From 2D to 3D: Numerical Grid Generation and the Visualization of Complex Surfaces

Bonita Saunders
Qiming Wang

National Institute of Standards and Technology
100 Bureau Drive Stop 8910
Gaithersburg, MD 20899-8910
bonita.saunders@nist.gov, qiming.wang@nist.gov

Abstract

The widespread use of high level mathematical functions to solve problems in the mathematical and physical sciences has led the National Institute of Standards and Technology to engage in a massive project to update and expand the National Bureau of Standards Handbook of Mathematical Functions [1]. The handbook, which discusses the definition and computations of "special functions", will be disseminated on the World Wide Web as the NIST Digital Library of Mathematical Functions.

A key feature of the digital library will be 3D visualization capabilities that allow a user to interactively examine the unique features of complicated mathematical functions. The complex nature of these functions makes the visualization task quite difficult. Many contain singularities and poles which make the computational domains irregular, discontinuous, or multi-connected. This paper discusses the use of grid generation techniques to facilitate the plotting of the complicated 3D surfaces that represent these higher mathematical functions.

Introduction

High level mathematical functions such as the Bessel functions, the gamma and beta functions, hypergeometric functions and others are important for solving many problems in the mathematical and physical sciences. For example, the Airy functions Ai and Bi, which are Bessel functions of fractional order, provide closed form solutions to field equations that arise in quantum mechanics, optics and electromagnetism. The gamma and beta functions provide the starting point for the computation of more complex functions such as the Riemann zeta function and others that occur in number theory and mathematical physics. Because of their importance, references which discuss the definition and computations of these "special functions" continue to be widely used. One such reference is the National Bureau of Standards (NBS) Handbook of Mathematical Functions [1]. Its popularity has led the National Institute of Standards and Technology (NIST), the successor organization to NBS, to begin a large scale project to update and expand the handbook and disseminate it on the World Wide Web as the NIST Digital Library of Mathematical Functions (DLMF). A key feature of the DLMF will be 3D graphics and visualization capabilities that allow a user to interactively examine the unique features of complicated mathematical functions.

The complex nature of these functions makes the visualization task quite difficult. The presence of singularities and poles usually means that the computational domain will be irregular, discontinuous, or multi-connected. This paper discusses the use of grid generation techniques to facilitate the plotting of surfaces that are the graphs of functions, that is, surfaces that can be described by equations of the form $z = f(x, y)$. Some commercial packages have a few of the functions built in and may allow the user to produce a plot, usually over a rectangular cartesian mesh. However, this often produces a very poor and in many cases misleading graph of the function. In addition, the packages often have problems clipping the surface properly when values fall outside the range of interest specified by the user. Furthermore, we have often found that what looks satisfactory inside the package, may not when we transform the data to a format more suitable for interactive graphics on the Web.

We are looking at various techniques for solving the problems we have encountered using commercial packages. These include using a computational grid whose boundary coincides with the contours of the surface, adapting the grid lines to obtain more concentration in areas of large curvature, or designing the entire coordinate system so that the grid lines correspond to contours of the surface and the curves orthogonal to the contours. This

paper discusses the grid generation techniques that have been tried to date and in particular looks at the effectiveness of an updated version of a tensor product B-spline grid generation algorithm designed by one of the authors [2]. The feasibility of using unstructured techniques and the affect of their use on the translation of the data to other formats are also discussed.

