DEPARTMENT OF MATHEMATICS

A Quasi-Conservative Version of the Semi-Lagrangian Advection Scheme.

by

A. Priestley

Numerical Analysis Report **2/92**

UNIVERSITY OF READING

0 Abstract

The semi-Lagrangian method is now, perhaps, the most widely researched algorithm in connection with Numerical Weather Prediction (NWP) codes. Monotonicity has been added to the basic method by the use of shape preserving interpolation and, more recently, by using ideas from Flux Corrected Transport (FCT). In this paper we describe how to make the scheme quasi-conservative. Although the lack of conservation of the semi-Lagrangian method is not widely regarded as a serious problem, for climate studies, where many tens of thousands of time-steps are needed, it could become so. The method proposed here is very cheap and hence is a viable proposition for addition to existing semi-Lagrangian codes. Making the scheme conservative as well as monotone gives the scheme shock capturing properties making the method much more useful in application areas outside of meteorology.

1 Introduction

Numerical schemes that follow characteristics backwards in time and then interpolate at their feet have a history stretching back to the very early days of computational fluid dynamics, Courant et al (1952). In the last decade these methods have been catching favour again. In the finite element literature there has been a body of work on the Lagrange-Galerkin method by, amongst others, Benqué et al (1982), Bercovier & Pironneau (1982), Douglas & Russell (1982), Lesaint (1977), Morton et al (1988), Russell (1980) & Süli (1988). Meanwhile the finite difference version of this idea has progressed in the form of the semi-Lagrangian method. A useful survey of this method can be found in Staniforth & Côté (1991). It is this form of the Lagrangian method that we shall be using here.

Consider the Cauchy problem for the scalar, linear advection equation for $u(\underline{x}, t)$:

$$u_t + \underline{a}.\underline{\nabla} u = 0, \qquad \underline{x} \in \mathbb{R}^{\mathrm{d}}, \quad \mathrm{t} > 0, \tag{1.1}$$

$$u(\underline{x}, 0) \;=\; u_0(\underline{x}), \tag{1.2}$$

where u_0 is the initial data and d is the number of dimensions. We can now define characteristics paths or trajectories, $\underline{X}(\underline{x}, s; t)$, in two ways, either as the solution to an ordinary differential equation,

$$\underline{X}(\underline{x}, s; s) \;=\; \underline{x}, \tag{1.3}$$

$$\frac{d\underline{X}(\underline{x}, s; t)}{dt} = \underline{a}(\underline{X}(\underline{x}, s; t), t); \tag{1.4}$$

or, if desired, as the solution of the integral equation

$$\underline{X}(\underline{x}, s; t) = \underline{x} + \int_s^t \underline{a}(\underline{X}(\underline{x}, s; \tau), \tau) d\tau.$$

In order to simplify the notation, for $t^{n+1} = t^n + \Delta t$, we will denote the foot of the characteristic path at time t^n to be at $\underline{x}$, usually termed the departure point, and its arrival point at time t^{n+1} to be at $\underline{y}$. In terms of the more general notation, these are

$$\underline{x} = \underline{X}(\underline{y}, t^{n+1}; t^n)$$

and

$$\underline{y} = \underline{X}(\underline{x}, t^n; t^{n+1}).$$

A unique (absolutely continuous) solution to equations (1.3) & (1.4) can be guaranteed under fairly mild assumptions about the velocity field $\underline{a}$, see Mizohata (1973) for example. The solution to the original partial differential equation (1.1) & (1.2) is now given by the relation

$$u(\underline{X}(\cdot, t, t + \tau), t + \tau) = u(\cdot, t), \tag{1.5}$$

which is just saying that the solution, u, is constant along a trajectory. Putting $\tau = \Delta t$ and simplifying the notation as above we have,

$$u(\underline{y}, t^{n+1}) = u(\underline{x}, t^n).$$

If $\underline{y}$, the arrival point, is then put equal to a grid point, $\underline{y}_j$ for example, the foot of the trajectory $\underline{x}$, the departure point, will not in general be a grid point and hence some sort of interpolation will need to be performed. Cubic interpolation seems to have been favoured in the development of the semi-Lagrangian method, giving a reasonable compromise between accuracy and computational effort. The two most popular are, in this context, the somewhat confusingly termed cubic-Lagrange interpolation and cubic-spline interpolation.

In Williamson & Rasch (1989) and Rasch & Williamson (1990) monotonicity was introduced to the semi-Lagrangian method. Although they experimented with many types of interpolation perhaps the most pleasing was that using Hermite-cubic interpolation (the monotonicity being obtained by limiting the derivative values). More recently Bermejo & Staniforth (1992), using ideas from the FCT schemes of Boris & Book (1973) and Zalesak (1979), have created a semi-Lagrangian scheme that achieves monotonicity whilst still using the cubic-Lagrange or cubic-spline interpolation, or for that matter any non-specialized interpolation.

In this paper we will now introduce conservation to the semi-Lagrangian method. This algorithm fits very neatly into the method of Bermejo & Staniforth (1992) and this paper will be referred to frequently in the following. Although the method presented here has more in common with the second method of introducing monotonicity mentioned above there is no reason why it couldn't be combined with the approach of Williamson & Rasch (1989) to again

achieve the aim of a monotonic and conservative scheme.

In the remainder of this section the basis of the monotonic algorithm described in Bermejo & Staniforth (1992) will be given. In the following section the algorithm for recovering conservation will be given. Section 3 contains two test problems that will be solved with the currently proposed method and then finally, in section 4, a summary of the work will be given.

The first stage of any Lagrangian scheme is the solution of the trajectory problem, equations (1.3) and (1.4). In principle any ODE solver could be used to do this. In the first test case to be done later the exact trajectories are known and used. In the second, nonlinear, problem Euler's forward difference method is used to determine the the departure point of the grid point. Since we are only doing a comparative study, this is adequate. In practice this $O(\Delta t)$ time-stepping is not regarded as adequate, however, see Staniforth & Côté (1991), and $O(\Delta t^2)$ time-stepping schemes are generally preferred. Working in spherical coordinates can also raise problems, see Ritchie (1987).

Having found the departure point we must now interpolate the value of the solution. The most common types used have already been mentioned and the mechanics of these methods can be found in most books on numerical methods, see Johnson & Riess (1977) for example. Here we will only be using cubic-spline interpolation (bicubic-splines in 2-dimensions). This is purely because this interpolation comes out well in the tests of Bermejo & Staniforth

(1992). The results presented here are all compared with like schemes and so the precise choice of interpolation is somewhat arbitrary and a cheaper and/or more local interpolation could be used. The only difference would then be a slight increase in the errors presented.

