High-order Numerical Quadratures in a Tetrahedron with an Implicitly Defined Curved Interface

TAO CUI, WEI LENG, HUAQING LIU, LINBO ZHANG, and WEIYING ZHENG, Chinese Academy of Sciences, China and University of Chinese Academy of Sciences, China

Given a shape regular tetrahedron and a curved surface that is defined implicitly by a nonlinear level set function and divides the tetrahedron into two sub-domains, a general-purpose, robust, and high-order numerical algorithm is proposed in this article for computing both volume integrals in the sub-domains and surface integrals on their common boundary. The algorithm uses a direct approach that decomposes 3D volume integrals or 2D surface integrals into multiple 1D integrals and computes the 1D integrals with Gaussian quadratures. It only requires finding roots of univariate nonlinear functions in given intervals and evaluating the integrand, the level set function, and the gradient of the level set function at given points. It can achieve arbitrarily high accuracy by increasing the orders of Gaussian quadratures, and it does not need extra *a priori* knowledge about the integrand and the level set function. The code for the algorithm is freely available in the open-source finite element toolbox Parallel Hierarchical Grid (PHG) and can serve as a basic building block for implementing 3D high-order numerical algorithms involving implicit interfaces or boundaries.

CCS Concepts:

Additional Key Words and Phrases: Quadrature, tetrahedral mesh, curved surface, extended finite element, high order

ACM Reference format:
Tao Cui, Wei Leng, Huaqing Liu, Linbo Zhang, and Weiying Zheng. 2020. High-order Numerical Quadratures in a Tetrahedron with an Implicitly Defined Curved Interface. *ACM Trans. Math. Softw.* 46, 1, Article 3 (February 2020), 18 pages.
https://doi.org/10.1145/3372144

1 INTRODUCTION

Let Ω_h be a shape-regular tetrahedral mesh for a domain $\Omega \subset \mathbb{R}^3$ and Γ a piecewisely smooth interface in Ω defined by the zero level set of a piecewisely smooth function $L(\mathbf{x})$: $\Gamma = \{\mathbf{x} \in \Omega \mid L(\mathbf{x}) = 0\}$. Let T be a tetrahedron of Ω_h. We make the following assumptions on T and Γ:

This work was supported by the National Key Research and Development Program of China under Grant No. 2016YFB0201304, National Magnetic Confinement Fusion Science Program of China under Grant No. 2015GB110003, National Natural Science Foundation of China under Grants No. 91430215, No. 91530323, No. 11725106, No. 11831016, and No. 11771440, State Key Laboratory of Scientific and Engineering Computing (LSEC), and National Center for Mathematics and Interdisciplinary Sciences of Chinese Academy of Sciences (NCMIS).
Authors' address: T. Cui, W. Leng, H. Liu, L. Zhang, and W. Zheng, Chinese Academy of Sciences, Academy of Mathematics and Systems Science, No. 55, East Zhongguancun Road, Beijing, 100190, China, University of Chinese Academy of Sciences, School of Mathematical Sciences, 19A Yuquan Road, Beijing, 100049, China; emails: {tcui, wleng, liuhq, zlb, zwy}@lsec.cc.ac.cn.

0098-3500/2020/02-ART3 $15.00
https://doi.org/10.1145/3372144

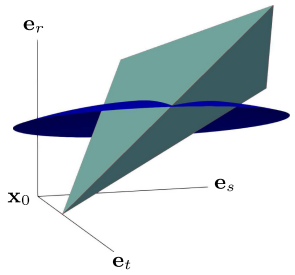


Fig. 1. The three integration directions $\mathbf{e}_r$, $\mathbf{e}_s$, and $\mathbf{e}_t$.

(A1) Γ is smooth in T.

(A2) Γ is locally flat with respect to T, i.e., κh is small where κ denotes the maximum magnitude of the curvatures of Γ in T and h the diameter of T, which means the interface is sufficiently resolved by the tetrahedral mesh.

Our goal is to design a general-purpose, robust, and high-order numerical algorithm to compute the following integrals:

$$I^- = \int_{T\cap\Omega^-} u(\mathbf{x})\,\mathrm{d}\mathbf{x}, \quad I^0 = \int_{T\cap\Gamma} u(\mathbf{x})\,\mathrm{d}\Gamma, \quad I^+ = \int_{T\cap\Omega^+} u(\mathbf{x})\,\mathrm{d}\mathbf{x}, \tag{1}$$

where $\Omega^- := \{\mathbf{x} \mid L(\mathbf{x}) < 0\}$, $\Omega^+ := \{\mathbf{x} \mid L(\mathbf{x}) > 0\}$ and $u(\mathbf{x})$ is a user function that is smooth in T.

Computations of the above types of integrals are often required in the implementations of many numerical algorithms involving an interface that is unfitted with the underlying mesh such as immersed finite element methods [1–4, 13, 14]. We restrict ourselves to the case of tetrahedral meshes in this article, the algorithm can be readily extended to other types of polyhedral meshes in three dimensions.

If the interface is planar within T or only second order accuracy is required (in this case the interface can be locally approximated with a planar one), then the problem can be easily solved by splitting the tetrahedron into subtetrahedra at the intersection points of the interface with the edges of the tetrahedron [5, 6]. However, if the interface is non-planar and higher order accuracy (>2) is required, then the problem becomes very hard. Although many algorithms based on various techniques have been proposed for various types of elements (see, for examples, References [7–12, 14] and the references therein for existing work in the literature), few of them work for tetrahedral elements. Until now there still lacks a general-purpose, robust, and high-order code for computing the integrals in Equation (1), especially when T is a tetrahedron.

The algorithm proposed in this article is based on a simple and direct approach. It consists of choosing a local coordinate system with an origin $\mathbf{x}_0$ and three orthogonal directions, called *integration directions* and represented by three orthonormal vectors $\mathbf{e}_r$, $\mathbf{e}_s$, and $\mathbf{e}_t$, such that

$$T \subset \left\{\mathbf{x}_0 + r\mathbf{e}_r + s\mathbf{e}_s + t\mathbf{e}_t \;\middle|\; r \in (0,a), s \in (0,b), t \in (0,c)\right\},$$

decomposing integrals in Equation (1) into multiple 1D integrals along these directions, and using 1D Gaussian quadratures to compute the 1D integrals; see Figure 1 for an illustration of the local coordinate system. The key point for the algorithm to work is how to deal with singularities in the integrands of the 1D integrals. The proposed algorithm has four main advantages: (1) it only requires finding roots of univariate nonlinear functions in given intervals, (2) the resulting quadrature rules have all positive weights and points strictly inside the integration domain, (3) it can achieve arbitrarily high accuracy by increasing the orders of 1D Gaussian quadratures, and (4) it works without the need of extra *a priori* knowledge about the integrand and the level set function.

The rest of the article is organized as follows. In Section 2 the main ideas and steps of the algorithm are introduced. In Section 3 criteria and methods for determining the integration directions $\mathbf{e}_r$, $\mathbf{e}_s$, and $\mathbf{e}_t$ are described. In Section 4 the computation of the surface integral I^0 is briefly discussed. In Section 5 numerical results are presented. Finally, in Section 6 concluding remarks are given.

2 THE ALGORITHM

We shall use the computation of I^- to describe our algorithm. The computation of I^+ is similar and the procedure for computing I^0 will be briefly discussed in Section 4.

Throughout this article, by smoothness of an univariate function, we mean the function has continuous derivatives up to the desired order to ensure accuracy of the underlying Gaussian quadrature.

First, we give two definitions.

Definition 1. A point $x_0 \in \mathbb{R}$ is said to be a *singularity* of an univariate function $v(x)$ if the function is unsmooth at x_0. It's called a *non-essential singularity* if $\exists\, \varepsilon > 0$ such that $v(x)$ is smooth in both $(x_0 - \varepsilon, x_0]$ and $[x_0, x_0 + \varepsilon)$. Otherwise, it is called an *essential singularity*.

Definition 2. Let F be a face of a tetrahedron. The *trace* of a plane or a surface on F is defined to be its intersection with F, which is a line or a curve on F.

