Cubature rules with positive weights and internal nodes on spline curvilinear domains

A. Sommariva [a], M. Vianello [a]

[a] *Dept. of Mathematics, University of Padova*
via Trieste 63, 35121 Padova (Italy)

Abstract

In this paper we introduce an algorithm that computes an algebraic cubature rule of degree n with at most $(n+1)(n+2)/2$ internal nodes and positive weights, over spline curvilinear regions $\mathcal{S}$. The key ingredients are a theorem by Davis on Tchakaloff sets in compact domains and a new *in-domain* algorithm in such $\mathcal{S}$. Numerical Matlab codes are provided.

Key words: Algebraic cubature, splines, curvilinear polygons, moment computation, compression, monnegative-least squares, point containment, winding number.

2010 AMS subject classification: Primary 41A10, 65D32; Secondary: 65D10.

1 Introduction

During the last years, there has been an increasing interest on algebraic cubature rules over polygonal and curvilinear domains.

In particular, in Virtual Elements methods, there has been requests of formulae on regions that have non standard shapes, but whose boundaries can be typically well approximated by parametric splines (see e.g. [3] and references therein). In these instances, the algebraic degree of precision, term often shortened by the acronym ADE, is usually not too high, but rules of PI-type,

⋆ Work partially supported by the DOR funds and the biennial project BIRD163015 of the University of Padova, and by the GNCS-INdAM.

Email addresses: alvise@math.unipd.it (A. Sommariva), marcov@math.unipd.it (M. Vianello).

i.e. with nodes in the domain and positive weights, are in many senses more appealing.

In the past we wrote several contributions on these topics. In [14], [15], [11] we discussed cubature rules respectively over simple polygons, spline curvilinear and simple domains whose boundary can be described by parametric functions, also providing Matlab routines.

In all these works, the rules were of PI-type in convex domains but this property was not always guaranteed in non convex regions.

To fulfil these requests, we have recently introduced in [2] an algorithm that determines formula of PI-type and prescribed degree of precision, even on polygons with complicated shapes, that can be non convex, not simple and not simply connected domains. In particular, if the cardinality of the rules is an issue, we were able to provide rules of PI-type, ADE equal to n and a number of nodes at most $N_n = (n+1)(n+2)/2$ by means of a numerical implementation of Tchakaloff theorem (see [18], [19], [16]). The relative Matlab codes make use of the new `polyshape` environment and can be retrieved at [13].

The spline curvilinear counterpart is more delicate and we propose here an algorithm that provides rules of PI-type not only over polygons but also on *simple* spline curvilinear domains.

Here we propose an algorithm that first, by means of a new in-domain routine for spline curviliear polygons, determines a *sufficiently dense* set of bivariate points $\{\xi_i\}$ belonging to the spline curvilinear domain $\mathcal{S}$ and then it evaluates the Vandermonde-like matrix $V = \{V_n(X)\}_{i,j} = \phi_j(\xi_i)$, w.r.t. a certain polynomial basis $\{\phi_i\}$ of total degree δ over $\mathcal{S}$. Next, it computes the moments $\gamma_j = \int_{\mathcal{S}} \phi_j d\mathcal{S}$, and finally uses the numerical implementation of Tchakaloff theorem, computing a sparse solution of the system $V \cdot w = \gamma$ where the vector of the weights w has at most $(n+1)(n+2)/2$ non null components. The existence of such solution relies on a result on Tchakaloff theorem, proved in [20].

The paper is organized as follows. In Section 2, we introduce our *in-domain* routine for spline curvilinear domains $\mathcal{S}$. In Section 3 we briefly recall some main results about Tchakaloff theorem and its application In Section 4 we explain our new method for determining cubature rules of PI-type in $\mathcal{S}$, pointing out how to compute the required moments γ_k by means of Gauss-Green theorem as well as reporting the fundamental results of [20], describing in detail the proposed algorithm and some approximation properties. Finally in Section 5 we show the numerical results on two non convex domains, regarding either the cubature either the *in-domain* procedure.

2 In domain algorithm for spline curvilinear regions

Let $\mathcal{S} \subset \mathbb{R}^2$ be a simply connected domain and suppose that its boundary $\partial\mathcal{S}$

(1) can be described by parametric equations $x = \tilde{x}(t)$, $y = \tilde{y}(t)$, $t \in [a, b]$, $\tilde{x}, \tilde{y} \in C([a, b])$ such that $\tilde{x}(a) = \tilde{x}(b)$ and $\tilde{y}(a) = \tilde{y}(b)$;
(2) is not self-intersecting (i.e. $\mathcal{S}$ is *simple*);
(3) there is a partition $\{I^{(k)}\}_{k=1,\dots,M}$ of $[a, b]$, and partitions $\{I_j^{(k)}\}_{j=1,\dots,m_k}$ of each $I^{(k)}$, such that the restrictions of $\tilde{x}$, $\tilde{y}$ on each $I^{(k)}$ are splines of degree δ_k, w.r.t. the subintervals $\{I_j^{(k)}\}_{j=1,\dots,m_k}$.

Typical instances are *approximations* of more general and piecewise regular, not self-intersecting, simply connected domains Ω by means of spline curvilinear regions $\mathcal{S}$. To this purpose, given the *vertices* $V_k \in \partial\Omega$, $k = 1, \dots, M + 1$, then $\partial\Omega \approx \partial\mathcal{S} := \cup_{k=1}^{M} V_k \frown V_{k+1}$ where each curvilinear side $V_k \frown V_{k+1}$ is tracked by a spline curve of degree δ_k, interpolating an ordered subsequence of control points $P_{1,k} = V_k, P_{2,k}, \dots, P_{m_k-1,k}, P_{m_k,k} = V_{k+1}$ with a suitable parametrization that determines each $I_j^{(k)}$ (and thus each $I^{(k)}$).

