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

Tao Cui[1,2], Wei Leng[1,2], Huaqing Liu[1,2], Linbo Zhang[1,2,*] and Weiying Zheng[1,2]

1. State Key Laboratory of Scientific and Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China.
2. School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing 100049, China.

**Abstract**

Given a shape regular tetrahedron and a curved surface which is defined implicitly by a nonlinear level set function and divides the tetrahedron into two sub-domains, a general-purpose, robust, and arbitrarily high order numerical algorithm is proposed in this paper for computing both volume integrals in the sub-domains and surface integrals on their common boundary. The algorithm uses a direct approach which 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 order by increasing the orders of Gaussian quadratures, and 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.

**Keywords:** quadrature, tetrahedral mesh, curved surface, extended finite element, high order

## 1 Introduction

Let $\Omega_h$ be a shape-regular tetrahedral mesh for a domain $\Omega \subset \mathbb{R}^3$ and $\Gamma$ a piecewisely smooth interface in $\Omega$ 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 $\Omega_h$. We make the following assumptions on $T$ and $\Gamma$:

**(A1)** $\Gamma$ is smooth in $T$.

**(A2)** $\Gamma$ is locally flat with respect to $T$, i.e., $\kappa h$ is small where $\kappa$ denotes the mean curvature of $\Gamma$ 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:

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

where $\Omega^- := \{\mathbf{x} \mid L(\mathbf{x}) < 0\}$ and $u(\mathbf{x})$ is an user function which is smooth in $T$.

Computations of the above types of integrals are often required in the implementations of many numerical algorithms involving an interface which 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 paper

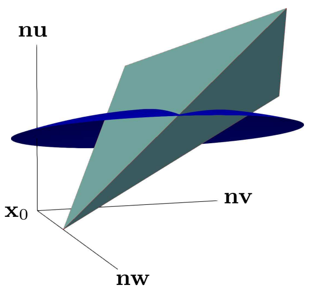


Figure 1: The integration directions.

because it turns out to be the most difficult one in three dimensions compared to other cases and is in fact also the most basic one since any polyhedral element can be easily subdivided into tetrahedra, so once a reliable algorithm for computing the integrals in (1) is available for tetrahedra, it can be used to compute the integrals for any type of elements 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), 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, the problem becomes very hard. Although many algorithms based on various techniques have been proposed for various types of elements, see, for examples, [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 arbitrarily high order code for computing the integrals in (1), especially when $T$ is a tetrahedron.

The algorithm proposed in this paper is based on a simple and direct approach which chooses an origin $\mathbf{x}_0$ and three orthogonal directions represented by three orthonormal vectors $\mathbf{nu}$, $\mathbf{nv}$ and $\mathbf{nw}$ such that

$$T \subset \left\{\mathbf{x}_0 + r\mathbf{nu} + s\mathbf{nv} + t\mathbf{nw} \;\middle|\; r \in (0,a), s \in (0,b), t \in (0,c)\right\},$$

decomposes integrals in (1) into multiple 1D integrals along these directions, and uses 1D Gaussian quadratures to compute the 1D integrals, see Figure 1 for an illustration. The key point for the algorithm to work is how to deal with discontinuities 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 paper 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 $\mathbf{nu}$, $\mathbf{nv}$ and $\mathbf{nw}$ are described. In Section 4 details on identifying smooth subintervals in the integrands of the 1D integrals are provided. In Section 5 the computation of the surface integral $I^0$ is briefly discussed. In Section 6 the user interface of our code for the algorithm is summarized. In Section 7 numerical results are presented. Finally in Section 8 concluding remarks are given.

## 2 The algorithm

We shall use the computation of $I^-$ to describe our algorithm. The procedure for computing $I^0$ will be briefly discussed in Section 5. First, we give two definitions.

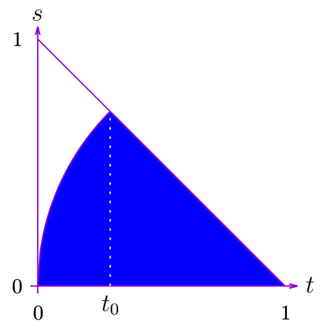


Figure 2: Illustration of a non-essential discontinuity at $t = t_0$ and an essential discontinuity at $t = 0$ in the function $h(t) = \int_{D(t)} g(s,t)\,\mathrm{d}s$, with $g(s,t) \equiv 1$ and $D(t) = \{s \mid s \in (0, 1-t),\ s^2 + (t-1)^2 - 1 < 0\}$.

**Definition 1.** *A point $x_0 \in \mathbb{R}$ is said to be a* discontinuity *of an univariate function $v(x)$ if the function is unsmooth at $x_0$. It's called a* non-essential discontinuity *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's called an* essential discontinuity*.*

**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 $F$.*

In the proposed algorithm, the integral $I^-$ is simply 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 $T \cap \Omega^-$, $u(r,s,t) := u(\mathbf{x}_0 + r\mathbf{nu} + s\mathbf{nv} + t\mathbf{nw})$, and $\chi^-(r,s,t) := \chi^-(\mathbf{x}_0 + r\mathbf{nu} + s\mathbf{nv} + t\mathbf{nw})$. The integrands in the above 1D integrals are defined as follows:

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

Note that the function $g(s,t)$ is an 1D integral in $r$ on a line segment along the direction **nu** for given $s$ and $t$, while the function $h(t)$ is a double integral in $r$ and $s$ on a plane parallel to **nu** and **nv** for given $t$.

The 1D integrals in (2) are computed by splitting the integration intervals into subintervals in which the corresponding integrands are smooth, and applying a Gaussian quadrature rule to each subinterval. The key steps of the algorithm are:

(1) choosing the directions **nu**, **nv** and **nw** to avoid essential discontinuities in the integrands,

(2) subdividing the tetrahedron if essential discontinuities are unavoidable, and

(3) identifying non-essential discontinuities in the integrands.

