SHAPE AWARE QUADRATURES

by

Vaidyanathan Thiagarajan

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Mechanical Engineering)

at the

UNIVERSITY OF WISCONSIN–MADISON

2017

Date of final oral examination: 12/02/2016

Acknowledgements

I would like to express my heartfelt gratitude to my advisor, Prof. Vadim Shapiro, for being a phenomenal mentor and support throughout my PhD. I would also like to thank him for giving me uninhibited freedom in exploring the research topics of my interest. His positive energy and enthusiasm has always been very infectious. It is this energy and enthusiasm that kept me going in spite of the several setbacks that I have had over the last 6 years.

I would like to thank Prof. Catalin Picu of Rensselaer Polytechnic Institute for teaching me advanced courses in mechanics. His classes gave me the necessary mathematical background to attempt a thesis of this nature. Special thanks to Prof. Krishnan Suresh, Prof. Mario Trujillo, Prof. Eftychios Sifakis, and Prof. Xiaoping Qian for agreeing to be on my committee and for their helpful suggestions/questions.

Many thanks to Nils Zander of Technische Universität München for patiently responding to my emails on FCMLab. Likewise, sincere gratitude to Dr. S.E. Mousavi of University of Texas at Austin for the helpful email exchanges and discussions on moment fitting equations.

I would always remember and cherish all the fun I had working with my colleagues at Spatial Automation Lab (SAL). Special thanks to Saigopal Nelaturi, Mikola Lysenko, Goldy Kumar, Xingchen Liu, Brian McCarthy, Randi Wang, and Yaqi Zhang of SAL for all those interesting/stimulating discussions over the course of my PhD.

My sincere gratitude to my brother and sister-in-law for taking good care of our aging parents over the last several years. Without their support and encouragement I could not have imagined pursuing this for 6 long years away from my home. I am immensely grateful and indebted to my parents for their unconditional love and innumerable sacrifices that they have made for making me who I am today.

Words cannot express how grateful I am to my spiritual master Sadhguru Vasudev of Isha Foundation. Daily practice of the self-transformational tools that he offered in the form of various yogic practices helped me to remain balanced and stay focused during testing times. More importantly, this work would not have been possible without his grace and blessings.

Contents

1 Introduction **1**

1.1 Motivation . . . 1

1.2 Challenges in Numerical Integration . . . 5

1.3 Problem Statement . . . 8

1.4 Overview . . . 8

2 Background **11**

2.1 Volumetric Integration . . . 11

2.1.1 Polygonal/Polyhedral Domains . . . 12

2.1.2 Adaptation by Hierarchical Partitioning . . . 15

2.1.3 Adaptation by Characteristic Function . . . 17

2.1.4 Adaptation by Shape Sensitivity Analysis . . . 23

2.1.5 Summary . . . 25

2.1.6 Open Issues . . . 26

2.2 Sensitivity Analysis . . . 28

2.2.1 Shape Sensitivity . . . 29

2.2.2 Topological Sensitivity . . . 32

2.2.3 Feature Sensitivity . . . 33

2.2.4 Modification Sensitivity . . . 37

2.2.5 Summary . . . 40

3 Shape Aware Quadratures (SAQ) 41

3.1 Small Features 41

3.2 Formulation 44

3.3 Sensitivity Analysis of Moments 50

3.3.1 SSA of Moments 51

3.3.2 TSA of Moments 55

3.4 Algorithm 58

3.5 Algorithm Analysis 62

3.6 Experimental Validation 64

3.6.1 Example 1 - Quadrant of a circle 67

3.6.2 Example 2 - Notched Domain 71

3.6.3 Example 3 - Wavy domain 73

3.6.4 Example 4 - Non-Convex domain with small holes 80

3.6.5 Example 5 - Unit Sphere 81

3.6.6 Example 6 - Microstructures 87

3.7 Summary 89

4 SAQ in Finite Cell Method 92

4.1 Finite Cell Method 92

4.2 Implementation 97

4.2.1 2D Implementation 97

4.2.2 3D Implementation 102

4.3 Convergence Studies 107

4.3.1 Annular Ring 107

4.3.2 Cylinder under uniaxial compression 112

4.4 Summary 117

5 *A priori* and *a posteriori* error estimation of SAQ **119**

5.1 Background 119

5.2 Sources of error 121

5.2.1 Insufficient/inappropriate choice of basis functions 121

5.2.2 Insufficient number of domain quadrature points 122

5.2.3 Inappropriate location of domain quadrature points 122

5.2.4 Inaccurate Moment Computations 123

5.2.5 Fixed precision arithmetic 125

5.3 *A priori* error analysis 126

5.3.1 Example 1 - Quadrant of a Circle 133

5.3.2 Example 2 - Family of noisy domains 138

5.4 *A posteriori* error analysis 141

5.4.1 Example 3 - Quadtree 143

5.5 Summary 146

6 Contributions and Open Issues **148**

Appendices **167**

A SSA of Moments 168

B TSA of Moments 173

C Taylor's Remainder Theorem 180

D Eigen Value Lemma 182

E Computed Basis Function Values 184

List of Figures

1.1 Meshing failure for a perforated plate with 1,521 holes of size 1 mm in Solidworks 1

1.2 The fictitious domain approach : the physical domain Ω is extended by the fictitious domain Ω_{fict} into an embedding domain Ω to allow easy meshing of complex geometries. The influence of Ω_{fict} is penalized by α [118] 3

1.3 Illustration of excessive boundary cell fragmentation in quadtree based integration [118] 6

1.4 Features smaller than the integration grid resolution are missed 7

1.5 Capturing the effect of small features by (a) uniform subdivision (b) overlaying of fine grid and (c) non-uniform adaptive subdivision 7

2.1 Hierarchical partitioning using quadtrees 17

2.2 Integrand adaptivity 17

2.3 The supporting area is partitioned according to its contribution to the material or void [12] 19

2.4 Each Gauss point is replaced by two new points at the centroids of the partitioned polygons [12] 20

2.5 Integration cell with linear discontinuous interface 21

2.6 Reference and Deformed domains for SSA 24

2.7 Defeatured and fully featured defeatured domains with region of interest [123] 29

2.8 Shape Sensitivity : Reference and Deformed Domains 31

2.9 Feature Sensitivity : Transformation [123] 34

3.1 Small features definition applied to positive features on the boundary 42

3.2 Small features definition applied to thin positive features 43

3.3 Small features definition applied to negative features 43

3.4 A typical car wheel assembly containing numerous small positive and negative features 44

3.5 SAQ - Simplified domain Ω_0 is not homeomorphic to Ω 47

3.6 SAQ - Simplified domain Ω_0 is homeomorphic to Ω 47

3.7 Flowchart for Shape Aware Quadratures 50

3.8 Reference and Deformed domains for SSA 53

3.9 Geometric corrections based on first-order SSA is exact for computation of area (or volume) provided Γ_0 is homeomorphic to Γ and all the vertices (or edges) of Γ_0 lie on Γ 54

3.10 Approximate polygon (Ω_0) construction by ray casting. The approximate polygon vertices are generated by casting rays parallel to integration rays and intersecting it with the original domain (Ω). 67

3.11 Example 1 70

3.12 Example 2 72

3.13 Cases in marching squares 74

3.14 Example 3 - Basis functions 76

3.15 Example 3 - Polygonal edge variation 77

3.16 Example 3 - Quadrature rule variation 78

3.17 Example 3 - Quadtree depth variation 79

3.18 A 2D non-convex domain with numerous holes 80

3.19 Non-Convex domain - Rel. Error Vs Number of Edges 81

3.20 Cases in marching cubes [2] 82

3.21 Relative error in computing volume of a unit sphere 83

3.22 Relative error in computing $\int x^2 d\Omega$ over a unit sphere 83

3.23 The L^2-norm of the residual error in integrating all trivariate polynomials of order up to three ($x^p y^q z^r$ with $p, q, r \geq 0\,, p + q + r \leq 3$) over a unit sphere . 83

3.24 The L^2-norm of the residual vector in integrating all trivariate polynomials of order up to three ($x^p y^q z^r$ with $p, q, r \geq 0\,, p + r + r \leq 3$) over a unit sphere for three different quadrature point generation schemes 85

3.25 Quadrature points generated using scaled Cartesian product rule 85

3.26 Quadrature points generated randomly inside the domain 86

3.27 Quadrature points generated randomly on the faces of the approximate polyhedron 86

3.28 Unit sphere with small spherical voids 88

3.29 A typical leaf cell of the unit sphere contains several hundred voids 88

3.30 Error plot for various hole distribution 89

4.1 The fictitious domain approach : the physical domain Ω is extended by the fictitious domain Ω_{fict} into an embedding domain Ω_e to allow easy meshing of complex geometries. The influence of Ω_{fict} is penalized by α [118] 93

4.2 Polynomial interpolation of a Heaviside function: (a) through 5 points; (b) through 21 points [22] 96

4.3 Illustration of boundary cell fragmentation in characteristic function method [118] 96

4.4 Approximate polygon (Ω_0^i) construction and quadrature point generation over a leaf cell (I_i) by ray casting. 99

4.5 Cases in marching squares 100

4.6 Quadrature point distribution for SAQ scheme over a 2D domain 100

4.7 Design velocity is simply the distance to the boundary in a direction normal to Γ_0 101

4.8 Cases in marching cubes [2] 104

4.9 Savings in quadrature points (per leaf cell) for the SAQ scheme in 2D and 3D 105

4.10 (a) The leaf cells of the octree are first classified into one of the 15 cases using PMC (b) Depending on the cell type, an approximate polyhedron homeomorphic to the domain in each of the leaf cells is constructed (c) Design velocity is computed at the surface quadrature points of the approximate polyhedron (Γ_0) (d) Volumetric quadrature points are generated such that it lies within the actual and polyhedral domain using ray casting (actual surface not shown for clarity) 106

4.11 2D annular ring problem from [118] 108

4.12 Radial displacement plot obtained using SAQ (with 3 subdivisions) 108

4.13 von Mises stress plot obtained using SAQ (with 3 subdivisions) 108

4.14 Convergence of SAQ and characteristic function method w.r.t. p-refinement in energy norm 109

4.15 Convergence of SAQ and characteristic function method w.r.t. quadtree depth in energy norm 110

4.16 Convergence of SAQ and characteristic function method w.r.t. quadrature rule in energy norm 112

4.17 Cylinder under uniaxial compression 114

4.18 Convergence w.r.t h-refinement for the cylinder problem 114

4.19 Convergence w.r.t octree refinement for the cylinder problem 115

4.20 von Mises stress plot (top face) for the uniaxial cylinder problem obtained from an overkill FEM solution (with 313,842 DOFs) in SOLIDWORKS [3] . 115

4.21 von Mises stress plot (top face) for the uniaxial cylinder problem obtained from (a) one (b) two and (c) three octree subdivisions using the characteristic integration scheme (4x4x4 mesh) in FCMLab [95] 115

4.22 von Mises stress plot (top face) for the uniaxial cylinder problem obtained from (a) one (b) two and (c) three octree subdivisions using the SAQ scheme (4x4x4 mesh) in FCMLab [95] 116

4.23 Displacement norm plot of a human femur model problem as obtained by employing SAQ in FCMLab with $E = 15.2 \times 10^3$ N/mm^2 and $\nu = 0.4$. An hip contact pressure of 1 MPa is applied on the non-boundary conforming discretization using a separate surface discretization of the hip cap. 117

5.1 Design velocity is simply the distance to the boundary in a direction normal to Γ_0 123

5.2 Quadrant of a Circle (Ω) and its approximate domain (Ω_0) 133

5.3 Error in the area computations of a quadrant (Ω) using zeroth and first-order Taylor series approximation various polygonal approximations (Ω_0) 136

5.4 Different polygonal approximations (Ω_0) used for computing SAQ over a quadrant domain (Ω). The vertices of the polygon (green dots) lie on the original boundary (Γ) 136

5.5 Convergence plot for integral of bivariate polynomials (of order up to 3) over the quadrant obtained using SAQ 137

5.6 A family of noisy domains obtained by setting various values of k in $y(x) = x^4 + e^x + 0.75\, sin(k\pi x) + 0.5$ for $x \in [0, 2]$ 138

5.7 Convergence plot for area computation over the noisy domain for various values of k 139

5.8 Illustration of violation of homeomorphic assumption at coarser polygonal approximations ($n_e = 7$) for the noisy domain with $k = 4$ 139

5.9 Convergence plot for integral of bivariate polynomials (of order up to 3) over the noisy domain with $k = 2$ 140

5.10 Convergence plot for integral of bivariate polynomials (of order up to 3) over the noisy domain with $k = 4$ 140

5.11 Convergence plot for integral of bivariate polynomials (of order up to 3) over the noisy domain with $k = 6$ 141

5.12 Convergence plot for integral of bivariate polynomials (of order up to 3) over the noisy domain with $k = 8$ 141

5.13 SAQ using a quadtree over different domains 144

5.14 Comparison of actual and predicted errors for the unit circle, noisy domain, and non-convex domain 144

5.15 *A posteriori* error plot for integral of $f(x, y)$ over the non-convex domain . . 145

5.16 *A posteriori* error plot for integral of $g(x, y)$ over the non-convex domain . . 145

A.1 Vertices (black dots) of Γ_0 lying on Γ makes $V_N = 0$ at the vertices 172

B.1 Reference and deformed domains for Topological-Shape Sensitivity Analysis [17] 175

List of Tables

2.1 Comparison of Different Integration Schemes 27

3.1 SSA of Moments 52
3.2 TSA of Moments (for bivariate/trivariate polynomial basis) 55
3.3 Relative errors, order, and computational time for integral of f over the wavy domain 79
3.4 Relative error and order of convergence in volume computations for a unit sphere 86
3.5 Relative error and order of convergence in second moment computations for a unit sphere 87

4.1 Rel. error in energy norm and time comparison for SAQ and Characteristic method w.r.t quadtreee refinement 110
4.2 Mesh sizes and corresponding DOFs for the uniaxial compression problem . . 116
4.3 Time required to generate the stiffness matrix in generating the converged points for various mesh sizes 117

I Integral of basis functions over the quadrant 184
II Integral of basis functions over the notched domain 185
III Integral of basis functions over the wavy domain 186

Abstract

With Shape Aware Quadratures (SAQ), for a given set of quadrature nodes, order and domain of integration, the quadrature weights are obtained by solving a system of suitable moment fitting equations in least square sense. The moments in the moment equations are approximated over a simplified domain (Ω_0) that is a reasonable approximation of the original domain (Ω) that are then corrected for the deviation of the shape of Ω_0 from Ω via shape correction factors. The shape correction factors can be derived based on a variety of sensitivity analysis techniques such as shape sensitivity and topological sensitivity. Using the right kind/order of shape correction factors for moment approximation enables the resulting quadrature (from the moment fitting equations) to efficiently adapt to the shape of the original domain even in the presence of numerous small features (such as holes, notches and gaps). We demonstrate the efficacy of the method in integrating bivariate/trivariate polyn omials over several 2D/3D domains in the presence of small features.

SAQ is formulated irrespective of how the original domain Ω is represented, and it can be used with or without an integration mesh. More specifically, SAQ is suitable for the computation of integrals arising in immersed boundary (IB) methods such as solution structure methods, fictitious domain methods, and Finite Cell Method (FCM). Hence, we study the efficacy of SAQ to achieve a given accuracy in the context of FCM (one of the IB methods) which must perform numerical integration over arbitrary domains without meshing. The standard integration technique employed in FCM is the characteristic function method that converts the continuous integrand over a complex domain into a discontinuous integrand over a simple (box) domain. Then, well known integrand adaptivity techniques are employed to integrate the resulting discontinuous integrand over the box domain. Although this method is simple to implement, it becomes computationally very expensive for realistic complex 3D domains such as sculptures, bones and engines.

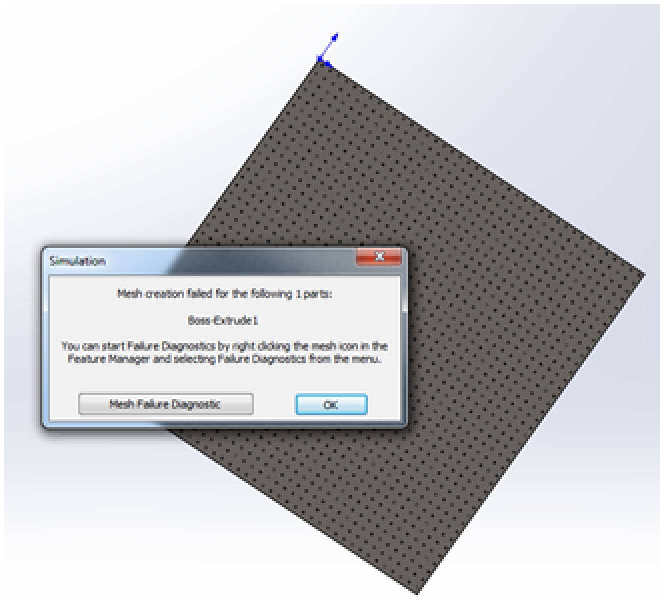
In contrast, the shape aware quadratures automatically adapt to the shape of the original domain, due to the incorporation of suitable shape correction factors, without the need to making the integrand discontinuous leading to superior computational properties. We demonstrate the computational efficiency of SAQ over characteristic function method (in the context of FCM) as it requires fewer subdivisions and less time to achieve a given accuracy in both two and three dimensions. In addition, SAQ offers a number of advantages including flexibility in the choice of quadrature points and basis functions that is usually not available for more traditional approaches.

However, moment approximations in SAQ leads to error in the computed integral. Thus, there is need to bound/estimate this error. Hence, we present an *a priori* and *a posteriori* error analysis for SAQ (with SSA correction factor) based on Taylor's remainder theorem and higher-order Taylor series respectively. Specifically, we prove that SAQ (with SSA correction factor) exhibits quadratic convergence w.r.t. (integration) mesh refinement under suitable assumptions. We also propose a simple *a posteriori* error estimate that can be used for adaptive integration of complex geometric domains. We further validate the error estimates using several computational examples in two dimensions.

Chapter 1

Introduction

1.1 Motivation



Mesh creation failed for the following 1 parts:

Boss-Extrude1

You can start Failure Diagnostics by right clicking the mesh icon in the Feature Manager and selecting Failure Diagnostics from the menu.

Figure 1.1: Meshing failure for a perforated plate with 1,521 holes of size 1 mm in Solidworks

Mesh generation is a major problem in simulation based Engineering, involving considerable computational resources, especially for complex geometries. Approximately 80% of the total time invested for a FEA is expended in geometric modeling and mesh generation [118]. In addition, mesh generation suffers from several other drawbacks such as lack of robustness, lack of guarantees, and inability to robustly handle small geometric/topological details. Specifically, the presence of small geometric/topological features such as notches, gaps and holes makes the meshing process difficult or even impossible. For example, mesh generation failed for the perforated plate example shown in Fig. 1.1 in one of the most popular CAD softwares owing to the presence of numerous small holes.

These challenges in mesh generation motivated the research in several meshless approaches such as the Generalized Finite Element Method (GFEM) [56] and Extended Finite Element Method (XFEM) [40]. A family of such meshfree methods, known as Immersed Boundary (IB) methods [33, 105], have gained popularity in the recent past owing to its simplicity and wide applicability. Some examples of the IB methods include fictitious domain methods [45, 8, 113], embedded domain methods [6, 7], solution structure methods [58, 85, 86, 72, 73], certain versions of XFEM [26, 51, 77, 91, 117, 124, 126], Finite Cell Method (FCM) [66, 13], and fixed-grid methods [4, 5, 9, 10, 11]. In these methods, the original domain is immersed or embedded into a geometrically simpler domain (usually box-like domain). This simpler domain is then discretized (trivially) to allocate basis functions. These basis functions are then used to discretize the variational (or weak) form of the boundary value problem. This results in a linear system which when solved produces an approximation to the unknown field (such as displacements). The formulation of the linear system requires computation of volumetric and surface integrals. The surface integration is carried out by means of local surface mesh generation. The volumetric integration is carried out using some kind of adaptive integration procedure such as quadtree/octree [48, 49].

Thus, the use of a structured voxel grid eliminates most of the bottlenecks of conforming-mesh based methods. However, the use of a voxel grid introduces other problems. To understand this, let us consider the Finite Cell Method (FCM) [66, 13], a kind of IB method, that has become very popular in recent times. The FCM was originally introduced in [66] as an extension to the p-version of the finite element method [24, 54]. FCM combines the fictitious domain approach with a higher-order approximation basis, the representation of the geometry by adaptive quadrature based on recursive bisection, and the weak imposition of unfitted boundary conditions [118]. FCM can operate on any geometry as along as the geometry supports point membership classification query i.e. whether a point is located inside or outside the physical domain. Fig. 1.2 illustrates the fictitious domain concept that lies at the heart of the FCM.

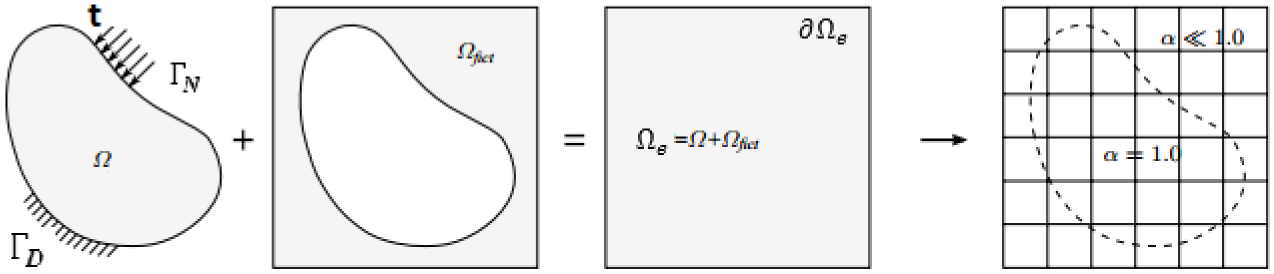


Figure 1.2: The fictitious domain approach : the physical domain Ω is extended by the fictitious domain Ω_{fict} into an embedding domain Ω to allow easy meshing of complex geometries. The influence of Ω_{fict} is penalized by α [118]

The embedding domain Ω_e consists of the physical domain of interest Ω and the fictitious domain extension Ω_{fict}. Analogous to classical FEM, the first step in FCM is to derive the variational (or weak) form for the boundary value problem under consideration. Let us consider the standard Poisson problem for the purpose of illustration. The FCM variational form for the Poisson problem reads as follows

find $u \in H^1(\Omega)$ *such that*

$$\int_\Omega \alpha \nabla \delta u . \nabla u d\Omega = \int_\Omega \alpha f \delta u d\Omega + \int_{\Gamma_N} g_n \delta u d\Gamma_N \ \ \forall \delta u \in H^1(\Omega) \tag{1.1}$$

$$u = u_0 \ \forall x \in \Gamma_D \tag{1.2}$$

where $g_n = \nabla u.\mathbf{n}$ is the Neumann boundary condition on Γ_N. Neumann boundary conditions are also specified over the boundary of the embedding domain $\partial\Omega_e$, where $\nabla u.\mathbf{n} = 0$ by definition. Dirichlet boundary conditions are specified over Γ_D of the physical domain by the prescribed value of the field variable u_0. The scalar factor α is defined as follows

$$\alpha(x) = \begin{cases} 1.0 \quad \forall x \in \Omega \\ \\ 10^{-q} \; \forall x \in \Omega_{fict} \end{cases}$$

In Ω_{fict}, α must be chosen as small as possible, but large enough to prevent extreme ill-conditioning of the stiffness matrix [66, 13]. Typical values of α range between 10^{-5} and 10^{-10}. This scalar parameter α ensures that we are actually solving the boundary value problem on Ω and not on the embedded domain Ω_e. Using a structured grid of higher-order elements (as shown in Fig. 1.2), the field variable u and its variation δu are discretized as follows

$$u = \sum_{i=1}^{n} N_i u_i \tag{1.3}$$

$$\delta u = \sum_{i=1}^{n} N_i \delta u_i \tag{1.4}$$

Substituting the above in Eq.(1.1) results in the following discretized weak form

$$(\mathbf{K})\{\mathbf{u}\} = \{\mathbf{f}\} \tag{1.5}$$

with

$$K_{ij} = \int_{\Omega} \alpha \nabla N_i \nabla N_j d\Omega \tag{1.6}$$

$$f_i = \int_{\Omega} \alpha f N_i d\Omega + \int_{\Gamma_N} g_n N_i d\Gamma_N \tag{1.7}$$

where $\mathbf{K}$ is the stiffness matrix, $\mathbf{u}$ the unknown field coefficients, and $\mathbf{f}$ the load vector. Solving the above linear system for $\mathbf{u}$ and substituting the resulting coefficients in Eq.(1.3) results in the estimation of the field solution u. Notice that the formulation of stiffness matrix and the load vector requires volumetric integration of discontinuous function over a box-like domain. In certain other IB methods such as Scan and Solve [85, 86], it is required to perform volumetric integration of continuous functions over arbitrary domain. Thus, unlike FEM, volumetric integration over a non-conforming background grid is much more complicated. Thus, FCM (and most IB methods) replaces the problem of mesh generation with that of numerical integration of discontinuous (or continuous) functions over simple (or arbitrary) domains. Also, imposition of Dirichlet boundary conditions is not straightforward and requires special consideration. These problems aggravate in the presence of small features such as fillets, notches, small gaps, and small holes that is found in any typical CAD model. Accurately capturing the physics near these small features is of utmost importance as they are usually the regions under high stress and hence are considered critical from the designer's standpoint. Moreover, recent advances in additive manufacturing and the ability to optimally generate microstructures have furthered interest in efficient (speed and accuracy) resolution of these small features. The first step in efficient resolution of these small features is the efficient numerical integration of stiffness/force integrands (Eq.(1.6) and Eq.(1.7)). Hence, one of the main objectives of this thesis is to develop an efficient integration scheme that can be employed in IB methods. However, in this thesis, we will develop a new quadrature scheme that is quite general and can be used even without a conforming/non-conforming integration mesh.

1.2 Challenges in Numerical Integration

The accuracy of the solution field depends very much on the accuracy of stiffness and force coefficients in Eq.(1.5). As already mentioned, the stiffness and force coefficients are integrals over the geometric domain Ω. Specifically, computation of $\mathbf{K}$ requires volumetric integration. However, setting up $\mathbf{f}$ usually requires surface and/or volumetric integration. Surface integration can be easily accomplished by local surface mesh generation. However, accurate and speedy volumetric integration over arbitrary domains is non-trivial. Volumetric Integration over complex domains, such as those arising in most engineering applications, usually require that domain Ω be approximated by a union of simple cells, so that the integration can be carried out in a cell by cell fashion. In order to avoid integration errors, the union of cells must closely approximate the domain either by cells that conform to the boundary $\partial\Omega$ or by non-conforming cells that are recursively decomposed to cover the original domain. The first approach leads to a difficult (theoretically and computationally) problem of meshing. The latter case, illustrated in Fig. 1.3, leads to excessive fragmentation near the boundary, resulting in significant computational cost and/or loss of accuracy.

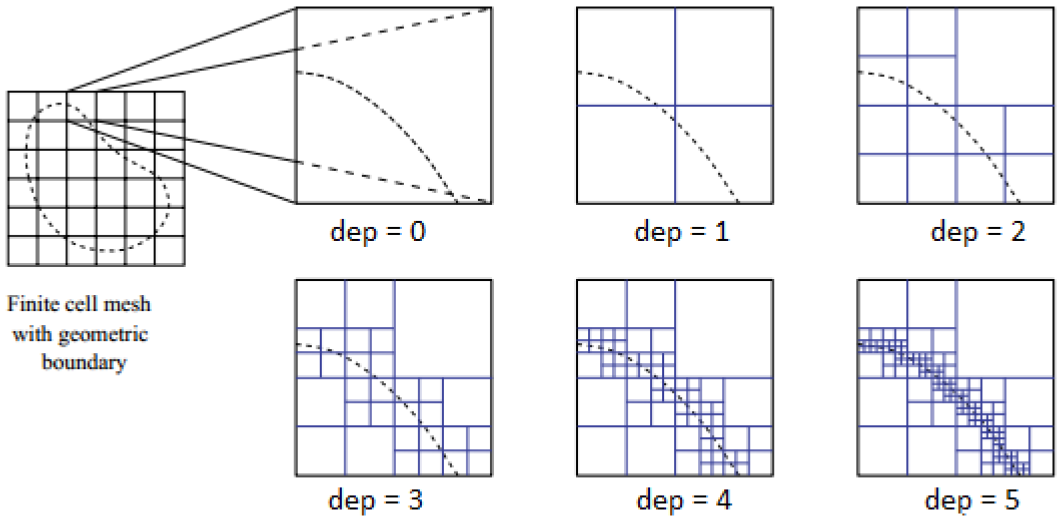


Figure 1.3: Illustration of excessive boundary cell fragmentation in quadtree based integration [118]

This becomes even more pronounced in the presence of numerous small features as in the perforated problem of Fig. 1.1. The presence of small features such as a voids or inclusions smaller than an integration cell (in both 2D and 3D) poses a serious issue. Fig. 1.4 illustrates this situation in which the void is completely situated inside the cell. Since all vertices of a cell are inside the geometric domain, this void is missed completely by conventional integration techniques. There are several ways to detect such small features. If the user knows that small features exist in the domain geometry, an initial grid with tighter spacing (Fig. 1.5(a)) can be provided to ensure a cell corner intersects the feature. This is a simple way to globally ensure that features down to a given size are detected and accounted for. However, because the increased grid resolution applies everywhere in the domain, including regions where it is not needed, it can unnecessarily drive up the computational cost. This approach can be optimized if an additional information about location of small geometric features is available. In this case, as it is illustrated by Fig. 1.5 (b) and Fig. 1.5 (c), a denser grid is imposed within a cell or a non-uniform adaptive subdivision can be done depending on the location of small features.

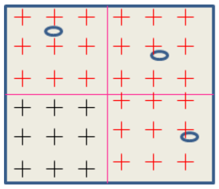


Figure 1.4: Features smaller than the integration grid resolution are missed

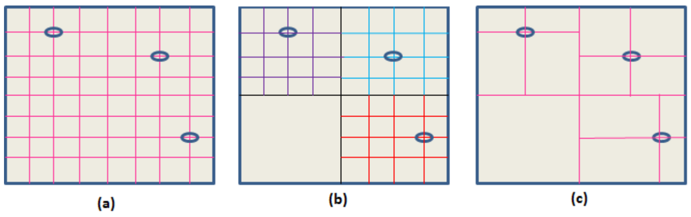


Figure 1.5: Capturing the effect of small features by (a) uniform subdivision (b) overlaying of fine grid and (c) non-uniform adaptive subdivision

However, these techniques become prohibitively expensive for a model with large number of small features such as the perforated plate problem of Fig. 1.1. This explains the need for accurate volumetric integration scheme with minimal subdivision to accelerate the whole solution process. This requires that the integration scheme adapt to the geometry and topology (i.e. shape) of the domain in addition to the integrand. Issues related to integrand adaptivity is well understood. On the other hand, efficient shape adaptivity is far from complete and will be the main focus of this thesis.

1.3 Problem Statement

In this thesis, we will address efficient numerical integration of any integrable function over arbitrary domains in the presence of small features.

Formally, the problem is to numerically evaluate the integral

$$\int_{\Omega} f(\mathbf{x}) d\Omega \tag{1.8}$$

where $f : \Omega \to \mathbb{R}$ is an arbitrary integrable function defined over an arbitrary[a] domain $\Omega \subset \mathbb{R}^d$ ($d = 2$ or 3) that is typically represented by a solid model [108].

[a]By arbitrary domain we mean any 2D or 3D closed regular set [108] whose boundary is an orientable manifold

We further assume that f can be suitably approximated by a function $\hat{f} \in span(\{b_i\}_{i=1}^m)$ such that the basis functions $b_i : \mathbb{R}^d \to \mathbb{R}$ are symbolically integrable functions that are sufficiently smooth. In other words, $\hat{f}$ can be written as a linear combination of sufficiently smooth basis functions b_i i.e. $\hat{f}(\mathbf{x}) = \Sigma_{i=1}^m c_i b_i(\mathbf{x})$. The coefficients c_i need not necessarily be known ahead of time.

1.4 Overview

The main contributions of this thesis are :

1. A new quadrature scheme, called the Shape Aware Quadratures (SAQ), that can be used to efficiently integrate *a priori* unknown functions over arbitrary domains in the presence of small features based on moment fitting equations [137, 121], divergence theorem [30, 41] and sensitivity analysis [75, 76, 67, 17, 123, 80]

2. Convergence and performance studies of SAQ in the context of Finite Cell Method [66, 13]

3. *A priori* and *a posteriori* error estimation of SAQ

In Shape Aware Quadrature (SAQ), given an arbitrary domain Ω with a set of appropriately chosen quadrature nodes and order of integration, we compute the quadrature weights by solving a system of linear moment fitting equations [64, 116, 137] for an appropriate set of basis functions in the least square sense. Setting up the moment-fitting equations involves computing integrals of the basis functions (known as moments) over an arbitrary geometric domain. This task itself is non-trivial, because it either entails some kind of domain decomposition, or can be reduced to repeated boundary integration as was proposed in [16] for bivariate domains. Adaption of the latter approach avoids excessive domain fragmentation, but leads to a significant computational overhead, particularly in 3D. We overcome this challenge by first computing the moments over a simpler domain Ω_0, usually a polygon/polyhedron, that is a reasonable approximation of the original domain Ω. The computed moments are then corrected for the deviation of the shape of Ω_0 from Ω by means of shape correction factors. The shape correction factors can be computed via various sensitivity analysis techniques such as the shape sensitivity [75, 76], topological sensitivity [67, 17], feature sensitivity [123] and modification sensitivity [80]. The moments over approximate domain are further reduced to boundary integrals by the application of divergence theorem [136, 36, 30, 23, 41]. The approximate moment fitting equations, thus obtained, can then be easily solved for quadrature weights. In other words, the shape correction factor ensures that the quadrature rule determined by the moment fitting equations is "aware" of the shape of integration domain – hence the name *Shape Aware Quadratures (SAQ)*. The resulting quadrature weights are not exact, but are accurate enough to integrate functions over any arbitrary domain when they are well approximated by the chosen set of basis functions. In addition, small features such as holes, notches, and thin features can be accounted for automatically by employing the correct order/type of shape correction factor in the moment approximations.

The main application of SAQ is in the boundary cells of any quadtree/octree based integration. Thus, SAQ could be advantageously employed in integrating system matrices/vectors arising in any IB (or meshless) method. Hence, in this thesis, we demonstrate the efficacy of this method in integrating stiffness/force integrands arising in elastostatic problems using the Finite Cell Method (FCM). In addition, SAQ is found to offer a great deal of flexibility in the choice of quadrature points and basis functions that is not usually available for other integration schemes such as the characteristic function method [134] employed in IB methods.

The use of sensitivities in computing the shape correction factor results in an approximation of the moments. Thus, there is a need to estimate the resulting error in the computed integral using SAQ. Hence, we develop an *a priori* error estimate for SAQ (with SSA shape correction factor) based on Taylor's Remainder Theorem. In addition, we propose a simple *a posteriori* error estimator based on higher-order Taylor series approximation that can be used in the context of efficient adaptive integration over arbitrary 2D/3D domains.

Outline A detailed survey of volumetric integration and sensitivity analysis is presented in chapter 2. A new integration procedure called as the *Shape Aware Quadrature* (SAQ) is developed in chapter 3 as a means to efficiently integrate *a priori* unknown functions over arbitrary domains in the presence of small features. Chapter 4 studies the convergence/performance of SAQ relative to the traditional characteristic function approach [12] in the context of FCM [66, 13]. *A priori* and *a posteriori* error analysis of SAQ is the topic of chapter 5. Finally, chapter 6 summarizes the contribution of this thesis along with open issues that warrant further research.

Chapter 2

Background

In this chapter we will review relevant literature on volumetric integration and sensitivity analysis.

2.1 Volumetric Integration

As already pointed out in the introduction, formulation of stiffness matrix (Eq.(1.6)) and the load vector (Eq.(1.7)) requires volumetric integration of continuous/discontinuous functions over arbitrary domains. In certain IB methods such as the FCM [66, 13], embedded domain methods [6, 7], and fictitious domain methods [45, 8, 113], it is required to integrate discontinuous functions over a simple (box-like) domain. In other IB methods such as the solution structure method [58, 85, 86, 72, 73], it is required to integrate continuous functions over arbitrary domains. In either case, the integration procedure is complicated and results in fragmentation of the integration boundary. Thus, there is a necessity for the integration procedure to adapt to the geometry/topology (shape) of the domain in addition to adapting to the integrand. The issues related to integrand adaptivity is well understood (e.g., see [132]). On the other hand, shape adaptivity is far from complete. Hence, in this section, we will survey relevant ideas and literature on numerical integration techniques with emphasis on geometric adaptivity. In this section, we will closely follow our exposition in [129].

2.1.1 Polygonal/Polyhedral Domains

Simple Shapes

Numerical integration over a regular n-dimensional interval, i.e. a box, is well understood (see for e.g. [35]). Stroud [20] describes cubature rules for a number of different simple domains (such as hexagon, octahedron, and cubical shell) in his book four decades ago. Since then, many additional specialized cubature formulas have been developed for a variety of simple shapes and spaces [104, 103, 1].

Symbolically integrable functions - Divergence Theorem

Several authors [136, 30, 36, 23, 41] have advocated the use of divergence theorem to integrate symbolically integrable functions over polygonal/polyhedral domains. Here, application of divergence theorem (once for polygonal domain and twice for polyhedral domain) converts the 2D/3D integral into an 1D integral over the edges of the polygon/polyhedron. The resulting integral is then efficiently evaluated using 1D Gauss Quadrature rules. For the purpose of illustration let us consider the integration of a symbolically integrable function $b_i(\mathbf{X})$ over the polygonal domain $\Omega_0 \subset \mathbb{R}^2$. Our objective is to simplify $\int_{\Omega_0} b_i(\mathbf{X}) d\Omega_0$ to an integral over the polygonal boundary. In order to apply the divergence theorem, we first require a vector function $\boldsymbol{\phi}^i = \phi^i_X(X,Y)\hat{\imath} + \phi^i_Y(X,Y)\hat{\jmath}$ such that the following is satisfied

$$\int\int_{\Omega_0} b_i(X,Y) dX dY = \int\int_{\Omega_0} \nabla \cdot (\boldsymbol{\phi}^i) dX dY \tag{2.1}$$

It can be easily proved [41] that for $\boldsymbol{\phi}^i(X,Y) = \int b_i(X,Y) dX \hat{\imath} + 0\hat{\jmath}$ the above is satisfied. Now, application of divergence theorem to the RHS of Eq.(2.1) yields

$$\int\int_{\Omega_0} b_i(X,Y) dX dY = \int_{\Gamma_0} \phi^i_X N_X d\Gamma_0 \tag{2.2}$$

where $N_X = \mathbf{N}.\hat{\mathbf{i}}$ is the X-component of the normal to the boundary. For a polygon, the normal is constant over its edges. Hence, for a polygon with n_e-edges (i.e. $\Gamma_0 = \cup_{k=1}^{n_e} E_0^k$) one could rewrite the above more explicitly as:

$$\int\int_{\Omega_0} b_i(X,Y)dXdY = \sum_{k=1}^{n_e} \int_{E_0^k} \phi_X^i(X,Y) N_X^k dE_0^k \tag{2.3}$$

It is to be noted that $\phi_X^i(X,Y) = \int b_i(X,Y)dX$ can be obtained symbolically (for e.g., if $b_i(X,Y) = X^p Y^q$, then $\phi_X^i(X,Y) = \frac{X^{(p+1)}Y^q}{(p+1)}$).

Analogously, the integral over the polyhedral domain $\Omega_0 \subset \mathbb{R}^3$ can be derived to be an integral over its faces as [41]

$$\int\int\int_{\Omega_0} b_i(X,Y,Z)dXdYdZ = \sum_{k=1}^{n_f} \int_{F_0^k} \chi_X^i(X,Y,Z) N_X^k dF_0^k \tag{2.4}$$

where $\chi_X^i(X,Y,Z) = \int b_i(X,Y,Z)dX$ and F_0^k being the k^{th} polygonal face of the polyhedron with n_f faces. Thus, in general if $\Omega_0 \subset \mathbb{R}^d$ $(d = 2\,or\,3)$ is the polygonal/polyhedral domain of integration with boundary $\Gamma_0 \subset \mathbb{R}^{d-1}$ comprising of n_* edges/faces (i.e. $\Gamma_0 = \cup_{k=1}^{n_*} \Gamma_0^k$), then

$$\boxed{\int_{\Omega_0} b_i(\mathbf{X})d\Omega_0 = \int_{\Gamma_0} \beta_X^i(\mathbf{X}) N_X d\Gamma_0 = \sum_{k=1}^{n_*} \int_{\Gamma_0^k} \beta_X^i(\mathbf{X}) N_X^k d\Gamma_0^k \qquad (2.5)}$$

where $\beta_X^i(\mathbf{X}) = \phi_X^i(X,Y)$ for polygons and $\beta_X^i(\mathbf{X}) = \chi_X^i(X,Y,Z)$ for polyhedrons. In the case of polyhedra, one more application of divergence theorem to Eq.(2.5) would reduce the original integral into a 1D integral similar to that of Eq.(2.3).

Arbitrary integrable functions - Moment Fitting

For arbitrary integrable functions, Sommariva and Vianello [14] used Green's integral formula with thin-plate splines basis functions to obtain a meshless cubature method for convex,non-convex and even multiply connected polygonal domains. In general, a cubature or quadrature in $\mathbb{R}^d$ is a formula of the form

$$\sum_{i=1}^{n} w_i f(\mathbf{x}_i) \approx \int\limits_{\Omega} W(\mathbf{x}) f(\mathbf{x}) d\Omega \tag{2.6}$$

where $\Omega \subset \mathbb{R}^d$ is the integration region, f is an integrable function defined on Ω, and $W(\mathbf{x})$ is a non-negative weight function. A quadrature rule is determined by points $\mathbf{x}_i \in \mathbb{R}^d$ that are usually called quadrature nodes, and the quadrature weights w_i [137]. A standard technique for constructing quadrature rules is to solve the following system of moment fitting equations [64, 116, 137]

$$\begin{bmatrix} \int_\Omega W(\mathbf{x}) b_1(\mathbf{x}) \mathrm{d}\Omega \\ \int_\Omega W(\mathbf{x}) b_2(\mathbf{x}) \mathrm{d}\Omega \\ \cdots \\ \int_\Omega W(\mathbf{x}) b_m(\mathbf{x}) \mathrm{d}\Omega \end{bmatrix} = \begin{bmatrix} b_1(\mathbf{x_1}) & b_1(\mathbf{x_2}) & \cdots & b_1(\mathbf{x_n}) \\ b_2(\mathbf{x_1}) & b_2(\mathbf{x_2}) & \cdots & b_2(\mathbf{x_n}) \\ \cdots & \cdots & \cdots & \cdots \\ b_m(\mathbf{x_1}) & b_m(\mathbf{x_2}) & \cdots & b_m(\mathbf{x_n}) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \cdots \\ w_n \end{bmatrix} \tag{2.7}$$

The same equations may be written in a more compact form as

$$\{\mathbf{M}\}_{m\times 1} = [\mathbf{A}]_{m\times n} \{\mathbf{w}\}_{n\times 1} \tag{2.8}$$

with

$$\{\mathbf{M}\} = \begin{bmatrix} \int_\Omega W(\mathbf{x}) b_1(\mathbf{x}) \mathrm{d}\Omega \\ \int_\Omega W(\mathbf{x}) b_2(\mathbf{x}) \mathrm{d}\Omega \\ \cdots \\ \int_\Omega W(\mathbf{x}) b_m(\mathbf{x}) \mathrm{d}\Omega \end{bmatrix} \tag{2.9}$$

where $\{b_i\}_{i=1}^m$ is the set of basis functions[1] and $\{\mathbf{M}\}$ is the vector of moments defined over the domain of integration Ω.

The quadrature nodes and the corresponding weights $\{\mathbf{x}_i, w_i\}_{i=1}^n$ can be determined nu-

[1]Popular choices of basis functions include the bivariate polynomials $x^p y^r$ $(p + r \leq o)$ and trivariate polynomials $x^p y^r z^s$ $(p + r + s \leq o)$ in 2D and 3D respectively (for the given integration order o). Other recommended choices of basis functions include Legendre and Chebyshev polynomials [15].

merically by solving the above set of moment fitting equations. The resulting quadrature rule can be used to integrate any function that is in the function space spanned by these basis functions, assuming that the integral of basis functions and solution to moment fitting equations were computed exactly. Each integration point in d-dimensions contribute $d+1$ unknowns: d coordinate components and one weight. Thus, $\lceil \frac{m}{d+1} \rceil$ could serve as an estimate for the number of points required to integrate m basis functions in d dimensions [121]. Further, Xiao and Gimbutas [137] devised a numerical algorithm for constructing efficient, higher-order quadratures in two and higher dimensions by using moment fitting equations along with node elimination scheme. This algorithm was successfully applied to integrate general functions over triangle and square [137].

Mousavi et al. [92, 120] used the same algorithm to construct efficient quadrature rules for bivariate polynomials to integrate functions over convex and non-convex polygons. Specifically, Mousavi and Sukumar [121] constructed efficient quadrature for integration over convex polygons and polyhedra based on moment fitting equations and monomial basis functions. They used Lasserre's method [68, 69] for the integration of homogeneous functions arising in the moment computations. The authors also observed that integration of the known basis functions may be transformed to boundary integrals using divergence theorem, at least in principle. Recently, Joulaian et al. [71] applied this idea to the leaf cells of octree and demonstrated its application in the context of FCM. However, in practice, repeated and accurate integration over the boundary of an arbitrary domain is computationally expensive.

2.1.2 Adaptation by Hierarchical Partitioning

The boundary of an arbitrary domain is a piecewise smooth curve in 2D or surface in 3D. Numerical integration over such domains must adapt to the domain's boundary. This can be achieved by meshing – subdividing the domain into a union of cells that conform to the domain's boundary and then applying well established quadrature rules to each of the cells. Recall that our goal is to avoid mesh generation. However, in principle, the boundary may be polygonized, but none of the proposed methods scale to handle polygons or polyhedra of such complexity.

A common approach to geometric adaptation is hierarchical partitioning. In its simplest form, a quadtree [70] or octree [96] subdivision breaks the domain into non-conforming cells recursively. This kind of subdivision primarily leads to two types of integration cells: interior cells and boundary cells (see Fig. 2.1a). Interior cells can be easily handled using ordinary lattice rules (also called the Cartesian product rule). Several approaches to dealing with boundary cells have been proposed. For example:

(a) The simplest method is to randomly include/exclude boundary cells (see Fig. 2.1b) and then use ordinary lattice rules on the randomly included cells [109].

(b) Alternatively, one could map the boundary cells to smoothly deformed rectangles [72] or other canonical regions [78] and then use ordinary lattice rules on it.

(c) More recently, convenient space parametrization of boundary cells was found to produce good results [131, 58, 22].

(d) Monte-Carlo based methods [132] can be used to accurately integrate the boundary cells.

(e) Finally, level-set based approaches [31, 100] to deal with boundary cells.

All of the proposed approaches to dealing with boundary cells are problematic. Specifically, referring to the above, (a) lacks accuracy, (b) doesn't generalize to 3D, (b) and (d) are computationally expensive, (b) requires function evaluation outside of the domain, (e) is specific to a particular representation of the domain and (c) involves heuristic steps.

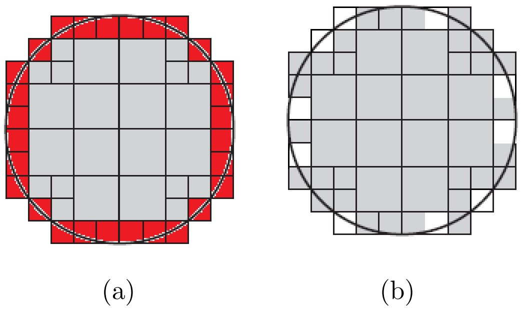


Figure 2.1: Hierarchical partitioning (quadtree) (a) Grey: interior cells; Red: boundary cells (b) Boundary cells are randomly excluded from integration

2.1.3 Adaptation by Characteristic Function

The problem of geometric adaptivity can be reduced in to integrand adaptivity over regular rectangular/box domain. Thus, well known integrand adaptivity techniques (over regular domains) can be used to compute the integral over arbitrary domains. To be precise, let f be the function to be integrated over the arbitrary domain $\Omega \subset D$, where D is a regular domain. Let us define a characteristic (Heaviside) function $H(\mathbf{x})$ as

$$H(\mathbf{x}) = \begin{cases} 0 & : \mathbf{x} \notin \Omega \\ 1 & : \mathbf{x} \in \Omega \end{cases} \qquad \text{so that} \quad \int_{\Omega} f(\mathbf{x}) d\Omega = \int_{D} f(\mathbf{x}) H(\mathbf{x}) dD. \tag{2.10}$$

Now we have transformed the original integration domain Ω of the integral (Fig. 2.2a) into a regular box domain D (Fig. 2.2b).

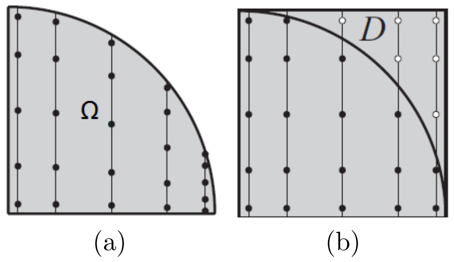


Figure 2.2: Integrand adaptivity (a) Original integration domain (b) Extended integration domain

Thus, the required integral can be computed by sampling the modified function $f(\mathbf{x})H(\mathbf{x})$ over the extended domain D using the ordinary lattice rules. Although this transformation is mathematically perfectly legitimate and sensible, it poses a serious problem when evaluated numerically owing to the discontinuity of the Heaviside function ($H(\mathbf{x})$). This method is particularly popular in most IB methods [105, 66]. A recent survey [12] discusses a number of strategies employed in the Finite Cell Method [66] in dealing with the discontinuity of the Heaviside function. These include

Local Meshing

The simplest way to deal with the discontinuity of the Heaviside function is to locally triangulate or mesh around the region of discontinuity. This method is often used in meshfree methods [46, 47] and in methods that deal with the simulation of cracks [47, 134] and other discontinuous phenomenon. Here, the triangulation conforms to the discontinuous interface and thereby captures the interface/boundary. However, the cost of applying this idea in IB method is high since cells are normally much bigger than standard finite elements, and capturing the boundary by local meshing can be expensive. Furthermore, this defeats the very purpose of avoiding conforming mesh generation.

Heuristic modification of integration weights and nodes

Rabczuk et al. [127] developed an heuristic way to modify the integration weights under the assumption that the discontinuous interface is linear. Consider a cell with line of discontinuity as shown in Fig. 2.3.

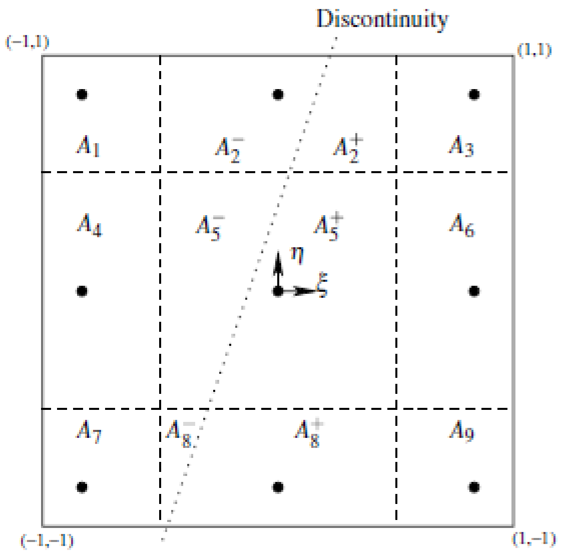


Figure 2.3: The supporting area is partitioned according to its contribution to the material or void [12]

Further, consider the supporting area of the Gauss Point (ψ_i, η_i), whose quadrature weight is w_i, which is cut by the line of discontinuity. Thus, the supporting area of w_i is split into two different polygons, with areas A^- and A^+ as shown in Fig. 2.3. It can be proved that the classical Gauss quadrature weight for a box-like region obeys $w_i = A_i^- + A_i^+$. Thus, Rabczuk et al. [127] heuristically replace w_i with either w_i^- (for void side) or w_i^+ (for material side) depending on where the quadrature point is located (void or material)

$$w_i^- = w_i \frac{A_i^-}{A_i} \tag{2.11}$$

$$w_i^+ = w_i \frac{A_i^+}{A_i} \tag{2.12}$$

The obvious disadvantage of this method is that it can't be applied for curved discontinuities. Further, the method is applicable only to holes and boundaries but not for multi-material problems [12].

