Dynamic 3D Visualizations of Complex Function Surfaces Using X3DOM and WebGL

Bonita V. Saunders*, Brian Antonishek†, Qiming Wang‡, Bruce R. Miller§
National Institute of Standards and Technology (NIST)

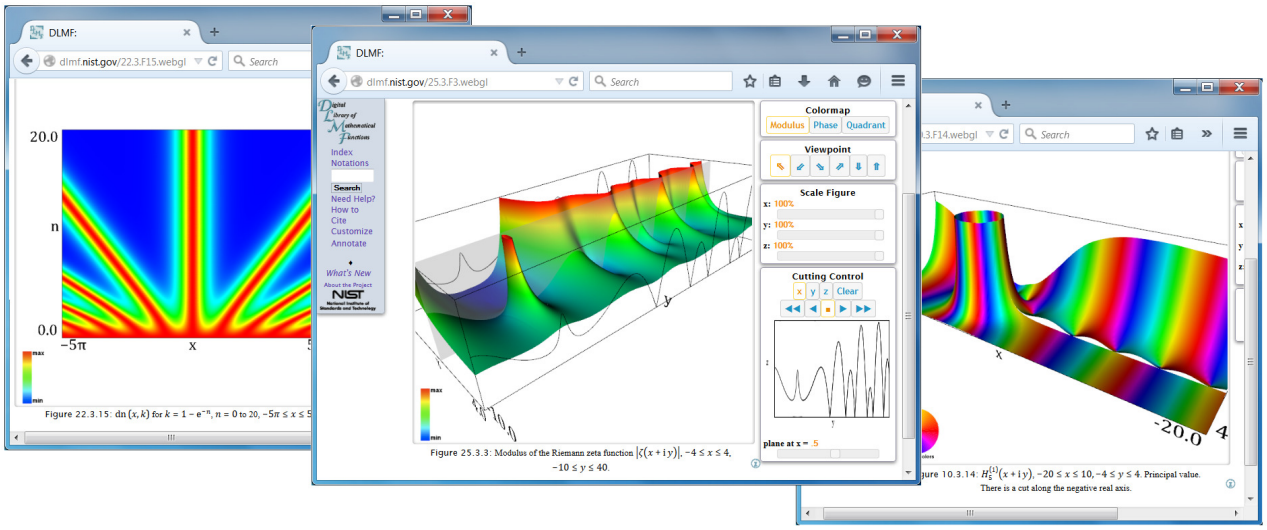

Figure 1: *WebGL Visualizations of Functions in the NIST Digital Library of Mathematical Functions (DLMF)*

Abstract

In 1997 the National Institute of Standards and Technology (NIST) embarked on a huge project to replace one of the most cited resources for mathematical, physical and engineering scientists, the *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables* [Abramowitz and Stegun 1964], originally released by the National Bureau of Standards (NBS) in 1964. The 1997 project, designed to update and modernize the handbook, culminated in May 2010 with the launch of a freely available website, the NIST Digital Library of Mathematical Functions [DLMF] (`http://dlmf.nist.gov/`), and its print companion, the *NIST Handbook of Mathematical Functions* [Olver et al. 2010]. While the presence of graphics was sparse in the original handbook, the new resource contains more than 600 illustrations of high level mathematical functions, including close to 200 interactive 3D visualizations on the website. We provide the motivation for the visualization work through the context of the project and discuss our current implementation using X3DOM and WebGL.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—Languages, Standards I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation, Virtual Reality;

Keywords: 3D web graphics, 3D visualization, WebGL, X3DOM, HTML5, X3D, VRML, special functions, digital library

1 Introduction

The NIST Digital Library of Mathematical Functions (DLMF) project started in 1997 with the goal of updating and modernizing the 1964 *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables* [Abramowitz and Stegun 1964] published by the National Bureau of Standards (NBS), the predecessor of NIST. Early in the project a few team members realized that a state of the art digital library of mathematical functions should include innovative visualizations that illuminate the behavior of high level, or special, functions that model interesting mathematical and physical phenomena. However, the challenge of creating the visualizations was magnified by the enormity of the project. This was a large project, both in terms of the size of the resource being created, and in the number of people needed to complete the work. The original handbook contained more than a thousand pages with a sizable number of pages devoted to tables of function values. While the prevalence of computer algebra systems made it possible to eliminate most of the tables, the addition of much new information and more graphs pushed the size of the new handbook to close to a thousand pages also. The website contains even more information, including additional graphs that were left out of the printed resource to ensure that it remained a manageable size.

