Journal of Computational Physics 228 (2009) 3816–3836

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Discretizing delta functions via finite differences and gradient normalization

John D. Towers

MiraCosta College, 3333 Manchester Avenue, Cardiff-by-the-Sea, CA 92007-1516, USA

ARTICLE INFO

Article history:
Received 16 June 2007
Received in revised form 27 October 2008
Accepted 12 February 2009
Available online 27 February 2009

Keywords:
Delta function
Level set method
Discretization
High codimension
Finite difference
Approximation
Regular mesh

ABSTRACT

In [J.D. Towers, Two methods for discretizing a delta function supported on a level set, J. Comput. Phys. 220 (2007) 915–931] the author presented two closely related finite difference methods (referred to here as FDM1 and FDM2) for discretizing a delta function supported on a manifold of codimension one defined by the zero level set of a smooth mapping $u:\mathbb{R}^n \mapsto \mathbb{R}$. These methods were shown to be consistent (meaning that they converge to the true solution as the mesh size $h \to 0$) in the codimension one setting.

In this paper, we concentrate on $n \leqslant 3$, but generalize our methods to codimensions other than one – now the level set function is generally a vector valued mapping $\vec{u}:\mathbb{R}^n \mapsto \mathbb{R}^m,\ 1 \leqslant m \leqslant n \leqslant 3$. Seemingly reasonable algorithms based on simple products of approximate delta functions are not generally consistent when applied to these problems. Motivated by this, we instead use the wedge product formalism to generalize our FDM algorithms, and this approach results in accurate, often consistent approximations. With the goal of ensuring consistency in general, we propose a new gradient normalization process that is applied before our FDM algorithms. These combined algorithms seem to be consistent in all reasonable situations, with numerical experiments indicating $O(h^2)$ convergence for our new gradient-normalized FDM2 algorithm.

In the full codimension setting ($m = n$), our gradient normalization processing also improves accuracy when using more standard approximate delta functions. This combination also yields approximations that appear to be consistent.

1. Introduction

In this paper, we are interested in approximating two types of integrals involving Dirac delta functions. The first type of integral is of the form

$$\begin{aligned}\mathcal{I}_1 &= \int_{\mathbb{R}^n} f(\vec{x}) \prod_{i=1}^{m} \delta(u^i(\vec{x})) \|\wedge_m \nabla \vec{u}(\vec{x})\| \, d\vec{x}, \\ \wedge_m \nabla \vec{u}(\vec{x}) &:= \nabla u^1(\vec{x}) \wedge \nabla u^2(\vec{x}) \wedge \cdots \wedge \nabla u^m(\vec{x}).\end{aligned} \tag{1}$$

(See e.g. [6] for a review of wedge products.) The integral $\mathcal{I}_1$ is an equivalent representation for the integral

$$\int_{\Gamma} f(\vec{x}) \, dV^r, \tag{2}$$

E-mail address: john.towers@cox.net
URL: http://www.miracosta.edu/home/jtowers/

0021-9991/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2009.02.012

where $\vec{x} = (x^1, \ldots, x^n) \in \mathbf{R}^n$, $f : \mathbf{R}^n \mapsto \mathbf{R}$, and Γ is a manifold of dimension $r = n - m$ defined by the intersection of the zero level sets of m smooth functions $u^i : \mathbf{R}^n \mapsto \mathbf{R}$, $i = 1, \ldots, m$ with $1 \leqslant m \leqslant n$. Here dV^r denotes r-dimensional volume. Integrals of the form $\mathcal{I}_1$ occur frequently when applying level set methods [11–13,15,20]. More specifically, one often requires an approximation to the integral (2). What makes this computation not completely straightforward is the fact that the integrand f and the level function $\vec{u}$ are typically only known at the grid points of a regular mesh. In this situation, the representation (1) is more amenable to discretization than the equivalent integral (2), which would require some sort of parametrization of the manifold Γ.

Problems like this also arise outside of level set applications. An example is the task of recovering spherical harmonic amplitudes from data given on a regular three-dimensional mesh. This requires approximating a codimension one integral of the type above, where Γ is a sphere. Ref. [10] gives an interesting, and evidently quite accurate algorithm for accomplishing this.

There are two reasons why we use the somewhat technical wedge product notation in (1). First, it provides a unified representation of several integrals that seem somewhat disparate when more standard notation is used. Second, the wedge product formalism provides us with a novel and accurate way to discretize the delta function products that appear in (1) and (6). We will discuss this discretization in Section 1.2, but for now we provide more familiar representations of the formula (1) that result for $n \leqslant 3$. When $m = 1$, with $u^1 = u$ formula (1) takes the form

$$\mathcal{I}_1 = \int_{\mathbb{R}^n} f(\vec{x})\delta(u(\vec{x}))\|\nabla u(\vec{x})\|\, d\vec{x}. \tag{3}$$

When $m = n$, formula (1) is equivalent to

$$\mathcal{I}_1 = \int_{\mathbb{R}^n} f(\vec{x}) \prod_{i=1}^{n} \delta(u^i(\vec{x}))|\det \nabla\vec{u}(\vec{x})|\, d\vec{x}. \tag{4}$$

Finally, when $n = 3,\ m = 2$ (intersection of two surfaces in $\mathbb{R}^3$), formula (1) is the same as

$$\mathcal{I}_1 = \int_{\mathbb{R}^3} f(\vec{x})\delta(u^1(\vec{x}))\delta(u^2(\vec{x}))\|\nabla u^1(\vec{x}) \times \nabla u^2(\vec{x})\|\, d\vec{x}. \tag{5}$$

In this last formula, if $f \equiv 1$, the integral $\mathcal{I}_1$ gives the arclength of the curve Γ, and this arclength case of formula (5) can already be found in various places in the level set literature, e.g. [2].

The second type of integral that we wish to approximate is of the form

$$\mathcal{I}_2 := \int_{\mathbb{R}^n} f(\vec{x}) \prod_{i=1}^{n} \delta(u_i(\vec{x}))\, d\vec{x}. \tag{6}$$

Integrals of type $\mathcal{I}_2$ occur when applying level set methods for computing multivalued solutions to the semiclassical limit of the Schrödinger equation and the high frequency limit of the wave equation [4,8,9,14]. In this situation, Γ will generally consist of a finite set of points $\Gamma = \{\vec{x}_v : v = 1, \ldots, N\}$, and the integral (6) can be written as a finite sum:

$$\mathcal{I}_2 = \sum_{v=1}^{N} f(\vec{x}_v)/|\det \nabla\vec{u}(\vec{x}_v)|. \tag{7}$$

1.1. Computing observables for the Schrödinger equation

We give a very brief sketch of how integrals of type $\mathcal{I}_2$ arise in problems of high frequency wave propagation, focusing on the task of computing physical observables for the semiclassical limit of the Schrödinger equation, as presented in [8]. For the purposes of the present paper, the problem boils down to first solving $n + 1$ Cauchy problems for the linear Liouville equation

$$w_t + \vec{p} \cdot \nabla_{\vec{x}} w - \nabla_{\vec{x}} V(\vec{x}) \cdot \nabla_{\vec{p}} w = 0 \tag{8}$$

for the n functions $w = \phi^i(\vec{x}, \vec{p}, t)$, and the single function $w = f(\vec{x}, \vec{p}, t)$. In the PDE (8), $V(\vec{x})$ is a given potential, $\vec{x}$ represents the spatial variables, and $\vec{p}$ is a vector of auxiliary variables introduced as a device to capture multivalued solutions [7]. The initial data for the Cauchy problems (8) is

$$\begin{aligned} &\phi^i(\vec{x}, \vec{p}, 0) = p^i - \partial_{x^i} S_0(\vec{x}), \quad i = 1, \ldots, n, \\ &f(\vec{x}, \vec{p}, 0) = \rho_0(\vec{x}), \end{aligned} \tag{9}$$

where S_0 (the phase), and ρ_0 (the density) are prescribed functions. Using the solutions of (8) and (9), the averaged density $\bar{\rho}$ and velocity components $\bar{v}^i$ (these are physical observables) can be computed via

$$\begin{aligned} \bar{\rho}(\vec{x}, t) &= \int_{\mathbb{R}^n} f(\vec{x}, \vec{p}, t) \prod_{i=1}^{n} \delta(\phi^i(\vec{x}, \vec{p}, t))\, d\vec{p}, \\ \bar{v}^i(\vec{x}, t) &= \frac{1}{\bar{\rho}(\vec{x}, t)} \int_{\mathbb{R}^n} p^i f(\vec{x}, \vec{p}, t) \prod_{i=1}^{n} \delta(\phi^i(\vec{x}, \vec{p}, t))\, d\vec{p}, \quad i = 1, \ldots, n. \end{aligned} \tag{10}$$

Thus at each point $(\vec{x}, t)$, each physical observable requires the computation of an $\mathcal{I}_2$ type of integral. One potential source of notational confusion is that in the integrals appearing in (10), $\vec{x}$ is playing the role of a parameter, while $\vec{p}$ is playing the role that was played by $\vec{x}$ in our earlier discussion, e.g. (6). Also, note that here $\vec{p} \mapsto \vec{\phi}(\vec{x}, \vec{p}, t)$ is playing the role of the level function.

1.2. Discretization algorithms

We seek a discretized version of the product of delta functions appearing in the integrals $\mathcal{I}_1$ and $\mathcal{I}_2$, and our goal is to obtain this discretization by differencing Heaviside functions, generalizing the approach in [19]. To this end, we start by considering the following wedge product:

$$\nabla H(u^1) \wedge \cdots \wedge \nabla H(u^m) = H'(u^1) \cdots H'(u^m) \nabla u^1 \wedge \cdots \wedge \nabla u^m = \delta(u^1) \cdots \delta(u^m) \nabla u^1 \wedge \cdots \wedge \nabla u^m. \tag{11}$$

Taking the dot product of both sides with $\nabla u^1 \wedge \cdots \wedge \nabla u^m$, and then solving for the product of delta functions gives

$$\delta(u^1) \cdots \delta(u^m) = \frac{\nabla H(u^1) \wedge \cdots \wedge \nabla H(u^m) \cdot \nabla u^1 \wedge \cdots \wedge \nabla u^m}{\|\nabla u^1 \wedge \cdots \wedge \nabla u^m\|^2}. \tag{12}$$

This is the basic formula underlying our approach – the right hand side of (12) is what we will discretize. (See Section A for a translation of this and related formulas into ones involving determinants instead of wedge products.)

To describe our approximation methods, we first discretize $\mathbb{R}^n$ by defining the mesh points

$$\left\{\vec{x}_{\mathbf{k}} = \left(x^1_{k_1}, \ldots x^n_{k_n}\right) : \mathbf{k} := (k_1, \ldots, k_n) \in \mathbf{Z}^n\right\}$$

of a regular grid. For simplicity of notation, we assume that the mesh spacing h is the same in all dimensions, $x^i_{k_i} = k_i h,\ k_i \in \mathbf{Z}$. If $v_{\mathbf{k}} = v(\vec{x}_{\mathbf{k}})$ is a function defined at each meshpoint $\vec{x}_{\mathbf{k}}$, we define the discrete gradient operator ∇^h via

$$\nabla^h v_{\mathbf{k}} = \sum_{i=1}^{n} \left(\frac{v(\vec{x}_{\mathbf{k}} + h\vec{e}_i) - v(\vec{x}_{\mathbf{k}} - h\vec{e}_i)}{2h}\right)\vec{e}_i, \tag{13}$$

where $\{\vec{e}_1, \ldots, \vec{e}_n\}$ is the standard basis for $\mathbf{R}^n$.

We approximate the integrals $\mathcal{I}_1$ and $\mathcal{I}_2$ using

$$\mathcal{I}^h_1 := h^n \sum_{\mathbf{k} \in \mathcal{S}} f(\vec{x}_{\mathbf{k}}) \delta^h(\vec{x}_{\mathbf{k}}; \vec{u}) \left\| \wedge_m \nabla^h \vec{u}_{\mathbf{k}} \right\|, \quad \mathcal{I}^h_2 := h^n \sum_{\mathbf{k} \in \mathcal{S}} f(\vec{x}_{\mathbf{k}}) \delta^h(\vec{x}_{\mathbf{k}}; \vec{u}), \tag{14}$$

where $\delta^h(\vec{x}_{\mathbf{k}}; \vec{u})$ is a discretized version of the delta function product $\prod_{i=1}^m \delta(u^i(\vec{x}_{\mathbf{k}}))$, and $\mathcal{S}$ is a subset of $\mathbf{Z}^n$ containing those indices $\mathbf{k}$ where $f(\vec{x}_{\mathbf{k}})\delta^h(\vec{x}_{\mathbf{k}}; \vec{u}) \neq 0$.

