Second-order characteristic methods for advection–diffusion equations and comparison to other schemes

Mohamed Al-Lawatia a,*, Robert C. Sharpley b & Hong Wang b

aDepartment of Mathematics and Statistics, Sultan Qaboos University, P.O. Box 36, Al-Khod Postal Code 123, Muscat, Oman
bDepartment of Mathematics, University of South Carolina, Columbia, South Carolina 29208, USA

(Received 3 February 1998; revised 27 August 1998; accepted 14 September 1998)

We develop two characteristic methods for the solution of the linear advection diffusion equations which use a second order Runge–Kutta approximation of the characteristics within the framework of the Eulerian–Lagrangian localized adjoint method. These methods naturally incorporate all three types of boundary conditions in their formulations, are fully mass conservative, and generate regularly structured systems which are symmetric and positive definite for most combinations of the boundary conditions. Extensive numerical experiments are presented which compare the performance of these two Runge–Kutta methods to many other well perceived and widely used methods which include many Galerkin methods and high resolution methods from fluid dynamics. © 1999 Elsevier Science Ltd. All rights reserved

Key words: Characteristic methods, Comparison of numerical methods, Eulerian–Lagrangian methods, Numerical solutions of advection–diffusion equations, Runge–Kutta methods.

1 INTRODUCTION

Advection–diffusion equations are an important class of partial differential equations that arise in many scientific fields including fluid mechanics, gas dynamics, and atmospheric modeling. These equations model physical phenomenon characterized by a moving front. In fluid dynamics, for example, the movement of a solute in ground water is described by such an equation. Since these equations normally have no closed form analytical solutions, it is very important to have accurate numerical approximations. When diffusion dominates the physical process, standard finite difference methods (FDM) and finite element methods (FEM) work well in solving these equations. However, when advection is the dominant process, these equations present many numerical difficulties. In fact, standard finite element and finite difference methods produce solutions which exhibit non-physical oscillations, excessive numerical diffusion which smears out sharp fronts, or a combination of both 22,36.

Many specialized schemes have been developed to overcome the difficulties mentioned. One class of these methods, usually referred to as the class of Eulerian methods, uses an Eulerian fixed grid and improved techniques, such as upstream weighting, to generate more accurate approximations. Most of the methods in this class are characterized by ease of formulation and implementation, however their solutions suffer from excessive time truncation errors. Moreover, they put a strong limitation on the Courant number and hence require very small time steps to generate stable solutions. Among these methods are the Petrov–Galerkin FEM methods 1,4,7,10,60 which are improvements over the standard Galerkin FEM that incorporate upwinding in...
the space of the test functions. Also included in the class of Eulerian methods are the streamline diffusion FEM methods (SDM)\cite{SDM1,SDM2,SDM3} and the continuous and discontinuous Galerkin methods (CGM, DGM)\cite{CGM1,CGM2,CGM3}. The SDM improve over the standard space-time Galerkin FEM by adding a multiple of the (linearized) hyperbolic operator of the problem considered to the standard test functions. Thus they add numerical diffusion only in the direction of the streamlines. The SDM formulations have a free parameter which determines the amount of diffusion applied and therefore has a great effect on the accuracy of these methods. In practice, this parameter should be large enough to avoid oscillations in the solution, but not too large to damp the solution. A clear choice of this parameter is not known, in general, and is heavily problem dependent. The CGM and DGM are well suited for purely hyperbolic equations and recently have been extended to solve the advection–diffusion equations. They are space-time explicit methods in which, starting with the initial time solution and Dirichlet data at the inflow boundary, one would successively iterate over the elements of a quasi-uniform triangulation of the space-time domain in an order consistent with the domain, solving a local system over each element. In addition to the methods mentioned above, the class of Eulerian methods includes the high resolution methods in fluid dynamics such as the Godunov methods, the total variation diminishing methods (TVD), and the essentially non-oscillatory methods (ENO)\cite{ENO1,ENO2,ENO3,ENO4,ENO5,ENO6}. These methods, as well as the CGM and DGM, are well suited for advection–diffusion equations with small diffusion coefficients and in general impose an extra stability restriction on the size of the time step taken based on the magnitude of this coefficient. Therefore, they are very sensitive to changes in the diffusion coefficient, which in practical problems is likely to have large values at certain points.

Another class of methods, usually known as characteristic methods, makes use of the hyperbolic nature of the governing equation. These methods use a combination of Eulerian fixed grids to treat the diffusive component, and Lagrangian coordinates by tracking particles along the characteristics to treat the advective component. Included in this class are the Eulerian–Lagrangian methods (ELM), the modified methods of characteristics (MOC), and the operator splitting methods\cite{OSSM1,OSSM2,OSSM3,OSSM4,OSSM5,OSSM6}. These methods have the desirable advantage of alleviating the restrictions on the Courant number, thus allowing for large time steps. Furthermore, the Lagrangian treatment in these methods greatly reduces the time truncation errors which are present in Eulerian methods. On the other hand these methods have difficulty in conserving mass and treating general boundary conditions. The Eulerian–Lagrangian localized adjoint methods (ELLAM) were developed as an improved extension of the characteristic methods that maintains their advantages but enhances their performance by conserving mass and treating general boundary conditions naturally in their formulations. The first ELLAM formulations were developed for constant coefficient advection–diffusion equations\cite{ELLAM1,ELLAM2}. The strong potential that these FEM based formulations and their numerical results have demonstrated have led to the development of additional formulations for variable-coefficient advection diffusion equations\cite{ELLAM3,ELLAM4} and for non-linear equations\cite{ELLAM5,ELLAM6} as well as finite volume formulations\cite{ELLAM7,ELLAM8}. However, because the characteristics for variable-coefficient advection–diffusion equations cannot be tracked exactly in general, many characteristic and ELLAM methods that have been developed use a backward Euler approximation for these characteristics due to its simplicity and stability. These (backward Euler) schemes are second order accurate in space but only first order accurate in time.

An ELLAM based formulation which uses a second order approximation for the characteristics was developed recently for first-order advection-reaction equations\cite{ELLAM9}. This formulation was shown to be of second order in both space and time. Unlike the first-order equations, similar treatment for the advection–diffusion equations is more problematic since one needs to treat the diffusive component and its partial derivatives (with respect to the rectangular and characteristic coordinates) carefully along the characteristics to produce systems having desirable structure. In addition, boundary treatment is more involved because at both the inflow and outflow boundary data can be specified in many forms which then need to be incorporated into the formulations. In this paper we develop characteristic methods for the one-dimensional linear advection–diffusion equations, based on a second order Runge–Kutta approximation of the characteristics. We present a backward tracking (BRKC) and a forward tracking (FRKC) Runge–Kutta characteristic scheme, both of which are mass conservative and incorporate boundary conditions naturally in their formulations. These methods, which can be thought of as generalizations of the ELLAM schemes, generate tridiagonal (regularly structured in multidimensions) matrices which are symmetric (except at the inflow boundary), positive definite, and therefore easily and efficiently solved. Moreover, we provide the results of fairly extensive numerical experiments which compare the performance of the two methods developed in this paper to many of the methods mentioned above.

This paper is organized as follows. In Section 2 we develop a reference equation based on exact characteristic tracking. In Section 3 we present the two characteristic schemes BRKC and FRKC which are based on a backward tracking and a forward tracking algorithm, respectively. We also give a detailed description of boundary treatment, in addition to numerical experiments which demonstrate the order of convergence of
the two schemes. In Section 4 we give a brief description of some well studied and widely used methods which are known to give good approximations to the advection diffusion equations. Section 5 contains the results of the numerical experiments that compare the performance of the two schemes developed in Section 3 and the other methods described in Section 4.

2 DEVELOPMENT OF THE METHOD

We consider the one-dimensional linear variable-coefficient advection–diffusion equation

\[ \dot{u} + (V(x, t)u - D(x, t)\partial_x u) = f(x, t), \]

\[ u(x, 0) = u_0(x), \quad x \in [a, b], \]

where \( V(x, t) \) and \( D(x, t) \) are the velocity field and the diffusion coefficient, respectively. \( D(x, t) \) is assumed to be positive throughout the domain. To simplify our presentation, we also assume \( V(a, t) \) and \( V(b, t) \) are positive, i.e., we set \( x = a \) and \( x = b \) to be the inflow and outflow boundaries, respectively. In many advection-dominated applications, \( |D(x, t)| < |V(x, t)| \), therefore, methods devised to solve eqn (1) should handle this case accurately. We consider general boundary conditions with any combination of Dirichlet, Neumann, or Robin conditions at the inflow and outflow boundaries. The function \( g(t) \) is used to denote the time dependence to specify the boundary condition at \( x = a \) and \( h(t) \) for the outflow boundary at \( x = b \). The Dirichlet, Neumann, and Robin conditions are formulated by the requirements

\[ u(c, t) = g(t), \quad t \in (0, T], \]

\[ -(Du_x)(c, t) = h(t), \quad t \in (0, T], \]

\[ (VJ - Du_x)(c, t) = \phi(c, t), \quad t \in (0, T], \]

respectively, where \( \phi = g \) when \( c = a \) and \( \phi = h \) when \( c = b \).

2.1 Variational formulation and characteristic curves

The domain of problem (1), is \([a, b] \times [0, T]\) which we partition in space and time as follows:

\[ 0 = t_0 < t_1 < \ldots < t_N = T, \]

\[ a = x_0 < x_1 < \ldots < x_N = b, \quad n = 0, 1, \ldots, N, \]

for positive integers \( N \), \( N_0 \) (\( n = 0, 1, \ldots, N \)). We utilize space-time test functions that vanish outside \([a, b] \times (t^n, t^{n+1}]\), which enables us to concentrate on one time period \((t^n, t^{n+1}]\). Our test functions are discontinuous in time and allow for different meshes at different time periods. For notational simplicity, we suppress the temporal index on the grid. The variational formulation of eqn (1) on the domain \( \Omega = [a, b] \times (t^n, t^{n+1}] \), obtained by multiplying that equation by a test function \( w \) (which we describe in more detail below) and integrating by parts, becomes

\[ \int_a^b u(x, t^{n+1})w(x, t^{n+1}) \, dx + \int_a^b Du_xw_t \, dx \, dt \]

\[ + \int_a^b (VJ - Du_x)w \bigg|_{t^n} \, dx \, dt - \int_a^b u(w_t + Vw_x) \, dx \, dt \]

\[ = \int_a^b u(x, t^n)w(x, t^n) \, dx + \int_a^b f w \, dx \, dt, \]  (4)

where we use the notation \( w(x, r^n) = \lim_{t \to t^n} w(x, t) \) due to the discontinuity of \( w(x, t) \) at time \( t^n \).

The principle of the localized adjoint methods (LAM) requires the test functions to be chosen from the solution space of the homogeneous adjoint equation

\[ \dot{w} - V(x, t)w_t + (D(x, t)w) = 0. \]  (5)

However, the solution space of this equation is infinite dimensional. Thus we need to split this equation to determine a finite number of test functions. In fact, different splittings lead to different approximation schemes (see §5). Due to the Lagrangian nature of the exact solution, a natural splitting is

\[ w_t + V(x, t)w_x = 0, \]

\[ (D(x, t)w) = 0. \]  (6)

The ELLAM generalizes the framework of the localized adjoint methods by requiring the test functions to satisfy the first equation in (6) exactly or even approximately (so that the last term on the left-hand side of eqn (4) vanishes exactly or approximately). However, the ELLAM does not require the second equation in (6) to be satisfied, thus one does not have to choose the test functions \( w(x, t) \) to be of a complicated form in space. In our formulation, we choose the test functions to be piecewise linear in space, as in the linear standard FEM, and to be constant along characteristics which we discuss below.

The characteristic curves of eqn (1) are given by the solutions of initial value problems for the ordinary differential equation

\[ dy \over dt = V(y, t). \]  (7)

We denote the characteristic curve emanating from a given point \((\bar{x}, \bar{t})\) with \( t \in [t^n, t^{n+1}] \), by

\[ y = X(\theta; \bar{x}, \bar{t}), \]  (8)

where \( \theta \) is the time position parameter along that characteristic. Furthermore, we introduce below some notation that is described by the following relations:

\[ \bar{x} = X(t^{n+1}; x, t^n), \]

\[ x* = X(t^n; x, t^{n+1}), \]

\[ b*(t) = X(t^n; b, t), \]

\[ a = X(t^n; x, t^{n+1}), \]

\[ b = X(t^n; x, t^n). \]  (9)
We define \( \tilde{x} \) and \( x' \) as the head (using forward tracking) and the foot (using backward tracking), respectively, of the characteristics. The foot of a characteristic with head on the outflow boundary is denoted by \( t^o(t) \) for \( t \in [t^o, t^{o+1}] \). We also define \( \tilde{t}(x) \) and \( t'(x) \) as the exit times of the characteristics \( X(\tilde{t}; x, t^o) \) and \( X(\tilde{t}; x, t^{o+1}) \) at the outflow and inflow boundaries, respectively (see Fig. 1(b)). The general time increment over the domain may then be written as

\[
\Delta t(x) := \tilde{t}(x) - t'(x),
\]

where we define \( t'(x) = t^o \) for characteristics with feet not on the inflow boundary, and similarly \( \tilde{t}(x) = t^{o+1} \) for characteristics with heads not belonging to the outflow boundary. By implicitly differentiating the fourth relation in (9) for \( t'(x) \) with respect to \( x \), and the relation \( X(\tilde{t}; b, t) = b + \int_{0}^{t} V(X(s; b, t), s) \, ds \), for characteristics \( X(\tilde{t}; b, t) \) with \( \tilde{t} \in [t^o, t] \) originating at points \((b, t)\) on the outflow boundary, with respect to \( t \), we get the following equations (for partial derivatives of the characteristics)

\[
X_t(t'(x); x, t^{o+1}) = -V(a, t'(x)) \frac{dt'(x)}{dx},
\]

\[
X_t(t(x); x, b) = -V(b, t),
\]

in respective order. In the following section, we consider approximations of the characteristics, defined by second order Runge–Kutta approximations.

