CHomP ソフトウェア入門

平岡裕章（広島大学）
Paweł Pilarczyk（京都大学）

1　はじめに

ホモロジー群計算ソフトウェア CHomP はホモロジー群の計算機を用いた高速計算，及びその応用を目指したプロジェクト Computational Homology Project [3] によって開発された．現在サポートしている OS は Windows，Mac，Unix，Linux であり，ウェブページ [3] からダウンロード可能である．ここで主に扱う幾何的対象は方体集合と呼ばれる区間の直積で構成されるものであり，そのデータ構造の特徴（三角形分割を必要としない）から様々な分野への応用が展開されている．

そこで本稿では CHomP の基本的な扱い方について解説を試みる．特に，頻繁に使われるいくつかのコマンドの入出力関連についての説明を中心に進めていく．インストール方法についてはウェブページ [3] 内にプログラム開発者による詳しい解説があるのでここでは省略する．ちなみにこのウェブページにはここでは取り扱わないコマンドの使用法，豊富な例題，様々な応用例等の有益な情報が公開されているので一読をお勧めする．

まず始めに 2 節で単体複体のホモロジー群，3 節で方体集合のホモロジー群についての CHomP による扱い方を説明する．CHomP では方体集合間のあるクラスの連続写像に対してその誘導準同型写像を計算することが可能であり，それに関する解説を 4 節にまとめた．最後の 5 節では C++インターフェースについて基本的な使用方法を紹介している．なお方体集合のホモロジー群については文献 [1] に詳しく解説されているので参照されたい．

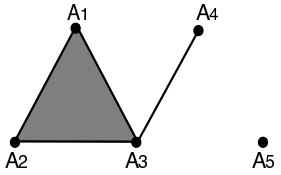
2　単体複体のホモロジー群計算

この節では単体複体のホモロジー群計算に関連した CHomP の使用方法について解説していく．図 1 に示された 2 つの多面体を例に挙げて考えてみよう．それぞれの単体複体は

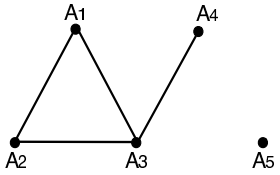
$$K_1 = \{|A_1A_2A_3|, |A_1A_2|, |A_2A_3|, |A_1A_3|, |A_3A_4|, |A_1|, |A_2|, |A_3|, |A_4|, |A_5|\},$$
$$K_2 = \{|A_1A_2|, |A_2A_3|, |A_1A_3|, |A_3A_4|, |A_1|, |A_2|, |A_3|, |A_4|, |A_5|\}$$

で与えられる．ここで単体複体 $K = \{s_i \mid i = 1, \cdots, N\}$ が与えられたとき対応する多面体を $|K| = \cup_{i=1}^{N} s_i$ で表すことにしよう．

CHomP はこのような単体複体のホモロジー群計算コマンドとして `homsimpl` を提供している．入力ファイルは単体複体の情報であり，図 1(a)，(b) の場合では各頂点 A_i の添字 i を用いてそれぞれ次のテキストファイルで単体複体が定義される．



(a) $|K_1|$



(b) $|K_2|$

図 1: 多面体

```
; (a) ファイル：sc1.txt
{1,2,3}
{3,4}
{5}
```

```
; (b) ファイル：sc2.txt
{1,2}
{2,3}
{1,3}
{3,4}
{5}
```

コマンド `homsimpl` は入力ファイルを読み込む際，単体複体を構成する為に必要となる面（図 1(a) であれば $|A_1A_2|$, $|A_2A_3|$, $|A_1A_3|$, $|A_1|$, $|A_2|$, $|A_3|$, $|A_4|$）を自動的に補間する機能を持っている．よって入力ファイル内に全ての単体を記述する必要はない．ちなみに CHomP が取り扱う入力ファイルではセミコロン; の後には使用者のコメントが記入できるようになっている．

このように用意された入力ファイル（`filename` とする）に対して単体複体のホモロジー群はターミナル上で

```
homsimpl filename
```

