A SEMI-LARGANGIAN METHOD ON DYNAMICALLY ADAPTED OCTREE MESHES *

KIRILL M. TEREKHOV†, KIRILL D. NIKITIN‡, MAXIM A. OLSHANSKII§, AND YURI V. VASSILEVSKI¶

Abstract. The paper develops a semi-Lagrangian method for the numerical integration of the transport equation discretized on adaptive Cartesian cubic meshes. We use dynamically adaptive graded Cartesian grids. They allows for a fast grid reconstruction in the course of numerical integration. The suggested semi-Lagrangian method uses a higher order interpolation with a limiting strategy and a back-and-forth correction of the numerical solution. The interpolation operators have compact nodal stencils. In a series of experiments with dynamically adapted meshes, we demonstrate that the method has at least second order convergence and acceptable conservation and monotonicity properties.

Key words. semi-Lagrangian method, octree meshes, adaptivity

1. Introduction. The well known higher order Eulerian methods TVD or WENO [12, 31] for the numerical integration of the transport equation require CFL time step restrictions. Semi-Lagrangian advection schemes are not limited by the CFL condition and hence are more flexible in adapting time steps to accuracy requirements. For non-uniform dynamically adapted Cartesian grids semi-Lagrangian schemes become even more attractive since the construction of high order accurate non-oscillatory Eulerian methods is more challenging in this case.

The goal of this paper is to develop a semi-Lagrangian method for octree meshes allowing dynamic adaptation with coarsening and refinement performed according to error indicators in the course of numerical integration. Octree grids enjoy a growing reliance in scientific computing community due to the simple Cartesian structure and embedded hierarchy, which makes mesh adaptation, reconstruction and data access fast and easy. Such grids were used successfully for numerical simulation of hyperbolic conservation laws in the frameworks of discontinuous Galerkin and finite volume discretizations [9, 20, 29, 33]. Fast remeshing with octree grids makes them a natural choice for the simulation of moving interfaces and free surface flows [11, 16, 17, 21, 22, 27, 32] as well as more general non-Newtonian and high-speed Newtonian flows, see, e.g., [1, 2, 18, 23, 25, 26, 36]. Semi-Lagrangian methods were previously employed for octree based simulations in [6, 16, 21–23, 25]. The simplicity and stability of semi-Lagrangian methods usually come at a price: lower order methods are known to be numerically diffusive whereas higher order methods behave non-monotonically by trading numerical dissipation for numerical dispersion. To enhance the performance of a lower order monotone semi-Lagrangian method, the authors of [6,16] introduce a particle based correction mechanism. The particle correction improves conservation properties and reduces numerical diffusion of a semi-Lagrangian method, however it may destroy certain smoothness properties of the recovered solution. Another possible approach is to use nonlinear WENO-type reconstruction [10] as

*This work has been supported by Russian Science Foundation through the grant 14-31-00434.

†Stanford University; kirill.terehov@gmail.com

‡Institute of Numerical Mathematics, Institute of Nuclear Safety, Russian Academy of Sciences, Moscow; nikitin.kira@gmail.com

§Department of Mathematics, University of Houston; molshan@math.uh.edu

¶Institute of Numerical Mathematics, Russian Academy of Sciences, Moscow Institute of Physics and Technology, Moscow; yuri.vassilevski@gmail.com

an interpolation procedure for a semi-Lagrangian method as was done for structured grids in [3, 28]. In the case of adaptive dynamic grids it will require costly matrix inversions for each interpolation step. A higher order semi-Lagrangian method on block-structured adaptive meshes (AMR) is discussed in [34]. Similar to the present paper, the method of [34] uses a higher order interpolation with a limiter. A higher order semi-Lagrangian method for the level set equation was introduced for non-graded octree meshes [19]. That method employs the triquadratic interpolation. We develop further these approaches by introducing a compact stencil tricubic interpolation, refining the interpolation limiter from [34], applying back-and-forth correction procedure [5] with the correction limiter [14] and validating the method for dynamically reconstructed grids. As a result, the method derived here demonstrates a higher order convergence on adaptive grids and acceptable conservation and monotonicity properties. This makes the method efficient for solving transport equations on dynamically adapted grids. It can be used for simulation of mass transport and recovering a free-surface evolution from the level set equation.

The rest of the paper is organized as follows. After a brief recall of basics for the semi-Lagrangian method in section 2 we introduce the necessary ingredients of this numerical technique: a numerical integrator in time (section 2.1), a back-and-forth error compensation method (section 2.2) together with a suitable limiting strategy (section 2.3). Section 3 deals with spatial discretization. First, we introduce discretizations of the second order derivatives in section 3.1. We use these finite difference derivatives further in section 3.2 to construct a higher order interpolation operator with a compact nodal stencil. The interpolation operator invokes a limiter defined in section 3.3. The gradient of a nodal grid function useful in local grid adaptation is defined in section 3.4. Section 4 collects the results of several numerical experiments. The experiments are aimed on assessing the performance of the presented semi-Lagrangian method as the numerical tool for the simulation of mass transport and level set function transport. We also compare the method with a few more standard semi-Lagrangian techniques.

2. Semi-Lagrangian method. A passive advection of a scalar field φ with a given velocity vector field $\mathbf{u}(\mathbf{x},t)$, $\mathbf{x}\in\mathbb{R}^3$, $t\in\mathbb{R}_+$, is modeled by the transport equation

$$\frac{\partial\varphi}{\partial t}+\mathbf{u}\cdot\nabla\varphi=0 \tag{2.1}$$

equipped with appropriate initial conditions.

A semi-Lagrangian method is a computational technique to solve (2.1). Within the semi-Lagrangian approach the approximate solution is found in time instances t^n, $n=0,1,2,\dots$ by numerical integration backward in time of the characteristic equations

$$\frac{d\mathbf{x}(\tau)}{d\tau}=\mathbf{u}(\mathbf{x}(\tau),\tau),\quad \mathbf{x}(t^{n+1})=\mathbf{x}_0,\quad \tau\in[t^{n+1},t^n] \tag{2.2}$$

for all $\mathbf{x}_0$ lying in a computational domain at time t^{n+1}. If $\varphi^n\approx\varphi(\cdot,t^n)$ is known everywhere, one sets $\varphi^{n+1}(\mathbf{x}_0)=\varphi^n(\mathbf{x}(t^n))$.

In practice, a spacial discretization is applied, for example by defining φ in a finite number of nodes, which form a *grid*. Since $\mathbf{x}(t^n)$ is not necessarily the grid node, an *interpolation* φ^n_I from nodal values of φ^n is done to define its value in $\mathbf{x}(t^n)$, $\varphi^{n+1}(\mathbf{x}_0)=\varphi^n_I(\mathbf{x}(t^n))$. The numerical integration of (2.2) and the interpolation error

contribute to the numerical error of a semi-Lagrangian method. This error is not monotonic with respect to time step Δt and has the form [8,35]:

$$O\left((\Delta t)^k + \frac{(\Delta x)^{p+1}}{\Delta t}\right), \tag{2.3}$$

where k refers to the order of numerical integration of (2.2), Δx is the spacial mesh step, and p is the interpolation order of φ_I^n.

The error estimate (2.3) calls for a higher order interpolation. However, standard linear higher order interpolation techniques lead to the loss of the monotonicity property, which is critical for numerical stability in many applications. Therefore, several limiting and accuracy improving techniques have been suggested in the literature to allow for accurate and stable semi-Lagrangian approach. We discuss a few of them below.

Further we also need the reverse semi-Lagranigan method given by a numerical integration forward in time:

$$\frac{d\widetilde{\mathbf{x}}(\tau)}{d\tau} = \mathbf{u}(\widetilde{\mathbf{x}}(\tau),\tau), \quad \widetilde{\mathbf{x}}(t^n) = \mathbf{x}_0, \quad \tau \in [t^n, t^{n+1}] \tag{2.4}$$

and setting $\widetilde{\varphi}^n(\mathbf{x}_0) = \widetilde{\varphi}_I^{n+1}(\widetilde{\mathbf{x}}(t^{n+1}))$. Here $\widetilde{\varphi}_I^{n+1}$ denotes a suitable interpolation of $\widetilde{\varphi}^{n+1}$ at $\widetilde{\mathbf{x}}(t^{n+1})$.

2.1. Numerical integration. In this paper, we shall use the trapezoidal rule for the numerical integration in (2.2) and (2.4). Thus, the numerical counterpart of (2.2) reads

$$\begin{aligned} \mathbf{x}(t^n + \frac{\Delta t}{2}) &= \mathbf{x}_0 - \frac{\Delta t}{2}\mathbf{u}(\mathbf{x}_0, t^n), \\ \mathbf{x}(t^n) &= \mathbf{x}_0 - \Delta t \widetilde{\mathbf{u}}^{n+\frac{1}{2}}, \end{aligned} \tag{2.5}$$

with $\Delta t = t^n - t^{n+1}$. If $\mathbf{u}(\mathbf{x},\tau)$ is known for intermediate times, then we set $\widetilde{\mathbf{u}}^{n+\frac{1}{2}} := \mathbf{u}(\mathbf{x}(t^n + \Delta t/2), t^n + \Delta t/2)$. If the velocity field is not given a priori, but recovered numerically from separate equations, then it may happen that $\mathbf{u}$ is known only at times t^k, $k = 0, \ldots, n$. In this case, the linear extrapolation is used:

$$\widetilde{\mathbf{u}}^{n+\frac{1}{2}} = (1+\eta)\mathbf{u}(\mathbf{x}(t^n + \Delta t/2), t^n) - \eta\mathbf{u}(\mathbf{x}(t^n + \Delta t/2), t^{n-1}), \quad \eta = \frac{t^{n+1} - t^n}{t^n - t^{n-1}}.$$

The same technique is applied to integrate numerically (2.4).

