Distance function and extension in normal direction for implicitly defined interfaces

Peter Frolkovič[1,2], Karol Mikula[1,2], Jozef Urbán[2]

Department of Mathematics and Descriptive Geometry, Slovak University of Technology, Bratislava, Slovakia

Abstract

In this paper we present a novel application of extrapolation procedure for three popular numerical algorithms to compute the distance function for an interface that is given only implicitly. The methods include the fast marching method [9], the fast sweeping method [10] and the linearization method [3]. The extrapolation procedure removes the necessity of a special initialization procedure for the grid nodes next to the interface that is used so far with the methods, thus it represents a natural extension of these methods. The extrapolation procedure can be used also for an extension of a function that is defined only locally on the interface in the direction given by the gradient of distance function [2].

Keywords: distance function, extrapolation, fast marching method

1. Introduction

When using level set methods to describe an evolving curve or surface [8, 6], the interface is given only indirectly as a zero level set of some function. Consequently, to obtain the distance function to such interface one needs to solve eikonal equation with Dirichlet boundary conditions defined on implicitly given boundary.

To avoid any explicit reconstruction of the boundary we propose here to use extrapolation procedure near the boundary for three popular numerical

Email address: peter.frolkovic@stuba.sk (Peter Frolkovič)

[1]The author was supported by VEGA 1/1137/12.

[2]The author was supported by APVV 0184-10.

algorithms: the fast marching method [9], the fast sweeping method [10], and the linearization of eikonal equation [3]. In such a way, one can skip any initialization of numerical values in the grid nodes next to the interface as it is realized for these methods so far. For the fast marching method with the extrapolation procedure near the interface we can report an improvement when compared with the fast marching method in [1] for the computations of the distance function of implicitly defined interface.

Moreover, these numerical methods using the extrapolation near interface can be applied also for so called extension in normal direction to the interface [11, 1, 2]. Such procedure is required when some quantity known only on the interface must be extended away from the interface to a surrounded computational domain. The extension can be obtained by solving a stationary linear advection equation with Dirichlet boundary conditions defined on the implicitly given interface.

We note that our aim is not to compare the available numerical algorithms for the computations of distance function or for the solution of stationary linear advection equation with respect to efficiency and accuracy. We recognize that each of them is preferable in some special situations, see e.g. [5].

The paper is organized as follows. In section 2 we present the related mathematical models. In section 3 the numerical scheme of Rouy-Tourin [7] is extended by extrapolation procedure for the grid nodes next to the interface. In section 4 some numerical experiments are presented.

2. Mathematical models

Let Γ be a closed interface contained in $D \subset R^2$. The domain enclosed by Γ will be denoted by Ω, i.e. $\Gamma \equiv \partial\Omega$. We suppose that the interface is given only implicitly as the zero level set of some function $\phi : D \to R$ such that $\phi(x) < 0$ for $x \in \Omega$ and $\phi(x) > 0$ for $x \in D \setminus \overline{\Omega}$, so

$$\phi(x) = 0\,,\ x \in D \quad \Leftrightarrow \quad x \in \Gamma\,. \tag{1}$$

The implicit definition (1) of Γ using the zero level set of ϕ is typical when some evolving interface is described using level set methods. We avoid in our approach any explicit reconstruction of Γ.

The distance function d of Γ is defined by

$$d(x) = \min_{\gamma \in \Gamma} |x - \gamma|\,, \tag{2}$$

where $|\cdot|$ denotes the Euclidean norm.

The gradient of distance function plays an important role in level set methods. One can show [6] that if for $x \in D$ there exists a unique closest point $\gamma \in \Gamma$ such that $d(x) = |x-\gamma|$ then $\nabla d(x)$ is well defined and $|\nabla d(x)| = 1$. The existence of $\nabla d(x)$ or the property $|\nabla d(x)| = 1$ need not to be true when the closest point on Γ for x is not unique.

Following [8, 6], one can find d as a viscosity solution of eikonal equation

$$|\nabla d(x)| = 1\,,\ x \in D\,, \quad d(\gamma) = 0\,,\ \gamma \in \Gamma\,. \tag{3}$$

The problem (3) can be solved independently for two subdomains of D, once for $x \in \Omega$ and once for $x \in D\backslash\overline{\Omega}$. For the first subdomain one has $\Gamma \equiv \partial\Omega$, for the second one the boundary is given by $\Gamma \cup \partial D$, but no boundary conditions are required on ∂D.

