An Integrated Software SKB of SnapPea, K2K, and bTd

（SnapPea, K2K, and bTdのJavaによる統合化ソフトウェアSKB）

by M. Ochiai and F. Kako

講演概要

- SnapPea、K2K、bTdをJavaインタフェースを利用して統合化したソフトウェアSKBについて紹介する。
- 主な機能は、

1. 結び目のマウス入力、ブレイド入力と結び目成分の追加機能、正則射影とブレイド表示との相互変換
2. 結び目補空間の基本群、isometry群によるミュータント結び目の識別、ノーマルサーフェスの探索
3. Hecke環の線形表現による不変量計算
4. 基底タングル分解とタングルの細分
5. 大量データの結び目の自明性判定（（1，1）サージェリと基本群）、ミュータント結び目の類別（体積計算とHOMFLY多項式）
6. 結び目テーブルからのamphiceiral結び目の探索
7. WeeksによるMacOSX版SnapPeaについて

SKBの動作環境

- システムの入力インターフェースはJavaで、プログラムの主要な部分はC言語で記述されている. グラフィックスライブラリとしてglutを利用している.
- Windows Vista, XP, MacOSX Leopard, Linux上で動作する. Windows Vista, XP ,MacOSX Pantherでは一部の機能が正しく動作しない. SnapPeaを除くほとんどのプログラムは落合が作成し, Linux上への移植を加古が行った。加古は17から21交点までの結び目表のデータ作成を行った.　ただし,21交点の非交代結び目については計算中である.

簡単な実験結果 1

- amphiceiral結び目の探索: nonalternate15.prdについて探索したところ唯一の結び目を見つけた(amphiceiral5.png). これは数年前にThislethwaiteが発見し,論文で発表しているものである.
- 8,14交点でdegenerate solutionがいくつか現れた(degenerate16_fig1, degenerate16_fig2)、その内の1つは2つの非圧縮トーラスを含む?(補空間内のノーマルサーフェスの探索機能の実現).
- タングルの細分機能の実現 (bTd).
- 3-平行化不変量が一致し、4-平行化不変量が異なる12交点以上の結び目を発見した(bTd).
- ほとんどの結び目は双曲的結び目である(Thurstonの定理). 19交点までの非交代結び目の数千万個のデータの内,非双曲的結び目は千個以内である.

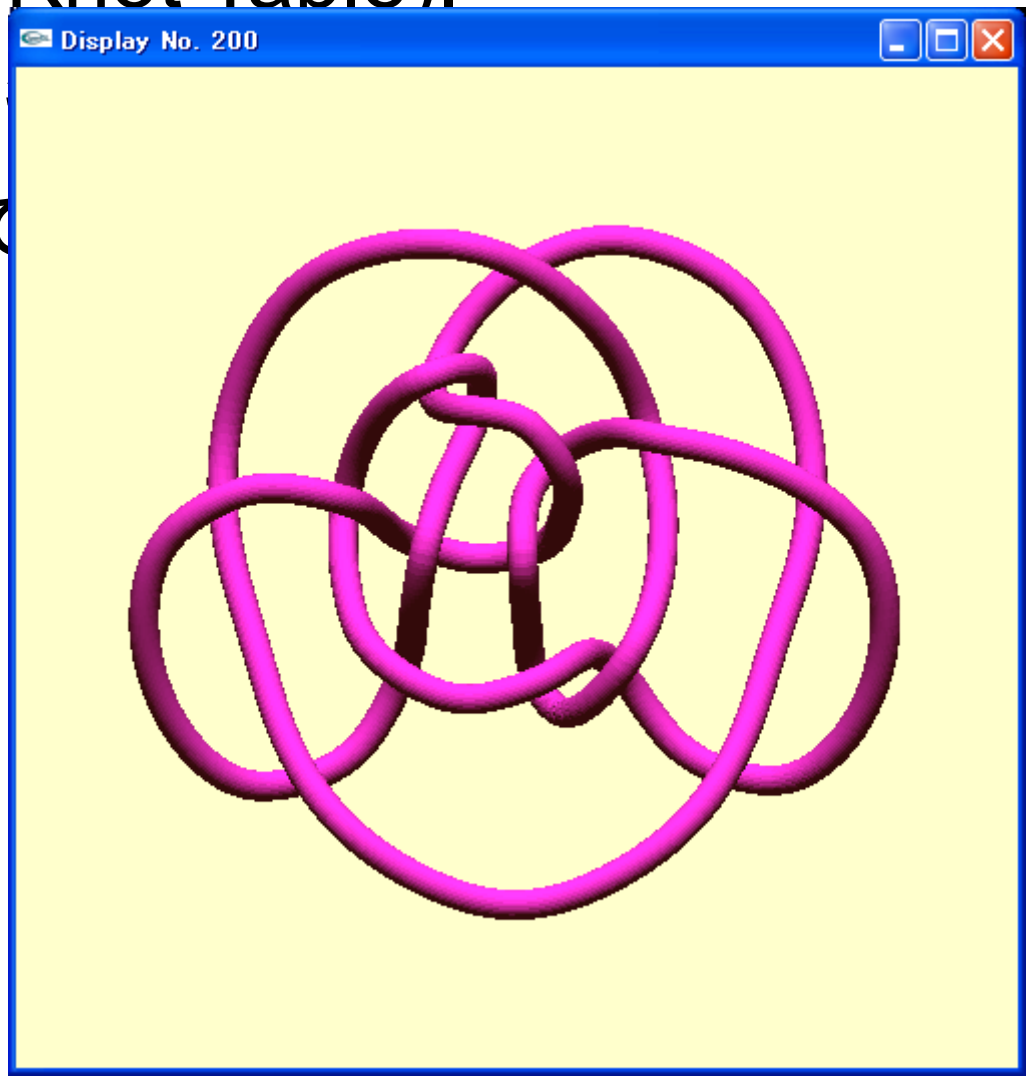
簡単な実験結果 2

- 18交点までの非交代結び目表のすべての結び目に対して、nongeometric solutionと表示されるデータに対して,今回発見したプログラムコードを適用した結果,すべてgeometric solutionという結果を得た。この実験では, not_attempted, other_solution, no_solutionという出力は現れていない.また、17交点ではflat solutionは現れなかった.これはトーラス結び目の最小交点数に関する公式から得られる結果と一致する(村杉先生の指摘).
- Bug: (1) 2-橋結び目を入力したとき、(p, q)を出力する機能があるが正しく動作しない（Weeksによれば旧版では動作した？,Sakuma-Weeks予想の解決）

  (2) surgeryして得られる多様体に対してisometryが動作しない（Weeksによれば、マニュアルのミス）.

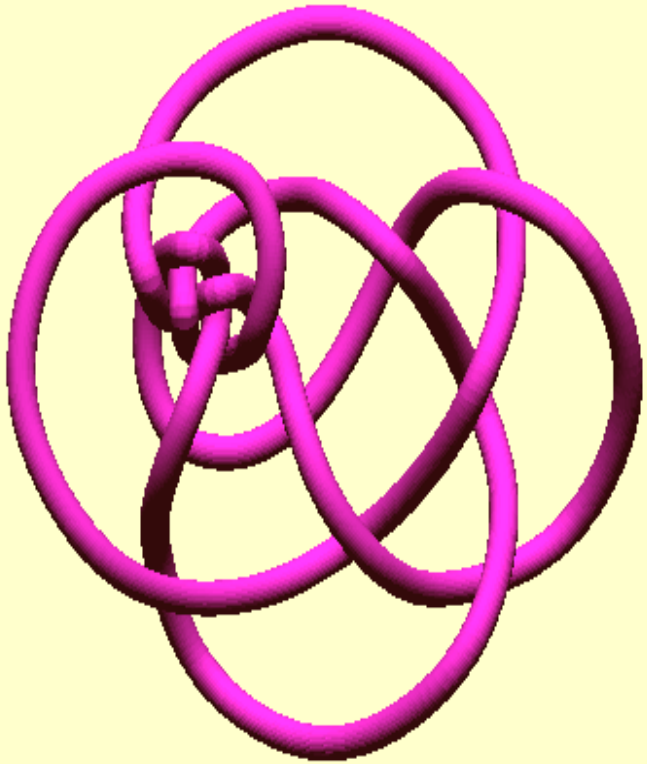
15交点のamphicheiral結び目

- 139717-th p-data is an amphicheiral knot (Thislethwaite's Knot Table).
- 15交点の非交代
- Windows XP上で



