

## NUMERICAL RECOVERY OF THE SIGNED DISTANCE FUNCTION

TOMÁŠ OBERHUBER<sup>1</sup>

**Abstract.** We test several methods for computing the signed distance function which is very useful not just for the level set methods. Both approaches, evolutionary based on the equation  $\Phi_t = \text{sign}\Phi_0(1 - |\nabla\Phi|)$  and direct based on the eikonal equation  $|\nabla\Phi| = 1$ , are compared. We also derive new algorithm to solve more general problem  $|\nabla u(x)| = F(x)$  based on the fast sweeping method. This algorithm is more efficient than the fast sweeping method mainly for the narrow band methods. We show some numerical examples even on practical problems.

**Key words.** free boundary problems, signed distance function, viscosity solution, Hamilton-Jacobi equation, level set methods, fast sweeping method, fast marching method, monotone schemes, upwind scheme, Godunov scheme, eikonal equation

**AMS subject classifications.** 35F25, 65B99, 65D99, 68U10, 68W25

**1. Introduction.** The free boundary problems are one of the most important branch in the theory of partial differential equations. There are several approaches how to model evolving curves or surfaces - front tracking method, phase field model ([3], [4]) and level set methods. Some comparisons can be found for example in [13], [14] or [5]. For the level set methods the signed distance function to given curve (surface) is usually taken as the initial condition. Since the property of being the signed distance function to evolving curve (surface) is usually lost after few iterations we need some efficient algorithm to recover the function back. In this article we deal just with structured meshes (for level set methods on unstructured meshes, see [17]).

**2. The definition of the signed distance function.** We consider a closed set  $\Gamma = \partial\Omega_S$  where  $\Omega$  is bounded domain  $\Omega_S \subset \Omega \subset \mathbb{R}^n$ ,  $n = 1, 2, 3$ .  $\Gamma$  can be given as manifold using a function  $\Phi_0$  with additional properties:  $\Gamma = \{x \in \mathbb{R}^n \mid \Phi_0(x) = 0\}$ ,  $\Phi_0(x) < 0$  for  $x \in \text{Int}\Gamma$ ,  $\Phi_0(x) > 0$  for  $x \notin \text{Int}\Gamma$  and  $\Phi_0$  is Lipschitz continuous.

Main purpose is to find an accurate approximation of the signed distance function  $d_\Gamma$ . That means

$$d_\Gamma(x) = \begin{cases} \rho(x, \Gamma); & x \notin \text{Int}\Gamma & = \Omega \setminus \Omega_S \\ -\rho(x, \Gamma); & x \in \text{Int}\Gamma & = \Omega_S \end{cases} .$$

Recent research results showed that the theory of PDE offers an efficient method for getting the signed distance function (SDF). From the PDE point of view the definition as it was mentioned above is not entirely convenient. To describe SDF in terms of PDE let us do simple observation in 1D case. Consider  $\Omega \equiv (0, 1)$  and  $\text{Int}\Gamma \equiv (0, 1)$  which gives  $d_\Gamma(x) = |x - \frac{1}{2}| - \frac{1}{2}$ . Now we differentiate  $d_\Gamma$  with respect to  $x$  for  $x \neq 0$  and we observe that  $|\frac{\partial d_\Gamma}{\partial x}| = 1$  for all  $x \neq 0$ . So we may try to define SDF as function  $f$  satisfying  $|\nabla f| = 1$  a. e. and  $f \equiv 0$  on  $\Gamma$ . However there are some difficulties. First of all  $d_\Gamma$  is not differentiable everywhere. Moreover function  $f$  satisfying  $|\nabla f| = 1$  a.e. does not have to be SDF - see (2.1).

---

<sup>1</sup>Department of Mathematics, Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University in Prague, Trojanova 13, 120 00 Prague, Czech Republic.

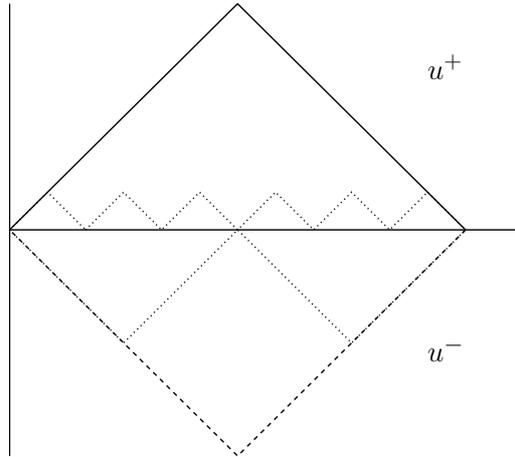


FIG. 2.1. Several functions for which  $|\nabla f| = 1$  holds almost everywhere. But only  $u^+$  or  $u^-$  can be SDF.

To overcome these difficulties we need some kind of weak solution which will ensure uniqueness. Such property offers so called viscosity solution.

**3. Viscosity solution.** The theory of viscosity solution was introduced in 1980s by Crandall and Lions [10]. It deals with so called Hamilton-Jacobi equations of the form

$$(3.1) \quad F(x, u(x), Du(x)) = 0.$$

In general, the theory of viscosity solution ([2], [11] and [7]) can be applied even for the second order PDE's but it is not our case. In this article we will use the following definition:

DEFINITION 3.1. Let  $\Omega$  be an open subset of  $\mathbb{R}^n$ ,  $F$  be a mapping  $F : \Omega \times \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}$ . We say  $u$  is a viscosity subsolution of (3.1) in  $\Omega$  if it is upper semicontinuous and for each  $\varphi \in C^2(\Omega)$  and local maximum point  $x_0 \in \Omega$  of  $u - \varphi$  we have

$$F(x_0, u(x_0), D\varphi(x_0)) \leq 0.$$

We say  $u$  is upper semicontinuous if for all  $x \in \Omega$   $\limsup_{y \rightarrow x} u(y) \leq u(x)$ .

DEFINITION 3.2. Let  $\Omega$  be an open subset of  $\mathbb{R}^n$ ,  $F$  be a mapping  $F : \Omega \times \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}$ . We say  $u$  is a viscosity supersolution of (3.1) in  $\Omega$  if it is lower semicontinuous and for each  $\varphi \in C^2(\Omega)$  and local minimum point  $x_0 \in \Omega$  of  $u - \varphi$  we have

$$F(x_0, u(x_0), D\varphi(x_0)) \geq 0.$$

We say  $u$  is lower semicontinuous if for all  $x \in \Omega$   $\liminf_{y \rightarrow x} u(y) \geq u(x)$ .

DEFINITION 3.3.  $u$  is a viscosity solution of (3.1) if it is both viscosity subsolution and viscosity supersolution.

Notice that if  $u$  is differentiable, we have  $D(u - \varphi)(x_0) = 0 \Rightarrow Du(x_0) = D\varphi(x_0)$  and so  $0 = F(x_0, u(x_0), Du(x_0)) = F(x_0, u(x_0), D\varphi(x_0))$ . So our definition holds for classical solution.

The following statement holds ([15]):

THEOREM 3.4. For  $\Gamma$  Lipschitz continuous  $d_\Gamma$  is also Lipschitz continuous and is a viscosity solution of the eikonal equation

$$|Du| = 1.$$

Now we will show the meaning of the viscosity solution in a simple 1D case (see also. [15], [8]).