と入力すればよい．図 2 はターミナル上に出力された結果であり，ホモロジー群は画面下

```
HOMSIMPL, ver. 0.01, 11/09/04. Copyright (C) 2003-2004 by Pawel Pilarczyk.
This is free software. No warranty. Consult 'license.txt' for details.
[Tech info: simpl 4, chain 12, addr 4, intgr 2.]
Reading simplices to X from 'sc1.txt'... 3 simplices read.
Collapsing faces in X... .. 8 removed, 2 left.
Note: The dimension of X decreased from 2 to 0.
Creating the chain complex of X... Done.
Time used so far: 0.00 sec (0.000 min).
Computing the homology of X over the ring of integers...
H_0 = Z^2
Total time used: 0.00 sec (0.000 min).
Thank you for using this software. We appreciate your business.
```

(a) `homsimpl sc1.txt`

```
HOMSIMPL, ver. 0.01, 11/09/04. Copyright (C) 2003-2004 by Pawel Pilarczyk.
This is free software. No warranty. Consult 'license.txt' for details.
[Tech info: simpl 4, chain 12, addr 4, intgr 2.]
Reading simplices to X from 'sc2.txt'... 5 simplices read.
Collapsing faces in X... . 2 removed, 7 left.
Creating the chain complex of X... . Done.
Time used so far: 0.00 sec (0.000 min).
Computing the homology of X over the ring of integers...
Reducing D_1: 0 + 2 reductions made.
H_0 = Z^2
H_1 = Z
Total time used: 0.00 sec (0.000 min).
Thank you for using this software. We appreciate your business.
```

(b) `homsimpl sc2.txt`

図 2: ターミナル上の出力画面

部に

$$\text{(a) } H_0(K_1) = \mathbb{Z}^2, \qquad \text{(b) } H_0(K_2) = \mathbb{Z}^2,\ H_1(K_2) = \mathbb{Z}$$

で与えられていることがわかる．実際 (a)，(b) ともに連結成分は 2 つあり，(b) では 1 つのループが頂点 A_1, A_2, A_3 により構成されている．なおその他の出力内容としてはバージョ

ン情報，計算過程，計算時間が示される．

さて次にコマンド homsimpl が持っているオプション機能について解説する．まず CHomP で用意されているコマンドはターミナル上でそのコマンド名のみを入力すると，簡単な解説や使用可能なオプション群が表示される．homsimpl についても様々なオプションが用意されているが，ここでは応用上重要になってくる生成元の保存について紹介しよう．入力される単体複体（filename1）に対して，そのホモロジー群の生成元を出力ファイル（filename2）に保存するには homsimpl に-g オプションをつけて

```
homsimpl filename1 -g filename2
```

と入力すればよい．なお生成元の選択には任意性が含まれることに注意しておく．よってここでの出力は homsimpl が内蔵している固有のアルゴリズムに従って適当に選ばれたものであり，生成元の代表元として特に意味を持つものではない．

homsimpl の使い方として最後に相対ホモロジー群について述べておく．単体複体 K_1 に対してその部分複体を $K_2 \subset K_1$ としたとき相対ホモロジー群 $H_*(K_1, K_2)$ の計算コマンドは再び homsimpl であり，K_1 を記述したファイルを filename1，K_2 を記述したファイルを filename2 として

```
homsimpl filename1 filename2
```

と入力すればよい．

3　方体集合のホモロジー群計算

この節では Computational Homology Project の中で最もその本領を発揮する方体集合のホモロジー群を取り扱う．ここで方体集合 $X \subset \mathbb{R}^n$ とは基本方体と呼ばれる閉区間の直積からなる集合

$$
\begin{aligned}
&Q = I_1 \times I_2 \times \cdots \times I_n \\
&I_i = [l_i, l_i + 1] \text{ もしくは } [l_i, l_i],\ l_i \in \mathbb{Z}
\end{aligned}
$$