Abedian et al. [12] proposed a modification to the above algorithm by replacing the integration point (cut by discontinuity) by two different points each at the centroid of its corresponding polygon (Fig. 2.4). Thus the integral (in this case) over the reference element is written as

$$\int_{+1}^{-1}\int_{+1}^{-1} f(\psi,\eta)d\psi d\eta = \sum_{i=1,3,4,6,7,9} w_i f_i + \sum_{j=2,5,8} [w_j^- f(\psi_j^-, \eta_j^-) + w_j^+ f(\psi_j^+, \eta_j^+)] \qquad (2.13)$$

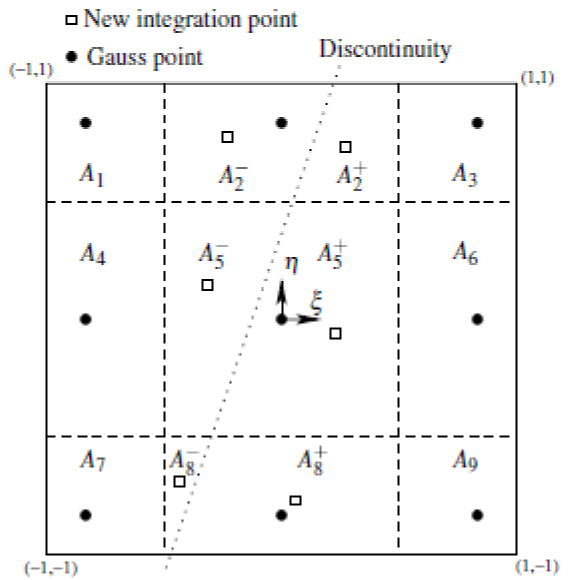


Figure 2.4: Each Gauss point is replaced by two new points at the centroids of the partitioned polygons [12]

The authors [12] claim that the method is applicable even for curved discontinuity. However, it is not clear how this can be applied in practice to arbitrary 3D curved domains.

Smoothed step function approach

Some authors [115, 21] smoothen the Heaviside function of Eq.(2.10) and then use octree/quadtree based subdivisions along with regular Gauss quadrature to integrate functions over arbitrary domains. There are several possible choices for smoothening the Heaviside function. One simple choice is the linear approximation of the Heaviside step function as given below

$$H(\phi(\mathbf{x})) = \begin{cases} 1 & : \phi \geq \epsilon \\ \frac{1}{2} + \frac{\phi}{2\epsilon} & : -\epsilon \leq \phi \leq \epsilon \\ 0 & : \phi \leq -\epsilon \end{cases} \tag{2.14}$$

where ϕ is an implicit function such that $\phi \leq 1$, $\phi = 0$, and $\phi \geq 1$ corresponds respectively to the interior, boundary, and exterior of the domain of integration. The approximate step function tends to the exact Heaviside function as $\epsilon \rightarrow 0$ but has a value equal to half at $\phi = 0$. Thus, the function $H(\phi)$ is an implicit function of the solid that has a value equal to half at the boundary of the domain where $\phi = 0$. As $\epsilon \rightarrow 0$, this function has a value equal to 1 inside the domain and 0 outside. There are several other possible choices of approximate Heaviside functions. For a list of such approximations refer to [21].

Equivalent polynomial Approach

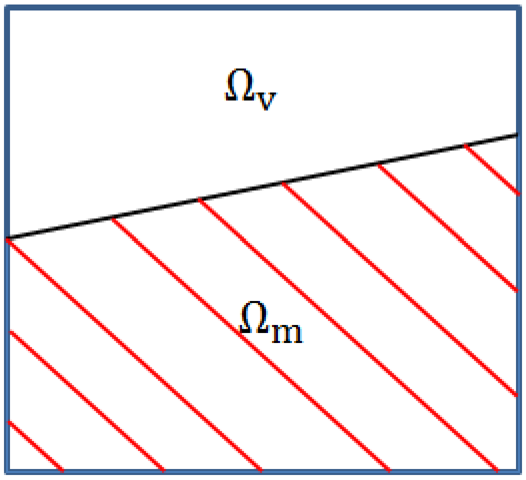


Figure 2.5: Integration cell with linear discontinuous interface

When the discontinuous interface is linear, Ventura [43] showed that an equivalent polynomial exists whose integral over the continuous part of the domain gives the exact value of the discontinuous/non-differentiable function integrated over the whole domain [12]. Thus, this avoids the subdivision of boundary cells in to sub-cells. Consider the case of 2D quadrilateral cells whose domain of integration (Ω) can be decomposed into material part (Ω_m) and void

(Ω_v), when it is cut by the discontinuity. For simplicity, let us consider the evaluation of the integrand f that is discontinuous over a linear interface as shown in Fig. 2.5 i.e. $\int_{\Omega_m \cup \Omega_v} f d\Omega$. Here, Ω_m and Ω_v refers respectively to the material and void regions of the domain. However, we know that in IB methods such as the FCM $\int_{\Omega_v} f d\Omega = 0$. Thus it is only required to evaluate the integral f, a polynomial of some degree, over Ω_m i.e. $\int_{\Omega} f d\Omega = \int_{\Omega_m} f d\Omega$. The method is based on finding a polynomial function g such that the following equation holds

$$\int_{\Omega_m} f d\Omega = \int_{\Omega} g f d\Omega \tag{2.15}$$

where $\Omega = \Omega_m \cup \Omega_v$ is the full (rectangular) cell. Thus if g exists, then an integration over a full cell regardless of the discontinuity can be performed to reach exactly the same result as when the discontinuity is considered. If we assume that g and f to be polynomials of the same order i.e.

$$f = b_0 + b_1 x + b_2 y + ... + b_{j-1} x^i + b_j y^i \tag{2.16}$$

$$g = c_0 + c_1 x + c_2 y + ... + c_{j-1} x^i + c_j y^i \tag{2.17}$$

then the following equations can be set up and solved to find $c_0, c_1, c_2, ..., c_j$

$$\int_{\Omega} g d\Omega = \int_{\Omega_m} d\Omega \tag{2.18}$$

$$\int_{\Omega} g x d\Omega = \int_{\Omega_m} x d\Omega \tag{2.19}$$

$$\int_{\Omega} g y d\Omega = \int_{\Omega_m} y d\Omega \tag{2.20}$$

$$. \tag{2.21}$$

$$. \tag{2.22}$$

$$. \tag{2.23}$$

$$\int_{\Omega} g x^i d\Omega = \int_{\Omega_m} x^i d\Omega \tag{2.24}$$

$$\int_{\Omega} g y^i d\Omega = \int_{\Omega_m} y^i d\Omega \tag{2.25}$$

The RHS of the above can be determined by the application of divergence theorem which converts them into contour integrals [43]. Thus, the problem of integration of a discontinuous function over a given material domain is now converted to integrating each term in the integrand analytically using divergence theorem with the extra cost of evaluating the function g. As before, one of the main disadvantages of this method is that it doesn't scale well for general 3D curved domains. Furthermore, evaluation of g for higher-order integrands is computationally expensive. To alleviate the latter disadvantage, [12] suggests to choose g to be a lower order polynomial compared to f. However, this places a restriction on the size of the integration cell and in fact warrants a finer integration grid.

2.1.4 Adaptation by Shape Sensitivity Analysis

We could also try to evaluate integral $\int_{\Omega} f(\mathbf{x})\, \mathrm{d}\Omega$ in two steps: first integrate function $f(x)$ over another domain Ω_0 that approximates Ω, and then correct the result by the difference between the two integrals. To the best of our knowledge, this approach has not been considered in the context of integration over arbitrary domain.

One popular approximation is based on the application of first-order Shape Sensitivity Analysis (SSA) [75] from the shape optimization [75] literature. As a first order approximation, consider a reference domain Ω_0 that is homeomorphic to and in the neighborhood of the original domain Ω. In other words, there exists a continuous bijective mapping $\boldsymbol{T}(\mathbf{X}, t)$ (with continuous inverse) that deforms the reference domain into the original domain. In this context, the original domain Ω could also be referred to as the deformed domain. For simplicity, let us choose a homeomorphic piecewise linear domain as our reference domain (see for e.g. Fig. 2.6). On applying SSA, and assuming that f can be suitably extended outside of Ω, we get the following approximation to the integral

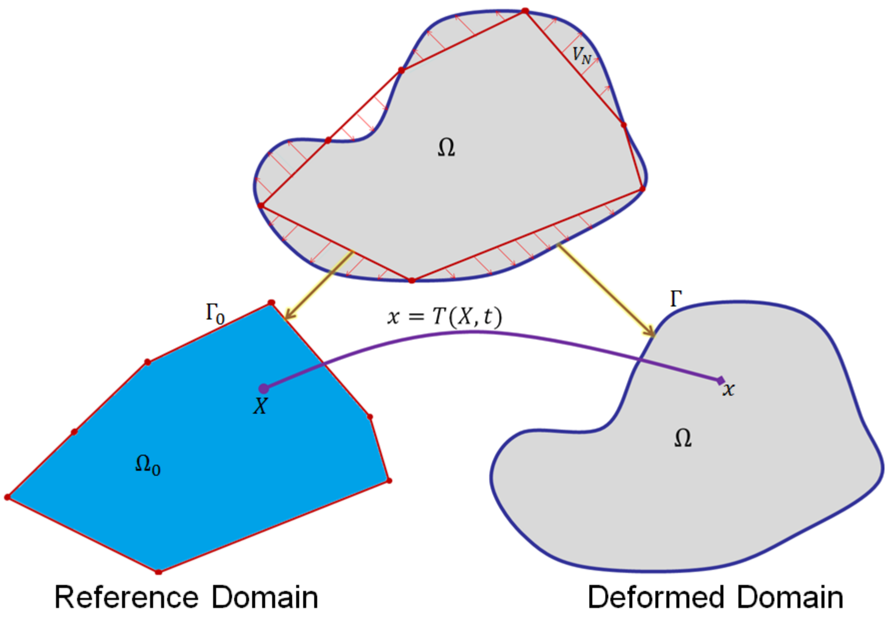


Figure 2.6: Reference and Deformed domains for SSA

$$I = \int_{\Omega} f d\Omega \approx \int_{\Omega_0} f d\Omega_0 + \int_{\Gamma_0} f V_N d\Gamma_0, \tag{2.26}$$

where V_N is the normal component of the design velocity that is defined formally in Section 2.2.1. Thus the original integral over Ω was approximated using SSA over a simpler domain Ω_0. Furthermore, at least in principle, the integrals over the polygonal/polyhedral domain (Ω_0) can be computed using the quadrature rules via the moment fitting equations as explained in [137, 92] (or see Section 2.1.1). Although this method would give a reasonable approximation to the original integral, it suffers from a serious drawback. In order to evaluate Eq.(2.26), function $f(\mathbf{x})$ must be sampled over the polygonal/polyhedral boundary Γ_0. This either assumes that Γ_0 lies completely within the domain, or requires extension of the function to outside the domain (Ω), usually based on application specific heuristic arguments as in [79]. To circumvent this rather severe restriction, the boundary term in Eq.(2.26) could be written in domain form using divergence theorem:

$$I \approx \int_{\Omega_0} [f + \nabla \cdot (f\mathbf{V})] d\Omega_0, \tag{2.27}$$

reducing the original integration over Ω to that over approximate simplified domain Ω_0. Although this eliminates the need for function extension, it does require the domain velocity computation which is computationally expensive, ambiguous and/or not easily implemented [76].

These difficulties perhaps explain why SSA has not been used for integration purposes until now. Note, however, that SSA can be used without any problems when the integrand f is defined everywhere in $\mathbb{R}^d$, as is the case with basis functions used in moment fitting.

2.1.5 Summary

1. *Divergence Theorem* [136, 30, 36, 23, 41] and *Moment fitting equations* [14, 64, 116, 137] work really well for polygonal/polyhedral domains. However, when applied to arbitrary domains leads to significant performance problems. Moreover, *Divergence Theorem* [136, 30, 36, 23, 41] can't be applied to functions that are not symbolically integrable.

2. Out of the *Hierarchical Partitioning* approaches, *Randomly excluding/including boundary cells* [109], *Mapping cells to canonical regions* [78], and *Level-Set Approaches* [31, 100] suffer from lack of accuracy, generality and representation independences respectively. Likewise, *Monte-Carlo method* [132] is computationally expensive as a

lot of quadrature points need to be generated to resolve the boundary. Although, *Convenient Space Parametrization* [131, 58, 22] of boundary cells does provide reasonably accurate results in reasonable time, it has certain heuristic assumptions built into its algorithm that might jeopardize its accuracy for complex domains.

3. Out of the *Characteristic Function* approaches, *Local Meshing* [46, 47, 134] defeats the very purpose of avoiding conforming mesh generation. *Modification of integration weights / nodes* [127, 12] is heuristic in nature and not expected to perform well for arbitrary domains owing to its linear edge/face assumption in its formulation. *Smoothed Step Function Approach* [115, 21] is computationally expensive owing to the introduction of discontinuity in the integrand. *Equivalent Polynomial Approach* [43] works well for simpler domains and might prove to be computationally expensive for real world arbitrary domains owing to its inherent assumption of linear edges/faces in its formulation.

4. *Adaptation by Shape Sensitivity* requires either function evaluation outside the actual domain (of integration) or design velocity computation inside the domain - both of which are not desirable.

The properties of different integration schemes are compared in Table.2.1.

2.1.6 Open Issues

1. An integration technique that efficiently integrates arbitrary integrable functions over arbitrary 3D domains without unreasonable/heuristic assumptions is needed.

2. None of the integrators show any promise of handling small geometric features such as fillets, notches, and other curved regions efficiently.

Integration Schemes	**Accuracy**	**Speed**	**Arbitrary Functions**	**Arbitrary Domain**	**Limiting assumptions**	**Representation Dependencies**	**Heuristic Steps**
Classical Schemes							
Divergence Theorem [136, 30, 36, 23, 41]	High	Average	No	Yes	None	None	None
Moment Fitting [14, 64, 116, 137]	High	Low	Yes	Yes	None	None	None
Heirarchical Partitioning							
Randomly exclude/include boundary cells [109]	Low	High	Yes	Yes	None	None	Yes
Map cells to canonical regions [78]	High	High	Yes	No	Yes	None	None
Convenient Space Parametrization [131, 58, 22]	Medium	High	Yes	Yes	None	None	Yes
Monte-Carlo [132]	High	Low	Yes	Yes	None	None	None
Level-Set Approaches [31, 100]	High	Medium	Yes	May be	None	Yes	None
Characteristic Function Approach							
Local Meshing [46, 47, 134]	High	Medium	Yes	Yes	None	Yes	None
Modification of integration weights / nodes [127, 12]	Average	High	Yes	No	None	None	Yes
Smoothed Step Function Approach [115, 21]	Medium	Low	Yes	Yes	None	None	None
Equivalent Polynomial Approach [43]	High	High	Yes	No	None	None	None
Adaptation by Shape Sensitivity [75]	Medium	High	Yes	Yes	Yes	None	None

Table 2.1: Comparison of Different Integration Schemes

3. Small topological features such as small holes and washers are part of any realistic CAD model. However, none of the integrators listed in this section can handle these small topological features efficiently.

4. A realistic performance comparison of various integrators in integrating arbitrary integrable functions over arbitrary domains is very much needed to further understand the computational aspects of these methods.

2.2 Sensitivity Analysis

In this section we will briefly review certain sensitivity analysis techniques that were primarily developed for shape/topology optimization [75, 50] and defeaturing error estimation [80, 60]. However, it is important to note that we only need the basic definition/ideas from this section to formulate our new quadrature scheme (which will be the subject of chapter 3). This is because, the quantity of interest that we will be dealing with in this thesis is relatively simple and doesn't depend on a field solution that is governed by some PDEs. However, for the sake of completeness, we will present sensitivity analysis in the context of defeaturing error estimation in boundary value problems. Specifically, we will consider the standard Poisson's problem in order to understand the different ways to estimate the effect of small geometric/topological features on any given Quantity of Interest (QOI). To be specific, we will consider the following strong form

$$\begin{aligned} \nabla^2 u &= f \ \forall x \in \Omega - \omega \\ u &= \bar{u} \ \forall x \in \Gamma_D \\ \nabla u.\mathbf{n} &= g_n \ \forall x \in \Gamma_N \\ \nabla u.\mathbf{n} &= 0 \ \forall x \in \Gamma_\omega \end{aligned} \tag{2.28}$$

where ω is an arbitrarily shaped small feature in the domain Ω that is traction free.

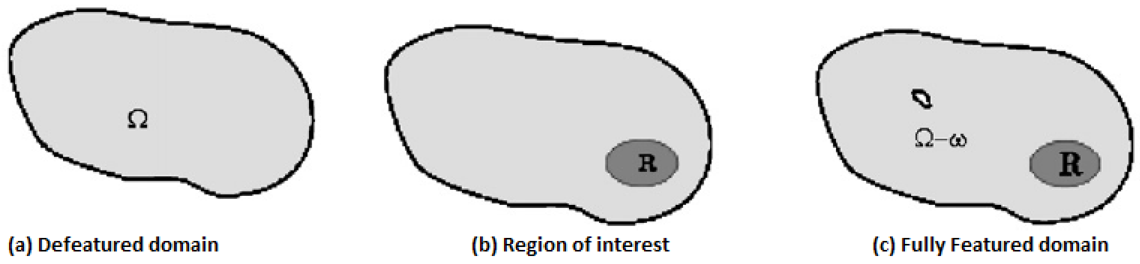


Figure 2.7: Defeatured and fully featured defeatured domains with region of interest [123]

Instead of solving the problem over the fully featured domain (Eq.(2.28)), we will pose and solve a modified form of the problem on the defeatured domain Ω and then correct for the presence of small features using various types of first/second order approximations. The modified problem will thus be posed as

$$\begin{aligned} \nabla^2 u_0 &= f \; \forall x \in \Omega \\ u_0 &= \bar{u} \; \forall x \in \Gamma_D \\ \nabla u_0 . \mathbf{n} &= g_n \; \forall x \in \Gamma_N \end{aligned} \tag{2.29}$$

Also, we will assume that the quantity of interest (QOI) is of the following form

$$Q = \int_R q(u, \nabla u) d\Omega \tag{2.30}$$

where q is some non-linear function defined over a region of interest $R \subset \Omega - \omega$.

2.2.1 Shape Sensitivity

Shape sensitivity [75] is the rate of change of any given Quantity of Interest (QOI) w.r.t. boundary perturbations. In the context of this approach, the positive feature on the boundary of the domain is simply viewed as a deformation of the boundary. Thus, we establish a mapping between the defeatured domain (Ω) and the fully featured domain (Ω_t). Let $\mathbf{T}$ be a homeomorphic mapping that transforms the defeatured domain (reference domain) in to the fully featured domain (deformed domain) i.e. $\mathbf{x} = \mathbf{T}(\mathbf{X}, t)$ (see Fig. 2.8). Thus, $\mathbf{x} \in \Omega_t$ is a function of the shape parameter t. The parameter t denotes the amount of shape change in the design variable direction such that $t = 0$ represents the reference domain Ω_0 [75]. If $\mathbf{T}(\mathbf{X}, t)$ is assumed to be regular enough in the neighborhood of $t = 0$, then it can be expanded using the Taylor series around the initial mapping point $\mathbf{T}(\mathbf{X}, 0)$ as [75]

$$\begin{aligned} \mathbf{x} &= \mathbf{T}(\mathbf{X}, t) \\ &= \mathbf{T}(\mathbf{X}, 0) + t\frac{\partial \mathbf{T}}{\partial t}(\mathbf{X}, 0) + \cdots \\ &= \mathbf{X} + t\mathbf{V}(\mathbf{X}) + \cdots \end{aligned} \tag{2.31}$$

where $\mathbf{V}(\mathbf{X}) = \frac{\partial \mathbf{T}}{\partial t}(\mathbf{X}, 0) = \left.\frac{d\mathbf{x}}{dt}\right|_{t=0}$ is the design velocity vector that is defined over the reference domain Ω_0. First-order shape sensitivity is developed by considering only the first two terms of the above Taylor series. Thus, ignoring the higher order terms, $\mathbf{T}$ is assumed to be a homeomorphic linear mapping in t given by

$$\mathbf{x} = \mathbf{T}(\mathbf{X}, t) = \mathbf{X} + t\mathbf{V}(\mathbf{X}),\ \mathbf{X} \in \Omega. \tag{2.32}$$

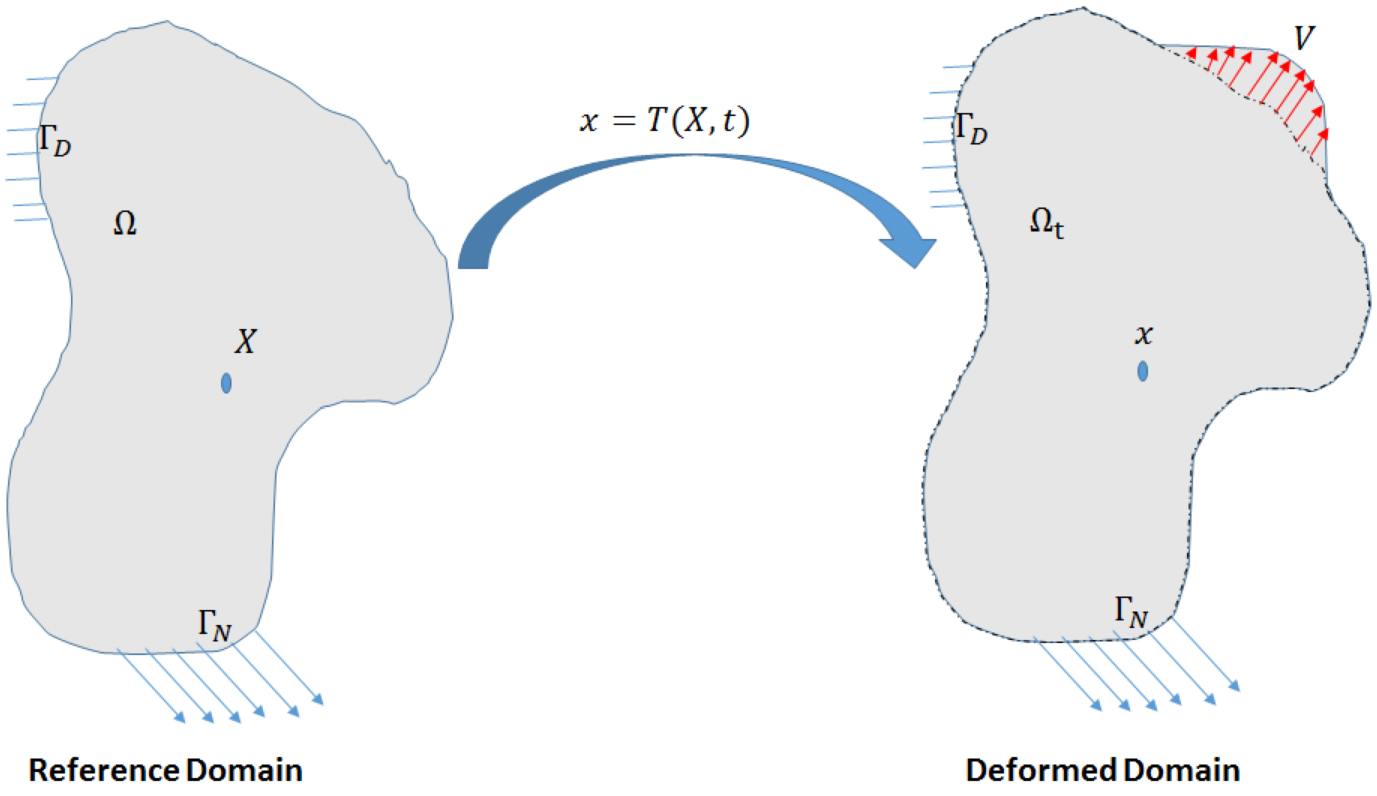


Figure 2.8: Shape Sensitivity : Reference and Deformed Domains

Having defined the mapping, the sensitivity of any *Quantity of Interest* (QOI) to shape perturbation follows directly from the Reynold's Transport Theorem and the definition of adjoint [75]

$$\frac{dQ}{dt} = \int_{d\Gamma_t} [\nabla\lambda.\mathbf{V}\nabla u.\mathbf{n} + \nabla u.\mathbf{V}\nabla\lambda.\mathbf{n} + V_n(q + f\lambda - \nabla u.\nabla\lambda)]d\Gamma_t - \int_{\Gamma_N} \lambda V_n g_n d\Gamma_t \quad (2.33)$$

where $V_n = \mathbf{V}.\mathbf{n}$ is the normal component of the design velocity and λ is the solution of the adjoint weak form : Find $\lambda \in H_0^1$ s.t.

$$\int_{\Omega_t} \nabla\lambda.\nabla v d\Omega_t = \int_{\Omega_t} [\nabla_u q v + \nabla_{\nabla u} q \nabla v] d\Omega_t \ \forall \ \in H_0^1 \quad (2.34)$$

It is important to note that for the example in Fig. 2.8 the velocity is zero everywhere except on the feature boundary (as indicated by red arrows in the deformed domain).

Having determined the shape sensitivity for the QOI, we can estimate the QOI over the fully featured domain ($Q(u)$) using first-order Taylor series as

$$Q(u) \approx Q(u_0) + t\frac{dQ}{dt}\bigg|_{t=0} \tag{2.35}$$

Ming Li et al.[87] extended this idea to positive/negative boundary features using second-order shape sensitivity [76].

2.2.2 Topological Sensitivity

The topological derivative is the sensitivity of a quantity of interest (Q) defined over a region of interest (in the domain) when an infinitesimal hole is introduced in the domain [67]. Formally, topological sensitivity is defined as

$$D_T^1(\mathbf{x}) = \lim_{\epsilon \to 0} \frac{Q(\Omega_\epsilon) - Q(\Omega)}{f_1(\epsilon)} \tag{2.36}$$

where $\Omega_\epsilon = \Omega - B_\epsilon$ and $f_1(\epsilon)$ is a monotonically decreasing function that depends on the boundary value problem under consideration. The ball B_ϵ has a radius of ϵ with its center at the point $\mathbf{x} \in \Omega$. For the standard Poisson's problem, we have $f_1(\epsilon)$ to be the measure of the ball B_ϵ and hence the topological derivative can be established as $D_T^1(\mathbf{x}) = g_n(\mathbf{x})\lambda_0(\mathbf{x})$ for $g_n \neq 0$ and $D_T^1(\mathbf{x}) = 2[\nabla u_0(\mathbf{x})]^T \nabla\lambda_0(\mathbf{x}) - f(\mathbf{x})\lambda_0(\mathbf{x})$ for $g_n = 0$ [107]. g_n is the Neumann data given by Eq.(2.29) and λ_0 is the adjoint solution of the following adjoint problem

$$\begin{aligned} \nabla^2\lambda_0 &= \nabla_u[q] \; \forall x \in \Omega \\ \lambda_0 &= 0 \; \forall x \in \Gamma_D \\ \nabla\lambda_0.\mathbf{n} &= 0 \; \forall x \in \Gamma_N \end{aligned} \tag{2.37}$$

Various methods [106, 17, 18, 19] have been proposed to compute the topological derivative. For a finite spherical hole of radius r with center at $\hat{\mathbf{x}} \in \Omega$, one could approximate its effect on the QOI as

$$Q(u) \approx Q(u_0) + \mu(B_r) * D_T^1(\hat{\mathbf{x}}) \tag{2.38}$$

where $Q(u_0)$ is the QOI computed from the solution (u_0) of the boundary value problem over the defeatured domain Ω and $\mu(B_r)$ is the measure (area or volume) of the ball of radius r. For an arbitrary small feature (ω) of size ϵ one could equivalently define the following estimate for the QOI in the fully featured domain

$$Q(u) \approx Q(u_0) + \mu(B_\epsilon) * D_T^1(\hat{\mathbf{x}}) \tag{2.39}$$

For relatively larger features, this estimate could be erroneous.

2.2.3 Feature Sensitivity

Feature sensitivity [123, 59, 60] is a generalization of topological sensitivity to finite sized features. The central idea in this method is to parametrize the hole boundary using a scaling parameter η as follows

$$T(\eta) : \mathbf{p} = \begin{cases} (\mathbf{P} - \mathbf{P_c})\eta + \mathbf{P_c} & : \mathbf{P} \in \Gamma_\omega \\ \mathbf{P} & : \mathbf{P} \in \Gamma \end{cases}$$

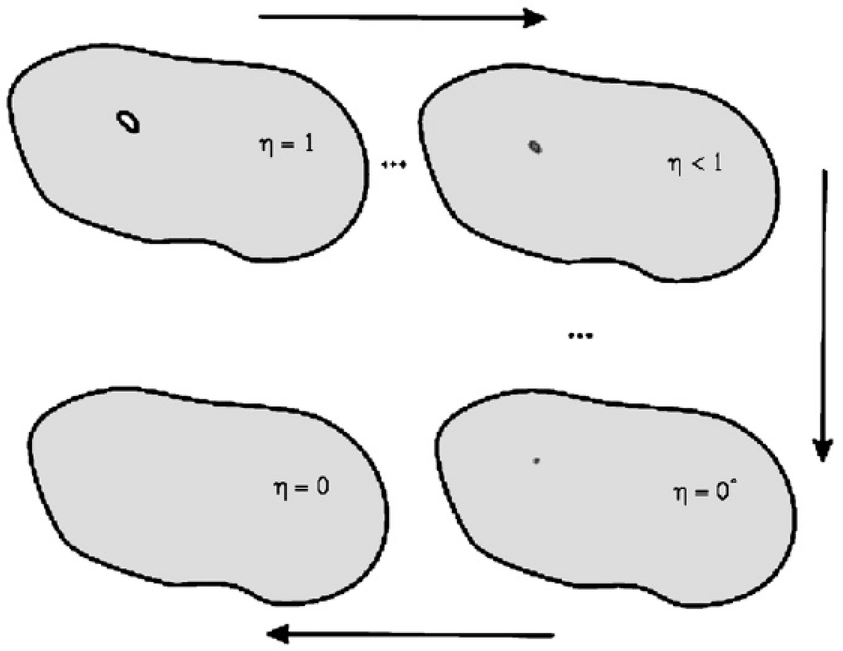


Figure 2.9: Feature Sensitivity : Transformation [123]

where $\mathbf{p}$ is any point on the hole boundary and $\mathbf{P_c}$ is the center of the smallest spherical hole that contains the given hole. For $\eta = 1$ and $\eta = 0$ we get the fully featured and defeatured domains respectively. For $\eta = 0^+$ we get the domain with a point hole. For $0 < \eta < 1$, we get the domain with a shrunk hole (see Fig. 2.9). The above parametrization also leads to the following strong form on the parametrized domain

$$\begin{aligned} \nabla^2 u_\eta &= f \ \forall x \in \Omega - \omega_\eta \\ u_\eta &= \bar{u} \ \forall x \in \Gamma_D \\ \nabla u_\eta.\mathbf{n} &= g_n \ \forall x \in \Gamma_N \\ \nabla u_\eta.\mathbf{n} &= 0 \ \forall x \in \Gamma_{\omega_\eta} \end{aligned} \tag{2.40}$$

Analogously the QOI over the parametrized domain can be defined as

$$Q_\eta = \int_R q(u_\eta, \nabla u_\eta) d\Omega \tag{2.41}$$

Evaluating the above at $\eta = 0$, we obtain the QOI as defined over the defeatured domain as

$$Q_0 = \int_R q(u_0, \nabla u_0) d\Omega \tag{2.42}$$

Thus, the error in the QOI could be written as

$$Q_1 - Q_0 = \int_0^1 \frac{dQ_\eta}{d\eta} d\eta \tag{2.43}$$

In order to compute the sensitivity $\frac{dQ_\eta}{d\eta}$, it is required to solve the following adjoint problem for the adjoint solution λ_η on the fully featured domain $\Omega - \omega_\eta$

$$\begin{aligned} \nabla^2 \lambda_\eta &= \nabla_u[q] \ \forall x \in \Omega - \omega_\eta \\ \lambda_\eta &= 0 \ \forall x \in \Gamma_D \\ \nabla \lambda_\eta.\mathbf{n} &= 0 \ \forall x \in \Gamma_N \cup \Gamma_{\omega_\eta} \end{aligned} \tag{2.44}$$

The corresponding adjoint problem over the defeatured domain Ω could be stated as

$$\begin{aligned} \nabla^2 \lambda_0 &= \nabla_u[q] \ \forall x \in \Omega \\ \lambda_0 &= 0 \ \forall x \in \Gamma_D \\ \nabla \lambda_0.\mathbf{n} &= 0 \ \forall x \in \Gamma_N \end{aligned} \tag{2.45}$$

From standard shape sensitivity analysis and the definition of adjoint we have

$$\frac{dQ_\eta}{d\eta} = \int_{\Gamma_{\omega_\eta}} [q + f\lambda_\eta - \nabla u_\eta . \nabla \lambda_\eta] V_n d\Gamma \tag{2.46}$$

where V_n is the normal component of the design velocity and $V = \frac{\mathbf{p}-\mathbf{P_c}}{\eta}$ on the hole boundary and zero everywhere else. However, the above equation requires that we solve boundary value problems on the fully featured (or scaled) domain which we want to avoid. Hence, the solutions Λ_η and u_η are approximated as follows

$$u_\eta\Big|_{\Gamma_{\omega_\eta}} \approx u_0\Big|_{\mathbf{P_c}} + \eta u_E\Big|_{\Gamma_{\omega_1}} \tag{2.47}$$

$$\nabla u_\eta\Big|_{\Gamma_{\omega_\eta}} \approx \nabla u_0\Big|_{\mathbf{P_c}} + \eta \nabla u_E\Big|_{\Gamma_{\omega_1}} \tag{2.48}$$

where u_E satisfies the following exterior boundary value problem

$$\begin{aligned} \nabla^2 u_E &= 0 \;\forall x \in \mathbb{R}^n - \omega \\ u_E \to 0 \quad as &\quad \mathbf{P} \to \infty \\ \nabla u_E.\mathbf{n} &= \nabla u_0\Big|_{\mathbf{P_c}} .\mathbf{n}\,\forall x \in \Gamma_\omega \end{aligned} \tag{2.49}$$

Now, feature sensitivity could be used to estimate the effect of an arbitrary feature (ω) on the QOI as

$$Q(u) \approx Q(u_0) + \int_{\Gamma_\omega} [F_1 + F_2] V_n d\Gamma \tag{2.50}$$

where

$$\begin{aligned} F_1 &= \frac{(q + f\lambda_0)\Big|_{\mathbf{P_c}} - [\nabla u_0\Big|_{\mathbf{P_c}} + \nabla u_E].[\nabla \lambda_0\Big|_{\mathbf{P_c}} + \nabla \lambda_E]}{2} \\ F_2 &= \frac{u_E \nabla_u [q]\Big|_{\mathbf{P_c}} + f\lambda_E}{3} \end{aligned}$$

Thus, the first step in the approximation to the QOI is the computation of λ_0 and u_0 by solving the defeatured primal and adjoint problems. The second step is to compute λ_E and u_E by solving the exterior primal and adjoint problems. Finally, using these quantities in Eq.(2.50), we can compute the approximation to the QOI over the fully featured domain. This method in general can't handle non-zero/zero Dirichlet boundary conditions and non-zero Neumann boundary conditions over the feature boundary. In addition, this method is limited to small/moderately sized features.

2.2.4 Modification Sensitivity

This was first introduced by Ming Li et al. [80] to capture the effect of introducing a negative feature using the idea of modeling error estimation of Oden et al. [65]. The problem of estimating the modeling error is defined for the following two abstract nonlinear problems expressed in weak form : find solutions u and u_0 such that

$$N(u; v) = F(v) \ \forall v \in V \tag{2.51}$$

and

$$N_0(u_0; v) = F_0(v) \ \forall v \in V \tag{2.52}$$

where $N(,;.)$ and $N_0(,;.)$ are semilinear forms defined on a Banach Space V, and $F(.)$, $F_0(.)$ are linear functionals on V. Eq.(2.51) and Eq.(2.52) represents two different models modeling the same physical phenomena. Thus, the problem of modeling error estimation is to estimate $Q(u) - Q(u_0)$ without explicitly solving u from Eq.(2.51). The adjoint problems for the two models are defined as follows

$$N^{'}(u; v, \lambda) = Q^{'}(u; v) \ \forall v \in V \tag{2.53}$$

and

$$N_0^{'}(u_0; v, \lambda_0) = Q_0^{'}(u_0; v) \ \forall v \in V \tag{2.54}$$

where $'$ denotes Gateaux derivative [65]. The residual functional to characterize the degree to which u_0 fails to satisfy the problem in Eq.(2.51) is defined as

$$R(u_0; v) = F(v) - N(u_0; v)\ \forall v \in V \tag{2.55}$$

Oden et al. [65] prove that the modeling error can be approximated as

$$Q(u) - Q(u_0) \approx R(u_0; \lambda) \tag{2.56}$$

Ming Li et al. [80] uses this idea of modeling error estimation in defining their modification sensitivity by introducing an intermediate problem. Considering our model Poisson's problem, the strong form for the Poisson's problem over the fully featured and defeatured domains are once again given by Eq.(2.28) and Eq.(2.29) respectively. However, to make use of the modeling error estimation idea we require that the two models (BVPs) are defined over the same domain. Ming Li et al. [80] observe that the fully featured domain is contained in the defeatured domain, that is $\Omega - \omega \subset \Omega$. Hence, the defeatured solution u_0 defined over the geometry $\Omega - \omega$ is well defined, and is denoted by

$$\bar{u_0} = u_0\Big|_{\Omega - \omega} \tag{2.57}$$

Thus, the solution $\bar{u_0}$ can be seen as the solution of another engineering analysis problem defined over the model $\Omega - \omega$ with particular Neumann conditions prescribed over the internal boundary Γ_ω : find $\bar{u_0}$ such that

$$\begin{aligned}
\nabla^2 \bar{u_0} &= f\ \forall x \in \Omega - \omega \\
\bar{u_0} &= \bar{u}\ \forall x \in \Gamma_D \\
\nabla \bar{u_0}.\mathbf{n} &= g_n\ \forall x \in \Gamma_N \\
\nabla \bar{u_0}.\mathbf{n} &= h_n\ \forall x \in \Gamma_\omega
\end{aligned} \tag{2.58}$$

Now, we have Eq.(2.28) and Eq.(2.58) are both defined on the same domain and thus the modeling error in this case leads to the following modeling sensitivity

$$Q(u) - Q(u_0) = Q(u) - Q(\bar{u_0}) \approx R(\bar{u_0}; \lambda) \tag{2.59}$$

Further, Ming Li et al. [80] prove based on certain heuristics that

$$Q(u) - Q(u_0) \approx R(\bar{u_0}; \lambda_0) \tag{2.60}$$

For the standard Poisson's problem the above turns out to be a boundary integral over the feature boundary

$$Q(u) \approx Q(u_0) + \int_{\Gamma_\omega} (g_n - \nabla \bar{u_0}.\mathbf{n})\lambda_0 d\Gamma \tag{2.61}$$

Thus, solving the primal and adjoint problems for u_0 and λ_0 over the defeatured domain (Ω), the QOI over the fully featured domain can be estimated using the above equation.

Ming Li et al. [80] extended their idea of applying modification sensitivity to positive boundary features [79] by extending the solution u_0 to the domain $\Omega + \omega$. This is required as, unlike the negative features, $\Omega + \omega \not\subset \Omega$. Hence, instead of Eq.(2.57) we have $\bar{u_0}$ defined using the solution u_0^f to the following boundary value problem

$$\begin{aligned} \nabla^2(u_0^f) &= f \; \forall x \in \omega \\ \nabla u_0^f.\mathbf{n} &= g_n \; \forall x \in \Gamma_\omega - I \\ u_0^f &= u_0 \; \forall x \in I \end{aligned} \tag{2.62}$$

where I is the common boundary portion (known as the feature interface) between the defeatured model Ω and the positive feature ω. Thus, we have the extended solution $\bar{u_0}$

defined as

$$\bar{u_0} = \begin{cases} u_0 \; in \; \Omega \\ u_0^f \; in \; \omega \end{cases}$$

With this extended solution $\bar{u_0}$, the QOI over the fully featured domain is once again given by Eq.(2.61).

This method nicely generalizes to 3D domains and can handle even large features. In addition, it can handle non-zero Neumann boundary conditions over the hole boundary. One major shortcoming of the method is that the use of $\lambda = \lambda_0$ in the derivation of modification sensitivity is not very well justified. In [81], Ming Li et al. approximate the dual solution λ by computing an exterior solution. However, this in turn limits the application of the method to domain features that are far away from the boundary. Also, the method can't in general handle zero/non-zero Dirichlet boundary conditions over the feature boundary. Moreover, the heuristic assumption used in the extension of solution to $\Omega + \omega$ in the case of positive features is not fully justified.

2.2.5 Summary

1. Shape Sensitivity [75] is the rate of change of any given Quantity of Interest (QOI) w.r.t. boundary perturbations. SSA can be effectively used to estimate the effect of adding or removing small features over the boundary.

2. The topological derivative is the sensitivity of any QOI defined over a region of interest (in the domain) when an infinitesimal hole is introduced in the domain. This can be effectively used to estimate the effect of adding a small negative feature on the boundary or interior of the domain.

3. Feature sensitivity (FSA) [123, 59, 60] and modification sensitivity (MSA) [80] extends the notion of SSA/TSA to finite sized features. Thus, FSA and MSA can be used to efficiently estimate the effect of finite sized small/large features on any given QOI.

Chapter 3

Shape Aware Quadratures (SAQ)

In this chapter, we will develop a new quadrature scheme called the "Shape Aware Quadratures" (SAQ) that can be used to efficiently integrate arbitrary integrable functions over arbitrary domains in the presence of small features. We will present a clear formulation of the method, its algorithmic details including the analysis, and benchmark studies to illustrate its computational properties in 2D and 3D. However, we will first begin with a formal definition and classification of small features based on morphological operators.

3.1 Small Features

In this section we will formally define and classify small features. In order to define small features, we shall first define the following morphological operators [125]

1. **Erosion** of a set A by a structuring set B is defined mathematically as
   $A \ominus B = \{a \in A \mid B_a \subseteq A\}$ where $B_a = \{b + a \mid b \in B\}, \forall a \in A$

2. **Dilation** of a set A by a structuring set B is defined mathematically as
   $A \oplus B = \{a + b \mid a \in A, b \in B\}$

3. **Opening** is the dilation of the erosion of a set A by B i.e.
   $A \circ B = (A \ominus B) \oplus B$

4. **Closing** is the erosion of the dilation of a set A by B i.e.

   $A \bullet B = (A \oplus B) \ominus B$

Definition

The union of small features of a domain A relative to an user defined size δ is given by one of the following operations

1. $A - (A \circ B)$
2. $(A \bullet B) - A$

where the structuring set B is a ball of diameter δ. We classify small features into two primary types - (1) positive and (2) negative small features. All positive features of a domain A are given by $A - (A \circ B)$ and all the negative features of a domain are filled by $(A \bullet B) - A$. Fig. 3.1 and Fig. 3.2 illustrates the application of small features definition to positive features. Likewise, Fig. 3.3 illustrates the application of small features definition to negative features.

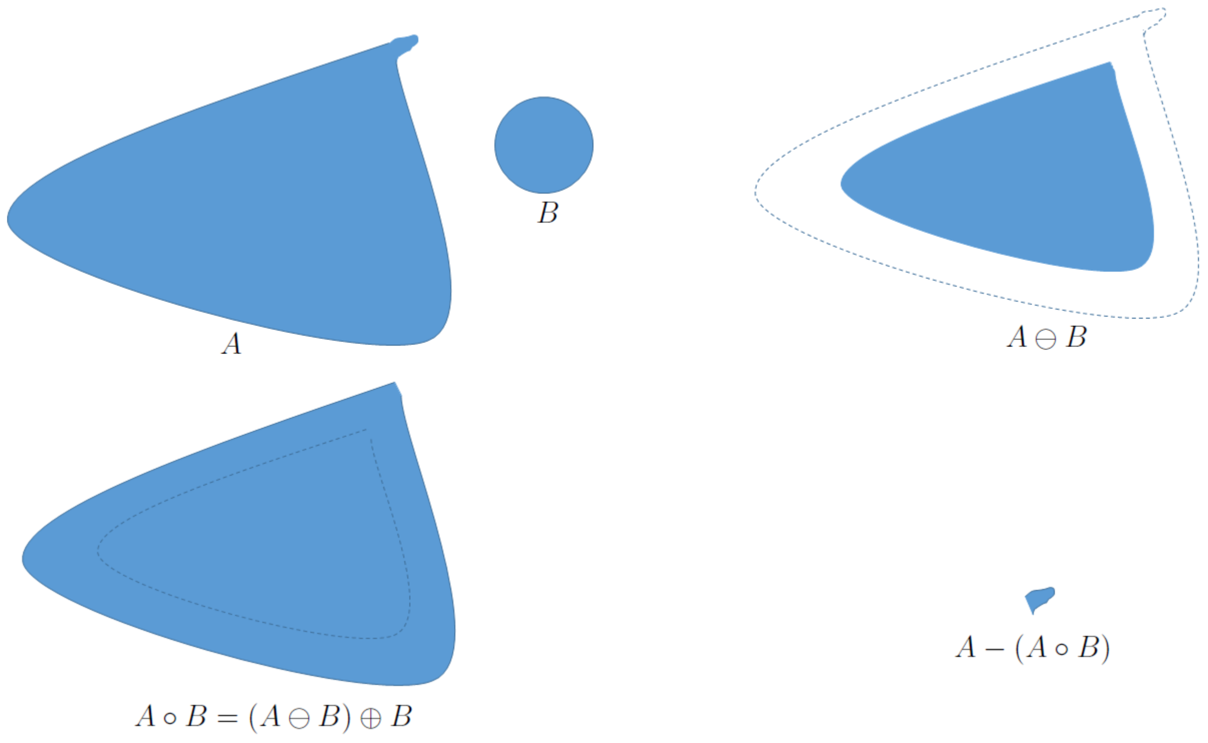


Figure 3.1: Small features definition applied to positive features on the boundary

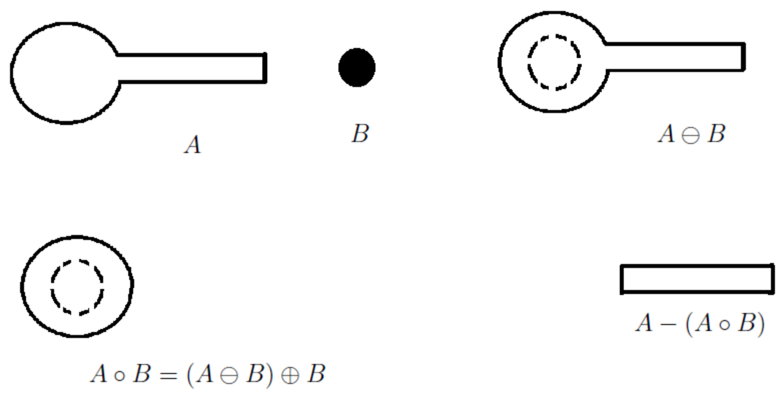


Figure 3.2: Small features definition applied to thin positive features

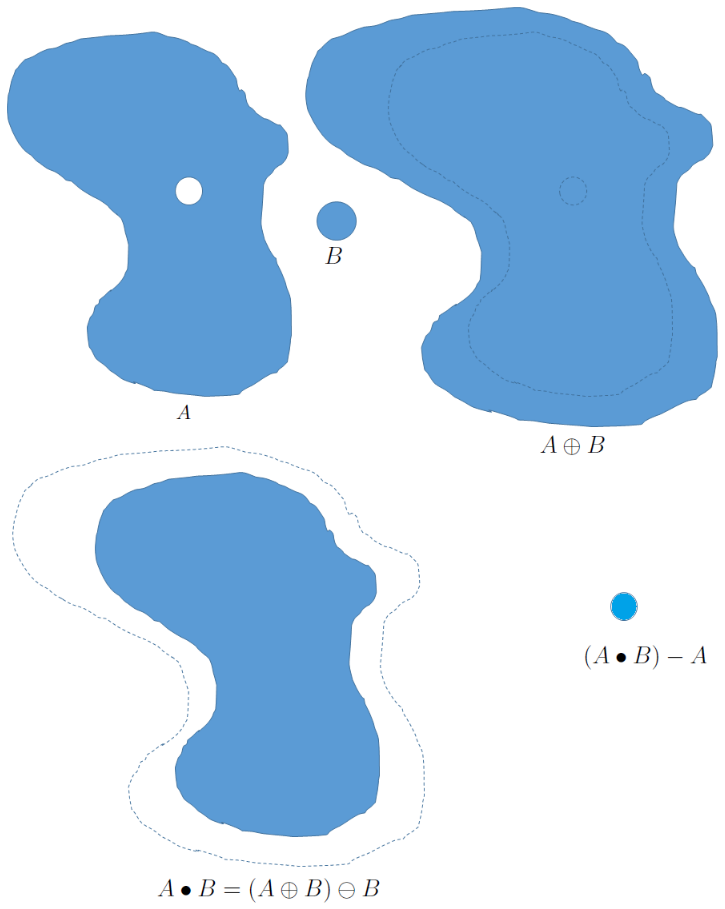


Figure 3.3: Small features definition applied to negative features

Small features can be found in most Engineering components. One typical example is the car wheel assembly as shown in Fig. 3.4.

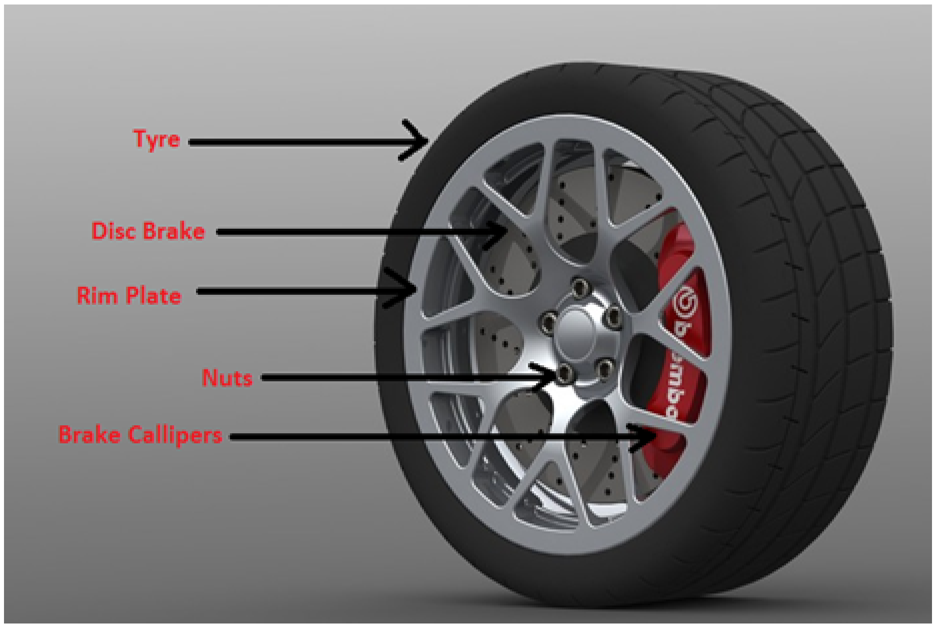


Figure 3.4: A typical car wheel assembly containing numerous small positive and negative features

In the above, the rim plate has a lot of thin positive regions, the disc brake has a lot of small holes (negative features), the nuts are small positive regions, and the surface of the tire has numerous small positive/negative features.

3.2 Formulation

Shape Aware Quadratures (SAQ) are quadratures that adapt to the geometry and topology (i.e. shape) of the integration domain automatically/efficiently. The primary application of this is in the integration of arbitrary integrable functions over an arbitrary domain (in the presence of small features) using a non-conforming mesh. However, SAQ can be used with or without a mesh.

Let us begin by rewriting the moment fitting equations in Eq.(2.7) by setting $W(\mathbf{X}) = 1$ as

$$\begin{bmatrix} \int_\Omega b_1(\mathbf{x})\mathrm{d}\Omega \\ \int_\Omega b_2(\mathbf{x})\mathrm{d}\Omega \\ \cdots \\ \int_\Omega b_m(\mathbf{x})\mathrm{d}\Omega \end{bmatrix} = \begin{bmatrix} b_1(\mathbf{x_1}) & b_1(\mathbf{x_2}) & \cdots & b_1(\mathbf{x_n}) \\ b_2(\mathbf{x_1}) & b_2(\mathbf{x_2}) & \cdots & b_2(\mathbf{x_n}) \\ \cdots & \cdots & \cdots & \cdots \\ b_m(\mathbf{x_1}) & b_m(\mathbf{x_2}) & \cdots & b_m(\mathbf{x_n}) \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \cdots \\ w_n \end{bmatrix} \tag{3.1}$$

The same equations may be written in a more compact form as

$$\{\mathbf{M}\}_{m\times 1} = [\mathbf{A}]_{m\times n}\{\mathbf{w}\}_{n\times 1} \tag{3.2}$$

The above is a system of non-linear equations that can be solved for the position ($\mathbf{x_i}$) and weights (w_i) using an iterative solution scheme such as the Newton-raphson [132]. However, such solution schemes are computationally very expensive and often do not scale for larger problems. But, if we fix the position of the integration points as in [121], the equations become linear (in weights) and can be solved easily. Knowing $\{\mathbf{M}\}$, one could solve the moment equations for weights $\{\mathbf{w}\}$ using linear least squares as

$$\{\mathbf{w}\}_{n\times 1} = [\mathbf{A}^{\dagger}]_{n\times m}\{\mathbf{M}\}_{m\times 1} \tag{3.3}$$

where $[\mathbf{A}^{\dagger}]$ is the Moore-Penrose pseudo inverse [42]. One popular stable algorithm to numerically compute $[\mathbf{A}^{\dagger}]$ is based on the QR factorization [82].

Thus, given an arbitrary domain Ω with a set of appropriately chosen quadrature nodes and order of integration, quadrature weights are obtained by solving a system of linear moment fitting equations [64, 116, 137] for an appropriate set of basis functions in the least square sense. Setting up the moment-fitting equations involves computing integrals of the basis functions (known as moments) over an arbitrary geometric domain. This task itself is non-trivial, because it either entails some kind of domain decomposition, or can be reduced to repeated boundary integration as was proposed in [16] for bivariate domains. Adaption of the latter approach avoids excessive domain fragmentation, but leads to a significant computational overhead, particularly in 3D.

