Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 212 (2006) 338–357
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

A quadrature-free discontinuous Galerkin method for the level set equation

Emilie Marchandise [a,c,*], Jean-François Remacle [a,b], Nicolas Chevaugeon [a]

[a] Department of Civil Engineering, Université catholique de Louvain (UCL), Place du Levant 1, 1348 Louvain-la-Neuve, Belgium
[b] Center for Systems Engineering and Applied Mechanics (CESAME), Université catholique de Louvain (UCL), 1348 Louvain-la-Neuve, Belgium
[c] Fonds National de la Recherche Scientifique, rue d'Egmont 5, 1000 Bruxelles, Belgium

Received 18 January 2005; received in revised form 4 July 2005; accepted 5 July 2005
Available online 19 August 2005

Abstract

A quadrature free, Runge–Kutta discontinuous Galerkin method (QF-RK-DGM) is developed to solve the level set equation written in a conservative form on two- and tri-dimensional unstructured grids. We show that the DGM implementation of the level set approach brings a lot of additional benefits as compared to traditional ENO level set realizations. Some examples of computations are provided that demonstrate the high order of accuracy and the computational efficiency of the method.

Keywords: Level set; Runge–Kutta discontinuous Galerkin method

1. Introduction

In multiphase flow simulations, when it comes to the computation of interface motions, two types of approaches are of common use: interface tracking and interface capturing methods. The main difference between the methods is that interface tracking methods are Lagrangian, i.e., the mesh explicitly represents the interface while interface capturing methods are Eulerian, i.e., the interface is an implicit function defined on a fixed mesh.

The *interface tracking* techniques use a deforming mesh that conforms to the interface (arbitrary Lagrangian–Eulerian (ALE) methods [1–4]), or explicitly track the interface (MAC methods [5]). In MAC methods,

* Corresponding author. Tel.: +3210472061; fax: +3210472179.
E-mail address: marchandise@gce.ucl.ac.be (E. Marchandise).

0021-9991/$ - see front matter
doi:10.1016/j.jcp.2005.07.006

Lagrangian particles are advected by the flow and the distribution of particles identifies the regions occupied by a given fluid. Those methods are very accurate and very efficient for flexible moving boundaries with small deformations. However, they are difficult to use in cases where the interface reconnect or disconnect. In addition, significant re-meshing is needed when large deformations of the interface occur.

In *interface capturing* methods, some kind of auxiliary function is needed. Those methods are very robust and have a wide range of applicability. However, they usually require higher mesh resolution. Examples of interface capturing methods are volume of fluid methods (VOF) and level set methods. VOF methods [6,7] are based on the resolution of a conservation law for the volume fraction of fluid. Those methods have excellent conservation properties. However, VOF methods are usually only first or second order methods and the geometric properties of the interface like normals or curvatures are difficult to compute accurately. Besides, the reconstructed interface is discontinuous between cells. The level set method, initially proposed by Osher and Sethian [8], potentially addresses VOF issues.

The level set method has been used in a wide variety of problems such as crack propagation [9,10], flame fronts in reacting flows [11], photolithography [12], analysis of complex micro-structures [13] and multiphase flows.

Application of level sets in multiphase flow calculations has been extensively described by Sussman et al. [14–16]. The level set function is able to represent an arbitrary number of bubbles or drop interfaces and complex changes of topology are naturally taken into account by the method.

The level set function $\phi(\vec{x},t)$ is defined to be a smooth function that is positive in one region and negative in the other. The implicit surface $\phi(\vec{x},t)=0$ represents the current position of an interface. This interface is advected by a vector field $\mathbf{u}(\vec{x},t)$ that is, in case of multiphase flows, the solution of the Navier–Stokes equations. The elementary advection equation for interface evolution is:

$$\partial_t\phi+\mathbf{u}\cdot\nabla\phi=0. \tag{1}$$

This *level set equation* is a first order hyperbolic PDE.

Geometrical information about the interface (represented by $\phi=0$), e.g., normals and curvature, can be easily obtained from ϕ. The unit outward normal N and the curvature κ are defined by:

$$\vec{N}=\frac{\nabla\phi}{|\nabla\phi|},\quad \kappa=\operatorname{div}\vec{N}.$$

There are several techniques to solve the level set equation (1) in space and time. One of the most popular is the high order Hamilton–Jacobi ENO method (HJ-ENO) combined with a TVD-RK method. It has been mainly developed by [17,18] and used by several authors [14,15,19].

Despite the high order accurate approximation of the level set equation, those numerical methods may become unstable when the level set function develops steep or flat gradients. One solution is to reshape the level set function to a distance function. This method, called reinitialization, has been shown to stabilize those weakly unstable numerical methods [20]. Unfortunately, one of the major drawbacks of the reinitialization process is the difficulty in not moving the original location of the interface, often leading to breakdown in the conservation of mass. To overcome this problem of mass loss with the level set method, various solutions have been proposed:

- coupling the level set with a Lagrangian method: particle level set methods [19],
- coupling the level set with a volume of fluid method [21,22],
- mass correction procedures [14–16],
- adaptive tree methods [23].

However, the simplicity of the original level set method is lost. Those hybrid methods might be either complex to implement or to combine with fluid solvers in order to compute multiphase flow applications.

Since a decade, the Runge–Kutta discontinuous Galerkin method (RK-DGM) has become a very popular numerical technique for solving hyperbolic conservation laws [24–26]. Its popularity is mainly due to the fact it is highly accurate, compact, robust, trivially parallelizable [27], and that it can easily handle complex geometries.

In this work, we develop a RK-DGM method for solving the level set equation written in a conservative form. We show that the method is able to solve a variety of complex problems with higher accuracy than ENO and WENO schemes for smaller meshes an lesser computational time. Besides its very high accuracy, the other advantage of the method is its ease-of implementation, even in parallel.

With our conservative formulation of the level set equation and our DG implementation, we do not need to compute gradients of the level set and our numerical scheme is stable even if ϕ diverges from a signed distance function. It follows that within this framework, we do not need to reinitialize the level set at all.

The outline of the paper is as follows: we start by describing the level set method and discuss the need to reinitialize the level set to a distance function in Section 2. Section 3 gives a description of the RK-DGM method and Section 4 provides several computational examples.

2. The level set and the distance function

When a signed distance function ϕ is advected by a non-uniform flow, it does not necessarily correspond to a distance function any longer. We detail here why it is sometimes mandatory to ensure that the levelset remains a distance function.

In [28], it was pointed out that certain flow fields can generate large or small gradients of the level set function ϕ which can cause instability problems. As a consequence, researchers have focused on methods that forbids ϕ from developing large or small gradients. They have chosen to force ϕ to approximate a signed distance function: $\|\nabla\phi\| = 1$. This choice can also be useful for multiphase flow applications.

Indeed, one of the difficulty when solving the Navier–Stokes equation for the multiphase fluid motion is that the fluid variables such as the density ρ and the viscosity μ are discontinuous across the interface. Besides, for a curved interface, one has to take into account surface tensions that create a jump in pressure across the interface. As those large jumps may cause numerical difficulties, some authors have proposed to introduce an interface thickness to smooth the density and viscosity at the interface. This interface thickness is also important when the surface tension forces are rather distributed over a number of grid cells than integrated along the interface. In this context, it is important to keep a uniform interface thickness in time. And, to keep a uniform thickness, the level set function ϕ must remain a distance function at the interface. Similarly, in the case of the extended finite element method, some of the enrichment functions (e.g., the crack tip enrichment function [9]) are explicitly depending on the distance to the interface.

There are many different approaches to reinitialize ϕ as a distance function:

- Fast marching methods [29,30] solve the Eikonal equation $\|\nabla\phi\| = 1$ over the whole computational domain in $\mathcal{O}(N \log N)$ operations, where N is the number of grid points in the domain. In this method a binary tree data structure is created that allows one to march through cells in a special order that allows only one pass per cell. Those methods are designed not to move the interface as compared with the next two methods presented.
- Solving in pseudo-time the first order partial differential equation:

$$\phi_t + S(\phi_0)(\|\nabla\phi\| - 1) = 0, \tag{2}$$

where $S(\phi_0)$ is the smoothed sign function $S(\phi_0) = \phi_0/\sqrt{\phi_0^2 + \alpha^2}$ and α is a parameter of amplitude that is a function of the grid size. The principal issue when solving this equation is that the zero level set, i.e.,

the position of the interface, might be shifted due to numerical diffusion of the underling numerical scheme. Mass errors can then occur. Eq. (2) can also be reformulated in a classical Hamilton–Jacobi form:

$$\phi_t + v(\phi) \cdot \nabla\phi = S(\phi_0) \quad \text{with } v(\phi) = S(\phi_0)\frac{\nabla\phi}{\|\nabla\phi\|}. \tag{3}$$

Solving (3) is complex and, again, it does not guarantee that the zero level set does not move. To circumvent this, Sussman and Fatemi [16] introduced a constraint to the reinitialization algorithm to prevent changing the position of the zero level set when solving (3) numerically.

