Tensor Product Grid Generation for Complex Surface Visualizations in a Digital Library of Mathematical Functions

Bonita Saunders
Qiming Wang

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

Abstract

We have used tensor product B-spline grids to develop accurate plots that enhance the understanding of high level mathematical functions described in the National Institute of Standards and Technology Digital Library of Mathematical Functions. The graph data is placed in a web based format such as VRML (Virtual Reality Modeling Language) or X3D (Extensible 3D) to provide an environment where users can interactively manipulate complex function surfaces. The effectiveness of the grid generation mapping in producing visualizations that accurately capture key function features such as zeros, poles, and branch cuts is examined.

1. Introduction

The National Institute of Standards and Technology (NIST) is developing the NIST Digital Library of Mathematical Functions (DLMF) to replace the Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables, a popular resource in the mathematical and physical sciences, first published in 1964 [1]. The NIST DLMF will be available both in hardcopy and in a freely available web-based format. It will contain formulas, methods of computation, reference and software information for close to forty high level mathematical functions. Website users will be able to conduct an advanced mathematical equation search using the DLMF search engine, obtain instant information about mathematical terms, or interactively manipulate 3D visualizations of complex function surfaces.

This paper discusses our use of numerical grid generation techniques to facilitate the design of accurate visualizations for the NIST DLMF. By modifying an algebraic tensor product spline mesh generation algorithm that we originally designed for problems in aerodynamics and solidification

theory [2], [3], [4], we have been able to create boundary/contour fitted grids that capture significant function attributes such as poles, zeros, branch cuts and other singularities.

The DLMF visualization features are not designed to compete with the on-line computational and plotting capabilities of many commercial computer algebra packages, but the motivation for our work stems from some of the inadequacies of such packages.The complicated nature of many high level mathematical functions means that their computational domains are often irregular, discontinuous, or multiply connected. Although commercial packages may have some of these functions built-in, their 3D plots are generally, by default, over a rectangular Cartesian mesh, leading to poor and misleading graphs. Also, the packages often have trouble properly clipping a surface when values lie outside the range of interest to the user. Other packages may properly clip the function, but provide no reasonable way to export the clipped data for use outside the package. This is particularly important if the data needs to be transformed to a web-based format such as X3D (Extensible 3D) or VRML (Virtual Reality Modeling Language) [5], [6], [7].

We have discovered that many problems can be eliminated by designing a computational domain for the function whose boundary coincides with a particular contour of the surface. This not only produces an appropriate clipping of the surface, but also improves the smoothness of the color mapping. Either structured or unstructured techniques could be used to create the computational grids, but structured techniques make it easier to write efficient code to drive the interactive features of the visualizations. We will examine the problems that can arise when trying to display visualizations on the web, discuss the effectiveness of our grid generation technique, and look at some possibilities for improvements.

2. Constructing 3D Graphs for Interactive Visualizations in a Digital Library

The first release of the NIST DLMF will consist of thirty six chapters with content authored by experts in the field of special functions throughout the U.S. and abroad. The number, location and type of visualizations for each chapter have been determined by consulting with the authors and DLMF editors. To ensure data accuracy we are computing each function by at least two different methods, using commercial packages, publicly available software, or the author's personal codes. In many cases plot accuracy, that is, how well a displayed plot accurately represents the graph of a function, can be somewhat more difficult to achieve.

Most commercial packages do very well with 2D plots. They handle discontinuities automatically or with easy to use special options. They properly clip the function when the user asks for values within a specified range. This may not be the case with 3D plots. Figure 1 shows the plot of the incomplete gamma function, $\gamma^*(a, x)$, rendered using a popular commercial package and restricted to lie between -10 and 20. The shelf-like area might be confusing to students and others unfamiliar with the behavior of the function.

