Kan/k0 Manual

Kan/k0 User's Manual
Edition 1.1.3 for OpenXM/kan/k0
Aug 29, 2025

by Nobuki Takayama

1 STANDARD function

Execute

```
load["lib/all.k"];;
```

to load all library functions. This manual is automatically generated by Doc/genhelp.k.

1.1 A list of functions

1.1.1 !ReservedNames

`!ReservedNames`
:: reserved symbols.

- The names k00*, K00*, sm1* , arg1,arg2,arg3,arg4,...., Helplist, Arglist, FunctionValue, @*, db.*, k.*, tmp002*, tmp00* are used for system functions. Percent, LeftBracket, RightBracket, Dollar, Newline are constants

1.1.2 AddString

`AddString(`*list*`):: concatenate strings.`
- AddString(*list*) returns the concatnated string (array *list*).

1.1.3 Annfs

`Annfs (?)`
::

- Annfs(f,v) computes the annihilating ideal of f^r and the Bernstein-Sato polynomial b(s) of f Return value: [Ann(f^r), r, b(s)] where r is the minimal integral root of b(s) = 0.

```
Example:  Annfs(x^2+y^2,"x,y"):
```

`See also` Not yet written.

1.1.4 Append

`Append(`*f1*`,`*fn*`,`*g*`):: append two lists or arrays.`
- Append([*f1*,...,*fn*],*g*) returns the list [*f1*,...,*fn*,*g*]

`See also` `Join`, `NewArray`, `Rest`

1.1.5 AsciiToString

`AsciiToString(`*ascii_code*`):: translate from ascii code to a string.`
- AsciiToString(*ascii_code*) returns the string of which ascii code is *ascii_code* (integer *ascii_code*).

`See also` `!ReservedNames`, `AddString`, `StringToAsciiArray`

1.1.6 BfRoots1

`BfRoots1 (?)`
::

- BfRoots1(g,v) returns the integral roots of g with respect to the weight vector (1,1,...,1) and the b-function itself

```
Example:  BfRoots1([x*Dx-2, y*Dy-3],[x,y]);
```

`See also` Not yet written.

1.1.7 Boundp

`Boundp(`*s*`):: check if a symbol is assigned a value or not.`
- Boundp(*s*) checks if the symbol *s* is bounded to a value or not (string *s*).

1.1.8 Cancel

`Cancel (?)`
::

- Cancel the greatest common multiple of the denominators and the numerator.

`See also` Not yet written.

1.1.9 CancelNumber

`CancelNumber(`*rn*`):: factor out the greatest common divisor.`
- CancelNumber(*rn*) reduces the rational number *rn* (rational *rn*).

```
CancelNumber( 2/6 ) :
```

`See also` `Cancel`

1.1.10 Characteristic

`Characteristic(`*ring*`):: get the characteristic of a given ring.`
- Characteristic(*ring*) returns the characteristic of the *ring* .

`See also` `GetRing`, `RingD`, `RingPoly`, `SetRing`

1.1.11 Coefficients

`Coefficients(`*f*`,`*v*`):: get the exponents and coefficients.`
- Coefficients(*f*,*v*) returns [exponents, coefficients] of *f* with respect to the variable *v* (polynomial *f*,*v*).

```
Coefficients(Poly("(x+1)^2"),Poly("x")):
```

`See also` `Exponent`, `Init`, `Init_w`

1.1.12 CopyArray

`CopyArray (?)`
::

- It duplicates the argument array recursively.

```
Example: m=[1,[2,3]];
         a=CopyArray(m); a[1] = "Hello";
         Println(m); Println(a);
```

See also Not yet written.