*e-mail:bonita.saunders@nist.gov
†e-mail:brian.antonishek@nist.gov
‡e-mail:qiming.wang@nist.gov
§e-mail:bruce.miller@nist.gov

The DLMF contains 36 chapters with content authored by renowned experts in special functions from throughout the U.S. and abroad. By the launch of the DLMF in May 2010, more than fifty people had contributed to the project as editors, authors, developers and validators. The collaboration of DLMF editors and authors with graphics developers led to the creation of more than 600 graphical illustrations, including almost 200 images of function surfaces, also rendered as interactive 3D visualizations on the website.

The span of the development of the DLMF over more than a decade led to additional challenges. Over time the web, and in particular, 3D graphics on the web, evolved significantly. In order to stay current it was necessary for our work to advance in tandem. When we started the project in 1997, we used VRML (Virtual Reality Modeling Language). Over the years we added X3D as it superceded VRML and eventually redesigned our visualizations using X3DOM/WebGL as the default.

In the next section we track the development of our visualizations to motivate our current implementation in X3DOM and WebGL. In Section 3 we describe the structure of the new graphics files. Section 4 describes the interactive controls available for users and Section 5 offers some conclusions and plans for future enhancements.

2 Background

When the NIST Digital Library of Mathematical Functions (DLMF) project began, not surprisingly, most of the discussion centered around the mathematical content of the chapters and finding appropriate authors to write the details. There was an implicit assumption that there would be graphs, but little thought about who would do them or how the graphs would be rendered on the web. It was apparent after the first attempts to create function graphs for a prototype chapter that rendering the plots on the web would not be a trivial matter. There were concerns about data accuracy, i.e., determining a reliable means of computing accurate data points for the graphs, and plot accuracy issues, determining how to precisely compute and show significant attributes of functions such as zeros, poles, branch cuts and other singularities. To validate data accuracy we computed function values using at least two different methods, for example, using standard computer algebra packages such as Mathematica or Maple, codes from reliable repositories such as the NIST Guide to Available Mathematical Software [GAMS], or sometimes even personal codes from chapter authors.

However, constructing accurate plots was another challenge. We use the expression "plot accuracy" to refer to the correctness of the visual display of the data. Are zeros, poles, branch cuts and other important features clearly and correctly represented? Some authors provided initial versions of plots they wanted included in their chapters, but all functions were recomputed at NIST to ensure data and plot accuracy. When we began the project in the late 1990s and early 2000s we discovered that most commercially available packages accurately displayed 2D curves, but 3D plots were often not of the same quality. Instead of properly clipping a graph when the height was restricted to a specified range, some produced a strange "shelf" in areas where the graph fell outside the bounding box of the plot. A display of poles, that is, isolated singularities where the function converges to infinity, often looked like a rugged mountain range. Also, even when clipping was done properly, the color map often looked shadowy and uneven because of the underlying computational grid. We realized some of these problems could be solved, or at least lessened, by using our expertise in numerical grid generation.

Numerical grid generation, also known as structured grid/mesh generation, is the development of a curvilinear coordinate system where the associated tranformation maps a canonical domain, such as a square or rectangle in 2D, to the oddly shaped domain prescribed by the application. Structured grid algorithms were originally developed to create efficient finite difference codes to solve computational fluid dynamics problems over oddly shaped domains like airplane wings and fuselages, ships, and automobiles. While many researchers now prefer unstructured triangular and tetrahedral meshes typically used with finite element codes, there are still many structured codes being developed or maintained today. One of the authors modified a structured code she previously designed for modeling Bridgman growth of binary alloys in materials science [Saunders 1995; Saunders and Wang 2010] to develop boundary/contour fitted grids to create accurate plots of function graphs. Figure 2 shows a grid for the Riemann zeta function visualization seen in the foreground of Figure 1.