In the proposed algorithm, the integral I^- is computed as

$$\begin{aligned} I^- &= \int_0^c \int_0^b \int_0^a u(r,s,t)\chi^-(r,s,t)\,\mathrm{d}r\,\mathrm{d}s\,\mathrm{d}t \\ &= \int_0^c \int_0^b \int_0^a f(r,s,t)\,\mathrm{d}r\,\mathrm{d}s\,\mathrm{d}t \\ &= \int_0^c \int_0^b g(s,t)\,\mathrm{d}s\,\mathrm{d}t \\ &= \int_0^c h(t)\,\mathrm{d}t, \end{aligned} \tag{2}$$

where $\chi^-(\mathbf{x})$ denotes the characteristic function of $T \cap \Omega^-$, $\chi^-(r,s,t) := \chi^-(\mathbf{x}_0 + r\mathbf{e}_r + s\mathbf{e}_s + t\mathbf{e}_t)$, and $u(r,s,t) := u(\mathbf{x}_0 + r\mathbf{e}_r + s\mathbf{e}_s + t\mathbf{e}_t)$. The 1D integrands in the above integrals are defined as follows:

$$f(r,s,t) := u(r,s,t)\chi^-(r,s,t), \quad g(s,t) := \int_0^a f(r,s,t)\,\mathrm{d}r, \quad h(t) := \int_0^b g(s,t)\,\mathrm{d}s.$$

The 1D integrals in Equation (2) are computed by splitting the integration intervals into subintervals at non-essential singularities of the corresponding integrands such that the integrands are smooth in the subintervals, and applying a Gaussian quadrature rule to each subinterval. It is easy to see that the generated quadrature rule has all quadrature points inside the integration domain and all the quadrature weights are positive. Figure 2 illustrates distributions of the quadrature points on a tetrahedron obtained with the fifth-order (three-point) Gaussian quadrature rule.

The key steps of the algorithm consist of:

(1) choosing the directions $\mathbf{e}_r$, $\mathbf{e}_s$, and $\mathbf{e}_t$ to avoid essential singularities in the 1D integrands,
(2) recursively subdividing the tetrahedron if essential singularities are unavoidable, and
(3) identifying non-essential singularities in the 1D integrands, splitting the 1D integration intervals into subintervals at the non-essential singularities, then applying a Gaussian quadrature rule to each subinterval.

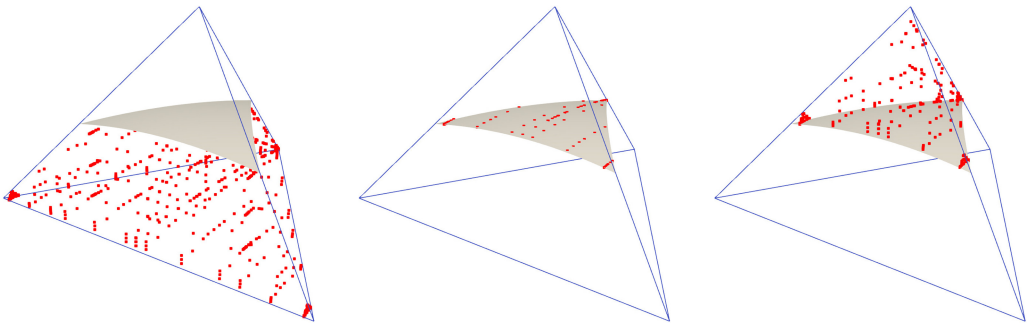


Fig. 2. Examples of quadrature points obtained on a tetrahedron with the fifth-order Gaussian quadrature rule. Left: $T \cap \Omega^-$. Middle: $T \cap \Gamma$. Right: $T \cap \Omega^+$.

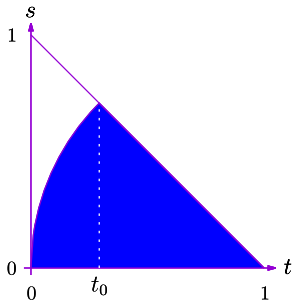


Fig. 3. Illustration of a non-essential singularity at $t = t_0$ and an essential singularity at $t = 0$ in the function $h(t) = \int_{D(t)} g(s,t)\,\mathrm{d}s$, with $g(s,t) \equiv 1$ and $D(t) = \{s \mid s \in (0, 1-t),\ s^2 + (t-1)^2 - 1 < 0\}$.

Before going into details of the algorithm, we first introduce a few notations.

Definition 3. For given s and t, $\mathcal{L}_r(s,t)$ denotes the line segment

$$\{\mathbf{x}_0 + r\mathbf{e}_r + s\mathbf{e}_s + t\mathbf{e}_t \mid r \in [0,a]\}$$

and is called a *r-line*. The *s-line* $\mathcal{L}_s(r,t)$ and the *t-line* $\mathcal{L}_t(r,s)$ are similarly defined.

Definition 4. For given r, $\mathcal{P}_{st}(r)$ denotes the rectangle

$$\{\mathbf{x}_0 + r\mathbf{e}_r + s\mathbf{e}_s + t\mathbf{e}_t \mid s \in [0,b],\ t \in [0,c]\}$$

and is called an *st-plane*. The *rs-plane* $\mathcal{P}_{rs}(t)$ and the *rt-plane* $\mathcal{P}_{rt}(s)$ are similarly defined.

Note that the function $g(s,t)$ is computed by an 1D integral along the r-line $\mathcal{L}_r(s,t)$, while the function $h(t)$ is computed by a double integral over the rs-plane $\mathcal{P}_{rs}(t)$.

We will first summarize types of singularities that may arise in the 1D integrands in Sections 2.1 and 2.2. Then, we will discuss how to choose the integration directions to avoid essential singularities in the 1D integrands in Section 3.

2.1 Non-essential Singularities

Non-essential singularities in $f(r,s,t)$, $g(s,t)$, and $h(t)$ are created by changes in the geometric configuration of the integration domain. A 2D illustration of a non-essential singularity is given in Figure 3 at $t = t_0$, where the element is the standard triangle $\{(s,t) \mid s \geq 0,\ t \geq 0,\ s + t \leq 1\}$, the interface is the unit circle centered at $(1,0)$, the integrand $u(\mathbf{x}) = 1$, and the integration domain is the greyed region.

Below is a complete list of the locations of non-essential singularities for each 1D integrand. They depend on the element T, the interface Γ, and the choice of the local coordinate system.

(1) For given s and t, non-essential singularities in $f(r,s,t)$ may occur at the intersection points of the line segment $\mathcal{L}_r(s,t)$ with
   - the interface Γ and
   - the 4 faces of T.

(2) For given t, non-essential singularities in $g(s,t)$ may occur at the intersection points of the line segment $\mathcal{L}_s(0,t)$ with
   - projections to the plane $\mathcal{P}_{st}(0)$ of the 6 edges of T and
   - projections to the plane $\mathcal{P}_{st}(0)$ of the traces of Γ on the 4 faces of T.

(3) Non-essential singularities in $h(t)$ may occur at the following places:
   - projections to the line segment $\mathcal{L}_t(0,0)$ of the 4 vertices of T and
   - projections to the line segment $\mathcal{L}_t(0,0)$ of the intersection points of the 6 edges of T with Γ.

In the implementation, for each 1D integrand, the non-essential singularities are collected and sorted into an increasing list, and the integration interval is split into subintervals at these locations.

2.2 Essential Singularities

We have identified two kinds of essential singularities as described below. They only exist in the functions $g(s,t)$ and $h(t)$ and are the only kinds we have so far encountered.

(1) An essential singularity of the first kind occurs if the line segment $\mathcal{L}_r(s,t)$ is tangential to Γ at their intersection point.

   A 2D example of this kind of essential singularities is illustrated in Figure 3 at $t = 0$. They can be generally avoided if Γ is relatively flat within T by choosing $\mathbf{e}_r$ close to the normal direction of Γ.

(2) Essential singularities of the second kind only exist in $h(t)$, which occur on a face of T, when the trace of the rs-plane $\mathcal{P}_{rs}(t)$ is tangential to the trace of Γ at their intersection point.

   An example of this kind of essential singularities is shown in Figures 4 and 5. In this example, $\mathbf{w}$ is used in place of $\mathbf{e}_t$ so that the range of t is scaled to $[0,1]$ for convenience of interpretation. The function $h(t)$ (defined in the caption of Figure 5) is smooth in the intervals $[0,t_1]$, $(t_1,t_2]$ and $[t_2,1]$, where $t_1 = (26-\sqrt{262})/24 \approx 0.4089$, which corresponds to the point $\mathbf{p}$ on the face opposite to $\mathbf{v}_0$ where the trace of the plane $\mathcal{P}_{rs}(t_1)$ (the top side of the rectangle in Figure 4) is tangential to the trace of Γ, and $t_2 = (13-\sqrt{31})/16 \approx 0.4645$, which corresponds to the intersection points of Γ with the edges $\mathbf{v}_1$-$\mathbf{v}_3$ and $\mathbf{v}_2$-$\mathbf{v}_3$. An essential singularity in $h(t)$ at $t = t_1$ is clearly shown in Figure 5: $\lim_{t\to t_1^+} h''(t) \to -\infty$.

