ソフトウェアによるグレブナー基底計算

野呂 正行 (神戸大学大学院理学研究科 · JST CREST)

February 21, 2012

グレブナー基底の計算ができるソフトウェア

グレブナー基底計算を行うことができるソフトウェアはたくさんある.

- **商用のもの**：Mathematica, Maple, Magma
  後者 2 つは信頼性に問題があるし, わざわざお金を出して買わなくてもフリーソフトで十分
- KnoppixMath **に含まれているフリーなもの**
  道場本第 3 章では Macaulay2, Singular, CoCoA, Risa/Asir **について紹介.**
  特殊用途専用のソフトはさらにいろいろある.

Macaulay2, Singular, CoCoA, Risa/Asir

- Macaulay2, Singular, CoCoA : **計算を始める前に, 基礎環の定義をする.**
  - **一つの環を固定して種々の計算を行う場合には都合がよい.**
  - **環を固定するというのは, 項順序を固定するという意味もあるため, 例えば途中で消去計算を行いたい, 斉次化を行いたいなどの場合に煩わしい操作, 不自然な操作を強いられる場合もある.**
- Risa/Asir : **計算の都度, 変数順序 (イデアルが属する環) および項順序を指定する.**
  - **常に変数, 項順序を意識する必要があり煩わしい.**
  - **消去計算, 斉次化などは簡単に行える.**

予定

- 初心者に敷居が低いであろう Risa/Asir を用いてグレブナー基底, 関連するイデアル演算について説明する.
- 環の定義を必要とするソフト代表として Macaulay2 を用いて同様の計算をどうやって行うか説明する.
- 配布資料に基づいて実習を行う.
  実行結果 (結果はこれこれ, うまく計算できた, よくわからない, バグではないか etc.) はチェックシートに書いて提出をお願いします (スクール終了時で OK です).
- 実習用データは
  `http://www.math.kobe-u.ac.jp/~noro/` に掲載.
  (kobe noro で検索)

ソフトウェアによるグレブナー基底計算の心得 1

ソフトウェアは複雑な計算を高速に間違いなく実行できるが, 使い方を間違えると計算が滞る.

- 一般に, 全次数逆辞書式順序の計算は高速である.
  この計算結果を改めて入力とすれば, その後の計算がスムーズになる場合もある.

- 純辞書式順序の計算は停滞しやすい.
  対処法: 斉次化してから計算する.
  あるいは, 一旦他の計算しやすい順序 (例:全次数逆辞書式順序) でグレブナー基底計算してから, 基底変換法 (FGLM, Hilbert driven アルゴリズム) などを適用する.

ソフトウェアによるグレブナー基底計算の心得2

- 消去イデアルを純辞書式順序で計算すると停滞しやすい.
  対処法: 消去イデアル計算は消去順序 (ブロック順序; 後述) で行う.
  この計算も斉次化したほうが安全である.
- 非斉次イデアルの計算は停滞しやすい.
  対処法: 斉次化してから計算する.
- 係数体が無限体の場合の計算は停滞しやすい.
  対処法: 斉次化してから計算する.

⇒ どのような場合でも, 計算が滞る場合はとりあえず斉次化を試みるのが有効.

斉次化

- **多項式の斉次化**
  $f \in R = K[x_1, \ldots, x_n]$, $f \neq 0$ **に対し**, f **の斉次化**

$$f^h = x_0^{\mathrm{tdeg}(f)} f(x_1/x_0, \ldots, x_n/x_0) \in R[x_0] = K[x_0, \ldots, x_n]$$

  で定義する ($\mathrm{tdeg}(f)$ **は全次数**).

- R **の項順序** $<$ **の斉次化** $<^h$
  $<^h$ **を**, $R[x_0]$ **の項順序で**, **単項式** $t, s \in R$ **に対し**

$$x_0^i t <^h x_0^j s \Leftrightarrow i + \mathrm{tdeg}(t) < j + \mathrm{tdeg}(s) \text{ または}$$
$$(i + \mathrm{tdeg}(t) = j + \mathrm{tdeg}(s) \text{ かつ } t < s)$$

  と定義する.
  すなわち, **全次数比較を行ったあと**, $(x_1, \ldots, x_n)$ **を先に**
  元の順序で比較するブロック順序を適用する.

