COMMUN. MATH. SCI. © 2008 International Press
Vol. 6, No. 1, pp. 199–216

A FAST ALGORITHM FOR SOLVING FIRST-ORDER PDES BY L^1-MINIMIZATION[*]

JEAN-LUC GUERMOND[†], FABIEN MARPEAU[‡], AND BOJAN POPOV[§]

Abstract. In this paper, we state a convergence result for an L^1-based finite element approximation technique in one dimension. The proof of this result is constructive and provides the basis for an algorithm for computing L^1-based almost minimizers with optimal complexity. Several numerical results are presented to illustrate the performance of the method.

Key words. finite elements, best L^1-approximation, viscosity solution, transport, ill-posed problem, HJ equation, eikonal equation

AMS subject classifications. 65N35, 65N22, 65F05, 35J05

1. Introduction

This paper is concerned with the approximation of first-order PDEs using finite element–based best L^1-approximations. This type of approximation technique has been introduced by Lavery [13, 14] and further explored in Guermond [8]. Numerical tests reported in these references suggest that L^1-based minimization techniques can compute the viscosity solutions of some first-order PDEs. This fact has been proved in one space dimension for linear first-order PDEs equipped with ill-posed boundary conditions in Lavery [14] and Guermond and Popov [11]. The proofs in the two above references are quite technical and rely essentially on explicit computations of the minimizers. The technicalities therein are such that it is difficult to really understand from these two proofs why the L^1-minimizer performs so well. The first objective of the present work is to revisit [11] and to give a very simple proof of the above statement. The key argument is to show that the L^1-minimizer selects the up-wind solution. To the best of our knowledge, the present paper is the first showing that L^1-minimization techniques automatically introduce up-winding on linear transport problems. Based on the constructive argument from the new proof, the second objective of the paper is to propose a fast algorithm for computing L^1-minimizers. This algorithm involves $\mathcal{O}(N)$ operations where N is the number of degrees of freedom.

The paper is organized as follows. In Section 2 we revisit the one-dimensional ill-posed model problem considered in Guermond and Popov [11], and we give an elementary proof of the fact that L^1-minimizers converge to the unique viscosity solution of this problem. The key to this result is that local L^1-minimization selects the up-wind information as proved in Lemma 2.1. Based on the local minimization argument unveiled in Lemma 2.1, we construct in Section 3 a fast algorithm for solving the L^1-minimization problem associated with the one-dimensional ill-posed model problem. The algorithm is tested on an ill-posed problem and on a transport problem with discontinuous velocity. In Section 4 we generalize the algorithm to nonlinear one-dimensional first-order PDEs. We essentially focus our attention on stationary

[*]Recieved: June 19, 2007; accepted (in revised version): December 11, 2007. Communicated by François Bouchut. This material is based upon work supported by the National Science Foundation grant DMS-0510650.

[†]Department of Mathematics, Texas A&M University 3368 TAMU, College Station, TX 77843, USA. On leave from LIMSI, UPRR 3251 CNRS, BP 133, 91403 Orsay cedex, France.

[‡]Department of Mathematics, University of Houston, 651 PGH Houston, TX 77004, USA.

[§]Department of Mathematics, Texas A&M University 3368 TAMU, College Station, TX 77843, USA.

Hamilton-Jacobi equations. The algorithm is illustrated on various nonlinear test cases. Concluding remarks are reported in Section 5.

2. The one-dimensional linear model problem

In this section we restrict ourselves to a model one-dimensional differential equation equipped with a set of ill-posed boundary conditions which has been considered in [11]. Lemma 2.1, which is the main result of this section, will be the basis for the algorithm developed in Section 3.

2.1. The continuous problem. Let $\Omega=(0,1)$, $f\in L^1(\Omega)$, $\beta\in\mathcal{C}^1(\overline{\Omega})$, and solve for the unique viscosity solution $u\in W^{1,1}(\Omega)$ of

$$\begin{cases} u(x)+\beta(x)u'(x)=f(x) & \text{in } \Omega,\\ u(0)=0, \quad u(1)=0. \end{cases} \tag{2.1}$$

The boundary conditions are to be understood in the entropy sense as defined by Bardos–le Roux–Nédélec, [1]. We recall that the viscosity solution to (2.1) is obtained by regularizing the PDE by adding $-\epsilon u''$, i.e., it is the limit as $\epsilon\to 0$ of the sequence $(u_\epsilon)_{\epsilon>0}$ defined by

$$\begin{cases} u_\epsilon(x)+\beta(x)u_\epsilon'(x)-\epsilon u_\epsilon''(x)=f(x) & \text{in } \Omega,\\ u_\epsilon(0)=0, \quad u_\epsilon(1)=0. \end{cases} \tag{2.2}$$

Despite its appearance, the problem (2.1) is not purely formal. It arises when one tries to approximate (2.2) on meshes that are not refined enough. For instance, consider a mesh of typical size h and assume that $\epsilon/h^2\ll\|\beta\|_{L^\infty}/h$ (i.e., the mesh is not fine enough to resolve boundary layers). Then the discrete counterpart of the second-order term is dominated by the first-order one and the discrete system does not really see the diffusion $-\epsilon u_\epsilon''(x)$. Approximating (2.2) in these circumstances amounts to trying to solve (2.1) with the boundary conditions understood in the classical sense instead of the entropy sense.

To avoid unnecessary technicalities we further assume that

$$0<\inf_{x\in\Omega}\beta(x), \tag{2.3}$$

$$\sup_{x\in\Omega}\beta'(x)<1. \tag{2.4}$$

The condition $\beta'\le 1$ is the one-dimensional counterpart of the condition $\nabla\cdot\beta\le 1$, which is standard for the multidimensional version of (2.1). The assumption (2.3) implies that the flow associated with β has characteristics flowing from left to right. This in turns implies that the viscosity solution satisfies only the boundary condition $u(0)=0$; the other boundary condition is discarded. The uniqueness of a viscosity solution to (2.1) in $W^{1,1}(\Omega)$ is well known even under weaker assumptions on β and f. To simplify the notation we define the linear operator

$$L: W^{1,1}(\Omega)\ni v\longmapsto v+\beta v'\in L^1(\Omega). \tag{2.5}$$

2.2. The discrete problem. Let $\mathcal{T}_h=\cup_{i=0}^n I_i$ be a mesh of Ω composed of $n+1$ cells I_i, $i=0,\dots,n$. Let $x_0,x_1,\dots x_{n+1}$ be the vertices of this mesh and assume that the enumeration is such that $x_0=0$, $x_{n+1}=1$ and each cell I_i is defined by $I_i=[x_i,x_{i+1}]$. The midpoint of each cell is denoted by $x_{i+\frac12}=\frac{x_{i+1}+x_i}{2}$. We set $h_i=x_{i+1}-x_i>0$, $i=0,1,\dots,n$, and we define $h=\max_i h_i$.

To construct an approximation to (2.1), we introduce the approximation space

$$X_h = \{v_h \in \mathcal{C}^0(\overline{\Omega});\ v_h|_{I_i} \in \mathbb{P}_1, \forall I_i \in \mathcal{T}_h;\ v_h(0) = v_h(1) = 0\}, \tag{2.6}$$

where $\mathbb{P}_1$ denotes the set of polynomials of degree at most one. Note that the functions in X_h are zero at both ends of the interval Ω; i.e., both boundary conditions in (2.1) are enforced. Upon defining the functional

$$J(v_h) = \int_0^1 |L(u_h)(x) - f(x)|\,dx, \tag{2.7}$$

we consider the following finite element L^1-minimization problem: Seek $u_h \in X_h$ such that

$$J(u_h) = \min_{v_h \in X_h} J(v_h). \tag{2.8}$$

To simplify things a little bit more, we use the midpoint rule to approximate the integral over each mesh cell. We then replace the functional J by the functional

$$J_h(v_h) := \sum_{i=0}^{n} h_i \left| L(v_h)(x_{i+\frac{1}{2}}) - f_i \right|, \tag{2.9}$$

where we have set $f_i := h_i^{-1} \int_{I_i} f(x)dx$ and $\beta_i := \beta(x_{i+\frac{1}{2}})$. Similarly, for every $v \in X_h$, we denote $v_i := v(x_i)$. Problem (2.8) is then replaced by the following one: Seek $u_h \in X_h$ such that

$$J_h(u_h) = \min_{v_h \in X_h} J_h(v_h). \tag{2.10}$$

It is shown in [11] that the sequence $(u_h)_{h>0}$ solving (2.10) converges to the viscosity solution of (2.2) in $W^{1,1}_{\text{loc}}[0,1)$, and the rate of convergence is $\mathcal{O}(h)$ if f is in $BV(\Omega)$. We want now to offer a new proof of this fact which is significantly simpler than that in [11].

