss-Legendre, Gauss-Radau or Chebyshev nodes. Starting from u_n , we give the algorithm to calculate u_{n+1} in the following.

Compute the initial approximation

$$u_{n,0}^1 = u_n.$$

For non-stiff/stiff problems, use the forward/backward Euler method to compute a first order accurate approximate solution u^1 at the nodes $\{t_{n,m}\}_{m=1}^P$.

For $m = 0, \dots, P - 1$

1. For non-stiff problems,

$$u_{n,m+1}^1 = u_{n,m}^1 + \Delta t_{n,m} F(t_{n,m}, u_{n,m}^1) \quad (2.3)$$

2. For stiff problems,

$$u_{n,m+1}^1 = u_{n,m}^1 + \Delta t_{n,m} F(t_{n,m+1}, u_{n,m+1}^1) \quad (2.4)$$

Compute successive corrections

For $k = 1, \dots, K$

$$u_{n,0}^{k+1} = u_n.$$

For $m = 0, \dots, P - 1$

1. For non-stiff problems, with $0 \leq \theta \leq 1$,

$$u_{n,m+1}^{k+1} = u_{n,m}^{k+1} + \theta \Delta t_{n,m} (F(t_{n,m}, u_{n,m}^{k+1}) - F(t_{n,m}, u_{n,m}^k)) + I_m^{m+1}(F(t, u^k)), \quad (2.5)$$

2. For stiff problems, with $\frac{1}{2} \leq \theta \leq 1$,

$$u_{n,m+1}^{k+1} = u_{n,m}^{k+1} + \theta \Delta t_{n,m} (F(t_{n,m+1}, u_{n,m+1}^{k+1}) - F(t_{n,m+1}, u_{n,m+1}^k)) + I_m^{m+1}(F(t, u^k)), \quad (2.6)$$

where $I_m^{m+1}(F(t, u^k))$ is the integral of the P -th degree interpolating polynomial on the $P+1$ points $(t_{n,m}, F(t_{n,m}, u_{n,m}^k))_{m=0}^P$ over the subinterval $[t_{n,m}, t_{n,m+1}]$, which is the numerical quadrature approximation of

$$\int_{t_{n,m}}^{t_{n,m+1}} F(\tau, u(\tau)) d\tau. \quad (2.7)$$

Finally we have $u_{n+1} = u_{n,P}^{K+1}$.

Definition 2.1. For non-stiff problems, the method (2.3)-(2.5) will be called the explicit $SDC_P^K(\theta)$ scheme. For stiff problems, the method (2.4)-(2.6) will be called the implicit $SDC_P^K(\theta)$ scheme.

Lemma 2.1 (Local Truncation Error). *The Local Truncation Error (LTE) obtained with the explicit or the implicit $SDC_P^K(\theta)$ scheme is*

$$\mathcal{O}(h^{\min[K+1, P+1]}) \quad (2.8)$$

where $h = \max_{n,m} \Delta t_{n,m}$.

Proof: First, we consider the LTE of the explicit SDC method.

Assume $u_{n,0}^k = u(t_n)$, $k = 1, \dots, K+1$. Taking the difference of

$$u(t_{n,m+1}) = u(t_{n,m}) + \int_{t_{n,m}}^{t_{n,m+1}} F(\tau, u(\tau)) d\tau \quad (2.9)$$

and (2.5), we have

$$\begin{aligned} u(t_{n,m+1}) - u_{n,m+1}^{k+1} &= u(t_{n,m}) - u_{n,m}^{k+1} - \theta \Delta t_{n,m} (F(t_{n,m}, u_{n,m}^{k+1}) - F(t_{n,m}, u_{n,m}^k)) \\ &\quad + \int_{t_{n,m}}^{t_{n,m+1}} F(\tau, u(\tau)) d\tau - I_m^{m+1}(F(t, u^k)) \end{aligned} \quad (2.10)$$

By induction on both m and k , we assume, when $k \leq P$,

$$u(t_{n,l}) - u_{n,l}^k = \mathcal{O}(h^{k+1}), \quad \forall l \text{ in level } k, \quad (2.11)$$

$$u(t_{n,l}) - u_{n,l}^{k+1} = \mathcal{O}(h^{k+2}), \quad \text{for } l \leq m \text{ in level } k+1. \quad (2.12)$$

Then

$$u(t_{n,m}) - u_{n,m}^{k+1} = \mathcal{O}(h^{k+2}), \quad (2.13)$$

$$\theta \Delta t_{n,m} (F(t_{n,m}, u_{n,m}^{k+1}) - F(t_{n,m}, u_{n,m}^k)) = \mathcal{O}(h^{k+2}), \quad (2.14)$$

$$\begin{aligned} \int_{t_{n,m}}^{t_{n,m+1}} F(\tau, u(\tau)) d\tau - I_m^{m+1}(F(t, u^k)) &= \left[\int_{t_{n,m}}^{t_{n,m+1}} F(\tau, u(\tau)) d\tau - I_m^{m+1}(F(t, u)) \right] \\ &\quad + [I_m^{m+1}(F(t, u)) - I_m^{m+1}(F(t, u^k))] \\ &= \mathcal{O}(h^{P+2}) + \mathcal{O}(h^{k+2}) \\ &= \mathcal{O}(h^{k+2}), \end{aligned} \quad (2.15)$$

where we have used the fact that $I_m^{m+1}(F(t, u))$ is the integral of the P -th degree interpolating polynomial on $(t_{n,m}, F(u(t_{n,m})))_{m=0}^P$ over the subinterval $[t_{n,m}, t_{n,m+1}]$, hence it is accurate to the order $\mathcal{O}(h^{P+2})$. Thus we have

$$u(t_{n,m+1}) - u_{n,m+1}^{k+1} = \mathcal{O}(h^{\min[k+2, P+2]}). \quad (2.16)$$

The proof is the same for the implicit $SDC_P^K(\theta)$ method. \square

Remark 2.1. In the proof of Lemma 2.1, we find the terms $\theta \Delta t_m (F(t_{n,m}, u_m^{k+1}) - F(t_{n,m}, u_m^k))$ in (2.5) and $\theta \Delta t_m (F(t_{n,m+1}, u_{m+1}^{k+1}) - F(t_{n,m+1}, u_{m+1}^k))$ in (2.6) do not affect the accuracy. They are added purely for the purpose of enhancing stability. When $\theta = 1$, the methods are the original ones presented in [15, 22].