degenerate16_fig1

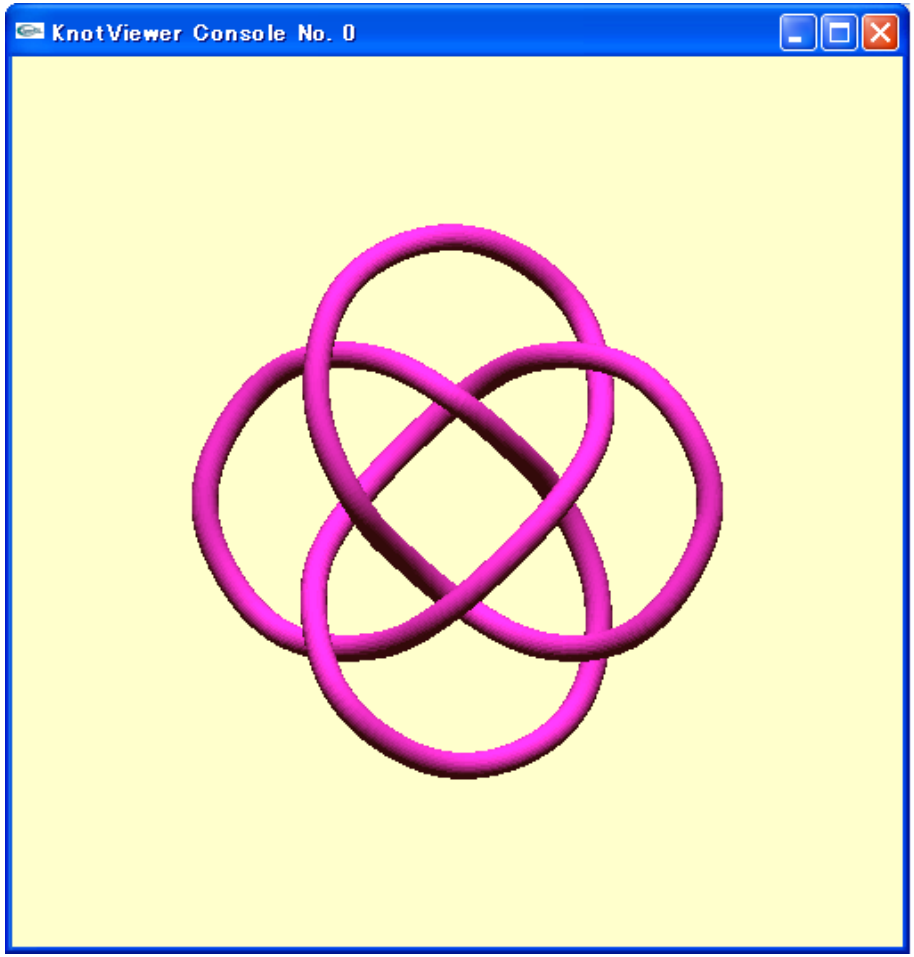
Display No. 207



Console No. 207

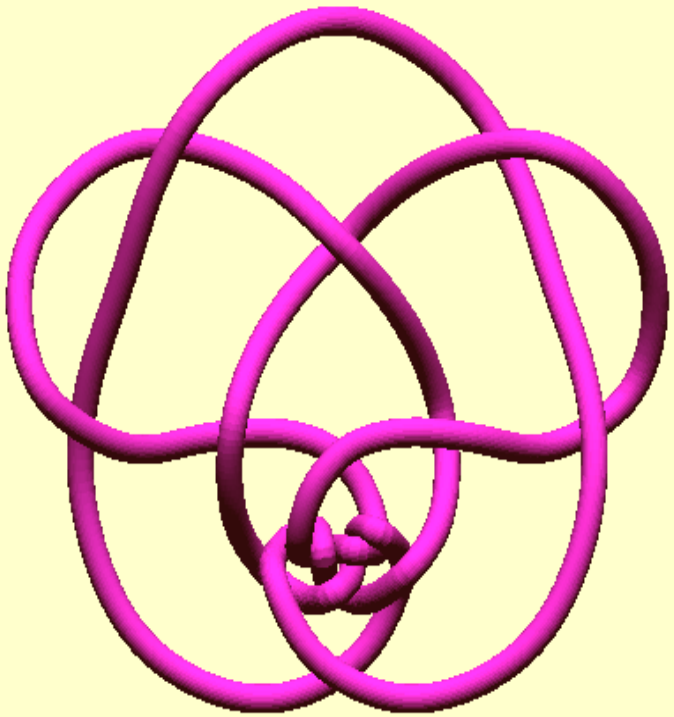
```
Normal surface decomposition:
The 0-th normal surface is orientable.
The 0-th normal surface is 2-sided.
The 0-th normal surface has 0 as Euler characteristic.
The volume of (0,0)-th piece = 0.000000
The fundamental group of (0,0)-th piece:
number of generators = 2
number of relations = 1
aB^2a^2BAb^2A^2b
----------------------------------------
The volume of (0,1)-th piece = -0.000000
The fundamental group of (0,1)-th piece:
number of generators = 2
number of relations = 1
a^2b^3
----------------------------------------
The 1-th normal surface is orientable.
The 1-th normal surface is 2-sided.
The 1-th normal surface has 0 as Euler characteristic.
The volume of (1,0)-th piece = 0.000002
The fundamental group of (1,0)-th piece:
number of generators = 2
number of relations = 1
a^2B^2abA^2b^2AB
----------------------------------------
The volume of (1,1)-th piece = 0.000500
The fundamental group of (1,1)-th piece:
number of generators = 2
number of relations = 1
a^2b^3
```

degenerate16_fig1のReidemeister変形



degenerate16_fig2

Display No. 208



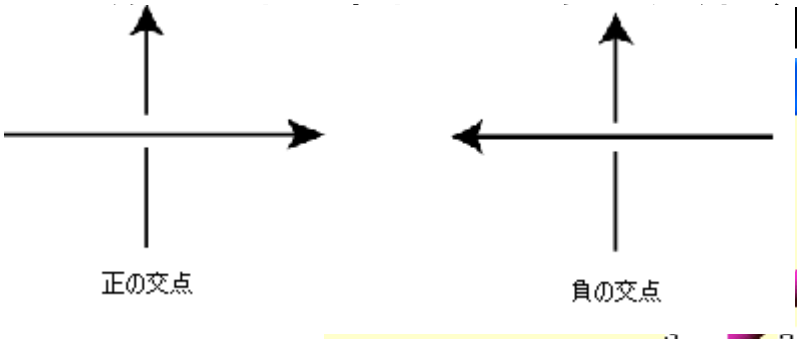
Console No. 208

```
Normal surface decomposition:
The 0-th normal surface is orientable.
The 0-th normal surface is 2-sided.
The 0-th normal surface has 0 as Euler characteristic.
The volume of (0,0)-th piece = -0.000000
The fundamental group of (0,0)-th piece:
number of generators = 3
number of relations = 2
aBab^2
aCAc
----------------------------------------
The volume of (0,1)-th piece = 0.000500
The fundamental group of (0,1)-th piece:
number of generators = 2
number of relations = 1
a^2b^3
----------------------------------------
The 1-th normal surface is orientable.
The 1-th normal surface is 2-sided.
The 1-th normal surface has 0 as Euler characteristic.
The volume of (1,0)-th piece = -0.000000
The fundamental group of (1,0)-th piece:
number of generators = 3
number of relations = 2
ac^2aC
aBAb
----------------------------------------
The volume of (1,1)-th piece = 0.000500
The fundamental group of (1,1)-th piece:
number of generators = 2
number of relations = 1
a^2b^3
```

