Nucleus Decomposition and Complex Function Visualization

Will Bolden

We explore two very different areas of visualization. The first relies on the nucleus decomposition operation, which turns a graph into a tree of subgraphs of ever increasing density. In visualizing this tree of subgraphs, along with the intersections between subgraphs, we hope to quickly highlight regions of interest in large graphs and allow users to explore them. Our second focus was visualization of functions of a complex variable in a 2D and 3D context, and make these potentially difficult to grasp concepts more accessible. Our efforts here were very successful, however as this was a secondary project we spend less time analyzing this than we do on our first topic.

1 Nucleus Decomposition Visualization

1.1 Introduction

Several real-world datasets, from citation information to social networks, can be represented as graphs. Although each dataset might represent very different topics, a graphical representation opens the possibility of investigating them with standard graph-theoretic techniques. In this initial step in our work we will focus primarily on "citation networks", graphs where papers are represented by vertices and citations are represented by edges. Although this might seem like a very narrow focus, we believe that that techniques developed for use on citation networks may prove very useful in a wide variety of real-world networks.

In large graphs, which can have well over a million vertices and an even larger number of edges, it can be difficult to pick out regions of interest to examine. However most real-world graphs are sparse, meaning that the number of edges in the graph is much less than the number of possible edges in the graph, which can be used to our advantage. Given this generalization it is reasonable to say that a dense subgraph, which has near the maximum number of possible edges, is a potentially interesting part of the data and a good candidate to focus our visualization resources on. Finding these dense subgraphs is often easier said than done, and in fact finding the densest k subgraphs of a graph is an NP-hard problem in general. As such, approximation methods are needed.

Nucleus decomposition is one such approximation which, at a high level, transforms a graph into a tree of subgraphs. The root node of the tree is the least dense, and represents the entire graph, while as the depth increases the children represent denser subgraphs of their parents. In the context of citation networks, we decided that this approach was potentially very advantageous in the sense that this tree could potentially clearly show related fields and subfields in an area of research. It is important to note that vertices in the original graph can be "duplicated" when creating this tree, in the sense that two subgraphs can contain the same vertex despite not being ancestors or descendants of one another. To provide a simple example, if a graph contained a paper A, it is possible for any number of its subgraphs (children in the nucleus decomposition tree) to also contain the paper A. Intuitively places where these "intersections" between subgraphs occur could also be interesting, possibly representing an important paper that linked two fields or a paper from one field that spawned an entirely new field, so examining these regions also became a topic of interest in this project.

The original publication of this method included some simple visualizations, as seen below, but a more interactive and easy-to-interpret visualization was desired. This was the main focus of our work.

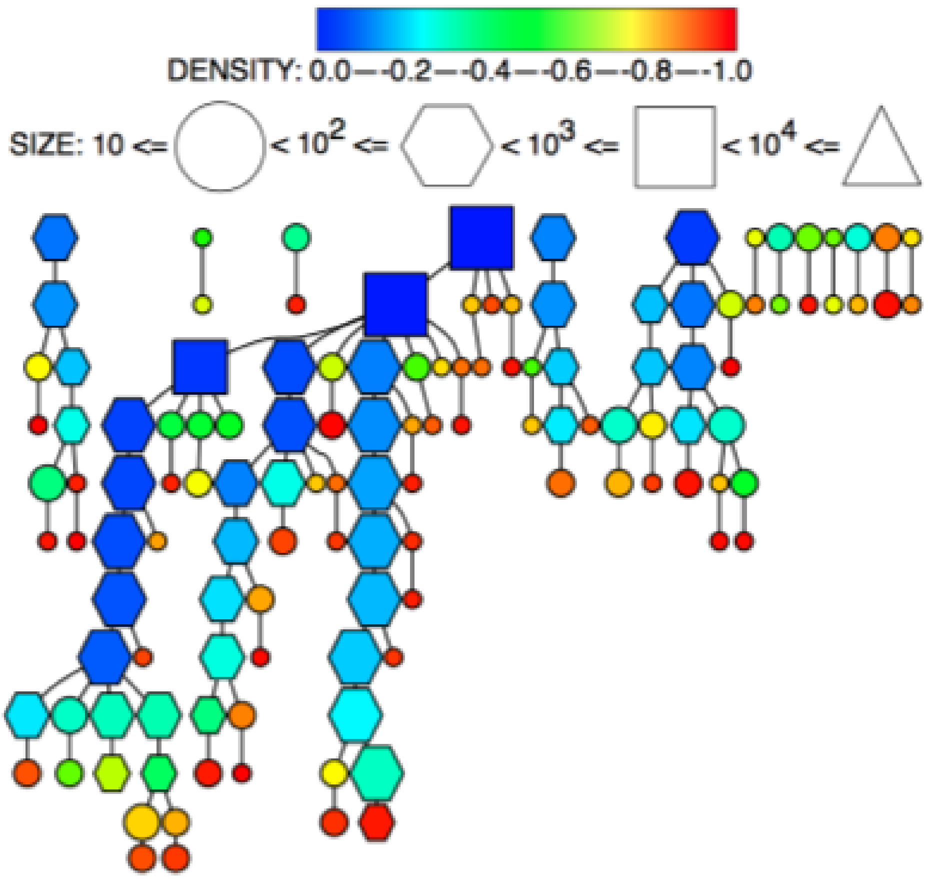

DENSITY: 0.0—-0.2—-0.4—-0.6—-0.8—-1.0
SIZE: 10 <= < 10^2 <= < 10^3 <= < 10^4 <=

Figure 1: Visualization shown in the original paper

Note that in the above image of a forest can be made into a tree, as previously specified, by connecting the root node of each tree in the forest to a new node representing the original graph.