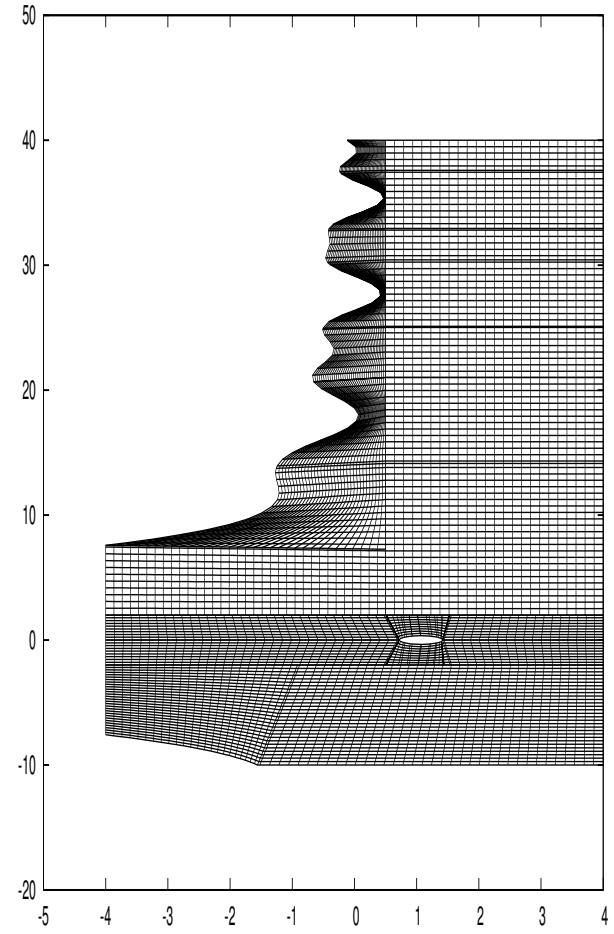

Figure 2: *Grid for DLMF function visualization.*

An even trickier problem was determining the type of file formats to use, while considering the availability of graphics plugins that might be needed by the user. In the late 1990s, VRML (Virtual Reality Modeling Language) was the most viable option since VRML browsers for a variety of platforms could be freely downloaded. Unfortunately, over the years, maintenance of some very good browsers was discontinued and the quality of new browsers varied widely. Furthermore, as we increased the complexity of our visualizations by expanding feature capabilities, it became harder to find browsers that could handle our graphics files, even when our codes appeared to follow VRML standards [Wang et al. 2007]. Noting the industry transition from VRML to X3D, one of the au-

thors designed a VRML to X3D converter [Wang], allowing us to offer our graphics files in both formats. By the launch of the DLMF in 2010 we had created almost two hundred interactive 3D visualizations of mathematical function surfaces in both VRML and X3D format [DLMF ; Olver et al. 2011].

Still, our ultimate goal was to find a way to make our visualizations accessible to users on Windows, Mac, or Linux platforms. While we successfully found VRML/X3D plugins that worked for Windows or Mac operating systems, we were never able to find one that could consistently handle our complex visualization files in Linux. Furthermore, the requirement that users download a plugin was inconvenient both for users and maintainers of the visualizations since updates of the plugin, web browser, and even the operating system could affect the quality of the graphics display. Motivated by these concerns, we began monitoring the development of WebGL, a JavaScript API (application programming interface) for rendering graphics in a web browser without a plugin. Then, thanks to the work of Johannes Behr and colleagues [Behr et al. 2009] on X3DOM, which permits the direct integration of X3D nodes into HTML DOM (Document Object Model) content, we decided that converting the DLMF visualizations to WebGL by using the X3DOM framework to build the application around our X3D codes was a feasible approach. Our first attempts at creating a WebGL file were tested using one of the few WebGL enabled browsers available at the time, a Mozilla Firefox beta version aptly called Minefield. Our success led us to research other work involving the development of X3DOM and WebGL applications. Our early work was bolstered by tips obtained from examining preliminary X3DOM/WebGL work done by Sandy Ressler [Ressler] and the work of Steven Birr, et al., on the LiverAnatomyExplorer WebGL Tool [Birr et al. 2013]. 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 Update: Version 1.0.7 released on March 21, 2014 [DLMF].

3 X3DOM/WebGL Integration

By building our application using the X3DOM framework we were able to reuse most of our X3D code to create the WebGL files. The most intensive work was the recoding of the dynamic displays and interactive features. We first created stand alone WebGL files. We then worked with Bruce Miller, architect of the DLMF website, to make coding changes to ease the integration of the visualizations into DLMF HTML files. We also made style changes to achieve a more polished look. The main goal was to reproduce and enhance all the capabilities available in our VRML/X3D visualization and provide additional capabilities where possible. Figure 3 shows a portion of the HTML code for the complex gamma function. JQuery, X3DOM and DLMF_webGL JavaScript files produce the animation and surface interaction capabilities. The X3D code located in the center of the HTML code provides the surface data and settings for the scene.

