

Complex Variables, Mesh Generation, and 3D Web Graphics: Research and Technology Behind the Visualizations in the NIST Digital Library of Mathematical Functions

Bonita V. Saunders

ABSTRACT. In 2010, the National Institute of Standards and Technology (NIST) launched the Digital Library of Mathematical Functions (DLMF), a free online resource containing definitions, recurrence relations, differential equations, and other crucial information about mathematical functions useful to researchers working in application areas in the mathematical and physical sciences. Although the DLMF was designed to replace the widely cited National Bureau of Standards(NBS) Handbook of Mathematical Functions commonly known as Abramowitz and Stegun (A&S), the goal was a compendium far beyond a book on the web, incorporating web tools and technologies for accessing, rendering, and searching math and graphics content. This paper focuses primarily on the research and technical challenges involved in creating the DLMF's graphics content, and in particular, its interactive 3D visualizations, where users can explore more than 200 graphs of high level mathematical function surfaces.

1. Introduction

In 2010, after a multi-year effort dating back to the late 1990s, the National Institute of Standards and Technology (NIST) released the Digital Library of Mathematical Functions (DLMF)[9], a free online resource containing definitions, recurrence relations, differential equations, and other crucial information to aid in the understanding and computation of mathematical functions that arise in application areas in the mathematical and physical sciences. Although the DLMF might be viewed as an update and replacement for the 1964 Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables (A&S) [1], in reality, it is quite different in both focus and content.

A&S was originally known for its tables. Its existence can be traced back to the Mathematical Tables Project created in 1938 by NIST's predecessor, the National Bureau of Standards (NBS), to address a crucial need for accurate tables to assist in the computation of functions commonly occurring in practical problems [5]. The project was administered by the Works Projects Administration, a New Deal

2010 *Mathematics Subject Classification.* Primary 65D17.

agency of President Franklin Roosevelt. Highly educated, but out of work mathematicians and physicists supervised a staff of human ‘computers’ who performed calculations for reference tables of function values. From 1938 to 1946, 37 volumes of tables were published, including tables of trigonometric functions, logarithms, the exponential function, and probability functions [5]. Realizing the importance of having the information all in one place, NBS mathematician Milton Abramowitz, a technical leader of the Mathematical Tables Project, eventually pushed for the publication of a compendium of tables and related material. This compendium, with emphasis on higher level functions such as Bessel functions, hypergeometric functions, and elliptic functions was published as the Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables in 1964. It is often simply called Abramowitz & Stegun or A&S in honor of its editors Milton Abramowitz and Irene Stegun who were Chief and Assistant Chief, respectively, of the Computation Laboratory of the NBS Applied Mathematics Division around the start of the project in 1956. Stegun took over the project and shepherded it to completion when Abramowitz died suddenly of a heart attack in 1958[5].

A casual glance at A&S clearly shows that tables predominate the handbook, but each function also includes related material such as formulas representing differential equations, definite and indefinite integrals, inequalities, recurrence relations, power series, asymptotic expansions, polynomial and rational approximations, graphs, and other qualitative information that might be useful for understanding and computing values of the function. For practitioners this “related material” has moved to the forefront over the years, while the tables have receded in importance due to the prevalence of reliable numerical software and computer algebra packages that have severely decreased the need for tables for computing function values by interpolation. Acknowledging this, it was decided that tables would not be included in the design for the DLMF. Nevertheless, the addition of new chapters on functions of growing significance and information on new properties of existing A&S functions increased the DLMF’s content significantly over that of A&S.

The fact that the DLMF is web-based has opened up many possibilities that are still being explored. Navigational tools and hyperlinks allow the user to move around the site in a variety of ways. A database of metadata provides information for pop-up boxes where users can find links to sources, defined variables, cross references, alternative text formats such as LaTeX, MathML, and image formats, or notes on changes made to content. With the DLMF’s math-aware search engine users can search by function names, particular formulas, and in some cases types of functions, for example, ‘trig’. The site includes 600 2D and 3D plots along with 200 dynamic interactive visualizations where users can explore the graphs of elementary and high level mathematical functions. An overview of the technological capabilities in the DLMF was recently published in a Physics Today article[20]. In this paper we take a more in-depth view of the DLMF’s graphics by looking at the ongoing research and development behind its interactive 3D visualizations.

Section 2 describes the techniques used to construct the computational grids for plotting the graphs shown in the visualizations. In Section 3 we look at our utilization and advancement of techniques for displaying interactive 3D graphics on the web and discuss some of the interesting capabilities available in the DLMF visualizations. Section 4 discusses ongoing challenges and future areas for research.