Consider function  $u^+ = 1 - |x|$  in  $\Omega = (-1, 1)$  and  $u^+(-1) = u^+(1) = 0$ . We will show that it is a viscosity solution of  $|Du| = 1$ . We need to show that for all  $\varphi \in C^\infty$  if  $u^+ - \varphi$  has a maximum at  $x_0$  then  $|D\varphi(x_0)| \leq 1$ . It is clear for all  $x \in \Omega$ ,  $x_0 \neq 0$ . Assume  $u^+ - \varphi$  has a maximum at  $x_0 = 0$  and  $|D\varphi(x_0)| > 1$  then since  $\varphi \in C^\infty$  there exists neighbourhood  $H_{x_0}$  such that  $D\varphi > 1$  or  $D\varphi < -1$  for all  $x \in H_{x_0}$ . If  $D\varphi > 1$  then there exists  $x \in H_{x_0}$  and  $x < x_0$  such that  $(u^+ - \varphi)(x) > (u^+ - \varphi)(x_0)$  which is a contradiction. For  $D\varphi < -1$  we can get contradiction in very similar way. To complete the proof consider the case when  $u^+ - \varphi$  has a minimum at some  $x_0 \in \Omega$  for  $\varphi \in C^\infty$ . We must show  $|D\varphi(x_0)| \geq 1$ . It is clear for  $x_0 \neq 0$ . We will just show that  $u^+ - \varphi$  can not have minimum at  $x_0 = 0$ . Indeed we have  $1 = \lim_{x \uparrow 0} Du^+ \leq \lim_{x \uparrow 0} D\varphi = \lim_{x \downarrow 0} D\varphi \leq \lim_{x \downarrow 0} Du^+ = -1$  which is a contradiction.

Now let us show that  $u_- = |x| - 1$  in  $\Omega = (-1, 1)$  and again  $u^-(-1) = u^-(1) = 0$  is not a viscosity solution of  $|Du| = 1$ . For  $\varphi = x^2 - 1$   $u^- - \varphi$  has minimum at  $x_0 = 0$  but  $D\varphi(x_0) = 0$  which is a contradiction with  $|D\varphi(x_0)| \geq 1$ .

One can easily show that  $u^-$  is a viscosity solution of  $-|Du| = -1$ . It may look a little bit strange that multiplying an equation by  $-1$  can change the solution but it is not surprising since the solution is defined by inequalities.

So now we can see that the viscosity solution is exactly what we want. The important thing is that the viscosity solution of  $|Du| = 1$  does not allow any minimum like the one which  $u^-$  has. Moreover we have a tool to define not just a distance function but even signed distance function. From our definition of  $\Gamma$  by  $\Phi_0$  we can define the SDF of  $\Gamma$  as

$$(3.2) \quad d_\Gamma(x) = \begin{cases} \text{viscosity solution of } |Du| = 1 \text{ for } \Phi_0 > 0, \\ 0 \text{ for } \Phi_0 = 0, \\ \text{viscosity solution of } -|Du| = -1 \text{ for } \Phi_0 < 0, \end{cases}$$

or

$d_\Gamma$  is a viscosity solution of the Hamilton-Jacobi equation  $sign\Phi_0 |Du| = sign\Phi_0$  resp.

$$(3.3) \quad sign\Phi_0 (|Du| - 1) = 0,$$

and  $d_\Gamma = 0$  on  $\Gamma$ .

Similar, but evolutionary equation ( see. [20] ) is

$$(3.4) \quad \begin{aligned} \frac{\partial u}{\partial t}(t, x) &= sign u(t, x) \cdot (1 - |\nabla u(t, x)|), \\ x &\in \mathbb{R}^n, \\ t &\geq 0, \\ u|_{t=0} &= u_0. \end{aligned}$$

The stationary solution of this equation is expected to be just the viscosity solution of (3.3). This equation was not studied theoretically for these papers and as far

as we know the mathematical analysis of this equation is still an open problem. In fact we do not know what is a viscosity solution of (3.4). For the rest of this paper we will talk about the viscosity solution of (3.3)

**4. Monotone schemes.** While solving numerically we can use only the schemes which will ensure the convergence to the viscosity solution. In [19] is shown that so called monotone schemes have such property. To define them we will use the following notation.

Consider subset  $\Omega$  of  $\mathbb{R}^n$  to be  $\Omega \equiv \langle a_1, b_1 \rangle \times \langle a_2, b_2 \rangle \cdots \langle a_n, b_n \rangle$  and define numerical grid  $\Delta$  on  $M$ ,  $\Delta \equiv \{x_{i_1 \dots i_n}; i_j = 0, 1 \dots N_j, j = 1 \dots n\}$  where  $x_{i_1 \dots i_n} := (a_1 + i_1 \cdot h_1, \dots, a_n + i_n \cdot h_n)$  and  $h_j := \frac{b_j - a_j}{N_j}; j = 1 \dots n$ . We will also write just  $x_i$  instead of  $x_{i_1 \dots i_n}$ . For given  $T \in \mathbb{R}$  we define  $\Omega_T = \Omega \times (0, T)$ ,  $\Delta_t := \frac{T}{N_T}$ ,  $t_k := k \cdot \Delta t$  for  $k = 0, 1 \dots N_T$  and  $\Delta_T \equiv \{(x_i, t_k); x_i \in \Delta, k = 0, 1, \dots, N_T\}$ . Let  $S^h := L_\infty(\Delta)$  and  $S_T^h := L_\infty(\Delta_T)$  then the values of the functions from  $S^h$  at node  $i$  will be written as  $u_i$  and similarly  $S_T^h$  at node  $i$  and time  $t_k$  will be written as  $u_i^k$ .

The best known monotone schemes are the regularised scheme, the upwind scheme and the Godunov (Max) scheme - [1], [8], [9] and [12]. In the following we will describe them and show several numerical results.

First we define central, forward and backward differences.

$$(4.1) \quad D^c u_i = \begin{cases} \frac{u_1 - u_0}{h}; & i = 0, \\ \frac{u_{i+1} - u_{i-1}}{2h}; & 0 < i < N, \\ \frac{u_N - u_{N-1}}{h}; & i = N, \end{cases}$$

$$(4.2) \quad D^+ u_i = \begin{cases} \frac{u_{i+1} - u_i}{h}; & i < n, \\ \frac{u_n - u_{n-1}}{h}; & i = n, \end{cases}$$

$$(4.3) \quad D^- u_i = \begin{cases} \frac{u_1 - u_0}{h}; & i = 0, \\ \frac{u_i - u_{i-1}}{h}; & i > 0, \end{cases}$$

$$(4.4) \quad D^2 u_i = \begin{cases} \frac{u_0 - 2u_1 + u_2}{2h^2}; & i = 0, \\ \frac{u_{i-1} - 2u_i + u_{i+1}}{2h^2}; & 0 < i < N, \\ \frac{u_{N-2} - 2u_{N-1} + u_N}{2h^2}; & i = N, \end{cases}$$

In very similar way we define finite differences in 2D with respect to  $x$  ( $D_x^c u_{ij}$ ,  $D_x^+ u_{ij}$  and  $D_x^- u_{ij}$ ) and  $y$  ( $D_y^c u_{ij}$ ,  $D_y^+ u_{ij}$  and  $D_y^- u_{ij}$ ).