In the algorithm essential singularities are avoided by proper choice of the local coordinate system, i.e., the integration directions $\mathbf{e}_r$, $\mathbf{e}_s$, and $\mathbf{e}_t$, which will be described in details in Section 3.

3 THE INTEGRATION DIRECTIONS

In this section, we give details of the procedures for determining the integration directions. The main purpose is to prevent essential singularities from appearing in the 1D integrands.

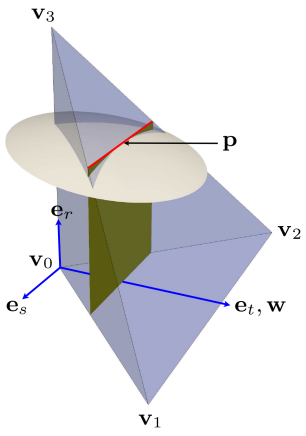


Fig. 4. An illustration of a second kind essential singularity. The four vertices of the tetrahedron are $\mathbf{v}_0 = (\frac{1}{2}, \frac{1}{2}, \frac{5}{8})$, $\mathbf{v}_1 = \mathbf{v}_0 + (\frac{1}{5}, 0, 0)$, $\mathbf{v}_2 = \mathbf{v}_0 + (0, \frac{1}{5}, 0)$ and $\mathbf{v}_3 = \mathbf{v}_0 + (0, 0, \frac{1}{5})$, the interface Γ is the zero level set of $L(\mathbf{x}) = (x - \frac{1}{2})^2 + (y - \frac{1}{2})^2 + (z - \frac{1}{2})^2 - \frac{1}{16}$, and the integration directions are $\mathbf{e}_r = (0, 0, 1)$, $\mathbf{e}_s = (\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}, 0)$ and $\mathbf{e}_t = \mathbf{w}/|\mathbf{w}| = (\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, 0)$, where $\mathbf{w} = (\frac{1}{10}, \frac{1}{10}, 0)$.

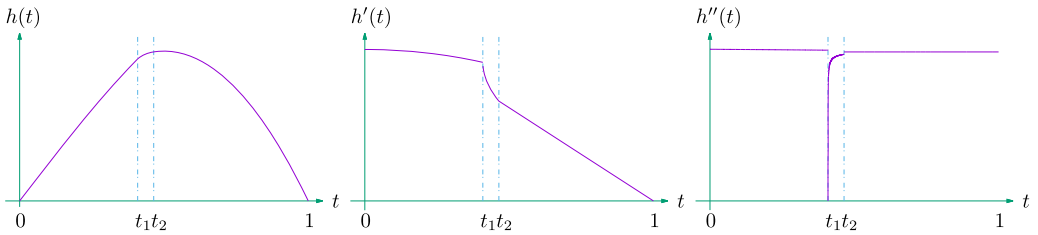


Fig. 5. Plot of $h(t)$ (left), $h'(t)$ (middle), and $h''(t)$ (right), where $h(t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \chi^-(\mathbf{v}_0 + r\mathbf{e}_r + s\mathbf{e}_s + t\mathbf{w})\, \mathrm{d}r\, \mathrm{d}s$, and χ^-, $\mathbf{v}_0$, $\mathbf{e}_t$, $\mathbf{e}_s$, $\mathbf{e}_r$, and so on, are the same as in Figure 4.

3.1 The Direction $\mathbf{e}_r$

To avoid essential singularities of the first kind, the direction $\mathbf{e}_r$ should be close to the normal direction of the interface. This is similar to the strategy used in Reference [7] for choosing integration directions. The interface can then be regarded as the graph of an implicitly defined "height" function in this case.

We have implemented two alternatives for determining $\mathbf{e}_r$ in the algorithm:

(1) $\mathbf{e}_r$ is set to the direction $\nabla L(\mathbf{c})$, where $\mathbf{c}$ is the barycenter of the element, which is an approximation to the normal of Γ.
(2) $\mathbf{e}_r$ is the best approximation to $\nabla L(\mathbf{c})$ among the four unit normal vectors of the 4 faces of T. This choice slightly simplifies the implementation, since $\mathbf{e}_s$ and $\mathbf{e}_t$ are aligned with one face. With this choice, we have so far failed to get exponential decay of the quadrature errors with respect to the orders of the Gaussian quadrature rules used for the 1D integrals.

The first choice is the default one in our code. All the numerical results presented in Section 5 were obtained with it. The second choice is kept in the code only for future investigations.

3.2 The Directions $\mathbf{e}_s$ and $\mathbf{e}_t$

Once $\mathbf{e}_r$ is determined, $\mathbf{e}_s$ and $\mathbf{e}_t$ can be parameterized with a scalar parameter $w \in [-1, 1]$ as follows:

$$\begin{aligned} \mathbf{w}_w &= w\mathbf{e}_1 + \sqrt{1-w^2}\mathbf{e}_2, \\ \mathbf{e}_t &= \mathbf{w}_w/|\mathbf{w}_w|, \\ \mathbf{e}_s &= \mathbf{e}_r \times \mathbf{e}_t, \end{aligned} \tag{3}$$

where $\mathbf{e}_1$ and $\mathbf{e}_2$ are two arbitrarily chosen and linearly independent vectors orthogonal to $\mathbf{e}_r$. Second kind essential singularities are avoided by choosing a suitable value of w, which is done through introducing an indicator function $\alpha(w)$ and setting $w = w_{\text{opt}} := \text{argmin}_w\, \alpha(w)$, as explained below.

3.2.1 The Indicator Function $\alpha(w)$. Denote by $\mathcal{F}(T)$ the set of faces of T. For $F \in \mathcal{F}(T)$, denote its unit normal vector by $\mathbf{n}_F$ and the trace of Γ on F by $\mathcal{T}_\Gamma(F)$. $\mathcal{T}_\Gamma(F)$ is empty if Γ does not intersect with F, or it may contain multiple segments. When no ambiguity arises, we use $\mathcal{T}_\Gamma(F)$ to denote either the set of points on the trace or the set of continuous segments of the trace. Given a point $\mathbf{p}$ on $\mathcal{T}_\Gamma(F)$, denote the unit tangent vector of $\mathcal{T}_\Gamma(F)$ at $\mathbf{p}$ by $\mathbf{t}_F(\mathbf{p})$.

As discussed in Section 2.2, second kind essential singularities occur if the trace of the rs-plane $\mathcal{P}_{rs}(t)$ is parallel to the trace of Γ on some face of T. Let F be a face of T. Since $\mathbf{e}_t$ is an unit normal vector of $\mathcal{P}_{rs}(t)$, the direction of the trace of $P_{rs}(t)$ on F is given by $\mathbf{e}_t \times \mathbf{n}_F$. Let $\mathbf{p}$ be a point on $\mathcal{T}_\Gamma(F)$, define

$$\beta_{\mathbf{p}}(w) := (\mathbf{e}_t \times \mathbf{n}_F) \cdot \mathbf{t}_F(\mathbf{p}).$$

Note, in the right-hand side of the above equation only $\mathbf{e}_t = \mathbf{w}_w/|\mathbf{w}_w|$ depends on w. $\beta_{\mathbf{p}}(w)$ has values in $[-1, 1]$. It equals to the cosine of the angle between the trace of $\mathcal{P}_{rs}(t)$ and $\mathcal{T}_\Gamma(F)$ at the point $\mathbf{p}$ for given value of w. If $\beta_{\mathbf{p}}(w) = \pm 1$, then the two traces are parallel at the point $\mathbf{p}$ for the given value of w.

Let F be a face of T. Let $\mathbf{p}$ and γ denote, respectively, a point and a segment of $\mathcal{T}_\Gamma(F)$. For convenience of discussions, we introduce the following functions:

$$\begin{aligned} \alpha_{\mathbf{p}}(w) &:= |\beta_{\mathbf{p}}(w)|, \\ \alpha_\gamma(w) &:= \max_{\mathbf{p}\in\gamma} \alpha_{\mathbf{p}}(w), \\ \alpha_F(w) &:= \max_{\mathbf{p}\in\mathcal{T}_\Gamma(F)} \alpha_{\mathbf{p}}(w) = \max_{\gamma\in\mathcal{T}_\Gamma(F)} \alpha_\gamma(w). \end{aligned}$$

With the above notations, the indicator function $\alpha(w)$ is defined as

$$\alpha(w) := \max_{F\in\mathcal{F}(T)} \alpha_F(w). \tag{4}$$

For given w, the function $\alpha(w)$ measures the minimum angle between the trace of $\mathcal{P}_{rs}(t)$ and the trace of Γ on all faces of T. It is defined for $w \in [-1, 1]$ and has its range included in $[0, 1]$. Since $\alpha(-1) = \alpha(1)$, it can be continuously extended to a periodic function in $\mathbb{R}$. If $\alpha(w) = 1$, then the trace of $\mathcal{P}_{rs}(t)$ is parallel to the trace of Γ at some points on some faces, which means second kind essential singularities exist in the integrand $h(t)$. If $\alpha(w) = 0$ (which may only happen if Γ is flat on all faces of T), then the trace of $\mathcal{P}_{rs}(t)$ is always orthogonal to the trace of Γ on all faces of T.

