Numerical Quadrature for Gregory Quads

Jun Zhou[1], Pieter J. Barendrecht[2], Michael Bartoň[3,4], Jiří Kosinka[1,*]

[1]*Bernoulli Institute, University of Groningen, Nijenborgh 9, 9747 AG, Groningen, the Netherlands*

[2]*Department of Mathematics and Computer Science, University of Basel, Spiegelgasse 1, 4051 Basel, Switzerland*

[3]*Basque Center for Applied Mathematics, Alameda Mazarredo 14, 48009 Bilbao, Spain*

[4]*Ikerbasque – Basque Foundation for Sciences, María Díaz de Haro 3, 48013 Bilbao, Basque Country, Spain*

Abstract

We investigate quadrature rules in the context of quadrilateral Gregory patches, in short Gregory quads. We provide numerical and where possible symbolic quadrature rules for the space spanned by the twenty polynomial/rational functions associated with Gregory quads, as well as the derived spaces including derivatives, products, and products of derivatives of these functions. This opens up the possibility for a wider adoption of Gregory quads in numerical simulations.

Keywords: Numerical integration, Gregory patches, Gaussian quadrature rules

1. Introduction

Numerical quadratures are indispensable in most numerical analyses and simulations. They provide an efficient means of numerically evaluating or approximating integrals of (spline) functions over various domains [1]. In the bivariate setting, a quadrature rule for a function f from a certain space is defined as

$$\int_{\Omega} f(\mathbf{x})\,\mathrm{d}\mathbf{x} = \sum_{i=1}^{m} \omega_i f(\mathbf{t}_i) + R_m(f), \tag{1}$$

where $\Omega \subset \mathbb{R}^2$ represents the domain under consideration, m is the number of quadrature points $\mathbf{t}_i$ (often called nodes) and their corresponding weights ω_i, and $R_m(f)$ represents the error term of the rule. The rule is said to be exact if the error term vanishes for all elements of a linear function space under consideration. The rule is called Gaussian if there is no numerically exact rule that uses fewer nodes. In this paper, we focus on the basis functions associated with Gregory quads as well as the derived spaces useful in the context of numerical analysis.

As isogeometric analysis [2] gained in popularity, so did the search for (Gaussian) quadrature rules for various function and spline spaces. In the fully structured tensor-product setting, the situation is fairly well developed and understood. B-spline quadrature has been studied in [3, 4, 5, 6], NURBS quadrature in [7, 8], and the tensor-product setting has been dealt with in [9, 10, 11, 12].

In contrast, the case of non-tensor product splines and other function spaces has received much less attention. To date, only a few recent advances in this direction have been made. In the case of Catmull-Clark subdivision, an approach to the quadrature for subdivision surfaces based on Catmull-Clark quad meshes was introduced in [13, 14]. This research discussed grouping strategies of quads around extraordinary vertices (EVs), as the surface around EVs has non-tensor product structure. This is an improvement over the naive, per-quad integration. In the area of triangles, the quadrature rules for spline spaces over macro-triangles have been studied in [15, 16]. The Hammer and Stroud

*Corresponding author
Email address: `j.kosinka@rug.nl` (Jiří Kosinka)

rules generalise to the macro-element spaces of Clough-Tocher and Powell-Sabin provided that some conditions on the split points are met.

Quadrature rules for rational (spline) spaces have not received much attention [7]. Gregory quads, the focus of the present paper, which belong to the class of rational surfaces, were used in [17, 18, 19]. The last reference mentions the use of a 4×4 quadrature scheme, which is in general non-exact in the given context. We improve on this in this paper.

As shown in [20], Gregory patches provide an effective way of constructing globally G^1 surfaces of arbitrary manifold topology defined by quadrilateral meshes, for example as approximations to the more numerically demanding Catmull-Clark subdivision surfaces. This further motivated us to investigate quadrature rules for the space of functions spanned by the basis functions of Gregory quads as well as for the derived spaces needed in (isogeometric) analysis.

We start by introducing the concepts and basis functions behind bicubic Bézier patches and quadrilateral Gregory patches in Section 2, as well as standard techniques for deriving quadrature rules. Our results regarding the quadrature rules for quadrilateral Gregory patches are presented in Section 3 for the basis functions themselves. This is followed by rules for partial derivatives (Section 4) and mixed derivatives (Section 5). In Section 6 we devise a numerical approach to quadrature finding, which we then apply to the cases of basis function products and products of their derivatives in Section 7. Several numerical tests of the quadratures are presented in Section 8. The paper is concluded in Section 9.

2. Preliminaries

To fix notation and for the convenience of the reader, we recall here the definitions of bicubic Bézier patches and the Gaussian quadrature rule for their basis functions (Section 2.1), Gregory quads (Section 2.2), and provide a brief overview of standard methods for deriving quadrature rules (Section 2.3).

2.1. *Bicubic Bézier Patches*

A bicubic Bézier patch is a parametric surface of polynomial bidegree $(3, 3)$ defined over the unit square $\square \subset \mathbb{R}^2$. Let $I = \{0, 1, 2, 3\}$. Given 16 control points p_{ij}, $(i, j) \in I^2$ (see Figure 1, left), the bicubic Bézier patch is defined as

$$p(u, v) = \sum_{(i,j)\in I^2} p_{i,j} B_{i,j}(u, v), \quad (u, v) \in \square, \tag{2}$$

where $B_{i,j}(u, v) = B_i(u)B_j(v)$ with

$$B_k(w) = \frac{3!}{k!(3-k)!} w^k (1-w)^{3-k}, \quad k = 0, \ldots, 3 \tag{3}$$

the cubic Bernstein polynomials. The 16 polynomial functions span the linear space

$$\mathcal{B} = \operatorname{span}\left(B_{i,j}, (i, j) \in I^2\right)$$

with $\dim(\mathcal{B}) = 16$.

Table 1: The four quadrature nodes and weights for bicubic polynomials over the unit square.

i	u_i	v_i	ω_i
1	$\frac{1}{2} - \frac{1}{6}\sqrt{3}$	$\frac{1}{2} - \frac{1}{6}\sqrt{3}$	$\frac{1}{4}$
2	$\frac{1}{2} + \frac{1}{6}\sqrt{3}$	$\frac{1}{2} + \frac{1}{6}\sqrt{3}$	$\frac{1}{4}$
3	$\frac{1}{2} - \frac{1}{6}\sqrt{3}$	$\frac{1}{2} + \frac{1}{6}\sqrt{3}$	$\frac{1}{4}$
4	$\frac{1}{2} + \frac{1}{6}\sqrt{3}$	$\frac{1}{2} - \frac{1}{6}\sqrt{3}$	$\frac{1}{4}$

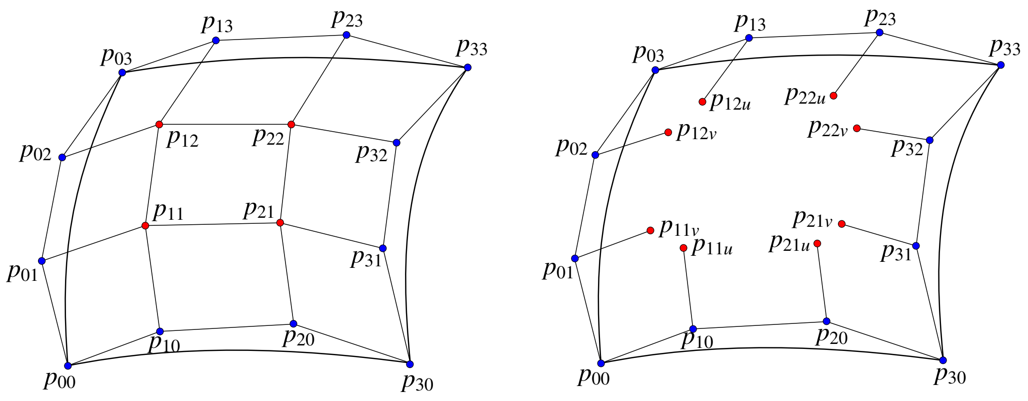

Figure 1: The control net and labeling scheme of a bicubic Bézier patch (left) and a Gregory patch (right).

It is well known that $\mathcal{B}$ requires four quadrature nodes, known as the 2×2 tensor product Gauss-Legendre rule. Specifically,

$$\int_{\square} B(u, v)\ \mathrm{d}u\ \mathrm{d}v = \sum_{i=1}^{4} \omega_i B(u_i, v_i), \tag{4}$$

with the nodes (u_i, v_i) and weights ω_i specified in Table 1, for any function $B(u, v) \in \mathcal{B}$. Note the symmetry of the rule with respect to the bimedians and diagonals of $\square$; we will rely on this later on.

