YARN BALL KNOTS AND FASTER COMPUTATIONS

DROR BAR-NATAN, ITAI BAR-NATAN, IVA HALACHEVA, AND NANCY SCHERICH

Abstract. We make use of the 3D nature of knots and links to find savings in computational complexity when computing knot invariants such as the linking number and, in general, most finite type invariants. These savings are achieved in comparison with the 2D approach to knots using knot diagrams.

Contents

1. Introduction

A recurring question in knot theory is "Do we have a 3D understanding of a knot invariant?". We often think about knots as planar diagrams and compute invariants from the diagrams. In this paper, we consider computation from a planar diagram to be a "2D" understanding of the invariant, For example, Kauffman [Kau90] gave a description of the Jones polynomial using planar diagrams, bringing the understanding of this invariant to 2D.[1] Witten [Wit89] developed Chern-Simons theory, a 3-dimensional topological quantum field theory, partially in order to understand the Jones polynomial in a 3D way. However, this theory does not seem to lead to easier computations of the invariant. So, while the Chern-Simons theory description is 3D, from the point of view of this paper, it is (at least computationally) still an incomplete 3D understanding of the Jones polynomial.

For complexity measurements in this paper, we care only about the polynomial degree (i.e. big O modulo log terms). We write $A \sim B$ for two expressions A, B depending on n if $O(A) = O(B)$, with the assumption that $\log(n) \sim 1$. For example, when comparing sizes or complexities, we agree that $7n^3(\log n)^5$ is the same as n^3 and as $\frac{1}{120}n^3/\log n$, and write $7n^3(\log n)^5 \sim n^3 \sim \frac{1}{120}n^3/\log n$.

The complexity of planar knot diagrams are often measured by their crossing number, denoted by n in this paper, and the computational complexity of a knot invariant is typically described in terms of n. Three-dimensionally, planar diagrams are embedded in "pancakes"– very flat and wide subsets of $\mathbb{R}^3$, as in Figure 1 (A). This pancake description of a knot

Key words and phrases. finite type invariants.

[1]In some sense, the original description of the Jones polynomial [Jon85], by Vaughan Jones and using braids, is 1D. For, while the full 3D rotation group $SO(3)$ acts on 3D space, and the 2D rotation group $SO(2)$ acts on planar diagrams, no continuous symmetries are left when one comes down to braids.

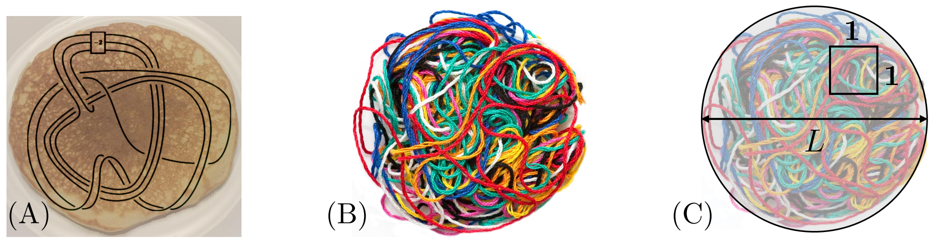


FIGURE 1. (A) A planar diagram of a knot in a pancake (knot diagram by Piccirillo [Pic20]). (B) A yarn ball. (C) Measurements on a yarn ball knot.

is somewhat artificial and does not realistically describe knots that occur in nature. For example, knotted DNA is shaped much more like the ball of yarn in Figure 1 (B), than the pancake knot in (A). Since we aim to have a 3D understanding of knot invariants, we study knots in the shape of the yarn ball as opposed to the pancake. We define a *yarn ball knot* to be a knotted tube of uniform width 1 that is tightly packed in a ball, as in Figure 1 (B). Equivalently, a yarn ball knot is a knot embedded in a grid, which we discuss in depth in Section 2. Let L be the diameter of the yarn ball knot, then the volume of the knot is $V \sim L^3$. The volume also measures the length of the yarn, or how much yarn was used to make the knot.

By projecting the yarn ball knot in a generic direction onto a disk, we attain a planar knot diagram. The crossing number of this projection can be estimated by subdividing the disk into 1×1 squares, as in Figure 1 (C). Most such 1×1 squares will have L layers of strands above them, and one can expect these strands to cross around $\sim \left(\binom{L}{2}\right) \sim L^2$ times in that square. Since there are $\sim L^2$ squares, the total crossing number of this projection is around $n \sim L^2 L^2 = L^4 = V^{4/3}$.

We see that to describe a yarn ball knot of volume (or length) V as a planar diagram, we would need $\sim V^{4/3}$ crossings. As $V^{4/3} \gg V$, it requires many more bits to describe a yarn ball knot via its projection rather than directly as a yarn ball.

If ζ is a knot invariant, we denote by $C_\zeta(2D, n)$ the worst-case complexity of computing ζ on a knot given by a planar diagram with n crossings, and by $C_\zeta(3D, V)$ the worst-case complexity of computing ζ on a knot given as a yarn ball of volume V. Given a yarn ball of volume V we can always compute ζ by first projecting to the plane[2], obtaining a planar diagram with $\sim V^{4/3}$ crossings, and then computing ζ using our best 2D techniques. Hence always, $C_\zeta(3D, V) \leq C_\zeta(2D, V^{4/3})$. It is interesting to know whether we can do better:

Conversation Starter 1. *A knot invariant ζ is said to be* computationally 3D*, or C3D, if*

$$C_\zeta(3D, V) \ll C_\zeta(2D, V^{4/3}).$$

In other words, ζ is C3D if substantial savings can be made to the computation of ζ on a yarn ball knot, relative to the complexity of computing ζ by first projecting the yarn ball to the plane.

This is not a formal definition! The notion of an invariant being computationally 3D is dependent on the current knowledge of the invariant. As our understanding grows and our computational techniques get better, an invariant might become *C3D*, or lose its *C3D* status.

[2]The projection itself can be computed quickly, in time $\sim V^{4/3}$, and for all interesting ζ, this extra work is negligible

However, the question whether an invariant is *C3D*, as we understand it at a given time, still has merit as it measures our understanding, as a community, of knot theory as a 3D subject. Our first example of a *C3D* invariant is the linking number:

Theorem 2.1. (Proof in Section 2) *Let lk denote the linking number of a 2-component link. Then $C_{lk}(2D, n) \sim n$ while $C_{lk}(3D, V) \sim V$, so lk is C3D.*

The linking number of a link is an example of a *finite type invariant.* Finite type invariants underlie many of the classical knot invariants, for instance they give the coefficients of the Jones, Alexander, and more generally HOMPFLY-PT polynomials [BL93, BN95a].

