# A conservative scheme for 2D and 3D adaptive semi-Lagrangian advection

Jörn Behrens and Lars Mentrup

Abstract. This article describes a 2D and 3D adaptive and mass conserving semi-Lagrangian advection scheme for atmospheric transport problems. From the integral form of the conservation law we derive a semi-Lagrangian scheme based on conservation of mass along trajectories. The mapping of mass from the old (adaptively refined and possibly different) grid to the upstream control volume is performed by a mass packet based scheme, essentially consisting of a sub-grid discretization. We validate the new adaptive and conservative semi-Lagrangian scheme with four different analytic test cases.

## 1. Introduction

Adaptive mesh refinement has not yet gained general acceptance in the atmospheric modeling community. Despite several and even early approaches [**11, 7**], there is still resistance to include adaptive methods into the repertoire of atmospheric models. Two of the main obstacles are

(1) the algorithmic complexity of adaptive computational code, and
(2) the difficulty to implement adaptive **and** conservative operators.

While the first topic can be tackled by advanced coding techniques and the use of recently available software libraries, briefly described in this section, the following sections of this paper contribute a viable solution for the second problem.

To make the description of our methods more concrete, let us consider the simple linear advection equation (given in flux form, here)

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0 \quad \text{on } \bar{\Omega}, \tag{1.1}$$

where the computational domain is defined by $(\mathbf{x}, t) \in \bar{\Omega} = \Omega \times \mathbb{I}$, $\Omega \subset \mathbb{R}^d$, $d = 2, 3$, the spatial domain, $\mathbb{I} \subset [0, \infty[$ the time interval, and $\mathbf{v} = \mathbf{v}(\mathbf{x}, t) \in \mathbb{R}^d$ is a given multi-dimensional flow. The Lagrangian formulation of the homogeneous advection

---

1991 *Mathematics Subject Classification.* Primary 65M50, 86A10.

*Key words and phrases.* Adaptive mesh refinement, atmospheric modeling, semi-Lagrangian advection, mass conservation.

Supported by the German Climate Research Program DEKLIM (Nr. 01 LD 0037) and Deutsche Forschungsgemeinschaft DFG (Nr. BE2314/3-1).

equation, assuming a divergence-free wind $\mathbf{v}$, is given by:

$$\frac{d\rho}{dt} = 0 \quad \text{on } \bar{\Omega}, \tag{1.2}$$

where $\frac{d\rho}{dt} = \frac{\partial\rho}{\partial t} + \mathbf{v} \cdot \nabla\rho$ is the material (or total) derivative. For being well defined, (1.1) and (1.2) need (inflow) boundary conditions $\rho(\mathbf{x}_b, t) = \rho_b(\mathbf{x}_b, t)$ for $\mathbf{x}_b \in \Gamma_{\text{in}} = \partial\Omega_{\text{in}}$ the inflow boundary, and initial conditions $\rho(\mathbf{x}, t = 0) = \rho_0(\mathbf{x})$ for $\mathbf{x} \in \Omega$, suitably chosen.

In this article we will use a semi-Lagrangian discretization scheme, briefly introduced in section 2. This discretization method is unconditionally stable and very well suited for adaptive mesh refinement. While sections 2 and 3 describe the numerical details of the advection algorithms independently of grid adaptation, some remarks on refinement and mesh management have to be made here.

Mesh refinement is a non-trivial task in its own. The library supporting adaptive mesh refinement used in our computational tests is *amatos* [**3**]. *amatos* can be downloaded from an internet site and is open source[1]. It provides a triangular mesh handling and creation algorithm based on triangle bisection in 2D and 3D.

The philosophy of *amatos* is to provide a container for data. In order to perform numerical calculations, a gather operation has to take place in order to gain vectorizable data structures for the numerical calculation. By this, the mesh handling is technically decoupled from numerics, yielding optimal data layouts for both parts of an adaptive computation: mesh handling and numerical calculation.

*amatos* provides methods for multivariate scattered data interpolation with either spline techniques or radial basis function techniques. Radial basis functions are also used to calculate smooth approximations to derivatives (gradients) of gridded values. Additionally, computational geometry algorithms support boundary calculations and mass conservative semi-Lagrangian time integration methods.

In order to perform calculations on high performance computing equipment, *amatos* is parallelized. The refinement strategy in *amatos* follows the bisection algorithms given by Rivara and Bänsch [**1, 10**]. This refinement induces a powerful grid partitioning and node ordering scheme, namely a space-filling curve approach which helps to establish a distribution of mesh cells to multiple processors [**3, 5**].

## 2. Adaptive Semi-Lagrangian Method

In this section we give a brief overview of the adaptive semi-Lagrangian scheme, introduced in [**2**]. We start with describing the non-adaptive semi-Lagrangian method (SLM), introduced in [**14**]. A good overview of the different flavors of the SLM can be found in [**13**]

We first consider equation (1.2) and use a second order time-centered discretization given by

$$\frac{\rho(\mathbf{x}, t + \Delta t) - \rho(\mathbf{x} - 2\alpha(\mathbf{x}), t - \Delta t)}{2\Delta t} = 0 \quad \text{on} \quad \bar{\Omega}, \tag{2.1}$$

where $\Delta t$ is the discrete time step and $\alpha(\mathbf{x})$ denotes the path to the upstream position corresponding to $\mathbf{x}$. $\alpha(\mathbf{x})$ is given by a simple ordinary differential equation (ODE), namely

$$\dot{\mathbf{x}} = \frac{d\mathbf{x}}{dt} = \mathbf{v}(\mathbf{x}, t), \tag{2.2}$$

[1]See URL `http://www-m3.ma.tum.de/m3/software/amatos`

where $\mathbf{v} = \mathbf{v}(\mathbf{x}, t)$ is the given velocity field as in (1.1). Then, $2\alpha(\mathbf{x}) = \mathbf{x}(t + \Delta t) - \mathbf{x}(t - \Delta t)$, where $\mathbf{x}(t)$ is a particle's position at time $t$. We usually denote by $\mathbf{x} = \mathbf{x}(x + \Delta t)$ the (future) grid point position. (2.2) can be solved by a suitable numerical method for ODEs. A second order method for (2.2) is given by the following fixed point iteration:

$$\alpha^{k+1}(\mathbf{x}) = \Delta t \cdot \mathbf{v}\left(\mathbf{x} - \frac{\alpha^k(\mathbf{x})}{2}, t + \frac{\Delta t}{2}\right). \tag{2.3}$$

