An Adaptive Semi-Lagrangian Advection Scheme and Its Parallelization

JÖRN BEHRENS

Alfred-Wegener-Institut, Institute for Polar and Marine Research, Bremerhaven, Germany

(Manuscript received 17 July 1995, in final form 25 March 1996)

ABSTRACT

A semi-Lagrangian advection scheme has been implemented for the linear passive advection equation. The advected scalar field is represented in terms of finite-element bases on a self-adaptive grid.

Results concerning the accuracy of the numerical scheme compare well to those with finite-difference schemes on regular meshes found in the literature. Quasi-monotone and quasi-conservative interpolation schemes have been implemented.

A parallelized version of the code shows good performance on a KSR-1 virtual shared memory computer.

1. Introduction

Numerical models of atmospheric or ocean circulation still represent highly resource consuming applications, so called grand challenges (Simon 1992). There are two main methods (sometimes complementary) of overcoming restrictions due to high computational demands: parallel algorithms can be used in order to reduce time complexity of the application, and adaptive strategies can help to reduce the arithmetic complexity. Both attempts are combined in this article. A numerically efficient method for advection problems—the semi-Lagrangian method—is well suited for parallelization. An introduction to the semi-Lagrangian method is given in (Staniforth and Côté 1991). Additionally, an adaptive strategy, refining the discretization mesh locally, is included in this advection scheme.

There are two major problems to be addressed: When dealing with locally adapted triangular grids, an accurate semi-Lagrangian interpolation scheme for irregularly distributed data points is needed. When aiming at efficient parallelization, load balancing and data locality are essential.

Both problems have only recently been investigated in a related context. Le Roux et al. (1995) have developed accurate methods for the semi-Lagrangian interpolation on irregular meshes, but their advection algorithm is not equipped with a self-adapting strategy nor does it include the use of triangular unstructured meshes. On the other hand, the interpolation utilized in this paper differs from their method that is based on dual kriging schemes. Parallelization of a semi-Lagrangian scheme has been investigated by Thomas and Côté (1994, personal communication). Static data distribution, as used in their approach, imposes a restriction to the Courant number. Here a dynamic data distribution mechanism is applied avoiding such a restriction.

The methods described in this article reflect a first step toward an adaptive finite-element model of the shallow-water equations. Finite-element methods have been successfully combined with the semi-Lagrangian advection by several authors (Tanguay et al. 1989; Staniforth and Temperton 1986; Côté et al. 1993). Adaptive methods, however, have not been applied in this context.

As passive advection is a good starting point for investigation of advection schemes, this application has been chosen as a test problem. Many authors have already investigated properties of new advection schemes in the framework of this particular problem (Bermejo and Staniforth 1992; Priestley 1993; Zalesak 1979), so using the same model problem helps in comparison of the results.

This article describes some details of an implementation of a finite-element semi-Lagrangian model for the passive advection equation

$$\frac{dF}{dt} = 0 \quad \text{with} \quad \frac{dF}{dt} = \frac{\partial F}{\partial t} + \mathbf{a}\cdot\nabla F, \qquad (1)$$

where $F = F(\mathbf{x}, t)$ is a scalar advected by a flow field $\mathbf{a} = \mathbf{a}(\mathbf{x}, t)$, $(\mathbf{x}, t) \in \Omega \times I$, $\Omega \subset R^2$ is a two-dimensional domain, and $I \subset [0, T]$ is a time interval. The initial condition is $F(\mathbf{x}, 0) = f_0(\mathbf{x})$, $\mathbf{x} \in \Omega$, and boundary conditions $F(\mathbf{x}, t) = f_t(\mathbf{x})$, for all $t \in I$, where $\mathbf{x}$

Corresponding author address: Jörn Behrens, Alfred-Wegener-Institut, Institute for Polar and Marine Research, P.O. Box 120161, D-27515 Bremerhaven, Germany.
E-mail: jbehrens@awi-bremerhaven.de

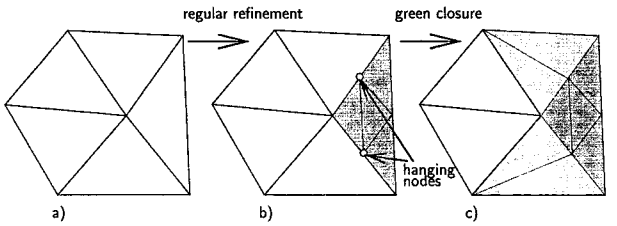

FIG. 1. An initial coarse triangulation (a) is locally refined by regular refinement (b). This leads to hanging nodes, which are not admissible in a FEM triangulation. Triangles containing hanging nodes are then bisected by a *green closure* (c) in order to reestablish admissible triangulation.

$\in \Gamma = \partial\Omega$, $\partial\Omega$ the boundary of Ω, have to be provided. Discretizing (1) by the semi-Lagrangian method combined with some spatial discretization yields

$$\frac{F(\mathbf{x}_i, t_n + \Delta t) - F(\mathbf{x}_i - \alpha_i, t_n)}{\Delta t} = 0, \qquad (2)$$

where $\mathbf{x}_i$, $i = 1, \cdots, N$ are the mesh coordinates, α_i is the displacement vector corresponding to the mesh point $\mathbf{x}_i$, and Δt denotes the time step.