In this section we introduce an algorithm that establishes if a point $P \in \mathbb{R}^2$ is inside, outside or on the boundary of $\partial\mathcal{S}$.

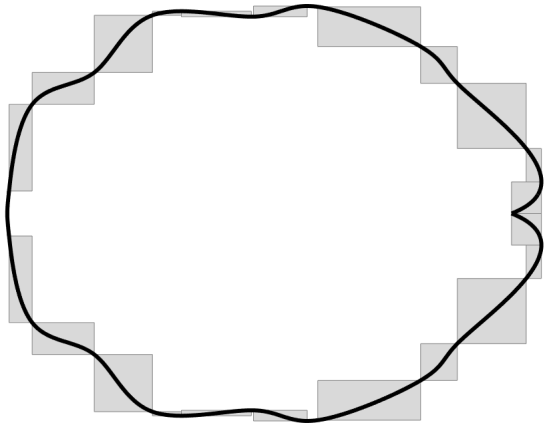

Fig. 1. A domain $\mathcal{S}$ of (spline) curvilinear type and its *monotone boxes*.

We start computing a domain that consists of union of rectangles and contains $\partial\mathcal{S}$. To this purpose, let $I_j^{(k)} = [t_j^{(k)}, t_{j+1}^{(k)}]$ the partitions introduce above of the parametrization interval $[a, b]$. Furthermore, if $\tilde{x}'$ changes sign in $(t_j^{(k)}, t_{j+1}^{(k)})$, then define as $\mathcal{N}_j^{(k)} = \{t_i^{(j,k)}\}_{i=1,\dots,l_{j,k}}$ the set of all the points $t_i^{(j,k)} \in (t_j^{(k)}, t_{j+1}^{(k)})$ such that $\tilde{x}'(t_i^{(j,k)}) = 0$ (observe that the restriction to $I^{(j,k)}$ is a polynomial of degree δ_k, hence its derivative exists), otherwise put $\mathcal{N}_j^{(k)} = \emptyset$. Let $T^{(j,k)} = \{t_j^{(k)}, t_{j+1}^{(k)}\} \cup \mathcal{N}_j^{(k)}$, where we suppose that its elements $T_i^{(j,k)}$ are in increasing

order. Next introduce the rectangles $\mathcal{B}_i^{(j,k)}$, that we will call *monotone boxes*,

$$\mathcal{B}_i^{(j,k)} := [\min_{t\in\{T_i^{(j,k)}\cup T_{i+1}^{(j,k)}]\}} \tilde{x}(t), \max_{t\in\{T_i^{(j,k)}\cup T_{i+1}^{(j,k)}]\}} \tilde{x}(t)]\times[\min_{t\in[T_i^{(j,k)},T_{i+1}^{(j,k)}]} \tilde{y}(t), \max_{t\in[T_i^{(j,k)},T_{i+1}^{(j,k)}]} \tilde{y}(t)].$$

Observe that by definition if $\mathcal{N}_j^{(k)} = \emptyset$, then there is only the monotone box $\mathcal{B}_1^{(j,k)}$.

Furthermore, since $\tilde{y}$ restricted to $[T_i^{(j,k)}, T_{i+1}^{(j,k)}]$ is a polynomial of degree δ_k, the evaluation of

$$\min_{t\in[T_i^{(j,k)},T_{i+1}^{(j,k)}]} \tilde{y}(t), \quad \max_{t\in[T_i^{(j,k)},T_{i+1}^{(j,k)}]} \tilde{y}(t)$$

can be easily determined once the derivative $\tilde{y}'$ is at hand, by computing its zeros in $[T_i^{(j,k)}, T_{i+1}^{(j,k)}]$ and the evaluation of $\tilde{y}$ at $T_i^{(j,k)}$ and $T_{i+1}^{(j,k)}$.

By construction, the restriction of $\tilde{x}$ to each monotone box $\mathcal{B}_i^{(j,k)}$ is a monotone function.

Once the set $\mathcal{B} := \{\mathcal{B}_i^{(j,k)}\}$ is at hand, we apply the crossing theorem to determine if $P = (P_x, P_y)$ is in the domain $\mathcal{S}$. We consider a point $Q = (Q_x, Q_y) \notin \mathcal{S}$ with $Q_y < P_y$, and see how many times the segment QP crosses $\partial\mathcal{S}$.

To this purpose we the consider the monotone boxes

$$\mathcal{B}(\mathcal{P}) = \{B = [\alpha_1, \beta_1] \times [\alpha_2, \beta_2] \in \mathcal{B} : P_x \in [\alpha_1, \beta_1], P_y \geq \alpha_2\}$$

Take the generic monotone box $B_l = [\alpha_1^{(l)}, \beta_1^{(l)}]\times[\alpha_2^{(l)}, \beta_2^{(l)}] \in \mathcal{B}(\mathcal{P})$. If $P_y > \beta_2^{(l)}$ then the point P is *over* the monotone box B_l and necessarily the segment QP crosses the boundary $\partial\mathcal{S}$ once in B_l and *below* P, due to the monotonicy of $\tilde{x}$ in B_l.

Otherwise the point $P \in B_l$. Let t^* be the unique solution of the polynomial equation $\tilde{x}(t) = P_x$. Next,

- if $\tilde{y}(t^*) < P_y$ then the segment QP crosses the boundary $\partial\mathcal{S}$ once in the monotone box, *below* P;
- if $\tilde{y}(t^*) > P_y$ then the segment QP does not cross the boundary $\partial\mathcal{S}$ once in B, *below* P;
- if $\tilde{y}(t^*) = P_y$ then P is on the boundary $\partial\mathcal{S}$.

