Numerical Quadrature Over Smooth Closed Surfaces

Jonah A. Reeger

Air Force Institute of Technology (AFIT)
Department of Mathematics and Statistics
Wright-Patterson Air Force Base, OH 45433 USA

in collaboration with

Bengt Fornberg

University of Colorado
Department of Applied Mathematics, 526 UCB
Boulder, CO 80309 USA

and

Maloupu L. Watts

The Problem

Evaluate approximately

$$\mathcal{I}_S(f) := \iint\limits_S f(\mathbf{x})dS$$

S is a smooth closed surface (2D) embedded in 3D (i.e. $\mathbf{x} \in \mathbb{R}^3$).

Note:

- Typically only simple integrands $f(\mathbf{x})$ lead to explicit representations of the value of the surface integral.
- In application $f(\mathbf{x})$ may be given only as values sampled at discrete locations.

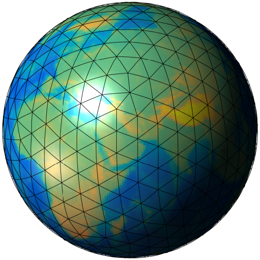
In the case of spherical geometries applications include

- geophysics
- optics
- numerical weather prediction

where integrated quantities such as

- total energy
- total radiance
- average temperature

are required by themselves or to supplement systems of partial differential equations.

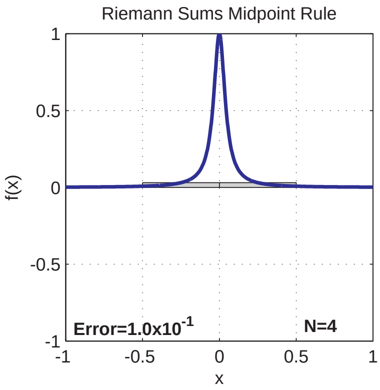


For other geometries we imagine similar uses along with

- Surface areas
- Volumes
- etc.

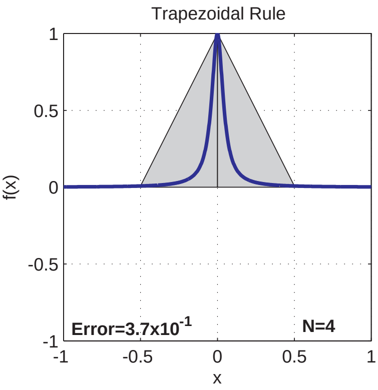
Rectangular Approximations

On each of the N subintervals construct a constant function equal to f at the midpoint (creating many rectangular areas).



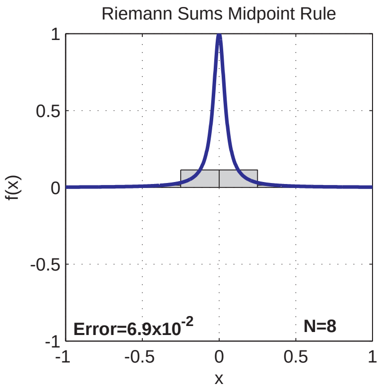
Trapezoidal Approximations

On each of the N subintervals construct a line passing through f at the endpoints (creating many trapezoidal areas).



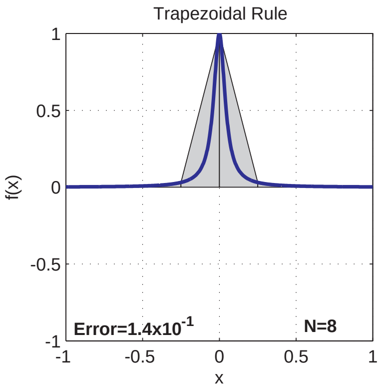
Rectangular Approximations

On each of the N subintervals construct a constant function equal to f at the midpoint (creating many rectangular areas).



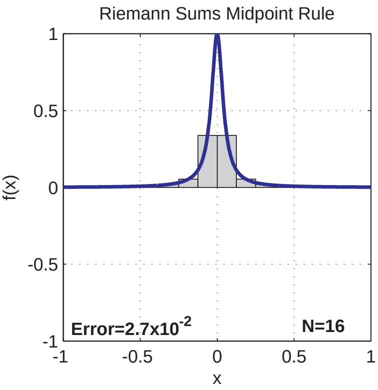
Trapezoidal Approximations

On each of the N subintervals construct a line passing through f at the endpoints (creating many trapezoidal areas).



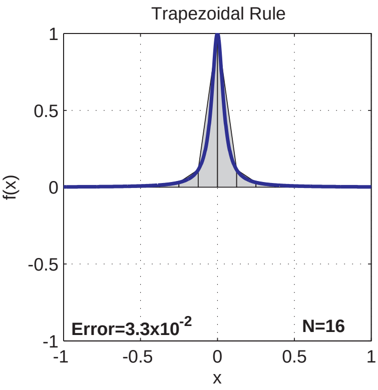
Rectangular Approximations

On each of the N subintervals construct a constant function equal to f at the midpoint (creating many rectangular areas).



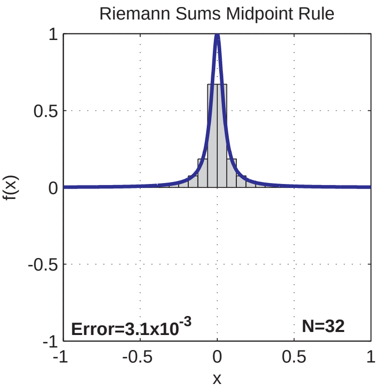
Trapezoidal Approximations

On each of the N subintervals construct a line passing through f at the endpoints (creating many trapezoidal areas).



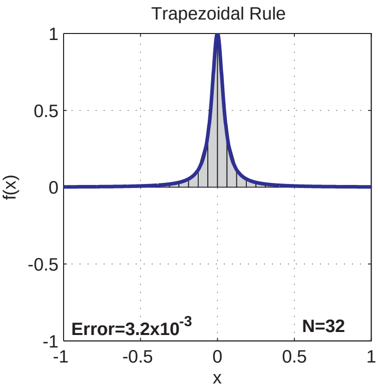
Rectangular Approximations

On each of the N subintervals construct a constant function equal to f at the midpoint (creating many rectangular areas).



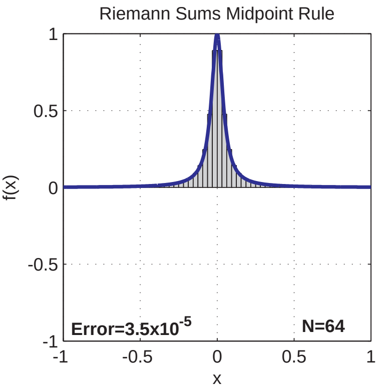
Trapezoidal Approximations

On each of the N subintervals construct a line passing through f at the endpoints (creating many trapezoidal areas).



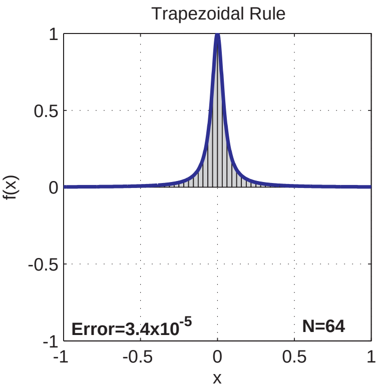
Rectangular Approximations

On each of the N subintervals construct a constant function equal to f at the midpoint (creating many rectangular areas).



