Mathematica Balkanica

Mathematical Society of South-Eastern Europe
A quarterly published by
the Bulgarian Academy of Sciences – National Committee for Mathematics

---

Mathematica
Balkanica

New Series Vol.10, 1996, Fasc.2-3

Contouring of Interpolating Surfaces [1]

Ljubiša M. Kocić , Dušan M. Milošević

Presented by Bl. Sendov

An algorithm for generating a map of level-lines of the polynomial interpolants is presented. It is based on so called tracing algorithm developed earlier by the authors, and described here shortly.

1. Introduction

Let $z = f(x,y)$ be a function of two variables defined on $D \subset \mathbb{R}^2$ and $G = \{(x,y,f(x,y)),\ (x,y) \in D\}$ be its graph. The contour G (or f) means to define a bijective map $\Phi : c \to L$ of the sequence $c = \{c_i\}_{i=1}^m$ where $\min_D f \leq c_i \leq \max_D f$ to the set of level-lines $L = \{L_i\}_{i=1}^m$ where

$$L_i = \{(x,y) \in D \mid f(x,y) = c_i\}.$$

If the function f is given by analytic expression then many methods of contouring are known. But if f is a function defined through some algorithm then the problem of contouring is not so easy. This problem rises if one wants to visualize the surface defined through some discrete data, or through some constructive algorithm, like in the case of free-form curves for purposes of geometric modeling. Some methods in use [1], [18], [19] are based on subdivision which gives essentially slow algorithms. Others, [21], [9], make use of rational curves and are limited to the second degree surfaces.

The aim of this paper is to use tracing algorithm, developed in previous papers [11], [12], [15] for increasing the speed and improve the accuracy of contouring.

[1]This research was partly supported by Science Fund of Serbia, grant number 0401A, through Matematički institut

2. Algorithm

The problem is to trace the curve L given by

$$F(x,y)=0, \quad (x,y)\in G,$$

where $G\subseteq \mathbb{R}^2$. By differentiating this formula one gets

$$F'_x dx + F'_y dy = 0, \quad (x_0,y_0)\in G, \tag{1}$$

which is the basis of the algorithm. Being a two-stage process, this algorithm is an improved version of the algorithms given in [10] and [20].

Stage I (Searching for the starting points).

First of all, a rectangular domain $D=[a,b]\times[c,d]$ $(a<b, c<d)$, such that $G\subseteq D$ have to be determined. Then, using the *modified Regula falsi* method, the following two sets of equations are solved

$$F\left(a+\frac{b-a}{N_x}i, y\right) = 0, \quad i=0,\cdots,N_x \quad y\in[c,d], \tag{2}$$

$$F\left(x, c+\frac{d-c}{N_y}j\right) = 0, \quad j=0,\cdots,N_y \quad x\in[a,b], \tag{3}$$

where N_x and N_y are natural numbers, defining the degree of subdivision along the y- and x-axis respectively. If the solution is not found, the values of N_x and N_y have to be increased. If there exists (i,j) in (2) and (3) such that $F(a+\frac{b-a}{N_x}i, c+\frac{d-c}{N_y}j)=0$, then the point $(x_0,y_0)=(a+\frac{b-a}{N_x}i, c+\frac{d-c}{N_y}j)$ will be found twice, in the process of solving (2) and then (3). So, the union of two sets of points, derived from (2) and (3) is made.

Stage II (Curve tracing)

In this stage, the algorithm calculates the set of points $\{(x_0,y_0),(x_1,y_1),\cdots (x_M,y_M)\}$ that are the vertices of the polygonal line which approximates the curve L. So, the algorithm is iterative, and in i-th iteration it calculates the point (x_i,y_i), and has M iterations. This number can be limited by three factors: 1. Computer's memory, where all the vertices of the polygonal line have to be stored; 2. Closeness to the starting point; 3. Passing the bounds of the rectangle D.