3D Visualization in a Web-Based Digital Library

For the interactive visualizations in the DLMF we begin with a preprocessing stage, using available packages such as MATLAB, MAPLE and MATHEMATICA to plot the data so that we can examine the graphical representation and adjust the scaling to bring out interesting features. The data is then converted to VRML (Virtual Reality Modeling Language) format. VRML [3] is a standard 3D file format for describing the behavior and geometry of a 3D virtual world, or scene. Its accessibility on the Internet and interactive capabilities make it an ideal candidate for this development work. It is not a foregone conclusion that the final version of the DLMF will use VRML. This may depend on whether VRML browsers continue to be readily available. We are looking at alternatives to VRML such as JAVA 3D which would not require the download of a browser, but still would require the user to obtain the graphics package. In the mockup DLMF already developed and available for viewing at *http://dlmf.nist.gov*, the user has the option of viewing a still 3D image if a VRML browser is not available. Figure 1 shows a VRML display from the prototype chapter on Airy functions in the mockup Web site. The display shows $|\mathrm{Ai}(z)|$. The CosmoPlayer browser controls allow the user to rotate the figure, zoom in and out, and move the figure in an arbitrary direction. We have added custom vcr type controls that let the user move a cutting plane through the surface and observe the motion of the intersection curve. When investigating commercial packages we were surprised to discover that many do not perform 3D clipping properly when points fall outside the plotting range. In some cases the default method of clipping is to reset values outside the plotting range to the same constant. This produces the misleading shelf effect seen in the Mathematica plot of $|\mathrm{Bi}(z)|$ over an equally spaced rectangular domain in Figure 2. This technique is extensively used by William J. Thompson in **Atlas for Computing Mathematical Functions** [4]. By computing the function over a grid whose boundary matches a contour of the function, this problem can be eliminated. Also, the contour fitted grid tends to produce a smoother shading when the data is translated to VRML format.

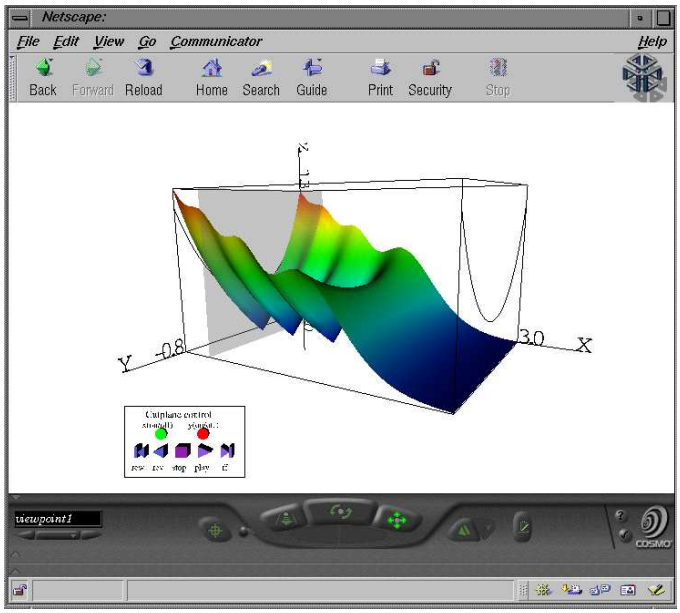

Figure 1: VRML display on CosmoPlayer.

Grid Generation Techniques

To date two techniques have been used to create the grids used to develop the surface plots: simple transfinite interpolation and a tensor product spline algorithm developed by one of the authors [2]. The tensor product algorithm uses the mapping

$$\mathbf{T}(\xi,\eta) = \begin{pmatrix} x(\xi,\eta) \\ y(\xi,\eta) \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{m}\sum_{j=1}^{n} \alpha_{ij} B_{ij}(\xi,\eta) \\ \sum_{i=1}^{m}\sum_{j=1}^{n} \beta_{ij} B_{ij}(\xi,\eta) \end{pmatrix}, \tag{1}$$

where $0 \le \xi, \eta \le 1$ and each B_{ij} is the tensor product of cubic B-splines. Hence, $B_{ij}(\xi,\eta) = B_i(\xi)B_j(\eta)$ where B_i and B_j are elements of cubic B-spline sequences associated with finite nondecreasing knot sequences, say, $\{s_i\}_1^{m+4}$ and $\{t_j\}_1^{n+4}$, respectively. The initial coefficients are chosen so that the mapping approximates transfinite blending function interpolation. More specifically, the coefficients are selected to produce a variation diminishing spline approximation to the transfinite blending function interpolant. In short, this means the coefficients are obtained by evaluating the interpolant at average knot values as discussed in [5]. If more orthogonality and