2.2 Reference equation

After we have introduced the notion of the characteristics, we are able to find a reference equation by expanding certain of the integrals in the variational formulation (4). First we assume that we can exactly track the characteristics, thus the same reference equation is generated if we use either forward or backward tracking algorithms.

2.2.1 Treatment of the source term

We start with the source term (second term on the right-hand side of eqn (4)) which we break as follows,

\[
\int_{r^o}^{r^{o+1}} \int_{\Omega_1} f(x, y) \, dx \, dt = \int_{\Omega_2} \int_{\Omega_3} f(x, y) \, ds + \int_{\Omega_2} \int_{\Omega_3} f(x, y) \, ds + \int_{\Omega_2} \int_{\Omega_3} f(x, y) \, ds.
\]

Here the region \( \Omega_1 \) represents all points which lie on characteristics with feet on the inflow boundary, \( \Omega_2 \) represents all points on characteristics with heads on the outflow boundary, and finally \( \Omega_3 \) represents the remainder of \( \Omega \) (see Fig. 1(a)). For clarity of presentation, we use the variable \( y \) to represent the spatial coordinate of any point in \( \Omega \), and reserve \( x \) for points on the spatial mesh of \( \Omega \) at time \( t^o \) or \( t^{o+1} \), representing either heads or feet of characteristics. Similarly we let \( s \) denote the temporal variable instead of \( t \), which we reserve for the temporal coordinate of heads or feet of characteristics which lie on either the inflow or outflow boundary. The general time increment can also be described for points \( x \) at time \( t^{o+1} \) or time \( t^o \) in respective order by \( \Delta t^{(i)}(x) \) and \( \Delta t^{(o)}(x) \) which are defined as follows:

\[
\Delta t^{(i)}(x) = t^{o+1} - t'(x), \quad x \in [a, b],
\]

\[
\Delta t^{(o)}(x) = \tilde{t}(x) - t^o, \quad x \in [a, b],
\]

where \( t'(x) \) and \( \tilde{t}(x) \) extend to \( t^o \) and \( t^{o+1} \), respectively, if the characteristics that define them lie in \( \Omega_2 \). With the change of variable \( y = y(x, s) = X(s; x, t^{o+1}) \) for the first integral on the right-hand side of (11), we obtain

\[
\int_{\Omega_1} \int_{\Omega_2} f(y, s) w(y, s) \, dy \, ds
\]

\[
= \int_{\Omega_2} \int_{\Omega_3} f(w) X(s; x, t^{o+1}) \, ds \, dx
\]

\[
= \int_{\Omega_2} \int_{\Omega_3} \frac{\Delta t^{(i)}(x)}{2} \left( f(x, t^{o+1}) w(x, t^{o+1}) \right) \, dx + E_{f, \Omega_2}(w)
\]
In this expression the trapezoidal error term is given by

\[ \int_{a}^{b} \frac{\Delta t^{(2)}}{2} f(x, t_{n+1}) w(x, t_{n+1}) \, dx \]

\[ + \int_{a}^{b} \frac{(t_{n} - t)}{2} f(a, t)w(a, t)V(a, t) \, dt + E_{f,\Omega_1}(w), \]

where in the second equality we used a trapezoidal approximation for the inner integral whose error \( E_{f,\Omega_1}(w) \) is given below. In the last equality we moved the second integral to the inflow boundary by the change of variable \( a = X(t_x(x); x, t_{n+1}) \) and the first equation in (10). The trapezoidal error term introduced is given by

\[ E_{f,\Omega_1}(w) = \int_{a}^{b} \int_{r(s)}^{r_{n+1}} \left( f'(x) - s \right) \frac{(t_{n+1} - s)}{2} \, dx \, ds, \]

\[ \frac{d^2}{dx^2} [X_t(x; x, t_{n+1}) f(X(x; x, t_{n+1}), s)] w(x, t_{n+1}) \, dx, \]

since \( w \) is constant along characteristics. With the same change of variable, \( y = X(s; x, t_{n+1}) \), and similar treatment as in (13), we expand the second integral on the right-hand side of equation (11) to get

\[ \int_{a}^{b} \int_{r(s)}^{r_{n+1}} f(y, s)w(y, s) \, dy \, ds \]

\[ = \int_{a}^{b} \int f(x; s) X_t(x; x, t_{n+1}) \, dx \, ds \]

\[ = \int_{a}^{b} \frac{\Delta t}{2} \left( f(x, t_{n+1}) w(x, t_{n+1}) \right) \]

\[ + f(x^*, t) w(x^*, t^*) \frac{dx^*}{dt} \, dx + E_{f,\Omega_1}(w) \]

\[ = \int_{a}^{b} \frac{\Delta t}{2} f(x, t_{n+1}) w(x, t_{n+1}) \, dx \]

\[ + \int_{a}^{b} \frac{\Delta t}{2} f(x, t) w(x, t) \, dx + E_{f,\Omega_1}(w), \]

where we have used the fact that \( dx^*/dx = X_t(t^*; x, t_{n+1}) \). In this expression the trapezoidal error term is given by

\[ E_{f,\Omega_1}(w) = \int_{a}^{b} \int_{r(s)}^{r_{n+1}} \frac{(t_{n} - s)(t_{n+1} - s)}{2} \]

\[ \frac{d^2}{dx^2} [X_t(x; x, t_{n+1}) f(X(x; x, t_{n+1}), s)] w(x, t_{n+1}) \, dx, \]

The treatment of the source term over \( \Omega_1 \) is slightly different since for the characteristics, we backtrack from points originating on the outflow boundary. With the change of variable \( y = X(s; b, t) \), the third term on the right-hand side of (11) becomes

\[ \int_{b}^{t} \int f(y, s)w(y, s) \, dy \, ds \]

\[ = \int_{b}^{t} \int f(X(s; b, t), s) (X(s; b, t), s)_x X_x(s; b, t) \, ds \, dt \]

\[ = \int_{b}^{t} \left( \frac{t - t}{} \right) \left( f(b, t) w(b, t)X_t(t; b, t) \right) \]

\[ + f(X(t^*; b, t), t^*) X_t(t^*; b, t) \, dt + E_{f,\Omega_2}(w) \]

\[ = \int_{b}^{t} \frac{(t - t)}{2} V(b, t) f(b, t) w(b, t) \, dt \]

\[ + \int_{b}^{t} \frac{\Delta t^{(2)}}{2} f(x, t) w(x, t^*) \, dx + E_{f,\Omega_2}(w), \]

where in the last equality we used the second equation in (10) for the first integral and the change of variable \( x = X(t^*; b, t) \) for the second. The trapezoidal error term introduced is given by

\[ E_{f,\Omega_2}(w) = \int_{b}^{t} \frac{(t - t)}{2} \]

\[ \frac{d^2}{dx^2} [X_t(x; b, t) f(X(s; b, t), s)] w(b, t) \, ds \, dt. \]

2.2.2 Treatment of the diffusion term

Next we treat the second term on the left-hand side of eqn (4), which we break, similar to the source term in eqn (11), over the three subdomains \( \Omega_i (i = 1, 2, 3) \) discussed above. Due to the similarity in the treatment of this integral over \( \Omega_1 \) and \( \Omega_2 \), we combine the two and use the same change of variable \( y = X(s; x, t_{n+1}) \) as before, and obtain

\[ \int_{\Omega_{1,\Omega_2}} \int D(y, s)_x u_x(y, s) w_x(y, s) \, dy \, ds \]

\[ = \int_{a}^{b} \int X_t(s; x, t_{n+1}) (D_x u_x)(X(s; x, t_{n+1}), s) \]

\[ w_x(X(s; x, t_{n+1}), s) \, dx \]

\[ = \int_{a}^{b} \frac{\Delta t}{2} (D_x u_x)(x, t_{n+1}) w_x(x, t_{n+1}) \]

\[ + X_t(t^*; x, t_{n+1}) (D_x u_x)(X(t^*; x, t_{n+1}), t^*) \]
becomes

\[ w_x(X(t^*(x); x, t^{n+1}), t^*(x)) \, dx \]

\[ + \int_a^b \Delta f(x)(Du_x)(x, t^{n+1})w_s(x, t^{n+1}) \, dx + E_{D,\Omega_2}(w) \]

\[ = \int_a^b \Delta f(x)(Du_x)(x, t^{n+1})w_s(x, t^{n+1}) \, dx \]

\[ + \int_a^b \frac{\Delta f(x)}{2}(Du_x)(x, t^{n+1})w_s(x, t^{n+1}) \, dx \]

\[ + \int_a^b \frac{\Delta f(x)}{2}(Du_x)(x, t^r)w_s(x, t^r) \, dx + E_{D,\Omega_2}(w), \quad (19) \]

where in the second equality we used a backward Euler approximation for the integral over \((a, \bar{a})\) and a trapezoidal approximation over \((a, b)\) with error given below, and in the third equality we made the change \(w_s(X(s; x, t^{n+1}), s) = w_s(X(s; x, t^{n+1}), s) \times X(s; x, t^{n+1})\). The error term in (19) is given by

\[ E_{D,\Omega_2}(w) \]

\[ = - \int_a^b \int_0^{t^{n+1}} \frac{d}{d\theta} (Du_x)(X(\theta; x, t^{n+1}), \theta) \, d\theta \, ds \]

\[ \times w_s(x, t^{n+1}) \, dx + \int_a^b \int_{t^r}^{t^{n+1}} \frac{(t^s - s)(t^{n+1} - s)}{2} \]

\[ \times \frac{d^2}{ds^2} \left[ (Du_x)(X(s; x, t^{n+1}), s) \right] w_s(x, t^{n+1}) \, ds \, dx. \quad (20) \]

Using a trapezoidal approximation for this diffusion flux term over \((a, \bar{a})\) when the inflow boundary condition is Dirichlet or Robin condition introduces unknowns on the inflow boundary and severely complicates the scheme. Therefore we chose to use only backward Euler approximation on that interval. Neumann inflow condition on the other hand allows us to use the more accurate trapezoidal approximation for the diffusive integral uniformly over the whole spatial domain. This treatment does not lead to any non-symmetric terms in the formulation. Details of this treatment and boundary implementation are discussed in the following section.

Finally we treat remaining diffusion integral over \(\Omega_3\).

With the change of variable \(y = X(s; b, t)\) this integral becomes

\[ \int_{\Omega_3} \int \frac{D(y, s)}{w_y(y, s)} w_y(y, s) \, dy \, ds \]

\[ = - \int_a^b \int_{t^r}^{t^{n+1}} X(s; b, t)(Du_x)(X(s; b, t), t^r) \]

\[ w_y(X(s; b, t), s) \, ds \, dt \]

\[ = - \int_a^b \frac{(t^r - t^s)}{2} (Du_x)(b, t) w_t(b, t) \]

\[ + X_i(t^r; b, t)(Du_x)(X(t^r; b, t), t^r) \]

\[ \times w_s(X(t^r; b, t), t^r) \, dt + E_{D,\Omega_3}(w) \]

\[ = - \int_a^b \frac{(t^r - t^s)}{2} (Du_x)(b, t) w_t(b, t) \]

\[ + \int_a^b \frac{\Delta f(x)}{2} (Du_x)(x, t^r) w_t(x, t^r) \, dx + E_{D,\Omega_3}(w), \quad (21) \]

where in the second equality, we have used the relation \(w_s(b, t)X_i(t; b, t) = w_t(b, t)\) for the first integral. The error term \(E_{D,\Omega_3}\) is given by

\[ E_{D,\Omega_3}(w) = \int_a^b \int_{t^r}^{t^{n+1}} \frac{(t^r - t^s)(t^r - s)}{2} \]

\[ \times \frac{d^2}{dt^2} \left[ (Du_x)(X(s; b, t), s) \right] w_t(x, t^r) \, ds \, dt. \quad (22) \]

### 2.2.3 Assembly of the reference equation

Substituting the integrals expanded in eqns (13), (15), (17), (19) and (21) back into the variational equation (4) yields the following reference equation

\[ \int_a^b \int u(x, t^{n+1}) w(x, t^{n+1}) \, dx \]

\[ + \int_a^b \Delta f(x)(Du_x)(x, t^{n+1}) w_s(x, t^{n+1}) \, dx \]

\[ + \int_a^b \frac{\Delta f(x)}{2} (Du_x)(x, t^{n+1}) w_s(x, t^{n+1}) \, dx \]

\[ + \int_a^b \frac{\Delta f(x)}{2} (Du_x)(x, t^r) w_s(x, t^r) \, dx \]

\[ + \int_a^b \frac{(t^r - t^s)}{2} (Du_x)(b, t) w_t(b, t) \, dt \]

\[ + \int_a^b (Vu - Du_x)(b, t) w(b, t) \, dt \]

\[ - \int_a^b (Vu - Du_x)(a, t) w(a, t) \, dt \]

\[ = \int_a^b u(x, t^r) w(x, t^r) \, dx \]

\[ + \int_a^b \frac{\Delta f(x)}{2} f(x, t^{n+1}) w(x, t^{n+1}) \, dx \]
The last two terms in eqn (23) are
\[ E(w) + R_{w}(w). \]
(23)

The last two terms in eqn (23) are \( E(w) = \sum_{i=1}^{3}(E_{f_{i}}A_{i} + E_{D_{i}}B_{i}) \) which represents the collective approximation error and \( R_{w}(w) = \int_{t_{i}}^{t_{i+1}} \int_{x_{i}}^{x_{i+1}} u(w_{i} + \nu w_{i})dxdr \) represents the adjoint term which in our case vanishes by eqn (6).