First of all, the algorithm checks if the current point, $p_i=(x_i,y_i)$ is the singular point, i.e. if

$$|F'_x(x_i,y_i)| + |F'_y(x_i,y_i)| < \epsilon_2, \tag{4}$$

where ϵ_2 is small enough. The logical value of (4) is the main switch in the second stage of the algorithm. If (4) is true, then, the new point $p_{i+1} = (x_{i+1}, y_{i+1})$ is found by the linear *extrapolation* based on the previous point $p_{i-1} = (x_{i-1}, y_{i-1})$, such that

$$p_{i+1} - 2p_i + p_{i-1} = 0.$$

Here a problem arose for the case $i = 0$, i.e. when $p_0 = (x_0, y_0)$ is a singularity of the curve. Then,

$$x_{-1} = x_0 \pm h, \qquad y_{-1} = y_0 \pm h, \tag{5}$$

have to be taken, where $h > 0$ is the prescribed tracing step. The signs in (5) are chosen arbitrarily if $p_0 \in intD$. If p_0 is on the border of D, $p_0 \in \partial D$, then the signs are adjusted so that p_{-1} will place outside the domain D.

If (4) is not true, the point p_i is not singular, and then $F'_x(x_i, y_i) \neq 0$ or $F'_y(x_i, y_i) \neq 0$, so the algorithm determines the next point, p_{i+1} by solving the initial value problem (1), using the procedure given in [**20**].

The point p_{i+1} is evaluated by a predictor-corrector procedure. In the predictor phase, according to the sign of

$$\sigma_i = |F'_x(x_i, y_i)| - |F'_y(x_i, y_i)|,$$

we use the iterations

$$\begin{aligned} x_{i+1} &= x_i + S_x h(1 - H(\sigma_i)) - S_y h \frac{F'_y(x_i,y_i)}{F'_x(x_i,y_i)} H(\sigma_i), \\ y_{i+1} &= y_i - S_x h \frac{F'_x(x_i,y_i)}{F'_y(x_i,y_i)}(1 - H(\sigma_i)) + S_y h H(\sigma_i), \end{aligned}$$

where $H(t)$, $t \in \mathbb{R}$ is the Heaviside function, $H(t) = 0$, $t < 0$; $H(t) = 1$, $t \geq 0$.

What we want to underline is the automation of choosing the signs S_x and S_y at the initial point $p_0 = (x_0, y_0)$. If $p_0 \in intD$, S_x and S_y are set arbitrarily. If $p_0 \in \partial D$, then $S_x = S_y = 1$ and if $p_1 \notin D$ then, $S_x = S_y = -1$.

In the corrector phase, p_{i+1} is corrected using the Newton-Raphson method:

$$y_{i+1} = y_i - \frac{F(x_i, y_i)}{F'_y(x_i, y_i)}(1 - H(\sigma_i)), \quad x_{i+1} = x_i - \frac{F(x_i, y_i)}{F'_x(x_i, y_i)} H(\sigma_i),$$

The stopping criteria of this correction is

$$|F(x_i, y_i)| < \epsilon_3.$$

This completes the algorithm. Let us recall what are input/output information:

The *input* information:

- $F(x,y), F_x(x,y), F_y(x,y)$ – the function and its derivatives;
- Rectangular area $[a,b] \times [c,d] = D$, in which the graph of $F(x,y) = 0$ is supposed to reside;
- $h > 0$, the step for the initial value solver;
- ϵ_1, ϵ_2 and ϵ_3 – small values numbers, respectively defining the neighborhood of the starting point, closeness of the singular point, and the stopping criteria.

The *output* information:

- The polygonal approximation $\{(x_i, y_i)\}_{i=0}^m$ of the curve $L : F(x,y) = 0, (x,y) \in D$, i.e. the vertices of the polygonal line $\{(x_i, y_i)\}_{i=0}^m$.
- The *tracing error* $E(s)$, defined as the error versus the arc length

$$E(s) = |F(x,y)|,$$

where s is the arc length from the beginning point p_0 to the current point $p = (x,y)$. Numerically, s is approximated by the polygonal line $\{(x_0, y_0), \ldots, (x_i, y_i)\}$ $(0 \leq i \leq M)$ where (x_i, y_i) is the current point.

Note that vertices of the polygonal line that approximates our curve satisfy

$$\max(x_{i+1} - x_i,\ y_{i+1} - y_i) \leq h,$$

i.e. the i-th segment of the polygonal approximation is contained in the square $[x_i, x_i + h] \times [y_i, y_i + h]$.

