A Characteristic Domain Decomposition and Space–Time Local Refinement Method for First-Order Linear Hyperbolic Equations with Interfaces

Hong Wang,[1] **Mohamed Al-Lawatia,**[2] **Robert C. Sharpley**[1]

[1] *Department of Mathematics,*
University of South Carolina,
Columbia, South Carolina 29208

[2] *Department of Mathematics and Statistics,*
Sultan Qaboos University,
P. O. Box 36, Al–Khod Postal Code 123,
Muscat, Sultanate of Oman

Received July 31, 1997; accepted March 9, 1998

We develop a characteristic-based domain decomposition and space–time local refinement method for first-order linear hyperbolic equations. The method naturally incorporates various physical and numerical interfaces into its formulation and generates accurate numerical solutions even if large time-steps are used. The method fully utilizes the transient and strongly local behavior of the solutions of hyperbolic equations and provides solutions with significantly improved accuracy and efficiency. Several numerical experiments are presented to illustrate the performance of the method and for comparison with other domain decomposition and local refinement schemes. © 1999 John Wiley & Sons, Inc. Numer Methods Partial Differential Eq 15: 1–28, 1999

Keywords: adaptive refinement; advection–reaction equations; characteristic methods; domain decomposition methods; Eulerian–Lagrangian methods; linear hyperbolic problems; local refinement techniques

I. INTRODUCTION

First-order linear hyperbolic partial differential equations model the reactive transport of solutes in groundwater and surface water, the movement of aerosols and trace gases in the atmosphere,

Correspondence to: Robert C. Sharpley
Contract grant sponsor: Dept. of Energy, Office of Naval Research
Contract grant no.: DE-FG05-95ER25266 ONR N00014-94-1-1163
 CCC 0749-159X/99/010001-28

meteorology, displacement process in oil production, seismic, fluid dynamics, gas dynamics, and many other important applications. It is well known that the solutions of these equations present steep fronts and even shock discontinuities, which need to be resolved accurately in applications and often cause severe numerical difficulties [1–5]. Conventional finite difference or finite element methods normally yield numerical solutions with severe nonphysical oscillation, numerical dispersion, or a combination of both. Moreover, practical problems often have various interfaces that introduce extra difficulties. Physical interfaces arise, for example, in the modeling of transport processes in composite media, leading to linear hyperbolic equations with discontinuous coefficients. Numerical interfaces occur in the application of domain decomposition and local refinement techniques. An identifying feature of ground-water contaminant transport and many other applications is the presence of large-scale fluid flows coupled with transient transport of physical quantities such as pollutants, chemical species, radionuclides, and temperature, which are generally smooth outside some small regions and may have sharp fronts inside where important chemistry and physics take place. An extremely fine global mesh in both space and time is not feasible due to the excessive computational cost. An alternative approach is to apply domain decomposition and local refinement techniques by partitioning the global domain into a number of sub-domains and solving the problem with fine meshes in both space and time within the sharp front regions (sub-domains) and coarse meshes outside (other sub-domains). This way, both accuracy and efficiency can be guaranteed, but at a cost of introducing numerical interfaces between different sub-domains. Another type of numerical interface occurs when one solves a problem defined on an infinite domain where one has to ''truncate'' the domain in numerical simulations.

Many domain decomposition and local refinement techniques have been developed for elliptic and parabolic equations [6–8], but it is more difficult to develop these techniques for hyperbolic equations. In this case, locally generated errors at the interfaces can be propagated into the domain so that the overall accuracy is decreased. Improper treatment of the interfaces might destroy the stability of the numerical methods. Furthermore, most methods for hyperbolic equations with interfaces are Eulerian methods, which require extremely small time-steps to maintain the accuracy and stability of the methods.

We develop a characteristic-based, noniterative and nonoverlapping domain decomposition and space–time local refinement method for solution of first-order linear hyperbolic equations with various physical and numerical interfaces. The method treats all the physical and numerical interfaces effectively in a systematic and uniform manner without introducing any extra scheme for the interfaces. It naturally incorporates the space–time local refinement capability in the scheme. It significantly reduces the time truncation errors present in the Eulerian methods and is not subject to the CFL restrictions. It generates accurate and stable solutions without oscillations, even if large time-steps are taken in the simulations. The method can be implemented in parallel and fully utilizes the physical properties of the governing equations. Finally, the method allows the use of different schemes within the framework of the Eulerian–Lagrangian localized adjoint method (ELLAM) [9–13]. Somewhat related approaches have also been utilized by others to develop domain decomposition methods for advection–diffusion equations [6, 7, 14, 15].

This article is organized as follows. In Section II, we describe a representative characteristic method for first-order linear hyperbolic equations with continuous coefficients, which will serve as an underlying numerical scheme in the development of our domain decomposition and local refinement method. In Section III, we develop a noniterative and nonoverlapping domain decomposition method. In Section IV, we describe a space–time local refinement scheme. In Section V, for comparison purposes, we describe some interface schemes that are currently in use. In Section VI, we conduct various numerical experiments to compare our method with those methods and to

observe the performance of our method for problems with static or adaptive space–time local grid refinement, and with discontinuous coefficients. In Section VII, we summarize our observations and results.

II. UNDERLYING NUMERICAL METHOD

A. Model Problem

In this section, we describe a characteristic method for first-order linear hyperbolic equations with continuous coefficients, which will serve as an underlying scheme in the development of our domain decomposition and local refinement method in the subsequent sections. As mentioned in Section I, the transient transport behavior and the steep fronts present in the exact solutions of the governing equations make the numerical simulation of first-order hyperbolic equations a challenging task. Traditional methods typically generate numerical solutions with severe non-physical oscillation, numerical dispersion, or a combination of both. Recent improvements have been made in two categories: Eulerian methods and characteristic methods.

Eulerian methods use fixed grids and incorporate some upstream weighting in their formulations to stabilize the schemes. Among the class of Eulerian methods are the Petrov–Galerkin methods, which improve over the standard Galerkin methods by adding some upwinding in the test functions [16–18]. Also included in this class are the streamline diffusion method and the continuous and discontinuous Galerkin methods [19–22]. The streamline diffusion method is accomplished by adding numerical diffusion only in the streamline direction of the governing equation. The continuous and discontinuous Galerkin methods, starting from an initial condition and a given inflow boundary condition, solve a local system over each element of a quasi-uniform space–time triangulation in an order consistent with the space–time domain considered. The class of Eulerian methods also includes many other methods such as the high-resolution methods from computational fluid dynamics, in particular, the Godunov methods and the essentially nonoscillatory methods (ENO) [23–26]. All these Eulerian methods are characterized by ease of formulation and implementation. However, time truncation errors dominate their solutions. In addition, these methods are subject to the CFL stability conditions, which put a restriction on the size of the time-step taken in numerical simulations. This may be a strong disadvantage in certain applications (e.g., atmospheric), where considerable time must be spent in computing coefficients (turbulence terms) from coupled equations or through model formulation [27].

Characteristic methods, on the other hand, make use of the transport nature of the governing equations. They combine the fixed Eulerian grids with a particle tracking along the characteristic curves of the governing equations. Among the characteristic methods are the Eulerian–Lagrangian method, the modified method of characteristics, and the operator splitting method [28–35]. The Lagrangian treatment in these methods greatly reduces the time truncation errors in the Eulerian methods. In addition, these methods alleviate the restrictions on the Courant number, thus allowing for large time-steps in the simulations. However, these methods fail to conserve mass and treat boundary conditions in an *ad hoc* manner.

In this section, we present a Runge–Kutta characteristic method for the initial-boundary value problems for first-order linear hyperbolic equations with continuous coefficients, which can be viewed as a higher-order improvement of the ELLAM schemes developed previously [9–13]. We consider the following model problem:

$$\mathcal{L}u := u_t + (V(x,t)u)_x + K(x,t)u = f(x,t),\ x \in (a,b), t \in (0,T],$$

$$u(a,t) = g(t),\ t \in [0,T],$$

$$u(x,0) = u_o(x),\ x \in [a,b]. \tag{2.1}$$

Here $V(x,t)$ is a velocity field, $K(x,t)$ is a first-order reaction coefficient, $f(x,t)$ is a given source term, $g(t)$ is a prescribed inflow boundary condition, $u_o(x)$ is a given initial condition, $u_x := \frac{\partial u}{\partial x}$, and $u_t := \frac{\partial u}{\partial t}$. For simplicity of exposition, we assume $V(x,t)$ positive, so that $x = a$ and $x = b$ are the inflow and outflow boundaries, respectively.

B. Variational Formulation and Characteristic Curves

In this subsection, we briefly outline a forward-tracked Runge–Kutta characteristic scheme, which is used as the underlying numerical scheme in developing our domain decomposition and space–time local refinement method. We refer readers to [36] for detailed information. We partition the space–time domain $\Omega := (a,b) \times [0,T]$ of problem (2.1) as follows:

$$a =: x_0 < x_1 < \cdots < x_I := b,$$

$$0 =: t_0 < t_1 < \cdots < t_N := T, \tag{2.2}$$

for positive integers I and N. The scheme uses a time marching algorithm, so we need to focus on only the time interval $(t_{n-1}, t_n]$. Moreover, it can accommodate a varying spatial grid on different time intervals. We will illustrate its flexibility when domain decomposition and local refinement techniques are developed.

Multiplying Eq. (2.1) by a continuous and piecewise smooth space–time test function that vanishes outside $\Omega_n := (a,b) \times (t_{n-1}, t_n]$, and integrating over the domain Ω, we obtain a space–time variational formulation

$$\begin{aligned}
&\int_a^b u(x,t_n)w(x,t_n)\,dx + \int_{t_{n-1}}^{t_n} u(b,t)w(b,t)V(b,t)\,dt \\
&\quad - \int_{t_{n-1}}^{t_n}\int_a^b u(w_t + Vw_x - Kw)\,dxdt \\
&= \int_a^b u(x,t_{n-1})w(x,t_{n-1}^+)\,dx + \int_{t_{n-1}}^{t_n}\int_a^b fw\,dxdt \\
&\quad + \int_{t_{n-1}}^{t_n} g(t)w(a,t)V(a,t)\,dt,
\end{aligned} \tag{2.3}$$

where we use the notation $w(x,t_{n-1}^+) := \lim_{t \to t_{n-1}^+} w(x,t)$ due to the discontinuity of $w(x,t)$ at time t_{n-1}.

The principle of ELLAM [10, 11] suggests selecting the test functions from the solution space of the homogeneous adjoint equation

$$\mathcal{L}^* w := -w_t(x,t) - V(x,t)w_x(x,t) + K(x,t)w(x,t) = 0, \tag{2.4}$$

so that the last term on the left-hand side of Eq. (2.3) is eliminated. Equation (2.4) can be rewritten as the following ordinary differential equation:

$$-\frac{d}{d\theta} w(X(\theta;\bar{x},\bar{t}),\theta) + K(X(\theta;\bar{x},\bar{t}),\theta)w(X(\theta;\bar{x},\bar{t}),\theta) = 0,$$

$$w(X(\theta;\bar{x},\bar{t}),\theta)|_{\theta=\bar{t}} = w(\bar{x},\bar{t}), \tag{2.5}$$

along the characteristics $X(\theta;\bar{x},\bar{t})$ being defined by

$$\frac{dX}{d\theta} = V(X,\theta)$$

$$X(\theta;\bar{x},\bar{t})|_{\theta=\bar{t}} = \bar{x}, \tag{2.6}$$

for any given point $(\bar{x},\bar{t})$ with $\bar{t} \in [t_{n-1},t_n]$. Thus, Eq. (2.5) leads to the following expression for the test function w inside the space–time domain $(a,b)\times(t_{n-1},t_n)$:

$$w(X(\theta;\bar{x},\bar{t}),\theta) = w(\bar{x},\bar{t})e^{-\int_\theta^{\bar{t}} K(X(\gamma;\bar{x},\bar{t}),\gamma)d\gamma}. \tag{2.7}$$

In the numerical scheme, we use a second-order Runge–Kutta quadrature, known as the Heun's method, to approximate the characteristics defined by Eq. (2.6). Namely, we define the approximate characteristic curve emanating from a point $(\bar{x},\bar{t})$, with $\bar{t} \in [t_{n-1},t_n]$, by

$$X(\theta;\bar{x},\bar{t}) := \bar{x} - \frac{(\bar{t}-\theta)}{2}\,[V(\bar{x},\bar{t}) + V(\bar{x}-(\bar{t}-\theta)V(\bar{x},\bar{t}),\theta)], \tag{2.8}$$

where θ is the time position parameter along the approximate characteristic.