To obtain a monotone scheme via the FCT approach we need two approximations to the solution at the new time-level. These are usually referred to as the high-order solution, U^H, and the low-order solution U^L. These terms are slightly misleading, since the only constraint on these solutions is that the 'low-order' solution be monotone, or more precisely in our context it should introduce no new extrema. (This is a rather cumbersome phrase and so we shall continue to use the word monotone with its extension to higher order schemes and multidimensions being understood). We could therefore use the monotone Hermite-cubic interpolation of Williamson & Rasch (1989) as either our high-order scheme or our low-order scheme. As we have mentioned previously, we will use the cubic-spline interpolation for our high-order scheme and linear interpolation for the low-order, monotone, scheme. From these two solutions we then define our monotone solution, U^M at the point k, to be given by,

$$U_k^M = \alpha_k U_k^H + (1 - \alpha_k) U_k^L \tag{1.6}$$

with

$$0 \leq \alpha_k \leq 1, \tag{1.7}$$

where the α_k's are yet to be chosen. Clearly the objective is to find α's as large as possible whilst maintaining monotonicity. A solution is guaranteed to exist, corresponding to $\alpha_k = 0$, because of the monotone nature of U^L. Upper and

lower bounds are now placed on eq. (1.6) to ensure a monotone solution. Denote by U^n the monotone solution from the previous time-level and by $\{U^n, k\}$ the set of solution values at points surrounding the departure point of the grid point $\underline{y}_k$. This is $\underline{y}(\underline{x}_k)$ in our previous notation. In the notation of Bermejo & Staniforth (1992) $\{U^n, k\} \equiv \{U_{k1}, U_{k2}, U_{k3}, U_{k4}\}$. We now take the highest value of α_k that satisfies both (1.7) and the condition

$$\min(\{U^n, k\}, U_k^L) \leq \alpha_k U_k^H + (1 - \alpha_k) U_k^L \leq \max(\{U^n, k\}, U_k^L). \tag{1.8}$$

Equation (1.8) only differs from the one given in Bermejo & Staniforth (1992) in that the value of the low-order solution has been included in the bounds. When calculated by linear interpolation from the values $\{U^n, k\}$ for pure advection problems it will clearly not affect anything. For more general problems, i.e. when source terms are present or we are solving a nonlinear system, it is necessary to include this value. We now proceed to discuss the recovery of monotonicity.

2 Recovering Conservation

We shall refer to the maximum values of α that satisfy constraints (1.7) and (1.8) as $\{\alpha_k^{\max}\}$. Sub-optimal values of the α_k's can now be chosen to try and make the scheme conservative. That is we choose α_k's such that

$$0 \leq \alpha_k \leq \alpha_k^{\max}$$

whilst at the same time trying to enforce

$$\int U^M(\underline{x})d\underline{x} = \int U^0(\underline{x})d\underline{x} = C, \text{ say.} \tag{2.9}$$

In general, of course, the best we can do is to minimize the difference between the two quantities. In the examples given later we have precisely the situation stated in (2.9) but in other problems it may be necessary to take into account source terms and boundary conditions. There are many ways this problem can be solved, by linear programming methods for example, but the following algorithm is found to be a very efficient and direct way of obtaining a solution. The efficiency of the algorithm will be demonstrated in the results section and also the fact that in using sub-optimal α's we sacrifice comparatively little accuracy compared with the introduction of the monotonicity.

Firstly we define S_i to be the area associated with node i. With a regular mesh this is just h in 1-dimension and h^2 in two-dimensions. A more general case is illustrated in figure (1). Secondly, let

$$\beta_i = (U_i^H - U_i^L)S_i.$$

The problem now is to maximize the α's subject to the following condition arising from eqs. (1.6) and (2.9),

$$\sum_i \alpha_i(U_i^H - U_i^L)S_i = C - \sum_i U_i^L S_i = C^*.$$

Assume that

$$\sum_i \alpha_i^{\max} \beta_i > C^*.$$

If this is not the case then the definitions are just changed so that

$$\beta_i \longrightarrow -\beta_i$$

$$C^* \longrightarrow -C^*.$$

Step 1

if $\beta_i \leq 0$ then $\alpha_i = \alpha_i^{\max}$

$\text{iflag}(i) = 1$

otherwise

$\alpha_i = 0$

$\text{iflag}(i) = 0.$

Step 2 Define a surplus

$$= C^* - \sum_{\text{iflag}(k)=1} \alpha_k \beta_k.$$

Step 3 Define the average value of α

$$\alpha_{AV} = \frac{\text{surplus}}{\sum_{\text{iflag}(k)=0} \beta_k}.$$

Step 4 if $\alpha_{AV} < \alpha_k^{\max}$ everywhere $\text{iflag}(k) = 0$

then for these k set $\alpha_k = \alpha_{AV}$.

END

Step 5 else for those k that satisfy both

$\text{iflag}(k) = 0$ and $\alpha_{AV} > \alpha_k^{\max}$

put

$$\alpha_k = \alpha_k^{\max}$$

$$\text{iflag}(k) = 1.$$

Step 6 GOTO 2.

If the surplus is negative then there is no conservative solution and the best solution, as regards conservation, is given by the initial setup of the α's in step 1.

In step 4, where the successful algorithm finishes, there is a certain amount of freedom in what we do. Here, we have just set the remaining α's to the average value. This is quick and is certainly a reasonable thing to do. However, if desired, these extra degrees of freedom, which correspond to the solution lying on a plane in the linear programming case, could be used to try and achieve some other goal. In the next section we shall monitor the second moment of the solution, which ideally should also be conserved. This, of course, is not conserved by this algorithm but these extra degrees of freedom could be used to try and achieve conservation of the second moment or indeed any other conserved quantity.

3 Results

The first problem is that of the slotted cylinder. This is Example 3 of Bermejo & Staniforth (1992). On the domain $[-\frac{1}{2},\frac{1}{2}]\times[-\frac{1}{2},\frac{1}{2}]$, with a uniform grid of size $h=\frac{1}{100}$, we define the slotted cylinder. It is centred at $(-\frac{1}{4},0)$ with radius $15h$. Its height is 4. The slot has width $6h$ and length $22h$. The initial data is shown in figure (2).

This profile is then advected by a given velocity field, $\underline{a}=\omega(-y,x)$ which causes the cylinder to rotate with angular velocity ω. The value of ω is taken to be $0.3636\times10^{-4}s^{-1}$. This corresponds to one rotation every twenty days. Taking a time-step of $\Delta t=1800s$ a revolution is completed every 96 time-steps. Six revolutions are performed.

Following Bermejo & Staniforth (1992) we define some errors for this problem. The first moment of the approximate solution U, normalised by the first moment of the exact solution u, is given by

$$RFM=\frac{\int_\Omega U(\underline{x},t_{output})d\underline{x}}{\int_\Omega u(\underline{x},0)d\underline{x}},$$

and the second moment is similarly defined by

$$RSM=\frac{\int_\Omega U^2(\underline{x},t_{output})d\underline{x}}{\int_\Omega u^2(\underline{x},0)d\underline{x}}.$$

A fairly obvious discretization of the above integrals is used. The maximum and minimum values are also given. Again following Bermejo & Staniforth (1992) and

Takacs (1985) we write the discrete l_2 error as

$$||u - U||_{l_2}^2 = \frac{1}{K}\sum_{k=1}^{K}(u_k^n - U_k^n)^2,$$

where K is the total number of points. This is then split into the dissipation error (DISSER) and the dispersion error (DISPER), i.e. we can write

$$||u - U||_{l_2}^2 = DISSER + DISPER$$

where

$$DISSER = [\sigma(u) - \sigma(U)]^2 + (\overline{u} - \overline{U})^2 \tag{3.10}$$

$$DISPER = 2(1 - \rho)\sigma(u)\sigma(U). \tag{3.11}$$

In eqs. (3.10) and (3.11) the overbar denotes an average value, σ is the standard deviation (σ^2 the variance). The correlation between u and U on the mesh is denoted by ρ.

Results are presented for three versions of the semi-Lagrangian method. All use cubic-spline interpolation. The first is just the basic method, the second is the quasi-monotone version (as in Bermejo & Staniforth (1992)) and the third is the new quasi-monotone and conservative method. Plots of the results after 6 revolutions are shown in figures (3), (4) and (5) respectively. The figures indicate that the monotone version of the method lives up to its name. The slot remains well defined in both of the monotonic implementations. Indeed

there is no visible difference between the quasi-monotone solution, fig. (4), and the quasi-monotone and conservative solution, fig. (5). More objective comparisons can be made from tables (i), (ii) and (iii). Obvious statements that can be made after looking at these tables are that the monotonic implementations are indeed monotone and that the conservative version is conservative. The results in table (ii) appear to be slightly better than those for the identical problem in Bermejo & Staniforth (1992). This is almost certainly entirely due to the fact that we made use of the known trajectory paths in this problem for the calculations presented here. Looking at the errors DISSER and DISPER, the two quantities making up the discrete l_2 error norm, we can see how the modifications to the basic algorithm have changed the accuracy.

