STUDIA UNIV. "BABEŞ–BOLYAI", MATHEMATICA, Volume **LI**, Number 4, December 2006

THE STUDY OF AN ADAPTIVE ALGORITHM FOR SOME CUBATURE FORMULAS ON TRIANGLE

ILDIKÓ SOMOGYI AND RADU TRÎMBIŢAŞ

Dedicated to Professor Gheorghe Coman at his 70^{th} *anniversary*

Abstract. We study two nonproduct quadrature formulas of algebraic degree 2 and 3, respectively. The second is then turned into an adaptive quadrature algorithm. A MATLAB implementation and some examples are given.

1. The formulas

The purpose of this paper is to give some practical cubature formulas when the integration domain is a triangle and also to study an adaptive algorithm for these cubature formulas in approximation of the integral

$$I = \int_{T_h} f(x,y)\, dxdy, \tag{1.1}$$

where T_h is a triangular domain, $T_h = \{(x,y)/x \geq 0, y \geq 0, x+y \leq h\}$, and $f : T_h \to \mathbb{R}$ is an integrable function on T_h. We shall consider two cubature formulas from [8]. One of them is a cubature formula which satisfy the minimal condition of Stroud regarding the minimal number of knots of a cubature formula [9]. The degree of exactness of this formula is equal to 2. The other one is a cubature formula which has more knots, but a greater degree of exactness. We consider the following practical cubature formula:

$$\int_{T_h} \int f(x,y)dxdy = \frac{h^2}{6}\left[f(0,\frac{h}{2}) + f(\frac{h}{2},0) + f(\frac{h}{2},\frac{h}{2})\right] + R_2(f). \tag{1.2}$$

Received by the editors: 08.06.2006.

2000 *Mathematics Subject Classification.* 65D30, 65D32.

Key words and phrases. numerical integration, adaptive cubature, MATLAB.

The degree of exactness of this formula is 2, therefore we can use the Peano theorem for the representation of the error, an we can give the following delimitation of the approximation error:

Theorem 1. *If* $f^{(3,0)}(\cdot,0) \in C[0,h], f^{(2,1)}(\cdot,0) \in C[0,h], f^{(0,3)}(0,\cdot) \in C[0,h]$ *and* $f^{(1,2)}(s,t) \in C(T_h)$ *than we have*

$$|R_2(f)| \leq M_{30}f\frac{h^5}{720} + M_{21}f\frac{h^5}{364} + M_{03}f\frac{h^5}{720} + M_{12}f\frac{h^5}{24} \tag{1.3}$$

where

$$M_{30}f = \max_{s\in[0,h]}\left|f^{(3,0)}(s,0)\right|, M_{21}f = \max_{s\in[0,h]}\left|f^{(2,1)}(s,0)\right|,$$
$$M_{03}f = \max_{t\in[0,h]}\left|f^{(0,3)}(0,t)\right|, M_{12}f = \max_{T_h}\left|f^{(1,2)}(s,t)\right|.$$

Remark 1. *The cubature formula (2) has an optimal character, because it satisfies the condition established by Stroud in* [9] *regarding the minimal number of knots for a cubature formula. If the degree of exactness of a cubature formula is equal to* 2*, then the minimal number of knots is* $N = n + 1$*, where* n *is the dimension number. The cubature formula (1.2) with the degree of exactness* 2 *and three knots, has a minimal number of knots.*

Let us now consider a cubature formula with a higher degree of exactness:

$$I = \int_{T_h}\int f(x,y)dxdy = \frac{h^2}{120}\left[3f(0,0) + 3f(h,0) + 3f(0,h) + 8f(\frac{h}{2},0)+\right.$$
$$\left.+8f(\frac{h}{2},\frac{h}{2}) + 8f(0,\frac{h}{2}) + 27f(\frac{h}{3},\frac{h}{3})\right] + R_3(f). \tag{1.4}$$

Because the degree of exactness of this formula is equal to 3, we can give the following theorem for the delimitation of the absolute error:

