グレブナー基底 : 計算機を使う観点からの導入

野呂 正行 (神戸大学大学院理学研究科 · JST CREST)
濱田 龍義 (福岡大学理学部 · JST CREST)

平成 21 年 9 月 14 日

1　各種ソフトウェアの準備

1.1　VMware Player または VMware Fusion のインストール

1.1.1　VMware Player のインストール

Windows の場合, VMware Player (無料) を入手, インストールする. Google で VMware Player を検索して, ダウンロードページに行く. 最新版は 9/3 時点で 2.5.3 である. 入手の際に, email アドレスを含む質問があるが, 正しく入力しても, いままで実際に email が届いたことは一度もない. ダウンロードできたらあとはインストールするだけである.

1.1.2　VMware Fusion のインストール

Intel Mac の場合, VMware Fusion (残念ながら有料) を入手, インストールする. こちらは売り物なので, 説明書その他に従って下さい.

1.2　仮想マシンのインストール

仮想マシンは, KNOPPIX/Math DVD 版の ISO イメージ (`knxm2008n-kobe.iso`; 4GB) と, 仮想マシン本体 (`math2008n-crest` フォルダ) からなる. ISO イメージは 4GB 弱のファイルである. 仮想マシン本体は, Windows 用の自己解凍 exe ファイル (`math2008n-crest.exe` 160MB) または Mac 用 zip ファイル (+verb+math2008n-crest.zip+; 240MB) を用いる.

外付け HDD を購入したままの状態で使っている場合, ファイルシステムが FAT32 の場合がある. また, Windows の C: 以外のパーティションも, FAT32 の場合がある. この場合, 仮想 HDD が 2GB ファイルに分割されている `math2008n-crest-2G.exe` または `math2008n-crest-2G.zip` を使う必要がある.

1. 仮想マシン本体の展開

   配布 DVD, USB メモリ, SD カード, USB ポータブル HDD などのうち, 都合のよいものから各自の PC の HDD に仮想マシン本体を展開する. 展開するパーティションは, 十分な空き領域のあるものを選ぶこと.

2. ISO イメージのコピー

   これも, なんらかのメディアから, 仮想マシンフォルダ内にコピーする.

1.3 仮想マシンの起動

仮想マシンフォルダ内の math2008n.vmx ファイルをダブルクリックする. こちらはメモリを 512MB 使用する設定であるが, もし実メモリが少ない場合には, math2008n-256M.vmx を使うこともできる. VMware Player->トラブルシューティング (Mac の場合仮想マシン->設定) から適当な大きさに変更することができる. メモリがふんだんにある場合には, 適宜増やすのもよい.

1.4 共有フォルダ

デフォルトでは共有フォルダは無効になっている. 仮想マシンが起動したあと, Windows または Mac 側で, 共有フォルダに指定するディレクトリを作成し, VMware Player->共有フォルダ (Mac の場合 仮想マシン->共有フォルダ) で共有フォルダを有効にし, 作成したディレクトリを共有フォルダに指定する. そのあと, 仮想マシンの下部のペンギンから Mount Shared Folder を実行すると, デスクトップの shared_folder アイコンがフォルダの形に変わる. このフォルダに KNOPPIX 側からファイルをドロップすると, ホスト側の対応するディレクトリにコピーされたことになる. 逆も同様である. 実体は /mnt/hgfs/shared_folder である. シェルからアクセスする場合はこのパス名を用いる.

Mac の場合, 一度デフォルトの設定を削除してから, 新規に共有フォルダを追加する必要がある. この場合, 実在のフォルダを共有フォルダに指定してから, 名前をクリックして shared_folder に変更する.

1.5 suspend, resume

仮想マシンウィンドウを×で閉じると, 現在の状態をセーブして suspend 状態となる. この状態で, ホストマシンをシャットダウンすることができる. この状態では, 仮想マシンフォルダ内に, 一時停止のマークのついたアイコンが見える. この状態で, vmx ファイルをダブルクリックすると, resume する.

1.6 プリンタの設定

ペンギン->Confugure->Configure Printer を実行すると, プリンタ設定のためのダイアログが現れる. PS プリンタを設定する場合には次の手順を実行する.

1. 追加->プリンタ/クラスを追加
2. バックエンド選択でリモート LPD キューを選択
3. LPD キュー情報でプリンタホスト, キュー名を入力

   実習で使用する B 棟 4 階のプリンタの場合

   ホスト : p-418.math.kobe-u.ac.jp

   キュー : PS_DUP
4. プリンタ機種選択で Postscript プリンタを選択

5. プリンタテスト->設定

   Page Size : A4

   Double-Sided Printing : Long Edge

   Miscellaneous->GhostScript pre-filtering : Convert to PS level2

6. 一般情報で名前をつける

   p-418 としておく.

以上により, lpr -Pp-418 ... でファイルが印刷できる.