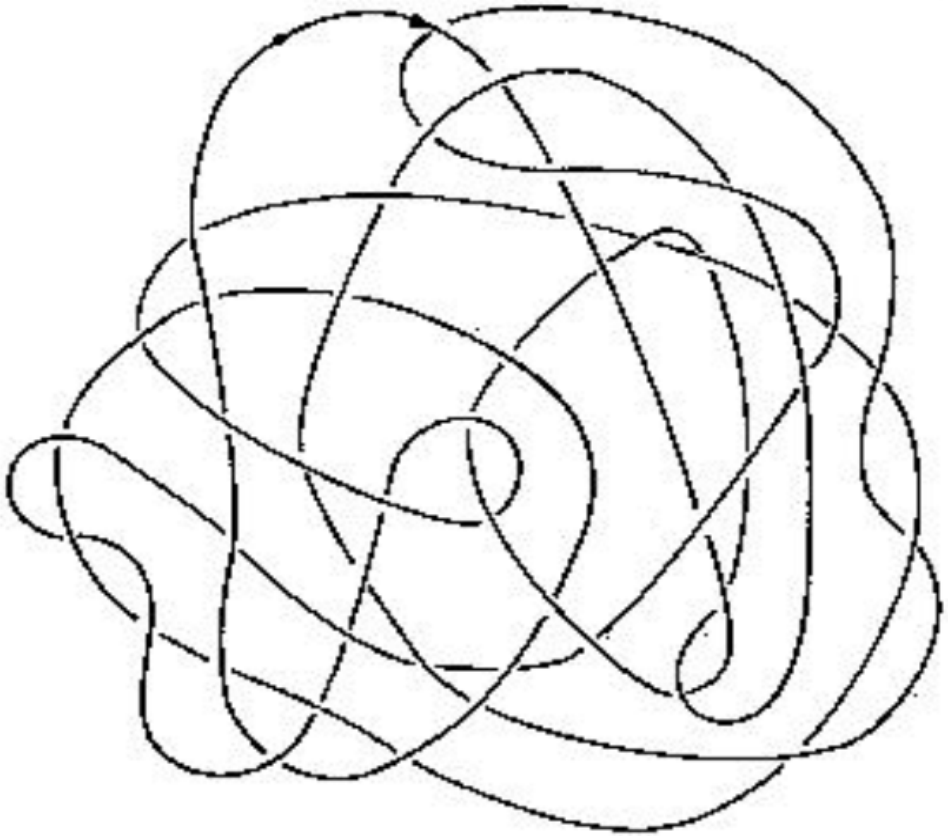
K2Kのデータ構造：p-データ

向きの付いた絡み目の各成分に出発点を決め,各結び目の出発点から向きに沿って交点を通るごとに順に数字を割り振る.交点数がnなら2n個の番号が割り振られる.
再度,各結び目の出発点から向きに沿って,上方交点を通るごとに下方交点の番号を

目または絡み目のp-データと呼ぶ.

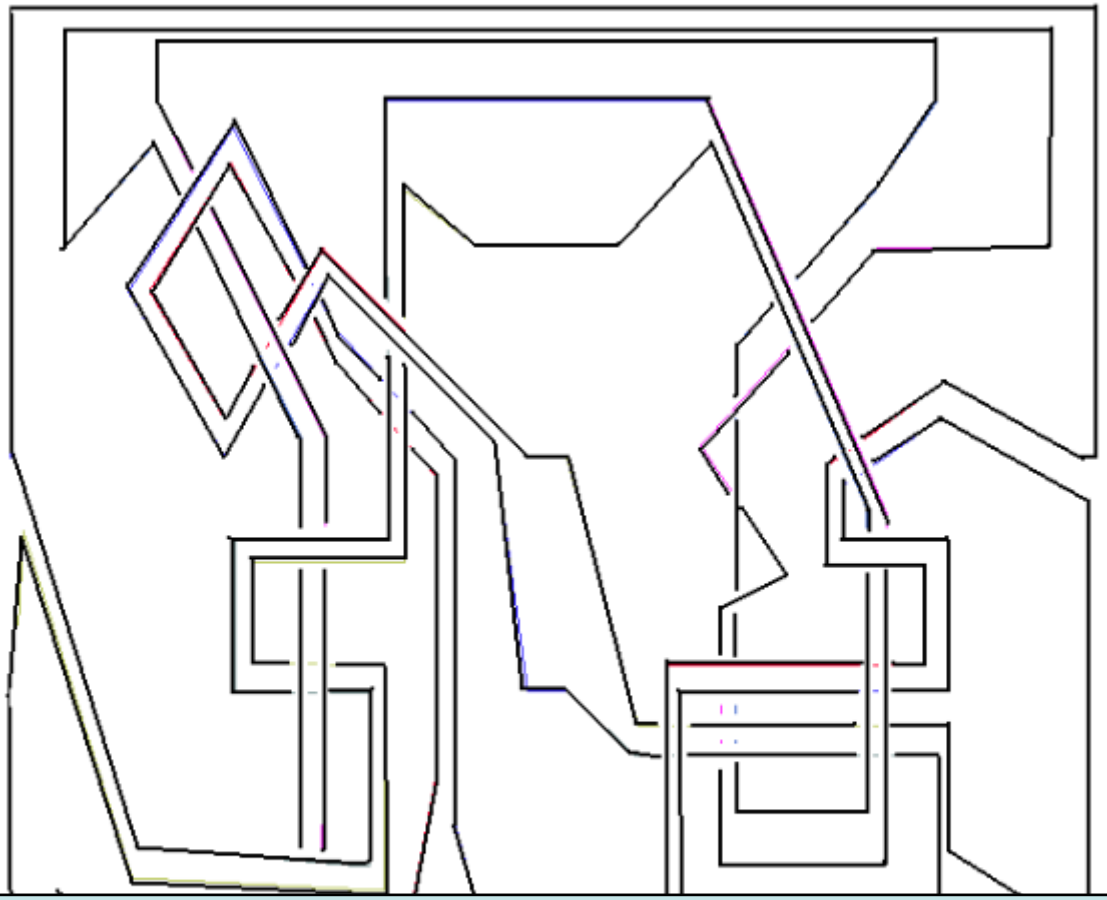


A trivial knot without waves with 45 crossing points



90 1
90
-30 -13 45 18 -20 61 -48 78 31 -43 -40 -59 -75 -66 -3 64 -37 79 -63 6 -17 -15 -39 85 74 24 69 -88 -23 -21 -47 12 56
89 -25 52 -83 -27 35 10 49 -73 77 28 -57

A trivial knot without waves with 67 crossing points



134 1
134
-88 -21 -108 -41 -126 -59 -114 -47 26 93 -68 -1 16 83 62 129 54 121 -76 -9 -72 -5 -94 -27
134 67 38 105 34 101 -78 -11 130 132 -89 -22 -109 -42 -127 -60 -115 -48 25 92 -69 -2 15 82
61 128 53 120 -77 -10 -73 -6 -95 -28 133 66 37 104 33 100 -79 -12 64

SnapPeaのデータ構造

```
% Link projection by K2K programmed by M. Ochiai.
1
0 0
5
121 418 496 418 162 171 301 543 488 169
5
0 1
1 2
2 3
3 4
4 0
5
2 0
0 3
3 1
1 4
4 2
-1
```

対応するp-データ(外部データ):

```
10 1
10
-7 -9 -1 -3 -5
```

対応する完全p-データ(内部データ):

```
    1  2  3  4  5   6  7  8  9 10
a  6  7  8  9 10  1  2  3  4   5
b  2  1  2  1   2  1  2  1  2   1
c  1  1  1  1   1  1  1  1  1   1
```