- Tree-based algorithms for computing closest points, distance transforms [31–33]. Several efficient algorithms exist and are available on the Internet as open source codes (see, for example, ANN [34] and closest point transform [35]. In order to use them in a level set approach, one has to compute a set of points on the zero level set.

Those reinitialization methods might be complex to implement and expensive in c.p.u time. In the context of multiphase flow problems, reinitialization methods are only mandatory in the following cases:

- for stability reasons: if the numerical method is unstable when ϕ diverges from a signed distance function,
- if one choose to define a constant interface thickness in time to smooth the discontinuities across the interface.

In this work, we take the point of view of choosing a stable and accurate numerical scheme for doing the level set update. Therefore, we should not have to perform any reinitialization. For our future multiphase flow application, we will rather use the "discontinuous integration" approach for the discontinuous variables and compute the surface tension forces as line integrals [36,37].

3. The discontinuous Galerkin method

In this section, we will first derive the quadrature-free DG formulation, then highlight some dispersive and dissipative properties of the DGM and finally describe our explicit Runge–Kutta method.

3.1. Discretization in space: quadrature-free formulation

We can rewrite the level set equation (1) in a *conservative form*:

$$\partial_t\phi + \nabla \cdot (\mathbf{u}\phi) = \phi\nabla \cdot \mathbf{u}. \tag{4}$$

Since we want to deal with incompressible flows, we have that $\nabla \cdot \mathbf{u} = 0$ and the conservation law simplifies in

$$\partial_t\phi + \nabla \cdot (\mathbf{u}\phi) = 0. \tag{5}$$

Within this framework, only possible for incompressible flow, we do not need to evaluate $\nabla\phi$ anymore. Remember that poor accuracy of the evaluation of $\nabla\phi$ was leading to instabilities in the absence of renormalization for the ENO method cited above. Our formulation, in case of incompressible flow, has the advantage to allow us to avoid the cost of the renormalization step.

Next, we multiply Eq. (5) by a test function $\hat{\phi}$, integrate over the domain Ω and use the divergence theorem to obtain the following variational formulation:

$$\int_\Omega \partial_t \phi \hat{\phi}\, \mathrm{d}v = \int_\Omega u_x \phi \partial_x \hat{\phi}\, \mathrm{d}v + \int_\Omega u_y \phi \partial_y \hat{\phi}\, \mathrm{d}v + \int_\Omega u_z \phi \partial_z \hat{\phi}\, \mathrm{d}v - \int_\Omega f \hat{\phi}\, \mathrm{d}s \quad \forall \hat{\phi}, \tag{6}$$

where $f(\phi) = \phi \mathbf{u} \cdot \vec{n}$ is the normal trace of the fluxes and $\mathbf{u} = (u_x, u_y, u_z)$ is the velocity field. Since the DGM allows discontinuities at the interface, the flux is not uniquely determined on it and a flux formula has to be supplied to complete the discretization process. In the simple advection case, we choose the *upwind flux* to be the value of ϕ on $\partial\,\Omega$. Let $\vec{x}$ be a point of $\partial\Omega$. We define the upwind value ϕ^{UP} as the trace of ϕ on $\partial\Omega$ (see Fig. 1):

- $\phi^{\mathrm{UP}} = \phi^+$ if $\mathbf{u} \cdot \vec{n} \leqslant 0$, i.e., the flow goes inside the domain,
- $\phi^{\mathrm{UP}} = \phi^-$ if $\mathbf{u} \cdot \vec{n} > 0$, i.e., the flow goes outside the domain.

The physical domain Ω is discretized into a collection of $\mathcal{N}_e$ elements

$$\mathscr{T} = \bigcup_{e=1}^{\mathcal{N}_e} e \tag{7}$$

called a mesh. In each element e of $\mathscr{T}$, ϕ is discretized using a finite expansion. It is usually polynomial spaces that are chosen for approximating the ϕ. It is common, in the finite element world, to distinguish reference coordinates ξ, η, ζ and real coordinates x, y, z. We use piecewise continuous approximations on each element

$$\phi(\xi, \eta, \zeta) = \sum_{k=1}^{d} N_k(\xi, \eta, \zeta) \phi_k^e. \tag{8}$$

We see that, in (8), the coefficients ϕ_k^e of the interpolation are associated to the element. Interpolation are then discontinuous at inter-element boundaries, contrary to classical finite element methods. The unknown coefficients ϕ_k^e may be represented, in the memory of a computer, as a double-precision matrix of size $\mathcal{N}_e \times d$.

The problem (5) that we aim to solve is a linear hyperbolic PDE with non-constant coefficients if the velocity field $\mathbf{u}$ is non-uniform. The following very important assumption is done for treating non-linearities. If $F(\phi)$ is a function of the unknown ϕ, e.g., $F = u_x(x,y,z)\phi$, we apply the following rule to compute F:

$$F(\phi) = F\left(\sum_{k=1}^{d} N_k \phi_k^e\right) \simeq \sum_{k=1}^{d} N_k F(\phi_k^e). \tag{9}$$

The general idea of assumption (9) is that any function is approximated on the "same grid as ϕ". For example, if $F = \phi^2$ and if $\phi^e = \sum_k N_k \phi_k^e$ is in $\mathbb{P}^p$, then $(\phi^e)^2 = (\sum_k N_k \phi_k^e)^2$ is in $\mathbb{P}^{2p}$. With assumption (9), $\phi^2 \simeq \sum_k N_k (\phi_k^e)^2 \in \mathbb{P}^p$. Eq. (9) is, of course, only exact for linear functions F.

In this work, we use Lagrange shape functions for interpolating ϕ. This choice is certainly not the only one available. Orthogonal shape functions [25], for example, have specific interest. Here, we have d

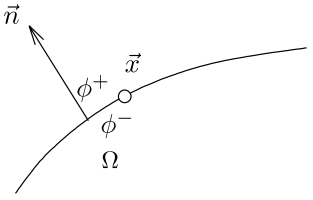

Fig. 1.

Lagrange points in each element e. We note F_k^e for the value of a given function F at node k of element e. We have therefore[1]

$$F(\phi) = \sum_k F_k^e N_k = F_k^e N_k.$$

The interpolations being disconnected, it is possible to write (6) for each element e of $\mathscr{T}$ as

$$\partial_t \phi_k^e \int_e N_k N_j \, \mathrm{d}v = (u_x\phi)_k^e \int_e N_k \partial_x N_j \, \mathrm{d}v + (u_y\phi)_k^e \int_e N_k \partial_y N_j \, \mathrm{d}v + (u_z\phi)_k^e \int_e N_k \partial_z N_j \, \mathrm{d}v - \sum_{l=1}^{n_e} \underbrace{\mathbf{u}\cdot\vec{n}\phi_k^{UP}}_{f_k^{e,l}} \int_{\partial e_l} N_k N_j \, \mathrm{d}s = 0 \quad \forall i,j, \tag{10}$$

where we have decomposed the boundary ∂e of element e into n_e parts ∂e_l corresponding, in 3D, to the four triangular faces of the tetrahedron. In (10), $f_k^{e,l}$ is the upwind value of f on node k of boundary l of element e.

We further assume that the Jacobian matrix

$$J = \begin{bmatrix} \partial_\xi x & \partial_\eta x & \partial_\zeta x \\ \partial_\xi y & \partial_\eta y & \partial_\zeta y \\ \partial_\xi z & \partial_\eta z & \partial_\zeta z \end{bmatrix}$$

is constant for any element e. This is true if all edges of the mesh are straight sided. We note $\|e\| = \det J$. Some important DGM operators appear in (10). We define two mass matrix operators, one relative to element e

$$M_{ij}^e = \int_e N_i N_j \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z = \int_e N_i N_j \, \mathrm{d}\xi \, \mathrm{d}\eta \, \mathrm{d}\zeta \|e\| = M_{ij}\|e\|$$

and one relative to element boundaries ∂e_k

$$M_{ij}^{\partial e_l} = \int_{\partial e_l} N_i N_j \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z = \int_{\partial e_l} N_i N_j \, \mathrm{d}\xi \, \mathrm{d}\eta \, \mathrm{d}\zeta \|\partial e_l\| = M_{ij}^l \|\partial e_k\|.$$

M_{ij} and M_{ij}^l are constant matrices, independent of e, and of size $d \times d$. We finally define three derivatives operators

$$D_{ij}^\xi = \int_e N_i(\partial_\xi N_j) \, \mathrm{d}\xi \, \mathrm{d}\eta \, \mathrm{d}\zeta, \quad D_{ij}^\eta = \int_e N_i(\partial_\eta N_j) \, \mathrm{d}\xi \, \mathrm{d}\eta \, \mathrm{d}\zeta, \quad D_{ij}^\zeta = \int_e N_i(\partial_\zeta N_j) \, \mathrm{d}\xi \, \mathrm{d}\eta \, \mathrm{d}\zeta.$$

Those operators are square matrices of size $d \times d$. They are all independent of e.