3 NUMERICAL APPROXIMATION

Since we do not impose a particular form for the velocity field \( V(x, t) \), explicitly solving the ordinary differential equation (7) which defines the characteristics is not possible in general and introduces additional difficulty. Therefore in our numerical approximation of the solution of the advection diffusion equation (1), we consider an approximation of the characteristics \( X(\theta; x, t) \) which is based on a second order Runge–Kutta approximation known as Heun’s method. In particular, we define the approximate characteristic curve emanating from a point \((\tilde{x}, \tilde{t})\), with \( \tilde{t} \in [\tilde{t}^0, \tilde{t}^1]\), by
\[
Y(\theta; x, t) = \tilde{x} - \frac{(t - \theta)}{2} \left[V(\tilde{x}, \tilde{t}) + V(\tilde{x} - (t - \theta)V(\tilde{x}, \tilde{t}), \theta)\right].
\]
(24)

Furthermore, if the Courant number \( Cr = \max(V(\Delta t/\Delta x) \) is at least 2, we partition the outflow boundary \( \{x = b, t \in [\tilde{t}^0, \tilde{t}^1]\} \) as follows:
\[
t_{i} = \begin{cases} 
\tilde{t}^1 - (i - 1)\Delta x/ \max(V), & i = I, \ldots, I + IC - 1, \\
\tilde{t}^0, & i = I + IC;
\end{cases}
\]
(25)

where \( IC \) in this case is the integer part of \( Cr \). When no subdivision of the outflow boundary is introduced (\( Cr < 2 \)), we let \( IC = 1 \) so that both cases are treated uniformly. This partition introduces some unknowns on the outflow boundary, however it has the advantage of insuring that the schemes we develop are suitable even for large Courant numbers. We define the test functions at time \( \tilde{t}^1 \) as hat functions given by
\[
w_{i}(x, \tilde{t}^1) = \begin{cases} 
\frac{\Delta x}{\Delta x_{i}}, & x \in [x_{i-1}, x_{i}], \\
\frac{\Delta x_{i+1}-x}{\Delta x_{i+1}}, & x \in [x_{i}, x_{i+1}], \\
0, & \text{otherwise},
\end{cases}
\]
(26)

for \( i = 0, \ldots, I - 1 \). At the outflow boundary the test functions are given by
\[
w_{i}(b, t) = \begin{cases} 
\frac{b-t_{i-1}}{\Delta t_{i-1}}, & t \in [t_{i-1}, t_{i}], \\
\frac{t_{i+1} - t}{\Delta t_{i+1}}, & t \in [t_{i}, t_{i+1}], \\
0, & \text{otherwise},
\end{cases}
\]
(27)

for \( i = I + 1, \ldots, I + IC \). Here \( \Delta x_{i} = x_{i} - x_{i-1} \) and \( \Delta t_{i} = t_{i+1} - t_{i} \). For the interior of the domain \( \Omega \), we extend these test functions defined by (26) and (27) to be constant along the approximate characteristics. The test function \( w_{i} \) is a combination of both, that is, \( w_{i}(x, \tilde{t}^1) \) is defined by (26) for the interval \([x_{i-1}, x_{i}]\), and \( w_{i}(b, t) \) is defined by (27) on the interval \([t_{i+1}, t_{i+1}^{+}]\).

The numerical schemes are based on approximating the exact solution \( u \), which satisfies the reference equation (23), by a piecewise linear function \( U \) which satisfies a similar equation with two differences: (i) the new equation will be developed using the change of variables resulting from the approximate characteristic tracking given by (24), and (ii) the error and adjoint terms, similar to \( E(w) \) and \( R_{w} \), in (23), are not included. Since the terms neglected contribute global errors which are of order \( O((\Delta x)^2 + (\Delta t)^2) \), the resulting schemes will be of desired accuracy. In the following two subsections we develop two schemes to solve the advection–diffusion equation (1) based on backward and forward characteristic tracking, respectively. Here we emphasize that the test functions defined above are constant along the approximate characteristics.

3.1 Backtracking Runge–Kutta characteristic scheme

The first numerical scheme (BRKC) is based on a backward tracking of characteristics. We first let the domains \( \Omega_{i} \) be defined in a similar manner as before, but use the approximate characteristics \( Y(\theta; x, \tilde{t}^1) \) in place of the true characteristics (see Fig. 1(a)). In this case, our characteristics originate at points \((x, \tilde{t}^1)\) and are given by \( Y(\theta; x, \tilde{t}^1) \) in \( \Omega_{i} \) and \( \Omega_{i+1} \). In \( \Omega_{i} \), the characteristics originate at points \((b, t)\) on the outflow boundary and are given by \( Y(\theta; b, t) \). The notation introduced in Section 2 for exact tracking is also used in our numerical scheme formulation, however we modify it for this approximate characteristic tracking. Accordingly, we define \( \tilde{x} \) satisfying \( x = Y(\tilde{t}^1; \tilde{x}, \tilde{t}^1) \) and \( \tilde{x} = Y(\tilde{t}^1; \tilde{x}, \tilde{t}^1) \) as the head and the foot respectively, of the Runge–Kutta approximate characteristics. The foot of a characteristic with head on the outflow boundary is denoted by \( b'(t) = Y(\tilde{t}^1; b, t) \) for \( t \in [\tilde{t}^0, \tilde{t}^1] \). We also define \( \tilde{t}'(x) \) (given by the relation \( a = Y(\tilde{t}'(x); x, \tilde{t}^1) \) as the exit time of the characteristic \( Y(\tilde{t}'(x); x, \tilde{t}^1) \) at the inflow boundary. On the other hand \( \tilde{t}(x) \) (given by \( x = Y(\tilde{t}(x); b, \tilde{t})) \) is such that the point \((b, \tilde{t}(x))\) on the outflow boundary backtracks to the point \((x, \tilde{t}'(x))\). The two time increments \( \Delta t' \) and \( \Delta t'' \) are as defined before in eqn (12).

The reference equation using this approximate characteristic tracking is derived in a similar manner as was
done for eqn (23). Since, however, the solution of \( \alpha \) at time level \( t^n \) is sensitive to errors arising from the evaluation of the terms of eqn (23) which involve integrals of the trial function \( U(\cdot, \tau) \) at the previous time level, these terms must be treated with care. That these integrals are difficult to evaluate (especially in higher dimensions) is due to the fact that the test functions are defined by extending back along the approximate characteristics to determine the trial function values at \( \tau \); \( \hat{\alpha} \). In this way, the integrals are rewritten as integrals at \( \tau \), since several grid points from the previous time level may occur as backtracked points within a given interval at the current time. To illustrate how to overcome this difficulty, we will focus on the first term on the right-hand side of eqn (28) applied with test function \( \omega_i \). In this case, when \( x \) varies over the interval \([x_{i-1}, x_i]\) (i.e., the left half of the support of \( \omega_i(\cdot, t^n) \)), \( x \) will vary over the interval \([x_i - IC, x_i + IC]\). Therefore, due to the distortion by the characteristic tracking, \( U(x^n, \tau^n) \) is not a piecewise linear function, but rather a piecewise smooth function of \( x \in [x_i - 1, x_i] \). This possible loss of smoothness, however, affects the accuracy of high order quadrature methods. Therefore, we simply subdivide the interval so that \( U(x^n, \tau^n) \) is smooth on each subinterval and apply numerical integration to each subinterval separately. To illustrate the idea, we suppose \( x_{i-1} \) is in \([x_{i-1}, x_i] \) and \( x_i \) belongs to \([x_{i+2}, x_{i+2}] \), then we would split the integral into three parts along the intervals \([x_{i-1}, \bar{x}], [\bar{x}, \bar{x}_{i+1}], \) and \([\bar{x}_{i+1}, x_i] \). Newton iteration is applied to determine the points at the current time level that track back to the grid points at the previous time level. (Recall that approximate Runge–Kutta forward tracking and back tracking may not be inverse operations for variable velocity fields). Once these intervals are determined, we may numerically integrate by determining quadrature points in each subinterval and then backtracking them to perform the quadrature with values of the trial function. The amount of piecewise smoothness will be determined by the corresponding smoothness of the velocity field. Additional information

\[
\begin{align*}
&\int_{a}^{b} U(x, t^{n+1}) w(x, t^{n+1}) \, dx \\
&\quad + \int_{a}^{b} \Delta t^{(i)}(x)(DU_{i})(x, t^{n+1}) \omega_{i}(x, t^{n+1}) \, dx \\
&\quad + \int_{a}^{b} \frac{\Delta t}{2} ((DU_{i})(x, t^{n+1}) + (DU_{i})(x, \tau^{n})) \omega_{i}(x, t^{n+1}) \, dx \\
&\quad + \int_{\tau^{n+1}}^{b} (VU - DU_{i})(b, t) \omega(b, t) \, dt \\
&\quad - \int_{\tau^{n+1}}^{b} (VU - DU_{i})(a, t) \omega(a, t) \, dt \\
&\quad - \int_{\tau^{n+1}}^{b} \frac{(t - \tau^{n})}{2} ((DU_{i})(b, t) \\
&\quad + (DU_{i})(b^{*}(t), \tau^{n})) \omega_{i}(b, t) \, dt \\
&\quad + \int_{\tau^{n+1}}^{b} Y_{i}(\tau^{*}; x, \tau^{n+1}) U(x^{*}, \tau^{n}) w(x^{*}, t^{n+1}) \, dx \\
&\quad - \int_{\tau^{n+1}}^{b} Y_{i}(\tau^{*}; b, t) U(b^{*}(t), \tau^{n}) w(b, t) \, dt \\
&\quad + \int_{a}^{b} \frac{\Delta t^{(i)}(x)}{2} (f(x, t^{n+1}) \\
&\quad + Y_{i}(\tau^{*}; x, \tau^{n+1}) f(x^{*}, t^{*}(x)) w(x, t^{n+1}) \, dx \\
&\quad + \int_{\tau^{n+1}}^{b} \frac{(t - \tau^{n})}{2} (V(b, t) f(b, t) \\
&\quad - Y_{i}(\tau^{n}; b, t) f(b^{*}(t), t^{n+1}) w(b, t) \, dt,}
\end{align*}
\]
on this general problem and how to treat multivariate problems may be found in^{29,40,57}.

Another difficulty in incorporating the known values of the trial function in eqn (28) arises from evaluating the expression $U(x^*, t^*)$ when $x^*$ happens to be one of the nodal points $\{x_i\}_{i=0}^I$ at which $U$ may have a jump discontinuity. Since our characteristic tracking is only approximate, we assume that all points within a small tolerance of $x_i$ belong to the interval $(x_i, x_{i+1})$. This introduces an error which is controlled to within the desired order provided the tolerance is of the same order.

The scheme for nodes near the inflow boundary (i.e., $x_i (i = 0, \ldots, I + 1)$) differs since the characteristics traced back strike that boundary and lead to boundary terms appearing in the scheme. In the case of Robin flux boundary condition, we can easily change the fifth integral on the left-hand side of eqn (28) as follows

$$\int_{\tilde{a}}^{\tilde{b}} (VU - DU_x)(a, t)w(a, t) \, dt = \int_{\tilde{a}}^{\tilde{b}} g_3(t)w(a, t) \, dt. \quad (29)$$

Dirichlet inflow boundary condition introduces extra difficulty, since then the unknown diffusive term in the fifth integral on the left-hand side of eqn (28) appears in the scheme equations for nodes $x_i < \tilde{a}$. An alternative treatment for the diffusion term discussed in^{29,57} avoids this difficulty. Instead of integrating the term $\int_{\tilde{a}}^{\tilde{b}} -(Du_x)w \, dx \, dt$ by parts as we did in the derivation of the variational formulation (4) and then applying the integral approximation (trapezoidal, Euler), we can reverse the order, whence we get the term

$$\int_{\tilde{a}}^{\tilde{b}} \frac{d\Delta^{f(1)}(x)}{dx}(DU_x)(x, t^*)w(x, t^*) \, dx \quad (30)$$

instead of the diffusive flux in the fifth term on the left-hand side of eqn (28). The fifth term on the left-hand side of eqn (28) simplifies to

$$\int_{\tilde{a}}^{\tilde{b}} V(a, t)g_1(t)w(a, t) \, dt. \quad (31)$$

Since the value of the trial function $U$ is known at node $x_0$ from the prescribed Dirichlet boundary condition, equations need not be formulated there. Hence our scheme will be stipulated only for nodes $x_i (i = 1, \ldots, I)$. However, we do modify $w_1 = w_0 + w_1$ so that the test functions sum to one in order to maintain mass conservation.

