Implicit Function with Natural Behavior over Entire Domain[*]

Taku ITOH[a], Ayumu SAITOH[1], Atsushi KAMITANI[2] and Hiroaki NAKAMURA[3]

Seikei University, 3-3-1 Kichijoji-Kitamachi, Musashino, Tokyo 180-8633, Japan
[1]*University of Hyogo, 2167 Shosha, Himeji, Hyogo 671-2280, Japan*
[2]*Yamagata University, 4-3-16 Johnan, Yonezawa, Yamagata 992-8510, Japan*
[3]*National Institute for Fusion Science, 322-6 Oroshi-cho, Toki, Gifu 509-5292, Japan*

(Received 10 December 2011 / Accepted 7 March 2012)

To generate a smooth implicit function that behaves naturally over an entire domain, a method to smoothly combine an implicit function $f(\boldsymbol{x})$ with a global support function $g(\boldsymbol{x})$ has been proposed. The proposed method can be applied to large scattered point data, since the implicit function $f(\boldsymbol{x})$ is generated by a partition-of-unity-based method. The global support function $g(\boldsymbol{x})$ is generated by a radial basis function-based method or by the least-squares method. To ensure a smooth combination of $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$, an appropriate weight function is employed. In numerical experiments, the proposed method is applied to large point data. The results illustrate that the proposed method can generate a smooth implicit function $F(\boldsymbol{x})$ with natural behavior over the entire domain. In addition, on the given points, the accuracy of $F(\boldsymbol{x})$ is exactly the same as that of $f(\boldsymbol{x})$. Furthermore, the computational cost for generation of $F(\boldsymbol{x})$ is almost the same as that of $f(\boldsymbol{x})$.

Keywords: implicit function, meshless method, extended boundary-node method, partition of unity, radial basis function, smooth combination

DOI: 10.1585/pfr.7.2406068

1. Introduction

Many kinds of meshless methods have been proposed, and applied to numerical simulations in various fields, including plasma physics and fusion science. In the meshless methods, although elements representing a geometrical structure are not necessary, an analysis domain must be defined. To define the analysis domain, an implicit function [1–3] is sometimes employed in meshless methods such as the eXtended Boundary-Node Method (X-BNM) [4]. In the X-BNM, it is assumed that, by using the implicit function $f(\boldsymbol{x})$, the boundary of the analysis domain is represented by $f(\boldsymbol{x}) = 0$.

In general, an implicit function $f(\boldsymbol{x})$ has the following properties:

$$\begin{cases} f(\boldsymbol{x}) < 0, & \text{(inside of surface)}, \\ f(\boldsymbol{x}) > 0, & \text{(outside of surface)}. \end{cases} \tag{1}$$

To generate an implicit function $f(\boldsymbol{x})$ from large scattered point data, partition-of-unity-based methods such as the Multi-level Partition of Unity implicits (MPU) method [1] are often employed. However, the generated implicit function $f(\boldsymbol{x})$ has a support; namely, $f(\boldsymbol{x}) = 0$ is distributed not only on the surface but also outside the support. This is not natural behavior for an implicit function, since the requirements of Eq. (1) are not satisfied. Especially in the X-BNM, since a procedure of finding the boundary $f(\boldsymbol{x}) = 0$ is indispensable for evaluating the influence coefficients [4], it is desirable that $f(\boldsymbol{x}) = 0$ is only distributed on the boundary.

The purpose of the present study is to generate an implicit function satisfying the requirements of Eq. (1) over the entire domain. To this end, an implicit function $f(\boldsymbol{x})$ and a global support function $g(\boldsymbol{x})$ are smoothly combined by using an appropriate weight function.

2. Generation of Implicit Function

In this section, we consider generating an implicit function $f(\boldsymbol{x})$ from large scattered point data. The original surface from which the given points were obtained is reconstructed as the implicit representation: $f(\boldsymbol{x}) = 0$, where $\boldsymbol{x} = [x, y, z]^{\mathrm{T}} \in \mathbb{R}^3$. In addition, the function $f(\boldsymbol{x})$ satisfies Eq. (1).

