



On Mathematical Visualization in Education

Alexey Karpov¹, Vladimir Klepov², and Alexey Nikitin^{3,4}

¹ University of Bonn, Bonn, Germany

karpovad@yandex.ru

² Yandex, Moscow, Russia

V.klepov@gmail.com

³ Lomonosov Moscow State University, Moscow, Russia

nikitin@cs.msu.ru

⁴ National Research University Higher School of Economics, Moscow, Russia

Abstract. This article treats the use of modern information technologies in the classroom educational process. It emphasizes the need to combine symbolic and visual mathematics, describes the problems associated with this issue, provides a review of the existing systems and a list of the requirements a modern mathematical visualization system must meet. The article is conceived as a manifesto for the use of mathematical visualization in education. The article then goes on to describe the developments of the authors' research group. The functionality of visual-math.ru website is described. This resource contains an ample collection of visual and text modules for teachers to create presentations based largely on visual materials. The most important part of the article is the description of fast and powerful JavaScript visualization libraries developed specifically for the project: Skeleton and Grafar. The former is designed to display two-dimensional graphs, while the latter visualizes three-dimensional objects with transparency and illumination effects. Both libraries are capable of processing large element sets in near-real time. In conclusion, selected examples of visualizations created using the libraries use are provided, including the ones used in courses on mathematical analysis and analytical geometry.

Keywords: Information technologies in the education · JavaScript graphics libraries · Mathematical analysis

1 Motivation

1.1 Images in Mathematics

Images in mathematics bridge the gap between symbolism and imagination. David Hilbert [3] notes the conflict between the two trends: mathematicians strive for a logically consistent symbolic abstraction while trying to maintain intuitive understanding of a problem. In ancient India geometric conjectures

were proven in a very peculiar manner. Having formulated the premise, the mathematician plotted the shapes necessary for the proof, provided brief comments, and wrote “Watch!” afterwards. It was assumed that the person willing to understand the problem at hand can do so himself by studying the images provided, with no further explanation [7]. As an example, we include a plot (Fig. 1) attributed to Ganesha, 16th century [30].

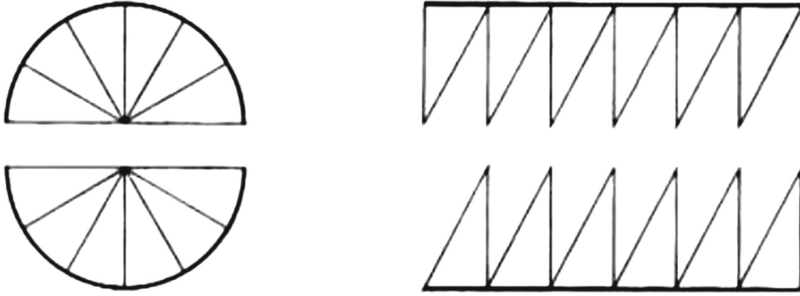


Fig. 1. An illustrated explanation of the area equality between a circle and a rectangle with sides equal to the radius and the half-arc length.

1.2 Symbolic Mathematics

However, after that period the dominant method in mathematics began to change. René Descartes’ works on algebraic geometry have turned mathematics into a primarily symbolic science. Images were reduced to guesswork and example aids, losing the role of the primary means of proof. By the end of the 19th century the discovery and further studies of the more abstract objects, such as abstract algebras and higher-dimensional spaces, had made mathematics even less intuitive. Nowadays the less-formal studies would not be considered real mathematics.

Symbolic description of mathematical objects is precise but it takes a lot of training to gain intuition into symbolic manipulations. Scientists can easily lose track of the real-world problem they are solving and come to an absurd conclusion. The key skill of an applied mathematician is correlating the abstraction with the reality. Even in pure mathematics the studies of abstract concepts begin by studying special cases that are easy to understand [2, 5]. A famous Russian mathematician and educator Vladimir Arnold referred to mathematics as “experimental science” [11]. Visualization can help resolve these issues. Science is easier and more humane with images. A layman unfamiliar with the notation can readily see similarities between real-world objects and charts, which is a key to introducing younger students to advanced mathematical concepts.

In Sect. 2.1 we highlight the importance of computer technologies in education. In Sects. 2.2, 2.3 and 3.1 we describe the current state of interactive images

in classical mathematical education. At the same time, we introduce technical constraints on an interactive system suitable for educational use. A list of popular educational tools that could fit these requirements is provided in Sect. 3.2. After that, we describe our own technical developments in Sects. 3.3 and 3.5. Finally, we show some examples created with these libraries.

2 Visualization in Education

2.1 Computer Technologies in Pedagogy

Computers are responsible for several effects that would transform education in the years to come. Computer Algebra Systems (CAS) and Automated Theorem Proving (APT) software are already capable of solving well formalized problems intractable to humans in an efficient and correct manner. Solutions to numerous problems not solvable symbolically (most prominently, differential equations, including scientific and engineering applications) can be approximated using numerical computing. Future mathematicians need not be trained to carry out computations according to a predefined algorithm with no errors — computers are better suited for such tasks. The key skill of a mathematician working in the industry is to formalize applied problems and to validate the sanity of the computer-produced solution. Mathematical education can no longer ignore these trends, and therefore it needs visualization.