の有限和

$$
X = \bigcup_{k=1}^{m} Q_k, \quad Q_k : \text{基本方体}
$$

として定義される．計算機内でのデータ構造を考慮すると，このように区間の直積を構成単位とすることでその取り扱いは格段に容易になる．また実際の観測・実験データ等は有理数の直積で構成される為，適当なスケーリングを施すことで上記の設定に持ち込むことが可能であり，その汎用性は高い．また得られた数値データの誤差（観測誤差や数値誤差）があらかじめわかっている場合には，その誤差分を区間として丸め込むことで，誤差を考慮した方体集合として数値データを幾何学的に表現することもできる．

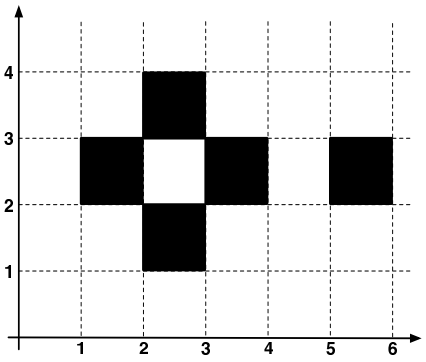
まずはじめに方体集合を記述する入力テキストファイルの書式について説明する．全ての基本方体の次元が扱っている空間の次元 n と等しい場合と，幾つかの基本方体が縮退している場合に応じて 2 通りの入力書式が用意されている．例を用いてそれぞれの書式を見

ていこう.

(I) 全ての基本方体の次元が n の場合
　この場合各基本方体は

$$[l_1, l_1+1] \times \cdots \times [l_n, l_n+1]$$

と表されている為，端点 $(l_1, \cdots, l_n)$ を定めれば基本方体が一意に定まる．よって方体集合を構成するそれぞれの基本方体に対して，この端点の情報を入力ファイルに記述すればよい．図 3 に示した方体集合の場合はその右に示してあるファイル `fullcubes.txt` が入力ファイルとして用いられる．ちなみに基本方体は閉区間の直積で表されているので，例




```
; ファイル：fullcubes.txt
(1,2)
(2,1)
(2,3)
(3,2)
(5,2)
```

図 3: 方体集合の例（右は入力ファイル）

えば基本方体 `(1,2)` と `(2,1)` は図 3 では交わっているものとして扱われる.

(II) 幾つかの基本方体が縮退している場合
　この場合は縮退している基本方体があるため，それぞれの基本方体を各座標成分ごとに記述する必要がある．例えば k 番目の区間が縮退している基本方体

$$[l_1, l_1+1] \times \cdots \times [l_{k-1}, l_{k-1}+1] \times [l_k, l_k] \times [l_{k+1}, l_{k+1}+1] \times \cdots \times [l_n, l_n+1]$$

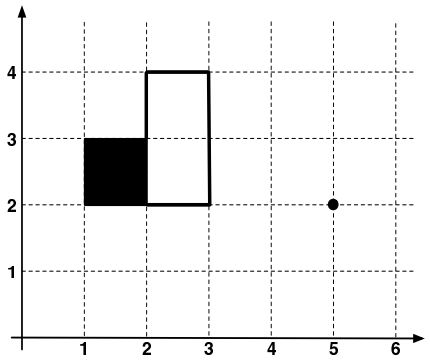
を考えた場合，そのまま各座標ごとにそれぞれの区間の情報を記載する書式

$$[\mathtt{l}_1, \mathtt{l}_1+1]\ \mathtt{x}\ \cdots\ \mathtt{x}\ [\mathtt{l}_{k-1}, \mathtt{l}_{k-1}+1]\ \mathtt{x}\ [\mathtt{l}_k, \mathtt{l}_k]\ \mathtt{x}\ [\mathtt{l}_{k+1}, \mathtt{l}_{k+1}+1]\ \mathtt{x}\ \cdots\ \mathtt{x}\ [\mathtt{l}_n, \mathtt{l}_n+1]$$