Given a collection of n points that are scattered in a domain, $\mathcal{P} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, together with normals $\mathcal{N} = \{\boldsymbol{n}_1, \boldsymbol{n}_2, \ldots, \boldsymbol{n}_n\}$ on each of the given points [3]. We can think of the surface as a scalar-valued function $f(\boldsymbol{x})$ such that $f(\boldsymbol{x}_i) = 0$ $(i = 1, 2, \ldots, n)$.

To generate an implicit function $f(\boldsymbol{x})$ from a number of points, the partition-of-unity-based methods have been proposed by Ohtake et al. [1] and Tobor et al. [2]. In these methods, a bounded domain Ω is divided into a set of subdomains $\Omega^{(k)}$ $(k = 1, 2, \ldots, M)$, where M is the number of subdomains. Note that each subdomain is defined as a sphere in Ref. [1] and as an ellipsoid or a rectangle in Ref. [2]. In addition, adjacent subdomains are slightly overlapped. For each subdomain, a local function is com-

author's e-mail: taku@m.ieice.org
[a] Present address: Tokyo University of Technology, 1404-1 Katakura-machi, Hachioji, Tokyo 192-0982, Japan
[*] This article is based on the presentation at the 21st International Toki Conference (ITC21).

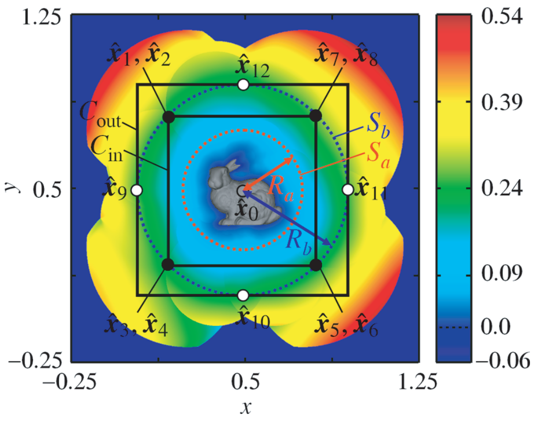

Fig. 1 Distribution of $f(\boldsymbol{x})$ on the x-y plane with $z = 0.5$. Here, $f(\boldsymbol{x})$ is normally generated by the method of Tobor et al. [2]. An implicit surface $f(\boldsymbol{x}) = 0$ is also illustrated in the middle of this figure. In addition, fifteen sub-points $\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_{14}$ to generate $g(\boldsymbol{x})$ are illustrated, except for $\hat{\boldsymbol{x}}_{13}$ and $\hat{\boldsymbol{x}}_{14}$. The sub-point positions are determined by spheres S_a and S_b and cubes C_{in} and C_{out}. Here, $\hat{\boldsymbol{x}}_0$ is the center of S_a and S_b, and R_a and R_b are the radii of S_a and S_b, respectively. Note that the point data is "Bunny" model.

puted from the points in the subdomain. A function $f(\boldsymbol{x})$ defined on Ω is then defined as a combination of the local functions weighted by the partition functions $w^{(k)}(\boldsymbol{x})$ such that

$$\sum_{k=1}^{M} w^{(k)}(\boldsymbol{x}) \equiv 1 \text{ on } \Omega. \tag{2}$$

Thus, $f(\boldsymbol{x})$ has the following expression [1, 2]:

$$f(\boldsymbol{x}) = \sum_{k=1}^{M} w^{(k)}(\boldsymbol{x}) f^{(k)}(\boldsymbol{x}), \quad w^{(k)}(\boldsymbol{x}) = \frac{\omega^{(k)}(\boldsymbol{x})}{\sum_{j=1}^{M} \omega^{(j)}(\boldsymbol{x})}, \tag{3}$$

where $f^{(k)}(\boldsymbol{x})$ is a local function in $\Omega^{(k)}$, and $\omega^{(k)}(\boldsymbol{x})$ is a nonnegative compactly supported function on $\Omega^{(k)}$. In the MPU method, the quadratic B-spline is adopted as $\omega^{(k)}(\boldsymbol{x})$ [1]. Note that $w^{(k)}(\boldsymbol{x})$ $(k = 1, 2, \ldots, M)$ are called "Partition of Unity (PU) functions."