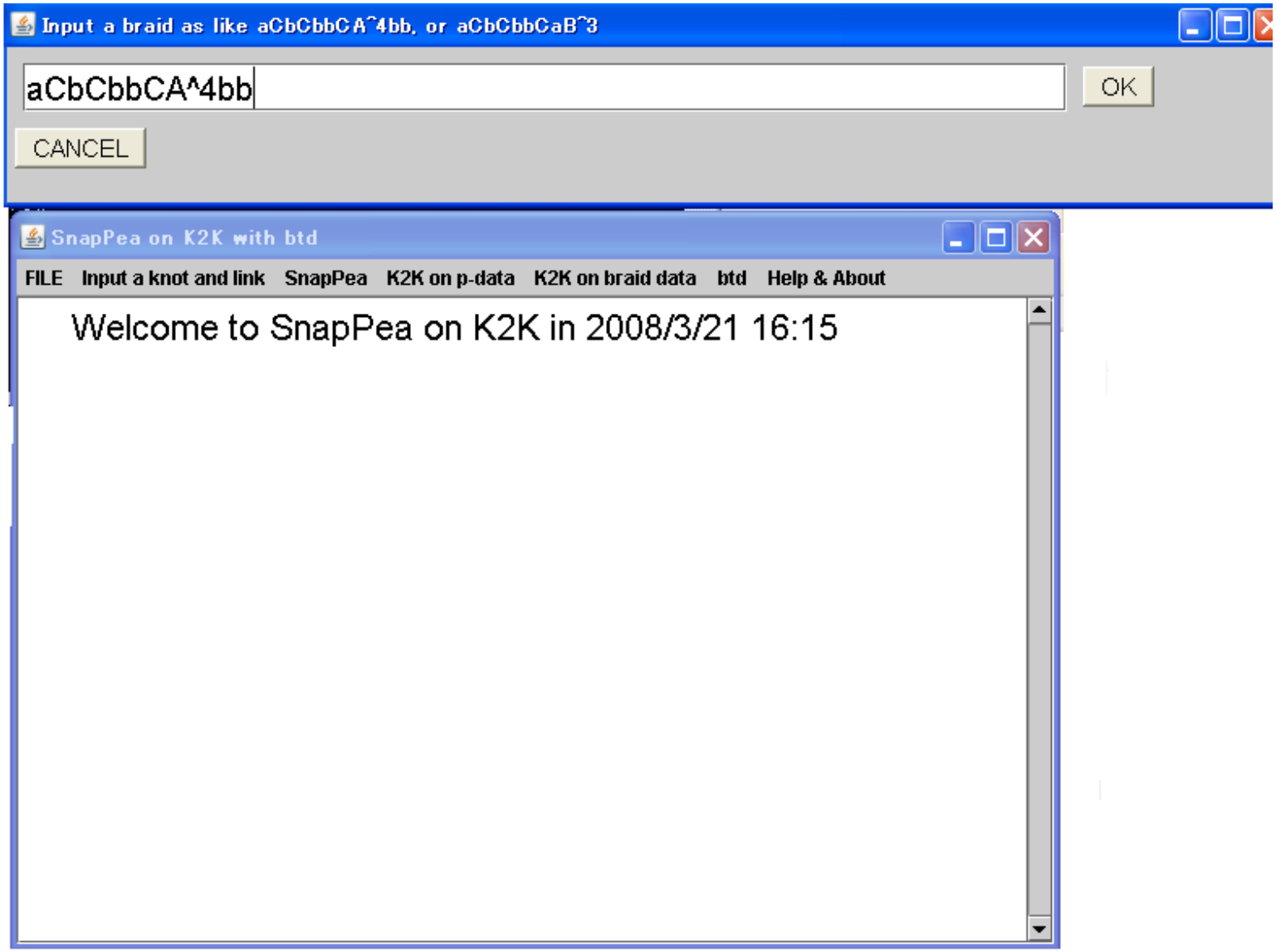
→ ”結び目ピクチャ1”参照

ブレイド入力、ブレイド表示

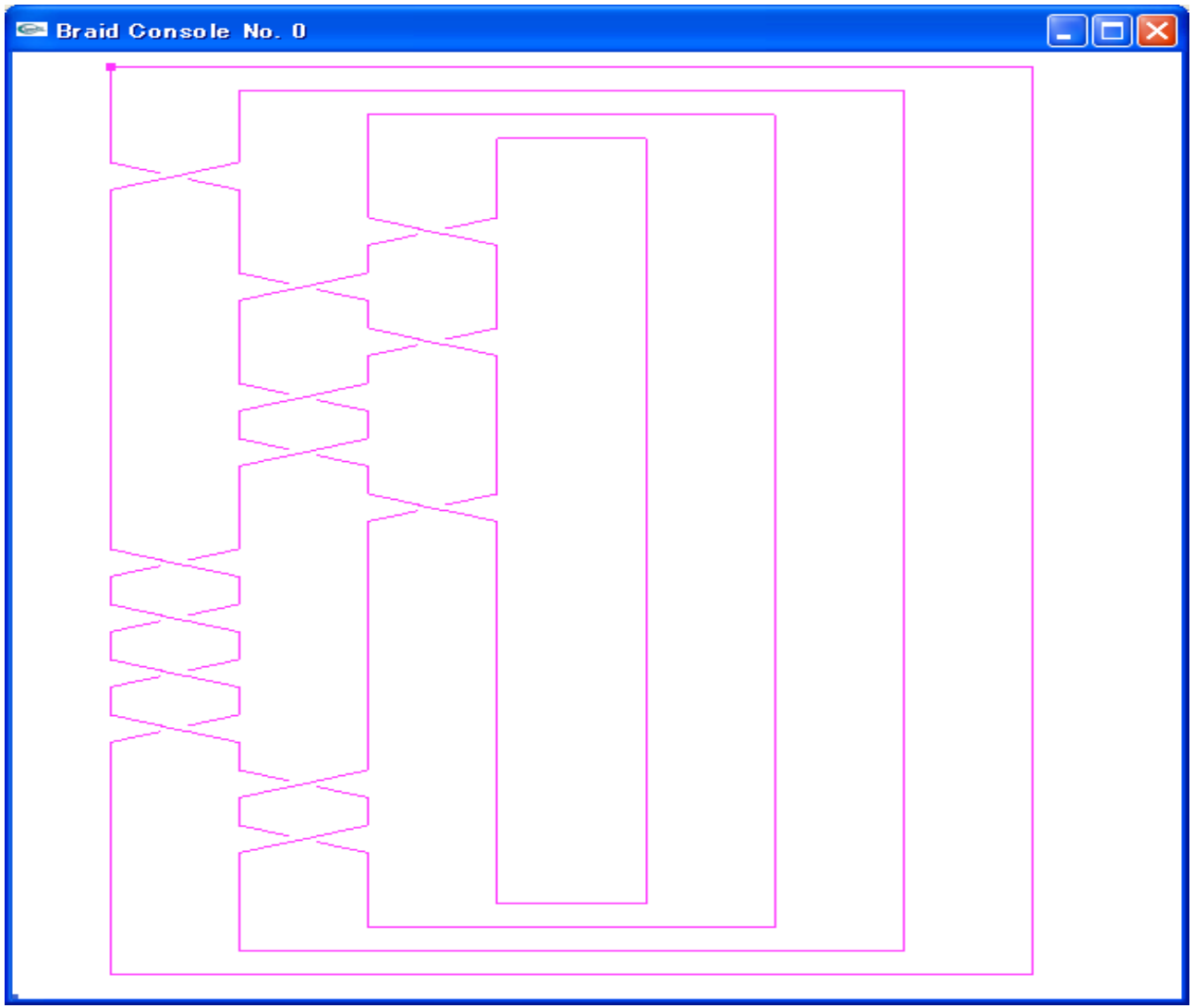
- 樹下-寺阪結び目 > aCbCbbCA^4bb
- コンウェー結び目 > aCbCbbCaB^3

大文字は逆元で、ベキ乗は“^4”で表す.

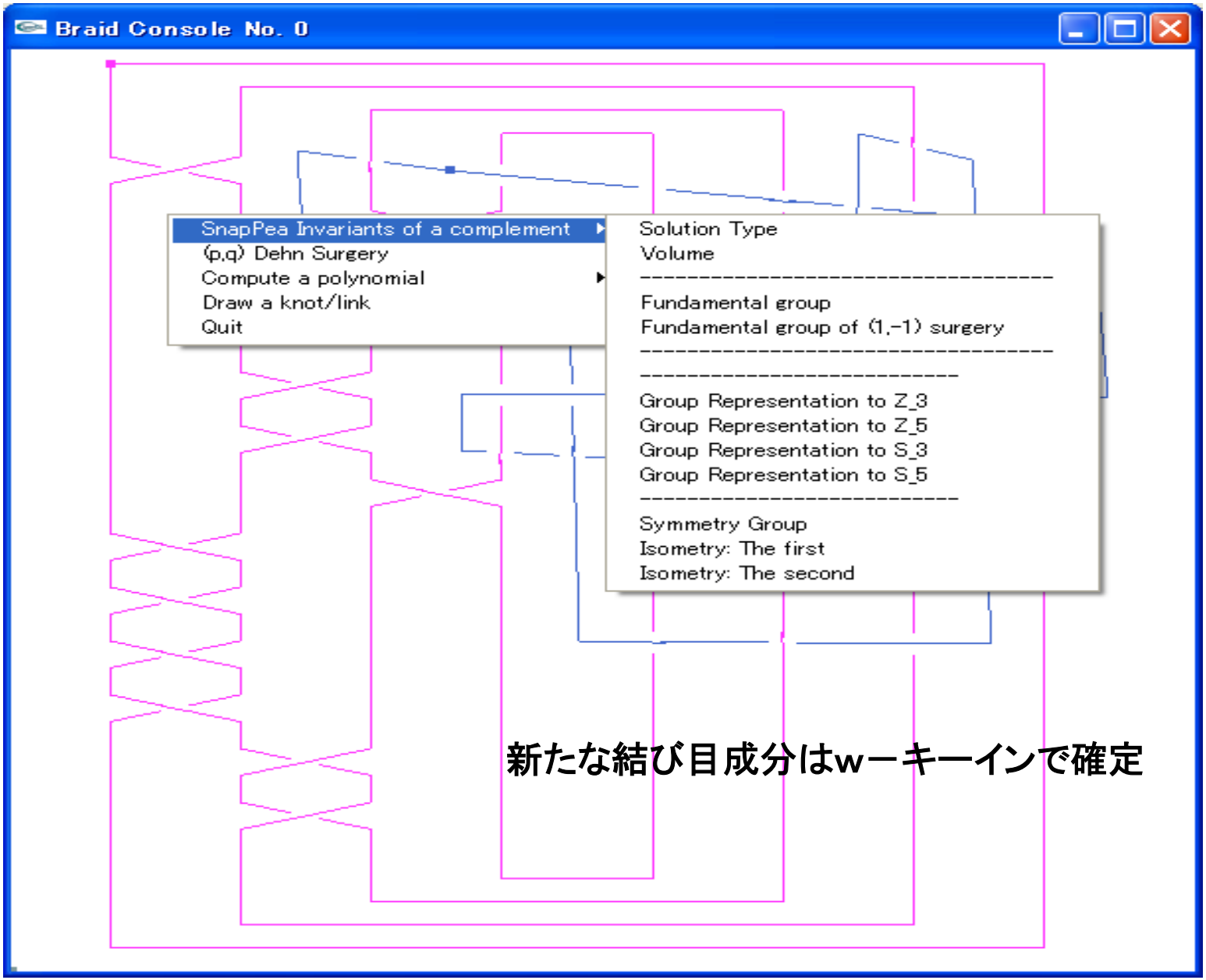
ブレイド入力実行


Input a braid as like aCbCbbCA^4bb, or aCbCbbCaB^3
aCbCbbCA^4bb
OK
CANCEL
SnapPea on K2K with btd
FILE
Input a knot and link
SnapPea
K2K on p-data
K2K on braid data
btd
Help & About
Welcome to SnapPea on K2K in 2008/3/21 16:15