Neumann inflow boundary condition generates a more natural scheme in the sense that a trapezoidal approximation, instead of the backward Euler approximation described above, can be implemented for the diffusive integral over $\Omega$. The advantage of this treatment is that it symmetrizes the inflow terms thus maintaining the symmetry of the whole formulation. Implementing this boundary condition leads to the following two changes to eqn (28): (i) a factor of $1/2$ multiplies the second integral on the left-hand side, and (ii) the additional term of

$$\int_{a}^{\tilde{b}} \frac{\Delta^{f(1)}(x)}{2} g_2(t(x))w_0(x, t^*) \, dx \quad (32)$$

is added on the left-hand side. The fifth term on the left-hand side of eqn (28) is replaced by

$$\int_{\tilde{a}}^{\tilde{b}} (VU)(a, t)w(a, t) \, dt + \int_{\tilde{a}}^{\tilde{b}} g_2(t)w(a, t) \, dt. \quad (33)$$

Wang, Ewing, and Russell^{57} describe in detail several possible ways to treat the first integral in eqn (33). Their findings suggest combining this term with the first term on the left-hand side of the reference equation (28) and replacing these with the term

$$\int_{\tilde{a}}^{\tilde{b}} U(x, t^*)w(x, t^*) \, dx. \quad (34)$$

Although this term is not exact for a variable velocity field $V$, the error introduced is within the desired order.

Due to the fact that we generate unknowns at the outflow boundary at nodes $t_i$, boundary treatment here becomes very important. First we note that $U(b, t^*)$ is known from the previous time step solution, hence, we do not impose an equation at $t_{I + IC} = t^*$. Therefore, as we did earlier in the case of Dirichlet inflow conditions, we redefine the test function $w_{I+IC-1} := w_{I+IC-1} + w_{I+IC}$ to maintain mass balance. For test functions $w_i, (i = I + 1, \ldots, I + IC - 1)$, the terms on the inflow boundary and terms evaluated at time $t^*$ of eqn (28) vanish. Thus for nodes $t_i (i = I + 1, \ldots, I + IC - 1)$, the reference eqn (28) simplifies to the following:

$$\int_{\tilde{a}}^{\tilde{b}} (VU - DU_x)(b, t)w_i(b, t) \, dt$$

$$= \int_{\tilde{a}}^{\tilde{b}} \frac{(t - t^*)}{2} (DU_x)(b, t) \, dt + (DU_x)(b^*(t), t^*))w_0(b, t) \, dt$$

$$- \int_{\tilde{a}}^{\tilde{b}} Y_t(t^*; b, t)U(b^*(t), t^*))w_i(b, t) \, dt$$

$$+ \int_{\tilde{a}}^{\tilde{b}} \frac{(t - t^*)}{2} (V(b, t))f(b, t)$$

$$- Y_t(t^*; b, t)(f(b^*(t), t^*)))w_0(b, t) \, dt. \quad (35)$$

We describe below how this equation changes for all three types of outflow boundary conditions. The
equation for node \( x_i \), has in addition to the terms in eqn (35) some other terms that are in eqn (28) which do not vanish for this node. Neumann outflow boundary condition can be incorporated in eqn (35), simply by making the change \(- (DU_i)(b, t) = h_i(t)\) in the first and second integrals on the left-hand side of this equation. Similarly the flux outflow condition can be incorporated in eqn (35) by applying the condition to the first integral on the left hand side, and using the relation \((DU_i) \times (b, t) = h_i(t) - (VU)(b, t)\) for the second integral on the left hand side of eqn (35). In the case of outflow Dirichlet condition, the solution \( U(b, t^{n+1}) \) is known from the boundary condition, therefore the equations for \( i \leq I - 1 \) decouple and can be solved for \( U(x_i, t^{n+1}), (i = 0 \ldots I - 1) \). Unless we desire the values of the derivative of the solution at the outflow boundary, we do not generate equations there.

3.2 Forward tracking scheme

In this brief section we indicate the development of a scheme (FRKC) based on a ‘forward tracking’ which is a feasible alternative for the backward tracking scheme of the last subsection especially for multidimensional problems. With the test functions as defined earlier by eqns (26) and (27), we use the reference equation (23) to derive the numerical scheme for Runge–Kutta forward tracking. Here again we wish to avoid the expensive task of backtracking the geometry to perform the integration of the test functions at the previous time level. Instead we perform this integration by using the fixed spatial grid of the trial function at \( t = t^n \) and apply numerical quadrature. The values of the test function required by the quadrature are obtained by forward tracking the quadrature points and evaluating \( w_i(x, t^{n+1}) \) or its derivatives at time \( t^{n+1} \). This forward scheme is only used to evaluate certain terms to incorporate known values and therefore does not lead to distorted grids that comprise a major drawback of explicit numerical schemes.

In this scheme, the characteristics originate at points \((x, t^i)\) and are given by \( Y(b; x, t^n) \) in \( \Omega_2 \) and \( \Omega_3 \) or they originate at points \((a, t)\) and are given by \( Y(b; a, t) \) in \( \Omega_1 \). Here again we use the same notation established in Sections 2 and 3, except we redefine them for this particular forward tracked approximate characteristics. Accordingly, we define \( X = Y(t^{n+1}; x, t^n) \) and \( x^+ \) satisfying \( x = Y(t^{n+1}; x^+, t^n) \) as the head and foot of the approximate characteristic. Moreover, we let \( t^i(x) \) (given by \( x = Y(t^{n+1}; a, t^i(x)) \)) and \( \tilde{I}(x) \) (given by \( b = Y(\tilde{I}(x); x, t^n) \)) denote the exit time of the approximate characteristics at the inflow and outflow boundaries, respectively. The time increments \( \Delta t^i \) and \( \Delta t^{i+1} \) are then given by eqn (12), where \( t^i(x) \) and \( \tilde{I}(x) \) extend to \( t^n \) and \( t^{n+1} \), respectively, if they are defined by characteristics in \( \Omega_2 \). The reference equation for the forward scheme, derived in a similar manner as eqn (23), is given by

\[
\begin{align*}
\int_{a}^{b} U(x, t^{n+1}) w(x, t^{n+1}) \, dx \\
+ \int_{a}^{b} \frac{\Delta t^{i}(x)}{2} (DU_i)(x, t^{n+1}) w_i(x, t^{n+1}) \, dx \\
+ \frac{\Delta t^{i}(x)}{2} (DU_i)(x, t^{n+1}) w_i(x, t^{n+1}) \, dx \\
- \frac{(t - t^n)}{2} (DU_i)(b, t) w_i(b, t) \, dt \\
+ \int_{a}^{b} (VU - DU_i)(a, t) w(a, t) \, dt \\
= \int_{a}^{b} U(x, t^i) w(x, t^i) \, dx + \frac{(t^{n+1} - t^n)}{2} (VFW)(a, t) \, dt \\
- \frac{(t - t^n)}{2} Y_i(t; b^i(t), t^n) f(b, t) w(b, t) \, dt \\
+ \frac{\Delta t^{i}(x)}{2} f(x, t^{n+1}) w(x, t^{n+1}) \, dx \\
+ \frac{\Delta t^{i+1}(x)}{2} f(x, t^n) w(x, t^n) \, dx.
\end{align*}
\]

The boundary treatment is similar to that of the backtracking scheme BRKC.

3.3 Experimental order of convergence

In this subsection we establish numerically the order of convergence of the two schemes developed for eqn (1) as well as the backward Euler ELLAM\(^\text{1,7}\), referred to as BE-ELLAM. The test problem involves the transport of a Gaussian distribution (described in Section 5.1) initially centered at zero. The spatial domain is \([0, 1]\) with a temporal domain \([0, 1]\). In this test we choose the small diffusion coefficient of \( D = 10^{-4} \), and consider the two velocity fields: \( V = 1 + 0.1x \) and the more rapidly changing \( V = 1 - 0.5x \). The right-hand side in eqn (1) is generated from this data and the analytical solution. The \( L_2 \) and \( L_1 \) norms of the residual error of the solution \( U \) of all three schemes are given by
we fix constants. To obtain the order of convergence in space Table 2. Order of convergence in time with $C$.

\[
\max_{x=0...N} \| U(x, T) - u(x, T) \|_{L^p_{[a,b]}} \leq C_x (\Delta x)^p + C_p (\Delta t)^\beta, \quad p = 1, 2
\]

where $\alpha$, $\beta$ give the order of convergence of the error in space and time respectively, and $C_x$, $C_p$ are positive constants. To obtain the order of convergence in space we fix $\Delta t = 1/500$ to insure time truncation errors are appropriately small. We then perform runs varying $\Delta x$ and apply a linear regression on the $L_\infty$ ($p = 1, 2$) norms of the error to determine the parameter $\alpha$. Tables 1 and 3 present the results of these runs and the computed value of the parameter $\alpha$ for the two velocity fields described above.

In a similar manner we fix $\Delta t = 1/700$ and perform runs varying $\Delta x$ to estimate the value of the $\alpha$.

Table 1. Order of convergence in space with $V(x, t) = 1 + 0.1x$

<table>
<thead>
<tr>
<th>Method</th>
<th>$\Delta t$</th>
<th>$\Delta x$</th>
<th>$L_2$ Error</th>
<th>$L_1$ Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRKC</td>
<td>1/500</td>
<td>1/40</td>
<td>$1.708301 \times 10^{-2}$</td>
<td>$1.008468 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/30</td>
<td>$9.762816 \times 10^{-3}$</td>
<td>$5.012535 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/20</td>
<td>$6.028822 \times 10^{-3}$</td>
<td>$2.822963 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/10</td>
<td>$3.989709 \times 10^{-3}$</td>
<td>$1.738382 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/80</td>
<td>$2.793234 \times 10^{-3}$</td>
<td>$1.191840 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$x = 2.6181$</td>
<td>$x = 3.0979$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$C_x = 270.4686$</td>
<td>$C_x = 919.3184$</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/40</td>
<td>$3.280644 \times 10^{-2}$</td>
<td>$1.560330 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/50</td>
<td>$2.993149 \times 10^{-2}$</td>
<td>$1.335517 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/60</td>
<td>$2.290385 \times 10^{-2}$</td>
<td>$1.005026 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/70</td>
<td>$1.415099 \times 10^{-2}$</td>
<td>$6.052967 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/80</td>
<td>$5.040757 \times 10^{-3}$</td>
<td>$2.071540 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$x = 2.4812$</td>
<td>$x = 2.6757$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$C_x = 418.3782$</td>
<td>$C_x = 405.2484$</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/40</td>
<td>$1.727842 \times 10^{-2}$</td>
<td>$1.019763 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/50</td>
<td>$1.004383 \times 10^{-2}$</td>
<td>$5.125955 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/60</td>
<td>$6.370859 \times 10^{-3}$</td>
<td>$2.974945 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/70</td>
<td>$4.375513 \times 10^{-3}$</td>
<td>$1.910709 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/80</td>
<td>$3.212402 \times 10^{-3}$</td>
<td>$1.356822 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$x = 2.4374$</td>
<td>$x = 2.9236$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$C_x = 138.478$</td>
<td>$C_x = 481.5463$</td>
</tr>
</tbody>
</table>

Table 2. Order of convergence in time with $V(x, t) = 1 + 0.1x$

<table>
<thead>
<tr>
<th>Method</th>
<th>$\Delta t$</th>
<th>$\Delta x$</th>
<th>$L_2$ Error</th>
<th>$L_1$ Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRKC</td>
<td>1/6</td>
<td>1/700</td>
<td>$5.653338 \times 10^{-3}$</td>
<td>$2.079667 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td>1/8</td>
<td>1/700</td>
<td>$3.189920 \times 10^{-3}$</td>
<td>$1.225680 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td>1/10</td>
<td>1/700</td>
<td>$2.066911 \times 10^{-3}$</td>
<td>$8.268313 \times 10^{-4}$</td>
</tr>
<tr>
<td></td>
<td>1/12</td>
<td>1/700</td>
<td>$1.460440 \times 10^{-3}$</td>
<td>$5.999894 \times 10^{-4}$</td>
</tr>
<tr>
<td></td>
<td>1/14</td>
<td>1/700</td>
<td>$1.094716 \times 10^{-3}$</td>
<td>$4.562014 \times 10^{-4}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$\beta = 1.9385$</td>
<td>$\beta = 1.7867$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$C_\beta = 0.1808$</td>
<td>$C_\beta = 0.0508$</td>
</tr>
<tr>
<td></td>
<td>1/6</td>
<td>1/700</td>
<td>$5.920335 \times 10^{-3}$</td>
<td>$2.503178 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td>1/8</td>
<td>1/700</td>
<td>$3.310857 \times 10^{-3}$</td>
<td>$1.400726 \times 10^{-3}$</td>
</tr>
<tr>
<td></td>
<td>1/10</td>
<td>1/700</td>
<td>$2.117405 \times 10^{-3}$</td>
<td>$8.985797 \times 10^{-4}$</td>
</tr>
<tr>
<td></td>
<td>1/12</td>
<td>1/700</td>
<td>$1.467171 \times 10^{-3}$</td>
<td>$6.138583 \times 10^{-4}$</td>
</tr>
<tr>
<td></td>
<td>1/14</td>
<td>1/700</td>
<td>$1.075166 \times 10^{-3}$</td>
<td>$4.520550 \times 10^{-4}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$\beta = 2.0123$</td>
<td>$\beta = 2.0222$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$C_\beta = 0.2177$</td>
<td>$C_\beta = 0.0939$</td>
</tr>
<tr>
<td></td>
<td>1/6</td>
<td>1/700</td>
<td>$6.476222 \times 10^{-2}$</td>
<td>$2.883357 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>1/8</td>
<td>1/700</td>
<td>$4.817777 \times 10^{-2}$</td>
<td>$2.151369 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>1/10</td>
<td>1/700</td>
<td>$3.836627 \times 10^{-2}$</td>
<td>$1.716823 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>1/12</td>
<td>1/700</td>
<td>$3.187941 \times 10^{-2}$</td>
<td>$1.428786 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>1/14</td>
<td>1/700</td>
<td>$2.727120 \times 10^{-2}$</td>
<td>$1.223664 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$\beta = 1.0207$</td>
<td>$\beta = 1.0115$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$C_\beta = 0.4028$</td>
<td>$C_\beta = 0.1764$</td>
</tr>
</tbody>
</table>
order of convergence in time with parameter $b$. Tables 2 and 4 give corresponding results for $b$ for the two velocity fields. From these results we clearly see that the two schemes developed are corroborated to be second order in space and time. We also see that BE-ELLAM, as was expected, is second order in space but only first order in time. The orders are more apparent in the more rapidly changing velocity field $V = 1 - 0.5x$ (Tables 3 and 4). Tables 1 and 3 show that when the time step $\Delta t$ is very small, the BRKC, FRKC, and BE-ELLAM schemes generate solutions with comparable errors. This is expected since all the schemes are second-order accurate in space and the temporal errors are negligible. In practice one wishes to use as large as possible time step in numerical simulations to enhance the efficiency without sacrificing accuracy. Therefore, Tables 2 and 4 bear more compu-