Theorem 3.2. (Proof in Section 3) *If ζ is a finite type invariant of type d then $C_\zeta(2D, n)$ is at most $\sim n^d$.*

Theorem 3.3. (Proof in Section 3) *If ζ is a finite type invariant of type d then $C_\zeta(3D, V)$ is at most $\sim V^d$.*

Naive Conclusion. As $n^d \sim (V^{4/3})^d \gg V^d$, finite type invariants are tentatively C3D.

The above conclusion is naive, for Theorems 3.2 and 3.3 give only one sided bounds. The actual complexities $C_\zeta(2D, n)$ for some specific though "special" finite type ζ's, such as the coefficients of the Alexander polynomial, are known to be much smaller. Even for generic ζ's, in a later publication [BNBNHS] we plan to improve the bounds in both theorems by substantial amounts. Yet we believe that our naive conclusion remains valid, at least in the form "most finite type invariants are C3D".

The opinion we present in this paper is that in general knot invariants should be *C3D*; knots are three-dimensional and the best way to understand a knot should be three-dimensionally. Unfortunately, as of the time this paper is written, very few knot invariants are known to be *C3D*. Are the Alexander, Jones, or HOMFLY-PT polynomials *C3D*? Why or why not? Are the Reshetikhin-Turaev invariants *C3D*? Are knot homologies *C3D*? While we seem to have a weak understanding of these fundamental invariants from a 3D perspective, this is cause for optimism; there is still much work to be done.

Instead of using computational complexity to compare 2D and 3D understandings of invariants, we can also use the notion of the maximal value of other quantities relating to the size of the knot, which motivates the next conversation starter.

Conversation Starter 2. *If η is a stingy quantity (i.e. we expect it to be small for small knots), we say η has savings in 3D, or has 'S3D' if*

$$M_\eta(3D, V) \ll M_\eta(2D, V^{4/3}),$$

where $M_\eta(kD, s)$ is the maximum value of η on all knots described k-dimensionally and of size s.

For example, the hyperbolic volume is a stingy quantity–the more complicated a knot is, the more complicated its complement in S^3 will be, which makes the question of putting a hyperbolic structure on it harder. We expect hyperbolic volume to have savings in 3 dimensions.

Conjecture (Bar-Natan, van der Veen) Hyperbolic volume has *S3D*.

The genus of a knot is another example of a stingy quantity, but we do not know if the genus of a knot has *S3D*, or not. If a knot is given in 3-dimensions, is the best way to find the genus truly to compute the Seifert surface from a projection to 2D, at a great cost? The genus is by all means a 3D property of a knot, and it seems as though it *should* be best computed in a 3D manner.

We hope that these conversation starters will encourage our readers to think about more 3D computational methods. The remaining two sections of this paper are dedicated to proving Theorems 2.1 and 3.3.

Acknowledgements. This work was partially supported by NSERC grant RGPIN-2018-04350.

2. Grid Knots and Linking Number

To emphasize the 3D nature of knots, we think of them as yarn ball knots, instead of as pancake knots. An equivalent notion is "grid knots" (see also [BL12]). A *grid knot (or link) of size* L is a labeled parametrized knot or link embedded as a subset of a grid with side length L. The arcs of a grid knot are enumerated in order of the parametrization of the knot along the unit grid segments of the grid lines. For a grid link, the link components are enumerated, and each arc of the link is labeled by a pair (c, p), where c is the arc's component enumeration, and p is the arc's parametrization enumeration. Some examples of grid knots are shown in Figure 2, and Figure 3 (A) and (B) shows an example of a $4 \times 4 \times 4$ grid from different perspectives.

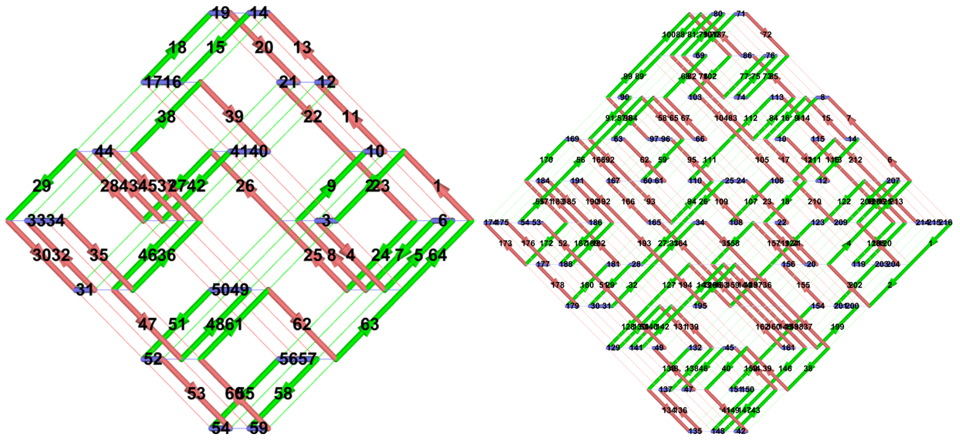


Figure 2. Two examples of grid knots. The left grid knot has $L = 3$ and 64 labeled arcs, and the right has $L = 5$ and 216 labeled arcs.

The process of converting an oriented yarn ball knot of length/volume V to a grid knot is as follows. Replace the yarn by an approximation along grid lines with grid spacing say $\frac{1}{10}$'th the unit width of the yarn. Rescale so that the grid squares are unit length again. Starting at any corner of the grid knot, label the arcs of the knot in order according to the orientation. The resulting knot is bounded in a box of size $\sim 10^3 V$, and this process takes $\sim V$ computation steps. To convert a grid knot to a yarn ball knot, scale the grid so the

