From Knot Groups to Keis

Tongchen He

June 17, 2022

Abstract

Kei homomorphisms can be used to verify the Meridional Rank Conjecture using. By designing and using a python algorithm, we have found that out of 59937 knots under crossing number 14, 31385 admit a homomorphism to at least one connected kei under order 36 [2], which thus verify the Meridional Rank Conjecture for those knots.

1 Introduction

In this research, we designed an algorithm to search for surjective homomorphisms from fundamental keis of knots to keis. The existence of such homomorphism can be used to verify the Meridional Rank Conjecture on certain knots, which asks whether the bridge number of a knot is equal to its meridional rank.

Definition 1.1. Let h be a standard height function where $h : \mathbb{R}^3 \to \mathbb{R}, h(x, y, z) = z$, and K be a knot in $\mathbb{R}^3$. The minimal number of maxima of $h(K)$ over all ambient isotopic embeddings is the *bridge number* of K, denoted $\beta(K)$.

Definition 1.2. The *fundamental group* for a knot, or the *knot group*, denoted $\pi(\mathbb{R}^3 \setminus K, x_0)$, is a group that is an invariant of every path-connected topological space, whose elements are path-homotopy classes of loops based at a point x_0

Definitions 1.3. A *meridian* is a loop bounding a disk that intersects the knot at least once. The *meridional rank* of a knot K, denoted $mr(K)$, is the minimal number of meridian loops to generate $\pi(\mathbb{R}^3 \setminus K, x_0)$.

It is commonly known that the bridge number $\beta(K)$ is an upper bound of $mr(K)$. Therefore, the main goal of our algorithm is to establish that $\beta(K)$ is also a lower bound of $mr(K)$ so that they end up being equal. How we accomplish that will be discussed in Sections 3 and 5.

2 Background

Definition 2.1. A *kei* [3] is a set X with an operation $\triangleright : X \times X \to X$ that satisfies the following axioms:

1. For all $x \in X, x \triangleright x = x$,
2. For all $x, y \in X, (x \triangleright y) \triangleright y = x$,
3. For all $x, y, z \in X, (x \triangleright y) \triangleright z = (x \triangleright z) \triangleright (y \triangleright z)$.

And the operation $\triangleright$ is called a *kei operation*

Example 2.1.1. If we take the kei $X = \mathbb{Z}_4$ with operation $x \triangleright y = 2y - x$, we get the operation table:

$\triangleright$	**0**	**1**	**2**	**3**
0	0	2	0	2
1	3	1	3	1
2	2	0	2	0
3	1	3	1	3

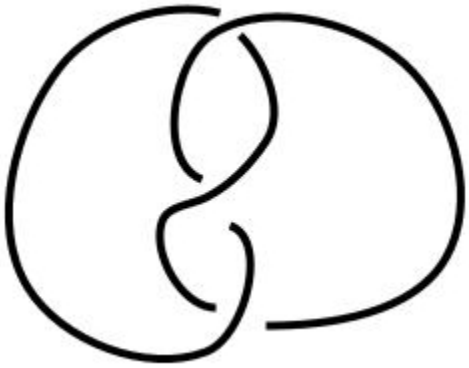
Definition 2.2. The *rank* of a kei Q, denoted $rank(Q)$, is the cardinality of the smallest generating set of Q.

Definition 2.3. A *connected kei* is a kei X where there exists an element $a \in X$ such that for all $c \in X$, there is a $b \in X$ where $a \triangleright b = c$.

Corollary 2.3.1. If X is a connected kei, then for every $a \in X$, for all $c \in X$, there is a $b \in X$ where $a \triangleright b = c$.

Definition 2.4. A *knot* is a smooth embedding of a circle into $\mathbb{R}^3$.
Two knots are equivalent if they are related by a smooth ambient isotopy.

