SIAM J. SCI. COMPUT.
Vol. 37, No. 2, pp. A993–A1019

HIGH-ORDER QUADRATURE METHODS FOR IMPLICITLY DEFINED SURFACES AND VOLUMES IN HYPERRECTANGLES*

R. I. SAYE†

Abstract. A high-order accurate numerical quadrature algorithm is presented for the evaluation of integrals over curved surfaces and volumes which are defined implicitly via a fixed isosurface of a given function restricted to a given hyperrectangle. By converting the implicitly defined geometry into the graph of an implicitly defined height function, the approach leads to a recursive algorithm on the number of spatial dimensions which requires only one-dimensional root finding and one-dimensional Gaussian quadrature. The computed quadrature scheme yields strictly positive quadrature weights and inherits the high-order accuracy of Gaussian quadrature: a range of different convergence tests demonstrate orders of accuracy up to 20th order. Also presented is an application of the quadrature algorithm to a high-order embedded boundary discontinuous Galerkin method for solving partial differential equations on curved domains.

Key words. quadrature, integration, implicit surfaces, level set function, level set methods, high order

AMS subject classifications. 65D30, 65N30

DOI. 10.1137/140966290

1. Introduction.

In this paper, we develop high-order accurate numerical quadrature methods for the evaluation of integrals over curved surfaces and volumes whose geometry is defined implicitly via a fixed isosurface/level set of a smooth function $\phi : \mathbb{R}^d \to \mathbb{R}$. In particular, assuming that the zero level set of ϕ defines the geometry, let $\Gamma = \{x : \phi(x) = 0\}$ denote the surface ("interface") and let $\Omega = \{x : \phi(x) < 0\}$ be the region on the negative side of the surface. We consider surface integrals and volume integrals of the form

$$\int_{\Omega \cap U} f \, dx \quad \text{and} \quad \int_{\Gamma \cap U} g \, dS, \tag{1.1}$$

where $U \subset \mathbb{R}^d$ is a given hyperrectangle (e.g., a rectangle in two dimensions, or rectangular box in three dimensions), with sufficiently smooth but otherwise arbitrary integrands f and g. Integrals like these arise in a variety of applications involving implicitly defined geometry, such as in level set methods [19, 25, 18] for propagating interfaces in computational physics, embedded boundary methods for solving partial differential equations on curved domains [13], and in the treatment of jump conditions and singular source terms [3, 28, 26, 12, 6]. As an example, the weak formulation of a finite volume or finite element method may require integration over curved parts of a mesh element, thus requiring integrals of the form (1.1) where U is an individual

*Submitted to the journal's Methods and Algorithms for Scientific Computing section April 23, 2014; accepted for publication (in revised form) January 14, 2015; published electronically April 23, 2015. This research was supported by a Luis W. Alvarez Postdoctoral Fellowship at Lawrence Berkeley National Laboratory, by the Laboratory Directed Research and Development Program of LBNL, and by the Applied Mathematics Program of the U.S. DOE Office of Advanced Scientific Computing Research under contract DE-AC02-05CH11231. Some computations used the resources of the National Energy Research Scientific Computing Center (NERSC), which is supported by the Office of Science of the U.S. DOE under contract DE-AC02-05CH11231.
http://www.siam.org/journals/sisc/37-2/96629.html
†Applied Mathematics Department, Lawrence Berkeley National Laboratory, Berkeley, CA 94720 (rsaye@lbl.gov).

mesh element. It is often necessary to calculate these integrals with a high degree of accuracy in order to ensure consistency in the weak formulation (i.e., reduce associated "variational crimes"). Since the corresponding integrands are often high-order polynomials, this in turn requires a high-order quadrature scheme to calculate the integrals in (1.1) on individual mesh elements. In this paper, we restrict our attention to the case that U is a hyperrectangle; extension of the presented approach to other cases, e.g., triangular or tetrahedral elements, is briefly discussed.

For a sufficiently smooth but otherwise arbitrary level set function ϕ, it is difficult to directly construct accurate quadrature schemes for the integrals in (1.1). It follows that a quadrature scheme must be computed; i.e., a set of quadrature nodes and weights needs to be found. The geometry of $\Omega \cap U$ and $\Gamma \cap U$ will necessarily be approximated in the process. The goal is to find an efficient quadrature scheme with an acceptable number of quadrature nodes which yields a high-order approximation to the integrals in (1.1).

The quadrature algorithm developed in this work has the following features:

- The output is a quadrature scheme of the form
$$\int_{\Omega\cap U} f\,dx \approx \sum_i w_i f(x_i) \quad \text{and} \quad \int_{\Gamma\cap U} g\,dS \approx \sum_j \tilde{w}_j g(\tilde{x}_j),$$
where the weights w_i and $\tilde{w}_j$ are strictly positive, with the quadrature nodes strictly inside their respective domains: $x_i \in \Omega \cap U$ and $\tilde{x}_j \in \Gamma \cap U$. Thus, there is no need to define or extend f or g throughout all of U.
- The algorithm is based on Gaussian quadrature and inherits its high-order convergence rates. In particular, in d dimensions, based on a Gaussian quadrature scheme of order q, the total number of quadrature nodes is $\mathcal{O}(q^d)$ in the case of the volume integral and $\mathcal{O}(q^{d-1})$ in the case of the surface integral. For sufficiently smooth problems, the order of accuracy is approximately $2q$.
- Continuity of the level set function between different grid cells/mesh elements U is not assumed; it is only required that ϕ be smoothly defined in U itself. The method can therefore be used, for example, in a discontinuous Galerkin setting in which ϕ is evolved using level set methods discretized by discontinuous Galerkin methods.
- The overall approach is dimension independent; in particular, the same formulation can be used in both two- and three-dimensional applications.

The outline of the paper is as follows. In section 2, common strategies for evaluating integrals on implicitly defined geometry are briefly reviewed for comparison. Section 3 presents the high-order quadrature algorithm developed in this work, followed by a series of convergence tests in section 4 which examine the accuracy and behavior of the algorithm in different scenarios. Lastly, section 5 presents an application of the method to an embedded boundary discontinuous Galerkin method for solving partial differential equations on curved geometry before concluding with a short discussion in section 6.

2. General approaches and related work. General strategies for computing integrals on implicitly defined curved surfaces or volumes typically involve one of the following techniques: approximating the geometry of the interface by geometrically reconstructing it, using discrete Dirac delta or Heaviside functions, or appealing to the divergence theorem to reduce the dimension of the problem by replacing the integral with a boundary integral.

Methods which explicitly reconstruct the interface typically use a form of piecewise linear interpolation to find a faceted mesh representation of the surface, for example, through marching cubes [14] or marching tetrahedra [9, 20, 4] algorithms. Standard quadrature schemes are then applied to the surface and volume elements (e.g., triangles or tetrahedra) and result in a second-order accurate approximation of the surface and volume integrals; see, e.g., [15, 11, 31]. For increased accuracy, subdivision techniques are often used to locally refine the mesh geometry. However, the number of subdivisions, and thus ultimately the accuracy, is often limited as otherwise this technique can become too computationally expensive. This can be avoided by using higher-order interface reconstruction schemes, which typically involve transforming a quadrature scheme defined on reference shape (e.g., a triangle) into one appropriate for a curved shape by means of a smooth mapping; see, e.g., [5], which demonstrates a third-order accurate method. However, the construction of well-defined mappings with smooth Jacobians suitable for high-order quadrature can be intricate and difficult, due to their sensitive dependence on the topology of the interface inside an element.

An approach which does not require explicit reconstruction of the interface is to use discrete delta functions or discrete Heaviside functions. For a level set function ϕ defined on a uniform Cartesian grid, this method replaces the surface integral and the volume integral by a weighted summation of the integrand over all grid points x_i of the Cartesian grid:

$$\int_\Gamma f = \int_V f\delta(\phi)|\nabla\phi| \approx h^d \sum_i f(x_i)\delta_h(\phi(x_i))|\nabla_h\phi(x_i)|$$

and

$$\int_\Omega f = \int_V fH(-\phi) \approx h^d \sum_i f(x_i)H_h(-\phi(x_i)).$$

Here, V is a rectangular domain enclosing all of Ω, while δ_h and H_h are discrete versions of the Dirac delta and Heaviside functions which are smoothed out by an amount depending on the grid cell size h. Tornberg and Engquist [29] showed that a common choice for the regularized delta function may lead to a nonconvergent scheme with $\mathcal{O}(1)$ errors as $h \to 0$. Improved discretizations take into account gradient information of the level set function to locally modify the amount of smoothing and lead to first- or second-order accurate schemes [8, 27, 30, 35, 16]. Higher-order discretizations are possible, as demonstrated by Wen [32, 33, 34], where fourth-order accurate discrete delta functions are computed. However, all of these schemes, both low-order and high-order, have a strong reliance on the cancellation of errors in the summation over regularly spaced grid points. It follows that they have limited use on highly unstructured meshes, for piecewise discontinuous level set functions, or for surfaces which are not closed.

Another possibility is to make use of the divergence theorem to rewrite the integral as a boundary integral, as follows. Let U be a mesh element (such as a single cell of a Cartesian grid or a tetrahedron in a three-dimensional mesh) and consider the volume integral $\int_{\Omega\cap U} f$, where f is known. Suppose we can construct a vector-valued function $\mathbf{u} : \mathbb{R}^d \to \mathbb{R}^d$ whose divergence is f, e.g., in three dimensions, one possibility is to define $\mathbf{u}(x, y, z) = \left(\int_0^x f(s, y, z)\, ds, 0, 0\right)$, computed via analytical or numerical

integration. Then

$$\int_{\Omega\cap U} f = \int_{\Omega\cap U} \nabla\cdot\mathbf{u} = \int_{\partial(\Omega\cap U)} \mathbf{u}\cdot\mathbf{n} = \int_{\Gamma\cap U} \mathbf{u}\cdot\mathbf{n} + \int_{\Omega\cap\partial U} \mathbf{u}\cdot\mathbf{n}, \tag{2.1}$$

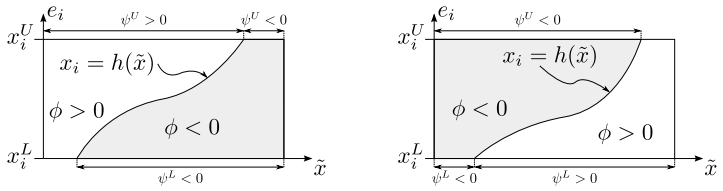
where $\mathbf{n}$ is the unit outward pointing normal vector on the boundary of $\Omega\cap U$. Note that the last integral on the right is on the boundary of the mesh element U, while the second last integral is of the form of an implicitly defined surface integral. Now consider the surface integral alone, i.e., $\int_{\Gamma\cap U} g$, where g is known. In this case, suppose that we can find or construct a different vector-valued function $\tilde{\mathbf{u}}$ satisfying two conditions: $\nabla\cdot\tilde{\mathbf{u}} = 0$ in U and $\tilde{\mathbf{u}}\cdot\mathbf{n} = g$ on the surface Γ. Then by a similar application of the divergence theorem,

$$\int_{\Gamma\cap U} g = -\int_{\Omega\cap\partial U} \tilde{\mathbf{u}}\cdot\mathbf{n}. \tag{2.2}$$

Thus, both the volume integral in (2.1) and the surface integral in (2.2) can be converted into an integral over that part of the boundary of U which is inside Ω, provided the proxy vector-valued fields can be found. The resulting integrals are themselves integrals over implicitly defined domains, and so the procedure can be repeated in one fewer dimensions, leading to a recursive scheme on the number of spatial dimensions. Müller, Kummer, and Oberlack [17] used this technique to construct high-order quadrature schemes for quadrilateral, triangular, and hexahedral elements. In that work, it was assumed that the surface integral integrand, g, could be smoothly extended off the surface Γ and that the normal vector field induced by ϕ, i.e., $\nabla\phi/|\nabla\phi|$, was smooth throughout U; with these assumptions, a moment-fitting method was used to choose the best representative for $\tilde{\mathbf{u}}$ coming from a finite-dimensional space of divergence-free vector-valued functions via a least-squares problem. In particular, for p quadrature nodes per dimension (i.e., p^d in total for each U), approximate convergence rates of order $p+1$ were demonstrated [17]. We note, however, that techniques which use integration by parts may not directly work if the boundary term is empty, i.e., $\Omega\cap\partial U = \varnothing$. If it is empty, then (2.2) necessarily requires that $\int_{\Gamma\cap U} g = 0$, which is not true for arbitrary g. This inconsistency arises in assuming that a divergence-free vector-valued function with the required property does indeed exist. As we later show in Figure 2, situations for which $\Omega\cap\partial U$ is empty can arise even for highly resolved surfaces. Special care must be taken in such circumstances (such as using subdivision strategies), in order to ensure that the geometry of Γ has been accurately captured.

