Braiding Dynamics in Active Nematics - Appendices

Spencer Ambrose Smith, Ruozhen Gong

I. APPENDIX 1: BRAIDING ON THE SPHERE - DETAILS

In this section (following the introduction of spherical braids in section 9 of the main text), we explain in detail how to calculate the topological entropy given a braid on a sphere. The algorithm for calculating the topological entropy for a general surface braid can be found in this paper[1]. The general procedure is that we use a triangulation to set up a coordinate system for closed curves, calculate the action of braid generators on these coordinates, and measure the exponential growth rate of curve length, which constitutes the topological entropy. Here, we focus on the specific case of spherical braids.

We have set up the tetrahedral graph to model the motion of 4 +1/2 topological defects on the sphere, as labeled in fig. 7 of the main text. As seen in section 4 of the main text, the topological entropy is the log of the braid dilation factor λ, which is the stretching rate of curves along the leaves of the unstable foliation, or alternatively, the compression rate of curves along the leaves of the stable foliation. We encode the way curves wind around each vertex of the tetrahedral graph by specifying a triangulation. In this case, the triangulation is the tetrahedral graph we already have. Any closed curve around the graph can be topologically described by the triangulation. An intersection coordinate is a weight assigned to each edge that counts the number of transverse intersections of the curve with that edge. The intersection coordinates can encode a set of closed curves, or equivalently, they can record the flux of foliations through that edge.

To illustrate how the intersection coordinates help us encode the stretching of material curves, consider the curves in fig.1. Here we use an initial curve that is described by the set of edge weights $E = (1, 1, 0, 0, 1, 1)$, meaning the curve intersects once respectively with the first, second, fifth, and sixth edges, as demonstrated by the leftmost graph of fig.1. After the CCW rotation of edge 1, the intersection coordinates become $E = (1, 1, 1, 1, 2, 2)$. After the CCW rotation of edge 5, the intersection coordinates become $E = (1, 1, 3, 3, 2, 2)$. The total edge weights has grown from 4 to 8 and then to 12. With each braid operation, the intersection coordinates are updated to reflect the stretching and folding of the curve. Fortunately, the general update rule for intersection coordinates is simple to derive.

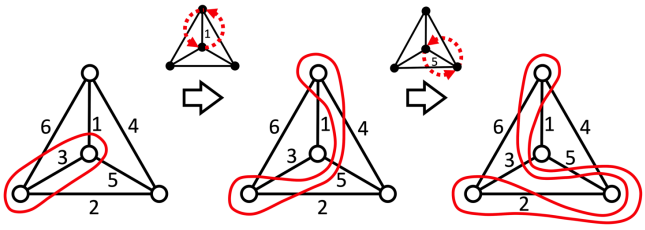

FIG. 1: An example closed curve is stretched out due to the action of two braid generators: σ_1 and σ_5.

To derive a general update rule, we must know the effect of each braid generator on the intersection coordinates. Recall that we refer to the pair switches between two points by the edge connecting them, and note that there are 6 such edges. We use a $\pm$ exponent to denote a counter clockwise (CCW +) or clockwise (CW −) exchange. So, the 12 generators are: $\sigma_i^{\pm}$ for $i \in [1, 6]$. Because of the rotational and mirror symmetry of the tetrahedron, we can map each edge to the position of edge 1 through some $2\pi/3$ rotation or mirror reflection. Therefore, we only need to fully define the action of $S = \sigma_1^+$, the CCW rotation of edge 1. All other generators can be related to this one by conjugating with a set of symmetries. The update rule for S is constructed by breaking down the point interchange into a series of "flip" operations on the triangulation, sometimes called Whitehead moves, and using our edge updating formula at each step. As shown in fig. 2, we denote the edge between the two triangles as E; A,B,C,D are the edges of the quadrilateral in cyclic order; and E' is the new edge after the flip. The coordinate of the new edge E' is updated according to the formula

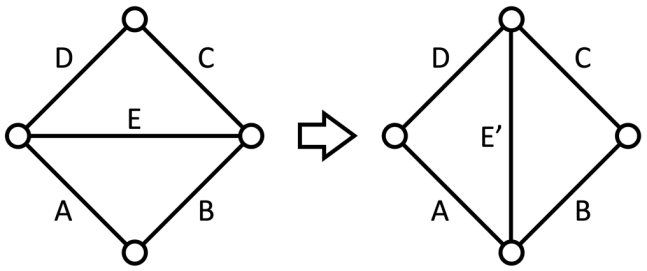

FIG. 2: A triangulation flip, or Whitehead move. The intersection coordinate of the new edge, E', is give by eq. 1.