The remainder of this article is organized as follows: section 2 describes the construction of a locally adaptive triangular mesh. The techniques involved are presented to the convenience of the reader unfamiliar with triangular grid generation. As interpolation of the upstream values is crucial for semi-Lagrangian methods, section 3 covers this issue in detail. Section 4 describes some details on parallelization of a semi-Lagrangian code using a KSR-1 virtual shared memory computer. It is pointed out that the virtual shared memory mechanism of the machine helps to ease the efficient parallelization of an unstructured problem like the one described here. Test results are given in section 5, and concluding remarks are made in section 6.

2. The adaptive grid

The domain Ω is triangulated semiautomatically by a grid generator based on regular refinement. A coarse triangulation has to be defined manually consisting of only a few, for example 10, elements. These initial triangles are refined regularly and globally up to a given refinement level. Further refinement is performed adaptively and locally up to a given finest level.

Regular (or red) refinement of a triangle is achieved by joining the midpoints of all three edges. Daughter triangles of the refined mother triangle are similar to their mother, which means that shape regularity of the triangulation (i.e., the lower bound for the interior angles) is preserved. Unfortunately the procedure produces hanging nodes. A hanging node is a node on an unrefined edge of a triangle. After local red refinement those triangles containing hanging nodes are treated separately in order to close the triangulation (i.e., removing all hanging nodes). Hanging nodes are prevented by bisecting the triangle (i.e., joining the hanging node to the opposite vertex), a so-called green re-

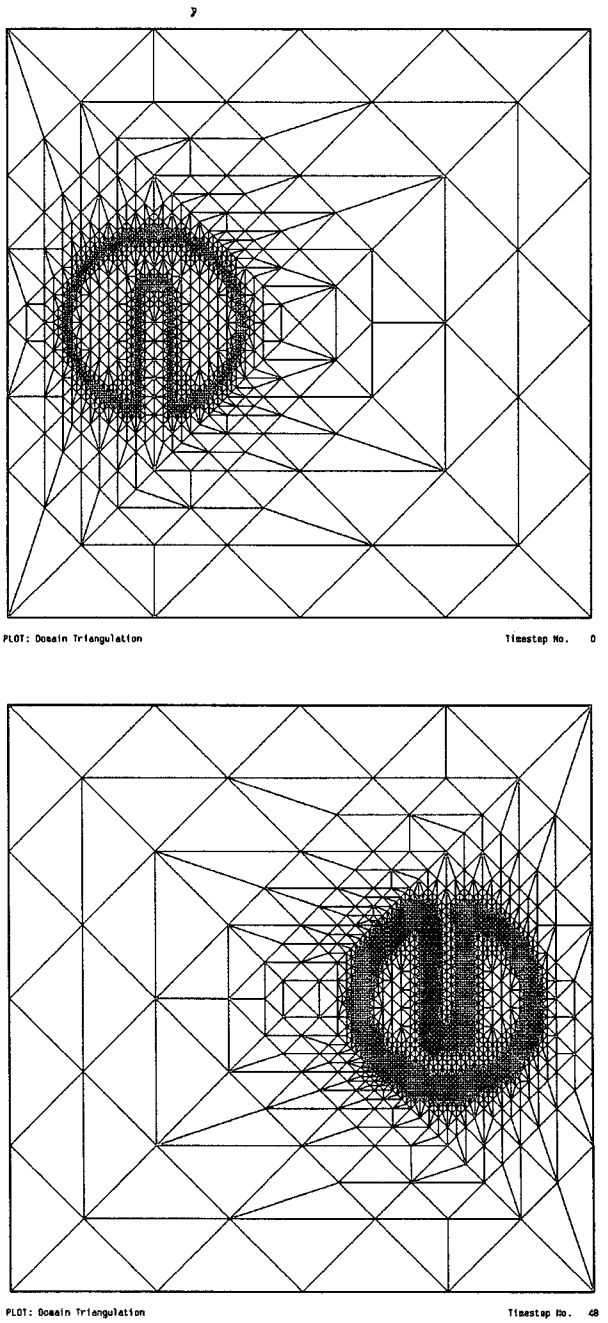

FIG. 2. Locally refined grid with four initial elements, global refinement to the third level, and additional adaptive refinement to level 8; initial time step and time step 48 after one-half revolution.

TABLE 1. Bicubic interpolation, exact trajectory calculation.

Time step	Nodes	Elements	RFM	RSM	Max	Min	l^2 norm
96	3954	7886	1.00672	0.94417	4.49886	−0.57320	0.01868
192	4364	8706	1.01313	0.93079	4.49664	−0.59777	0.02589
288	4620	9212	1.01594	0.92208	4.47798	−0.65417	0.03117
384	4797	9564	1.01383	0.91575	4.47267	−0.74487	0.03547
480	5017	10 004	1.01350	0.91077	4.47662	−0.78444	0.03913
576	5101	10 172	1.01295	0.90646	4.49849	−0.82402	0.04233

finement or *green closure.* Bisected triangles are not allowed to be bisected again in order to preserve shape regularity. A detailed description of triangulations is given in (Mitchell 1989) or (Ho-Le 1988). Figure 1 shows a locally and regularly refined triangulation with green closures to prevent hanging nodes. For the test case described in section 5, the corresponding locally refined grid is shown at the initial time, and after half a revolution in Fig. 2.

As the procedure for integrating the passive advection equation is an explicit one, and no partial derivatives are involved in the context of the semi-Lagrangian method, a solution of an elliptic problem is not required. Therefore, no real finite-element method is used, but the scalar function is represented in terms of finite-element basis functions on a triangular grid. Continuous piecewise linear basis functions are utilized in this case. A finite-element discretization of a function $F(x, y)$ is then given by a linear combination of the basis functions b_i:

$$F(x, y) = \sum_{i=1}^{N} F_i b_i(x, y), \tag{3}$$

where N is the number of nodes and $F_i = F(x_i, y_i)$ are nodal function values (for details on the FEM see Ciarlet 1978; Johnson 1990; Wait and Mitchell 1985).