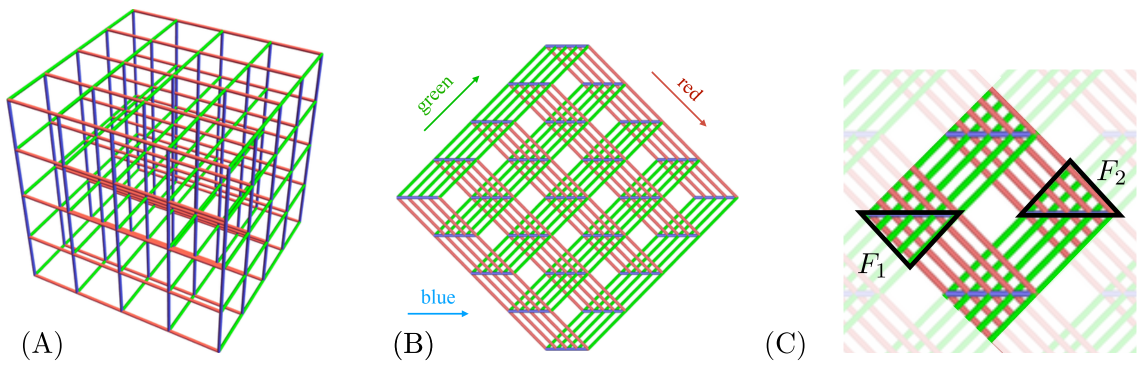


FIGURE 3. The grid in (B) shows a slightly askew top down view of the grid from (A). (C) highlights two crossing fields, F_1 and F_2, of a grid.

distance between neighbouring grid points is say 3 or 5 units. Replace the arcs of the knot with yarn of width 1 and round out the corners. This process takes time proportional to the length of the knot. When computing an invariant of a yarn ball knot, first converting the knot into a grid knot adds a negligible amount of computation time.

For the remainder of this paper, we conventionally view grids with the slightly askew top-down view as in Figure 3 (B). From this perspective, all of the crossings of a grid knot occur in triangular *crossing fields* of the grid– highlighted in Figure 3 (C). The grid lines in the x, y directions are colored in green and red, and the grid lines in the vertical direction are colored blue. We will keep the convention that "/" lines are called "green", "\" lines are called "red", and "–" lines are called "blue". All of the crossings in a crossing field occur between green and red grid lines. The vertical blue grid lines never participate in a crossing.

There are $2L^2$ triangular crossing fields; 2 at each $(L-1)^2$ interior corners, and one along each exterior corners except two corners, which is $2L + 2(L-1)$, for a total of $2(L-1)^2 + 2L + 2(L-1) = 2L^2$.

2.1. **Linking Number.** For a two-component link $\mathcal{L}$, the *linking number* of $\mathcal{L}$, denoted $lk(\mathcal{L})$, is a classical link invariant that measures how the two components are linked. From a planar projection of $\mathcal{L}$, $lk(\mathcal{L})$ can be computed as follows: Only counting "mixed" crossings that involve both components (the over strand is from one component and the under strand is from the other component), $lk(\mathcal{L})$ is one half the number of positive crossings minus the number of negative crossings. Using this 2D method for a planar diagram with n crossings, computing lk requires $\sim n$ steps– one for every crossing– which shows that $C_{lk}(2D, n) \sim n$. Using grid links, we provide a 3D algorithm that computes lk in time $\sim V$, and since $n = V^{\frac{4}{3}} \gg V$, this proves that lk is *C3D*.

Theorem 2.1. *$C_{lk}(3D, V) \sim V$, while $C_{lk}(2D, n) \sim n$, so lk is C3D.*

Proof. It is clear that $C_{lk}(3D, V) \gtrsim V$ and $C_{lk}(2D, n) \gtrsim n$: if that wasn't the case, it would mean that lk could be computed without looking at parts of the yarn (in the 3D case, for the length of the yarn is $\sim V$) or at some of the crossings (in the 2D case, for there are n crossings). But this is absurd: changing any crossing could change the linking number, and likewise, slightly moving any piece of yarn. Also, the standard "sum of signs over crossings" formula for lk shows that $C_{lk}(2D, n) \lesssim n$.

So we only need to prove that $C_{lk}(3D, V) \lesssim V$. Let $\mathcal{L}$ be a grid link with size L and volume $V = L^3$. There are 8 possible "mixed" crossing types in $\mathcal{L}$ according to the orientations of the two strands, whether the green or red strand crosses on top.

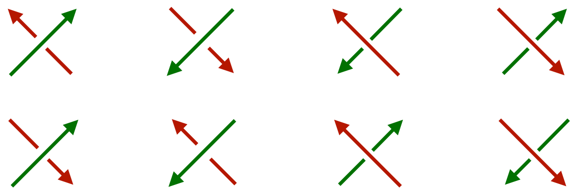


FIGURE 4. The 8 crossing types. For later use we label these crossings x_1 through x_8.

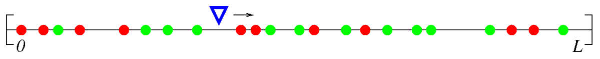
The linking number is computed by counting how many crossings of each of the 8 types are in the grid knot, adding these numbers together (with signs) and dividing by 2. We further break this down by counting how many crossings of each type occur in a single crossing field, then sum over all the crossing fields.

We show here how to count instances of the first crossing listed in the diagram above where we assume the green strand comes from the link component labeled 1, and the red strand from the link component labeled 2.

The other seven cases are counted similarly and with the same computational complexity.

Fix a crossing field F_k and let G be a subset of the green arcs of the grid link labeled $(1, p)$, for any p, in F_k and let R be a subset of the red edges of the grid link in F_k labeled $(2, p)$, for any p. For any such crossing field F_k, we can define a height map on the sets R and G, denoted $z : R, G \to [0, L]$, which gives the vertical strand height in F_k.

Within the fixed F_k, to count the number of times a green strand crossed over a red strand, we need to count the number of pairs $\{(r, g) \in R \times G : z(r) < z(g)\}$. To do this, line up the elements of R and G in increasing order based on their z-value, as shown below.



If there is a pair (r, g) of $R \times G$ such that $z(r) = z(g)$, there is a convention for which order to place the dots on the line. If we are counting greens over red, put the green dot first, and red dot second, as shown in Figure 5. If we count red crossing over green, put the red dot first and green dot second. Start with $rb = cf = 0$ ("reds before" and "cases found") and with the place holder ∇ before the leftmost dot. Slide ∇ from left to right, incrementing rb by one each time ∇ crosses a red dot and incrementing cf by rb each time ∇ crosses a green dot. The value of cf is the desired number of pairs that we wished to count. An example of the computation is shown in Figure 5.