と，次のように各座標成分の最小点と最大点の座標をスペースで区切って対として表示する方法

$$[(\mathtt{l}_1, \cdots, \mathtt{l}_{k-1}, \mathtt{l}_k, \mathtt{l}_{k+1}, \cdots, \mathtt{l}_n)\ (\mathtt{l}_1+1, \cdots, \mathtt{l}_{k-1}+1, \mathtt{l}_k, \mathtt{l}_{k+1}+1, \cdots, \mathtt{l}_n+1)]$$

の 2 通りがある．ここでアルファベット `x` は直積記号 $\times$ を表すために用いられる．縮退している区間が 2 つ以上ある場合もそれぞれ同様に拡張される．例えば図 4 からなる方体集合の場合は，その右に示してある `reducubes.txt` が入力ファイルとして用いられる.
　このようにして構成される入力ファイルに対して，CHomP の基本ソフトウェアはベッチ数を計算するコマンド `chomp` を用意している．例えば図 3 と図 4 の例では方体集合を記述したファイル（`filename`）に対して




```
; ファイル：reducubes.txt
[1,2] x [2,3]                [(1,2) (2,3)]
[2] x [3,4]                  [(2,3) (2,4)]
[2,3] x [2]                  [(2,2) (3,2)]
[2,3] x [4]      もしくは    [(2,4) (3,4)]
[3] x [2,3]                  [(3,2) (3,3)]
[3] x [3,4]                  [(3,3) (3,4)]
[5] x [2]                    [(5,2) (5,2)]
```

図 4: 方体集合の例（右は入力ファイル）

```
chomp filename
```

を入力すれば，出力としてベッチ数”2, 1, 0”が得られる．なおこのコマンドにはベッチ数の計算に使われるアルゴリズムを指定するオプションがある．最もよく使われるアルゴリズムは MM_CR と PP であり，その指定の方法は--engine オプションの後にアルゴリズム名を次のように指定すればよい．

```
chomp filename --engine MM_CR
```

一方 CHomP 拡張ソフトウェアではコマンド homcubes が用意されており，方体集合のホモロジー群が

```
homcubes filename
```

により出力される．また生成元を保存するには homsimpl の場合と同様に

```
homcubes filename1 -g filename2
```

とすることで，filename2 にホモロジー群の適当な生成元が各次元ごとに出力される．相対ホモロジー群についても同様で，書式自体は homsimpl の場合と同様に方体集合の対を

```
homcubes A.txt B.txt
```

のように入力すればよい．しかしその出力はホモロジー群 $H_*(A \cup B, B)$ を計算している点に注意しないといけない．

以上はあらかじめ用意された方体集合を入力データとして取り扱ったが，実験や観測から得られる画像ファイルを直接入力として用いたい状況も当然考えられる．この要望に対して，CHomP では bitmap 形式の 2 次元画像データの方体集合表現を上記 (I) の形式で出力するコマンド bmp2pset を用意している．例えば入力 bitmap ファイル picture.bmp が白黒画像のとき，コマンド

```
bmp2pset picture.bmp picture.txt
```

により `picture.bmp` 画像の方体集合表現が `picture.txt` に出力される．この出力ファイルを `homcubes` の入力として使用することで，bitmap 画像のホモロジー群計算が容易に行われる．入力画像データがカラーデータのときは自動的にグレイスケール $(0 \sim 255)$ に変換されて処理が行われる．その場合はコマンド `bmp2pset` とオプション `-t` を組み合わせて閾値のグレイスケールレベルを指定し，対応する範囲の方体部分集合を出力させることが可能である．オプション `-t` で閾値を指定しない場合は画像データから自動的に閾値のグレイスケールレベルを判断して対応してくれる．

4 誘導準同型写像の計算

方体集合対 (X, A) と $(Y, B), (A \subset X, B \subset Y)$ の間の連続写像 $f\colon (X, A) \to (Y, B), f(A) \subset B$ が与えられたとき，その誘導準同型写像は CHomP を用いて計算可能である．そこで使用されているアルゴリズムの詳細については [1][2] を参照されたい．このアルゴリズムは方体集合に付随する様々な特徴を用いており，単体複体間の連続写像に対する誘導準同型写像には適用できない．単体複体の誘導準同型写像に関しては効率的なアルゴリズムが現在のところ存在しておらず，このことは方体集合を基礎としたホモロジー群計算の有用性を意味している．

