BIT Numerical Mathematics (2006)46:000-000
DOI:10.1007/s10543-000-0000-x
© Springer 2006.

PRODUCT GAUSS CUBATURE OVER POLYGONS BASED ON GREEN'S INTEGRATION FORMULA *

A. SOMMARIVA[1] and M. VIANELLO[2] †

[1] *Department of Pure and Applied Mathematics, University of Padova, via Belzoni 7, Padova 35131, (Italy). email: alvise@math.unipd.it*

[2] *Department of Pure and Applied Mathematics, University of Padova, via Belzoni 7, Padova 35131, (Italy). email: marcov@math.unipd.it*

Abstract.

We have implemented in Matlab a Gauss-like cubature formula over convex, non-convex or even multiply connected polygons. The formula is exact for polynomials of degree at most $2n-1$ using $N \sim mn^2$ nodes, m being the number of sides that are not orthogonal to a given line, and not lying on it. It does not need any preprocessing like triangulation of the domain, but relies directly on univariate Gauss-Legendre quadrature via Green's integral formula. Several numerical tests are presented.

AMS subject classification (2000): 65F20.

Key words: Gauss-like cubature, polygons, Green's formula.

1 Introduction.

We consider the problem of constructing a cubature formula

$$\sum_{(\xi,\eta)\in\Xi_{2n-1}} w_{\xi,\eta}\, f(\xi,\eta) \approx \iint_{\Omega} f(x,y)\,dxdy\ ,\quad \Omega \subset \mathbb{R}^2 \ \text{ polygon}\ , \tag{1.1}$$

which is exact for all bivariate polynomials of degree at most $2n-1$, stable (i.e. such that $\sum_{(\xi,\eta)\in\Xi_{2n-1}} |w_{\xi,\eta}|$ is bounded), and simple to implement by one of the most popular computing tools, Matlab (cf. [17]).

The literature on cubature over polygons is not very wide, despite the fact that polygons (and polyhedra) are at the core of computational geometry. The main topics have been the construction of cubature formulas over regular polygons (cf. e.g. [12, 19]), of exact formulas for moments and polynomials (more generally also over polyhedra, see e.g. [15, 21]), and of special methods for particular densities in statistical applications, like e.g. [4]. A more general approach has been pursued in [11].

A common feeling in the numerical community is probably that, since reliable and efficient polygon triangulators are at disposal (cf. [18]) as well as adaptive integrators over collection of triangles (cf. [2]), the problem of cubature

*Received October 2006. Revised February 2007. Communicated by Tom Lyche.

†Work supported by the "ex-60%" funds of the University of Padova, and by the INdAM GNCS.

over polygons is virtually solved. But, fast triangulators that manage general polygons (for example polygons with holes) are typically written in C/C++ (cf. e.g. [13, 18]), and are not presently easy to interface with good integrators over triangles within the Matlab system (at least, by Matlab "standard" users). On the other hand, a brute-force approach like that usually suggested by many automatic integrators (like e.g. the Matlab `dblquad` function, cf. [17]) to manage nonstandard domains, that is integrating the product of the given integrand by the characteristic function of the domain (for polygons, by the Matlab `inpolygon` function) on some enclosing rectangle, can work but is often unreliable and clearly inefficient since an artificial discontinuity at the boundary is introduced.

On the other hand, one of the corner-stones of multivariate calculus, Green's integral formula [10, 1], is seldom used explicitly in the numerical cubature context. Such a formula, which in one of its basic formulations can be written as

$$\iint_{\Omega} f(x,y)\,dxdy = \oint_{\partial\Omega} \mathcal{F}(x,y)\,dy\ , \quad \mathcal{F}(x,y) = \int f(x,y)\,dx\ , \tag{1.2}$$

(f being continuous on a domain Ω with piecewise smooth boundary described counterclockwise), gives in principle an appealing tool for numerical cubature, since it transforms a 2-dimensional into a 1-dimensional problem. Its practical use, however, requires the knowledge of a primitive of the integrand, which seems to restrict the field of application to a subclass of analytically known functions.

In a recent paper [23], we have exploited Green's formula to construct cubature formulas from scattered data over general polygons, via interpolation by radial basis functions (thin-plate splines). There, Green's formula has been applied directly to the radial basis functions, as a first step in computing the cubature weights. In such a way we have been able to extend the cubature formula originally constructed for squares and rectangles in [22].

In this paper, Green's formula over polygons is again the key for solving problem (1.1). A fixed x-primitive $\mathcal{F}(x,y)$ in (1.2) is computed by univariate Gauss-Legendre quadrature, and integrated along the sides still by Gauss-Legendre quadrature. When the integrand is a bivariate polynomial of degree at most $2n-1$, the first quadrature is exact with n nodes and gives a polynomial of degree at most $2n$, which restricted to a side is a polynomial of degree at most $2n$ in the side parametrization. This can be integrated exactly by $n+1$ Gauss-Legendre nodes on the side. No triangulation is required, since Green's formula needs only the boundary as a counterclockwise sequence of vertices.

The whole construction and the corresponding stability and error estimates are given in section 2. In section 3, we show the behavior of the cubature formula by integrating some test functions over polygons with different geometries.