For a fixed crossing field F_k, this computation can be carried out in time $\sim L$. There are $\sim L^2$ crossing fields and 8 types of colored orientation crossing types, which yields an overall computation time of $\sim L^3 = V$. □

3. FINITE TYPE INVARIANTS

A *Gauss diagram* of an n-crossing knot diagram parametrized by an interval I is given by the interval I along with n arrows. Each arrow corresponds to one of the n crossings and connects a point in the parameter space I which parametrizes the top of the crossing to a

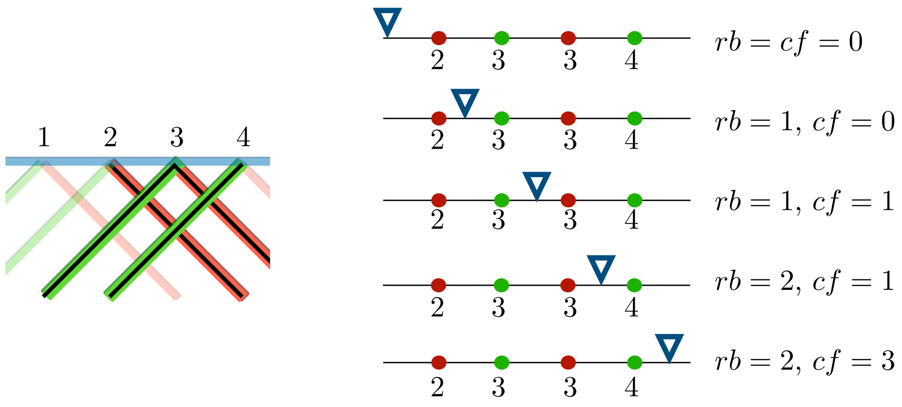


FIGURE 5. On the left is a portion of a grid knot passing through a crossing field where the green strands cross over the red strands. On the right, we show an example of the algorithm to count the number of crossings in this crossing field.

point which parametrizes the bottom of the crossing. Each arrow is also decorated with a sign corresponding to the sign of the crossing. We give an example of a Gauss diagram in Figure 6.

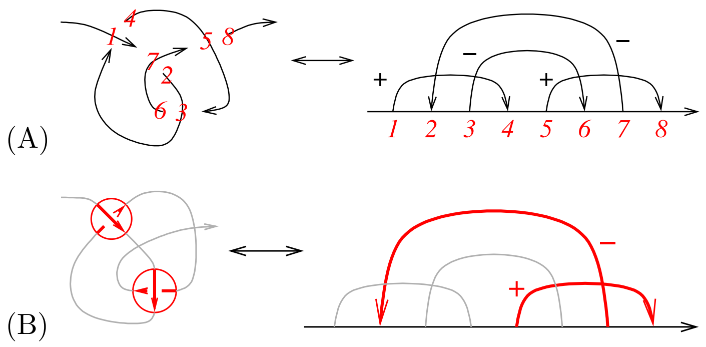


FIGURE 6. (A) An example of the Gauss diagram of a tangle diagram. (B) A 2-arrow subdiagram of a Gauss diagram.

A *d-arrow subdiagram* of a Gauss diagram D is a Gauss diagram consisting of a subset of d arrows from D. This subdiagram corresponds to a choice of d crossings in the knot diagram represented by the Gauss diagram. An example is shown in Figure 6. The space $GD = \langle \text{Gauss diagrams} \rangle$ is the $\mathbb{Q}$-vector space of all formal linear combinations of Gauss diagrams. We will denote the subspace of Gauss diagrams with d or fewer arrows by $GD_d = \langle \text{Gauss diagrams} \rangle_d$.

Gauss diagrams play an important role in the theory of finite type, or Vassiliev, invariants [Vas90, Vas92]. Any knot invariant V taking numerical values can be extended to an invariant of knots with finitely many double points (i.e. immersed circles whose only singularities are transverse self-intersections), using the following locally defined equation:

$$V(\times) = V(\nearrow\!\!\!\nwarrow) - V(\nwarrow\!\!\!\nearrow)$$

The above should be interpreted in the setting of knot diagrams which coincide outside of the given crossing. A knot invariant is a *finite type invariant of type d* if it vanishes on all knot diagrams with at least $d + 1$ double points [BL93, BN95a].

Let $\varphi_d : \{\text{knot diagrams}\} \to GD_d$ be the map that sends a knot diagram to the sum of all of the subdiagrams of its Gauss diagram which have at most d arrows. A Gauss diagram with n arrows has $\sum_{i=1}^{d} \binom{n}{i}$ subdiagrams with d or fewer arrows. Because $\binom{n}{i} \sim n^i$, a Gauss diagram with n arrows has $\sum_{i=1}^{d} \binom{n}{i} \sim \sum_{i=1}^{d} n^i \sim n^d$ subdiagrams with d or fewer arrows. So φ_d evaluated at a knot diagram with n crossings will be a sum of $\sim n^d$ subdiagrams. The map φ_d is *not* an invariant of knots, but, every finite type invariant factors through φ_d, as described in the following theorem.

Theorem 3.1 (Goussarov-Polyak-Viro [GPV00], see also [Rou]). *A knot invariant ζ is of type d if and only if there is a linear functional ω on GD_d such that $\zeta = \omega \circ \varphi_d$.*

A corollary of Theorem 3.1 is that any type d invariant can be computed from an n-crossing planar diagram D in the time that it takes to inspect all $\binom{n}{d}$ size d subdiagrams of D for the purpose of computing φ_d:

Theorem 3.2. *(see also [BN95b]) If ζ is a finite type invariant of type d then $C_\zeta(2D, n)$ is at most $\sim n^d$.* □

Next, we prove that in fact finite type invariants can be computed more efficiently from a 3D presentation:

Theorem 3.3. *If ζ is a finite type invariant of type d then $C_\zeta(3D, V)$ is at most V^d.*

Proof. Let ζ be a finite type invariant of type d and let K be a grid knot with side lengths L viewed as a diagram from the top-down perspective as in Figure 3. By Theorem 3.1, $\zeta(K) = \omega \circ \varphi_d(K)$, for some linear functional ω. Since GD_d is a fixed finite-dimensional vector space, the complexity of computing ω does not depend on K or V.

Thus, to prove the theorem, it suffices to show that $\varphi_d(K)$ can be computed in time V^d.

By definition, $\varphi_d(K) = \sum c_D D$ where D ranges over all possible Gauss diagrams with at most d arrows, and c_D is the number of times D occurs as a subdiagram in the Gauss diagram for K. The diagram D has arrows decorated with a $\pm$ sign corresponding to $\pm$ crossings in K. In a grid knot, there are 8 different realizations of oriented crossings depending on the colorings of the strands, which are shown in Figure 4.

