Paper 61

©Civil-Comp Press, 2012
Proceedings of the Eighth International Conference
on Engineering Computational Technology,
B.H.V. Topping, (Editor),
Civil-Comp Press, Stirlingshire, Scotland

Visualizing Nonlinear Implicit Functions

I. Kožar
Department for Computer Modelling
Faculty of Civil Engineering
University of Rijeka, Croatia

Abstract

This paper describes a novel procedure for visualization (contouring) of implicit functions. It is based on a combination of the Newton algorithm and the 'arc-length' method. The new procedure has many advantages e.g. alleviates the singularity problem of the Newton algorithm. The resulting points that belong to the function are automatically ordered along the line (it can be difficult to properly connect the points resulting from some other algorithms). In addition, the new algorithm is self-starting and has no problems with multiple domains.
The developed contouring procedure can be extended to visualize space surfaces by intersecting them with a number of arbitrary planes. Such a procedure, when parallelised belongs to the 'embarrassingly parallel algorithms', i.e. the n-planes intersection problems are solved separately and asynchronously from each other.

Keywords: implicit functions, contouring problem, arc-length method, space surface visualization, parallel programming.

1 Introduction

Systems of nonlinear equations are quite a common task in engineering. However, nonlinear problems are never straightforward to solve and for educational and intuitive reasons it can be quite helpful to visualize the equations that we are trying to solve. Also, we could be interested in finding contours of a function given in the implicit form

$$F(x,y,z)=0 \tag{1}$$

The problem can be reduced to finding all solutions of the implicit function given as

$$F(x,y)=0 \tag{2}$$

where z is assumed constant (contouring problem).

Visualization of explicitly given functions is easily performed with most graphical packages available today. On the contrary, implicitly given functions could still present a challenge. One should not confuse visualization of three-dimensional implicitly given functions with visualization of parametrically given space functions (e.g. splines). In the later case it is enough to vary the function parameter and obtain the plotting points. To visualize Equation (1) or (2) one should somehow calculate the points that satisfy the given equation and plot them only afterwards.

There are several possibilities to make graph of the Equation (2). It is possible to convert the function into ordinary differential equation and solve it using one of many procedures available for solution of differential equations. It is also possible to obtain the formulation in a form of differential algebraic equations and proceed from that point on. However, transformation into differential forms is not without problems [1] and the possibility of existence of disconnected domains brings additional complications. There are also brute force methods like the marching squares method [2]. One could say that the method produces raster graphics and special care is required when connecting the resulting points.

The most intuitive procedure for graphical representation of implicit functions is modified Newton algorithm. In the presences of vertical and/or horizontal tangents that method does not converge. Application of the Newton algorithm and illustration of problems is presented in the subsequent part of the paper.

In order to alleviate problems in the Newton method a novel method for visualization of implicit functions is proposed. The method produces vector graphics and is based on the arc-length procedure that is well established in the field of computational mechanics. The method can be easily extended into three dimensions by separating the problem into independent tasks that produce function graphs in 'n' independent planes. The proposed algorithm has an additional benefit of easy parallelization. Moreover, it belongs to the so-called 'embarrassingly parallel algorithms'. MPI FORTRAN has been employed for calculation of points and results are collected in files that serve as input into a graphical presentation program.

Numerical examples of plane and space implicit functions illustrate the proposed method.

2 Visualization

2.1 Newton Method in Graph Processing

The usual procedure for graphing Equation (2) is to find corresponding y variables for selected x variables, only for implicitly given functions it is not straightforward. One of often-used procedures is Newton method that could be easily derived using Taylor's formula for function expansion. In the end we obtain the following iterative formula that for given x determines the corresponding y.