Furthermore, we define $\tilde{x}$ at time t_n to be the head of the characteristic given by $x = X(t_{n-1};\tilde{x},t_n)$ of (2.7), which meets the foot x at time t_{n-1}. Similarly, x^* is the foot of characteristic at time t_{n-1} with the head x at time t_n, i.e., $x^* = X(t_{n-1};x,t_n)$. $b^*(t) = X(t_{n-1},b,t)$ is the foot of the characteristic at time t_{n-1} with its head $(b,t), t \in [t_{n-1},t_n]$, on the outflow boundary. We define $t^*(x) = t_{n-1}$ if the characteristic $X(\theta;x,t_n)$ does not backtrack to the inflow boundary $x = a$ during the time period $[t_{n-1},t_n]$, and $t^*(x)$ to be the time instant when the characteristic $X(\theta;x,t_n)$ backtracks to the inflow boundary $x = a$ (i.e., $X(t^*(x);x,t_n) = a$) otherwise. We let $\tilde{t}(x)$ be the time instant such that the characteristic $X(\theta;b,\tilde{t}(x))$ with the head $(b,\tilde{t}(x))$ meets the foot x at time t_{n-1} (i.e., $x = X(t_{n-1};b,\tilde{t}(x))$). The time increments over the domain Ω_n can be written as

$$\Delta t^{(I)}(x) := t_n - t^*(x),\ x \in [a,b],$$

$$\Delta t^{(O)}(x) := \tilde{t}(x) - t_{n-1},\ x \in [a,b],$$

$$\Delta t := t_n - t_{n-1}. \tag{2.9}$$

In the numerical formulation, we also introduce a local time refinement at both the inflow boundary $\Gamma_n^{(I)} := \{(a,s)|s \in [t_{n-1},t_n]\}$ and the outflow boundary $\Gamma_n^{(O)} := \{(b,s)|s \in [t_{n-1},t_n]\}$ of Ω_n by

$$t_{n,i}^{(I)} := t_n - i\frac{\Delta t}{CI_n},\ i = 0,\ldots,CI_n,$$

$$t_{n,i}^{(O)} := t_n - i\frac{\Delta t}{CO_n},\ i = 0,\ldots,CO_n, \tag{2.10}$$

respectively, where we have the flexibility of selecting the two positive integers CI_n and CO_n, which determine these partitions. In practice, these parameters should be selected to be on the order of the Courant number to guarantee stable boundary treatment, because the discretization on the two boundaries is in time not along the characteristics.

C. Numerical Scheme

To derive the numerical scheme, we approximate the second term on the right-hand side of Eq. (2.3) along the characteristics. For clarity of presentation, we use the variables y and θ to represent the spatial and temporal coordinates, respectively, of points in Ω_n, and reserve x and t for points on the space–time boundary of Ω_n, representing heads or feet of characteristics. We note that $y \in [a, X(\theta; b, t_n)]$ can be written as $y = X(\theta; x, t_n)$ for a unique $x \in [a, b]$ and that $y \in [X(\theta; b, t_n), b]$ can be written as $y = X(\theta; b, t)$ for a unique $t \in [\theta, t_n]$. The source term of Eq. (2.3) can then be approximated by being split into three parts, with each inner integral being approximated by the trapezoidal rule as follows [36]:

$$
\begin{aligned}
&\int_{t_{n-1}}^{t_n} \int_a^b f(y,\theta) w(y,\theta)\, dy d\theta \\
&= \int_{t_{n-1}}^{t_n} \int_a^{X(\theta;\tilde{a},t_n)} f(y,\theta) w(y,\theta)\, dy d\theta \\
&\quad + \int_{t_{n-1}}^{t_n} \int_{X(\theta;\tilde{a},t_n)}^{X(\theta;b,t_n)} f(y,\theta) w(y,\theta)\, dy d\theta \\
&\quad + \int_{t_{n-1}}^{t_n} \int_{X(\theta;b,t_n)}^{b} f(y,\theta) w(y,\theta)\, dy d\theta \\
&= \int_{\tilde{a}}^{b} \frac{\Delta t}{2} f(x,t_n) w(x,t_n) dx + \int_a^{b^*} \frac{\Delta t}{2} f(x,t_{n-1}) w(x,t_{n-1}^+)\, dx \\
&\quad + \int_a^{\tilde{a}} \frac{\Delta t^{(I)}(x)}{2} f(x,t_n) w(x,t_n) dx + \int_{t_{n-1}}^{t_n} \frac{(t_n - t)}{2} f(a,t) w(a,t) V(a,t)\, dt \\
&\quad + \int_{t_{n-1}}^{t_n} \frac{(t - t_{n-1})}{2} f(b,t) w(b,t) V(b,t)\, dt \\
&\quad + \int_{b^*}^{b} \frac{\Delta t^{(O)}(x)}{2} f(x,t_{n-1}) w(x,t_{n-1}^+)\, dx + R(f,w),
\end{aligned}
\tag{2.11}
$$

where $R(f, w)$ represents the error term due to the trapezoidal approximation of the integrals. Incorporating Eq. (2.11) into the variational formulation (2.3), yields a reference equation satisfied by the exact solution.

The numerical scheme is based on approximating the exact solution $u(x, t)$ of Eq. (2.1) by a piecewise-linear trial function at time t_n and at the outflow boundary $\Gamma_n^{(O)}$. Therefore, we set up test functions at time t_n and at the outflow boundary as follows: At time t_n we define the test functions $w_i(x, t_n)(i = 1, \ldots, I - 1)$ to be the hat function

$$
w_i(x,t_n) := \begin{cases} \frac{x - x_{i-1}}{\Delta x_i}, & x \in [x_{i-1}, x_i], \\ \frac{x_{i+1} - x}{\Delta x_{i+1}}, & x \in [x_i, x_{i+1}], \\ 0, & \text{otherwise}, \end{cases}
\tag{2.12}
$$

where $\Delta x_i := x_i - x_{i-1}$. At the outflow boundary the test functions $w_{I+i}(b, t)(i = 1, \ldots, CO_n - 1)$ are given by

$$
w_{I+i}(b,t) := \begin{cases} \frac{t_{n,i-1}^{(O)} - t}{\Delta t^f}, & t \in [t_{n,i}^{(O)}, t_{n,i-1}^{(O)}], \\ \frac{t - t_{n,i+1}^{(O)}}{\Delta t^f}, & t \in [t_{n,i+1}^{(O)}, t_{n,i}^{(O)}], \\ 0, & \text{otherwise}, \end{cases}
\tag{2.13}
$$

where $\Delta t^f := \frac{\Delta t}{CO_n}$. The test function $w_0(x,t_n)$ is defined only on $[x_0,x_1]$ by (2.12) and, similarly, w_{I+CO_n} is defined only on $[t_{n-1},t^{(O)}_{n,CO_n-1}]$ by (2.13), while w_I is so that $w_I(x,t_n)$ is defined by (2.12) for the interval $[x_{I-1},x_I]$, and $w_I(b,t)$ is defined by (2.13) on the interval $[t^{(O)}_{n,1},t_n]$. We extend these test functions to the interior of the domain by

$$w_i(X(\theta;\bar{x},\bar{t}),\theta) := w_i(\bar{x},\bar{t})e^{-\frac{(\bar{t}-\theta)}{2}[K(\bar{x},\bar{t})+K(X(\theta;\bar{x},\bar{t}),\theta)]}, \tag{2.14}$$

which are (trapezoid) approximations to the test functions defined in (2.7). Here for test functions $w_i(x,t_n)(i=0,\ldots,I),(\bar{x},\bar{t})=(x,t_n)$ with $x\in[a,b]$ and $\theta\in[t^*(x),t_n]$ and for test functions $w_i(b,t)(i=I,\ldots,I+CO_n),(\bar{x},\bar{t})=(b,t)$ with $t\in[t_{n-1},t_n]$ and $\theta\in[t_{n-1},t]$.

The trial function U, which approximates the exact solution u, has the form

$$U(x,t_n) := \sum_{i=0}^{I} U(x_i,t_n)w_i(x,t_n),\ x\in[a,b],$$

$$U(b,t) := \sum_{i=0}^{CO_n} U(b,t_i)w_{I+i}(b,t),\ t\in[t_{n-1},t_n], \tag{2.15}$$

for $n=1,\ldots,N$. Thus, we replace the exact solution u and the test functions w in the reference equation by the trial function U and test functions $w_i(i=0,\ldots,I+CO_n)$. Since $U(a,t_n)=g(t)$ is known from the prescribed inflow boundary condition and $U(b,t_{n-1})$ is known from the solution at the previous time level t_{n-1}, our scheme will be stipulated only for nodes $x_i(i=1,\ldots,I)$ and $t^{(O)}_{n,i}(i=1,\ldots,CO_n-1)$. However, to conserve mass, all the nodal test functions should sum to one (when no reaction is present) [10]. Taking this into consideration, the formulation of our scheme becomes

$$\begin{aligned}
&\int_a^b U(x,t_n)\hat{w}_i(x,t_n)\,dx + \int_{t_{n-1}}^{t_n} U(b,t)\hat{w}_i(b,t)V(b,t)\,dt \\
&= \int_a^b U(x,t_{n-1})\hat{w}_i(x,t^+_{n-1})\,dx + \int_{t_{n-1}}^{t_n} g(t)\hat{w}_i(a,t)V(a,t)\,dt \\
&\quad + \int_a^b \frac{\Delta t^{(I)}(x)}{2} f(x,t_n)\hat{w}_i(x,t_n)\,dx + \int_a^b \frac{\Delta t^{(O)}(x)}{2} f(x,t_{n-1})\hat{w}_i(x,t^+_{n-1})\,dx \\
&\quad + \int_{t_{n-1}}^{t_n} \frac{(t-t_{n-1})}{2} f(b,t)\hat{w}_i(b,t)V(b,t)\,dt \\
&\quad + \int_{t_{n-1}}^{t_n} \frac{(t_n-t)}{2} f(a,t)\hat{w}_i(a,t)V(a,t)\,dt,
\end{aligned} \tag{2.16}$$

where $\hat{w}_1 := w_1+w_0, \hat{w}_i := w_i$ for $i=2,\ldots,I+CO_n-2$, and $\hat{w}_{I+CO_n-1} := w_{I+CO_n-1} + w_{I+CO_n}$. In (2.16), we dropped the error term due to the trapezoidal approximation.

The numerical scheme (2.16) is known as the forward Runge–Kutta characteristic method (FRKC) [36]. The FRKC scheme generates tridiagonal (regularly structured in higher dimensions), well-conditioned, symmetric, and positive definite coefficient matrices, which can be solved efficiently even for multidimensional problems. It allows for large time-steps in the simulation without loss of accuracy. Moreover, it conserves mass and treats boundary conditions in a systematic way without requiring any artificial outflow boundary condition. This boundary treatment provides a natural way of developing domain decomposition and local refinement

techniques using this method. The reader is referred to [36] for implementational details of the FRKC scheme and its numerical experiments, which show that the FRKC scheme outperforms many widely used methods, including the Galerkin and Petrov–Galerkin finite element methods [16–18], streamline diffusion method [20, 22], continuous and discontinuous Galerkin methods [19, 21], the monotonic upstream-centered scheme for conservation laws (MUSCL), and the essentially nonoscillatory (ENO) scheme [23–26].

In the following sections, we use the FRKC scheme to develop a domain decomposition method with space–time local refinement capability for solution of problem (2.1) with interfaces. The domain decomposition method is general in that any other scheme within the ELLAM framework, such as the backward Euler ELLAM scheme or the backward Runge–Kutta characteristic method developed previously [9–13, 36], can be used instead of the FRKC scheme.

