AIAA SciTech Forum
January 3-7, 2022, San Diego, CA & Virtual
AIAA SCITECH 2022 Forum

10.2514/6.2022-1862

An Explicit Level-Set Formula to Approximate Geometries

Jason E. Hicken* and Sharanjeet Kaur†
Rensselaer Polytechnic Institute, Troy, New York, 12180

We present a smooth, differentiable formula that can be used to approximate an existing geometry as a level-set function. The formula uses data from a finite number of points on the surface and does not require solving a linear or nonlinear system, i.e. the formula is explicit. The baseline method is a smooth analog of a piecewise linear approximation to the surface, but a quadratic correction can be constructed using curvature information. Numerical experiments explore the accuracy of the level-set formula and the influence of its free parameters. For smooth geometries, the results show that the linear and quadratic versions of the method are second- and third-order accurate, respectively. For non-smooth geometries, the infinity norm of the error converges at a first-order rate.

I. Introduction

Level-set methods [1] use an implicit function to define domains, boundaries, and interfaces. For example, a domain $\Omega \subset \mathbb{R}^D$ and its boundary Γ can be defined as

$$\Omega = \{\boldsymbol{x} \in \mathbb{R}^D \mid \phi(\boldsymbol{x}) > 0\}, \qquad \text{and} \qquad \Gamma = \{\boldsymbol{x} \in \mathbb{R}^D \mid \phi(\boldsymbol{x}) = 0\}, \tag{1}$$

where $\phi : \mathbb{R}^D \to \mathbb{R}$ is the level-set function (LSF). Level-set methods have been successfully applied to a wide range of applications, including moving boundaries/interfaces [1, 2], image segmentation [3], and topology optimization [4–7].

Our interest in level-set methods stems from an esoteric problem that arises in cut-cell finite-volume [8–10] and finite-element methods [11–13]. Cut-cell methods use grids that consist of regular elements and "cut" elements that intersect with the domain boundary; see Figure 1b. To evaluate the finite-element weak form, cut-cell methods require specialized integration rules for the irregularly-shaped cut elements. To address this, Saye [14] proposed and implemented an algorithm that constructs high-order quadrature rules for hypercubes that are "cut" by the zero contour of a LSF.

Saye's algorithm relies on an expansion of the LSF in a first-order Taylor series with a bounded remainder, which places limitations on the form that the function ϕ can take. In particular, commonly used non-smooth functions, such as `max` and `min`, cannot be employed, and conditional statements must be avoided. Instead, ϕ must be composed of common mathematical operations, such as powers, logarithms, and trigonometric functions.

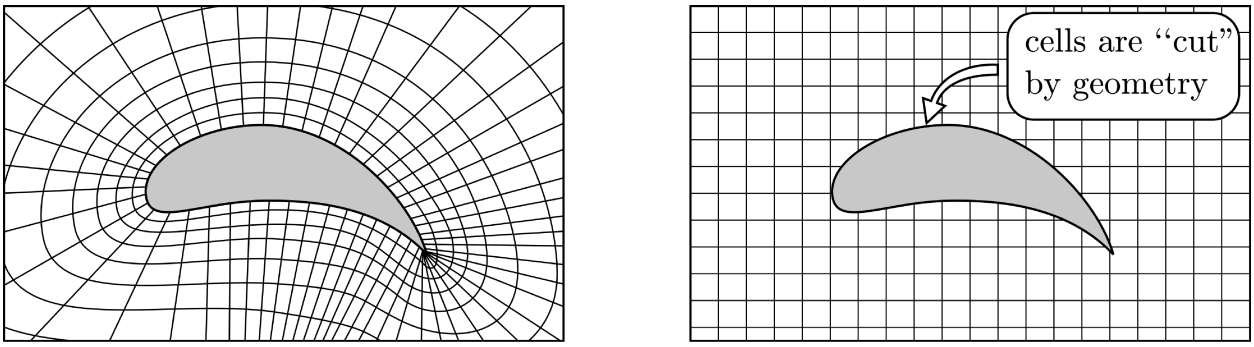
In order to use Saye's algorithm for cut-cell finite-element discretizations, we need smooth LSF approximations of complex geometries. This provides the motivation and objective behind the current work:

> *For a target geometry, Γ, find a smooth level set function ϕ such that the set of points that satisfy $\phi(\boldsymbol{x}) = 0$ is a sufficiently accurate approximation of Γ. Furthermore, $\phi(\boldsymbol{x})$ should be computationally inexpensive to evaluate.*

Several methods have been developed to construct a level-set for an existing geometry, but they are not well suited to our intended application. For instance, the LSF can be represented using the same piecewise polynomial space as the finite-element method [14]; however, polynomials yield poor approximations of sharp corners, which are ubiquitous in practice (e.g. the trailing edge of an airfoil). Radial-basis functions (RBFs) can be used to reconstruct an LSF from a given point cloud [15, 16], but the RBF approach requires the solution of a large, possibly dense, linear system. We would prefer to avoid this additional cost, because the LSF may need to change many times during an optimization or unsteady flow simulation. An example of a fast-to-evaluate LSF is the L^p distance function [17], which uses an explicit formula for the signed distance function; unfortunately, our experiments (not reported here) found that the discrete L^p distance function produces poor approximations of Γ.

The approach we propose herein adapts constraint aggregation techniques — specifically, an induced aggregate [18] — to construct an explicit formula for the LSF. For an arbitrary point $\boldsymbol{x}$, the formula approximates the signed distance

*Associate Professor, Mechanical, Aerospace, and Nuclear Engineering, AIAA Associate Fellow.
†Graduate Student, Mechanical, Aerospace, and Nuclear Engineering, AIAA Student Member.



(a) Conformal mesh. **(b) Cut-cell mesh.**

Fig. 1 Examples illustrating a conventional conformal mesh (left) and a cut-cell mesh (right). Specialized quadrature rules are often required for the elements cut by the geometry in the case of the cut-cell mesh.

function using the surface point closest to $\boldsymbol{x}$ and its unit normal. The discrete surface data does not need to be structured, so point clouds suffice. Like most constraint aggregation functions, a parameter controls the smoothness of the LSF, and the limiting approximation is a piecewise linear function. Consequently, the approximation converges at a second-order rate with the spacing between surface points. We show how this rate can be increased to third-order by including local curvature information.

The remaining paper is structured as follows. Section II describes the proposed LSF. Section III investigates the method using some numerical experiments, and verifies its predicted rate of convergence. We conclude with a summary and discussion of future work in Section IV.

II. Proposed Level-Set Function

A. Inspiration from the signed distance function

Our level-set formula is inspired by the signed distance function, so we begin with a brief review of this function. As in (1), let Γ denote the boundary of some domain — or, in practice, the closed surface of some geometry of interest. The signed distance function corresponding to Γ is

$$d_\Gamma(\boldsymbol{x}) = \begin{cases} \inf_{\boldsymbol{y}\in\Gamma} \|\boldsymbol{x}-\boldsymbol{y}\|, & \boldsymbol{x}\in\Omega, \\ -\inf_{\boldsymbol{y}\in\Gamma} \|\boldsymbol{x}-\boldsymbol{y}\|, & \boldsymbol{x}\in\mathbb{R}^D\setminus\Omega, \end{cases}$$

where inf denotes the infimum*. In words, the magnitude of $d_\Gamma(\boldsymbol{x})$ is the shortest distance from $\boldsymbol{x}$ to Γ. The sign of $d_\Gamma(\boldsymbol{x})$ is positive if $\boldsymbol{x}$ lies in the domain Ω, and negative otherwise.

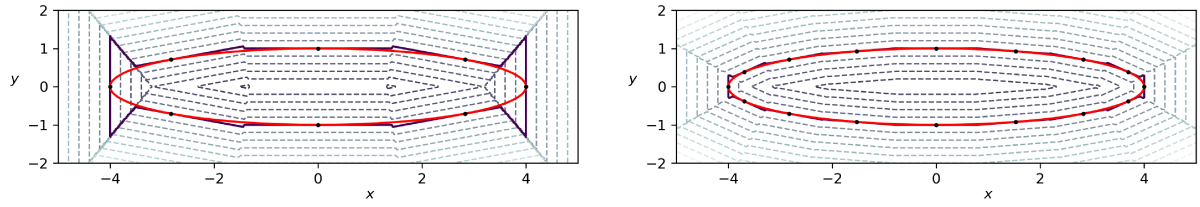
The signed distance function is a LSF that is suitable for some applications, but it has some issues that make it unattractive for our purposes. For example, evaluating the signed distance function for an arbitrary Γ can be computationally expensive. One way to reduce this expense is to replace Γ with a finite set of points. Let $\Gamma_h = \{\boldsymbol{x}_i\}_{i=1}^{n_\Gamma} \subset \Gamma$ denote such a finite set, with $h = \max_i \min_{j\neq i} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$ defining the largest distance between neighboring points. Then, a discrete approximation to the signed distance function is

$$d_{\Gamma_h}(\boldsymbol{x}) = \begin{cases} \min_i \|\boldsymbol{x}-\boldsymbol{x}_i\|, & \boldsymbol{x}\in\Omega, \\ -\min_i \|\boldsymbol{x}-\boldsymbol{x}_i\|, & \boldsymbol{x}\in\mathbb{R}^D\setminus\Omega. \end{cases}$$

While $d_{\Gamma_h}(\boldsymbol{x})$ is more tractable to compute than $d_\Gamma(\boldsymbol{x})$, it does not eliminate all the issues with the signed distance function, and it introduces its own issues:

1) The min operator is a non-smooth function, which is not permitted by Saye's algorithm. Non-smoothness also arises in the signed-distance function, so this problem is not unique to the discrete version $d_{\Gamma_h}(\boldsymbol{x})$.
2) Determining if $\boldsymbol{x}\in\Omega$ may be difficult; indeed, if we could determine this, we would probably have a good level-set already.

*The infimum is a generalization of the minimum to sets that may not have a minimum.



(a) $n_\Gamma = 8$ **(b)** $n_\Gamma = 16$

Fig. 2 The piecewise-linear LSF (2) applied to an ellipse with semi-major axis 4 and semi-minor axis 1. The boundary Γ is shown in red, and the approximation given by $\phi(\boldsymbol{x}) = 0$ is shown in black. Additional contour lines are shown as dashed lines.

3) Even if issues 1 and 2 could be resolved, the level-set produced by $d_{\Gamma_h}(\boldsymbol{x})$ is discontinuous at points on $\Gamma \setminus \Gamma_h$; that is, at points on the true surface but not in the finite subset of sampled boundary points.