Remark 2.2. In the proof of Lemma 2.1, where we assume $\{u_{n,0}^k = u(t_n)\}_{k=1}^{K+1}$, if $K = P$ and after we have obtained $\{u_{n,m}^{K+1} - u(t_{n,m}) = \mathcal{O}(h^{P+2})\}_{m=0}^P$, through the following Gauss-Lobatto quadrature

$$u_{n+1} = u_{n,0} + I_0^P(F(t, u^{K+1})), \quad (2.17)$$

we have $u_{n+1} - u(t_{n+1}) = \mathcal{O}(h^{P+3})$ because

$$\begin{aligned} \int_{t_n}^{t_{n+1}} F(\tau, u(\tau)) d\tau - I_0^P(F(t, u^{K+1})) &= \left[\int_{t_n}^{t_{n+1}} F(\tau, u(\tau)) d\tau - I_0^P(F(t, u)) \right] \\ &\quad + [I_0^P(F(t, u)) - I_0^P(F(t, u^{K+1}))] \\ &= \mathcal{O}(h^{2P+1}) + \mathcal{O}(h^{K+3}) \\ &= \mathcal{O}(h^{P+3}), \quad P \geq 2. \end{aligned} \quad (2.18)$$

where we have used the fact that the Gauss-Lobatto quadrature over the whole interval $[t_n, t_{n+1}]$ is accurate of order $\mathcal{O}(h^{2P+1})$. Therefore, the $SDC_P^P(\theta)$ scheme coupled with (2.17) has the LTE one order higher $\mathcal{O}(h^{P+2})$. From Remark 2.1, the usage of (2.17) is equivalent to adding one additional correction step $k = P + 1$. We denote this version of the scheme by $SDC_P^{P+1}(\theta)$, which has the LTE $\mathcal{O}(h^{P+2})$ when $P \geq 2$. This is the version of the scheme actually used in the numerical experiments in Section 5.

From Lemma 2.1, Remark 2.2 and the Lax stability for one-step time integrated methods, we have

Theorem 2.2. *The explicit or implicit $SDC_P^K(\theta)$ methods are $\min(K + 1, P + 2)$ order accurate methods for $P \geq 2$. When $P = 1$, the $SDC_P^K(\theta)$ methods are $\min(K + 1, P + 1)$ order accurate.*

When the right hand side $F(t, u)$ of the ODE (2.1) can be written as the sum of a non-stiff term $F_N(t, u)$ and a stiff term $F_S(t, u)$, we have

$$\begin{cases} u_t = F_N(t, u(t)) + F_S(t, u(t)), & t \in [0, T], \\ u(0) = u_0. \end{cases} \quad (2.19)$$

The construction of the semi-implicit SDC (SISDC) method is simply

$$\begin{aligned} u_{n,m+1}^{k+1} = & u_{n,m}^{k+1} + \theta_1 \Delta t_{n,m} (F_N(t_{n,m}, u_{n,m}^{k+1}) - F_N(t_{n,m}, u_{n,m}^k)) \\ & + \theta_2 \Delta t_{n,m} (F_S(t_{n,m+1}, u_{n,m+1}^{k+1}) - F_S(t_{n,m+1}, u_{n,m+1}^k)) + I_m^{m+1}(F(t, u^k)), \end{aligned} \quad (2.20)$$

where θ_1 and θ_2 are taken to be the same number θ in our numerical experiments in Section 5.

Generally, the numerical methods for stiff problems are analyzed by applying it to the equation

$$\begin{cases} u_t = \lambda u(t), & t \in [0, 1], \\ u(0) = 1, \end{cases} \quad (2.21)$$

where $\lambda \in \mathbb{C}$ and $\Delta t_n = 1$. The **amplification factor**, $Am(\lambda)$, is defined by the formula

$$Am(\lambda) = u_1, \quad (2.22)$$

where u_1 is the numerical solution of $u(1)$. If $|Am(\lambda)| \leq 1$, then the numerical method is said to be stable for the given value of λ . If the method is stable for all λ in the left-half plane ($Re(\lambda) \leq 0$), then the method is said to be **A-stable**. A method is said to be **A(α)-stable** if it is stable for all λ such that $\pi - \alpha \leq arg(\lambda) \leq \pi + \alpha$. A method is said to be **L-stable** if

$$\lim_{\lambda \rightarrow -\infty} Am(\lambda) = 0. \quad (2.23)$$

We have plotted the stability regions of the implicit schemes $SDC_P^K(\theta)$ for several choices of the parameters P and K ($SDC_1^1(\theta)$, $SDC_2^2(\theta)$, $SDC_2^3(\theta)$, $SDC_3^3(\theta)$, $SDC_3^4(\theta)$, $SDC_4^4(\theta)$ and $SDC_4^5(\theta)$), with different values of θ ($\theta = 0.8, 0.9$ and 1.0). These schemes are **A-stable**, but not **L-stable**.

As in [15], we have the following theorem.

Theorem 2.3. For any pair of natural numbers P, K , the amplification factor $Am(\lambda)$ associated with the implicit scheme $SDC_P^K(\theta)$ is a rational function of λ . Moreover, there exists a real number $\mu(P, K, \theta)$ such that

$$\lim_{\lambda \rightarrow -\infty} Am(\lambda) = \mu(P, K, \theta). \quad (2.24)$$

This theorem provides a mechanism for combining two different schemes with different P, K and θ to obtain L -stable schemes [15].

