Journal of Computational Physics **180,** 710–735 (2002)
doi:10.1006/jcph.2002.7113

Unconditionally Stable Methods for Hamilton–Jacobi Equations

Kenneth Hvistendahl Karlsen and Nils Henrik Risebro

Department of Mathematics, University of Bergen, Johs. Brunsgt. 12, N-5008 Bergen, Norway; and Department of Mathematics, University of Oslo, P.O. Box 1053, Blindern, N-0316 Oslo, Norway
E-mail: kenneth.karlsen@mi.uib.no and nilshr@math.uio.no

Received May 2, 2000; revised May 1, 2002

We present new numerical methods for constructing approximate solutions to the Cauchy problem for Hamilton–Jacobi equations of the form $u_t + H(D_x u) = 0$. The methods are based on dimensional splitting and front tracking for solving the associated (non-strictly hyperbolic) system of conservation laws $p_t + D_x H(p) = 0$, where $p = D_x u$. In particular, our methods depend heavily on a front tracking method for one-dimensional scalar conservation laws with discontinuous coefficients. The proposed methods are unconditionally stable in the sense that the time step is not limited by the space discretization and they can be viewed as "large-time-step" Godunov-type (or front tracking) methods. We present several numerical examples illustrating the main features of the proposed methods. We also compare our methods with several methods from the literature. © 2002 Elsevier Science (USA)

Key Words: Hamilton–Jacobi equation; conservation law; discontinuous coefficient; numerical method; front tracking; operator splitting; numerical example.

1. INTRODUCTION

In this paper we present unconditionally stable numerical methods for the Cauchy problem for multidimensional Hamilton–Jacobi equations

$$\begin{cases} u_t + H(D_x u) = 0, & \text{in } \mathbf{R}^d \times \{t > 0\}, \\ u = u_0, & \text{on } \mathbf{R}^d \times \{t = 0\}. \end{cases} \tag{1}$$

In (1), $u = u(x, t)$ is the scalar unknown function that is sought, $u_0 = u_0(x)$ is a Lipschitz continuous initial function, H is a Lipschitz continuous Hamiltonian, and D_x denotes the gradient with respect to $x = (x_1, \ldots, x_d)$ defined by $D_x u = (u_{x_1}, \ldots, u_{x_d})$. Hamilton–Jacobi equations arise in a variety of applications, ranging from image processing, via mathematical finance, to the description of evolving interfaces (front propagation problems).

It is well-known that solutions of (1) generically develop discontinuous derivatives in finite time even with a smooth initial condition. Moreover, generalized solutions (i.e., locally Lipschitz continuous functions satisfying the equation almost everywhere) are not uniquely determined by their initial data and an additional selection principle—a so-called entropy condition—is needed to single out a physically relevant generalized solution. The most commonly used entropy condition is the vanishing viscosity condition which requires that the (correct) solution of (1) to be the vanishing viscosity limit of smooth solutions of corresponding viscous problems.

The vanishing viscosity entropy condition gives raise to the notion of viscosity solutions introduced by Crandall and Lions [7]. In particular, these authors established the existence, uniqueness, and stability of a viscosity solution of (1). Since then the theory of viscosity solutions has been intensively studied and even extended to large class of fully nonlinear second-order partial differential equations. We refer to Bardi *et al.* [2] for recent references to the theory of viscosity solutions and applications. In passing, we mention that Kružkov has developed an alternative (equivalent) theory for Hamilton–Jacobi equations with a convex Hamiltonian (see, e.g., [29]).

It is known that the Hamilton–Jacobi equations are closely related to (scalar) conservation laws

$$\begin{cases} v_t + \sum_{i=1}^{d} f_i(v)_{x_i} = 0, & \text{in } \mathbf{R}^d \times \{t > 0\}, \\ v(x,0) = v_0(x) & \text{on } \mathbf{R}^d \times \{t = 0\}. \end{cases} \tag{2}$$

Here $v = v(x,t)$ is the scalar unknown, $v_0 = v_0(x)$ is a bounded initial function, and $f_1, \dots, f_d$ are Lipschitz continuous flux functions. In contrast to the Hamilton–Jacobi equations, which possess at least continuous solutions, solutions to (2) develop discontinuities (shock waves) in finite time and therefore one has to consider solutions to (2) in the sense of distributions. However, distributional solutions are not uniquely determined by their initial data and one needs also here the vanishing viscosity entropy condition to pick out the correct solution. In the context of scalar conservation laws (2), the vanishing viscosity condition gives rise to the notion of entropy solutions in the sense of Kružkov. Kružkov [31] proved that the well posedness of (2) is ensured within the framework of entropy solutions.

In the one-dimensional case ($d = 1$), it is well-known that the existence of viscosity solutions of (1) is equivalent to the existence of entropy solutions of (2) (see [6, 22, 24, 29, 36]). More precisely, if $u = u(x,t)$ is the unique viscosity solution of (1), then $v = D_x u$ is the unique entropy solution of (2). Conversely, if $v = v(x,t)$ is the unique entropy solution of (2), then u defined via $u(x,t) = \int_{-\infty}^{x} v(\xi,t)\,d\xi$ is the unique viscosity solution (1). In the multidimensional case ($d > 1$), this one-to-one correspondence no longer exists. Instead the gradient $p = (p_1, \dots, p_d) = D_x u$ satisfies (at least formally) a $d \times d$ system of conservation laws [22, 29, 36].

$$\begin{cases} (p_1)_t + H(p_1, \dots, p_d)_{x_1} = 0, \\ \qquad\qquad \ddots \\ (p_d)_t + H(p_d, \dots, p_d)_{x_d} = 0. \end{cases} \tag{3}$$

If p is known, one may recover u from p by integrating the ordinary differential equation

$$u_t + H(p_1, \dots, p_d) = 0. \tag{4}$$

One should notice that (3) is only weakly hyperbolic. Nevertheless, in [22, 29, 36] it is proved that the vanishing viscosity limit solutions of (1) and (3) (when such exist for both problems!) are equivalent. Roughly speaking, one may therefore in the multidimensional case also think of viscosity solutions to (1) as primitives of (vanishing viscosity) solutions to (3).

Equipped with this view, it becomes natural to exploit some of the numerical concepts developed for hyperbolic conservation laws when developing numerical methods for Hamilton–Jacobi equations. Indeed, many well-known shock-capturing methods for conservation laws have been extended to Hamilton–Jacobi equations (see [8, 37] for finite difference schemes of upwind type (see also [30]); and see [1, 28] for finite volume schemes, [40, 41, 21] for (W)ENO schemes, [35, 32] for central schemes, [3, 19] for finite element methods, and [22] for relaxation schemes).

In contrast to the shock-capturing schemes just cited, we will in this paper be concerned with extending to Hamilton–Jacobi equations (1) a so-called front tracking method for conservation laws. The front tracking method was introduced by Dafermos [9] as a (mathematical) tool for constructing entropy solutions to one-dimensional scalar conservation laws. Holden and co-workers, [15, 16] later proved that Dafermos' construction procedure was well defined and developed it into an L^1 linearly(!) convergent numerical method. Front tracking was later extended to systems of equations by Bressan [4] and Risebro [43], who used the method to give an alternative proof of Glimm's famous existence result for hyperbolic systems. Very recently a modification of the front tracking method was used by Bressan *et al.* [5] to prove stability and uniqueness of weak solutions of strictly hyperbolic systems of conservation laws. The front tracking method was used by Risebro and Tveito [44, 45] to numerically solve the Euler equations of gas dynamics and a non-strictly hyperbolic system modeling polymer flow.

Holden and Risebro [18] extended the scalar front tracking method to multidimensional scalar conservation laws by means of dimensional splitting. These authors also proved that the method converges to the unique entropy solution of the governing problem. An L^1-error estimate of order $1/2$ was proved in [23]. Although the convergence rate drops from 1 in the one-dimensional case to $1/2$ in the multidimensional case, it should be noted that no CFL condition is associated with the multidimensional numerical method, which implies that the method is fast compared with conventional difference methods. Computations using CFL numbers as high as 10–20 (with satisfactory results) have been reported (see Lie *et al.* [34]). Computational results for multidimensional hyperbolic systems can be found in [14, 17] and in Haugse *et al.* [13].

The purpose of this paper is to device front tracking methods for Hamilton–Jacobi equations. In the one-dimensional case (see [24]), we simply rely on the equivalence between (1) and (2) and define a numerical method for (1) by "integrating" the front tracking method [15, 16]. The resulting numerical method for (1) is well defined and L^∞ linearly convergent toward the unique viscosity solution of the governing problem. The linear convergence rate follows from the results in [15, 16] or [38] (see also [24]).

The multidimensional case is much more difficult and is the main focus of this paper. The basis for our numerical methods is the (formal) relation between (1) and the weakly hyperbolic system (3). The methods that we present can all be written as explicit marching schemes of the type

$$u_J^{n+1} = u_J^n - \Delta t \mathcal{H}_J, \tag{5}$$

where $J = (j_1, \dots, j_d) \in \mathbf{Z}^d$ is a multiindex and $\mathcal{H}_J$ is the numerical Hamiltonian that has to be determined. Typically, $\mathcal{H}_J$ is a convex combination of one-dimensional numerical Hamiltonians $\mathcal{H}_J^1, \dots, \mathcal{H}_J^d$.

To construct the numerical Hamiltonians $\mathcal{H}_J^1, \dots, \mathcal{H}_J^d$, we first apply a sort of dimensional splitting to reduce the $d \times d$ system of conservation laws (3) to a sequence of (decoupled) one-dimensional scalar conservation laws of the form

$$(p_i)_t + H(p_1, \dots, p_i, \dots, p_d)_{x_i} = 0, \quad i = 1, \dots, d, \tag{6}$$

where $p_j = p_j(x)$, $j \neq i$, are fixed, possibly discontinuous coefficients. These equations can all be viewed as one-dimensional scalar conservation laws of the type

$$v_t + f(a, v)_x = 0, \quad x \in \mathbf{R}, t > 0,$$

where f is some flux function and $a = a(x, t)$ is a given, possibly discontinuous coefficient. The fact that $a(x, t)$ can be discontinuous makes analysis of numerical methods for such conservation laws rather difficult. Front tracking for conservation laws with a flux function that depends discontinuously on the space variable is analyzed in Gimse and Risebro [12], Klingenberg and Risebro [26, 27], and Klausen and Risebro [25]. Recently some difference schemes for such conservation laws were proved to be convergent by Towers [47]. Roughly speaking, we shall in this paper build our numerical Hamiltonians $\mathcal{H}_J^1, \dots, \mathcal{H}_J^d$ (to be used in (5)) by applying the front tracking method to the scalar conservation laws in (6).