Thus counting all these crossings, we are *usually* able to determine if a point is inside, outside or on the boundary of $\mathcal{S}$.

Unfortunately $\partial\mathcal{S}$ can have points $S_k = (\tilde{x}(t_k), \tilde{y}(t_k))$ where

$$\lim_{t\to t_k^-} \tilde{y}'(t) \lim_{t\to t_k^+} \tilde{y}'(t) < 0,$$

that geometrically means that there is locally a vertical turn of boundary from left to right (or conversely from right to left). Even worst, there can be points P where the boundary is locally a vertical segment and it is not straightforward to determine how many times the line l crosses $\partial\mathcal{S}$ *below* P. In these *singular* cases we have used an algorithm that is based on the well-known *winding* theorem to ascertain whether P belongs or not to $\mathcal{S}$. In particular, to determine if $P = (P_x, P_y)$ belongs to $\mathcal{S}$, we computed by a (shifted) Gauss-Legendre rule of sufficiently high degree of precision the so called *winding number* $\mathrm{wind}(P, \tilde{x}, \tilde{y}) \in \mathbb{Z}$,

$$\mathrm{wind}(P, \tilde{x}, \tilde{y}) := \frac{1}{b-a} \int_a^b \frac{\tilde{y}'(\tilde{x}(t) - P_x) - \tilde{x}'(\tilde{y}(t) - P_y)}{(\tilde{x}(t) - P_x)^2 + (\tilde{y}(t) - P_y)^2} dt.$$

If this quantity is odd then the point belongs to $\mathcal{S}$ otherwise is not inside such domain. We point out that the Hormann-Agathos algorithm is based on a clever computation of this quantity whenever $\mathcal{S}$ is a polygon.

Remark 1 *A common istance is to determine what points of a set $X \subset \mathbb{R}^2$ belong to $\mathcal{S}$. In this case, the construction of the rectangles $\mathcal{B}$ is made once for all.*

Remark 2 *In our implementation, each monotone box $\mathcal{B}^{(j,k)}$ has abscissae ranging in the interval*

$$\left[\min(\tilde{x}(T_i^{(j,k)}), \tilde{x}(T_{i+1}^{(j,k)})), \max(\tilde{x}(T_i^{(j,k)}), \tilde{x}(T_{i+1}^{(j,k)}))\right].$$

If we partition $[T_i^{(j,k)}, T_{i+1}^{(j,k)}]$ as $\cup_{s=1}^{n_{i,j,k}-1} [T_{i,s}^{(j,k)}, T_{i,s+1}^{(j,k)}]$ where

$$T_i^{(j,k)} = T_{i,1}^{(j,k)} < T_{i,2}^{(j,k)} < \ldots T_{i,n_{i,j,k}}^{(j,k)} = T_{i+1}^{(j,k)}$$

and define the monotone boxes

$$\mathcal{B}_{i,s}^{(j,k)} := [T_{i,s}^{(j,k)}, T_{i,s+1}^{(j,k)}] \times [\min_{t\in[T_{i,s}^{(j,k)}, T_{i,s+1}^{(j,k)}]} \tilde{y}(t), \max_{t\in[T_{i,s}^{(j,k)}, T_{i,s+1}^{(j,k)}]} \tilde{y}(t)]$$

we have again that $\cup\mathcal{B}_{i,s}^{(j,k)} \subseteq \mathcal{B}$ contains $\mathcal{S}$, but being usually a strict subset of $\mathcal{B}$, in general the solution of an inferior number of polynomial equations is needed to establish what points of X belong to $\mathcal{S}$. In passing, we observe that there is numerical evidence that a too high number of boxes slows down the in domain process.

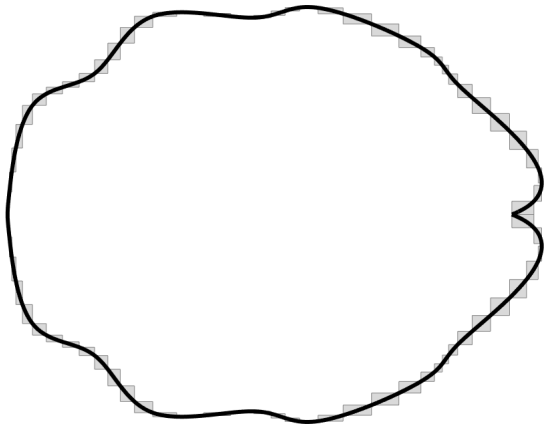

Fig. 2. A domain $\mathcal{S}$ of (spline) curvilinear type and a *thinner* set of monotone boxes (w.r.t. the previous figure).

Remark 3 *In the algorithm, to see if a point $P = (P_x, P_y)$ belongs to the boundary, we solved a polynomial equation $\tilde{x}(t^*) = P_x$ and then tested that $\tilde{y}(t^*) = P_y$. In view of numerical errors, we can only establish that a point is very close to the boundary, that is $|\tilde{y}(t^*) - P_y|$ is below a certain threshold.*

Remark 4 *In the case each degree δ_k is equal to 1, i.e. $\mathcal{S}$ is a polygon, it is more profitable to use to the well known Hormann-Agathos algorithm, implemented in Matlab by the* `inpolygon` *routine.*

3 Caratheodory-Tchakaloff subsampling

The purpose of this section is to show how from a rule with high cardinality, positive weights, and $ADE = \delta$ we can extract another with the same degree of precision, positive weights, but with cardinality equal at most to the dimension N_δ of the vector space $\mathbb{P}_\delta$.