2.3. Convergence analysis. We start with a definition. With each cell I_i we associate the residual over that cell as follows:

$$r_i(r,s) = h_i\left(\tfrac{1}{2}(r+s) + \beta_i h_i^{-1}(s-r) - f_i\right), \tag{2.11}$$

so that by setting $R_i(v) = r_i(v_i, v_{i+1})$ for all $v \in X_h$, we have

$$J_h(v) = \sum_{i=0}^{n} |R_i(v)|. \tag{2.12}$$

Let i be an arbitrary integer in $\{0,\dots,n\}$. Define the maps $t_{i,l}$, $l \in \{i, i+1\}$ so that for every $r \in \mathbb{R}$, $t_{i,i}(r)$ and $t_{i,i+1}(r)$ are the unique real numbers solving

$$r_i(r, t_{i,i}(r)) = 0, \qquad r_i(t_{i,i+1}(r), r) = 0. \tag{2.13}$$

The role of these maps is clarified by the following lemma.

Lemma 2.1. *Assume* (2.3),(2.4). *Then, there exists* $h_0 := 2\inf_{x\in\Omega} \beta(x)$ *so that for all* $h < h_0$, *for all* $i \in \{1,\dots n\}$, *and for all* $r,s \in \mathbb{R}$

$$|r_i(t_{i-1,i-1}(r), s)| = \min_{z \in \mathbb{R}} [|r_{i-1}(r,z)| + |r_i(z,s)|], \tag{2.14}$$

and the minimum is strict if $r_i(t_{i-1,i-1}(r),s)\neq 0$.

Proof. Define $J_{i-1,i}(z)=|r_{i-1}(r,z)|+|r_i(z,s)|$. The graph of $J_{i-1,i}(z)$ is convex and composed of three linear branches (see Figure 2.1). The functional reaches its minimum at one of the two angular points of the graph, say z_- and z_+ where z_- and z_+ are defined so that $r_{i-1}(r,z_-)=0$ and $r_i(z_+,s)=0$. Note that $z_-=t_{i-1,i-1}(r)$ and $z_+=t_{i,i+1}(s)$. Let us set

$$\omega_{i-1}=\tfrac{1}{2}+\beta_{i-1}h_{i-1}^{-1}, \qquad \omega_i=\tfrac{1}{2}+\beta_i h_i^{-1},$$
$$\omega'_{i-1}=\tfrac{1}{2}-\beta_{i-1}h_{i-1}^{-1}, \qquad \omega'_i=\tfrac{1}{2}-\beta_i h_i^{-1}.$$

With this set of notation we rewrite $r_{i-1}(r,z)=h_{i-1}(\omega_{i-1}z+\omega'_{i-1}r-f_{i-1})$ and $r_i(z,s)=h_i(\omega_i s+\omega'_i z-f_i)$, which implies

$$z_-=\omega_{i-1}^{-1}(f_{i-1}-r\omega'_{i-1}), \qquad z_+=\omega_i'^{-1}(f_i-s\omega_i).$$

To determine whether the minimum of $J_{i-1,i}$ is $J_{i-1,i}(z_-)$ or $J_{i-1,i}(z_+)$, we must compare $J_{i-1,i}(z_-)=|r_i(z_-,s)|$ with $J_{i-1,i}(z_+)=|r_{i-1}(r,z_+)|$. Using the above definitions, we infer

$$r_{i-1}(r,z_+)=h_{i-1}\omega_i'^{-1}((f_i-s\omega_i)\omega_{i-1}+\omega'_i\omega'_{i-1}r-\omega'_i f_{i-1}),$$
$$r_i(z_-,s)=h_i\omega_{i-1}^{-1}((f_{i-1}-r\omega'_{i-1})\omega'_i+\omega_{i-1}\omega_i s-\omega_{i-1}f_i).$$

If $\omega_{i-1}\omega_i s-\omega'_{i-1}\omega'_i r+f_{i-1}\omega'_i-\omega_{i-1}f_i=0$, then $r_{i-1}(r,z_+)=0=r_i(z_-,s)$ and

$$\min_{z\in\mathbb{R}} J_{i-1,i}(z)=\min(|r_{i-1}(r,z_+)|,|r_i(z_-,s)|)=0\geq |r_i(z_-,s)|,$$

thus proving the claim. Note in passing that $\omega_{i-1}\omega_i s-\omega'_{i-1}\omega'_i r+f_{i-1}\omega'_i-\omega_{i-1}f_i=0$ if and only if $z_+=z_-$, and in this case the functional $J_{i-1,i}(z)$ has only two branches (see Figure 2.1) and at the minimum $r_{i-1}(r,z_+)=0=r_i(z_-,s)$. If $\omega_{i-1}\omega_i s-\omega'_{i-1}\omega'_i r+f_{i-1}\omega'_i-\omega_{i-1}f_i\neq 0$ (or equivalently $z_+\neq z_-$), we then infer

$$|r_{i-1}(r,z_+)|=h_{i-1}|\omega'_i|^{-1}h_i^{-1}\omega_{i-1}|r_i(z_-,s)|,$$

and we have to examine the ratio $h_{i-1}\omega_{i-1}/(h_i|\omega'_i|)$. Observe first that (2.3) implies that if $h<h_0$, then ω'_i is negative. Then the above ratio is larger than 1 if we can establish that $h_{i-1}\omega_{i-1}+h_i\omega'_i$ is positive. The above definitions together with the one-sided bound (2.4) yield

$$h_{i-1}\omega_{i-1}+h_i\omega'_i=\tfrac{1}{2}h_{i-1}+\tfrac{1}{2}h_i+\beta_{i-1}-\beta_i=\tfrac{1}{2}(h_{i-1}+h_i)-\int_{x_{i-1/2}}^{x_{i+\frac{1}{2}}}\beta'(x)dx$$
$$\geq \tfrac{1}{2}(1-\sup_{x\in\Omega}\beta'(x))(h_{i-1}+h_i)>0.$$

This immediately implies $|r_{i-1}(r,z_+)|>|r_i(z_-,s)|$, thus confirming the claim.

Assume now that $r_i(z_-,s)\neq 0$. Then $\omega_{i-1}\omega_i s-\omega'_{i-1}\omega'_i r+f_{i-1}\omega'_i-\omega_{i-1}f_i\neq 0$, and the above argument implies that

$$J_{i-1,i}(z_-)=|r_i(z_-,s)|<|r_{i-1}(r,z_+)|=J_{i-1,i}(z_+),$$

i.e., the graph of $J_{i-1,i}$ is strictly monotone on the interval $[z_-,z_+]$. It is clear also that the graph of $J_{i-1,i}$ is strictly monotone on the two other branches that go to $-\infty$ and $+\infty$. As a result the minimum of $J_{i-1,i}$ at z_- is strict. ∎

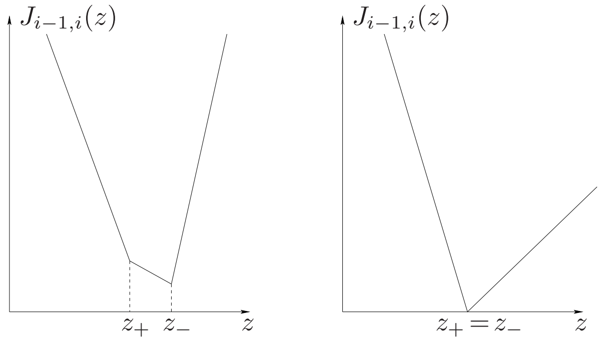


FIGURE 2.1. *Notation for the proof of Lemma 2.1*

REMARK 2.1. *The conditions* (2.3)–(2.4) *on the field β imply that the flow associated with β has characteristics flowing from left to right. It is remarkable that the solution to the minimization problem* (2.14) *is obtained by enforcing the upwind residual to be zero. In some sense, the local L^1-minimizer naturally selects the upwind information.*