We now present an alterative level-set function that resolves issues 2 and 3. Non-smoothness will be addressed in the subsequent section.

Consider the particularly simple case where Γ is given by a hyperplane in D dimensions, i.e. a line in two-dimensions or a plane in three-dimensions. In this case, a single point, $\boldsymbol{x}_i$, and normal, $\hat{\boldsymbol{n}}_i$, suffice to define Γ and its signed distance function, which we will denote by $d_i(\boldsymbol{x})$:

$$d_i(\boldsymbol{x}) \equiv (\boldsymbol{x} - \boldsymbol{x}_i)^T \hat{\boldsymbol{n}}_i.$$

We can use the distance function for a hyperplane to develop an approximate distance function for more complex surfaces. The idea is to use the formula $d_i(\boldsymbol{x})$ corresponding to whichever point $\boldsymbol{x}_i$ is closest to $\boldsymbol{x}$.

$$\phi(\boldsymbol{x}) \equiv \left\{ d_i(\boldsymbol{x}) \mid i = \text{argmin}_{j \in \{1,2,\ldots,n_\Gamma\}} \Delta_j(\boldsymbol{x}) \right\}, \tag{2}$$

where, for later convenience, we have introduced

$$\Delta_i(\boldsymbol{x}) \equiv \|\boldsymbol{x} - \boldsymbol{x}_i\|.$$

The approximation (2) to the signed distance function will produce a piecewise linear level-set function, with discontinuities between the linear approximations. This is illustrated in Figure 2, which shows the approximation to the signed-distance function applied to an elliptical boundary.

B. Addressing non-smoothness with aggregation

We adapt constraint aggregation methods, also known as smooth max/min functions, to address the non-smoothness issue described in the previous section. For example, a smooth approximation to $\max_i \Delta_i(\boldsymbol{x})$ is given by the Kreisselmeier–Steinhauser (KS) aggregation

$$\text{KS}(\boldsymbol{x}) = \frac{1}{\rho} \ln \left(\frac{1}{\alpha} \sum_{i=1}^{n_\Gamma} e^{\rho \Delta_i(\boldsymbol{x})} \right) \tag{3}$$

where $\rho > 0$ and $\alpha > 0$ are parameters. It is straightforward to show that $\text{KS}(\boldsymbol{x}) \to \max_i \Delta_i(\boldsymbol{x})$ as the parameter $\rho \to \infty$.

Unfortunately, we were unable to adapt the KS aggregation (3) to approximate (2). Notice that the distance approximation (2) returns one value based on another value: $d_i(\boldsymbol{x})$ is returned if $\Delta_i(\boldsymbol{x})$ is the smallest distance. By contrast, the KS function returns an approximation to $\Delta_i(\boldsymbol{x})$ based on the other $\Delta_j(\boldsymbol{x})$ values.

An alternative smooth max/min function that can be adapted to approximate (2) is given by the induced aggregate [18]

$$S(\boldsymbol{x}) = \frac{\sum_{i=1}^{n_\Gamma} \Delta_i(\boldsymbol{x}) e^{\rho \Delta_i(\boldsymbol{x})}}{\sum_{i=1}^{n_\Gamma} e^{\rho \Delta_i(\boldsymbol{x})}}, \tag{4}$$

where $\rho > 0$ gives an approximation to $\max_i \Delta_i(\boldsymbol{x})$ and $\rho < 0$ gives an approximation to $\min_i \Delta_i(\boldsymbol{x})$. To make this function approximate (2), we replace the $\Delta_i(\boldsymbol{x})$ that appears in the numerator multiplying $e^{\rho \Delta_i(\boldsymbol{x})}$ with the function that we want as the "return" value, i.e. $d_i(\boldsymbol{x})$.

Thus, we arrive at our proposed level-set function:

$$\phi(\boldsymbol{x}) = \frac{\sum_{i=1}^{n_\Gamma} d_i(\boldsymbol{x}) e^{-\rho\Delta_i(\boldsymbol{x})}}{\sum_{j=1}^{n_\Gamma} e^{-\rho\Delta_j(\boldsymbol{x})}}, \tag{5}$$

where $\rho > 0$ now, since we have introduced a minus sign in the argument of the exponentials.

The level-set function (5) can be interpreted as a generalized linear model with a spatially-varying coefficient. That is, we can rewrite the function as a sum of weighted basis functions, where each basis function is associated with a $\boldsymbol{x}_i \in \Gamma_h$:

$$\phi(\boldsymbol{x}) = \sum_{i=1}^{n_\Gamma} d_i(\boldsymbol{x})\psi_i(\boldsymbol{x}), \tag{6}$$

where the ith basis function is given by

$$\psi_i(\boldsymbol{x}) = \frac{e^{-\rho\Delta_i(\boldsymbol{x})}}{\sum_{j=1}^{n_\Gamma} e^{-\rho\Delta_j(\boldsymbol{x})}}. \tag{7}$$

Some comments on these basis functions are warranted.

- The basis functions form a partition of unity, since $\sum_{i=1}^{n_\Gamma} \psi_i(\boldsymbol{x}) = 1$.
- The basis functions are strictly positive and bounded above by one: $0 < \psi_i(\boldsymbol{x}) < 1$.
- In the limit as $\rho \to \infty$, the basis function $\psi_i(\boldsymbol{x})$ tends toward the discontinuous function

$$\lim_{\rho\to\infty} \psi_i(\boldsymbol{x}) = \begin{cases} 1, & \text{if } \min_j \Delta_j(\boldsymbol{x}) = \Delta_i(\boldsymbol{x}), \\ 0, & \text{otherwise.} \end{cases}$$

We conclude this section by remarking on the close similarity between the proposed LSF and the partition of unity networks described in [19]. The primary distinction is that partition of unity networks are general-purpose surrogates whereas (5) can be viewed as a specialized surrogate that is tailored to the approximation of boundaries.

C. Quadratic approximation using curvature information

The approximation (5) is based on a planar representation of the surface. However, if we have local curvature information at each $\boldsymbol{x}_i \in \Gamma_h$, then we can introduce higher-order corrections that improve the approximation accuracy. In this section we describe such corrections for the two-dimensional case. The three-dimensional extension is straightforward and is summarized in the next section.

To find the curvature correction terms, we replace the planar approximation at $\boldsymbol{x}_i$ with one based on the osculating circle. We assume that the osculating circle has curvature κ_i at $\boldsymbol{x}_i$, or, equivalently, that it has a radius of curvature of $R_i = 1/\kappa_i$. The curvature here is the signed curvature, i.e. it can be negative. The sign convention we follow is that $\kappa_i > 0$ if the curve turns away from $\hat{\boldsymbol{n}}_i$ while moving in the positive sense along the curve's parameterization. Recall that the vector $\hat{\boldsymbol{n}}_i$ denotes outward-pointing normal to Γ at the point $\boldsymbol{x}_i$.

The coordinates of the center of the osculating circle corresponding to $\boldsymbol{x}_i \in \Gamma_h$ are given by $\boldsymbol{c}_i = \boldsymbol{x}_i - R_i\hat{\boldsymbol{n}}_i$. Note that, since $R_i = 1/\kappa_i$ is signed, the center of the circle can be on either side of Γ; however, for the moment, assume $R_i > 0$. Then the distance from an arbitrary point $\boldsymbol{x}$ to the osculating circle is given by

$$d_i(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{c}_i\| - R_i = \|\boldsymbol{x} - \boldsymbol{x}_i + R_i\hat{\boldsymbol{n}}_i\| - R_i.$$

Assuming the curvature is sufficiently small, the above expression simplifies as shown below. Note that, to distinguish between the linear and quadratric approximations to the signed distance function in the following, we use $\delta_{\perp,i}(\boldsymbol{x}) \equiv (\boldsymbol{x} - \boldsymbol{x}_i)^T\hat{\boldsymbol{n}}_i$ to denote the linear approximation. Furthermore, we drop the dependence of $\Delta_i(\boldsymbol{x})$ and $\delta_i(\boldsymbol{x})$ on $\boldsymbol{x}$ to simplify the presentation.

$$\begin{aligned} d_i(\boldsymbol{x}) &= R_i\|\kappa_i(\boldsymbol{x} - \boldsymbol{x}_i) + \hat{\boldsymbol{n}}_i\| - R_i \\ &= R_i\sqrt{\kappa_i^2\Delta_i^2 + 2\kappa_i\delta_{\perp,i} + 1} - R_i \\ &= R_i\left[1 + \frac{1}{2}\left(2\kappa_i\delta_{\perp,i} + \kappa_i^2\Delta_i^2\right) - \frac{1}{8}\left(2\kappa_i\delta_{\perp,i} + \kappa_i^2\Delta_i^2\right)^2 + \mathrm{O}(\kappa^3)\right] - R_i \\ &= \delta_{\perp,i} + \frac{\kappa_i}{2}(\Delta_i^2 - \delta_{\perp,i}^2) + \mathrm{O}(\kappa_i^2). \end{aligned}$$

Substituting $\Delta_i^2 = \|\boldsymbol{x} - \boldsymbol{x}_i\|^2$ and $\delta_{\perp,i} = (\boldsymbol{x} - \boldsymbol{x}_i)^T \hat{\boldsymbol{n}}_i$, we arrive at

$$d_i(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{x}_i)^T \hat{\boldsymbol{n}}_i + \frac{\kappa_i}{2}(\boldsymbol{x} - \boldsymbol{x}_i)^T \left(\mathrm{I} - \hat{\boldsymbol{n}}_i \hat{\boldsymbol{n}}_i^T\right)(\boldsymbol{x} - \boldsymbol{x}_i) + \mathrm{O}(\kappa_i^2). \tag{8}$$

Thus, the quadratic correction consists of the squared distance from $\boldsymbol{x}_i$ to $\boldsymbol{x}$ that is parallel to the planar approximation, scaled by half the curvature. A careful bookkeeping of signs shows that (8) also applies when $\kappa_i < 0$.

The corrected LSF uses $d_i(\boldsymbol{x})$ given by the first two terms on the right-hand side of (8) in place of the planar approximation in the expression (5). The results in Section III demonstrate that (8) produces an approximation error that converges to zero at a cubic rate in h for smooth Γ.