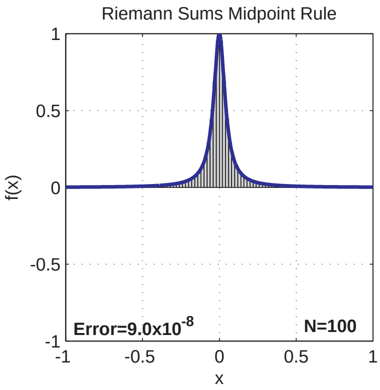
Trapezoidal Approximations

On each of the N subintervals construct a line passing through f at the endpoints (creating many trapezoidal areas).



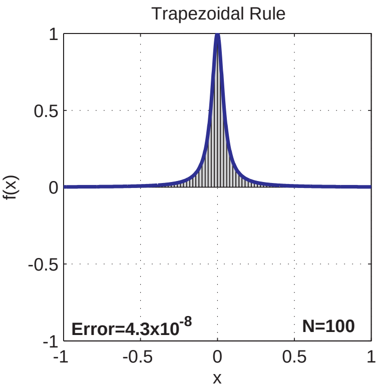
Rectangular Approximations

On each of the N subintervals construct a constant function equal to f at the midpoint (creating many rectangular areas).



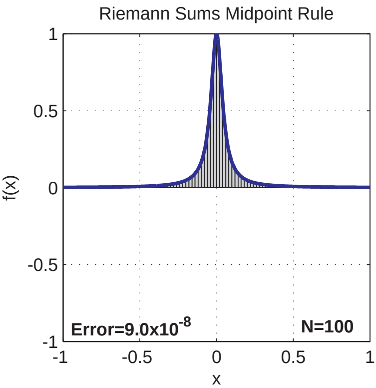
Trapezoidal Approximations

On each of the N subintervals construct a line passing through f at the endpoints (creating many trapezoidal areas).



Rectangular Approximations

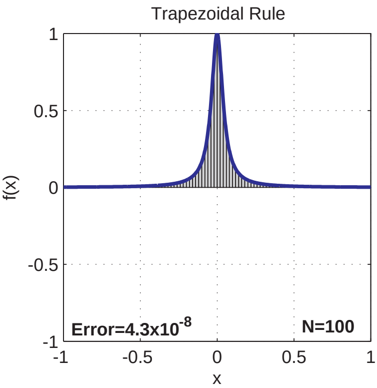
On each of the N subintervals construct a constant function equal to f at the midpoint (creating many rectangular areas).



$$I_{[a,b]}(f) \approx \sum_{i=0}^{N-1} \frac{(b-a)}{N} \cdot f\left(\frac{x_i + x_{i+1}}{2}\right)$$

Trapezoidal Approximations

On each of the N subintervals construct a line passing through f at the endpoints (creating many trapezoidal areas).



$$I_{[a,b]}(f) \approx \frac{(b-a)}{2N} \cdot \left(1 \cdot f(x_0) + \sum_{i=1}^{N-1} 2 \cdot f(x_i) + 1 \cdot f(x_N)\right)$$

Extending the Approximation Idea to Spheres

In either case we approximate the integral in 1D by

$$I_{[a,b]}(f) = \int_a^b f(x)dx \approx \sum_{i=0}^{N} w_i f(x_i),$$

where

- x_i, $i = 0, 1, 2, \ldots, N$, are discrete points in $[a, b]$, and
- w_i depend on
  - the locations of x_i
  - the approximation chosen for f over each subinterval, not on f itself

Goal:

Find W_i, $i = 1, 2, \ldots, N$, such that

$$\mathcal{I}_S(f) = \iint_S f(\mathbf{x})dS \approx \sum_{i=1}^{N} W_i f(\mathbf{x}_i)$$

where

- $\mathbf{x}_i$, $i = 1, 2, \ldots, N$, are discrete points on S
- W_i depend on
  - the locations of the points $\mathbf{x}_i$
  - the approximation chosen for f, not on f itself

- Required accuracy– How many points N do we need?
- Construction of node–sets– How do we place N points on the surface?
- Computational cost/run time– For a given N, how long does it take to find the W_k? Can we distribute the cost to any number of processors?
- Storage requirements– For a given N, how much memory (RAM) is needed to find the W_k?

RBF interpolation of a function $f(\mathbf{x})$ at a set of points $\{\mathbf{x}_i\}_{i=1}^N$ (all in $\mathbb{R}^d$) is accomplished by enforcing

$$f(\mathbf{x}_k) \approx s(\mathbf{x}_k) := \sum_{i=1}^{N} c_i^{RBF} \phi(r_i(\mathbf{x}_k))$$

The interpolant is constructed by solving the linear system

$$A\mathbf{c}^{RBF} := \begin{bmatrix} \phi(r_{11}) & \phi(r_{12}) & \cdots & \phi(r_{1N}) \\ \phi(r_{21}) & \phi(r_{22}) & \cdots & \phi(r_{2N}) \\ \vdots & \vdots & \ddots & \vdots \\ \phi(r_{N1}) & \phi(r_{N2}) & \cdots & \phi(r_{NN}) \end{bmatrix} \begin{bmatrix} c_1^{RBF} \\ c_2^{RBF} \\ \vdots \\ c_N^{RBF} \end{bmatrix} = \begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_N) \end{bmatrix} =: \mathbf{f}$$

(with $r_{ji} = \left\|\mathbf{x}_j - \mathbf{x}_i\right\|_2$), which comes from satisfying the interpolation conditions

- The matrix A can be shown to be nonsingular in the case of GA, MQ, and IMQ RBF interpolants (see, e.g., M.D. Buhmann. "Radial basis functions." *Acta Numerica*, pages 1–38, 2000).
- In the case of even and odd powers the matrix (and interpolation problem) must be supplemented by multivariate polynomial terms.

When multivariate polynomial terms must be included the interpolant is modified to be

$$f(\mathbf{x}) \approx \hat{s}(\mathbf{x}) = \sum_{i=1}^{N} c_i^{RBF} \phi(r_i(\mathbf{x})) + \sum_{l=1}^{M} c_l^m \pi_l(\mathbf{x}),$$

where $\{\pi_l\}_{l=1}^{M}$ is the set of all polynomial terms up to degree m.

In $\mathbb{R}^2$, $M = \frac{(m+1)(m+2)}{2}$ and this set includes the terms

$$\underbrace{1}, \underbrace{x, y}, \underbrace{x^2, xy, y^2}, \underbrace{x^3, x^2y, xy^2, y^3}, \ldots, \underbrace{x^m, x^{m-1}y, x^{m-2}y^2, \ldots, x^2y^{m-2}, xy^{m-1}, y^m}.$$