The method is second order in time, i.e. $k = 2$ in (2.3).

2.2. Back-and-forth error compensation. Back-and-forth error compensation and correction method from [4,5] is a well-known numerical technique to improve the accuracy of a semi-Lagrangian method without evoking higher order interpolation. This predictor-corrector type method is based on the observation that if we solve (2.1) forward in time for one time step using a numerical integrator and then backward in time for one time step with the same method, the difference between the two copies of the solution gives us information about the numerical error which we can use to improve the accuracy. The method was used in this way for the semi-Lagrangian convection in [5, 30], where it was observed to improve the convergence rate by one order in space and time. When applied to the numerical transport of the level set

function, the back-and-forth error compensation and correction method is known to significantly reduce the volume conservation error, see [24]. Introduction of a limiter in this predictor-corrector method [14] helps to eliminate spurious oscillations without loosing the property of improved convergence.

For a given (discrete) solution φ^n at time t^n, the semi-Lagrangian back-and-forth error compensation and correction (BFECC) method finds φ^{n+1} in several steps:

Algorithm 1 Basic semi-Lagrangian BFECC method.

1: Solve (2.1) forward in time with the semi-Lagrangian method (2.2) to obtain $\widehat{\varphi}^{n+1} = \varphi_I^n(\mathbf{x}(t^n))$. The interpolation at $\mathbf{x}(t^n)$ can be combined with a limiter to enforce monotonicity. Integrating, interpolating and limiting define a nonlinear operator $\widehat{\varphi}^{n+1} = \mathcal{F}(\varphi^n)$.
2: Solve (2.1) backward in time with the same semi-Lagrangian method to obtain $\widehat{\varphi}^n = \widehat{\varphi}_I^{n+1}(\widetilde{\mathbf{x}}(t^{n+1}))$. The interpolation and limiting procedures for the backward step are the same as for step 1. By analogy with step 1, this defines a nonlinear operator $\widehat{\varphi}^n = \mathcal{B}(\widehat{\varphi}^{n+1})$.
3: Calculate the defect $e = \frac{1}{2}(\varphi^n - \widehat{\varphi}^n)$.
4: Update $\widetilde{\varphi}^n = \varphi^n + e$.
5: Solve (2.1) forward in time with the same semi-Lagrangian method to obtain $\varphi^{n+1} = \mathcal{F}(\widetilde{\varphi}^n)$.

The above predictor-corrector scheme is expected to improve the order of the solver to $O(\frac{h^{p'+2}}{\Delta t} + (\Delta t)^3)$, where $O(h^{p'+1})$ is the interpolation accuracy subject to the limiting postprocessing and $O((\Delta t)^2)$ is the accuracy of the trapezoidal rule for the numerical integration along characteristics. While the algorithm can be proved to retain the unconditional stability of the original (one-way) semi-Lagrangian method, it is still prone to produce spurious oscillations due to step 4. To suppress these oscillations, we follow [14] and introduce a limiting procedure for BFECC.

2.3. BFECC limiting. We assume for a moment that an interpolation without limiting is applied in the semi-Lagrangian method. In this case, both operators $\mathcal{F}$ and $\mathcal{B}$ are linear and are denoted by $\mathcal{F}_L$ and $\mathcal{B}_L$. We can then rewrite the algorithm 1 as $\varphi^{n+1} = \mathcal{F}_L(\frac{3}{2}\varphi^n - \mathcal{B}_L(\mathcal{F}_L(\varphi^n)))$. For the defect e from step 3 it holds $e = \frac{1}{2}(\varphi^n - \mathcal{B}_L(\mathcal{F}_L(\varphi^n)))$. One observes the following chain:

$$\begin{aligned}
\hat{e} &:= \varphi^n - \mathcal{B}_L(\varphi^{n+1}) - e \\
&= \varphi^n - \mathcal{B}_L(\mathcal{F}_L(\varphi^n + e)) - e \\
&= \varphi^n - \mathcal{B}_L(\mathcal{F}_L(\varphi^n)) - e - \mathcal{B}_L(\mathcal{F}_L(e)) \\
&= 2e - e - \mathcal{B}_L(\mathcal{F}_L(e)) \\
&= e - \mathcal{B}_L(\mathcal{F}_L(e)).
\end{aligned} \tag{2.6}$$

According to [14], the violation of $|\hat{e}(\mathbf{x}_0)| \le |e(\mathbf{x}_0)|$ for a node $\mathbf{x}_0$ indicates the appearance of oscillations due to the correction and one has to limit e in such nodes. If $|\hat{e}(\mathbf{x}_0)| > |e(\mathbf{x}_0)|$, we perform the limiting of e at all nodes involved in the interpolation procedure for a node $\mathbf{x}_0$. The limiting is done by inspecting a row $r_{\mathcal{F}}(\mathbf{x}_0)$ and a row $r_{\mathcal{B}}(\mathbf{x}_0)$ of the discrete operators $\mathcal{F}$ and $\mathcal{B}$ (possibly nonlinear) for each node $\mathbf{x}_0$ and

performing the following correction of function $e(\mathbf{x})$:

$$\begin{aligned} &\text{for all } \mathbf{x}_0 \text{ initialize} \quad \tilde{e}(\mathbf{x}_0) = e(\mathbf{x}_0) \\ &\text{for all } \mathbf{x}_0 \text{ s.t. } |\hat{e}(\mathbf{x}_0)| > |e(\mathbf{x}_0)| \\ &\qquad \text{for all } \mathbf{x}_i \text{ contributing to } r_{\mathcal{F}}(\mathbf{x}_0) \text{ or } r_{\mathcal{B}}(\mathbf{x}_0) \\ &\qquad\qquad \tilde{e}(\mathbf{x}_i) = \text{minmod}(e(\mathbf{x}_0), \tilde{e}(\mathbf{x}_i)) \end{aligned} \tag{2.7}$$

Here $\mathbf{x}_i$ is the position of a grid node that corresponds to nonzero entry in a row of the discrete operators. The function $\text{minmod}(a, b)$ is given by:

$$\text{minmod}(a,b) = \begin{cases} \min(a,b) & a,b > 0, \\ \max(a,b) & a,b < 0, \\ 0, & \text{otherwise.} \end{cases}$$

We summarize the semi-Lagrangian method with BFECC and the limiting in

Algorithm 2 Semi-Lagrangian BFECC method with limiting.

1: Perform forward semi-Lagrangian step $\widehat{\varphi}^{n+1} = \mathcal{F}(\varphi^n)$.
2: Perform backward semi-Lagrangian step $\widehat{\varphi}^n = \mathcal{B}(\widehat{\varphi}^{n+1})$.
3: Calculate the defect $e = \frac{1}{2}(\varphi^n - \widehat{\varphi}^n)$ and correct $\widetilde{\varphi}^n = \varphi^n + e$.
4: Perform forward semi-Lagrangian step $\widetilde{\varphi}^{n+1} = \mathcal{F}(\widetilde{\varphi}^n)$.
5: Perform backward semi-Lagrangian step $\tilde{\varphi}^n = \mathcal{B}(\widetilde{\varphi}^{n+1})$ and calculate $\hat{e} = \varphi^n - \tilde{\varphi}^n - e$.
6: Compute $\tilde{e}$ by performing limiting of e at nodes where $|\hat{e}| > |e|$ using (2.7).
7: Perform forward semi-Lagrangian step $\varphi^{n+1} = \mathcal{F}(\varphi^n + \tilde{e})$.

Now we proceed with the definition of the discrete spatial operators that we use.

3. Spatial discretization. For the spatial discretization we use octree cubic meshes, which allow fast dynamic mesh adaptation based on error indicators. We introduce spatial derivatives operators for grid functions defined in mesh nodes. The derivatives are further needed to define a tricubic nodal interpolation operation for a graded octree mesh. The interpolation is of higher order and hence is not monotone. Therefore, we also introduce an appropriate limiter. Finally, we also discuss the grid adaptation strategy.

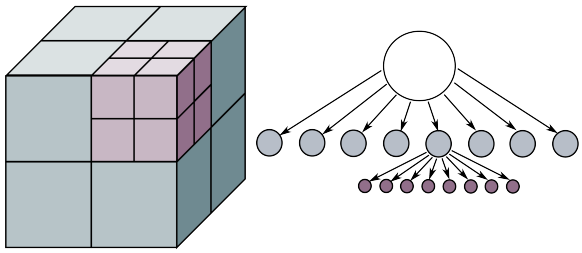

FIG. 3.1. *An octree mesh (left) and its representation as a tree (right).*

Consider a graded octree mesh with cubic cells, see Fig. 3.1. An octree mesh is *graded* if the size of cells sharing (a part of) an edge or a face can differ in size only

by the factor of two. This restriction simplifies support of mesh connectivity and construction of discrete differential operators.

