# DISPLAY OF THREE-DIMENSIONAL CURVILINEAR SURFACES

Dr. J. Raymond

Mr. P. Tymchyk

Mr. H. Vandette

Department of Computer Science

University of Ottawa

## Résumé

Un programme conversationel pour visualiser des surfaces curvilignes a été mis en oeuvre sur un ordinateur IBM/360. On utilise le pupitre de commande Tektronix 4010 pour l'introduction de données ainsi que pour la visualisation de la surface. La surface désirée est décrite par l'introduction des trois équations: $X = f_1(U,V)$, $Y = f_2(U,V)$ et $Z = f_3(U,V)$. La notation et la syntaxe ressemble à celle du Fortran. Le programme demande les équations et ensuite visualise la surface.

D'autres interactions avec le système peuvent prendre place par l'introduction de l'échelle, du domaine et du pas des paramètres, la position de l'oeil et la suppression des arrêtes cachées.

## Abstract

An interactive program to display any curvilinear surface has been implemented on an IBM/360 machine, using a Tektronix 4010 for input and display. The surface is described by its 3 equations: $X = f_1(U,V)$, $Y = f_2(U,V)$ and $Z = f_3(U,V)$. The syntax and notation used are similar to Fortran. The program asks for the equations and displays the corresponding surface on the screen.

Further interactive action can be taken by defining the scale, range and the step variations of the parameters, the eye position, and the removal of the hidden lines.

The basic objective in the development of this system was to produce a user-oriented display package, where the inexperienced user could easily display any 3-D curvilinear surface he desires.

The process begins with the input of three functions on command from the system. The system queries the user for these functions by displaying 'X=', whereby the 'X' function is input in terms of U and V. Functions must be written according to Fortran syntactical rules. It is important to note that all arithmetic routines are executed in floating-point. The user terminates his function definition with an End of Transmission code (EOT). The defined function is then compiled and checked for errors. If an error should be found, the appropriate error message is displayed and a corrected function is requested. If no errors are detected the equation has been properly compiled. The above procedure is then repeated for the definition of the 'Y' and 'Z' functions.

The following is an example of three such input functions:

```
X = U + 2.5
Y = V
Z = EPT(-SQRT(U**2+V**2))*COS(SQRT(U**2+V**2))
```

We are now in a position to clearly define the domain and range of the input functions within the region of interest. This is done by initializing the following seven parameters.

The first parameter the system requests is the scale, which is displayed as follows: 'ECHELLE =' .
The system scans the input to determine whether the input value is a signed digit lying in the interval -9 to +9; all subsequent characters are ignored. If an unsigned digit is found, it is assumed to be positive. The scale is then calculated to be 10 raised to the power of the input value.

| | | |
|---|---|---|
| E.G. | ECHELLE = -4 | scale is $10^{-4}$ |
| | ECHELLE = +2 | scale is $10^{+2}$ |
| | ECHELLE = 2 | scale is $10^{+2}$ |
| | ECHELLE = 25 | scale is $10^{+2}$ |
| | | (Note: The 5 is ignored) |

An appropriate error message is displayed when any other sequence of characters is encountered and a corrected scale value is requested. The value computed for the scale is then used to determine the unit length on the X, Y and Z coordinate axes.

The second parameter the system requests is the initial value of U. This is displayed in the form 'ORIGINE DE U = ' .

The user is now called upon to input a signed or unsigned (assumed positive) floating or integer number which contains at least one, but at most eight digits. A number containing more than eight digits will be rightmost truncated. Characters other than '+', '-', '.' (decimal point), and digits will result in the generating of an error message. The user is then requested to input a corrected value. All input values of the subsequent parameters are subject to the above stated criteria.

The third parameter requested is the upper limit of U. This is displayed as follows: 'DESTINATION DE U = ' .

The fourth parameter requested is the step-size of U, which is displayed as follows: 'PAS DE U = ' .

If the following criterion for the second, third and fourth parameters is not satisfied, the system will demand a re-initialization of the second, third and fourth parameters.

$$(\text{Initial value of U}) + n*(\text{Step-size of U}) \geq (\text{Upper limit of U})$$

$$\text{where } n \geq 1, \ (\text{Step-size of U}) \neq 0$$

It is essential that the above equation be satisfied, otherwise

(a) (Initial value of U) = (Upper limit of U)

or (b) (Upper limit of U) will never be reached with the given (Initial value of U) and (Step-size of U).