Furthermore, (2.1) can be modified such that a double time step without loss of accuracy can be achieved (for details see [**13**]). We observe that two decoupled calculations occur, when stepping forward in time: one acting on even, the other one on odd time steps. (2.1) is then modified to

$$\frac{\rho(\mathbf{x}, t) - \rho(\mathbf{x} - \alpha(\mathbf{x}), t - \Delta t)}{\Delta t} = 0 \Longrightarrow \rho(\mathbf{x}, t) = \rho(\mathbf{x} - \alpha(\mathbf{x}), t - \Delta t). \tag{2.4}$$

We end up with three steps of the so called *two-time-level algorithm*:

Algorithm 2.1. Basic semi-Lagrangian Algorithm

1. Calculate $\alpha(\mathbf{x})$ by solving (2.2) with (2.3).
2. Interpolate upstream value $\rho(\mathbf{x} - \alpha(\mathbf{x}), t - \Delta t)$ according to (2.4).
3. Update grid value $\rho(\mathbf{x}, t)$ using (2.4).

Comparing (1.1) with (1.2), the Eulerian flux form with the Lagrangian form of the advection equation, it can easily be seen that in general no conservation properties can be guaranteed. A semi-Lagrangian method, as formulated above, capable of solving the conservation form of the equation, would have to include the additional divergence term as a right hand side source. In order to construct the mass conserving semi-Lagrangian advection scheme, we consider the integral (conservation) form of the advection equation [**6**]:

$$\frac{d}{dt}\int_{V(t)} \rho \, d\mathbf{x} = 0, \tag{2.5}$$

with $V(t)$ a reference volume moving with the flow. If we apply the semi-Lagrangian machinery to equation (2.5), we obtain

$$\int_{V(t)} \rho \, d\mathbf{x} = \int_{V(t-\Delta t)} \rho \, d\mathbf{x}, \tag{2.6}$$

where $V(t - \Delta t)$ is the upstream reference volume.

So far, we have not mentioned a mesh. In fact, this algorithm in principle works on arbitrary meshes and even in mesh-less situations [**4**]. In this study we consider adaptively refined triangular and tetrahedral meshes in 2D and 3D. A typical two-dimensional mesh is shown in figure 1.

In order to formulate the adaptive semi-Lagrangian algorithm for equation (2.4), two different meshes have to be considered, one at time $t$ which will be modified according to a suitable refinement criterion, and one fixed mesh at time $t - \Delta t$. We denote the $k$-th iterate of the mesh (grid) at time $t$ with $G^{(k)}(t)$ and the old fixed mesh with $G(t - \Delta t)$. We embed the semi-Lagrangian algorithm into an adaptive iteration with *a posteriori* refinement criterion and obtain the following algorithm for each time-step:

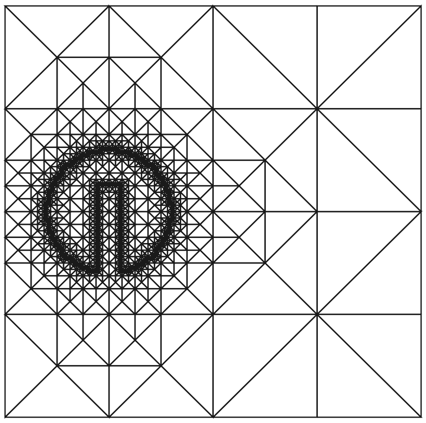

FIGURE 1. Locally adapted triangular grid for modelproblem 4.3.

ALGORITHM 2.2. Adaptive semi-Lagrangian Algorithm

(1) Duplicate mesh $G(t-\Delta t)$, obtaining an initial mesh for the adaptive iteration $G^{(1)}(t)$ (set $G^{(0)}(t)=\emptyset$).
(2) `While` $G^{(k)}(t) \neq G^{(k-1)}(t)$ $(k>0)$:
(3) Perform the semi-Lagrangian Algorithm for all $\mathbf{x} \in G^{(k)}(t) \backslash G^{(k-1)}(t)$:
  (a) Calculate $\alpha(\mathbf{x})$ using equation (2.3).
  (b) Calculate $\rho(\mathbf{x}-\alpha(\mathbf{x}), t-\Delta t)$ (e.g. by Interpolation).
  (c) Update $\rho(\mathbf{x}, t)$ according to (2.4).
(4) Estimate the local error $\eta_\tau$ for each cell $\tau$ of $G^{(k)}(t)$.
(5) Refine those cells $\tau$, where $\eta_\tau > \theta_{\text{ref}} \cdot \eta_{\max}$, with $\eta_{\max} = \max_{\tau \in G^{(k)}(t)} \eta_\tau$, and $\theta_{\text{ref}}$ a given tolerance. Coarsen the mesh analogously, to obtain $G^{(k+1)}(t)$.
(6) Set $k \leftarrow k+1$, `GOTO`: step (2).

Algorithm 2.2 is defined point-wise, where we evaluate $\rho(\mathbf{x}, t)$ at nodes. If we consider equation (2.6), we need to define associated reference volumes $V(\mathbf{x}, t)$ for each grid point. For the MPSLM described in the next section, we take the node's surrounding cells as a control volume. The scheme subdivides each cell into small mass packets which are then associated to the nodes (the scheme is a bit more sophisticated, for details see section 3).

Note that for all our experiments we do not really use an error estimator, but use the gradient of $\rho$ restricted to the element $\tau$ as an indicator for refinement: $\eta_\tau = \nabla\rho|_\tau$.

For the sake of a simple instructive description, let us choose the dual cell as control volume corresponding to $\mathbf{x}$. The dual cell is the polyhedron, formed by the dual points $\xi_i(\mathbf{x})$, $i = 1 : n(\mathbf{x})$, surrounding $\mathbf{x}$ (see figure 2). Here we take the surrounding cell's center-points as dual points.

