Q. J. R. Meteorol. Soc. (1999), **125**, pp. 197–212

Monotonic cascade interpolation for semi-Lagrangian advection

By RAMACHANDRAN NAIR, JEAN CÔTÉ and ANDREW STANIFORTH*
Meteorological Research Branch, Environment Canada

(Received 2 June 1997; revised 3 December 1997)

SUMMARY

A simpler, more efficient, and more robust form of cascade interpolation is proposed. It significantly reduces the considerable overhead required to determine the mesh points of the intermediate mesh, without degrading accuracy. A simple improvement of an existing monotonic filter is also proposed to address a deficiency and enhance robustness. These ideas are demonstrated using two standard test-problems.

KEYWORDS: Cascade interpolation Monotonic filter Semi-Langrangian advection

1. INTRODUCTION

At every time-step of a semi-Lagrangian advection-scheme employing backward (forward) trajectories, the advected fields need to be evaluated at the upstream (downstream) points via interpolation of the known surrounding grid-point values. A typical tensor-product interpolation in three dimensions (3-D), based on a one-dimensional (1-D) interpolator, requires $O(p^3)$ operations per grid point per field, where p is the formal order of accuracy of the interpolator ($p = 4$ for cubics). A high-order interpolation formally improves the accuracy of the advection scheme but unfortunately engenders additional computational expense. Purser and Leslie (1991), hereinafter referred to as PL91, proposed an alternative scheme termed 'cascade' interpolation. Their technique employs a sequence of 1-D interpolations for the multi-dimensional interpolation-problem associated with semi-Lagrangian advection, thereby reducing the cost of using high-order interpolation schemes in 2-D and 3-D to $O(p)$ operations per grid point per field—a significant saving.

The first step in applying cascade interpolation is to generate an intermediate mesh defined by the intersection of the regular Eulerian and the curvilinear Lagrangian meshes. Values are then transferred from the Eulerian mesh to the intermediate mesh by 1-D interpolation. The final step is to interpolate values defined on the intermediate mesh to the target points of the Lagrangian mesh, again using 1-D interpolation. Cascade interpolation can also be used for semi-Lagrangian advection with forward trajectories (Purser and Leslie 1994), for which conventional interpolation techniques are difficult and expensive, by simply reversing the order of the sequence of 1-D interpolations. Furthermore, Leslie and Purser (1995) and Rančić (1995) have shown that 1-D mass-conserving monotonic remapping algorithms can also be incorporated within a cascade interpolation framework. Sun and Yeh (1997), hereinafter referred to as SY97, have also proposed a semi-Lagrangian advection-scheme using forward trajectories. The interpolation procedure used, ingeniously termed the 'internet' procedure, is a variant of cascade interpolation, and it is combined with the monotonic filter of Sun *et al.* (1996), hereinafter referred to as SYS96.

As noted by PL91, generation of the intermediate mesh represents a significant overhead of cascade interpolation. The purposes of the present work are to propose a simpler and more efficient form of cascade interpolation that significantly reduces this overhead

* Corresponding author, present address: NWP Division, Meteorological Office, London Road, Bracknell, Berkshire, RG12 2SZ, UK.

without degrading accuracy, to propose a simple and robust improvement of the SYS96 monotonic filter, and to clarify some points raised in SY97.

2. Cascade interpolation

For the traditional semi-Lagrangian advection-problem employing backward trajectories, function values defined at the points of a regular uniform Eulerian mesh need to be transferred by polynomial interpolation to the corresponding points of the Lagrangian mesh, i.e. the curvilinear mesh defined by the upstream positions. For semi-Lagrangian advection employing forward trajectories, the data transfer is in the opposite sense, viz. from the Lagrangian mesh (defined by the arrival points) to the Eulerian ones (defined by the departure points). In both cases, a cascade scheme performs this grid-to-grid transfer through an intermediate mesh by means of 1-D interpolation. In what follows, the algorithm is presented in terms of backward trajectories, but the development is similar for forward trajectories.

Consider the 2-D advection of a passive scalar ψ, on a rectangular Cartesian domain. Thus

$$\frac{\mathrm{d}\psi}{\mathrm{d}t} \equiv \frac{\partial\psi}{\partial t} + u\frac{\partial\psi}{\partial x} + v\frac{\partial\psi}{\partial y} = 0, \tag{1}$$

where u and v are the components of velocity in the x and y directions respectively. Let (x_i, y_j), $i = 1, \ldots, m$; $j = 1, \ldots, n$ define the Eulerian meshpoints of the computational domain $\mathcal{D}$ (see Fig. 1(a) for a schematic illustration). Assume that for every mesh point (x_i, y_j) in $\mathcal{D}$ at time $t + \Delta t$ there exists a unique Lagrangian point (x_{ij}, y_{ij}) at time t in $\mathcal{D}$, and that its position (Fig. 1(c)) is determined by the upstream trajectory of the flow. That is, a fluid particle that arrives at the point (x_i, y_j) at time $t + \Delta t$, departed from the point (x_{ij}, y_{ij}) at time t. The arrival point (x_i, y_j) and the departure point (x_{ij}, y_{ij}) satisfy

$$(x_i, y_j) - (x_{ij}, y_{ij}) = (\alpha_{ij}, \beta_{ij}), \tag{2}$$

where $(\alpha_{ij}, \beta_{ij})$ is the vector displacement determined by the numerical approximation (Robert 1981; Staniforth and Côté 1991) of

$$\left(\frac{\mathrm{d}x}{\mathrm{d}t}, \frac{\mathrm{d}y}{\mathrm{d}t}\right) = (u, v). \tag{3}$$

Let (X, Y) denote the curvilinear Lagrangian coordinate-system corresponding to the Eulerian (x, y) coordinate-system (Fig. 1(a)). A y-line of the regular Eulerian mesh is a straight line defined by $x = x_i$, whereas the associated mapped Lagrangian Y-curve (or simply Y-curve) is a curve corresponding to the mapped y-line that joins the upstream points $(x_{i1}, y_{i,1})$, (x_{i2}, y_{i2}), $\ldots$, (x_{in}, y_{in}).

A key component of the cascade-interpolation technique is the generation of a well-defined intermediate mesh. An intermediate mesh can be constructed as the intersection of Y-curves with x-lines or as the intersection of X-curves with y-lines. Ideas are arbitrarily fixed here on the former: i.e., an intermediate mesh is constructed such that the x-lines defined by $y = y_j$, $j = 1, \ldots, n$ intersect the Y-curves Y_i, $i = 1, \ldots, m$. This is schematically illustrated in Fig. 1(c) for a specific Y-curve Y_i, where the points of intersection are denoted by '$\times$'.