The fifth, sixth and seventh parameters repeat those of the second, third and fourth but in terms of V.

An interactive program to display any curvilinear surface has been implemented on an IBM/360 machine, using a Tektronix 4010 for input and display. The surface is described by its 3 equations: $X = f_1(U,V)$, $Y = f_2(U,V)$ and $Z = f_3(U,V)$.

It is desirable that $f_1(U,V)$ and $f_2(U,V)$ be orthogonal, so as to produce a well defined drawing. The most simple form of orthogonal curves is: $f_1(U,V) = U$ and $f_2(U,V) = V$.

A sufficient number of points X, Y and Z is calculated by keeping U constant at each step within its range while the value of V is varied within its domain. The same is done for V remaining constant and U varying.

The X, Y and Z axes are represented with angles of 120 degrees. Therefore, transformation into 2 dimensions is straightforward.

The screen used was 1024X780 points. The (0,0) point used was the

lower left hand corner of the screen. Thus the origin of the graphs were located at the position (512,390). The scale desired by the user was made to represent half of the vertical dimension of the screen (390 points).

The removal of hidden lines, a more complex problem, is certainly of great value to the user and, therefore, shall be implemented in the system.

References:

B. Kubert, J. Szabo and S. Giulieri, "The Perspective Representation of Functions of Two Variables", Journal of the Association for Computing Machinery, Vol. 15, No. 2, pp. 193-204, April 1968.

Thomas J. Wright, "A Two-Space Solution to the Hidden Line Problem for Plotting Functions of Two Variables", IEE Transactions on Computers, Vol. C-22, pp. 28-33, January 1973.