Let us now construct $u_h \in X_h$ as follows:

$$\begin{cases} u_h(0) = 0, \\ u_h(x_i) = t_{i-1,i-1}(u_h(x_{i-1})), \qquad 1 \le i \le n, \\ u_h(x_{n+1}) = 0. \end{cases} \tag{2.15}$$

In other words, u_h is the unique element of X_h such that $R_i(u_h) = 0$ for all $i = 0,\dots,n-1$. The following holds:

THEOREM 2.2. *Let $h < 2\inf_{x\in\Omega}\beta(x)$ and u_h be defined by* (2.15)*. Then, u_h uniquely solves the minimization problem* (2.10)*.*

Proof. Let v an arbitrary member of X_h. Assume that v is different from u_h. Since (2.15) uniquely defines u_h, there is $i \in \{1,\dots,n\}$ such that $v(x_i) \neq t_{i-1,i-1}(v(x_{i-1}))$, i.e., $r_{i-1}(v_{i-1},v_i) \neq 0$. Let us define $\tilde{v} \in X_h$ so that $\tilde{v}_l = v_l$ for all $l \neq i$ and set $\tilde{v}_i = t_{i-1,i-1}(v_{i-1})$. Then observe that

$$J_h(v) - J_h(\tilde{v}) = |r_{i-1}(v_{i-1},v_i)| + |r_i(v_i,v_{i+1})| - |r_i(\tilde{v}_i,v_{i+1})|.$$

If $r_i(\tilde{v}_i,v_{i+1}) = 0$, then $J_h(v) - J_h(\tilde{v}) \ge |r_{i-1}(v_{i-1},v_i)| > 0$. If $r_i(\tilde{v}_i,v_{i+1}) \neq 0$, then Lemma 2.1 implies $J_h(v) > J_h(\tilde{v})$. In both cases, v is not a minimizer of (2.10). The result is proved. □

For completeness, let us finally recall the following result:

THEOREM 2.3 (Convergence). *Let u be the viscosity solution to* (2.1)*. Let u_h solve* (2.10)*. Then $\lim_{h\to 0}\|u-u_h\|_{W^{1,1}(0,x_n)} = 0$ for all f in $L^1(\Omega)$, and there is c independent of h such that*

$$\|u-u_h\|_{W^{1,1}(0,x_n)} \le ch\|f\|_{\mathrm{BV}[0,1]}$$

for all $f \in \mathrm{BV}[0,1]$.

Proof. See Guermond and Popov [11, Thoerem 7]. □

2.4. Sparsity and L^1 versus L^2. Let us interpret the result of Theorem 2.2 and put it in perspective. Observe that the functional J_h as defined in (2.9) is the sum of $n+1$ residuals. According to (2.15), one striking property of the L^1-minimizer (as defined in (2.10)) is that the residuals $r_0(u_0,u_1),\dots,r_{n-1}(u_{n-1},u_n)$ are zero. That is, among the $n+1$ residuals composing J_h, the L^1-minimizer sets n of those to zero. The residual vector $(r_0(u_0,u_1),\dots,r_{n-1}(u_{n-1},u_n),r_n(u_n,u_{n+1}))$ is extremely sparse since only the $(n+1)$-th entry, $r_n(u_n,u_{n+1})$, is non zero. This sparsity property has been recognized by Donoho [6, 5] in a more general context and can be used to recover signals from incomplete and inaccurate measurements, *cf.* Candès, Tao [3], Candès, Romberg, Tao [2]. If instead of computing the L^1-minimizer we compute the L^2-minimizer (say by minimizing $K_h(v)=\sum_{i=0}^{n} h_i^{-1} r_i(v_i,v_{i+1})^2)$, then it is a general fact that the L^2-minimizer yields a dense residual vector, i.e., none of the residual is zero in general. In particular, if there should be a sharp boundary layer in one particular cell (say the last one for instance), then instead of committing a large error in this cell (which the L^1-minimizer would do), the L^2-minimizer spreads out the error over all the cells.

Let us finally recall also that, as proved in Guermond [8], the Least Squares approximation to the ill-posed problem (2.1) does not converge to the viscosity solution in general (see also Section 3.3.1 for a counter-example).

3. A fast algorithm for solving (2.10) in the linear case

The objective of this section is to show how the result of Lemma 2.1 can be used to construct a fast algorithm for solving (2.10).

3.1. Short review. The main difficulty we encounter for solving (2.10) is that this is a linear programming problem. To see this we define the $(n+1)\times n$ matrix A and the vector $b\in\mathbb{R}^{n+1}$ so that for all $k\in\{1,\dots,n+1\}$, $l\in\{1,\dots,n\}$

$$b_k := h_{k-1} f_{k-1} \qquad \text{and} \qquad A_{kl} := \begin{cases} h_{k-1} w'_{k-1} & \text{If } l=k-1, \\ h_{k-1} w_{k-1} & \text{If } l=k, \\ 0 & \text{Otherwise.} \end{cases} \tag{3.1}$$

Then (2.10) can be recast as follows: Seek $u\in\mathbb{R}^{n+1}$ and $x\in\mathbb{R}^n$ so that

$$(u,x) \longleftarrow \min_{\substack{v\in\mathbb{R}^{n+1}\\ y\in\mathbb{R}^n}} \sum_{i=1}^{n} v_i \quad \text{subject to} \quad \begin{cases} Ay-b-v\le 0, \\ -Ay+b-v\le 0. \end{cases} \tag{3.2}$$

This problem can be solved by the simplex method, but since the late 1980's more efficient methods, collectively known as interior point methods, have been developed, see e.g., Nocedal, Wright [17] for a review. This type of technique is used by Yong, Shu-Cherng, and Lavery [18] to solve large scale multi-variate L^1-spline interpolation problems. This is also the approach used by Candès, Tao, Romberg [2] to solve signal recovery problems using ℓ^1-minimization.

Another technique used in Guermond [8] consists of regularizing the absolute value function $x\longmapsto |x|$ by $\psi_\varepsilon(x)=\frac{x^2}{|x|+\varepsilon}$, where $\varepsilon>0$. Then upon introducing the regularized functional

$$\mathcal{J}_\varepsilon(v_h)=\sum_{i=0}^{n} h_i \psi_\varepsilon(L_h(v_h)(x_{1+\frac12})-f(x_{1+\frac12})), \tag{3.3}$$

problem (2.10) is replaced by the following: Seek u_h^ε so that

$$\mathcal{J}_\varepsilon(u_h^\varepsilon) = \min_{v_h \in E_h} \mathcal{J}_\varepsilon(v_h). \tag{3.4}$$

Owing to the regularization, $\mathcal{J}_\varepsilon$ is twice differentiable (in the Fréchet sense), and the first-order optimality condition for (3.4) is

$$\sum_{i=0}^{n} h_i D\psi_\varepsilon(L_h(u_h^\varepsilon)(x_{1+\frac{1}{2}}) - f(x_{1+\frac{1}{2}})) L_h(v_h)(x_{1+\frac{1}{2}}) = 0, \quad \forall v_h \in E_h. \tag{3.5}$$

The algorithm described in Guermond [8] consists of solving (3.5) using Newton's method and iteratively driving the parameter ε to zero.

The main drawback we see in the above two methods (interior point methods and regularization) is that they are all based on Newton iterations requiring solving large symmetric linear systems. Although efficient algorithms like conjugate gradient can be applied, the overall complexity does not scale linearly with the size of the system, since the matrices change at each Newton iteration and thus are difficult to precondition efficiently. A possible way out could be to use multigrid preconditioning, but we have not explored this path further.

