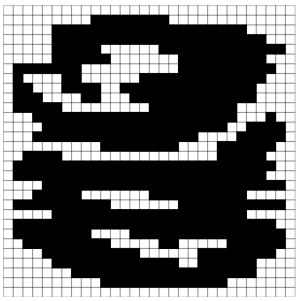
計算機による代数的位相幾何

大林

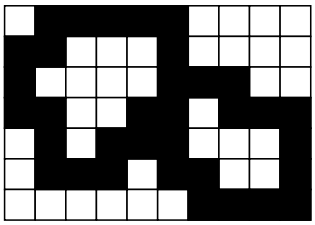
穴の数を数える

図 1(a) のようなビットマップ画像があったとしましょう。この黒い領域の個数、黒い領域に空いた穴の個数を計算機によって数える、という問題を考えます。この図では見ればわかるように黒い領域が 2 個、穴の個数は 3 個です。これをいかにして計算機に計算させればよいでしょうか。

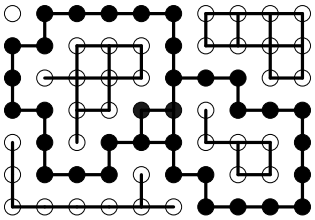
解法の 1 つとしてグラフの連結成分の問題に置き換える方法が考えられます。簡単のため図 1(b) のような画像に対してこの問題を考えることにしましょう。これに対し、白色の画素をグラフの白丸の頂点とし、白色の画像が隣接しているときにはその 2 つをつないだグラフを考えます (図 1(c))。黒色の画素に対しても同様のグラフを作ります。このグラフの連結成分を数えれば求める結果が得られます。白の連結成分の個数は外枠と繋がっている部分は捨てる必要があることに注意しましょう。



(a) 黒い部分が 2 つで穴が 3 つある画像



(b) 穴が 2 つある画像



(c) グラフ化したもの

図 1　2 次元画像の穴の数

3 次元の穴

さて、2 次元はできたので次は 3 次元の問題を考えてみましょう。3 次元の場合はピクセルデータではなくボクセルデータを使います。3 次元で考えるにあたっては、まず「3 次元空間での穴とは何か？」という問題を考える必要があります。

図 2(a) のような筒状の物体を考えてみましょう。確かにこれには穴が 1 つあいていると言えそうです。次に図 2(b) のような風船を考えてみましょう。するとこれは中に空気が入っているわけですから、何か穴のようなものが 1 つありそうだ、と言えます。穴というよりも空洞というべきかもしれません。

この 2 つの「穴」「空洞」はちょっと考えると別物であることがわかります。「穴」のほうは外側から内側を見ることができます。図 2(c) のような曲がったパイプだと内側を見るのは難しいですが、胃カメラとか内視鏡のようなものを持ってくれば内側も見えます。一方「空洞」のほうはそういうことができません。風船に小さな穴でも開けないと内側が見えません。また別の見方として「穴」は紐を通すことができますが、「空洞」は紐を通すことができない、という違いがあります。

図 2(d) のような浮き輪を考えるともっと複雑になります。まず真ん中に 1 つ「穴」があいています。そして浮き輪の空気の入っているところに「空洞」があります。ただこの空洞はさきほどの空洞と少し違い紐をぐるっと通すことができます (図 2(d) の破線)。つまり浮き輪の空気の入っている部分は「穴」と「空洞」の複合体ではないか、と思えてくるわけです。

