DSC 291-E00 (Topological Data Analysis) Final Project Report: Persistence of the Conley Index for Combinatorial Multivector Fields

Report by Dylan Rowe

Submitted June 9th, 2022

Introduction and Goals

For my final project in DSC 291-E00 (Topological Data Analysis), my goal was to implement tools for topological data analysis on vector fields. Specifically, my goal was to implement elements from the paper "Persistence of the Conley Index in Combinatorial Dynamical Systems" by Tamal K. Dey, Marian Mrozek, and Ryan Slechta [1]. This paper contains methods for generating persistence diagrams that describe the topological dynamics of *combinatorial multivector fields*, extensions to the original *combinatorial vector fields* studied by Robin Forman in the context of Morse theory [5]. I also aimed to make my results suitable for implementation on meshes, i.e. general enough that they would function even outside of the plane. I will begin by reviewing some of the main ideas of the paper by Dey et al. before moving on to my results, as the results are hard to understand without context.

Combinatorial Multivector Fields and the Conley Index

A *multivector field* is a partitioning of a simplicial complex K into sets (multivectors) where every set (multivector) A in the partition is *convex*, i.e. for any simplices σ, τ, σ' where $\sigma, \sigma' \in A$ and $\sigma \leq \tau \leq \sigma'$, $\tau \in A$. Two important notions in the theory of multivectors are the idea of the *closure* of a multivector or simplex (denoted $cl(\cdot)$) and the *mouth* of a multivector or simplex (denoted $mo(\cdot)$). The closure is the set of all (not necessarily proper) faces of the simplices in the multivector; the mouth is defined as the part of the closure of the multivector which is outside of the multivector itself, i.e. $mo(A) = cl(A) \backslash A$. These operators are important because they help in describing the flow of a multivector field.

In a multivector field, flow dynamics are defined by a multivalued map $F_V(\sigma)$ on the simplices $\sigma \in K$. The map sends simplices to the union of their closure and their multivector: with $[\sigma]$ as the unique multivector containing σ, $F_V(\sigma) = cl(\sigma) \cup [\sigma]$. Informally, simplices can flow freely within their multivector, or to incident simplices of lower dimension.

Another important notion is that of the *invariant part* ($Inv(\cdot)$) of a set of simplices. The details of this operator involve machinery beyond the scope of this report, but the basic idea is that the invariant part of a set of simplices describes those simplices which are starting points for flow paths that remain in the set.

Finally, there is also the idea of the *pushforward* ($pf(\cdot)$) of a set of simplices; this is simply the result of flowing a set of simplices forward in time with respect to the multivector field dynamics. In the context of the Dey et al. paper, the pushforward functions as a means of growing a set of simplices to make it more likely to intersect with similar sets of simplices in the next time step, which is necessary to make zigzag persistence robust to larger timesteps and coarser meshes.

Finally, there is also the notion of the *index pair* for an isolated invariant set S, which is a pair of sets (P, E) such that $F_V(E) \cap E \subseteq E$, $F_V(P \backslash E) \subseteq P$, and $S = Inv(P \backslash E)$. P and E describe the flow mechanics surrounding the isolated invariant set S using sets; things in E will flow either into P or E, but things in P outside of E cannot flow back into E; on top of this, S is contained in E and flows outward to cover P, which is what allows the index pair to describe behavior around S. The idea of the *Conley index*

of dimension p is that the relative homologies $H_p(P, E)$ for any P and E corresponding to the same S are isomorphic. The Dey et al. paper uses this concept to compute the zigzag persistence of the relative homology of these sets as a multivector field changes over time (or with respect to some parameter).

Implementation

For this project, I implemented many aspects of the mesh geometry and connectivity, vector field discretisation, multivector field dynamics, and multivector operators. I also implemented routines for computing the persistence of the Conley Index for specific isolated invariant sets. Most of my implementation was performed in Python, and I used Matplotlib for plotting, NumPy for most numerical routines, and Dionysus 2 for homology / zigzag persistence.

One thing that was crucial for most of the computations I performed was the assumption that they were taking place on a 2-manifold, i.e. a triangle mesh. My first steps in implementation were to define classes that would help in implementing operations on this mesh. I chose to use the *halfedge mesh*, a data structure ubiquitous in graphics and geometry processing. The halfedge mesh describes a mesh's connectivity using *halfedges*, which are directed members of the mesh that lie along edges. These halfedges have pointers to and from each other as well as to nearby vertices, edges, and faces, which allowed me to implement many local operations on the mesh smoothly. I implemented the halfedge mesh in pure Python. I also implemented methods that allowed me to display the mesh using Matplotlib, get the center points of arbitrary simplices, and create the mesh from a set of triangles and vertex positions. This step helped a lot in the overall implementation, as it enabled me to freely modify the halfedge data structure and create new methods specific to my goals.