2.2. *Gregory quads*

Quadrilateral Gregory patches [21] are a generalisation of bicubic Bézier surfaces. At the expense of introducing rational blending functions, they provide the possibility to join patches with G^1 continuity over unstructured quad meshes [20, 22]. This is achieved by separating the two mixed partial derivatives at each of the four patch corners. Each of the four internal control points of a bicubic Bézier patch is 'split' into two control points, and each such pair is blended rationally into one 'moving' point according to the $(u, v) \in \square$ parameter position. More precisely,

$$\begin{aligned} p_{11} &= p_{11}(u,v) = \frac{u p_{11u} + v p_{11v}}{u+v}, \\ p_{21} &= p_{21}(u,v) = \frac{(1-u) p_{21u} + v p_{21v}}{1-u+v}, \\ p_{22} &= p_{22}(u,v) = \frac{(1-u) p_{22u} + (1-v) p_{22v}}{2-u-v}, \\ p_{12} &= p_{12}(u,v) = \frac{u p_{12u} + (1-v) p_{12v}}{1+u-v}, \end{aligned} \tag{5}$$

with removable singularities [23] at the four corners of $\square$. The other 12 control points (along the boundary) are associated with the same basis functions as in the bicubic Bézier quad. As such, the boundaries of a Gregory quad are cubic Bézier curves. Note that there exist also *rational* boundary Gregory patches [24], but these are not considered here. The Gregory quad is thus governed by 20 control points, indexed by the set (see Figure 1, right)

$$J = \{00, 01, 02, 03, 10, 13, 20, 23, 30, 31, 32, 33, 11u, 11v, 12u, 12v, 21u, 21v, 22u, 22v\},$$

and defined as

$$q(u, v) = \sum_{j \in J} p_j G_j(u, v), \quad (u, v) \in \square. \tag{6}$$

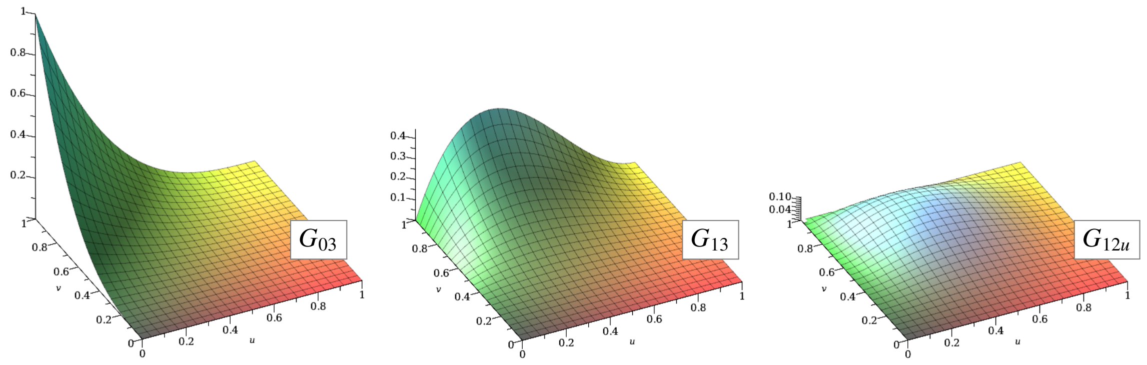

Figure 2: Up to symmetries, there are only three basis function shapes in $\mathcal{G}$. From left to right: G_{03}, G_{13}, and G_{12u}.

The 20 functions, some of which are shown in Figure 2, span the linear space

$$\mathcal{G} = \operatorname{span}\left(G_j, j \in J\right),$$

where

$$\begin{aligned}
&G_{00} = (1-u)^3(1-v)^3, && G_{01} = 3(1-u)^3 v(1-v)^2, && G_{02} = 3(1-u)^3 v^2(1-v), && G_{03} = (1-u)^3 v^3,\\
&G_{10} = 3u(1-u)^2(1-v)^3, && G_{13} = 3u(1-u)^2 v^3, && G_{20} = 3u^2(1-u)(1-v)^3, && G_{23} = 3u^2(1-u)v^3,\\
&G_{30} = u^3(1-v)^3, && G_{31} = 3u^3 v(1-v)^2, && G_{32} = 3u^3 v^2(1-v), && G_{33} = u^3 v^3,\\
&G_{11u} = \tfrac{9u^2(1-u)^2 v(1-v)^2}{(u+v)}, && G_{11v} = \tfrac{9u(1-u)^2 v^2(1-v)^2}{u+v}, && G_{21u} = \tfrac{9u^2(1-u)^2 v(1-v)^2}{(1-u+v)}, && G_{21v} = \tfrac{9u^2(1-u)v^2(1-v)^2}{(1-u+v)},\\
&G_{22u} = \tfrac{9u^2(1-u)^2 v^2(1-v)}{2-u-v}, && G_{22v} = \tfrac{9u^2(1-u)v^2(1-v)^2}{2-u-v}, && G_{12u} = \tfrac{9u^2(1-u)^2 v^2(1-v)}{1+u-v}, && G_{12v} = \tfrac{9u(1-u)^2 v^2(1-v)^2}{1+u-v}.
\end{aligned} \tag{7}$$

We now show that these functions are in fact basis functions and that $\dim(\mathcal{G}) = 20$. While this may be considered obvious, we have not found a formal proof of this fact in the existing literature.

Theorem 2.1. *The twenty Gregory quad functions G_j, $j \in J$ are linearly independent and therefore* $\dim(\mathcal{G}) = 20$.

Proof. Suppose for contradiction that $\sum_{k \in J} c_k G_k = 0$ over $\square$ where not all c_k vanish. Using G_j also as 'test' functions and integrating over $\square$ gives the 20×20 system $\mathbf{Nc} = \mathbf{0}$, where the elements of $\mathbf{N}$ have the form $\int_\square G_j G_k \, \mathrm{d}u \, \mathrm{d}v$ with $j, k \in J$ and the elements of $\mathbf{c}$ are simply c_k. Any solution $\mathbf{c}$ must be an element of $\ker(\mathbf{N})$. A direct computation confirms that $\dim(\ker(\mathbf{N})) = 0$, which concludes the proof. $\square$

2.3. *Classical Techniques for Deriving Quadratures*

We now review classical quadrature derivation techniques, outline why they are insufficient in our context, and prepare the ground for our contribution in this direction.

When considering a univariate function space $\mathcal{S}$ spanned by a weighted polynomial basis — that is, a polynomial basis where each function is multiplied by one and the same *weighting function* $w(x)$ — quadratures can be derived by considering polynomials $P_k(x)$ that are orthogonal with respect to a known *weighted* inner product (using the same weighting function as mentioned earlier) on a certain domain Ω. It works by dividing a function $w(x)f(x) \in \mathcal{S}$, where $f(x)$ is a polynomial of at most degree $2d-1$, by the polynomial function $P_d(x)$ of degree d. This yields $w(x)f(x) = w(x)Q(x)P_d(x) + w(x)R(x)$, where both $Q(x)$ and $R(x)$ are polynomials of at most degree $d-1$. Expressing $Q(x) = \sum_k c_k P_k(x)$ and subsequently integrating over Ω then gives $\int_\Omega w(x)f(x) \, \mathrm{d}x = \int_\Omega w(x)R(x) \, \mathrm{d}x$ because of orthogonality. Finally, expressing $R(x)$ in a degree $d-1$ Lagrange basis using the d roots x_m of $P_d(x)$ as nodes, i.e., $R(x) = \sum_m d_m L_m(x)$ with $d_m = R(x_m)$, the integral of $w(x)f(x)$ follows as

$$\sum_m R(x_m) \int_\Omega w(x)L_m(x) \, \mathrm{d}x = \sum_m f(x_m) \int_\Omega w(x)L_m(x) \, \mathrm{d}x = \sum_m f(x_m) w_m.$$

A familiar example are the Gauss-Legendre rules, which follow from the Legendre basis which is orthogonal with respect to the standard L^2 inner product (with $w(x) = 1$). This means that the resulting quadratures apply to polynomial spaces. Other examples include Gauss-Jacobi and Gauss-Chebyshev quadratures, which are based on the Jacobi and Chebyshev polynomials orthogonal with respect to the inner products using $w(x) = (1-x)^\alpha(1+x)^\beta$ and $w(x) = \frac{1}{\sqrt{1-x^2}}$, respectively.

In the setting where a weighting function is given but where its associated set of orthogonal polynomials is unknown, another method considering the *moments* of $w(x)$ can be used to derive quadratures. These moments M_k are the values of the integrals $\int_\Omega w(x)x^k \,\mathrm{d}x$, with $k \in \mathbb{N}_0$. With k going up to $2d-1$, we then assume the existence of d nodes x_m and associated weights w_m such that $M_k = \sum_m x_m^k w_m \forall k \in \{0,\dots,2d-1\}$. Next, we define a polynomial $P_d(x) = \prod_m(x - x_m) = x^d + \sum_m \alpha_m x^m$, where the α_m now have become the unknowns. In order to solve for these unknowns, we use them to define d weighted combinations C_n of the moments M_k as follows: $C_n = M_{d+n} + \sum_m \alpha_m M_{m+n} = \sum_l x_l^n w_l \left(x_l^d + \sum_m \alpha_m x_l^m\right) = 0$ for $n \in \{0,\dots,d-1\}$. This yields the linear system

$$\begin{pmatrix} M_0 & M_1 & \dots & M_{d-1} \\ M_1 & M_2 & \dots & M_d \\ \vdots & \vdots & \ddots & \vdots \\ M_{d-1} & M_d & \dots & M_{2d-2} \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{d-1} \end{pmatrix} = - \begin{pmatrix} M_d \\ M_{d+1} \\ \vdots \\ M_{2d-1} \end{pmatrix}, \tag{8}$$

whose solvability is discussed for instance in [25, Chapter 19]. Upon solving for the α_m, we know $P_d(x)$ and can solve for its d roots x_m. Finally, we can form another linear system using the equations for M_k for $k \in \{0, d-1\}$ to solve for the d weights w_m.