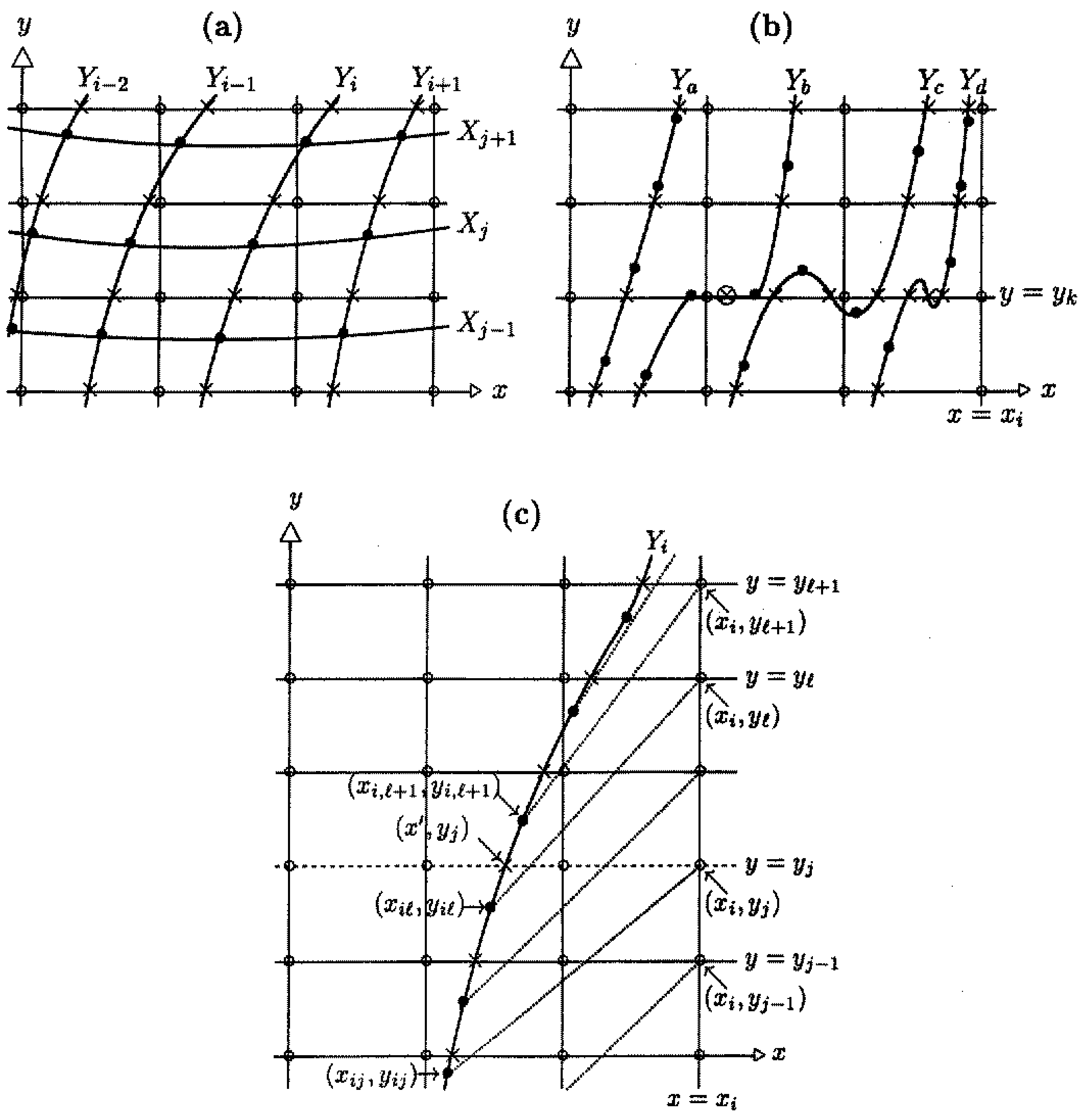


Figure 1. Schematics for cascade interpolation. Eulerian mesh-points denoted by '∘', Lagrangian mesh-points by '•', and intermediate mesh-intersections by '×': (a) Lagrangian (thick lines) and Eulerian (thin lines) meshes; (b) various intersections of Lagrangian Y-curves with the Eulerian x-line defined by $y = y_k$ (symbol '⊗' represents the mid-point of the line segment of Y_b for which $y = y_k$); (c) a specific Lagrangian Y-curve Y_i that intersects the standard Eulerian x-line defined by $y = y_j$ (dashed line) at an intermediate mesh-point (x', y_j) (dotted lines denote backward trajectories).

(*a*) *Determination of mesh intersection points via linear interpolation*

The simplest way of determining an intermediate mesh is by using linear interpolation to determine the points of intersection of each Y-curve with x-lines. This corresponds to an important simplification of both the PL91 and the SY97 algorithms, and has the virtue of being significantly more economical. Theoretically, Eq. (18) of McDonald's (1987) analysis indicates that when using cubic interpolation to evaluate advected quantities, it is sufficient to use linear interpolation when computing the trajectories that define the Lagrangian mesh, and this has been confirmed experimentally (e.g. Temperton and Staniforth 1987; Bates *et al.* 1990). Although it is possible to use interpolation of higherthan-linear order in this context, this is of little benefit since it does not reduce the order of the truncation errors. Also, commonly used algorithms usually assume linear spatio-temporal trajectories. These observations strongly suggest that linear interpolation should, for many, if not most, problems, be sufficient to determine the points of intersection of the Eulerian and Lagrangian meshes, and that little is likely to be gained by paying the price of cubic interpolation for this component of cascade interpolation.

Consider (see Fig. 1(c)) a specific Y-curve Y_i and recall that it joins the points $(x_{i1}, y_{i1}), (x_{i2}, y_{i2}), \ldots, (x_{in}, y_{in})$. If it intersects the x-line defined by $y = y_j$, then it will, in general, do so such that $y_{i\ell} \leqslant y_j \leqslant y_{i,\ell+1}$ for some value of the integer index ℓ, where $1 \leqslant \ell \leqslant n-1$. By passing a straight line between the two points $(x_{i\ell}, y_{i\ell})$ and $(x_{i,\ell+1}, y_{i,\ell+1})$, its point of intersection with $y = y_j$ is simply (x', y_j), where

$$x' = x_{i\ell} + \frac{y_j - y_{i\ell}}{y_{i,\ell+1} - y_{i\ell}}(x_{i,\ell+1} - x_{i\ell}). \tag{4}$$

The set of all (x', y_j) for all x-lines $(j = 1, \ldots, n)$ and for all Y-curves Y_i $(i = 1, \ldots, m)$, constitutes the intersection points of the sought intermediate mesh.

If $y_{i1}, y_{i2}, \ldots, y_{in}$ forms a monotonically increasing sequence (this will almost always be the case, and it can always be arranged to be so by a suitable reduction of time-step length), then a specific Y-curve Y_i will intersect each x-line exactly once. In this case, i and j indices could be added to x'. The intermediate mesh then exactly consists of the $m \times n$ set of points (x'_{ij}, y_j), where $i = 1, \ldots, m$ and $j = 1, \ldots, n$.

Multiple intersections of a specific Y-curve with a specific x-line are, however, possible (see the Y-curves Y_b and Y_c in Fig. 1(b) for an illustration of this). The intermediate mesh then has more points than the $m \times n$ points of both the Eulerian and Lagrangian meshes. For the Y-curve Y_c, the algorithm described above uniquely determines its two points of intersections with the x-line defined $y = y_j$. However, for the Y-curve Y_b, there are an infinite number of intersections, viz. all points that lie on the straight-line segment joining the points $(x_{i\ell}, y_k)$ and $(x_{i,\ell+1}, y_k)$. It is then convenient simply to take the midpoint value of this segment, i.e.

$$x' = \frac{x_{i\ell} + x_{i,\ell+1}}{2} \quad \text{if } y_{i,\ell+1} \equiv y_{i\ell}. \tag{5}$$

Note that this corresponds to, and addresses, the singular case of Eq. (4).

(*b*) *Cascading the interpolation for schemes based on backward trajectories*

In a 2-D rectangular domain, the cascade interpolation procedure for semi-Lagrangian advection employing backward trajectories can be summarized as follows.

(i) For $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$, obtain, in the usual way, the upstream departure points (x_{ij}, y_{ij}) that define the Lagrangian mesh.

(ii) Determine the location of the points that define the intermediate mesh (i.e. the intersection points of Y-curves with x-lines) using the linear interpolation algorithm (i.e. Eqs. (4) and (5)) of subsection 2(a).

(iii) For $j = 1, 2, \ldots, n$: interpolate, using 1-D polynomial interpolation along the x-lines $y = y_j$, all advected fields from the mesh points of the Eulerian mesh to the mesh points of the intermediate mesh.

(iv) For $j = 1, 2, \ldots, n$: interpolate, using 1-D polynomial interpolation along the Y-curves Y_i, all advected fields from the mesh points of the intermediate mesh to the target mesh points of the Lagrangian mesh.

The interpolation in step (iv) is performed using the distance s along the Y-curve as the independent variable. Recall that the Y_i-curve is defined by joining (Fig. 1(c)) the points $(x_{i1}, y_{i1}), (x_{i2}, y_{i2}), \ldots, (x_{in}, y_{in})$ by straight lines between adjacent points (i.e. 'piecewise'-linearly). Dropping the i subscript for convenience, the distances s_ℓ of the points (x_ℓ, y_ℓ) along the curve are recursively defined by

$$\begin{aligned} s_1 &= 0, \\ s_{\ell+1} &= s_\ell + \sqrt{(x_{\ell+1} - x_\ell)^2 + (y_{\ell+1} - y_\ell)^2} \quad \text{for } \ell = 1, 2, \ldots, n-1. \end{aligned} \tag{6}$$

A point (x', y_j) of the intermediate mesh (cf. Fig. 1(c)) that lies on the straight line joining (x_ℓ, y_ℓ) to $(x_{\ell+1}, y_{\ell+1})$, where x' is given by Eq. (4), is therefore located at

$$s' = s_\ell + (s_{\ell+1} - s_\ell)\frac{(y_j - y_\ell)}{(y_{\ell+1} - y_\ell)}, \tag{7}$$

and the second term on the right-hand side of Eq. (7) is guaranteed positive.

If the time-step is chosen sufficiently small to eliminate multiple intersections of specific Y-curves with specific x-lines (see Fig. 1(b)), then y may be used as the independent variable instead of s, since it remains monotonic. However, these two choices of independent variable do not in general lead to the same result (e.g. when interpolating ψ cubically) because of different spacing of mesh points. Our preference is to use s rather than y, since the algorithm is then more robust and the computational benefit of using y is slight anyway.

Cubic interpolation, with a monotonicity constraint (see section 3), is adopted in the present work for the 1-D polynomial interpolators of steps (iii) and (iv), since cubic interpolation of advected fields has been found to be a good compromise between accuracy and efficiency for semi-Lagrangian schemes (Staniforth and Côté 1991). Higher-order (e.g. quintic) interpolators also merit serious consideration (PL91) provided that the advected fields have sufficient continuity, since this reduces the inherent damping caused by interpolation, and the incremental cost of increasing the polynomial order for cascade interpolation is much less than that for conventional Cartesian-product interpolation. Linear and quadratic interpolation are, however, not generally recommended since they can lead to an unacceptable amount of damping (Staniforth and Pudykiewicz 1985).