Next, I implemented the discretization of vector fields into multivector fields. This is a nontrivial task, since from the algebraic/set theoretic definition of a multivector, it is not obvious how one can build a multivector field from a standard continuous vector field. I wanted my program's input to be a continuous function describing a vector field (or, alternatively, samples of some vector field); for this, I consulted the original paper on multivector fields, "Conley-Morse-Forman Theory for Combinatorial Multivector Fields" by Marian Mrozek [2]. This paper describes an algorithm called CMVF ("combinatorial multivector field") which is suitable only for samples at evenly-spaced gridpoints, and which works only for square grid cell complexes. Unfortunately, for my purposes, I needed an implementation that functioned on triangle meshes; though the settings seem similar, I found that the CMVF algorithm was heavily dependent on the grid structure for its discretisation, and was forced to generalise the algorithm.

Roughly, the original implementation of CMVF begins by sampling the vector field at the vertices of the grid before applying an operator $D_{\mu,\epsilon}$ which snaps the vectors onto vertices, edges, and faces (with respect to some thresholds on the length and angles of the original vectors). Next, it pairs each simplex in the mesh with itself before pairing edges with incident faces if their endpoints match in direction. Finally, each vertex is merged with edges and faces based on compatibility with those edges/faces as well as compatibility with nearby vertices; this final step involves a series of complicated conditions. The output of CMVF is a function $\theta(\cdot)$, where the preimages of the elements in the image of $\theta(\cdot)$ are precisely the sets of simplices defining the multivectors on the mesh. My generalization of the CMVF procedure first involved a similar snapping of vectors onto edges, but instead of just snapping the coordinates of the vectors, I created pointers to nearby simplices. This step involved some extra geometry as well, as I needed to find a perpendicular axis to edges which would be suitable for meshes embedded in 3 dimensions, as well as projections of the edges incident to a vertex onto a common plane that would allow for angle comparisons. However, my difficulties did not end with the

discretization of $D_{\mu,\epsilon}$, and I also had to generalise much of the pairing section of the CMVF procedure by interpreting the pseudocode of the original paper in mesh terms. In the end, my generalisation seemed to give reasonable results in practice which certainly capture the major topological features of the multivector field.

My next step was to implement other operators on multivectors that would aid me in implementing the persistence algorithm. I implemented first $cl(\cdot)$ and $mo(\cdot)$ using a recursive procedure that appended simplices from higher to lower dimensions. Next, I implemented a method for $pf(\cdot)$ which used depth first search to explore the mesh using the previously-defined dynamics of $F_V(\cdot)$. I also implemented methods to determine whether or not a multivector was *critical* (defined in this context as having a nontrivial $H_p(cl(A), mo(A))$ for some p), a condition which helps in finding the invariant part of a multivector field.

