A COMPUTATIONAL SURVEY OF THE THETA INVARIANT

ALI KARIM LALANI

Abstract. This project explores the theta invariant, a powerful, genuinely-computable invariant for knots developed by Dror Bar-Natan and Roland van der Veen. We calculate this invariant for several families of knots, including torus, pretzel, and twisted torus knots, to search for the information it holds.

Keywords. Knots, Topology

Contents

1. Introduction and preliminaries

In this write-up, we explore a powerful and modern invariant called the theta invariant (Θ), developed by Dror Bar-Natan and Roland van der Veen.

The Θ invariant is a highly discerning and easy to compute tool that is believed to contain more information than the Jones Polynomial.

Our work is based on the foundational paper by Bar-Natan and van der Veen, *A very fast, very strong, topologically meaningful and fun knot invariant.* Readers seeking the full mathematical details are encouraged to read it. [BV1]

The purpose of this project is to apply the θ invariant to entire 'families' of knots to search for patterns, test its behavior, and form new conjectures. To accomplish this, we use the KnotTheory package in Mathematica to code up families like torus, pretzel, and twisted torus knots and compute θ.

1.1. **Computational Tools and Preliminaries.**

Before presenting our results, we briefly introduce the computational framework used in this project. All computations and visualizations were performed using the `KnotTheory` package in Mathematica. This package provides a environment for defining, manipulating, and computing invariants of knots. [KA]

1.2. **Knot Representations.**

Within the package, knots can be defined in several ways. The two most relevant for our work are:

- **Planar Diagram** (`PD`): This notation describes a knot by first labelling all the edges an then explicitly listing its crossings. Each crossing connects four points on the knot. For example, the trefoil knot 3_1 is represented as `PD[X[1,4,2,5], X[3,6,4,1], X[5,2,6,3]]`.
- **Braid Representation** (`BR`): This notation describes a knot as the closure of a braid on a certain number of strands. For instance, the trefoil knot can also be written as `BR[3, {1, 1, 1}]`. This works because every knot or link can be represented as a closed braid by Alexander's Theorem.

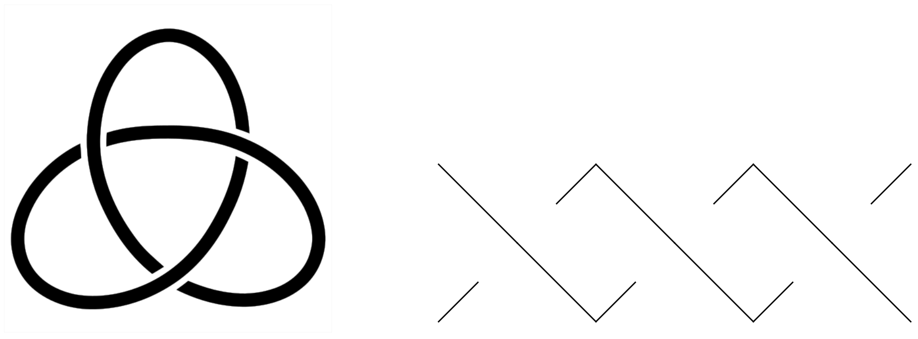


Figure 1. The trefoil knot 3_1 as a planar diagram (left) and as the closure of a 3-strand braid (right).

1.3. **Key Functions in the Theta Implementation.** The implementation of the Θ invariant is based on several helper functions and one main program. We outline the key components here, following the description in [BV1].

The main program, $\Theta[K\_]$, computes the invariant. The mathematical formula follows by first constructing a matrix A, inverting it to find G, and then summing the terms corresponding to the different parts of the formula.

We urge the reader to read the definitions of Rot and the helper polynomials in the original paper.

```
Θ[K_]:=Θ[K]=Module[{X,φ,n,A,Δ,G,ev,θ},
 (* 01 *) {X,φ}=Rot[K]; n=Length[X];
 (* 02 *) A=IdentityMatrix[2n+1];
 (* 03 *) Cases[X,{s_,i_,j_}:>(A⟦{i,j},{i+1,j+1}⟧+=( -T^s  T^s-1 ))];
                                                   (  0     -1   )
 (* 04 *) Δ=T^((-Total[φ]-Total[X⟦All,1⟧])/2) Det[A];
 (* 05 *) G=Inverse[A];
 (* 06 *) ev[ℰ_]:=Factor[ℰ/.g_{ν_,α_,β_}:>(G⟦α,β⟧/.T→T_ν)];
 (* 07 *) θ=ev[∑_{k=1}^{n} F_1[X⟦k⟧]];
 (* 08 *) θ+=ev[∑_{k1=1}^{n} ∑_{k2=1}^{n} F_2[X⟦k1⟧,X⟦k2⟧]];
 (* 09 *) θ+=ev[∑_{k=1}^{2n} F_3[φ⟦k⟧,k]];
 (* 10 *) Factor@{Δ,(Δ/.T→T_1)(Δ/.T→T_2)(Δ/.T→T_3)θ}
];
```

The θ component theoretically admits hexagonal symmetry

PolyPlot, **ParityPlot**, and **RainbowPlot** serve as compact "QR codes" for each Θ polynomial. All three plots place every non-zero Laurent coefficient at its exponent pair (i, j) on the triangular grid, so that any hexagonal symmetry is immediately visible.