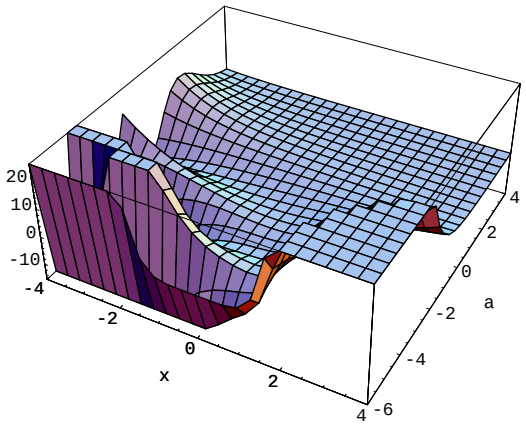

Figure 1. Plot of Incomplete gamma function $\gamma^*(a, x)$ illustrating bad clipping and poor resolution of poles.

Also, although experienced users can use special commands that will result in a properly clipped display, we found that this was still not sufficient for our requirements. Since the function is computed over a rectangular Cartesian mesh, the figure might look fine inside the package, but produce an irregular color map when the data is transformed to other formats such as VRML (Virtual Reality Modeling Language) or X3D (Extensible 3D), standard 3D file formats for interactive web-based visualizations [5], [6], [7]. The figure could also be improved by using a much larger number of data points, but large data files degrade the performance of our visualizations. We have found that the rendering problems can be eliminated or reduced in severity by computing the function over a boundary/contour fitted mesh. In the next section we discuss the grid generation algorithm we have used.

3. Grid Generation Mapping

The grid generation problem can be quite challenging, depending not only on the shape of the computational domain, but also on the behavior of

the function. For example, function domains can range from rectangles to complicated multipy connected domains with poles and branch cuts, but large function gradients can produce an irregular color map even when the domain is simple. Our grid generation technique is based on an algorithm we developed to solve partial differential equations(pdes) related to aerodynamics and solidification theory [2], [4], [8], [9]. We define a curvilinear coordinate system by a mapping $\mathbf{T}$ from the unit square I_2 to a physical domain of arbitrary shape. We let

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

where $0 \le \xi, \eta \le 1$ and each B_{ij} is the tensor product of cubic B-splines. Therefore, $B_{ij}(\xi,\eta) = B_i(\xi)B_j(\eta)$ 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. To obtain the initial α_{ij} and β_{ij} coefficients for $\mathbf{T}$, we first construct the transfinite blending function interpolant that matches the boundary of the complex function domain. The interior coefficients are obtained by evaluating the interpolant at knot averages as described in [10] to produce a variation diminishing spline approximation, a shape preserving approximation that reproduces straight lines and preserves convexity. We use the same technique for the boundary coefficients if the boundary side is a straight line, or use DeBoor's SPLINT routine [10] to find coefficients that produce a cubic spline interpolant of the boundary. For simple boundaries, the initial grid is often sufficient, but for more complicated or highly nonconvex boundaries we can improve the grid by using a variational technique to find coefficients that minimize the functional

$$F = \int_{I_2} \left(w_1 \left\{ \left(\frac{\partial J}{\partial \xi} \right)^2 + \left(\frac{\partial J}{\partial \eta} \right)^2 \right\} + w_2 \left\{ \frac{\partial \mathbf{T}}{\partial \xi} \cdot \frac{\partial \mathbf{T}}{\partial \eta} \right\}^2 + w_3 \{ u J^2 \} \right) dA \tag{2}$$

where $\mathbf{T}$ denotes the grid generation mapping, J is the Jacobian of the mapping, w_1, w_2 and w_3 are weight constants, and u represents external criteria for adapting the grid. The integral controls mesh smoothness, orthogonality, and depending on the definition of u, the adaptive concentration of the grid lines. When solving pdes, u might be the gradient of the evolving solution or an approximation of truncation error. Ideally, for our problem, we want u to be based on curvature and gradient information related to the function surface. We have made a slight modification of the integral we used in [9]. The adaptive term $w_3\{Ju\}$ has been replaced by $w_3\{uJ^2\}$. The new term produces a better concentration of grid lines, and also can be shown to be equivalent to the adaptive term used in the variational technique of Brackbill and Saltzman [11],[12].