III. DOMAIN DECOMPOSITION

A. Various Types of Interfaces

In addition to the difficulties encountered in solving first-order hyperbolic equations mentioned in Section II.A, practical problems often have various physical and numerical interfaces, which introduce further complexities. When the physical domain (a, b) in Eq. (2.1) is composed of different sub-domains (d_{l-1}, d_l) that consist of different media for $l = 1, 2, \ldots, L$ where $d_0 := a$ and $d_L := b$, the velocity field $V(x, t)$ in Eq. (2.1) will have jump discontinuities at points $d_l \in (a, b)$ for $(l = 1, 2, \ldots, L - 1)$ and is continuous elsewhere. This leads to hyperbolic equations with discontinuous coefficients $V(x, t)$. In this case, Eq. (2.1) still holds on each sub-domain (d_{l-1}, d_l) for $l = 1, 2, \ldots, L$, but not on the physical interfaces $d_l (l = 1, 2, \ldots, L - 1)$. Instead, the mass conservation principle ensures the continuity of the flux across the interfaces:

$$V(d_l^-, t)u(d_l^-, t) = V(d_l^+, t)u(d_l^+, t),\ t \in [0, T],\ l = 1, 2, \ldots, L - 1, \tag{3.1}$$

where $u(d_l^-, t) := \lim_{x \to d_l, x < d_l} u(x, t)$ and $u(d_l^+, t) := \lim_{x \to d_l, x > d_l} u(x, t)$.

Equation (3.1) is used to enclose the initial-boundary value problem (2.1) across the physical interfaces. Note that Eq. (3.1) implies that the jump discontinuities in the coefficient $V(x, t)$ yield similar discontinuities in the exact solution $u(x, t)$. One has to approximate the solution $u(d_l^-, t)$ and $u(d_l^+, t)(l = 1, 2, \ldots, L - 1)$ very carefully at the interfaces so that the numerical solutions obtained are accurate, stable, and physically reasonable without any accompanying overshoot or excessive artificial diffusion nor artificially reflected waves [37, 38].

Numerical interfaces arise in the application of domain decomposition techniques, which have been applied widely in decomposing problems imposed over complicated geometric domains into a set of subproblems on much simpler sub-domains, in solving large size problems in parallel, and in solving problems in an efficient way. In all these cases, one divides the physical domain into several overlapping or nonoverlapping sub-domains and solves the problem on these sub-domains instead. This generates interfaces among different sub-domains [6–8, 39].

Numerical interfaces also occur when one solves problems imposed over unbounded physical domains. Since no computer can handle infinitely many data, one has to ''truncate'' the physical domains in numerical simulations. This truncation introduces an artificial open boundary, which requires special care. Otherwise, artificial waves may be generated and reflected back from the artificial outflow boundary, which is another type of numerical interface. Consequently, the accuracy or the stability of numerical methods will be destroyed and nonphysical numerical solutions obtained. A lot of work has been done in designing reasonable artificial outflow boundary conditions in the development of numerical schemes for hyperbolic equations [40, 41].

In this section, we develop a domain decomposition method for solution of problem (2.1) possibly coupled with various physical and numerical interfaces described above. Although it is derived for physical interfaces, Eq. (3.1) still holds at numerical interfaces due to the mass conservation principle. Therefore, in this article we always utilize Eq. (3.1) to treat both physical and numerical interfaces in a uniform manner. In the remaining part of this section, we first describe possible ways of passing information between different sub-domains sharing an interface, then we present a global algorithm to solve problems (2.1) and (3.1) by incorporating the interface treatment with the FRKC scheme (2.16) applied on each sub-domain.

B. Spatial Interfaces

For simplicity of exposition, we consider problem (2.1) with one interface in space at the point $d \in (a, b)$. The global space–time domain $\Omega := (a, b) \times [0, T]$ is divided into two sub-domains $\Omega_1 := (a, d) \times [0, T]$ and $\Omega_2 := (d, b) \times [0, T]$, which share the interface $I^d := \{(d, s) | s \in [0, T]\}$. When $V(x, t)$ is assumed positive, I^d is the outflow boundary for Ω_1 and the inflow boundary for Ω_2. The algorithm developed here applies to other velocity distributions.

Applying the FRKC scheme (2.16) on $\Omega_{(1,n)} := (a, d) \times [t_{n-1}, t_n]$ yields a numerical solution on (a, d) at time t_n and at the outflow boundary $I^d_n := \{(d, s) | s \in [t_{n-1}, t_n]\}$ of $\Omega_{(1,n)}$, which allows a uniform partition $P^{(O)}_{(1,n)}$ of CO_n intervals on I^d_n from the left. With I^d_n as the inflow boundary of $\Omega_{(2,n)} := (d, b) \times [t_{n-1}, t_n]$, which has a uniform partition $P^{(I)}_{(2,n)}$ of CI_n intervals from the right, one can obtain the numerical solution on $\Omega_{(2,n)}$ by using the FRKC scheme (2.16). These two partitions on I^d_n, accumulated over all the time-steps $[t_{n-1}, t_n]$ of the simulation, define the following two partitions on I^d: (i) partition $P^{(O)}_1 := \cup^N_{n=1} P^{(O)}_{(1,n)}$ determined by the grid points $t^{(O)}_{n,i}$ for $i = 0, 1, \ldots, CO_n - 1$ (as in (2.10)) and $n = 1, 2, \ldots, N$ from the left of I^d, and (ii) the partition $P^{(I)}_2 := \cup^N_{n=1} P^{(I)}_{(2,n)}$ given by the grid points $t^{(I)}_{n,i}$ for $i = 1, \ldots, CI_n - 1$ (as in (2.10)) and $n = 1, \ldots, N$ from the right of I^d. To complete the domain decomposition method, we need only to give an algorithm on how to pass the information across the interface I^d. In the subsequent subsections, we will discuss different ways of passing information between the two sub-domains Ω_1 and Ω_2, so that it is consistent with the case when no interface is present.

To avoid multiple indices for the elements in either partition, we assume that they are ordered in an increasing order by a single index ranging from 0 to the size of the partition considered. The treatment of the interface I^d between Ω_1 and Ω_2 can be carried out by a projection or an interpolation of the piecewise linear solution $U(d^-, t)$ to get $U(d^+, t)$. However, implementing this projection varies depending on the relation that can be assumed between the two partitions considered.

1. Non-conforming Interfacial Matching

The FRKC scheme (2.16) allows the use of both nonuniform and uniform time-steps in the simulation over each sub-domain. Thus, when different time-steps are used on the two sub-domains Ω_1 and Ω_2, the partitions $P^{(O)}_1$ and $P^{(I)}_2$ are not compatible in general and lead to a nonconforming matching on the interface I^d. In this case, an $\mathcal{L}_2$-projection can be used to pass the information from $P^{(O)}_1$ to $P^{(I)}_2$:

$$\int_0^T V(d^+, t) U(d^+, t) w_i(d^+, t)\, dt = \int_0^T V(d^-, t) U(d^-, t) w_i(d^+, t)\, dt, \tag{3.2}$$

for $i = 0, 1, \ldots, \sum^N_{n=1} CI_n$.

We now briefly discuss some implementational issues regarding Eq. (3.2). The left-hand side of Eq. (3.2) resides on the partition $P_2^{(I)}$ and is standard in finite element methods. However, the evaluation of the right-hand side requires extra care. Notice that the trial function $U(d^-, t)$ and the test function $w_i(d^+, t)$ reside on the partitions $P_1^{(O)}$ and $P_2^{(I)}$, respectively, which are not compatible in general, and that the right-hand side of Eq. (3.2) is assembled based on the index of the test function. When one uses a quadrature to evaluate the right-hand side of Eq. (3.2) on the intervals $[t_{i-1}^{(I)}, t_i^{(I)}]$ and $[t_i^{(I)}, t_{i+1}^{(I)}]$ (i.e., the support of w_i) of the partition $P_2^{(I)}$, one may integrate the product of a linear function $w_i(d^+, t)$ with a piecewise-linear function $U(d^-, t)$ on each of these two intervals. Hence, a blind numerical integration of this term may result in loss of mass and oscillations in the numerical solutions [10, 32]. To overcome these difficulties, we further subdivide the intervals $[t_{i-1}^{(I)}, t_i^{(I)}]$ and $[t_i^{(I)}, t_{i+1}^{(I)}]$ into a number of subintervals such that $U(b^-, t)$ is linear on each of these subintervals. Then, we apply a numerical quadrature on each of these subintervals to evaluate the right-hand side of Eq. (3.2). While the nonconforming interfacial matching works for general nonuniform partitions $P_1^{(O)}$ and $P_2^{(I)}$, the process of determining how many subintervals $[t_{j-1}^{(O)}, t_j^{(O)}]$ of $P_1^{(O)}$ are contained in the intervals $[t_{i-1}^{(I)}, t_i^{(I)}]$ and $[t_i^{(I)}, t_{i+1}^{(I)}]$ of $P_2^{(I)}$ may not be very efficient in general. To enhance the efficiency, we consider a conforming matching in the next subsection.

Finally, we notice that when the partitions $P_1^{(O)}$ and $P_2^{(I)}$ on I^d are identical and $V(x, t)$ is continuous across I^d, Eq. (3.2) implies that $U(d^+, t) = U(d^-, t)$. Thus, the nonconforming matching of the interfaces is consistent with the scheme when no interface is present.

2. *Conforming Interfacial Matching*

The partition $P_1^{(O)}$ on the interface I^d (as the outflow boundary of Ω_1) and $P_2^{(I)}$ on I^d (as the inflow boundary of Ω_2) are conforming, if one of them is a subset of the other and each coarse global time-step $[t_{n-1}^c, t_n^c]$ is an integer multiple of a uniform fine global time-step $[t_{m-1}^f, t_m^f]$. The number of fine time intervals in each coarse time interval is allowed to vary with n. In this case, Eq. (3.2) is confined to the coarse time-step $[t_{n-1}^c, t_n^c]$ instead of $[0, T]$:

$$\int_{t_{n-1}^c}^{t_n^c} V(d^+, t)U(d^+, t)w_i(d^+, t)\, dt = \int_{t_{n-1}^c}^{t_n^c} V(d^-, t)U(d^-, t)w_i(d^+, t)\, dt. \qquad (3.3)$$

A conforming matching is fairly flexible in that it allows a nonuniform coarse time-stepping procedure in numerical simulations. Meanwhile, since each coarse global time-step is uniformly partitioned into a number of finer global time-steps, no search algorithm is needed in evaluating the right-hand side as in the case of a nonconforming matching. This leads to ease of implementation and improvement of the CPU time involved in evaluating the right-hand side of Eq. (3.3).

When the velocity field is continuous across the interface I^d and when one projects from a coarse grid to a finer one, then $U(d^+, t)$ is exactly the same as $U(d^-, t)$. In other words, Eq. (3.3) is equivalent to a piecewise linear interpolation, which makes passing information very simple and efficient.

C. Temporal Interfaces

The procedure developed in the last subsection can also be applied to treat temporal interfaces. We solve Eq. (2.1) by a domain decomposition technique in time. Again, we assume that the global space–time domain $\Omega := (a, b) \times [0, T]$ is divided into two sub-domains $\Omega_1 := (a, b) \times [0, T_1]$ and $\Omega_2 := (a, b) \times [T_1, T]$ with $0 < T_1 < T$. In this case, the temporal interface I^{T_1} is given

by $I^{T_1} := \{(x, T_1)|x \in [a, b]\}$. Once the solution is computed on the sub-domain Ω_1 up to the temporal interface I^{T_1}, we need to pass the solution to the other side of the interface I^{T_1}, where it is used as the initial condition for the simulation on the sub-domain Ω_2. As in the case of a spatial interface, if the partition on Ω_1 is not the same as that on Ω_2, the following projection is used to pass the numerical solution across the interface I^{T_1}:

$$\int_a^b U(x, T_1^+) w_i(x, T_1^+)\, dx = \int_a^b U(x, T_1^-) w_i(x, T_1^+)\, dx \tag{3.4}$$

for $i = 0, 1, \ldots, I$, where I is the number of the test functions used on the spatial part of Ω_2. The piecewise linear test functions $w_i(x, T_1^+)$ are given by Eq. (2.12) with T_1 replacing t_n. The detailed treatment is similar to that for the spatial interfaces and is omitted here.