D. Approximating three-dimensional geometries

The level-set function for three-dimensional geometries has the same form as two-dimensional geometries, namely Equation (5) or (6). The partition-of-unity basis functions remain the same — see Equation 7 — with $\Delta_i(\boldsymbol{x})$ now the Cartesian distance in $\mathbb{R}^3$. Furthermore, the linear signed-distance functions also remain the same in three-dimensions, since the projection formula $(\boldsymbol{x} - \boldsymbol{x}_i)^T \hat{\boldsymbol{n}}_i$ is dimension independent. Thus, only the quadratic distance function needs to be generalized in the three-dimensional case.

For points near $\boldsymbol{x}_i$, the local (quadratic) distance function in three dimensions is

$$d_i(\boldsymbol{x}) \equiv (\boldsymbol{x} - \boldsymbol{x}_i)^T \hat{\boldsymbol{n}}_i + \frac{\kappa_{1,i}}{2}(\boldsymbol{x} - \boldsymbol{x}_i)^T \hat{\boldsymbol{t}}_{1,i} \hat{\boldsymbol{t}}_{1,i}^T (\boldsymbol{x} - \boldsymbol{x}_i) + \frac{\kappa_{2,i}}{2}(\boldsymbol{x} - \boldsymbol{x}_i)^T \hat{\boldsymbol{t}}_{2,i} \hat{\boldsymbol{t}}_{2,i}^T (\boldsymbol{x} - \boldsymbol{x}_i). \tag{9}$$

As in the two-dimensional case, $\hat{\boldsymbol{n}}_i$ is a unit normal at $\boldsymbol{x}_i$ that is oriented outward with respect to Ω. The (signed) scalars $\kappa_{1,i}$ and $\kappa_{2,i}$ denote the principal curvatures at $\boldsymbol{x}_i$, and the unit (tangent) vectors $\hat{\boldsymbol{t}}_{1,i}$ and $\hat{\boldsymbol{t}}_{2,i}$ are the corresponding principal directions. Together, the set $\{\hat{\boldsymbol{n}}_i, \hat{\boldsymbol{t}}_{1,i}, \hat{\boldsymbol{t}}_{2,i}\}$ constitutes an orthonormal basis, or frame, for the surface at $\boldsymbol{x}_i$.

The method of computing the normal, tangents, and principal curvatures will depend on the underlying geometry we have access to. If we have an explicit parameterization (e.g. from a CAD model), then we can query the model to find the necessary data at each $\boldsymbol{x}_i$. If we only have a point cloud, then the data can be reconstructed using methods from the computer graphics and CAD literature; see, for example, [20].

III. Numerical Experiments and Results

This section includes several numerical experiments to help quantify the accuracy of the LSF (5). The experiments are also used to explore, qualitatively, the role of the parameter ρ and the number of surface nodes, n_Γ. However, before presenting the results, we begin with some details regarding our particular implementation.

A. Implementation

The partition-of-unity LSF was implemented in Julia [21] as the package `LevelSets.jl`[†]. We highlight three important changes to the LSF expression (5) as implemented in the package.

The Euclidean distance is replaced with the following regularized version:

$$\Delta_i(\boldsymbol{x}) \equiv \sqrt{(\boldsymbol{x} - \boldsymbol{x}_i)^T (\boldsymbol{x} - \boldsymbol{x}_i) + \epsilon},$$

where $\epsilon = 10^{-10}$. This change was made to avoid numerical issues when computing the gradient of $\phi(\boldsymbol{x})$ when $\|\boldsymbol{x} - \boldsymbol{x}_i\|$ is small; the gradient of $\|\boldsymbol{x} - \boldsymbol{x}_i\|$ is not defined at $\boldsymbol{x}_i$, whereas the gradient of $\Delta_i(\boldsymbol{x})$, as defined above, is well defined for all $\boldsymbol{x} \in \mathbb{R}^3$. Note that the value of ϵ is based on the geometries considered herein, and, more generally, this parameter should be scaled appropriately depending on the typical length scale of the geometry.

The exponential terms in the partition-of-unity basis functions can also lead to numerical issues, particularly when $\rho \geq 10$. For instance, for $\boldsymbol{x}$ sufficiently far from the surface points in Γ_h, the denominator in (5) can become zero to working precision, leading to division by zero. The solution, which is standard in similar constraint aggregation functions, is to scale the top and bottom of (5) by $e^{\rho\Delta_{\min}}$, where $\Delta_{\min} = \min_{i=1,2,\ldots,n_\Gamma} \Delta_i(\boldsymbol{x})$:

$$\phi(\boldsymbol{x}) = \frac{e^{\rho\Delta_{\min}} \sum_{i=1}^{n_\Gamma} d_i(\boldsymbol{x}) e^{-\rho\Delta_i(\boldsymbol{x})}}{e^{\rho\Delta_{\min}} \sum_{i=1}^{n_\Gamma} e^{-\rho\Delta_i(\boldsymbol{x})}} = \frac{\sum_{i=1}^{n_\Gamma} d_i(\boldsymbol{x}) e^{-\rho(\Delta_i(\boldsymbol{x}) - \Delta_{\min})}}{\sum_{i=1}^{n_\Gamma} e^{-\rho(\Delta_i(\boldsymbol{x}) - \Delta_{\min})}}.$$

[†] https://github.com/jehicken/LevelSets.jl

For the term i corresponding to $\Delta_i(\boldsymbol{x}) = \Delta_{\min}$ in the sums on the right, we have $e^{-\rho(\Delta_i(\boldsymbol{x})-\Delta_{\min})} = 1$, so there is no risk of division by zero.

The last implementation detail worth highlighting is our use of a k-d tree to accelerate the evaluation of (5). The cost of computing the partition-of-unity LSF can become intolerable for large numbers of surface points ($n_\Gamma \geq 1000$), which are often necessary for three-dimensional geometries. Unless otherwise stated, we limit the sums in (5) to the ten closest points as determined by a standard k-d tree implementation [22]. The choice of ten works well in practice based on our experience, but a more systematic analysis should be undertaken.

B. Investigation into the role of ρ and n_Γ

For the first set of experiments, we are interested in understanding how the parameter ρ and number of elements in Γ_h impact the accuracy of the proposed LSF. To this end, we use the LSF (5) to approximate an ellipse with a semi-major axis of four units and a semi-minor axis of one unit. While we are ultimately concerned with approximating complex engineering geometries, these geometries do not admit an analytical level-set representation, in general, whereas the ellipse does.

Figure 3 displays several LSF approximations to the ellipse using different parameters. The rows of sub-figures correspond to increasing numbers of surface approximation points, specifically $n_\Gamma \in \{8, 16, 32, 64\}$. The two columns correspond to two different values of the parameter ρ in terms of the ratio ρ/n_Γ.

We study the ratio ρ/n_Γ, rather than ρ itself, because the parameter ρ should be proportional to the reciprocal of the spacing of points in Γ_h, in order to ensure convergence of the level-set $\phi(\boldsymbol{x}) = 0$ to Γ; this is evident in the convergence-study results presented in the next section. Thus, we need $\rho \propto 1/h$ and, since $1/h \propto n_\Gamma$ for the two-dimensional domains considered here, the ratio ρ/n_Γ should be a constant for convergence. For this reason, Figure 3 considers two values for the ratio ρ/n_Γ, namely one and ten.

The relationship between the LSF (5) and the piecewise linear LSF (2) is most obvious from the first row in Figure 3. In particular, for $\rho/n_\Gamma = 10$ (second column), we see a close correspondence with the contours in Figure 2.

With only $n_\Gamma = 8$ points in Γ_h, it is clear that the zero contour of (5) is a poor approximation to the ellipse. However, the approximation converges well as n_Γ increases; again, this convergence is quantified in the next section. We also note that the differences between $\rho/n_\Gamma = 1$ and $\rho/n_\Gamma = 10$ becomes less (visually) obvious as the number of the surface points increases.

The results in Figure 3 are based on the planar approximation to the surface. By contrast, Figure 4 presents results for the LSF (5) using the quadratic correction in (8). Compared with the linear/planar approximation using the same n_Γ, the quadratic approximation is clearly superior. Indeed, the contour $\phi(\boldsymbol{x}) = 0$ based on the quadratic correction with $n_\Gamma = 8$ is similar in appearance to the linear approximations using four to eight times as many points.

C. Rate of convergence for smooth boundaries

We conducted a convergence study to quantify the accuracy of the LSF (5). The target geometry was, again, an ellipse with semi-major axis of four units and semi-minor axis of one unit. To quantify the error in the zero contour of $\phi(\boldsymbol{x})$, we sampled $n_{\text{samp}} = 1024$ points along the contour of the ellipse and computed the root-mean-squared (RMS) error:

$$\text{RMS Error}(n_\Gamma, \rho) = \sqrt{\frac{\sum_{j=1}^{n_{\text{samp}}} \left|\phi(\boldsymbol{x}_j)\right|^2}{n_{\text{samp}}}}, \tag{10}$$

