

Accurate visualization of graphs of functions of two real variables

Zeitoun D.G.* and Thierry Dana-Picard**

Abstract—The study of a real function of two real variables can be supported by visualization using a Computer Algebra System (CAS). One type of constraints of the system is due to the algorithms implemented, yielding continuous approximations of the given function by interpolation. This often masks discontinuities of the function and can provide strange plots, not compatible with the mathematics. In recent years, point based geometry has gained increasing attention as an alternative surface representation, both for efficient rendering and for flexible geometry processing of complex surfaces. In this paper we present different artifacts created by mesh surfaces near discontinuities and propose a point based method that controls and reduces these artifacts. A least squares penalty method for an automatic generation of the mesh that controls the behavior of the chosen function is presented. The special feature of this method is the ability to improve the accuracy of the surface visualization near a set of interior points where the function may be discontinuous. The present method is formulated as a minimax problem and the non uniform mesh is generated using an iterative algorithm. Results show that for large poorly conditioned matrices, the new algorithm gives more accurate results than the classical preconditioned conjugate algorithm.

Keywords—Function singularities, mesh generation, point allocation, visualization, collocation least squares method, Augmented Lagrangian method, Uzawa's Algorithm, Preconditioned Conjugate Gradient

I. GENERAL FRAME OF STUDY

With the introduction of the computer into the learning environment the way Mathematics is conveyed has changed. The traditional sequence Definition- Theorem-Proof has received a complement with examples where visualization plays an important role. This is true in numerous mathematical domains, such as Geometry, using the so-called Dynamical Geometry Packages, and also Analysis and Algebra with Computer Algebra Systems (CAS). Questions about the study of functions and curve discussion have been studied by [20], [11], [12] and others. In particular, various limitations of the usage of the computer have been discussed. Many educators have replaced the traditional sequence mentioned above by another one, beginning with computer assisted experimentation. A well known case of CAS experimentation prior to theoretical study is curve discussion. It happens that a discontinuity of the function or the non-existence of a limit at some point are not shown by the display, despite the fact that a proof is easy to write. Such a cognitive conflict can appear, in a stronger form,

when studying functions of two real variables. In order for the eye to catch the situation, most Computer Algebra Systems enable a dynamical point of view, using the mouse to turn the surface around. This can help to discover discontinuities or points where partial derivatives cannot exist, but how to be confident of the exactness of the visual impression? As we will see, discontinuities can be hidden. Functions of two real variables are generally introduced in an Advanced Calculus course, where the students discover generalizations of notions learnt in their first Calculus course. The respective roles of the first and second derivatives are extended in the new frame to discover extrema, saddle points and points of inflection. When arriving at the visualization stage, drawings are harder to obtain by hand-work and computerized help is welcome.

Plots are obtained via commands similar to the 2D-commands and the student is confident that what is seen is what has to be seen. We can illustrate the first problem encountered with the example of a paraboloid: when drawn on the black/white board it looks as in Figure 1(a), but generally a CAS plots something like (b) or (c), because of a bounding box. Even when the bounding box is no displayed, it cuts the paraboloid in a visible way.

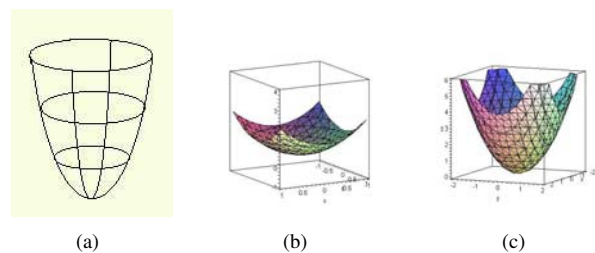


Fig. 1. Three views of the same paraboloid

In Figure 1(a), the general shape of the surface can be imagined, but this is doubtful for Figure 1(b). The different choices of the ranges for x and y are responsible for this disparity in the display.

We illustrate now a problem of another nature. Consider the function defined by

$$\begin{cases} f(x, y) = \frac{xy}{x^2 + y^2} & \text{if } (x, y) \neq (0, 0) \\ f(0, 0) = 0 \end{cases}$$

and whose graph is displayed in Figure 2 for $-1 \leq x \leq 1$, $-1 \leq y \leq 1$ (we used the program called DPGraph¹). It

* E.S.G. Ecole Supérieure de Gestion
11 Rue Amboise Paré; 75015 Paris; France

** Department of Mathematics -Jerusalem College of Technology
Havaad Haleumi Street, 21 -Jerusalem 91160 Israel
e-mail: ed.technologie@gmail.com - dana@jct.ac.il

¹<http://www.dpgraph.com>

”seems” that something special happens in a neighborhood of $(0, 0)$, leading the student to check whether the function has a finite limit at $(0, 0)$ or not. The grid itself supports the intuition that when approaching the origin on different straight lines, something changes, whence the non-existence of a limit at $(0, 0)$.

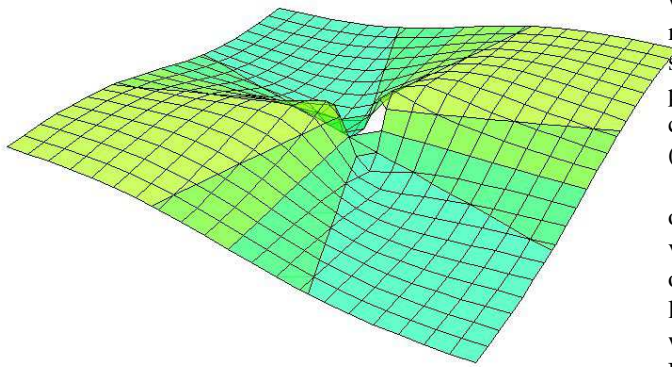


Fig. 2. The function is not continuous at the origin.

A. Different plots with different computer packages

Consider the function of two real variables given by $f(x, y) = \frac{1}{1-(x^2+y^2)}$. It is a rational function, continuous all over the plane, excepted at the points of the unit circle. When approaching these points from inside the unit circle, the function has a positive infinite limit and a negative positive infinite limit when (x, y) approaches a point of the unit circle from outside. We should mention that a precise proof of this fact is rarely written in a standard course in Multivariable Calculus; sometimes the situation is studied in more details using polar coordinates. The cylinder given in the 3-space by the equation $x^2 + y^2 = 1$ plays the role of an asymptote to the graph of the function, the graph approaching the cylinder in two different ways, according to which part is considered, either inside or outside. Three different plots, obtained with DPGraph are displayed in Figure 3, according to the data: (a) $-2 \leq x, y \leq 2, -3 \leq z \leq 3$; (b) $-3 \leq x, y, z \leq 3$; (c) $-2 \leq x, y \leq 2, -5 \leq z \leq 5$.

The three plots are similar but show differences: the cylinder is always ”nice”, but the graph of the function presents various indentations²:

- the inner-upper sheet is indented in (a), not in (b) and (c), but the z-values seem to become too large for them to be fully plotted in (c).
- the lower sheet has strong indentations in all cases, instead of being ”full”, as could have been expected.

The impossibility for a CAS to plot a graph close to a singularity has been already studied for plane curves, and various behaviors described by [20], [11] and others. Here

²The color plots can be viewed at the following URL: <http://ndp.jct.ac.il/> ...