Taking this definition of reference volume, the corresponding upstream reference volume $V(\mathbf{x}-\alpha(\mathbf{x}), t-\Delta t)$ is then formed by the polyhedron of all upstream dual points $\xi_i^- := \xi_i(\mathbf{x}) - \alpha(\xi_i(\mathbf{x}))$. It is intuitively clear that for the method

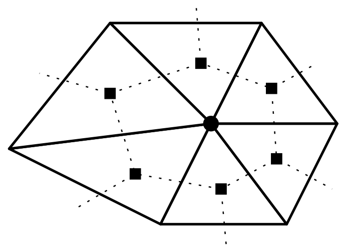

FIGURE 2. Dual nodes (■), forming a dual cell, corresponding to an original node (•).

to remain stable, upstream reference volumes must not degenerate or fold over (Lipschitz-Stability [**12**]).

Now, we are able to define mass conservation. We need two notions: global and local mass conservation.

DEFINITION 2.3. Let $E(\sigma)_{t_0}^{t_1} : \rho(\mathbf{x} - \alpha(\mathbf{x}), t_0) \to \rho(\mathbf{x}, t_1)$ the discrete evolution equation implementing a numerical scheme denoted by $\sigma$, $\mathbf{x} - \alpha(\mathbf{x})$ being the upstream position, and suppose $\Omega = \mathbb{R}^d$. Let furthermore $\mathbb{I} = [0, T]$. Then we say that the scheme $\sigma$ is *globally mass conserving*, if

$$\int_\Omega \rho(\mathbf{x}, T)\ d\mathbf{x} = \int_\Omega E(\sigma)_0^T[\rho_0(\mathbf{x})]\ d\mathbf{x} = \int_\Omega \rho_0(\mathbf{x})\ d\mathbf{x}.$$

REMARK 2.4. Note that the above definition can easily be extended to the cases where $\Omega \subsetneqq \mathbb{R}^d$, $\mathbb{I} = [t, T]$, or $T = \infty$.

DEFINITION 2.5. With the assumptions from 2.3 and denoting a volume corresponding to $(\mathbf{x}, t)$ by $V(\mathbf{x}, t)$ we say that the scheme $\sigma$ is *locally mass conserving* if

$$\begin{aligned} \int_{V(\mathbf{x},t)} \rho(\mathbf{x}, t)\ d\mathbf{x} &= \int_{V(\mathbf{x},t)} E(\sigma)_{t-\Delta t}^{t}(\rho(\mathbf{x}, t - \Delta t))\ d\mathbf{x} \\ &= \int_{V(\mathbf{x}-\alpha(\mathbf{x}),t-\Delta t)} \rho(\mathbf{x}, t - \Delta t)\ d\mathbf{x}. \end{aligned} \tag{2.7}$$

REMARK 2.6. It is easy to see that if a scheme is locally mass conserving, then it is globally mass conserving, while the converse is not necessarily true.

## 3. Description of the MPSLM

The locally mass conserving semi-Lagrangian scheme, described in this section is based on (2.7). So, it discretizes

$$\int_{V(\mathbf{x},t)} \rho(\mathbf{x}, t)\ d\mathbf{x} = \int_{V(\mathbf{x}-\alpha(\mathbf{x}),t-\Delta t)} \rho(\mathbf{x}, t - \Delta t)\ d\mathbf{x}. \tag{3.1}$$

Note that (3.1) is the discrete counterpart of (2.6).

To describe the Mass-Packet Semi-Lagrangian Method (MPSLM), we need to define mass packets. These are mass-volume units which are defined by their mass, volume and barycenter's position. In general they are much smaller than the grid cells. They are used to subdivide the cells into smaller parts. The mass packets, created in each time step, will be advected according to the semi-Lagrangian idea.

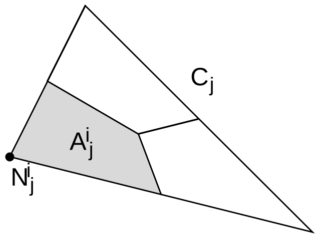


FIGURE 3. Volume $|A_j^i|$ in Cell $C_j$ is associated to node $N_j^i$.

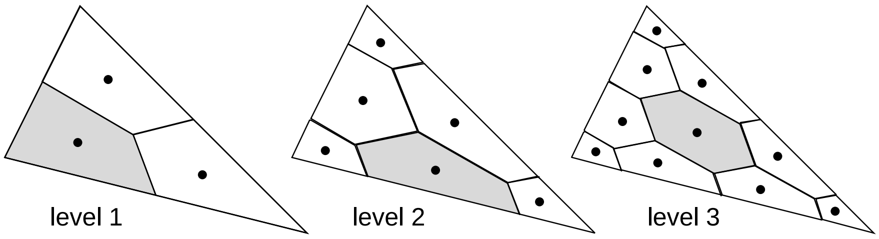


FIGURE 4. Mass packet refinement within one cell. Different mass packet types are shaded. The number of mass packets is given by $L = \sum_{i=1}^{\text{level}+1} i$. The adaptive mesh refinement requires an adaptive number of mass packets, since we require a continuous mapping of mass to the upstream cells.

In this section, we will denote an item $\nu$ in the old mesh $G(t - \Delta t)$ by $\nu^{(-)}$ while a new item at time $t$ is denoted by $\nu^{(+)}$.

First, the density $\rho^{(-)}$ must be transformed to mass values. Therefore we define a mass attached at each node $N_j^{i(-)}$ per cell $C_j^{(-)}$ by

$$m(N_j^{i(-)}) = \rho(N_j^{i(-)})|A_j^i|, \tag{3.2}$$

where we denote the volume by $|\cdot|$, $|A_j^{i(-)}|$ is the corresponding area of $C_j^{(-)}$ as depicted in figure 3.