3.1. Second-order partial derivatives. Assume that φ is a scalar quantity given in all nodes of a graded octree mesh. In this section, we define a nodal approximation to $\frac{\partial^2\varphi}{\partial x^2}$, $\frac{\partial^2\varphi}{\partial y^2}$, and $\frac{\partial^2\varphi}{\partial z^2}$. Without loss of generality we consider only the approximation of $\frac{\partial^2\varphi}{\partial x^2}$ at a node i.

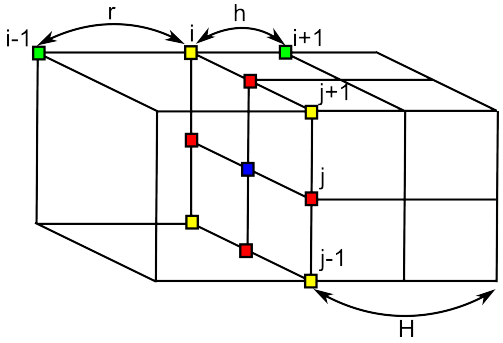

FIG. 3.2. *Nodes marked by yellow, red and blue boxes indicate three possible type of nodes for the approximation of* φ_{xx}.

One may encounter three possible types of nodes presented in Fig.3.2:

1. A node i has type A (marked yellow in Fig. 3.2), if there are two immediate adjacent nodes $i-1$ and $i+1$ in the direction Ox. The distances $x_i - x_{i-1}$ and $x_{i+1} - x_i$ may appear different.
2. A node has type B (marked red in Fig. 3.2), if it is a hanging node and has one immediate neighbor in the direction Ox, but may have two immediate neighbors of type A in either direction Oy or direction Oz. In Fig.3.2 node j has type B and $j-1$ and $j+1$ are of type A.
3. A node has type C (marked blue in Fig.3.2), if it is a hanging node and all of its immediate neighbors in Oy and Oz directions are hanging nodes.

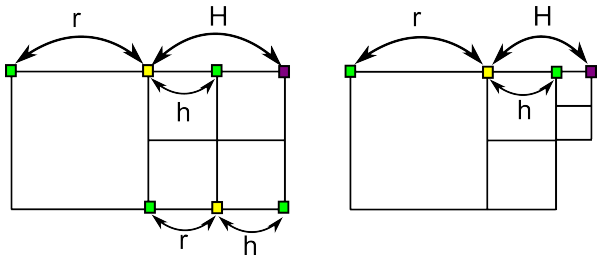

FIG. 3.3. *The node marked by yellow box is the current node i of type A. Green boxes are the nodes that are $i-1$ and $i+1$ intimidate neighbors in the given direction Ox. Purple boxes mark the $i+2$ non-immediate neighbor node.*

Let us first consider a node i of type A. In this case $r = x_i - x_{i-1}$ and $h = x_{i+1} - x_i$ are the distances to the adjacent nodes in direction Ox. If $r = h$, the simple second

order formula for the regular grid is used, see Fig. 3.3 (left, bottom):

$$\left.\frac{\partial^2\varphi}{\partial x^2}\right|_i = \frac{\varphi_{i-1} - 2\varphi_i + \varphi_{i+1}}{h^2}. \tag{3.1}$$

Assume now $r > h$. On a graded octree mesh there may be no adjacent node $i-2$ behind the node $i-1$, but there will be always an adjacent node $i+2$ beyond the node $i+1$ due to the constraint of a *graded* octree meshes. Two possible configurations for the node $i+2$ are illustrated in Fig.3.3. For the case $H = 2h = r$ the second-order approximation of φ_{xx} in the node i becomes, see Fig. 3.3 (left, top):

$$\left.\frac{\partial^2\varphi}{\partial x^2}\right|_i = \frac{\varphi_{i-1} - 2\varphi_i + \varphi_{i+2}}{4h^2}, \tag{3.2}$$

while for the case $H = \frac{3h}{2}$ the second-order approximation is, see Fig. 3.3 (right):

$$\left.\frac{\partial^2\varphi}{\partial x^2}\right|_i = \frac{15\varphi_{i-1} - 21\varphi_i - 42\varphi_{i+1} + 48\varphi_{i+2}}{63h^2}. \tag{3.3}$$

To compute discretization of φ_{xx} in mesh nodes of other types, we first compute it using (3.1)-(3.3) for type A nodes. Further, we calculate approximation of φ_{xx} in type B nodes by averaging the discrete second derivatives at two neighboring type A nodes:

$$\left.\frac{\partial^2\varphi}{\partial x^2}\right|_j = \frac{1}{2}\left(\left.\frac{\partial^2\varphi}{\partial x^2}\right|_{j-1} + \left.\frac{\partial^2\varphi}{\partial x^2}\right|_{j+1}\right). \tag{3.4}$$

Finally, approximation of φ_{xx} in type C nodes is found by averaging the discrete second derivatives at four neighboring type B nodes:

$$\left.\frac{\partial^2\varphi}{\partial x^2}\right|_{j,k} = \frac{1}{4}\left(\left.\frac{\partial^2\varphi}{\partial x^2}\right|_{j-1,k} + \left.\frac{\partial^2\varphi}{\partial x^2}\right|_{j+1,k} + \left.\frac{\partial^2\varphi}{\partial x^2}\right|_{j,k-1} + \left.\frac{\partial^2\varphi}{\partial x^2}\right|_{j,k+1}\right). \tag{3.5}$$

Indices j and k denote the numbering in Oy and Oz directions, respectively.

Now we are ready to define the higher order interpolation operator that we use for the semi-Lagrangian method as well as for the re-interpolation procedure within an adaptive mesh refinement/coarsening step.

3.2. Interpolation. Trilinear interpolation is a popular and natural choice for an octree mesh with cubic cells. Consider a point (x, y, z) in a cubic cell with the edge size h; the cell center is (c_x, c_y, c_z) and the values $\varphi_{1\ldots8}$ are given in vertices, cf. Fig.3.4. The corresponding trilinear interpolant $\varphi_L(x, y, z)$ can be written as

$$\begin{array}{rllll} \varphi_L(x,y,z) = & \varphi_1 & (1-k_x)(1-k_y)(1-k_z) & +\varphi_2 & k_x(1-k_y)(1-k_z) \\ & \varphi_3 & (1-k_x)k_y(1-k_z) & +\varphi_4 & k_xk_y(1-k_z) \\ & \varphi_5 & (1-k_x)(1-k_y)k_z & +\varphi_6 & k_x(1-k_y)k_z \\ & \varphi_7 & (1-k_x)k_yk_z & +\varphi_8 & k_xk_yk_z, \end{array} \tag{3.6}$$

with $k_x = \frac{x-c_x}{h} + \frac{1}{2}$, $k_y = \frac{y-c_y}{h} + \frac{1}{2}$, $k_z = \frac{z-c_z}{h} + \frac{1}{2}$. One may consider the difference between the trilinear interpolation and the tricubic interpolation at point (x, y, z) in

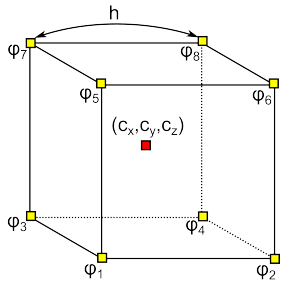

FIG. 3.4. *Illustration for the cube with the side of the size h and the center (c_x, c_y, c_z) (red box) and variables defined at vertices of the cube $\varphi_{1\dots8}$ (yellow boxes).*

the form of correction [19]:

$$\delta\varphi_C(x,y,z) = \frac{1}{2}\left(\left.\frac{\partial^2\varphi}{\partial^2 x}\right|_{(x,y,z)} k_x(1-k_x) + \left.\frac{\partial^2\varphi}{\partial^2 y}\right|_{(x,y,z)} k_y(1-k_y) + \left.\frac{\partial^2\varphi}{\partial^2 z}\right|_{(x,y,z)} k_z(1-k_z)\right) h^2. \tag{3.7}$$

The type of resulting interpolation will depend on the order of approximation of the second partial derivatives in (3.7). Of course, the second partial derivatives of φ at point (x,y,z) are not known. We approximate them using nodal values $\left.\frac{\partial^2\varphi}{\partial^2 x}\right|_{1\dots8}$, $\left.\frac{\partial^2\varphi}{\partial^2 y}\right|_{1\dots8}$, $\left.\frac{\partial^2\varphi}{\partial^2 z}\right|_{1\dots8}$:

$$\begin{aligned}\left.\frac{\partial^2\varphi}{\partial^2 x}\right|_{(x,y,z)} = \frac{1}{3}\,[& \\ & \left.\frac{\partial^2\varphi}{\partial^2 x}\right|_1 (2-k_x)(1-k_y)(1-k_z) + \left.\frac{\partial^2\varphi}{\partial^2 x}\right|_2 (1+k_x)(1-k_y)(1-k_z) \\ & \left.\frac{\partial^2\varphi}{\partial^2 x}\right|_3 (2-k_x)k_y(1-k_z) + \left.\frac{\partial^2\varphi}{\partial^2 x}\right|_4 (1+k_x)k_y(1-k_z) \\ & \left.\frac{\partial^2\varphi}{\partial^2 x}\right|_5 (2-k_x)(1-k_y)k_z + \left.\frac{\partial^2\varphi}{\partial^2 x}\right|_6 (1+k_x)(1-k_y)k_z \\ & \left.\frac{\partial^2\varphi}{\partial^2 x}\right|_7 (2-k_x)k_yk_z + \left.\frac{\partial^2\varphi}{\partial^2 x}\right|_8 (1+k_x)k_yk_z\ ]\end{aligned} \tag{3.8}$$

and approximate similarly $\left.\frac{\partial^2\varphi}{\partial^2 y}\right|_{(x,y,z)}$ and $\left.\frac{\partial^2\varphi}{\partial^2 z}\right|_{(x,y,z)}$, note that constants added to k_x in (3.8) should be transfered to k_y and k_z respectively. Discretization of the second derivatives at nodes was described in section 3.1.

