THE UNIVERSITY OF READING

IMPROVING SECOND MOMENT CONSERVATION IN SEMI-LAGRANGIAN METHODS

by

A. Priestley and P. Garcia-Navarro

Numerical Analysis Report 11/95

The University of Reading
Department of Mathematics
P O Box 220,
Reading RG6 2AX
Berkshire, UK

DEPARTMENT OF MATHEMATICS

Improving Second Moment Conservation in Semi-Lagrangian Methods

A. Priestley and P. Garcia-Navarro[1]

Fluid Mechanics Group, C.P.S.,

University of Zaragoza,

50015 Zaragoza, Spain.

[1]Corresponding author

Abstract

In this note an algorithm for improving the conservation of the second moment when using a semi-lagrangian method is presented. This is achieved whilst maintaining the positivity and conservation of the first moment recovered in previously published algorithms.

1 Introduction

Semi-Lagrangian based methods have proved, in various guises, very successful in Computational Fluid Dynamics (CFD), mainly in connection with atmospheric flow prediction where they have gained widespread acceptance (see Staniforth and Côte, 1991, for an overview). They can be described as a technique using a fixed grid that essentially combines the method of characteristics with a suitable interpolating procedure.

It is known that the use of a linear interpolation between computational points sometimes produces an excessive amount of attenuation. The numerical damping can be reduced by using a more refined interpolation algorithm which will determine the spatial accuracy of the scheme. Over recent years various higher order interpolants have been proposed.

Consider the semi-Lagrangian solution to the scalar problem:

$$u_t + \mathbf{a}(\mathbf{x}, t).\nabla u = 0 \tag{1.1}$$

which describes the advection of $u(\mathbf{x}, t)$. The invariance of a scalar quantity

$$u(\mathbf{x}, t) = u(\mathbf{x}_0, t_0)$$

along a trajectory,

$$\mathbf{x}(t) = \mathbf{x}_0(t_0) + \int_{t_0}^{t} \mathbf{a}(\mathbf{x}, \tau) d\tau.$$

is common to a wide variety of fluid dynamics topics. The aim is to obtain a good approximation of the function $u(\mathbf{x}, t)$ at all the $\mathbf{x}_i$ points of a fixed discrete grid, assuming that u and $\mathbf{a}$ are known everywhere in the grid at an earlier time t_0. When

solving the advection equation, apart from achieving an accurate numerical approximation we may also desire, for many reasons, to mimic other properties of the analytic solution to (1.1). In particular, if $\nabla \mathbf{a} = 0$, we may wish our numerical solution to introduce no new extrema (positivity property) or we may desire that

$$\frac{d}{dt}\int_{\Omega} u d\Omega = 0 \tag{1.2}$$

where Ω is the domain we are interested in solving (1.1) over. Note that boundary conditions might mean the right hand side of (1.2) is replaced by some non-zero quantity. However, to keep the discussion simpler, and because the ultimate goal is to solve equation (1.1) on the sphere we ignore this complication.

Whilst (1.2) is, by far, the most common conservation law that numerical schemes try to reproduce, in fact we could replace the integrand with any power of u. In particular

$$\frac{d}{dt}\int_{\Omega} u^2 d\Omega = 0. \tag{1.3}$$

This could be referred to as conservation of energy if (1.2) was considered as conservation of mass but we will use the less ambiguous description that (1.3) represents conservation of the second moment, (1.2) representing conservation of the first moment.

Semi-Lagrangian schemes do not, inherently, give us any of the properties discussed above. Earlier work has addressed two of these properties. Positivity can be obtained using the implementation due to Rasch and Williamson (1990) and Williamson and Rasch (1989) but here the method proposed by Bermejo and Staniforth (1992) is used. This takes the solution at a point i, u_i, to be the weighted average of two semi-Lagrangian solutions

$$u_i = u_i^L + \alpha_i(u_i^H - u_i^L) \quad 0 \leq \alpha_i \leq 1. \tag{1.4}$$

u^L is restricted to be positive and is hence generally a low-order solution, whilst there is no restriction on u^H which is hence usually taken to be a high-order solution. To be compatible with previous work, a bilinear interpolation is used for u^L and a cubic-spline interpolation is used to provide u^H.