2 Gauss-like cubature by Green's formula.

We begin by stating the main result of the paper (construction of Gauss-like cubature formulas over polygons) as a theorem.

THEOREM 2.1. *Let $\Omega \subset \mathbb{R}^2$ be the closure of a bounded and simply connected polygon with boundary described counterclockwise by the sequence of vertices*

$$V_i = (\alpha_i, \beta_i) \ , \ \ i = 1, \dots, L \ , \ \ L \geq 3 \ ,$$

$$\partial\Omega = [V_1, V_2] \cup [V_2, V_3] \cup \cdots \cup [V_L, V_{L+1}] \ , \ \ V_{L+1} = V_1 \ . \tag{2.1}$$

Let $f \in C(\mathcal{R})$ and let α be fixed, where

$$\Omega \subseteq \mathcal{R} = [a, b] \times [c, d] \ , \ \ \alpha \in [a, b] \ . \tag{2.2}$$

Let $\{\tau_j^s\}$ and $\{\lambda_j^s\}$, $1 \leq j \leq s$, be the nodes and weights of the Gauss-Legendre quadrature formula of degree of exactness $2s-1$ on $[-1, 1]$, cf. [7].

Then, the following cubature formula is exact over Ω for all bivariate polynomials of degree at most $2n-1$

$$I_{2n-1}(f) = \sum_{i \in \mathcal{I}_{\Omega,\alpha}} \sum_{j=1}^{n_i} \sum_{k=1}^{n} w_{ijk} \, f(\xi_{ijk}, \eta_{ij}) \ , \tag{2.3}$$

$$\mathcal{I}_{\Omega,\alpha} = \{i : \Delta\beta_i \neq 0\} \cap \{i : \alpha_i \neq \alpha \text{ or } \alpha_{i+1} \neq \alpha\} \subseteq \{1, \dots, L\} \ , \tag{2.4}$$

$$n_i = \begin{cases} n \ , & \Delta\alpha_i = 0 \\ n+1 \ , & \Delta\alpha_i \neq 0 \end{cases} \tag{2.5}$$

(i.e. $\mathcal{I}_{\Omega,\alpha}$ consists of the indexes of the sides which are not orthogonal to the line $x = \alpha$ and not lying on it), and the nodes and weights are given by

$$\xi_{ijk} = \frac{x_i(\tau_j^{n_i}) - \alpha}{2} \, \tau_k^n + \frac{x_i(\tau_j^{n_i}) + \alpha}{2} \ , \ \ x_i(t) = \frac{\Delta\alpha_i}{2} \, t + \frac{\alpha_i + \alpha_{i+1}}{2} \ , \tag{2.6}$$

$$\eta_{ij} = y_i(\tau_j^{n_i}) \ , \ \ y_i(t) = \frac{\Delta\beta_i}{2} \, t + \frac{\beta_i + \beta_{i+1}}{2} \ , \tag{2.7}$$

$$w_{ijk} = \frac{1}{4} \, \Delta\beta_i \, \left(x_i(\tau_j^{n_i}) - \alpha \right) \, \lambda_j^{n_i} \, \lambda_k^n \ , \tag{2.8}$$

Δ denoting the usual forward difference operator. Setting $m = card(\mathcal{I}_{\Omega,\alpha})$, the overall number of cubature nodes is

$$\mathcal{V} = \mathcal{V}_{n,\Omega,\alpha} = n \sum_{i \in \mathcal{I}_{\Omega,\alpha}} n_i \ , \tag{2.9}$$

with

$$\frac{L}{2} \, n^2 \leq mn^2 \leq \mathcal{V} \leq mn(n+1) \leq Ln(n+1) \ . \tag{2.10}$$

Moreover, we have the stability estimate

$$\sum_{i\in\mathcal{I}_{\Omega,\alpha}}\sum_{j=1}^{n_i}\sum_{k=1}^{n}|w_{ijk}| \le C_{\Omega,\alpha} = \max_{i\in\mathcal{I}_{\Omega,\alpha}}|\alpha-\alpha_i|\left(\sum_{i\in\mathcal{I}_{\Omega,\alpha}}|\Delta\beta_i|\right), \tag{2.11}$$

and the error estimate

$$\left|\iint_\Omega f(x,y)\,dxdy - I_{2n-1}(f)\right| \le (meas(\Omega)+C_{\Omega,\alpha})\,E_{2n-1}(f;\mathcal{R}),$$

$$E_{2n-1}(f;\mathcal{R}) = \min_{p\in\mathbb{P}^2_{2n-1}}\|f-p\|_{\infty,\mathcal{R}}. \tag{2.12}$$