$$E' = \max(A + C, B + D) - E \equiv \Delta(A, B, C, D; E). \quad (1)$$

The braid generator σ_1 can be realized with two edge flips (edges 3 and 4), and a CCW motion of the points adjacent to edge 1, as shown in fig. 3. Applying the coordinate update formula to this series of Whitehead move gives the overall update rule for S. First, we update the coordinates of edge 3 and edge 4

$$E_3' = \Delta(E_1, E_6, E_2, E_5; E_3) \quad (2)$$

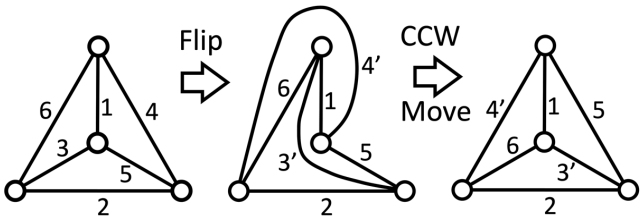

FIG. 3: The triangulation flips and point motions that realize braid generator σ_1. Updating the intersection coordinates consists of two applications of eq. 1, and a permutation of the coordinates.

$$E_4' = \Delta(E_1, E_5, E_2, E_6; E_4). \tag{3}$$

Then notice that the new set of coordinates is a permutation of the initial ones. So the set of new intersection coordinates is given by

$$E^n = (E_1, E_2, E_6, E_5, E_3', E_4'). \tag{4}$$

Other braid generators can be connected to S by a series of rotational and mirror symmetries. The idea is that every edge can get into the position of edge 1 through some rotations and reflections. First we will have R to denote a CCW rotation by $2\pi/3$ about the center point, $\overline{R}$ for a CW rotation by $2\pi/3$, and a mirror inversion about the vertical center line M. We will also use T for a right handed (positive) rotation of $2\pi/3$ about an axis going through the top point and bottom face. The inverse is $\overline{T}$. For $i \in [1,2,3,4,5,6]$, we give the action of the symmetry by the permutation $\pi(i)$. For the moment, we are just thinking about how the symmetries act on the lattice edges.

$$R : \pi = (5,6,1,2,3,4) \tag{5}$$
$$\overline{R} : \pi = (3,4,5,6,1,2) \tag{6}$$
$$M : \pi = (1,2,5,6,3,4) \tag{7}$$
$$T : \pi = (6,5,2,1,3,4) \tag{8}$$
$$\overline{T} : \pi = (4,3,5,6,2,1) \tag{9}$$

Notice that M will switch CCW and CW braid generators, and $R^3 = \overline{R}^3 = M^2 = T^3 = \overline{T}^3 = \mathbf{1}$.

Since we have fully defined the action of the generator $S = \sigma_1^+$ on the coordinates, all other generators can be related to this one by conjugating with a set of symmetries:

$$\sigma_1^+ = S \tag{10}$$
$$\sigma_2^+ = \overline{R}T S \overline{T} R \tag{11}$$
$$\sigma_3^+ = RS\overline{R} \tag{12}$$
$$\sigma_4^+ = TS\overline{T} \tag{13}$$
$$\sigma_5^+ = \overline{R}SR \tag{14}$$
$$\sigma_6^+ = \overline{T}ST \tag{15}$$

The operators act via left action; in other words, right most operator acts first. The negative generators are given by substituting S by MSM.

To compute the topological entropy, we start with an initial set of intersection coordinates that correspond to a closed curve. For pA braids, the initial intersection coordinates do not matter, since all non-trivial loop would produce the same exponential stretching rate asymptotically. In our algorithm, we use the sum of the intersection coordinates as an effective proxy for the length of the curve. Given a braid β composed of N braid operations, we record the updated intersection coordinates at each step to get a sequence of intersection coordinates. At step k, we calculate the sum of intersection coordinates, $W_k = \sum_i [\vec{E_k}]_i$. Then the topological entropy per operation (TEPO) for the braid β, $\bar{h}$, is given by

$$\bar{h} = \lim_{k\to\infty} \frac{1}{N} \log(\frac{W_k}{W_{k-1}}). \tag{16}$$

To obtain the asymptotic maximum TEPO, we iteratively compute $\bar{h}_k = \frac{1}{N}\log(\frac{W_k}{W_{k-1}})$, and such iteration stops after $\mid \bar{h}_k - \bar{h}_{k-1} \mid$ drops below a small enough tolerance.