Then the mesh cell $C_j^{(-)}$ is fully partitioned into $L$ disjoint pieces: $\sum_{l=1}^{L} |M_j^l| = |C_j^{(-)}|$ where the $M_j^l$ are the $L$ mass packets of cell $C_j^{(-)}$ (see figure 4). According to the mass packet's volume $|M_j^l|$ and barycenter's position $\lambda_i^l$ with respect to the cell's nodes $N_j^{i(-)}$, packets get mass assigned by the formula

$$m(M_j^l) = \sum_{i=1}^{d+1} \lambda_i^l m(N_j^{i(-)})|M_j^l| \tag{3.3}$$

Note that the barycentric coordinates have the property that $\sum_l \lambda^l = 1$. Thus, keeping in mind the disjoint partition of cells in mass packets, summing over all mass packets in a cell, we obtain the exact mass assigned to each node. By this means the mass in mesh $G(t - \Delta t)$ is virtually transformed into mass packets.

Generally, mass packets are only created, if a cell carries mass. The sub-refinement of a cell into mass packets is driven by the refinement level of the cell

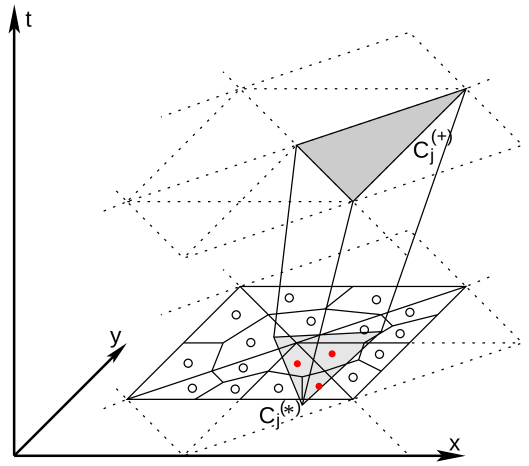


FIGURE 5. Mapping of mass packets to the upstream (new) cell $C_j^{(*)}$ corresponding to the downstream new cell $C_j^{(+)}$. The red/solid mass packets are associated to $C_j^{(*)}$ and therefore to $C_j^{(+)}$.

and the density concentration. However, there is no rigorous criterion for sub-refinement. Empirically a sub-refinement of 7 levels has been shown to be satisfactory, and is used in the numerical tests in section 4. Clearly, the adaptive mesh refinement strategy is driven by an error criterion and is used to improve the representation of the PDE solution, while increasing the sub-refinement improves the mapping of mass, thereby reducing numerical diffusion. A high level of sub-refinement yields an improved representation of the integrals in equation (2.6), or more precisely an improved representation of the upstream control volume on the right hand side.

For each node of the new grid the upstream position is calculated following the plain SLM (2.4). Knowing the upstream grid cells and the mass packets on the old grid, a mapping step follows. The mapping consists of assigning each upstream (new) cell $C_j^{(*)}$ the corresponding (old) mass packets (see figure 5). A mass packet is assigned to cell $C_j^{(+)}$ if its barycenter is inside the corresponding upstream cell $C_j^{(*)}$. It should be noted that this search algorithm represents the most time consuming part of the whole scheme. Then the barycentric coordinates $\mu_i^l$ with respect to the (new) nodes $N_j^{i(*)}$ are calculated. Finally, the nodes $N_j^{i(+)}$ get mass assigned corresponding to the mass packets found in cell $C_j^{(*)}$:

$$m(N_j^{i(+)}) = \sum_{l:M_j^l \in C_j^{(*)}} \mu_i^l m(M_j^l) \tag{3.4}$$

As for the $\lambda$'s it holds that $\sum_l \mu^l = 1$. Therefore, the mass is reassigned to the nodes without loss, except for mass packets lying outside the upstream (new) mesh.

Now that we know the mass at the (new) nodes, we can compute the density value

$$\rho(N_j^{i(+)}) = \frac{m(N_j^{i(+)})}{|V^{i(+)}|} \tag{3.5}$$

where $V^{i(+)}$ is the control volume associated to node $N^{i(+)}$ composed of all $A_j^i$ in the patch of the node. This scheme guarantees that $\rho$ is non-negative. Details of the implementation and theoretical proofs of the scheme's conservation properties can be found in [**9**]. Summing up, the MPSLM algorithm is given by

Algorithm 3.1. Mass Packet semi-Lagrangian Algorithm

(1) Calculate nodal masses $m(N_j^{i(-)})$ using (3.2).
(2) Subdivide cells into mass packets and assign masses according to (3.3).
(3) Calculate upstream cells $C_j^{(*)}$ and map mass packets to $C_j^{(*)}$.
(4) Calculate new nodal masses using (3.4).
(5) Reconstruct nodal densities with (3.5).

## 4. Results

In order to test the proposed numerical method, we establish four different test cases for advection of a density distribution function. A detailed description of test cases follows. We will give computational times along with the results of the test cases. A fifth test case with a smooth density distribution is used for convergence tests.

**4.1. Description of Test Cases.** In order to be properly defined, equation (2.5) needs additional specifications for the computational domain $\bar{\Omega}$, for the boundary conditions $\rho_b(\mathbf{x}, t)$, for the initial condition $\rho_0(\mathbf{x})$, and the wind $\mathbf{v}(\mathbf{x}, t)$. For all 2D tests we will assume

$$\bar{\Omega} = \bar{\Omega}_{\mathrm{2D}} = \{(\mathbf{x}, t) :\ \mathbf{x} \in [-0.5, 0.5]^2;\ t \in [0, T]\},$$

where $T$ is specified in the test case. For the 3D tests, we assume

$$\bar{\Omega} = \bar{\Omega}_{\mathrm{3D}} = \{(\mathbf{x}, t) :\ \mathbf{x} \in [0, 1]^3;\ t \in [0, T]\}.$$

Additionally, in 3D we will denote with $\varphi_H$ the horizontal component of a vector-valued entity $\varphi \in \mathbb{R}^3$. Thus, if $\varphi = (\varphi_x, \varphi_y, \varphi_z)^T$ then $\varphi_H = (\varphi_x, \varphi_y)^T$. In the computations we use a uniform time step $\Delta t$ of $1,800\ s$.

We will assume all lateral boundaries to be either outflow boundaries or trivial inflow boundaries (depending on the wind field specified). That means, mass can escape from the computational domain, but can not enter. It remains to be specified the initial condition, the wind, and the time interval.