Denoting by u_i^{min} the minimum of the set of solution values used to interpolate at the foot of the trajectory passing through x_i, and similarly u_i^{max}, we can choose α_i

such that $u_i^{min} \leq u_i \leq u_i^{max}$. The maximum such value is denoted by α_i^{max} and the corresponding solution

$$u_i = u_i^L + \alpha_i^{max}(u_i^H - u_i^L)$$

is positive.

Priestley (1993) introduced a further modification to this approach to conserve the discrete version of the first moment. That is, denoting by S_i the area associated with node i, then it is attempted to enforce

$$\sum_i S_i u_i = constant. \tag{1.5}$$

Suboptimal values $0 \leq \alpha_i^* \leq \alpha_i^{max}$ are taken to satisfy (1.5). The complete algorithm for doing this is described in Priestley (1993). Essentially it calculates an average value of α, α_{av}, that, if applied to all the nodes still allowed to be altered, would give a solution u that satisfied (1.5). If it happens that for some nodes $\alpha_{av} > \alpha_i^{max}$, then for these nodes $\alpha_i^* = \alpha_i^{max}$ and they are removed from the allowable set. The value of α_{av} is then recalculated until the algorithm terminates.

Upon concluding the algorithm a number of nodes are left that have as their α^* the average, suboptimal value α_{av}. Over long time scales, energy or second moment conservation may become significant, in climate modelling for example, and so it is desirable to minimize any error in this quantity induced purely by the advection scheme. In this paper we present an algorithm that is applied to these remaining nodes to improve, and hopefully obtain, conservation of the second moment as well. In other words we want to enforce

$$\sum_i S_i u_i^2 = constant \tag{1.6}$$

whilst maintaining positivity and conservation of the first moment.

2 Second Moment Conservation

For this purpose the nodes that are still able to be changed are flagged by $iflag(i) = 0$ whereas those whose α has been already set are flagged by $iflag(i) = 1$. The

requirement expressed in (1.6) will be used in the form

$$\sum_{i:iflag(i)=0} S_i u_i^2 = constant - \sum_{i:iflag(i)=1} S_i u_i^2 = target = T$$

Introducing the notation $\beta_i = u_i^H - u_i^L$, note that in Priestley (1993) all the $\beta_i's$ with $iflag(i) = 0$ were of one sign and it was possible to assume $\beta_i > 0$ (by multiplying by -1 if necessary). In the following the cases must be treated separately, so, if the sign of the $\beta's$ was changed in the algorithm for conservation of the first moment, then it must be undone before proceeding.

Let $u_i^* = u_i^L + \alpha_i^*(u_i^H - u_i^L)$ be the current solution value, i.e. with the latest values of α_i^* and define

$$T^* = \sum_i S_i u_i^{*2}$$

We will present two algorithms next. First, the 'best' algorithm as this is the one most likely to find the solution sought. A second version is given later for the 'fair' algorithm. It is similar but may not recover as much second moment. It has nevertheless the advantage of being better suited for parallel computation.

2.1 'Best' algorithm

It can be described through the following sequence:

1) Define $T^{need} = T - T^*$

2) Order the points for which iflag is still zero such that

$$S_1 u_1 > S_2 u_2 > \ldots > S_n u_n \quad T^* < T$$

$$S_1 u_1 < S_2 u_2 < \ldots < S_n u_n \quad T^* > T$$

Note that, for simplicity, the points have just been labelled $1, ..., n$. The label just refers to the order in the list and will relate to an actual node via a pointer. See also remark 1.