This challenge is overcome by first computing the moments (i.e. integral of basis functions b_i) over a simpler domain Ω_0, usually a polygon/polyhedron, that is a reasonable approximation (in a Hausdorff distance sense) of the original domain Ω (see Fig. 3.5). The computed moments are then corrected for the deviation of the shape of Ω_0 from Ω based on sensitivity analysis. Thus,

$$M_i \approx M_i^0 + C_i^S \tag{3.4}$$

C_i^S is the i^{th} shape correction factor and M_i^0 is the i^{th} moment computed over the simplified domain Ω_0 i.e. $M_i^0 = \int_{\Omega_0} b_i(\mathbf{X})\,\mathrm{d}\Omega_0$. In general, the application of these sensitivities results in correction factors that are boundary integrals over some domain. In other words, the shape correction factor (C_i^S) is of the following form

$$C_i^S = \sum_{f=1}^{N_f} \int_{\Gamma_f} g_f(b_i, \mathbf{X}) d\Gamma_f \tag{3.5}$$

where $\Gamma_f \subset \mathbb{R}^{d-1}$ is the boundary of a domain that depends on the type of sensitivity employed in computing the corrections. For example, if the correction factor was computed based on shape sensitivity analysis (SSA), then Γ_f is usually a polygonal/polyehdral boundary that approximates Γ as shown in Fig. 3.8.

The moment approximations can entirely be written as a boundary integral by the application of divergence theorem [136, 30, 36, 23, 41] i.e. M_i^0 in Eq.(3.4) can be replaced by Eq.(2.5) as

$$M_i\Big|_{\Omega} \approx \int_{\Gamma_0} \beta_X^i(\mathbf{X}) N_X d\Gamma_0 + \sum_{f=1}^{N_f} \int_{\Gamma_f} g_f(b_i, \mathbf{X}) d\Gamma_f \; for \; i \in \{1, 2, ..., m\} \tag{3.6}$$

g_f depends on the type and order of sensitivity used. Its exact form can be deduced from Table.3.1 and Table.3.2 for SSA and TSA respectively. For the sake of convenience and understanding, let us write the shape correction factor as the sum of geometric (C_i^G) and topological correction factors (C_i^T) i.e.

$$M_i \approx M_i^0 + C_i^G + C_i^T \tag{3.7}$$

C_i^G and C_i^T respectively account for the geometric and topological deviations of Ω_0 from Ω in computing the i^{th} moment approximation. If the simplified domain Ω_0 is chosen homeomorphic[2] to Ω then $C_i^T = 0$ (see Fig. 3.6) and hence the above reduces to the following :

$$M_i \approx M_i^0 + C_i^G \tag{3.8}$$

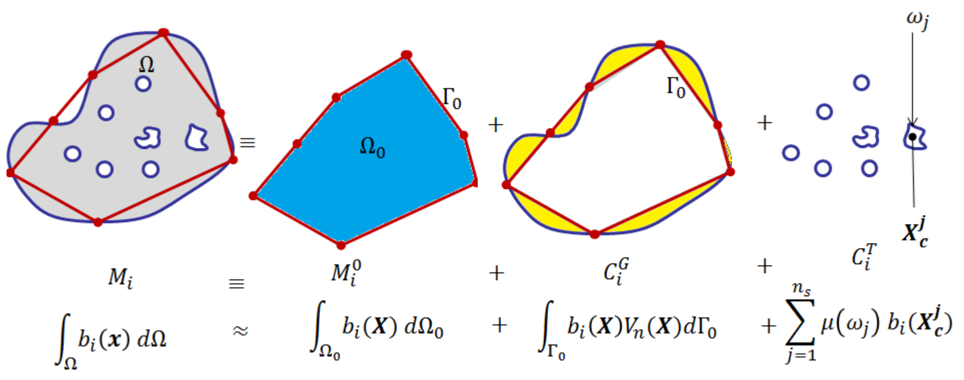

Ω
$\equiv$
Γ_0
Ω_0
$+$
Γ_0
$+$
ω_j
$\boldsymbol{X}_c^j$
M_i
$\equiv$
M_i^0
$+$
C_i^G
$+$
C_i^T
$\int_\Omega b_i(\boldsymbol{x})\, d\Omega$
$\approx$
$\int_{\Omega_0} b_i(\boldsymbol{X})\, d\Omega_0$
$+$
$\int_{\Gamma_0} b_i(\boldsymbol{X}) V_n(\boldsymbol{X}) d\Gamma_0$
$+ \sum_{j=1}^{n_s} \mu(\omega_j)\, b_i(\boldsymbol{X}_c^j)$

Figure 3.5: SAQ - Simplified domain Ω_0 is not homeomorphic to Ω

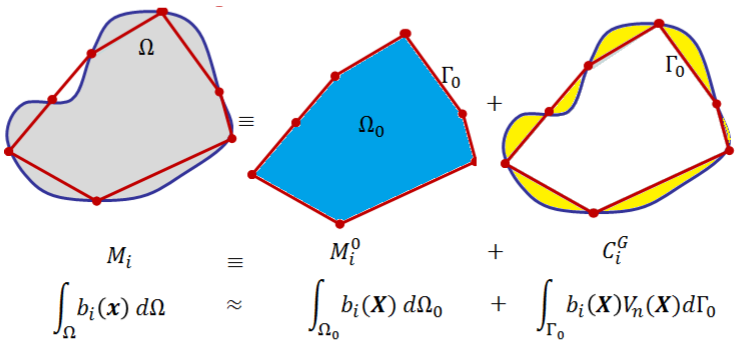

Ω
$\equiv$
Γ_0
Ω_0
$+$
Γ_0
M_i
$\equiv$
M_i^0
$+$
C_i^G
$\int_\Omega b_i(\boldsymbol{x})\, d\Omega$
$\approx$
$\int_{\Omega_0} b_i(\boldsymbol{X})\, d\Omega_0$
$+$
$\int_{\Gamma_0} b_i(\boldsymbol{X}) V_n(\boldsymbol{X}) d\Gamma_0$

Figure 3.6: SAQ - Simplified domain Ω_0 is homeomorphic to Ω

[2]by homeomorphic we mean the mapping $\mathbf{T}(\mathbf{x}, t)$ that we assume in Eq.(2.32) is continuous and bijective with a continuous inverse

C_i^G and C_i^T can be obtained using a variety of sensitivity analysis techniques as discussed in the previous chapter. For instance, if we consider the domain Ω and its approximation Ω_0 as shown in Fig. 3.5, then one could employ employ first-order Shape Sensitivity Analysis (SSA) [75] for C_i^G and Topological Sensitivity Analysis (TSA) [67, 17] for C_i^T to obtain the following moment approximations

$$M_i \approx \int_{\Gamma_0} \beta_X^i(\mathbf{X}) N_X d\Gamma_0 + \int_{\Gamma_0} b_i(\mathbf{X}) V_N(\mathbf{X}) d\Gamma_0 - \sum_{j=1}^{n_s} \mu(\omega_j) b_i(\mathbf{X_c^j}) \tag{3.9}$$

where $\mathbf{V} = t\hat{\mathbf{V}}$ is the *design velocity* and $V_N(\mathbf{X}) = \mathbf{V}(\mathbf{X}).\mathbf{N}(\mathbf{X})$ is the normal component of the design velocity at a point $\mathbf{X}$ on the boundary Γ_0 of the reference (piecewise linear) domain Ω_0. $\mu(\omega_j)$ and $\mathbf{X_c^j}$ are respectively the measure and centroid of the j^{th} negative small feature ω_j. Here, C_i^G corrects for the deviation of Ω_0 (without holes) from Ω and C_i^T accounts for the presence of the small elliptical holes. The correction C_i^G can be easily derived by the straight forward application of first-order SSA [75] to the moments (for details refer to Appendix A). Likewise, the correction term C_i^T can be obtained by the straight forward application of first-order TSA [17] (for details refer to Appendix B).

Likewise, employing first-order SSA for both C_i^G and C_i^T we get

$$M_i \approx \int_{\Gamma_0} \beta_X^i(\mathbf{X}) N_X d\Gamma_0 + \int_{\Gamma_0} b_i(\mathbf{X}) V_N(\mathbf{X}) d\Gamma_0 + \sum_{j=1}^{n_s} \int_{\Gamma_j^0} [\beta_X^i(\mathbf{X}) N_X + b_i(\mathbf{X}) V_N(\mathbf{X})] d\Gamma_j^0 \tag{3.10}$$

where Γ_j^0 is the approximate polygonal/polyhedral domain chosen homeomorphic to the boundary of the small feature ω_j. Alternatively, one could employ first-order SSA for C_i^G and divergence theorem (or zeroth-order SSA) for C_i^T resulting in the following

$$M_i \approx \int_{\Gamma_0} \beta_X^i(\mathbf{X}) N_X d\Gamma_0 + \int_{\Gamma_0} b_i(\mathbf{X}) V_N(\mathbf{X}) d\Gamma_0 + \sum_{j=1}^{n_s} \int_{\Gamma_j^0} [\beta_X^i(\mathbf{X}) N_X] d\Gamma_j^0 \tag{3.11}$$

If there are no topological features (as in Fig. 3.6), and if we employ first-order SSA for C_i^G, then Eq.(3.10) reduces to the following

$$M_i \approx \int_{\Gamma_0} \beta_X^i(\mathbf{X}) N_X d\Gamma_0 + \int_{\Gamma_0} b_i(\mathbf{X}) V_N(\mathbf{X}) d\Gamma_0 \tag{3.12}$$

For 2D domains, this boundary integral could be obtained simply by employing the usual Gaussian quadrature over the edges of the polygon. Similarly, integration over 3D domains bounded by triangles or convex quadrilaterals is straightforward. For more general 3D domains, we proceed in two steps. First, we compute the appropriate weights for a chosen set of quadrature nodes for each of the polygonal faces of the approximating polyhedron by solving the moment fitting equations (Eq.(3.3)) with exact $\{\mathbf{M}\}$ given by Eq.(2.5). Then, using the determined weights for the polygonal faces, we obtain approximate $\{\mathbf{M}\}$ for the arbitrary 3D domain by numerically evaluating Eq.(3.5) over the faces of the polyhedron. Using this approximate $\{\mathbf{M}\}$, the approximate weights for the arbitrary domain are obtained by solving the moment fitting equations i.e. Eq.(3.3).

The approximate moment fitting equations, thus obtained, can then be easily solved for quadrature weights that adapt to the shape of the original domain. The resulting weights are not exact, but are accurate enough to integrate functions over any arbitrary domain when they are well approximated by the chosen set of basis functions. This integration scheme could be advantageously employed in cell decomposition methods, for example using quadtrees [70] or octrees [96], to reduce fragmentation. In summary, C_i^G and C_i^T in Eq.(3.7) are the shape correction factors that accounts for the shape deviation of the original domain Ω from the polygonal/polyhedral domain Ω_0. In other words, these two terms ensures that the quadrature rule determined by the moment fitting equations (Eq.(3.3)) is "aware" of the shape of the integration domain – hence the name *Shape Aware Quadratures (SAQ)*. In fact, the exact moments for a polygonal/polyhedral domain can be recovered from Eq.(3.7) by simply omitting these two terms. In general, as the polygonal/polyhedral approximation (Γ_0) approaches (in Hausdorff sense) the boundary of the original domain (Γ), the two terms goes to 0, and the first term approaches the exact moments for the original domain. An high-level overview of the method is given as a flowchart in Fig. 3.7.

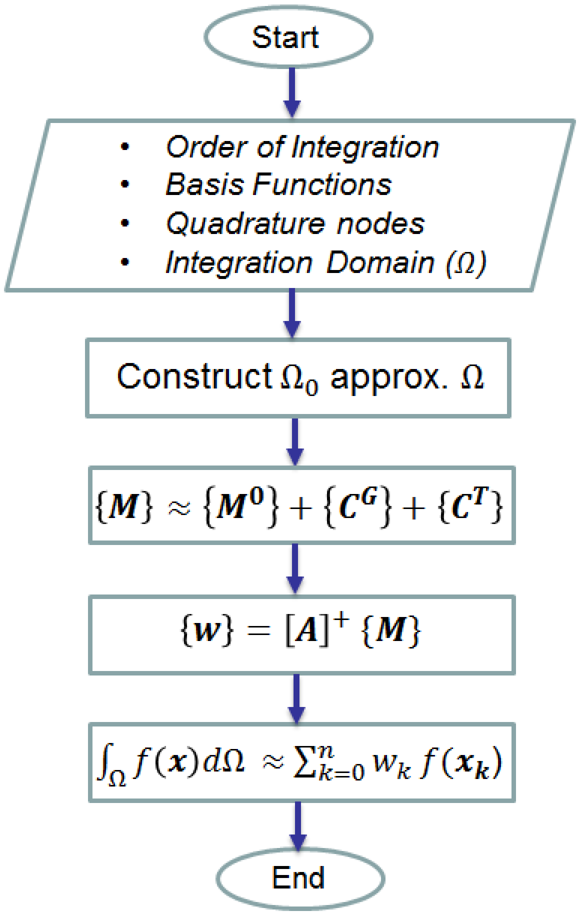

Start
Order of Integration
Basis Functions
Quadrature nodes
Integration Domain (Ω)
Construct Ω_0 approx. Ω
$\{M\} \approx \{M^0\} + \{C^G\} + \{C^T\}$
$\{w\} = [A]^+ \{M\}$
$\int_\Omega f(x)d\Omega \approx \sum_{k=0}^{n} w_k f(x_k)$
End

Figure 3.7: Flowchart for Shape Aware Quadratures

3.3 Sensitivity Analysis of Moments

M_i^G and M_i^T can in general be obtained using any order SSA [75], TSA [19], Feature Sensitivity (FSA) [123] or Modification Sensitivity [80]. However, in this thesis, we will limit ourselves to SSA and TSA. Nevertheless, it is important to note that the correction factors can be deduced using other types of sensitivities such as Feature Sensitivity (FSA) [123] or Modification Sensitivity [80] usually resulting in a boundary integral as given by Eq.(3.5).

One could write Eq.(3.7) in the following more general form

$$M_i \approx \int_{\Gamma_0} \beta_X^i(\mathbf{X})\,\mathrm{d}\Gamma_0 + \sum_{k=1}^{n_g} C_i^{G^k} + \sum_{l=1}^{n_t} C_i^{T^l} \tag{3.13}$$

$C_i^{G^k}$ is the k^{th} order geometric correction factor while $C_i^{T^l}$ is the l^{th} order topological correction factor. For the example domain shown in Fig. 3.5, if we consider up to second-order SSA for geometric corrections and first-order TSA for topological corrections then we have the following moment approximations :

$$\begin{aligned} M_i &\approx \int_{\Gamma_0} \beta_X^i N_X\, d\Gamma_0 + \sum_{k=1}^{2} C_i^{G^k} + \sum_{l=1}^{1} C_i^{T^l} \\ &= \int_{\Gamma_0} \beta_X^- N_X\, d\Gamma_0 + C_i^{G^1} + C_i^{G^2} + C_i^{T^1} \\ &= \int_{\Gamma_0} \beta_X^i N_X\, d\Gamma_0 + t\int_{\Gamma_0} b_i V_N d\Gamma_0 + \frac{t^2}{2}\int_{\Gamma_0} [\nabla b_i]^T \mathbf{N}\, V_N^2 d\Gamma_0 + \sum_{j=1}^{n_s} \mu(\omega_j)[-b_i(\mathbf{X_c^j})] \\ &= \int_{\Gamma_0} \beta_X^i N_X\, d\Gamma_0 + \int_{\Gamma_0} b_i V_N d\Gamma_0 + \frac{1}{2}\int_{\Gamma_0} [\nabla b_i]^T \mathbf{N}\, V_N^2 d\Gamma_0 - \sum_{j=1}^{n_s} \mu(\omega_j)[b_i(\mathbf{X_c^j})] \end{aligned} \tag{3.14}$$

Although, in principle, both SSA and TSA could be applied to compute C^G and C^T, it is more efficient to consider SSA for geometric corrections (C^G) and TSA for topological corrections. Hence, we will limit our discussion in this section to SSA for geometric corrections and TSA for topological corrections.

3.3.1 SSA of Moments

In general, the n_g^{th}-order geometric correction factor using SSA can be written as

$$C_i^G \approx tD^1_{S_i}(\Gamma_0, V_N) + \frac{t^2}{2!}D^2_{S_i}(\Gamma_0, V_N) + \frac{t^3}{3!}D^3_{S_i}(\Gamma_0, V_N) + ...\frac{t^{n_g}}{n_g!}D^{n_g}_{S_i}(\Gamma_0, V_N) \tag{3.15}$$

where $t \in [0,1]$ is the shape parameter and D^k_S is the k^{th}-order shape derivative of the i^{th} moment for the given approximate polygonal/polyhedral boundary Γ_0 and normal velocity $V_N(\mathbf{X})$. Table 3.1 lists the SSA of moments up to 2^{nd} order. In the table, $b_i(\mathbf{X})$ is the basis function corresponding to the i^{th} moment (M_i) and $V_N(\mathbf{X}) = \hat{\mathbf{V}}(\mathbf{X}).\mathbf{N}(\mathbf{X})$ is the normal component of the design velocity. It is important to note that the second-order shape sensitivity given in Table 3.1 is only valid as long as the vertices in 2D/edges in 3D of the approximating polygon/polyhedron Ω_0 lie on Ω as shown in Fig. 3.9. If this condition is not satisfied, then $D^2_{S_i}(\Gamma_0, V_N) = \int_{\Gamma_0} \left[[\nabla b_i]^T \mathbf{N} + \kappa b_i \right] V_N^2 \, d\Gamma_0$. Here, $\kappa(\mathbf{X})$ is the curvature in 2D and mean curvature in 3D for a given point $X \in \Gamma_0$ (for more details see Appendix A).

Table 3.1: SSA of Moments

k	$D^k_{S_i}(\Gamma_0, V_N)$
1	$\int_{\Gamma_0} b_i \, V_N \, d\Gamma_0$
2	$\int_{\Gamma_0} [\nabla b_i]^T \mathbf{N} \, V_N^2 \, d\Gamma_0$

In this thesis, we will always assume that the condition as stated above (i.e. all vertices/edges of Γ_0 lie on Γ) holds as otherwise the design velocity would become discontinuous and thereby violating the fundamental assumptions of SSA [75]. This condition can almost always be met in 2D by suitably constructing Γ_0 via point sampling or quadtree decomposition of Γ [70]. However, the same can't be said for arbitrary domains in 3D. Nevertheless, this error can be minimized by choosing a finer polyhedral approximation Γ_0 in 3D.

Further, for our purposes, we will assume that the design velocity vector $\hat{\mathbf{V}}(\mathbf{X})$ is piecewise continuous over every edge/face of Ω_0. Then, the perturbation of the domain by distance γ in the direction normal to the boundary is $\mathbf{x} - \mathbf{X} = \gamma \mathbf{N}(\mathbf{X})$, which implies that

$$V_N(\mathbf{X}) = V(\mathbf{X}).\mathbf{N}(\mathbf{X}) = t\hat{\mathbf{V}}(\mathbf{X}).\mathbf{N}(\mathbf{X}) = t\frac{\mathbf{x} - \mathbf{X}}{t}.\mathbf{N}(\mathbf{X}) = \gamma \mathbf{N}(\mathbf{X}).\mathbf{N}(\mathbf{X}) = \gamma \tag{3.16}$$

Thus, $V_N(\mathbf{X})$ is simply the shortest distance (γ) from $\mathbf{X}$ to the original boundary (Γ) as measured in the normal direction. For details of SSA of moments we refer the reader to Appendix A.

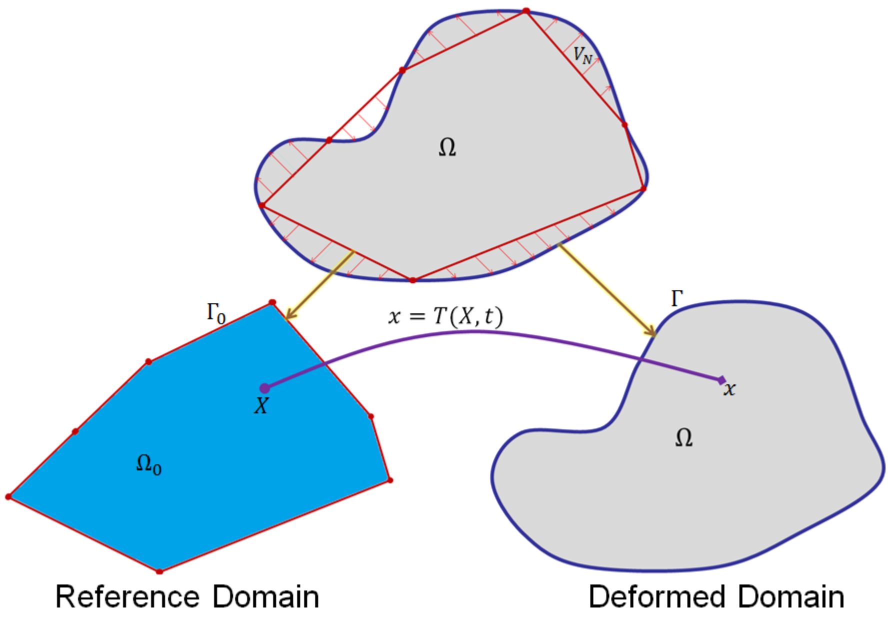


Figure 3.8: Reference and Deformed domains for SSA

Even for a coarse approximation Ω_0 of Ω one can get accurate results from SSA based correction factors as long as we include enough higher order shape sensitivities. For instance, for the set of trivariate basis functions up to order 1 i.e. $\{1, X, Y, Z\}$, the following second-order approximation can be proved to be exact even for a very coarse Ω_0 approximating Ω (as long as the boundary integrals are computed exactly)

$$\begin{aligned} C_i^G &= tD_S^1(\mathbf{X}_j) + \frac{t^2}{2!}D_S^2(\mathbf{X}_j) \\ &= \int_{\Gamma_0} b_i V_N d\Gamma_0 + \frac{1}{2}\int_{\Gamma_0} \nabla b_i^T \mathbf{N} V_N^2 d\Gamma_0 \end{aligned}$$

Thus, the second-order approximation of the moments over an arbitrary domain (using SSA for geometric corrections and assuming $C^T = 0$) can be written as a boundary integral over the approximate polygonal/polyhedral boundary Γ_0 as

$$\boxed{M_i \approx \int_{\Gamma_0} \left[\beta_X^i N_X + b_i V_N + \frac{1}{2}\nabla b_i^T \mathbf{N} V_N^2\right] d\Gamma_0 \qquad (3.17)}$$

where β_X^i is obtained by applying divergence theorem to $\int_{\Omega_0} b_i d\Omega_0$ as explained in section 2.1.1. The first-order approximation is obtained by neglecting the last term in the above equation

$$M_i \approx \int_{\Gamma_0} \left[\beta_X^i N_X + b_i V_N\right] d\Gamma_0 \tag{3.18}$$

For area/volume computations (i.e. $b_i(\mathbf{X}) = 1$), it is easy to verify geometrically (see Fig. 3.9) that the following equality holds

$$\begin{aligned} M_1 &= M_1^0 + C_G \\ &= M_1^0 + tD_S^1(\mathbf{X}_j) \\ &= \int_{\Omega_0} (1) d\Omega_0 + \int_{\Gamma_0} (1) V_N d\Gamma_0 \\ &= \int_{\Gamma_0} \left[X N_X + V_N\right] d\Gamma_0 \end{aligned} \tag{3.19}$$

In the above equation (and in Eq.(3.17)), the equality holds as long as the mapping $\mathbf{T}$ that we assume in Eq.(2.32) is a homeomorphism and the vertices (or edges in 3D) of Γ_0 lie exactly on Γ (for more details see Appendix A).

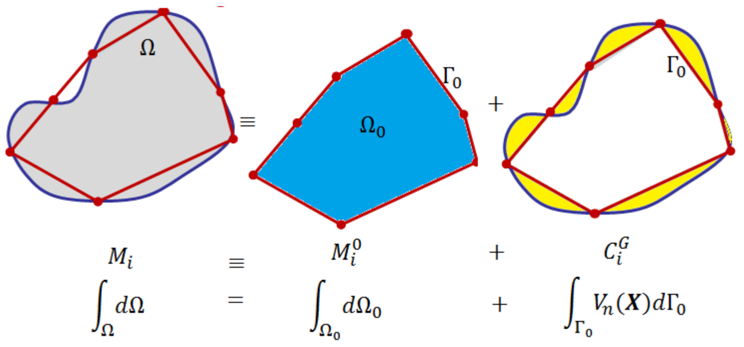


Figure 3.9: Geometric corrections based on first-order SSA is exact for computation of area (or volume) provided Γ_0 is homeomorphic to Γ and all the vertices (or edges) of Γ_0 lie on Γ

In this thesis, we only employ first/second-order SSA to compute C_G. Hence, for lower order correction factors the error in the approximation can't be eliminated totally when approximating higher order moments. Thus, it is important to choose Ω_0 to closely approximate Ω so as to minimize this error.

3.3.2 TSA of Moments

In general, the n_t^{th}-order topological correction factor (using TSA) for the i^{th} moment (M_i) over a ball of radius ϵ with center at $\mathbf{X}_c$ can be written as

$$C_i^T = f_1(\epsilon)D^1_{T_i}(\mathbf{X}_c) + f_2(\epsilon)D^2_{T_i}(\mathbf{X}_c) + f_3(\epsilon)D^3_{T_i}(\mathbf{X}_c) + ...f_{n_t}(\epsilon)D^{n_t}_{T_i}(\mathbf{X}_c) + \mathcal{R}(f_{n_t}(\epsilon)) \quad (3.20)$$

where $f_1, f_2, ..., f_{n_t}$ are monotonically decreasing functions of the size (ϵ) of the ball, $D^k_{T_i}$ is the k^{th}-order topological derivative of the i^{th} moment and $\mathcal{R}(f_{n_t}(\epsilon))$ is the remainder for this approximation. In general, the functions f_j should satisfy the following conditions

Table 3.2: TSA of Moments (for bivariate/trivariate polynomial basis)

	$D^1_T(\mathbf{X_c})$	$f_1(\epsilon)$	$D^2_T(\mathbf{X_c})$	$f_2(\epsilon)$
2D	$-X_c^pY_c^q$	$\pi\epsilon^2$	$\frac{-1}{2\pi}\left[\binom{p}{2}X_c^{p-2}Y_c^q + \binom{q}{2}X_c^pY_c^{q-2}\right]$	$\frac{1}{2}\pi^2\epsilon^4$
3D	$-X_c^pY_c^qZ_c^r$	$\frac{4}{3}\pi\epsilon^3$	$\frac{-1}{4\pi}\left[\binom{p}{2}X_c^{p-2}Y_c^qZ_c^r + \binom{q}{2}X_c^pY_c^{q-2}Z_c^r + \binom{r}{2}X_c^pY_c^qZ_c^{r-2}\right]$	$\frac{8}{9}\pi^2\epsilon^5$

$$\begin{aligned}
\lim_{\epsilon\to 0} f_j(\epsilon) &\to 0\ \forall j \in 1, 2, ...n_t \\
\lim_{\epsilon\to 0} \frac{f_j(\epsilon)}{f_{j-1}(\epsilon)} &\to 0\ \forall j \in 2, 3, ...n_t \\
\lim_{\epsilon\to 0} \frac{\mathcal{R}(f_j(\epsilon))}{f_j(\epsilon)} &\to 0\ \forall j \in 1, 2, ...n_t
\end{aligned} \quad (3.21)$$

The first ($D^1_{T_i}$) and second order ($D^2_{T_i}$) topological derivatives for moments (corresponding to bivariate/trivariate polynomials) along with the functions f_1 and f_2 are listed in Table. 3.2. Thus, in 2D, we have the following second-order topological correction for a circular hole of radius ϵ at (X_c, Y_c) with bivariate polynomial basis functions (X^pY^q)

$$\begin{aligned} C_i^T &\approx -\pi\epsilon^2[X_c^pY_c^q] - \frac{1}{4}\pi\epsilon^4\left[\binom{p}{2}X_c^{p-2}Y_c^q + \binom{q}{2}X_c^pY_c^{q-2}\right] \\ &= -\pi\epsilon^2\left[[X_c^pY_c^q] + \frac{\epsilon^2}{2}\frac{\left[\binom{p}{2}X_c^{p-2}Y_c^q + \binom{q}{2}X_c^pY_c^{q-2}\right]}{2}\right] \end{aligned} \tag{3.22}$$

Likewise, in 3D, we have the following second-order topological correction for a spherical hole of radius ϵ at (X_c, Y_c, Z_c) with trivariate polynomial basis functions $(X^pY^qZ^r)$

$$\begin{aligned} C_i^T &\approx -\frac{4}{3}\pi\epsilon^3[-X_c^pY_c^qZ_c^r] - \frac{2}{9}\pi\epsilon^5\left[\binom{p}{2}X_c^{p-2}Y_c^qZ_c^r + \binom{q}{2}X_c^pY_c^{q-2}Z_c^r + \binom{r}{2}X_c^pY_c^qZ_c^{r-2}\right] \\ &= -\frac{4}{3}\pi\epsilon^3\left[[X_c^pY_c^qZ_c^r] + \frac{\epsilon^2}{2}\frac{\left[\binom{p}{2}X_c^{p-2}Y_c^qZ_c^r + \binom{q}{2}X_c^pY_c^{q-2}Z_c^r + \binom{r}{2}X_c^pY_c^qZ_c^{r-2}\right]}{3}\right] \end{aligned} \tag{3.23}$$

For details of first and second order TSA of moments in 2D/3D we refer the reader to Appendix B. Now, from Eq.(3.22) and Eq.(3.23) one can deduce the following more general form for the second-order correction factor

$$C_i^T \approx -\mu(B_\epsilon)\left[b_i(\mathbf{X})\Big|_{\mathbf{X}=\mathbf{X_c}} + \frac{\epsilon^2}{2}\frac{\nabla^2(b_i(\mathbf{X}))}{2d}\Big|_{\mathbf{X}=\mathbf{X_c}}\right] \tag{3.24}$$

where μ and $\mathbf{X_c}$ are respectively the measure and centroid of the ball $B_\epsilon \subset \mathbb{R}^d$ of radius ϵ. Here, $d = 2$ or 3 is the dimension of the domain Ω. For a finite sized small negative/positive feature ω_l we define $\mathbf{X}_c^l$ as the centroid of the feature ω_l and S as the sign of the feature

$$S(\omega_l) = \begin{cases} +1 & \text{if } \omega_l \text{ is a positive feature,} \\ -1 & \text{if } \omega_l \text{ is a negative feature.} \end{cases}$$

A feature ω_l can be classified positive or negative using morphological operations as explained in section 3.1. Now, we can define an equivalent topological correction factor for arbitrary small negative/positive feature as

$$\boxed{C_i^T(\omega_l) = C_i^{T_l} \approx S(\omega_l)\mu(\omega_l)\left[b_i(\mathbf{X})\Big|_{\mathbf{X}=\mathbf{X_c^l}} + \frac{\epsilon_l^2}{2}\frac{\nabla^2(b_i(\mathbf{X}))}{2d}\Big|_{\mathbf{X}=\mathbf{X_c^l}}\right] \qquad (3.25)}$$

For most primitive shapes ω_l, μ can be computed using closed form expressions (for e.g. $\mu = \frac{1}{3}\pi r_c^2 h_c$ for a cylinder of radius r_c and height h_c). For more general small features, μ can be estimated accurately by applying divergence theorem (over the original domain ω_l) or via first-order SSA (over an approximate polygonal/polyhedral domain ω_l^0) as a boundary integral. For a collection of small features $\{\omega_l\}_{l=1}^{n^l}$, one can simply use Eq.(3.25) and sum the result to obtain the total topological correction C^T as

$$C_i^T = \sum_{l=1}^{n^l} C_i^{T_l} = \sum_{l=1}^{n_l} S(\omega_l)\mu(\omega_l)\left[b_i(\mathbf{X})\Big|_{\mathbf{X}=\mathbf{X_c^l}} + \frac{\epsilon_l^2}{2}\frac{\nabla^2(b_i(\mathbf{X}))}{2d}\Big|_{\mathbf{X}=\mathbf{X_c^l}}\right] \qquad (3.26)$$

Note that the above approach is perfectly valid as we are dealing with topological sensitivity of integrals (moments) that don't depend on an underlying field (such as displacements in elasticity) that gets perturbed due to material addition/removal. Hence, unlike topological sensitivity in boundary value problems, principle of superposition is perfectly valid in our case. Hence, topological corrections can be computed independently for each of the features and can be added together as in Eq.(3.26).

Usually, the user specifies a relative (small) feature size δ and hence the approximations given in Eq.(3.25) yields best results when the feature size $f_s(\omega_l) \leq \frac{\delta}{2}$ (as otherwise the feature is not small as per our definition in section 3.1). δ is usually problem dependent and can be estimated easily. For instance, in octree/quadtree based integration, one can define δ as a fraction of the smallest leaf cell size h i.e. $\delta = \alpha_f h$ where $0 < \alpha_f < 1$. There are many ways to define the size $f_s(\omega_l)$ of an arbitrary feature ω_l. A standard way is to use local feature size based on medial axis transform as in [94, 112]. However, defining the feature size this way could lead to poor TSA correction estimates for long thin features as in Fig. 3.2. Hence, in order to avoid such problems, we will simply define feature size as the radius of the smallest ball that contains the given feature ω_l.

In order to compute C^T, one needs the size ϵ_l of the feature ω_l in Eq.(3.25). One way to estimate this feature is to set $\epsilon_l \equiv f_s(\omega_l)$. However, an even better way is to conceptually replace the small feature with a ball B_{ϵ_l} centered at the centroid of the feature ($\mathbf{X^l_c}$) having the same measure as the small feature (as in [123]) i.e. $\mu(B_{\epsilon_l}) = \mu(\omega_l)$. Hence, the radius ϵ_l could be estimated as $\sqrt{\frac{\mu(\omega_l)}{\pi}}$ in 2D and $\sqrt[3]{\frac{3}{4\pi}\mu(\omega_l)}$ in 3D. This is consistent with the standard definition of TSA which considers the sensitivity of the quantity of interest w.r.t to the introduction of an infinitesimal "ball" in the domain. It is also important to note that TSA correction factors can be best applied for finite sized small features that are homeomorphic to a sphere (3D) / disc (2D). For features that are not homeomorphic to sphere (such as the torus), in principle, we could still apply TSA by breaking the small features into pieces that are homeomorphic to a ball. In this thesis, we will always assume that the small features are homeomorphic to a sphere in 3D and disc in 2D. In order to compute TSA based correction factors for thin features, it is best to break them into small pieces whose size $\epsilon_l < \frac{\delta}{2}$. Alternatively, for thin features it could be more efficient to employ first/second-order SSA to compute C^T.

3.4 Algorithm

The procedure follows essentially the same steps for 2D and 3D domains, and produces a set of integration nodes and weights that can be used to integrate any sufficiently smooth function over domain Ω. For the purpose of analysis, we assume that the shape is bounded by a degree k curve/surface and that k is known *a priori* (or at least can be estimated *a posteriori*). Conceptually, the procedure consists of the following three subtasks that we describe below in detail.

Initialization involves setting the data structures required for solving the system of moment equations (Eq.(3.3)). Note that the dimensions of the matrix $[\mathbf{A}^{\dagger}]$ depend on the choice of basis functions and on the number of integration nodes, both of which depend on the required order of integration. Specifically, initialization consists of the following steps:

1. Choose a piecewise linear domain Ω_0 (polygon in 2D or polyhedron in 3D) that is a reasonable approximation of the original domain Ω with or without the small features. There are many ways to accomplish this and we will see some examples in section 3.6.

2. Determine the order of integration (o) based on the function to be integrated. Choose a set of basis functions $\{b_i\}_{i=1}^{m}$ of order up to o. A simple choice is the trivariate polynomials $x^p y^r z^s$ (3D) or bivariate polynomials $x^p y^r$ (2D).

3. For the chosen order of integration (o), determine the number and location of quadrature nodes. The minimum number of quadrature points required is dictated by the solvability of moment fitting equations. By having as many equations (basis functions) as the number of unknowns (weights), we make matrix $[\mathbf{A}]$ invertible and thus eliminate the need for least squares solution. Hence, the minimum number of quadrature points required is equal to the number of basis functions (m) chosen in the previous step. However, in general, choosing more quadrature points (n) than this minimum number would result in a rectangular system that when solved in a least square sense results in $n - m$ quadrature points with zero weights. Moment fitting equations also offer a lot of flexibility in positioning the quadrature points. There are a number of ways to choose the quadrature points. One way is to use the tensor-product rule ensuring all points are inside the domain as explained in [22] (for e.g., see Fig. 2.2a). Alternatively, quadrature points could be generated by randomly sampling points inside the domain or over the boundary as we will demonstrate in subsections 3.6.4, 3.6.5, and 3.6.6. In addition, it is best to ensure that all the quadrature nodes lie within the approximate polygon/polyhedron (Ω_0) as choosing points outside Ω_0 will lead to dealing with

the discontinuity of an otherwise continuous integrand (as in characteristic function approach) and thereby deteriorating SAQ's accuracy.

Compute Moments The left hand side of equation (3.2) is a vector of m moments, one moment for each basis function. The SAQ evaluation procedure follows the development in section 3.2 and section 3.3, namely:

4. Determine the quadrature nodes and weights for each edge (2D) or face (3D) of the simplified domain Ω_0 in order to evaluate the moment approximations in Eq.(3.13). The correction factors in Eq.(3.13) can be obtained by using appropriate sensitivities from Table 3.1 and Table 3.2. It is important to note that these correction factors are usually a boundary integral of the form given by Eq.(3.5).

   SSA Correction factor The quadrature nodes and weights required to evaluate the SSA correction factors are not easy to determine accurately as SSA correction terms usually produces non-polynomial integrands (as we will see in one of the examples in chapter 5). However, one can arrive at a good estimate by looking at the curve/surface that the approximate edge/face approximates. If the edge/face approximates a surface of degree $2k-1$ in the original boundary Γ, choosing the integration order to be $(o+k)$ (for example, $k=2$ for conic sections and quadric surfaces) gives reasonably accurate results for SSA correction factors. A better method is to use well-known integrand adaptivity techniques for lines/triangles such as the one in [98, 32] (or using MATLAB's [90] *int* and *quad2d* functions) although it is computationally expensive.

   TSA Correction factor For TSA correction factors, the sensitivities are relatively easy to compute as it only involves the computation of the measure/size of the feature and sampling of the basis function (and/or its derivatives) at discrete points (see subsection 3.3.2). For simple features such as a circle (or sphere) and ellipse (or ellipsoid) there are closed form expressions available for measure computations. For more general features, we use divergence theorem or first-order SSA to compute the measure ($\mu(\omega_l)$) approximately as a boundary integral. Thus, in this more general case, we choose an integration order of $k+1$ to compute the boundary integrals arising in TSA corrections. Appropriate sign for the TSA correction factor can be determined by knowing if the feature is positive or negative. This can be easily accomplished in general by using the morphological opening and closing operations as discussed in section 3.1. As already discussed in subsection 3.3.2, the size ϵ_l can be estimated as $\sqrt{\frac{\mu(\omega_l)}{\pi}}$ in 2D and $\sqrt[3]{\frac{3}{4\pi}\mu(\omega_l)}$ in 3D. If $\epsilon_l > \frac{\delta}{2}$, then it is best to break the small features into more manageable pieces for TSA computations. Alternatively, for such features, C^T could be computed using SSA without breaking it into smaller pieces.

5. Using the basis functions selected in step 1, evaluate the approximation to $\{\mathbf{M}\}$ over the original domain Ω from Eq.(3.13), by summing contribution over individual edges (2D) or faces (3D) of the approximate domain Ω_0 (using the quadrature nodes and weights determined in step 4).

Solve for weights and evaluate

6. Compute $[\mathbf{A}^{\dagger}]$ for the preselected set of basis functions (step 1) and quadrature nodes (step 2) using say QR factorization [82].

7. Using the $\{\mathbf{M}\}$ from Step 5 and $[\mathbf{A}^{\dagger}]$ from step 6, compute the approximate weights for the arbitrary domain Ω from $\{\mathbf{W}\} = [\mathbf{A}^{\dagger}]\{\mathbf{M}\}$ (Eq.(3.3)).

8. Finally, evaluate the approximation to the required integral by using these weights in Eq.(2.6) i.e. $\int_{\Omega} f(\mathbf{x}) \mathrm{d}\Omega \approx \sum_{k=1}^{n} W_k f(\mathbf{x}_k)$.

It should be clear that the same algorithm may be used in most situations, with steps 4 and 5 depending strongly on type, dimension, and representations of the geometric domain

Ω and its approximation Ω_0. Note that these two steps also dominate the computational complexity of the proposed approach, as we discuss below.

3.5 Algorithm Analysis

The analysis of the algorithm is dependent upon the order and type of sensitivity analysis used for the computation of correction terms in Eq.(3.7). For the purpose of this analysis, we will assume that we use first-order SSA for geometric corrections (C_i^G) and first-order TSA for topological correction (C_i^T) of n_s small features. In other words, we will assume that the moment approximations are given by Eq.(3.9).

Thus, the main operations that are needed to implement the SAQ procedure are construction of simplified approximation of Ω_0 (Step 4), distance to boundary computation (for design velocity computation), normal computations for polygonal edges/faces (for computing normal component of design velocity), measure computations (for TSA) and ability to sample points/basis functions on a 2D/3D domain/boundary (for quadrature node generation and topological corrections using TSA). All these operations are supported by any standard CAD software and hence are readily available.

For a more careful asymptotic analysis, let $2k-1$ be the degree of the domain boundary (Γ), o be the order of integration, n_* be the number of edges/faces in the approximating polygon/polyhedron Ω_0, and $m(o) = O(o^d)$ be the number of d-variate polynomial basis functions in d-dimensional space. We will assume that the cost of all arithmetic operations is constant. With this, we can estimate the running time of the algorithm. The initialization steps 1-2 are performed only once and do not affect the worst-case running time; step 3 is a construction of piecewise linear domain Ω_0, a task that is well-understood and can be done in a number of ways. We note that such approximations are often used by many CAD systems and may be available at no additional cost. Steps 6-8 are dominated by computation of $[\mathbf{A}^\dagger]$ (step 6) and matrix multiplication (step 7). The cost of computing $[\mathbf{A}^\dagger]$ using QR factorization can easily be proved to be $O(o^{3d})$ [82]. Also, the matrix multiplication costs $O(o^{3d})$. This leaves steps 4 and 5 that are at the very heart of the SAQ.

Typically, the simplified domain Ω_0 is bounded by triangles or convex quadrilaterals, step 4 is a trivial constant time operation. Close examination of Eq.(3.9) reveals that evaluation of moments in step 5 requires:

- $O(n_*)$ normal computations;
- $O(n_s)$ measure computations;
- $O(n_*(o+k)^{d-1})$ distance computations;
- $O(n_*(o+k)^{d-1}o^d + n_s)$ multiplications/additions; and
- $O(n_*(o+k)^{d-1}o^d + n_s)$ basis function evaluations.

Assuming that all of these operations are constant, the running time of step 5 is $O(n_*(o+k)^{d-1}o^d + n_s)$. Generally speaking, computing distance between two arbitrary domains is not a constant time operation; however, when correspondence between the faces in Γ and Γ_0 is known, the distance computations are localized, and may be considered constant time for any practical purpose. Likewise, computing the measure (volume/area) of a small feature is not a constant. However, in most cases there are closed form expressions available for measure computations. For more general small features, one could employ divergence theorem or first-order SSA to efficiently compute the measure of small features.

We conclude that in most practical situations, the total cost of SAQ based integration is $O(o^{3d} + n_*(o+k)^{d-1}o^d + n_s)$. For a given order of integration o, degree of bounding surfaces k, dimension of space d and number of small features n_s, the procedure is linear in the number of bounding faces/edges n_* of the simplified domain Ω_0. This conclusion underscores one of the most appealing properties of the SAQ procedure: it does not assume any particular representation of the original domain Ω (such as voxels, mesh, splines, and implicit surfaces), and can be used with any such representation, as long as it supports efficient distance computations and can be approximated by a polygonal domain Ω_0. In other words, k can be viewed as a measure of frequency of oscillations in the boundary of Γ relative to the simplified boundary Γ_0. Large k leads to the increase in the number of required integration points and distance computations which could undermine the advantages of SAQ procedure. On the other hand, SAQ procedure can be also used with arbitrarily complex approximate domains Ω_0. When a 3D Ω_0 is bounded by more general polygonal faces, such as those found in [133, 89, 74], step 4 of the procedure requires constructing a quadrature for each face using the moment fitting equations as described in Section 2.1.1, with moments computed from Eq.(2.5). In principle, this could increase the theoretical complexity of the algorithm to $O(n_*(o+k)^{3(d-1)}+n_s)$ but could also reduce the overall cost of integration by allowing a much larger class of approximate domains in SAQ based integration.

3.6 Experimental Validation

In this section, in the first three examples, we compare the computational properties of SAQ to five other methods and a reference symbolic integration. The five selected methods are:

SS: a direct application of shape sensitivity analysis as explained in Section 2.1.4;

C: the Cartesian product or tensor-product rule scaled to fit inside the geometry of the cell as recommended in [132];

GA: the Geometrically Adaptive integration method described in [22];

P: the quadrature rule obtained by solving the linear moment fitting equations (in least square sense) over a polygonized boundary (without correction).

Characteristic: the characteristic function approach as explained in section 2.1.3

In examples 1,2,3, and 5 we assume that Ω_0 is a polygonal/polyhedral domain homeomorphic to the given domain Ω. Thus, in these four examples, $C_i^G = 0$ and thus $C_i^S = C_i^G$. In example 4, we demonstrate the application of SAQ in integrating a cubic polynomial over a non-convex 2D domain in the presence of numerous small features (using various type of correction factors). Thus, for example 4 and 6, $C_i^T \neq 0$ i.e. $C_i^S = C_i^G + C_i^T$. For all the examples, we use first-order SSA for C_i^G computations. For example 4, we compute and compare C_i^T using first-order SSA, first-order TSA and zero-order SSA (or divergence theorem). For example 6, we only compute and compare C_i^T using first/second-order TSA.

All algorithms were implemented in MATLAB 7.10 (on a Intel Core i7 CPU with 2.8 GHz speed and 8 GB memory), which was also used for symbolic integration, and tested on four 2D domains and two 3D domains : quadrant of a circle in Section 3.6.1, notched domain in subsection 3.6.2, wavy domain in subsection 3.6.3, non-convex 2D domain with small circular holes in subsection 3.6.4, unit sphere in subsection 3.6.5, and unit sphere with numerous spherical voids in 3.6.6. We will denote the value of integral obtained from MATLAB's symbolic integrator as I_A and use I_* to stand for the value of integral computed by method (*), where (*) could be one of the above methods. With this notation, we will compare the accuracy of integration for each domain in terms of the following error measures:

- Relative Error (%) $= |\frac{I_A - I_*}{I_A}| * 100$;
- Actual Error$= I_A - I_C$.

For the first three examples, we will perform a series of tests aimed to establish how the accuracy of the algorithms compare in terms of their ability to approximate integrals of polynomial basis functions and integrands, adapt with better polygonal approximation, and improve with increased order of integration. Specifically, for each domain Ω, we perform the following tests:

(a) Integration of Basis Functions The integrand for this test would be the bivariate polynomial basis functions of order up to 5. The approximating polygon Ω_0 will have 7 edges (i.e. $n_e = 7$), and 3-pt tensor-product rule (i.e. $n = 3 \times 3$) will be used to generate the quadrature nodes.

(b) Effect of polygonization Integral of $f = x^2y^2 + x^2y^3 + x^3 + 100x + 10y + 2$ with 3-pt tensor-product quadrature rule (i.e. $n = 3 \times 3$) is evaluated, as the number of edges in the approximating polygon Ω_0 is varied. It is worth noting that increasing the number of edges doesn't always improve the polygonal approximation to Ω as we will see in one of the examples (Example 3). In other words, increasing the number of edges (finer polygonization) doesn't necessarily reduce the distance between Ω and Ω_0 (in Hausdorff sense).

(c) Dependence on quadrature rule Integral of $f = x^2y^2 + x^2y^3 + x^3 + 100x + 10y + 2$ is evaluated over polygonal approximation with seven edges (i.e. $n_e = 7$), while the quadrature rule is varied.

For the third example in section 3.6.3, which is based on quadtree decomposition, we perform one additional experiment showing how the accuracy of the three methods (SAQ, SS, GA) improves with increased depth of the approximating quadtree, while fixing $n_e = 7$ and $n = 3 \times 3$ in the leaf cells for the integrand f as defined above.

The approximating polygon (Ω_0) could be constructed in a number of ways including (i) ray casting [119], (ii) coarse quadtree/octree decomposition [70] of Ω, and (iii) coarse surface meshing [97] of Ω. For the first three examples, we employed method (i) to construct Ω_0 homeomorphic to Ω (i.e. $C_i^T = 0$). Specifically, we casted rays parallel to integration rays and intersected it with the original domain Ω to generate the polygon vertices (see Fig. 3.10). For example four, we will assume that Ω is only an approximation to Ω_0 without considering the numerous small features i.e. Ω_0 is not homeomorphic to Ω and so $C_i^T \neq 0$. For this example, we construct Ω_0 simply by the quadtree/octree decomposition of Ω (ignoring the small features). In example 5, we illustrate the application of SAQ over a unit sphere using an octree based integration. Here, we construct Ω_0 homeomorphic to Ω (using the octree decomposition of the domain) and hence $C_i^T = 0$ for this example. In example 6, we give an application of SAQ in integrating a cubic function over a sphere with several thousand spherical voids. Here, we construct Ω_0 homeomorphic to $\Omega - \bigcup_{f=1}^{n_s} \omega_f$ (using the octree decomposition of the domain). However, we compute C_T using first/second-order TSA in order to account for the presence of voids (ω_f).

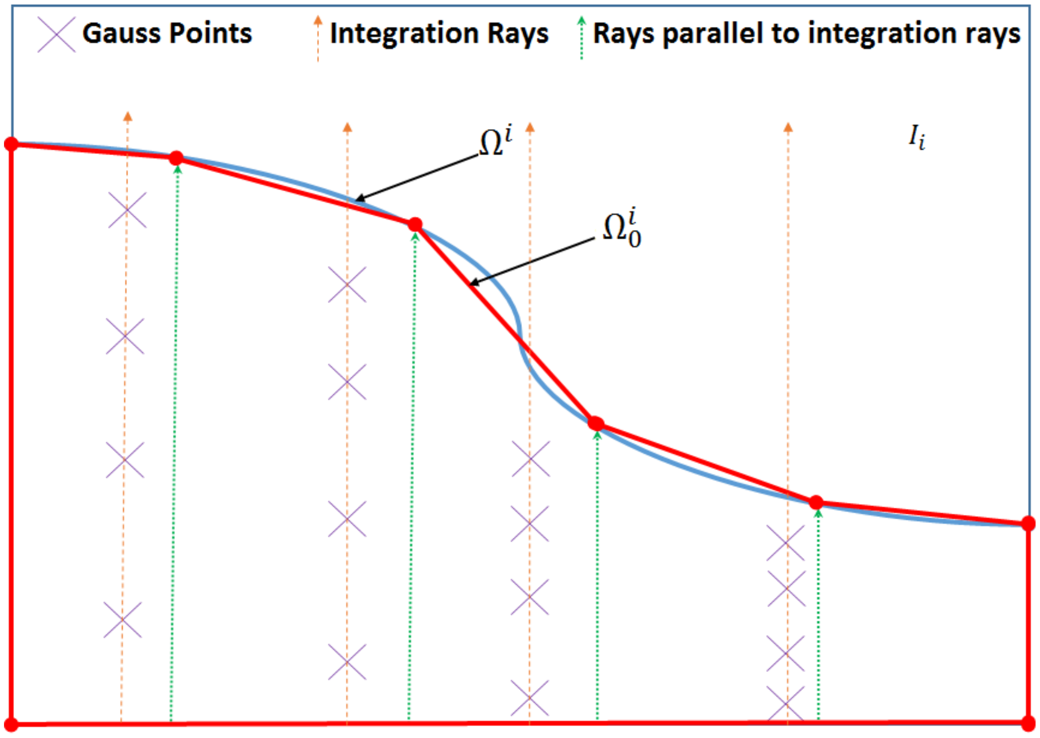


Figure 3.10: Approximate polygon (Ω_0) construction by ray casting. The approximate polygon vertices are generated by casting rays parallel to integration rays and intersecting it with the original domain (Ω).

3.6.1 Example 1 - Quadrant of a circle

We will consider the domain Ω to be a quadrant of a circle (Fig. 3.11a) with the approximating polygon Ω_0 as shown in Fig. 3.11b. This example would demonstrate that, in an average sense, SAQ is at least as good or better than SS, C and P methods.