To compute the coefficient c_D, we need to count the number of times D occurs as a subdiagram of K. Since K is a grid knot, we will subdivide this count by further specifying what type of $\pm$ crossing is associated to each $\pm$ arrow in D. To do this, we need to consider a more detailed labeling of Gauss diagrams. Let $LGD_d = \langle\text{Labeled Gauss Diagrams}\rangle_d$ be the space of Gauss diagrams with at most d arrows where each arrow is decorated with a label in $\{x_1, \cdots, x_8\}$. These labels will denote the *crossing type of an arrow*. LGD_d is large but finite dimensional, and for each diagram $D \in GD_d$ with ℓ arrows, there are 8^ℓ related diagrams $D_\chi \in LGD_d$ where $\chi \in \{x_1, \cdots, x_8\}^\ell$ is a sequence specifying the labelling of the arrows of D_χ. Thus, c_D can be computed by

$$c_D = \sum_{\chi} c_{D_\chi}$$

where χ ranges over all sequences in $\{x_1, \cdots x_8\}^\ell$. The coefficient c_{D_χ} is the number of times D_χ occurs as a subdiagram of K where an arrow j in D_χ with label χ_j corresponds to a crossing in K of type χ_j.

To compute $\varphi_d(K)$ it suffices to compute c_{D_χ} for every $D_\chi \in LGD_d$. The following argument will compute c_{D_χ} and will be repeated for every $D_\chi \in LGD_d$. Such repetition will not contribute to the complexity of φ_d up to $\sim$ as LGD_d is a fixed finite dimensional space independent of K.

Let $D_\chi \in LGD_d$ be a labeled Gauss diagram with $\ell \leq d$ arrows. The most computationally difficult case is when $\ell = d$, so we will count instances of diagrams with $\ell = d$ and all other counts will be similar and easier. We count all instances of D_χ that fall into specific crossing fields of the grid knot K.

Number each arrow of D_χ with $j \in \{1, \cdots, d\}$, in the order in which the arrows first occur from left to right in D_χ. As a labeled Gauss diagram, each arrow j of D_χ is also decorated with $\chi_j \in \{x_1, \cdots, x_8\}$ which specifies the crossing type associated to arrow j. We also label the ends of the arrows. Following the example in Figure 7, label the head of arrow j with $\alpha(j) \in \{1, \cdots, 2d\}$ and each tail by $\beta(j) \in \{1, \cdots, 2d\}$ in increasing order according to the parametrization. We will count the occurrences of D_χ in K by scanning the grid knot in the following way:

1. We consider all possible d-tuples $\overline{F} = (F_{k_1}, \ldots, F_{k_d})$ of crossing fields in K. For such a tuple, the jth arrow in D_χ gets assigned the crossing field denoted F_{k_j}. This choice specifies that the arrow j corresponds to a crossing of type χ_j which occurs in the crossing field F_{k_j} in the grid knot diagram for K (if such a crossing exists in F_{k_j}, see Example 3.4). There are $\sim L^2$ choices of crossing fields for each of the d arrows, so there $\sim L^{2d}$ possible choices of crossing field tuples for D_χ.

2. For any d-tuple $\overline{F}$ of crossing fields, we also assign to each arrow of D_χ a pair of sets: B_i, associated to the head, and $B_{i'}$ associated to the tail of the arrow. The index i of B_i will not necessarily match the label j of the arrow, but rather the labelling of the B_i's is according to the order in which the ends of the arrows appear along the parametrization, as in Figure 7. Thus, the set B_i associated to the head of arrow j will have $i = \alpha(j)$, and the $B_{i'}$ associated to the tail of arrow j will have $i' = \beta(j)$.

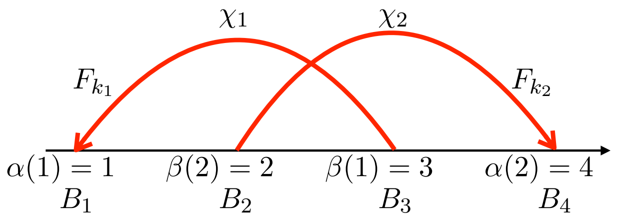


FIGURE 7. An example of a labeled Gauss diagram with $d = 2$.

The sets B_i and $B_{i'}$ for arrow j will depend on both the crossing field F_{k_j} and the crossing type χ_j of the arrow. The crossing type χ_j determines whether we count green strands on top of red, or red on top of green, and which orientation of the strands to count. In general, B_i will be the set of strands of the knot in F_{k_j} that have the same color and orientation as the under strand in crossing type χ_j. The set $B_{i'}$ will be the strands of the knot in F_{k_j} that have the same color and orientation as the over strands in crossing type χ_j.

Example 3.4. Suppose $\chi_j = x_1$ and refer to Figure 8. The crossing type x_1 determines we are counting green and red arrows both oriented upwards, with green on top. We look to the crossing field F_{k_j} of our knot diagram for K. In F_{k_j}, if all of the red strands cross on top, then both B_i and $B_{i'}$ are empty, as the crossings are not compatible with the x_1 type crossing. If the green strands in F_{k_j} cross on top, then the set B_i, associated to the head of arrow j, will be all of the red strands of the knot in F_{k_j} oriented upwards. The set $B_{i'}$,

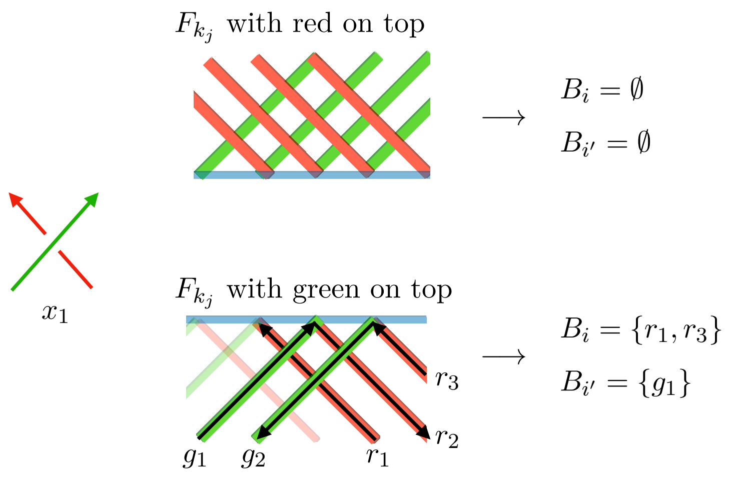


FIGURE 8. An example of computing the sets B_i and $B_{i'}$ for a crossing of type $\chi_i = x_1$ and two different crossing fields.

associated to the tail of arrow j, will be all of the green strands of the knot in F_{k_j} oriented upwards.

