noro_pd

noro_pd User's Manual
Edition 1.0
Feb 2011

by Masayuki Noro

1 準素分解パッケージ noro_pd.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, 準素パッケージ 'noro_pd.rr' について解説する. このパッケージを使うには, まず 'noro_pd.rr' をロードする.

```
[1539] load("noro_pd.rr");
```

このパッケージの函数を呼び出すには, 全て noro_pd. を先頭につける.

1.1 準素分解

1.1.1 noro_pd.syci_dec

noro_pd.syci_dec(*I*,*vars*[|*options*])
: イデアル *I* の最短準素分解を計算する.

return
: *[QL(1),QL(2),...]* なるリスト, 各 *QL(i)* は *[[Q(i1),P(i1)],[(i2),P(i2)],...]* なるリスト.

I
: 多項式リスト

vars
: 変数リスト

options
: 下の説明参照.

- イデアル *I* の最短準素分解を計算する. 'noro_pd.rr' で実装されている準素分解アルゴリズムは SYCI アルゴリズムと呼ばれるもので, 'primdec' に実装されている Shimoyama-Yokoyama (SY アルゴリズム) を改良したものである.

  イデアル *I* の付属素イデアル全体 *Ass(I)* を次のように分ける: *A(1)* を極小付属素イデアル全体とし, *Ass(I)* から *A(1),...,A(i-1)* を除いたものの中で極小なもの全体を *A(i)* とする. SYCI アルゴリズムは, *A(i)* を *i* の小さい順に計算しつつ, 対応する準素成分を計算するアルゴリズムである. 準素成分の計算は省略することができる. この場合, *I* の付属素イデアルのみを計算するアルゴリズムとなる. *A(i)* に属する付属素イデアルおよび対応する準素成分はレベル *i* であるという.

  出力において, *Q(ij)* は *I* の *Pij*-準素成分であり, *Q(ij)* 全体が *I* の最短準素分解を与える. 各 *QL(i)* はレベル *i* の成分全体を与える. 特に *QL(1)* は孤立成分および極小付属素イデアル全体を与える. *QL(1)* の各成分のみ, 3 つ目の要素として極大独立集合を持つ.
- オプション *ass=1* が指定された場合, *QL(1)* に現われる成分 *Q(1j)* のみ準素成分 (孤立準素成分) となるが, *QL(2)* 以降に現われる *Q(ij)* は *I* のある中間分解成分となる. しかし, *P(ij)* は *I* の付属素イデアルであり, *P(ij)* 全体が *I* の付属素イデアル全体の集合を与えることは保証される.
- デフォルトでは有理数体上での分解を計算するが, オプション *mod=p* (*p* は 30 ビット以下の素数) を指定すると *p* 元体上での分解を計算する. ただし, *p* が小さいときには正しく計算できないか, 無限ループに陥る. (*p* が 5 桁以上なら問題ないであろう.)
- *iso=n* (*n* は 0,1,2,3 のいずれか) を指定すると, saturation 計算の方法が変わる. デフォルトでは *n=1* である.
- オプション *time=1* を指定すると, 計算時間の内訳を表示する.

- オプション *para=proclist* を指定すると, 部分的な分散計算を行う. *proclist* は `noro_pd.init_pprocs` により生成されたプロセスのリストである.
- オプション *f4=1* が指定された場合, 可能な限り F4 アルゴリズムを用いる. デフォルトでは Buchberger アルゴリズムを用いる.
- オプション *trace=1* が指定された場合, 可能な限り trace アルゴリズムを用いる. デフォルトでは trace なしの Buchberger または F4 アルゴリズムを用いる.
- オプション *intgb=1* が指定された場合, 複数のイデアルの共通部分を, 2 つずつのイデアルの共通部分計算の繰り返しとして計算する際に, 毎回その共通部分のグレブナー基底を計算する. デフォルトではグレブナー基底を計算せずに繰り返す.

```
[1539] load("noro_pd.rr");
[1707] B=[x00*x11-x01*x10,x01*x12-x02*x11,x02*x13-x03*x12,x03*x14-x04*x13,
-x11*x20+x21*x10,-x21*x12+x22*x11,-x22*x13+x23*x12,-x23*x14+x24*x13]$
[1708] V=[x00,x01,x02,x03,x04,x10,x11,x12,x13,x14,x20,x21,x22,x23,x24]$
[1709] QD=noro_pd.syci_dec(B,V|time=1)$
[total,1.08407,ass,0.620039,pd,0.33202,iso,0.260016,int,0.024003,
ext,0.464029]
[elapsed,1.09038,ass,0.624087,pd,0.338769,iso,0.244057,int,0.0343642,
ext,0.466293]
[1710] map(length,QD);
[10,5,3,1]
[1711] QD[2][0];
[[x03*x01*x14*x20-x21*x04*x03*x10,(x23*x21*x02-x22*x21*x03)*x10,
x23*x21*x03*x10,x01*x12*x20-x21*x02*x10,-x01*x13*x20+x21*x03*x10,
-x21*x03*x14+x23*x04*x11,-x22*x03*x14+x23*x04*x12,x01^2,x03^2,
-x00*x11+x01*x10,x10^2,x01*x11,-x01*x12+x02*x11,x01*x13-x03*x11,
x10*x11,x11^2,-x02*x13+x03*x12,-x11*x12,x12^2,x03*x13,-x03*x14+x04*x13,
x11*x13,-x12*x13,x13^2,x13*x14,x14^2,-x11*x20+x21*x10,x21*x11,
x21*x12-x22*x11,x21*x13-x23*x11,x21^2,x22*x13-x23*x12,-x23*x13,
-x23*x14+x24*x13,x23^2],
[x01,x03,x10,x11,x12,x13,x14,x21,x23]]
```