$$y_{i+1} = y_i - \frac{F(x_i, y_i)}{\frac{\partial F}{\partial y}} \tag{3}$$

Among the problems of this approach is that this equation is not self-starting. One needs an initial value for y and the solution is strongly determined by its choice. Generally, we know little about the function and the choice for initial value can become questionable. Also, it is evident that formula has problems in areas where the tangent is (almost) parallel to the x-axis (where $\partial F/\partial y \Rightarrow 0$). The problem manifests itself through lack of convergence and lack of possibility to use one equation for graphing the whole function [3]. The problem is illustrated in the following figure where we try to visualize the equation

$$F(x, y) = \cos(0.4y + x^2) + x^2 + y^2 - 1.6 \tag{4}$$

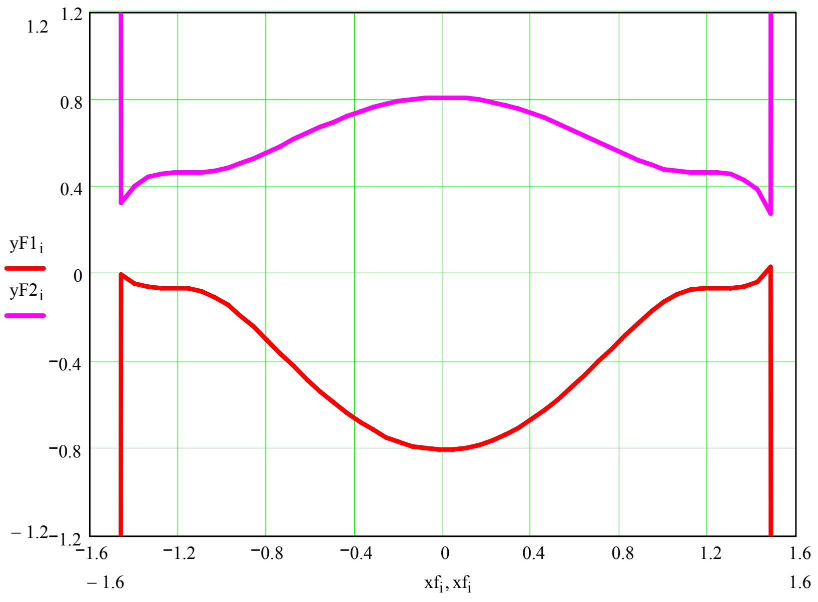

Figure 1: Attempt to graph the function using Newton algorithm.

Problems with convergence are represented with (nonexistent) vertical tangents in Figure 1. The bottom and the upper part of the graph were produced from two separate intervals as it is indicated with their different colours. To overcome the difficulties one could reformulate the procedure in a form where y is given and x is calculated. Complete graphs could be obtained combining both formulations on selected intervals. Those intervals could not be determined in advance so elaborate schemes should be used in automated computer procedures. The procedure remains sensitive and requires human intervention.

2.2 Arc-Length Method in Visualisation of Functions

2.2.1 Arc-Length Method in Graph Processing

Arc-length method was originally introduced by Riks and Wempner in 70's and has been in use in structural engineering for over 20 years; a more detailed history can be found in [4].

The original arc-length method is modified in order to be applied in visualization of functions but the principle remains the same: additional equation is introduced to avoid singularity of the tangent operator (Jacobian). The additional equation that represents constraint on both increments reads

$$a(\Delta x, \Delta y) = \Delta x^2 + \Delta y^2 - \Delta l^2 \tag{5}$$

where Δl is an arc radius (sort of scaling parameter). One advance 'along' the function in steps of predetermined length Δl (but without knowing the exact values of Δl_x and Δl_y in advance). Acquiring notation in which {.} parenthesis represent vectors and [.] matrices the solution procedure now looks like

$$\begin{Bmatrix} \Delta x_{i+1} \\ \Delta y_{i+1} \end{Bmatrix} = \begin{Bmatrix} \Delta x_i \\ \Delta y_i \end{Bmatrix} - \begin{bmatrix} \dfrac{\partial F}{\partial x} & \dfrac{\partial F}{\partial y} \\ 2\Delta x_i & 2\Delta y_i \end{bmatrix}^{-1} \begin{Bmatrix} F(x+\Delta x_i, y+\Delta y_i \\ a(\Delta x_i, \Delta y_i) \end{Bmatrix} \tag{6}$$

The main advantage is that the tangent matrix never gets singular. (neither Δx nor Δy could equal zero at the same time and the same is truth for $\partial F/\partial x$ and $\partial F/\partial y$). The other advantage is that the equation is self-starting. We could use any point P(x,y) as a starting point for our graph providing that arc-length Δl is large enough to reach the function from that point. If that is not the case, we could simply (automatically) extend Δl. During calculation Δl could change its value in order to improve the speed. However, while being very robust in calculation of function points, the self-starting property appeared to be too sensitive and another algorithm has been developed for determination of the first function point. Properties and possibilities of the method will be illustrated in the following examples.