To avoid solving the Euler equations for the variational problem, F is approximated in the computer code by the sum

$$\begin{aligned} G &= \sum_{i,j} w_1 \left[\left(\frac{J_{i+1,j} - J_{ij}}{\triangle \xi} \right)^2 + \left(\frac{J_{i,j+1} - J_{ij}}{\triangle \eta} \right)^2 \right] \triangle \xi \triangle \eta \\ &+ \sum_{i,j} w_2 Dot_{ij}^2 \triangle \xi \triangle \eta \\ &+ \sum_{i,j} w_3 u_{ij} J_{ij}^2 \triangle \xi \triangle \eta \end{aligned} \tag{3}$$

where J_{ij} is the Jacobian value, u_{ij} is the value of u, 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. When $w_3 = 0$, G is actually a fourth degree polynomial in each spline coefficient so the minimum can be found by using a cyclic coordinate descent technique which sequentially finds the minimum with respect to each coefficient. This technique allows the minimization routine to take advantage of the small support of B-splines when evaluating the sums that comprise G.

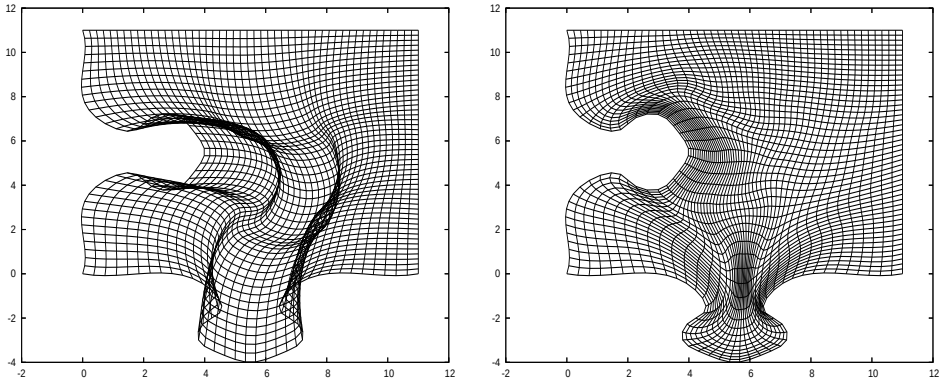

Figure 2. Initial and optimized puzzle grids.

In Figure 2 we applied the algorithm to a puzzle shaped domain. The initial grid, constructed using linear Lagrange polynomials for the transfinite blending functions, is shown on the left. Note the extreme amount of folding in the interior and the overlap of boundary lines. The grid on the right shows the mesh obtained after the spline coefficients are modified to minimize G with weights $w_1 = 2$, $w_2 = 5$, and $w_3 = 0$. The overlapping

grid lines have been pulled into the interior. When a grid line folds over a boundary or over another grid line in the interior, the Jacobian value changes sign. The Jacobian terms of G try to minimize the changes in the Jacobian between adjacent grid cells. Consequently, lines outside the boundary are pulled into the interior and interior grid folding is eliminated. In the original code[4], this effect was obtained by computing the interval for each coefficient that would guarantee that the Jacobian remained positive, but experimenting showed this to be a time consuming procedure that was not needed in most cases. Gonsor and Grandine use the same idea to obtain similar results in [13].

