Spectral mesh-free quadrature for planar regions bounded by rational parametric curves

David Gunderman[a,b,*], Kenneth Weiss[b], John A. Evans[c]

[a]*Department of Applied Mathematics, University of Colorado Boulder, Boulder, CO 80309, USA*
[b]*Lawrence Livermore National Laboratory, 7000 East Avenue, Livermore, CA 94550, USA*
[c]*Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado Boulder, Boulder, CO 80309, USA*

Abstract

This work presents spectral, mesh-free, Green's theorem-based numerical quadrature schemes for integrating functions over planar regions bounded by rational parametric curves. Our algorithm proceeds in two steps: (1) We first find intermediate quadrature rules for line integrals along the region's boundary curves corresponding to Green's theorem. (2) We then use a high-order quadrature rule to compute the numerical antiderivative of the integrand along a coordinate axis, which is used to evaluate the Green's theorem line integral. We present two methods to compute the intermediate quadrature rule. The first is spectrally accurate (it converges faster than any algebraic order with respect to number of quadrature points) and is relatively easy to implement, but has no guarantee of polynomial exactness. The second guarantees exactness for polynomial integrands up to a pre-specified degree k with an *a priori*-known number of quadrature points and retains the convergence properties of the first, but is slightly more complicated. The quadrature schemes have applications to computation of geometric moments, immersogeometric analysis, conservative field transfer between high-order meshes, and initialization of simulations with rational geometry. We compare the quadrature schemes produced using our method to other methods in the literature and show that they are much more efficient both in terms of number of quadrature points and computational time.

Keywords: Quadrature, Rational, NURBS, Spectral Convergence

1. Introduction

Quadrature over arbitrary regions bounded by rational parametric curves is important in a variety of applications including computer-aided design (CAD) and computer-aided manufacturing (CAM), in which geometric objects are typically represented in terms of their boundaries (BREPs) in a basis of non-uniform rational B-splines (NURBS). Efficient and accurate computation of integrated quantities, such as geometric moments and mass, are often key ingredients in the design process.

While CAD has relied on NURBS as a basis for decades, rational parametrization of problem geometry has also recently become more popular in computer-aided analysis in an effort to more closely couple the design/analysis pipeline. In isogeometric analysis [1, 2, 3], NURBS parameterizations are used to represent the interior of the objects. However, as only boundaries are typically represented in CAD, this requires the computation of a high-quality interior parametrization in the form of a mesh. Thus, recently, there has been a push to combine isogeometric analysis with immersed boundary treatment using immersogeometric analysis [4, 5]. In immersogeometric analysis, the basis functions are defined with respect to a background grid which is immersed within the geometry of interest. This avoids the need to construct a high-quality interior parameterization, but requires integration over the intersections of the background grid with the geometry, as shown in Figure 1.

There already exist relatively efficient integration strategies for traditional high-order parametric elements (e.g. triangles/te-
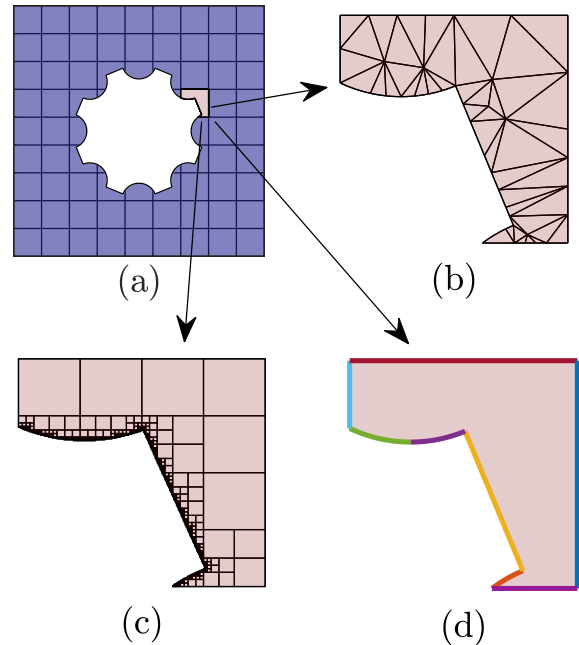

Figure 1: (a) In immersogeometric analysis, background cells intersecting the immersed boundary require special quadrature rules. (b) Rational meshing takes the geometry into account, but requires expensive pre-processing. (c) Adaptive quadtree integration converges at most linearly. (d) Our boundary-based method takes the geometry into account without expensive meshing.

*Corresponding author
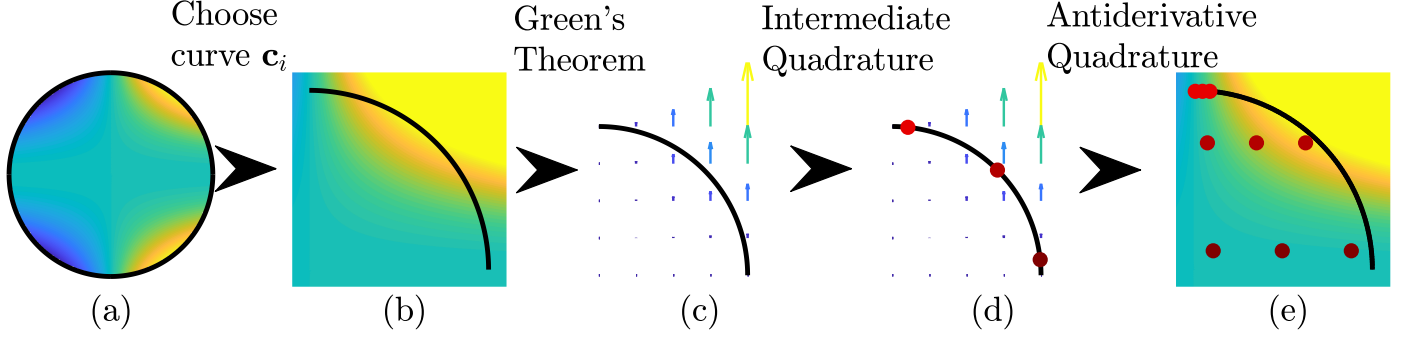Email address:* david.gunderman@colorado.edu (David Gunderman)

Figure 2: Overview of our quadrature algorithms for integration over a region bounded by rational curves. The integrand function $f(x, y)$ is given as a color plot in (a), (b), and (e), while its antiderivative $A_f(x, y)$ is given as a colored vector plot in (c) and (d). The intermediate quadrature points are notated by the shaded red dots in (d). The final quadrature points, produced by evaluating the antiderivative function using antiderivative quadrature at the red dots in (d), are shown as correspondingly shaded dots in (e).

trahedra or quads/hexahedra) which occur in isogeometric analysis using parametric mappings of high-order simplicial and tensor product-type quadrature rules. However, these quadrature rules cannot be seamlessly or efficiently applied to arbitrary rational regions due to the inherent difficulties in high-order meshing of arbitrary geometries. In contexts such as immersogeometric analysis [6], as well as in cutFEM [7], field transfer between high-order meshes [8], initialization of volume of fluid fields for rational shapes [9, 10], and moment-fitting [11], efficient integration over arbitrary rational regions (such as intersections of conic sections) is fundamental.

The need for efficient quadrature schemes for arbitrary planar rational regions could potentially be met by the rich body of literature on efficient integration over arbitrary planar regions. However, when considering rational geometry, these methods either require expensive preprocessing in the form of meshing or only achieve low-order convergence due to their low-order approximation of the geometry. Our algorithm is able to achieve spectral convergence (i.e. faster than any algebraic order with respect to the number of quadrature points) without the need for expensive meshing.

Existing methods for integrating over arbitrary 2D geometries fall into two general categories: (1) domain decomposition methods and (2) Green's theorem methods. Domain decomposition methods generally rely on two steps: tessellation using a set of basic elements (such as a triangulation or a decomposition into nested rectangles) and integration over these basic elements. Another recently introduced method is the class of correction-based methods, in which geometry is approximated by linear boundaries, then a geometric correction term is applied [12, 13].

Low-order domain decomposition methods tessellate the region using linear polygons (quadtrees, triangulation, and quadrangulation being the most common cases). The tessellation and integration steps are both very robust in the case of linear polygonal elements and have been studied extensively. The robustness is paid for in low-order convergence [14, 15, 16, 17], which cannot be overcome by increasing the quadrature order, since the underlying geometric approximation is low-order.

Higher-order or exact domain decomposition methods are generally less robust and require more preprocessing time, because they attempt to tessellate the geometry using high-order or rational parametric elements to achieve higher-order convergence. The tessellation problem over planar regions becomes a high-order 2D meshing problem, which has been studied extensively [18, 19, 20, 21, 22]. Once the parametric elements have been defined, integrals over the elements are computed using well-known Gaussian quadrature rules for high-order parametric mappings of triangles or rectangles. High-order (and even spectral, in the case of exact meshing) convergence can be attained, because both the quadrature scheme and the geometric approximation can be made high-order.

The other common method for quadrature over complicated geometries avoids meshing the interior by using Green's theorem to transform double integrals over the domain into a sum of line integrals over its boundaries. These methods have been employed extensively to increase computational efficiency of integration in a variety of situations, including for linear polygonal regions, regions defined by polynomial parametric boundaries, and implicitly-defined regions [23, 24, 25].

A key limitation of Green's theorem-based methods is that they require some parametric description of the boundary. However, an approximation of the boundary using splines or other high-order parametric curves can be used to guarantee high-order convergence [26, 27, 28], while still avoiding the potential difficulties of meshing. If boundaries are given parametrically (as is the case for NURBS objects), this extra information is available without any extra approximation, and should be fully exploited to attain the most efficient quadrature rules possible – a fact which we take advantage of in our scheme. Another popular belief about Green's theorem methods is that they can only be used on integrands for which analytic antiderivatives are available, and these methods have been used in this case for decades [29, 30]. However, recently it has become more common to use Green's theorem-based methods for arbitrary integrands [25, 27, 31], since numerical antiderivatives can be efficiently computed to high precision using a high-order quadrature rule, such as Gaussian quadrature – another fact which we take advantage of in our scheme.

In this paper, we consider the problem of integrating arbitrary functions over planar regions bounded by rational curves, with special emphasis on those that may be difficult to efficiently mesh with simple parametric elements. While specialized integration strategies based on Green's theorem have been previously published in the literature for regions bounded by linear segments, polynomial curves, and implicit curves, to the best of our knowledge, no such optimized integration method has been proposed for regions bounded by rational curves.