The choice of a specific family of shape functions makes the structure of some DGM operators computationally interesting. For Lagrange shape functions, boundary operators M_{ij}^l have lots of zeros. In case of orthogonal polynomials, derivative operators are upper triangular matrices.

Note that

$$D_{ij}^x = \int_e N_i(\partial_x N_j) \, \mathrm{d}x \, \mathrm{d}y \, \mathrm{d}z = \|e\| \left(D_{ij}^\xi J_{11}^{-1} + D_{ij}^\eta J_{12}^{-1} + D_{ij}^\zeta J_{13}^{-1} \right).$$

Thanks to previous definitions, the DGM formulation is written, for element e, as

[1] Starting here, we use the Einstein summation rule for repeated indices.

$$\partial_t \phi_k^e M_{kj} = (v_\xi \phi)_k^e D_{kj}^\xi + (v_\eta \phi)_k^e D_{kj}^\eta + (v_\zeta \phi)_k^e D_{kj}^\zeta - \frac{1}{\|e\|} \sum_{l=1}^{n_e} \|\partial e_l\| f_k^{e,l} M_{kj}^l. \tag{11}$$

In (11), we have computed fluxes in the reference system of coordinates, i.e.,

$$\begin{aligned}
(v_\xi \phi)_k^e &= (u_x \phi)_k^e J_{11}^{-1} + (u_y \phi)_k^e J_{21}^{-1} + (u_z \phi)_k^e J_{31}^{-1}, \\
(v_\eta \phi)_k^e &= (u_x \phi)_k^e J_{12}^{-1} + (u_y \phi)_k^e J_{22}^{-1} + (u_z \phi)_k^e J_{32}^{-1}, \\
(v_\zeta \phi)_k^e &= (u_x \phi)_k^e J_{13}^{-1} + (u_y \phi)_k^e J_{23}^{-1} + (u_z \phi)_k^e J_{33}^{-1}.
\end{aligned}$$

Formulation (11) is a quadrature-free version of the DGM. An efficient way to implement (11) on a computer is to use basic linear algebra subroutines (BLAS). In particular, (11) involves three large matrix–matrix multiplications ($R_{ej}^\xi = (v_\xi \phi)_k^e D_{kj}^\xi$). Level 3 BLAS (BLAS3) subroutines can be advantageously used.

Typically, for the DGM of order $p = 4$ on hexahedra, we have $d = (p+1)^3 = 125$ Lagrange shape functions for each element and the matrixes of problem (11) have a size 125×125. Fig. 2 shows us that from those size of matrixes, the Matrix Kernel Library of Intel is able to make a classical desktop computer deliver about 4 GFLOPS.

The ATLAS open source optimized BLAS library [38] delivers substantially less GFLOPS for the same d but is usually more efficient for small matrix sizes. Note that the NETLIB BLAS library provided in our SUSE-LINUX distribution was giving performances around 1 GFLOP, i.e., a DGM code that runs 4 times slower!

3.2. Spectral super-convergence of the DGM: dispersive and dissipative properties

The term "wave number" refers to the number of complete wave cycles that exist in one meter of linear space. If h is the mesh size and k is the dimensional wave number, we define a non-dimensional wave number as $k_h = kh$. The non-dimensional wave number kh is interpreted as a wave number where the length measure is taken as the mesh size h. For example, a non-dimensional wave number of $kh = 1/5$ corresponds to a wave length $5h$, i.e., of five element sizes. The important issue we aim to address here is the following. Considering a mesh of spacing h and polynomials interpolant of order p, what are scales that are correctly

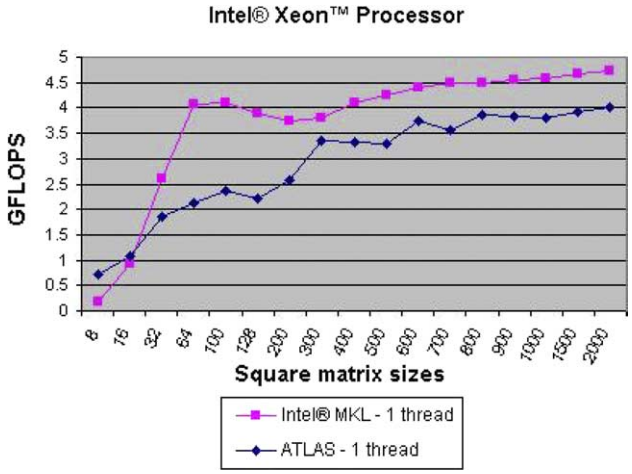

Fig. 2. Double-precision general matrix–matrix multiply (DGEMM) performance test. Comparison of DGEMM performance of Intel MKL 7.0 vs. ATLAS. Reprinted from INTEL web site.

approximated by our DGM discretization. In other words, what is the range of wave numbers that are accurately approximated.

The diffusive and dispersive properties of high order discontinuous Galerkin method have been studied extensively by Hu and Atkins [39] and, more recently, by Ainsworth [40]. In [40], Ainsworth shows that there exists three separate regimes (convergence rates) of error that correspond to ranges of non-dimensional wave numbers (kh).[2]

The first range of wave numbers is the one of *resolved wave numbers*. It corresponds to relatively small kh, i.e., kh that are such that $2p+1 > (2\pi kh) + \mathcal{O}(kh)^{1/3}$. Resolved numerical wave numbers are super-convergent, i.e., the relative error (the difference between the exact wave number and the numerical one) converges very rapidly:

$$\begin{aligned} &\text{dispersion error}: \quad \mathcal{O}((hk)^{2p+3}),\\ &\text{dissipation error}: \quad \mathcal{O}((hk)^{2p+2}). \end{aligned} \tag{12}$$

Typically, the real part of the numerical kh that correspond to diffusion (or instabilities) is the one that dominates in the resolved range. In DGM, resolved waves are computed with a super-accuracy. With such a super-convergence rate, p should be chosen to be (one of) the smallest p that falls into the super convergence range. All higher p will lead to neglectible gains in the absolute accuracy but will lead to substantial increase in computational cost.

At the opposite extreme, for large wave numbers, when kh is much bigger than p (more precisely when $2p+1 < (2\pi kh) - \mathcal{O}(kh)^{1/3}$), there is a high relative error (high damping, i.e., big imaginary part in the numerical kh and typically $\mathcal{O}(100\%)$ of error on the real part of kh). If the RK-DGM has to convect a wave corresponding to a large kh, our RK time stepping scheme will damp the wave very quickly. This is the feature of DGM that gives it a key advantage to spectral methods e.g. The DGM scheme is auto-adaptive: it self filters unresolved modes while producing super-accuracy on resolved ones. The numerical dissipation of the DGM schemes acts like an adaptive filter.

The transition between those two extremes occurs when the order p lies in the following narrow range: $(2\pi kh) - \mathcal{O}(kh)^{1/3} < 2p+1 < (2\pi kh) + \mathcal{O}(kh)^{1/3}$. In this region, the error is of order of 100% but decreases at an algebraic rate $(p)^{-1/3}$.

Fig. 3 shows the DG numerical error in the real and imaginary parts of the numerical wave numbers vs. kh. We see that the super convergence is observed below a certain wave number that depends on p. Here, we can decide that we aim to limit the error to 10^{-4}. Therefore, Fig. 3 shows that it is necessary to use at least 10 elements per wavelength for $p=2$, 6 elements per wavelength for $p=4$ and 2 elements per wavelength for $p=6$. As it was previously stated, the number of resolved modes grows roughly like $2p+1$. With 10 elements per wavelength, we have 1 resolved mode for $p=2$, 10/6 resolved modes for $p=4$ and 5 resolved modes for $p=6$. In theory, we should have 9/5 more resolved mode for $p=4$ than for $p=2$ and 13/5 more resolved mode for $p=6$ than for $p=2$.