2.2.2 Stability of the Method

In the analysis of the stability of the proposed procedure, we should treat separately the problem of determination of the first point on the function and the problem of determination of all the other points.

During experimentation, it has been found that one should not try to use as few increments as possible since that requires large Δl that leads to poor graphs and could result in skipping parts of the graph.

2.2.3 Determination of the First Point

Determination of the first point is crucial for success of the algorithm and its ease of use. Although we could use the self-starting capability in determination of the first point, it has been found that the behaviour of the procedure is not easy to predict. After some analysis it has been concluded that the source of the problem is the fact that the procedure includes tangents on the function (tangent matrix includes derivatives) and the point of convergence could be rather away from the starting point. In order to have the convergent point where we expect it, we should use some algorithm that is not based on tangents. Based on the secant method [5] an algorithm has been developed that can reliably detect a point on the intersection of a function with a given line.

The idea of the algorithm is based on the change of sign of a function F(x,y) on the opposite sides of the zero curve. The first part of the algorithm determines if there is an intersection between the curve and the circle with the centre at point (p,q). After we have detected existence of an intersection we are halving the line that crosses the curve until length of the line is within the prescribed tolerance. That algorithm is linear convergent and stable. The example of its use is illustrated with the following figure

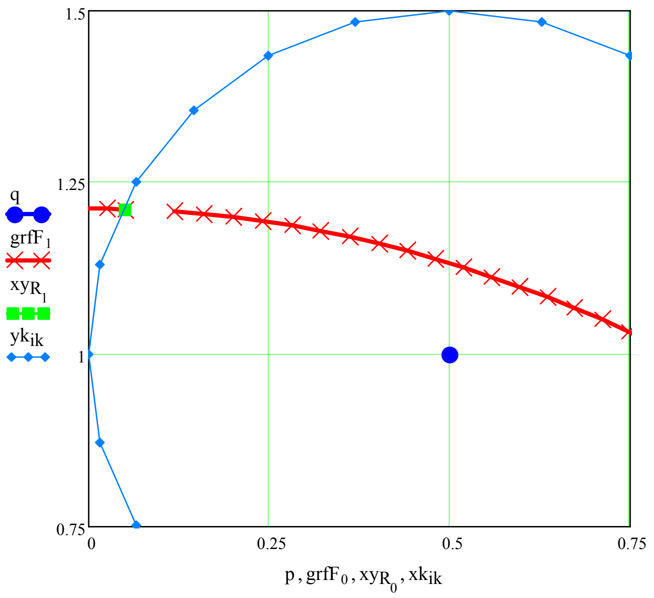

Figure 2: Graph of the intersection of a function and a starting point.

If detection is unsuccessful, the radius of the initial circle automatically increments itself (by a half). Introduction of the algorithm for starting point and use of an appropriate net of points for centres of initial circles allows for easy detection of unconnected parts of curves. One example of such a curve is described with the function