To avoid second kind essential singularities, we need to choose a value of w such that $\alpha(w) \neq 1$, or equivalently $\alpha(w) < 1$. It is intuitively true and also observed in numerical experiments that the numerical quadrature error of our algorithm is positively correlated with the value of $\alpha(w)$. For examples, in the graphs in Figure 6, the numerical quadrature errors obtained using our algorithm with different values of w are plotted against the values of $\alpha(w)$ for three randomly chosen different

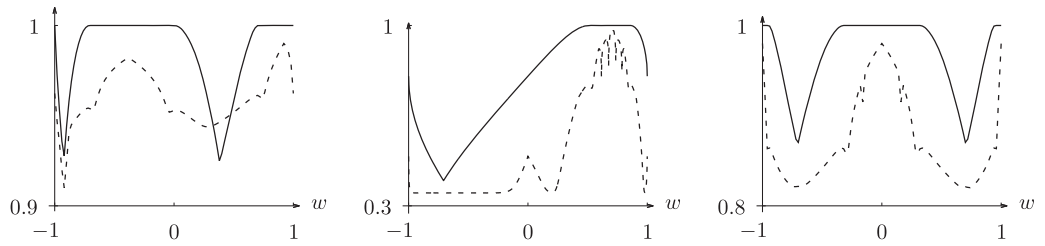


Fig. 6. Plots of $\alpha(w)$ and numerical quadrature errors as functions of w for three different tetrahedra, with each graph corresponding to a tetrahedron. The function $\alpha(w)$ is represented by solid lines, while the logarithm of the absolute values of the quadrature errors, suitably scaled and translated, are represented by dashed lines. The values of $\alpha(w)$ are marked on the y-axis. The quadrature errors are in the ranges $[10^{-16}, 10^{-7}]$, $[10^{-16}, 10^{-4}]$, and $[10^{-11}, 10^{-5}]$ for the three test cases, respectively.

tetrahedra, they show that larger value of $\alpha(w)$ usually indicates larger numerical quadrature error. Thus, $w = w_{\text{opt}} := \text{argmin}_w \alpha(w)$ seems to be a good choice and is used in our algorithm.

3.2.2 The Inadmissible Intervals. Computation of w_{opt} is nontrivial, since the function $\alpha(w)$ is usually unsmooth and has multiple local minima, as can be seen in Figure 6. To cope with this difficulty, we introduce the concept of *inadmissible values* and *inadmissible intervals*, which are to be excluded from the search region.

Definition 5.

(1) A value of w is called *inadmissible* if $\alpha(w) = 1$. An interval of inadmissible values is called an *inadmissible interval*.
(2) Let $F \in \mathcal{F}(T)$, w is said to be *inadmissible for* F if $\alpha_F(w) = 1$.
(3) Similarly, let γ be a segment of $\mathcal{T}_\Gamma(F)$, w is said to be *inadmissible for* γ if $\alpha_\gamma(w) = 1$.

Inadmissible intervals can be determined using the following procedure.

Let F be a face of T. Suppose $\mathcal{T}_\Gamma(F)$ is not empty and let $\gamma \in \mathcal{T}_\Gamma(F)$ be a continuous segment that has two end points on ∂F. Denote the two end points of γ by $\mathbf{p}_0$ and $\mathbf{p}_1$, respectively. Let γ be parameterized as $\mathbf{p}(u)$ for $u \in [0, 1]$ such that $\mathbf{p}(0) = \mathbf{p}_0$, $\mathbf{p}(1) = \mathbf{p}_1$ and $\mathbf{p}'(u) \neq 0$, $\forall u \in [0, 1]$. Define

$$\phi(u) := \Big(\mathbf{p}'(u) \times (\mathbf{e}_t \times \mathbf{n}_F)\Big) \cdot \mathbf{n}_F.$$

The sign of $\phi(u)$ indicates on which side of the vector $\mathbf{e}_t \times \mathbf{n}_F$ lies the vector $\mathbf{p}'(u)$, on the face F.

Lemma 1. *Suppose F is a face of T and γ is a continuous segment of $\mathcal{T}_\Gamma(F)$, which has two end points on ∂F. Let $\mathbf{p}(u)$ and $\phi(u)$ be defined as above. Then, w is inadmissible for γ if and only if the function $\phi(u)$ has roots in $[0, 1]$.*

Proof. Since $\mathbf{t}_F(\mathbf{p}(u)) = \pm\mathbf{p}'(u)/|\mathbf{p}'(u)|$, w is inadmissible for γ if and only if $\mathbf{p}'(u)$ is parallel to $\mathbf{e}_t \times \mathbf{n}_F$ for some $u \in [0, 1]$, or equivalently $\mathbf{p}'(u) \times (\mathbf{e}_t \times \mathbf{n}_F) = 0$ for some $u \in [0, 1]$. Since both $\mathbf{p}'(u)$ and $\mathbf{e}_t \times \mathbf{n}_F$ are parallel to F, so $\mathbf{p}'(u) \times (\mathbf{e}_t \times \mathbf{n}_F)$ is parallel to $\mathbf{n}_F$, thus $\mathbf{p}'(u) \times (\mathbf{e}_t \times \mathbf{n}_F) = 0$ is equivalent to $(\mathbf{p}'(u) \times (\mathbf{e}_t \times \mathbf{n}_F)) \cdot \mathbf{n}_F = 0$, i.e., $\phi(u) = 0$, which proves the lemma. □

Lemma 2. *Under the same assumptions as in Lemma 1, we have:*

1. *A sufficient condition for w being inadmissible for γ is $\phi(0)\phi(1) \leq 0$, or*

$$\Big(\Big(\mathbf{p}'(0) \times (\mathbf{e}_t \times \mathbf{n}_F)\Big) \cdot \mathbf{n}_F\Big)\Big(\Big(\mathbf{p}'(1) \times (\mathbf{e}_t \times \mathbf{n}_F)\Big) \cdot \mathbf{n}_F\Big) \leq 0. \tag{5}$$

2. *If γ is convex and relatively flat, then inequality Equation (5) is also a necessary condition for w being inadmissible for γ.*

Proof. The statement 1 follows directly from Lemma 1 and the Intermediate Value Theorem.

Let $\theta(u)$ be the angle between $\mathbf{p}'(u)$ and $\mathbf{e}_t \times \mathbf{n}_F$. Since γ is convex, $\theta(u)$ is a monotone function of u. If $\theta(u) = \pm\frac{\pi}{2}$ for some $u \in [0,1]$, then since γ is relatively flat, $\theta(u)$ varies in a small neighbour of $\pm\frac{\pi}{2}$, so $\phi(u) \neq 0$, $\forall u \in [0,1]$. Otherwise, $\theta(u)$ is bounded away from $\pm\frac{\pi}{2}$, thus $\phi(u)/|\mathbf{p}'(u)| = \pm\sin\Big(\theta(u)\Big)$ is a monotone function of u. It is not difficult to see that the statement 2 holds in both cases. □

Substituting the expression for $\mathbf{e}_t$ in Equation (3) into the inequality in Equation (5), we get

$$\left(a_0 w + b_0\sqrt{1-w^2}\right)\left(a_1 w + b_1\sqrt{1-w^2}\right) \leq 0, \tag{6}$$

where $a_i = (\mathbf{p}'(i) \times (\mathbf{e}_1 \times \mathbf{n}_F)) \cdot \mathbf{n}_F$ and $b_i = (\mathbf{p}'(i) \times (\mathbf{e}_2 \times \mathbf{n}_F)) \cdot \mathbf{n}_F$, $i = 0, 1$. Assume $w \neq 0$. Dividing the inequality in Equation (6) by w^2 and letting $\delta := \frac{\sqrt{1-w^2}}{w}$, we get the following quadratic inequality for δ:

$$(a_0 + b_0\delta)(a_1 + b_1\delta) \leq 0.$$

The above inequality can be easily solved to get intervals for δ, they are then converted to inadmissible intervals for w.

The above steps for finding inadmissible intervals are performed with each continuous segment of $\mathcal{T}_\Gamma(F)$ on each face F of T. The complement of the union of all the inadmissible intervals thus found, which is also composed of none or several intervals, is called the *candidate intervals*.