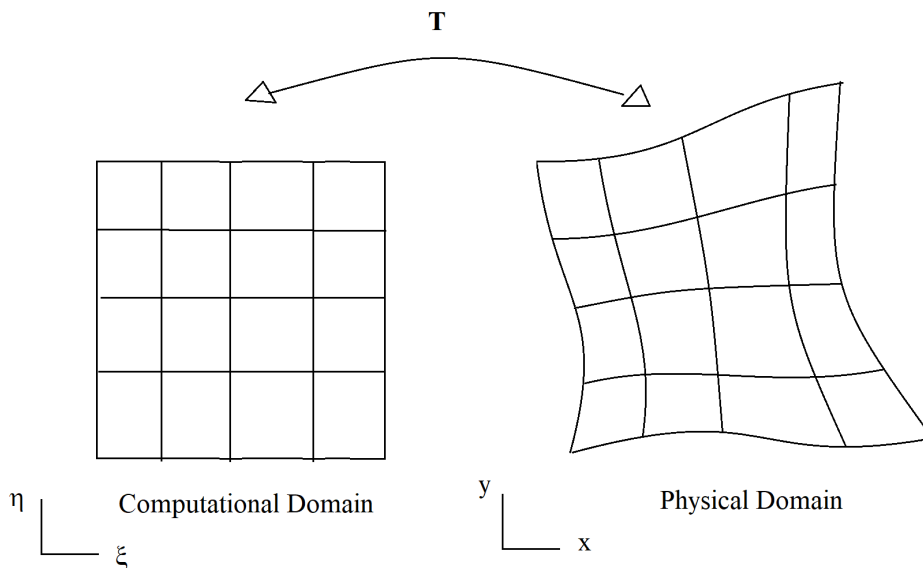


FIGURE 1. Numerical grid generation is defined by a curvilinear coordinate map from a canonical domain to the oddly shaped physical domain prescribed by the application.

2. Grid Generation

During the design phase of the DLMF in the late 1990s and early 2000s little thought was given to how one might plot and render complex function graphs on the web. The focus was on the primary mathematical content that chapter authors would be asked to write. However, once a draft of the first chapter, on Airy and related functions, was written, we began looking at the best way to create the illustrative graphs the chapter needed. On looking at computer algebra systems we found that most did an excellent job plotting 2D graphs, which could be exported to an acceptable format for viewing on the web. For 3D graphs of function surfaces, the story was quite different. Most systems had rudimentary or non-existent machinery for properly clipping the graph when it was necessary to restrict the displayed range to illuminate significant features, such as poles or zeros. In one case the surface was properly clipped when viewed inside the system, but the unclipped data reappeared when the plot data was exported to a file. The problems we observed led us to design our own software for grid generation and, as we show in the next section, inspired our development of visualizations utilizing emerging web 3D technology.

We solved the clipping problem by computing the function over a 2D grid whose boundary included a level curve, or contour, of the function. We created the grid by using numerical grid generation, which defines a curvilinear coordinate system through a map from a canonical domain, such as a square in 2D, to the physical domain of interest, as shown in Figure 1.

Numerical grid generation is just one of several methods for creating a grid, or mesh, for solving problems over an oddly shaped domain. It has often been used with finite difference methods to solve partial differential equations (PDEs)

governing flow around interesting geometries such as an airplane wing, ship's hull, or automobile body. Numerical grid generation is sometimes called structured grid generation because of the natural array order of the grid points on the physical domain [23]. Unstructured methods such as Delaunay triangulations, quadtree methods, or hybrid methods that combine both structured and unstructured meshes are often the methods of choice for extremely complex geometries. However, they require the storage of grid node connectivity information that can sometimes cause memory issues.

Structured methods can require a bit of ingenuity if the boundary shape is complex, but they may allow one to write more efficient code for some applications. In our case the structured order facilitated the coding of the interactive features for our visualizations. Our code is based on an algorithm we initially designed for problems in aerodynamics and solidification theory [13, 14, 16]. We modified the code to accurately approximate function boundary and contour data as well as capture significant function attributes such as poles, zeros, branch cuts, and other singularities.

When numerical grid generation is used for solving PDEs, the coordinate mapping must be one-to-one and onto to ensure invertibility. The goal is to transform the equations from the physical domain to equations over a simpler canonical domain where the difference equations and boundary conditions are easier to apply. Although, in our case, the 2D grid over the physical domain becomes the computational grid for the function being plotted, the same one-to-one correspondence is still needed since it ultimately affects the accuracy of the surface clipping and the smoothness of the colormap when the surface is rendered on the web [15, 17]

Our basic algorithm constructs a curvilinear coordinate spline mapping \mathbf{T} from the unit square I_2 to the physical domain and is defined by

$$(2.1) \quad \mathbf{T}(\xi, \eta) = \begin{pmatrix} x(\xi, \eta) \\ y(\xi, \eta) \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^m \sum_{j=1}^n \alpha_{ij} B_{ij}(\xi, \eta) \\ \sum_{i=1}^m \sum_{j=1}^n \beta_{ij} B_{ij}(\xi, \eta) \end{pmatrix}$$

where 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, $\{p_i\}_1^{m+4}$ and $\{q_j\}_1^{n+4}$, respectively [13].