4. Results

We have used boundary/contour grid generation to facilitate the development of over two hundred visualizations for the NIST DLMF. To create a visualization, we first compute the contours for the upper and lower ranges of the function we want to graph. We then connect the contour curves with the parts of a rectangle to create a closed boundary for the grid generation algorithm. If the boundary is very complicated, the computational domain is divided into several sections. The function is then computed over the computational grid to produce data points that are placed into a VRML or X3D file format to produce an interactive display on the web. Figure 3 shows the computational domain grid and a snapshot of the DLMF display of Struve function $H_\nu(x)$. To create the grid we used the contour, or level, curves where $H_\nu(x) = 4$, the maximum height we want to display, and $H_\nu(x) = -3$, the minimum height. The dark horizontal bar on the grid is a concentration of grid lines designed to ensure the accurate representation of a pole.

In Figure 4 we revisit the incomplete gamma function $\gamma^*(a, x)$ that was shown in Figure 1. One of the features that will be available to users of the DLMF is the ability to scale down the surface in each coordinate direction. Scaling the surface representation of $\gamma^*(a, x)$ down to near zero in the vertical direction produces the density plot shown on the left in Figure 4. The density plot also shows the outline of the computational domain grid used to produce the surface on the right. Note the accurate clipping of the surface and the smooth color map.

All the visualizations in the NIST DLMF represent either real-valued or complex-valued functions of the form, $w = f(x, y)$. For complex-valued functions, the user has the option of using a height based color mapping where height $= |w|$, or a mapping based on the phase, or argument, of w. Figure 5 shows a plot of the modulus of the Hankel function $H_5^{(1)}(z)$ with

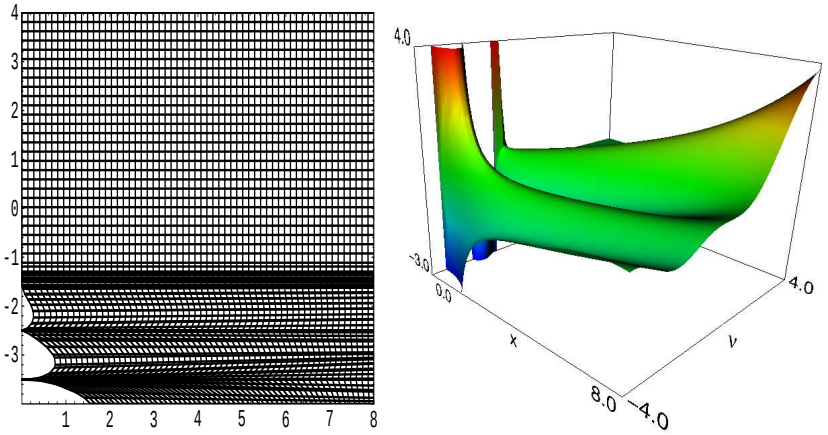

Figure 3. Grid and plot of Struve Function $H_\nu(x)$

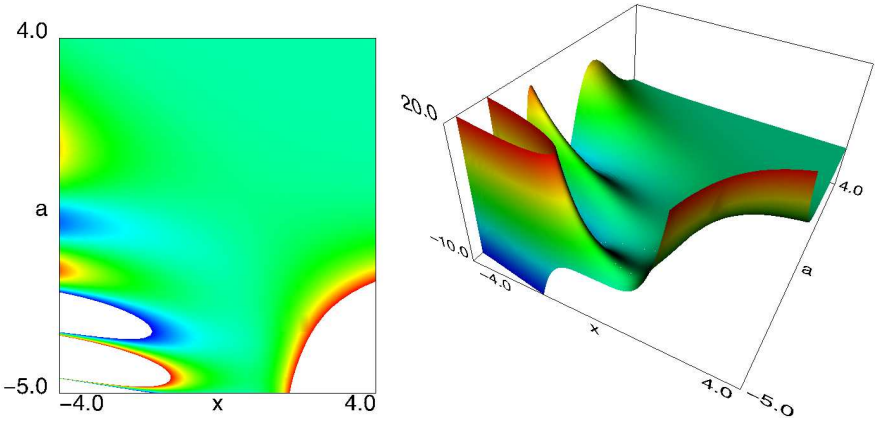

Figure 4. Density plot and plot of Incomplete gamma function $\gamma^*(a, x)$.

a phase based color map. On the left is a phase density plot obtained by scaling the figure down in the vertical direction. A branch cut, zeros and pole are visible in the figures.