More formally, we develop two numerical algorithms for computing quadrature rules for integrals of the form

$$\iint_\Omega f(x,y)dA,$$

over an arbitrary planar region Ω bounded by a set of n_r rational and n_p polynomial curves of degrees $\{m_i\}_{i=1}^{n_r}$ for the rational curves and degrees $\{m_i\}_{i=n_r+1}^{n_c}$ for the polynomial curves (where $n_c = n_r + n_p$),

$$\partial\Omega = \cup_{i=1}^{n_c}\mathbf{c}_i :$$

$$\mathbf{c}_i(s) = \begin{cases} x_i(s) \\ y_i(s) \end{cases} \quad 1 \le i \le n_c,$$

for $0 \le s \le 1$ and the curves $\mathbf{c}_i(s)$ are polynomial for $i > n_r$.

The first algorithm produces quadrature rules which, for smooth integrands, converge to the correct integral exponentially quickly with respect to the number of quadrature points. Exponential convergence is a particular class of spectral convergence (in which error decreases faster than any algebraic order), so we refer to this algorithm as SPECTRAL quadrature. The second algorithm produces quadrature rules which retain the fast convergence properties of the first, but are also exact (up to machine precision) for polynomial integrands up to a pre-specified degree k, so we refer to it as SPECTRAL PE (spectral, polynomially exact) quadrature.

As outlined in Figure 2, our algorithms both consist of the same two primary steps, which are applied to each component curve $\mathbf{c}_i$ individually: (1) we find appropriate intermediate quadrature rules for line integrals corresponding to Green's theorem, taking into account the degrees of the corresponding boundary curves and (2) we use a high-order quadrature rule to compute the numerical antiderivative of the integrand along a coordinate axis, which is used to evaluate the Green's theorem line integral. In the SPECTRAL method, we use Gaussian quadrature for both quadrature rules, the order of which can be increased to converge faster than any algebraic order with respect to the total number of quadrature points. In the SPECTRAL PE method, we use a rational quadrature rule instead of Gaussian quadrature for the intermediate quadrature rule. These methods are described in detail in Section 2.

In addition to their spectral convergence, our algorithms are also computationally efficient. The cost for producing the SPECTRAL quadrature rules scales linearly with the number of points in each of the underlying Gaussian quadrature rules. The cost of producing a SPECTRAL PE quadrature rule that is exact for polynomials up to degree k scales as $\mathcal{O}(n_c \max_{i \le n_r}(m_i)^3 k^2)$, where the component curve degrees m_i are constant in most applications. We derive the number of quadrature points and the computational costs in Section 2.

For smooth integrands over arbitrary rational regions, we show on numerical examples in Section 3 that both the SPECTRAL and SPECTRAL PE methods are much more efficient than existing methods both in terms of number of quadrature points and computational time. We compare our two methods to four methods used in the literature: quadtree integration [6], linear meshing with high-order Gaussian integration [32], rational meshing with high-order Gaussian integration [20], and cubic spline approximation of the boundary combined with a Green's theorem approach [27].

*Contributions.* The main contributions of our work include:
- We present algorithms which compute the locations and weights of SPECTRAL quadrature rules for integration of arbitrary smooth integrands and SPECTRAL PE (spectral, polynomially exact) quadrature rules for exact (up to machine precision) integration of polynomials up to a given degree k over a region bounded by rational curves. Both algorithms converge faster than any algebraic order, as we demonstrate on a variety of test shapes and integrands.
- We derive a formula for the number of quadrature points needed in the SPECTRAL PE quadrature rule for exact (up to machine precision) integration of polynomials up to degree k over an arbitrary region bounded by rational curves. This is analogous to Gaussian quadrature for polynomial regions, for which an exact number of quadrature points can be calculated to integrate polynomials up to a pre-specified degree in 1D.
- A claimed difficulty of our type of approach in the literature is that it requires antiderivatives. We show that these are available numerically and easy to calculate exactly for a broad class of useful integrands, including any polynomial. Moreover, we can compute highly accurate numerical antiderivatives of an even broader class of integrands, as can be seen in our experiments in Figure 9.
- We compare our proposed quadrature schemes to four other commonly-used quadrature schemes in the literature, which represent the state of the art for integrating over rational regions, and show that the proposed algorithms produce quadrature rules which are significantly more efficient both in terms of the number of quadrature points needed and the computation time needed to achieve a given integration accuracy.

Outline. The remainder of the paper is organized as follows: In Section 2, we introduce the new algorithms for the SPECTRAL and SPECTRAL PE quadrature rules for planar regions bounded by rational curves. We present numerical results on a variety regions and for a variety of integrands with comparison methods in Section 3 and conclude in Section 4 with a discussion of our results and ideas for future work.

2. Numerical Integration over Rational Regions

We first consider a genus-zero planar region Ω with a closed, connected, oriented boundary loop Γ such that the normal to Γ points toward the interior of the region Ω. We take the boundary loop Γ to be composed of n_r rational and n_p polynomial oriented component curves $\{\mathbf{c}_i\}_{i=1}^{n_c}$ (where $n_c = n_r + n_p$), where the $\mathbf{c}_i$ are given in the rational Bernstein-Bézier basis:

$$\mathbf{c}_i(s) = \begin{cases} x_i(s) = \dfrac{\sum_{j=0}^{m_i} w_{i,j} x_{i,j} B_j^{m_i}(s)}{\sum_{j=0}^{m_i} w_{i,j} B_j^{m_i}(s)} \\[4ex] y_i(s) = \dfrac{\sum_{j=0}^{m_i} w_{i,j} y_{i,j} B_j^{m_i}(s)}{\sum_{j=0}^{m_i} w_{i,j} B_j^{m_i}(s)} \end{cases} \quad 1 \le i \le n_c,$$
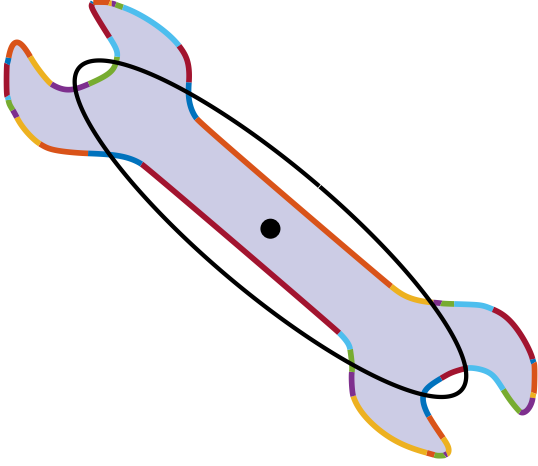
$$n_c = 46, m = 3$$

Figure 3: A wrench-shaped region bounded by rational curves over which polynomials can be integrated to machine precision using the SPECTRAL PE algorithm. This example illustrates the object's first few geometric moments, as computed by the SPECTRAL PE algorithm: The grey area represents its area (0th moment), the black dot represents its centroid (1st moment), and the ellipse represents its moment of inertia (2nd moment).

for $0 \leq s \leq 1$ such that $\Gamma = \cup_{i=1}^{n_c} \mathbf{c}_i$, where m_i are the degrees of the component curves, $w_{i,j}$ are the control weights, and $x_{i,j}, y_{i,j}$ are the control points. The Bernstein-Bézier basis is defined as

$$B_j^m(s) = \binom{m}{j}(1-s)^j s^{m-j}, \qquad (1)$$

and has a multitude of useful properties [33, 34]. For rational curves, $i \leq n_r$, the control weight matrix $w_{i,j}$ has at least one entry such that $w_{i,j} \neq 1$ for some i, j. For polynomial curves $i > n_r$, every entry in the control weight matrix is unity, $w_{i,j} = 1$ for all i, j. We refer to the collection of rational boundary curves as Γ^r and the collection of polynomial boundary curves as Γ^p. We denote the maximum degree of the rational component curves as $m = \max_{i \leq n_r} m_i$. We denote the bounding box of the control points of the region Ω as

$$\tilde{\Omega} = [\min_{i,j}\{x_{i,j}\}, \max_{i,j}\{x_{i,j}\}] \times [\min_{i,j}\{y_{i,j}\}, \max_{i,j}\{y_{i,j}\}].$$

Both the SPECTRAL and SPECTRAL PE algorithms compute quadrature points $\{(x_l, y_l)\}_{l=1}^{n_q}$ and weights $\{w_l\}_{l=1}^{n_q}$ such that the error

$$E = \left| \iint_\Omega f(x,y)dxdy - \sum_{l=1}^{n_q} w_l f(x_l, y_l) \right| < \mathcal{O}(e^{-cn_q}),$$

decreases exponentially with respect to number of quadrature points for some constant c for smooth integrands in the bounding box of Ω, $f(x,y) \in \mathscr{C}^\infty(\tilde{\Omega})$, where n_q is the number of quadrature points. This is because they are based on one-dimensional high-order quadrature rules, which can be shown to have exponential convergence (c.f. [35]). If the function has only k continuous derivatives, $f(x,y) \in \mathscr{C}^k(\tilde{\Omega})$, then the scheme attains the maximum algebraic order of convergence possible, $E = \mathcal{O}(n_q^{-k+1})$.

Moreover, because it uses rational quadrature for the intermediate quadrature rule, the SPECTRAL PE quadrature rule is exact up to machine precision, ε, for polynomials $p(x,y)$ of maximal degree k or less with an *a priori* known number of quadrature points n_q, which, as we derive in Section 2.4, is:

$$n_q = (k\lceil k/2 \rceil + 3k) \sum_{i \leq n_r} m_i + \lceil k/2 \rceil^2 \sum_{i > n_r} m_i. \qquad (2)$$

Figure 3 shows an example of a wrench-shaped region composed of $n_c = 46$ component curves, all of which are rational and of degree $m_i = 3$ (i.e. $n_r = 46$, $\Gamma^r = \Gamma$, $n_p = 0$, and $\Gamma^p = \emptyset$). The SPECTRAL PE quadrature scheme requires $n_q = 1104$ quadrature points to exactly integrate polynomials up to degree $k = 2$ and $n_q = 2070$ quadrature points to exactly integrate polynomials up to degree $k = 3$. This works out to 24 and 45 quadrature points per component curve, respectively, for $k = 2$ and $k = 3$. Note that the SPECTRAL PE quadrature scheme still retains the spectral convergence property for arbitrary smooth integrands, so fewer quadrature points could be used if machine precision is not necessary.