(*c*) *Cascading the interpolation for schemes based on forward trajectories*

For a semi-Lagrangian scheme employing forward trajectories, the interpolation procedure is very similar to that described above for schemes based on the use of backward trajectories. The upstream points (x_{ij}, y_{ij}) of step (i) are now to be interpreted as being the downstream arrival points associated with the departure points (x_i, y_j) and their locations are determined in an analogous manner. The step (ii) construction of the intermediate mesh remains unchanged, and steps (iii) and (iv) are interchanged; i.e., the fields are interpolated from the Lagrangian mesh to the intermediate mesh to the target Eulerian mesh rather than the converse.

(*d*) *Comparative computational cost*

The computational cost of the 'economic' and 'complete' interpolation algorithms for 2-D passive advection is estimated by SY97 to be, respectively, $6K^2 + 90$ and $6K^2 + 150$ arithmetic operations per grid point, where $K - 1$ is the degree of the polynomials employed. The constant component of these estimates is the cost of determining mesh intersection points. By using the linear algorithm proposed here instead of the SY97 cubic one, this component of the cost is reduced from 90 and 150 operations per grid point to 6 and 12 respectively (see Eq. (4)). For $K = 4$ (i.e. cubic interpolation of the advected field), the total costs of the SY97 economic and complete algorithms are therefore reduced from 186 and 254 to 102 and 116 arithmetic operations per grid point respectively. Thus, using linear instead of cubic polynomials to determine mesh intersection points approximately halves the total number of arithmetic operations of the SY97 algorithms.

3. An improved monotonic filter

In the work of SY97, a modified version (see SYS96) of the QMSL (quasi-monotone semi-Lagrangian) filter of Bermejo and Staniforth (1992) is applied immediately after each 1-D interpolation of the interpolation cascade. The SYS96 alogorithm identifies the existence of a local extremum by examining slope variations of the function in the three-mesh-length interval surrounding the target evaluation-point: the QMSL filter is then turned off whenever an extremum is identified in order to capture it. This modification is designed to address a shortcoming of the QMSL filter, viz. that under certain circumstances it clips some local extrema that one would ideally wish to retain. However, the SYS96 filter is only partially successful, since it trades off an improvement in one circumstance against a deterioration in another. This is illustrated schematically in Fig. 2 for a 1-D interpolation function in a region with a strong Gibbs (i.e. overshoot/undershoot) phenomenon.

Assume that a function is sampled at the points represented by solid circles in Fig. 2(a) and that the curve joining these circles is a high-order interpolating function. Assume also that this interpolating function is to be evaluated at the upstream points $x_i - \alpha_i$. The resulting evaluation is denoted by the hollow stars, and the Gibbs phenomenon is obvious.

The QMSL filter is defined as follows:

(a) identify the local minimum and maximum surrounding the target point $x_i - \alpha_i$ as

$$f^- = \min(f_\ell, f_{\ell+1}), \tag{8}$$

$$f^+ = \max(f_\ell, f_{\ell+1}), \tag{9}$$

where $x_i - \alpha_i$ is in the interval $[x_\ell, x_{\ell+1}]$;

(b) reset $f(x_i - \alpha_i)$ such that

$$f(x_i - \alpha_i) = f^- \quad \text{if } f(x_i - \alpha_i) < f^-, \tag{10}$$

$$f(x_i - \alpha_i) = f^+ \quad \text{if } f(x_i - \alpha_i) > f^+. \tag{11}$$

The SYS96 filter also applies the QMSL filter, but turns it off whenever an extremum is identified by the monotonicity condition

$$(f_\ell - f_{\ell-1})(f_{\ell+2} - f_{\ell+1}) < 0, \tag{12}$$

provided $f_{\min} \leqslant f(x_i - \alpha_i) \leqslant f_{\max}$, where $f_{\min}$ and $f_{\max}$ are, respectively, the global minimum and maximum of f.

The results after applying the QMSL and SYS96 filters to the data of Fig. 2(a) are shown in Fig. 2(b). The QMSL filter correctly enforces monotonicity (open-circle points) and realistically controls the Gibbs behaviour. However, although the SYS96 filter mostly controls this behaviour, it fails to do so at the point $x_i - \alpha_i$, resulting in a violation of monotonicity, an isolated small-scale perturbation being superimposed on the expected solution, and the introduction of a spurious mass-source.

All, however, is not lost. By making a further small modification, it is possible to get the best of two worlds, viz. the robustness of the QMSL algorithm, and the capture of local subgrid-scale extrema by the SYS96 one. The reason why the SYS96 algorithm breaks down in a Gibbs-phenomenon situation is that in identifying local extrema, the local behaviour of the function is considered only in the *three*-mesh-length-wide interval containing the extremum, which is insufficient to exclude the two-mesh-length noise signature of the Gibbs phenomenon. By extending the width to *five* mesh-lengths (or more if desired), and by insisting that the function have only a single extremum in this larger interval (see the schematic of Fig. 2(c)), the problem is avoided. The revised monotonic procedure can be expressed algorithmically as follows.

(i) If all the following four inequalities hold, do not apply the QMSL filter:

$$f_{\min} \leqslant f(x_i - \alpha_i) \leqslant f_{\max}, \tag{13}$$

$$(f_{\ell-1} - f_{\ell-2})(f_\ell - f_{\ell-1}) > 0, \tag{14}$$

$$(f_\ell - f_{\ell-1})(f_{\ell+2} - f_{\ell+1}) < 0, \tag{15}$$

$$(f_{\ell+2} - f_{\ell+1})(f_{\ell+3} - f_{\ell+2}) > 0, \tag{16}$$

where the target meshpoint $x_i - \alpha_i$ is in the interval $[x_\ell, x_{\ell+1}]$.

(ii) If one or more of these inequalities is violated, then apply the QMSL filter.

Inequality (15) (see Fig. 2(c)) identifies $f(x_i - \alpha_i)$ as an extremum in the interval $[x_\ell, x_{\ell+1}]$, whereas inequalities (14) and (16) ensure monotonicity in the two-mesh-length-wide intervals on either side of this interval. Note that the new filter-formulation reduces to the SYS96 one by dropping conditions (14) and (16), and then further reduces to the Bermejo and Staniforth (1992) (QMSL) one by dropping conditions (13)–(16) altogether.

4. Discussion

For the SY97 internet cascade algorithm, the points of the intermediate mesh are determined by parametrically describing the coordinates of the Y_i-curves as piecewise cubics of the form

$$x(s) = a_3 s^3 + a_2 s^2 + a_1 s + a_0, \tag{17}$$

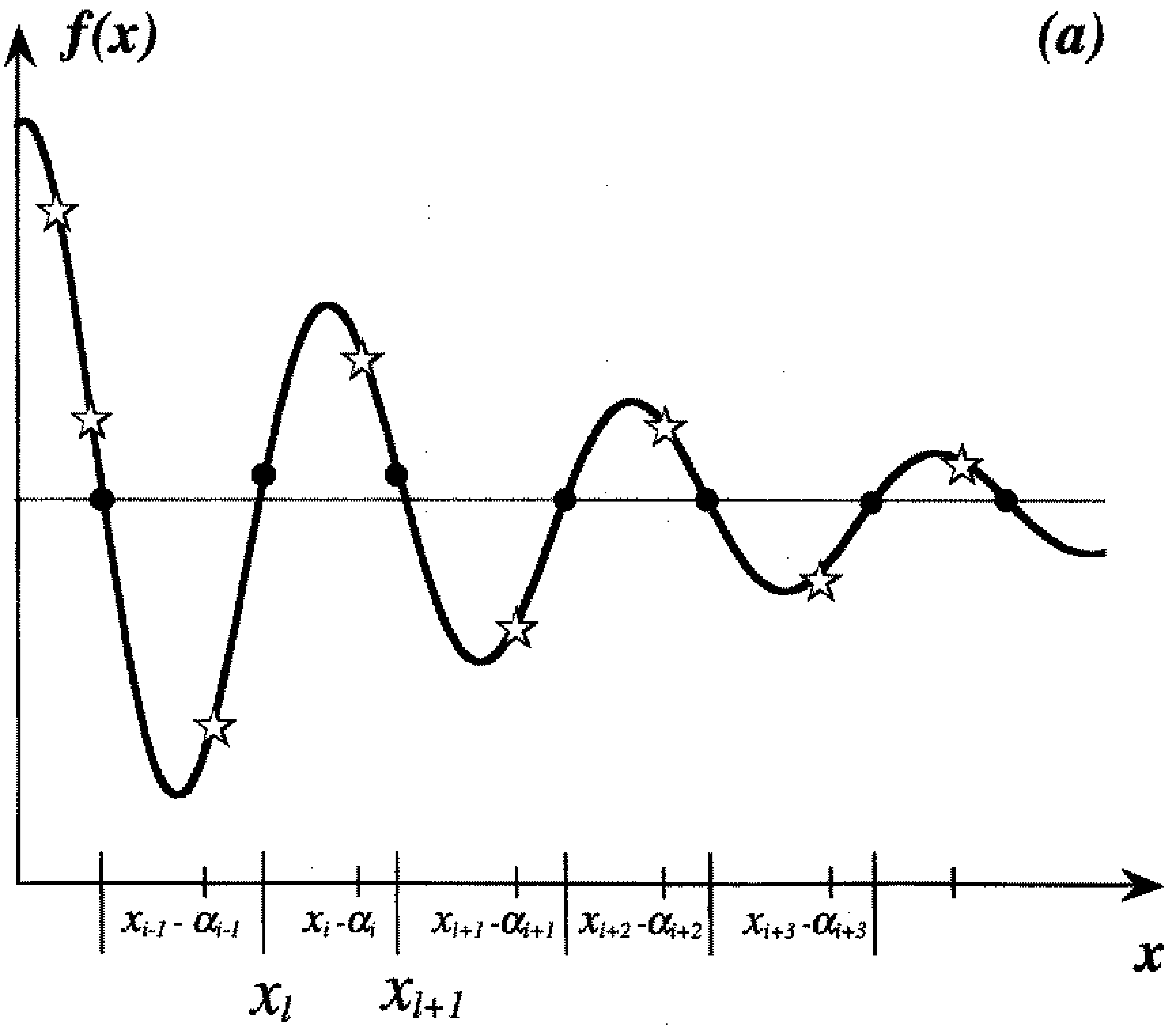
$$y(s) = b_3 s^3 + b_2 s^2 + b_1 s + b_0, \tag{18}$$