In comparison, the high-order method presented in this paper produces a quadrature scheme for each individual mesh element U which only requires the integrands to be evaluated on $\Omega\cap U$ (for volume integrals) or $\Gamma\cap U$ (for surface integrals). As a result, the scheme does not rely on cancellation of errors when summing over the entire grid, nor does it rely on extension functions of the integrand. In addition, the quadrature weights are strictly positive—this can be advantageous, for example, in the context of finite element methods since then the associated mass matrices computed by the quadrature scheme will automatically be positive definite.

3. Quadrature on implicitly defined domains via dimension reduction. Since the implicit representation of a surface or volume has little explicit dependence on the number of spatial dimensions, a motivating goal in designing a quadrature scheme for implicitly defined domains is for the algorithm to be applicable in any number of spatial dimensions. In addition, we would like it to be high-order accurate,



(a) $\partial_{x_i}\phi > 0$ (b) $\partial_{x_i}\phi < 0$

FIG. 1. *Depending on the sign of $\partial_{x_i}\phi$, assumed constant throughout the hyperrectangle, the region $\{\phi < 0\}$ is either* (a) *beneath or* (b) *above the graph of the height function $x_i = h(\tilde{x})$ characterizing the zero level set of ϕ. Here, $\tilde{x} \in \mathbb{R}^{d-1}$ and so the $\tilde{x}$-axis abstractly represents a $(d-1)$-dimensional space.*

suggesting that methods like Gaussian quadrature be used if possible. An approach adopted here that achieves these objectives is to convert the implicitly defined surface Γ into the graph of an implicitly defined height function. Using such a characterization of the domain, the quadrature scheme can proceed in a natural way: by (i) performing an integration in the axis corresponding to the height direction, together with (ii) an integration in the tangential direction. Part (i) can be accomplished with standard Gaussian quadrature schemes, while part (ii) can itself be described as an integral over an implicitly defined region, in one fewer spatial dimensions, whose associated integrand evaluates part (i). This suggests using a recursive scheme in which the number of spatial dimensions is reduced one at a time. By identifying and treating separately the regions in which the resulting integrands are smooth, high-order accuracy can be achieved throughout the dimension-reduction process. Meanwhile, the geometry of the interface is inferred through the evaluation of the implicitly defined height functions, which, in turn, can be performed via one-dimensional root finding.

To illustrate, suppose that $\phi : \mathbb{R}^d \to \mathbb{R}$ is a sufficiently smooth level set function defining a volume $\Omega = \{x : \phi(x) < 0\}$ and a smooth codimension-one surface $\Gamma = \{x : \phi(x) = 0\}$, and let $\mathcal{U} = (x_1^L, x_1^U) \times \cdots \times (x_d^L, x_d^U) \subset \mathbb{R}^d$ be a given hyperrectangle. Suppose further that we can find a coordinate direction, e_i say,[1] such that $|\partial_{x_i}\phi|$ is bounded away from zero on $\mathcal{U}$.[2] Then the implicit function theorem of multivariable calculus guarantees the existence of a "height function" $h = h(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d)$ which represents the surface $\Gamma \cap \mathcal{U}$ as a graph of h such that

$$\phi\big(x_1, \ldots, x_{i-1}, h(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d), x_{i+1}, \ldots, x_d\big) = 0. \tag{3.1}$$

For brevity, define $\tilde{x} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d)$ and denote by $\tilde{x} + ye_i$ the point $(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_d)$. Based on the sign of $\partial_{x_i}\phi$, which by assumption does not change throughout $\mathcal{U}$, the region Ω is either above or below the graph $x_i = h(\tilde{x})$; see Figure 1. The location of the interface can also be determined in terms of the sign of ϕ when it is restricted to the upper and lower faces. Define $\psi^L, \psi^U : \mathbb{R}^{d-1} \to \mathbb{R}$, by

$$\psi^L(\tilde{x}) = \phi(\tilde{x} + x_i^L e_i), \quad \psi^U(\tilde{x}) = \phi(\tilde{x} + x_i^U e_i),$$

to be the restriction of ϕ to the lower and upper faces. Then with the aim of evaluating the volume integral $\int_{\Omega\cap\mathcal{U}} f$ and surface integral $\int_{\Gamma\cap\mathcal{U}} g$, we have the following scenarios:

[1] e_i denotes the standard basis vector in the direction of the ith coordinate.

[2] If $\Gamma \cap \mathcal{U} \neq \varnothing$ and $\mathcal{U}$ is sufficiently small, finding such a direction is always possible for a well-defined level set function, i.e., when $|\nabla\phi|$ is bounded away from zero on Γ.

- *Volume integral.* If $\partial_{x_i}\phi > 0$ in $\mathcal{U}$, then Ω is beneath the graph of h (see Figure 1(a)). The volume integral $\int_{\Omega\cap\mathcal{U}} f$ can then be rewritten as

$$\int_{\Omega\cap\mathcal{U}} f\,dx = \int_{V_1\cap V_2} \tilde{f}(\tilde{x})\,d\tilde{x}, \tag{3.2}$$

where the implicitly defined domain in one fewer spatial dimensions is defined by $V_1 \cap V_2$, where

$$V_1 = \{\tilde{x} : \psi^L(\tilde{x}) < 0\}, \quad V_2 = \{\tilde{x} : \psi^U(\tilde{x}) < 0\} \cup \{\tilde{x} : \psi^U(\tilde{x}) > 0\},$$

while the new integrand is defined by $\tilde{f}(\tilde{x}) := \int_{I(\tilde{x})} f(\tilde{x} + ye_i)\,dy$, where

$$I(\tilde{x}) := \{y \in (x_i^L, x_i^U) : \phi(\tilde{x} + ye_i) < 0\}.$$

We note two important aspects. First, the definition of V_1 depends only on ψ^L, while V_2 depends only on ψ^U. Second, we have made a careful choice in the definition of V_2 to exclude the zero level set of ψ^U. This is because the functional $\tilde{f}$ is not always a smooth function on the closure of $V_1 \cap V_2$ due to $I(\tilde{x})$ not being a smooth function of $\tilde{x}$: referring to Figure 1, we observe that $I(\tilde{x})$ is constrained by the height function for $\tilde{x} \in \{\psi^L < 0\} \cap \{\psi^U > 0\}$, while for $\tilde{x} \in \{\psi^U < 0\}$ it is constrained only by the hyperrectangle extent. However, by excluding the zero level set of ψ^U from V_2, it follows that $\tilde{f}$ is smooth on each connected component of $V_1 \cap V_2$. In order to yield a high-order accurate quadrature scheme, the strategy therefore is to treat each component separately and apply the quadrature algorithm to each connected component of $V_1 \cap V_2$.

If, on the other hand, $\partial_{x_i}\phi < 0$ in $\mathcal{U}$, then Ω is above the graph of h (see Figure 1(b)). A similar result also holds in this case:

$$\int_{\Omega\cap\mathcal{U}} f\,dx = \int_{V_1\cap V_2} \tilde{f}(\tilde{x})\,d\tilde{x}, \tag{3.3}$$

where $\tilde{f}$ is defined as earlier, but this time the implicitly defined domain in one fewer dimensions is defined via

$$V_1 = \{\tilde{x} : \psi^L(\tilde{x}) < 0\} \cup \{\tilde{x} : \psi^L(\tilde{x}) > 0\}, \quad V_2 = \{\tilde{x} : \psi^U(\tilde{x}) < 0\}.$$

Here the zero level set of ψ_1 has been excluded from V_1 so that we may assume that $\tilde{f}$ is smooth on each connected component of $V_1 \cap V_2$.

- *Surface integral.* A similar dimension-reduction method applies to the case of the surface integral $\int_{\Gamma\cap\mathcal{U}} g$. However, one must correctly account for the curvature of the surface. The surface area element on the graph of h is $\sqrt{1 + |\nabla_{\tilde{x}} h|^2}\,d\tilde{x}$; by utilizing (3.1), we can avoid calculating the derivatives of h by noting that $\sqrt{1 + |\nabla_{\tilde{x}} h|^2} = |\nabla\phi|/|\partial_{x_i}\phi|$ evaluated on Γ. It follows that

$$\int_{\Gamma\cap\mathcal{U}} g\,dS = \int_{V_1\cap V_2} g \frac{|\nabla\phi|}{|\partial_{x_i}\phi|}\bigg|_{\tilde{x}+h(\tilde{x})e_i} d\tilde{x}, \tag{3.4}$$

where V_1 and V_2 depend on the sign of $\partial_{x_i}\phi$ as follows:

$$V_1 = \begin{cases} \{\tilde{x} : \psi^L(\tilde{x}) < 0\} & \text{if } \partial_{x_i}\phi > 0 \text{ in } \mathcal{U}, \\ \{\tilde{x} : \psi^L(\tilde{x}) > 0\} & \text{if } \partial_{x_i}\phi < 0 \text{ in } \mathcal{U}, \end{cases}$$

$$V_2 = \begin{cases} \{\tilde{x} : \psi^U(\tilde{x}) > 0\} & \text{if } \partial_{x_i}\phi > 0 \text{ in } \mathcal{U}, \\ \{\tilde{x} : \psi^U(\tilde{x}) < 0\} & \text{if } \partial_{x_i}\phi < 0 \text{ in } \mathcal{U}. \end{cases}$$

Observe that in all of the above cases, by means of an implicitly defined height function, the integrals over $\Omega \cap \mathcal{U}$ or $\Gamma \cap \mathcal{U}$ can be converted into a volumetric integral in one fewer dimensions, as in (3.2), (3.3), and (3.4). The domain of this new integral is determined implicitly by conditions on the sign of ϕ when restricted to the lower and upper faces of $\mathcal{U}$. Meanwhile, evaluation of $I(\tilde{x})$ and the height function can be accomplished by means of one-dimensional root finding on ϕ, as discussed shortly.

Consider applying this procedure recursively. Each recursive call leads to a new integration problem with a new implicitly defined domain and integrand, and each of these will require finding a suitable "height function direction" to work with. The new implicitly defined domains (similar to V_1 and V_2 above), each of which is confined to lower-dimensional versions of $\mathcal{U}$, are determined by sign conditions of the level set function restricted to a particular lower and upper hyperplane (similar to ψ^L and ψ^U above). We construct these dimension-reduced problems by manipulating a set of multiple level set functions, each of which will ultimately be the restriction of the original level set function ϕ to a particular face, edge, etc., of $\mathcal{U}$. For example, at the top level of recursion, the set consists of just ϕ; after one level, the new set consists of ψ^L and ψ^U; after two levels, each of these may again split into two. In general, as the number of spatial dimensions is reduced, the recursive approach may lead to an increasing number of functions and associated sign conditions. However, as discussed shortly, this list can often be efficiently "pruned" so that typically just one or two functions persist.

We now make these ideas more concrete by fully developing the numerical quadrature algorithm, one step at a time. An outline for the remainder of this section is as follows:

- In order to find a height function direction and deem it suitable, i.e., to ensure monotonicity, it is necessary to determine whether particular derivatives of ϕ are uniform in sign. This is special case of the more general problem of placing bounds on the attainable values of a given function in a given hyperrectangle, and section 3.1 discusses how this can be achieved.
- Section 3.2 formulates the general, dimension-independent, recursive algorithm for implicitly defined domains defined by a set of multiple level set functions for both volume and surface integrals. The algorithm proceeds by first "pruning" the set as a preprocessing step, which may allow the use of a tensor product Gaussian quadrature scheme in the case that the domain of integration is the entire hyperrectangle. It is then discussed how a suitable height function direction can be chosen, and if this is not possible, the need for a subdivision process. The recursive call is then established by constructing a new set of functions and sign conditions together with a new integrand.
- The one-dimensional base case is treated in section 3.3.
- Lastly, section 3.4 provides a concise description of the algorithms in pseudocode, together with a brief discussion on related implementation choices.