The general idea behind the present algorithms is to use Green's theorem to transform the integral over the interior region Ω into a sum of line integrals along each curve $\mathbf{c}_i$,

$$\iint_\Omega f(x,y)dxdy = \int_{\partial\Omega} A_f(x,y)dx$$
$$= \sum_{i=1}^{n_c} \int_{\mathbf{c}_i} A_f(x,y)dx, \qquad (3)$$

where $A_f(x,y)$ is the y-antiderivative of the integrand.

The algorithms are overviewed graphically in Figure 2 and, for each component curve $\mathbf{c}_i$, can both be summarized in two basic steps:

(1) Compute a set of intermediate quadrature points and weights for the Green's Theorem line integral corresponding to the particular curve $\mathbf{c}_i$, and

(2) Compute quadrature rules to evaluate the antiderivative function numerically at each intermediate quadrature point.

The two quadrature rules' weights are then combined in a tensor-product fashion to produce the quadrature weights for each of the final quadrature points. The SPECTRAL and SPECTRAL PE algorithms only differ in the implementation for the first step in the above two-step process. We describe the SPECTRAL algorithm initially, then describe how the SPECTRAL PE algorithm differs.

2.1. Intermediate Quadrature Rule

The first step in the algorithm is to convert the area integral inside the domain Ω into line integrals along each of the curves, $\{\mathbf{c}_i\}_{i=1}^{n_c}$, and to find appropriate intermediate quadrature rules for each of these line integrals. For the purposes of this section, we assume that the antiderivative function

$$A_f(x,y) = \int_C^y f(x,t)dt, \qquad (4)$$

can be evaluated to high precision at any (x,y) point on any of the component curves $\mathbf{c}_i$. We explain our approach for this in the next section (Section 2.2).

4

Because the curves are given parametrically, the physical space integrals appearing on the right-hand side of Equation (3) can be transformed into the parametric space of each component curve:

$$\int_{\mathbf{c}_i} A_f(x,y)dx = \int_0^1 A_f(x_i(s), y_i(s)) \frac{dx_i(s)}{ds} ds. \quad (5)$$

Each of these integrals can be evaluated using a high-order 1D quadrature scheme, such as Gaussian quadrature with Q_i-points, which yields a set of Q_i quadrature points $\{s_{i,q}\}_{q=1}^{Q_i}$ and weights $\{\gamma_{i,q}\}_{q=1}^{Q_i}$. The intermediate quadrature points in $\mathbb{R}^2$ can then be calculated as $(x_{i,q}, y_{i,q}) = (x_i(s_{i,q}), y_i(s_{i,q}))$ for each $q = 1, \dots Q_i$. This evaluation can be performed stably for rational Bernstein-Bézier curves using de Casteljau's algorithm [33].

The intermediate quadrature rule then reads as

$$\iint_\Omega f(x,y)dxdy \approx \sum_{i=1}^{n_c} \sum_{q=1}^{Q_i} \gamma_{i,q} A_f(x_{i,q}, y_{i,q}) \frac{dx_i}{ds}(s_{i,q}). \quad (6)$$

The resulting integrand in each of the line integrals is composed of two parts, one which requires evaluation of the integrand's antiderivative and one which requires evaluation of the derivative of a Bézier curve. Derivatives of rational or polynomial Bézier curves can be efficiently and stably computed to machine precision using de Casteljau's algorithm. As long as the antiderivative is evaluated to high enough precision, the intermediate quadrature rule will converge exponentially fast as Q_i increases [35].

Next, we describe how to evaluate the antiderivative.

2.2. Antiderivative Quadrature Rule

A key ingredient in the evaluation of the interior function of the intermediate quadrature rule in Equation (6) is the evaluation of the antiderivative. A high-order quadrature rule, such as Gaussian quadrature with P points, can be used to evaluate the integrand with spectral convergence as P is increased [35]. In our implementation, for simplicity, we set $P = Q_i$.

The antiderivative quadrature rule will then have quadrature points $\{x_{i,q}, y_{i,q,\zeta}\}_{\zeta=1}^P$, and weights $\{\gamma_{i,q,\zeta}\}_{\zeta=1}^P$. For a particular curve $\mathbf{c}_i$ and intermediate quadrature point $(x_{i,q}, y_{i,q})$, the antiderivative can be evaluated as

$$A_f(x_{i,q}, y_{i,q}) = \int_C^{y_{i,q}} f(x_{i,q}, y)dy \approx \sum_{\zeta=1}^P \gamma_{i,q,\zeta} \, f(x_{i,q}, y_{i,q,\zeta}). \quad (7)$$

Note that the $x_{i,q}$ do not depend on the index ζ, since the antiderivative is only with respect to y. If the function $f(x,y)$ is a polynomial of degree k, then Equation (7) will be exact (up to machine precision) with $P = \lceil k/2 \rceil$ points. If the function is smooth, then convergence to the correct integral will be spectral with respect to the number of quadrature points as P is increased.

Although the arbitrary integration constant C in the calculation of $A_f(x,y)$ has no bearing on the validity of equation (3), it can affect the numerical stability as well as the locations and weights of the resulting quadrature scheme. We choose C according to the strategy described in Section 2.5.2.

2.3. Full Quadrature Rule

Finally, the intermediate quadrature rule and the numerical antiderivative quadrature rule can be combined to obtain a spectral full quadrature scheme:

$$\iint_\Omega f(x,y)dxdy \approx \sum_{i=1}^{n_c} \sum_{q=1}^{Q_i} \sum_{\zeta=1}^P w_{i,q,\zeta} f(x_{i,q}, y_{i,q,\zeta}), \quad (8)$$

where $w_{i,q,\zeta} = \gamma_{i,q}\gamma_{i,q,\zeta}\frac{dx_i}{ds}(s_{i,q})$. The derivative terms $\frac{dx_i}{ds}$ can be computed efficiently using de Casteljau's algorithm and can be viewed as geometric correction terms to the component Gaussian quadrature weights. The total number of quadrature points used in the final quadrature rule will scale with the number of quadrature points used in each component quadrature rule, $n_q = P\sum_{i=1}^{n_c} Q_i$. Since we use $P = Q_i$, the number of quadrature points will be $n_q = n_c P^2$.

2.4. A Modified Algorithm for Polynomial Exactness

In some situations, such as calculation of geometric moments and moment-fitting techniques, it is of particular importance to efficiently calculate integrals of polynomial functions over arbitrary rational regions. Calculating these integrals exactly with an *a priori* number of quadrature points can remove the need for potentially expensive adaptive quadrature that attempts to converge to the solution through succesively increased number of quadrature points. With a modification to the intermediate quadrature step of the SPECTRAL algorithm presented above, we can achieve exactness up to machine precision for polynomial integrands over these regions. We call this quadrature scheme SPECTRAL PE(spectral, polynomially exact). Exactness can be achieved by guaranteeing that the two components steps of intermediate quadrature and antiderivative quadrature are exact.

The antiderivative quadrature rule is already exact for polynomial integrands $f(x,y)$ up to a pre-specified degree k if Gaussian quadrature is used with $P = \lceil k/2 \rceil$ points. The intermediate quadrature rule, on the other hand, has no guarantees of exactness for polynomial integrands if Gaussian quadrature is used, since both the antiderivative function $A_f(x(s), y(s))$ and the parametric derivative term $\frac{dx}{ds}$ are rational, rather than polynomial, functions. However, a rational quadrature rule exact for an appropriate class of rational functions can be used for the intermediate quadrature rule in order to achieve polynomial exactness (up to machine precision).

2.4.1. Review of Rational Quadrature

Exact quadrature formulas for particular classes of functions such as polynomials have been studied for centuries. Quadrature rules which are exact for polynomials are generally formulated in terms of classes of orthogonal polynomials [36]. Two of the most commonly used quadrature rules are Gaussian quadrature and Fejer (i.e. Clenshaw-Curtis) quadrature, which are based on the orthogonal Legendre and Chebyshev polynomials, respectively. Exact quadrature formulas for classes of nonpolynomial functions have also been studied in the literature, and are typically based on generalizations of the classical orthogonal polynomials. The first study of orthogonal rational functions was made by Djirbashian in the 1960's, a review of which can be found in [37]. A connection to exact Gauss-type quadrature

rules for rational functions with known real poles was first made by Gautschi [38].

More recently, Bultheel, Deckers, and Van Deun have published a series of papers with very efficient algorithms for computing Fejér-type quadrature rules exact for rational functions with known poles outside the unit interval in linear time with respect to the total number of poles of the rational functions for which exactness is guaranteed [39, 40, 41, 42]. The algorithm only requires knowing the locations and multiplicities of the poles of the rational functions of interest.

Next, we describe an algorithm which computes locations and multiplicities of the poles of the rational functions which must be integrated exactly in order to achieve polynomial exactness.

2.4.2. Poles of Rational Functions

For a particular curve c_i, the rational functions which must be integrated exactly are those that arise as the product of the antiderivative function $A_f(x_i(s))$ and the derivative term $\frac{dx(s)}{ds}$, as seen in Equation (5). As we shall prove, the poles of the product function $A_f(x_i(s))\frac{dx(s)}{ds}$ are simply the poles of the original curve c_i but with the multiplicity of each pole multiplied by $k+3$. To prove this, consider the two terms in the product separately. For clarity, we call $\{p_{i,j}\}_{j=1}^{m_i}$ the poles of the curve c_i, defined as the zeros of the weight function

$$w_i(s) = \sum_{j=0}^{m_i} w_{i,j} B_j^{m_i}(s).$$

By the quotient rule, the derivative of a rational function has the effect of doubling the multiplicity of each of its poles. Therefore, $\frac{dx_i(s)}{ds}$ will have the same poles $\{p_{i,j}\}_{j=1}^{m_i}$ as $x_i(s)$, but each will have multiplicity of 2.