The adaptive refinement strategy requires some measure to determine the location of refinement. A posteriori error estimation is commonly used in order to estimate the calculation error and react to large local errors by refining the grid (Verführth 1993). In the special case of passive advection, no real error estimation is performed, in fact, the local gradient of the scalar is used as an indicator for refinement. The local gradient $|\nabla F|_\tau$ is defined by the gradient of F restricted to element $\tau \in T$, where $T = \{\tau: \tau$ is an element$\}$ represents the triangulation. Refining the grid where the gradient is steep, which indicates that the function is nonsmooth, leads to a better localization and resolution of the interpolation.

The adaptive semi-Lagrangian method is given in the following algorithm. Note that the semi-Lagrangian calculation is executed several times (inner iteration). The adaptive algorithm is efficient, when the number of unknowns is highly reduced, otherwise the computational effort for adaptivity nullifies the gain from decreased complexity of the problem.

Algorithm 1 (adaptive semi-Lagrangian method)

1) Use (old) grid from last time step as an "initial guess" for the new (current) grid.

2) Perform a semi-Lagrangian integration step (calculate displacements, interpolate upstream values from old grid, update grid values) on the new grid.

3) Perform local refinement of those elements τ where $|\nabla F|_\tau \geq \theta_{\text{ref}} \max_{\tau \in T}\{|\nabla F|_\tau\}$, where $\theta_{\text{ref}} < 1$ is a given tolerance.

4) Perform local coarsening of those elements τ where $|\nabla F|_{\tau_{\text{child}}} \leq \theta_{\text{crs}} \max_{\tau \in T}\{|\nabla F|_\tau\}$ for at least three of four child triangles τ_{child} of mother τ, θ_{crs} again is a given tolerance, $\theta_{\text{crs}} < \theta_{\text{ref}}$.

5) If the current grid has changed, then go to step 2 (inner iteration), otherwise increase time-step counter by one and go to step 1 (outer or time-step iteration).

Note that tolerances θ_{ref} and θ_{crs} impose a relative rather than an absolute bound for refinement. Suppose $\theta_{\text{ref}} = 0.5$, then all elements τ for which the local gra-

TABLE 2. Bicubic interpolation with quasi-monotone scheme, exact trajectory calculation.

Time step	Nodes	Elements	RFM	RSM	Max	Min	l^2 norm
96	2747	5472	1.00129	0.89353	4.0	0.0	0.02423
192	2957	5892	1.00175	0.87260	4.0	0.0	0.03238
288	3271	6520	1.00242	0.85886	4.0	0.0	0.03802
384	3439	6856	1.00285	0.84816	4.0	0.0	0.04266
480	3519	7016	1.00361	0.83959	4.0	0.0	0.04658
576	3666	7310	1.00418	0.83228	4.0	0.0	0.04994

TABLE 3. Bicubic interpolation with quasi-monotone and quasi-conservative scheme, exact trajectory calculation.

Time step	Nodes	Elements	RFM	RSM	Max	Min	l^2 norm
96	2790	5558	1.0	0.88899	4.0	0.0	0.02521
192	3113	6204	1.0	0.86617	4.0	0.0	0.03401
288	3390	6758	1.0	0.85076	4.0	0.0	0.04010
384	3515	7008	1.0	0.83903	4.0	0.0	0.04494
480	3638	7254	1.0	0.82878	4.0	0.0	0.04917
576	3745	7468	1.0	0.82009	4.0	0.0	0.05285

dient is larger than 50% of the maximum local gradient will be refined.

3. Semi-Lagrangian interpolation

Interpolation is the crucial part in semi-Lagrangian advection schemes, because upstream points are located between grid points. It is widely accepted, that fourth-order interpolation (e.g., bicubic spline interpolation) represents a good compromise between accuracy and computational cost (McDonald 1984; Staniforth and Côté 1991). In the passive advection equation, no forces are present, thus dissipation and dispersion are due to numerical error only. This depends on both the accuracy of the interpolation and accuracy of the trajectory estimation. As displacements (trajectories) can be calculated exactly, differences to the results of other studies (e.g., Bermejo and Staniforth 1992) are due to interpolation only.

We use cubic-spline interpolation for arbitrarily distributed data points. Algorithms have been adopted from codes available from netlib. Accurate results are obtained from a modified algorithm TOMS 677 (Montefusco and Casciola 1989). This interpolation scheme calculates a piecewise cubic C^1 interpolant on triangles from given function values and first partial derivatives at the nodes of each triangle. To estimate the partial derivatives, a method by Akima has been adopted (Akima 1984). This combination yields the best results with respect to accuracy. Other choices for the interpolation included Renka's algorithm TOMS 624 (Renka 1984) and for gradient estimation an algorithm by Klucewicz (1978).

Cubic-spline interpolation leads to over- and undershooting near singularities or nonsmooth areas of the interpolated function. To maintain monotonicity properties of the interpolated function, Bermejo and Staniforth propose a quasi-monotone semi-Lagrangian interpolation in (Bermejo and Staniforth 1992). The basic idea is to combine a (low order) monotone interpolation with a high-order interpolation such that the combination is monotone, and still as accurate as possible. A further enhancement to this scheme is the quasi-conservative scheme of Priestley (Priestley 1993). This scheme is based on the idea of combining high-order and low-order interpolation from the quasi-monotone scheme subject to the condition that the interpolant is conservative with respect to its mass (i.e., the integral over the domain Ω is kept constant over time).