1.2 Related Works

As we did not work on the implementation of Nucleus Decomposition algorithm ourselves, we will focus purely on the related visualization techniques in this section.

As we primarily focused on citation networks, existing software used to visualize citation networks was of primary interest to us. We found a few instances of this kind software, notably CitNetExplorer and VOSviewer, which are both free tools released by Leiden University in the Netherlands. Their work was discussed in depth in a paper they published, however all of their techniques focus on operating on the original citation network, rather than a tree of subgraphs [1]. As it is clear that their work did not employ a tree-based decomposition of the original graph like we did, so from a user perspective at least our technique provides a very different, and potentially more useful, visualization of the data.

1.3 Technical Detail

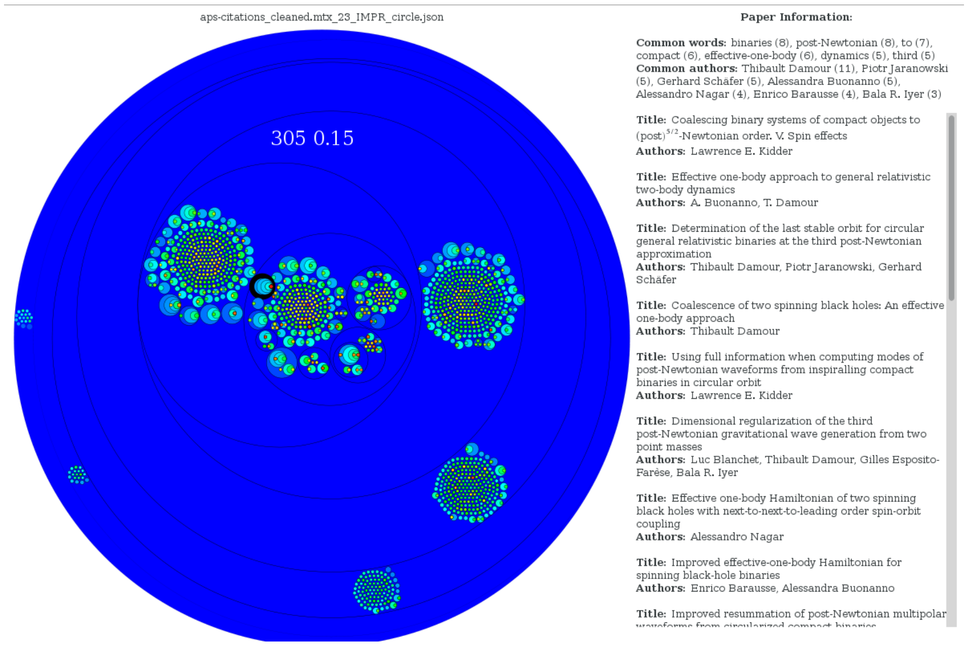
The primary tool we used in creating our visualization was the JavaScript library d3, which provides a variety of useful features for quickly visualizing data [2]. Although d3 was the backbone of our vis, it was sometimes cumbersome and slow to work with JavaScript-based methods, so we occasionally employed Gephi, a free graph visualization tool, to examine some parts of the data [3]. We also used python in conjunction with networkx, a graph library, to obtain some statistics about our graphs. [4]

Because of the amount of data provided, some 500,000 papers, and ease of access we focused on visualizing citations from the many Physical Review journals provided by the American Physical Society. Although we were somewhat limited by the fact that we all had limited experience in physics, we figured quantity was somewhat important if we want to extend out techniques to other large networks.

The specific code to perform a nucleus decomposition was never provided to us, and instead we were given the output of a nucleus decomposition that had been run on the aforementioned citation network. The result, a tree as mentioned, took the form of a JSON file which contained information about all of the papers in each node of the tree (where each node represents a subgraph). Some information was missing from this representation, notably we did not have any information on the papers which comprised a subgraph for subgraphs with a density less than 0.01 and initially did not have any information about the actual structure of the subgraph, though we were later provided this.

The initial iteration of the project simply represented this tree via circle packing in d3, which didn't provide much more information than the densities and number of vertices per subgraph at different levels of the tree.

Improving upon this we added the ability to pan and zoom around the visualization, along with the ability for users to select a node of the tree to load information about all of the papers contained within it. To help us identify common topics and authors we also provided a list of the most common keywords and authors in the subgraph, sorted for greatest to least. Together these strategies were relatively effective in allowing us to quickly look around the tree and identify different subcomponents.



Paper Information:

Common words: binaries (8), post-Newtonian (8), to (7), compact (6), effective-one-body (6), dynamics (5), third (5)
Common authors: Thibault Damour (11), Piotr Jaranowski (5), Gerhard Schäfer (5), Alessandra Buonanno (5), Alessandro Nagar (4), Enrico Barausse (4), Bala R. Iyer (3)

Title: Coalescing binary systems of compact objects to $(\text{post})^{5/2}$-Newtonian order. V. Spin effects
Authors: Lawrence E. Kidder

Title: Effective one-body approach to general relativistic two-body dynamics
Authors: A. Buonanno, T. Damour

Title: Determination of the last stable orbit for circular general relativistic binaries at the third post-Newtonian approximation
Authors: Thibault Damour, Piotr Jaranowski, Gerhard Schäfer

Title: Coalescence of two spinning black holes: An effective one-body approach
Authors: Thibault Damour

Title: Using full information when computing modes of post-Newtonian waveforms from inspiralling compact binaries in circular orbit
Authors: Lawrence E. Kidder

Title: Dimensional regularization of the third post-Newtonian gravitational wave generation from two point masses
Authors: Luc Blanchet, Thibault Damour, Gilles Esposito-Farèse, Bala R. Iyer

Title: Effective one-body Hamiltonian of two spinning black holes with next-to-next-to-leading order spin-orbit coupling
Authors: Alessandro Nagar

Title: Improved effective-one-body Hamiltonian for spinning black-hole binaries
Authors: Enrico Barausse, Alessandra Buonanno

Title: Improved resummation of post-Newtonian multipolar waveforms from circularized compact binaries

Figure 2: Our early progress on the visualization. 305 is the number of vertices in the selected subgraph (near center), 0.15 is the density.