We can now discretize the right side of formula (12). The result is

Finite difference method 1 (FDM1)

$$\delta^h_{FDM1}(\vec{x}_{\mathbf{k}}; \vec{u}) := \frac{\nabla^h H^h(u^1_{\mathbf{k}}) \wedge \cdots \wedge \nabla^h H^h(u^m_{\mathbf{k}}) \cdot \nabla^h u^1_{\mathbf{k}} \wedge \cdots \wedge \nabla^h u^m_{\mathbf{k}}}{\left\|\nabla^h u^1_{\mathbf{k}} \wedge \cdots \wedge \nabla^h u^m_{\mathbf{k}}\right\|^2}. \tag{15}$$

In (15), $H^h(\cdot)$ represents a possibly smoothed version of the Heaviside function $H(\cdot)$, with the regularization parameter depending on the mesh size h. FDM1 is a direct generalization of the Method 1 found in [19], meaning that the two algorithms are the same if $m = 1$.

Examples of smoothed Heaviside functions used in (15) are

$$H^{C,\epsilon}(z) = \begin{cases} 0, & z < \epsilon, \\ \frac{1}{2} + \frac{z}{2\epsilon} + \frac{1}{2\pi} \sin\left(\frac{\pi z}{\epsilon}\right), & -\epsilon \leqslant z \leqslant \epsilon, \\ 1, & \epsilon < z \end{cases} \tag{16}$$

with $\epsilon = 1.5h$ and

$$H^{L,\epsilon}(z) = \begin{cases} 0, & z \leqslant \epsilon, \\ \frac{1}{2} + \frac{1}{\epsilon}\left(z - \frac{\mathrm{sign}(z) z^2}{2\epsilon}\right), & |z| < \epsilon, \\ 1, & z \geqslant \epsilon \end{cases} \tag{17}$$

with $\epsilon = h$ or $2h$. In many cases, FDM1 also gives acceptable results using $H(\cdot)$ without any smoothing, but a small dose of smoothing generally gives better accuracy.

The approximate Heaviside function $H^{C,\epsilon}$ is associated with the approximate delta function $\delta^{C,\epsilon}$ defined by

$$\delta^{C,\epsilon}(z) = \begin{cases} \frac{1}{2\epsilon}\left(1 + \cos\left(\frac{\pi z}{\epsilon}\right)\right), & |z| < \epsilon, \\ 0, & |z| \geqslant \epsilon, \end{cases} \tag{18}$$

and the approximate Heaviside function $H^{L,\epsilon}$ is associated with the linear hat approximate delta function:

$$\delta^{L,\epsilon}(z) = \begin{cases} \frac{1}{\epsilon}\left(1 - \left|\frac{z}{\epsilon}\right|\right), & |z| < \epsilon, \\ 0, & |z| \geqslant \epsilon. \end{cases} \tag{19}$$

To derive our second finite difference method, we start with the function $I(z) = \int_0^z H(\zeta)\,d\zeta = \max(z, 0)$, and the relationship

$$\nabla I(u^i) = H(u^i)\nabla u^i, \quad i = 1, \ldots, m. \tag{20}$$

Taking the dot product of both sides with the vector ∇u^i, and solving for $H(u^i)$ gives

$$H(u^i) = \frac{\nabla I(u^i) \cdot \nabla u^i}{\|\nabla u^i\|^2}, \quad i = 1, \ldots, m. \tag{21}$$

Next, we discretize (12) as in the case of FDM1, but this time we also discretize the formula (21), which gives the following two-stage algorithm:

Finite difference method 2 (FDM2)

$$\begin{aligned} H^h(u^i_{\mathbf{k}}) &:= \frac{\nabla^h I(u^i_{\mathbf{k}}) \cdot \nabla^h u^i_{\mathbf{k}}}{\left\|\nabla^h u^i_{\mathbf{k}}\right\|^2}, \quad i = 1, \ldots, m, \\ \delta^h_{FDM2}(\vec{x}_{\mathbf{k}}; \vec{u}) &:= \frac{\nabla^h H^h(u^1_{\mathbf{k}}) \wedge \cdots \wedge \nabla^h H^h(u^m_{\mathbf{k}}) \cdot \nabla^h u^1_{\mathbf{k}} \wedge \cdots \wedge \nabla^h u^m_{\mathbf{k}}}{\left\|\nabla^h u^1_{\mathbf{k}} \wedge \cdots \wedge \nabla^h u^m_{\mathbf{k}}\right\|^2}. \end{aligned} \tag{22}$$

FDM2 is approximately a generalization of Method 2 in [19]. Note that we use the function I in FDM2 as is, i.e., without any regularization. This is an advantage of Method 2 – there are no parameters to specify.

We are also interested in approximate delta functions constructed as a product of pointwise approximate delta functions:

Product of pointwise approximate delta functions (PDF)

$$\delta^h_{PDF}(\vec{x}_{\mathbf{k}}; \vec{u}) := \prod_{i=1}^{m} \delta^{P,\epsilon}(u^i(\vec{x}_{\mathbf{k}})). \tag{23}$$

Here $\delta^{P,\epsilon}$ denotes a pointwise approximate delta function such as $\delta^{C,\epsilon}$ or $\delta^{L,\epsilon}$, and $\epsilon = \nu h$, where ν is some constant. For example, we usually take $\nu = 1.5$ for $\delta^{C,\epsilon}$, and $\nu = 1$ or $\nu = 2$ for $\delta^{L,\epsilon}$.

For PDF-type delta functions, it is often more effective to allow ϵ to vary spatially, depending on the local behavior of the level function $\vec{u}$. For the case where $m = n$, we will also consider the following local version of PDF, proposed in [8,9]:

Product of pointwise approximate delta functions – local version (PDFL)

$$\delta^h_{PDFL}(\vec{x}_{\mathbf{k}}; \vec{u}) := \prod_{i=1}^{n} \delta^{L,\epsilon_{\mathbf{k}}}(u^i(\vec{x}_{\mathbf{k}})), \quad \epsilon_{\mathbf{k}} = 2\max(1, |\det \nabla^h \vec{u}_{\mathbf{k}}|)h. \tag{24}$$

Here we are using the notation $\det \nabla^h \vec{u}_{\mathbf{k}}$ in place of $\nabla^h u^1_{\mathbf{k}} \wedge \cdots \wedge \nabla^h u^n_{\mathbf{k}}$. This is justified, since for the special case where $m = n$, the wedge product $\nabla u^1 \wedge \cdots \wedge \nabla u^n$ is equal to the Jacobian $\det \nabla \vec{u}$.

Remark 1.1. We are mostly interested in the FDM algorithms; we include PDF and PDFL primarily for the purpose of comparison. However, a secondary reason for discussing the PDF/PDFL algorithms is related to the gradient normalization process that we will discuss in Section 3. We will see in our numerical experiments that the PDF/PDFL algorithms can often be improved by combining them with gradient normalization.

2. Rationale for discretizing the wedge product formulation

It is already well known that seemingly reasonable methods for approximating codimension one integrals of type $\mathcal{I}_1$ may not be consistent [5], and several methods that overcome this difficulty have been proposed [5,16,19]. By consistent, we mean that the approximation converges to the true solution as the mesh size $h \to 0$.

To discuss the kind of difficulties that arise when the codimension is greater than one, let us focus on the full codimension problem with $m = n = 2$ to be specific, and assume that both of the level functions are signed distance functions, whose gradients are orthogonal. In this case, the matrix $\nabla \vec{u}$ is orthogonal, $|\det \nabla \vec{u}| = 1$, and $\mathcal{I}_1 = \mathcal{I}_2$.

In one space dimension, a sufficient condition for consistency when the level function is of the form $u(x) = x - \bar{x}$ (a signed distance function) is [1,17]

$$h \sum_{j \in \mathbf{Z}} \delta^h(x_j; u) = 1 + O(h^{\mu}), \quad \mu > 0. \tag{25}$$

There are many approximate delta functions that satisfy this property, for example $\delta^{C,\epsilon}$ with $\epsilon = 1.5\nu h$, $\delta^{L,\epsilon}$ with $\epsilon = \nu h$, $\nu \in \mathbf{Z}^+$ [18]. Other examples are provided by the one-dimensional versions of the FDM1 and FDM2 algorithms.

Now consider the two-dimensional situation. Suppose that our level functions are $u(x,y)$, and $v(x,y)$, and that $\delta^h(x_j, y_k; u, v) \approx \delta(u(x_j, y_k))\,\delta(v(x_j, y_k))$ is a discrete approximation to the product of delta functions $\delta(u)\delta(v)$. In this setting, the analog of (25) is

$$h^2 \sum_{j\in\mathbf{Z}} \sum_{k\in\mathbf{Z}} \delta^h(x_j, y_k; u, v) = 1 + O(h^\mu), \quad \mu > 0. \tag{26}$$

This two-dimensional condition seems to be much harder to satisfy than the one-dimensional version. For example, it is natural to try to construct $\delta^h(x_j, y_k; u, v)$ by forming a product of codimension one approximations

$$\delta^h(x_j, y_k; u, v) = \delta^h(x_j, y_k; u)\delta^h(x_j, y_k; v). \tag{27}$$

Unfortunately, unless it happens that u and v are aligned with the mesh, i.e.,

$$u = u(x), \quad v = v(y), \quad \text{or} \quad u = u(y), \quad v = v(x).$$

(26) generally fails, and this approach will not be consistent, see Example 1 of Section 4. This lack of consistency may occur even if the codimension one delta functions are FDM1 or FDM2, despite the fact they yield consistent approximations for codimension one instances of $\mathcal{I}_1$. The main source of difficulty here is misalignment of the level sets with the mesh. For codimension one problems, lack of consistency due to this type of misalignment has been well studied [5,17,18].

It is the lack of consistency for product type approximations of the form (27) that leads us to discretize the wedge product formulation on the right side of (12) rather than the more straightforward product formulation on the left side of (12). Our numerical experiments indicate that in many cases, this wedge product formulation does indeed give consistent approximations. In those cases where consistency fails, it can be recovered by the gradient normalization processing described in the next section.

Above we have explained that simple products of one-dimensional or codimension one delta functions are not adequate for higher codimension problems, but have not explained why the new FDM algorithms based on wedge products might be more effective. The following proposition shows that for a very simple linear full codimension problem in $\mathbb{R}^2$, the FDM1 algorithm gives the exact solution. (Numerical experiments indicate that FDM2 also gives the exact solution for this type of problem.) The simplicity of the setup notwithstanding, this is significant because none of the more standard types of algorithms discussed above are generally consistent when applied to this type of problem; see Example 1 of Section 4.

Proposition 2.1. *Assume that $u : \mathbb{R}^2 \mapsto \mathbb{R}$ and $v : \mathbb{R}^2 \mapsto \mathbb{R}$ are linear, $\mathcal{D} := u_x v_y - u_y v_x \neq 0$, and $f(x,y) \equiv 1$. If we approximate $\mathcal{I}_2$ by $\mathcal{I}_2^h$ defined by*

$$\mathcal{I}_2^h := h^2 \sum_{j\in\mathbb{Z}} \sum_{k\in\mathbb{Z}} \delta^h_{FDM1}(x_j, y_k; u, v), \tag{28}$$

then $\mathcal{I}_2^h = \mathcal{I}_2 = 1/|\mathcal{D}|$, i.e., the approximation to $\mathcal{I}_2$ using FDM1 is exact.