Proof. By Green's formula (1.2) and (2.1) we can write

$$\iint_\Omega f(x,y)\,dxdy = \sum_{i=1}^{L}\int_{[V_i,V_{i+1}]}\mathcal{F}(x,y)\,dy = \sum_{i:\,\Delta\beta_i\neq 0}\int_{[V_i,V_{i+1}]}\mathcal{F}(x,y)\,dy, \tag{2.13}$$

where $\mathcal{F}(x,y)$ is any fixed x-primitive of $f(x,y)$ and the sum can be clearly restricted to the sides that are not parallel to the x-axis. Parametrizing each "active" side as

$$P_i(t) = (x_i(t),y_i(t)) = \frac{\Delta V_i}{2}\,t + \frac{V_i+V_{i+1}}{2}, \quad t\in[-1,1], \tag{2.14}$$

we can approximate the integral of $f(x,y)$ on that side by Gauss-Legendre quadrature of degree of exactness $2n_i-1$ (cf. (2.5))

$$\int_{[V_i,V_{i+1}]}\mathcal{F}(x,y)\,dy \approx \frac{\Delta\beta_i}{2}\sum_{j=1}^{n_i}\lambda_j^{n_i}\,\mathcal{F}(x_i(\tau_j^{n_i}),y_i(\tau_j^{n_i})). \tag{2.15}$$

Now, observe that if $f(x,y)$ is a polynomial of degree at most $2n-1$, then $\mathcal{F}(x,y)$ is a polynomial of degree at most $2n$, where the degree increase pertains only the x variable. This entails that $\mathcal{F}(x_i(t),y_i(t))$ is a polynomial of degree at most $2n$ in t, unless the side is parallel to the y-axis (i.e. $\Delta\alpha_i=0$) in which case it is still of degree at most $2n-1$, and thus formula (2.15) is exact for such polynomials.

An x-primitive of $f(x,y)$ is given by

$$\mathcal{F}(x,y) = \int_\alpha^x f(u,y)\,du, \tag{2.16}$$

which in turn can be approximated by Gauss-Legendre quadrature of degree of exactness $2n-1$ on the interval (α,x) (irrespectively to its actual orientation)

$$\mathcal{F}(x,y) \approx \frac{x-\alpha}{2}\sum_{k=1}^{n}\lambda_k^n\,f\left(\frac{x-\alpha}{2}\,\tau_k^n+\frac{x+\alpha}{2},y\right). \tag{2.17}$$

By combining (2.17) and (2.15) we obtain the cubature formula (2.3) over the polygon Ω which is exact for all bivariate polynomials of degree at most $2n-1$ (notice that $\mathcal{F}(x,y)$ in (2.16) vanishes on the sides that possibly lie on the line $x=\alpha$).

Formula (2.10) is a direct consequence of the definition of n_i in (2.5), the lower bound being attained when all sides are parallel to the co-ordinate axes, and the upper when no side is parallel to the x-axis.

Estimate (2.11) can be immediately obtained observing that $\sum_{j=1}^{s}\lambda_j^s = 2$ for the Gauss-Legendre quadrature weights, and that $|x-\alpha| \leq \max|\alpha-\alpha_i|$ for every point $(x,y)\in\partial\Omega$, Ω being a polygon. Finally, (2.12) is the extension to the cubature framework of the well-known error estimate for Polya-Steklov-like quadrature formulas (cf. e.g. [14, 25]). In fact, denoting by p^*_{2n-1} the best uniform polynomial approximation to f on Ω with degree $2n-1$, by polynomial exactness we have

$$\begin{aligned}\left|\iint_\Omega f(x,y)\,dxdy - I_{2n-1}(f)\right| &\leq \left|\iint_\Omega f(x,y)\,dxdy - \iint_\Omega p^*_{2n-1}(x,y)\,dxdy\right| \\ &+ \left|\iint_\Omega p^*_{2n-1}(x,y)\,dxdy - I_{2n-1}(p^*_{2n-1})\right| + \left|I_{2n-1}(p^*_{2n-1}) - I_{2n-1}(f)\right| \\ &\leq \left(\mathrm{meas}(\Omega) + \sum_{i\in\mathcal{I}_{\Omega,\alpha}}\sum_{j=1}^{n_i}\sum_{k=1}^{n}|w_{ijk}|\right) E_{2n-1}(f;\mathcal{R})\,. \quad \text{q.e.d.}\end{aligned}$$

In order to deepen some geometric, analytic, and computational features of the cubature formula, we make below some remarks.

REMARK 2.1. (multiply connected polygons)

The cubature formula (2.3) can be easily extended to multiply connected polygons, via the corresponding extension of Green's formula. Indeed, assume that the boundary of Ω be the union of an external boundary Γ^{ext} with a finite number of internal boundaries Γ^{int}_k, $k=1,\ldots,s$ (describing holes). Then we have

$$\iint_\Omega f(x,y)\,dxdy = \oint_{\Gamma^{\mathrm{ext}}}\mathcal{F}(x,y)\,dy - \sum_{k=1}^{s}\oint_{\Gamma^{\mathrm{int}}_k}\mathcal{F}(x,y)\,dy\,, \tag{2.18}$$

where all line integrals are taken counterclockwise and can be computed by (2.3). We stress that in these cases the integrand f has to be continuous and computable (at least) in the convex hull, i.e. also in the hole.

It is worth noticing that the polygon Ω can be multiply connected, but must be "simple" in a slightly generalized sense, i.e. self-intersections are allowed only at some vertices, that become the only multiple points of the boundary path.