The rest of this paper is organized as follows: In the next section, we describe the front tracking algorithm for one-dimensional scalar conservation laws with discontinuous coefficients; in particular, we discuss the solution of the Riemann problem. Section 3 first describes a front tracking method for Hamilton–Jacobi equations in one dimension, then it details the various numerical methods for multidimensional Hamilton–Jacobi equations which can be build from the front tracking method. These schemes are then tested on several problems in Section 4. Finally, we draw some conclusions in Section 5.

2. FRONT TRACKING IN ONE DIMENSION

In this section we describe the front tracking algorithm for one-dimensional conservation laws in some detail. Therefore we consider the one-dimensional scalar conservation law

$$v_t + f(a(x), v)_x = 0, \quad v(x, 0) = v_0(x). \tag{7}$$

Here the unknown $v = v(x, t)$ is a scalar and the "coefficient" $a(x)$ is assumed to be a bounded, piecewise differentiable function, but not necessarily continuous. We shall always assume that f is a Lipschitz continuous function.

Front tracking is a method to compute approximate weak solutions to (7). Let first δ be a parameter indicating the accuracy of the approximation, and let v_0^δ and a^δ be piecewise constant approximations to v_0 and a, respectively, such that

$$v_0^\delta \to v_0 \quad \text{and} \quad a^\delta \to a \quad \text{in } L^1_{\text{loc}} \text{ as } \delta \to 0.$$

The Riemann problem for (7) is the initial value problem where v_0 and a take the form

$$v_0(x) = \begin{cases} v_l & \text{for } x \leq 0, \\ v_r & \text{for } x > 0, \end{cases} \quad a(x) = \begin{cases} a_l & \text{for } x \leq 0, \\ a_r & \text{for } x > 0. \end{cases} \tag{8}$$

Hence, v_0^δ and a^δ defines a series of Riemann problems located at their discontinuities. If $\partial f/\partial v$ is bounded, then (7) has finite speed of propagation, and the solutions of neighboring Riemann problems will not interact for small t. Therefore, we can compute the entropy solutions to the initial Riemann problems, and thereby the solution of (7), with $v_0 = v_0^\delta$, for sufficiently small t. However, being able to compute the solution of Riemann problems does not help us to compute the solution past the time where waves from different Riemann problems interact. Generally, the solution of the Riemann problem (7)–(8) is a function of x/t and is not always piecewise constant.

Front tracking is a strategy to remedy this. We choose a piecewise constant (in x/t) approximation $v^\delta(x, t)$ to the solution of the Riemann problem such that

$$v^\delta(\,\cdot\,, t) \to v(\,\cdot\,, t) \quad \text{in } L^1 \text{ as } \delta \to 0.$$

If we approximate all the initial Riemann problems defined by v_0^δ and a^δ in this manner, the resulting function will be piecewise constant in x, with discontinuities emanating in fans from each initial discontinuity. Collisions between these discontinuities will define new Riemann problems (since v^δ is piecewise constant). We can approximately solve these Riemann problems in the same way (giving new discontinuities that move in straight lines) and thereby continuing the approximation beyond the interaction time. We call the function defined in this way v^δ and the discontinuities in v^δ *fronts*. The approximation process we call *front tracking*.

Note that it is not clear whether we are able to continue the front tracking approximation up to any prescribed time t (this depends on how we construct the approximate Riemann solution). Moreover, we must be able to construct an approximate solution of any Riemann problem arising from collisions.

2.1. *The Riemann Problem*

When constructing a front tracking algorithm for (7), we must solve two types of Riemann problems depending on whether $a_l = a_r$ or not. If $a_l = a_r$, then we have a Riemann problem for a scalar conservation law. This can be solved by taking envelopes (see e.g., Holden and Holden [15] for a description of the solution procedure). Note that if f is piecewise linear and continuous in v, the solution of the Riemann problem will be a piecewise constant function of x/t.

If $a_l \neq a_r$, the solution of (7)–(8) is more complicated. We need to determine those v values that can be connected to v_l via a Riemann solution containing only waves of negative speed, and those v values that can be connected to v_r via waves of positive speed, and to determine the limits $\lim_{x\uparrow 0} v(x/t)$ and $\lim_{x\downarrow 0} v(x/t)$. We now follow [11] and describe how the solution can be found.

First we define the functions

$$h_l(v; v_l) = \begin{cases} \sup\left\{ \begin{matrix} g \mid g \text{ continuous and nonincreasing,} \\ g(v) \leq f(a_l, v),\ g(v_l) = f(a_l, v_l) \end{matrix} \right\} & \text{for } v \geq v_l, \\ \inf\left\{ \begin{matrix} g \mid g \text{ continuous and nonincreasing,} \\ g(v) \geq f(a_l, v),\ g(v_l) = f(a_l, v_l) \end{matrix} \right\} & \text{for } v \leq v_l \end{cases} \tag{9}$$

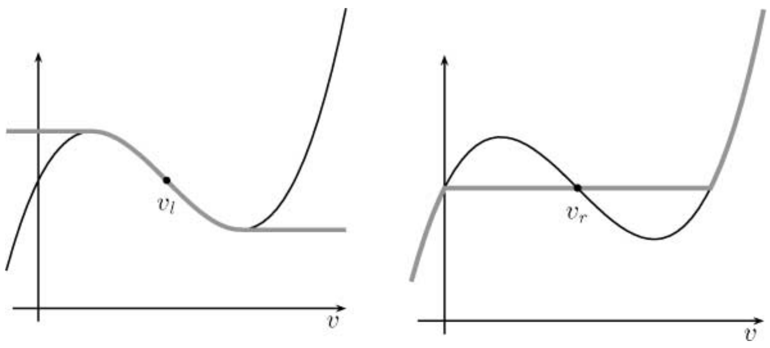

FIG. 1. (Left) $h_l(v, v_l)$; (right) $h_r(v; v_r)$.

and

$$h_r(v; v_r) = \begin{cases} \inf\left\{\begin{matrix} g \mid g \text{ continuous and nondecreasing,} \\ g(v) \geq f(a_r, v),\ g(v_r) = f(a_r, v_r) \end{matrix}\right\} & \text{for } v \geq v_r, \\ \sup\left\{\begin{matrix} g \mid g \text{ continuous and nondecreasing,} \\ g(v) \leq f(a_r, v),\ g(v_r) = f(a_r, v_r) \end{matrix}\right\} & \text{for } v \leq v_r. \end{cases} \tag{10}$$

Note that h_l is nonincreasing and h_r nondecreasing. In Fig. 1 we show an example of these functions, the flux function f is shown in a thin line, and the functions $h_{l,r}$ are shown in a thicker gray line. Next we define

$$H_l = \{v \mid f(a_l, v) = h_l(v; v_l)\},$$
$$H_r = \{v \mid f(a_r, v) = h_r(v; v_r)\}.$$

For any $\tilde{v}_r \in H_l$ the Riemann problem

$$v_t + f(a_l, v)_x = 0, \quad v_0(x) = \begin{cases} v_l & x < 0, \\ \tilde{v}_r & x \geq 0, \end{cases} \tag{11}$$

has a solution that consists only of waves of nonpositive speed. Similarly for any $\tilde{v}_l \in H_r$ the Riemann problem

$$v_t + f(a_r, v)_x = 0, \quad v_0(x) = \begin{cases} \tilde{v}_l & x < 0, \\ v_r & x \geq 0, \end{cases} \tag{12}$$

has a solution with waves of nonnegative speed only. Next, we observe that the Rankine–Hugoniot condition implies that, for the solution of (7)–(8),

$$\lim_{x\uparrow 0} f(a_l, v(x/t)) = \lim_{x\downarrow 0} f(a_r, v(x/t)),$$

i.e., the flux is continuous across discontinuities in a. This means that the flux value at $x = 0$ must be given by

$$h_l(\cdot; v_l) = h_r(\cdot; v_r). \tag{13}$$

If the graphs of h_l and h_r do not intersect, then the Riemann problem has no bounded solution, and if these graphs intersect, then the flux value determined by (13) is unique. For the flux functions considered in this paper, we always have an intersection. Once the flux value at $x = 0$ has been determined, we choose a $\tilde{v}_r \in H_l$ and a $\tilde{v}_l \in H_r$ such that

$$h_l(\tilde{v}_r; v_l) = h_r(\tilde{v}_l; v_r) \quad \text{and} \quad |\tilde{v}_r - \tilde{v}_l| \text{ is minimized.} \tag{14}$$

This is the *minimal jump condition* taken from [11] (see also Diehl [10]). Note that although $\tilde{v}_{r,l}$ may not be unique (there may be two choices both minimizing the jump), the resulting solution $v(x/t)$ will be unique almost everywhere. If we also demand, among the possible solutions to (14), that

$$|v_l - \tilde{v}_r| \quad \text{and} \quad |\tilde{v}_l - v_r| \tag{15}$$

are minimal, then $\tilde{v}_r$ and $\tilde{v}_l$ are unique. Now the solution to the Riemann (7)–(8) can be found by piecing together the solutions to (11) and (12). Precisely, let $\bar{v}_l(x/t)$ denote the solution to (11) and $\bar{v}_r(x/t)$ the solution to (12) where $\tilde{v}_r$ and $\tilde{v}_l$ are chosen according to (14) and (15). Then the solution of (7)–(8) is

$$v(x/t) = \begin{cases} \bar{v}_l(x/t) & x < 0, \\ \bar{v}_r(x/t) & x > 0. \end{cases}$$