まず方体写像についての定義を述べる．方体集合 X, Y, A, B が縮退していない基本方体の集まり $\mathcal{X}, \mathcal{Y}, \mathcal{A}, \mathcal{B}$ を用いてそれぞれ構成されているとする．このとき $\mathcal{X}$ から $\mathcal{Y}$ への方体写像 $\mathcal{F} : \mathcal{X} \multimap \mathcal{Y}$（$\multimap$ は方体写像を表す記号）とは $\mathcal{X}$ 内の各基本方体 Q_i に対して $\mathcal{Y}$ 内の基本方体の集まりを対応付ける多価写像として定義される．また連続写像 $f\colon X \to Y$ に対して，全ての基本方体 $Q \in \mathcal{X}$ について $f(Q) \subset |\mathcal{F}(Q)|$ が成り立つとき $\mathcal{F}$ を f の表現と呼ぶ．連続写像 f の誘導準同型写像 $f_* : H_*(X, A) \to H_*(Y, B)$ は，その表現 $\mathcal{F} : \mathcal{X} \multimap \mathcal{Y}$ が $\mathcal{F}(Q) \neq \emptyset$，$\mathcal{F}(\mathcal{A}) \subset \mathcal{B}$ かつ次の非輪状性の条件を満たすとき $\mathcal{F}$ を用いて計算可能となる [1][2].

定義 1 $\bigcap_{Q_i \in \mathcal{R}} Q_i \neq \emptyset$ となる基本方体の集まり $\mathcal{R} \subset \mathcal{X}$ に対して $|\mathcal{F}(\mathcal{R})|$ がつねに非輪状となるとき，方体写像 $\mathcal{F}\colon \mathcal{X} \multimap \mathcal{Y}$ は非輪状であると呼ぶ.

すぐわかるように対象としている方体写像がこの定義を満たしているかどうかを調べることは容易ではない．しかし各基本方体の像の実現 $|\mathcal{F}(Q)|$ が凸集合になっている場合には自動的にこの性質は満たされることが知られている．よって誘導準同型写像の計算にとっては各基本方体の像の実現が凸集合になるような写像は扱いやすいことになる．

CHomP では方体写像を定めるテキストファイルに対して 2 通りの書式を用意している．1 つ目は異なる行ごとに対応 $\mathcal{X} \ni Q \mapsto \{Q_1, \ldots, Q_n\} \subset \mathcal{Y}$ を記入する方法である．ここで基本方体の集まりは空白で区切って指定される．例えば `(1,2) -> {(4,1) (4,2) (5,1) (5,2) (6,1) (6,2)}` は図 5 に示されている $\mathcal{F}\big([1,2] \times [2,3]\big) = \big\{[4,5] \times [1,2],\ [4,5] \times [2,3],\ [5,6] \times [1,2],\ [5,6] \times [2,3],\ [6,7] \times [1,2],\ [6,7] \times [2,3]\big\}$ を意味する．ここで各基本方体は最も座標の値が小さい頂点で指定される．

方体写像を定義する 2 つ目の書式は各基本方体 Q の像の実現 $|\mathcal{F}(Q)|$ が凸集合になる場合に適用される．この場合最も座標が小さい頂点と大きい頂点を指定すれば凸集合が定まることに注意しよう．$\mathcal{X}$ の各基本方体も同様に表すと，この書式を用いた場合は図 5 の方