All features available in the VRML/X3D visualizations are still available except the options for different types of axes, which appeared to be unnecessary. The colors for the WebGL display are more vibrant and curves located on bounding boxes and pop-up windows much clearer. We did notice that color maps for some figures showed anomalies not seen in the VRML/X3D visualizations. We traced the problem to a slight overlap, or self-intersection, of grid lines in some areas. This problem was solved by improving the computational grids for the figures affected. Slider bars, which offered users more control of some features, were redesigned to provide more information and allow for user input of values.

```
<!DOCTYPE html><html>
<head>
<title>DLMF: </title>
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
<link rel="shortcut icon" href="./style/DLMF-16.png"
type="image/png">
<script type="text/javascript"><!--
      var PATH="DLMF:/5.3.F4.webgl";
      var ROOT="./";
      //--></script>

<script src="./style/jquery.js"
type="text/javascript"></script>
<link rel="stylesheet" href="./style/jquery-ui.css"
type="text/css">
<script src="./style/jquery-ui.js"
type="text/javascript"></script>
<link rel="stylesheet" href="./style/x3dom.css"
type="text/css">
<script src="./style/x3dom.js"
type="text/javascript"></script>
<link rel="stylesheet" href="./style/DLMF_webGL.css"
type="text/css">
<script src="./style/DLMF_webGL.js"
type="text/javascript"></script>

<div id="mainScene">
   <X3D xmlns= ... id="x3dElement" ...
       <Scene><Background skyColor="1.0 1...
              showS...description="Bottom">...
              </Viewpoint></Group>
       </Scene>
   </X3D>
</div>
.
.
<div class="buttons">
<input checked id="modulus" name="colormap" title="Co...
.
<div class="sliderControl">
<div>
<label class="sliderLabel"><span class
.
</div>
</body>
</html>
```

Figure 3: *Sample WebGL code for embedded DLMF function visualization.*

The WebGL visualizations appear to work well on WebGL enabled browsers on Windows, Mac, or Linux platforms. For now, we offer WebGL as the default format but continue to offer VRML and X3D options for those who prefer that format.

4 User Controls

As shown in Figure 4, the control panel for a DLMF visualization, located to the right of the function graph, is subdivided into three or four sub-panels that provide users with manipulative capabilities that complement typical interactive features like rotation and zoom. The topmost panel shows the options “Modulus”, “Phase”, or “Quadrant” which allow the user to vary the surface color mapping. If the function is real valued, rather than complex, “Modulus”, or height, is the only option and the color map panel does not appear.

Although a user may click on a surface and rotate it in any direction, the second panel offers several “canned” viewpoints, including two representing an observer looking at the surface from directly above or below. Scale Figure controls let a user scale the surface down in any of the three coordinate directions.

The Cutting Control panel provides the most extensive capabilities, allowing the user to observe the curves of intersection created as a plane moves through a surface in each coordinate direction.

We discuss the features of each control panel in more detail below.

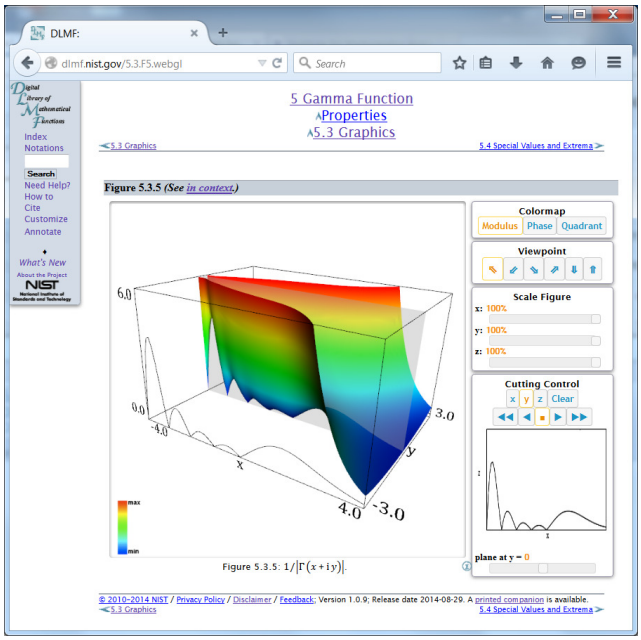

Figure 4: *WebGL visualization embedded in DLMF webpage.*

4.1 Color Map Control

Surface visualizations in the DLMF represent functions of the form $z = f(x, y)$ by the height z or the modulus, $|z|$, for complex functions, over the $x \times y$ plane. We use color to augment these visualizations, either to reinforce the recognition of the height, or to convey an extra dimension to represent the argument, or phase, of complex valued functions.

4.1.1 Height/Modulus Mapping

To provide an easily interpreted encoding of surface heights, a rainbow-like mapping of height to color is used. The following figure illustrates the piece-wise linear mapping of the height to each of the color components red, green and blue, written as $\langle R,\ G,\ B\rangle$.