The equation (3) can be written formally as a stationary advection equation with right hand side

$$\vec{v}(x) \cdot \nabla d = 1\,, \quad \vec{v} = \frac{\nabla d}{|\nabla d|}\,. \tag{4}$$

The formulation (4) can be used for a linearization of eikonal equation [3].

Once the distance function d is available and ∇d can be computed, one can utilize the linear advection equation for an unknown function s,

$$\nabla d \cdot \nabla s(x) = 0\,,\ x \in D\,, \quad s(\gamma) = S(\gamma)\,,\ \gamma \in \Gamma\,. \tag{5}$$

The interpretation of (5) can be seen [2] as an extension (extrapolation) of known values S (that are given only on the interface Γ) to the values s defined in the whole domain D. The extension using (5) is constant in normal direction to the interface, and the value $s(x)$ is equal to $S(\gamma)$ if $\gamma \in \Gamma$ is the unique closest point to x. Again, the problem (5) has to be solved for two subproblems.

3. Numerical methods

We present a numerical scheme of Rouy-Tourin proposed in [7] that is popular to use when solving the eikonal equation (3) on rectangular grids. The method seems to give the smallest error among several first order accurate schemes [9, 10].

We use a standard notation for rectangular grids. For a simplicity of notation let D be a square $(0, L)^2$. Let $N > 0$ be a chosen number and $h = L/N$. We aim to find the values d_{ij} that approximate the exact distances $d(x_i, y_j)$ with $x_i = ih$ and $y_j = jh$, $i, j = 0, 1, \ldots, N$ by numerical solution of (3). Analogously, the values $s_{ij} \approx s(x_i, y_j)$ will be found by solving (5) numerically.

The main idea is to choose an appropriate approximation of gradient $\nabla d_{ij} \approx \nabla d(x_i, y_j)$. It can be obtained by finite difference scheme

$$\nabla d_{ij} = (\partial_x d_{ij}, \partial_y d_{ij}) \approx \frac{1}{h}(\delta_x d_{ij}, \delta_y d_{ij})\,, \tag{6}$$

where the "upwind" differencing shall be used:

$$\delta_x d_{ij} := \begin{cases} d_{ij} - d_{i-1j} & d_{i-1j} \leq \min\{d_{ij}, d_{i+1j}\} \\ d_{i+1j} - d_{ij} & d_{i+1j} \leq \min\{d_{ij}, d_{i-1j}\} \\ 0 & d_{ij} \leq \min\{d_{i-1j}, d_{i+1j}\} \end{cases} \tag{7}$$

and analogously

$$\delta_y d_{ij} = \begin{cases} d_{ij} - d_{ij-1} & d_{ij-1} \leq \min\{d_{ij}, d_{ij+1}\} \\ d_{ij+1} - d_{ij} & d_{ij+1} \leq \min\{d_{ij}, d_{ij-1}\} \\ 0 & d_{ij} \leq \min\{d_{ij-1}, d_{ij+1}\} \end{cases} \tag{8}$$

When using (7) and (8) for the nodes lying on ∂D, one has to simply skip the unavailable difference in (7) or (8).

Before using the approximation (6) with (7) and (8) for the fast marching [9] or the fast sweeping method [10], the eikonal equation (3) is written in the form $|\nabla d|^2 = 1$. Consequently, the numerical scheme to find the approximative distance function can be written in the form

$$(\delta_x d_{ij})^2 + (\delta_y d_{ij})^2 = h^2\,. \tag{9}$$

When solving the eikonal equation using (4) the scheme takes the form

$$\vec{v}_{ij} \cdot \nabla d_{ij} = 1\,, \quad \vec{v}_{ij} = \frac{\nabla d_{ij}}{|\nabla d_{ij}|}\,. \tag{10}$$

Note that when linearizing (10), one must insure that no division by zero occurs in (10), see later the related discussion in section 3.2.

