Determinant Formulas for the Alexander Polynomial

Daniel Martchenkov
University of Toronto

September 4, 2023

Abstract

This is a summary of various determinant formulas for the Alexander polynomial, along with their implementations in Mathematica using the *KnotTheory* package.

Contents

1 Introduction **2**

2 Formulas **3**
2.1 Alexander's Definition (crossings × faces) 3
2.1.1 Formula 3
2.1.2 Implementation 4
2.1.3 Run-through 6
2.1.4 Complete Code 7
2.2 Bar-Natan-Dancso (crossings × crossings) 7
2.2.1 Formula 7
2.2.2 Implementation 8
2.2.3 Run-through 10
2.2.4 Complete Code 11
2.3 Long Position ((edges + 1) × (edges + 1)) 11
2.3.1 Formula 11
2.3.2 Implementation 12
2.3.3 Run-through 15
2.3.4 Complete Code 16
2.4 Arc Presentation 16
2.4.1 Formula 16
2.4.2 Implementation 18
2.4.3 Run-through 19
2.4.4 Complete Code 21
2.5 Burau Representation (Braids) 21

2.5.1 Formula 21
2.5.2 Implementation 22
2.5.3 Run-through 24
2.5.4 Complete Code 25
2.6 Seifert Matrix (via Braids) 25
2.6.1 Formula 25
2.6.2 Implementation 27
2.6.3 Run-through 28
2.6.4 Complete Code 29

3 Acknowledgement and References 29

1 Introduction

The Alexander polynomial of a knot K is an invariant taking values in the ring of integer-coefficient Laurent polynomials, $\mathbb{Z}[T, T^{-1}]$, discovered by James Waddell Alexander II in 1923. In principle, it is defined as a generator of an ideal of this ring, so it is only well-defined up to multiplication by a unit (i.e., by $\pm T^{\pm m}$, $m \in \mathbb{N}$). However, there are two facts which can be used to make a canonical choice:

1. The coefficients of the Alexander polynomial are palindromic.

2. The Alexander polynomial evaluated at 1 is a unit.

Therefore, multiplying by an appropriate unit $\pm T^{\pm m}$, a generator $\Delta_K(T)$ can be chosen, satisfying

$$\Delta_K(T) = \Delta_K(T^{-1}) \text{ and } \Delta_K(1) = 1.$$

In this report, we present a collection of formulas for the Alexander polynomial whose final step, aside from some minor corrections, is taking the determinant of a matrix with entries in $\mathbb{Z}[T, T^{-1}]$. Where possible, we describe an algorithm for computing the unit required to produce Δ_K. For each formula, we start by describing the algorithm for computing it by hand. We then describe its implementation in Mathematica using the *KnotTheory* package. Next, we run the algorithm on the knot 6_1, outputting the values of intermediate objects. Finally, we present the complete code. The starting point for every algorithm is either a PD presentation[1] of the knot, or a different presentation of the knot along with an algorithm which can be followed to obtain it from a PD presentation. Anything that is necessary for the implementation which is not already included in the *KnotTheory* package is also implemented.

See figure 1 for a drawing of the knot 6_1 generated by *KnotTheory*'s function *DrawPD*. For reference, its Alexander polynomial is

$$\Delta_{6_1}(T) = -\frac{2}{T} + 5 - 2T.$$

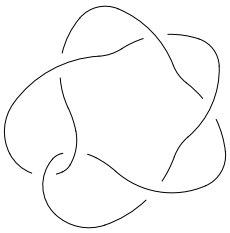


Figure 1: Knot 6_1, as drawn by *KnotTheory*'s function *DrawPD*.

Throughout this report, let K be a knot.
Import *KnotTheory‘* (with the backtick) for all Mathematica implementations:[2]

```
In[•]:= << KnotTheory`
```

2 Formulas

2.1 Alexander's Definition (crossings × faces)

This is a summary of [1], which is analogous to Alexander's original definition.

2.1.1 Formula

Consider a planar diagram for K with n crossings. Viewing the planar diagram as a graph, each vertex (crossing) has four adjacent faces. For every crossing, place the following markings on the adjacent faces.

Figure 2: Markings of faces surrounding a positive and negative crossing, respectively.

Enumerate the crossings $1, \dots n$ in any order, and the faces $1, \dots, n+2$ in any order. Create an $n \times (n+2)$ matrix A, whose (i, j)'th entry is the marker placed on face j at crossing i as in figure 2, or 0 if the face and crossing are not adjacent. Remove two columns of A corresponding to a pair of neighbouring faces (ones separated by a single strand), and take the determinant. The result is the Alexander polynomial for K, up to a unit.

[1] See `http://katlas.org/wiki/Planar_Diagrams`.
[2] A guide to installing this package can be found here: `http://katlas.org/wiki/Setup`.

One cannot expect the sign of the resulting polynomial to be correct as the crossings and faces are labelled arbitrarily. If a systematic labelling were used, then perhaps the sign could be deduced and corrected, however I do not currently know how to do so.

Testing on all prime knots with up to 11 crossings shows that dividing by T^ℓ where ℓ is the number of -1's in the coordinate-wise product of σ and d as defined in 2.2.1 symmetrizes the polynomial. However, I have not proved this yet.

2.1.2 Implementation

The main obstacle to this implementation is identifying and labelling the faces of K. The following code, a slight modification of that written by Dror Bar-Natan,[3] is used to accomplish this.

```
XingsByTurns =
 PD[K] /. x : X[i_, j_, k_, l_] :>
   If[PositiveQ[x], X₊[-i, j, k, -l], X₋[-j, k, l, -i]];
bends = Times @@ XingsByTurns /.
  _[X][a_, b_, c_, d_] :> p_a,-d p_b,-a p_c,-b p_d,-c;
faces = bends //. p_x__,y_ p_y_,z__ :> p_x,y,z;
```

Next, we create the matrix A. Recall that in *XingsByTurns*, positive crossings $X[i,j,k,l]$ become $X_+[-i,j,k,-l]$ and negative crossings become $X_-[-j,k,l,-i]$. Furthermore, faces are labelled by the strands they lie to the left and right of, with negative signs indicating lying to the left. Thus, the face to the right (resp. left) of the strand numbered α is the unique face containing α (resp. $-\alpha$) in its label. See figure 3.

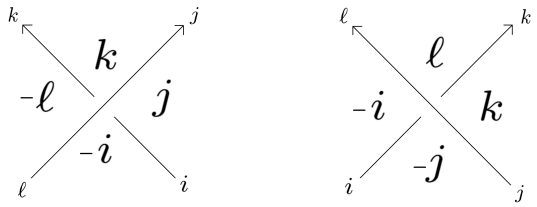


Figure 3: The faces surrounding a crossing

Comparing figures 2 and 3, we see that in the case of a positive crossing, with $[\alpha]$ (resp. $[-\alpha]$) denoting the face to the right (resp. left) of the edge numbered α, we add the following block to the row of A corresponding to this crossing:

$$\begin{array}{cccc} [-i] & [j] & [k] & [-\ell] \\ (-T & 1 & -1 & T\ ) \cdot \end{array}$$

[3] For more information about this code, see http://www.math.toronto.edu/ drorbn/Talks/CMS-2112/index.html

Similarly, in the case of a negative crossing, we add:

$$\begin{matrix} [-j] & [k] & [\ell] & [-i] \\ (\ 1 & -1 & T & -T\ ) \end{matrix} \cdot$$

Here is the above, stated in Mathematica language.

```
A = Table[0, Length@XingsByTurns, Length@faces];
Do[is = Position[faces, #][[1, 1]] & /@ List @@ XingsByTurns[[j]];
 A[[{j}, is]] = If[Head[XingsByTurns[[j]]] === X+,
   ( -T 1 -1 T ), ( 1 -1 T -T )],
 {j, Length@XingsByTurns}];
```

Finally, we remove the columns corresponding to the faces separated by the strand labelled by 1 (i.e., the faces containing 1 and -1 in their labelling)[4], and take the determinant.

```
poly = Det@A[[All, Delete[Range[Length@faces],
      {Position[faces, #][[1, 1]]} & /@ {1, -1}]]];
```

To get the canonical sign, we multiply by our polynomial evaluated at 1, and to symmetrize, we divide by the mean of the minimal and maximal exponents of T.

```
(poly /. T → 1) poly
----------------------------------------------------]
T^(Exponent[poly,T,Max]+Exponent[poly,T,Min])/2
```

[4]This is an arbitrary choice, what is important is that two adjacent faces are being removed.