An example using the above procedure is $w(x) = -\ln(x)$, which yields the *logarithmic* quadratures used in the 2D collocation boundary element method [26].

The notion of orthogonal functions can be extended to the multivariate setting. However, the quadrature derivation method based on these does not directly apply to the Gregory setting: there is no common factor that can be extracted from these 20 functions (acting as the weighting function), and clearly there is no bivariate orthogonal *polynomial* basis spanning the same space as $\mathcal{G}$. Similarly, the method based on moments of the weighting function ($w(x) = 1$ in our setting) does not apply as we are in a rational setting (rather than polynomial, allowing the use of moments in the first place). As such, another method must be devised, which we describe in Section 6, to derive quadrature rules for the challenging case of products of (derivatives of) Gregory basis functions. However, before that, we investigate the simpler cases first.

3. Gaussian quadrature for Gregory quads

We now prove that the quadrature for bicubic Bézier patches (see Table 1) is also exact for Gregory patches. In other words, the quadrature of (4) is exact not only for $\mathcal{B}$ but also for $\mathcal{G}$. Naturally, this can be proved directly by checking that all 20 basis functions $G_j(u, v)$ in (6) are integrated exactly by it. However, this offers no insight into the structure of the basis. Therefore, we now provide a more geometric proof.

We first observe that, due to symmetry, it holds

$$B_{12} = G_{12u} + G_{12v}, \quad B_{11} = G_{11v} + G_{11u}, \quad B_{21} = G_{21u} + G_{21v}, \quad B_{22} = G_{22v} + G_{22u}. \tag{9}$$

This, unsurprisingly, shows that $\mathcal{B} \subset \mathcal{G}$. The next step is to find four 'convenient' linearly independent functions D_i, $i \in I$ that extend the basis of $\mathcal{B}$ to a basis of $\mathcal{G}$. By convenient, we mean the following: the four functions D_i should have vanishing integrals over $\square$ based on the four nodes listed in Table 1. As the 8 rational basis functions in (7), again due to symmetry, share the same integral over $\square$ (the actual value is not important but, for the curious reader, it equals $1/32$), we can simply take

$$D_{12} = G_{12u} - G_{12v}, \quad D_{11} = G_{11v} - G_{11u}, \quad D_{21} = G_{21u} - G_{21v}, \quad D_{22} = G_{22v} - G_{22u}. \tag{10}$$

One of these blends, D_{11}, is shown in Figure 3. Note that it is, by construction, anti-symmetric with respect to one of the diagonals of $\square$. It follows that all four D functions are anti-symmetric with respect to one of the diagonals of $\square$. Further, for each of them, the values at the four quadrature nodes are either 0 or pair-wise opposite to each other; this is the case due to the symmetry of the quadrature and anti-symmetry of the D_i. It remains to show linear independence.

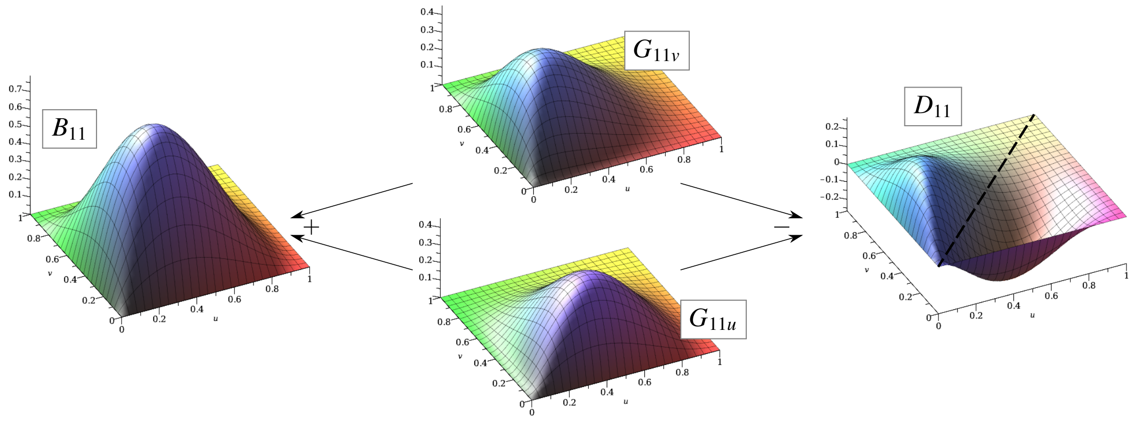

Figure 3: Basis functions G_{11v} (top) and G_{11u} (bottom), their sum B_{11} (left) and their difference D_{11} (right), all scaled up by a factor of 4 for visualization purposes. Note that D_{11} is antisymmetric with respect to the dashed diagonal of $\square$ (along which it vanishes).

Lemma 3.1. *It holds that*

$$\mathcal{G} = \mathcal{B} \oplus \operatorname{span}(D_{12}, D_{11}, D_{21}, D_{22}). \tag{11}$$

Proof. The two equations (9) and (10) can be understood as a change of basis, going from 12 polynomial and 8 rational functions to 16 polynomial and 4 rational functions. It thus suffices to show that the corresponding change of basis matrix $\mathbf{M}$ is regular. Indeed,

$$\mathbf{M} = \left(\begin{array}{cc} \mathbf{I}_{12\times 12} & \mathbf{O}_{12\times 8} \\ \mathbf{O}_{8\times 12} & \begin{array}{cccccccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{array} \end{array} \right),$$

where $\mathbf{I}$ and $\mathbf{O}$ are the unit and zero matrices of suitable sizes, respectively. This leads to $\det(\mathbf{M}) = 16$, which completes the proof. $\square$

All combined, and given the fact that exact quadrature is a linear operator, we have proved the following

Theorem 3.2. *For any function $G(u, v) \in \mathcal{G}$ it holds that*

$$\int_{\square} G(u, v)\ \mathrm{d}u\ \mathrm{d}v = \sum_{i=1}^{4} \omega_i G(u_i, v_i), \tag{12}$$

with the nodes (u_i, v_i) and weights ω_i specified in Table 1. The quadrature is Gaussian.

Furthermore, a symbolic computation in Maple shows that this is the only exact four-node quadrature for $\mathcal{G}$ that exists.

4. Quadrature for partial derivatives

We now turn our attention to the space $\mathcal{G}_u = \operatorname{span}\left(\frac{\partial G_j}{\partial u}, j \in J\right)$, i.e., the space spanned by the 20 partial derivatives of the Gregory quad basis functions with respect to u. Analogous results, due to symmetry, hold for the partials in the v direction.

The 20 functions giving $\mathcal{G}_u$ do not form a basis. That is apparent from Lemma 3.1: $\mathcal{B}_u$ (defined analogously to $\mathcal{G}_u$) does not have dimension 16, but rather only 12 as the space is spanned by tensor-product polynomials of degree 3 in v and degree 2 in u. Indeed, when using the approach taken in the proof of Theorem 2.1, we obtain $\dim(\ker(\mathbf{N}_u)) = 4$ with $\mathbf{N}_u$ now constructed with respect to $\mathcal{G}_u$, which confirms that $\dim \mathcal{G}_u = \dim \mathcal{G}_v = 16$.

The search for quadrature for this space leads to equations of the form

$$\int_{\square} \frac{\partial G_j(u,v)}{\partial u}\, \mathrm{d}u\, \mathrm{d}v = \sum_{i=1}^{m} \omega_i \frac{\partial G_j(u,v)}{\partial u}\Big|_{(u,v)=(u_i,v_i)}, \quad j \in J \tag{13}$$

with ω_i and (u_i, v_i) as unknowns. Due to $\dim \mathcal{G}_u = 16$ it is sufficient to consider only 16 linearly independent equations. As $\mathcal{G}_u$ contains all biquadratic polynomials, one cannot expect the existence of a quadrature with fewer than four nodes. We thus first set $m = 4$. Using the computer algebra system Maple and the Gröbner basis computation it provides (after converting the rational equations in the system to polynomial ones) shows that the system admits no solution. We thus conclude that there is no four-node quadrature rule for $\mathcal{G}_u$.

Next up, we investigate the case when $m = 5$. We first observe that the derivatives in $\mathcal{G}_u$ exhibit, as a set, (only) one-fold symmetry: that with respect to $v = 1/2$. We thus set $v_1 = 1/2$ and set the remaining four nodes as corners of a rectangle symmetric with respect to $v = 1/2$. Using Maple shows again that this leads to no solutions. Although an asymmetric solution is unlikely, we tried to find one both using Maple and numerically using Python (following the approach described in Section 6). The numerical approach did not converge to a solution from any of our random initialisations, and Maple's computation with fully generic nodes and weights ran out of memory (64GB). We thus conjecture that the system admits no solution when $m = 5$.