3. Regular data interpolation

Suppose f_{ij} are numerical values read off the function $f(x,y)$ defined on a rectangular domain D, so that $f_{ij} = f(x_i, y_i)$. In this sense, (x_i, y_i) is a node of a regular rectangular mesh. The task is to visualize the surface obtained by interpolation to data $\{f_{ij}\}$.

As an interpolating tool we choose Hermite bicubic polynomial which is defined by 16 data, four par each vertex of the mesh rectangle. Namely, for

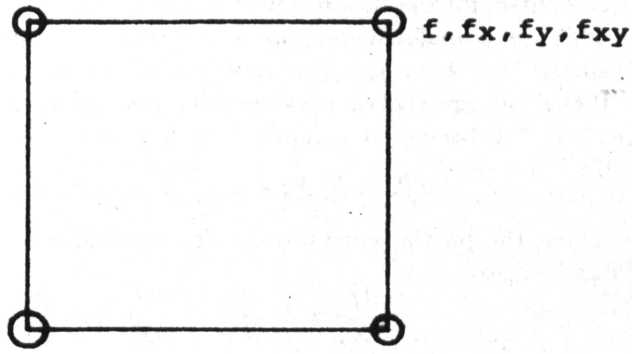

Figure 1: The Hermite rectangle

each vertex we need the value of function f, its first derivatives f_x, f_y and the second derivative f_{xy}, as it is shown in Figure 1.

The resulting bicubic Hermite patch is defined as a span of the bicubic polynomial basis

$$\{x^i y^j \mid 0 \leq i \leq 3,\ 0 \leq j \leq 3\},$$

see for ex. [**13**]. According to our data, we have only f at each vertex, so f_x, f_y and f_{xy} have to be estimated. Here we use H. A k i m a [**3**] procedure for estimation of derivatives. It selects n_c nodes P_i, $i = 1, \cdots, n_c$, that are closest to the node P. For every combination (i, j) a vector product $V_{ij} = \vec{PP_i} \times \vec{PP_j}$ $i, j = 1, 2, \cdots, n_c$, where P, P_i, P_j are arranged to be counterclockwise, have to be evaluated, so that z component of V_{ij} is positive. Let $\vec{V} = (V_1, V_2, V_3) = \sum \vec{V_{ij}}$. The estimation of partial derivatives, are than given by

$$\frac{\partial f}{\partial x} = -\frac{V_1}{V_3}, \quad \frac{\partial f}{\partial y} = -\frac{V_2}{V_3}.$$

For the second order derivatives an analogue procedure is used. As $\partial^2 f/\partial x \partial y$ can be obtained either as $\frac{\partial}{\partial x}(\partial f/\partial y)$ or $\frac{\partial}{\partial y}(\partial f/\partial x)$ it is the best to take an arithmetic average of these values. The first value is obtained by using values for $\partial f/\partial x$ instead of f in interpolating nodes, while for the second value $\partial f/\partial y$ is considered as interpolating data.

On the basis of 16 data per each mesh rectangle, the linear system of the format 16×16 is formed which gives 16 coefficients of the interpolant patch $p(x, y)$. In our examples the Gauss algorithm with pivoting is used.

Of course, interpolation is performed on the unit square $\{(0,0),(0,1),(1,0),(1,1)\}$ which is transformed by an affine transform to the real mesh rectangle $R = \{(x_0, y_0),\ (x_0 + h_x, y_0),\ (x_0, y_0 + h_y),\ (x_0 + h_x, y_0 + h_y)\}$.

If the "square coordinates" are denoted by X_s and Y_s, then the transformations into "rectangle coordinates" (X, Y) are

$$X = h_x X_s + x_0, \quad Y = h_y Y_s + y_0.$$

Then, the partial derivatives, estimated on the square have to be transformed as well, by

$$\frac{\partial f}{\partial X_s} = h_x \frac{\partial f}{\partial X}, \quad \frac{\partial f}{\partial Y_s} = h_y \frac{\partial f}{\partial Y},$$

which is necessary for contouring. Now, all the input information are available and contouring is performed as it is described in Section 2.