- **PolyPlot** colors a point **red** when the coefficient is positive and **blue** when it is negative. Sign patterns and sign-symmetry violations therefore stand out at a glance.
- **ParityPlot** ignores the sign and instead colors coefficients **orange** when they are odd and **emerald** when they are even, revealing congruence phenomena that PolyPlot can conceal.
- **RainbowPlot** synthesizes the previous two ideas. Its *hue* encodes the sign–parity class (positive odd, positive even, negative odd, negative even), while the color's *lightness* scales logarithmically with $|c|$: small coefficients appear as washed-out pastels, large ones as fully saturated colors. Thus a single picture conveys magnitude, sign, and parity simultaneously—hence the name "rainbow".

Together, these plots provide a visual check on the expected symmetry of Θ and reveal family-specific patterns. Full plotting routines are available in the code repository referenced later.

1.4. **Some Computations with Theta.** We move on to examples. Here is Θ on the trefoil, and its corresponding ParityPlot and PolyPlot diagrams.

```
trefoil = Knot[3,1];
trefoilΘ = Θ[trefoil];
Expand[trefoilΘ]
PolyPlot[trefoilΘ]
ParityPlot[trefoilΘ]
```

Output

$$\left\{T+\frac{1}{T}-1,-T_2^2T_1^2+T_2T_1^2-T_1^2+T_2^2T_1+\frac{T_1}{T_2}-T_2^2-\frac{1}{T_1^2}+\frac{1}{T_1^2T_2}-\frac{1}{T_2^2}+\frac{1}{T_1T_2^2}-\frac{1}{T_1^2T_2^2}+\frac{T_2}{T_1}\right\}$$

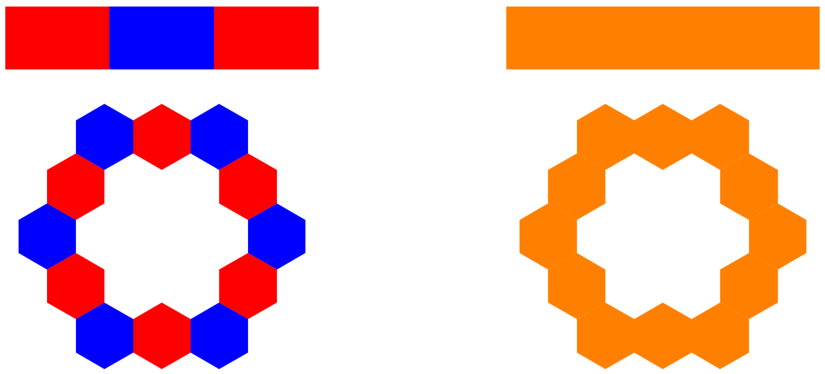


FIGURE 2. The PolyPlot and ParityPlot output of the trefoil knot.

Notice the hexagonal symmetry of the plots, and the ParityPlot is orange since all the coefficients of the second component are 1.

1.5. **Canonical PD Representations.** To manage knot definitions robustly, we implemented a technical function, `CanonicalPD`.

A given knot diagram can be represented by a `PD` object in many equivalent ways, depending on the arbitrary integer labels assigned to its edge segments. The `CanonicalPD` function mitigates this ambiguity by generating a unique, canonical representative for any given diagram. It operates by traversing the knot from every possible starting edge, relabeling the edges in the order they are visited. This process produces a set of all possible valid `PD` expressions for the diagram. The function then returns the first element from this set, ensuring that any two diagrams that are structurally identical,but differently labeled, yield the same

`PD` object. This allows us to define and compare knots based on their geometric structure.

The full implementation is provided in Appendix A.

2. Definitions of Knot Families

In this section, we formally define the families of knots investigated in this project. For each family, we provide a brief description and the corresponding implementation code.

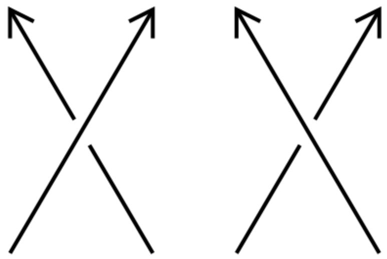


Figure 3. The convention for the positive (left) and negative (right) crossing X[a,,b,c,d].

2.1. **Torus Knots.**

Definition 2.1 (Torus Knot)**.** A **torus knot**, denoted $T(p,q)$ for coprime integers p and q, is a knot that lies on the surface of an unknotted torus in $\mathbb{R}^3$. It can be represented as the closure of a braid on p strands. The braid word consists of $|q|$ full twists, where the sign of q determines their handedness. This structure is illustrated in Figure 4.

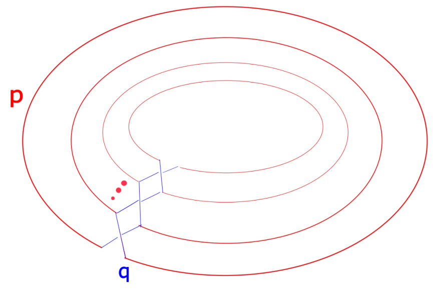


Figure 4. A diagram for $T(p,q)$.

Definition 2.2 (Twisted Torus Knot)**.** A **twisted torus knot**, denoted $T(p,q,r,s)$, is obtained by applying s additional full twists to r adjacent strands within the braid of a $T(p,q)$ torus knot.

Remark 2.3. It is easy to see Twisted Torus Knots subsume the $T(p,q)$ Torus Knots by setting s to be 0.

