nn_ndbf

nn_ndbf User's Manual
Edition 1.0
Nov 2009

by Masayuki Noro and Kenta Nishiyama

In this manual we explain about a new b-function package 'nn_ndbf.rr' in asir-contrib. To use this package one has to load 'nn_ndbf.rr'.

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

A prefix ndbf. is necessary to call the functions in this package. In this manual we also explain about some related built-in functions.

0.1 Computation of b-function

0.1.1 ndbf.bfunction

`ndbf.bfunction(f[|weight=w,heruristic=yesno,vord=v,op=yesno]) :: computes the global b-function of a polynomial f`

return a polynomial

f a polynomial

w a list [*v1*,*w1*,...,*vn*,*wn*]

yesno 0 or 1

v a list of variables

- This function is defined in an asir-contrib package 'nn_ndbf.rr'.
- This function computes the global b-function of a polynomial *f*. By default only the global b-function is returned. If an option op=1 is given, a pair *[b,P]* of the global b-function and a differential operator satisfying *Pf^(s+1)=b(s)f^s*. The operator P is represented as a commutative polynomial of variables *v1,...,vn,dv1,...,dvn*. The d-variables are treated as commutative indeterminates in this representation and the polynomial should be regarded as a canonical representation with each polynomial coefficient placed at the left of d-variables.
- If an option weight=[*v1*,*w1*,...,*vn*,*wn*] is given, the computation is done with a weight *(w1,...,wn)* for *(v1,...,vn)*. This option is useful when *f* is weighted homogeneous with respect to *(w1,...,wn)*.
- If an option heuristic=1 is given a change of ordering is done before entering elimination. In some cases this improves the total efficiencty.
- The variable order used in the whole computation is automatically set by default. If an option vord=*v* is given, a variable order *v* is used istead.

```
[...] 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
```

0.1.2 ndbf.bf_local

`ndbf.bf_local(`*f*`,`*p*`[|weight=`*w*`,heruristic=`*yesno*`,vord=`*v*`,op=`*yesno*`]) :: computes the local b-function of a polynomial `*f*` at `*p*`.`

return a list

f a polynomail

p a list `[v1,a1,...,vn,an]`

w a list `[v1,w1,...,vn,wn]`

yesno 0 or 1

v a list of variables

- This function is defined in an asir-contrib package '`nn_ndbf.rr`'.
- This function computes the local b-function of a polynomial *f* at a point *(v1,...,vn)=(a1,...,an)*. The output is a list of pairs of each factor of the local b-function and its multiplicity.
- By default only the local b-function is returned. If an option `op=1` is given, a triple *[b,a,P]* of the local b-function, a polynomial and a differential operator satisfying *Pf^(s+1)=ab(s)f^s*. The operator P is represented as a commutative polynomial of variables *v1,...,vn,dv1,...,dvn*. The d-variables are treated as commutative indeterminates in this representation, the polynomial should be regarded as a canonical representation with each polynomial coefficient placed at the left of d-variables.
- If an option `weight=[v1,w1,...,vn,wn]` is given, the computation is done with a weight *(w1,...,wn)* for *(v1,...,vn)*. This option is useful when *f* is weighted homogeneous with respect to *(w1,...,wn)*.
- If an option `heuristic=1` is given a change of ordering is done before entering elimination. In some cases this improves the total efficiencty.
- The variable order used in the whole computation is automatically set by default. If an option `vord=`*v* is given, a variable order *v* is used istead.

```
[...] 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]
```

0.1.3 ndbf.bf_strat

`ndbf.bf_strat(`*f*`[|weight=`*w*`,heruristic=`*h*`,vord=`*v*`])`

:: computes a stratification associated with local b-function of a polynomial *f*.

return a list

f a polynomial

w a list `[v1,w1,...,vn,wn]`

h 0 or 1

v li ist of variables

- This function is defined in an asir-contrib package '`nn_ndbf.rr`'.
- This function computes a stratification assoficated with local b-function of a polynomial *f*. The output is a list *[s1,...sl]* where each *si* is a list *[l1,l2,bi]*. In this list, *l1* and *l2* is generators of ideals and they represents the local b-function is *bi* over V(*l1*)-V(*l2*).
- If an option `weight=[v1,w1,...,vn,wn]` is given, the computation is done with a weight *(w1,...,wn)* for *(v1,...,vn)*. This option is useful when *f* is weighted homogeneous with respect to *(w1,...,wn)*.
- If an option `heuristic=1` is given a change of ordering is done before entering elimination. In some cases this improves the total efficiencty.
- The variable order used in the whole computation is automatically set by default. If an option `vord=v` is given, a variable order *v* is used istead.

```
[...] 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+...],[]]]
```

0.1.4 `ndbf.action_on_gfs`

`ndbf.action_on_gfs(op,v,gfs)`
:: computes the action of an operatior *op* on *gf^(s+a)*

return a list

op a differential operator

gfs a list *[g,f,s+a]*

v list of variables of *f* (*v=[v1,...,vn]*)

- This function computes the action of a differential operator *op* on *gf^(s+a)*.
- *g* is a polynomial with variables *v1,...,vn*.
- *op* is represented by a polynonmial with *[v1,...,vn,dv1,...,dvn]*.
- The input list *[g,f,s+a]* represents *gf^(s+a)*.
- The result is a list *[h,f,s+c]* and it means *hf^(s+c)*, where *c* is an integer. If *op* is an operator giving b-function *b(s)*, then *c=0* for *a=1* and *h=b(s)* (global case) or *h=b(s)d(v)* (local case).

```
[...] 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]
```

0.2 Computation of annihilator ideal

0.2.1 ndbf.ann

`ndbf.ann(`*f*`[|weight=`*w*`]) :: computes the annihilator ideal of` *f*`^`*s* `for a polynomial` *f*`.`

return a list of differential operators

f a polynomial

w a list `[`*v0*`,`*w1*`,...,`*vn*`,`*wn*`]`

- This function is defined in an asir-contrib package '`nn_ndbf.rr`'.
- This function computes the annihilator ideal of *f*^*s* for *f*. The output is a list of defferential operators containing *s* in thier coefficients. The differential operators are represented in the same manner as `ndbf.bf_local`.
- If an option `weight=[`*v1*`,`*w1*`,...,`*vn*`,`*wn*`]` is given, the computation is done with a weight *(w1,...,wn)* for *(v1,...,vn)*. This option is useful when *f* is weighted homogeneous with respect to *(w1,...,wn)*.

```
[...] 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

Index 5

Table of Contents

0.1 Computation of b-function 1
0.1.1 ndbf.bfunction 1
0.1.2 ndbf.bf_local 2
0.1.3 ndbf.bf_strat 2
0.1.4 ndbf.action_on_gfs 3
0.2 Computation of annihilator ideal 4
0.2.1 ndbf.ann 4

Index 5