We note that (3.8) is different from the trilinear interpolation (3.6). The coefficients in (3.8) are computed in such a way that higher order terms are cancelled in the corresponding Taylor series. The tricubic interpolation is defined as $\varphi_C(x,y,z) = \varphi_L(x,y,z) + \delta\varphi_C(x,y,z)$. The interpolation is not monotone; therefore, a limiter should be introduced to reduce oscillations.

3.3. Interpolation limiter. One possible limiting strategy is the following. Instead of (3.8) one computes

$$\left.\frac{\partial^2\varphi}{\partial^2 x}\right|_{(x,y,z)} = \operatorname{minmod}\left(\left.\frac{\partial^2\varphi}{\partial^2 x}\right|_{1\dots8}\right) \tag{3.9}$$

with

$$\text{minmod}(x_1,...,x_n) = \begin{cases} x_k,\ s.t.\ |x_k| = \min\{|x_1|,...,|x_n|\}, \text{if } x_i x_j \geq 0\, \forall i,j \in \{1,...,n\}, \\ 0, \text{otherwise.} \end{cases}$$

Substituting (3.9) and similar expressions for other second order derivatives into (3.7), one obtains a triquadratic correction $\delta\varphi_Q$ from [19]. We will denote the corresponding interpolant by $\varphi_Q = \varphi_L + \delta\varphi_Q$. Below we build a different limiter delivering better accuracy.

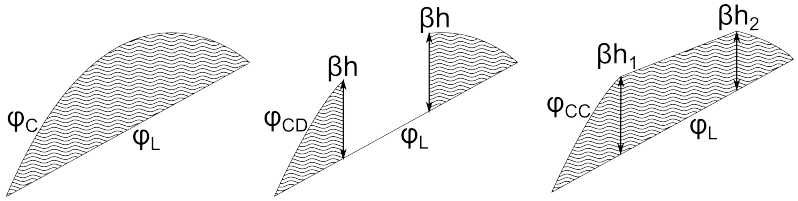

FIG. 3.5. *Filled region denotes a correction to the original linear interpolation φ_L. Left picture shows the original cubic interpolant φ_C. Middle picture shows the discontinuous limiting of the cubic correction φ_{CD}. Right picture shows the continuous limiting of the cubic correction φ_{CC}.*

While the correction $\delta\varphi_C$ from (3.7) have the magnitude of the order $O(h^2)$, the factor multiplying h^2 may be quite large. In [34] the following limiter for a high order interpolation is introduced:

$$\varphi_{CD}(x,y,z) = \begin{cases} \varphi_L(x,y,z), & |\delta\varphi_C(x,y,z)| > \beta h, \\ \varphi_L(x,y,z) + \delta\varphi_C(x,y,z), & \text{otherwise.} \end{cases} \tag{3.10}$$

The authors of [34] noted that for $\beta = \frac{1}{20}$ the limiter in (3.10) essentially means that one switches to the lower order interpolation whenever the local curvature for a level-set function is higher then 3 grid cells of the size h. As illustrated in the middle picture of Figure 3.5, this approach leads to a discontinuity in the interpolant. This discontinuity becomes even more pronounced in the presence of hanging nodes. To overcome this issue, we slightly modify (3.10) to obtain a more regular interpolation:

$$\varphi_{CC}(x,y,z) = \begin{cases} \varphi_L(x,y,z) + \beta h(x,y,z), & |\delta\varphi_C(x,y,z)| > \beta h(x,y,z) \\ \varphi_L(x,y,z) + \delta\varphi_C(x,y,z), & \text{otherwise.} \end{cases} \tag{3.11}$$

Here we define $h(x,y,z)$ as the trilinear interpolant (3.6) for nodal values $h_{1...8}$, where h_i for the node i is defined as the minimal size of a cell over all cells sharing i. While this definition leads to continuous interpolation on a regular grid, we still may encounter a discontinuity of $O(h^2)$ near a hanging node.

3.4. Discrete gradient. The discrete gradient of a nodal function φ can be useful for adaptivity purposes and for the sake of completeness we present the method of the first derivates discretization.

Assume that for a given node i, there exists a pair of immediate neighbors $i-1$ and $i+1$ in the direction Ox, see Fig.3.6. The discrete variable φ has values φ_i, φ_{i-1} and φ_{i+1} at these nodes. For $r = x_i - x_{i-1}$ and $h = x_{i+1} - x_i$, we define a $\left.\frac{\partial\varphi}{\partial x}\right|_i$ component of the gradient at the node i with the second order of accuracy by:

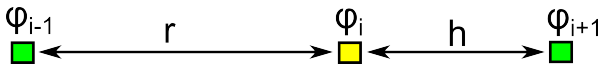

FIG. 3.6. *Two immediate neighbors $i-1$ and $i+1$ for the node i with the grid spacing $r = x_i - x_{i-1}$ and $h = x_{i+1} - x_i$ and values of the variable φ_{i-1}, φ_i and φ_{i+1}.*

$$\left.\frac{\partial\varphi}{\partial x}\right|_i = \frac{(\varphi_{i+1}-\varphi_i)\,r^2 + (\varphi_i - \varphi_{i-1})\,h^2}{hr(h+r)}. \tag{3.12}$$

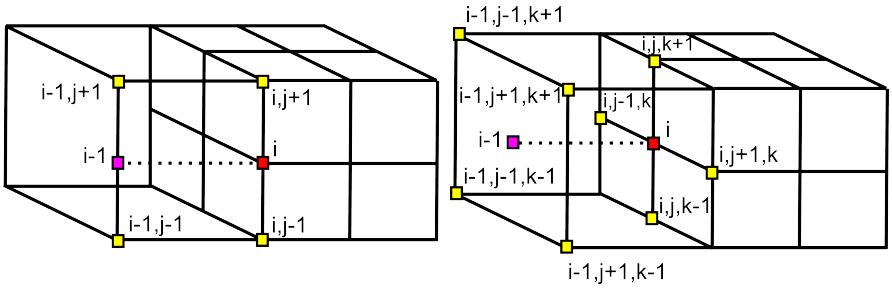

FIG. 3.7. *Two possible configuration of hanging nodes. Red box denotes node i, yellow boxes denote interpolation stencil and purple box denote the node where the interpolation is being sought.*

In the case of a hanging node one of the immediate neighbors may not exist, see Fig.3.2. In this case, we define the missing neighbor value by the third-order interpolation following [19][1]. Let i be a hanging node and assume that there is no immediate adjacent node $i-1$ in the direction Ox. Without loss of generality assume that $r > h$. In this case there are two possible configurations shown in Fig.3.7. For the left configuration in Fig.3.7 we define φ_{i-1} as:

$$\varphi_{i-1} = \frac{1}{2}\left(\varphi_{i-1,j+1} - \varphi_{i,j+1} + \varphi_{i-1,j-1} - \varphi_{i,j-1}\right) + \varphi_i, \tag{3.13}$$

whereas for the right configuration in Fig.3.7 we define the missing value by

$$\begin{aligned}\varphi_{i-1} = {} & \frac{1}{4}\left(\varphi_{i-1,j+1,k+1} + \varphi_{i-1,j+1,k-1} + \varphi_{i-1,j-1,k+1} + \varphi_{i-1,j-1,k-1}\right) \\ & - \frac{1}{2}\left(\varphi_{i,j+1,k} + \varphi_{i,j-1,k} + \varphi_{i,j,k+1} + \varphi_{i,j,k-1}\right) + 2\varphi_i.\end{aligned} \tag{3.14}$$

3.5. Grid adaptivity. One refines/coarsens the grid in the course of computations based on the information given by the nodal values of φ and discrete derivatives of φ. If one is interested in computing the transport of a concentration density function, then the adaptivity criterion can be based either on the values of φ or on the values of $\nabla\varphi$. For the level set function transport, the rule can rely on the estimated distance to the zero level of φ or on the estimated local curvature $\kappa = \nabla\cdot(\nabla\varphi/|\nabla\varphi|)$.

Numerical experiments in section 4 use for an adaptivity criterion only the values of φ in grid nodes. More precisely, we adopt the following rules:

[1]The only reason not to use the interpolation defined in sections 3.2 and 3.3 here is the computational convenience.

- Concentration field transport in section 4.1: the cell is split if $|\varphi_{1\ldots8} - 0.7| \leq 0.4$ at least for one nodal value. The criterion allows to cover both the region of high gradients and the tip of the solution where the largest error is observed if one uses a uniform Cartesian grid.
- Concentration field transport in section 4.2: the cell is split if the nodal values of φ belong to specified intervals or if the nodal function $(\varphi_{1\ldots8} - 0.5)$ has different signs. For this test, it leads to the grid refinement in the region of high gradients.
- Level set function transport in section 4.3: the cell is split if its nodal values have both negative and positive signs. This results in a thin layer of highly refined cells near the free surface.