$$F(x,y) = \sin(4y+4x)^2 + x^2 + y^2 - 1.6 \tag{7}$$

and the graph is

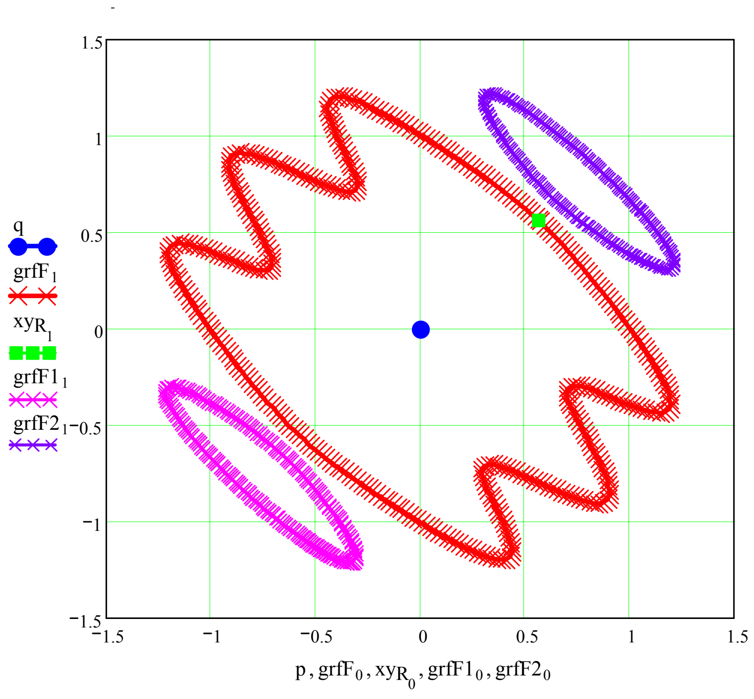

Figure 3: Graph of a function with multiple domains.

Starting points in Figure 3 have been determined from a mesh of different circle centres after which all the curve points have been automatically calculated.

2.2.4 Treatment of Singularities

There are several types of singularities that are possible in implicitly given functions. Only bifurcation point will be mentioned here. Generally, difficulties concerning singularities are dealt with through use of direction vectors. Two vectors are formed between the previous and the current and between the current and the future points. The future point is selected so that an angle between the vectors is closest to 2π.

This approach is demonstrated in Figure 4. The bifurcation point is detected as a case with more than one solution. One line is determined as an extension from the previous point and the other one as a connection from one new solution point to the other. In both cases we obey the principle to keep the angle as close to 2π as possible.

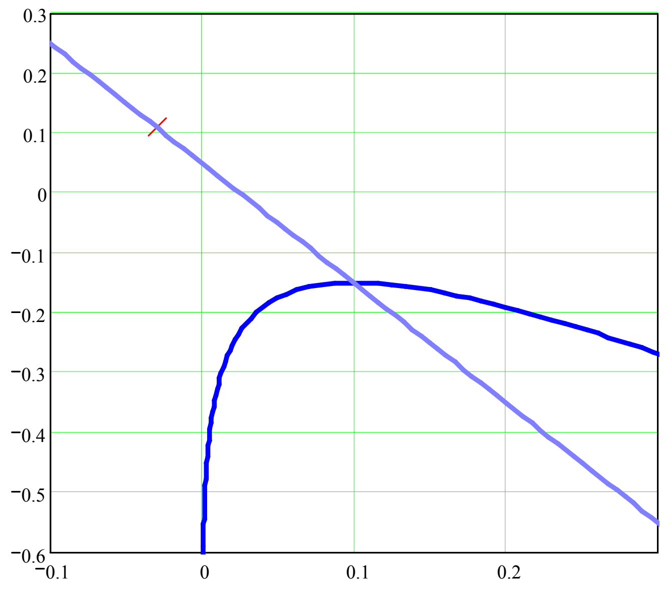

Figure 4: Graph with a bifurcation.

3 Examples

3.1 Graph in 2-D

Figure 5 shows how should Equation (4) look on the graph

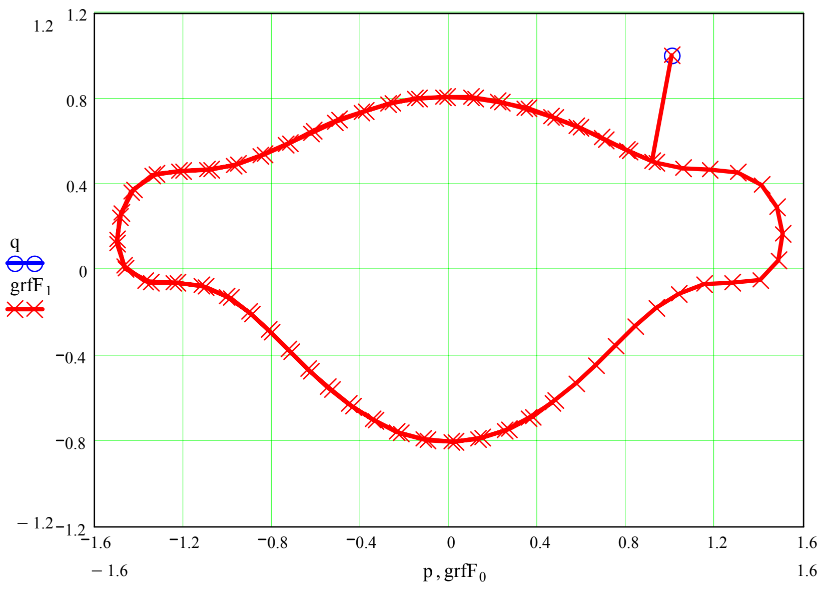

Figure 5: Complete graph of the Equation (4).