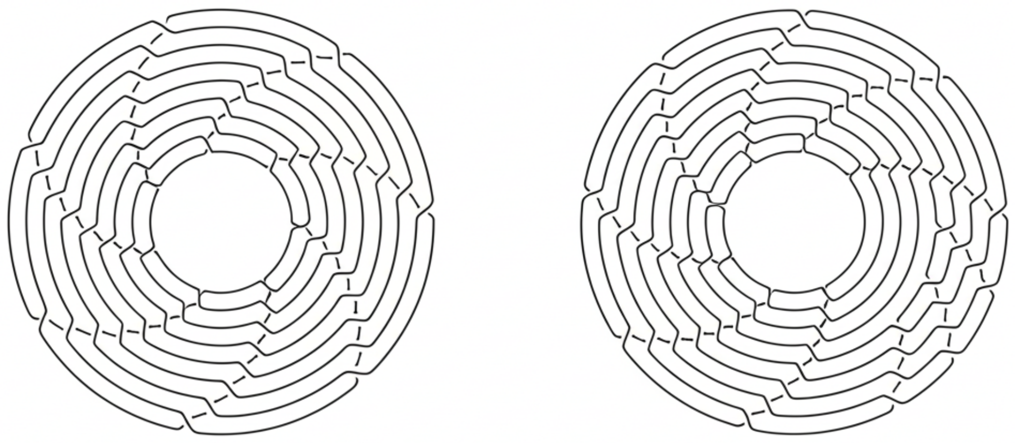


FIGURE 5. Diagrams of the regular $T(9,7)$ (left) and twisted torus knot $T(9,7,5,3)$ (right).

Here is an example.

The braid word β for $T(p,q,r,s)$ on p strands in the case where $r < p$ is given by [Lee17]:

$$\beta = (\sigma_1\sigma_2 \ldots \sigma_{p-1})^q(\sigma_1\sigma_2 \ldots \sigma_{r-1})^s \tag{2.1}$$

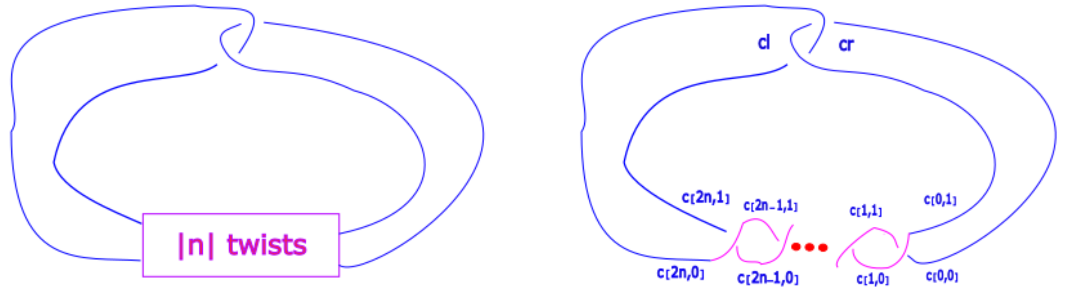
where σ_i are the standard braid group generators.

Here is the implementation in Mathematica:

```
(* Repeat the braid word `seq` |e| times. *)
twPower[seq_List, e_Integer] :=
  Flatten @ Table[
    If[e ≥ 0, seq, -Reverse[seq]],
    {Abs[e]}
  ];

(* Braid word for the twisted-torus knot T(p,q; r,s). *)
TwistedTorusBraidWord[
  p_Integer?Positive, q_Integer,
  r_Integer?Positive, s_Integer] /; 1 < r < p :=
 Module[{blockPQ, blockRS},
   blockPQ = twPower[Range[p - 1],  q]; (* (σ₁ … σ_{p-1})^q  *)
   blockRS = twPower[Range[r - 1], r s]; (* (σ₁ … σ_{r-1})^(r s) *)
   Join[blockPQ, blockRS]
 ];

(* Convert it into a Planar Diagram *)
TwistedTorusKnot[
  p_Integer?Positive, q_Integer,
  r_Integer?Positive, s_Integer] :=
  PD[BR[p, TwistedTorusBraidWord[p, q, r, s]]];
```

2.2. **Twist Knots.**

Definition 2.4 (Twist Knot). The family of **twist knots**, denoted K_n, is made by adding n full twists to the strands of a simple clasp. n can be positive or negative, determining the handedness of the crossings.



(A) The general form of a twist knot. (B) Edge labeling convention.

FIGURE 6. The general structure and labeling for a twist knot K_n.

Our implementation generates the planar diagram (PD) for a twist knot K_n with $n > 0$ twists by systematically constructing its geometric components, as labeled in Figure 6. The code defines the knot by assembling four lists of crossings:

(1) Two **clasp crossings** that form the upper loop of the knot.
(2) A series of n **twist crossings** that form the right-handed ”rungs” of the lower twisted section.
(3) A series of **connecting crossings** that form the ”rails” linking the twists together.

These lists are joined to form a complete PD object, which is then processed by our CanonicalPD function to produce a unique, standardized representation. An analogous procedure is followed for knots with negative twists ($n < 0$), where the structure of the twist crossings is mirrored to produce left-handed twists.

```
PositiveTwist[n_Integer?Positive] := Module[
  {crossing1, loop1, crossing2, loop2, finalpd},



crossing1 = {X[c[0, 0], c[2 n, 0], cl, cr]};
loop1 = Table[X[c[2 i, 1], c[2 i + 1, 1], c[2 i + 1, 0], c[2 i, 0]],
{i, 0, n - 1}];
crossing2 = {X[c[2 n, 1], c[0, 1], cr, cl]};
loop2 = Table[X[c[k, 0], c[k - 1, 0], c[k - 1, 1], c[k, 1]],
{k, 2 n, 2, -2} ];
finalpd = Join[crossing1, loop1, crossing2, loop2];

 CanonicalPD[PD @@ finalpd]
  ]
 NegativeTwist[n_Integer?Positive] := Module[
  {crossing1, loop1, crossing2, loop2, finalpd, nabs = Abs[n] },



crossing1 = {X[c[0, 0], c[2 n, 0], cl, cr]};
loop1 = Table[X[c[2 i + 1, 0], c[2 i + 1, 1], c[2 i + 2, 1],
c[2 i + 2, 0]],{i, 0, nabs - 1}];
crossing2 = {X[c[2 n, 1], c[0, 1],cr,cl]};
loop2 = Table[X[c[k, 1], c[k, 0], c[k - 1, 0], c[k - 1, 1]],
{k, 2 nabs - 1, 1, -2}];

finalpd = Join[crossing1, loop1, crossing2, loop2];

 CanonicalPD[PD @@ finalpd]
  ]
```

```
TwistKnot[n_Integer] := Which[
    n > 0,  PositiveTwist[n],
    n < 0,  NegativeTwist[Abs[n]],
    n == 0, Loop[1] (* If n=0, return the unknot *)
]
```