3.1. Bounds evaluation. A key mechanism in the operation of the quadrature algorithm is the ability to place bounds on the range of attainable values of a given function in a given hyperrectangle U. Specifically, given a smooth function[3] $\psi : U \to \mathbb{R}$, we seek to calculate lower and upper bounds c and C such that

$$c \le \inf_{x \in U} \psi(x) \le \sup_{x \in U} \psi(x) \le C.$$

As will be seen later, the general desire is for these bounds to be as tight as possible, increasing in accuracy as the size of U shrinks. Depending on the particular form of ψ, various possibilities exist for calculating such bounds. For example, one possibility is to expand ψ about some fixed point in U via a Taylor series and then bound each of the resulting terms. In the appendix, we outline an approach which was found to be very convenient in this work; the technique is similar to automatic differentiation and interval arithmetic and can be combined with template programming to automatically compute a first-order Taylor series with bounded remainder of typical functions implemented by a computer program. In particular, the method allows one to automatically evaluate bounds using the same piece of code that evaluates $x \mapsto \psi(x)$.

3.2. General dimension-reduction approach. To describe the general algorithm applicable to any spatial dimension d, we mainly consider the case of the volume integral; the surface integral requires minor adjustments that are mostly deferred to section 3.4. We also assume in this section that $d > 1$.

In the most general setting, suppose we are given a hyperrectangle[4] $U = (x_1^L, x_1^U) \times \cdots \times (x_d^L, x_d^U) \subset \mathbb{R}^d$ and a collection of n functions[5] $\psi_i : U \to \mathbb{R}$, $i = 1, \ldots, n$, with an associated set of conditions s_i on the signs of ψ_i which determine a domain of integration V as follows: for each of these functions, define

$$V_i = \begin{cases} \{x \in U : \psi_i(x) > 0\} & \text{if } s_i = +1, \\ \{x \in U : \psi_i(x) < 0\} & \text{if } s_i = -1, \\ \{x \in U : \psi_i(x) > 0\} \cup \{x \in U : \psi_i(x) < 0\} & \text{if } s_i = 0 \end{cases} \tag{3.5}$$

and define the overall domain of integration as $V = \bigcap_{i=1}^n V_i$. The goal is to find a quadrature scheme for $\int_V f$. For example, the original volume integral $\int_{\Omega \cap \mathcal{U}} f$ would be evaluated with the input $n = 1$, $\psi_1 \equiv \phi$, and $s_1 = -1$. In other cases, the ψ_i functions will be restrictions of ϕ to particular faces/edges/etc. of the original hyperrectangle $\mathcal{U}$. To compute $\int_V f$, we use recursion on the dimension d by following the idea of implicitly finding height functions, one coordinate direction at a time. The general approach consists of the following steps: (i) pruning the list of functions, (ii) applying a tensor product integral if possible, (iii) determining a height function coordinate direction, and (iv) formulating the recursive call, as follows.

3.2.1. Pruning. In the first step, we attempt to simplify the definition of V by removing all those functions ψ_i in the list that we can prove are uniformly positive or

[3] Typically ψ will be the original level set function ϕ or one of its derivatives, restricted to the original hyperrectangle $\mathcal{U}$ or to one of its faces, edges, etc. Moreover, in practice, ψ will usually be a polynomial or a smooth composition of analytic functions like sin, exp, etc.

[4] In this section, the hyperrectangle U will either be the original hyperrectangle $\mathcal{U}$ or represent lower-dimensional restrictions of $\mathcal{U}$.

[5] In practice, it is typically the case that n is one or two—possibly a few more when the geometry of the problem is complex.

negative on U. This preprocessing step, which "prunes" the list of functions, improves the overall efficiency of the recursive algorithm without affecting its accuracy. To accomplish this, for each i we use bounds evaluation as discussed in section 3.1 to calculate lower and upper bounds c and C such that

$$c \leq \inf_{x \in U} \psi_i(x) \leq \sup_{x \in U} \psi_i(x) \leq C.$$

If $c > 0$ and $s_i \geq 0$, then we can remove ψ_i from the list since it is uniformly positive on U and so $V_i = U$. Similarly, if $C < 0$ and $s_i \leq 0$, then ψ_i is uniformly negative and can also be removed from the list. On the other hand, if $s_i = +1$ and $C < 0$, or if $s_i = -1$ and $c > 0$, then $V_i = \varnothing$. In this latter case, $V = \varnothing$, i.e., the domain of integration is empty, in which case the quadrature scheme to evaluate $\int_V f$ is empty and the algorithm immediately terminates.

3.2.2. Tensor product integral. It may be the case that the previous pruning step leaves an empty list of functions, i.e., $n = 0$. This will occur when the bounds evaluation determines that the domain of integration is the entire hyperrectangle U. If this is the case, a simple tensor product Gaussian quadrature scheme can be used:

$$\int_V f\,dx = \int_U f\,dx \approx \sum_{i_1=1}^{q} \cdots \sum_{i_d=1}^{q} w_{i_1} \cdots w_{i_d} f(x_{i_1}, \ldots, x_{i_d}),$$

where q is the order of the Gaussian quadrature method and w_i and x_i are the corresponding quadrature weights and points suitably transformed to the hyperrectangle U. After evaluating the quadrature rule, the algorithm returns the result and the recursion terminates.

3.2.3. Determine a height function coordinate direction. If it could not be determined via bounds evaluation that the domain of integration is empty or all of U, we employ height functions as motivated by the introduction of section 3. This consists of two steps: (i) proposing a height coordinate direction, and (ii) ensuring that the direction is a suitable one.

(i) A simple method for choosing a height function direction can be used, which is to use the component of $\nabla\psi_1(x_c)$ with largest absolute value where x_c is the center of U:

$$k = \arg\max_{i=1,\ldots,d} |\partial_{x_i} \psi_1(x_c)|.$$

(ii) The direction is deemed suitable if two conditions hold for all of the ψ_i functions: (a) $|\partial_{x_k} \psi_i| > 0$ in U for all i, and (b) $|\nabla\psi_i| / |\partial_{x_k} \psi_i| < C$ for all i for some fixed parameter C. Condition (a) guarantees the existence of the height function as well as monotonicity of ψ_i in the direction e_k. Condition (b) concerns the curved-surface Jacobian factor that explicitly arises in the surface integral (3.4) and implicitly arises in the volume integrals (3.2) and (3.3) (as derivatives of the integrands). Since Gaussian quadrature will be used to evaluate these integrals, ideally this ratio (relating to how "steep" the height function is) should be as smooth as possible throughout U. To measure the degree of smoothness, in this work we have used a simple heuristic, which is to require that the ratio is bounded above by a user-defined constant; in particular we have used $C \approx 4$. Other methods for measuring the degree of smoothness of the Jacobian factor are also possible.

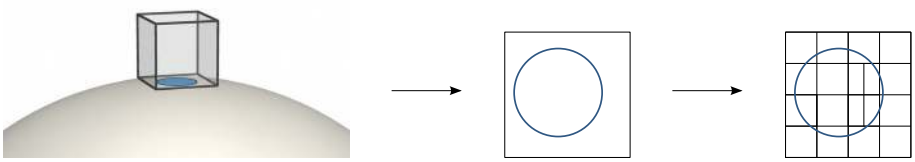


FIG. 2. *(left) A sphere embedded in three dimensions partially intersects a cube on one of its faces. (middle) Restricted to the face, the original interface is circular. (right) The subdivision process, which recursively divides rectangles into halves, eventually results in the shown subdivision. Some pieces are empty and are either discarded by the pruning process (section* 3.2.1*) or treated with a tensor product integral; on the other pieces there exists a suitable direction to form a height function.*

> In practice, to determine whether conditions (a) and (b) hold, we evaluate bounds on the range of attainable values of the derivatives of ψ_i in U. This can be accomplished with the same techniques for evaluating a first-order Taylor series with bounded remainder outlined in section 3.1 and the appendix where, in this case, they are applied to the expression for evaluating the gradient of ψ_i. Conditions (a) and (b) can then be checked by using these bounds, e.g., by evaluating $\max |\nabla \psi_i| / \min |\partial_{x_k} \psi_i|$.

In the vast majority of cases, the above procedure will successfully find a suitable height function direction. Geometrically, this corresponds to the property that a well-resolved surface Γ can be locally well approximated by a hyperplane. However, this may not always be the case. An example is as follows: Consider a spherical surface Γ (embedded in three dimensions) and a small three-dimensional cube $\mathcal{U}$ partially intersecting Γ on one of its faces—see Figure 2 (left). The dimension-reduction algorithm first chooses a height direction orthogonal to this face, thereby reducing the volume integral into a two-dimensional integral on the face. However, the original surface when restricted to this face is circular (Figure 2 (middle)). There is no coordinate direction on this face for which the circular interface could be described as a graph of a height function: neither $\partial_x \phi$ nor $\partial_y \phi$ have uniform sign across the entire face. To resolve this situation, we use a simple subdivision strategy: if the above procedure did not successfully find a height direction, the hyperrectangle U is subdivided into two halves and the quadrature algorithm is re-executed separately on each half, summing up the results. Such a subdivision strategy essentially refines U until the interface (restricted to the face) can be made to look locally flat, as required by the implicit function theorem. In this example of a spherical interface intersecting with the face of a cube, the result of the subdivision strategy is shown in Figure 2 (right). In practice, the number of subdivisions required by this process is almost always very few. Nevertheless, we limit the number of subdivisions so as to ensure the recursive algorithm terminates, as discussed in section 3.4 and further scrutinized in our results in section 4.3.

In summary, the above procedure chooses a height function direction and checks to see if it is suitable. If indeed it is, the algorithm proceeds to the next step; otherwise the hyperrectangle U is divided into two halves and a quadrature scheme is evaluated on each half and the results summed.

3.2.4. Dimension reduction. Given a height function direction e_k, the volume integral $\int_V f$ is then converted into a volume integral in one fewer dimensions. Recall that each ψ_i describes an implicitly defined domain in U via a condition on its sign

s_i, as in (3.5). The overall domain of integration is the intersection of all the V_i, i.e., all given functions must satisfy their respective sign conditions. As motivated by the introduction of section 3, since ψ_i is monotone in the direction e_k, the domain V_i can be equivalently characterized as the region above or below a height function (locating the zero level set of ψ_i in U), together with conditions on the sign of ψ_i when restricted to the lower and upper faces of U. To be more precise, define the new hyperrectangle $\tilde{U} = (x_1^L, x_1^U) \times \cdots \times (x_{k-1}^L, x_{k-1}^U) \times (x_{k+1}^L, x_{k+1}^U) \times \cdots \times (x_d^L, x_d^U) \subset \mathbb{R}^{d-1}$ and $\psi_i^L : \tilde{U} \to \mathbb{R}$ and $\psi_i^U : \tilde{U} \to \mathbb{R}$ by

$$\psi_i^L(x) := \psi(x + x_k^L e_k), \quad \psi_i^U(x) := \psi(x + x_k^U e_k).$$

Here we have used the notation where, given a point $x \in \mathbb{R}^{d-1}$, the expression $x + ye_k$ denotes the point in $\mathbb{R}^d$ given by $(x_1, \ldots, x_{k-1}, y, x_k, \ldots, x_{d-1})$. The sign conditions on the lower and upper faces are encoded in the two functions $\mathrm{sgn}_L(m, s, S) := \mathrm{sgn}(m, s, S, -1)$ and $\mathrm{sgn}_U(m, s, S) := \mathrm{sgn}(m, s, S, +1)$, where

$$\mathrm{sgn}(m, s, S, \sigma) = \begin{cases} \sigma m & \text{if } m = \sigma s \text{ or } S \text{ is true,} \\ 0 & \text{otherwise.} \end{cases}$$

Here, m is the sign of $\partial_{x_k} \psi_i$ in U, s is the sign condition s_i corresponding to ψ_i, and S is a boolean flag which indicates whether a surface integral or volume integral is being performed. Using these two functions, the new domain of integration in one fewer dimensions, inside the new hyperrectangle $\tilde{U}$, is specified by simultaneously requiring that ψ_i^L satisfy the sign condition $\mathrm{sgn}_L(\mathrm{sign}(\partial_{x_k} \psi_i), s_i, S)$ on $\tilde{U}$ and ψ_i^U satisfy the sign condition $\mathrm{sgn}_U(\mathrm{sign}(\partial_{x_k} \psi_i), s_i, S)$ on $\tilde{U}$. We denote this domain of integration as $\tilde{V}_i$ and define the overall new domain as $\tilde{V} = \bigcap_i \tilde{V}_i$.