By using an implicit function $f(\boldsymbol{x})$ generated by a PU-based method, an object can be represented as $f(\boldsymbol{x}) = 0$. However, $f(\boldsymbol{x}) = 0$ also appears in a domain beyond a certain distance from the given points (see Fig. 1). In other words, a PU-based method generates an implicit function $f(\boldsymbol{x})$ so that $f(\boldsymbol{x})$ satisfies Eq. (1) around the given points. Thus, $f(\boldsymbol{x})$ has a support; that is, $f(\boldsymbol{x}) = 0$ is distributed not only on the surface but also outside the support. This is not natural behavior for an implicit function since Eq. (1) is not satisfied. In the next section, we present a method to generate a smooth implicit function with natural behavior over the entire domain.

3. Smooth Implicit Function with Natural Behavior

In the following section, $f(\boldsymbol{x})$ denotes an implicit function generated by a PU-based method, and $g(\boldsymbol{x})$ denotes a global support function. In this section, we present a method to erase the support of $f(\boldsymbol{x})$ by smoothly combining $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$.

3.1 Rescaling given points

The given points $\mathcal{P}$ are first rescaled so that an axis-aligned bounding cube C_1 has a unit-length main diagonal. Concretely, the given points $\mathcal{P}$ are normalized and lie in the cube C_1 so that C_1 is placed in the central part of a unit cube C_2: $[0, 1] \times [0, 1] \times [0, 1]$. The normalization procedures are as follows:

1. $\boldsymbol{x}_k = \boldsymbol{x}_k - a_{\min}\boldsymbol{u}$ $(k = 1, 2, \ldots, n)$, where $a_{\min}$ is the minimum value of x_k, y_k, and z_k $(k = 1, 2, \ldots, n)$, and $\boldsymbol{u} = [1, 1, 1]^{\mathrm{T}}$;
2. $\boldsymbol{x}_k = \boldsymbol{x}_k / (\sqrt{3} a_{\max})$ $(k = 1, 2, \ldots, n)$, where $a_{\max}$ is the maximum value of x_k, y_k, and z_k $(k = 1, 2, \ldots, n)$;
3. $x_k = x_k + (\sqrt{3}^{-1} - x_{\min} - x_{\max})/2$ $(k = 1, 2, \ldots, n)$, where $x_{\min}$ and $x_{\max}$ are the minimum and maximum values of x_k $(k = 1, 2, \ldots, n)$, respectively [y_k and z_k $(k = 1, 2, \ldots, n)$ are similarly updated]; and
4. $\boldsymbol{x}_k = \boldsymbol{x}_k + \beta\boldsymbol{u}$ $(k = 1, 2, \ldots, n)$, where $\beta = (1 - \sqrt{3}^{-1})/2$.

After executing the above procedures, $f(\boldsymbol{x})$ is generated inside C_2 by a PU-based method.

3.2 Generation of global support function

To generate a global support function $g(\boldsymbol{x})$, we first consider a sphere S_a of radius R_a and center $\hat{\boldsymbol{x}}_0 = [0.5, 0.5, 0.5]^{\mathrm{T}}$, and a sphere S_b of radius R_b and center $\hat{\boldsymbol{x}}_0$, where R_a is determined so that all given points $\mathcal{P}$ are contained inside S_a, and R_b satisfies $R_b > R_a$. In addition, we also consider a cube C_{in} inscribed in S_b and a cube C_{out} circumscribed about S_b.

A global support function $g(\boldsymbol{x})$ is generated from fifteen sub-points $\hat{\mathcal{P}} = \{\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_{14}\}$, where $\hat{\boldsymbol{x}}_1, \hat{\boldsymbol{x}}_2, \ldots, \hat{\boldsymbol{x}}_8$ are the eight vertices of C_{in}, and $\hat{\boldsymbol{x}}_9, \hat{\boldsymbol{x}}_{10}, \ldots, \hat{\boldsymbol{x}}_{14}$ are six tangent points between S_b and C_{out} (see Fig. 1). In the following, we present two types of methods to generate $g(\boldsymbol{x})$.