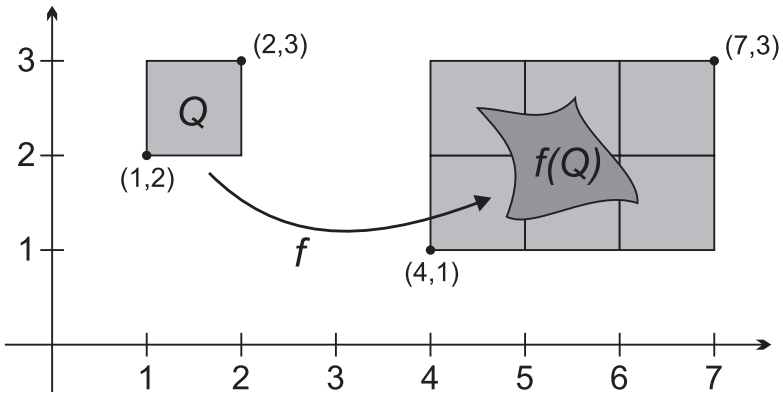


図 5: 基本方体 Q の f とその表現である方体写像 $\mathcal{F}$ による像

体写像は `[(1,2)(2,3)] [(4,1)(7,3)]` で定められる．像が多くの基本方体からなる凸集合で与えられる場合には，その入力が書式 1 に比べて非常に簡単になる．ただし書式 2 の入力ファイルにはその先頭に付加的な情報（空間次元，$\mathcal{X}$ の構成要素数，写像の種類）を指定する必要がある．

さてこのようにして用意される非輪状な方体写像 $\mathcal{F}$ から `homcubes` を用いて誘導準同型写像 f_* を計算することができる．空間対の写像 $\mathcal{F}\colon (\mathcal{X},\mathcal{A}) \multimap (\mathcal{Y},\mathcal{B})$ を扱っている場合は $\mathcal{F}$, $\mathcal{X}$, $\mathcal{A}$, $\mathcal{Y}$, $\mathcal{B}$ を表す 5 つのテキストファイルが必要となる．$\mathcal{A}=\mathcal{B}=\emptyset$ の場合にはそれらを除いた 3 つのファイルが用いられる．例えば $H_*(X,A), H_*(Y,B)$ の生成元も保存する場合にはコマンド

```
homcubes F.map X.cub A.cub Y.cub B.cub -g XA.gen -g YB.gen
```

で誘導準同型写像が計算される．ここで `XA.gen`, `YB.gen` にはホモロジー群 $H_*(X,A), H_*(Y,B)$ の生成元がそれぞれ保存されることになる．なお，CHomP では与えられた方体写像が非輪状性を満たしているか確認するコマンド `chkmvmap` を提供している．このコマンドは `homcubes` と同様に用いられ，方体写像 `F.map`，方体集合 `X.cub`, `A.cub`, `Y.cub`, `B.cub` で与えられるテキストファイルに対して次のように用いられる．

```
chkmvmap F.map X.cub A.cub Y.cub B.cub
```

5 C++インターフェース

CHomP ライブラリには C++言語で記述されたプログラム上でホモロジー群計算関連の関数へアクセスを可能にするインターフェースが整備されている．より高度な数値計算の為にはこのようなインターフェースは必須であり，あらかじめデータを保存しその後 `chomp` や `homcubes` といったコマンドで計算させるといった手間が大幅に省ける．ここでは幾つかの基本的なインターフェースについて解説を与える．なおここで紹介する内容の詳細については CHomP が提供しているサンプルプログラム等を参照されたい．

5.1 基本インターフェース

基本インターフェースではコマンド chomp で用いられていたプログラムへの C++言語からのアクセスを可能にしている．ここで基本方体は縮退していないものが扱われ，それらから構成される方体集合は bitmap 形式で用意する必要がある．ちなみに chomp でオプション指定をしていた engine はここでも選択可能になっている．