We also made some edits to the underlying d3 circle packing code in order to provide a more spread out look, as we feared that the default clumped look provided by d3 would provide users with the illusion of correlation where there was none. In this iteration we also allowed users to filter by subgraph size, which cleaned up the look of our image in general and mode the program run much more quickly. Although we added many more features, this general look did not change for the remainder of the project.

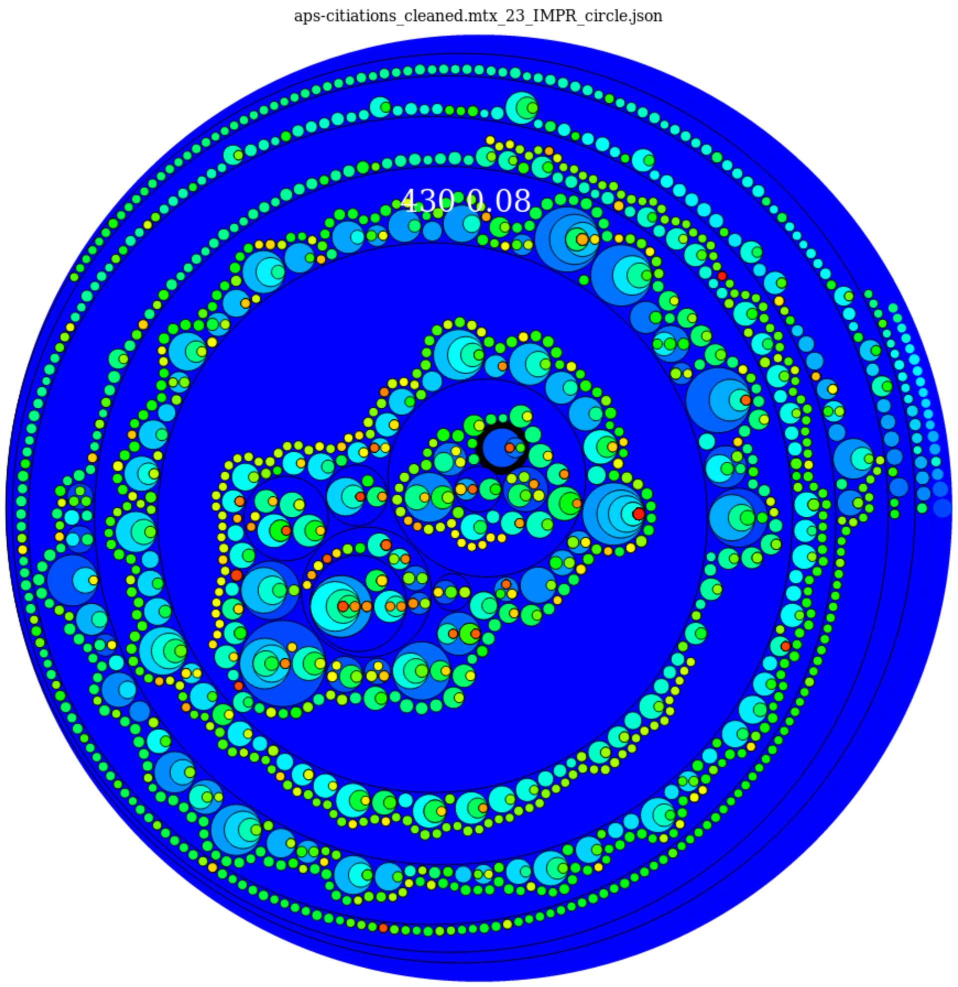


Figure 3: A more spread-out view

With this basic component of the visualization done we decided to take a closer look at intersections among subgraphs, which in the end consumed the remainder of our time on this project. We decided to only record the intersections of maximum depth, meaning that a node would not be marked as intersecting with a paper if a descendant of it also intersected with the paper. To do this we used a simple recursive algorithm, which updated an array where each index represented a paper number. At each step of the recursive process if a paper appeared in a child the child was added to that paper's index in the array, and the parent was likewise removed, such that when the algorithm completed the array would have exactly the information desired.

From this, in addition to some statistics which will be further discussed in the results section, we obtained the information required to work on the last two components of the project we had time to work on this quarter. The first of these components involved taking the intersection data and using it to generate graphs which show what other subgraphs you can reach by traversing intersections as if they formed edges.