3) Take the pair of points u_1 and u_n and proceed to add δ to u_1 and substract it from u_n to maintain first moment conservation. This results in an increase/decrease of second moment of $(S_1 + S_n)\delta^2 + 2\delta(S_1 u_1 - S_n u_n)$. Solving

$$(S_1 + S_n)\delta^2 + 2\delta(S_1 u_1 - S_n u_n) = T^{need} \tag{2.7}$$

the amount δ^{need}, mass that has to be transferred to achieve conservation of the second moment, is obtained. See remark 2.

4) Calculate δ_1, the most mass that can be added to u_1 to avoid violating positivity constraints:

$$\begin{cases} \text{if } \beta > 0 & \delta_1 = (\alpha_1^{max} - \alpha_1^*)\beta_1 \\ \text{else} & \delta_1 = -\alpha_1^*\beta_1 \end{cases} \tag{2.8}$$

Calculate δ_n, the most mass that can be substracted from u_n:

$$\begin{cases} \text{if } \beta > 0 & \delta_n = \alpha_n^*\beta_n \\ \text{else} & \delta_n = (\alpha_n^* - \alpha_n^{max})\beta_n \end{cases} \tag{2.9}$$

5) If both δ_1 and δ_n are greater than δ^{need} then the algorithm has succeeded. What remains is just to transfer the mass by altering the $\alpha's$, i.e.,

$$\alpha_1^* = \alpha_1^* + \frac{\delta^{need}}{\beta_1}$$

$$\alpha_n^* = \alpha_n^* + \frac{\delta^{need}}{\beta_n}$$

See remark 3.

6) Otherwise, choose $\delta = min(\delta_1, \delta_n)$ and calculate the appropiate $\alpha's$, updating u_1^*, u_n^* and T^*. Also set the flag to 1 for the node that limited the choice. And GOTO 2.

2.1.1 Remarks

1. For efficient ordering algorithms, see W.H. Press et al.(1992) for example. More importantly, after the first pass we only need to apply an insertion algorithm. This is particularly trivial when $T^* < T$.

2. When T^{need} is positive (2.7) always has a (real) solution. When T^{need} is negative it may not. In this case we take

$$\delta^{need} = \frac{S_n u_n - S_1 u_1}{S_1 + S_n}$$

This gives a minimum of second moment for this pair of points. In this case then it is necessary to loop through the algorithm again regardless of step 5.

3. Clearly division by zero must be avoided here. On the other hand, when $\beta = 0$ altering the value of α^* does not change the solution u^*. In practice, though, this means ignoring the nodes i where $|\beta_i| < \epsilon$ and ϵ is some tolerance, by setting their flags to 1. The algorithm itself only requires an ϵ distinguishable from zero by the computer. However, enormous savings can be made by taking ϵ as large as possible. Within the range $10^{-5} - 10^{-3}$ the extra cost of the algorithm cannot be discerned. The results obtained are, to all purposes, identical to those obtained with $\epsilon = 10^{-9}$. However, the cost of the recovery of the second moment can become the most expensive part of the whole solution in this latter case.

2.2 The 'fair' algorithm

The only real difference to the 'best' algorithm is that all the points in the ordering are put into pairs $(u_1, u_n), (u_2, u_{n-1}), \ldots$. If there are np such pairs then redefine

$$T^{need} = \frac{T - T^*}{np}$$

for step 3. Steps 3 to 6 are then applied to each pair and loop through the algorithm again if any of the pairs failed in Step 5.

This algorithm is closer, in spirit, to the one given in Priestley (1993) for the conservatin of the first moment. Since there is now a significant amount of calculation that can be done independently of what is happening at other points it is better suited to parallel computations. Another alternative, if there are k processors say, would be to split the list in Step 2 into k parts, taking the bottom $n/2k$ points with the top $n/2k$ points, etc., and sending each of these lists to a different processor with a target T^{need}/k and using the 'best' algorithm on each separate list.