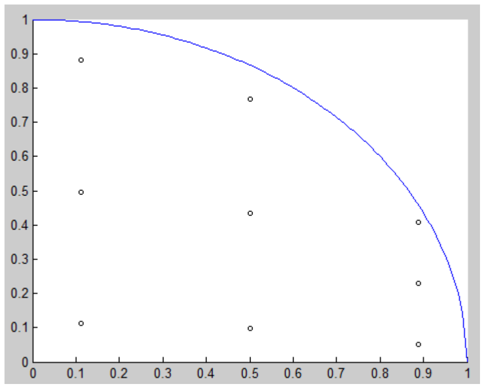
a. Integration of Basis Functions The results are shown in Fig. 3.11c. The relative errors of SAQ, SS, and C are in the range $0.012\%-1.718\%$, $0.012\%-1.821\%$, and $0\%-8.270\%$ respectively. The absolute value of the integrals and the relative errors are listed in Table I

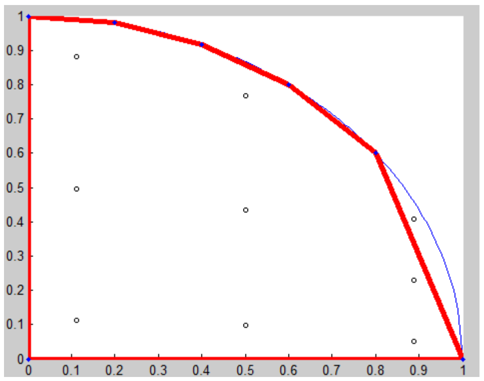
(Appendix B). Clearly, SAQ and SS are better than C in most cases, except for some lower order basis functions. Note that SAQ and SS correlate well with each other.

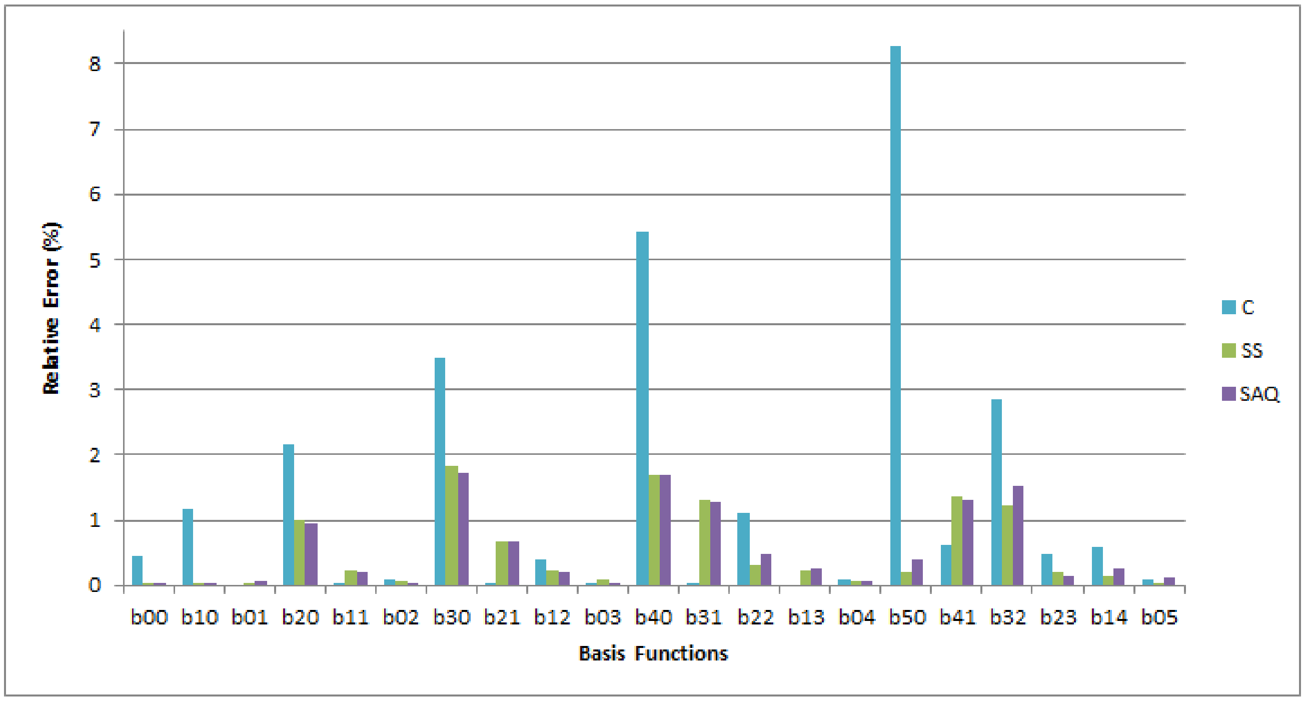
b. Effect of polygonization The results are shown in Fig. 3.11d. We observe a significant decrease in error for the P method, as expected. However, the results also indicate that for the P method to be as accurate as the SAQ or SS methods, it needs a finer polygonal approximation. We also note that the error in SAQ saturates for very fine polygonal approximations. This is not an inherent problem of the algorithm as we will see in chapter 5. This is primarily due to the error arising in moment approximation computations due to approximate distance computations. For this example, we used a rather coarse polygonal approximation of the original domain for the normal velocity (via ray casting from Γ_0 to Γ). We will revisit this example in subsection 5.3.1 where we will demonstrate quadratic convergence of the method (for this example) w.r.t design velocity using exact distance function.

c. Dependence on quadrature rule The results are shown in Fig. 3.11e. It is clear from Fig. 3.11e that only for '6-pt' quadrature rule, the error in C is smaller than that of SAQ.

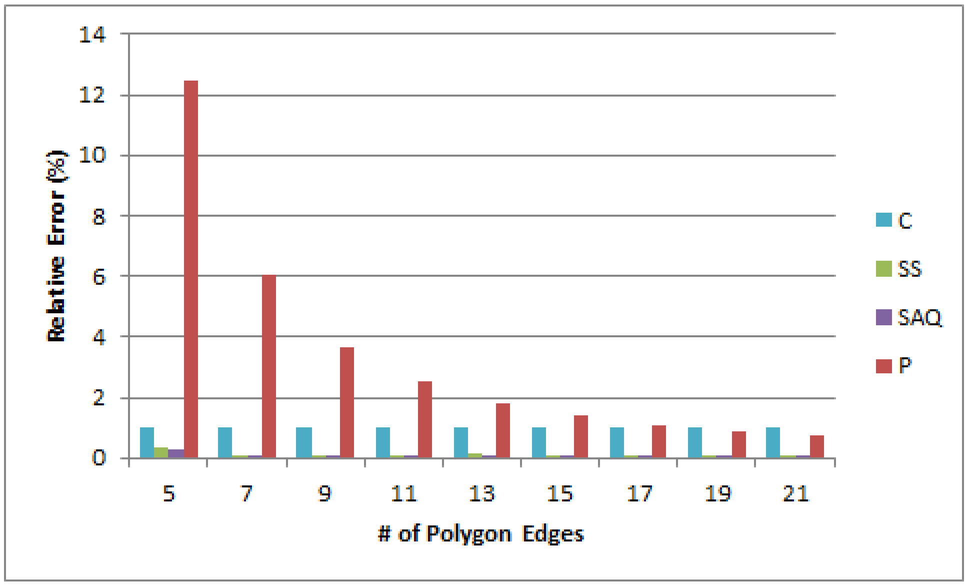
Note that in this example, one of the quadrature nodes lie outside the approximating polygon Ω_0. The presence of nodes outside the polygon usually decreases the accuracy of the integral computed. Hence, whenever possible, it is best to choose a polygonal approximation that contains all the quadrature nodes well within the polygonal boundary. In this particular example, the presence of outside node for 3-pt rule does not have a serious impact on the results computed by SAQ, as the quadrature weight corresponding to the node lying outside the polygon is close to zero. However, for higher order quadrature (4-pt and higher), outside nodes do have an effect on the results as can be seen from Fig. 3.11e. However, the error (in SAQ and SS) stabilizes and becomes a constant with increase in quadrature rule. This is because, unlike the C method, increasing the quadrature rule beyond a limit doesn't usually improve the accuracy of SAQ. This is consistent with the fact that SAQ doesn't rely on the distribution of quadrature points to capture the geometry of the integration domain.


(a)

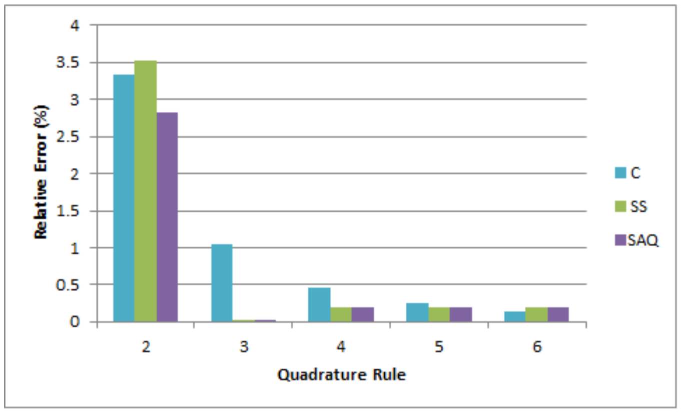

(b)



(c)



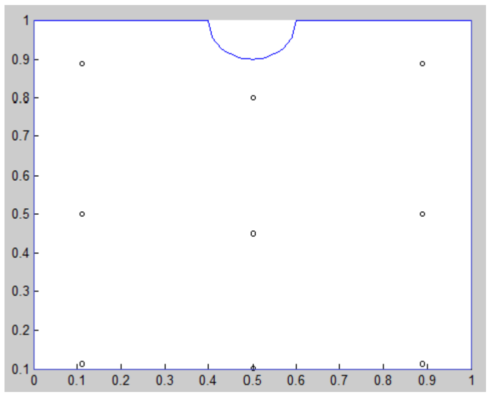
(d)



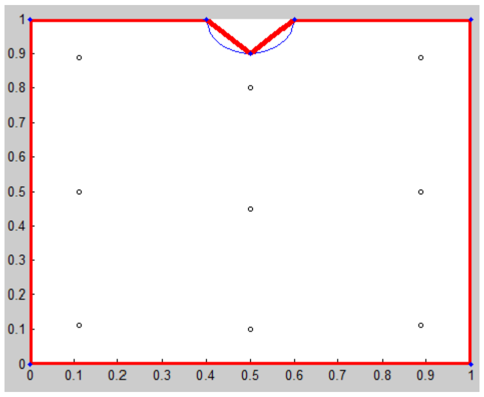
(e)

Figure 3.11: (a) Quadrant of a circle (Ω) (b) Approximating polygon (Ω_0) (c) Rel. error vs basis functions (for $bij = x^i y^j$, $n = 3 \times 3$, $n_e = 7$) (d) Rel. error vs no. of polygon edges (for f, $n = 3 \times 3$) and (e) Rel. error vs quadrature rule (for f, $n_e = 7$)

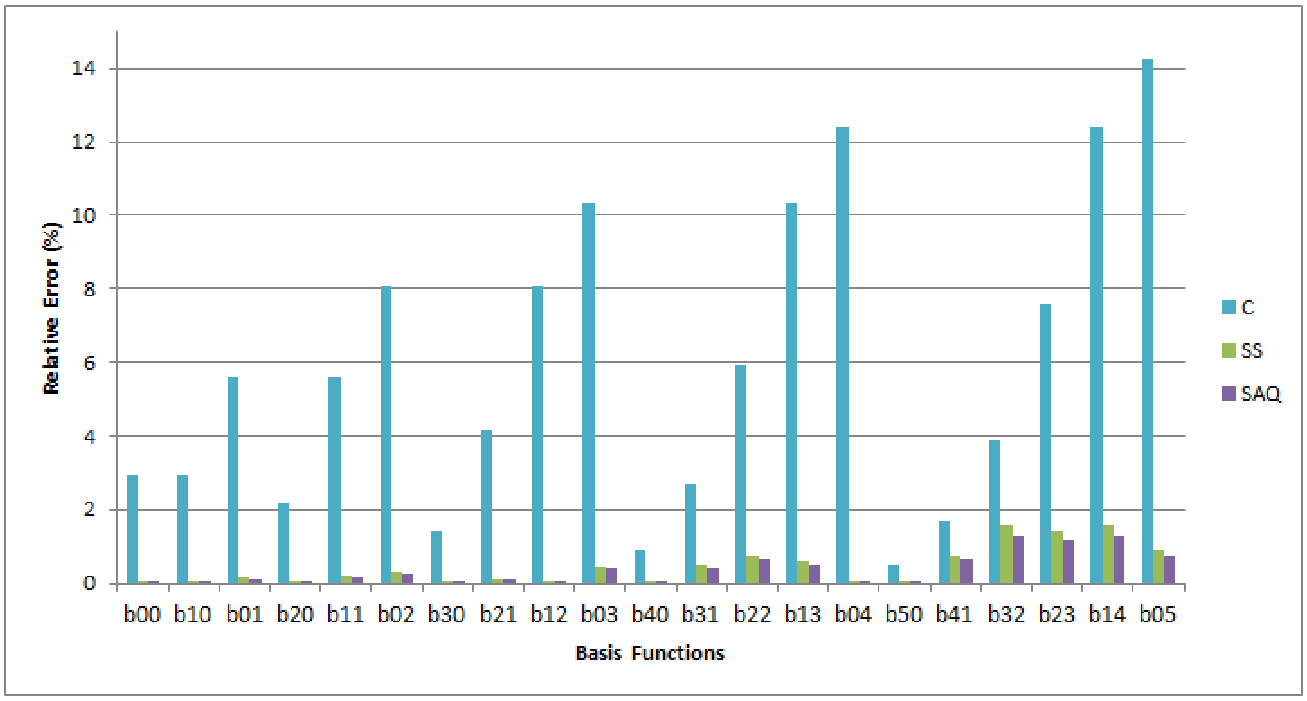
3.6.2 Example 2 - Notched Domain



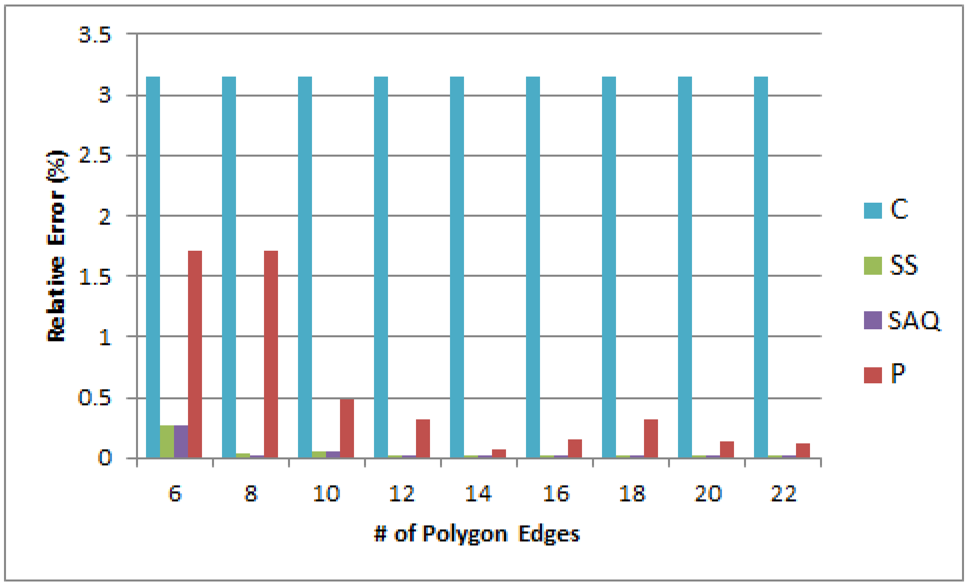
(a)



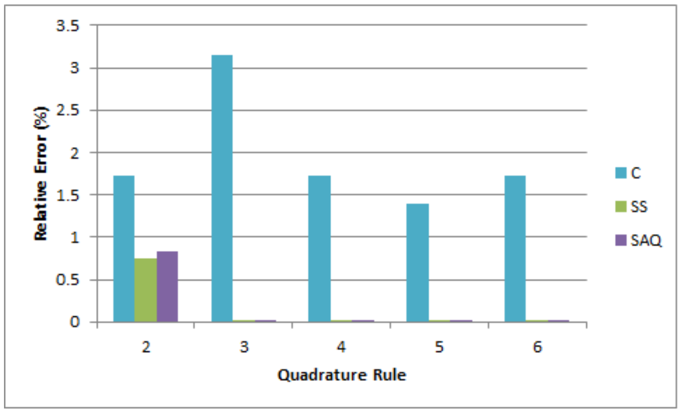
(b)



(c)



(d)



(e)

Figure 3.12: (a) Notched domain (Ω) (b) Approximating polygon (Ω_0) (c) Rel. error vs basis functions (for $bij = x^i y^j$, $n = 3 \times 3$, $n_e = 7$) (d) Rel. error vs no. of polygon edges (for f, $n = 3 \times 3$) and (e) Rel. error vs quadrature rule (for f, $n_e = 7$)

We will consider the notched domain Ω (Fig. 3.12a) with the approximating polygon Ω_0 as shown in Fig. 3.12b. This example demonstrates that SAQ produces accurate results where C fails in spite of using a higher-order quadrature.

a. Integration of Basis Functions The results are shown in Fig. 3.12c. The relative errors of SAQ, SS, and C are in the range $0.010\% - 1.301\%$, $0.008\% - 1.550\%$, and $0.510\% - 14.271\%$ respectively. The absolute value of the integrals and the relative errors are listed in Table II (Appendix B). Clearly, SAQ and SS are better than C for all basis functions as the C method doesn't detect the notch. As before, SAQ and SS are found to correlate well with each other.

b. Effect of polygonization The results are shown in Fig. 3.12d. Again, there is a significant decrease in error for the P method as expected. And, once again, for the P method to be as accurate as SAQ or SS methods, it needs to have a finer polygonal approximation.

c. Dependence on quadrature rule The results are shown in Fig. 3.12e and demonstrate that C cannot compete with SAQ and SS even for 6-pt quadrature because it fails to detect the presence of the notch.

3.6.3 Example 3 - Wavy domain

In this example, we will demonstrate how SAQ can be used in a quadtree based integration. For comparison purposes, SAQ, SS and GA methods are applied to integration over the leaf cells of the quadtree that decomposes the wavy domain[3](Fig. 3.14a). The approximating polygons and the quadrature nodes were constructed in the leaf cells following the basic rules of the GA method [22], by accounting for the type of leaf cell that arises in marching squares (see Fig. 3.13). The interior cells (case 15) were integrated using the regular Cartesian product rule while the exterior cells (case 0) were ignored. The quadrature nodes for the leaf cells were allocated based on Cartesian coordinates (with appropriate scaling) for cases 3,6,7,9,11,12, 13 & 14 and polar coordinates for cases 1,2,4 & 8. The ambiguous cases (case 5 and 10) were handled simply by recursively dividing the cell until the ambiguity is resolved.

[3]the wavy surface is defined by the equation $y(x) = 3.2 + 0.09sin(10.5\pi x)$

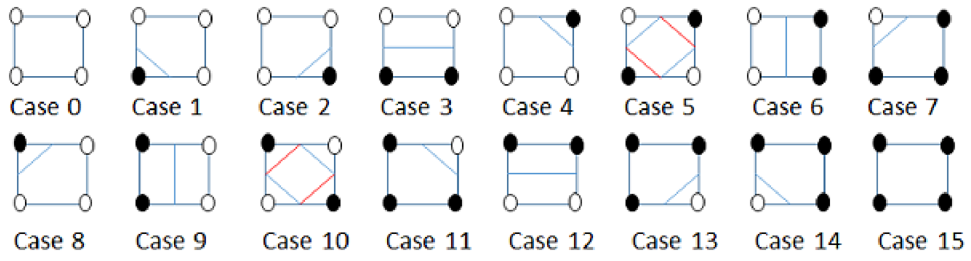


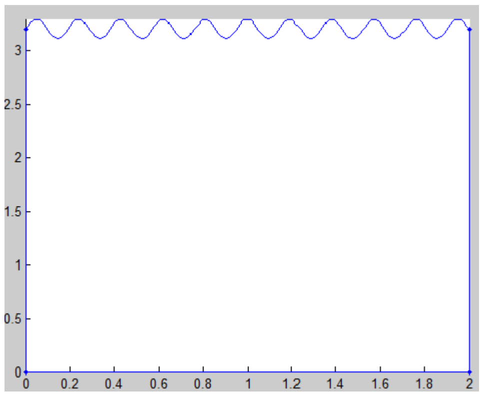
Figure 3.13: Cases in marching squares

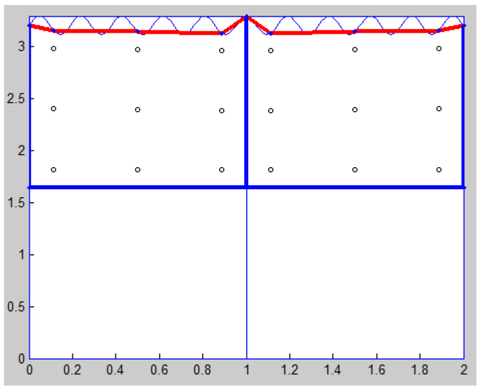
a. Integration of Basis Functions The results are shown in Fig. 3.14c. The quadtree of depth one (i.e. D = 1) along with the approximating polygons ($n_e = 7$) used in the leaf cells are shown in Fig. 3.14b.The absolute value of the integrals and the relative errors are listed in Table III (Appendix C). The relative errors of SAQ, SS and GA are in the range 0.004% – 0.557%, 0.009% – 0.557% and 1.975% – 12.180% respectively. It is apparent that SAQ and SS are more accurate than GA for all integrands. As in previous examples, SAQ and SS correlate well with each other.

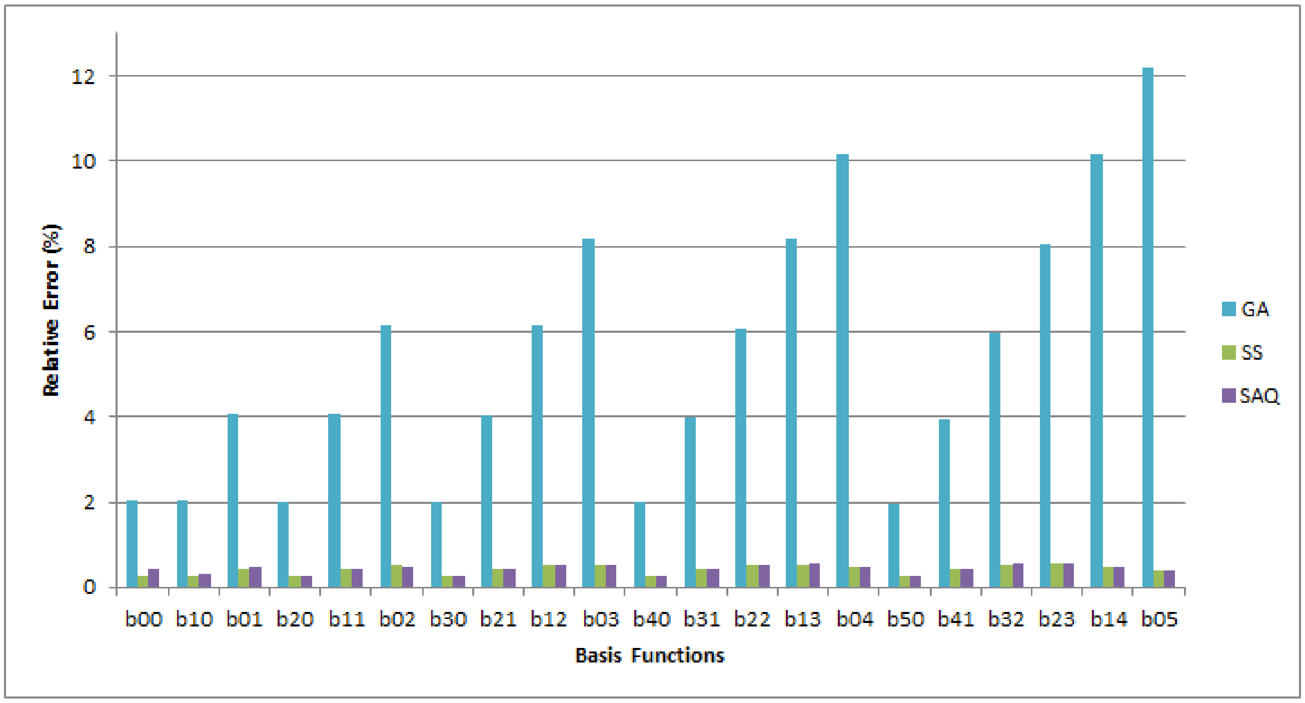
b. Effect of polygonization The number of edges in polygons approximating the leaf cells is varied, while all other parameters remain the same: D = 1, $n = 3 \times 3$ (Fig. (3.15a-3.15d) and the integrand f over the wavy domain. For this test only, we also include integration results with P method applied to the leaf nodes of the quadtree. The relative errors for SAQ, SS, P and GA methods are in the range 0.003% – 1.569% ,0.005% – 1.424%,0.05% – 2.455% and 2.91% – 2.91% respectively. However, GA method is not adapting to finer polygonization as one would expect, while P method is improving but somewhat inconsistently. This is because, in this particular case, finer polygonization doesn't necessarily mean better approximation of the original domain Ω. In other words, increasing the number of polygon edges doesn't necessarily reduce the distance between Ω_0 and Ω (in Hausdorff sense). This also explains why the error in SAQ oscillates with finer polygonization in contrast to the previous two examples (Fig. 3.11d and Fig. 3.12d). The number of edges required for the relative error to fall below 0.1% are 9 for SAQ, 9 for SS and 14 for P method. This suggests that P method requires finer and better polygonization to produce accurate integrals over domains with rapidly varying boundaries and likely to be more expensive than SAQ and SS methods.

c. Dependence on quadrature rule Number of quadrature nodes are varied in the leaf cells with D = 1 and $n_e = 7$ (Fig. 3.16a - 3.16c). Again, even with '6-pt' rule in the leaf cells, the error in GA could not be made smaller than that of SS and SAQ.

d. Accuracy vs Depth of Quadtree Depth of the quadtree is varied with $n_e = 7$ and $n = 3 \times 3$ (Fig. 3.17a - 3.17d). From the results, it is clear that SAQ and SS are able to produce significantly more accurate results even at coarser resolutions (say D = 1 or 2) when compared to GA method. In fact, from Table I it is clear that for the relative error to fall below 0.05%, SAQ (and SS) requires 343 ms as against GA's 562 ms. Also, we find that, in general, the error decreases with refinement for all the 3 methods. However, the order of convergence varies. This is explained by the fact that the order of accuracy depends not only on the level of refinement but also on the kind of polygonal approximation (or design velocity) at any given resolution. Since, in this example, the degree of polygonal approximation varies non-uniformly from one level to another the design velocity does not decrease monotonically w.r.t. mesh refinement (see chapter 5 for more details). This explains why SAQ doesn't exhibit a consistent order of convergence for this example.

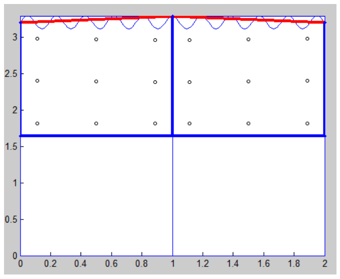

(a)

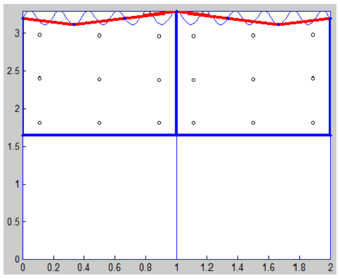

(b)

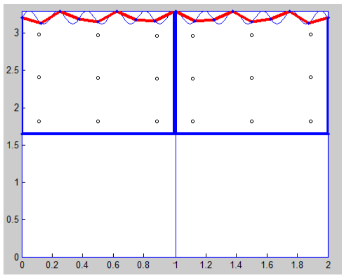


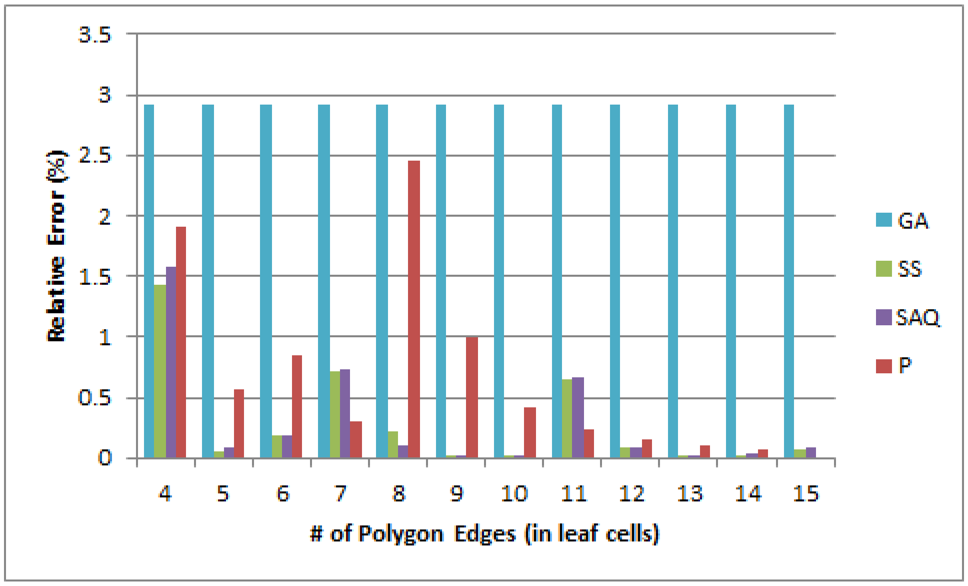
(c)

Figure 3.14: Quadtree integration of bivariate polynomial basis functions ($bij = x^i y^j$) of order up to five over the wavy domain for $n_e = 7$ and $n = 3 \times 3$. (a) Wavy domain (Ω) (b) Quadtree with D = 1 and (c) Rel. error vs basis functions


(a)

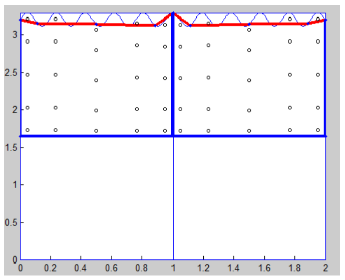

(b)


(c)

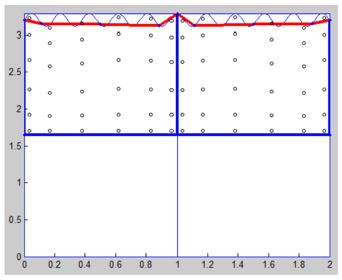


(d)

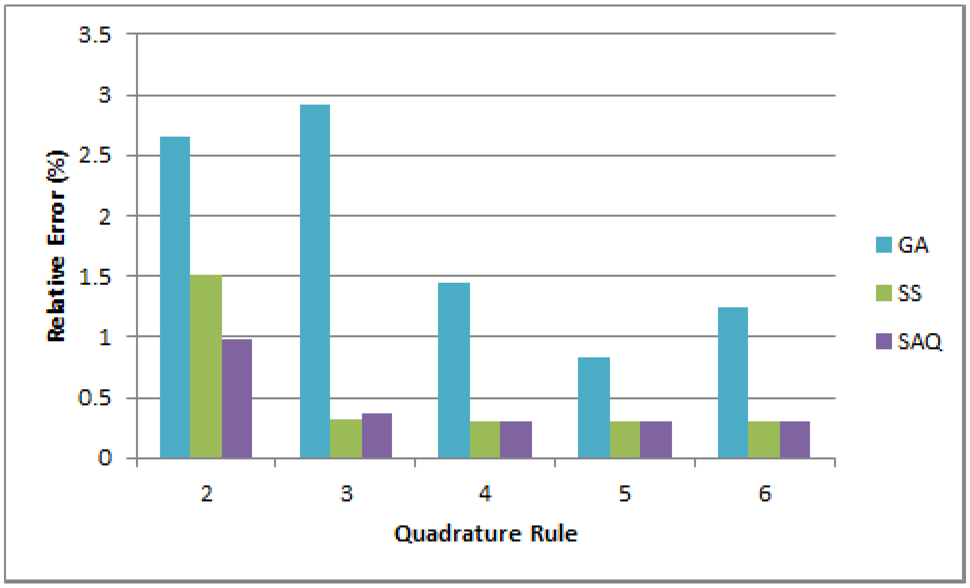
Figure 3.15: Quadtree integration of f over the wavy domain for various polygonal approximations of the leaf cells with D = 1 and $n = 3 \times 3$. (a) $n_e = 4$ (b) $n_e = 6$ (c) $n_e = 11$ and (d) Rel. error vs no. of polygon edges (in leaf cells)



(a)

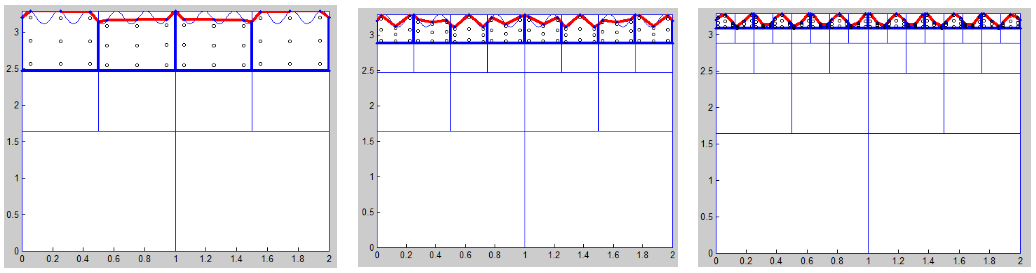


(b)

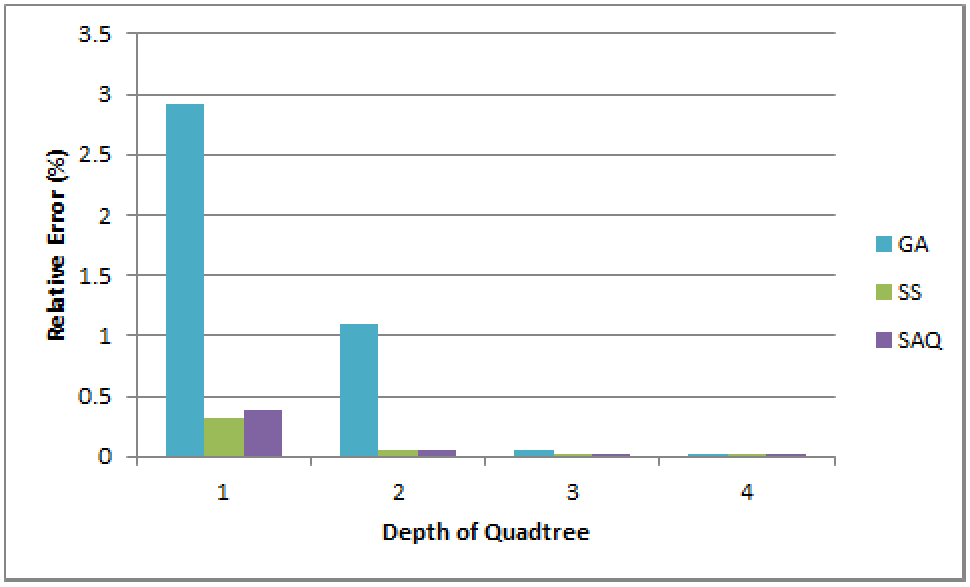


(c)

Figure 3.16: Quadtree integration of f over the wavy domain for various quadrature rules with D $= 1$ and $n_e = 7$. (a) $n = 5 \times 5$ (b) $n = 6 \times 6$ and (c) Rel. error vs quadrature rule



(a) (b) (c)



(d)

Figure 3.17: Quadtree integration of f over the wavy domain for various quadtree depths with $n_e = 7$ and $n = 3 \times 3$. (a) D = 2 (b) D = 3 (c) D = 4 and (d) Rel. error vs depth of quadtree

Table 3.3: Relative errors, order, and computational time for integral of f over the wavy domain

		Relative Error (%)			**Order**			**Time (ms)**		
Depth	**h**	**GA**	**SS**	**SAQ**	**GA**	**SS**	**SAQ**	**GA**	**SS**	**SAQ**
1	1.929	2.9106	0.2268	0.2357				47	125	78
2	0.965	1.0998	0.0640	0.0643	1.40	1.83	1.87	94	172	172
3	0.482	0.0633	**0.0255**	**0.0255**	4.12	1.33	1.34	187	**343**	**343**
4	0.241	**0.0007**	0.0004	0.0004	6.41	5.94	5.94	**562**	655	780

3.6.4 Example 4 - Non-Convex domain with small holes

In this example, we integrate a cubic function $g = 10 + 0.1x + 0.4y - x^2 + 5xy + 2y^2 + 9x^3 - 10x^2y + 10xy^2 - 10y^3$ over the domain shown in Fig. 3.18 with 105 small holes of radius 0.025 ($<< h = 0.25$). We fix the quadtree depth to one and integrate all interior cells using the usual box quadrature. For the leaf cells we generate quadrature points randomly such that they lie completely inside the domain. We use first-order SSA for geometric corrections and first-order TSA/first-order SSA/divergence theorem for topological corrections [i.e. Eq.(3.9) - Eq.(3.11)]. Fig. 3.19 compares the integration of g over the non-convex domain using various correction factors by varying the number of edges of the approximating polygon (Ω_0) for the small features. From the plot it is clear that employing first-order TSA or SSA is superior to simply using divergence theorem (or 0^{th}-order SSA or P method) for topological corrections. In fact, first-order TSA only requires sampling of the basis function and area/volume computations which can be computed with very little effort for most simple shapes and hence is an attractive choice for C_i^T computations for small features. Ignoring the topological correction factor for this problem results in a relative error of 16.201%.

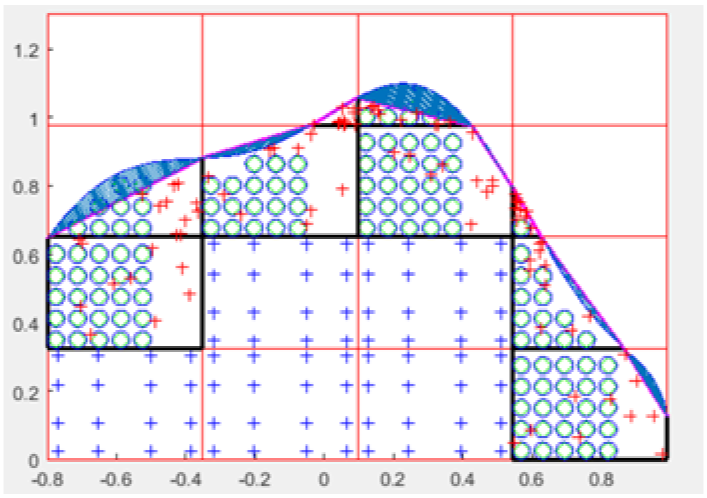


Figure 3.18: A 2D non-convex domain with numerous holes

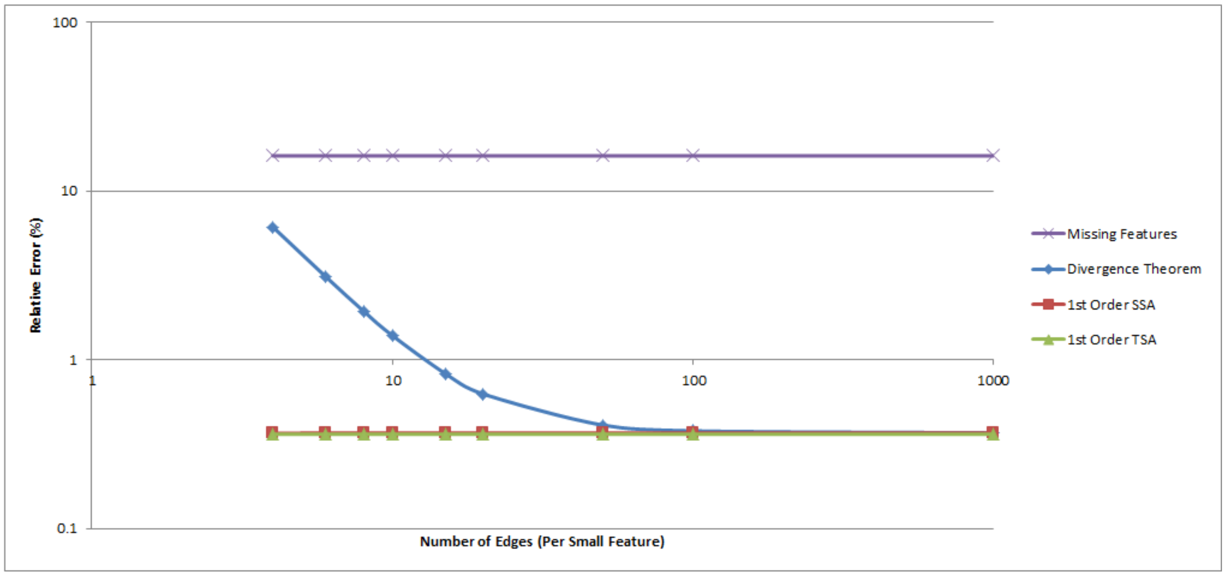


Figure 3.19: Non-Convex domain - Rel. Error Vs Number of Edges

3.6.5 Example 5 - Unit Sphere

In this example, we will compare the accuracy of our integration scheme in integrating trivariate polynomials ($x^p y^r z^r$) with that of the characteristic function method (see section 2.1.3) in an octree based integration setting. Specifically, we employ SAQ/characteristic function approach in the leaf cells of the octree decomposition of a unit sphere. The interior cells are integrated using ordinary lattice rules for a box. We use the standard implementation of characteristic function method available in the open source framework FCMLab [95]. Analogous to previous two examples of this section, Ω_0 (approximate polyhedra) was constructed based on the type of cells arising in the marching cubes (see Fig. 3.20). We employ 27 points for both the methods to integrate polynomials of order up to 3. However, SAQ requires far fewer points (actually 20 points for this example) for reasons that will become apparent in subsection 4.2.2.

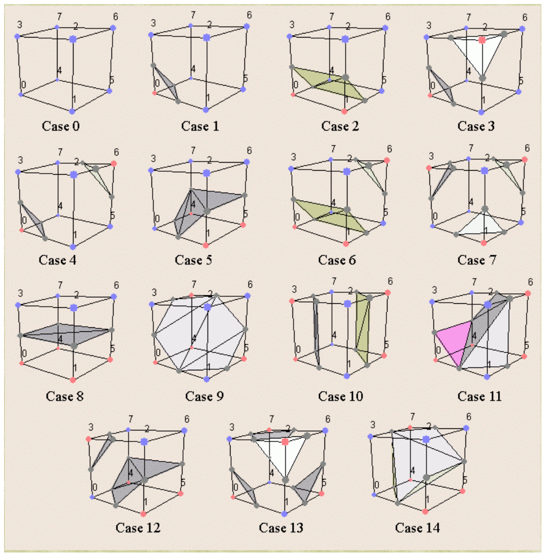


Figure 3.20: Cases in marching cubes [2]

Volume and moment computations Fig. 3.21 and Fig. 3.22 shows the relative error[4] in computing the volume ($\int d\Omega$) and second moment ($\int x^2 d\Omega$) over a unit sphere using the two methods. We employ scaled cartesian product rule to generate quadrature points for both the methods (as shown in Fig. 3.25).

The L^2-norm of the residual error in integrating all trivariate polynomials of order up to 3 is given in Fig. 3.23. It takes just 2 subdivisions for SAQ to drive the error below 0.01%, 0.1% and 0.001 respectively for volume, second moment and residual computations. On the other hand, the characteristic method requires 4 subdivisions to achieve the same level of accuracy clearly owing to the inability of the method to adapt to the underlying geometry.

[4]The exact analytical volume ($\frac{4}{3}\pi$) and second moment ($\frac{4}{15}\pi$) was used as the reference solution in computing these errors

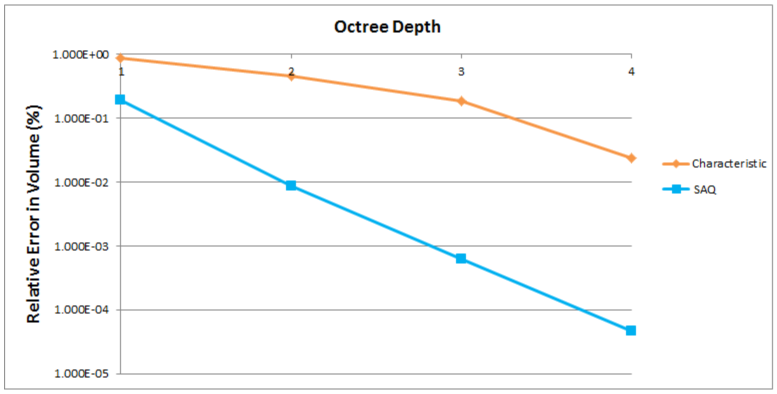


Figure 3.21: Relative error in computing volume of a unit sphere

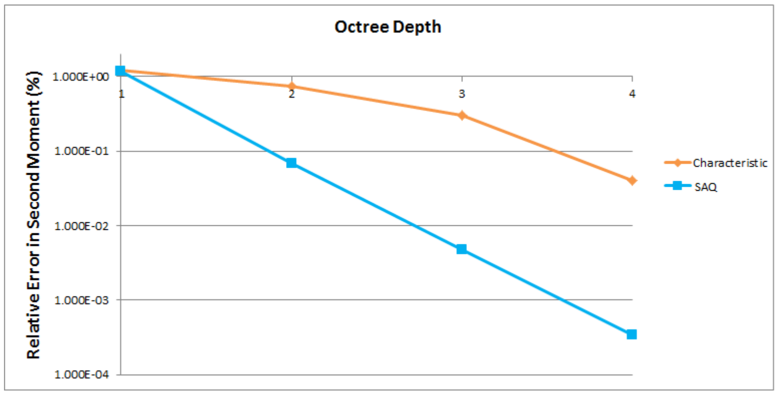


Figure 3.22: Relative error in computing $\int x^2 d\Omega$ over a unit sphere

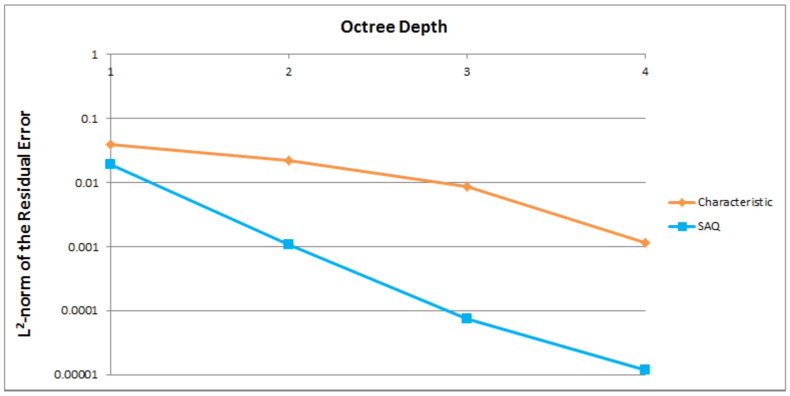


Figure 3.23: The L^2-norm of the residual error in integrating all trivariate polynomials of order up to three ($x^p y^q z^r$ with $p, q, r \geq 0\,, p + q + r \leq 3$) over a unit sphere

Table. 3.4 and Table. 3.5 lists the error and the order of convergence in computing the volume and second moment using the two methods (here **h** is the size of the smallest leaf cell). For this problem, we find that *SAQ has a consistent order of convergence close to 4.* However, the *characteristic function method doesn't seem to posses a consistent order of convergence.* For lower octree depths, it has an order of convergence close to 1 and then it increases to 3 for a octree depth of 4. This suggests that for characteristic function method to produce reliable accurate results one often requires 4 or more subdivisions leading to a substantial increase in the computational cost.

Choice of quadrature points Finally, we also study the effect of position of quadrature points on SAQ. We consider the following three ways to generate quadrature points in the leaf cells :

1. Scaled Cartesian product rule (Fig. 3.25)
2. Random points lying inside the domain (Fig. 3.26)
3. Random points on the polygonal faces of the polyhedral boundary (Fig. 3.27)

Sufficient care is taken to ensure that all generated points lie within the original domain (Ω) and the approximate polyhedral piece Ω_0^i for any given leaf cell I_i.

We compare the L^2-norm of the residual vector for the integration of polynomials of order up to 3 over the unit sphere for all the above. From Fig. 3.24, we find that random domain/boundary points performs slightly better than scaled Cartesian product method for higher octree depths. One main reason for this is that as the leaf cells become smaller and smaller, the quadrature points generated by the scaled Cartesian product method comes closer to one another and thereby affecting the condition number of the moment matrix $[\mathbf{A}]$. Random distribution of points within the domain/boundary avoids this problem completely resulting in better accuracy even for increased octree depths. Also, there is almost no difference in the residual plot for random domain and boundary points. This observation brings forth an important advantage of SAQ which is the *freedom in the choice of quadrature points.* Generating quadrature points using scaled Cartesian product rule could be time consuming as it involves ray-triangle (or ray-polygonal) intersection tests. On the other hand, it is a lot easier to generate random points lying within both the original domain (Ω) and the polygonal domain (Ω_0) or boundary (Γ_0) as it only involves the inside/outside (PMC) test.

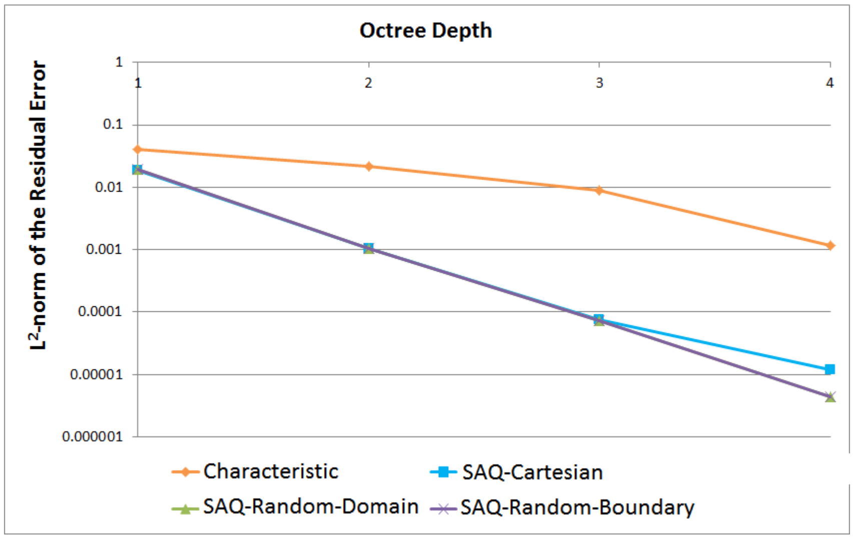


Figure 3.24: The L^2-norm of the residual vector in integrating all trivariate polynomials of order up to three ($x^p y^q z^r$ with $p, q, r \geq 0\,, p+r+r \leq 3$) over a unit sphere for three different quadrature point generation schemes

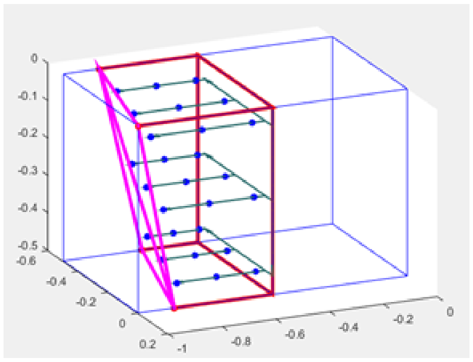


Figure 3.25: Quadrature points generated using scaled Cartesian product rule

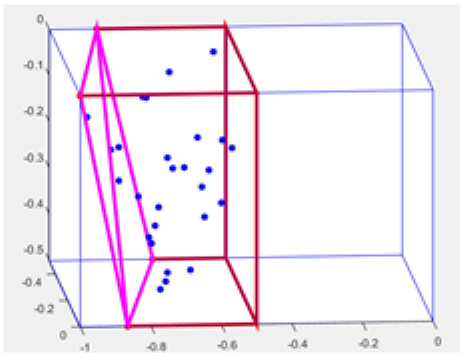


Figure 3.26: Quadrature points generated randomly inside the domain

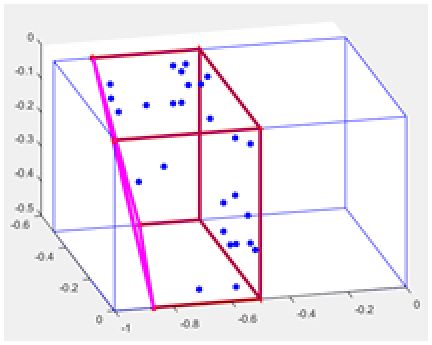


Figure 3.27: Quadrature points generated randomly on the faces of the approximate polyhedron

Table 3.4: Relative error and order of convergence in volume computations for a unit sphere

		Characteristic			**SAQ**		
Depth	**h**	**Volume**	**Error (%)**	**Order**	**Volume**	**Error (%)**	**Order**
1	0.25	4.2246228	8.554E-01		4.1808040	1.907E-01	
2	0.125	4.1699889	4.488E-01	0.930	4.1884232	8.761E-03	4.444
3	0.0625	4.1964163	1.821E-01	1.302	4.1887660	5.768E-04	3.925
4	0.0313	4.1897994	2.409E-02	2.918	4.1887888	3.448E-05	4.064

Table 3.5: Relative error and order of convergence in second moment computations for a unit sphere

		Characteristic			SAQ		
Depth	**h**	**$\int \mathbf{x}^2\, d\Omega$**	**Error (%)**	**Order**	**$\int \mathbf{x}^2\, d\Omega$**	**Error (%)**	**Order**
1	0.25	0.8478973	1.210E+00		0.8279464	1.171E+00	
2	0.125	0.8315221	7.444E-01	0.701	0.8371905	6.775E-02	4.112
3	0.0625	0.8402888	3.021E-01	1.301	0.8377194	4.618E-03	3.875
4	0.03125	0.8380973	4.050E-02	2.899	0.8377557	2.808E-04	4.040

3.6.6 Example 6 - Microstructures

In this example, we will consider the integral of a cubic function $-x^3 - y^3 - z^3 + 5x^2 + 6y^2 + 7z^2 + 8xy - 10xyz$ over a microstructure like domain. Specifically, we would assume the integration domain to be a unit sphere with several thousand small spherical voids as shown in Fig. 3.28. A typical leaf cell of the octree decomposition of this domain is shown in Fig. 3.30. As shown in the plot, the voids are generated randomly (both in size and position) over every leaf cell such that they don't overlap. Likewise, the quadrature points are generated randomly such that they lie within the domain. We use SAQ with first-order SSA based correction factor to account for the geometry of the unit sphere and first/second-order TSA to account for the spherical voids. The relative error in computing the integral using first-order TSA based correction, second-order TSA correction, and without the features (for an octree depth of one) are plotted in Fig. 3.30. It is clear from the plot that the first/second-order TSA predicts the integral accurately up to an error of 0.005 %. As expected, missing the features results in a relative error ranging from 0.013% to 0.264%. This example illustrates one of the important applications of SAQ in efficiently accounting for small features such as voids which is otherwise very difficult to account for in an octree based integration setting.