Since $x_i(s)$ and $y_i(s)$ are each rational functions with poles $\{p_{i,j}\}_{j=1}^{m_i}$ and $A_f(x,y)$ is a polynomial, $A_f(x(s),y(s))$ will also have the poles $\{p_{i,j}\}_{j=1}^{m_i}$. However, each pole's multiplicity will be $k+1$ times as large as the multiplicity it had for $x_i(s)$ or $y_i(s)$. Assuming that $f(x,y)$ is a degree k polynomial and, expressing it in the monomial basis as $f(x,y) = \sum_{k_x+k_y\le k} a_{k_x,k_y} x^{k_x} y^{k_y}$ for some coefficients a_{k_x,k_y}, then

$$A_f(x_i(s),y_i(s)) = \sum_{k_x+k_y\le k} a'_{k_x,k_y} x_i(s)^{k_x} y_i(s)^{k_y+1}$$

$$= \sum_{k_x+k_y\le k} \left(a'_{k_x,k_y} \left(\frac{\sum_{j=0}^{m_i} w_{i,j} x_{i,j} B_j^{m_i}(s)}{\sum_{j=0}^{m_i} w_{i,j} B_j^{m_i}(s)} \right)^{k_x} \left(\frac{\sum_{j=0}^{m_i} w_{i,j} y_{i,j} s^j}{\sum_{j=0}^{m_i} w_{i,j} B_j^{m_i}(s)} \right)^{k_y+1} \right)$$

$$= \sum_{k_x+k_y\le k} \left(d'_{k_x,k_y} \frac{\left(\sum_{j=0}^{m_i} w_{i,j} x_{i,j} B_j^{m_i}(s)\right)^{k_x} \left(\sum_{j=0}^{m_i} w_{i,j} y_{i,j} B_j^{m_i}(s)\right)^{k_y+1}}{\left(\sum_{j=0}^{m_i} w_{i,j} B_j^{m_i}(s)\right)^{k_x+k_y+1}} \right),$$

where $d'_{k_x,k_y} = a_{k_x,k_y}/(k_y+1)$ and k_x+k_y attains its maximum value of k, since $f(x,y)$ is a degree k polynomial.

Combining the above statements, the integrand $A_f(x(s),y(s))\frac{dx(s)}{ds}$ in equation (3) will have the same poles as the component curves did, namely $\{p_{i,j}\}_{j=1}^{m_i}$. However, each of these poles will have a multiplicity of $k+3$, where $k+1$ of

them come from the antiderivative calculation and 2 of them come from the derivative term.

Therefore, in order to find a rational intermediate quadrature rule exact for rational functions with the required poles, we must compute the set of poles $\{p_{i,j}\}_{j=1}^{m_i}$ of each component curve $\{c_i\}_{i=1}^{n_c}$. Because the rational curves are given in terms of control points and control weights, the poles of the rational curves $\{c_i\}_{i=1}^{n_c}$ can be easily calculated as the roots of the corresponding control weight (i.e. denominator) polynomial, $w_i(s) = \sum_{j=0}^{m_i} w_{i,j} B_j^{m_i}(s)$.

There is a wide body of literature on stable univariate polynomial complex root finding. The roots of a polynomial in the monomial basis can be found by computing the eigenvalues of the polynomial's companion matrix, A_i [43]. By converting the polynomial $w_i(s)$ to the monomial basis $w_i(s) = \sum_{j=0}^{m_i} a_{i,j} s^j$ (see Section 2.5.1), the companion matrix can be written as

$$A_i = \begin{bmatrix} 1 & 0 & 0 & \dots & -\frac{a_{i,1}}{a_{i,m_i}} \\ 1 & 1 & 0 & \dots & -\frac{a_{i,2}}{a_{i,m_i}} \\ 0 & 1 & 1 & \dots & -\frac{a_{i,3}}{a_{i,m_i}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & -1. \end{bmatrix}$$

The eigenvalues of A_i are the roots of $w_i(s)$. Computing the eigenvalues of this $(m_i+1)\times(m_i+1)$ matrix is generally an $\mathcal{O}(m_i^3)$ operation. Finding the roots for all of the rational component curves therefore scales as $\mathcal{O}\left(n_c \max_{i\le n_r} m_i^3\right)$ with the degree of the component rational curves, since finding the roots of the control weight polynomial $w_i(s) = \sum_{j=0}^{m_i} w_{i,j} B_j^{m_i}(s)$ requires $\mathcal{O}(m_i^3)$ operations. However, for relatively low degree rational curves, the cost of this root-finding step is minimal and the degree of the component rational curves is often taken to be constant in applications.

2.4.3. Rational Intermediate Quadrature

Once the rational poles, $\{p_{i,j}\}_{i=1}^{m_i}$ for each c_i and their $k+3$ multiplicities are known, the remaining step is to calculate an exact quadrature rule for functions with these poles in the parametric space $0 \le s \le 1$ of each $\{c_i\}_{i=1}^{n_c}$. For the polynomial portions of the boundary, a simple Gaussian quadrature rule will be sufficient to achieve an exact intermediate quadrature rule. For the rational portions of the boundary, we must use an exact rational quadrature rule. We choose to use a linear-time algorithm developed in [42]. Implementation details are given in Section 2.5.3. The quadrature points and weights from this algorithm can then be substituted for the Gaussian quadrature points and weights in Section 2.1.

The resulting exact quadrature rule will have n_q quadrature points:

$$n_q = (k\lceil k/2\rceil + 3k) \sum_{i=1}^{n_r} m_i + \lceil k/2\rceil^2 \sum_{i=n_r+1}^{n_c} m_i, \qquad (9)$$

where n_c is the number of component curves, m_i are the degrees of the component curves c_i, and k is the maximal degree of the polynomial integrand. Pseudocode for this algorithm is provided in Figure 4.

```
1   f = @(x,y) getIntegrand();
2   kDeg=getDegree(f);
3   Bd=loadShape();
4   for i=1:getNumCurves(Bd)
5     % Compute intermediate quadrature rule
6     if use_SpectralPE
7       poles=roots(getWeights(Bd(i)))    %
            2.5.1
8       intPoles=repmat(poles,kDeg+3);    %
            2.4.2
9       [intpts intwts]=
10           calcIntQRule(intPoles);      %
               2.5.3
11    else
12      [intpts intwts]= calcIntQRule();  %
            2.1
13    end
14
15    % Compute antiderivative quadrature
         rule
16    for j=1:length(intpts)
17      [Afx,Afy]=Bd{i}(intpts);
18      [ADpts, ADwts]=                   %
            2.2
19          calcAntiderivativeQRule([0,Afx])
               ;
20      finalptsx(j,:)=ADpts;
21      len = length(finalptsx)
22      finalptsy(j,:)=repmat(Afy,len,1);
23      finalwts(j,:)=intwts(j)*ADwts;
24    end
25    quadrules{i}=[finalptsx,
26                  finalptsy,
27                  finalwts ];            %
                    2.3
28  end
```

Figure 4: Matlab pseudocode for the quadrature algorithms given in Section 2.
The function calcIntQRule is the only difference between the SPECTRAL
algorithm and the SPECTRALPE algorithm. The first uses a typical high-order
quadrature rule, such as Gaussian quadrature. The second uses a quadrature
rule exact for rational functions with poles given by intPoles. Line-by-line
comments refer to sections of this article.

2.5. Implementation Notes

In this section, we describe the implementation details we
used to produce the numerical results in Section 3.

2.5.1. Poles of Rational Curves–Implementation

In the test cases that follow, we consider $\mathbf{c}_i$ to be given in the
Bernstein-Bézier basis. In order to find the roots of the control
weight polynomial,

$$w_i(s) = \sum_{j=0}^{m_i} w_{i,j} B_j^{m_i}(s),$$

we use Matlab's roots command [44], which has an optimized
algorithm for finding the eigenvalues of the companion matrix
of $w_i(s)$. However, to use Matlab's roots command, we must
convert from the Bernstein-Bézier basis to the monomial ba-
sis, which is an ill-conditioned basis conversion for high-degree
polynomials, as discussed in [45]. The condition number of
the conversion matrix is generally on the order of $10^{\frac{m}{2}}$, where
m is the degree of the polynomial. Approximately $\frac{m}{2}$ digits of
precision are therefore lost in this conversion. The conversion
matrix from the monomial basis to the Bernstein-Bézier basis is
defined as

$$MB_{i,j} = \begin{cases} \frac{\binom{n-k+1}{j-k}}{\binom{n}{j-1}} & \text{for } j \ge k \\ 0 & \text{otherwise}, \end{cases}$$

and the conversion from the Bernstein-Bézier basis to the mono-
mial basis is defined as

$$BM_{i,j} = \begin{cases} (-1)^{j-k} \binom{n}{j-1} \binom{j-1}{k-1} & \text{for } j \ge k \\ 0 & \text{otherwise}, \end{cases}$$

where $BM_{i,j}$ denotes the entry in the i-th row and j-th column of
the matrix. An optimal implementation would instead find roots
in the original input basis to avoid basis conversion stability
issues.

2.5.2. Green's Theorem–Implementation

The choice of C in the antiderivative computation in Equa-
tion (4) can affect the stability of the resulting calculation, since
the antiderivatives can become arbitrarily large depending on C.
In order to best circumvent this stability issue, we choose C to
be the minimum y-component of any control point making up
Γ. That is, we choose

$$C = \min_{i,j} y_{i,j}.$$

The Gaussian quadrature points used to evaluate the integral
(4) will therefore always lie between the intermediate quadra-
ture point y and the smallest control point of the boundary loop
Γ. Since the points y always lie on the component curves, the
quadrature points from the full algorithm will always lie within
the bounding box of the control points.

2.5.3. Rational Intermediate Quadrature–Implementation

To compute the intermediate quadrature rule for the modi-
fication to achieve polynomial exactness in Section 2.4.3, we
use a method developed in [42]. The method generates $m+1$
quadrature points when m (possibly non-unique) poles are spec-
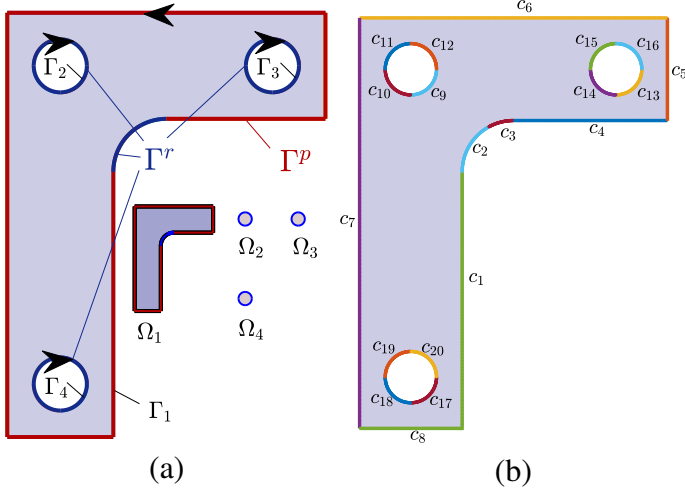ified in linear time with respect to the number of poles m. The

Figure 5: An example of a non genus-zero region with $M = 4$ boundary loops and $n_c = 20$ component curves. In this case, all of the polynomial part of the boundary, Γ^p, is linear ($m_i = 0$), and all of the rational part of the boundary, Γ^r, is quadratic ($m_i = 2$), so the maximum degree of the rational component curves is $m = 2$. In (a), the boundary curves $\{\Gamma_\alpha\}_{\alpha=1}^4$ and boundary regions $\{\Omega_\alpha\}_{\alpha=1}^4$ are labeled. In (b), the component curves $\{c_i\}_{i=1}^{n_c}$ are labeled.

algorithm also ensures a degree of polynomial exactness beyond the default rational exactness. If m poles are specified but $m + l$ quadrature points are requested, the algorithm will be exact for rational functions with the m poles as well as for (one-dimensional) polynomial functions up to degree l. Convergence of the SPECTRAL PE for arbitrary functions is achieved by increasing this l parameter past that required by the degree of polynomial exactness k, for example in Figure 9.