One possible source of the difficulty mentioned above is that by looking at the algebraic problem (3.1), one loses sight of the PDE origin of the problem, and one does not use the hyperbolicity which is revealed by Theorem 2.2. We explore this venue in the rest of the paper.

3.2. Definition of the algorithm. In view of Lemma 2.1, we propose the following algorithm for solving (2.10):

Algorithm 1 L^1-minimization for (2.10).

```
1: Initialize $v(0:n+1) \leftarrow 0$; initialize array visited(1:n) ← false
2: Initialize cell_list list: Put cells 0 and $n$ in cell_list
3: Initialize node_list list: Put nodes 0 and $n+1$ in node_list
4: while (cell_list not empty) do
5:    Take $c$ from cell_list list; Take $i$ from node_list list
6:    if ($c$ has no adjacent cell opposite to $i$) then
7:       Remove $c$ and $i$ from cell_list and node_list respectively
8:    else
9:       Let $c'$ be the cell adjacent to $c$ and opposite to $i$
10:      Let $k$ be the common vertex to $c$ and $c'$; Let $j$ be the other node of $c'$
11:      Compute $v_- = t_{c,i}(v_i)$ and $v_+ = t_{c',j}(v_j)$
12:      Remove $c$ and $i$ from cell_list and node_list respectively
13:      if ($|r_{c'}(v_-, v_j)| \le |r_c(v_i, v_+)|$) then
14:         if ($|r_{c'}(v_-, v_j)| = |r_c(v_i, v_+)|$) and (visited($k$) = true) then
15:            stop
16:         end if
17:         Put cell $c'$ in cell_list; put node $k$ in node_list
18:         $v_k \leftarrow v_-$; visited($k$) ← true
19:      end if
20:   end if
21: end while
```

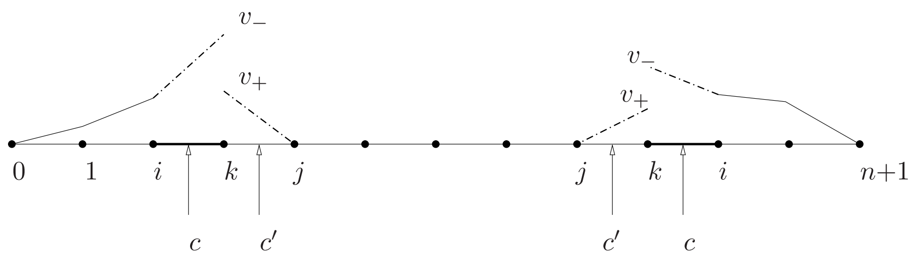


FIGURE 3.1. *Notation and schematic representation of Algorithm 1.*

Recall that the mesh is composed of cells $I_i=[x_i,x_{i+1}]$, $i\in\{0,\dots,n\}$, and the nodes are the points x_l, $l\in\{0,\dots,n+1\}$. Algorithm 1 proceeds from the left and the right boundary to the interior of the domain. The algorithm is initialized by setting the approximate solution to zero. If nonzero boundary conditions are prescribed, then one has to change v_0 and v_{n+1} to the appropriate values. The list `node_list` contains the (at most) two nodes where the approximate solution has been last updated. This list is initialized by the boundary conditions, i.e., it initially contains the indices of nodes x_0 and x_{n+1}. The list `cell_list` contains at most two cells and it initially contains the indices of cells I_0 and I_n. The purpose of this list is to update the numerical solution using local L^1-minimization. Let c be one cell from `cell_list` and i be the corresponding node from `node_list` (i could be the left of right vertex of cell c). We denote by c' the cell which is adjacent to c and opposite to i. The common node to c and c' is denoted by k; the other node of c' is denoted by j. (See Figure 3.1 for details.) The algorithm then consists of using Lemma 2.1. If $t_{c,i}(v_i)$ minimizes the local L^1-residual functional, then v_k is updated to the value $t_{c,i}(v_i)$, and cell c is replaced in the list `cell_list` by c' and node i is replaced in the list `node_list` by k. Otherwise, c and i are just removed from the lists `cell_list` and `node_list` respectively. The algorithm stops when either the list `cell_list` is empty or every node has been visited and it is not possible to reduce the local L^1-residual.

Note that owing to Lemma 2.1, the if-statement testing $|r_{i_1}(v_-,v_{k_1})|\le |r_{i_0}(v_{k_0},v_+)|$ in line 13 of Algorithm 1 is not needed if we assume (2.3)–(2.4). But this test is actually needed if we replace the restrictive set of assumptions (2.3)–(2.4) by the following:

$$\begin{cases} 0<\inf\limits_{x\in\Omega_l}\beta(x),\\ \sup\limits_{x\in\Omega_l}\beta'(x)<1,\end{cases} \qquad \text{and} \qquad \begin{cases} \sup\limits_{x\in\Omega_r}\beta(x)<0,\\ -1<\inf\limits_{x\in\Omega_r}\beta'(x),\end{cases} \tag{3.6}$$

where $\Omega_l=(0,\alpha)$, $\Omega_r=(\alpha,1)$ is a (possibly trivial) partition of Ω, with $\alpha\in[0,1]$.

PROPOSITION 3.1. *Algorithm 1 stops in $\mathcal{O}(\frac{3}{2}n)$ steps and gives the solution to* (2.10).

Proof. The most unfavorable case occurs when both cells from `cell_list` move forward until they collide ($\frac{n}{2}+\frac{n}{2}$ operations); at the collision moment the upwind cell wins owing to Lemma 2.1; thereafter the upwind cell undoes what the other one has done ($\frac{n}{2}$ operations). When the only cell left in `cell_list` reaches the downwind boundary, the algorithm stops, and, owing to Theorem 2.2, the output is the minimizer of (2.10).

The most favorable case occurs when, due to lucky initialization, the downwind cell (i.e., that moving in the upwind direction) stops at the first test; then only the upwind cell moves forward and the algorithm finishes in n steps. ∎

Remark 3.1. *Note that the algorithm does not refer to the upwind/downwind notion per se. The local L^1-minimization takes care of that naturally.*

3.3. Numerical results. To illustrate the above algorithm, we apply it to (2.1). Two situations are considered. In the first one the velocity field is continuous, and in the second one it is discontinuous. The unit interval $[0,1]$ is divided into cells of constant size, and $\mathbb{P}_1$ finite elements are used.

3.3.1. Constant advection: ill-posed problem. We set $\beta := \frac{1}{2}$ and $f := 1$ on $[0,1]$. As already mentioned above, the system (2.1) is ill-posed but nevertheless has a unique viscosity solution. As β is positive, this viscosity solution is obtained by solving the ODE $u + \frac{1}{2}u' = 1$ using $u(0) = 1$ as initial condition. The viscosity solution is $u_{\text{visc}}(x) = 1 - \mathrm{e}^{-2x}$. The least-squares solution to this problem solves the two-point boundary value problem $u_{\text{LS}} - \frac{1}{4}u''_{\text{LS}} = 1$, $u_{\text{LS}}(0) = 0$, $u_{\text{LS}}(1) = 0$. Clearly $u_{\text{LS}}(x) = ae^{2x} + be^{-2x} + 1$, where $a = -\frac{1-e^{-2}}{e^2 - e^{-2}}$ and $b = -\frac{e^2-1}{e^2-e^{-2}}$, and $u_{\text{LS}} \neq u_{\text{visc}}$.