1.1.2 noro_pd.prime_dec

`noro_pd.syci_dec(`*I*`,`*vars*`[|`*options*`])`
: イデアル *I* の根基の素イデアル分解を計算する.

return : リスト (成分の詳細は下で説明する)

I : 多項式リスト

vars : 変数リスト

options : 下の説明参照.

- デフォルトでは, *I* の根基の素イデアル分解の成分のリスト *[P(1),P(2),...]* を返す.
- オプション *indep=1* が指定された場合, *[[P(1),Y(1)],[P(2),Y(2)],...]* を返す. ここで, *Y(i)* は *P(i)* に対する極大独立集合である.
- オプション *radical=1* が指定された場合, 素イデアル成分のリスト *PL* と, *I* の根基 *rad* のペアのリスト *[PL,rad]* が返される.

```
[1712] PD=noro_pd.prime_dec(B,V|radical=1)$
[1713] PD[0][0];
[x10,-x11,x12,x13,x14]
[1714] PD[1];
[-x03*x02*x01*x14*x20+x24*x02*x01*x00*x13,...,x23*x14-x24*x13]
```

1.2 関連する関数

1.2.1 noro_pd.ideal_intersection

noro_pd.ideal_intersection(*I1*,*I2*,*vars*,*ord*[|*mod*=*p*])
: イデアル *I1*, *I2* の共通部分を計算する.

return : 多項式リスト

I : 多項式リスト

vars : 変数リスト

ord : 項順序

- 返されるリストは *I1* と *I2* の共通部分の *ord* に関するグレブナー基底になっている.
- デフォルトでは有理数体上で計算だが, オプション *mod*=*p* が指定された場合 *p* 元体上での計算を行う.

```
[1707] A=[j*h*g*f*e*d*b,j*i*g*d*c*b,j*i*h*g*d*b,j*i*h*e*b,i*e*c*b,z]$
[1708] B=[a*d-j*c,b*c,d*e-f*g*h]$
[1709] V=[a,b,c,d,e,f,g,h,i,j,z]$
[1710] noro_pd.ideal_intersection(A,B,V,0);
[(j*h*g*f*e^2*d^2-j*h^2*g^2*f^2*e*d)*b,j*h*g*f*e*d*b*a,
-j*h*g*f*e*d*c*b,j*i*h*g*f*e*b*a,(-j*i*h*e*d^2+j*i*h^2*g*f*d)*b,
(-j*i*h*e^2*d+j*i*h^2*g*f*e)*b,-j*i*h*e*d*b*a,-j*i*h*g*d*b*a,
j*i*g*d*c*b,i*e*c*b,-z*e*d+z*h*g*f,-z*c*b,-z*d*a+z*j*c]
```

1.2.2 noro_pd.ideal_intersection_m

noro_pd.ideal_intersection_m(*I1*,*I2*,*vars*,*ord*[|*mod*=*p*])
: イデアル *I1*, *I2* の共通部分を計算する.

return : 多項式リスト

I : 多項式リスト

vars : 変数リスト

ord : 項順序

I2 がグレブナー基底で, *I1* の生成系の個数が多い場合に noro_pd.ideal_intersection より高速な場合がある.

- 返されるリストは共通部分の生成系だが必ずしも共通部分のグレブナー基底になってはいない.
- デフォルトでは有理数体上で計算だが, オプション *mod*=*p* が指定された場合 *p* 元体上での計算を行う.

```
[1754] B=[z*j*i*e*d*c*b+(z*i*h*g+z*j*h)*f*e*d*c,...,z*j*e*c*b+4*z*i*h*g*e*c]
[1755] V=[b,c,d,e,f,g,h,i,j,z]
[1756] G=nd_gr(B,V,0,0)$
[1757] cputime(1)$
0sec(1.907e-06sec)
[1758] I1=noro_pd.ideal_intersection(G,G,V,0)$
0.316sec + gc : 0.012sec(0.3301sec)
[1759] I2=noro_pd.ideal_intersection_m(G,G,V,0)$
0.064sec + gc : 0.012sec(0.07933sec)
```

1.2.3 noro_pd.ideal_list_intersection

noro_pd.ideal_intersection(*ilist*,*vars*,*ord*[|*mod=p*])
: イデアルのリスト *ilist* の成分の共通部分を計算する.