1.7 数学ソフトウェアに関する文書の検索

数学ソフトウェアのマニュアル, 参考書はいくつかの場所に分散している. /usr/share/doc は, 種々の文書がおかれるディレクトリである. また, 日本語文書は/usr/local/Math-ja におかれていて, デスクトップの knoppix-math からリンクされている. これらの中から目的のものを探すのは骨が折れが, デスクトップの Math-Doc-Search を使えば, 大抵のものは簡単に探し当てることができる. 使い方は簡単で, Math-Doc-Search を起動し, Query にキーワード (日本語 OK) を並べてサーチするだけである.

2 Macaulay2 によるグレブナー基底の計算

2.1 Macaulay2 について

2.1.1 起動方法

- $\sqrt{x}$ メニュー, または KNOPPIX-Math-Start から起動する.

  Konsole が起動し, その中で Macaulay2 が起動する. (emacs) を選ぶと emacs のバッファ内で Macaulay2 が起動する. getting started で推奨されている使い方である.

- 端末エミュレータ (Konsole など) から起動する.

  自分で立ち上げた端末エミュレータのシェルからコマンド M2 を実行すると, その端末エミュレータ内で Macaulay2 が起動する.

2.1.2 ヘルプ, マニュアル

コマンド viewHelp を実行すると, ブラウザが起動する. 最初は,

```
Macaulay 2 -> getting started -> a first Macaulay 2 session
```

などをざっと眺めてみることをお勧めする. 個々のコマンドは, index から調べることができる.

2.1.3 ファイルのロード

ファイルのロードは load, パッケージのロードは loadPackage で行う.

2.2 基礎環および項順序

2.2.1 基礎環の宣言と多項式の入力

Macaulay2 では, 基礎環を明確に宣言する必要がある. 係数体として, 有理数体は QQ, 有限体 $\mathbf{F}_p = \mathbf{Z}/p\mathbf{Z}$ は ZZ/p により入力できる. ある基礎環が設定されている場合, そこに含まれない不定元を使用しようとしても拒否される. 各多項式はそれが生成された時の環に属するため, 異なる環に属する多項式は, たとえ含まれる不定元が一致していても, 一方を他方の環に写す (map を用いる) 必要がある.

基礎環の宣言と多項式の入力

```
i1 : R=QQ[x,y,z];
i2 : f=(x+y+z)^2
      2           2                   2
o2 = x  + 2x*y + y  + 2x*z + 2y*z + z
o2 : R
i3 : g=y+u
stdio:3:4:(1):[0]: error: no method for binary operator + applied to
objects:
--            y (of class R)
--      +     u (of class Symbol)

i4 : S=QQ[x,y,z,u]
i5 : f+u
stdio:5:2:(1):[0]: error: expected pair to have a method for '+'
i6 : h=(map(S,R))(f);
i7 : h+u
      2           2                   2
o7 = x  + 2x*y + y  + 2x*z + 2y*z + z  + u
o7 : S
```

2.2.2 項順序

項順序は基礎環を定義するときに指定する. デフォルトは全次数逆辞書式順序であり, それ以外の順序を指定する場合には MonomialOrder により指定する. 代表的なものを挙げる.

- 辞書式順序

  例 : QQ[x,y,z,MonomialOrder=>Lex]

  この例は, $x > y > z$ なる辞書式順序を持つ多項式環を宣言している.

- ブロック項順序

  例 : ZZ/37[x,y,z,u,v,MonomialOrder=>{2,3}]

  この例は, $\{x, y\} >> \{z, u, v\}$ で, 各ブロックに全次数逆辞書式を適用するブロック順序を持つ多項式環を宣言している.

多項式の先頭項は Macaulay2 では lead monomial と呼ばれ, `leadMonomial` で取り出せる. その係数は `leadCoefficient`, 係数つきの先頭項は `leadTerm` で取り出せる.

2.3 グレブナー基底の計算

本節では, イデアルの入力およびグレブナー基底計算について述べる. Macaulay2 でのイデアルの生成は `ideal`$(p_1, \ldots, p_l)$ による. グレブナー基底は `gb` で行う. この場合, 項順序は環が知っているので引数はイデアル 1 つのみである. 結果はグレブナー基底というオブジェクトで返される. 生成系は `gens` により行列 (行ベクトル) として取り出せる.

次の例は, cyclic-7 の全次数逆辞書式順序によるグレブナー基底計算である. 計算方法の制御を行うためのスイッチが多数用意されてはいるが, 特に指定を行わなくても比較的高速に計算が終了する. `gens` により生成系が 1×209 行列 `g` として得られる. その i 番目の要素は, `g_i` により取り出せる. i は 0 から始まることに注意する.

Macaulay2 によるグレブナー基底計算

```
i1 : R=QQ[c0,c1,c2,c3,c4,c5,c6];
i2 : I=ideal(c6*c5*c4*c3*c2*c1*c0-1,...
o2 : Ideal of R
i3 : G=gb I;
i4 : g=gens G;
              1       209
o4 : Matrix R  <--- R
i5 : g_0
o5 = | c0+c1+c2+c3+c4+c5+c6 |
       1
o5 : R
```

2.4 基本的な応用

2.4.1 イニシャルイデアルの計算

イデアル $I \subset R$ ($R = K[x_1, \ldots, x_n]$) のグレブナー基底 G の先頭項から I のイニシャルイデアル $\mathrm{in}(I)$ が得られる. R/I は G の先頭項で割れない単項式全体 M で K 上張られる. M の元を標準単項式 (standard monomial) と呼ぶ. 次の例では, まず I のイニシャルイデアル J の計算している. 実際には, グレブナー基底の計算が行われ, グレブナー基底の先頭項で生成されるイデアルとして J が得られる. I の次元を調べると 0 なので, 標準単項式は有限個である. これは R/J の $\mathbf{Q}$– 基底を与える `basis` により得られる. $\dim_K R/J = \dim_K R/I$ であり, この値は, $\overline{K}^n$ における I の零点の重複度込みの個数に等しい.