Finally, to solve (5) numerically we use standard first order accurate upwind differencing to obtain

$$\begin{aligned} &\max\{0, \delta_x d_{ij}\}(s_{ij} - s_{i-1j}) + \min\{0, \delta_x d_{ij}\}(s_{i+1j} - s_{ij}) + \\ &\max\{0, \delta_y d_{ij}\}(s_{ij} - s_{ij-1}) + \min\{0, \delta_y d_{ij}\}(s_{ij+1} - s_{ij}) = 0\,. \end{aligned} \tag{11}$$

3.1. Extrapolation procedure

For implicitly defined interfaces (as described in section 2) one has to use (9) - (11) for two subdomains. To identify to which subdomain the grid points (x_i, y_j) belong, we denote $\phi_{ij} := \phi(x_i, y_j)$. Clearly, $(x_i, y_j) \in \Omega$ when $\phi_{ij} < 0$, and $(x_i, y_j) \in D \setminus \overline{\Omega}$ when $\phi_{ij} > 0$. If $\phi_{ij} = 0$ one can use directly the Dirichlet boundary conditions on Γ for d_{ij} and s_{ij}, therefore we exclude these values later from unknowns in algebraic equations.

We say that the grid node (x_i, y_j) is *next to the interface in x direction* if at least one of the following two inequalities is valid:

$$\phi_{ij}\phi_{i-1j} < 0\,,\ \phi_{ij}\phi_{i+1j} < 0\,. \tag{12}$$

Analogously, we say the node (x_i, y_j) is *next to the interface in y direction* if at least one of following inequalities is true:

$$\phi_{ij}\phi_{ij-1} < 0\,,\ \phi_{ij}\phi_{ij+1} < 0\,. \tag{13}$$

We comment the computations of distance function for the grid nodes next to the interface. In previous efforts [1, 10], the values d_{ij} for the nodes next to the interface are computed by some kind of "brute force" method (or its approximation) using directly the definition (2). This might not always be straightforward to do and needs not to be compatible with the computations of d_{ij} obtained by the numerical scheme (9).

Here we propose to use an extrapolation procedure near the interface, so the values d_{ij} for the grid nodes next to the interface are computed by the numerical scheme (9).

To do so we characterize the position of a point on the interface between two neighboring grid points for which ϕ changes its sign. Let (x_i, y_j) be a grid node next to the interface in x direction. The required point, say $x_{i-\alpha\, j}$ or $x_{i+\alpha\, j}$, will be identified with some $\alpha_{i-1j} \in (0, 1)$ or $\alpha_{i+1j} \in (0, 1)$ such that

$$x_{i\pm\alpha\, j} = \alpha_{i\pm1j} x_{i\pm1} + (1 - \alpha_{i\pm1j}) x_i.$$

Analogously the values $\alpha_{ij\pm1} \in (0, 1)$ and the point $y_{i\, j\pm\alpha}$ shall be treated.

The values $\alpha_{i\pm1j}$ will be determined from the linear interpolation of ϕ_{ij} and $\phi_{i\pm1j}$ by requiring that $0 = \alpha_{i\pm1j}\phi_{i\pm1j} + (1 - \alpha_{i\pm1j})\phi_{ij}$ and analogously for $\alpha_{ij\pm1}$. Clearly,

$$\alpha_{i\pm1j} = \frac{\phi_{ij}}{\phi_{ij} - \phi_{i\pm1j}}\,, \quad \alpha_{ij\pm1} = \frac{\phi_{ij}}{\phi_{ij} - \phi_{ij\pm1}}\,. \tag{14}$$

Note that the values $\alpha_{i\pm1j}$ and $\alpha_{ij\pm1}$ are specific for each grid node (x_i, y_j) that we do not further emphasize in their notation.

The idea of extrapolation near the interface is to use the same procedure to compute the values that are not available in (7) and (8) for the grid nodes next to the interface. Particularly, the value $u_{i\pm1\,j}$ for the node next to the interface in x direction can be extrapolated from the values u_{ij} and $u_{i\pm\alpha\,j} := u(x_{i\pm\alpha\,j}, y_j)$ with the latter one given by Dirichlet boundary conditions on Γ. In our case, $d_{i\pm\alpha\,j} = d_{ij\pm\alpha} = 0$, $s_{i\pm\alpha\,j} = S(x_{i\pm\alpha\,j}, y_j)$ and $s_{i\,j\pm\alpha} = S(x_i, y_{i\,j\pm\alpha})$.

