MATHEMATICS OF COMPUTATION
VOLUME 52, NUMBER 186
APRIL 1989, PAGES 509–537

Nonlinear Filters for Efficient Shock Computation

By Björn Engquist,* Per Lötstedt,** and Björn Sjögreen***

Dedicated to Professor Eugene Isaacson on the occasion of his 70th birthday

Abstract. A new type of methods for the numerical approximation of hyperbolic conservation laws with discontinuous solution is introduced. The methods are based on standard finite difference schemes. The difference solution is processed with a nonlinear conservation form filter at every time level to eliminate spurious oscillations near shocks. It is proved that the filter can control the total variation of the solution and also produce sharp discrete shocks. The method is simpler and faster than many other high resolution schemes for shock calculations. Numerical examples in one and two space dimensions are presented.

1. Introduction. A major difficulty in the numerical approximation of nonlinear hyperbolic conservation laws is the presence of discontinuities in the solution. Traditional schemes generate spurious oscillations in the numerical solution near these discontinuities. Standard methods based on centered differencing together with artificial viscosity have often during the last few years been replaced by the so-called high-resolution schemes.

The high-resolution schemes are based on concepts like upwinding, local Riemann solvers, field by field decomposition and flux limiting, see, e.g., [1], [3], [12], [13]. These schemes are designed for shock capturing and produce sharp numerical discontinuities without oscillations. The algorithms are however not so computationally efficient. For higher-order numerical accuracy they are quite complicated.

It is our purpose to present a class of methods which retain most of the positive features of traditional schemes, but at the same time treat shocks and contacts similarly to the modern high-resolution schemes. The methods are based on standard schemes, the solutions of which are then processed with a nonlinear conservation law filter at every time step. The filter contains field by field decomposition and limiters in order to have good shock capturing properties. It is only activated at a few grid points, and therefore the overall algorithm has an order of accuracy and computational efficiency which are close to the traditional methods.

The filter step is essentially independent from the step with the basic difference scheme. It is therefore easy to implement in existing codes. See the numerical examples in Section 6.

Received January 18, 1988; revised June 20, 1988.

1980 *Mathematics Subject Classification* (1985 *Revision*). Primary 65M05, 65M10.

*Supported by NSF Grant No. DMS85-03294, NASA Consortium Agreement No. NCA2-IR360-403.

**Also at Aircraft Division, SAAB-SCANIA, S-581 88 Linköping, Sweden.

***Supported by a grant from the Thanks to Scandinavia Foundation and by UCLA.

Our problem is a system of nonlinear hyperbolic conservation laws in, e.g., two space dimensions

$$\begin{aligned} &\mathbf{u}_t + \mathbf{f}(\mathbf{u})_x + \mathbf{g}(\mathbf{u})_y = \mathbf{0}, \\ &\mathbf{u}(x, y, 0) = \mathbf{u}_0(x, y), \end{aligned} \tag{1.1}$$

with appropriate boundary conditions. The numerical Examples 3 and 4 in Section 6 are of this form.

The method will first be described for the simpler one-dimensional case

$$\begin{aligned} &\mathbf{u}_t + \mathbf{f}(\mathbf{u})_x = \mathbf{0}, \\ &\mathbf{u}(x, 0) = \mathbf{u}_0(x), \end{aligned} \tag{1.2}$$

when $\mathbf{u}$ and $\mathbf{f}$ are either scalars or vectors.

We are interested in numerical approximations of weak solutions to (1.2) and assume that a consistent basic difference scheme of conservation form is already given,

$$\begin{aligned} u_j^{n+1} &= G(u_{j-r}^n, u_{j-r+1}^n, \ldots, u_{j+r}^n), \\ u_j^0 &= u_0(x), \\ &\qquad n = 0, 1, 2, \ldots,\ j = \ldots, -1, 0, 1, \ldots, \\ &\qquad x_j = j\Delta x,\ t_n = n\Delta t,\ \Delta t = \lambda \Delta x. \end{aligned} \tag{1.3}$$

We want to couple this scheme (in short form, $u^{n+1} = G(u^n)$) with a filter or projection in the following way:

$$\begin{aligned} v^{n+1} &= G(u^n), \\ u^{n+1} &= P(v^{n+1}, u^n). \end{aligned} \tag{1.4}$$

Let us here discuss some natural conditions for P.

(1) *Consistency.* If the solution is smooth, the filter should not change the approximation v^{n+1} too much. In particular, we want the total method (1.4) to be consistent,

$$\|u^{n+1} - v^{n+1}\| = O(\Delta t^2) \quad \text{for smooth } v^{n+1}.$$

The norm can, e.g., be the l_1-norm, and smoothness may mean bounded higher divided differences. It is here possible to require that a higher order of accuracy of the basic scheme is not changed by the filter, see Subsection 2.2.

(2) *Conservation form.* For convergence to the correct weak solution it is necessary for the total scheme (1.4) to be of conservation form, which means

$$\sum_j \varphi(x_j)(u_j^{n+1} - v_j^{n+1}) \leq C \tag{1.5}$$

for φ in a suitable class of test functions.

(3) *TVD.* The filter should enforce some criterion that guarantees that there are no spurious oscillations near discontinuities. The usual criterion for a scalar conservation law is the following (TVD) inequality:

$$\begin{aligned} &\mathrm{TV}(u^{n+1}) \leq \mathrm{TV}(u^n), \\ &\mathrm{TV}(u^n) = \sum_j |u_{j+1}^n - u_j^n|. \end{aligned} \tag{1.6}$$

For systems of equations, (1.6) is applied in the characteristic fields. See Section 2 for details and a discussion of other criteria.

(4) *Minimal change.* The filter step should enforce (3), without violating (1) or (2), with few numerical operations. This means that

$$|u_j^{n+1} - v_j^{n+1}| \neq 0$$

for as few j-values as possible and that the filter should be neutral if v^{n+1} already satisfies the criterion (3). The filter step should thus be a projection

$$P(P(v^{n+1}, u^n), u^n) = P(v^{n+1}, u^n).$$

The methods we will present will not all satisfy (3) or (4) fully because we are also interested in simplicity of the algorithms. It may be possible to derive even simpler filters for particular difference schemes. We have here concentrated on the general case where the filter is not based on special properties of the difference algorithm.

The fundamental steps in the filter are first to find the extrema of v^{n+1}. If these extrema increase the variation of v^{n+1} over that of u^n, then v^{n+1} is corrected field by field in a conservative way such that (1.6) is satisfied. This means that the filter is only activated at very few mesh points and all corrections are local. For details see Sections 2 and 4.

The standard artificial viscosity method can be regarded as a filter for the elimination of high frequencies. It is quite different from the methods which are described here. All mesh values are affected and it is not based on field by field decomposition or limiters. In [4], a filter with a switch is introduced in order to reduce numerical oscillations at shocks. This filter affects all mesh values, but the main changes are close to steep gradients. Filters are also introduced to postprocess results from spectral method approximation of shock problems [2].

The filter algorithms are presented in Section 2. In Section 3 we prove some properties of the algorithms in Section 2. Section 4 deals with the field by field decomposition for solving systems of equations and in Section 5 we give a result on the existence of sharp shock profiles. The numerical results in Section 6 contain, among other things, the computed solution of the Euler equations for flow in a channel towards a forward facing step, and the steady flow around an airfoil.

2. Filter Algorithms for Scalar Equations. In the first part of this section filter algorithms of increasing capability and complexity are presented and discussed. In the second part, suggestions are given for generalizations of these types of methods. It is outside the scope of this paper to further investigate these suggestions.

2.1. *Description of Some Filters.* Let us assume that we are given $v^{n+1} = (v_1^{n+1} v_2^{n+1} \cdots v_N^{n+1})^T$, the result after taking one step with the difference scheme (1.3). The simplest filter, which is not TVD but still of practical use, works according to the following principles:

(1) Scan through the function values v_j^{n+1}, $j = 1, \dots, N$, and correct the v_j^{n+1} values that are local maxima or local minima.
(2) Correction is made by decreasing maxima and by increasing minima.

(3) When a correction is added to a point v_j^{n+1}, the same correction must be subtracted from a neighboring point, otherwise conservation is lost and one may obtain the wrong shock speed. The corrected neighbor is chosen as the one with the greatest distance to v_j^{n+1}.

(4) No value may be corrected so that it passes its neighbors. This means that we do not want to overcompensate so that new extrema are created.

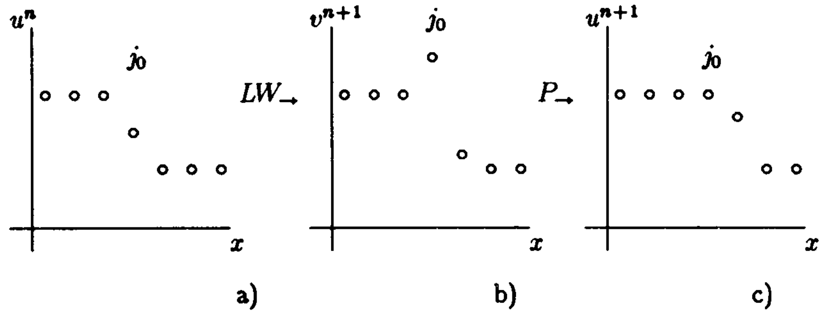


FIGURE 2.1
The Lax-Wendroff scheme is applied to a step function, and the filter corrects the error made.

The principles are illustrated by an example which exhibits the typical behavior of the Lax-Wendroff scheme when applied to a moving shock solution of the inviscid Burgers equation. In Figure 2.1, u^n is this solution at time level n, given as a step function (Figure 2.1a). One time step with the Lax-Wendroff difference scheme is taken, giving v^{n+1} in Figure 2.1b. An overshoot is introduced. The filter produces the result u^{n+1} in Figure 2.1c. It follows the four principles above:

(1) $v_{j_0}^{n+1}$ is discovered as a maximum $((\Delta_+ v_{j_0}^{n+1})(\Delta_- v_{j_0}^{n+1}) < 0)$.

(2) $v_{j_0}^{n+1}$ will be decreased.

(3) $v_{j_0-1}^{n+1}$ or $v_{j_0+1}^{n+1}$ must be increased by the same amount as $v_{j_0}^{n+1}$ is decreased. The filter chooses $v_{j_0+1}^{n+1}$ because the distance to the j_0+1 neighbor is larger than the distance to the j_0-1 neighbor $(|\Delta_+ v_{j_0}^{n+1}| > |\Delta_- v_{j_0}^{n+1}|)$.

(4) $v_{j_0}^{n+1}$ must not be decreased further than to the level of $v_{j_0-1}^{n+1}$, otherwise $v_{j_0}^{n+1}$ will pass the neighbor $v_{j_0-1}^{n+1}$, a case which we do not allow.

The symbols Δ_+ and Δ_- denote the forward and backward difference, respectively, $\Delta_\pm u_j = \pm(u_{j\pm 1} - u_j)$. These principles can be formulated in Algorithm 2.1 below. The vector u is to be understood as an array in a computer program, that initially contains the function to be filtered v^{n+1} and after completion of the algorithm holds the filtered solution u^{n+1}.