2.1.1. Convex f. If $f(a, v)$ is uniformly convex in v and monotone in a, the above construction simplifies considerably. Also the front tracking algorithm can be proved to be well defined. More precisely, from [25] we have the following theorem:

THEOREM 2.1. *Assume that a is in $L^1 \cap BV$ and is piecewise C^1 with a finite number of discontinuities. Assume also that $v_0(x)$ is such that $f(a, v_0)$ is of bounded variation. Then there exists a unique weak solution u to* (7) *such that $v^\varepsilon \to u$ in L^1, where v^ε solves the "regularized" problem*

$$\begin{cases} v^\varepsilon_t + f(a^\varepsilon, v^\varepsilon)_x = 0 & \text{in } \mathbf{R} \times \{t > 0\}, \\ v^\varepsilon = v_0 * \omega_\varepsilon & \text{on } \mathbf{R} \times \{t = 0\}, \end{cases} \tag{16}$$

where $a^\varepsilon = a * \omega_\varepsilon$ and ω_ε is the usual mollifying kernel with radius ε. Furthermore, u satisfies the wave entropy condition

$$\operatorname{sign}(f_{vv})\partial_x(f_v(a, v)) \geq K\left(\frac{1}{t} + |a'|\right) \tag{17}$$

in each interval where a' exists. The constant K depends on f, $\|a\|_\infty$, and v_0, but not on a'. Furthermore if v^δ denotes the front tracking approximation to v, then

$$\lim_{\delta \to 0} v^\delta = v \quad \text{in } L^1_{\text{loc}}.$$

Also, there are only a finite number of collisions between fronts in v^δ for all $t \in [0, \infty\rangle$.

The proof of this theorem can be found in [25]. Here we detail the approximate solution of the Riemann problem. The assumptions on f imply that for each a there is a unique v_T

such that

$$f_v(a, v_T) = 0.$$

For simplicity we set $v_T = 0$. Let $z(v, a)$ and $b(a)$ be defined as

$$\begin{aligned} z = z(a, v) &= \operatorname{sign}(v - v_T)(f(a, v) - f(a, v_T)), \\ b(a) &= f(a, v_T). \end{aligned} \tag{18}$$

Note that since $f_a \neq 0$, the mapping $b(a)$ is one to one. Hence, the mapping

$$(a, v) \mapsto w = (b, z)$$

is injective and regular everywhere except for $z = 0$. Thus the Riemann problem is determined by two states, w_l and w_r.

In the following we use the notation f_l for $f(a_l, v_l)$, and similarly for other functions of the left and right states. We say that two states, w_l and w_r, are connected by an a-wave if $\operatorname{sign}(z_l) = \operatorname{sign}(z_r)$ and $f_l = f_r$; similarly we say that they are connected by a z wave if $a_l = a_r$. The solution of the Riemann problem in the (z, b) plane is depicted in Fig. 2. To find a particular solution, pick a right state v_r and follow the arrows from v_l to v_r. This traces out a series of waves, e.g., zaz, and the solution is then a found by connecting the v_l to the state to the right of the first z wave and so on. This diagram is entirely similar to the corresponding diagrams in [27, 46]. The actual waves occurring in a z wave is found by solving the scalar Riemann problem with constant a (either a_l or a_r) and v_l and v_r given by the endpoints of the curve. If the solution is determined by a zaz sequence, the first z wave will have nonpositive speed, a waves will always have zero speed (they are discontinuities in $a(x)$), and the second z wave will have nonnegative speed. Note in particular that in the (z, b) plane, all waves trace curves which are either horizontal lines (z waves) or straight lines at an angle of 45° slope (a waves). Hence if we fix a grid $(z, b) = (i\delta, j\delta)$ (for $i, j \in \mathbf{Z}$ and some small number $\delta > 0$) in the (z, b) plane and if the initial states w_l and w_r are points on the grid, then all intermediate states will also be points on the grid. Furthermore, if we interpolate $f(a, v)$ linearly between grid points, the solution of the scalar Riemann problems determined by the z waves will consist of piecewise constant functions in x/t

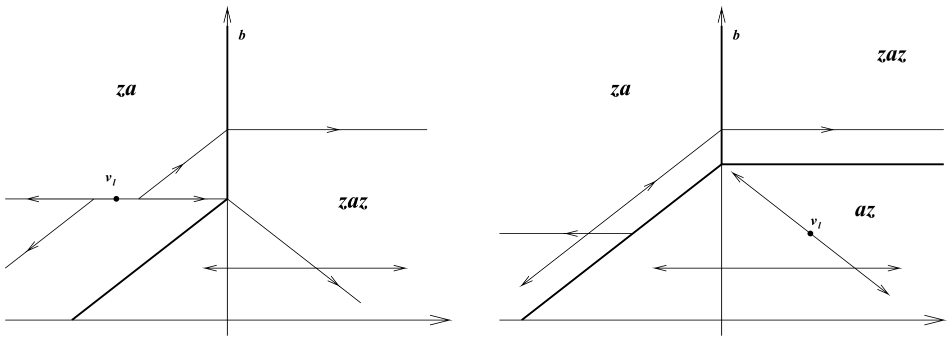

FIG. 2. The solution of the Riemann problem.

(see, e.g., [16]). Let this interpolation of f be denoted f^δ. Then the above construction yields an entropy weak solution to the initial value problem

$$v_t^\delta + f^\delta(a(x), v^\delta)_x = 0, \quad v^\delta(x,0) = \begin{cases} w^{-1}(i\delta, b(a_l)), & x \le 0, \\ w^{-1}(j\delta, b(a_r)), & x > 0, \end{cases} \tag{19}$$

for any integers i and j. This solution will be piecewise constant in x/t where $w(v^\delta(x,t), a(x))$ will be on the grid for all x and t.

We can also construct the approximation of the initial function v_0^δ and the "coefficient" a^δ such that

$$w\left(v_0^\delta(x), a^\delta(x)\right)$$

is on the grid in the (z, b) plane. For a fixed δ, we can then solve the initial value problem

$$v_t^\delta + f^\delta(a^\delta, v^\delta)_x = 0, \quad v^\delta(x,0) = v_0^\delta(x) \tag{20}$$

exactly using front tracking (see, e.g., [27]). To do this we must also solve the Riemann problems where $a_l = a_r$. This is not difficult, since f is convex in u. Concretely, the solution of the Riemann problem

$$u_t + f(a_l, u)_x = 0, \quad u_0(x) = \begin{cases} w^{-1}(i\delta, b(a_l)), & x \le 0, \\ w^{-1}(j\delta, b(a_l)), & x > 0, \end{cases}$$

is straightforward to find: For $i \ne j$ set

$$s(i,j) = \frac{f(a_k, w^{-1}(i\delta, b(a_l))) - f(a_k, w^{-1}(j\delta, b(a_l)))}{w^{-1}(i\delta, b(a_l)) - w^{-1}(j\delta, b(a_l))}.$$

If $i < j$, then

$$u(x,t) = \begin{cases} w^{-1}(i\delta, b(a_l)), & x \le ts(i, i+1), \\ w^{-1}(k\delta, b(a_l)), & ts(k-1, k) < x \le ts(k, k+1) \quad \text{for } k = i+1, \dots, j-1, \\ w^{-1}(j\delta, b(a_l)), & ts(j-1, j) \le x, \end{cases}$$

while for $i > j$ the solution is given by

$$u(x,t) = \begin{cases} w^{-1}(i\delta, b(a_l)), & x < s(i,j)t, \\ w^{-1}(j\delta, b(a_l)), & x \ge s(i,j)t. \end{cases}$$

This corresponds to taking the lower convex envelope of the flux function between u_l and u_r if $u_l < u_r$, and the upper concave envelope if $u_l > u_r$. When computing, we do not need to compute w^{-1}, etc., but merely to associate with each front a left and a right state that are

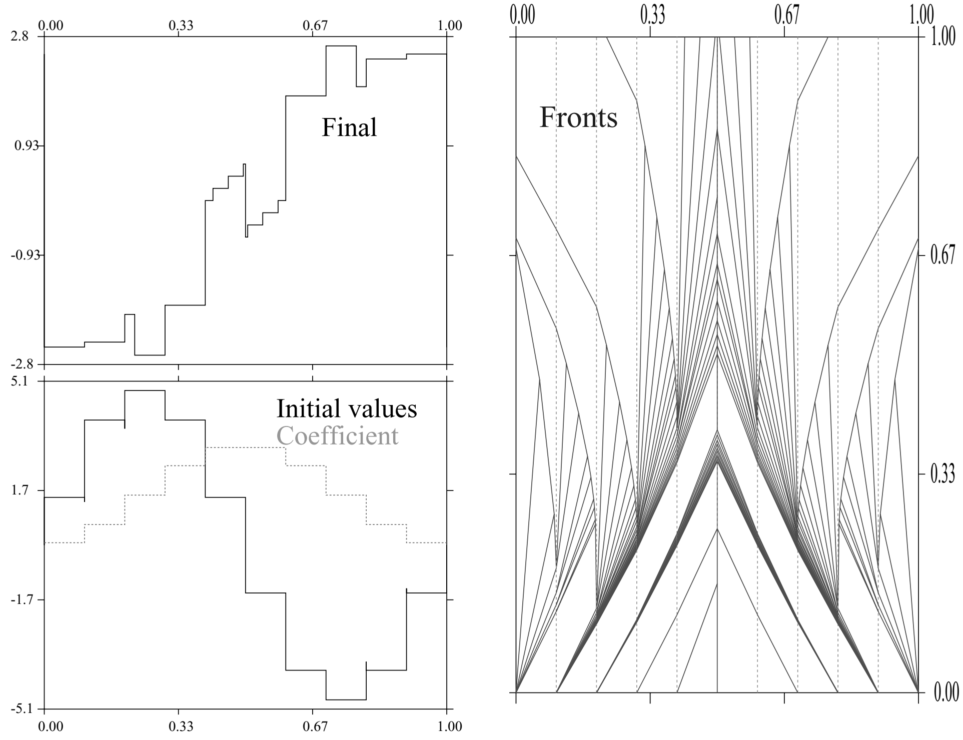

FIG. 3. An example of front tracking.

pairs of integers representing the grid coordinates of the state. The conversion to u can then be done at the end of the computation.