From $u_{i\pm\alpha\,j} = \alpha_{i\pm1j}u_{i\pm1j} + (1 - \alpha_{i\pm1j})u_{ij}$ and so on one obtains

$$u_{i\pm1j} = \frac{(\alpha_{i\pm1j} - 1)u_{ij} + u_{i\pm\alpha\,j}}{\alpha_{i\pm1j}}\,, \quad u_{ij\pm1} = \frac{(\alpha_{ij\pm1} - 1)u_{ij} + u_{i\,j\pm\alpha}}{\alpha_{ij\pm1}}\,. \tag{15}$$

Substituting (15) to (7) and (8) we obtain for the nodes next to the interface in x direction

$$\delta_x u_{ij} = \begin{cases} \frac{1}{\alpha_{i-1j}}\,(u_{ij} - u_{i-\alpha\,j}) & \phi_{ij}\phi_{i+1j} \geq 0 \text{ or } \alpha_{i-1j} \leq \alpha_{i+1j} \\ \frac{1}{\alpha_{i+1j}}\,(u_{i+\alpha\,j} - u_{ij}) & \phi_{ij}\phi_{i-1j} \geq 0 \text{ or } \alpha_{i+1j} < \alpha_{i-1j} \end{cases} \tag{16}$$

and analogously for the nodes next to the interface in y direction

$$\delta_y u_{ij} = \begin{cases} \frac{1}{\alpha_{ij-1}}\,(u_{ij} - u_{i\,j-\alpha}) & \phi_{ij}\phi_{ij+1} \geq 0 \text{ or } \alpha_{ij-1} \leq \alpha_{ij+1} \\ \frac{1}{\alpha_{ij+1}}\,(u_{i\,j+\alpha} - u_{ij}) & \phi_{ij}\phi_{ij-1} \geq 0 \text{ or } \alpha_{ij+1} < \alpha_{ij-1} \end{cases} \tag{17}$$

Summarizing, in numerical schemes (9) - (11) one has to use the definitions (16) and (17) for the grid nodes next to the interface, for all other grid nodes the definitions (7) and (8) shall be used.

In next section we introduce numerical methods to solve the algebraic equations (9) - (11).

3.2. Numerical solution of algebraic systems (9) - (11)

Before introducing three algorithms to solve the algebraic system of equations (9) - (11), we discuss the possible forms of these equations in details.

Firstly, we consider (9) used for the fast marching and the fast sweeping method. In a generic case when the both components of ∇d_{ij} computed from (7) and (8) are nonzero, the equation (9) has three unknowns and can be written in the form

$$(d_{ij} - d_{i\pm1\,j})^2 + (d_{ij} - d_{i\,j\pm1})^2 = h^2\,. \tag{18}$$

The choice of signs $\pm$ in (18) is determined from the valid cases in conditional definitions (7) or (8). If one component of ∇d_{ij} is zero, the equation (9) has only two unknowns and takes a simpler form

$$(d_{ij} - d_{i\pm1\,j})^2 = h^2 \quad \text{or} \quad (d_{ij} - d_{i\,j\pm1})^2 = h^2\,. \tag{19}$$

Clearly, the correct solutions of quadratic equations (19) are $d_{ij} = d_{i\pm1\,j} + h$ and $d_{ij} = d_{i\,j\pm1} + h$.

Now we discuss the form of (9) for the nodes next to the interface. Firstly, if only one inequality in (12) - (13) is valid, then one component of ∇d_{ij} is computed from (16) or (17) and the other one from (7) or (8). We recognize then two cases. The first case is when the component of ∇d_{ij} computed from (7) or (8) is nonzero, then the equation (9) turns to

$$\frac{d_{ij}^2}{\alpha_{i\pm1\,j}^2} + (d_{ij} - d_{i\,j\pm1})^2 = h^2 \quad \text{or} \quad (d_{ij} - d_{i\pm1\,j})^2 + \frac{d_{ij}^2}{\alpha_{i\,j\pm1}^2} = h^2\,. \tag{20}$$