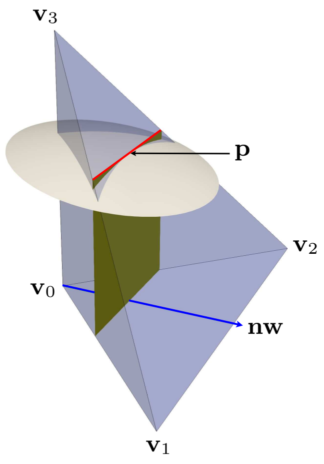


Figure 3: An illustration of second kind essential discontinuities. The four vertices of the tetrahedron are $\mathbf{v}_0 = (\frac{1}{2}, \frac{1}{2}, \frac{5}{8})$, $\mathbf{v}_1 = \mathbf{v}_0 + (\frac{1}{5}, 0, 0)$, $\mathbf{v}_2 = \mathbf{v}_0 + (0, \frac{1}{5}, 0)$ and $\mathbf{v}_3 = \mathbf{v}_0 + (0, 0, \frac{1}{5})$, the interface 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 $\mathbf{nu} = (0, 0, 1)$, $\mathbf{nv} = (\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}, 0)$ and $\mathbf{nw} = (\frac{1}{10}, \frac{1}{10}, 0)$.

A non-essential discontinuity in $f(r, s, t)$, $g(s, t)$ or $h(t)$ is created by a change in the geometric configuration of the integration domain of the inner integral. It typically happens when the integration line segment (parallel to $\mathbf{nu}$) or plane (parallel to $\mathbf{nu}$ and $\mathbf{nv}$) crosses the intersections between two faces of $T$ or the interface $\Gamma$ and a face of $T$. A 2D illustration of non-essential discontinuities is given in Figure 2 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.

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

Essential discontinuities of the first kind occur at intersection points of the interface with the integration line segment for computing $g(s, t)$ (parallel to $\mathbf{nu}$), at which the integration line segment is tangential to the interface. Discontinuities of this kind are illustrated by the 2D example in Figure 2 at $t = 0$. They can be easily avoided if the interface is relatively flat with respect to the element by choosing $\mathbf{nu}$ close to the normal direction of the interface.

Essential discontinuities of the second kind only exist in $h(t)$, which occur on a face of $T$ at an intersection point of the trace of the integration plane for computing $h(t)$ (parallel to $\mathbf{nu}$ and $\mathbf{nv}$) with the trace of $\Gamma$, at which the two traces are tangential. An example of this kind of discontinuities is shown in Figures 3 and 4. In the example, the function $h(t)$ as shown in Figure 4, with the range of $t$ scaled to $[0, 1]$ for convenience of discussion, 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 position where the trace (the top side of the rectangle in Figure 3) of the plane

$$\{\mathbf{v}_0 + t_1\mathbf{nw} + s\mathbf{nv} + r\mathbf{nu} \mid r,\, s \in (-\infty, \infty)\}$$

(the rectangle in Figure 3) on the face opposite to $\mathbf{v}_0$ is tangential to the trace of the interface at the intersection point $\mathbf{p}$, and $t_2 = (13 - \sqrt{31})/16 \approx 0.4645$ which corresponds to the position where the interface intersects with the edges $\mathbf{v}_1$-$\mathbf{v}_3$ and $\mathbf{v}_2$-$\mathbf{v}_3$. An essential discontinuity in $h(t)$ at $t = t_1$ is clearly shown in Figure 4: $\lim_{t \to t_1^+} h''(t) \to -\infty$.

In general we conjecture that on a $d$-simplex ($d \geq 2$) the essential discontinuities may only exist on

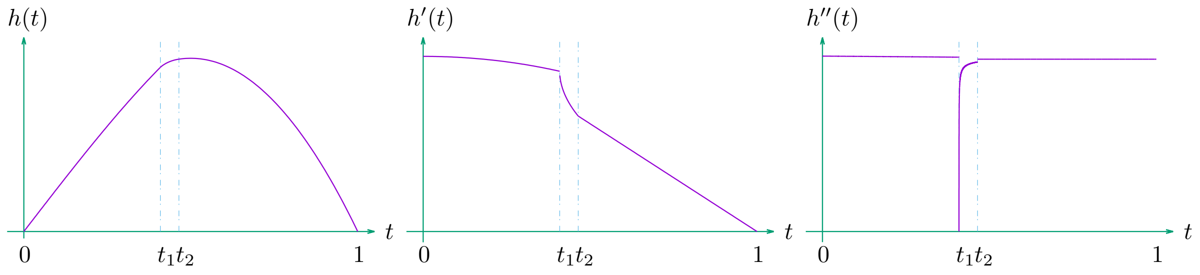


Figure 4: Plot of $h(t)$ (left), $h'(t)$ (middle) and $h''(t)$ (right) for $h(t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \chi^-(\mathbf{v}_0 + r\mathbf{nu} + s\mathbf{nv} + t\mathbf{nw})\,\mathrm{d}r\,\mathrm{d}s$. Here $\chi^-$, $\mathbf{v}_0$, $\mathbf{nw}$, $\mathbf{nv}$, $\mathbf{nu}$, etc., are the same as in Figure 3.

a $k$-subsimplex, at a point where the trace of an integration hyperplane intersects with and is tangential to the trace of the interface, for $1 < k \le d$.

## 3 The integration directions

### 3.1 The direction nu

To avoid essential discontinuities in $g(s,t)$, the direction $\mathbf{nu}$ should be close to the normal direction of the interface. This is similar to the strategy used in [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 $\mathbf{nu}$ in the algorithm:

(1) $\mathbf{nu}$ 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 $\Gamma$.