Once augmented, the linear system becomes

$$\hat{A}\hat{\mathbf{c}} = \begin{bmatrix} A & P^m \\ (P^m)^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{c}^{RBF} \\ \mathbf{c}^m \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{0} \end{bmatrix} = \hat{\mathbf{f}} \qquad \text{where} \qquad \mathbf{c}^m = \begin{bmatrix} c_1^m & c_2^m & \cdots & c_M^m \end{bmatrix}^T$$

and (again, in $\mathbb{R}^2$)

$$P^m = \begin{bmatrix} 1 & x_1 & y_1 & x_1^2 & x_1y_1 & y_1^2 & \cdots & x_1^m & x_1^{m-1}y_1 & x_1^{m-2}y_1^2 & \cdots & x_1^2y_1^{m-2} & x_1y_1^{m-1} & y_1^m \\ 1 & x_2 & y_2 & x_2^2 & x_2y_2 & y_2^2 & \cdots & x_2^m & x_2^{m-1}y_2 & x_2^{m-2}y_2^2 & \cdots & x_2^2y_2^{m-2} & x_2y_2^{m-1} & y_2^m \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ldots & \vdots & \vdots & \vdots & \\ 1 & x_N & y_N & x_N^2 & x_Ny_N & y_N^2 & \cdots & x_N^m & x_N^{m-1}y_N & x_N^{m-2}y_N^2 & \cdots & x_N^2y_N^{m-2} & x_Ny_N^{m-1} & y_N^m \end{bmatrix}.$$

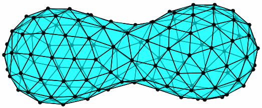
The last M equations come from the conditions

$$\sum_{i=1}^{N} c_i^{RBF} \pi_l(\mathbf{x}_i) = 0, \;\; l = 1, 2, \ldots, M.$$

Given: N nodes on the surface, S, and a (flat) triangulation, T, of the node set

1.
2.
3.
4.
5.

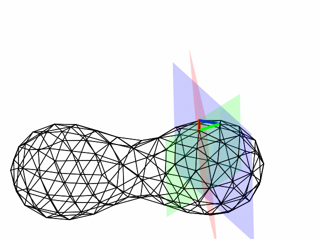
Concept Illustration



Given: N nodes on the surface, S, and a (flat) triangulation, T, of the node set

1. For each of the flat triangles in T, find a projection point.
2.
3.
4.
5.

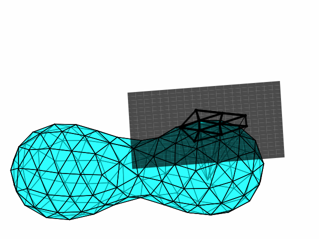
Concept Illustration



Given: N nodes on the surface, S, and a (flat) triangulation, T, of the node set

1. For each of the flat triangles in T, find a projection point.
2. Project a neighborhood (on S) of the three vertices of the triangle into the plane containing the vertices, including n nodes from $\mathcal{S}_N$.
3.
4.
5.

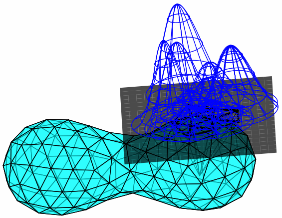
Concept Illustration



Given: N nodes on the surface, S, and a (flat) triangulation, T, of the node set

1. For each of the flat triangles in T, find a projection point.
2. Project a neighborhood (on S) of the three vertices of the triangle into the plane containing the vertices, including n nodes from $\mathcal{S}_N$.
3. Find quadrature weights over the local projected node set for the definite integral over the projected central flat planar triangle.
4.
5.

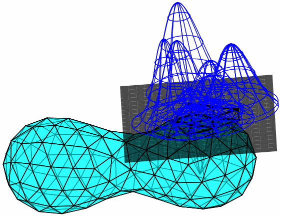
Concept Illustration



Given: N nodes on the surface, S, and a (flat) triangulation, T, of the node set

1. For each of the flat triangles in T, find a projection point.
2. Project a neighborhood (on S) of the three vertices of the triangle into the plane containing the vertices, including n nodes from $\mathcal{S}_N$.
3. Find quadrature weights over the local projected node set for the definite integral over the projected central flat planar triangle.
4. Convert quadrature weights in each plane to corresponding weights for the surface.
5.

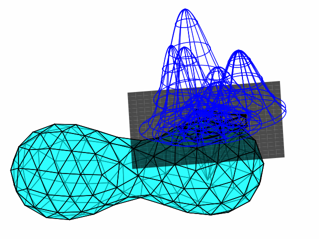
Concept Illustration



Given: N nodes on the surface, S, and a (flat) triangulation, T, of the node set

1. For each of the flat triangles in T, find a projection point.
2. Project a neighborhood (on S) of the three vertices of the triangle into the plane containing the vertices, including n nodes from $\mathcal{S}_N$.
3. Find quadrature weights over the local projected node set for the definite integral over the projected central flat planar triangle.
4. Convert quadrature weights in each plane to corresponding weights for the surface.
5. Combine the weights for the individual curved triangles to obtain the full weight set for the surface.

Concept Illustration

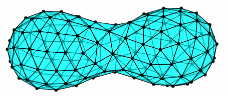


Initialization

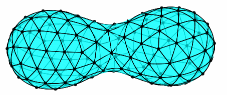
Given a set $\mathcal{S}_N := \{\mathbf{x}_i\}_{i=1}^N \subset S$, we associate to the set $T = \{t_{A_k B_k C_k}\}_{k=1}^K$ of *flat* triangles a set of *curved* triangles, $\mathcal{T} = \{\tau_{A_k B_k C_k}\}_{k=1}^K$, such that

- the curved triangle vertices are the elements of $\mathcal{S}_N$,
- the curved triangle edges are projections of the flat triangle edges to the surface,
- no curved triangle contains an element of $\mathcal{S}_N$ other than its vertices,
- the interiors of the curved triangles are pairwise disjoint, and
- the union of the set $\mathcal{T}$ covers S.

Flat Triangles in $\mathcal{T}$



Curved Triangles in T



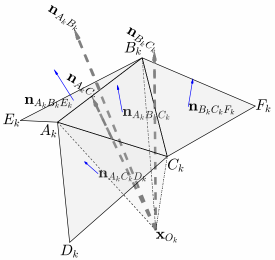
The requirements on $\mathcal{T}$ allow

$$\mathcal{I}_S(f) = \iint\limits_S f(\mathbf{x})dS = \sum_{k=1}^{K} \iint\limits_{\tau_{A_k B_k C_k}} f(\mathbf{x})dS$$

For each edge of a flat triangle in T define a unique "cutting" plane so that both of the two triangles containing a given edge will define the same plane.

The cutting plane along the edge $A_k B_k$ of triangle $t_{A_k B_k C_k}$ is defined to contain the edge and to be parallel to

$$\mathbf{n}_{A_k B_k} = \frac{1}{2}\left(\mathbf{n}_{A_k B_k C_k} + \text{sign}\left(\mathbf{n}^T_{A_k B_k C_k}\mathbf{n}_{A_k B_k E_k}\right)\mathbf{n}_{A_k B_k E_k}\right).$$

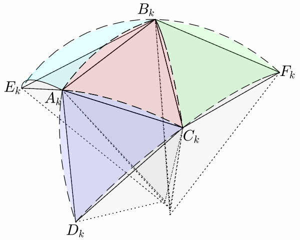


The projection point, $\mathbf{x}_{O_k}$, is the intersection of the three cutting planes

For each edge of a flat triangle in T define a unique "cutting" plane so that both of the two triangles containing a given edge will define the same plane.

The cutting plane along the edge A_kB_k of triangle $t_{A_kB_kC_k}$ is defined to contain the edge and to be parallel to

$$\mathbf{n}_{A_kB_k} = \frac{1}{2}\left(\mathbf{n}_{A_kB_kC_k} + \text{sign}\left(\mathbf{n}^T_{A_kB_kC_k}\mathbf{n}_{A_kB_kE_k}\right)\mathbf{n}_{A_kB_kE_k}\right).$$

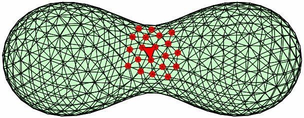


The projection point, $\mathbf{x}_{O_k}$, is the intersection of the three cutting planes

Local projection of a point **x** on S into a plane occurs

in 4 steps

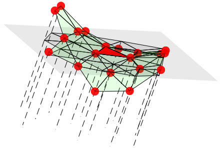
1.
2.
3.
4.