- **斉次多項式** $h \in R[x_0]$ **の非斉次化**

$$h|_{x_0=1} = h(1, x_1, \ldots, x_n)$$

斉次化によるグレブナー基底計算

$F = \{f_1 \ldots, f_m\} \subset K[x_1, \ldots, x_n]$ とする.

斉次化による $\langle F \rangle$ のグレブナー基底計算

1. $G^h \leftarrow \langle f_1^h, \ldots, f_m^h \rangle$ の $<^h$ に関するグレブナー基底
2. $G \leftarrow G^h|_{x_0=1}$
   G は $\langle F \rangle$ のグレブナー基底となる
3. $G \leftarrow G$ から冗長元を省いたもの
   すなわち, 他の元の先頭項で割れる先頭項をもつものを捨てる
4. $G \leftarrow G$ を相互簡約したもの
5. return G

項順序の斉次化

- 一般の場合 $\Rightarrow$ 定義どおり全次数つきのブロック順序.
- 全次数逆辞書式順序 $\Rightarrow$ 再び全次数逆辞書式順序でよい.
- 純辞書式順序 $\Rightarrow$ 全次数辞書式順序でよい.

Asir

- **起動方法**
  - Math **メニューから起動する**.
  - **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**.
- **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**.

Asir : ファイルのロード, 実行中断

- **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.**

- **計算の途中での中断**
  1. `Control-c` (`Control` **キーを押しながら** `c`) **を打つと,** `interrupt ?(q/t/c/d/u/w/?)` **と表示される.**
  2. `t` **を入力して** `Return` (`Enter`) **を打つと**
  3. `Abort this computation? (y or n)` **と表示される.**
  4. `y` **により計算が中断され, トップレベルに戻る.**

多項式の入力

- **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.** (x,y,a0 etc.)
- **アルファベット大文字で始まり, アルファベット, 数字, _ (アンダースコア) からなる文字列が変数 (入れ物) である.** (X,Y,A0 etc.)
- **掛け算は *, べき乗は ^ で, 式の末端にセミコロン ; が必要である.**

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
```

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.
- 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 $(0, 1, 2)$
    0:全次数逆辞書式, 1:全次数辞書式, 2:純辞書式順序
  - ブロック項順序型 $[[O_1, n_1], [O_2, n_2], \ldots, [O_l, n_l]]$
    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.
- 行列による項順序型 (省略)
- 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

項順序に関係する演算を行うには, dp_ptod **により分散表現に変換する必要がある**. **項順序は** dp_ord **で設定したものが適用される**. dp_ht **で先頭項を取り出す**.

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>>
```

実習 1

一日目の実習で手作業で行った項順序による整列を,
Risa/Asir **で実行してみよ**.

グレブナー基底の計算

gr および noro_pd.rr をロードしておく.

- nd_gr($Plist, Vlist, Char, Ord$)
  $Plist$: 多項式リスト, $Vlist$: 変数リスト, Ord : 項順序型
  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.

- nd_gr_trace($Plist, Vlist, Homo, 1, Ord$)
  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.
  $Homo : 1$ のとき斉次化を経由
  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.

- 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)

実行例

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
```

実習 2

上の例の cyclic-7 **の全次数辞書式順序グレブナー基底計算を, 有限体上, 有理数体上で実行してみよ.**
実行前に `dp_gr_print(1);` **を実行すると, 計算の過程が見える.**

手動斉次化によるグレブナー基底計算

- homogenize(F, V, H) : F を斉次化変数 H により斉次化する.
- nd_gr_postproc$(G, V, Char, Ord, 0)$

  項順序 (V,Ord) でグレブナー基底となっている G から簡約グレブナー基底を計算する.

斉次化によるグレブナー基底計算

```
[1983] B=[z2+z1-x1,z2^21+z1^21-x2,x1^4096-x1,x2^4096-x2,z1^46-z1,
z2^46-z2,z1*z2*sdiv(z1^45-z2^45,z1-z2)]$
[1984] V=[z1,z2,x2,x1]$
[1985] Bh=map(homogenize,B,V,h)$
[1986] Vh=append(V,[h])$
[1987] dp_gr_print(1)$
[1988] Gh=nd_gr(Bh,Vh,2,1)$
21+46++47.+50+53+(省略)4184........4185..4186....4187..4188.nd_gb done.
[1989] G=map(subst,Gh,h,1)$
[1990] G=nd_gr_postproc(G,V,2,2,0);
......
[x1^721+x1^676+x1^631+x1^586+x1^541+x1^496+x1^451+x1^406+x1^361+...]
```

実習 3

上の例を斉次化あり/なしで実行してみよ.

注意:

- 仮想マシンに割り当てるメモリが 1GB 程度ないと危険
- 実は全次数逆辞書式 $\Rightarrow$ 辞書式の方が簡単

イニシャルイデアルの計算

1. グレブナー基底を計算する.
2. dp_ptod で分散表現に変換し, dp_ht で先頭項を取り出す
3. dp_dtop により各先頭単項式を再帰表現に逆変換する.

0 次元の場合, 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
```