(2) $\mathbf{nu}$ is the best approximation to $\nabla L(\mathbf{c})$ among the four normal directions of element faces. This choice slightly simplifies the implementation since $\mathbf{nv}$ and $\mathbf{nw}$ are aligned with one face. But numerical experiments show that this choice generally leads to poor accuracy because it happens that any of the four normal directions of a tetrahedron is nearly tangential to $\Gamma$ at some point of $\Gamma$, especially when the interface is not sufficiently resolved by the element. 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 7 were obtained with it. The second choice is kept in the code only for future investigations.

### 3.2 The directions nv and nw

Given $\mathbf{nu}$, the directions $\mathbf{nw}$ and $\mathbf{nv}$ can be parameterized with a scalar parameter $w \in [-1,1]$ as:

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

where $\mathbf{e}_1$ and $\mathbf{e}_2$ are two arbitrarily chosen and linearly independent vectors orthogonal to $\mathbf{nu}$.

Denote the integration plane for $h(t)$ by $\mathcal{P} = (\mathbf{nu}, \mathbf{nv})$, which is orthogonal to $\mathbf{w}_w$. Given a face $F$ of $T$, denote its normal vector by $\mathbf{n}_F$. It is clear that the trace of $\mathcal{P}$ on $F$ is parallel to $\mathbf{w}_w \times \mathbf{n}_F$. Denote the trace of $\Gamma$ on $F$ by $\gamma_F$, which is empty if $\Gamma$ does not intersect with $F$. Given a point $\mathbf{p}$ on $\gamma_F$, denote the unit tangent vector of $\gamma_F$ at $\mathbf{p}$ by $\mathbf{t}_F(\mathbf{p})$.

To avoid second kind essential discontinuities, $w$ should be chosen such that $\mathbf{w}_w \times \mathbf{n}_F$ is not tangential to $\gamma_F$ at any point of $\gamma_F$. So we introduce the following definition.

**Definition 3.** *A value of $w$ is called* permissible *if for any face $F$ of $T$ and any point $\mathbf{p}$ of $\gamma_F$, $\mathbf{w}_w \times \mathbf{n}_F$ is not parallel to $\mathbf{t}_F(\mathbf{p})$. Otherwise it's called* impermissible. *An interval of impermissible values is called an* impermissible interval.

Let $F$ be a face of $T$. Let's suppose that $\gamma_F$ has two intersection points with $\partial F$ and denote them by $\mathbf{p}_0$ and $\mathbf{p}_1$ respectively. Let $\mathbf{t}_0 := \mathbf{t}_F(\mathbf{p}_0)$ and $\mathbf{t}_1 := \mathbf{t}_F(\mathbf{p}_1)$. Suppose $\gamma_F$ is parameterized as $\mathbf{g}(t)$ for $t \in [0,1]$ with $\mathbf{g}(0) = \mathbf{p}_0$ and $\mathbf{g}(1) = \mathbf{p}_1$, then $\mathbf{t}_0 = \mathbf{g}'(0)/|\mathbf{g}'(0)|$ and $\mathbf{t}_1 = \mathbf{g}'(1)/|\mathbf{g}'(1)|$. Define $\phi(t) := \big(\mathbf{g}'(t) \times (\mathbf{w}_w \times \mathbf{n}_F)\big) \cdot \mathbf{n}_F$. Since both $\mathbf{g}'(t)$ and $\mathbf{w}_w \times \mathbf{n}_F$ are parallel to $F$, so $\mathbf{g}'(t) \times (\mathbf{w}_w \times \mathbf{n}_F)$ is parallel to $\mathbf{n}_F$, and $\mathbf{g}'(t)$ is parallel to $\mathbf{w}_w \times \mathbf{n}_F$ if and only if $\phi(t) = 0$. Thus if $w$ is permissible then the function $\phi(t)$ should have no root in $[0,1]$. By Intermediate Value Theorem, a sufficient condition for $w$ being impermissible is $\phi(0)\phi(1) \le 0$, which is equivalent to

$$\Big(\big(\mathbf{t}_0 \times (\mathbf{w}_w \times \mathbf{n}_F)\big) \cdot \mathbf{n}_F\Big)\Big(\big(\mathbf{t}_1 \times (\mathbf{w}_w \times \mathbf{n}_F)\big) \cdot \mathbf{n}_F\Big) \le 0. \tag{4}$$

Substituting the expression for $\mathbf{w}_w$ in (3) into the above inequality it becomes

$$\Big(a_0 w + b_0\sqrt{1-w^2}\Big)\Big(a_1 w + b_1\sqrt{1-w^2}\Big) \le 0, \tag{5}$$

where $a_i = \big(\mathbf{t}_i \times (\mathbf{e}_1 \times \mathbf{n}_F)\big) \cdot \mathbf{n}_F$ and $b_i = \big(\mathbf{t}_i \times (\mathbf{e}_2 \times \mathbf{n}_F)\big) \cdot \mathbf{n}_F$, $i = 0, 1$. Assume $w \neq 0$. Dividing the inequality in (5) by $w^2$ and letting $\delta := \frac{\sqrt{1-w^2}}{w}$, we get the following quadratic inequality for $\delta$:

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

The above inequality can be easily solved to get impermissible intervals for $\delta$, they are then converted to impermissible intervals for $w$.

The above procedure for finding impermissible intervals is performed on each face of $T$. The complement of the union of all those impermissible intervals, which is also composed of none or several intervals, is called the *candidate intervals*. The value of $w$ should be chosen in the interior of the candidate intervals.