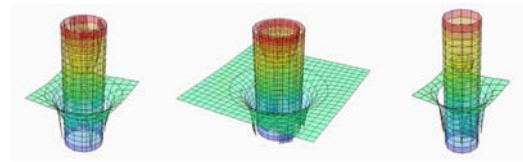


Fig. 3. The graph and its asymptotic cylinder, using DPGraph.

we have a similar situation for surfaces in 3-space, even more complicated as the singularities are not isolated. Let us see what happens with another computer package. Figure 4 presents plots of the same function obtained with Maple’s commands **plot3d** ((a) and (b)) and **implicitplot3d** ((c) and (d)), over different domains containing the unit circle.

The various plots, obtained either with different commands over the same domain, or with the same command over various domains, look very different. Figure 4(a) show needles oriented either upwards or downwards on both sides of what looks like a plane. Figure 4(b) shows such needles but together with some kind of a solid, looking like the solid appearing in Figure 4(c), and Figure 4(d) shows isolated needles, in greater number than Figure 4(a). Figure 4(b) and 4(c) seem to fit the intuition, but with irregular shape. Moreover, where intuition would like to see a rupture due to the discontinuity, there are at some place holes, and at other places bridges. What happened? These plots show too many differences to be correct, and they do not allow an accurate graphical study of the given two-variable function.

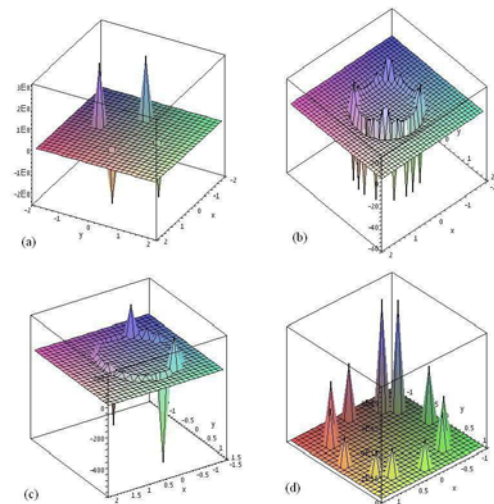


Fig. 4. Four strange plots for the same function, all false

In this paper, we present new algorithms in order to overcome the above problems.

II. TECHNIQUES OF VISUALIZATION OF PLOTS OF SURFACES USING MESH GENERATORS

Triangle meshes are the most common surface representation in computer graphical applications. Because of their

simplicity and flexibility, they replace traditional CAD surface representations, like NURBS surfaces (see [21]), in many areas where processing performance is an important issue. The reason for this is that triangle meshes are significantly more flexible, since surfaces of any shape and topology can be represented by a single mesh without the need to satisfy complicated interpatch smoothness conditions. The simplicity of the triangle primitive allows for easier and more efficient geometry generation and geometry processing algorithms.

Obviously, since the triangle primitive is mathematically much simpler compared to a NURBS patch, more of them have to be used to obtain the same approximation quality. However, if a smooth surface has to be represented by a triangle mesh (a piecewise linear surface), the approximation order is quadratic, i.e., halving the edge lengths reduces the error by a factor of 4 which means the number of triangles is inversely proportional to the approximation error. Hence, even with the weaker asymptotic behavior, a good approximation (for the typical precision requirements in graphics applications) can be achieved with a moderately fine mesh whose vertex density and distribution is adapted to the surface curvature, i.e., to the shape complexity.

Despite their being more flexible than NURBS, triangle meshes can also have restrictions and disadvantages in some special applications. Most algorithms working on triangle meshes require topologically consistent surfaces. As a consequence, manifold extraction or topology cleanup steps are necessary for mesh generation methods (see [2] and [3]).

Representing sharp features, like edges or corners in technical data sets, is a well studied problem for triangle meshes. Because the surface is no longer differentiable, the approximation power breaks down to linear order. Additionally, alias artifacts are introduced by insufficient sampling, which cannot be removed by increasing the sampling density [18]. In order to remove these artifacts and reduce normal noise, the sampling has to be aligned to the principal curvature directions [19]. If surface splats are to represent sharp features, all splats that sample these features have to be clipped against one or two clipping lines (for edges and corners respectively) that are specified in their local tangent frames. Therefore, for 2D plots, an accurate representation may be achieved when building the mesh on the isoclines of the function.

III. GRAPHIC REPRESENTATION OF A REAL FUNCTION OF TWO REAL VARIABLES

When dealing with functions of one or two real variables, the accuracy of the Cartesian grid used for the interpolation influences the graph quality, especially near points of discontinuity. In Computer Algebra Systems there are two types of algorithms for viewing plotting functions.

A. Visualization based on a Cartesian mesh

Three-dimensional plot commands represent a real function of two real variables in a three dimensional view by approximating the function on a Cartesian grid. The two dimensional

domain is discretized in a series of rectangular lagrangian elements, and on each element, the approximation is used. As an example, the two-dimensional Lagrangian interpolation arising out of linear bases defined on a rectangular element of size $h_x \times h_y$ are built as follows. Denote $\xi = \frac{x}{h_x}; \nu = \frac{y}{h_y}$. Then

$$\begin{aligned}\phi_1(\xi, \nu) &= \left[\frac{1}{2}(1 - \xi)\right] \left[\frac{1}{2}(1 - \nu)\right], \\ \phi_2(\xi, \nu) &= \left[\frac{1}{2}(1 + \xi)\right] \left[\frac{1}{2}(1 - \nu)\right], \\ \phi_3(\xi, \nu) &= \left[\frac{1}{2}(1 + \xi)\right] \left[\frac{1}{2}(1 + \nu)\right], \\ \phi_4(\xi, \nu) &= \left[\frac{1}{2}(1 - \xi)\right] \left[\frac{1}{2}(1 + \nu)\right].\end{aligned}$$

More sophisticated and accurate interpolations such as quadric, cubic, and Hermitian cubic 2-dimensional interpolation functions may be used (see [21], Section 2). These interpolations are often used in computational software. In order to build a graph of a function, the domain is decomposed into a finite number of elements based on a collection of $n + 1$ points $P_0(x_0, y_0), P_1(x_1, y_1), \dots, P_n(x_n, y_n)$ for each element $[x_k, x_{k+1}] \times [y_j, y_{j+1}]$. On this element the function is approximated by a function built in a way similar to what has been described in the previous subsection. For a given function f of the two real variables x and y , the interpolation used may be written as a double sum: a first summation over all the elements of the discretization of the domain, and a second one for the interpolation over the given element:

$$F_{app}(x, y) = \sum_{Cells} \sum_{i=1}^{i=4} f(x_i, y_i) \phi\left(\frac{x}{h_{ix}}, \frac{y}{h_{iy}}\right). \quad (1)$$

B. Visualization based on a parametric representation of the curve

In this type of plotting the Cartesian coordinates are functions of two real Parameters $x = x(s, t); y = y(s, t)$ ³. Then the discretization of the domain is performed in the (s, t) -plane. The domain in this plane is separated into a finite number of elements based on a collection of $n + 1$ points

$$P_0(x(s_0, t_0), y(s_0, t_0)), P_1(x(s_1, t_1), y(s_1, t_1)), \dots, P_n(x(s_n, t_n), y(s_n, t_n)) \quad (2)$$

for each cell $[s_k, s_{k+1}] \times [t_j, t_{j+1}]$. On this cell the function is approximated by a function built as above. For a given function f , the interpolation is given as follows:

$$f_{app}(x, y) = \sum_{Cells} \sum_{i=1}^{i=4} f(x(s_i, t_i), y(s_i, t_i)) \phi\left(\frac{s}{h_{ix}}, \frac{t}{h_{iy}}\right). \quad (3)$$

³Polar coordinates $x = r \cos t; y = r \sin t$, where $s \geq 0; t \in \mathbb{R}$, are often used but tens of other kinds of coordinates are generally available in standard CAS. We illustrate the case of polar coordinates in subsection III-E.