We only need to search for w_{opt} in the interior of the candidate intervals. It is observed that in each candidate interval $\alpha(w)$ is often an unimodal function, so w_{opt} can be computed by performing a few search steps using the Fibonacci search method within each candidate interval.

Remark 1. By Lemma 2, if Γ is convex and $\forall F \in \mathcal{F}(T)$, $\forall \gamma \in \mathcal{T}_\Gamma(F)$, γ has two end points on ∂F and is relatively flat such that the statement 2 of Lemma 2 holds, then the inadmissible intervals found using the procedure above contain all the inadmissible values and the candidate intervals contain only (and all the) admissible values. Otherwise, the inadmissible intervals found may only cover a subset of the inadmissible values.

3.2.3 Evaluation of $\alpha(w)$. If w belongs to one of the inadmissible intervals found using the procedure described above, then $\alpha(w) = 1$. So $\alpha(w)$ only needs to be evaluated in the candidate intervals.

Suppose w belongs to one of the candidate intervals. The evaluation of $\alpha(w)$ as defined in Equation (4) requires finding the extrema of univariate functions involving the first derivatives of the level set function $L(\mathbf{x})$, which in turn requires finding the roots of univariate functions involving the second derivatives of $L(\mathbf{x})$. To simplify the evaluation of $\alpha(w)$ and to avoid computing the second derivatives of $L(\mathbf{x})$, $\forall F \in \mathcal{F}(T)$, $\alpha_F(w)$ is approximated by sampling a few points on $\mathcal{T}_\Gamma(F)$. In the actual implementation, $\alpha_F(w)$ is approximately computed by replacing $\mathcal{T}_\Gamma(F)$ with $\mathcal{I}(F)$:

$$\alpha_F(w) \approx \max_{\mathbf{p} \in \mathcal{I}(F)} \alpha_{\mathbf{p}}(w), \tag{7}$$

where $\mathcal{I}(F)$ denotes the set of intersection points of Γ with the edges of F.

Remark 2. Under certain conditions the approximation in Equation (7) is exact. For example, if $\forall \gamma \in \mathcal{T}_\Gamma(F)$, γ has two end points on ∂F and is convex and relatively flat such that the statement 2 of Lemma 2 holds, then it is not difficult to verify that $\beta_{\mathbf{p}}(w)$, $\mathbf{p} \in \gamma$, is a monotone function along

γ if w belongs to one of the candidate intervals, so in this case $\alpha_\gamma(w) = \max\{|\beta_{\mathbf{p}_0}(w)|, |\beta_{\mathbf{p}_1}(w)|\}$, where $\mathbf{p}_0$ and $\mathbf{p}_1$ denote the two end points of γ, and Equation (7) becomes an equality.

In the present implementation $\mathcal{T}_\Gamma(F)$, if not empty, is only allowed to have two intersection points with ∂F, which means $\mathcal{T}_\Gamma(F)$ is always composed of one continuous segment with two end points on ∂F. If on some face F, $\mathcal{T}_\Gamma(F)$ has more than two intersection points with ∂F, or $\mathcal{T}_\Gamma(F)$ is a closed curve and has no intersection with ∂F, then $\alpha(w)$ is set to 1 for all w, which will cause the tetrahedron T to be subdivided.

3.2.4 A Summary on the Computation of $\mathbf{e}_s$ *and* $\mathbf{e}_t$. Now, we give the full steps for computing $\mathbf{e}_s$ and $\mathbf{e}_t$.

(1) Determine the inadmissible intervals and construct the candidate intervals.
(2) If the set of candidate intervals is empty, then subdivide T.
(3) Compute $w_{\text{opt}} := \text{argmin}_w \alpha(w)$ by performing a few search steps using the Fibonacci search method in each candidate interval.
(4) If $\alpha(w_{\text{opt}}) > A$, then subdivide T.
(5) Compute $\mathbf{e}_s$ and $\mathbf{e}_t$ according to Equation (3) with $w = w_{\text{opt}}$.

In the steps above, "subdivide T" means divide T into 2 or 8 subtetrahedra, using either the longest edge bisection [15] or the regular refinement [16], and recursively apply the whole algorithm to all the subtetrahedra. $A \in (0, 1)$ is a predefined threshold whose default value in our code is $A = 0.95$. We don't have guarantee on the finite termination of recursive subdivisions. Infinite recursions are simply avoided by using a planar approximation of the interface [5, 6] and quadrature rules for triangles and tetrahedra [17] when the tetrahedron is very small, e.g., with edge lengths approaching the machine precision. Numerical experiments show that the recursions are rare if the interface is well resolved by the mesh and thus have no noticeable impact on the overall accuracy and performance of the algorithm.

Remark 3. Apart from those discussed in this section, there may be other factors that influence the accuracy and performance of the algorithm. For example, numerical quadrature errors might be reduced by choosing the integration directions in a way to avoid them to be nearly parallel to a face of T. They are left as subjects of future studies.

4 SURFACE INTEGRAL

Computation of the surface integral I^0 is similar and is simpler than the computation of the volume integral. Denote $\mathbf{x}(r, s, t) := \mathbf{x}_0 + r\mathbf{e}_r + s\mathbf{e}_s + t\mathbf{e}_t$. We construct the integration directions $\mathbf{e}_r$, $\mathbf{e}_s$, and $\mathbf{e}_t$, determine the parameters for the integration domain $\mathbf{x}_0$, b, and c, and compute I^0 by

$$I^0 = \int_{T\cap\Gamma} u(\mathbf{x})\,\mathrm{d}\Gamma = \int_0^c \int_0^b \tilde{g}(s,t)\,\mathrm{d}s\,\mathrm{d}t = \int_0^c \tilde{h}(t)\,\mathrm{d}t,$$

just as for the volume integral, where

$$\tilde{g}(s,t) := \begin{cases} u(r_0,s,t)\dfrac{\left|\nabla L\left(\mathbf{x}(r_0,s,t)\right)\right|}{\left|\mathbf{e}_r \cdot \nabla L\left(\mathbf{x}(r_0,s,t)\right)\right|}, & \text{if } \exists r_0, \text{ s. t. } \mathbf{x}(r_0,s,t) \in T \text{ and } L(\mathbf{x}(r_0,s,t)) = 0, \\ 0, & \text{otherwise,} \end{cases}$$

$$\tilde{h}(t) := \int_0^b \tilde{g}(s,t)\,\mathrm{d}s.$$

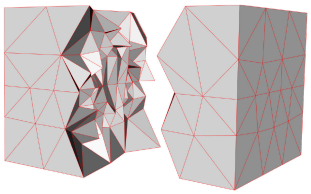
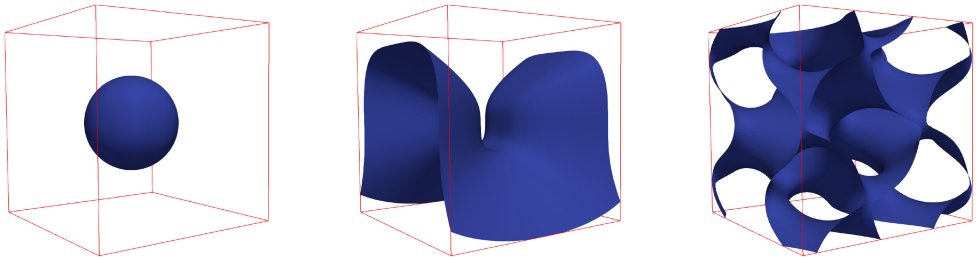


Fig. 7. The mesh containing 1,843 tetrahedra.



(a) (b) (c)

Fig. 8. The three test cases.

5 NUMERICAL RESULTS

The algorithm presented in this article has been implemented as a general purpose function for computing various types of integrals in a tetrahedron cut by a curved interface in the open-source parallel adaptive finite element toolbox PHG [19], which can be used in high-order extended finite element methods (XFEM) or other immersed interface or immersed boundary algorithms on tetrahedral meshes, by both PHG and non-PHG users. See `doc/quad-XFEM.pdf` in the newest distribution of PHG for related documentations.

The numerical experiments presented in this section were carried out on the cluster Lenovo DeepComp 8800 of the State Key Laboratory of Scientific and Engineering Computing, Chinese Academy of Sciences, which is equipped with dual Intel Gold 6140 CPU nodes (2×18 cores, 2.30 GHz) and 100 Gbps EDR Infiniband network.

5.1 Robustness and Accuracy Tests

The numerical results presented in this subsection were obtained with the test program `quad_test2.c`, which is available in the PHG package.