D. Domain Decomposition Method on a General Space–Time Partition

To fully describe the method, we consider a general space–time partition of the domain $\Omega := (a, b) \times [0, T]$ into sub-domains given by

$$a =: d_0^{(m)} < d_1^{(m)} < \cdots < d_{L_m}^{(m)} := b,\ m = 1, 2, \ldots, M,$$

$$0 =: T_0 < T_1 < \cdots < T_M := T. \tag{3.5}$$

These sub-domains are labeled as

$$\Omega_{(l,m)} := (d_{l-1}^{(m)}, d_l^{(m)}) \times [T_{m-1}, T_m],\ \text{for } l = 1, \ldots, L_m,\ \text{and } m = 1, \ldots, M. \tag{3.6}$$

In porous medium fluid flows and seismics, the physical interfaces arise when the medium properties change abruptly, which are typically time-independent. In this case the spatial partition should normally be time-independent, too, i.e., $d_l^{(m)} = d_l (l = 0, \ldots, L_m)$. On the other hand, the steep fronts presented in the numerical solutions are often dynamic, so a practical domain decomposition and local refinement method often needs to be adapted to the moving steep fronts. In this case, the spatial partition should be time-dependent. The partition (3.5) and the method presented in this subsection cover both cases. A general domain decomposition method for solving Eq. (2.1) over domain Ω goes as follows:

Step 1. Using the given initial and inflow boundary conditions in (2.1), one applies the FRKC scheme (2.16) over the space–time sub-domain $\Omega_{(1,1)}$. The solution over this domain defines the solution $U(d_1^-, t)$ for $t \in [0, T_1]$ and $U(x, T_1^-)$ for $x \in [a, d_1]$.
Step 2. Using the procedures developed in Section III.B, one can pass the solution $U(d_1^-, t)$ across the interface between $\Omega_{(1,1)}$ and $\Omega_{(2,1)}$, and obtains $U(d_1^+, t)$ for $t \in [0, T_1]$. Then, one can apply the FRKC scheme (2.16) on $\Omega_{(2,1)}$ to obtain the numerical solution on this sub-domain. Continuing this process, one can obtain the numerical solutions on $\Omega_{(l,1)}$ for $l = 1, 2, \ldots, L_1$. Namely, one can obtain the solution on $[a, b]$ for $t \in [0, T_1]$.
Step 3. In parallel to Step 2, one can apply the scheme (2.16) to solve Eq. (2.1) on the sub-domain $\Omega_{(1,2)}$ unless the sub-domain $\Omega_{(1,2)}$ has a larger spatial interval $[d_0^{(2)}, d_1^{(2)}]$ than the interval $[d_0^{(1)}, d_1^{(1)}]$ for the sub-domain $\Omega_{(1,1)}$. In this case, Step 2 needs to be carried out before this step to find the solution over the sub-domain $\Omega_{(2,1)}$ or even more sub-domains on time interval $[0, T_1]$ until the solution $U(x, T_1^-)$ is known on enough spatial intervals to cover the spatial domain of the sub-domain $\Omega_{(1,2)}$.
Step 4. Repeat Steps 2 and 3 for sub-domains that are adjacent to sub-domains $\Omega_{(2,1)}$ and $\Omega_{(1,2)}$ (and possibly others) until one finally covers all the sub-domains defined in (3.6).

In terms of systolic parallel implementation of this algorithm, it is evident that once the solution is computed on a domain $\Omega_{(l,m)}$, the part of the solution generated at the outflow boundary of that sub-domain can be projected to $\Omega_{(l+1,m)}$, and similarly the solution at the last time-step can be projected to domain $\Omega_{(l,m+1)}$ (with some possible contribution from domains adjacent to $\Omega_{(l,m)}$ when the spatial domain of $\Omega_{(l,m+1)}$ is different from that of $\Omega_{(l,m)}$). Then Steps 2 and 3 can be implemented concurrently to solve problem (2.1) over the two sub-domains $\Omega_{(l+1,m)}$ and $\Omega_{(l,m+1)}$. The global concurrency is more evident in the case when the spatial partition of the domains given in (3.5) is independent of time, whence one starts solving problem (2.1) over $\Omega_{(1,1)}$. Next, Steps 2 and 3 can be implemented concurrently to solve the problem over sub-domains $\Omega_{(2,1)}$ and $\Omega_{(1,2)}$. These procedures can be continued until all the sub-domains are processed.

IV. SPACE–TIME LOCAL REFINEMENT

In many applications modeled by Eq. (2.1), the solutions are relatively smooth outside some very small sharp front regions. These sharp fronts of the solutions need to be resolved accurately in practice [1, 2, 4]. Because of the extremely large size of these problems, an extremely fine global mesh in both space and time is not feasible due to its excessive computational cost. Therefore, one has to use locally refined space–time grids within these small regions to resolve the sharp fronts of the solutions accurately and use coarse space–time grids to obtain a satisfactory approximation outside the sharp front regions with a significantly reduced overall computational cost. The use of local refinement introduces numerical interfaces between the coarse and fine grids, which need special care especially for hyperbolic equations [42, 43].

Because it solves problem (2.1) accurately and treats boundary conditions systematically in a mass conservative manner, the FRKC scheme (2.16) can naturally be combined with a space–time local refinement algorithm. The local refinement algorithm falls in the general framework of the domain decomposition method developed in the previous section, but can be carried out in a more efficient way by selecting the partitions on the interface between two adjacent domains in an appropriate way. We begin by describing ways to improve the treatment of the interfaces resulting from space–time local refinement over that described in the previous section.

A. Interpolation/Projection-Free Space–Time Local Refinement

We consider solving problem (2.1) over the global domain $\Omega := (a, b) \times [0, T]$, which is divided into three sub-domains $\Omega_1 := (a, d_1) \times [0, T]$, $\Omega_2 := (d_1, d_2) \times [0, T]$, and $\Omega_3 := (d_2, b) \times [0, T]$. We assume that the exact solution $u(x, t)$ of the problem has steep fronts within the sub-domain Ω_2 in the middle and is smooth in the other two sub-domains. Therefore, we will use refined spatial and temporal grids on Ω_2 and coarse meshes outside. We now describe how to pass a solution across the interface from a sub-domain with a coarse space–time mesh to a sub-domain with a fine mesh and vise versa. The algorithm actually applies no matter how many sub-domains are used in the local refinement.

Let $I^{d_1} := \{(d_1, s)|s \in [0, T]\}$ be the interface shared by the two sub-domains Ω_1 and Ω_2. With I^{d_1} being the outflow boundary of Ω_1 and the inflow boundary of Ω_2, the FRKC scheme (2.16) has a flexibility in selecting the partition parameters $CO^{(1)}$ and $CI^{(2)}$ (where the superscript indicates the domain) on I^{d_1}. While for any choice of these parameters the algorithm in Section III.B can pass the solution $U(d_1^-, t)$ across the interface I^{d_1} to yield $U(d_1^+, t)$, a more efficient alternative can be obtained by matching the partition $CO^{(1)}$ on I^{d_1} (as the outflow boundary of

Ω_1) with the partition $CI^{(2)}$ on I^{d_1} (as the inflow boundary of Ω_2). With this particular choice of the partition $CO^{(1)}$, the FRKC scheme (2.16) applied to the sub-domain Ω_1 naturally generates the numerical solutions at the nodes on the interface I^{d_1} and $U(d_1+,t) = U(d_1-,t)$ for $t \in [0,T]$. Namely, the numerical solution given on the outflow boundary of Ω_1 directly defines the inflow boundary condition for problem (2.1) on sub-domain Ω_2 without any interpolation or projection involved.

The treatment of the interface $I^{d_2} := \{(d_2,s)|s \in [0,T]\}$ between the sub-domains Ω_2 and Ω_3 is carried out in a similar manner. In this case, the solution moves out of the sub-domain Ω_2 with a fine space–time partition and into the sub-domain Ω_3 with a coarse space–time mesh. In this case, we choose the partition $CI^{(3)}$ on I^{d_2} (as the inflow boundary of Ω_3) to match the partition $CO^{(2)}$ on I^{d_2} (as the outflow boundary of Ω_2). Then we naturally have $U(d_2^+,t) = U(d_2^-,t)$ for $t \in [0,T]$, i.e., we pass the numerical solution across the interface I^{d_2} in a natural manner, without using any interpolation or projection.

B. Adaptive Local Refinement/Nonlinear Approximation

Due to the transient property of the exact solutions of problem (2.1), the regions that contain the sharp fronts of the solutions normally move with time. Hence, one has to use dynamic space–time local refinement to solve problem (2.1) in an efficient and accurate manner. To carry out adaptive local refinement effectively, one has to locate sharp front regions accurately and apply local refinement only where it is needed. In numerical simulations, one can use *a posterior* error indicators/ estimators to locate the front regions.

Extensive research has been conducted in developing various *a posterior* error indicators/ estimators for elliptic and parabolic equations [44–48]. In the context of hyperbolic equations, one also needs to take into account the transient behavior of the solutions. We refer the readers to [49] for an excellent discussion and review on the latest development of adaptive refinement techniques. In our scheme, the adaptive local refinement algorithm can be described as follows:

Step 1. Apply an error indicator or estimator (e.g., those mentioned above) on the current solution (or initial condition, if current time is initial time) of problem (2.1) to locate its steep front regions, which are then identified by their spatial boundaries.
Step 2. For each time-step in the simulation, track these regions forward along the characteristics to predict where the steep front regions will locate at the next time-step. This will provide locally refined grids needed to resolve the steep fronts of the solutions. The contribution from the reaction term, the source term, and the boundary conditions, which could lead to the formation of new steep front regions, can also be identified in this manner.
Step 3. Apply the FRKC scheme (2.16) and the procedure described in Section IV.A on each space–time sub-domain (including those with coarse space–time grids and those with locally refined grids) at the current time interval. Go to Step 1 until one reaches the last time-step.

As one can see, the above adaptive space–time local refinement algorithm is accomplished based on a thorough understanding of all terms in the governing equation, and their influence on the solutions as time evolves. Thus, one expects that the algorithm should perform well. In Section VI, we perform experiments to observe its numerical performance.

Although we focus on a linear advection–reaction Eq. (2.1) in this article, we keep in mind that in applications Eq. (2.1) is often coupled to other equations and forms a nonlinear system of advection-dominated partial differential equations. Consequently, the regularity of the solutions is significantly reduced and the related theoretical results are meager. The reduced regularity in turn limits the convergence rates of numerical simulations. In the rest of this subsection,

we explore some possible theoretical benefits of the use of adaptive refinements, by looking at the numerical methods with or without adaptive local grid refinement from the viewpoint of approximation theory. The numerical methods without local grid refinement correspond to *linear* methods in approximation theory, where the numerical solutions are chosen from a fixed linear space (e.g., piecewise linear functions on a fixed partition), usually using a linear projection operator. The partition of domains is independent of the solutions being approximated. Linear methods are relatively simple to study and implement, but they may not be very efficient when the exact solutions being approximated have strongly local behavior (e.g., steep fronts). Furthermore, linear methods require higher regularity of the exact solutions being approximated.

Let $\mathcal{S}_{I,r}(a,b)$ be the space of all continuous functions that are piecewise polynomials of degree $r-1$ on the spatial partition given by (2.2). Set $\mathcal{E}_I(f,S_{I,r})$ to be the best approximation error given by

$$\mathcal{E}_I(f,\mathcal{S}_{I,r}) := \inf_{g\in\mathcal{S}_{I,r}(a,b)} \|f-g\|_{\mathcal{L}_p(a,b)}. \tag{4.1}$$

Here $\mathcal{L}_p(a,b)$ is the space of p^{th} power, Lebesgue integrable functions on (a,b) with norm (quasi-norms if $0<p<1$)

$$\|f\|_{\mathcal{L}_p(a,b)} = \left(\int_a^b |f(x)|^p dx\right)^{1/p}. \tag{4.2}$$

It has been proven [50, 51] that for a linear method to approximate a function f from the space of piecewise polynomials $\mathcal{S}_{I,r}(a,b)$ with a convergence rate of $\mathcal{O}(I^{-\alpha})$ in the $\mathcal{L}^p(a,b)$ norm is equivalent to f having α order of smoothness (or α order of derivatives, if α is an integer) in $\mathcal{L}_p(a,b)$ (i.e., f is in the Sobolev space $\mathcal{W}^\alpha(\mathcal{L}_p(a,b))$). Therefore, a higher convergence rate requires higher regularity of the function f being approximated. In particular, there is no linear method for one-dimensional nonlinear hyperbolic equations (conservation laws) that have higher than first-order convergence rate in the $\mathcal{L}^1(a,b)$ norm. Otherwise, the above result concludes that the exact solution should be in $\mathcal{W}^1_1(a,b)$, which implies that the exact solution should be absolutely continuous on (a,b) by the Sobolev embedding theorem. However, it is well known that the exact solutions of conservation laws can develop shock discontinuities in finite time no matter how smooth the initial condition may be. This is the theoretical justification that there are no higher-order *linear* methods for nonlinear hyperbolic equations.