Definition 2.5. A *knot diagram* is a regular of projection of a knot onto a two-dimensional plane together with crossing information. The following is a knot diagram of a trefoil knot:



Prelimilary. Let D be a knot diagram of knot K and let $W(D)$ be a subset of the set of strands in D, refered to as *colored strands*. At any crossing c of D, there are two understrands u_1, u_2 and an overstrand o. When $\{u_1, o\} \subset W(D)$ and $u_2 \notin W(D)$, we can perform a *coloring move* on c, by setting $W'(D) := W(D) \cup \{u_2\}$. And we can write $W(D) \to W'(D)$ for this coloring move. If $W(D) := \{s_1, s_2, ..., s_n\}$, and after a certain number of coloring moves,

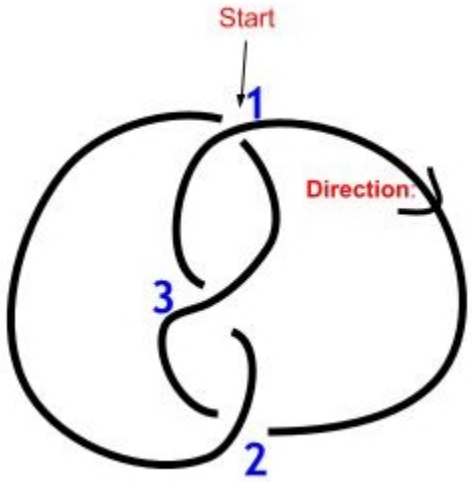
$$W(D) \to W_1(D) \to W_2(D) \to ...W_m(D)$$

we get $W_m(D)$ that contains all strands in D, then we can call the initial set $W(D)$ a seed strand set of knot D and $s_1, s_2, ..., s_n$ the seed strands.

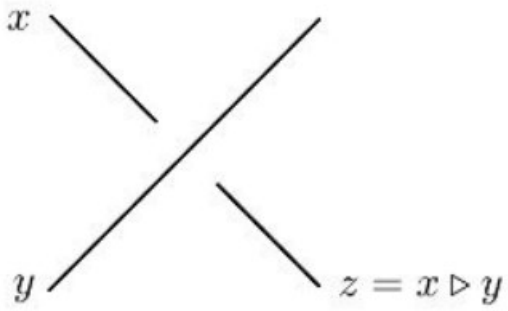
Definition 2.6. The *Wirtinger Number* of a knot K, denoted $w(K)$, is the cardinality of the smallest seed strand set of K.

Notation 2.7. A *Gauss Code* of a knot is a representation of the knot by a sequence of integers. Each crossing is labeled with an integer n, and when traversing through the knot, an overcrossing is noted as a positive number n, and an undercrossing is noted as a negative number n.

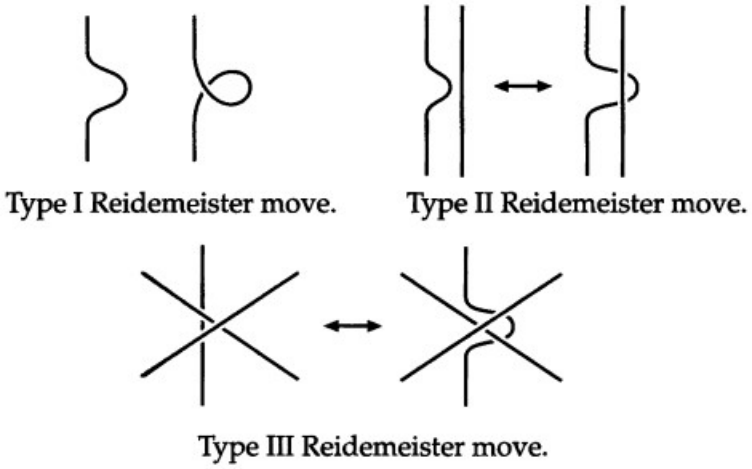
Example 2.7.1. The Gauss code of the trefoil knot can be: 1,-2,3,-1,2,3.
Note: Gauss code of a knot is not unique.



