

noromodule.syz

noromodule.syz User's Manual
Edition 2.0
Sep 2020

by Masayuki Noro

1 noro_module_syz.rr

このマニュアルでは, asir-contrib パッケージに収録されている, 加群の syzygy および自由分解を計算するパッケージ‘noro_module_syz.rr’について解説する. このパッケージを使うには, まず‘noro_module_syz.rr’をロードする.

```
[...] load("noro_module_syz.rr");
```

このパッケージの関数を呼び出すには, 全てnewsyz. を先頭につける.

1.1 多項式環上の加群

多項式環上の自由加群の元は, 加群単項式 te_i の線型和として内部表現される. ここで t は多項式環の単項式, e_i は自由加群の標準基底である. 加群単項式は, 多項式環の単項式に位置 i を追加した $\langle\langle a, b, \dots, c : i \rangle\rangle$ で表す. 加群多項式, すなわち加群単項式の線型和は, 設定されている加群項順序にしたがって降順に整列される. 加群項順序には以下の 3 種類がある.

TOP 順序

これは, $te_i > se_j$ となるのは $t > s$ または $(t=s \text{ かつ } i < j)$ となるような項順序である. ここで, t, s の比較は多項式環に設定されている順序で行う. この型の順序は, `dp_ord([0,Ord])` により設定する. ここで, `Ord` は多項式環の順序型である.

POT 順序

これは, $te_i > se_j$ となるのは $i < j$ または $(i=j \text{ かつ } t > s)$ となるような項順序である. ここで, t, s の比較は多項式環に設定されている順序で行う. この型の順序は, `dp_ord([1,Ord])` により設定する. ここで, `Ord` は多項式環の順序型である.

Schreyer 型順序

各標準基底 e_i に対し, 別の自由加群の加群単項式 T_i が与えられていて, $te_i > se_j$ となるのは $tT_i > sT_j$ または $(tT_i = sT_j \text{ かつ } i < j)$ となるような項順序である. ここで tT_i, sT_j の比較は, これらが所属する自由加群に設定されている順序で行う. この型の順序は, 通常再帰的に設定される. すなわち, T_i が所属する自由加群の順序も Schreyer 型であるか, またはボトムとなる TOP, POT などの項順序となる. この型の順序は `dpm_set_schreyer([H_1, H_2, ...])` により指定する. ここで, $H_i = [T_1, T_2, \dots]$ は加群単項式のリストで, $[H_2, \dots]$ で定義される Schreyer 型項順序を tT_i らに適用するという意味である.

加群多項式を入力する方法としては, $\langle\langle a, b, \dots : i \rangle\rangle$ なる形式で直接入力する他に, 多項式リストを作り, `dpm_ltod()` により変換する方法もある.

1.2 加群の syzygy

1.2.1 newsyz.module_syz

```
newsyz.module_syz(f, v, h, O[|weyl=1, dp=1, f4=1])
```

syzygy の生成系(グレブナー基底)を計算する.