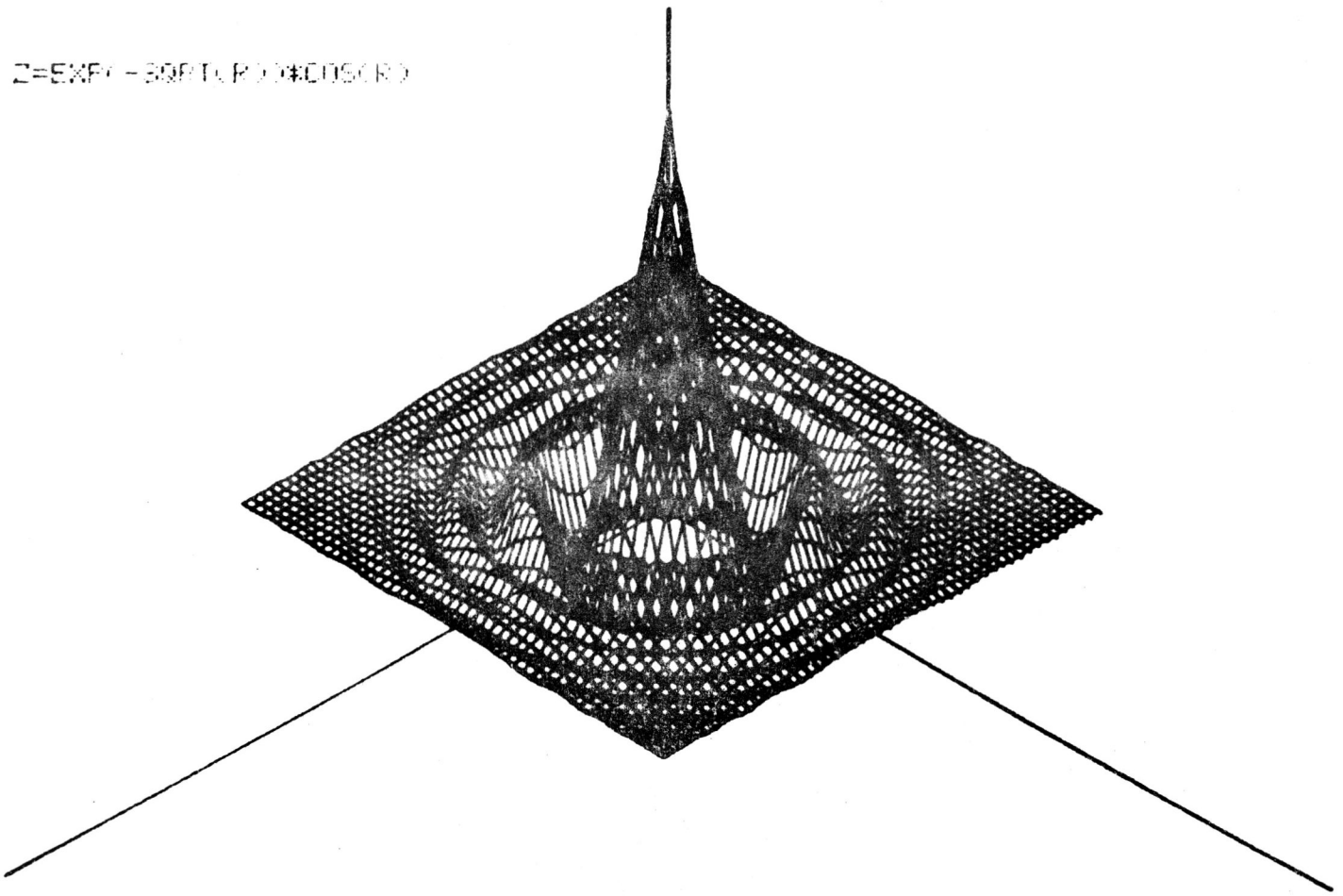


Fig. 1 example of output