On the other hand, in a *nonlinear* method the spatial partition in (2.2) is allowed to depend on the function being approximated. Let $\mathcal{X}_{I,r}(a,b)$ be the space of all continuous functions that are piecewise polynomials (of degree $r-1$) with I pieces on $[a,b]$. Since different functions in $\mathcal{X}_{I,r}(a,b)$ may have different break points, the sum of two functions in $\mathcal{X}_{I,r}(a,b)$ may not necessarily belong to $\mathcal{X}_{I,r}(a,b)$ (but does belong to $\mathcal{X}_{2I,r}(a,b)$). The space–time local grid refinement algorithm in this section can be viewed as a practical implementation of a nonlinear method. Let $\mathcal{F}_I(f,\mathcal{X}_{I,r})$ be the best approximation error from the nonlinear manifold $\mathcal{X}_{I,r}(a,b)$:

$$\mathcal{F}_I(f,\mathcal{X}_{I,r}) := \inf_{g\in\mathcal{X}_{I,r}(a,b)} \|f-g\|_{\mathcal{L}_p(a,b)}. \tag{4.3}$$

In comparing the rigorous treatments of linear and nonlinear approximation, DeVore [50] and Petrushev [51] go further and prove that a nonlinear method has a convergence rate of $\mathcal{O}(I^{-\alpha})$ in the $\mathcal{L}_p(a,b)$ norm if and only if f has α order of smoothness (or α order derivatives, if α is an integer) in $\mathcal{L}_\tau(a,b)$ with $\tau = p/(1+\alpha p)$, more precisely, f belongs to the Besov space $\mathcal{B}^\alpha_q(\mathcal{L}_\tau(a,b))$, where the parameter $q(0<q\le\infty)$ is a finer scaling of smoothness than Sobolev spaces provide (see [52, 53] for details). Since $0<\tau<1$ is significantly smaller than p

(as long as one allows nonlocally convex spaces $\mathcal{L}_\tau(a,b)$), the regularity requirement of f for a nonlinear method is much weaker than that for a linear method. Furthermore, DeVore and Lucier [54, 55] have proven that the Besov spaces are regularity spaces for one-dimensional conservation laws. Namely, if the initial condition u_0 is in a Besov space (which is true for almost all the realistic applications) the solution will remain in the same space for all later time. The importance of this result is that it shows that solutions to conservation laws retain high orders of regularity when measured in the correct way. The combination of this result with that for nonlinear approximation mentioned above predicts that higher-order nonlinear methods to conservation laws can be developed with $\mathcal{O}(I^{-\alpha})$ accuracy in the $\mathcal{L}_p(a,b)$ norm for any $\alpha > 0$ (with $r > \alpha$ in $\mathcal{X}_{I,r}(a,b)$). These observations theoretically justify the use of adaptive local grid refinement, because it can significantly increase efficiency and reduce the regularity requirement on the exact solutions for hyperbolic equations, since they typically have strongly local behavior. Numerical experiments are presented in Section VI to demonstrate the strong potential of the adaptive local refinement method developed in this article.

C. Local Refinement with Patches/Nested Grids

In the previous subsection, we presented an adaptive space–time local grid refinement algorithm and justified it from a theoretical viewpoint. We now discuss some implementational issues for adaptive local grid refinement. A blind application of the local refinement algorithm can significantly enhance the efficiency of the numerical simulation to problem (2.1), but it could introduce some nonconforming grids at spatial or temporal interfaces. In that case, one has to use an $\mathcal{L}_2$ projection to pass information across interfaces as described in Section III.B. This could also require a complex coarse–fine grid, leading to complicated data structures.

To address these potential difficulties we use nested grids and patches in the local grid refinement and modify the algorithm in Section IV.B as follows: We first define an underlying coarse space–time partition given by (2.2) on the global domain $\Omega := (a,b) \times [0,T]$, with the spatial partition being independent of time. At each time-step, we define a patch to be the cluster of macro elements that have a nonempty overlap with a steep front region. We then use a refined space–time grid in each of the macro elements in the patch. This way, we obtain a space–time partition with nested space–time grids. Although this implementation enlarges the refined region slightly (up to the size of two macro elements in each spatial coordinate direction) than the algorithm presented in Section IV.B, it uses a static coarse space–time partition and dynamic local refinement within certain patches. Therefore, globally and locally one solves problems with the same partition and data structure, so it greatly simplifies the implementation of the adaptive local grid refinement algorithm especially for multidimensional problems.

Finally, the above adaptive local grid refinement algorithm with patches/nested grids is a two level (coarse–fine grid) method. This algorithm can be applied repeatedly, yielding multilevel local refinement with nested grids.

V. DESCRIPTION OF SOME GRID REFINEMENT SCHEMES

In this section, we describe some existing finite difference local refinement schemes for problem (2.1), which are widely used in applications [27]. In the next section, these schemes are compared with the local refinement method developed in this article, to observe their numerical performance. For simplicity we present these schemes for problem (2.1) with a constant velocity field, but with no reaction or source terms involved.

These schemes typically use a conventional finite difference scheme (e.g., Lax–Wendroff or leapfrog) to solve problem (2.1) on a uniform coarse grid on the global domain $\Omega := (a, b)\times[0, T]$:

$$x_i := a + i\Delta x^c,\ \Delta x^c := \frac{b-a}{I},$$

$$t_n := n\Delta t^c,\ \Delta t^c := \frac{T}{N}. \tag{5.1}$$

For definiteness, we take the leapfrog scheme as an example. Applying the leapfrog scheme, we advance the solution from the previous time-steps t_{n-1} and t_n to t_{n+1}, as follows:

$$U(x_i, t_{n+1}) = U(x_i, t_{n-1}) - \lambda[U(x_{i+1},\ t_n) - U(x_{i-1}, t_n)], \tag{5.2}$$

where $\lambda = V\Delta t^c/\Delta x^c$ is the Courant number.

Then one uses a locally refined space–time grid that is nested in the coarse grid (5.1). We define the refined grid size and time-step to be

$$\Delta x^f := \frac{\Delta x^c}{M_x},\ \Delta t^f := \frac{\Delta t^c}{M_t}, \tag{5.3}$$

where the positive integers M_x and M_t are the space and time refinement ratio, respectively.

Notice that the first node x_0^f in the fine mesh is also a coarse grid node $x_{\bar{i}}$ for some $\bar{i}$ (since the grids are nested), which could be either a cell center or a grid point, depending on the type of scheme used. To apply the leapfrog scheme (5.2) on the refined grid, one has to provide the values of the solution at x_0^f for all the fine time-steps, which is an interface due to the use of locally refined spatial grids and temporal steps. Different interface schemes were developed previously to treat these interfaces, including the Clark–Farley scheme, a characteristic interface scheme, and two integration schemes: a node integration and a cell-centered integration scheme. Let

$$x_{-k}^f := x_0^f - k\Delta x^c,\ k = 0, 1, 2, \ldots,$$

$$x_k^f := x_0^f + k\Delta x^f,\ k = 0, 1, 2, \ldots,$$

$$t_{n,m} := n\Delta t^c + m\Delta t^f,\ m = 0, 1, \ldots, M_t \tag{5.4}$$

be the refined spatial grids and time-steps. These interface schemes may require the values of U at x_{-k}^f for different fine time-steps $t_{n,m}$. With the values of U at the coarse space–time grid being known, the values of U at fine time-steps are approximated by a second-order Taylor polynomial in time, with the time derivatives being replaced by centered differences as follows:

$$\begin{aligned} U(x_{-k}^f, t_{n,m}) :&= U(x_{-k}^f, t_n) + \frac{[U(x_{-k}^f, t_{n+1}) - U(x_{-k}^f, t_{n-1})]}{2\Delta t^c}\,(m\Delta t^f) \\ &+ \frac{[U(x_{-k}^f, t_{n+1}) - 2U(x_{-k}^f, t_n) + U(x_{-k}^f, t_{n-1})]}{2(\Delta t^c)^2}\,(m\Delta t^f)^2. \end{aligned} \tag{5.5}$$

The solution procedure goes as follows: First one applies the leapfrog scheme (5.2) on the coarse grid to advance the solution from the previous time levels t_{n-1} and t_n to t_{n+1}. Second, with the values of $U(x_{-1}^f, t_{n,m})$ and $U(x_{-2}^f, t_{n,m})$ evaluated by (5.5), the solution $U(x_0^f, t_{n,m})$ at the interface x_0^f at fine time-steps can be obtained by the interface schemes described below. The treatment at the other end of the fine grid is similar. Then these values are used as the artificial

boundary conditions for the leapfrog scheme (5.2) applied on the refined grid for all the fine time-steps within the coarse time level. Below we present the different interface schemes.

1. Clark–Farley Scheme

The Clark–Farley scheme solves a least square fit with quadratic collocation to generate the solution at the interface x_0^f for fine time-steps as follows:

$$\begin{aligned} U(x_0^f, t_{n,m+1}) \;:=\; & \left[\alpha + \frac{1}{8}\left(\frac{1}{M_t^2} - 1\right)\right] U(x_{-2}^f, t_{n,m}) \\ & + \left[\left(\frac{3}{2} - \frac{1}{2M_t}\right)\left(\frac{1}{2} + \frac{1}{2M_t}\right) - 2\alpha\right] U(x_{-1}^f, t_{n,m}) \\ & + \left[\frac{1}{2}\left(\frac{1}{2} - \frac{1}{2M_t}\right)\left(\frac{3}{2} - \frac{1}{2M_t}\right) + \alpha\right] U(x_0^f, t_{n,m}), \end{aligned} \tag{5.6}$$

where the value $\alpha = [\frac{1}{M_t^2} - 1]/24$ is chosen to ensure local mass conservation [56]. In the event that α is chosen to be zero, Eq. (5.6) is reduced to the Lagrange interpolation.