On the other hand, computers can tackle the task of visualizing complex objects with ease. Contemporary scientists process and analyze billions of observations and publish their findings in a comprehensible format. A study scale this wide was unimaginable even a couple of decades ago. No more are mathematicians bound by the rough, static two-dimensional blackboard. In [10] the new wave of interest to mathematical imaging is referred to as the “visualization renaissance”. Another advantage of computer graphics is its inherent interactivity. The users can better understand the model at hand while altering the parameters and watching the change induced on the image.

Mathematical visualization still has a lot of ideas to borrow from several connected fields of study. Information graphics is a well-developed and popular topic of study with a high volume of literature summarizing and describing best practices (e.g. the works of Edward Tufte, [6]). Theory of automated visualization is largely confined to proprietary technologies but some academic works are available and we can expect to acquire further material as the patents expire. Finally, software developers and psychologists have advanced the theory of human-computer interaction [1,4] and there is no shortage of the most experimental concepts, such as the use of augmented/virtual reality.

2.2 Problem Statement

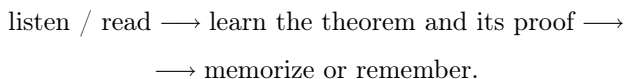
In this section we approach to the realm of school and university education. Currently, students and children are dominantly visual learners. However, visual

materials are scarce in classrooms, mostly due to technical and methodological difficulties. In the best case, blackboard sketches or presentations are used. Movement is usually displayed schematically. We believe that using computer technologies can greatly boost the students' understanding of the material and must therefore be pursued by all means.

Several libraries of visual materials exist worldwide such as *MIT Open Course Ware*, [12], *The d'Arbeloff Interactive Mathematics Project*, [13] etc. Powerful information systems such as *Wolfram Mathematica*, [14] allow their users to create new materials themselves. However, the primary focus of our work is lecture use, and the described solutions are require adaptation for such cases. Mathematics has a wide array of remarkable pedagogical findings, scattered throughout the diverse courses, with powerful geometrical ideas beside them. Unfortunately, the vast majority of these has not been illustrated, while the rest can be improvement hugely by uncapping the potential of computed-driven interactive visualization.

2.3 An Interactive System Concept

Our task is to create a framework that allows students understand the educational material deeper. The following diagram shows the current state of the education process:



After the *listening/read* step, we suggest that the students should look at visual explanation of theorems and statements or even to interact with these images. It helps a person understand the subject better and positively impacts the effectiveness and the duration of remembering the educational material.

The importance and the quality of illustrations are also emphasized in modern mathematical textbooks distributed by OpenStax, [17].

A large library of so-called visual modules shall be developed soon, covering at least classical mathematical courses. Each module must include well-arranged and relevant text.

In addition, there must be a convenient and easy-to-use web platform storing the visual modules library. The platform is supposed to be used by both students and lecturers. The lecturers should have access to the library that can be used to construct a presentation for coming lectures. Later the presentation is to be demonstrated during a class using the lecturer's computer or mobile device.

At the same time all students with student accounts log in to the platform using their mobile devices, find the lecturer in the list, and connect to the current lecture broadcast. The state of the web page on a student's device is synchronized with the lecturer's page. The lecturer comments the slides content and proves the theorems/statements, while also showing the visual modules integrated into the lecture.

The lecturer is able to ask questions or conduct tests during the lecture via the platform. In a short time, they can see the statistics of answers including the number of participants and the number of people who picked a specific option. These interactive tests help lecturers get an instant feedback and thus adjust the course effectively.

3 The Design Requirements and an Overview of the Existing Solutions

3.1 What Should a Visualization System Be Like

A good mathematical visualization system bridges the gap between the logic and the intuition. Developing such a system is an inherently interdisciplinary endeavor, taking cues from formal systems, numerical methods, programming, and visual design. It is very rare for a single person to be acquainted with all of these areas, let alone be proficient in these. This calls for a higher-dimensional programming interface that is easily usable by people trained in mathematics with sensible visual defaults, and encouraging simple programming.

The system should make use of its underlying media, the computer. To help the users explore complex mathematical objects, the system should not only be interactive but also function in real time. Another use case computers excel at is creating three-dimensional graphics. As such, computer adds two usable dimensions to a printed graphic: one spatial and one temporal. It will take us, the scientific community, a lot of time to unleash the full potential of this freedom.

The use of the system in education adds further design constraints. As the education should be free and generally available, the system should be cross-platform and accessible, with no major functional deficiencies, on low-end and mobile devices.

To sum up, here is a brief list of design constraints along with the relevant solutions:

- **ease of use:** high-level interface with sensible defaults;
- **availability:** a JavaScript program works across all major devices, aided by the browser runtime;
- **fast 2D graphics:** rely on Scalable Vector Graphics (SVG) [35] for interoperability with standard web technologies and ease of distribution;
- **3D graphics:** WebGL technology. Compatibility with legacy browsers should be provided via canvas or SVG fallbacks [36, 37];
- **interactivity:** efficient computational models.

3.2 Overview of the Existing Systems

Major computer mathematical systems support plotting out of the box. Maple [15], MatLab [16], Mathematica [14] are powerful, widely adopted packages. However, their use in educational visualization is limited: even the special educational licenses are expensive, mobile device support is lacking, and the users need to

learn a special programming language. Open-source alternatives, such as R [18], Octave [19], Julia [20], only solve the license cost issue.

Mathematical libraries for general-purpose programming languages offer a more pragmatic alternative. SciPy [21] and matplotlib [22] have emerged as the go-to data analysis tools following the increased popularity of python as a scientific computing language. While this is great step forward, python programs are still not easily distributed, especially for mobile devices. Lately there has been a lot of enthusiasm around combining a python back-end with the browser interface using Jupyter and JavaScript extensions such as Bokeh but this solution still offers only limited interactivity.