Theorem 2. *If* $f^{(4,0)}(s,0) \in C[0,h]$, $f^{(3,1)}(s,0) \in C[0,h]$, $f^{(0,4)}(0,t) \in C[0,h]$, $f^{(1,3)}(0,t) \in C[0,h]$, *and* $f^{(2,2)}(s,t) \in C(T_h)$ *then*

$$|R_3(f)| \leq M_{40}f\frac{h^6}{8640} + M_{31}f\frac{7h^6}{1440} + M_{13}f\frac{7h^6}{1440} + M_{04}f\frac{h^6}{8640} + M_{22}f\frac{h^6}{768},$$

where

$$M_{40}f = \max_{s\in[0,h]} \left|f^{(4,0)}(s,0)\right|, M_{31}f = \max_{s\in[0,h]} \left|f^{(3,1)}(s,0)\right|,$$
$$M_{13}f = \max_{t\in[0,h]} \left|f^{(1,3)}(0,t)\right|, M_{04}f = \max_{t\in[0,h]} \left|f^{(0,4)}(0,t)\right|,$$
$$M_{22}f = \max_{(s,t)\in T_h} \left|f^{(2,2)}(s,t)\right|.$$

We shall use an affine transformation to transform these cubature formulas from the standard triangle T_h to an arbitrary triangle Δ with the vertices $V_i(x_i, y_i), i = 1, 2, 3$.

Let $\varphi : \mathbb{R}^2 \to \mathbb{R}^2$ denote the affine transformation from T_h to Δ,

$$\varphi(\overline{x}, \overline{y}) = A(\overline{x}, \overline{y}) + b \tag{1.5}$$

where

$$A = \begin{pmatrix} \dfrac{x_2 - x_1}{h} & \dfrac{x_3 - x_1}{h} \\ \dfrac{y_2 - y_1}{h} & \dfrac{y_3 - y_1}{h} \end{pmatrix} \tag{1.6}$$

and

$$b = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \tag{1.7}$$

Let J be the Jacobian matrix of the transformation, in this case J is independent of $(\overline{x}, \overline{y})$, $\det J(\overline{x}, \overline{y}) = \det A$, and the transformation rule is

$$\int_{T_h} f(x,y)dxdy = |\det A| \int_{\Delta} f(\varphi(\overline{x}, \overline{y}))d\overline{x}d\overline{y}.$$

2. **Implementation**

For a detailed description of an adaptive numerical integration algorithm see [10, 7].

In this section we focus our attention to an adaptive algorithm, based on formula (1.4). For implementation details on an adaptive algorithm based on formula (1.2), see [3].

Now, using the transform given by (1.5), (1.6), and (1.7), we rewrite (1.4) in the form

$$I \approx \frac{\text{area}(\Delta)}{60}\left(3\sum_{i=1}^{3} f(V_i) + 8\sum_{i=1}^{3} f(P_i) + 27f(G)\right), \tag{2.1}$$

where

$$P_i = \frac{1}{2}(P_i + P_j), \qquad \{i,j,k\} = \{1,2,3\},$$

are the midpoints of the edges of Δ, and $G = \frac{1}{3}(V_1 + V_2 + V_3)$ is the barycenter of Δ (see Figure 1).

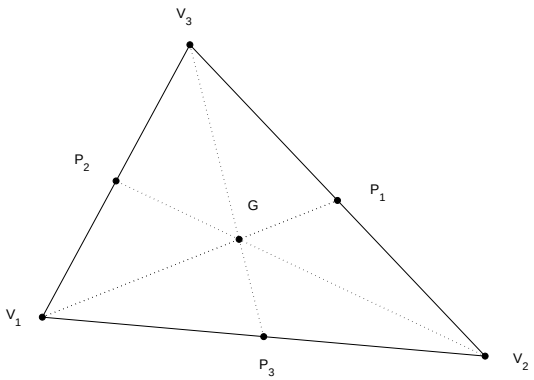

FIGURE 1. The elementary third degree formula

The intial triangle, Δ, is decomposed into four triangles, Δ_1, Δ_2, Δ_3, and Δ_4, determined by verices and the middle points (see Figure 2).

In the first step we apply the formula given by (2.1) to Δ. Then we apply the same formula to each of the triangles Δ_i, $i = 1, \ldots, 4$. Let I_1 be the value provided by (2.1), and I_2 the value obtained by summing the four valued obtained applying (2.1) to each of the four triangle of the subdivision. A possible stopping criterion is

$$|I_1 - I_2| < \varepsilon,$$

where ε is the desired tolerance. If the criterion is not fulfilled, then we apply the same procedure recursively to each triangle of the subdivision. A detailed description is given in Algorithm 1.