3. With the crossing fields chosen and the B_i's assigned, we look for instances of D_χ within the grid knot, by choosing a strand b_i in each B_i and checking if the collection of chosen strands $\{b_1, \cdots, b_{2d}\}$ is compatible with the parametrization and gives rise to exactly d crossings in K that recover the diagram D_χ. To check this compatibility, there are two functions defined on the sets B_i.

- $t : \cup B_i \to \mathbb{Z}$ gives the order in which the arrow endpoints labeled by the B_i's occur in the parametrization of the grid diagram for K.
- $z : \cup B_i \to \{0, \cdots, L\}$ gives the vertical height of the strand in the grid knot. Notice that for each i, the elements of B_i are strands of the same color inside a single crossing field, all of which have distinct heights. So $z|_{B_i}$ is injective for every i.

First, we need the crossings to occur in the correct order along the grid knot according to the parametrization. This means we need $t(b_1) < t(b_2) < ... < t(b_{2d})$. Second, we need to ensure that the chosen strands for a given arrow in the diagram actually cross. For an arrow j, $b_{\beta(j)}$ needs to be the over strand of the crossing and $b_{\alpha(j)}$ is the under strand. To get that $b_{\beta(j)}$ crosses over $b_{\alpha(j)}$, we need the height of $b_{\alpha(j)}$ to be lower than the height of $b_{\beta(j)}$, i.e. $z(b_{\alpha(j)}) < z(b_{\beta(j)})$. Notice that $b_{\alpha(j)}$ and $b_{\beta(j)}$ can be elements from different B_i's.

This problem reduces to the following counting problem: Given $\alpha(j), \beta(j) \in \{1, \cdots, 2d\}$ for $j \in \{1, \cdots, d\}$, the $2d$ sets B_i with $i \in \{1, \cdots, 2d\}$ and functions $t : \cup B_i \to \mathbb{Z}$ and $z : \cup B_i \to \{0, \cdots, L\}$ such that $z|_{B_i}$ and t are injective, we want to compute $|A|$ where

$$A = \left\{ b \in (b_i)_{i=1}^{2d} \in \textstyle\prod_i B_i \;\middle|\; \begin{array}{c} t(b_1) < t(b_2) < \cdots < t(b_{2d}), \\ \forall j \in \{1, \cdots, d\}, z(b_{\alpha(j)}) < z(b_{\beta(j)}) \end{array} \right\}.$$

In Section 4 we will prove Proposition 4.2, which asserts that this computation can be carried out in time L^d.

Since we had $\sim L^{2d}$ choices of d-tuples of crossing fields, a large but finite constant number of labeled Gauss diagrams with d or fewer arrows, and for each choice we have L^d computations to find instances of this diagram in the knot, we get a total computation time of $\sim L^{2d}L^d = V^d$ as claimed. □

4. Combinatorial Results

In this section, we prove Proposition 4.2, which was used in the proof of Theorem 3.3. We start with the following lemma.

Lemma 4.1. *Suppose we have sets B_i, for $i \in \{1, \cdots, 2d\}$, and a map $t : \cup B_i \to \mathbb{N}$ such that $t|_{B_i}$ is injective for all i. Let $K \coloneqq \max(|B_i|)$. Then the quantity*

$$N = \left|\left\{ b = (b_i)_{i=1}^{2d} \in \prod_{i=1}^{2d} B_i : t(b_1) < t(b_2) < \cdots < t(b_{2d}) \right\}\right|$$

can be computed in time $\sim K$.

Proof. In time $\sim K$ each of the B_i's can be sorted by the values of t on it and replaced by its indexing interval. So without loss of generality, each B_i is just a list of integers $\{1, \ldots, |B_i|\}$, the function t is replaced by increasing functions $t_i \colon \{1, \ldots, |B_i|\} \to \mathbb{N}$, and we need to count

$$N = \left|\left\{ b = (b_i)_{i=1}^{2d} \in \prod_{i=1}^{2d} \{1, \ldots, |B_i|\} : t_1(b_1) < t_2(b_2) < \cdots < t_{2d}(b_{2d}) \right\}\right|$$

For $1 \le \iota \le 2d$ and $1 \le \tau \le |B_\iota|$, let

$$N_{\iota,\tau} = \left|\left\{ b = (b_i)_{i=1}^{\iota} \in \prod_{i=1}^{\iota} \{1, \ldots, |B_i|\} : t_1(b_1) < t_2(b_2) < \cdots < t_\iota(b_\iota) \le t_\iota(\tau) \right\}\right|,$$

i.e. $N_{\iota,\tau}$ is the number of sequences of length ι with increasing t-values that terminate at a value less than or equal to $t_\iota(\tau)$. Also set $N_{\iota,0} = 0$ for all ι. Then clearly $N = N_{2d,|B_{2d}|}$, $N_{1,\tau} = \tau$ for all τ and for $\iota > 1$,

$$N_{\iota,\tau} = N_{\iota,\tau-1} + N_{\iota-1,\tau'},$$

where $\tau' = \max(\{0\} \cup \{\tau'' \colon t_{\iota-1}(\tau'') < t_\iota(\tau)\})$. Note that τ' can be computed in time $\log K \sim 1$ and hence the $N_{\iota,-}$'s can be computed from the $N_{\iota-1,-}$'s in time $\sim K$. To find the $N_{2d,-}$'s and hence N this process needs to be repeated $2d \sim 1$ times, and the overall computation time remains $\sim K$. $\square$

Proposition 4.2. *Given a collection of $2d$ sets B_i with $i \in \{1, \cdots, 2d\}$, and functions $\alpha, \beta : \{1, \cdots, d\} \to \{1, \cdots, 2d\}$, $t : \cup B_i \to \mathbb{Z}$ and $z : \cup B_i \to \{0, \cdots, L\}$ such that $im(\alpha) \cup im(\beta) = \{1, \ldots, 2d\}$, and $z|_{B_i}$ and $t|_{B_i}$ are injective, the size of the following set can be computed in time $\sim L^d$,*

$$A = \left\{ b \in (b_i)_{i=1}^{2d} \in \prod B_i \;\middle|\; \begin{array}{c} t(b_1) < t(b_2) < \cdots < t(b_{2d}), \\ \forall j \in \{1, \cdots, d\},\ z(b_{\alpha(j)}) < z(b_{\beta(j)}) \end{array} \right\}.$$

Proof. Lemma 4.1 shows us how to count elements in the set A without the conditions on z. We will show that $|A|$ can be computed by writing A as a union of sets with the t-conditions and *one Cartesian condition*, so that we can apply Lemma 4.1 to count elements in each set of the union.

