High Dimensional Event Exploration Over Multiple Simulations

Steven Scott
Department of Computer Science
Utah State University
Logan, Utah
stevenderonscott@gmail.com

Jaxon Willard
Department of Computer Science
Utah State University
Logan, Utah
jaxon.willard@aggiemail.usu.edu

John Edwards
Department of Computer Science
Utah State University
Logan, Utah
john.edwards@usu.edu

***Abstract*—We introduce a technique for user-interactive discovery of families of input parameter sets for simulation based on visualization of topological evolution of discrete events through large numbers of simulations. Discovering how events behave across runs of a simulation has applications in financial market analysis, military simulations, physical mechanics, and other settings. Our approach is to use established methods to produce a linearized tour through parameter space of arbitrary dimension and visualize events of interest in two dimensions, where the first dimension is the tour ordering and the second dimension is usually time. This paper presents our approach and gives examples in the context of simulations of magnet dynamics.**

I. Introduction

In recent years, simulations have been growing in number of attributes, or dimensions, number of simulation runs, resolution of output, and complexity of the output. Simulation with traditional, scalar output might predict, e.g., temperature of discrete voxels in a three-dimensional space, but other simulations produce output that includes discrete events. For example, a military simulation might predict an event, like detection of a threat, or a neurophysiology simulation might predict axon depolarization. Other applications exist in stock market trading algorithms and physics simulations. Many approaches exist for analyzing and visualizing data from scalar functions, but we know of no existing approaches for exploring discrete event families across simulations.

In this work, we explore high-dimensional event data across simulations. In particular, we seek to create a visualization system where a user could explore how changes to inputs cause event births, deaths, bifurcations, and merges. Using our previous example, suppose a military analyst is considering how a series of missile strikes will affect key assets. Pursuant to this, they run simulations of missiles launched toward various cities and explore which simulations resulted in the missiles being detected by friendly assets. We cast the problem of exploring this data in terms of visualizing the simulation runs in series and discovering problem simulations where threats were not detected. A second example relates to financial markets: while stock market trading algorithms may use machine learning or some other technique to determine how much weight is given to different events in determining whether to buy or sell, it may be difficult for the analysts to explain why the algorithm chooses to buy or sell at a certain time. By tweaking the input weights slightly for each simulation of the algorithm, the analyst can gain a clearer understanding of which thresholds and events are most impactful on the output set.

Using the taxonomy given in [12], our contribution is two-fold. The first contribution is data transformational, using z-order linearization to provide a 1-dimensional ordering of simulations. The second contribution, given an ordering of simulations, is a visual mapping of events that allows the user to interactively discover topological features of the events across simulations. After discussing related work (Sec. II), we describe our linearization approach (Sec. III-A) followed by our visualization technique with examples (Sec. III-B). We conclude with implications of our work and future directions (Sec. IV).

II. Related work

Simulation data is typically analyzed in terms of scalar functions in the spatiotemporal domain and other temporally-driven domains across a single simulation. Dimension reduction techniques (e.g. PCA [11]) reduce the number of attributes to a quantity that is easier to visualize. Clustering (e.g. k-means clustering [1]) is an unsupervised learning technique that partitions samples into meaningful groups. Subspace clustering (e.g. [3]) clusters dimensions, often along with clustering of samples. Topological data analysis (TDA) [5, 12, 13, 16] allows users to analyze data using topological constructions that both provide compact descriptions of the data and reveal features. There is a body of work that compares features across simulations, such as comparing clusterings [9] and topological merge trees [2], but none to our knowledge that directly compares discrete events.

Our goal is a level of abstraction away from typical methods. Rather than analyzing scalar or multi-dimensional, quantitative output functions, we are interested in families of discrete, temporal events across simulations. To our knowledge, no visualization technique of this type of data has been published. Approaches have been suggested for exploring and predicting single events such as extinction and reignition in a combustion engine [10] and successful binding of an antibody to a noxious molecule [14], but these approaches do not analyze the topological structure of multiple events across time across simulations.

III. Description of the visualization

We describe our visualization approach to allow users to explore event families in high-dimensional simulations. We demonstrate the techniques using the *MagPhyx* simulation software [8]. *MagPhyx* is a software package that simulates spatial interactions between pairs of spherical dipole magnets. In our examples here, we vary combinations of three input parameters, β, θ, and r [8].