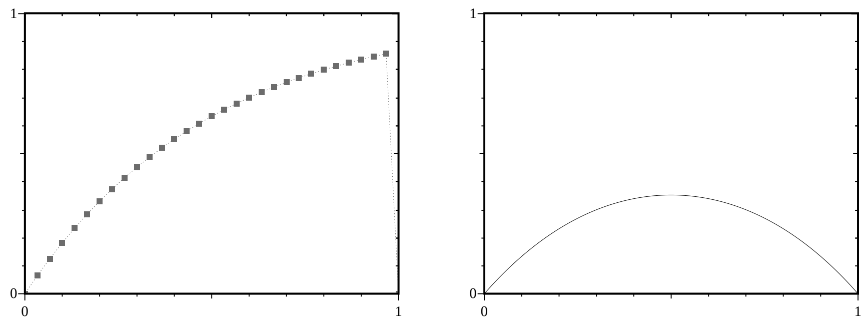


Figure 3.2. *Constant advection: L^1-solution with 30 mesh cells (Left); least-squares solution u_{LS} (Right).*

We compare in Figure 3.2 the L^1-solution and the least-squares solution. The L^1-solution is computed using Algorithm 1 on a mesh composed of 30 uniformly distributed cells. Clearly the L^1-solution approximates the viscosity solution, and in spite of the irrelevant right-hand boundary condition, it is accurate everywhere in the domain but in the rightmost cell. The L^1-minimization recognizes the left-hand boundary condition to be the correct one and propagates it continuously in the interior of the domain. The discontinuity created in the last cell does not perturb the solution anywhere else. Indeed, no oscillations or instabilities of any kind are generated. By contrast, the least-squares solution suffers from the ill-posedness of the problem and yields an erroneous solution. All these observations confirm the theoretical analysis from Guermond and Popov [11].

3.3.2. Linear discontinuous advection: shock. To mimic what happens when a shock occurs in nonlinear conservation laws, we now consider the following discontinuous advection field:

$$\beta = \begin{cases} 0.1 & \text{if} \quad 0 \leq x \leq \frac{1}{2}, \\ -1 & \text{if} \quad \frac{1}{2} < x \leq 1. \end{cases}$$

The characteristic lines associated with this flow are entering the domain at both boundaries, creating a shock at the middle of the domain, cf. Figure 3.3. As above, the L^1-algorithm propagates continuously the boundary conditions from both boundaries to the interior of the domain, and, therefore, the shock is non-oscillatory and supported in one cell. Note that this phenomenon is independent of the number of cells partitioning the domain (the result for 30 cells is shown on the left of the figure and the result for 100 cells is shown on the right). This validates our method for discontinuous velocity advection problems.

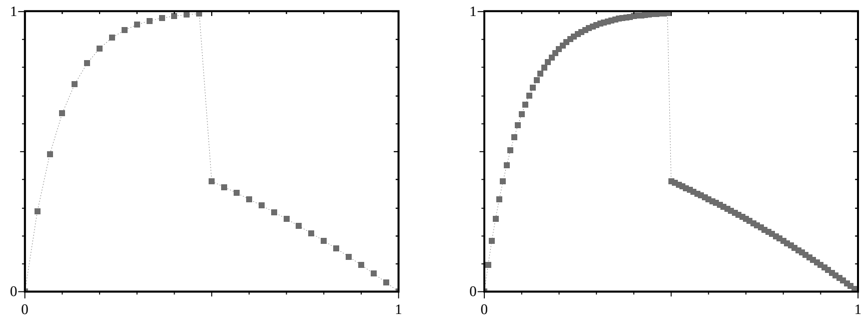


FIGURE 3.3. *L^1 solution for linear discontinuous advection:* 30 *cells (Left);* 100 *cells (Right).*

4. A fast algorithm for nonlinear problems

In this section, we extend Algorithm 1 to nonlinear problems. The main difficulty we have to deal with is non-uniqueness of solutions, which possibly lead to multiple numerical local minimizers. We propose a new algorithm to address this issue, and we illustrate its performance on stationary Hamilton-Jacobi equations.

4.1. A model problem. As in the linear case, we want to use a local minimization argument, but to avoid possible non-uniqueness issues we have to invoke an entropy selection mechanism. To clarify this point let us illustrate it on an example.

Consider the following stationary Hamilton-Jacobi equation:

$$H(x,u,u')=0, \quad \text{in } (a,b), \quad \text{with } u(a)=\alpha,\ u(b)=\beta, \tag{4.1}$$

where $[a,b]$ is a bounded interval, and assume that the Hamiltonian H satisfies the following properties:

$$|q|\le c_s\,(|H(x,v,q)|+|v|+1) \quad \forall (x,v,q)\in[a,b]\times\mathbb{R}\times\mathbb{R}, \tag{4.2}$$

$$H(x,v,q) \text{ is uniformly Lipschitz on } [a,b]\times[-R,R]\times\overline{B(0,R)} \text{ for all } R>0. \tag{4.3}$$

We assume that (4.1) has a unique viscosity solution u in $W^{1,\infty}(a,b)$ which is semi-concave. A typical example is the eikonal equation or any Hamilton-Jacobi equations derived from scalar conservation laws with convex flux, see Evans [7], Kružkov [12], or Lions and Souganidis [16].

It is shown in Guermond and Popov [10] that this problem can be solved by means of a minimization technique in L^1. More precisely, let $p>1$ be a fixed real number. Define the following functional:

$$J(v)=\int_a^b |H(x,v,v')|dx+\sum_{i=0}^{n} h_i\int_{I_i}(v''(x))_+^p dx+\sum_{i=1}^{n} h_{i+\frac{1}{2}}^{2-p}\left([\![v'(x_i)]\!]_+\right)^p, \tag{4.4}$$

where $h_{i+\frac{1}{2}} = \frac{1}{2}(h_{i-1}+h_i)$. The symbol $(z)_+$ denotes the positive part, i.e., $(z)_+ := \frac{1}{2}(|z|+z)$. The jump across cell interfaces is defined by $[[\phi(x_i)]] := \lim_{\epsilon\to 0}\phi(x_i+\epsilon) - \phi(x_i-\epsilon)$. The two extra terms in the right-hand side of (4.4) are referred to as the volume entropy and the interface entropy, respectively. It is shown in Guermond and Popov [10] that the function in X_h that minimizes J_h converges in $W^{1,1}(\Omega)$ to the unique viscosity solution of (4.1). Actually, the result proved therein holds for piecewise polynomial approximations of degree one and higher.

We now propose to simplify J and to specialize it to piecewise linear approximation by using the midpoint quadrature rule and by modifying the interface entropy. The new fully discrete functional that we henceforth consider is

$$J_h(v) := \sum_{i=0}^{n} R_i(v) + h^{2-2p}\sum_{i=1}^{n} E_i(v), \tag{4.5}$$

where we use the notation

$$R_i(v) = h_i |H(x_{i+\frac{1}{2}}, v_{i+\frac{1}{2}}, v'_{i+\frac{1}{2}})|, \tag{4.6}$$

$$E_i(v) = \omega_i(v)\left(v'_{i+\frac{1}{2}} - v'_{i-\frac{1}{2}}\right)_+^p. \tag{4.7}$$

The ω_i function is defined as follows:

$$\omega_i(v) = h_{i-1}^p S(v'_{i-\frac{1}{2}}, v'_{i+\frac{1}{2}}) + h_i^p S(v'_{i+\frac{1}{2}}, v'_{i-\frac{1}{2}}), \tag{4.8}$$

$$S(a,b) = \tfrac{1}{2}(\operatorname{sgn}(|a|-|b|)+1), \qquad \text{where sgn is the sign function.} \tag{4.9}$$

Note that $S(a,b)$ returns 1 if $|a|>|b|$, $\frac{1}{2}$ if $|a|=|b|$, and 0 otherwise. Therefore, the ω_i function returns h_{i-1}^p if the absolute value of $v'_{i-\frac{1}{2}}$ is larger than the absolute value of $v'_{i+\frac{1}{2}}$. When the mesh is uniform, i.e., $h_i = h$ for all $i \in \{0,\dots,n\}$, $\omega_i(v) = h^p$ and the entropy term in $J_h(v)$ reduces to $E_i(v) = (v_{i+1} - 2v_i + v_{i-1})_+^p$ and coincides with the entropy term of $J(v)$. If the mesh is not uniform, but quasi-uniform, the entropies from $J(v)$ and $J_h(v)$ are equivalent.