Definition 2.8. A *fundamental kei* of a knot has elements as strands of the knot and the operation $\triangleright$ defined as the following figure:



Definitions 2.9. A *Reidemeister move* [1] is one of three local moves on a knot diagram:



Although each of these moves changes the projection of the knot, it does not change the knot represented by the projection.

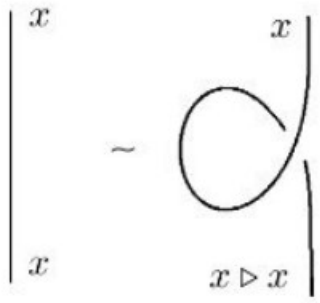
Note that in a fundamental kei, each of the three kei axioms corresponds to one type of Reidemeister move, as figure 1 shows. This is very important because it enables us to correlate fundamental keis to fundamental groups of knots.

3 About The Algorithm

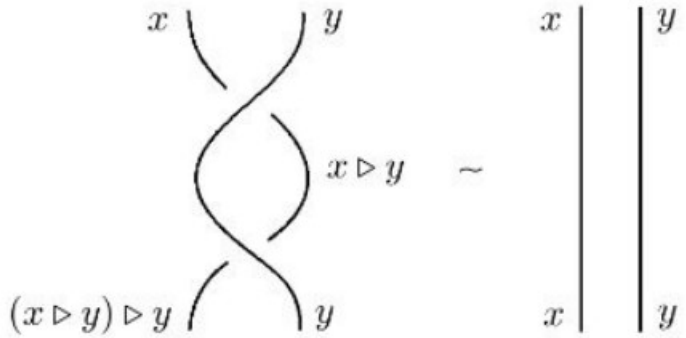
This section discusses the algorithm that is used to find the results presented in Section 4. The algorithm conducts a search for a surjective homomorphism from the fundamental kei of a knot K to a connected kei Q, given that $rank(Q) = w(K)$. The existence of such homomorphism will prove the Meridional Rank Conjecture on the knot K (as discussed later in Section 5. This algorithm also makes use of [5] which, given the gauss code g_K of a knot K, calculates the Wirtinger number $w(K)$, finds the seed strand set of a coloring of the knot,

Figure 1: Kei Axioms on Knots [3]

Kei axiom (i) follows from the type I Reidemeister move:



Kei axiom (ii) follows from the type II Reidemeister move:



Kei axiom (iii) follows from the type III Reidemeister move:

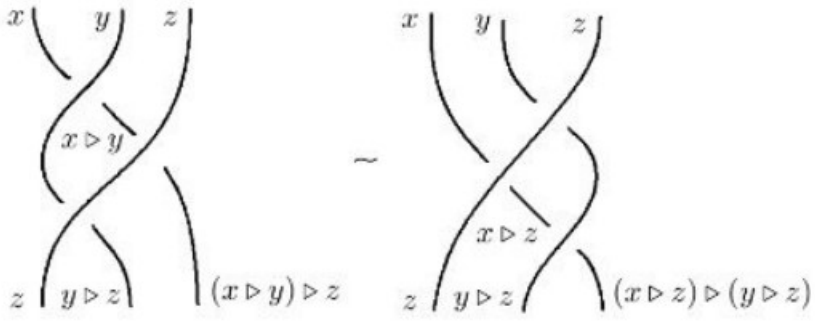
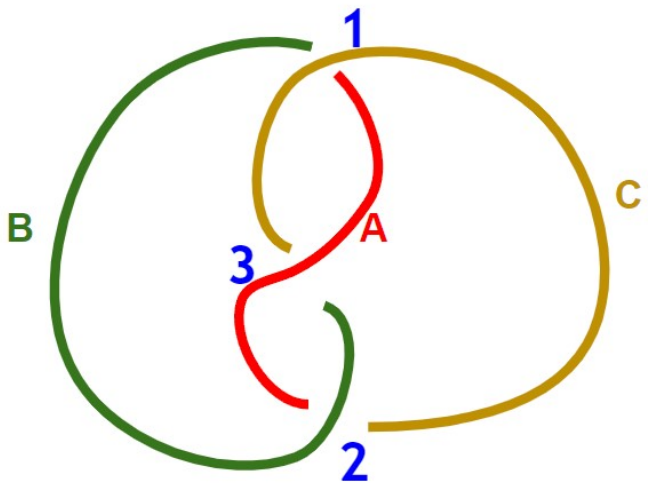