REMARK 2.2. (convergence rate)

Concerning the convergence rate of (2.3) as $n\to\infty$, by the multivariate extension of Jackson theorem (cf. e.g. [20]) and (2.12), we get immediately

$$\iint_\Omega f(x,y)\,dxdy = I_{2n-1}(f) + \mathcal{O}\left((2n-1)^{-(p+\theta)}\right)\,,\quad f\in C^{p+\theta}(\mathcal{R})\,, \tag{2.19}$$

for every function f with Hölder continuous p-th partial derivatives, i.e. $p \geq 0$ and $\theta \in (0,1]$.

REMARK 2.3. (polynomial exactness)

The cubature formula (2.3) has been constructed just in order to be exact on all bivariate polynomials of degree at most $2n-1$. Thus, when it is applied to a polynomial of degree d, it is sufficient to choose $n \geq (d+1)/2$ to get the integral up to machine precision. If the matter is integrating a polynomial over a polygon, one could object that there exist other more direct approaches, for example using available analytic formulas for moments (cf. e.g. [15]). However, such formulas can typically manage only polynomials written in the classical monomial basis.

On the contrary, (2.3) is exact up to degree $2n-1$ irrespectively of the specific polynomial basis. This flexibility could be very useful in applications where integrals of arbitrary polynomials have to be computed with very high accuracy. For example, using a suitably scaled and shifted Legendre basis $\{P_{p_1}(c_1 x + c_2)P_{p_2}(c_3 y + c_4)\}$ instead of the monomial basis $\{x^{p_1} y^{p_2}\}$, $0 \leq p_1 + p_2 \leq d$, could be important for stability reasons in various computations involving moments of a polygon, like generating a bivariate orthogonal basis w.r.t. the Lebesgue measure by an orthogonalization process (cf., e.g., [5]), or computing the Gram matrix of the basis and then the continuous Least Squares approximation of a given function. See e.g. [7] for the role of "modified" moments in connection with univariate orthogonal polynomials, and [6] for the theory of multivariate orthogonal polynomials.

It is worth noticing that in the case of modified moments with respect to any bivariate polynomial basis like

(2.20)
$$\boldsymbol{\phi} = \{\phi_p(x,y)\}\ ,\ \ p = (p_1,p_2)\ ,\ \ \phi_p(x,y) = \Pi_{p_1}(x)\Pi_{p_2}(y)\ ,\ \ 0 \leq p_1 + p_2 \leq d\ ,$$

where $\Pi_k(\cdot)$ is any univariate polynomial basis, the modified moments can be computed by the following specialized version of the cubature formula

(2.21)
$$\iint_\Omega \phi_p(x,y)\,dxdy = \sum_{i \in \mathcal{I}_{\Omega,\alpha}} \frac{\Delta\beta_i}{4} \sum_{j=1}^{\mu} \left(x_i(\tau_j^\mu) - \alpha\right)\ \lambda_j^\mu\ \Pi_{p_2}(\eta_{ij}) \sum_{k=1}^{\nu} \lambda_k^\nu\ \Pi_{p_1}(\xi_{ijk})\ ,$$

where

$$\nu = \left\lceil \frac{p_1+1}{2} \right\rceil\ ,\ \ \mu = \begin{cases} \left\lceil \frac{p_1+p_2+1}{2} \right\rceil\ , & \Delta\alpha_i = 0 \\ \left\lceil \frac{p_1+p_2+2}{2} \right\rceil\ , & \Delta\alpha_i \neq 0 \end{cases} \tag{2.22}$$

$\lceil \cdot \rceil$ denoting the smallest not lower integer. Notice that, since we deal with a polynomial, we can take for example $\alpha = 0$ in (2.21) without problems. In the case of the classical monomial basis, $\{x^{p_1} y^{p_2}\}$, (2.21)-(2.22) provides an apparently new and curious formula for computing standard moments of a polygon.

REMARK 2.4. (the nodes location)

A possible drawback of the cubature formula (2.3) with respect to other techniques, like e.g. polygon triangulation followed by cubature over triangles, is

that the cubature nodes fall in general not only inside but also outside the polygon (in the enclosing rectangle $\mathcal{R} \supseteq \Omega$). This is the reason why f is assumed to be continuous and computable also in $\mathcal{R}$, and the error estimate (2.12) involves the best uniform polynomial approximation on $\mathcal{R}$.

However, this situation can be avoided by a simple change of variables, without partitioning the polygon into subpolygons, within a large class of polygons. This class is given by those for which there exists a "base-line" (say ℓ), whose intersection with the polygon is connected, and such that in addition each line orthogonal to it (say q) has a connected intersection (if any) with the polygon, containing the point $\ell \cap q$. Clearly such class contains all convex polygons, for example by taking the line connecting a pair of vertices with maximal distance (we omit the easy proof for brevity); see Figure 1-right. But it contains also nonconvex polygons, see Figure 2-right.