Firstly, comparing the results for the basic method and the quasi-monotone method, tables (i) and (ii), we see that the dissipation error (DISSER) has increased by a factor of just over 4. The larger dispersion error (DISPER), that dominates the l_2 error, rises by between about 10% and 20%. Comparing tables (ii) and (iii) to see how the introduction of the quasi-conservative algorithm affects accuracy, we see an approximate rise in the dissipation error of 10% and in the larger dispersion error of less than 1%. This demonstrates that very little accuracy is sacrificed in achieving conservation compared to what has already been lost due to the introduction of monotonicity.

As to cost, an increase of $\frac{9}{10}$% in CPU time was needed, compared to calculating the quasi-monotone scheme. This was for a problem where the tra-

jectories were calculated once and then stored and the cubic-spline interpolation was able to take advantage of the interpolation points being fixed. Timings can be found in Table (iv) for both the results obtained here on a SUN Sparc 1 and for results obtained by Gravel & Staniforth (1992) on a CDC 4680, with 32-bit precision, and a Cray XMP, with 64-bit precision. Times are for 192 time-steps of the above problem. The cost seems to be very machine dependent but it must be emphasised that the code used for the Cray had not been specially written and hence it may not have vectorized as fully as it might. It is believed that this timing could be reduced significantly.

The second test problem is that of the one-dimensional inviscid Burgers' equation. This equation has been solved before using the semi-Lagrangian method in Kuo & Williams (1990), but here we will use the same problem as Bermejo & Staniforth (1992). We are required to find the solution to

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0 \quad x,t \in [-1,1] \times [0,T], \tag{3.12}$$

$$u(x,0) = \frac{1}{4} + \frac{1}{2}\sin \pi x, \tag{3.13}$$

$$u(-1,t) = u(1,t) \quad \forall t.$$

The initial data, eq. (3.13), is depicted in figure (6). We do not take the velocity from eq. (3.12) to be u but rather define it to be $a(u)$ where $a(u) = \partial f/\partial u$ and f is the flux function defined by $f(u) = \frac{1}{2}u^2$ for Burgers' equation. This is then

approximated by a simple central difference as

$$a(u_i) = \frac{f(u_{i+1}) - f(u_{i-1})}{u_{i+1} - u_{i-1}}.$$

In smooth regions of the flow this should be a good approximation to u, whereas in non-smooth regions it should give a good approximation to the shock speed. The velocity thus generated is then used in Euler's method to determine the departure point, i.e.,

$$y_i = x_i - a(u_i)\Delta t.$$

For this problem it was not felt necessary to use a more sophisticated approach. Eighty grid points were used giving a value of $h = 1/40$. Figure (7) shows the result at $T = 0.65$. All three methods give virtually identical results at this time, before the shock develops, and we only show the result for the quasi-monotone and conservative method. This solution was obtained in 10 time-steps implying a maximum CFL number of just under 2. The next output time in Bermejo & Staniforth (1992) is $T = 0.9$. However, we found very little to distinguish the methods at this time: the basic method had a slight oscillation, but was otherwise identical to the other two solutions.

In Bermejo & Staniforth (1992) a shock fitting technique was used to obtain their results. It should be stressed that when we later get rather poor results for the quasi-monotone semi-Lagrangian method the shock fitting has not been used and it is hence not comparable with the scheme presented by Bermejo & Staniforth (1992). We reject the use of shock fitting for two main reasons. Firstly, although shock fitting may work very well for the one-dimensional Burgers' equa-

tion (3.12), for a one-dimensional system of equations, the Euler equations of gas dynamics say, we already face difficulties. Numerically, in this situation the Rankine-Hugoniot equations predict three separate shock speeds, the conservation laws three separate shock positions. These difficulties can be overcome, see Morton & Sweby (1987), although it is not a very aesthetically pleasing approach. In two dimensions things become even worse in that it is even a problem to detect shocks unless they are lying along, or close to, grid lines. If tolerance s are reduced to the extent that most of the shock is recognized it usually implies that 'ghost' shocks will also be caught. A notable exception to this rule would appear to be that of Moretti (1987).

However, it is known that with conservation we are guaranteed the correct shock position, see Lax & Wendroff (1960), while monotonicity ensures convergence, see Sanders (1983). An entropy condition is needed to finally complete the picture of convergence to the correct physical solution, Harten et al (1976). Thus, by adding conservation to the quasi-monotone scheme we are freed from the need to fit shocks and hence the scheme becomes much more widely applicable even in higher dimensions. To demonstrate the advantages of having conservation the previous problem was run to $T = 1.5$. This time was reached in 30 time-steps giving a maximum CFL number of 1.5. A control solution, obtained with the quasi-monotone and conservative method on a grid with 2000 points, is shown in figure (8). Figures (9), (10) and (11) show the solution as calculated by the basic method, the quasi-monotone method, and the quasi-monotone and conservative method. At first glance all three methods have performed well,

with the basic semi-Lagrangian method only suffering from a slight oscillation behind the shock. However, on closer inspection we see that both the basic and quasi-monotone methods have not transported the shock far enough. The quasi-monotone and conservative version has got the shock in the right place. This is confirmed by checking the function values to see where the shock lies. Allowing one intermediate point, in the control solution the shock lies somewhere in the interval $[-0.626, -0.624]$. The basic and quasi-monotone semi-Lagrangian methods both predict a shock position somewhere in the interval $[-0.7, -0.65]$ whilst the quasi-monotone and conservative algorithm correctly predicts the shock to be somewhere in the interval $[-0.65, -0.6]$. We must stress again that this is not the implementation of the quasi-monotone semi-Lagrangian scheme advocated for this problem by Bermejo & Staniforth (1992). However, we believe that the introduction of monotonicity used here to be a much more general and robust procedure for obtaining correct shock positions than the shock fitting technique.

The extra cost involved in calculating the conservative solution, over and above that of calculating the monotone solution, represents an extra $\frac{3}{4}\%$ of CPU time. The reduction over the previous case is because the trajectories have to be calculated at each time-step now and the interpolation routine cannot take advantage of the interpolation points being fixed. Clearly if a more expensive trajectory solver were used, as is likely to be the case in practice, then the relative cost will go down slightly, just as it will go up a little if a cheaper interpolation is used. In either event conservation is achieved at very low cost.

4 Treatment of Source and Boundary Terms

In both the test cases performed here the equations were homogeneous and the domains had periodic boundary conditions. This meant that there was a constant, denoted by C in eq. (2.9), that equalled the amount of the conserved quantity u for all time. We now consider the non-homogeneous problem corresponding to (1.1),

$$u_t + \underline{a}.\underline{\nabla} u = b(\underline{x}, t),$$

with the initial conditions still supplied by (1.2). The constant C now becomes a function of time, $C(t)$. For our purposes we now need to replace the C used in the algorithm described previously by its value at $t = t^{n+1}$. That is, we take

$$C \equiv C^{n+1} = C^n + \int_{t^n}^{t^{n+1}} \int_\Omega b(\underline{x}, t) d\underline{x} dt, \tag{4.14}$$

where Ω represents the spatial domain and C^0 is just given by the spatial integral of the initial data $u_0(\underline{x})$.

