

# nn\_ndbf

---

nn\_ndbf User's Manual  
Edition 1.0  
Nov 2009

by Masayuki Noro and Kenta Nishiyama

---



# 1 New b-function package nn\_ndbf.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, 新 b 関数パッケージ 'nn\_ndbf.rr' について解説する. このパッケージを使うには, まず 'nn\_ndbf.rr' をロードする.

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

このパッケージの関数を呼び出すには, 全て ndbf. を先頭につける. このマニュアルでは, 関連する組込み関数についても解説する.

## 1.1 b 関数計算

### 1.1.1 ndbf.bfunction

`ndbf.bfunction(f[|weight=w,heruristic=yesno,vord=v,op=yesno])` :: 多項式  $f$  の大域 b 関数を計算する.

*return*      多項式

$f$             多項式

$w$              $[v1,w1,...,vn,wn]$  なるリスト

*yesno*       0 または 1

$v$             変数のリスト

- この関数は asir-contrib のパッケージ 'nn\_ndbf.rr' で定義されている.
- 多項式  $f$  の大域 b 関数 (global b-function) を計算する. デフォルトでは大域 b 関数のみが出力されるが, オプション `op=1` が指定された場合, 大域 b 関数  $b$ , および微分作用素  $P$  の組  $[b,P]$  を返す. これらは  $Pf^{s+1}=b(s)f^s$  を満たす. 微分作用素は  $v1,...,vn,dv1,...,dvn$  の可換多項式として表現されている. この表現においては, 微分を表す  $d$  のついた変数も単なる不定元として扱われているため, 係数多項式環の変数の前に表示されることもありうるが, 多項式係数を左に置く正規表現として理解する必要がある.
- オプション `weight=[v1,w1,...,vn,wn]` が指定された場合, 変数リスト  $(v1,...,vn)$  に対して `weight`  $(w1,...,wn)$  を設定して計算が行われる. このオプションは,  $f$  が  $(w1,...,wn)$  に関して weighted homogeneous の場合に有効に働く.
- オプション `heuristic=1` が指定された場合, あるイデアルのグレブナー基底を別の項順序に変換してから消去計算を行う. この方法により全体の計算が高速化する場合がある.
- デフォルトでは, 内部で用いられる変数順序は自動的に決定されるが, オプション `vord=v` が指定された場合その変数順序が使われる.

```
[...] load("nn_ndbf.rr");
[...] ndbf.bfunction(x^3-y^2*z^2);
-11664*s^7-93312*s^6-316872*s^5-592272*s^4-658233*s^3-435060*s^2
-158375*s-24500
[...] ndbf.bfunction(x^3-y^2*z^2|op=1);
[-11664*s^7-93312*s^6-316872*s^5-592272*s^4-658233*s^3-435060*s^2
-158375*s-24500, (108*z^3*x*dz^3+756*z^2*x*dz^2+1080*z*x*dz+216*x)*dx^4
...
+(729/8*z^3*dz^5+9477/8*z^2*dz^4+5103/2*z*dz^3+2025/2*dz^2)*dy^2]
```

```
[...] F=256*u1^3-128*u3^2*u1^2+(144*u3*u2^2+16*u3^4)*u1-27*u2^4
-4*u3^3*u2^2$
[...] ndbf.bfunction(F|weight=[u3,2,u2,3,u1,4]);
576*s^6+3456*s^5+8588*s^4+11312*s^3+8329*s^2+3250*s+525
```

### 1.1.2 ndbf.bf\_local

`ndbf.bf_local(f,p[|weight=w,heruristic=yesno,vord=v,op=yesno])` :: 多項式  $f$  の点  $p$  における局所  $b$  関数を計算する.