Corollary 2.4. Suppose $\mu(P_1, K_1, \theta_1) \neq \mu(P_2, K_2, \theta_2)$, then the implicit scheme $SDC_{P_1, P_2}^{K_1, K_2}(\theta_1, \theta_2)$ defined by the formula

$$SDC_{P_1, P_2}^{K_1, K_2}(\theta_1, \theta_2) = \frac{\mu(P_1, K_1, \theta_1) SDC_{P_2}^{K_2}(\theta_2) - \mu(P_2, K_2, \theta_2) SDC_{P_1}^{K_1}(\theta_1)}{\mu(P_1, K_1, \theta_1) - \mu(P_2, K_2, \theta_2)} \quad (2.25)$$

is L -stable.

When the signs of $\mu(P_1, K_1, \theta_1)$ and $\mu(P_2, K_2, \theta_2)$ are opposite, $SDC_{P_1, P_2}^{K_1, K_2}(\theta_1, \theta_2)$ is A -stable if $SDC_{P_1}^{K_1}(\theta_1)$ and $SDC_{P_2}^{K_2}(\theta_2)$ are both A -stable, since $SDC_{P_1, P_2}^{K_1, K_2}(\theta_1, \theta_2)$ is a convex combination of $SDC_{P_1}^{K_1}(\theta_1)$ and $SDC_{P_2}^{K_2}(\theta_2)$.

Remark 2.3. In the implicit scheme $SDC_P^K(\theta)$, if we change θ to θ_1 only for one sub-interval in one of the correction steps, e.g. in the last step $k = K, m = P - 1$

$$u_{n, P}^{K+1} = u_{n, P-1}^{K+1} + \theta_1 \Delta t_{n, P-1} (F(t_{n, P}, u_{n, P}^{K+1}) - F(t_{n, P}, u_{n, P}^K)) + I_{P-1}^P(F(t, u^K)) \quad (2.26)$$

then the corresponding $\mu(P, K, \theta, \theta_1)$ of this modified scheme is different from $\mu(P, K, \theta)$. If we combine the original $SDC_P^K(\theta)$ scheme with this modified scheme as in Corollary 2.4, we obtain a new scheme, denoted by $SDC_P^K(\theta, \theta_1)$, which is L -stable but involves much less computational cost than (2.25). In fact, (2.25) involves twice the cost of the original scheme while $SDC_P^K(\theta, \theta_1)$ involves only a factor $1 + \frac{1}{P(K+1)}$ of the original cost.

In Section 5, we give a comparison among the L -stable $SDC_{2,3}^{3,3}(1, 1)$ and $SDC_2^3(1, 0.8)$ schemes, and the non- L -stable $SDC_2^3(1)$ scheme, for different time steps in Example 5.2, to demonstrate the advantage of L -stable schemes for larger time steps in reducing the errors.

3 The additive Runge-Kutta and exponential time differencing methods

In this section, we introduce the additive Runge-Kutta (ARK) and exponential time differencing (ETD) methods.

Following the work [3, 19], the ARK methods are used to solve equation (2.19). They are given in the following form

$$u^{(i)} = u_n + \Delta t \sum_{j=0}^{i-1} a_{ij}^{[N]} F_N(t_n + c_j \Delta t, u^{(j)}) + \Delta t \sum_{j=0}^i a_{ij}^{[S]} F_S(t_n + c_j \Delta t, u^{(j)}), \quad i = 1, \dots, s \quad (3.1)$$

$$u_{n+1} = u_n + \Delta t \sum_{j=0}^s b_j^{[N]} F_N(t_n + c_j \Delta t, u^{(j)}) + \Delta t \sum_{j=0}^s b_j^{[S]} F_S(t_n + c_j \Delta t, u^{(j)}), \quad (3.2)$$

where $u^{(0)} = u_n$ and $u^{(i)}$ approximates $u(t_n + c_i \Delta t)$. The non-stiff and stiff terms are integrated by their own $(s+1)$ -stage Runge-Kutta methods respectively. The Butcher coefficients $a_{ij}^{[N]}$, $a_{ij}^{[S]}$, $b_j^{[N]}$, $b_j^{[S]}$ and c_j are constrained by order of accuracy and stability considerations. In [19], the implicit-explicit additive Runge-Kutta (ARK) methods from third- to fifth-order are presented in which the stiff terms are integrated by an L-stable, stiffly-accurate, singly diagonally implicit Runge-Kutta method while the non-stiff terms are integrated with a traditional explicit Runge-Kutta method. For a detailed description of the methods as well as their implementation and applications, we refer the readers to [19].

The ETD method was constructed to solve the equation of the following form

$$u_t = \mathcal{L}u + \mathcal{F}(t, u) \quad (3.3)$$

where \mathcal{L} is a linear term and \mathcal{F} is nonlinear. Many interesting equations are of this form, where typically \mathcal{L} represents the stiff part of the equation. When discretizing in space we obtain a system of ODEs of the form

$$u_t = Lu + F(t, u) \quad (3.4)$$

The exponential time differencing (ETD) methods can be described in the context of solving (3.4). Integrating the equation over a single time step from $t = t_n$ to $t_{n+1} = t_n + \Delta t$, we get

$$u(t_{n+1}) = e^{L\Delta t} u(t_n) + e^{L\Delta t} \int_0^{\Delta t} e^{-L\tau} F(t_n + \tau, u(t_n + \tau)) d\tau \quad (3.5)$$