In this set of numerical experiments, Ω is the unit cube $(0,1)^3$, the mesh is an unstructured and irregular tetrahedral mesh containing 1,843 elements as shown in Figure 7, and the integrand $u(\mathbf{x}) \equiv 1$. In this case the sum of I^- over all elements equals to the volume of Ω^- and the sum of I^0 over all elements equals to the surface area of $\Gamma \cap \Omega$. We have tested the algorithm with the following three interfaces, which are illustrated in Figure 8:

(a) The first interface is a sphere of radius $\frac{1}{4}$ centered at the center of Ω, defined by the level-set function:

$$L(x,y,z) = \left(x-\frac{1}{2}\right)^2 + \left(y-\frac{1}{2}\right)^2 + \left(z-\frac{1}{2}\right)^2 - \frac{1}{16}.$$

The test program for this case can be compiled in the `test/` subdirectory of PHG's source tree with the following command:

```
make USER_CFLAGS="-DCASE=0" quad_test2
```

(b) The level-set function for the second interface is the following order 4 polynomial:

$$L(x,y,z) = 2\left(x-\frac{1}{2}\right)^2 - 8\left(y-\frac{1}{2}\right)^3 - 16\left(z-\frac{1}{2}\right)^4 - \frac{1}{50}.$$

The test program for this case can be compiled in the `test/` subdirectory of PHG's source tree with the following command:

```
make USER_CFLAGS="-DCASE=3" quad_test2.
```

(c) The level-set function for the third interface is an order 5 piecewise polynomial that is the interpolation in the P_5 Lagrangian finite element space of the following nonpolynomial function[1]:

$$\cos(4\gamma x - 2\gamma)\sin(4\gamma y - 2\gamma) + \cos(4\gamma y - 2\gamma)\sin(2\gamma z - \gamma) + \cos(2\gamma z - \gamma)\sin(4\gamma x - 2\gamma),$$

with $\gamma = \frac{19}{8}$. This example is similar to the "gyroid" examples in References [7, 8]. To better resolve the interface, the elements near the interface are refined three times using PHG's newest-vertex bisection scheme [18] (refining a tetrahedron three times with newest vertex bisection would bisect all its edges and split it into 8 sub-tetrahedra) and the refined mesh containing 14,621 tetrahedra is used in the test. The test program for this case can be compiled in the `test/` subdirectory of PHG's source tree with the following command:

```
make USER_CFLAGS="-DCASE=5 -DLS_ORDER=5 -DREFINE=3" quad_test2.
```

All the results presented in this subsection were obtained using quadruple-precision floating-point operations (GNU C's `__float128` type, with a precision of `1.9259e-34`).

5.1.1 p-convergence Tests. In the p-convergence tests, the mesh is fixed and the decay rates of relative quadrature errors with respect to increasing quadrature orders are observed. In all the numerical examples presented, the term "quadrature order" will refer to the order of Gaussian quadrature rules used to compute the 1D integrals. Since we do not know the analytical results for cases (b) and (c), the errors for quadrature order p are computed using the differences between the results of quadrature orders $p-2$ and p for all three cases.

The results are shown in Figure 9. Exponential convergence can be observed for all three cases.

5.1.2 h-convergence Tests. In the h-convergence tests, the quadrature order is fixed and relative quadrature errors are computed on meshes that are obtained by successive refinements towards the interface using the newest vertex bisection scheme. Since the computations were quite expensive (the largest refined mesh contained over 25 billion tetrahedra, see Table 1, and the computations were carried out using 256 nodes of the cluster), we have only performed computations of the volume integral I^- for case (a). For this case the analytic result is known as $\frac{4}{3}\pi r^3$ with $r = \frac{1}{4}$ and was used to compute the errors.

Relative quadrature errors with respect to refinement levels for orders $p = 3$, 5, 7, and 9 are shown in Figure 10 and Table 1. The optimal convergence rate h^{p+1} is obtained for $p = 3$ and for $p = 5$ on levels ≥ 18. For higher orders, the optimal convergence is not convincing at present and will be left for future investigations.

[1]The current implementation supports nonpolynomial level set functions by using a combined univariate root-finding algorithm based on polynomial approximation, Newton's method, and bisections. But for nonpolynomial functions the present univariate root-finding code is not robust enough to allow very high accuracy tests, so a piecewise polynomial approximation is used in this example.

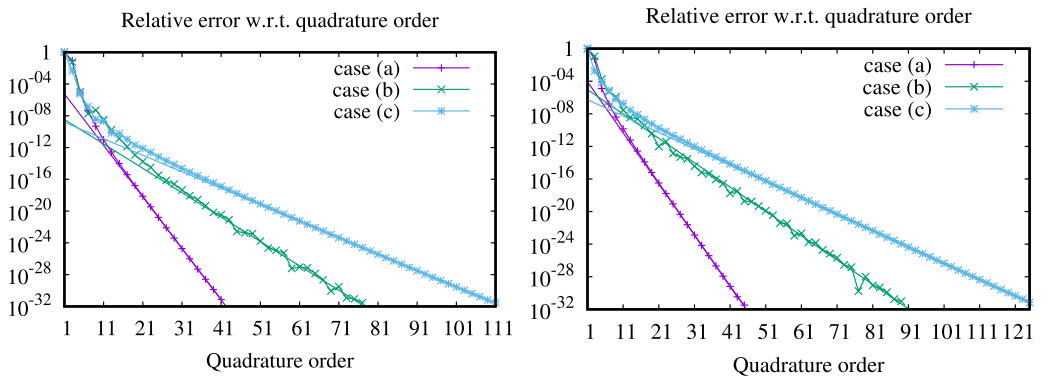


Fig. 9. Relative quadrature errors with respect to quadrature orders (p-convergence). The straight lines represent fitted functions of the form $\lambda e^{-\theta p}$, where p = order and λ and θ denote positive constants. Left: volume integral. Right: surface integral.

Table 1. Quadrature Errors with Respect to Refinement Levels

Refinement level	Number of elements	Relative quadrature errors			
		Order 3	Order 5	Order 7	Order 9
0	1,843	9.3051e-06	4.4160e-08	4.8823e-10	1.0003e-11
3	14,052	7.2981e-07	1.2092e-09	5.8041e-12	2.6140e-14
6	82,771	8.0094e-08	1.6479e-10	2.3656e-12	3.3632e-14
9	380,664	1.0089e-08	3.3505e-11	1.3683e-12	9.2433e-14
12	1,583,742	1.0707e-09	1.0066e-12	9.6012e-15	2.4311e-16
15	6,342,623	5.0719e-11	1.4829e-14	3.0699e-16	1.1571e-17
18	25,319,490	1.6160e-12	4.6455e-15	4.1453e-17	5.3285e-19
21	101,119,218	1.3736e-13	8.9861e-19	4.9715e-20	1.8811e-22
24	404,017,892	1.0111e-14	7.8115e-19	2.2411e-21	2.7504e-23
27	1,614,901,623	3.7069e-16	5.7609e-21	2.2468e-23	1.4213e-25
30	6,457,307,207	2.4504e-17	3.9368e-21	9.8243e-24	3.4002e-26
33	25,825,097,019	3.3826e-18	1.0273e-23	1.5851e-27	3.2664e-31

5.2 Application to an Elliptic Interface Problem

In this section, we apply our algorithm to the numerical solution of an elliptic interface problem. Let $\Omega_1 := \Omega^-$ and $\Omega_2 := \Omega^+$. We consider the elliptic interface problem:

$$\begin{cases} -\nabla \cdot \left(a \nabla u\right) = f, & \text{in } \Omega_1 \cup \Omega_2, \\ u = g, & \text{on } \partial\Omega \setminus \Gamma, \\ [u] = g_{\mathrm{D}}, & \text{on } \Gamma, \\ [a \nabla u \cdot \mathbf{n}] = g_{\mathrm{N}}, & \text{on } \Gamma, \end{cases} \tag{8}$$

where $[\,\cdot\,]$ denotes the jump of a function across the interface Γ, i.e., $[v] := v|_{\Omega_1} - v|_{\Omega_2}$, $\mathbf{n}$ is the unit outward normal of $\partial\Omega_1$, and the coefficient $a := a(\mathbf{x})$ is bounded from below and above by some positive constants and can be discontinuous across the interface Γ.