Define

$$\alpha(w) = \max_{F\in\mathcal{F}(T)} \left\{ \max_{\mathbf{p}\in\gamma_F} \Big|\Big(\frac{\mathbf{w}_w}{|\mathbf{w}_w|} \times \mathbf{n}_F\Big) \cdot \mathbf{t}_F(\mathbf{p})\Big| \right\}, \tag{6}$$

where $\mathcal{F}(T)$ denotes the set of faces of $T$. The function $\alpha(w)$ equals to the maximum absolute value of the cosine of the angle between the trace of $\Gamma$ and the trace of $\mathcal{P}$ on all faces of $T$, and has its range included in $[0,1]$. If $\alpha(w) = 1$ then $w$ is impermissible, i.e., the trace of $\Gamma$ is tangential to the trace of $\mathcal{P}$ at some point on some face of $T$. Since $\alpha(-1) = \alpha(1)$, it can be extended to a periodic function in $\mathbb{R}$.

The evaluation of $\alpha(w)$ as defined in (6) requires finding the extrema of univariate functions involving the first derivatives of the level set function $L(\mathbf{x})$, or 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 $L(\mathbf{x})$, a further simplification is made in the algorithm by assuming that $\gamma_F$ is convex on all faces of $T$. Then $\alpha(w)$ can be computed by:

$$\alpha(w) = \max_{F\in\mathcal{F}(T)} \left\{ \max_{p\in\mathcal{I}(F)} \Big|\Big(\frac{\mathbf{w}_w}{|\mathbf{w}_w|} \times \mathbf{n}_F\Big) \cdot \mathbf{t}_F(\mathbf{p})\Big| \right\},$$

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

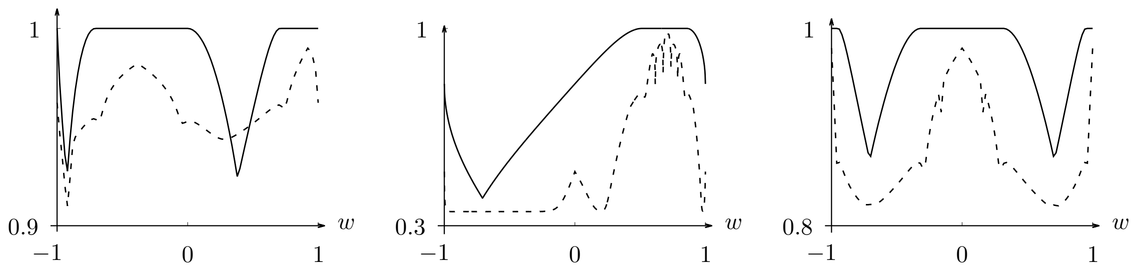


Figure 5: Plots of $\alpha(w)$ and quadrature errors as functions of $w$ obtained with three different tetrahedra. The function $\alpha(w)$ is represented by solid lines, while the logarithm of the absolute values of the quadrature errors, suitably scaled and translated, are represented by dashed lines.

There exist cases in which $\gamma_F$ has more than two intersections with $\partial F$, or is a closed curve and has no intersection with $\partial F$. If such cases are detected then $\alpha(w)$ is set to 1 for all $w$.

Intuitively, we should choose $w$ in such a way that the trace of $\mathcal{P}$ is most orthogonal to the trace of $\Gamma$ on all faces of $T$, thus we set $w = \operatorname{argmin}_w \alpha(w)$ in the algorithm. This choice is validated by Figure 5, which clearly shows a positive correlation between $\alpha(w)$ and the quadrature errors.

For the computation of $w$ it is clear that $\alpha(w) = 1$ if $w$ belongs to one of the impermissible interval, so we only need to search in the candidate intervals. In each candidate interval, $\alpha(w)$ is usually an unimodal function, and we simply perform a few search steps using the 0.618 method.

In the actual algorithm a threshold $A \in (0,1)$ is set for $\alpha(w)$. If the smallest value of $\alpha(w)$ found using the procedure above is greater than $A$, then the tetrahedron is subdivided using either longest edge bisection [15] or regular refinement [16], and the algorithm is recursively applied to the subtetrahedra. The default value for the threshold 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 a volume approaching the machine precision. Numerical experiments show that this kind of recursions are rare if the interface is well resolved by the mesh and thus have no noticeable impact on the overall performance of the algorithm.

There may be other factors which influence the accuracy and the performance of the algorithm. They are left as subjects of future studies.

## 4 Discontinuities in the integrands

In order to compute accurately the 1D integrals in (2) using Gaussian quadrature rules, we need to locate all discontinuities in the integrands, divide the integration intervals into subintervals in which the corresponding integrands are smooth, and use a Gaussian quadrature rule to each subinterval.

### 4.1 Discontinuities in $f(r,s,t)$ for given $s$ and $t$

For given $s$ and $t$, the discontinuities in $f(r,s,t)$ with respect to $r$ only occur at the intersection points of the line segment $\mathcal{L} = \{\mathbf{x}_0 + r\mathbf{nu} + s\mathbf{nv} + t\mathbf{nw} \mid r \in [0,a]\}$ with:

(1) the interface $\Gamma$;

(2) the 4 faces of $T$.

We collect and sort all the above intersection points into an increasing list $\{r_i \in (0,a) \mid 1 \le i \le n\}$ and write the integral as:

$$\int_0^a f(r,s,t)\,\mathrm{d}r = \sum_{i=0}^{n} \int_{r_i}^{r_{i+1}} f(r,s,t)\,\mathrm{d}r$$

with $r_0 = 0$ and $r_{n+1} = a$. Note that only one of the above intervals lies entirely inside $T \cap \Omega^-$, all others lie entirely outside $T \cap \Omega^-$. The integral is computed by applying a Gaussian quadrature rule to the interval inside $T \cap \Omega^-$.