Proof 1. First, note that we are dealing with the special case of formula (7) where $N = 1$, $f(x,y) \equiv 1$, and so the exact value of $\mathcal{I}_2$ is $\mathcal{I}_2 = 1/|\mathcal{D}|$. Let Δ_x^0 and Δ_y^0 denote centered difference operators, i.e., $\Delta_0^x p_{j,k} = (p_{j+1,k} - p_{j-1,k})/2$, $\Delta_0^y p_{j,k} = (p_{j,k+1} - p_{j,k-1})/2$. With this notation,

$$\delta^h_{FDM1}(x_j, y_k; u, v) = \frac{1}{h^2\mathcal{D}}\left(\Delta_0^x H(u_{j,k})\Delta_0^y H(v_{j,k}) - \Delta_0^y H(u_{j,k})\Delta_0^x H(v_{j,k})\right). \tag{29}$$

Assume that the lines $u = 0$ and $v = 0$ intersect at the point (x^0, y^0). Consider the setup shown in Fig. 1, which shows a closed rectangular region $\mathcal{R}$ containing (x^0, y^0), and having sides C_α, $\alpha \in \{L, B, R, T\}$. In addition to the line $u = 0$, we show a narrow strip of width $O(h)$ containing that line. It is bounded by a pair of dashed lines. The quantities $\Delta_0^x H(u_{j,k})$ and $\Delta_0^y H(u_{j,k})$ are supported within that strip. Similarly, we show a narrow strip centered on the line $v = 0$, marked by dashed lines, where $\Delta_0^x H(v_{j,k})$ and $\Delta_0^y H(v_{j,k})$ are supported. Note that the discrete delta function $\delta^h_{FDM1}(x_j, y_k; u, v)$ is supported within the region Q where the two strips intersect. Clearly, we may take the rectangle $\mathcal{R}$ large enough that the support of $\delta^h_{FDM1}(x_j, y_k; u, v)$ (contained in the region Q) lies in the interior $\mathcal{R}$. Moreover, it is possible to construct $\mathcal{R}$ so that the lines $u = 0$ and $v = 0$, as well as the surrounding transition zones, intersect the sides of the rectangle away from the corners. Finally, by making one final small adjustment, if necessary, we can arrange that the sides of the rectangle coincide with grid lines. Let J_1, J_2, K_1, and K_2 be indices such that $\mathcal{R} = [x_{J_1}, x_{J_2}] \times [y_{K_1}, y_{K_2}]$.

Plugging (29) into (28), and using the fact that $\delta^h_{FDM1}(x_j, y_k; u, v)$ is supported in the interior of $\mathcal{R}$, we have

$$\mathcal{I}_2^h = \frac{1}{\mathcal{D}} \sum_{k=K_1}^{K_2} \sum_{j=J_1}^{J_2} \left(\Delta_0^x H(u_{j,k})\Delta_0^y H(v_{j,k}) - \Delta_0^y H(u_{j,k})\Delta_0^x H(v_{j,k})\right). \tag{30}$$

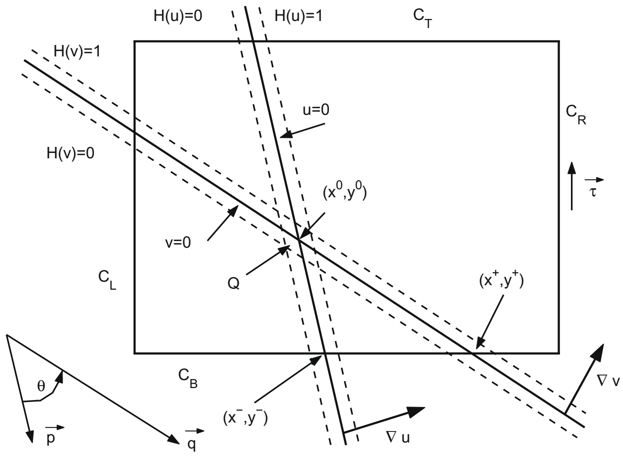


Fig. 1. The geometry of Proposition 2.1.

The following identities hold for the terms in this sum:

$$\Delta_0^x H(u_{j,k})\Delta_0^y H(v_{j,k}) = \Delta_0^x\big(H(u_{j,k})\Delta_0^y H(v_{j,k})\big) - H(u_{j,k})\Delta_0^x\Delta_0^y H(v_{j,k}) - \frac{1}{2}\left[\Delta_+^x H(u_{j,k})\Delta_+^x\Delta_0^y H(v_{j,k}) - \Delta_+^x H(u_{j-1,k})\Delta_+^x\Delta_0^y H(v_{j-1,k})\right],$$
$$\Delta_0^y H(u_{j,k})\Delta_0^x H(v_{j,k}) = \Delta_0^y\big(H(u_{j,k})\Delta_0^x H(v_{j,k})\big) - H(u_{j,k})\Delta_0^y\Delta_0^x H(v_{j,k}) - \frac{1}{2}\left[\Delta_+^y H(u_{j,k})\Delta_+^y\Delta_0^x H(v_{j,k}) - \Delta_+^y H(u_{j,k-1})\Delta_+^y\Delta_0^x H(v_{j,k-1})\right]. \quad (31)$$

Two observations about (31) allow us to get the sum (30) into a form where we can sum by parts. The first is that $\Delta_0^x\Delta_0^y H(v_{j,k}) = \Delta_0^y\Delta_0^x H(v_{j,k})$. The second observation is that the terms in square brackets are only nonzero at a few grid points near the intersection point (x^0, y^0), and in fact they telescope, making no contribution to the sum. Thus,

$$\mathcal{I}_2^h = \frac{1}{\mathcal{D}}\sum_{k=K_1}^{K_2}\sum_{j=J_1}^{J_2}\left(\Delta_0^x(H(u_{j,k})\Delta_0^y H(v_{j,k})) - \Delta_0^y(H(u_{j,k})\Delta_0^x H(v_{j,k}))\right) = \frac{1}{\mathcal{D}}(S_L + S_R + S_B + S_T), \quad (32)$$

where we have summed by parts to get the second equality, and

$$\begin{aligned}
S_L &= -\sum_{k=K_1}^{K_2}\frac{1}{2}\left(H(u_{J_1-1,k})\Delta_0^y H(v_{J_1-1,k}) + H(u_{J_1,k})\Delta_0^y H(v_{J_1,k})\right),\\
S_R &= +\sum_{k=K_1}^{K_2}\frac{1}{2}\left(H(u_{J_2,k})\Delta_0^y H(v_{J_2,k}) + H(u_{J_2+1,k})\Delta_0^y H(v_{J_2+1,k})\right),\\
S_B &= +\sum_{j=J_1}^{J_2}\frac{1}{2}\left(H(u_{j,K_1-1})\Delta_0^x H(v_{j,K_1-1}) + H(u_{j,K_1})\Delta_0^x H(v_{j,K_1})\right),\\
S_T &= -\sum_{j=J_1}^{J_2}\frac{1}{2}\left(H(u_{j,K_2})\Delta_0^x H(v_{j,K_2}) + H(u_{j,K_2+1})\Delta_0^x H(v_{j,K_2+1})\right).
\end{aligned} \quad (33)$$

Consider the sums S_α appearing in (33). Each S_α corresponds to a side C_α of $\partial\mathcal{R}$, and the plus and minus signs are consistent with $-C_L + C_R + C_B - C_T$, i.e., a counterclockwise traversal of $\partial\mathcal{R}$. Due to the factors $\Delta_0^x H(v)$ and $\Delta_0^y H(v)$ appearing in each term of these sums, it is clear that the only nonzero contribution comes from the narrow transition region where the jump in $H(v)$ is concentrated. This transition region intersects $\partial\mathcal{R}$ in two places, but we can ignore the one where $H(u) = 0$. Thus all of the contribution to the sums S_α comes from the segment of $\partial\mathcal{R}$ where the transition zone for $H(v)$ intersects $\partial\mathcal{R}$, and simultaneously $H(u) = 1$. In particular, this segment is centered on the point (x^+, y^+), where $H(u) = 1$, and the line $v = 0$ intersects $\partial\mathcal{R}$. Moreover, only one of the S_α sums, call it S_{α^*} is nonzero. Since $H(u) \equiv 1$ in a neighborhood of the transition zone containing (x^+, y^+), the single nonzero sum S_{α^*} reduces to an average of two telescoping sums, and it is easy to see that $S_{\alpha^*} = \pm 1$. So, we have

$$\mathcal{I}_2^h = S_{\alpha^*}/\mathcal{D} = \pm 1/\mathcal{D}, \quad (34)$$

and thus, recalling that $\mathcal{I}_2 = 1/|\mathcal{D}|$, it only remains to show that $\text{sign}(S_{\alpha^*}) = \text{sign}(\mathcal{D})$.

To this end, let $\vec{\tau}(x, y)$ denote the unit tangent vector to $\partial\mathcal{R}$, with the sign of $\vec{\tau}$ taken to be consistent with counterclockwise traversal of $\partial\mathcal{R}$. Due to the plus and minus signs attached to the sums in (33),

$$\operatorname{sign}(S_{\alpha^*}) = \operatorname{sign}(\vec{\tau}(x^+, y^+) \cdot \nabla v). \tag{35}$$

There are two points where the line $u = 0$ intersects $\partial\mathcal{R}$. Of these two points, let (x^-, y^-) denote the one where

$$\operatorname{sign}(\vec{\tau}(x^-, y^-) \cdot \nabla u) = \operatorname{sign}(\vec{\tau}(x^+, y^+) \cdot \nabla v). \tag{36}$$

Define the vectors $\vec{p} = (x^- - x^0, y^- - y^0)$ and $\vec{q} = (x^+ - x^0, y^+ - y^0)$. By considering the possible cases, one finds that because of (36), the angle θ, measured positive counterclockwise from $\vec{p}$ to $\vec{q}$, satisfies $\operatorname{sign}(\theta) = \operatorname{sign}(\vec{\tau}(x^+, y^+) \cdot \nabla v)$. Note also that $\operatorname{sign}(\det(\vec{p}, \vec{q})) = \operatorname{sign}(\det(\nabla u, \nabla v)) = \operatorname{sign}(\mathcal{D})$. Combining the observation that $\operatorname{sign}(\theta) = \operatorname{sign}(\det(\vec{p}, \vec{q}))$ with (35), the proof is complete. □

Remark 2.1. A key step in the proof above is getting the sum appearing in (30) into a form where we can sum by parts. This does not seem to generalize to the three-dimensional setting. Indeed, it is possible to construct three-dimensional examples (Examples 3 and 4 in Section 4) where the FDM algorithms (and also the PDF/PDFL algorithms) do not converge to the correct solution. However, when combined with the gradient normalization that we describe in the next section, the FDM algorithms appear to be consistent for all data that is smooth and nonsingular.

Remark 2.2. For codimension two problems (two surfaces in $\mathbb{R}^3$ that intersect in one or more curves), the FDM algorithms seem to be consistent and accurate for data that is not close to singular. No cases of inconsistency of the type observed for certain full codimension problems in $\mathbb{R}^3$ have been found. We do not presently have an analysis of even a simple problem like that of Proposition (2.1) for the codimension two situation.

3. Gradient normalization

The accuracy of most approximations to delta functions is somewhat sensitive to the particular form of the level set function $\vec{u}$. The best possible scenario is to have all components u^i of the level function $\vec{u}$ satisfy the following three conditions:

- **G1**: Their gradients are pairwise orthogonal.
- **G2**: Their gradients are aligned with the coordinate axes, i.e., perfectly aligned with the mesh.
- **G3**: They are signed distance functions.

In level set applications, the first and third of these conditions are often met to a high degree of accuracy. This is accomplished by re-orthogonalizing and re-distancing the level set functions at regular intervals in the evolution process [11,3,4,9]. The second condition is the least likely to be satisfied because it is difficult or impossible to enforce except in special cases. In any event, we allow for the possibility that none of **G1, G2, G3** is satisfied.

Below we propose two methods, referred to collectively as gradient normalization, of modifying the level set function $\vec{u}$. We do this in such a way that the level set Γ is close to stationary, but the gradient changes so that as many as possible of **G1**, **G2**, **G3** are approximately satisfied. We apply these methods to the discrete level set function $\vec{u}_{\mathbf{k}}$ before applying FDM1, FDM2, PDF, or PDFL to compute our approximate delta function.

3.1. Full gradient normalization (FGN)

The gradient normalization method that we will discuss now is applicable when $m = n$, i.e., full codimension. In $\mathbb{R}^2$ it is possible to construct full codimension examples where neither the PDF nor the PDFL method is consistent, even if the level functions are signed distance functions whose zero level sets intersect orthogonally. As explained in Section 2, this lack of consistency stems from misalignment of the grid with the zero level sets of the functions u^i. In $\mathbb{R}^3$, we can construct full codimension examples where not only PDF and PDFL, but also our FDM algorithms fail to be consistent. With the additional processing that we now describe, it seems that all of these algorithms are generally consistent.