Finally, we set $m = 6$, which does lead to solutions. Not unexpectedly, Maple is not able to handle the system (13) in full generality or even with the $v = 1/2$ symmetry built in. We therefore turned to our numerical approach again, which delivered multiple solutions. Upon inspection, it led us to the case

$$\begin{aligned} &u_2 = u_1,\ u_3 = u_4 = 1 - u_1,\ u_5 = u_6 = \tfrac{1}{2}; \\ &v_1 = v_3 = v_5 = \tfrac{1}{2} + \tfrac{\sqrt{3}}{6},\ v_2 = v_4 = v_6 = 1 - v_1; \\ &\omega_1 = \omega_2 = \omega_3 = \omega_4,\ \omega_5 = \omega_6 = \tfrac{1}{2} - 2\omega_1. \end{aligned} \tag{14}$$

The last line of equalities follows from symmetry and the fact that the weights have to sum to one. This leaves only u_1 and ω_1 as unknowns. The Gröbner basis of the system, according to Maple, is

$$\begin{gathered} 38123712\omega_1^3 + 1260u_1^2 - 6994016\omega_1^2 - 1260u_1 + 315440\omega_1 - 4887, \\ 152494848\omega_1^4 - 27976064\omega_1^3 + 1261760\omega_1^2 - 20808\omega_1 + 105. \end{gathered}$$

Maple is able to solve this in radical form, but the expressions are far too long to be useful. With the precision of 30 digits, one obtains the following solution:

$$u_1 = 0.0934779568755708578531910506923,\ \omega_1 = 0.126063849132135287691741790291,$$

which yields, when combined with (14), a 6-node symmetric quadrature for $\mathcal{G}_u$.

5. Gaussian Quadrature for Mixed Partial Derivatives

The mixed partial derivatives $\frac{\partial^2 G_j}{\partial u \partial v}$, $j \in J$ of the Gregory quad basis functions exhibit two-fold symmetry when considered as a set. Let $\mathcal{G}_{uv} = \operatorname{span}\left(\frac{\partial^2 G_j}{\partial u \partial v}, j \in J\right)$. Using Lemma 3.1 again, it is clear that the dimension of $\mathcal{G}_{uv}$ is at most $9 + 4 = 13$, where the 9 comes from biquadratic polynomials spanning $\mathcal{B}_{uv}$. Turning again to the proof of Theorem 2.1, one obtains that $\dim(\ker(\mathbf{N}_{uv})) = 7$ with $\mathbf{N}_{uv}$ constructed according to $\mathcal{G}_{uv}$, confirming that $\dim \mathcal{G}_{uv} = 13$.

We approached the problem of quadrature-finding for $\mathcal{G}_{uv}$ as in Section 4. To our surprise, we found a solution in the very first step, using only four nodes. And more surprisingly still, the quadrature rule for mixed derivatives of Gregory quads is the same as the one for Gregory quads; see Table 1. Further, a Gröbner basis computation confirms that this is the only solution with two-fold symmetry and four nodes (although 4 non-symmetric four-node solutions exist; see Figure 4). Since $\mathcal{G}_{uv}$ contains biquadratic polynomials, this quadrature is optimal. We have thus proved the following

Theorem 5.1. *For any function $H(u, v) \in \mathcal{G}_{uv}$ it holds that*

$$\int_{\square} H(u, v)\ \mathrm{d}u\ \mathrm{d}v = \sum_{i=1}^{4} \omega_i H(u_i, v_i), \tag{15}$$

with the nodes (u_i, v_i) and weights ω_i specified in Table 1. The quadrature is Gaussian.

Upon closer inspection, the result is actually not that surprising after all. Lemma 3.1 allows us to focus only on the mixed partial derivatives of the D-functions defined in (10); the other functions are simply biquadratic and thus integrated exactly by the rule. And thus again, the symmetry of the quadrature rule and the anti-symmetry of these functions allows us to arrive at Theorem 5.1.

6. Numerical optimisation using Python

Before moving on to developing/deriving quadratures for products of the Gregory functions or their first-order partial derivatives, we describe a general numerical method that can be used to obtain quadratures for virtually any set of functions on a given domain Ω. This follows the approach of [27] and similar non-linear optimisation schemes.

Given n bivariate functions $f_j(u, v)$, we assume the existence of m quadrature nodes (u_k, v_k) and weights ω_k such that

$$\begin{pmatrix} f_1(u_1, v_1) & f_1(u_2, v_2) & \dots & f_1(u_m, v_m) \\ f_2(u_1, v_1) & f_2(u_2, v_2) & \dots & f_2(u_m, v_m) \\ \vdots & \vdots & \ddots & \vdots \\ f_n(u_1, v_1) & f_n(u_2, v_2) & \dots & f_n(u_m, v_m) \end{pmatrix} \begin{pmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_m \end{pmatrix} = \int_{\square} \begin{pmatrix} f_1(u, v) \\ f_2(u, v) \\ \vdots \\ f_n(u, v) \end{pmatrix} \mathrm{d}u\ \mathrm{d}v. \tag{16}$$

Unlike the two approaches described Section 2.3, the quadrature nodes and weights must now be solved for *simultaneously*. This can be done using a nonlinear least-squares approach, for which we use Python's SciPy library: `scipy.optimize.least_squares` with the trust region reflective (trf) minimisation algorithm and numerical approximations of Jacobians.

Writing the system of equations above as $F\omega = \mathcal{I}$, we define the n residuals $r_j = \sum_k f_j(u_k, v_k)\omega_k - \mathcal{I}_j$. Their partial derivatives with respect to the unknowns u_k, v_k and ω_k required for the optimisation procedure are therefore

$$\frac{\partial r_j}{\partial u_k} = \frac{\partial f_j}{\partial u}(u_k, v_k)\omega_k, \qquad \frac{\partial r_j}{\partial v_k} = \frac{\partial f_j}{\partial v}(u_k, v_k)\omega_k, \qquad \frac{\partial r_j}{\partial \omega_k} = f_j(u_k, v_k). \tag{17}$$

The values of the exact integrals $\mathcal{I}_j$ can be computed using e.g. Python's SymPy library or using Maple. This only has to be done once.

There are multiple challenges in solving for the m triples (u_k, v_k, ω_k). First, there is the choice of m. On the one hand, it is desirable to set this as low as possible (we note that lower bounds on m are only known for specific cases, which means that in other cases they have to be estimated). On the other hand, a *symmetric* rule potentially using a slightly higher number of nodes/weights might be preferred. We get back to this point below.

Secondly, as is usually the case in the context of optimisation, an initial guess is required. For a decent initial guess reasonably close to a solution (note that there might indeed be multiple solutions), we need to have extra insights. Without these, the only resort is a random initial guess, which does not come with any guarantee of convergence.

Remark 1. *While the random guesses may not seem to be the best initialization, it is an option that allows us to quickly explore the solution spaces with the least number of quadrature points. One could eventually follow the approach of [28] and initialise the quadratures, for example, via the pivoted Gram–Schmidt method, starting with a higher degree tensor product Gauss rule and eliminating those quadrature points with the smallest positive weight. However, such a search requires an excessive number of initial quadrature points, and does not guarantee to find a quadrature with the least possible quadrature points (see Table 1 in [28]). In contrast, we impose symmetry assumptions, which greatly reduces the search space, as we detail below.*

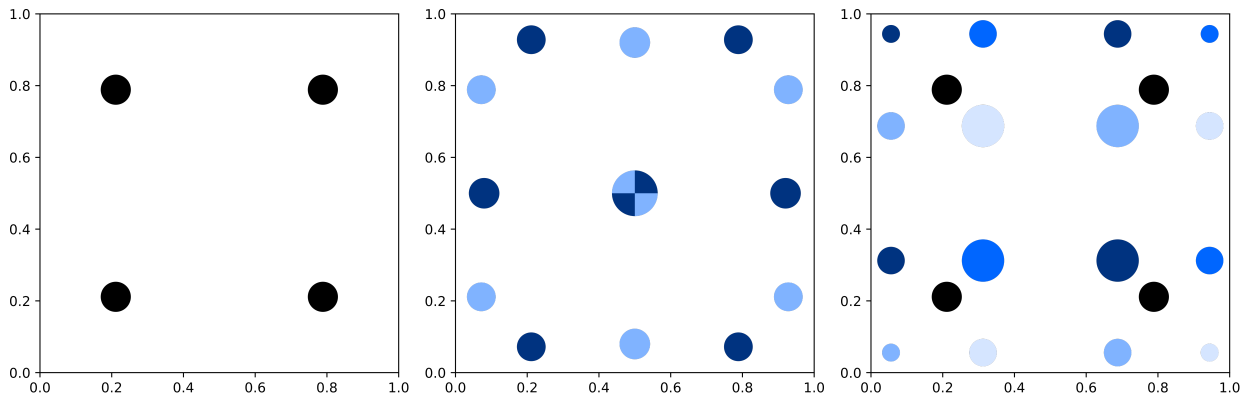

Figure 4: From left to right: Quadrature for G, two quadratures for $\mathcal{G}_u$ and $\mathcal{G}_v$ combined, and five quadratures for $\mathcal{G}_{uv}$. The colours of the nodes correspond to different quadrature rules. In all cases, these are the only quadratures found given the specific number of nodes. The nodes are scaled proportionally to their associated weights.