Step 2: Project Locally to a Plane

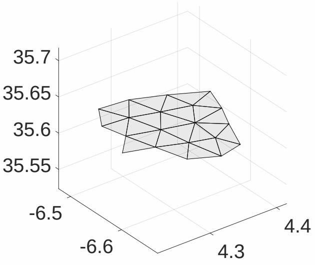
Local projection of a point $\mathbf{x}$ on S into a plane occurs in 4 steps

1. Determine the intersection of the plane containing the triangle and the line through projection point in the direction of $\left(\mathbf{x} - \mathbf{x}_{O_k}\right)$.
2.
3.
4.



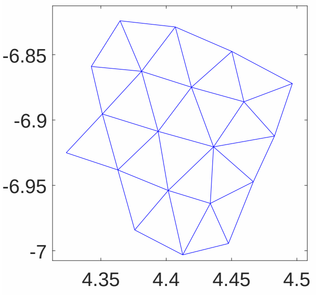
Local projection of a point $\mathbf{x}$ on S into a plane occurs in 4 steps

1. Determine the intersection of the plane containing the triangle and the line through projection point in the direction of $\left(\mathbf{x} - \mathbf{x}_{O_k}\right)$.
2. Translate the projection point to the origin in 3D and rotate the coordinate system so that the normal of the current triangle points vertically.
3.
4.



Local projection of a point $\mathbf{x}$ on S into a plane occurs in 4 steps

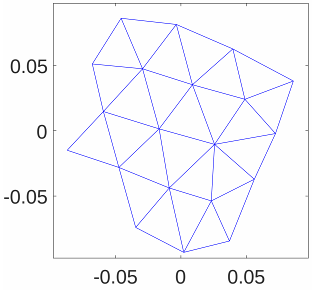
1. Determine the intersection of the plane containing the triangle and the line through projection point in the direction of $\left(\mathbf{x} - \mathbf{x}_{O_k}\right)$.
2. Translate the projection point to the origin in 3D and rotate the coordinate system so that the normal of the current triangle points vertically.
3. Drop the third coordinate.
4.



Step 2: Project Locally to a Plane

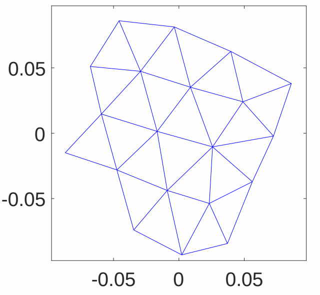
Local projection of a point $\mathbf{x}$ on S into a plane occurs in 4 steps

1. Determine the intersection of the plane containing the triangle and the line through projection point in the direction of $\left(\mathbf{x} - \mathbf{x}_{O_k}\right)$.
2. Translate the projection point to the origin in 3D and rotate the coordinate system so that the normal of the current triangle points vertically.
3. Drop the third coordinate.
4. Translate so that the midpoint of the current triangle is at the origin in 2D.



Local projection of a point $\mathbf{x}$ on S into a plane occurs in 4 steps

1. Determine the intersection of the plane containing the triangle and the line through projection point in the direction of $(\mathbf{x} - \mathbf{x}_{O_k})$.
2. Translate the projection point to the origin in 3D and rotate the coordinate system so that the normal of the current triangle points vertically.
3. Drop the third coordinate.
4. Translate so that the midpoint of the current triangle is at the origin in 2D.



These four steps can be completed using

$$\chi_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} R_k \frac{1}{\mathbf{n}_{A_k B_k C_k} \cdot (\mathbf{x} - \mathbf{x}_{O_k})} \left(\mathbf{n}_{A_k B_k C_k} \times \left(\left(\mathbf{x} - \mathbf{x}_{O_k}\right) \times \left(\mathbf{x}_{M_k} - \mathbf{x}_{O_k}\right)\right)\right)$$

The projection amounts to a local parameterization and a change of variables in the integral over $\tau_{A_k B_k C_k}$, so that

$$\mathcal{I}_S(f) = \sum_{k=1}^{K} \iint_{\tau_{A_k B_k C_k}} f(\mathbf{x}) dS$$

$$= \sum_{k=1}^{K} \iint_{t_{A_k B_k C_k}} f(\mathbf{x}(\boldsymbol{\chi}_k)) \frac{\mathbf{n}_{P_k} \cdot (\mathbf{x}(\boldsymbol{\chi}_k) - \mathbf{x}_{O_k})}{\mathbf{n}_S(\mathbf{x}(\boldsymbol{\chi}_k)) \cdot (\mathbf{x}(\boldsymbol{\chi}_k) - \mathbf{x}_{O_k})} \left(\frac{\mathbf{n}_{A_k B_k C_k} \cdot (\mathbf{x}(\boldsymbol{\chi}_k) - \mathbf{x}_{O_k})}{\mathbf{n}_{A_k B_k C_k} \cdot (\mathbf{x}_{A_k} - \mathbf{x}_{O_k})} \right)^2 dA,$$

where

$$\mathbf{n}_S(\mathbf{x}) := \frac{\nabla h(\mathbf{x})}{\|\nabla h(\mathbf{x})\|_2} \text{ or } \mathbf{n}_S(\mathbf{x}) := \frac{\frac{\partial}{\partial u}\mathbf{x}(u,v) \times \frac{\partial}{\partial v}\mathbf{x}(u,v)}{\left\| \frac{\partial}{\partial u}\mathbf{x}(u,v) \times \frac{\partial}{\partial v}\mathbf{x}(u,v) \right\|_2}$$

and

$$\mathbf{n}_{P_k} := \frac{\mathbf{n}_{A_k B_k C_k}}{\left\| \mathbf{n}_{A_k B_k C_k} \right\|_2}.$$

If a global parameterization, $\mathbf{x}(u, v)$, of S is not available, then the local projection $\mathbf{x}(\boldsymbol{\chi}_k)$ provides a local parameterization that is known at $\mathbf{x}(\boldsymbol{\chi}_{k,j})$, $j = 1, 2, \ldots, n$.