In some situations $b(\underline{x}, t)$ is a known function, where a pollutant is released at a known rate for example see Pudykiewicz (1989), and it may be possible to evaluate the integral in (4.14) analytically. More generally the function b will also depend upon the solution u and the integral will have to be approximated. A simple and obvious choice would be to put

$$C^{n+1} = C^n + \Delta t \int_\Omega b(\underline{x}, t^n) d\underline{x}.$$

If this is a part of some iteration, i.e. there is already a guess for u^{n+1} then clearly a more accurate treatment of the integral may be used.

Boundary terms affecting conservation can be treated in an entirely analogous manner, a surface integral of the flux function now arising in (4.14).

5 Summary

In this paper we have shown how the semi-Lagrangian method can be made conservative. Although the procedure presented here falls in most naturally with the monotone scheme of Bermejo & Staniforth (1992) there is no reason why it should not be used with the monotone schemes of Rasch & Williamson (1990). Conservation is achieved at very little expense above that needed for the calculation of the monotone scheme.

This makes it a practicable addition to semi-Lagrangian codes for climate modelling, or other applications, where loss (or gain) in conserved quantities can become an issue.

In addition, the shock capturing properties endowed upon the semi-Lagrangian method due to the addition of conservation and monotonicity makes the scheme much more widely applicable.

References

Benqué, J.P., Labadie, G. & Ronat, J., 1982: "A New Finite Element Method for the Navier-Stokes Equations Coupled with a Temperature Equation." Proc. 4^{th} Int. Symp. on Finite Element Methods in Flow Problems (Ed. T. Kawai), North-Holland, Amsterdam, Oxford, New York, pp. 295-301.

Bercovier, M. & Pironneau, O., 1982: *"Characteristics and the Finite Element Method."* Proc. 4^{th} Int. Symp. on Finite Element Methods in Flow Problems (Ed. T. Kawai), North-Holland, Amsterdam, Oxford, New York, pp. 67-73.

Bermejo, R. & Staniforth, A., 1992: "On the Conversion of semi-Lagrangian Advection Schemes to Quasi-Monotone Schemes." Mon. Wea. Rev., July 1992.

Boris, J.P. & Book, D.L., 1973: "Flux Corrected Transport, I, SHASTA, A Fluid Transport Algorithm that Works." J. Comp. Phys., **11**, pp. 38-69.

Courant, R., Isaacson, E. & Rees, M., 1952: "On the solution of Nonlinear Hyperbolic Differential Equations by Finite Differences." Comm. Pure Appl. Maths., **5**, pp. 243-255.

Douglas Jr., J. & Russell, T.F., 1982: "Numerical Methods for Convection-Dominated Diffusion Problems based on combining the Method of Characteristics with Finite Element or Finite Difference Procedures." SIAM J. Numer. Anal., **19**, pp. 871-885.

Gravel, S. & Staniforth, A., 1992: Private Communication.

Harten, A., Hyman, J.M. & Lax, P.D., 1976: "On Finite-Difference Approximations and Entropy Conditions for Shocks." Comm. Pure Appl. Maths., **29**, pp. 297-322.

Johnson, L.W. & Riess, R.D., 1977: "Numerical Analysis." Addison-Wesley Publishing Company.

Kuo, H-C. & Williams, R.T., 1990: "Semi-Lagrangian Solutions to the Inviscid Burgers' Equation." Mon. Wea. Rev., **118**, pp. 1278-1288.

Lax, P.D. & Wendroff, B., 1960: "Systems of Conservation Laws." Comm. Pure Appl. Maths., **13**, pp. 217-237.

Lesaint, P., 1977: "Numerical Solution of the Equation of Continuity." Topics in Numerical Analysis III (Ed. J.J.H. Miller), Academic Press, London, New

York, San Francisco, pp. 199-222.

Mizohata, S., 1973: "The Theory of Partial Differential Equations." Cambridge University Press.

Moretti, G., 1987: "A Technique for Integrating Two-Dimensional Euler Equations." Comp. Fluids, **15**, pp. 59-75.

Morton, K.W. & Sweby, P.K., 1987: "A Comparison of Flux Limited Difference Methods and Characteristic Galerkin Methods for Shock Modelling." J. Comp. Phys., **73**, pp. 203-230.

Morton, K.W., Priestley, A. & Süli, E.E., 1988: "Stability of the Lagrange-Galerkin Method with Non-Exact Integration." RAIRO Modél. Math. Anal. Numér., **22**, no. 4, pp. 625-653.

Pudykiewicz, J., 1989: "Simulation of the Chernobyl dispersion with a 3-D Hemispheric Tracer Model." Tellus, **41B**, pp. 391-412.

Rasch, P.J. & Williamson, D.L., 1990: "On Shape-Preserving Interpolation and semi-Lagrangian Transport." SIAM J. Sci. Stat. Comp., **11**, pp.656-687.

Ritchie, H., 1987: "Semi-Lagrangian Advection on a Gaussian Grid." Mon. Wea. Rev., **115**, pp. 608-619.

Russell, T.F., 1980: "Time Stepping along Characteristics with Incomplete Iteration for a Galerkin Approximation of Miscible Displacement in Porous Media." Ph.D. Thesis, University of Chicago.

Sanders, R., 1983: "On Convergence of Monotone Finite Difference Schemes with Variable Spatial Differencing." Math. Comp., **40**, pp. 91-106.

Staniforth, A. & Côté, J., 1991: "Semi-Lagrangian integration schemes for atmospheric models — a review." Mon. Wea. Rev., **119**, pp. 2206-2223.

Süli, E.E., 1988: "Convergence and Nonlinear Stability of the Lagrange-Galerkin Method for the Navier-Stokes Equations." Numer. Math., **53**, pp. 459-483.

Takacs, L.L., 1985: "A two-step Scheme for the Advection Equation with Minimized Dissipation and Dispersion Errors." Mon. Wea. Rev., **113**, pp. 1050-1065.

Williamson, D.L. & Rasch, P.J., 1989: "Two-dimensional semi-Lagrangian Transport with Shape-Preserving Interpolation." Mon. Wea. Rev., **117**, pp. 102-129.

Zalesak, S.T., 1979: "Fully Multidimensional Flux-Corrected Transport Algorithms for Fluids." J. Comp. Phys., **31**, pp. 335-362.

List of Tables

List of Figures

$N°\Delta t$	RFM	RSM	MAX	MIN	DISSER	DISPER
96	1.00002	0.93879	4.51856	-0.54069	9.2896×10^{-4}	3.5973×10^{-2}
192	0.99996	0.92709	4.55160	-0.63111	1.3266×10^{-3}	4.3818×10^{-2}
288	0.99989	0.91948	4.53480	-0.62486	1.6244×10^{-3}	4.8731×10^{-2}
384	1.00006	0.91357	4.48548	-0.65015	1.8788×10^{-3}	5.2231×10^{-2}
480	1.00027	0.90860	4.49618	-0.64298	2.1083×10^{-3}	5.4920×10^{-2}
576	1.00032	0.90423	4.50556	-0.61775	2.3204×10^{-3}	5.7112×10^{-2}

Table i: Errors for slotted cylinder problem using semi-Lagrangian method with cubic-spline interpolation.