ここまでで、3 次元で穴の数を数えるにあたっては次のような課題があることがわかりました。

- 考えている穴なるものは「穴」と「空洞」の 2 種類あって区別する必要がありそう
- 浮き輪の例のように「穴」であり「空洞」であるようなものがある

つまり穴や空洞といった概念をちゃんと定義する必要があります。

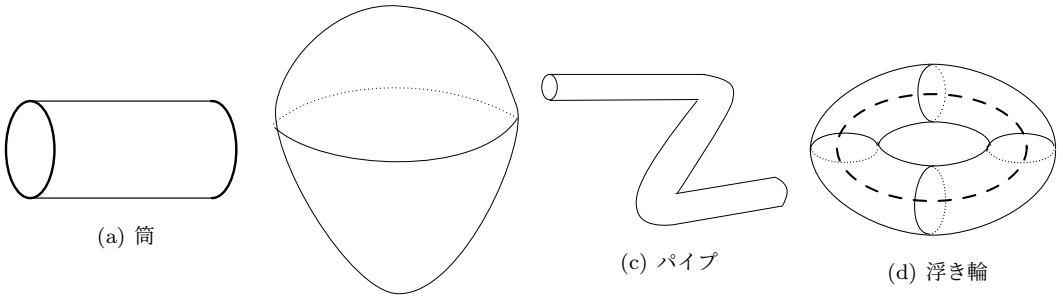


図 2　3 次元の物体の穴

ホモロジー

ここで数学の出番です。幾何学の一分野ではこのような「穴」や「空洞」はホモロジー論と呼ばれる分野で取り扱われます。詳しい理論を知りたい人は参考文献を見て勉強してもらうことにして、簡単に説明しましょう。ホモロジー論においてここでいう「穴」や「空洞」はベクトル空間のよう

なもので[*1]表されます。そしてそのベクトル空間の次元が穴の個数です。このベクトル空間は

$$H_0(M) : \text{図形の連結成分}$$
$$H_1(M) : \text{穴}$$
$$H_2(M) : \text{空洞}$$

と書くことにします。この M は対象となっている物体のことです。そしてこの次元がそれぞれの個数を表わしています。例えば

$$\begin{array}{lll} \dim(H_0(\text{図 1(a)})) = 2, & \dim(H_1(\text{図 1(a)})) = 3, & \dim(H_2(\text{図 1(a)})) = 0 \\ \dim(H_0(\text{筒})) = 1, & \dim(H_1(\text{筒})) = 1, & \dim(H_2(\text{筒})) = 0 \\ \dim(H_0(\text{風船})) = 1, & \dim(H_1(\text{風船})) = 0, & \dim(H_2(\text{風船})) = 1 \\ \dim(H_0(\text{浮き輪})) = 1, & \dim(H_1(\text{浮き輪})) = 2, & \dim(H_2(\text{浮き輪})) = 1 \end{array}$$

となります。平面に空洞はないので $\dim(H_2(\text{図 1}(a))) = 0$ と定義されます。浮き輪では、「穴」の数に相当するものが 2 に、「空洞」の数に相当するものが 1 になっています。内側の空洞が「穴」と「空洞」の両方の性質を持っているのではないか、というさきほどの話に対応しています[*2]。もっと大きな k に対しても H_k が定義されますが、この場合はすべて 0 になります。4 次元の図形などを考えると、「『穴』でも『空洞』でもないけどそれに似た何か」が表れてくるため、それを表すためにより大きな k に対しても H_k が定義されています。

計算機によるホモロジーの計算

ある一定の条件の下、ホモロジーは計算機によって計算できます[*3]。これは計算ホモロジーと呼ばれています。計算のためのソフトウェアはいくつかあるのですが、ここでは Vidit Nanda が開発した Perseus [*4] を使ってみましょう。

まずこの URL のページの Source Code, Executables and Usage をクリックして Here のところからソースの zip ファイルをダウンロードできます。Windows 用のバイナリなども同じところから取得できるので、そちらを使ってもよいでしょう。コンパイルする場合は、これを適当なディレクトリに展開して、

```
g++ -fpermissive -O2 Pers.cpp -o perseus
```

などとしてコンパイルできます。

[*1] $\mathbb{Z}$ 係数加群、つまりアーベル群です。

[*2] もちろんそんなにいいかげんな話ではなく、ちゃんと数学的に根拠があってこの値になっています。

[*3] ホモロジーを勉強したことのある人に対する説明: 単体ホモロジーは単体複体で表現された図形に対して定義されます。そして単体複体は有限個の単体の組み合わせとして実現されます。各単体の境界作用素 (∂) が定義されれば各単体を基底とする自由アーベル群によってホモロジー群が定義できます。代数的には $\mathbb{Z}$ 係数の有限次元線形代数で、境界作用素 ($\mathbb{Z}$ 線形写像) の像やカーネルは計算機で計算できるので、ホモロジー群の次元が計算できるわけです。単体を立方体に取り替えて同じことができます (Cubical homology といいます)。