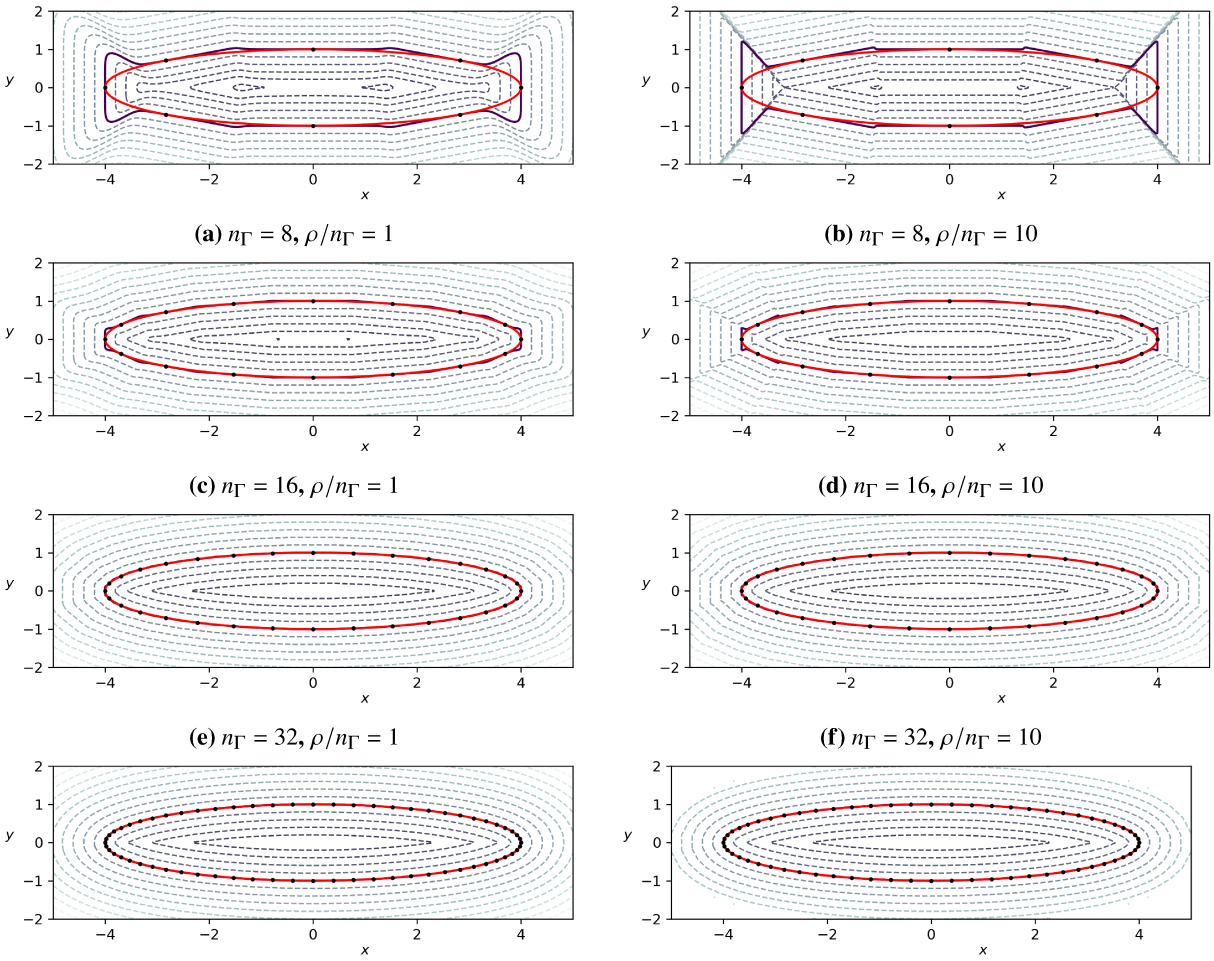
where the sample points are given by

$$\boldsymbol{x}_j = \left[4\cos(\theta_j), \sin(\theta_j)\right]^T, \qquad \theta_j = 2\pi(j-1)/n_{\text{samp}}, \quad j \in \{1, 2, \ldots, n_{\text{samp}}\}.$$

The unit normals required by (5), namely $\{\hat{\boldsymbol{n}}_i\}_{i=1}^{n_\Gamma}$, were computed analytically. For the quadratic correction, the curvature was also computed analytically.

The convergence study considered $n_\Gamma \in \{10, 20, 40, 80, 160\}$ to ensure the asymptotic convergence behavior would be clear. Note that the largest number of surface points, $n_\Gamma = 160$, is still 6.4 times smaller than the number of sample points used to define the RMS error. We also chose $\rho/n_\Gamma \in \{0.1, 1, 10\}$ to explore the impact of the parameter ρ, with values spanning two orders of magnitude.

Figure 5 plots the RMS error as a function of n_Γ for both the linear and quadratic approximations. We make the following conclusions regarding the accuracy of the LSF (5) in the context of the ellipse.



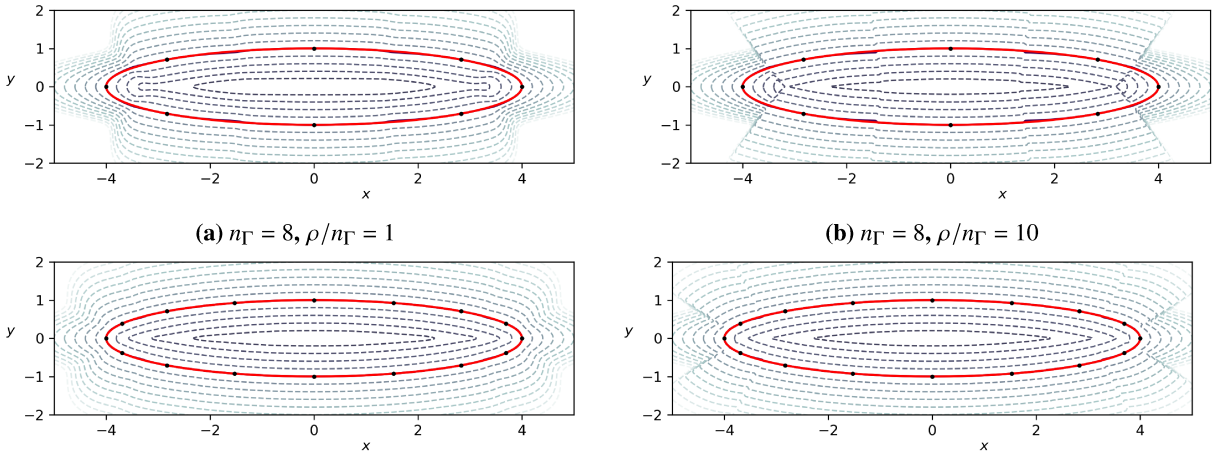
(a) $n_\Gamma = 8$, $\rho/n_\Gamma = 1$ **(b)** $n_\Gamma = 8$, $\rho/n_\Gamma = 10$

(c) $n_\Gamma = 16$, $\rho/n_\Gamma = 1$ **(d)** $n_\Gamma = 16$, $\rho/n_\Gamma = 10$

(e) $n_\Gamma = 32$, $\rho/n_\Gamma = 1$ **(f)** $n_\Gamma = 32$, $\rho/n_\Gamma = 10$

(g) $n_\Gamma = 64$, $\rho/n_\Gamma = 1$ **(h)** $n_\Gamma = 64$, $\rho/n_\Gamma = 10$

Fig. 3 Parameter study of the LSF (5) applied to an ellipse. The curve Γ is shown in red, and the zero-curve $\phi(\boldsymbol{x}) = 0$ is shown in black. The black dots correspond to the points in Γ_h, and additional contours are shown as dashed lines.



(a) $n_\Gamma = 8$, $\rho/n_\Gamma = 1$ **(b)** $n_\Gamma = 8$, $\rho/n_\Gamma = 10$

(c) $n_\Gamma = 16$, $\rho/n_\Gamma = 1$ **(d)** $n_\Gamma = 16$, $\rho/n_\Gamma = 10$

Fig. 4 Parameter study of the LSF (5) applied to an ellipse *using the quadratic correction* (8). The curve Γ is shown in red, and the zero-curve $\phi(\boldsymbol{x}) = 0$ is shown in black. The black dots correspond to the points in Γ_h, and additional contours are shown as dashed lines.

- Asymptotically, as $n_\Gamma \to \infty$, the linear approximation converges at a rate of $1/n_\Gamma^2 \propto h^2$, and the quadratic approximation converges at a rate of $1/n_\Gamma^3 \propto h^3$. These convergence rates are consistent with approximation theory based on using piecewise polynomials to approximate the surface.
- $\rho/n_\Gamma = 0.1$ is highly inaccurate; the error using this ratio is one order of magnitude larger compared to the other ratios for the linear approximation, and the error is almost three orders of magnitude larger for the quadratic approximation using the other ratios.
- Increasing ρ/n_Γ above one does not improve the accuracy appreciably. In fact, the error increases slightly going from $\rho/n_\Gamma = 1$ to $\rho/n_\Gamma = 10$ when using the quadratic $d_i(\boldsymbol{x})$.
- The RMS errors are based on using all n_Γ points when evaluating (5) rather than the ten closest points from the k-d tree. The results using the ten closest points (not shown) are essentially identical for $\rho/n_\Gamma = 1$ and $\rho/n_\Gamma = 10$. For $\rho/n_\Gamma = 0.1$, using the ten closest points in (5) produces a slightly smaller error, but, given the other observations, such a small value of ρ is unlikely to be used in practice.

Based on our experience with other geometries, the statements above seem to apply more generally, i.e., to other smooth boundaries. The case of non-smooth geometries is studied in the next section.

D. Rate of convergence for non-smooth boundaries

In practice, aircraft and spacecraft have non-smooth geometries consisting of corners, edges, and points; consider, for instance, the trailing edge of an airfoil. Consequently, it is important to study the effectiveness of the LSF (5) in approximating such features.

Figure 6 plots the contours of $\phi(\boldsymbol{x})$ for (5) based on surface points and normals from a right triangle. The triangle has a base of one unit and a height of 0.5 units. We use the same number of points to approximate each side of the triangle; that is, we use $n_\Gamma/3$ points on the bottom, side, and hypotenuse. We assume zero curvature, $\kappa_i = 0$, for all points $i \in \{1, 2, \ldots, n_\Gamma\}$, so the quadratic approximation is equivalent to the linear approximation and, therefore, omitted from the results.

As with the ellipse, the contour $\phi(\boldsymbol{x}) = 0$ converges to Γ as n_Γ increases. Unlike the ellipse, we observe significant errors at the corners of the triangle. These errors arise for a couple reasons.

- Let $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ denote the two points in Γ_h that are closest to a particular vertex. Since one of these two points will be closer to the vertex, in general, the line that is an equidistance from $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ does not pass through the vertex. Consequently, the less accurate linear approximation is used in some regions, even as $\rho \to \infty$.
- Toward a corner, points on one side of the corner begin to influence the LSF on the other side of the corner, even though their corresponding normal vectors do not agree.