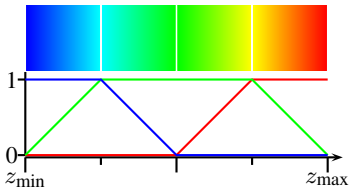

Figure 5: *Mapping of height to color*

Mathematically, we scale the height to h lying in the interval $[0, 4]$ and the components are computed as follows

$$\langle R,\ G,\ B\rangle = \begin{cases} \langle 0,\ h,\ 1\rangle & \text{if } 0 \le h < 1 \\ \langle 0,\ 1,\ 2-h\rangle & \text{if } 1 \le h < 2 \\ \langle h-2,\ 1,\ 0\rangle & \text{if } 2 \le h < 3 \\ \langle 1,\ 4-h,\ 0\rangle & \text{if } 3 \le h \le 4 \end{cases}$$

The function graph seen in Figure 4 is displayed with a height (modulus) color mapping.

4.1.2 Phase Mappings

By painting the surfaces with a color that encodes the argument, or phase, $\mathrm{ph} f$, both the magnitude and phase of complex valued functions can be displayed. We offer two options for encoding the phase.

Quadrant Mapping The quadrant mapping uses a four color scheme for phase that quickly indicates in which quadrant z lies: the colors blue, green, red and yellow are used to indicate the first, second, third and fourth quadrants, respectively. As a mnemonic, the colors are sorted alphabetically.

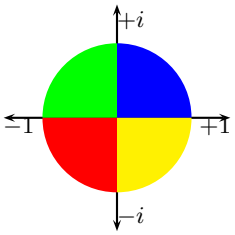

Figure 6: *Four-color map*

Continuous Phase Mapping For the continuous phase mapping, denoted simply as "Phase" in the DLMF, we map the phase continuously onto the hue, as both are periodic. In doing this, however, we would like to place the mathematically significant phase values, specifically the multiples of $\pi/2$ correponding to the real and imaginary axes, at more immediately recognizable colors.

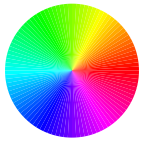

Figure 7: *CMYK color wheel*

The conventional CMYK color wheel (not to be confused with the traditional Artist's color wheel) places the additive colors (red, green, blue) and the subtractive colors (yellow, cyan, magenta) at multiples of 60 degrees. In particular, the colors at 90 and 180 degrees are some vague greenish and purplish hues.

We therefore use a piecewise linear mapping as illustrated below, that takes phase 0 to red, $\pi/2$ to yellow, π to cyan and $3\pi/2$ to blue. Specifically, by scaling the phase angle in $[0, 2\pi)$ to q in the

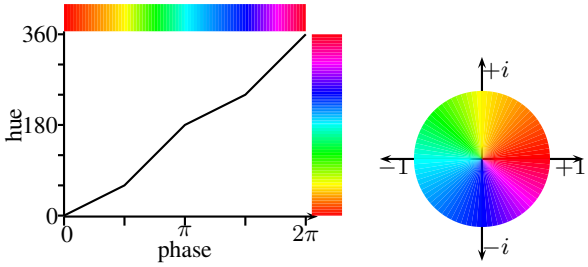

Figure 8: *Continuous phase mapping*

interval $[0, 4)$, the hue (in degrees) is computed as

$$\text{hue} = 60 \begin{cases} q & \text{if } 0 \le q < 1 \\ 2q - 1 & \text{if } 1 \le q < 2 \\ q + 1 & \text{if } 2 \le q < 3 \\ 2(q - 1) & \text{if } 3 \le q < 4 \end{cases}$$

Figure 9 shows a Hankel function displayed with continuous phase and quadrant color mappings.