In Figure 6 we attracted grid points in an equally spaced square grid to a circle and a sine curve, respectively. To obtain the attraction to the circle we defined the u in the adaptive term of our functional approximation G by

$$u(x, y) = e^{-20[(x-2)^2+(y-2)^2-2.25]^2} \tag{4}$$

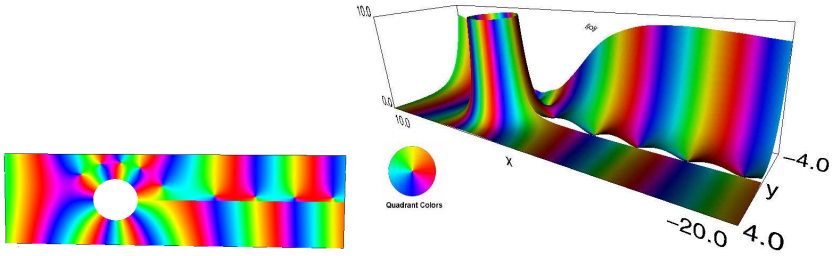

Figure 5. Plot of Hankel function $H_5^{(1)}$ with phase colormap.

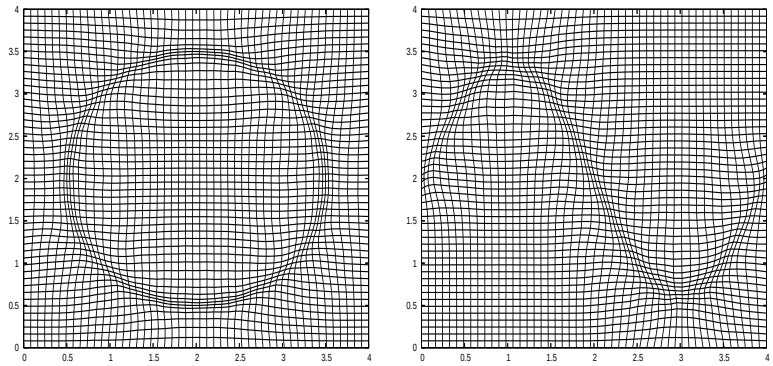

Figure 6. Grids adapted to circular and sinusoidal shapes.

and for attraction to the sine curve we defined

$$u(x,y) = e^{-30[1.4\sin(\pi x/2)+2-y]^2} \tag{5}$$

where $x = x(\xi,\eta)$ and $y = y(\xi,\eta)$ are the coordinates defined by the tensor product spline mapping in equation (1). For both grids we chose the smoothness weight, $w_1 = .2$, the orthogonality weight, $w_2 = 1.0$, and the adaptive weight, $w_3 = 10$. In earlier work [9] we defined u strictly in terms of the curvilinear coordinates ξ and η, and chose the parameters of u so that the curves would lie completely within the unit square. This definition guaranteed that the grid lines were attracted to a circle (or sine curve) in the unit square I_2 but not in the xy plane. Choosing u in this manner allowed us to quickly test the affect of u with few modifications to our code since the functional approximation G remained a fourth degree polynomial in each spline coefficient. To show the effect of u we simply mapped the unit square to a larger square.

However, defining u in terms of x and y means that a more general minimization code must be used to deal with the possible nonlinear dependence of u on the spline coefficients. This was necessary in order to get closer to our goal of adapting our grids to function curvature data. Preliminary progress was made by using Nash's truncated Newton algorithm [14], but we eventually settled on a more simple nonlinear solver, L-BFGS-B [15], which converges much slower, but is very robust.