Macaulay2 によるイニシャルイデアルの計算

```
i1 :  R=QQ[x,y,z];
i2 : I=ideal(x^2*y^2-z^2,x^3-y*z^2,x^2*z^4-y^2);
o2 : Ideal of R
i3 : J=ideal leadTerm I
              3   2 2   3 2   5   6   2 4
o3 = ideal (x , x y , y z , y , z , x z )
o3 : Ideal of R
i4 : dim I
o4 = 0
i5 : S=R/J
o5 = S
o5 : QuotientRing
i6 : basis S
o6 = | 1 x x2 x2y x2yz x2yz2 x2yz3 x2z x2z2 x2z3 xy xy2 xy3 xy4    ...
   ------------------------------------------------------------- ...
   xy2z2 xy2z3 xy2z4 xy2z5 xyz xyz2 xyz3 xyz4 xyz5 xz xz2 xz3 xz4  ...
   ------------------------------------------------------------- ...
   y4 y4z y3z y2z y2z2 y2z3 y2z4 y2z5 yz yz2 yz3 yz4 yz5 z z2 z3 z4 z5 |
              1       52
o5 : Matrix S  <--- S
```

2.4.2 商および剰余の計算

Macaulay2 では, 多項式をグレブナー基底または行列 (イデアルの場合, 生成系を並べた行ベクトル) で割った商および剰余が計算できる. 関連する関数は以下の通りである.

- remainder(f, g) : f を g で割った剰余 r を返す.
- quotient(f, g) : f を g で割った商 q を返す.
- quotientRemainder(f, g) : f を g で割った商 q, 剰余 r に対し sequence (q, r) を返す.

引数 f は行列, g はグレブナー基底または行列である. g がグレブナー基底の場合, 商は 0 が返されるようである. g が行列の場合, $gq + r = f$ を満たす q, r が計算される. 例えば g がイデアルの生成系を並べた行ベクトルの場合, $q_0g_0 + \cdots + q_lg_l = f$ を満たす $q_0, \ldots, q_l$ が列ベクトルとして返される. 剰余計算の最も簡単な応用として, イデアル I, J に対する $I \subset J$ のテストがある. これは, J の任意項順序でのグレブナー基底 G を計算しておけば, I の生成系の各元の G による剰余が 0 となることを確かめることに帰着される.

商および剰余の計算

```
i1 : R=QQ[x,y,z];
i2 : I=ideal(x^4*y^2+z^2-4*x*y^3*z-2*y^5*z,x^2+2*x*y^2+y^4);
o2 : Ideal of R
i3 : G=gb I;
i4 : g=gens G;
             1       3
o4 : Matrix R  <--- R
i5 : f=y*z-x^3;
i6 : remainder(matrix{{f}},G)
o6 = | -x3+yz |
             1       1
o6 : Matrix R  <--- R
i7 : remainder(matrix{{f^2}},G)
o7 = | 2x2y3z+2x3yz+2y2z2+2xz2 |
             1       1
o7 : Matrix R  <--- R
i8 : remainder(matrix{{f^3}},G)
o8 = 0
             1       1
o8 : Matrix R  <--- R
i9 : qr=quotientRemainder(matrix{{f^3}},g);
o9 : Sequence
i10 : q=qr_0;
              3       1
o10 : Matrix R  <--- R
i11 : g*q
o11 = | -x9+3x6yz-3x3y2z2+y3z3 |
              1       1
o11 : Matrix R  <--- R
i12 : g*q-f^3
o12 = 0
```

この例では, イデアル I による f, f^2, f^3 の剰余を計算して, $f^3 \in I$ を示している. f^3 を G で割ることで商が得られる. さらに得られた商 q を G の生成系に掛けることで, $f^3 = gq$ が確かめられる. この場合には $f \notin I$ だが $f^3 \in I$ である. よって $f \in \sqrt{I}$ が示せたことになる. このやり方では, $f \notin \sqrt{I}$ を示すことはできない. $f \in \sqrt{I}$ の判定法は 2.4.4 節で説明する.

2.4.3 消去法

I を多項式環 $K[Z]$ $(Z = X \cup Y, X \cap Y = \emptyset)$ のイデアルとするとき, $I_Y = I \cap K[Y]$ の生成系は, $X >> Y$ なる任意の消去順序 $<$ に関する I のグレブナー基底 G に対し $G_Y = G \cap K[Y]$ により与えられる. さらに, G_Y は I_Y の $<_Y = <|_{K[Y]}$ に関するグレブナー基底になっている. 消去順序としては辞書式順序も使えるが, 計算効率の問題から, 通常はブロック順序を使うのが望

ましい. G を計算した後 G_Y を求めるには selectInSubring を用いる.