<table>
<thead>
<tr>
<th>Method</th>
<th>$\Delta t$</th>
<th>$\Delta x$</th>
<th>$L_2$ Error</th>
<th>$L_1$ Error</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BRKC</strong></td>
<td>1/500</td>
<td>1/60</td>
<td>6.117323 x 10^{-3}</td>
<td>2.475200 x 10^{-3}</td>
</tr>
<tr>
<td>1/500</td>
<td>1/70</td>
<td>3.495166 x 10^{-3}</td>
<td>1.323890 x 10^{-3}</td>
<td></td>
</tr>
<tr>
<td>1/500</td>
<td>1/80</td>
<td>2.429728 x 10^{-3}</td>
<td>9.213634 x 10^{-4}</td>
<td></td>
</tr>
<tr>
<td>1/500</td>
<td>1/90</td>
<td>1.847878 x 10^{-3}</td>
<td>6.923624 x 10^{-4}</td>
<td></td>
</tr>
<tr>
<td>1/500</td>
<td>1/100</td>
<td>1.408213 x 10^{-3}</td>
<td>5.172978 x 10^{-4}</td>
<td></td>
</tr>
</tbody>
</table>

$\beta = 2.88265$  
$C_x = 610.3031$  
$C_x = 475.0919$

| **FRKC** | 1/500 | 1/60 | 1.159784 x 10^{-2} | 5.085293 x 10^{-3} |
| 1/500 | 1/70 | 8.458101 x 10^{-3} | 3.559665 x 10^{-3} |
| 1/500 | 1/80 | 5.902566 x 10^{-3} | 2.528298 x 10^{-3} |
| 1/500 | 1/90 | 3.967780 x 10^{-3} | 1.713444 x 10^{-3} |
| 1/500 | 1/100 | 3.488353 x 10^{-3} | 1.511827 x 10^{-3} |

$\beta = 2.4833$  
$C_x = 308.5422$  
$C_x = 134.2145$

| **BE-ELLAM** | 1/500 | 1/60 | 6.673372 x 10^{-3} | 2.733189 x 10^{-3} |
| 1/500 | 1/70 | 4.124203 x 10^{-3} | 1.573640 x 10^{-3} |
| 1/500 | 1/80 | 3.124454 x 10^{-3} | 1.190370 x 10^{-3} |
| 1/500 | 1/90 | 2.606328 x 10^{-3} | 9.991517 x 10^{-4} |
| 1/500 | 1/100 | 2.240822 x 10^{-3} | 8.398474 x 10^{-4} |

$\beta = 2.1018$  
$C_x = 33.5250$  
$C_x = 17.3620$

Table 4. Order of convergence in space with $V(x, t) = 1 - 0.5x$
tional relevance since very large time steps are normally desired. One sees that the BRKC and FRKC schemes further reduce the temporal errors in BE-ELLAM, which are themselves significantly smaller than most Eulerian methods. This justifies the appropriateness of these schemes when large time steps are to be taken. The constant $C_p$ is much smaller than $C_s$ for all three schemes corroborating that time truncation errors are smaller than spatial errors, a strong advantage of characteristic methods in general.

4 DESCRIPTION OF SOME OTHER METHODS

The necessity to numerically solve advection-dominated advection–diffusion equation with high accuracy has lead to the development of many specialized methods. In this section we briefly describe some well perceived methods which are widely used in practice. In the next section we carry out experiments to compare the performance of the BRKC and FRKC schemes with these methods. In our description of these methods, we impose the same general assumptions on the velocity field $V(x, t)$ and the diffusion coefficient $D(x, t)$ as in Section 2 and describe the methods with Dirichlet boundary conditions.

4.1 Galerkin and Petrov–Galerkin finite element methods

The linear Galerkin (GAL), quadratic Petrov–Galerkin (QPG), and cubic Petrov–Galerkin (CPG) FEM methods with a Crank–Nicholson time discretization and Dirichlet inflow and outflow boundary conditions for eqn (1) are described by the following formulation

$$\begin{align*}
\int_a^b U(x, t^{n+1}) w_i(x) \, dx - \int_a^b \frac{\Delta t}{2} V U x_d - D U_i (x, t^{n+1}) w_i(x) \, dx \\
+ \int_a^b \frac{\Delta t}{2} (V U - D U_i) (x, t^{n+1}) w_i(x) \, dx \\
+ \int_a^b \frac{\Delta t}{2} \left( f(x, t^{n+1}) + f(x, t^n) \right) w_i(x) \, dx
\end{align*}$$

where the trial function $U(x, t^{n+1})$ is piecewise linear on the intervals defined by the partition. The three methods differ in their choices of the test functions $w_i(x)$ ($i = 1, 2, \ldots, I - 1$). In the GAL method, the test functions are chosen from the space of the trial functions, thus, $w_i(x)$ are the hat functions defined by eqn (26) where $w_i(x)$ replaces $w_i(x, t^{n+1})$. The Petrov–Galerkin methods were designed to improve the GAL method by introducing some upwinding in the space. The QPG methods use test functions, which are the sum of the standard hat functions and quadratic asymmetric perturbation terms, defined by

$$w_i(x) = \begin{cases} 
\frac{(x-x_{i-1})}{\Delta x_i} + 3x_i \frac{(x-x_{i-1})(x-x_i)}{\Delta x_i^3}, & x \in [x_{i-1}, x_i], \\
\frac{(x-x_{i-1})}{\Delta x_i} - 3x_i \frac{(x-x_{i-1})(x-x_i)}{\Delta x_i^3}, & x \in [x_i, x_{i+1}], \\
0, & \text{otherwise},
\end{cases}$$

(39)

where the parameter $x_i = \coth(V_i \Delta x_i/2D_i) - (2D_i/V_i \Delta x_i)$ with $V_i$ and $D_i$ being the arithmetic mean of the velocity field and the diffusion coefficient over the interval $(x_{i-1}, x_i)$. The CPG methods on the other hand use test functions which have symmetric cubic perturbation terms added to the hat functions as follows

$$w_i(x) = \begin{cases} 
\frac{(x-x_{i-1})}{\Delta x_i} + \gamma_i \frac{(x-x_{i-1})(x-x_i)(x-x_{i+1})(x_{i+1}-2x_i)}{\Delta x_i^3}, & x \in [x_{i-1}, x_i], \\
\frac{(x-x_{i-1})}{\Delta x_i} - \gamma_i \frac{(x-x_{i-1})(x-x_i)(x-x_{i+1})(x_{i+1}-2x_i)}{\Delta x_i^3}, & x \in [x_i, x_{i+1}], \\
0, & \text{otherwise},
\end{cases}$$

(40)

where $\gamma_i = Cr_t$, with $Cr_t = V_i \Delta t/\Delta x_i$ is the Courant number averaged over the interval $(x_{i-1}, x_i)$. Although many other Petrov–Galerkin formulations are also known to solve eqn (1) reasonably well, the coefficients used in the QPG and CPG methods here have been chosen optimally and these methods are among the most popular ones in practice.

4.2 The streamline diffusion FEM methods

The Streamline Diffusion method (SDM) is applied to the non-conservative form of eqn (1) given by

$$-\mathcal{L} u + V(x, t) u_t + V_i(x, t) u - (D(x, t) u_t)_x = f(x, t),$$

(41)

Here we describe the linear SDM formulation for this equation. We divide the domain into the time slabs $[a, b] \times (t^n, t^{n+1}]$, and successively on each slab, we seek a continuous and piecewise linear function $U(x, t)$ (discontinuous in time at $t^n$ and $t^{n+1}$) which satisfies the following formulation

$$\int_a^{t^{n+1}} \int_a^b [U_t + V U_x + V_i U] [w + \delta(w_t + V w)] \, dx \, dt$$

$$+ \int_a^{t^{n+1}} \int_a^b D U_i w_i \, dx \, dt$$

$$+ \int_a^{t^{n+1}} \int_a^b (D U_i)_x(w_t + V w)_x \, dx \, dt + \int_a^{t^{n+1}} U^n w^n \, dx$$

$$= \int_a^{t^{n+1}} f[w + \delta(w_t + V w)] \, dx \, dt + \int_a^{t^{n+1}} U^n w^n \, dx.$$  

(42)

The test function $w$ is piecewise linear in both space and time (bilinear) in the slab, and is discontinuous at time $t^n$.
and \( r^{n+1} \) and is zero at \( x = a \) and \( x = b \). In eqn (42) \( w_i^* = \lim_{\|e^\| \to 0} w(x, t) \), \( U_0 = u_0(x) \), and \( \delta \) is a free parameter, described below, that has significant influence on the accuracy of the scheme. There are many relations that can be used for the parameter \( \delta \). One of the most widely used relation is \( \delta = C (h/\sqrt{1 + V^2}) \), where \( h \) is the mesh diameter and \( C \) is a constant to be chosen \( ^{36} \). The choice of \( \delta \) determines the amount of diffusion applied in the direction of the characteristics and therefore has a great effect on the accuracy of the scheme. One requires that \( C \) is large enough to produce non-oscillatory solution, but not too large to damp the solution. This choice is, in general, problem dependent and not clear in practice. Although there are more improved versions of SDM method with shock capturing capacity which produce better approximations, they usually have non-linear formulations (even though they model linear equations), have more free parameters similar to \( \delta \) (described above) that need to be chosen carefully, and also have higher computational cost. The formulation (42) is the one we chose for numerical experiments described in the next section.

4.3 The continuous and discontinuous Galerkin FEM methods

The Continuous and Discontinuous Galerkin methods (CGM and DGM) \(^{46,47} \) apply to the non-conservative form of eqn (1) given by eqn (41). The domain \( \Omega_e = (a, b) \times (0, T) \) is divided into a quasi-uniform triangulation with side length \( h \), and Dirichlet data is assumed on the inflow portion of the boundary denoted by \( \Gamma_{in} \) and given by \( V(x, t) \cdot n < 0 \), where \( V(x, t) = (V(x, t),1) \) gives the characteristic direction and \( n = (n_1, n_2) \) is the outward unit normal vector. The boundary data given at the outflow portion of the boundary are not used in the formulation. In the DGM methods \(^{35,36,47} \) one seeks a discontinuous approximation \( U(x, t) \), which lies in the space \( P_n(\mathcal{T}) \) of polynomials of degree at most \( n \) on each triangle \( \mathcal{T} \), and satisfies the equation

\[
\int_{\mathcal{T}} (U_t + VU_x + V_t U) \, dx \, dt = \int_{\Gamma_{in}(\mathcal{T})} U^+ w \, V \cdot n \, dS + \int_{\Gamma_{out}(\mathcal{T})} DU^+ w \, n \, dS
\]

\[
= \int_{\mathcal{T}} fw \, dx \, dt - \int_{\Gamma_{in}(\mathcal{T})} U^- w \, V \cdot n \, dS + \int_{\Gamma_{out}(\mathcal{T})} DU^- w \, n \, dS \quad \text{for all } w \in P_n(\mathcal{T}),
\]

where \( \Gamma_{in}(\mathcal{T}) \) is the inflow boundary of \( \mathcal{T} \) exclusive of any sides on the boundary of the domain, \( U^+ (x, t) \equiv \lim_{\|e^\| \to 0} U((x, t) \pm \epsilon V) \), and \( U^- \) is the solution at the previous element or an interpolation of the prescribed Dirichlet data for sides on \( \Gamma_{in} \). The CGM \(^{44,46} \) is formulated in the same way by requiring a solution \( U(x, t) \in P_n(\mathcal{T}) \) which satisfies eqn (43), however there are two main differences: First, the trial function \( U(x, t) \) is required to be continuous over the domain \( \Omega_e \), and the test functions are in \( P_n(\mathcal{T}) \), where \( n(\mathcal{T}) \) is the number of inflow sides of \( \mathcal{T} \) has. Secondly, the continuity requirement in CGM makes the second terms on both sides of eqn (43) cancel each other. In both of these methods, one iterates over the elements solving a linear system of order equal to the degree of freedom on each element which makes these schemes quasi-explicit. In the next section, we consider the lowest-order CGM (\( n = 2 \)), and DGM (\( n = 1 \)).