ALGORITHM 2.1
```
for $j := 2$ to $N - 1$ do
    if $(\Delta_+ u_j)(\Delta_- u_j) < 0$ then
        if $|\Delta_+ u_j| > |\Delta_- u_j|$ then
            $\delta_+ := |\Delta_+ u_j|$
            $\delta_- := |\Delta_- u_j|$
            $jcorr := j + 1$
        else
            $\delta_+ := |\Delta_- u_j|$
            $\delta_- := |\Delta_+ u_j|$
            $jcorr := j - 1$
        endif
        $\delta := \min(\delta_-, \delta_+/2)$
        $s := \mathrm{sgn}(\Delta_+ u_j)$
        $u_j := u_j + s\delta$
        $u_{jcorr} := u_{jcorr} - s\delta$
    endif
endfor
```

This filter has proved to work very well in the computations we have made. Numerical results will be presented in Section 6, and in Section 4 the algorithm will be generalized to systems of equations.

There are, however, some properties of Algorithm 2.1 that are not entirely satisfactory:

(1) It cannot discover extrema consisting of more than one point (Figure 2.2).

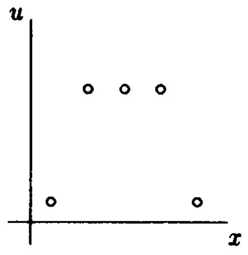


FIGURE 2.2
Plateau maximum.

(2) It does not give a method which is TVD. The total variation may increase, $\mathrm{TV}(u^{n+1}) > \mathrm{TV}(u^n)$ (Figure 2.3). This means that the method allows oscillations around the shocks, and indeed in numerical experiments there are oscillations present in the solution. These, however, have in most cases been observed to be of very small amplitude, cf. Table 6.1.

(3) It flattens extrema that are not the result of overshooting and thus gives a low order of accuracy locally around smooth extrema, a property shared with all TVD-schemes [6].

Algorithm 2.1 can be modified such that these difficulties are overcome.

We continue by modifying Algorithm 2.1 to obtain a TVD-enforcing filter. As can be seen in Figure 2.3, $v_{j_0}^{n+1}$ should sometimes decrease further. The question is how much. A stop condition is required. We have chosen to accept an extremum at j_0 if the condition

$$\text{(C1)} \qquad \begin{array}{ll} \min(u_{j_0-1}^n, u_{j_0}^n, u_{j_0+1}^n) \le u_{j_0}^{n+1} & \text{if } u_{j_0}^{n+1} \text{ is a minimum,} \\ u_{j_0}^{n+1} \le \max(u_{j_0-1}^n, u_{j_0}^n, u_{j_0+1}^n) & \text{if } u_{j_0}^{n+1} \text{ is a maximum,} \end{array}$$

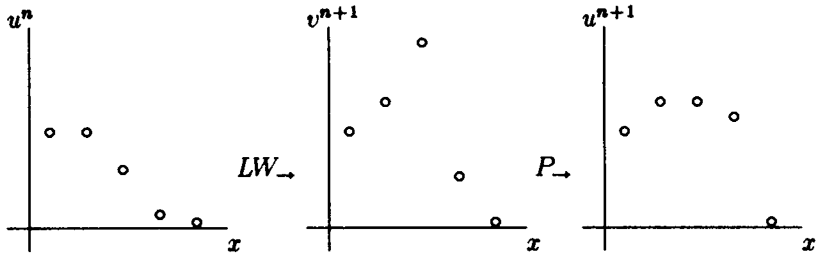


FIGURE 2.3
Algorithm 2.1 *fails to be TVD.*

holds for all extrema $u_{j_0}^{n+1}$ of u^{n+1}. This condition is not sufficient to guarantee TVD. An additional constraint excluding the possibility that an extremum at x_j is followed by one at x_{j+1} is necessary in the following theorem.

THEOREM 2.1. *If* u^{n+1} *have no consecutive extrema, and if* (C1) *holds at all extrema of* u^{n+1}, *then*

$$\mathrm{TV}(u^{n+1}) \le \mathrm{TV}(u^n).$$

Proof. We have $\mathrm{TV}(u^{n+1}) = \sum_j |u_{j+1}^{n+1} - u_j^{n+1}| = \sum_\nu |u_{j_{\nu+1}}^{n+1} - u_{j_\nu}^{n+1}|$, where $\{u_{j_\nu}^{n+1}\}_\nu$ is an enumeration of the extrema in u^{n+1}. By (C1) there holds

$$\sum_\nu |u_{j_{\nu+1}}^{n+1} - u_{j_\nu}^{n+1}| \le \sum_\nu |u_{j'_{\nu+1}}^n - u_{j'_\nu}^n|, \qquad j'_\nu \in \{j_\nu - 1, j_\nu, j_\nu + 1\}.$$

If the distance between two extrema contains at least one point, it follows that $j'_{\nu+1} \ge j_{\nu+1} - 1 \ge j_\nu + 1 \ge j'_\nu$. Thus the sequence $\{j'_\nu\}$ is nondecreasing and

$$\begin{aligned}\sum_\nu |u_{j'_{\nu+1}}^n - u_{j'_\nu}^n| &= \sum_\nu |u_{j'_{\nu+1}}^n - u_{j'_{\nu+1}}^n + u_{j'_{\nu+1}-1}^n - u_{j'_{\nu+1}-2}^n \\ &\qquad + u_{j'_{\nu+1}-2}^n - \cdots - u_{j'_\nu+1}^n + u_{j'_\nu+1}^n - u_{j'_\nu}^n| \\ &\le \sum_{l=j'_0}^{j'_1-1} |u_{l+1}^n - u_l^n| + \sum_{l=j'_1}^{j'_2-1} |u_{l+1}^n - u_l^n| + \dots \\ &= \sum_l |u_{l+1}^n - u_l^n| = \mathrm{TV}(u^n). \quad \square\end{aligned}$$

Remark 2.1. The condition of no consecutive extrema in the theorem is necessary. Consider the mesh function

$$\begin{aligned} u_j^n &= 0, \qquad j \ne 0,1, \\ u_0^n &= -1, \qquad u_1^n = 1, \\ u_j^{n+1} &= 0, \qquad j \ne 0,1,2, \\ u_0^{n+1} &= u_2^{n+1} = 1, \qquad u_1^{n+1} = -1. \end{aligned}$$

The condition (C1) is valid but $\mathrm{TV}(u^n) = 4$, $\mathrm{TV}(u^{n+1}) = 6$.

We can modify Algorithm 2.1 as follows:

(a) Continue the decrease of maxima and increase of minima until (C1) holds.
(b) If an extremum is accepted at j_0, check whether there is one at $j_0 - 1$. If that is the case, take a special step to remove this, by Theorem 2.1, forbidden situation.

We arrive at Algorithm 2.2. The output of this algorithm will satisfy the conditions needed in Theorem 2.1, provided there are no plateaus as extremas. The notation is the same as in Algorithm 2.1. The vector u is an array which initially contains the unfiltered function v^{n+1} and returns the filtered function u^{n+1} as output. Notice that this algorithm requires that we have saved the solution u^n from the previous time level.

ALGORITHM 2.2

```
$j := 2$
while $j < N$ do
    if $(\Delta_+ u_j)(\Delta_- u_j) < 0$ and not admissible $(j, u, u^n)$ then
        correct $u_j$ in the same way as in Algorithm 2.1
    elseif $(\Delta_+ u_j)\ (\Delta_- u_j) < 0$ and $(\Delta_+ u_{j-1})(\Delta_- u_{j-1}) < 0$ then
        comment: this removes a zig-zag situation
        $\delta := \min(|\Delta_- u_{j-1}|, |\Delta_+ u_{j-1}|/2, |\Delta_+ u_j|)$
        $s := \operatorname{sgn}(\Delta_+ u_j)$
        $u_{j-1} := u_{j-1} - s\delta$
        $u_j := u_j + s\delta$
        $j := j - 1$
    else
        comment: if $u_j$ does not need any correction go on to $j + 1$
        $j := j + 1$
    endif
endwhile
```

The algorithm uses a function *admisssible* which checks the condition (C1). *admissible*(j, u^{n+1}, u^n) returns *true* if

$$\min(u_{j-1}^n, u_j^n, u_{j+1}^n) < u_j^{n+1} < \max(u_{j-1}^n, u_j^n, u_{j+1}^n),$$

otherwise the value *false* is returned.

In the filter used for the numerical experiments in Section 6 we introduced one further improvement in this algorithm. From the proof of Theorem 2.1, we infer that the case when there is a minimum at $j-1$ and a maximum at j can be allowed if the following hold:

$$\text{(C2)}\qquad \begin{aligned} &\min(u_j^n, u_{j+1}^n) < u_j^{n+1} < \max(u_j^n, u_{j+1}^n) \text{ and}\\ &\min(u_{j-2}^n, u_{j-1}^n, u_j^n) < u_{j-1}^{n+1} < \max(u_{j-2}^n, u_{j-1}^n, u_j^n)\\ &\text{or}\\ &\min(u_{j-1}^n, u_j^n, u_{j+1}^n) < u_j^{n+1} < \max(u_{j-1}^n, u_j^n, u_{j+1}^n) \text{ and}\\ &\min(u_{j-2}^n, u_{j-1}^n) < u_{j-1}^{n+1} < \max(u_{j-2}^n, u_{j-1}^n). \end{aligned}$$

We can add a function *admissible2*$(j, j-1, u^n, u^{n+1})$ checking this condition. *admissible2* returns *true* if the condition (C2) holds. Otherwise the result is *false*. Introduce this function together with the test $(\Delta_+ u_j)(\Delta_- u_j) < 0$ and $(\Delta_+ u_{j-1})(\Delta_- u_{j-1}) < 0$ into the **elseif** text in Algorithm 2.2. The new condition becomes

$$(\Delta_+ u_j)(\Delta_- u_j) < 0 \textit{ and } (\Delta_+ u_{j-1})(\Delta_- u_{j-1}) < 0 \textit{ and } \textit{not admissible2}(j, j-1, u^n, u).$$

This modification improves the quality of the filtered function in the sense that the accuracy is improved. We conclude this from numerical experiments.

We now modify Algorithm 2.2 to take into account the fact that some maxima or minima can contain more than one value. The modifications needed consist entirely of bookkeeping to keep track of various plateaus with pointers in the computer program, and introduce no new ideas. We define a plateau of length r to be a set of indices, $\{i, i+1, \dots, i+r\}$ such that

$$u_{i-1} \neq u_i = u_{i+1} = \cdots = u_{i+r} \neq u_{i+r+1},$$

see Figure 2.4. A plateau is treated as a unit by the filter. When searching for extrema, the condition

$$(\Delta_+ u_j)(\Delta_- u_j) < 0$$

is replaced by

$$(\Delta_+ u_j)(\Delta_- u_l) < 0,$$

where j and l satisfy

$$u_{l-1} \neq u_l = u_{l-1} = \cdots = u_k = \cdots = u_j \neq u_{j+1}$$

in Algorithm 2.2. The condition (C1) is replaced by

$$\min(u_{l-1}^n, u_l^n, \dots, u_{j+1}^n) < u_j^{n+1} < \max(u_{l-1}^n, \dots, u_{j+1}^n).$$

Finally, all the points in the extremum, $u_l, \dots, u_j$, are decreased by the same amount. An example is given in Figure 2.4.