**Regularised scheme.** In general for the Hamilton-Jacobi equation

$$(4.5) \quad u_t(x) + H(x, u(x), Du(x), D^2 u(x)) = 0,$$

the regularised scheme is defined by equation

$$(4.6) \quad u_t(x) + H(x, u(x), Du(x), D^2 u(x)) = \epsilon \cdot \Delta u(x).$$

The term on the right hand side is so called artificial viscosity term. When  $\epsilon \rightarrow 0$  the solution of (4.6) converges to the viscosity solution of (4.5). This method is known

as a method of vanishing viscosity and can be used even for mathematical analysis of Hamilton-Jacobi equations.

In our particular problem the regularised scheme has the form

$$(4.7) \quad \frac{u_i^{k+1} - u_i^k}{\Delta t} = \text{sign } u_i^0 \cdot (1 - |D^c u_i^k|) + \epsilon D^2 u_i^k$$

for 1D and

$$(4.8) \quad \frac{u_{ij}^{k+1} - u_{ij}^k}{\Delta t} = \text{sign } u_{ij}^0 \cdot \left[ 1 - \left( (D_x^c u_{ij}^k)^2 + (D_y^c u_{ij}^k)^2 \right)^{\frac{1}{2}} \right] + \epsilon (D_x^2 u_{ij}^k + D_y^2 u_{ij}^k)$$

for 2D.

**Upwind scheme.** The upwind schemes were developed for the first order conservation laws. The idea of these schemes is to use forward or backward finite differences in dependence on the information propagation. For the first order Hamilton-Jacobi equation

$$(4.9) \quad u_t + F |Du| = 0$$

the scheme has the form

$$(4.10) \quad \frac{u_{ij}^{k+1} - u_{ij}^k}{\Delta t} = [F_{ij}]_+ \nabla_U^+ (u_{ij}^k) + [F_{ij}]_- \nabla_U^- (u_{ij}^k)$$

where

$$(4.11) \quad \nabla_U^+ = \left( [D_{ij}^{-x}]_+^2 + [D_{ij}^{+x}]_-^2 + [D_{ij}^{-y}]_+^2 + [D_{ij}^{+y}]_-^2 \right)^{\frac{1}{2}}$$

and

$$(4.12) \quad \nabla_U^- = \left( [D_{ij}^{+x}]_+^2 + [D_{ij}^{-x}]_-^2 + [D_{ij}^{+y}]_+^2 + [D_{ij}^{-y}]_-^2 \right)^{\frac{1}{2}}.$$

We use notation  $[a]_+ = \max\{a, 0\}$  and  $[a]_- = \min\{a, 0\}$ . The scheme for the equation (3.4) in 2D takes the form

$$(4.13) \quad \frac{u_{ij}^{k+1} - u_{ij}^k}{\Delta t} = [\text{sign } u_{ij}^0]_+ \nabla_U^+ (u_{ij}^k) + [\text{sign } u_{ij}^0]_- \nabla_U^- (u_{ij}^k) - \text{sign } u_{ij}^0.$$

The scheme is similar in 1D.

**Godunov scheme.** The Godunov scheme [1] is similar to the upwind scheme. For equation (4.9) it has a form

$$\frac{u_{ij}^{k+1} - u_{ij}^k}{\Delta t} = [F_{ij}]_+ \nabla_M^+ (u_{ij}^k) + [F_{ij}]_- \nabla_M^- (u_{ij}^k)$$

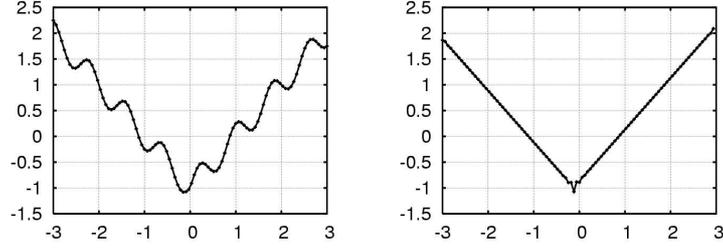


FIG. 4.1. Use of regularised scheme (4.7). Original function is  $f(x) = |x| + 0.25 \cdot \sin(2.5 \cdot \pi x)$  on  $\langle -3, 3 \rangle$ , mesh size  $N = 100$ .

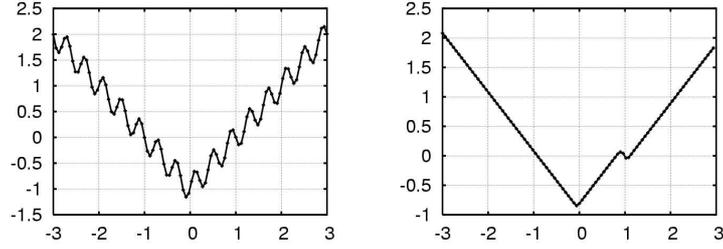


FIG. 4.2. Use of 1D version of upwind scheme (4.13). Original function is  $f(x) = |x| + 0.25 \cdot \sin(5 \cdot \pi x)$  on  $\langle -3, 3 \rangle$ , mesh size  $N = 100$ .

where

$$\nabla_M^+ = \left( \max \left( [D_{ij}^{-x}]_+, -[D_{ij}^{+x}]_- \right)^2 + \max \left( [D_{ij}^{-y}]_+, -[D_{ij}^{+y}]_- \right)^2 \right)^{\frac{1}{2}}$$

and

$$\nabla_M^- = \left( \max \left( [D_{ij}^{+x}]_+, -[D_{ij}^{-x}]_- \right)^2 + \max \left( [D_{ij}^{+y}]_+, -[D_{ij}^{-y}]_- \right)^2 \right)^{\frac{1}{2}}.$$

In 2D, the scheme has a form

$$(4.14) \quad \frac{u_{ij}^{k+1} - u_{ij}^k}{\Delta t} = [\text{sign } u_{ij}^0]_+ \nabla_M^+ (u_{ij}^k) + [\text{sign } u_{ij}^0]_- \nabla_M^- (u_{ij}^k) - \text{sign } u_{ij}^0$$

and it is similar in 1D.

**4.1. Numerical examples.** Some numerical examples computed by the monotone numerical schemes are showed in this section. We did both 1D and 2D computations.

**1D case** - we do not show results obtained by the Godunov scheme because it is very similar to the upwind scheme. As the initial condition we choosed the SDF for interval  $\langle -1, 1 \rangle$  perturbed by sinus. We did the computations on quite coarse grid with just 100 meshes.

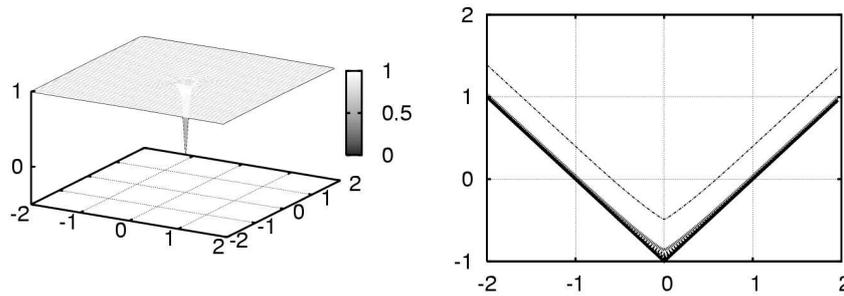


FIG. 4.3. Singularity of unit circle SDF at  $(0,0)$  - graph of  $|\nabla_h d_\Gamma|$  on the left. Evolution of (4.13) with SDF as initial condition on the right (cut along the  $x$  axis) - the graph should not change (zero level set corresponds to shrinking circle).