2.1.3 Run-through

```
In[•]:= K = Knot[6, 1];

In[•]:= faces

Out[•]= p_{-11,-7,-11} p_{-5,-1,-5} p_{3,9,3} p_{6,12,6} p_{4,-9,2,4} p_{10,-3,8,10} p_{-2,-10,7,-12,5,-2} p_{1,-6,11,-8,-4,1}

In[•]:= A // MatrixForm

Out[•]//MatrixForm=
```

$$\begin{pmatrix} 0 & -T & 0 & 0 & -1 & 0 & T & 1 \\ -T & 0 & 0 & 0 & 0 & -1 & 1 & T \\ 0 & 0 & 1 & 0 & -1 & -T & 0 & T \\ 0 & 0 & 1 & 0 & -T & -1 & T & 0 \\ 0 & -T & 0 & -1 & 0 & 0 & 1 & T \\ -T & 0 & 0 & -1 & 0 & 0 & T & 1 \end{pmatrix}$$

```
In[•]:= matrix = A[[All, Delete[Range[Length@faces],
            {Position[faces, #][[1, 1]]} & /@ {1, -1}]]];
        poly = matrix // Det;

In[•]:= {matrix // MatrixForm, Apart[matrix // Det]}
```

$$\left\{ \begin{pmatrix} 0 & 0 & 0 & -1 & 0 & T \\ -T & 0 & 0 & 0 & -1 & 1 \\ 0 & 1 & 0 & -1 & -T & 0 \\ 0 & 1 & 0 & -T & -1 & T \\ 0 & 0 & -1 & 0 & 0 & 1 \\ -T & 0 & -1 & 0 & 0 & T \end{pmatrix}, -2\,T + 5\,T^2 - 2\,T^3 \right\}$$

```
In[•]:= (poly /. T → 1) poly / T^((Exponent[poly,T,Max]+Exponent[poly,T,Min])/2) // Apart
```

$$\text{Out[•]= } 5 - \frac{2}{T} - 2\,T$$

2.1.4 Complete Code

```
AlexanderOriginal[K_] := T ↦ Apart@Module[
    {XingsByTurns, bends, faces, p, A, is, poly},
    XingsByTurns =
     PD[K] /. x : X[i_, j_, k_, l_] :>
       If[PositiveQ[x], X₊[-i, j, k, -l], X₋[-j, k, l, -i]];
    bends = Times @@ XingsByTurns /.
      _[X][a_, b_, c_, d_] :> p_{a,-d} p_{b,-a} p_{c,-b} p_{d,-c};
    faces = bends //. p_{x__,y_} p_{y_,z__} :> p_{x,y,z};
    A = Table[0, Length@XingsByTurns, Length@faces];
    Do[is = Position[faces, #][[1, 1]] & /@ List @@ XingsByTurns[[j]];
     A[[{j}, is]] = If[Head[XingsByTurns[[j]]] === X₊,
       (-T 1 -1 T), (1 -1 T -T)],
     {j, Length@XingsByTurns}];
    poly = Det@A[[All, Delete[Range[Length@faces],
         {Position[faces, #][[1, 1]]} & /@ {1, -1}]]];
    (poly /. T → 1) poly / T^((Exponent[poly,T,Max]+Exponent[poly,T,Min])/2)
]
```

2.2 Bar-Natan-Dancso (crossings × crossings)

This is a summary of [2] and [3].

2.2.1 Formula

Consider a planar diagram for K with n crossings in long position, cut at the $2n \to 1$ strand.

We give a preliminary definition: given a crossing of K indexed by i, its *span*, $Sp(i)$, is the oriented open segment joining the lower-indexed outgoing strand to the higher-indexed incoming strand (i.e., it is the open segment of the knot which is traversed between the two times that the crossing indexed by i is passed, in the direction given by the orientation of the long knot). See figure 4 below.

Construct an $n \times n$ matrix A by the rule that $a_{ij} = 1$ if $Sp(i)$ passes under the crossing labelled j, and 0 if it does not[5]. For example, in figure 4, indexing the crossings from bottom to top by 1 through 3, we have that $a_{21} = 1$ whereas $a_{23} = 0$.

Next, define the array σ by σ_i being the sign of crossing i, and the array d by $d_i = 1$ if the upper strand at crossing i has smaller indices than the lower strand (i.e., appears first when the knot is traversed) and -1 otherwise. Note

[5] since $Sp(i)$ is defined to be the *open* segment joining a vertex to itself, $a_{ii} = 0$

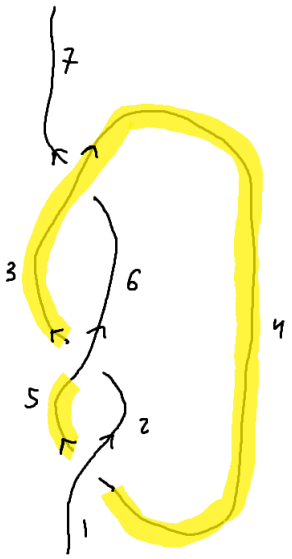


Figure 4: The span of the crossing 2635

that d_i is well-defined as the indices of a single strand differ by 1 in a crossing (eg. $4 \to 5$ vs $1 \to 2$ in crossing 1 of figure 4).

Next, construct the diagonal matrix S whose (i, i) entry is given by:

$$T^{1-\sigma_i d_i}.$$

Taking the determinant of the matrix $I + AS$ gives the Alexander polynomial for K, with no correction necessary.