4.4 High resolution methods (MUSCL and MINMOD)

High resolution methods from computational fluid dynamics are known to be good for purely hyperbolic equations. An extension of these methods to eqn (1) is based on time-splitting of the equation, as described below, and then a high-order Godunov method can be used to solve the advective part, with a mixed FEM method to solve the diffusive part \(^{17} \). We consider two such schemes, the first based on Monotone Upstream-centered Scheme for Conservation Laws (MUSCL) which was developed by van Leer \(^{54} \), and a second, based on a generalization of the first, called the MINMOD, which was developed by Harten et al. \(^{28,51} \). Assuming that \( U^n(x) \) approximates the solution \( u(x, t^n) \) of eqn (1), we can generate an approximation of \( u(x, t^{n+1}) \) as follows: First the MUSCL or MINMOD scheme, described below, can be applied to find a solution of the advective equation

\[
\tilde{u}_t + (V \tilde{u})_x = 0 \quad \text{for } (x, t) \in [a, b] \times (t^n, t^{n+1}],
\]

\[
\tilde{u}(x, t^n) = U^n(x) \quad \text{for } x \in [a, b],
\]

which we denote \( \tilde{u}^{n+1}(x) \). Then the mixed method can be used to solve

\[
\tilde{u}^*_t - (D \tilde{u}^*_x)_x = f \quad \text{for } (x, t) \in [a, b] \times (t^n, t^{n+1}],
\]

\[
u^*(x, t^n) = U^{n+1}(x) \quad \text{for } x \in [a, b]
\]

whose solution is the approximation \( U^{n+1}(x) \) of \( u(x, t^{n+1}) \). We note the well-known fact that the mixed method in lowest-order approximation space and a trapezoidal rule of integration is equivalent to the block-centered finite difference scheme \(^{22} \).

Now we describe the MUSCL and MINMOD schemes. Unlike the other methods discussed here which are node based, MUSCL and MINMOD are cell-centered based methods; i.e. the solution is approximated at the points \( x_{i-1/2} \) (\( i = 1, \ldots, I \)), where \( x_{i-1/2} \) is the mid point of the interval \( [x_{i-1}, x_i] \). To simplify the presentation, we
assume that the partition (2) is uniform. The MUSCL or \textit{MINMOD} formulation of eqn (44) is given by

\[
\begin{align*}
U_{i-1/2}^{n+1} & = U_{i-1/2}^{n} - \frac{\Delta t}{\Delta x} \left( V(x_{i-1/2}, t^*) U_{i-1/2}^{n} ight) - V(x_{i+1/2}, t^*) U_{i+1/2}^{n}, \\
& \quad \text{for } i = 1, \ldots, I, \tag{46}
\end{align*}
\]

where the Courant number is assumed not to exceed one (a stability requirement) and the left state \( U_{i-1/2}^{n} \) is given by

\[
U_{i-1/2}^{n} = \begin{cases} 
U_{i-1/2}^{n} + \frac{\Delta t}{\Delta x} \left( 1 - V(x_{i-1/2}, t^*) \frac{\Delta x}{\Delta t} \right) \delta U_{i-1/2}^{n}, & \text{for } i = 1, \ldots, I, \\
U_{1/2}^{n}, & \text{for } i = 0.
\end{cases}
\]

The \textit{MINMOD} formulation uses the slope given by

\[
\begin{align*}
\delta U_{i-1/2}^{n} & = \left\{ \begin{array}{ll}
\Delta_{+} U_{i-1/2}^{n} & \text{if } |\Delta_{+} U_{i-1/2}^{n}| \leq |\Delta_{-} U_{i-1/2}^{n}|, \\
\Delta_{-} U_{i-1/2}^{n} & \text{otherwise},
\end{array} \right.
\tag{47}
\end{align*}
\]

where the difference operators are given by

\[
\Delta_{\pm} U_{i-1/2}^{n} = \begin{cases} 
\frac{2U_{i-1/2}^{n} - U_{i}^{n} - U_{i-2}^{n \pm 1/2}}{\Delta x}, & \text{for } i = 1, \\
\frac{U_{i-1/2}^{n} - U_{i-3/2}^{n \pm 1/2}}{\Delta x}, & \text{for } i = 2, \ldots, I, \tag{48}
\end{cases}
\]

and

\[
\Delta_{\pm} U_{i-1/2}^{n} = \begin{cases} 
\frac{U_{i+1/2}^{n} - U_{i-1/2}^{n \pm 1/2}}{\Delta x}, & \text{for } i = 1, \ldots, I - 1, \\
\frac{2U_{i+1/2}^{n} - 3U_{i+3/2}^{n}}{\Delta x}, & \text{for } i = I. \tag{49}
\end{cases}
\]

The MUSCL formulation on the other hand uses the following definition for the slope

\[
\begin{align*}
\delta U_{i-1/2}^{n} & = \min \left\{ \Delta_{\text{lim}} U_{i-1/2}^{n}, |\Delta_{-} U_{i-1/2}^{n}| \right\} \cdot \text{sgn}(\Delta_{-} U_{i-1/2}^{n}) \\
& \quad \text{for } i = 1, \ldots, I, \tag{50}
\end{align*}
\]

The parameter \( x_{\text{lim}} \) in eqn (52) is 2 for \( i = 1, \ldots, I - 1 \) and 1 otherwise, which is the upper bound that allows the steeper representation of sharp fronts.

5 NUMERICAL EXPERIMENTS

In this section we describe numerical experiments which we use to compare the Runge–Kutta characteristic methods developed in this paper (using both forward- and back-tracking) with several generally well-regarded numerical schemes, including various Galerkin and Petrov–Galerkin finite element methods, streamline diffusion finite element methods, continuous and discontinuous Galerkin methods, and high resolution methods in computational fluid dynamics. We apply these methods to two standard test problems (a smooth Gaussian distribution and a step function) for which we have analytic solutions to the advection–diffusion equation. In addition, each of these functions are typically used to test for numerical artifacts of proposed schemes, such as numerical stability and diffusion, spurious oscillations, phase errors, and Gibbs type effects near sharp fronts.

In order to test these proposed schemes for advection-dominated transport, we consider the model equation (1) over the time period \( t \in [0, 1] \) with a variable velocity \( V(x, t) = 1 + 0.1x \) for Test Problem I and a constant velocity field of \( V(x, t) = 1 \) for Test Problem II, and a relatively small diffusion coefficient of \( D = 10^{-4} \). We consider both test problems for our initial-boundary conditions with the results for the Gaussian shown in Figs. 2–8 beginning with label I and the step function in Figs. 9–13. For clarity of exposition, we have arranged the numerical methods into 5 groups based on common characteristics of their behavior and implementation. These groups are organized according to the following table:

<table>
<thead>
<tr>
<th>Group</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Runge–Kutta characteristic methods (BRKC and FRKC)</td>
</tr>
<tr>
<td>2</td>
<td>Crank–Nicholson FEM (Galerkin, Quadratic and Cubic Petrov–Galerkin)</td>
</tr>
<tr>
<td>3</td>
<td>Streamline Diffusion (with various selections of the control parameter)</td>
</tr>
<tr>
<td>4</td>
<td>Continuous and Discontinuous Galerkin</td>
</tr>
<tr>
<td>5</td>
<td>High resolution methods in fluid dynamics (MUSCL and \textit{MINMOD})</td>
</tr>
</tbody>
</table>

In our experiments, we have systematically varied the space and time steps to examine the performance of each method. For each grouping we have chosen to display 3 plots which provide a fair illustration of the accuracy of each method, their potential beneficial properties, as well as their possible undesirable numerical artifacts. For comparison to BRKC and FRKC, the first plot in each figure, labeled (a), presents the evolved solution for all methods in the group.
using a common space mesh ($\Delta x = 1/60$ for Problem I, $\Delta x = 1/100$ for Problem II) and a time step as close as possible to the BRKC/FRKC time step of $\Delta t = 1/10$, but chosen small enough to ensure stability. The third plot of the figure (labeled (c)) for each grouping shows the solutions with an optimally efficient and reasonable choice of space and time steps to produce a qualitatively comparable solution to that of the BRKC and FRKC schemes in Figs. 2 and 9, respectively. The second plot in each figure (labeled (b)) shows an intermediate stage for each grouping. For example, Fig. 5 consists of 3 plots ((a)–(c)) which shows the solutions for model problem I at time $T = 1$ for the methods in group 4 (continuous and discontinuous Galerkin finite element methods) with $(\Delta x, \Delta t)$ taken as (1/60, 1/60), (1/120, 1/180), and (1/180, 1/180), respectively. The only exception to this labeling pattern is the collection of plots for the TVD schemes (MUSCL and Minmod) applied to model problem I, where we have included additional plots (Figs. 7 and 8) of solutions to simplified problems to help explain what appears at first to be atypical behavior for these schemes. More explanation is provided at the end of Section 5.1.

To gauge algorithm efficiency, we also compare the timings for each method with different space and time steps, especially the timings for each method to achieve the accuracy depicted in plot (c) of each figure. These results are presented in Tables 5–9, using the groupings of algorithms as we described in our list above. We have presented both $L^2$ and $L^1$ errors in the tables, since the $L^2$ error is often used for linear advection–diffusion equations while the $L^1$ error is preferred for first-order hyperbolic equations. We realize, of course, that some code optimization may be possible but feel that these timings are representative of each scheme’s efficiency on these model problems.

### 5.1 Model problem I: Gaussian

Our first model problem considers the transport of the one dimensional Gaussian hill over the spatial domain $[0, 1.4]$. The initial condition is given by

$$u_0(x) = \exp\left(-\frac{(x-x_0)^2}{2\sigma^2}\right),$$  \hspace{1cm} (54)

where the center $x_0 = 0$ and the spread $\sigma = 10^{-3/2} \approx 0.0316$, which accordingly determines the steepness of the Gaussian. In case the coefficients $V$ and $D$ are constant, it is easy to verify that

$$u(x, t) = \sqrt{\kappa(t)} \exp\left(-\frac{1}{2} \kappa(t)\eta(x, t)^2\right)$$  \hspace{1cm} (55)

is the exact solution to the homogeneous equation (1) (i.e., $f(x, t) = 0$) where

$$\eta(x, t) = \frac{x - x_0 - lt}{\sigma}, \quad \kappa(t) = \left(1 + \frac{2Dt}{\sigma^2}\right)^{-1}. \hspace{1cm} (56)$$

We have chosen to display the results for relatively small diffusion ($D = 10^{-4}$) and a relatively simple linear velocity field $V(x) = 1 + ax$, since these are representative of results we have observed for both constant and variable coefficient equations. In this case the model equation (1) is no longer homogeneous, and the right-hand side becomes

$$f(x, t) = -\frac{a}{\sigma^4} \kappa(t)^{5/2} P(x, t; a, D, \sigma) \exp\left(-\frac{1}{2} \kappa(t)\eta(x, t)^2\right)$$  \hspace{1cm} (57)

where $P$ is a multivariate polynomial defined by

$$P(x, t; a, D, \sigma) = adV(x)^2t - 2D(D + (x-x_0)V(x))^t + \left(2D(x-x_0) + D(3x^2 - 4x_0x + x_0^2 - \sigma^2) + \sigma^2 V(x)^2\right)t^2 - \left(2D (x-x_0)^2 + \sigma^2\right) + \sigma^2 (x-x_0)V(x)\right)t - \sigma^4.$$  \hspace{1cm} (58)

As in the earlier experiment in Section 3.3 for determining the order of convergence, we set $a = 0.1$.