Let $p \in \mathbb{N}$ so that $2^{p-1} < L \le 2^p$. Every z-value is in $\{0, \cdots, L\}$ and can be written as a binary expansion with exactly p binary digits, padding with zeros in front if needed. For example, if $p = 5$ the number 2 can be written as 00010. For two binary numbers z_1 and z_2 in this form, $z_1 < z_2$ if they have the same binary expansions from left to right up to a point, and in the first place they differ z_1 has a 0 and z_2 has a 1. The notation we use to describe this is as follows. Let σ be the binary sequence from left to right which z_1 and z_2 share in common. We write $z_1 = \sigma 0*$ to mean the binary expansion of z_1 read left to right is σ followed by 0 followed by an arbitrary remainder of 0's and 1's. Similarly, we write $z_2 = \sigma 1*$.

For each $j \in \{1, \cdots, d\}$, if the condition $z(b_{\alpha(j)}) < z(b_{\beta(j)})$ is satisfied, there exists a binary sequence σ_j of length $|\sigma_j| < p$ so that $z(b_{\alpha(j)}) = \sigma_j 0*$ and $z(b_{\beta(j)}) = \sigma_j 1*$. This binary structure is key to construct the desired *Cartesian* conditions to compute $|A|$.

For each $j \in \{1, \cdots, d\}$, the relation $z(b_{\alpha(j)}) < z(b_{\beta(j)})$ requires the pair $(z(b_{\alpha(j)}), z(b_{\beta(j)}))$ to be in the triangle below the diagonal shown in Figure 9. Notice that in a single pair, $(z(b_{\alpha(j)}), z(b_{\beta(j)}))$, the $b_{\alpha(j)}$ and $b_{\beta(j)}$ are not necessarily elements of the same B_i. Since the outputs of z are integral, we can divide the triangle into a finite number of squares below the diagonal. The sides of the squares in the triangle are labeled by binary sequences of length less than or equal to p. Each square is determined by all but the last entry in the sequences labeled on the right and bottom sides of the square.

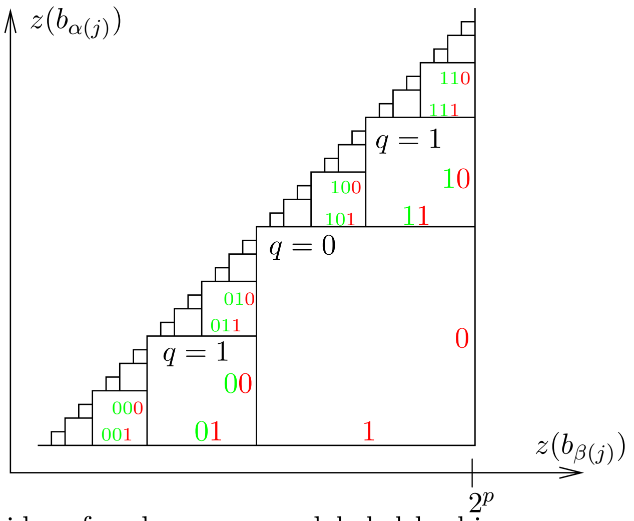


FIGURE 9. The sides of each square are labeled by binary sequences which describe the set of integral pairs within the square. For example, the square with south side labeled by 11 and east side labeled by 10 contains pairs of the form $(z(b_{\alpha(j)}), z(b_{\beta(j)}))$ with $z(b_{\alpha(j)}) = 1 \cdot 2^{p-1} + 0 \cdot 2^{p-2} + \cdots$ and $z(b_{\beta(j)}) = 1 \cdot 2^{p-1} + 1 \cdot 2^{p-2} + \cdots$.

Since j can range from 1 to d, there are d relations $z(b_{\alpha(j)}) < z(b_{\beta(j)})$ that must simultaneously hold. So, we consider d copies of the above triangle, one for each j. We will write A as a union of sets with the t-conditions and one Cartesian condition corresponding to a specific square in each of the triangles.

The collection of squares on a subdiagonal all have the same size and are labeled by binary sequences of the same length. To each subdiagonal, we can associate the q-value

$$q = (\text{the length of the binary sequences on the subdiagonal}) - 1.$$

The subdiagonals with q-values 0 and 1 are shown in Figure 9.

Let $\bar{\sigma} = (\sigma_j)_{j=1}^d$ be a d-tuple of binary sequences, where the length of σ_j is $|\sigma_j| = q_j \in \{0, \cdots, p-1\}$. As demonstrated in Figure 10, the coordinates of $\bar{\sigma}$ pick out a specific square in each of the d triangles, (σ_j is a binary sequence that labels a square in triangle j).

The length of σ_j, q_j, is the q-value for the subdiagonal that contains the square labeled by σ_j. The coordinate $(z(b_{\alpha(j)}), z(b_{\beta(j)}))$ is in the square labeled by σ_j if $z(b_{\alpha(j)}) = \sigma_j 0*$ and $z(b_{\beta(j)}) = \sigma_j 1*$.

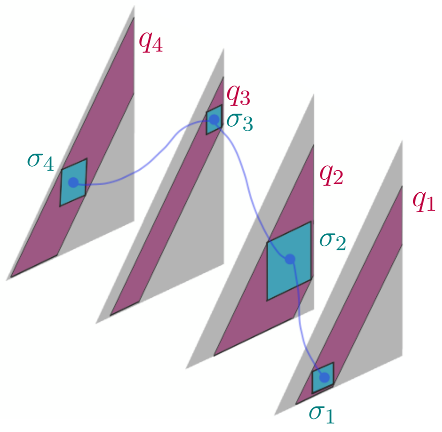


FIGURE 10. A schematic diagram of $\bar{\sigma} = (\sigma_1, \cdots, \sigma_4)$ and $\bar{q} = (q_1, \cdots, q_4)$. The coordinate σ_j specifies a square in jth triangle, on the subdiagonal corresponding to $|\sigma_j| = q_j$.