where s is a parameter. When the high-order interpolator is chosen to be cubic for the PL91 algorithm, the points of the intermediate mesh are determined by describing the coordinates of the Y_i-curves as piecewise cubics of the form

$$y(x) = c_3 x^3 + c_2 x^2 + c_1 x + c_0. \tag{19}$$

The coefficients in Eqs. (17)–(19) are determined for both of these algorithms by fitting to the known points $(x_{i1}, y_{i1}), (x_{i2}, y_{i2}), \ldots, (x_{in}, y_{in})$ that lie on the Y_i-curve. Note that if a_2, a_3, b_2, b_3, and c_2, c_3, were each to be set to zero, the SY97 and PL91 algorithms for determining the points of the intermediate mesh would reduce to the linear one described in subsection 2(a).

There is, however, a potential weakness (other than computational cost) in using cubic representations of the Y_i-curves. Implicitly, the validity of cascade interpolation depends upon the smoothness of the velocity field (PL91). Numerically, high-order interpolators are generally of no benefit, and often detrimental, when their continuity assumptions are



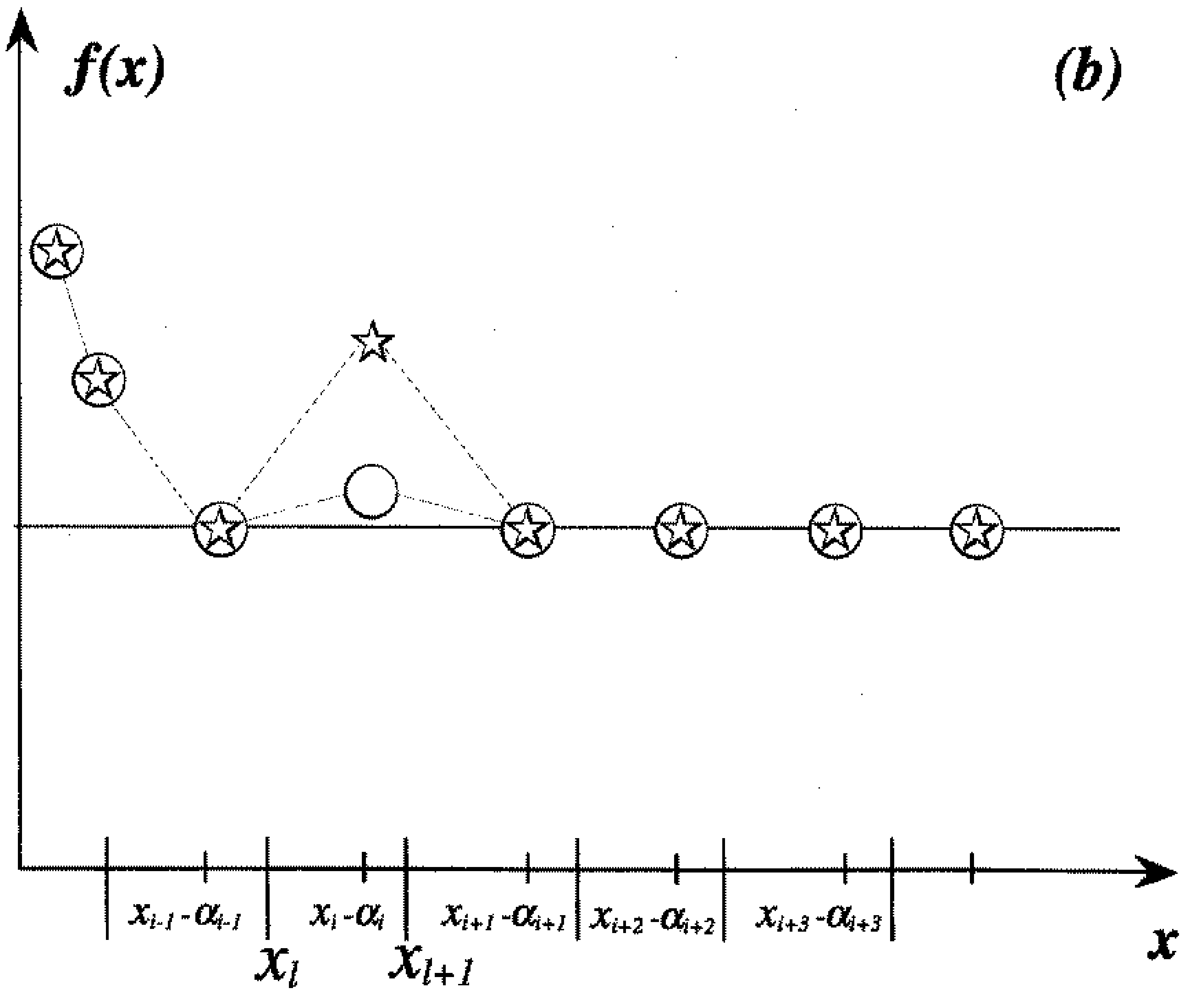


Figure 2. Schematics for monotonic filtering: (a) sampling a high-order interpolating function (raw values at the Eulerian meshpoints denoted by '•', and interpolated values at Lagrangian mesh-points by '☆'; (b) interpolated values at Lagrangian mesh-points after application of the SYS96 filter (hollow stars) and the new monotonic filter (open circles) (coincident values denoted by a hollow star within a circle); (c) the 5-mesh-length support used to identify acceptable local extrema (values before interpolation denoted by '•', and after interpolation to $x_i - \alpha_i$ by a hollow star).