実習 4

$I = \langle abcd-1, abc+bcd+cda+dab, ab+bc+cd+da, a+b+c+d \rangle \subset \mathbf{Q}[a,b,c,d]$ の 0 次元性を判定せよ.
(I のイニシャルイデアルが, 各変数に関してその変数のみのベキを含むとき I は 0 次元イデアルであるという.)

剰余計算

- p_nf(F, G, V, Ord)

  F を多項式リスト G (イデアル $\langle G \rangle$) で割った剰余の定数倍を計算する. 通常は G は (V, Ord) に関するグレブナー基底.

- p_true_nf(F, G, V, Ord)

  p_nf と同様だが, $[num, den]$ 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]
```

実習 5

$a+b+c=1, a^2+b^2+c^2=2, a^3+b^3+c^3=3$ **のとき,**
$a^{10}+b^{10}+c^{10}$ **の値を求めよ.**

消去法

$Z = X \cup Y, X \cap Y = \emptyset$ とする.

- $K[Z]$ のイデアル I に対する $I \cap K[Y]$ の計算

  消去順序によるグレブナー基底で計算できる. すなわち, $X \gg Y$ なるブロック順序で I のグレブナー基底 G を計算すれば $G \cap K[Y]$ が $I \cap K[Y]$ のグレブナー基底となる.

- 有理数体上での計算

  `nd_gr_trace` を斉次化ありで使うとよい.

- `noro_pd.elimination(G,Y)`

  I の消去順序グレブナー基底 G から $I \cap K[Y]$ のグレブナー基底の取り出し

消去イデアルの計算

```
[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] noro_pd.elimination(G1,[u2]);
[8*u2^9+72*u2^8+292*u2^7-2036*u2^6-198*u2^5+20682*u2^4-57429*u2^3+...]
```

消去法の応用として $x_1 = f_1(t_1, \ldots, t_m), \ldots, x_n = f_n(t_1, \ldots, t_m)$ **を代入すると** 0 **になる** $x_1, \ldots, x_n$ **の多項式全体のなすイデアル (パラメタ** $t_1, \ldots, t_m$ **を消去したイデアル) の計算ができる. これは** $I = \langle x_1 - f_1, \ldots, x_n - f_n \rangle$ **に対し** $I \cap K[x_1, \ldots, x_n]$ **を求めればよい.**

実習 6

$I = \langle b - a^2, c - a^3, d, e - 1, x - a + 2a(y - b) + 3a^2(z - c), z - f, (x - a)^2 + (y - b)^2 + (z - c)^2 - (x - d)^2 - (y - e)^2 - (z - f)^2 \rangle \subset \mathbf{Q}[a, b, c, d, e, f, x, y, z]$ **に対し** $I \cap \mathbf{Q}[x, y, z]$ **を求めよ.**

直接純辞書式順序で計算, 斉次化して計算, ブロック順序で計算など, さまざまな方法を試し, ブロック順序の有効性を確認せよ.

実習 7

$x = t^3, y = t^4, z = t^5$ から t を消去したイデアルを求めよ.
このイデアルが, $\langle xz - y^2, x^3 - yz, x^2y - z^2 \rangle$ と等しいことを確認せよ. (項順序によっては自明)

イデアルと一変数多項式環の共通部分 $I \cap K[z]$ の計算

- **一般の場合**：**消去イデアル計算**
- **0 次元の場合**：**直接的な方法により $I \cap K[z]$ の単項生成元のみを計算**
  `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**.

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
```

実習 8

$2^{\frac{1}{3}} + 3^{\frac{1}{4}} + 5^{\frac{1}{5}}$ の Q 上の最小多項式を求めよ.

($\langle a^3 - 2, b^4 - 3, c^5 - 5, t - (a + b + c)\rangle$ を考える.)