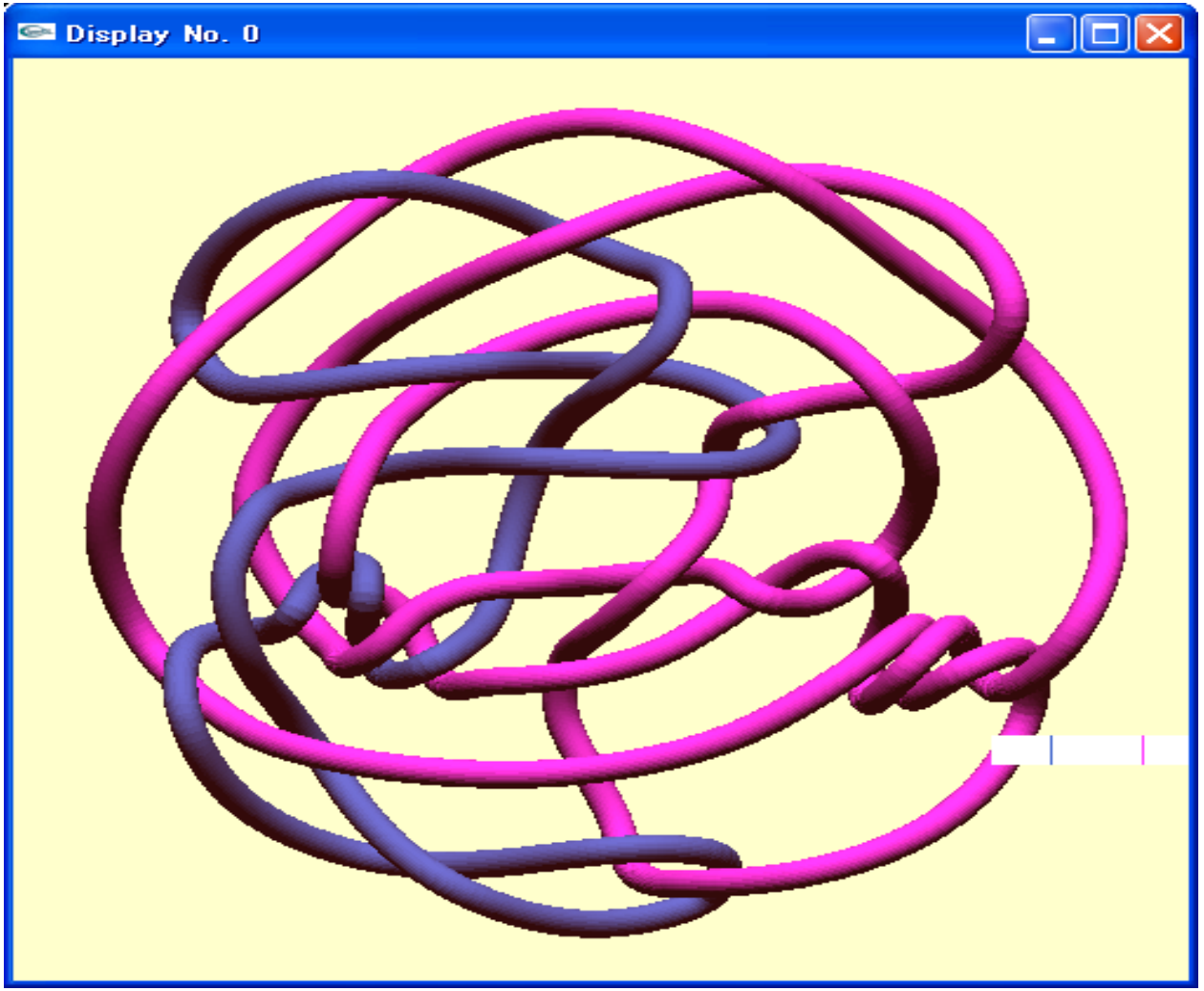
実行結果(1)



実行結果(2)


Braid Console No. 0
SnapPea Invariants of a complement
(p,q) Dehn Surgery
Compute a polynomial
Draw a knot/link
Quit
Solution Type
Volume
Fundamental group
Fundamental group of (1,-1) surgery
Group Representation to Z_3
Group Representation to Z_5
Group Representation to S_3
Group Representation to S_5
Symmetry Group
Isometry: The first
Isometry: The second
新たな結び目成分はwーキーインで確定

実行結果(3)



ドーカーコードによるデータ

19 1 4 8 10 -14 2 -16 -20 -6 -22 -26 -12 -28 -32 -18 -34 -36 -24 -38 -30
19 2 4 8 10 -14 2 -16 -20 -6 -22 -26 -12 -28 -32 -18 34 36 -24 38 30
19 3 4 8 10 -14 2 -16 -20 -6 -22 -26 -12 -28 32 -18 -34 -36 24 -38 -30
19 4 4 8 10 -14 2 -16 -20 -6 -22 -26 -12 -28 32 -18 34 36 24 38 30
19 5 4 8 10 -14 2 -16 -20 -6 -22 -26 -12 28 32 -18 34 36 24 38 30
19 6 4 8 10 -14 2 -16 -20 -6 -22 26 -12 -28 -32 18 -34 -36 -24 -38 -30
19 7 4 8 10 -14 2 -16 -20 -6 -22 26 -12 28 32 18 34 36 24 38 30
19 8 4 8 10 -14 2 16 20 -6 22 -26 12 -28 -32 -18 -34 -36 -24 -38 -30
19 9 4 8 10 -14 2 16 20 -6 22 -26 12 28 32 -18 34 36 24 38 30
19 10 4 8 10 -14 2 16 20 -6 22 26 12 -28 -32 18 -34 -36 -24 -38 -30

最初の19は交点数, 次の1,2,..,10は何番目,それ以降の19個の符号付き偶数列は奇数番目に対応する交点番号を表し, 正の場合は,上方交点、負の場合は下方交点を表す.上記のデータ列が示すように,ネット上で公開されているものは,辞書式順序で最小のものから順にソートされている.

インターネット上に16交点までの結び目表が公開されている. 17交点から20交点までの結び目表は加古が, 落合が1992年に作成したプログラムを改良して作成したもので, まだ重複データが含まれる. 加古作成の結び目表は, 既にホームページで公開されている. 重複データの排除に皆様の協力期待します！！！

ﾄﾞｰｶｰコードは絡み目を扱うには問題があるので, 拡張コードが必要である. 数列が結び目の正則表示を与えるかどうかを判定するためのﾄﾞｰｶｰアルゴリズムは, 金沢大学の岩瀬氏により絡み目の場合に拡張されている.

ﾄﾞｰｶｰコード以外にも, ガウスコードと呼ばれるものがある.