We can see that both the regularised scheme (4.1) and the upwind scheme (4.2) really converge to the viscosity solution. The disadvantage of the regularised scheme is the need of choice of  $\epsilon$  which can differ from case to case. Upwind scheme offers better convergence. We were able to use even more perturbed initial condition and the convergence was much faster then for the regularised scheme.

**2D case** - in the previous section we dealt with SDF in 1D which is a piecewise linear function and can be easily approximated even on coarse numerical grids. However in 2D SDF is not so "simple". Usually it has one or more singularities which we are not able to approximate with a rectangular numerical grid. It leads to systematical numerical errors which can spoil the scheme convergence. Let us show this effect on particular example.

Consider SDF for the unit circle with equation  $d(x, y) = (x^2 + y^2)^{\frac{1}{2}} - 1$ . It is easy to see that  $\frac{\partial d}{\partial x} = 1$  and  $\frac{\partial d}{\partial y} = 0$  for  $y \equiv 0$  and so  $|\nabla d| = 0$  for  $y \equiv 0$ . On the other hand consider numerical grid on  $\langle -1, 1 \rangle \times \langle -1, 1 \rangle$  with space step  $h$ . For now it is convenient to number the nodes with respect to the origin. It means that the nodes of numerical grids have coordinates  $(-\frac{1}{h}, \dots, 0, \dots, \frac{1}{h}) \times (-\frac{1}{h}, \dots, 0, \dots, \frac{1}{h})$  and node  $(0, 0)$  is at the origin. So we have  $D_x^+ u_{-1,0} = \frac{d(0,0) - d(-h,0)}{h} = 1$  but  $D_y^+ u_{-1,0} = \frac{d(-h,h) - d(-h,0)}{h} = \sqrt{2} - 1 \neq 0$  so either  $|\nabla_h d| = [(D_x^+)^2 + (D_y^+)^2]^{\frac{1}{2}} \neq 1$ . So even if we start with exact SDF all numerical schemes using finite differences on a rectangular grid will see quite big error - see Figure 4.3 on the left. Moreover numerical experiments show that all mentioned schemes have also diffusive effect so they can spread the error to much larger domains. Significant problems can arise when such error get all over the interior of given curve  $\Gamma$ .

On Figure 4.3 on the right there are shown several cuts along the axes  $x$  during the evolution of equation (3.4) using the upwind scheme (4.13) where the initial condition was exact SDF. We can see that the graph starts to rise up at the origin. This process is due to the small error at singularity of the unit circle SDF. It does not matter how small the error is, we can decrease it by decreasing  $h$ , after a while it covers the whole domain where the initial condition was negative. Since we have no boundary conditions there is no feedback to stop this process. At this point whole graph starts to rise up and it will never stop.

It seems to be really essential problem. It means that we are not able to reconstruct SDF with iterative methods based on monotone schemes because as our

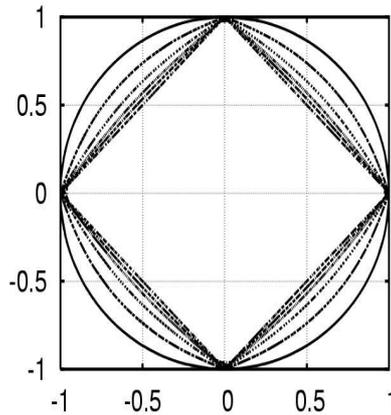


FIG. 4.4. Pointwise error of (4.13) - evolution of the curve obtained by unit circle SDF  $d_\circ$  as an initial condition. At the intersections of the unit circle and the axes,  $d_\circ$  is exactly equal to 0 so it can not change at these points during the evolution.

function converge to SDF some singularities will usually develop. In many applications from interface motion we do not need to have SDF over the whole domain  $\Omega$  but it is sufficient to recover it only at given narrow band along the curve  $\Gamma$ . However we never know if some singularity can develop at the narrow band.

Theoretically it may be right( but we remind - there is no mathematical analysis for given equation). We have SDF for given  $\Gamma$  (with small error) plus a constant which is increasing in time. We need some kind of Dirichlet boundary condition. It is not possible to define it on the boundaries of  $\Omega$  because we do not know values of SDF there. We would like to define this condition on  $\Gamma$  where SDF is equal to zero. Unfortunately there are just few or even none points of the numerical grid intersected by  $\Gamma$ .

On Figure 4.4 we show zero level set obtained by upwind scheme (4.13) with the initial condition equal to the unit circle SDF but now we changed values of the initial function at points  $(1,0)$ ,  $(0,1)$ ,  $(-1,0)$  and  $(0,-1)$  to be equal to zero. Even if the changes are small, original values were approximately of the order  $10^{-8}$ , we obtained qualitatively different solution which have a steady state - SDF to a rectangle given by the four fixed points.

Our idea is to build a narrow band of fixed points along  $\Gamma$ . So we choose  $\delta$  and after each time iteration we look for all nodes where  $|u^k| \leq \delta$  and  $|\nabla_h u^k| \simeq 1$ . Such nodes are marked as fixed and we do not change them anymore. It should also ensure that  $\Gamma$  will not change during the computation.

However on the Figure 4.5 on the right we can see that even with such improvement the original curve is not preserved properly.

**5. Direct algorithms for getting the signed distance function.** Except the method mentioned in the previous section there are also yet another algorithms for getting SDF. They do not solve the evolutionary equation (3.4) but (3.3). In this section we describe two of them - fast marching method introduced by Sethian [16] and fast sweeping method by Tsai, Cheng, Osher and Zhao [21]. We will show that these algorithms are very fast and offer good accuracy. We also come with our new algorithm which is even faster and combines advantages of both algorithms.

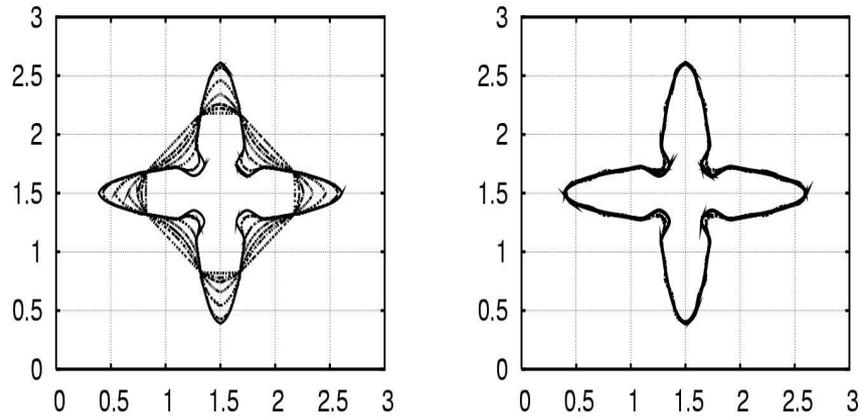


FIG. 4.5. Redistancing of a function given by a dendritic growth. On the left the evolution of the original curve with no fixed points is shown. On the right there is a result obtained by fixing points along the curve - still there is some small error.