4. Scattered data interpolation

Suppose that $\{M_i = (x_i, y_i)\}_{i=1}^m$ are scattered interpolating nodes in $D \subset \mathbb{R}^2$ and $\{f(M_i)\}_{i=1}^m$ are corresponding data set. Here the method of H. A k i m a [2], [4], [5], [6] for bivariate interpolation is used. This method is based on max-min angle triangulation of the points $\{M_i\}$ suggested by C. L a w s o n [14]. As it is shown by G. N i e l s o n [16], [17] max-min triangulation of Lawson, is equivalent to min-max criteria of F. L i t t l e and R. B a r n h i l l [7]. Characterization of these two triangulations is similar: each triangulation is associated with a vector having n_i entries representing either the largest or smallest angle of each triangle. These entries are ordered and then a lexicographic ordering of the vectors is used to impose an ordering on the set of all triangulations. In the case of min-max criteria, the smallest of these vectors based on their lexicographic ordering gives the optimal triangulation, while in the case of max-min criteria, the largest vector is associated with the optimal triangulation.

After the optimal triangulation is obtained, H. A k i m a [2] suggests a polynomial interpolant on each triangle T which is defined by the following facts:

The value of the function $f(x, y)$, $(x, y) \in T$ is interpolated by a bivariate fifth-degree polynomial

$$q(x, y) = \sum_{i=0}^{5} \sum_{j=0}^{5-i} a_{ij} x^i y^j,$$

which has 21 coefficients to be determined. The values of the function and its first-order and second-order partial derivatives, $f, f_x, f_y, f_{xx}, f_{xy}$ and f_{yy},

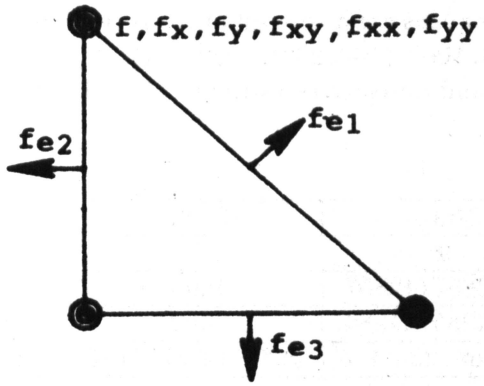

Figure 2: The Argyris triangle

are given at each vertex of T (this provides 18 data per triangle) and three directional derivatives normal to three sides of T. This triangle is known as the Argyris triangle [8], see Figure 2.

The linear system for the coefficients a_{ij} is solved by Gauss algorithm with pivoting, as in Section 3.

The interpolation is performed on the unit triangle $T = \{(0,0), (0,1), (1,0)\}$ and then it is transformed to an arbitrary one $\{(x_0, y_0), (x_1, y_1), (x_2, y_2)\}$, such that

$$\begin{aligned} X &= (x_2 - x_0)X_t + (x_1 - x_0)Y_t + x_0, \\ Y &= (y_2 - y_0)X_t + (y_1 - y_0)Y_t + y_0, \end{aligned}$$

where (X_t, Y_t) are coordinates in the unit triangle. The necessary transformations of partial derivatives reads

$$\begin{aligned} \frac{\partial f}{\partial X_t} &= (x_2 - x_0)\frac{\partial f}{\partial X} + (y_2 - y_0)\frac{\partial f}{\partial Y}, \\ \frac{\partial f}{\partial Y_t} &= (x_1 - x_0)\frac{\partial f}{\partial X} + (y_1 - y_0)\frac{\partial f}{\partial Y}. \end{aligned}$$

Now we can pass to the contouring procedure, defined by our algorithm.