2.2.2 Implementation

First, we define a local function which takes in a crossing, and returns a list containing the labels of the two outgoing strands.

```
outgoingStrands[x_] := List @@ x[[{If[PositiveQ[x], 2, 4], 3}]];
```

We start by cutting along the $2n \to 1$ strand by identifying the crossing which has an outgoing strand labelled by 1.

```
(*Cutting along 2n→1 strand*)
Xings = PD[K] /. x : X[a_, b_, c_, d_] /;
    MemberQ[outgoingStrands[x], 1] :> Replace[x, 1 → 2 n + 1, {-1}];
```

Next, we define a function which returns the span of a crossing, given its index. Note that, by definition, the span starts from the outgoing strand with the smaller index. However, since a knot has one component, we return to this crossing along the other strand (the one with the larger index).

```
Sp[i_] := Delete[
  Range[Sequence @@ Sort@outgoingStrands[Xings[[i]]],
  -1];
```

We create the matrix A and the arrays σ and d.

```
A = Table[If[DisjointQ[List @@ Xings[[j, {1, 3}]], Sp[i]], 0, 1],
   {i, n}, {j, n}] - I;
σ = Table[If[PositiveQ[Xings[[i]], 1, -1], {i, n}];
d = Table[If[Xings[[i, 1]] ≥ Xings[[i, 2]], 1, -1], {i, n}];
```

Finally, we take the determinant:

```
Det[I + A. (I - DiagonalMatrix[T^-σ*d])]];
```

2.2.3 Run-through

```
In[•]:= K = Knot[6, 1];

In[•]:= Xings = PD[K] /. x : X[a_, b_, c_, d_] /;
          MemberQ[outgoingStrands[x], 1] :> Replace[x, 1 → 2 n + 1, {-1}]

Out[•]= PD[X[1, 4, 2, 5], X[7, 10, 8, 11], X[3, 9, 4, 8],
       X[9, 3, 10, 2], X[5, 12, 6, 13], X[11, 6, 12, 7]]

In[•]:= MatrixForm[A = Table[If[DisjointQ[List @@ Xings[[j, {1, 3}]], Sp[i]], 0, 1],
          {i, n}, {j, n}] - I_n]
```

Out[•]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 \end{pmatrix}$$

```
In[•]:= σ = Table[If[PositiveQ[Xings[[i]]], 1, -1], {i, n}]

Out[•]= {-1, -1, 1, 1, -1, -1}

In[•]:= d = Table[If[Xings[[i, 1]] ≥ Xings[[i, 2]], 1, -1], {i, n}]

Out[•]= {-1, -1, -1, 1, -1, 1}

        MatrixForm[I_n - DiagonalMatrix[T^(-σ*d)]]
```

Out[•]//MatrixForm=

$$\begin{pmatrix} 1-\frac{1}{T} & 0 & 0 & 0 & 0 & 0 \\ 0 & 1-\frac{1}{T} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1-T & 0 & 0 & 0 \\ 0 & 0 & 0 & 1-\frac{1}{T} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1-\frac{1}{T} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1-T \end{pmatrix}$$

```
In[•]:= MatrixForm[A.(I_n - DiagonalMatrix[T^(-σ*d)])]
```

Out[•]//MatrixForm=

$$\begin{pmatrix} 0 & 0 & 1-T & 0 & 0 & 0 \\ 0 & 0 & 0 & 1-\frac{1}{T} & 0 & 0 \\ 0 & 1-\frac{1}{T} & 0 & 0 & 1-\frac{1}{T} & 0 \\ 0 & 1-\frac{1}{T} & 1-T & 0 & 1-\frac{1}{T} & 0 \\ 0 & 1-\frac{1}{T} & 0 & 1-\frac{1}{T} & 0 & 1-T \\ 0 & 1-\frac{1}{T} & 0 & 1-\frac{1}{T} & 0 & 0 \end{pmatrix}$$

```
In[•]:= MatrixForm[In + A.(In - DiagonalMatrix[T^(-σ*d)])]
```

Out[•]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & 1-T & 0 & 0 & 0 \\ 0 & 1 & 0 & 1-\frac{1}{T} & 0 & 0 \\ 0 & 1-\frac{1}{T} & 1 & 0 & 1-\frac{1}{T} & 0 \\ 0 & 1-\frac{1}{T} & 1-T & 1 & 1-\frac{1}{T} & 0 \\ 0 & 1-\frac{1}{T} & 0 & 1-\frac{1}{T} & 1 & 1-T \\ 0 & 1-\frac{1}{T} & 0 & 1-\frac{1}{T} & 0 & 1 \end{pmatrix}$$

```
In[•]:= Det[In + A.(In - DiagonalMatrix[T^(-σ*d)])] // Apart
```

Out[•]= $5 - \frac{2}{T} - 2T$

2.2.4 Complete Code

```
AlexanderBND[K_] := T ↦ Apart@Module[{Xings, n = Length@PD[K], Sp,
    I = IdentityMatrix[Length@PD[K]], outgoingStrands, A, σ, d},
  outgoingStrands[x_] := List @@ x[[{If[PositiveQ[x], 2, 4], 3}]];
  (*Cutting along 2n→1 strand*)
  Xings = PD[K] /. x : X[a_, b_, c_, d_] /;
      MemberQ[outgoingStrands[x], 1] :↦ Replace[x, 1 → 2 n + 1, {-1}];
  Sp[i_] := Delete[
    Range[Sequence @@ Sort@outgoingStrands[Xings[[i]]]],
    -1];
  A = Table[If[DisjointQ[List @@ Xings[[j, {1, 3}]], Sp[i]], 0, 1],
     {i, n}, {j, n}] - I;
  σ = Table[If[PositiveQ[Xings[[i]]], 1, -1], {i, n}];
  d = Table[If[Xings[[i, 1]] ≥ Xings[[i, 2]], 1, -1], {i, n}];
  Det[I + A.(I - DiagonalMatrix[T^(-σ*d)])]];
```

2.3 Long Position ((edges + 1) × (edges + 1))

This is a summary of [4].

2.3.1 Formula

Consider a long position planar diagram for K with n crossings, cut along the $2n \to 1$ strand. Note that there are $2n+1$ edges.

Start with A being the $(2n+1) \times (2n+1)$ identity matrix. For each crossing, let (s, i, j) be the triple where s is the sign of the crossing, i is the index of the incoming upper strand, and j is the index of the incoming lower strand. Then,

add the following 2×2 block to A:

$$\begin{array}{c} \\ i \\ j \end{array} \begin{array}{c} \begin{array}{cc} i+1 & j+1 \end{array} \\ \begin{pmatrix} -T^s & T^s - 1 \\ 0 & -1 \end{pmatrix} \end{array}$$

The determinant of the resulting matrix gives the Alexander polynomial for K, up to a unit. In fact, the sign is correct, as plugging in $T = 1$ results in the following block being added at each stage

$$\begin{array}{c} \\ i \\ j \end{array} \begin{array}{c} \begin{array}{cc} i+1 & j+1 \end{array} \\ \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \end{array}$$

which only modifies the superdiagonal[6] entries of A. As a result, A is an upper triangular matrix, so its determinant is the product of its diagonal entries, which are all 1.

To symmetrize the polynomial, we need to calculate the rotation number of the long knot diagram. To do so, it must be put into upright form. That is, it must be presented as a smooth immersion of $[0, 1]$ with strands entering and exiting crossings pointing directly upward. Furthermore, the beginning and end of the knot diagram must also be pointing directly upward. In practice, this means that the diagram must be smoothed, and the strands around crossings and at the endpoints of the diagram must be rotated slightly to be upright. See figures 5 and 6.

Recall that the rotation number of an immersion $\gamma : [0, 1] \to \mathbb{R}^2$, with $\gamma'(0)$ and $\gamma'(1)$ pointing in the same direction is defined to be the winding number of its unit tangent map $T_\gamma(s) = \gamma'(s)/|\gamma'(s)|$ around the origin. Note that the requirement that $\gamma'(0)$ and $\gamma'(1)$ point in the same direction ensures that T_γ is a closed curve, hence has a winding number.

For example, in figure 6, the rotation number for edge 4 is -1, and 0 for all other edges.

To symmetrize the polynomial, multiply by $T^{(-\varphi - w)/2}$, where φ is the total rotation number of the knot (i.e., the sum of the rotation numbers of the edges) and w is the writhe of the diagram (the sum of the signs of its crossings).