In [27], it is shown that front tracking can be continued up to any predefined time, and that for each δ there will only be a finite number of collisions between fronts in v^δ.

In Fig. 3 we show the fronts and the initial and final states for the initial value problem (20) with

$$f(a, v) = \sqrt{1 + a^2 + v^2}, \quad v_0(x) = \frac{\pi^2}{2} \sin(2\pi x), \quad a(x) = \pi(1 - \cos(2\pi x)), \tag{21}$$

and periodic boundary data. In this example we used $\delta = 0.25$. Figure 3 shows v_0^δ and a in the lower left corner, and $v^\delta(x, 1)$ in the upper left corner. To the right we see the fronts in the (x, t) plane. The z waves are shown as solid lines and the a waves as broken lines.

3. NUMERICAL ALGORITHMS

3.1. *One-Dimensional Algorithm*

In this section, we recast the front tracking method from the previous section as a method for solving one-dimensional Hamilton–Jacobi equations with a discontinuous coefficient. This method will be used as an important building block in the multidimensional algorithm presented in the next section.

The relevant initial value problem reads

$$u_t + H(a, u_x) = 0, \quad u(x, 0) = u_0(x), \tag{22}$$

where $H(a, p)$ is assumed to be differentiable.

Setting $p = u_x$ and formally differentiating the above problem, we find that p satisfies the scalar conservation law

$$p_t + H(a, p)_x = 0, \quad p(x, 0) = p_0(x) := u_{0x}(x). \tag{23}$$

When a is Lipschitz continuous, we recall that the viscosity solution of (22) is equivalent to the entropy solution of (23). However, when $a(\cdot)$ is discontinuous, the classical viscosity and entropy solution theories do not apply. Instead, we will rely on the solution theory developed in [25–27] for scalar conservation laws with discontinuous coefficients. The relevant results from this theory are summed up in Section 2. In particular, if H is convex, we know that the problem (23) has a unique solution—the so-called entropy solution—that is the limit of the corresponding regularized solutions. Furthermore, according to recent results of Ostrov [42], if $p \mapsto H(a, p)$ is convex, we can define "viscosity solutions" of (22) even when a has a finite number of jump discontinuities. These are defined as the (unique!) limit of viscosity solutions of the "smoothed" equations

$$u_t^\varepsilon + H\big(a^\varepsilon, u_x^\varepsilon\big) = 0, \quad a^\varepsilon = a * \omega_\varepsilon.$$

Let p^δ be the front tracking approximation of (23). This algorithm is viable also as an algorithm for (22) almost without alterations. To define front tracking for (22), we need to keep track of the value of the approximate solution u^δ along each front in p^δ. Since p^δ is piecewise constant, u^δ will be continuous and piecewise linear between fronts. All fronts in p^δ will move with constant speed between collision points, so the position of a front is given by

$$x(t) = x_0 + s(t - t_0),$$

where (x_0, t_0) is the starting point of the front. Let (p_l, a_l) and (p_r, a_r) denote the left and right states of the front. Then

$$\begin{aligned} u^\delta(x(t), t) &= u^\delta(x_0, t_0) + (t - t_0)(sp_l - H(a_l, p_l)) \\ &= u^\delta(x_0, t_0) + (t - t_0)(sp_r - H(a_r, p_r)), \end{aligned} \tag{24}$$

because of the Rankine–Hugoniot condition

$$s(p_l - p_r) = H(a_l, p_l) - H(a_r, p_r).$$

In pseudocode, this algorithm is shown in Algorithm 1.

ALGORITHM 1 (Front Tracking Algorithm for Hamilton–Jacobi Equations).

```
function $[u, x]$=fthj1d($u$,$x$,$a$,$T$,$\delta$)
%  $u$:  piecewise linear approximation to $u_0$.
%  $x$:  the locations of the breakpoints in $u_0$
%        and of the discontinuities in $a$.
%  $a$:  piecewise constant approximation to $a(x)$.
%
  Find the grid in $(z, b)$ space defined by $u$, $a$ and $\delta$
  Solve the initial Riemann problem problems
  Find the smallest collision time: $t_{\mathrm{coll}}$
  while ($t_{\mathrm{coll}} < T$)
    Find the $u$-value at the collision point by (24)
    Solve the Riemann problem defined by the collision
    Find the smallest collision time: $t_{\mathrm{coll}}$
  end while
  Move all fronts $x$ and update the $u$-values by (24)
```

Figure 4 shows the front tracking approximation to the "Hamilton–Jacobi version" of (21), with $H(a, p) = f(a, p)$ and $u_0(x) = \int_0^x v_0(\sigma)\, d\sigma$, i.e.,

$$u_0(x) = \frac{\pi}{4}(1 - \cos(2\pi x)).$$

In Fig. 4, to the left we see the initial approximation, in the middle $u^\delta(x, 0.5)$, and to the right $u^\delta(x, 1)$.

We can modify this algorithm slightly by requesting that the input/output values u be on a *fixed* grid x_i rather than the u values on the fronts. To do this we define the grid in the (z, b) plane by combining the grid defined by δ by the grid defined by the initial values. This slight extension, Algorithm 2, is the one we will use in the dimensional splitting algorithm.

3.2. *A Godunov-Type Formulation*

The method just described is a good method for the one-dimensional problem (22). We now present a formulation which is easy to use as a building block in the multidimensional

ALGORITHM 2 (Front Tracking Algorithm for Hamilton–Jacobi Equations with a Fixed Grid $\{x_i\}$).

```
function $u$=grid_fthj1d($u$,$x$,$a$,$T$,$\delta$)
%  $u$:  piecewise linear approximation to $u_0$.
%  $a$:  piecewise constant approximation to $a(x)$.
%  $x$:  the grid points.
%
  $y = x$
  $[u, y]$=fthj1d($u$,$y$,$a$,$T$,$\delta$)
  Find $u(x_i)$ for all $i$
```

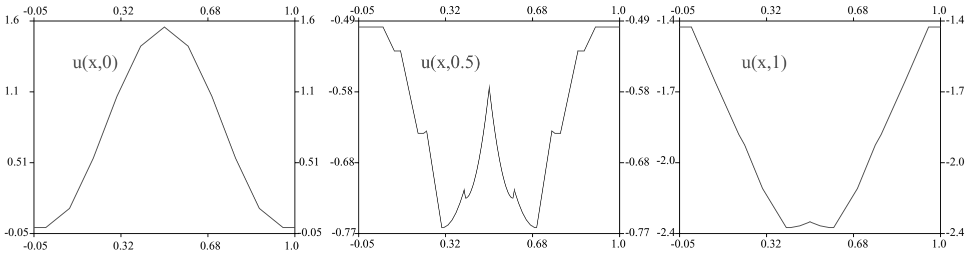

FIG. 4. Front tracking for Hamilton–Jacobi equations.

algorithms described in the next section. Namely, we would like to rewrite the method as an explicit marching scheme of the form

$$u(\cdot,(n+1)\Delta t) = u(\cdot, n\Delta t) - \Delta t\mathcal{H}(a, u(\cdot, n\Delta t)), \quad n = 0, 1, 2, \ldots, \tag{25}$$

for some numerical Hamiltonian $\mathcal{H}$ and time step $\Delta t > 0$.

Although the front tracking approximation did not use any predefined time step Δt, we can restart the front tracking algorithm at $t_n = n\Delta t$, $n = 1, 2, 3, \ldots$. To this end, let Δx be given and set

$$\begin{aligned} p_j^n &= \frac{1}{\Delta x}\left(u^n\left(x_{j+1/2}\right) - u^n\left(x_{j-1/2}\right)\right),\\ a_j &= \frac{1}{\Delta x}\int_{x_{j-1/2}}^{x_{j+1/2}} a(x)\,dx,\\ a_{\Delta x}(x) = a_j, &\quad \text{and} \quad p_{\Delta x}^n(x) = p_j^n \quad \text{for } x \in \left[x_{j-1/2}, x_{j+1/2}\right\rangle, \end{aligned}$$

where $x_j = j\Delta x$. Let δ be some small parameter and define a grid in the (z, b) plane by combining a regular grid of size δ with the grid determined by the points $w(p_j^0, a_j)$, as in Algorithm 2. This grid is the one we use to interpolate H (see Section 2), giving a function we label H^δ. For $t_n \le t < t_{n+1}$, let u^n be the front tracking solution of

$$u_t^n + H^\delta\left(a_{\Delta x}, u_x^n\right) = 0, \quad u^n(x, n\Delta t) = u^n\left(x_{1/2}\right) + \int_{x_{1/2}}^{x} p_{\Delta x}^n(\sigma)\,d\sigma. \tag{26}$$

Finally set

$$p_j^{n+1} = \frac{1}{\Delta x}\left(u^n\left(x_{j+1/2}, t_{n+1}-\right) - u^n\left(x_{j-1/2}, t_{n+1}-\right)\right).$$

We start this process by setting $u^0(x) = u_0(x)$. Note that $u^n(x, t_{n+1})$ is a piecewise linear continuous function in x, with break points located at $\{x_{j+1/2}\}$. By (24), we directly read off the value of $u_{j+1/2}^{n+1}$ from the front located at $x_{j+1/2}$. Since this is a point of discontinuity for $a_{\Delta x}$ there will be a front present at this location. If by chance $a_{\Delta x}$ is continuous here, we can easily add an extra front in the front tracking process.

Remark. Although we have (re)formulated the front tracking method as an explicit marching scheme (25) and thereby introduced a time step into the method, one should note that there is no CFL condition associated with (25), i.e., large time steps are allowed.