イデアルの共通部分, イデアル商

- **イデアルの共通部分** :
  noro_pd.ideal_intersection(I, J, V, Ord)
  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**

  $$I \cap J = \langle tf_1, \ldots, tf_k, (1-t)g_1, \ldots, (1-t)g_l \rangle \cap R \quad (inK[x_1, \ldots, x_n, t])$$

  $\Rightarrow$ **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** : noro_pd.colon(I, F, V),
  noro_pd.ideal_colon(I, J, V)
  - R **のイデアル** I, J **に対し** $I : J = \{f \mid fJ \subset I\}$.
  - $J = \langle g_1, \ldots, g_l \rangle$ **なら** $I : J = \bigcap_{i=1}^{l} I : \langle g_i \rangle$.
  - $I : \langle g \rangle$ **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**.

  $\Rightarrow I : J$ **は共通部分計算により計算できる**.

saturation, radical メンバーシップ

- saturation : `noro_pd.sat`(I, F, V),
  `noro_pd.ideal_sat`(I, J, V)
  - R **のイデアル** I, J **に対し** $I : J^{\infty} = \bigcup_{m=1}^{\infty}(I : J^m)$.
  - $J = \langle g_1, \ldots, g_l \rangle$ **なら** $I : J = \bigcap_{i=1}^{l}(I : \langle g_i \rangle^{\infty})$.
  - $I : \langle g \rangle^{\infty}$ **を** $I : g^{\infty}$ **と書く**. $I : g^{\infty} = (I + \langle tg - 1 \rangle) \cap R$.

  $\Rightarrow$ saturation **は共通部分計算により計算できる**.
- $f \in \sqrt{I}$ **の判定** : `noro_pd.radical_membership`(F, I, V)
  - R **のイデアル** I, $f \in R$ **に対し**,
    $f \in \sqrt{I} \Leftrightarrow I + \langle tf - 1 \rangle = K[x_1, \ldots, x_n, t]$.
  - $I + \langle tf - 1 \rangle = K[x_1, \ldots, x_n, t]$ **は** $I + \langle tf - 1 \rangle$ **の簡約グレブナー基底が** $\{1\}$ **であることと同値**

  $\Rightarrow f \in \sqrt{I}$ **か否かはグレブナー基底で判定できる**.

イデアルの共通部分の計算

イデアルの共通部分

```
[1640] B=[g*a-f*b,h*b-g*c,i*c-h*d,j*d-i*e,l*f-k*g,
      m*g-l*h,n*h-m*i,o*i-n*j]$
[1708] V=[a,b,c,d,e,f,g,h,i,j,k,l,m,n,o]$
[1709] G=nd_gr(B,V,0,0)$
[1710] PD=noro_pd.syci_dec(B,V)$
[1711] length(PD);
4
[1712] map(length,PD);
[10,5,3,1]
[1713] for(I=0,T=[1];I<4;I++)
  for(J=0,L=length(PD[I]);J<L;J++)
    T=noro_pd.ideal_intersection(T,PD[I][J][0],V,0);
[1649] gb_comp(T,G);
1
```

radical メンバーシップ判定

noro_pd.radical_membership は, $f \in \sqrt{I}$ のとき 0 を返すことに注意する.

radical メンバーシップ

```
[1707]  B=[g*a-f*b,h*b-g*c,i*c-h*d,j*d-i*e,l*f-k*g,
        m*g-l*h,n*h-m*i,o*i-n*j]$
[1708] V=[a,b,c,d,e,f,g,h,i,j,k,l,m,n,o]$
[1709] PD=noro_pd.prime_dec(B,V|radical=1)$
[1710] Rad=PD[1]$
[1711] G=nd_gr(B,V,0,0)$
[1712] p_nf(Rad[0],G,V,0);
-o*i*c*b*a+l*f*e*d*c
[1713] p_nf(Rad[0]^2,G,V,0);
0
[1714] noro_pd.radical_membership(Rad[0],B,V);
0
```

実習 9

上の例で, PD[0] には $\sqrt{\langle B \rangle}$ の素イデアル分解のリストが入っている. これらの共通部分を計算して, それが $\langle Rad \rangle$ に等しいことを確かめよ.

Macaulay2

- **起動方法**
  Math **メニューから起動すると**, Emacs **内で起動する**.
- **ヘルプ**, **マニュアル**
  viewHelp **を実行する**.
- **ファイルのロード**
  ユーザファイルは `load`, **パッケージは** `loadPackage` **でロードする**.
- **計算の途中での中断**
  `Control-c` **で計算中断できる**. **トップレベルに戻れる場合もあるが**, **戻れない場合もある**.