$N°\Delta t$	RFM	RSM	MAX	MIN	DISSER	DISPER
96	1.00387	0.873845	3.99947	0.0	4.1257×10^{-3}	4.3240×10^{-2}
192	1.00452	0.85508	3.99849	0.0	5.5065×10^{-3}	5.0314×10^{-2}
288	1.00563	0.84259	3.99429	0.0	6.5539×10^{-3}	5.4966×10^{-2}
384	1.00689	0.83307	3.99113	0.0	7.4235×10^{-3}	5.8595×10^{-2}
480	1.00801	0.82514	3.99011	0.0	8.1949×10^{-3}	6.1660×10^{-2}
576	1.00911	0.81842	3.98925	0.0	8.8848×10^{-3}	6.4380×10^{-2}

Table ii: Errors for slotted cylinder problem using semi-Lagrangian method with cubic-spline interpolation and monotonicity.

$N°\Delta t$	RFM	RSM	MAX	MIN	DISSER	DISPER
96	1.0	0.86936	3.99990	0.0	4.4037×10^{-3}	4.3300×10^{-2}
192	1.0	0.84970	3.99842	0.0	5.8975×10^{-3}	5.0384×10^{-2}
288	1.0	0.83560	3.99398	0.0	7.1163×10^{-3}	5.5111×10^{-2}
384	1.0	0.82431	3.99062	0.0	8.1843×10^{-3}	5.8790×10^{-2}
480	1.0	0.81484	3.98967	0.0	9.1441×10^{-3}	6.1885×10^{-2}
576	1.0	0.80657	3.98757	0.0	10.031×10^{-3}	6.4621×10^{-2}

Table iii: Errors for slotted cylinder problem using semi-Lagrangian method with cubic-spline interpolation, monotonicity and conservation.

Method	SUN Sparc 1	CDC 4680	Cray XMP
Spline only	597.5	88.4	17.3
Spline + monotonicity	736.1	148.3	18.3
Spline + monotonicity + conservation	742.6	155.8	23.5
Percentage overhead	0.9	5.1	29

Table iv: CPU times for the various schemes on different machines.

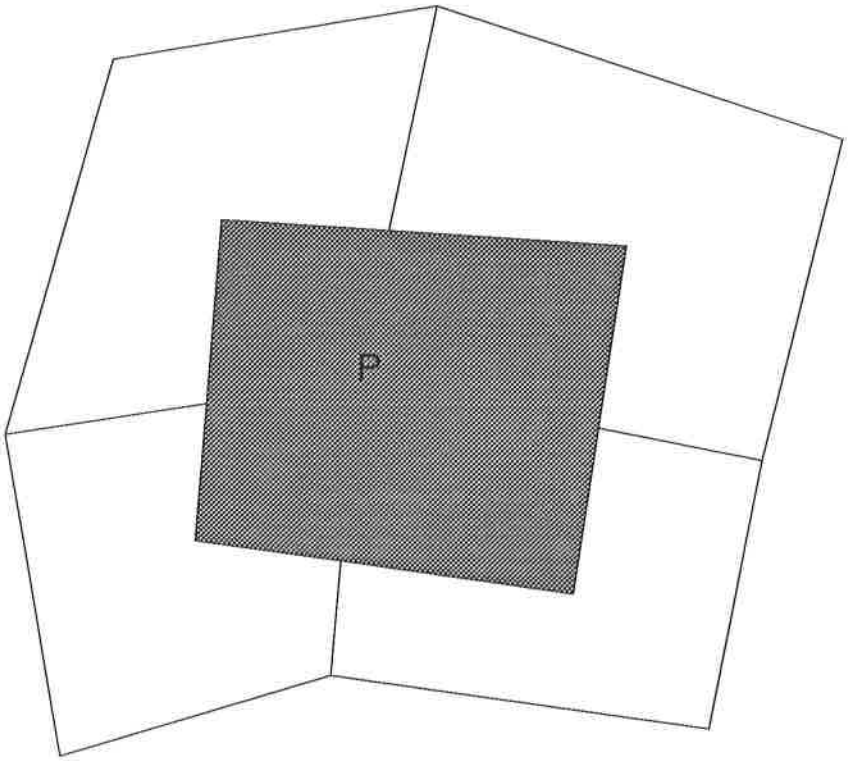

Figure 1: The shaded area shows the value of S_i for the point P on a non-uniform mesh.

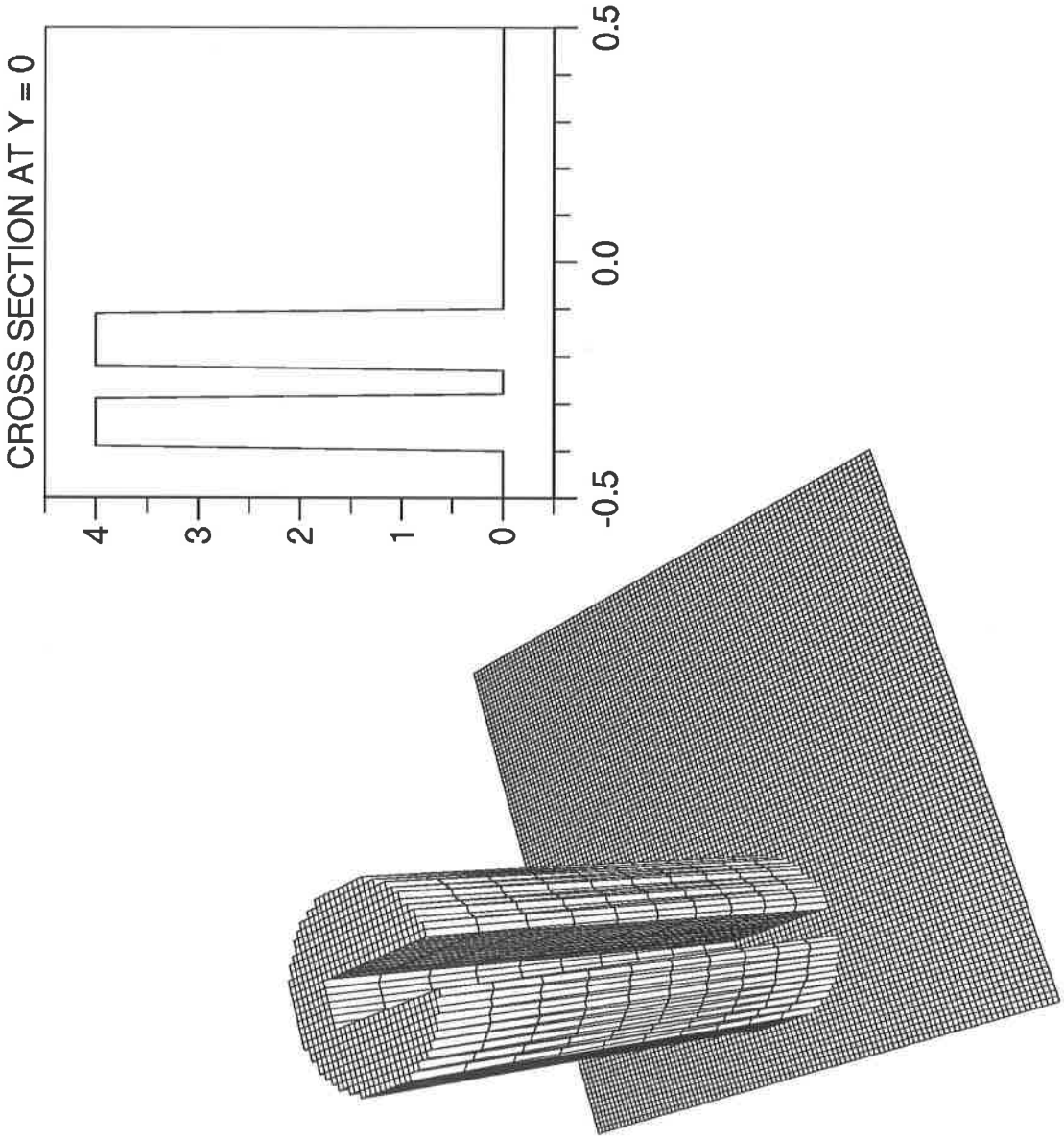

Figure 2: Initial data for the rotation of the slotted cylinder problem.