Distributing visualizations as images and videos is adequate for some cases but offers little to no interactivity to the end user. Computers are capable of much more. Finally, these visualizations still need to be generated somehow — presumably using one of the aforementioned mathematical packages.

The golden standard of web-based visualization is set by d3.js [23], the eighth most starred library on GitHub as of September 2018. The library is built using JavaScript programming language and conventional web technologies. This enables the end users to freely interact with the data looking for interesting patterns. While d3 could be our library of choice, it still lacks in certain aspects. The system is built on vector graphics, and displaying three-dimensional objects is not possible without serious rework. Approximating algebraic objects with datasets is a separate complex task not handled by d3.

There are several online systems for mathematical visualization. Desmos offers no 3D capabilities. GraphyCalc [24] only supports explicit functions of two variables. Wolfram Alpha [25] delegates computations and rendering to a server back-end, limiting real-time client interactivity. MathBox is, as of 2018, still an early prototype with unclear development status. Finally, GeoGebra is the system most suitable for our use case but the 3D support appears to be an afterthought.

As shown in this section, the solutions available do not meet our design constraints. In the following sections we go on to describe our own solutions to the problem — JavaScript libraries Skeleton and Grafar Sect. 3.5, suitable for 2D and 3D mathematical visualization, respectively.

3.3 VisualMath.ru Web Application

A discussion of mathematical visualization is inseparable from the educational use cases. As stated in Sect. 2.3, we keep in mind the typical scenario we have all participated in. The students attend the lecture where the lecturer walks them through the necessary theory, supporting his arguments with illustrations.

Our team has already developed a functional prototype of the system described in [46]. It is a conventional client-server application consisting of a REST API and a single page application (SPA). We have chosen the SPA approach because it is well suited for the massively interactive concepts described in Sect. 2.3.

Apart from our custom-made solutions, the application relies heavily on KaTeX for client-side TeX rendering [27]. While omitting some of the more advanced TeX features, it offers an improved performance over MathJax, [28, 29]. This library also powers the popular online learning platform, Khan Academy.

Developing a content management interface for lecturers is another high-priority goal of our project. Mathematics professors are often conservative when it comes to technology. Few of us can be truly called open minded. To capture the attention of this audience, we need to put a lot of effort into the ease-of-use. The concept is fairly simple: a lecture is a board, onto which a lecturer places colored cubes, each representing some content module. The content types supported include but are not limited to, illustrations, theoretical blocks, tests. Having assembled the lecture, the content editor proceeds to edit the modules. Importing educational content from the editor’s computer is also supported.

The web application prototype implements many of the ideas described. The users can create and edit lectures, show the visual modules to the audience and run tests to evaluate the students’ understanding of the topic (see Fig. 2).

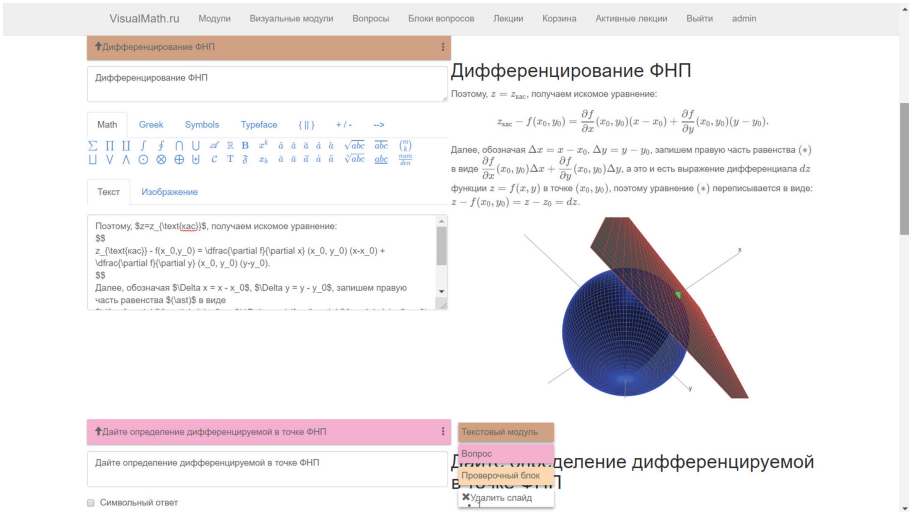


Fig. 2. Creating a lecture on visualmath.ru

For even further immersion of the students, our team has developed an Android application that can be used to view the lectures [38]. The iOS application is currently under development.

3.4 Grafar, a Library for Three-Dimensional Interactive Visualization

Grafar is a JavaScript library for creating interactive three-dimensional visualization developed specifically for VisualMath.ru project. As opposed to Wolfram,

the library works in a browser with no proprietary plug-ins. Grafar is also among the fastest visualization libraries: the tests with computing explicitly defined surfaces over 1000×1000 grid have shown usable performance of 30 frames per second.

Grafar is designed from the ground up for mathematical visualization, featuring a high-level interface for working with analytical objects. Instead of generating the dataset to be displayed ad-hoc, the developers define the values of free variables along with their mappings. This model encourages very concise and readable code:

```
const x = grafar.range(-1, 1, 100).select(); const y = grafar.range(-1, 1, 100).select(); const z = grafar.map([x, y], (x, y) => Math.cos(x) * Math.sin(y)); grafar.pin([x, y, z], surfContainer);
```

The system relies on a three-step conceptual model of visualization:

- Find an algorithmic form of the objects, compile functions for efficient use on numeric arrays;
- Sample the free variables and find a finite set of points that reasonably approximate the object;
- Render the dataset, respecting inter-point topology.

