# 結び目の近傍抜き出しソフトウェア

**Software constructing neighborhood of Knots**

**谷　研究室　　伊藤 雄祐**
**Yusuke Ito**

**概要**

結び目の近傍を抜き出すソフトウェアを作成した.

## 1　はじめに

自明性判定は結び目理論の基本的問題である. 自明性判定とは自明な結び目との同型判定である. 結び目の自明性判定において結び目の近傍を抜き出した空間を得ることは非常に有用である. そこで入力した結び目の近傍を抜き出し描画するソフトウェアを作成した.

### 1.1　グラフィカルな操作

結び目理論は数学の一分野である. 一般に数学者はプログラマであるとは限らない. そこでソフトウェアの入力を視覚的にすることで扱いを容易にした. コンピュータを扱う機会の少ないものに取っても簡便に操作をすることが可能である.

### 1.2　可視化

数学者は実際に結び目を描いて同型性判定を考える. 今まで結び目の近傍を見ることができるソフトウェアでプログラマ以外に対し操作が容易なものはなかった. 故にソフトウェアの出力を映像で出力することとしたが, 非常に有用である.

## 2　球面

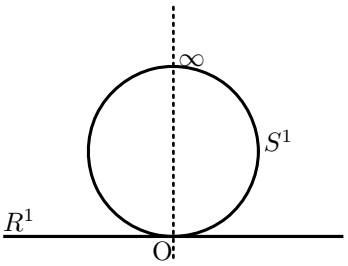
### 2.1　定義

$n$ 次元球面は以下で定義される.

$$\begin{aligned} S^n = (x_0, x_1, ..., x_n) \qquad & s.t. \\ x_0, x_1, ..., x_n \quad & \in \quad \mathbb{R} \\ x_0^2 + x_1^2 + \cdots + x_n^2 \quad & = \quad 1 \\ S^n \text{は } n \text{ 次元球面} & \end{aligned}$$

#### 2.1.1　球面とユークリッド 空間の関係

$\infty \in S^n$ は無限遠の 1 点
$R^n \bigcup \{\infty\} = S^n$




## 3　結び目

### 3.1　定義

$$K : S^1 \to S^3$$
$$K \text{ は } Knot$$

### 3.2　折線

線分は二つの点のペアである. $p_x, p_y$ を端点とする線分を $|p_x p_y|$ または $\overline{p_x p_y}$ などと表す.
折線は連続した線分の列である.

$$s_1 = |p_0, p_1|, s_2 = |p_1, p_2|, ..., s_n = |p_{n-1}, p_n|$$

で $R = \{s_0 s_1 ... s_{n-1}\}$ は折線.
頂点のリストを用いて $K(p_0 p_1 ... pn)$ とも表す.

### 3.3　単純閉折線

閉じた折線を閉折線とよぶ.　$K(p_0 p_1 ... p_n)$ のとき $p_0 = p_n$ となる折線が閉折線である. 閉折線は多角形である.
連続する線分の共有の端点以外に共有点を持たない折線を単純折線という. 折線上で隣り合っていない線分同士がどこかで接していない折線が単純折線である.
閉折線でありかつ単純折線なものは単純閉折線である.

### 3.4　結び目の表現

有限個の線分からなる単純閉折線として結び目を表現する.

# 4 結び目理論の基本的問題

## 4.1 同型の定義

$$\begin{aligned} K, K' &: Knot \\ \forall x &\in K \\ \exists h(x) &\in K' \\ s.t. &\quad h : S^3 \to S^3 \\ \Longrightarrow &\quad K, K'\text{は同型} \end{aligned}$$

## 4.2 同型判定

同型判定は結び目理論の基本的問題のひとつである.

## 4.3 同型の直観的イメージ

結び目は伸縮自在なゴム紐のようなものとして想像される. 伸ばす. 縮める. ひねる. 絡ませる. などの変形によって作ることのできる結び目は全て同型な結び目と思うことができる.

### 4.3.1 同型判定のクラス

$NP$ 困難よりも難しいとして知られている.[2]

## 4.4 自明な結び目

$$\begin{aligned} K &: Knot \\ K &\subset S^2 \\ \Longrightarrow &\quad K\text{は自明} \end{aligned}$$

交差や絡みを持たないことが自明のイメージである.

## 4.5 自明性判定

自明性判定は結び目理論の基本的な問題のひとつである. 自明性判定は自明な結び目との同型判定と考えることができる.