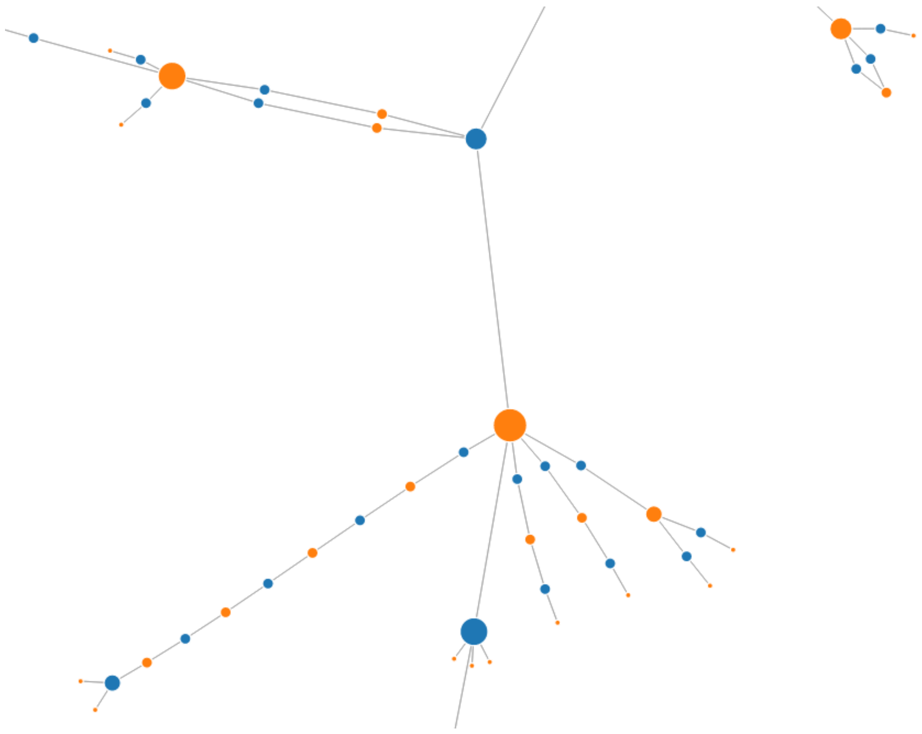


Figure 4: A fragment of an intersection graph, orange nodes represent subgraphs while blue nodes represent papers

The second, perhaps more obvious, component this enabled us to implement was the ability to highlight click on a circle, while intersection mode was enable, in order to highlight all nodes which intersected with the given node. Although we feared that this method might appear to messy, and indeed if we highlight all possible intersections stored in the papers array it does, when limiting the papers to those intersecting with a selection the results are very manageable. This allows users to quickly see different intersecting sets and look at them in order gain better understanding of their relations.

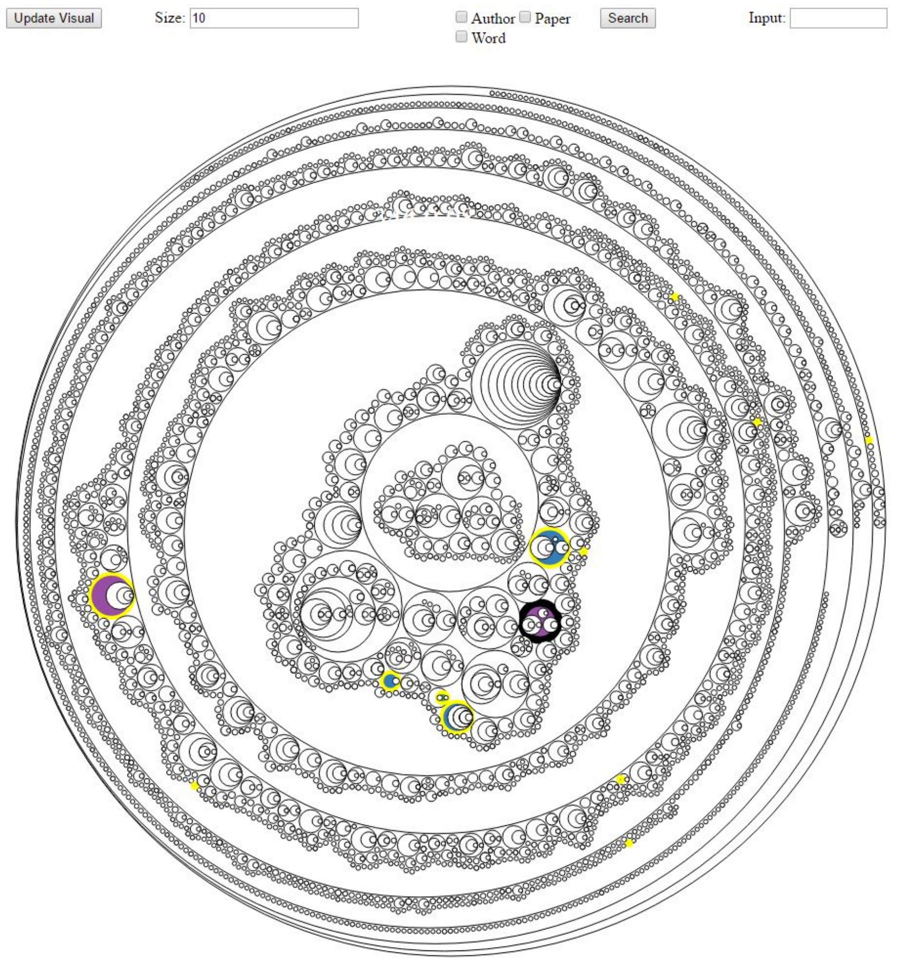
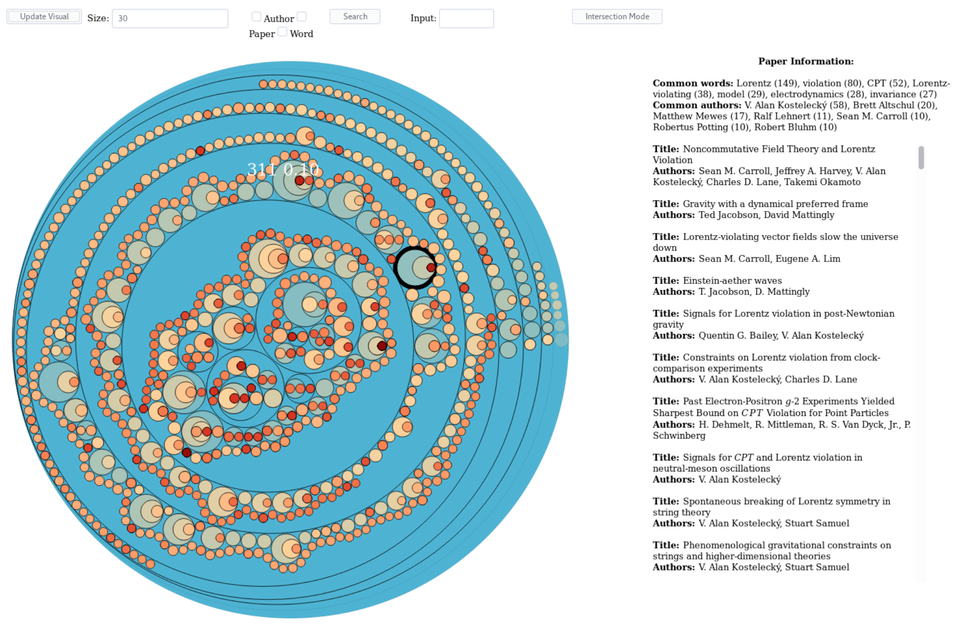