The algorithm can be used to compute exact quadrature rules on any real interval for any class of rational functions with arbitrary poles outside of the domain of integration. It is well known that if all of the coefficients of a polynomial in the Bernstein-Bézier basis are of the same sign, then the polynomial has no roots within the unit interval [33]. Therefore, as long as all of the control weights $w_{i,j}$ in all of the component curves $\mathbf{c}_i$ are positive (a common situation in most applications), the quadrature scheme given in [42] is theoretically valid.

2.6. Remark: Location of Quadrature Points

As can be seen in Figure 6, the quadrature points do not, in general, lie within the integration region Ω. However, the quadrature points in our scheme are always contained within the bounding box of the component curves, $\tilde{\Omega}$. It is sometimes desirable that the locations of the quadrature points be contained within the boundaries of the quadrature region Ω, particularly when dealing with potentially non-analytic integrands. A moment-fitting approach could potentially be used combined with our algorithm to compute a quadrature rule with only interior quadrature points [11]. We leave exploration of such a method to a future work.

2.7. Remark: Multiple Components

In some cases, the region of interest may not be genus-zero (i.e., it may have holes or disconnected components), see, for example, the region in Figure 5. In this case, there are multiple boundary loops $\partial\Omega = \{\Gamma_\alpha\}_{\alpha=1}^M$, each of which bounds a

genus-zero domain Ω_α. The original domain Ω is then a finite Boolean (union, intersection, and subtraction) combination of the component genus-zero domains Ω_α. As long as the orientations of each of the boundary curves Γ_α are given, the algorithm as stated transfers seamlessly, since it treats each of the component curves $\mathbf{c}_i$ independently. In this case, the integral over Ω can be thought of as the signed sum of the integrals over each Ω_α: $\iint_\Omega f(x,y)dxdy = \sum_{\alpha=1}^N s(\alpha) \iint_{\Omega_\alpha} f(x,y)dxdy$, where the indicator function $s(\alpha)$ can be sorted out according to the orientations of the boundary curves:

$$s(\alpha) = \begin{cases} 1 & \text{if } \Gamma_\alpha \text{ is counter-clockwise oriented} \\ -1 & \text{if } \Gamma_\alpha \text{ is clockwise oriented.} \end{cases}$$

Our aforementioned algorithm is then applied to each integral individually.

3. Numerical Test Cases

In this section, we test the SPECTRAL and SPECTRAL PE algorithms derived in Section 2 to compute integrals of various functions over six different domains. We first investigate the methods for monomial integrands on a circular region defined by four rational quadratic Bernstein-Bézier curves. Then, we investigate the methods on five other relatively more complicated regions: (1) a plate with a hole, commonly used as an isogeomtric test case [1], (2) an L-bracket with three holes, also used as an isogeomtric test case [46], (3) a guitar-shaped region, (4) a treble clef-shaped region, and (5) a region representing a cross section of two interlocked screws [47]. In our experiments, our quadrature schemes display the same properties on simpler shapes, such as the region in Figure 1, as on the six test shapes in this section.

We show numerically that our methods produce spectrally convergent quadrature schemes. In addition we show that the SPECTRAL PE method, with exact rational intermediate quadrature, is exact for polynomials up to degree k when

$$n_q = (k\lceil k/2 \rceil + 3k) \sum_{i=1}^{n_r} m_i + \lceil k/2 \rceil^2 \sum_{i=n_r+1}^{n_c} m_i \qquad (10)$$

quadrature points are used for regions enclosed by n_c curves of degree m_i and maximum polynomial integrand degree k with n_r rational curves and $n_p = n_c - n_r$ polynomial curves.

In the numerical tests, including for the results displayed in Figures 7, 8, 9, and 10, we refer to our SPECTRAL scheme as **GT-SPECTRAL** and our SPECTRAL PE scheme as **GT-SPECTRAL PE**, since they rely on Green's theorem (GT). We define integration error as the absolute value of the difference between the correct integral and the calculated integral. We calculate the correct value of each integral as the value obtained when using $P = Q_i = 55$ quadrature points in each component quadrature rule with our SPECTRAL algorithm. All plots in this section are Log-Log plots to emphasize the orders of magnitude differences in efficiency between the various methods.

For polynomial integrands with our SPECTRAL PE method, we only calculate one quadrature rule which uses the number of quadrature points required to integrate the given integrand exactly up to machine precision, given by the formula in Equation (10). For all integrands with our SPECTRAL method and for nonpolynomial integrands with our SPECTRAL PE method,

we increase the number of quadrature points in each component quadrature rule according to the strategies given in Sections 2.1, 2.2, and 2.5.3. In particular, we let the number of quadrature points in the antiderivative and intermediate quadrature rules be equal and gradually increase both in order to obtain convergence.

3.1. Comparison Methods

To show the relative benefit of our methods, we compare with three domain decomposition-based (DD) quadrature schemes and one Green's theorem-based (GT) quadrature scheme from the literature, which represent the state of the art for integrating over rational regions.

DD-Quadtree. Our implementation of quadtree integration refines a background Cartesian grid uniformly, then performs three levels of adaptive refinement anywhere where the boundary intersects the background grid and uses 3rd-order (4 point) Gaussian quadrature on each element [48].

DD-Linear mesh. We use an implementation of linear meshing from *mesh2d* which refines the discretization by remeshing the geometry with a specified maximum element size and uses 3rd-order (4 point) Gaussian quadrature on each element [32].

DD-Rational mesh. We use an implementation of exact rational meshing from TRIGA with increasing order Gaussian quadrature on each element [20].

GT-Cubic Spline. To compare with another Green's theorem-based method, we use *SplineGauss*, which approximates the boundary curves with cubic splines and uses high-order Gaussian quadrature to evaluate the antiderivatives [27]. In their terminology, we use 10th degree of precision for the polynomial integrands and 22nd degree of precision for the nonpolynomial integrands.

We note that we have not optimized any of the implementations of these algorithm (nor have we optimized implementations of our own algorithms), but that we expect the broad trends, particularly with respect to differences in pre-processing time and convergence rates, to hold true in any implementation. In our experiments, using higher-order Gaussian quadrature rules for the quadtree, linear meshing, and cubic spline strategies did not improve the error much, but yielded a much higher number of quadrature points. This implies that the geometric approximation itself is the primary source of error in these methods.

3.2. Exact Monomial Quadrature over a Circle

First, we test the algorithm in-depth on a simply connected, convex test case to show its fundamental properties. We integrate the monomial functions $1, x, y, x^2, \ldots$ up to degree 5 over a circular region Ω centered at (x_0, y_0) with radius 1.

3.2.1. Domain Definition

The circle is defined by $n_c = n_r = 4$ quadratic rational Bernstein-Bézier curves $\mathbf{c}_0, \mathbf{c}_1 \mathbf{c}_2, \mathbf{c}_3$ such that $\mathbf{c}_i = \mathbf{c}'_i - (x_0, y_0)$

where

$$
\mathbf{c}'_0 = \left\{ \begin{pmatrix} \frac{B_0(s) + \frac{\sqrt{2}}{2} B_1(s)}{B_0(s) + \frac{\sqrt{2}}{2} B_1(s) + B_2(s)} \\ \frac{\frac{\sqrt{2}}{2} B_1(s) + B_2(s)}{B_0(s) + \frac{\sqrt{2}}{2} B_1(s) + B_2(s)} \end{pmatrix} \right. \qquad
\mathbf{c}'_1 = \left\{ \begin{pmatrix} \frac{-\frac{\sqrt{2}}{2} B_1(s) - B_2(s)}{B_0(s) + \frac{\sqrt{2}}{2} B_1(s) + B_2(s)} \\ \frac{B_0(s) + \frac{\sqrt{2}}{2} B_1(s)}{B_0(s) + \frac{\sqrt{2}}{2} B_1(s) + B_2(s)} \end{pmatrix} \right.
$$

$$
\mathbf{c}'_2 = \left\{ \begin{pmatrix} \frac{-B_0(s) - \frac{\sqrt{2}}{2} B_1(s)}{B_0(s) + \frac{\sqrt{2}}{2} B_1(s) + B_2(s)} \\ \frac{-\frac{\sqrt{2}}{2} B_1(s) - B_2(s)}{B_0(s) + \frac{\sqrt{2}}{2} B_1(s) + B_2(s)} \end{pmatrix} \right. \qquad
\mathbf{c}'_3 = \left\{ \begin{pmatrix} \frac{\frac{\sqrt{2}}{2} B_1(s) + B_2(s)}{B_0(s) + \frac{\sqrt{2}}{2} B_1(s) + B_2(s)} \\ \frac{-B_0(s) - \frac{\sqrt{2}}{2} B_1(s)}{B_0(s) + \frac{\sqrt{2}}{2} B_1(s) + B_2(s)} \end{pmatrix} \right. ,
$$

and $B_0(s) = (1-s)^2$, $B_1(s) = 2(1-s)s$, and $B_2(s) = s^2$. The quadrature points produced using the SPECTRAL PE algorithm with a maximal degree of polynomial exactness of $k = 3$ are shown in Figure 6. The SPECTRAL algorithm produces similarly distributed quadrature points.

As can be seen in Figure 7, the SPECTRAL PE quadrature strategy yields exact results with the expected number of quadrature points $n_q = 24$ quadrature points in the case of $p(x, y) = 1$ and $n_q = 64$ quadrature points in the case of $p(x, y) = x, y$, as per Eq. (10). Moreover, the SPECTRAL algorithm produces quadrature rules which converge faster than any algebraic order with respect to number of quadrature points and are orders of magnitude more efficient than the comparison methods.

3.3. Exact Polynomial Quadrature over Rational Shapes

We also test our algorithms on a variety of more complicated shapes, including (1) a plate with a hole, (2) an L-bracket with three holes, (3) a guitar-shaped region, (4) a treble clef-shaped region, and (5) (6) a region representing a cross section of two interlocked screws, [47] The regions are displayed in Figure 8.