次の例では $I \cap \mathbf{Q}[z]$ の生成系を計算している. $\{x, y\}$ >> $\{z\}$ で, 各ブロックで全次数逆辞書式順序を適用するブロック順序を設定してグレブナー基底 G を計算したあと, 変数が z のみからなる多項式を G から取り出して Gz としている. selectInSubring(i, m) は, 行列 m から i 番目 (この場合は $i \geq 1$) までのブロックに属する変数を含まない列のみを取り出した行列を返す.

消去イデアルのグレブナー基底の計算

```
i1 : R=QQ[x,y,z,MonomialOrder=>{2,1}];

i2 : I=ideal(x^2-z,x*y-1,x^3-x^2*y-x^2-1);
o2 : Ideal of R
i3 : G=gens gb I;
              1       3
o3 : Matrix R  <--- R
i4 : Gz=selectInSubring(1,G)
o4 = | z3-3z2-z-1 |
```

2.4.4 イデアルの共通部分, イデアル商, saturation, radical メンバーシップ

イデアルの共通部分, イデアル商, saturation, $f \in \sqrt{I}$ の判定はは次の方法により行うことができる. $R = K[x_1, \ldots, x_n]$ とする.

- イデアルの共通部分

  R のイデアル $I = \langle f_1, \ldots, f_k \rangle$, $J = \langle g_1, \ldots, g_l \rangle$ に対し, t を新しい変数とすれば

  $$I \cap J = \langle tf_1, \ldots, tf_k, (1-t)g_1, \ldots, (1-t)g_l \rangle \cap R$$

  ただし, 右辺のイデアルは $K[x_1, \ldots, x_n, t]$ で考える. よって, 消去イデアル計算により共通部分が計算できる.

- イデアル商

  R のイデアル I, J に対し $I : J = \{f \mid fJ \subset I\}$ である. $J = \langle g_1, \ldots, g_l \rangle$ なら $I : J = \bigcap_{i=1}^{l} I : \langle g_i \rangle$ である. $I : \langle g \rangle$ を $I : g$ と書く. $I : g = (I \cap \langle g \rangle)/g$ である. 右辺は, $I \cap \langle g \rangle$ の生成系の各元を g で割ったもので生成されるイデアルである. よって, $I : J$ は共通部分計算により計算できる.

- saturation

  R のイデアル I, J に対し $I : J^{\infty} = \bigcup_{m=1}^{\infty} (I : J^m)$ である. $J = \langle g_1, \ldots, g_l \rangle$ なら $I : J = \bigcap_{i=1}^{l} (I : \langle g_i \rangle^{\infty})$ である. $I : \langle g \rangle^{\infty}$ を $I : g^{\infty}$ と書く. $I : g^{\infty} = (I + \langle tg - 1 \rangle) \cap R$ である. ただし右辺のイデアルは $K[x_1, \ldots, x_n, t]$ で考える. よって saturation は共通部分計算により計算できる.

- $f \in \sqrt{I}$ の判定 (radical メンバーシップ判定)

  R のイデアル I, $f \in R$ に対し, $f \in \sqrt{I} \Leftrightarrow I + \langle tf - 1 \rangle = K[x_1, \ldots, x_n, t]$ である. $I + \langle tf - 1 \rangle = K[x_1, \ldots, x_n, t]$ は $I + \langle tf - 1 \rangle$ の (任意項順序に関する) 簡約グレブナー基底が $\{1\}$ であることと同値だから, $f \in \sqrt{I}$ か否かはグレブナー基底を計算することで判定できる.

Macaulay 2 においては, イデアル $I_1, I_2, \ldots$ の共通部分は `intersect`$(I_1, I_2, \ldots)$, イデアル I, J に対するイデアル商 $I : J$ は `quotient`(I, J), saturation $I : J^\infty$ は `saturate`(I, J) により計算できる.

イデアルの演算

```
i1 : R=QQ[x,y];
i2 : I=ideal(x^4-y^5,x^3-y^7);
o2 : Ideal of R
i3 : I1=quotient(I,x)
             5    4   3 2    2   7    2 3
o3 = ideal (y  - x , x y  - x , x  - x y )
o3 : Ideal of R
i4 : I2=quotient(I,x^2)
             2 2       5    4   6      3
o4 = ideal (x y  - x, y  - x , x  - x*y )
o4 : Ideal of R
i5 : I3=quotient(I,x^3)
               2       5    4   5    3
o5 = ideal (x*y  - 1, y  - x , x  - y )
o5 : Ideal of R
i6 : I4=quotient(I,x^4)
               2       5    4
o6 = ideal (x*y  - 1, y  - x )
o6 : Ideal of R
i7 : J=saturate(I,x)
               2       5    4   5    3
o7 = ideal (x*y  - 1, y  - x , x  - y )
o7 : Ideal of R
i8 : I3==I4
o8 = true
i9 : I2==I3
o9 = false
```