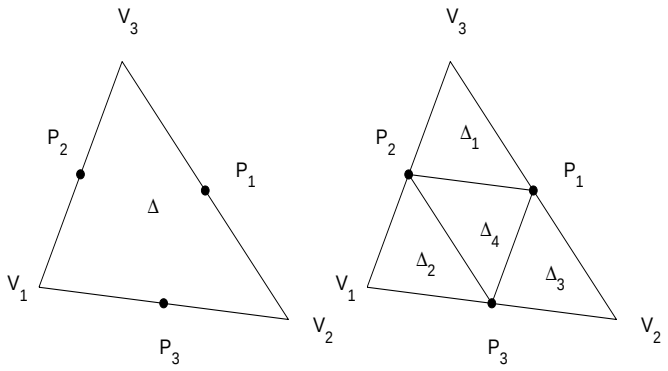

FIGURE 2. The initial triangle and the subdivision

Algorithm 1 An adaptive cubature algorithm on triangle; call *result* := adapt(f, Δ, ε), where f is the integrand, Δ is the triangle, and ε is the desired tolerance; elem_formula implemntes the elementary cubature given by (2.1).

Let Δ_1, Δ_2, Δ_3, Δ_4 be the triangles determined by vertices and midpoints;
$I1 :=$ elem_formula(f, Δ);
$I2 :=$ elem_formula(f, Δ_1) + elem_formula(f, Δ_2)+
elem_formula(f, Δ_3) + elem_formula(f, Δ_4);
if $|I_1 - I_2| < \varepsilon$ **then**
result $:= I_2$;
else
result := adapt$(f, \Delta_1, \varepsilon)$ + adapt$(f, \Delta_2, \varepsilon)$+
adapt$(f, \Delta_3, \varepsilon)$ + adapt$(f, \Delta_4, \varepsilon)$;
end if

The papers [6, 5] give useful guidelines for implementation of adaptive cubatures on triangle. We have implemented this algorithm in MATLAB[1]. The implementation follows the description given by Algorithm 1. The optional input parameter `trace`, when it is nonzero, allows us to represent graphically the process of computing. The optional output parameter `stat` gives us the number of function evaluation and

[1] MATLAB© is a trademark of the MathWorks Inc., Natick, MA 01760-2098

the number of triangles. Some optimizations which save several function evaluations are possible. Since the value I_1 is the value of the integral on Δ, we compute it once and provide it further as an input parameter. We do the same thing with the values of function at midpoints and barycenter.

In the sequel we give the MATLAB code.

```
function [vi,stat]=mpcubatd3mb(f,V,err,trace)
%MPCUBATD3MB - cubature with midpoints and barycenter,
% exact for P_3^2
%call [vi,stat]=mpcubatd3mb(f,V,err,trace,...)
%f function
%V - coordinates of vertices
%err - error
%trace - tracing indicator

global FEN TRIN sfl
if nargin <4, trace=0;
else
    if trace
        clf
    end
end
if nargin < 3, err=1e-3; end
sfl=[nargout==2];
if sfl
    FEN=0; TRIN=0;
end
P=midpoints(V); G=sum(V,2)/3;
fv=feval(f,V); fp=feval(f,P); fg=feval(f,G);
area = 1/2*abs(det([V',ones(3,1)]));
```

```
I1=elform(area,fv,fp,fg);
if trace, tracefun([V,P,G]); hold on; end
vi=quadrg3(f,V,P,G,fv,fp,fg,err,area,I1,trace);
if sfl
    FEN = FEN+7;
    stat=struct('nev',FEN,'ntri',TRIN);
end

function vi=quadrg3(f,V,P,G,fv,fp,fg,err,area,I1,trace)
%QUADRG3 - cubature with midpoints and barycenter, internal use
%call vi=quadrg3(f,V,P,G,fv,fp,fg,err,area,I1,trace)
%f - function
%V - coordinates of vertices
%P - midpoints coordinates
%G - barycenter coordinates
%fv - values of f at verices
%fp - values of f at midpoints
%fg - values at barycenter
%err - error
%area - area of triangle
%I1 - the first estimation (elementary formula)
%trace - tracing indicator

global FEN TRIN sfl
area=area/4;
V1=[V(:,1),P(:,[2,3])]; fv1=[fv(1),fp([2,3])];
P1=midpoints(V1); fp1=feval(f,P1);
G1=sum(V1,2)/3; fg1=feval(f,G1);
I11=elform(area,fv1,fp1,fg1);
V2=[V(:,2),P(:,[1,3])]; fv2=[fv(2),fp([1,3])];
```

```
P2=midpoints(V2); fp2=feval(f,P2);
G2=sum(V2,2)/3;   fg2=feval(f,G2);
I12=elform(area,fv2,fp2,fg2);
V3=[V(:,3),P(:,[1,2])]; fv3=[fv(3),fp([1,2])];
P3=midpoints(V3); fp3=feval(f,P3);
G3=sum(V3,2)/3; fg3=feval(f,G3);
I13=elform(area,fv3,fp3,fg3);
V4=P; fv4=fp; P4=[P1(:,1),P2(:,1),P3(:,1)];
fp4=[fp1(1),fp2(1),fp3(1)]; G4=G; fg4=fg;
I14=elform(area,fv4,fp4,fg4);
I2=I11+I12+I13+I14;
if sfl
    FEN=FEN+12;
    TRIN=TRIN+4;
end
if trace, tracefun([P1,P2,P3,G1,G2,G3]); end
if abs(I2-I1)<err
    vi=I2;
else
    vi=quadrg3(f,V1,P1,G1,fv1,fp1,fg1,err,area,I11,trace)+...
        quadrg3(f,V2,P2,G2,fv2,fp2,fg2,err,area,I12,trace)+...
        quadrg3(f,V3,P3,G3,fv3,fp3,fg3,err,area,I13,trace)+...
        quadrg3(f,V4,P4,G4,fv4,fp4,fg4,err,area,I14,trace);
end %if

function v=elform(area,fv,fp,fg)
v=area/60*(3*sum(fv)+8*sum(fp)+27*fg);

function P=midpoints(V);
P(:,1)=(V(:,2)+V(:,3))/2; P(:,2)=(V(:,1)+V(:,3))/2;
```

```
P(:,3)=(V(:,1)+V(:,2))/2;


function tracefun(L)
%TRACEFUN - represents points where function is evaluated
plot(L(1,:),L(2,:),'.k','Markersize',4);
```