3.2.1 generation of $g(\boldsymbol{x})$ by radial basis function-based method

By using a Radial Basis Function-based Method (RBFM) [3], an implicit function with natural behavior can be directly generated from $\mathcal{P}$ and $\mathcal{N}$. However, the RBFM is not suitable to generate an implicit function from large point data, since the computational cost for solving a linear system that depends on the number of nodes is very large. Note that, for generating $g(\boldsymbol{x})$, we only use the fifteen sub-points $\{\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_{14}\}$. Hence, the computational cost to generate $g(\boldsymbol{x})$ is small.

We assume that a global support function $g(\boldsymbol{x})$ is expressed as

$$g(\boldsymbol{x}) = \sum_{i=0}^{14} \lambda_i \phi(\|\boldsymbol{x} - \hat{\boldsymbol{x}}_i\|_2) + p(\boldsymbol{x}), \tag{4}$$

where $\phi(r)$ is a radial basis function (RBF), $\lambda_i (i = 0, 1, \ldots, 14)$ are weights, $\hat{\boldsymbol{x}}_i = [\hat{x}_i, \hat{y}_i, \hat{z}_i]^{\mathrm{T}} (i = 0, 1, \ldots, 14)$ are the sub-points, and $p(\boldsymbol{x})$ is a degree-one polynomial: $p(\boldsymbol{x}) = \alpha_0 + \alpha_1 x + \alpha_2 y + \alpha_3 z$. Under the assumptions $g(\hat{\boldsymbol{x}}_i) = f(\hat{\boldsymbol{x}}_i)$ $(i = 0, 1, \ldots, 14)$, the unknowns λ_i and α_i are determined by solving the following linear system [3]:

$$\begin{bmatrix} A & P \\ P^{\mathrm{T}} & O \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\alpha} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{0} \end{bmatrix}, \tag{5}$$

where $\boldsymbol{\lambda} = [\lambda_0, \lambda_1, \ldots, \lambda_{14}]^{\mathrm{T}} \in \mathbb{R}^{15}$, $\boldsymbol{\alpha} = [\alpha_0, \alpha_1, \alpha_2, \alpha_3]^{\mathrm{T}} \in \mathbb{R}^4$, $\boldsymbol{f} = [f(\hat{\boldsymbol{x}}_0), f(\hat{\boldsymbol{x}}_1), \ldots, f(\hat{\boldsymbol{x}}_{14})]^{\mathrm{T}} \in \mathbb{R}^{15}$, ith-rows of $P \in \mathbb{R}^{15\times 4}$ are $[1, \hat{x}_i, \hat{y}_i, \hat{z}_i]$ $(i = 0, 1, \ldots, 14)$ and (i, j)-elements of $A \in \mathbb{R}^{15\times 15}$ are $\phi(\|\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_j\|_2)$ $(i, j = 0, 1, \ldots, 14)$. Note that a globally supported RBF such as the triharmonic: $\phi(r) = r^3$ has to be employed, since $g(\boldsymbol{x})$ must be generated as a global support function.

3.2.2 generation of $g(x)$ by the least-squares method

In a different approach, we employ the Least-Squares Method (LSM) to generate a global function $g(\boldsymbol{x})$. To generate $g(\boldsymbol{x})$ using LSM, we assume that $g(\boldsymbol{x})$ is a three-dimensional (3D) quadratic function expressed as

$$\begin{aligned} g(\boldsymbol{x}) = a_1 x^2 + a_2 y^2 + a_3 z^2 + a_4 xy + a_5 yz + a_6 zx \\ + a_7 x + a_8 y + a_9 z + a_{10}. \end{aligned} \tag{6}$$

The unknowns $a_k (k = 1, 2, \ldots, 10)$ are determined by minimizing $\sum_{i=0}^{14} [g(\hat{\boldsymbol{x}}_i) - f(\hat{\boldsymbol{x}}_i)]^2$.

