Risa/Asir 代数曲線論用パッケージ説明書

利用説明書
1.0 版
2004 年 8 月

by Shuhei Todo

1 関数簡易マニュアル

1.1 概要

このパッケージには、代数曲線の諸性質を調べるための関数が集められている。主な機能は、代数曲線に対して定義される以下の対象を計算できることである：

- 2 曲線の交点の座標
- 特異点の座標
- neighborhood graph（二次変換によって特異点がどのように分解されるかを表す tree）
- 既約曲線の種数
- 随伴曲線 (adjoint curves)
- 二次曲線上の有理点
- 有理曲線（種数 0 の曲線）をパラメトライズする有理関数

その他、多項式の全次数を計算するといったような予備的な関数群が用意されている。ユーザーの入力する代数曲線の定義多項式は必ず有理数体上の変数 x,y,z の**斉次**多項式でなければならない。

1.2 Notation

本書で用いられる記号について、次のような約束をしておく。

- 点 `[x,y,z]` とは射影平面の点の斉次座標 $(x:y:z)$ を意味し、特に断りがなければ、$z=0$ でないときは必ず $z=1$ となるように正規化されている。
- Q は有理数体、$\overline{Q}$ は代数的数全体のなす体を意味する。

1.3 主な関数

1.3.1 `intersect`

`intersect(`*F*`,`*G*`)`
:: 2 曲線 $F=0,G=0$ の交点の座標からなるリストを返す.

return リスト

F G 変数 x,y,z の斉次多項式

- 2 曲線 $F=0,G=0$ の交点 `[x,y,z]`からなるリストを返す。
- F,G は共通因子を持っていてはいけない。

```
[1] intersect(y^2-x*z,(x^2+y^2)^3-4*x^2*y^2*z^2);
[[0,0,1],[(#4),(#5),1]]
[2] defpoly(alg(4));
t#4^3+3*t#4^2+3*t#4-3
[3] defpoly(alg(5));
t#5^2-t#4
[4] intersect(x^2-y^2,x^3+y*x^2+(y^2-z^2)*x+y^3-z^2*y);
***two curve have common components***
```

1.3.2 sing

`sing(F)` :: 曲線 *F=0* の特異点の座標からなるリストを返す.

return リスト

F 変数 x,y,z の斉次多項式

- 曲線 *F=0* の特異点 `[x,y,z]`（$F_x(x,y,z) = F_y(x,y,z) = F_z(x,y,z) = 0$ を満たす点）からなるリストを返す。
- *F* は重複因子を持っていてはいけない（定義より重複因子の零点はすべて特異点である)。

```
[1] sing(16*x^6-24*z^2*x^4+9*z^4*x^2+4*z^2*y^4-4*z^4*y^2);
[[0,0,1],[(#4),0,1],[1/2,(#3),1],[-1/2,(#3),1],[0,1,0]]
[2] defpoly(alg(3));
2*t#3^2-1
[3] defpoly(alg(4));
4*t#4^2-3
[4] sing((x-y)*(y^2-x*z));
[[1,1,1],[0,0,1]]
[5] sing((x-y)^2*(y^2-x*z));
***Argument has multiple divisor***
```

参照 Section 1.3.3 [nbh], page 2 Section 1.4.4 [multia], page 6

1.3.3 nbh

`nbh(F)` :: 曲線 *F*=0 の neighborhood graph を返す。

return リスト

F 変数 x,y,z の斉次多項式

- 曲線 *F=0* の neigborhood graph を表すリストを返す。neighborhood graph とは二次変換によって特異点がどのように分解されるかを表す tree である。分解によって現れる点のことを隣接点と呼ぶ。特異点、隣接点の情報は、それぞれ次のような**ベクトル**によって表される。

  特異点 [点の個数, 点の座標, [重複度, 通常特異点 (=1) かそうでない (=-1) か], [この（これらの）特異点から出てくる隣接点の情報 (これ以上隣接点が現れない場合は"terminal")]]

  隣接点 [点の個数, [重複度, 通常特異点 (=1) かそうでないか (=-1) か], [この（これらの）隣接点から出てくる隣接点の情報 (これ以上隣接点が現れない場合は"terminal")]]