Given the level function vector $\vec{u}_{\mathbf{k}}$, we compute the $n \times n$ discrete gradient matrix $\nabla^h \vec{u}_{\mathbf{k}}$ using (13). We then replace $\vec{u}_{\mathbf{k}}$ by $\vec{v}_{\mathbf{k}}$ and $f_{\mathbf{k}}$ by $g_{\mathbf{k}}$ where

$$\vec{v}_{\mathbf{k}} := \left[\nabla^h \vec{u}_{\mathbf{k}}\right]^{-1} \vec{u}_{\mathbf{k}}, \quad g_{\mathbf{k}} = \begin{cases} f_{\mathbf{k}}, & \text{for } \mathcal{I}_1, \\ f_{\mathbf{k}} / \left|\det \nabla^h \vec{u}_{\mathbf{k}}\right|, & \text{for } \mathcal{I}_2. \end{cases} \tag{37}$$

and then apply FDM1, FDM2, PDF, or PDFL to $\vec{v}_{\mathbf{k}}$ and $g_{\mathbf{k}}$. Using the fact that $\vec{u} = \vec{0}$ on the level set Γ, it is clear that on Γ, $\nabla \vec{v} = I_n$, where I_n denotes the $n \times n$ identity matrix. In other words, by computing $\vec{v}$ we are performing a local realignment of the level sets with the coordinate axes.

Note that FGN processing enforces all three of **G1, G2, G3** for points $\vec{x}$ lying on Γ.

Let us see how FGN works in a simple example, specifically a full codimension problem in $\mathbb{R}^2$. We are interested in approximating $\mathcal{I}_1$ using a PDF-FGN algorithm. For the sake of concreteness, let $\epsilon = 2h$, and take $\delta^{L,\epsilon}$ to be the one-dimensional pointwise approximate delta function used to construct our PDF-FGN algorithm. With this choice, the one-dimensional consistency requirement (25) is satisfied, with the $O(h^{\mu})$ term equal to zero.

Assume that the two level set functions u and v intersect at the single point $(x,y) = (\bar{x},\bar{y})$. In this case, by considering the first few terms in a Taylor series centered at $(\bar{x},\bar{y})$, and recalling that $\vec{v}(\bar{x},\bar{y}) = \vec{0}$, we obtain the following approximation, valid for $(x_j,y_k) \approx (\bar{x},\bar{y})$:

$$\vec{v}_{j,k} = (\tilde{u}(x_j,y_k), \tilde{v}(x_j,y_k)) \approx (x_j - \bar{x}, y_k - \bar{y}). \tag{38}$$

Thus,

$$\delta^h_{PDF}(x_j,y_k;\vec{v}) = \delta^{L,\epsilon}(\tilde{u}(x_j,y_k))\delta^{L,\epsilon}(\tilde{v}(x_j,y_k)) \approx \delta^{L,\epsilon}(x_j - \bar{x})\delta^{L,\epsilon}(y_k - \bar{y}). \tag{39}$$

It follows that

$$h^2 \sum_{k\in\mathbb{Z}} \sum_{j\in\mathbb{Z}} \delta^h_{PDF}(x_j,y_k;\vec{v}) f_{j,k} \approx h^2 \sum_{k\in\mathbb{Z}} \delta^{L,\epsilon}(y_k - \bar{y}) \sum_{j\in\mathbb{Z}} \delta^{L,\epsilon}(x_j - \bar{x}) f_{j,k}. \tag{40}$$

Since $\delta^{L,\epsilon}(x_j - \bar{x})$ is zero for $|x_j - \bar{x}| > O(h)$ and $\delta^{L,\epsilon}(y - y_k)$ is zero for $|y_k - \bar{y}| > O(h)$, our approximation should still be valid if we make the approximation $f_{j,k} \approx f(\bar{x},\bar{y})$. From this we obtain

$$h^2 \sum_{k\in\mathbb{Z}} \sum_{j\in\mathbb{Z}} \delta^h_{PDF}(x_j,y_k;\vec{v}) f_{j,k} \approx f(\bar{x},\bar{y}) h \sum_{k\in\mathbb{Z}} \delta^{L,\epsilon}(y_k - \bar{y}) h \sum_{j\in\mathbb{Z}} \delta^{L,\epsilon}(x_j - \bar{x}). \tag{41}$$

Recalling the consistency condition (25), we get the desired approximation:

$$h^2 \sum_{k\in\mathbb{Z}} \sum_{j\in\mathbb{Z}} \delta^h_{PDF}(x_j,y_k;\vec{v}) f_{j,k} \approx f(\bar{x},\bar{y}) = \mathcal{I}_1. \tag{42}$$

Obviously this is only a heuristic argument meant to explain why we might expect the FGN processing to be effective. The key point is that the level sets of the modified level set function $\vec{v}$ become (approximately) aligned with the coordinate axes, at least near the point $(\bar{x},\bar{y})$. Due to the realignment, the consistency of the component one-dimensional approximate delta functions takes effect, enforcing consistency for the combined multidimensional approximate delta function. An illustration of this local realignment can be seen in Fig. 4, associated with Example 8 in Section 4.

There is a potential difficulty that must be dealt with when implementing (37) in combination with the FDM algorithms. We generally assume that $\det \nabla \vec{u}_{\mathbf{k}}$ does not vanish for $\vec{x}$ near Γ. However, there may be zeros of $\det \nabla \vec{u}_{\mathbf{k}}$ located away from Γ. To see the problem caused by such a zero, take the one-dimensional case, and suppose that for some point $\bar{x}$, $u(\bar{x}) > 0$, but $u'(\bar{x}) = 0$, and that $\operatorname{sign}(u'(x)) = \operatorname{sign}(x - \bar{x})$. We will have $u(x)/u'(x) < 0$ for $x < \bar{x}$, and $u(x)/u'(x) > 0$ for $x > \bar{x}$. Thus,

$$H(u(x)/u'(x)) = H(x - \bar{x}), \text{ and so } \frac{d}{dx} H(u(x)/u'(x)) = \delta(x - \bar{x}).$$

Since our FDM algorithms are based on differencing approximate Heaviside functions, it is not surprising that the discrete version of this mechanism can produce spurious delta functions at points where $\det \nabla \vec{u} = 0$.

Fortunately, there is a simple fix for this problem. We set our approximate delta function at $\vec{x}_{\mathbf{k}}$ to zero if $\operatorname{sign}(\det \nabla^h \vec{u}_{\mathbf{i}})$ is not constant for all points $\vec{x}_{\mathbf{i}}$ close to $\vec{x}_{\mathbf{k}}$. Examples 8 and 10 in Section 4 are cases where this processing is necessary.

The PDF and PDFL algorithms can be combined with the FGN processing without causing any problems of the type described above.

3.2. Partial gradient normalization (PGN)

Clearly, the FGN process described above only makes sense if $m = n$. For problems where $n = 3$, $m = 2$, we can still define a gradient normalization process which makes the zero level sets of (a modified version of) $\vec{u}$ approximately orthogonal. Our approach is based on the Gram–Schmidt method for orthogonalizing a set of vectors. Since $m = 2$, there are two level set functions u and v. In this case we start by computing

$$\tilde{v}_{\mathbf{k}} = v_{\mathbf{k}} - c u_{\mathbf{k}}, \quad c = \frac{\nabla^h u_{\mathbf{k}} \cdot \nabla^h v_{\mathbf{k}}}{\nabla^h u_{\mathbf{k}} \cdot \nabla^h u_{\mathbf{k}}},$$

which makes $\nabla \tilde{v}_{\mathbf{k}}$ approximately orthogonal to $\nabla u_{\mathbf{k}}$ for $\vec{x}_{\mathbf{k}}$ near Γ.

We then normalize:

$$\hat{u}_{\mathbf{k}} = u_{\mathbf{k}} \Big/ \sqrt{\nabla^h u_{\mathbf{k}} \cdot \nabla^h u_{\mathbf{k}}}, \quad \hat{v}_{\mathbf{k}} = \tilde{v}_{\mathbf{k}} \Big/ \sqrt{\nabla^h v_{\mathbf{k}} \cdot \nabla^h v_{\mathbf{k}} - c^2 \nabla^h u_{\mathbf{k}} \cdot \nabla^h u_{\mathbf{k}}}, \tag{43}$$

so that, in addition to having approximately orthogonal level sets, $\hat{u}$ and $\hat{v}$ are approximately signed distance functions for $\vec{x}$ near Γ.

PGN enforces **G1** and **G3** for points $\vec{x}$ lying on Γ. The alignment requirement **G2** will not generally be satisfied.

Remark 3.1. For codimension one problems, where there is just one level set function $u : \mathbb{R}^n \mapsto \mathbb{R}^1$, the gradient normalization idea was already discussed by Engquist et al. [5]. They proposed replacing $u_{\mathbf{k}}$ by $v_{\mathbf{k}}$, where

$$v_{\mathbf{k}} = u_{\mathbf{k}}/\|\nabla^h u_{\mathbf{k}}\|. \quad (44)$$

Near the level set $u = 0$, the new level set function $v_{\mathbf{k}}$ is approximately a signed distance function.

4. Numerical examples

In this section, we describe a number of numerical experiments. In almost all of these examples (the exception is Example 10), the codimension is greater than one. See [19] for numerical experiments involving FDM1 and FDM2 in the codimension one setting.

The errors shown in the tables that follow are averages of absolute values of relative errors. The averages are taken over a number of runs, each run incorporating a small random grid shift. The number of runs required to obtain a reliable average varies from one example to another, and also from one technique to another. To avoid repetition, we will not give the number of runs required in each case, but simply note here that it could be as small as four and as large as 256.

In all cases where we used FDM1 or FDM1-FGN, the underlying approximate Heaviside function was $H^{C,\epsilon}$ with $\epsilon = 1.5h$.

We apply the FGN processing to the PDF algorithm rather than PDFL. In all cases, the rate of convergence of PDFL-FGN is the same as that for PDF-FGN, except that the errors are somewhat larger. In other words, FGN works better with the simpler PDF algorithm. For the PDF algorithm, we used $\delta^{L,\epsilon}$ with $\epsilon = 2h$, unless otherwise stated.

Finally, to simplify notation, we use (x,y) or (x,y,z) for $\vec{x}$, (u,v) or (u,v,w) for $\vec{u}$, (ϕ,ψ) for $\vec{\phi}$, etc.

Example 1. $(n = 2, m = 2)$ The purpose of this test is to study the effect of grid misalignment on approximation errors for a very simple codimension two problem in $\mathbb{R}^2$. This example applies to both $\mathcal{I}_1$ and $\mathcal{I}_2$, since they are equal in this particular case. We take $f(x,y) = 1$, and

$$\begin{aligned} u(x,y) &= \cos(\theta)\cdot x - \sin(\theta)\cdot y, \\ v(x,y) &= \sin(\theta)\cdot x + \cos(\theta)\cdot y \end{aligned} \quad (45)$$

with $\theta \in [0, \pi/2]$. With this setup, u and v are signed distance functions, and their gradients are orthogonal. The true value of the integral is $\mathcal{I}_1 = \mathcal{I}_2 = 1$. Our computations amount to checking the extent to which (26) is satisfied.

We tested PDF using the linear hat delta function $\delta^{L,\epsilon}$ with $\epsilon = h,\ 2h,\ \sqrt{h}$. We also ran with FDM1 and FDM2. Fig. 2 shows the errors as a function of θ. For FDM1 and FDM2, the errors are near the rounding error of the computer; gaps in the graphs indicate values of θ where the averaged absolute value of the relative error was zero. For the PDF method, the error is dependent on the angle θ. When $\theta = 0$ or $\theta = \pi/2$ (zero grid misalignment), the error is comparable to that of the FDM methods, but for intermediate values of θ, the error is much larger. Note that these are errors that will not decrease as h decreases, since this problem is purely linear. In other words, this example demonstrates a lack of consistency for the PDF algorithm. Since $\det \nabla\vec{u} \equiv 1$ for this problem, the $\epsilon = 2h$ version of PDF is equivalent to PDFL, so we are also seeing a lack of consistency for the PDFL algorithm.

Finally, although we do not show the results in Fig. 2, we also tested a product of approximate delta functions, as in (27), based on the codimension one versions of FDM1 and FDM2. The result is the same type of angle-dependent error (implying a lack of consistency) observed for the PDF algorithm. This shows the advantage of discretizing the wedge product formulation (the right side of (12)), as opposed to simply forming a product of approximate codimension-one delta functions (the left side of (12)).

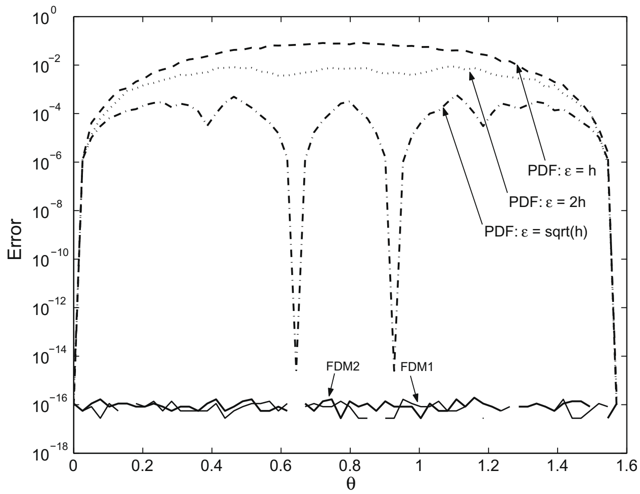


Fig. 2. Example 1. Error in approximating $\mathcal{I}_1 = \mathcal{I}_2$ for the case of $n = 2,\ m = 2$ as a function of grid misalignment.