4. Numerical experiments. This section collects the results of numerical experiments for several test problems. We assess the performance of the semi-Lagrangian method described in the previous sections. We also compare the new scheme with a few more standard or simplified versions of the above semi-Lagrangian method. These are the semi-Lagrangian scheme with linear interpolation as used in [6], the BFECC scheme with trilinear interpolation and the semi-Lagrangian with higher order interpolation (and limiting), but without back-and-forth correction. We experiment with a smooth analytical solution to the transport problem in order to check and to compare the accuracy and convergence orders of the schemes. Further we consider concentration transport in a complex velocity field to study the mass conservation and monotonicity properties of the schemes. Finally, we show numerical results for the "Enright test" [6, 15]. This test demonstrates how well the schemes are suited for the numerical integration of the level set equation, describing the evolution of a free-surface passively advected by a velocity field.

In all experiments the time step is chosen according to formula

$$\Delta t = \min_{cells} \frac{\mathrm{CFL}\, h_{cell}}{\max\{\bar{u}_1, \bar{u}_2, \bar{u}_3\}},$$

where $\bar{u}_i$ denotes averaged over the cell i-th component of the advection vector. In sections 4.1 and 4.2 we use CFL = 1, in section 4.3 we use CFL = 2, although essentially larger CFL values are applicable as well.

4.1. Analytical solution test. We consider the case of the smooth solution to (2.1) given by

$$C(x, y, z, t) = e^{-64\,[(x-x_c(t))^2+(y-y_c(t))^2+(z-z_c(t))^2]}.$$

The gaussian hat function is transported and rotated by the velocity field $\mathbf{v} = \{4\pi(0.6 - y), 4\pi(x - 0.6), 1.0\}$ so that the hat center coordinates are $x_c = 0.6 - 0.25\,\cos(4\pi\,t)$, $y_c = 0.6 - 0.25\,\sin(4\pi\,t)$, $z_c = 0.5 + t$. Starting with a given $C(x, y, z, 0)$, we compute the numerical solution for the velocity field and measure the numerical error in the maximum (C-) and L_2-norms at $t = 0.1$. Since the Gaussian impulse almost vanishes on the boundary of computational domain $[0, 1]^3$, we neglect the impact of the boundary conditions.

Fig.4.1 shows the solution computed with the new BFECC method on the uniform grid, $h = 1/64$. We observe no overshoots and a minor undershoot of order 10^{-7}. Further, Fig.4.2 shows the error fields (numerical solution minus exact solution) for BFECC with high-order interpolation and limiting, BFECC with linear interpolation, semi-Lagrangian method (without BFECC) with high-order interpolation and limiting

and semi-Lagrangian method with linear interpolation. All error plots are presented for the same time $t = 0.1$. For the semi-Lagrangian methods without BFECC the maximum of the error absolute values is achieved in the center of the Gaussian hat, while for the BFECC methods the error is less and is more smeared.

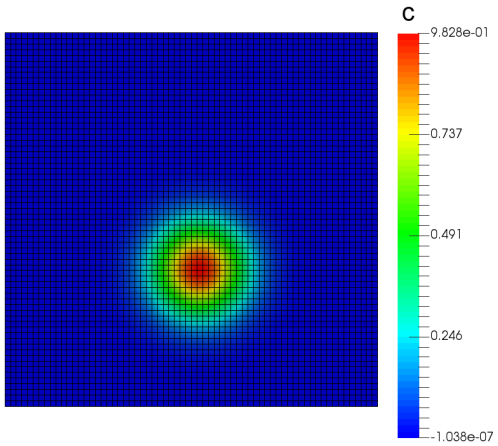

FIG. 4.1. *Solution for the new BFECC method with high-order interpolation. Cutplane $z = 0.6$ of the uniform grid, $h = 1/64$.*

Table 4.1 provides norms of the errors for the above mentioned advection methods on uniform cubic grids. One can see that high-order interpolation provides only slightly more accurate solution since the largest error is located near the center of the impulse where the interpolation limiter may cut the high-order correction. On the other hand, BFECC schemes show more accurate results compared to semi-Lagrangian methods without error correction. The new BFECC scheme demonstrates higher than the second order convergence in the L^2-norm while the standard semi-Lagrangian with linear interpolation has less than the first order.

Grid $h^{-1}/\,N_{steps}$	N_{cells}	BFECC high-order		BFECC linear		Semi-Lagrangian high-order		Semi-Lagrangian linear	
		Err_C	Err_{L_2}	Err_C	Err_{L_2}	Err_C	Err_{L_2}	Err_C	Err_{L_2}
32/ 25	32 768	1.17e-1	5.24e-3	1.28e-1	5.73e-3	4.03e-1	1.65e-2	4.40e-1	2.01e-2
64/ 49	262 144	2.32e-2	7.56e-4	2.45e-2	8.82e-4	2.40e-1	8.67e-3	2.78e-1	1.20e-2
128/ 98	2 097 152	1.01e-2	1.77e-4	1.00e-2	2.07e-4	1.24e-1	3.74e-3	1.62e-1	6.70e-3
order		1.77	2.44	1.84	2.40	0.85	1.07	0.72	0.79

TABLE 4.1
Errors in C- and L_2-norms for the uniform grids.

Now we repeat the experiment on dynamically adapted octree grids. Fig.4.3 shows the computed solution and the corresponding error for the new BFECC method with high-order interpolation on the dynamic grid with $h_{min} = 1/128, h_{max} = 1/32$. We note that we still have no overshoots in solution and the undershoot is of order 10^{-10}.

Table 4.2 presents the error norms for the examined semi-Lagrangian methods on dynamically adapted grids. The new BFECC scheme demonstrates the second order

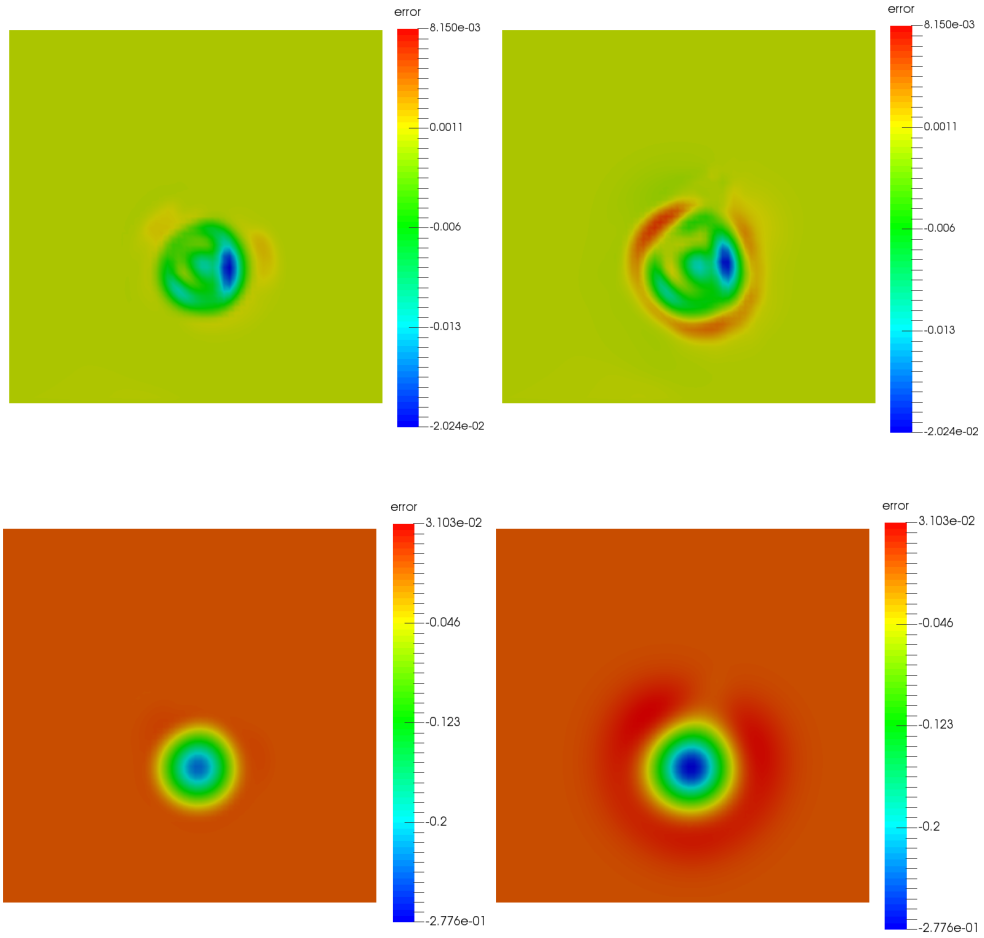

FIG. 4.2. *Error fields for BFECC with high-order interpolation (top-left), BFECC with linear interpolation (top-right), semi-Lagrangian method with high-order interpolation (bottom-left) and semi-Lagrangian method with linear interpolation (bottom-right). Cutplane* $z = 0.6$ *of the uniform grid,* $h = 1/64$.

convergence in L_2-norm, while the other simplified variants of the method are clearly inferior in terms of the accuracy. Comparing the results in Tables 4.2 and 4.1 we see that grid adaptation for this problem can reduce the number of active degrees of freedom and achieve the same error.