## 4.2 Discontinuities in $g(s,t)$ for given $t$

For given $t$, the integration line segment for $s$ is $\mathcal{L} = \{\mathbf{x}_0 + s\mathbf{nv} + t\mathbf{nw} \mid s \in [0,b]\}$, which lies in the base plane

$$\mathcal{P} = \{\mathbf{x}_0 + s\mathbf{nv} + t\mathbf{nw} \mid s \in (0,b) \text{ and } t \in (0,c)\},$$

and the discontinuities in $g(s,t)$ with respect to $s$ occur at the intersection points of $\mathcal{L}$ with:

(1) the projections to $\mathcal{P}$ of the 6 edges of $T$;

(2) the projections to $\mathcal{P}$ of the traces of $\Gamma$ on the 4 faces of $T$.

We collect and sort the above intersection points into an increasing list $\{s_j \in (0,b) \mid 1 \le j \le m\}$, write the integral as:

$$\int_0^b g(s,t)\,\mathrm{d}s = \sum_{j=0}^{m} \int_{s_j}^{s_{j+1}} g(s,t)\,\mathrm{d}s$$

with $s_0 = 0$ and $s_{m+1} = b$, and apply a Gaussian quadrature rule to each interval in which the integrand is not identically zero.

## 4.3 Discontinuities in $h(t)$

The integration is along the line segment

$$\mathcal{L} = \{\mathbf{x}_0 + t\mathbf{nw} \mid t \in [0,c]\}$$

and the discontinuities in the integrand $h(t)$ may occur at the following places:

(1) the projections to $\mathcal{L}$ of the 4 vertices of $T$;

(2) the projections to $\mathcal{L}$ of the intersection points of the 6 edges of $T$ with $\Gamma$.

Again, we collect and sort the above points into an increasing list $\{t_k \in (0,c) \mid 1 \le k \le l\}$, write the integral as:

$$\int_0^c h(t)\,\mathrm{d}t = \sum_{k=0}^{l} \int_{t_k}^{t_{k+1}} h(t)\,\mathrm{d}t$$

with $t_0 = 0$ and $t_{l+1} = c$, and apply a Gaussian quadrature rule to each interval in which the integrand is not identically zero.

## 5 Surface integral

Computing the surface integral $I^0$ is in fact simpler than computing the volume integral $I^-$. We construct the integration directions $\mathbf{nu}$, $\mathbf{nv}$ and $\mathbf{nw}$ and determine the integration domain $\mathbf{x}_0$, $b$ and $c$, just as for the volume integral, and compute $I^0$ by:

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

where

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

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

Here $\mathbf{x}(r,s,t) := \mathbf{x}_0 + r\mathbf{nu} + s\mathbf{nv} + t\mathbf{nw}$. The above 1D integrals are computed using the same procedure as described in the subsections 4.2 and 4.3.

## 6 Implementation

The algorithm presented in this paper has been implemented in the files `src/quad-interface.c` and `include/phg/quad-interface.h` in the open source parallel adaptive finite element toolbox PHG [19]. It's encapsulated as the following general-purpose function, which can be used in implementations of extended finite element methods (XFEM) or other immersed interface or immersed boundary methods on tetrahedral meshes:

```
int phgQuadInterface(DOF *ls, DOF *ls_grad, ELEMENT *e,
                     DOF_USER_FUNC func, int dim, DOF_PROJ proj,
                     int quad_type, int quad_order,
                     FLOAT *res, FLOAT **prule_data);
```

The types `DOF`, `ELEMENT`, `DOF_USER_FUNC`, `DOF_PROJ` and `FLOAT` are defined in PHG's header files. The return value of the function is the total number of evaluations of the integrand.

Below is a brief description of the arguments of the function.

- The level set function $L(\mathbf{x})$ and its gradient $\nabla L(\mathbf{x})$ are specified by the arguments `ls` and `ls_grad`. `ls` can represent a finite element function (usually a piecewise polynomial) or an analytic function (with the type `DOF_ANALYTIC`). `ls_grad` is optional and can be set to `NULL` if `ls` is an ordinary finite element function, since in this case $\nabla L(\mathbf{x})$ can be computed using `ls`.

  For convenience, and as a special convention, the argument `ls` can be set to `NULL` and then the integral on the whole tetrahedron will be computed if `quad_type` $\neq 0$ (see below). In this case `ls_grad` must not be `NULL`, and can be set to any valid `DOF` attached to the grid such that `ls_grad->g` is valid.

- The tetrahedron $T$ is specified by the argument `e`.

- The integrand $u(\mathbf{x})$ is specified by the arguments `func`, `dim` and `proj`. The type `DOF_USER_FUNC` is defined as:

```
typedef void (*DOF_USER_FUNC)(FLOAT x, FLOAT y, FLOAT z, FLOAT *res);
```

which evaluates the function at the point $(x, y, z)$ and returns the result in `res`, and can encapsulate either a scalar function or a vector function of any dimension. `dim` specifies the dimension of $u(\mathbf{x})$. `proj` is only effective for the surface integral, it specifies the projection operation using the normal direction of the interface $\Gamma$. Let $\mathbf{n}(\mathbf{x})$ be the unit normal vector of $\Gamma$ at $\mathbf{x}$ then:

$$u(\mathbf{x}) = \begin{cases} \texttt{func}(\mathbf{x}), & \text{if } \texttt{proj} = \texttt{DOF\_PROJ\_NONE} \text{ or } \texttt{quad\_type} \neq 0, \\ \texttt{func}(\mathbf{x}) \cdot \mathbf{n}(\mathbf{x}), & \text{if } \texttt{proj} = \texttt{DOF\_PROJ\_DOT} \text{ and } \texttt{quad\_type} = 0, \\ \texttt{func}(\mathbf{x}) \times \mathbf{n}(\mathbf{x}), & \text{if } \texttt{proj} = \texttt{DOF\_PROJ\_CROSS} \text{ and } \texttt{quad\_type} = 0. \end{cases}$$