Notice that in practical simulations, a fixed (relatively coarse) spatial mesh is often given and a time step is often chosen as large as possible to enhance the efficiency of simulations without sacrificing accuracy. Hence, in our numerical experiments we have picked baseline parameters of $\Delta x = 1/60$ and $\Delta t = 1/10$, for each numerical scheme in our testbed. We choose the spatial grid size of $\Delta x = 1/60$ so that the steep analytical solution can be represented accurately, with which we compare all the numerical solutions. With the relatively coarse time step of $\Delta t = 1/10$, the Runge–Kutta and the ELLAM schemes have already generated accurate nu-

<table>
<thead>
<tr>
<th>Method</th>
<th>$\Delta t$</th>
<th>$\Delta x$</th>
<th>$L_2$ Error</th>
<th>$L_1$ Error</th>
<th>CPU</th>
<th>Figures</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRKC</td>
<td>1/5</td>
<td>1/60</td>
<td>8.370973 x $10^{-03}$</td>
<td>3.151613 x $10^{-3}$</td>
<td>1.2</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/10</td>
<td>1/60</td>
<td>2.581547 x $10^{-03}$</td>
<td>9.348955 x $10^{-4}$</td>
<td>2.1</td>
<td>2</td>
</tr>
<tr>
<td>FRKC</td>
<td>1/5</td>
<td>1/60</td>
<td>8.426960 x $10^{-03}$</td>
<td>3.571262 x $10^{-3}$</td>
<td>0.7</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/10</td>
<td>1/60</td>
<td>2.628836 x $10^{-03}$</td>
<td>1.039830 x $10^{-3}$</td>
<td>1.1</td>
<td>2</td>
</tr>
</tbody>
</table>
Second-order characteristic methods for advection–diffusion equations and comparison to other schemes

Table 6. Results for GAL, QPG, and CPG (CPU time is in seconds)

<table>
<thead>
<tr>
<th>Method</th>
<th>$\Delta t$</th>
<th>$\Delta x$</th>
<th>$L_2$ Error</th>
<th>$L_1$ Error</th>
<th>CPU</th>
<th>Figures</th>
</tr>
</thead>
<tbody>
<tr>
<td>GAL</td>
<td>1/60</td>
<td>1/60</td>
<td>$7.47789 \times 10^{-2}$</td>
<td>$4.676085 \times 10^{-2}$</td>
<td>15.6</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/69</td>
<td>1/60</td>
<td>$6.313931 \times 10^{-2}$</td>
<td>$3.769911 \times 10^{-2}$</td>
<td>17.9</td>
<td>3(a)</td>
</tr>
<tr>
<td></td>
<td>1/120</td>
<td>1/60</td>
<td>$2.928317 \times 10^{-2}$</td>
<td>$1.513940 \times 10^{-2}$</td>
<td>31.2</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/180</td>
<td>1/60</td>
<td>$1.606416 \times 10^{-2}$</td>
<td>$7.876317 \times 10^{-3}$</td>
<td>46.3</td>
<td>3(b)</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/60</td>
<td>$5.742975 \times 10^{-3}$</td>
<td>$2.797454 \times 10^{-3}$</td>
<td>128.9</td>
<td>–</td>
</tr>
<tr>
<td>QPG</td>
<td>1/60</td>
<td>1/60</td>
<td>$6.165167 \times 10^{-2}$</td>
<td>$3.299499 \times 10^{-2}$</td>
<td>15.6</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/69</td>
<td>1/60</td>
<td>$5.042626 \times 10^{-2}$</td>
<td>$2.597113 \times 10^{-2}$</td>
<td>17.9</td>
<td>3(a)</td>
</tr>
<tr>
<td></td>
<td>1/120</td>
<td>1/60</td>
<td>$2.444460 \times 10^{-2}$</td>
<td>$1.47130 \times 10^{-2}$</td>
<td>31.2</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/180</td>
<td>1/60</td>
<td>$2.329064 \times 10^{-2}$</td>
<td>$1.095447 \times 10^{-2}$</td>
<td>46.3</td>
<td>3(b)</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/60</td>
<td>$2.745785 \times 10^{-2}$</td>
<td>$1.292518 \times 10^{-2}$</td>
<td>128.9</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/60</td>
<td>1/120</td>
<td>$7.182440 \times 10^{-2}$</td>
<td>$4.207188 \times 10^{-2}$</td>
<td>31.2</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/120</td>
<td>1/120</td>
<td>$2.497636 \times 10^{-2}$</td>
<td>$1.711133 \times 10^{-2}$</td>
<td>62.0</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/180</td>
<td>1/120</td>
<td>$1.120812 \times 10^{-2}$</td>
<td>$5.025411 \times 10^{-3}$</td>
<td>92.7</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/120</td>
<td>$4.472463 \times 10^{-3}$</td>
<td>$1.907250 \times 10^{-3}$</td>
<td>257.5</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/60</td>
<td>1/180</td>
<td>$7.382145 \times 10^{-2}$</td>
<td>$4.433499 \times 10^{-2}$</td>
<td>46.3</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/120</td>
<td>1/180</td>
<td>$2.676195 \times 10^{-2}$</td>
<td>$1.257207 \times 10^{-2}$</td>
<td>92.7</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/180</td>
<td>1/180</td>
<td>$1.248906 \times 10^{-2}$</td>
<td>$5.547136 \times 10^{-3}$</td>
<td>138.7</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/180</td>
<td>$1.286150 \times 10^{-3}$</td>
<td>$5.298965 \times 10^{-4}$</td>
<td>385.5</td>
<td>3(c)</td>
</tr>
<tr>
<td>CPG</td>
<td>1/60</td>
<td>1/60</td>
<td>Unbounded</td>
<td>Unbounded</td>
<td>15.6</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/69</td>
<td>1/60</td>
<td>$1.502762 \times 10^{-2}$</td>
<td>$6.299219 \times 10^{-3}$</td>
<td>18.0</td>
<td>3(a)</td>
</tr>
<tr>
<td></td>
<td>1/120</td>
<td>1/60</td>
<td>$6.803069 \times 10^{-3}$</td>
<td>$3.010018 \times 10^{-3}$</td>
<td>31.3</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/180</td>
<td>1/60</td>
<td>$5.125371 \times 10^{-3}$</td>
<td>$2.40985 \times 10^{-3}$</td>
<td>46.6</td>
<td>3(b)</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/60</td>
<td>$4.476924 \times 10^{-3}$</td>
<td>$2.218433 \times 10^{-3}$</td>
<td>129.6</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/60</td>
<td>1/120</td>
<td>Unbounded</td>
<td>Unbounded</td>
<td>31.3</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/120</td>
<td>1/120</td>
<td>Unbounded</td>
<td>Unbounded</td>
<td>62.3</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/180</td>
<td>1/120</td>
<td>$2.221885 \times 10^{-3}$</td>
<td>$8.849107 \times 10^{-4}$</td>
<td>93.2</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/120</td>
<td>$5.708042 \times 10^{-4}$</td>
<td>$2.392513 \times 10^{-4}$</td>
<td>258.9</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/60</td>
<td>1/180</td>
<td>Unbounded</td>
<td>Unbounded</td>
<td>46.6</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/120</td>
<td>1/180</td>
<td>Unbounded</td>
<td>Unbounded</td>
<td>93.2</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/180</td>
<td>1/180</td>
<td>Unbounded</td>
<td>Unbounded</td>
<td>139.8</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/180</td>
<td>$3.023540 \times 10^{-4}$</td>
<td>$1.219287 \times 10^{-4}$</td>
<td>387.6</td>
<td>3(c)</td>
</tr>
</tbody>
</table>

Numerical solutions. However, all other methods in the testbed have implicit Courant restrictions on the time step either for the reason of stability or for the reason of accuracy, hence the time step may not be chosen as large as that permitted by the Runge–Kutta and ELLAM schemes. Therefore, for each of these methods (with the exception of MUSCL and MINMOD schemes), we first fix the space grid size of $\Delta x = 1/60$ and choose the time steps of $\Delta t = 1/60$, 1/120, 1/180, and 1/500. The time step of $\Delta t = 1/60$ is the largest possible time step for these methods to generate reasonable numerical solutions, while the time step of $\Delta t = 1/500$ is chosen to observe the performance of these methods for very fine time steps since in advection-dominated transport the temporal errors dominate the numerical solutions. We then reduce the space grid size $\Delta x$ to $\Delta x = 1/120$ and 1/180 and use the same time steps above to observe the effect of spatial errors. In this way, we can see what combination of $\Delta x$ and $\Delta t$ (and so the CPU) used with each of these methods can generate solutions comparable to the two Runge–Kutta methods (BRKC and FRKC) with $\Delta t = 1/60, \Delta t = 1/10$. As for the MUSCL and MINMOD schemes, since the maximum velocity is 1.14 over the interval [0,1.4], we were required by the CFL constraint to begin with $\Delta t = 1/69$ for $\Delta x = 1/60$, with $\Delta t = 1/137$ for $\Delta t = 1/120$, and with $\Delta t = 1/206$ for $\Delta x = 1/180$. The smallest time step is $\Delta t = 1/700$. We also used finer space grid sizes of $\Delta x = 1/300$ and 1/400 to match qualitatively the results of other schemes.

Fig. 2 shows the initial condition for model problem I (plotted with solid line at the inflow boundary) along with the evolved solution at time $t = 1$. Near the right boundary, we see the analytic solution (solid line), the
back-tracked BRKC solution (marker o), and the forward-tracked FRKC solution (dotted line). Both methods use a spatial step of $\Delta x = 1/60$ with a relatively large time step of $\Delta t = 1/10$. Each of the two solutions give very accurate approximations which are free of numerical oscillations, artificial diffusion, phase error, and adverse boundary effects. The timings for the BRKC and FRKC schemes are presented in Table 5 and provide baseline timings for all experiments.

Fig. 3(a) contains the plots of the analytic solution (solid line), the Galerkin approximation (labeled with symbol +), quadratic Petrov–Galerkin (dotted line) and cubic Petrov–Galerkin (labeled with symbol o) at time $t = 1$ with $\Delta x = 1/60$ where the time-stepping method employed is Crank-Nicholson. Initially, a time step of $\Delta t = 1/60$ was to have been used, however the CPG methods generated unbounded solutions. Hence in Fig. 3(a), a time step of $\Delta t = 1/60$ is used to guarantee stability according the CFL constraint. This plot shows that there are significant trailing oscillations for both the Galerkin and quadratic Petrov–Galerkin methods and to a lesser extent for the cubic Petrov–Galerkin method. All methods in this group however have a mild downstream phase error. As we decrease our mesh sizes to try to match the performance of the two Runge–Kutta characteristic schemes, we see in Fig. 3(a) and (c) that the trailing oscillations and numerical diffusion are less pronounced (for all but the cubic Petrov–Galerkin method), but still persist for all three methods until we decrease $\Delta x$ to 1/180 and $\Delta t$ to 1/500. In this case the numerical solutions are comparable to that of the BRKC and FRKC methods with $\Delta x = 1/60$ and $\Delta t = 1/10$, however, the CPU requirement for these methods is two orders of magnitude larger than that required to achieve similar results using the BRKC or FRKC methods (see Table 6).

Fig. 4 presents the corresponding results for the streamwise diffusion finite element method. This method requires the use of a control parameter C, and we present in Fig. 4 the plots for three values of this parameter in each of the subplots (a)–(c). In Fig. 4(a) we

<table>
<thead>
<tr>
<th>$C$</th>
<th>$\Delta t$</th>
<th>$\Delta x$</th>
<th>$L_2$ Error</th>
<th>$L_1$ Error</th>
<th>CPU</th>
<th>Figures</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>1/60</td>
<td>1/60</td>
<td>$1.123311 \times 10^{-1}$</td>
<td>$6.024287 \times 10^{-2}$</td>
<td>68.7</td>
<td>4(a)</td>
</tr>
<tr>
<td>0.1</td>
<td>1/60</td>
<td>1/60</td>
<td>$5.241310 \times 10^{-2}$</td>
<td>$2.723927 \times 10^{-2}$</td>
<td>68.6</td>
<td>4(a)</td>
</tr>
<tr>
<td>0.0001</td>
<td>1/60</td>
<td>1/60</td>
<td>$3.279756 \times 10^{-2}$</td>
<td>$1.715544 \times 10^{-2}$</td>
<td>69.1</td>
<td>4(a)</td>
</tr>
</tbody>
</table>

Table 7. Results for SDM with $C = 1$, 0.1, and 0.0001, (CPU is in seconds)
have used a grid of $\Delta x = \Delta t = 1/60$ which is the largest we have chosen since this method prefers the $\Delta t$ and $\Delta \hat{x}$ to be of the same order of magnitude. As Fig. 4 demonstrates, there are both leading and trailing oscillations, along with relatively strong numerical diffusion and a downstream phase error. These persist to a milder magnitude, we were able to take $\Delta x = 1/60$ and $\Delta \hat{x} = 1/180$ as required by the CFL constraint.

Table 7 presents the timings for this method. For each selection of $C$, the CPU time is presented to compute the solution for $t = 1$. The CPU requirement for this method is two to three orders of magnitude larger than that required to achieve similar results using the BRKC or FRKC methods. Another drawback of this method is that it is not clear in general how to choose the parameter $C$, which indicates that an iteration of this parameter may be necessary and the expense of this method will increase accordingly. Under the conditions in model I, the optimal value of $C$ among the three choices is $10^{-4}$. In model problem II, however, the optimal choice is $C = 0.1$.

Fig. 5 presents the results for the continuous (dotted line) and discontinuous (labeled with symbol +) Galerkin methods. Again because these methods prefer the space and time meshes to be of the same order of magnitude, we were able to take $\Delta t = \Delta \hat{x} = 1/60$. We see that there are significant leading and trailing oscillations and a mild phase error for these two schemes with the discontinuous Galerkin method performing somewhat better of the two. However, this later method does exhibit some overshoot near the maximum of the Gaussian, which persists in plot (b) where $\Delta x = 1/120$ and $\Delta t = 1/180$. The accuracy of the two Runge–Kutta characteristic schemes are matched in plot (c) by taking $\Delta x = 1/180$ and $\Delta t = 1/180$, but both the CGM and DGM require a CPU expense of more than 120 seconds (see Table 8) as compared to 1.1 seconds for the FRKC and 2.1 s for the BRKC (see Table 5).