3.3. *Multidimensional Algorithms*

Now we use the Godunov-type method (25) to formulate "large-time-step" methods for the multidimensional problem (1). For ease of presentation, we shall restrict ourselves to two space dimensions, but the generalization to three or more dimensions is obvious. Therefore we study problems of the form

$$\begin{cases} u_t + H(u_x, u_y) = 0 & \text{in } \mathbf{R}^2 \times \{t > 0\}, \\ u = u_0 & \text{on } \mathbf{R}^2 \times \{t = 0\}, \end{cases} \tag{27}$$

where the Hamiltonian H is of the types discussed in the previous sections. We can write (27) as a 2×2 system conservation law formally obtained by differentiating (27),

$$\begin{aligned} p_t + H(p, q)_x &= 0, \\ q_t + H(p, q)_y &= 0, \end{aligned} \tag{28}$$

where $(p, q) = (u_x, u_y)$ and

$$(p, q)(x, y, 0) = ((u_0)_x, (u_0)_y)(x, y).$$

As already mentioned, in [36, 22] it is shown that the vanishing viscosity limit solution of (27) is equivalent to the vanishing viscosity limit solution of (28).

In order to define our scheme, we let $\delta > 0$ be some small number. All our computed quantities will depend on this number, but for simplicity our notation does not always indicate this dependency. We use a computational grid $x_j = j\Delta x$, $y_k = k\Delta y$, and $t_n = n\Delta t$ for small numbers Δx, Δy, Δt, and integers $j, k \in \mathbf{Z}$, $n = 0, \dots, N$, where $N\Delta t = T$. To integrate (28) numerically, we can use dimensional splitting or a direct approach.

3.3.1. Dimensional splitting. Dimensional splitting for (27) is based on the sequential solution of the two conservation laws in (28) for a time step Δt, using the result for one equation as coefficients in the other. Concretely, this gives the following scheme. First set,

$$U^0_{j+1/2,k} = u_0\big(x_{j+1/2}, y_k\big), \tag{29}$$

$$V^0_{j,k+1/2} = u_0\big(x_j, y_{k+1/2}\big), \tag{30}$$

$$p^0_{j,k} = \frac{1}{\Delta x}\big(U^0_{j+1/2,k} - U^0_{j-1/2,k}\big), \tag{31}$$

$$q^0_{j,k} = \frac{1}{\Delta y}\big(V^0_{j,k+1/2} - V^0_{j,k-1/2}\big). \tag{32}$$

For $n \geq 0$, for t in the interval $[t_n, t_{n+1}\rangle$, and for each k, we let $U^n_k(t)$ be the front tracking solution to

$$\begin{gathered} \big(U^n_k\big)_t + H^\delta\big(\big(U^n_k\big)_x, q^n_k\big) = 0, \\ U^n_k(x, t_n) = U^n_{j-1/2,k} + p^n_{j,k}\big(x - x_{j-1/2}\big) \quad \text{for } x \in \big[x_{j-1/2}, x_{j+1/2}\big\rangle, \end{gathered} \tag{33}$$

where the functions p^n_k and q^n_k are defined as

$$\left.\begin{aligned} q^n_k(x) &= q^n_{j,k} \\ p^n_k(x) &= p^n_{j,k} \end{aligned}\right\} \quad \text{for } x \in \big[x_{j-1/2}, x_{j+1/2}\big\rangle.$$

Then we can define

$$U^{n+1}_{j+1/2,k} = \lim_{t\uparrow t_{n+1}} U^n_k\big(x_{j+1/2}, t\big). \tag{34}$$

Next we set

$$p^{n+1}_{j,k} = \frac{1}{\Delta x}\big(U^{n+1}_{j+1/2,k} - U^{n+1}_{j-1/2,k}\big). \tag{35}$$

This finishes the first part of the splitting step. As U^n was the solution of the first equation in (28), we let V^n denote the solution of the second. Precisely, for t in the interval $[t_n, t_{n+1}\rangle$ and for each j, define V^n_j as the front tracking solution of

$$\begin{gathered}\big(V^n_j\big)_t + H^\delta\big(p^{n+1}_j, \big(V^n_j\big)_y\big) = 0,\\ V^n_j(y, t_n) = V^n_{j,k-1/2} + q^n_{j,k}\big(y - y_{k-1/2}\big) \quad \text{for } y \in \big[y_{k-1/2}, y_{k+1/2}\big\rangle,\end{gathered} \tag{36}$$

where

$$\left.\begin{aligned} p^{n+1}_j(y) &= p^{n+1}_{j,k}\\ q^n_j(y) &= q^n_{j,k}\end{aligned}\right\} \quad \text{for } y \in \big[y_{k-1/2}, y_{k+1/2}\big\rangle.$$

Similarly to (34), we now can define $V^{n+1}_{j,k+1/2}$ by

$$V^{n+1}_{j,k+1/2} = \lim_{t\uparrow t_{n+1}} V^n_j\big(y_{j+1/2}, t\big). \tag{37}$$

To start the next time step, we define

$$q^{n+1}_{j,k} = \frac{1}{\Delta y}\big(V^{n+1}_{j,k+1/2} - V^{n+1}_{j,k-1/2}\big). \tag{38}$$

This process is then continued for $n = 0, 1, 2, \ldots, N - 1$, where $T = N\Delta t$. Now we have two approximations to the solution of (27), namely $U^n_{j+1/2,k}$ and $V^n_{j,k+1/2}$. Note that these are defined on spatial grids which are staggered with respect to each other. In pseudocode this algorithm is shown in Algorithm 3. We can define the final approximation by nearest-neighbor linear interpolation between these two grids to the grid defined by the points (x_j, y_k). This corresponds to using the update formula

$$u^{n+1}_{j,k} = \frac{1}{4}\big(U^{n+1}_{j+1/2,k} + U^{n+1}_{j-1/2,k} + V^{n+1}_{j,k+1/2} + V^{n+1}_{j,k-1/2}\big), \tag{39}$$

if $\Delta x = \Delta y$. If we are primarily interested in the solution at $T = t$, this update formula does not have to be used at each time step, and we can merely interpolate at the end of the splitting process where $n + 1 = N$.

A variant of this method is to update u^{n+1} *before* setting p^{n+1} and q^{n+1} to be used in the next time step. These are then defined by

$$\begin{aligned} p^{n+1}_{j,k} &= \frac{1}{\Delta x}\big(U^{n+1}_{j+1/2,k} - U^{n+1}_{j-1/2,k}\big),\\ q^{n+1}_{j,k} &= \frac{1}{\Delta y}\big(V^{n+1}_{j,k+1/2} - V^{n+1}_{j,k-1/2}\big),\end{aligned} \tag{40}$$

where we now set $U^n_{j+1/2,k}$ and $V^n_{j,k+1/2}$ to be the values of the approximation $u^n_{j,k}$ found by

ALGORITHM 3 (Dimensional Splitting/Front Tracking Algorithm Hamilton–Jacobi Equations).

```
function $[U, V]$=dimsplit($U$,$V$,$x$,$y$,$T$,$\Delta t$,$\delta$)
% $U$,$V$: the initial values, cf.,(29), (30).
% $(x_j, y_k)$: the grid points, $1 \le j \le N$, $1 \le k \le M$.
% $\Delta t$: the time step.
%
  $t = 0$
  while $t < T$,
    for $j = 1 : N$,
      for $i = 1 : M$,
        $a_i = \left(V_{j,i+1/2} - V_{j,i-1/2}\right)/\Delta y$
        $u_i = U_{j+1/2,j}$
      end for
      $U_{\cdot,j}$ = grid_fthj1d$\left(u,a,x_{\cdot+1/2},\Delta t,\delta\right)$
    end for
    for $k = 1 : M$,
      for $i = 1 : N$,
        $a_i = \left(U_{k+1/2,i} - U_{k-1/2,i}\right)/\Delta x$
        $u_i = V_{k,i+1/2}$
      end for
      $V_{k,\cdot}$ = grid_fthj1d$\left(u,a,y_{\cdot+1/2},\Delta t,\delta\right)$
    end for
    $t = t + \Delta t$
  end while
```

nearest-neighbor linear interpolation, i.e.,

$$\begin{aligned} U^n_{j+1/2,k} &= \frac{1}{2}\left(u^n_{j,k} + u^n_{j+1,k}\right), \\ V^n_{j,k+1/2} &= \frac{1}{2}\left(u^n_{j,k} + u^n_{j,k+1}\right). \end{aligned} \tag{41}$$

We remark that this interpolation is quite simplistic and can probably be improved upon. We call this method dimensional splitting with restarting, and it is summarized in Algorithm 4. In this algorithm, the interpolation steps can be done in a variety of ways. For simplicity, we have used a linear interpolation between nearest neighbors (cf., (40)), but it would probably be better to use a (W)ENO-type interpolation.

3.3.2. A direct method. Rather than solve the p equation and the q equation sequentially, we can solve both for U^n_k and V^n_j using the values from the previous time step as coefficients. This we call a direct method. The initial values are defined as before, (31), (32). For $t \in [t_n, t_{n+1}\rangle$, we define U^n_k and V^n_j to be the front tracking solutions of

$$\begin{aligned} \left(U^n_k\right)_t + H^\delta\left(\left(U^n_k\right)_x, q^n_k\right) &= 0, \\ U^n_k(x, t_n) = U^n_{j-1/2,k} + p^n_{j,k}\left(x - x_{j-1/2}\right) &\quad \text{for } x \in \left[x_{j-1/2}, x_{j+1/2}\right\rangle, \end{aligned} \tag{42}$$

ALGORITHM 4 (Dimensional Splitting/Front Tracking with Restarting).

```
function $u$=dimsplit_restart($u$,$x$,$y$,$T$,$\Delta t$,$\delta$)
%   $u$:  the initial values at $(x_j, y_k)$.
%   $(x_j, y_k)$:  the grid points, $1 \le j \le N$, $1 \le k \le M$.
%   $\Delta t$:  the time step.
%
    $t = 0$
  while $t < T$,
    Find $U$ and $V$ from $u$ by interpolation, cf., (41)
    for $j = 1 : N$,
      for $i = 1 : M$,
        $a_i = (V_{j,i+1/2} - V_{j,i-1/2})/\Delta y$
        $v_i = U_{i+1/2,j}$
      end for
      $U_{\cdot,j}$ = grid_fthj1d($v$,$a$,$x_{\cdot+1/2}$,$\Delta t$,$\delta$)
    end for
    for $k = 1 : M$,
      for $i = 1 : N$,
        $a_i = (U_{k+1/2,i} - U_{k-1/2,i})/\Delta x$
        $v_i = V_{k,i+1/2}$
      end for
      $V_{k,\cdot}$ = grid_fthj1d($v$,$a$,$y_{\cdot+1/2}$,$\Delta t$,$\delta$)
    end for
    Find $u$ from $U$ and $V$ by interpolation, cf., (39)
    $t = t + \Delta t$
  end while
```

$$\begin{aligned} &\left(V_j^n\right)_t + H^\delta\left(p_j^n, \left(V_j^n\right)_y\right) = 0, \\ &V_j^n(y, t_n) = V_{j,k-1/2}^n + q_{j,k}^n\left(y - y_{k-1/2}\right), \quad \text{for } y \in \left[y_{k-1/2}, y_{k+1/2}\right\rangle. \end{aligned} \tag{43}$$

We can use either U^n or V^n as an approximation to u, or use the interpolation defined by (39). We can define p^{n+1} and q^{n+1} using (40). This method is then called a direct method with restarting.