- The integration type is specified by the argument `quad_type`:
  - volume integral in $T \cap \Omega^-$ if `quad_type` $< 0$;
  - volume integral in $T \cap \Omega^+$ if `quad_type` $> 0$, where $\Omega^+ := \{\mathbf{x} \in \Omega \mid L(\mathbf{x}) > 0\}$;
  - surface integral on $\Gamma \cap T$ if `quad_type` $= 0$.
- The order of the quadrature rules to use is specified by the argument `quad_order`. Since 1D Gaussian rules have odd orders, the rule of order `quad_order` $+ 1$ is used if `quad_order` is even. In the case where the integral is computed by a planar approximation of the interface (for example when the interface is locally planar, or `quad_order` $\leq 1$, or the tetrahedron is very small), quadrature rules for triangles and tetrahedra of order `quad_order`, either those presented in [17] or generated on the fly using tensor products of 1D Gaussian rules, are used.
- The computed integral is returned in the argument `res`, which should be a pointer to an array of `FLOAT`s of size $\geq$ `dim`.
- Finally `prule_data`, if $\neq$ `NULL`, returns a pointer to a dynamically allocated buffer, to be freed by the calling function, containing the data for the computed quadrature rule. If `prule_data` $\neq$ `NULL`, the argument `func` can be `NULL` and the integral is actually computed iff `func` $\neq$ `NULL`.

  The data for the quadrature rule saved in `*prule_data` is an array of `FLOAT`s consisting of an one number header followed by a list of quadrature points and weights. The points are strictly inside the integration domain and the weights are all nonnegative. Please see comments in the source code of the function for detailed format of the data, which can be reused to compute integrals of the same type on the same element more efficiently, either directly or by calling the following function:

  ```
  int phgQuadInterfaceDo(DOF_USER_FUNC func, int dim,
                         const FLOAT *rule_data, FLOAT *res);
  ```

Apart from the arguments, a set of options for setting internal parameters or debugging the algorithm are provided. They have the prefix "`-qi_`" and can be used either as command-line options or at run-time with the `phgOptionsSet*` functions, for example:

```
phgOptionsPush();        /* save current options */
phgOptionsSetOptions("-qi_threshold=0.9 -qi_subdiv_type=regular");
phgQuadInterface(... ...);
phgOptionsPop();         /* restore saved options */
```

The full list of options available for the algorithm can be obtained by running any PHG program with the command-line option "`-help quad_interface`".

An auxiliary function `phgQuadInterfaceMarkElements(DOF *ls)` is provided. It sets the "`mark`" member of all elements in the mesh to 0 if the element might intersect with the interface, $-1$ if the element is entirely contained in $\Omega^-$, and 1 if the element is entirely contained in $\Omega^+$. It helps to determine

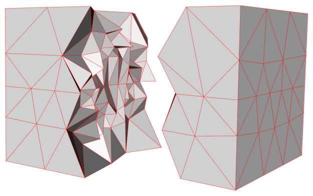


Figure 6: The mesh containing $1,843$ tetrahedra

with which elements this algorithm, which is much more expensive then numerical quadratures without an interface, needs to be used.

A test program, `test/quad_test2.c`, is also included which can serve as a sample code on using the above functions. All the numerical results presented in the subsection 7.1 were obtained with it.

Finally a wrapper function, `phgQuadInterface2`, is provided for using the algorithm in non PHG programs:

```
int phgQuadInterface2(DOF_USER_FUNC ls, int ls_order,
                      DOF_USER_FUNC ls_grad, FLOAT (*tet)[3],
                      DOF_USER_FUNC func, int dim, DOF_PROJ proj,
                      int quad_type, int quad_order,
                      FLOAT *res, FLOAT **prule_data);
```

in which the argument `ls_order` specifies the polynomial order of the level set function (`ls_order` $< 0$ means `ls` is non polynomial) and the array `tet[4][3]` gives the coordinates of the four vertices of the tetrahedron. An example for using this function is provided in the program `test/quad_test3.c`. In this example the tetrahedron gets subdivided because $\alpha(w) \equiv 1$, which can be observed using the command-line option `-qi_show_recursions` when running the program.

## 7 Numerical results

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

### 7.1 Robustness and accuracy tests

In this set of numerical experiments $\Omega$ is the unit cube $(0,1)^3$, the mesh is an unstructured and irregular tetrahedral mesh containing $1,843$ elements as shown in Figure 6, and the integrand $u(\mathbf{x}) \equiv 1$. In this case the sum of $I^-$ over all elements equals to the volume of $\Omega^-$ 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 7:

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

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

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
```

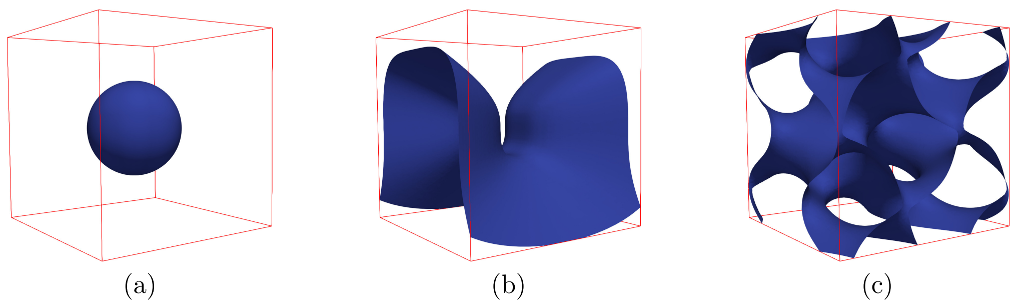


Figure 7: The three test cases.

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

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

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 which is the interpolation in the $P_5$ Lagrangian finite element space of the following non polynomial 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 [7, 8]. In order to better resolve the interface, the elements near the interface, i.e., those marked with 0 by the function `phgQuadInterfaceMarkElements`, 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
```