1.1.13 DC

`DC(obj,key):: translate data types.`

- DC(*obj*,*key*) converts *obj* to a new object in the primitive class *key* (object *obj*, string *key*)

```
DC(" (x+1)^10 ", "polynomial"):
```

See also ToString

1.1.14 DeRham4

`DeRham4(f):: de Rham cohomology groups of the complement of the zero set of a polynomial`

- DeRham4(f) gives the dimensions of de Rham cohomology groups of C^4-V(f). f (string) is a polynomial in Z[x,y,z,www]. DeRham2 is for Z[x,y], DeRham3 is for Z[x,y,z].

```
Example: c=DeRham4("x^3+y^2+z^2+www^2-1"); c:
         It will take more than 10 minutes. Answer will be [2,0,0,1,1]
                           standing for dim of [H^4,H^3,H^2,H^1,H^0].
```

See also ToString, Sintegration

1.1.15 Denominator

`Denominator(f):: denominator`

- Denominator(f) returns the denominator of the rational *f*.

See also Cancel, Numerator

1.1.16 Dvar

`Dvar(v):: differential operator variables standing for space variables.`

- Dvar(v) gives dvars standing for v. vv is a list of variables

```
Example: RingD("x,y"); Dvar([x,y]):
         Dvar("z"): Dvar(["z","s"]):
```

See also UseSmallD, @.Dsymbol

1.1.17 Eliminatev

`Eliminatev(`*list*`,`*var*`)::`

- Eliminatev(*list*,*var*) prunes polynomials in *list* (array of polynomials) which contains the variables in *var* (array of strings)

```
Eliminatev([Poly(" x+h "),Poly(" x ")],[ "h" ]):
```

See also `eliminatev (sm1)`

1.1.18 Error

`Error(`*s*`):: cause an error.`

- Error(*s*) causes an error and outputs a message *s*.

1.1.19 Exponent

`Exponent(`*f*`,`*vars*`):: exponents of a given polynomial.`

- Expoent(*f*,*vars*) returns the vector of exponents of the polynomial *f*

```
Exponent( x^2*y-1,[x,y])
```

See also `Coefficients`

1.1.20 Factor

`Factor (?)`
::

- Factor a given polynomial.

See also Not yet written.

1.1.21 GKZ.GKZ

`GKZ.GKZ (?)`
::

- GKZ(a,b) returns the GKZ systems associated to the matrix a and the vector b The answer is given by strings.

```
Example: GKZ([[1,1,1,1],[0,1,3,4]],[0,2]);
```

See also Not yet written.

1.1.22 Gb

`Gb (?)` ::

- Gb(f) returns the Groebner basis of f. cf. Kernel, Weyl.

See also Not yet written.

1.1.23 GetEnv

`GetEnv(`*s*`):: value of an environmental variable.`

- GetEnv(*s*) returns the value of the environmental variable string *s*.

1.1.24 GetPathName

`GetPathName(s):: find a file in the search path.`

- GetPathName(*s*) checks if the file *s* exists in the current directory or in LOAD_K_PATH. If there exists, it returns the path name (string *s*).

`See also` `GetEnv`

1.1.25 Gmp.

`Gmp.` :: class Gmp

- Gmp is a class which supports the following methods: BitAnd, BitOr, ModuloPower, ProbabilisticPrimeP, Sqrt, Gcd, Div, Mod. These methods call functions of GNU MP package. Note that there is no method to create an instance.

```
r = Gmp.Gcd(5,8);
```

1.1.26 Groebner

`Groebner(input):: compute the Grobner basis.`

- Groebner(*input*) returns Groebner basis of the left ideal or the submodule defined by *input* (array of polynomials) The order is that of the ring to which each element of *input* belongs. The input is automatically homogenized.

```
RingD("x,y",[["x", 10, "y", 1]]);
          Groebner([Poly(" x^2+y^2-4"),Poly(" x*y-1 ")]):
cf. RingD, Homogenize
```

`See also` `Gb`, `Init_w`, `RingD`, `groebner (sm1)`

1.1.27 Help

`Help(key):: display a help message.`

- Help(*key*) or help(*key*) shows an explanation on the *key* (string *key*).