A. Linearization

One of the key challenges in exploring event families across simulations is placing simulations in an order that allows the user to find patterns, points of interest, or divergences. In order to visualize event families in a meaningful way, we need a method of linearizing (i.e. reducing to a single-dimensional ordering) the simulations. For input sets following a grid pattern, the ubiquitous and intuitive row- and column-major orderings are simple but have poor spatial locality [4]. That is, the average distance in parameter space between two consecutive simulations is large compared to distances between consecutive points in other, better orderings. In order to traverse the data so as to minimize large jumps between simulations that are displayed as consecutive, we use z-ordering as constructed using a quadtree [15] . While this approach occasionally results in a large jump, it usually will place simulations that are close together in input space next to each other in the simulation display. We recognize that Hilbert curves have better spatial locality than z-ordering, but we choose the simpler z-ordering because of ease of construction.

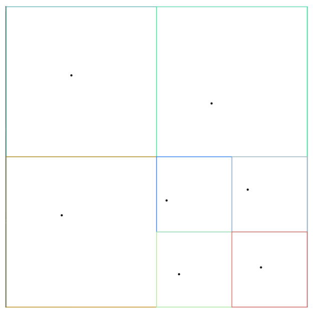
Quadtrees (Fig. 1a) are data structures commonly used in collision detection and shape modeling. The structure is recursive, with each non-leaf cell containing exactly four children. Leaf nodes contain exactly one or zero points. In our implementation, a cell will subdivide if it already contains a point and a new point is inserted into it.

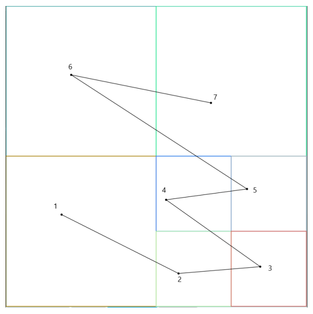
Z-ordering is a graph traversal technique used to order data in arbitrary dimensions into a one-dimensional ordering while preserving spatial locality. While a z-order traversal can be calculated by interleaving binary coordinates of the data, a depth-first traversal of a quadtree will produce the same output. In our implementation, the traversal (Fig. 1b) starts at the bottom-left cell that contains four sub-children and visits the bottom-left, bottom-right, top-left, and top-right sub-children. It then moves up a level recursively and repeats itself until every cell in the graph is visited.

This technique will produce a one-dimensional representation of all the points on the graph. The benefit of linearized simulations is that we can see the effects of many small changes in input parameters. This way we can identify events like births, deaths, merges, and bifurcations in multidimensional simulations.

B. Visualization

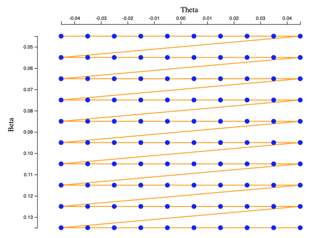
Our visualization includes three charts: a scatter plot of input configurations (Parameter Space chart), a scatter plot


(a)

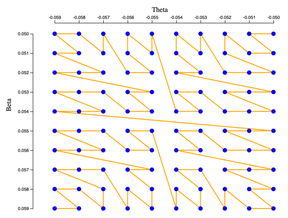


(b)

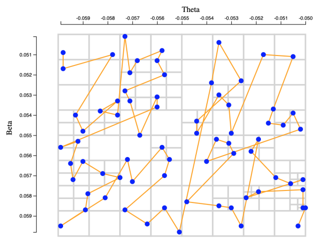
Fig. 1: (a) A quadtree subdivides until there are one or zero points in each cell. (b) Z-Order Traversal. The algorithm starts at the bottom-left leaf of every node and continues to the bottom-right, top-left, and top-right leaves until every point is visited.



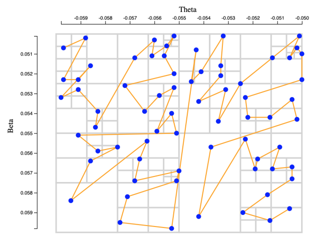
(a)



(b)



(c)



(d)

Fig. 2: These examples of a Parameter Space Chart show how the linearization of simulations is displayed to the user. (a) Row Major Order. (b) Z-Ordering. Note the decrease in large jumps from row-major order. (c) and (d) Z-Ordering on randomly sampled parameter space. In both cases, the data were produced by randomly varying θ and β from -0.05 through -0.06 and 0.05 through 0.06, respectively. (c) shows a potential limitation of z-ordering, as several large jumps between consecutive simulations can be seen. (d) was created through the same process, but has fewer large jumps between consecutive simulations.

of all events, and timeline chart of events. We describe each chart in detail.