結び目表

n	3	4	5	6	7	8	9	10	11	12	13	14	15	16
t	1	1	2	3	7	21	49	165	552	2176	9988	46972	253293	1388705
a	1	1	2	3	7	18	41	123	367	1288	4878	19536	85263	379799
na						3	8	42	185	888	5110	27436	168030	1008906
k						3	8	43	191	935	5411	29702	184876	1152120

---

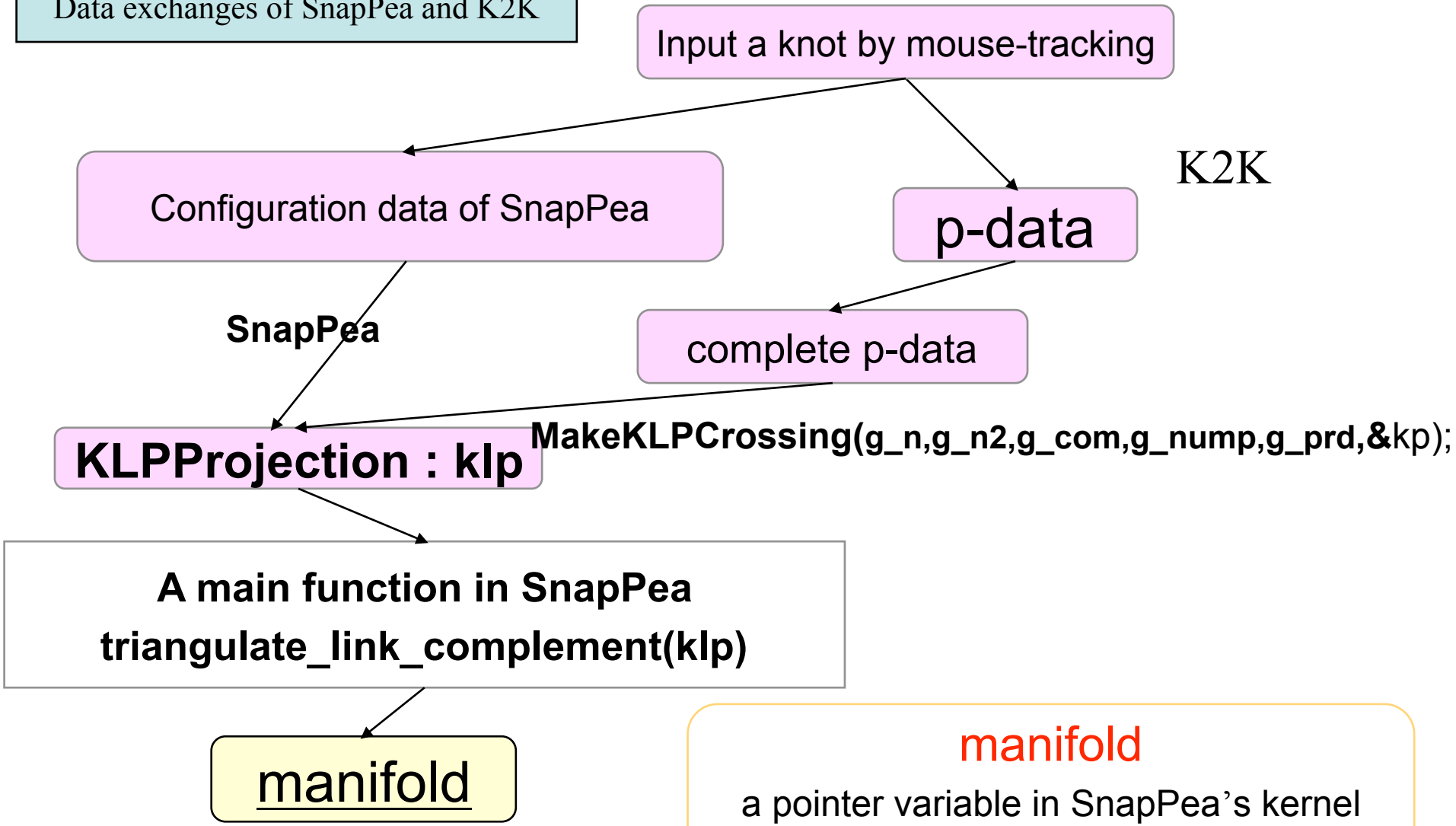
n	17	18	19	20	21
t					
a	1769979	8400285	40619385	199631989	990623857
na	*	*	*	*	*
k	7413688	49128005	326046639	2214670927	14370000000

交点数: n,　全体個数: t,　交代結び目個数: a,　非交代結び目個数: na
加古版個数 : k

* : 未確定,　k – na :重複する個数

Integration of SnapPea, K2K, and bTd

Data exchanges of SnapPea and K2K



manifold

a pointer variable in SnapPea's kernel

to compute invariants such as volume

MakeKLPCrossing(g_n, g_n2, g_com, g_nump, g_prd, &kp);

MakeKLPCrossing()は, 任意のp-data から SnapPea の KLPCrossing構造体を初期化する関数である.

nongeometric_solutionを回避するプログラムコードを理解するには, この関数を理解することが必要である.

この関数は, SKBのC言語プログラムファイル snappeajnilib.cで定義されている.

Flat, degenerate, geometric solution

- Snappea.h

```
typedef int SolutionType;
enum
{
    not_attempted,          /* solution not attempted, or user cancelled */
    geometric_solution,     /* all positively oriented tetrahedra; not flat or degenerate*/
    nongeometric_solution,  /* positive volume, but some negatively oriented tetrahedra*/
    flat_solution,          /* all tetrahedra flat, but no shapes = {0, 1, infinity}  */
    degenerate_solution,    /* at least one tetrahedron has shape = {0, 1, infinity} */
    other_solution,         /* volume <= 0, but not flat or degenerate               */
    no_solution             /* gluing equations could not be solved                  */
};
```

問題は”nongeometric_solutionを如何に回避するか？”

２－橋絡み目の場合は、標準的な4面体分割で,all positively oriented tetrahedraのみの分割が存在することが数学的に証明されている.

```
/*
 *      The KLPStrandType and KLPDirectionType enums are used to index
 *      array entires, so their values must be 0 and 1.  (But the code
 *      does not rely on which has value 0 and which has value 1.)
 *
 *      JRW 2000/11/12   Use fake "typedef enums" instead of real ones,
 *      for the reasons explained at the top of kernel_typedefs.h.
 */


/*
 *      If you view a crossing (from above) so that the strands go in the
 *      direction of the postive x- and y-axes, then the strand going in
 *      the x-direction is the KLPStrandX, and the strand going in the
 *      y-direction is the KLPStrandY.  Note that this definition does not
 *      depend on which is the overstrand and which is the understrand.
 *
 *                                                      KLPStrandY
 *                                                          ^
 *                                                          |
 *                                                     ----| ---> KLPStrandX
 *                                                          |
 *                                                          |
 */
```

構造体: KLPStrandType

```
typedef int KLPStrandType;
enum
{
   KLPStrandX = 0,
   KLPStrandY,
   KLPStrandUnknown
}; // link_projection.h
```

構造体: KLPDirectionType

```
typedef int KLPDirectionType;
enum
{
   KLPBackward = 0,
   KLPForward,
   KLPDirectionUnknown
}; // link_projection.h
```

構造体: KLPCrossingType

```
/*
*   A crossing is either a clockwise (CL) or counterclockwise (CCL)
*   half twist.
        KLPHalfTwistCL  : +         KLPHalfTwistCCL : -
*/

typedef int KLPCrossingType;
enum
{
    KLPHalfTwistCL,
    KLPHalfTwistCCL,
    KLPCrossingTypeUnknown
}; // link_projection.h
```

構造体: KLPProjection

```
struct KLPProjection
{
    /* *   How many crossings are there?        */
    int    num_crossings;

    /* * How many free loops (i.e. link components with no
    crossings) are there?  For a hyperbolic link, the number of free
    loops must be 0. */
    int    num_free_loops;

    /* *How many link components (including the free loops) are
    there? */
    int    num_components;

    /* *   Here's a pointer to the array of crossings.   */
    KLPCrossing          *crossings;
}; // link_projection.h
```

構造体: KLPCrossing

```
struct KLPCrossing
{
      /*  *       The four neighbors are
      *                        neighbor[KLPStrandX][KLPBackward]
      *                        neighbor[KLPStrandX][KLPForward ]
      *                        neighbor[KLPStrandY][KLPBackward]
      *                        neighbor[KLPStrandY][KLPForward ]
      *          For example, if you follow the x-strand in the backward direction,
      *          you'll arrive at neighbor[KLPStrandX][KLPBackward].              */
      KLPCrossing                       *neighbor[2][2];

      /* *       When you arrive at a neighbor, you could arrive at either the
      *          x-strand or the y-strand.  The strand[][] field says which it is.
      *          For example, if you follow the x-strand in the backward direction,
      *          you'll arrive at strand[KLPStrandX][KLPBackward].                */
      KLPStrandType       strand[2][2];

      /* *       The crossing is either a clockwise or counterclockwise half twist.   */
      KLPCrossingType    handedness;
      /* *       To which component of the link does each strand belong?  A link
      *          component is given by an integer from 0 to (#components - 1).
      *          For example, if component[KLPStrandX] == 0 and
      *          component[KLPStrandY] == 2, then the x-strand is part of component
      *          number 0 and the y-strand is part of component number 2.          */
      int       component[2];
}; // link_projection.h
```

nongeometric_solution
を回避するためのプログラムコード

```
if (g_com == 1 && solutiontype == nongeometric_solution) { // knot only
    count = 0;                          ",
    while (count++ < g_n2) {            // g_com: component number
       for (i=1; i<=g_n2; i++)          // g_n2: 2*total corossings"
          g_prd[i-1] = g_prd[i];        // g_prd[]: a pdata of the knot",
          g_prd[g_n2] = g_prd[0];       // g_nump[]: 2*total corossings of each component
          for (i=1; i<=g_n2; i++) {
             if (g_prd[i].a == 1)
                g_prd[i].a = g_n2;
             else
                g_prd[i].a -= 1;
          }                             ",
          free_triangulation(manifold_t);
          manifold_t = NULL;
          MakeKLPCrossing(g_n, g_n2, g_com, g_nump, g_prd, &kp);
          manifold_t = triangulate_link_complement(&kp);
          solutiontype = manifold_t->solution_type[complete];
          if (solutiontype == geometric_solution)
             goto next1; // success
    }
}
```

Nongeometric filteringにおけるプログラム制御ファイル

- ファイル名：KEY_degenerate_flat_geometric
- 内容

1
---------- Comment by M. Ochiai 2008/3/12 ----------
if this file does not exist, then computation is SnapPea's original one.
if shift_num == 0, then computation is SnapPea's original one.
if shift_num == 1, then SKB does try to obtain geometric solutions from non-geometric solutions.
if shift_num == 4, then SKB does the same as shift_num == 1 and also save pdata with geometric solutions.
if shift_num == 5, then SKB does the same as shift_num == 1 and also save pdata with original non-geometric solutions and the corresponding pdata with geometric solutions.
if shift_num == 6, then SKB does save all pdata converted from dt-code, when drawing is running.