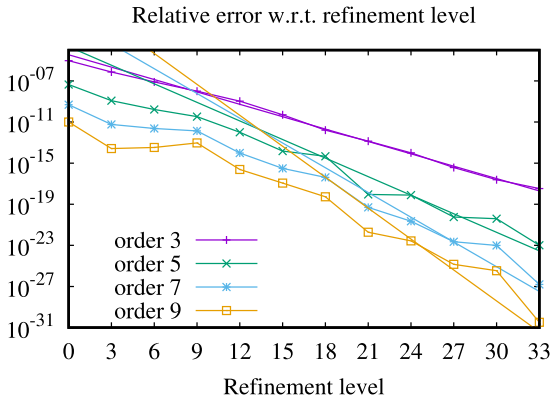


Fig. 10. Relative quadrature errors with respect to mesh refinement levels (h-convergence). The straight lines denote the slopes for the optimal theoretical convergence rate h^{p+1} expected when using the order-p quadrature rules for $p = 3, 5, 7$, and 9.

For convenience of descriptions, each tetrahedron in Ω_h is considered to be closed throughout this subsection. Define the set of patches of Γ induced by Ω_h as

$$\mathcal{I}_h := \left\{ e \;\middle|\; e = T \cap \Gamma \text{ and } \operatorname{area}(e) \neq 0,\ \forall\, T \in \Omega_h \right\}.$$

Each element $e \in \mathcal{I}_h$ either is entirely contained in an element $T \in \Omega_h$ and has non-empty intersection with the interior of T, or is the common face of two neighbouring elements of Ω_h. For both cases $T^e \in \Omega_h$ denotes one of the elements containing e.

We shall adopt the unfitted hp-interface penalty finite element method to solve Equation (8) (see Reference [13]). Define the space of piecewise regular functions by

$$V := \left\{ v \;\middle|\; v \in L^2(\Omega) \text{ and } v|_{\Omega_i} \in H^1(\Omega_i),\ i = 1, 2 \right\}.$$

For $p \in \mathbb{N}$, define the extended finite element space by

$$V(p, \Omega_h) := \left\{ v_h \;\middle|\; v_h \in V \text{ and } v_h|_{T \cap \Omega_i} \in \mathbb{P}_p(T \cap \Omega_i),\quad i = 1, 2,\quad \forall\, T \in \Omega_h \right\},$$

where $\mathbb{P}_p(D)$ denotes the space of polynomials of degree p on domain D. Moreover, we also need the function spaces with homogeneous boundary conditions

$$V_0 := \left\{ v \;\middle|\; v \in V \text{ and } v = 0 \text{ on } \partial\Omega \setminus \Gamma \right\}, \qquad V_0(p, \Omega_h) := V(p, \Omega_h) \cap V_0.$$

Let β be a real parameter and let $\gamma_0 > 0$, $\gamma_1 > 0$ be the parameters for interior penalties. Define the bilinear form $a_h \colon V(p, \Omega_h) \times V(p, \Omega_h) \to \mathbb{R}$ as follows:

$$a_h(u, v) := \sum_{i=1}^{2} \int_{\Omega_i} a \nabla u \cdot \nabla v \, \mathrm{d}\mathbf{x} - \sum_{e \in \mathcal{I}_h} \int_e \left(\{ a \nabla u \cdot \mathbf{n} \} [v] + \beta [u] \{ a \nabla v \cdot \mathbf{n} \} \right) \mathrm{d}\Gamma + J_0(u, v) + J_1(u, v),$$

$$J_0(u, v) := \sum_{e \in \mathcal{I}_h} \frac{\gamma_0 p^2}{h_e} \int_e [u][v] \, \mathrm{d}\Gamma,$$

$$J_1(u, v) := \sum_{e \in \mathcal{I}_h} \frac{\gamma_1 h_e}{p^2} \int_e [a \nabla u \cdot \mathbf{n}][a \nabla v \cdot \mathbf{n}] \, \mathrm{d}\Gamma,$$

where h_e denotes the diameter of e and $\{\cdot\}$ denotes the average of a function across the interface Γ, that is, $\{v\} := (v|_{\Omega_1} + v|_{\Omega_2})/2$. Define the linear form $F_h\colon V(p, \Omega_h) \to \mathbb{R}$ as follows:

$$F_h(v) := \int_\Omega fv \, \mathrm{d}\mathbf{x} + \int_\Gamma g_N\{v\} \, \mathrm{d}\Gamma - \beta \int_\Gamma g_D\{a\nabla v \cdot \mathbf{n}\} \, \mathrm{d}\Gamma + J_D(v) + J_N(v),$$

$$J_D(v) := \sum_{e \in \mathcal{I}_h} \frac{\gamma_0 p^2}{h_e} \int_e g_D[v] \, \mathrm{d}\Gamma,$$

$$J_N(v) := \sum_{e \in \mathcal{I}_h} \frac{\gamma_1 h_e}{p^2} \int_e g_N[a\nabla v \cdot \mathbf{n}] \, \mathrm{d}\Gamma.$$

Let g_h be finite element interpolation of g based on the mesh. An extended finite element approximation to Equation (8) reads: Find $u_h \in V(p, \Omega_h)$ such that $u_h = g_h$ on $\partial\Omega \setminus \Gamma$ and

$$a_h(u_h, v_h) = F_h(v_h), \quad \forall v_h \in V_0(p, \Omega_h). \tag{9}$$

The actual settings of our numerical experiments are as follows. The domain $\Omega = (0, 1)^3$. The interface Γ is the sphere of radius 0.25 centered at $(0.5, 0.5, 0.5)$. The exact solution is given by

$$u(x, y, z) = \begin{cases} \exp(xyz), & (x, y, z) \in \Omega_1, \\ \sin(x + y + z), & (x, y, z) \in \Omega_2. \end{cases}$$

The discontinuous coefficient function is defined such that $a(\mathbf{x}) \equiv 1$ for $\mathbf{x} \in \Omega_1$ and $a(\mathbf{x}) \equiv 100$ for $\mathbf{x} \in \Omega_2$. The weighted $L^2(\Omega)$ and (semi) $H^1(\Omega)$ errors of the discrete solution u_h are defined, respectively, as

$$|u - u_h|_0^2 = \sum_{i=1}^{2} \int_{\Omega_i} a|u - u_h|^2 \, \mathrm{d}\mathbf{x}, \quad |u - u_h|_1^2 = \sum_{i=1}^{2} \int_{\Omega_i} a|\nabla u - \nabla u_h|^2 \, \mathrm{d}\mathbf{x}.$$

The computations of the integrals in the above linear and bilinear forms (with u or v replaced by finite element basis functions), as well as those in the estimation of the L^2 and H^1 errors, are done either using the algorithm proposed in this article if the element intersects with the interface, or using a quadrature rule for tetrahedra as presented in Reference [17] otherwise. The parameters used are $\beta = -1$ and $\gamma_0 = \gamma_1 = 1$. The discrete linear system in Equation (9) is solved using the parallel sparse direct linear solver MUMPS [20], either directly or as the preconditioner of the GMRES method [21].

Relative errors and convergence rates of numerical solutions for P_p elements for $p = 1, 2, 3$, and 4 are listed in Table 2. They were obtained on meshes generated by uniform refinements using the newest vertex bisection algorithm of an initial mesh consisting of 6 congruent tetrahedra. The quadrature order q is set to $q = 2p + 3$. For P_1, P_2, and P_3 elements the computations were done using the 64-bit double precision and the linear systems were solved using MUMPS, but for P_4 element, to eliminate influences of roundoff errors, the computations were done using the x86's 80-bit extended double precision (the `long double` type) and the linear systems were solved using the GMRES method with MUMPS in double precision as its preconditioner, which can be regarded as an iterative refinement method for MUMPS. The convergence rates are optimal for both $H^1(\Omega)$ errors (order p) and $L^2(\Omega)$ errors (order $p + 1$). The numbers of degrees of freedom listed in the table do not include the overlapped ones in the elements intersecting with the interface introduced by the XFEM algorithm, so they are slightly smaller than the actual numbers.

Also listed in Table 2 are the elapsed (wall clock) times for building (the "Build" column) and solving (the "Solve" column) the linear systems using 256 MPI processes on 16 nodes. They should only be regarded as a rough indication of the relative costs of the proposed numerical quadrature algorithm in finite element computations, since the linear solvers used are certainly not optimal.