Theoretically, any visualization library (d3, highcharts, echarts) can be used for the rendering step through custom bindings. However, the problem with plugging generic statistical visualization systems is their focus on aggregated representations over performance. While such preprocessing makes sense when helping users make sense of massive statistical datasets, it is useless for algebraic objects. The primary mathematical use-case is displaying thousands of points moving unpredictably every frame. The existing SVG-based libraries are simply inadequate for this scenario.

Numerical computing, even in the simplest form of applying a complex function over a set of points is not a classical web-development problem either. Grafar uses a concept called Reactive Programming (RP) to efficiently implement real-time updates. This paradigm has made its way into mainstream front-end programming lately: the massively popular Vue.js framework [31] uses pull-based RP (the submodel Grafar also uses), while the more experimental cycle.js [33] pushes further the original electrical engineering metaphor. RxJS [32] is the go-to library for state management in Angular applications, with versions for all the major languages developed by Microsoft. While handling complex systems with interconnected dependencies, reactive programming still manages to be easy for the less-technical users: this is the model used by Microsoft Excel to recompute the formulas on source cell updates. Architects employ a visual programming software, Grasshopper, for designing procedural buildings.

Reactive Programming is an umbrella term for programming techniques using explicit Data Flow Graph (DFG). The nodes of such a graph replace the conventional variables, with the edges connecting the variables that depend on one another. This allows for reliable and efficient updates of all the dependent variables when a free variable changes — we locate the node’s children and recompute their values, repeating recursively. Not only does this avoid unnecessary

recomputation of those variables that do not depend on the changed one but it also allows dead code elimination and automatic parallelization. Finally, as a declarative technique, reactive programming is easy to understand for those already familiar with mathematical notation.

Unfortunately, the existing JavaScript libraries for reactive computations can not be readily used for data volumes of the magnitude encountered routinely in mathematical visualization and numerical computing. Connecting the SciJS ecosystem developed for numerical computing with reactive programming presents an exciting challenge for the open source community, if only too specialized for mainstream adoption.

Finally, Grafar uses WebGL for fast rendering. This technology allows the browser to access the GPU, unleashing the power of massively parallel computations to update the image in real time. Vector graphics used by d3 can not be used to display a large number of objects, while with the native canvas API we are still stuck in the single browser thread for projecting the multidimensional data onto the viewport. There is a multitude of specialized WebGL libraries, most prominent in cartography — MapboxGL.JS and deck.gl are two popular examples.

While the mathematical and scientific imaging possesses a distinctive visual style setting it apart from the commercial information graphics, this more often than not arises unintentionally from the lack of care. Scientists do not assign significant weight to matters of “aesthetics”, often falling back to the default values of their chosen plotting library and validating this behavior by their fellow academics. However, the right choices of representations and color maps can make all the difference. Grafar can be used to display higher-dimensional objects in various ways: through small multiples, parallel coordinates or coloring. Optimal color maps can be generated by standalone packages, yielding more attractive and comprehensible results.

Thanks to its massive performance and layered architecture, Grafar can also be used for generic data visualization tasks. Discrete mathematical objects, such as digital signal processing or graph theory, are not out of reach either. Datasets with hundreds of thousands observations can be displayed with a usable frame rate even on lower-end mobile devices.

A stable version of Grafar is distributed through npm, the de-facto package manager of JavaScript development. Source code, written in TypeScript, is available on github, [34].

3.5 Skeleton

Another part of our work is the library for two dimensional visualization called Skeleton. It is a foundation for many of our 2D programs. The library implements our other idea — state synchronization between the lecturer’s computer and the student’s one. If a lecturer changes coordinates of a graph, moves some elements or interacts with buttons and sliders then exactly the same changes happen on computers (or smartphones) of students simultaneously. The core concept of Skeleton library is being an easy-to-use tool to build mathematical visualizations

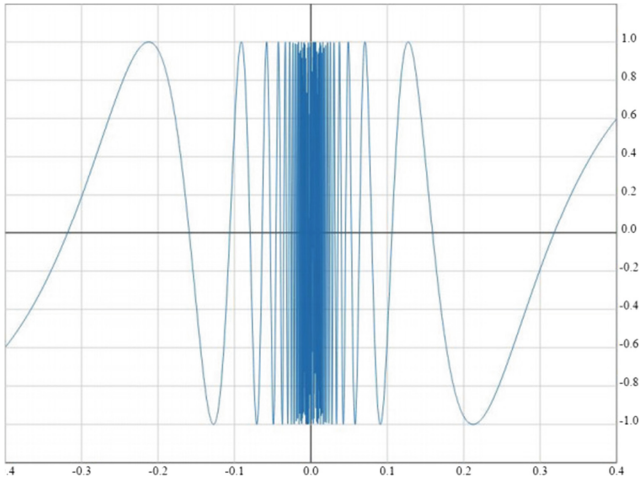


Fig. 3. An example of Skeleton-based program drawing $\sin(1/x)$ function.