The dimension-reduction algorithm is completed with the specification of the new integrand functional. In the case of the volume integral, the integrand is a one-dimensional integral along the height function coordinate direction, similar to (3.2) and (3.3), except that the domain of the one-dimensional integral should consist of that part of (x_k^L, x_k^U) for which the sign conditions on all ψ_i functions hold. To be precise, define

$$I_i(x) := \begin{cases} \{y \in (x_k^L, x_k^U) : s_i \psi_i(x + ye_k) > 0\} & \text{if } s_i = \pm 1, \\ \{y \in (x_k^L, x_k^U) : \psi_i(x + ye_k) \neq 0\} & \text{if } s_i = 0, \end{cases}$$

and $I(x) := \bigcap_{i=1}^n I_i(x)$. Then by construction it follows that

$$\int_V f = \int_{\tilde{V}} \left(\int_{I(x)} f(x + ye_k)\, dy \right) dx. \tag{3.6}$$

A new integrand functional which approximates the value of $x \mapsto \int_{I(x)} f(x + ye_k)\, dy$ is defined by using Gaussian quadrature on each of the connected components of I. For a fixed x, this involves calculating the roots of ψ_i along the coordinate direction e_k. Define the following operator that returns all roots of a continuous function $\lambda : \mathbb{R} \to \mathbb{R}$ in the interval (x_1, x_2):

$$\mathrm{roots}(\lambda, x_1, x_2) := \{x \in \mathbb{R} : x_1 < x < x_2 \text{ and } \lambda(x) = 0\}.$$

This operator is used to find which components of the interval (x_k^L, x_k^U) simultaneously satisfy all the sign conditions on ψ_i, as follows. Let

$$\mathcal{R} = \{x_k^L, x_k^U\} \cup \bigcup_{i=1}^n \mathrm{roots}(y \mapsto \psi_i(x + ye_k), x_k^L, x_k^U)$$

and sort $\mathcal{R}$ into ascending order such that $\mathcal{R} = \bigcup_{i=1}^{\ell} \{r_i\}$ with $r_1 \leq \cdots \leq r_\ell$. We keep those subintervals (r_j, r_{j+1}) for which $s_i \psi_i \geq 0$ for all i and apply Gaussian quadrature to each of these. For a given x, the end result is an approximation to the value of the functional $x \mapsto \int_{I(x)} f(x + y e_k)\, dy$ which is used by the quadrature scheme (via the recursive algorithm) on $\tilde{V}$.

In the case of the surface integral, there will be only one ψ_i function, and the integrand in (3.4) is simply a point evaluation of f on the original surface Γ, scaled by a Jacobian factor relating to the curvature of the surface. In this case, we also use root finding to determine the value of the height function. The details are left to section 3.4.

In summary, the above procedure constructs two new functions, ψ_i^L and ψ_i^U, for each i, with associated sign conditions s_i^L and s_i^U, which define a new domain $\tilde{V}$ on a hyperrectangle $\tilde{U}$ corresponding to the original hyperrectangle U with dimension k removed. The original integral is then rewritten as an integral over $\tilde{V}$ with a new integrand; this integrand involves one-dimensional root finding on the ψ_i functions in the coordinate direction e_k. Since $\tilde{V}$ is defined by a new set of ψ_i functions with associated sign conditions on $\tilde{U}$, the same algorithm can be used to evaluate the new integral over $\tilde{V}$. Thus we use recursion on the dimension d, eventually leading to the base case with $d = 1$. The base case executes a Gaussian quadrature scheme in one dimension, and this in turn evaluates the above constructed integrand functional at discrete points. This process travels back up through the tree of recursion, evaluating integrands by adding one more spatial dimension at a time, and eventually terminates by evaluating the original user-supplied integrand.

3.3. One-dimensional base case. In $d = 1$ dimensions, we are given n functions $\psi_i : \mathbb{R} \to \mathbb{R}$, $i = 1, \ldots, n$, with sign conditions s_i that determine a domain of integration $V = \bigcap_{i=1}^{n} V_i$, where V_i is defined by (3.5) and $U = (x_1^L, x_1^U)$ is a one-dimensional interval. The goal is to find $\int_V f\, dx$. We accomplish this by using Gaussian quadrature on each connected component of V. To find the connected components, we perform one-dimensional root finding on each of the functions ψ_i to find all their roots in the interval U, thereby partitioning the interval into subintervals. Gaussian quadrature is then applied to each individual subinterval that belongs to V.

3.4. Implementation. In this section we summarize the quadrature algorithm as pseudocode and discuss some implementation choices. We begin with the specification of the integrand functionals. Algorithm 1 shows the case for the volume integral, in which the integrand functional is a one-dimensional integral over a given height function coordinate direction; Algorithm 2 shows the case for the surface integral, in which the integrand functional is evaluation of the original integrand f on the surface Γ, scaled by the curved-surface Jacobian factor. A few remarks are in order:

- Recall that roots(λ, x_1, x_2) returns all the roots of a one-dimensional continuous function $\lambda : \mathbb{R} \to \mathbb{R}$ in the interval (x_1, x_2). Clearly, this operator must be approximated in a numerical setting, with a variety of possible implementations. For example, if the original level set function $\phi : \mathbb{R}^d \to \mathbb{R}$ is a polynomial of degree two, then the restriction of ϕ to any coordinate axis is a quadratic polynomial for which one could use the numerically stable variant of the quadratic formula. Throughout this work, we have used a hybrid of Newton's method and the bisection method, similar to that described in [21]. The method attempts to use Newton's method and reverts to bisection whenever Newton's method fails to converge quickly enough. In order to use

Algorithm 1. Integrand evaluation. Given $\psi_i : \mathbb{R}^d \to \mathbb{R}$ and s_i, $i = 1, \ldots, n$, a height direction k, an interval (x_1, x_2), a point $x \in \mathbb{R}^{d-1}$, a function $f : \mathbb{R}^d \to \mathbb{R}$, and the order of Gaussian quadrature, q, evaluate $\mathcal{F}(x; f, \{(\psi_i, s_i)\}_{i=1}^n, (x_1, x_2), q)$.

1: Calculate the set of roots and union with the interval endpoints:
$$\mathcal{R} = \{x_1, x_2\} \cup \bigcup_{i=1}^{n} \text{roots}(y \mapsto \psi_i(x + ye_k), x_1, x_2).$$
2: Sort $\mathcal{R}$ into ascending order such that $\mathcal{R} = \bigcup_{i=1}^{\ell} \{r_i\}$ and $r_1 \le \cdots \le r_\ell$.
3: Set $I = 0$.
4: **for** $j = 1$ **to** $\ell - 1$ **do**
5: Define $L = r_{j+1} - r_j$ and $x_c = x + \frac{1}{2}(r_j + r_{j+1})e_k$.
6: **if** $s_i \psi_i(x_c) \ge 0$ **for all** i **then**
7: Update $I := I + L \sum_{i=1}^{q} w_{q,i} f(x + (r_j + L x_{q,i})e_k)$.
8: **return** I.

Algorithm 2. Surface integrand evaluation. Given $\phi : \mathbb{R}^d \to \mathbb{R}$, a height direction k, an interval (x_1, x_2), a point $x \in \mathbb{R}^{d-1}$, and a function $f : \mathbb{R}^d \to \mathbb{R}$, evaluate $\mathcal{F}_{\text{surf}}(x; f, \phi, (x_1, x_2))$.

1: Calculate the roots of ϕ in the interval: $\mathcal{R} = \text{roots}(y \mapsto \phi(x + ye_k), x_1, x_2)$.
2: **assert**$(|\mathcal{R}| \le 1)$.
3: **if** $|\mathcal{R}| = 1$ **then**
4: Let $\mathcal{R} = \{r\}$.
5: **return** $f(x + re_k)|\nabla\phi(x + re_k)|/|\partial_{x_k}\phi(x + re_k)|$.
6: **else**
7: **return** 0.

such a hybrid, we need to know that the bracket being employed does indeed bracket a single root of λ. Fortunately, for all but the base case of the dimension-reduction recursive algorithm, we require that each ψ_i function be monotone in the height function coordinate direction, and so there is always at most one root. In the base case with $d = 1$ (see section 3.3), we did not require monotonicity of $\psi_i : \mathbb{R} \to \mathbb{R}$. In this particular case, in our implementation we used the same bounding techniques used in pruning to divide, if necessary, the interval (x_1, x_2) into subintervals for which the hybrid method can be applied. In all of our tests cases, the hybridized Newton's method performed well and essentially always yielded rapid convergence to the roots.

- In Algorithm 1, $\{w_{q,i}\}_{i=1}^q$ and $\{x_{q,i}\}_{i=1}^q$ denote the weights and nodes of the Gaussian quadrature scheme of order q corresponding to the unit interval $[0, 1]$. For example, for $q = 3$,
$$w_{3,1} = \tfrac{5}{18}, \quad x_{3,1} = \tfrac{1}{2}\big(1 - \sqrt{3/5}\big),$$
$$w_{3,2} = \tfrac{4}{9}, \quad x_{3,2} = \tfrac{1}{2},$$
$$w_{3,3} = \tfrac{5}{18}, \quad x_{3,3} = \tfrac{1}{2}\big(1 + \sqrt{3/5}\big).$$
- In Algorithm 2, on line 2, it is asserted that there is at most one root of the height function. This is because $\mathcal{F}_{\text{surf}}$ is only ever evaluated with a function (in fact, the original level set function ϕ) which is monotone in the direction e_k over the associated line segment.

Algorithm 3. Evaluate $\mathcal{I}(f;\{(\psi_i,s_i)\}_{i=1}^n,U,S,q)$. Given $\psi_i:\mathbb{R}^d\to\mathbb{R}$ and s_i, $i=1,\dots,n$, a hyperrectangle $U=(x_1^L,x_1^U)\times\cdots\times(x_d^L,x_d^U)$, a function $f:\mathbb{R}^d\to\mathbb{R}$, a boolean flag S, and the order of Gaussian quadrature, q, approximate the surface integral $\int_{\Gamma\cap U} f$ (if S is true) or the volume integral $\int_V f$ (if S is false).

1: **assert**(S is false or else $d>1$, $n=1$, and $\psi_1\equiv\phi$).
2: **if** $d=1$ **then return** $\mathcal{F}(0;f,\{(\psi_i,s_i)\}_{i=1}^n,(x_1^L,x_1^U),q)$.
3: Define $x_c\in\mathbb{R}^d$ to be the center of U, i.e., $x_{c,i}=\frac{1}{2}(x_i^L+x_i^U)$.
4: **for** $i=n$ **downto** 1 **do**
5: Evaluate bounds on the value of ψ_i on U such that $\sup_{x\in U}\big|\psi_i(x)-\psi_i(x_c)\big|\le\delta$.
6: **if** $|\psi_i(x_c)|\ge\delta$ **then**
7: **if** $s_i\psi_i(x_c)\ge 0$ **then**
8: Remove ψ_i from the list and decrement n by one.
9: **else**
10: The domain of integration is empty; **return** 0.
11: **if** $n=0$ **then**
12: Apply a tensor-product Gaussian quadrature scheme:
13: **return** $I=|U|\displaystyle\sum_{i_1,\dots,i_d=1}^{q} w_{q,i_1}\cdots w_{q,i_d} f\Big(\sum_{j=1}^{d}\big(x_j^L+(x_j^U-x_j^L)x_{q,i_j}\big)e_j\Big)$.
14: Set $k=\arg\max_j|\partial_{x_j}\psi_1(x_c)|$ and initialize $\tilde\psi=\varnothing$.
15: **for** $i=1$ **to** n **do**
16: Evaluate $g:=\nabla\psi_i(x_c)$.
17: Determine bounds $\delta\in\mathbb{R}^d$ such that $\sup_{x\in U}\big|\partial_{x_j}\psi(x)-g_j\big|\le\delta_j$ for each j.
18: **if** $|g_k|>\delta_k$ **and** $\big(\sum_{j=1}^d(g_j+\delta_j)^2\big)/(g_k-\delta_k)^2<20$ **then**
19: Define $\psi_i^L:\mathbb{R}^{d-1}\to\mathbb{R}$ and $\psi_i^U:\mathbb{R}^{d-1}\to\mathbb{R}$ by

$$\psi_i^L(x):=\psi(x+x_k^Le_k),\quad \psi_i^U(x):=\psi(x+x_k^Ue_k).$$

20: Evaluate signs: $s_i^L=\mathrm{sgn}_L(\mathrm{sign}(g_k),s_i,S)$, $s_i^U=\mathrm{sgn}_U(\mathrm{sign}(g_k),s_i,S)$.
21: Add to the collection: $\tilde\psi:=\tilde\psi\cup\{(\psi_i^L,s_i^L),(\psi_i^U,s_i^U)\}$.
22: **else**
23: The height function direction e_k is not suitable for ψ_i. If already subdivided too many times, revert to a low-order method (see discussion). Otherwise split U into two halves along its largest extent $\arg\max_j(x_j^U-x_j^L)$ such that $U=U_1\cup U_2$ and **return** $\mathcal{I}(f;\{(\psi_i,s_i)\}_{i=1}^n,U_1,S,q)+\mathcal{I}(f;\{(\psi_i,s_i)\}_{i=1}^n,U_2,S,q)$.
24: Define a new integrand $\tilde f:\mathbb{R}^{d-1}\to\mathbb{R}$ by

$$\tilde f(x):=\begin{cases}\mathcal{F}(x;f,\{(\psi_i,s_i)\}_{i=1}^n,(x_k^L,x_k^U),q) & \text{if } S \text{ is false,}\\ \mathcal{F}_{\mathrm{surf}}(x;f,\psi_1,(x_k^L,x_k^U)) & \text{if } S \text{ is true.}\end{cases}$$

25: **return** $\mathcal{I}\big(\tilde f;\tilde\psi,(x_1^L,x_1^U)\times\cdots\times(x_{k-1}^L,x_{k-1}^U)\times(x_{k+1}^L,x_{k+1}^U)\times\cdots\times(x_d^L,x_d^U),\mathrm{false},q\big)$.