このファイルの最初の数字のみが意味をもつ。特に、5のときは、プログラム内の変数shift_numに数字5が読み込まれ、このときは、non-geometric solution である結び目の初期データとgeometric solution であるシフト後のデータが同時にログファイルSKB.logに出力される.

Nongeometric ー＞ geometricに関する実験

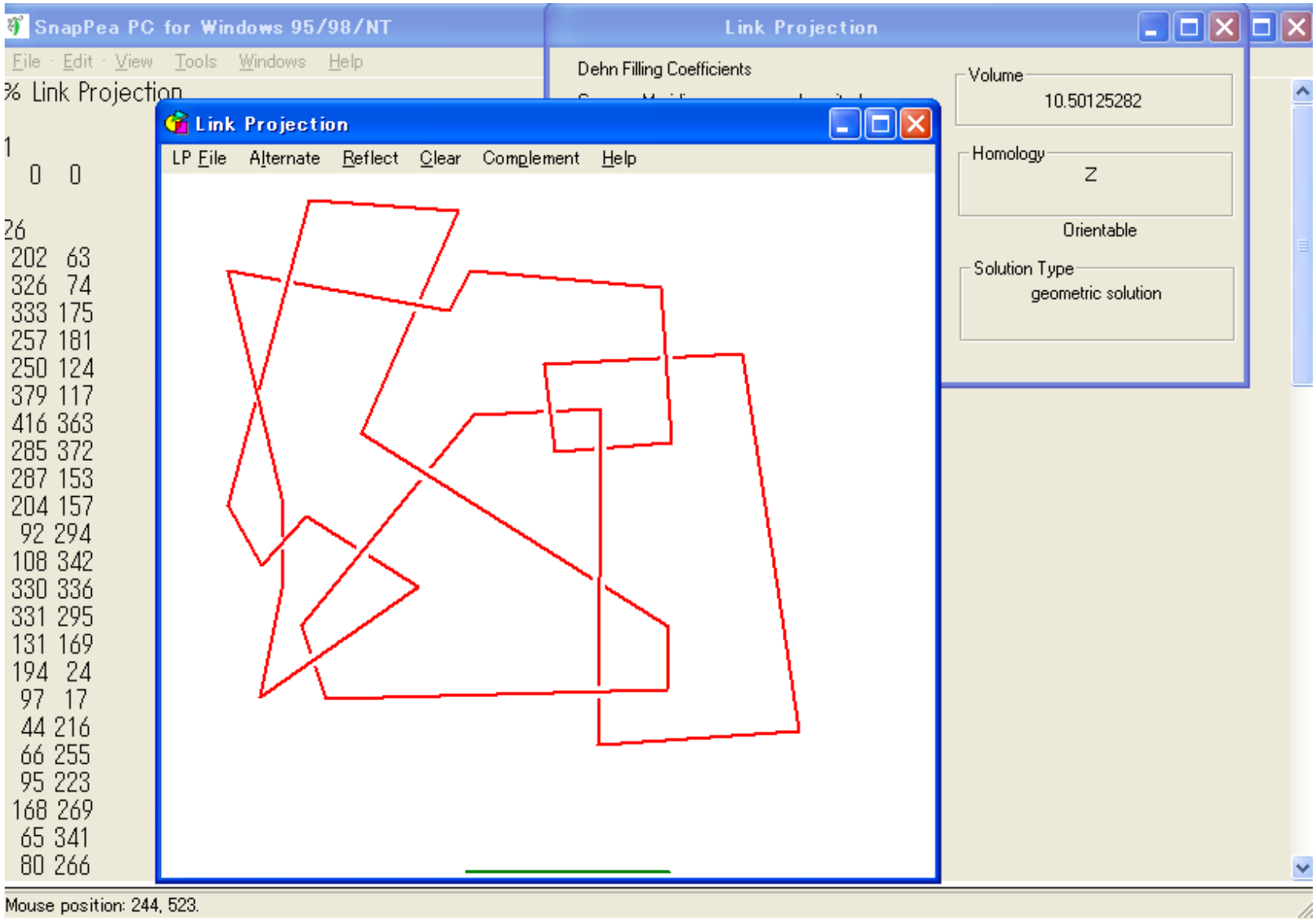
- ファイル: nongeometric12.prd

このファイルには3個の結び目のデータがあり、最初のデータのみnon-geometric solutionで他の2つはgeometric solutionを持つ。これはメニュ“Solution type”で確認できる。メニュ“Isometry I”で最初のデータを内部ファイルに格納し、メニュ“Quit/Next”で次のデータを読み込み、メニュ“Isometry II”で最初のデータのIsometry groupと比較する.

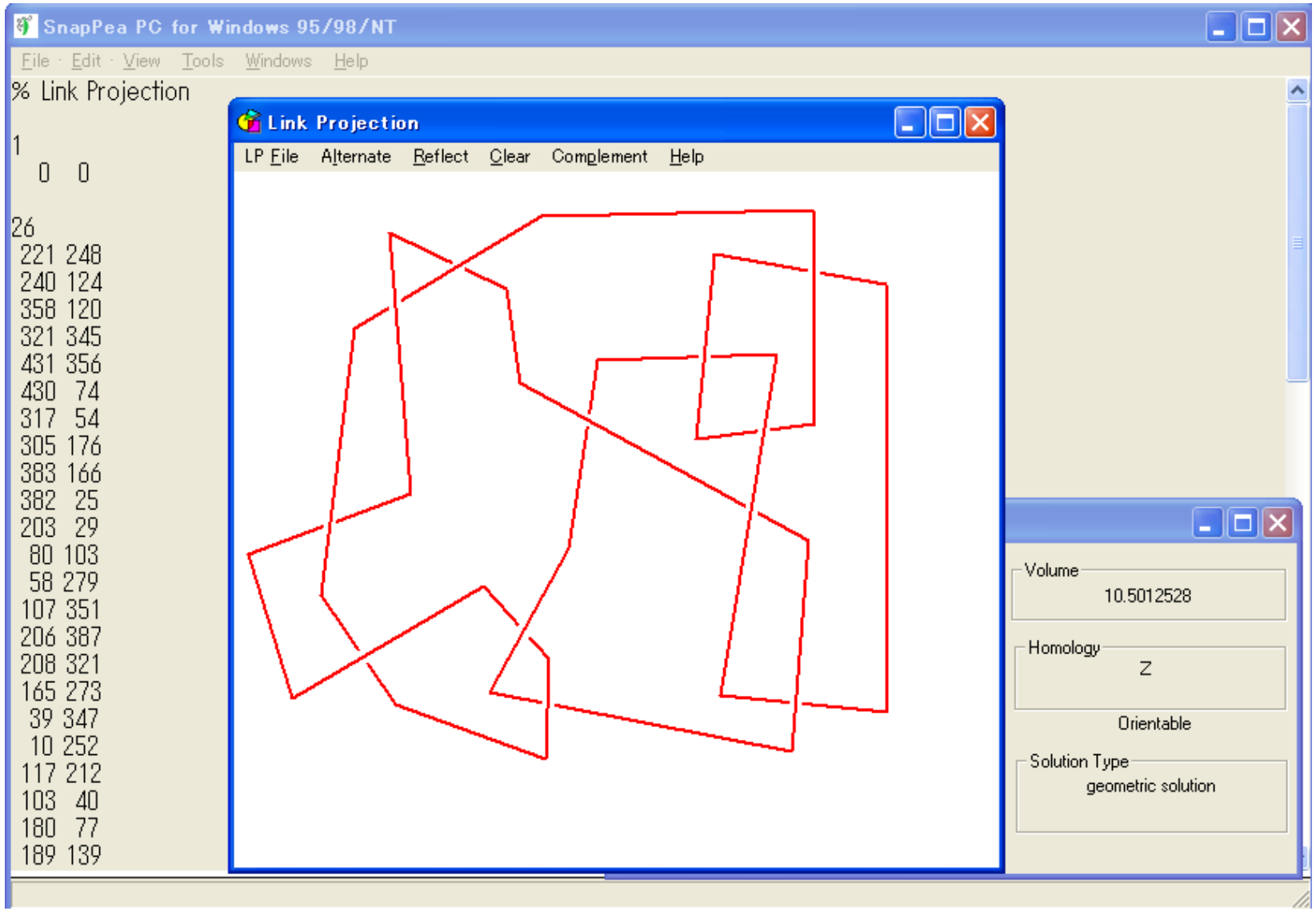
このファイルの最初と最後のデータは異なるが,同じ結び目のデータである。

これら2つのデータをWindows版のSnapPeaで順番を逆にして読み込むと、solutionに関して異なる結果が得られる？多分、SnapPea自身non-geometric solutionからgeometric solutionを得るためのランダムな操作を試みている？