(e.g. Fig. 3). Later we realized that the library shows itself best when we create visualization of calculus objects and theorems. Each graph object has its own API making it simple to change the state. As a result, it allows to create animated graphs easily [41]. The synchronization is done by WebSocket browser API [43]. This requires the state of each object, including the graph itself, to be serializable to string. However, the library is not tied to any transport and WebSocket can be replaced with similar tools, both browser-native and libraries, transferring data from server to client in real time. For example, it can be WebRTC which transfers data peer-to-peer [44, 45].

4 Results and Conclusion

At the end of the article we provide some example visualizations from the Visual-Math.ru project. Most of them are from the calculus course. Because of our teaching background, this area is better represented in our project than the other university and school courses.

In the first part, there are images of 2D library Skeleton. For some of the images we also comment on rendering time if there is a large number of elements. We used a laptop with Intel Core i5 2.3GHz, 8 Gb RAM. This setup is typically cost less than 1000\$ and usually available for many laptop owners.

In the second part we provide examples of Grafar visualizations. One of the examples contains a comparison with a similar image created in MathWorks' MATLAB.

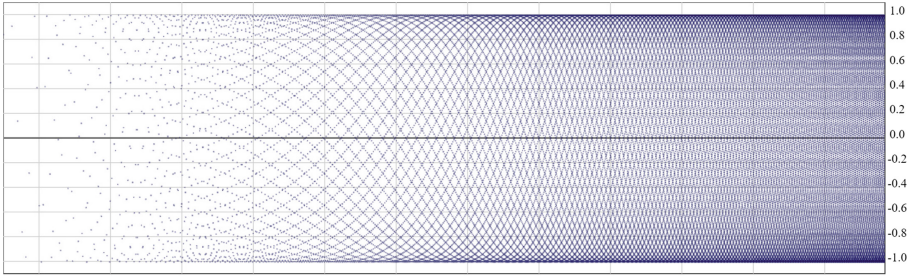


Fig. 4. The sequence $x_n = \sin n$ in logarithmic scale.

4.1 Examples of Skeleton Programs

The first program shows the sequence $x_n = \sin(n)$ in logarithmic scale. The image shows the first 50000 elements of the sequence. The rendering time was 1232 ms.

The image (see Fig. 4) is an example of a sequence having the closed interval $[-1, 1]$ as a set of subsequential limits. In our opinion, the image can help to see this fact clearly.

The next object is a slightly modified Thomae’s function in the open interval $(0, 1)$ also known as Riemann function or popcorn function (see Fig. 5).

$$R(x) = \begin{cases} \frac{(-1)^n}{n}, & x = \frac{m}{n} \in \mathbf{Q}, \\ 0, & x \in \mathbf{R} \setminus \mathbf{Q}. \end{cases} \quad (1)$$

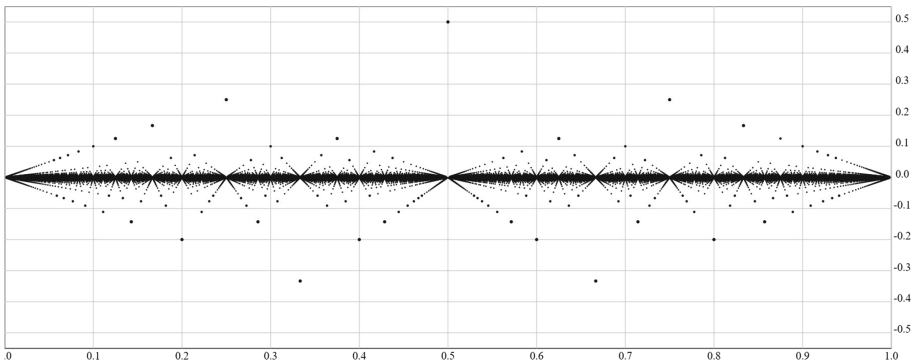


Fig. 5. Thomae’s function in the open interval $(0, 1)$.

The Fig. 5 shows that the function is continuous in every irrational point and discontinuous in every rational point. Also it has a supremum and infimum for every interval of the real axis.

A modified Tomae's function is:

$$\tilde{R}(x) = \begin{cases} n, & x = \frac{m}{n} \in \mathbf{Q}, \\ 0, & x \in \mathbf{R} \setminus \mathbf{Q}. \end{cases} \quad (2)$$

The new function (see Fig. 6) is notable because it is finite but not bounded on any numeric interval. Even a geometrically intuitive person can hardly imagine the appearance of the function. The figure shows all the points in the interval $(0, 1)$ with the denominator not exceeding 1000. The rendering time was 5005 ms.

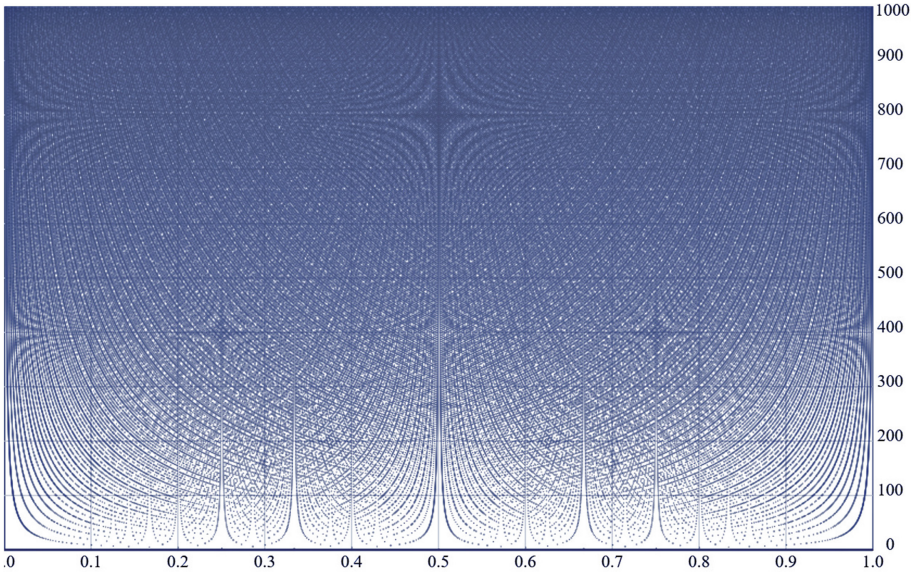


Fig. 6. The modified Tomae's function in $(0, 1)$.