Fig. 4 shows the DGM eigenmodes in the space domain relative to several resolved and non-resolved wave numbers. It is very interesting to see that resolved modes are smooth functions without inter-element jumps. Non-resolved modes have jumps that are getting larger while kh is increasing. The numerical damping of the DGM is known to be contained in the inter-element jumps.

3.3. Discretization in time: Runge–Kutta method

The time stepping scheme used here is a TVD-Runge–Kutta of order $p+1$. The combination RK of order $p+1$ with DG at order p can be proven [41] to be stable under the CFL condition

[2] In Aisworth's paper, wave numbers are defined differently, i.e., they refer to the number of cycles in 2π meters of linear space.

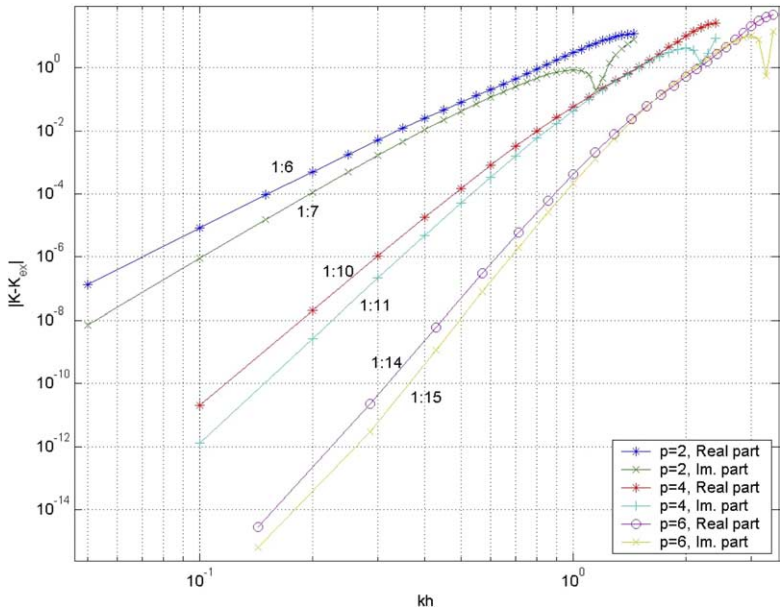

Fig. 3. Plot of the error on the real and imaginary parts of numerical wave numbers vs. kh. This is the case of a uniformly discretized 1D DGM operator applied to an advection equation. The mesh was composed of 20 equally spaced 1D elements for $p = 2$, 10 elements for $p = 4$ and 8 elements for $p = 6$.

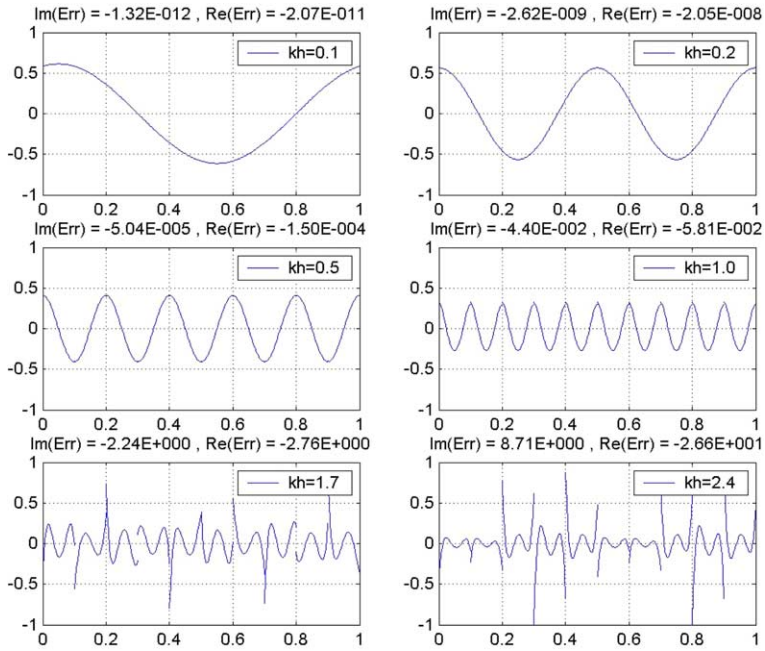

Fig. 4. Plot of the numerical eigenmodes of the DGM operator. The mesh was composed of 10 equally spaced 1D $p = 4$ elements. On top of each figure, we give both real and imaginary parts of the error on the numerical wave numbers.

$$\Delta t < \frac{h}{c(2p+1)}, \tag{13}$$

where h is the mesh size of element e and $c = |\mathbf{u}|$ is the maximum velocity on e. Note that this bound is a weak one when p is high. In other words, the RK$(2p+1)$ stability range grows faster than the spectral radius of the DGM amplification matrix divided by $2p+1$. Table 1 compares, in the 1D case, the spectral radius of the amplification matrix $S(\mathbf{A}_p)$ of the DGM(p) divided by $c(2p+1)$ with the corresponding stability limit of RK(p + 1). Of course, stability requires that, if we choose the proposed CFL condition (13), $S(\mathbf{A}_p)/(c(2p+1)) \leqslant RK(p+1)$. In the table, we see that the proposed CFL condition is sharp at low orders and gets sub-optimal when p increases.

The overall cost of our RK-DGM scheme involves some matrix–matrix multiplications at each evaluation of the spatial operator, $p+1$ of such evaluations for one RK step and a time step that decreases like $1/(2p+1)$. The computational cost for advancing of a fixed time of the RK-DGM is growing like p^6 in 2D and like p^8 in 3D. Because of cache effects, this behavior only happens when p is high enough to hit the peek performance of the CPU.

4. Examples

In our examples, we use unstructured meshes of triangles or quadrilaterals in 2D and of hexahedra or tetrahedra in 3D. We validate our method by computing a variety of interfaces moving with a given velocity, with non-trivial topology and curvature. Our methods were implemented in Standard C++, compiled with GNU g++ v3.3, and run on one CPU of a 2.4 GHz Intel(R) Pentium(R) 4 with 512 KB of L2 cache. The BLAS implementation is the one of the Intel(R) Math Kernel Library.

4.1. Bubbles in a uniform flow

We try here to measure a dissipation error that is relevant in interface capturing computations. If the velocity field $\mathbf{u}$ is incompressible, then areas in 2D and volumes in 3D delimited by the iso-zero level set $\phi = 0$ should be conserved in time. Our first problem consists in the advection of a collection of circular

Table 1
$S(\mathbf{A}_p)$ is the spectral radius of the amplification matrix of the discontinuous Galerkin scheme for a given polynomial order p and $RK(p+1)$ is the norm of the maximal eigenvalue

p	$\frac{S(\mathbf{A}_p)}{c(2p+1)}$	$RK(p+1)$
0	2.00	2.00
1	2.00	2.00
2	2.36	2.57
3	2.74	2.78
4	3.10	3.21
5	3.44	3.55
6	3.78	3.95
7	4.10	4.31
8	4.42	4.70
9	4.72	5.06
10	5.02	5.45
11	5.32	5.82
12	5.62	6.20
13	5.90	6.57

bubbles. The bubbles are advected with a uniform velocity from the left to the right in a computational domain of size 100×10. The mesh is made of regular quadrilaterals of size h. The nine bubbles have a radius of, respectively, $r = 1/2h, 1/3h, 1/4h, 1/5h, 1/6h, 1/7h, 1/8h, 1/9h, 1/10h$. The initial field for the level set is the following:

$$\phi_0 = \begin{cases} -1 & \text{if } d < -1, \\ d & \text{if } -1 < d < 1, \\ 1 & \text{if } d > 1, \end{cases} \tag{14}$$

where d is the distance function to the bubble ($d = x^2 + y^2 - r^2$).

A Fourier analysis of the initial field has given us the wave numbers and the frequency content of each bubble. The main wave numbers are given by $k = (1/\lambda)$ with $\lambda = 4r$. We have respectively $(2\ \pi kh) = 3.14, 4.71, 6.28, 7.85, 9.42, 10.99, 12.56, 14.3, 15.7$. Polynomial order is set to $p = 4$. Fig. 5 shows the bubbles at different time steps.