5. Examples

E x a m p l e 1. Let the nodes are displaced at the rectangular mesh (0, 0),(0, 100), (0, 200), (100, 0), (100, 100), (100, 200), (200, 0),(200, 100), (200, 200). The values of the function f and derivatives estimated by Akima method, are given in the tables bellow:

nodes	(0,0)	(0,100)	(0,200)
f	0	0	0
f_x	0.042857142857	0.06	0.042857142857
f_y	0.014285714286	0.02	-0.014285714286
f_{xy}	2.6530612245E-5	-8.8571428571E-5	-0.00026933877551

nodes	(100,0)	(100,100)	(100,200)
f	0	10	0
f_x	0.02	0.05	0.02
f_y	0.06	-0.05	-0.06
f_{xy}	8.8571428572E-5	-0.00025	-0.00025142857143

nodes	(200,0)	(200,100)	(200,200)
f	0	0	0
f_x	-0.014285714286	-0.06	-0.014285714286
f_y	0.042857142857	0.02	-0.042857142857
f_{xy}	1.1102230246E-16	0.00028571428572	8.0612244898E-5

After applying our algorithm, the level-lines map of the polynomial interpolant is obtained and it is shown in Figure 3 (left).

E x a m p l e 2. For the data from Example 1, the method of triangulation is performed. Then, the method for scattered data is applied to these regular data. The derivatives f_x, f_y and f_{xy} are estimated to be the same as in Example 1. Due to the requirement of additional information for Argyris triangle, two extra derivatives are necessary: f_{xx} and f_{yy}. They are estimated as

nodes	(0,0)	(0,100)	(0,200)
f_{xx}	-0.00017346938776	-0.00015428571429	-7.7551020408E-5
f_{yy}	-0.00022244897959	-0.00033428571429	-0.00019387755102

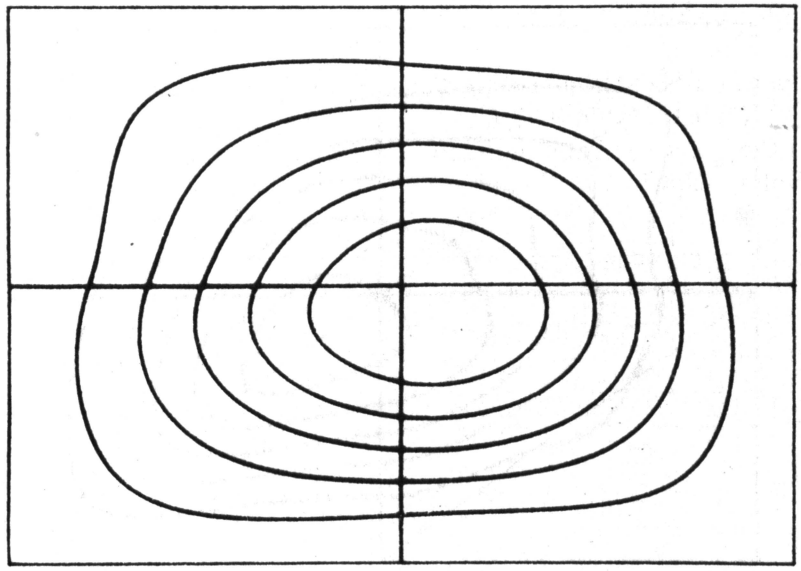

Figure 3: Levels lines for Examples 1 and 2

nodes	(100,0)	(100,100)	(100,200)
f_{xx}	-0.00024857142857	-0.00045	-0.00024857142857E-5
f_{yy}	-0.00088857142857	-0.00025	-0.00019428571429

nodes	(200,0)	(200,100)	(200,200)
f_{xx}	-0.00041836734694	-0.00011628571429	-0.00041836734694
f_{yy}	-0.00060204081633	-0.00056285714286	-0.00022244897959

The surfaces is contoured by our algorithm and the result is shown in Figure 3 (right).

Two surfaces differs slightly. This non-coincidence of two interpolants are caused by different polynomials used. But, it is worth to mention that contouring of these interpolants is a good way to study theirs difference. Namely, from the rendered sophisticated computer-graphic $3D$ picture of two surfaces, this difference will be harder to detect.

Also, it is easy to notice, from the level-lines shown in Figure 3, that both interpolant surfaces are not symmetric in spite of symmetry of data. This effect is caused by the nonsymmetrical values of the derivatives being estimated. Namely, the method of estimation, used by Akima is not affinely invariant, and the transformation spoils the symmetry.