3. Numerical examples

Consider the triangle $T_1 = \{(x,y)/x \geq 0, y \geq 0, x+y \leq 1\}$, and the function $f : T_1 \to \mathbb{R}$, $f(x,y) = \text{humps}(x)\text{humps}(y)$, where

$$\text{humps}(x) = \frac{1}{(x-0.3)^2+0.01} + \frac{1}{(x-0.9)^2+0.04} - 6.$$

The graph of f is given in Figure 3 as surface and as contour. First, we approximate

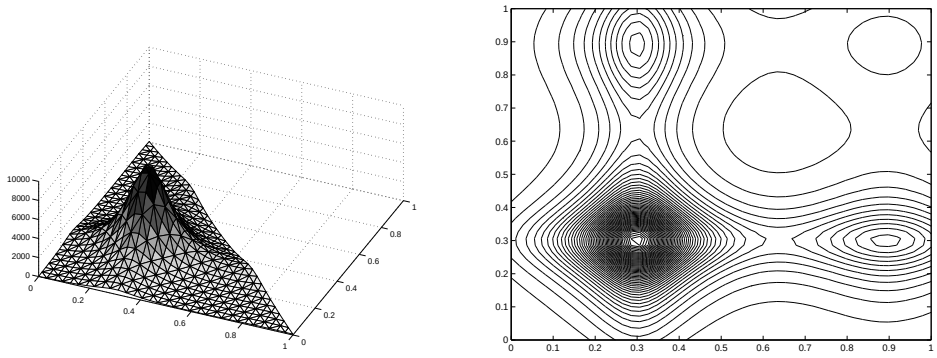

FIGURE 3. The graph of f, as surface (left) and as contour

the integral for a tolerance $\varepsilon = 10^{-5}$ using the adaptive quadrature based on (1.2) and (2.1), respectively. The `trace` flag is set.

```
>> [vib,statb]=mpcubatd2mb(@humps2dv,V,1e-5,1)
vib =
    5.997039610414015e+002
statb =
     nev: 57684
    ntri: 25636
```

```
>> [vib3,statb3]=mpcubatd3mb(@humps2dv,V,1e-5,1)
vib3 =
    5.997039668483903e+002
statb3 =
     nev: 30499
    ntri: 10164
```

The figure 4 shows the points where the MATLAB functions evaluate the integrands. Now, for a higher accuracy (10^{-9}) and a timer included we got the following results