The change of variables in the integration consists simply in a rotation of the co-ordinate system such that the base-line becomes parallel to the (new) y-axis, and one chooses as α the abscissa of its intersection with the (new) x-axis. Our implementation of Gauss-like cubature over polygons accepts a pair of points, say $A = (x_A, y_A)$ and $B = (x_B, y_B)$, which define the base-line if the user can provide them, otherwise takes by default a pair of maximal distance vertices (which work well e.g. in the convex case). In practice, this entails only that in (2.6)-(2.8) α_i and β_i must be substituted by

$$\hat{\alpha}_i = \alpha_i \cos\phi + \beta_i \sin\phi \ , \quad \hat{\beta}_i = -\alpha_i \sin\phi + \beta_i \cos\phi \ , \tag{2.23}$$

$$\alpha = x_A \cos\phi + y_A \sin\phi \ , \tag{2.24}$$

where $\phi = \arccos\left(|y_B - y_A|/\|B - A\|_2\right)$, $0 \leq \phi \leq \pi/2$, is the rotation angle. Clearly, if all the cubature nodes fall inside the polygon, e.g. after the change of variables above when possible, in estimate (2.12) $E_{2n-1}(f; \mathcal{R})$ can be replaced by $E_{2n-1}(f; \Omega)$.

In Figures 1 and 2 we show two examples of polygons, one convex and the other nonconvex, each with two base-lines, the y-axis (left) or a selected diagonal (right), and the corresponding cubature nodes for $n = 10$: there are 660 points in Fig. 1 left and right, 960 points in Fig. 2-left and 990 points in Fig. 2-right (since in Fig. 2-left three sides are parallel to the base-line, cf. (2.5) and (2.9)). For the convex polygon the selected diagonal is the longest one, accordingly to the observation above. As one expects, in all cases the cubature nodes cluster at the base-line, at the sides and especially at the vertices.

REMARK 2.5. (equivalence with a decomposition into trapezoidal panels)

Using Green's formula in connection with a "base-line" has a simple geometric meaning, since it corresponds to decomposing the polygon into trapezoidal panels. Indeed, given the base-line and two neighbor vertices in the polygon, we may draw two lines through these two points orthogonal to the base-line. This creates a trapezoidal panel, with two noteworthy special cases, i.e. rectangular panels and triangular panels (when the side crosses the base-line we get even two triangles). Then we may approximate the integral over all such panels by a

product of Gauss rules, and finally sum up these contributions (taking into account the correct sign in overlapping cases). A special choice of the base-line (see the previous remark) can avoid overlapping and guarantee that all the cubature nodes are inside the domain.

The strength and elegance of Green's theorem, together with Gaussian discretization of a suitable primitive, allow to avoid performing explicitly the geometric decomposition into panels as well as analyzing the correct sign of each panel contribution, since both become intrinsic in the construction of the cubature formula.

REMARK 2.6. (reducing the number of function evaluations)

In many applications of cubature, for example when function evaluations are very costly, or when several different functions have to be integrated on the same domain, it is important to keep as low as possible the number of cubature nodes and thus of function evaluations at a given degree of exactness $2n-1$.

In the special case of rectangles, choosing as base-line one of the sides, the Gaussian-like formula (2.3) uses exactly n^2 nodes, since the only "active" side is parallel to the base-line. To this respect, in some sense (2.3) can be viewed also as an estension of tensor-product formulas to polygons. More generally, when a polygon has only mutually parallel or orthogonal sides, choosing as base-line one of the sides the number of nodes is at most $(L/2-1)n^2$, since then half of them are orthogonal to the base-line and one lies on it.

In the general case of sides which are "oblique" with respect to the base-line, formula (2.3) requires up to $Ln(n+1)$ nodes to obtain degree of exactness $2n-1$, due to the presence of the linear factor $x-\alpha$ in the discretized primitive (2.17). There is, however, an easy way to reduce the number of nodes still preserving the degree of exactness, which mimics the approach used for the construction of the so-called Stroud conical rules for a triangle (cf. [16, 25]). The price to be paid is an increase of the computational complexity for the construction of nodes and weights.

Indeed, composing the primitive (2.17) with the parametrization (2.14) of the i-th side we get a linear factor in t, namely $\Delta\beta_i(x_i(t)-\alpha)/4 = c_i t + d_i$. When the integrand f is a polynomial of degree $2n-1$, such a linear factor multiplies a polynomial of degree $2n-1$ in t. Taking the linear factor as weight function, we can then compute the nodes and weights of the corresponding n-point Gauss quadrature on $[-1,1]$, say $\{\hat\tau_{ij}^n\}$ and $\{\hat\lambda_{ij}^n\}$, $1 \le j \le n$. This can be done, side by side, using Gautschi's OPQ Matlab routine `chri1.m` for the recurrence coefficients of the orthogonal polynomials when the measure is modified by a linear factor, together with the OPQ routine `gauss.m` which computes nodes and weights from the recurrence coefficients; cf. [7, 8].

The resulting cubature formula is exactly like (2.3), where for indexes i corresponding to "oblique" sides (neither parallel nor orthogonal to the base-line) the nodes and weights are simply replaced by

$$\hat\xi_{ijk} = \frac{x_i(\hat\tau_{ij}^n)-\alpha}{2}\,\tau_k^n + \frac{x_i(\hat\tau_{ij}^n)+\alpha}{2}\,, \quad \hat\eta_{ij} = y_i(\hat\tau_{ij}^n)\,, \quad \hat w_{ijk} = \hat\lambda_{ij}^n\,\lambda_k^n\,, \tag{2.25}$$

$1 \le j \le n$, $1 \le k \le n$. The overall numer of nodes is then not bigger than Ln^2,

instead of $L(n^2+n)$, for a polygon with L sides.