We know that the minimization of G must also include the adjustment of the boundary coefficients, but this must be done carefully so that we obtain a reparameterization of the boundary curves without altering the shape. We can then concentrate on defining a u that captures the necessary function gradient information, but we realize that the specialized nature of some high level mathematical functions may make it challenging to access and link the codes needed to compute the data.

5. Conclusions

We have used boundary/contour fitted grid generation to complete over two hundred interactive 3D visualizations for the NIST Digital Library of Mathematical Functions. Our tensor product B-spline technique has helped us address many problems that appear in commercial packages including the inaccurate resolution of poles, bad clipping, and poor color mappings. Although the clarity of the surface color map is still an issue in areas where the gradient is large, we believe this can be improved by using an adaptive technique based on function gradient and curvature information. The adaptive method should also result in smaller data file sizes, which can improve the efficiency of some of our interactive features.

As primary developers of the graphs and visualizations for the NIST DLMF project we must tackle several issues simultaneously, including the production of the visualizations, data validation, the availability of VRML/X3D plugins, and accessibility on major platforms, but we have made steady progress toward the development of the adaptive code. We have successfully adapted our grids to curves using adaptive criteria having a nonlinear dependence on the spline coefficients.

We are currently incorporating the adjustment of the boundary coefficients into the minimization functional. Once that work is completed, we will focus on describing suitable adaption criteria to capture gradient and curvature information, and integrating the computation of this function information with the grid generation code.

.

Disclaimer

All references to commercial products are provided only for clarification of the results presented. Their identification does not imply recommendation or endorsement by NIST.

References

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

[2] B. V. Saunders. A boundary conforming grid generation system for interface tracking. *J. Computers Math Applic.*, 29:1–17, 1995.

[3] B. V. Saunders. The application of numerical grid generation to problems in computational fluid dynamics. *Council for African American Researchers in the Mathematical Sciences: Volume III, Contemporary Mathematical Series, American Mathematical Society*, 275:95–106, 2001.

[4] B. V. Saunders. Algebraic grid generation using tensor product b-splines. *NASA CR-177968*, 1985.

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

[6] R. Carey and G. Bell. *The Annotated VRML 2.0 Reference Manual*. Addison-Wesley, Boston, 1997.

[7] D. Brutzman and L. Daly. *Extensible 3D Graphics for WEB Authors*. Morgan Kaufmann (Elsevier), San Francisco, CA, 2007.

[8] B. V. Saunders and Q. Wang. From 2d to 3d: numerical grid generation and the visualization of complex surfaces. In B. K. Soni, J. F. Thompson, J. Haeuser, and P. R. Eiseman, editors, *Proceedings of the 7th International Conference on Numerical Grid Generation in Computational Field Simulations*, Whistler, British Columbia, Canada, September 2000.

[9] B. V. Saunders and Q. Wang. From b-spline mesh generation to effective visualizations for the nist digital library of mathematical functions. In P. Chenin, T. Lyche, and L. Schumaker, editors, *Curve and Surface Design: Avignon 2006*, Nashboro Press, Brentwood, 2007.

[10] C. de Boor. *A Practical Guide to Splines, Revised Edition*. Springer-Verlag, New York, 2001.

[11] J. U. Brackbill and J. S. Saltzman. Adaptive zoning for singular problems in two dimensions. *J. Comput. Phys.*, 46:342–368, 1982.

[12] J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin. *Numerical Grid Generation: Foundations and Applications*. North Holland, New York, 1985.

[13] D. Gonsor and T. Grandine. A curve blending algorithm suitable for grid generation. In M. L. Lucian and M. Neamtu, editors, *Geometric Modeling and Computing: Seattle 2003*, Nashboro Press, Brentwood, 2004.

[14] S. G. Nash. Newton-type minimization via the lanczos method. *SIAM Journal on Numerical Analysis*, 21(4):770–788, 1984.

[15] R. H. Byrd, P. Lu, and J. Nocedal. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific and Statistical Computing*, 16(5):1190–1208, 1995.