C. Discontinuities and critical points of a function of two variables on a Cartesian mesh

The plot command yields visualization as a surface plotting. The main difference between the two situations described above is in the mathematical treatment of the discontinuity and of the critical points of the function. For a function f defined on a domain D in the (x, y) -plane, local maxima, local minima or saddle points can occur either at boundary points of R , or at interior points (x_0, y_0) of D where the first partial derivatives vanish, i.e. $f_x(x_0, y_0) = f_y(x_0, y_0) = 0$, or at points where f_x or f_y fail to exist.

Here is the core of the strange apparitions in Figure 4 (the needles): depending on the type of interpolation, the condition

$$f_x(x_0, y_0) = f_y(x_0, y_0) = 0 \tag{4}$$

is different from the condition

$$f_{approx}(x_1, y_1) = f_{approx}(x_1, y_1) = 0. \tag{5}$$

Moreover, these extrema conditions may appear at different points $(x_0, y_0) \neq (x_1, y_1)$.

For polynomial quadratic interpolation functions (Hermite and cubic basis functions), these last conditions request the solution of a system of linear equations in order to determine the (possible) local extrema. Such critical points may exist on each discretization cell of the domain. This simple analysis permits to understand why we are viewing plots with local extrema without any connection with the known mathematical behavior of the function: the local extrema plotted by the software correspond to extrema of the approximation function and not of the actually given function.

For linear interpolation functions of cubic type (most frequently used in mathematical software), it could be shown that the condition given by Equation (5) is not dependent on the discretization steps h_{ix} and h_{iy} for a command based on a Cartesian grid (**plot3d**). In this case, the local extremum of the discretization is obtained on the corner of the elements.

To illustrate (and understand) the creation of needles in the plot of a function of two variables, consider the rational function defined by

$$f(x, y) = \frac{1}{x + y - 1}.$$

A MatLab 7 plot of this function for $-4 \leq x \leq 4; -2 \leq y \leq 2$ is displayed in Figure 5.

The needles appear near the discontinuity line whose equation is $x + y = 1$. In order to understand this effect consider a Cartesian discretization for the square given by $0 \leq x \leq 2; 0 \leq y \leq 2$, using n^2 points. We have $h = h_x = h_y = \frac{2}{n}$. The approximated function for plotting is determined by the values of the points $x = ih; y = jh$ and an interpolation function

$$f_{app}(i, j, h) = \frac{1}{h(i + j) - 1}.$$

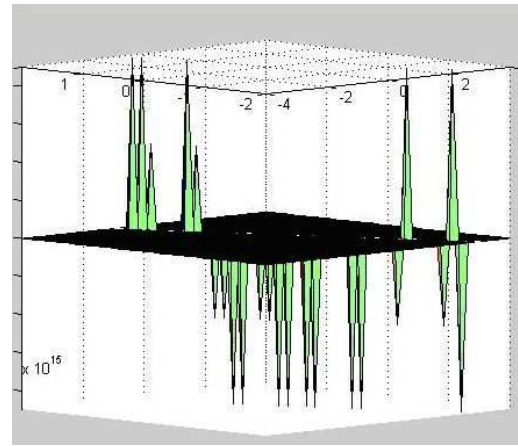


Fig. 5. MatLab plot of $f(x, y) = 1/(x + y - 1)$ with mesh discretization

Using a linear interpolation, one may estimate the plotting value along a line close to the discontinuity line $x + y = 1$, let's say the line whose equation is $x + y = 1 + \epsilon$, where ϵ is an arbitrary small real number. Along the line whose equation is $x + y = 1 + \epsilon$, the function should be constant and equal to $f_{x+y=1+\epsilon} = \frac{1}{\epsilon}$; $\epsilon > 0$. However, because of the Cartesian grid discretization, the plotting software uses approximate values of the function at the relevant corner of the grid. Figure 6 shows plots of the approximated function along a line close to the discontinuity line for constant h and different values of ϵ .

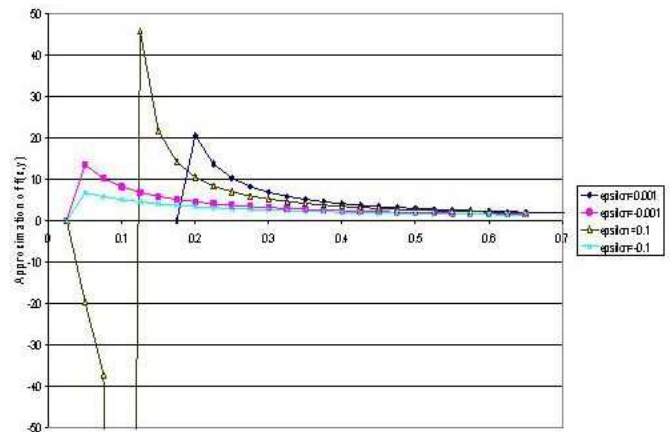


Fig. 6. Numerical Solution of $f(x, y) = 1/(x + y - 1)$ along the line whose equation is $x + y - 1 = \epsilon$

D. Discontinuities of a function of two variables using a parametric representation of the surface.

The condition given by Equation (5) for polar coordinates leads to a solution dependent on the grid discretization steps. Then, the viewing of local extrema on the graph based on Cartesian grid cannot be repaired by playing with the size of

the mesh but may be repaired by playing with the discretization size and the domain range for plots based on parametric representations (e.g. Maple's command **parametricplot3d**).

E. Transforming the question from Cartesian coordinates into polar coordinates

An illustration of the problem may be seen for the different plots obtained for the given function. Figure 7 shows a partial plot of the graph of the function defined in Section 2, constructed using polar coordinates. In Figure 4, no discontinuity appears near the unit circle, but the plot in polar coordinates (Figure 7) shows clearly the discontinuity of the function and the asymptotic behavior.

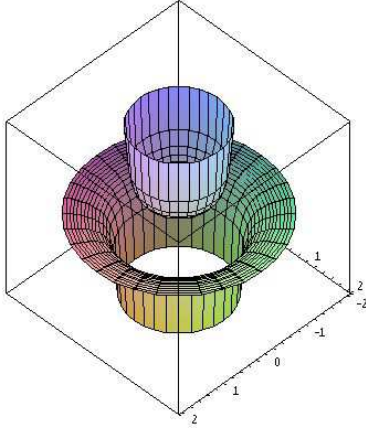


Fig. 7. A plot in polar coordinates.

The use of isoclines of the function for the definition of the parametrization of the function is a very efficient solution for the reduction of artifact. Unfortunately, the determination of isoclines for any function may be very difficult and require special algorithms for automatic meshing.

IV. OPTIMAL POINT ALLOCATION FOR ACCURATE MESH GENERATION

The other strategy for an optimal allocation of mesh points is to control the error on the mesh by using a point based technique. In this section, we present an algorithm for automatic meshing where the error of the function is controlled on a zone of discontinuity.