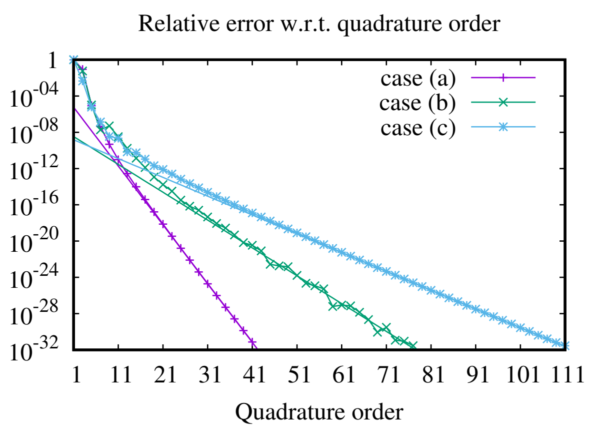
All the results presented in this subsection were obtained using quadruple-precision floating-point operations (GNU C's `__float128` type, with a precision of `1.9259e-34`).

### 7.1.1 $p$-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. Since analytical results are unknown for cases (b) and (c), the errors for order $p$ are computed using the differences between the results of order $p-2$ and order $p$ for all three cases.

The results are shown in Figure 8. Exponential convergence can be observed for all three cases. There are some oscillations for case (b) which we believe are due to the fact that the interface is not well resolved by the mesh.

[1]The current implementation of `phgQuadInterface` supports non polynomial level set functions by using a combined univariate root-finding algorithm based on polynomial approximation, Newton's method and bisections. But for non polynomial 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.



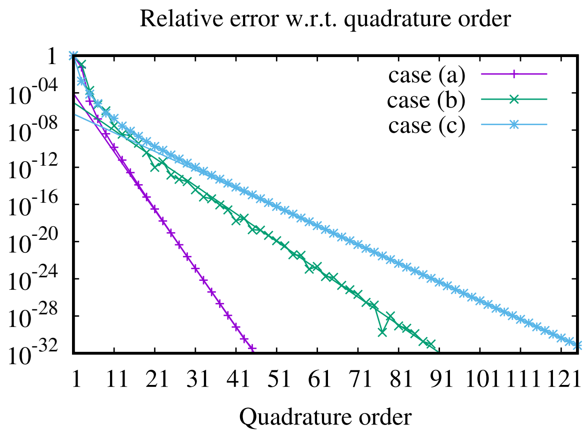


Figure 8: Relative quadrature errors with respect to quadrature orders ($p$-convergence). The straight lines represent fitted functions of the form $\lambda e^{-\theta p}$, where $p$ = order and $\lambda$ and $\theta$ denote positive constants. Left: volume integral. Right: surface integral.

### 7.1.2 $h$-convergence tests

In the $h$-convergence tests, the quadrature order is fixed and relative quadrature errors are computed on meshes which 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 9 and Table 1. The optimal convergence rate $h^{p+1}$ is obtained for $p = 3$ and for $p = 5$ on levels $\geq 18$. For higher orders, the optimal convergence is not convincing at present and will be left for future investigations.

## 7.2 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:

$$\begin{cases} -\nabla \cdot \big(a\nabla u\big) = f, & \text{in } \Omega_1 \cup \Omega_2, \\ u = g, & \text{on } \partial\Omega \setminus \Gamma, \\ [u] = g_{\mathrm{D}}, & \text{on } \Gamma, \\ [a\nabla u \cdot \mathbf{n}] = g_{\mathrm{N}}, & \text{on } \Gamma, \end{cases} \tag{7}$$

where $[\,\cdot\,]$ denotes the jump of a function across the interface $\Gamma$, 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 $\Gamma$.

For convenience of descriptions, each tetrahedron in $\Omega_h$ is considered to be closed throughout this subsection. Define the set of patches of $\Gamma$ induced by $\Omega_h$ as:

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

Each element $e \in \mathcal{I}_h$ either is entirely contained in an element $T \in \Omega_h$ and has non-empty intersection

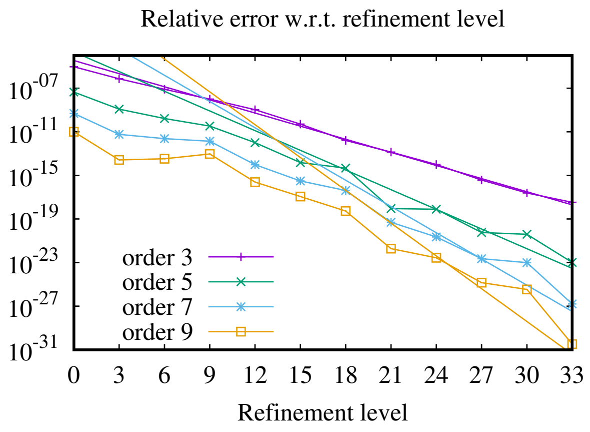


Figure 9: Relative quadrature errors with respect to mesh refinement levels ($h$-convergence). The straight lines denote the slopes for the optimal theoretical convergence rate $h^{p+1}$ expected when using the order-$p$ quadrature rules for $p = 3$, 5, 7 and 9.