  一般に、特異点の座標は代数的数になる。この場合、代数的数を共役な代数的数で置き換えて得られる点もまた、特異点になる。この性質を利用して複数の特異点を一度に表示するのであるが、特異点ベクトルの最初の引数「点の個数」はこのような表示によって、いくつの特異点が表されているかを示している。したがって、特異点が有理点ならば、点の個数`=1` である。隣接点ベクトルの最初の引数である「点の個数」は親ベクトルの表す各点から、この数だけ同じタイプの隣接点が出てくることを意味する。
- neighborhood graph はこれらのベクトルを入れ子にしたリストによって表現されている。

```
[1] F=x^6+3*y^2*x^4+(3*y^4-4*z^2*y^2)*x^2+y^6;
x^6+3*y^2*x^4+(3*y^4-4*z^2*y^2)*x^2+y^6
[2] sing(F);
[[0,0,1],[(#0),1,0]]
[3] nbh(F);
[ 1 [0,0,1] [4,-1] [[ 1 [2,1] [terminal] ],[ 1 [2,1] [terminal] ]] ]
[ 2 [(#0),1,0] [2,-1] [[ 1 [1,1] [terminal] ]] ]
```

特異点 `[0,0,1]` は重複度 4 の通常でない特異点であり、2 つの隣接点をもつ。それらはどちらとも重複度 2 の通常特異点である。特異点 `[(#0),1,0]`の隣接点は単純点である。

- F は重複因子を持っていてはいけない。

参照　Section 1.3.2 [sing], page 2

1.3.4 genus

`genus(`*F*`)` :: 曲線 *F=0* の特異点の座標からなるリストを返す.

return　0 以上の整数

F　変数 x,y,z の斉次多項式

- 曲線 *F=0* の種数を返す。
- F は $\overline{Q}[x,y,z]$ において既約でなければならない。この条件の下でしか正確な値が返される保証がない。Q[x,y,z] において既約であったとしても、$\overline{Q}[x,y,z]$ で既約とは限らないので注意を要する。入力がこの条件を満たしているかどうかはチェックされない。

```
[1] genus(x^6+3*y^2*x^4+(3*y^4-4*z^2*y^2)*x^2+y^6);
0
[2] genus(y^2*z-x^3-z^3);
1
[3] genus(x^2+y^2+z^2-x*y-y*z-z*x);
-1
[4] fctr(x^2+y^2+z^2-x*y-y*z-z*x);
[[1,1],[x^2+(-y-z)*x+y^2-z*y+z^2,1]]
[5] irr_conic(x^2+y^2+z^2-x*y-y*z-z*x);
reducible
```

参照　Section 1.4.5 [irr_conic], page 7

1.3.5 adjoint1,adjoint2

`adjoint1(`*F*`)`
`adjoint2(`*F*`)`
:: それぞれ曲線 *F=0* の n-1 次,n-2 次の随伴曲線 (adjoint curve) を返す (n=deg(F))。

return　線形のパラメーターを含む変数 x,y,z の斉次多項式

F　変数 x,y,z の斉次多項式

- n-2 次の曲線 *G=0* が曲線 *F=0* の重複度 r の点を少なくとも重複度 r-1 にもつとき、曲線 *G=0* を曲線 *F=0* の n-2 次の随伴曲線 (adjoint curve) と呼ぶ。n-1 個の随伴曲線 $G_0 = 0, G_1 = 0, \ldots, G_{n-2} = 0$ が存在して、n-2 次の随伴曲線の定義多項式全体は $c_0G_0 + c_1G_1 + \ldots + c_{n-2}G_{n-2}$ (c_i は係数体の元) と表される。`adjoint2`(F) は、この n-1 個の線形のパラメーターを含んだ斉次多項式を返す。n-1 次の随伴曲線も同様に定義される。n-1 次の随伴曲線の定義多項式全体も上と同様に、2n-1 個の線形パラメーターを含んだ n-1 次の斉次多項式で表される。`adjoint1`(F) はこの多項式を返す。
- 最初にパラメーターのリストと、その長さが表示される。
- F は重複因子を持っていてはいけない。

```
[1] adjoint2(x^6+3*y^2*x^4+(3*y^4-4*z^2*y^2)*x^2+y^6);
[c2,c3,c4,c6,c7] 5
(c2-c4)*x^4+c3*y*x^3+(c2*y^2+c6*z*y)*x^2+(c3*y^3+c7*z*y^2)*x+c4*y^4
[2] adjoint1(F);
[c1,c7,c11,c12,c13,c15,c16,c17,c18,c19,c20] 11
(c1*y+(c11-c15+c18-c20)*z)*x^4+(c13*y^2+c7*z*y+c11*z^2)*x^3+(c17*z*y^2+c12*z^2*y
+c15*z^3)*x^2+(c13*z^2*y^2+c16*z^3*y+c18*z^4)*x+c17*z^3*y^2+c19*z^4*y+c20*z^5
```

参照　Section 1.4.7 [restriction], page 7

1.3.6 intpt

`intpt(`*F*`)` :: 二次曲線 *F=0* 上の整数点 `[x,y,z]` をひとつ見つけて返す。整数点が存在しなければ、文字列`no integer solution`を返す。

return　リスト、あるいは文字列`no integer solution`.