A. Algorithm for Mesh Generation

A pointbased geometry representation can be considered as a sampling of a continuous surface, resulting in 3D positions p_i , optionally with associated normal vectors n_i or auxiliary surface properties like, e.g., colors or other material properties. If normal vectors are not given, they can be estimated by

analyzing the local neighborhood of each sample point. Because there is no connectivity information available, these local neighborhoods are usually constructed using either Euclidean neighborhoods or k nearest neighbors (see [18]).

Consider the general problem of finding the optimal mesh points p_i for the representation of the function $z = f(x, y)$ defined on the whole two dimensional domain of the visualization is called Ω . The function $f(x, y)$ has a discontinuity along the line $g(x, y) = 0$. Suppose that we define an initial Cartesian mesh defined by a set of points $q_j = (x_j, y_j)$; $j = 1, \dots, N$. We may define a discontinuity boundary B as the set of points of the mesh located at a neighborhood of the discontinuity: $B = \{q_l \mid l = 1, \dots, m, g(q_l) = \varepsilon\}$, where ε is a small number. For mesh points belonging to B , a continuous surface builded by a mesh generator will create discontinuities in the graphic representation. As we discuss in the previous section, a way to avoid this is to control the gradient of the functions $f(x, y)$ near the surface $g(x, y) = 0$.

Consider now the displacement vector $u = (u_i, v_i)$ of each mesh point q_i , the condition to impose should be $grad(g)|_i \cdot u = \varepsilon_i$. The optimal displacement of each point of the mesh may be formulated as a least squares problem. This L^2 least squares may be described in terms of the following Formulation. It consists of the minimization of a functional I_d with respect to the unknown displacement vector \mathbf{a} ; where

$$I_d = \sum_{i=1}^k [R_L(\mathbf{a}, \mathbf{x}_i)]^2 + c \sum_{i=k+1}^m [R_B(\mathbf{a}, \mathbf{x}_i)]^2 \quad (6)$$

in which $R_L(\mathbf{a}, \mathbf{x}) = Lv_h - f$ \mathbf{x} in Ω
 $R_B(\mathbf{a}, \mathbf{x}) = Bv_h - g$ \mathbf{x} on Γ The operators L and B , and the vectors f and g are defined as follows:

$$L = grad(f)_{q_i}, B = grad(g)_{q_i}, f = 0, g = -g(q_i); \quad (6)$$

A corresponding discrete formulation of the least squares method, also called *collocation least squares*, consists of selecting a series of collocation points inside the domain and on the boundary, and minimizing the following functional :

For $i = 1, \dots, k$, \mathbf{x}_i corresponds to the interior collocation points and for $i = k + 1, \dots, m$ to the boundary collocation points (see [14]).

Important boundary conditions can be emphasized with large weights c , forcing the boundary residuals to be small. However, the addition of the integral of the boundary residual may cause the integral of the interior residuals for a given solution vector $\bar{\mathbf{a}}$ to remain large. This scaling problem may defeat the computational objective of satisfying the conservation laws. This is particularly true when the weight c is very large. In this case, the condition number of the least squares matrix becomes very large (Sermer and Mathon in [23]).

The purpose of the new method is to control the spatial distribution of the error in the classical collocation least squares method. The improvement of the solution is generally needed at specific points of the domain such as area of large gradients, edge of rectangular domain, etc... The classical way to improve the solution is to refine the mesh in order to reduce

the numerical error. The present paper present an alternative method based on constraint optimization. The discretization is fixed and we introduce a local error parameter ε . The improvement of the error is performed by constraining the least squares method. The goal of the new method is to control the spatial distribution of the error more than to reduce drastically. It is important to realize that the improvement of the error was obtained without changing the size of the cell of the collocation points but by reformulating the least squares method.

B. General Algorithm for Mesh Generation

The different steps of the mesh generation algorithm may be summarized as follows:

- Define the initial cartesian mesh M_0 .
- Define a level ε .
- Compute the discontinuity Zone B .
- Formulate Problem (P_h).
- Solve Problem (P_h) and determine the optimal displacements u_i .
- Generate the point cloud $(\{q_i, f(q_i)\}; (q_i + u_i, f(q_i + u_i)))$.
- Use the marching cube method for the generation of the mesh from the point cloud.

In the following section, we will present an algorithm of solution for Problem (P_h).

C. Least squares formulation

In this section, an alternative formulation of the approximate least squares problem, called **Problem** P_h , is first presented. Then we present a penalty method similar to the augmented Lagrangian method for the solution of the minimization problem P_h . The special feature of this method is the possibility of working with a given value of c and improving the conditioning of the least squares matrix by introducing an iterative algorithm. Then error estimates of the approximate solution are given. Finally, a comparison of the present method with the classic least squares method is developed.

At a point \mathbf{x} of any subregion $\Omega' \subseteq \bar{\Omega}$, the point error may be characterized by the function $\varepsilon(\mathbf{x})$ defined as follows:

$$\varepsilon(\mathbf{x}) = T(u_h(\mathbf{x}) - u(\mathbf{x})) \quad (7)$$

where the differential operator T is defined by :

$$T = \begin{cases} L & \text{if } \mathbf{x} \in \Omega \\ B & \text{if } \mathbf{x} \in \Gamma. \end{cases} \quad (8)$$

Let the subregion Ω' be a set of points where a high degree of accuracy of the approximated solution is required. For example, in the collocation least squares method, Ω' may be a selected set of interior collocation points, or it may be defined as a set of boundary points.

When the operator T is linear with respect to the solution, the pointwise operator equation (Equations (7) and (8)) at any point $\mathbf{x} \in \Omega'$ may be written as a function of $\varepsilon(\mathbf{x})$ as follows:

$$Tu_h(\mathbf{x}) - b = \varepsilon(\mathbf{x}) \quad (9)$$

where the function b is defined by:

$$b = \begin{cases} f & \text{if } \mathbf{x} \in \Omega \\ g & \text{if } \mathbf{x} \in \Gamma. \end{cases} \quad (10)$$

Note that when Equation (9) is satisfied, the approximate solution $u_h(\mathbf{x})$ satisfies exactly the partial differential equation (Equations (7) and (8)) at a point \mathbf{x} of Ω' .

In the proposed least squares formulation, the interior residual and the boundary residual are set equal to the function $\pm\alpha\varepsilon(\mathbf{x})$ at each point of Ω' . The term $\alpha\varepsilon(\mathbf{x})$ represents the range of variation allowed at each point inside Ω' and α is a positive scaling factor. The condition of zero residual is replaced by the alternative condition

$$\text{Condition A: } (Tv_h - b - \varepsilon(\mathbf{x}))^2 = \alpha^2\varepsilon(\mathbf{x})^2, \quad (11)$$

at each point of Ω' .

The proposed least squares approach can be formulated via the minimization of the integral of the interior residuals that still satisfy condition A at each point of Ω' .

For a given function $\varepsilon(\mathbf{x})$, the formulation of this minimization problem can be written as follows:

$$\begin{aligned} & \min_{\mathbf{a}} \int_{\Omega} [R_L(\mathbf{a}, \mathbf{x})]^2 d\Omega \\ & \text{Such that } [T(\lfloor \Phi(\mathbf{x}) \rfloor)]\mathbf{a} - b - \varepsilon(\mathbf{x})^2 = \alpha^2\varepsilon(\mathbf{x})^2 \mathbf{x} \in \Omega' \\ & \text{Problem } (L_h) \end{aligned}$$

When the collocation least squares method is used, the function $\varepsilon(\mathbf{x})$ is a vector of small numbers of dimension equal to the number of collocation points.