The DG scheme is not able to represent accurately the advection of the small bubbles (with wave number $(2\pi kh) > 2p + 1 = 9$). The smallest bubble of radius $1/10h$ quickly disappears. Bubbles of intermediate length $r = 1/6h, 1/7h$ that correspond to wave length $(2\pi kh) = 9.42,\ 10.99$ diffuse at a smaller rate (transition region) but are finally damped due to the numerical diffusion of the space discretization. We have done the same computation for different polynomials orders p and plot the mass loss of each bubble (each bubble correspond to a given wave number kh). Fig. 6 shows the area errors calculated by:

$$\epsilon_A = \frac{A(t_f) - A(0)}{A(0)},$$

where $A(t_f)$ is the total area of the bubble at the final time t_f.

For low orders, the limit $2\pi kh = 2p + 1$ is optimistic. For $p = 3$, wave numbers $2\pi kh > 5$ are completely unresolved. For higher p, the situation goes closer to the theoretical limit. For $p = 5, 2\pi\ kh = 10$ is still a resolved mode. Note that Ainsworth considers, in his paper, polynomial orders ranging from $p = 1$ to $p = 200$, which may not be very usable in real life computations.

The dissipation mechanism of the DGM is illustrated in Fig. 7. Unresolved modes contain inter-element discontinuities. They appear when the iso-zero crosses an element boundary (see Fig. 7, time step t_3). When a bubble crosses an interface it is rapidly smeared out. The resulting numerical diffusion is responsible of mass loss.

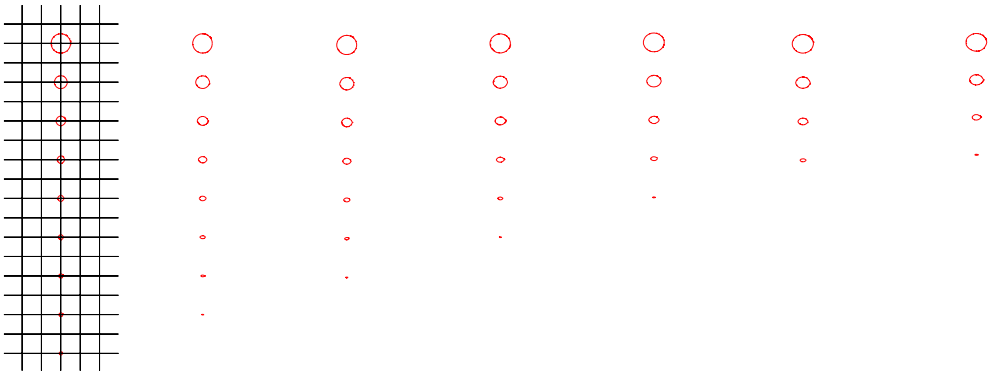

Fig. 5. Visualization of the bubbles (iso-value $\phi = 0$) at different time steps t_i.

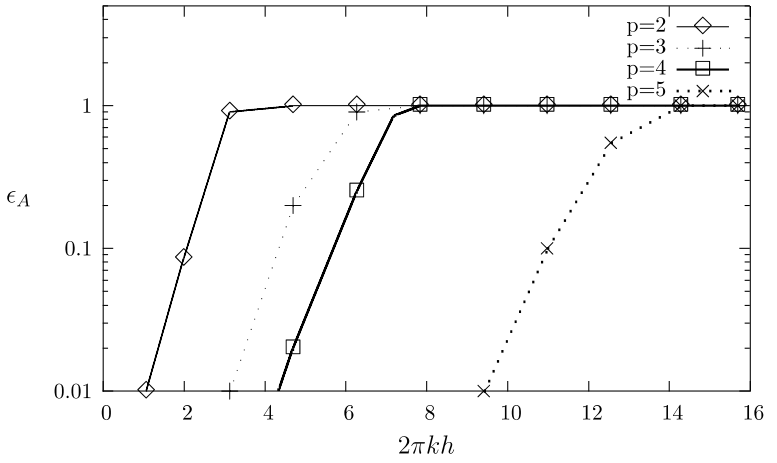

Fig. 6. Dispersion properties: area error versus wavenumber $2\pi kh$ for different wave lengths.

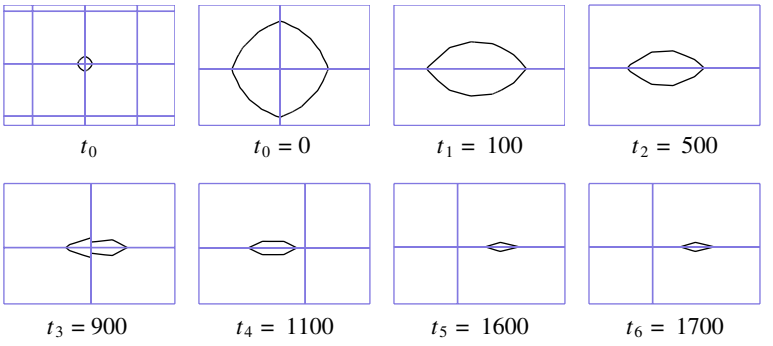

Fig. 7. Visualization of the bubble of radius $1/7h$ after t_i time steps. We see discontinuities of the level set $\phi = 0$ across the interface in this under-resolved region.

From this observation, we establish the following rule. Most of our computations will be done with $p = 4$ and, if the characteristic length (radius of curvature of the level set) r is greater than $h/4$, i.e., $2\pi kh > 2p + 1 = 9$, then our DG scheme should not be the cause of significant dissipation. In other words, one element per wavelength is sufficient for providing super-accuracy with $p = 4$.

4.2. Zalesak's rotating disk

The advection test of Zalesak [42] is often used in the literature to demonstrate the accuracy of an interface-advection algorithm [15,16,19]. This example is able to give some indication on diffusion errors of an interface capturing method. In this example, the initial data is a slotted disk centered at $(50, 75)$ with a radius of 15, a width of 5, and a slot length of 25. The advection is driven by a constant vorticity velocity field:

$$\begin{aligned} u_x &= (\pi/314)(50 - y), \\ u_y &= (\pi/314)(x - 50). \end{aligned} \tag{15}$$

The disk completes one revolution every 628 time units. The computational domain is a circle of radius 50 so that the flow satisfies $\mathbf{u} = (u_x, u_y) = 0$ on the boundaries (see Fig. 8(a)).

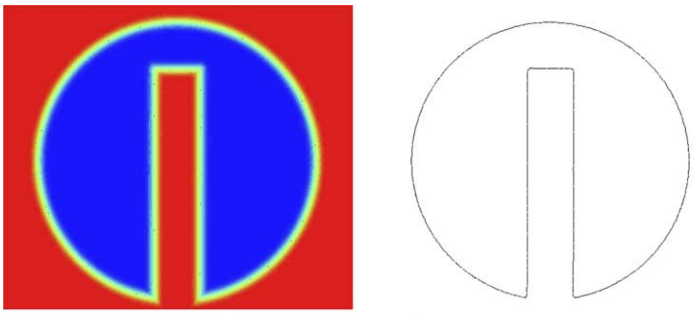

Fig. 8. Initial data for the Zalesak's disk test case (DG TRI(4) h=2): (a) ϕ_0 given as a continuous smooth function, (b) $\phi_0 = 0$, $\phi =$ initially smooth function.

The level set function ϕ is initialized with a smooth exponential function (Fig. 8):

$$\phi_0 = \begin{cases} -1 & \text{if } (\exp(d) - 1) < -1, \\ (\exp(d) - 1) & \text{if } -1 < (\exp(d) - 1) < 1, \\ 1 & \text{if } (\exp(d) - 1) > 1, \end{cases} \tag{16}$$

where d is the signed distance to the notched disk. We have done one complete revolution of the notched disk using triangular meshes of different characteristic lengths h and using different polynomial orders p. To fix the ideas, it took us 3500 time iterations to achieve a complete revolution of the disk ($T = 625$ s), and the computation required 0.983 MB DDR of memory. Table 2 compares the area loss (or gain) of the DG level set method with [15,16,19,43].

In [19,43], a comparison is made between a HJ-WENO(5)/RK(3) level set method with reinitialization and an improved method with particles (particle level set method). In [15,16], a third order ENO/RK(3) scheme is used to discretize the level set and a constrained re-distancing equation.