The last figure (see Fig. 7) was captured in the program demonstrating a property of differentiable functions to be linear close to differentiable points. The figure also shows the control elements (dropdowns and buttons) that a lecturer or a student can use to see the property in an animation.

Unfortunately, the animations cannot be shown in this article. You can find some of the animations on a web site «A Programmer Library». For example, our work «9 GIFs on numerical sequences» [40, 41].

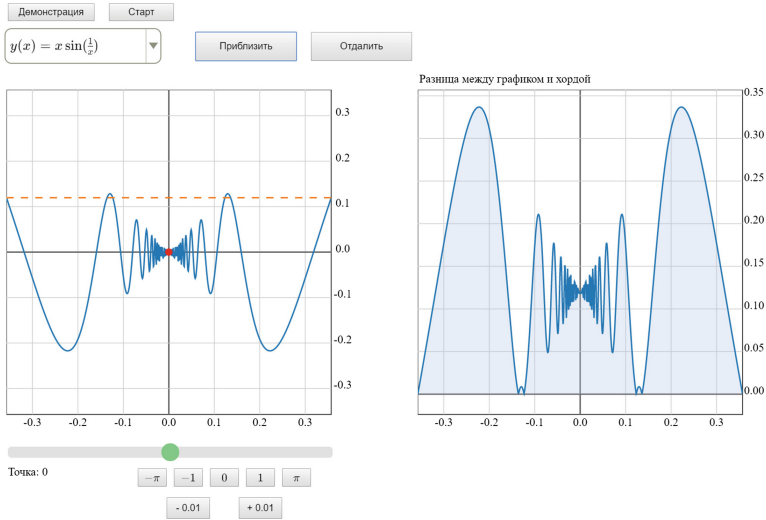


Fig. 7. The function $y = x \cdot \sin \frac{1}{x}$ is not differentiable at $x = 0$.

4.2 Grafar Examples

The figure below shows a volume bounded by the following surfaces:

$$\begin{aligned}
 V &= \{(x, y, z) \mid x \geq |z| \\
 &\quad x = z^2 + x^2, \\
 &\quad 2x = z^2 + x^2, \\
 &\quad 0 \leq y \leq 3 - z^2 - x^2\}
 \end{aligned}$$

The left picture of the Fig. 8 shows the domain and the bounding surfaces. The right picture is a projection of the cut orthogonal to the axis Oy to the xOz plane. The program allows to move the cut, the projection. The bounding lines change in real time.

The next example visualizes the method of Lagrange multipliers. The problem is to find a local maximum of a function $f(x, y) = 2x + 3y$ subject to a condition $x^2 + y^2 = 1$. The function describes a plane and the condition is a cylinder.

The Lagrange functions for this problem are paraboloids of revolution (depending on the multiplier's sign upward- or downward-facing). They have unconditional minimum and maximum at the points of the conditional extremum of the given problem (Fig. 9).