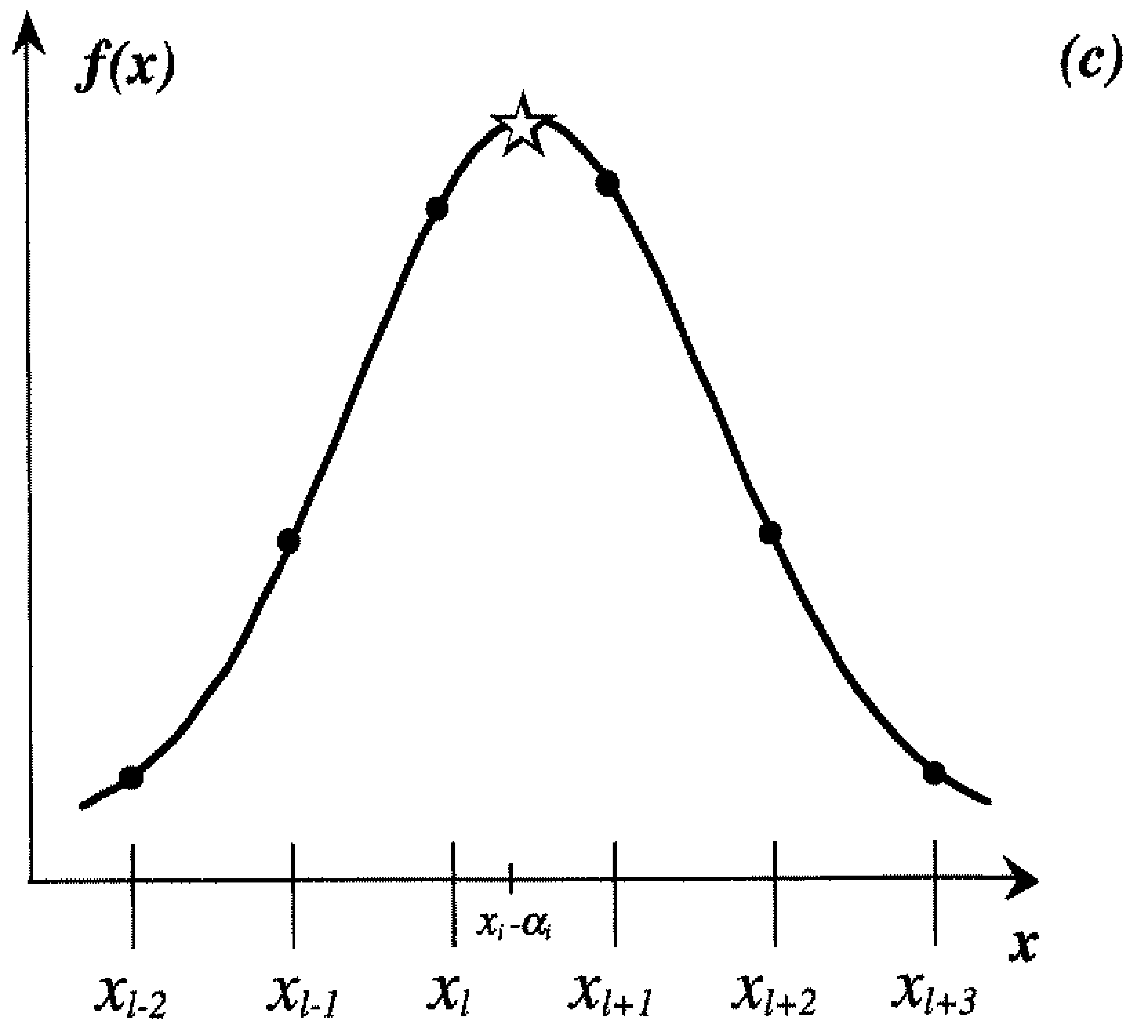


Figure 2. Continued.

inadequately respected. In the present context, if the velocity field is insufficiently smooth, this can lead to an ill-behaved intermediate mesh with too much mesh-distortion for cascade interpolation to be accurate. This is most likely to occur for regions of intense deformation (e.g. frontal regions) and for wind-fields with irregular fluctuations ('noise'). The use of cubic interpolation to define the intermediate mesh exacerbates this problem since a Lagrangian curve can now intersect an Eulerian mesh-line more than once, leading to a breakdown of uniqueness in the intermediate mesh generation and reduced algorithmic robustness.

Consider the situation (see the solid circles of the Y_d curve of Fig. 1(b)), where the data defining this Y-curve are a little noisy but still such that the y-coordinate is a monotonic function of the x- coordinate. Because the data are monotonic, it is expected that the Y_d-curve should intersect the x-line $y = y_k$ only once. When using cubic interpolation to define the intermediate mesh, it is nevertheless possible to generate (Fig. 1(b) of the present paper and Fig. 4 of PL91) three subgrid-scale intersection points, two of which are spurious, instead of the expected single intersection. This is, basically, a Gibbs phenomenon, induced by a very strong local gradient, which spuriously distorts the Lagrangian mesh and thereby reduces the accuracy of cascade interpolation. On the other hand, the linear algorithm of subsection 2(a) cannot generate such spurious subgrid-scale intersections. As a result, the Lagrangian mesh is defined more smoothly and there is an increased likelihood that the continuity assumptions required for cubic interpolation of the advected fields on this mesh will be respected. However, it should be remembered (PL91) that while this enhanced robustness is likely to provide some benefit, it only mitigates but does not circumvent the usual restriction that the directional deformation Courant number should be less than unity (Pudykiewicz *et al.* 1985; PL91).

In this regard, the statement by SY97 that "... the stability criterion of conventional semi-Langrangian schemes comes from the iteration process for backward trajectories" is somewhat misleading. The criterion referred to is really one of convergence rather than stability per se. To illustrate what happens, it suffices to consider the problem in 1-D. When the iterative procedure fails to converge, it is because the deformation is so strong that the numerically determined trajectories cross, thereby violating the condition for solution uniqueness. While an explicit computation of the trajectories always yields an upstream or downstream location, this does not really help matters since, all other things being equal, it will be so wrong as to render the ensuing solution meaningless.

To address this numerical problem, either the time-step has to be sufficiently reduced to prevent trajectory crossing, or a different method (possibly iterative, possibly explicit) must be adopted or devised to prevent this occurring during the specified time-step. This is, however, easier said than done. Trajectories can be computed in a variety of ways, but all break down for flows with strong deformation at sufficiently large time-step (typically characterized by the directional-deformation Courant-number being of the order of unity), because of the finite spatio-temporal sampling of the velocity field. Trajectory algorithms exist that allow for curved trajectories instead of the straight-line ones usually employed, and, in principle, they should therefore allow somewhat longer time-steps. In practice however, these algorithms can be dangerous since they make further assumptions concerning the spatio-temporal continuity of the velocity field. These may not be respected, e.g. by noisy data, and may degrade rather than improve solution accuracy.

In our opinion, the claims made by SY97 that "Compared to the conventional semi-Lagrangian schemes that employ backward trajectories, semi-Lagrangian schemes employing forward trajectories have the advantage that trajectories can be easily calculated with great accuracy ..." and that their procedure "... has virtually no restrictions" are speculative. The reader is not told how to compute forward trajectories with greater accuracy than backward ones, nor why such a procedure could not be made to work equally well for backward trajectories. Furthermore, there is no experimental substantiation since all the integrations described were performed using exact trajectories. As demonstrated below, when the trajectories are specified exactly, backward-trajectory variants of cascade interpolation for passive advection give as good a solution as forward-trajectory ones.

For both forward- and backward-trajectory semi-Lagrangian schemes, the trajectories leading to the definition of the Lagrangian mesh must, in general, be determined numerically from known values on a Cartesian mesh, rather than being specified analytically. For both cases, the first-order ordinary differential equation (1) is solved over the same time-interval using the same known wind information, and the only difference is in the specification of the initial/final condition. For forward trajectories, the initial condition is that the trajectory starts at a given mesh-point and Eq. (3) is integrated forward in time, whereas for backward trajectories the final condition is that the trajectory ends at a given meshpoint and Eq. (3) is integrated backwards in time. Therefore, we argue that if an accurate method exists to compute either backward or forward trajectories, then it should be equally applicable for the other case by simply integrating Eq. (3) in the opposite direction in time, subject to the appropriate specified initial/final condition. As noted by Staniforth and Côté (1991), it is of particular importance that the approximation of the trajectory equation (3) be $O(\Delta t^2)$ accurate.

For coupled sets of equations where the velocity field also has to be forecast, and in the context of an implicit or semi-implicit treatment of gravitational-oscillation terms, it appears to be more straightforward to couple the variables and equations together on a regular Eulerian mesh with a backward-trajectory variant of semi-Lagrangian advection, than it is to do so on the curvilinear Lagrangian mesh of a forward-trajectory one.

As noted by PL91, the overhead penalty for determining the mesh-points of the intermediate mesh depends only upon the velocity field. As such, it needs to be done only once, regardless of the number of fields to be interpolated at upstream or downstream points. Its costs can therefore be amortized over the cost of evaluating multiple fields. For a typical baroclinic forecast-model there will be a half-dozen or so such fields. For a typical air-quality transport-model there will be dozens of chemical-species fields, and cascade interpolation is then even more attractive and cost effective.

5. Numerical experiments

To test the proposed cascade-interpolation schemes, two test-problems are considered: solid-body rotation and deformational flow. The trajectories are evaluated exactly in all experiments, as by SY97, to focus attention on the performance of cascade interpolation better. All 1-D interpolations of the advected variable ψ are performed using cubic splines. A series of eight tests is performed for each test problem to evaluate solution accuracy as a function of a scheme's trajectory-direction (backward vs. forward), its monotonic filter (SYS96 vs. the one proposed here), and the order of the interpolator used to determine the intermediate mesh (linear vs. cubic spline). Regarding the latter, note that the 'economic' version of the SY97 algorithm corresponds to using cubic-spline interpolation to determine the points of intersection of the Y-curves of the Lagrangian mesh with the x-lines of the Eulerian one.

The performance measures chosen to evaluate the integrations are

$$E_1 = \frac{\iint\{\psi - \min(\psi_0)\}\,\mathrm{d}x\,\mathrm{d}y}{\iint\{\psi_0 - \min(\psi_0)\}\,\mathrm{d}x\,\mathrm{d}y}, \tag{20}$$

$$E_2 = \frac{\iint(\psi^2)\,\mathrm{d}x\,\mathrm{d}y}{\iint(\psi_0^2)\,\mathrm{d}x\,\mathrm{d}y}, \tag{21}$$

$$E_{\mathrm{rms}} = \left\{\iint(\psi - \psi_{\mathrm{a}})^2\,\mathrm{d}x\,\mathrm{d}y\right\}^{1/2}, \tag{22}$$

where the integrals are evaluated using the trapezoidal rule, ψ_{a} is the analytic solution, and ψ_0 is its evaluation at initial time. The first two quantities are related to the mass and square mass. They are conserved for the exact solution, and have been normalized to unity. Note that a constant has been introduced in the first of these to avoid possible division by zero and allow its normalization.