The finite-element representation of functions, evaluated between grid points, is used as low-order interpolation. With continuous piecewise linear basis functions this interpolation is of first order and is equivalent to bilinear interpolation. There are no major differences to the original methods cited above, except for the calculation of the integral. As the grid spacing is not uniform, integration needs a modified calculation of the area associated to each node. Let $\mathcal{T}_i$ be the set of all triangles τ containing node i, then the area $\mathcal{A}_i$ associated to node i is given by

$$\mathcal{A}_i = \sum_{\tau \in \mathcal{T}_i} \frac{1}{3} |\tau|,$$

where $|\tau|$ is the area of triangle τ.

4. Parallelization

The semi-Lagrangian method is very attractive when aiming at parallelization. In principle, all the calcula-

TABLE 4. Bicubic interpolation, trajectory estimation by iteration.

Time step	Nodes	Elements	RFM	RSM	Max	Min	l^2 norm
96	3954	7886	1.00721	0.94394	4.50344	−0.55089	0.01881
192	4394	8766	1.01417	0.93059	4.48707	−0.57451	0.02631
288	4596	9169	1.01822	0.92204	4.48314	−0.64780	0.03188
384	4825	9624	1.02040	0.91508	4.47585	−0.73154	0.03682
480	4972	9918	1.02020	0.91012	4.47109	−0.79072	0.04110
576	5121	10 215	1.01793	0.90593	4.50039	−0.82090	0.04507

TABLE 5. Bicubic interpolation with quasi-monotone scheme, trajectory estimation by iteration.

Time step	Nodes	Elements	RFM	RSM	Max	Min	l^2 norm
96	2756	5490	1.00108	0.89294	4.0	0.0	0.02439
192	2982	5942	1.00167	0.87248	4.0	0.0	0.03267
288	3287	6552	1.00239	0.85893	4.0	0.0	0.03855
384	3458	6894	1.00290	0.84808	4.0	0.0	0.04356
480	3536	7050	1.00347	0.83961	4.0	0.0	0.04793
576	3691	7360	1.00397	0.83227	4.0	0.0	0.05194

tions corresponding to passive advection are independent, and can thus be parallelized. On a shared memory computer this will not pose any problem. On a distributed memory computer a copy of the entire scalar field could be sent to each of the local memories associated to the processors so that calculations could then be performed in parallel without communication. However, this involves high local memory requirements and wastes most of the computer's memory resources.

Data partitioning becomes an issue as soon as each processor holds only that amount of data on which it works. With adaptive grid refinement, data partitioning becomes an irregular problem. A very convenient way of handling irregular problems on distributed memory machines is the utilization of virtual shared memory as provided by the KSR-1. The *AllCache* virtual shared memory mechanism moves data to the location where it is updated by hardware-implemented automatic caching methods (Kendall Square Research Corporation 1992).

For the user it is therefore important to keep in mind where the majority of data items are processed. As long as a major amount of data (say 85%) stays local and only a minor part of irregularly distributed data is moved, performance will be high. Typical efficiency rates for the KSR-1 range between 70% and 85% for up to 30 processors. Data movement for updating (left-hand side) is punished heavily while data movement for getting a copy (right-hand side) is relatively cheap (this is due to the fact that read access can be executed asynchronously, while write access has to be synchronized).

Nodal calculations are fixed to the processors using index arrays as introduced in Behrens (1995). An array stores node indices for each processor. Nodes have to be redistributed within the inner iteration when the grid has changed. After distributing the nodes each processor writes exclusively to its nodes. Some reads (e.g., for the estimation of first derivatives in the cubic spline interpolation) have to be performed over the processor bounds. Nodes are distributed equally in chunks of the size of one *subpage* over the processors. A *subpage* is the unit of data that is communicated by the *AllCache* mechanism in one step.

The above is quite a heuristic type of data distribution. It is, however, successful, as the results in the next section demonstrate. Observations of the index array show that neighbor indices of a node are located on the same subpage as the node itself and only one additional subpage in most cases. This is due to the order of computation in the refinement algorithm. Thus, a read access has to be performed once for computations related to the neighbor elements of one node (e.g., in the estimation of gradients). When, as seen here, the number of arithmetic operations is large enough data access does not decrease performance of the algorithm.

A parallel algorithm for the adaptive semi-Lagrangian method is given below.

Algorithm 2 (parallel adaptive semi-Lagrangian method)

1) Use (old) grid from last time step as an "initial guess" for the new (current) grid.

2) Redistribute nodes equally among the processors in chunks of the size of a subpage.

3) Perform a semi-Lagrangian integration step *in parallel,* doing all nodal calculations locally with the nodes fixed to corresponding processors using index arrays.

TABLE 6. Bicubic interpolation with quasi-monotone and quasi-conservative scheme, trajectory estimation by iteration.

Time step	Nodes	Elements	RFM	RSM	Max	Min	l^2 norm
96	2801	5580	1.0	0.88939	4.0	0.0	0.02527
192	3139	6256	1.0	0.86681	4.0	0.0	0.03411
288	3364	6706	1.0	0.85129	4.0	0.0	0.04045
384	3514	7006	1.0	0.83946	4.0	0.0	0.04571
480	3658	7294	1.0	0.82946	4.0	0.0	0.05039
576	3785	7548	1.0	0.82051	4.0	0.0	0.05476

TABLE 7. Bicubic interpolation, trajectory estimation by iteration, globally refined.