In some recent papers, the authors have applied a mathematical tool named CATCH (acronym of (Caratheodory-Tchakaloff) subsampling), for such compression of discrete measures, proposing its application to discrete polynomial Least squares by sparse moment matching. In this framework, the method selects from a large discretization of a given region a much smaller number of (weighted) sampling points, even on a complex shape as can be the polygons investigated in this paper, keeping numerically invariant the Least Squares approximation estimates.

The key point is the following discrete version of Tchakaloff theorem [18] proved by Caratheodory theorem on finite-dimensional conic/convex combinations [5], [9],

Theorem 1 *Let μ be a multivariate measure whose support is a $\mathbb{P}_\delta$-determining finite set $X = \{\xi_i\} \subset \mathbb{R}^d$ (i.e., n-degree polynomials vanishing there vanish ev- erywhere), with correspondent positive weights (masses) $\lambda = \{\lambda_i\}$, $i = 1, \ldots, M$, $M = card(X) > N_\delta$, N_δ being the dimension of $\mathbb{P}_\delta$. Then, there exist a cubature formula for the discrete measure μ, with nodes $T_n = \{t_j\} \subset X$ and positive weights $\mathbf{w} = \{w_i\}$, $1 \leq j \leq m$, with $m \leq N_\delta$, such that*

$$\int_X p(x)d\mu = \sum_{i=1}^{M} \lambda_i p(x_i) = \sum_{j=1}^{m} w_j p(t_j), \ \forall p \in \mathbb{P}_\delta.$$

From the numerical side, given any polynomial basis $\{\phi_k\}$ of $\mathbb{P}_\delta$, define the Vandermonde-like matrix $V = \{V_n(X)\}_{i,j} = \phi_j(\xi_i)$ and let γ the vector of moments of the polynomial basis $\{\phi_j\}$ with respect to the original discrete measure and consider the underdetermined moment system $V^T u = \gamma$.

Caratheodory/Tchakaloff theorem asserts that there exists a sparse nonnegative solution u to the system above, whose nonvanishing components (i.e., the weights $\{w_j\}$) are at most N_δ and determines the corresponding reduced sampling points $T_\delta = \{t_j\} \subseteq X$, that we may term the Caratheodory-Tchakaloff (CATCH) points of X.

The computation of these nodes has been considered in several papers (see e.g. [16], [19]). To our knowledge, essentially two approaches have been developed to get these rules, i.e. via Linear Programming (LP) and Quadratic Programming (QP).

About the LP approach, it consists in solving via simplex-method

$$\begin{cases} \min_{u \geq 0} c^T u \\ V^T u = b, \quad u \geq 0 \end{cases} \tag{1}$$

where the constraints identify a polytope (the feasible region) in $\mathbb{R}^M$ and the vector c is chosen to be linearly independent from the rows of V^T, so that the objective functional is not constant on the polytope [9], [19].

The QP based algorithm requires instead the solution of the NonNegative Least Squares (NNLS) problem

$$\text{compute } u^* : \|Au^* - \gamma\|_2 = \min_u \|Au - \gamma\|_2, u \geq 0, \tag{2}$$

in which u^* can be obtained by the well-known *Lawson-Hanson* active set optimization method [8], which determines a sparse solution to (2). Its application gives a residual $\epsilon = \|Au^* - \gamma\|_2$ that is typically very small, say $< 10^{-14}$ for $\delta \leq 30$.

Our numerical experience with now available Matlab software has shown that NNLS usually performs better than LP in computing the CATCH weights, at least for moderate degrees δ (namely, N_δ for mild degrees) [9]. Consequently all our codes are based on this the application of Lawson-Hanson method to compute the Caratheodory/Tchakaloff sets.

We point out that there are several versions of NNLS codes available in Matlab. One is the built-in function `lsqnonneg`, based on the Lawson-Hanson algorithm while an open-source version present in the package NNLSlab in [13]. Other alternatives are often obtained by MEX files and for generality purposed will not be used here.

Remark 5 *As alternative, one can compress the rule via the QR algorithm with column pivoting proposed in 1965 by Businger and Golub [4], which is implemented for example by the Matlab* backslash *operator. This approach is equivalent to a greedy selection of the columns of A in order to maximize the successive volumes, and eventually the (absolute value of the) determinant (the column selection problem being NP-hard). If the matrix has full rank, the final result is a weight vector w^* where only $N \leq N_\delta = (\delta+1)(\delta+2)/2$ components are nonzero, so that we can extract the cubature nodes $T_\delta = \{t_j\}$ from X by the column indexes corresponding to such components. A drawback is that the resulting weights $w^* = \{w_k^*\}$ may not be all positive, but typically the negative ones are few and of small size, so that the relevant stability parameter*

$$cond(\{w_k^*\}) = \frac{\sum_{k=1}^N |w_k^*|}{|\sum_{k=1}^N w_k^*|} \geq 1$$

is not far from 1 *(see also [16]).*

Remark 6 *In the cubature framework an algorithm termed Recursive Halving Forest, based on a hierarchical SVD, has been proposed in [19]. Performances are reported for large scale problems (say that the order of N_δ is 10^3, 10^4). Unfortunately the software is not available and thus cannot be applied here as comparison.*

4 Algebraic cubature on curvilinear domains

Let $\mathcal{S}$ be a simple and simply connected domain and suppose that the boundary is defined parametric by splines $\tilde{x}$, $\tilde{y}$, as required in the previous chapter.