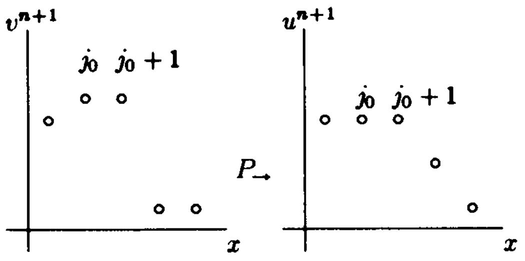


FIGURE 2.4
An inadmissible maximum is treated as a unit.

With these modifications the correction part in Algorithms 2.1 and 2.2 becomes:

ALGORITHM 2.3 (first part)

```
    newind(u, 2, l, j)
    while $j < N$ do
        if $(\Delta_+ u_j)(\Delta_- u_l) < 0$ and not admissible(l, j, u, u^n) then
            if $|\Delta_+ u_j| > |\Delta_- u_l|$ then
                $\delta_+ := |\Delta_+ u_j|$
                $\delta_- := |\Delta_- u_l|$
                $jcorr := j + 1$
                $lcorr := j + 1$
            else
                $\delta_+ := |\Delta_- u_l|$
                $\delta_- := |\Delta_+ u_j|$
            **comment:** a backward correction must not destroy
            the corrected solution to the left of $l$, therefore we
            make the backward step in the sense of plateaus.
            $newind(u, l-1, lcorr, jcorr)$
        **endif**
        $\omega_1 := j - l + 1$
        $\omega_2 := jcorr - lcorr + 1$
        $\delta := \min(\delta_-, \omega_2\delta_+/(\omega_1 + \omega_2))$
        $s := \mathrm{sgn}(\Delta_+ u_j)$
        **for** $i := l$ **to** $j$ **do**
            $u_i := u_i + s\delta$
        **endfor**
        **for** $i := lcorr$ **to** $jcorr$ **do**
            $u_i := u_i - s\delta\omega_1/\omega_2$
        **endfor**
        $newind(u, j, l, j)$
    **elseif** $(\Delta_+ u_j)(\Delta_- u_j) < 0$ *and* $(\Delta_+ u_{j-1})(\Delta_- u_{j-1}) < 0$ **then**
...
```

The rest of Algorithm 2.2 can be rewritten to handle plateaus in the same manner as indicated above to obtain the first part of Algorithm 2.3. Note that we need to weigh the corrections depending on the number of points in the plateaus corrected. Here the weights ω_1 and ω_2 are chosen such that the sum $\sum_{j=1}^{N} u$ is not changed, i.e., u is conserved, and the plateaus are not passing each other. The algorithm uses some subroutines, described below.

newind(u, ind, l, j)—The vector u and the index *ind* are given as input. The indices l, j such that

$$u_{l-1} \neq u_l = u_{l+1} = \cdots = u_{ind} = \cdots = u_j \neq u_{j+1}$$

are returned as output.

admissible(l, j, u, u^n)—checks whether

$$\min(u_{l-1}^n, u_l^n, \ldots, u_{j+1}^n) < u_j < \max(u_{l-1}^n, \ldots, u_{j+1}^n).$$

All plateaus are kept track of by using l as the leftmost point in the plateau and j as the rightmost.

Thus, a plateau is treated as one point with the weight adjusted for conservation. It is sometimes necessary, however, to release a point and regard it as separate from the neighboring plateau. Filtering of piecewise constants would otherwise not be local, and the local conservation property (1.5) would be violated. Since another point is added to another plateau in the same filter step, the total number of points belonging to plateaus is constant.

We now have a filter that forces TVD upon the solution u regardless of the difference scheme. We now discuss some possibilities to increase the accuracy at smooth extrema. Instead of decreasing $u_{j_0}^{n+1}$ to the same level as a neighboring point, the value $u_{j_0}^{n+1}$ is lowered only so that it fulfills (C1). This means a replacement of the statement

$$\delta := \min(\delta_+/2, \delta_-)$$

in Algorithms 2.1–2.3 by

$$\delta := \min(\delta_+/2, \delta_-, \max(d_1, d_2)),$$

where $d_1 = u_{j_0} - \max(u^n_{l-1}, \dots, u^n_{j+1})$ and $d_2 = \min(u^n_{l-1}, \dots, u^n_{j+1}) - u_{j_0}$, if j_0 belongs to a plateau between u^n_l and u^n_j. If there is a one-point extremum, then $l = j = j_0$. The result is another filter algorithm. We do not give a detailed description, since most of it is similar to the previous ones.

ALGORITHM 2.4

```
$j := 2$
**while** $j < N$ **do**
    $d_1 := u_j - \max(u^n_{l-1}, \dots, u^n_{j+1})$
    $d_2 := \min(u^n_{l-1}, \dots, u^n_{j+1}) - u_j$
    **if** $(\Delta_+ u_j)(\Delta_- u_l) < 0$ *and* $(d_1 > 0$ *or* $d_2 > 0)$ **then**
        correct $u_j$ in the same way as in Algorithm 2.1, taking
        into account plateaus as described above, but with
        $\delta := \min(\delta_+/2, \delta_-, \max(d_1, d_2))$
    **elseif** $(\Delta_+ u_j)(\Delta_- u_l) < 0$ *and* $(\Delta_+ u_{l-1})(\Delta_- u_p) < 0$ **then**
        remove the zig-zagging as in Algorithm 2.2 with plateaus taken
        into account ($p$ is the left index of the plateau that has
        $l - 1$ as right index)
    **else**
        **comment:** if $u_j$ does not need any correction go on to $j + 1$
        *newind*$(u, j + 1, l, j)$
    **endif**
**endwhile**
```

The step that handles a consecutive maximum-minimum is not changed. This algorithm is optimal in the sense that it makes the smallest possible correction, still being TVD.

2.2. *Generalizations.* The filters that were presented in Subsection 2.1 were either designed to produce TVD-solutions or they were simplifications of the TVD-filters.

TVD-methods have a significant drawback. Enforcing a strict total variation bound

$$\sum_j |u^{n+1}_{j+1} - u^{n+1}_j| \leq \sum_j |u^n_{j+1} - u^n_j|$$

makes it impossible for the method to be of higher-order accuracy at smooth extrema [6].

We will suggest three possible ways of avoiding this deficiency for filter type methods:

(1) It is possible not to trigger the filter at smooth extrema. The condition "if $(\Delta_+ u_j)(\Delta_- u_j) < 0$" in the filter algorithm can, e.g., be augmented by testing if there is an inflection point. A natural test is to see if $\Delta_+\Delta_- u_j$ changes sign close to the extremum. The second difference does not change sign near a typical smooth extremum. The generic spurious oscillation however contains inflection points.

(2) Another possibility is to correct, e.g., a new maximum only if it is higher than the extrapolated solution from the left and the right. Furthermore, the correction should not modify the solution to a value lower than the extrapolated one. Algorithm 2.1 is in fact such a method based on constant extrapolation. Higher-order standard one-sided or ENO extrapolations [7] are possible.

(3) We can use other criteria than TVD for the filter design. The method can be constructed to produce nonoscillatory solutions, cf. the ENO scheme [7]. The search for new extrema will still be the first step in such an algorithm. The field by field decomposition will also be the same. The goal of the correction will then be to ensure that no new extrema occur.

The suggestions above are all for less restrictive filters or similarly for projections onto larger classes of solutions. In one respect it would be desirable with a more strict filter.

There is no guarantee that the filtered solution satisfies an entropy condition. As it is with the filters in Subsection 2.1, the entropy has to be taken care of by the basic difference scheme. It would be possible to check for an entropy inequality in the same way as we now check for extrema. If the entropy inequality is not satisfied, the solution can be modified until the inequality is valid.

Since the solution is divided into eigenvectors at extrema, it is possible to include artificial compression in the filter for the field containing contacts [3].

Finally, if the filter will be used inside an implicit algorithm, the correction should depend continuously on the previous step.

3. Analysis of Properties.

The first question to ask is whether we can keep on changing the function like we do in the filters, without affecting the convergence to the right solution. One result that indicates how much we are allowed to change v^{n+1} is the following theorem.

THEOREM 3.1. *Assume that*

$$v_j^{n+1} = u_j^n - \lambda(F(u_{j+k+1}^n, u_{j+k}^n, \dots, u_{j-m}^n) - F(u_{j+k}^n, u_{j+k-1}^n, \dots, u_{j-m-1}^n))$$

is a finite difference scheme consistent with (1.2), *that F is a Lipschitz continuous function of its arguments and that the correction c_j^n added by a filter is of the form*

$$c_j^n = \sum_{k=-s}^{r} a_{jk}^n \Delta_+ v_{j+k}^{n+1}, \qquad \sum_{k=-s}^{r} a_{j-k,k}^n = 0,$$
$$u_j^{n+1} = v_j^{n+1} + c_j^n,$$

with r, s bounded independently of Δx. If $u^n \to u$ in L^1_{loc} with u in $L^1 \times L^1$ as $\Delta x \to 0$, $\Delta t = \lambda \Delta x$, then u is a weak solution to (1.2).

Proof. The proof is an extension of the convergence proof in [5] to include filter corrections. Let $\varphi(x,t)$ be a test function in $C_0^\infty(\mathbf{R} \times \mathbf{R}_+)$. Multiply the relation $u_j^{n+1} = u_j^n - \lambda(\Delta_+ F_{j-1/2}^n) + c_j^n$ by $\varphi_j^n = \varphi(x_j, t_n)$ and sum over n and j,

$$\sum_{n=0}^{N} \sum_{j=-J}^{J} (\varphi_j^n (u_j^{n+1} - u_j^n) + \varphi_j^n \lambda (\Delta_+ F_{j-1/2}^n)) = \sum_{n=0}^{N} \sum_{j=-J}^{J} \varphi_j^n c_j^n.$$