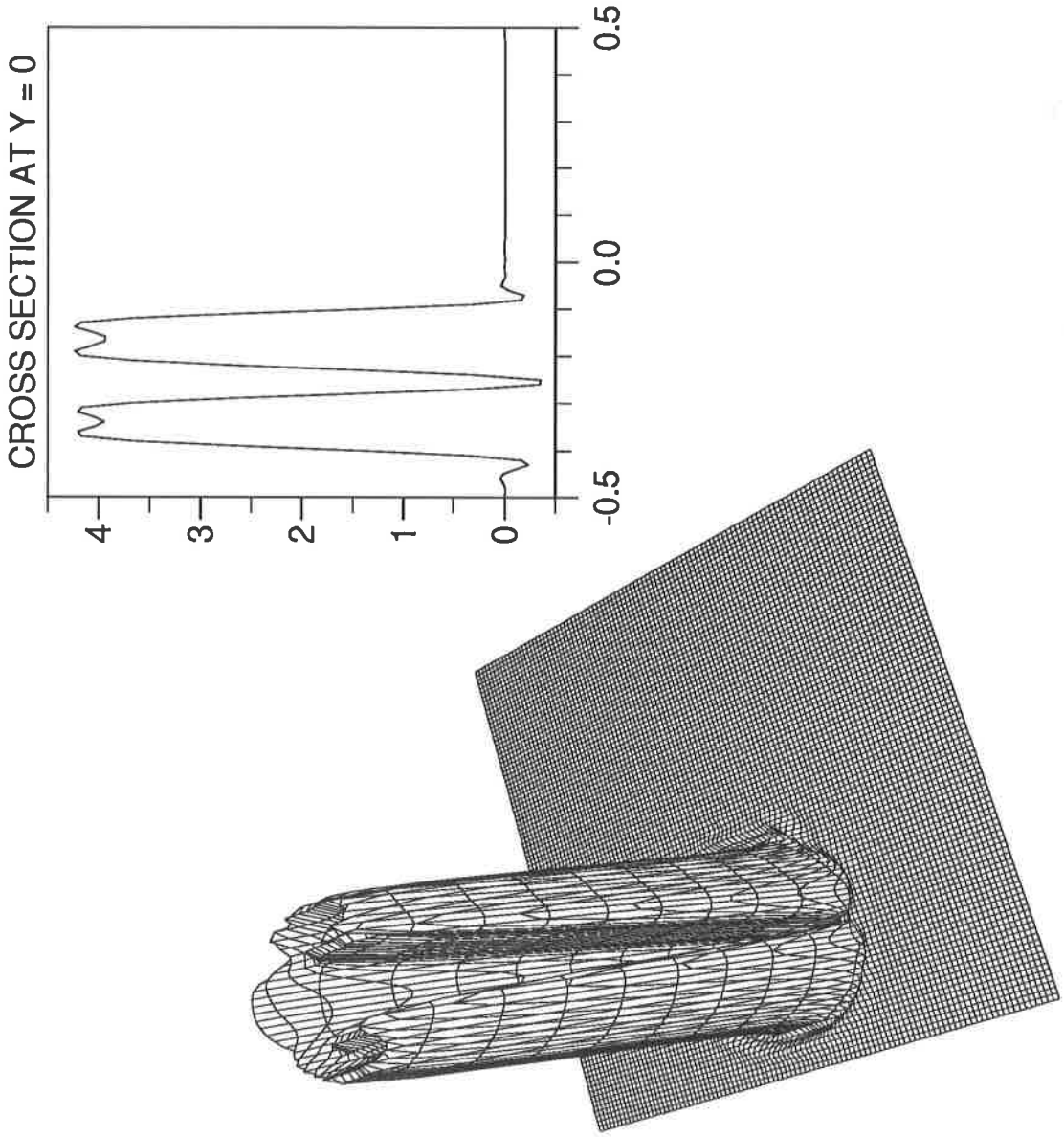

Figure 3: Results after 6 revolutions for semi-Lagrangian method with cubic-spline interpolation.

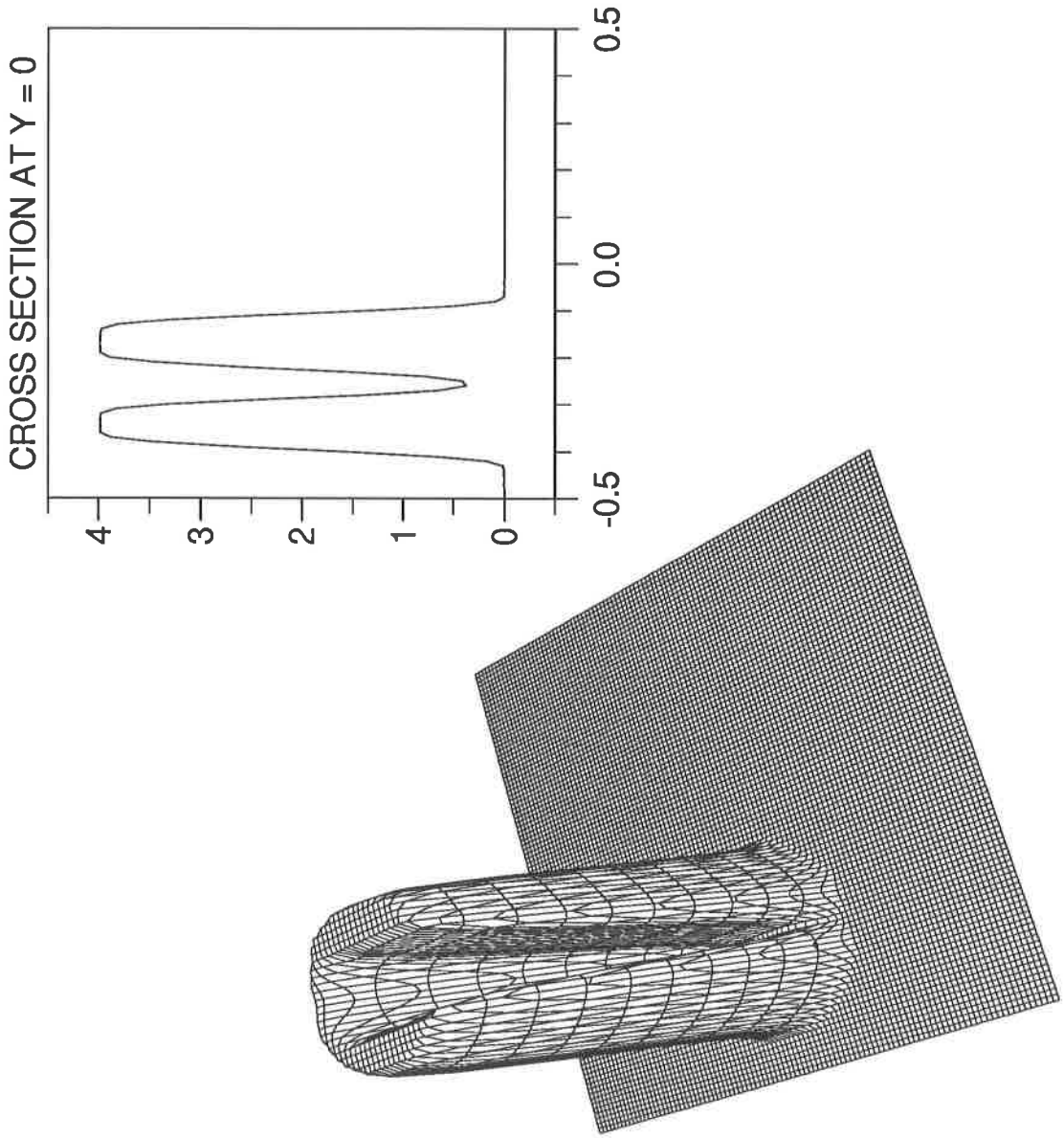

Figure 4: Results after 6 revolutions for monotone semi-Lagrangian method with cubic-spline interpolation.