Note that for $\varepsilon(\mathbf{x}) = 0$ and $\Omega' = \Gamma$, Problem L_h reduces to the approximate least squares formulation (**Problem** P_h), i.e.

$$\begin{aligned} & \min_{\mathbf{a}} \int_{\Omega} [R_L(\mathbf{a}, \mathbf{x})]^2 d\Omega \\ & \text{Such that } B(\lfloor \Phi(\mathbf{x}) \rfloor)\mathbf{a} - g = R_B(\mathbf{a}, \mathbf{x}) = 0 \mathbf{x} \text{ on } \Gamma \\ & \text{Problem } (P_h) \end{aligned}$$

Problem (P_h) is the special case where the constraints associated with the minimization are only defined on the boundary Γ . In the following, we present a penalty method similar to the augmented Lagrangian method for the solution of the minimization problem L_h .

V. ALGORITHMS OF RESOLUTION

The algorithm proposed for the determination of the saddle points of the mini-max problem (L_h) is a version of the Uzawa's algorithm for quadratic programming as presented by different authors ([24], [15], [5]). The proposed algorithm may be described as follows:

$$\lambda^0(\mathbf{x}) \text{ specified arbitrarily} \quad (12)$$

With $\lambda^n(\mathbf{x})$ known, calculate \mathbf{a}^n , then $\lambda^{n+1}(\mathbf{x})$ (13)

$$(A_1 + 2cA_2) \mathbf{a}^n + 2 \int_{\Omega'} T(\Phi(\mathbf{x})) \lambda^n(\mathbf{x}) d\mu = F \quad (14)$$

$$\lambda^{n+1}(\mathbf{x}) = \lambda^n(\mathbf{x}) + \rho_n (c(T([\Phi(\mathbf{x}]]) \mathbf{a}^n - b) + (15)$$

$$(1 - \alpha^2) \lambda^n(\mathbf{x})) \quad (16)$$

The parameter ρ_n is an acceleration parameter defined in Uzawa algorithm and depending on the iteration n .

A. Collocation Least Squares Method

The methodology presented above may be applied to the determination of the optimal error associated with collocation points in the sense defined above. Consider now the minimum acceptable error necessary to respect the global mass balance, at each collocation point. This problem may be formulated as follows:

$$\min \int_{\Omega} [R_L(\mathbf{a}, \mathbf{x})]^2 dx$$

Such that $R_B(\mathbf{a}, \mathbf{x}_j) = (1 \pm \alpha) \varepsilon(\mathbf{x}_j) \quad j \in J$

Such that $R_L(\mathbf{a}, \mathbf{x}_i) = (1 \pm \alpha) \xi(\mathbf{x}_i) \quad i \in I$

Problem (Pc)

where $\varepsilon(\mathbf{x}_j)$ and $\xi(\mathbf{x}_i)$ represent the point error at a boundary collocation point (set J) and an interior collocation point (set I), respectively.

Using the preceding approach for each collocation point, the first order condition for the minimum corresponding to problem (Pc) may be written as:

$$(A_1 + A_2) \bar{\mathbf{a}} + 2 \sum_{j \in J} \lambda(\mathbf{x}_j) B(\Phi(\mathbf{x}_j)) + 2 \sum_{i \in I} \mu(\mathbf{x}_i) L(\Phi(\mathbf{x}_i)) = F$$

$$(1 - \alpha^2) \lambda(\mathbf{x}_j) = -c_j (B([\Phi(\mathbf{x}_j]]) \bar{\mathbf{a}} - g)$$

$$(1 - \alpha^2) \mu(\mathbf{x}_i) = -c_i (L([\Phi(\mathbf{x}_i]]) \bar{\mathbf{a}} - f)$$

(17)

where

$$F = \int_{\Omega} L(\Phi(\mathbf{x})) f(\mathbf{x}) d\Omega + \sum_{j \in J} 2c_j B(\Phi(\mathbf{x}_j)) g(\mathbf{x}_j) + \sum_{i \in I} 2c_i L(\Phi(\mathbf{x}_i)) f(\mathbf{x}_i)$$

$$A_2 = \sum_{j \in J} 2c_j B(\Phi(\mathbf{x}_j)) B([\Phi(\mathbf{x}_j]]) + \sum_{i \in I} 2c_i L(\Phi(\mathbf{x}_i)) L([\Phi(\mathbf{x}_i]]) \quad (18)$$

The functions $\lambda(\mathbf{x}_j)$ and $\mu(\mathbf{x}_i)$ correspond to the residual errors associated with the boundary collocation points and the interior collocation points respectively.

$$\lambda(\mathbf{x}_j) = -c_j \varepsilon(\mathbf{x}_j); \quad \mu(\mathbf{x}_i) = -c_i \xi(\mathbf{x}_i) \quad (19)$$

The version of the Uzawa algorithm corresponding to our formulation of the collocation least squares method for this particular problem reduces to:

$$\lambda^0(\mathbf{x}_j), \quad \mu^0(\mathbf{x}_i); \quad j = 1, 2, \dots, k$$

$i = k + 1, \dots, m$ specified arbitrarily

With $\lambda^n(\mathbf{x}_j)$ and $\mu^n(\mathbf{x}_i)$

known, calculate \mathbf{a}^n then $\lambda^{n+1}(\mathbf{x}_j); \mu^{n+1}(\mathbf{x}_i)$

$$(A_1 + A_2) \mathbf{a}^n + 2 \sum_{j=k+1}^m \lambda^n(\mathbf{x}_j) B(\Phi(\mathbf{x}_j))$$

$$+ 2 \sum_{i=1}^k \mu^n(\mathbf{x}_i) L(\Phi(\mathbf{x}_i)) = F$$

$$\lambda^{n+1}(\mathbf{x}_j) = \lambda^n(\mathbf{x}_j) + \rho_n (c_j (B([\Phi(\mathbf{x}_j]]) \mathbf{a}^n - g) + (1 - \alpha^2) \lambda^n(\mathbf{x}_j))$$

$$\mu^{n+1}(\mathbf{x}_i) = \mu^n(\mathbf{x}_i) + \rho_n (c_i (L([\Phi(\mathbf{x}_i]]) \mathbf{a}^n - f) + (1 - \alpha^2) \mu^n(\mathbf{x}_i)) \quad (20)$$

For a given value of the penalty coefficient c the above algorithm converges to the three following vectors (see section 4.2):

$$\bar{\mathbf{a}} = \lim_{n \rightarrow \infty} \mathbf{a}^n,$$

$$\lambda(\mathbf{x}_j) = \lim_{n \rightarrow \infty} \lambda^n(\mathbf{x}_j)$$

$$\text{and } \mu(\mathbf{x}_i) = \lim_{n \rightarrow \infty} \mu^n(\mathbf{x}_i).$$

It is important to note that the values $\frac{\lambda(\mathbf{x}_j)}{c_j}$ and $\frac{\mu(\mathbf{x}_i)}{c_i}$ correspond to the punctual error that one can allow at the collocation points \mathbf{x}_j and \mathbf{x}_i , respectively, and still respect the global mass balance.