The equation (3.5) is exact. We denote the numerical approximation to $u(t_n)$ by u_n and write $F(t_n, u_n)$ as F_n . The simplest approximation to the integral (3.5) is to take F as a constant, $F = F_n + \mathcal{O}(\Delta t)$, on the interval $t_n \leq t \leq t_{n+1}$. Then we obtain the first order scheme ETD1, given by

$$u_{n+1} = e^{L\Delta t} u_n + L^{-1}(e^{L\Delta t} - I) F_n. \quad (3.6)$$

If instead of assuming that F is constant over the interval $t_n \leq t \leq t_{n+1}$, we use the linear approximation that

$$F = F_n + t(F_n - F_{n-1})/\Delta t + \mathcal{O}(\Delta t^2). \quad (3.7)$$

Then we obtain the second order scheme ETD2, given by

$$u_{n+1} = e^{L\Delta t} u_n + \frac{1}{\Delta t} L^{-2} ((I + \Delta t L) e^{L\Delta t} - I - 2\Delta t L) F_n + \frac{1}{\Delta t} L^{-2} (-e^{L\Delta t} + I + \Delta t L) F_{n-1}. \quad (3.8)$$

A second order ETD method of Runge-Kutta type, analogous to the ETD2 method, is as follows. First, the ETD1 (3.6) is taken to give

$$a_n = e^{L\Delta t} u_n + L^{-1} (e^{L\Delta t} - I) F_n. \quad (3.9)$$

Then the approximation

$$F = F(t_n, u_n) + (t - t_n)(F(t_n + \Delta t, a_n) - F(t_n, u_n))/\Delta t + O(\Delta t^2) \quad (3.10)$$

is applied on the interval $t_n \leq t \leq t_{n+1}$, and is substituted into (3.5) to yield the scheme ETD2RK given by

$$u_{n+1} = a_n + \frac{1}{\Delta t} L^{-2} (e^{L\Delta t} - I - \Delta t L) (F(t_n + \Delta t, a_n) - F_n). \quad (3.11)$$

Cox and Matthews derive a set of ETD methods bases on Runge-Kutta time-stepping, which they call ETDRK schemes in [13], among general formulas for ETD-schemes to arbitrary order. In a recent paper Kassam and Trefethen [18] compare various fourth order methods for solving equations of the form (3.4), and conclude that the best by a clear margin was a modification to the ETDRK schemes. In essence, for nonlinear time dependent equations, the ETD schemes provide a systematic coupling of the explicit treatment of nonlinearities and the implicit and possibly exact integration of the stiff linear parts of the equations, while achieving high accuracy and maintaining good stability.

To overcome the vulnerability of the error cancellations in the high order ETD and ETDRK schemes, and to generalize the ETD schemes to non-diagonal problems, in [18], modified ETD schemes are proposed by using the complex contour integrals

$$f(L) = \frac{1}{2\pi i} \int_{\Gamma} f(z)(zI - L)^{-1} dz \quad (3.12)$$

on a suitable contour Γ to evaluate the coefficients (e.g. $f(L) = L^{-2}(e^{L\Delta t} - I - \Delta t L)$ in (3.11)) in the update formula for ETDRK. We choose the unit circle in the complex space as Γ and the trapezoidal rule to approximate (3.12) in Section 5.

4 The local discontinuous Galerkin method

In this section, we give a simple example to illustrate the essential ideas of the local discontinuous Galerkin method [10]. Consider the following convection diffusion equation

$$u_t + f(u)_x - u_{xx} = 0. \quad (4.1)$$

Let us denote the mesh by $I_j = [x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}]$ for $j = 1, \dots, N$, with the center of the cell denoted by $x_j = \frac{1}{2}(x_{j-\frac{1}{2}} + x_{j+\frac{1}{2}})$ and the size of each cell by $\Delta x_j = x_{j+\frac{1}{2}} - x_{j-\frac{1}{2}}$. We denote

by $u_{j+\frac{1}{2}}^+$ and $u_{j+\frac{1}{2}}^-$ the values of u at $x_{j+\frac{1}{2}}$ from the right cell, I_{j+1} , and from the left cell, I_j , respectively. We define the space \mathcal{V} :

$$\mathcal{V} = \{v : v \text{ is a polynomial of degree at most } k \text{ for } x \in I_j, j = 1, \dots, N\} \quad (4.2)$$

as the solution and test function space. We rewrite the equation (4.1) into a first order system

$$\begin{cases} u_t + f(u)_x - q_x & = 0 \\ q - u_x & = 0 \end{cases} \quad (4.3)$$

and apply the discontinuous Galerkin method to the system (4.3): find the $u, q \in \mathcal{V}$, such that for all test functions $v, z \in \mathcal{V}$, we have

$$\int_{I_j} u_t v dx - \int_{I_j} (f(u) - q) v_x dx + (\hat{f} - \hat{q})_{j+\frac{1}{2}} v_{j+\frac{1}{2}}^- - (\hat{f} - \hat{q})_{j-\frac{1}{2}} v_{j-\frac{1}{2}}^+ = 0 \quad (4.4)$$

$$\int_{I_j} q z dx + \int_{I_j} u z_x - \hat{u}_{j+\frac{1}{2}} z_{j+\frac{1}{2}}^- + \hat{u}_{j-\frac{1}{2}} z_{j-\frac{1}{2}}^+ = 0 \quad (4.5)$$

for all $1 \leq j \leq N$.