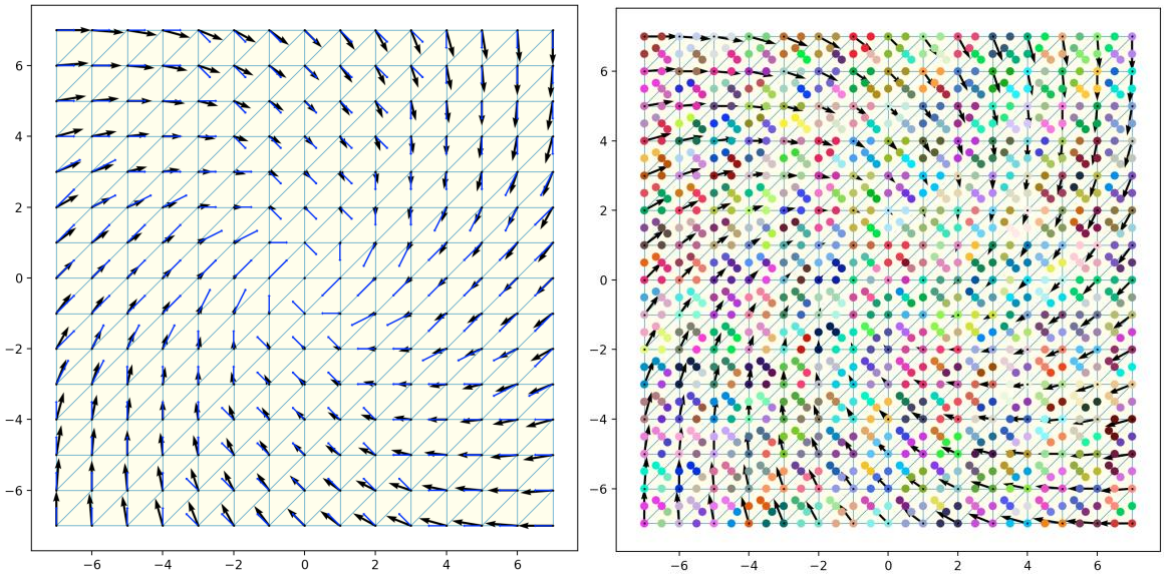
Finally, I also implemented a function which returned the invariant part of a given multivector field; this was the most difficult operator to implement, as I could not find a paper or resource which explicitly gave an algorithm that would compute $Inv(\cdot)$ efficiently. I came up with a rudimentary algorithm for calculating this part of the vector field; my algorithm is to first take the union of all the critical multivectors in the vector field, and then for every multivector in the mesh, I run DFS to determine whether or not it can return to itself using only the flow dynamics. The first step is meant to capture fixed points of the field, and the second step is meant to capture cyclical parts of the invariant part; though my algorithm would certainly find all of the cyclic parts of the field and all of the critical multivectors, I am not sure if this is sufficient to capture the entirety of the invariant part. I also quickly scanned some of the older literature on the topological dynamics of vector fields, and was unable to find any complete characterisations of the invariant part which would confirm the correctness of my algorithm. Fortunately, my algorithm appears to give subsets which look roughly like the invariant part of input vector fields, and the persistence algorithm defined in Dey et al. pushes forward these parts anyway, which may make up for any deficiencies in approximations I've made.

My final step was to implement the actual persistence algorithm for the multivector field. The algorithm in Dey et al. functions by first finding the invariant part of the mesh with respect to the multivector field at time t, before taking the closure and mouth of this invariant part. From there, the pushforwards of both closure and mouth are computed to expand the size of these subsets. This procedure is performed at every timestep. At the end, the algorithm computes the intersection of adjacent Ps and Es before computing the persistence of the relative homologies of these index pairs. One issue I encountered in this step is that none of the libraries I found on the course website were able to directly compute relative zigzag homology. Dionysus 2 is able to compute relative homology persistence alone, as well as zigzag persistence alone, but was not able to directly compute both for a given sequence. To counter this, I created a dummy vertex and connected all vertices in E with the dummy vertex to simulate the "cone" construction in Professor Wang's textbook on topological data analysis. This seemed somewhat costly, but appears to run quickly in practice. I also ran into issues with the Dionysus library trying to compute zigzag persistence; the library confirmed that my input was a simplicial complex, but gave me a segmentation fault when I tried to compute the persistence diagram.

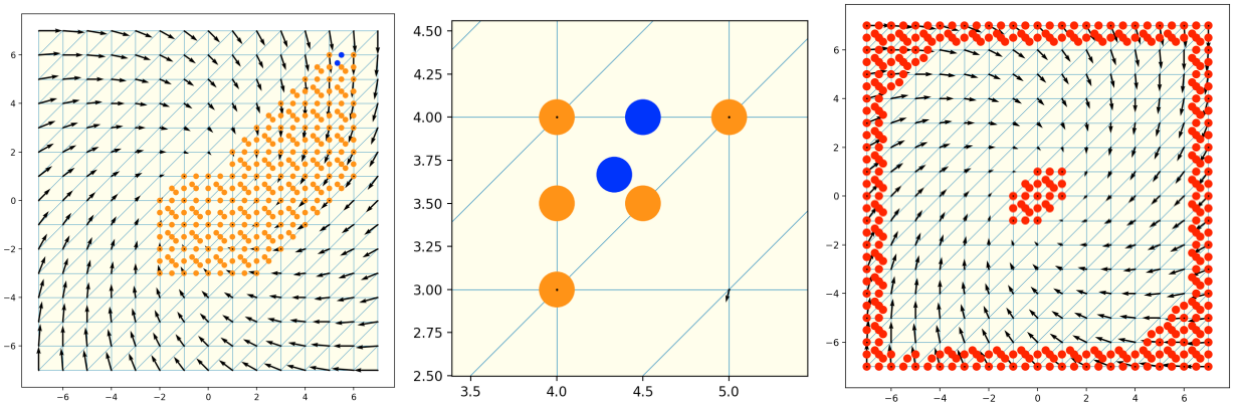
Observations

The following is the result of my modified CMVF algorithm. On the left, I've displayed the result of snapping the continuous vectors to the simplices of the mesh (blue) as well as the original continuous vectors (black). Clearly, the result of snapping the vectors using $D_{\mu,\epsilon}$ is similar to the original vector