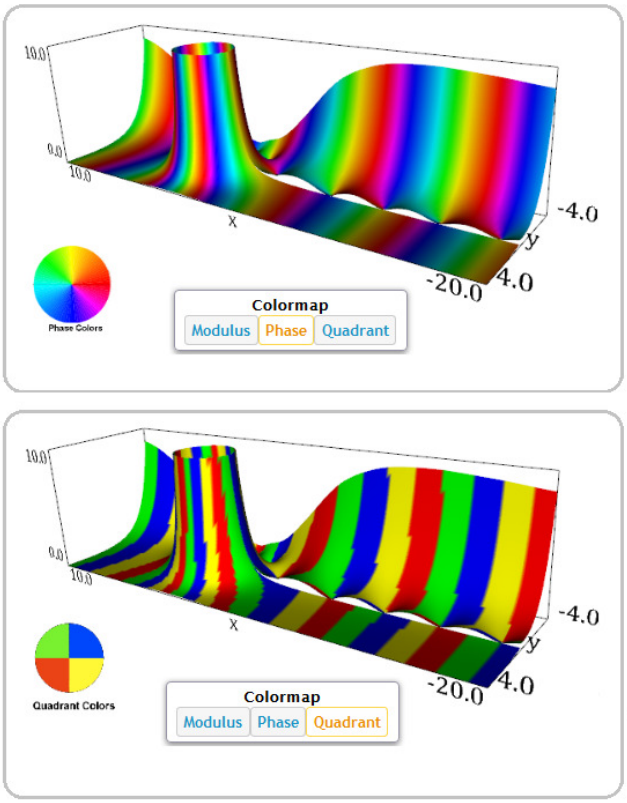

Figure 9: *Continuous phase and quadrant colormaps.*

4.2 Cutting Plane Control

The cutting plane feature allows a user to observe the intersection curves produced as a coordinate plane moves through a function surface in the x, y, or z coordinate direction. Digital Video Recorder (DVR)-like controls make it easy to play, rewind, fast-forward or pause the animation. The user can also interactively move the plane by using the slider bar, or place the plane at a specific location by entering the value in the field above the bar. The intersection curves are shown on opposite sides of the bounding box enclosing the surface and also displayed in a pop-up window located on the control panel.

Currently, all cutting plane computations are approximations based on initial function data computed using the customized computational grids. The possibility of computations using real-time function data may be investigated in the future, but the complexity of some special function algorithms may make this infeasible in some cases. Figure 10 and Figure 11 show two examples from the DLMF where cutting planes are moving in the x direction or z direction.

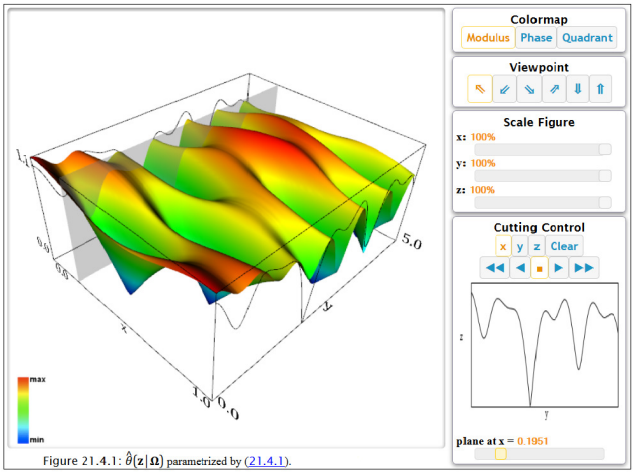

Figure 10: *Cutting plane moving in direction of x axis.*

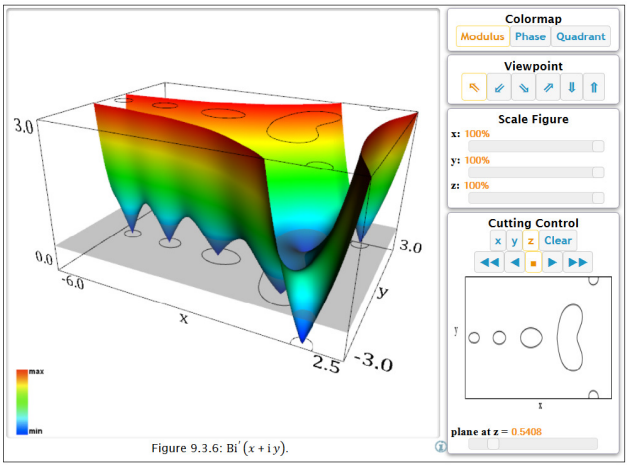

Figure 11: *Cutting plane moving in direction of z axis.*

4.3 Viewpoint and Scale Controls

A user can easily click on a surface and rotate it in any direction, but there are also hard-coded viewpoints based on axis orientation that a user may select. In the center of Figure 12 we show our standard viewpoint where one domain coordinate increases from left to right and the other, from front to back. While this is often the default view seen when first accessing a visualization, there are cases where an alternate, more informative, view is shown initially. The other viewpoints shown in Figure 12 represent what an observer sees when looking at the function from the indicated locations. Additional viewpoint options show a view from the "back", "top" or "bottom" of the function. When a user selects one of the hard-coded viewpoints, the axis labels rotate appropriately.

Users can change the scaling of a surface in each coordinate direction by moving a slider bar or by entering a number between 0 and 100 in the field above the bar. More than one direction may be scaled simultaneously if desired. In Figure 13 we create a density plot by selecting a top viewpoint and scaling the figure down to near zero in the z, or vertical, direction.