F　変数 x,y,z の二次の斉次多項式

- 二次曲線 *F=0* 上に整数点 (affine でいう有理点) があれば、その座標 `[x,y,z]`を返す。`x,y,z` はすべて整数である。整数点が存在しないときは文字列`no integer solution` を返す。
- 三元二次形式の整数解を求める古典的な Legendre の方法を用いている。サブルーチンで二次の合同方程式を解く際、単に総当り法を用いているだけので、F の係数が大きくなると非常に時間がかかる。

```
[1] intpt(22*x^2-10*y^2+z^2+5*x*y+13*y*x-z*x);
[71,-121,473]
[2] intpt(22*x^2-10*y^2+z^2+5*x*y+12*y*x-z*x);
no integer solution
```

1.3.7 parametrize

`parametrize(`*F*`)`
:: 有理曲線 *F=0* をパラメトライズする多項式の組を返す。

return　リスト

F　有理曲線の定義多項式（変数 x,y,z の斉次多項式）

- 有理曲線 *F=0*（種数が 0 の曲線）は、変数 t の多項式 P(t),Q(t),R(t) および x,y,z の斉次多項式 S(x,y,z),T(x,y,z) を用いて (x:y:z)=(P(t):Q(t):R(t)), t=T(x,y,z)/S(x,y,z) とパラメーター表示される。`parametrize`(F) はこれらの多項式からなるリスト

[P(t),Q(t),R(t),T(x,y,z)/S(x,y,z)] を返す（GCD(P(t),Q(t),R(t))=1 である）。一般には P(t),Q(t),R(t) は係数に有理数の平方根を含む多項式となるが、有理数係数の多項式で曲線をパラメトライズできる場合は、**常に**有理数係数の多項式の組を返す（例えば曲線の次数が奇数の場合）。

- F は $\overline{Q}[x,y,z]$ において既約で、かつ種数が 0 でなければならないが、これらの条件が満たされているかどうかのチェックはなされない。

```
[1] parametrize(x^4+(2*y^2-z^2)*x^2+y^4+z^2*y^2);
[-t^3-t,t^3-t,t^4+1,(-x^2-y^2)/(z*x+z*y)]
[2] parametrize((x^2+y^2)^3-4*x^2*y^2*z^2);
heuristic2 failed...
heuristic3 succeed
[32256*t^6-133120*t^5-129024*t^4+1064960*t^3-516096*t^2
-2129920*t+2064384,-127008*t^6+1048320*t^5-2671232*t^4
+10684928*t^2-16773120*t+8128512,274625*t^6-3194100*t^5
+15678780*t^4-41555808*t^3+62715120*t^2-51105600*t+17576000,
(-126*x^4+1040*y*x^3-382*y^2*x^2+1040*y^3*x-256*y^4)
/(-65*x^4+520*y*x^3+(-65*y^2-32*z*y)*x^2+(520*y^3+256*z*y^2)*x)]
[3] parametrize(22*x^2-10*y^2+z^2+5*x*y+12*y*x-z*x);
[(220*#6-10)*t^2+(-22*#6+1),(374*#6-17)*t^2+(-22*#6-43)*t,
(220*#6+210)*t^2+(-374*#6+17)*t+22,(-y)/((22*#6-1)*x+z)]
```

参照 Section 1.3.4 [genus], page 3

1.4 その他の関数

1.4.1 tdeg

`tdeg(Poly)`

:: 多項式 *Poly* の全次数を返す。

return 0 以上の整数

Poly 多項式

- 多項式 *Poly* の全次数を返す。

```
[1] tdeg(u^3+v^3-x*y*z*w);
4
[956] tdeg((x^3+y^2+z)*(a^2+b+1));
5
```

1.4.2 homzation

`homzation(AF)`

:: 変数 x,y の多項式を斉次化して x,y,z の斉次多項式にする。

return 変数 x,y,z の斉次多項式

F 変数 x,y の多項式

- 変数 x,y の多項式を斉次化して x,y,z の斉次多項式にする。入力する多項式の変数は x,y でなければならない。

```
[1] homzation((x^2+4*x^3+6*x^4)-4*x^4*y
+(-2*x-4*x^2-2*x^3)*y^2+y^4);
(-4*y+6*z)*x^4+(-2*y^2+4*z^2)*x^3
+(-4*z*y^2+z^3)*x^2-2*z^2*y^2*x+z*y^4
[958] homzation(u*v+1);
Input must be polynomial of variable x,y
```

1.4.3 random_line

random_line(*Pt*,B[,*Seed*])