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 |

with the interior of $T$, or is the common face of two neighbouring elements of $\Omega_h$. 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 (7) (see [13]). Define the space of piecewise regular functions by

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

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

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

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

$$V_0 := \{v \mid v \in V \text{ and } v = 0 \text{ on } \partial\Omega \setminus \Gamma\}, \quad V_0(p, \Omega_h) := V(p, \Omega_h) \cap V_0.$$

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

$$\begin{aligned}
a_h(u, v) &:= \sum_{i=1}^{2} \int_{\Omega_i} a\nabla u \cdot \nabla v \, \mathrm{d}\mathbf{x} - \sum_{e\in\mathcal{I}_h} \int_e \big(\{a\nabla u \cdot \mathbf{n}\}[v] + \beta[u]\{a\nabla v \cdot \mathbf{n}\}\big) \, \mathrm{d}\Gamma + J_0(u, v) + J_1(u, v),\\
J_0(u, v) &:= \sum_{e\in\mathcal{I}_h} \frac{\gamma_0 p^2}{h_e} \int_e [u][v] \, \mathrm{d}\Gamma,\\
J_1(u, v) &:= \sum_{e\in\mathcal{I}_h} \frac{\gamma_1 h_e}{p^2} \int_e [a\nabla u \cdot \mathbf{n}][a\nabla v \cdot \mathbf{n}] \, \mathrm{d}\Gamma,
\end{aligned}$$

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

$$\begin{aligned}
F_h(v) &:= \int_\Omega fv \, \mathrm{d}\mathbf{x} + \int_\Gamma g_N\{v\} \, \mathrm{d}\Gamma - \beta \int_\Gamma g_D\{a\nabla v \cdot \mathbf{n}\} \, \mathrm{d}\Gamma + J_D(v) + J_N(v),\\
J_D(v) &:= \sum_{e\in\mathcal{I}_h} \frac{\gamma_0 p^2}{h_e} \int_e g_D[v] \, \mathrm{d}\Gamma,\\
J_N(v) &:= \sum_{e\in\mathcal{I}_h} \frac{\gamma_1 h_e}{p^2} \int_e g_N[a\nabla v \cdot \mathbf{n}] \, \mathrm{d}\Gamma.
\end{aligned}$$

Let $g_h$ be finite element interpolation of $g$ based on the mesh. An extended finite element approximation to (7) reads: Find $u_h \in V(p, \Omega_h)$ such that $u_h = g_h$ on $\partial\Omega \setminus \Gamma$ and

$$a_h(u_h, v_h) = F_h(v_h), \quad \forall v_h \in V_0(p, \Omega_h). \tag{8}$$

The actual settings of our numerical experiments are as follows. The domain $\Omega = (0, 1)^3$. The interface $\Gamma$ 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 function `phgQuadInterface` with `quad_order` $= q$ if the element intersects with the interface, or with an order $q$ quadrature rule for tetrahedra as presented in [17] otherwise, where $q$ denotes the numerical quadrature order. The parameters used are $\beta = -1$ and $\gamma_0 = \gamma_1 = 1$. The discrete linear system in (8) 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, with the quadrature order $q = 2p + 3$. Note that 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 x87'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. 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.

## 8 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 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 non polynomial 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 elements. It can also be readily applied to the much simpler case of 2D elements in which only the essential discontinuities of the first kind need to be avoided. 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 avoid essential discontinuities caused by the traces of the interface on all $k$-subsimplices for $1 < k < d$.

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 |

Table 3: $H^1(\Omega)$ errors of the numerical solutions obtained using different quadrature orders on the mesh containing $393,216$ elements. $p$: finite element order, $q$: quadrature order.

| 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 |

## Acknowledgements

This work was supported by the National Key Research and Development Program of China under grant number 2016YFB0201304, National Natural Science Foundation of China under grant numbers 91430215 and 91530323, State Key Laboratory of Scientific and Engineering Computing (LSEC), and National Center for Mathematics and Interdisciplinary Sciences of Chinese Academy of Sciences (NCMIS).

## References

[1] Y. Gong, B. Li and Z. Li, 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, 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, An unfitted finite element method, based on Nitsche's method, for elliptic interface problems, Comput. Methods Appl. Mech. Engrg., 191 (2002), pp. 5537–5552.

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

[5] Zhiming Chen, Yuanming Xiao, Linbo Zhang, The adaptive immersed interface finite element method for elliptic and Maxwell interface problems, Journal of Computational Physics, 228, pp.5000–5019, 2009

[6] Yan Xie, Tiantian Liu, Bin Tu, Benzhuo Lu and Linbo Zhang, Automated parallel and body-fitted mesh generation in finite element simulation of macromolecular systems, Commun. Comput. Phys., Vol. 19, No. 3, pp.582-602, 2016

[7] R. I. Saye, High-order quadrature methods for implicitly defined surfaces and volumes in hyperrectangles, SIAM J. Sci. Comput., Vol. 37, No. 2, pp. A993–A1019, 2015

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

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

[10] Catherine Kublik and Richard Tsai, An extrapolative approach to integration over hypersurfaces in the level set framework, Mathematics of Computation, Volume 87, Number 313, pp.2365–2392, 2018

[11] Lukas Drescher, Holger Heumann, and Kersten Schmidt, A high order method for the approximation of integrals over implicitly defined hypersurfaces, SIAM J. Numer. Anal., Vol. 55, No. 6, pp.2592–2615, 2017

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

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

[14] Peiqi Huang, Haijun Wu and Yuanming Xiao, An unfitted interface penalty finite element method for elliptic interface problems, Comput. Methods Appl. Mech. Engrg. 323, pp.439–460, 2017

[15] M. C. Rivara, Mesh refinement processes based on the generalized bisection of simplices, SIAM J. Numer. Anal., 21, 604–613, 1984

[16] J. Bey, Tetrahedral grid refinement, Computing, 55:355-378, 1995

[17] L. B. Zhang, T. Cui, and H. Liu, A set of symmetric quadrature rules on triangles and tetrahedra, J. Comput. Math., Vol. 27, No. 1, pp. 89–96, 2009

[18] L. B. Zhang, A parallel algorithm for adaptive local refinement of tetrahedral meshes using bisection, Numer. Math. Theor. Meth. Appl., Vol. 2, No. 1, pp. 65–89, 2009

[19] The toolbox Parallel Hierarchical Grid (PHG), `http://lsec.cc.ac.cn/phg`

[20] MUMPS: a parallel sparse direct solver, `http://mumps.enseeiht.fr/`

[21] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd edition, Society for Industrial and Applied Mathematics, 2003