Figure 5: Highlighting subgraphs which intersect with the selected subgraph

1.4 Results

The final result of our work was a fully functional tool for visualizing our citation data, complete with the ability to search for papers and authors, and quickly access the paper and intersection information related to each node. In the image below we also show a quick test of a new color scheme, which we hoped would be easier to look at than the very saturated color scheme we previously used.



Paper Information:

Common words: Lorentz (149), violation (80), CPT (52), Lorentz-violating (38), model (29), electrodynamics (28), invariance (27)
Common authors: V. Alan Kostelecký (58), Brett Altschul (20), Matthew Mewes (17), Ralf Lehnert (11), Sean M. Carroll (10), Robertus Potting (10), Robert Bluhm (10)

Title: Noncommutative Field Theory and Lorentz Violation
Authors: Sean M. Carroll, Jeffrey A. Harvey, V. Alan Kostelecký, Charles D. Lane, Takemi Okamoto

Title: Gravity with a dynamical preferred frame
Authors: Ted Jacobson, David Mattingly

Title: Lorentz-violating vector fields slow the universe down
Authors: Sean M. Carroll, Eugene A. Lim

Title: Einstein-aether waves
Authors: T. Jacobson, D. Mattingly

Title: Signals for Lorentz violation in post-Newtonian gravity
Authors: Quentin G. Bailey, V. Alan Kostelecký

Title: Constraints on Lorentz violation from clock-comparison experiments
Authors: V. Alan Kostelecký, Charles D. Lane

Title: Past Electron-Positron g-2 Experiments Yielded Sharpest Bound on CPT Violation for Point Particles
Authors: H. Dehmelt, R. Mittleman, R. S. Van Dyck, Jr., P. Schwinberg

Title: Signals for CPT and Lorentz violation in neutral-meson oscillations
Authors: V. Alan Kostelecký

Title: Spontaneous breaking of Lorentz symmetry in string theory
Authors: V. Alan Kostelecký, Stuart Samuel

Title: Phenomenological gravitational constraints on strings and higher-dimensional theories
Authors: V. Alan Kostelecký, Stuart Samuel

Figure 6: Our current progress

We also collected several statistics about intersections in the graph, noting that while 8132 subgraphs participated in intersections of size 2, only 738 subgraphs participated in intersections of size 3. This rapid decay continues to the point where only 11 subgraphs participate in intersections of size greater than 9. When creating graphs out of the intersections we noticed a somewhat similar phenomenon. Most, 1577, of the intersection graphs formed had 3 vertices, two vertices representing subgraphs and one vertex representing their linking paper, while there was one intersection graph which contained 6192 vertices. Aside from this outlier, the largest number of vertices was only 216. All other intersection graphs involved appeared to be planar, while the 6192-vertex graph appeared to be non-planar. It should be noted that we only know these statistics about planarity because of our attempts to visualize the graphs, and as determining the crossing number of a graph is an NP-hard problem we have not formally checked whether the 6192-vertex graph is planar.

Examining several random intersections we found that there usually was some correlation between the keywords in intersecting sets, though given our lack of expertise in physics it is still difficult for us to say exactly how significant this correlation is. Allowing a physics professor to take a look at our data would be a good step to take in the future, and should help us gauge the effectiveness of our technique.

1.5 Conclusion

While our efforts were successful, perhaps the main obstacle which held back our progress was our inability to pinpoint what, specifically, was useful information to get out nucleus decomposition tree. Because of this most of our work was focused more on giving us tools, and reasonably effective ones at that, to analyze our graph and display various statistics about it. While this is useful, it unfortunately has not brought us to our end goal of displaying truly useful information about citation networks, or large graphs in general, at a glance.

As we continue to work on the project, we hope to approach this goal, and along we way plan on conducting many additional tests. Notably we wish to see how the nucleus decomposition graph changes if its data is restricted to a period of time, and also how the topics of interest in each subgraph change over time. There is still more work to be done on the intersection problem, notably about how we can use intersections and their properties to help convey information to our users, if such a thing is possible at all. There is a significant amount of work ahead of us, but we will continue to make progress.