基礎環の宣言と多項式の入力

- **有理数体は** QQ, **有限体** $\mathbf{F}_p = \mathbf{Z}/p\mathbf{Z}$ **は** ZZ/p
- **異なる環に属する多項式を使う場合** map **で写す**.

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
o2 = x^2  + 2x*y + y^2  + 2x*z + 2y*z + z^2
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
o7 = x^2  + 2x*y + y^2  + 2x*z + 2y*z + z^2 + u
```

実習 10

適当な環を定義し, 適当な多項式を入力して, 多項式計算ができることを確かめよ.

項順序

- **全次数逆辞書式順序**
  指定がない場合この順序になる.
- **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**: `leadMonomial`
- **先頭項の係数**: `leadCoefficient`
- **係数つきの先頭項**: `leadTerm`

グレブナー基底の計算

- **イデアルの生成** : ideal($p_1, \ldots, p_l$)
- **グレブナー基底計算** : gb (**結果:グレブナー基底オブジェクト**)
- **生成系** : gens **により行列 (行ベクトル) として取り出せる**.
  i **番目の生成元は** g_i. i **は** 0 **から始まる**.
- gbTrace=n (n **は** 0,1,2,3) **により**, **計算経過が表示される**.

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,...
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 |
```

実習 11

イデアル $\langle (x-y)^3 - z^2, (z-x)^3 - y^2, (y-z)^3 - x^2 \rangle \subset \mathbf{Q}[x,y,z]$ **の全次数逆辞書式順序, 純辞書式順序に関するグレブナー基底を計算せよ.**

イニシャルイデアルの計算

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    ...
     ------------------------------------------------------------ ...
              1       52
o5 : Matrix S  <--- S
```

実習 12

上の例において, 実際にグレブナー基底を計算して, グレブナー基底の各多項式の先頭項が上の例のイニシャルイデアルの生成系になっていることを確かめよ.

商および剰余の計算

- remainder(f, g)
  f を g で割った剰余 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.

- quotient(f, g)
  f を g で割った商 q を返す.
  g がグレブナー基底の場合, 商は 0.
  g が行列の場合, $gq + r = f$ を満たす q, r が計算される.

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

- イデアル I, J に対する $I \subset J$ のテスト
  J の任意項順序でのグレブナー基底 G を計算しておき, 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.

商と剰余の計算

Macaulay2 による商と剰余の計算

```
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
```

消去法

selectInSubring(i, m) は, 行列 m から i 番目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.
次の例では $I \cap \mathbf{Q}[z]$ の生成系を計算している.

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 |
```

実習 13

$I = \langle b - a^2, c - a^3, d, e - 1, x - a + 2a(y - b) + 3a^2(z - c), z - f, (x - a)^2 + (y - b)^2 + (z - c)^2 - (x - d)^2 - (y - e)^2 - (z - f)^2 \rangle \subset \mathbf{Q}[a, b, c, d, e, f, x, y, z]$ **に対し** $I \cap \mathbf{Q}[x, y, z]$ **を求めよ. 直接純辞書式順序で計算, 斉次化して計算, ブロック順序で計算など, さまざまな方法を試し, ブロック順序の有効性を確認せよ. (実習 6 と同じである. 斉次化はあらかじめ変数を追加した環で,** `homogenize` **を呼ぶ. 手で行ってもよい.)**

イデアル商, saturation

```
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);
o3 : Ideal of R
i4 : I2=quotient(I1,x);
o4 : Ideal of R
i5 : I1==I2
o5 = false
i6 : I3=quotient(I2,x);
o6 : Ideal of R
i7 : I2==I3
o7 = false
```

```
i8 : I4=quotient(I3,x);
o8 : Ideal of R
i9 : I3==I4
o9 = true
i10 : J=saturate(I,x);
o10 : Ideal of R
i11 : I3==J;
o11 = true
```

radical メンバーシップ判定

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 |
```

実習 14

イデアル

$I = \langle (x-y)^3 - z^2, (z-x)^3 - y^2, (y-z)^3 - x^2 \rangle \subset \mathbf{Q}[x, y, z]$,

$J = \langle x, y, z \rangle$ **に対し**, $I : J^\infty$ **を計算せよ**.

$I = (I : J^\infty) \cap \langle x^2, y^2, z^2 \rangle$ **を確かめよ**.