In this section we describe an algorithm that computes an algebraic cubature

formula of degree n over $\mathcal{S}$, i.e. such that

$$\int_{\mathcal{S}} p_n(x,y)d\mathcal{S} = \sum_{i=1}^{N_n} w_i p_n(x_i,y_i)$$

for any bivariate polynomial p_n of total degree at most n.

In particular the nodes (x_i, y_i) will belong to $\mathcal{S}$, are at most $(n+1)(n+2)/2$ and the respective weights w_i are all positive i.e., using a common acronym, the rule is of PI type that stands for *positive* weights and points *inside* the domain.

We proceed as follows.

(1) First we compute the *moments* γ_i of a certain polynomial basis $\{\phi_i\} = i = 1, \ldots, (n+1)(n+2)/2$ of degree n over $\mathcal{S}$, i.e.

$$\gamma_i = \int_{\mathcal{S}} \phi_i(x,y)d\mathcal{S}.$$

These values can be efficiently obtained by means of Gauss-Green theorem without the need of numerical rules over $\mathcal{S}$.

In our Matlab software as polynomial basis $\{\phi_j\}$ we have used the total-degree product Chebyshev basis $\{T_p(\alpha_1(x))T_q(\alpha_2(y))\}$, $(x,y) \in [a_1,b_1] \times [a_2,b_2]$ (the smallest cartesian rectangle containing the domain, easily available when the all the monotone boxes are determined), with $T_h(\cdot) = \cos(h \arccos(\cdot))$ is the h-degree Chebyshev polynomial and $\alpha_i(s) = (2s - b_i - a_i)/(b_i - a_i)$, $s \in [a_i, b_i]$, $i = 1, 2$.

By Gauss-Green theorem (see e.g. [1]),

$$\gamma_i = \int_{\mathcal{S}} \phi_i(x,y)dxdy = \oint_{\partial\mathcal{S}} \Phi_i(x,y)dy \tag{3}$$

where, for some p, q,

$$\Phi_i(x,y) = \int \phi_i(x,y)dx = T_q(\alpha_2(y)) \int T_p(\alpha_1(x))dx$$

In particular, for $p = 0$

$$\int T_p(\alpha_1(x))dx = x,$$

for $p = 1$

$$\int T_p(\alpha_1(x))dx = \frac{b_1 - a_1}{4} \cdot \alpha_1^2(x),$$

while for $p \geq 2$

$$\int T_p(\alpha_1(x))dx = \frac{b_1 - a_1}{2} \cdot \left(\frac{p}{p^2-1} T_{p+1}(\alpha_1(x)) - \frac{x}{p-1} T_p(\alpha_1(x)) \right).$$

Setting $I_j^{(k)} = [t_j^{(k)}, t_{j+1}^{(k)}]$, $P_{k,j} = (\tilde{x}(t_j^{(k)}), \tilde{y}(t_j^{(k)}))$ and denoting with $P_{k,j} \frown P_{k,j+1}$ the arc of $\partial\mathcal{S}$ joining $P_{k,j}$ with $P_{k,j+1}$, we have

$$\gamma_i = \oint_{\partial\mathcal{S}} \Phi_i(x,y)dy = \sum_{k,j} \int_{P_{k,j} \frown P_{k,j+1}} \Phi_i(x,y)dy$$
$$= \sum_{k,j} \int_{t_j^{(k)}}^{t_{j+1}^{(k)}} \Phi_i(\tilde{x}(t), \tilde{y}(t))\ \tilde{y}'(t)dt. \tag{4}$$

Observing that

- since ϕ_i is a polynomial of total degree n then Φ is a polynomial of total degree $n+1$,
- in the interval $I_j^{(k)}$ both $\tilde{x}$, $\tilde{y}$ are polynomials of degree δ_k,

each integrand is a polynomial of degree $(n+1)\delta_k + \delta_k - 1 = (n+2)\delta_k - 1$ that can be exactly integrated by a (shifted) Gauss-Legendre formula with $\lceil \frac{(n+2)\delta_k}{2} \rceil$ points.

(2) At the j-th iteration of the algorithm we define a set $\mathcal{P}_j$, consisting in the union of $\mathcal{P}_{j-1}$ with some new M_j points that belong to $\mathcal{S}$ to $\mathcal{P}_{j-1}$. To this purpose, it is fundamental to apply the *in-domain* algorithm previously defined. Next, we apply the algorithm that computes the Tchakaloff formula with nodes $\{(x_i^{(j)}, y_i^{(j)})\}_{i=1,\dots,N_n}$ and weights $\{w_i^{(j)}\}_{i=1,\dots,N_n}$, finally testing if the so obtained rule is such that

$$\gamma_s^{(j)} = \sum_{i=1}^{N_n} w_i^{(j)} \phi_s(x_i^{(j)}, y_i^{(j)}),\ \ s = 1, \dots, N_n$$

well approximates γ_s, i.e.

$$\|\gamma_s^{(k)} - \gamma_s\|_2 \leq \texttt{toll} \tag{5}$$

where `toll` is the tolerance fixed by the user. If (5) does not hold we iterate the procedure until (5) is satisfied or a maximum number of iterations is reached, providing in this case an error message.

In our implementation of the Caratheodory-Tchakaloff compression we used the Matlab built-in routine `lsqnonneg` but one can alternatively use the algorithm proposed in [13].

The fact that this procedure terminates in a finite number of iterations comes from a theorem proved in [20] and that we state below.