The area of the notched disk is calculated using a very accurate contouring algorithm [44]. In addition we calculate the accuracy of the interface location using the first order accurate error measure introduced in [16]:

$$\frac{1}{L}\int_{\Omega} \|H_{-}(\phi_{\text{expected}}) - H_{-}(\phi_{\text{computed}})\| \, \mathrm{d}\Omega, \tag{17}$$

Table 2
Zalesak's disk: area loss (or gain) after one revolution

Method	h	Area	% Area loss	L_1 error	CPU time (s)
Exact	–	582.2	–	–	
DG TRI(3)	4.0	580.4	0.30%	0.12	49.27
DG TRI(4)	4.0	583.27	−0.18%	0.05	101.22
DG TRI(5)	4.0	581.01	0.20%	0.04	211.51
DG TRI(3)	2.0	581.22	0.16%	0.019	466.06
DG TRI(4)	2.0	582.17	−0.005%	0.011	955.5
DG TRI(5)	2.0	582.21	−0.0017%	0.008	2040.03
WENO(5)*	1.0	613.0	−5.3%	0.61	–
WENO(5)□	1.0	580.4	0.31%	0.07	134.043
WENO(5)□	0.5	581.7	0.08%	0.031	840.038
ENO(3)◇	1.0	–	–	0.262	–

* Level set [19,43], □ particle level set [19,43] and ◇ level set [15] method.

where L is the perimeter size of the initial interface ($L = 144.29$) and H_- an indicator function given by

$$H_-(\phi) = \begin{cases} 1, & \phi \leqslant 0, \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

Fig. 9 shows the evolution in time of the notched disk (represented by the iso-zero level set) as well as a superposition of the initial solution and the solution after one revolution of the disk. A zoom of the left right corner is very interesting. The solution, at time $t = 0$ s, obtained with a Lagrange interpolation of the initial continuous level set has represented the corner by a quarter of a circle of radius $r \simeq 1/7h$. This corresponds to a wave length of $2\pi kh = 10.99$. Since we use a DGM of order $p = 4$, this high frequency is not resolved ($2\pi kh > 2p + 1 = 9$) and we expect the method to smooth this wave of high frequency into a lower frequency wave that is super-convergent. After one revolution, we indeed see that the corner has been smoothened and transformed into a quarter of circle of radius $r \simeq 1/4h$ which corresponds to a super-convergent wave number. This confirms our established rule that if the radius of curvature of the level set r is greater than $h/4$, then numerical dissipation should be negligible.

4.3. Vortex-in-a-box

This example tests the ability of the scheme to accurately resolve thin filaments on the scale of the mesh which can occur in stretching and tearing flows. We consider the following stream function:

$$\psi = \frac{1}{\pi}\sin^2(\pi x)\sin^2(\pi y) \tag{19}$$

that defines the following velocity field

$$\begin{aligned} u_x &= \frac{\partial \psi}{\partial y} = \sin(2\pi y)\sin^2(\pi x), \\ u_y &= -\frac{\partial \psi}{\partial x} = -\sin(2\pi x)\sin^2(\pi y). \end{aligned} \tag{20}$$

We consider a disk of radius 0.15 placed at $(0.5, 0.75)$ and the distance function $d_0 = (x - 0.5^2) + (y - 0.75)^2 - 0.15^2$.

The computational domain is a square of size $[0,1]\times[0,1]$ and the initial level set is defined as $\phi_0 = d_0$. The flow satisfies $u = (u_x, u_y) = 0$ on the boundaries of the unit square. The resulting velocity field stretches out the circle into a very long, thin fluid element.

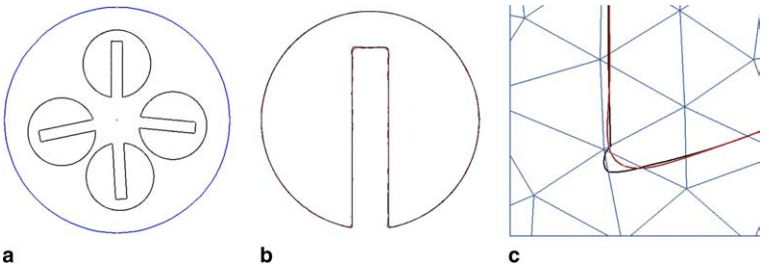

Fig. 9. DG TRI(4) method with $h = 2$. (a) Evolution in the position of the notched disk in time. (b) Comparison of the level set solution at initial time (black) and after one revolution (red). (c) Zoom of figure (b) of the lower right corner on the computational mesh. As expected the corner is slightly smoothed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

In [19], the authors propose to make the problem reversible. After a certain time, function ϕ comes back to its initial state. This is done by multiplying the space velocity function by a periodic function in time $g(t) = \cos(\pi t/T)$. In the time interval $0 \leqslant t \leqslant T$, the initial circle is stretched, the flow slows down and reverses direction in such a way that the initial data should be recovered at time T, see [45]. The reversal period used in the error analysis is $T = 8$. Table 3 reports the computed errors on different meshes and Figs. 10 and 11 show the time evolution of the level set and the iso-zero level set (interface).

It is interesting here to describe briefly the way we treat the high order visualization data accurately. In [44], we provide an efficient and reliable methodology for visualization of high order finite element solutions. The main idea is to use classical h-refinement techniques to provide an optimized visualization grid. In addition to h-refinement, we add some goal oriented visualization filters to the process in order to focus the mesh adaptation to the visualization goals. If, for example, we are only interested in visualizing the iso-zero level set, then only elements that are crossed by it will be considered in the visualization. Fig. 12 shows an example of the technique applied to our vortex problem.

4.4. Three-dimensional deformation field

In [45], Leveque proposed a three-dimensional incompressible flow field which combines a deformation in the x–y plane (Fig. 13(a)) with a similar one in the x–z plane. A sphere of radius 0.15 is placed within a unit computational domain at $(0.35, 0.35, 0.35)$ in a velocity field given by:

Table 3
Vortex-in-box: area loss (or gain) after one period of vortex flow ($T = 8$)

Method	h	Area	% Area loss	L_1 error	CPU time (s)
Exact	–	0.0707	–	–	–
DG TRI(4)	1/64	0.07053	0.24%	$2.5e^{-4}$	8600.21
DG TRI(4)	1/32	0.07130	−0.85%	$2.8e^{-3}$	815.09
DG QUAD(4)	1/32	0.07120	−0.71%	$5.5e^{-4}$	301.83
DG TRI(6)	1/16	0.07188	−1.56%	$5.1e^{-3}$	355.16
WENO(5)*	1/128	0.425	39.8%	$3.1e^{-2}$	–
WENO(5)◇	1/128	0.0702	0.71%	$1.4e^{-3}$	885.94
WENO(5)◇	1/256	0.0705	0.32%	$5.43e^{-4}$	6609.78

* Level set [19,43] and ◇ particle level set [19,43] method.

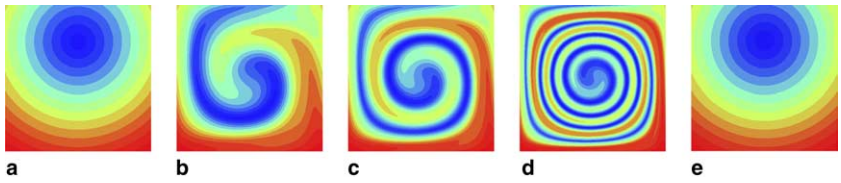

Fig. 10. Time evolution of the distance function in the vortex flow: (a) $t = 0$, (b) $t = 0.9$, (c) $t = 1.8$, (d) $t = 4.5$, and (e) $t = 8$.

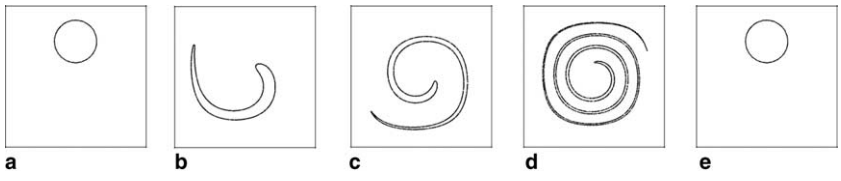

Fig. 11. Time evolution of the iso-zero level set in the vortex flow: (a) $t = 0$, (b) $t = 0.9$, (c) $t = 1.8$, (d) $t = 4.5$, and (e) $t = 8$.