Finally, there is the matter of accuracy. Although symbolical solutions are arguably the best outcome, numerical solutions that are accurate to `float64` machine precision are certainly an acceptable alternative. After all, it is likely that the obtained quadratures will be used in a numerical setting, where exact integration is not required. The tolerance we use is $\|Fw - \mathcal{I}\|_\infty < 1e - 16$, which we achieve by using Python's NumPy's `longdouble` format for all computations.

In order to test the implementation, we set $f = \mathcal{G}$ and $m = 4$. Unfortunately, without further insight, we initialise the triples (u_k, v_k, ω_k) with random values in $(0, 1)^3$.

Running the script 10.000 times resulted (somewhat surprisingly) in an equal number of successfully converged runs. The 4-node solution, which Maple confirms to be unique, is shown in a single scatter plot in Figure 4 (left).

Next, setting $f = \mathcal{G}_u$, we get converging runs starting from $m = 6$. There appear to be many different solutions. The results of 1000 successfully converged runs are plotted in a single scatter plot (semi-transparently using an alpha/blending value of $\frac{1}{100}$), which we refer to as a *density scatter plot*, shown in Figure 5 (left). One of these solutions was mentioned/discussed in Section 4. We note that not all runs converged satisfactorily; about 25 of them did not. Likewise, for $f = \mathcal{G}_v$ we obtain the density scatter plot shown in Figure 5 (right).

Out of curiosity, we also considered $f = \mathcal{G}_u \cup \mathcal{G}_v$. We obtain converging runs for $m = 7$. Interestingly enough, there appear to be just two solutions integrating all 40 functions; see Figure 4 (middle). Still using random initialisation, in addition to 1000 successfully converged runs, we get about 350 that do not converge satisfactorily (to local minima or saddle points).

Finally, we return to $m = 4$ to check the uniqueness of the quadrature rules for $f = \mathcal{G}_{uv}$. There turn out to be four (mutually symmetric) solutions in addition to the 2×2 Gauss-Legendre rule mentioned and verified in Section 5; see Figure 4 (right). We note that in this case convergence is problematic; only about 1 in every 45 runs converges successfully.

6.1. Exploiting symmetry

Up to this point, no symmetry has been enforced in the optimisation process. However, when moving on to deriving quadratures for products of (derivatives of) functions, as described in the next section, the numbers of functions and quadrature nodes/weights generally increase considerably. In such a setting, restricting the set of feasible solutions to symmetric quadratures can greatly speed up the optimisation process. A slight drawback is that potentially more efficient non-symmetric quadrature rules remain undiscovered. However, this potential drawback is outweighed by the independence/invariance with respect to the orientation of the quadrilateral element and the more efficient storage of symmetric quadratures. Moreover, it is a common assumption that, when dealing with a symmetric parametric domain, the corresponding quadrature is expected to be symmetric as well [28, 29].

The symmetries of the unit square are rather straightforward. For a point *not* on a bimedian or diagonal, we have an 8-fold symmetry. For those that *do* lie on a bimedian or diagonal, there is only 4-fold symmetry (though by using

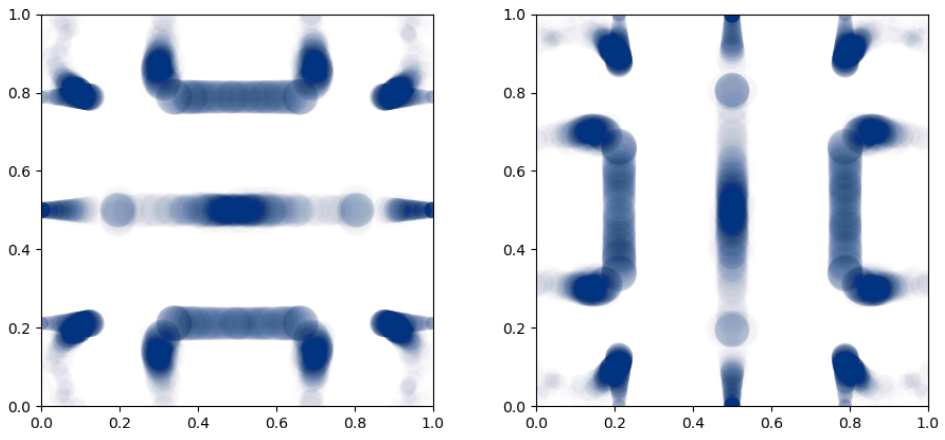

Figure 5: Density scatter plots for 1000 successfully converged runs (see also [14]) for $\mathcal{G}_u$ and $\mathcal{G}_v$. An alpha value of $\frac{1}{100}$ was used.

a pair of points, it could be interpreted as an 8-fold symmetry again).

As such, we now consider the functions

$$\begin{aligned} g_j(u,v) = f_j(u,v) + f_j(1-u,v) + f_j(1-v,u) + f_j(1-v,1-u) + \\ f_j(1-u,1-v) + f_j(u,1-v) + f_j(v,1-u) + f_j(v,u). \end{aligned} \tag{18}$$

For nodes that move to a location on a bimedian or diagonal, this symmetry approach results in a pair of overlapping nodes. Note that this comes with a certain challenge: a tolerance to decide whether nodes indeed overlap or not. Rather than setting up such tolerances, we add the option to explicitly constrain nodes to a bimedian or diagonal. That is, for (u_k, v_k) on a bimedian, we consider the functions

$$g_j^+(u,v) = f_j\left(\frac{1}{2},v\right) + f_j\left(1-v,\frac{1}{2}\right) + f_j\left(\frac{1}{2},1-v\right) + f_j\left(v,\frac{1}{2}\right), \tag{19}$$

whereas for (u_k, v_k) on a diagonal, we consider the functions

$$g_j^\times(u,v) = f_j(u,u) + f_j(1-u,u) + f_j(1-u,1-u) + f_j(u,1-u). \tag{20}$$

Naturally, this introduces a different issue, which is the *a-priori* choice of the numbers of nodes on a bimedian, diagonal or 'ordinary' position. It is very difficult to obtain a good initial guess for a non-linear optimisation problem of this kind. Unlike in the case of spline spaces (and tensor products built upon them), where one can deduce the number of quadrature points lying in each particular knot span, see e.g. [30], in our case the support of all the involved functions is the whole unit square. Another common initialisation strategy, namely a uniform distribution of quadrature points and equal weights, which is simple for a 1D domain, is impossible over the unit square for a given number of nodes such as $m = 7$ not forming a perfect square. Therefore, we opted for random initial guesses, which turned out to be a good choice in our setting.

Next, an important observation is that there might be certain sets of basis functions satisfying the same symmetries. That is, we could have e.g. $f_p(u,v) = f_q(1-u,v)$, which would then allow us to remove either f_p or f_q from the system of equations. Although this does not influence the space of feasible solutions, it can considerably speed up the optimisation. We have implemented this function elimination procedure symbolically in Maple (using parameter substitutions such as $u \leftarrow (1-u)$ and detecting scaled versions by division) to reduce the number of functions involved in the optimisations to the *relevant* functions forming a set with no symmetries and scaled copies left.

The resulting system of equations (here, as an example, using a single point on a bimedian and another on a diagonal, s as the number of relevant functions and t as the number of symmetric quadrature points) is now as follows:

$$\begin{pmatrix} g_1^+(\frac{1}{2},v_1) & g_1^\times(u_2,u_2) & g_1(u_3,v_3) & \dots & g_1(u_t,v_t) \\ \vdots & \vdots & \vdots & & \vdots \\ g_s^+(\frac{1}{2},v_1) & g_s^\times(u_2,u_2) & g_s(u_3,v_3) & \dots & g_s(u_t,v_t) \end{pmatrix} \begin{pmatrix} \omega_1 \\ \vdots \\ \omega_t \end{pmatrix} = \mathcal{I}. \tag{21}$$