The main recursive algorithm using dimension reduction is presented in Algorithm 3. Several remarks are in order:

- To find a quadrature scheme for the volume integral, the algorithm is executed by calling $\int_{\Omega\cap\mathcal{U}} f\approx\mathcal{I}(f;(\phi,-1),\mathcal{U},\mathrm{false},q)$. To find a quadrature scheme for the surface integral, the algorithm is executed by calling $\int_{\Gamma\cap\mathcal{U}} g\approx\mathcal{I}(g;(\phi,0),\mathcal{U},\mathrm{true},q)$.
- In the case of the surface integral, the flag S is true only at the topmost level—the surface integral is immediately converted to a volume integral on one of the faces of $\mathcal{U}$ via a height function. Thus on line 1 of Algorithm 3, we

make the assertion that either S is false, or else we have not yet performed any dimension reduction.

- On line 18, the numerical value of 20 implements the $C \approx 4$ parameter discussed in section 3.2.3 regarding bounding the Jacobian factor. It was determined empirically to provide a good balance between high accuracy and excessive subdivision.
- Line 23 is concerned with the subdivision algorithm in the case that a suitable height function direction was not found by the algorithm. We discussed in section 3.2.3 the necessity for subdivision, wherein the simple strategy of dividing U into two equal halves was used. It follows that the subdivision process may occur recursively. Although rare, it is possible for the process to never terminate; an example is shown in section 4.3. In such cases, we must terminate the recursion and revert to a low-order method. To do this, Algorithm 3 can be slightly modified to keep track of the number of "levels" or depth of recursion, and in the case that this exceeds a fixed number, 16 say, we apply the following scheme:

$$\mathcal{I}(f; \{(\psi_i, s_i)\}_{i=1}^n, U, S, q) = \begin{cases} |U| f(x_c) & \text{if } S \text{ is false and } s_i \psi_i(x_c) \geq 0 \text{ for all } i, \\ 0 & \text{otherwise.} \end{cases}$$

  In other words, in the case of the volume integral, if at the center x_c of U all the sign conditions of every ψ_i function are satisfied, then the algorithm returns the value of the integrand at the center scaled by the volume of U, and zero in all other cases. Clearly, this is not necessarily an accurate approximation to $\int_{\Omega \cap U} f$ or $\int_{\Gamma \cap U} f$. However, if this low-order method is executed, then it should be the case that U is already "small." Section 4.3 further discusses and analyzes the error incurred when such a situation arises.
- Although the presentation of Algorithm 3 is in a somewhat abstract form, particularly with regard to the construction and use of the ψ_i functions, in an actual implementation one can make use of the fact that each ψ_i function is simply the original level set function ϕ with one or more of the coordinate values "frozen" at x_k^L or x_k^U. Thus, instead of having to define and manipulate new ψ_i functions, one may instead record which coordinate values have been frozen and make use of this when calculating bounds on function values, root finding, quadrature points, etc.
- We also note that the pruning process generally means only one or two functions persist through the dimension-reduction process. For example, it is often the case that a level set function restricted to one of the lower or upper faces will be uniformly positive or negative.
- Lastly, we note that a few minor modifications of Algorithms 1, 2, and 3 would allow the overall quadrature scheme constructed by the method to be recorded. The quadrature scheme could thus be used multiple times on different integrands. In the case of the volume integral, the algorithm yields a quadrature scheme of the form $\int_{\Omega \cap \mathcal{U}} f \approx \sum_{i=1}^n w_i f(x_i)$, where the weights $w_i \in \mathbb{R}$ are positive and the points $x_i \in \mathbb{R}^d$ all lie in $\Omega \cap \mathcal{U}$. Thus, $\sum_i w_i$ provides an approximation to the volume of $\Omega \cap \mathcal{U}$. In the case of the surface integral, $\int_{\Gamma \cap \mathcal{U}} g \approx \sum_{i=1}^n w_i g(x_i)$, where the weights w_i are positive and the points x_i lie on $\Gamma \cap \mathcal{U}$; the sum $\sum_i w_i$ provides an approximation to the surface area of $\Gamma \cap \mathcal{U}$.

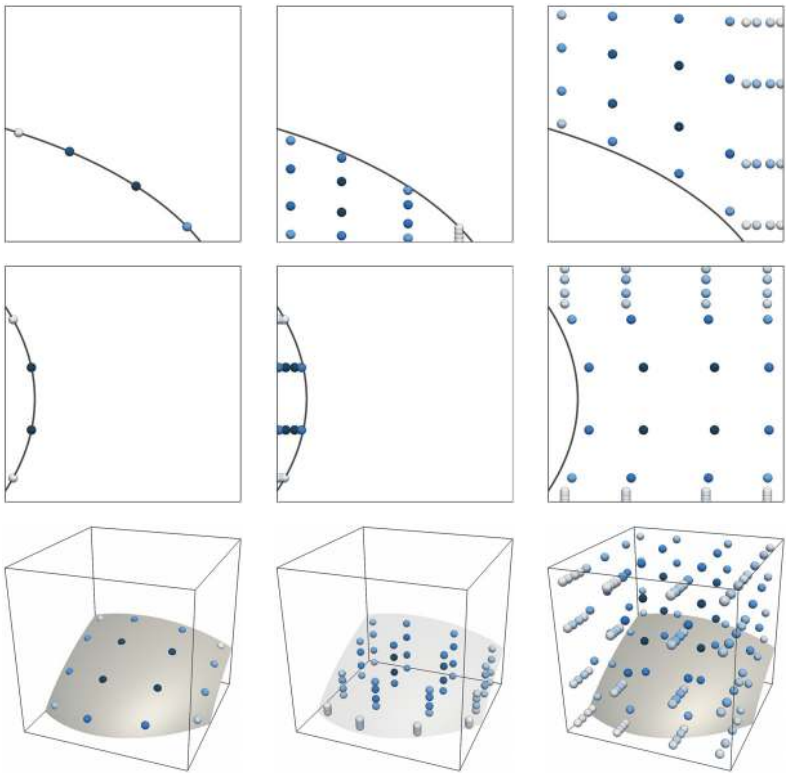


FIG. 3. *Some examples illustrating the quadrature schemes constructed by the algorithm (based on a Gaussian quadrature scheme of order $q = 4$) for the surface integral on the interface (left column) and volume integrals for the two regions on either side of the interface (middle and right columns). The weights are colored according to a scale that is normalized for each particular case: pale indicates a low-valued weight and dark blue indicates a high-valued weight (see online version for color).*

Figure 3 illustrates the output of the quadrature algorithm with some two- and three-dimensional examples. In each case the computed quadrature scheme is shown for the interface and the two regions on either side of the interface, based on a Gaussian quadrature order of $q = 4$. One can see how the geometry of the interface determines the choice of height function direction and how the connected components of the resulting implicitly defined domains V_i affect the resulting quadrature scheme.

4. Convergence tests. In this section, we perform convergence studies to assess the accuracy of the quadrature algorithm. In general, for sufficiently smooth problems the algorithm has the same order of accuracy as a Gaussian quadrature scheme of order q. As such, very high convergence rates can be obtained, but it is often the case that these are unobservable due to the limited accuracy imposed by finite precision floating point arithmetic. For example, using double precision, convergence rates of ten or higher are difficult to observe since errors, even on relatively coarse grids, are on the order of machine epsilon. In the following, we aim to nevertheless demonstrate high-order convergence rates. To do so, in some of the following calculations we have made use of the QD library [2], which provides quadruple-double floating point arithmetic routines, having approximately 62 digits of accuracy. It should be made clear,

however, that using high-precision arithmetic is not a requirement of the presented quadrature algorithms—it is only being used to demonstrate high-order convergence rates.

4.1. Surface area and volume. A simple but important test is to confirm that the surface area and volume of an implicitly defined region can be accurately computed. Here we consider an ellipse in two dimensions and an ellipsoid in three dimensions:

- The ellipse has semimajor axis 1, semiminor axis $\frac{1}{2}$, and is defined by the zero level set of $\phi(x, y) = x^2 + 4y^2 - 1$ on the domain $(-1.1, 1.1)^2$. The area of the ellipse is $V = \frac{\pi}{2}$ and the length of its perimeter is $A = 4E(\sqrt{3/4}) = 4.844224\ldots$, where $E(\cdot)$ is the complete elliptic integral of the second kind.
- The ellipsoid has semiprincipal axes 1, $\frac{1}{2}$, and $\frac{1}{3}$ and is defined by the zero level set of $\phi(x, y, z) = x^2 + 4y^2 + 9z^2 - 1$ on the domain $(-1.1, 1.1)^3$. The volume of the ellipsoid is $V = \frac{2\pi}{9}$ and its surface area A is

$$A = \tfrac{2\pi}{3}\left(\tfrac{1}{3} + \sqrt{2}E(\sqrt{\theta}) + \tfrac{1}{4\sqrt{2}}F\left(\theta, \sqrt{\tfrac{5}{8}}\right)\right) = 4.40080956466497\ldots,$$

  where $\theta = \sin^{-1}(\sqrt{8/9})$ and F is the incomplete elliptic integral of the first kind.

The domain is discretized with a uniform Cartesian grid of $n \times n$ cells (in two dimensions) or $n \times n \times n$ cells (in three dimensions) such that $h = 2.2/n$. The quadrature method is then applied to each cell of the Cartesian grid and the results summed to define the following approximations $A_{q,h}$ and $V_{q,h}$ to the surface area and volume:

$$\int_\Gamma 1 \approx A_{q,h} := \sum_i \mathcal{I}(1; (\phi, 0), U_i, \text{true}, q),$$
$$\int_\Omega 1 \approx V_{q,h} := \sum_i \mathcal{I}(1; (\phi, -1), U_i, \text{false}, q),$$

where the summation is over all grid cells U_i. Figure 4 plots the absolute error in surface area and volume, i.e., $|A - A_{q,h}|$ and $|V - V_{q,h}|$, for the two-dimensional and three-dimensional cases, for a selection of different Gaussian quadrature orders $q = 1, 2, 6$, and 10. A line with slope $2q$ is also drawn for each case, showing that the convergence rate of the scheme is approximately $2q$. Since the convergence rate measured between two successive grids is not always consistent (due to the effects of grid alignment), we define an average convergence rate which measures convergence over several grid sizes. Specifically, we define the rate as the slope of the line resulting from simple linear regression on the log-log plot of the errors. Table 1 shows the measured convergence rates for the calculation of surface area and volume of the ellipse and ellipsoid, for $q = 1, 2, \ldots, 10$, using the same range of n as shown in Figure 4. The results show that the approximate convergence rate is at least $2q$ for all q.

To provide an approximate indication of the computational cost of the algorithm, Table 2 measures the time in microseconds needed to construct and evaluate the quadrature scheme,[6] per grid cell, averaged over all cells U_i such that $U_i \cap \Gamma \neq \varnothing$. Although the number of quadrature points per cell is $\mathcal{O}(q^d)$ (in the case of the volume integral) or $\mathcal{O}(q^{d-1})$ (in the case of the surface integral), the timings in Table 2

[6] These timing tests used standard double precision floating point arithmetic.