## 4.6 自明性判定のクラス

$NP$ に属するとして知られている.[2]

# 5 複体

## 5.1 単体

$r+1$ 個の位置ベクトル $\forall a_0, a_1, ..., a_r \in R^n$ を各々$S^n$ の $r-1$ 次元以下の部分空間に含まれないとする.

$$\{\sum_{i=0}^{r} \lambda_i a_i | \lambda_i \in \mathbb{R}, \sum_{i=0}^{r} \lambda_i = 1, \lambda_0, \lambda_1, ..., \lambda_r \geq 0\}$$

を $n$ 次元単体と呼ぶ.

### 5.1.1 $n$ 次元単体の例

代表的な $n$ 次元単体は次の図形である.
0 次元単体 : 点
1 次元単体 : 線分
2 次元単体 : 三角形
3 次元単体 : 四面体
4 次元単体 : 五胞体

## 5.2 定義

以下を満たすとき $K$ は $n$ 次元複体である.
1) 有限個の $n$ 次元単体からなる
2) $\alpha \in K, \delta \in \alpha$
s.t.
$\alpha$ : $n$ 次元単体
$\delta$ : $n$ 次元未満の単体
$\Longrightarrow \delta \in K$
3)$\forall \alpha, \beta \in K$
s.t.
$\alpha, \beta$ : $n$ 次元単体
$\delta = \alpha \bigcap \beta$
$\Longrightarrow \delta$ : $n$ 次元未満の単体

### 5.2.1 $n$ 次元複体のイメージ

$n$ 次元複体は $n$ 次元単体を密に並べたもののように見える.

# 6 近傍

## 6.1 空間の表現

このソフトウェアでは 3 次元複体として空間を表現する.
空間に含まれる線分の集合として結び目を描く.

## 6.2 定義

近傍は結び目に隣接した 3 次元単体の集合である.

# 7 重心細分

## 7.1 分割

次を満たす $\{\Delta_0, \Delta_1, ..., \Delta_n\}$ は $I$ の分割である.
1) $\Delta_i \in I$
$s.t. \forall i \in \{0, 1, ..., n\}$
2) $\Delta_i \bigcap \Delta_j = \phi$
$s.t. i \neq j, \forall i, j \in \{0, 1, ..., n\}$
3) $\Delta_0 \bigcup \Delta_1 \bigcup \cdots \bigcup \Delta_n = I$

### 7.2 重心

$r$ 次元単体の重心は $\frac{\sum_{i=0}^{r} a_i}{r}$ 記号 $g$ で表す.

### 7.3 三角形への重心細分

#### 7.3.1 記号の定義

位置ベクトル a, b, c により定義した 2 次元単体を $\Delta abc$ と表す.
その重心は $\frac{a+b+c}{3}$ である.

#### 7.3.2 定義

$\Delta a\frac{a+b}{2}g$, $\Delta a\frac{c+a}{2}g$, $\Delta b\frac{b+c}{2}g$, $\Delta b\frac{b+a}{2}g$, $\Delta c\frac{c+a}{2}g$, $\Delta a\frac{b+c}{2}g$ は $\Delta abc$ の分割である.
それらを $\Delta abc$ の重心細分と呼ぶ.

### 7.4 四面体への重心細分

位置ベクトル $a, b, c, d$ により定義される 3 次元単体の重心は $\frac{a+b+c+d}{4}$ である.
$a, b, c, d$ により定義される 3 次元単体は $\Delta abc, \Delta abd, \Delta acd, \Delta bcd$ を含んでいる.
$\Delta abc, \Delta abd, \Delta acd, \Delta bcd$ を重心細分することで得られた各 2 次元複体の各頂点へのベクトルと 3 次元複体の重心へのベクトルにより新たな 3 次元複体を定義できる.
その集合は $a, b, c, d$ により定義される 3 次元単体の分割になっている.
その集合を $a, b, c, d$ により定義される 3 次元単体の重心細分と呼ぶ.

### 7.5 近傍と重心細分

Hass *et al.* は結び目を含む空間全体を 2 回重心細分してから近傍を考えている.[2]
ソフトウェアもそれに沿い近傍を定義.

## 8 ソフトウェアについて

### 8.1 入力

結び目をダイアグラムにより入力する.

### 8.2 出力

入力された結び目の近傍の映像をウィンドウ上に出力する.

### 8.3 開発言語

C++
ライブラリとして LEDA を使用.

## 参考文献

[1] John Hopcroft and Robert Tarjan, "Efficient Planarity Testing" Journal of the ACM Vol. 21 (October 1974)

[2] Joel Hass, Jeffrey C. Lagarias and Nicholas Pippenger, "The Computational Complexity of Knot and Link Probrems" Journal OF THE ACM Vol. 46 pp.185-211 (March 1999)