2.3.2 Implementation

We start by initializing the matrix A to be the $(2n+1) \times (2n+1)$ identity matrix, and defining a function $s[x]$ which returns the sign of crossing x.

```
A = IdentityMatrix[2 * Length@PD[K] + 1];
s[x_] := If[PositiveQ@x, 1, -1];
```

[6] That is, the entries directly above the diagonal.

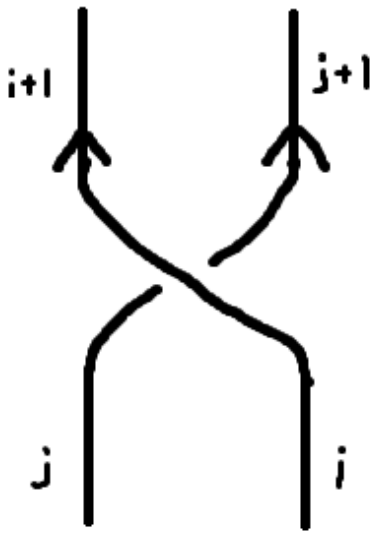


Figure 5: A crossing with strands pointing upward.

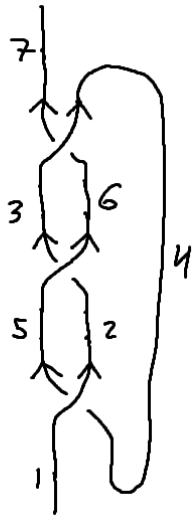


Figure 6: Long trefoil in upright form.

For each crossing x, we set is to be the pair consisting of the indices of the incoming upper and incoming lower strands of x, respectively, and add the necessary 2×2 block.

```
Do[is = If[PositiveQ@x, List @@ x[[{4, 1}]], List @@ x[[{2, 1}]]];
 A[[is, # + 1 & /@ is]] = ( -T^s[x]  T^s[x] - 1
                             0       -1        ),
 {x, List @@ PD[K]}];
```

To compute the total rotation number, we use a supplementary function, written by Dror Bar-Natan.[7]

[7] See *Rot* in `http://drorbn.net/AcademicPensieve/Projects/APAI/`.

```
Rot[pd_PD] := Module[{n, xs, x, rots, Xp, Xm, front = {1}, k},
  n = Length@pd; rots = Table[0, {2 n}];
  xs = Cases[pd, x_X :> {Xp[x[[4]], x[[1]]]  PositiveQ@x
                         Xm[x[[2]], x[[1]]]  True}];
  For[k = 1, k ≤ 2 n, ++k,
   If[FreeQ[front, -k],
     front = Flatten@Replace[front, k → (xs /. {
             Xp[k, l_] | Xm[l_, k] :> {l + 1, k + 1, -l},
             Xp[l_, k] | Xm[k, l_] :> (++rots[[l]]; {-l, k + 1, l + 1}),
             _Xp + _Xm :> {}
            }), {1}],
     Cases[front, k | -k] /. {k, -k} :> --rots[[k]];
    ];];
  {xs /. {Xp[i_, j_] :> {+1, i, j}, Xm[i_, j_] :> {-1, i, j}}, rots}];
Rot[K_] := Rot[PD[K]]
```

Using *Rot*, we compute φ and w, the total rotation number and writhe.

```
φ = Total[Rot[K][[2]]]; w = Total[s /@ List @@ PD[K]];
```

Finally, we take the determinant and symmetrize the result by multiplying by $T^{-(\varphi+w)/2}$.

```
T^-(φ+w)/2 Det[A]]
```

2.3.3 Run-through

```
In[•]:= K = Knot[6, 1];

In[•]:= {φ = Total[Rot[K][[2]]], w = Total[s /@ List @@ PD[K]]}

Out[•]= {-2, -2}

In[•]:= A // MatrixForm
```

Out[•]//MatrixForm=

$$\begin{pmatrix}
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -T & 0 & 0 & 0 & 0 & 0 & 0 & -1+T & 0 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1+\frac{1}{T} & 0 & 1 & -\frac{1}{T} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & -\frac{1}{T} & 0 & 0 & 0 & 0 & -1+\frac{1}{T} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1+T & 0 & 0 & 0 & 1 & -T & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1+\frac{1}{T} & 0 & 1 & -\frac{1}{T} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1+\frac{1}{T} & 0 & 0 & 0 & 0 & 0 & 1 & -\frac{1}{T} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

```
In[•]:= {T^-(φ+w)/2, Det[A] // Apart}

Out[•]= {T^2, -2/T^3 + 5/T^2 - 2/T}

In[•]:= T^-(φ+w)/2 Det[A] // Apart

Out[•]= 5 - 2/T - 2 T
```

2.3.4 Complete Code

```
Rot[pd_PD] := Module[{n, xs, x, rots, Xp, Xm, front = {1}, k},
   n = Length@pd; rots = Table[0, {2 n}];
   xs = Cases[pd, x_X :> {Xp[x[[4]], x[[1]]]  PositiveQ@x
                          Xm[x[[2]], x[[1]]]  True}];
   For[k = 1, k ≤ 2 n, ++k,
    If[FreeQ[front, -k],
      front = Flatten@Replace[front, k → (xs /. {
             Xp[k, l_] | Xm[l_, k] :> {l + 1, k + 1, -l},
             Xp[l_, k] | Xm[k, l_] :> (++rots[[l]]; {-l, k + 1, l + 1}),
             _Xp + _Xm :> {}
            }), {1}],
      Cases[front, k | -k] /. {k, -k} :> --rots[[k]];
     ];];
   {xs /. {Xp[i_, j_] :> {+1, i, j}, Xm[i_, j_] :> {-1, i, j}}, rots}];
Rot[K_] := Rot[PD[K]]
AlexanderLongPosition[K_] := T ↦ Apart@Module[{A, s, is, φ, w},
    A = IdentityMatrix[2 * Length@PD[K] + 1];
    s[x_] := If[PositiveQ@x, 1, -1];
    Do[is = If[PositiveQ@x, List @@ x[[{4, 1}]], List @@ x[[{2, 1}]]];
     A[[is, # + 1 & /@ is]] = ( -T^s[x]   T^s[x] - 1
                                  0          -1      ),
     {x, List @@ PD[K]}];
    φ = Total[Rot[K][[2]]]; w = Total[s /@ List @@ PD[K]];
    T^-(φ+w)/2 Det[A]]
```

2.4 Arc Presentation

This is a summary of chapter 3 of [5] with notation and conventions adjusted to match that of the Knot Atlas[8] and the Mathematica *KnotTheory* package.

2.4.1 Formula

First, we need a definition. An *arc presentation* G of a knot K is an oriented planar diagram of K with all edges either vertical or horizontal, and with vertical edges crossing over horizontal ones. Furthermore, all x-coordinates of vertical edges are distinct, and all y-coordinates of horizontal edges are distinct. If G is

[8] http://katlas.org/wiki/Arc_Presentations

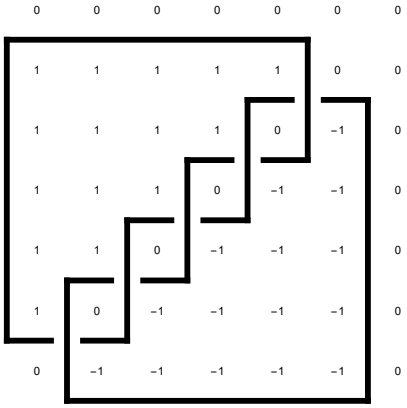


Figure 7: An arc presentation of the cinquefoil with its minesweeper matrix.