The discrete problem on which we now focus our attention is the following: Seek $u_h \in X_h$ so that

$$J_h(u_h) = \min_{v_h\in X_h} J_h(v_h). \tag{4.10}$$

4.2. Definition of the algorithm. We start by defining the local residual

$$r_i(z,s) = h_i H(x_{i+\frac{1}{2}}, \tfrac{1}{2}(z+s), h_i^{-1}(s-z)), \qquad \forall z,s\in\mathbb{R}. \tag{4.11}$$

Then we define the multi-valued nonlinear functions $t_{i,l}$, $i\in\{0,\dots,n\}$, $l\in\{i,i+1\}$ so that $r_i(z,t_{i,i}(z)) = 0$ and $r_i(t_{i,i+1}(s),s) = 0$. Note that due to the possible nonlinear character of the Hamiltonian H, $t_{i,i}(z)$ and $t_{i,i+1}(z)$ are sets and that these sets may be empty.

The algorithm that we propose to solve (4.10) is composed of two phases: (i) initialization; (ii) minimization. Due to the nonlinearity of the Hamiltonian, problem (4.10) is no longer convex. The initialization process now becomes important in order to avoid being trapped in local minimums. Initialization can be done in many ways. For instance, one could think of a hierarchical or adaptive algorithm that proceeds by successive refinements of the grid. Then the initialization could be done by projecting the coarse solution onto the new grid. Another possibility could be to roughly solve the problem (4.1) using a more standard L^2-based approximation and to use this approximate solution as initialization. As an alternative, we propose a stand-alone L^1-based technique in Algorithm 2.

(i) The initialization proceeds from the boundary to the interior and on the way selects a guess of minimal entropy. The steps describing the initialization process are detailed in Algorithm 2.

(ii) The minimization is done locally and is based on Lemma 2.1. This stage reproduces what is done in Algorithm 1. The details are reported in Algorithm 3.

Algorithm 2 L^1-initialization for (4.10).

```
1: Define u_init large enough; Set $v_0 = v_{n+1} = 0$ and define $v_{-1} = v_{n+2} = 0$
2: Initialize array updated(1:n) ← false
3: Initialize cell_list; Put cells of index 0 and $n$ in cell_list
4: Initialize node_list; Put nodes of index 0 and $n+1$ in node_list
5: while cell_list not empty do
6:    Take $c \in$ cell_list; Take $i \in$ node_list; Let $j \neq i$ be the other node of $c$
7:    Let $c' \neq c$ be s.t. $c' \cap c = \{x_j\}$; Let $k \neq j$ be the other node of $c'$
8:    Let $c'' \neq c'$ be s.t. $c'' \cap c' = \{x_k\}$; Let $l \neq k$ be the other node of $c''$
9:    if (updated(j)=true) then
10:      Store $v$ and cell index $c_{\text{break}} \leftarrow c$; Stop
11:   end if
12:   $\tilde{v} \leftarrow v$; $\tilde{v}_j \leftarrow$ u_init; $\tilde{v}_k \leftarrow$ u_init; $\tilde{v}_l \leftarrow$ u_init
13:   $J_{\text{old}} \leftarrow |R_c(\tilde{v})| + |R_{c'}(\tilde{v})| + h^{2-2p}(E_i(\tilde{v}) + E_j(\tilde{v}) + E_k(\tilde{v}))$
14:   Compute the set $t_{c,i}(v_i)$
15:   Remove cell $c$ from cell_list; Remove node $i$ from node_list
16:   if (set $t_{c,i}(v_i)$ not empty) then
17:      $\tilde{v} \leftarrow v$; $\tilde{v}_j \leftarrow t_{c,i}(v_i)$; $\tilde{v}_k \leftarrow$ u_init; $\tilde{v}_l \leftarrow$ u_init
18:      Pick $\bar{v} \in \tilde{v}$ with smallest residual $|R_{c'}(\bar{v})| + h^{2-2p}(E_i(\bar{v}) + E_j(\bar{v}) + E_k(\bar{v}))$
19:      $J_{\text{new}} \leftarrow |R_{c'}(\bar{v})| + h^{2-2p}(E_i(\bar{v}) + E_j(\bar{v}) + E_k(\bar{v}))$
20:      if ($J_{\text{new}} \leq J_{\text{old}}$) then
21:         $v_j \leftarrow \bar{v}_j$; updated($j$) ← true
22:         Put cell $c'$ in cell_list; Put node $j$ in node_list
23:      end if
24:   end if
25:   if (cell_list list empty) then
26:      Breakdown; Problem is ill-posed; Stop
27:   end if
28: end while
```

At the end of Algorithm 2, we have a field v that satisfies $r_i(v_i, v_{i+1}) = 0$ for all $i \in \{0, \ldots, n\} \setminus \{c_{\text{break}}\}$. The goal of Algorithm 3 is to move around the breakdown cell c_{break} by performing local L^1-minimization until the functional J_h cannot be further minimized. A detailed analysis of Algorithm 2 and Algorithm 3 and a convergence proof is reported in [9].

Algorithm 3 L^1-minimization for (4.10).

```
Start with initial guess from Algorithm 2: $v$ and $c_{\text{break}}$
loop
  Let $C$ be the list of the two cells adjacent to $c_{\text{break}}$
  for all ($c \in C$) do
    Let $j$ be the node of $c_{\text{break}} \cap c$
    Let $i \neq j$ the other node of $c_{\text{break}}$; Let $k \neq j$ be the other node of $c$
    $J_{\text{old}} \leftarrow |R_{c_{\text{break}}}(v)| + h^{2-2p}(E_i(v)+E_j(v)+E_k(v))$
    Compute the set $t_{c_{\text{break}},i}(v_i)$; nothing_done ← true
    if (set $t_{c_{\text{break}},i}(v_i)$ not empty) then
      $\tilde{v} \leftarrow v$; $\tilde{v}_j \leftarrow t_{c_{\text{break}},i}(v_i)$
      $J_{\text{new}} \leftarrow |R_c(\tilde{v})| + h^{2-2p}(E_i(\tilde{v})+E_j(\tilde{v})+E_k(\tilde{v}))$
      Pick $\bar{v} \in \tilde{v}$ with smallest functional (i.e., $\bar{v} = \text{argmin} J_{\text{new}}$)
      if ($J_{\text{new}}(\bar{v}) < J_{\text{old}}$) then
        $v \leftarrow \bar{v}$; $c_{\text{break}} \leftarrow c$; nothing_done ← false; Exit loop on $C$
      end if
      nothing_done ← true
    end if
  end for
  if (nothing_done=true or #loop $\geq n$) then
    Done; Stop
  end if
end loop
```

As pointed out in [10], it is not really important to compute the exact minimizer of (4.10). Actually, in the terminology of [10], computing an almost minimizer is sufficient. An almost minimizer is any sequence $(v_h)_{h>0}$ for which there exists a constant $c > 0$, uniform in h, so that $J_h(v_h) \leq ch$. Almost minimizers are known to converge to the unique viscosity solution of (4.1), see [10]. It can be proved that under reasonable assumptions on the Hamiltonian, Algorithm 2 and Algorithm 3 delivers a sequence of almost minimizers to (4.10), (see [9]).

4.3. Numerical results in the nonlinear case. In this section we give four examples to illustrate the efficiency of Algorithms 2 and 3. The first three examples are stationary Hamilton-Jacobi equations. The last example features a Burgers-like steady-state equation with a redefined entropy functional.

4.3.1. Example 1: Eikonal equation. We consider the eikonal equation in one space dimension on the interval $\Omega = (0,1)$:

$$|u'| = 1 \qquad u(0) = 0, \qquad u(1) = 0. \tag{4.12}$$

the unique viscosity solution is $u(x) = \frac{1}{2} - |x - \frac{1}{2}|$. We show in the left panel of Figure 4.1 the result obtained using Algorithm 2 and Algorithm 3 with 19 uniformly distributed cells. We see that the residual is zero in all cells but the middle one where the derivative of the solution is discontinuous. Actually, since the viscosity solution is piecewise linear, the approximation obtained with 19 uniform cells coincides with the exact solution on $[0, \frac{1}{2} - \frac{1}{38}] \cup [\frac{1}{2} + \frac{1}{38}, 1]$. Whether the mesh is uniform or not, Algorithm 2 and Algorithm 3 always gives the exact solution to (4.12) if there is a mesh interface at $\frac{1}{2}$.