2. Characteristic Interface Scheme

In the characteristic interface scheme [43] the values of U at the interface x_0^f are obtained by backtracking the characteristic at a distance of $|V|\Delta t^f$ and then using a linear interpolation in space as follows:

$$U(x_0^f, t_{n,m+1}) := \begin{cases} \frac{-2\lambda}{M_t - 1} U(x_{-1}^f, t_{n,m}) + \frac{M_t - 1 + 2\lambda}{M_t - 1} U(x_0^f, t_{n,m}), & \text{if } V > 0, \\ (1+\lambda) U(x_0^f, t_{n,m}) - \lambda U(x_1^f, t_{n,m}), & \text{if } V < 0. \end{cases} \tag{5.7}$$

3. Node-Based Interface Integration Scheme

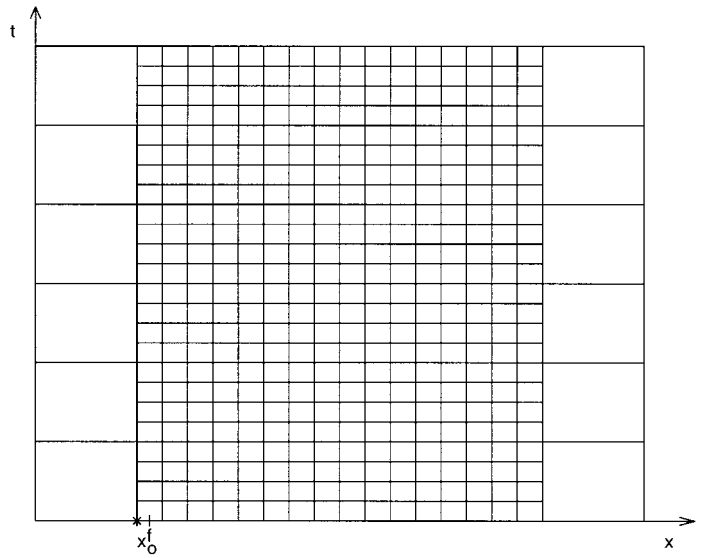

FIG. 1. Partition of the domain in Experiment 6.1

The node integration scheme is obtained by integrating Eq. (2.1) over the element $[x_0^f - \frac{\Delta x^c}{2}, x_0^f + \frac{\Delta x^f}{2}]$ as follows [27]:

$$U(x_0^f, t_{n,m+1}) := U(x_0^f, t_{n,m-1}) - V\frac{2\Delta t^f}{\Delta x^c + \Delta x^f}\,[U(x_1^f, t_{n,m}) - U(x_{-1}^f, t_{n,m})]. \quad (5.8)$$

4. Cell-Centered Interface Integration Scheme

The cell-centered integration scheme is obtained by integrating Eq. (2.1) over the element $[x_{-1}^f, x_1^f]$ as follows [27]:

$$U(x_0^f, t_{n,m+1}) := U(x_0^f, t_{n,m-1}) - V\frac{4\Delta t^f}{\Delta x^c + \Delta x^f}\,[U(x_1^f, t_{n,m}) - U(x_{-1}^f, t_{n,m})]. \quad (5.9)$$

In this case, x_0^f is a cell center.

VI. NUMERICAL EXPERIMENTS

In this section, we carry out various numerical experiments to test the performance of the domain decomposition and space–time local refinement method developed in this article. Three types of experiments are included: (1) a comparison of the method with the local refinement methods reviewed in the previous section; (2) a comparison of the domain decomposition method for problem (2.1) with discontinuous coefficients with the results reported in [57]; and finally (3) a comparison of the adaptive local refinement method with a globally refined grid.

A. Space–Time Local Refinement on a Static Domain Decomposition

The numerical experiments for the Eulerian finite difference local refinement method described in the previous section (including the Clark–Farley interface scheme, a characteristic interface scheme, and two integration interface schemes) were reported in [27]. In this section, we test the same example to compare the performance of the local refinement method developed in this article with those.

In this example, the velocity field is $V(x,t) = -1$ and no reaction or source term is involved (i.e., $K(x,t) = 0$ and $f(x,t) = 0$). The initial condition is a Gaussian distribution given by

$$u_0(x) := \exp\left[-\frac{(x - x_c)^2}{2\sigma^2}\right], \quad (6.1)$$

where x_c and σ are the center and standard deviation of the pulse. The corresponding analytic solution of Eq. (2.1) is given by

$$u(x,t) = \exp\left[-\frac{(x + t - x_c)^2}{2\sigma^2}\right]. \quad (6.2)$$

The data for this simulation are illustrated in Fig. 1 and are as follows [27]: The global spatial domain is $(a,b) = (0, 250)$ with a uniform coarse grid of $\Delta x^c = 1.0$. The refined region is $[d_1, d_2] = [75, 105]$ with a spatial refinement ratio of $M_x = 10$ in (5.3) (so $\Delta x^f = 0.1$). To insure the CFL-stability of the finite difference local refinement methods, a coarse time-step of $\Delta t^c = 0.9$ was chosen [27]. The global time interval is $[0, T] = [0, 51.3]$, which is the time after 57 time-steps. In the refined region [75, 105], a fine time-step of $\Delta t^f = 0.09$ was used. In the simulation, a homogeneous Dirichlet boundary condition is imposed at the inflow boundary of

the global domain. Initially, the Gaussian pulse is centered at $x_c = 90$ with a standard deviation of $\sigma = 1$. These values ensure that the Gaussian pulse is initially contained in the refined region. The numerical experiments were reported in [27] for different time-steps. In Figs. 2 and 3 (a)–(d), we cite those results at $t = 18.9$ (21 time-steps), which corresponds to the time when the solution moved out of the refined region, and the final time $t = 51.3$ (57 time-steps).

In these experiments, we are able to use a much larger time-step with our method to generate accurate solutions, but otherwise retain the same data parameters as the other methods. The first of the experiment run is to compare with their solutions at time $t = 18.9$ (21 time-steps), which is the time when the solution has moved out of the fine region. Any effects due to the interface treatment will be apparent. We use two time-steps, which gives a coarse time-step of $\Delta t^c_{FRKC} = 9.45$. In the refined region [75, 105] we still use a refinement of ratio $M_t = 10$ as in [27], to observe the performance of the method across an interface with a considerable mesh difference. This yields a fine time step of $\Delta t^f_{FRKC} = 0.945$, instead of $\Delta t^f = 0.09$ for the finite difference local refinement method. The numerical solution is plotted against the analytical one in Fig. 2(e). The second run is to compare with their solutions at the final time $t = 51.3$ (57 time-steps). We use four time-steps, yielding a coarse time step of $\Delta t^c_{FRKC} = 12.825$. On the refined region [75, 105] the refinement ratio M_t is still chosen to be 10 as in [27], so the fine time-step $\Delta t^f_{FRKC} = 1.2825$. The numerical solution is plotted against the analytical one in Fig. 3(e).

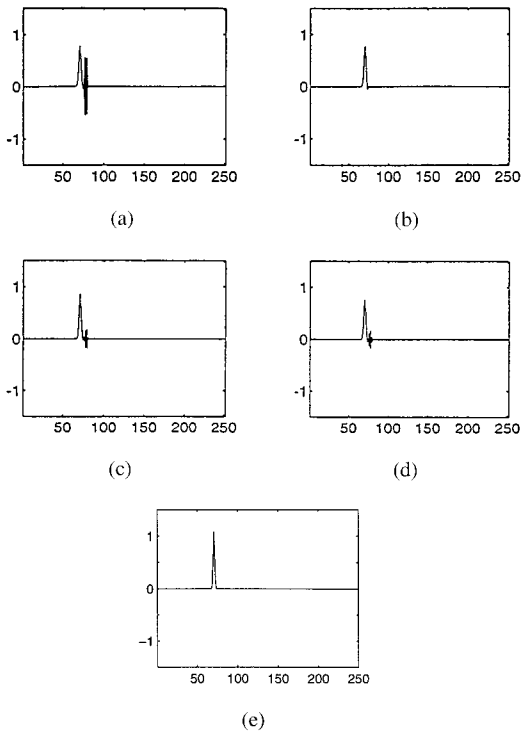

FIG. 2. Comparative plots of the schemes at time 18.9. (a) Clark-Farley Method; (b) Characteristic Method; (c) Nodal Integration Method; (d) Cell Integration Method; (e) FRKC − Local Refinement

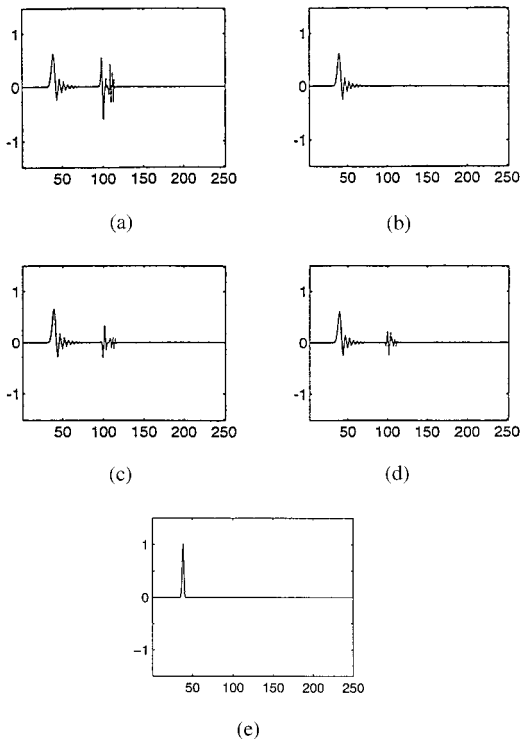

FIG. 3. Comparative plots of the schemes at time 51.3. (a) Clark-Farley Method; (b) Characteristic Method; (c) Nodal Integration Method; (d) Cell Integration Method; (e) FRKC − Local Refinement

These numerical comparisons show that the solutions with the method in this article pass an interface smoothly without any numerical artifacts, even though a much larger time-step has been used. In contrast, the finite difference local refinement method has started to develop trailing disturbances in the form of oscillations. The Clark–Farley solution has the greatest such reflections among the four interface schemes, while the solution with the characteristic interface scheme has the least. As the time of the simulation evolves, the trailing disturbances in the Clark–Farley and the two integration schemes are reflected toward the fine region. Figure 3 is a plot of the solutions of the five methods at time $t = 51.3$. At this time, the trailing oscillations in the solutions of the Clark–Farley and the two integration schemes have reached the fine region boundary where they are amplified. The solution with the characteristic interface method continues to have large trailing oscillations. The solution with the method developed in this article is very accurate without any numerical artifacts.

B. Domain Decomposition for Problem (2.1) with Discontinuous Coefficients

In this experiment, we use the domain decomposition method developed in Section III.D to solve problem (2.1) with discontinuous coefficients. We choose the same test problem used in [57]: a space domain $(a, b) = (-1, 1), 0 \le t \le 0.56$, and a velocity field $V(x, t)$ defined by

$$V(x,t) = \begin{cases} 1, \text{if } x \in (-1,0), \\ 2, \text{if } x \in (0,1). \end{cases} \tag{6.3}$$

Homogeneous reaction and source terms are used (i.e., $K(x,t) = f(x,t) = 0$). Also, a homogeneous inflow Dirichlet boundary condition is specified at $x = -1$. The initial condition is set to

$$u_o(x) = \begin{cases} \exp\left(\frac{-1}{10000(x+0.3)^2(x+0.1)^2}\right), & \text{if } x \in (-0.3, -0.1), \\ 0, & \text{otherwise.} \end{cases} \tag{6.4}$$

Since $V(x,t)$ has a jump discontinuity at $x = 0$, the interface condition (3.1) is reduced to the following condition:

$$u(0^-, t) = 2u(0^+, t), t \in [0, T], \tag{6.5}$$

which leads to the analytical solution

$$u(x,t) = \begin{cases} u_o(x-t), & \text{if } x \in (-1, 0), \\ \frac{1}{2}u_o\left(\frac{x-2t}{2}\right), & \text{if } x \in (0, 1). \end{cases} \tag{6.6}$$

A natural domain decomposition is given by the physical interface at $x = 0$, i.e., $\Omega_1 := (-1, 0) \times [0, T]$ and $\Omega_2 := (0, 1) \times [0, T]$. This problem was simulated using some finite difference interface schemes [57] with a spatial grid size of $\Delta x^{(1)} = 1/200$ for Ω_1 and with $\Delta x^{(2)} = 1/400$ for Ω_2. Time-steps of $\Delta t = 0.004$ and $\Delta t = 0.002$ were chosen for the simulation with numerical solutions plotted at times $t = 0.2$, 0.32, and 0.56.

In our numerical simulation we use a uniform space grid of $\Delta x = 1/120$ and have chosen a time-step of $\Delta t = 0.04$, which is the largest time-step that can be used as a common divisor of $t = 0.2, t = 0.32$, and $t = 0.56$ for the bases of comparison to the results in [57]. The corresponding numerical solutions are plotted against the analytical one in Fig. 4 for $t = 0.2, 0.32$, and 0.56. The time $t = 0.2$ corresponds to the time when the solution moves across the interface. The right half of the pulse has only half the height of the left half, due to the effect of the interface condition (6.5). At the same time the right half of the solution has twice the width of the solution on the left half, because of the change of the velocity field and mass conservation. The solutions at time $t = 0.32$ and $t = 0.56$ move completely into Ω_2.