Example 4.1. *Diagonal Wind (2D)*

$$\begin{aligned}
\rho_0(\mathbf{x}, t) &= \begin{cases} 1, & \text{in } \{\mathbf{x} : |\mathbf{x} - (-0.25, -0.25)^T| < 0.15\}, \\ 0, & \text{else.} \end{cases} \\
\mathbf{v}(\mathbf{x}, t) &= (1, 1)^T \cdot s,\ \text{with a scaling factor } s = 0.36361^{-5} \\
T &= 10 \text{ time steps}
\end{aligned}$$

*Diagonal Wind (3D)*

$$\begin{aligned}
\rho_0(\mathbf{x},t) &= \begin{cases} 1, & \text{in } \{\mathbf{x} : |\mathbf{x}-(0.25,0.25,0.5)^T| < 0.15\}, \\ 0, & \text{else.} \end{cases} \\
\mathbf{v}_H(\mathbf{x},t) &= (1,1)^T \cdot s, \text{ with a scaling factor } s = 0.36361^{-5} \\
\mathbf{v}_z(\mathbf{x},t) &= 0 \\
T &= 10 \text{ time steps}
\end{aligned}$$

EXAMPLE 4.2. *Converging Wind (2D)*

$$\begin{aligned}
\rho_0(\mathbf{x},t) &= \begin{cases} 1, & \text{in } \{\mathbf{x} : |\mathbf{x}-(-0.25,0.0)^T| < 0.15\}, \\ 0, & \text{else.} \end{cases} \\
\mathbf{v}(\mathbf{x},t) &= ((0.75,0)^T - \mathbf{x}^T)\cdot\frac{\omega}{2}, \text{ with } \omega = 0.36361^{-4} \\
T &= 10 \text{ time steps}
\end{aligned}$$

*Converging Wind (3D)*

$$\begin{aligned}
\rho_0(\mathbf{x},t) &= \begin{cases} 1, & \text{in } \{\mathbf{x} : |\mathbf{x}-(0.25,0.5,0.5)^T| < 0.15\}, \\ 0, & \text{else.} \end{cases} \\
\mathbf{v}(\mathbf{x},t) &= ((1.25,0.5,0.5)^T - \mathbf{x}^T)\cdot\frac{\omega}{2}, \text{ with } \omega = 0.36361^{-4} \\
T &= 10 \text{ time steps}
\end{aligned}$$

Note that for the preceding two test cases, the center of the profile in the last step is at $(-0.185,-0.185)$ and $(0.03,0.0)$ respectively. The following *slotted cylinder test case* has been adopted from a test case originally proposed in [**15**]. Note that the angular velocity factor $\omega$ is chosen such that 96 time steps are exactly one revolution.

EXAMPLE 4.3. *Slotted Cylinder Test Case (2D)*

$$\begin{aligned}
\rho_0(\mathbf{x},t) &= \begin{cases} 1, & \text{in } \{\mathbf{x} : |\mathbf{x}-(-0.25,0.0)^T| < 0.15, \\ & \quad\text{and } \mathbf{x} \notin [-0.28,-0.22]\times[-0.5,0]\}, \\ 0, & \text{else.} \end{cases} \\
\mathbf{v}(\mathbf{x},t) &= ((-y,x)^T\cdot\omega, \text{ with } \omega = 0.36361^{-4} \\
T &= 96 \text{ time steps}
\end{aligned}$$

*Slotted Cylinder Test Case (3D)*

$$\begin{aligned}
\rho_0(\mathbf{x},t) &= \begin{cases} 1, & \text{in } \{\mathbf{x} : |\mathbf{x}-(0.25,0.5,0.5)^T| < 0.15, \\ & \quad\text{and } \mathbf{x} \notin [0.22,0.28]\times[0.5,1.0]\times[0,1.0]\}, \\ 0, & \text{else.} \end{cases} \\
\mathbf{v}_H(\mathbf{x},t) &= ((-y,x)^T\cdot\omega, \text{ with } \omega = 0.36361^{-4} \\
\mathbf{v}_z(\mathbf{x},t) &= 0 \\
T &= 96 \text{ time steps}
\end{aligned}$$

The *accelerating wind* test case has been adopted and extended to 3D from [**8**]. Additionally, the scaling has been adjusted such that after 72 time steps a full revolution is complete, and analytic error evaluation can be performed.

EXAMPLE 4.4. *Accelerating Wind (2D)*

$$\rho_0(\mathbf{x},t) = \begin{cases} 1, & \text{in } \{\mathbf{x} : |\mathbf{x} - (-0.25, 0.0)^T| < 0.15, \\ & \text{and } \mathbf{x} \notin [-0.28, -0.22] \times [-0.5, 0]\}, \\ 0, & \text{else.} \end{cases}$$

$$\mathbf{v}(\mathbf{x},t) = \begin{cases} (-y,x)^T \cdot \omega, & \text{if } = (x,y) \in \{\mathbf{x} : x \leq 0\}, \\ (-y,x)^T \cdot \omega(1.5\cos(2\phi) + 2.5), & \text{if } = (x,y) \in \{\mathbf{x} : x > 0, y > 0\}, \\ (-y,x)^T \cdot \omega(1.5\cos(2\psi) + 2.5), & \text{if } = (x,y) \in \{\mathbf{x} : x > 0, y \leq 0\}, \end{cases}$$

$$\text{with } \omega = 0.36361^{-4},\ \phi = \arctan(\frac{y}{x}),\ \psi = \arctan(\frac{-y}{x}).$$

$$T = 72 \text{ time steps}$$

*Accelerating Wind (3D)*

$$\rho_0(\mathbf{x},t) = \begin{cases} 1, & \text{in } \{\mathbf{x} : |\mathbf{x} - (0.25, 0.5, 0.5)^T| < 0.15, \\ & \text{and } \mathbf{x} \notin [0.22, 0.28] \times [0.5, 1.0] \times [0, 1.0]\}, \\ 0, & \text{else.} \end{cases}$$