In the other case one obtains directly

$$\frac{d_{ij}^2}{\alpha_{i\pm1\,j}^2} = h^2 \;\Rightarrow\; d_{ij} = \alpha_{i\pm1\,j} h \quad \text{or} \quad \frac{d_{ij}^2}{\alpha_{i\,j\pm1}^2} = h^2 \;\Rightarrow\; d_{ij} = \alpha_{i\,j\pm1} h\,. \tag{21}$$

Of course, the values d_{ij} in (21) represent the exact distance between x_i and $x_{i\pm\alpha}$ and the exact distance between y_j and $y_{j\pm\alpha}$.

Finally, if the both component of ∇d_{ij} are computed from (16) and (17), the value d_{ij} can be determined from (9) directly by

$$\frac{d_{ij}^2}{\alpha_{i\pm1\,j}^2} + \frac{d_{ij}^2}{\alpha_{i\,j\pm1}^2} = h^2 \;\Rightarrow\; d_{ij} = \frac{\alpha_{i\pm1\,j}\alpha_{i\,j\pm1}}{\sqrt{\alpha_{i\pm1\,j}^2 + \alpha_{i\,j\pm1}^2}} h\,. \tag{22}$$

Again, the choice of plus or minus sign in α is given by the corresponding valid cases in conditional definitions (16) and (17). The value d_{ij} in (22) represents the exact distance of (x_i, y_j) to the line segment between two points $(x_{i\pm\alpha}, y_j)$ and $(x_i, y_{j\pm\alpha})$.

Note that if (9) turns to any particular form (18) - (22), one has that $|\nabla d_{ij}| = 1$. We note that the exact distances in (21) and (22) for the grid nodes next to the interface are fixed in the fast marching method as described in [1], and the form (20) of numerical scheme is never used in [1]. We can show later in section on numerical experiments that using the extrapolation

procedure (20) - (22) with the fast marching method one can improve slightly the results for the chosen examples comparing with the method given in [1].

Now we are ready to present the solution procedures to solve the algebraic system of equations (9). The basic feature of fast marching and fast sweeping method is to suggest such consecutive steps in solving (9) that one has to solve in each step only one scalar quadratic equation with single unknown d_{ij}. It means that if any particular form (18) - (20) has to be solved then the required neighbor values $d_{i\pm1j}$ or $d_{ij\pm1}$ are known at that moment. It is interesting to remark that the scheme (9) can take different particular form (18) - (22) during different steps of solution procedure.

We begin with the introduction of *fast marching method* [9] using the extrapolation procedure. We skip the implementation details like a heapsort technique that can be found in literature and we emphasize the algebraic part of the algorithm.

The basic idea is to label all grid nodes by one of the marks [1]: *Accepted*, *Close* and *Far*. At the beginning of fast marching method with the extrapolation procedure the set of *Accepted* nodes is empty, the set of *Close* nodes is given by all grid nodes next to the interface, and the rest of nodes are labeled as *Far*. Note that in the previous implementations of fast marching method [9, 1], the set of *Accepted* nodes consists of all grid nodes next to the interface for which the values of distance function are computed by some kind of brute force method.

The basic idea of fast marching method is to use the conditional definitions (7) and (8) for the node (x_i, y_j) only with the neighbors $(x_{i\pm1}, y_j)$ and $(x_i, y_{j\pm1})$ that are labeled as *Accepted*. This can be formally viewed as if the neighbor values $d_{i\pm1\,j}$ and $d_{i\,j\pm1}$ for the nodes (x_i, y_j) labeled as *Close* or *Far* are set to infinity. Consequently, at the initial part of algorithm, the values d_{ij} for all *Close* nodes can be computed using only the direct definitions (21) or (22).

The fast marching method consists now in repeating the following step until all nodes are labeled as *Accepted*:

- the minimal value d_{ij} among *Close* nodes is fixed and the corresponding node is labeled as *Accepted*;
- the neighbors of this node with label *Far* (if any) are labeled as *Close*;
- the values for all neighbors of this node that are not yet *Accepted* are recomputed using (18) - (22).