**Fast marching method.** This method assumes that we already have SDF in a narrow band along  $\Gamma$ . All of the nodes inside this band are fixed except the nodes which have a neighbour out of this band - they are marked as tentative. At each step of the algorithm we find the node which has the smallest value from all nodes marked as tentative. This point is marked as fixed and all of its unknown and tentative neighbours are updated and marked as tentative. Algorithm stops when all nodes are marked as fixed.

For updating tentative values ( at a grid node  $x_{ij}$  ) we use upwind

$$(5.1) \quad [D_x^- u_{ij}]_+^2 + [D_x^+ u_{ij}]_-^2 + [D_y^- u_{ij}]_+^2 + [D_y^+ u_{ij}]_-^2 = 1$$

or Godunov scheme

$$(5.2) \quad \max \left( [D_x^- u_{ij}]_+, -[D_x^+ u_{ij}]_- \right)^2 + \max \left( [D_y^- u_{ij}]_+, -[D_y^+ u_{ij}]_- \right)^2 = 1.$$

The key idea of the fast marching method is observation that in the case of upwind scheme information propagates from smaller values to larger values. So we start at the points closest to  $\Gamma$  and use them to compute SDF at further points. To solve (5.1) or (5.2) we always choose the neighbours with the smallest magnitude to substitute to finite differences. It leads to a quadratic equation which can have one or two solutions. From these solutions we choose the one with largest magnitude to obtain the right (viscosity) solution. Fast marching method gives much better results than iterative methods we discussed in the previous section. First of all one should notice that it uses a band of fixed points along  $\Gamma$  which is the same assumption as we needed above. In this case we can consider it as a initial condition for a problem of monotonically advancing front.

**Fast sweeping method.** This method is a type of the Gauss-Seidel iterative method. Again we assume to have a band of fixed points along  $\Gamma$  with a good approximation

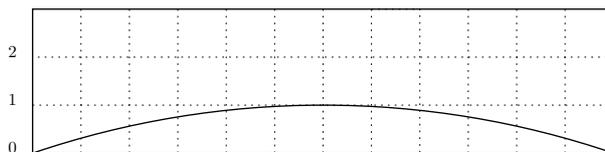


FIG. 5.1. When reconstructing SDF for the curve on the figure we must start in the middle with the points closest to the curve and then proceed towards the edges.

of SDF and the rest is set to  $\pm\infty$ . We sequentially change the direction of iterations: top→bottom left→right, bottom→top left→right, top→bottom right→left and bottom→top right→left. If there is one or more finite neighbours of currently visited point we use (5.2) to update value at this point. It is mainly because of good properties of (5.2) that after few iterations we have good approximation of SDF. For more details see [21].

Fast sweeping method gives almost the same accuracy as fast marching method. A disadvantage of fast marching method is the searching for a point with smallest magnitude. It slows down this method significantly. There are some attempts to speed this step up - using binary tree [18] or hash tables [16]. Fast sweeping method does not use any stack which we should go through and that is why it is much faster. It is also much simpler to implement. On the other hand fast marching method can be stopped whenever you want and you have at least approximation of the SDF at some band along  $\Gamma$ . As we said before this is sufficient for many applications, for example in interface motion. This cannot be done with fast sweeping method. If you stop it too soon you are going to have a function which is far from good approximation of SDF even at points close to initial band.

In the next section we describe an algorithm which combines advantages of both methods - it constructs the solution from the given initial band and it does not need any searching for closest point.

**Front tracing method.** The idea of this algorithm (from now we will also refer it as FTM) is to start at the curve and then propagate to further regions. Since it is the same way in which the information propagates we can expect improvement in the efficiency of the algorithm.

In the following we consider more general equation  $|\nabla u(x)| = F(x)$ . Here  $u$  has a meaning of first arrival time of the monotonically advancing front and  $\frac{1}{F(x)} > 0$  is speed of the advancing front. We will use the scheme (5.2)

$$(5.3) \quad \max \left( [D_x^- u_{ij}]_+, -[D_x^+ u_{ij}]_- \right)^2 + \max \left( [D_y^- u_{ij}]_+, -[D_y^+ u_{ij}]_- \right)^2 = F_{ij}^2.$$

See Figure 5.1. We assume to have good approximation of the solution at the closest neighbours of the initial curve. In this case they are the rows 0 and 1. Now we want to approximate the solution to the neighbourhood which is the row number 2. The best approximation can be obtained by updating first the neighbours of the smallest points as it is done in FMM. In this case it is the point in the middle. So if we sweep over the updating row from the left to the right only the right half is correct and the left half has larger values than the correct solution. Since we know that the scheme (5.3) always use smaller values to update to new one we can sweep from right to the left and get the correct solution over the whole row. This is similar to FSM in 1D.

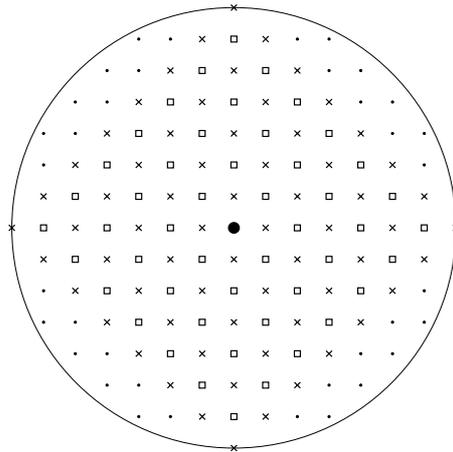


FIG. 5.2. *Difference between real physical and virtual front. Crosses and boxes show stack of the new algorithm after each marching step without synchronising steps. Dots show the points that should already in the stack but they are not.*

Now we can construct the first version of our algorithm. In the first step we construct the initial narrow band. It contains just the points which have a neighbour with different sign. We assume to have them ordered in the clockwise direction along the curve  $\Gamma$ . Now we can sweep in the same direction over all neighbours and then in the opposite direction and we have the right approximation. In the same way we can proceed further and at the end we have the solution over the whole grid.

So far we did not take care about the speed of propagation. It means that our virtual front does not have to correspond with the real physical one. This can be a problem at points where more characteristics intersect each other. The parts following different characteristics can have different values and a discontinuity can develop there. To solve it we must synchronize the algorithm with the real front. We stop the virtual front at the points where the real one is slow and proceed just with the fast parts of the front. It is done by taking the maximum of already known values and then propagate further as we did above until all values along the virtual front are approximately equal to the maximum. In fact even if we solve a problem with  $F(x) = 1$  the virtual front does not correspond with the real one - see Figure 5.2.

In the following we describe the algorithm in details.