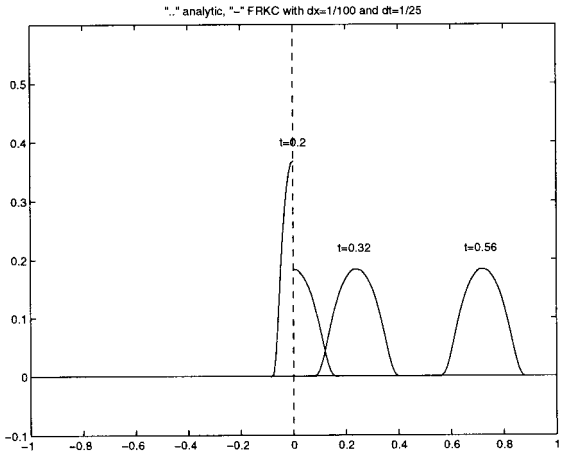

FIG. 4. Results for Experiment 6.2 at time $t = 0.2$, 0.32, and 0.56

Even though we use a much larger time-step ($\Delta t = 0.04$) and spatial grid ($\Delta x = 1/120$) with our method than those in [57] ($\Delta x^{(1)} = 1/200$ on Ω_1, $\Delta x^{(2)} = 1/400$ on Ω_2, and $\Delta t = 0.004$ or 0.002), our method generates comparable solutions.

C. Adaptive Space–Time Local Grid Refinement

In our previous work [36], we developed the FRKC scheme for problem (2.1) with continuous coefficients and conducted extensive numerical experiments to compare the FRKC scheme with many well received and widely used methods. The compared schemes include the Galerkin and Petrov–Galerkin finite element method [16–18], the streamline diffusion method [20, 22], the continuous and discontinuous Galerkin method [19, 21], the monotonic upstream-centered scheme for conservation laws (MUSCL), and the essentially nonoscillatory (ENO) scheme [23–26]. The results show that the FRKC method outperforms these methods in terms of both accuracy and efficiency in solving problem (2.1).

In this subsection, we show that the adaptive local grid refinement method, which is developed in this article based on the FRKC scheme, can further improve the efficiency of the numerical simulation by fully utilizing the transient and strongly local behavior of the solution. In this experiment, a variable velocity field $V(x,t) := V_0 + V_1 x$ is used, and a negative reaction

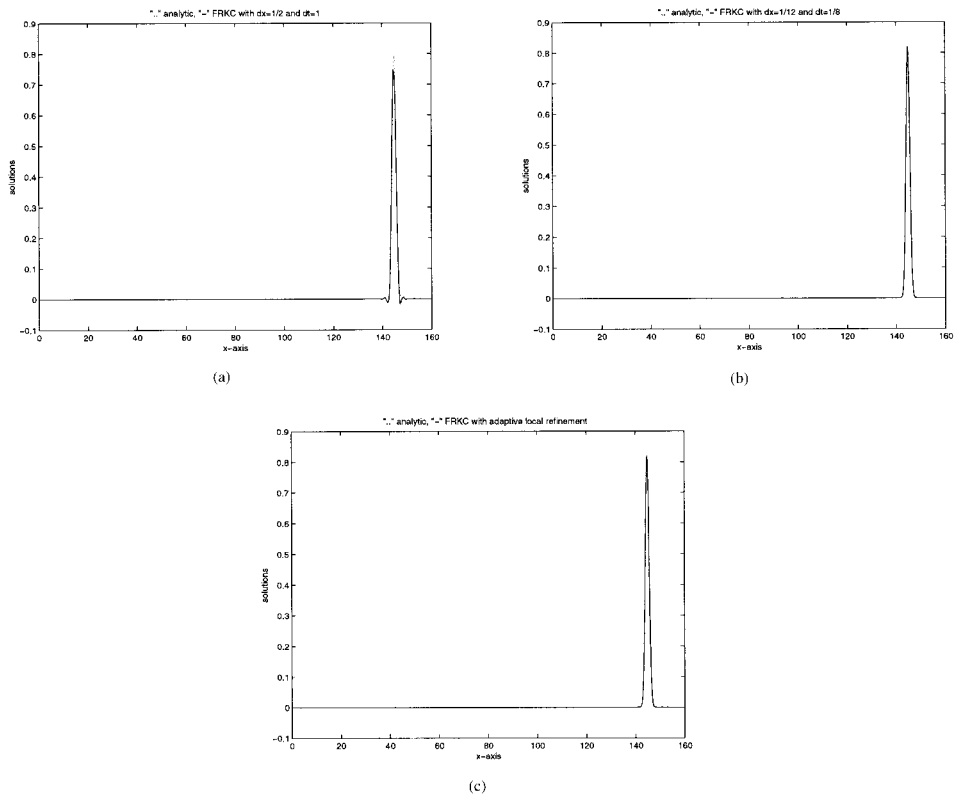

FIG. 5. Results for Experiment 6.3. (a) uniformly coarse grid, $\Delta x^c = \frac{1}{2}$ and $\Delta t^c = 1$; (b) uniformly fine grid, $\Delta x^f = \frac{1}{12}$ and $\Delta t^f = \frac{1}{8}$; (c) adaptive local refinement, $\Delta x^c = \frac{1}{2}, \Delta t^c = 1, \Delta x^f = \frac{1}{12}$, and $\Delta t^f = \frac{1}{8}$.

$K(x,t) := -0.75V_1$ is imposed to prevent the exact solution from decaying too rapidly. We again choose a homogeneous source term. For the initial condition $u_o(x)$ given by (6.1), the analytic solution is given by

$$u(x,t) = \exp\left[-\frac{1}{2\sigma^2}\left[\left(x+\frac{V_0}{V_1}\right)\exp(-V_1 t) - \frac{V_0}{V_1} - x_c\right]^2\right]\exp(-0.25V_1 t). \quad (6.7)$$

The global spatial domain and time interval are $(a,b) = (0,160)$ and $[0,T] = [0,16]$, respectively. In the expression of $V(x,t)$ and $K(x,t)$, $V_0 = 5$ and $V_1 = 0.05$ are chosen, yielding a velocity field of $V(x,t) = 5 + 0.05x$ and $K(x,t) = -0.0375$. In the initial configuration (6.1), the standard deviation $\sigma = 0.35$ and the centered deviation $x_c = 10$ are chosen. A homogeneous Dirichlet boundary condition is specified at the inflow boundary $x = 0$.

The *first test run* applies the FRKC scheme (2.16) using a *uniform coarse* grid of $\Delta x^c = 1/2$ and a coarse time-step of $\Delta t^c = 1$. The Courant number of this simulation is 26. Figure 5(a) is a plot of the numerical and analytic solutions at the final time $T = 16$. Table I contains the $\mathcal{L}_2$ and the $\mathcal{L}_1$ norms of the truncation error of the solution as well as the CPU time used, which is measured on a SUN Sparc LX workstation. Figure 5(a) shows that the numerical solution has some slight wiggles around the bottom of the solution and some error near the top. Because the standard deviation $\sigma = 0.35$, the coarse grid $\Delta x^c = 1/2$ is not fine enough to resolve the steep front of the solution accurately. This introduces the numerical artifacts, which in turn show the need for refinement.

The *second test run* uses a *uniformly refined* spatial grid of $\Delta x^f = \frac{1}{12}$ and time-step of $\Delta t^f = \frac{1}{8}$, i.e., with space and time refinement ratios of $M_x = 6$ and $M_t = 8$. Figure 5(b) shows the plot of the numerical solution at the final time vs. the analytic one, while the error information and the CPU time used are presented in Table I. The accuracy of the numerical solution is significantly improved over the coarse grid solution (e.g., the $\mathcal{L}_2$ error is reduced from 1.082×10^{-1} to 1.826×10^{-3}). However, the CPU time used increased from 2.1 s to 81.9 s, approximately 4,000% increase. This is understandable, since the size of the problem has been increased 48 times. Although this CPU time might at first seem high, we have shown in [36] that significantly higher (one to two orders of magnitude) CPU times than this must be consumed for many other well-regarded methods to generate comparably accurate solutions.

To improve the efficiency of the numerical simulation while maintaining its accuracy, the *third test run* uses the *adaptive local grid refinement* method developed in Section IV.B. To begin, we use a uniformly coarse grid of $\Delta x^c = \frac{1}{2}$ and time-step of $\Delta t^c = 1$, as in the first test run. The same refined grid size of $\Delta x^f = \frac{1}{12}$ and time-step of $\Delta t^f = \frac{1}{8}$, as in the second test run, is used locally in the steep front region. In the numerical simulation we used the second-order Runge–Kutta quadrature (2.8) to track the spatial boundaries of the steep front regions forward from the current coarse time-step t^c_{n-1} to the next coarse time-step t^c_n to predict the future location of the steep front region. Next we use a nested locally refined grid of $\Delta x^f = \frac{1}{12}$ and time-step of $\Delta t^f = \frac{1}{8}$ to refine the space–time sub-domain that contains the steep front of the solution during the time period $[t^c_{n-1}, t^c_n]$. Figure 6 contains the space-time partition information used in this

TABLE I. Comparison of the $\mathcal{L}_2$ and $\mathcal{L}_1$ norms of the error and the CPU time (s).

Δx	Δt	$\mathcal{L}_2$ error	$\mathcal{L}_1$ error	CPU
$\Delta x^c = \frac{1}{2}$	$\Delta t^c = 1$	1.081939×10^{-1}	2.044497×10^{-1}	2.1
$\Delta x^f = \frac{1}{12}$	$\Delta t^f = \frac{1}{8}$	1.826050×10^{-3}	2.961247×10^{-3}	81.9
$\Delta x^c = \frac{1}{2}, \Delta x^f = \frac{1}{12}$	$\Delta t^c = 1, \Delta t_f = \frac{1}{8}$	1.831953×10^{-3}	3.083958×10^{-3}	14.5

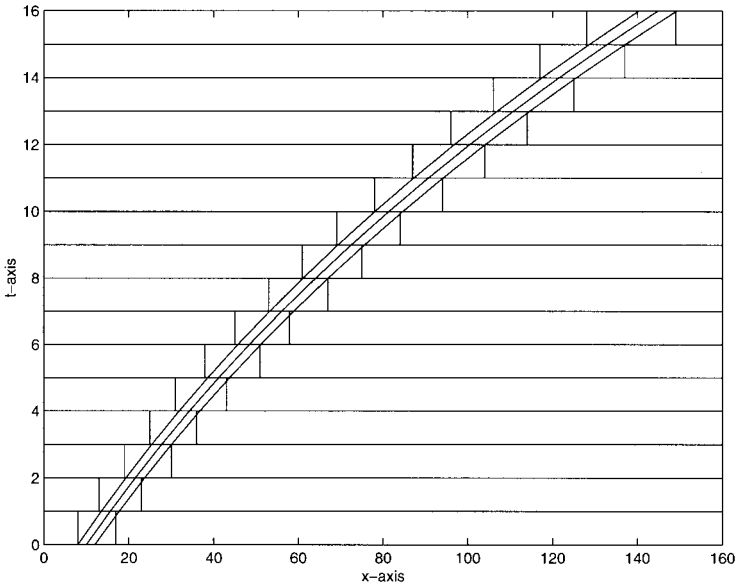

FIG. 6. Moving grid for adaptive local refinement (Experiment 6.3)

simulation. The numerical solution obtained with the adaptive local grid refinement is plotted in Fig. 5(c). The error and CPU time information appears in Table I. Observe that the adaptive local refinement method generates a numerical solution with the same accuracy as the uniformly fine grid solution, but uses a CPU time of only 14.5 s. This is about 18% of the CPU time used by the uniform grid refinement and shows the strong potential of the adaptive local grid refinement method developed in this article.

VII. SUMMARY

In this article, we develop a characteristic-based domain decomposition and space–time local refinement method for first-order linear hyperbolic equations (possibly including various physical and numerical interfaces) based on a forward Runge–Kutta characteristic (FRKC) scheme we have developed previously [36]. In the context of first-order linear hyperbolic equations with continuous coefficients, it was shown [36] that the FRKC scheme outperforms many well received and widely used methods, including the Galerkin and Petrov–Galerkin finite element methods [16–18], the streamline diffusion method [20, 22], the continuous and discontinuous Galerkin methods [19, 21], the monotonic upstream-centered scheme for conservation laws (MUSCL), and the essentially nonoscillatory (ENO) scheme [23–26]. The domain decomposition and local refinement method developed in this article is naturally implemented in the FRKC framework and has the following advantages:

1. It retains the numerical advantages of the FRKC scheme and generates accurate numerical solutions even if large time-steps are used.
2. The method fully utilizes the transient and strongly local behavior of the solutions to further improve the efficiency of the FRKC scheme while maintaining its accuracy.