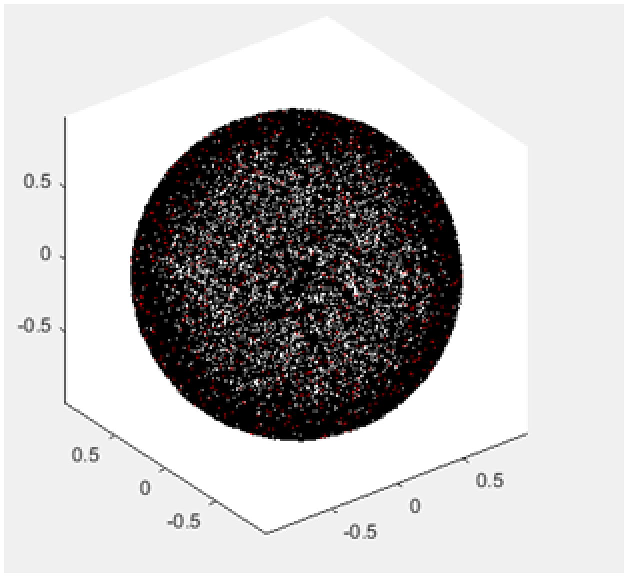


Figure 3.28: Unit sphere with small spherical voids

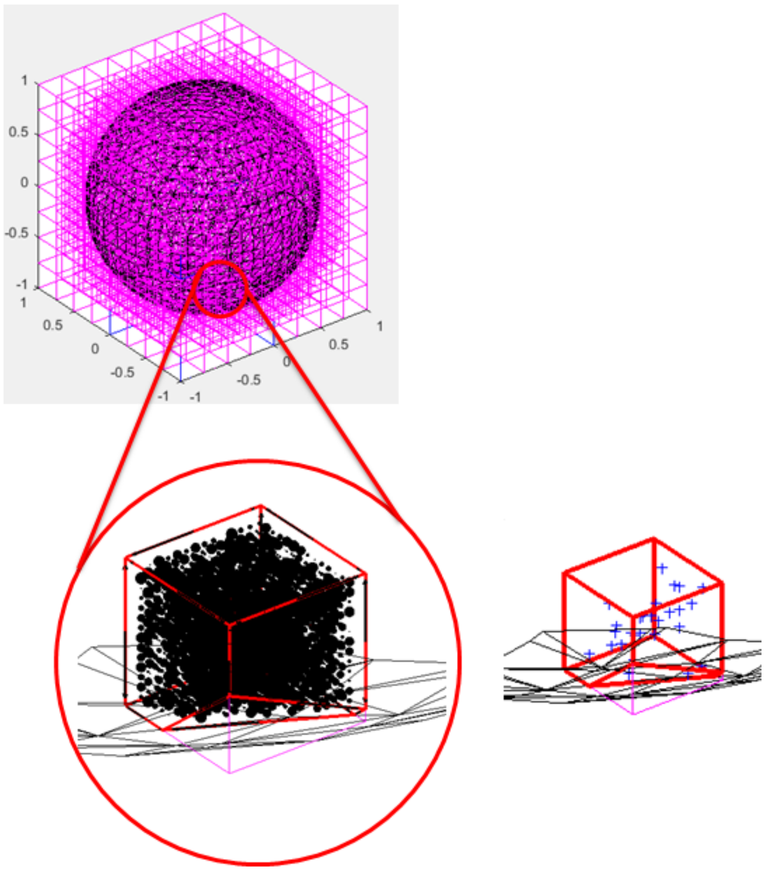


Figure 3.29: A typical leaf cell of the unit sphere contains several hundred voids

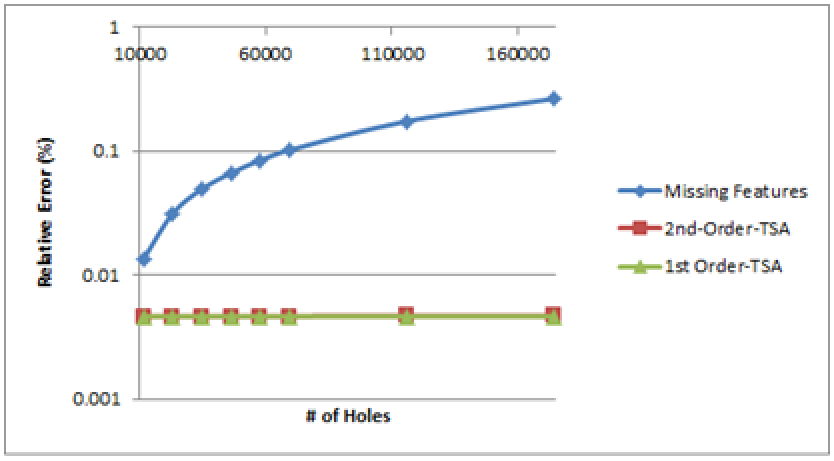


Figure 3.30: Error plot for various hole distribution

3.7 Summary

Shape Aware Quadratures (SAQ) are quadratures that can be used to accurately and efficiently integrate arbitrary 2D/3D domains accounting for small geometric/topological features. A generic formulation of SAQ based on moment fitting equations [64, 116, 137], divergence theorem [41] and sensitivity analysis [75, 106, 123, 80] was given in Section 3.2. SAQ employs a simplified domain (Ω_0) to approximate the moments in the moment fitting equations. Appropriate geometric and topological correction factors are used to correct for the deviation of the shape from Ω. The use of these shape correction factors enables the quadrature rule computed using the moment fitting equations “shape aware”. Thus, the main step in this formulation is the derivation of geometric and topological correction factors using SSA and TSA. First/second-order SSA/TSA based correction factors were derived based on SSA/TSA of moments and tabulated in Table. 3.1 and Table. 3.2. We recommend the use of SSA for geometric corrections and TSA (or SSA) for topological corrections. However, the framework is quite general as one can employ other types of sensitivities such as feature sensitivity [123] and modification sensitivity [80] to derive the correction factors.

We demonstrated the use of SAQ in integrating arbitrary polynomial functions over arbitrary 2D/3D domains in the presence of small features (such as notches, boundary noise and holes) and compared it with some standard methods: scaled Cartesian product rule as recommended in [132], geometric adaptive (GA) integration method proposed in [22], polygonal (P) approximation method, shape sensitivity (SS) method and characteristic function approach. SAQ was shown to be superior to GA, P and characteristic methods in terms of accuracy and comparable to SS method in most cases.

Recall from Section 2.1.4 that we abandoned the SS approach due to its unrealistic requirements such as function extension outside the domain and domain velocity computation inside the domain. Yet, experimental results show that SS and SAQ correlate well with each other when applied to functions that can be readily defined even outside the original domain of integration. In fact this is not a coincidence. To understand this, consider an alternate way to formulate SAQ (with SSA based correction factors). To begin with, let us rewrite the boundary integral in Eq.(2.26) in domain form as

$$\begin{aligned} \int_{\Gamma_0} f(\mathbf{X}) V_N(\mathbf{X}) d\Gamma_0 &= \int_{\Omega_0} \nabla \cdot (f(\mathbf{X})\mathbf{V}(\mathbf{X})) d\Omega_0 \\ &= \int_{\Omega_0} f(\mathbf{X})\Psi(\mathbf{X}) d\Omega_0 \end{aligned} \tag{3.27}$$

where

$$\Psi(\mathbf{X}) = \begin{cases} \frac{\nabla\cdot(f(\mathbf{X})\mathbf{V}(\mathbf{X}))}{f(\mathbf{X})} & : \mathbf{X} \in A \\ \nabla \cdot (f(\mathbf{X})\mathbf{V}(\mathbf{X})) & : \mathbf{X} \in \Omega_0 - A \end{cases}$$

where $A = \{\mathbf{X} \in \Omega_0 | f(\mathbf{X}) \neq 0\}$ is the set of points in the domain where the function f is non-zero. It can be easily proved that $f(\mathbf{X})\Psi(\mathbf{X})$ is integrable and that the equality holds in Eq.(3.27). Substituting Eq.(3.27) in Eq.(2.26) we get

$$I \approx \int_{\Omega_0} f(\mathbf{X}) d\Omega_0 + \int_{\Omega_0} f(\mathbf{X})\Psi(\mathbf{X}) d\Omega_0 = \int_{\Omega_0} f(\mathbf{X})(1 + \Psi(\mathbf{X})) d\Omega_0 \tag{3.28}$$

Setting $W(\mathbf{X}) = (1 + \Psi(\mathbf{X}))$, we get:

$$I \approx \int_{\Omega_0} W(\mathbf{X}) f(\mathbf{X}) d\Omega_0 \tag{3.29}$$

Now, we have $M_i \approx \int_{\Omega_0} W(\mathbf{X}) b_i(\mathbf{X}) d\Omega_0$ with $W(\mathbf{X}) = (1 + \Psi(\mathbf{X}))$. Further, substituting for $W(\mathbf{X})$ and simplifying we obtain exactly the same expression for M_i as before i.e. Eq.(3.18). In other words, whether we first approximate the integral of f (over Ω by applying SSA over Ω_0) and then derive M_i or approximate M_i directly using SSA, we obtain the same weights. Equivalently, this implies that SAQ is as good as SS but without the limitations of SS as outlined in Section 2.1.4. Moreover, SAQ generalizes this idea to other types of sensitivities and provides a more general framework for integration of arbitrary domains in two and three dimensions.

The main application of SAQ is in non-conforming cell based integration. We observed that SAQ, when used in the leaf cells of quadtrees/octrees, required fewer subdivisions / quadrature nodes to resolve the geometry of the integration domain when compared to other methods. We also note that the method is formulated irrespective of how the original domain Ω is represented, and it can be used with or without an integration mesh. The latter observation suggests that SAQ is suitable for a variety of IB methods such as Scan & Solve [85, 86], Finite Cell Method [66], and other immersed boundary methods [105], which will be the subject of the next chapter.

Chapter 4

SAQ in Finite Cell Method

In this chapter, we will present convergence/performance studies of SAQ in the context of an immersed boundary method called the Finite Cell Method (FCM). We will assume in this chapter that Ω_0 is always homeomorphic to Ω and hence $C_i^T = 0$. Further, we will only use first-order SSA for C_i^G computations i.e. we will employ Eq.(3.12) for moment approximations. In this chapter, we will closely follow our exposition in [130].

4.1 Finite Cell Method

We have already presented a brief overview of FCM in chapter 1. Here, we will simply restate FCM in the context of elastostatics which will be the basis for our convergence/performance studies. FCM was originally introduced in [66] as an extension to the p-version of the finite element method [24, 54]. FCM combines the fictitious domain approach with a higher-order approximation basis, the representation of the geometry by adaptive quadrature based on recursive bisection, and the weak imposition of unfitted boundary conditions [118]. FCM can operate on any geometry as along as the geometry supports point membership classification (PMC) query i.e. whether a point is located inside or outside the physical domain. Fig. 4.1 illustrates the fictitious domain concept that lies at the heart of the FCM.

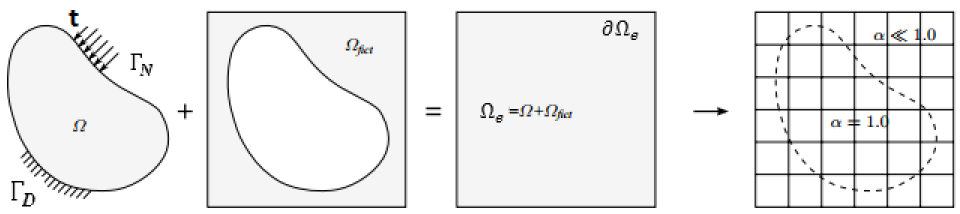


Figure 4.1: The fictitious domain approach : the physical domain Ω is extended by the fictitious domain Ω_{fict} into an embedding domain Ω_e to allow easy meshing of complex geometries. The influence of Ω_{fict} is penalized by α [118]

The embedding domain Ω_e consists of the physical domain of interest Ω and the fictitious domain extension Ω_{fict}. Analogous to classical FEM, the first step in FCM is the discretization of the variational (or weak) form for the boundary value problem under consideration. Let us consider the standard elastostatics problem for the purpose of illustration. The FCM variational form for this problem reads as follows :

find $\mathbf{u} \in H^1(\Omega_e)$ *such that*

$$\int_{\Omega_e} \alpha\boldsymbol{\sigma} : \delta\boldsymbol{\epsilon} d\Omega_e - \int_{\Omega_e} \alpha\delta\mathbf{u}.\mathbf{b}\, d\Omega_e - \int_{\Gamma_N} \delta\mathbf{u}.\mathbf{t}\ d\Gamma_N \ \ \forall\delta\mathbf{u} \in H^1(\Omega_e) \tag{4.1}$$

$$\mathbf{u} = \mathbf{u_0}\, \forall\mathbf{x} \in \Gamma_D \tag{4.2}$$

$\boldsymbol{\sigma}$, $\mathbf{b}$, and $\mathbf{t}$ are respectively the Cauchy stress tensor, the body force vector, and the surface traction vector. Neumann boundary conditions are also specified over the boundary of the embedding domain $\partial\Omega_e$, where $\sigma.\mathbf{n} = 0$ by definition. Dirichlet boundary conditions are specified over Γ_D of the physical domain by the prescribed value of the field variable $\mathbf{u_0}$. The scalar factor α is defined as follows

$$\alpha(x) = \begin{cases} 1.0 \quad \forall x \in \Omega \\ \\ 10^{-q} \ \forall x \in \Omega_{fict} \end{cases} \tag{4.3}$$

In Ω_{fict}, α must be chosen as small as possible, but large enough to prevent extreme ill-conditioning of the stiffness matrix [66, 13]. Typical values of α range between 10^{-5} and

10^{-10}. This scalar parameter α ensures that we are actually solving the boundary value problem on Ω and not on the extended domain Ω_e. Using a structured grid of higher-order elements (as shown in Fig. 4.1), the field variable $\mathbf{u}$ and its variation $\delta\mathbf{u}$ are discretized using polynomial basis functions resulting in a linear system $[\mathbf{K}]\{\mathbf{u}\} = \{\mathbf{f}\}$. The global stiffness $[\mathbf{K}]$ and global force vector $\{\mathbf{f}\}$ are obtained by assembling the following cell stiffness matrix ($[\mathbf{k_c}]$) and force vector ($\{\mathbf{f_c}\}$) for all the cells ($c = 1$ to n_c) :

$$[\mathbf{k_c}] = \int_{\Omega_c} [\mathbf{B_c}]^T \alpha [\mathbf{C_c}][\mathbf{B_c}] d\Omega_c \tag{4.4}$$

$$\{\mathbf{f_c}\} = \int_{\Omega_c} \alpha [\mathbf{N_c}]^T \{\mathbf{b_c}\} d\Omega_c + \int_{\Gamma_N^c} [\mathbf{N_c}]^T \{\mathbf{t_c}\} d\Gamma_N^c \tag{4.5}$$

$\mathbf{B_c}$, $\mathbf{C_c}$, $\mathbf{N_c}$, $\mathbf{t_c}$ and $\mathbf{b_c}$ are respectively the strain-displacement matrix, constitutive matrix, shape function matrix, traction vector and body force vector associated with the cell c. Solving the above linear system results in the nodal displacements $\mathbf{u}$. As can be seen from the above equations, the imposition of Neumann boundary conditions requires the computation of integrals over the surface of Ω. This can be easily accomplished by local surface meshing. However, *enforcement of Dirichlet boundary conditions* (Eq.(4.2)) is not straight forward primarily owing to the non-conformity of the mesh and/or the basis functions not satisfying the Kronecker delta property[5] [114]. Lot of research have gone into the enforcement of Dirichlet boundary conditions in meshfree methods. There are a number of schemes such as the *Lagrangian Multiplier Method* [52, 37], *Penalty Method* [53, 56, 93], *Augmented Lagrangian Method* [114], *Nitsche's Method* [135, 63, 102, 99] and *Kantorovich's Approach* [73, 72, 85, 86] that are available to weakly or strongly enforce Dirichlet boundary conditions.

Notice that the formulation of stiffness matrix and the load vector requires volumetric integration of discontinuous functions (due to α) over a box domain (see Eq.(4.4) and Eq.(4.5)). In other words, due to the introduction of the α factor in the weak form of the FCM, we have effectively transformed the integral of a continuous integrand over the ar-

[5]A set of basis functions $\eta_i(x_j)$, associated to a set of nodes or particles x_i, is said to satisfy the Kronecker delta property if $\eta_i(x_j) = \delta_{ij} \forall i, j$ [114]

bitrary original domain Ω into an integral of a discontinuous integrand over a box domain $\cup_{i=1}^{n_c} c_i$. Then, the integral over each of these cells can be computed by employing the classical Gaussian quadrature in combination with an hierarchical partitioning scheme [13, 138] such as a quadtree [48, 49] (2D) or octree [96, 13, 48, 49] (3D). This approach of integrating a continuous integrand over an arbitrary domain by transforming it into an integral of a discontinuous integrand over a regular domain is called as the characteristic function approach and is the popular integration scheme employed in FCM [12]. Although this transformation is mathematically perfectly legitimate and sensible, it poses a serious problem when evaluated numerically owing to the discontinuity of α. This is because classical Gauss quadrature schemes are designed assuming that the integrand is a polynomial of some degree [132]. For instance, n point Gauss quadrature gives accurate results for integral of polynomials of degree up to $2n-1$ in 1D. If the same integration rule is applied to a non-polynomial function, it gives a value of the integral of a polynomial approximation to the integrand. Thus, the accuracy of the integrand by polynomials determines the integration accuracy. It is also well known that polynomial approximation of discontinuous functions tend to oscillate in the neighborhood of the discontinuity. For example, plots in Fig. 4.2 illustrate polynomial approximation of a Heaviside function by polynomials of 4th (Fig. 4.2(a)) and 20th (Fig. 4.2(b)) degree on the segment [1, 1]. If we raise the degree of an approximating polynomial, the amplitude of oscillations raises as well [22].

This implies that for the characteristic function method to produce accurate results one often needs a very fine partitioning of the domain leading to excessive boundary cell fragmentation as shown in Fig. 4.3. This makes the method prohibitively expensive for realistic 3D CAD models such as sculptures, engines, and bones. **Thus, *numerical integration* is one of the major bottlenecks of FCM (or any IB method for that matter)**. Hence, in this chapter, we examine the suitability / efficacy of SAQ in the context of FCM. Specifically, we compare the convergence/performance of SAQ and the characteristic function method [12] in solving 2D/3D elastostatic problems using the FCM. For this purpose, we implemented SAQ in the open source FCM framework in MATLAB called the FCMLab [95]. From the experiments, we observe that SAQ has superior order of convergence and also requires fewer subdivisions/less time to achieve a given accuracy compared to the characteristic function method. In addition, SAQ is found to offer a great deal of flexibility in the choice of quadrature points and basis functions that is not usually available for other integration schemes such as the characteristic function method.

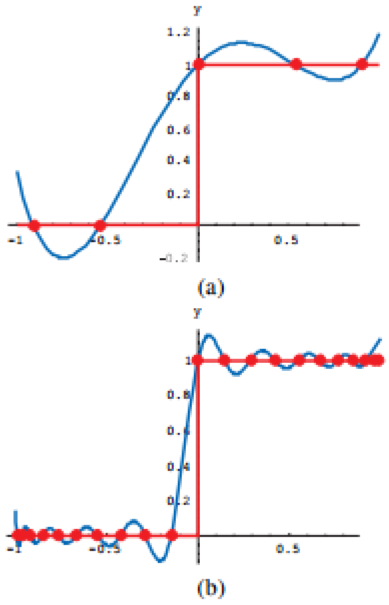


Figure 4.2: Polynomial interpolation of a Heaviside function: (a) through 5 points; (b) through 21 points [22]

.

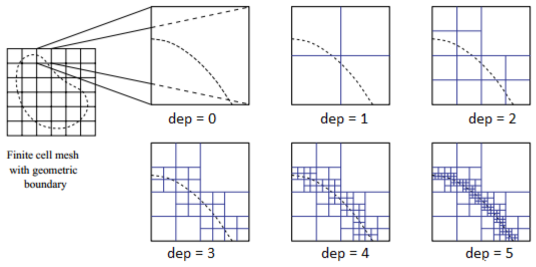


Figure 4.3: Illustration of boundary cell fragmentation in characteristic function method [118]

4.2 Implementation

SAQ based integration could be implemented in any 2D / 3D meshfree FEA system. To demonstrate its application in a meshfree system, we implemented SAQ based integration in FCMLab [95], an open source framework in MATLAB that enables 2D/3D meshfree analysis based on the Finite Cell Method (FCM) [66].

FCMLab currently supports the *Penalty Method* [53, 56, 93] and *Nitsche's Method* [135, 63, 102, 99] for the enforcement of Dirichlet boundary conditions (in weak sense). Furthermore, it supports higher order polynomial basis functions for the discretization of any given field of interest (such as displacements or temperature). Neumann boundary conditions are enforced by means of surface integration over the surface mesh. The domain integration is carried out over a separate integration mesh different from the grid of basis functions. The integration mesh is partitioned using space trees such as quadtrees (2D) / octrees (3D) and classical Gauss quadrature nodes (for boxes) are allocated in each of the integration cells. Depending on whether a quadrature node is inside or outside the domain, appropriate scaling factor ($\alpha = 1$ or $\alpha = 10^{-q}$) is used for the domain integrands. In short, FCMLab currently supports volumetric integration based on the characteristic function method (discussed in section 2.1.3). For a detailed discussion of the capabilities and design of FCMLab we refer the reader to [95].

We implemented SAQ in the FCMLab framework in order to compute the 2D/3D integrals arising in FCM and in general. In this section, we will give a brief overview of implementation aspects of SAQ both in two and three dimensions.

4.2.1 2D Implementation

For 2D domains, we used a quadtree based integration to compute the desired area integrals. A quadtree decomposition of the domain (Ω) results in three types of cells i.e. inner, outer, and leaf (magenta cells in Fig. 4.6) cells. For the inner and outer[6] (blue cells in Fig. 4.6) cells, we allocated quadrature nodes based on the classical Gauss quadrature rule for rectangles [132]. Hence, in this section, we will limit our discussion only to the integration of leaf cells using SAQ.

Initialization

Approximate polygon (Ω_0) construction The approximating polygon (Ω_0) homeomorphic to the domain (Ω) could be constructed in a number of ways including (i) ray casting [119], (ii) coarse quadtree decomposition [70] of Ω, and (iii) coarse polygonalization [97] of Ω. For the sake of simplicity, we employed ray casting to construct the approximate polygon (Ω_0). However, what is needed for applying SAQ in a quadtree based integration setting are the polygonal pieces $\Omega_0^i = I_i \bigcap \Omega_0$ that approximates $\Omega^i = I_i \bigcap \Omega$ for every integration cell I_i. In other words, Ω is the union of pieces Ω^i ($\Omega = \bigcup_{i=1}^{N_I} \Omega^i$) that is approximated by $\Omega_0 = \bigcup_{i=1}^{N_I} \Omega_0^i$. Since, the inner and outer cells are integrated using classical quadrature scheme for boxes, it is enough if we just construct approximate polygonal pieces Ω_0^i for the leaf cells.

Specifically, first the leaf cells are classified into one of the 15 marching square cases of Fig. 4.5 using point membership classification (PMC). For cases 1,2,3,4,6,7,8,9,11,12,13 and 14, we construct approximate polygons homeomorphic to the leaf cell by casting rays parallel to integration rays and intersecting it with the original domain (Ω) as shown in Fig. 4.4.

[6]In principle, we could ignore the outer cells as they don't contribute to the integral. However, we included outer cells in the computation in order to avoid stability issues that may arise when very little of the support of the basis function is in Ω (in solving boundary value problems). We note here that there are better ways to improve stability as was suggested by Höllig et al. [73] and Babuška I. and Banerjee U. [55].

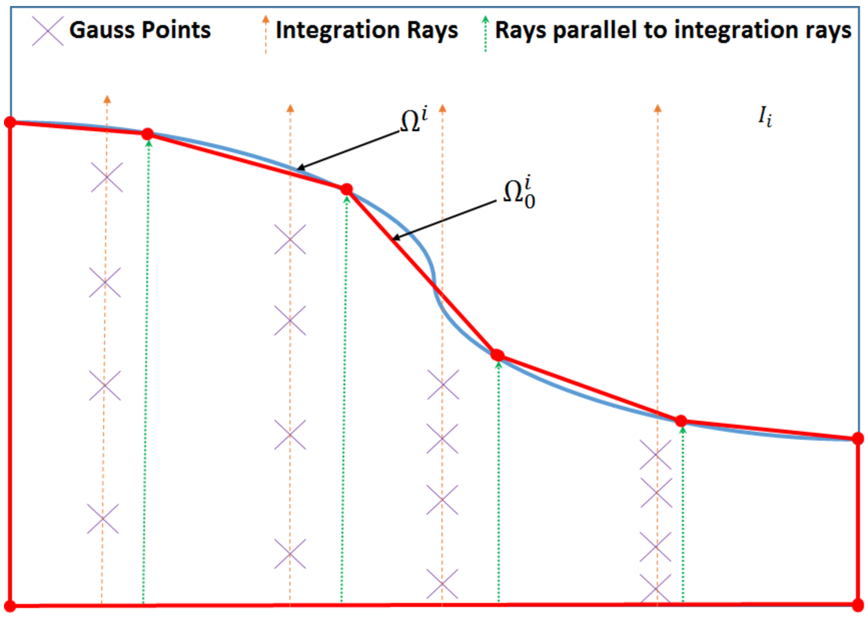


Figure 4.4: Approximate polygon (Ω_0^i) construction and quadrature point generation over a leaf cell (I_i) by ray casting.

The ambiguous cases (case 5 & 10) were handled simply by recursively subdividing the cell until it falls into one of the basic cases (0,1,2,3,4,6,7,8,9,11,12,13,14 and 15) or the size of cells were consistent with the acceptable errors in integration.

Choice of basis functions There are a number of choices for the basis functions in 2D. Some popular choices include bivariate, Legendre and Chebyshev polynomials. In our 2D implementation, for the desired order of integration (o), we chose a set of bivariate polynomials ($\{x^p y^r \Bigm| p, r \geq 0 \ , \ p + r \leq o\}$) as our set of basis functions.

Quadrature point generation Quadrature nodes (in the leaf cells) were generated depending on the type of leaf cell arising in marching squares (see Fig. 4.5). To be precise, the quadrature nodes for cases 1,2,3,4,6,7,8,9,11,12, 13 & 14 were allocated based on scaled Cartesian product rule via ray casting (see Fig. 4.4) such that the points lie within both the original domain (Ω) and the approximate polygon (Ω_0). This is because choosing points outside Ω will lead to dealing with the discontinuity of an otherwise continuous integrand (as in characteristic function approach) and thereby deteriorating SAQ's accuracy. Moreover, choosing points outside Ω_0 is known to slightly decrease the accuracy of SAQ as was observed in our experiments in the previous chapter. Typical distribution of quadrature points in a 2D domain for the SAQ scheme is illustrated in Fig. 4.6.

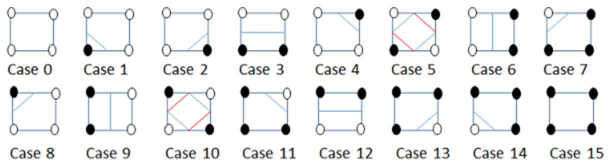


Figure 4.5: Cases in marching squares

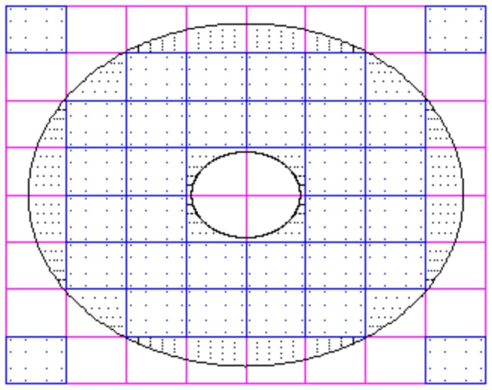


Figure 4.6: Quadrature point distribution for SAQ scheme over a 2D domain

Moment computations

In order to set up the moment fitting equations (Eq.(3.1)), it is required to compute the moments (Eq.(3.12)) and the $[\mathbf{A}]$ matrix (of Eq.(3.1)). The moments (Eq.(3.12)) are basically boundary integrals over the approximate polygon (Ω_0) and therefore requires computation of line integrals over the edges of Γ_0. Notice that this can be easily accomplished by employing the classical 1D Gauss quadrature scheme [132] over the edges of the polygon provided the design velocity (V_N) is known at each of these 1D quadrature points. Recall from the previous chapter that the design velocity (V_N) is simply the shortest distance (γ) from the given point on Γ_0 (here the 1D quadrature points) to the original boundary (Γ) as measured in the normal direction. One simple way to compute this distance is by a simple ray casting algorithm [119] as illustrated in Fig. 4.7. Thus, using the generated quadrature nodes, the approximating polygon and the design velocity, the approximate moments (Eq.(3.12)) were computed using the bivariate ($x^p y^r$) polynomial basis functions for the desired order of integration. Then, [$\mathbf{A}$] matrix can easily be setup by simply evaluating the chosen set of basis functions (in this case $x^p y^r$) at all the generated quadrature points.

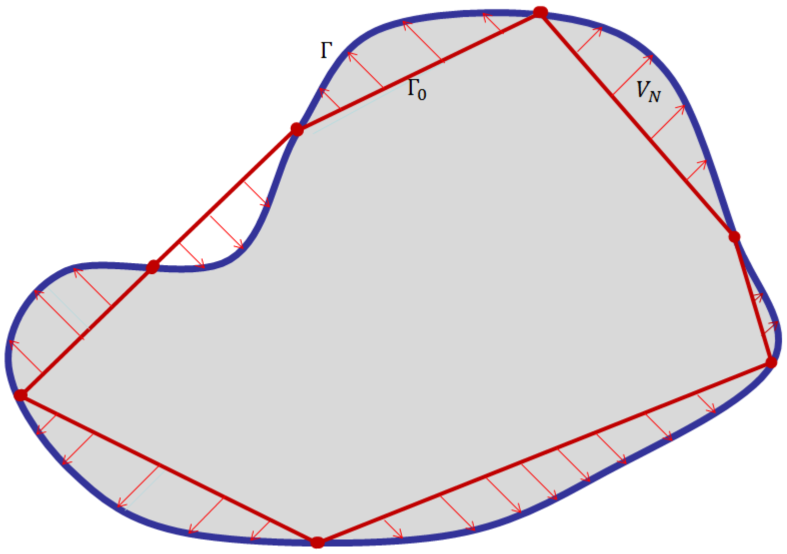


Figure 4.7: Design velocity is simply the distance to the boundary in a direction normal to Γ_0

Solving for weights and evaluation

Thus, for every leaf cell, we setup and solve a separate set of moment fitting equations in least square sense (using the QR algorithm [82]) in order to obtain a set of weights that adapt to the geometry in that leaf cell. In other words, we generate a new quadrature rule on the fly for every leaf cell accounting for the type of geometry in that cell. Each of these generated quadrature is then used in Eq.(2.6) to estimate the desired area integral contribution of the leaf cells.

As mentioned in the beginning of this subsection, inner/outer cell's contribution was computed using the classical Gauss quadrature scheme for rectangles [132]. Thus, summing up the contributions from all inner, outer and leaf cells gives us an estimation of the desired area integral.

4.2.2 3D Implementation

For 3D domains, we used a octree based integration to compute the desired volume integrals. Similar to quadtree decomposition, an octree decomposition of the domain (Ω) results in three types of cells i.e. inner, outer, and leaf cells. The inner and outer cells can be handled by standard Gauss quadrature for boxes [132]. Hence, in this section, we will limit our discussion only to the integration of leaf cells using SAQ.

Initialization

Approximate polyhedron (Ω_0) construction The approximating polyhedron Ω_0 homeomorphic to the domain Ω can be constructed in a number of ways including (i) ray casting [119], (ii) coarse octree decomposition [96] of Ω, and (iii) coarse surface meshing [97] of Ω. For the sake of simplicity, we construct this by the polygonalization of Ω through an octree decomposition. However, what is needed for applying SAQ in an octree based integration setting are the polyhedron pieces $\Omega_0^i = I_i \bigcap \Omega_0$ that approximates $\Omega^i = I_i \bigcap \Omega$ for every integration cell I_i. In other words, Ω is the union of pieces Ω^i ($\Omega = \bigcup_{i=1}^{N_I} \Omega^i$) that is approximated by $\Omega_0 = \bigcup_{i=1}^{N_I} \Omega_0^i$. Since, the inner and outer cells are integrated using classical quadrature scheme for boxes, it is enough if we just construct approximate polyhedron pieces Ω_0^i for the leaf cells. Thus, for each leaf cell I_i, we generate the approximate polyhedron Ω_0^i as shown in Fig. 4.10(b).

To be precise, first the leaf cells are classified into one of the 15 cases of Fig. 4.8 using point membership classification (PMC). For cases 1,2,5,8,9,11 and 14, we first compute the polygonal boundary Γ_{p1}^i (magenta triangles in Fig. 4.10(b)) approximating Γ^i based on marching cubes algorithm [83]. This polyhedral boundary (Γ_{p1}^i) forms only a portion of the approximate polyhedral boundary Γ_0^i. The other faces of the polyhedron are obtained by trimming the leaf cell I_i by Γ_{p1}^i resulting in the polygons (Γ_{p2}^i) bounded by red lines in Fig. 4.10(b)). Thus, $\Gamma_0^i = \Gamma_{p1}^i \bigcup \Gamma_{p2}^i$ and Ω_0^i is the volume bounded by this boundary Γ_0^i. Γ_{p1}^i is stored as a bunch of triangles while Γ_{p2}^i is stored as a bunch of planar n-gons. The ambiguous and disjoint cases (cases 3,4,6,7,10,12 and 13) were handled by recursively subdividing until it falls into one of the above basic cases (case 0,1,2,5,8,9,11 and 14) or the size of cells were consistent with the acceptable errors in integration.

Choice of basis functions There are a number of choices for the basis functions in 3D. Some popular choices include trivariate, Legendre and Chebyshev polynomials. In our 3D implementation, for the desired order of integration (o), we chose a set of trivariate polynomials ($\{x^p y^q z^r \,\Big|\, p, q, r \geq 0 \;, \; p+q+r \leq o\}$) as our set of basis functions.

Quadrature point generation There is a great deal of flexibility in the choice of generating quadrature points. One way to generate quadrature points is based on the type of leaf cell arising in marching cubes. However, in 3D, we have 256 marching cube cases to handle. But, by employing rotations and symmetries this can be reduced to the 15 basic cases shown in Fig. 4.8. The quadrature nodes for the non-ambiguous/non-disjoint cases 1,2,5,8,9,11 and 14 were allocated by scaled Cartesian product rule such that the points lie both within the original domain (Ω) and the approximate polyhedron (Ω_0). This is because choosing points outside Ω will lead to dealing with the discontinuity of an otherwise continuous integrand (as in characteristic function approach) and thereby deteriorating SAQ's accuracy. Likewise, choosing points outside Ω_0 is known to slightly decrease the accuracy of SAQ as was observed by us in the previous chapter. Generation of quadrature nodes in a typical leaf cell is illustrated in Fig. 4.10(d). For a leaf cell I_i, one could also generate quadrature points randomly such that the points lie both within the approximate polyhedron piece Ω_0^i and the actual domain Ω as we have already see in section 3.6.5. The ambiguous/disjoint cases (cases 3,4,6,7,10,12, and 13) were handled as before.

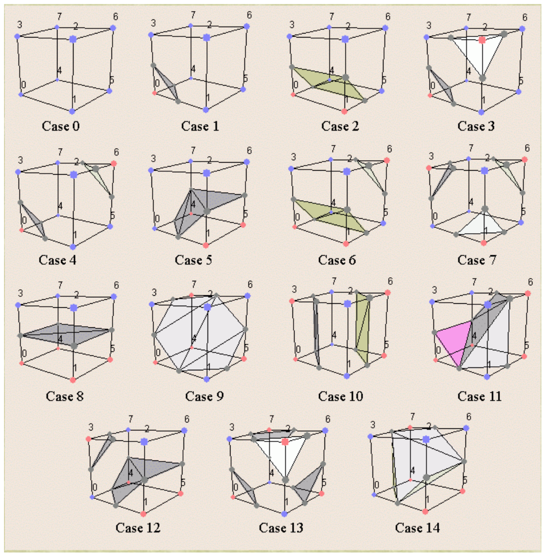


Figure 4.8: Cases in marching cubes [2]

Minimum number of quadrature points for SAQ (in leaf cells) is dictated by the total number of basis functions chosen for the given order o. This is because, to make the moment fitting equations solvable we at least need as many equations (basis functions) as the number of unknowns (quadrature weights). For example, if we choose all trivariate polynomials of order up to 3 (i.e. $\{1, x, y, z, x^2, xy, yz, xz, y^2, z^2, x^3, x^2y, x^2z, xyz,$ $y^2x, y^2z, y^3, z^2x, z^2y, z^3\}$) as our basis functions, then the minimum number of quadrature points required for SAQ (per leaf cell) is only 20. However, comparing this with the characteristic function method, we find that one requires at least 27 points ($3 \times 3 \times 3$) to integrate a polynomial of order 3. This is because quadrature nodes in characteristic function method are allocated based on the Cartesian product of 1D Gauss quadrature rule and the smallest cube greater than or equal to 20 is $3^3 = 27$.

Thus, minimum number of quadrature points for SAQ is equal to the number of complete polynomials of given order (o) and is equal to $\frac{(o+d)!}{d!o!}$ [20] (where $d = 2$ for 2D and $d = 3$ for 3D domains). For characteristic method, it is simply equal to $(o + 1)^d$ as it is based on the Cartesian product of 1D Gauss quadrature rule. Fig. 4.9 shows the savings in quadrature points per leaf cell for the SAQ relative to the characteristic function method for different orders of integration. From the plot one can see that, unlike 2D, the savings in 3D is considerable especially for higher orders of integration which is critical from the context of FCM.

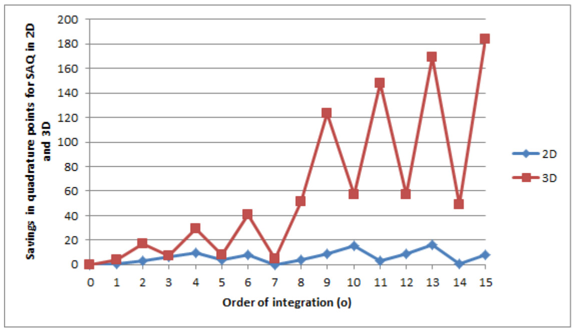


Figure 4.9: Savings in quadrature points (per leaf cell) for the SAQ scheme in 2D and 3D

Moment computations

In order to set up the moment fitting equations (Eq.(3.1)), it is required to compute the moments (Eq.(3.12)) and the [**A**] matrix (of Eq.(3.1)). The moments (Eq.(3.12)) are basically boundary integrals over the approximate polyhedron (Ω_0) and therefore requires computation of surface integrals over the polygonal faces of Γ_0. Notice that this can be accomplished by first performing a coarse triangulation of the general polygonal faces and then applying the standard quadrature rule for triangles [64]. Further, in order to compute the moments it is required to compute the design velocity (V_N) at each of these surface quadrature points. Recall from the previous chapter that the design velocity (V_N) is simply the shortest distance (γ) from the give point on Γ_0 (here the surface quadrature points) to the original boundary (Γ) as measured in the normal direction. One simple way to compute this distance is by a simple ray casting algorithm [119] as illustrated in Fig. 4.10(c). Thus, using the generated quadrature nodes, the approximating polyhedron and the design velocity, the approximate moments (Eq.(3.12)) were computed using the trivariate ($x^p y^q z^r$) polynomial basis functions for the desired order of integration. Then, [**A**] matrix can easily be setup by simply evaluating the chosen set of basis functions (in this case $x^p y^q z^r$) at all the generated quadrature points.

Solving for weights and evaluation

This step is exactly the same as in 2D and hence we refer the reader to subsection 4.2.1 for details.

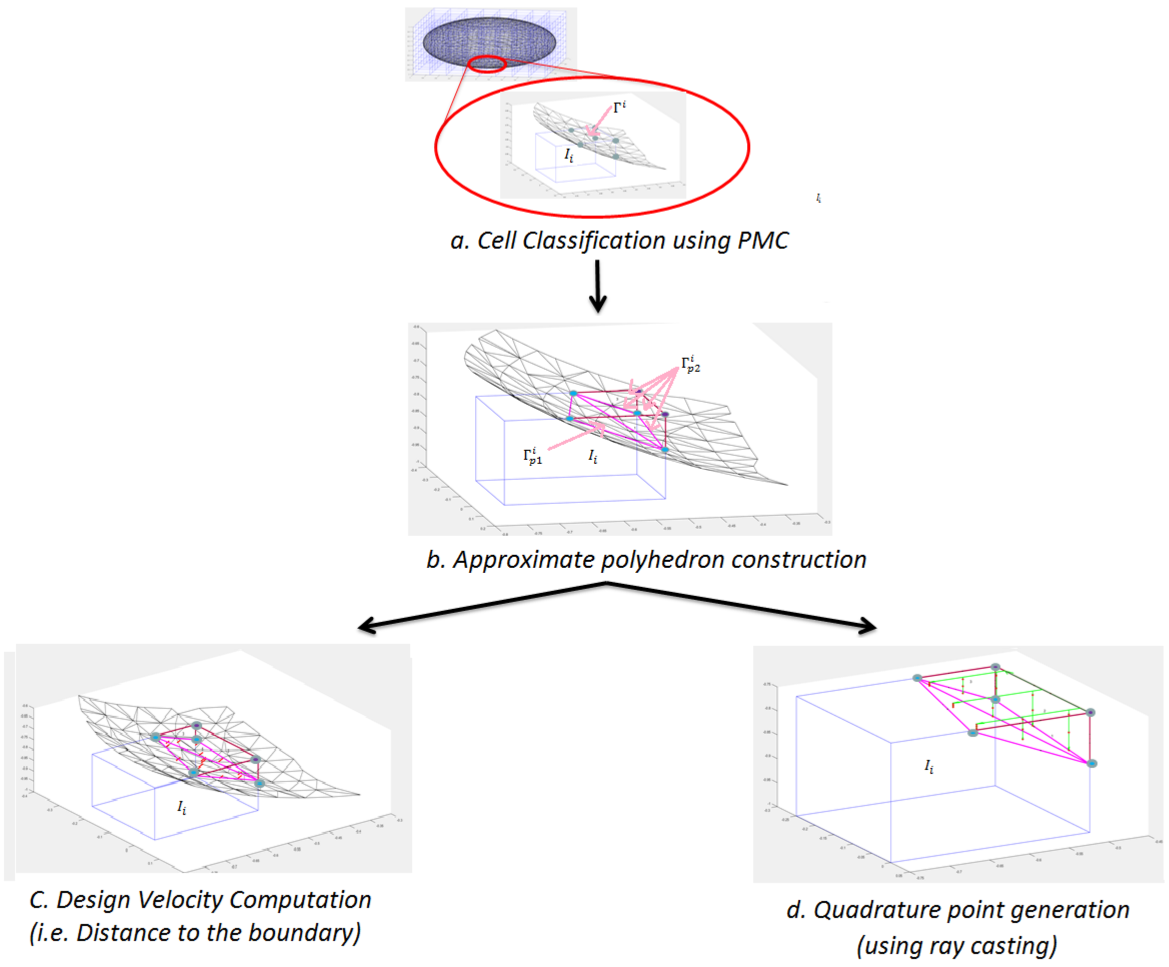

Γ^i
I_i
a. Cell Classification using PMC
Γ^i_{p2}
Γ^i_{p1}
I_i
b. Approximate polyhedron construction
I_i
C. Design Velocity Computation
(i.e. Distance to the boundary)
I_i
d. Quadrature point generation
(using ray casting)

Figure 4.10: (a) The leaf cells of the octree are first classified into one of the 15 cases using PMC (b) Depending on the cell type, an approximate polyhedron homeomorphic to the domain in each of the leaf cells is constructed (c) Design velocity is computed at the surface quadrature points of the approximate polyhedron (Γ_0) (d) Volumetric quadrature points are generated such that it lies within the actual and polyhedral domain using ray casting (actual surface not shown for clarity)

Note on PMC There are many fast algorithms for PMC catering to different geometric representations. For geometries represented by implicit functions, PMC is straight forward and so is extremely fast. For geometries represented by triangular meshes, PMC by ray boundary intersection is rapid since ray-cell intersections have closed form solutions [84]. Even faster techniques take advantage of signed triangles or enclosing tetrahedra sharing a common vertex whose membership is known [38]. For non-tessellated B-reps, PMC is based on Newton-type methods that operate on the parametric geometry describing a solid's boundary [88]. For a good discussion of PMC algorithms we refer the reader to [39, 128].

4.3 Convergence Studies

We now compare the computational properties of SAQ and the characteristic function method (section 2.1.3) for 2D/3D elastostatic problems. We used FCMLab's [95] implementation of the Finite Cell Method (FCM) to solve the 2D/3D elastostatics. FCMLab [95] supports quadtree/octree based integration using the characteristic function method. We used this implementation of characteristic function method for the comparisons. We also extended FCMLab [95] to support quadtree/octree based integration using SAQ.

We assess accuracy and convergence of the two integration schemes in the context of FCM using the energy norm given by

$$E_e = \sqrt{\frac{|U_{ex} - U_{FCM}|}{U_{ex}}} \times 100\,\% \tag{4.6}$$

U_{ex} and U_{FCM} being the exact and FCM computed strain energies respectively.

4.3.1 Annular Ring

In this example, we will consider the plane stress problem over a 2D annular ring as reported in [118]. The geometry, modeling parameters and the reference solution are given in Fig. 4.11. The application of Dirichlet boundary conditions for these problems was done in a weak sense using the Nitsche's method [135, 63, 102, 99] with $\beta = 10^3$. The radial displacement and von Mises stress plot for this plane stress problem obtained using SAQ (with 3 subdivisions) is shown in Fig. 4.12 and Fig . 4.13 respectively.

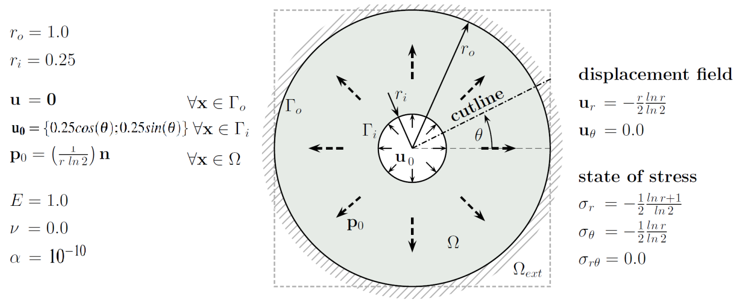

$r_o = 1.0$
$r_i = 0.25$
$\mathbf{u} = \mathbf{0}$ $\forall \mathbf{x} \in \Gamma_o$
$\mathbf{u_0} = \{0.25cos(\theta):0.25sin(\theta)\}$ $\forall \mathbf{x} \in \Gamma_i$
$\mathbf{p}_0 = \left(\frac{1}{r\,ln\,2}\right)\mathbf{n}$ $\forall \mathbf{x} \in \Omega$
$E = 1.0$
$\nu = 0.0$
$\alpha = 10^{-10}$
Γ_o
r_o
r_i
cutline
Γ_i
θ
$\mathbf{u}_0$
$\mathbf{p}_0$
Ω
Ω_{ext}
displacement field
$\mathbf{u}_r = -\frac{r}{2}\frac{ln\,r}{ln\,2}$
$\mathbf{u}_\theta = 0.0$
state of stress
$\sigma_r = -\frac{1}{2}\frac{ln\,r+1}{ln\,2}$
$\sigma_\theta = -\frac{1}{2}\frac{ln\,r}{ln\,2}$
$\sigma_{r\theta} = 0.0$

Figure 4.11: 2D annular ring problem from [118]

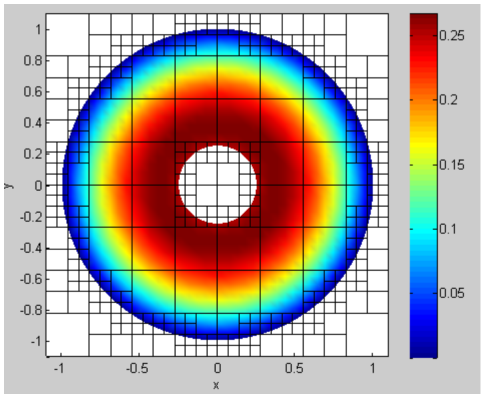


Figure 4.12: Radial displacement plot obtained using SAQ (with 3 subdivisions)

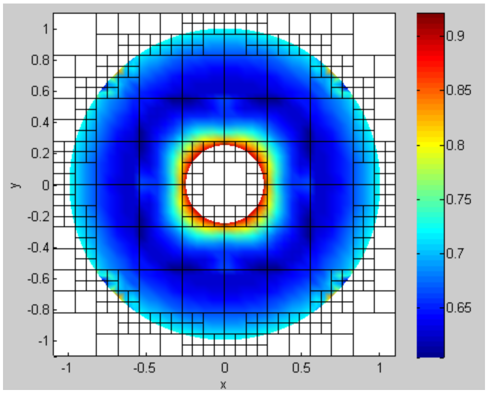


Figure 4.13: von Mises stress plot obtained using SAQ (with 3 subdivisions)

Convergence w.r.t. p-refinement First, we study the convergence w.r.t p-refinement for the two integration schemes. Here, p-refinement refers to increasing the polynomial degree (p) of the basis functions uniformly from 1 to 8 for an 8×8 mesh of quad elements. We study the convergence (as measured in energy norm) of the annular ring problem w.r.t. p-refinement for the two integration schemes using various levels of integration tree (quadtree) refinement. We use a $g \times g$ quadrature in each of the integration cells where $g = p + 4$. Fig. 4.14 shows the convergence of the two schemes w.r.t p-refinement for various integration tree (quadtree) depths. It is clear from the plot that *SAQ requires just 2 subdivisions to provide the accuracy of characteristic's 6 subdivisions.* For lower quadtree depths ($dep < 5$), the integration error dominates to the point that *characteristic function method diverges for higher order basis functions ($p \geq 4$).*

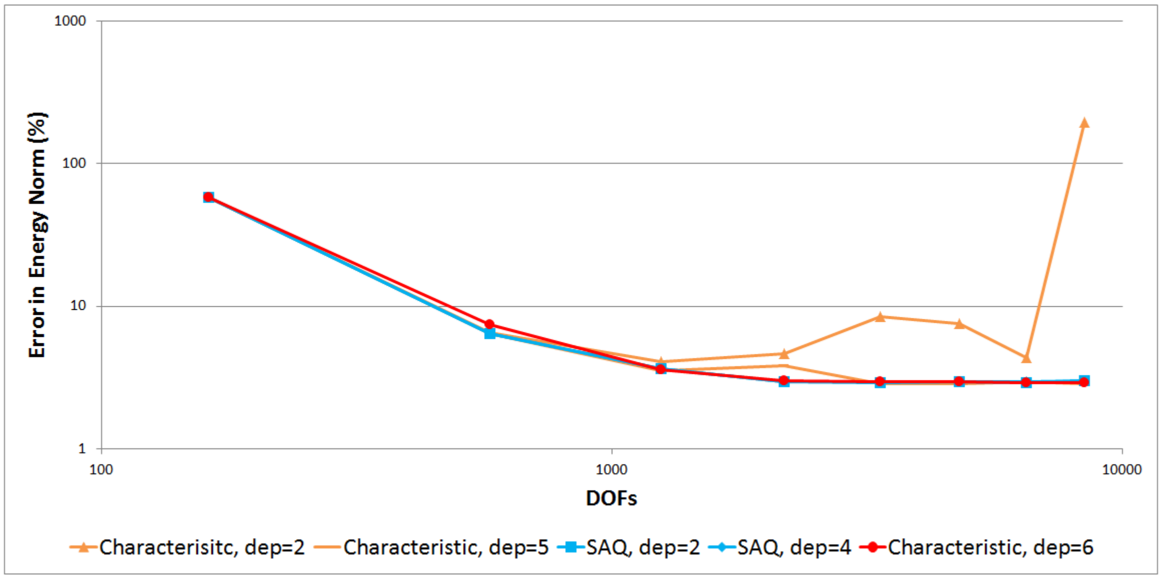


Figure 4.14: Convergence of SAQ and characteristic function method w.r.t. p-refinement in energy norm

From Fig. 4.14 it is clear that the error does not go to zero w.r.t. p-refinement as the convergence also depends on the mesh size. It is important to note that, in this study, we fix the mesh size (8×8) and vary only the degree of basis functions uniformly to achieve p-refinement. In order to drive the error close to zero, it is also required to do an h-refinement in combination with p-refinement so that the underlying finite element function space is rich enough to capture the solution.