3.3.1. Domain definition

The first two domains (plate with hole and L-bracket with 3 holes) were formed by taking finite Boolean (i.e. unions, intersections, and differences) combinations of quadratic rational Bézier regions with known control weights and control points (namely, rectangles and ellipses). Booleans of arbitrary rational Bernstein-Bézier shapes can be computed using methods briefly outlined in Appendix A. This same procedure could be used to find boundary parameterizations for intersections of elements in, for example, immersogeometric analysis. The plate with a hole is composed of $M = 2$ boundary loops and the boundary contains a total of $n_r = 4$ quadratic ($m = 2$) rational Bézier curves and $n_c = 4$ linear polynomial Bézier curves. The L-bracket with 3 holes is composed of $M = 4$ boundary loops and the boundary contains a total of $n_r = 14$ quadratic ($m = 2$) rational Bézier curves and $n_p = 6$ linear polynomial Bézier curves.

The guitar, treble clef, and screws regions were designed by editing the control weights of NURBS curves, then projecting the NURBS curves onto rational Bézier curves. The algorithm for converting NURBS to rational Bernstein-Bézier curves is briefly discussed in Appendix B. The guitar region is composed of $M = 2$ boundary loops with a total of $n_c = 60$ component cubic ($m = 3$) rational Bézier curves. The treble clef region is composed of $M = 4$ boundary loops containing $n_c = 99$ quadratic ($m = 2$) rational Bézier curves. The two interlocked screws region is composed of $M = 3$ boundary loops with a total of $n_c = 128$ cubic ($m = 3$) rational Bézier curves.
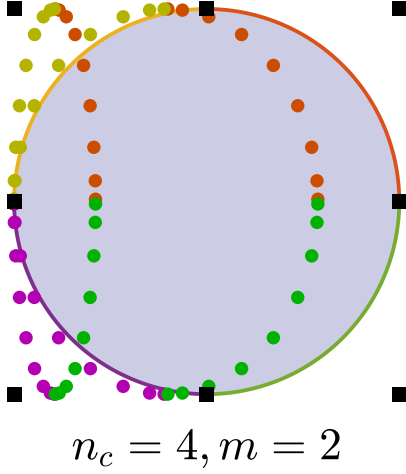
$$n_c = 4, m = 2$$

Figure 6: SPECTRAL PE quadrature points for a circle defined by 4 rational parametric Bernstein-Bézier curves integrating polynomials up to degree $k = 3$ exactly. Quadrature points are contained within the bounding box of the curve control points (black squares). The quadrature points are color-coded according to their corresponding component curve in the evaluation of Green's theorem.
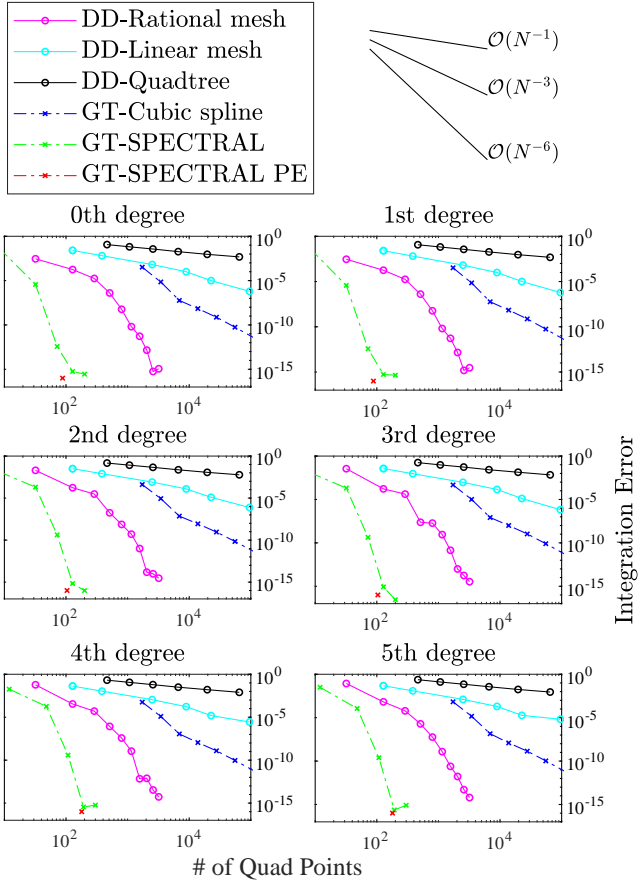

Figure 7: Integrals of monomials over the circle can be computed exactly using the SPECTRAL PE algorithm. Moreover, the SPECTRAL algorithm achieves faster convergence than any algebraic order and is much more efficient than other common methods. Note that the error values are averages over the monomials of a particular degree. In this case, all curves are of degree $m_i = 2$.

3.3.2. Results

We applied the SPECTRAL and SPECTRAL PE algorithms to the shapes defined above for various polynomials of degrees 2, 4, and 6. As can be seen in Figure 8, both of the presented quadrature schemes outperform the four comparison methods in terms of number of quadrature points for a given degree of accuracy by orders of magnitude. In addition, the SPECTRAL PE scheme is able to achieve exactness with the expected number of quadrature points. We observed the same behavior for a variety of polynomial integrands, including higher-degree polynomials. We note here that the comparison methods achieve the expected convergence rates of 1st order convergence for linear mesh and quadtree integration, 3rd order convergence for cubic spline integration, and spectral convergence for rational mesh integration.

3.4. Quadrature of Non-polynomial Integrands over Rational Shapes

We also compare our algorithms on a representative set of non-polynomial integrands, including a rational integrand, an exponential integrand, and a square root integrand. These integrands require adaptive integration, even for the SPECTRAL PE algorithm. The *a priori* formula for the number of quadrature points will not guarantee exactness, since the antiderivative computation using Gaussian quadrature is no longer exact in this case. We therefore use varying amounts of quadrature points in the SPECTRAL PE algorithm to observe convergence of the method. To do this, we find a quadrature rule exact for polynomials up to degree $k = 2$, then increase the number of quadrature points in the intermediate rational quadrature algorithm as explained in Section 2.5.3 (by increasing l) and in the antiderivative Gaussian quadrature order as explained in Section 2.2 (by increasing P), while keeping the number of quadrature points in each component rule equal.

As can be seen in Figure 9, both the SPECTRAL and SPECTRAL PE methods are orders of magnitude more efficient than other methods in the literature on a per-quadrature point basis and converge to machine precision with far fewer quadrature points. We observed similar behavior for a variety of non-polynomial integrands. In this case, it is likely more beneficial to use the SPECTRAL algorithm, since it has the same convergence behavior as the SPECTRAL PE algorithm, but is likely relatively simpler to implement.

3.5. Timing Results

In addition to performing more efficiently with respect to the number of quadrature points, we found that both the SPECTRAL and SPECTRAL PE quadrature schemes are also orders of magnitude more efficient with respect to computational time required for nearly all cases shown in the previous sections. As representative examples, we show timing results in Figure 10 to calculate the various quadrature points and weights (pre-processing) and apply them to a single function (evaluation) for the lower-rightmost plot in each of Figures 8 and 9. Timings were performed using MATLAB's `timeit` function. As can be seen, the present algorithms require less evaluation time than any of the comparison methods for a given level of accuracy. In addition, the present algorithms require far less pre-processing time than the only other spectral method, rational meshing.

Timing tests were performed using MATLAB version R2019a on Windows 10 with 32GB of RAM. It is important
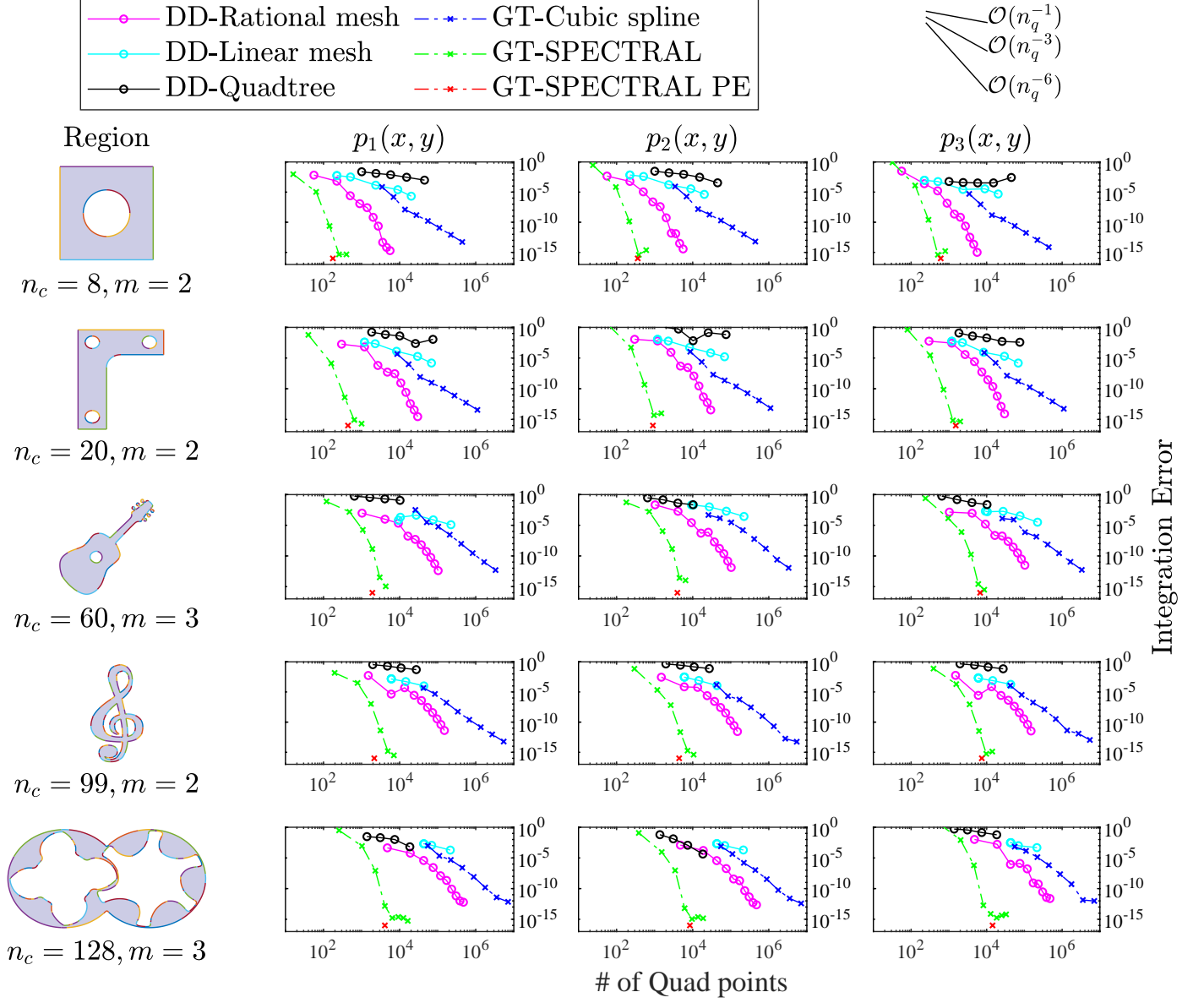
10