Example 2. $(n = 2, m = 2)$ We are approximating $\mathcal{I}_1$ in this example. We take

$$u(x,y) = y - x(x-1), \quad v(x,y) = \frac{1}{2}x^2 - y, \quad f(x,y) = (x-1)^4 + (y+1)^2 - 1.$$

Before applying the grid, we rotated all coordinates by $\pi/4$. The curves $u = 0$ and $v = 0$ intersect at an angle of 45° at the single point (0,0) within the computational domain $[-1,1] \times [-1,1]$, and $f(0,0) = 1$, so the exact solution is $\mathcal{I}_1 = f(0,0) = 1$. Fig. 3 shows how the various level curves meet at $(x,y) = (0,0)$.

Table 1 shows that PDFL appears not to be consistent for this problem. However, PDF-FGN appears to converge at a rate of $O(h)$. FDM1 also converges at a rate of a little better than $O(h)$, and both FDM1-FGN and FDM2 seem to converge at a rate of $O(h^2)$. The FDM2-FGN algorithm is not shown in Table 1. For this example, it gives results very similar to FDM1-FGN.

Example 3. $(n = 3, m = 3)$ This is a linear problem in $\mathbb{R}^3$, with full codimension. We are computing both $\mathcal{I}_1$ and $\mathcal{I}_2$, since they are equal in this particular case. In this example there are three level set functions $u, v, w : \mathbb{R}^3 \mapsto \mathbb{R}^1$, and with $\vec{u}(\vec{x}) = (u(x,y,z), v(x,y,z), w(x,y,z))^T$,

$$\vec{u}(\vec{x}) = M\vec{x},$$

where the orthogonal matrix M is defined by

$$M = \begin{bmatrix} 0.12408589278267 & -0.31264694662283 & 0.94172956732798 \\ 0.73918615017622 & 0.66227231287279 & 0.12247129863678 \\ -0.66197169622272 & 0.68091649294869 & 0.31328294404654 \end{bmatrix}. \qquad (46)$$

The integrand is given by $f(x,y,z) = 1$, so the value of the integral is 1. We first ran FDM1, FDM2, and PDF on this problem. Note that for this linear problem with $\det \nabla \vec{u} \equiv 1$, PDF and PDFL are the same. Without FGN processing, none of FDM1, FDM2, PDF/PDFL converges to the exact solution. Table 2 shows that there are nonzero errors. These errors persist as we shrink the mesh, since the problem is linear. When we added the FGN processing to FDM1, FDM2, and PDF/PDFL, all of them gave the exact solution to within rounding error. Since each of the level set functions u, v, w is a signed distance function,

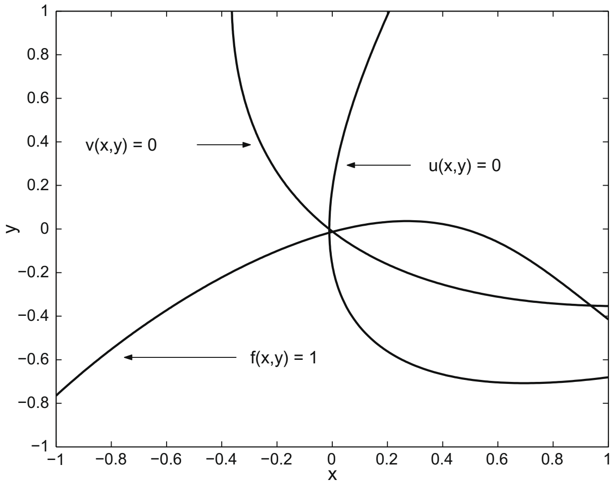


Fig. 3. Example 2. The level curves $u = 0$ and $v = 0$ intersect at an angle of 45°.

Table 1
Example 2.

h	FDM1		FDM1-FGN		FDM2		PDFL Error	PDF-FGN	
	Error	Rate	Error	Rate	Error	Rate		Error	Rate
0.04	1.21e−2		7.78e−3		1.77e−2		6.15e−2	9.00e−3	
0.02	3.15e−3	1.9	1.94e−3	2.0	4.49e−3	2.0	6.16e− 2	3.48e−3	1.4
0.01	7.71e−4	2.0	4.86e−4	2.0	1.12e−3	2.0	5.96e− 2	1.73e−3	1.0
0.005	1.79e−4	2.1	1.21e−4	2.0	2.77e−4	2.0	6.18e− 2	8.86e−4	.97
0.0025	5.42e−5	1.7	3.03e−5	2.0	6.89e−5	2.0	6.01e− 2	4.43e−4	1.0
0.00125	2.40e−5	1.2	7.59e−6	2.0	1.73e−5	2.0	6.00e− 2	2.29e−4	.95
0.000625	1.06e−5	1.2	1.89e−6	2.0	4.27e−6	2.0	5.89e−2	1.17e−4	.97

Table 2
Example 3.

h	FDM1 Error	FDM1-FGN Error	FDM2 Error	FDM2-FGN Error	PDFL Error	PDF-FGN Error
0.003125	8.11e−5	2.05e−16	1.10e−3	1.63e−16	4.45e−3	3.05e−16

Table 3
Example 4.

h	FDM1 Error	FDM2 Error	PDFL Error	FDM1-FGN		FDM2-FGN		PDF-FGN	
				Error	Rate	Error	Rate	Error	Rate
0.025	9.98e−4	2.36e−3	2.72e−3	5.69e−4		6.38e−4		4.07e−3	
0.025/2	2.36e−4	2.05e−3	3.10e−3	1.41e−4	2.0	1.58e−4	2.0	2.40e−3	0.76
0.025/4	7.83e−5	2.06e−3	3.26e−3	3.57e−5	2.0	4.00e−5	2.0	1.10e−3	1.1
0.025/8	8.08e−5	2.08e−3	3.88e−3	8.88e−6	2.0	1.00e−5	2.0	5.68e−4	0.95

and their level sets intersect orthogonally, we see that the difficulty posed by the problem is purely due to grid misalignment. The FGN processing is effective because it re-aligns the level sets with the grid.

Example 4. $(n = 3, m = 3)$ This is a full codimension problem in $\mathbb{R}^3$. We compute $\mathcal{I}_1$, but the basic results are applicable to the $\mathcal{I}_2$ version. In this example, there are three level set functions $u,\ v,\ w$ given by

$$\begin{aligned} u(x,y,z) &= \sqrt{(x-2)^2 + (y-3)^2 + (z-1)^2} - \sqrt{14}, \\ v(x,y,z) &= \sqrt{(x+1)^2 + (y-2)^2 + (z-3)^2} - \sqrt{14}, \\ w(x,y,z) &= \sqrt{(x+3)^2 + (y-1)^2 + (z+2)^2} - \sqrt{14}, \end{aligned} \tag{47}$$

and $f(x,y,z) = e^x \cos(y)\cos(z)$. We restrict the computational domain to a region near the origin, so that the three spherical level sets only intersect at $(x,y,z) = (0,0,0)$, and so $\mathcal{I}_1 = f(0,0,0) = 1$. Each of u, v, w is a signed distance function. The gradients are not orthogonal, but $\det \nabla\vec{u}(0,0,0) \approx 0.802$, so this problem is not in any way singular.

It is clear from Table 3 that none of FDM1, FDM2, PDFL converges to the true solution for this problem. However, FDM1, FDM2, and PDF appear to be consistent when the FGN processing is added.

Example 5. $(n = 3, m = 2)$ We compute $\mathcal{I}_1$ for this codimension two problem in $\mathbb{R}^3$. For this example

$$\begin{aligned} u(x,y,z) &= z - \beta_1(1 - (4x)^2 - y^2), \\ v(x,y,z) &= z - \beta_2(1 - (4x)^2 - y^2), \end{aligned} \tag{48}$$

where $\beta_1 = 1,\ \beta_2 = 1.01$. The zero level sets intersect in the xy-plane in the ellipse $(4x)^2 - y^2 = 1$. Note that with β_1 and β_2 so nearly equal, the level sets are very close to parallel. For the integrand, we used $f(x,y,z) = e^z(1 + x^2 + y^2)$, and the value of the integral (2) is ≈ 6.0417403.

We rotated all coordinates by the matrix A defined by

$$A = \begin{bmatrix} 1/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3} \\ 0 & -1/\sqrt{2} & 1/\sqrt{2} \\ 2/\sqrt{6} & -1/\sqrt{6} & -1/\sqrt{6} \end{bmatrix} \tag{49}$$

before applying the grid.

We tested FDM1, FDM2, and PDF with PGN processing. Without the PGN processing, none of the three methods gave useful results – this is due to the extreme collinearity of the level sets. However, as shown in Table 4, all of the methods gave what appears to be $O(h^2)$ approximations when used with the PGN processing.

Example 6. $(n = 3, m = 2)$ This is another codimension two problem in $\mathbb{R}^3$, and we compute $\mathcal{I}_1$ again. For this example

$$\begin{aligned} u(x,y,z) &= 1 - (x^2 + y^2), \\ v(x,y,z) &= x\sin(z) - y\cos(z), \end{aligned} \tag{50}$$

and

$$f(x,y,z) = \begin{cases} \sin^4(z/2 - \pi/2)\cos(v(x,y,z)/2), & -\pi < z < \pi, \\ 0, & \text{otherwise}. \end{cases} \tag{51}$$

Table 4
Example 5.

h	FDM1-PGN		FDM2-PGN		PDF-PGN	
	Error	Rate	Error	Rate	Error	Rate
0.125	9.40e−3		2.32e−2		1.19e−1	
0.125/2	3.75e−3	1.3	3.51e−3	2.7	2.73e−2	2.1
0.125/4	1.03e−3	1.9	6.94e−4	2.3	6.55e−3	2.1
0.125/8	4.71e−4	1.9	1.32e−4	2.4	1.61e−3	2.0

We rotated all coordinates by the matrix A (defined in Eq. (49) above) before applying the grid. The level set functions u and v are not signed distance functions, but ∇u and ∇v are orthogonal where the zero level sets intersect.

The intersection of the two zero level sets is a pair of helices. The integral can be computed in closed form, $\mathcal{I}_1 = 3\pi/\sqrt{2}$. Tables 5 and 6 show the results of our algorithm on this problem. It appears that all of FDM1, FDM2, FDM1-PGN, and FDM2-PGN approximations are converging at a rate of $O(h^2)$, with the PGN processing providing an improvement in accuracy for both FDM1 and FDM2.

The PDF algorithm appears from Table 5 not to be consistent. It is hard to tell from Table 6 whether or not the PDF-PGN algorithm is consistent.

Finally, for this example we also tested products of codimension one FDM1 and FDM2 algorithms, as in (27), both with and without PGN processing. In each case, we observed $O(h^2)$ convergence, but with errors larger (by a factor of 2 for the FDM1 algorithms, and a factor of 5 for the FDM2 algorithms) than those shown in Tables 5 and 6 for the corresponding wedge product versions.

Example 7. $(n = 3, m = 2)$ This is a final $\mathcal{I}_1$ computation of a codimension two problem in $\mathbb{R}^3$. For this example

$$u(x,y,z) = \begin{cases} R - |x|, & |y| \leqslant L/2 \\ R - \sqrt{x^2 + (y - L/2)^2}, & y \geqslant L/2 \\ R - \sqrt{x^2 + (y + L/2)^2}, & y \leqslant -L/2 \end{cases}, \quad v(x,y,z) = z. \tag{52}$$

Here $L = 2$, $R = 0.12\sqrt{2}$. We let $f(x,y,z) = \exp(-z^2)$. Before applying the mesh, we rotate all coordinates by 45° about the z-axis. This is basically the capsule example of [5,17], except that we have embedded the curve in $\mathbb{R}^3$ in a very simple way, and added the exponential integrand f. The value of the integral (2) is the length of the curve. For the codimension 1 version of this problem, pointwise defined approximate delta functions such as $\delta^{C,\epsilon}$ and $\delta^{L,\epsilon}$ are known to produce approximations to the arclength that are not consistent.