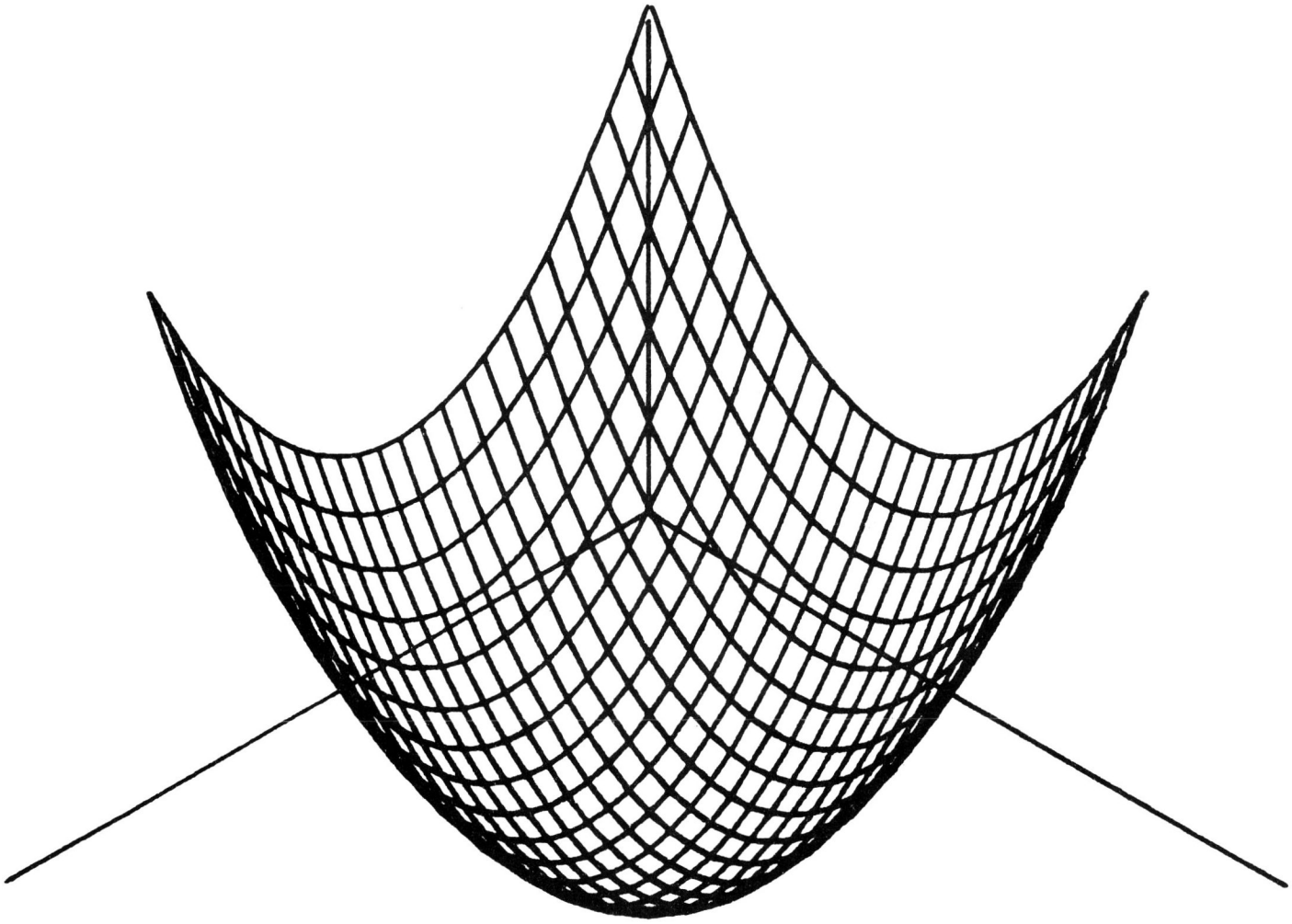

Fig 2 Paraboloid

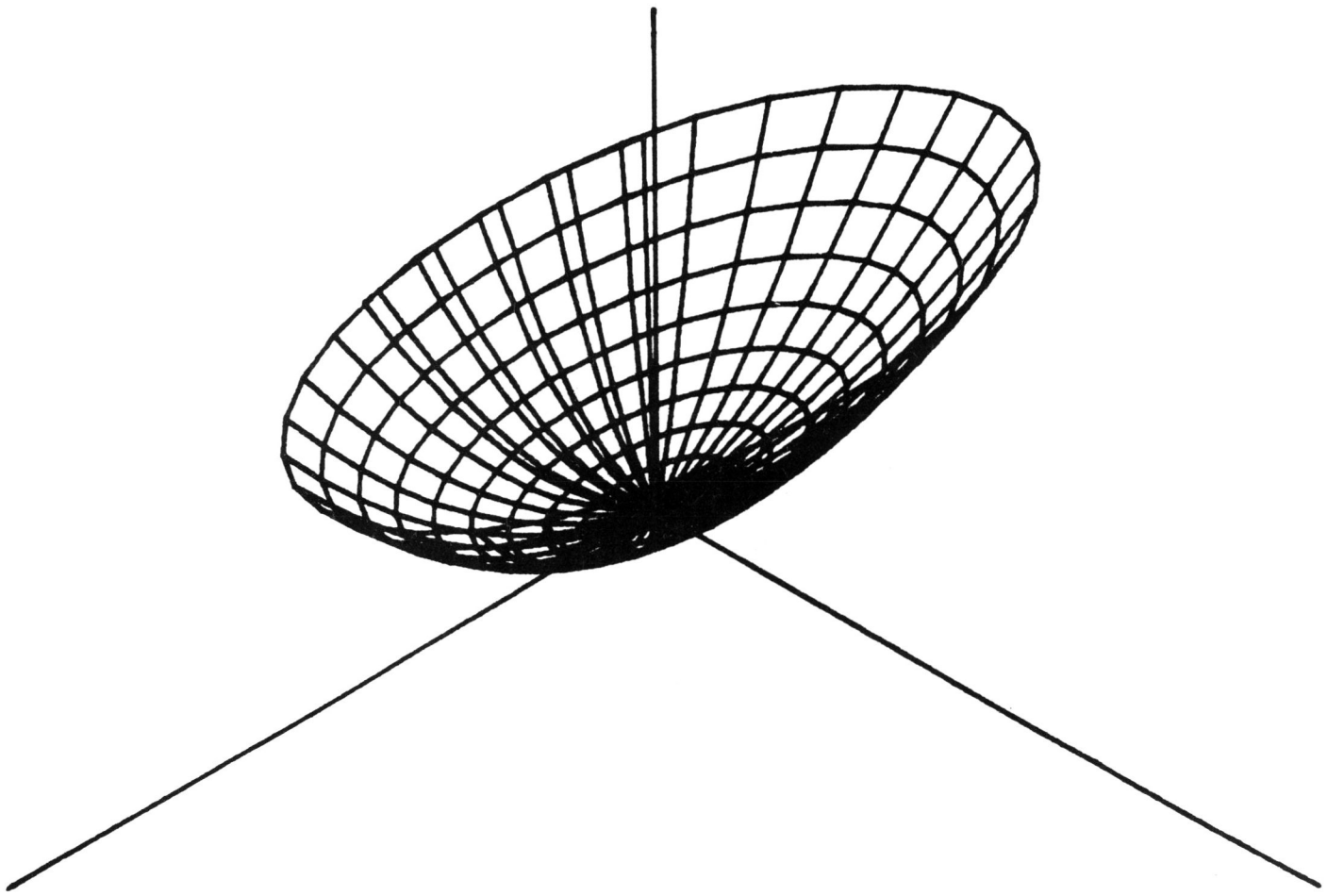