Time step	Nodes	Elements	RFM	RSM	Max	Min	l^2 norm
96	33 025	65 536	0.99939	0.93829	4.45096	−0.49834	0.01920
192	33 025	65 536	0.99895	0.92603	4.47204	−0.52839	0.02668
288	33 025	65 536	0.99858	0.91805	4.46678	−0.60097	0.03216
384	33 025	65 536	0.99828	0.91207	4.45886	−0.68423	0.03693
480	33 025	65 536	0.99801	0.90727	4.43875	−0.74229	0.04117
576	33 025	65 536	0.99778	0.90325	4.45297	−0.77262	0.04507

4) Perform grid adaption as in algorithm 1.

5) If the current grid has changed, then go to step 2 (inner iteration), otherwise increase time-step counter by one and go to step 1 (outer or time-step iteration).

This algorithm is parallel for the computations associated with the semi-Lagrangian method, and scales well with the number of processors as is shown in section 5. However, as soon as a moderate number of processors is involved, the grid generation routines become the dominant part of the program. According to Amdahl's law (Hockney and Jesshope 1988), the total speedup for increasing numbers of processors approaches only 2.86, when assuming a proportion of 35% of the serial time for the grid generation. Therefore, the total time is not decreased considerably by adding processors.

As this model for passive advection is only a test suite for the components of a model for the shallow-water equations (including an elliptic solver not present here), parallelization of the gridding routines is postponed for future investigation.

5. Tests and results

Different tests have been performed with the semi-Lagrangian adaptive finite-element model. First of all, it is important to investigate accuracy of the components by comparing it to similar models described in the literature (Bermejo and Staniforth 1992; Priestley 1993). A common test case is the slotted cylinder [originally proposed in (Zalesak 1979)] of radius 0.15, height 4, and a slot of width of 0.06 and length 0.22, in the domain $[-0.5, 0.5] \times [-0.5, 0.5]$, rotated by the flow field $\mathbf{a} = -\omega(-y, x)$, where ω is set to 0.3636×10^{-4}.

As in Bermejo and Staniforth (1992), values for the first and second moments (RFM, respectively, RSM) are calculated for diagnostic reasons. The l^2 norm of the error is also given as an indicator for numerical dissipation and dispersion. Diagnostic values are given in Tables 1, 2, and 3 for the case, where trajectories are calculated exactly. For iteratively computed trajectories, Tables 4, 5, and 6 give the corresponding numbers. Grid resolution within locally refined regions is slightly finer than that used in the cited literature (local refinement to the eighth level, that is, grid size 2^{-7}). Values in the tables indicate that the new method, involving adaptive finite-element representation of the functions, yields similar accuracy. Tables 7, 8, and 9 show the same diagnostics for the case without adaptive gridding. The numbers indicate that adaptivity does not alter the results significantly. It should be noted that, maintaining the resolution, the adaptive grid consists of only approximately 3300 nodes in the mean, whereas the uniformly refined grid is made of approximately 33 000 nodes. That means one order of magnitude can be saved with respect to memory requirements, without losing accuracy.

Figures 3 and 4 show the test case at start and after six rotations for the different semi-Lagrangian interpolation schemes mentioned in section 3. The results given here are obtained from the adaptive model with iterative estimation of the trajectories. There is no visual difference to the other cases (i.e., uniform grid, respectively, adaptive grid with exact trajectory calculation). We set the parameters $\theta_{\text{ref}} = 0.04$ and $\theta_{\text{crs}} = 0.01$. These values lead to refinement wherever only

TABLE 8. Bicubic interpolation with quasi-monotone scheme, trajectory estimation by iteration, globally refined.

Time step	Nodes	Elements	RFM	RSM	Max	Min	l^2 norm
96	33 025	65 536	1.00066	0.89446	4.0	0.0	0.02394
192	33 025	65 536	1.00132	0.87396	4.0	0.0	0.03228
288	33 025	65 536	1.00190	0.86034	4.0	0.0	0.03815
384	33 025	65 536	1.00243	0.84987	4.0	0.0	0.04313
480	33 025	65 536	1.00292	0.84124	4.0	0.0	0.04754
576	33 025	65 536	1.00339	0.83384	4.0	0.0	0.05159

TABLE 9. Bicubic interpolation with quasi-monotone and quasi-conservative scheme, trajectory estimation by iteration, globally refined.

Time step	Nodes	Elements	RFM	RSM	Max	Min	l^2 norm
96	33 025	65 536	1.0	0.89189	4.0	0.0	0.02457
192	33 025	65 536	1.0	0.86949	4.0	0.0	0.03336
288	33 025	65 536	1.0	0.85422	4.0	0.0	0.03965
384	33 025	65 536	1.0	0.84227	4.0	0.0	0.04497
480	33 025	65 536	1.0	0.83232	4.0	0.0	0.04968
576	33 025	65 536	1.0	0.82365	4.0	0.0	0.05399

a small local gradient occurs. However, in this case results do not change considerably when increasing θ_{ref} and θ_{crs}, because of the structure of the problem: wherever there are gradients, these are steep.