`See also` `HelpAdd`, `help`, `man`

1.1.28 HelpAdd

`HelpAdd(key,explanation):: Add a help message in the system.`

- HelpAdd([*key*,*explanation*]) (string *key*, string *explanation*) or (string *key*, array *explanation*).

  HelpAdd([*key*,*explanation*,*category*]) is used to specify the *category* of the topics.

`See also` `help`, `man`

1.1.29 Init

`Init(f):: return the initial term.`

- Init(*f*) returns the initial term of the polynomial *f* (polynomial) Init(*list*) returns the array of initial terms of the array of polynomials *list* (array)

`See also` `Init_w`

1.1.30 Init_w

`Init_w(f,vars,w):: return the initial terms.`

- Init_w(*f*,*vars*,*w*) returns the initial terms with respect to the weight vector *w* (array of integer) of the polynomial *f* (polynomial). Here, *f* is regarded as a polynomial with respect to the variables *vars* (array of polynomials).

```
RingD("x,y"); Init_w(x^2+y^2+x,[x,y],[1,1]):
```

See also Gb, Init

1.1.31 Init_w_m

`Init_w_m (?)`
::

- Init_w_m(f,w,m) returns the initial of f with respect to w with the shift m. Note that the order of the ring and the weight w must be the same.

```
Example: Sweyl("x,y",[["x",-1,"Dx",1]]);
         Init_w_m([x*Dx+1,Dx^2+x^5],["x",-1,"Dx",1],[2,0]):
```

See also Not yet written.

1.1.32 IntegerToSm1Integer

`IntegerToSm1Integer(i):: translates integer to 32 bit integer (sm1 integer)`

- IntegerToSm1Integer(*i*) translates integer *i* to sm1 integer (integer *i*).

1.1.33 IsArray

`IsArray(f):: check if an given object is an array.`

- If *f* is an array (or list) object, then IsArray(*f*) returns true, else IsArray(*f*) returns false.

See also Is*, Tag

1.1.34 IsConstant

`IsConstant(f):: check if a given object is a constant.`

- IsConstant(*f*) returns true if the polynomial *f* is a constant.

See also DC, Is*, Tag

1.1.35 IsDouble

`IsDouble(a):: check if a given object is double.`

- IsDouble(*a*) returns true if *a* is a double (object *a*). It returns false if *a* is not.

See also DC, Is*, Tag

1.1.36 IsExact_h

`IsExact_h (?)`
::

- IsExact_h(complex,var): bool It checks the given complex is exact or not in D<h> (homogenized Weyl algebra) cf. ReParse

`See also` Not yet written.

1.1.37 IsInteger

`IsInteger(a):: check if a given object is an integer.`

- IsInteger(a) returns true if a is an integer (object a). It returns false if a is not. cf. IsSm1Integer

`See also` `DC`, `Is*`, `Tag`

1.1.38 IsObject

`IsObject(a):: check if a given object is primitive or not.`

- IsObject(a) return true if a is an Object.

`See also` `Tag`

1.1.39 IsPolynomial

`IsPolynomial(`*obj*`):: check if a given object is a polynomial.`

- IsPolynomial(f) returns true if f (object) is a polynomial.

`See also` `DC`, `Is*`, `Tag`

1.1.40 IsRational

`IsRational(a):: check if a given object is a rational.`

- IsRational(a) returns true if a is a rational (object a). It returns false if a is not.

`See also` `DC`, `Is*`, `Tag`

1.1.41 IsRing

`IsRing(`*obj*`):: check if a given object is a ring.`

- IsRing(obj) returns true if obj is a ring (object obj).

`See also` `DC`, `Is*`, `Tag`

1.1.42 IsSameIdeal_h

`IsSameIdeal_h (?)`
::

- IsSameIdeal_h(ii,jj,var): bool It checks the given ideals are the same or not in D<h> (homogenized Weyl algebra) cf. ReParse

`See also` Not yet written.