Step 3: Find Quadrature Weights Locally in the Plane

After projecting into the plane we consider evaluating

$$I_{t_{A_k B_k C_k}}(g) := \iint\limits_{t_{A_k B_k C_k}} g(\boldsymbol{\chi}_k)\, dA$$

Notice: integrating the RBF interpolant of $g(\boldsymbol{\chi}_k)$

$$\begin{aligned} I_{t_{A_k B_k C_k}}(g) \approx I_{t_{A_k B_k C_k}}(\hat{s}) =& I_{t_{A_k B_k C_k}}\left(\sum_{j=1}^{n} c_j^{RBF}\phi\left(r_j(\boldsymbol{\chi}_k)\right) + \sum_{l=1}^{M} c_l^{M}\pi_l(\boldsymbol{\chi}_k)\right) \\ =& \sum_{j=1}^{n} c_j^{RBF} I_{t_{A_k B_k C_k}}\left(\phi\left(r_j(\boldsymbol{\chi}_k)\right)\right) + \sum_{l=1}^{M} c_l^{M} I_{t_{A_k B_k C_k}}\left(\pi_l(\boldsymbol{\chi}_k)\right) \\ =& \hat{\mathbf{c}}^T\hat{\mathbf{I}}, \end{aligned}$$

where

$$\hat{\mathbf{I}} = \begin{bmatrix} \mathbf{I}^{RBF} \\ \mathbf{I}^{m} \end{bmatrix},\ \mathbf{I}^{RBF} = \begin{bmatrix} I_{t_{A_k B_k C_k}}(\phi(r_1)) \\ I_{t_{A_k B_k C_k}}(\phi(r_2)) \\ \vdots \\ I_{t_{A_k B_k C_k}}(\phi(r_n)) \end{bmatrix},\ \text{and } \mathbf{I}^{m} = \begin{bmatrix} I_{t_{A_k B_k C_k}}(\pi_1) \\ I_{t_{A_k B_k C_k}}(\pi_2) \\ \vdots \\ I_{t_{A_k B_k C_k}}(\pi_M) \end{bmatrix}.$$

Step 3: Find Quadrature Weights Locally in the Plane

After projecting into the plane we consider evaluating

$$I_{t_{A_k B_k C_k}}(g) := \iint\limits_{t_{A_k B_k C_k}} g(\boldsymbol{\chi}_k)\, dA$$

Notice: integrating the RBF interpolant of $g(\boldsymbol{\chi}_k)$

$$\begin{aligned} I_{t_{A_k B_k C_k}}(g) \approx I_{t_{A_k B_k C_k}}(\hat{s}) =& I_{t_{A_k B_k C_k}}\left(\sum_{j=1}^{n} c_j^{RBF} \phi\left(r_j(\boldsymbol{\chi}_k)\right) + \sum_{l=1}^{M} c_l^{M} \pi_l(\boldsymbol{\chi}_k)\right) \\ =& \sum_{j=1}^{n} c_j^{RBF} I_{t_{A_k B_k C_k}}\left(\phi\left(r_j(\boldsymbol{\chi}_k)\right)\right) + \sum_{l=1}^{M} c_l^{M} I_{t_{A_k B_k C_k}}\left(\pi_l(\boldsymbol{\chi}_k)\right) \\ =& \hat{\mathbf{c}}^T \hat{\mathbf{I}}, \end{aligned}$$

where

$$\hat{\mathbf{I}} = \begin{bmatrix} \mathbf{I}^{RBF} \\ \mathbf{I}^{m} \end{bmatrix}, \mathbf{I}^{RBF} = \begin{bmatrix} I_{t_{A_k B_k C_k}}(\phi(r_1)) \\ I_{t_{A_k B_k C_k}}(\phi(r_2)) \\ \vdots \\ I_{t_{A_k B_k C_k}}(\phi(r_n)) \end{bmatrix}, \text{ and } \mathbf{I}^{m} = \begin{bmatrix} I_{t_{A_k B_k C_k}}(\pi_1) \\ I_{t_{A_k B_k C_k}}(\pi_2) \\ \vdots \\ I_{t_{A_k B_k C_k}}(\pi_M) \end{bmatrix}.$$

Recall that if $\hat{A}$ is invertible, then $\hat{\mathbf{c}} = \hat{A}^{-1}\hat{\mathbf{g}}$. So

$$\begin{aligned} I_{t_{ABC}}(g) \approx I_{t_{ABC}}(\hat{s}) =& \hat{\mathbf{c}}^T \hat{\mathbf{I}} \\ =& \left(\hat{A}^{-1}\hat{\mathbf{g}}\right)^T \hat{\mathbf{I}} \\ =& \hat{\mathbf{g}}^T \left(\left(\hat{A}^{-1}\right)^T \hat{\mathbf{I}}\right) \\ =& \hat{\mathbf{g}}^T \left(\left(\hat{A}^T\right)^{-1} \hat{\mathbf{I}}\right) \\ =& \hat{\mathbf{g}}^T \hat{\mathbf{w}} \end{aligned}$$

Here $\hat{A}^T\hat{\mathbf{w}} = \hat{\mathbf{I}}$ and $\hat{\mathbf{g}} = \left[\begin{array}{cccccccc} g\left(\chi_{k,1}\right) & g\left(\chi_{k,2}\right) & \cdots & g\left(\chi_{k,n}\right) & 0 & \cdots & 0 \end{array}\right]^T$

Let $\mathbf{w}$ be the first n entries in the solution of this system of equations so that we have

$$I_{t_{A_k B_k C_k}}(g) \approx \mathbf{g}^T\mathbf{w} = \sum_{j=1}^{n} w_{k,j}\, g\left(\chi_{k,j}\right).$$

Finding the weights $w_{k,j}$, $j = 1, 2, \ldots, n$ requires two pieces of information

1. The (hopefully closed form) integrals ($j = 1, 2, \ldots, n$ and $l = 1, 2, \ldots, M$)

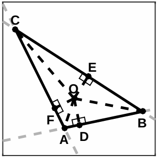
$$I_{t_{A_k B_k C_k}}\left(\phi\left(r_j\right)\right) = \iint\limits_{t_{A_k B_k C_k}} \phi\left(r_j\left(\boldsymbol{\chi}_k\right)\right) dA$$

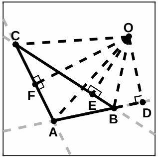
and

$$I_{t_{A_k B_k C_k}}\left(\pi_l\right) = \iint\limits_{t_{A_k B_k C_k}} \pi_l\left(\boldsymbol{\chi}_k\right) dA \text{ (Elementary)}$$

2. The solution of the linear system $\hat{A}^T \hat{\mathbf{w}} = \hat{\mathbf{I}}$

For the RBF terms, consider integration over right triangles only:





Define (minding the order of $A_k B_k C_k$ and likewise for the other triangles)

$$s_{t_{A_k B_k C_k}} := \operatorname{sign}\left(\left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\left(\boldsymbol{\chi}_{k,B_k} - \boldsymbol{\chi}_{k,A_k}\right)\right) \cdot \left(\boldsymbol{\chi}_{k,C_k} - \boldsymbol{\chi}_{k,A_k}\right)\right).$$

The integral over an arbitrary triangle is the sum of integrals over six right triangles:

$$\begin{aligned} I_{t_{A_k B_k C_k}}(\phi) = s_{t_{A_k B_k C_k}} \Big(& s_{t_{O_k A_k D_k}} I_{t_{O_k A_k D_k}}(\phi) + s_{t_{O_k D_k B_k}} I_{t_{O_k D_k B_k}}(\phi) + \\ & s_{t_{O_k B_k E_k}} I_{t_{O_k B_k E_k}}(\phi) + s_{t_{O_k E_k C_k}} I_{t_{O_k E_k C_k}}(\phi) + \\ & s_{t_{O_k C_k F_k}} I_{t_{O_k C_k F_k}}(\phi) + s_{t_{O_k F_k A_k}} I_{t_{O_k F_k A_k}}(\phi) \Big). \end{aligned}$$

Step 3: Find Quadrature Weights Locally in the Plane

Finding the weights $w_{k,j}$, $j = 1, 2, \ldots, n$ requires two pieces of information

1. The (hopefully closed form) integrals ($j = 1, 2, \ldots, n$ and $l = 1, 2, \ldots, M$)

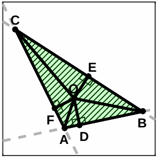
$$I_{t_{A_k B_k C_k}}\left(\phi\left(r_j\right)\right) = \iint\limits_{t_{A_k B_k C_k}} \phi\left(r_j\left(\boldsymbol{\chi}_k\right)\right) dA$$