3 Results

Two test problms are considered. The first problem is that of the slotted cylinder of Bermejo and Staniforth (1992) advected by a given velocity field, $\mathbf{a} = \omega(-\mathbf{y}, \mathbf{x})$ which causes the cylinder to rotate with constant angular velocity ω. The second is the same problem with the slotted cyliner replaced by a sin^2 cone of the same radius and maximum height.

Following Bermejo and Staniforth (1992) the errors for this problem are split into the dissipation error (DISSER) and the dispersion error (DISPER). The first and second moments are estimated in the same way as in Priestley (1993). The maximum and minimum values are also given.

In table 1 the results to the slotted cylinder problem using the cubic spline semi-Lagrangian method with positivity and first and second moment conservation algorithms applied are given. This is directly comparable with table 3 in Priestley (1993). Very close observation of the results shows that the conservation of the second moment has improved by about 3/10%. There are a number of reasons for this. The algorithm may not be able to recover much second moment if a) there are too few points passed to the algorithm, b) $\beta_i \approx 0$ for many nodes, c)$\alpha_{av} \approx \alpha_i^{max}$ for many nodes. In actual fact, both b) and c) seemed to be true. Matters are made worse by the fact that so much of the second moment is lost in the first time steps. Whilst it is probably unsafe to infer too much from this result, it should also be noted that the error has also improved by a similar amount.

Purely in order to create a situation where there are some nodes for which β_i is not so close to zero and α_{av} not so close to α_i^{max} we use a variation on the scheme. Essentially, instead of u_i^L we use u_i^{min} and then take

$$\alpha_i^{max} = \frac{u_i^{max} - u_i^{min}}{u_i^H - u_i^{min}}$$

In effect this allows us to use, in terms of equation (1.4), $\alpha's$ less than zero or greater than one. Looking at the errors in tables 2 and 3 will convince the reader that we are not recommending this approach. We stress it is solely used to show what the algorithm for the conservation of the second moment can do when there is enough room for it to manoeuvre in.

Table 2 shows the result obtained for the slotted cylinder problem. It can be seen that the algorithm has vastly improved the conservation of the second moment. For the sin^2 cone, conservation of the second moment is actually achieved as table 3 makes plain. In table 4, the results for this problem with the method from Priestley (1993) are given for comparison.

4 Conclusions

We have given an algorithm for conserving the second moment of a semi-Lagrangian generated numerical solution whilst maintaining positivity and conservation of the first moment. It has been shown that, provided there is enough 'freedom' in the solution, conservation, or vastly improved conservation, of the second moment can be achieved by the algorithm. The remaining question is, then, how well it will work for more realistic problems? This is the basis for future work.

References

[1] Bermejo, R. and Staniforth, A., "The Conversion of Semi-Lagrangian Advection Schemes to Quasi-Monotone Schemes." Monthly Weather Review, **120**, pp. 2622-2632, 1992.

[2] Press, W. H., Teukolsky, S.A., Vetterling W.T. and Flannery B.P.,"Numerical Recipes in Fortran. The Art of Scientific Computing." 2nd Ed. Cambridge Univerity Press, 1992.

[3] Priestley, A., "A Quasi-Conservative Version of the Semi-Lagrangian Advection Scheme." Monthly Weather Review, **121**, pp. 621-629, 1993.

[4] Rasch, P.J. and Williamson, D.L., "On Shape-Preserving Interpolation and Semi-Lagrangian Transport." SIAM J. Sci. Stat. Comp., **11**, pp.656-687.

[5] Staniforth, A. and Côte, J., "Semi-Lagrangian Integration Schemes for Atmospheric Models- A Review." Monthly Weather Review, **119**, pp. 2206-2223, 1991.

[6] Williamson, D.L. and Rasch, P.J., 1989: "Two-dimensional semi-Lagrangian Transport with Shape-Preserving Interpolation." Monthly Weather Review, **117**, pp. 102-129.