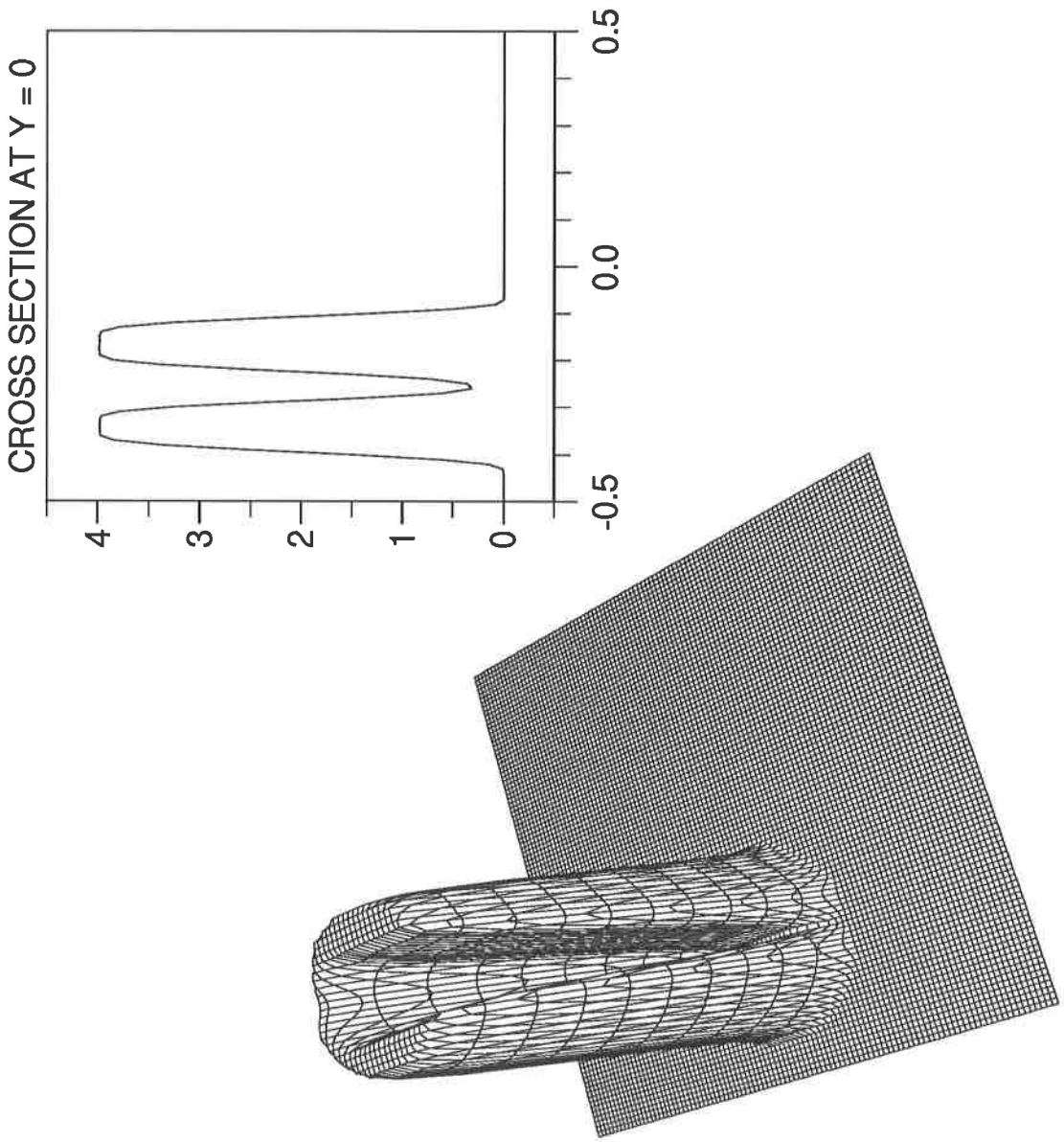

Figure 5: Results after 6 revolutions for monotone and conservative semi-Lagrangian method with cubic-spline interpolation.

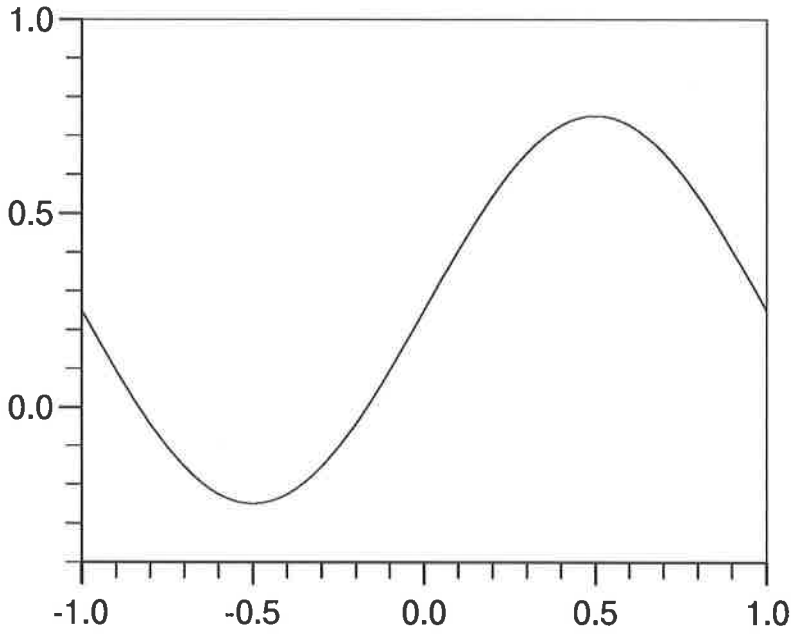

Figure 6: Initial data for Burgers' equation.

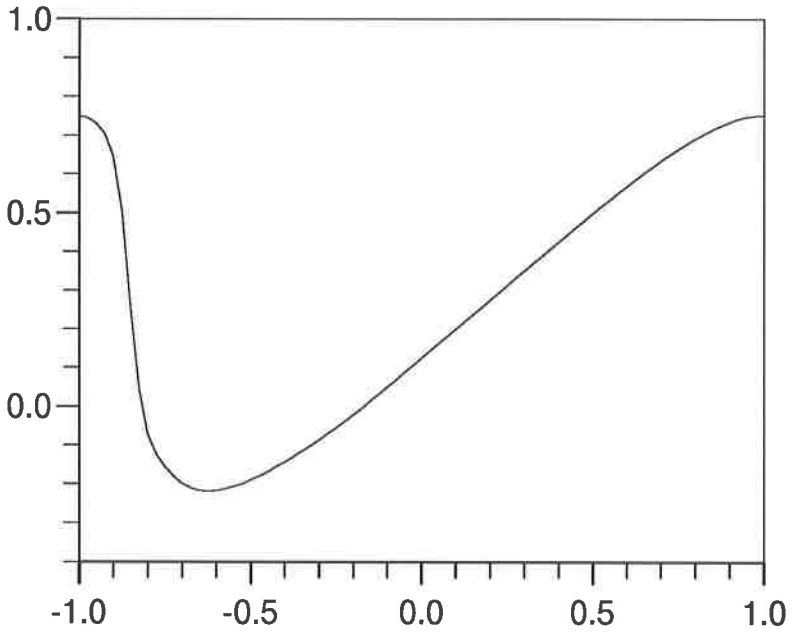

Figure 7: Solution at $T = 0.65$ using quasi-monotone and conservative semi-Lagrangian method.