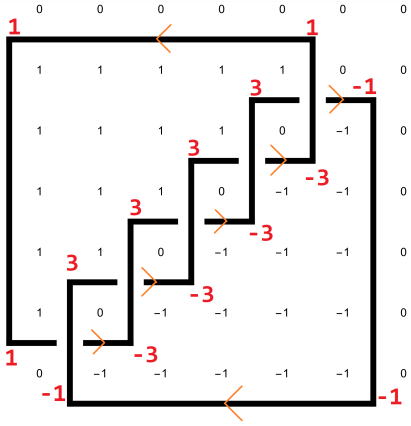


Figure 8: $a(G) = 0$ for this arc presentation.

an arc presentation with n vertical (and horizontal) edges, then we may pick all x- and y-coordinates to be integers from 1 to n. We will assume this convention. See figure 7 for an arc presentation of the cinquefoil with its orientation suppressed. See also section 2.4.3 for an arc presentation of the knot 6_1, also with suppressed orientation.

Let G be an $n \times n$ arc presentation of K.

Compute its *minesweeper matrix*, M – the matrix whose (ij)'th entry is the winding number of the arc presentation around a point in the region whose x-coordinate is between i and $i+1$, and whose y-coordinate is between j and $j+1$.[9] For $i = n$ or $j = n$, the winding number is 0 since that is in the outer face of the arc presentation (recall that the arc presentation consists of lines with integer coordinates between 1 and n). Figure 7 is an arc presentation of the cinquefoil with its minesweeper matrix overlaid on it.[10]

Then,

$$(T^{-1/2} - T^{1/2})^{1-n} \det\left(\left(T^{m_{ij}}\right)\right)$$

is the Alexander polynomial for K, up to a unit.

To get the canonical sign, multiply by the sign of the permutation $\sigma(G)$, where $\sigma(G)(i)$ is the x-coordinate of the starting point of the i'th horizontal edge, counted from the bottom. For the arc presentation in figure 7, the permutation $\sigma(G)$ is

$$(\sigma(G)(1), \ldots, \sigma(G)(7)) = (2, 3, 4, 5, 6, 7, 1),$$

which has sign $+1$.

[9] Recall that an arc presentation is oriented.

[10] With the definition of $MinesweeperMatrix$ given in 2.4.4, the code to generate this is

```
With[{ap = ArcPresentation[Knot[5, 1]]}, Draw[ap, OverlayMatrix → MinesweeperMatrix[ap]]]
```

To symmetrize the polynomial, multiply by $T^{a(G)}$, where $a(G)$ is defined as follows:

For every "corner" in G, sum the winding numbers surrounding it and divide the result by 8. Figure 8 shows this calculation for the arc presentation from figure 7, giving $a(G) = 0$.

Finishing off the example, one obtains the polynomial

$$(+1)\cdot T^0\cdot(T^{-1/2}-T^{1/2})^{-6}\cdot(T^5+\frac{1}{T^5}-7T^4-\frac{7}{T^4}+22T^3+\frac{22}{T^3}-42T^2-\frac{42}{T^2}+57T+\frac{57}{T}-62)$$
$$= \frac{1}{T^2} - \frac{1}{T} + 1 - T + T^2,$$

which is the Alexander polynomial of the cinquefoil.

2.4.2 Implementation

The *KnotTheory* package has an implementation for producing an arc presentation of a given knot, called *ArcPresentation*. Given a knot K, *ArcPresentation*$[K]$ returns an ordered list of pairs

$$\{\{x_1^1, x_2^1\}, \ldots, \{x_1^n, x_2^n\}\},$$

which is parsed as follows: the k'th horizontal edge (counted from the bottom) of the arc presentation joins the x_1^k'th vertical edge (counted from the left) with the x_2^k'th one. For example, in figure 8, we would have the list

$$\{\{7,2\},\{1,3\},\{2,4\},\{3,5\},\{4,6\},\{5,7\},\{6,1\}\}.$$

See also 2.4.3.

Calculating the minesweeper matrix is surprisingly simple using Alexander numbering – a combinatorial approach to determining the winding number of a given connected component of the complement of an oriented curve. This approach is based on three facts:

1. The curve partitions the plane into a disjoint union of connected regions, each of which has a well-defined winding number, i.e., the winding numbers around two points in the same region are equal.

2. The winding numbers of two adjacent regions differ by 1, the region with the larger winding number being to the left of the curve.

3. The winding number around a point in the outer (unbounded) region is 0.

In the arc presentation of a knot, scanning upward from the bottom, the only places where the current connected component changes is when a horizontal segment is crossed, which happens precisely between the two given vertical segments.

To implement this, we start with a row of 0s in the outer (unbounded) face of the arc presentation. For each horizontal segment listed in ap, let c_1 (resp.

c_2) be the x-coordinate of the left (resp. right) vertical segment joined by this horizontal segment. Set s to be 1 if the horizontal strand is oriented to the right (i.e., if the second number is bigger), and -1 if it is oriented to the left (if the first is bigger). For each entry of the row corresponding to a region between the left and right vertical segments, add s. See the following code, obtained from the Knot Atlas[11].

```
MinesweeperMatrix[ap_] := Module[{l = Length[ap], currentRow, c1, c2, s},
  currentRow = Table[0, {l}];
  Table[{c1, c2} = Sort[ap[[k]]];
   s = Sign[{-1, 1}.ap[[k]]];
   Do[currentRow[[c]] += s, {c, c1, c2 - 1}];
   currentRow,
   {k, l}]]
```

We set M to be the minesweeper matrix for K.

```
M = MinesweeperMatrix[ap];
```

To symmetrize, it is first necessary to sum the four entries surrounding each "corner" in the arc presentation. In principle this is simple, however there is an issue with corners containing an $x-$ or $y-$coordinate equal to 1 (as there is no row or column with index 0). To account for this, we create the indices function.

```
indices[c_, i_] := Sequence[
  If[i === 1, {i}, {i, i - 1}], If[c === 1, {c}, {c, c - 1}]];
```

To get the correct sign, we multiply by the sign of the permutation described in the formula above.

$$\text{Signature}[\text{List} @@ \text{ap}[[\text{All}, 2]]]\; T^{\text{fixPower}} \left(T^{-1/2} - T^{1/2}\right)^{1-\text{Length}[\text{ap}]} \text{Det}\left[T^{M}\right]$$

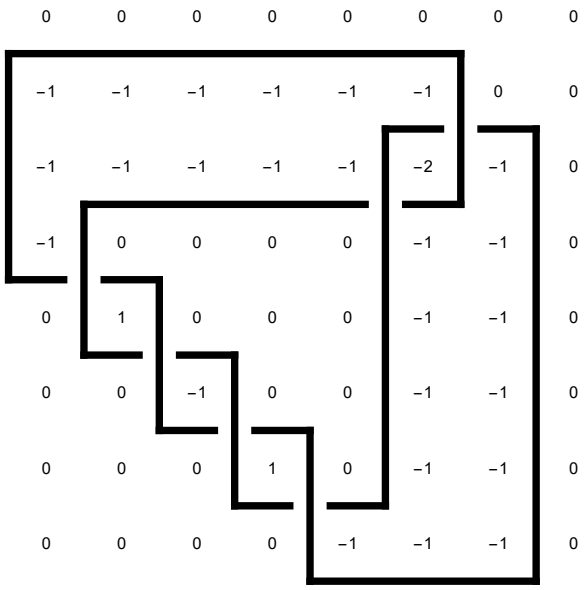
2.4.3 Run-through

```
In[•]:= K = Knot[6, 1];

In[•]:= ap = ArcPresentation[K]

Out[•]= ArcPresentation[{8, 5}, {4, 6}, {5, 3}, {2, 4}, {3, 1}, {7, 2}, {6, 8}, {1, 7}]
```