The effect of the vertex errors is reflected in the convergence plot shown in Figure 7. The plot includes both the RMS

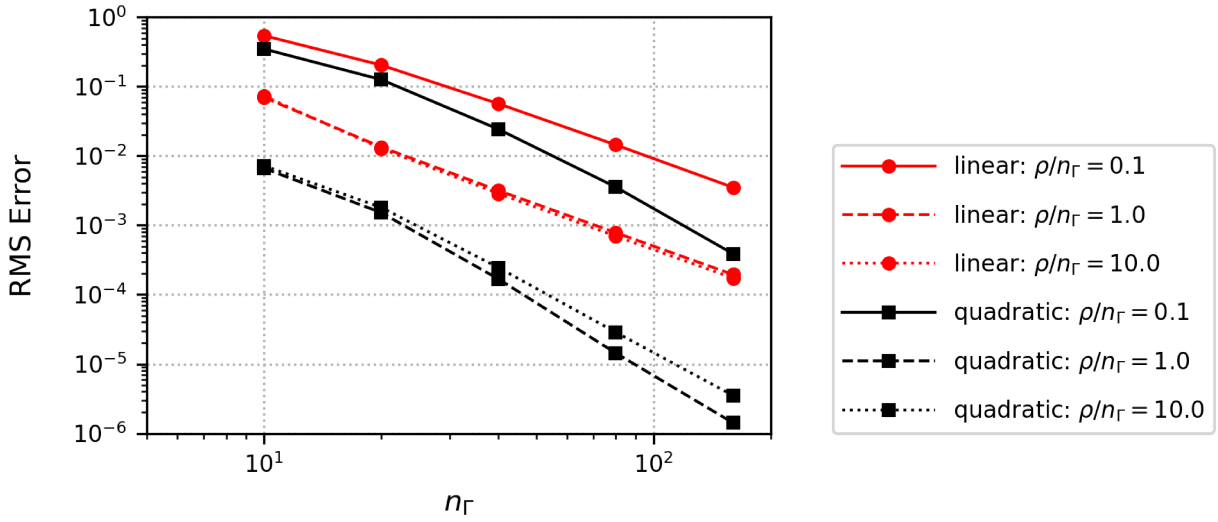
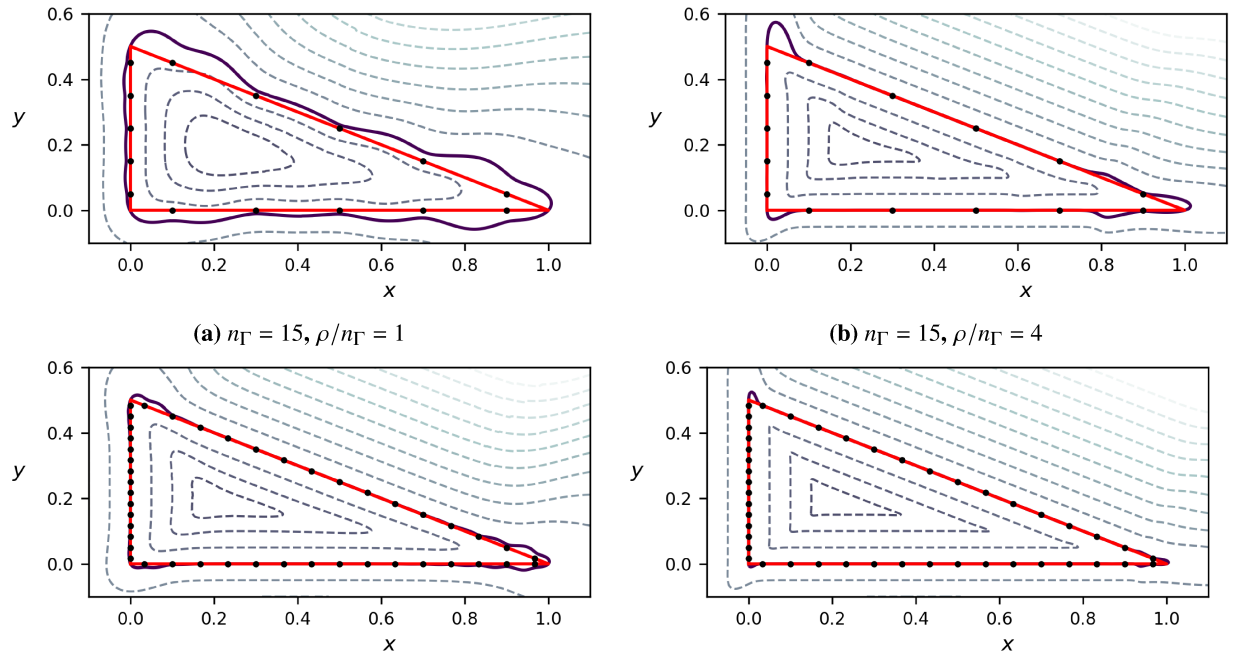


Fig. 5 RMS error as a function of n_Γ for the LSF (5) applied to an ellipse with semi-major axis 4 and semi-minor axis 1.



(a) $n_\Gamma = 15$, $\rho/n_\Gamma = 1$

(b) $n_\Gamma = 15$, $\rho/n_\Gamma = 4$

(c) $n_\Gamma = 45$, $\rho/n_\Gamma = 1$

(d) $n_\Gamma = 45$, $\rho/n_\Gamma = 4$

Fig. 6 Parameter study of the LSF (5) applied to a right-angle triangle. The curve Γ is shown in red, and the zero-curve $\phi(\boldsymbol{x}) = 0$ is shown in black; additional contours are shown as dashed lines

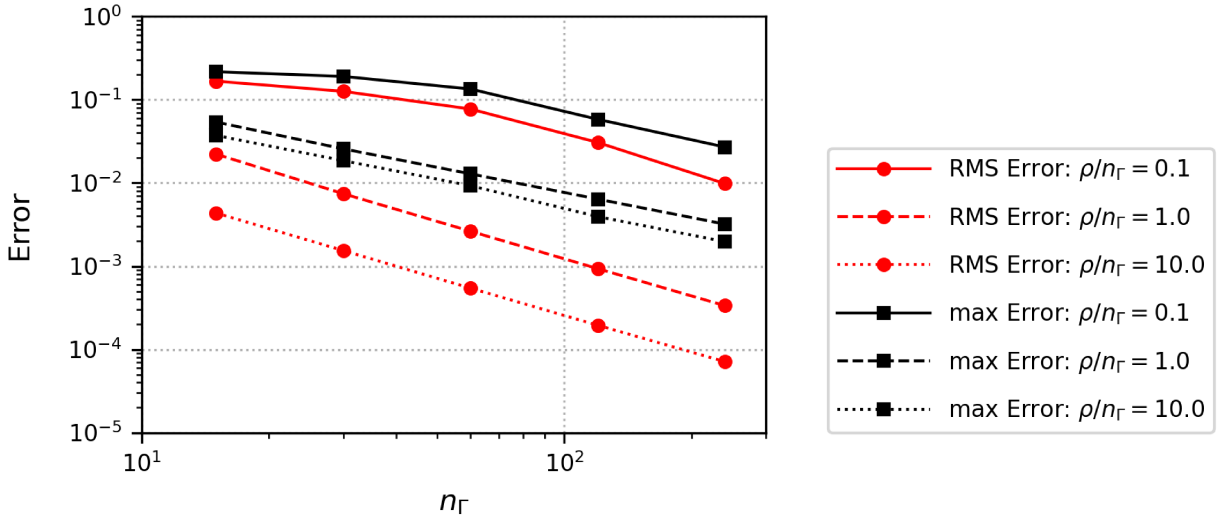


Fig. 7 RMS and max **error as a function of n_Γ for the LSF (5) applied to a right triangle with length of 1 and height of 0.5 units.**

error and the max error for $n_\Gamma \in \{15, 30, 60, 120, 240\}$. Furthermore, we again consider the ratios $\rho/n_\Gamma \in \{0.1, 1, 10\}$. The RMS error is computed based on $n_{\text{samp}} = 1500$ sample points — 500 points per side. The max error is based on the same 1500 sample points and is defined by

$$\max \ \text{Error}(n_\Gamma, \rho) = \max_{j=1,2,\ldots,n_{\text{samp}}} |\phi(\boldsymbol{x}_j)|.$$

The max error converges at a first-order rate proportional to $1/n_\Gamma \propto h$, which reflects the error committed at the vertices of the triangle. Since this $O(h)$ error occurs at a finite number of points in this case — the three vertices to be precise — the RMS error is reduced to $O(h^{3/2})$.

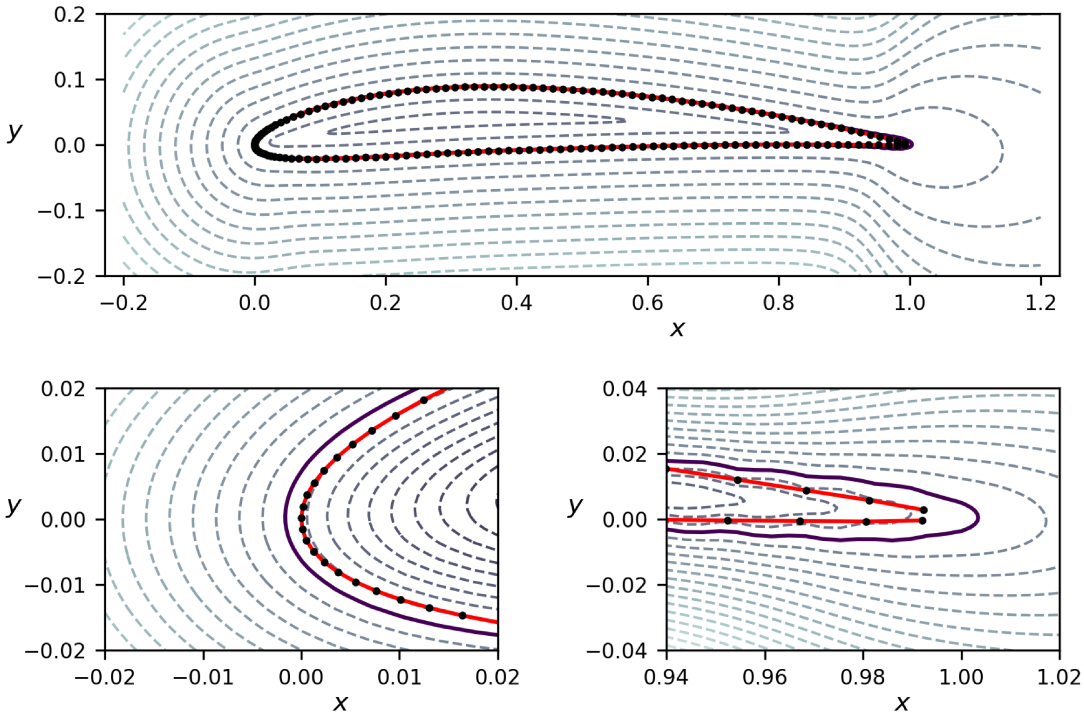
Finally, the effect of the ratio ρ/n_Γ is significantly more pronounced for the triangle RMS errors compared to the ellipse RMS errors. In particular, increasing from $\rho/n_\Gamma = 1$ to $\rho/n_\Gamma = 10$ reduces the RMS error by almost an order of magnitude. On the other hand, increasing the ratio has diminishing returns for the max error. This suggests that increasing the ratio has the effect of restricting the errors to a smaller fraction of points near the vertex, which is consistent with the contour plots in Figure 6.