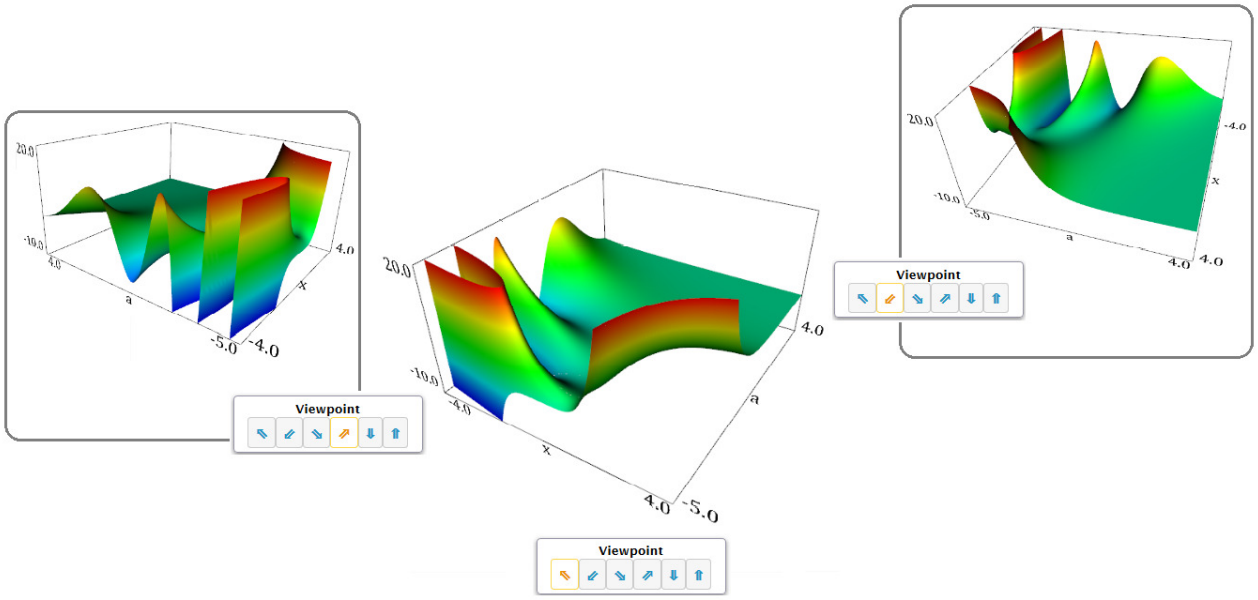

Figure 8.3.6: $\gamma^*(a,x)\ (= x^{-a}P(a,x)),\ -4 \le x \le 4,\ -5 \le a \le 4.$

Figure 12: *Users can rotate surfaces in any direction or choose hard-coded viewpoints.*

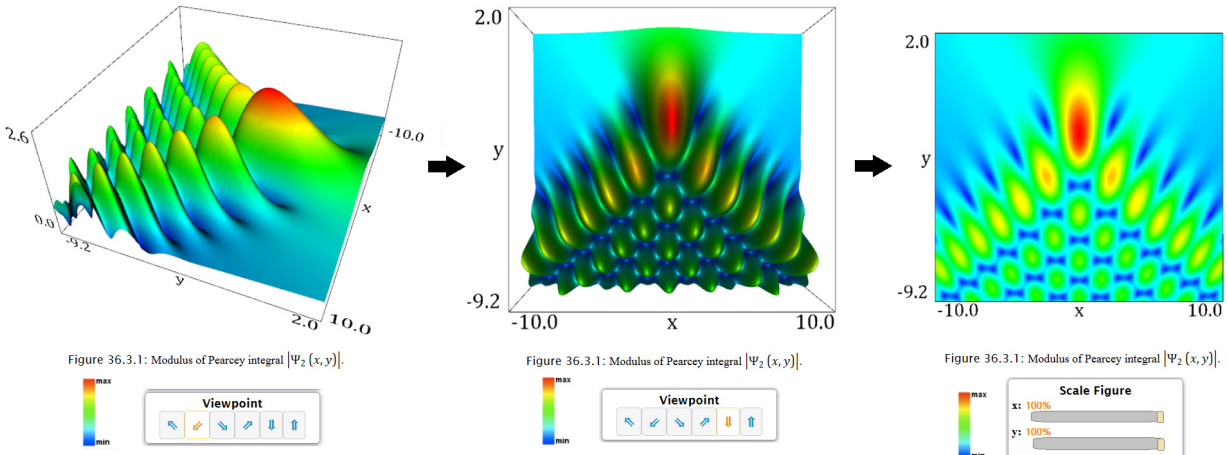

Figure 36.3.1: Modulus of Pearcey integral $|\Psi_2(x,y)|$.

Figure 36.3.1: Modulus of Pearcey integral $|\Psi_2(x,y)|$.

Figure 36.3.1: Modulus of Pearcey integral $|\Psi_2(x,y)|$.