Serial computing time is moderate. An average time step for the plain interpolation test case with a locally adapted grid of 3357 nodes takes 10.8 s on an IBM RS/6000 320h. As an average of 1801 floating point op-

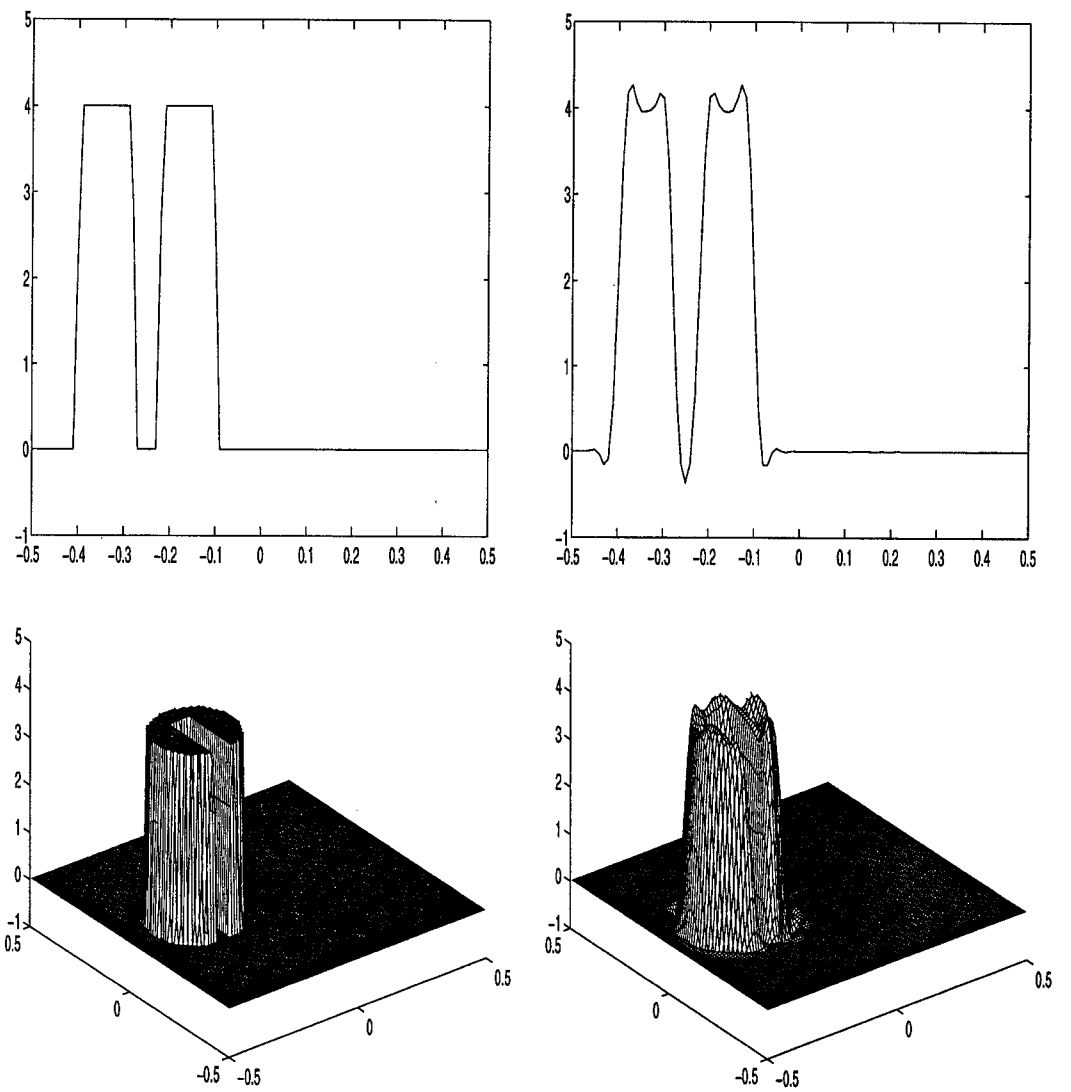

FIG. 3. From left to right: initial state of a slotted cylinder test case and the cylinder after six revolutions with plain cubic spline interpolation.