For simplicity we assume having a 2D mesh  $M$  of size  $N \times N$  with space step  $h$  and function  $u_{ij}^0$  defined on  $M$ . We also need a stack  $S_i$  for  $i = 0 \dots N^2 - 1$  of coordinates of points from  $M$ . Set of neighbours  $B(i, j)$  of point  $(i, j)$  is defined as  $B \equiv \{(i - 1, j), (i + 1, j), (i, j - 1), (i, j + 1)\}$ . We write  $E(i_1, j_1, i_2, j_2)$  for an edge given by points  $(i_1, j_1)$  and  $(i_2, j_2)$ . We say  $E(i_1, j_1, i_2, j_2)$  is intersected by  $\Gamma$  if  $u_{i_1 j_1}^0 \cdot u_{i_2 j_2}^0 \leq 0$ . By updating a value at given point we mean use of (5.2) where we do not use points  $(i, j)$  such that  $u_{ij} = \pm\infty$ .

**Initiation**

I0: set  $m_0 = 0, m_1 = 0$

I1: for all edges  $E(i_1, j_1, i_2, j_2)$  of the mesh  $M$

if  $E(i_1, j_1, i_2, j_2)$  is intersected by  $\Gamma$

set  $S_{m_1} = (i_1, j_1), S_{m_1+1} = (i_2, j_2)$  and  $m_1 = m_1 + 2$

- set  $u_{i_1j_1} = u_{i_1j_1}^0$  and  $u_{i_2j_2} = u_{i_2j_2}^0$
- I2: for all  $(i, j) \notin S$  set  $u_{ij} = \text{sign}u_{ij}^0 \cdot \infty$
- I3: set  $n = 1$
- I4: set  $s_0 \equiv \{(i, j) \mid (i, j) \equiv S_m; m \in [m_0, m_1]\}$

We assume that for all  $(i, j) \in s_0$   $u_{ij}$  is good approximation of SDF.

### Marching step

- M0: set  $m_{n+1} = m_n$
- M1: for all  $m \in s_{n-1}$  -  $m$  increasing ( from  $m_{n-1}$  to  $m_n$  ) - **upward step**  
     for all points  $(i, j) \in B(m) \setminus S$   
     update  $u_{ij}$
- M2: for all  $m \in s_{n-1}$  -  $m$  decreasesing ( from  $m_n$  to  $m_{n-1}$  ) - **downward step**  
     for all points  $(i, j) \in B(m) \setminus S$   
     update  $u_{ij}$   
     set  $S_{m_{n+1}} = (i, j)$   
     set  $m_{n+1} = m_{n+1} + 1$
- M3: set sequence  $s_n \equiv \{(i, j) \mid (i, j) \equiv S_m; m \in [m_n, m_{n+1}]\}$
- M4: set  $u_{max} = \max_{(i,j) \in s_n} u_{ij}$
- M5: set  $n_0 = n$
- M6: set  $n = n + 1$

### Synchronising step

- S0: for all  $m \in s_{n-1}$  -  $m$  increasing - **upward step**  
     for all points  $(i, j) \in B(m) \setminus S$   
     update  $u_{ij}$
- S1: for all  $m \in s_{n-1}$  -  $m$  decreasesing - **downward step**  
     for all points  $(i, j) \in B(m) \setminus \bigcup_{k=0}^{n_0} s_k$   
     update  $u_{ij}$   
     if  $u_{ij} \leq u_{max}$   
     set  $S_{m_{n+1}} = (i, j)$   
     set  $m_{n+1} = m_{n+1} + 1$
- S2: set sequence  $s_n \equiv \{(i, j) \mid (i, j) \equiv S_m; m \in [m_n, m_{n+1}]\}$
- S3: set  $n = n + 1$
- S4: if  $s_{n-1}$  is not empty go to step S0

### Repeat

- R0: if there was at least one nod updated go to M0

For simplicity in the initial step we just said that we take vertices of all edges intersected by  $\Gamma$ . For the sweeping along  $\Gamma$  we need to assume some kind of continuity of the first sequence. Such continuous sequence can be obtained as follows: Find the first square of the mesh intersected by  $\Gamma$ , put all vertices of intersected edges into the stack and then recursively process all squares adjacent to intersected edges. Of course we need to ensure that we do not go several times through any of the mesh squares. In fact, what we need is a 2D version of so called marching cubes algorithm by Bloomenthal [6].

Note that all points added to the stack  $S$  during the synchronising step are not fixed before the step ends (see condition: update all points  $(i, j) \in B(m) \setminus \bigcup_{k=0}^{n_0} s_k$ ).

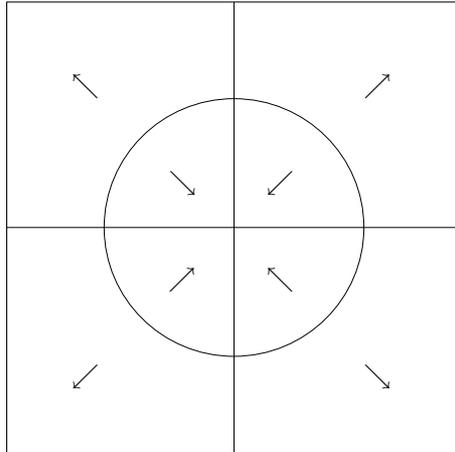


FIG. 5.3. This figure shows what direction of the sweeping is needed in the particular domain when solving (3.3).

That is because within the synchronising step two virtual fronts can intersect each other at wrong place. In this case we just let them go over each other and as we saw above the scheme (5.3) has the property to fix it.

**5.0.1. SDF reconstruction along  $\Gamma$ .** The last thing we have to solve is reconstruction of the SDF along  $\Gamma$ . All of the algorithms mentioned in this section are able just to extend SDF to further regions.

Our first attempt was to extend FMM. When studying this method we found out little difference in the explanation by Sethian [16] and by Osher [18]. In his book Osher says "tentative values are never used to compute new tentative values" but Sethian does use tentative values. We tried not to fix any point at the beginning so we had initial band just with tentative values and each of them was allowed to be changed. Then we tried if FMM with this initial band could recover SDF even inside the band. Unfortunately this is not possible. Consider simple example in 1D. We have a function  $f(x) = 0.1 \cdot x$  for  $x \geq 0$  and we want to recover it to SDF which is  $d(x) = x$ . If the space step  $h$  is 1 then we have  $f(x_0) = 0$ ,  $f(x_1) = 0.1$ ,  $f(x_2) = 0.2$  etc. At the first step FMM choose point  $x_0$  fix it and recompute  $x_1$  which is  $f(x_1) = f(x_0) + h = 1$ . In the next step FMM is supposed to take  $x_1$  which is the closest one to  $x_0$  but instead of it FMM uses  $x_2$  to recompute  $f(x_3) = f(x_2) + h = 1.2$ . In general we can say that FMM is not able to recover SDF at all points  $x$  where  $|\nabla f_0(x)| \leq 1$ . On the other hand FMM can work well if  $|\nabla f_0(x)| \geq 1$ .

Another attempt was use of the scheme (4.13) just for the very narrow band obtained by the initial step of FTM. It does not work even if there are no singularities in the band.

In the end we had to use simple method - approximation of the curve and then computing SDF for all neighbouring points. There are two ways how to do it. One can either compute just intersections of  $\Gamma$  and mesh edges and with this data approximate SDF at vertices of intersected edges (we always use the smallest value for vertices which belong to more intersected edges) or one can approximate  $\Gamma$  at each mesh square by a linear function and then compute distance between the square vertices and approximation of  $\Gamma$ . Since we do it only for the closest neighbours of  $\Gamma$  it is not

computationally expensive.