Figure 8: Integrals of polynomials over arbitrary rational parametric regions can be computed exactly using the presented SPECTRAL PE quadrature scheme. Moreover, the SPECTRAL scheme converges faster than any algebraic order and is much more efficient than other methods in the literature. The left column shows the region being integrated in light grey and the different colors on the boundaries represent the component curves $\{\mathbf{c}_i\}_{i=1}^{n_c}$. See Section 3.1 for a more detailed description of the comparison methods. The integrands are $p_1(x,y) = 2x^2 + xy - y + 2$, $p_2(x,y) = 2x^2y^2 + \frac{3}{10}x^2y - y^4 + 3x + 2$, and $p_3(x,y) = x^5 - 5y^3x^3 + 2yx^2 + \frac{1}{5}x^2 + 3$.
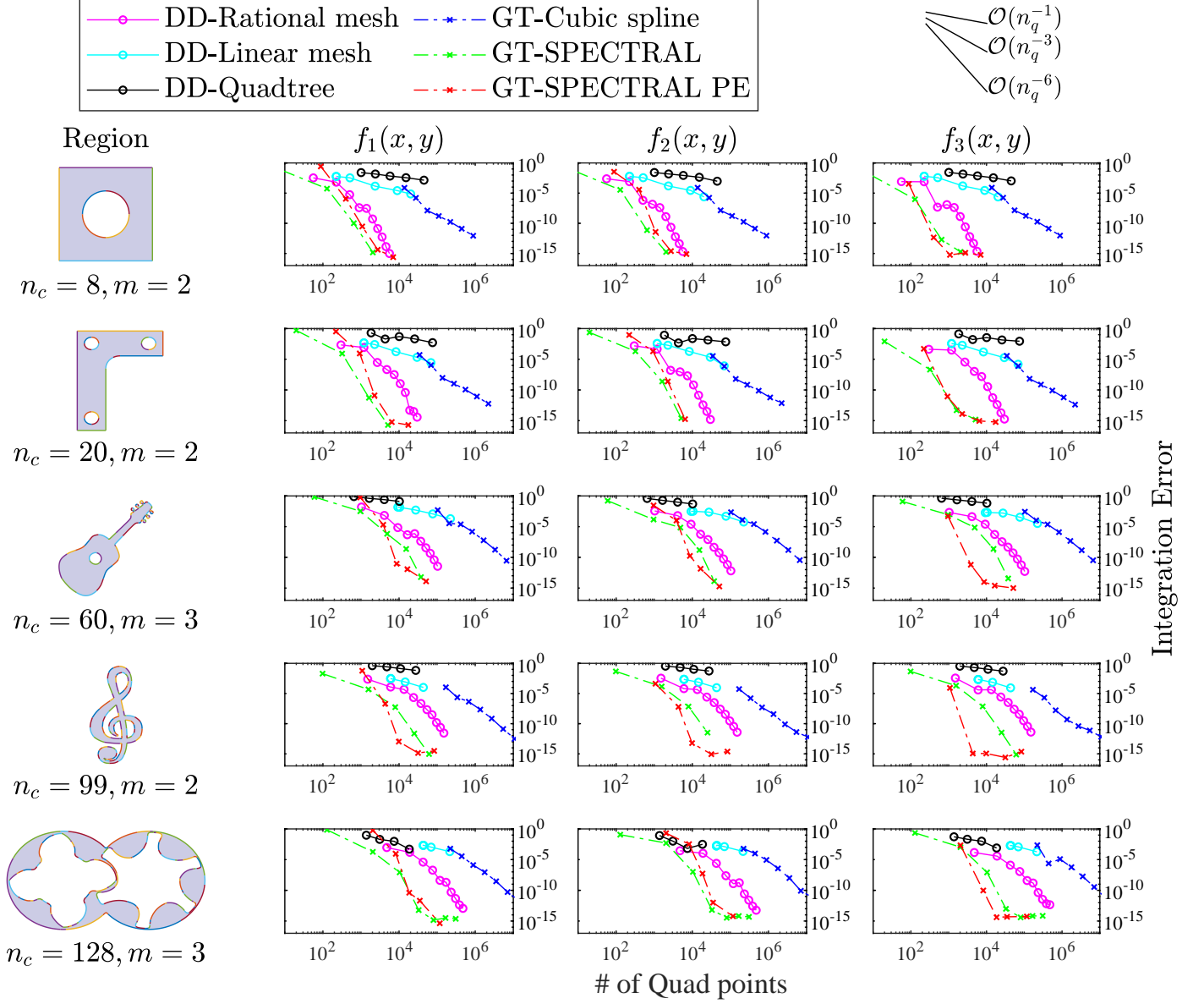
DD-Rational mesh — — ×— — GT-Cubic spline

DD-Linear mesh — · — ×— · — GT-SPECTRAL

DD-Quadtree — · — ×— · — GT-SPECTRAL PE

$\mathcal{O}(n_q^{-1})$
$\mathcal{O}(n_q^{-3})$
$\mathcal{O}(n_q^{-6})$

Region $\qquad\qquad f_1(x,y) \qquad\qquad f_2(x,y) \qquad\qquad f_3(x,y)$

$n_c = 8, m = 2$

$n_c = 20, m = 2$

$n_c = 60, m = 3$

$n_c = 99, m = 2$

$n_c = 128, m = 3$

Integration Error

# of Quad points

Figure 9: Both the SPECTRAL and the SPECTRAL PE methods given in Section 2 converge faster than any algebraic order on a broad class of nonpolynomial integrands. The left column shows the region being integrated in light grey and the different colors on the boundaries represent the component curves $\{c_i\}_{i=1}^{n_c}$. See Section 3.1 for a more detailed description of the comparison methods. The integrands are $f_1(x,y) = \frac{y^3 - x^3y^2 - xy - 3}{x^2y^2 + 30}$, $f_2(x,y) = e^{-x^2 + 2y}$, and $f_3(x,y) = \sqrt{(x+10)^2 + (x+10)(y+10) + x}$.
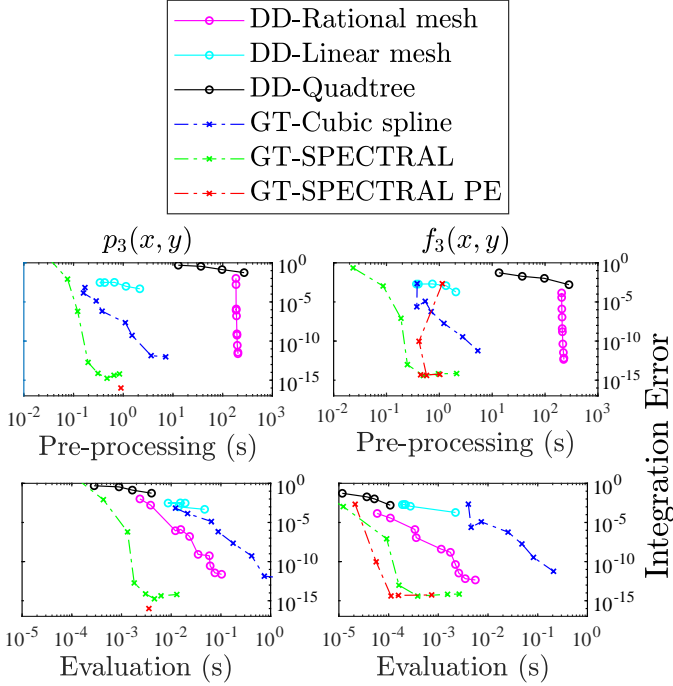
Figure 10: Our proposed methods are much more efficient for a given error in terms of computational time. Pre-processing (top) and evaluation (bottom) timings to integrate functions p_3 and f_3 for the interlocked screws mesh. Timings correspond to the lower-right most plots of Figure 8 (left) and Figure 9 (right).

to note that the timings given here use non-optimized MATLAB code. Results would likely change with optimized compiled implementations, although we don't expect changes in the trends or relative performance of the various algorithms, particularly with respect to convergence behavior and pre-processing time.

4. Discussion

In this work, we have presented two spectral meshless algorithms for computing quadrature rules for arbitrary planar regions bounded by rational curves. Both the SPECTRAL and SPECTRAL PE algorithms converge faster than any algebraic order with respect to total number of quadrature points for arbitrary smooth integrals. Moreover, the SPECTRAL PE algorithm is exact (up to machine precision) for polynomial integrands up to a pre-specified degree, k. In contrast to other schemes in the literature, which either require expensive meshing to achieve spectral convergence or have algebraic orders of convergence, both of our algorithms achieve spectral convergence without meshing. Moreover, our SPECTRAL PE method prescribes the number of quadrature points necessary to exactly integrate polynomial functions over these regions, whereas other methods must spend extra effort adaptively converging to the correct solution.

We presented numerical test cases which show that the SPECTRAL PE algorithm is exact for computing integrals of polynomials over rational geometries, reducing the number of quadrature points necessary to achieve a given precision for polynomial integrands even compared with our SPECTRAL rule. We also presented numerical test cases which show that both the SPECTRAL and SPECTRAL PE algorithms presented here are much more efficient than other methods when integrating arbitrary

functions over rational geometry, both on a per-quadrature point basis and a computational time basis.

Our intention is for this paper to be the first in a series which describes approaches for high-order or spectrally convergent numerical quadrature over general parametric regions in $\mathbb{R}^2$ and $\mathbb{R}^3$. The broad idea is to iteratively use the parametric version of generalized Stokes' theorem combined with numerical antiderivative computation to compute efficient quadrature rules. This paper outlines the approach applied to integration over arbitrary two-dimensional rational (and, as a special case, polynomial) parametric regions and achieves spectral convergence. Part two will address the general three-dimensional parametric case using an extension of the algorithm presented here and we expect it to attain high algebraic orders of convergence.