In such a way the fast marching method gives the final results in the finite number of steps where the number of steps equals to the number of grid nodes. In each step at most three values associated with *Close* nodes must be computed by solving one of a quadratic equation (18) - (22) having only one unknown variable d_{ij}. Following [9] the larger solution of two possible solutions to the quadratic equation shall be assigned to d_{ij}, see also the text after (19).

The *fast sweeping method* [10] is iterative method that is based on nonlinear Gauss-Seidel method with four (in 2D case) different orderings ("sweepings") of updates for unknown values d_{ij}. The initial guess is $d_{ij}^0 = C$ where C is a large constant value that can not be reached by the distance function for the given interface and the domain D. The consecutive iterations d_{ij}^{k+1} are obtained by solving the quadratic equation (9) in four alternating orders

$$\begin{aligned} &(1)\, i = 1 : N,\ j = 1 : N, \quad (2)\, i = N : 1,\ j = 1 : N, \\ &(3)\, i = N : 1,\ j = N : 1, \quad (4)\, i = 1 : N,\ j = N : 1. \end{aligned}$$

In each quadratic equation the only unknown is d_{ij}^{k+1} with other values $d_{i\pm1\,j}$ or $d_{i\,j\pm1}$ in (9) (if used) available in the manner of Gauss-Seidel iterations - either from the previous k-th iteration or already computed in the $(k+1)$-th iteration. The detailed description for the solution of quadratic equations is given in [10]. It is important to note that the value d_{ij}^{k+1} is updated in each iteration only if $d_{ij}^{k+1} < d_{ij}^{k}$. Standard criteria can be used based on residuum to decide when to stop the iterations. It can be shown that in a theory a finite number of iterations is necessary to reach the convergence [10].

In the last case we consider the solution procedure for the linearization method based on the linearization of numerical scheme (10). Let $d_{ij}^{(0)}$ be some initial guess of unknowns d_{ij} such that $|\nabla d_{ij}^0| \neq 0$ everywhere, e.g. $d_{ij}^{(0)} = \phi_{ij}$. The linearized form of the scheme (10) takes the form

$$\vec{v}_{ij}^{(m)} \cdot \nabla d_{ij}^{(m+1)} = 1\,. \tag{23}$$

The algebraic system (23) is linear and can be solved by the Gauss-Seidel iterations in four alternating directions. In general it may happen that for some inappropriate choice of d_{ij}^0 one gets $|\nabla d_{ij}^m| = 0$ when $\vec{v}_{ij}^{(m)}$ is undefined if using (10). Therefore the method (23) is suitable only when some good initial guess d_{ij}^0 for the approximative distance function is available.

Finally we comment the solution of linear advection equation (5). Clearly, the proposed numerical scheme (11) is in a character very similar to the left

hand side of (23), so one can again use the Gauss-Seidel iterative methods in four alternating directions for the solution of linear algebraic system. No nonlinear (quadratic) equations have to be solved, therefore we do not consider the fast sweeping method for (5).

The fast marching method to solve (5) is considered in [1] where, in fact, the both equations (3) and (5) are solved in parallel using (9) and (11). In each step of the method, the scheme (11) is a linear equation with a single unknown s_{ij}, therefore it is trivial to solve. The order of nodes (x_i, y_j) in which the unknowns s_{ij} are determined is given by the order in which the *Close* nodes are labeled as *Accepted* in the computations of d_{ij}.

In next section we apply all solution procedures for some standard benchmark examples.

4. Numerical experiments

The purpose of following numerical experiments is to illustrate that the extrapolation procedure for the nodes next to the interface does not decrease the order of accuracy for three solution algorithms. We begin with the computations of approximative distance functions to a smooth interface (a circle), an interface with corners (a square) and a nontrivial interface (a quatrefoil). For the circle and the square the exact solution is available, for the quatrefoil it is computed by an approximative brute force method. The brute force method is approximative in the sense that the distance $d(x_i, y_j)$ to a polygonal interface is computed as the minimal distance to the points that defines the polygon.

In Table 1 we present the discrete L_1 error for four consecutively refined grids computed with three methods with the extrapolation procedure for three interfaces. The experimental order of convergence (EOC) is computed for each method and each interface. The EOC takes approximately the same value for each method, with the value around 1 for the circle and the quatrefoil, and the value around 0.8 for the square.