Here, $N\Delta t = \max\{t\colon \varphi(x,t) \neq 0\}$ and $J\Delta x = \max\{|x|\colon \varphi(x,t) \neq 0\}$. These numbers are finite, since φ has compact support, but N and $J \to \infty$ as $\Delta x, \Delta t \to 0$. Multiply by Δx and sum by parts. The left-hand side becomes a Riemann sum which converges to the integral

$$\iint_{\mathbf{R}\times\mathbf{R}_+} \varphi_t u + \varphi_x f(u)\, dx\, dt, \tag{3.1}$$

and the right-hand side becomes

$$\Delta x \sum_{n=0}^{N} \sum_{j=-J}^{J} \varphi_j^n c_j^n. \tag{3.2}$$

It remains to prove that this quantity goes to zero as $\Delta x, \Delta t \to 0$. Let positive constants be denoted by C in the sequel. We can prove this in the following way:

$$\begin{aligned}
\left|\Delta x \sum_{n=0}^{N} \sum_{j=-J}^{J} \varphi_j^n c_j^n\right| &= \left|\Delta x \sum_{n=0}^{N} \sum_{j=-J}^{J} \sum_{k=-s}^{r} a_{jk}^n \varphi_j^n \Delta_+ v_{j+k}^{n+1}\right| \\
&= \Delta x \left|\sum_{n=0}^{N} \sum_{l=-L}^{L} \sum_{k=-s}^{r} a_{l-k,k}^n \varphi_{l-k}^n \Delta_+ v_l^{n+1}\right| \\
&= \Delta x^2 \left|\sum_{n=0}^{N} \sum_{l=-L}^{L} \Delta_+ v_l^{n+1} \sum_{k=-s}^{r} k a_{l-k,k}^n \frac{\partial\varphi(x_l+\xi_k, t_n)}{\partial x}\right| \\
&\le \Delta x^2 C \sum_{n=0}^{N} \sum_{l=-L}^{L} |\Delta_+ v_l^{n+1}| \\
&\le \Delta x^2 C \sum_{n=0}^{N} \sum_{l=-L}^{L} |v_{l+1}^{n+1} - u_{l+1}^n| + |u_{l+1}^n - u_l^n| + |u_l^n - v_l^{n+1}|.
\end{aligned}$$

Since F is Lipschitz continuous, we have

$$\begin{aligned}
\sum_{l=-L}^{L} |v_l^{n+1} - u_l^n| &= \sum_{l=-L}^{L} \lambda |F(u_{l+k+1}^n, u_{l+k}^n, \dots, u_{l-m}^n) \\
&\qquad - F(u_{l+k}^n, u_{l+k-1}^n, \dots, u_{l-m-1}^n)| \\
&\le \lambda C_1 \sum_{l=-L}^{L} \sum_{i=-m}^{k+1} |u_{l+i}^n - u_{l+i-1}^n| \\
&\le \lambda C_1 (k+m+2) \sum_{l=-L-m-1}^{L+k} |u_{l+1}^n - u_l^n|.
\end{aligned}$$

Thus,

$$\begin{aligned}
\left|\Delta x \sum_{n=0}^{N} \sum_{j=-J}^{J} \varphi_j^n c_j^n\right| &\le \Delta x^2 C_2 \sum_{n=0}^{N} \sum_{l=-L-m-1}^{L+k} |u_{l+1}^n - u_l^n| \\
&\le C_2 \|u(x+\Delta x, t) - u(x,t)\|_{L^1\times L^1}.
\end{aligned}$$

By a well-known theorem for u in $L^1 \times L^1$ the upper bound vanishes as $\Delta x \to 0$. □

This theorem states that if we do not allow any correction to spread out over an increasing number of mesh points as $\Delta x \to 0$, then we still have convergence to the correct weak solution, if we have convergence. It is evident that the filter described in Algorithm 2.1 fulfills the conditions in Theorem 3.1.

Another question which arises is the following: Is the TVD-filter a finite algorithm? Will it terminate in a finite number of steps on each time level? To answer this question, one can argue like in the next theorem.

Let us first introduce some notation. If v is a function on a mesh for a one-dimensional problem, we let M be the number of values in v if we count occurrences of more than one consecutive equal value as one. For the function $\{v_j\} = \{1.1\ 2.3\ 3.4\ 3.4\ 3.4\ 2.5\ 1.1\}, M$ is 5. We let the pair (j, M) represent the state of the filter algorithm at a given moment. The index j is the location where the filter is working and M is as defined above. The operators T_p, $p = 1, 2, 3, 4$, describe the action of the filter on v when one step of a filter algorithm is executed.

THEOREM 3.2. *Assume that a filter is applied to a point j and that one of the following modifications of v is made*:

1. *the solution at two neighboring points is given equal value and $M := M - 1$,*

$$T_1(j, M) = (j, M - 1),$$

or

2. *the end point of one plateau is released and a new point is included in another plateau,*

$$T_2(j, M) = (j, M),$$

or

3. *one step backward is taken, $j := j - 1$, and a new point is included in a plateau, $M := M - 1$,*

$$T_3(j, M) = (j - 1, M - 1),$$

or

4. *no solution values are changed and a step forward is taken, $j := j + 1$,*

$$T_4(j, M) = (j + 1, M).$$

Then the total number of filter corrections in the filter algorithm is bounded from above.

Proof. We shall prove that if

$$T = \prod_{i=1}^{L} T_{k_i}(1, N) = (a, b) \quad \text{with } a \le N,\ b \ge 1,\ k_i = 1, 2, 3 \text{ or } 4,$$

then L must be a finite number. This means that the filter cannot take an infinite number of steps, since the state where $M = 1$ or $j = N$ will be reached in a finite number of steps.

Let n_1, n_2, n_3 and n_4 be the number of times T_1, T_2, T_3 and T_4 occurs in T, respectively. It follows from the assumption that

$$a = n_4 - n_3, \qquad b \le N - n_3.$$

Since $a \le N, b \ge 1$, we have

$$n_3 \le N - 1, \qquad n_4 \le 2N - 1.$$

We need upper bounds on n_1' and n_2', the number of operators T_1 or T_2 in T between two operators T_3 or T_4. It follows from the definition of T_1 that

$$n_1' \leq M - 1 \leq N - 1.$$

The number of points in a plateau from which T_2 releases points gives the upper bound

$$n_2' \leq N - 1.$$

Therefore,

$$n_1 \leq n_1'(n_3 + n_4) \leq (N-1)(3N-2),$$
$$n_2 \leq n_2'(n_3 + n_4) \leq (N-1)(3N-2),$$

and

$$L = n_1 + n_2 + n_3 + n_4 < 6N^2. \quad \square$$

Remark 1. In practice, the number of filter iterations is always much smaller than the upper bound derived in the proof.

Remark 2. It follows trivially from the description of Algorithm 2.1 that it is finite, but the final result may not be a TVD solution.

The conditions in the theorem are satisfied by the filter Algorithm 2.3. The statement inside the **while** loop in Algorithm 2.3 consists of three different branches corresponding to the three different alternatives in the **if**-statement. In the first branch the filter operator is T_1, in the second branch T_3 and in the third branch T_4. If a modification of the values at the edges of a plateau is included as indicated after the description of the algorithm, then this operation is represented by T_2. Observe that all four alternatives need not be incorporated in a filter in order for the theorem to be applicable.

During the practical implementation of Algorithm 2.3 it happened to us several times that programming errors caused the algorithm to become infinite. The question of finiteness is thus extremely important, since very small changes in the algorithm can make it infinite.

Algorithm 2.1 *on Conservation Form.* The simple filter Algorithm 2.1 can be written as a correction to the numerical flux of the underlying difference scheme. Let $v^{(k)}$ be the resulting solution vector, when $k-2$ loops in the for-loop of Algorithm 2.1 have been performed. With notations as in (1.4), we have

$$v^{(2)} = v^{n+1}, \quad v^{(N)} = u^{n+1}, \quad v_j^{(k)} = v_j^{n+1}, \qquad j > k. \tag{3.3}$$

Define

$$r_j = \frac{\Delta_+ v_j^{(j)}}{\Delta_- v_j^{(j)}}$$

and

$$\varsigma(r) = \begin{cases} 1, & r \leq -2, \\ -r/2, & -2 < r < -1, \\ 1/4, & r = -1, \\ 0, & r > -1. \end{cases}$$

Then one step with Algorithm 2.1 can be written:

$$
\begin{aligned}
v_j^{(j+1)} &= v_j^{(j)} - \varsigma(r_j)\Delta_- v_j^{(j)} + \varsigma(1/r_j)\Delta_+ v_j^{(j)},\\
v_{j-1}^{(j+1)} &= v_{j-1}^{(j)} - \varsigma(1/r_j)\Delta_+ v_j^{(j)},\\
v_{j+1}^{(j+1)} &= v_{j+1}^{(j)} + \varsigma(r_j)\Delta_- v_j^{(j)}.
\end{aligned}
\tag{3.4}
$$

Here we have treated the case $\Delta_- v_j^{(j)} = -\Delta_+ v_j^{(j)}$ differently from how it is done in Algorithm 2.1. With the original algorithm it would not have been possible to make ς a single-valued function. After completion of the filter iterations we get

$$
\begin{aligned}
u_j^{n+1} &= v_j^{n+1} - \varsigma(r_j)\Delta_- v_j^{(j)} + \varsigma(1/r_j)\Delta_+ v_j^{(j)}\\
&\quad - \varsigma(1/r_{j+1})\Delta_+ v_{j+1}^{(j+1)} + \varsigma(r_{j-1})\Delta_- v_{j-1}^{(j-1)}\\
&= v_j^{n+1} - \Delta_+(\varsigma(r_{j-1})\Delta_- v_{j-1}^{(j-1)} + \varsigma(1/r_j)\Delta_+ v_j^{(j)})\\
&= u_j^n - \lambda\Delta_+(F(u_{j+q}^n, \dots, u_{j-p}^n) + \tfrac{1}{\lambda}\varsigma(r_{j-1})\Delta_- v_{j-1}^{(j-1)} + \tfrac{1}{\lambda}\varsigma(1/r_j)\Delta_+ v_j^{(j)}),
\end{aligned}
$$

where F is the numerical flux function of the basic difference scheme (1.3). The numerical flux, F_f, of the filter scheme becomes

$$F_f = F + \tfrac{1}{\lambda}\varsigma(r_{j-1})\Delta_- v_{j-1}^{(j-1)} + \tfrac{1}{\lambda}\varsigma(1/r_j)\Delta_+ v_j^{(j)},$$

where $r_{j-1}, v_{j-1}^{(j-1)}$, and $v_j^{(j)}$ are well-defined functions of u^n. The flux simplifies to

$$F_f = F + \tfrac{1}{\lambda}\varsigma(r_{j-1})\Delta_- v_{j-1}^{n+1} + \tfrac{1}{\lambda}\varsigma(1/r_j)\Delta_+ v_j^{n+1}$$

if the filter is not triggered in the neighborhood of j, i.e., $\Delta_+ u_i \Delta_- u_i \geq 0$, for $i = j-2, j-1, j+1$.

4. Systems of Equations.

Let us consider the system

$$\mathbf{u}_t + \mathbf{f}(\mathbf{u})_x = \mathbf{0}, \quad \mathbf{u}(x,t) \in \mathbf{R}^m, \quad \mathbf{f}\colon \mathbf{R}^m \to \mathbf{R}^m. \tag{4.1}$$

The generalization to systems is done by a field by field decomposition in the same manner as in [3]. Let m be the number of equations. Expand $\Delta_+\mathbf{u}_j$ in a basis of vectors $\mathbf{e}_{j+1/2}^k$,

$$\Delta_+\mathbf{u}_j = \sum_{k=1}^m \alpha_{j+1/2}^k \mathbf{e}_{j+1/2}^k. \tag{4.2}$$

The vectors $\mathbf{e}_{j+1/2}^k$ are eigenvectors of a matrix $A(\mathbf{u}_j, \mathbf{u}_{j+1})$ representing some kind of average between the matrices $A(\mathbf{u}_j)$ and $A(\mathbf{u}_{j+1})$, where $A(\mathbf{u}) = \mathbf{f}_\mathbf{u}$. $A(\mathbf{u}_j, \mathbf{u}_{j+1})$ is here taken to be the Roe matrix [10]. We then apply the filter, componentwise, to the coefficients $\alpha_{j+1/2}^k$. The condition for extrema $(\Delta_+ u_j)(\Delta_- u_j) < 0$ in the scalar filter algorithms is replaced by

$$(\Delta_+(u_k)_j)(\Delta_-(u_k)_j) < 0 \quad \text{for any } k = 1, \dots, m.$$

It is important to observe that we only have to compute the eigenvectors when a correction is needed. This is not the case for upwind schemes, where the expensive computation of eigenvectors is required at every mesh point.