To quickly obtain initial α_{ij} and β_{ij} coefficients for \mathbf{T} we construct a transfinite blending function mapping [11, 23, 10] that interpolates the boundary of the physical domain. Conveniently, the spline coefficients can be divided into boundary coefficients that map the boundary of the square onto the boundary of the physical domain, and interior coefficients [13, 14], which hopefully map the interior of the square onto the interior of the physical domain. Their initial values are obtained by evaluating the transfinite interpolant at knot averages as described in [4], to produce a shape preserving approximation that reproduces straight lines and preserves convexity. If more accuracy is needed on part of the boundary, we use de Boor's SPLINT routine [4] to find coefficients that produce a cubic spline interpolant of that side. It is important that the spline knots and boundary coefficients be chosen carefully to produce an accurate representation of the physical boundary.

For simple boundaries, the initial coefficients produce a grid that is adequate for most applications, but if the boundary is more complicated or highly nonconvex, modifications of the coefficients are often necessary. To improve the grid, we fix

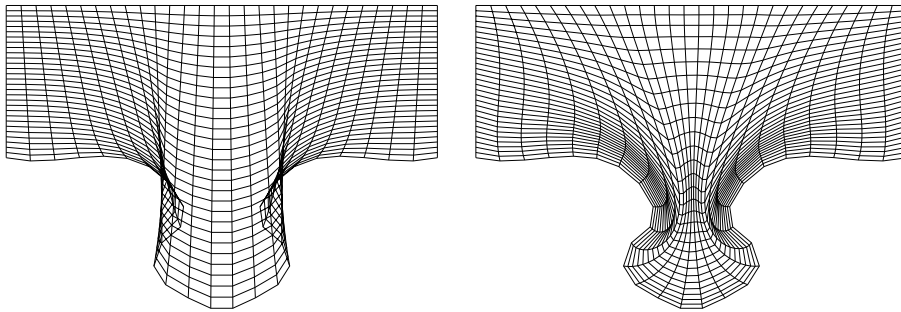


FIGURE 2. Initial and optimized puzzle grids.

the boundary coefficients and use a variational method to find interior coefficients that minimize the functional

$$(2.2) \quad F = \int_{I_2} \left(w_1 \left\{ \left(\frac{\partial J}{\partial \xi} \right)^2 + \left(\frac{\partial J}{\partial \eta} \right)^2 \right\} + w_2 \left\{ \frac{\partial \mathbf{T}}{\partial \xi} \cdot \frac{\partial \mathbf{T}}{\partial \eta} \right\}^2 \right) dA$$

where \mathbf{T} denotes the grid generation mapping, J is the Jacobian of the mapping, and w_1 and w_2 are weight constants. This integral controls mesh smoothness and orthogonality. A large value for w_1 will decrease the variance in Jacobian values at nearby points, making the grid smoother. The w_2 weighted term represents the dot product of the tangent vectors $\partial \mathbf{T} / \partial \xi$ and $\partial \mathbf{T} / \partial \eta$. Therefore, minimizing this term enhances grid orthogonality. A change of variables shows this term to be equivalent to the volume weighted version of the orthogonality term in the Brackbill and Saltzman functional [6]. Figure 2 shows the initial and optimized grids for a physical domain shaped like a puzzle piece.

Figure 3 shows the computational grid and Riemann zeta function surface plot created using it. The grid boundaries, including the exterior boundary and the interior one around the pole, contain contour data for a height of 3. Computing the function over the grid produces a smooth clipping of the surface. A number of the “non-trivial” zeros of the Riemann zeta function can be viewed by exploring the figure on the DLMF site [9]. In our original code we input the location of the zeros to guarantee that there are gridpoints there. We are currently working on an algorithm that will automatically move gridpoints to the vicinity of a zero.

The current algorithm contains two significant changes over the original. First, we have added an adaptive term $w_3 \{uJ^2\}$ to the integrand of the functional, where w_3 is a weight constant, and u represents external criteria for adapting the grid. If we were solving a system of partial differential equations, u might represent the gradient of the evolving solution or an approximation of truncation error. For our purposes, we want u to contain curvature and gradient information related to the function surface. The goal is to create a grid generation system that adaptively moves gridpoints to areas of high curvature or large gradient. With a change of variables this term is equivalent to the weighted volume variation, or adaptive, component of the Brackbill and Saltzman functional [6, 23]. Therefore, the enhanced integral should allow some control over mesh smoothness, orthogonality, and through u , permit an adaptive concentration of grid lines.