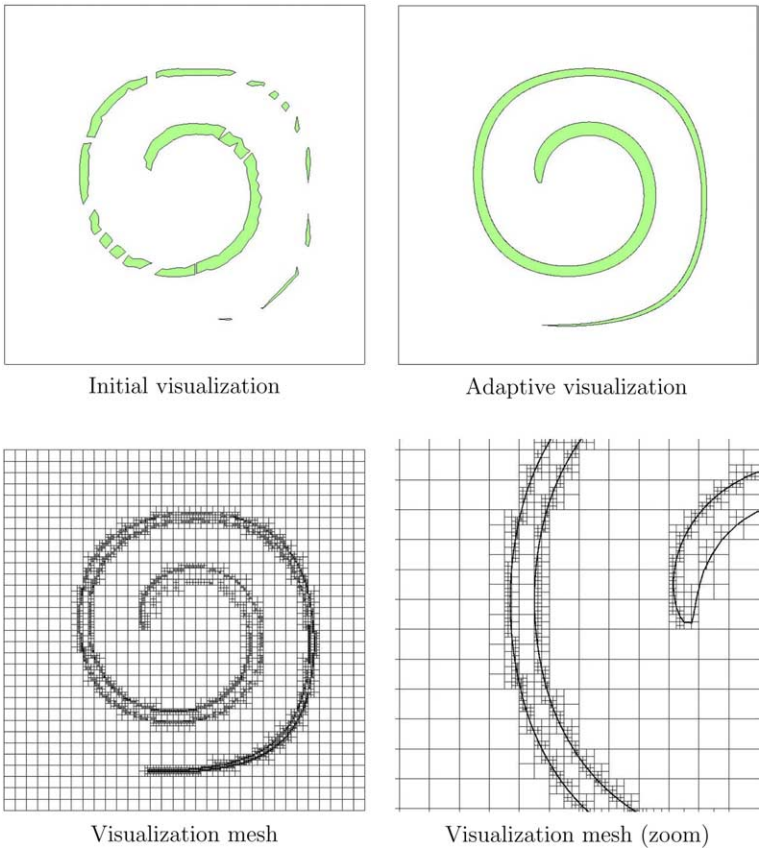

Fig. 12. Visualization of the iso-value $\phi = 0$ at time step 1000. The computations were run on a regular mesh made of $(32 \times 32) = 1024$ quadrangles. The adaptive refined visualization mesh contains 6550 quadrangles. The area in green (A) has been evaluated exactly ($A = 4.691857 \times 10^{-2}$ in the unrefined case and $A = 7.063757 \times 10^{-2}$ in the refined case). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$\begin{aligned} u_x &= 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)g(t), \\ u_y &= -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)g(t), \\ u_z &= -\sin(2\pi x)\sin(2\pi y)\sin^2(\pi z)g(t). \end{aligned} \tag{21}$$

The time dependence $g(t)$ is given by: $g(t) = \cos(\pi t/T)$, where the reversal time period is chosen to be $T = 3$. Again, the temporal term reverses the velocity half way through the calculation so that the final state should be identical to the initial state. Fig. 13 shows the velocity field in the x–y plane as well as the snapshot of the refined hexahedral mesh (32×32).

Table 4 reports the errors computed with the DGM of order $p = 4$ for different meshes of size h and different types of elements (hexahedra and tetrahedra). For the refined meshes, h refers to the average mesh size, that is the total length divided by the number of divisions.

Fig. 14 shows the DG level set solution, respectively, at time $t = 0, 0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4, 1.8, 2.2, 2.6$ and 3 s. As the flow is reversed the final time $t = 3$ s should be identical to the initial data at time $t = 0$ s. In addition frame 1.6 s should be identical to frame 0.4 s and so on.

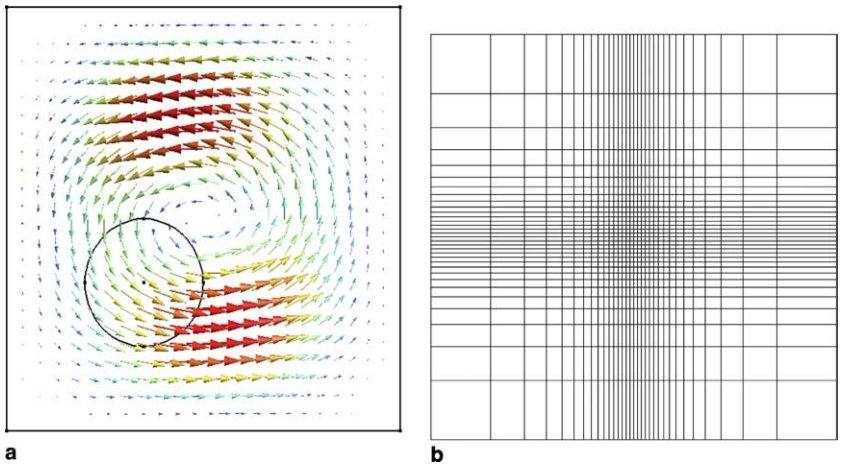

Fig. 13. (a) 3D deformation field: initial data and velocity in the x–y plane. (b) View of the refined hexahedral mesh (32×32) in the y–z plane.

Table 4
3D deformation field: area loss (or gain) at $T/2$ and T

Method	h	% Area		CPU time (h)
		Loss at $T/2$	Loss at T	
DG HEX (4)	1/24	1.7%	1.63%	8.3
DG HEX (4)	1/32	1.56%	1.53%	26.4
DG TET (4)	1/24	0.85%	0.64%	41.1
WENO(5)*	1/100	49%	80%	–
WENO(5)**	1/100	−1.9%	2.6%	–

* Level set [19,43] and ** particle level set [19,43] method.

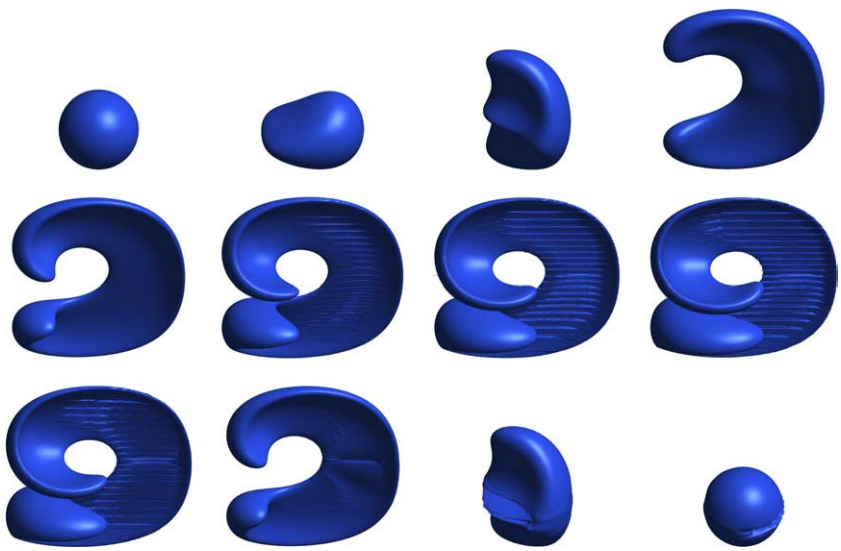

Fig. 14. Deformation test case. DGM ($p = 4$) solution on a 32×32 refined grid of hexahedra with almost no mass loss.

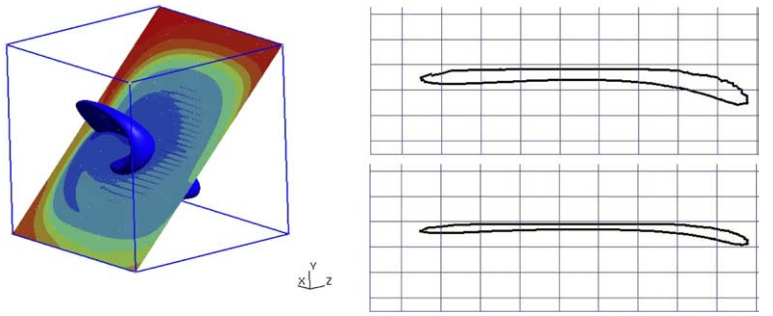

Fig. 15. DG HEX(4) method with $h = 1/32$. (a) Analysis of the iso-zero level set in the plane $y = z$. (b) Zoom in the $y = z$ plane of the level set at time $t = 1$ s and $t = 1.4$ s.

We observe in Fig. 14 a slight under-resolution of the iso-zero level set starting from time $t = 1$ s. We expect and explain this by the dispersion analysis for the DGM. Fig. 15 shows a cut of the deformed sphere in the $y = z$ plane and a zoom of the iso-zero level set on this plane. As the time goes, the width of the iso-zero level set becomes thinner and thinner. We know from the dispersion analysis that for a DGM of order $p = 4$, if the width d becomes too thin ($d/2 < h/4$) we have high wave numbers and those are damped. This transition happens exactly at time $t = 1$ s, explaining our slight under-resolution.

In [19], the authors used this test case to demonstrate the great capability of the particle level set method to conserve the volume of the sphere, while on the other hand their level set method severely fails. For their original level set method discretized with HJ-WENO(5) and including a reinitialization procedure, they observed a non-physical loss of mass (about 80%) (see Fig. 30 in [19]).