For the chosen examples the fast marching method and the linearization method give always identical numerical errors. Although all three methods are based on the same numerical scheme (9) of Rouy-Tourin, the fast sweeping method gives different errors for the circle and the quatrefoil. To understand this fact, we modified the original fast sweeping method [10] by removing the condition that the value d_{ij}^{k+1} in each iteration is updated only if $d_{ij}^{k+1} < d_{ij}^{k}$, see the description of the method in section 3.2. After this

N	ML		S		MLS		ML		S	
40	3.553		3.49		4.154		1.746		1.72	
80	1.692	1.07	1.68	1.05	2.267	0.87	0.855	1.03	0.852	1.01
160	0.825	1.04	0.821	1.04	1.283	0.82	0.416	1.04	0.413	1.05
320	0.407	1.02	0.406	1.02	0.734	0.81	0.204	1.03	0.203	1.03

Table 1: The discrete L_1 errors (the magnitude is 10^{-3}) and the EOCs for the distance function to the circle (the columns $2-5$), the square (the columns $6-7$), and the quatrefoil (the columns $8-11$). The fast marching and the linearization method give always identical errors which are given in the columns with ML in headers, the errors obtained by the fast sweeping method are given in the columns with the header S.

modification the fast sweeping method gives identical numerical errors as the other two methods for the chosen examples.

In Figure 1 we plot numerical solutions where the values of computed distance functions inside of interface are multiplied by -1 to obtain so called signed distance functions [8]. In Figure 1 one can see a visual comparison of numerical solution obtained by the fast marching method with the exact solution (for the circle and the square) or the distance obtained by approximative brute force method. We do not compare visually the numerical solutions obtained by the fast sweeping method, because practically no difference is visible.

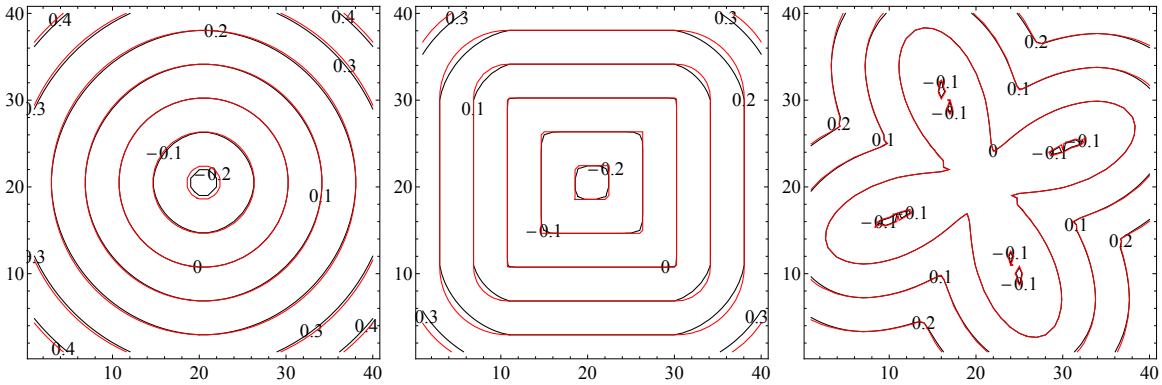

Figure 1: The numerical solutions using the fast marching method on the grid 40×40 (the black contours) compared with the exact solution in the case of circle and square (the red contours in the color version of article) and with the approximative brute force method in the case of quatrefoil.

Next we present a comparison when using the fast marching method with an approximative brute force initialization for the grid nodes next to the interface as described in [1] and the fast marching method with the extrapolation procedure (20) - (22). As we commented in section 3.2 the method in [1] uses only the forms (21) and (22) of (9) for the grid nodes next to the interface.

For the computations of distance to the square the identical numerical errors are obtained. For the computations with the circle and the quatrefoil an improvement of results can be reported for the fast marching method with the extrapolation procedure as given in Table 2.