B. Convergence of the proposed Algorithm

In this section, we present a study of the convergence of the algorithms proposed in this work for a constant ρ ($\rho_n = \rho$) and a constant c ($c_i = c_j = c$). Proof of convergence for the classic Uzawa's Algorithm may be found in ([15]). As far as the authors know, the proposed version of the Uzawa algorithm has not been studied yet.

The algorithm presented in the preceding section, in Equations (21) to (20), may be written in matrix form:

$$\xi^0; \text{ specified arbitrarily} \quad (21)$$

$$\text{With } \xi^n \text{ known, calculate } \mathbf{a}^n \text{ then } \xi^{n+1} \quad (22)$$

$$(A_1 + 2cT^T T) \mathbf{a}^n + 2T^T \xi^n = F \quad (23)$$

$$\xi^{n+1} = \xi^n + \rho(c(T\mathbf{a}^n - \mathbf{b}) + (1 - \alpha^2)\xi^n). \quad (24)$$

If one denotes by N the dimension of the unknown vector \mathbf{a} and by m the total number of collocation points, then the $m \times N$ matrix T , and the vectors ξ^n and \mathbf{b} , of dimension m are defined as:

$$T(\mathbf{x}_i) = \left\{ \begin{array}{ll} L([\Phi(\mathbf{x}_i)]) & \text{if } 0 \leq i \leq k \\ B([\Phi(\mathbf{x}_i)]) & \text{if } k+1 \leq i \leq m \end{array} \right\}. \quad (25)$$

$$\xi^n(\mathbf{x}_i) = \left\{ \begin{array}{ll} \lambda^n(\mathbf{x}_i) & \text{if } 0 \leq i \leq k \\ \mu^n(\mathbf{x}_i) & \text{if } k+1 \leq i \leq m \end{array} \right\}. \quad (26)$$

$$\mathbf{b} = \left\{ \begin{array}{ll} g(\mathbf{x}_i) & \text{if } 0 \leq i \leq k \\ f(\mathbf{x}_i) & \text{if } k+1 \leq i \leq m \end{array} \right\}. \quad (27)$$

$$(28)$$

The product $T^T \xi$ represents the following column vector of dimension N :

$$T^T \xi = \sum_{i=1}^m \xi(\mathbf{x}_i) \mathbf{T}(\mathbf{x}_i). \quad (29)$$

The convergence of the Algorithm (Equations (24)) is a consequence of the following theorem:

Theorem 5.1 (Theorem C): For any $\xi_0 \in \mathbf{R}^m$, the algorithm (Equations (24)) converges if and only if, for any positive eigenvalue of the matrix $A_1^{-1} T^T T$:

$$0 < \Lambda_1 < \dots < \Lambda_i \dots < \Lambda_{m_0} \quad (30)$$

$$\left| \frac{1 + 2c\Lambda_i + \rho(1 - \alpha^2(1 + 2c\Lambda_i))}{1 + 2c\Lambda_i} \right| < 1 \quad (31)$$

For these conditions and the restriction,

$$1 < \alpha^2 < 1 + \frac{2}{\rho}, \quad \rho > 0,$$

the following holds

$$\lim_{n \rightarrow \infty} \{\mathbf{a}^n, \xi^n\} = \{\bar{\mathbf{a}}, \bar{\xi}\},$$

where $\{\bar{\mathbf{a}}, \bar{\xi}\}$ is the unique solution of equations 17. The proof is given in Appendix A.

Here are two important remarks about Theorem C:

Remark 5.2: By the analysis of the mathematical curve, we

have:

$$\rho \rightarrow \left| \frac{1 + 2c\Lambda_i + \rho(1 - \alpha^2(1 + 2c\Lambda_i))}{1 + 2c\Lambda_i} \right| < 1, \quad (32)$$

and the proof of Theorem C, another equivalent criterion of convergence, may be found in terms of the smallest and the largest eigenvalues Λ_1 and Λ_{m_0} respectively.

The algorithm [(24)] converges if, and only if, the following two conditions are satisfied:

$$\Lambda_1 > \frac{1 - \alpha^2}{2\alpha^2 c} = \frac{1}{2c'} \quad (33)$$

$$0 < \rho < 2 \frac{2c\Lambda_{m_0} + 1}{\alpha^2(2c\Lambda_{m_0} + 1) - 1} \quad (34)$$

Equation (33) expresses the fact that a necessary condition of convergence of the present algorithm is that the smallest eigenvalue Λ_1 is greater than half of the inverse of the corrected penalty coefficient.

Remark 5.3: The optimal value of ρ for rapid convergence may also be found by the geometric considerations (See [15]).

$$\rho_{opt} = \frac{2P_{m_0}P_{m_1}}{2\alpha^2P_{m_0}P_{m_1} - P_{m_1} - P_{m_0}} \quad (35)$$

where :

$$P_{m_0} = 1 + 2c\Lambda_{m_0} \quad (36)$$

$$P_{m_1} = 1 + 2c\Lambda_1 \quad (37)$$

A consequence of this last result is that for large values of c , the optimal value of ρ is $\rho = \frac{1}{\alpha^2}$. In the general case, ρ and α are chosen simultaneously and must satisfy the inequality

$$1 < \alpha^2 < 1 + \frac{2}{\rho}, \rho > 0.$$

A comparison between the penalty algorithm and the PCG algorithm has been presented in [24].

VI. EXAMPLES

In this section we compare three different approaches:

- 1) The classical surface mesh approach
- 2) The parametric approach.
- 3) The point based approach using the penalty algorithm.

Two different functions were chosen: $f(x, y) = \frac{1}{x+y-1}$ for Case 1 (v.i. subsection VI-A and Figure 8) and $f(x, y) = \frac{1}{x^2-y^2}$ for Case 2 (v.i. subsection VI-B and Figure 11).

A. Case 1

Figure 8 illustrates the plots of the function using classical Cartesian regular mesh. Artifacts near the discontinuity line $x + y - 1 = 0$ appear.

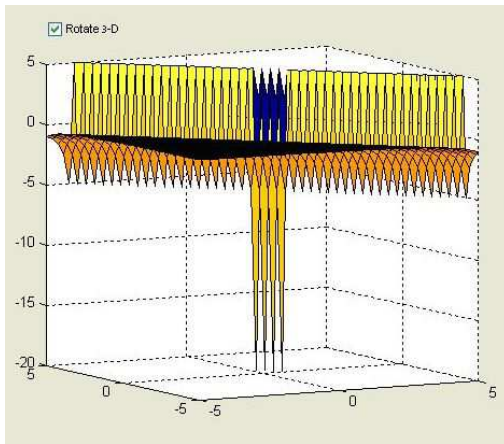


Fig. 8. Plot of $f(x, y) = \frac{1}{(x+y-1)}$ using surface plotting

When choosing the parametric representation $u = x+y; v = x-y$, the graph plot is regular (see Figure 9).

On Figure 10, the automatic algorithm based on the penalty method produces a regular plot without introducing parametric variables.

B. Case 2

Figure 11 illustrates the plots of the function using classical Cartesian regular mesh. Artifacts near the discontinuity line $x + y - 1 = 0$ appears.

When choosing the parametric representation $u = x+y; v = x-y$, the graph plot is regular (Figure 12).