Let Φ be the linear span of continuous, real-valued, linearly indepentent functions $\{\phi_k\}_{k=1,\dots,n}$ defined on a compact set D. Assume Φ satisfies the Krein condition (i.e. there is at least one $f \in \Phi$ which does not vanish on D) and that L is a positive linear functional, i.e. $Lf \geq 0$ whenever $f \geq 0$. If $\{x_i\}_{i=1}^{+\infty}$ is an everywhere dense subset of D, then for sufficiently large m, the set $\{x_i\}_{i=1}^{m}$

is a Tchakaloff set, i.e.

$$Lf = \sum_{i=1}^{m} \lambda_i f(x_i), \;\; f \in \Phi$$

where $\lambda_i > 0$ *and* $\{x_i\}_{i=1,\dots,m} \subseteq D$.

Setting $D = \mathcal{S}$, $Lf = \int_{\mathcal{S}} f \; d\mathcal{S}$, and $\Phi = \mathbb{P}_n$, i.e. the space of bivariate polynomials of total degree n, this theorem implies that our algorithm determines the desired cubature rule of PI-type after a finite number of iterations, extracting the formula by the Lawson-Hanson algorithm from a set of certain m points of D.

Remark 7 *(Convergence rate). Concerning the approximation quality, if* f *is a continuous function over* $\mathcal{S}$, $\mu(S)$ *is the measure of* $\mathcal{S}$, *and we denote by* $I_{\mathcal{S}}(f)$, $I_n(f)$ *respectively the integral to be computed and its numerical approximation via this new cubature rule having degree of precision* n, *denoting by* p_n^* *the best uniform approximation of degree* n, $\|I_{\mathcal{S}}\|_\infty = \mu(\mathcal{S})$, $\|I_n\|_\infty = \sum_i |w_i| = \mu(\mathcal{S})$ *(here we exploited the fact that all the weights are positive),*

$$\begin{aligned} |I_{\mathcal{S}}(f) - I_n(f)| &\leq |I_{\mathcal{S}}(f) - I_{\mathcal{S}}(p_n^*)| + |I_{\mathcal{S}}(p_n^*) - I_n(p_n^*)| \\ &\leq \|I_{\mathcal{S}}\|_\infty \|f - p_n^*\|_\infty + \|I_n\|_\infty \|f - p_n^*\|_\infty \\ &= (\mu(\mathcal{S}) + \sum_i |w_i|) E_n(f, \mathcal{S}) \\ &= 2\mu(\mathcal{S}) E_n(f, \mathcal{S}) \end{aligned}$$

where $E_n(f, \mathcal{S})$ *is the best approximation error at degree* n *over* $\mathcal{S}$. *We observe that with respect to the error of the algorithm in [15] these results offer in general a tighter estimate.*

In particular by multivariate versions of Jackson theorem, we have

$$E_n(f, \mathcal{S}) = \mathcal{O}(n^{-(p+\theta)}), \;\; f \in C^{p+\theta}(\mathcal{S})$$

for every function f *that has* θ*-Hölder continuous* p*-th partial derivatives, with* $\theta \in (0, 1]$.

Remark 8 *(Boundary Approximation). As observed in [15],* $\mathcal{S}$ *can be an approximation of a certain domain* Ω.

As stated before, assume that the vertices $V_k \in \partial\Omega$, $k = 1, \dots, M+1$, *are computed and that* $\partial\Omega \approx \partial\mathcal{S} := \cup_{k=1}^{M} V_k \frown V_{k+1}$ *where each curvilinear side* $V_k \frown V_{k+1}$ *is tracked by a spline curve of degree* $\delta_k \leq 3$, *interpolating an ordered subsequence of control points* $P_{1,k} = V_k, P_{2,k}, \dots, P_{m_k,k} = V_{k+1}$.

Furthermore, suppose that it is adopted a cumulative chordal parametrization, i.e. each $V_k = (\tilde{x}(t^{(k)}), \tilde{y}(t^{(k)}))$ *so that* $I^{(k)} = [t^{(k)}, t^{(k+1)}]$*, being*

$$t^{(k+1)} = t^{(k)} + \sum_{j=1}^{m_k - 1} |P_{j+1,k} - P_{j,k}|.$$

In these assumptions, also supposing that the boundary has a global C^4 *parametrization, that* $H = \max_{j,k} |P_{j+1,k} - P_{j,k}|$*, and setting* $\Omega \Delta \mathcal{S} = (\Omega \backslash \mathcal{S}) \cup (\mathcal{S} \backslash \Omega)$*, with an approach similar to that adopted in [15], by a result in [10], we get*

$$\begin{aligned} |I_\Omega(f) - I_n(f)| &\leq I_{\Omega\Delta\mathcal{S}}(|f|) + |I_\mathcal{S}(f) - I_n(f)| \\ &\leq \|f\|_{\infty,\Omega\Delta\mathcal{S}}\ \mu(\Omega\Delta\mathcal{S}) + 2\mu(\mathcal{S}) E_n(f, \mathcal{S}) \\ &\approx \|f\|_{\infty,\Omega\Delta\mathcal{S}}\ l(\partial\Omega) O(H^4) + 2\mu(\Omega) E_n(f, \Omega). \end{aligned}$$

where $E_n(f, \mathcal{S})$ *is the best polynomial approximation error of degree* n *of the function* f*, over* $\mathcal{S}$ *(notice that differently from [15] all the nodes are inside the domain* $\mathcal{S}$ *and the weights are positive).*