List of Tables

$N^o\Delta t$	RFM	RSM	MAX	MIN	DISSER	DISPER
96	1.00000	0.869577	3.99988	-6.28837$\times 10^{-17}$	4.3882$\times 10^{-3}$	4.4309$\times 10^{-2}$
192	1.00000	0.850333	3.99847	1.7448$\times 10^{-19}$	5.8454$\times 10^{-3}$	5.1511$\times 10^{-2}$
288	1.00000	0.83665	3.99403	3.3377$\times 10^{-19}$	7.0207$\times 10^{-3}$	5.6284$\times 10^{-2}$
384	1.00000	0.82585	3.99071	4.4215$\times 10^{-19}$	8.0332$\times 10^{-3}$	5.9971$\times 10^{-2}$
480	1.00000	0.816827	3.98983	4.9297$\times 10^{-19}$	8.9371$\times 10^{-3}$	6.3064$\times 10^{-2}$
576	1.00000	0.809004	3.98840	5.5226$\times 10^{-19}$	9.7644$\times 10^{-3}$	6.5818$\times 10^{-2}$

Table i: Errors for slotted cylinder problem using 'best' algorithm for second moment conservation with low order scheme .

$N^o\Delta t$	RFM	RSM	MAX	MIN	DISSER	DISPER
96	1.00000	1.00000	4.00000	0.0000	2.7981$\times 10^{-22}$	0.761600
192	1.00000	1.00000	4.00000	0.0000	9.7257$\times 10^{-17}$	1.14240
288	1.00000	0.998870	4.00000	0.0000	3.0618$\times 10^{-7}$	1.34923
384	1.00000	0.997355	4.00000	0.0000	1.6805$\times 10^{-6}$	1.28435
480	1.00000	0.995317	4.00000	0.0000	5.2714$\times 10^{-6}$	1.20990
576	1.00000	0.991611	4.00000	0.0000	1.6953$\times 10^{-5}$	1.31311

Table ii: Errors for slotted cylinder problem using 'best' algorithm for second moment conservation without low order scheme.

$N^o\Delta t$	RFM	RSM	MAX	MIN	DISSER	DISPER
96	1.00000	1.00000	3.84885	0.0000	1.8654×10^{-27}	3.8321×10^{-3}
192	1.00000	1.00000	3.60297	0.0000	1.2878×10^{-13}	6.5552×10^{-3}
288	1.00000	1.00000	3.53551	0.0000	5.4579×10^{-24}	7.8643×10^{-3}
384	1.00000	1.00000	3.53194	0.0000	6.5460×10^{-13}	9.7556×10^{-3}
480	1.00000	1.00000	3.53158	0.0000	5.1410×10^{-14}	1.1081×10^{-3}
576	1.00000	1.00000	3.49792	0.0000	1.4544×10^{-14}	1.3529×10^{-3}

Table iii: Errors for sin^2 cone problem using 'fair' algorithm for second moment conservation without low order scheme.

$N^o\Delta t$	RFM	RSM	MAX	MIN	DISSER	DISPER
96	1.00000	0.996275	3.92855	-2.3174×10^{-18}	6.8844×10^{-7}	2.12×10^{-6}
192	1.00000	0.993942	3.90136	5.2935×10^{-23}	1.8234×10^{-6}	4.9562×10^{-6}
288	1.00000	0.991910	3.88197	5.0292×10^{-22}	3.2552×10^{-6}	8.0885×10^{-6}
384	1.00000	0.99005	3.86616	1.0153×10^{-21}	4.9271×10^{-6}	1.1447×10^{-5}
480	1.00000	0.988301	3.85256	2.6200×10^{-21}	6.8203×10^{-6}	1.4989×10^{-5}
576	1.00000	0.986625	3.84046	4.4204×10^{-21}	8.9228×10^{-6}	1.8700×10^{-5}

Table iv: Errors for sin^2 cone problem using method with just the first moment conservation.