This option has been incorporated in the numerical code, see [24]. Several numerical tests have shown that the modified cubature formula (2.25) has a comparable accuracy with respect to the basic formula (2.3), whereas the cost for the computation of nodes and weights increases roughly proportionally to the number of sides of the polygon (indeed, the cost of `gauss.m` dominates over that of `chri1.m`). Its use is recommended mainly when a very large number of costly function evaluations is expected.

3 Numerical results.

In this section we present several numerical tests of cubature by formula (2.3) over the two polygons in Figures 1 and 2, with points distributions generated at different degrees by the selected base-lines (see Remark 4). The cubature formula has been implemented by a Matlab code (cf. [24]), which needs in input only the sequence of polygon vertices (counterclockwise), the integrand function, and the parameter n (theoretical exactness at degree $2n-1$). In particular, Gauss-Legendre nodes and weights are computed by Gautschi's Matlab routines for orthogonal polynomials, see [8]. All the tests have been done by an Intel-Centrino Duo processor with 1 Gb RAM.

We have considered the following six bivariate test functions

$$f_1(x,y) = \frac{3}{4}\, e^{-\frac{1}{4}((9x-2)^2+(9y-2)^2)} + \frac{3}{4}\, e^{-\frac{1}{49}(9x+1)^2-\frac{1}{10}(9y+1)}$$

$$+\frac{1}{2}\, e^{-\frac{1}{4}((9x-7)^2+(9y-3)^2)} - \frac{1}{5}\, e^{-((9y-4)^2+(9y-7)^2)}\,,$$

$$f_2(x,y) = \sqrt{(x-0.5)^2+(y-0.5)^2}\,,\quad f_3(x,y) = (x+y)^{19}\,,$$

$$f_4(x,y) = e^{-((x-0.5)^2+(y-0.5)^2)}\,,\quad f_5(x,y) = e^{-100((x-0.5)^2+(y-0.5)^2)}\,,$$

$$f_6(x,y) = \cos\left(30(x+y)\right)\,, \tag{3.1}$$

which are in order the well-known Franke test function, the distance function from $(0.5, 0.5)$, a polynomial of degree 19, two Gaussians centered at $(0.5, 0.5)$ with different variance parameters, and a (moderately) oscillating function.

In Tables 1-4 we have displayed the relative cubature errors of the six test functions above, for a sequence of polynomial degrees with step 5 , $n = 5, \ldots, 30$ (the corresponding number of cubature nodes appears in the second row). The reference integrals have been computed by the Matlab `dblquad` function (adaptive cubature routine) for the integrand multiplied by the characteristic function of the domain (which can be implemented via the Matlab `inpolygon` function, cf. [17]). This works, however, only by a suitable splitting of the enclosing square into subsquares. In fact, the procedure applied directly to the whole enclosing square gives unreliable results: for example, even integrating the constant 1 on the domain of Fig. 2 with a tolerance of 1E-10 by `dblquad` (release 1.13), we get an error of about 1E-03 (with a CPU time of more than 2 minutes!)

The behavior of our Gauss-like cubature formula is quite satisfactory: for all the regular integrands, f_1, f_3, f_4, f_5 and f_6, the error at $n = 30$ is extremely small, and at least 4 correct figures are obtained even for f_2, which has a singularity of the gradient at a point "in the middle" of the integration domains (where the nodes cluster slowly).

As expected, for the polynomial f_3 and the oscillating function f_6 the error jumps down abruptly and then stabilizes as soon as the degree exceeds a threshold. Observe also that, again not surprisingly, with the nodes distribution on the right of Figures 1-2 where the nodes are all inside the domain, the cubature formula is more accurate.

Finally it is worth reporting the CPU times, which are very low. In all the examples above, they range from 0.02 up to 0.04 seconds. Moreover, we stress that our Gauss-like cubature formula over polygons works without problems even for degrees in the hundreds. For example, the less regular test function f_2 can be integrated on the domain of Fig. 2 at $n = 500$, that is with more than 2 million nodes giving 8 correct figures, in less than 2 seconds.

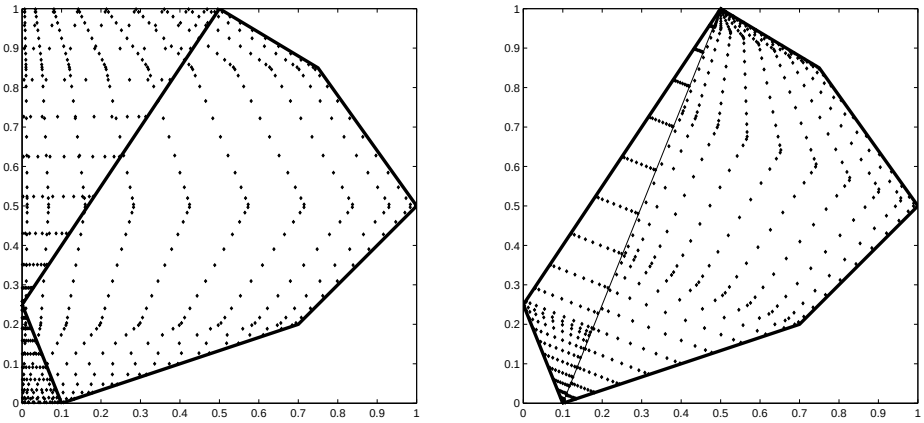

Figure 3.1: Two examples of distribution of cubature points ($n = 10$) for a convex domain.