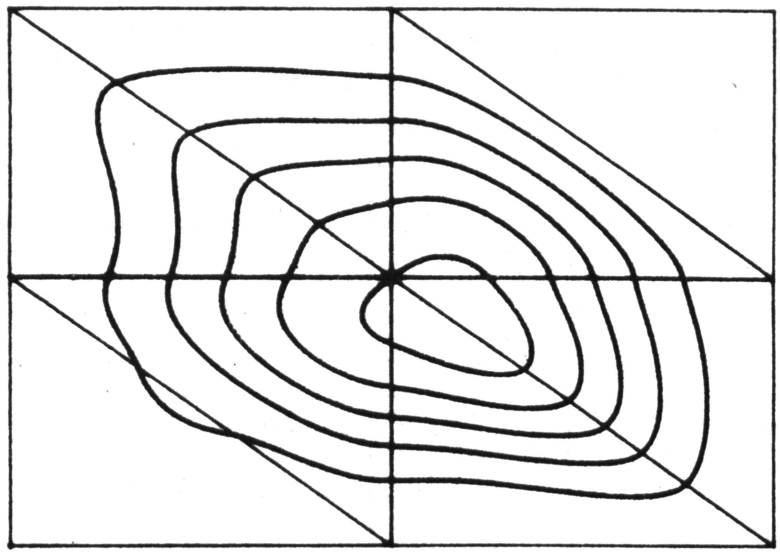

Figure 4: Triangulation and corresponding level-lines (Example 3)

E x a m p l e 3. Here, the scattered data from [16], [17] are used. Triangulation is done by the max-min algorithm and the same result as in [16], [17] obtained. This optimal triangulation is shown in Figure 4 (left). Partial derivatives are estimated as it is shown in the tables. Finally, the level-lines, produced by our algorithm are shown in Figure 4 (right).

nodes	(0.50,0.90)	(0.50,0.80)	(0.20,0.50)
f	0	0	0
f_x	-0.24096385542	-0.28571428571	0.35353535353
f_y	-0.78313253012	-0.92857142857	-1.1111111111
f_{xy}	1.1875156743	1.267786427	1.2423005307
f_{xx}	-2.0040908036	-1.9501921805	-3.2872767352
f_{yy}	1.082564792	1.0135117277	1.1893804166

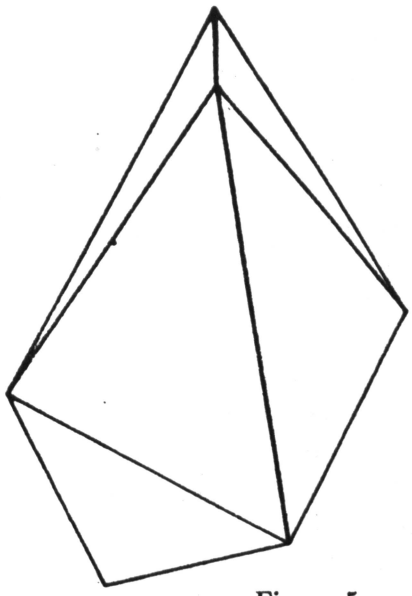

Figure 5

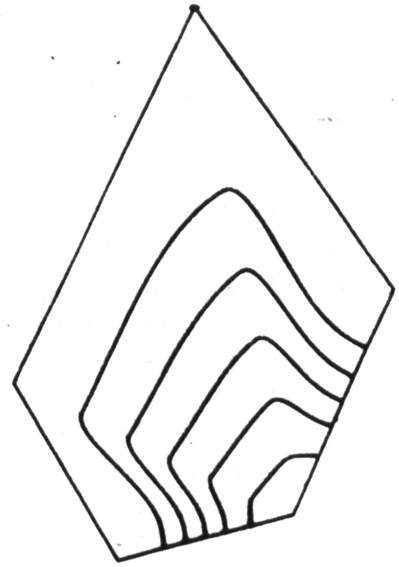

Figure 6

nodes	(0.85,0.40)	(0.40,0.20)	(0.70,0.15)
f	0	1	0
f_x	-1.1825192802	-1.5718157182	-1.3477088949
f_y	-0.56555269923	-2.0054200542	-0.59299191375
f_{xy}	1.8270240065	2.8986268223	2.1062345281
f_{xx}	-0.34521221808	-0.68807476123	-0.89830274422
f_{yy}	0.54090140899	2.0214264377	0.38376749686