We compare FDM1, FDM2, and PDF in Table 7. For the PDF algorithm, we used $\delta^{C,\epsilon}$ with $\epsilon = 1.5\Delta x$. The FDM algorithms appear to be converging at a rate of about $O(h^2)$, while the PDF algorithm does not seem to be consistent. PGN processing does not significantly alter the results shown in Table 7, probably because the level set functions u and v already intersect orthogonally, and they are signed distance functions.

Table 5
Example 6 without PGN.

h	FDM1		FDM2		PDF	
	Error	Rate	Error	Rate	Error	Rate
0.4	1.91e−2		3.73e−2		3.56e−2	
0.2	4.25e−3	2.2	8.55e−3	2.1	9.21e−3	2.0
0.1	1.05e−3	2.0	2.15e−3	2.0	3.47e−3	1.4
0.05	2.90e−4	1.9	5.41e−4	2.0	2.36e−3	0.56

Table 6
Example 6 with PGN.

h	FDM1-PGN		FDM2-PGN		PDF-PGN	
	Error	Rate	Error	Rate	Error	Rate
0.4	8.14e−3		1.52e−2		2.13e−1	
0.2	1.07e−3	2.9	1.65e−3	3.2	3.00e−2	2.8
0.1	2.57e−4	2.1	4.16e−4	2.0	6.14e−3	2.3
0.05	6.36e−5	2.0	1.04e−4	2.0	2.15e−3	1.5

Table 7
Example 7.

h	FDM1		FDM2		PDF	
	Error	Rate	Error	Rate	Error	Rate
0.15	1.15e−1		1.18e−1		1.09e−1	
0.15/2	9.10e−3	3.7	1.41e−2	3.1	7.90e−3	3.8
0.15/4	2.18e−3	2.1	3.44e−3	2.0	2.83e−3	1.5
0.15/8	5.40e−4	2.0	8.65e−4	2.0	1.19e−3	1.3
0.15/16	1.43e−4	1.9	2.16e−4	2.0	2.37e−3	0

Example 8. ($n = 2, m = 2$) In this example we compute $\mathcal{I}_2$ for a full codimension problem in $\mathbb{R}^2$. We define

$$u(x,y) = -x^2 + x + y, \quad v(x,y) = \frac{1}{2}(x\cos\theta - y\sin\theta)^2 - (x\sin\theta + y\cos\theta), \tag{53}$$

where $\theta = \pi/4 - \pi/128$. For this example, the zero level sets intersect at the single point $(x,y) = (0,0)$ within the computational domain $= (-0.06, 0.06) \times (-0.06, 0.06)$, and at that point the Jacobian is $= \cos(\theta) - \sin(\theta)$. We take $f(x,y) = e^x \cos y$. The value of the integral $\mathcal{I}_2$ is

$$\mathcal{I}_2 = f(0,0)/|\det \nabla\vec{u}(0,0)| = 1/|\cos\theta - \sin\theta|.$$

Thus the Jacobian vanishes at $\theta = \pi/4$, and with our choice of $\theta = \pi/4 - \pi/128$, the problem is nearly singular because the zero level sets of u and v are nearly collinear.

In this example there is a spurious contribution to both the FDM1-FGN and FDM2-FGN delta function due to the fact that the Jacobian vanishes along a curve, see Fig. 4. This is the potential difficulty described in Section 3.1. We use the fix described in that section. For FDM1-FGN, we set the approximate delta function $\delta^h_{j,k}$ equal to zero if $\det \nabla^h \vec{u}_{j+r,k+s}$, do not all have the same sign for $-1 \leqslant r \leqslant 1,\ -1 \leqslant s \leqslant 1$. For FDM2-FGN, we perform the same check but for $-2 \leqslant r \leqslant 2,\ -2 \leqslant s \leqslant 2$. Without these fixes, neither FDM1-FGN nor FDM2-FGN gives useful results. The PDF-FGN algorithm is uneffected by the vanishing Jacobian, and does not require any fix.

When used without FGN, none of FDM1, FDM2, PDF, PDFL gave useful results – this is due to the near singularity of the problem. From Table 8, it seems that both FDM1-FGN and PDF-FGN are converging at a rate of about $O(h)$. FDM2-FGN appears to be converging at a rate of about $O(h^2)$.

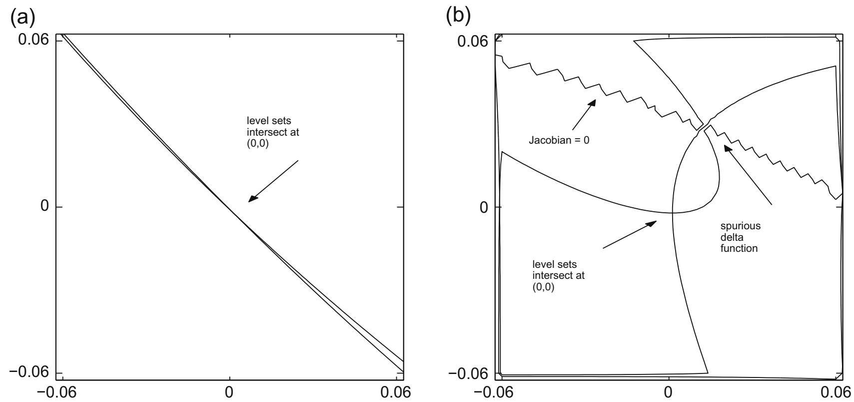


Fig. 4. Example 8. Zero level sets of u and v without (a) and with (b) FGN processing. In (b), the Jacobian vanishes along the jagged curve, potentially causing a spurious contribution to the FDM-FGN approximate delta function at the indicated point.

Table 8
Example 8.

h	FDM1-FGN		FDM2-FGN		PDF-FGN	
	Error	Rate	Error	Rate	Error	Rate
0.0025	1.96e−3		1.86e−3		1.57e−2	
0.0025/2	4.69e−4	2.1	4.61e−4	2.0	8.80e−3	0.84
0.0025/4	1.16e−4	2.0	1.13e−4	2.0	4.38e−3	1.0
0.0025/8	2.76e−5	2.1	2.84e−5	2.0	1.94e−3	1.2
0.0025/16	7.93e−6	1.8	7.14e−6	2.0	1.13e−3	0.78
0.0025/32	3.58e−6	1.1	1.79e−6	2.0	5.47e−4	1.0
0.0025/64	1.68e−6	1.1	4.40e−7	2.0	2.75e−4	1.0

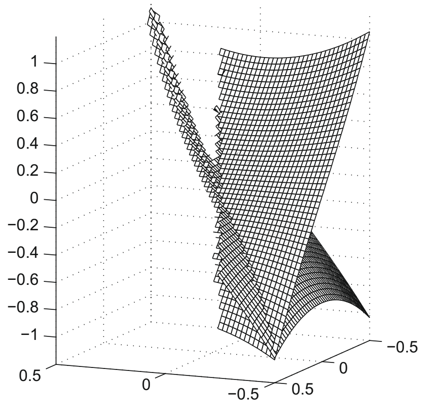
Example 9. $(n = 2, m = 2)$ This is another computation of $\mathcal{I}_2$ for a full codimension problem in $\mathbb{R}^2$. We try to model the situation near a caustic for the two-dimensional Schrödinger problem of Example 11, see Fig. 13. We take

$$
\begin{aligned}
u(x,y) &= x^3 + \alpha x - y^3 - \alpha y, \\
v(x,y) &= x^3 + \alpha x + y^3 + \alpha y,
\end{aligned}
\tag{54}
$$

and $f(x,y) = e^x \cos y$. There is a single intersection point at $(x,y) = (0,0)$, and $\mathcal{I}_2 = f(0,0)/2\alpha^2 = 1/2\alpha^2$. For this experiment, we vary the parameter α. As α approaches zero, the problem becomes more singular, since $\det \nabla\vec{u}(0,0) = 2\alpha^2$, see Fig. 5. The singularity is due to the fact that the gradients ∇u and ∇v become small, as opposed to the situation where the gradients are nearly parallel (which was the case in the previous example). In fact the gradients are orthogonal at the point of intersection for this example.

Table 9 shows the results of running with $h = 0.01$. In all cases, the accuracy decreases as the problem becomes more singular (as α decreases). The FGN processing significantly improves the accuracy of both FDM1 and PDFL for small α. For this problem, the FDM2 algorithm seems to be slightly more accurate without the FGN processing. For small values of α, FDM1, FDM2, and PDFL all underestimate the true solution. When we add FGN processing, they overestimate the true solution. These observations are relevant for Example 11, specifically the approximations shown in Figs. 10–12.

Finally, we check for consistency in the nearly singular case $\alpha = 1/256$, corresponding to the last row of Table 9. The results are shown in Table 10. All of the FGN algorithms appear to be converging at a rate of $O(h^2)$, as does FDM2. FDM1 and PDFL are possibly converging, but at a very slow rate. It is interesting that for this nearly singular example, PDF-FGN gives the best results.



Example 10. In this example, we are computing observables for the one-dimensional Schrödinger equation as described in Section 1.1. More specifically, this is basically Example 6.1 of [9]. This problem has zero potential, $V = 0$, and thus referring to (8), we must solve two Liouville equations

$$w_t + pw_x = 0, \tag{55}$$

one for $w = \phi$ and one for $w = f$, with initial data

$$\begin{aligned} &\phi(x,0) = -\sin(\pi x)|\sin(\pi x)|, \\ &f(x,0) = \exp(-(x-0.5)^2). \end{aligned} \tag{56}$$

We discretize the rectangle $[-1,1]\times[-4,4]$ of x,p space, using $x_j = j\Delta x,\ p_k = k\Delta p$, with $|j| \leqslant J$ and $|k| \leqslant K$ and $J\Delta x = 1,\ K\Delta p = 4$.

Given a time level $t > 0$, we compute for each mesh point (x_j, p_k)

$$\phi(x_j,p_k,t) = \phi(x_j - p_k t, p_k, 0), \quad f(x_j,p_k,t) = f(x_j - p_k t, p_k, 0). \tag{57}$$

This gives the exact solution to the Liouville equations (55). In the general case where V is not so simple, one would use a finite difference scheme to generate approximate solutions [9].

We then compute approximations $\bar{\rho}^h,\ \bar{v}^h$ to the observables $\bar{\rho}$ and $\bar{v}$:

$$\begin{aligned} &\bar{\rho}^h(x_j,t) = h\sum_k f(x_j,p_k,t)\delta^h(p_k;\phi(x_j,\cdot,t)), \\ &\bar{v}^h(x_j,t) = \frac{h}{\bar{\rho}^h(x_j,t)}\sum_k p_k f(x_j,p_k,t)\delta^h(p_k;\phi(x_j,\cdot,t)), \end{aligned} \tag{58}$$

where $h = \Delta p$, and $\delta^h(p_k;\phi(x_j,\cdot,t))$ is an approximate delta function based on the level function $p \mapsto \phi(x_j,p,t)$. In other words, for a fixed time t, and a fixed x-gridpoint x_j, we compute a one-dimensional approximate delta function algorithm in the p-dimension.

The computations were done with $\Delta x = 0.01,\ \Delta p = 0.02$, and the plots in Figs. 6 and 7 show the solutions at time $t = 0.625$. In Fig. 6, plots (a) and (b) show the results using FDM1-FGN. Plots (c) and (d) show the results with FDM2. Plots (e) and (f) show FDM1-FGN with a refined mesh, $\Delta x = 0.0025,\ \Delta p = 0.0050$.

When we apply FDM1-FGN to this problem, the difficulty described in Section 3.1 arises because $\partial_p\phi$ has numerous zeros. If we use FDM1-FGN without some sort of fix, the result is unusable due to spurious contributions to the approximate delta function. The specific fix that we used was to set the FDM1-FGN approximate delta function at (x_j,p_k) equal to zero if the differences $\phi(x_j,p_{k+1}) - \phi(x_j,p_k)$ and $\phi(x_j,p_k) - \phi(x_j,p_{k-1})$ do not have the same sign.

Fig. 7 shows the results with PDFL and PDF-FGN. Plots (e) and (f) show PDFL with a refined mesh, $\Delta x = 0.0025,\ \Delta p = 0.0050$.