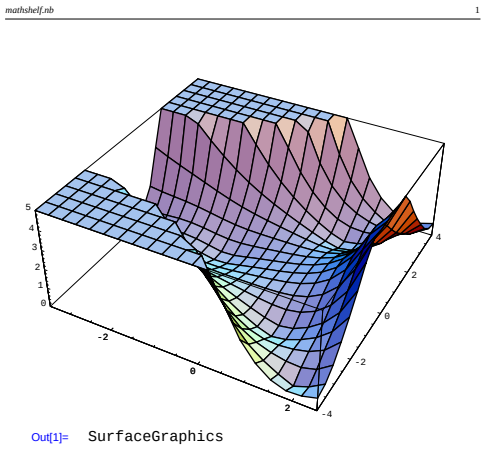

Figure 2: Clipped version of $|\mathrm{Bi}(z)|$ using Mathematica.

smoothness are needed, the coefficients can be adjusted to minimize the discrete functional

$$\begin{aligned} G &= \sum_{i,j} w_1 \left[\left(\frac{J_{i+1,j} - J_{ij}}{\triangle \xi} \right)^2 + \left(\frac{J_{i,j+1} - J_{ij}}{\triangle \eta} \right)^2 \right] \triangle \xi \triangle \eta \\ &\quad + \sum_{i,j} w_2 Dot_{ij}^2 \triangle \xi \triangle \eta \end{aligned} \tag{2}$$

where J_{ij} is the Jacobian value and Dot_{ij} is the dot product of $\partial \mathbf{T}/\partial \xi$ and $\partial \mathbf{T}/\partial \eta$ at mesh point (ξ_i, η_j) on the unit square. The code has been updated to handle larger problems. Both the transfinite code and the tensor product spline code allow the user to easily change the size of the grid while guaranteeing that certain boundary grid points are fixed to maintain the accuracy of the boundary approximation.

Results

The results achieved to date have been very promising. Unfortunately, one of the difficulties has been obtaining accurate contour data. Many commer-

cial packages display very accurate contours if enough points are requested, but outputting the data and translating it into boundary data that can be input into grid generation code can be a time consuming process. The grid on the right in Figure 3 has a boundary formed by connecting the $Z = 5$ contour curves of Airy function $|\mathrm{Bi}'(z)|$. Although a simple transfinite interpolation map was used to create the boundary fitted mesh shown, a modified map which interpolated selected interior curves was used to ensure that grid lines hit the zeros of the function. After computing the function

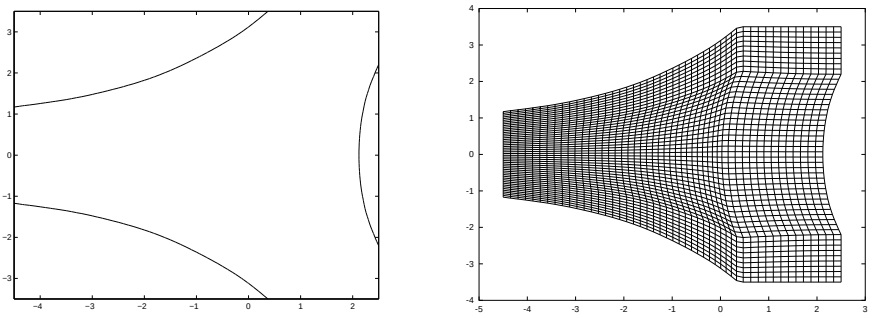

Figure 3: Contour curves where $Z = 5$ and Contour mesh.

over the contour mesh, the data was translated to VRML format to obtain the display in Figure 4. Both transfinite interpolation and the tensor

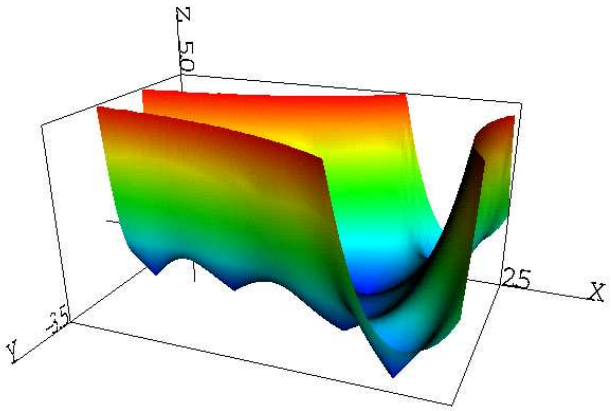

Figure 4: $|\mathrm{Bi}'(z)|$.