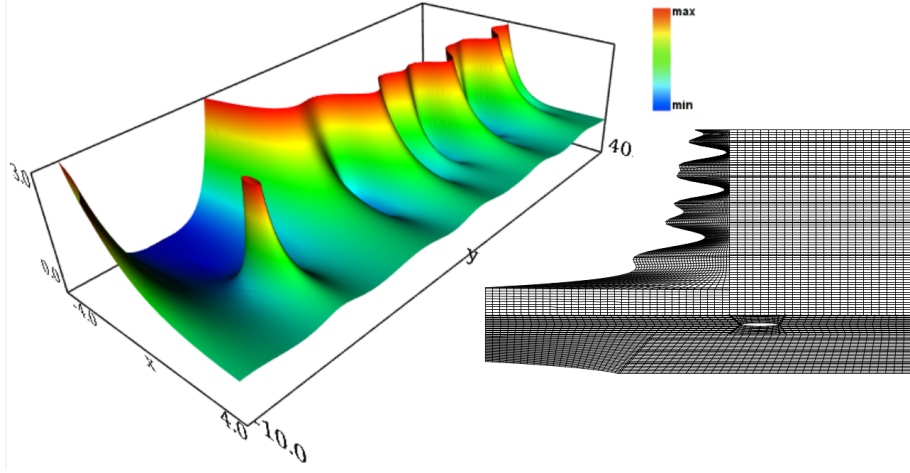


FIGURE 3. Riemann zeta function surface obtained by computing function over grid shown.

Second, a more fundamental change in our algorithm is replacing the mapping \mathbf{T} by a composite mapping $\mathbf{T}^* = \mathbf{T} \circ \Phi$ where Φ is a tensor product spline mapping from the unit square I_2 to I_2 with its own coefficients and knot sequences [18]. Adaptive methods typically construct a reference grid [22] or distribution mesh [21, 13] by moving points on the canonical domain based on some adaptive criteria. The reference/distribution mesh is then mapped to the physical domain to create an adaptive mesh there. Mathematically this could be viewed as the composition of two maps where one maps the canonical domain to itself and the other maps the canonical domain to the physical domain. DeRose, et al., created composite maps in B-spline or Bézier form [7, 8].

For now, we have not tried to create a simple representation for our composite map, that is, we leave Φ and \mathbf{T} in their separate forms. The boundary coefficients of \mathbf{T} can remain fixed while the coefficients of Φ are adjusted to reparameterize the boundary points. The Φ map can be used to create a reference grid that produces the desired adaptive effect without disturbing the accuracy of the boundary approximation.

After choosing initial \mathbf{T} and Φ coefficients that approximate transfinite interpolation we can improve our final physical grid, that is, the smoothness, orthogonality, or concentration of grid points by minimizing the following functional with respect to either the Φ coefficients or interior \mathbf{T} coefficients:

$$(2.3) \quad F^* = \int_{I_2} \left(w_1 \left\{ \left(\frac{\partial J^*}{\partial \xi} \right)^2 + \left(\frac{\partial J^*}{\partial \eta} \right)^2 \right\} + w_2 \left\{ \frac{\partial \mathbf{T}^*}{\partial \xi} \cdot \frac{\partial \mathbf{T}^*}{\partial \eta} \right\}^2 + w_3 \{ u J^{*2} \} \right) dA$$

where $*$ has been added to indicate the terms are associated with the composite mapping \mathbf{T}^* . Then J^* , the Jacobian of \mathbf{T}^* is the product of J and J_Φ where J is the Jacobian of \mathbf{T} and J_Φ , the Jacobian of Φ . To simplify our notation, $*$ is not added to the weight constants or u . Figure 4 shows a puzzle shaped grid adapted to the vertical line $x = 5.5$. We first optimize with respect to the interior \mathbf{T} coefficients