$$\mathbf{v}_H(\mathbf{x},t) = \begin{cases} (-y,x)^T \cdot \omega, & \text{if } = (x,y) \in \{\mathbf{x} : x \leq 0\}, \\ (-y,x)^T \cdot \omega(1.5\cos(2\phi) + 2.5), & \text{if } = (x,y) \in \{\mathbf{x} : x > 0, y > 0\}, \\ (-y,x)^T \cdot \omega(1.5\cos(2\psi) + 2.5), & \text{if } = (x,y) \in \{\mathbf{x} : x > 0, y \leq 0\}, \end{cases}$$

$$\text{with } \omega = 0.36361^{-4},\ \phi = \arctan(\frac{y}{x}),\ \psi = \arctan(\frac{-y}{x}).$$

$$\mathbf{v}_z(\mathbf{x},t) = 0$$

$$T = 72 \text{ time steps}$$

Note that since we deal with linear advection, all the previous test cases can be solved analytically and, therefore, are very well suited for accuracy investigation. However, convergence tests need a smooth density distribution function, since even high order schemes usually do not achieve high order convergence near discontinuities in the data. Therefore, a fifth test case is given for convergence tests

EXAMPLE 4.5. *Convergence test case (2D)*

$$\rho_0(\mathbf{x},t) = \begin{cases} 4\cos\left(\frac{r\pi}{2R}\right), & \text{in } \{\mathbf{x} : r < R\} \\ & \text{with } r = |\mathbf{x} - (-0.25, 0.0)^T|,\ R = 0.15; \\ 0, & \text{else.} \end{cases}$$

$$\mathbf{v}(\mathbf{x},t) = (-y,x)^T \cdot \omega,\ \text{with } \omega = 0.36361^{-4}$$

$$T = 24 \text{ time steps}$$

*Convergence test case (3D)*

$$\rho_0(\mathbf{x},t) = \begin{cases} \cos\left(\frac{r\pi}{2R}\right), & \text{in } \{\mathbf{x} : r < R\} \\ & \text{with } r = |\mathbf{x} - (0.25, 0.5, 0.5)^T|,\ R = 0.15; \\ 0, & \text{else.} \end{cases}$$

$$\mathbf{v}_H(\mathbf{x},t) = (-y,x)^T \cdot \omega,\ \text{with } \omega = 0.36361^{-4}$$

$$\mathbf{v}_z(\mathbf{x},t) = 0$$

$$T = 24 \text{ time steps}$$

**4.2. Numerical Results.** For demonstration of the capability of adaptive grid refinement to achieve accuracy while maintaining low computational cost, we compare one uniform and one adaptive configuration each in 2D and 3D. In the

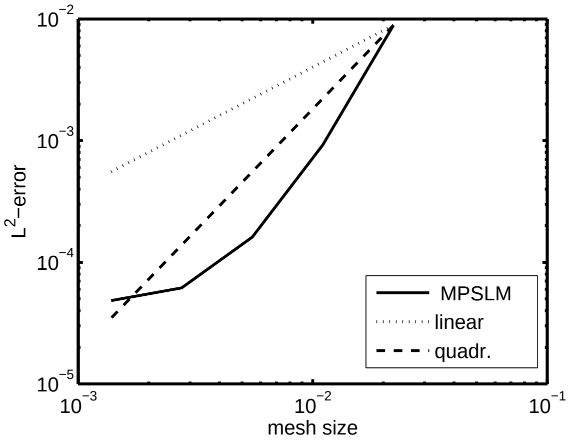


FIGURE 6. Convergence of the 2D MPSLM. The $L^2$-error is plotted against the number of unknowns in a logarithmic scale.

TABLE 1. Convergence of the 3D MPSLM

| mesh type | mesh size | $L^2$-Error |
|---|---|---|
| $[12, 12]$ | 0.06250 | $1.207 \cdot 10^{-3}$ |
| $[15, 15]$ | 0.03125 | $3.717 \cdot 10^{-4}$ |

2D case we compare the results and computational cost for an adaptive grid with 6 global (and uniform) refinement levels (coarse mesh) and additional local refinement up to level 17 (we call this a $[6, 17]$-mesh) versus a uniformly refined mesh of 17 levels (a $[17, 17]$-mesh). While the adaptive mesh is composed of between 1,709 and 9,255 unknowns, the uniform mesh has 131,585 unknowns. In 3D we compare a $[8, 15]$-mesh with 1,123 to 2,368 nodes versus a $[15, 15]$-mesh with 22,065 nodes. Note that for the 2D mesh the minimum edge length is $2.76 \cdot 10^{-3}$ while for the 3D mesh it is $3.13 \cdot 10^{-2}$ units.

To prove the convergence of the proposed MPSLM scheme experimentally a series of uniform mesh refinements are performed with test case 4.5. See figure 6 for the 2D result of the convergence test. It shows a slightly decaying but most importantly monotone convergence rate. We have no analytical results on the order of convergence of the MPSLM, the experimental results suggest an order 1.6 for the 3D case and between 1.2 and 2.8 for the 2D case. The time step for all experiments is fixed to 1800 s, which results in Courant numbers between 1.5 (for a $[\cdot, 11]$-mesh) and 32 (for a $[\cdot, 19]$-mesh). For the 3D case due to limited computing resources we give the convergence results in table 1.

For the 2D test cases 4.1-4.4, results for the $L^2$-error, the total mass, and the computational requirements on one processor of a Sun Fire V880 (1.2 GHz) are given in table 2. The corresponding results for the 3D test cases can be found in table 3. Additionally, plots of the difference of the analytical solution and the MPSLM solution are given in figures 7 to 10 for the 2D cases.