[*4] `http://www.math.rutgers.edu/~vidit/perseus/index.html`

とりあえず例として 図 1(a) の 30x30 の画像を計算してみましょう。データファイル[*5]をダウンロードしてください。データは

```
2
30
30
-1 -1 -1 -1 -1 -1 -1 -1 -1 -1 ...
-1 -1 -1 -1 -1 -1 -1 -1 -1  1 ...
-1 -1 -1 -1 -1  1  1  1  1  1 ...
-1 -1 -1 -1 -1  1  1  1  1  1 ...
-1 -1 -1 -1 -1  1  1  1  1  1 ...
```

という形式で、最初の 3 行は図形の次元、X 座標方向のピクセル数、Y 座標方向のピクセル数を表し、それ以降は 1 でピクセルが存在することを、-1 で存在しないことを表現しています。

```
./perseus cubtop bitmap.txt
```

で計算します。いくつかファイルが出力されますが、ここでは output_betti.txt を見てください。

```
Frame [1]: 2 3
```

と出力されているはずです。この 2 と 3 が $\dim(H_0)$ と $\dim(H_1)$ です。連結成分が 2 個、穴が 3 つあることが計算できています。

次に 3 次元の筒[*6]、風船[*7]、浮き輪[*8] の 3 つのデータについて perseus を使いましょう。3 次元を 1 と-1 で埋めていくとファイルが大きすぎるので別のフォーマットを使います。cylinder.txt を見ると

```
3
18 44 10 1
18 44 11 1
18 44 12 1
18 44 13 1
18 44 14 1
    :
```

というデータになっています。最初の 3 は次元です。それ以降の行はボクセルの x,y,z 座標です[*9]。

[*5] http://www.kmc.gr.jp/~ohai/kmcpress2013winter/bitmap.txt

[*6] http://www.kmc.gr.jp/~ohai/kmcpress2013winter/cylinder.txt

[*7] http://www.kmc.gr.jp/~ohai/kmcpress2013winter/balloon.txt

[*8] http://www.kmc.gr.jp/~ohai/kmcpress2013winter/torus.txt

[*9] これらのデータは gnuplot で `splot "cylinder.txt"` などとすることで 3 次元に表示できます。本当にこのデータが筒や浮き輪を表しているのか気になる方はこのようにして確認すればよいでしょう。

最後の 1 はここではとりあえずすべて 1 にしておけば OK です。このファイルに対し

```
./perseus scubtop cylinder.txt
```

として、output_betti.txt を見ると

```
Frame [1]: 1 1
```

と出力されます。balloon.txt, torus.txt ではそれぞれ

```
Frame [1]: 1 0 1
```

と

```
Frame [1]: 1 2 1
```

と出力されます。上で説明した $\dim(H_0), \dim(H_1), \dim(H_2)$ に一致しています。このようにして、図形の穴や空洞の個数を計算機で計算できます。

最後に

3 次元の穴の個数を数えるのが何の役に立つのか、というのを最後にちょっと考えましょう。軽石のような多孔質の材料やスポンジの穴を数えるというのはどうでしょう。穴や空洞の数や大きさはこれらの材料の機能に重要な役割を果たしていると考えられます。実は穴や空洞の数を数えるだけでなく、その大きさを「なんとなく」測ることもできるので[*10]、これを使って新材料の性能評価ができるかもしれません。現在は何に使えるかを模索している段階です。他にも金属ガラスや蛋白質の構造解析のような材料科学、画像処理、機械学習などへの応用が試みられています。

参考文献

平岡裕章. タンパク質構造とトポロジー:パーシステントホモロジー群入門. 共立出版, 東京, 2013.

[*10] persistent homology というアイデアを用います。Perseus の 3 次元入力での各行の最後の 1 や、出力で無視したファイルは実はこれに関連しています。