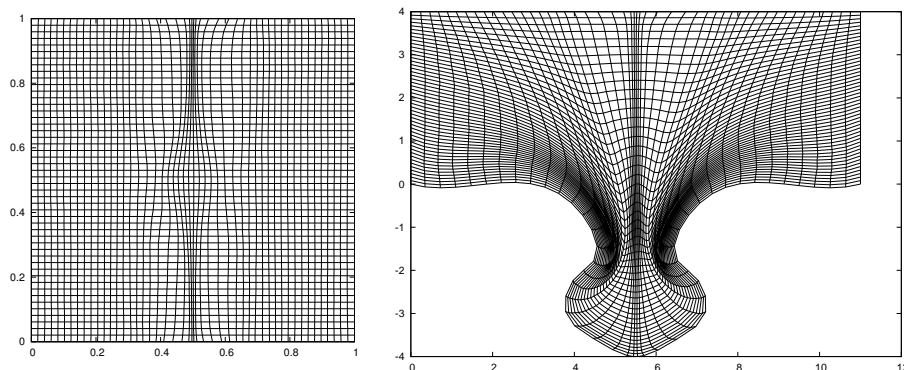


FIGURE 4. Reference grid and adapted puzzle grid.

to obtain an acceptable physical grid as shown in Figure 2. We then optimize with respect to the Φ coefficients to adapt the grid to the line. Our adaptive function u is defined by

$$(2.4) \quad u(x, y) = e^{-50(x-5.5)^2}.$$

We have defined other expressions for u to adapt grids to circular arcs, circles, and intersecting lines [18]. We are now focusing on improving the performance of the code and experimenting with various definitions of u to capture function curvature and gradient data. Also, since our function data is computed using a variety of codes and computer algebra packages, we are also working on the integration of our grid generation code with various software packages and systems.

3. 3D Web Graphics and DLMF Implementation

The individual chapters of both A&S and the DLMF were authored by various mathematicians and physicists of note. One A&S author was Philip J. Davis, who was at NBS at that time. Davis prepared the chapter on gamma and related functions, which he designed to serve as a model for the other authors. Davis hired Frank W.J. Olver who authored the A&S chapter on Bessel functions. Years later, Olver would serve as DLMF Mathematics Editor, Editor and Chief, and author several chapters in the DLMF.

More than 35 years after the publication of A&S, Davis, then a professor at Brown, was invited back to hear about NIST's plans for the development of the DLMF. Davis' tepid response to our preliminary colorful, but static 3D graphs for the first DLMF chapter, Airy and related functions, actually sparked our research and design of 3D function surface visualizations that grew in sophistication as technology for displaying 3D graphics on the web advanced.

3.1. 3D Web Graphics. As mentioned earlier, we found that at the start of the DLMF project in the late 1990s and 2000s, the export of 3D graphics data by well-known computer algebra systems was inadequate for our needs. After looking into the graphics technology being used and studied at NIST, we concluded that VRML (Virtual Reality Modeling Language) was our best option. VRML is a 3D file format for creating interactive graphics for viewing on the web. There were

a variety of VRML viewer browsers that could be freely downloaded, but over time the maintenance of some of the best was discontinued and the quality of new browsers was mixed. Furthermore, as we began to better understand what features we wanted to see in DLMF visualizations, the complexity of our visualizations increased, making it more and more difficult to find VRML browsers that could handle our graphics files, even when our codes appeared to follow VRML standards.

Noting the industry transition from VRML to X3D (Extensible 3D), graphics team member Qiming Wang designed a VRML to X3D converter. By the launch of the DLMF in 2010 we had created close to 200 interactive visualizations of mathematical function surfaces accessible in both formats [12].

However, our ultimate goal was to make the DLMF visualizations accessible on Windows, Mac, and Linux platforms. We found VRML/X3D browsers that worked for Windows and Mac, but never found a Linux browser that could successfully handle our graphics files. Also, having to download a viewer browser/plugin to see the visualizations was a headache for both maintainers and users of the DLMF site. Problems arose whenever the browsers needed to be updated, or whenever there were changes to the platform operating system.

Motivated by these concerns, in mid 2011 we began monitoring the development of WebGL, a JavaScript API (application programming interface) for rendering graphics in a web browser without a viewer plug-in. Then, thanks to the work of Johannes Behr and colleagues [2] on X3DOM, which permitted the direct integration of X3D nodes into HTML content, we were able to make a crucial decision. We would convert all the DLMF visualizations to WebGL by using the X3DOM framework to build the application around our X3D codes. We were encouraged by our early success in creating a few initial visualizations that worked in a beta WebGL accessible Mozilla Firefox browser. We were also bolstered by preliminary X3DOM/WebGL work by NIST researcher Sandy Ressler and the work of Steven Birr, et al., on the LiverAnatomyExplorer WebGL Tool [3]. We began an intensive effort to convert all the DLMF 3D visualizations to WebGL and seamlessly integrate the displays into the HTML pages of the associated chapters. The new visualizations first appeared in DLMF Version 1.0.7 released on March 21, 2014.