In many applications it is important that no spurious undershoots and overshoots occur. Therefore, we supply minimum and maximum values for the 2D test

Table 2. Results of 2D test cases for the MPSLM

| test | $L^2$-error | | relative mass | | time [s] | |
|---|---|---|---|---|---|---|
| no. | $[17, 17]$ | $[6, 17]$ | $[17, 17]$ | $[6, 17]$ | $[17, 17]$ | $[6, 17]$ |
| 4.1 | $6.11 \cdot 10^{-4}$ | $1.32 \cdot 10^{-3}$ | 1.000 | 1.000 | 631 | 37 |
| 4.2 | $1.82 \cdot 10^{-3}$ | $3.07 \cdot 10^{-3}$ | 1.000 | 1.000 | 649 | 38 |
| 4.3 | $4.07 \cdot 10^{-2}$ | $4.21 \cdot 10^{-2}$ | 1.000 | 1.000 | 35,720 | 2,108 |
| 4.4 | $2.13 \cdot 10^{-3}$ | $2.23 \cdot 10^{-3}$ | 1.000 | 1.000 | 6,267 | 1,356 |

Table 3. Results of 3D test cases for the MPSLM

| test | $L^2$-error | | relative mass | | time [s] | |
|---|---|---|---|---|---|---|
| no. | $[15, 15]$ | $[9, 15]$ | $[15, 15]$ | $[9, 15]$ | $[15, 15]$ | $[9, 15]$ |
| 4.1 | $2.26 \cdot 10^{-3}$ | $2.26 \cdot 10^{-3}$ | 1.000 | 1.000 | 920 | 206 |
| 4.2 | $7.66 \cdot 10^{-3}$ | $7.55 \cdot 10^{-3}$ | 1.000 | 1.000 | 1,019 | 281 |
| 4.3 | $5.94 \cdot 10^{-3}$ | $5.88 \cdot 10^{-3}$ | 0.982 | 0.966 | 10,530 | 5,103 |
| 4.4 | $5.06 \cdot 10^{-3}$ | $5.03 \cdot 10^{-3}$ | 0.990 | 0.973 | 7,633 | 3,923 |

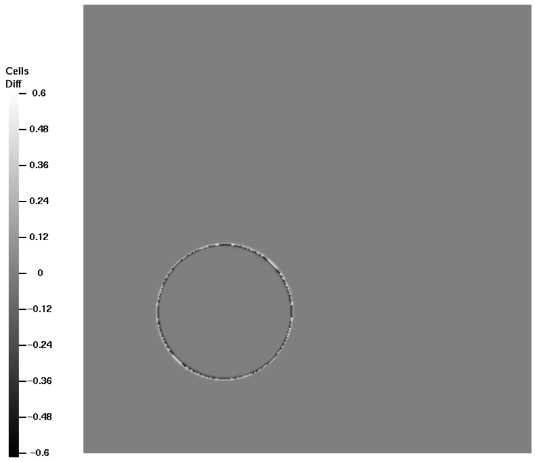


Figure 7. Difference plot of analytic vs. computed solution for test case 4.1 on a $[17, 17]$ mesh.

cases in table 4. Note that there is no undershooting. However, there is some severe overshooting. This can be explained by too little mass packets per element. If a large element with only a few mass packets is mapped to many small elements, then some of the small elements obtain too much mass. This effect can be reduced by a higher number of mass packets per cell. For example, increasing the number of sub-refinement levels to a maximum of 26, yields maximum values of 1.142 and 2.157 for test cases 4.1 and 4.2 respectively with the $[6, 17]$ adaptive grid simulation. However, the computational cost is also increasing.

In general, the $L^2$-error for the adaptive case and the uniform grid case resemble each other very well. Mass conservation can be observed in all cases. Note that in the 3D cases, the grid resolution is not high enough to gain exact mass conservation. The proposed MPSLM scheme still exhibits some numerical diffusion. Therefore,

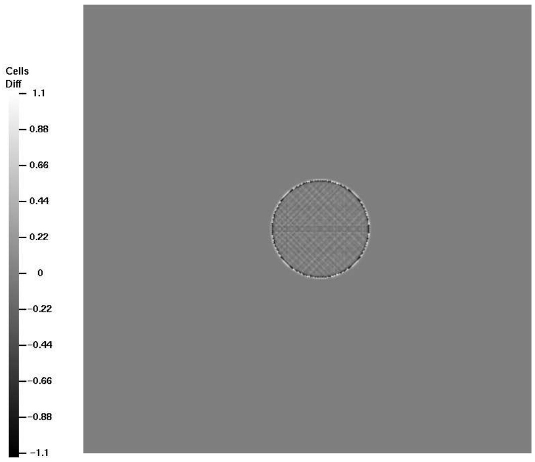


FIGURE 8. Same as figure 7 for test case 4.2.

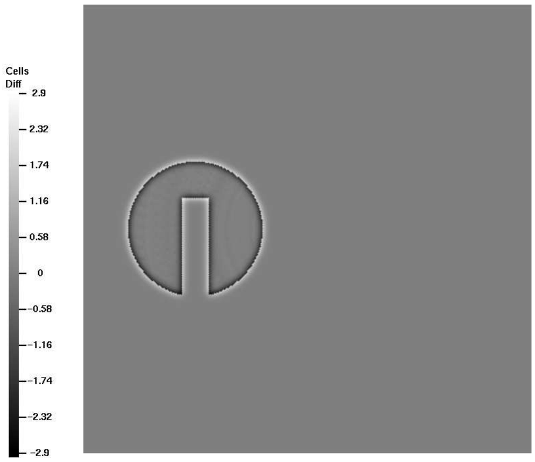


FIGURE 9. Same as figure 7 for test case 4.3.

TABLE 4. Minima and maxima of 2D test cases for the MPSLM

| test | minimum | | | maximum | | |
|---|---|---|---|---|---|---|
| no. | analytic | $[17, 17]$ | $[6, 17]$ | analytic | $[17, 17]$ | $[6, 17]$ |
| 4.1 | 0.000 | 0.000 | 0.000 | 1.000 | 1.004 | 1.801 |
| 4.2 | 0.000 | 0.000 | 0.000 | 1.924 | 2.251 | 3.359 |
| 4.3 | 0.000 | 0.000 | 0.000 | 4.000 | 4.260 | 4.734 |
| 4.4 | 0.000 | 0.000 | 0.000 | 1.000 | 1.068 | 1.207 |

mass is lost over the domain boundaries (remember that all boundaries are declared outflow boundaries). This behavior can be seen in figure 11: During the first 20 or so steps, the algorithm conserves mass exactly. Then, the slotted cylinder profile approaches the boundary at a quarter revolution, causing material that has been

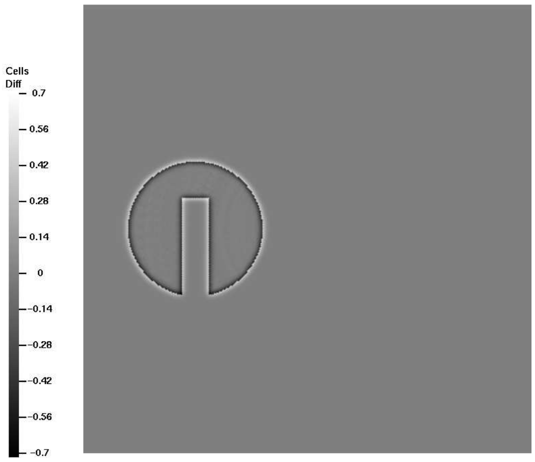


FIGURE 10. Same as figure 7 for test case 4.4.