4.2. Mass conservation and monotonicity. The second test studies the conservative and monotone properties of the method. The computational domain is $[0, 2]^3$, the initial concentration equals 1 node-wise inside the sphere of radius $r = 0.3$ centered at (1.2, 1.2, 1.2) and 0 in the rest of domain. The sphere is transported by

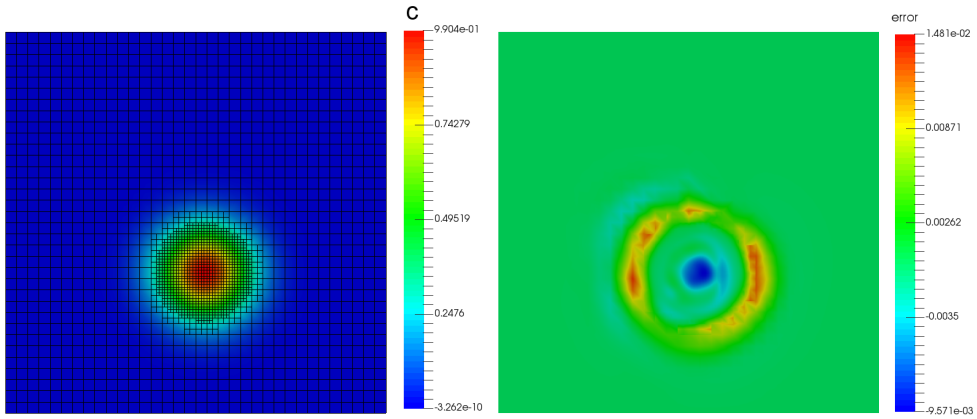

FIG. 4.3. *Solution and error for the new BFECC method with high-order interpolation on the dynamic octree grid, $h_{min} = 1/128$, $h_{max} = 1/32$. Cutplane $z = 0.6$.*

Grid	N_{cells} at $t = 0.1$	BFECC high-order		BFECC linear		Semi-Lagrangian high-order		Semi-Lagrangian linear	
$h_{min}^{-1}/h_{max}^{-1}/N_{steps}$		Err_C	Err_{L_2}	Err_C	Err_{L_2}	Err_C	Err_{L_2}	Err_C	Err_{L_2}
64/16/ 49	7428	3.65e-2	3.64e-3	6.48e-2	7.75e-3	2.40e-1	9.54e-3	2.78e-1	1.63e-2
128/32/ 98	57 170	1.48e-2	8.79e-4	3.69e-2	3.04e-3	1.24e-1	3.88e-3	1.62e-1	1.02e-2
256/64/ 194	544 986	9.70e-3	2.00e-4	1.54e-2	1.01e-3	5.25e-2	1.15e-3	9.03e-2	5.59e-3
order		0.96	2.10	1.04	1.47	1.10	1.53	0.81	0.77

TABLE 4.2
Errors in C- and L_2-norms for dynamically adapted octree grids.

the velocity defined analytically by

$$u_1 = -a\left(e^{ax}\sin(ay+dz) + e^{az}\cos(ax+dy)\right)e^{-\nu d^2 t},$$
$$u_2 = -a\left(e^{ay}\sin(az+dx) + e^{ax}\cos(ay+dz)\right)e^{-\nu d^2 t},$$
$$u_3 = -a\left(e^{az}\sin(ax+dy) + e^{ay}\cos(az+dx)\right)e^{-\nu d^2 t}.$$

This velocity field is the incompressible Navier-Stokes equations solution from [7] proposed for the purpose of benchmarking. The velocity field has no principle direction and has a non-trivial vortical structure. In our experiments we set $a = \pi/4$, $d = \pi/2$, $\nu = 0.1$.

The simulations are run up to $t = 0.108$ when we measure the total mass and compare it to the initial mass for the same grid resolution. The initial mass can deviate from the analytically computed mass due to meshing effects. Two schemes with the high-order interpolation are tested and compared: the new BFECC scheme and the semi-Lagrangian scheme without BFECC correction. We also measure the minimum and maximum concentration values which for the exact solution should be between 0 and 1.

We first experiment with uniform grids. Fig.4.4 shows the initial concentration field with irregular isolines due to mesh effects (left) and the final field obtained with

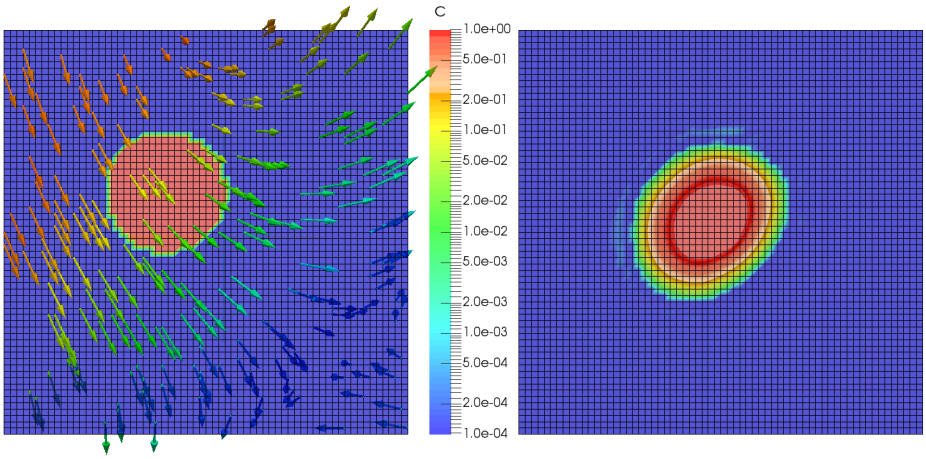

FIG. 4.4. *Initial state (left) and the numerical solutions at $t = 0.108$ for the new BFECC method (right) on the uniform grid, $h = 1/64$.*

the BFECC scheme (right). Table 4.3 shows the mass conservation data for this test on the sequence of the uniform grids. The BFECC scheme provides somewhat better mass conservation, yet the performance of the methods is comparable.

Grid	N_{cells}	Initial mass	BFECC		Semi-Lagrangian	
$h^{-1}/\,N_{steps}$			Mass	Error	Mass	Error
32/ 6	32 768	0.113281	0.112670	6.1e-4	0.110609	2.7e-3
64/ 11	262 144	0.113129	0.113480	3.5e-4	0.111855	1.3e-3
128/ 22	2 097 152	0.113106	0.113403	2.9e-4	0.112566	5.4e-4

TABLE 4.3
Mass conservation for the BFECC and semi-Lagrangian schemes. Uniform grids.

Table 4.4 presents the minimum and maximum values for the numerical solutions. Both schemes have undershoots and overshoots, and for both schemes these spikes are decreasing with the first order rate.

Grid	BFECC		Semi-Lagrangian	
h^{-1}	Maximum	Minimum	Maximum	Minimum
32	1.01	-0.018	1.008	-0.01
64	1.004	-0.005	1.004	-0.005
128	1.002	-0.003	1.002	-0.003

TABLE 4.4
Minimum and maximum concentration values for the BFECC and semi-Lagrangian schemes. Uniform grids.

We repeat the test for the BFECC scheme on the dynamically refined octree grids. The grids are refined in the region where the concentration lies in $[0.1, 0.9]$ or $[0.001, 0.999]$ intervals. Fig.4.5 shows the initial concentration for the dynamically adapted octree grid with $h_{min} = 1/128, h_{max} = 1/32$ (left), as well as the final concentration at $t = 0.108$ (right). Table 4.5 shows the mass conservation data and

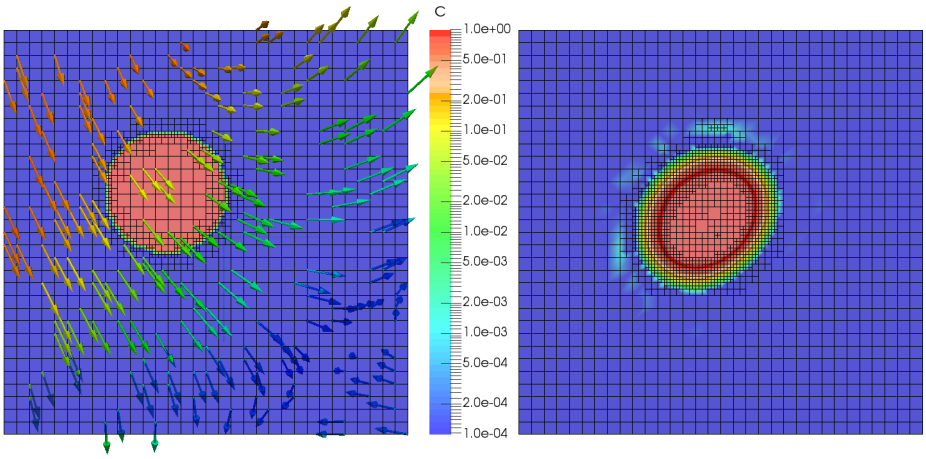

FIG. 4.5. *Initial state (left) and the numerical solutions at $t = 0.108$ for the new BFECC method (right) on the dynamically adapted octree grid with $h_{min} = 1/128, h_{max} = 1/32$.*

the minimum and maximum values for the numerical solutions on the sequence of the dynamically adapted octree grids. Wider refinement region provides more accurate solution for the BFECC method.