Table 2. Errors and Convergence Orders of the Numerical Solutions

P_1 element ($p = 1$, $q = 2p + 3 = 5$)							
Number of elements	Degrees of freedom	Relative H^1 error		Relative L^2 error		Elapsed time (s)	
		Error	Order	Error	Order	Build	Solve
768	189	2.144e-01	--	1.351e-02	--	0.03	0.05
6,144	1,241	1.059e-01	1.02	5.183e-03	1.38	0.03	0.05
49,152	9,009	4.153e-02	1.35	9.055e-04	2.52	0.14	0.29
393,216	68,705	1.800e-02	1.21	1.861e-04	2.28	0.60	2.29
3,145,728	536,769	8.606e-03	1.06	3.188e-05	2.55	2.69	19.5
P_2 element ($p = 2$, $q = 2p + 3 = 7$)							
Number of elements	Degrees of freedom	Relative H^1 error		Relative L^2 error		Elapsed time (s)	
		Error	Order	Error	Order	Build	Solve
768	1,241	3.570e-03	--	9.430e-05	--	0.01	0.05
6,144	9,009	8.189e-04	2.12	1.102e-05	3.01	0.05	0.31
49,152	68,705	1.959e-04	2.06	1.327e-06	3.05	0.26	2.57
393,216	536,769	4.830e-05	2.02	1.641e-07	3.02	1.18	22.3
3,145,728	4,243,841	1.202e-05	2.01	2.050e-08	3.00	5.49	225
P_3 element ($p = 3$, $q = 2p + 3 = 9$)							
Number of elements	Degrees of freedom	Relative H^1 error		Relative L^2 error		Elapsed time (s)	
		Error	Order	Error	Order	Build	Solve
768	9,009	2.697e-04	--	4.450e-06	--	0.03	0.13
6,144	29,449	3.434e-05	2.97	2.714e-07	4.04	0.14	1.16
49,152	228,241	4.328e-06	2.99	1.702e-08	4.00	0.72	9.76
393,216	1,797,409	5.427e-07	3.00	1.065e-09	4.00	3.25	86.1
3,145,728	14,266,945	6.793e-08	3.00	6.655e-11	4.00	16.0	1257
P_4 element ($p = 4$, $q = 2p + 3 = 11$)							
Number of elements	Degrees of freedom	Relative H^1 error		Relative L^2 error		Elapsed time (s)	
		Error	Order	Error	Order	Build	Solve
768	9,009	4.806e-06	--	6.929e-08	--	0.21	0.45
6,144	68,705	2.768e-07	4.12	2.028e-09	5.09	1.43	3.05
49,152	536,769	1.725e-08	4.00	6.414e-11	4.98	7.37	29.4
393,216	4,243,841	1.081e-09	4.00	2.022e-12	4.99	32.6	305

For now, we do not have an efficient and scalable iterative solver for this problem yet, which has prevented us from running larger or higher order numerical experiments.

Finally, to illustrate the effect of the quadrature order on the precision of the numerical solution, $H^1(\Omega)$ errors obtained with different quadrature orders on the mesh containing 393,216 elements are shown in Table 3. For this case it seems $q = 2p + 1$ is adequate for P_1, P_2, and P_3 elements, but $q = 2p + 3$ is required for P_4 element. For P_1 element $q = 2p - 1 = 1$ is recommended, which uses a planar approximation of the interface and so is much faster.

6 CONCLUSIONS

A general-purpose, high-order algorithm for computing volume integrals in a part of a tetrahedron bounded by an implicitly defined curved interface, or surface integrals on the intersection of the interface with the tetrahedron, is presented. The robustness and high accuracy of the algorithm are demonstrated with a set of numerical experiments. The code for the algorithm is freely available in

Table 3. $H^1(\Omega)$ Errors of the Numerical Solutions Obtained using Different Quadrature Orders on the Mesh Containing 393,216 Elements

Quadrature order	Relative $H^1(\Omega)$ errors			
	P_1	P_2	P_3	P_4
$q = 2p - 1$	1.806e-02	5.968e-05	9.548e-07	1.441e-07
$q = 2p + 1$	1.800e-02	4.830e-05	5.428e-07	3.995e-09
$q = 2p + 3$	1.800e-02	4.830e-05	5.427e-07	1.081e-09
$q = 2p + 5$	1.800e-02	4.830e-05	5.427e-07	1.096e-09

p: finite element order, q: quadrature order.

the open-source parallel adaptive finite element toolbox PHG and can be used in implementations of high-order numerical algorithms for solving three-dimensional problems involving an implicit interface.

We will continue to work on various aspects of both the algorithm and its implementation to improve the robustness and performance of the code, including factors affecting the accuracy of results, more efficient and reliable univariate root-finding algorithms for polynomial and nonpolynomial functions, adaptive selection of Gaussian quadrature order according to the length of the integration interval, and applications to numerical solutions of real world problems.

The algorithm can be adopted to other types of 3D polyhedral elements. It can also be readily applied to the much simpler case of 2D elements in which only the essential singularities of the first kind need to be addressed. Extensions to higher dimensions are possible but become more complicated and involved than the 3D case, since, for example, on a d-simplex one has to deal with essential singularities caused by the traces of the interface on all k-subsimplices for $1 < k < d$.

REFERENCES

[1] Y. Gong, B. Li, and Z. Li. 2008. Immersed-interface finite-element methods for elliptic interfaces problems with non-homogeneous jump conditions. *SIAM J. Numer. Anal.* 46 (2008), 472–495.

[2] Z. Chen, Z. Wu, and Y. Xiao. 2015. An adaptive immersed finite element method with arbitrary Lagrangian-Eulerian scheme for parabolic equations in time variable domains. *Int. J. Numer. Anal. Model.* 12 (2015), 567–591.

[3] A. Hansbo and P. Hansbo. 2002. An unfitted finite element method, based on Nitsche's method, for elliptic interface problems. *Comput. Methods Appl. Mech. Engrg.* 191 (2002), 5537–5552.

[4] R. Massjung. 2012. An unfitted discontinuous Galerkin method applied to elliptic interface problems. *SIAM J. Numer. Anal.* 50, 6 (2012), 3134–3162.

[5] Zhiming Chen, Yuanming Xiao, and Linbo Zhang. 2009. The adaptive immersed interface finite element method for elliptic and Maxwell interface problems. *J. Comput. Phys.* 228 (2009), 5000–5019.

[6] Yan Xie, Tiantian Liu, Bin Tu, Benzhuo Lu, and Linbo Zhang. 2016. Automated parallel and body-fitted mesh generation in finite element simulation of macromolecular systems. *Commun. Comput. Phys.* 19, 3 (2016), 582-602.

[7] R. I. Saye. 2015. High-order quadrature methods for implicitly defined surfaces and volumes in hyperrectangles. *SIAM J. Sci. Comput.* 37, 2 (2015), A993–A1019.

[8] Christoph Lehrenfeld. 2016. High-order unfitted finite element methods on level set domains using isoparametric mappings. *Comput. Methods Appl. Mech. Engrg.* 300 (2016), 716–733.

[9] B. Müller, F. Kummer, and M. Oberlack. 2013. Highly accurate surface and volume integration on implicit domains by means of moment-fitting. *Int. J. Numer. Meth. Engng* 96 (2013), 512–528.

[10] Catherine Kublik and Richard Tsai. 2018. An extrapolative approach to integration over hypersurfaces in the level set framework, *Mathematics of Computation*, Vol. 87, No. 313, pp. 2365–2392.

[11] Lukas Drescher, Holger Heumann, and Kersten Schmidt. 2017. A high order method for the approximation of integrals over implicitly defined hypersurfaces. *SIAM J. Numer. Anal.* 55, 6 (2017), 2592–2615.

[12] Thomas-Peter Fries and Samir Omerović. 2016. Higher-order accurate integration of implicit geometries. *Int. J. Numer. Meth. Engng* 106 (2016), 323–371.

[13] Haijun Wu and Yuanming Xiao. 2010. An unfitted hp-interface penalty finite element method for elliptic interface problems, arXiv preprint arXiv:1007.2893.

[14] Peiqi Huang, Haijun Wu, and Yuanming Xiao. 2017. An unfitted interface penalty finite element method for elliptic interface problems. *Comput. Methods Appl. Mech. Engrg.* 323 (2017), 439–460.
[15] M. C. Rivara. 1984. Mesh refinement processes based on the generalized bisection of simplices. *SIAM J. Numer. Anal.* 21 (1984), 604–613.
[16] J. Bey. 1995. Tetrahedral grid refinement. *Computing* 55 (1995), 355–378.
[17] L. B. Zhang, T. Cui, and H. Liu. 2009. A set of symmetric quadrature rules on triangles and tetrahedra. *J. Comput. Math.* 27, 1 (2009), 89–96.
[18] L. B. Zhang. 2009. A parallel algorithm for adaptive local refinement of tetrahedral meshes using bisection. *Numer. Math. Theor. Meth. Appl.* 2, 1 (2009), 65–89.
[19] The toolbox Parallel Hierarchical Grid (PHG). Retrieved from http://lsec.cc.ac.cn/phg.
[20] MUMPS: A parallel sparse direct solver. Retrieved from http://mumps.enseeiht.fr/.
[21] Y. Saad. 2003. *Iterative Methods for Sparse Linear Systems*, 2nd ed., Society for Industrial and Applied Mathematics.