Another potential avenue for future research could include expanding the SPECTRAL PE scheme for polynomial integrands to 3D rational geometries, although the poles of rational parametric surfaces can in general be algebraic curves, for which exact rational quadrature has not been studied extensively. We would also like to apply these methods to immersogeometric analysis, field transfer between high-order meshes, and VOF initialization.

Acknowledgments

Appendix A. Booleans of Rational Shapes

Intersections, unions, and differences of rational shapes can be computed using the algorithm given in Section 2 by using Bézier intersection and reparametrization. An algorithm for computing parametrizations of Boolean combinations (i.e. unions, intersections, and differences) of regions defined by polynomial triangular Bernstein-Bézier elements is described in [49]. We have adapted this algorithm to the rational case to help design some of the shapes for our empirical results in Section 3. Our approach combined with efficient intersection could be used to efficiently perform the integration necessary in immersogeometric analysis, field transfer between high-order meshes, and VOF field initialization, in which integrals over intersections between rational geometries must be computed.

Appendix B. NURBS

The algorithm described in Section 2 can easily be used on regions bounded by non-uniform rational basis splines (NURBS). Because NURBS can be decomposed exactly into rational Bézier curves, the algorithm only requires one extra step: Bézier extraction [3], which converts a degree-m NURBS curve into component degree-m rational Bézier curves.

References

[1] T. J. Hughes, J. A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Computer Methods in Applied Mechanics and Engineering 194 (39-41) (2005) 4135–4195.

[2] Y. Bazilevs, V. M. Calo, J. A. Cottrell, J. A. Evans, T. J. R. Hughes, S. Lipton, M. A. Scott, T. W. Sederberg, Isogeometric analysis using T-splines, Computer Methods in Applied Mechanics and Engineering 199 (5-8) (2010) 229–263.

[3] M. J. Borden, M. A. Scott, J. A. Evans, T. J. Hughes, Isogeometric finite element data structures based on Bézier extraction of NURBS, International Journal for Numerical Methods in Engineering 87 (1-5) (2011) 15–47.

[4] D. Kamensky, M.-C. Hsu, D. Schillinger, J. A. Evans, A. Aggarwal, Y. Bazilevs, M. S. Sacks, T. J. Hughes, An immersogeometric variational framework for fluid–structure interaction: Application to bioprosthetic heart valves, Computer Methods in Applied Mechanics and Engineering 284 (2015) 1005–1053.

[5] D. Schillinger, L. Dede, M. A. Scott, J. A. Evans, M. J. Borden, E. Rank, T. J. Hughes, An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces, Computer Methods in Applied Mechanics and Engineering 249 (2012) 116–150.

[6] A. Krishnamurthy, S. McMains, Accurate moment computation using the GPU, in: Proceedings of the 14th ACM Symposium on Solid and Physical Modeling, ACM, 2010, pp. 81–90.

[7] E. Burman, P. Hansbo, Fictitious domain finite element methods using cut elements: I. A stabilized Lagrange multiplier method, Computer Methods in Applied Mechanics and Engineering 199 (41-44) (2010) 2680–2686.

[8] R. W. Anderson, V. A. Dobrev, T. V. Kolev, R. N. Rieben, V. Z. Tomov, High-order multi-material ALE hydrodynamics, SIAM Journal on Scientific Computing 40 (1) (2018) B32–B58.

[9] J. C. Verschaeve, High order interface reconstruction for the volume of fluid method, Computers & Fluids 46 (1) (2011) 486–492.

[10] V. Dyadechko, M. Shashkov, Moment-of-fluid interface reconstruction, Los Alamos Report LA-UR-05-7571 (2005).

[11] A. Taber, G. Kumar, M. Freytag, V. Shapiro, A moment-vector approach to interoperable analysis, Computer-Aided Design 102 (2018) 139–147.

[12] V. Thiagarajan, V. Shapiro, Adaptively weighted numerical integration over arbitrary domains, Computers & Mathematics with Applications 67 (9) (2014) 1682–1702.

[13] F. Scholz, A. Mantzaflaris, B. Jüttler, First order error correction for trimmed quadrature in isogeometric analysis, in: Chemnitz Fine Element Symposium, Springer, 2017, pp. 297–321.

[14] J. Flusser, T. Suk, On the calculation of image moments, Research Report #1946 1946 (1999).

[15] C.-H. Wu, S.-J. Horng, P.-Z. Lee, A new computation of shape moments via quadtree decomposition, Pattern Recognition 34 (7) (2001) 1319–1330.

[16] M. H. Singer, A general approach to moment calculation for polygons and line segments, Pattern Recognition 26 (7) (1993) 1019–1028.

[17] L. Yang, F. Albregtsen, Fast and exact computation of Cartesian geometric moments using discrete Green's theorem, Pattern Recognition 29 (7) (1996) 1061–1073.

[18] X. Roca, A. Gargallo-Peiró, J. Sarrate, Defining quality measures for high-order planar triangles and curved mesh generation, in: Proceedings of the 20th International Meshing Roundtable, Springer, 2011, pp. 365–383.

[19] S. Sherwin, J. Peiró, Mesh generation in curvilinear domains using high-order elements, International Journal for Numerical Methods in Engineering 53 (1) (2002) 207–223.

[20] L. Engvall, J. A. Evans, Isogeometric triangular Bernstein–Bézier discretizations: Automatic mesh generation and geometrically exact finite element analysis, Computer Methods in Applied Mechanics and Engineering 304 (2016) 378–407.

[21] L. Engvall, J. A. Evans, Isogeometric unstructured tetrahedral and mixed-element Bernstein–Bézier discretizations, Computer Methods in Applied Mechanics and Engineering 319 (2017) 83–123.

[22] L. Engvall, J. A. Evans, Mesh quality metrics for isogeometric Bernstein–Bézier discretizations, arXiv preprint arXiv:1810.06975 (2018).

[23] E. B. Chin, J. B. Lasserre, N. Sukumar, Numerical integration of homogeneous functions on convex and nonconvex polygons and polyhedra, Computational Mechanics 56 (6) (2015) 967–981.

[24] Y. Sudhakar, W. A. Wall, Quadrature schemes for arbitrary convex/concave volumes and integration of weak form in enriched partition of unity methods, Computer Methods in Applied Mechanics and Engineering 258 (2013) 39–54.

[25] R. Saye, High-order quadrature methods for implicitly defined surfaces and volumes in hyperrectangles, SIAM Journal on Scientific Computing 37 (2) (2015) A993–A1019.

[26] A. Sommariva, M. Vianello, Meshless cubature by Greens formula, Applied mathematics and computation 183 (2) (2006) 1098–1107.

[27] A. Sommariva, M. Vianello, Gauss–Green cubature and moment computation over arbitrary geometries, Journal of Computational and Applied Mathematics 231 (2) (2009) 886–896.

[28] G. Santin, A. Sommariva, M. Vianello, An algebraic cubature formula on curvilinear polygons, Applied Mathematics and Computation 217 (24) (2011) 10003–10015.

[29] S. Sheynin, A. Tuzikov, Moment computation for objects with spline curve boundary, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (10) (2003) 1317–1322.

[30] B. Li, The moment calculation of polyhedra, Pattern Recognition 26 (8) (1993) 1229–1233.

[31] T. Jonsson, M. G. Larson, K. Larsson, Cut finite element methods for elliptic problems on multipatch parametric surfaces, Computer Methods in Applied Mechanics and Engineering 324 (2017) 366–394.

[32] D. Engwirda, Locally optimal Delaunay-refinement and optimisation-based mesh generation (2014).

[33] G. Farin, Curves and surfaces for CAGD: A practical guide, Morgan Kaufmann, 2002.

[34] R. T. Farouki, The Bernstein polynomial basis: A centennial retrospective, Computer Aided Geometric Design 29 (6) (2012) 379–419. doi:10.1016/j.cagd.2012.03.001.

[35] P. J. Davis, P. Rabinowitz, Methods of numerical integration, Courier Corporation, 2007.

[36] W. Gautschi, Orthogonal polynomials and quadrature, Electron. Trans. Numer. Anal 9 (1999) 65–76.

[37] M. Djrbashian, A survey on the theory of orthogonal system and some open problems, in: Orthogonal polynomials, Springer, 1990, pp. 135–146.

[38] W. Gautschi, Gauss-type quadrature rules for rational functions, in: Numerical integration IV, Springer, 1993, pp. 111–130.

[39] K. Deckers, J. Van Deun, A. Bultheel, Rational Gauss-Chebyshev quadrature formulas for complex poles outside [-1, 1], Mathematics of Computation 77 (262) (2008) 967–983.

[40] J. Van Deun, K. Deckers, A. Bultheel, J. Weideman, Algorithm 882: Near-best fixed pole rational interpolation with applications in spectral methods, ACM Transactions on Mathematical Software (TOMS) 35 (2) (2008) 14.

[41] K. Deckers, J. Van Deun, A. Bultheel, Computing rational Gauss–Chebyshev quadrature formulas with complex poles: The algorithm, Advances in Engineering Software 40 (8) (2009) 707–717.

[42] K. Deckers, A. Mougaida, H. Belhadjsalah, Algorithm 973: Extended rational Fejér quadrature rules based on Chebyshev orthogonal rational functions, ACM Transactions on Mathematical Software (TOMS) 43 (4) (2017) 37.

[43] A. Edelman, H. Murakami, Polynomial roots from companion matrix eigenvalues, Mathematics of Computation 64 (210) (1995) 763–776.

[44] Matlab roots function, the MathWorks, Natick, MA, USA (2019a).

[45] R. T. Farouki, V. Rajan, Algorithms for polynomials in Bernstein form, Computer Aided Geometric Design 5 (1) (1988) 1–26.

[46] J. Benzaken, A. J. Herrema, M.-C. Hsu, J. A. Evans, A rapid and efficient isogeometric design space exploration framework with application to structural mechanics, Computer Methods in Applied Mechanics and Engineering 316 (2017) 1215–1256.

[47] J. Hinz, M. Möller, C. Vuik, Spline-based parameterization techniques for twin-screw machine geometries, in: IOP Conference Series: Materials Science and Engineering, Vol. 425, IOP Publishing, 2018, p. 012030.

[48] L. Kudela, N. Zander, T. Bog, S. Kollmannsberger, E. Rank, Efficient and accurate numerical quadrature for immersed boundary methods, Advanced Modeling and Simulation in Engineering Sciences 2 (1) (2015) 10.

[49] D. Hermes, P.-O. Persson, High-order solution transfer between curved triangular meshes, arXiv preprint arXiv:1810.06806 (2018).