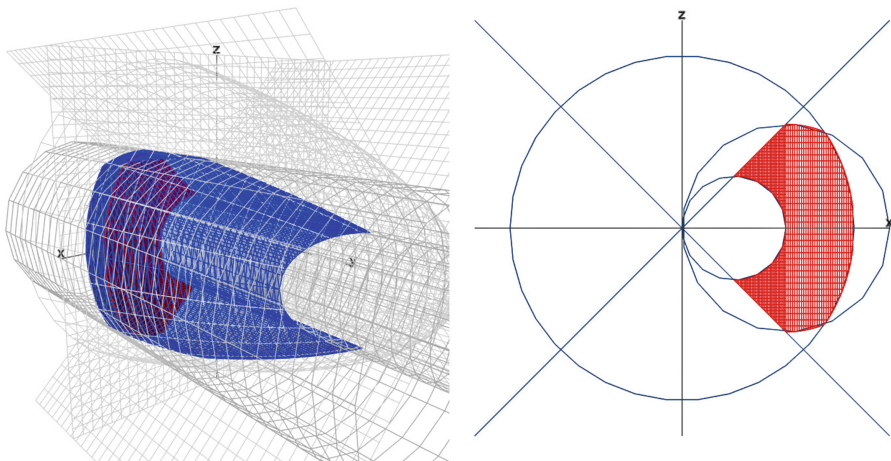


Fig. 8. An illustration on triple integrals

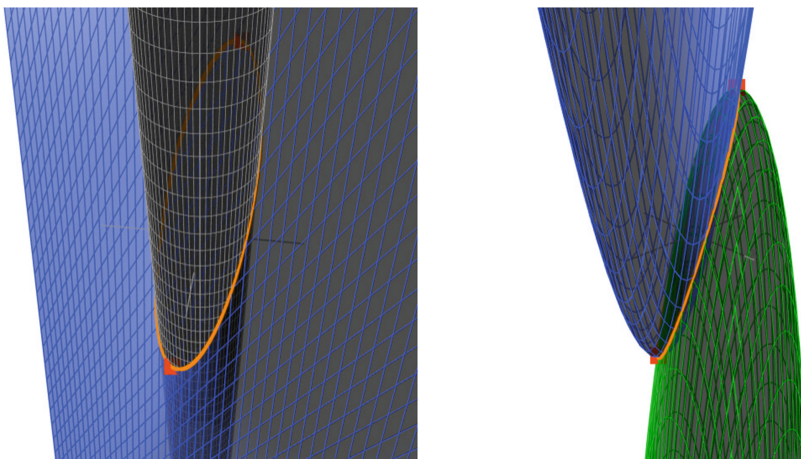


Fig. 9. An illustration on Lagrange multiplier method

The next image illustrates a topic from analytic geometry.

The Fig. 10 shows a one-sheet hyperboloid

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} = 1.$$

The program allows to change all the three parameters of the surface in a real time – a , b , c . Students can see how these variables affect the surface.

The last figure compares how the surface looks in

$$\begin{aligned} x^2 + y^2 + z^2 = a^2, \quad x^2 + y^2 + z^2 = b^2, \quad x^2 + y^2 = z^2, \\ x \geq 0, \quad 0 < a < b, \end{aligned} \quad (3)$$

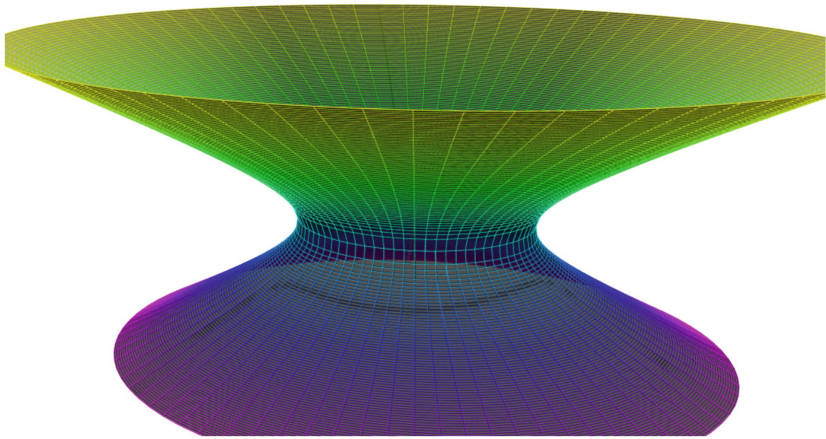


Fig. 10. A one-sheet hyperboloid

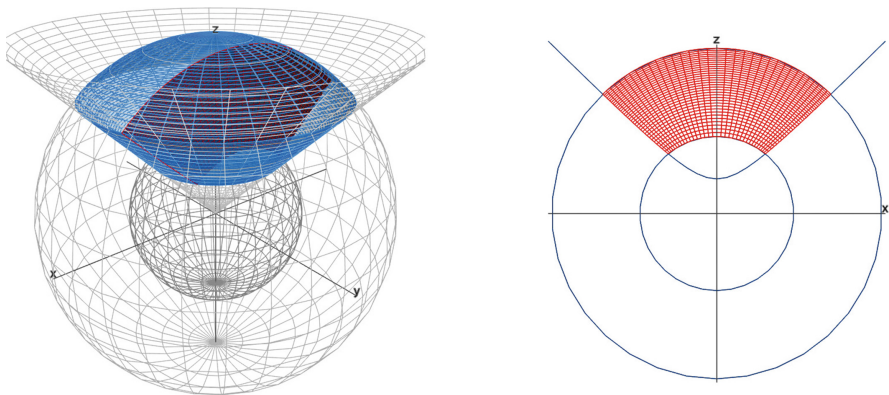


Fig. 11. Scope (3), depicted in the Grafar library.

Grafar (Fig. 11) and in The MathWorks Matlab, [16], (Fig. 12).

As the image beauty is a rather subjective concept, we note that in our project we could not use Matlab system for the reasons described in the part Sect. 3.2. First of all, because Matlab does not work in the browser.

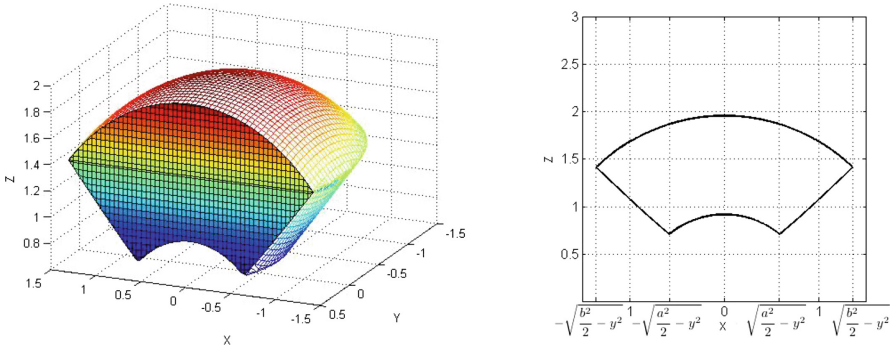


Fig. 12. Scope (3), depicted in the MathWorks Matlab.