`return`      リスト  
 $f$             多項式  
 $p$              $[v1,a1,...,vn,an]$  なるリスト  
 $w$              $[v1,w1,...,vn,wn]$  なるリスト  
 $yesno$        0 または 1  
 $v$             変数のリスト

- この関数は asir-contrib のパッケージ ‘nn\_ndbf.rr’ で定義されている.
- 多項式  $f$  の  $(v1,...,vn)=(a1,...,an)$  における局所  $b$  関数 (local b-function) を計算する. 出力は局所  $b$  関数の因子, 重複度のペアのリストである.
- デフォルトでは局所  $b$  関数のみが出力されるが, オプション `op=1` が指定された場合, 局所  $b$  関数  $b$ , 微分作用素の共通分母  $a(x)$  および微分作用素  $P$  の組  $[b,a(x),P]$  を返す. これらは  $a(x)P\hat{f}(s+1)=b(s)\hat{f}s$  を満たす. 微分作用素は  $v1,...,vn,dv1,...,dvn$  の可換多項式として表現されている. この表現においては, 微分を表す  $d$  のついた変数も単なる不定元として扱われているため, 係数多項式環の変数の前に表示されることもありうるが, 多項式係数を左に置く正規表現として理解する必要がある.
- オプション `weight=[v1,w1,...,vn,wn]` が指定された場合, 変数リスト  $(v1,...,vn)$  に対して `weight (w1,...,wn)` を設定して計算が行われる. このオプションは,  $f$  が  $(w1,...,wn)$  に関して weighted homogeneous の場合に有効に働く.
- オプション `heuristic=1` が指定された場合, あるイデアルのグレブナー基底を別の項順序に変換してから消去計算を行う. この方法により全体の計算が高速化する場合がある.
- デフォルトでは, 内部で用いられる変数順序は自動的に決定されるが, オプション `vord=v` が指定された場合その変数順序が使われる.

```
[...] load("nn_ndbf.rr");
[...] ndbf.bf_local(y*((x+1)*x^3-y^2),[x,-1,y,0]);
[[-s-1,2]]
[...] ndbf.bf_local(y*((x+1)*x^3-y^2),[x,-1,y,0]|op=1);
[[[-s-1,2]],12*x^3+36*y^2*x-36*y^2,(32*y*x^2+56*y*x)*dx^2
+((-8*x^3-2*x^2+(128*y^2-6)*x+112*y^2)*dy+288*y*x+(-240*s-128)*y)*dx
+(32*y*x^2-6*y*x+128*y^3-9*y)*dy^2+(32*x^2+6*s*x+640*y^2+39*s+30)*dy
+(-1152*s^2-3840*s-2688)*y]
```

### 1.1.3 ndbf.bf\_strat

`ndbf.bf_strat(f[|weight=w,heruristic=h,vord=v])`  
:: 多項式  $f$  の, 局所  $b$  関数に付随する滑層分割 (stratification) を計算する.

*return* リスト

*f* 多項式

*w*  $[v1, w1, \dots, vn, wn]$  なるリスト

*h* 0 または 1

*v* 変数のリスト

- この関数は asir-contrib のパッケージ ‘nn\_ndbf.rr’ で定義されている.
- 多項式  $f$  の大域  $b$  関数 (global b-function) を計算する. 出力は変数  $s$  の多項式である.
- オプション  $\text{weight}=[v1, w1, \dots, vn, wn]$  が指定された場合, 変数リスト  $(v1, \dots, vn)$  に対して  $\text{weight}(w1, \dots, wn)$  を設定して計算が行われる. このオプションは,  $f$  が  $(w1, \dots, wn)$  に関して weighted homogeneous の場合に有効に働く.
- オプション  $\text{heuristic}=1$  が指定された場合, あるイデアルのグレブナー基底を別の項順序に変換してから消去計算を行う. この方法により全体の計算が高速化する場合がある.
- デフォルトでは, 内部で用いられる変数順序は自動的に決定されるが, オプション  $\text{vord}=v$  が指定された場合その変数順序が使われる.

```
[...] load("nn_ndbf.rr");
[...] F=256*u1^3-128*u3^2*u1^2+(144*u3*u2^2+16*u3^4)*u1-27*u2^4
-4*u3^3*u2^2$
[...] ndbf.bf_strat(F);
[[[u3^2,-u1,-u2],[-1],[[-s-1,2],[16*s^2+32*s+15,1],[36*s^2+72*s+35,1]]],
[[-4*u1+u3^2,-u2],[96*u1^2+40*u3^2*u1-9*u3*u2^2,...],[[-s-1,2]]],
[[-2048*u1^3-...],[-u3*u2,u2*u1,...],[[-s-1,1],...]]],
[[-256*u1^3+128*u3^2*u1^2+...],[...],[[-s-1,1]]],
[[],[-256*u1^3+128*u3^2*u1^2+...],[[]]]
```

#### 1.1.4 ndbf.action\_on\_gfs

$\text{ndbf.action\_on\_gfs}(op, v, gfs)$   
 :: 微分作用素  $op$  の  $gf^{\wedge}(s+a)$  への作用を計算する.