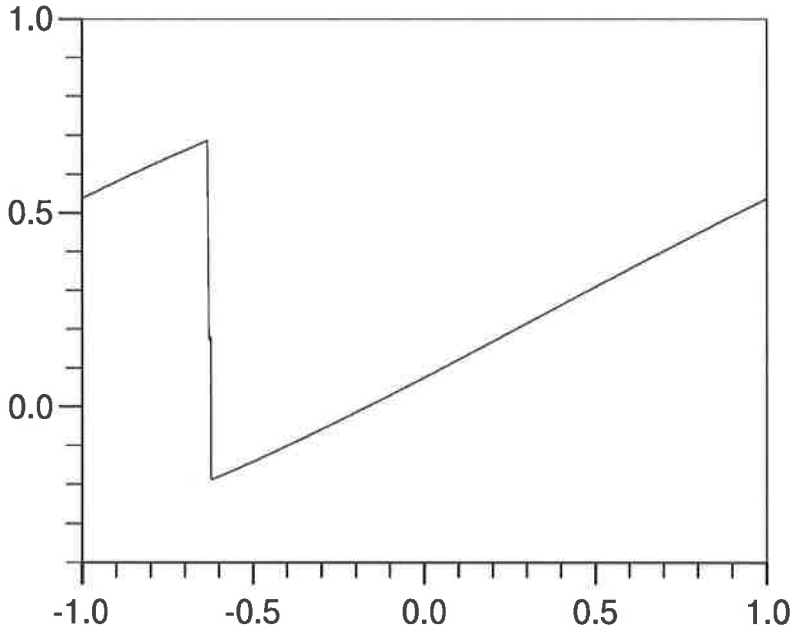

Figure 8: Control solution at $T = 1.5$.

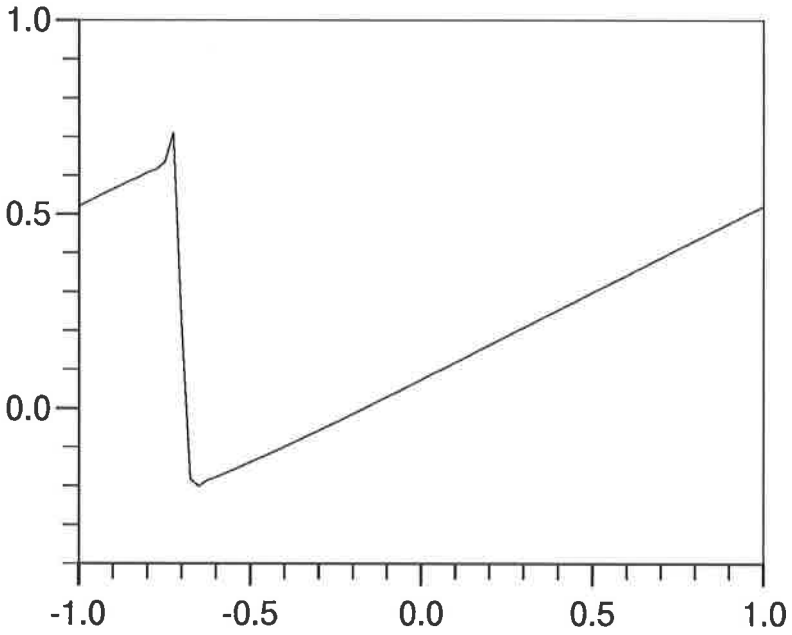

Figure 9: Solution at $T = 1.5$ with basic semi-Lagrangian method.

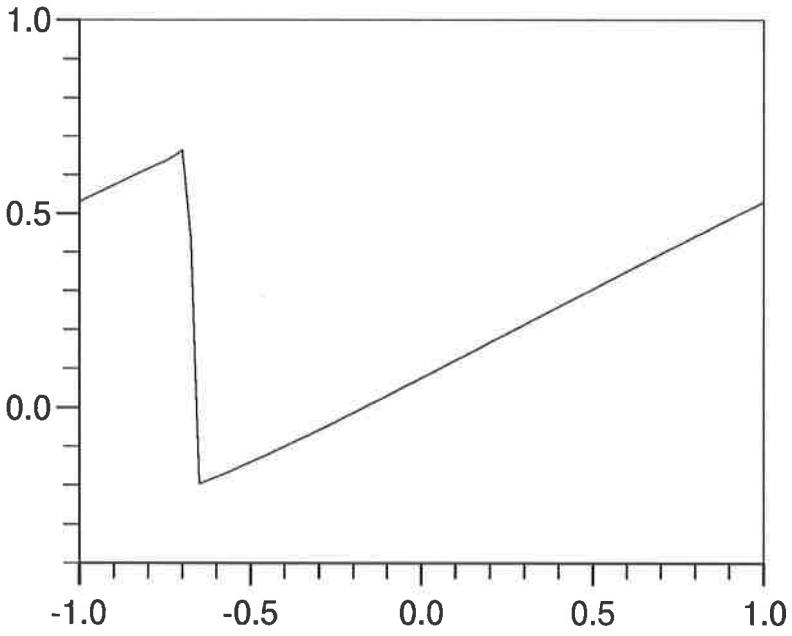

Figure 10: Solution at $T = 1.5$ with quasi-monotone semi-Lagrangian method.

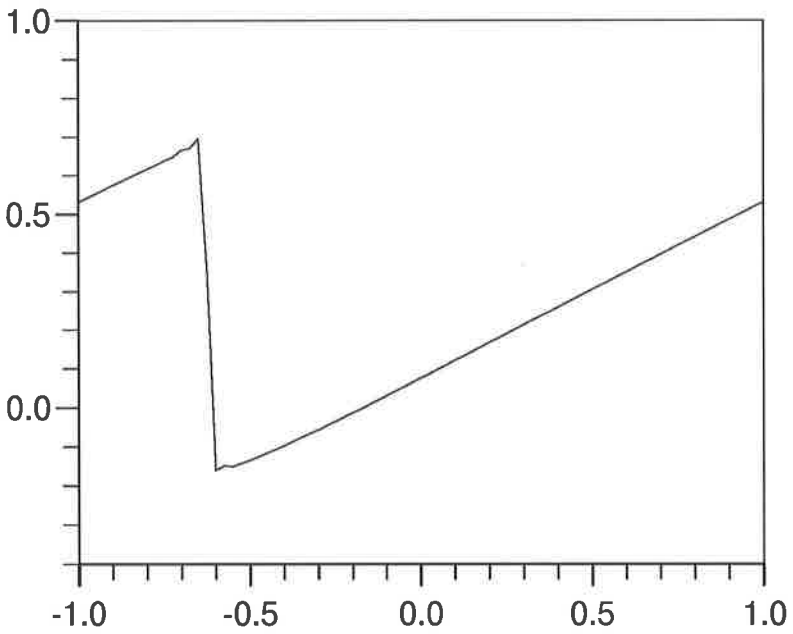

Figure 11: Solution at $T = 1.5$ with quasi-monotone and conservative semi-Lagrangian method.