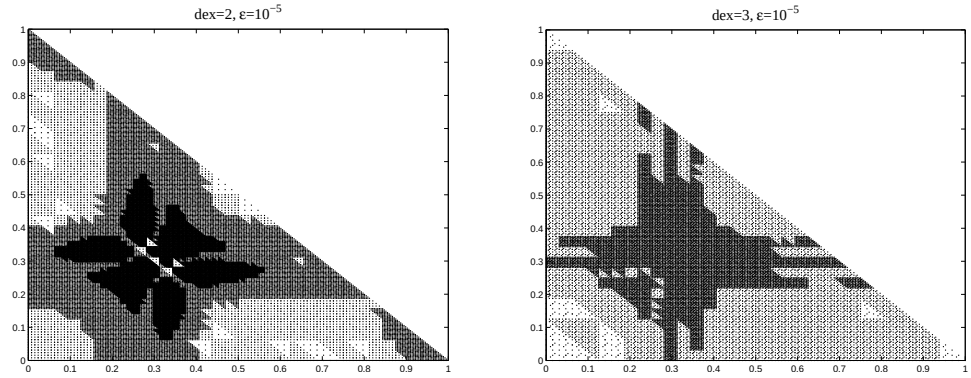

FIGURE 4. Evaluation points for the second order formula (left) and for the third order formula, for function f, $\varepsilon = 10^{-5}$

```
>> tic,[vib,statb]=mpcubatd2mb(@humps2dv,V,1e-9); toc
Elapsed time is 43.444590 seconds.
>> tic,[vib3,statb3]=mpcubatd3mb(@humps2dv,V,1e-9);toc
Elapsed time is 19.121086 seconds.
>> vib,statb
vib =
    5.997039625857019e+002
statb =
     nev: 2219736
    ntri: 986548
```

```
>> vib3,statb3
vib3 =
    5.997039625817022e+002
statb3 =
     nev: 640915
    ntri: 213636
```

Thus, for this function the third degree formula is faster.

For functions that do not exhibit high oscillations and for modest accuracy requirements, the second degree formula requires fewer function evaluation. Consider the function $f(x,y) = y \sin x$ (implemented by MATLAB function `fintegrv`) to be integrated on T_1. For $\varepsilon = 10^{-4}$, one obtains:

```
>> tic,[vib,statb]=mpcubatd2mb(@fintegrv,V,1e-4); toc
Elapsed time is 0.001425 seconds.
>> tic,[vib3,statb3]=mpcubatd3mb(@fintegrv,V,1e-4); toc
Elapsed time is 0.012022 seconds.
>> vib,statb
vib =
   0.04030110314738
statb =
     nev: 48
    ntri: 20
>> vib3,statb3
vib3 =
   0.04030317282902
statb3 =
     nev: 67
    ntri: 20
```

References

[1] Barnhill, R.E., Gordon, W.J., and Thomas, D.H., *The method of successive decomposition for multiple integration*, Research Rep. GMR-1281, General Motors, Warren, Mich., 1972.

[2] Coman, Gh., *Analiza numerică*, Libris, Cluj, 1995.

[3] Coman, Gh., Pop, I., Trîmbiţaş, R., *An adaptive cubature on triangle*, Studia UBB, Mathematica, vol. XLVII, No.4, pp. 27-36, 2002.

[4] Coman, Gh., Stancu, D.D., Blaga, P., *Analiză numerică şi teoria aproximării*, vol II, Presa Universitară Clujeană, Cluj-Napoca, 2002.

[5] Cools, R., Laurie, D.P., Pluym, L., *Cubpack++: A C++ package for Authomatic Two-dimensional Cubature*, ACM TOMS, vol. 23, No. 1, 1997, pp. 1-15.

[6] Laurie, D.P., *Algorithm 584: CUBTRI: Automatic Cubature over a triangle*, ACM TOMS, vol. 8, No. 2, 1982, 210-218.

[7] Rice, J., *A metaalgorithm for adaptive quadrature*, JACM, vol. 22, No. 1, 1975, 61-82.

[8] Somogyi, I., *Practical cubature formulas in triangles with error bounds*, Seminar on Numerical and Statistical Calculus, 2004, 131-137.

[9] Stroud, A.H., *Approximate Calculation of Multiple Integrals*, Englewood Cliffs, N.J. Prentice-Hall, Inc. 1971.

[10] Überhuber, Cr., *Computer Numerik*, Band II, Springer, 1995.

[11] Welfert, B., *Numerical Analysis*, Lecture Notes, University of Arizona.

Babeş-Bolyai University,
Faculty of Mathematics and Computer Science,
Str. Kogălniceanu Nr. 1, RO-400084 Cluj-Napoca, Romania
E-mail address: tradu@math.ubbcluj.ro