次の例は, 2.4.2 節の例における $f \in \sqrt{I}$ の判定を, ここで述べた方法で行ったものである. グレブナー基底が $\{1\}$ なので, $f \in \sqrt{I}$ と判定できる.

radical メンバーシップ判定

```
i1 : R=QQ[t,x,y,z];
i2 : I=ideal(x^4*y^2+z^2-4*x*y^3*z-2*y^5*z,x^2+2*x*y^2+y^4);
o2 : Ideal of R
i3 : f=y*z-x^3;
i4 : gens gb (I+ideal(t*f-1))
o4 = | 1 |
```

3 Asir によるグレブナー基底の計算

3.1 Asir について

3.1.1 起動方法

- $\sqrt{x}$ メニュー, または KNOPPIX-Math-Start から起動する.
  (openxm) の方を起動すれば, 種々のライブラリファイルを自動的に読みこんで起動する.
- 端末エミュレータから起動する.
  Asir 単体ではコマンドライン編集機能を持たないので, `openxm fep asir` を実行する.

3.1.2 ヘルプ, マニュアル

ヘルプは `help("function")` で引ける. マニュアルはデスクトップの Math-Doc-Search で引くか, `helph()` コマンドでブラウザを立ち上げて HTML 形式のマニュアルを見るのが便利である.

3.1.3 ファイルのロード

ファイルは `load` により行う. 環境変数 `ASIRLOADPATH` で指定されたディレクトリを順に探す. この値は, シェルから `openxm env` を実行すると見ることができる.

3.2 基礎環および項順序

3.2.1 多項式の入力

Asir ではアルファベット小文字で始まり, アルファベット, 数字, _ (アンダースコア) からなる文字列が不定元である. 不定元を含む多項式が入力された場合, システムが内部的に保持する不定元の順序に従い内部形式に変換され保持される. 通常係数は自動的に有理数と判断される. すなわち, この場合には, 有理数係数の無限変数のユニバーサルな多項式環が設定されていると考えてよい. よって, 保持している多項式は, 内部の不定元順序が変更されない限り, 自由に加減乗算が可能である. しかし, 項順序に関する情報はなんら保持されていないため, グレブナー基底関連計算など, 項順序が必要が計算の都度, 項順序を指定する必要がある.

多項式の入力

```
[1518] F=(x+y+z)^2;
x^2+(2*y+2*z)*x+y^2+2*z*y+z^2
[1519] G=F+u;
x^2+(2*y+2*z)*x+y^2+2*z*y+z^2+u
```

この表示から分かるように, Asir における多項式は再帰表現により保持されている. 再帰表現とは, 多項式を, 主変数に関する一変数多項式として表現するもので, 係数は, 主変数を含まない多項式である. これに対し, グレブナー基底に関連する計算では, 多項式を単項式の和として表現するのが便利である. これを分散表現と呼ぶ. Asir においては, グレブナー基底関連計算を行う場合, 暗黙あるいは明示的に分散表現への変換を行う. これについては後述する.

3.2.2 項順序

Asir においては, 項順序は変数順序と項順序型により指定される. 変数順序は不定元を並べたリストで表現する. この順序は単項式を指数ベクトルで表示する場合の各指数のインデックスを決める. 例えば, 変数順序が $[x, y, z, u, v, w]$ で与えられた場合, $x^a y^b z^c u^d v^e w^f$ は (a, b, c, d, e, f) で表示される. n 変数の変数リストが与えられているとき, 次のような項順序型が設定できる.

- 単純な項順序型

  整数値で表される. 0 は全次数逆辞書式, 1 は全次数辞書式, 2 は辞書式順序を表す. これらは, 上の変数順序によりベクトルで表された単項式に対し適用される.

- ブロック項順序型

  $[[O_1, n_1], [O_2, n_2], \ldots, [O_l, n_l]]$ なるリストのリストで表される. これは, 変数リストを左から $n_1, n_2, \ldots, n_l$ $(n_1 + \cdots + n_l = n)$ ずつのブロックに分け, i 番目のブロックに単純項順序型 O_i を適用する項順序である. 項順序比較は, 1 番目のブロックから, 大小が決まるまで順に行う. よく使われるのは $[[0, n_1], [0, n_2]]$ なるもので, 先頭の n_1 変数を消去するための消去順序の一つである.

- 行列による項順序型

  $m \times n$ 整数行列 M で表される. これは, 二つの単項式 $e = (e_1, \ldots, e_n)$, $f = (f_1, \ldots, f_n)$ に対し

  $$e > f \Leftrightarrow M(e - f) \text{ の } 0 \text{ でない最も上の要素が正}$$

  として定義される. M が項順序を表すためには,

  - 整数ベクトル e に対し $Me = 0 \Leftrightarrow e = 0$
  - 各列の 0 でない最も上の要素が正

  という条件を満たす必要があるが, これを保証するのはユーザの責任である.

項順序型は, dp_ord により設定できる. あるいは関数の引数として与える場合もある.