Remark. The reader should be cautioned that in order to keep the presentation simple, our notation is somewhat misleading. The functions denoted "H^δ" in, e.g., (43) and (42) are *not* the same function! But rather two different piecewise linear approximations of H. Remember that when doing front tracking for, e.g., (43), we use a piecewise linear (in q) and piecewise constant (in x) approximation to $H(p(x), q)$. This approximation depends on δ, so that the distance between the interpolation points tends to zero as $\delta \to 0$, as well as on the initial values $q(x, 0)$ and the coefficients $p(x)$. Since only δ is the same for (43) and (42), H^δ in (43) and (42) are not the same, nor are they the same for different j and k. The same also applies to the dimensional splitting equations (33) and (36).

Note that none of the methods we present are monotone. This makes a convergence analysis complicated, and we have not been able to prove that the methods produce a sequence of approximate solutions that converges to the unique viscosity solution. However, the numerical experiments indicate that the approximations all converge to the viscosity solution.

4. NUMERICAL EXAMPLES

To test the above methods, we have compared them with several other methods: the Lax–Friedrichs method, the Engquist–Osher scheme, and the relaxation method by Jin and Xin [22], more precisely the method called Scheme III. This method is based on replacing (28) with the system

$$\begin{aligned} p_t + w_x &= 0,\\ q_t + w_y &= 0,\\ w_t + a(p_x + q_y) &= -\frac{1}{\varepsilon}(w - H(p,q)),\\ u_t + w &= 0,\\ w(x,0) &= H(p_0, q_0), \end{aligned} \tag{44}$$

where ε is a (very) small parameter. For the implementation of Scheme III, we followed the recipe in [22]. The Lax–Friedrichs scheme we used is given by

$$u_{j,k}^{n+1} = \frac{1}{4}\left(u_{j-1,k}^n + u_{j+1,k}^n + u_{j,k-1}^n + u_{j,k+1}^n\right) - \Delta t H\left(\frac{u_{j+1,k}^n - u_{j-1,k}^n}{2\Delta x}, \frac{u_{j,k+1}^n - u_{j,k-1}^n}{2\Delta y}\right). \tag{45}$$

Finally the Engquist–Osher scheme reads

$$\begin{aligned} p_1 &= \frac{1}{\Delta x}\left(u_{j,k}^n - u_{j-1,k}^n\right), \quad p_2 = \frac{1}{\Delta x}\left(u_{j+1,k}^n - u_{j,k}^n\right),\\ q_1 &= \frac{1}{\Delta y}\left(u_{j,k}^n - u_{j,k-1}^n\right), \quad q_2 = \frac{1}{\Delta y}\left(u_{j,k+1}^n - u_{j,k}^n\right),\\ u_{j,k}^{n+1} &= u_{i,j}^n - \Delta t\left(H(p_1,q_1) + \int_{p_1}^{p_2} \min\left(\frac{\partial H}{\partial p}(p,q_1), 0\right) dp \right.\\ &\qquad \left. + \int_{q_1}^{q_2} \min\left(\frac{\partial H}{\partial q}(p_1,q), 0\right) dq\right). \end{aligned} \tag{46}$$

Details on the implementation of front tracking for one-dimensional Hamilton–Jacobi equations can be found in [24], and for details of implementation of front tracking and dimensional splitting, see [17, 18, 44]. In the numerical examples we use all our methods: (39) and (34)–(37), as well as the method with restarting (40) and (42)–(43). Furthermore, when applicable, we used Strang splitting, i.e., we start and finish with (34) using a time step $\Delta t/2$.

In our first two examples we use the convex Hamiltonian

$$H(p,q) = \sqrt{1 + p^2 + q^2}. \tag{47}$$

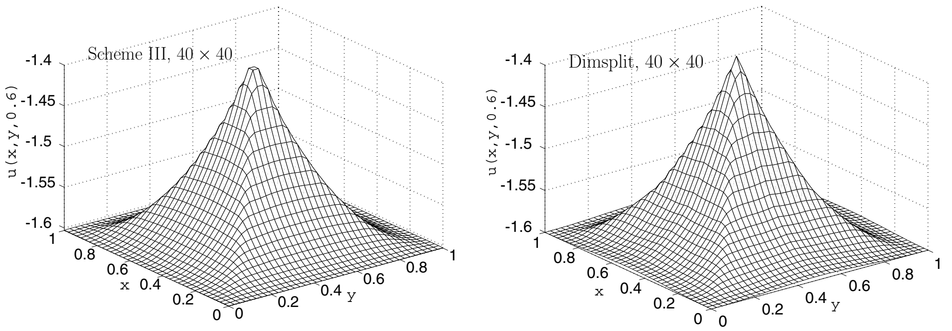

FIG. 5. (Left) Scheme III; (right) dimensional splitting.

EXAMPLE 1. Our first example is taken from [22]. The initial data is given by

$$u_0(x, y) = \frac{1}{4}(\cos(2\pi x) - 1)(\cos(2\pi y) - 1) + 1, \tag{48}$$

for x and y in the unit square $0 \leq x \leq 1, 0 \leq y \leq 1$, and we use periodic boundary data. The exact solution is unknown, and as a reference solution we used an approximation computed by the Engquist–Osher scheme with $\Delta x = \Delta y = 1/1024$. We calculated[1] the approximate solutions until $t = 0.6$, at this time the surface had moved down and a sharp peak had formed. In Fig. 5 we show solutions computed on a 50×50 grid by Scheme III and by dimensional splitting using CFL = 5. When doing dimensional splitting, both the quality of the solution and the CPU time depend on the parameter δ. In order to avoid too many parameters, we set $\delta = 2\sqrt{\min(\Delta x, \Delta y)}$.

In Table I we show the supremum errors and the CPU time (in seconds) for dimensional splitting as well as for Scheme III, the Engquist–Osher scheme, and the Lax–Friedrichs scheme on several grid sizes (indicated by N in the table). The most salient feature of this table is that the error and the CPU time for dimensional splitting seem to be independent of the CFL number. The Lax–Friedrichs scheme was very fast but produced much larger errors than the other schemes. We also remark that the CPU time does not decrease much by choosing larger CFL numbers. The reason for this is that although we take larger time steps, the number of Riemann problems to solve during each time step grows with the time step.

Figure 6 shows (the logarithms of) the supremum errors for the various methods as a function of grid size, as well as the straight lines obtained by a least-squares linear fit. From this table we see that for this problem, all the methods have numerical convergence rates between 1/2 and 1. On a given grid, the front tracking/dimensional splitting approach compares favorably with the three other schemes, and we note that the error is not very sensitive to the choice of CFL number.

EXAMPLE 2. The errors produced by dimensional splitting seem to be quite insensitive to the choice of Δt. Our next example investigates this feature closer. We use the same

[1] All computations were done on a PowerBook G3, 500 MHz, and the CPU times reported include only the computations, not the time used for initializations and memory allocations. All algorithms were coded in the "C" programming language and compiled with Metrowerks CodeWarrior 6.0 with optimization level 2.

TABLE I
$100 \times l^\infty$ Error and CPU Time for Example 1

	Dimensional splitting–front tracking											
	CFL = 5		CFL = 10		CFL = 15		Scheme III		Lax–Friedrichs		Engquist–Osher	
N	l^∞-error	Time	l^∞-error	Time	l^∞-error	Time	l^∞-error	Time	l^∞-error	Time	l^∞-error	Time
16	1.83	0.05	5.01	0.03	31.48	0.02	4.13	0.43	7.06	0.01	4.43	0.10
32	0.82	0.28	1.38	0.18	2.47	0.15	2.66	1.88	3.80	0.01	3.06	0.27
64	0.53	2.0	0.53	1.37	1.13	1.08	1.09	13.17	2.31	0.30	1.78	7.62
128	0.25	14.75	0.29	10.20	0.35	8.67	0.74	93.28	0.99	2.00	0.91	51.50
256	0.16	112.75	0.16	80.92	0.22	70.38	0.53	654.57	0.41	15.73	0.40	416.37

Hamiltonian as before (see (47)), but the initial function is now given by

$$u_0(x, y) = \begin{cases} r - 0.4, & r \le 0.4, \\ 0.4 - r, & r \ge 0.4, \end{cases} \qquad r = \sqrt{x^2 + 0.4y^2}, \tag{49}$$

and we use periodic boundary data on $[-1, 1] \times [-1, 1]$. Figure 7 shows the initial data and the approximate solution at $t = 1$ produced by dimensional splitting on a 64×64 grid with CFL number 2. Figure 8 shows contour plots of the corresponding p and q produced by the dimensional splitting algorithm on a 64×64 grid with CFL number 5. Notice that these are somewhat oscillatory in the vicinity of the singularities. Fortunately, these oscillations are not as prominent in u. To check the errors produced by dimensional splitting, we used a reference solution computed by the Engquist–Osher scheme on a 1024×1024 grid. Table II shows the l^∞ errors made by dimensional splitting on various grids with CFL numbers 2, 5, and 25. This table also shows errors produced by dimensional splitting with restarting (see (40)), as well as those produced by the Lax–Friedrichs scheme and by the Engquist–Osher scheme. We remark that the errors produced by dimensional splitting in this example were

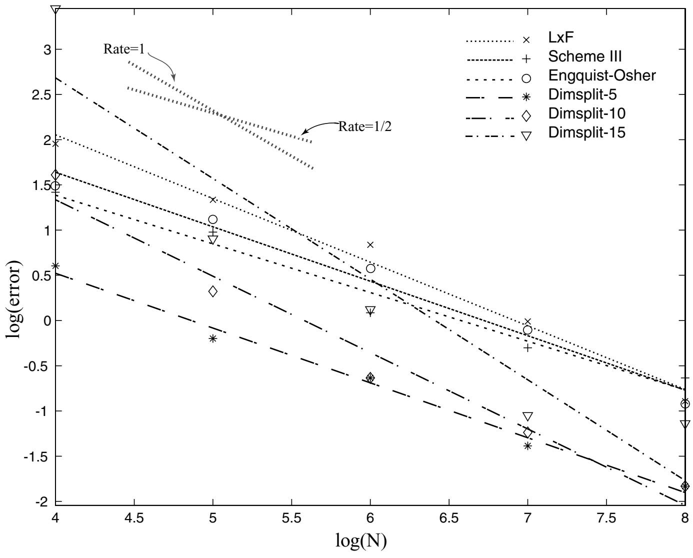

FIG. 6. A log–log plot of the supremum errors versus grid size for Example 1.