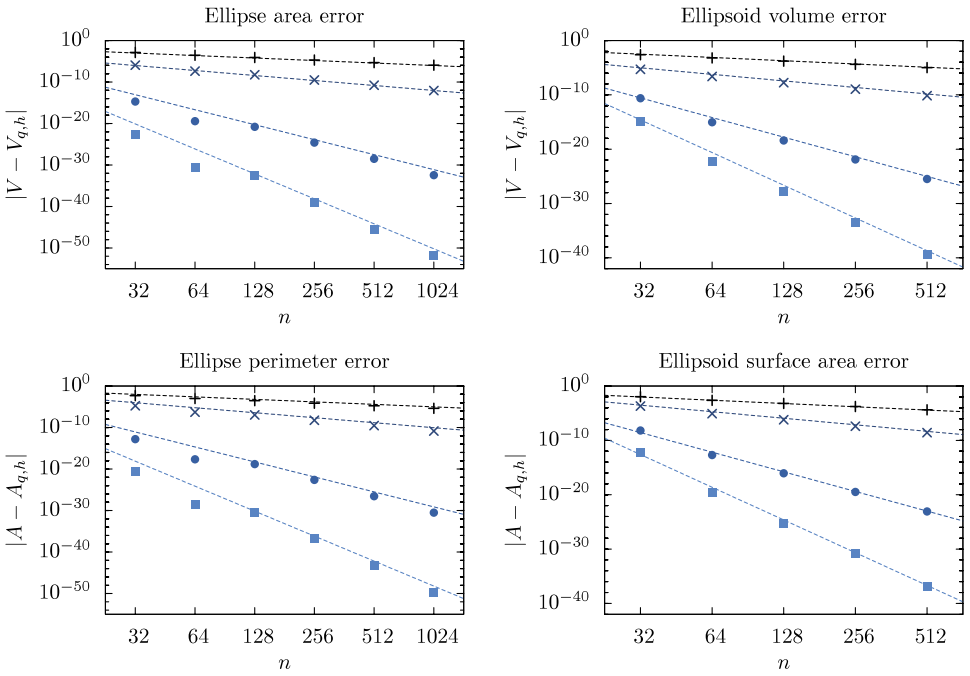


FIG. 4. *Error in the computed area and volume of the ellipse and ellipsoid test problems in section* 4.1 *as a function of grid size* n *for four different values of* q*. The case* $q = 1$ *is denoted in the graphs with* $+$*;* $\times$ *denotes* $q = 2$*;* $\bullet$ *denotes* $q = 6$*; and* $\blacksquare$ *denotes* $q = 10$*. In each case, a line with slope* $2q$ *is drawn in order to illustrate the approximate rate of convergence.*

TABLE 1
Measured convergence rates for the test problems in section 4.1 *for different choices of* q*.*

Test problem	Order q									
	1	2	3	4	5	6	7	8	9	10
Ellipse area	2.0	4.2	6.4	8.5	10.7	12.8	14.9	17.0	19.1	21.2
Ellipse perimeter	2.1	4.3	6.4	8.6	10.7	12.8	15.0	17.1	19.1	21.2
Ellipsoid volume	2.0	4.0	6.0	8.1	10.1	12.1	14.1	16.1	18.1	20.0
Ellipsoid surface area	2.0	4.0	6.1	8.1	10.1	12.2	14.1	16.1	18.1	20.0

TABLE 2
Average time per grid cell (in microseconds) needed to construct and evaluate the quadrature schemes for the test problems in section 4.1*, where the average is taken over all cells intersecting* Γ*. Experimental results obtained on an Intel Core i7* 3.5*GHz desktop machine (single core), based on a* 8192×8192 *grid in two dimensions (2D) and a* $256 \times 256 \times 256$ *grid in three dimensions (3D).*

Test problem, μs per cell	Order q									
	1	2	3	4	5	6	7	8	9	10
2D boundary integral	1.1	1.2	1.3	1.5	1.6	1.7	1.8	1.9	2.1	2.2
2D area integral	1.2	1.5	1.8	2.1	2.4	2.7	3.0	3.4	3.9	4.1
3D surface integral	1.8	2.6	3.8	5.3	7.2	9.4	12	15	18	22
3D volume integral	2.0	3.7	6.0	9.1	13	18	24	31	40	48

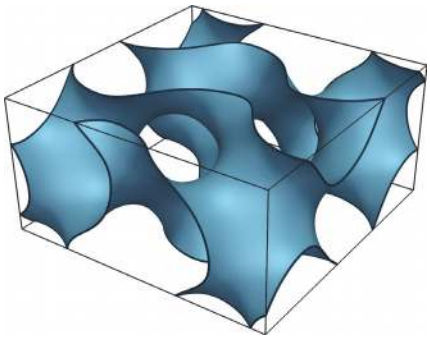


FIG. 5. *A nonclosed surface defined by the zero level set of ϕ in* (4.1)*, confined to the indicated rectangular box and used in the convergence tests of section* 4.2.

do not directly follow this scaling in q. This is because the cost of the algorithm is shared between different aspects: evaluating bounds on the attainable values of ϕ or its gradient (typically $\mathcal{O}(1)$ cost per cell); one-dimensional root finding (typically $\mathcal{O}(q^{d-1})$); and evaluating the integrand or recording the quadrature nodes and weights ($\mathcal{O}(q^d)$ or $\mathcal{O}(q^{d-1})$ for volume or surface integrals, respectively). Generally speaking, the dominant component observed in our experiments is root finding, which typically contributed to at least half the overall time (with a larger percentage for larger q). Hence, if it is possible to reuse a precomputed quadrature scheme, then doing so would likely give a nontrivial improvement in efficiency, rather than recomputing the quadrature scheme. This assumes, however, that the cost of evaluating the integrand is small; if instead the integrand is very costly to evaluate, then this aspect may dominate the overall cost of applying the quadrature method.

4.2. Nonpolynomial functions on a nonclosed surface. The next convergence test verifies that the quadrature scheme retains high-order convergence rates when applied to nonpolynomial integrand and level set functions on a nonclosed surface. A three-dimensional level set function is defined by

$$\phi(x,y,z) = \cos x \sin y + \cos y \sin z + \cos z \sin x \tag{4.1}$$

on the domain $(-L,L)\times(-L,L)\times(-L/2,L/2)$, where $L = 4.25$. Figure 5 illustrates the interface defined by the zero level set of ϕ. The integrand function f is defined by

$$f(x,y,z) = \ln\left(\tfrac{1}{L^2}(x^2+y^2+z^2) + \tfrac{3}{8}\right).$$

Similar to before, we study convergence by discretizing the domain with a uniform Cartesian grid of $n \times n \times n/2$ cells and compute the following approximations to the surface and volume integral:

$$\int_\Gamma f \approx I^s_{q,h} := \sum_i \mathcal{I}(f;(\phi,0),U_i,\text{true},q),$$
$$\int_\Omega f \approx I^v_{q,h} := \sum_i \mathcal{I}(f;(\phi,-1),U_i,\text{false},q).$$

Since the exact answers $\int_\Gamma f$ and $\int_\Omega f$ are unknown, a reference solution is computed by using the same quadrature algorithm on a fine $1024\times1024\times512$ three-dimensional

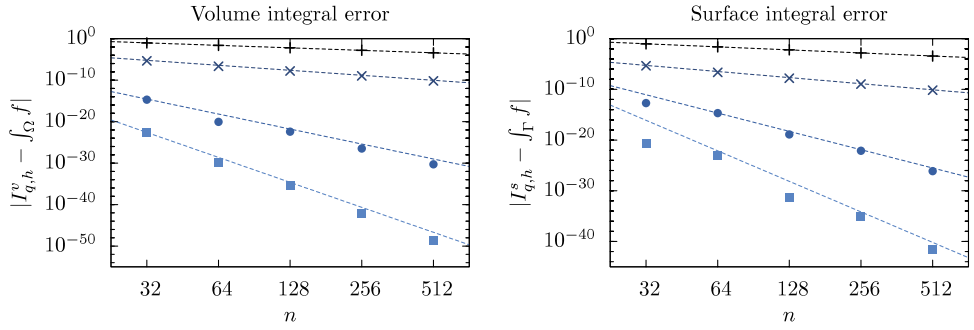


FIG. 6. *Error in the computed surface integral and volume integral of the test problem in section 4.2 as a function of grid size n for four different values of q. The case $q = 1$ is denoted in the graphs with +; × denotes $q = 2$; • denotes $q = 6$; and ■ denotes $q = 10$. In each case, a line with slope $2q$ is drawn in order to illustrate the approximate rate of convergence.*

TABLE 3
Measured convergence rates for the test problems in section 4.2 for different choices of q.

Test problem	Order q									
	1	2	3	4	5	6	7	8	9	10
Volume integral $\int_{\Omega} f$	2.0	3.9	7.2	8.7	10.6	11.5	14.5	16.4	18.8	20.9
Surface integral $\int_{\Gamma} f$	2.0	4.0	6.8	9.4	10.8	12.5	14.3	16.2	18.0	19.8

grid with $q = 10$; convergence tests indicate[7] that

$$\int_{\Gamma} f = 6.897665194490618059924850963768989519102402631696 + \epsilon_1,$$
$$\int_{\Omega} f = 6.261923761662944764662591994149333275702846237971 + \epsilon_2,$$

where $|\epsilon_i| < 10^{-48}$. For a selection of different Gaussian quadrature orders, $q = 1, 2, 6$, and 10, Figure 6 plots the absolute error corresponding to the computed surface integral and volume integrals as a function of the grid size n. As in the previous test problem, we can see that the approximate convergence rate is $2q$. Using the same range of grid sizes, Table 3 shows the measured convergence rates for all orders $q = 1, \ldots, 10$, confirming that convergence rates of approximately $2q$ is obtained for all q.

4.3. Additional convergence tests and analysis. The supplementary material of this paper contains additional convergence tests which examine the quadrature scheme in different scenarios:

- *Approximation of the level set function*: In many applications of implicit interface methods, the level set function is known only at the grid points of a computational grid/mesh (see, e.g., [25, 18, 22, 23, 24]), and therefore must be interpolated in order to apply the quadrature scheme on each mesh element. The supplementary examples show how the overall order of accuracy for the

[7] The first handful of significant digits were also confirmed by using a low-order method based on discrete delta functions and discrete Heaviside functions.

quadrature scheme depends not only on q, but also on the accuracy of the interpolation scheme and regularity of the interface.

- *Detecting and resolving subcell/subgrid features*: An important consideration in connection with high-order implicit interface methods, which are often capable of resolving and maintaining subgrid features in an interface, is whether the quadrature algorithm also detects such features. The supplementary example demonstrates that the subdivision strategy automatically detects and accurately resolves subgrid features, such as a closed surface entirely contained inside a single grid cell.
- *Termination of subdivision strategy*: Although rare, it may be possible for the simple subdivision strategy to never terminate. This situation arises when the level set function becomes degenerate in a grid cell or on one of its faces or edges, and in such cases, the subdivision must be terminated. The supplementary example analyzes this aspect further and examines what errors are incurred in reverting to a low-order method when the subdivision is terminated.

5. Application to a high-order embedded boundary discontinuous Galerkin method. In this section we demonstrate the merits of high-order accuracy by applying the quadrature algorithm to an embedded boundary discontinuous Galerkin method [13] for solving partial differential equations on implicitly defined domains. Similar to "cut-cell"-type methods, the framework avoids the need to generate body-conforming meshes for curved geometry; instead, a simple Cartesian reference grid can be used to automatically generate a finite element mesh. In particular, the approach presented in [13], briefly described here, provides bounds on the conditioning and convergence properties when applied to elliptic PDEs; as such it avoids the ill-conditioning problems which commonly arise with "tiny" cut-cells.