2.3. **Pretzel Knots.**

Definition 2.5 (Pretzel Link). A **pretzel link**, denoted $P(p_1, p_2, \ldots, p_n)$, is formed by connecting n tangles in a circular arrangement. Each tangle i consists of p_i half-twists, with the sign of p_i determining the direction of the twists.

A pretzel link is a knot (a link with a single component) only under certain conditions on its parameters. We state a well-known proposition.

Proposition 2.6. *The pretzel link $P(p_1, p_2, \ldots, p_n)$ is a knot under the following conditions:*

(1) *If n is odd, the link is a knot if and only if the number of even parameters among the p_i is at most one.*
(2) *If n is even, the link is a knot if and only if exactly one of the parameters p_i is even.*

We generate the planar diagram by defining each crossing according to a geometric labeling. The code first defines a set of helper functions that permute the arguments of a crossing `X[a,b,c,d]` to handle the four possible crossing orientations. The choice of orientation for each crossing in a tangle depends on two

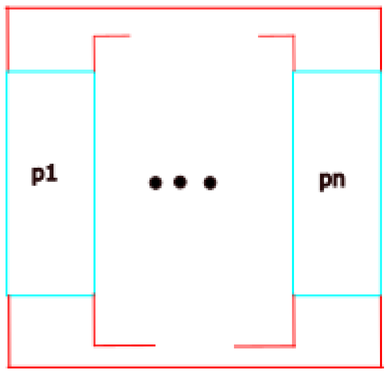


Figure 7. Diagram of the general pretzel knot.

factors: the sign of its parameter p_i (determining right- or left-handed twists) and the parity of the tangle's index i (determining how it weaves into the main loops).

The main function then assembles the knot by building three distinct lists of crossings:

(1) **Interior Crossings:** A list of crossings forming the internal "rungs" of each tangle. These are represented via lc (left columns) and rc (right columns) respectively.
(2) **Top & Bottom Crossings:** Two lists of crossings that form the junctions between the ends of each tangle and the main top (t_i) and bottom (b_i) arcs.

Finally, these lists are combined and processed by `CanonicalPD` function to produce a unique representation of the knot. We include the code in the case where n is even below, and WLOG the even parameter is the first one.

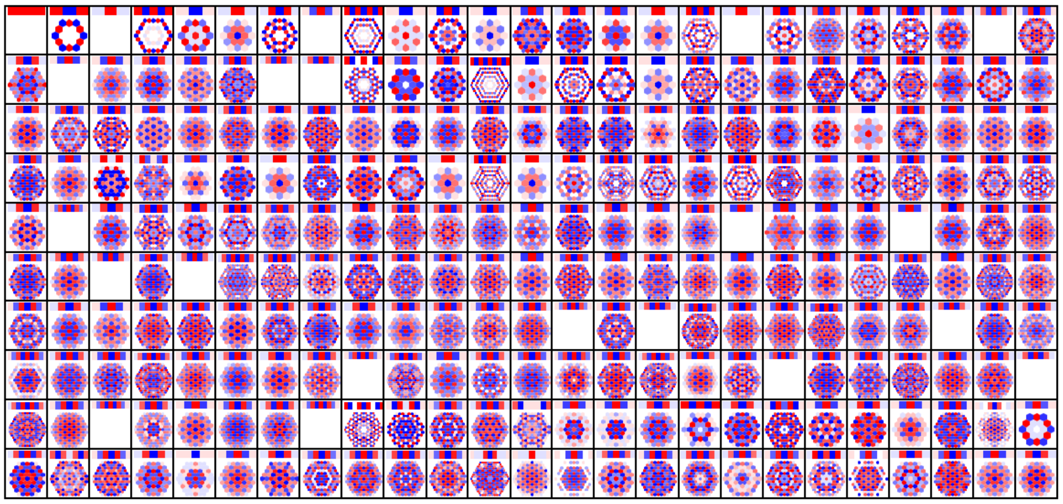
```
X1,1[a_, b_, c_, d_] := X[a, b, c, d]
X1,0[a_, b_, c_, d_] := X[c, d, a, b]
X-1,1[a_, b_, c_, d_] := X[d, a, b, c]
X-1,0[a_, b_, c_, d_] := X[b, c, d, a]
```

```
EvenPretzel[a_List] :=
 Module[{n = Length[a], topCrossings, bottomCrossings, interiorCrossings},


  topCrossings = Table[If[Abs[a⟦i⟧] == 1,Nothing,
  XSign[a⟦i⟧], Mod[i,2][t[Mod[i-1,n]], lc[i,Abs[a⟦i⟧]-1], rc[i,Abs[a⟦i⟧]-1], t[Mod[i,n]]]],
  {i,1,n}];


  bottomCrossings = Table[
     If[Abs[a⟦i⟧] == 1,
     XSign[a⟦i⟧], Mod[i,2][t[Mod[i-1,n]], b[Mod[i-1,n]], b[Mod[i,n]], t[Mod[i,n]]],
     XSign[a⟦i⟧], Mod[i,2][lc[i,1], b[Mod[i-1,n]], b[Mod[i,n]], rc[i,1]]],
    {i,1,n}
  ];


  interiorCrossings = Flatten @ Table[
     XSign[a⟦i⟧], Mod[i,2][
       lc[i,j], lc[i,j-1], rc[i,j-1], rc[i,j]],
     {i,1,n}, {j,2,Abs[a⟦i⟧]-1}
  ];


   CanonicalPD [PD @@ Join[
    DeleteCases[topCrossings, Nothing],
    bottomCrossings,
    interiorCrossings ]
]
]
```