[11] http://katlas.org/wiki/Arc_Presentations

In[ ]:= `Draw[ap, OverlayMatrix → M]`

Out[ ]=



In[ ]:= `MatrixForm[T^M]`

Out[ ]//MatrixForm=

$$\begin{pmatrix} 1 & 1 & 1 & 1 & \frac{1}{T} & \frac{1}{T} & \frac{1}{T} & 1 \\ 1 & 1 & 1 & T & 1 & \frac{1}{T} & \frac{1}{T} & 1 \\ 1 & 1 & \frac{1}{T} & 1 & 1 & \frac{1}{T} & \frac{1}{T} & 1 \\ 1 & T & 1 & 1 & 1 & \frac{1}{T} & \frac{1}{T} & 1 \\ \frac{1}{T} & 1 & 1 & 1 & 1 & \frac{1}{T} & \frac{1}{T} & 1 \\ \frac{1}{T} & \frac{1}{T} & \frac{1}{T} & \frac{1}{T} & \frac{1}{T} & \frac{1}{T^2} & \frac{1}{T} & 1 \\ \frac{1}{T} & \frac{1}{T} & \frac{1}{T} & \frac{1}{T} & \frac{1}{T} & \frac{1}{T} & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

In[ ]:= `permutation = List @@ ap[[All, 2]]`

Out[ ]= {5, 6, 3, 4, 1, 2, 8, 7}

In[ ]:= `fixPower`

Out[ ]= $\frac{5}{2}$

In[ ]:= `all = {Signature[permutation], T^fixPower, (T^-1/2 - T^1/2)^1-Length[ap], Det[T^M]}`

Out[ ]= $\left\{-1,\ T^{5/2},\ \frac{1}{\left(\frac{1}{\sqrt{T}} - \sqrt{T}\right)^7},\ \frac{2\,T - 19\,T^2 + 79\,T^3 - 189\,T^4 + 287\,T^5 - 287\,T^6 + 189\,T^7 - 79\,T^8 + 19\,T^9 - 2\,T^{10}}{T^8}\right\}$

In[ ]:= `Apart[Times @@ all]`

Out[ ]= $5 - \frac{2}{T} - 2\,T$

2.4.4 Complete Code

```
MinesweeperMatrix[ap_] := Module[{l = Length[ap], currentRow, c1, c2, s},
  currentRow = Table[0, {l}];
  Table[{c1, c2} = Sort[ap[[k]]];
   s = Sign[{-1, 1}.ap[[k]]];
   Do[currentRow[[c]] += s, {c, c1, c2 - 1}];
   currentRow,
   {k, l}]]
AlexanderViaArcPresentation[K_] :=
 T ↦ Apart@Module[{ap = ArcPresentation[K], M, fixPower, indices},
    M = MinesweeperMatrix[ap];
    indices[c_, i_] := Sequence[
      If[i === 1, {i}, {i, i - 1}], If[c === 1, {c}, {c, c - 1}]];
    fixPower = -1/8 Sum[
       Total[M[[indices[ap[[i, 1]], i]]], 2] + Total[M[[indices[ap[[i, 2]], i]]], 2],
       {i, Length@ap}];
    Signature[List @@ ap[[All, 2]]] T^fixPower (T^-1/2 - T^1/2)^(1-Length[ap]) Det[T^M]]
```

2.5 Burau Representation (Braids)

This is essentially Theorem 3.11 of [6].

2.5.1 Formula

Recall that the braid group on n strings, B_n, is naturally generated by $\sigma_1, \ldots, \sigma_{n-1}$, where σ_i is the n-stringed braid whose only crossing is the i'th strand passing over the $(i+1)$'th strand. See figure 9 for the braid σ_3^{-1} in B_4.

Begin by finding a braid whose closure is K.[12] Write this braid as a product of these σ_k. For example, the braid in figure 10 could be written as the product

$$\sigma_1^{-2}\sigma_2\sigma_1^{-1}\sigma_2^2.$$

See 2.5.3 for another example.

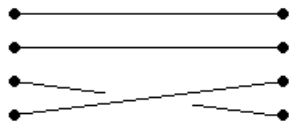


Figure 9: The inverse of σ_3 in B_4. All strings are oriented to the right.

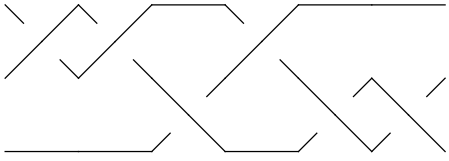


Figure 10: The braid $\sigma_1^{-1}\sigma_1^{-1}\sigma_2\sigma_1^{-1}\sigma_2\sigma_2$.

[12] For example, using Vogel's algorithm.

The reduced Burau representation can be given explicitly by $\sigma_1 \mapsto \begin{pmatrix} -T \end{pmatrix}$ for $n = 2$, and by

$$\sigma_1 \mapsto \left(\begin{array}{cc|c} -T & 1 & 0 \\ 0 & 1 & 0 \\ \hline 0 & 0 & I_{n-3} \end{array}\right),$$

$$\sigma_i \mapsto \left(\begin{array}{c|ccc|c} I_{i-2} & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 \\ 0 & T & -T & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 & I_{n-i-2} \end{array}\right), \text{ for } 2 \leq i \leq n-2,$$

$$\sigma_{n-1} \mapsto \left(\begin{array}{c|cc} I_{n-3} & 0 & 0 \\ \hline 0 & 1 & 0 \\ 0 & t & -T \end{array}\right)$$

for $n \geq 2$.

Let the matrix A be the reduced Burau representation of this braid.

The Alexander polynomial for K, up to a unit, is then given by

$$\frac{1-T}{1-T^n} \det(I - A).$$

Testing on all prime knots with up to 11 crossings shows that the sign is already correct, and that dividing by

$$T^{(1+w-n)/2}$$

where w is the writhe symmetrizes the polynomial. However, I have not proved this yet.

2.5.2 Implementation

The main obstacle to this implementation is finding a braid whose closure is K. The function *BR* in the *KnotTheory* package finds such a braid and returns a list $\{n, br\}$ where n is the number of strings of this braid, and br is an ordered list of integers with absolute value between 1 and $(n-1)$, with a positive integer k representing σ_k and a negative integer $-k$ representing σ_k^{-1}. The braid is the product of these σ_k in the given order.