return 要素が 3 つのリスト, 各要素は多項式リストまたは加群多項式のリスト

f 多項式リスト, または多項式リストのリスト

v 変数リスト
 h 非負整数
 O 項順序

- 多項式多項式列または多項式ベクトル列に対する syzygy 加群のグレブナー基底を計算する.
- $f=[f_1,\dots,f_m]$ に対し, $h_1*f_1+\dots+h_m*f_m=0$ を満たす多項式ベクトル (h_1,\dots,h_m) 全体のなす加群のグレブナー基底を計算する.
- f_i が多項式リストの場合, 自然に多項式ベクトルと見なす.
- 返される結果は $[S,G,C]$ の形のリストである. ここで S は, 与えられた項順序 O に対し, 加群の項順序 $[1,O]$ すなわち O で定まる POT (position over term) 項順序でのグレブナー基底である. G は, 入力加群の POT 項順序でのグレブナー基底である. C は, 入力生成系から G の各元を生成する係数リストのリストである.
- h が 0 のとき有理数体上で trace アルゴリズムにより計算する. h が 1 のとき有理数体上で斉次化 trace アルゴリズムにより計算する. h が 2 以上の素数のとき有限体上で計算する.
- オプション f_4 が 1 のとき, F4 アルゴリズムにより計算する.
- オプション $weyl$ が 1 のとき Weyl 代数上で, 左イデアル(左加群)として計算する.
- オプション dpl が 1 のとき, 返される結果は, 加群の要素を表す多項式リストの代わりに加群単項式が用いられる.

```
[0] load("noromodule_syz.rr")$
[43] load("cyclic")$
[53] F=cyclic(4);
[c3*c2*c1*c0-1,((c2+c3)*c1+c3*c2)*c0+c3*c2*c1,(c1+c3)*c0+c2*c1+c3*c2,
c0+c1+c2+c3]
[54] V=[c0,c1,c2,c3]$
[55] L=newsyz.module_syz(F,V,0,0)$
[56] L[0];
[[-c2^2+c3^2)*c1-c3*c2^2+c3^3,-c3^2*c2^2+1,(c3*c2^3-c3^3*c2)*c1+...],
..., [0,0,c0+c1+c2+c3,(-c1-c3)*c0-c2*c1-c3*c2]]
[57] L[1];
[[-(c2+c3)*c1-c3^4*c2^2-c3*c2+2*c3^2],[-c3^2*c2^3-c3^3*c2^2+c2+c3],
..., [c1^2+2*c3*c1+c3^2],[c0+c1+c2+c3]]
[58] L[2];
[[c2-c3)*c1+c3*c2-2*c3^2,c3^2*c2,(-c3*c2^2+c3^2*c2)*c1-c3*c2^3,...],
..., [0,0,-1,c1+c3],[0,0,0,1]]
[59] C0=L[2][0];
[(c2-c3)*c1+c3*c2-2*c3^2,c3^2*c2,(-c3*c2^2+c3^2*c2)*c1-c3*c2^3,
(c3*c2^3-c3^2*c2^2)*c1+c3^2*c2^3-c3^3*c2^2]
[60] L[1][0][0]-(C0[0]*F[0]+C0[1]*F[1]+C0[2]*F[2]+C0[3]*F[3]);
0
[61] M=newsuz.modules_syz(F,V,0,0dp=1)$
[62] M[0];
[(-1)*<<0,1,2,0:1>>+(-1)*<<0,0,2,1:1>>+(1)*<<0,1,0,2:1>>+...,
..., (1)*<<1,0,0,0:3>>+(1)*<<0,1,0,0:3>>+(1)*<<0,0,1,0:3>>+
...+(-1)*<<0,1,1,0:4>>+(-1)*<<1,0,0,1:4>>+(-1)*<<0,0,1,1:4>>]
```

1.3 加群の自由分解

R を多項式環とし、 F_i を R 上の自由加群、 n_i を F_i のランクとする。本節の関数は、 F_0 の部分加群 I に対し、 F_0/I の自由分解

$$0 \rightarrow F_l \rightarrow \cdots \rightarrow F_0 \rightarrow F_0/I \rightarrow 0$$