product algorithm were used to obtain the remaining grids shown. Very little difference could be detected in the grids developed by each method because the quality of the transfinite grid was such that little or no optimizing was needed to obtain a smoother grid. The main advantage of the tensor product algorithm would appear when transfinite interpolation produces a grid in which the lines overlap. The optimization code in the tensor product program could then be used to eliminate the overlap and produce less skewness and more orthogonality. Although the spacing was uneven in most of the grids because of the fixed points on the boundaries, the problem did not appear to be enough to effect the smoothness of the shading when the data was translated to VRML format. The requirement for orthogonality and smoothness is probably less stringent than it would be if the grids were being used to solve computational fluid dynamics problems. In any case, the tensor product code is capable of producing smoother grids if necessary.

The last figures show contour meshes and surfaces obtained for the gamma function defined on the complex plane and for a special type of Bessel function called the Hankel function. In both cases the mesh was formed by reflecting a grid defined for $y < 0$ along the $y = 0$ axis. An exponential function is used to concencentrate the grid points around the contour boundary.

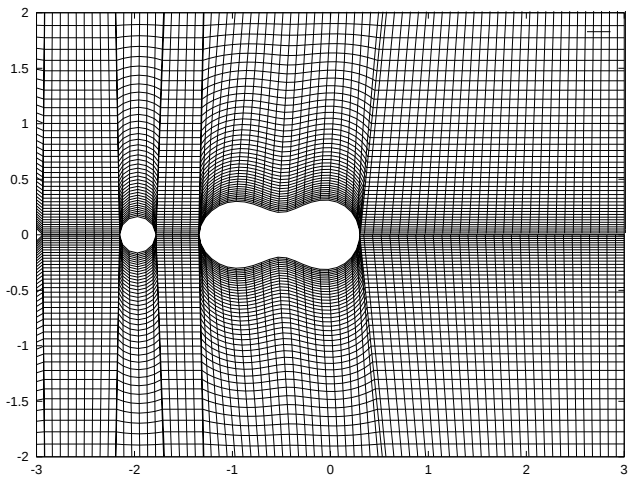

Figure 5: Contour mesh for Gamma Function.

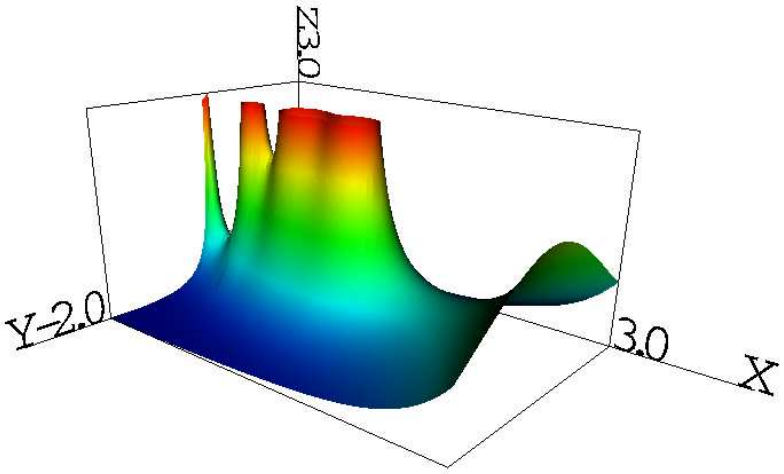

Figure 6: Gamma Function.