Building our application using the X3DOM framework allowed us to reuse most of our X3D code to create the WebGL files. The most challenging work was recoding the dynamic displays and interactive features. After first creating stand alone WebGL files, we worked with NIST computer scientist Brian Antonishek and Bruce Miller, information architect of the DLMF website, to make the coding changes needed to integrate the visualizations into DLMF HTML files. We also made style changes to achieve a more polished look. Most importantly we successfully achieved our main goal: To reproduce or enhance the capabilities available in our VRML/X3D visualizations and provide additional capabilities where possible. WebGL is now the default format for viewing the DLMF visualizations and VRML/X3D files are being phased out [19].

3.2. DLMF Graphics Features. The best way to experience the DLMF visualizations is to go directly to the graphics sections found in most chapters and explore the capabilities available. The visualizations can now be viewed in most common web browsers on Windows, Mac, or Linux platforms. A few features are highlighted in this section.

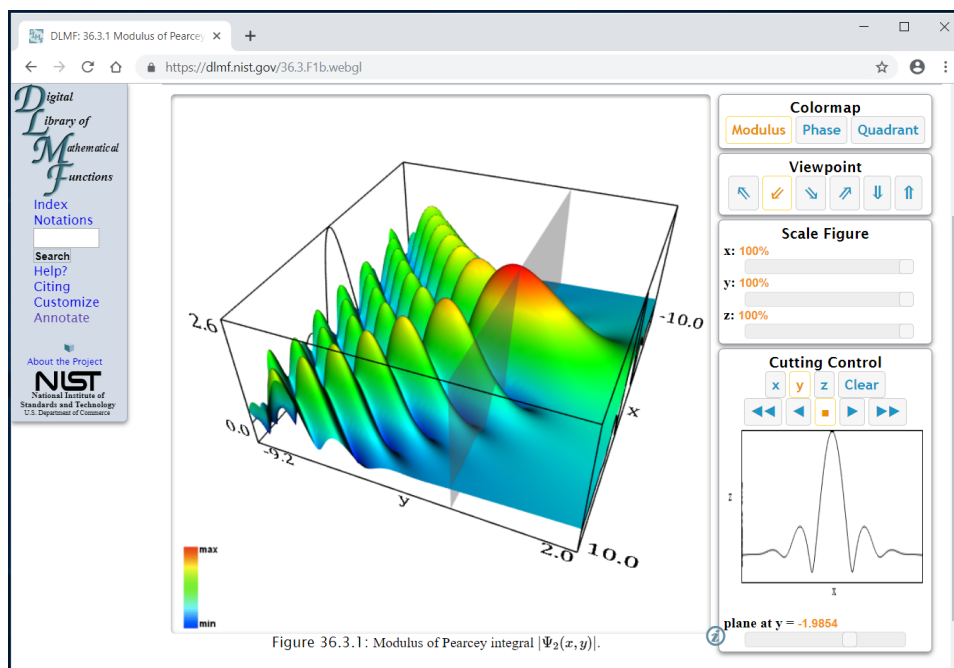


FIGURE 5. Modulus of Pearcey integral visualization embedded in DLMF webpage. Intersection of y direction cutting plane with surface displayed on bounding box sides and in pop-up display on side panel.