The "hat" terms are the boundary terms that emerge from integration by parts. Among them,

$$\hat{f} = \hat{f}(u^-, u^+) \quad (4.6)$$

is a monotone flux, i.e. Lipschitz continuous in both arguments, consistent ($\hat{f}(u, u) = f(u)$), and non-decreasing in the first argument and non-increasing in the second argument. We could, for example, use the simple Lax-Friedrichs flux

$$\hat{f} = \frac{1}{2}(f(u^-) + f(u^+) - \alpha(u^+ - u^-)), \quad \alpha = \max_u |f'(u)| \quad (4.7)$$

where the maximum is taken over a relevant range of u . The fluxes \hat{q}, \hat{u} can be chosen as

$$\hat{q} = q^+, \quad \hat{u} = u^-, \quad (4.8)$$

or

$$\hat{q} = q^-, \quad \hat{u} = u^+. \quad (4.9)$$

These "numerical fluxes" should be designed based on different guiding principles for different PDEs to ensure stability. For example, upwinding should be used as a guideline for odd derivatives which correspond to waves, and eventual symmetric treatment, such as an alternating choice of the fluxes for a quantity and its derivative as in (4.8) or (4.9), should be used for even derivatives. For a detailed description of the LDG method as well as its implementation and applications, we refer the readers to the lecture notes [5], the review paper [11], and papers in the two special issues on discontinuous Galerkin methods [12, 14]. In the next section we will consider the linear KdV, linear bi-harmonic, linear fifth order derivative, KdV, Kuramoto-Sivashinsky, and Kawahara equations. The corresponding LDG methods can be found in [30, 29, 25, 28].

Table 5.1: Accuracy test for the linear one-dimensional equation (5.1) with the exact solution $u(x, t) = \sin(x + t)$. Periodic boundary conditions. Uniform meshes with N cells at time $t = 1$.

	N	SDC		ARK	
		L^∞ error	order	L^∞ error	order
P^2	10	3.63E-03	–	5.91E-03	–
	20	5.72E-04	2.67	9.09E-04	2.70
	40	7.42E-05	2.95	1.15E-04	2.98
	80	1.02E-05	2.87	1.46E-05	2.98
	160	1.33E-06	2.94	1.85E-06	2.98
P^3	10	3.67E-04	–	2.91E-04	–
	20	4.34E-05	3.08	1.90E-05	4.01
	40	2.94E-06	3.88	1.17E-06	4.02
	80	1.92E-07	3.94	7.27E-08	4.01
	160	1.27E-08	3.92	4.55E-09	4.00

5 Numerical results

In this section, we perform numerical experiments of the LDG scheme coupled with different time integration methods (SDC, ARK and ETD) to linear and nonlinear equations containing second to fifth order spatial derivatives. We use quadruple precision and the direct linear solver in *LAPACK* to compute these problems on uniform spatial meshes.

Example 5.1. We show an accuracy test for the linear KdV equation (5.1) in 1D:

$$u_t + u_{xxx} = 0 \tag{5.1}$$

with the initial condition $u(x, 0) = \sin(x)$ and periodic boundary conditions. The exact solution is given by $u(x, t) = \sin(x + t)$. The time steps are taken as $\Delta t = \Delta x$. When the piecewise P^2 elements are used in the LDG method, the implicit $SDC_2^2(1)$ scheme and the implicit part of the third order ARK scheme [19] are used in the time integration. When the piecewise P^3 elements are used in the LDG method, we use the implicit $SDC_2^3(1)$ scheme and the implicit part of the fourth order ARK scheme. The numerical errors and orders of accuracy can be found in Table 5.1. Apparently the errors are comparable between these two different time discretizations.

Example 5.2. We show an accuracy test for the linear bi-harmonic equation (5.2) in 1D:

$$u_t + u_{xxxx} = 0 \tag{5.2}$$

with the initial condition $u(x, 0) = \sin(x)$ and periodic boundary conditions. The exact solution is given by $u(x, t) = e^{-t} \sin(x + t)$. The time steps are taken as $\Delta t = \Delta x$. When the piecewise P^2 elements are used in the LDG method, the implicit $SDC_2^2(1)$ scheme and the implicit part of the third order ARK scheme [19] are used in the time integration. When the

Table 5.2: Accuracy test for the linear one-dimensional equation (5.2) with the exact solution $u(x, t) = e^{-t} \sin(x)$. Periodic boundary conditions. Uniform meshes with N cells at time $t = 1$.

	N	SDC		ARK	
		L^∞ error	order	L^∞ error	order
P^2	10	3.85E-04	–	4.09E-04	–
	20	5.14E-05	2.90	5.45E-05	2.91
	40	6.48E-06	2.99	7.12E-06	2.94
	80	8.15E-07	2.99	9.13E-07	2.96
	160	1.02E-07	2.99	1.16E-07	2.98
P^3	10	2.89E-05	–	4.15E-05	–
	20	2.39E-06	3.59	2.60E-06	4.00
	40	1.57E-07	3.93	1.64E-07	3.98
	80	9.84E-09	4.00	1.03E-08	4.00
	160	6.14E-10	4.00	6.45E-10	4.00

Table 5.3: A comparison of the L -stable $SDC_{2,3}^{3,3}(1, 1)$, $SDC_2^3(1, 0.8)$ schemes and the non- L -stable $SDC_2^3(1)$ scheme coupled with the p_3 LDG method for the linear 1D equation (5.2) with different time steps. L^∞ errors. Uniform meshes with $N = 160$ cells at time $t = 10$.

Δt	$0.2 \Delta x$	Δx	$2 \Delta x$	$4 \Delta x$	$8 \Delta x$	$16 \Delta x$
$SDC_2^3(1)$	7.13E-14	5.08E-12	7.42E-11	9.97E-10	1.16E-08	1.12E-07
$SDC_{2,3}^{3,3}(1, 1)$	7.88E-14	8.29E-13	1.42E-11	2.30E-10	3.71E-09	6.06E-08
$SDC_2^3(1, 0.8)$	7.64E-14	1.12E-12	3.29E-11	6.05E-10	8.14E-09	7.92E-08

piecewise P^3 elements are used in the LDG method, we use the implicit $SDC_2^3(1)$ scheme and the implicit part of the fourth order ARK scheme. The numerical errors and orders of accuracy can be found in Table 5.2. Apparently the errors are again comparable between these two different time discretizations.