Figure 2: The trefoil knot with labelings



and creates a knot dictionary consisting of each strand's label. The keys of the dictionary are the strands, labeled as alphabetic letters, and the values contain (a) a sequence of positive and negative numbers representing the gauss code of the strand and (b) every pair of its understrands.

Example 3.1. If we input the gauss code of a trefoil knot g_K = [1, -2, 3, -1, 2, -3], the algorithm will output Wirtinger number 2, knot dictionary of strand labeling D_K ={A: [(-2, 3, -1), (C, B)], B: [(-1, 2, -3), (A, C)], C: [(-3, 1, -2), (B, A)]}, and seed strand set s_K = (A, B).
For example, the labeling {A: [(-2, 3, -1), (C, B)]} means that the strand A goes from the understrand at crossing #2, as an overstrand through crossing #3, and ends as an understrand at crossing #1. And it has understrands C and B at crossing #3.

This information is vital to our algorithm as it searches for a surjective homomorphism. Based on the Nathaniel Morrison's code [4], the main idea of our algorithm is to assign each seed strand to an element in the kei's generating set, and check if each crossing satisfies the kei operation. The algorithm can be broken into these steps:

1. Given the multiplication table and the rank of a kei, find every minimal set of elements that generate the entire kei. [1]
2. For every kei generating set, check to see if there is a valid mapping to a knot (meaning that every strand is mapped to an element of Q, and every crossing follows the kei

[1]Note: In this program, we labeled kei element by integers from 0 to n-1 instead of 1 to n (where n is the total number of elements in the kei) to match with python indexing, which also starts at 0. However, this makes no difference in the mathematical sense since they are only representations of the elements.

Table 1: Multiplication Table of the Three-Element Kei Q_3

$\triangleright$	**a**	**b**	**c**
a	a	c	b
b	c	b	a
c	b	a	c

operation $x \triangleright y = z$, where the understrands of the crossing maps to $x, z \in Q$, and the overstrand maps to $y \in Q$) by doing the following:

(a) Assign each seed strand to a generator
(b) Map every strand to a kei element $x \in Q$, according to the kei operation
(c) Check if it is a valid mapping, and if so, output the mapping. Otherwise repeat the process on other generating sets until a valid homomorphism is found, or if no valid mapping is found, output a "false" indicator.

Now, here is a more detailed description of how we perform each step.
(1) *Finding minimal generating sets of a kei* using a function called **generator_finder**. As previously mentioned, this step requires the rank $rank(Q)$ and the multiplication table T of a kei Q (order n) to be executed (we obtained these information from [2]). Once these are passed into the function, we can create a list containing every length-r ($r = rank(Q)$) combinations of numbers from 0 to $n-1$.
For each of theses combination $C \subset Q$, we declare a new set of generated elements $G_C = C$. Then, we take every pair of elements $a, b \in G_C$, and in T, shade the cells $(a,b), (b,a), (a,a)$, and (a,b). If, in the shaded area, there is a new element $c \notin G_C$, we add c to the set G_C and shade the cells $(a,c), (b,c)$, and etc.
The program will run the process until no new elements can be added to G_C. If G_C ends up containing every element in Q, then C is a generating set of the kei Q.

Example 3.2. For the 3-element kei, we first test out the combination {a,b} by shading the cells (a,b), (b,a), (a,a), and (b,b). Now $G_C = \{a, b\}$. As we can see from the Table 1, the shaded area contains a new element c. Thus, we add c to G_C, so it now contains all the elements $\{a, b, c\}$ of the kei. Because of this, the set $\{a, b\}$ is a minimal generating set and will be returned at the end of the function.