The following 2D function is ‘difficult’ in a sense that there are many sharp changes in curvature

$$F(x,y)=\cos(4y^2+x^2)^2+x^2+\exp(x+0.5y)+0.1y^2-5 \tag{8}$$

The graph is

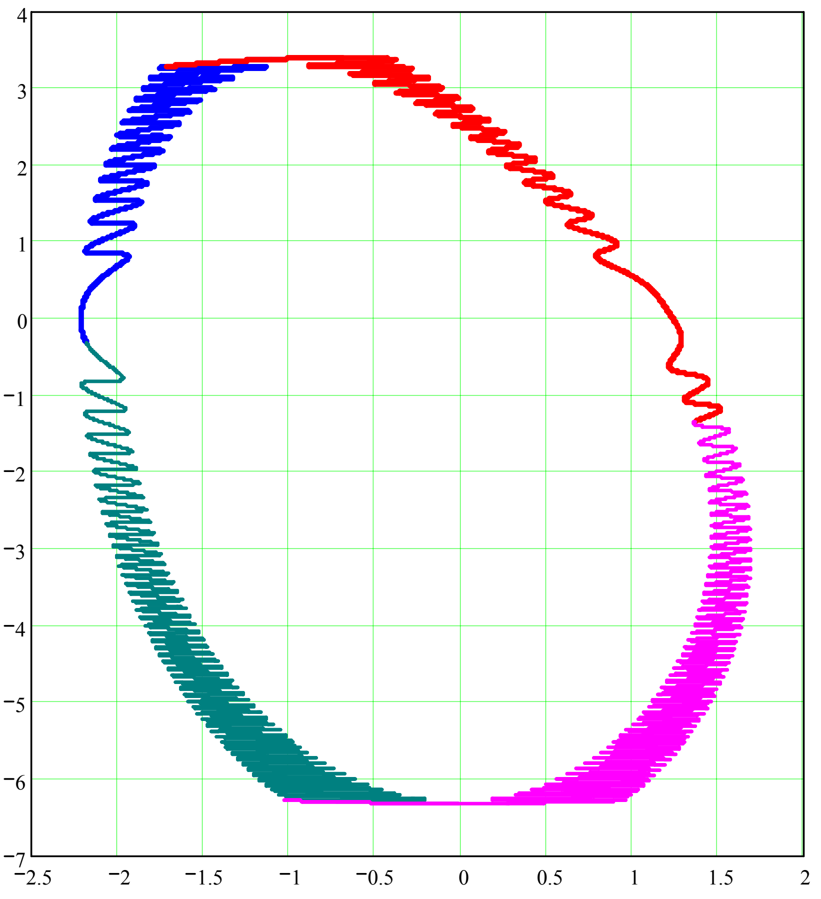

Figure 6: Graph of a “difficult” function.

Function in Figure 6 has been obtained with 4 different starting points and the corresponding parts are in different colours.

Finally there is a nice graph of the following equation

$$F(x,y)=1.6\cos(5y-\frac{1}{5}x^4+x^2)^2-\frac{0.005}{x^2}+0.4x^2-\frac{3}{2}y\cos(x)+\frac{1}{4}y+0.9y^2-2.2 \tag{9}$$

The graph is depicted in Figure 7 and it looks like a Christmas tree!