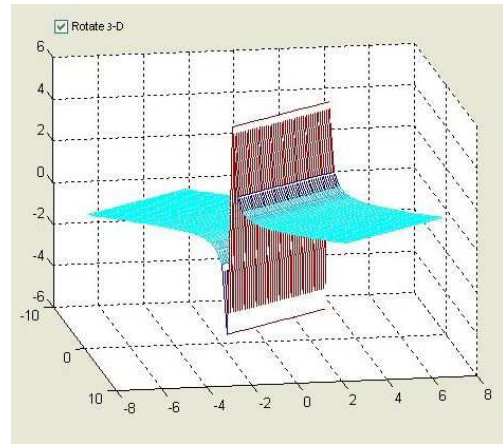


Fig. 9. Plot of $f(x, y) = \frac{1}{(x+y-1)}$ using parametric plotting

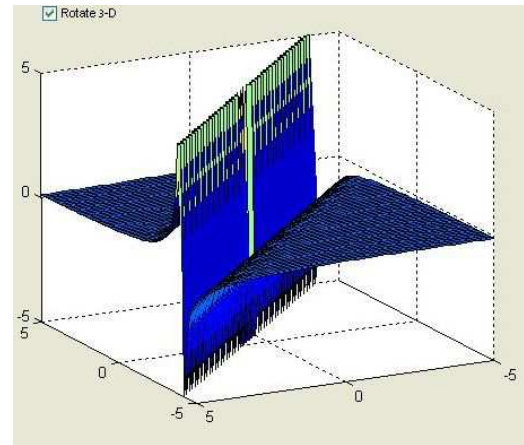


Fig. 10. Plot of $f(x, y) = \frac{1}{(x+y-1)}$ using control gradient algorithm

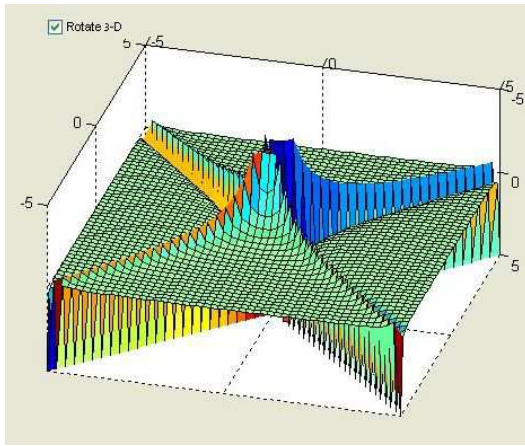
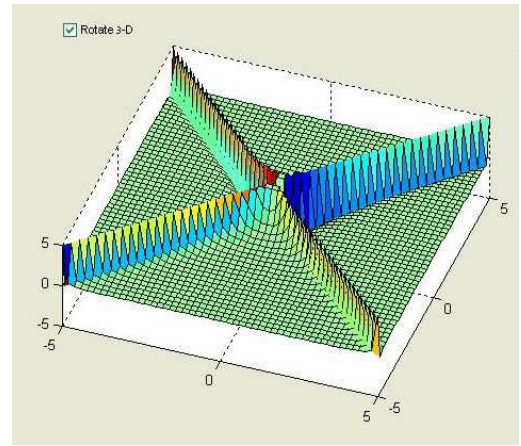
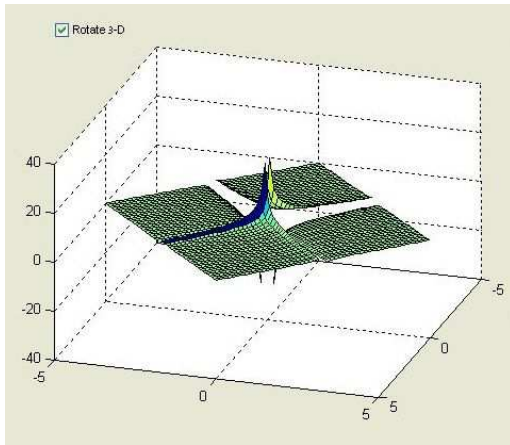
On Figure 13, the automatic algorithm based on the penalty method produces a regular plot without introducing parametric variables

1) *Convergence of the Algorithm: Case 1:* In all analysis convergence occurred in about ten iterations. Convergence was defined to occur when successive norms of the boundary residual were equal to six significant figures. It was found that optimal convergence was achieved with $\rho = 0.71$. For values greater than $\rho = 0.71$ non monotonic convergence occurs, while for values of ρ less than 0.71 monotonic convergence was observed.

Case 2: Table I illustrates the influence of the parameter ρ on the number of iterations when the weighting function c is large and the parameter α is very close to one. The number of iterations corresponds to an error on the boundary of less than 10^{-2} .

VII. CONCLUSION

An alternative least squares formulation has been proposed for the optimal allocation of points using a point based approach. The advantage of the present formulation is in achieving control of the local error at each collocation point


 Fig. 11. Plot of $f(x, y) = \frac{1}{(x^2 - y^2)}$

 Fig. 13. Plot of $f(x, y) = \frac{1}{(x^2 - y^2)}$ using Control gradient algorithm

 Fig. 12. Plot of $f(x, y) = \frac{1}{(x^2 - y^2)}$ using Parametric surface plotting

near discontinuity points without degrading the accuracy of the global surface meshing.

The practical advantages of this approach are the possibility of automatic plotting of 2D functions having discontinuities. The influence of the penalty weight c on the accuracy of the classic least squares formulation is found to be important. Moreover, we show the existence of optimal weights that improve the accuracy of the least squares.

In the new formulation proposed, the determination of this

TABLE I
INFLUENCE OF ρ ON THE NUMBER OF ITERATIONS FOR CASE 2 $c = 10^3$
AND $\alpha = 1 + 10^{-9}$

	$\rho = 0.5$	$\rho = 1$	$\rho = 1.5$	$\rho = 2$
Number of Iterations	50	25	17	13

optimal weighting is more easily found than in the classic method.

APPENDIX

For any $n \in \mathbf{N}$, there exists a unique decomposition of ξ^n :

$$\xi^n = \xi_1^n + \xi_2^n \quad (38)$$

where $\xi_1^n \in R(T)$ and $\xi_2^n \in Ker(T^T)$ (Kernel of T^T)

$$R(T) = \{q | q \in R^m, \exists v \in R^N \text{ such that } q = Tv\} \quad (39)$$

From equation (50); $\xi^{n+1} - [1 + \rho(1 - \alpha^2)]\xi^n \in R(T)$ and then :

$$\xi_2^{n+1} = [1 + \rho(1 - \alpha^2)]\xi_2^n = [1 + \rho(1 - \alpha^2)]^n \xi_2^0 \quad (40)$$

Let us define $\bar{\xi}^n$ and \bar{a}^n by:

$$\begin{aligned} \bar{\xi}^n &= \xi^n - (\bar{\xi} + [1 + \rho(1 - \alpha^2)]^n \xi_2^0) \\ \bar{a}^n &= a^n - \bar{a} \end{aligned} \quad (41)$$

where $\bar{\xi}$ and \bar{a} are solutions of the equations 17; ξ^n , a^n are solutions of equations 24.

Notice that by definition, $\bar{\xi}^n \in R(T)$ for all n , and if $|1 + \rho(1 - \alpha^2)| < 1$ (eq. 41) then $\lim_{n \rightarrow \infty} \bar{\xi}^n = 0$, and $\lim_{n \rightarrow \infty} \xi^n = \bar{\xi}$.