**5.0.2. Numerical experiments.** In this section we compare the algorithms mentioned above in 2D. We tested the accuracy using SDF to unit circle which can be solved in analytical form -  $d_{\circlearrowleft} = \sqrt{x^2 + y^2} - 1$ . We compare all algorithms from this section and the iterative scheme (4.13). On the Figures and Tables (5.5), (5.1) and (5.4) one can see that the iterative methods based on the evolutionary equation (3.4) are not optimal. As we said before they do not preserve the given curve  $\Gamma$  properly. The direct algorithms ( FMM, FSM and FTM ) seems to be better. Even if the initial condition is close to SDF, iterative methods need tens or hundreds of iterations. On the other hand, FSM method converges after less then 10 iterations for arbitrary initial condition. Since FMM, FSM and FTM give almost the same accuracy the only criterion is their efficiency. From our results one can see that FMM is the slowest one. Here we must notice that we did not use any method to speed up searching in the stack of FMM (for comparison of improved FMM and FSM see. [22] ). FSM seems to be good choice, it is fast and it is the simplest method to implement. FTM shows good efficiency in the case when we need SDF only at some band along  $\Gamma$  which is usual in many applications based on the level set methods (see Figure 5.1).

On the Figure 5.1 you can see application of the new algorithm to real physical problem which is model of dendritic growth [3].

**5.1. Conclusion.** This article shows that the direct methods based on the equation (3.3) are much better for SDF reconstruction then the iterative methods based on the equation (3.4) which are very slow, less accurate and do not preserve the original curve properly. FSM is good choice for its high efficiency and simple implementation. If one needs even faster algorithm FTM should be considered mainly for the narrow band methods or for more general problems like monotonically advancing front with curved characteristics.

**Acknowledgement.** Author prepared this article during his stay at the University of Sussex as a part of the Marie Curie Fellowship founded by the European Union. The author was also partly supported by project MSM 6840770010 of the Ministry of Education of the Czech Republic and by the project No. CTU 0415314 of Czech Technical University in Prague. Author would like to acknowledge Prof. Charles M. Elliott<sup>1</sup> for fruitful discussions.

#### REFERENCES

- [1] M. BARDI AND S. OSHER. *The nonconvex multi-dimensional Riemann problem for Hamilton-Jacobi equations*, SIAM Journal of Mathematical Analysis **22** (1991), 344–351.
- [2] G. BARLES. *Solutions de viscosité des équations de Hamilton-Jacobi*, Springer-Verlag, Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, NY 10010, USA, 1994.
- [3] M. BENEŠ. *Mathematical and computational aspects of solidification of pure substances*, Acta Mathematica Universitatis Comenianae **70** (2000), 123–151.
- [4] ———, *Mathematical analysis of phase-field equations with numerically efficient coupling terms*, Interfaces and Free Boundaries **3** (2001), 201–221.
- [5] M. BENEŠ AND K. MIKULA. *Simulation of anisotropic motion by mean curvature - comparison of phase field and sharp interface approaches*, Acta Mathematica Universitatis Comenianae **67** (1998), 17–42.
- [6] J. BLOOMENTHAL. *An implicit surface polygonizer*, Graphics Gems IV, Academic Press, 1994, pp. 324–349.
- [7] A. BRIANI. *Notes on viscosity solution for partial differential equation*, (2002).

---

<sup>1</sup>Centre for Mathematical Analysis and its Applications, School of Mathematical Sciences, University of Sussex, Falmer, Brighton BN1 9QH, UK

N	h	Iter.			FMM		
		$L_\infty$	$L_1$	CPU	$L_\infty$	$L_1$	CPU
100	0.04	0.0279	0.1772	3	0.0310	0.0871	0.4
200	0.02	0.0246	0.1293	21	0.0166	0.0396	2.0
500	0.008	0.0185	0.1053	411	0.0092	0.0154	34.0
1000	0.004	0.0168	0.0975	41142	0.0051	0.0081	312.0
2000	0.002				0.0030	0.0041	4147.0

N	h	FSM			FTM		
		$L_\infty$	$L_1$	CPU	$L_\infty$	$L_1$	CPU
100	0.04	0.0275	0.1070	0.03	0.0273	0.1078	0.04
200	0.02	0.0174	0.0577	0.16	0.0172	0.0583	0.12
500	0.008	0.0088	0.0232	1.00	0.0087	0.0235	0.70
1000	0.004	0.0050	0.0112	4.00	0.0050	0.0114	3.00
2000	0.002	0.0029	0.0054	26.00	0.0029	0.0055	15.00

FIG. 5.4. Comparison of  $\|d_h - d_\Gamma\|_{L_1}$  and  $\|d_h - d_\Gamma\|_{L_\infty}$  and CPU time in seconds for the upwind scheme (4.13), FMM, FSM and FTM -  $d_h$  is numerical approximation and  $d_\Gamma$  is the exact SDF for a unit circle on  $\Omega = \langle -2, 2 \rangle \times \langle -2, 2 \rangle$ . In the case of FSM we stopped the computation after 8 sweepings. However it gives just a little bit more accurate results then 4 sweepings in which case both FSM and FTM need the same CPU time.

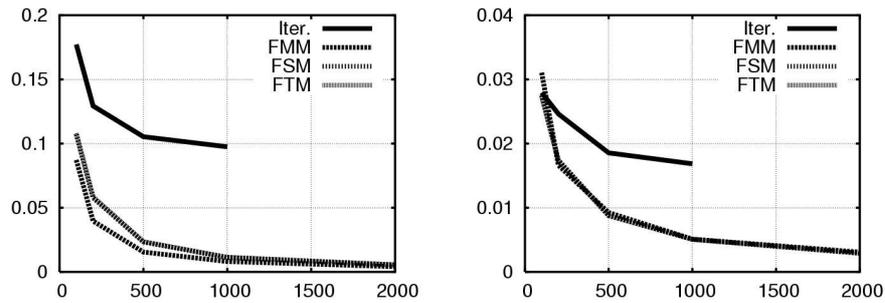


FIG. 5.5. Comparison of  $\|d_h - d_\Gamma\|_{L_1}$  (on the left) and  $\|d_h - d_\Gamma\|_{L_\infty}$  (on the right) for the upwind scheme (4.13), FMM, FSM and FTM -  $d_h$  is numerical approximation and  $d_\Gamma$  is the exact SDF for an unit circle on  $\Omega = \langle -2, 2 \rangle \times \langle -2, 2 \rangle$ . Horizontal axes shows mesh size, vertical axes shows the error.

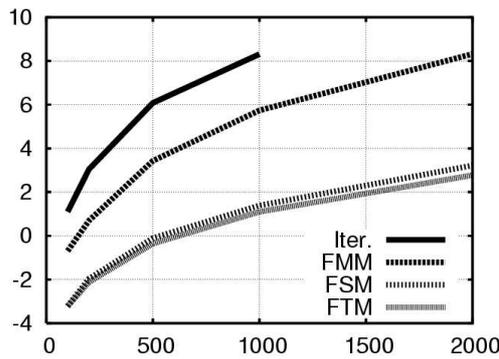


FIG. 5.6. Comparison of CPU time in logarithmic scale for FMM, FSM and FTM.