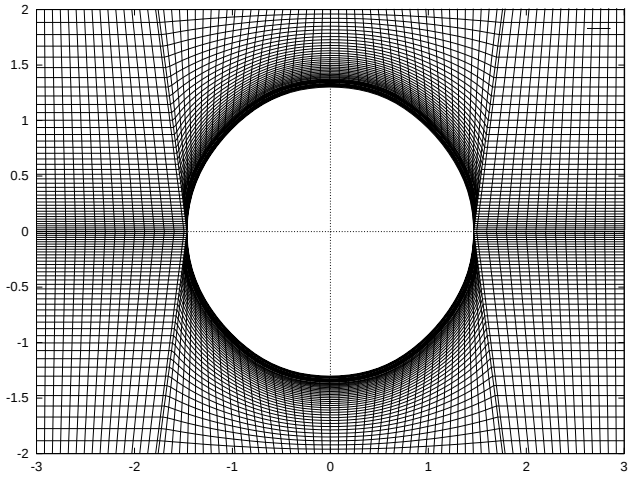

Figure 7: Contour mesh for Hankel function

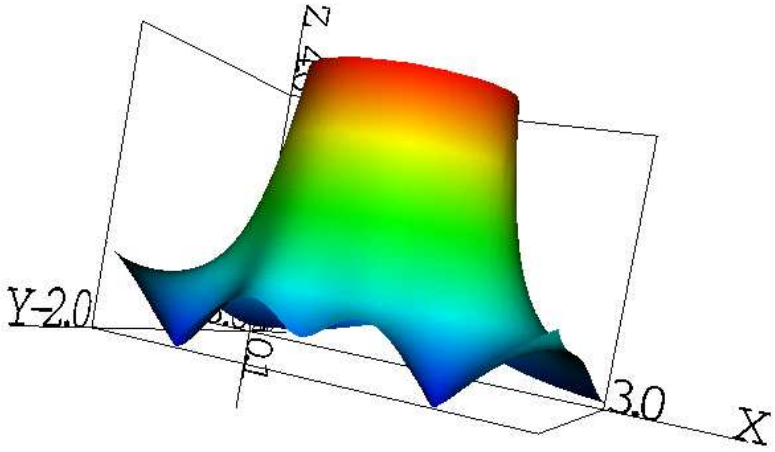

Figure 8: Hankel function $|\mathrm{H}_{3.5}^{(1)}(z)|$

Conclusions

The use of grid generation appears to be an effective tool in facilitating the plotting of 3D surfaces, but the complex nature of many special functions makes it difficult to design tools that work for all types of domains. The development of clear and informative visualizations for the NIST DLMF project will provide continued opportunities and motivation for exploring this problem. For the functions examined to date, somewhat simple structured grids have sufficed, but more advanced techniques will be needed for domains containing more numerous poles, zeros and other singularities. Some testing of unstructured grids has been done, but the authors have found that when the resulting data is translated to VRML, the surface shading is not as smooth. This problem might be diminished with the use of block structured grids.

After looking at off-the-shelf packages, it appears that commercial developers of 3D graphics packages might be interested in this work, especially as it relates to the use of contour meshes to efficiently clip a function.

Currently, we are exploring ways to feed the grid generation algorithm information about the special function so that it concentrates grid points in areas of high curvature. This will help the grid capture zeros of functions more accurately. Much work remains to be done on the visualization aspects

of the DLMF project, but the hope is that what has been learned so far will make the development of visualizations for more complicated functions somewhat easier.

References

[1] M. Abramowitz and I.A. Stegun, editors. **Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables, Vol. 55**, National Bureau of Standards Applied Mathematics Series. U.S. Government Printing Office, Washington, D.C., 1964.

[2] B. V. Saunders, "A Boundary Fitted Grid Generation System for Interface Tracking," **Numerical Grid Generation in Computational Field Simulations** (B.K. Soni et al., eds.), Mississippi State University, Mississippi, 1996.

[3] VRML. **The Virtual Reality Modeling Language**, International Standard ISO/IEC 14772-1:1997.

[4] W.J. Thompson, **Atlas for Computing Mathematical Functions**, John Wiley and Sons, Inc., New York, 1997, pp. 414-432.

[5] C. de Boor, **A Practical Guide to Splines**, Springer-Verlag, New York (1978).