The final schemes which we wish to observe are the two high resolution methods from computational fluid dynamics (MUSCL and MINMOD). In Fig. 6, we have plotted the results of the simulation of these schemes where we again must take a relatively small initial time step of $\Delta t = 1/69$ as required by the CFL constraint. The analytic solution is plotted as the solid line, while the MINMOD scheme uses a dotted line and the MUSCL scheme is plotted using the symbol +. The monotonicity of these methods is quite apparent, but there is a pronounced trailing non-negative oscillation as well as an overshoot near the peak of the pulse for both methods, with the MINMOD scheme producing the most overshoot (Notice that we are now solving the eqn (1) with a non-homogeneous right-hand side). Parts (b) and (c) of this figure, show that these artifacts persist until we greatly reduce the mesh sizes.

Fig. 7 is presented to help explain the apparent uncharacteristic behavior of these schemes as shown by Fig. 6. Here we simplify the model conditions so that the Gaussian initial condition is supported in the interior of the domain and centered at $x = 0.2$. This assumption implies that the corresponding Dirichlet inflow conditions are homogeneous. The two schemes may then be observed to behave generally in their more familiar

---

**Table 8. Results for CGM and DGM, (CPU is in seconds)**

<table>
<thead>
<tr>
<th>Method</th>
<th>$\Delta t$</th>
<th>$\Delta x$</th>
<th>$L_2$ Error</th>
<th>$L_1$ Error</th>
<th>CPU</th>
<th>Figures</th>
</tr>
</thead>
<tbody>
<tr>
<td>CGM</td>
<td>1/60</td>
<td>1/60</td>
<td>$4.574094 \times 10^{-2}$</td>
<td>$2.216362 \times 10^{-2}$</td>
<td>13.9</td>
<td>5(a)</td>
</tr>
<tr>
<td></td>
<td>1/120</td>
<td>1/60</td>
<td>$3.136981 \times 10^{-2}$</td>
<td>$1.399978 \times 10^{-2}$</td>
<td>27.7</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/180</td>
<td>1/60</td>
<td>$2.674044 \times 10^{-2}$</td>
<td>$1.150175 \times 10^{-2}$</td>
<td>41.5</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/60</td>
<td>$2.106013 \times 10^{-2}$</td>
<td>$8.915907 \times 10^{-3}$</td>
<td>115.1</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/60</td>
<td>1/120</td>
<td>$2.501450 \times 10^{-2}$</td>
<td>$1.191302 \times 10^{-2}$</td>
<td>28.4</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/120</td>
<td>1/120</td>
<td>$1.025814 \times 10^{-2}$</td>
<td>$4.547582 \times 10^{-3}$</td>
<td>55.3</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/180</td>
<td>1/120</td>
<td>$6.763004 \times 10^{-3}$</td>
<td>$2.919515 \times 10^{-3}$</td>
<td>85.1</td>
<td>5(b)</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/120</td>
<td>$3.618455 \times 10^{-3}$</td>
<td>$1.510866 \times 10^{-3}$</td>
<td>229.9</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/60</td>
<td>1/180</td>
<td>$1.956744 \times 10^{-2}$</td>
<td>$9.230535 \times 10^{-3}$</td>
<td>41.5</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/120</td>
<td>1/120</td>
<td>$6.110354 \times 10^{-3}$</td>
<td>$2.687964 \times 10^{-3}$</td>
<td>82.9</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/180</td>
<td>1/180</td>
<td>$3.381005 \times 10^{-3}$</td>
<td>$1.457474 \times 10^{-3}$</td>
<td>124.3</td>
<td>5(c)</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/180</td>
<td>$1.313403 \times 10^{-3}$</td>
<td>$5.526282 \times 10^{-4}$</td>
<td>345.1</td>
<td>–</td>
</tr>
<tr>
<td>DGM</td>
<td>1/60</td>
<td>1/60</td>
<td>$3.418172 \times 10^{-2}$</td>
<td>$1.749322 \times 10^{-2}$</td>
<td>18.1</td>
<td>5(a)</td>
</tr>
<tr>
<td></td>
<td>1/120</td>
<td>1/60</td>
<td>$2.197196 \times 10^{-2}$</td>
<td>$1.051318 \times 10^{-2}$</td>
<td>36.2</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/180</td>
<td>1/60</td>
<td>$1.810357 \times 10^{-2}$</td>
<td>$8.461050 \times 10^{-3}$</td>
<td>54.3</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/60</td>
<td>$1.335912 \times 10^{-2}$</td>
<td>$6.129611 \times 10^{-3}$</td>
<td>150.5</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/60</td>
<td>1/120</td>
<td>$2.206570 \times 10^{-2}$</td>
<td>$1.083687 \times 10^{-2}$</td>
<td>36.2</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/120</td>
<td>1/120</td>
<td>$9.208662 \times 10^{-3}$</td>
<td>$4.198609 \times 10^{-3}$</td>
<td>72.3</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/180</td>
<td>1/120</td>
<td>$6.136133 \times 10^{-3}$</td>
<td>$2.727802 \times 10^{-3}$</td>
<td>85.1</td>
<td>5(b)</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/120</td>
<td>$3.266766 \times 10^{-3}$</td>
<td>$1.406095 \times 10^{-3}$</td>
<td>301.0</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/60</td>
<td>1/180</td>
<td>$1.855220 \times 10^{-2}$</td>
<td>$8.954882 \times 10^{-3}$</td>
<td>54.3</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/120</td>
<td>1/180</td>
<td>$6.062851 \times 10^{-3}$</td>
<td>$2.722088 \times 10^{-3}$</td>
<td>108.5</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1/180</td>
<td>1/180</td>
<td>$3.502214 \times 10^{-3}$</td>
<td>$1.536899 \times 10^{-3}$</td>
<td>162.7</td>
<td>5(c)</td>
</tr>
<tr>
<td></td>
<td>1/500</td>
<td>1/180</td>
<td>$1.091111 \times 10^{-3}$</td>
<td>$4.559198 \times 10^{-4}$</td>
<td>451.0</td>
<td>–</td>
</tr>
</tbody>
</table>
manner, without the trailing hump that appeared in Fig. 6. In this case, one may still observe that a small but noticeable trailing oscillation persists for the MINMOD scheme, although the velocity term is relatively simple. Fig. 8 shows the improvement in accuracy for these schemes if we further simplify the velocity to be constant, since in this case the right-hand side of the model equation vanishes and we are dealing with a conservation law. Fig. 6 represents the composite effect of non-homogeneous inflow boundary and right-hand side influences upon these two schemes.

Under the common model conditions, in order to achieve the level of approximation of BRKC and FRKC, the MUSCL and MINMOD schemes required a CPU time of one order of magnitude larger than that used by the BRKC and FRKC schemes. This however is the least amount of time among all other methods in the testbed. Moreover, the MUSCL and MINMOD were originally designed and well suited for non-linear hyperbolic conservation laws with compactly supported initial data on the line.

5.2 Model problem II: Step function

To observe the performance of all the methods in the testbed for problems that have analytical solutions with a steep front, in this subsection we consider the transport of a one-dimensional diffused square wave. The initial condition is a step function of unit height supported on the interval $[0.2, 0.7]$.

$$u_0(x) = \begin{cases} 1, & \text{if } x \in [0.2, 0.7] \subset (0, 2), \\ 0, & \text{otherwise}. \end{cases}$$

We assume that the one-dimensional transport equation has constant coefficients so that the analytical solution for the homogeneous equation (1) can be determined in a closed form. In the experiments $V = 1$ and $D = 10^{-4}$ were chosen. Homogeneous inflow and outflow conditions were considered.
outflow Dirichlet boundary conditions are specified at $x = 0$ and $x = 2$. As long as the support of diffused square wave does not intersect the outflow boundary during the time period $[0, 1]$, the (semi-) analytical solution $u(x, t)$ can be expressed as

$$
\begin{align*}
\frac{1}{\sqrt{4\piDt}} \int_{-\infty}^{\infty} u_0(x - \mathcal{V}t - s) \exp \left( -\frac{s^2}{4Dt} \right) \, ds \\
= \frac{1}{2} \left[ \text{erf} \left( \frac{x - \mathcal{V}t - 0.2}{\sqrt{4Dt}} \right) - \text{erf} \left( \frac{x - \mathcal{V}t - 0.7}{\sqrt{4Dt}} \right) \right].
\end{align*}
$$

(60)

where $\text{erf}(x) = \frac{2}{\sqrt\pi} \int_{0}^{x} \exp(-s^2) \, ds$ denotes the standard error function.

For this model problem we choose a baseline spatial grid of $\Delta x = 1/100$. We have taken a time step of $\Delta t = 1/10$ for the BRKC and the FRKC schemes. Fig. 9 shows the initial condition and analytic solution (solid line), together with the BRKC and FRKC solutions.

The observations made from Problem I are still valid for all of the numerical schemes tested, but in the present case there several additional features we wish to point out which are now more pronounced.
The Galerkin and Petrov–Galerkin methods with Crank–Nicholson time stepping (see Fig. 10) shows strong oscillations (both up- and down-stream) near each steep front. Fig. 11 shows that the streamline diffusion finite element method exhibits a Gibbs' effect near each of the two steep fronts in the analytical solution. In this case, it appears that reducing the value of $C$ continues to improve the error, while in previous experi-
ments, we have shown that the error may have local minimum as a function of $C$ for this method. The continuous and discontinuous Galerkin finite element methods (see Fig. 12 parts b and c) have a behavior very similar to the streamline diffusion finite element method. This is to be expected since these methods are closely related. However, the streamline diffusion method has approximately double the unknowns which must be

---

**Fig. 6.** High resolution methods MUSCL and MINMOD.
(a) $\Delta x = 1/60$, $\Delta t = 1/69$. (b) $\Delta x = 1/120$, $\Delta t = 1/137$. (c) $\Delta x = 1/400$, $\Delta t = 1/700$.

**Fig. 7.** MUSCL and MINMOD (homogeneous Dirichlet b.c. with initial solution centered at 0.2). (a) $\Delta x = 1/60$, $\Delta t = 1/69$. (b) $\Delta x = 1/120$, $\Delta t = 1/137$. (c) $\Delta x = 1/400$, $\Delta t = 1/700$. 

---

Second-order characteristic methods for advection–diffusion equations and comparison to other schemes
determined and is correspondingly more expensive to compute. The MUSCL and Minmod schemes (Fig. 13) generate strictly monotone solutions with slight numerical diffusion as expected.

6 SUMMARY AND DISCUSSION

In this paper we developed two Runge–Kutta characteristic schemes (BRKC and FRKC) for linear advection–diffusion equations in one space dimension, which can be viewed as improved high-order time integration schemes over the previously developed ELLAM schemes. The derived schemes generate accurate solutions even if large time steps are used, and treat various combinations of boundary conditions in a mass-conservative manner. The numerical experiments presented in this paper showed that these two schemes have second-order asymptotic convergence rates in both space and time, and outperform many well-regarded methods in terms of the accuracy and efficiency in the context of one-dimensional, linear advection–diffusion equations. Previously, the authors developed an ELLAM scheme, which is a first-order (in-time) analogue of the Runge–Kutta schemes presented in this paper, for multidimensional, linear advection–diffusion equations. The numerical results showed that the ELLAM scheme already outperformed many widely used methods applied to multidimensional, linear advection–diffusion equations. We refer readers to the earlier work for details. To date, all the comparisons have been performed in the context of (one- or multidimensional) linear advection–diffusion equations. For the development of an ELLAM scheme for one-dimensional, nonlinear (Buckley–Leverett) advection–diffusion equations, we refer readers to the work of Dahle et al. For a more complete discussion of the advantages and disadvantages of many numerical methods cited in this paper, applied to one- and multidimensional, linear or non-linear advection–diffusion equations, and different computational conditions, we refer the readers to the excellent books.

In general, the Eulerian methods are relatively easy to formulate and implement as compared with character-
Second-order characteristic methods for advection–diffusion equations and comparison to other schemes

Characteristic methods (e.g., the ELLAM, BRKC, and FRKC, and other Eulerian–Lagrangian methods), which require more implementational work, especially for multidimensional problems. On the other hand, handling of boundary conditions seem problematic for several of the schemes presented.

The authors are currently developing and implementing these schemes in reservoir simulations with

**Fig. 10.** Galerkin, quadratic and cubic Petrov–Galerkin. (a) $\Delta x = \Delta t = 1/100$. (b) $\Delta x = 1/100$, $\Delta t = 1/200$. (c) $\Delta x = 1/100$, $\Delta t = 1/1000$.

**Fig. 11.** Streamline diffusion method, $C = 1, 0.1, 0.0001$. (a) $\Delta x = 1/100$, $\Delta t = 1/10$. (b) $\Delta x = \Delta t = 1/100$. (c) $\Delta x = 1/100$, $\Delta t = 1/200$. 

\[ \hat{D}_x \hat{D}_t \hat{1} = 100. \]
highly heterogeneous media and source/sink terms (injection/production wells). In this case, the velocity field is actually defined element wise. Hence, the backward Euler or Runge–Kutta characteristic tracking algorithms do not necessarily work well. Notice that in practice the governing advection–diffusion transport equation is coupled to a pressure equation, whose nu-
merical solution defines the velocity field for the transport equation. Therefore, we utilize the special form of the numerical velocity field (say, given by the Raviart–Thomas finite element space in the mixed finite element method) and use a semi-analytical technique to solve the characteristics analytically on a cell-by-cell basis. These results will be published elsewhere once they are completed.

REFERENCES