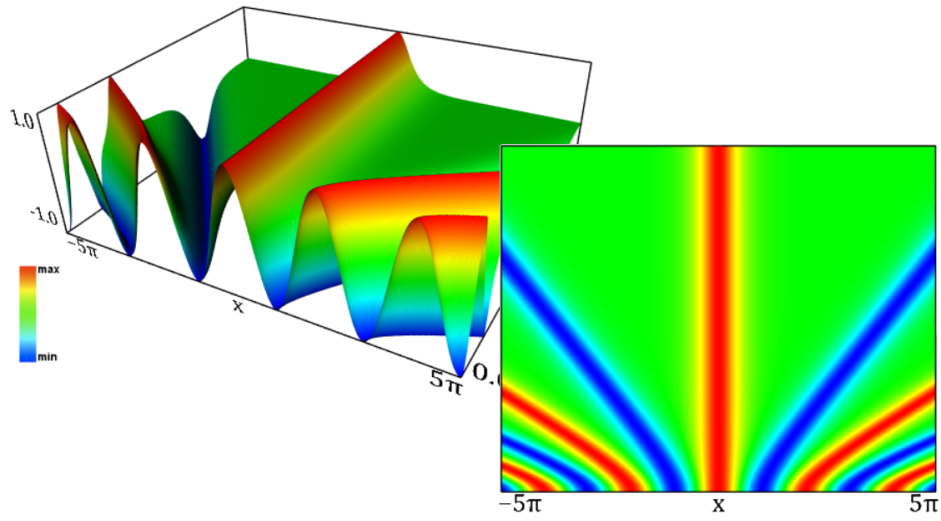
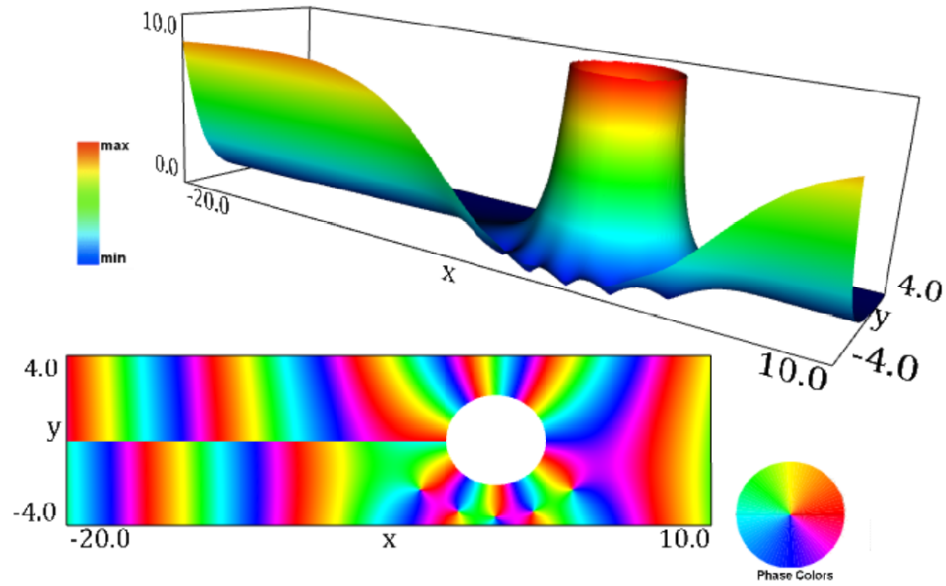
Figure 5 shows the general display for our surface visualizations. The user may click on the figure and rotate it freely or choose a stored viewpoint from the selection offered on the panel to the right. If a surface represents a complex valued function, the user is offered a phase, or argument, based color map in addition to the height, or modulus, color map. This option will not appear if the function is real valued.

Figure 6 shows a density plot for a Jacobian Elliptic function adjacent to its surface plot. A user can apply the scaling option to collapse the function in the vertical direction to obtain the density plot.

At the top of Figure 7 a type of Bessel function known as a Hankel function is shown with a height-based color map. Its branch cut is evident when one switches to a phase color map, and on scaling the surface height to zero, the phase density plot shown at the bottom emerges. On the website, one should note the difference in how the color spectrum is traversed as one travels around a pole versus a zero.

4. Current and Future Areas for Research

Clearly, a significant amount of work is involved in the design, creation, and maintenance of the visualizations in the DLMF. Initially, much of the 3D graphics work was motivated by deficiencies we saw in available software and computer algebra systems at the time. The rendering of 3D plots has improved in many

FIGURE 6. Jacobian Elliptic function $\text{cn}(x, k)$ and density plot.FIGURE 7. Modulus of Hankel function $H_{5.5}^{(1)}(x + iy)$ and phase density plot.

systems, and export options have expanded tremendously, but we still notice that the quality of the 3D data exported may not match what is seen on the screen.

Creating our own grids and visualizations gives us access to the data and routines that control our visualizations. This is helpful if we want to expand existing

features or create new ones. Also, since our work is open to the public, we can get feedback from other researchers through publications and presentations at conferences.

There are several directions to go with our grid generation work. The ultimate goal is to develop a robust method that can be used to create quality grids in a reasonable amount of time. For now we will continue the work on adaptive curvature/gradient grids. We may also explore a parametric grid generation mapping which might work better if there are poles or other areas where there are steep gradients. Also, there have been some initial discussions with other grid generation researchers on the feasibility of creating a true zoom where the grid is refined and function values recomputed. Such an implementation would require a fast grid generation algorithm and hierarchical or locally refined techniques. Also, exploring unstructured triangulations and hybrid methods are still a possibility.

In addition to the true zoom, we might consider other changes to the visualizations such as adding or improving color maps, or including plots of real and imaginary parts of functions along with the modulus. In any case, while we expect to stick with our X3DOM/WebGL platform for the near future, we will strive to stay informed about trends in 3D web technology that might enhance our visualizations.