1.1.43 IsSm1Integer

`IsSm1Integer(obj):: check if a given object is a 32 bit integer.`

- IsSm1Integer(*obj*) returns true if *obj* is an integer of sm1(object *obj*).

See also DC, Is*, Tag

1.1.44 IsString

`IsString(obj):: check if a given object is a string.`

- IsString(*obj*) returns true if *obj* is a string (object *obj*).

```
if (IsString("abc")) Println("Hello"); ;
```

See also DC, Is*, Tag

1.1.45 Join

`Join(f1,fn,g1,gm):: join two lists or arrays`

- Join([*f1*,...,*fn*],[*g1*,...,*gm*]) returns the list [*f1*,...,*fn*,*g1*,...,*gm*]

See also Append, NewArray, Rest

1.1.46 Kernel

`Kernel (?)`
::

- Kernel(f) returns the syzygy of f. Return value [b, c]: b is a set of generators of the syzygies of f : c=[gb, backward transformation, syzygy without dehomogenization

```
Example:  Weyl("x,y",[["x",-1,"Dx",1]]);
          s=Kernel([x*Dx+1,Dx^2+x^5]); s[0]:
```

See also Not yet written.

1.1.47 Length

`Length(vec):: length of a given array or a list.`

- Length(*vec*) returns the length of the array *vec* (array *vec*)

1.1.48 Ln

`Ln():: print newline.`

- Print newline.

See also Print, Println

1.1.49 Load_sm1

`Load_sm1(s,flag):: load a sm1 program`

- Load_sm1(*s*,*flag*) loads a sm1 program from *s*[0], *s*[1], ... If loading is succeeded, the already-loaded *flag* is set to true. (list *s*, string *flag*).

See also load

1.1.50 Map

`Map(karg,func):: apply a function to each element of a list.`

- Map(*karg*,*func*) applies the function *func* to the *karg* (string *func*).

```
Map([82,83,85],"AsciiToString"):
```

1.1.51 Mapto

`Mapto(obj,ring):: parse a polynomial in a given ring.`

- Mapto(*obj*,*ring*) parses *obj* as elements of the *ring* . (ring *ring* , polynomial *obj* or array of polynomial *obj*).

```
R = RingD("x,y"); SetRingVariables();
   f = (x+y)^2; R2 = RingD("x,y,z",[["y",1]]);
   f2 = Mapto(f,R2); f2:
```

`See also` `ReParse`

1.1.52 Max

`Max (?)` ::

- Max(v) returns the maximal element in v.

`See also` Not yet written.

1.1.53 Mod

`Mod(f,p):: modulo`

- Mod(*f*,*p*) returns f modulo *n* where *f* (polynomial) and *p* (integer).

`See also` `gbext (sm1)`

1.1.54 NewArray

`NewArray(n):: it returns an array of a given size.`

- NewArray(*n*) returns an array of size integer *n*.

`See also` `NewMatrix`

1.1.55 NewMatrix

`NewMatrix(m,n):: generate a matrix of a given size.`

- NewMatrix(*m*,*n*) returns the (*m*,*n*)-matrix (array) with the entries 0.

`See also` `NewArray`

1.1.56 Numerator

`Numerator(f):: numerator`

- Numerator(*f*) returns the numerator of the rational *f*.

`See also` `Cancel`, `Denominator`

1.1.57 Ord_w_m

`Ord_w_m (?)`
::

- Ord_w_m(f,w,m) returns the order of f with respect to w with the shift m. Note that the order of the ring and the weight w must be the same. When f is zero, it returns -intInfinity = -999999999.

```
Example:  Sweyl("x,y",[["x",-1,"Dx",1]]);
          Ord_w_m([x*Dx+1,Dx^2+x^5],["x",-1,"Dx",1],[2,0]):
```

`See also` Not yet written.

1.1.58 OutputPrompt

`OutputPrompt`
:: output the prompt.

- Output the prompt. Files should end with this command.

1.1.59 Pmat

`Pmat(`*m*`):: print an given array in a pretty way.`

- Pmat(*m*): Print the array *m* in a pretty way.