return : 多項式リスト

ilist : 多項式リストのリスト

vars : 変数リスト

- 返されるリストは共通部分のグレブナー基底になっている.
- デフォルトでは有理数体上で計算だが, オプション *mod=p* が指定された場合 p 元体上での計算を行う.

```
[1709] PL=noro_pd.prime_dec(B,V|radical=1)$
[1710] Int=noro_pd.ideal_list_intersection(PL[0],V,0)$
[1711] gb_comp(Int,PL[1]);
1
```

1.2.4 noro_pd.colon

noro_pd.colon(*I*,*f*,*vars*[|*mod=p*])
: *I:f* を計算する.

return : 多項式リスト

I : 多項式リスト

f : 多項式

vars : 変数リスト

- 返されるリストは *I:f* のグレブナー基底とは限らない.
- デフォルトでは有理数体上で計算だが, オプション *mod=p* が指定された場合 p 元体上での計算を行う.

```
[1640] B=[(x+y+z)^50,(x-y+z)^50]$
[1641] V=[x,y,z]$
[1642] noro_pd.colon(B,y^98,V);
[-x-z,-y]
```

1.2.5 noro_pd.ideal_colon

noro_pd.colon(*I*,*J*,*vars*[|*mod*=*p*])
: *I*:*J* を計算する.

return 多項式リスト

I 多項式リスト

J 多項式リスト

vars 変数リスト

- 返されるリストは *I*:*J* のグレブナー基底になっている.
- デフォルトでは有理数体上で計算だが, オプション *mod*=*p* が指定された場合 *p* 元体上での計算を行う.

```
[1640] B=[(x+y+z)^50,(x-y+z)^50]$
[1641] V=[x,y,z]$
[1642] noro_pd.ideal_colon(B,[(x+y+z)^49,(x-y+z)^49],V);
[-y^49*x-z*y^49,-y^50,-x^2-2*z*x+y^2-z^2]
```

1.2.6 noro_pd.sat

noro_pd.sat(*I*,*f*,*vars*[|*mod*=*p*])
: *I* の *f* による saturation を計算する.

return 多項式リスト

I 多項式リスト

f 多項式

vars 変数リスト

- 返されるリストは *I*:*f* のグレブナー基底になっている.
- デフォルトでは有理数体上で計算だが, オプション *mod*=*p* が指定された場合 *p* 元体上での計算を行う.

```
[1640] B=[(x+y+z)^50,(x-y+z)^50]$
[1641] V=[x,y,z]$
[1642] noro_pd.sat(B,y,V);
[1]
```

1.2.7 noro_pd.ideal_sat

noro_pd.ideal_sat(*I*,*J*,*vars*[|*mod*=*p*])
: *I*:*J* を計算する.

return 多項式リスト

I 多項式リスト

J 多項式リスト

vars 変数リスト

- 返されるリストは *I:J* のグレブナー基底になっている.
- デフォルトでは有理数体上で計算だが, オプション *mod=p* が指定された場合 *p* 元体上での計算を行う.

```
[1640] B=[(x+y+z)^50,(x-y+z)^50]$
[1641] V=[x,y,z]$
[1642] noro_pd.ideal_sat(B,[(x+y+z)^49,(x-y+z)^49],V);
[1]
```

1.2.8 noro_pd.init_pprocs

noro_pd.init_pprocs(*m*[|*nox=1*])
: 分散計算用プロセスを起動する.

return : 整数リスト

m : 正整数

- 分散計算に用いるためのプロセス (ox_asir) を起動し, その番号のリストを返す.
- ホームディレクトリの '.asirrc' に load("noro_pd.rr")$ を入れておくことで, ox_asir の起動時に 'noro_pd.rr' が読み込まれ, 分散計算の準備ができる.
- オプション *nox=1* が指定された場合, 起動されたプロセスからの画面出力のためのウィンドウが開かない.

```
[1544] P=noro_pd.init_pprocs(6|nox=1)$
[1545] B=[x00*x11-x01*x10,x01*x12-x02*x11,x02*x13-x03*x12,x03*x14-x04*x13,
x04*x15-x05*x14,x05*x16-x06*x15,x06*x17-x07*x16,-x11*x20+x21*x10,
-x21*x12+x22*x11,-x22*x13+x23*x12,-x23*x14+x24*x13,-x24*x15+x25*x14,
-x25*x16+x26*x15,-x26*x17+x27*x16]$
[1546] V=[x00,x01,x02,x03,x04,x05,x06,x07,x10,x11,x12,x13,x14,x15,x16,
x17,x20,x21,x22,x23,x24,x25,x26,x27]$
[1547] noro_pd.syci_dec(B,V|time=1)$
[total,205.581,ass,108.743,pd,31.582,iso,64.9081,int,11.7367,ext,96.8381]
[elapsed,206.177,ass,109.052,pd,31.679,iso,65.0682,int,11.7853,ext,97.1254]
[1548] noro_pd.syci_dec(B,V|time=1,para=P)$
[total,30.0339,ass,29.5498,pd,23.7695,iso,1.96412,int,3.32021,ext,0.48403]
[elapsed,79.0897,ass,62.5683,pd,26.0532,iso,28.037,int,7.97536,ext,16.5214]
```

Index

(Index is nonexistent)

(Index is nonexistent)

Short Contents

Table of Contents