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.A. Abramowitz, I.A. Stegun, *Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables*, NBS Applied Mathematics Series 55, National Bureau of Standards, Washington, DC, 1964.
2. J. Behr, P. Eschler, and M. Zollner, X3DOM: A DOM-based HTML5/X3D Integration Model, in *Proceedings of the 14th International Conference on 3D Web Technology(Web3D '09)*, ACM, S.N. Spencer, ed., pp. 127-135, 2009.
3. S. Birr, J. Monch, D. Sommerfeld, U. Preim, and B. Preim, The Liver Anatomy Explorer: A WebGL-based Surgical Teacing Tool, *IEEE Computer Graphics and Applications*, **33**, 5, pp.48-58, 2013.
4. C. de Boor, *A Practical Guide to Splines (revised edition)*, Springer, New York, 2001.
5. R.F. Boisvert, D.W. Lozier, Handbook of Mathematical Functions, in *A Century of Excellence in Measurements Standards and Technology*(D.R. Lide, ed.), CRC Press, pp. 135-139, 2001.
6. J. U. Brackbill, J. S. Saltzman, Adaptive zoning for singular problems in two dimensions, *J.Comput.Phys.* **46** (1982), 342-368.
7. T. D. DeRose, Composing Bezier simplexes, *ACM Transactions on Graphics* (3) **7** (1988), 198-221.
8. T. D. DeRose, R. N. Goldman, H. Hagen & S. Mann, Functional composition algorithms via blossoming, *ACM Transactions on Graphics* (1993), 113-135.
9. *NIST Digital Library of Mathematical Functions*. <https://dlmf.nist.gov/>, Release 1.022 of 2019-03-15. F.W.J. Olver, A.B. Olde Daalhuis, D.W. Lozier, B.I. Schneider, R.F. Boisvert, C.W. Clark, B.R. Miller, and B.V. Saunders, eds.
10. D. Gonsor, T. Grandine, A curve blending algorithm suitable for grid generation, in *Geometric Modeling and Computing: Seattle 2003*, Nashboro Press, Brentwood, 2004.
11. W.J. Gordon, C.A. Hall, Construction of Curvilinear Coordinate Systems and Applications to Mesh Generation, *International Journal for Numerical Methods in Engineering*, **7**, pp.461-477, 1973.

12. F.W.J. Olver, D.W. Lozier, R.F. Boisvert, C.W. Clark, A Special Functions Handbook for the Digital Age, *Notices Amer. Math Soc.* 58, 7, pp. 905-911, 2011.
13. B. V. Saunders, Algebraic grid generation using tensor product B-splines, *NASA CR-177968*, 1985.
14. B. V. Saunders, A boundary conforming grid generation system for interface tracking, *Computers Math. Applic.* **29** (1995), 1-17.
15. B. V. Saunders, Q. Wang, From 2d to 3d: numerical grid generation and the visualization of complex surfaces, in *Proceedings of the 7th International Conference on Numerical Grid Generation in Computational Field Simulations*, 51-60, Whistler, British Columbia, Canada, 2000.
16. B. V. Saunders, The application of numerical grid generation to problems in computational fluid dynamics, in *Council for African American Researchers in the Mathematical Sciences: Vol: III*, 95-106, Contemporary Mathematics Series **275**, American Mathematical Society, 2001.
17. B. V. Saunders, Q. Wang, From B-spline mesh generation to effective visualizations for the NIST digital library of mathematical functions, in *Curve and Surface Design: Avignon 2006*, 235-243, Nashboro Press, Brentwood, 2007.
18. B.V. Saunders, Q. Wang, B. Antonishek, Adaptive Composite B-Spline Grid Generation for Interactive 3D Visualizations, in *Proceedings of MASCO T12/ISGG2012(International IMACS Workshop and Bi-annual International Society for Grid Generation Conference)*, Las Palmas de Gran Canarias, 2012, IMACS Series in Computational and Applied Mathematics (Special Volume), 2014.
19. B. Saunders, B. Antonishek, Q. Wang, B. Miller, Dynamic 3d visualizations of complex function surfaces using X3DOM and WebGL, in *Proceedings of the 20th International Conference on 3D Web Technology (Web3D 2015)*, Crete, Greece, 219-225, ACM, New York, 2015.
20. B. Schneider, B. Miller, B. Saunders, NIST's Digital Library of Mathematical Functions, *Physics Today*, **71**(2):48-53, 2018, <https://doi.org/10.1063/PT.3.3846>.
21. B. K. Soni, Grid generation for internal flow configurations, *Computers Math. Applic.* **24** (1992), 191-201.
22. S. Steinberg, P. J. Roache, Variational grid generation, *Num. Meth. for P.D.E.s* **2** (1986), pp. 71-96, 1986.
23. J. F. Thompson, Z. U. A. Warsi & C. W. Mastin, *Numerical Grid Generation: Foundations and Applications*, North Holland, New York, 1985.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY, 100 BUREAU DRIVE, STOP 8910,
 GAITHERSBURG, MD 20899, USA
E-mail address: bonita.saunders@nist.gov