分散表現への変換

```
[1532] F=x^2*y+y^3*z+x*z+x+1;
y*x^2+(z+1)*x+z*y^3+1
[1533] dp_ord(0)$
[1534] DF0=dp_ptod(F,[x,y,z]);
(1)*<<0,3,1>>+(1)*<<2,1,0>>+(1)*<<1,0,1>>+(1)*<<1,0,0>>+(1)*<<0,0,0>>
[1535] dp_ord(2)$
[1536] DF2=dp_ptod(F,[x,y,z]);
(1)*<<2,1,0>>+(1)*<<1,0,1>>+(1)*<<1,0,0>>+(1)*<<0,3,1>>+(1)*<<0,0,0>>
[1537] G=F+u;
y*x^2+(z+1)*x+z*y^3+u+1
[1538] DG=dp_ptod(G,[u,x,y,z]);
(1)*<<1,0,0,0>>+(1)*<<0,2,1,0>>+(1)*<<0,1,0,1>>+(1)*<<0,1,0,0>>
+(1)*<<0,0,3,1>>+(1)*<<0,0,0,0>>
[1539] dp_ht(DG);
(1)*<<1,0,0,0>>
```

この例では, 多項式 F, G を明示的に分散表現に変換している. dp_ord(0) により DF0 は全次数逆辞書式で整列され, dp_ord(2) により DF2 は辞書式に整列される. また, DG は, u が最大の辞書式となるため, <<1,0,0,0>> が先頭となっている. dp_ht は先頭項 (係数 1), dp_hc は先頭係数, dp_hm は係数つきの先頭項を返す. (用語は初期に使われていたものを採用しており, 最近の用法と異なることに注意されたい.) 分散表現多項式間の演算は, 同じ項順序で変換されたものに限られる. Macaulay2 と異なり, これを守るのはユーザの責任である.

3.3 グレブナー基底の計算

本節では, イデアルの入力およびグレブナー基底計算について述べる. 以下で述べる関数はライブラリ gr に定義されているものも gr をロードしておく必要がある. ただし, KNOPPIX/Math にインストールされている環境では, デフォルトでロードされているので操作は不要である.

Asir では, イデアルは多項式のリストで表現される. この段階では基礎環にあたるものはまだ未定である. グレブナー基底計算の際に指定される変数リストと項順序型, および係数体を指定する引数により基礎環が係数体を含めて決定される. 主なグレブナー基底計算関数を挙げる.

- nd_gr(*Plist*, *Vlist*, *Char*, *Ord*)

  Plist はイデアルを表す多項式リストである. 変数リスト *Vlist*, 項順序型 *Ord* で指定される項順序をもつ多項式環のイデアル $\langle Plist \rangle$ の簡約グレブナー基底を計算する. $Char = 0$ のとき有理数体係数, *Char* が素数のとき有限体 $\mathbf{F}_{Char}$ 上で計算する. 結果は多項式のリストである.

- nd_gr_trace(*Plist*, *Vlist*, *Homo*, *Prime*, *Ord*)

この関数は有理数体上のグレブナー基底を, 有限体上でのショートカット計算を用いて効率よく計算するための関数である.

Plist はイデアルを表す多項式リストである. 変数リスト *Vlist*, 項順序型 *Ord* で指定される項順序をもつ多項式環のイデアル $\langle Plist \rangle$ の簡約グレブナー基底を計算する. *Prime* は 1 を指定しておく. (他の値の場合はマニュアルを参照.) *Homo* が 1 のとき, 斉次化を経由して計算する. *Homo* が 0 のとき斉次化を経由しないで計算する. どちらも結果は同一であるが, 中間係数が膨張する可能性がある場合, *Homo* = 1 で計算するのが安全である. (Macaulay2 のソースコードで確かめたわけではないが, 中間基底の生成の様子を見ると, Macaulay2 はデフォルトで斉次化を行っているように見える.)

Asir によるグレブナー基底計算

```
[1517] load("cyclic")$
[1527] C=cyclic(7);
[c6*c5*c4*c3*c2*c1*c0-1,...]
[1528] V=vars(C);
[c0,c1,c2,c3,c4,c5,c6]
[1529] nd_gr(C,V,31991,0)$
...
2.016sec + gc : 0.072sec(2.089sec)
[1530] nd_gr(C,V,0,0)$
(5 分待って中断)
[1530] G=nd_gr_trace(C,V,1,1,0)$
...
19.54sec + gc : 5.428sec(25.02sec)
[1531] G[0];
(((23853922665902000713066 2*c6*c4-28276599708297972450024 2*c5^2-...
[1532] length(G);
209
```

この例では, 有名なベンチマーク問題 cyclic-7 の全次数逆辞書式順序グレブナー基底計算を有限体 $\mathbf{F}_{31991}$ および有理数体上で行っている. 計算時間は, Intel Xeon X5470 (3.33GHz) でのものである (以下同様). 最初の nd_gr は有限体上の計算で, 2 秒程で終っているが, これを有理数体上で行う (2 番目の nd_gr) と, 計算途中で係数膨張のため計算が進まなくなる. 一方で, nd_gr_trace を *Homo* = 1 で実行すると, 25 秒で計算が終了する. この例に限らず, 有理数体上の計算の場合, 常に係数膨張の危険があるため, *Homo* = 1 で計算することを推奨する.