For a given $\bar{\sigma} = (\sigma_j)_{j=1}^d$, we can define $A_{\bar{\sigma}}$, which collects all the choices of $2d$-tuples of b_i's that satisfy the t-conditions and for each j satisfy the jth z-condition inside the square σ_j.

$$A_{\bar{\sigma}} = \left\{ \bar{b} = (b_i)_{i=1}^{2d} \in \prod B_i \;\middle|\; \begin{array}{c} t(b_1) < t(b_2) < \cdots < t(b_{2d}), \\ \forall j \in \{1, \cdots, d\}, z(b_{\alpha(j)}) = \sigma_j 0* \\ \text{and} \;\; z(b_{\beta(j)}) = \sigma_j 1* \end{array} \right\}. \tag{1}$$

If we let $\bar{q} = (q_j)_{j=1}^d \in \{0, \cdots, p-1\}^d$, we can consider all $\bar{\sigma}$'s so that the jth coordinate in $\bar{\sigma}$ has length equal to the jth entry in $\bar{q}$, or $|\sigma_j| = q_j$. Then for we can define

$$A_{\bar{q}} = \bigcup_{\bar{\sigma}: \forall j, |\sigma_j| = q_j} A_{\bar{\sigma}}, \tag{2}$$

which collects all the choices of b_i that satisfy the t-conditions and satisfy the jth z-condition inside the diagonal q_j. Finally, collecting up all possible $A_{\bar{q}}$'s gathers every possible desired choice of b_i's that satisfy the t-conditions and the z-conditions, which is the desired set A,

$$A = \bigcup_{\bar{q} \in \{0, \cdots, p-1\}^d} A_{\bar{q}}. \tag{3}$$

In essence, we break A into pieces by specifying that the z-condition must be satisfied on certain diagonals, the sets $A_{\bar{q}}$. Then we further break down $A_{\bar{q}}$ by specifying that the z-condition must be satisfied in specific squares within the diagonals $\bar{q}$, the sets $A_{\bar{\sigma}}$.

To compute the size of these sets, first notice from Equation 1 that $A_{\bar{\sigma}}$ can be written as the set N from Lemma 4.1 with the sets B_i replaced by subsets $B_i' \subseteq B_i$, where

$$B_i' = \left\{ b \in B_i \;\middle|\; \begin{array}{l} \text{if} \;\; \alpha(j) = i, \; \text{then} \; z(b) = \sigma_j 0* \\ \text{if} \;\; \beta(j) = i, \; \text{then} \; z(b) = \sigma_j 1* \end{array} \right\},$$

and where $j \in \{1, \ldots, d\}$ is such that either $\alpha(j) = i$ or $\beta(j) = i$ (exactly one such j exists and exactly one of the conditions is met as $im(\alpha) \cup im(\beta) = \{1, \ldots, 2d\}$).

We are given that $z|_{B_i}$ is injective, so $|z(B_i)| = |B_i| \sim L \sim 2^p$. The elements of $z(B_i)$ are binary sequences of length p. In the new subset $z(B_i')$, we are constraining the first

$|\sigma_j|+1 = q_j+1$ entries of the sequences, so there are $p-(q_j+1)$ free digits in each element of $z(B'_i)$. With this, we can approximate the size of B'_i,

$$|B'_i| = |z(B'_i)| \le 2^{p-(q_j+1)} = \frac{2^p}{2^{q_j+1}} \sim \frac{2^p}{2^{q_j}} \sim \frac{L}{2^{q_j}} \le \frac{L}{2^{min(q_j)}}.$$

Using Lemma 4.1 we can compute $|A_{\bar\sigma}|$ in time $\sim \max|B'_i| \sim \frac{L}{2^{min(q_j)}}$. In Equation 2, $A_{\bar q}$ is written as a union of $A_{\bar\sigma}$'s where there are $2^{\sum q_i}$ possible choices for $\bar\sigma$. So, $|A_{\bar q}|$ can be computed in time $\sim 2^{\sum q_i}\left(\frac{L}{2^{min(q_j)}}\right) = 2^{\sum' q_i}L$, where $\sum'$ denotes "sum with the smallest summand omitted". The worst case is when in $\bar q = (q_j)_{j=1}^d$ all but one of the entries are $p-1$ and one (the one omitted in $\sum'$) is unconstrained, and in that case the complexity is $(2^{p-1})^{d-1}L \sim L^d$.

Lastly, from Equation 3, A is the union of $A_{\bar q}$'s where the number of choices for $\bar q$ is $p^d \sim (\log_2 L)^d \sim 1$. So up to $\sim$, we can compute $|A|$ with at most the complexity of the most expensive $|A_{\bar q}|$, which is $\sim L^d$. $\square$

References

[BL93] J. S. Birman and X.-S. Lin. Knot polynomials and vassiliev's invariants. *Inventiones Mathematicae*, 111:225–270, 1993. 3, 7

[BL12] S. Baldridge and A. Lowrance. Cube diagrams and 3-dimensional reidemeister-like moves for knots. *Journal of Knot Theory and Its Ramifications*, 21:1–39, 2012. 4

[BN95a] D. Bar-Natan. On the vassiliev knot invariants. *Topology*, 34:423–472, 1995. 3, 7

[BN95b] D. Bar-Natan. Polynomial invariants are polynomials. *Mathematical Research Letters*, 2:239–246, 1995. 8

[BNBNHS] D. Bar-Natan, I. Bar-Natan, I. Halacheva, and N. Scherich. Yarn ball knots and even faster computations. In preparation. 3

[GPV00] M. Goussarov, M. Polyak, and O. Viro. Finite type invariants of classical and virtual knots. *Topology*, 39(5):1045–1068, 2000. 8

[Jon85] V. F. R. Jones. A polynomial invariant for knots via von neumann algebras. *Bulletin of the American Mathematical Society*, 12:103–111, 1985. 1

[Kau90] L. Kauffman. An invariant of regular isotopy. *Transactions of the American Mathematical Society*, 312:417–471, 1990. 1

[Pic20] L. Piccirillo. The conway knot is not slice. *Annals of Mathematics*, 191(2):581–591, 2020. 2

[Rou] F. Roukema. Goussarov-polyak-viro combinatorial formulas for finite type invariants. 8

[Vas90] V. A. Vassiliev. *Cohomology of Knot Spaces.* American Mathematical Society, Providence, RI, 1990. 7

[Vas92] V.A. Vassiliev. *Complements of Discriminants of Smooth Maps: Topology and Applications*, volume 98 of *Translations of Mathematical Monographs.* American Mathematical Society, Providence, RI, 1992. 7

[Wit89] E. Witten. Quantum field theory and the jones polynomial. *Communications in Mathematical Physics*, 121:351–399, 1989. 1

University of Toronto
Email address: drorbn@math.toronto.edu
URL: http://www.math.toronto.edu/drorbn

University of California, Los Angeles
Email address: itaibn@math.ucla.edu

Northeastern University
Email address: i.halacheva@northeastern.edu
URL: https://sites.google.com/site/ivahalacheva3/

University of Toronto
Email address: n.scherich@utoronto.ca
URL: http://www.nancyscherich.com