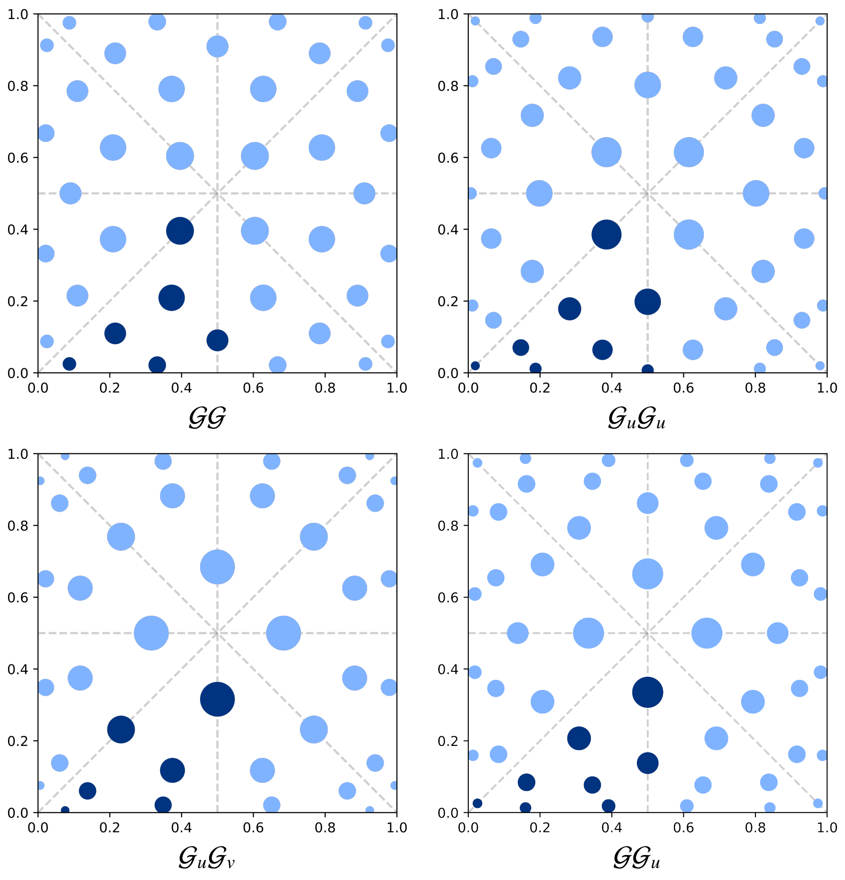

Figure 6: Quadratures for the spaces $\mathcal{G}\mathcal{G}$ (we conjecture this one to be unique), $\mathcal{G}_u\mathcal{G}_u$, $\mathcal{G}_u\mathcal{G}_v$, and $\mathcal{G}\mathcal{G}_u$. The highlighted nodes are listed in Tables 3–6, respectively. The nodes are again scaled proportionally to their associated weights.

Without bimedial or diagonal nodes, the partial derivatives of the residuals $r_j = \sum_k g_j(u_k, v_k)\omega_k - \mathcal{I}_j$ now correspond to

$$\frac{\partial r_j}{\partial u_k} = \left(\frac{\partial f_j}{\partial u}(u_k, v_k) - \frac{\partial f_j}{\partial u}(1-u_k, v_k) + \frac{\partial f_j}{\partial v}(1-v_k, u_k) - \frac{\partial f_j}{\partial v}(1-v_k, 1-u_k) - \right.$$
$$\left. \frac{\partial f_j}{\partial u}(1-u_k, 1-v_k) + \frac{\partial f_j}{\partial u}(u_k, 1-v_k) - \frac{\partial f_j}{\partial v}(v_k, 1-u_k) + \frac{\partial f_j}{\partial v}(v_k, u_k)\right)\omega_k. \quad (22)$$

Note that due to the chain rule, half of the terms come with a negative sign. In addition, half of the entries are partial derivatives of f_j with respect to v rather than u. The partial derivatives with respect to v_k are obtained similarly. As for the partials w.r.t. ω_k, these are just the functions g_j (similar to the ordinary setting; see (17)).

The partial derivatives of the residuals involving g_j^+ or $g_j^\times$ are slightly different. For bimedial nodes, only the partial derivative w.r.t. v_k matters (and consists of four terms rather than eight). As for diagonal nodes, only the partial derivative w.r.t. u_k matters (and again consists of four terms, which are generally more complex in this setting as it involves the product rule).

Using the above approach to derive quadratures for $\mathcal{G}$, while restraining the (now single) quadrature point to a diagonal, we indeed obtain the same result as obtained in Section 3 (that is, the 2×2 Gauss-Legendre rule). In addition, note that there are merely three (rather than twenty) *relevant* functions in the symmetric setting; see Figure 2. The function spaces $\mathcal{G}_u$, $\mathcal{G}_v$, and $\mathcal{G}_{uv}$, already addressed in Sections 4 and 5, were not considered in this symmetric context.

Table 2: An overview for the spaces $\mathcal{GG}$, $\mathcal{G}_u\mathcal{G}_u$, $\mathcal{G}_u\mathcal{G}_v$, and $\mathcal{GG}_u$. The second column lists the number of *relevant* functions in each space (thus up to symmetries and scaling), which is followed by the counts of nodes on bimedians, diagonals, and general nodes over □. The final column lists the total number of nodes in the quadratures.

	Relevant products	Bimedial nodes	Diagonal nodes	Ordinary nodes	Total nodes
$\mathcal{GG}$	29	1	1	4	$1\cdot4+1\cdot4+4\cdot8=40$
$\mathcal{G}_u\mathcal{G}_u$	56	2	2	4	$2\cdot4+2\cdot4+4\cdot8=48$
$\mathcal{G}_u\mathcal{G}_v$	51	1	1	4	$1\cdot4+1\cdot4+4\cdot8=40$
$\mathcal{GG}_u$	87	2	1	5	$2\cdot4+1\cdot4+5\cdot8=52$

Table 3: A 40-node quadrature for the space $\mathcal{GG}$. Only the 6 dark nodes as shown in Figure 6, top left, are listed; the remaining nodes are obtained by symmetry.

i	u_i	v_i	ω_i
1^+	0.5000 0000 0000 0000	0.0905 6973 0284 8170	0.0262 0952 9693 9717
$2^\times$	0.3956 8294 1869 1628	0.3956 8294 1869 1628	0.0422 4139 6527 7472
3	0.3322 2167 8051 4396	0.0214 9440 2913 3952	0.0169 0274 6523 0801
4	0.0874 7945 9811 0069	0.0248 3028 9629 3109	0.0100 4208 4122 3864
5	0.3723 3639 6980 4577	0.2090 9808 5379 9072	0.0380 6215 8608 4612
6	0.2151 5319 7700 8579	0.1098 3319 4339 1384	0.0257 6754 7635 2129

7. Integrating products and products of derivatives

In this section, we describe our main numerical contribution of the paper: symmetric quadrature rules for the product of $\mathcal{G}$ with itself, $\mathcal{G}_u = \frac{\partial \mathcal{G}}{\partial u}$ with itself, $\mathcal{G}_u\mathcal{G}_v = \frac{\partial \mathcal{G}}{\partial u}\frac{\partial \mathcal{G}}{\partial v}$, as well as the mixed product space $\mathcal{GG}_u$.

The first step is to combine the product of functions with the notion of symmetry as described in the previous section. For two functions $f_p(u,v)$ and $f_q(u,v)$, we define

$$\begin{aligned} h_{pq}(u,v) = {} & f_p(u,v)f_q(u,v) + f_p(1-u,v)f_q(1-u,v) + f_p(1-v,u)f_q(1-v,u) + f_p(1-v,1-u)f_q(1-v,1-u) + \\ & f_p(1-u,1-v)f_q(1-u,1-v) + f_p(u,1-v)f_q(u,1-v) + f_p(v,1-u)f_q(v,1-u) + f_p(v,u)f_q(v,u). \quad (23) \end{aligned}$$

Note that $h_{pq}(u,v) \neq g_p(u,v)g_q(u,v)$, as the $g_j(u,v)$ are sums of functions rather than factors (in which case we would have had $h_{pq} = g_p \cdot g_q$). The functions $h^+_{pq}(u,v)$ and $h^\times_{pq}(u,v)$ are defined analogously, but with symmetries with respect to the bimedians and the diagonals of □, respectively.

As for symmetries of products, note that to start with, we have $\frac{n(n-1)}{2}$ pairs of functions that are pairwise identical. In other words, and focusing on our setting, for $n = 20$ we obtain a 20×20 matrix of products, for which all 190 entries below (or above) the diagonal can be ignored/removed.

In addition, following the same symmetry detection approach as described above, we can remove scaled and/or symmetrical versions of functions. In the case of $\mathcal{GG}$, we are left with merely 29 of the 400 products. For the other two cases, the number of relevant products turns out to be 56 and 51, respectively; see the second column of Table 2.

The system of equations has the same structure as (21), which we now express using the shorthand notation $Hw = \mathcal{J}$, where $\mathcal{J}$ contains the exact integrals of the *relevant* products of functions (computed symbolically using Maple). The partial derivatives of the residuals are more involved in this setting, as every term now triggers the product rule (and those of $h^\times_{pq}$ even a *quadruple* product rule). As the resulting expressions do not have much added value, they are omitted from the paper.

With everything now in place, we resort to educated guesses for the number of bimedial, diagonal, and ordinary nodes which we denote as triples (b,d,o), and initialise them randomly.

For $\mathcal{GG}$, we have found rules satisfying the same tolerance as before, i.e. $\|Hw - \mathcal{J}\|_\infty < 1e-16$, starting from $(b,d,o) = (1,1,4)$. Moreover, the rule using this nodal count appears to be unique.