The algorithms in Section 2 are generalized to systems by replacing every occurrence of $\Delta_\pm u_j$ by $\alpha^k_{j\pm 1/2}$. The statement $u_j := u_j + s\delta$ has to be reformulated as

$$\Delta_+ u_j := \Delta_+ u_j - s\delta, \qquad \Delta_- u_j := \Delta_- u_j + s\delta,$$

in the scalar algorithms. It is then easy to see how to do the generalization to systems.

When the new values of $\Delta_+ \mathbf{u}_j$ have been computed, we update the original solution. Here we want to stress one important point. For conservation it is necessary to add the same correction to $\Delta_+ \mathbf{u}_j$ as is subtracted from $\Delta_- \mathbf{u}_j$. We must realize that a correction in a characteristic field is a vector, for example if the first field is corrected by the amount δ, then

$$\begin{aligned}\Delta_+ \mathbf{u}_j &= (\alpha^1_{j+1/2} + \delta)\mathbf{e}^1_{j+1/2} + \sum_{k=2}^{m} \alpha^k_{j+1/2}\mathbf{e}^k_{j+1/2} \\ &= \sum_{k=1}^{m} \alpha^k_{j+1/2}\mathbf{e}^k_{j+1/2} + \delta\mathbf{e}^1_{j+1/2},\end{aligned}$$

and the correction is in reality $\delta\mathbf{e}^1_{j+1/2}$. If we try to subtract δ from the first characteristic field in $\Delta_- \mathbf{u}_j$, the correction will be $-\delta\mathbf{e}^1_{j-1/2}$ and conservation will be violated. Each correction must thus be added and subtracted in the same coordinate system for all values involved in this correction.

Algorithm 2.1 generalized to systems in the way described above becomes

ALGORITHM 4.1

```
for $j := 2$ to $N-1$ do
    if $extremum(j,u)$ then
        determine the eigenvectors of the Roe matrix and the decomposition (4.2)
        for $k := 1$ to $m$ do
            if $\alpha^k_{j-1/2}\alpha^k_{j+1/2} < 0$ then
                if $|\alpha^k_{j+1/2}| < |\alpha^k_{j-1/2}|$ then
                        $\delta_+ := |\alpha^k_{j-1/2}|$
                        $\delta_- := |\alpha^k_{j+1/2}|$
                        $jcorr := j - 1$
                else
                        $\delta_+ := |\alpha^k_{j+1/2}|$
                        $\delta_- := |\alpha^k_{j-1/2}|$
                        $jcorr := j + 1$
                endif
                $\delta := \min(\delta_-, \delta_+/2)$
                $s := \mathrm{sgn}(\alpha^k_{j-1/2})$
                $\Delta_+ u_j := \Delta_+ u_j + s\delta e^k_{j+1/2}$
                $\Delta_- u_j := \Delta_- u_j - s\delta e^k_{j+1/2}$
                $\Delta_+ u_{jcorr} := \Delta_+ u_{jcorr} - s\delta e^k_{j+1/2}$
                $\Delta_- u_{jcorr} := \Delta_- u_{jcorr} + s\delta e^k_{j+1/2}$
            endif
            endfor
            for $i := \min(j, jcorr)$ to $\max(j, jcorr) + 1$ do
                $u_i := u_{i-1} + \Delta_+ u_{i-1}$
            endfor
        endif
    endfor
```

In the algorithm, u is an array with $m \times N$ entries and the assignment statement (e.g., $u_i := u_{i-1} + \Delta_+ u_{i-1}$) means component by component assignment of the value of the right-hand side to the variable on the left-hand side. The algorithm uses the function *extremum*(j, u) which returns the value *true* if any component of u has an extremum at j and *false* otherwise. This check is made in the physical variables. The numerical results with this filter are reported in Section 6.

The TVD property is not true for the original variables in the case of systems. The limiting in Algorithm 4.1 is therefore done in the locally characteristic variables. The advantage of the strict TVD algorithm over the simpler Algorithm 2.1 is not at all clear when applied to systems. The great advantage of using a filter is the comparatively low cost and the simplicity of the implementation. Most of these properties are lost in the generalization of the TVD filter to systems.

For equations in more than one space dimension the filter is applied to each dimension separately. The basic difference step is decoupled from the filter step and therefore the difference method does not need to be based on dimensional splitting. See Section 6 for numerical examples of two-dimensional problems.

5. Sharp Shock Profiles. In this section we will investigate the behavior of the filter at an isolated shock when a scalar conservation law is approximated by a basic 3-point difference scheme. We will show that the method generates very sharp discrete shocks without oscillations. As an example, the condition on the difference algorithm is interpreted for a Lax-Wendroff scheme.

The basic 3-point scheme is assumed to be consistent and of conservation form,

$$\begin{gathered} v_j^{n+1} = u_j^n - \lambda(F(u_{j+1}^n, u_j^n) - F(u_j^n, u_{j-1}^n)), \\ F(u,u) = f(u). \end{gathered} \tag{5.1}$$

We will consider an approximation of a piecewise constant scalar shock solution to (1.2):

$$\begin{gathered} u = \begin{cases} u_L, & x \le st + \bar{x}, \\ u_R, & x > st + \bar{x}, \end{cases} \\ f(u_L) \ge f(u_R), \qquad u_L > u_R, \\ s = (f(u_L) - f(u_R))/(u_L - u_R). \end{gathered} \tag{5.2}$$

The case $f(u_L) < f(u_R)$ is equivalent.

Definition. A difference approximation to the above problem has a p-point discrete shock solution if there exists a solution of the form

$$\begin{aligned} &u_j^n = u_L, && j < j_n, \\ &u_R < u_{j_n+r}^n < u_L, && r = 0, 1, \ldots, p-1, \\ &u_j^n = u_R, && j \ge j_n + p, \end{aligned}$$

for all n. Let us introduce the notation

$$\begin{aligned} u_j &= u_j^n, \qquad v_j = v_j^{n+1}, \\ F_{ij} &= F(u_i, u_j), \\ F_i &= F(u_i, u_i) = f(u_i), \end{aligned}$$

where i and j may include L, R.

THEOREM 5.1. *The difference scheme* (5.1) *with the TVD-filter* (*Algorithm* 2.3) *has a solution with at most one point in the shock if*

$$\text{(5.3)} \qquad \begin{aligned} F_L &> F(u, u_L), \qquad u \in (u_R, u_L), \\ F_R &> F(u_R, u), \qquad u \in (u_R, u_L), \end{aligned}$$

and if the stepsizes satisfy a CFL-condition

$$\text{(5.4)} \qquad \lambda \max_{u_R \le u \le u_L} |f'(u)| \le 1.$$

Proof. Assume that at the time level n the solution satisfies

$$\begin{aligned} u_j &= u_L, \qquad j < 0, \\ u_R \le u_0 &\le u_L, \\ u_j &= u_R, \qquad j > 0. \end{aligned}$$

Consider first the case $u_0 - u_L$. The effect of (5.1) and (5.3) is then

$$\begin{aligned} v_j &= u_L, \qquad j < 0, \\ v_0 &= u_L - \lambda(F_{RL} - F_L) \ge u_L, \\ v_1 &= u_R - \lambda(F_R - F_{RL}) \le u_R, \\ v_j &= u_R, \qquad j > 1. \end{aligned}$$

Correcting the local maximum at v_0 implies

$$\begin{aligned} \tilde{v}_0 &= u_L, \\ \tilde{v}_1 &= v_1 - \lambda(F_{RL} - F_L). \end{aligned}$$

The notation $\tilde{v}_j$ is used for intermediate steps in the filter algorithm. Substituting for v_1 and using (5.2) gives

$$\tilde{v}_1 = u_R - \lambda(F_R - F_L) \ge u_R.$$

Since λ satisfies the CFL-condition (5.4), we have

$$u_R \le \tilde{v}_1 \le u_L.$$

The filter step is executed and $u_0^{n+1} = \tilde{v}_0$, $u_1^{n+1} = \tilde{v}_1$ satisfy the original assumption.

The case $u_0 = u_R$ is identical after an index shift. We thus need to analyze the final situation

$$u_R < u_0 < u_L.$$

From (5.1) and (5.3) we have

$$\begin{aligned} v_j &= u_L, \qquad j < -1, \\ v_{-1} &> u_L, \\ v_1 &< u_R, \\ v_j &= u_R, \qquad j > 1. \end{aligned}$$

Consider first $v_0 \geq u_L - (v_{-1} - u_L)$. The value at $j = -1$ will then be corrected as a plateau with v_0 until

$$\tilde{v}_{-1} = \tilde{v}_0 = u_L$$

if after conservative correction $\tilde{v}_1 \leq u_L$. The latter is true from total conservation and (5.4),

$$\tilde{v}_{-1} + \tilde{v}_0 + \tilde{v}_1 = v_{-1} + v_0 + v_1 = u_L + u_0 + u_R - \lambda(F_R - F_L) \leq 3u_L.$$

For $v_0 < u_L - (v_{-1} - u_L), v_{-1}$ will be corrected directly, resulting in

$$\begin{aligned} \tilde{v}_{-1} &= u_L, \\ \tilde{v}_0 &= v_0 + (v_{-1} - u_L) < u_L, \\ \tilde{v}_0 &= u_0 - \lambda(F_{R0} - F_{0L}) - \lambda(F_{0L} - F_L) = u_0 - \lambda(F_{R0} - F_L). \end{aligned}$$

From (5.2) and (5.3) we get $\tilde{v}_0 \geq u_0 > u_R$.

The final extremum is v_1, which will be corrected to u_R, changing $\tilde{v}_0$ to

$$u_0^{n+1} = u_0 - \lambda(F_{R0} - F_L) - \lambda(F_R - F_{R0}) = u_0 - \lambda(F_R - F_L).$$

Thus,

$$u_0^{n+1} \geq u_0 > u_R. \quad \square$$

The condition (5.3) guarantees an oscillatory behavior at the left and right edges of the shock and a smoothing of the solution by the filter. In order to illustrate the result of this section, we choose the same formulation of the Lax-Wendroff scheme as in [3] and $f_i = f(u_i)$. Then the numerical flux function is

$$\begin{aligned} F_{i+1,i} &= \tfrac{1}{2}[f_i + f_{i+1} - \tfrac{1}{\lambda}\overline{\nu}_{i+1/2}^2(u_{i+1} - u_i)], \\ \overline{\nu}_{i+1/2} &= \lambda\overline{a}_{i+1/2} = \lambda\overline{a}_{i,i+1} = \lambda(f_{i+1} - f_i)/(u_{i+1} - u_i). \end{aligned} \tag{5.5}$$

The sufficient condition in Theorem 5.1 at the left edge for the scheme defined by (5.5) is

$$\begin{aligned} F_L - F(u_i, u_L) &= \tfrac{1}{2}[2f_L - f_L - f_i + \lambda(f_i - f_L)^2/(u_i - u_L)] \\ &= \tfrac{1}{2}(f_L - f_i)[1 - \lambda(f_i - f_L)/(u_i - u_L)] > 0. \end{aligned} \tag{5.6}$$

It follows from (5.6) that if $f(u_L) > f(u_i)$ and (5.4) are satisfied for $u_i \in (u_R, u_L)$, then the conditions for the state to the left of the shock is valid. The condition at the right edge for the scheme (5.5) is derived analogously.