(a) *Solid-body rotation*

The solid-body rotation of a symmetric slotted cylinder is examined with parameters set to be identical to those used in one of the SY97 experiments. The cylinder (displayed at initial time in Fig. 3(a) of SY97) rotates six times with constant angular velocity $\omega = 0.3635 \times 10^{-4}\ \mathrm{s}^{-1}$ about the centre of the domain $[-1/2, 1/2] \otimes [-1/2, 1/2]$, and it is sampled on a uniform mesh of mesh length $h = 1/100$. The time-step for the experiments is 1800 s, and one revolution of the cylinder is completed every 96 time-steps.

Results of four integrations after six revolutions (576 time-steps) using linear interpolation to determine the intermediate mesh are summarized in Table 1. A further four integrations reveal that, to the number of figures displayed, the values are identical when the mesh-intersection points are determined by cubic-spline interpolation. Figure 3 displays the slotted cylinder after six revolutions for the forward-trajectory case using linear interpolation to determine these points. Corresponding plots for the other seven experiments

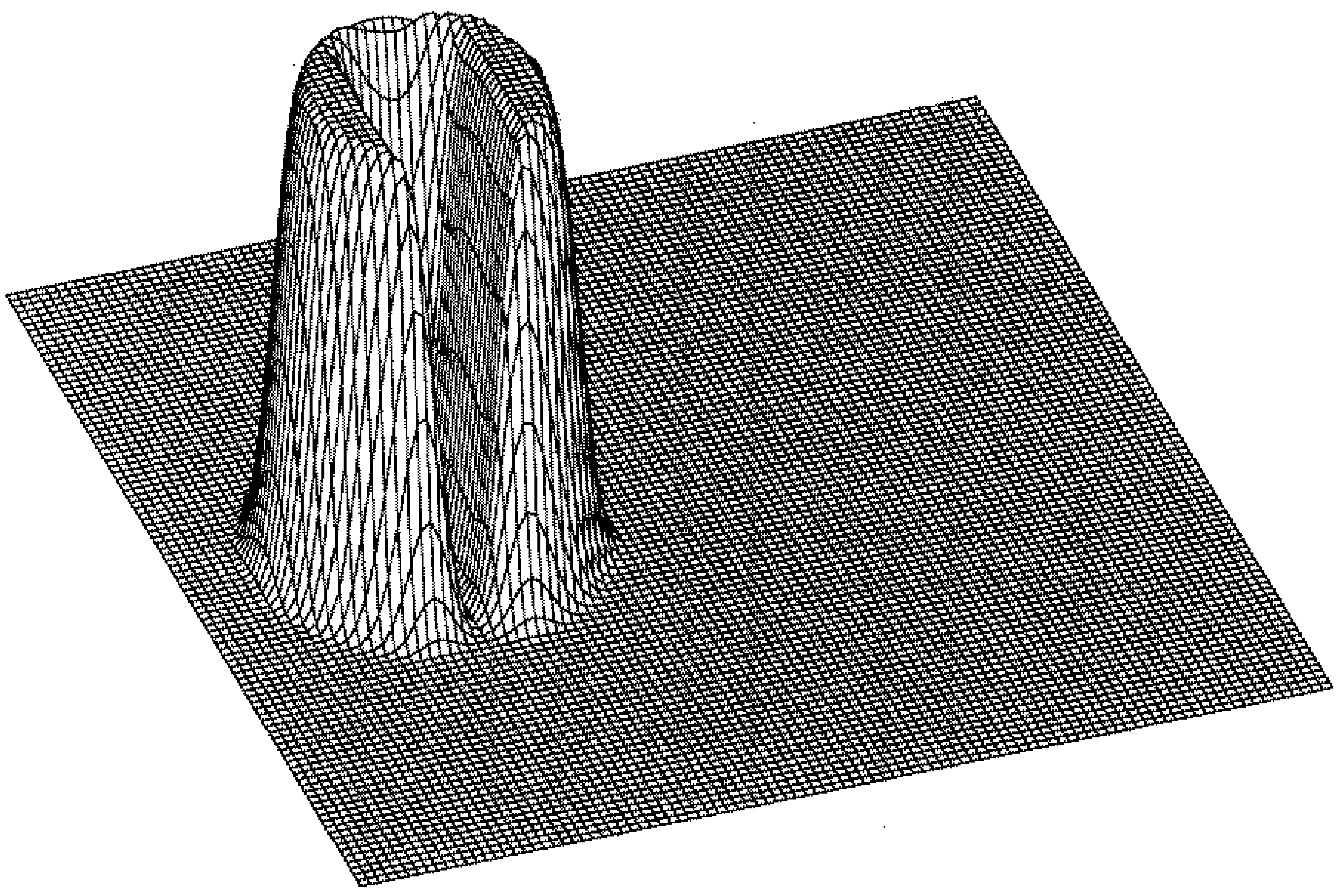


Figure 3. The 'slotted cylinder' after 6 revolutions (576 time-steps) using a forward-trajectory cubic-spline cascade scheme with the new monotonic filter, and linear interpolation to determine mesh intersection points.

TABLE 1. SLOTTED CYLINDER TEST USING BACKWARD AND FORWARD TRAJECTORY SCHEMES AND CASCADE INTERPOLATION

Analytic trajectory	Monotonic limiter	$\frac{\int \psi}{\int \psi_0}$	$\frac{\int \psi^2}{\int \psi_0^2}$	R.m.s. error
Backward	New	1.007	0.808	0.0684
Forward	New	1.009	0.809	0.0684
Backward	SYS96	1.009	0.809	0.0683
Forward	SYS96	1.010	0.811	0.0683

The intermediate mesh is determined by linear interpolation. Results are tabulated after 6 rotations (576 time-steps).

are almost indistinguishable and are not shown: the only visible difference noted is that the new filter captures the flatness of the solution a little better in the immediate vicinity of the discontinuity, as expected from the above arguments regarding the Gibbs phenomenon. Figure 3 can be compared with the corresponding results shown in Fig. 3(b)) of SY97 for their 'full-cost' algorithm. From the eight integrations performed, it is concluded that for this test problem there is no loss of accuracy when using the simpler and more economical linear-interpolation algorithm proposed herein to determine mesh-intersection points, the new monotonic filter addresses the Gibbs-phenomenon deficiency of the SYS96 filter, and the cascade interpolation gives equally good solutions, whether schemes use backward or forward trajectories.

(*b*) *Idealized cyclogenesis*

The idealized cyclogenesis problem described by Doswell (1984) and Davies-Jones (1985) has been used for scalar-advection tests by Rančić (1992), Hólm (1995), and SY97. The test case considered here is that given by SY97. The flow field is highly deformational and more challenging than solid-body rotation. A steady circular vortex with a tangential

TABLE 2. IDEALIZED CYCLOGENESIS TEST USING BACKWARD AND FORWARD TRAJECTORY SCHEMES AND CASCADE INTERPOLATION

Analytic trajectory	Monotonic limiter	$\frac{\int \psi - \min \psi_0}{\int \psi_0 - \min \psi_0}$	$\frac{\int \psi^2}{\int \psi_0^2}$	R.m.s. error
Backward	New	1.0001	0.985	0.0692
Forward	New	1.0001	0.985	0.0693
Backward	SYS96	1.0001	0.985	0.0692
Forward	SYS96	1.0001	0.985	0.0693

The intermediate mesh is determined by linear interpolation. Results are tabulated after five time-units (16 time-steps).

velocity $v_T(r) = v_0 \operatorname{sech}^2(r) \tanh(r)$ is defined, where r is the radius of the vortex and v_0 is a value chosen such that the maximum value of v_T never exceeds unity. The initial condition of the advected scalar is $\psi(x, y, 0) = -\tanh\{(y - y_c)/\delta\}$, where δ is the characteristic width of the front zone. The analytic solution at time t is

$$\psi(x, y, t) = -\tanh\left\{\frac{(y - y_c)}{\delta}\cos(\omega t) - \frac{(x - x_c)}{\delta}\sin(\omega t)\right\} \tag{23}$$