Table 3.1: Relative cubature errors for the six test functions, with cubature points distributed as in Fig. 1-left.

function	$n = 5$ 180 pts	$n = 10$ 660 pts	$n = 15$ 1440 pts	$n = 20$ 2520 pts	$n = 25$ 3900 pts	$n = 30$ 5580 pts
f_1	3.2E-03	1.4E-05	1.7E-08	4.9E-12	4.1E-15	5.1E-15
f_2	6.7E-03	7.2E-04	3.0E-04	9.9E-05	6.9E-05	3.0E-05
f_3	2.5E-04	2.8E-15	1.7E-15	8.4E-16	1.8E-15	5.2E-15
f_4	4.3E-09	1.7E-15	1.1E-16	8.0E-16	1.6E-15	2.2E-15
f_5	4.2E-01	1.2E-02	8.6E-05	1.9E-07	1.7E-10	5.0E-14
f_6	1.2E-01	2.0E-01	1.4E-05	2.5E-11	1.3E-14	1.2E-15

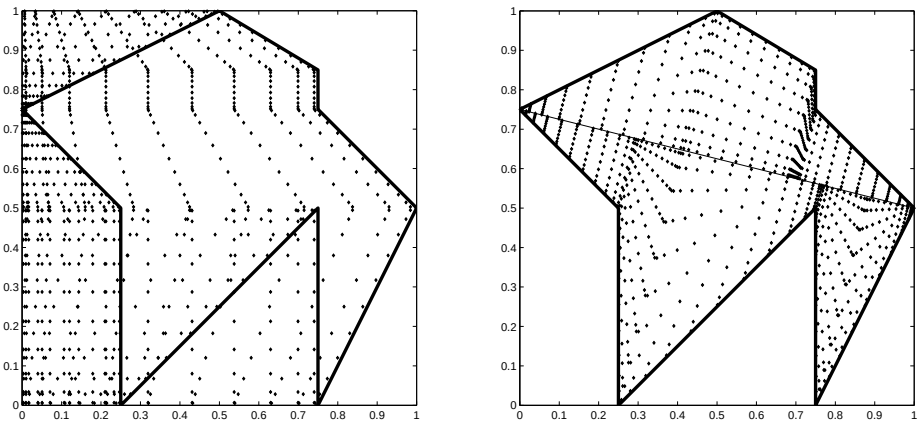

Figure 3.2: Two examples of distribution of cubature points ($n = 10$) for a nonconvex domain.

Table 3.2: Relative cubature errors for the six test functions, with cubature points distributed as in Fig. 1-right.

function	$n = 5$	$n = 10$	$n = 15$	$n = 20$	$n = 25$	$n = 30$
	180 pts	660 pts	1440 pts	2520 pts	3900 pts	5580 pts
f_1	5.7E-04	3.9E-06	3.9E-09	4.0E-13	3.8E-15	4.9E-15
f_2	1.0E-03	7.8E-05	3.3E-06	1.2E-06	1.5E-05	2.1E-06
f_3	2.1E-05	6.7E-16	3.3E-16	1.0E-15	8.4E-16	3.3E-15
f_4	2.3E-11	1.7E-15	8.0E-16	9.2E-16	1.5E-15	2.5E-15
f_5	4.6E-02	2.2E-05	6.7E-09	4.8E-12	1.5E-14	8.6E-15
f_6	2.9E+00	2.9E-01	2.7E-05	5.3E-11	1.9E-15	4.9E-15

Table 3.3: Relative cubature errors for the six test functions, with cubature points distributed as in Fig. 2-left.

function	$n = 5$	$n = 10$	$n = 15$	$n = 20$	$n = 25$	$n = 30$
	255 pts	960 pts	2115 pts	3720 pts	5775 pts	8280 pts
f_1	2.2E-04	6.8E-06	2.0E-09	5.3E-13	2.1E-15	3.3E-15
f_2	8.8E-03	7.3E-04	4.0E-04	1.0E-04	9.0E-05	3.2E-05
f_3	2.7E-04	8.5E-15	8.3E-15	6.3E-15	8.5E-15	1.7E-15
f_4	5.5E-09	2.2E-15	1.1E-15	8.9E-16	1.8E-15	2.4E-15
f_5	5.2E-01	1.4E-02	1.1E-04	2.5E-07	2.1E-10	6.9E-14
f_6	4.0E+00	1.1E-02	3.6E-08	8.5E-14	7.4E-14	7.7E-14

Table 3.4: Relative cubature errors for the six test functions, with cubature points distributed as in Fig. 2-right.

function	$n=5$ 270 pts	$n=10$ 990 pts	$n=15$ 2160 pts	$n=20$ 3780 pts	$n=25$ 5850 pts	$n=30$ 8370 pts
f_1	4.3E-05	9.3E-10	5.0E-14	1.9E-15	3.0E-15	3.6E-15
f_2	2.8E-04	5.4E-05	1.5E-05	5.2E-06	3.9E-07	1.6E-06
f_3	8.5E-07	6.7E-15	7.6E-15	6.3E-15	6.5E-15	3.3E-15
f_4	9.4E-12	2.2E-15	1.4E-16	1.4E-15	2.2E-15	2.4E-15
f_5	6.9E-03	7.2E-06	1.1E-09	6.4E-14	7.9E-15	7.8E-15
f_6	1.3E+00	6.6E-04	3.6E-09	7.8E-14	7.5E-14	7.8E-14

Acknowledgement.