Acknowledgements. Authors should note that a very large number of people worked on the project described in this article. Basically these were A. Nikitin’s students, who wrote their course works and theses on this topic. We express our deep gratitude to all of them. We would like to single out among them people who participated in the creation of the illustrations involved in this article: N. Kondrashov, A. Korytova, A. Anokhin, M. Gritsaev, I. Kushnir, A. Shemendyuk and M. Lukashova.

References

1. Eick, S.G., Wills, G.J.: High interaction graphics. *Eur. J. Oper. Res.* **81**(3), 445–459 (1995)
2. de Guzman, M.: The role of visualization in the teaching and learning of mathematical analysis (2002)
3. Hilbert, D., Cohn-Vossen, S.: *Geometry and the Imagination*, vol. 87. American Mathematical Society (1999)
4. Karadag, Z.: Improving online mathematical thinking. In: 11th International Congress on Mathematical Thinking in Elementary and Advanced Mathematics. *Educational Studies in Mathematics*, vol. 38, pp. 111–113 (2011)
5. Polya, G.: *How to Solve It: A New Aspect of Mathematical Method*. Princeton University Press, Princeton (2014)
6. Tufte, E.R., Graves-Morris, P.R.: *The Visual Display of Quantitative Information*, vol. 2. Graphics Press, Cheshire (1983)
7. Uspenskiy, V.A.: *Appologiya matematiki (sbornik statej)*. ANF, Moscow (2017). (in Russian)
8. Vickers, P., Faith, J., Rossiter, N.: Understanding visualization: a formal approach using category theory and semiotics. *IEEE Trans. Vis. Comput. Graph.* **19**(6), 1048–1061 (2013)
9. Ziemkiewicz, C., Kosara, R.: Understanding information visualization in the context of visual communication. Technical report, Technical Report CVCUNCC-07-08 (2007)
10. Zimmermann, W., Cunningham, S.: Editors’ introduction: what is mathematical visualization. In: *Visualization Teaching Learning Mathematics*, pp. 1–7 (1991)
11. Arnold, V.I.: *Experimentalnaya matematika*. FAZIS, Moscow (2005). (in Russian)

12. MIT Open Course Ware. <https://ocw.mit.edu/>
13. The d'Arbeloff Interactive Mathematics Project. <http://web.mit.edu/edtechfair/projects/interactive-math.html>
14. Wolfram Mathematica: Modern Technical Computing. <https://www.wolfram.com/mathematica/>
15. Maple for STEM Education & Research - Maplesoft. <https://www.maplesoft.com/MapleEducation/>
16. MathWorks - MATLAB & Simulink. <https://www.mathworks.com/products/matlab.html>
17. OpenStax Access. The future of education. 1999–2018, Rice University. <https://openstax.org/subjects/math/>
18. R: The R Project for Statistical Computing. <https://www.r-project.org/>
19. GNU Octave. <https://www.gnu.org/software/octave/>
20. The Julia Language. <https://julialang.org/>
21. SciPy.org. <https://www.scipy.org/>
22. Matplotlib: Python plotting. <https://matplotlib.org/index.html>
23. D3.js - Data-Driven Documents. <https://d3js.org/>
24. GraphyCalc - 3D Graphing Calculator. <http://www.graphycalc.com/>
25. Wolfram|Alpha: Computational Intelligence. <http://www.wolframalpha.com/>
26. Michael Mikowski, Josh Powell - Single Page Web Applications: JavaScript end-to-end (2013)
27. Khan/KaTeX: Fast math typesetting for the web. <https://github.com/Khan/KaTeX>
28. KaTeX and MathJax Comparison Demo. <https://www.intmath.com/cg5/katex-mathjax-comparison.php>
29. KaTeX - a new way to display math on the Web. <https://www.intmath.com/blog/mathematics/katex-a-new-way-to-display-math-on-the-web-9445>
30. Yushkevich, A.P.: The History of Mathematics in the Middle Ages. Fizmatlit, Moscow (1961). (in Russian)
31. Reactivity in Depth — Vue.js. <https://vuejs.org/v2/guide/reactivity.html>
32. Angular - The RxJS library. <https://angular.io/guide/rx-library>
33. Cycle.js. <https://cycle.js.org/#-functional-and-reactive>
34. Grafar 4 GitHub. <https://github.com/thoughtspile/Grafar/>
35. Scalable Vector Graphics (SVG) Full 1.2 Specification. <https://www.w3.org/TR/SVG12/>
36. HTML Canvas 2D Context. <https://www.w3.org/TR/2dcontext/>
37. WebGL 2.0 Specification. <https://www.khronos.org/registry/webgl/specs/latest/2.0/>
38. MaximPestryakov/visualmath-android. <https://github.com/MaximPestryakov/visualmath-android>
39. cherurg/skeleton: The drawing mathematical 2D visualizations. <https://github.com/cherurg/skeleton>
40. Programmer's library. <https://proglib.io/>
41. 9 gifs, which illustrate numerical sequences | Programmer's library (in Russian). <https://proglib.io/p/sequences/>
42. 9 gifs, clearly illustrating the notion of differentiability | Programmer's library (in Russian). <https://proglib.io/p/diff/>
43. RFC 6455 - The WebSocket Protocol. <https://tools.ietf.org/html/rfc6455>
44. WebRTC 1.0: Real-time Communication Between Browsers. <https://www.w3.org/TR/webrtc/>
45. Can I use... WebRTC. <https://caniuse.com/#search=webrtc>
46. VisualMath.ru: Platform for the blended learning. <http://www.visualmath.ru/>