Convergence w.r.t. quadtree refinement Next, we study the effect of quadtree refinement on the two schemes. For this study, we use a 4×4 mesh of quad elements of order 4 (578 dofs), for which we vary the quadtree depth (*dep*) from 1 to 4. We use 7×7 quadrature in each of the integration cells. Fig. 4.15 shows the convergence of the two schemes w.r.t quadtree refinement in energy norm. It is very clear from Table. 4.1 that *characteristic function method requires at least 4 subdivisions more than SAQ for the error in energy norm to fall below* 5.8%. We also measured the total time required to compute stiffness matrix and force vector for this problem. Table. 4.1 lists the computer time and error in energy norm for the two methods. From the table we find that *SAQ is at least 17 times faster than characteristic function method* to achieve a given accuracy (of error $\leq$ 5.8%).

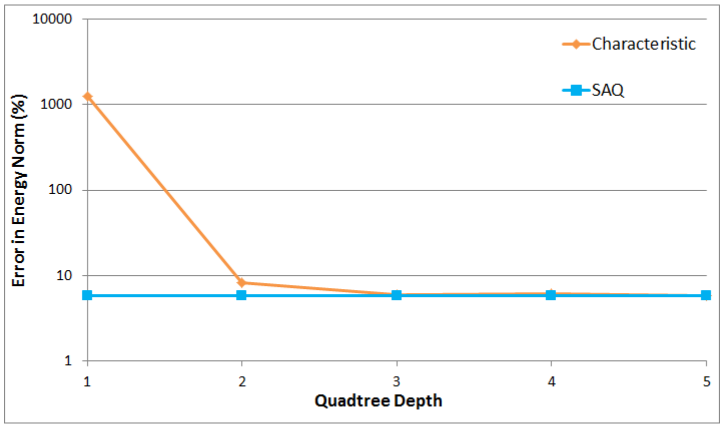


Figure 4.15: Convergence of SAQ and characteristic function method w.r.t. quadtree depth in energy norm

Table 4.1: Rel. error in energy norm and time comparison for SAQ and Characteristic method w.r.t quadtreee refinement

	Characteristic		**SAQ**	
Depth	**Error(%)**	**Time(s)**	**Error(%)**	**Time(s)**
1	1263.7	4.26	5.785	**6.06**
2	8.102	11.74	5.796	15.40
3	5.987	27.24	5.780	35.55
4	6.048	58.31	5.779	71.86
5	5.754	**104.85**	5.790	131.225

Convergence w.r.t. quadrature rule Finally, we study the convergence w.r.t quadrature rule. For this study, we use a 4×4 mesh of quad elements of order 4 (578 dofs), for which we vary the quadrature rule from $g = 1$ to $g = 10$ in each of the integration cells. We use a quadtree depth of 1 for both the schemes. Thus, the total number of integration points for a given quadrature rule (g) is given by $4 \times 4 \times 4 \times g^2 = 64g^2$. Fig. 4.16 shows the convergence of the two schemes w.r.t quadrature rule in energy norm. From the plots, it is clear that for $g \geq 7$, the error in energy norm for SAQ scheme is approximately 6%. However, the characteristic function method diverges for higher order quadrature with the error ranging from 25% to 1230%. *This experiment demonstrates why, unlike characteristic function method, SAQ is a reliable method especially for higher order integration.* As already pointed out in the introduction, classical Gauss quadrature schemes are designed assuming that the integrand is a polynomial of some degree [132]. Characteristic function method applies Cartesian product of this 1D integration scheme to integrate a discontinuous (i.e. non-polynomial) integrand giving the value of a integral of its polynomial approximation. However, polynomial approximation of discontinuous functions tend to oscillate in the neighborhood of the discontinuity as was explained in the section 4.1 (also see Fig. 4.2). Thus, raising the degree of an approximating polynomial, the amplitude of oscillations raises as well [22] (see Fig. 4.2). This explains why the characteristic function method diverges for higher order quadrature in this experiment. However, this problem doesn't arise in SAQ as we are always dealing with a continuous integrand. Moreover, quadrature rule is carefully generated accounting for the type of integrand and geometric domain.

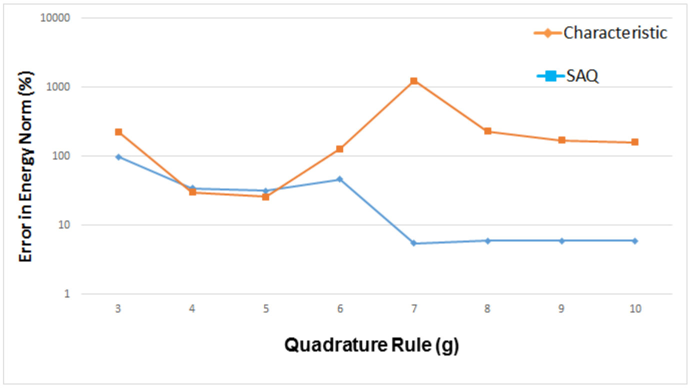


Figure 4.16: Convergence of SAQ and characteristic function method w.r.t. quadrature rule in energy norm

4.3.2 Cylinder under uniaxial compression

In this example, we will compare SAQ with that of the characteristic function method in solving a 3D uniaxial compression problem. A cylinder of radius 1 m and height 0.5 m is subjected to a compressive pressure of 1 MPa on the top face with the bottom fixed (Fig. 4.17). We assume a linear isotropic constitutive relationship with $E = 2.05 \times 10^{11} N/m^2$ and $\nu = 0.29$ and study the convergence of the two integration schemes with respect to h-refinement using FCM. We use triquadratic hexahedron elements (27 node, 81 DOFs) of various mesh sizes as tabulated in Table. 4.2. For both the methods, $3 \times 3 \times 3$ quadrature points were used in all the integration cells. The strong penalty method [53] with a penalty value of 10^{16} was employed to enforce the zero Dirichlet boundary condition on the bottom face. An explicit surface discretization of the top face was introduced on which the traction integral was evaluated. This surface integration mesh is independent of the actual solution mesh and does not introduce additional degrees of freedom [95]. The reference solution ($U_{ex} = 3.47926$ Nm) for comparison was obtained by an overkill FEM solution of the same problem using 313,842 DOFs in SOLIDWORKS [3].

The convergence plot with respect to global h-refinement for various octree depths in computing the stiffness matrix of FCM using the two integration schemes is shown in Fig.

4.18. We find that *SAQ exhibits monotonic convergence whereas characteristic function's convergence is oscillatory in nature.* It is also clear from Fig. 4.18 that the error plot for the characteristic function method oscillates about the error plot for our SAQ (blue lines in Fig. 4.18). Moreover, the characteristic function method requires at least 5 subdivisions (red lines in Fig. 4.18) for the oscillations to settle down and come closer to the accuracy of SAQ's 2 subdivisions.

It is also clear from Fig. 4.19 that *SAQ requires just one subdivision to achieve the same level of accuracy as that of characteristic's five subdivision.* Moreover, Fig. 4.19 also suggests the oscillatory nature of convergence of the characteristic function method w.r.t octree refinement. This is further confirmed from the von Mises stress plots obtained for the two methods for various levels of octree refinement shown in Fig. 4.21 and Fig. 4.22. The reference von Mises stress plot obtained from an overkill FEM solution in SOLIDWORKS [3] is shown in Fig. 4.20. Fig. 4.21 clearly shows spurious stresses at the edges of the top face due to the inability of the characteristic integration scheme to resolve the geometry at lower levels of refinement. On the other hand, from Fig. 4.22 it is clear that SAQ does not exhibit such a phenomena and the von Mises stress plots for all three subdivisions are quite similar to one another. The reason for this is that even though we use a coarse polyhedral approximation (Ω_0) at lower subdivisions (obtained by ray casting in 2D and polygonization in 3D), the incorporation of the shape sensitive term in the moment fitting equations suitably accounts for the difference between Ω_0 and Ω and thereby enabling a good approximation of the moment integral resulting in accurate shape aware integration weights. This in turn implies that, unlike SAQ, *one often requires more than 4 subdivisions for the characteristic function method to produce reliable accurate results for realistic problems with complicated geometry in 3D.* Table. 4.3 lists the time required to compute the stiffness matrix in generating the converged points of the h-refinement convergence graph (Fig. 4.18) for both the methods. From the table we find that SAQ is at least 43-60 times faster than characteristic function method for this problem. For the SAQ scheme to generate the weights, on an average, 72%

of the time is spent in quadrature point generation (via scaled Cartesian product rule), 8% in approximate polyhedra (Ω_0) construction, 18.5% in setting up the linear system ($\{\mathbf{M}\}$ and $[\mathbf{A}]$) and 1.5% in solving the linear system. However, by using random quadrature point generation schemes (as discussed in section 3.6.5) the total time for SAQ can be reduced even further. This suggests that, unlike the characteristic function method, SAQ provides a scalable/viable solution for integration over complex 3D domains.

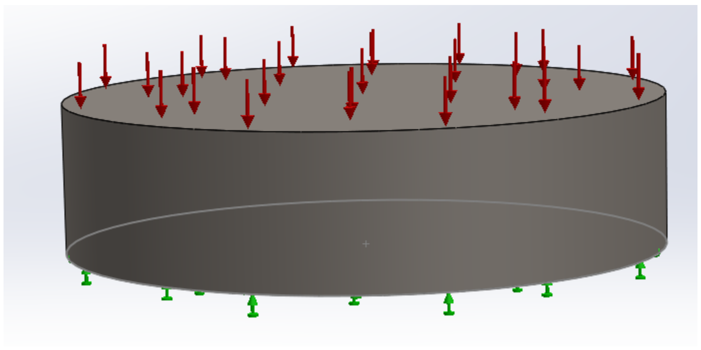


Figure 4.17: Cylinder under uniaxial compression

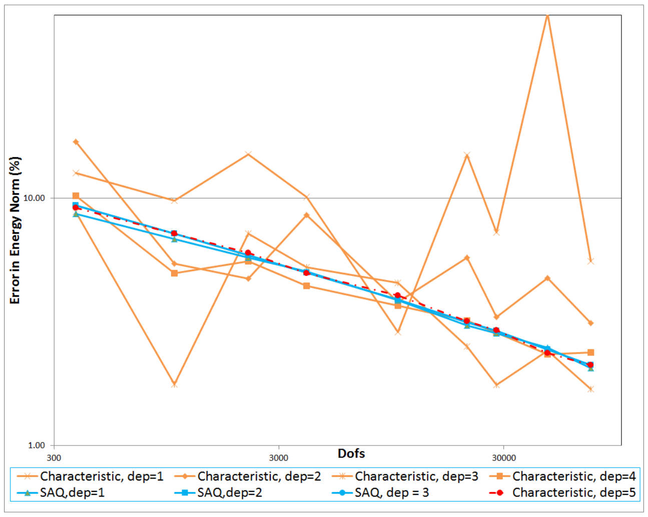


Figure 4.18: Convergence w.r.t h-refinement for the cylinder problem

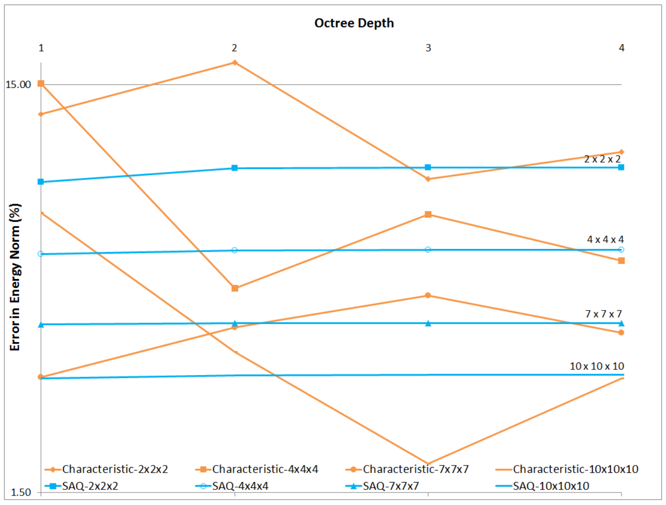


Figure 4.19: Convergence w.r.t octree refinement for the cylinder problem

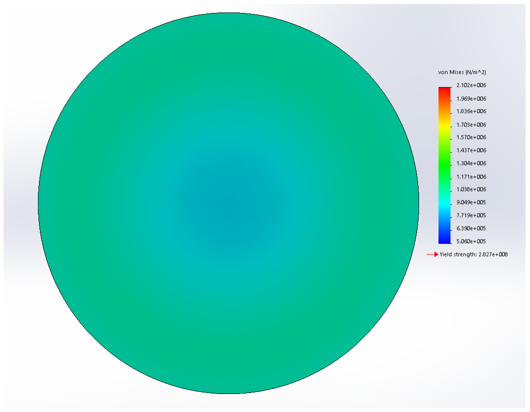


Figure 4.20: von Mises stress plot (top face) for the uniaxial cylinder problem obtained from an overkill FEM solution (with 313,842 DOFs) in SOLIDWORKS [3]

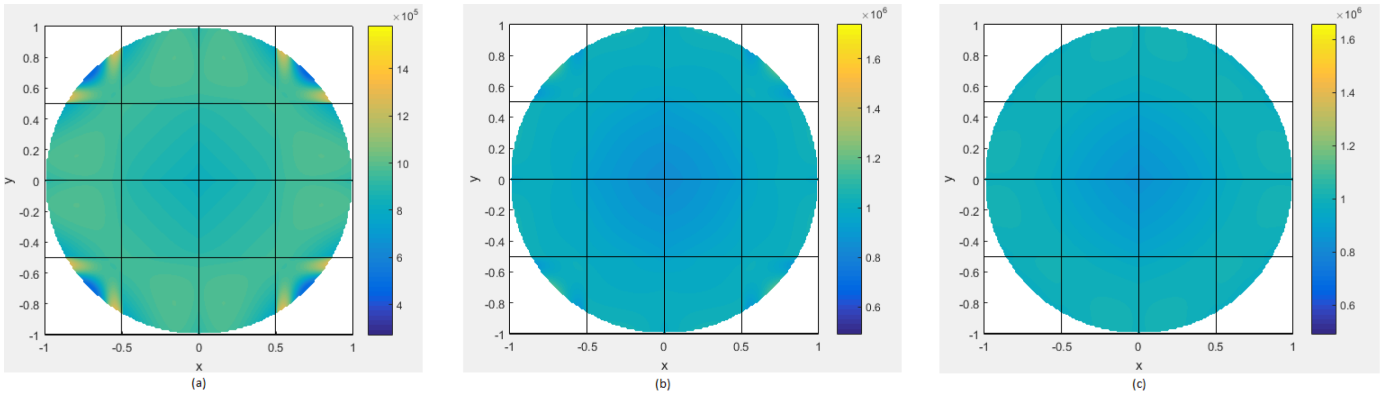


Figure 4.21: von Mises stress plot (top face) for the uniaxial cylinder problem obtained from (a) one (b) two and (c) three octree subdivisions using the characteristic integration scheme (4x4x4 mesh) in FCMLab [95]

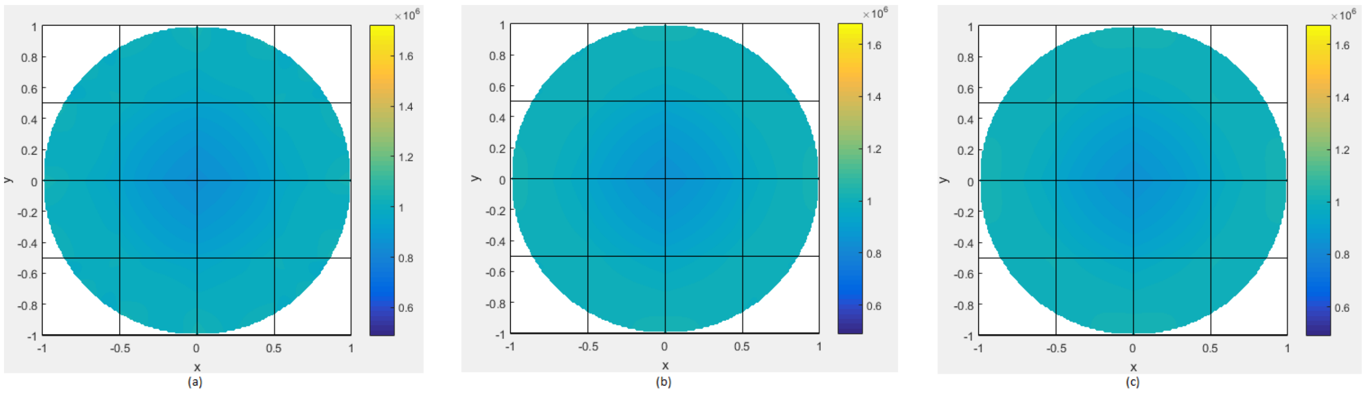


Figure 4.22: von Mises stress plot (top face) for the uniaxial cylinder problem obtained from (a) one (b) two and (c) three octree subdivisions using the SAQ scheme (4x4x4 mesh) in FCMLab [95]

Table 4.2: Mesh sizes and corresponding DOFs for the uniaxial compression problem

Grid Size	**DOFs**
2 x 2 x 2	375
3 x 3 x 3	1,029
4 x 4 x 4	2,187
5 x 5 x 5	3,993
7 x 7 x 7	10,125
9 x 9 x 9	20,577
10 x 10 x 10	27,783
12 x 12 x 12	46,875
14 x 14 x 14	73,167

Table 4.3: Time required to generate the stiffness matrix in generating the converged points for various mesh sizes

Grid Size	Time (s)		Speed up
	Characteristic	**SAQ**	
2 x 2 x 2	714.13	11.90	60.03
3 x 3 x 3	1540.32	29.30	52.57
4 x 4 x 4	2733.53	54.35	50.29
5 x 5 x 5	4199.05	82.41	50.96
7 x 7 x 7	8590.39	183.82	46.73
9 x 9 x 9	14291.23	327.35	43.66
10 x 10 x 10	19705.23	411.55	47.88

4.4 Summary

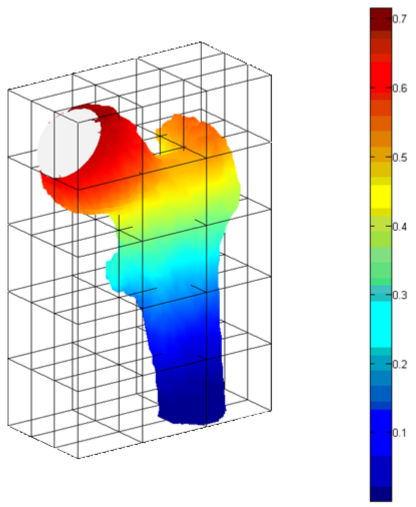


Figure 4.23: Displacement norm plot of a human femur model problem as obtained by employing SAQ in FCMLab with $E = 15.2 \times 10^3$ N/mm^2 and $\nu = 0.4$. An hip contact pressure of 1 MPa is applied on the non-boundary conforming discretization using a separate surface discretization of the hip cap.

We successfully demonstrated the application of SAQ in the context of FCM. We implemented the method in FCMLab [95] and performed convergence studies for 2D plane stress and 3D isotropic elasticity. The experimental results clearly indicates that, to achieve a given accuracy, SAQ requires far fewer subdivisions / less time compared to the standard integration scheme (characteristic function approach) used in FCM. In addition, SAQ is found to posses superior order of convergence compared to the characteristic function approach. Moreover, the characteristic function approach is found to produce erroneous results at lower levels of refinement for certain problems in addition to exhibiting oscillatory convergence. Reliable and consistent results from characteristic function method can be expected only with 4 or more subdivisions. This is essentially because of the inability of the method to resolve the geometry at lower grid resolutions. This makes the characteristic function approach computationally very expensive for problems with complicated geometry such as the human femur model shown in Fig. 4.23. On the other hand, the introduction of shape correction factor in the computation of moments in SAQ makes the computed quadrature weights shape aware and thus enables the weights to adapt to the domain of integration automatically. This explains why SAQ produces accurate results even at lower grid resolution. SAQ is found to exhibit monotonic convergence for integration over simple geometries and we anticipate similar behavior even for more complex geometries. Moreover, SAQ offers a lot of flexibility in the choice of quadrature points and in the choice of basis functions as its formulation is based on the moment fitting equations. We also note that SAQ was formulated irrespective of how the original domain Ω is represented, and it can be used with or without an integration mesh. This suggests that SAQ is not only applicable to FCM but also to other immersed boundary and meshfree methods.

The introduction of shape correction factors results in an approximation error in computation of moments and hence in the integral computed using SAQ. Hence, there is a need to develop *a priori* and *a posteriori* error estimates for SAQ which will be the subject of the next chapter.

Chapter 5

A priori and *a posteriori* error estimation of SAQ

5.1 Background

Adaptive numerical integration refers to the process of approximating the integral of a given function to a specified precision by adaptively subdividing the integration domain into smaller domains over which a set of local quadrature rules are applied [98]. All adaptive algorithms for numerical integration have a general structure that consists of the following four main steps [61]

1. Choose some subregion from a set of subregion(s)

2. Subdivide the chosen subregion(s)

3. Apply the integration rule to the new subregions; update the subregion set

4. Update global integral and error estimates; check for convergence

The corresponding components that distinguish these adaptive algorithms are (1) type of representation used for subregion set, (2) a subdivision strategy and (3) an integration rule and an error estimator. Error estimation is a key step in this process that enables adaptivity. Error estimation in one dimension (interval) is very well understood and many estimators have been developed since 1960s [98]. One popular technique is to estimate the error in the actual integral $\int_a^b f(x)dx$ as

$$e \approx |I_{n_1}[a,b] - I_{n_2}[a,b]| \tag{5.1}$$

where I_{n_1} and I_{n_2} are integrals computed using quadrature rule of different degrees. This error estimate is based on the assumption that if the estimate I_{n_2} is a better approximation of the integral than I_{n_1}, then the difference between both estimates will be a good estimate of the difference between I_{n_1} and the actual integral [61]. Another class of error estimators are based on the following

$$e \approx |I_n^{(m_1)}[a,b] - I_n^{(m_2)}[a,b]| \tag{5.2}$$

where $I_n^{(m_1)}$ and $I_n^{(m_2)}$ are integrals computed using two different quadrature rule (such as Gauss quadrature and Clenshaw-Curtis [98]) of the same degree (n).

Other types of estimator work on analytically bounding the error in the quadrature using some form of Taylor series expansion and invoking the Taylor's Remainder Theorem. These type of estimates (known as “a priori error estimate”) are useful to know the order of convergence of the quadrature scheme. These estimates involve higher order derivatives of the integrand that are often not very easy to compute as the function f is not known “a priori”. However, some authors [44, 57] advocate approximating these derivatives in order to come up with a computable error estimate. Such error estimates can also be used to drive the adaptive integration process. Many error estimators have been developed in the last 50 years for the 1D interval and we refer the reader to [98] for details. Likewise, many error estimates have been developed for a variety of regions such as hyper-rectangular regions [28], triangles [32] and tetrahedra [27].

In short, adaptive quadrature over one and two dimensions is very well understood and in fact many commercial packages such as QUADPACK [101] and MATLAB [90] supports adaptive quadrature. Specifically, MATLAB [90] provides two functions *int* and *quad2d* for adaptive quadrature over intervals and planar regions respectively. These two functions also return an upper bound of the relative/absolute error in the final computed integral.

Much of the work on error estimators that is discussed above and in the literature were developed in the context of integrand adaptivity over simple domains. However, to the best of our knowledge, not much has been done with respect to shape adaptivity. Hence, in this chapter we will focus on estimating the error in SAQ w.r.t. shape adaptivity. Specifically, we will develop error estimators for SAQ to estimate the error due to approximation of moments (see Eq.(3.4)). Hence, we will develop an *a priori* and *a posteriori* error estimator for SAQ that estimates the error in the integral due to moment approximations. However, to begin with, we will discuss all the possible sources of error in SAQ and ways to estimate or eliminate the error. Then, we will develop *a priori* and *a posteriori* error estimates using Taylor's Remainder Theorem for SAQ with SSA based correction factors. Error estimates for other types of correction factors (including TSA) can be derived based on the ideas presented in this chapter.

5.2 Sources of error

In this section we will look at all the possible sources of error in SAQ and ways to avoid and/or estimate the errors. We will refer to the algorithmic steps presented in section 3.4 in order to identify the sources of error in SAQ.

5.2.1 Insufficient/inappropriate choice of basis functions

When the choice of basis functions $\{b_i\}_{i=1}^{m}$ (Step 2) is inappropriate/insufficient then the integrand $f \notin span(\{b_i\}_{i=1}^{m})$ resulting in integrand error. In classical FEM [25] and its variations [66, 26], it is not difficult to determine the degree or nature of the integrand exactly. In such cases, this problem can be completely avoided by choosing appropriate number/type of basis functions. In certain other applications such as [85, 72, 73], it might be difficult to estimate/know this beforehand and hence can significantly affect the integral accuracy. However, as we have already stated in the previous section, estimating integrand error is a well understood subject and hence will not be the focus here.

5.2.2 Insufficient number of domain quadrature points

Choosing fewer domain quadrature points (Step 3) than the number of basis functions can result in an over-determined linear system i.e. $m > n$ in Eq.(3.2). This can result in error due to linear least squares [122] in Step 7 of the algorithm. This error can be completely eliminated by making $[A]$ a square invertible matrix by choosing number of quadrature points (n) to be equal to the number of basis functions (m).

5.2.3 Inappropriate location of domain quadrature points

In general, choosing domain quadrature points (Step 3) outside Ω or Ω_0 is known to affect the accuracy of SAQ (see section 4.5). In general, choosing random points such that it lies completely inside both Ω and Ω_0 can help eliminate this error completely. However, choosing quadrature points that lie close to each other can affect the condition number of $[A]$ thereby deteriorating the accuracy of the method. One simple way to eliminate this problem is to first check the conditioning of $[A]$ before solving the linear system. This way, one can throw away the generated quadrature points if $[A]$ is poorly conditioned and generate a new set of points until the condition number improves. Alternatively, one could use approximate Fekete points as suggested by Sommariva and Vianello [15] to completely avoid this problem.

5.2.4 Inaccurate Moment Computations

Inaccuracy in moment computations (Eq.(3.9)) can be attributed to the following :

a. **Inaccurate distance and/or measure computations** Distance computations (V_N) are required in the calculation of moments using SSA. Likewise measure computations are required for shape correction factors based on TSA. The accuracy of distance/measure computation is dependent on the representation of Ω and the algorithm employed. One simple way to compute this distance is by a simple ray casting algorithm [119] as illustrated in Fig. 5.1.

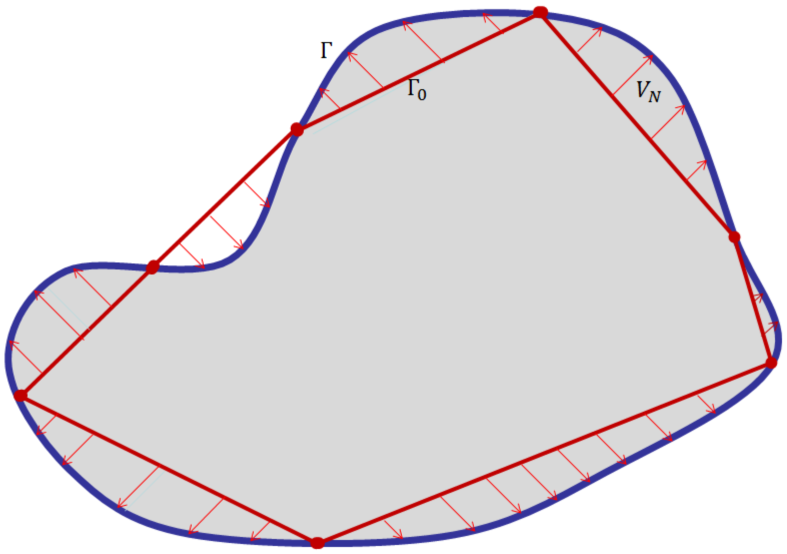


Figure 5.1: Design velocity is simply the distance to the boundary in a direction normal to Γ_0

Likewise, measure of small features can be computed accurately using closed form expressions or by first-order SSA. If we know that ζ is the absolute maximum error in distance computations and η is the absolute maximum error in measure computations, then the first-order error in the integral of i^{th} basis function (dM_i) due to inaccurate distance/measure computations can be shown to be bounded above by the following :

$$\begin{aligned} |dM_i^d| &\leq \zeta \left| \int_{\Omega_0} b_i(\mathbf{X}) d\Omega \right| + \eta \sum_{j=1}^{n_s} b_i(\mathbf{X_j}) \\ &= \zeta \left| \int_{\Gamma_0} \beta_X^i(\mathbf{X}) N_X d\Gamma_0 \right| + +\eta \left| \sum_{j=1}^{n_s} b_i(\mathbf{X_j}) \right| \text{ (using divergence theorem)} \end{aligned} \quad (5.3)$$

b. **Boundary integration error** The evaluation of moments (Eq.(3.12)) is essentially an integral over the polygonal/polyhedral boundary (Γ_0). Unlike the first term in Eq.(3.12), the second term is hard to evaluate due to the presence of the velocity term V_N that in turn depends on the complexity of Ω. Thus, one often requires 1D/2D integrand adaptivity to accurately integrate the velocity term. However, fortunately, integrand adaptivity is very well understood as already stated in section 5.1 and these techniques can be employed directly to estimate the boundary integral error in the evaluation of moments. If, e_i^b is the upper bound of the error in integral of b_i due to boundary integration, then

$$|dM_i^b| \leq e_i^b \quad (5.4)$$

c. **Error due to n^{th}-order Taylor series approximation** In SAQ, we use n^{th}-order Taylor series approximation (using sensitivities such as SSA or TSA) to compute the shape correction factors. This in turn enables us to estimate the moments over Ω using a simplified domain Ω_0. This error can be estimated using Taylor's Remainder Theorem as we will see in the next section. If, e_i^s is the upper bound of the error due to n^{th}-order Taylor series approximation then

$$|dM_i^s| \leq e_i^s \quad (5.5)$$

Thus, the total error in the computation of i^{th} moment is bounded by

$$|dM_i| \leq |dM_i^d + dM_i^b + dM_i^s| \leq \zeta \left| \int_{\Gamma_0} \beta_X^i(\mathbf{X}) N_X d\Gamma_0 \right| + \eta \left| \sum_{j=1}^{n_s} b_i(\mathbf{X_j}) \right| + e_i^b + e_i^s \quad (5.6)$$

and an *a posteriori* error estimate of the computed integral (using SAQ) can be estimated as (see section. 5.3 for details of this derivation)

$$|e| \leq \{\mathbf{dM}\}^T \{\mathbf{z}\} \quad (5.7)$$

where $\mathbf{z}$ is the adjoint solution obtained by solving $[\mathbf{A}]^T\{\mathbf{z}\} = \{f(\mathbf{X}_i)\}$ and $\{\mathbf{dM}\}$ is the estimate of the total error in moments.

Henceforth, we will assume $\zeta = \eta = 0$ and $e_i^b = 0$ and so $|dM_i| \leq e_i^s$. In other words, we will ignore/eliminate the error due to boundary integration/distance computations and consider only the error in moments due to Taylor series approximation. Further, for simplicity, we will assume that the shape correction factors were estimated only using first-order shape sensitivity analysis (SSA) [75] and that Ω_0 is homeomorphic to Ω (i.e. $C_i^T = 0$). Estimates for other types/order of sensitivity analysis can be easily derived following the ideas/theorems that we present in the next section.

5.2.5 Fixed precision arithmetic

Numerical errors due to fixed precision arithmetic is a common problem in any numerical process and SAQ is no exception (Steps 4-8). There are different ways to estimate/eliminate these errors and we will refer the reader to [122] for details.

The focus of this chapter is to develop an *a priori* and a posteriori error estimate of SAQ due to first-order Taylor series approximation of the moments using SSA. Hence, for the sake of simplicity, we will neglect all other errors in our analysis.

5.3 *A priori* error analysis

In this section, we will develop an *a priori* error estimate for SAQ due to first-order Taylor series approximation of moments using Shape Sensitivity Analysis (SSA) [75]. In other words, we assume that the shape correction factor was derived purely using first-order SSA and we also assume that the simplified Ω_0 is always homeomorphic to Ω (i.e. $C_i^T = 0$). We will provide numerical results to support the theoretical estimates. We neglect all other sources of error (including numerical errors) as discussed in the previous section.

Theorem 5.3.1 (Taylor's Remainder Theorem)**.** *Let $M_i : [0, \alpha] \to \mathbb{R}$ be a $\mathscr{C}^r$ continuous function and for $k \in \mathbb{Z}^+$ let $P_k(t) = M_i(0) + t.M_i'(0) + \frac{t^2}{2!}M_i''(0) + ... + \frac{t^k}{k!}M_i^k(0)$ $(r \geq k+1)$. Then, for some $\epsilon \in [0, \alpha]$ we have the following error bound*

$$|E_k(t)| = |M_i(t) - P_k(t)| \leq \frac{|t^{k+1}|}{(k+1)!}|M_i^{k+1}(\epsilon)| \tag{5.8}$$

Proof. See proof in Appendix C. □

Corollary 5.3.1.1. *For a linear approximation $P_1(t) = M_i(0) + t.M_i'(0)$ we have the following error bound*

$$|E_1(t)| = |M_i(t) - P_1(t)| \leq \frac{t^2}{2!}|M_i''(\epsilon)| \tag{5.9}$$

Proof. Follows directly from Theorem 5.3.1 by setting $k = 1$ □

Referring to the moments in Eq.(3.2), we know $M_i(\Omega) = \int_\Omega b_i(\mathbf{x})d\Omega$. However, this can also be written as $M_i(\Omega) = M_i(\Omega_0, \hat{\mathbf{V}}, t)$. For a given initial domain (Ω_0) and design velocity vector $(\hat{\mathbf{V}})$, we can write $M_i(\Omega) = M_i^{\Omega_0, \hat{\mathbf{V}}}(t)$. For the sake of simplicity, we will drop the superscripts and henceforth denote $M_i(t) \equiv M_i^{\Omega_0, \hat{\mathbf{V}}}(t)$

Theorem 5.3.2. *Let $\Omega \subset \mathbb{R}^d$ and $\Omega_0 \subset \mathbb{R}^d$ be two r-sets [108] such that there exist a linear homeomorphic mapping $\mathbf{T} : \Omega_0(\mathbf{X}) \times [0, 1] \to \Omega(\mathbf{x})$ given by $\mathbf{T}(\mathbf{X}, t) = \mathbf{x} = \mathbf{X} + t\hat{\mathbf{V}}(\mathbf{X})$. Here, $\hat{\mathbf{V}}(\mathbf{X}) \subset \mathbb{R}^d$ is the velocity vector defined in the neighborhood of Ω_0 and is C^k continuous* *with $k \geq 1$. If $b_i : \Omega \cup \Omega_0 \to \mathbb{R}$ is a $\mathscr{C}^r$ continuous function with $r \geq 2$, $M_i(t) = \int_\Omega b_i(\mathbf{x})d\Omega$ and $M_i^*(t) = \int_{\Omega_0}[b_i(\mathbf{X}) + t\nabla.(b(\mathbf{X})\mathbf{V}(\hat{\mathbf{X}}))]d\Omega_0$ then for some $\epsilon \in [0,1]$ we have*

$$|M_i(t) - M_i^*(t)| \leq \frac{t^2}{2!}\left|\int_\Omega \left(\nabla.\left[\nabla.(b_i\hat{\mathbf{V}}(\mathbf{x}))\hat{\mathbf{V}}(\mathbf{x})\right] - \nabla.\left[b_i(\mathbf{x})\nabla\hat{\mathbf{V}}(\mathbf{x}).\hat{\mathbf{V}}(\mathbf{x})\right]d\Omega\right)\right|_{t=\epsilon}\right| \quad (5.10)$$

Proof. Since, b_i is at least $\mathscr{C}^2$ continuous it can be proved that $M_i(t)$ is at least C^2 continuous (see Theorem 5.6 in [62]). Then, from Corollary 5.3.1.1 for any $\epsilon \in [0,1]$ we have the following

$$|M_i(t) - M_i^*(t)| \leq \frac{t^2}{2!}\left|\frac{d^2M_i}{dt^2}\right|_{t=\epsilon} \quad (5.11)$$

However, $\frac{d^2M_i}{dt^2} = \int_\Omega \nabla.\left[\nabla.(b_i\hat{\mathbf{V}}(\mathbf{x}))\hat{\mathbf{V}}(\mathbf{x})\right] - \nabla.\left[b_i(\mathbf{x})\nabla\hat{\mathbf{V}}(\mathbf{x}).\hat{\mathbf{V}}(\mathbf{x})\right]d\Omega$ (see proof in Appendix A) and the result follows plugging in this in Eq.(5.11). □

Theorem 5.3.3. *Let $\Omega \subset \mathbb{R}^d$ and $\Omega_0 \subset \mathbb{R}^d$ be two r-sets [108]. Let $\{b_i(\mathbf{x})\}_{i=1}^n$ be a set of C^k continuous basis functions of degree p with $k \geq 2$ ($b_i : \Omega \cup \Omega_0 \to \mathbb{R}$). Let $f : \Omega \to \mathbb{R}$ be a ($\mathscr{C}^k$ continuous function) in span($\{b_i(\mathbf{x})\}_{i=1}^m$). Let $\{\mathbf{x_i}\}_{i=1}^n \subset \Omega \cap \Omega_0$ be a set of quadrature points chosen to render the moment fitting matrix $\mathbf{A}$ invertible. If $\{\mathbf{w}\}$ and $\{\mathbf{w}^*\}$ be the quadrature weights computed using the moments $M_i(t) = \int b_i(\mathbf{x})d\Omega$ and $M_i^*(t) = \int_{\Omega_0} b_i(\mathbf{x}) + \nabla.(b_i(\mathbf{x})\hat{\mathbf{V}}).d\Omega_0$ respectively then the error $|e(q^*)| = \left|\int_\Omega f d\Omega - \sum_{i=1}^n W_i^* f((\mathbf{x_i}))\right|$ in integrating f over Ω using the shape aware quadratures $q^* = \{x_i, w_i*\}_{i=1}^n$ is bounded by*

$$|e(q^*)| \leq (V_{max})^2\lambda_{max}||\mathbf{dM}||_2||\mathbf{f}(\mathbf{x})||_2 \quad (5.12)$$

where $dM_i = \frac{(B_i+2DE_i)}{2}$ is a constant that depends only on the basis functions, gradient of the design velocity and geometry of the domain, λ_{max} is the maximum eigenvalue of $\mathbf{A}^{-T}\mathbf{A}^{-1}$ and $V_{max} = \sup_{\mathbf{X}\in\Omega_0}\left\|\hat{\mathbf{V}}(\mathbf{X})\right\|_2$

Proof. By definition, the weights $\mathbf{w}$ and $\mathbf{w}^*$ are obtained by solving the moment fitting equations as

$$
\begin{aligned}
\{\mathbf{w}\} &= [\mathbf{A}]^{-\mathbf{1}}\{\mathbf{M}\} && (5.13)\\
\{\mathbf{w}^*\} &= [\mathbf{A}]^{-\mathbf{1}}\{\mathbf{M}^*\} && (5.14)
\end{aligned}
$$

Subtracting the two equations we get

$$
\begin{aligned}
\{\mathbf{w}\} - \{\mathbf{w}^*\} &= [\mathbf{A}]^{-\mathbf{1}}\Big(\{\mathbf{M}\} - \{\mathbf{M}^*\}\Big)\\
\{\mathbf{\Delta w}\} &= [\mathbf{A}]^{-\mathbf{1}}\{\mathbf{\Delta M}\} && (5.15)
\end{aligned}
$$

□

$\{\mathbf{w}\}$ *corresponds to the exact quadrature rule if the moments* $\mathbf{M}$ *were computed exactly (without any numerical errors). Thus, we have the following*

$$
\begin{aligned}
|e(q^*)| &= \left| \int_\Omega f d\Omega - \sum_{i=1}^{n} w_i^* f((\mathbf{x_i}))\right|\\
&= \left| \sum_{i=1}^{n} w_i f((\mathbf{x_i})) - \sum_{i=1}^{n} w_i^* f(\mathbf{x_i})\right|\Big|\\
&= \Big|\{\mathbf{w} - \mathbf{w}^*\}^T\{\mathbf{f}(\mathbf{x})\}\Big|\\
&= \Big|\{\mathbf{\Delta w}\}^T\{\mathbf{f}(\mathbf{x})\}\Big|\\
&= \Big|\{[\mathbf{A}]^{-\mathbf{1}}\{\mathbf{\Delta M}\}\}^T\{\mathbf{f}(\mathbf{x})\}\Big| && (5.16)
\end{aligned}
$$

where the last step was obtained by substituting for $\{\mathbf{\Delta w}\}$ *from Eq.(5.15). Now, normalizing* $\mathbf{\Delta M}$ *and applying Cauchy-Schwartz inequality [111] we get*

$$\begin{aligned}|e(q^*)| &= \left| ||\mathbf{\Delta M}||_2 \left\{ [\mathbf{A}]^{-\mathbf{1}} \frac{\{\mathbf{\Delta M}\}}{||\mathbf{\Delta M}||_\mathbf{2}} \right\}^T \{\mathbf{f}(\mathbf{x})\} \right| \\ &\leq ||\mathbf{\Delta M}||_2 \left\| [\mathbf{A}]^{-\mathbf{1}} \frac{\{\mathbf{\Delta M}\}}{||\mathbf{\Delta M}||_\mathbf{2}} \right\|_\mathbf{2} ||\mathbf{f}(\mathbf{x})||_\mathbf{2} \end{aligned} \tag{5.17}$$

Now, applying the eigen value lemma in Appendix D we have

$$|e(q^*)| \leq ||\mathbf{\Delta M}||_2 \lambda_{max} ||\mathbf{f}(\mathbf{x})||_2 \tag{5.18}$$

where λ_{max} is the maximum eigenvalue of $\mathbf{A}^{-T}\mathbf{A}^{-1}$.

From Theorem 5.3.2, we have the following

$$|M_i(t) - M_i^*(t)| \leq \frac{t^2}{2!} \left| \int_\Omega \left(\nabla. \left[\nabla.(b_i(\mathbf{x})\hat{\mathbf{V}}(\mathbf{x}))\hat{\mathbf{V}}(\mathbf{x}) \right] - \nabla. \left[b_i(\mathbf{x}) \nabla \hat{\mathbf{V}}(\mathbf{x}).\hat{\mathbf{V}}(\mathbf{x}) \right] d\Omega \right) \Big|_{t=\epsilon} \right| \tag{5.19}$$

In our application, we will always assume $\hat{\mathbf{V}} = V_n \mathbf{n}$ (i.e. normal perturbations). Thus, multiplying and dividing the above equation by $V_{max} = \sup_{\mathbf{X} \in \Omega_0} \left\| \hat{\mathbf{V}}(\mathbf{X}) \right\|_2 = \sup_{\mathbf{X} \in \Omega_0} V_N(X)$ and introducing the notation $\mathbf{V} = \frac{\hat{\mathbf{V}}}{V_{max}}$ we have the following :

$$|M_i(t) - M_i^*(t)| \leq \frac{(t.V_{max})^2}{2!} \left| \int_\Omega \left(\nabla. \left[\nabla.(b_i \mathbf{V}(\mathbf{x}))\mathbf{V}(\mathbf{x}) \right] - \nabla. \left[b_i(\mathbf{x}) \nabla \mathbf{V}(\mathbf{x}).\mathbf{V}(\mathbf{x}) \right] d\Omega \right) \Big|_{t=\epsilon} \right| \tag{5.20}$$

where we have used the fact that $V_{max} \geq \sup_{\mathbf{x} \in \Omega} V_n(\mathbf{x}(t))$ for any $t \in [0, 1]$.

Now, applying divergence theorem and using the fact that $\mathbf{V}.\mathbf{n} \leq 1$ and $t \leq 1$, we get

$$\begin{aligned}
|M_i(t) - M_i^*(t)| &\leq \frac{(t.V_{max})^2}{2}\left|\int_\Gamma \left(\left[\nabla.(b_i\mathbf{V}(\mathbf{x}))\mathbf{V}(\mathbf{x}).\mathbf{n}\right] - \left[b_i(\mathbf{x})\nabla\mathbf{V}(\mathbf{x}).\mathbf{V}(\mathbf{x})\right].\mathbf{n}d\Gamma\right)\bigg|_{t=\epsilon}\right| \\
&\leq \frac{(V_{max})^2}{2}\int_\Gamma \left|\nabla.(b_i\mathbf{V}(\mathbf{x}))\right| + |b_i(\mathbf{x})||\mathbf{n}^T\nabla\mathbf{V}\mathbf{n}|d\Gamma\bigg|_{t=\epsilon} \\
&\leq \frac{(V_{max})^2}{2}\int_\Gamma \|\nabla(b_i(\mathbf{x}))\|_2\,\|\mathbf{V}(\mathbf{x})\|_2 + |b_i(\mathbf{x})|\,|\nabla.(\mathbf{V}(\mathbf{x}))| + |b_i(\mathbf{x})|\,\|\nabla V(\mathbf{x})\|_2\,d\Gamma
\end{aligned} \tag{5.21}$$

However, $|\nabla.(\mathbf{V}(\mathbf{x}))| \leq \|\nabla V(\mathbf{x})\|_2 \leq \|\nabla V_1(\mathbf{x})\|_2^2 + \|\nabla V_2(\mathbf{x})\|_2^2 + \|\nabla V_3(\mathbf{x})\|_2^2$ *and* $||\mathbf{V}|| \leq \mathbf{1}$. *Further, since* $b_i(\mathbf{x})$ *is* $\mathscr{C}^2$ *continuous,* $\int_\Gamma \|\nabla b_i(\mathbf{x})\|_2 \leq B_i < \infty$. *This is because* $b_i \in BV(\Omega)$ *and so* $\int_\Gamma \|\nabla b_i(\mathbf{x})\|_2$ *is finite and equal to the total variation. Likewise, since* $\mathbf{V}(\mathbf{x})$ *is* $\mathscr{C}^1$ *continuous,* $\int_\Omega \|\nabla V(\mathbf{x})\|_2\, d\Omega \leq \int_\Omega [\|\nabla V_1(\mathbf{x})\|_2^2 + \|\nabla V_2(\mathbf{x})\|_2^2 + \|\nabla V_3(\mathbf{x})\|_2^2] \leq D < \infty$. *Also,* $b_i(\mathbf{x})$ *is a continuous function defined on a compact set and therefore it is bounded as well i.e.* $b_i(\mathbf{x}) \leq E_i < \infty$. *Thus, substituting the bounds* B_i, D *and* E_i *in Eq.(5.21) we get :*

$$|M_i(t) - M_i^*(t)| \leq \frac{(V_{max})^2}{2}(B_i + 2DE_i)$$

Now, introducing the notation $dM_i = \frac{(B_i+2DE_i)}{2}$ *we get*

$$|\Delta M_i(t)| \leq (V_{max})^2 |dM_i|$$

or

$$||\mathbf{\Delta M}||_2 \leq (V_{max})^2 ||\mathbf{dM}||_2 \tag{5.22}$$

Substituting Eq.(5.22) in Eq.(5.18) we get the desired result.

Theorem 5.3.4. *SAQ converges quadratically w.r.t. uniform (integration) mesh refinement i.e. there exists a constant C independent of the integration mesh size h such that*

$$|e(q^*)| \leq Ch^2 \tag{5.23}$$

provided the maximum normal velocity V_{max} always satisfies the relation $V_{max} \leq h$

Proof. From Theorem 5.3.3 we have established

$$|e(q^*)| \leq (V_{max})^2 \lambda_{max} ||\mathbf{dM}||_2 ||\mathbf{f}(\mathbf{x})||_2 \tag{5.24}$$

Since f is continuous and is defined over a compact set Ω, it attains its maximum f_{max} for some $\mathbf{x} \in \Omega$ and thus

$$||\mathbf{f}(\mathbf{x})||_2 \leq \sqrt{n} f_{max} \tag{5.25}$$

Likewise, since b_i and $\mathbf{V}$ are both C^2 continuous and are defined over a compact set $\Omega \cup \Omega_0$, we can define dM_{max} as

$$dM_{max} = max_{1 \leq i \leq n} |dM_i| \tag{5.26}$$

and hence

$$||\mathbf{dM}||_2 \leq \sqrt{n} (dM)_{max} \tag{5.27}$$

and now substituting Eq.(5.27) and Eq.(5.25) in Eq.(5.24) we get

$$|e(q^*)| \leq (V_{max})^2 \lambda_{max} n (dM)_{max} f_{max} \tag{5.28}$$

Now, defining a constant $C_1 = n(dM)_{max} f_{max}$ we get

$$|e(q^*)| \leq C_1 \lambda_{max} (V_{max})^2 \tag{5.29}$$

If we always choose Ω_0 homeomorphic to Ω such that $V_{max} \leq h$, we get the following bound

$$|e(q^*)| \leq C_1 \lambda_{max} h^2 \tag{5.30}$$

Here, λ_{max} is the maximum eigenvalue of $(A^T A)^{-1}$ where we assume A to be a real non-singular matrix. In order to bound this let us first bound the entires of $\mathbf{A^T A}$ by the basis function values b_i as

$$\begin{aligned}
(\mathbf{A^T A})_{ij} &= \mathbf{A}_i^T \mathbf{A}_j \\
&\leq ||\mathbf{A}_i||_2 ||\mathbf{A}_j||_2 \text{ (by Cauchy-Schwarz inequality)} \\
&\leq \sqrt{n}(b_i)_{max} \sqrt{n}(b_j)_{max} \\
&= n(b_i)_{max}(b_j)_{max} \\
&\leq n[\underset{k}{max} \underset{x\in\Omega}{sup} [b_k(\mathbf{x})]^2]
\end{aligned} \tag{5.31}$$

The maximum eigenvalue ζ_{max} of $(\mathbf{A^T A})$ can be proved [110] to be bounded above by the maximum row sum i.e.

$$|\zeta_{max}| \leq \underset{i}{max} \sum_{j=1}^{n} |(\mathbf{A^T A})_{ij}| \tag{5.32}$$

Thus, using the above and Eq.(5.31) and the fact that $0 < \zeta_{min} \leq \zeta_{max}$ (as $\mathbf{A^T A}$ is positive definite since $\mathbf{A}$ is non-singular) we have

$$|\zeta_{min}| \leq n^2 [\underset{k}{max} \underset{x\in\Omega}{sup} [b_k(\mathbf{x})]^2] \tag{5.33}$$

However, $\zeta_{min} = \frac{1}{\lambda_{max}}$ and thus we have the following upper bound for λ_{max}

$$|\lambda_{max}| \leq \frac{1}{n^2[\underset{k}{max} \underset{x\in\Omega}{sup} [b_k(\mathbf{x})]^2]} = \Lambda \tag{5.34}$$

here $0 < \Lambda < \infty$ is a constant independent of h. Thus, using this bound in Eq.(5.28) we get the following convergence result

$$|e(q^*)| \leq C_1 \Lambda h^2 = Ch^2 \tag{5.35}$$

where C is a constant independent of h.

□

5.3.1 Example 1 - Quadrant of a Circle

In this example, we will first compute the area of quadrant of a circle (Ω) analytically by applying first-order Shape Sensitivity Analysis (SSA) using a simple right angled triangle as the approximate domain (Ω_0) as shown in Fig. 5.2. Then, we will study the convergence of SAQ for various bivariate polynomials $x^m y^n$ (of order up to 3) over this Ω w.r.t a variety of polygonal approximations (Ω_0).