By subtraction of equations 24 and 17, one can easily see that :

$$(A_1 + 2cT^T T) \bar{a}^n + 2T^T \bar{\xi}^n = 0 \quad (42)$$

$$\bar{\xi}^{n+1} = (1 + \rho(1 - \alpha^2))\bar{\xi}^n + \rho c T \bar{a}^n \quad (43)$$

By elimination of \bar{a}^n between the two last equations, one can

obtain:

$$A_1^{-1}T^T\bar{\xi}^{n+1} = (1 + \rho(1 - \alpha^2))A_1^{-1}T^T\bar{\xi}^n - 2\rho cA_1^{-1}T^TT(I + 2cA_1^{-1}T^TT)^{-1}A_1^{-1}T^T\bar{\xi}^n \quad (44)$$

Let us define the sequence $\{z^n\}_{n \geq 0}$ by $z^n = A_1^{-1}T^T\bar{\xi}^n$, by definition of $\bar{\xi}^n, z^n \in R(A_1^{-1}T^TT)$ and :

$$z^{n+1} = [(1 + \rho(1 - \alpha^2))I - 2\rho cA_1^{-1}T^TT(I + 2cA_1^{-1}T^TT)^{-1}]z^n \quad (45)$$

Since $R(T)$ and $R(A_1^{-1}T^TT)$ are isomorphic, the convergence of z^n to zero will imply the convergence of $\bar{\xi}^n$ and \bar{u}^n to zero.

Consider now $\{\Lambda_i\}_{i=1}^{m_0}$, the strictly positive eigenvalues of $A_1^{-1}T^TT$ ordered such that:

$$0 < \Lambda_1 < \dots < \Lambda_i < \dots < \Lambda_{m_0}. \quad (46)$$

There exists a vector basis $\{w_i\}_{i=1}^{m_0}$ of $R(A_1^{-1}T^TT)$ such that

- 1) w_i is an eigenvector of $A_1^{-1}T^TT$ associated with Λ_i
 - 2) $((A_1w_i, w_j)) = 0$ for $1 \leq i, j \leq m_0$
- $((.,.))$ represent the scalar product on \mathbb{R}^N .

Since $z^n \in R(A_1^{-1}T^TT)$; $z^n = \sum_{i=1}^{m_0} z_i^n w_i$ and from equation 45 one can easily obtain :

$$z_i^{n+1} = \left[\frac{1 + 2c\Lambda_i + \rho(1 - \alpha^2(1 + 2c\Lambda_i))}{1 + 2c\Lambda_i} \right] z_i^n \quad (47)$$

This last relation implies that z^n converges to zero, if and only if :

$$\left| \frac{1 + 2c\Lambda_i + \rho(1 - \alpha^2(1 + 2c\Lambda_i))}{1 + 2c\Lambda_i} \right| < 1. \quad (48)$$

For these conditions and with the restriction $1 < \alpha^2 < 1 + \frac{2}{\rho}$ (see Equation 41), we have

$$\lim_{n \rightarrow \infty} \{a^n, \xi^n\} = \{\bar{a}, \bar{\xi}\}.$$

REFERENCES

[1] Amenta N, Bern M, Kamvysselis M. *A new Voronoi-based surface reconstruction algorithm*. In: Proceedings of ACM SIGGRAPH 98; 1998, 415-21.

[2] Avriel Mordechai, *Nonlinear Programming Analysis and methods*, Prentice-Hall, Inc, 1976.

[3] Axelsson O. and V.A. Barker, *Finite Element Solution of Boundary Value Problems, Theory and Computation*, Computer Science and Applied Mathematics, Academic Press, INC. 1984.

[4] Bensabat J. and D.G. Zeitoun, *A least squares formulation for the solution of transport problems*, Int. J. of Num. Meth. in Fluids, Vol 10, pp. 623-636, 1990.

[5] Bertsekas D. P., *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press 1982.

[6] Botsch M, Kobbelt L., *Resampling feature and blend regions in polygonal meshes for surface antialiasing*. In: Proceedings of Eurographics 01; 2001, 402-10.

[7] Bramble J.H. and J. Nitsche, *A Generalized Ritz- Least Squares Method for Dirichlet Problems*, S.I.A.M. J. Numer. Anal. **10** (1), 1973, 81-93.

[8] Bramble J.H. and A.H. Schatz, *Least Squares Methods for 2nd Order Elliptic Boundary Value Problems*, Math. Comp. **25** (113), 1971, 1-32.

[9] Bristeau M.O., O. Pironneau, R. Glowinski, J. Periaux and P. Perrier, *On the numerical solution of nonlinear problems in fluid dynamics by least squares and finite element methods*, Comp. Methods Appl. Mech. Eng. **17-18**, 1979, 619 - 657.

[10] Chen Tsu-Fen, *On least squares Approximations to Compressible Flow Problems*, Num. Meth. for P.D.E. **2** (1986), 207-228.

[11] Th. Dana-Picard: *Enhancing conceptual insight: plane curves in a computerized learning environment*, International Journal of Technology in Mathematics Education **12** (1), 33-43, 2005.

[12] Th. Dana-Picard, I. Kidron and D. Zeitoun: *To See or not To See II*, International Journal of Technology in Mathematics Education **15** (4), 157-166.

[13] Dos Santos S.R. and Brodlie K.W., *Visualizing and Investigating Multi-dimensional Functions*, IEEE TCVG Symposium on Visualization, 2002, 1-10. Joint EUROGRAPHICS.

[14] Eason E.D. *A review of least squares methods for solving partial differential equations*, Int. J. Num. Meth. Eng. **10**, 1976, 1021-1046.

[15] Fortin M. and R. Glowinski, *Augmented Lagrangian Methods: Applications to the numerical solution of boundary value problems*, Studies in Mathematics and its Applications **15**, North Holland, 1983.

[16] Golub G.H. and Van Loan C.F., *Matrix Computations*, The John Hopkins University Press, Baltimore, 1984.

[17] Jespersen D.C., *A least squares decomposition method for solving elliptic equations*, Mathematics of Computation **31** (140), 1977, 873-880.

[18] Kobbelt L. and Botsh M., *A survey of point-based techniques in computer graphics*, Computer and Graphics **28**, 2004, 801-814.

[19] Kobbelt L, Botsch M, Schwanecke U, Seidel HP. *Feature sensitive surface extraction from volume data*. In: Proceedings of ACM SIGGRAPH 01; 2001, 57-66.

[20] W. Koepf: *Numeric Versus Symbolic Computation*, Plenary Lecture at the 2nd Int. Derive Conf., Bonn, 1995. Available: <http://www.zib.de/koepf/bonn.ps.Z>.

[21] Lapidus L. and G.F. Pinder, *Numerical Solution of Partial Differential Equations in Science and Engineering*, John Wiley & Sons, 1982.

[22] Levy Bruno, "Constrained Texture Mapping for Polygonal Meshes", ACM SIGGRAPH 2001, 12-17 August 2001.

[23] Sermer P. and R. Mathon, *Least squares methods for mixed-type equations*, SIAM Journal of Numerical Analysis **18** (4), 1981, 705 - 723.

[24] Zeitoun D.G., Laible J.P. and G.F. Pinder, *An Iterative Penalty Method for the Least Squares Solution of Boundary Value Problems*, Numer. Meth. for P.D.E., Vol.13 (1997), 257-281.