5. Conclusion

In this paper, we have proposed a quadrature free, Runge–Kutta discontinuous Galerkin method (QF-RK-DGM) to solve the level set equation on unstructured grids. The proposed method has good conservation properties and exhibits very few mass loses compared to other schemes. Furthermore, we showed that there is no need to reinitialize the level set in case of incompressible flow.

The results presented in this work, especially Tables 2–4, demonstrate that the DGM can be an advantageous alternative to ENO methods for accurate level set updates since:

- DGM seems to give more accurate results and to exhibit less mass loss than ENO methods,
- the implemented QF-RK-DGM combined with BLAS libraries is faster than any other DGM,
- DGM is easy to implement and to parallelize,
- DGM can easily be made p-adaptive. That option could be advantageous here because only the iso-zero is interesting us,
- DGM is implemented as an unstructured method. It can be used in general 3D domain, including curved boundaries and complex topologies.

Currently, we are in the process of applying this methodology to capture the interface in multiphase incompressible flows. Since we already have a parallel version of the DG level set method, we are going to couple this interface solver with our parallel incompressible fluid solver.

References

[1] C.W. Hirt, A.A. Amsden, J.L. Cook, An arbitrary Lagrangian–Eulerian computing method for all flow speeds, Journal of Computational Physics 135 (2) (1997) 203–216.
[2] T. Hughes, W. Liu, T. Zimmermann, Lagrangian–Eulerian finite element formulation for incompressible viscous flows, Computer Methods in Applied Mechanics and Engineering 29 (1981) 239–349.
[3] J. Donea, Arbitrary Lagrangian–Eulerian finite element methods, Computational Methods for Transient Analysis 1 (1983) 473–516.
[4] W. Dettmer, P. Saksono, D. Peric, On a finite element formulation for incompressible Newtonian fluid flows on moving domains in the presence of surface tension, Computer Methods in Applied Mechanics and Engineering 19 (2003) 659–668.
[5] F. Harlow, J. Welch, Volume tracking methods for interfacial flow calculations, Physics of fluids 8 (1965) 21–82.
[6] C. Hirt, B. Nichols, Volume of Fluid Method (VOF) for the dynamics of free boundaries, Journal of Computational Physics 39 (1981) 201–225.
[7] J. Pilliot, E. Puckett, Second order accurate Volume-of-Fluid algorithms for tracking material interfaces, Journal of Computational Physics 199 (2004) 465–502.
[8] S. Osher, J.A. Sethian, Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formulations, Journal of Computational Physics 79 (1988) 12–49.
[9] N. Sukumar, D. Chopp, N. Moes, T. Belytschko, Modeling holes and inclusions by level sets in the extended finite-element method, Computer Methods in Applied Mechanics and Engineering 190 (2001) 6183–6200.
[10] N. Sukumar, D. Chopp, B. Moran, Extended finite element method and fast marching method for three-dimensional fatigue crack propagation, Engineering Fracture Mechanics 70 (2003) 29–48.
[11] R. Fedkiw, T. Aslam, S. Xu, The ghost fluid method for deflagration and detonation discontinuities, Journal of Computational Physics 154 (1999) 393–427.
[12] D. Adalsteinsson, J. Sethian, A level set approach to a unified model for etching, deposition and lithography, iii: re-deposition and re-emission, Journal of Computational Physics 138 (1997) 193–227.
[13] N. Moes, M. Cloirec, P. Cartraud, J. Remacle, A computational approach to handle complex microstructure geometries, Computer Methods in Applied Mechanics and Engineering 192 (2003) 3163–3177.
[14] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, Journal of Computational Physics 114 (1994) 146–159.
[15] M. Sussman, E. Fatemi, P. Smereka, S. Osher, An improved level set method for incompressible two-fluid flows, Computers and Fluids 27 (1998) 663–680.
[16] M. Sussman, E. Fatemi, An efficient, interface preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow, SIAM Journal of Scientific Computing 20 (1999) 1165–1191.
[17] S. Osher, C.-W. Shu, High order essentially non-oscillatory schemes for Hamilton–Jacobi equations, SIAM Journal of Numerical Analysis 728 (1991) 902–921.
[18] C. Hu, C.-W. Shu, Weighted ENO schemes for Hamilton–Jacobi equations, SIAM Journal of Scientific Computing 21 (6) (1999) 2126–2143.
[19] D. Enright, R. Fedkiw, J. Ferziger, I. Mitchell, A hybrid particle level set method for improved interface capturing, Journal of Computational Physics 183 (2002) 83–116.
[20] D.L. Chopp, Computing minimal surfaces via levelset curvature flow, Journal of Computational Physics 106 (1993) 71–91.
[21] S. van der Pijl, A. Segal, C. Vuik, A mass-conserving levelset method for modeling of multiphase flows, Technical Report, Department of Applied Mathematical Analysis, Delft, 2003.
[22] M. Sussman, E. Puckett, A coupled level set and volume-of-fluid method for computing 3d and axisymmetric incompressible two-phase flows, Journal of Computational Physics 162 (2000) 301–337.
[23] J. Strain, Tree methods for moving interfaces, Journal of Computational Physics 151 (1999) 616–648.
[24] B. Cockburn, C.-H. Shu, TVD Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws, Mathematics of Computations 52 (186) (1989) 411–435.
[25] J.-F. Remacle, J. Flaherty, M. Shephard, An adaptive discontinuous Galerkin technique with an orthogonal basis applied to compressible flow problems, SIAM Review 45 (2003) 53–72.
[26] P. Rasetarinera, M. Hussaini, An efficient implicit discontinuous spectral Galerkin method, Journal of Computational Physics 172 (2) (2001) 718–735.
[27] M. Sussman, M. Hussaini, A discontinuous spectral element method for the level set equation, Journal of Scientific Computing 19 (2003) 479–500.
[28] W. Mulder, S. Osher, J.A. Sethian, Computing interface motion in compressible gas dynamics, Journal of Computational Physics 100 (1992) 209–228.
[29] J.A. Sethian, Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Cambridge University Press, Cambridge, 1999.

[30] D.L. Chopp, Some Improvements of the Fast Marching Method, SIAM Journal of Scientific Computing 23 (1) (2001) 230–244.
[31] J. Strain, A fast modular semi-lagrangian method for moving interfaces, Journal of Computational Physics 161 (2000) 512–536.
[32] M. Mount, N.S. Arya, R. Netanyahu, Silverman, An optimal algorithm for approximate nearest neighbor searching, Journal of the ACM 45 (6) (1998) 891–923.
[33] S. Mauch, A fast algorithm for computing the closest point and distance transform, Preprint.
[34] ANN library. URL http://www.cs.umd.edu/~mount/ANN/.
[35] CPT library. URL http://www.acm.caltech.edu/~seanm/projects/cpt/cpt.html.
[36] A. Smoliansky, Numerical modeling of two-fluid interfacial flows, Ph.D. Thesis, University of Jyvaskyla, Finland, 2001.
[37] A.-K. Tornberg, Interface tracking methods with applications to multiphase flows, Ph.D. Thesis, Royal Institute of Technology (KTH), Finland, 2000.
[38] ATLAS (Automatically Tuned Linear Algebra Software). URL http://www.netlib.org/atlas/.
[39] F. Hu, H. Atkins, Eigensolution analysis of the discontinuous Galerkin method with nonuniform grids i: one space dimension, Journal of Computational Physics 182 (2) (2002) 516–545.
[40] M. Ainsworth, Dispersive and dissipative behavior of high order discontinuous Galerkin finite element methods, Journal of Computational Physics 198 (1) (2004) 106–130.
[41] B. Cockburn, C.-W. Shu, Runge–Kutta discontinuous Galerkin methods for convection-dominated problems, Journal of Scientific Computing 16 (2001) 173–261.
[42] S.T. Zalesak, Fully multidimensional flux-corrected transport algorithms for fluids, Journal of Computational Physics 31 (1979) 335–362.
[43] D. Enright, F. Losasso, R. Fedkiw, A fast and accurate semi-Lagrangian particle level set method, Computers and Structures 83 (6–7) (2005) 479–490.
[44] N. Chevaugeon, E. Marchandise, C. Geuzaine, J.-F. Remacle, Efficient visualization of high order finite elements, International Journal for Numerical Methods in Engineering, (2005) (submitted).
[45] R. LeVeque, High-resolution conservative algorithms for advection in incompressible flow, SIAM Journal of Numerical Analysis 33 (1996) 627–665.