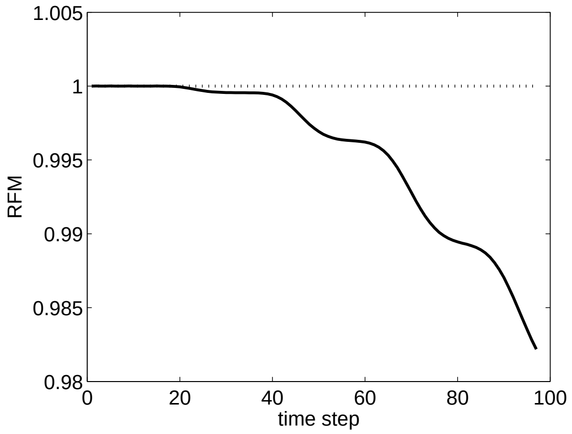


FIGURE 11. Evolution of mass with time for the 3D test case 4.3. Mass is lost due to numerical diffusion.

diffused to leave the domain. While the profile leaves the vicinity of the boundary, the mass is conserved again (the first plateau), after half a revolution the diffusive processes again take action.

Finally, observing the compute times, it can be stated that the adaptive mesh refinement strategy gains a factor of 2 to 15, depending on the case, in computational efficiency without loss of accuracy. Furthermore, the required memory for storing the unknowns decreases by a factor of 9 to 75. Note that in the 3D case, the coarse and fine levels are not very far apart, thus leading to less speedup. For higher resolution and a larger ratio of coarse-to-fine level an even larger speedup can be expected.

## 5. Conclusions

This article introduces a novel adaptive semi-Lagrangian and mass conserving advection scheme, suitable for 2D and 3D applications. The method is tested utilizing several test cases that exhibit different characteristics of real life applications, like non-grid-aligned flow (test case 4.1), converging flow fields (test case 4.2), and shear flow (test case 4.4). The proposed scheme is converging rapidly and proves to be mass conserving. The adaptive mesh refinement is efficient compared to the uniform grid case and is similarly accurate.

It is planned to compare the scheme with other mass conserving semi-Lagrangian schemes, suitable for adaptive mesh refinement in a future article. We plan to use the scheme in a project funded by the German Climate Research Program *DEKLIM*.

## References

1. E. Bänsch, *An adaptive finite-element strategy for the three-dimensional time-dependent Navier-Stokes equations*, J. Comput. App. Math. **36** (1991), 3–28.
2. J. Behrens, *An adaptive semi-Lagrangian advection scheme and its parallelization*, Mon. Wea. Rev. **124** (1996), no. 10, 2386–2395.
3. ______, *Adaptive mesh generator for atmospheric and oceanic simulations – amatos*, Tech. Report TUM-M0409, Technische Universität München, Zentrum Mathematik, Boltzmannstr. 3, 85747 Garching, 2004, URL: http://www-lit.ma.tum.de/veroeff/html/040.65008.html.
4. J. Behrens and A. Iske, *Grid-free adaptive semi-Lagrangian advection using radial basis functions*, Comp. Math. Appl. **43** (2002), 319–327.
5. J. Behrens, N. Rakowsky, W. Hiller, D. Handorf, M. Läuter, J. Päpke, and K. Dethloff, *amatos: Parallel adaptive mesh generator for atmospheric and oceanic simulation*, Ocean Modelling, in press `doi:10.1016/j.ocemod.2004.06.003`, 2004.
6. A. J. Chorin and J. E. Marsden, *A mathematical introduction to fluid mechanics*, 3 ed., Springer-Verlag, New York, 1993.
7. G. S. Dietachmayer and K. K. Droegemeier, *Application of continuous dynamic grid adaptation techniques to meteorological modeling. Part i: Basic formulation and accuracy*, Mon. Wea. Rev. **120** (1992), 1675–1706.
8. A. Iske and M. Käser, *Conservative semi-Lagrangian advection on adaptive unstructured meshes*, Tech. Report TUM-M0207, TU München, München, 2003.
9. L. Mentrup, *Development of a mass-conserving semi-Lagrangian method for the simulation of tracer transport in the atmosphere on an adaptive three-dimensional grid*, Diplomathesis in German, 2003.
10. M. C. Rivara, *Algorithms for refining triangular grids suitable for adaptive and multigrid techniques*, Internat. J. Numer. Methods Engrg. **20** (1984), 745–756.
11. W. C. Skamarock and J. B. Klemp, *Adaptive grid refinement for two-dimensional and three-dimensional nonhydrostatic atmospheric flow*, Mon. Wea. Rev. **121** (1993), 788–804.
12. P. K. Smolarkiewicz and J. A. Pudykiewicz, *A class of semi-Lagrangian approximation for fluids*, J. Atmos. Sci. **49** (1992), 2082–2096.
13. A. Staniforth and J. Côté, *Semi-Lagrangian integration schemes for atmospheric models - a review*, Mon. Wea. Rev. **119** (1991), 2206–2223.
14. A. Wiin-Nielsen, *On the application of trajectory methods in numerical forecasting*, Tellus **11** (1959), 180–196.
15. S. T. Zalesak, *Fully multidimensional flux-corrected transport algorithms for fluids*, J. Comput. Phys. **31** (1979), 335–362.

Technische Universität München, Zentrum Mathematik (M3), 85747 Garching, Germany

*E-mail address*: `{behrens,mentrup}@ma.tum.de`