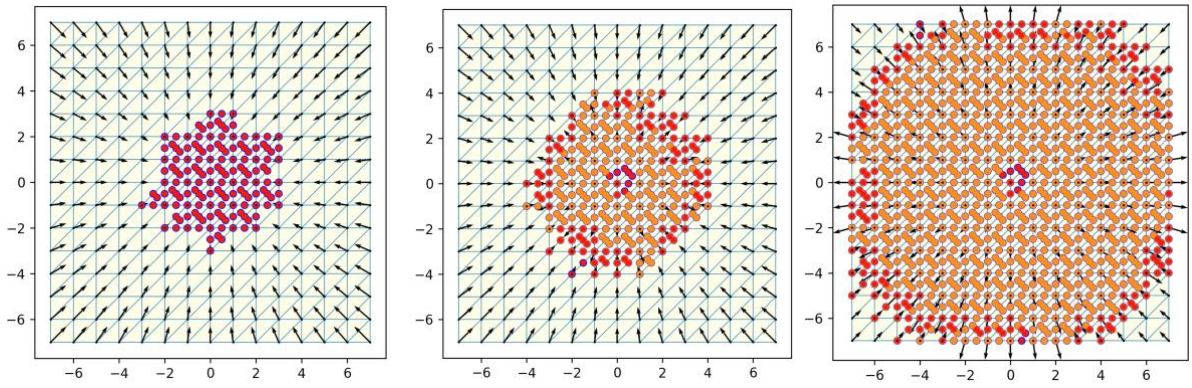
field; one notable difference is that on the outside of the grid, some of the snapped vectors face inward rather than outward. This is an artifact of the border alone, as I've enforced the condition that snapped vectors should not point outside of the mesh, but rather at the closest face/edge within the mesh (along with the conditions that the vectors should snap based on the angle/length thresholds). On the right is a visualisation of my implementation of CMVF. Members of the same multivector are displayed in the same color. Superimposed on the multivector field is the original continuous vector field, sampled at vertices. The dynamics of the multivector field are such that within a multivector, flow occurs upward or downward in dimension, and between multivectors, flow occurs only downward in dimension. This suggests that my implementation is mostly correct, as the dynamics of the multivectors appears similar to the dynamics one would expect from the vector field.



The results of my implementation of pf, cl, mo, and Inv are also . For pf (left), I've displayed the pushforward (orange) of a single multivector (blue) in the flow field; the multivector appears to be pushed forward by the dynamics. For cl and mo (center), I've done the same; the multivector is displayed in blue below, and its mo is displayed in orange. Finally, I've evaluated Inv on a test vector field (right) which has an additional outer cycle; the invariant multivectors appear to be at the fixed point in the center and along the invariant cycle around the outside, which matches one's intuition about the behavior of this vector field.



Finally, I will come to my results relating to the relative zigzag persistence. For the relative zigzag persistence part of my project, I used a vector field with a bifurcation; this had an attracting (and rotating) fixed point for $t < 0$, after which it becomes repelling inside of some growing radius and attracting outside of that radius. This vector field should initially have a large invariant set in the middle, which becomes a small invariant set with a ring outside after the bifurcation occurs. I chose this type of vector field because it should showcase the topological changes in the dynamics well, and be very visible in the resulting persistence bars. Below are some samples of the dynamics at several time points showing the behavior of this vector field. S is shown in red for each time point; (P, E) are in blue and orange respectively. When a simplex is colored two different colors, it has outlines that are the other colors (this is especially evident in the center and in E, where the red/orange simplices are consistently blue as well, as they are subsets of P).