References

1. H. A k i m a. *A.* Method of Bivariate Interpolation and Smooth Surface Fitting Based on Local Procedures, *Comm. ACM*, **17**, 1974, no. 1, 18–20.
2. H. A k i m a. *A.* Method of Bivariate Interpolation and Smooth Surface Fitting for Irregularly Distributed Data Point, *ACM Trans. on Math. Software*, **4**, 1978, no. 2, 148–159.
3. H. A k i m a. *O.*n Estimating Partial Derivatives for Bivariate Interpolation of Scattered Data, *Rocky Mountain Journal of Mathematics*, **14**, 1984, no. 1, 41–52.

4. H. A k i m a. *A*.lgorithm 526, Bivariate Interpolation and Smooth Surface Fitting for Irregularly Distributed Data Points [E1], *ACM Trans*, Math. Software **4**, 1978, 160–164.

5. H. A k i m a. *R*.emark on Algorithm 526, *ACM Trans. on Math. Software*, **5**, 1979, no. 2, 242–243.

6. H. A k i m a. *R*.emark on Algorithm 526, *ACM Trans. on Math. Software*, **11**, 1985, no. 2, 186–187.

7. R. B a r n h i l l, F. L i t t l e. *T*.hree- and four-dimensional surfaces, *Rocky Mt. J. Math.*, **14**, 1984, 77–102.

8. J. C a r n i c e r, M. G a s c a. *O*.n Finite Element Interpolation Problems Mathematical Methods in CAGD, (T. Lyche and L. L. Schumaker eds.), *Academic Press*, Boston 1989, 105–113.

9. G. F a r i n. *T*.riangular Bernstein-Bézier patches, *Computer Aided Geometric Design*, **3**, 1986, 83–127.

10. Lj. M. K o c i ć. *A*. graphical method for separating extrema of implicit function, *Wiss. Z. TH Ilmenau*, **35**, 1989, no 6, 157–160.

11. Lj. M. K o c i ć, D. M. M i l o š e v i ć. *O*.n level sets of Bernstein-Bézier operators, *Zbornik radova Filozofskog fakulteta u Nišu, Serija Matematika* **6**, 1992, 19–25.

12. Lj. K o c i ć, D. M i l o š e v i ć. *N*.umerical Characteristics of Algorithm for Implicit Curve Tracing, *Facta Universitatis (Niš), Ser. Math. Inform.* **8**, 1993, 97–107.

13. P. L a n c a s t e r, K. Š a l k a u s k a s. *C*.urve and Surface Fitting: An Introduction, *Acad. Press*, 1986.

14. C. L a w s o n. *S*.oftware for C^1 surface interpolation in Mathematical Software III, J. R. Rice (ed.), *Academic Press*, New York, 1977, 161–194.

15. D. M. M i l o š e v i ć. *I*.mplicit function graphs and application in geometric modeling, *MS degree work*, University of Niš, Faculty of Electronic Engrg, Niš 1992.

16. G. N i e l s o n. *A*.n Example with a Local Minimum for the MinMax Ordering of Triangulations, *Technical Report TR-87-014*, Arizona State University, 1987.

17. G. N i e l s o n. *A*. Characterization of an Affine Invariant Triangulation, *Technical Report TR-88-023*, Arizona State University, 1988.

18. C. P e t e r s e n. *A*.daptive contouring of three dimensional surfaces, *Computer Aided Geometric Design*, 1, 1984, 61–74.
19. C. P e t e r s e n, B. P i p e r, A. W o r s e y. *A*.daptive contouring of a trivariate interpolant, G. Farin. ed., Geometric Modeling, SIAM, Philadelphia, PA , 1986.
20. D. T o š i ć, D. T o š i ć. *T*.wo methods for the curve drawing in the plane, *Numerical Methods and Approximation Theory*, Niš, 1984, 61–65.
21. A. W o r s e y, G. F a r i n. *C*.ontouring a bivariate quadratic polynomial over triangle, *CAGD*, 7, 1990, no. 1-4, 337–352.

Faculty of Electronic Engineering
Department of Mathematics, P. O. Box 73
University of Niš, 18000 Niš
YUGOSLAVIA

Received 21.10.93