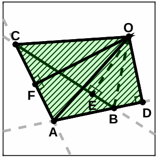
and

$$I_{t_{A_k B_k C_k}}\left(\pi_l\right) = \iint\limits_{t_{A_k B_k C_k}} \pi_l\left(\boldsymbol{\chi}_k\right) dA \text{ (Elementary)}$$

2. The solution of the linear system $\hat{A}^T \hat{\mathbf{w}} = \hat{\mathbf{I}}$

For the RBF terms, consider integration over right triangles only:





Define (minding the order of $A_k B_k C_k$ and likewise for the other triangles)

$$s_{t_{A_k B_k C_k}} := \operatorname{sign}\left(\left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\left(\boldsymbol{\chi}_{k,B_k} - \boldsymbol{\chi}_{k,A_k}\right)\right) \cdot \left(\boldsymbol{\chi}_{k,C_k} - \boldsymbol{\chi}_{k,A_k}\right)\right).$$

The integral over an arbitrary triangle is the sum of integrals over six right triangles:

$$\begin{aligned} I_{t_{A_k B_k C_k}}(\phi) = s_{t_{A_k B_k C_k}} \Big(& s_{t_{O_k A_k D_k}} I_{t_{O_k A_k D_k}}(\phi) + s_{t_{O_k D_k B_k}} I_{t_{O_k D_k B_k}}(\phi) + \\ & s_{t_{O_k B_k E_k}} I_{t_{O_k B_k E_k}}(\phi) + s_{t_{O_k E_k C_k}} I_{t_{O_k E_k C_k}}(\phi) + \\ & s_{t_{O_k C_k F_k}} I_{t_{O_k C_k F_k}}(\phi) + s_{t_{O_k F_k A_k}} I_{t_{O_k F_k A_k}}(\phi) \Big). \end{aligned}$$

Finding the weights $w_{k,j}$, $j = 1, 2, \ldots, n$ requires two pieces of information

1. The (hopefully closed form) integrals ($j = 1, 2, \ldots, n$ and $l = 1, 2, \ldots, M$)

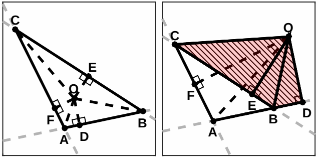
$$I_{t_{A_k B_k C_k}}\left(\phi\left(r_j\right)\right) = \iint\limits_{t_{A_k B_k C_k}} \phi\left(r_j\left(\boldsymbol{\chi}_k\right)\right) dA$$

and

$$I_{t_{A_k B_k C_k}}\left(\pi_l\right) = \iint\limits_{t_{A_k B_k C_k}} \pi_l\left(\boldsymbol{\chi}_k\right) dA \text{ (Elementary)}$$

2. The solution of the linear system $\hat{A}^T \hat{\mathbf{w}} = \hat{\mathbf{I}}$

For the RBF terms, consider integration over right triangles only:



Define (minding the order of $A_k B_k C_k$ and likewise for the other triangles)

$$s_{t_{A_k B_k C_k}} := \operatorname{sign}\left(\left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\left(\boldsymbol{\chi}_{k,B_k} - \boldsymbol{\chi}_{k,A_k}\right)\right) \cdot \left(\boldsymbol{\chi}_{k,C_k} - \boldsymbol{\chi}_{k,A_k}\right)\right).$$

The integral over an arbitrary triangle is the sum of integrals over six right triangles:

$$\begin{aligned} I_{t_{A_k B_k C_k}}(\phi) =& s_{t_{A_k B_k C_k}} \left(s_{t_{O_k A_k D_k}} I_{t_{O_k A_k D_k}}(\phi) + s_{t_{O_k D_k B_k}} I_{t_{O_k D_k B_k}}(\phi) + \right. \\ & s_{t_{O_k B_k E_k}} I_{t_{O_k B_k E_k}}(\phi) + s_{t_{O_k E_k C_k}} I_{t_{O_k E_k C_k}}(\phi) + \\ & \left. s_{t_{O_k C_k F_k}} I_{t_{O_k C_k F_k}}(\phi) + s_{t_{O_k F_k A_k}} I_{t_{O_k F_k A_k}}(\phi) \right). \end{aligned}$$

Finding the weights $w_{k,j}$, $j = 1, 2, \ldots, n$ requires two pieces of information

1. The (hopefully closed form) integrals ($j = 1, 2, \ldots, n$ and $l = 1, 2, \ldots, M$)

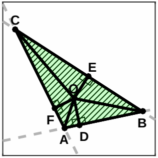
$$I_{t_{A_k B_k C_k}}\left(\phi\left(r_j\right)\right) = \iint\limits_{t_{A_k B_k C_k}} \phi\left(r_j\left(\boldsymbol{\chi}_k\right)\right) dA$$

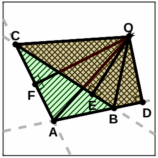
and

$$I_{t_{A_k B_k C_k}}\left(\pi_l\right) = \iint\limits_{t_{A_k B_k C_k}} \pi_l\left(\boldsymbol{\chi}_k\right) dA \text{ (Elementary)}$$

2. The solution of the linear system $\hat{A}^T \hat{\mathbf{w}} = \hat{\mathbf{I}}$

For the RBF terms, consider integration over right triangles only:



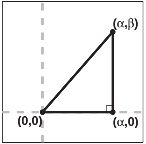


Define (minding the order of $A_k B_k C_k$ and likewise for the other triangles)

$$s_{t_{A_k B_k C_k}} := \operatorname{sign}\left(\left(\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}\left(\boldsymbol{\chi}_{k,B_k} - \boldsymbol{\chi}_{k,A_k}\right)\right) \cdot \left(\boldsymbol{\chi}_{k,C_k} - \boldsymbol{\chi}_{k,A_k}\right)\right).$$

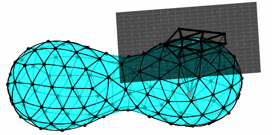
The integral over an arbitrary triangle is the sum of integrals over six right triangles:

$$\begin{aligned} I_{t_{A_k B_k C_k}}(\phi) = s_{t_{A_k B_k C_k}} \Big(& s_{t_{O_k A_k D_k}} I_{t_{O_k A_k D_k}}(\phi) + s_{t_{O_k D_k B_k}} I_{t_{O_k D_k B_k}}(\phi) + \\ & s_{t_{O_k B_k E_k}} I_{t_{O_k B_k E_k}}(\phi) + s_{t_{O_k E_k C_k}} I_{t_{O_k E_k C_k}}(\phi) + \\ & s_{t_{O_k C_k F_k}} I_{t_{O_k C_k F_k}}(\phi) + s_{t_{O_k F_k A_k}} I_{t_{O_k F_k A_k}}(\phi) \Big). \end{aligned}$$