2 Complex Function Visualization

2.1 Introduction

The purpose of this project was to provide a fast and simple to use method of displaying functions of a complex variable, a variable of the form $z = x + yi$, where i is the imaginary unit. As this project was secondary to our work on nucleus decomposition visualization we will not go into as great of detail here, and merely provide a brief overview of the project.

2.2 Related Works

Visualizing complex functions has been a topic of discussion for centuries, and as such none of the concepts we implement here are truly original. The basic layout and color scheme used in the initial 2D representation were based almost entirely on a similar site created by David Bau [5]. The coloring scheme used by this site is based on a common technique known as domain coloring, which colors points based on their angle and magnitude on the complex plane. One feature of this site that we had not seen before, however, was the presence of a regular grid of points to emphasize how areas were stretched or squished, which can be very difficult to do when looking at seemingly continuous colors. This site, however, does all of its rendering slowly in Javascript and does not support many of the more advanced features we added to our work.

The 3D component of our project is based on an idea called a Riemann surface, which essentially allows us to display a deformed portion of the complex plane as a surface in 3D. In general complex functions would require 4 dimensions to present as they take a point $x + yi$ and output a point $z + wi$, for a total of 4 components that would need to be plotted. In our work we quickly generate a Riemann surface by creating this 4D point vector $[x, y, z, w]$ and projecting down into 3D.

2.3 Detail and Results

The overall structure of the program is fairly simple. The core graphics engine is a collection of GLSL shaders, many of them filled with placeholder tags such as \$userinput\$ or \$userfunctions\$. These tags are used to indicate where code can be inserted into the shaders after the user enters the expression they wish to compute. The user inputted expressions are not valid GLSL code, and instead have to be put through a parser to compile the fairly easy-to-write user input into the much more complicated sequence of function definitions and calls needed to produce the desired result from GLSL. We perform this process of parsing user input, splicing it into a shader, and then recompiling the shader each time the user changes their input as this allows GLSL to generate results extremely quickly, especially on computers with a good GPU.

Although some code depends entirely upon user input, and thus cannot be added to the shader in advance of said input, the majority of supporting code can remain statically in the shaders. The majority of this code consists of function definitions, as although GLSL has builtin math functions for real variables some additional work is required to define these functions for complex variables. In addition to math functions, the core code graphics code does not chance based on user input. One piece of code that remains in the shader is the code for numerically computing the inverse of a function, which can often be accomplished using Newton's method.

A new feature we added was support for two additional definitions of i. In regular complex numbers $i^2 = -1$, but we added support for $i^2 = 1$ (where i is not 1, the hyperbolic numbers) and $i^2 = 0$ (where i is not 0, the dual numbers). To avoid requiring a costly numerical computation for any of these alternative functions we used taylor expansions and employed well-known identities in order to find closed-form expressions for all of them, which we then implemented.

Below we list a small number of examples of the visualization's capabilities:

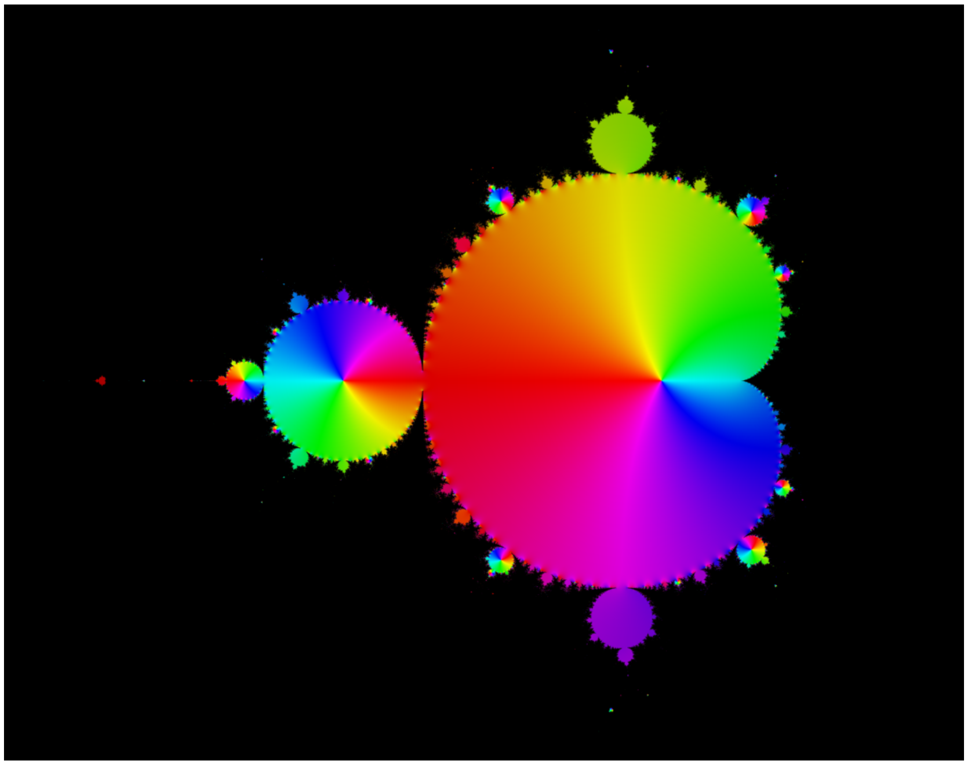


Figure 7: The Mandelbrot set

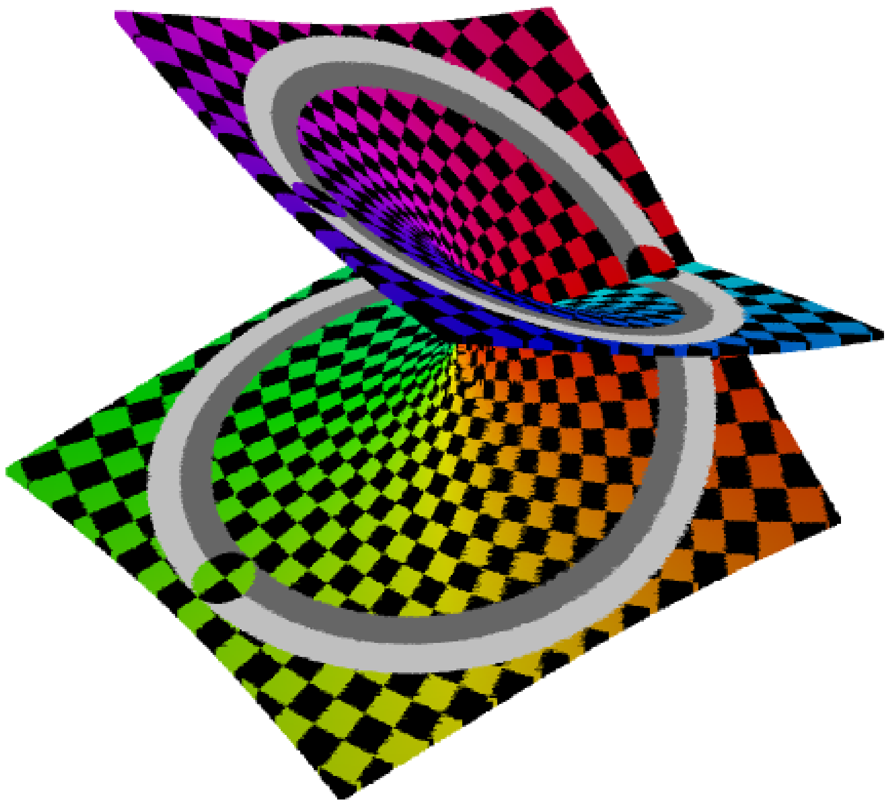


Figure 8: The square root function, plotted in 3D

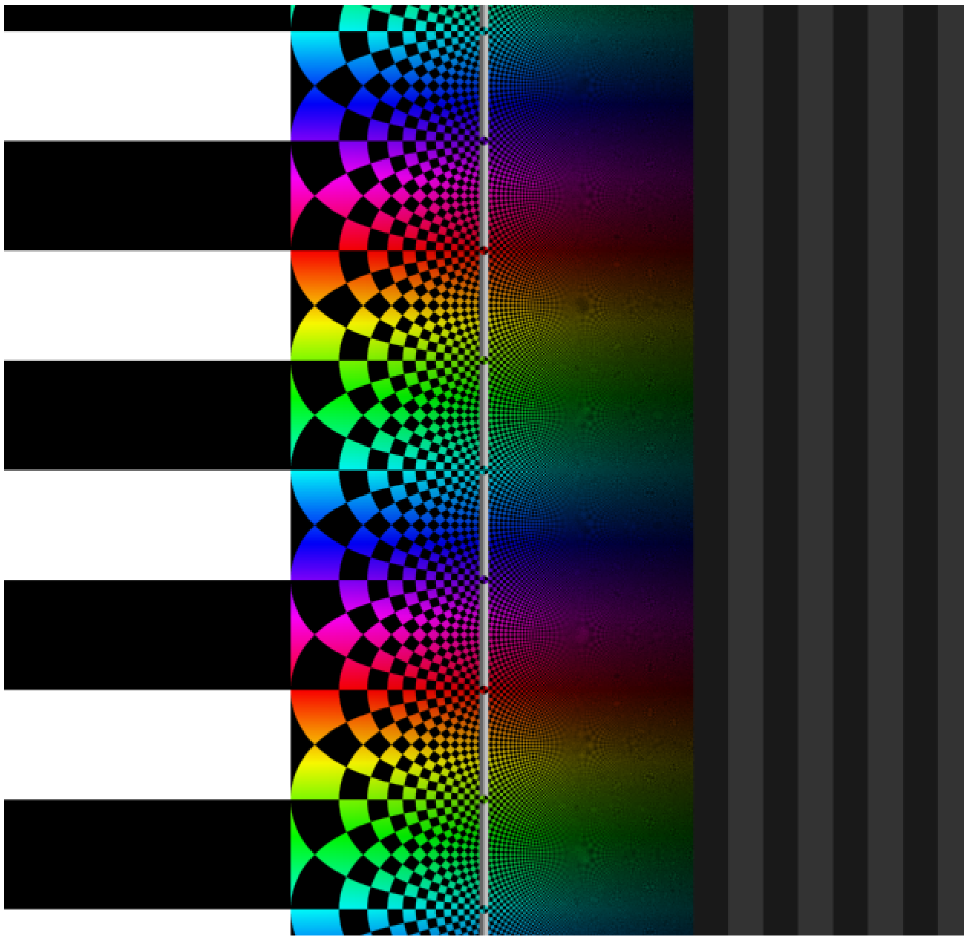


Figure 9: The exponential function

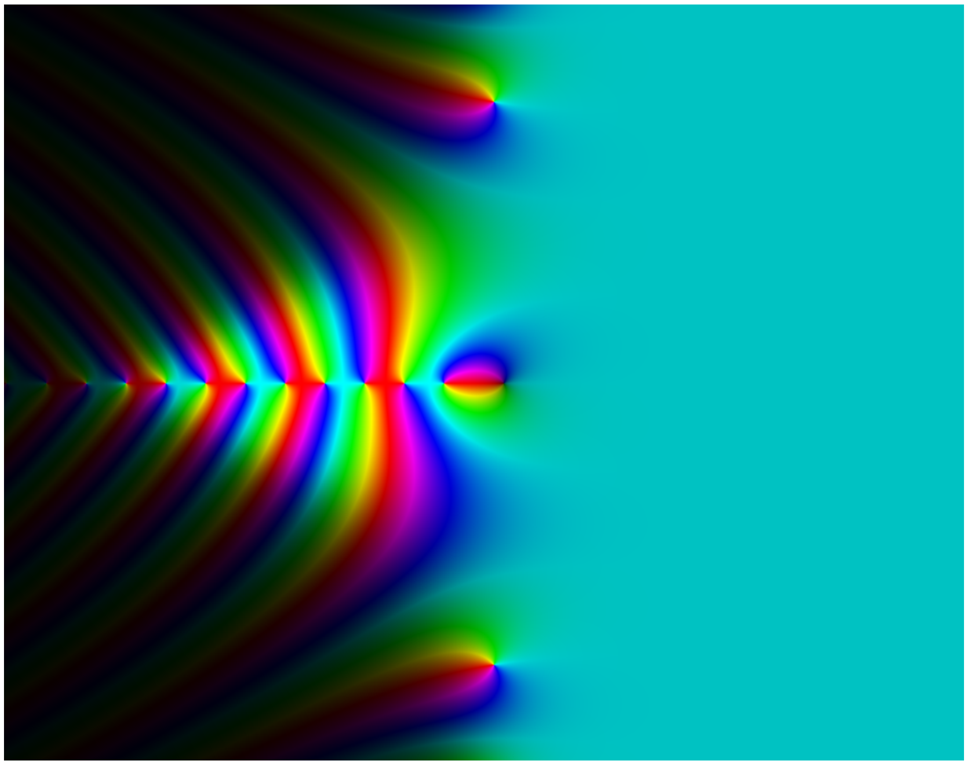


Figure 10: The Riemann zeta function

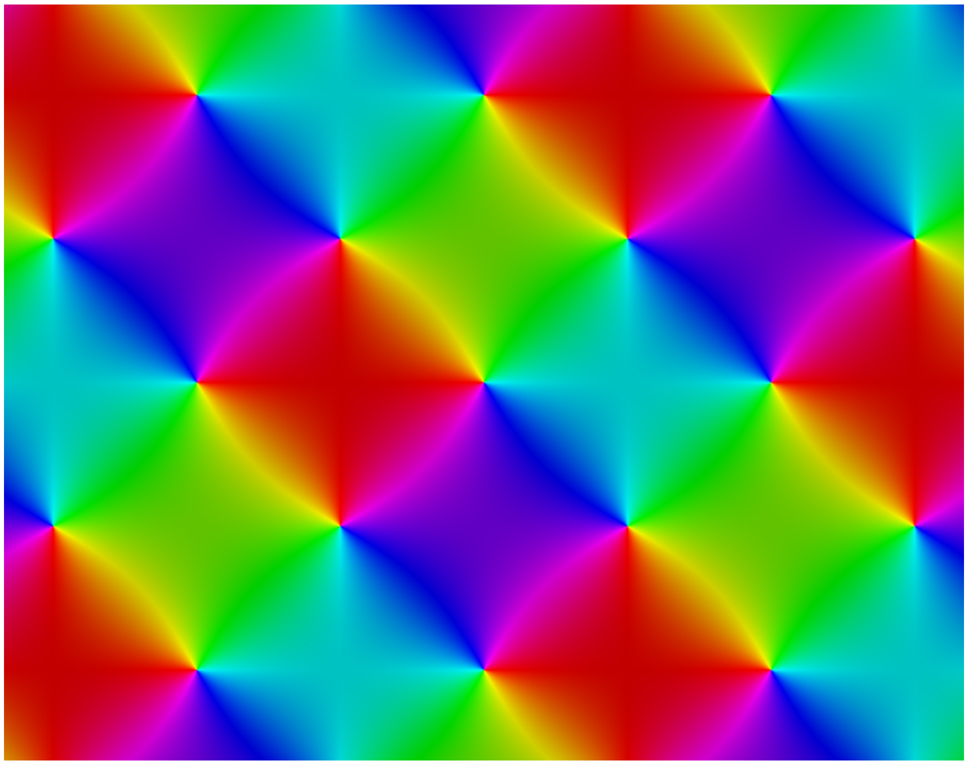


Figure 11: The sin function in the hyperbolic plane ($i^2 = 1$)

2.4 Conclusion

Although there is always room for improvement, all the core features of the complex function visualization are implemented. Our visualization offers many different modes to the user, and is quite configurable within each of these modes, so for most users our tool should provide sufficient flexibility to be useful.

References

[1] Van Eck, N.J., & Waltman, L. (2014). CitNetExplorer: A new software tool for analyzing and visualizing citation networks. Journal of Informetrics, 8(4), 802-823.

[2] Mike Bostock et al, d3: Data Driven Development, available from https://d3js.org/

[3] Bastian M., Heymann S., Jacomy M. (2009). Gephi: an open source software for exploring and manipulating networks. International AAAI Conference on Weblogs and Social Media.

[4] Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart, ”Exploring network structure, dynamics, and function using NetworkX”, in Proceedings of the 7th Python in Science Conference (SciPy2008), Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11–15, Aug 2008

[5] David Bau, Complex Function Viewer, available from http://davidbau.com/conformal/