*return* リスト

*op* 微分作用素

*gfs*  $[g, f, s+a]$  なるリスト

*v*  $f$  の変数のリスト ( $v=[v1, \dots, vn]$ )

- 微分作用素  $op$  を  $gf^{\wedge}(s+a)$  に作用させた結果を計算する.
- $g$  は  $v1, \dots, vn$  を変数とする多項式である.
- $op$  は  $[v1, \dots, vn, dv1, \dots, dvn]$  を変数とする多項式で表現する.
- 入力リスト  $[g, f, s+a]$  は  $gf^{\wedge}(s+a)$  を表す.
- 結果は  $[h, f, s+c]$  なるリストで,  $hf^{\wedge}(s+b)$  を意味する. ここで  $c$  は整数である.  $op$  が  $b$ -関数  $b(s)$  を与える作用素なら,  $a=1$  に対し  $c=0$  で,  $h=b(s)$  (global case) または  $h=d(v)b(s)$  (local case) である.

```
[...] load("nn_ndbf.rr");
[...] F=x^5-y^2*z^2$
```

```
[...] B=ndbf.bfunction(F|op=1)$
[...] ndbf.action_on_gfs(B[1],[x,y,z],[1,F,s+1]);
[-62500000000*s^13-...-2985505717194*s-245434132944,x^5-z^2*y^2,s]
[...] L=ndbf.bf_local(F,[x,0,y,0,z,1]|op=1)$
[...] ndbf.action_on_gfs(L[2],[x,y,z],[1,F,s+1]);
[(-100000*s^5-500000*s^4-990000*s^3-970000*s^2-470090*s-90090)*z^2,
x^5-z^2*y^2,s]
```

## 1.2 Annihilator イデアル計算

### 1.2.1 ndbf.ann

`ndbf.ann(f[|weight=w])` :: 多項式  $f$  に対し  $f^s$  の annihilator ideal を計算する.

*return*      微分作用素のリスト

$f$             多項式

$w$              $[v1,w1,...,vn,wn]$  なるリスト

- この関数は asir-contrib のパッケージ ‘nn\_ndbf.rr’ で定義されている.
- 多項式  $f$  に対し,  $f^s$  の annihilator ideal を計算する. 出力は,  $s$  を係数に含む微分作用素のリストである. 微分作用素の表現方法は, `ndbf.bf_local` と同様である.
- オプション `weight=[v1,w1,...,vn,wn]` が指定された場合, 変数リスト  $(v1,...,vn)$  に対して `weight (w1,...,wn)` を設定して計算が行われる. このオプションは,  $f$  が  $(w1,...,wn)$  に関して weighted homogeneous の場合に有効に働く.

```
[...] load("nn_ndbf.rr");
[...] ndbf.ann(x*y*z*(x^3-y^2*z^2));
[(-x^4*dy^2+3*z^4*x*dz^2+12*z^3*x*dz+6*z^2*x)*dx+4*z*x^3*dz*dy^2
-z^5*dz^3-6*z^4*dz^2-6*z^3*dz,
(x^4*dy-3*z^3*y*x*dz-6*z^2*y*x)*dx-4*z*x^3*dz*dy+z^4*y*dz^2+3*z^3*y*dz,
(-x^4+3*z^2*y^2*x)*dx+(4*z*x^3-z^3*y^2)*dz,2*x*dx+3*z*dz-11*s,
-y*dy+z*dz]
```

# Index

(Index is nonexistent)

(Index is nonexistent)

## Short Contents

1	New b-function package <code>nn_ndbf.rr</code> . . . . .	1
Index	. . . . .	5



# Table of Contents

<b>1</b>	<b>New b-function package nn_ndbf.rr .....</b>	<b>1</b>
1.1	b 関数計算.....	1
1.1.1	ndbf.bfunction.....	1
1.1.2	ndbf.bf_local.....	2
1.1.3	ndbf.bf_strat.....	2
1.1.4	ndbf.action_on_gfs.....	3
1.2	Annihilator イデアル計算.....	4
1.2.1	ndbf.ann .....	4
<b>Index .....</b>		<b>5</b>