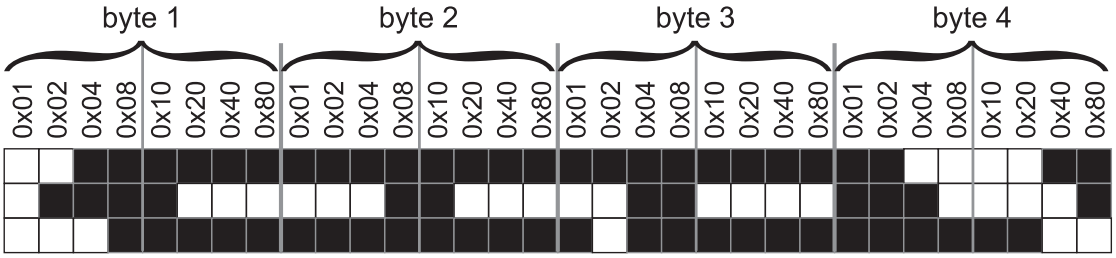


図 6: ビット列で表される bitmap 形式の例

さて，基本インターフェースでは方体集合を表すために bitmap 形式を採用しており，その n 次元 bitmap データは 1 次元配列として格納されることになる．ここでビット 1 は対応する場所に基本方体が存在しビット 0 は存在しないことを意味する．また取り扱う方体集合を内部に含む $\mathbb{R}^d$ 内の長方形領域 $[0, n_1] \times \cdots \times [0, n_d]$ をあらかじめ指定する必要があり，その為に整数 $n_1, \cdots, n_d$ を配列に格納することになる．その際技術的制約から n_1 の値は 32 ビット計算機では 32 の倍数，64 ビット計算機では 64 の倍数にする必要がある．各 1 バイトは 8 つの連続した基本方体に対応し，下位のビットほど座標の値が小さい基本方体を表す（図 6 を参照）．つまり bitmap データの最初のビットは基本方体 $(0, \cdots, 0)$ を表し，それに続く $n_1 - 1$ 個のビットが第 1 座標以外の全ての座標成分が 0 の基本方体を表すことになる．その後の n_1 個のビットは第 2 座標が 1 の基本方体に対応し，以下同様に各座標成分を大きくすることで d 次元方体集合が指定される．このように用意された方体集合に対してそのベッチ数を計算する関数は次で与えられる．

```
void ComputeBettiNumbers (const void *buffer, int *sizes, int dim,
int *result, const char *engine = 0, const int *wrapping = 0,
bool quiet = false);
```

- クラス CubicalSet

ComputeBettiNumbers を使うには bitmap 形式で方体集合を構成する必要があるが，その作成や操作を簡単に行う為に CHomP ではクラス CubicalSet が用意されている．CubicalSet のオブジェクトを作るには方体集合が定義される長方形領域

$$[n_1^-, n_1^+] \times \cdots \times [n_d^-, n_d^+]$$

を，d，$n_1^-, \ldots, n_d^-$，$n_1^+, \ldots, n_d^+$ で指定する必要がある．ここで用意された長方形領域に基本方体を加えるには関数 Add を，消去するには関数 Delete を用いればよい．これまでと同様に基本方体は座標の値が最小の頂点 $(k_1, \ldots, k_d)$ で表されるが，当然それらは不等

式 $n_i^- \leq k_i < n_i^+$ を満たさなければならない．このようにして作られる方体集合に対してそのベッチ数を計算させるには関数

```
void ComputeBettiNumbers (const CubicalSet &s,
int *result, const char *engine = 0, bool quiet = false);
```

を用いればよい（プログラム 1 参照）.

プログラム 1 クラス CubicalSet の使用例

```
01 #include <iostream>
02 #include "capd/homengin/cubiset.h"
03
04 int main ()
05 {
06   int left_coords [] = {-6, -5, 0};
07   int right_coords [] = {6, 1, 4};
08   CubicalSet Q (left_coords, right_coords, 3);
09
10   int cube1 [] = {1, -5, 0};
11   Q. Add (cube1);
12   int cube2 [] = {5, -2, 2};
13   Q. Add (cube2);
14
15   int betti [4];
16   ComputeBettiNumbers (Q, betti, "MM_CR", true);
17   for (int i = 0; i < 4; ++ i)
18     std::cout << (i ? " " : "") << betti [i];
19   std::cout << '\n';
20   return 0;
21 }
```

5.2 拡張インターフェース

拡張インターフェースでは一般の基本方体（辺が縮退していてもよい），方体集合，方体複体，方体写像などに対応するクラスを用意している．またこれらのクラスに対してホモロジー群を計算する関数も整っている．これらは名前空間 chomp::homology 内で使われることになる.