We would like to thank an anonymous referee for having improved the presentation of the paper and the numerical code.

REFERENCES

1. T.M. Apostol, *Calculus*, vol. II, 2nd edition, Blaisdell, 1969.
2. J. Berntsen and T.O. Espelid, *Algorithm 706: DCUTRI: An algorithm for adaptive cubature over a collection of triangles*, ACM Trans. Math. Softw. 18 (1992), no.3, 329–342.
3. R. Cools, *An encyclopaedia of cubature formulas*, Numerical integration and its complexity (Oberwolfach, 2001), J. Complexity 19 (2003), no. 3, 445–453.
4. A.R. DiDonato and R.K. Hageman, *A method for computing the integral of the bivariate normal distribution over an arbitrary polygon*, SIAM J. Sci. Statist. Comput. 3 (1982), no. 4, 434–446.
5. C.F. Dunkl, Orthogonal polynomials on the hexagon, SIAM J. Appl. Math. 47 (1987), no. 2, 343–351.
6. C.F. Dunkl and Y. Xu, *Orthogonal Polynomials of Several Variables*, Encyclopedia of Mathematics and its Applications 81, Cambridge University Press, Cambridge, 2001.
7. W. Gautschi, *Orthogonal polynomials: computation and approximation*, Numerical Mathematics and Scientific Computation, Oxford Science Publications, Oxford University Press, New York, 2004 (software available at `http://www.cs.purdue.edu/archives/2002/wxg/codes`).
8. W. Gautschi, *Orthogonal polynomials (in Matlab)*, J. Comput. Appl. Math. 178 (2005), 215–234.
9. G.H. Golub and C.F. Van Loan, *Matrix computations*, third edition, Johns Hopkins Studies in the Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, 1996.
10. G. Green, *An Essay on the Application of Mathematical Analysis to the Theories of Electricity and Magnetism*, Nottingham, 1828.

11. A. Guessab, *On the approximate calculation of integrals on a polygon in R^2*, Numerical methods and approximation theory, III (Niš, 1987), 225–239, Univ. Niš, Niš, 1988.

12. A.I. Ivanova, *Certain cases of L. A. Lyusternik's cubature formula for regular polygons* (Russian), Vyčisl. Mat. Vyčisl. Tehn. 1, (1953). 27–36.

13. M. Held, *FIST: Fast Industrial-Strength Triangulation of Polygons*, Algorithmica 30 (2001), no. 4, 563–596.

14. V.I. Krylov, *Approximate Calculation of Integrals*, The Macmillan Co., New York-London, 1962.

15. J.A. Liggett, *Exact formulae for areas, volumes and moments of polygons and polyhedra*, Comm. Appl. Numer. Methods 4 (1988), no. 6, 815–820.

16. J.N. Lyness and R. Cools, *A survey of numerical cubature over triangles*, Mathematics of Computation 1943–1993: a half-century of computational mathematics (Vancouver, BC, 1993), 127–150, Proc. Sympos. Appl. Math., 48, Amer. Math. Soc., Providence, RI, 1994.

17. The MathWorks, *MATLAB documentation set*, 2006 version (available online at `http://www.mathworks.com`).

18. A. Narkhede and D. Manocha, *Graphics Gems 5*, Editor: Alan Paeth, Academic Press, 1995.

19. M. Nooijen, G. te Velde and E.J. Baerends, *Symmetric numerical integration formulas for regular polygons*, SIAM J. Numer. Anal. 27 (1990), no. 1, 198–218.

20. W. Pleśniak, *Remarks on Jackson's theorem in $\mathbb{R}^N$*, East J. Approx. 2 (1996), no. 3, 301–308.

21. H.T. Rathod and H.S. Govinda Rao, *Integration of trivariate polynomials over linear polyhedra in Euclidean three-dimensional space*, J. Austral. Math. Soc. Ser. B 39 (1998), no. 3, 355–385.

22. A. Sommariva and M. Vianello, *Numerical cubature on scattered data by radial basis functions*, Computing 76 (2006), 295–310.

23. A. Sommariva and M. Vianello, *Meshless cubature by Green's formula*, Appl. Math. Comput. 183 (2006), 1098–1107.

24. A. Sommariva and M. Vianello, *Polygauss: Matlab code for Gauss-like cubature over polygons*, software downloadable from: `www.math.unipd.it/~marcov/software.html`.

25. A.H. Stroud, *Approximate calculation of multiple integrals*, Prentice-Hall Series in Automatic Computation, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1971.