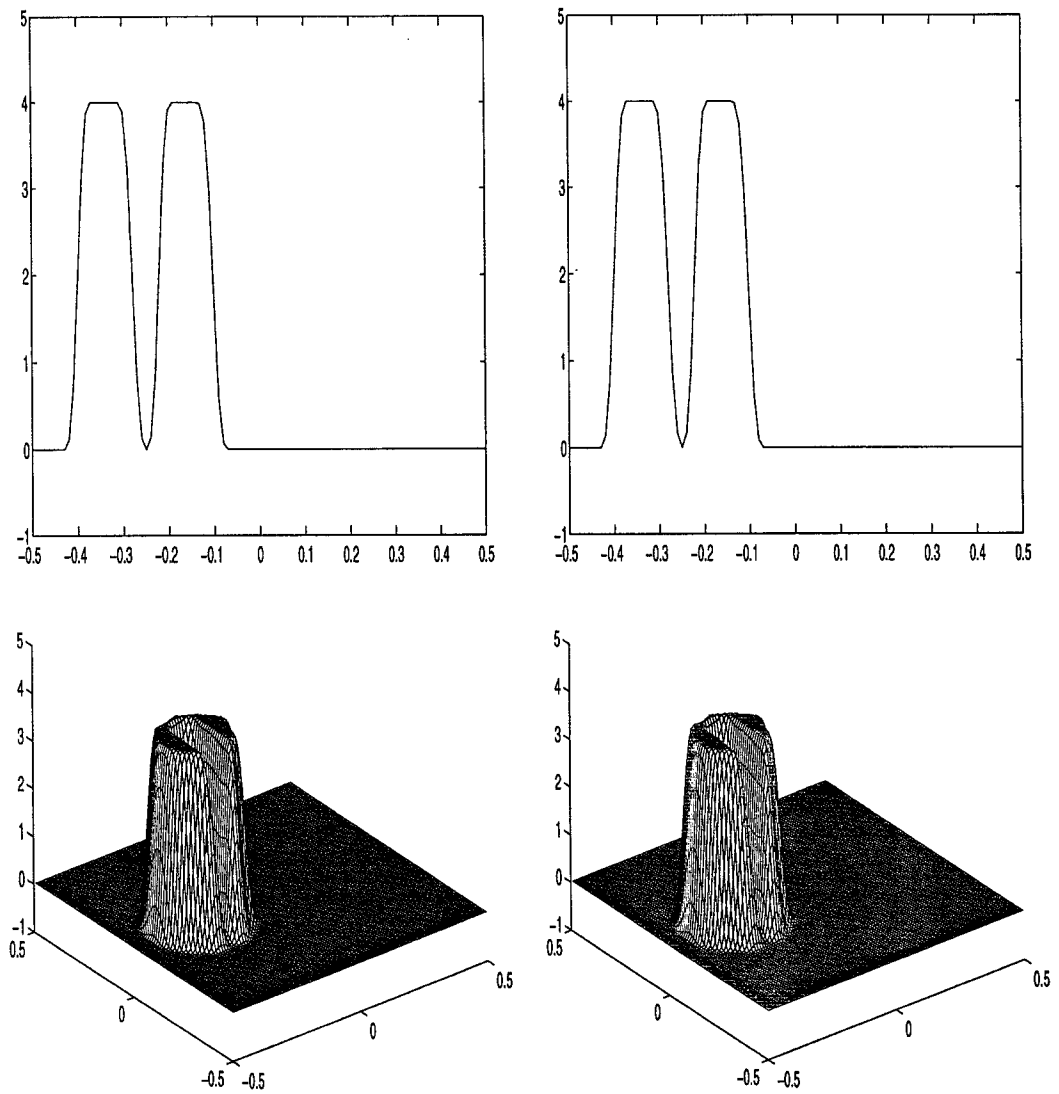

FIG. 4. From left to right: a slotted cylinder after six revolutions with quasi-monotone, respectively, quasi-conservative interpolation.

erations are executed per grid point considering three inner iterations (assuming that the number of nodes does not change considerably) this corresponds to a performance of 1.7 Mflop s^{-1}.

The proportion of the grid generation ranges between 30% and 35%. Grid generation is strictly serial, thus not parallelizable. As mentioned in section 4, serial grid generation dominates the total performance of the program as soon as more processors are involved. This fact is demonstrated by Figs. 5 and 6. Speedup and efficiency are plotted for a globally refined and a locally refined experiment, respectively. As there is no gridding necessary with global refinement, the algorithm scales well. Parallel efficiency is high (about 90% for up to 26 processors). Looking at the locally refined experiment, the efficiency of the parallel part is still high, but the overall performance deteriorates dramatically for a larger number of processors.

The tests were performed employing a model that utilizes the quasi-conservative semi-Lagrangian interpolation procedure. Time for the parallel part of one time step is 1.54 s for the globally refined example and 0.85 s for the locally refined problem with 26 processors involved. These timing results correspond to 38.7 and 27.7 Mflop s^{-1}, respectively, while the single processor performance approximately equals that of the IBM RS/6000.

Note that the parallel results for the locally adapted test case include communication costs for redistributing data. It is therefore argued that the serial grid generation does not undermine the potential for parallelism. Parallel grid generation is a nontrivial task in itself and was not aimed at within this project. However, there are several parallel grid generators available, which could be used here (Shostko and Löhner 1995; Globisch 1995).

6. Conclusions and further advances

A semi-Lagrangian adaptive finite-element algorithm for the passive advection equation has been implemented. A simple a posteriori criterion for controlling the grid refinement suffices to maintain high accuracy while saving computational costs and minimizing memory requirements. The method has been shown to compete with previously published finite difference methods on equidistant rectangular grids. Common interpolation methods for conserving properties of the interpolated function can be implemented analogously.

Parallelization proves to be efficient for the semi-Lagrangian part of the program. Overall efficiency drops because of serial grid generation. Optimization of the gridding routines will be an issue for future research.

It is planned to extend the methods implemented so far in order to solve the shallow-water equations. A finite-element elliptic equations solver has already been set up and parallelized (Hiller and Behrens 1991). The combination of these two components will yield a parallel semi-Lagrangian adaptive finite-element method for the shallow-water equations.

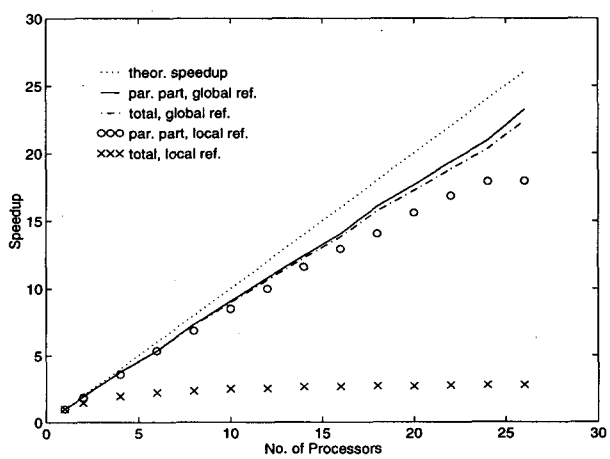

FIG. 5. Speedup of the parallel part and the total program for an experiment with eight global refinement levels (i.e., 33 025 nodes, 65 536 elements), and an experiment with eight local refinement levels maximum, three levels on the coarsest grid (i.e., 1858 nodes, 3695 elements).