1) Parameter Space chart (Fig. 2) - This first visualization accomplishes several purposes. First, it allows the user to see the ordering of simulations used in the Event Timeline chart. Second, in our *MagPhyx* setting, clicking any point in the Parameter Space chart will replay that simulation, allowing the user to see detail on the simulation behavior. Finally, hovering over a point will highlight all events in the other two charts that were produced by that particular set of parameters, linking the charts together.
2) All Events chart (Fig. 3) - This visualization charts all output events from all simulations, charted on a scatter plot of theta and beta. This allows the user to see clusters of events or correlations in the data. Hovering over any of these points will highlight the point in the Parameter Space chart.

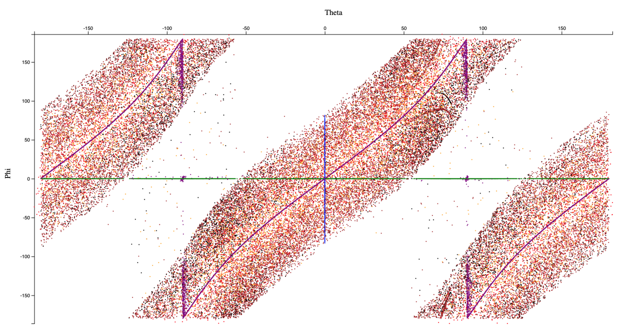


Fig. 3: All Events Chart

3) Event Timeline chart (Figs. 4 - 6a) - This visualization allows the user to explore patterns in the simulation data caused by variations in input parameters. Each row of the chart corresponds to a single simulation. The rows are ordered according to the order displayed in the parameter space component.

Fig. 4 shows an example of the Event Timeline chart. This particular dataset has some interesting periodicity in its output, as indicated by the arrows. The periodicity is not an artifact of linearization in this case, but rather, is inherent in the physical phenomenon [6, 7].

Fig. 5 displays several interesting phenomena. At (a), we can see an event family death. At that point, a threshold of some point had been reached, causing that event to stop occurring. At (b), we see an event family birth (in this case, a re-birth). This is essentially the inverse of a death. At (c), we see examples of bifurcations, where an event family divides into two distinct event families.

With this visualization, we can see the necessity of good linearization among simulations. Consider Fig. 6a, which used column-major order to linearize its simulations. Every 10 simulations, a shift occurs. These shifts draw the attention of the user but don't correspond to any phenomena in the data. Instead, these shifts are caused by large jumps in the parameter

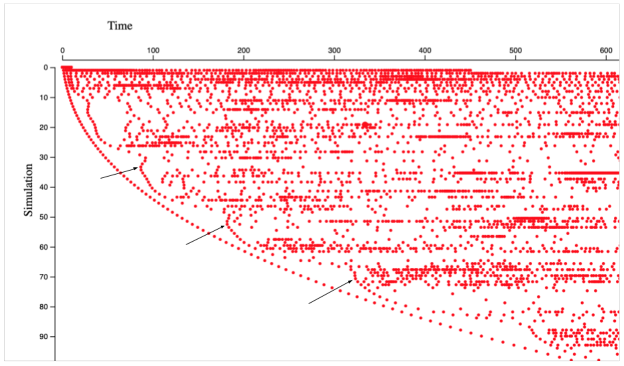


Fig. 4: This Event Timeline shows periodicity in event families. The data were produced by linearly varying the starting radius from 1 to 10 and keeping beta and theta at 0. Each simulation is given a starting momentum angle of 80 degrees and a starting location, relative to the origin, of 80 degrees. The simulations are ordered based on the starting radius. Collisions are shown.

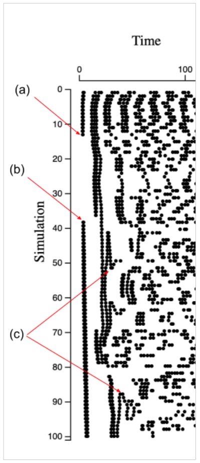


Fig. 5: This Event Timeline displays several interesting phenomena, including an event family death (a), an event family birth (b), and several bifurcations (c). These simulations were produced by varying the starting radius from 1 to 2, varying theta from 0.045 to 0.055, and varying beta from -0.045 to -0.055. As the simulations were produced in a linear manner, no linearization was necessary. Events are defined as zero crossings of the magnet's angular moment.

space. This is not ideal, as the shifts distract from meaningful patterns in the data.