The difference corresponding to (5.3) for Harten's TVD scheme [3] at a discrete shock is

$$\begin{aligned} F_L - F(u_j, u_L) &= \tfrac{1}{2}[2f_L - f_L - f_j + \tfrac{1}{\lambda}Q_{L_j}(u_j - u_L)] \\ &= \tfrac{1}{2\lambda}(u_L - u_j)(\lambda\overline{a}_{L_j} - Q_{L_j}), \end{aligned} \tag{5.7}$$

where Q_{L_j} is "the coefficient of numerical viscosity" [3], and j is the point in the shock. In [3], Q_{L_j} is chosen such that $Q_{L_j} \geq |\lambda\overline{a}_{L_j}|$ when $|\lambda\overline{a}_{L_j}| \leq 1$. By (5.7) and the fact that $u_L \geq u_j$ we have that $F_L \leq F_{jL}$. There is no oscillation at the left edge, which is one purpose of the construction in [3], and no filtering is necessary.

6. Numerical Results.

The results of the numerical experiments with the filter and various difference schemes are presented. The conservation laws in our examples are the scalar inviscid Burgers' equation, the one-dimensional Euler equations in a shock tube problem, the two-dimensional Euler equations in a forward facing step problem and the two-dimensional steady Euler equations in the computation of flow around an airfoil.

The first test problem is the inviscid Burgers' equation

$$u_t + (u^2/2)_x = 0,$$
$$u(x,0) = \begin{cases} 1, & x < 0, \\ 0, & x > 0. \end{cases}$$

TABLE 6.1

A propagating shock at $t = 1.5$; Burgers' equation.

	Lax-Wendroff+filter 2.1	2nd-order TVD scheme [3]	Lax-Wendroff+filter 2.4
j	u_j	u_j	u_j
81	1.000000004420621	0.9999999999999990	1.000000000000000
82	1.000000004420621	0.9999999999998279	1.000000000000000
83	0.9999993061366822	0.9999999999679732	1.000000000000000
84	0.9999993061366822	0.9999999942705017	1.000000000000000
85	1.000037711135951	0.9999989269090302	1.000000000000000
86	1.015281066912056	0.9998095242194506	1.000000000000000
87	1.015281066912056	0.9670489441867344	1.000000000000000
88	0.2576304741863363	0.3183558845540151	0.2882263480002554
89	$1.210594951885533 0E - 04$	$3.126192681904862 9E - 03$	$1.236519997446279 5E - 04$
90	$2.713149710446193 2E - 16$	$1.050168769411626 6E - 05$	$4.013220627906287 8E - 16$
91	$0.000000000000000 0E + 00$	$3.143267041915546 2E - 08$	$0.000000000000000 0E + 00$

This problem has been solved up to time = 1.5 with a CFL-number = 0.8. In the results in Table 6.1 the computed numbers are presented, because the difference between the results from different methods can be seen only in the second and higher decimal places.

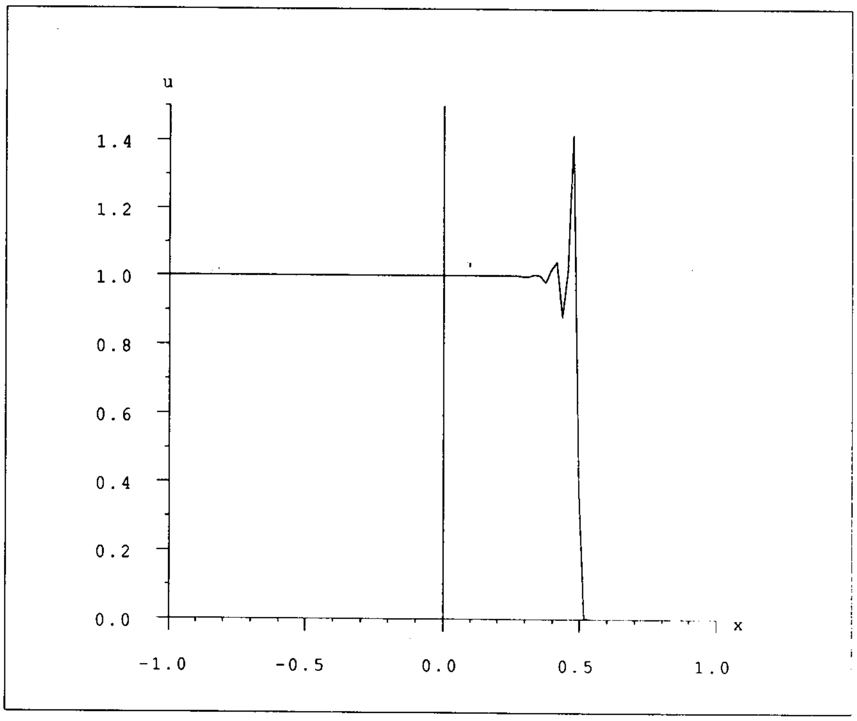
For comparison, computations made with a 2nd-order TVD scheme (Harten [3]) are displayed. The filters 2.1 and 2.4 have been used together with the Lax-Wendroff scheme in [8]. This form of the Lax-Wendroff scheme is not the same as (5.5) at the end of Section 5, and (5.3) is not satisfied, but the schock is still fairly sharp. The results with and without filtering are illustrated in Figure 6.1.

The solution to Burgers' equation with a sine wave as initial condition,

$$u(x,0) = 0.25 + 0.5 \sin \pi x,$$

was computed with the Lax-Wendroff scheme and the filter. At $t = 0.75$ a shock has developed, see Figure 6.2. In Table 6.2 the computed results in the neighborhood of the shock are displayed. The number of points in the shock is one also here. Theorem 5.1 seems to be valid even if the solution is not piecewise constant.

(a)



(b)

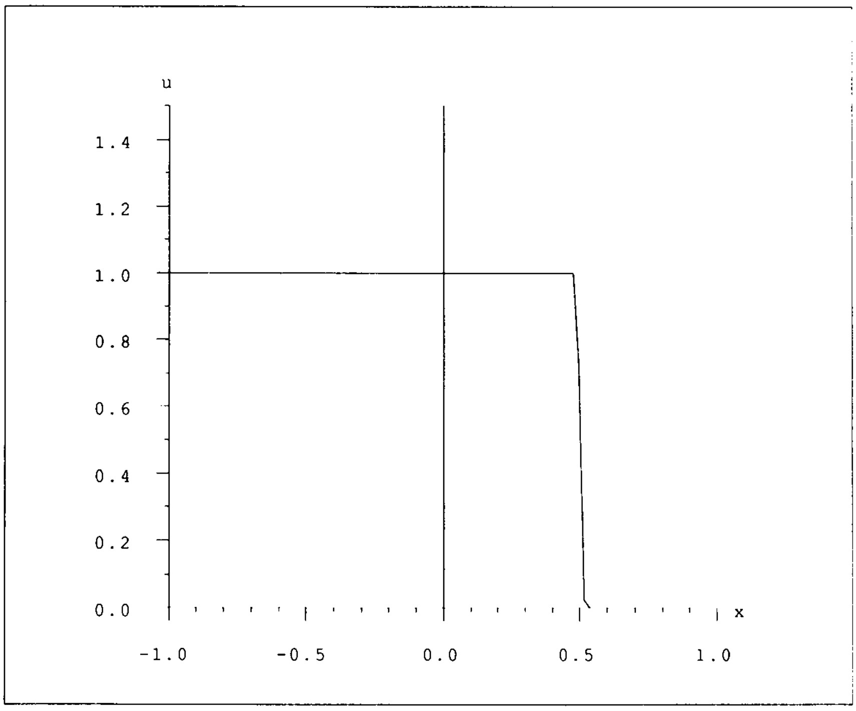


FIGURE 6.1

Shock solution for Burgers' equation with the Lax-Wendroff scheme.
(a) *without filter.*
(b) *with the filter in Algorithm* 2.4.

The numerical solution is compared with the exact solution at $t = 0.75$. The numerical l_1 errors in the smooth part of the solution, i.e., at least a distance 0.1 away from the shock, are presented for three different spatial discretizations in Table 6.3. The results indicate that the smooth solution is second-order accurate.

TABLE 6.2

An initial sine wave has developed into a shock.

	Lax-Wendroff+filter 2.1
j	u_j
5	0.7510193076844476
6	0.7473223222626426
7	0.7333636414152179
8	0.7333636414152179
9	0.7333636414152179
10	0.4015613729134132
11	− 0.2195990067488021
12	− 0.2195990067488021
13	− 0.2354176977104853
14	− 0.2438072343976059

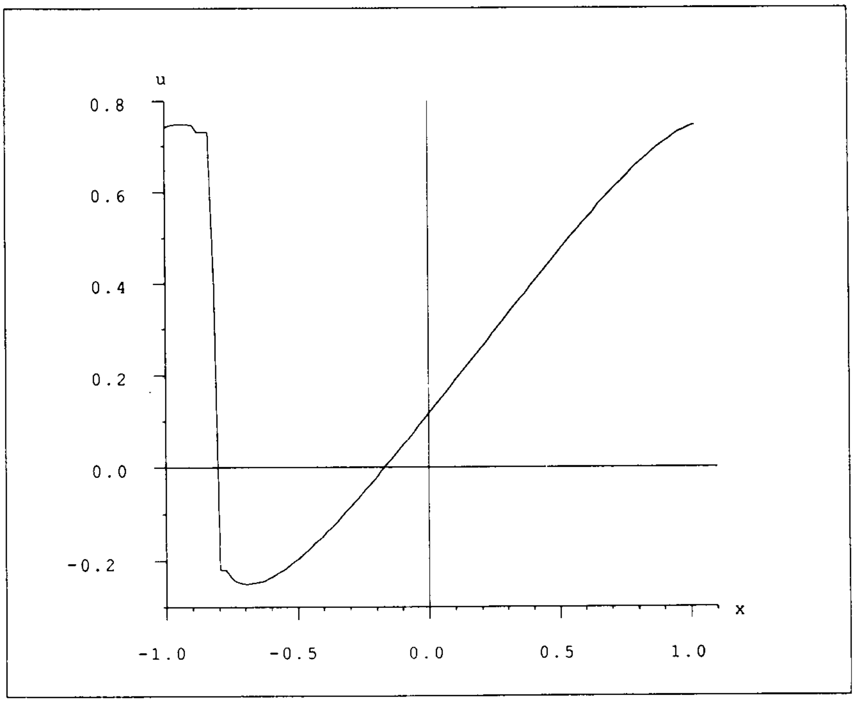


FIGURE 6.2

Shock solution at $t = 0.75$ for Burgers' equation with the Lax-Wendroff scheme and filter and a sine wave as initial condition.