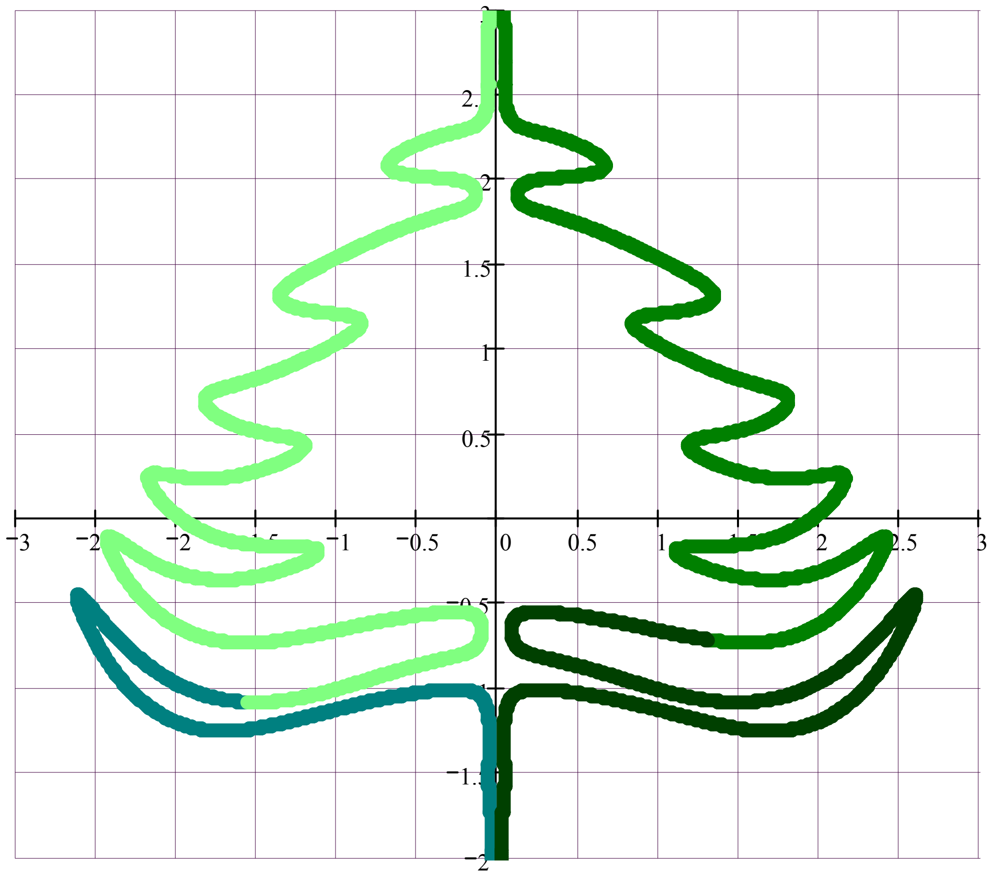
Figure 7: Graph of the Christmas Equation (9).

3.2 Graph in 3-D

The method can be easily extended into three dimensions if we adopt an intersection of a surface and an arbitrary plane as a way of visualizing space curves. Since the plane equation is linear it is easy to substitute one unknown (for instance z) into the surface equation and obtain an equivalent two-dimensional problem. The new equation is traced (solved) using the same procedure as before, only this time we are tracing the XY projection of the intersection curve (if z was the substituted variable). In the same manner we could trace the curve in XZ and YZ planes.

Plotting algorithm is parallelized by intersecting space function with a mesh of planes parallel to the coordinate planes. This approach enables asynchronous execution of the same program on different data. This type of algorithm belongs to the so-called embarrassingly parallel algorithms [6]. Speed-up is obtained if each computer node employs several processors and the execution program incorporates MPI routines. First, pool of tasks is formed and then MPI programs read from it and write results. The pool of tasks is formed using the appropriate data structure.