Remark 9 *(Moments approximation). Let* $\{\phi_k\}$ *be a basis of the space of polynomials of total degree* n*, and* $\gamma_k = \int_\Omega \phi_k d\Omega$*, being* Ω *a general domain. In our examples* Ω *corresponds to* $\mathcal{S}$*, but it can be a more general compact domain. Suppose that the formula has nodes* (x_i, y_i)*, weights* w_i*,* $i = 1, \ldots, N_n$ *and is of PI-type. If* $\tilde{\gamma}_k = \sum_{i=1}^{N_n} w_i \phi_k(x_i, y_i)$ *and* $\epsilon_{mom} = \|\tilde{\gamma}_k - \gamma_k\|_2$*, then, for* $C \leq 2(\mu(\Omega) + \sqrt{\mu(\Omega)})$*, we have*

$$|\int_\Omega f(x, y) d\Omega - \sum_{i=1}^{N_n} w_i f(x_i, y_i)| \leq C E_n(f, \Omega) + \|f\|_{\mathcal{L}_2(\Omega)} \epsilon_{mom}$$

for any $f \in C(\Omega)$ *(see [16] for details, using the Lebesgue measure* $d\lambda$ *instead of the product Chebyshev measure on the smallest rectangle containing* Ω*, having sides parallel to* x *and* y *axes).*

5 Numerical experiments

The purpose of this section is twicefold, on a side we test the new cubature algorithm and on the other the in-domain routines.

We observe the Matlab routine `splinegauss` proposed in [15] is not necessarily of PI-type on non convex domains $\mathcal{S}$, since in these instances some nodes could be outside the domain and some weights could be even negative.

However, since it was based on a sequential use of tensorial Gauss-Legendre formula, the determination of the cubature rule was rather fast, w.r.t. the geometry of the domain and the degree of precision required.

For mild degrees, say $n \leq 15$, the new routine `splcub`, implementing the ideas developed in the previous sections, has similar cputimes but produces rules of PI-type with a number of nodes $N_n \leq (n+1)(n+2)/2$.

Being all the weights positive, these new formulas are optimally stable, i.e. the cubature condition number

$$\mathrm{cond}(\{w_i\}) = \frac{\sum_{i=1}^{N_n} |w_i|}{\sum_{i=1}^{N_n} w_i}$$

is equal to 1, the best possible result for rules with ADE at least equal to 0. Furthermore, they are appropriate for problems were the sampling of the integrand is not possible outside the domain.

We implemented an algorithm that can be sketched as follows:

(1) define a mesh on the smallest rectangle containing the domain $\mathcal{S}$, say $[a_1, b_1] \times [a_2, b_2]$; in particular if $ADE = n$, then setting $k = \lfloor n^{1.5} \rfloor$, we considered the points $P = ((P_x)_i, (P_y)_j)$ where

$$(P_x)_i = a_1 + i\frac{a_2 - a_1}{k-1}, (P_y)_j = b_1 + j\frac{b_2 - b_1}{k-1}$$

i.e. a tensor grid mesh, based on k equispaced data in each direction.
(2) establish those P strictly inside the domain, say $\mathcal{P}^* = \{P_1^*, \ldots, P_m^*\}$;
(3) compute the moments γ_k over $\mathcal{S}$ of the total-degree product Chebyshev basis

$$\{T_p(\alpha_1(x))T_q(\alpha_2(y))\},\ (x,y) \in [a_1, b_1] \times [a_2, b_2],$$

by means of Gauss-Green theorem,
(4) extract the nodes and the weights using the technique introduced in [16] and described in section II.

In case of failure, i.e. the norm 2 of the moment error is bigger than a fixed tolerance, say 10^{-12}, we proceed by defining a finer grid, determining those points in $\mathcal{S}$, providing the union with the previous $P_1^*, \ldots, P_m^*$, and retrying points 3 and 4 of the algorithm above.

Due to the fact that for higher ADE, ill-conditioning of Vandermonde matrix V on $\mathcal{P}^*$ may occur, whenever it was necessary, we have reothortogonalised the basis by the *economy size* QR algorithm.

All the Matlab routines for the numerical experiments are available at [13].

We start our numerical test by analysing the region $\mathcal{S}_1$, whose boundary is defined by $\tilde{x}$, $\tilde{y}$, that in $I^{(1)} = \{I_j^{(1)}\}_{j=1,\dots,4}$ are linear splines, i.e. $\delta_1 = 2$, while in $I^{(2)} = \{I_j^{(2)}\}_{j=1,\dots,5}$ are cubic splines satisfying *not-a-knot* conditions. The region is depicted in Figure 3 and it consists of a polygon with a side substituted by a curve.

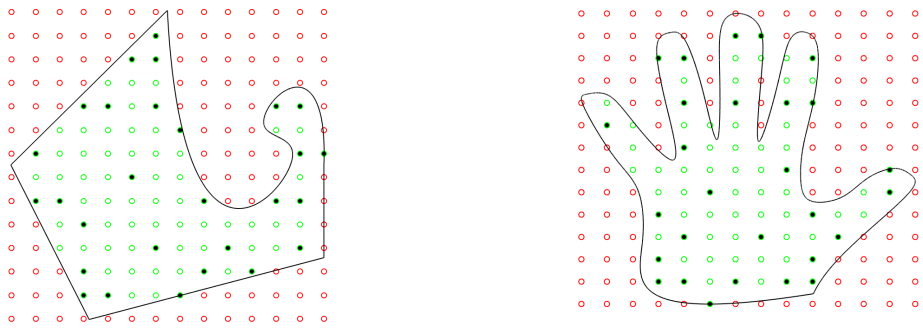

Fig. 3. The spline curvilinear domains $\mathcal{S}_i$ with $i = 1, 2$, the grid points P outside the domain (in red), those inside the domain (in green) and the nodes of a cubature formula of PI-type for $n = 6$ (black dots).

In particular if the routine in [15] is applied to these tests, several nodes are external to the domain, and negative weights are present.