計算結果のリストは, 入力と同一のイデアルを生成するグレブナー基底である. リスト *G* の *i* 番目の要素は *G*[*i*] (*i* は 0 から始まる) で取り出せる.

3.4 基本的な応用

3.4.1 イニシャルイデアルの計算

Asir でイニシャルイデアルを計算するには, まずグレブナー基底を計算し, その各生成元の先頭項を取り出す. このためには, 再帰表現された多項式を dp_ptod で分散表現に変換し,

dp_ht で先頭項を取り出すという操作が必要となる. さらに, 例では dp_dtop により各先頭単項式を再帰表現に逆変換している. また, 0 次元の場合のみ, dp_mbase により, 標準単項式全体を計算できる.

Asir によるイニシャルイデアルの計算

```
[1517] B=[x^2*y^2-z^2,x^3-y*z^2,x^2*z^4-y^2];
[y^2*x^2-z^2,x^3-z^2*y,z^4*x^2-y^2]
[1518] V=[x,y,z]$
[1519]  G=nd_gr(B,V,0,0);
[z^4*x^2-y^2,-y^4+z^6,-y^2*x+y^5,-z^2*x+z^2*y^3,y^2*x^2-z^2,x^3-z^2*y]
[1520] D=map(dp_ptod,G,V)$ H=map(dp_ht,D)$
[1521] [1522] map(dp_dtop,H,V);
[z^4*x^2,z^6,y^5,z^2*y^3,y^2*x^2,x^3]
[1523] map(dp_dtop,dp_mbase(H),V);
[z^5*y^2*x,z^4*y^2*x,z^5*y*x,z^5*y^2,z*y^4*x,z^3*y*x^2,...]
[1524] length(@@);
52
```

3.4.2 剰余計算

多項式を多項式リストに現れる多項式で割った剰余を計算する関数が p_nf および p_true_nf である. 前者は, 剰余の分母を払って, 整数係数で返すもので, 剰余が 0 か否かの判定に用いる. 後者は, [*num*, *den*] なるリストを返す. *num* は p_nf が返すもので, *num*/*den* が真の剰余となる.

剰余計算

```
[1517] B=[u2*u0-2*u2+3,(2*u1-1)*u0^2-u0-2*u2,2*u1^3+u2+4]$
[1518] V=[u0,u1,u2]$
[1519] G=nd_gr(B,V,0,0);
[10*u2^4+126*u2^3+637*u2^2+(586*u1-907)*u2-816*u0^2-588*u0-216*u1^2-...]
[1520] Q=p_nf(u0^5+u1^5+u2^5,G,V,0);
2851262910*u2^3+30078832770*u2^2+(22194374760*u1-21995962245)*u2-...
[1521] QR=p_true_nf(u0^5+u1^5+u2^5,G,V,0);
[2851262910*u2^3+30078832770*u2^2+(22194374760*u1-...,35373600]
```

3.4.3 消去法

2.4.3 節で述べたように, $K[Z]$ ($Z = X \cup Y$, $X \cap Y = \emptyset$) のイデアル I に対する消去イデアル $I_Y = I \cap K[Y]$ の生成系の計算には消去順序によるグレブナー基底を使う. 有理数体上で計算する場合には nd_gr_trace を斉次化ありで使うことをお勧めする. I の消去順序によるグレブナー基底 G から I_Y のグレブナー基底 G_Y を取り出すには, elimination (ライブラリ primdec_mod に定義されているがマニュアルにはない) を使う.

次の例では, B が生成するイデアルの $\{u0, u1\} >> \{u2\}$ なる消去順序によるグレブナー基底を計算し, elimination により $u2$ のみを含む多項式を取り出している.

消去イデアルの計算

```
[1518] load("primdec_mod")$
[1664] B=[u2*u0-2*u2+3,(2*u1-1)*u0^2-u0-2*u2,2*u1^3+u2+4]$
[1665] V=[u0,u1,u2]$
[1666] G1=nd_gr_trace(B,V,1,1,[[0,2],[0,1]])$
[1667] elimination(G1,[u2]);
[8*u2^9+72*u2^8+292*u2^7-2036*u2^6-198*u2^5+20682*u2^4-57429*u2^3+...]
```

3.4.4 最小多項式の計算

消去法の特別な例として, イデアルと一変数多項式環の共通部分 $I \cap K[z]$ の計算がある. これも前節で述べた一般的な方法で計算できるが, I が 0 次元イデアルの場合, より直接的な方法により $I \cap K[z]$ の単項生成元のみを計算することができる. gr で定義されている minipoly は, 有理数体係数多項式環の 0 次元イデアル I および多項式 f に対し, $m(f) \in I$ を満たすような 0 でない最小次数の多項式 m を計算する.

次の例では, これも有名なベンチマーク問題 *katsura* − 7 において, $u7$ の最小多項式を minipoly により求めている. minipoly の引数は, 最初の $(G, V, 0)$ で, グレブナー基底と項順序, $u7$ が, 最小多項式を計算したい多項式, t が, 最小多項式の変数の指定である. 最後の変数は, V に現れないものを指定する必要がある. この例を, 消去順序グレブナー基底計算で行ってみれば, その大変さがよくわかる.