TABLE 6.3

The solutions to the sine wave problem have been processed with filter 2.1 *at each time level.*

Δx	Scheme	l_1 error	Numerical order
0.005	LW	1.2×10^{-5}	1.9
0.01	LW	4.4×10^{-5}	2.0
0.02	LW	1.8×10^{-4}	2.4
0.04	LW	9.8×10^{-4}	
0.005	D_0	4.1×10^{-4}	1.0
0.01	D_0	8.3×10^{-4}	1.1
0.02	D_0	1.8×10^{-3}	1.3
0.04	D_0	4.5×10^{-3}	

The TVD-filter enforces TVD independently of difference scheme and CFL-number. It is possible to use large CFL-numbers and unstable schemes like $u_j^{n+1} = u_j^n - \Delta t D_0 f(u_j^n)$ together with this filter, but the quality of the solution is of course affected negatively by large CFL-numbers. As an example, we have included results from using the filter together with the pure centered scheme above in Table 6.3. The obtained order of accuracy agrees with the expected one.

In order to test the ability of Algorithm 4.1 to remove oscillations for systems of equations, we use the Euler equations and as initial data a function which consists of two constant states. This is a shock tube problem that has been used by many others as a test problem (e.g., [3], [11]). The equations are

$$\begin{pmatrix} \rho \\ \rho u \\ e \end{pmatrix}_t + \begin{pmatrix} \rho u \\ p + \rho u^2 \\ u(e+p) \end{pmatrix}_x = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},$$

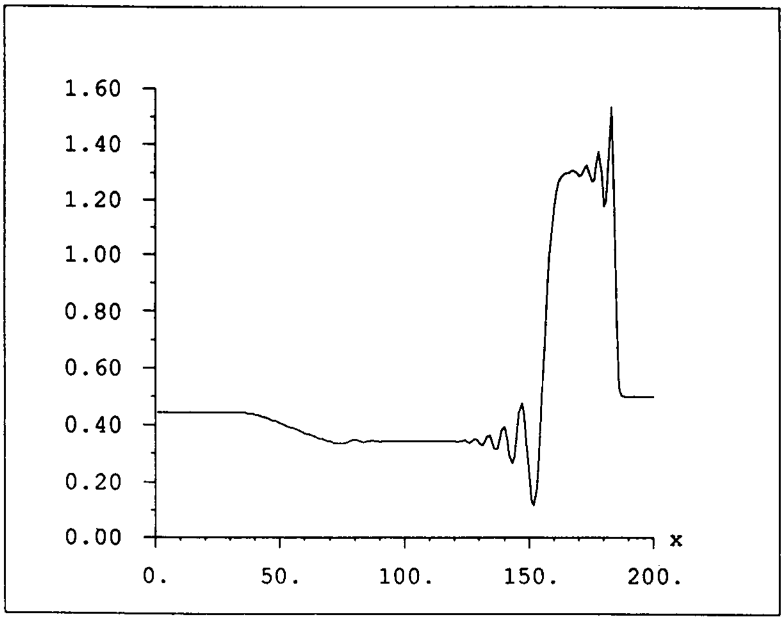
where $p = (\gamma - 1)(e - \frac{1}{2}\rho u^2)$ and $\gamma = 1.4$. The initial data are

$$u(x,0) = \begin{cases} \begin{pmatrix} 0.445 \\ 0.311 \\ 8.928 \end{pmatrix}, & x < 0, \\ \begin{pmatrix} 0.5 \\ 0 \\ 1.4275 \end{pmatrix}, & x > 0. \end{cases}$$

The solution consists of a shock followed by a contact discontinuity travelling to the right and a rarefaction wave going left. The solution at $t = 2$ is shown in Figure 6.3. There we used a CFL-number of 0.7.

This problem could be run quite easily, but in problems containing very strong shocks there is a possibility that the difference scheme produces an overshooting point which is outside the region where $c^2 > 0$. The square of the local sound speed c^2 is computed as $\gamma p/\rho$. In order to correct such an overshoot, it is necessary to replace the decomposition into characteristic fields with an approximate one, which we have done in the two-dimensional computations below.

(a)



(b)

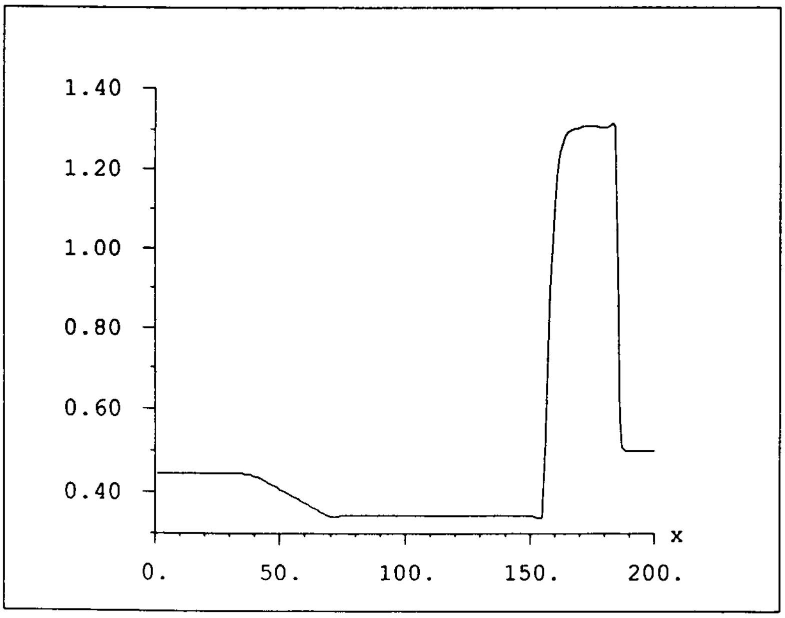


FIGURE 6.3

Solution of a one-dimensional shock tube problem with the Lax-Wendroff scheme.

(a) *the density without filter.*

(b) *the density with the filter in Algorithm* 2.1.

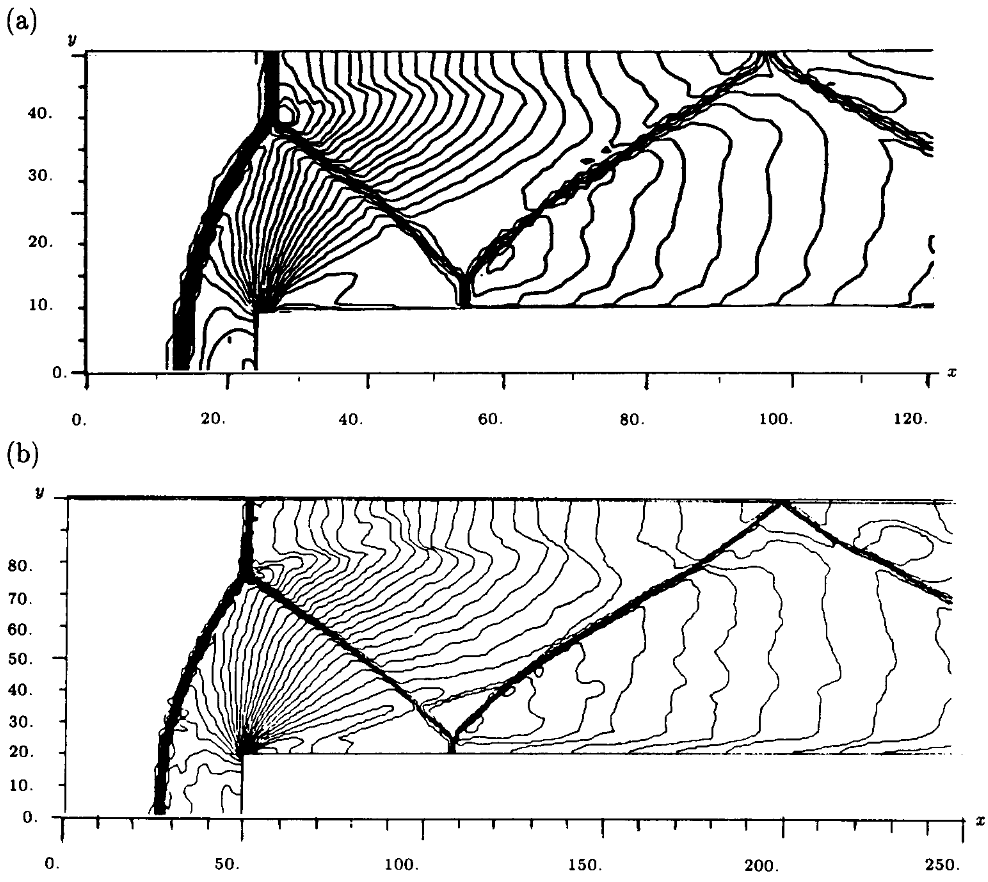


FIGURE 6.4
Solution of the forward facing step problem with the Lax-Wendroff scheme and the 2D filter.
(a) *the density with* 120×40 *points.*
(b) *the density with* 240×80 *points.*

We measured the CPU-time used to run this problem on a microVAX, with the following result:

TABLE 6.4

Method	CPU-seconds
Roe's method	11
ULT1	15
Lax-Wendroff+Filter	7
Lax-Wendroff	4

Roe's method is a first-order upwind scheme, described in [10], and ULT1 is a second-order TVD-scheme by Harten [3].

Let us also present some results from 2D-computations with the filter. The generalization to 2D is made by dimension splitting of the filter. That is, one time step is composed of the steps:

(a) Use a difference scheme to advance the solution to the next time level, not necessarily by dimension splitting.
(b) Apply the filter 4.1 in the x-direction.
(c) Apply the filter 4.1 in the y-direction.