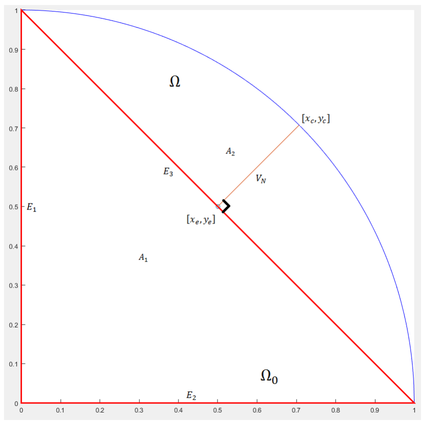


Figure 5.2: Quadrant of a Circle (Ω) and its approximate domain (Ω_0)

From first-order SSA we have the following :

$$\begin{aligned}\int_{\Omega} d\Omega &\approx \int_{\Omega_0} d\Omega_0 + \int_{\Omega_0} V_N d\Omega_0 \\ &= \oint_{E_1} x n_x dt + \oint_{E_2} x n_x dt + \oint_{E_3} [x n_x + V_N(t)] dt \end{aligned} \tag{5.36}$$

However, $x = 0$ on E_1 and $n_x = 0$ on E_2. Thus, the above reduces to the following

$$\int_{\Omega} d\Omega \approx \oint_{E_3} [x n_x + V_N(t)] dt \tag{5.37}$$

The unit outward normal vector to E_3 is $\frac{1}{\sqrt{2}} \left\{ \begin{array}{c} 1 \\ 1 \end{array} \right\}$. Thus, the first term in the above equation reduces to the following

$$\oint_{E_3} x n_x dt = \frac{1}{\sqrt{2}} \int_0^{\sqrt{2}} [1 - \frac{t}{\sqrt{2}}] dt = 0.5 \tag{5.38}$$

The contribution from the second term can be computed by first writing the velocity V_N as a function of the line parameter t. Any point (x_e, y_e) on E_3 can be written as

$$x_e = 1 - \frac{t}{\sqrt{2}};\ y_e = \frac{t}{\sqrt{2}} \tag{5.39}$$

Let (x_c, y_c) be the point of intersection of the normal at (x_e, y_e) and the quadrant boundary. Then,

$$x_c = x_e + \frac{V_N}{\sqrt{2}};\ y_c = y_e + \frac{V_N}{\sqrt{2}} \tag{5.40}$$

Substituting Eq.(5.39) in Eq.(5.40) we get

$$x_c = 1 - \frac{t}{\sqrt{2}} + \frac{V_N}{\sqrt{2}};\ y_c = \frac{t}{\sqrt{2}} + \frac{V_N}{\sqrt{2}} \tag{5.41}$$

However, $y_c = \sqrt{1 - x_c^2} = \sqrt{1 - [1 - \frac{t}{\sqrt{2}} + \frac{d}{\sqrt{2}}]^2}$. Substituting this for y_c in the above and then simplifying we get the following quadratic equation for V_n

$$V_N^2 + \sqrt{2}V_N + [t^2 - \sqrt{2}t] = 0 \tag{5.42}$$

Solving for V_N, we get

$$V_N(t) = \frac{-\sqrt{2} + \sqrt{2 - 4[t^2 - \sqrt{2}]}}{2} \tag{5.43}$$

Then, using this expression in the second term of Eq.(5.37) we get the contribution from this term as

$$\oint_{E_3} V_N(t)\, dt = \oint_{E_3} \frac{-\sqrt{2} + \sqrt{2 - 4[t^2 - \sqrt{2}]}}{2}\, dt \tag{5.44}$$

Now, substituting Eq.(5.38) and Eq. (5.44) in Eq.(5.37), we compute the Area of the quadrant as

$$\begin{aligned} \int_\Omega d\Omega &\approx \oint_{E_3} x n_x dt + \oint_{E_3} V_N(t)]dt \\ &= 0.5 + 0.285398163397448 \\ &\approx \frac{\pi}{4} \end{aligned}$$

Note that the only approximation involved in the above is due to the fixed precision arithmetic involved in the computation of the second boundary integral containing the velocity term (V_N). Thus,

$$\int_\Omega d\Omega = \oint_{E_3} [x n_x + V_N(t)]dt \tag{5.45}$$

Referring to Fig. 5.2, one can easily observe that the first term above corresponds to the area of the approximate domain (A_1) and the second term to the remaining area under the quadrant (A_2). In other words, area of a quadrant can be obtained exactly by applying first-order Taylor series using a triangle (or any polygon homeomorphic to the quadrant) as the approximate domain as long as the boundary integral involving the velocity term can be computed exactly. However, evaluating the boundary integral involving the velocity term (Eq.(5.44)) is non-trivial as it involves square roots. Hence, when computed numerically one will have to use higher order quadrature or adaptive quadrature schemes to compute this integral as demonstrated by Fig. 5.3. We used MATLAB's [90] *int* function for adaptive quadrature and the regular $n - pt$ Gauss quadrature over the boundary of Ω_0 (i.e. line integral) for computing the moment approximations.

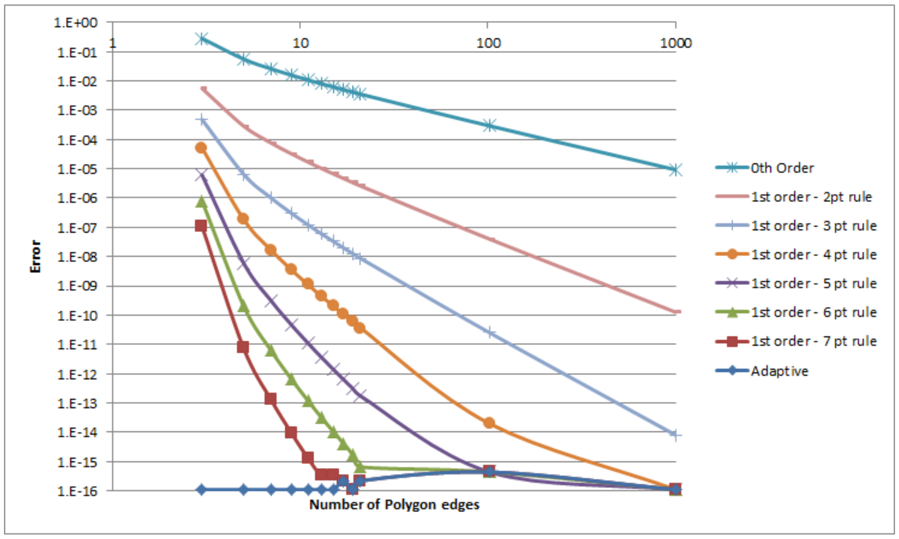


Figure 5.3: Error in the area computations of a quadrant (Ω) using zeroth and first-order Taylor series approximation various polygonal approximations (Ω_0)

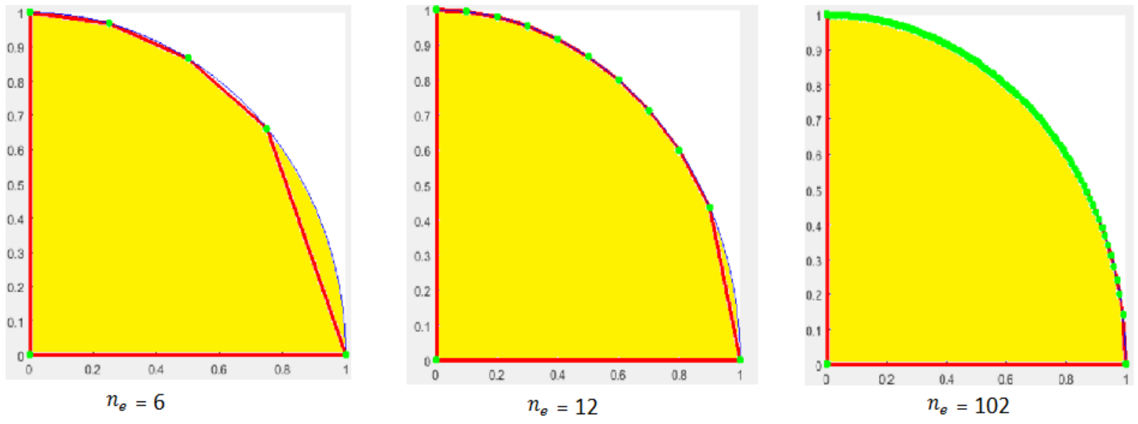


Figure 5.4: Different polygonal approximations (Ω_0) used for computing SAQ over a quadrant domain (Ω). The vertices of the polygon (green dots) lie on the original boundary (Γ)

Fig. 5.3 plots the error in the integral due to first-order moment approximations using various 1D integration rules to compute the boundary integral of Eq.(5.44) for various polygonal approximations. We find that the adaptive quadrature scheme gives the best results as the error is close to machine precision even for the coarsest polygonal approximation (i.e. for the triangle). This observation is quite general and hence can be generalized to any domain of integration in 2D (or 3D) as long as the vertices (or edges) of the approximate polygon (or polyhedron) Ω_0 lie on the original boundary (Γ) as shown in Fig. 5.4. We also superpose the result obtained by omitting the boundary integral term arising from first-order SSA which would then correspond to 0^{th} order Taylor series approximation. We clearly see from Fig. 5.3 that omitting the second term would result in poor convergence.

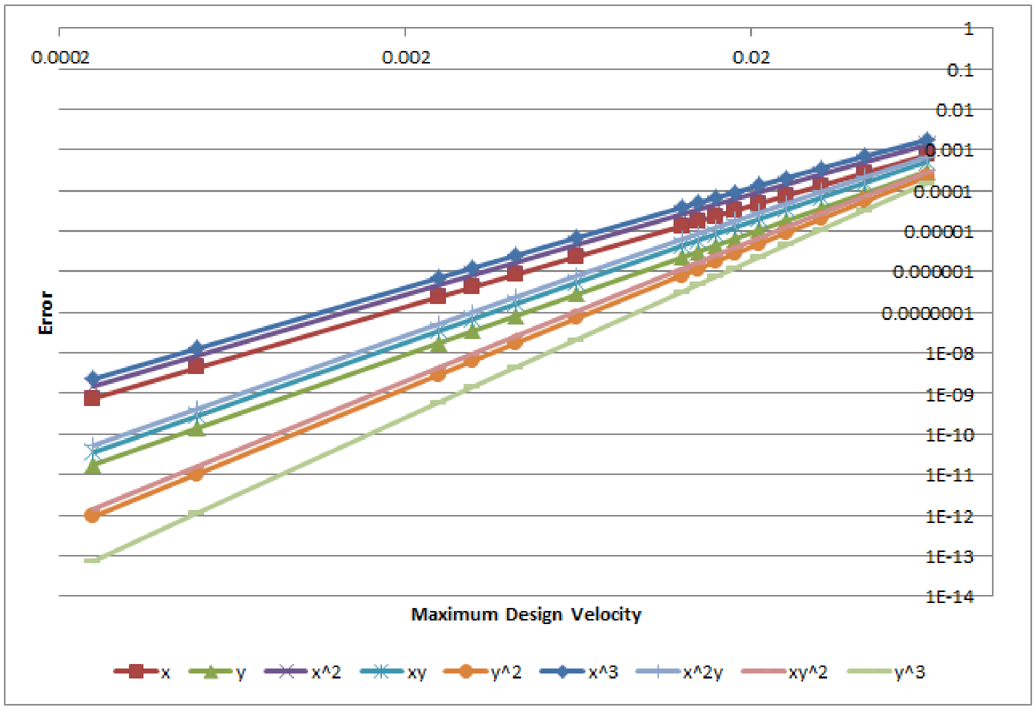


Figure 5.5: Convergence plot for integral of bivariate polynomials (of order up to 3) over the quadrant obtained using SAQ

Fig. 5.5 is the convergence plot for the SAQ scheme in integrating all bivariate polynomials ($x^m y^n$) of order up to 3 using 10 randomly generated integration points lying completely inside Ω and Ω_0. We find that the order of convergence (w.r.t magnitude of the maximum design velocity) is always greater than 2 (ranging from 2.5 to 3.95) for all the considered integrands.

5.3.2 Example 2 - Family of noisy domains

In this example, we will study the convergence of SAQ in integrating all bivariate polynomials ($x^m y^n$) of order up to 3 over a family of noisy domains obtained by setting different values of k in the equation $y(x) = x^4 + e^x + 0.75 sin(k\pi x) + 0.5$ for $x \in [0, 2]$ (see Fig. 5.6). As before, we use 10 randomly generated points (lying completely inside Ω and Ω_0) as our integration points.

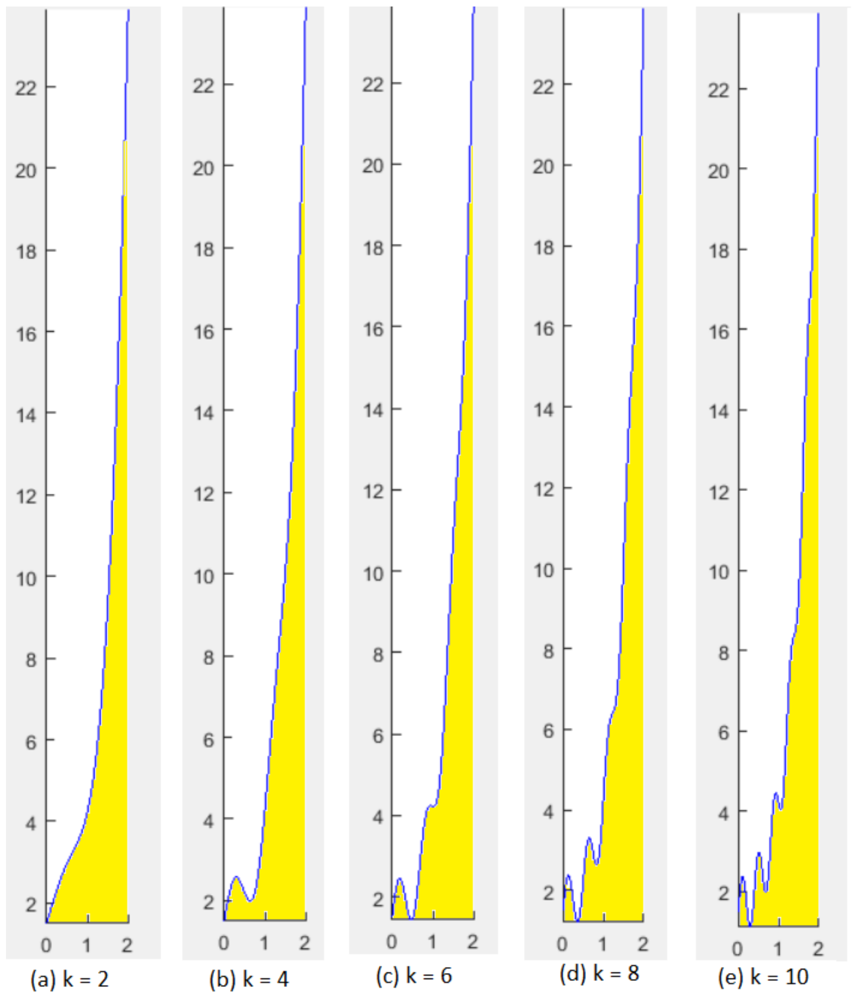


Figure 5.6: A family of noisy domains obtained by setting various values of k in $y(x) = x^4 + e^x + 0.75\, sin(k\pi x) + 0.5$ for $x \in [0, 2]$

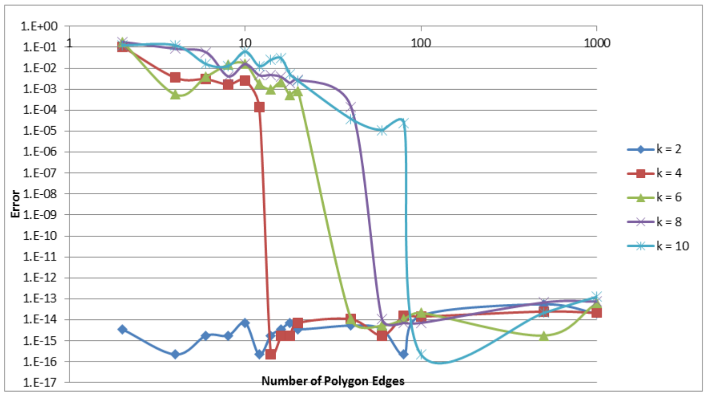


Figure 5.7: Convergence plot for area computation over the noisy domain for various values of k

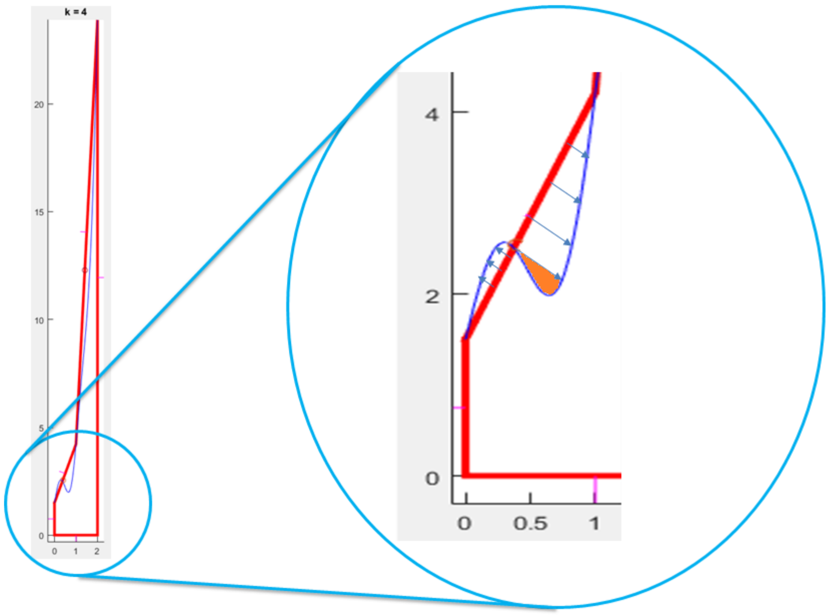


Figure 5.8: Illustration of violation of homeomorphic assumption at coarser polygonal approximations ($n_e = 7$) for the noisy domain with $k = 4$

The convergence of SAQ (w.r.t. number of polygonal edges) for the area computation over this family of noisy domains is given in Fig. 5.7. It is interesting to note that, unlike the quadrant domain, there is significant error for coarser polygonal approximations even for area computations. This is because, at coarser polygonal approximation, the bijective normal mapping that we assume between Ω and Ω_0 is not homeomorphic as illustrated in Fig. 5.8. This problem gets resolved by using sufficiently higher number of edges so that

Ω and Ω_0 are homeomorphic to each other. As in the quadrant problem, we observe that the error slightly increases with increase in number of edges due to numerical noise. Fig. 5.9-5.12 plots the convergence of SAQ w.r.t. maximum design velocity for $k = 2, 4, 6$ and 8. Fig. 5.9 exhibits a consistent order of convergence close to 2 (ranging from 2.05 to 2.09). However, as the boundary becomes more and more noisy (with increasing value of k), we don't have a consistent order of convergence for coarser polygonal approximations. Once there are sufficient number of edges to ensure homeomorphic mapping between Ω and Ω_0, there is a consistent order of convergence close to 2 (ranging from 1.96 to 2.47).

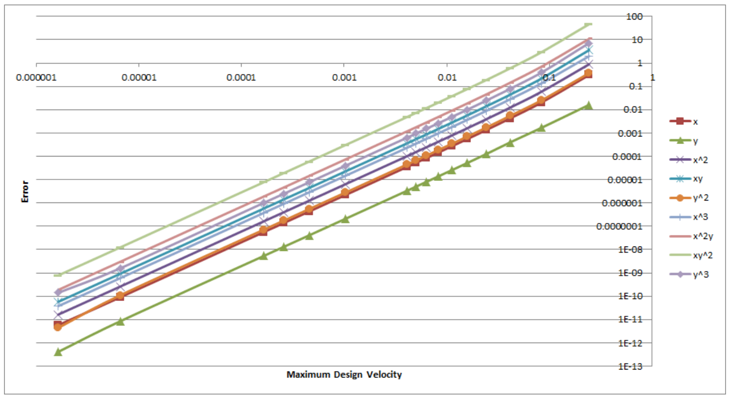


Figure 5.9: Convergence plot for integral of bivariate polynomials (of order up to 3) over the noisy domain with $k = 2$

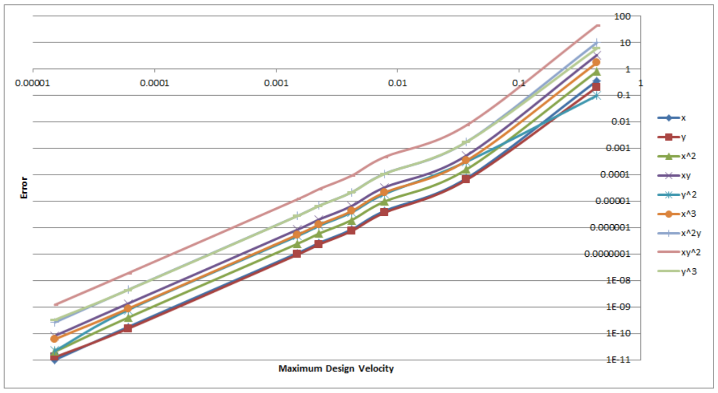


Figure 5.10: Convergence plot for integral of bivariate polynomials (of order up to 3) over the noisy domain with $k = 4$

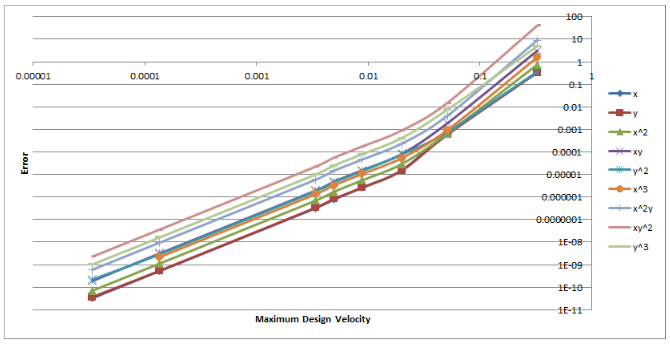


Figure 5.11: Convergence plot for integral of bivariate polynomials (of order up to 3) over the noisy domain with $k = 6$

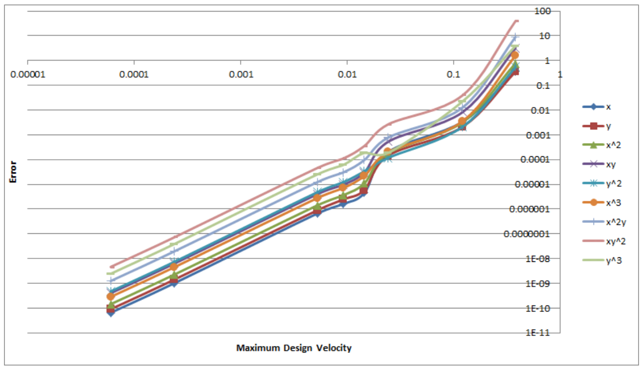


Figure 5.12: Convergence plot for integral of bivariate polynomials (of order up to 3) over the noisy domain with $k = 8$

5.4 *A posteriori* error analysis

In this section, we will develop an *a posteriori* error estimate for SAQ due to first-order shape correction factors obtained using SSA [75]. We will also provide numerical results to support the theoretical estimates. We neglect all other sources of error (including numerical errors) as discussed in section 5.2.

Theorem 5.4.1. *An a posteriori error estimate due to moment approximations (by SSA) in SAQ is given by the following*

$$e(q^*) \approx \{\mathbf{\Delta \hat{M}}\}^T\{\mathbf{z}\} \tag{5.46}$$

where $\{\mathbf{z}\}$ *is the adjoint solution obtained by solving* $(\mathbf{A})^T\{\mathbf{z}\} = \{\mathbf{f}(\mathbf{x})\}$ *and the error in moments* $\mathbf{\Delta M}$ *is approximated by* $\Delta\hat{M}_i \approx \frac{1}{2}\int_{\Gamma_0}\Big[\nabla b_i(\mathbf{X}).\mathbf{N}(\mathbf{X})V_N^2(\mathbf{X})\Big]d\Gamma_0$

Proof. From Eq.(5.16) we have the following

$$\begin{aligned} e(q^*) &= ([\mathbf{A}]^{-1}\{\mathbf{\Delta M}\})^T\{\mathbf{f}(\mathbf{x})\} \\ &= \{\mathbf{\Delta M}\}^T\{\mathbf{z}\} \text{ (where } [\mathbf{A}]^T\{\mathbf{z}\} = \{\mathbf{f}(\mathbf{x})\} \text{)} \end{aligned} \tag{5.47}$$

$\{\mathbf{\Delta M}\} = \{dM_i\}_{i=1}^m$ is the actual error in moments. Here, we will estimate the actual error in the moments using a higher order Taylor series. In other words, we know that, first-order Taylor series approximation of moments is given by

$$M_i^1(t) = M_i\big|_{t=0} + t\frac{dM_i}{dt}\big|_{t=0} \tag{5.48}$$

Likewise, one can obtain a second order Taylor series approximation of the moments by adding the second order SSA to the above as follows :

$$M_i^2(t) = M_i\big|_{t=0} + t\frac{dM_i}{dt}\big|_{t=0} + \frac{t^2}{2!}\frac{d^2M_i}{dt^2}\big|_{t=0} \tag{5.49}$$

The error in moments can thus be approximated as the difference between the above two approximations :

$$\begin{aligned}|\Delta M_i| &\approx |\Delta \hat{M}_i| \\ &= |M_i^2(t) - M_i^1(t)| \\ &= \frac{t^2}{2!}\left|\frac{d^2 M_i}{dt^2}\right|_{t=0} \\ &\leq \frac{1}{2}\left|\frac{d^2 M_i}{dt^2}\right|_{t=0}\end{aligned} \tag{5.50}$$

The second order SSA $\frac{d^2 M_i}{dt^2}\big|_{t=0}$ can be shown to be equal to the following (see proof in Appendix A) integral over the polygonal/polyhedral boundary (Γ_0) if we assume normal perturbations ($\hat{\mathbf{V}} = V_N \mathbf{N}$)

$$\left.\frac{d^2 M_i}{dt^2}\right|_{t=0} = \int_{\Gamma_0} \left[\nabla[b_i(\mathbf{X})].\mathbf{N}(\mathbf{X}) V_N^2(\mathbf{X})\right] d\Gamma_0 \tag{5.51}$$

substituting this expression in Eq.(5.50) completes the proof. □

5.4.1 Example 3 - Quadtree

In this example, we will use the *a posteriori* error estimate developed in this section to predict the SSA error in integrating an arbitrary polynomial $f(x, y) = 10 + 0.1x + 0.4y - x^2 + 5xy + 2y^2 + 9x^3 - 10x^2y + 10xy^2 - 10y^3$ over a unit circle (Fig. 5.13(a)), noisy domain (Fig. 5.13(b)) and non-convex domain (Fig. 5.13(c)). As $f(x, y)$ is a degree 3 polynomial, we will use 4×4 box quadrature for interior cells (blue filled circles in Fig. 5.13) and a 10 pt SAQ in each of the leaf cells (red plus marks in Fig. 5.13). As before, we generate quadrature points for SAQ in the leaf cells randomly. The approximate polygon (Ω_0) is constructed over every leaf cell taking into account the type of leaf cell that arises in marching squares (see Fig. 4.5). Thus, we compute the integral over each of these cells and then sum up the contributions to get the integral over the original domain. Likewise, we compute the error term (Eq.(5.46)) over each of the leaf cells and sum them up to get an estimate for the overall error. The overall predicted error for each of these domains is plotted along with the actual error (computed using symbolic integration in MATLAB [90]) for various levels of quadtree refinement in Fig. 5.14. From the plot it is clear that the *a posteriori* error estimate is pretty accurate for all three domains.

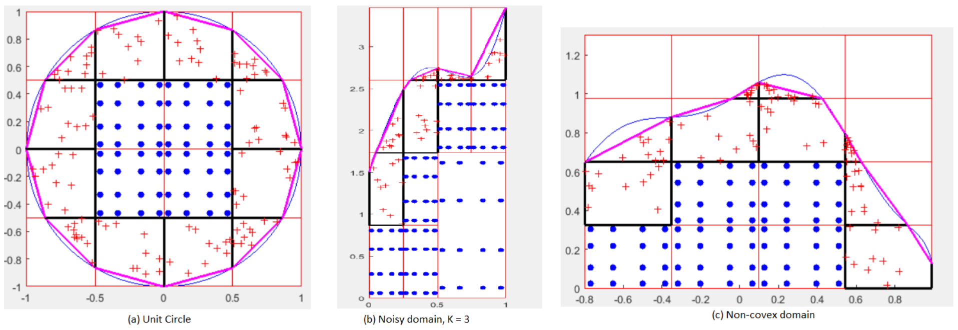


Figure 5.13: SAQ using a quadtree over different domains

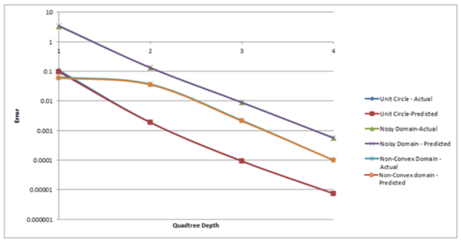


Figure 5.14: Comparison of actual and predicted errors for the unit circle, noisy domain, and non-convex domain

One main application of our *a posteriori* error estimate is that it can be used drive an adaptive refinement process as the error can be localized over every leaf cell as shown in Fig. 5.15 for the non-convex domain (of Fig. 5.13(c)). From this plot, it is clear that close to 94% of the total error is concentrated in just two cells C_1 (yellow) and C_2 (light blue). This is primarily because, unlike the other cells, Ω_0 in these two cells do not approximate the original domain (Ω) closely. Hence, it is more effective to subdivide these two cells instead of others. In this example, the error seems to be proportional to integral of velocity in each of these cells. However, this need not always be true as illustrated by Fig. 5.16. Fig. 5.16 is the error plot corresponding to the integral of the following continuous function over the non-convex domain (Ω) using SAQ

$$g(x,y) = \begin{cases} x - 0.095 & \text{if } x \in [-0.8, 0.095] \\ f(x - 0.095, y) & \text{if } x \in [0.095, 0.99] \end{cases}$$

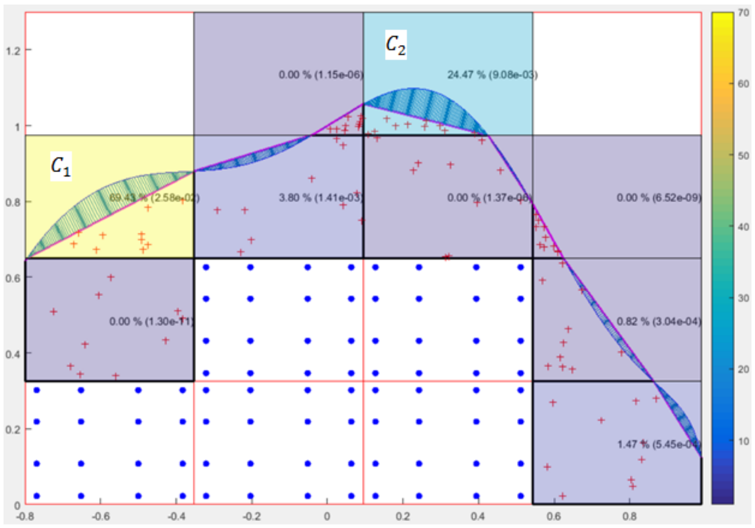


Figure 5.15: *A posteriori* error plot for integral of $f(x, y)$ over the non-convex domain

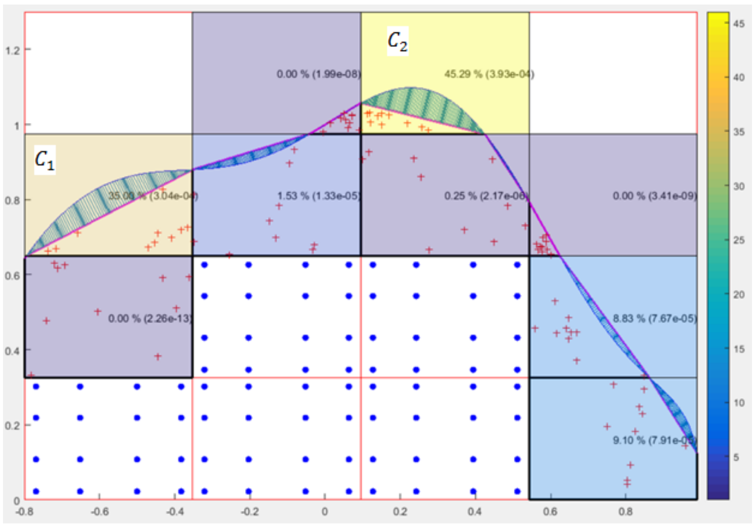


Figure 5.16: *A posteriori* error plot for integral of $g(x, y)$ over the non-convex domain

From this plot, we see that the error distribution is altered significantly to the point where the maximum error cell (yellow cell) has shifted and the total error in the two cells

(C_1, C_2) put together has dropped to 80.29%. Thus, this example illustrates how the SSA error depends not only on the velocity but also on the integrand.

5.5 Summary

Errors in SAQ can arise from one or more of the following :

1. Insufficient/inappropriate choice of basis functions
2. Inappropriate number/location of domain quadrature points
3. Inaccurate Moment computations
4. Fixed precision arithmetic

Out of the above, inaccurate moment computations is the main source of error in SAQ and all the others can be easily eliminated as discussed in section 5.2. There are three possible sources of errors that can lead to inaccurate moment computations and They are

1. Inaccurate distance and/or measure computations (dM_i^d)
2. Boundary integration error (dM_i^b)
3. n^{th} order Taylor series approximation (dM_i^s)

Thus, the total error in the computation of i^{th} moment is bounded by

$$|dM_i| \leq |dM_i^d + dM_i^b + dM_i^s| \leq \zeta \left| \int_{\Gamma_0} \beta_X^i(\mathbf{X}) N_X d\Gamma_0 \right| + \eta \left| \sum_{j=1}^{n_s} b_i(\mathbf{X_j}) \right| + e_i^b + e_i^s \tag{5.52}$$

and an *a posteriori* error estimate of the computed integral (using SAQ) can be estimated as

$$|e| \leq \{\mathbf{dM}\}^T \{\mathbf{z}\} \tag{5.53}$$

where $\mathbf{z}$ is the adjoint solution obtained by solving $[\mathbf{A}]^T\{\mathbf{z}\} = \{f(\mathbf{X}_i)\}$ and $\{\mathbf{dM}\}$ is the estimate of the total error in moments. We have proved in Theorem 5.3.4 that SAQ converges quadratically w.r.t maximum design velocity (and hence with the mesh size). We have also given an *a posteriori* error estimate in Theorem 5.4.1 to estimate this error computationally. This *a posteriori* error estimate can be localized and hence can be used to drive an adaptive grid refinement process. Several 2D examples were given to corroborate these two theorems.

Chapter 6

Contributions and Open Issues

Contributions We proposed a new integration framework called *Shape Aware Quadratures* (SAQ) using moment fitting equations [64, 116, 137], divergence theorem [41], and sensitivity analysis [75, 106, 123, 80] to efficiently integrate arbitrary integrable functions over arbitrary 2D/3D domains (Ω) even in the presence of small features. A simplified domain (Ω_0) was used to approximate the moments in moment fitting equations. Appropriate geometric and topological correction factors were used to correct for the shape deviation of Ω_0 from Ω. Shape correction factors were derived by the application of first/second-order Shape Sensitivity Analysis (SSA) [75] and Topological Sensitivity Analysis (TSA) [106] of the moment integrals. The use of shape correction factors in the moment fitting equations enabled the resulting quadrature rule "shape aware".

We demonstrated the use of SAQ in integrating arbitrary polynomial functions over arbitrary 2D/3D domains in the presence of small features (such as notches, boundary noise and holes) and compared it with some standard methods: scaled Cartesian product rule as recommended in [132], geometric adaptive (GA) integration method proposed in [22], polygonal (P) approximation method, shape sensitivity (SS) method (section 2.1.4) and characteristic function approach [134]. SAQ was shown to be superior to GA, P and characteristic methods in terms of accuracy and comparable to SS method in most cases. We also observed that SAQ, when used in the leaf cells of quadtrees/octrees, required fewer subdivisions / quadrature nodes to resolve the geometry of the integration domain when compared to other methods. We also note here that the method was formulated irrespective of how the original domain Ω is represented, and it can be used with or without an integration mesh. SAQ offers a lot of flexibility in the choice of quadrature points and in the choice of basis functions as its formulation is based on the moment fitting equations. In fact, our experiments indicate that we can indeed generate quadrature points randomly either in the domain or on boundary as long as the moment matrix $[A]$ is invertible. The weights automatically adjusts itself depending on the position of quadrature points. A wide variety of basis functions can be chosen in forming the moment fitting equations depending on the type of integrand that arises in any given application.

One of the main applications of SAQ is in immersed boundary methods such as the Finite Cell Method (FCM). We successfully demonstrated the application of SAQ in FCM. We implemented the method in FCMLab [95] and performed convergence studies for 2D plane stress and 3D isotropic elasticity. The experimental results clearly indicates that, to achieve a given accuracy, SAQ requires far fewer subdivisions / less time compared to the standard integration scheme (characteristic function approach) used in FCM. The characteristic function approach is found to produce erroneous results at lower levels of refinement for certain problems in addition to exhibiting oscillatory convergence. Reliable and consistent results from characteristic function method can be expected only with 4 or more subdivisions. This is essentially because of the inability of the method to resolve the geometry at lower grid resolutions. This makes the characteristic function approach computationally very expensive for problems with complicated geometry such as the human femur model shown in Fig. 4.23. On the other hand, the introduction of shape correction factor in the computation of moments in SAQ makes the computed quadrature weights shape aware and thus enables the weights to adapt to the domain of integration automatically. This explains why SAQ produces accurate results even at lower grid resolution.

The introduction of shape correction factors results in an approximation of the moments thereby affecting the accuracy of the integral computed using SAQ. Hence, we performed an a priori error analysis of SAQ with SSA correction factors (of first-order) and proved that the method is quadratic w.r.t. the integration mesh size (Theorem 5.3.4). We also derived a simple *a posteriori* error estimate using second-order SSA (Theorem 5.4.1). This *a posteriori* error estimate can be localized and hence can be used to drive an adaptive grid refinement process. Several 2D examples were given to support these two error estimates.

The following are some possible impacts of this work :

1. Since SAQ was formulated independent of the representation of the domain, it would be interesting to explore the possibility of building a efficient 3D integration system purely based on basic geometric queries (such as PMC, distance computations and intersection tests) that will enable interoperable numerical integration over any given solid representation (such as meshes, implicit representation, B-reps, and voxels).

2. Another significant impact of this work is studying the effect of other types of shape correction factors such as feature sensitivity [123] and modification sensitivity [80] and its comparison to TSA/SSA based correction factors.

3. Application of SAQ to evolving level sets and non-linear problems is a straight forward extension

Open issues One minor weakness of this method is that the use of random quadrature points is not fully justified. Hence, it is required to more rigorously establish the effect of position of quadrature points on the moment matrix. Especially, it is important to study the effect of position of quadrature points on the conditioning and invertability of the moment matrix. However, the use of approximate Fekete points [15] could help eliminate this problem completely. Nevertheless, it is required to further study the effect of Fekete points in 2D/3D in the context of SAQ.

One other limitation of the method is that the integral computed using SAQ is dependent on the approximate domain (Ω_0). Hence, there is a need to eliminate the use of intermediate domain or at least to rigorously understand the effect of this approximate domain on the integral computed.

The moment computations in moment fitting equations (Eq. (3.1)) requires domain integration of smooth basis functions. In SAQ, we convert this into the boundary integration of a much more complicated function (due to the presence of distance to boundary) via divergence theorem and appropriate sensitivities in boundary form. However, this boundary integration is non-trivial and often requires higher-order or adaptive integration [90]. Adaptive integration, although an elegant solution, can prove to be very expensive especially in three dimensions. Also, it is not clear on how to decide the order of integration when one opts for higher-order boundary integration. Thus, there is a need to improve the efficiency of moment computations in SAQ by coming up with a suitable higher-order or adaptive integration scheme.

As already discussed, one of the main applications of SAQ is in meshfree methods where Legendre, spline, and other types of polynomial basis functions are employed to study various field problems. Hence, it is more natural/useful to consider Legendre/Chebyshev/spline polynomial basis function in moment fitting equations (for arbitrary 2D/3D domains) so as to improve the accuracy and relevance of SAQ to meshfree analysis.

Bibliography

[1] Encyclopaedia of cubature formulas. `http://nines.cs.kuleuven.be/ecf/`. [Online; accessed 2016-06-07].

[2] MARCHING CUBES. `http://users.polytech.unice.fr/~lingrand/MarchingCubes/algo.html`. [Online; accessed 2016-06-07].

[3] SOLIDWORKS. http://www.solidworks.com/. [Online; accessed 2016-06-07].

[4] Baiges J and Codina R. The fixed-mesh ALE approach applied to solid mechanics and fluid-structure interaction problems. *International Journal For Numerical Methods in Engineering*, 81:1529–1557, 2010.

[5] Gerstenberger A. and Wall W.A. Enhancement of fixed-grid methods towards complex fluid-structure interaction applications. *International Journal For Numerical Methods in Fluids*, 57:1227–1248, 2008.

[6] Löhner R, Cebral R.J, Camelli F.E, Appanaboyina S, Baum J.D, Mestreau E.L, and Soto O.A. Adaptive embedded and immersed unstructured grid techniques. *Computer Methods in Applied Mechanics and Engineering*, 197:2173–2197, 2008.

[7] Neittaanmäki P and Tiba D. An embedding of domains approach in free boundary problems and optimal design. *SIAM Journal on Control and Optimization*, 33:1587–1602, 1995.

[8] Ramière I, Angot P, and Belliard M. A fictitious domain approach with spread interface for elliptic problems with general boundary conditions. *Computer Methods in Applied Mechanics and Engineering*, 196:766–781, 2007.

[9] Rüberg T and Cirak F. Subdivision-stabilised immersed B-spline finite elements for moving boundary flows. *Computer Methods in Applied Mechanics and Engineering*, 209–221:266–283, 2012.

[10] Wall W.A, Gamnitzer P, and Gerstenberger A. Fluid-structure interaction approaches on fixed grids based on two different domain decomposition ideas. *International Journal of Computational Fluid Dynamics*, 22:411–427, 2008.

[11] Zhang L, Gerstenberger A, Wang X, and Liu W.K. Immersed finite element method. *Computer Methods in Applied Mechanics and Engineering*, 193:2051–2067, 2004.

[12] Abedian A., Parvizian J., Düster A., Khademyzadeh H., and Rank E. Performance of different integration schemes in facing discontinuities in the Finite Cell Method. *International Journal of Computational Methods*, 10:1–24, 2013.

[13] Düster A., Parvizian J., Yang Z., and Rank E. The finite cell method for three-dimensional problems of solid mechanics. *Computer Methods in Applied Mechanics and Engineering*, 197:3768–3782, 2008.

[14] Sommariva A. and Vianello M. Product Gauss cubature over polygons based on Green's integration formula. *BIT Numerical Mathematics*, 47:441–453, 2007.

[15] Sommariva A. and Vianello M. Computing approximate Fekete points by QR factorizations of vandermonde matrices. *Computers and Mathematics with Applications*, 57:1324–1336, 2009.

[16] Sommariva A. and Vianello M. Gauss–Green cubature and moment computation over

arbitrary geometries. *Journal of Computational and Applied Mathematics*, 231:886–896, 2009.

[17] Novotny A.A., Feijóo R. A., E. Taroco, and Padra C. Topological-Shape Sensitivity Analysis. *Computer Methods in Applied Mechanics and Engineering*, 192:803–829, 2003.

[18] Novotny A.A., Feijóo R. A., E. Taroco, and Padra C. Topological Sensitivity Analysis for Three-Dimensional Linear Elasticity Problems. In *6th World Congress on Structural and Multidisciplinary Optimization*, 2005.

[19] Novotny A.A., Feijóo R. A., E. Taroco, and Padra C. Topological-Shape Sensitivity Method : Theory and Applications. *Solid Mechanics and its Applications*, 137:469–478, 2006.

[20] Stroud A.H. *Approximate Calculation of Multiple Integrals.* Prentice–Hall, Englewood Cliffs, NJ, 1971.

[21] Kumar A.V. and Lee J. Step function representation of solid models and application to mesh free engineering analysis. *Journal of Mechanical Design (Transactions of the ASME)*, 128:46–56, 2005.

[22] Luft B., Shapiro V., and Tsukanov I. Geometrically adaptive numerical integration. In *2008 ACM symposium on Solid and physical modeling*, pages 147–157, NY, 2008.

[23] Mirtich B. Fast and accurate computation of polyhedral mass properties. *Journal of Graphics Tools*, 1:31–50, 1996.

[24] Szabó B and Babuška I. *Finite Element Analysis.* John Wiley & Sons, Inc : New York, 1991.

[25] Ivo Babuska and Theofanis Strouboulis. *The finite element method and its reliability.* Numerical mathematics and scientific computation. Clarendon Press New York, Oxford, 2001.

[26] Belytschko T., Parimi C., Moës N., Sukumar N., and Usui S. Structured extended finite element methods for solids defined by implicit surfaces. *International Journal For Numerical Methods in Engineering*, 56:609–635, 2003.

[27] Jarle Berntsen, Ronald Cools, and Terje O. Espelid. Algorithm 720: An algorithm for adaptive cubature over a collection of 3-dimensional simplices. *ACM Trans. Math. Softw.*, 19(3):320–332, September 1993.

[28] Jarle Berntsen, Terje O. Espelid, and Alan Genz. An adaptive algorithm for the approximate calculation of multiple integrals. *ACM Trans. Math. Softw.*, 17(4):437–451, December 1991.

[29] S.P. Boyd and L. Vandenberghe. *Convex optimization.* Cambridge University Press, 2004.

[30] Cattani C. and Paoluzzi A. Boundary integration over linear polyhedra. *Computer-Aided Design*, 22:130–135, 1990.

[31] Min C. and Gibou F. Geometric integration over irregular domains with application to level-set methods. *Journal of Computational Physics*, 226:1432–1443, 2007.

[32] Ricolindo Cario, Ian Robinson, and Elise De Doncker. Adaptive cubature over a collection of triangles using the d-transformation. *Journal of Computational and Applied Mathematics*, 50(1):171 – 183, 1994.

[33] Peskin C.S. The immersed boundary method. *Acta Numerica*, 11:479–517, 2002.

[34] J. Rocha de Faria, A.A. Novotny, R.A. Feijo, E. Taroco, and C. Padra. Second order topological sensitivity analysis. *International Journal of Solids and Structures*, 44(1415):4958 – 4977, 2007.

[35] Hunkins D.R. Cubatures of precision 2k and 2k+1 for hyperrectangles. *Mathematics of Computation*, 29:1098–1104, 1975.

[36] Bernardini F. Integration of polynomials over n-dimensional polyhedra. *Computer-Aided Design*, 23:51–58, 1991.

[37] Brezzi F. On the existence, uniqueness and approximation of saddle-point problems arising from Lagrangian multipliers. *Rev.Francaise d'Automat.Informat.Recherche Opérationnelle Sér.*, 8:129–151, 1974.

[38] Feito F., Torres J.C., and Urena A. Orientation, simplicity and inclusion test for planar polygons. *Computers and Graphics*, 19:595–600, 1995.

[39] Klein F. A New Approach to Point Membership Classification in B-rep Solids. *Mathematics of Surfaces XIII, Lecture Notes in Computer Science*, 5654:235–250, 2009.

[40] Fries T.-P. and Belytschko T. The extended/generalized finite element method : An overview of the method and its applications. *International Journal for Numerical Methods in Engineering*, 84:253–304, 2010.

[41] Dasgupta G. Integration within polygonal finite elements. *J. Aerosp. Eng.*, 16:9–18, 2003.

[42] Strang G. The Fundamental Theorem of Linear Algebra. *The American Mathematical Monthly*, 100:848–855, 1993.

[43] Ventura G. On the elimination of quadrature subcells for discontinuous functions in the eXtended Finite-Element Method. *International Journal for Numerical Methods in Engineering*, 66:761–795, 2006.

[44] S. Garribba, L. Quartapelle, and G. Reina. Algorithm 36 sniff: Efficient self-tuning algorithm for numerical integration. *Computing*, 20(4):363–375, 1978.

[45] Glowinski R and Kuznetsov Y. Distributed lagrange multipliers based on fictitious domain method for second order elliptic problems. *Computer Methods in Applied Mechanics and Engineering*, 196:1498–1506, 2007.

[46] Liu G.R. *Meshfree Methods : Moving Beyond the Finite Element Method.* CRC Press, 2009.

[47] Liu G.R. and Nguyen T.T. *Smoothed Finite Element Methods.* CRC Press, 2010.

[48] Samet H. *The design and analysis of spatial data structures.* Addison-Wesley Publishing Company INC., MA, 1990.

[49] Samet H. *Foundations of Multidimensional and Metric Data Structures.* Morgan Kaufmann Publishers Inc., SF, 2005.

[50] Eschenauer H.A. and Olhoff N. Topology optimization of continuum structures: A review. *Appl. Mech.*, 54:331–390, 2001.

[51] Haslinger J and Renard Y. A new fictitious domain approach inspired by the extended finite element method. *SIAM Journal on Numerical Analysis*, 47:1474–1499, 2009.

[52] Babuška I. The finite element method with Lagrangian multipliers. *Numer.Math.*, 20:179–192, 1973.

[53] Babuška I. The finite element method with Penalty. *Mathematics of Computation*, 27:221–228, 1973.

[54] Babuška I., Szabó B.A, and Katz I.N. The p-version of the finite element method. *SIAM Journal on Numerical Analysis*, 18:515–545, 1981.

[55] Babuška I. and Banerjee U. Stable Generalized Finite Element Method (SGFEM). *Computer Methods in Applied Mechanics and Engineering*, 201–204:91–111, 2012.

[56] Babuška I., Banerjee U, and Osborn J.E. Meshless and generalized finite element methods: A survey of some major results. *Meshfree Methods for Partial Differential Equations,Lecture notes in Computational Science and Engineering*, 26:1–20, 2002.

[57] Ninomiya I. Improvements of newton-cotes quadrature methods. *J.Info Process*, 3(3):162–170, 1980.

[58] Tsukanov I. and Shapiro V. The architecture of SAGE - a meshfree system based on RFM. *Engineering with Computers*, 18:295–311, 2002.

[59] Turevsky I., Gopalakrishnan S.H., and Suresh K. Generalization of Topological Sensitivity and its Application to Defeaturing. In *ASME-IDETC*, Las Vegas, 2007.

[60] Turevsky I., Gopalakrishnan S.H., and Suresh K. Defeaturing: A posteriori error analysis via feature sensitivity. *International Journal for Numerical Methods in Engineering*, 76:1379–1401, 2008.

[61] Bernstein J., Epselid T., and Genz A. An adaptive algorithm for the approximate calculation of multiple integrals. *ACM Transactions on Mathematical Software*, 17:437451, 1991.

[62] Elstrodt J. Maβ- und integrationstheorie. In *Springer Verlag*, chapter 4, pages 266–290.

[63] Freund J. and Stenberg R. On weakly imposed boundary conditions for second order problems. In *International Conference on Finite Elements in Fluids - New trends and applications*, pages 327–336, 1995.

[64] Lyness J. and Jespersen D. Moderate degree symmetric quadrature rules for the triangle. *IMA Journal of Applied Mathematics*, 15:19–32, 1975.

[65] Oden J and Prudhomme S. Estimation of modeling error in computational mechanics. *Journal of computational physics*, 182:496–515, 2002.

[66] Parvizian J., Düster A., and Rank E. Finite Cell Method : h- and p-extension for embedded domain methods in solid mechanics. *Computational Mechanics*, 41:121–133, 2007.

[67] Sokolowski J. and Zochowski A. On Topological Derivative in Shape Optimization. *SIAM Journal on Control and Optimization*, 37:1251–1272, 1999.

[68] Lasserre J.B. Integration on a convex polytope. *Proceedings of the American Mathematical Society*, 126:2433–2441, 1998.

[69] Lasserre J.B. Integration and homogeneous functions. *Proceedings of the American Mathematical Society*, 127:813–818, 1999.

[70] Laguardia J.J., Cueto E., and Doblaré M. A natural neighbour Galerkin method with quadtree structure. *International Journal for Numerical methods in Engineering*, 63:789–812, 2005.

[71] Joulaian M., Hubrich S. and Düster A. Numerical integration of discontinuities on arbitrary domains based on moment fitting. *Computational Mechanics*, 57:979–999, 2016.

[72] Höllig K. *Finite Element Methods with B-Splines.* Frontiers in Applied Mathematics. SIAM, 2003.

[73] Höllig K., Apprich C., and Streit A. Introduction to the Web-method and its applications. *Advances in Computational Mathematics*, 23:215–237, 2005.

[74] P. Kaufmann, S. Martin, M. Botsch, and M. Gross. Flexible simulation of deformable models using discontinuous Galerkin FEM. In *2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 105–115, 2008.

[75] Choi K.K. and Kim N.H. *Structural Sensitivity Analysis and Optimization I: Linear Systems*. New York: Springer, 2005.

[76] Choi K.K. and Kim N.H. *Structural Sensitivity Analysis and Optimization II: Non-Linear Systems*. New York: Springer, 2005.

[77] Legrain G., Chevaugeon N, and Dréau K. High order X-FEM and levelsets for complex microstructures: uncoupling geometry and approximation. *Computer Methods in Applied Mechanics and Engineering*, 241–244:172–189, 2012.

[78] Shampine L.F. MATLAB program for quadrature in 2D. *Appl. Math. Comput.*, 202:266–274, 2008.

[79] Ming Li and Shuming Gao. Estimating defeaturing-induced engineering analysis errors for arbitrary 3d features. *Computer-Aided Design*, 43:1587–1597, 2011.

[80] Ming Li, Shuming Gao, and Ralph Martin. Estimating effects of removing negative features on engineering analysis. *Computer-Aided Design*, 43:1402–1412, 2011.

[81] Ming Li, Shuming Gao, and Kai Zhang. A goal oriented error estimator for the analysis of simplified designs. *Computer Methods in Applied Mechanics and Engineering*, 255:89–103, 2013.

[82] Trefethen L.N. and Bau D. *Numerical Linear Algebra*. SIAM, 1997.

[83] Lorensen W.E. and Cline H.E. Marching cubes: A high resolution 3D surface construction algorithm. *ACM Computer Graphics*, 21:163–169, 1987.