(2) *Searching for a valid mapping from the kei to a knot.* Once we obtain the minimal generating sets from step (1), along with the information of a knot K from [5], including the knot dictionary, we can proceed with the function **homomorphism_finder**. This function will run the following on every generating set S_Q of Q:
(2)(a)*Assigning each seed strand to an element in the generating set.* We first obtain all permutations of the seed strands before calling the function **generator_assign.** For each of these permutation, the function will create a dictionary D_p with its keys being seed strands and values being a generator in corresponding order (i.e. $\{A : a\}$).

Example 3.3. As a continuation of Examples 3.1 and 3.2. One of the seed strand assignment dictionaries we'll get from step (2)(a) is $D_S = \{A : a, B : b\}$.

(2)(b)*Mapping each strand to a kei element.* In this step, we will assign each strand a kei element according to the operation table T, using the function **strand_assignment**. The programs runs through every strand/element pairing $(s_O : x) \in D_p$ and check each of the crossings where S_O is an overstrand. If for any of these crossings, one of the understrand $(u_1 : y) \in D_p$ and the other understrand u_2 has not been assigned a kei element, then we will assign u_2 to an element $z \in Q$ such that $x \triangleright y = z$ and add $(u_2 : z)$ to D_p. The program will iterate this process until no more assignment can be made, and then we will proceed to the step (2)(c)

Example 3.4. Using the trefoil knot dictionary D_K ={A: [(-2, 3, -1), (C, B)], B: [(-1, 2, -3), (A, C)], C: [(-3, 1, -2), (B, A)]}, the kei Q_3 and the see strand mapping $D_S = \{A : a, B : b\}$, we can see that strand A has understrands C and B at crossing 3. Since A is mapped to element a and B to element b, and $b \triangleright a = c$, we will assign c to strand C.

(2)(c)*Checking if the mapping is valid.* Once we have mapped each strand to a kei element, we run the function **hom_tester** on the mapping. The function will run through every pairing $(s_O : x) \in D_p$. For each crossing where s_O is an overstrand, the function grabs the understrands' pairing $(u_1 : y)$ and $(u_2 : z)$ and check if x, y, and z follows the kei operation $y \triangleright x = z$. If any of the crossing doesn't follow the kei operation, then the mapping is invalid and we return the "false" indicator and conclude that there is no valid mapping between the fundamental kei of the knot and the kei that we are testing it on. Otherwise, if the mapping is valid, we will return the mapping dictionary D_p.

4 Results

We ran the algorithm on every kei under order 36 [2] and every knot under crossing number 14. We found that out of the total of 59,937 knots, 31,385 have a kei homomorphisms to at least one kei under order 36. Thus, we have verified the Meridional Rank Conjecture on approximately 52.4% of the knots under crossing number 14. It is also worth noting that we have found kei homomorphisms to 63 of the 63 keis under order 36 of rank 2 to 5. Nothing is found for rank 6 and 7 keis because there is no knots of wirtinger number 6 or 7 to be mapped from.

5 Relating to MRC

In this section, we will discuss how the algorithm proves the Meridional Rank Conjecture for the knots that admits a homomorphism to a kei. First, let us look at a theorem about fundamental keis, whose definition is given at 2.8.

Table 2: Kei with homomorphism from knots

Rank	Total Keis	Hm from knots
2	17	17
3	27	27
4	14	14
5	5	5
6	2	0
7	1	0
Total	66	63

Table 3: Knots with kei homomorphisms

Crossing number	Prime Knots	Hms to keis	Keis
3	1	1	1
4	1	1	1
5	2	2	2
6	3	3	4
7	7	7	9
8	21	19	20
9	49	37	37
10	165	122	43
11	552	364	54
12	2176	1275	57
13	9988	5421	58
14	46972	24133	63
Total	59937	31385	

Theorem 5.1. The fundamental kei of a knot is a complete invariant (up to reflection). Meaning that if two knots K_1 and K_2 have kei isomorphic fundamental keis, then K_1 is equivalent to K_2 (or the reflection of K_2).