クラスのオブジェクトを指定するには頂点の座標を表す整数列を入力する必要があるが，標準では coordinate 型と呼ばれるものが用いられる（16 ビット整数 short int と同じ）．実用上は $[-32768, 32767]$ の整数で十分であると思われるが，それ以上の範囲を使用する際はソースコードを参照されたい.

- クラス Cube，SetOfCubes

Cube は $\mathbb{R}^d$ 内で縮退していない基本方体を表す為に使われるクラスである．このクラスのコンストラクタは基本方体を指定する座標と次元の情報を必要とする．一方 Cube で指定されたオブジェクトから関数 coord によって座標の列，関数 dim で基本方体の次元が取り出せるようになっている.

このような基本方体で構成される方体集合を記述するクラスとして SetOfCubes がある．このクラスに基本方体を加えるには add 関数，消去するには remove 関数がそれぞれ用いられる．また与えられた基本方体が SetOfCubes のオブジェクトに既に含まれているかど

うかを調べる関数 check も用意されている．この SetOfCubes で表される方体集合 X や方体集合対 (X, A) のホモロジー群を計算する関数としては Homology が用意されている．(プログラム 2 参照)

プログラム 2 クラス Cube，SetOfCubes の使用例

```
01 #include <iostream>
02 #include "chomp/homology/homology.h"
03
04 using namespace chomp::homology;
05
06 int main ()
07 {
08   coordinate coords1 [] = {1, -5, 0};
09   Cube Q (coords1, 3);
10   SetOfCubes S;
11   S. add (Q);
12   coordinate coords2 [] = {5, -2, 2};
13   S. add (Cube (coords2, 3));
14
15   Chain *hom = 0;
16   int maxLevel = Homology (S, "S", hom);
17   for (int q = 0; q <= maxLevel; ++ q)
18     std::cout << (q ? " " : "") << BettiNumber (hom [q]);
19   std::cout << '\n';
20   delete [] hom;
21   return 0;
22 }
```

- クラス CubicalCell，CubicalComplex

$\mathbb{R}^d$ 内の一般の基本方体を表すクラスとしては CubicalCell が用意されている．そのコンストラクタとしては縮退していない基本方体を作る為に Cube が必要とした入力や，縮退している場合に対応する最小と最大の頂点座標を表した整数列の対などが必要となる．

方体複体を表すクラスは CubicalComplex で与えられ，そのオブジェクトに基本方体を付け加えるには関数 add を用いる．与えられた方体複体 C の次元は関数 dim で参照でき，また指定された次元 k の基本方体の集まりを取り出すには [] を用いて C[k] とすればよい．ここで方体複体を定めるには全ての基本方体の面をあらかじめ CubicalComplex に入力しておく必要は無く，必要な面は計算の際に自動的に補間される．ここでもホモロジー群の計算には Homology を用いればよい．

- クラス CubicalMap

最後に方体写像を表すクラス CubicalMap について説明しておく．これまでと同様に方体写像を扱う場合は全ての基本方体は縮退していないものとする．例えば Q を Cube のオブジェクト，F を CubicalMap のオブジェクトとすると，その像は SetOfCubes 型として F[Q] に格納される．像の指定にはこれまでにも出てきた基本方体を加える関数 add を用いるか，=を使って基本方体の集まりを直接与える方法がある．このようにして構成される CubicalMap の誘導準同型写像を計算する際にも関数 Homology を用いればよい．

参考文献

[1] T. Kaczynski, K. Mischaikow, and M. Mrozek, *Computational Homology*, Applied Mathematical Science Vol. 157, Springer, 2004.

[2] K. Mischaikow, M. Mrozek, P. Pilarczyk, Graph approach to the computation of the homology of continuous maps, Foundations of Computational Mathematics 5 (2005) 199–229.

[3] `http://chomp.rutgers.edu/`