Integrals of Popular RBFs Over Right Triangles	
$\phi(r)$	$\displaystyle\int_0^{\alpha}\int_0^{\frac{\beta}{\alpha}\begin{bmatrix}1 & 0\end{bmatrix}\boldsymbol{x}_k} \phi\left(\lVert\boldsymbol{x}_k\rVert_2\right) dA$ 
r^3	$\frac{1}{40}\alpha\left(3\alpha^4\sinh^{-1}\left(\frac{\beta}{\alpha}\right)+\beta\sqrt{\alpha^2+\beta^2}\left(5\alpha^2+2\beta^2\right)\right)$
r^5	$\frac{1}{336}\alpha\left(15\alpha^6\sinh^{-1}\left(\frac{\beta}{\alpha}\right)+\beta\sqrt{\alpha^2+\beta^2}\left(33\alpha^4+26\alpha^2\beta^2+8\beta^4\right)\right)$
$r^2\ln(r)$	$\frac{1}{144}\alpha\left(24\alpha^3\tan^{-1}\left(\frac{\beta}{\alpha}\right)+6\beta\left(3\alpha^2+\beta^2\right)\ln\left(\alpha^2+\beta^2\right)-33\alpha^2\beta-7\beta^3\right)$
$e^{-(\epsilon r)^2}$	[illegible]
$\sqrt{1+(\epsilon r)^2}$	… $-i\left(\ln\left(\frac{\beta\left(\sqrt{\epsilon^2(\alpha^2+\beta^2)+1}-\beta\epsilon\right)+\alpha^2(-\epsilon)+i\alpha}{\alpha\epsilon-i}\right)+\ln\left(\frac{\beta\left(\sqrt{\epsilon^2(\alpha^2+\beta^2)+1}+\beta\epsilon\right)+\alpha^2\epsilon+i\alpha}{\alpha\epsilon+i}\right)\right)\Bigg]$
$\frac{1}{\sqrt{1+(\epsilon r)^2}}$	$\frac{1}{\epsilon^2}\left[-\tan^{-1}\left(\frac{\alpha\sqrt{\epsilon^2(\alpha^2+\beta^2)+1}}{\beta}\right)+\alpha\epsilon\sinh^{-1}\left(\frac{\beta\epsilon}{\sqrt{\alpha^2\epsilon^2+1}}\right)+\tan^{-1}\left(\frac{\alpha}{\beta}\right)\right]$

Closed form expressions exist!

When approximating $g(\boldsymbol{\chi}_k)$ we interpolate over n points $\{\boldsymbol{\chi}_{k,j}\}_{j=1}^{n}$



These points are taken to be the projections of the n nearest quadrature nodes (in $\mathcal{S}_N$) to the midpoint of $t_{A_k B_k C_k}$.

For each midpoint, the n nearest neighbors in Euclidean distance are found at the initialization of the proposed method using the kd-tree algorithm in $O(N\ n \log N)$ operations (see e.g., J. H. Friedman, J. L. Bentley, and R. A. Finkel. "An algorithm for finding best matches in logarithmic expected time." *ACM Trans Math Softw*, 3(3), 1977).

Applying step 3 to the double integral over each of the triangles $t_{A_kB_kC_k}$ gives (with $\mathbf{x}_{k,j} := \mathbf{x}(\chi_k)$)

$$\iint_{\tau_{A_kB_kC_k}} f(\mathbf{x})dS = \iint_{t_{A_kB_kC_k}} f(\mathbf{x}(\chi_k)) \frac{\mathbf{n}_{P_k} \cdot (\mathbf{x}(\chi_k) - \mathbf{x}_{O_k})}{\mathbf{n}_S(\mathbf{x}(\chi_k)) \cdot (\mathbf{x}(\chi_k) - \mathbf{x}_{O_k})} \left(\frac{\mathbf{n}_{A_kB_kC_k} \cdot (\mathbf{x}(\chi_k) - \mathbf{x}_{O_k})}{\mathbf{n}_{A_kB_kC_k} \cdot (\mathbf{x}_{A_k} - \mathbf{x}_{O_k})} \right)^2 dA$$

$$\approx \sum_{j=1}^{n} w_{k,j} f(\mathbf{x}_{k,j}) \frac{\mathbf{n}_{P_k} \cdot (\mathbf{x}_{k,j} - \mathbf{x}_{O_k})}{\mathbf{n}_S(\mathbf{x}_{k,j}) \cdot (\mathbf{x}_{k,j} - \mathbf{x}_{O_k})} \left(\frac{\mathbf{n}_{A_kB_kC_k} \cdot (\mathbf{x}_{k,j} - \mathbf{x}_{O_k})}{\mathbf{n}_{A_kB_kC_k} \cdot (\mathbf{x}_{A_k} - \mathbf{x}_{O_k})} \right)^2 .$$

Hence,

$$\mathcal{I}_S(f) \approx \sum_{k=1}^{K} \sum_{j=1}^{n} w_{k,j}^{\text{RBF}} f(\mathbf{x}_{k,j}) \frac{\mathbf{n}_{P_k} \cdot (\mathbf{x}_{k,j} - \mathbf{x}_{O_k})}{\mathbf{n}_S(\mathbf{x}_{k,j}) \cdot (\mathbf{x}_{k,j} - \mathbf{x}_{O_k})} \left(\frac{\mathbf{n}_{A_kB_kC_k} \cdot (\mathbf{x}_{k,j} - \mathbf{x}_{O_k})}{\mathbf{n}_{A_kB_kC_k} \cdot (\mathbf{x}_{A_k} - \mathbf{x}_{O_k})} \right)^2 .$$

Weights in the plane and on the surface differ only by a factor.

Let $\mathcal{K}_i$, $i = 1, 2, \ldots, N$ be the set of all pairs (k, j) such that $\chi_{k,j} \mapsto \mathbf{x}_i$. Then the surface integral over S can be written as

$$\mathcal{I}_S(f) \approx \sum_{i=1}^{N} \left(\sum_{(k,j)\in\mathcal{K}_i} w_{k,j} \frac{\mathbf{n}_{P_k} \cdot (\mathbf{x}_{k,j} - \mathbf{x}_{O_k})}{\mathbf{n}_S(\mathbf{x}_{k,j}) \cdot (\mathbf{x}_{k,j} - \mathbf{x}_{O_k})} \left(\frac{\mathbf{n}_{A_kB_kC_k} \cdot (\mathbf{x}_{k,j} - \mathbf{x}_{O_k})}{\mathbf{n}_{A_kB_kC_k} \cdot (\mathbf{x}_{A_k} - \mathbf{x}_{O_k})} \right)^2 \right) f(\mathbf{x}_i) = \sum_{i=1}^{N} W_i f(\mathbf{x}_i) =: \tilde{\mathcal{I}}_{\mathbb{S}^2}(f).$$

Results are presented for quasi-uniform node sets on three test surfaces and three test integrands.

Note: Present method default settings when computing quadrature weights for each spherical triangle:

- $\phi(r) = r^7$
- 80 nearest neighbors
- Bivariate polynomial terms up to degree 7

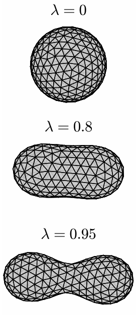
Worst case error shown when the test function has been randomly rotated $1,000$ times.