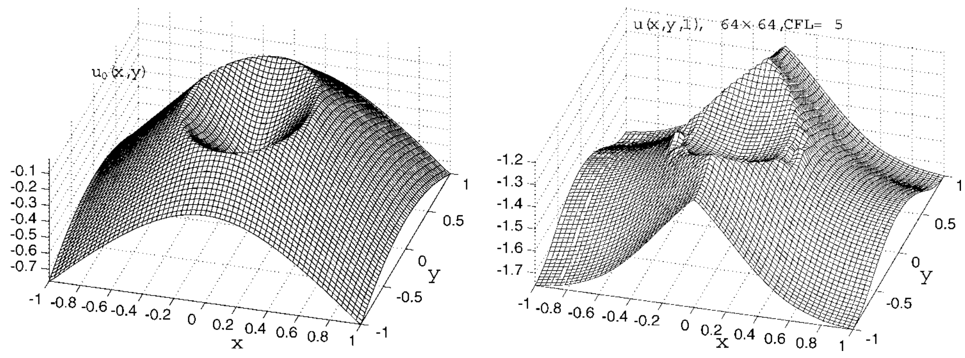

FIG. 7. Example 2. Dimensional splitting on a 64 × 64 grid with CFL = 2. (Left) $u_0(x, y)$; (right) $u(x, y, 1)$.

of roughly the same order as those produced by the Lax–Friedrichs scheme and larger than those produced by the Engquist–Osher scheme. We see, however, that the error is not very sensitive to the choice of CFL number. With CFL number 25, dimensional splitting still produces acceptable results. If we were interested in the derivatives, we would perhaps not find the accuracy in p and q quite satisfactory for this large CFL number (see Fig. 8). Perhaps the most pertinent feature of Table II is that dimensional splitting with restarting does *not* seem to converge for large CFL numbers. We think that this is caused by the simple linear interpolation we use and suspect that restarting would work better if we had used a ENO-type interpolation, i.e., avoided smearing the front in the coefficients. See [22] for a discussion of a similar phenomenon.

EXAMPLE 3. To test dimensional splitting on a nonconvex case, we chose an example taken from Osher and Shu [41]. The relevant Hamiltonian reads

$$H(p, q) = \sin(p + q). \tag{50}$$

The initial function is given by

$$u_0(x, y) = \pi(|y| - |x|). \tag{51}$$

We compute approximations in the square $[-1, 1] \times [-1, 1]$ and impose periodic boundary

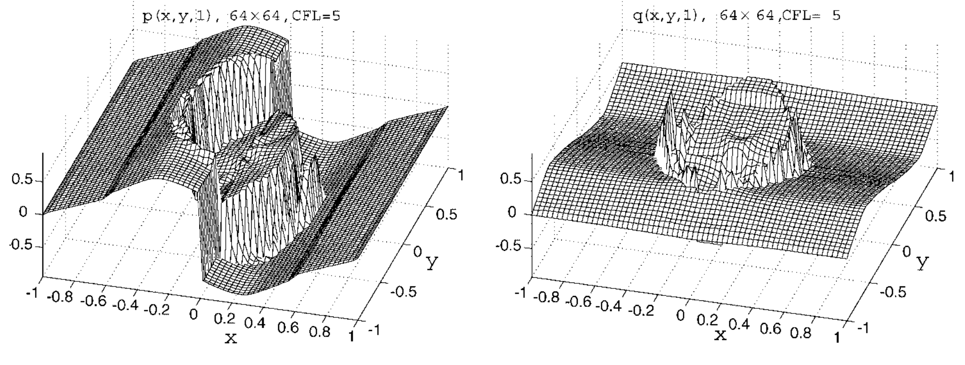

FIG. 8. Dimensional splitting on a 64 × 64 with CFL = 5. (Left) $p(x, y, 1)$; (right) $q(x, y, 1)$.

TABLE II
$100 \times l^\infty$ Error for the Initial Value Problem (49)

	CFL = 2		CFL = 5		CFL = 25			
N	Dimsplit	Restart	Dimsplit	Restart	Dimsplit	Restart	LF	EO
16	5.26	9.87	5.25	5.32	5.67	5.63	17.36	5.17
32	4.80	6.68	3.46	3.67	6.05	6.05	10.98	3.44
64	4.89	4.08	4.22	2.41	4.45	5.17	6.53	2.13
128	3.12	2.38	2.62	1.36	3.38	13.21	3.82	1.24
256	1.45	1.32	1.08	0.75	2.89	771.43	2.20	0.64
512	1.08	0.78	0.92	0.57	2.42	†	1.25	0.25

conditions. A version (without periodic boundary conditions) of this example was tested on the Lax–Friedrichs scheme and on a number of ENO-type schemes in [41]. This problem is inherently one dimensional; if we introduce coordinates (ξ, η) by $x = \xi + \eta$ and $y = \xi - \eta$, (27) and (51) reads

$$\begin{aligned} u_t + \sin(u_\xi) &= 0, \\ u(\xi, \eta, 0) &= \pi(|\xi - \eta| - |\xi + \eta|), \end{aligned} \tag{52}$$

and the periodic boundary condition says that $u(\xi + 4, \eta, t) = u(\xi, \eta, t)$ for all t. Therefore, to compute a reference solution we used one-dimensional front tracking for (52), Algorithm 2, using $\delta = 2\pi/\sqrt{1024}$ and 1024 grid points in the interval $[0, 4]$, for $\eta = 0, 4/1024, 8/1024, \ldots, 4$. In Fig. 9 we show the reference solution, as well as the approximation computed by the direct method with a CFL number 2 on a 32×32 grid. We show the results of this comparison in Table III. Our results clearly show that this example was more challenging for all the methods. Of our methods, the dimensional splitting with restarting and the direct method seem to produce the smallest errors, and these methods are comparable to the Engquist–Osher method, both of which regard errors and CPU time. We note that in this case, a CFL number of 5 was "too large" for the front tracking methods, and other experiments indicated that for nonconvex Hamiltonians, the large-step methods converged very slowly for CFL numbers larger than 2–3.

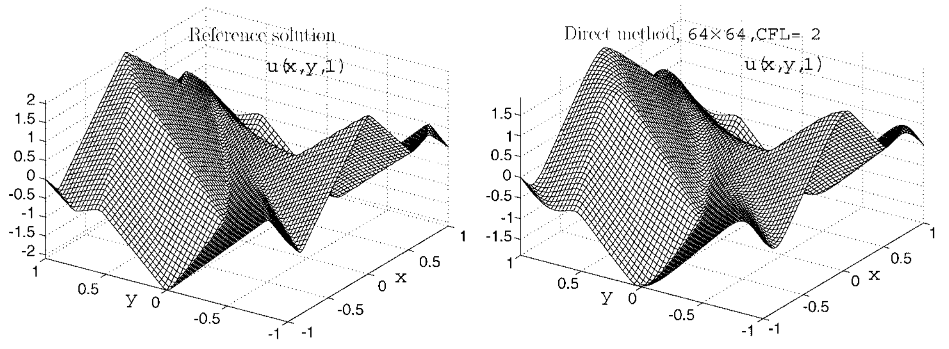

FIG. 9. (Left) The reference solution; (right) an approximation computed by the direct method.

TABLE III
$100 \times l^\infty$ Error for the Initial Value Problem (51)

	CFL = 2			CFL = 5				
N	Dimsplit	Restart	Direct	Dimsplit	Restart	Direct	LF	EO
16	131.14	70.80	63.53	68.45	58.41	64.73	125.40	74.94
32	98.06	45.11	40.63	84.65	53.66	49.90	86.01	51.71
64	88.40	29.29	26.89	85.13	45.04	51.60	58.60	34.64
128	88.36	18.96	17.66	68.89	43.82	60.95	39.72	23.16
256	85.64	12.42	13.87	67.03	46.05	51.97	26.76	15.44

5. CONCLUSIONS

We have devised and implemented a family of numerical methods for solving the initial value problem for the Hamilton–Jacobi equation

$$u_t + H(D_x u) = 0.$$

The methods are all based on solving the d conservation laws (with discontinuous coefficients)

$$\begin{gathered}(p_1)_t + H(p_1, p_2, \dots, p_d)_{x_1} = 0\\ \ddots\\ (p_d)_t + H(p_1, p_2, \dots, p_d)_{x_d} = 0\end{gathered}$$

using a front tracking method. This can be done sequentially, in which case we label the method dimensional splitting, or "in parallel," i.e., use of the same coefficients for all equations. The pertinent feature of our methods is that there is no intrinsic CFL condition associated with the time step, so we can choose our time step independently of other parameters.