[84] Freytag M. and Shapiro V. B-rep SE: simplicially enhanced boundary representation. In *ACM Symposium on Solid Modeling and Applications, Switzerland*, pages 157–168, 2004.

[85] Freytag M., Shapiro V., and Tsukanov I. Scan and solve: Acquiring the physics of artifacts. In *ASME 2007 International Design Engineering Technical Conferences and*

Computers and Information in Engineering Conference, pages 345–356, Las Vegas, USA, 2007.

[86] Freytag M., Shapiro V., and Tsukanov I. Finite element analysis in situ. *Finite Elements in Analysis and Design*, 47:957–972, 2011.

[87] Li M., Zhang B., and Martin R.R. Second-order defeaturing error estimation for multiple boundary features. *International Journal for Numerical Methods in Engineering*, pages 1–25, 2013.

[88] Mortenson M. *Geometric Modeling.* John Wiley and Sons, Inc., New York, 1997.

[89] S. Martin, P. Kaufmann, M. Botsch, M. Wicke, and M. Gross. Polyhedral Finite Elements using harmonic basis functions. In *Eurographics Symposium on Geometry Processing (Copenhagen, Denmark, July 2-4, 2008), Computer Graphics Forum*, pages 1521–1529, 2008.

[90] MATLAB. *version 7.10.0 (R2010a).* The MathWorks Inc., Natick, Massachusetts, 2010.

[91] Moës N., Cloirec M., Cartraud P., and Remacle J.-F. A Computational approach to handle complex microstructure geometries. *Computer Methods in Applied Mechanics and Engineering*, 192:3163–3177, 2003.

[92] Mousavi S.E., Xiao H. and Sukumar N. Generalized Gaussian quadrature rules on arbitrary polygons. *International Journal for Numerical Methods in Engineering*, 82:99–113, 2010.

[93] Danwitz M.V. Automated application of Dirichlet boundary conditions in voxel based analyses using the Finite Cell Method. Bachelor's thesis, Technische Universität München, September 2013.

[94] Amenta N. and Bern M. Surface reconstruction by Voronoi filtering. In *Fourteenth annual symposium on Computational geometry*, pages 39–48, 1998.

[95] Zander N., Bog T., Elhaddad M., Espinoza R., Hu H., Joly A., Wu C., Zerbe P., Düster A., Kollmannsberger S., Parvizian J., Ruess M., Schillinger D., and Rank E. FCMLab: A finite cell research toolbox for MATLAB. *Advances in Engineering Software*, 74:49–63, 2014.

[96] Klaas O. and Shephard M. Automatic generation of octree-based three dimenisonal discretizations for partition of unity methods. *Computational Mechanics*, 25:296–304, 2000.

[97] Steven J. Owen. A survey of unstructured mesh generation technology. In *INTERNATIONAL MESHING ROUNDTABLE*, pages 239–267, 1998.

[98] Gonnet P. A Review of Error Estimation in Adaptive Quadrature. *ACM Computing Surveys*, 44, 2010.

[99] Hansbo P. and Larson M.G. Discontinuous Galerkin methods for incompressible and nearly incompressible elasticity by Nitsche's method. *Computer Methods in Applied Mechanics and Engineering*, 191:1895–1908, 2002.

[100] Smereka P. The numerical approximation of a delta function with application to level set methods. *Journal of Computational Physics*, 211:77–90, 2006.

[101] R. Piessens, E. deDoncker Kapenga, C. Uberhuber, and D. Kahaner. *QUADPACK: A Subroutine Package for Automatic Integration*. Springer–Verlag, 1983.

[102] Becker R. Mesh adaptation for Dirichlet flow control via Nitsche's method. *Commun.Numer.Methods Engrg*, 18:669–680, 2002.

[103] Cools R. An encyclopaedia of cubature formulas. *Journal of Complexity*, 19:445–453, 2003.

[104] Cools R. and Rabinowitz P. Monomial cubature rules since stroud : a compilation. *J. Comput. Appl. Math.*, 48:309–326, 1993.

[105] Mittal R. and Iaccarino G. Immersed boundary methods. *Annual Review of Fluid Mechanics*, 37:239–261, 2005.

[106] Feijóo R.A., Novotny A. A., Taroco E., and Padra C. The Topological Derivative For The Poisson'S Problem. *Math. Models Methods Appl. Sci.* , 13:1825–1844, 2003.

[107] Feijoo R.A., Novotny A.A., Taroco E., and Padra C. The topological derivative for poisson's problem. *Mathematical Models and Methods in Applied Sciences*, 13:1925–1844, 2003.

[108] A. Requicha. Representations of rigid solid objects. *Computer Aided Design Modelling, Systems Engineering, CAD-Systems*, pages 1–78, 1980.

[109] Sarraga R.F. Computation of surface areas in GMSolid. *IEEE Computer Graphics and Applications*, 2:65–70, 1982.

[110] Garren R.K. Bounds for the Eigenvalues of a Matrix. Technical Report NASA-TN-D-4373, National Aeronautics and Space Administration, Washington D.C., March 1968.

[111] Walter Rudin. *Principles of mathematical analysis*. McGraw-Hill Book Co., New York, third edition, 1976. International Series in Pure and Applied Mathematics.

[112] J. Ruppert. A delaunay refinement algorithm for quality 2-dimensional mesh generation. *Journal of Algorithms*, 18(3):548 – 585, 1995.

[113] Del Pino S and Pironneau O. A fictitious domain based general PDE solver. In *Numerical methods for scientific computing: Variational problems and applications, CIMNE, Barcelona*, 2003.

[114] Fernández-Méndez S. and Huerta A. Imposing essential boundary conditions in mesh-free methods. *Computer Methods in Applied Mechanics and Engineering*, 193:1257–1275, 2004.

[115] Osher S. and Fedkiw R. *Level Set Methods and Dynamic Implicit Surfaces*. Springer-Verlag, New York, 2003.

[116] Wandzura S. and Xiao H. Symmetric quadrature rules on a triangle. *Computers and Mathematics with Applications*, 45:1829–1840, 2003.

[117] Sauerland H and Fries T.-P. The extended finite element method for two-phase and free-surface flows: A systematic study. *Journal of Computational Physics*, 230:3369–3390, 2011.

[118] Schillinger D and Ruess M. The Finite Cell Method: A Review in the Context of Higher-Order Structural Analysis of CAD and Image-Based Geometric Models. *Archives of Computational Methods in Engineering*, 22:391–455, 2014.

[119] Roth S.D. Ray casting for modeling solids. *Computer Graphics and Image Processing*, 18:109–144, 1982.

[120] Mousavi S.E. and Sukumar N. Generalized Gaussian quadrature rules for discontinuities and crack singularities in the Extended Finite Element Method. *Computer Methods in Applied Mechanics and Engineering*, 199:3237–3249, 2010.

[121] Mousavi S.E. and Sukumar N. Numerical integration of polynomials and discontinuous functions on irregular convex polygons and polyhedrons. *Computational Mechanics*, 47:535–554, 2011.

[122] Mikhlin S.G. Error Analysis in Numerical Processes, 1991.

[123] Gopalakrishnan S.H. and Suresh K. Feature sensitivity : A generalization of topological sensitivity. *Finite Elmenets in Analysis and Design*, 44:696–704, 2008.

[124] Shahmiri S, Gerstenberger A, and Wall W.A. An XFEM-based embedding mesh technique for incompressible viscous flows. *International Journal For Numerical Methods in Engineering*, 65:166–190, 2011.

[125] Pierre Soille. *Morphological Image Analysis: Principles and Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2 edition, 2003.

[126] Sukumar N., Chopp D.L., Moës N., and Belytschko T. Modeling holes and inclusions by level sets in the extended finite-element method. *Computer methods in applied mechanics and engineering*, 190:6183–6200, 2001.

[127] Rabczuk T., Areias P.M.A., and Belytschko.T. A meshfree thin shell method for non-linear dynamic fracture. *International Journal for Numerical Methods in Engineering*, 72:524–548, 2007.

[128] Shapiro V. Solid Modeling. In *Handbook of Computer Aided Geometric Design*, pages 473–518. Elsevier Science Publishers: Amsterdam, 2002.

[129] Thiagarajan V. and Shapiro V. Adaptively weighted numerical integration over arbitrary domains. *Computers and Mathematics with Applications*, 67:1682–1702, 2014.

[130] Thiagarajan V. and Shapiro V. Adaptively weighted numerical integration in the Finite Cell Method. *Computer Methods in Applied Mechanics and Engineering*, 2016.

[131] Rvachev V.L., Shevchenko A.N., and Veretel'nik V.V. Numerical integration software for projection and projection-grid methods. *Cybernetics and Systems Analysis*, 30:154–158, 1994.

[132] Press W.H., Teukolsky S.A., Vetterling W.T., and Flannery B.P. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[133] M. Wicke, M. Botsch, and M. Gross. A Finite Element Method on convex polyhedra. *Computer Graphics Forum*, 26:355–364, 2007.

[134] Abdelaziz Y. and Hamouine A. A survey of the extended finite element. *Computers & Structures*, 86:1141–1151, 2008.

[135] Bazilevs Y. and Hughes T.J.R. Weak imposition of Dirichlet boundary conditions in fluid mechanics. *Computers & Fluids*, 36:12–26, 2007.

[136] Lee Y.T. and Requicha A. Algorithms for computing the volume and other integral properties of solids. I. known methods and open issues. *Communications of the ACM*, 25:635–641, 1982.

[137] Xiao H.and Gimbutas Z. A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions. *Computers and Mathematics with Applications*, 59:663–676, 2010.

[138] Yang Z., Ruess M., Kollmannsberger S., Düster A., and Rank E. An efficient integration technique for the voxel-based finite cell method. *International Journal for Numerical Methods in Engineering*, 91:457–471, 2012.

Appendices

Appendix A

SSA of Moments

First-Order SSA of Moments Recall that $M_i = \int_\Omega b_i(\mathbf{x})d\Omega$. It is required to compute the shape sensitivity of the moments M_i i.e. $\frac{dM_i}{dt}\Big|_{t=0}$. In order to evaluate this, it is first required to transform the moment integral over the reference domain Ω_0. This can be done by making use of the Jacobian matrix of the mapping $\mathbf{T}(\mathbf{X}, t)$. It can be easily established that $d\Omega = |J|d\Omega_0$ [75], where $|J|$ is the determinant of the Jacobian matrix . Thus, we have the following

$$\frac{dM_i}{dt}\bigg|_{t=0} = \frac{d}{dt}\bigg(\int_{\Omega_0} b_i(\mathbf{x})\,|J|\mathrm{d}\Omega_0\bigg)\bigg|_{t=0} \tag{A.1}$$

now, taking the derivative inside the integral and then applying the product and chain rule of differentiation we get

$$\begin{aligned}\frac{d}{dt}\bigg(\int_{\Omega_0} b_i(\mathbf{x})\,|J|\mathrm{d}\Omega_0\bigg)\bigg|_{t=0} &= \int_{\Omega_0} \frac{d}{dt}\Big(b_i(\mathbf{x})\,|J|\Big)\mathrm{d}\Omega_0\bigg|_{t=0} \\ &= \int_{\Omega_0} \bigg(\nabla b_i(\mathbf{x}).\hat{\mathbf{V}}(\mathbf{x})|J| + b_i(\mathbf{x})\frac{d|J|}{dt}\bigg)\mathrm{d}\Omega_0\bigg|_{t=0}\end{aligned} \tag{A.2}$$

where $\hat{\mathbf{V}}(\mathbf{x}) = \frac{d\mathbf{x}}{dt}$ is the design velocity vector at any point $\mathbf{x} \in \Omega$. Now, using the fact that $\frac{d|J|}{dt} = |J|\nabla.\hat{\mathbf{V}}(\mathbf{x})$ [75], we get

$$\begin{aligned}\frac{d}{dt}\left(\int_{\Omega_0} b_i(\mathbf{x})\,|J|\mathrm{d}\Omega_0\right)\Bigg|_{t=0} &= \int_{\Omega_0}\Big(\nabla b_i(\mathbf{x}).\hat{\mathbf{V}}(\mathbf{x})|J| + b_i(\mathbf{x})|J|\nabla.\hat{\mathbf{V}}(\mathbf{x})\Big)\mathrm{d}\Omega_0\Bigg|_{t=0}\\ &= \int_{\Omega_0}\Big(\nabla b_i(\mathbf{x}).\hat{\mathbf{V}}(\mathbf{x}) + b_i(\mathbf{x})\nabla.\hat{\mathbf{V}}(\mathbf{x})\Big)|J|\mathrm{d}\Omega_0\Bigg|_{t=0}\\ &= \int_{\Omega_0}\nabla.(b_i(\mathbf{x})\hat{\mathbf{V}}(\mathbf{x}))|J|\mathrm{d}\Omega_0\Bigg|_{t=0} \qquad \text{(A.3)}\end{aligned}$$

evaluating the RHS of the above at $t = 0$, we obtain the desired sensitivity as

$$\frac{dM_i}{dt}\Bigg|_{t=0} = \int_{\Omega_0}\nabla.(b_i(\mathbf{X})\hat{\mathbf{V}}(\mathbf{X}))\mathrm{d}\Omega_0 \qquad \text{(A.4)}$$

where $\mathbf{X} \in \Omega_0$.

Applying the divergence theorem we obtain the following boundary form

$$\boxed{\frac{dM_i}{dt}\Bigg|_{t=0} = \int_{\Gamma_0} b_i(\mathbf{X})V_N(\mathbf{X})\mathrm{d}\Gamma_0 \qquad \text{(A.5)}}$$

where $V_N = \hat{\mathbf{V}}(\mathbf{X}).\mathbf{N}(\mathbf{X})$ is the normal component of the design velocity at $\mathbf{X} \in \Gamma_0$.

Second-order SSA of Moments Now, It is required to compute the second-order shape sensitivity of the moments M_i i.e. $\frac{d^2M_i}{dt^2}\Big|_{t=0}$. This can be done by taking the first derivative of Eq.(A.3) as

$$\frac{d^2M_i}{dt^2} = \frac{d}{dt}\int_{\Omega_0}\Big(\nabla.(b_i(\mathbf{x})\hat{\mathbf{V}}(\mathbf{x}))|J|\mathrm{d}\Omega_0\Big) \qquad \text{(A.6)}$$

taking the derivative inside the integral and then applying the product and chain rule of differentiation we get :

$$\begin{aligned}\int_{\Omega_0}\frac{d}{dt}\Big(\nabla.(b_i(\mathbf{x})\hat{\mathbf{V}}(\mathbf{x}))|J|\Big)\mathrm{d}\Omega_0 &= \int_{\Omega_0}\frac{d}{dt}\Big(\nabla.(b_i(\mathbf{x})\hat{\mathbf{V}}(\mathbf{x}))\Big)|J|\\ &+ \nabla.(b_i(\mathbf{x})\hat{\mathbf{V}}(\mathbf{x}))\frac{d|J|}{dt}d\Omega_0 \qquad \text{(A.7)}\end{aligned}$$

However,

$$\begin{aligned}\frac{d}{dt}\Big(\nabla.(b_i(\mathbf{x})\hat{\mathbf{V}}(\mathbf{x}))\Big) &= \frac{\partial}{\partial t}(\nabla.(b_i(\mathbf{x})\hat{\mathbf{V}}(\mathbf{x})) + \nabla(\nabla.(b_i\hat{\mathbf{V}}(\mathbf{x}))).\hat{\mathbf{V}}(\mathbf{x}) \\ &= \nabla.(b_i(\mathbf{x})\frac{\partial}{\partial t}\hat{\mathbf{V}}(\mathbf{x})) + \nabla(\nabla.(b_i\hat{\mathbf{V}}(\mathbf{x}))).\hat{\mathbf{V}}(\mathbf{x}) \qquad \text{(A.8)}\end{aligned}$$

Since we assume a linear mapping, we have the following relationship for the total time derivative of the design velocity :

$$\frac{d}{dt}\hat{\mathbf{V}}(\mathbf{x}) = \frac{\partial}{\partial t}\hat{\mathbf{V}}(\mathbf{x}) + \nabla\hat{\mathbf{V}}(\mathbf{x}).\hat{\mathbf{V}}(\mathbf{x}) = 0 \qquad \text{(A.9)}$$

Thus, $\frac{\partial}{\partial t}\hat{\mathbf{V}}(\mathbf{x}) = -\nabla\hat{\mathbf{V}}(\mathbf{x}).\hat{\mathbf{V}}(\mathbf{x})$. Using this and $\frac{d|J|}{dt} = |J|\nabla.\hat{\mathbf{V}}(\mathbf{x})$ [75] in Eq.(A.7) the second derivative reduces to :

$$\begin{aligned}\frac{d^2M_i}{dt^2} &= \int_{\Omega_0}\Big(-\nabla.\Big[b_i(\mathbf{x})\nabla\hat{\mathbf{V}}(\mathbf{x}).\hat{\mathbf{V}}(\mathbf{x})\Big] + \nabla(\nabla.(b_i(\mathbf{x})\hat{\mathbf{V}}(\mathbf{x}))).\hat{\mathbf{V}}(\mathbf{x}) + \nabla.(b_i(\mathbf{x})\hat{\mathbf{V}}(\mathbf{x}))\nabla.(\hat{\mathbf{V}}(\mathbf{x})\Big)|J|d\Omega_0 \\ &= \int_{\Omega_0}\Big(\nabla.\Big[\nabla.(b_i(\mathbf{x})\hat{\mathbf{V}}(\mathbf{x}))\hat{\mathbf{V}}(\mathbf{x})\Big] - \nabla.\Big[b_i(\mathbf{x})\nabla\hat{\mathbf{V}}(\mathbf{x}).\hat{\mathbf{V}}(\mathbf{x})\Big]\Big)|J|d\Omega_0 \qquad \text{(A.10)}\end{aligned}$$

evaluating the the above at $t = 0$, we obtain the second-order shape sensitivity as :

$$\left.\frac{d^2M_i}{dt^2}\right|_{t=0} = \int_{\Omega_0}\Big(\nabla.\Big[\nabla.(b_i(\mathbf{x})\hat{\mathbf{V}}(\mathbf{X}))\hat{\mathbf{V}}(\mathbf{X})\Big] - \nabla.\Big[b_i(\mathbf{X})\nabla\hat{\mathbf{V}}(\mathbf{X}).\hat{\mathbf{V}}(\mathbf{X})\Big]d\Omega_0\Big) \qquad \text{(A.11)}$$

where $\mathbf{X} \in \Omega_0$.

Applying the divergence theorem we get the boundary form of the second order shape sensitivity as :

$$\begin{aligned}\left.\frac{d^2M_i}{dt^2}\right|_{t=0} &= \int_{\Gamma_0}\left[\nabla.[b_i(\mathbf{X})\hat{\mathbf{V}}(\mathbf{X})]V_N(\mathbf{X}) - b_i(\mathbf{X})\Big(\nabla\hat{\mathbf{V}}(\mathbf{X}).\hat{\mathbf{V}}(\mathbf{X})\Big).\mathbf{N}(\mathbf{X})\right]d\Gamma_0 \\ &= \int_{\Gamma_0}\left[\nabla[b_i(\mathbf{X})]\hat{\mathbf{V}}(\mathbf{X}) + b_i(\mathbf{X})\nabla.(\hat{\mathbf{V}}(\mathbf{X})) - b_i(\mathbf{X})\Big(\mathbf{N}^T\nabla\hat{\mathbf{V}}\mathbf{N}\Big)\right]V_N(\mathbf{X})d\Gamma_0 \\ &= \int_{\Gamma_0}\left[\nabla[b_i(\mathbf{X})].\hat{\mathbf{V}}(\mathbf{X}) + b_i(\mathbf{X})\Big(\nabla.(\hat{\mathbf{V}}(\mathbf{X})) - \mathbf{N}^T\nabla\hat{\mathbf{V}}\mathbf{N}\Big)\right]V_N(\mathbf{X})d\Gamma_0 \quad (A.12)\end{aligned}$$

It can be easily proved [75] that $[\nabla.(\mathbf{V}) - \mathbf{N}^T\nabla\mathbf{V}\mathbf{N}] = V_N(\mathbf{X})\kappa(\mathbf{X})$, where κ is the curvature of Γ_0 in 2D and twice the mean curvature in 3D.

Thus,

$$\boxed{\left.\frac{d^2M_i}{dt^2}\right|_{t=0} = \int_{\Gamma_0}[\nabla b_i(\mathbf{X})]^T\mathbf{N} + b_i(\mathbf{X})\kappa(\mathbf{X})]V_N^2(\mathbf{X})d\Gamma_0 \qquad (A.13)}$$

If we assume that the vertices/edges of the approximate polygon/polyhedron Ω_0 all lie on Γ as shown in Fig. A.1, then $V_N = 0$ at the vertices/edges of Ω_0. Also, $\kappa = 0$ in the interior of the edges/faces of the Γ_0.Hence, we can easily prove that $\int_{\Gamma_0}\kappa(\mathbf{X})V_N^2(\mathbf{X})d\Gamma_0 = 0$ as the integrand $[\kappa(\mathbf{X})V_N^2(\mathbf{X})]$ is identically zero. Hence, the above reduces to the following :

$$\boxed{\left.\frac{dM_i^2}{dt^2}\right|_{t=0} = \int_{\Gamma_0}[\nabla b_i(\mathbf{X})]^T\mathbf{N}V_N^2(\mathbf{X})d\Gamma_0 \qquad (A.14)}$$

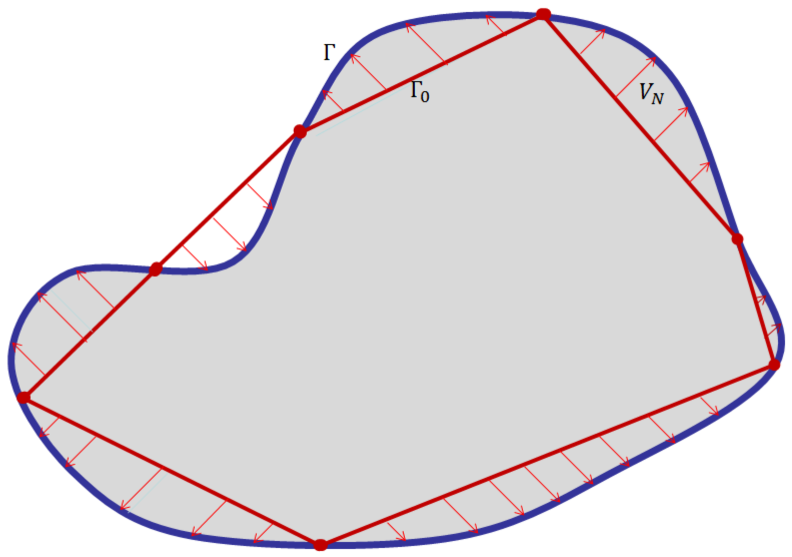


Figure A.1: Vertices (black dots) of Γ_0 lying on Γ makes $V_N = 0$ at the vertices

Appendix B

TSA of Moments

First-order TSA of Moments First-order topological sensitivity of moments at a point $\mathbf{X_0}$ can be established using the Topological-Shape Sensitivity definition as defined by Novotny et al. [17] as

$$D^1_{T_i}(\mathbf{X_0}) = \lim_{\epsilon\to 0} \frac{1}{f_1'(\epsilon)|V_N|} \left.\frac{dM_i}{dt}\right|_{t=0} \tag{B.1}$$

From Eq.(A.5) we have already established the first-order SSA as

$$\left.\frac{dM_i}{dt}\right|_{t=0} = \int_{\Gamma_0} b_i(\mathbf{X})V_N(\mathbf{X})\mathrm{d}\Gamma_0 \tag{B.2}$$

where Γ_0 is now the circular boundary B_ϵ as shown in Fig. B.1. Thus, we have

$$\left.\frac{dM_i}{dt}\right|_{t=0} = \int_{\partial B_\epsilon} b_i(\mathbf{X})V_N(\mathbf{X})\mathrm{d}B_\epsilon \tag{B.3}$$

Using Eq.(B.3) in Eq.(B.1), we get

$$D^1_T(\mathbf{X_0}) = \lim_{\epsilon\to 0} \frac{1}{f_1'(\epsilon)|V_N|} \int_{\partial B_\epsilon} b_i(\mathbf{X})V_N(\mathbf{X})\mathrm{d}B_\epsilon \tag{B.4}$$

Since V_N is a constant over the boundary of the ball (∂B_ϵ) we have

$$\begin{aligned} D_T^1(\mathbf{X_0}) &= \lim_{\epsilon\to 0}\frac{V_N}{f_1'(\epsilon)|V_N|}\int_{\partial B_\epsilon} b_i(\mathbf{X})\mathrm{d}B_\epsilon \\ &= \lim_{\epsilon\to 0}\frac{sign(V_N)}{f_1'(\epsilon)}\int_{\partial B_\epsilon} b_i(\mathbf{X})\mathrm{d}B_\epsilon \\ &= \lim_{\epsilon\to 0}\frac{-1}{f_1'(\epsilon)}\int_{\partial B_\epsilon} b_i(\mathbf{X})\mathrm{d}B_\epsilon \end{aligned} \tag{B.5}$$

For 2D problems, $f_1(\epsilon) = \pi\epsilon^2$ and hence $f_1'(\epsilon) = 2\pi\epsilon$. If we assume bivariate basis functions of the form $b_i = X^p Y^q$ and then write out the above integral in polar coordinates we get

$$\begin{aligned} D_T^1(\mathbf{X_0}) &= \lim_{\epsilon\to 0}\frac{-1}{2\pi\epsilon}\int_0^{2\pi}(X_0+\epsilon cos(\theta))^p(Y_0+\epsilon sin(\theta))^q\epsilon\mathrm{d}\theta \\ &= \lim_{\epsilon\to 0}\frac{-1}{2\pi}\int_0^{2\pi}(X_0+\epsilon cos(\theta))^p(Y_0+\epsilon sin(\theta))^q\mathrm{d}\theta \\ &= \frac{-1}{2\pi}\left[X_0^pY_0^q\int_0^{2\pi}\mathrm{d}\theta+\lim_{\epsilon\to 0}\int_0^{2\pi}[\binom{q}{1}X_0^pY_0^{q-1}\epsilon sin(\theta)+\binom{p}{1}Y_0^qX_0^{p-1}\epsilon cos(\theta)+O(\epsilon^2)]\mathrm{d}\theta\right] \\ &= -X_0^pY_0^q \end{aligned} \tag{B.6}$$

where we have used binomial expansion in the third step above to apply the limit. Thus, in 2D the first-order TSA is given by

$$\boxed{D_T^1(\mathbf{X_0}) = -X_0^pY_0^q\ \forall\mathbf{X_0}\in\Omega\subset\mathbb{R}^2 \text{ with } f_1(\epsilon)=\pi\epsilon^2} \tag{B.7}$$

Likewise in 3D, for trivariate polynomials of the form $b_i = X^pY^qZ^r$ and $f_1(\epsilon) = \frac{4}{3}\pi\epsilon^3$, one can easily establish the first-order topological derivative to be

$$\boxed{D_T^1(\mathbf{X_0}) = -X_0^pY_0^qZ_0^r\ \forall\mathbf{X_0}\in\Omega\subset\mathbb{R}^3 \text{ with } f_1(\epsilon)=\frac{4}{3}\pi\epsilon^3} \tag{B.8}$$

by using the fact that $X = X_0 + \epsilon cos(\phi)sin(\theta)$, $Y = Y_0 + \epsilon sin(\phi)sin(\theta)$ and $Z = Z_0 + \epsilon cos(\theta)$ where $\theta\in[0,\pi]$ and $\phi\in[0,2\pi]$.

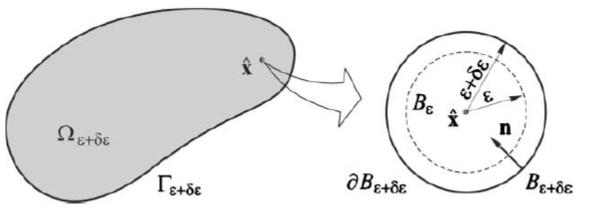

$\hat{\mathbf{x}}$
$\Omega_{\varepsilon+\delta\varepsilon}$
$\Gamma_{\varepsilon+\delta\varepsilon}$
B_ε
$\varepsilon+\delta\varepsilon$
ε
$\mathbf{n}$
$\partial B_{\varepsilon+\delta\varepsilon}$
$B_{\varepsilon+\delta\varepsilon}$

Figure B.1: Reference and deformed domains for Topological-Shape Sensitivity Analysis [17]

Second-order TSA of Moments The second-order TSA of moments can be defined using the Topological-Shape sensitivity method of [34] as

$$\begin{aligned} D_T^2(\mathbf{X_0}) &= \lim_{\epsilon\to 0}\frac{1}{f_2'(\epsilon)}\left[\frac{1}{|V_N|}\frac{dM_i}{dt}\bigg|_{t=0} - f_1'(\epsilon)D_T^1(\mathbf{X_0})\right] \\ &= \lim_{\epsilon\to 0}\frac{1}{f_2'(\epsilon)}\left[\frac{1}{|V_N|}\int_{\partial B_\epsilon} b_i(\mathbf{X})V_N(\mathbf{X})\mathrm{d}B_\epsilon - f_1'(\epsilon)D_T^1(\mathbf{X_0})\right] \\ &= \lim_{\epsilon\to 0}\frac{1}{f_2'(\epsilon)}\left[sign(V_N)\int_{\partial B_\epsilon} b_i(\mathbf{X})\mathrm{d}B_\epsilon - f_1'(\epsilon)D_T^1(\mathbf{X_0})\right] \\ &= \lim_{\epsilon\to 0}\frac{1}{f_2'(\epsilon)}\left[-\int_{\partial B_\epsilon} b_i(\mathbf{X})\mathrm{d}B_\epsilon - f_1'(\epsilon)D_T^1(\mathbf{X_0})\right] \end{aligned} \tag{B.9}$$

For 2D domains if we choose the bivariate polynomials $b_i = X^pY^q$ then $f_1'(\epsilon)D_T^1(\mathbf{X_0}) = -2\pi\epsilon X_0^pY_0^q$ and $f_2(\epsilon) = \frac{1}{2}\pi^2\epsilon^4$ (this is chosen such that $f_1(\epsilon)\to 0$, $f_2(\epsilon)\to 0$ and $\frac{f_2(\epsilon)}{f_1(\epsilon)}\to 0$ as $\epsilon\to 0$). Therefore, Eq.(B.9) becomes

$$\begin{aligned} D_T^2(\mathbf{X_0}) &= \lim_{\epsilon\to 0}\frac{1}{2\pi^2\epsilon^3}\left[-\int_0^{2\pi}(X_0+\epsilon cos(\theta))^p(Y_0+\epsilon sin(\theta))^q\epsilon\mathrm{d}\theta + 2\pi\epsilon X_0^pY_0^q\right] \\ &= \lim_{\epsilon\to 0}\frac{1}{2\pi^2\epsilon^2}\left[-\int_0^{2\pi}(X_0+\epsilon cos(\theta))^p(Y_0+\epsilon sin(\theta))^q\mathrm{d}\theta + 2\pi X_0^pY_0^q\right] \end{aligned} \tag{B.10}$$

Applying binomial theorem to the above and simplifying we get

$$\begin{aligned}
D^2_T(\mathbf{X_0}) &= \lim_{\epsilon\to 0}\frac{1}{2\pi^2\epsilon^2}\Bigg[-2\pi X_0^pY_0^q - \int_0^{2\pi}\Bigg[\binom{q}{1}X_0^pY_0^{q-1}\epsilon sin(\theta) + \binom{p}{1}Y_0^qX_0^{p-1}\epsilon cos(\theta)\\
&+ \binom{q}{2}X_0^pY_0^{q-2}\epsilon^2 sin^2(\theta) + \binom{p}{2}Y_0^qX_0^{p-2}\epsilon^2 cos^2(\theta) + \binom{p}{1}\binom{q}{1}X_0^{p-1}Y_0^{q-1}\epsilon^2 cos(\theta)sin(\theta)\\
&+ O(\epsilon^3)\Bigg]d\theta + 2\pi X_0^pY_0^q\Bigg]\\
&= \lim_{\epsilon\to 0}\frac{-1}{2\pi^2\epsilon^2}\Bigg[\int_0^{2\pi}\Bigg[\binom{q}{2}X_0^pY_0^{q-2}\epsilon^2 sin^2(\theta) + \binom{p}{2}Y_0^qX_0^{p-2}\epsilon^2 cos^2(\theta) + O(\epsilon^3)\Bigg]d\theta\\
&= \frac{-1}{2\pi^2}\Bigg[\int_0^{2\pi}\Bigg[\binom{q}{2}X_0^pY_0^{q-2}sin^2(\theta) + \binom{p}{2}Y_0^qX_0^{p-2}cos^2(\theta)\Bigg]d\theta\Bigg]\\
&= \frac{-1}{2\pi^2}\Bigg[\binom{q}{2}X_0^pY_0^{q-2}\pi + \binom{p}{2}Y_0^qX_0^{p-2}\pi\Bigg]\\
&= \frac{-1}{2\pi}\Bigg[\binom{q}{2}X_0^pY_0^{q-2} + \binom{p}{2}Y_0^qX_0^{p-2}\Bigg]
\end{aligned}$$

Thus, second-order TSA in 2D is given by

$$\boxed{D^2_T(\mathbf{X_0}) = \frac{-1}{2\pi}\Bigg[\binom{p}{2}Y_0^qX_0^{p-2} + \binom{q}{2}X_0^pY_0^{q-2}\Bigg]\ \forall \mathbf{X_0}\in\Omega\subset\mathbb{R}^2 \text{ with } f_2(\epsilon) = \frac{1}{2}\pi^2\epsilon^4 \qquad \text{(B.11)}}$$

For 3D domains if we choose the trivariate polynomials $b_i = X^pY^qZ^r$ then $f_1^{'}(\epsilon)D^1_T(\mathbf{X_0}) = -4\pi\epsilon^2 X_0^pY_0^qZ^r$ and $f_2(\epsilon) = \frac{1}{2\epsilon}[\frac{4}{3}\pi\epsilon^3]^2$. Therefore, Eq.(B.9) becomes

$$\begin{aligned}
D^2_T(\mathbf{X_0}) &= \lim_{\epsilon\to 0}\frac{3}{16\pi^2\epsilon^4}\Bigg[-\Bigg[\int_0^{\pi}[\int_0^{2\pi}(X_0+\epsilon cos(\phi)sin(\theta))^p(Y_0+\epsilon sin(\phi)sin(\theta))^q\\
(&\quad Z_0+\epsilon cos(\theta))^r\epsilon^2\mathrm{d}\phi]sin(\theta)d\theta\Bigg] + 4\pi\epsilon^2X_0^pY_0^qZ_0^r\Bigg]\\
&= \lim_{\epsilon\to 0}\frac{3}{16\pi^2\epsilon^2}\Bigg[\Bigg[-\int_0^{\pi}[\int_0^{2\pi}(X_0+\epsilon cos(\phi)sin(\theta))^p(Y_0+\epsilon sin(\phi)sin(\theta))^q\mathrm{d}\phi]\\
(&\quad Z_0+\epsilon cos(\theta))^r sin(\theta)d\theta\Bigg] + 4\pi X_0^pY_0^qZ_0^r\Bigg] \qquad \text{(B.12)}
\end{aligned}$$

Let us first expand the two terms in the inner integral using binomial theorem

$$\begin{aligned}(X_0+\epsilon cos(\phi)sin(\theta))^p &= X_0^p+\epsilon\binom{p}{1}X_0^{p-1}cos(\phi)sin(\theta)+\epsilon^2\binom{p}{2}X_0^{p-2}cos^2(\phi)sin^2(\theta)\\ &+ \epsilon^3\binom{p}{3}X_0^{p-3}cos^3(\phi)sin^3(\theta)+O(\epsilon^4) \qquad \text{(B.13)}\\ (Y_0+\epsilon sin(\phi)sin(\theta))^q &= Y_0^q+\epsilon\binom{q}{1}Y_0^{q-1}sin(\phi)sin(\theta)+\epsilon^2\binom{q}{2}Y_0^{q-2}sin^2(\phi)sin^2(\theta)\\ &+ \epsilon^3\binom{q}{3}Y_0^{q-3}sin^3(\phi)sin^3(\theta)+O(\epsilon^4) \qquad \text{(B.14)}\end{aligned}$$

Now,

$$\begin{aligned}(X_0+\epsilon cos(\phi)sin(\theta))^p(Y_0+\epsilon sin(\phi)sin(\theta))^q &= X_0^pY_0^q+\epsilon\binom{q}{1}Y_0^{q-1}X_0^psin(\phi)sin(\theta)\\ &+ \epsilon^2\binom{q}{2}Y_0^{q-2}X_0^psin^2(\phi)sin^2(\theta)\\ &+ \epsilon^3\binom{q}{3}Y_0^{q-3}X_0^psin^3(\phi)sin^3(\theta)\\ &+ \epsilon\binom{p}{1}X_0^{p-1}Y_0^qcos(\phi)sin(\theta)\\ &+ \epsilon^2\binom{p}{1}\binom{q}{1}X_0^{p-1}Y_0^{q-1}sin(\phi)cos(\phi)sin^2(\theta)\\ &+ \epsilon^3\binom{p}{1}\binom{q}{2}X_0^{p-1}Y_0^{q-2}cos(\phi)sin^2(\phi)sin^3(\theta)\\ &+ \epsilon^2\binom{p}{2}X_0^{p-2}Y_0^qcos^2(\phi)sin^2(\theta)\\ &+ \epsilon^3\binom{p}{2}\binom{q}{1}X_0^{p-2}Y_0^{q-1}sin(\phi)cos^2(\phi)sin^3(\theta)\\ &+ \epsilon^3\binom{p}{3}X_0^{p-3}Y_0^qcos^3(\phi)sin^3(\theta)+O(\epsilon^4)\end{aligned}$$

Now, integrating the above w.r.t. ϕ over the interval $[0, 2\pi]$ we get

$$\begin{aligned}\int_0^{2\pi}(X_0+\epsilon cos(\phi)sin(\theta))^p(Y_0+\epsilon sin(\phi)sin(\theta))^q d\phi &= 2\pi X_0^pY_0^q+\epsilon^2\binom{q}{2}Y_0^{q-2}X_0^p(\pi)sin^2(\theta)\\ &+ \epsilon^2\binom{p}{2}X_0^{p-2}Y_0^q(\pi)sin^2(\theta)\\ &+ O(\epsilon^4) \qquad (B.15)\end{aligned}$$

Now, using binomial theorem to expand $(Z_0+\epsilon cos(\theta))^r$ we get

$$\left(Z_0+\epsilon cos(\theta)\right)^r = Z_0^r+\epsilon\binom{r}{1}Z_0^{r-1}cos(\theta)+\epsilon^2\binom{r}{2}Z_0^{r-2}cos^2(\theta)+\epsilon^3\binom{r}{3}Z_0^{r-3}cos^3(\theta)+O(\epsilon^4) \qquad (B.16)$$

Now, let us evaluate the double integral in Eq.(B.12) using Eq.(B.15) and Eq.(B.16)

$$\begin{aligned}&\int_0^{\pi}[\int_0^{2\pi}(X_0+\epsilon cos(\phi)sin(\theta))^p(Y_0+\epsilon sin(\phi)sin(\theta))^q \mathrm{d}\phi](Z_0+\epsilon cos(\theta))^r sin(\theta)d\theta\Big]\\ &= \pi\int_0^{\pi}\left(2X_0^pY_0^q+\epsilon^2 sin^2(\theta)[\binom{q}{2}Y_0^{q-2}X_0^p+\binom{p}{2}X_0^{p-2}Y_0^q]+O(\epsilon^4)\right)\\ &\left(Z_0^r+\epsilon\binom{r}{1}Z_0^{r-1}cos(\theta)+\epsilon^2\binom{r}{2}Z_0^{r-2}cos^2(\theta)+\epsilon^3\binom{r}{3}Z_0^{r-3}cos^3(\theta)+O(\epsilon^4)\right)sin(\theta)d\theta\\ &= 4\pi X_0^pY_0^qZ_0^r+\epsilon^2\frac{4\pi}{3}\left[\binom{r}{2}X_0^pY_0^qZ_0^{r-2}+\binom{q}{2}Z_0^rY_0^{q-2}X_0^p+\binom{p}{2}Z_0^rX_0^{p-2}Y_0^q\right]+O(\epsilon^4) \qquad (B.17)\end{aligned}$$

Now, substituting the above in Eq.(B.12) and taking the limit we get the second-order TSA in 3D as

$$D_T^2(\mathbf{X_0}) = -\frac{1}{4\pi}\left[\binom{p}{2}X_0^{p-2}Y_0^qZ_0^r+\binom{q}{2}X_0^pY_0^{q-2}Z_0^r+\binom{r}{2}X_0^pY_0^qZ_0^{r-2}\right] \qquad (B.18)$$

Thus, second-order TSA in 3D is given by

$$\begin{aligned} D_T^2(\mathbf{X_0}) &= -\frac{1}{4\pi}\left[\binom{p}{2}X_0^{p-2}Y_0^qZ_0^r + \binom{q}{2}X_0^pY_0^{q-2}Z_0^r + \binom{r}{2}X_0^pY_0^qZ_0^{r-2}\right] \\ &\forall \quad \mathbf{X_0} \in \Omega \subset \mathbb{R}^3 \text{ with } f_2(\epsilon) = \frac{8}{9}\pi^2\epsilon^5 \end{aligned} \tag{B.19}$$

The first/second-order SSA and TSA of moments are summarized in Table. (3.1) and Table. (3.2) respectively

Appendix C

Taylor's Remainder Theorem

Let $M_i : [0, \alpha] \to \mathbb{R}$ be a $\mathscr{C}^r$ continuous function and for $k \in \mathbb{Z}^+$ let $P_k(t) = M_i(0) + t.M_i^{'}(0) + \frac{t^2}{2!}M_i^{''}(0) + ... + \frac{t^k}{k!}M_i^k(0)$ ($r \geq k+1$). Then, for some $\epsilon \in [0, \alpha]$ we have the following error bound

$$|E_k(t)| = |M_i(t) - P_k(t)| \leq \frac{|t^{k+1}|}{(k+1)!}|M_i^{k+1}(\epsilon)| \tag{C.1}$$

Proof. We begin by noting that the following follows directly from the definition of $P_k(t)$

$$\begin{aligned} |E_k(0)| &= \quad |M_i(0) - P_k(0)| = 0 \\ |E_k^{'}(0)| &= \quad |M_i^{'}(0) - P_k^{'}(0)| = 0 \\ &. \\ &. \\ |E_k^{k}(0)| &= \quad |M_i^{k}(0) - P_k^{k}(0)| = 0 \end{aligned}$$

Likewise, since P_k is a k^{th} degree polynomial in t we have

$$|E_k^{k+1}(t)| = |M_i^{k+1}(t) - P_k^{k+1}(t)| = |M_i^{k+1}(t) - 0| = |M_i^{k+1}(t)|$$

since M_i^{k+1} is a continuous function defined over a compact set $[0, \alpha]$, it attains its maximum for some $\epsilon \in [0, \alpha]$ [111] and hence we have

$$|E_k^{k+1}(t)| \leq |M_i^{k+1}(\epsilon)|$$

This also means

$$E_k^{k+1}(t) \leq |M_i^{k+1}(\epsilon)|$$

integrating the above once we get

$$E_k^k(t) = \int E_k^{k+1}(t)dt \leq \int |M_i^{k+1}(\epsilon)|dt = |M_i^{k+1}(\epsilon)|t + c_1$$

However, $c_1 = 0$ as at $t = 0$ we have $E_k^k(0) = 0$ and therefore

$$E_k^k(t) \leq |M_i^{k+1}(\epsilon)|t$$

integrating the above k times and applying boundary conditions as before we have

$$E_k(t) \leq \frac{t^{k+1}}{(k+1)!}|M_i^{k+1}(\epsilon)|$$

and the result follows by taking the absolute value on both sides. $\square$

Appendix D

Eigen Value Lemma

Suppose $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{x} \in \mathbb{R}^n$, $\|\mathbf{x}\|_2 = 1$ *and* λ_{max} *be the maximum eigenvalue of* $\mathbf{A}^T\mathbf{A}$ *then*

$$\|\mathbf{A}\mathbf{x}\|_2 \leq \sqrt{\lambda_{max}} \tag{D.1}$$

Proof. First we observe that $\mathbf{A}^T\mathbf{A}$ is a symmetric positive semi-definite matrix and hence its eigenvalues are all non-negative. Now, let us consider the following optimization problem

$$\begin{aligned} \underset{\mathbf{x}}{\text{maximize}} \quad & \mathbf{x}^T(\mathbf{A}^T\mathbf{A})\mathbf{x} \\ \text{subject to} \quad & \|\mathbf{x}\|_2 = 1 \end{aligned}$$

This can be converted into an unconstrained optimization problem using the Lagrange multiplier method as

$$\underset{\mathbf{x}}{\text{maximize}} \quad \mathbf{x}^T(\mathbf{A}^T\mathbf{A})x - \lambda(\mathbf{x}^T\mathbf{x} - 1)$$

The KKT conditions [29] for this problem gives us the necessary optimality condition as

$$(\mathbf{A}^T\mathbf{A})\mathbf{x} = \lambda\mathbf{x}$$

In other words, the quadratic form $\mathbf{x}^T(\mathbf{A}^T\mathbf{A})\mathbf{x}$ is maximized when λ corresponds to the maximum eigenvalue and $\mathbf{x}$ the corresponding eigenvector of $(\mathbf{A}^T\mathbf{A})$. This implies that for any unit vector $\mathbf{x} \in \mathbb{R}^n$ we have $\mathbf{x}^T(\mathbf{A}^T\mathbf{A})\mathbf{x} = (\mathbf{A}\mathbf{x})^T(\mathbf{A}\mathbf{x}) = \|\mathbf{A}\mathbf{x}\|_2^2 \leq \lambda_{max}$ $\square$

Appendix E

Computed Basis Function Values

Table I: Integral of basis functions over the quadrant

Basis Function	**Integral Value**				**Relative Error (%)**		
	C	**SS**	**SAQ**	**Analytical**	**C**	**SS**	**SAQ**
b00 = x^0y^0	0.7890	0.7850	0.7850	0.7854	0.4609	0.0497	0.0460
b10 = x^1y^0	0.3372	0.3334	0.3333	0.3333	1.1685	**0.0120**	**0.0122**
b01 = x^0y^1	0.3333	0.3332	0.3331	0.3333	**0.0000**	0.0487	0.0703
b20 = x^2y^0	0.2006	0.1944	0.1945	0.1963	2.1605	1.0001	0.9365
b11 = x^1y^1	0.1250	0.1247	0.1248	0.1250	0.0000	0.2166	0.1893
b02 = x^0y^2	0.1961	0.1962	0.1963	0.1963	0.1056	0.0676	0.0280
b30 = x^3y^0	0.1380	0.1309	0.1310	0.1333	3.5009	**1.8206**	**1.7183**
b21 = x^2y^1	0.0667	0.0662	0.0662	0.0667	0.0000	0.6758	0.6660
b12 = x^1y^2	0.0664	0.0665	0.0665	0.0667	0.3864	0.2352	0.2082
b03 = x^0y^3	0.1333	0.1332	0.1333	0.1333	0.0000	0.1014	0.0414
b40 = x^4y^0	0.1035	0.0965	0.0965	0.0982	5.4327	1.7039	1.6840
b31 = x^3y^1	0.0417	0.0411	0.0411	0.0417	0.0000	1.2972	1.2858
b22 = x^2y^2	0.0324	0.0326	0.0326	0.0327	1.1118	0.3209	0.4688
b13 = x^1y^3	0.0417	0.0416	0.0416	0.0417	0.0000	0.2289	0.2513
b04 = x^0y^4	0.0983	0.0981	0.0981	0.0982	0.0956	0.0649	0.0660
b50 = x^5y^0	0.0825	0.0760	0.0759	0.0762	**8.2698**	0.1982	0.4042
b41 = x^4y^1	0.0287	0.0282	0.0282	0.0286	0.6250	1.3631	1.2967
b32 = x^3y^2	0.0185	0.0188	0.0188	0.0190	2.8577	1.2174	1.5349
b23 = x^2y^3	0.0190	0.0191	0.0190	0.0190	0.4687	0.1990	0.1494
b14 = x^1y^4	0.0287	0.0285	0.0285	0.0286	0.6021	0.1579	0.2575
b05 = x^0y^5	0.0763	0.0762	0.0761	0.0762	0.0781	0.0422	0.1128

Table II: Integral of basis functions over the notched domain

Basis Functions	**Integral Value**				**Relative Error (%)**		
	C	**SS**	**SAQ**	**Analytical**	**C**	**SS**	**SAQ**
b00 = x^0y^0	0.9556	0.9845	0.9845	0.9843	2.9195	0.0256	0.0244
b10 = x^1y^0	0.4778	0.4922	0.4922	0.4921	2.9195	**0.0082**	**0.0096**
b01 = x^0y^1	0.4578	0.4843	0.4845	0.4850	5.6048	0.1274	0.1024
b20 = x^2y^0	0.3222	0.3294	0.3294	0.3294	2.1693	0.0182	0.0181
b11 = x^1y^1	0.2289	0.2429	0.2429	0.2425	5.6048	0.1856	0.1595
b02 = x^0y^2	0.2932	0.3198	0.3197	0.3189	8.0692	0.2904	0.2474
b30 = x^3y^0	0.2444	0.2481	0.2481	0.2480	1.4248	0.0315	0.0295
b21 = x^2y^1	0.1561	0.1627	0.1627	0.1629	4.1489	0.1151	0.0922
b12 = x^1y^2	0.1466	0.1594	0.1594	0.1595	8.0692	0.0311	0.0218
b03 = x^0y^3	0.2118	0.2372	0.2371	0.2362	10.3262	0.4531	0.3820
b40 = x^4y^0	0.1972	0.1990	0.1990	0.1990	0.8730	0.0401	0.0355
b31 = x^3y^1	0.1197	0.1225	0.1226	0.1231	2.7146	0.4730	0.3917
b22 = x^2y^2	0.1011	0.1083	0.1081	0.1075	5.9522	0.7364	0.6204
b13 = x^1y^3	0.1059	0.1174	0.1175	0.1181	10.3262	0.5875	0.4889
b04 = x^0y^4	0.1636	0.1868	0.1868	0.1867	12.3890	0.0544	0.0458
b50 = x^5y^0	0.1653	0.1662	0.1662	0.1661	**0.5107**	0.0436	0.0357
b41 = x^4y^1	0.0974	0.0983	0.0984	0.0990	1.6582	0.7489	0.6254
b32 = x^3y^2	0.0783	0.0827	0.0825	0.0815	3.8806	**1.5504**	**1.3015**
b23 = x^2y^3	0.0738	0.0810	0.0808	0.0798	7.5919	1.4125	1.1856
b14 = x^1y^4	0.0818	0.0919	0.0922	0.0934	12.3890	1.5496	1.2967
b05 = x^0y^5	0.1320	0.1526	0.1528	0.1539	**14.2709**	0.8892	0.7470

Table III: Integral of basis functions over the wavy domain

Basis Functions	**Integral Value**				**Relative Error (%)**		
	GA	**SS**	**SAQ**	**Analytical**	**GA**	**SS**	**SAQ**
b00 = x^0y^0	6.2743	6.4135	6.4238	6.4055	2.0479	0.1255	0.2866
b10 = x^1y^0	6.2743	6.4115	6.4159	6.4055	2.0479	0.0951	0.1636
b01 = x^0y^1	9.8417	10.2718	10.2772	10.2615	4.0907	0.1002	0.1524
b20 = x^2y^0	8.3715	8.5511	8.5525	8.5442	2.0217	0.0800	0.0969
b11 = x^1y^1	9.8417	10.2704	10.2716	10.2615	4.0907	0.0864	0.0985
b02 = x^0y^2	20.5837	21.9308	21.9259	21.9271	6.1268	0.0167	0.0055
b30 = x^3y^0	12.5659	12.8300	12.8296	12.8218	1.9955	0.0640	0.0605
b21 = x^2y^1	13.1405	13.7040	13.7030	13.6936	4.0392	0.0761	0.0682
b12 = x^1y^2	20.5837	21.9315	21.9295	21.9271	6.1268	0.0199	0.0106
b03 = x^0y^3	48.4319	52.6733	52.6618	52.7321	8.1548	0.1116	0.1333
b40 = x^4y^0	20.1172	20.5350	20.5339	20.5235	1.9799	0.0560	0.0503
b31 = x^3y^1	19.7380	20.5698	20.5689	20.5578	3.9877	0.0582	0.0538
b22 = x^2y^2	27.5020	29.2759	29.2746	29.2733	6.0509	**0.0089**	**0.0042**
b13 = x^1y^3	48.4319	52.6771	52.6786	52.7321	8.1548	0.1043	0.1015
b04 = x^0y^4	121.5551	134.9193	134.9275	135.3215	10.1731	0.2972	0.2912
b50 = x^5y^0	33.5444	34.2398	34.2407	34.2202	**1.9750**	0.0573	0.0599
b41 = x^4y^1	31.6175	32.9364	32.9373	32.9203	3.9572	0.0490	0.0518
b32 = x^3y^2	41.3386	43.9604	43.9614	43.9657	5.9753	0.0120	0.0098
b23 = x^2y^3	64.7551	70.3402	70.3403	70.4284	8.0554	0.1252	0.1251
b14 = x^1y^4	121.5551	134.9168	134.9165	135.3215	10.1731	0.2991	0.2993
b05 = x^0y^5	317.7952	359.8569	359.8555	361.8728	**12.1804**	**0.5571**	**0.5574**