We also test the performance of the L -stable $SDC_{2,3}^{3,3}(1, 1)$ and $SDC_2^3(1, 0.8)$ schemes in comparison with the non- L -stable $SDC_2^3(1)$ scheme for various time step sizes, coupled with the P^3 LDG method in the uniform mesh with $N = 160$ cells, in Table 5.3. We observe the advantage of the L -stable scheme for larger time step sizes in providing smaller errors.

Example 5.3. We solve the linear equation (5.3) in 1D

$$u_t + u_{xxxxx} = 0 \tag{5.3}$$

with the initial condition $u(x, 0) = \sin(x)$ and periodic boundary conditions. The exact solution is given by $u(x, t) = \sin(x - t)$. The time steps are taken as $\Delta t = \Delta x$. When the piecewise P^2 elements are used in the LDG method, the implicit $SDC_2^2(1)$ scheme and the

Table 5.4: Accuracy test for the linear one-dimensional equation (5.3) with the exact solution $u(x, t) = \sin(x - t)$. Periodic boundary conditions. Uniform meshes with N cells at time $t = 1$.

	N	SDC		ARK	
		L^∞ error	order	L^∞ error	order
P^2	10	6.74E-03	–	4.77E-03	–
	20	8.92E-04	2.92	7.91E-04	2.59
	40	1.02E-04	3.13	1.05E-04	2.92
	80	1.19E-05	3.09	1.39E-05	2.91
	160	1.44E-06	3.05	1.81E-06	2.95
P^3	10	3.80E-04	–	2.56E-04	–
	20	4.37E-05	3.12	1.76E-05	3.86
	40	2.95E-06	3.89	1.13E-06	3.96
	80	1.92E-07	3.94	7.15E-08	3.98
	160	1.26E-08	3.93	4.51E-09	3.99

implicit part of the third order ARK scheme [19] are used in the time integration. When the piecewise P^3 elements are used in the LDG method, we use the implicit $SDC_2^3(1)$ scheme and the implicit part of the fourth order ARK scheme. The numerical errors and orders of accuracy can be found in Table 5.4. Again the errors are comparable between these two different time discretizations.

Example 5.4. We show an accuracy test for the KdV equation

$$u_t - 3(u^2)_x + u_{xxx} = 0 \quad (5.4)$$

with the initial condition $u(x, 0) = -2\text{sech}^2(x)$. The exact solution is given by

$$u(x, t) = -2\text{sech}^2(x - 4t). \quad (5.5)$$

The computational domain is given by $[-20, 22]$, and we use a periodic boundary condition which is justified since the exact solution is negligibly small at the boundary of our computational domain. The time steps are taken as $\Delta t = C \frac{\Delta x}{\max_x |6u(x, 0)|}$, where C is the CFL number of the convection term associated with the stability of DG discretization (see [11]). When the piecewise P^2 elements are used in the LDG method, the implicit $SDC_2^2(1)$ scheme, the third order ARK scheme [19], and the third order ETD3RK scheme [18] are used in the time integration. When the piecewise P^3 elements are used in the LDG method, we use the implicit $SDC_2^3(1)$ scheme, the fourth order ARK scheme, and the fourth order ETD4RK scheme. For the SDC and ARK methods, the first order derivative term is treated explicitly (F_N in (2.19)), and the third order derivative term is treated implicitly (F_S in (2.19)). For the ETD method, the first order derivative term is treated as $F(t, u)$ in (3.4), and the third order derivative term is treated as Lu in (3.4). The numerical errors and orders of accuracy can be found in Table 5.5. The errors are basically comparable among these three different time discretizations, with ETD showing slightly larger errors.

Table 5.5: Accuracy test for the KdV equation (5.4) with the exact solution $u(x, t) = -2\text{sech}^2(x - 4t)$. Periodic boundary conditions. Uniform meshes with N cells at time $t = 1$.

	N	SDC		ARK		ETD	
		L^∞ error	order	L^∞ error	order	L^∞ error	order
P^2	80	1.23E-02	–	1.26E-02	–	2.63E-02	–
	160	2.23E-03	2.46	2.19E-03	2.52	2.89E-03	3.18
	320	3.09E-04	2.85	3.00E-04	2.87	4.11E-04	2.81
	640	3.92E-05	2.98	3.76E-05	2.99	4.23E-05	3.28
P^3	80	1.87E-03	–	1.86E-03	–	6.14E-03	–
	160	1.76E-04	3.41	1.75E-04	3.41	6.03E-04	3.35
	320	1.16E-05	3.93	1.15E-05	3.93	4.20E-05	3.84
	640	7.47E-07	3.96	7.39E-07	3.96	1.94E-06	4.43

Example 5.5. We show an accuracy test for the Kuramoto-Sivashinsky equation

$$u_t + uu_x + u_{xx} + \sigma u_{xxx} + u_{xxxx} = 0 \quad (5.6)$$

with the exact solution given by

$$u(x, t) = c + 9 - 15 \left(\tanh(k(x - ct - x_0)) + \tanh^2(k(x - ct - x_0)) - \tanh^3(k(x - ct - x_0)) \right). \quad (5.7)$$

where we take $\sigma = 4$, $c = 6$, $k = \frac{1}{2}$ and $x_0 = -10$. The computational domain is given by $[-30, 30]$, and we use a periodic boundary condition which is justified since the exact solution is negligibly small at the boundary of our computational domain. The time steps are taken as $\Delta t = C \frac{\Delta x}{\max_x |u(x, 0)|}$, where C is again the CFL number of the convection term

associated with the stability of DG discretization. When the piecewise P^2 elements are used in the LDG method, the implicit $SDC_2^2(1)$ scheme, the third order ARK scheme [19], and the third order ETD3RK scheme [18] are used in the time integration. When the piecewise P^3 elements are used in the LDG method, we use the implicit $SDC_2^3(1)$ scheme, the fourth order ARK scheme, and the fourth order ETD4RK scheme. For the SDC and ARK methods, the first order derivative term is treated explicitly (F_N in (2.19)), and the second to fourth order derivative terms are treated implicitly (F_S in (2.19)). For the ETD method, the first order derivative term is treated as $F(t, u)$ in (3.4), and the second to fourth order derivative terms are treated as Lu in (3.4). Table 5.6 gives the errors of the numerical solution at $t = 1$. Again, the errors are basically comparable among these three different time discretizations, with ETD showing slightly larger errors.