3. It treats the physical interfaces and various numerical interfaces with a universal scheme. In contrast, many previous finite difference local refinement methods require additional interface schemes to treat the interfaces between different sub-domains or coarse–fine sub-domains, but still do not provide a comparable accuracy.

The authors would like to extend their appreciation to the referees for their useful comments and for additional references which have been incorporated into the final manuscript.

References

1. R. E. Ewing, Ed., ''The mathematics of reservoir simulation,'' in Research Frontiers in Applied Mathematics, Vol. 1, SIAM, Philadelphia, 1984.
2. B. A. Finlayson, Numerical Methods for Problems with Moving Fronts, Ravenna Park Publishing, Seattle, 1992.
3. C. E. Leith, ''Numerical simulation of the earth's atmosphere,'' Methods in Computational Physics, 4, 1–28 (1965).
4. R. J. LeVeque, Numerical Methods for Conservation Laws, Birkhäser Verlag, Basel, 1992.
5. P. K. Smolarkiewicz, ''The multidimensional crowley advection scheme,'' Monthly Weather Review 110, 1968–1983 (1982).
6. R. Glowinski, J. Périaux, Z–C. Shi, and O. Widlund, Eds., ''Domain decomposition methods in science and engineering,'' Proceedings of Eighth International Conference on Domain Decomposition Methods, John Wiley, New York, 1997.
7. D. Keyes and J. Xu, Eds., ''Domain decomposition methods in scientific and engineering computing,'' Proceedings of the Seventh International Conference on Domain Decomposition Methods, Contemporary Mathematics, Vol. 180, American Mathematical Society, Providence, Rhode Island, 1994.
8. B. F. Smith, P. E. Bjørstad, and W. D. Gropp, Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, 1996.
9. P. J. Binning and M. A. Celia, ''A finite volume Eulerian–Lagrangian localized adjoint method for solution of the contaminant transport equations in two-dimensional multi-phase flow systems,'' Water Resource Research 32, 103–114 (1996).
10. M. A. Celia, T. F. Russell, I. Herrera, and R. E. Ewing, ''An Eulerian–Lagrangian localized adjoint method for the advection–diffusion equation,'' Advances in Water Resources 13, 187–206 (1990).
11. R. E. Ewing and H. Wang, ''An optimal-order error estimate to Eulerian–Lagrangian localized adjoint method for variable-coefficient advection–reaction problems,'' SIAM Num. Anal. 33, 318–348 (1996).
12. T. F. Russell and R. V. Trujillo, ''Eulerian–Lagrangian localized adjoint methods with variable coefficients in multiple dimensions,'' in Computational Methods in Surface Hydrology, Gambolati et al. Eds., Springer–Verlag, Berlin, 1990, pp. 357–363.
13. H. Wang, H. K. Dahle, R. E. Ewing, M. S. Espedal, R. C. Sharpley, and S. Man, ''An Eulerian–Lagrangian localized adjoint method for advection-dispersion equations in two dimensions and its comparison to other schemes,'' SIAM J. Sci. Comp., to appear.
14. R. Rannacher and G. Zhou, ''Analysis of a domain-splitting method for nonstationary convection-diffusion problems,'' East-West J. Num. Math. 2, 151–172 (1994).
15. X. Tai, T. Johansen, H. Dahle, and M. Espedal, ''A characteristic domain splitting method,'' in Domain Decomposition Methods in Science and Engineering, Proceedings of Eighth International Conference on Domain Decomposition Methods, Glowinski et al., Eds. John Wiley, New York, 1997, pp. 317–323.

16. J. W. Barrett and K. W. Morton, ''Approximate symmetrization and Petrov–Galerkin methods for diffusion–convection problems,'' Comp. Meth. Appl. Mech. Engrg. 45, 97–122 (1984).

17. E. T. Bouloutas and M. A. Celia, ''An improved cubic Petrov–Galerkin method for simulation of transient advection–diffusion processes in rectangularly decomposable domains,'' Comp. Meth. Appl. Mech. Engrg. 91, 289–308 (1991).

18. I. Christie, D. F. Griffiths, A. R. Mitchell, and O. C. Zienkiewicz, ''Finite element methods for second-order differential equations with significant first derivatives,'' Int. J. Num. Engrg. 10, 1389–1396 (1976).

19. R. Falk and G. R. Richter, ''Local error estimates for a finite element method for hyperbolic and convection–diffusion equations,'' SIAM J. Num. Anal. 29, 730–754 (1992).

20. T. J. R. Hughes and A. N. Brooks, ''A multi-dimensional upwind scheme with no crosswind diffusion,'' in Finite Element Methods for Convection Dominated Flows, Hughes, Ed. ASME, New York, 1985, pp. 19–35.

21. C. Johnson and J. Pitkäranta, ''An analysis of discontinuous Galerkin methods for a scalar hyperbolic equation,'' Math. Comp. 46, 1–26 (1986).

22. C. Johnson, A. Szepessy, and P. Hansbo, ''On the convergence of shock-capturing streamline diffusion finite element methods for hyperbolic conservation laws,'' Math. Comp. 54, 107–129 (1990).

23. M. G. Crandall and A. Majda, ''Monotone difference approximations for scalar conservation laws,'' Math. Comp. 34, 1–21 (1980).

24. C. N. Dawson, ''Godunov-mixed methods for advective flow problems in one space dimension,'' SIAM J. Num. Anal. 28, 1282–1309 (1991).

25. A. Harten, B. Engquist, S. Osher, and S. Chakravarthy, ''Uniformly high order accurate essentially nonoscillatory schemes, III,'' J. Comp. Phys. 71, 231–241 (1987).

26. C. Shu and S. Osher, ''Efficient implementation of essentially non-oscillatory shock capturing schemes,'' J. Comp. Phys. 77, 439–471 (1988).

27. R. Babarsky, R. E. Ewing, R. C. Sharpley, and J. Sochacki, ''Report on grid refinements schemes for RAMS,'' SRS Technical Report (1994).

28. J. P. Benque and J. Ronat, ''Quelques difficulties des modeles numeriques en hydraulique.'' In Computing Methods in Applied Sciences and Engineering, Glowinski and Lions, Eds., North–Holland, Amsterdam, 1982, pp. 471–494.

29. H. K. Dahle, M. S. Espedal, R. E. Ewing, and O. Sævareid, ''Characteristic adaptive sub-domain methods for reservoir flow problems,'' Num. Meth. PDEs 6, 279–309 (1990).

30. J. Douglas, Jr. and T. F. Russell, ''Numerical methods for convection-dominated diffusion problems based on combining the method of characteristics with finite element or finite difference procedures,'' SIAM J. Num. Anal. 19, 871–885 (1982).

31. M. S. Espedal and R. E. Ewing, ''Characteristic Petrov–Galerkin subdomain methods for two-phase immiscible flow,'' Comp. Meth. Appl. Mech. Engrg. 64, 113–135 (1987).

32. K. W. Morton, A. Priestley, and E. Süli, ''Stability of the Lagrangian–Galerkin method with nonexact integration,'' RAIRO Model. Math. Anal. Num. 22, 123–151 (1988).

33. S. P. Neuman, ''An Eulerian–Lagrangian numerical scheme for the dispersion–convection equation using conjugate space-time grids,'' J. Comp. Phys. 41, 270–294 (1981).

34. O. Pironneau, ''On the transport-diffusion algorithm and its application to the Navier-Stokes equations,'' Num. Math. 38, 309–332 (1982).

35. E. Varoglu and W. D. L. Finn, ''Finite elements incorporating characteristics for one-dimensional diffusion–convection equation,'' J. Comp. Phys. 34, 371–389 (1980).

36. H. Wang, M. Al–Lawatia, and S. A. Telyakovskiy, ''A Runge–Kutta characteristic method for first-order linear hyperbolic equations,'' Num. Meth. PDEs 13, 617–661 (1997).

37. D. Brown, ''A note on the numerical solution of the wave equation with piecewise smooth coefficients,'' Math. Comp. 42, 369–391 (1984).

38. T. Gimse and N. H. Risebro, ''A note on reservoir simulation for heterogeneous porous media,'' Trans port in Porous Media 10, 257–270 (1993).

39. L. Trefethen, ''Stability of finite-difference models containing two boundaries or interfaces,'' Math. Comp. 45, 279–300 (1985).

40. W. Lick, K. Ziegler, and J. Lick, ''Interior and boundary difference equations for hyperbolic differential equations,'' Num. Meth. PDEs, 2, 157–172 (1986).

41. J. Sochacki, ''An analysis of the continuous and finite difference equations for acoustic and elastic wave phenomena,'' Ph.D. Thesis, University of Wyoming, Wyoming, 1985.

42. M. Berger, ''Stability of interfaces with mesh refinement,'' Math. Comp. 45, 301–318 (1985).

43. T. Lin, J. Sochacki, R. Ewing, and J. George, ''Some grid refinement schemes for hyperbolic equations with piecewise constant coefficients,'' Math. Comp. 56, 61–86 (1991).

44. M. Ainsworth and A. Craig, ''A posteriori error estimator in the finite element method,'' Num. Math. 60, 429–463 (1992).

45. I. Babuška and W. C. Rheinboldt, ''A posteriori error analysis of finite element solutions for one dimensional problems,'' SIAM J. Num. Anal. 18, 556–589 (1981).

46. I. Babuška et al., Eds., Accuracy Estimates and Adaptive Refinements in Finite Element Computations, John Wiley, New York, 1986.

47. T. Lin and H. Wang, ''A class of error estimators based on interpolating the finite element solutions for reaction-diffusion equations,'' in Modeling, Mesh Generation, and Adaptive Numerical Methods for Partial Differential Equations, Babuska et al., Eds., IMA Volume in Mathematics and Its Applications, Vol. 75, Springer–Verlag, New York, 1995, pp. 129–152.

48. O. C. Zienkiewicz and J. Z. Zhu, ''The superconvergent patch recovery and a posteriori error estimates, Part I: the recovery technique,'' Int. J. Num. Meth. Engrg. 33, 1331–1364 (1992).

49. E. Süli and P. Houston, ''Finite element methods for hyperbolic problems: a posteriori error analysis and adaptivity,'' Oxford University Computing Lab. Research Reports, Numerical Analysis Group, NA-96-09, 1996.

50. R. DeVore and V. Popov, ''Interpolation spaces and nonlinear approximation,'' in Lecture Notes in Mathematics, Vol. 1302, Cwikel et al., Eds., Springer–Verlag, New York, 1988, pp. 191–205.

51. P. Petrushev, ''Direct and converse theorems for spline and rational approximation and Besov spaces,'' in Lecture Notes in Mathematics, Vol. 1302, Cwikel et al., Eds., Springer–Verlag, New York, 1988, pp. 363–377.

52. C. Bennett and R. C. Sharpley, Interpolation of Operators, Pure and Applied Mathematics, Vol. 129, Academic Press, New York, 1988.

53. R. DeVore and G. G. Lorentz, Constructive Approximation, Springer–Verlag, New York, 1993.

54. R. DeVore and B. Lucier, ‘‘High order regularity for the solution of the inviscid Burger’s equation,’’ in Lecture Notes in Mathematics, Vol. 1402, Carrasso et al., Eds., Springer–Verlag, New York, 1989, pp. 406–413.

55. R. DeVore and B. Lucier, ‘‘High order regularity for conservation laws,’’ Indiana Math. J. 39, 413–430 (1990).

56. T. L. Clark and R. D. Farley, ‘‘Severe downslope windstorm calculations in two and three spatial dimensions using anelastic interactive grid nesting: A possible mechanism for gustiness,’’ J. Atmospheric Sciences 41, 329–350 (1984).

57. T. Lin, ‘‘Numerical interfaces in finite difference methods for hyperbolic equations with discontinuous coefficients,’’ J. Comp. Acoust. 1, 151–184 (1993).