Let $\phi : \mathbb{R}^d \to \mathbb{R}$ be a level set function which implicitly defines a domain $\Omega = \{x : \phi(x) < 0\}$ with boundary $\Gamma = \{x : \phi(x) = 0\}$. Let U be a rectangular domain which encloses all of Ω, and define a Cartesian grid such that $U = \bigcup_i U_i$, where U_i are rectangular cells. Each cell can be classified as either *empty* (those which fall entirely outside Ω), *partial* (those which contain Γ), or *full* (those falling entirely within Ω); see Figure 7 (left). In the embedded boundary method of [13], partial cells which are deemed "small" are combined with nearby cells to form some of the elements of a discontinuous Galerkin finite element method. Many possibilities exist for deciding whether a cell is small; here we use a simple rule which states that a cell is small if its volume intersecting Ω is less than 20% of its total volume. Small cells are merged with a neighboring cell according to the following scheme: Of the cells sharing a face with the small cell, the cell which has the largest volume intersecting Ω is used. The result of this process is a mesh composed of standard rectangular elements, elements with curved boundaries, and elements which have been merged with small cells. Figure 7 (right) shows a two-dimensional example. Note that throughout this process, neither Ω nor its boundary Γ is explicitly constructed; instead the geometry of the curved elements will factor into the weak formulation of the finite element method via quadrature.

Here we use this approach, together with the presented quadrature algorithm, to demonstrate high-order convergence in solving Poisson's equation on a three-dimensional implicitly defined domain. The chosen domain Ω is a torus defined by the level set function $\phi(x, y, z) = (x^2 + y^2 + z^2 + R^2 - r^2)^2 - 4R^2(x^2 + y^2)$, where $R = 0.5,\ r = 0.3$, and we solve Poisson's equation with Neumann boundary condi-

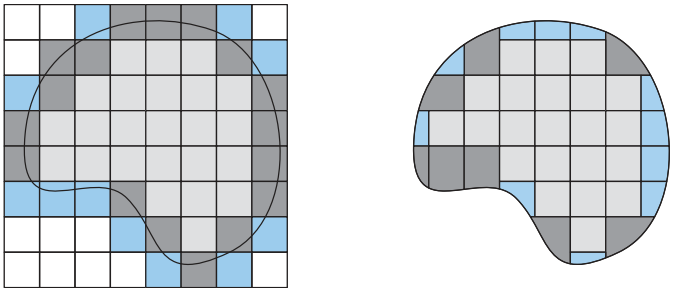


FIG. 7. *(left) The cells of a Cartesian grid enclosing a curved domain are classified according to whether they are entirely outside the domain ("empty," white), entirely within the domain ("full," light gray), or else are denoted as "partial." Partial cells are classified according to whether they have a small intersection with the domain ("small," blue) or a large intersection with the domain (dark gray). (right) Small cells are merged with neighboring cells to form extended curved elements (dark gray), large cells become curved elements cut by the domain boundary (blue), while all other cells are standard reference elements (light gray).*

tions:

$$\begin{aligned} -\Delta u &= f \text{ in } \Omega, \\ \tfrac{\partial u}{\partial n} &= g \text{ on } \Gamma, \\ \textstyle\int_\Omega u &= C, \end{aligned} \tag{5.1}$$

where C is a constant enforcing an integral condition on u to yield uniqueness of the solution. A Cartesian grid with $n \times n \times n$ cells on the rectangular domain $U = (-0.8(1-\frac{1}{n}), 0.8(1+\frac{1}{n}))^3$ is used to generate a mesh according to the above procedure. It follows that most of the elements will be rectangular with width $h \approx 1.6/n$. Define $\mathcal{K}$ to be the set of elements of the resulting mesh, and let $\mathcal{E}_I$ and $\mathcal{E}_N$ denote the set of internal faces and boundary faces, respectively. Let $\mathcal{Q}_p(K)$ denote the space of tensor products of polynomials of degree p on the element K; for example, $\mathcal{Q}_3$ is the space of tricubic polynomials, each one uniquely specified by 64 coefficients. Let V_h be the corresponding space of discontinuous piecewise polynomials on the mesh:

$$V_h = \left\{ u \in L^2(\Omega) : \ u|_K \in \mathcal{Q}_p(K) \text{ for all } K \in \mathcal{K} \right\}.$$

To solve (5.1), a symmetric interior penalty method [1, 7] is used, resulting in the following problem statement: Find $u_h \in V_h$ such that $a(u_h, v_h) = l(v_h)$ for all $v_h \in V_h$, where the bilinear form $a(\cdot,\cdot)$ is given by

$$a(u_h, v_h) = \sum_{K\in\mathcal{K}} (\nabla u_h, \nabla v_h)_K - \sum_{E\in\mathcal{E}_I} \left(\{\!\{\nabla u_h\}\!\}, [\![v_h]\!]\right)_E + \left([\![u_h]\!], \{\!\{\nabla v_h\}\!\}\right)_E + \tau\left([\![u_h]\!], [\![v_h]\!]\right)_E$$

and where the linear functional $l(\cdot)$ is defined by

$$l(v_h) = \sum_{K\in\mathcal{K}} (f, v_h)_K + \sum_{E\in\mathcal{E}_N} (g, v_h)_E.$$

Here, $(\cdot,\cdot)_K$ and $(\cdot,\cdot)_E$ denote the standard L^2 inner products on element K and face E, while $\{\!\{\cdot\}\!\}$ and $[\![\cdot]\!]$ denote the average and jump operators across an internal face E, defined by

$$\{\!\{u\}\!\} = \tfrac{1}{2}(u_1 + u_2), \quad [\![u]\!] = u_1 n_1 + u_2 n_2,$$

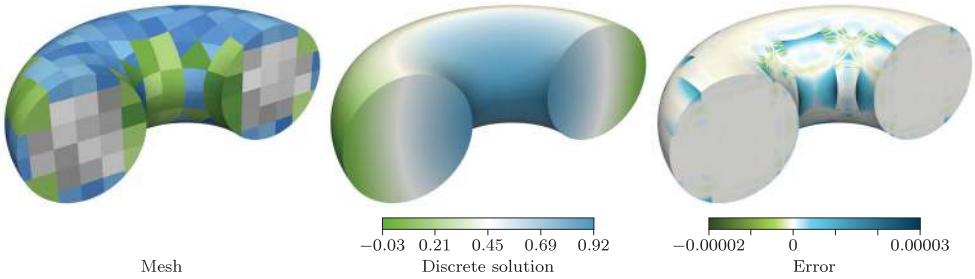


FIG. 8. *Discrete solution of a Poisson problem* (5.1) *inside a torus, computed using the embedded boundary discontinuous Galerkin method described in section* 5 *with order* $p = 3$ *elements on a mesh generated by a relatively coarse* $16 \times 16 \times 16$ *Cartesian grid. Depicted on the left is the corresponding mesh: standard Cartesian grid cell-based elements are shaded gray, extended elements green, and unextended curved elements blue. In the figures, the torus domain has been cut in half in order to reveal the interior.*

where n_1 and n_2 are the exterior pointing unit normal vectors of elements K_1 and K_2 which share the face, with $u_i = u|_{K_i}$. Lastly, $\tau > 0$ is a stabilization penalty parameter associated with the interior penalty method that is required to ensure well-posedness in the discrete formulation, and typically scales according to[8] $\mathcal{O}((p+1)^2/h)$.

We will be brief on implementation details. Evaluation of the bilinear form $a(\cdot,\cdot)$ and linear functional $l(\cdot)$ requires the computation of inner products of polynomials on both elements and faces. In the case where the element is a standard rectangular element, standard quadrature rules can be used that are exact for polynomial integrands (see, e.g., [10]). In the case of a curved element or face, we use the high-order quadrature algorithm developed in this paper. In particular, we must compute the integrals $(\cdot,\cdot)_K$ and $(\cdot,\cdot)_E$ with enough accuracy to account for the curved geometry of the element. As is typical when Jacobian factors are involved (in this case they arise from the curved boundary Γ), this requires the use of a higher-order quadrature method than one would otherwise need on a standard rectangular element. In other words, the value of q used in the quadrature algorithm will necessarily be larger than p. Finally, the weak formulation problem statement—find $u_h \in V_h$ such that $a(u_h, v_h) = l(v_h)$ for all $v_h \in V_h$—leads to a symmetric positive semidefinite linear system for which we have used a simple block-Jacobi preconditioned conjugate gradient method to solve.

We now demonstrate the approach by numerically solving the Poisson problem in (5.1) with data f, g, and C generated by the exact solution $u(x,y,z) = \cos 2x \cos 2y \cos 2z$. Figure 8 shows the computed solution and its error on a mesh generated by a $16 \times 16 \times 16$ Cartesian grid with $p = 3$. To study the order of accuracy, we perform a series of tests using grids of size $n = 16, 32, 64, 128$ with degree $p = 1, 2, 3, 4$ elements and measure the L^2 norm of the error, $(\int_\Omega (u_h - u)^2)^{1/2}$. Table 4 contains the results and also indicates which value of q was used in the quadrature algorithm; these values were chosen empirically as being the smallest possible such that consistent convergence behavior was obtained. The results in Table 4 show that the convergence rate for this embedded boundary discontinuous Galerkin scheme is approximately $p + 1$. This builds upon the work of [13], which provides a theoretical analysis for achieving optimal order accuracy as observed here. An analysis of the role of the quadrature algorithm (such as the parameter q) in regards to the finer details

[8] In the following numerical results, we have used $\tau = C_p/h$, where $C_p = 8$, 18, 32, and 200 for $p = 1$, 2, 3, and 4, respectively.

TABLE 4

L^2 norm of the error of the discrete solution of the Poisson problem (5.1)*, computed using order p elements on a mesh generated by an $n \times n \times n$ grid; associated convergence rates are indicated in boldface.*

n	$p=1$, $q=2$		$p=2$, $q=5$		$p=3$, $q=6$		$p=4$, $q=8$	
16	1.07×10^{-3}	–	2.97×10^{-5}	–	7.02×10^{-7}	–	5.21×10^{-8}	–
32	2.14×10^{-4}	**2.2**	1.91×10^{-6}	**3.8**	2.24×10^{-8}	**4.7**	3.85×10^{-10}	**6.8**
64	5.10×10^{-5}	**2.0**	2.03×10^{-7}	**3.2**	1.10×10^{-9}	**4.2**	1.04×10^{-11}	**5.1**
128	1.24×10^{-5}	**2.0**	2.12×10^{-8}	**3.2**	5.73×10^{-11}	**4.2**	2.53×10^{-13}	**5.3**

of the accuracy of the numerical solution as well as to the spectral properties of the resulting linear system, etc., is the subject of ongoing work.

6. Concluding remarks. The presented algorithm for computing high-order quadrature schemes on implicitly defined surfaces and volumes is based on representing the curved geometry as a graph of an implicitly defined height function. Once a suitable height function direction is determined, both types of integration can be rewritten as an integral over an implicitly defined domain in one fewer spatial dimensions. This leads to a recursive algorithm requiring three main tools: (i) the ability to place bounds on the values attainable by a function inside a given hyperrectangle; (ii) one-dimensional root finding to find the value of the height function at a specific location; and (iii) a one-dimensional numerical quadrature scheme. For (i), the appendix presents a convenient technique based on automatic differentiation which can be used with template programming and operator overloading techniques, though other methods are certainly possible. For (ii), we used a hybrid of the bisection method and Newton's method which was found to work well in practice. For (iii), we used standard Gaussian quadrature methods. Combined, the overall quadrature scheme yields high-order convergence rates: letting q denote the order of the Gaussian quadrature, the algorithm finds a quadrature scheme with approximately $\mathcal{O}(q^{d-1})$ many nodes per cell in the case of the surface integral, approximately $\mathcal{O}(q^d)$ nodes per cell in the case of the volume integral, and, for a sufficiently smooth problem, yields a convergence rate of approximately $2q$.

Several possibilities exist for more sophisticated implementations of the algorithm. The criteria for defining a suitable height function direction could be made more rigorous: our criterion was that the Jacobian factor (characterizing how steep the height function is) must be bounded by a constant. Instead, a criterion based on the smoothness of this factor could be used. We discussed in the development of the algorithm the necessity of subdivision, e.g., in the case that a spherical interface partially crosses a face of a cube, resulting in a circular interface contained entirely within a face. The subdivision strategy which we used was based on the simple method of dividing the hyperrectangle in half along its largest extent. More sophisticated subdivision strategies could be developed that take into account the geometry of the functions ψ_i with the goal of finding a suitable height function direction more quickly. Lastly, in the rare cases that subdivision must be terminated, we used a simple low-order method involving a single point evaluation; other methods could also be used.