:: 点 *Pt*(=[x,y,z]) を通る直線をひとつランダムに返す。

return 変数 x,y,z の一次式

Pt 点を表すリスト

B 自然数

Seed 自然数

- 点 *Pt*(=[x,y,z]) を通る直線の方程式で各係数の値が-B 以上 B 未満のものを、ひとつランダムに返す。
- Seed はサブルーチンで random([Seed]) を用いる際に使用される。

```
[1] random_line([0,0,1],1);
x-8*y
```

1.4.4 multia

multia(F,Pt)

:: 曲線 *F=0* の点 *Pt*(=[x,y,z]) における重複度を返す。

return 0 以上の自然数

F 変数 x,y,z の斉次多項式

Pt 点を表すリスト

- 曲線 *F=0* の点 *Pt*(=[x,y,z]) における重複度を返す。F を N 階偏微分して得られる多項式が初めて点 Pt で 0 にならないとき、整数 N を曲線 *F=0* の点 Pt における重複度という。

```
[1] multia((4*y^2+4*z^2)*x^4+8*z^3*x^3+8*z^2*y^2*x^2-8*z^5*x+
4*z^4*y^2-4*z^6,[0,0,1]);
0
[2] multia((4*y^2+4*z^2)*x^4+8*z^3*x^3+8*z^2*y^2*x^2-8*z^5*x+
4*z^4*y^2-4*z^6,[0,1,0]);
4
[3] multia((4*y^2+4*z^2)*x^4+8*z^3*x^3+8*z^2*y^2*x^2-8*z^5*x+
4*z^4*y^2-4*z^6,[1,0,0]);
2
```

参照 Section 1.3.2 [sing], page 2 Section 1.3.3 [nbh], page 2

1.4.5 irr_conic

`irr_conic(F)`
: $::$ 三元二次形式 F が $\overline{Q}[x,y,z]$ で既約かどうかを判定する。

return 文字列

F 変数 x,y,z の二次の斉次多項式

- 三元二次形式 F が $\overline{Q}[x,y,z]$ で既約ならば `irreducible`を、可約ならば `reducible` を返す。

```
[1] irr_conic(x^2+y^2+z^2-x*y-y*z-z*x);
reducible
[2] fctr(x^2+y^2+z^2-x*y-y*z-z*x);
[[1,1],[x^2+(-y-z)*x+y^2-z*y+z^2,1]]
```

1.4.6 lissajou

`lissajou(M,N)`
: $::$ $x=\sin(M\theta), y=\cos(N\theta)$ によって定義されるリサージュ曲線の陰関数表示

return 変数 x,y,z の斉次多項式

M N 互いに素な自然数

- $x=\sin(M\theta), y=\cos(N\theta)$ によって定義されるリサージュ曲線の陰関数表示（変数 x,y,z の斉次多項式）を返す。

```
[984] lissajou(3,4);
64*x^8-128*z^2*x^6+80*z^4*x^4-16*z^6*x^2+16*z^2*y^6
-24*z^4*y^4+9*z^6*y^2
[985] lissajou(2,7);
4096*x^14-14336*z^2*x^12+19712*z^4*x^10-13440*z^6*x^8
+4704*z^8*x^6-784*z^10*x^4+49*z^12*x^2+4*z^10*y^4-4*z^12*y^2
```

1.4.7 restriction

`restriction(A,List)`
: $::$ 特定の点を通る随伴曲線の定義多項式を計算したいときに用いる。

return 線形のパラメーターを含む x,y,z の斉次多項式

A `adjoint1`,`adjoint2`から返される形と同様の、線形パラメーターつきの変数 x,y,z の斉次多項式

List 点 `[x,y,z]`からなるリスト

- `adjoint1`,`adjoint2`から返される線形パラメーター付の斉次多項式が、*List* に含まれる各点を零点にもつためには、線形パラメーターの間にいくつかの（Q 上の）一次関係式が成り立てばよい。この条件を加味して、新たな線形パラメーター付の斉次多項式を作る。
- *List* に含まれる点は、`intersect`や `sing` から返される点を使うことを想定している。

参照 ⟨undefined⟩ [adjoint1], page ⟨undefined⟩

Index

(Index is nonexistent)

(Index is nonexistent)

Short Contents

1 関数簡易マニュアル 1
Index 8

Table of Contents

1 関数簡易マニュアル 1

1.1 概要 1
1.2 Notation 1
1.3 主な関数 1
1.3.1 `intersect` 1
1.3.2 `sing` 1
1.3.3 `nbh` 2
1.3.4 `genus` 3
1.3.5 `adjoint1`,`adjoint2` 3
1.3.6 `intpt` 4
1.3.7 `parametrize` 4
1.4 その他の関数 5
1.4.1 `tdeg` 5
1.4.2 `homzation` 5
1.4.3 `random_line` 6
1.4.4 `multia` 6
1.4.5 `irr_conic` 6
1.4.6 `lissajou` 7
1.4.7 `restriction` 7

Index 8