To start, we unpack $BR[K]$.

```
{n, br} = List @@ BR[K];
```

For later convenience, we implement the Kronecker delta.

```
δ /: δ_i_,j_ := If[i == j, 1, 0];
```

We define the reduced Burau representation as linear maps instead of matrices, and then define a function $m_{i\_}$ to generate the corresponding matrices. Here, $\mathcal{E}$ represents the vector $(v_1, \ldots, v_{n-1})$.

```
σ_i_[ℰ_] := { ℰ /. v_1 :> -T * v_1 + v_2          i === 1
            { ℰ /. v_i :> T * v_(i-1) - T * v_i + v_(i+1)   1 < i < n - 1;
            { ℰ /. v_(n-1) :> T * v_(n-2) - T * v_(n-1)    i === n - 1

m_i_ := σ_i[Table[v_j, {j, n - 1}]] /. v_j_ :> Table[δ_(j,k), {k, n - 1}];
```

Since br is presented as a product of braids whose representations are the σ_i, we can easily compute the reduced Burau representation of the given braid.

```
reducedBurau = Dot @@ br /. i_Integer :> If[i > 0, m_i, m_(Abs[i]) // Inverse];
```

We compute the Alexander polynomial for K, up to a unit.

$$\texttt{poly} = \frac{1 - T}{1 - T^n}\,\texttt{Det[IdentityMatrix[n - 1] - reducedBurau] // Apart;}$$

Finally, we fix the sign of *poly*, and symmetrize it.

$$\frac{(\texttt{poly /. } T \to 1)\ \texttt{poly}}{T^{\texttt{Mean[\{Exponent[poly,T,Max],Exponent[poly,T,Min]\}]}}}\Big];$$

As discussed in 2.5.1, testing on all prime knots with up to 11 crossings shows that the sign is already correct, and that dividing by

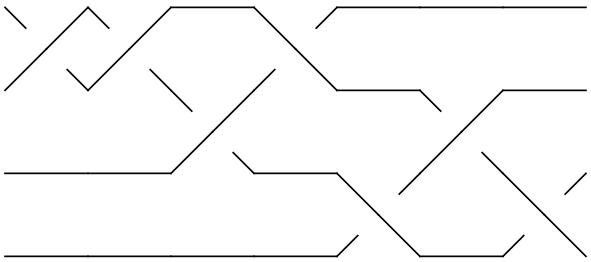
$$T^{(1+w-n)/2}$$

where w is the writhe symmetrizes the polynomial, but I have not proved this yet.

2.5.3 Run-through

```
In[•]:= K = Knot[6, 1];

In[•]:= BR[K]

Out[•]= BR[4, {-1, -1, -2, 1, 3, -2, 3}]

In[•]:= Show[BraidPlot[BR[K]]]
```

Out[•]=



```
In[•]:= MatrixForm /@
        (burauMatrices = List @@ br /. i_Integer :> If[i > 0, m_i, m_Abs[i] // Inverse])
```

$$\text{Out}[\bullet]= \left\{\begin{pmatrix} -\frac{1}{T} & \frac{1}{T} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} -\frac{1}{T} & \frac{1}{T} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 1 & -\frac{1}{T} & \frac{1}{T} \\ 0 & 0 & 1 \end{pmatrix},\right.$$

$$\left.\begin{pmatrix} -T & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & T & -T \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 1 & -\frac{1}{T} & \frac{1}{T} \\ 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & T & -T \end{pmatrix}\right\}$$

```
In[•]:= matrix = Apart[IdentityMatrix[n - 1] - Dot @@ burauMatrices];

In[•]:= Apart /@ {MatrixForm[matrix], Det[matrix], (1 - T)/(1 - T^n) Det[matrix]}
```

$$\text{Out}[\bullet]= \left\{\begin{pmatrix} 2 - \frac{1}{T^3} + \frac{2}{T^2} - \frac{2}{T} & 1 + \frac{1}{T^4} - \frac{3}{T^3} + \frac{4}{T^2} - \frac{3}{T} & -1 + \frac{1}{T^3} - \frac{2}{T^2} + \frac{3}{T} \\ -2 + \frac{1}{T} + T & -1 - \frac{1}{T^2} + \frac{3}{T} + T & 2 - \frac{1}{T} - T \\ -T & 1 - T + T^2 & 1 + T - T^2 \end{pmatrix},\right.$$

$$\left. 1 - \frac{2}{T^3} + \frac{3}{T^2} + \frac{1}{T} + 3T - 2T^2, -\frac{2}{T^3} + \frac{5}{T^2} - \frac{2}{T}\right\}$$

```
In[•]:= (poly /. T → 1) poly / T^Mean[{Exponent[poly, T, Max], Exponent[poly, T, Min]}] // Apart
```

$$\text{Out}[\bullet]= 5 - \frac{2}{T} - 2T$$

2.5.4 Complete Code

```
AlexanderViaBurau[K_] := T ↦ Apart@Module[{m, n, br, reducedBurau,
    poly, δ, σ},
  {n, br} = List @@ BR[K];
  δ /: δ_i_,j_ := If[i == j, 1, 0];
  σ_i_[ℰ_] := {ℰ /. v_1 :> -T * v_1 + v_2                 i === 1
               ℰ /. v_i :> T * v_i-1 - T * v_i + v_i+1    1 < i < n - 1;
               ℰ /. v_n-1 :> T * v_n-2 - T * v_n-1        i === n - 1
  m_i_ := σ_i[Table[v_j, {j, n - 1}]] /. v_j_ :> Table[δ_j,k, {k, n - 1}];
  reducedBurau = Dot @@ br /. i_Integer :> If[i > 0, m_i, m_Abs[i] // Inverse];
  poly = (1 - T)/(1 - T^n) Det[IdentityMatrix[n - 1] - reducedBurau] // Apart;
  (poly /. T → 1) poly / T^Mean[{Exponent[poly,T,Max],Exponent[poly,T,Min]}]];
```

2.6 Seifert Matrix (via Braids)

This is a summary of [7].

2.6.1 Formula

Start by finding a braid whose closure is K and represent it as a product of generators σ_k, as in 2.5.1. Writing this product as $\sigma_{k_1}^{s_1} \cdots \sigma_{k_n}^{s_n}$, with $s_i \in \{-1, 1\}$, let $x_i = s_i k_i$. For example, the braid in figure 11 is equal to the product

$$\sigma_1^{-1}\sigma_1^{-1}\sigma_1^{-1}\sigma_2^{-1}\sigma_1\sigma_2^{-1} = \sigma_1^{-3}\sigma_2^{-1}\sigma_1\sigma_2^{-1},$$

and the x_i would be

$$(x_1, x_2, x_3, x_4, x_5, x_6) = (-1, -1, -1, -2, 1, -2).$$

See 2.6.3 for another such example.

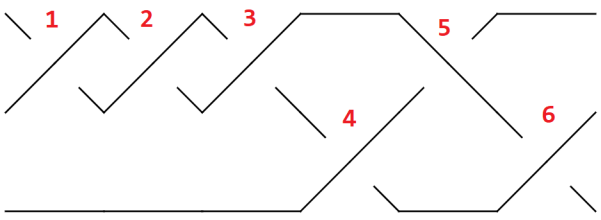


Figure 11: A braid with its crossings enumerated.

Define an array h of length $(n-1)$ with the property that the i'th entry of h is j, where j is the smallest index greater than i such that $k_j = k_i$, and $j = 0$ if

there are no further occurrences of k_i. For example, in figure 11, $h = [2, 3, 5, 6, 0]$. See 2.6.3 for another example of this calculation.

Create the $(n-1) \times (n-1)$ matrix $A = (a_{ij})$ according to the following rules, applying the *first* rule whose conditions are satisfied[13]:

- If $h[i] = 0$ or $h[j] = 0$, $a_{ij} =$*Nothing*.
- If $h[\min(i,j)] > h[\max(i,j)]$ or $h[\min(i,j)] < \max(i,j)$, $a_{ij} = 0$.
- If $i = j$:
  - If crossing i and crossing $h[i]$ are of opposite signs, $a_{ij} = 0$.
  - If both crossing i and crossing $h[i]$ are positive, $a_{ij} = -1$.
  - If both crossing i and crossing $h[i]$ are negative, $a_{ij} = 1$.
- If $h[i] = j$, and crossing j is negative, $a_{ij} = -1$.
- If $h[j] = i$, and crossing i is positive, $a_{ij} = 1$.
- If $i < j$ and $|x_i| - |x_j| = -1$, $a_{ij} = 1$.
- If $j < i$ and $|x_j| - |x_i| = 1$, $a_{ij} = -1$.
- Else, $a_{ij} = 0$.