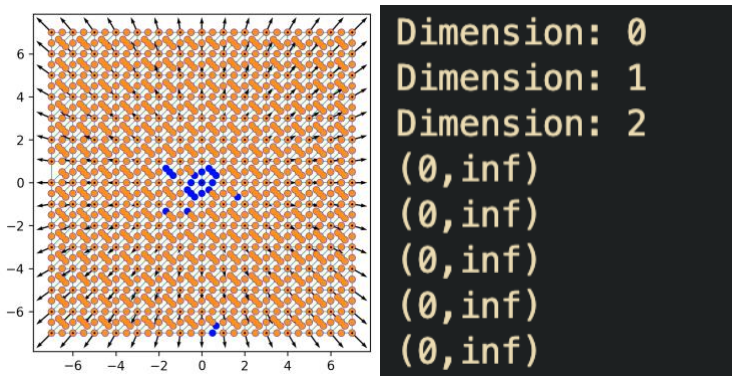
Next, I computed the persistence of the Conley index for this changing vector field at several time points from -2 to 3. Again, the Conley index for some dimension is the relative homology of the P set with respect to the E set of the index pair for some S. For my project, I used the invariant part of the entire mesh as S, and the pushforwards of S's closure and mouth as P and E. The relative homology was computed by constructing the cone complex for P/E. My expectation for the persistence barcode was that one would be able to see the evolution of the above Conley index, but unfortunately my implementation found a segmentation fault for most changing vector fields.

However, I was able to get results for stationary vector fields that match my intuition for the Conley index. For the vector field where every vector points towards the origin, my program gave the following result (where the first time point is at time 0):

```
Dimension: 0
(0,inf)
```

This makes sense, since the relative homology for the invariant part of this vector field with respect to its (nonexistent) E would be simply a single connected component.

I also attempted to determine the Conley indices for a vector field which is rotating near the center, but shoots outward from the origin after a certain radius. The multivector field's index pair appeared as below, with the corresponding output from the persistence algorithm



This result also makes sense, since the relative homology of the invariant part (the rotating region within the radius) with respect to the pushforward of its mouth (the regious outside of the radius) is equivalent to the quotient topology of a circle with respect to its boundary (by the excision theorem); this is precisely homeomorphic to the 2-sphere. Due to noise in the multivector field, however, the program saw 5 2-cycle homology generators.

Thus my program computes the Conley index correctly for stationary multivector fields; however, I was unable to get it to function for multivector fields that changed over time, due to the aforementioned segmentation fault issue.

I will also mention that while I did not test my implementation on meshes with vertices in 3D, the implementation is completely invariant to vertex positions and should work for arbitrary 3D meshes (subject to conditions on the quality of the mesh, such as that it should probably be orientable and not have any spurious edges which are not members of a face). Additionally, my implementation of the halfedge mesh data structure includes a constructor that works with "bag of triangles" formats such as .obj files, and would be very easy to port to such a setting in the future.

Future Directions

In the future, I'd like to extend this project to function for moving vector fields like the one shown in this report. I'd also like to attempt to apply it to 3D meshes, which I was unable to do due to time constraints; fortunately, this step should be fairly easy given that my implementation is already agnostic to the positions of vertices and the dihedral angles between adjacent triangles; my implementation is also agnostic to (well-behaved) boundaries, and thus would also be able to handle standard meshes.

Another direction I'd like to take this project in in the future is an implementation of the noise-resilient index pair algorithm given in [1]. In this section of the paper, the authors compute index pairs which are resilient to noise like the kind I experienced in my 2-cycle demonstration above. The idea of this is that one can decrease the size of E in a way which makes it robust against noise, and more likely to intersect with adjacent timesteps.

I'd also like to implement parts of similar systems from the authors' other works in the same area. One such paper [3] computes the "Conley-Morse graph" of a multivector field, which is the *Morse graph* of the multivector field (which is analogous to the strongly connected component decomposition for directed graphs), but where every vertex is annotated with the generating polynomial for the Conley indices at each dimension for that component of the multivector field. Such an improvement to my

program would allow it to give more information about the dynamical structure of the vector fields being analysed.

Finally, I would like to use my implementation on real-world data, perhaps by finding weather data or the other types of data which I mentioned in passing in my project proposal. I think that this formalism for discretization and analysis of vector fields is very powerful, and is widely applicable in scientific computing, graphics, computer vision, and beyond.

References

[1] Tamal K. Dey, Marian Mrozek, and Ryan Slechta. "Persistence of the Conley Index in Combinatorial Dynamical Systems". https://arxiv.org/pdf/2003.05579.pdf. March 2020.

[2] Marian Mrozek. "Conley-Morse-Forman Theory for Combinatorial Multivector Fields". https://arxiv.org/pdf/1506.00018.pdf. May 2016.

[3] Tamal K. Dey, Marian Mrozek, and Ryan Slechta. "Persistence of Conley-Morse Graphs in Combinatorial Dynamical Systems". https://arxiv.org/abs/2107.02115. July 2021.

[4] Robin Forman. "Combinatorial Vector Fields and Dynamical Systems". https://link.springer.com/article/10.1007/PL00004638. August 1995.