Fig 3 Ellipsoid 1

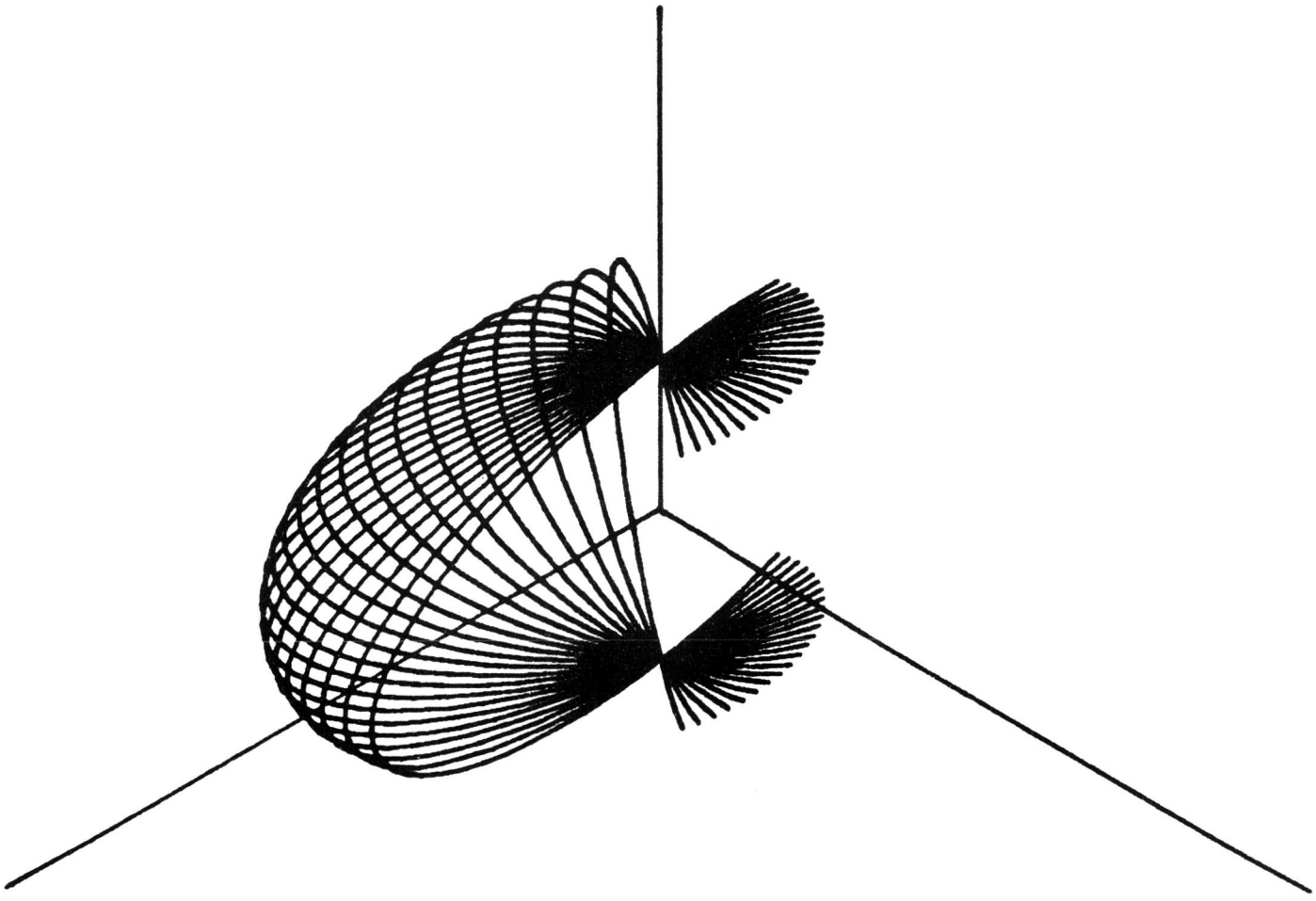

Fig 4 Ellipsoid 2