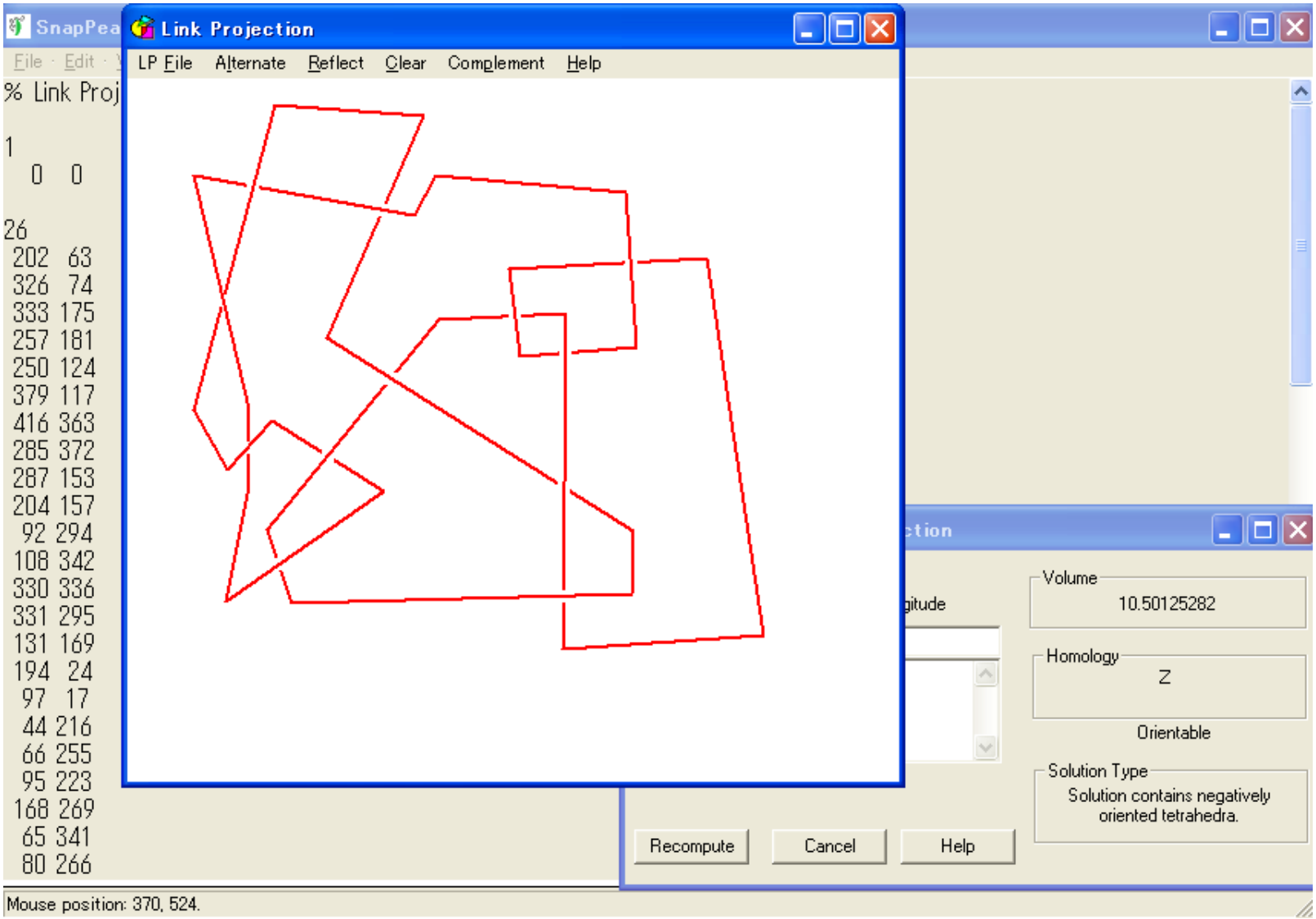
１２交点の結び目（１）


SnapPea PC for Windows 95/98/NT
File Edit View Tools Windows Help
% Link Projection
1
0 0
26
202 63
326 74
333 175
257 181
250 124
379 117
416 363
285 372
287 153
204 157
92 294
108 342
330 336
331 295
131 169
194 24
97 17
44 216
66 255
95 223
168 269
65 341
80 266
Link Projection
LP File Alternate Reflect Clear Complement Help
Link Projection
Dehn Filling Coefficients
Volume
10.50125282
Homology
Z
Orientable
Solution Type
geometric solution
Mouse position: 244, 523.

１２交点の結び目（２）


SnapPea PC for Windows 95/98/NT
File · Edit · View Tools Windows Help
% Link Projection
1
0 0
26
221 248
240 124
358 120
321 345
431 356
430 74
317 54
305 176
383 166
382 25
203 29
80 103
58 279
107 351
206 387
208 321
165 273
39 347
10 252
117 212
103 40
180 77
189 139
Link Projection
LP File Alternate Reflect Clear Complement Help
Volume
10.5012528
Homology
Z
Orientable
Solution Type
geometric solution

１２交点の結び目（３）


Link Projection
LP File
Alternate
Reflect
Clear
Complement
Help
% Link Proj
1
0 0
26
202 63
326 74
333 175
257 181
250 124
379 117
416 363
285 372
287 153
204 157
92 294
108 342
330 336
331 295
131 169
194 24
97 17
44 216
66 255
95 223
168 269
65 341
80 266
Volume
10.50125282
Homology
Z
Orientable
Solution Type
Solution contains negatively oriented tetrahedra.
Recompute
Cancel
Help
Mouse position: 370, 524.

nongeometric12.prd

24 1
24
4 8 13 2 19 5 9 23 21 11 17 15
24 1
24
4 10 15 -13 2 17 -21 23 11 5 -7 19
24 1
24
16 7 10 15 3 20 4 9 24 22 12 18

差出人: jrw@geometrygames.org
件名: non-geometric solutions
日時: 2008年3月5日 5:59:56:JST

- **Dear 落合豊行さん,**

- **>I will talk about SKB (SnapPea on K2K) in a parallel seminar of Annual**
- **>meeting of Mathematical society of Japan at next March 22.**
- **>This talk is the final one in my life.**

- **Ah, a bittersweet moment.**
- **Happy because it will be an interesting talk, and**
- **also happy because your retirement will soon begin**
- **and you will have time for your other hobbies and actitivies**
- **...but nevertheless a little melancholy too,**
- **that the talk will be your last?**

- **>I find a very simple method to avoid a problem of non-geometric**
- **>solutions of SnapPea. This method permit us "completely"**
- **>to get geometric solutions from all nonalternating**
- **>hyperbolic knots of up to 17 crossings. And I am trying**
- **>to test for all nonalternating knots of up to 18 and 19 crossings.**

- **Congratulations!**

- **Best wishes for the remainder of your year -- and for your**
- **new adventures beyond that.**

- **I assume your e-mail address will remain the same?**

- **Jeff**

謝辞

- この集会に招待してくれた濱田龍義氏に感謝します.
- 昨年の秋田セミナー後、毎週のように研究室を訪問し、改良に適切な助言をしてくれた市原一裕氏に感謝します.
- K2K、bTdの開発に協力してくれた村上順、金信泰造氏に感謝します.
- 今回の講演で,引退します, この分野(Computer Topology)の若い研究者の活躍に期待します. このソフトウェアの改良, 発展, 配布はすべてWeeks, 加古氏に委託します.
- ２００８年３月３１日http://www.ochiailab.ics.nara-wu.ac.jp/ochiai/freesoft.htmlに、MacOSX Leopard版をアップロードしました。

さようなら