A. Appendix 2: Extracting Braids from Experimental Data

This section follows from the discussion in section 10 of the main text. We give more details on extracting braid words from computational or experimental data, which contains the three unique angle differences at each time step. The defect motion consistent with any braid generator takes a tetrahedral configuration to another tetrahedral configuration by passing through a co-planar configuration. So, to identify the braid generator, we need to identify the initial configuration of the tetrahedron (right or left handed orientation) and which co-planar configuration that it passes through. The initial tetrahedron plus the series of subsequent co-planar configurations will uniquely give us a braid word.

To identify the initial orientation of the tetrahedron, we start with the cartesian positional vectors, $\vec{r}_i$, of each labeled defect. Then calculate $((\vec{r}_2 - \vec{r}_1) \times (\vec{r}_3 - \vec{r}_1)) \cdot \vec{r}_4$. If it is > 0, then the tetrahedron has a right handed orientation (C_1 vertex labeling class), while if this value is < 0, then it has a left handed orientation (C_2 vertex labeling class).

Next, note that starting with a given tetrahedral orientation, there are three possible co-planar configurations that are possible (see fig.4). We can determine which of these co-planar configurations happens by noting the pairs of labeled defects that are along the diagonals (i.e. have angle differences that are close to 180°). These pairs of diagonal points determine the relative position of all four points. Given the tetrahedral configuration prior to the co-planar one, we can identify the pair of points being swapped in this transformation, thus matching the transformation to a unique braid word.

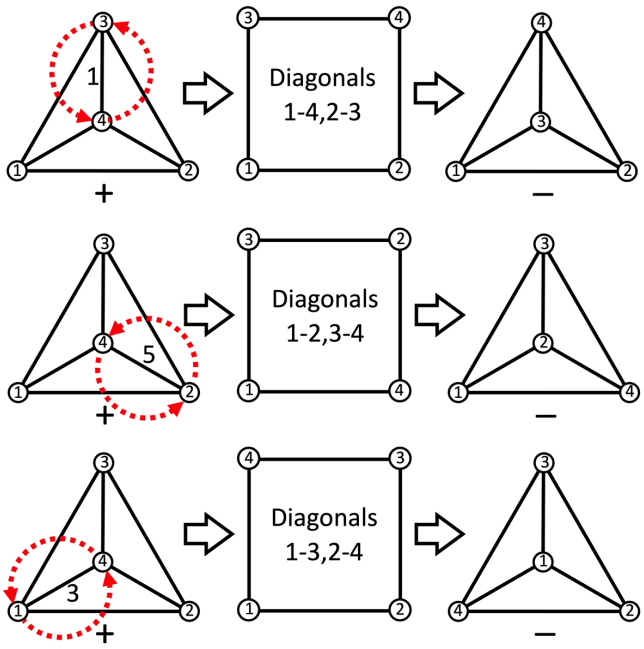

FIG. 4: The three co-planar configurations that are possible in going from the tetrahedron on the left to another tetrahedral configuration. These correspond to three braid generators.

To illustrate with an example, the tetrahedron on the left side of fig. 4 is right handed (labeled with a "+"). If we know from the computational or experimental data that point 1 and point 4 are the diagonals of the succeeding co-planar configuration, we know that point 3 and point 4 must swap in a CCW direction. So the initial tetrahedron must go through the braid generator σ_1 to turn into the tetrahedron at the top right corner. Likewise, if point 1 and point 2 are the diagonals, the tetrahedron goes through the σ_5 transformation; and if point 1 and point 3 are the diagonals, the tetrahedron goes through the σ_3 transformation. Given the initial position of the points, the diagonal points of the following co-planar configuration tell us which of the three generators we need in order to go to the next tetrahedron. The next pair of diagonal points will tell us how to generate the next tetrahedron. Thus giving us a string of braid words.

REFERENCES

[1]S. A. Smith and S. Dunn, "Topological entropy of surface braids and maximally efficient mixing," SIAM Journal on Applied Dynamical Systems **21** (2022).