E. Application to an airfoil

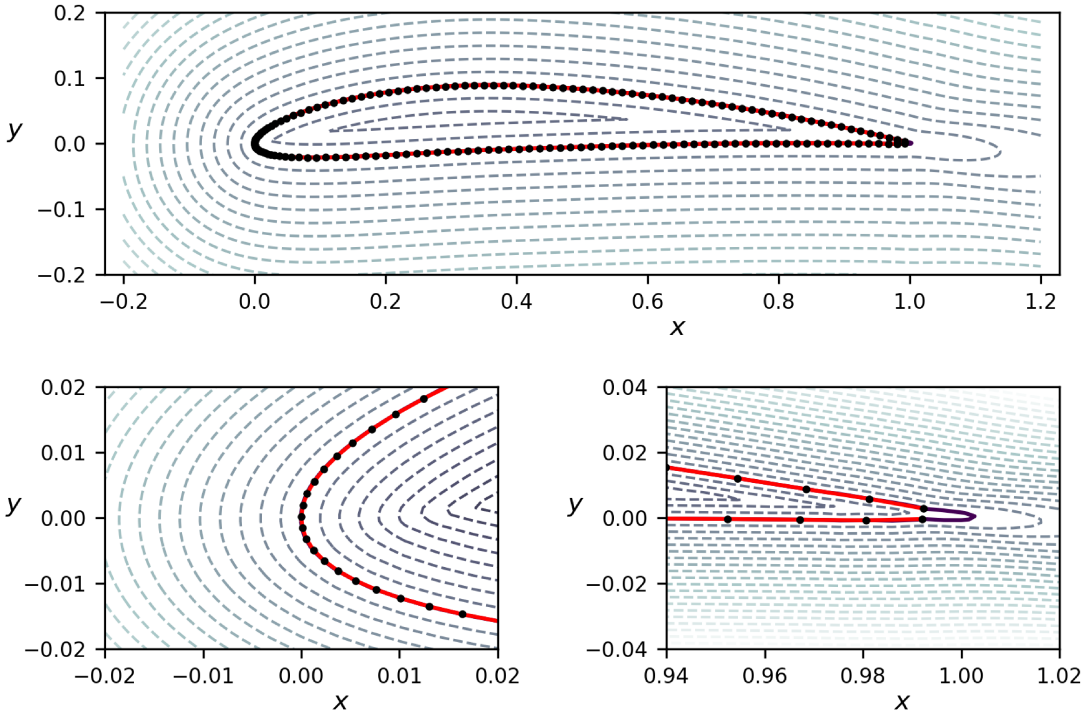
Having investigated and quantified the accuracy of the partition-of-unity LSF (5) on simple shapes, the remaining results focus on more complex or relevant geometries. In this section we consider the LSF applied to the NACA 4410 airfoil.

The surface points, $\Gamma_h = \{\boldsymbol{x}_i\}_{i=1}^{158}$, for the NACA 4410 were obtained from an airfoil database. The two points at the trailing edge were removed from the raw data set, since a more accurate level-set was obtained by not including these points. The normal vectors were estimated using second-order central difference approximations over the interior points, and forward- and backward-difference formulae at the first and last points. We investigated the quadratic correction using finite-difference approximations to the curvature, but did not find a substantial improvement in accuracy to warrant including the correction in the following results.

Figure 8 shows the airfoil contours of (5) using $\rho/n_\Gamma = 1$ and $\rho/n_\Gamma = 10$. We include contours of the overall airfoil, as well as details near the leading and trailing edges. While the $\rho/n_\Gamma = 1$ contours appear accurate when viewing the entire airfoil, the close-up views reveal that the contours are offset near the leading and trailing edge. When the ratio is increased to ten, we observe considerable improvement in accuracy near the leading and trailing edges.



(a) $n_\Gamma = 158$, $\rho/n_\Gamma = 1$



(b) $n_\Gamma = 158$, $\rho/n_\Gamma = 10$

Fig. 8 The LSF (5) applied to the NACA 4410 airfoil. The curve Γ is shown in red, and the zero-curve $\phi(x) = 0$ is shown in black; additional contours are shown as dashed lines

F. Three-dimensional analytical geometries

Next, we apply the partition-of-unity LSF to three-dimensional surfaces with simple parametric equations. As with the ellipse in two-dimensions, these analytical geometries give us more control over where we sample, as well as providing explicit curvature information, if needed.

The first surface we consider is an ellipsoid with parametric representation

$$\boldsymbol{x}(u,v) = \begin{bmatrix} a\cos(u)\sin(v) & b\sin(u)\sin(v) & -c\cos(v) \end{bmatrix}^T,$$

where $(u,v) \in \{0 \le u \le 2\pi, 0 \le v \le \pi\}$, and the axes lengths are $a = 2$, $b = 1/2$, and $c = 1$. The LSF points in Γ_h are generated by uniformly sampling the parameters u and v:

$$\boldsymbol{x}_i = \boldsymbol{x}(u_j, v_k), \qquad \text{where} \qquad u_j = 2\pi(j-0.5)/n, \quad v_k = \pi(k-0.5)/n, \quad \text{and} \quad i(j,k) = n(j-1)+k. \tag{11}$$

Note that we deliberately avoid the ends of the v range, since these would produce non-unique samples. The total number of samples used in the LSF is given by $n_\Gamma = n^2$. The hyperparameter $\rho = 10n$, which makes it proportional to the reciprocal of the mesh spacing in (u,v) space.

The normal, tangents, and curvatures are evaluated using standard formulae from differential geometry; see, for example, [23, Chap. 3]. In particular, the tangents and curvatures for (9) are found by computing the first- and second-partial derivatives of $\boldsymbol{x}(u,v)$ with respect to u and v, and then using the first- and second-fundamental forms to find the principal curvatures and directions.

In order to visualize the approximate level set, we sample the parametric representation at a higher resolution and "project" the exact surface points to the level set. In all cases we use $1200 \times 1200 = 1.44 \times 10^6$ points, uniformly sampled in (u,v) space to visualize the shapes. The projection is obtained by applying Newton's method to $\phi(\boldsymbol{x}) = 0$ with the exact surface point used as the initial estimate. This approach is robust, even without line searches or other safe-guarding mechanisms. On the other hand, the resulting visualization points are not necessarily the closest points on $\phi(\boldsymbol{x}) = 0$ to the initial estimates.

Figure 9 visualizes the ellipsoid LSF for $n_\Gamma = 20 \times 20 = 400$ and $n_\Gamma = 40 \times 40 = 1600$ points in Γ_h. Results are shown for both linear and quadratic versions of $d_i(\boldsymbol{x})$. Ridges between the points $\boldsymbol{x}_i$ are evident in the coarsest Γ_h, shown in Figures 9a and 9b, and take on Voronoi-like patterns.

The plots in Figure 9 demonstrate the value of including curvature in three-dimensional LSF approximations. Quantitative evidence for this is provided in Figure 10, which plots the RMS error in the LSF functions for a range of $\sqrt{n_\Gamma} = n$ values and two ρ/n ratios. The RMS error is computed using Equation (10), with $n_{\text{samp}} = 1.44 \times 10^6$; the RMS points are defined using (11), but with $n = 1200$ to obtain a finer resolution. While we use the same number of RMS sample points as visualization samples, we emphasize that these points are different, since the former are not "snapped" onto the LSF.

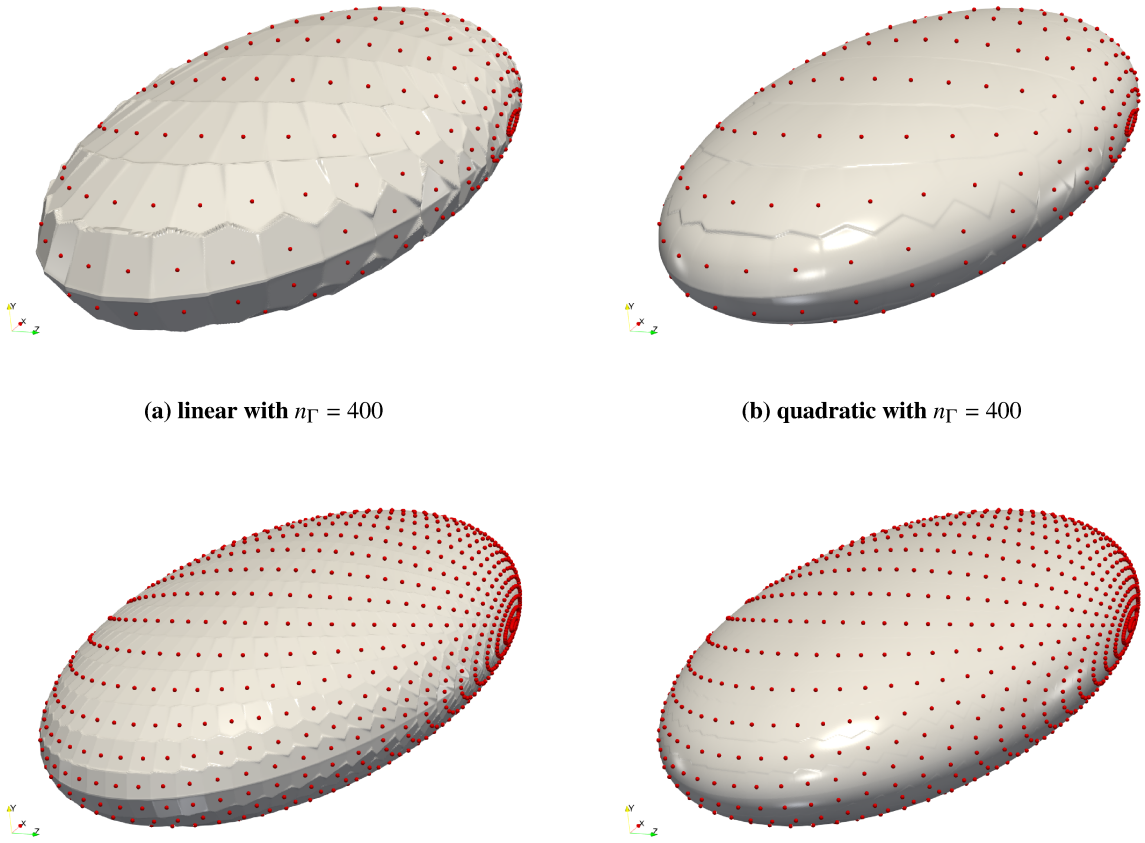
Figure 10 shows that the error in the ellipsoid LSF behaves much as it did for the ellipse. In particular, the error converges at second- and third-order for the linear and quadratic distance functions, respectively.