Example 5.6. We show an accuracy test for the Kawahara equation

$$u_t + uu_x + u_{xxx} - u_{xxxx} = 0 \quad (5.8)$$

with the exact solution

$$u(x, t) = \frac{105}{169} \text{sech}^4 \left[\frac{1}{2\sqrt{13}} \left(x - \frac{36t}{169} \right) \right]. \quad (5.9)$$

Table 5.6: Accuracy test for the Kuramoto-Sivashinsky equation (5.6) with the exact solution (5.7). Uniform meshes with N cells at time $t = 1$.

	N	SDC		ARK		ETD	
		L^∞ error	order	L^∞ error	order	L^∞ error	order
P^2	80	8.39E-02	–	8.90E-02	–	1.27E-01	–
	160	1.29E-02	2.71	1.36E-02	2.71	2.44E-02	2.39
	320	1.68E-03	2.94	1.78E-03	2.94	3.98E-03	2.61
	640	2.12E-04	2.98	2.25E-04	2.99	8.14E-04	2.29
P^3	80	1.02E-02	–	1.01E-02	–	2.46E-02	–
	160	7.80E-04	3.71	7.77E-04	3.71	3.20E-03	2.66
	320	5.08E-05	3.94	5.06E-05	3.94	2.88E-04	3.47
	640	2.88E-06	4.14	3.18E-06	3.99	2.47E-05	3.54

The computational domain is given by $[-35, 35]$, and we use a periodic boundary condition which is justified since the exact solution is negligibly small at the boundary of our computational domain. The time steps are taken as $\Delta t = C \frac{\Delta x}{\max_x |u(x,0)|}$, where C is the CFL number of the convection term associated with the stability of DG discretization. When the piecewise P^2 elements are used in the LDG method, the implicit $SDC_2^2(1)$ scheme, the third order ARK scheme [19], and the third order ETD3RK scheme [18] are used in the time integration. When the piecewise P^3 elements are used in the LDG method, we use the implicit $SDC_3^3(1)$ scheme, the fourth order ARK scheme, and the fourth order ETD4RK scheme. For the SDC and ARK methods, the first order derivative term is treated explicitly (F_N in (2.19)), and the third and fifth order derivative terms are treated implicitly (F_S in (2.19)). For the ETD method, the first order derivative term is treated as $F(t, u)$ in (3.4), and the third and fifth order derivative terms are treated as Lu in (3.4). Table 5.7 gives the errors of the numerical solution at $t = 1$. Again, the errors are basically comparable among these three different time discretizations, with ETD showing slightly larger errors.

For the sake of comparison of efficiency, we document the CPU time of the fourth order $SDC_3^3(1)$, ARK and ETD4RK methods coupled with the P^3 LDG method for solving the equation (5.8) for the same uniform mesh with $N = 160$ cells, and the same suitably chosen Δt which yields an L^∞ error at the level of 10^{-6} for all three methods. The machine we use is an Intel(R) Pentium(R) 4, with CPU 3.20GHz, Linux system with cache size 2048kb, memory is 3111120k total, and the compiler is the Intel(R) Fortran Compiler. The results are collected in Table 5.8. Comparing the CPU times, we notice that the ARK method requires less time than the SDC and ETD methods. Notice that we have not included the start-up cost (the CPU cost to evaluate the contour integral in the ETD method, which takes about 5070 seconds, and the CPU cost for the initial LU decomposition in the SDC and ARK methods, which takes about 3.7 seconds). If the number of function evaluations is compared, the ETD method requires fewer function evaluations than the ARK method, which in turn requires fewer function evaluations than the SDC method. However, we do see that all three methods are efficient in solving this highly stiff LDG problem.

Table 5.7: Accuracy test for the Kawahara equation (5.8) with the exact solution (5.9). Periodic boundary conditions. Uniform meshes with N cells at time $t = 1$.

	N	SDC		ARK		ETD	
		L^∞ error	order	L^∞ error	order	L^∞ error	order
P^2	20	1.23E-02	–	1.46E-02	–	8.13E-03	–
	40	1.64E-03	2.90	1.63E-03	3.17	1.84E-03	2.14
	80	2.06E-04	2.99	2.03E-04	3.00	3.75E-04	2.30
	160	2.60E-05	2.99	2.56E-05	2.98	6.52E-05	2.52
P^3	20	2.02E-03	–	1.97E-03	–	4.68E-03	–
	40	1.86E-04	3.44	1.77E-04	3.47	9.90E-04	2.64
	80	1.25E-05	3.89	1.20E-05	3.89	1.29E-04	2.62
	160	7.70E-07	4.03	7.57E-07	3.99	1.02E-05	3.66

Table 5.8: CPU time for the Kawahara equation (5.8) with the exact solution (5.9). Uniform mesh with 160 cells at time $t = 50$.

Precision	SDC		ARK		ETD	
	CPU time	F calls	CPU time	F calls	CPU time	F calls
10^{-6}	211.0s	4340	164.4s	3720	543.7s	2480

6 Concluding remarks

We have explored three different time discretization techniques for solving the stiff ODEs resulting from a local discontinuous Galerkin (LDG) spatial discretization to PDEs containing higher order spatial derivatives. These are the semi-implicit spectral deferred correction (SDC) method, the additive Runge-Kutta (ARK) method and the exponential time differencing (ETD) method. Numerical experiments are performed to verify that all three methods are efficient in discretizing the LDG schemes in time. In particular, the SDC method has the advantage of easy implementation for arbitrary order of accuracy. In future work we will explore efficient linear and nonlinear solvers for discretizing LDG schemes for multi-dimensional problems with possible nonlinear higher order spatial derivative terms.