ADE	#	$\#_O$	cond	res
5	21	0	1	$7.2e-17$
10	66	0	1	$1.3e-16$
15	136	0	1	$1.1e-16$
20	231	0	1	$1.2e-16$

Table 1
Cardinality # of the nodes, number of nodes $\#_O$ outside the domain, conditioning of the rule and residual of the rule w.r.t. the moments $\{\gamma_k\}$ of domain $\mathcal{S}_1$.

ADE	#	$\#_O$	cond	res
5	21	0	1	$6.7e-18$
10	66	0	1	$3.8e-18$
15	136	0	1	$3.9e-18$
20	231	0	1	$4.8e-18$

Table 2
Cardinality # of the nodes, number of nodes $\#_O$ outside the domain, conditioning of the rule and residual of the rule w.r.t. the moments $\{\gamma_k\}$ of the hand-like domain $\mathcal{S}_2$.

It not surprising that, with the exception of the residuals, the results are identical, in view of the fact that we have computed rules with at most $N_n = (n+1)(n+2)/2$ nodes. All the rules are of PI-type, and the residuals are particularly small.

As second domain $\mathcal{S}_2$, we considered a hand-like domain, whose boundary is defined by $\tilde{x}$, $\tilde{y}$, that in $I^{(1)} = \{I_k^{(1)}\}_{k=1,\dots,37}$ are both cubic splines satisfying *not-a-knot* conditions.

Remark 10 *(Nested rules). We point out that in both the examples, as shown in [17], if it is available a formula of PI-type, with ADE equal to n, then one can extract those with ADE less than n, i.e. nested rules, by further application of Caratheodory-Tchakaloff compression, with the advantage of having estimates of the cubature error, without further function evaluations.*

As final tests, we show the cputimes necessary to process m in-domain operations, on the regions $\mathcal{S}_i$, $i = 1, 2$. To this purpose we have defined the Matlab routine `incurvpolygon`, that implements the ideas introduced in section II.

m	$\mathcal{S}^{(1)}$	$\mathcal{S}^{(2)}$
10^2	$6e-03$	$1e-02$
10^3	$1e-02$	$3e-02$
10^4	$4e-02$	$7e-02$
10^5	$6e-01$	$6e-01$

Table 3
Cputime of the application of the *in-domain* algorithm for m points, respectively in $\mathcal{S}^{(1)}$ and $\mathcal{S}^{(2)}$.

The results are particularly positive, and allow the application of the cubature algorithm at higher degrees. In this case the bottleneck is the application of the Lawson-Hanson algorithm. If negative weights are not an issue, the alternative provided by QR method provides better cputimes and formulae that usually have quite good cubature condition number.

6 Acknowledgements

This work is partially supported by the DOR funds of the University of Padova, and by the GNCS-INdAM. Their research has been accomplished within the RITA *Research ITalian network on Approximation*.

References

[1] T.M. Apostol,, Calculus, 2nd ed., vol. II, Blaisdell, 1969.

[2] B. Bauman, A. Sommariva and M. Vianello, Compressed cubature over polygons with applications to optical design, submitted.

[3] L. Beirão da Veiga, A. Russo and G. Vacca, The Virtual Element Method with curved edges, ESAIM Mathematical Modelling and Numerical Analysis 53 (2019), pp.375–404.

[4] P.A. Businger, G.H. Golub, Linear least-squares solutions by Householder transformations, Numer. Math. 7 (1965), pp.269–276.

[5] C. Caratheodory, Über den Variabilittsbereich der Koeffizienten von Potenzreihen, die gegebene Werte nicht annehmen, Math. Ann. 64 (1907), pp.95–115.

[6] P.J. Davis, A construction of nonnegative approximate quadratures, Math. Comp. v.21 (1967), pp.578–582.

[7] K. Hormann and A. Agathos, The point in polygon problem for arbitrary polygons, Comput. Geom. Theory Appl. v.20, n.3 (2001), pp. 131–144.

[8] C.L. Lawson and R.J. Hanson, Solving least squares problems, Classics in Applied Mathematics 15, SIAM, Philadelphia, 1995.

[9] F. Piazzon, A. Sommariva and M. Vianello, Caratheodory-Tchakaloff Subsampling, Dolomites Res. Notes Approx. DRNA 10 (2017), pp.5–14.

[10] M. Sakai, R.A. Usmani, On orders of approximation of plane curves by parametric cubic splines, BIT Numerical Mathematics 30 (1990) 735–741.

[11] G. Santin, A. Sommariva and M. Vianello, An algebraic cubature formula on curvilinear polygons, Appl. Math. Comput. 217 (2011), pp.10003–10015.

[12] M. Slawski, Non-negative least squares: comparison of algorithms, `https://sites.google.com/site/slawskimartin/code`.

[13] A. Sommariva, `http://www.math.unipd.it/~alvise/software.html`.

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

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

[16] A. Sommariva and M. Vianello, Compression of multivariate discrete measures and applications, Numer. Funct. Anal. Optim. 36 (2015), pp.1198–1223.

[17] A. Sommariva and M. Vianello, Nearly optimal nested sensors location for polynomial regression on complex geometries, to appear on Sampling Theory in Signal and Image Processing.

[18] V. Tchakaloff, Formules de cubatures mécaniques à coefficients non négatifs, (French) Bull. Sci. Math. 81 (1957), pp.123–134.

[19] M. Tchernychova, *Caratheodory* cubature measures, Ph.D. dissertation in Mathematics (supervisor: T. Lyons), University of Oxford, 2015.

[20] Don.R. Wilhelmsen, A Nearest Point Algorithm for Convex Polyhedral Cones and Applications to Positive Linear approximation, Math. Comp. vol. 30, n. 133, January 1976, pp.48–57-