Grid	N_{cells}	Initial mass	Mass	Error	Maximum	Minimum
$h_{min}^{-1}/h_{max}^{-1}/\,N_{steps}$	at $t = 0.108$	Refinement for $0.1 \le \varphi \le 0.9$				
64/16/ 12	12342	0.113129	0.114177	1.05e-3	1.011	-0.018
128/32/ 23	71518	0.113106	0.113742	6.36e-4	1.015	-0.016
256/64/ 46	448769	0.113066	0.11336	2.94e-4	1.010	-0.011
		Refinement for $0.001 \le \varphi \le 0.999$				
64/16/ 12	17774	0.113129	0.112693	4.36e-4	1.009	-0.013
128/32/ 23	91363	0.113106	0.113296	1.90e-4	1.006	-0.009
256/64/ 46	537228	0.113066	0.113113	4.70e-5	1.004	-0.007

TABLE 4.5

Mass conservation and monotonicity for the BFECC scheme on the dynamically adapted octree grids.

4.3. Level set function transport. We study the performance of the new BFECC scheme as the numerical solver for the level set equation for the Enright test [6, 15].

Following [6, 15] we consider the transport of a level set function by the 3D incompressible velocity field

$$
\begin{aligned}
u_1 &= 2\,\cos(\pi\,t/3)\,\sin^2(\pi\,x)\,\sin(2\pi\,y)\,\sin(2\pi\,z),\\
u_2 &= -\,\cos(\pi\,t/3)\,\sin(2\pi\,x)\,\sin^2(\pi\,y)\,\sin(2\pi\,z),\\
u_3 &= -\,\cos(\pi\,t/3)\,\sin(2\pi\,x)\,\sin(2\pi\,y)\,\sin^2(\pi\,z).
\end{aligned}
$$

At $t = 0$, the level set function φ is the signed distance function for the sphere of radius $r = 0.15$ centered at $(0.35, 0.35, 0.35)$. For $t > 0$, φ solves the transport equation (2.1) and its zero level implicitly defines the position of the free surface at time t.

The total simulation time is $T = 3$. The adaptive grids are dynamically refined towards the surface, with $h_{max} = 1/16$ and h_{min} varying from 1/64 to 1/512. For the exact solution, the final position of the surface coincides with the initial one. An efficient numerical solver for the level set equation is expected to recover accurately the shape of the free surface, preserve its smoothness, connectivity and conserve the volume of the free surface interior. It is also common to supplement a numerical solver with a volume correction technique, see, e.g., [13]. We perform such a volume correction by adding a suitable constant to φ every time step.

Fig. 4.6 shows the snapshots of the surface at five time moments: $t = 0$, 0.76, 1.5, 2.26, 3. One observes the convergence of shapes and better smoothness with the increase of the number refinement levels. For the first three grids the surface is tearing while for $h_{min} = 1/512$ it remains simple connected.

For comparison we run the same experiment with the original semi-Lagrangian method and the linear interpolation. Fig.4.7 shows the snapshots of the surface at the same time instances. For the first four grids the surface looses simple connectivity much earlier than for the BFECC scheme with the high order interpolation. For the coarsest grid the volume disappears completely. Only for the finest grid with $h_{min} = 1/512$ the surface remains simple connected and the results are comparable to the BFECC.

5. Conclusion. We have presented the new semi-Lagrangian method for the numerical integration of the linear transport equation on graded octree meshes. The method features a higher order interpolation procedure with a compact nodal stencil. The higher order interpolation is supplemented with a limiting procedure to avoid undershoots and overshoots in numerical solutions. To increase the overall accuracy, we also use the back-and-forth error compensation correction technique. The method allows octree grids to adapt dynamically on every time step according to a prescribed criterion. In numerical experiments the method demonstrates at least second order accuracy and admits minor overshoots and undershoots. Conservation properties were found to be acceptable. Simplicity, numerical stability, higher order accuracy, good conservation and monotonicity properties make this method appealing whenever the use of the adaptive Cartesian cubic meshes is desired.

Acknowledgement. The authors thank R.Yanbarisov for the valuable help in the numerical tests.

REFERENCES

[1] A. Bonito, J.-L. Guermond, and S. Lee, *Numerical simulations of bouncing jets*, arXiv preprint arXiv:1501.04322, (2015).
[2] M. Braack and T. Richter, *Solutions of 3d Navier–Stokes benchmark problems with adaptive finite elements*, Computers & fluids, 35 (2006), pp. 372–392.
[3] J. A. Carrillo and F. Vecil, *Nonoscillatory interpolation methods applied to vlasov-based models*, SIAM Journal on Scientific Computing, 29 (2007), p. 1179.
[4] T. F. Dupont and Y. Liu, *Back and forth error compensation and correction methods for removing errors induced by uneven gradients of the level set function*, Journal of Computational Physics, 190 (2003), pp. 311–324.
[5] ———, *Back and forth error compensation and correction methods for semi-Lagrangian schemes with application to level set interface computations*, Mathematics of Computation, (2007), pp. 647–668.
[6] D. Enright, F. Losasso, and R. Fedkiw, *A fast and accurate semi-Lagrangian particle level set method*, Computers & structures, 83 (2005), pp. 479–490.

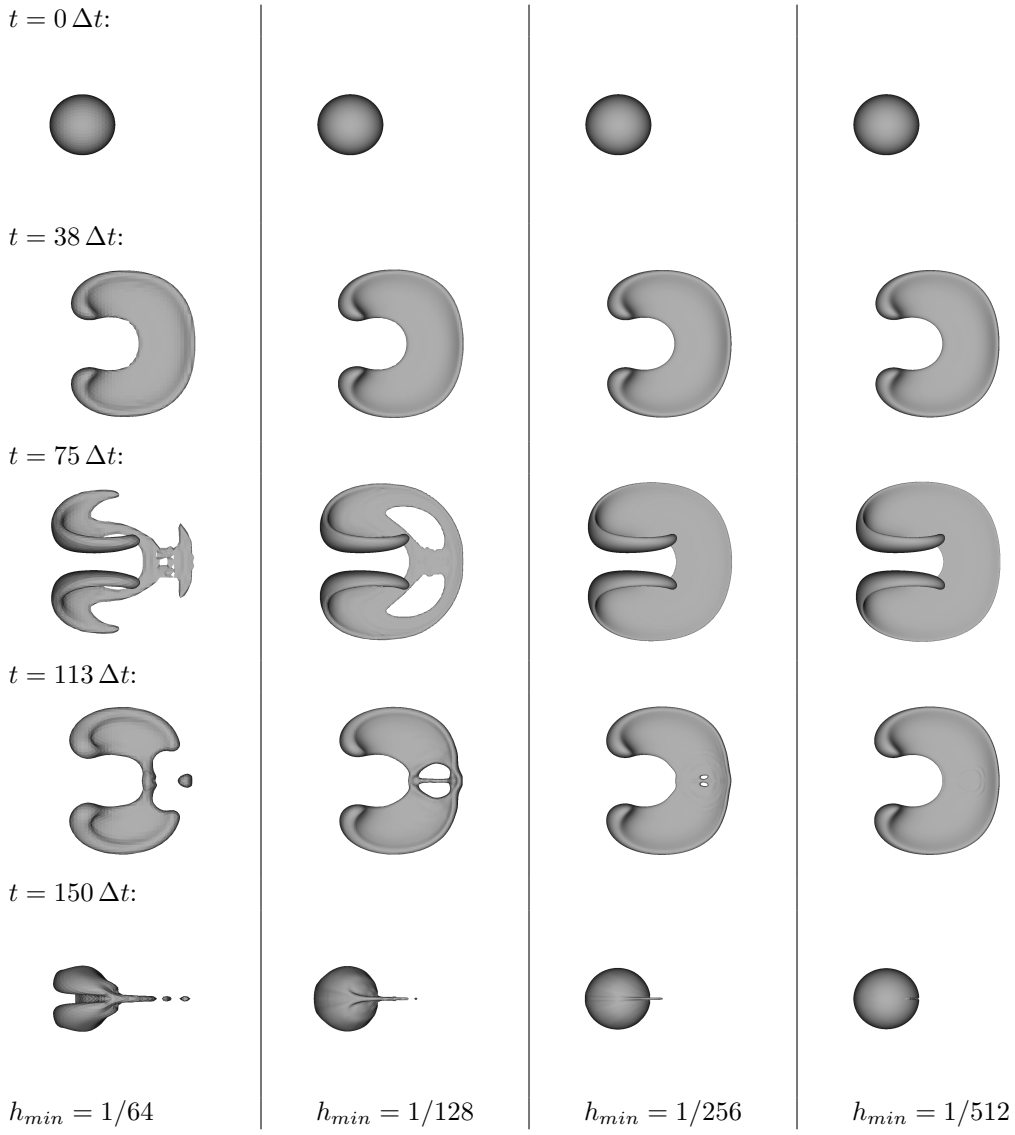

FIG. 4.6. *Zero level surfaces for the Enright test at different time steps and different dynamic grid resolutions. The new BFECC method with high order interpolation and limiting.*

[7] C. R. ETHIER AND D. STEINMAN, *Exact fully 3D Navier–Stokes solutions for benchmarking*, International Journal for Numerical Methods in Fluids, 19 (1994), pp. 369–375.

[8] M. FALCONE AND R. FERRETTI, *Convergence analysis for a class of high-order semi-Lagrangian advection schemes*, SIAM Journal on Numerical Analysis, 35 (1998), pp. 909–940.

[9] J. E. FLAHERTY, R. M. LOY, M. S. SHEPHARD, B. K. SZYMANSKI, J. D. TERESCO, AND L. H. ZIANTZ, *Adaptive local refinement with octree load balancing for the parallel solution of three-dimensional conservation laws*, Journal of Parallel and Distributed Computing, 47 (1997), pp. 139–152.