Figure 13: *Creating a density plot.*

5 Conclusions

To a large extent, the development of the 3D visualizations of complex functions for the NIST Digital Library of Mathematical Functions has paralleled the evolution of technology for displaying 3D graphics on the web. From the first renderings in VRML to translations into X3D to the current implementation in X3DOM and WebGL, we have continued to take advantage of new technologies as they emerged. The conversion from VRML to X3D produced a change in file format, but little change in the display or performance of the visualizations.

On the other hand, the implementation in X3DOM and WebGL amounted to a full scale overhaul. Some improvements appear to be a direct result of the new technology. DLMF visualizations are now directly embedded into website pages, surface color maps are more vibrant, cutting plane curves are clearer, and the movement of the planes, smoother and faster. These improvements motivated additional work such as a recalculation of contours and computational grids to improve plot accuracy, improvement of cutting plane calculations near interval endpoints, elimination of images from axis labels, addition of more control options, and the overall improved arrangement of the visual display. However, perhaps the most significant improvement relates to portability, a problem that has plagued us from the beginning. Since the embedded visualizations do not require a special plugin, for the first time, they can be seen on Windows, Mac, or Linux platforms using most common web browsers.

We are currently looking at several improvements. One task in progress is the restructuring of the graphics files to make it easier to regenerate visualizations when data or display attributes are changed. We are also studying the feasibility of adding a continuous spin feature similar to what was seen in some of the early VRML browsers and plugins. We would also like to implement a true zoom capability where the function values are actually recomputed when a user clicks a surface to zoom in or out. This may be somewhat challenging because of the complexity of some existing algorithms for computing special functions and may also require the use of special grid refinement and coarsening techniques such as hierarchical grid generation schemes.

Acknowledgements

We would like to thank Sandy Ressler for helpful suggestions, support, and encouragement for this work.

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

ABRAMOWITZ, M., AND STEGUN, I. A., Eds. 1964. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. No. 55 in National Bureau of Standards Applied Mathematics Series. U.S. Government Printing Office, Washington, D.C. Corrections appeared in later printings up to the 10th Printing, December, 1972. Reproductions by other publishers, in whole or in part, have been available since 1965.

BEHR, J., ESCHLER, P., JUNG, Y., AND ZOLLNER, M. 2009. 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., 127–135.

BIRR, S., MONCH, J., SOMMERFELD, D., PREIM, U., AND PREIM, B. 2013. The LiverAnatomyExplorer: A WebGL-based surgical teaching tool. *IEEE Computer Graphics and Applications 33*, 5, 48–58. http://liveranatomyexplorer.steven-birr.com/.

DLMF. NIST Digital Library of Mathematical Functions. http://dlmf.nist.gov/, Release 1.0.9 of 2014-08-29. Online companion to [Olver et al. 2010].

GAMS. Guide to Available Mathematical Software. http://gams.nist.gov/.

OLVER, F. W. J., LOZIER, D. W., BOISVERT, R. F., AND CLARK, C. W., Eds. 2010. *NIST Handbook of Mathematical Functions*. Cambridge University Press, New York, NY. Print companion to [DLMF].

OLVER, F. W. J., LOZIER, D. W., BOISVERT, R. F., AND CLARK, C. W. 2011. A special functions handbook for the digital age. *Notices Amer. Math. Soc. 58*, 7, 905–911.

RESSLER, S. WebGL/X3DOM experiments. http://math.nist.gov/~SRessler/webgl.html.

SAUNDERS, B., AND WANG, Q. 2010. Tensor product B-spline mesh generation for accurate surface visualizations in the NIST Digital Library of Mathematical Functions. In *Proceedings of the Seventh International Conference for Curves and Surfaces (MMCS 2008), Tonsberg, Norway*, Springer, 385–393.

SAUNDERS, B. V. 1995. A boundary conforming grid generation system for interface tracking. *Computers Math. Applic. 29*, 10, 1–17.

WANG, Q. VRML97 to X3D translator. http://ovrt.nist.gov/v2_x3d.html.

WANG, Q., AND SAUNDERS, B. 2005. Web-based 3D visualization in a digital library of mathematical functions. In *Proceedings of the Tenth International Conference on 3D Web Technology, Web3D 2005*, ACM, N. W. John, S. Ressler, L. Chittaro, and D. A. Duce, Eds., 151–157.

WANG, Q., SAUNDERS, B., AND RESSLER, S. 2007. Dissemination of 3D visualizations of complex function data for the NIST Digital Library of Mathematical Functions. In *Proceedings of the Twentieth International CODATA Conference, Beijing, China, 2006*, 146–154.