In addition to the ellipsoid, we also exercised the LSF approximation on the following geometry, which exhibits a larger range of curvature values:

$$\boldsymbol{x}(u,v) = \begin{bmatrix} \cos(v)\,[6 - (5/4 + \sin(3u))\sin(u-3v)] \\ \sin(v)\,[6 - (5/4 + \sin(3u))\sin(u-3v)] \\ -\cos(u-3v)\,[5/4 + \sin(3u)] \end{bmatrix},$$

where $(u,v) \in \{0 \le u \le 2\pi, 0 \le v \le 2\pi\}$. We will refer to this geometry as the "wreath." The points Γ_h for the wreath were computed using a formula similar to (11), but with the range for v_k updated appropriately. As with the ellipsoid, we used $\rho = 10n$. We also used the same method for visualization.

Two LSF approximations of the wreath are shown in Figure 11. We considered only the quadratic version of the local distance function, $d_i(\boldsymbol{x})$, for this shape, since the linear approximation was poor for the values of n_Γ considered ($n_\Gamma = 1600$ and $n_\Gamma = 6400$). The high curvature is largely responsible for the poor performance of the linear approximation. Using $n_\Gamma = 6400$ samples for Γ_h, the wreath's principal curvatures vary from 1.36×10^{-4} to 1.03×10^2 in absolute value. By contrast, the principal curvatures of the ellipsoid range from 1.25×10^{-1} to 1.60×10^1 only, for the same $n_\Gamma = 6400$.



(a) linear with $n_\Gamma = 400$

(b) quadratic with $n_\Gamma = 400$

(c) linear with $n_\Gamma = 1600$

(d) quadratic with $n_\Gamma = 1600$

Fig. 9 LSF applied to an ellipsoid with axes lengths $a = 2$, $b = 1/2$, and $c = 1$ in the x-, y-, and z-coordinate directions, respectively.

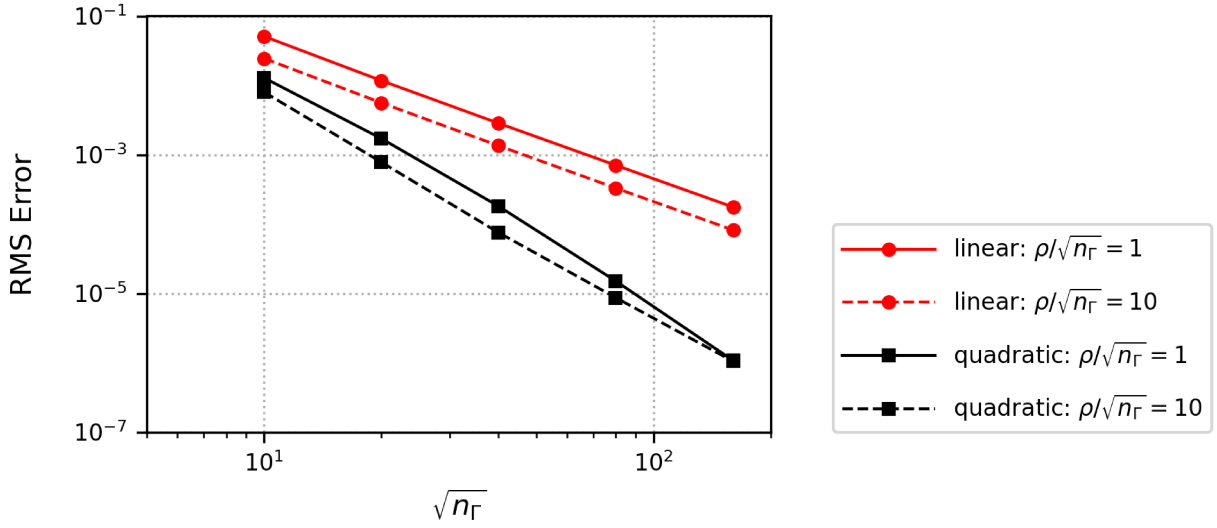
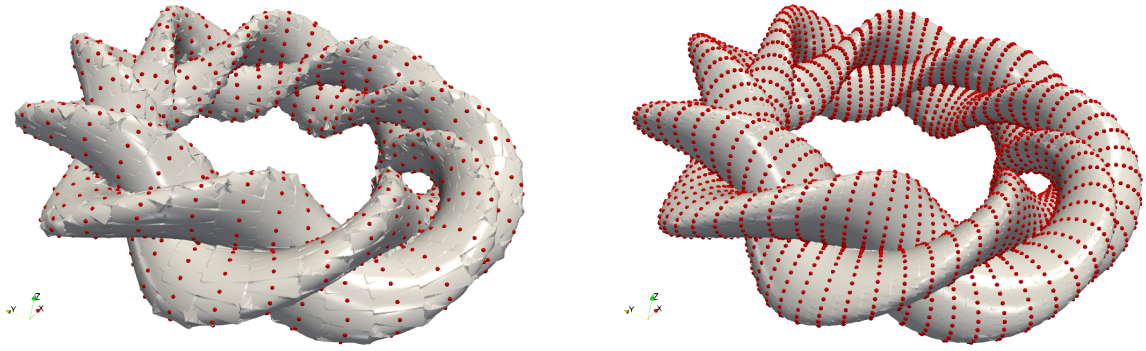


Fig. 10 RMS error as a function of $\sqrt{n_\Gamma}$ for the LSF (5) applied to the ellipsoid.



(a) $n_\Gamma = 1600$ **(b)** $n_\Gamma = 6400$

Fig. 11 LSF applied to a parametric surface with a wide range of curvature values. The quadratic correction is used in both figures.

G. Application to the space shuttle

For our final example, we used the partition-of-unity LSF to approximate the outer mold line of the space shuttle. Since we did not have immediate access to a parametric CAD model of the space shuttle, we used an STL file available from the `grabcad.com` website[‡]. The STL format limits this example to linear distance functions $d_i(\boldsymbol{x})$, but using the STL format illustrates the flexibility of the proposed LSF method.

The initial STL file consists of 345, 396 vertices and 115, 132 triangles. We sample each triangle six times to generate Γ_h; three samples are associated with the triangle's vertices and three are associated with the midpoints of the triangle's edges. Thus, we have $n_\Gamma = 6 \times 115, 132$ total points in Γ_h. The precise locations of the samples are given by

$$\boldsymbol{x} = (1 - 2\alpha)\boldsymbol{x}_{v1} + \alpha\boldsymbol{x}_{v2} + \alpha\boldsymbol{x}_{v3},$$

where $\boldsymbol{x}_{v1}$, $\boldsymbol{x}_{v2}$, and $\boldsymbol{x}_{v3}$ are the three vertices of a given STL triangle. The barycentric coordinate α takes on the values 0.05 and 0.475 to generate two points. The remaining four points are found by permuting the indices $v1$, $v2$ and $v3$. The normal vector, $\hat{\boldsymbol{n}}_i$, at each sample is based on the STL triangle that the sample is derived from.

The points in Γ_h are visualized as red dots in Figures 12a and 12c. These figures show that there is a wide range of sample densities. For instance, the shuttle bay doors, which curve in only one direction, have very few samples. By contrast, the nozzles and nose of the shuttle have significantly more samples. To account for this range in sample densities, we used $\rho = \sqrt{n_\Gamma} \propto 1/h$.

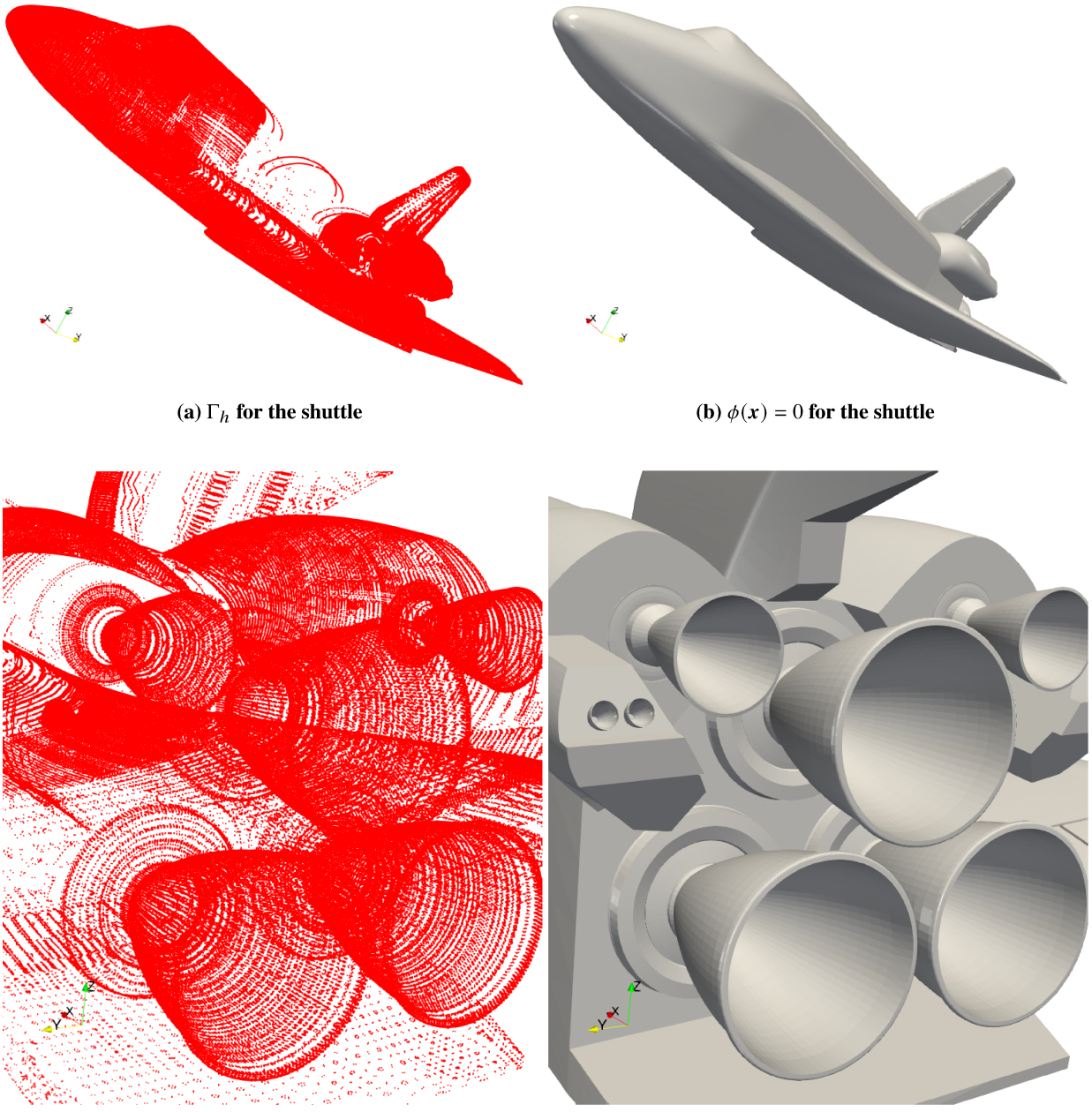
To visualize $\phi(\boldsymbol{x}) = 0$, we sampled the STL triangles at the vertices and midpoints — $\alpha = 0$ and $\alpha = 0.5$ in the above formula — and then used Newton's method to "snap" onto the level-set as we did for the ellipsoid and wreath examples. The resulting visualization points are formed into triangles and are plotted in Figures 12b and 12d. These figures suggest that the proposed LSF can leverage the differing sample densities to efficiently represent the shuttle geometry.