Important Fact: the minimum generators needed to generate the fundamental kei of knot Q_K (denoted $rank(Q_K)$) is equal to the meridional rank $mr(K)$ of the knot. The proof of this is fairly simple, but if you wonder what it is, please ask Dr. Ryan Blair.

Lemma 5.2. If Q_1 and Q_2 are keis and $f : Q_1 \to Q_2$ is a surjective kei homomorphism, than $rank(Q_2) \leq rank(Q_1)$.

Now let P be a kei, Q_K be the fundamental kei of a knot K. The aim of our algorithm is to find the surjective homomorphism $f : Q_K \to P$. By Lemma 5.2, the existence of such homomorphism f proves that

$$rank(P) \leq rank(Q_K) = mr(K)$$

Table 4: Knots with $w(K) = 2$ to rank 2 keis

Crossing number	Knots with $w(K) = 2$	Kei hms to rank 2 keis	Rank 2 Keis
3	1	1	1
4	1	1	1
5	2	2	2
6	3	3	4
7	7	7	9
8	12	12	11
9	24	17	15
10	45	24	16
11	91	46	16
12	176	89	17
13	352	199	17
14	693	391	17
Total	1407	792	

Table 5: Knots with $w(K) = 3$ to rank 3 keis

Crossing number	Knots with $w(K) = 3$	Kei hms to rank 3 keis	Rank 3 Keis
≤ 7	0	0	0
8	9	7	9
9	25	20	22
10	120	98	27
11	446	285	27
12	1952	1138	27
13	8614	4644	27
14	39291	21699	27
Total	50457	27891	

Remember in the algorithm, we picked the kei P such that $rank(P) = w(K)$, added to the well-known facts that the bridge number of knot K satisfies $\beta(K) \leq w(K)$ and $\beta(K) \geq mr(K)$, we can combine the the equations and inequalities to get

$$\beta(K) \leq w(K) = rank(P) \leq rank(Q_K) = mr(K) \leq \beta(K).$$

Since everything in the inequality is bounded by $\beta(K)$, they ended up all being equal to each other. Therefore, $mr(K) = \beta(K)$, and we have successfully proved the Meridional rank conjecture on the knot K.

Table 6: Knots with $w(K) = 4$ to rank 4 keis

Crossing number	Knots with $w(K) = 4$	Kei hms to rank 4 keis	Rank 4 Keis
≤ 10	0	0	0
11	15	15	11
12	48	48	13
13	1022	578	14
14	6958	2013	14
Total	8043	2654	

Table 7: knots with $w(K) = 5$ and rank 5 keis

Crossing number	Knots with $w(K) = 5$	Kei hms to rank 5 keis	Rank 5 Keis
≤ 13	0	0	0
14	30	30	5

6 Acknowledgement

I would like to thank my thesis advisor Dr. Ryan Blair for his guidance on knot theory and kei theory, as well as his assistance in constructing this research.

References

[1] Colin C. Adams. *The Knot Book: An Elementary Introduction to the Mathematical Theory of Knots*. American Mathematical Soc., 1994.

[2] W. Edwin Clark and Timothy Yeatman. Small connected quandles and their knot colorings. `http://shell.cas.usf.edu/~saito/QuandleColor/`, 2014.

[3] Mohamed Elhamdadi and Sam Nelson. *Quandles: An Introduction to the Algebra of Knots*. American Mathematical Soc., 2015.

[4] Nathaniel Morrison. Wirt hm. `https://github.com/ThisSentenceIsALie/Wirt_Hm`, 2022.

[5] Paul Villanueva. Calc wirt. `https://github.com/pommevilla/calc_wirt`, 2018.