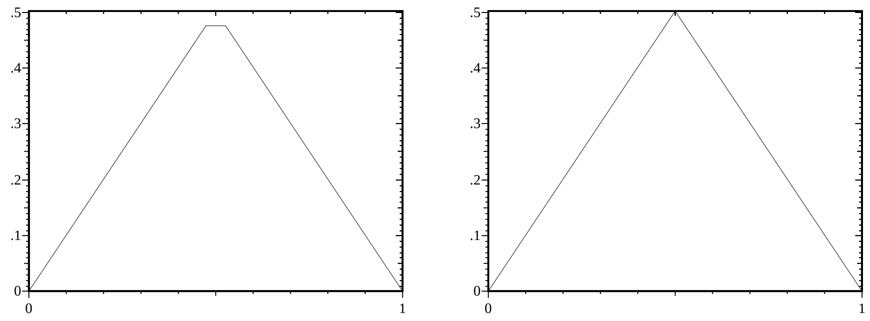


FIGURE 4.1. *Solution to* (4.12) *using different resolutions. Left:* $n+1=19$*; Right:* $n+1=20$.

4.3.2. Example 2: quadratic Hamiltonian. We now consider the following problem introduced in Cockburn and Yenikaya [4]:

$$\tfrac{1}{\pi^2}(u')^2+u+|\cos(\pi x)|-\sin(\pi x)^2=0, \quad u(0)=-1, \quad u(1)=-1, \tag{4.13}$$

whose viscosity solution is $u(x)=-|\cos(\pi x)|$. The results obtained using piecewise linear approximation on uniform grids are reported in Figure 4.2. The results for $n+1=9$, 19, and 39 cells are shown in the left panel and those obtained with $n+1=$ 10, 20, and 40 cells are shown in the right panel. In all cases the cells are uniformly distributed. The cell where the residual is not zero is clearly apparent when using an odd number of cells.

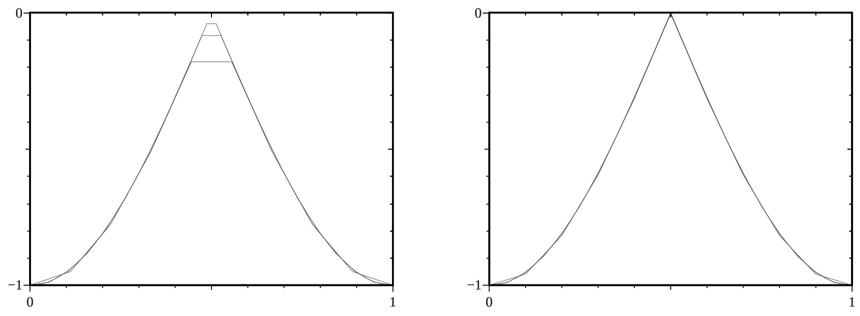
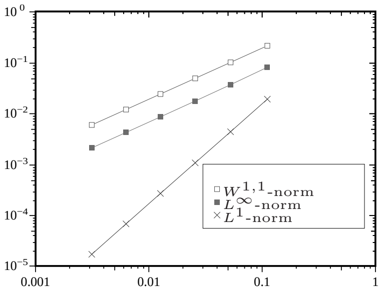


FIGURE 4.2. *Solution to* (4.13) *using different resolutions. Left:* $n+1=9$, 19, 39*; Right:* $n+1=10$, 20, 40.

Convergence tests for (4.13) are reported in Figure 4.3. The convergence order in the $W^{1,1}$-norm is 1 independently of the number of cells. The method is second-order in the L^1-norm independently of the number of cells. We observe super-convergence in the maximum-norm when the number of cells is even. This is due to the fact that the breaking point of the graph of the solution coincides with a mesh interface when the mesh is uniform and the number of cells is even.

4.3.3. Example 3: degenerate eikonal equation. In order to illustrate how the algorithm behaves with respect to the initialization process we now consider the following equation:

$$\tfrac{1}{2\pi}|u'(x)|-|\cos(2\pi x)|=0, \quad u(0)=0, \quad u(1)=0. \tag{4.14}$$



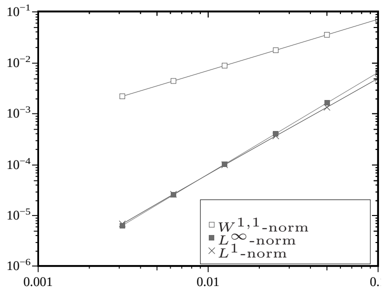


FIGURE 4.3. *Solution to* (4.13) *using different resolutions. Left:* $n+1=9$, 19, 39; *Right:* $n+1=10$, 20, 40.

This is a Hamilton-Jacobi equation with multiple semiconcave solutions. Its only positive viscosity solution is given by

$$u(x)=\begin{cases}\sin(2\pi x), & 0\le x\le \frac{1}{4},\\ 2-\sin(2\pi x), & \frac{1}{4}\le x\le \frac{1}{2},\\ 2+\sin(2\pi x), & \frac{1}{2}\le x\le \frac{3}{4},\\ \sin(2\pi x), & \frac{3}{4}\le x\le 1.\end{cases} \tag{4.15}$$

To test the robustness of the algorithm with respect to symmetry breaking, we modify Algorithm 2 so that the initialization process starts from cell 0 and moves inward until breakdown, then restarts from cell $n+1$ and moves inward until collision. Two tests are reported in Figure 4.4 using $n+1=100$ cells. In the first case we start the initialization process by setting `u_init` $=2.5$, and in the second case we start with `u_init` $=4$. The results obtained from Algorithm 2 are shown in the top left and top right panels of Figure 4.4. These two initialization fields are clearly different and are discontinuous; the discontinuity occurs in the breakdown cell. These two fields are then fed into Algorithm 3, which then produces the result shown in the bottom panel of Figure 4.4. This solution is independent of `u_init`, provided `u_init` it is large enough. More precisely, systematic tests show that if we set `u_init` $\ge 2=\max_{0\le x\le 1} u_{\text{visc}}(x)$, then Algorithm 3 always produces the viscosity solution. The observations reported in this sections have been investigated theoretically and are reported in [9].

4.3.4. Example 4: Stationary Burgers' Equation. We finish with a simple but very challenging problem. Consider the stationary inviscid Burgers equation:

$$u+\frac{d}{dx}\Big(\frac{u^2}{2}\Big)=0;\qquad u(0)=-1,\qquad u(1)=1. \tag{4.16}$$

A similar example is considered in Lavery [14] to illustrate the capability of L^1-based techniques to capture shocks. Two-dimensional versions of this problem are considered in Lavery [15] and Guermond [8]. This problem is ill-posed in many respects. To hopefully select a meaningful solution we (arbitrarily) introduce an entropy functional proportional to the total variation,

$$E_i(v)=h_{i-\frac{1}{2}}(v_i-v_{i-1})_+ + h_{i+\frac{1}{2}}(v_{i+1}-v_i)_+,\qquad i\in\{1,\dots,n\}, \tag{4.17}$$

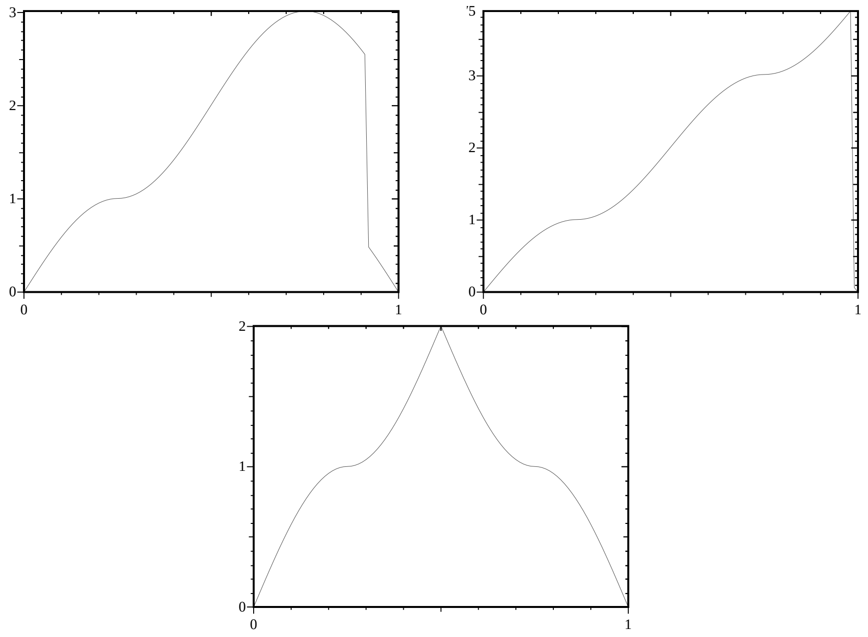


FIGURE 4.4. *Solution to* (4.14) *using* $n+1=100$. *Top left: initial guess delivered after initialization using* $u_{start}=2.5$*; Top right: initial guess delivered after initialization using* $u_{start}=4$*; Bottom: solution delivered by Algorithm 3 whatever* u_{start}*, provided* $u_{start}\geq 2$.