Example 11. This experiment is based on Example 6.3 of [9]. The setup is that of Section 1.1 with $n = 2$. We use the notation $\vec{x} = (x,y),\ \vec{p} = (p,q),\ \vec{\phi} = (\phi,\psi)$. For this problem, the potential is quadratic, $V(x,y) = (x^2+y^2)/2$, and the Liouville equations for $w = \phi,\ \psi,\ f$ take the form

$$w_t + pw_x + qw_y - xw_p - yw_q = 0. \tag{59}$$

With the notation $\vec{z} = (x,y,p,q)^T$, the superscript T denoting matrix transpose, the bicharacteristic equations associated with (59) satisfy the linear system of differential equations

$$\frac{d}{dt}\vec{z} = \Omega\vec{z}, \tag{60}$$

where

$$\Omega = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}, \quad e^{\Omega t} = \begin{bmatrix} \cos t & 0 & \sin t & 0 \\ 0 & \cos t & 0 & \sin t \\ -\sin t & 0 & \cos t & 0 \\ 0 & -\sin t & 0 & \cos t \end{bmatrix}. \tag{61}$$

Using the matrix $\exp(\Omega t)$, we can solve the PDE (59) in closed form by tracing the bicharacteristics back to the initial manifold at $t = 0$:

$$w(\vec{z},t) = w(\vec{z}^0,0), \quad \vec{z}^0 = e^{-\Omega t}\vec{z}. \tag{62}$$

As in the previous example, we are able to solve the Liouville equations in this simple manner due to the special form of the potential V. In general, it is necessary to solve the PDE's using numerical methods, as described in [9].

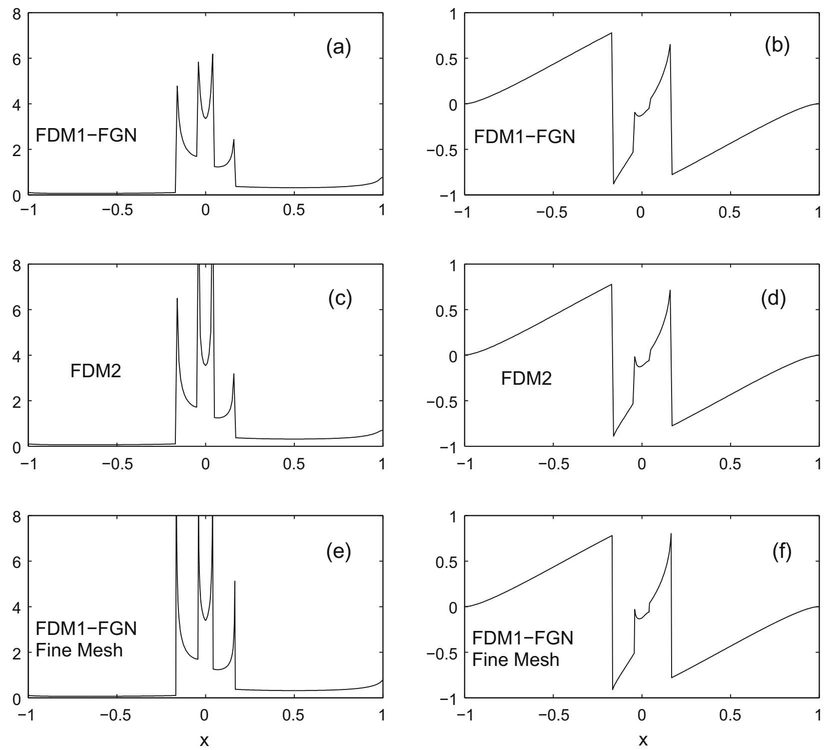


Fig. 6. Example 9. Observables computed using FDM algorithms. Plots on the left show $\bar{\rho}^h$, and plots on the right show $\bar{v}^h$.

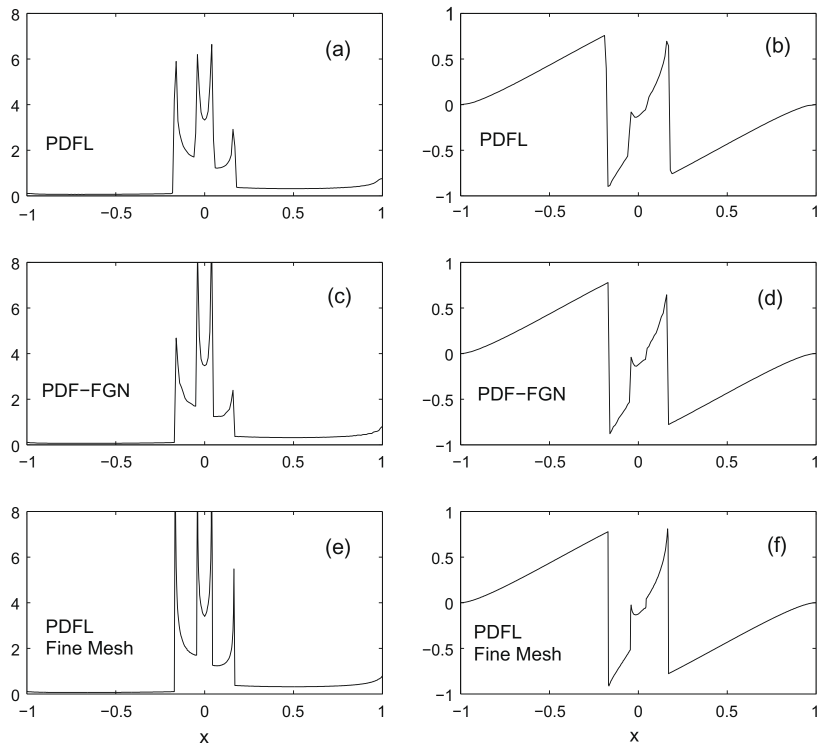
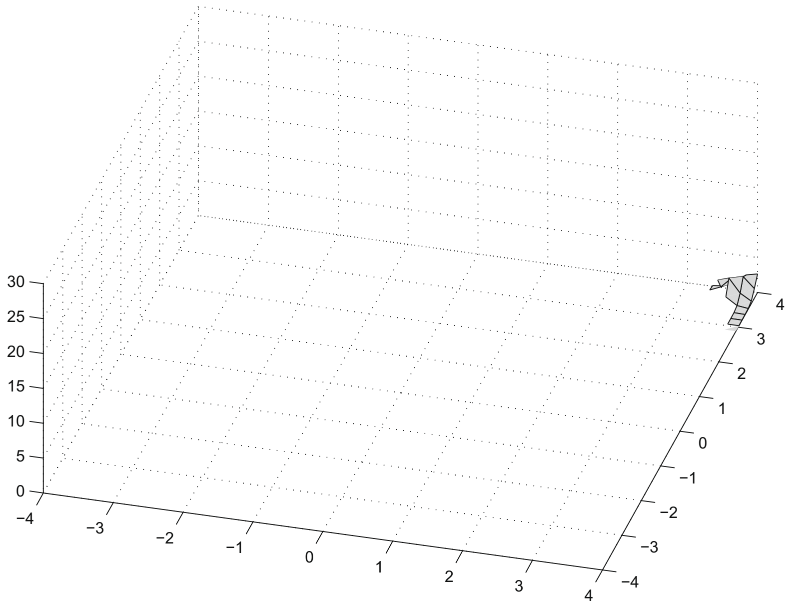


Fig. 7. Example 9. Observables computed using PDF algorithms. Plots on the left show $\bar{\rho}^h$, and plots on the right show $\bar{v}^h$.


30
25
20
15
10
5
0
−4
−3
−2
−1
0
1
2
3
4
−4
−3
−2
−1
0
1
2
3
4

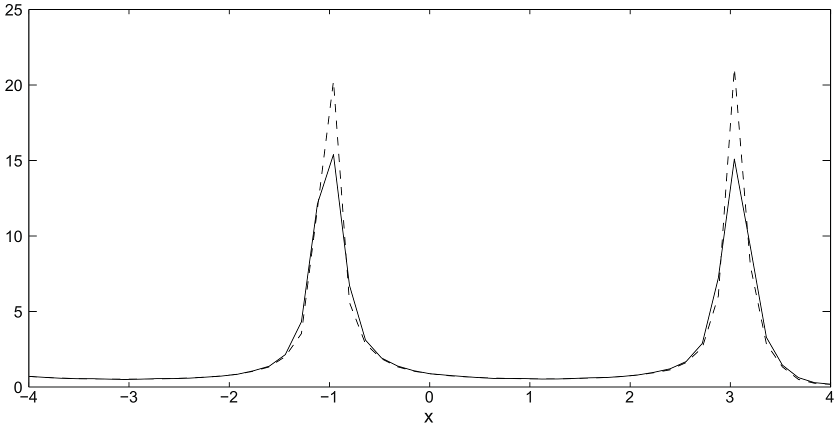


Fig. 11. Example 11. Averaged density $\bar{\rho}^h$ at $t = 6.9$, $y = -0.96$. PDFL (solid line) and PDF-FGN (dashed line).

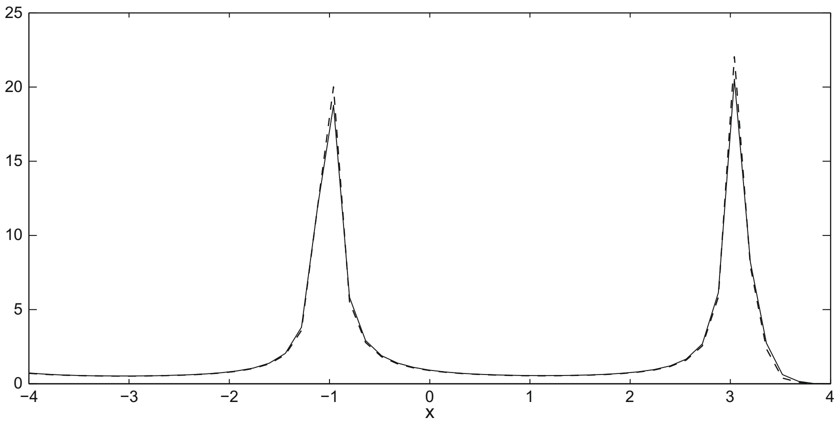


Fig. 12. Example 11. Averaged density $\bar{\rho}^h$ at $t = 6.9$, $y = -0.96$. FDM2 (solid line) and FDM2-FGN (dashed line).

The initial data for the Liouville equations is defined by (9) and

$$S_0(x,y) = 0.6(\sin(0.4\pi x) - 0.1)(\sin(0.4\pi y) - 0.2),\\ \rho_0(x,y) = \exp(-(x^2 + y^2)) + 1. \tag{63}$$

We discretize the rectangle $[-4,4]^2 \times [-7.5]^2$ of $x,\ y,\ p,\ q$ space, using $x_j = j\Delta x,\ y_k = k\Delta x$ (we are taking $\Delta x = \Delta y$), and $p_l = lh,\ q_m = mh$ (we are taking $\Delta p = \Delta q = h$) with $|j|,\ |k| \leqslant J$ and $|l|,\ |m| \leqslant L$ and $J\Delta x = 4,\ Lh = 7.5$.

At a given time level $t > 0$, and each grid point (x_j, y_k, p_l, q_m), we use formula (62) to compute

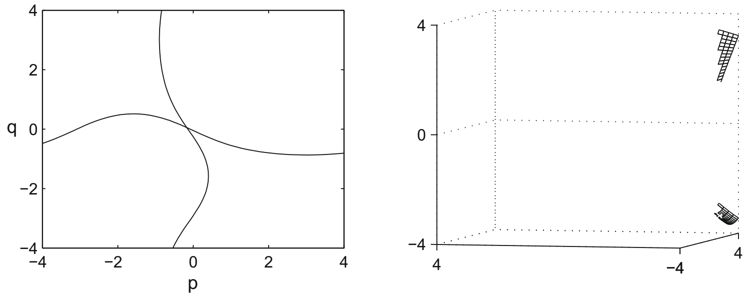
$$w(x_j, y_k, p_l, q_m, t) = w_0(x_j^0, y_k^0, p_l^0, q_m^0), \quad (x_j^0, y_k^0, p_l^0, q_m^0)^T = e^{-\Omega t}(x_j, y_k, p_l, q_m)^T. \tag{64}$$

We then compute the (approximate) observables, $\bar{\rho}^h \approx \bar{\rho},\ \bar{v}^{i,h} \approx \bar{v}^i,\ i = 1,2$:

$$\begin{aligned} \bar{\rho}^h(x_j, y_k, t) &= h^2 \sum_l \sum_m f(x_j, y_k, p_l, q_m, t)\delta^h(p_l, q_m; \vec{\phi}(x_j, y_k, \cdot, \cdot, t)),\\ \bar{v}^{1,h}(x_j, y_k, t) &= \frac{h^2}{\bar{\rho}^h(x_j, y_k, t)} \sum_l \sum_m p_l f(x_j, y_k, p_l, q_m, t)\delta^h(p_l, q_m; \vec{\phi}(x_j, y_k, \cdot, \cdot, t)),\\ \bar{v}^{2,h}(x_j, y_k, t) &= \frac{h^2}{\bar{\rho}^h(x_j, y_k, t)} \sum_l \sum_m q_m f(x_j, y_k, p_l, q_m, t)\delta^h(p_l, q_m; \vec{\phi}(x_j, y_k, \cdot, \cdot, t)). \end{aligned} \tag{65}$$

Here $\delta^h(p_l, q_m; \vec{\phi}(x_j, y_k, \cdot, \cdot, t))$ is an approximate delta function based on the level function $(p,q) \mapsto \vec{\phi}(x_j, y_k, p, q, t)$.