We found that these method all produce results comparable to standard methods. We have not been able to show theoretical convergence of these types of methods, except in the (trivial) one-dimensional case (see [24]), but our examples indicate that the methods all converge to the viscosity solutions. Moreover, the errors were found to be largely independent of the CFL number, something also found for dimensional splitting for scalar conservation laws (see [34]).

The numerical methods developed herein can be easily extended to yield large time-step methods for Hamilton–Jacobi equations with a zeroth order term

$$u_t + H(D_x u) = G(x, t, u)$$

by solving sequentially the equations

$$u_t + H(D_x u) = 0 \quad \text{and} \quad u_t = G(x, t, u)$$

using the methods presented here for the first equation. In Jakobsen *et al.* [20], it was shown that the temporal error associated with the above "source term" splitting is linear in the splitting (time) step, and as such the splitting can be used in conjunction with the methods proposed herein without loss of accuracy.

It seems to be very difficult to generalize methods based on dimensional splitting to unstructured meshes. For non-Cartesian structured meshes, e.g., curvilinear, a generalization is possible. Then one would have to solve one-dimensional Hamilton–Jacobi equations of the type

$$u_t + H(x, u_x) = 0$$

by front tracking. Here the x dependency in H comes from geometrical properties of the mesh and makes front tracking more complicated, though by no means impossible (see Lie [33]).

REFERENCES

1. R. Abgrall, Numerical discretization of the first-order Hamilton–Jacobi equation on triangular meshes, *Commun. Pure Appl. Math.* **49**(12), 1339 (1996).
2. M. Bardi, M. Crandall, L. Evans, H. Soner, and P. Souganidis, *Viscosity Solutions and Applications*, Lecture Notes in Math (Springer-Verlag, Berlin, 1997), Vol. 166.
3. T. J. Barth and J. A. Sethian, Numerical schemes for the Hamilton–Jacobi and level set equations on triangulated domains, *J. Comput. Phys.* **145**(1), 1 (1998).
4. A. Bressan, Global solutions to systems of conservation laws by wave-front tracking, *J. Math. Anal. Appl.* **170**, 414 (1992).
5. A. Bressan, T.-P. Liu, and T. Yang, *L^1 Stability Estimates for $n \times n$ Conservation Laws*, Preprint (S.I.S.S.A., Trieste, 1998).
6. L. Corrias, M. Falcone, and R. Natalini, Numerical schemes for conservation laws via Hamilton–Jacobi equations, *Math. Comp.* **64**(210), 555 (1995).
7. M. G. Crandall and P.-L. Lions, Viscosity solutions of Hamilton–Jacobi equations, *Trans. Am. Math. Soc.* **277**(1), 1 (1983).
8. M. G. Crandall and P.-L. Lions, Two approximations of solutions of Hamilton–Jacobi equations. *Math. Comp.* **43**(167), 1 (1984).
9. C. M. Dafermos, Polygonal approximation of solutions of the initial value problem for a conservation law, *J. Math. Anal. Appl.* **38**, 33 (1972).
10. S. Diehl, Scalar conservation laws with discontinuous flux functions. I. The viscous profile condition, *Commun. Math. Phys.* **176**(1), 23 (1996).
11. T. Gimse and N. H. Risebro, Riemann problems with a discontinuous flux function, in *Proc. of the Third Intern. Conf. on Hyp. Prob.* (Uppsala, Studenlitteratur, Lund, 1990), p. 488.
12. T. Gimse and N. H. Risebro, Solution of the Cauchy problem for a conservation law with a discontinuous flux function, *SIAM J. Math. Anal.* **23**(3), 635 (1992).
13. V. Haugse, K. H. Karlsen, K.-A. Lie, and J. Natvig, Numerical solution of the polymer system by front tracking, *Transport Porous Media* **44**(1), 63–83 (2001).
14. R. Holdahl, H. Holden, and K.-A. Lie, Unconditionally stable splitting methods for the shallow water equations, *BIT* **39**(3), 451 (1999).
15. H. Holden and L. Holden, On scalar conservation laws in one-dimension, in *Ideas and Methods in Mathematics and Physics*, edited by S. Albeverio, J. E. Fenstad, H. Holden, and T. Lindstrøm (Cambridge Univ. Press, Cambridge, MA, 1988), p. 480.
16. H. Holden, L. Holden, and R. Høegh-Krohn, A Numerical method for first order nonlinear scalar conservation laws in one dimension, *Comput. Math. Appl.* **15**(6–8), 595 (1988).
17. H. Holden, K.-A. Lie, and N. H. Risebro, An unconditionally stable method for the Euler equations, *J. Comput. Phys.* **150**, 76 (1999).

18. H. Holden and N. H. Risebro, A method of fractional steps for scalar conservation laws without the CFL condition, *Math. Comp.* **60**(201), 221 (1993).

19. C. Hu and C.-W. Shu, A discontinuous Galerkin finite element method for Hamilton–Jacobi equations, *SIAM J. Sci. Comput.* **21**(2), 666–690 (1999).

20. E. R. Jakobsen, K. H. Karlsen, and N. H. Risebro, *On the Convergence Rate of Operator Splitting for Hamilton–Jacobi Equations with Source Terms*, Preprint (Department of Mathematics, University of Bergen, 2000), available at http://www.math.ntnu.no/conservation/.

21. G.-S. Jiang and D. Peng, *Weighted ENO Schemes for Hamilton–Jacobi Equations*, UCLA-CAM Report 29 (1997).

22. S. Jin and Z. Xin, Numerical passage from systems of conservation laws to Hamilton–Jacobi equations, relaxation schemes, *SIAM J. Numer. Anal.* **35**(6), 2385 (electronic) (1998).

23. K. H. Karlsen, *On the Accuracy of a Dimensional Splitting Method for Scalar Conservation Laws*, Master's thesis (Department of Mathematics, University of Oslo, Norway, 1994).

24. K. H. Karlsen and N. H. Risebro, *A Note on Front Tracking and the Equivalence between Viscosity Solutions of Hamilton-Jacobi Equations and Entropy Solutions of Scalar Conservation Laws*, Preprint (Department of Mathematics, University of Bergen, 2000), available at http://www.math.ntnu.no/conservation/.

25. R. A. Klausen and N. H. Risebro, Stability of conservation laws with discontinuous coefficients, *J. Differential Equations* **157**, 41 (1999).

26. C. Klingenberg and N. H. Risebro, Stability of a resonant system of conservation laws modeling polymer flow with gravitation, *J. Differential Equations* **170**, 344–380 (2001).

27. C. Klingenberg and N. H. Risebro, Convex conservation laws with discontinuous coefficients. Existence, uniqueness and asymptotic behavior. *Commun. Partial Differential Equations* **20**(11–12), 1959 (1995).

28. G. Kossioris, C. Makridakis, and P. Souganidis, Finite volume schemes for Hamilton–Jacobi equations, *Numer. Math.* **83**(3), 427 (1999).

29. S. N. Kružkov, The Cauchy problem in the large for non-linear equations and for certain first-order quasi-linear systems with several variables, *Dokl. Akad. Nauk SSSR* **155**, 743 (1964).

30. S. N. Kružkov, The method of finite differences for a nonlinear equation of the first order with several independent variables, *Ž. Vyčisl. Mat. Mat. Fiz.* **6**, 884 (1966).

31. S. N. Kružkov, First order quasi-linear equations in several independent variables, *Math. USSR Sbornik* **10**(2), 217 (1970).

32. A. Kurganov and E. Tadmor, *New High-Resolution Semi-Discrete Central Schemes for Hamilton–Jacobi Equations*, UCLA-CAM Report 24 (1999).

33. K.-A. Lie, A front tracking method for one-dimensional nonlinear advection equations with variable coefficients, *Numer. Algorithms* **24**, 275 (2000).

34. K.-A. Lie, V. Haugse, and K. H. Karlsen, Dimensional splitting with front tracking and adaptive grid refinement, *Numer. Methods Partial Differential Equations* **14**(5), 627 (1998).

35. C.-T. Lin and E. Tadmor, *High-Resolution Non-Oscillatory Central Scheme for Hamilton–Jacobi Equations*, UCLA-CAM Report 38 (1998).

36. P.-L. Lions, *Generalized Solutions of Hamilton–Jacobi Equations* (Pitman (Advanced Publishing Program), Boston, 1982).

37. P.-L. Lions and P. E. Souganidis, Convergence of MUSCL and filtered schemes for scalar conservation laws and Hamilton–Jacobi equations, *Numer. Math.* **69**(4), 441 (1995).

38. B. J. Lucier, A moving mesh numerical method for hyperbolic conservation laws, *Math. Comp.* **46**(173), 59 (1986).

39. O. A. Oleĭnik, Discontinuous solutions of non-linear differential equations, *Am. Math. Soc. Transl. Ser. 2* **26**, 95 (1963).

40. S. Osher and J. A. Sethian, Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations, *J. Comput. Phys.* **79**(1), 12 (1988).

41. S. Osher and C.-W. Shu, High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations, *SIAM J. Numer. Anal.* **28**(4), 907 (1991).

42. D. Ostrov, *Solutions of Hamilton–Jacobi Equations and Scalar Conservation Laws with Discontinuous Space-Time Dependence*, Preprint (2000).
43. N. H. Risebro, A front-tracking alternative to the random choice method, *Proc. Amer. Math. Soc.* **117**(4), 1125 (1993).
44. N. H. Risebro and A. Tveito, Front tracking applied to a nonstrictly hyperbolic system of conservation laws, *SIAM J. Sci. Stat. Comput.* **12**(6), 1401 (1991).
45. N. H. Risebro and A. Tveito, A front tracking method for conservation laws in one dimension, *J. Comp. Phys.* **101**(1), 130 (1992).
46. B. Temple, Global solution of the Cauchy problem for a class of 2×2 nonstrictly hyperbolic conservation laws, *Adv. Appl. Math.* **3**(3), 335 (1982).
47. J. Towers, *Convergence of a Difference Scheme for Conservation Laws with a Discontinuous Flux*, Preprint, available at the URL http://www.math.ntnu.no/conservation/.