Journal of Computational Physics 355 (2018) 176–190

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Numerical quadrature over smooth surfaces with boundaries

Jonah A. Reeger [a,*,1], Bengt Fornberg [b]

[a] Air Force Institute of Technology, Department of Mathematics and Statistics, 2950 Hobson Way, Wright-Patterson Air Force Base, OH 45433-7765, USA
[b] University of Colorado, Department of Applied Mathematics, 526 UCB, Boulder, CO 80309, USA

ARTICLE INFO

Article history:
Received 2 August 2017
Received in revised form 8 November 2017
Accepted 9 November 2017
Available online 15 November 2017

Keywords:
Quadrature
Radial basis function
RBFs
RBF-FD
Gregory's method
Trapezoidal rule

ABSTRACT

This paper describes a high order accurate method to calculate integrals over curved surfaces with boundaries. Given data locations that are arbitrarily distributed over the surface, together with some functional description of the surface and its boundary, the algorithm produces matching quadrature weights. This extends on the authors' earlier methods for integrating over the surface of a sphere and over arbitrarily shaped smooth closed surfaces by also considering domain boundaries. The core approach consists again of combining RBF-FD (radial basis function-generated finite difference) approximations for curved surface triangles, which together make up the full surface. The provided examples include both curved and flat domains. In the highly special case of equi-spaced nodes over a regular interval in 1-D, the method provides a new opportunity for improving on the classical Gregory enhancements of the trapezoidal rule.

Published by Elsevier Inc.

1. Introduction

Algorithms for numerical quadrature typically determine weights, after which the evaluation of integrals becomes a matter of multiplying these with actual function values, and adding the results. In some cases (such as with Gaussian quadrature), the nodes (data locations) have to be chosen in a very special way. In applications, numerical quadrature is usually a follow-up to some other task (such as collecting data, or numerically solving PDEs), making it impractical to require node locations that are specific to the quadrature method. The present algorithm is therefore designed to find the quadrature weights at whatever node locations that are specified.

For N scattered nodes over the surface of a sphere, the algorithm described in [1] offers spectral accuracy, but at the relatively high cost of $O(N^3)$ operations and $O(N^2)$ memory. The RBF (radial basis function)-based method in [2] gives quadrature errors of size $O(1/N^2)$ (or equivalently $O(h^4)$ if h is a typical node separation distance) at costs of $O(N^2)$ for both operation count and memory. The first method in the authors' recent quadrature investigations [3] achieves $O(1/N^{3.5})$ (equivalent to $O(h^7)$) accuracy at the much reduced costs of $O(N\ \log N)$ and $O(N)$, respectively, as was also the case for the subsequent generalization from a sphere to arbitrarily shaped smooth closed surfaces [4]. These convergence rate and cost estimates hold again for the present further generalization to curved surfaces with smooth boundaries.

* Corresponding author.
E-mail address: jonah.reeger@afit.edu (J.A. Reeger).

[1] Major, United States Air Force, supported by the Department of Defense and the Office of Naval Research's Atmospheric Propagation Sciences of High Energy Lasers and the Air Force Office of Scientific Research's Radial Basis Functions for Numerical Simulation. The views expressed in this article are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or U.S. Government.

https://doi.org/10.1016/j.jcp.2017.11.010
0021-9991/Published by Elsevier Inc.

The core numerical method used in this paper is known as RBF-FD (radial basis function-generated finite differences). This approach has so far mostly been used to approximate partial derivatives, with the key difference to regular finite differences that the node points no longer need to be grid-based (in particular, Cartesian node layouts are now known to be less-than-optimal [5], which can also be seen in Figs. 10 and 11). For surveys of RBFs and of RBF-FD methods as these are applied to PDEs, see [6], [7].