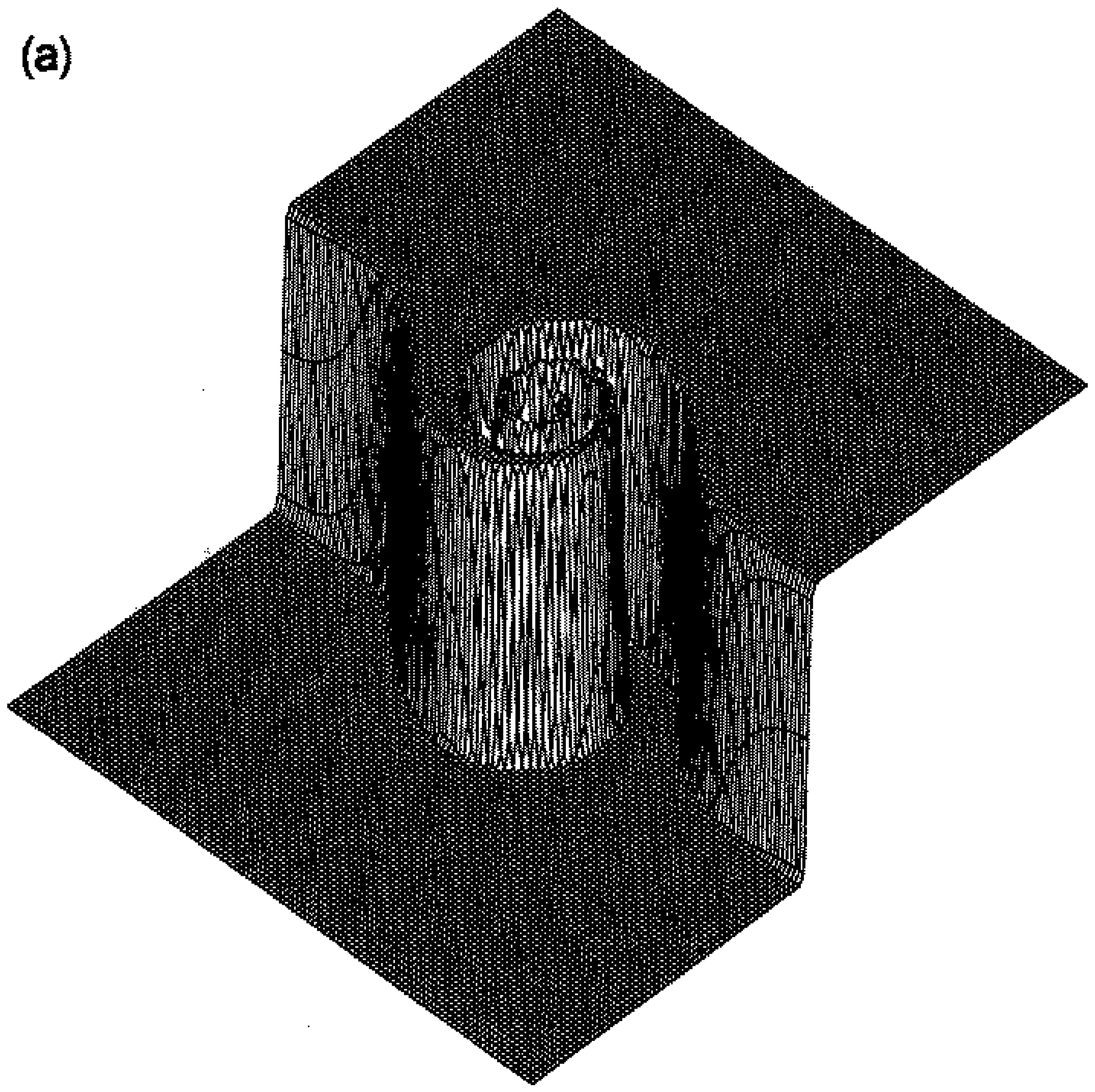
where (x_c, y_c) is the centre of the vortex that has angular velocity $\omega = v_T/r$. The domain of integration is a square of side ten non-dimensional units. The value of the parameter δ is set to 0.05, which corresponds to 'non-smooth' cyclogenesis, and the integration time is five time units.

Results after 16 time-steps for semi-Lagrangian advection, with both forward and backward trajectories, on a 129×129 mesh using linear interpolation to determine the intermediate mesh, are summarized in Table 2. This corresponds to five time-units with a maximum Courant-number of approximately four. For all integrations, the numerical solution is very close to the exact one. To the given number of decimal places, the performance values are identical when the mesh-intersection points are determined by cubic-spline interpolation, except that the root mean square (r.m.s.) error is reduced by one in the fourth decimal place, which is insignificant.

Figure 4(a) displays solutions after five time units using backward trajectories, linear interpolation to determine mesh-intersection points, and the new monotonic filter. It can be compared with both the corresponding result using the SYS96 monotonic filter (Fig. 4(b)), and that of SY97 (their Fig. 4) using forward trajectories, cubic-spline interpolation to determine mesh-intersection points, and the SYS96 filter. It is concluded (cf. Fig. 4(a, b)) that the new monotonic filter again captures the flatness of the solution in the immediate vicinity of the discontinuity a little better than the SYS96 filter. Corresponding plots for the other six experiments are visually almost indistinguishable and are similar to either Fig. 4(a) or Fig. 4(b), depending upon which filter is used. Also, given the very close agreement between the results of the experiments for this test case (as determined both visually and by the performance measures), it can again be concluded that the cascade algorithm described in the present paper has an accuracy similar to that of the SY97 one, but has the advantages of being more economical and of treating the Gibbs phenomenon a little better.

Results are also shown by SY97 after five time-units with a single time-step on a half resolution 65×65 mesh using both their 'complete' intermediate mesh (their Fig. 5(a)) and their 'economic' determination of the intermediate mesh (their Fig. 5(b)). The appearance of a spurious spike in this latter solution, and its absence in the former, leads the reader to conclude that the 'economic' algorithm is somehow less accurate and robust than their

(a)



(b)

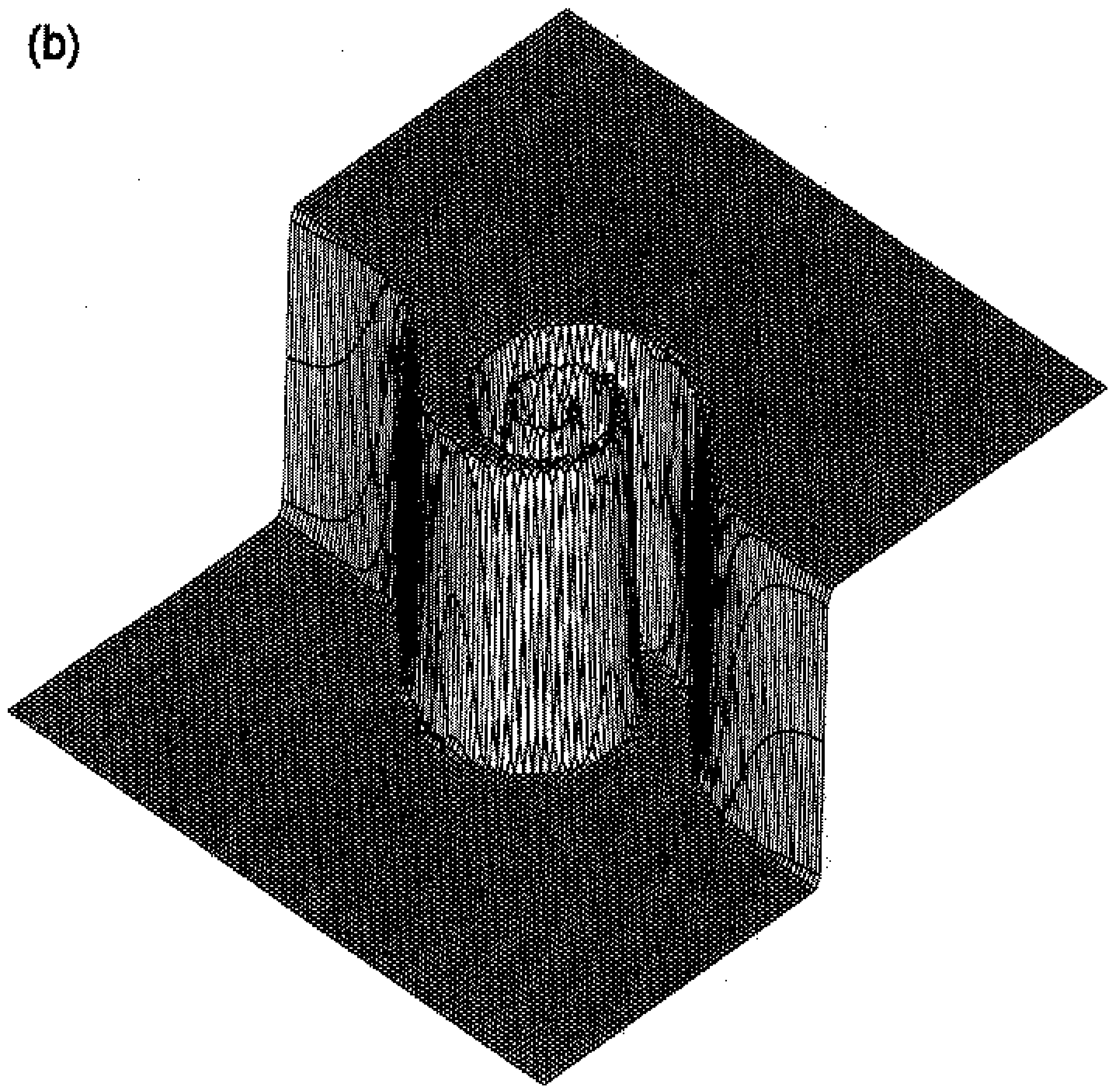


Figure 4. The cyclogenesis simulation on a 129 × 129 mesh after 5 time-units (16 time-steps), using a backward-trajectory cubic-spline cascade scheme, linear interpolation to determine mesh intersection points and two different filters: (a) the new monotonic filter; (b) the SYS96 filter.