IV. Conclusions

We have proposed an explicit formula that can be used to approximate geometries as a level set. The formula uses localized approximations to the surface — either planar or quadratic — and the appropriate local approximation is selected using a partition-of-unity basis that is adapted from the smooth-minimum function.

The proposed LSF is flexible, in the sense that the method uses only point data, such as the outward pointing normal and local curvature, and does not require connectivity, i.e. a mesh. Consequently, both parameterized geometries (i.e. CAD) and point clouds can be used to generate the LSF. In the case of point clouds, a means of estimating the surface normal and, possibly, principal curvatures/directions is necessary, but such methods are available in the computer graphics literature.

[‡]https://grabcad.com/library/space-shuttle–1



(a) Γ_h **for the shuttle**

(b) $\phi(\boldsymbol{x}) = 0$ **for the shuttle**

(c) Γ_h **near the shuttle aft**

(d) $\phi(\boldsymbol{x}) = 0$ **near the shuttle aft**

Fig. 12 The LSF (5) used to approximate the shuttle; left figures show points from Γ_h in red, and right figures show $\phi(\boldsymbol{x}) = 0$.

Numerical experiments demonstrated that the LSF converges at second- and third-order rates in the point spacing for smooth geometries. For geometries with corners, the approximation converges at a three-halves rate in the RMS error and a first-order rate in the max error.

We have not conducted a formal computational complexity analysis, nor have we presented CPU times for the examples. Nevertheless, we do not anticipate practical limitations due to computational cost, provided the number of points in the formula is reduced to the $O(10)$ closest points and these points are found using an efficient search algorithm, e.g., a k-d tree.

There are several directions for future work. We are currently using the proposed LSF together with Saye's algorithm [14] to discretize a cut-cell finite-element method. To this end, we are developing bounds for the LSF and its gradient that are needed by the algorithm in [14]. Another direction for future work is to extend the method beyond quadratic. Given the efficiency of the quadratic $d_i(\boldsymbol{x})$ relative to the linear $d_i(\boldsymbol{x})$, we expect that higher-order approximations to the local distance field will offer additional efficiency gains. Finally, we plan to investigate how the LSF can be used in the context of aerodynamic shape optimization.

Acknowledgments

S. Kaur was supported by, and J. Hicken was partially supported by, the National Science Foundation under Grant No. 1825991. The authors gratefully acknowledge this support.

The authors sincerely thank Tucker Babcock, Garo Bedonian, and Luiz Cagliari for their comments and feedback during the preparation of this paper.

The bulk of the results in this paper were obtained using software written in Julia [21]. The two dimensional plots were generated using Matplotlib [24], and the three dimensional plots were created using Paraview [25]. The `WriteVTK.jl` Julia package [26] was used to generate the VTK files needed by Paraview.

References

[1] Osher, S., and Sethian, J. A., "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations," *Journal of Computational Physics*, Vol. 79, No. 1, 1988, pp. 12–49. doi:10.1016/0021-9991(88)90002-2.

[2] Sussman, M., Smereka, P., and Osher, S., "A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow," *Journal of Computational Physics*, Vol. 114, No. 1, 1994, pp. 146–159. doi:10.1006/jcph.1994.1155.

[3] Malladi, R., Sethian, J., and Vemuri, B., "Shape modeling with front propagation: a level set approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 2, 1995, pp. 158–175. doi:10.1109/34.368173.

[4] Haber, R., and Bendsoe, M., "Problem formulation, solution procedures and geometric modeling-Key issues in variable-topology optimization," *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, 1998. doi:10.2514/6.1998-4948.

[5] Sethian, J., and Wiegmann, A., "Structural Boundary Design via Level Set and Immersed Interface Methods," *Journal of Computational Physics*, Vol. 163, No. 2, 2000, pp. 489–528. doi:10.1006/jcph.2000.6581.

[6] De Ruiter, M., and Van Keulen, F., "Topology optimization using a topology description function," *Structural and Multidisciplinary Optimization*, Vol. 26, No. 6, 2004, pp. 406–416. doi:10.1007/s00158-003-0375-7.

[7] van Dijk, N. P., Maute, K., Langelaar, M., and van Keulen, F., "Level-set methods for structural topology optimization: a review," Vol. 48, No. 3, 2013, pp. 437–472. doi:10.1007/s00158-013-0912-y.

[8] Berger, M., and Leveque, R., "An adaptive Cartesian mesh algorithm for the Euler equations in arbitrary geometries," *9th Computational Fluid Dynamics Conference*, American Institute of Aeronautics and Astronautics, 1989. doi:10.2514/6.1989-1930.

[9] Aftosmis, M. J., "Lecture notes for the 28th computational fluid dynamics lecture series: solution adaptive Cartesian grid methods for aerodynamic flows with complex geometries," Tech. rep., von Kármán Institute for Fluid Dynamics, Rhode-Saint-Genèse, Belgium, Mar. 1997.

[10] Aftosmis, M. J., Berger, M. J., and Melton, J. E., "Robust and efficient Cartesian mesh generation for component-based geometry," *AIAA journal*, Vol. 36, No. 6, 1998, pp. 952–960.

[11] Fidkowski, K. J., and Darmofal, D. L., "A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations," *Journal of Computational Physics*, Vol. 225, 2007, pp. 1653–1672.

[12] Lew, A. J., and Buscaglia, G. C., "A discontinuous-Galerkin-based immersed boundary method," *International Journal for Numerical Methods in Engineering*, Vol. 76, No. 4, 2008, pp. 427–454. doi:10.1002/nme.2312.

[13] Burman, E., Claus, S., Hansbo, P., Larson, M. G., and Massing, A., "CutFEM: Discretizing geometry and partial differential equations," *International Journal for Numerical Methods in Engineering*, Vol. 104, No. 7, 2015, pp. 472–501. doi: 10.1002/nme.4823.

[14] Saye, R. I., "High-Order Quadrature Methods for Implicitly Defined Surfaces and Volumes in Hyperrectangles," *SIAM Journal on Scientific Computing*, Vol. 37, No. 2, 2015, pp. A993–A1019. doi:10.1137/140966290.

[15] Carr, J. C., Beatson, R. K., Cherrie, J. B., Mitchell, T. J., Fright, W. R., McCallum, B. C., and Evans, T. R., "Reconstruction and representation of 3D objects with radial basis functions," *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, pp. 67–76.

[16] Dinh, H. Q., Turk, G., and Slabaugh, G., "Reconstructing surfaces using anisotropic basis functions," *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, Vol. 2, IEEE, 2001, pp. 606–613.

[17] Belyaev, A., Fayolle, P.-A., and Pasko, A., "Signed Lp-distance fields," *Computer-Aided Design*, Vol. 45, No. 2, 2013, pp. 523–528.

[18] Kennedy, G. J., and Hicken, J. E., "Improved constraint-aggregation methods," *Computer Methods in Applied Mechanics and Engineering*, Vol. 289, 2015, pp. 332–354. doi:10.1016/j.cma.2015.02.017.

[19] Lee, K., Trask, N. A., Patel, R. G., Gulian, M. A., and Cyr, E. C., "Partition of unity networks: deep hp-approximation," *arXiv preprint arXiv:2101.11256*, 2021.

[20] OuYang, D., and Feng, H.-Y., "On the normal vector estimation for point cloud data from smooth surfaces," *Computer-Aided Design*, Vol. 37, No. 10, 2005, pp. 1071–1079. doi:10.1016/j.cad.2004.11.005.

[21] Bezanson, J., Edelman, A., Karpinski, S., and Shah, V. B., "Julia: A Fresh Approach to Numerical Computing," *SIAM Review*, Vol. 59, No. 1, 2017, pp. 65–98. doi:10.1137/141000671.

[22] Carlsson, K., Karrasch, D., Bauer, N., Kelman, T., Schmerling, E., Hoffimann, J., Visser, M., San-Jose, P., Christie, J., Ferris, A., Blaom, A., Pasquier, B., Foster, C., Saba, E., Goretkin, G., Orson, I., Samuel, O., Choudhury, S., and Nagy, T., "KristofferC/NearestNeighbors.jl:v0.4.9," https://github.com/KristofferC/NearestNeighbors.jl, 2021. doi:10.5281/zenodo.4943232.

[23] Patrikalakis, N. M., and Maekawa, T., *Shape interrogation for computer aided design and manufacturing*, Vol. 15, Springer, 2002.

[24] Hunter, J. D., "Matplotlib: A 2D graphics environment," *Computing In Science & Engineering*, Vol. 9, No. 3, 2007, pp. 90–95.

[25] Ahrens, J., Geveci, B., and Law, C., "Paraview: An end-user tool for large data visualization," *The Visualization Handbook*, Vol. 717, No. 8, 2005.

[26] Polanco, J. I., "WriteVTK.jl: a Julia package for writing VTK XML files (1.10.1)," https://github.com/jipolanco/WriteVTK.jl, 2021. doi:10.5281/zenodo.5634113.