3.3 Smooth combination of $f(x)$ and $g(x)$

To smoothly combine $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$, we adopt the following weight function:

$$w(r) = \begin{cases} 1 & ; 0 \le r \le R_a, \\ 1 - 6\hat{r}^2 + 8\hat{r}^3 - 3\hat{r}^4 & ; R_a < r \le R_b, \\ 0 & ; r > R_b, \end{cases} \tag{7}$$

where $r = \|\boldsymbol{x} - \hat{\boldsymbol{x}}_0\|_2$, and $\hat{r} = (r - R_a)/(R_b - R_a)$. Figure 2 shows the shape of the weight function $w(r)$ and that of $1 - w(r)$. Note that R_a and R_b are determined by (i) $R_a^{(0)} = R_a^{(\mathrm{ini})}$, (ii) repeat $R_a^{(k)} = R_a^{(k-1)} + \beta_a R_a^{(\mathrm{ini})}$ $(k = 1, 2, \ldots)$ until $S_a^{(k)}$ contains all given points $\mathcal{P}$, where $S_a^{(k)}$ is a sphere of radius $R_a^{(k)}$ with center $\hat{\boldsymbol{x}}_0$, (iii) $R_a = R_a^{(k)}$, and (iv) $R_b = R_a + \beta_b$. In our implementation, we set $R_a^{(\mathrm{ini})} = 0.2$, $\beta_a = 0.05$, and $\beta_b = 0.1$.

By using the weight function above, $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ can be smoothly combined. A combined implicit function $F(\boldsymbol{x})$ is described as

$$F(\boldsymbol{x}) = w(r) f(\boldsymbol{x}) + [1 - w(r)] g(\boldsymbol{x}). \tag{8}$$

In the range $R_a < r \le R_b$, the distribution of $F(\boldsymbol{x})$ is determined by the smooth combination of $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$. Note

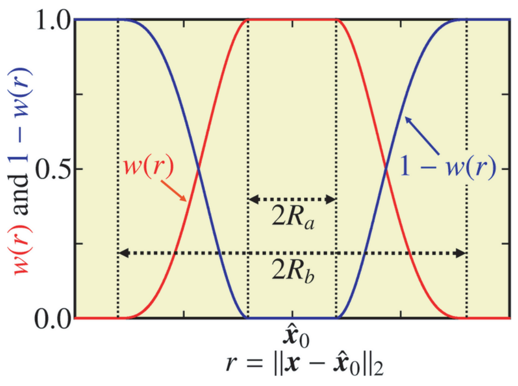

Fig. 2 Shape of the weight function $w(r)$ and that of $1 - w(r)$. Here, $\hat{\boldsymbol{x}}_0, R_a$, and R_b are the same defined in Fig. 1.

that, in the range $0 \le r \le R_a$, the distribution of $F(\boldsymbol{x})$ is the same as that of $f(\boldsymbol{x})$, since $w(r) = 1$ in this range. Similarly, in $r > R_b$, the distribution of $F(\boldsymbol{x})$ is the same as that of $g(\boldsymbol{x})$, which does not have a support. For these reasons, $F(\boldsymbol{x})$ satisfies Eq. (1) over the entire domain.

4. Numerical Experiments

Numerical experiments using the point data of the "Lucy" model are conducted to evaluate the method described in the previous section. Computations were performed on a computer equipped with a 2.66 GHz Intel Core i7 920 processor, 24 GB RAM, Ubuntu Linux ver. 11.10, and g++ ver. 4.6.1. Note that we only used a single core of this processor in the experiments described herein.