Finally, we make some comments about the possibility of extending this approach to different elemental shapes, particularly simplices. Throughout this paper we assumed that U was a hyperrectangle; this fits in naturally with the dimension-reduction algorithm since once a height function direction is found, that coordinate axis can be

removed, leaving a new hyperrectangle on which to perform an integration. Although not presented in this work, the algorithm has also been extended to the case that U is a triangle in two dimensions. The modifications are relatively simple: instead of the height function being constrained by two parallel edges of a box, it is now constrained by two nonparallel edges of the triangle. High-order convergence rates of $2q$ were also obtained for triangles. However, the three-dimensional case for which U is a tetrahedron is more subtle. The reason is that the geometry of the interface restricted to the "upper" face (once a height direction is found) is often considerably different from the geometry on the "lower" face. This in turn makes it difficult to find a height function direction which is suitable for both faces simultaneously. If not suitably addressed, the end result is that an unreasonable number of subdivisions may become necessary. Nevertheless, simplices (particularly reference simplices described by the polytope $u, v, w, \ldots \geq 0, u + v + w + \cdots \leq 1$) do indeed fit into the concept of dimension reduction, since once a coordinate direction is removed, the new object is a simplex in one fewer dimensions. Extending the presented quadrature algorithm to the case of general simplices is the subject of ongoing work.

Appendix. Bounds evaluation via automatic computation of first-order Taylor series with bounded remainder. An important requirement of the quadrature algorithm is the ability to place bounds on the attainable values of a function in a given hyperrectangle U. This was used to assist in finding a suitable height function direction (section 3.2.3) and in the pruning step (section 3.2.1) to remove unnecessary ψ_i functions. It also allows subgrid features to be properly identified and resolved. Here we briefly describe a technique, similar to "automatic differentiation," that can be combined with operator overloading programming methods to automatically calculate a first-order Taylor series with bounded remainder term of typical functions implemented by a computer program.

Let $U = (x_1^L, x_1^U) \times \cdots \times (x_d^L, x_d^U)$ be a hyperrectangle that remains fixed throughout a specific Taylor series computation; we imagine U to be "small" in size. Define $\delta \in \mathbb{R}^d$ to be the half-width of U, i.e., $\delta_i = \frac{1}{2}(x_i^U - x_i^L)$ for $i = 1, \ldots, d$, and define x_c to be the midpoint of U, i.e., $x_{c,i} = \frac{1}{2}(x_i^L + x_i^U)$. A first-order Taylor series with bounded remainder term for a function $f : \mathbb{R}^d \to \mathbb{R}$ centered at the midpoint point x_c in the hyperrectangle U is denoted by the expression $f = (\alpha, \beta, \epsilon)$ for $\alpha, \epsilon \in \mathbb{R}$ and $\beta \in \mathbb{R}^d$ and has the following property:

$$f(x_c + y) = \alpha + \beta \cdot y + r(y) \quad \text{for all } y \in \mathbb{R}^d \text{ such that } |y_i| \leq \delta_i,\ i = 1, \ldots, d,$$

where $r(y)$ is a remainder term which satisfies $r(y) = o(|y|)$ and has the bound $|r(y)| \leq \epsilon$ for all $|y| \leq \delta$. In particular, if $f = (\alpha, \beta, \epsilon)$, then $f(x_c) = \alpha$ and $\nabla f(x_c) = \beta$. We hereafter refer to such an expression for the first-order Taylor series with bounded remainder as a *linearization*. As examples, a linearization for a constant-valued function is simply $(C, 0, 0)$, while the function $(x, y, z) \mapsto x$ has the linearization $(x_{c,1}, (1, 0, 0), 0)$. Operations involving one or more such linearizations can be defined in the natural way. For example,

- addition by a constant $C \in \mathbb{R}$ yields $(\alpha, \beta, \epsilon) + C = (\alpha + C, \beta, \epsilon)$;
- multiplication by a constant $C \in \mathbb{R}$ yields $(\alpha, \beta, \epsilon) \times C = (C\alpha, C\beta, |C|\epsilon)$;
- addition gives $(\alpha_1, \beta_1, \epsilon_1) + (\alpha_2, \beta_2, \epsilon_2) = (\alpha_1 + \alpha_2, \beta_1 + \beta_2, \epsilon_1 + \epsilon_2)$;
- multiplication of two linearizations is somewhat more involved and leads to

$$(\alpha_1, \beta_1, \epsilon_1) \times (\alpha_2, \beta_2, \epsilon_2) = (\alpha_1\alpha_2, \alpha_1\beta_2 + \alpha_2\beta_1, \ell_1\ell_2 + (|\alpha_1| + \ell_1)\epsilon_2 + (|\alpha_2| + \ell_2)\epsilon_1 + \epsilon_1\epsilon_2),$$

where $\ell_i = \sum_{j=1}^d |\beta_{i,j}|\delta_j = |\beta_i| \cdot \delta$;

- division of two linearizations is more complicated and is omitted; it can be derived with the help of a Taylor series with remainder for the function $x \mapsto \frac{1}{1-x}$.

Linearizations of function compositions can also be derived: for a C^2 function $f : \mathbb{R} \to \mathbb{R}$ taking a linearization as its argument, a Taylor series calculation yields

$$f\big((\alpha, \beta, \epsilon)\big) = (f(\alpha), f'(\alpha)\beta, |f'(\alpha)|\epsilon + \tfrac{1}{2}C(\ell + \epsilon)^2),$$

where $\ell = |\beta| \cdot \delta$ and $C \geq \sup_{|\tau|<\ell+\epsilon} |f''(\alpha + \tau)|$. For example,

$$\sin\big((\alpha, \beta, \epsilon)\big) = (\sin(\alpha), \cos(\alpha)\beta, |\cos(\alpha)|\epsilon + \tfrac{1}{2}(\ell + \epsilon)^2).$$

After evaluating a function (such as ψ_i or $\nabla\psi_i$) defined by a chain of the above operations, the final result is a single linearization of that function. The values attainable by the function in the hyperrectangle can then be easily computed: we have for $f = (\alpha, \beta, \epsilon)$,

$$\sup_{x \in U} |f(x) - f(x_c)| = \sup_{x \in U} |f(x_c) + \beta \cdot (x - x_c) + r(x - x_c) - f(x_c)| \leq |\beta| \cdot \delta + \epsilon.$$

It follows that the accuracy of the bounds obtained with the above arithmetic is a function of the size of the hyperrectangle U. In particular, for a linearization arising from a composition of smooth functions, it is typically the case that $\epsilon = \mathcal{O}(|\delta|^2)$. One can verify that each of the above operations (addition, multiplication, function composition, etc.) individually preserves this order of accuracy. The second-order accuracy provided by this approach is sufficient for the quadrature algorithm presented in this paper, e.g., to effectively prune and detect when a function is monotone.

REFERENCES

[1] D. N. Arnold, *An interior penalty finite element method with discontinuous elements*, SIAM J. Numer. Anal., 19 (1982), pp. 742–760.

[2] D. H. Bailey, Y. Hida, X. S. Li, B. Thompson, K. Jeyabalan, and A. Kaiser, *QD Library*, http://crd.lbl.gov/~dhbailey/mpdist/, 2014.

[3] J. U. Brackbill, D. B. Kothe, and C. Zemach, *A continuum method for modeling surface tension*, J. Comput. Phys., 100 (1992), pp. 335–354.

[4] S. L. Chan and E. O. Purisima, *A new tetrahedral tesselation scheme for isosurface generation*, Comput. Graphics, 22 (1998), pp. 83–90.

[5] K. W. Cheng and T.-P. Fries, *Higher-order XFEM for curved strong and weak discontinuities*, Internat. J. Numer. Methods Engrg., 82 (2010), pp. 564–590.

[6] M. Discacciati, A. Quarteroni, and S. Quinodoz, *Numerical approximation of internal discontinuity interface problems*, SIAM J. Sci. Comput., 35 (2013), pp. A2341–A2369.

[7] J. Douglas and T. Dupont, *Interior penalty procedures for elliptic and parabolic Galerkin methods*, in Computing Methods in Applied Sciences, R. Glowinski and J. L. Lions, eds., Lecture Notes in Phys. 58, Springer, Berlin, Heidelberg, 1976, pp. 207–216.

[8] B. Engquist, A.-K. Tornberg, and R. Tsai, *Discretization of Dirac delta functions in level set methods*, J. Comput. Phys., 207 (2005), pp. 28–51.

[9] A. Guéziec and R. Hummel, *Exploiting triangulated surface extraction using tetrahedral decomposition*, IEEE Trans. Visualization Comput. Graphics, 1 (1995), pp. 328–342.

[10] J. S. Hesthaven and T. Warburton, *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*, Texts Appl. Math. 54, Springer, New York, 2008.

[11] D. J. Holdych, D. R. Noble, and R. B. Secor, *Quadrature rules for triangular and tetrahedral elements with generalized functions*, Internat. J. Numer. Methods Engrg., 73 (2008), pp. 1310–1327.

[12] J.-S. Huh and J. A. Sethian, *Exact subgrid interface correction schemes for elliptic interface problems*, Proc. Natl. Acad. Sci. USA, 105 (2008), pp. 9874–9879.

[13] A. Johansson and M. G. Larson, *A high order discontinuous Galerkin Nitsche method for elliptic problems with fictitious boundary*, Numer. Math., 123 (2013), pp. 607–628.
[14] W. E. Lorensen and H. E. Cline, *Marching cubes: A high resolution 3d surface construction algorithm*, Comput. Graphics, 21 (1987), pp. 163–169.
[15] C. Min and F. Gibou, *Geometric integration over irregular domains with application to level-set methods*, J. Comput. Phys., 226 (2007), pp. 1432–1443.
[16] C. Min and F. Gibou, *Robust second-order accurate discretizations of the multi-dimensional Heaviside and Dirac delta functions*, J. Comput. Phys., 227 (2008), pp. 9686–9695.
[17] B. Müller, F. Kummer, and M. Oberlack, *Highly accurate surface and volume integration on implicit domains by means of moment-fitting*, Internat. J. Numer. Methods Engrg., 96 (2013), pp. 512–528.
[18] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Appl. Math. Sci., Springer, New York, 2003.
[19] S. Osher and J. A. Sethian, *Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations*, J. Comput. Phys., 79 (1988), pp. 12–49.
[20] B. A. Payne and A. W. Toga, *Surface mapping brain function on 3d models*, IEEE Comput. Graphics Appl., 10 (1990), pp. 33–41.
[21] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Plannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed., Cambridge University Press, Cambridge, UK, 2007.
[22] R. I. Saye, *High-order methods for computing distances to implicitly defined surfaces*, Comm. Appl. Math. Comput. Sci., 9 (2014), pp. 107–141.
[23] R. I. Saye and J. A. Sethian, *The Voronoi Implicit Interface Method for computing multiphase physics*, Proc. Natl. Acad. Sci. USA, 108 (2011), pp. 19498–19503.
[24] R. I. Saye and J. A. Sethian, *Analysis and applications of the Voronoi Implicit Interface Method*, J. Comput. Phys., 231 (2012), pp. 6051–6085.
[25] J. A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Materials Sciences*, Cambridge University Press, Cambridge, UK, 1999.
[26] J. A. Sethian and P. Smereka, *Level set methods for fluid interfaces*, Ann. Rev. Fluid Mech., 35 (2003), pp. 341–372.
[27] P. Smereka, *The numerical approximation of a delta function with application to level set methods*, J. Comput. Phys., 211 (2006), pp. 77–90.
[28] M. Sussman, P. Smereka, and S. Osher, *A level set approach for computing solutions to incompressible two-phase flow*, J. Comput. Phys., 114 (1994), pp. 146–159.
[29] A.-K. Tornberg and B. Engquist, *Numerical approximations of singular source terms in differential equations*, J. Comput. Phys., 200 (2004), pp. 462–488.
[30] J. D. Towers, *Two methods for discretizing a delta function supported on a level set*, J. Comput. Phys., 220 (2007), pp. 915–931.
[31] G. Ventura, *On the elimination of quadrature subcells for discontinuous functions in the extended finite-element method*, Internat. J. Numer. Methods Engrg., 66 (2006), pp. 761–795.
[32] X. Wen, *High order numerical quadratures to one dimensional delta function integrals*, SIAM J. Sci. Comput., 30 (2008), pp. 1825–1846.
[33] X. Wen, *High order numerical methods to two dimensional delta function integrals in level set methods*, J. Comput. Phys., 228 (2009), pp. 4273–4290.
[34] X. Wen, *High order numerical methods to three dimensional delta function integrals in level set methods*, SIAM J. Sci. Comput., 32 (2010), pp. 1288–1309.
[35] S. Zahedi and A.-K. Tornberg, *Delta function approximations in level set methods by distance function extension*, J. Comput. Phys., 229 (2010), pp. 2199–2219.