を与える関数について解説する。この自由分解において $\phi_i: F_i \rightarrow F_{i-1}$ とする。

1.3.1 newsyz.fres, newsyz.minres

`newsyz.fres(f,v,h,O[|weyl=1])`

`newsyz.minres(f,v,h,O[|weyl=1])`

加群の自由分解を計算する。

return 多項式リストのリストのリスト

f 多項式リスト, または多項式リストのリスト

v 変数リスト

h 非負整数

O 項順序

$f=[f_1, \dots, f_m]$ を部分加群 I の生成系とすると、この関数は、 F_0/I の自由分解を計算する。

- 結果は $[M_1, \dots, M_l]$ なるリストで、 M_i は $[\phi_i(e_1), \dots, \phi_i(e_{n_i})]$ なるベクトルのリストで、 $\text{syz } M_{i-1}$ の、 O 上の POT 順序に関するグレブナー基底である。
- `newsyz.module_syz` を逐次的に実行する。 `newsyz.minres` では、得られた syzygy の生成系のうち、定数を成分に持つものがある限り簡約を行う。
- `newsyz.minres` は、 f が斉次の場合、極小自由分解を得る。
- h, O , オプション `weyl` については `newsyz.module_syz` と同様である。

```
[0] load("noromodule_syz.rr")$
[43] load("katsura")$
[47] F=hkatsura(4)$
[48] V=[t,u0,u1,u2,u3,u4]$
[49] R=newsyz.fres(F,V,0,0)$
[51] map(length,R);
[5,22,28,12,2]
[52] S=newsyz.minres(F,V,0,0)$
[5,10,10,5,1]
```

1.3.2 newsyz.lres, newsyz.sres, newsyz.minsres,

`newsyz.lres(f,v,h,O[|dp=1,top=1])`

`newsyz.sres(f,v,h,O[|dp=1])`

`newsyz.minsres(f,v,h,O[|dp=1])`

加群の自由分解を計算する。

return 多項式リストのリストのリスト

f 多項式リスト, または多項式リストのリスト

v 変数リスト

h 0 または 1

O 項順序

- $f=[f_1,\dots,f_m]$ を部分加群 I の生成系とすると、`newsyz.lres` は F_0/I の自由分解を La Scala-Stillman アルゴリズムにより計算する。`newsyz.sres`, `newsyz.minsres` は F_0/I の自由分解を Schreyer アルゴリズムにより計算する。
- 有理数体上の多項式環上の加群に対してのみ実装されている。
- 結果は $[M_1,\dots,M_l]$ なるリストで、 M_i は $\text{syz } M_{i-1}$ の、Schreyer 順序に関するグレブナー基底である。
- $h=1$ のとき、最初のグレブナー基底計算が斉次化経由で行われる。
- $dp=1$ のとき、結果を加群多項式で返す。
- `newsyz.lres` において $\text{top}=1$ のとき、 S -多項式の剰余計算は、先頭項が簡約できなかった時点で修了する。
- f が斉次の場合、`newsyz.lres` および `newsyz.minsres` は極小自由分解を得る。

```
[0] load("noro_module_syz.rr")$
[43] F=[x00*x11-x01*x10,x01*x12-x02*x11,x02*x13-x03*x12,-x11*x20+x21*x10,
-x21*x12+x22*x11,-x22*x13+x23*x12,x31*x20-x30*x21,x32*x21-x31*x22,x33*x22-x32*x23]$
[44] V=[x00,x01,x02,x03,x10,x11,x12,x13,x20,x21,x22,x23,x30,x31,x32,x33]$
[45] cputime(1)$
1.8e-05sec(1.502e-05sec)
[46] R=newsyz.minres(F,V,0,0)$
333.4sec(339.6sec)
[47] S=newsyz.lres(F,V,0,0)$
85.34sec(85.56sec)
[48] T=newsyz.minsres(F,V,0,0)$
241.2sec(250.3sec)
[49] cputime(0)$
[50] map(length,R);
[9,75,456,1602,3391,4680,4388,2849,1290,393,72,6]
[51] map(length,S);
[9,75,456,1602,3391,4680,4388,2849,1290,393,72,6,0]
[52] map(length,T);
[9,75,456,1602,3391,4680,4388,2849,1290,393,72,6,0]
```

Index

(インデックスがありません)

(インデックスがありません)

簡単な目次

1	noromodule_syz.rr	1
	Index	5

目次

1	noromodule_syz.rr	1
1.1	多項式環上の加群	1
1.2	加群の syzygy	1
1.2.1	newsyz.module_syz	1
1.3	加群の自由分解	3
1.3.1	newsyz.fres, newsyz.minres	3
1.3.2	newsyz.lres, newsyz.sres, newsyz.minsres,	3
	Index	5