References

- [1] R. Alexander, *Diagonally implicit Runge-Kutta methods for stiff ODEs*, SIAM J. Num. Anal., 14 (1977), pp.1006-1021.
- [2] R. Alexander and J.J. Coyle, *Implicit Runge-Kutta methods and differential-algebraic systems*, SIAM J. Num. Anal., 27 (1990), pp.736-752.
- [3] A.L. Araújo, A. Murua and J.M. Sanz-Serna, *Symplectic methods based on decomposition*, SIAM J. Num. Anal., 34 (1997), pp.1926-1947.
- [4] F. Bassi and S. Rebay, *A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations*, J. Comput. Phys., 131 (1997), pp.267-279.
- [5] B. Cockburn, *Discontinuous Galerkin methods for methods for convection-dominated problems*, in *High-Order Methods for Computational Physics*, T.J. Barth and H. Deconinck, editors, Lecture Notes in Computational Science and Engineering, volume 9, Springer, 1999, pp.69-224.
- [6] B. Cockburn, S. Hou and C.-W. Shu, *The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: the multidimensional case*, Math. Comp., 54 (1990), pp.545-581.
- [7] B. Cockburn, S.-Y. Lin and C.-W. Shu, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: one dimensional systems*, J. Comput. Phys., 84 (1989), pp.90-113.
- [8] B. Cockburn and C.-W. Shu, *TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: general framework*, Math. Comp., 52 (1989), pp.411-435.
- [9] B. Cockburn and C.-W. Shu, *The Runge-Kutta discontinuous Galerkin method for conservation laws V: multidimensional systems*, J. Comput. Phys., 141 (1998), pp.199-224.

- [10] B. Cockburn and C.-W. Shu, *The local discontinuous Galerkin method for time-dependent convection-diffusion systems*, SIAM J. Numer. Anal., 35 (1998), pp.2440-2463.
- [11] B. Cockburn and C.-W. Shu, *Runge-Kutta discontinuous Galerkin methods for convection-dominated problems*, J. Sci. Comp., 16 (2001), pp.173-261.
- [12] B. Cockburn and C.-W. Shu, *Foreword for the special issue on discontinuous Galerkin method*, J. Sci. Comp., 22-23 (2005), pp.1-3.
- [13] S.M. Cox and P.C. Matthews, *Exponential time differencing for stiff systems*, J. Comput. Phys., 176 (2002), pp.430-455.
- [14] C. Dawson, *Foreword for the special issue on discontinuous Galerkin method*, Comput. Meth. Appl. Mech. Engin., 195 (2006), p.3183.
- [15] A. Dutt, L. Greengard and V. Rokhlin, *Spectral deferred correction methods for ordinary differential equations*, BIT, 40 (2000), pp.241-266.
- [16] S. Gottlieb, C.-W. Shu and E. Tadmor, *Strong stability-preserving high-order time discretization methods*, SIAM Review, 43 (2001), pp.89-112.
- [17] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems*, second edition, Springer-Verlag, Berlin, 1996.
- [18] A.-K. Kassam and L.N. Trefethen, *Fourth-order time stepping for stiff problem*, SIAM J. Sci. Comput., 26 (2005), pp.1214-1233.
- [19] C.A. Kennedy and M.H. Carpenter, *Additive Runge-Kutta schemes for convection-diffusion-reaction equations*, Appl. Num. Math., 44 (2003), pp.139-181.
- [20] A. Kværnø, S.P. Nørsett and B. Owren, *Runge-Kutta research in Trondheim*, Appl. Num. Math., 22 (1996), pp.263-277.
- [21] D. Levy, C.-W. Shu and J. Yan, *Local discontinuous Galerkin methods for nonlinear dispersive equations*, J. Comput. Phys., 196 (2004), pp.751-772.
- [22] M.L. Minion, *Semi-implicit spectral deferred correction methods for ordinary differential equations*, Commun. Math. Sci., 1 (2003), pp.471-500.
- [23] W.H. Reed and T.R. Hill, *Triangular mesh method for the neutron transport equation*, Technical report LA-UR-73-479, Los Alamos Scientific Laboratory, Los Alamos, NM, 1973.
- [24] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock capturing schemes*, J. Comput. Phys., 77 (1988), pp.439-471.
- [25] Y. Xu and C.-W. Shu, *Local discontinuous Galerkin methods for three classes of nonlinear wave equations*, J. Comput. Math., 22 (2004), pp.250-274.

- [26] Y. Xu and C.-W. Shu, *Local discontinuous Galerkin methods for nonlinear Schrödinger equations*, J. Comput. Phys., 205 (2005), pp.72-97.
- [27] Y. Xu and C.-W. Shu, *Local discontinuous Galerkin methods for two classes of two dimensional nonlinear wave equations*, Physica D, 208 (2005), pp.21-58.
- [28] Y. Xu and C.-W. Shu, *Local discontinuous Galerkin methods for the Kuramoto-Sivashinsky equations and the Ito-type coupled KdV equations*, Comput. Meth. Appl. Mech. Engin., 195 (2006), pp.3430-3447.
- [29] J. Yan and C.-W. Shu, *A local discontinuous Galerkin method for KdV type equations*, SIAM J. Numer. Anal., 40 (2002), pp.769-791.
- [30] J. Yan and C.-W. Shu, *Local discontinuous Galerkin methods for partial differential equations with higher order derivatives*, J. Sci. Comput., 17 (2002), pp.27-47.