Table 4: A 48-node quadrature for the space $\mathcal{G}_u\mathcal{G}_u$. Only the 8 dark nodes as shown in Figure 6, top right, are listed; the remaining nodes are obtained by symmetry.

i	u_i	v_i	ω_i
1^+	0.5000 0000 0000 0000	0.1980 2558 3676 8669	0.0381 8500 8024 3771
2^+	0.5000 0000 0000 0000	0.0069 2954 8015 9801	0.0080 4075 9349 4086
$3^\times$	0.3852 8526 4554 8323	0.3852 8526 4554 8323	0.0492 3319 5133 8609
$4^\times$	0.0196 5851 0592 8419	0.0196 5851 0592 8419	0.0044 0136 6886 6572
5	0.3740 2402 1950 8236	0.0641 0217 6265 1844	0.0227 1943 2876 4887
6	0.1875 5392 3242 2541	0.0114 5761 5415 7695	0.0079 1293 7497 2532
7	0.2825 6546 7480 4259	0.1783 7707 0249 5502	0.0292 2893 5498 3503
8	0.1464 5385 7917 5176	0.0704 1415 0570 9511	0.0152 0852 9430 7559

Table 5: A 40-node quadrature for the space $\mathcal{G}_u\mathcal{G}_v$. Only the 6 dark nodes as shown in Figure 6, bottom left, are listed; the remaining nodes are obtained by symmetry.

i	u_i	v_i	ω_i
1^+	0.5000 0000 0000 0000	0.3157 2647 0938 0255	0.0661 8307 2480 3311
$2^\times$	0.2313 5808 6087 5473	0.2313 5808 6087 5473	0.0421 6084 1052 3185
3	0.3747 0407 1613 1695	0.1176 0911 5179 6072	0.0341 0492 8716 6164
4	0.0756 2648 4653 5047	0.0059 7387 7968 0662	0.0039 9869 6548 3779
5	0.1381 5104 9787 4147	0.0604 6459 5332 7444	0.0165 8001 9777 0600
6	0.3486 9754 3316 0818	0.0209 4942 1639 7607	0.0161 4439 8191 6209

As for the cases of $\mathcal{G}_u\mathcal{G}_u$ and $\mathcal{G}_u\mathcal{G}_v$, we obtained results starting from $(2, 2, 4)$ and $(1, 1, 4)$, respectively. In both cases there appear to be a small number of different rules. For the mixed space $\mathcal{G}\mathcal{G}_u$ we obtained several results with at least 5 regular nodes with additional bimedial and diagonal ones, such as with the $(2, 1, 5)$ node configuration. We selected the ones which appear the most uniform, i.e. have a fair distribution of nodes with gradually changing associated weights.

The results are summarised in Table 2 and Figure 6. Individual quadratures are listed in Tables 3–6. Note that all nodes are safely away from the corners of $\square$, thus avoiding any potential numerical instabilities in the quadratures when evaluating the functions near the singularities in the four corners of the domain.

8. Numerical tests

We perform several preliminary numerical tests on how the proposed quadrature rules perform on functions outside the considered linear spaces. To this end, we define the following three test functions

$$F_1 = u(u+7)^2(u-1)^4v^3(v+4)(v+1)^3, \tag{24}$$

$$F_2 = \sin(u)\cos(v), \tag{25}$$

$$F_3 = \ln(u^2+v^2+1), \tag{26}$$

which are shown over the unit square domain in Figure 7. Their exact integrals and the errors induced by a selection of our rules are summarised in Table 7.

To gain insight into the behaviour of the quadrature errors under refinement of the domain, as is often the case in adaptive (isogeometric) methods, we proceed as follows. We recursively (and uniformly) split the original domain $\square$ into four smaller squares, and apply our quadrature rules, appropriately shifted and scaled, to the sub-squares and sum all contributions, and compute the error with respect to the exact integral at each refinement step.

Table 6: A 52-node quadrature for the space $\mathcal{G}\mathcal{G}_u$. Only the 8 dark nodes as shown in Figure 6, bottom right, are listed; the remaining nodes are obtained by symmetry.

i	u_i	v_i	ω_i
1^+	0.5000 0000 0000 0000	0.1378 8683 8924 4187	0.0261 1995 4444 1711
2^+	0.5000 0000 0000 0000	0.3351 0974 8939 8083	0.0532 8115 2206 7506
3^*	0.0256 6732 7768 5628	0.0256 6732 7768 5628	0.0052 0634 0490 9486
4	0.1624 1893 7090 3416	0.0843 2729 4874 7703	0.0171 9715 6125 6456
5	0.3458 7872 6304 7543	0.0769 2552 6392 5755	0.0166 6548 4597 7532
6	0.3087 4726 6512 8944	0.2068 5052 4243 3376	0.0312 2475 6754 2992
7	0.3909 8690 0739 4927	0.0183 4638 3648 7645	0.0103 2974 5659 9723
8	0.1594 9740 7996 5401	0.0129 8539 8490 4721	0.0072 7913 3291 3946

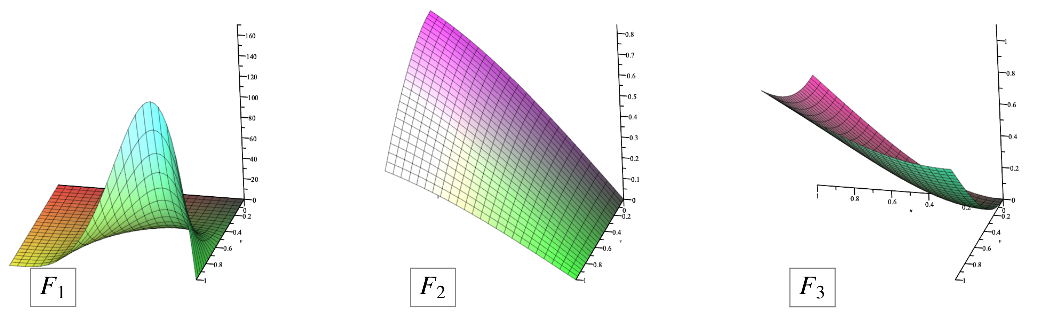

Figure 7: Our test functions. From left to right: F_1, F_2, and F_3, all plotted over the unit square domain $\square$.

Figure 8 shows, on a logarithmic scale, the result of this test using F_1 and F_2 across several of our rules. The results for F_3 are similar to those for F_2 and thus not reported. As expected, the absolute error goes down with every refinement step, at least until we hit the limits of our numerical precision, after which we do not see any improvement. Note that the $\mathcal{G}\mathcal{G}$ rule integrates F_1, a bi-degree 7 function, numerically exactly.

For instance in the case of $\mathcal{G}_u\mathcal{G}_v$, the error gets approximately $4^3 = 64$ times smaller with each refinement step, and for $\mathcal{G}$ and $\mathcal{G}_u$ this ratio is approximately $4^2 = 16$ for both F_1 and F_2. The corresponding exact rates 4^r are marked by r in the figure next to the dotted triangles indicating these rates.

It is worth noting that since the rules for the spaces $\mathcal{G}$ and $\mathcal{G}_{uv}$ are the standard 2×2 Gauss-Legendre rules, the known error estimates apply to them. The error analysis of our other rules when applied to functions outside our function spaces goes beyond the scope of the current paper. One could mimic the approach of [31, Theorem 5.2-3] and construct the corresponding uni- and bivariate kernels. However, the analysis in [31] assumes the exactness of the rule for the space of polynomials, while our spaces are rational. Another issues is that some of our rules were computed numerically, so up to some error, and since the quadrature points and weights appear in the estimates, the inaccuracy of the points/weights would affect also the estimates.

9. Conclusion and future work

We have explored quadrature rules for the space $\mathcal{G}$ spanned by the 20 basis functions of Gregory quads, but also for the derived spaces $\mathcal{G}_u$ and $\mathcal{G}_{uv}$. We have found symbolic quadratures for these three spaces. In the cases of $\mathcal{G}$ and $\mathcal{G}_{uv}$, the standard 2×2 Gauss-Legendre quadrature is exact and Gaussian. We have provided a 6-node quadrature for $\mathcal{G}_u$ and conjectured that a 5-node quadrature for this space does not exist.

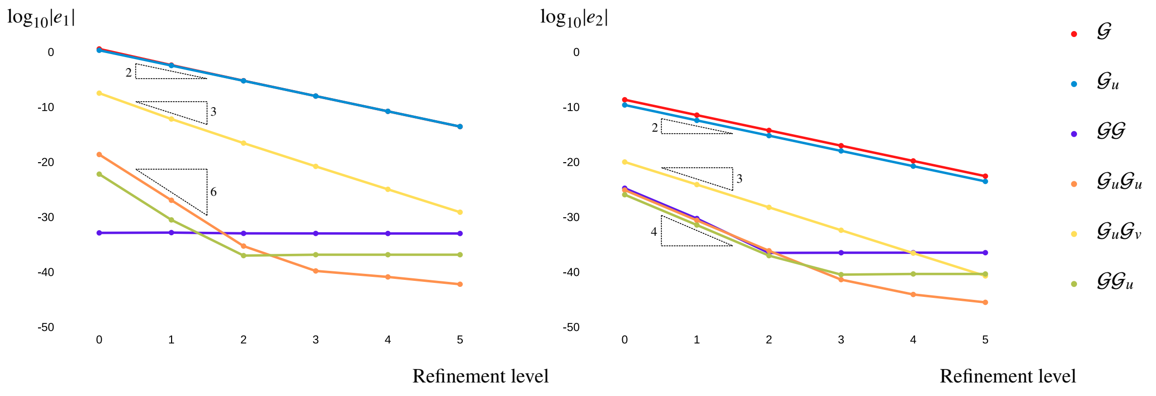

Figure 8: Plots of the logarithm of the absolute value of the error/difference e_i between the exact integral $I_i = \int_\square F_i \, du \, dv$ of F_1 (left) and F_2 (right) over the unit square and the progressively finer approximation obtained by applying our quadrature rules over the recursively refined unit square. Due to numerical precision limitations, the error stops decreasing after a certain number of refinement steps. Each dotted triangle, marked with its associated r, indicates the slope corresponding to error improvement rate of 4^r.