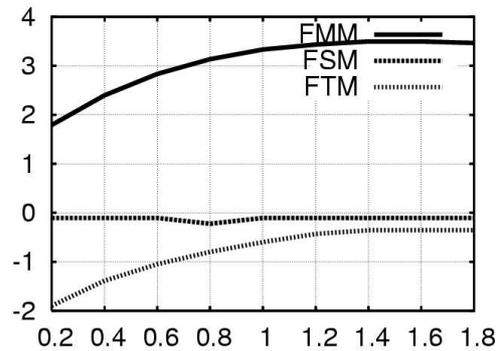


FIG. 5.7. Comparison of CPU time in logarithmic scale for FMM, FSM and FTM when computing SDF only on given band along  $\Gamma$ . The grid size is  $500 \times 500$  nodes, horizontal axis shows band width and vertical axis shows CPU time. FSM is able to compute SDF just over the whole domain so its time does not depend on band width.

- [8] A. J. BRIGGS. *Numerical solutions of Hamilton-Jacobi equation*, PHD dissertation, University of Sussex, Department of Mathematics, Mantell Building, University of Sussex, Falmer, Brighton, BN1 9RF, UK, 1999.
- [9] J. R. CLAISSE. *Vortex density motion in a cylindrical type 2 superconductor subject to a transverse applied magnetic field*, PHD dissertation, University of Sussex, Department of Mathematics, Mantell Building, University of Sussex, Falmer, Brighton, BN1 9RF, UK, 2000.
- [10] M. CRANDALL AND P.-L. LIONS. *Viscosity solution of Hamilton-Jacobi equations*, Trans. American Mathematical Society **277:1** (1983), 1–41.
- [11] M. G. CRANDALL. *Viscosity solutions: a primer*, Viscosity solutions and applications (Montecatini Terme, 1995), Lecture Notes in Math., vol. 1660, Springer, 1997, pp. 1–43.
- [12] K. DECKELNICK AND C. M. ELLIOTT. *Uniqueness and error analysis for Hamilton-Jacobi equations with discontinuities*, (2003).
- [13] C. M. ELLIOTT. *Approximations of curvature dependent motion*, State of the Art in Numerical Analysis (G.A. Watson I.S. Duff, ed.), Clarendon Press, Oxford, 1997, pp. 407–440.
- [14] C. M. ELLIOTT AND V. STYLES. *Computations of bidirectional grain boundary dynamics in thin metallic films*, Journal of Computational Physics **187** (2003), 524–543.
- [15] L. C. EVANS. *Partial differential equations*, Graduate Studies in Mathematics, American Mathematical Society, P. O. Box 6248, Providence, Rhode Island 02940-6248, USA, 1998.
- [16] J.A. SETHIAN. *A fast marching level set method for monotonical advancing fronts*, Proceedings of the National Academy of Sciences, vol. 93, National Academy of Sciences, 1996, pp. 1591–1595.
- [17] M. KIMURA AND H. NOTSU. *A level set method using the signed distance function*, Japan Journal of Industrial and Applied Mathematics **19** (2002), no. 3, 415–446.
- [18] S. OSHER AND R. FEDKIW. *Level set methods and dynamic implicit surfaces*, Springer-Verlag, Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, NY 10010, USA, 2003.
- [19] D. OSTROV. *Viscosity solutions and convergence of monotone schemes for synthetic aperture radar shape-from-shading equations with discontinuous intensities*, SIAM Journal of Applied Mathematics **59:6** (1999), 2060–2085.
- [20] J. A. SETHIAN. *Level set methods, evolving interfaces in geometry, fluid mechanics, computer vision, and materials science*, Cambridge University Press, New York, 1996.
- [21] S. OSHER, Y. R. TSAI, L. CHENG AND H. ZHAO. *Fast sweeping algorithms for a class of Hamilton-Jacobi equations*, SIAM Journal on Numerical Analysis **41**, No. 2 (2003), 673–694.
- [22] H. ZHAO. *The fast sweeping method and applications*, 2004, Interphase 2004, <http://www.mat.uniroma1.it/interphase04/talks/Zhao.pdf>.

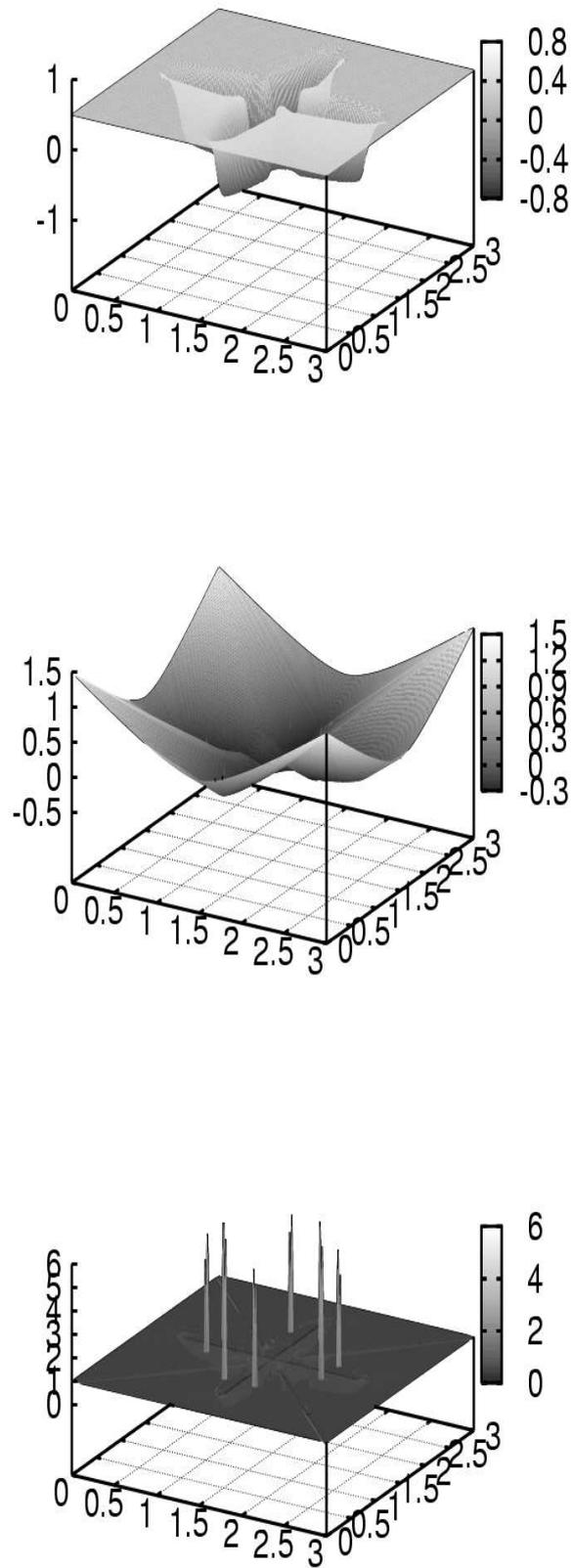


FIG. 5.8. Original function taken from phase field model of dendritic growth, its SDF  $d_\Gamma$  and  $|\nabla d_\Gamma|$  obtained by FTM.