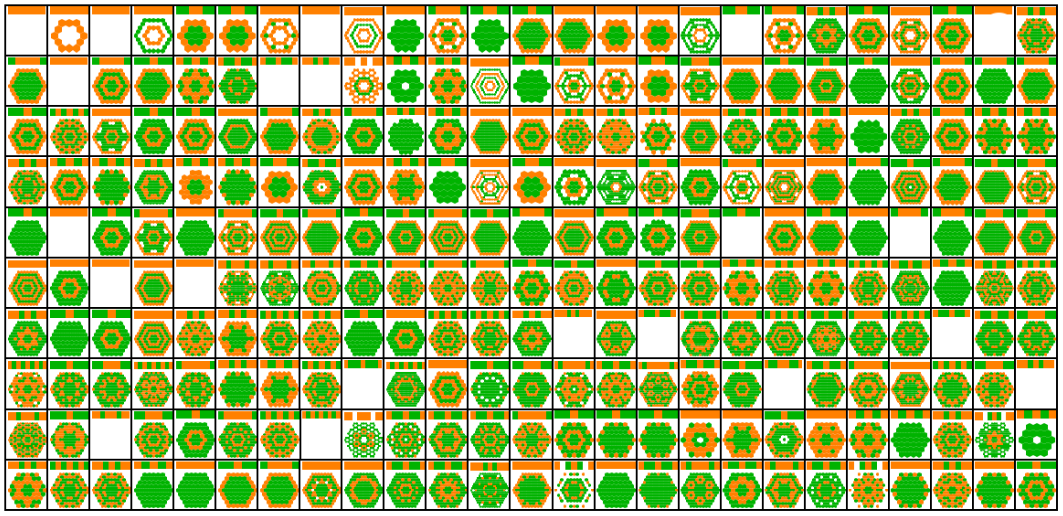
3. Computational Results

3.1. **The Rolfsen Table.** We ran Θ on the Rolfsen table, and include their Parity and Poly plots below.

Figure 8. PolyPlot and ParityPlot diagrams for prime knots up to 10 crossings.



(A) `PolyPlot`: Red and blue points mark positive and negative coefficients, respectively.



(B) `ParityPlot`: Orange and emerald points mark odd and even coefficients, respectively.

3.2. **Torus Knots.** Our computational experiments with torus knots have led to several observations and conjectures, primarily based on the visual patterns in their RainbowPlot diagrams.

Conjecture 3.1. For any torus knot $T(p,q)$ with $p,q > 0$, the total degree of the Laurent polynomial component, θ_K, is given by the formula $2(p-1)(q-1)$.

In addition to the degree, the visual structure of the plots suggests further properties:

Conjecture 3.2. The constant term of the θ invariant for torus knots always vanishes. Visually, this corresponds to the center of the hexagonal plot always being white.

Conjecture 3.3. The visual dispersion of the plot, specifically the number of "vanishing bands," appears to correlate with the distance between the parameters p and q. Knots where p and q are close, such as $T(p, 2)$ or $T(p, 3)$, tend to have solid, concentrated plots. As the difference $|p - q|$ increases, the plot becomes more dispersed.

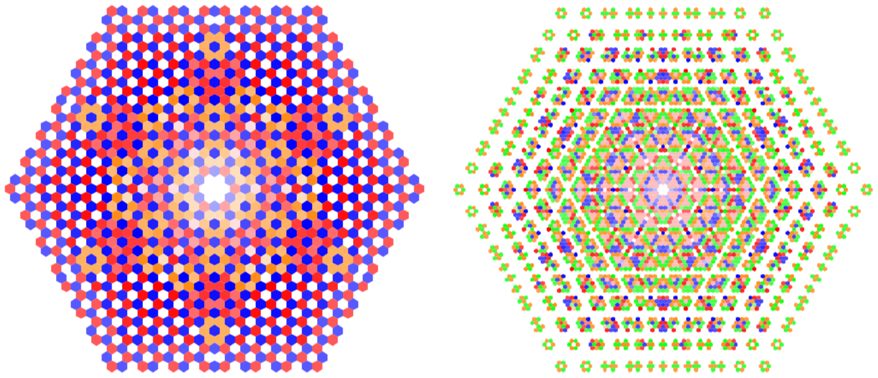


FIGURE 9. Comparison of plot dispersion for T(11,3) (left) and T(11,6) (right).

Conjecture 3.4. For sufficiently large p, the plots for torus knots of the form $T(p, p-1)$ exhibit a distinct, highly structured pattern. This pattern resembles a flower or star inscribed within the inner hexagon of the plot, characterized by alternating colored bands.

3.3. **Twist Knots.** The family of twist knots, K_n, exhibits a remarkable consistency in the structure of their Θ invariant. The PolyPlots and ParityPlots are highly structured, leading to the following conjectures.