We have also investigated quadratures for the product spaces $\mathcal{GG}$, $\mathcal{G}_u\mathcal{G}_u$, $\mathcal{G}_u\mathcal{G}_v$, as well as $\mathcal{GG}_v$. Using numerical optimisation and heavily relying on symmetry, we have found numerical quadratures for these three spaces.

Our investigation is, however, not complete; it provides several directions for future investigation. One concerns finding optimal quadrature rules in the case of the product spaces: the ones we have found are likely not Gaussian. Another direction is quadratures for higher-order derivatives, which may arise in higher-order problems. The second-order derivatives of the 20 Gregory functions are square integrable, which opens up this possibility. It would also be interesting to apply these new quadrature rules to solve actual problems using Gregory patches, in the vein of [17, 18, 19]. Further, on top of our initial tests presented in Section 8, it would be useful to investigate the errors that our quadratures introduce when applied to functions outside the spaces they were designed for. Finally, focusing on the case of triangular Gregory patches is a natural continuation of our investigations into quadratures for Gregory patches.

Acknowledgements

The first author thanks the Chinese Scholarship Council for financial support. This research was in part supported by the Swiss National Science Foundation (SNSF) under grant 200021_192041.

References

[1] F. B. Hildebrand, Introduction to numerical analysis, Courier Corporation, 1987.
[2] J. A. Cottrell, T. J. Hughes, Y. Bazilevs, Isogeometric analysis: toward integration of CAD and FEA, John Wiley & Sons, 2009.

Table 7: The integrals $I_i = \int_\square F_i \, du \, dv$, $i \in \{1, 2, 3\}$ of the test functions are $I_1 = 12.79$, $I_2 = 0.3868$, and $I_3 = 0.4790$, reported with four significant digits. The corresponding errors $e_i = I_i - \sum_{j=1}^{m} \omega_j F_i(u_j, v_j)$ introduced by the approximations computed by the rules presented in this paper are listed.

	e_1	e_2	e_3
$\mathcal{G}$	-1.897	0.0001849	0.00009486
$\mathcal{G}_u$	1.483	0.00007112	0.00001085
$\mathcal{GG}$	$-5.692 \cdot 10^{-15}$	$1.903 \cdot 10^{-11}$	$9.038 \cdot 10^{-8}$
$\mathcal{G}_u\mathcal{G}_u$	$-8.785 \cdot 10^{-9}$	$-1.370 \cdot 10^{-11}$	$-6.368 \cdot 10^{-8}$
$\mathcal{G}_u\mathcal{G}_v$	0.0006126	$-2.258 \cdot 10^{-9}$	$-1.820 \cdot 10^{-7}$
$\mathcal{GG}_u$	$-2.467 \cdot 10^{-10}$	$-5.857 \cdot 10^{-12}$	$-2.627 \cdot 10^{-8}$

[3] M. Bartoň, V. M. Calo, Gaussian quadrature for splines via homotopy continuation: rules for C^2 cubic splines, Journal of Computational and Applied Mathematics 296 (2016) 709–723.

[4] M. Bartoň, V. M. Calo, Optimal quadrature rules for odd-degree spline spaces and their application to tensor-product-based isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 305 (2016) 217–240.

[5] M. Bartoň, R. Ait-Haddou, V. M. Calo, Gaussian quadrature rules for C^1 quintic splines with uniform knot vectors, Journal of Computational and Applied Mathematics 322 (2017) 57–70.

[6] M. Bartoň, V. M. Calo, Gauss–Galerkin quadrature rules for quadratic and cubic spline spaces and their application to isogeometric analysis, Computer-Aided Design 82 (2017) 57–67.

[7] T. J. Hughes, A. Reali, G. Sangalli, Efficient quadrature for NURBS-based isogeometric analysis, Computer methods in applied mechanics and engineering 199 (5-8) (2010) 301–313.

[8] F. Auricchio, F. Calabro, T. J. Hughes, A. Reali, G. Sangalli, A simple algorithm for obtaining nearly optimal quadrature rules for NURBS-based isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 249 (2012) 15–27.

[9] F. Calabro, G. Sangalli, M. Tani, Fast formation of isogeometric Galerkin matrices by weighted quadrature, Computer Methods in Applied Mechanics and Engineering 316 (2017) 606–622.

[10] R. R. Hiemstra, F. Calabro, D. Schillinger, T. J. Hughes, Optimal and reduced quadrature rules for tensor product and hierarchically refined splines in isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 316 (2017) 966–1004.

[11] K. A. Johannessen, Optimal quadrature for univariate and tensor product splines, Computer Methods in Applied Mechanics and Engineering 316 (2017) 84–99.

[12] G. Sangalli, M. Tani, Matrix-free weighted quadrature for a computationally efficient isogeometric k-method, Computer Methods in Applied Mechanics and Engineering 338 (2018) 117–133.

[13] P. J. Barendrecht, M. Bartoň, J. Kosinka, Efficient quadrature rules for subdivision surfaces in isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 340 (2018) 1–23.

[14] P. Barendrecht, Splines for engineers: with selected applications in numerical methods and computer graphics, Ph.D. thesis, University of Groningen (2019).

[15] J. Kosinka, M. Bartoň, Gaussian quadrature for C^1 cubic Clough–Tocher macro-triangles, Journal of Computational and Applied Mathematics 351 (2019) 6–13.

[16] M. Bartoň, J. Kosinka, On numerical quadrature for C^1 quadratic Powell–Sabin 6-split macro-triangles, Journal of Computational and Applied Mathematics 349 (2019) 239–250.

[17] L. Greco, M. Cuomo, An implicit strong G^1-conforming formulation for the analysis of the Kirchhoff plate model, Continuum Mech. Thermodyn. 32 (2020) 621–645.

[18] L. Greco, M. Cuomo, An implicit G^1-conforming bi-cubic interpolation for the analysis of smooth and folded Kirchhoff-Love shell assemblies, Computer Methods in Applied Mechanics and Engineering 373 (2021) 113476.

[19] L. Greco, M. Cuomo, L. Contrafatto, A quadrilateral G^1-conforming finite element for the Kirchhoff plate model, Computer Methods in Applied Mechanics and Engineering 346 (2019) 913–951.

[20] C. Loop, S. Schaefer, T. Ni, I. Castano, Approximating subdivision surfaces with Gregory patches for hardware tessellation, in: ACM SIGGRAPH Asia 2009 papers, 2009, pp. 1–9.

[21] J. A. Gregory, Smooth interpolation without twist constraints, in: Computer aided geometric design, Elsevier, 1974, pp. 71–87.

[22] G. J. Hettinga, J. Kosinka, Multisided generalisations of Gregory patches, Computer Aided Geometric Design 62 (2018) 166–180.

[23] K. Ueda, A method for removing the singularities from Gregory surfaces, in: T. Lyche, L. L. Schumaker (Eds.), Mathematical Methods in Computer Aided Geometric Design II, Academic Press, 1992, pp. 597–606.

[24] H. Chiyokura, T. Takamura, K. Konno, T. Harada, G^1 surface interpolation over irregular meshes with rational curves, NURBS for Curve and Surface Design (1991) 15–34.

[25] R. Hamming, Numerical Methods for Scientists and Engineers, Dover Publications Inc., 1962.

[26] J. H. Kane, Boundary Element Analysis in Engineering Continuum Mechanics, Prentice Hall, 1994.

[27] J. Bremer, Z. Gimbutas, V. Rokhlin, A nonlinear optimization procedure for generalized Gaussian quadratures, SIAM Journal on Scientific Computing 32 (4) (2010) 1761–1788.

[28] H. Xiao, Z. Gimbutas, A numerical algorithm for the construction of efficient quadrature rules in two and higher dimensions, Computers & mathematics with applications 59 (2) (2010) 663–676.

[29] S. Mousavi, H. Xiao, N. Sukumar, Generalized gaussian quadrature rules on arbitrary polygons, International Journal for Numerical Methods in Engineering 82 (1) (2010) 99–113.

[30] G. Nikolov, On certain definite quadrature formulae, Journal of computational and applied mathematics 75 (2) (1996) 329–343.

[31] A. H. Stroud, Approximate calculation of multiple integrals, Prentice-Hall, 1971.