Let us first investigate a distribution of $f(\boldsymbol{x})$ and that of $F(\boldsymbol{x})$. Figure 3 (a) shows a distribution of $f(\boldsymbol{x})$ generated by the MPU method. In addition, distributions of two kinds of $F(\boldsymbol{x})$ are shown in Figs. 3 (b) and 3 (c), respectively. In Figs. 3 (b) and 3 (c), the functions $g(\boldsymbol{x})$ are generated by RBFM and LSM, respectively. In Fig. 3, the implicit surfaces $f(\boldsymbol{x}) = 0$ or $F(\boldsymbol{x}) = 0$ are also illustrated in the middle of each figure. To generate $f(\boldsymbol{x})$, we set $\alpha = 1.2$, $\lambda = 0.05$, and $\varepsilon = 10^{-3}$, as described in Ref. [1]. In addition, other parameters to generate $f(\boldsymbol{x})$ are set the same as in Ref. [1]. By using the procedures described in the previous section, R_a and R_b are determined to be 0.34 and 0.44, respectively. In addition, the triharmonic: $\phi(r) = r^3$ is employed as the RBF to generate $g(\boldsymbol{x})$ in Fig. 3 (b). We see from Figs. 3 (b) and 3 (c) that the functions $F(\boldsymbol{x})$ do not have a support. Figure 4 shows the dependence of $f(\boldsymbol{x})$ and $F(\boldsymbol{x})$ on x with $y = z = 0.5$. In this figure, "$f(\boldsymbol{x})$", "$F(\boldsymbol{x})$ (RBFM)," and "$F(\boldsymbol{x})$ (LSM)" correspond to the functions in Figs. 3 (a), 3 (b), and 3 (c), respectively. We see from Figs. 3 and 4 that, for $r \le R_a$, $f(\boldsymbol{x})$ and $F(\boldsymbol{x})$ are exactly the same. In the range $R_a < r \le R_b$, $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ are merged by the weight function. For $r > R_b$, both functions $F(\boldsymbol{x})$ change more smoothly than $f(\boldsymbol{x})$. Therefore we consider that the functions $g(\boldsymbol{x})$ generated by RBFM and LSM, together with the appropriate weight function, can generate a smooth implicit function $F(\boldsymbol{x})$ satisfying the requirements of Eq. (1)

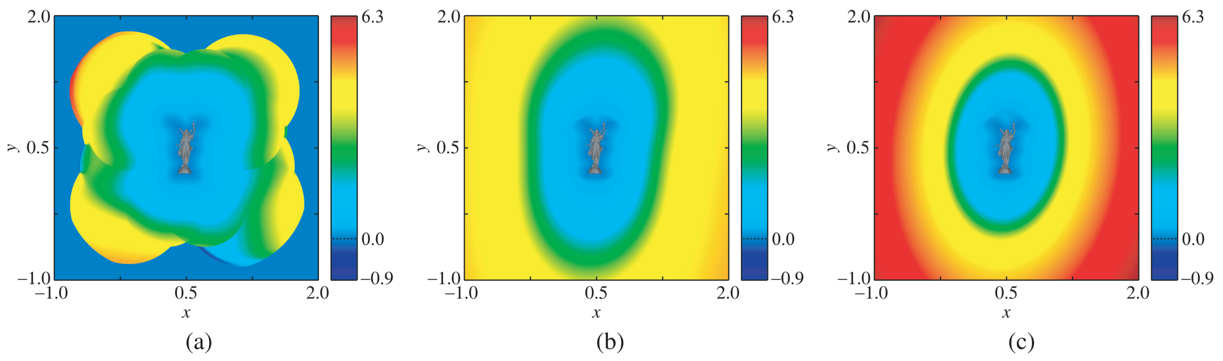

Fig. 3 Distributions of $f(\boldsymbol{x})$ and $F(\boldsymbol{x})$ on the x-y plane with $z = 0.5$. For (a), (b), and (c), $f(\boldsymbol{x})$ is generated by the MPU method [1]. In addition, for obtaining $F(\boldsymbol{x})$ using Eq. (8), $g(\boldsymbol{x})$ is generated by RBFM for (b) and by LSM for (c). Note that an implicit surface $f(\boldsymbol{x}) = 0$ is also shown in the middle of (a). Similarly, implicit surfaces $F(\boldsymbol{x}) = 0$ are shown in panels (b) and (c). The point data is the "Lucy" model and the number n of points is 1001991.