Removing the positions of A marked *Nothing* gives a square matrix V, which is known to be a Seifert matrix for K. The following gives the Alexander polynomial for K, up to a unit:

$$\det(V - TV^{\mathsf{T}}),$$

where V^{T} is the transpose.

Note that $\det(V - TV^{\mathsf{T}})$ evaluated at 1 must be ± 1 as it differs from the Alexander polynomial by a unit, and, since $V - V^{\mathsf{T}}$ is skew-symmetric, its determinant must be non-negative. It follows that the sign is correct.

To symmetrize, multiply by $T^{-\text{Length}(V)/2}$, where $\text{Length}(V)$ is the height of V (which is also its width, since it is a square matrix). Note that $\text{Length}(V)$ is even.[14] Then, with $\Delta_K(T) := T^{-\text{Length}(V)/2} \det(V - TV^{\mathsf{T}})$,

$$\begin{aligned}\Delta_K(T^{-1}) &= T^{\text{Length}(V)/2} \det(-T^{-1}(V - TV^{\mathsf{T}})) \\ &= T^{\text{Length}(V)/2} T^{-\text{Length}(V)} \det(V - TV^{\mathsf{T}}) \\ &= \Delta_K(T).\end{aligned}$$

[13] This is important as the conditions are not disjoint.

[14] One way to see this is that $\det(V - V^{\mathsf{T}})$ is not zero.

2.6.2 Implementation

First off, we set *br* to be a braid whose closure is K. See 2.5.2 for more details.

```
br = BR[K][[2]];
```

Next, we compute the h array.

```
h = With[{absBR = Abs /@ br},
  Table[Append[Position[absBR[[i + 1 ;;]], absBR[[i]] + i, {0}][[1, 1]],
   {i, Length@absBR - 1}]];
```

Since negative signs are used purely to indicate that a crossing has a negative sign, this doesn't affect the computation of the h array so we take the absolute value of each entry in the *br* array.

Next, we compute the Seifert matrix. This is essentially identical to the set of rules described in 2.6.1.

```
V = Table[Which[
    h[[i]] × h[[j]] === 0, Nothing,
    h[[Min[i, j]]] > h[[Max[i, j]]], 0,
    h[[Min[i, j]]] < Max[i, j], 0,

    i === j, -(Sign[br[[i]]] + Sign[br[[h[[i]]]]])/2,
    h[[i]] === j ∧ br[[j]] < 0, -1,
    h[[j]] === i ∧ br[[i]] > 0, 1,
    i < j ∧ Abs[br[[i]]] - Abs[br[[j]]] === -1, 1,
    j < i ∧ Abs[br[[j]]] - Abs[br[[i]]] === 1, -1,
    True, 0],
   {i, Length@h}, {j, Length@h}] /. {} → Nothing;
```

Mathematica automatically removes elements of lists which have value *Nothing*. Since matrices are encoded as lists of lists, the lists whose elements are exclusively valued *Nothing* are empty but not automatically removed. To ensure their removal, we replace them with *Nothing*.

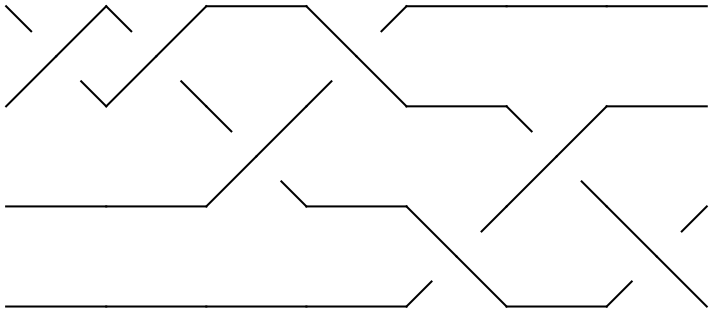
Finally, we take the determinant of $V - TV^{\mathsf{T}}$ and normalize the result by multiplying by $T^{-Length(V)/2}$.

```
T^-(Length[V]/2) Det[V - T * Transpose[V]]]
```

2.6.3 Run-through

```
In[•]:= K = Knot[6, 1];

In[•]:= BR[K]

Out[•]= BR[4, {-1, -1, -2, 1, 3, -2, 3}]

In[•]:= Show[BraidPlot[BR[K]]]
```

Out[•]=



```
In[•]:= h

Out[•]= {2, 4, 6, 0, 7, 0}

In[•]:= MatrixForm[V]
```

ut[•]//MatrixForm=

$$\begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

```
In[•]:= {MatrixForm[V - T * Transpose[V]], Det[V - T * Transpose[V]]}
```

Out[•]=

$$\left\{\begin{pmatrix} 1-T & -1 & 0 & 0 \\ T & 0 & 1 & 0 \\ 0 & -T & 1-T & 1 \\ 0 & 0 & -T & -1+T \end{pmatrix}, -2\,T + 5\,T^2 - 2\,T^3\right\}$$

```
In[•]:= T^(-Length[V]/2) Det[V - T * Transpose[V]] // Apart
```

Out[•]= $5 - \frac{2}{T} - 2\,T$

2.6.4 Complete Code

```
AlexanderViaSeifertAsBraids[K_] := T ↦ Apart@Module[{br, h, V},
    br = BR[K][[2]];
    h = With[{absBR = Abs /@ br},
      Table[Append[Position[absBR[[i + 1 ;;]], absBR[[i]] + i, {0}][[1, 1]],
       {i, Length@absBR - 1}]];
    V = Table[Which[
       h[[i]] × h[[j]] === 0, Nothing,
       h[[Min[i, j]]] > h[[Max[i, j]]], 0,
       h[[Min[i, j]]] < Max[i, j], 0,

       i === j, -(Sign[br[[i]]] + Sign[br[[h[[i]]]]])/2,
       h[[i]] === j ∧ br[[j]] < 0, -1,
       h[[j]] === i ∧ br[[i]] > 0, 1,
       i < j ∧ Abs[br[[i]]] - Abs[br[[j]]] === -1, 1,
       j < i ∧ Abs[br[[j]]] - Abs[br[[i]]] === 1, -1,
       True, 0],
      {i, Length@h}, {j, Length@h}] /. {} → Nothing;
    T^-(Length[V]/2) Det[V - T * Transpose[V]]]
```

3 Acknowledgement and References

I would like to thank Prof. Dror Bar-Natan for his guidance and feedback, and for suggesting this topic.

References

[1] K. Miller. All the ways i know how to define the alexander polynomial.

[2] P. Lee. Proof of a conjectured formula for the alexander invariant. *arXiv:1209.0668*, 2018.

[3] D. Bar-Natan and Z. Dancso. Finite type invariants of w-knotted objects: From alexander to kashiwara and vergne.

[4] D. Bar-Natan and R. Van Der Veen. A perturbed-alexander invariant. *arXiv:2206.12298v2*, pages 2–3.

[5] A. Stipsicz P. Ozsvath and Z. Szabo. *Grid Homology for Knots and Links.*

[6] J. Birman. *Braids, links, and mapping class groups.*

[7] J. Collins. An algorithm for computing the seifert matrix of a link from a braid representation.