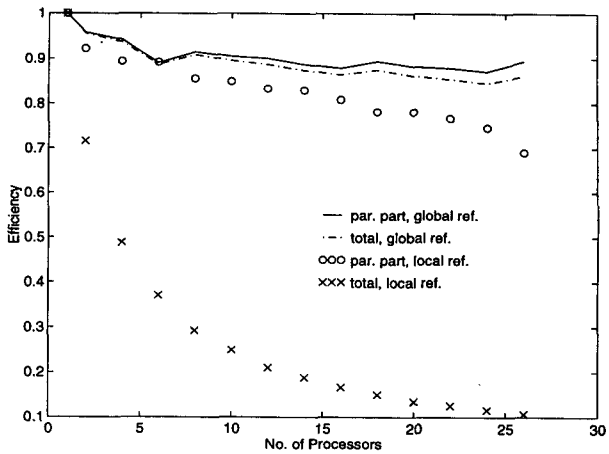

FIG. 6. Same as in Fig. 5 but efficiency is shown.

Acknowledgments. The author thanks Andrew Staniforth for helpful hints and discussions.

REFERENCES

Akima, H., 1984: On estimating partial derivatives for bivariate interpolation of scattered data. *Rocky Mt. J. Math.*, **14,** 41–52.

Behrens, J., 1995: Parallelization strategies for matrix assembly in finite element methods. *Parallel Algorithms for Irregular Problems: State of the Art,* A. Ferreira and J. D. P. Rolim, Eds., Kluwer Academic Publishers, 3–24.

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

Ciarlet, P. G., 1978: *The Finite Element Method for Elliptic Problems.* North-Holland, 530 pp.

Côté, J., M. Roch, A. Staniforth, and L. Fillion, 1993: A variable-resolution semi-Lagrangian finite-element global model of the shallow-water equations. *Mon. Wea. Rev.*, **121,** 231–243.

Globisch, G., 1995: PARAMESH, a parallel mesh generator. *Parallel Computing,* **21,** 509–524.

Hiller, W., and J. Behrens, 1991: Parallelisierung von Mehrgitteralgorithmen auf der Alliant FX/80 (Parallelization of multigrid algorithms on the Alliant FX/80). *Parallelisierung Komplexer Probleme* (Parallelization of complex problems), H. W. Meuer, Ed., Springer-Verlag, 37–82.

Hockney, R. W., and C. R. Jesshope, 1988: *Parallel Computers 2, Architecture, Programming and Algorithms.* 2d ed. Adam Hilger, 625 pp.

Ho-Le, K., 1988: Finite element mesh generation methods: A review and classification. *Comput. Aided Des.*, **20,** 27–38.

Johnson, C., 1990: *Numerical Solution of Partial Differential Equations by the Finite Element Method.* Cambridge University Press, 279 pp.

Kendall Square Research Corporation, 1992: Kendall Square Research, Technical Summary, 82 pp. [Available from Kendall Square Research Corp., Waltham, MA 02205.]

Klucewicz, I. M., 1978: A piecewise C^1 interpolant to arbitrarily spaced data. *Comp. Graph. Image Proc.*, **8,** 92–112.

Le Roux, D. Y., C. A. Lin, and A. Staniforth, 1995: An accurate interpolating scheme for semi-Lagrangian advection on an un-

structured mesh for ocean modelling. Report 95-11, 44 pp. [Available from Center for Climate and Global Change Research, McGill University, Montréal, PQ H3A 2K6, Canada.]

McDonald, A., 1984: Accuracy of multiply-upstream, semi-Lagrangian advective schemes. *Mon. Wea. Rev.,* **112,** 1264–1275.

Mitchell, W. F., 1989: A comparison of adaptive refinement techniques for elliptic problems. *ACM Trans. Math. Softw.,* **15,** 326–347.

Montefusco, L. B., and G. Casciola, 1989: Algorithm 677: C^1 surface interpolation. *ACM Trans. Math. Softw.,* **15,** 365–374.

Priestley, A., 1993: A quasi-conservative version of the semi-Lagrangian advection scheme. *Mon. Wea. Rev.,* **121,** 621–629.

Renka, R. J., 1984: Algorithm 624: Triangulation and interpolation at arbitrarily distributed points in the plane. *ACM Trans. Math. Softw.,* **10,** 440–442.

Shostko, A., and R. Löhner, 1995: Three-dimensional parallel unstructured grid generation. *Int. J. Numer. Meth. Eng.,* **38,** 905–925.

Simon, H. D., 1992: *Parallel Computational Fluid Dynamics—Implementations and Results.* The MIT Press, 342 pp.

Staniforth, A., and and C. Temperton, 1986: Semi-implicit semi-Lagrangian integration schemes for a barotropic finite-element regional model. *Mon. Wea. Rev.,* **114,** 2078–2090.

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

Tanguay, M., A. Simard, and A. Staniforth, 1989: A three-dimensional semi-Lagrangian scheme for the Canadian regional finite-element forecast model. *Mon. Wea. Rev.,* **117,** 1861–1871.

Verführth, R., 1993: A posteriori error estimators and adaptive mesh-refinement techniques for the Navier–Stokes equations. *Incompressible Computational Fluid Dynamics Trends and Advances,* M. D. Gunzenburger and R. A. Nicolaides, Eds., Cambridge University Press, 447–475.

Wait, R., and A. R. Mitchell, 1985: *Finite Element Analysis and Applications.* Wiley & Sons, 260 pp.

Zalesak, S. T., 1979: Fully multidimensional flux-corrected transport algorithms for fluids. *J. Comput. Phys.,* **31,** 335–362.