Conjecture 3.5. For any twist knot K_n, the corresponding hexagon plot is always "three-layered," meaning the Laurent polynomial component, θ_{K_n}, is of total degree 4. Furthermore, the constant term is always a negative, even integer.

This structure is visible in Figure 12, which shows the plot for the K_4 twist knot. The plot consists of a center point, an inner hexagon, and an outer hexagon.

Conjecture 3.6. The coefficients of the monomials on the "rims" of the hexagon follow a strict pattern. The outer rim alternates in pairs between blue (negative, even coefficients) and green (negative, odd coefficients).

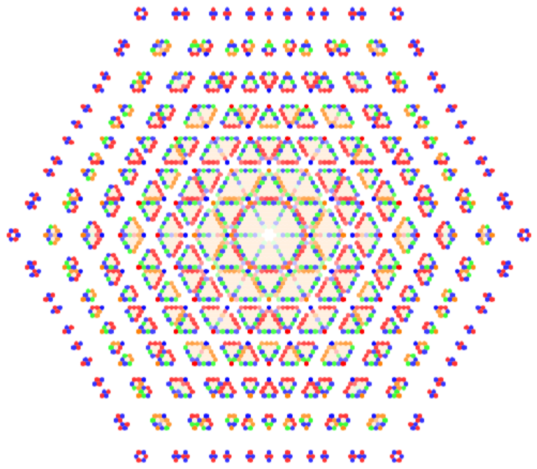


FIGURE 10. Characteristic "star" pattern observed in the RainbowPlot for the T(9,8) knot.

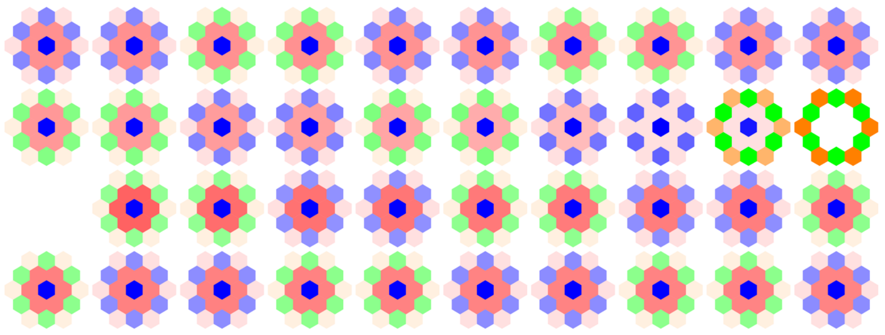


FIGURE 11. The RainbowPlots for twist knots from -20 to 20

These visual patterns suggest a precise algebraic structure for the θ invariant of twist knots. Based on polynomial interpolation of the coefficients in terms of n, we conjecture the following explicit formula for θ_{K_n}.

Conjecture 3.7. For a twist knot K_n with $n > 0$, its θ invariant is given by the formula:

$$\theta_{K_n} = C_C(n) \cdot S_C + C_P(n) \cdot S_P + C_O(n) \cdot S_O + C_K(n) \tag{3.1}$$

where S_C, S_P, S_O are sums of monomials corresponding to the inner hexagon, the mid-rim, and the outer rim, respectively:

$$S_C = T_1 + T_2 + T_1^{-1} + T_2^{-1} + T_1T_2 + (T_1T_2)^{-1}$$
$$S_P = T_1T_2^{-1} + T_2T_1^{-1} + T_1^{-2}T_2^{-1} + T_1^{-1}T_2^{-2} + T_1^2T_2 + T_1T_2^2$$
$$S_O = T_1^{-2} + T_2^{-2} + (T_1T_2)^{-2} + T_1^2 + T_2^2 + (T_1T_2)^2$$

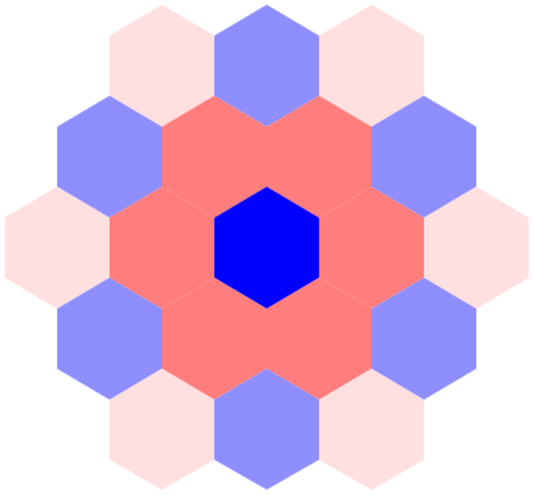


FIGURE 12. The PolyPlot for the twist knot K_8). The three-layered structure and alternating colors on the outer rim are clearly visible.

and the coefficients $C(n)$ are polynomials in n:

$$C_C(n) = \frac{n^4}{2} - \frac{n^2}{2}$$
$$C_P(n) = -\frac{n^4}{2} + \frac{2n^3}{3} - \frac{n}{6}$$
$$C_O(n) = \frac{n^4}{4} - \frac{n^3}{2} + \frac{n^2}{4}$$
$$C_K(n) = -\frac{3n^4}{2} - n^3 + \frac{3n^2}{2} + n$$

3.4. **Twisted Torus Knots.** As expected, twisted torus knots yield more intricate patterns than the usual torus knots.

To see this, we now fix a base pair and vary the twisting parameters. For $(p, q) = (7, 3)$, increasing r and s yields RainbowPlots with more intricate patterns. The grid below collects representative outputs.