We consider the rotated Cassini Ovals defined by

$$h(\mathbf{x}) = h(x,y,z) = (x^2+y^2+z^2)^2 - 2\lambda^2 b^2 (x^2 - y^2 - z^2) + b^4(\lambda^4 - 1) = 0$$

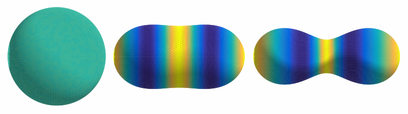
which can also be parameterized explicitly via

$$\begin{aligned} x(\theta,\phi) &= \rho(\phi)\cos(\phi) \\ y(\theta,\phi) &= \rho(\phi)\sin(\phi)\sin(\theta) \\ z(\theta,\phi) &= \rho(\phi)\sin(\phi)\cos(\theta) \\ \rho(\phi) &= b\sqrt{\sqrt{1+\lambda^4(\cos^2(2\phi)-1)} + \lambda^2\cos(2\phi)} \\ \theta &\in [0,2\pi) \\ \phi &\in [0,\pi] \end{aligned}$$



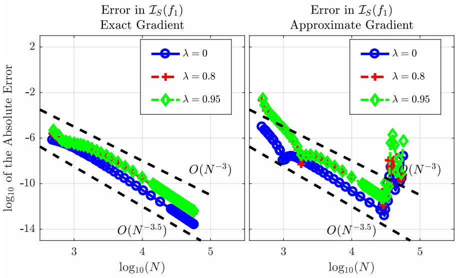
In all cases b is chosen so that the surface area is equal to 1 by finding numerically the root of

$$R(b) = 1 - 8 \int_0^{b\sqrt{\lambda^2+1}} \int_0^{\sqrt{b\sqrt{b^2+4\lambda^2x^2} - \lambda^2 b^2 - x^2}} \sqrt{\frac{\lambda^2 b^6 - b^3(b^2+4\lambda^2x^2)^{\frac{3}{2}} + 4\lambda^2 b^3 x^2 \sqrt{b^2+4\lambda^2x^2}}{\lambda^2 b^6 - b^3(b^2+4\lambda^2x^2)^{\frac{3}{2}} + 4\lambda^2 b^2 x^4 + 4\lambda^4 b^4 x^2 + b^4x^2 + b^4y^2 + 4\lambda^2 b^2 x^2 y^2}}\, dy dx$$



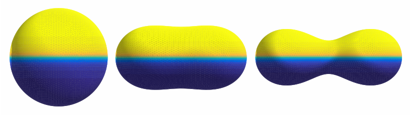
Integrand:

$$f_1(\mathbf{x}) = \frac{1}{3}\mathbf{x} \cdot \frac{\nabla h(\mathbf{x})}{\|\nabla h(\mathbf{x})\|_2}$$



Features: Infinitely smooth, computes the volume of the surface of revolution.

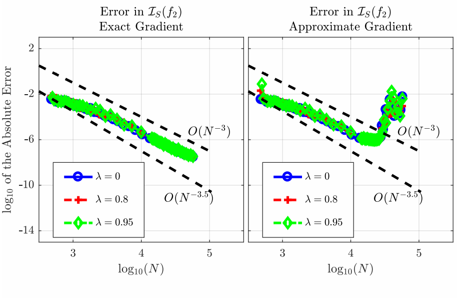
Test Integrands

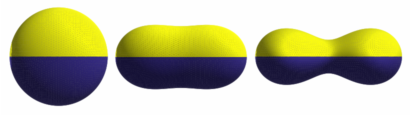


Integrand:

$$f_2(\mathbf{x}) = \frac{2}{\pi}\tan^{-1}(100\mathbf{e}_3^T\mathbf{x})$$

Features: Infinitely smooth with a steep gradient near where the third coordinate of **x** is zero.

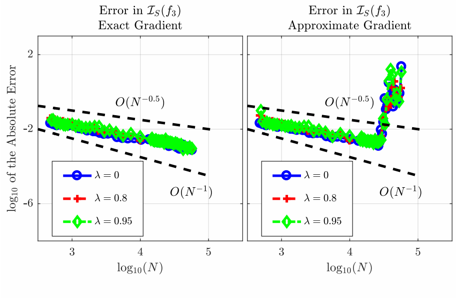




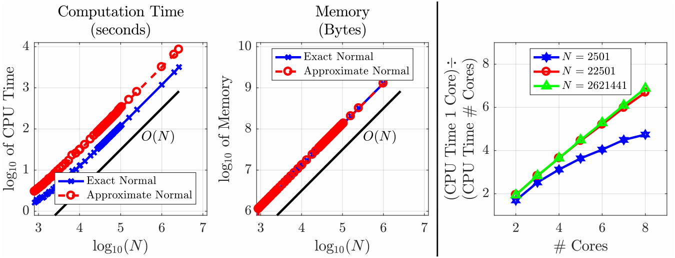
Integrand:

$$f_3(\mathbf{x}) = \text{sign}(\mathbf{e}_3^T \mathbf{x})$$

Features: Discontinuous where the third coordinate of **x** is zero.



All computations were performed in Matlab on machines with dual Intel Xeon E5-2687W 3.1 GHz, 8-core processors.



Figures indicate for the proposed method:

- $O(N)$ cost and memory for up to at least millions of nodes
- $O(N \log N)$ cost is expected when the nearest neighbor search starts to dominate the computation.
- 'Embarrassingly parallel' scalability with number of cores

Conclusions

- A high order accurate algorithm has been developed for quadrature over smooth closed surfaces
- The node sets can feature any types of density variations (e.g. local refinement in certain areas, etc.), demonstrated for the sphere in "Numerical quadrature over the surface of a sphere" (J.A. Reeger and B. Fornberg)
- The total cost is $O(N \log N)$ operations and $O(N)$ memory for finding weights for N nodes.
- The algorithm is 'embarrassingly parallel', making it trivial to use any number of available processors.
- Even on a standard PC, it can be run for N-values in the millions. This eliminates the need for tabulating weights for specific node distributions.

Publications:

- **J. A. Reeger** and B. Fornberg. "Numerical quadrature over the surface of a sphere." *Stud. Appl. Math.*, 137(2): 174-188, 2016.
- **J. A. Reeger**, B. Fornberg, and M. L. Watts. "Numerical quadrature over smooth, closed surfaces." *P. Roy. Soc. Lon. A Mat.*, 472:20160401, 2016. (doi:10.1098/rspa.2016.0401).
- **J. A. Reeger** and B. Fornberg. "Numerical quadrature over smooth surfaces with boundaries." submitted to *J. Comput. Phys.*

Implementations Available: http://www.jonahareeger.com (MATLAB, Julia, and Python)

- A high order accurate algorithm has been developed for quadrature over smooth closed surfaces
- The node sets can feature any types of density variations (e.g. local refinement in certain areas, etc.), demonstrated for the sphere in “Numerical quadrature over the surface of a sphere” (J.A. Reeger and B. Fornberg)
- The total cost is $O(N \log N)$ operations and $O(N)$ memory for finding weights for N nodes.
- The algorithm is ‘embarrassingly parallel’, making it trivial to use any number of available processors.
- Even on a standard PC, it can be run for N-values in the millions. This eliminates the need for tabulating weights for specific node distributions.

Publications:

- **J. A. Reeger** and B. Fornberg. “Numerical quadrature over the surface of a sphere.” *Stud. Appl. Math.*, 137(2): 174-188, 2016.
- **J. A. Reeger**, B. Fornberg, and M. L. Watts. “Numerical quadrature over smooth, closed surfaces.” *P. Roy. Soc. Lon. A Mat.*, 472:20160401, 2016. (doi:10.1098/rspa.2016.0401).
- **J. A. Reeger** and B. Fornberg. “Numerical quadrature over smooth surfaces with boundaries.” submitted to *J. Comput. Phys.*

Implementations Available: http://www.jonahareeger.com (MATLAB, Julia, and Python)