[10] J. FRST, *A third order wlsqr scheme on unstructured meshes with curvilinear boundaries*, in Numerical Mathematics and Advanced Applications, K. Kunisch, G. Of, and O. Steinbach, eds., Springer Berlin Heidelberg, 2008, pp. 289–296.

[11] D. FUSTER, G. AGBAGLAH, C. JOSSERAND, S. POPINET, AND S. ZALESKI, *Numerical simulation of droplets, bubbles and waves: state of the art*, Fluid Dyn. Res., 41 (2009), p. 065001.

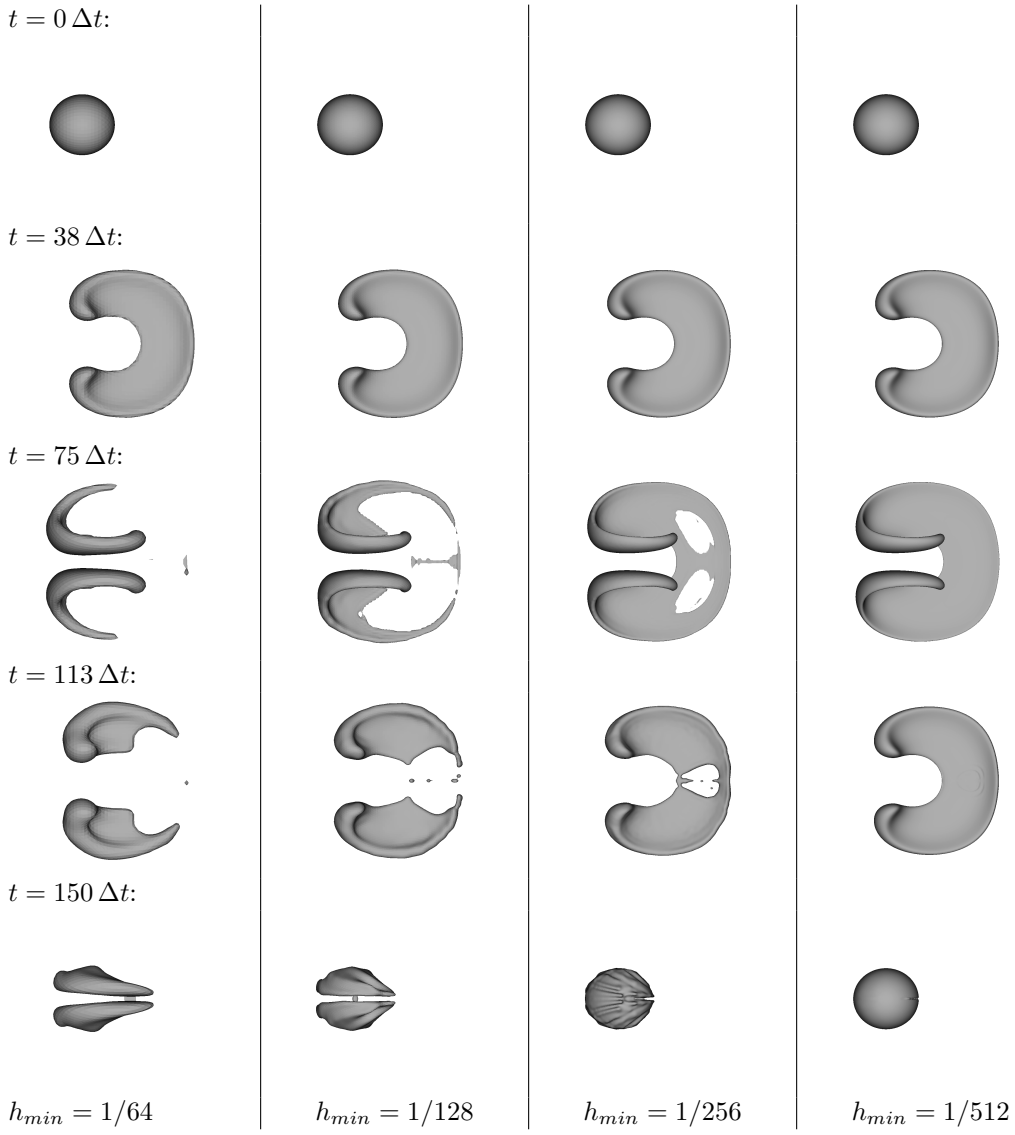

FIG. 4.7. *Zero level surfaces for the Enright test at different time steps and different dynamic grid resolutions. Semi-Lagrangian method with linear interpolation.*

[12] S. GOTTLIEB AND C.-W. SHU, *Total variation diminishing Runge-Kutta schemes*, Mathematics of Computation, 67 (1998), pp. 73–85.
[13] S. GROSS AND A. REUSKEN, *Numerical methods for two-phase incompressible flows*, vol. 40, Springer Science & Business Media, 2011.
[14] L. HU, Y. LI, AND Y. LIU, *A limiting strategy for the back and forth error compensation and correction method for solving advection equations*, Mathematics of Computation, (2016).
[15] R. J. LEVEQUE, *High-resolution conservative algorithms for advection in incompressible flow*, SIAM Journal on Numerical Analysis, 33 (1996), pp. 627–665.
[16] F. LOSASSO, R. FEDKIW, AND S. OSHER, *Spatially adaptive techniques for level set methods and incompressible flow*, Computers & Fluids, 35 (2006), pp. 995–1010.
[17] F. LOSASSO, F. GIBOU, AND R. FEDKIW, *Simulating water and smoke with an octree data structure*, ACM Transactions on Graphics (TOG), 23 (2004).
[18] K. MAHADY, S. AFKHAMI, AND L. KONDIC, *On the influence of initial geometry on the evolution*

of fluid filaments, Physics of Fluids, 27 (2015), p. 092104.

[19] C. Min and F. Gibou, *A second order accurate level set method on non-graded adaptive Cartesian grids*, J. Comput. Phys., 225 (2007), pp. 300–321.

[20] S. M. Murman, *Compact upwind schemes on adaptive octrees*, J. Comput. Phys., 229 (2010), pp. 167–1180.

[21] K. Nikitin, M. Olshanskii, K. Terekhov, and Y. Vassilevski, *A splitting method for numerical simulation of free surface flows of incompressible fluids with surface tension*, Comput. Methods Appl. Math, 15 (2015), pp. 59 – 78.

[22] K. Nikitin and Y. V. Vassilevski, *Free surface flow modelling on dynamically refined hexahedral meshes*, Rus. J. Numer. Anal. Math. Model., 23 (2008), pp. 469–485.

[23] K. D. Nikitin, M. A. Olshanskii, K. M. Terekhov, and Y. V. Vassilevski, *A numerical method for the simulation of free surface flows of viscoplastic fluid in 3D*, J. Comput. Math., 29 (2011), pp. 605–622.

[24] R. Nourgaliev, S. Wiri, N. Dinh, and T. Theofanous, *On improving mass conservation of level set by reducing spatial discretization errors*, International journal of multiphase flow, 31 (2005), pp. 1329–1336.

[25] M. A. Olshanskii, K. M. Terekhov, and Y. V. Vassilevski, *An octree-based solver for the incompressible NavierStokes equations with enhanced stability and low dissipation*, Computers & Fluids, 84 (2013), pp. 231 – 246.

[26] S. Popinet, *Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries*, Journal of Computational Physics, 190 (2003), pp. 572 – 600.

[27] S. Popinet, *An accurate adaptive solver for surface-tension-driven interfacial flows*, J. Comput. Phys., 228 (2009), pp. 5838–5866.

[28] J.-M. Qiu and A. Christlieb, *A conservative high order semi-Lagrangian weno method for the Vlasov equation*, Journal of Computational Physics, 229 (2010), pp. 1130–1149.

[29] J.-F. Remacle, J. E. Flaherty, and M. S. Shephard, *An adaptive discontinuous Galerkin technique with an orthogonal basis applied to compressible flow problems*, SIAM Review, 45 (2003), pp. 53–72.

[30] A. Selle, R. Fedkiw, B. Kim, Y. Liu, and J. Rossignac, *An unconditionally stable MacCormack method*, Journal of Scientific Computing, 35 (2008), pp. 350–371.

[31] C.-W. Shu and S. Osher, *Efficient implementation of essentially non-oscillatory shock-capturing schemes*, Journal of Computational Physics, 77 (1988), pp. 439–471.

[32] V. Sochnikov and S. Efrima, *Level set calculations of the evolution of boundaries on a dynamically adaptive grid*, Int. J. Numer. Meth. Engng., 56 (2003), pp. 1913–1929.

[33] J. Strain, *Tree methods for moving interfaces*, J. Comput. Phys., 151 (1999), pp. 616–648.

[34] Y. Wang, S. Simakhina, and M. Sussman, *A hybrid level set-volume constraint method for incompressible two-phase flow*, Journal of Computational Physics, 231 (2012), pp. 6438 – 6471.

[35] D. Xiu and G. E. Karniadakis, *A semi-Lagrangian high-order method for Navier–Stokes equations*, Journal of Computational Physics, 172 (2001), pp. 658–684.

[36] V. Zingan, J.-L. Guermond, J. Morel, and B. Popov, *Implementation of the entropy viscosity method with the discontinuous Galerkin method*, Computer Methods in Applied Mechanics and Engineering, 253 (2013), pp. 479 – 490.