Observation. For fixed $(p, q) = (7, 3)$, increases in r or s adds more vanishing bands, making the RainbowPlots progressively center-heavy. The same occurs for other (p, q) tested.

$$(p, q) = (7, 3)$$

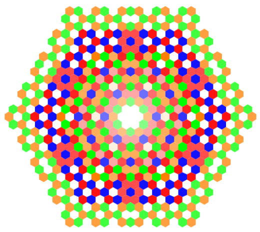
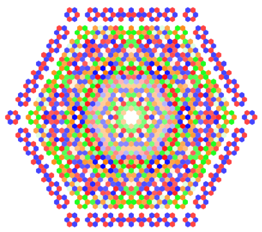
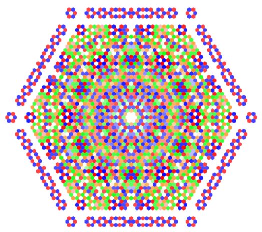
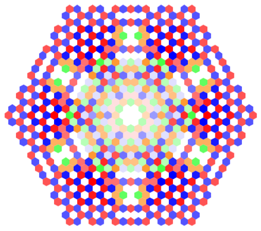
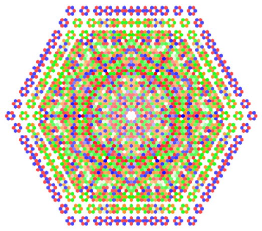
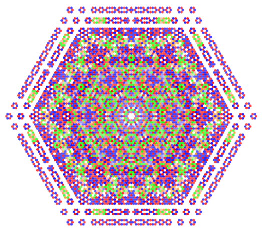
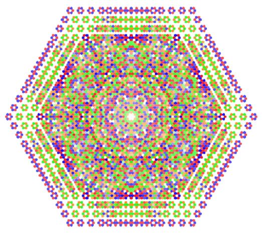
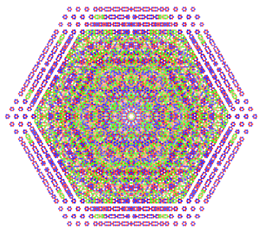
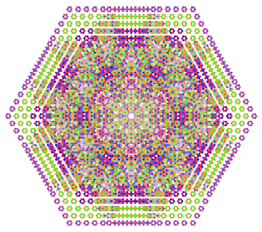
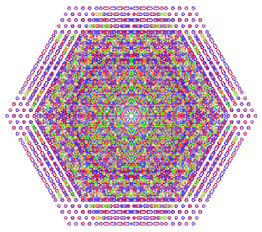
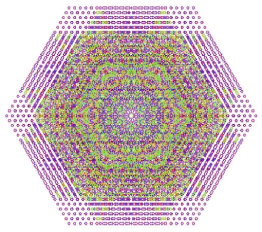
$s\downarrow \quad r\rightarrow$	2	4	5
1			
2			
3			
4			
5			

FIGURE 13. RainbowPlots for $T(7, 3, r, s)$ as r, s vary. Empty cells indicate parameter choices not displayed.

3.4.1. *The Analogue for Twisted Torus Knots Fails.* A natural question is whether Conjecture 3.2 extends to the broader class of twisted torus knots. This, however, is not the case. The analogous statement for twisted torus knots is false, as shown by the following counterexample.

Example 3.8 (Counterexample)**.** Direct computation for the twisted torus knot $T(7, 4, 2, 2)$ gives

$$\Theta\big(T(7, 4, 2, 2)\big) \;=\; 96 \;+\; \ldots,$$

which has a non-zero constant term.

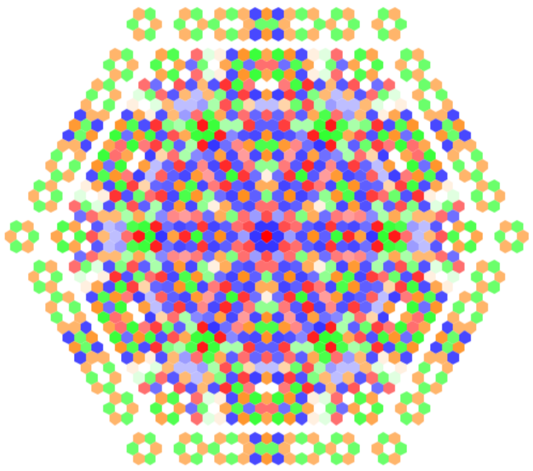


FIGURE 14. The RainbowPlot for the Twisted Torus Knot $T(7, 4, 2, 2)$. We can see the center is red instead of white.

Remark 3.9. This counterexample does not contradict Conjecture 3.2 for torus knots. Lee's classification of twisted torus knots that are themselves torus knots (Theorem 1.1 of [Lee15]) shows that $T(p, q, r, s)$ can be isotopic to a torus knot $T(p', q')$ only when the twist parameter $s = \pm 1$ or $s = -2$. Because $s = 2$ here fails that condition, $T(7, 4, 2, 2)$ is *not* equivalent to any torus knot, so its behaviour leaves the original torus-knot conjecture unaccounted for.

3.5. **Pretzel Knots.** We did not detect a simple overall pattern in the rainbow plots of higher–parameter pretzel knots. However, the author observed that some cases display a distinct red–blue *checkerboard* region, often fitting inside a larger hexagonal outline. Figure 15 contrasts a few such examples with pretzel knots whose plots do *not* show this feature.