Contrast Fig. 6a with Fig. 6b. This figure uses the same data as Fig. 6a, but uses z-ordering instead of column-major ordering. This change makes the output far more continuous, avoiding most of the unwanted shifts seen in Fig. 6a.

The structure of events as shown in the event timeline

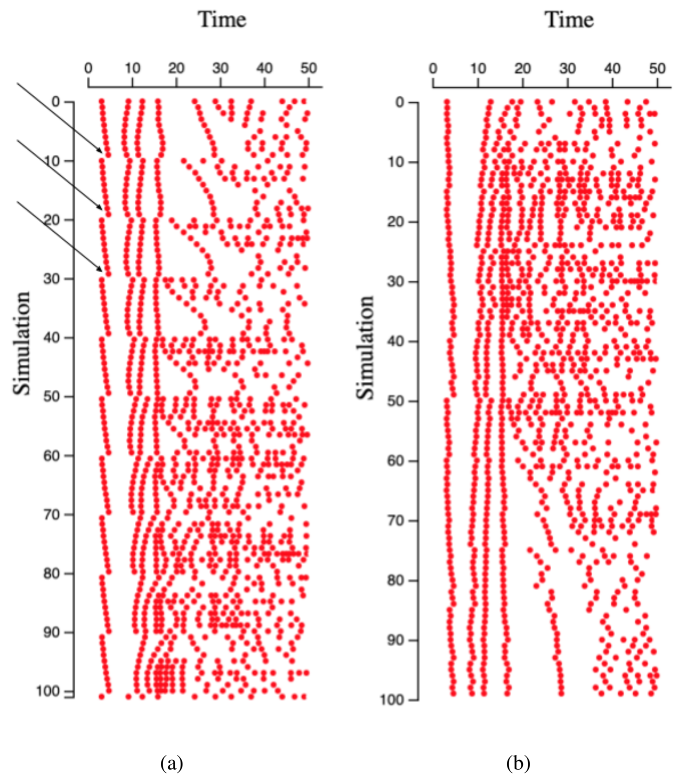


Fig. 6: (a) This Event Timeline displays a problem arising from poor linearization. The shifts occurring every 10 simulations are from the use of row-major order, not from any phenomena in the data. The large jumps occur between the simulation at the end of one row and the start of the next row. (b) This Event Timeline uses the same data as is used in 6a, but uses z-ordering to improve the linearization. This change to linearization removes the unwanted shifts. These data were produced by varying theta from 0.045 to 0.055 and beta from -0.045 to -0.055, producing a grid pattern. Radius was set to 1.5, starting momentum angle set to 80 degrees, and angle from the origin set to 80 degrees. The output events are collisions between the magnets.

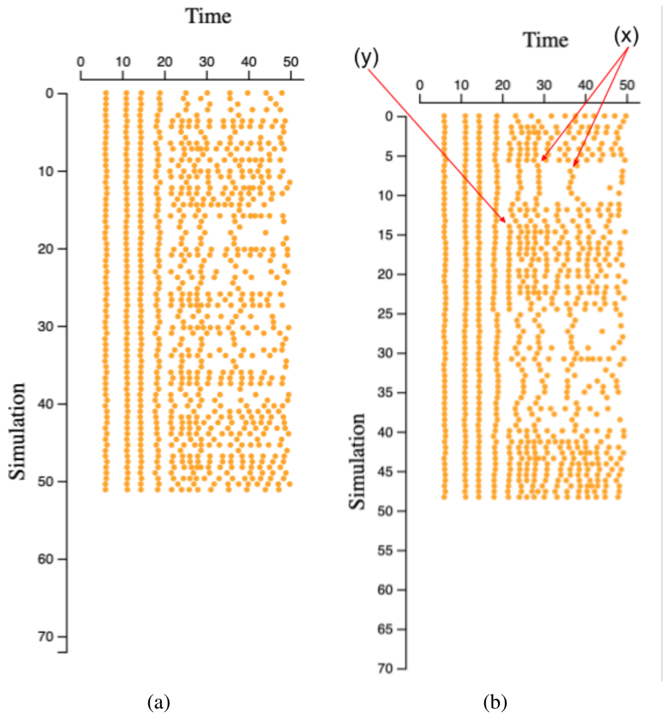


Fig. 7: These two examples of the Event Timeline Chart show how z-ordering can improve pattern recognition with randomly produced simulations. Both show the same data, which are the same data seen in 2d, and show events where the magnet's momentum was zero. Fig. (a) shows the visualization with z-ordering turned off. Fig. (b) shows the same data in z-order. Several event families that were not clear in (a) can be seen in Fig. (b), including two at (x). While both Event Timelines show the same first four event families, a fifth event family emerges in Fig. (b) at (y). Thus, we see it is much easier to find event families and patterns of events in Fig. (b). Note that the same set of simulations was removed from both of these charts because the simulations' input parameters were extremely close together.