with obvious modifications at $i=0$ and $i=n+1$. We want to minimize the following functional:

$$J_h(v)=\sum_{i=0}^{n}h_{i+\frac{1}{2}}|\tfrac{1}{2}(v_i+v_{i+1})+h^{-1}_{i+\frac{1}{2}}(v_i-v_i)|+\sum_{i=0}^{n+1}E_i(v). \tag{4.18}$$

The approximate minimizer obtained using Algorithm 2 and Algorithm 3 with $n+1=100$ cells is shown in the left panel of Figure 4.5. The solution exhibits a shock at $x=\frac{1}{2}$. This result is consistent with those from [8, 14, 15]. It is clear that the method does not introduce any artificial viscosity since the shock is sharp and supported in one cell only.

Unfortunately, this solution is not the viscosity solution to (4.16). Actually, the algorithm performs exactly as it should, but the entropy that we have chosen (4.17) is not that which yields the viscosity solution. We have not yet figured out which entropy should be used to select the viscosity solution, but by analyzing the limit of the viscous-regularized problem, one infers that an approximation of the viscosity solution should look like what is shown in the right panel of Figure 4.5. In other words, the viscosity solution is

$$u_{\text{visc}}(x)=0 \quad \forall x\in(0,1). \tag{4.19}$$

This solution exhibits two boundary layers at both ends of the interval $[0,1]$. Although (4.16) looks hyperbolic at first sight, u_{visc} is the result of an internal equilibrium that accounts for two boundary layers. As a result, u_{visc} cannot be obtained by solving initial value problems starting from the boundary of Ω. In some sense, this problem resembles more an elliptic problem than a hyperbolic one. In other words, even with the correct entropy, we conjecture that it is impossible to minimize $J_h(v)$ in two sweeps involving only local operations. Any minimization technique for solving this problem should involve global exchange of information.

This problem opens new questions that we are currently exploring.

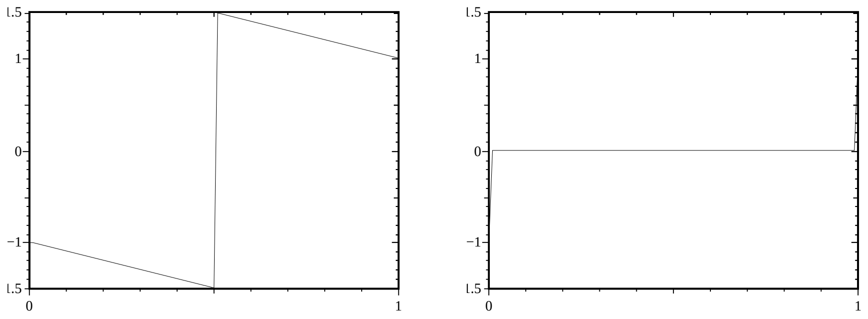


FIGURE 4.5. *Stationary Burgers' equation. (Left):* L^1*-solution with entropy* (4.17)*; (Right): Approximate viscosity solution,* 100 *cells.*

5. Conclusion

We have given a simple proof of the fact that L^1-minimization selects the viscosity solution of the linear first-order PDE (2.1) equipped with ill-posed boundary conditions. The new approach helped us to construct a $\mathcal{O}(N)$ algorithm for computing global L^1-minimizers (or almost minimizers) using sequences of local L^1-minimizations. We have extended the algorithm to stationary Hamilton-Jacobi equations. In these cases an appropriate entropy must be added to the functional to ensure convergence to the viscosity solution. The functional to be minimized is then non-smooth and non-convex. Our numerical results show that the algorithm is able to select the viscosity solution of Hamilton-Jacobi equations under appropriate assumptions on the Hamiltonian. That this is indeed the case has been proved in [9]. Extensions of the method proposed in the present paper to higher dimensions is the topic of ongoing research.

REFERENCES

[1] C. Bardos, A.Y. le Roux and J.-C. Nédélec, *First order quasilinear equations with boundary conditions*, Comm. Part. Diff. Equ., 4(9), 1017–1034, 1979.

[2] E. J. Candès, J. K. Romberg and T. Tao, *Stable signal recovery from incomplete and inaccurate measurements*, Comm. Pure Appl. Math., 59(8), 1207–1223, 2006.

[3] E. J. Candes and T. Tao, *Decoding by linear programming*, IEEE Trans. Inform. Theory, 51(12), 4203–4215, 2005.

[4] B. Cockburn and B. Yenikaya, *An adaptive method with rigorous error control for the Hamilton-Jacobi equations. Part I: The one-dimensional steady state case*, Appl. Numer. Math., 52(2-3), 175–195, 2005.

[5] D. L. Donoho, *For most large underdetermined systems of equations, the minimal* l_1*-norm near-solution approximates the sparsest near-solution*, Comm. Pure Appl. Math., 59(7), 907–934, 2006.

[6] D. L. Donoho, *For most large underdetermined systems of linear equations the minimal* l_1*-norm solution is also the sparsest solution*, Comm. Pure Appl. Math., 59(6), 797–829, 2006.

[7] L. C. Evans, *Partial Differential Equations*, Graduate Studies in Mathematics, American Mathematical Society, Providence, RI, 19, 1998.

[8] J.L. Guermond, *A finite element technique for solving first-order PDEs in L^P*, SIAM J. Numer. Anal., 42(2), 714–737, (electronic), 2004.

[9] J.L. Guermond and B. Popov, *L_1-minimization algorithms for Hamilton-Jacobi equations*, submitted, 2007.

[10] J.L. Guermond and B. Popov, *L^1-minimization methods for Hamilton-Jacobi equations: the one-dimensional case*, Numer. Math., in press, 2008.

[11] J.L. Guermond and B. Popov, *Linear advection with ill-posed boundary conditions via L^1 minimization*, Int. J. Numer. Anal. Model., 4(1), 39–47, 2007.

[12] S.N. Kružkov, *Generalized solutions of Hamilton-Jacobi equations of Eikonal type. I. Statement of the problems; existence, uniqueness and stability theorems; certain properties of the solutions*, Mat. Sb. (N.S.), 98(140)(3(11)), 450–493, 1975.

[13] J.E. Lavery, *Nonoscillatory solution of the steady-state inviscid Burgers' equation by mathematical programming*, J. Comput. Phys., 79(2), 436–448, 1988.

[14] J.E. Lavery, *Solution of steady-state one-dimensional conservation laws by mathematical programming*, SIAM J. Numer. Anal., 26(5), 1081–1089, 1989.

[15] J.E. Lavery, *Solution of steady-state, two-dimensional conservation laws by mathematical programming*, SIAM J. Numer. Anal., 28(1), 141–155, 1991.

[16] 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), 441–470, 1995.

[17] J. Nocedal and S. J. Wright, *Numerical Optimization*, Springer Series in Operations Research and Financial Engineering, Springer, New York, second edition, 2006.

[18] Y. Wang, S. Fang and J. E. Lavery, *A compressed primal-dual method for generating bivariate cubic L_1 splines*, J. Comput. Appl. Math., 201(1), 69–87, 2007.