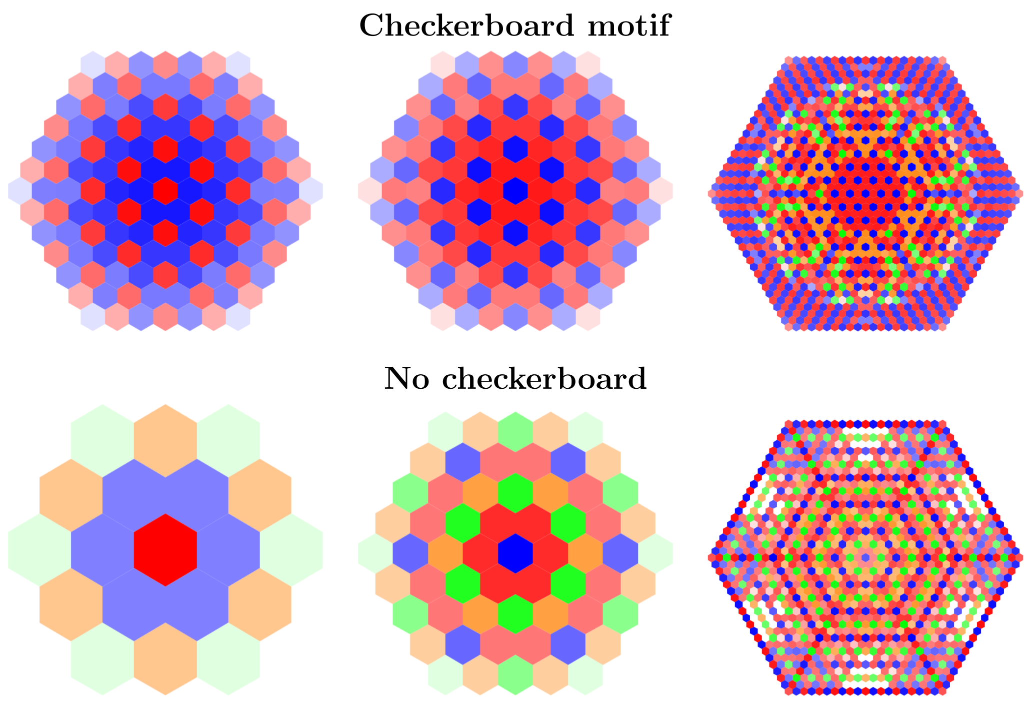


Figure 15. Rainbow plots of representative pretzel knots. Top row: images showing the red–blue checkerboard pattern. Bottom row: plots without that .

Acknowledgements

I would like to express my deepest gratitude to Professor Bar-Natan for his invaluable guidance throughout this project. I am also grateful to the University of Toronto Excellence Award (UTEA) program and the Department of Mathematics at the University of Toronto for funding this project.

References

BV1. D. Bar-Natan and R. van der Veen, *A very fast, very strong, topologically meaningful and fun knot invariant* https://drorbn.net/AcademicPensieve/Projects/Theta/Theta.pdf.

KA. The Knot Atlas, *The KnotTheory package manual*. https://katlas.org/wiki/Printable_Manual.

Lee17. S. Lee, *Knot types of twisted torus knots*, J. Knot Theory Ramifications **26** (2017), no. 12, 1750074.

Lee15. S. Lee, *Torus knots obtained by twisting torus knots*, Algebraic & Geometric Topology **15** (2015), 2817–2836.

Appendix A. Code for CanonicalPD

Here is the implementation of the `CanonicalPD` function used to generate unique planar diagram representations for knots.

```
CanonicalPD[pd_PD] := Module[{ crossings = List @@ pd,numEdges,candidatePDs = {},edgeToCrossingsMap},

    (* Validate input*)
    If[! MatchQ[crossings, {__X}], Return[PD["Invalid input"], Module]];
    numEdges = 2 * Length[crossings];

      edgeToCrossingsMap = GroupBy[
        Flatten[Table[{c⟦i⟧, c}, {c, crossings}, {i, 1, 4}], 1],
        First → Last
      ];

    If[! And @@ (Length[#] ⩵ 2 & /@ Values[edgeToCrossingsMap]),
       Return[
    PD["Invalid PD: Each edge must appear in exactly two crossings."], Module]];

    (* Iterate through all starting points *)
    Do[
       Module[{startEdge = crossing⟦pos⟧,edgeTraversal = {},visitedCrossings = {},oldToNewMap = <||>,newLabelCounter = 1,
           currentEdge,lastCrossing, currentCrossing},
          currentEdge = startEdge;
          lastCrossing = First[edgeToCrossingsMap[currentEdge]];

          (* Traverse through the knot *)
          Do[
             AppendTo[edgeTraversal, currentEdge];

      currentCrossing =
      First[DeleteCases[edgeToCrossingsMap[currentEdge], lastCrossing]];

             (* Build crossings in order *)
             If[! MemberQ[visitedCrossings, currentCrossing],
                AppendTo[visitedCrossings, currentCrossing]
              ];

             lastCrossing = currentCrossing;

     currentEdge = Replace[currentCrossing, X[i_, j_, k_, l_] ⧴

        Which[i ⩵ currentEdge, k, j ⩵ currentEdge, l,
         k ⩵ currentEdge, i, l ⩵ currentEdge, j]];
           , {numEdges}];

    Scan[(If[! KeyExistsQ[oldToNewMap, #],
        oldToNewMap[#] = newLabelCounter++]) &, edgeTraversal];

    AppendTo[candidatePDs, PD @@ (visitedCrossings /. oldToNewMap)];

        ],
       {crossing, crossings}, {pos, 1, 4}
     ];
    If[Length[candidatePDs] > 0, First[Sort[candidatePDs]], PD[]]
      ]
```

[1] Department of Mathematics, University of Toronto, Toronto, Canada.
Email address: alikarim.lalani@mail.utoronto.ca