The following Section 2 describes the present quadrature method. This starts with decomposing the surface into curved 'surface triangles' which, between them, cover the full surface without any overlaps. Each of these surface triangles is then projected to a local tangent plane, in which the RBF-FD approach provides accurate quadrature weights. The integrals required in this step can be evaluated in closed form for triangles that do not share a side with the domain boundary. For the ones that do, a line integral is instead evaluated numerically. Another key component of the algorithm is a formula for converting quadrature weights in the tangent plane to corresponding weights at the surface nodes. Section 3 describes some test examples, with illustrations of convergence rates and computational costs. A Matlab implementation of the method is available at Matlab Central's File Exchange [8].

To better motivate certain aspects of the implementation, especially focusing on the accuracy at the boundaries (for which the surface curvature is not a critical aspect), we turn in Section 4 to some test cases for flat bounded domains. We note in particular that it is beneficial at boundaries to use larger projected regions than what are needed for interior triangles (consistent with similar observations in the context of using RBF-FD for PDEs [9]).

The case for which the effect of boundaries on quadrature rules have been studied most thoroughly in the literature (and certainly for the longest time) is the enhancement to the trapezoidal rule that was developed by James Gregory in 1670 [10] (remarkably, before Leibnitz' and Newton's first publications on the topic of calculus, in 1684 and 1687, respectively). A recent discussion of these Gregory end corrections can be found in [11]. When applying the present RBF-FD method to this case of equispaced nodes in 1-D, it can not only match but will typically even improve on the Gregory procedure. Results of the present method applied to 1-D are given in section 5.

2. Description of the key steps in the algorithm

Consider computing ($\mathbf{x} \in \mathbb{R}^3$)

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

where S is a finite surface defined implicitly. That is, $\mathbf{x} \in S$ if both $h(\mathbf{x}) = 0$ and $b(\mathbf{x}) \geq 0$ (see Fig. 3 for examples). These equations define a surface with a boundary curve satisfying the initial value problem

$$\frac{d}{ds}\mathbf{x}(s) = \frac{\nabla h(\mathbf{x}(s)) \times \nabla b(\mathbf{x}(s))}{\|\nabla h(\mathbf{x}(s)) \times \nabla b(\mathbf{x}(s))\|_2} \tag{1a}$$

$$\mathbf{x}(0) = \mathbf{x}_0 \tag{1b}$$

where s is arc-length along the boundary curve and $\mathbf{x}_0$ is a point on the boundary curve. In this definition, both the surface h and the boundary b are described as level surfaces in such a way that the inequality for b defines whether the surface is "above" or "below" the boundary curve. Other definitions of the surface could also be used. For instance, any combination of explicit and implicit parameterizations for the surface and boundary, or boundary curve, could be used.

Here the approach described in [4] for computing quadrature weights for surface integrals will be extended to surfaces like S. In this sequel, it is still assumed that a set, $\mathcal{S}_N$, of N quadrature nodes on the surface and a triangulation $T = \{t_{A_k B_k C_k}\}_{k=1}^{K}$ (which maps one-to-one onto the set $\mathcal{T} = \{\tau_{A_k B_k C_k}\}_{k=1}^{K}$ of curved "surface triangles" that covers S) are provided by the user. Since the method in [4] considers each triangle in $\mathcal{T}$ separately so that

$$\mathcal{I}_S(f) = \sum_{k=1}^{K} \iint_{\tau_{A_k B_k C_k}} f(\mathbf{x}) dS \tag{2}$$

this generalization requires only that "boundary triangles" (with at least one edge, i.e. at least two vertices on the boundary) be handled differently from "interior triangles". The method in [4] can be used without specifying $h(x, y, z)$ by approximating normal vectors to the surface. Likewise, the present method could be adapted so that only a set of nodes, a triangulation of the set of nodes, and an ordered list of the nodes on the boundary are required from the user.

This method for determining quadrature weights for approximating the surface integral can be summarized in the same five steps as in [4]:

1. For each of the triangles in $\mathcal{T}$, find a projection point.
2. From the projection point, project a neighborhood of the three vertices of the triangle (points in $\mathcal{S}_N$) on S into the plane containing the corresponding triangle in T. This neighborhood will include the $n - 3$ nodes nearest to the triangles midpoint and beyond the three vertices.