For our computations, we used $\Delta x = \Delta y = 0.16$ and $h = \Delta p = \Delta q = 0.15$. The averaged density $\bar{\rho}^h$ at $t = 6.9$ is shown Fig. 8, and the averaged velocity components $\bar{v}^{i,h}$ at the same time slice are shown in Fig. 9. The approximate delta function used to generate these plots was FDM1-FGN using $H^{C,\epsilon}$.



Figs. 10–12 show the averaged density $\bar{\rho}^h$ at $t = 6.9$, along the slice $y = -0.96$. Based on the results of Example 9, it is likely that the versions of the algorithms without FGN processing are tending to underestimate the maxima that occur near the caustics, while the FGN versions are probably giving small overestimates.

Finally, in Fig. 13 we focus on the geometry of the level sets $\phi = 0$ and $\psi = 0$ first far from any caustics (plots (a) and (b)), and then near a caustic (plots (c) and (d)). The caustics correspond to the spikes in Fig. 8, and occur at points (x, y) where the Jacobian det $\nabla\vec{\phi}$ vanishes. What is interesting about plots (c) and (d) is that the (near) vanishing Jacobian is due to flattening of the level functions, as opposed to collinearity of the contours.

5. Conclusion

Using the wedge product formalism combined with finite differencing, we have generalized to higher codimension the methods (FDM1 and FDM2) for discretizing delta functions in [19]. We have used these algorithms to approximate certain integrals involving delta functions. We have also proposed a gradient normalization process that can be combined with our FDM algorithms, and also with more standard delta function approximations. We have presented a number of numerical experiments indicating that our FDM algorithms are accurate and often consistent (meaning that they converge to the true solution when $h \to 0$). Our experiments indicate that in those cases where FDM1 and/or FDM2 is inconsistent, consistency can be recovered with our new gradient normalization process. In the full codimension setting, the gradient normalization process works in a similar way with more standard delta function approximations. In a number of cases, our numerical experiments indicate $O(h^2)$ convergence for our new FDM algorithms, especially when combined with gradient normalization. In particular, the FDM2 algorithm, when combined with gradient normalization (FGN or PGN depending on the problem), appears to be second order accurate in all of the test cases where there is a known solution to compare against. (Examples 10 and 11 are the ones where a known solution is not available.)

With one exception, our tests indicate that the FDM algorithms are more accurate than the PDF/PDFL algorithms. The one exception is Example 9, where for a nearly singular problem, the three algorithms FDM1-FGN, FDM2-FGN, and PDF-FGN all give very similar results. Below we make some specific recommendations concerning choice of algorithm.

For full codimension problems (two level set functions in $\mathbb{R}^2$ or three level set functions in $\mathbb{R}^3$), we recommend the FDM2-FGN algorithm. Our numerical tests indicate $O(h^2)$ convergence, even for problems that are close to singular. For two-dimensional problems that are far from singular, FDM2 (without FGN) reliably gives second order accuracy. However, for three-dimensional problems, neither of the FDM algorithms should be used alone – the FGN processing should always be incorporated. A good alternative to FDM2-FGN that runs faster and is easier to code is FDM1-FGN. It appears to be second order accurate for data that is far from singular, but on one of our nearly singular test problems it was only first order accurate. This is why we consider FDM2-FGN the single best choice for full codimension problems. Finally, for full codimension problems arising in high frequency wave propagation, where caustics may cause nearly singular data, the FGN processing should always be used.

Concerning the PDF and PDFL algorithms in the full codimension setting, we recommend that they always be combined with gradient normalization (FGN in this case). For full codimension problems, the addition of gradient normalization appears to remedy those situations where these algorithms are inconsistent. The resulting combined algorithms seem to be at least first order accurate, and in some cases second order accurate. The PDF-FGN algorithm is particularly easy to implement, and runs faster than the FDM-FGN algorithms. Thus, for full codimension problems, if ease of coding and/or processing speed is more important than accuracy, the gradient-normalized PDF algorithms may be preferred over their FDM counterparts.

For codimension two problems (two level set functions in $\mathbb{R}^3$), we recommend FDM1-PGN. This algorithm seems to be second order accurate. The FDM2-PGN algorithm also seems to be second order accurate. Since FDM1-PGN gives similar results, but runs faster and is easier to implement, we view it as superior for this type of problem. We do not recommend PDF algorithms for codimension two problems. Even when the PGN processing is incorporated, these algorithms may not be consistent.

Acknowledgments

I thank the anonymous referees. Their comments helped to improve the paper.

Appendix A. Converting from wedge products to determinants

This appendix provides alternative formulas, using determinants instead of wedge products, for the basic formulas of this paper. These formulas, along with the recipe (13) for the discrete gradient operator, can be used directly (bypassing the wedge products if one wishes) when coding FDM1 and FDM2.

If $A = [a_{i,j}]$ is a $m \times n$ matrix with $m \leqslant n$, the quantity $J_{\vec{i}}(A)$ is defined by

$$J_{\vec{i}}(A) := \det \begin{bmatrix} a_{1,i_1} & a_{1,i_2} & \cdots & a_{1,i_m} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m,i_1} & a_{m,i_2} & \cdots & a_{m,i_m} \end{bmatrix}. \tag{66}$$

Here $\vec{i} \in S$ refers to all multi-indices $\vec{i} = (i_1, \ldots, i_m)$ of the form $1 \leqslant i_1 < i_2 < \cdots < i_m \leqslant n$. With this notation, formula (12) for the delta function product can be written as

$$\delta(u^1) \cdots \delta(u^m) = \frac{\sum_{\vec{i} \in S} J_{\vec{i}}(\nabla \vec{H}(\vec{u})) J_{\vec{i}}(\nabla \vec{u})}{\sum_{\vec{i} \in S} (J_{\vec{i}}(\nabla \vec{u}))^2}. \tag{67}$$

The gradients $\nabla \vec{u}$ and $\nabla \vec{H}(\vec{u})$ are $m \times n$ matrices:

$$\nabla \vec{u} = \begin{bmatrix} \nabla u^1 \\ \cdots \\ \nabla u^m \end{bmatrix}, \quad \nabla \vec{H}(\vec{u}) = \begin{bmatrix} \nabla H(u^1) \\ \cdots \\ \nabla H(u^m) \end{bmatrix}. \tag{68}$$

The quantity $\|\wedge_m \nabla \vec{u}(\vec{x})\|$ appearing in (1) can be written as

$$\|\wedge_m \nabla \vec{u}(\vec{x})\| = \sqrt{\sum_{\vec{i} \in S} (J_{\vec{i}}(\nabla \vec{u}))^2}. \tag{69}$$

The formulas (67 and 69) can be combined:

$$\delta(u^1) \cdots \delta(u^m) \|\wedge_m \nabla \vec{u}(\vec{x})\| = \frac{\sum_{\vec{i} \in S} J_{\vec{i}}(\nabla \vec{H}(\vec{u})) J_{\vec{i}}(\nabla \vec{u})}{\sqrt{\sum_{\vec{i} \in S} (J_{\vec{i}}(\nabla \vec{u}))^2}}. \tag{70}$$

We collect below the versions of (67 and 69) that result in certain specific cases.

Full codimension: $m = n$. In this case,

$$\delta(u^1) \cdots \delta(u^n) = \det \nabla \vec{H}(\vec{u}) / \det \nabla(\vec{u}), \tag{71}$$

and

$$\delta(u^1) \cdots \delta(u^n) \|\wedge_n \nabla \vec{u}\| = \det \nabla \vec{H}(\vec{u}) \cdot \operatorname{sign}(\det \nabla(\vec{u})). \tag{72}$$

Codimension 1: $m = 1$. In this case, we write $u^1 = u$, and the formulas are

$$\delta(u) = \frac{\nabla H(u) \cdot \nabla u}{\|\nabla u\|^2}, \tag{73}$$

and

$$\delta(u) \|\wedge_1 \nabla u\| = \frac{\nabla H(u) \cdot \nabla u}{\|\nabla u\|}. \tag{74}$$

Curves in $\mathbb{R}^3$: $n = 3,\ m = 2$. The formulas for this case are

$$\delta(u^1)\delta(u^2) = \frac{(\nabla H(u^1) \times \nabla H(u^2)) \cdot (\nabla u^1 \times \nabla u^2)}{\|\nabla u^1 \times \nabla u^2\|^2}, \tag{75}$$

and

$$\delta(u^1)\delta(u^2)\|\wedge_2 \nabla\vec{u}\| = \frac{(\nabla H(u^1) \times \nabla H(u^2)) \cdot (\nabla u^1 \times \nabla u^2)}{\|\nabla u^1 \times \nabla u^2\|}. \tag{76}$$

References

[1] R.P. Beyer, R.J. Leveque, Analysis of a one-dimensional model for the immersed boundary method, SIAM J. Numer. Anal. 29 (1992) 332–364.
[2] P. Burchard, L.-T. Cheng, B. Merriman, S. Osher, Motion of curves in three spatial dimensions using a level set approach, 2001. <http://www.math.ucla.edu/applied/cam/index.html>.
[3] L.-T. Cheng, P. Burchard, B. Merriman, S. Osher, Motion of curves constrained on surfaces using a level set approach, J. Comput. Phys. 175 (2) (2002) 604–644.
[4] L.-T. Cheng, H. Liu, S. Osher, Computational high-frequency wave propagation using the level set method, with applications to the semi-classical limit of Schrödinger equations, Commun. Math. Sci. 1 (2003) 593–621.
[5] B. Engquist, A.K. Tornberg, R. Tsai, Discretization of dirac delta functions in level set methods, J. Comput. Phys. 207 (2005) 28–51.
[6] H. Flanders, Differential Forms with Applications to the Physical Sciences, Dover Publications, New York, 1989.
[7] S. Jin, S. Osher, A level set method for the computation of multivalued solutions to quasilinear hyperbolic PDE's and Hamilton–Jacobi equations, Commun. Math. Sci. 1 (3) (2003) 575–591.
[8] S. Jin, H. Liu, S. Osher, R. Tsai, Computing multivalued physical observables for the semiclassical limit of the Schrödinger equation, J. Comput. Phys. 205 (2005) 222–241.
[9] H. Liu, S. Osher, R. Tsai, Multi-valued solution and level set methods in computational high frequency wave propagation, Commun. Comput. Phys. 1 (5) (2006) 765–804.
[10] C. Misner, Spherical harmonic decomposition on a cubic grid, 1999. arxiv:gr-qc/9910044 v1.
[11] S. Osher, R. Fedkiw, Level Set Methods and Dynamic Implicit Surfaces, Springer-Verlag, New York, 2003.
[12] S. Osher, J. Sethian, Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. 79 (1988) 12–49.
[13] D. Peng, B. Merriman, S. Osher, H. Zhao, M. Kang, A PDE-based fast local level set method, J. Comput. Phys. 155 (1999) 410–438.
[14] O. Runborg, Mathematical models and numerical methods for high frequency waves, Commun. Comput. Phys. 2 (5) (2007) 827–880.
[15] J.A. Sethian, Level Set Methods and Fast Marching Methods, Cambridge University Press, Cambridge, 1999.
[16] P. Smereka, The numerical approximation of a delta function with application to level set methods, J. Comput. Phys. 211 (2006) 77–90.
[17] A.K. Tornberg, B. Engquist, Numerical approximations of singular source terms in differential equations, J. Comput. Phys. 200 (2004) 462–488.
[18] A.K. Tornberg, B. Engquist, Regularization techniques for numerical approximation of PDEs with singularities, J. Sci. Comput. 19 (2003) 527–552.
[19] J.D. Towers, Two methods for discretizing a delta function supported on a level set, J. Comput. Phys. 220 (2007) 915–931.
[20] H.K. Zhao, T. Chan, B. Merriman, S. Osher, Variational level set approach to multiphase motion, J. Comput. Phys. 127 (1996) 179–195.