N	E	EOC	[1]	EOC	E	EOC	[1]	EOC
40	3.553		3.794		1.746		1.882	
80	1.692	1.07	1.844	1.04	0.855	1.03	0.935	1.01
160	0.825	1.04	0.959	0.94	0.416	1.04	0.474	0.98
320	0.407	1.02	0.457	1.07	0.205	1.03	0.240	0.98

Table 2: The discrete L_1 errors (the magnitude is 10^{-3}) and the EOCs for the distance function to the circle (the columns $2-5$) and the quatrefoil (the columns $6-9$). The fast marching method with the extrapolation method is given in the columns with E in headers, the errors obtained by the fast marching method in [1] are given in the columns with [1] in the headers.

Finally, an extension of a function $S = x$ of which the values are taken only on a circle is computed in normal direction and compared with available exact solution [4]. The numerical solutions obtained by the fast marching method and the linearization method are compared in Table 3 and in Figure 2.

N	M	EOC	L	EOC
40	3.59E-3		3.46E-3	
80	1.95E-3	0.88	1.84E-3	0.90
160	1.05E-3	0.89	9.88E-4	0.91
320	5.56E-4	0.91	5.24E-4	0.90

Table 3: The discrete L_1 errors and the EOCs for the extension of function $S = x$ defined only on a circle for the fast marching method (M) and the linearization method (L).

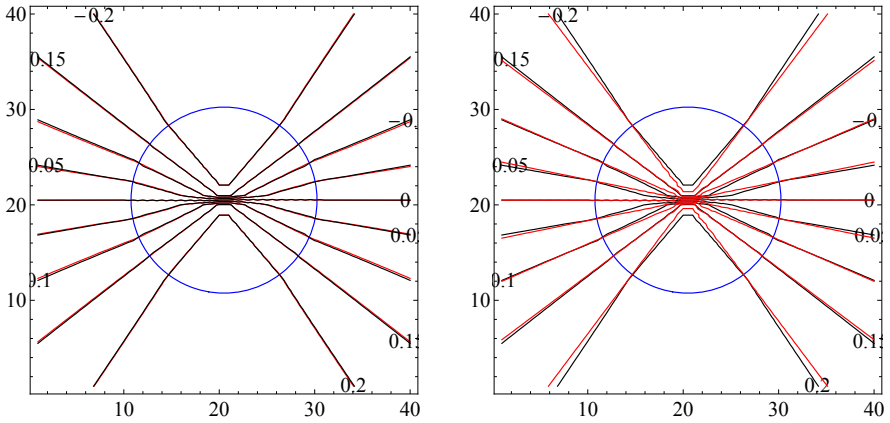

Figure 2: The left picture shows the numerical solution using the linearization method on the grid 40×40 (the black contours) compared with the fast marching method (the red contours in the color version of article) for the extension of a function defined on a circle. The right picture compares the numerical solution using the linearization method (the black contours) compared with the exact solution (the red contours). For a clarity the circle is also plotted in both pictures.

References

[1] D. Adalsteinsson and J. Sethian. The fast construction of extension velocities in level set methods. *J. Comput. Phys.*, 148:2–22, 1999.

[2] Tariq D. Aslam. A partial differential equation approach to multidimensional extrapolation. *J. Comput. Phys.*, 193:349–355, 2003.

[3] S. Fomel. Traveltime computation with the linearized eikonal equation. Technical report, SEP 94, 1997.

[4] P. Frolkovič. Flux-based level set method for extrapolation along characteristics using immersed interface formulation. In P. Struk, editor, *Magia*, pages 15–26. Slovak University of Technology, Bratislava, 2010.

[5] S Hysing and Stefan Turek. The Eikonal equation: numerical efficiency vs. algorithmic complexity on quadrilateral grids. In *Proceedings of Algoritmy, 2005*, pages 22–31, 2005.

[6] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2003.

[7] E. Rouy and A. Tourin. A viscosity solutions approach to shape-from-shading. *SIAM J. Num. Anal.*, 29:867–884, 1992.

[8] J. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[9] J A Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Nat. Acad. Sci.*, 93:1591–1595, 1996.

[10] H. Zhao. A fast sweeping method for eikonal equations. *Math. Comput.*, 74:603–627, 2005.

[11] H K Zhao, T Chan, B Merriman, and S Osher. A Variational Level Set Approach to Multiphase Motion. *J. Comput. Phys.*, 127:179–195, 1996.