最小多項式の計算

```
[1518] load("katsura")$
[1522] B=katsura(7)$
[1523] V=[u0,u1,u2,u3,u4,u5,u6,u7]$
[1524] G=nd_gr_trace(B,V,1,1,0)$
[1525] minipoly(G,V,0,u7,t)$
[1526] deg(@@,t);
128
```

3.4.5 0 次元イデアルの項順序変換

代数方程式の求解を行う場合, 辞書式, あるいはそれに近い消去順序でグレブナー基底を計算する必要が出て来るが, これらを Buchberger アルゴリズムで直接計算するのは一般に効率が大変悪い. このような場合, 例えば全次数辞書式順序でグレブナー基底を計算し, それから線形代数的手法により別の項順序に関するグレブナー基底を計算する方法がいくつかある. gr で定義されている tolex は, 0 次元イデアルの任意項順序のグレブナー基底を入力として, 辞書式順序のグレブナー基底を計算する関数である. この場合, 出力の変数順序のみ変更可能である. 次の例は, *katsura* − 7 の辞書式順序グレブナー基底を項順序変換で計算したものである. これを nd_gr_trace で直接辞書式順序で計算することは, 途中の係数膨張の様子からみてほぼ不可能であろう.

項順序変換による辞書式順序グレブナー基底の計算

```
[1524] G=nd_gr_trace(katsura(7),V=[u0,u1,u2,u3,u4,u5,u6,u7],1,1,0)$
2.676sec + gc : 1.356sec(4.032sec)
[1525] G2=tolex(G,V,0,V)$
279.5sec + gc : 57.68sec(337.5sec)
```

3.4.6 イデアル演算

イデアルの共通部分, イデアル商, saturation を計算する関数は, ライブラリのあちこちで定義され使われているが, マニュアルに書かれていないので, 消去イデアル計算を用いてこれらを実装してみると, よい練習になるであろう. 計算方法は 2.4.4 節にある.

4 練習問題

参考文献にあげた図書には, 種々のソフトウェアの使用法の解説とともに, 多数の練習問題が収録されている. 以下の問題のいくつかはこれらから引用した.

1. 以下の各項を行う方法を, Macaulay2, Asir それぞれについて調べよ.
   (a) ファイルに結果を書き出す.
   (b) 繰り返しを行う.
   (c) 多項式の因数分解を行う.
2. $\mathbf{Q}[x,y,z]$ のイデアル $I = \langle x^2+z, xy+y^2+z, xz-y^3-2yz, y^4+3y^2z+z^2\rangle$, $J = \langle x^2+z, xy+y^2+z, x^3-yz\rangle$ の包含関係を調べよ.
3. $f_1 = 3x^2yz^2+3z+(-2x+2)y+2x$, $f_2 = 3yz^5+(-xy^2+2)z-2y^4+2y$, $f_3 = xy^3z^3-2yz^2-z-2y+x^2$ に対し, $I = \langle f_1, f_2, f_3\rangle \subset \mathbf{Q}[x,y,z]$ とおく.
   (a) $\dim I = 0$ を確かめよ.
   (b) $\dim_{\mathbf{Q}} \mathbf{Q}[x,y,z]/I$ を求めよ.
   (c) I の $x > y > z$ なる辞書式順序での簡約グレブナー基底が $\{g_0(z), x-g_1(z), y-g_2(z)\}$ という形であることを確かめよ.
4. $f_1 = x^2+y^2+z^2-9$, $f_2 = 3x^2-y^2z$, $f_3 = x^2z-2y^2+2$ に対し, $f_1 = f_2 = f_3 = 0$ を満たす $(x,y,z) \in \mathbf{C}^3$ を全て求めよ.
5. $\alpha = 3^{\frac{1}{5}}, \beta = 5^{\frac{1}{3}}$ とする.
   (a) $\alpha+\beta$ の $\mathbf{Q}$ 上の最小多項式を求めよ.
   (b) $\frac{1}{\alpha+\beta}$ を α, β の有理数係数多項式で表せ.
6. (Asir でのプログラミング経験がある人向け) イデアルの共通部分, イデアル商, saturation の計算および radical メンバーシップを判定する関数を記述せよ.

参考文献

[1] W. Adams, P. Loustaunau, An Introduction to Gröbner basis Bases. Graduate Studies in Mathematics, Vol. 3, AMS (1994).

[2] D. Cox, J. Little, D. O'Shea, Using Algebraic Geometry. GTM Vol. 185, Springer (2005).

[3] D. Eisenbud, D. Grayson, M. Stillman, B. Sturmfels (Eds.), Computations in Algebraic Geometry with Macaulay 2. Algorithms and Computation in Mathematics **8**, Springer-Verlag (2000).

[4] G.-M. Greuel, G. Pfister, A Singular Introduction to Commutative Algebra. Springer (2007).

[5] M. Kreuzer, L. Robbiano, Computational Commutative Algebra 1. Springer (2008).