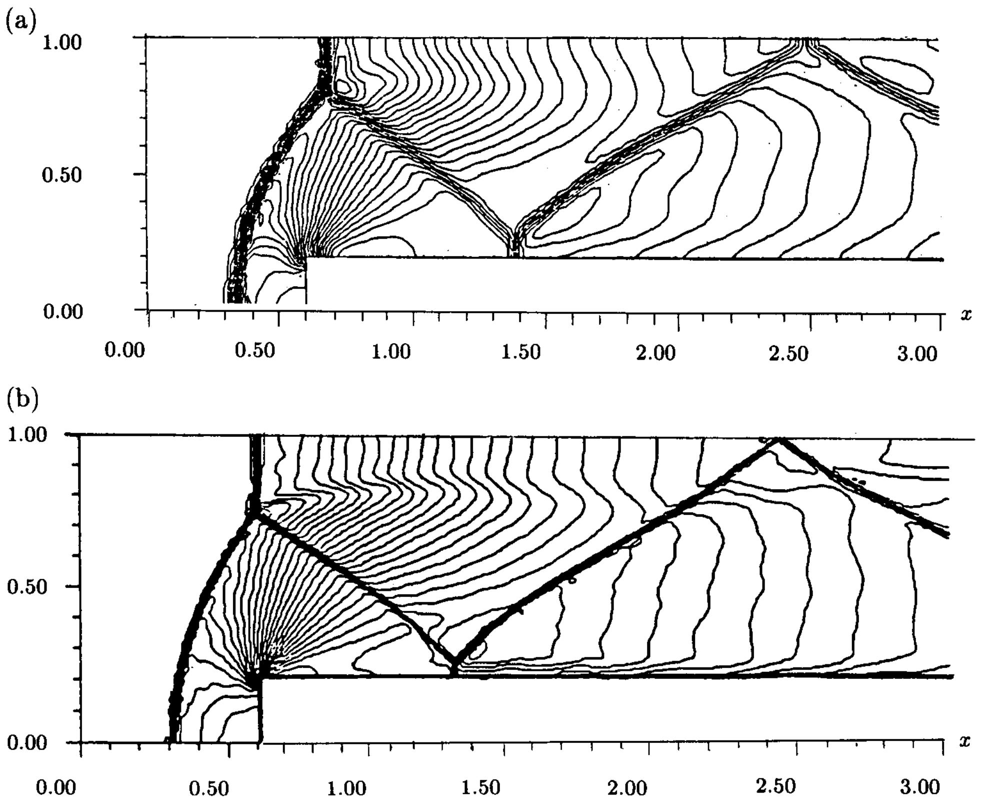
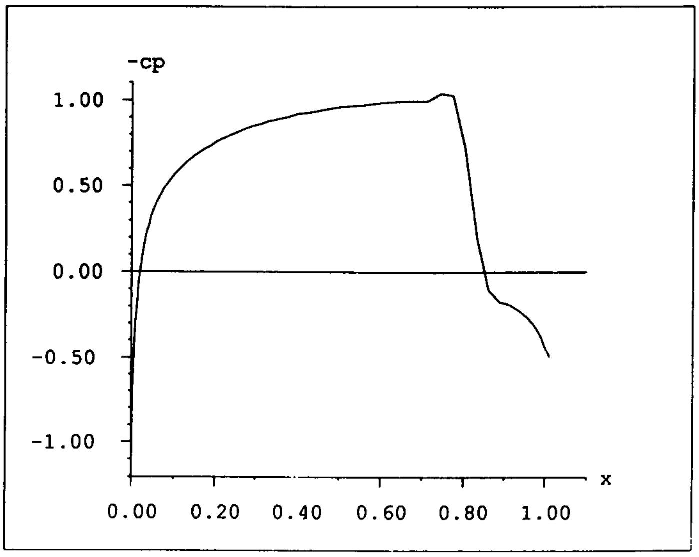


FIGURE 6.5
Solution of the forward facing step problem with the 4th-order *centered difference scheme and the* 2D *filter.*
(a) *the density with* 120×40 *points.*
(b) *the density with* 240×80 *points.*

We have here used the test problem "Mach-3 wind tunnel with a step" [13]. A gamma-law gas is fed in from the left into a channel of length 3 and width 1. The initial speed of the gas is Mach 3. At the lower side 0.6 from the right side of the channel, there is a step of height 0.2. The compressible Euler equations are solved for this problem, with $\gamma = 1.4$. The solution is computed at $t = 4$, with a CFL-number of ≈ 0.8. The step and the upper and lower walls of the channel are reflecting boundaries, with a Mach 3 uniform inflow on the left, and continuation boundary conditions on the right. In Figure 6.4 we show the solution computed with the dimension by dimension split Lax-Wendroff as the difference scheme. In Figure 6.5 we used a 4th-order centered difference scheme in space and 4th-order Runge-Kutta in time. The latter scheme gave initially an expansion shock emanating from the corner. In order to avoid this, we added a small amount of 4th-order dissipation to the scheme. The results presented in Figure 6.5 are computed with this dissipation added. Both these methods were unstable without the filter.

Finally, in order to illustrate the point that the filter can easily be implemented in an existing code, we have inserted it into a program that computes flow around an airfoil. The program was originally written by A. Rizzi and L.-E. Eriksson

(a)



(b)

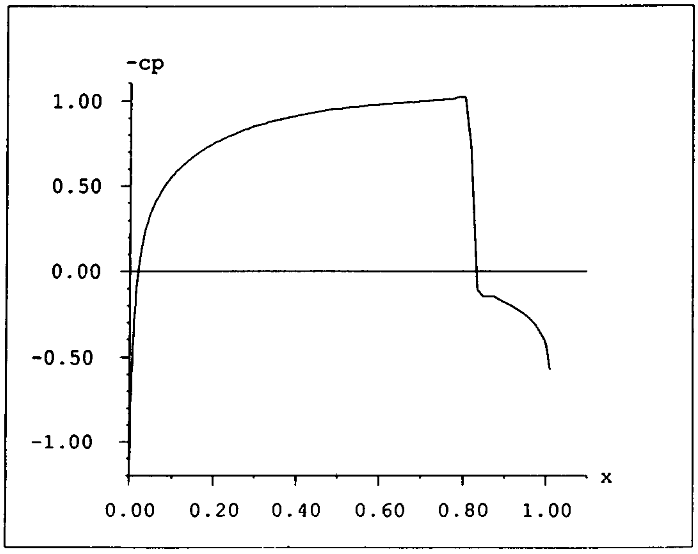


FIGURE 6.6

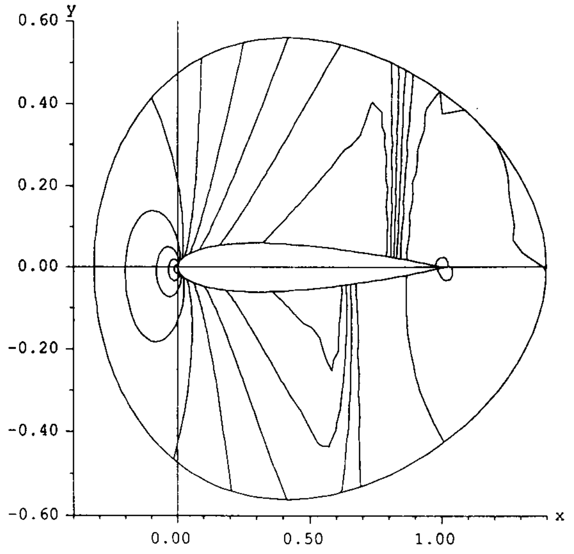
c_p-plot of the Euler solution on the upper side of the airfoil.

(a) *with 2nd-order artificial viscosity according to* [9].

(b) *The 2nd-order artificial viscosity is replaced by the 2D filter.*

[9] and uses 2nd-order centered difference approximation of the spatial derivatives with 2nd- and 4th-order artificial viscosity added. The 2nd-order dissipation term is introduced in order to damp oscillations at shocks. We removed the 2nd-order viscosity and inserted the filter. The time stepping is done with a Runge-Kutta type method. The grid used is of O-type and has 129×33 points. The freestream Mach number is 0.85 and the angle of attack $1°$. From this computation we present density contours and c_p plots. Figures 6.6a and 6.7a show the density contours and c_p along the upper surface of the airfoil computed by the original program. The 2nd-order artificial viscosity term is removed and the filter is introduced together with

(a)



(b)

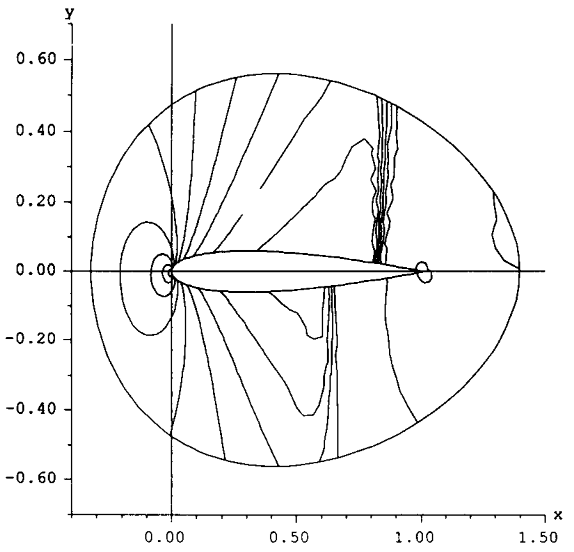


FIGURE 6.7

Density contours for the same problem as in Figure 6.6.
(a) *with 2nd-order artificial viscosity according to* [9].
(b) *the 2nd-order artificial viscosity is replaced by the 2D filter.*

the 4th-order viscosity, giving the result in Figures 6.6b and 6.7b. This problem has weak shocks and the artificial viscosity method also computes a sufficiently good solution. The point we wanted to make here is the simplicity of introducing the filter into existing computer codes.

Department of Mathematics
University of California at Los Angeles
Los Angeles, California 90024

1. P. COLELLA, "Glimm's method for gas dynamics," *SIAM J. Sci. Statist. Comput.*, v. 3, 1982, pp. 76–110.

2. D. GOTTLIEB, *Spectral Methods for Compressible Flow Problems*, Lecture Notes in Physics, No. 218 (Soubbaramayer and J. P. Boujot, eds.), Springer-Verlag, Berlin and New York, 1985, pp. 48–61.

3. A. HARTEN, "High resolution schemes for hyperbolic conservation laws," *J. Comput. Phys.*, v. 49, 1983, pp. 357–393.

4. A HARTEN & G. ZWAS, "Switched numerical Shuman filters for shock calculations," *J. Engrg. Math.*, v. 6, 1972, pp. 207–216.

5. P. LAX & B. WENDROFF, "Systems of conservation laws," *Comm. Pure Appl. Math.*, v. 13, 1960, pp. 217–237.

6. S. OSHER & S. CHAKRAVARTHY, "High resolution schemes and the entropy condition," *SIAM J. Numer. Anal.*, v. 21, 1984, pp. 955–984.

7. S. OSHER, A. HARTEN, B. ENGQUIST & S. CHAKRAVARTHY, "Some results on uniformly high-order accurate essentially nonoscillatory schemes," *J. Appl. Numer. Math.*, v. 2, 1986, pp. 347–377.

8. R. D. RICHTMYER & K. W. MORTON, *Difference methods for Initial Value Problems*, 2nd ed., Interscience, New York, 1967.

9. A. RIZZI & L.-E. ERIKSSON, "Computation of flow around wings based on the Euler equations," *J. Fluid Mech.*, v. 148, 1984, p. 45–71.

10. P. L. ROE, "Approximate Riemann solvers, parameter vectors, and difference schemes," *J. Comput. Phys.*, v. 43, 1981, pp. 357–372.

11. G. A. SOD, " A survey of several finite difference methods for systems of nonlinear hyperbolic conservation laws," *J. Comput. Phys.*, v. 27, 1978, pp. 1–31.

12. B. VAN LEER, "Towards the ultimate conservative difference scheme. V. A second-order sequel to Godunov's method," *J. Comput. Phys.*, v. 32, 1979, 101–136.

13. P. R. WOODWARD & P. COLELLA, "The numerical simulation of two-dimensional fluid flow with strong shocks," *J. Comput. Phys.*, v. 54, 1984, pp. 115–173.