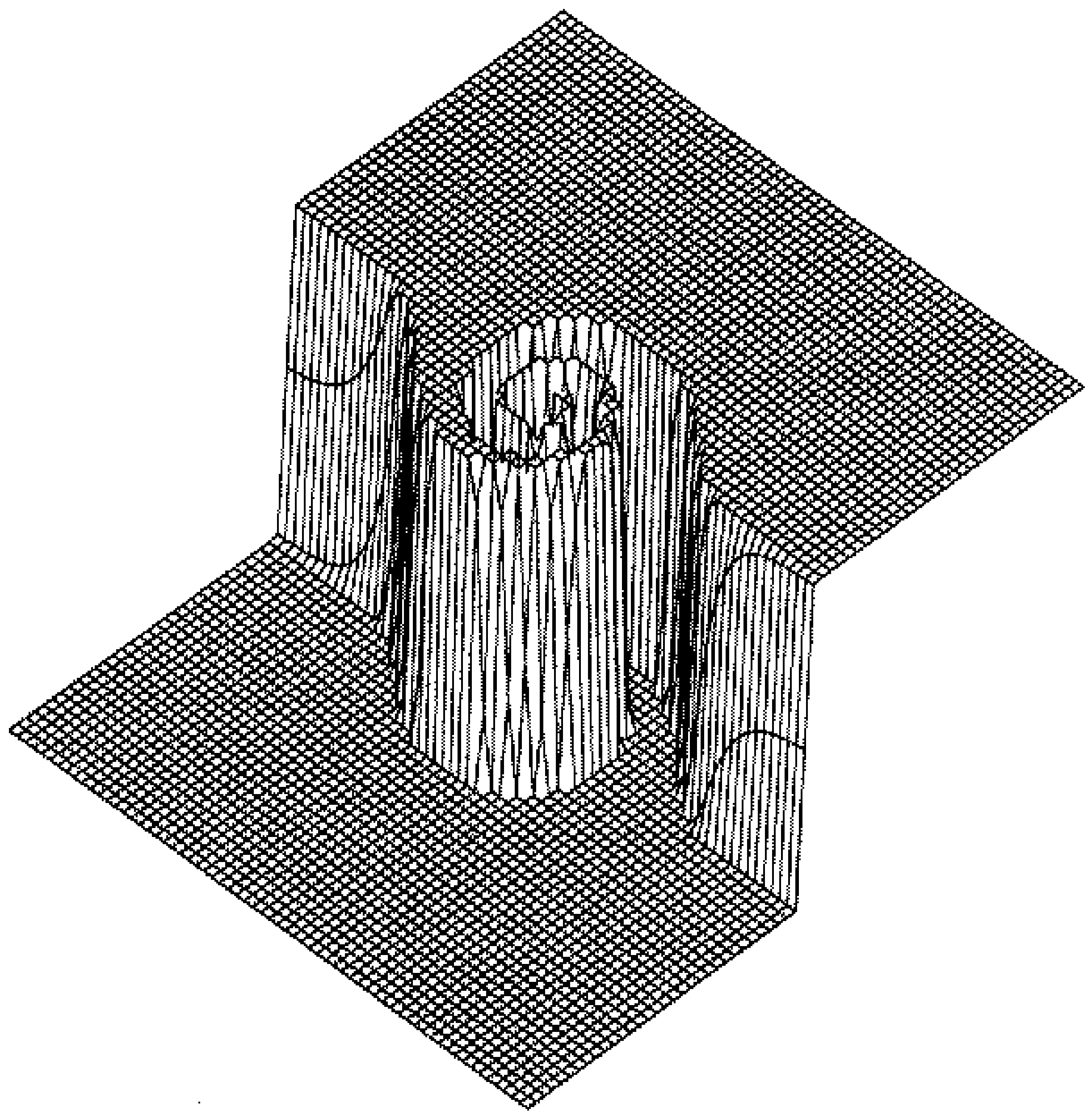


Figure 5. The cyclogenesis simulation on a 65 × 65 mesh after 5 time-units (1 time-step) using a forward-trajectory cubic-spline cascade scheme, cubic interpolation to determine mesh-intersection points, and the SYS96 filter.

'full-cost' one. We have not only repeated the experiment that led to their Fig. 5(b), but have also performed seven similar integrations to examine all eight possible combinations of backward vs. forward trajectories, of linear vs. cubic-spline interpolation to determine mesh-intersection points, and of the SYS96 filter vs. the new one. None of these solutions exhibits the spike observed in the SY97 results and, furthermore, they all closely resemble one another. In particular, they resemble the result displayed in Fig. 5 of the present paper, obtained under the same ostensible conditions as Fig. 5(b) of SY97. We therefore conclude that the spurious SY97 spike is a misleading artefact of their particular implementation, and that it does not represent a deficiency in the 'economic' variant of the algorithm per se.

6. Conclusions

From the test integrations it is concluded that:

- there is no loss of accuracy when using our proposed simpler, more efficient, and more robust algorithm to determine the mesh-intersection points for cascade interpolation;
- the proposed modified monotonic filter addresses the Gibbs-phenomenon deficiency of the SYS96 filter; and
- these proposed monotonic cascade interpolation algorithms work equally well for backward- or forward-trajectory variants of semi-Lagrangian schemes.

It is also concluded that the curious and spurious spike obtained in one of the SY97 integrations is a misleading product of their particular implementation, and does not represent a deficiency in their 'economic' variant of cascade interpolation per se.

Acknowledgements

The preparation of schematic figures by Yves Chartier, and Monique Tanguay's careful review of a draft of this paper, are both gratefully acknowledged. We also wish to thank Wen-Yih Sun and Kao-San Yeh for making their SY97 paper available to us before publication. Finally, the first author thanks the Natural Sciences and Engineering Research Council of Canada for providing a visiting fellowship.

References

Bates, J. R., Semazzi, F. H. M., Higgins, R. W. and Barros, S. R. M. 1990 Integration of the shallow-water equations on the sphere using a vector semi-Lagrangian scheme with a multi-grid solver. *Mon. Weather Rev.*, **118,** 1615–1627

Bermejo, R. and Staniforth, A. 1992 The conversion of semi-Lagrangian advection schemes to quasi-monotone schemes. *Mon. Weather Rev.*, **120,** 2622–2632

Davies-Jones, R. 1985 Comments on "A kinematic analysis of frontogenesis associated with a non-divergent vortex". *J. Atmos. Sci.*, **42,** 2073–2075

Doswell, C. A. 1984 A kinematic analysis of frontogenesis associated with a non-divergent vortex. *J. Atmos. Sci.*, **41,** 1242–1248

Hólm, E. V. 1995 A fully two-dimensional, non-oscillatory advection scheme for momentum and scalar transport equations. *Mon. Weather Rev.*, **123,** 536–552

Leslie, L. M. and Purser, R. J. 1995 Three-dimensional mass-conserving semi-Lagrangian schemes employing forward trajectories. *Mon. Weather Rev.*, **123,** 2551–2566

McDonald, A. 1987 Accuracy of multiply-upstream, semi-Lagrangian advective schemes II. *Mon. Weather Rev.*, **115,** 1446–1450

Pudykiewicz, J., Benoit, R. and Staniforth, A. 1985 Preliminary results from a partial LRTAP model based on an existing meteorological forecast model. *Atmos.–Ocean*, **23,** 267–303

Purser, R. J. and Leslie, L. M. 1991 An efficient interpolation procedure for high-order three-dimensional semi-Lagrangian models. *Mon. Weather Rev.*, **119,** 2492–2498

1994 An efficient semi-Lagrangian scheme using third-order semi-implicit time integration and forward trajectories. *Mon. Weather Rev.*, **122,** 745–756

Rančić, M. 1992 Semi-Lagrangian piecewise biparabolic scheme for two-dimensional horizontal advection of a passive scalar. *Mon. Weather Rev.*, **120,** 1394–1406

1995 An efficient, conservative, monotone remapping for semi-Lagrangian transport algorithm. *Mon. Weather Rev.*, **123,** 1213–1217

Robert, A. 1981 A stable numerical integration scheme for the primitive meteorological equations. *Atmos.–Ocean*, **19,** 35–46

Staniforth, A. and Côté, J. 1991 Semi-Lagrangian integration schemes for atmospheric models – a review. *Mon. Weather Rev.*, **119,** 2206–2223

Staniforth, A. and Pudykiewicz, J. 1985 Reply to comments on and addenda to "Some properties and comparative performance of the semi-Lagrangian method of Robert in the solution of the advection-diffusion equation." *Atmos.-Ocean*, **23,** 195–200

Sun, W.-Y. and Yeh, K.-S. 1997 A general semi-Lagrangian advection scheme employing forward trajectories. *Q. J. R. Meteorol. Soc.*, **123,** 2463–2476

Sun, W.-Y., Yeh, K.-S. and Sun, R.-Y. 1996 A simple semi-Lagrangian scheme for advection equations. *Q. J. R. Meterol. Soc.*, **122,** 1211–1226

Temperton, C. and Staniforth, A. 1987 An efficient two-time-level semi-Lagrangian semi-implicit integration scheme. *Q. J. R. Meteorol. Soc.*, **113,** 1025–1039