Graph of the following equation is presented as an example

$$F(x,y,z) = \cos(2y^2 + 5x^2) + x^2 + 2y^2 + z \tag{10}$$

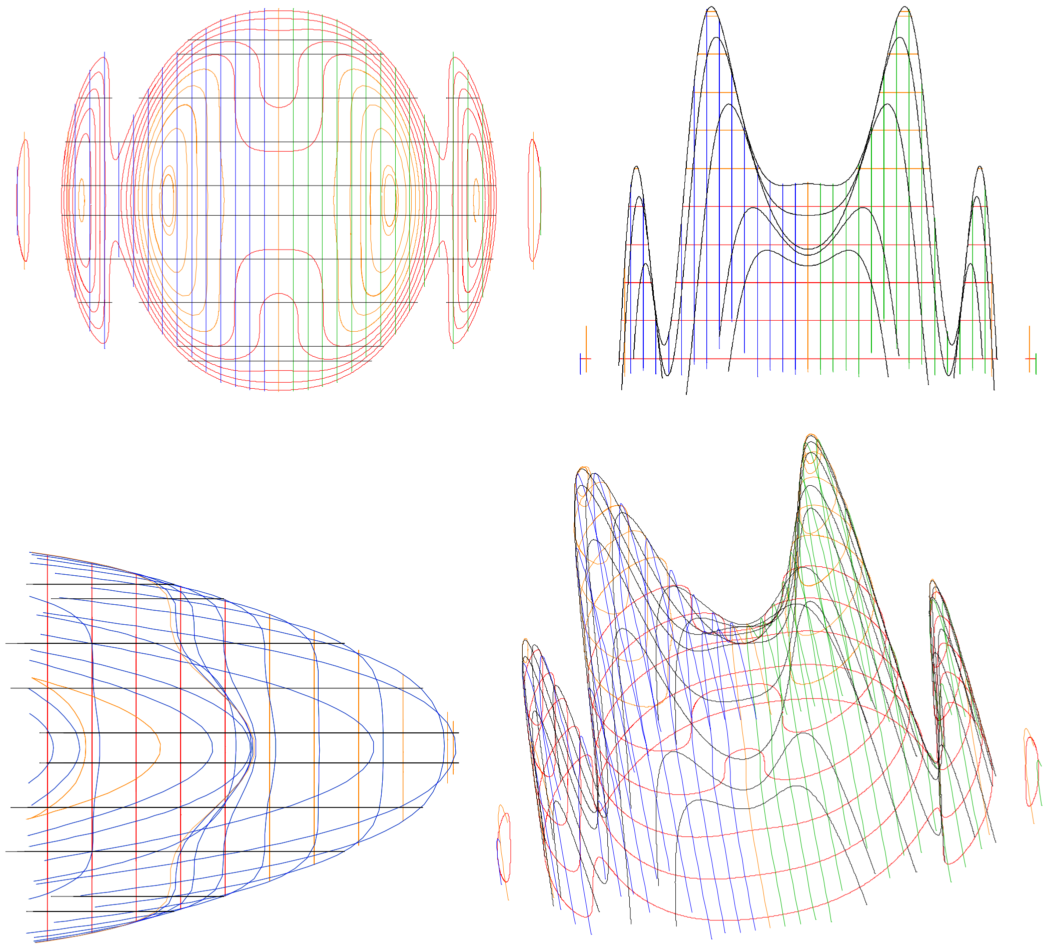

Figure 8: Graph of the 3D Equation (10) (XY, XZ, YZ and 3D views).

Graph in Figure 8 has been produced using parallel MPI code and consists of about 20000 points.

4 Conclusions

In this paper we have presented a novel method for producing graphs of implicitly given functions. The method has been compared with commonly used Newton method and was demonstrated that it poses superior characteristics. Examples have been presented as an illustration of the method. The method is adaptive as the arc length Δl can extend or shorten itself automatically. Space functions are plotted using a net of orthogonal planes each containing a contour plot of the given function. This method represents embarrassingly parallel algorithm that can be run on a cluster of computers. An appropriate parallel program based on a message passing paradigm has been outlined.

References

[1] J. Ockendon, S. Howison, A. Lacey, A. Movchan, “Applied Partial Differential Equations”, Oxford University Press, 2003.

[2] T.S. Newmann, H. Yi, “A survey of the marching cubes algorithm”, Computers & Graphics, 30(5), 854-879, 2006.

[3] I. Kožar, “Visualizing non-linear implicit functions”, in “Proceedings of MIPRO 2007: International Convention on Information and Communication Technology, Electronics and Microelectronics”, (Editors: K.Skala, L.Budin), Opatija - Croatia (2007), 79-83, 2007.

[4] M. Crisfield, “Non-linear Finite Element Analysis of Solids and Structures”, Volume 1, Wiley, Chichester, 1991.

[5] P. R. Brent, “Algorithms for Minimization without Derivatives”, Dover Publications, 2002.

[6] L. Budin, K. Skala, “Parallel programming and cluster computing”, (in Croatian), Proc. MIPRO 2005 (International Convention on Information and Communication Technology, Electronics and Microelectronics), Opatija – Croatia, 2005.