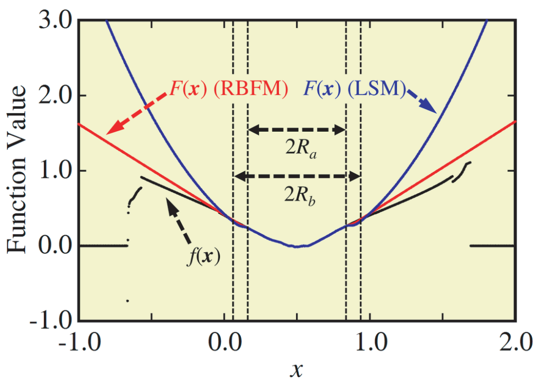

Fig. 4 Dependence of $f(\boldsymbol{x})$ and $F(\boldsymbol{x})$ on x with $y = z = 0.5$. Here, "$f(\boldsymbol{x})$", "$F(\boldsymbol{x})$ (RBFM)," and "$F(\boldsymbol{x})$ (LSM)" correspond to the functions in Figs. 3 (a), 3 (b), and 3 (c), respectively.

over the entire domain. The difference between the functions $F(\boldsymbol{x})$ is that, for $r > R_b$, $F(\boldsymbol{x})$ generated with RBFM increases more slowly than that generated with LSM.

Next, to evaluate the error of $f(\boldsymbol{x})$ in Fig. 3 (a), we employ the maximum error $\varepsilon_{\max} = \max\{|f(\boldsymbol{x}_i)|\}_{i=1}^{n}$. The result is $\varepsilon_{\max} = 5.9 \times 10^{-4}$. Similarly, we calculate the maximum errors of the functions $F(\boldsymbol{x})$, and find exactly the same as $\varepsilon_{\max}$ of $f(\boldsymbol{x})$. Thus, on the given points, the accuracy of $f(\boldsymbol{x})$ is not affected by Eq. (8).

Finally, we investigate the computational cost of the proposed method. The computational time to generate $f(\boldsymbol{x})$ is about 46.8 s. In addition, the computational time of 0.006 s to generate $g(\boldsymbol{x})$ by RBFM is almost equal to that by LSM. Therefore the computational cost to generate $F(\boldsymbol{x})$ is almost the same as that of $f(\boldsymbol{x})$, since $g(\boldsymbol{x})$ can be generated with a small computational cost.

5. Conclusion

To generate a smooth implicit function with natural behavior over an entire domain, we have proposed a method to smoothly combine an implicit function $f(\boldsymbol{x})$ and a global support function $g(\boldsymbol{x})$. To generate $f(\boldsymbol{x})$ from large point data, the implicit function $f(\boldsymbol{x})$ is generated by a PU-based method. To generate $g(\boldsymbol{x})$, we appropriately set the fifteen sub-points and generate $g(\boldsymbol{x})$ by RBFM or LSM. A smooth implicit function $F(\boldsymbol{x})$ is generated by smoothly combining $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ using the proposed weight function. The performance of the proposed method has been investigated by using the "Lucy" model. Conclusions obtained in the present study are summarized as follows:

1. The global support functions $g(\boldsymbol{x})$ generated by RBFM and LSM together with the appropriate weight function can generate a smooth implicit function $F(\boldsymbol{x})$ that satisfies the requirements of Eq. (1) over the entire domain.
2. On the given points, the accuracy of $F(\boldsymbol{x})$ is exactly the same as that of $f(\boldsymbol{x})$.
3. The computational cost for generation of $F(\boldsymbol{x})$ is almost identical to that of $f(\boldsymbol{x})$.

In future work, an implicit function generated by the proposed method will be built into meshless methods such as the X-BNM. In addition, the built meshless methods will be applied for solving partial differential equations of various fields, including plasma physics and fusion science.

Acknowledgments

We would like to thank the Stanford 3D Scanning Repository for the Bunny and Lucy models. This work was supported by KAKENHI (No. 22360042) and by the NIFS Collaboration Research Program (NIFS11KKGS009 and NIFS11KNTS011).

[1] Y. Ohtake, A. Belyaev, M. Alexa, G. Turk and H.-P. Seidel, ACM Trans. on Graph. **22**, 463 (2003).

[2] I. Tobor, P. Reuter and C. Schlick, WSCG **12**, 467 (2004).

[3] G. Turk and J.F. O'Brien, ACM Trans. on Graph. **21**, 855 (2002).

[4] T. Itoh, A. Saitoh, A. Kamitani and H. Nakamura, Plasma Fusion Res. **5**, S2111 (2010).