chart is highly dependent on the presentation order of the simulations. As can be seen in Fig. 7a, some events are consistent across all orderings (the first three families of events and somewhat the fourth), but once event times start to diverge and event families are born or die based on their input parameters, the ordering becomes vital to clustering like-simulations together in terms of their event similarities. In Fig. 7b we see that using an ordering (z-ordering) with better spatial locality results in events families that are easily parsed visually. We also see event families much later in Fig. 7b than was possible to distinguish in Fig. 7a.

Because of the ease of visual parsing of event families in the timeline chart, the analyst can identify thresholds causing births and deaths of events of interest and therefore isolate dimensions (input attributes) causing the topological changes of event families.

IV. Conclusions

In this paper, we have described a novel visualization approach to discovering features of events across simulations and displayed an application of this visualization for exploring magnet dynamic simulations. This work serves as a starting point for future efforts, as it raises many questions and opportunities for improvement. In particular, a method for evaluating the effectiveness of this visualization is needed. We intend to explore the development of a human-annotated gold standard dataset to compare future empirical results against. This will only be possible as we continue to refine the definition of the problem. Needs in this area also relate to the visualization technique itself, including providing user awareness of the distance between simulations in the event timeline chart.

References

[1] Khaled Alsabti, Sanjay Ranka, and Vineet Singh. An efficient k-means clustering algorithm. 1997.

[2] Kenes Beketayev, Damir Yeliussizov, Dmitriy Morozov, Gunther H Weber, and Bernd Hamann. Measuring the distance between merge trees. In *Topological Methods in Data Analysis and Visualization III*, pages 151–165. Springer, 2014.

[3] Chun-Hung Cheng, Ada Waichee Fu, and Yi Zhang. *Entropy-based subspace clustering for mining numerical data*. PhD thesis, Citeseer, 1999.

[4] Peter J. Denning. The locality principle. *Communications of the ACM*, 2005.

[5] Herbert Edelsbrunner, John Harer, Vijay Natarajan, and Valerio Pascucci. Morse-smale complexes for piecewise linear 3-manifolds . 2003.

[6] B. F. Edwards and J. M. Edwards. Periodic nonlinear sliding modes for two uniformly magnetized spheres. *Chaos*, 27(5), 2017.

[7] B. F. Edwards, B. A. Johnson, and J. M. Edwards. Periodic bouncing modes for two uniformly magnetized spheres I: Trajectories. *Chaos*, in press.

[8] Boyd F Edwards and John M Edwards. Dynamical interactions between two uniformly magnetized spheres. *European Journal of Physics*, 38(1):015205, 2016.

[9] Chris Fraley and Adrian E Raftery. How many clusters? which clustering method? answers via model-based cluster analysis. *The computer journal*, 41(8):578–588, 1998.

[10] Samuel Gerber, Peer-Timo Bremer, Valerio Pascucci, and Ross Whitaker. Visual exploration of high dimensional scalar functions. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1271–1280, 2010.

[11] Ian Jolliffe. *Principal component analysis*. Springer, 2011.

[12] Shusen Liu, Dan Maljovec, Bei Wang, Peer-Timo Bremer, and Valerio Pascucci. Visualizing high-dimensional data: Advances in the past decade. *IEEE transactions on visualization and computer graphics*, 23(3):1249–1268, 2016.

[13] Monica Nicolau, Arnold J Levine, and Gunnar Carlsson. Topology based data analysis identifies a subgroup of breast cancers with a unique mutational profile and excellent survival. *Proceedings of the National Academy of Sciences*, 108(17):7265–7270, 2011.

[14] Richard A Norman, Francesco Ambrosetti, Alexandre MJJ Bonvin, Lucy J Colwell, Sebastian Kelm, Sandeep Kumar, and Konrad Krawczyk. Computational approaches to therapeutic antibody design: established methods and emerging trends. *Briefings in Bioinformatics*, 2019.

[15] Hanan Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR)*, 16(2):187–260, 1984.

[16] Gurjeet Singh, Facundo Mémoli, and Gunnar E Carlsson. Topological methods for the analysis of high dimensional data sets and 3d object recognition. In *SPBG*, pages 91–100, 2007.