`See also` `Println`

1.1.60 Poly

`Poly(`*name*`):: translate a string to a polynomial.`

- Poly(*name*) parses a given string *name* in the current ring and returns a polynomial.

`See also` `DC`, `PolyR`, `ReParse`, `RingD`

1.1.61 PolyR

`PolyR(`*name*`,`*r*`):: translate a string to a polynomial in a given ring.`

- PolyR(*name*,*r*) parses a string *name* in the ring *r*

```
r = RingD("x,y"); y = PolyR("x+2*y",r);
```

`See also` `Poly`

1.1.62 Position

`Position(`*list*`,`*elem*`):: find a position of an element in a list.`

- Position(*list*,*elem*) returns the position p of the element *elem* in the array *list*. If *elem* is not in *list*, it return -1 (array *list*).

```
Position([1,34,2],34):
```

1.1.63 Print

`Print(f):: display a given object.`

- Print(f) prints f without the newline.

`See also` `Println`

1.1.64 Println

`Println(f):: display a given object with the newline.`

- Println(f) prints f and goes to the new line.

`See also` `Ln`, `Print`, `Stderr`.

1.1.65 Protect

`Protect(name,level):: add read-only property for a given variable.`

- Protect(*name*) protects the symbol *name* (string) Protect(*name*,*level*) protects the symbol *name* (string) with *level*

`See also` `extension (sm1)`

1.1.66 QuoteMode

`QuoteMode(number):: Change into the quote mode.`

- QuoteMode(1) sets the parser in the quotemode; if unknown function symbol comes, it automatically translates the expression into a tree.

```
Example 1: class polymake extends PrimitiveObject {local ; def hogera() { return(1);} }
         QuoteMode(1); polymake.afo(1,2):
Example 2: polymake="polymake";
         QuoteMode(1); polymake.afo(1,2):
QuoteMode(0) turns off the quotemode.
```

`See also` `Tag`

1.1.67 ReParse

`ReParse(obj):: parses a given object in the current ring.`

- Reparse(*obj*): It parses the given object *obj* in the current ring.

`See also` `Mapto`, `RingD`

1.1.68 ReducedBase

`ReducedBase(base):: remove unnecessary elements.`

- ReducedBase(*base*) prunes redundant elements in the Grobner basis *base* (array).

`See also` `Gb`, `Groebner`

1.1.69 Reduction

`Reduction(f,g):: get the remainder and the quotients.`

- Reduction(f,G) returns the remainder and sygygies when f is devided by G (polynomial f, array G).

`See also` `Gb`

1.1.70 Replace

`Replace(f,rule):: substitute variables by given values`

- Replace(*f*,*rule*) rewrites *f* by the *rule* (polynomial *f*, array *rule*).

```
RingD("x,y"); Replace( (x+y)^3, [[x,Poly("1")]])
```

`See also  replace (sm1)`

1.1.71 Rest

`Rest(a):: it returns the rest of a given list.`

- Rest(*a*) returns the rest (cdr) of *a* (list *a*).

`See also  Append, Join`

1.1.72 RingD

`RingD(names,weight_vector):: define a new ring of differential operators.`

- RingD(*names*) defines a new ring of differential operators (string *names*). RingD(*names*,*weight_vector*) defines a new ring with the weight vector (string *names*, array *weight_vector*). RingD(*names*,*weight_vector*,*characteristic*) Dx is the associated variable to x where Dx*x - x*Dx = 1 holds.

```
RingD("x,y",[["x",2,"y",1]])
```

`See also  GetRing, PolyR, ReParse, SetRing`

1.1.73 RingPoly

`RingPoly(names,weight_vector):: define a ring of polynomials.`

- RingPoly(*names*) defines a ring of polyomials (string *names*). The names of variables of that ring are *names* and the homogenization variable h. cf. SetRingVariables, RingD

```
R=RingPoly("x,y");

RingPoly(names,weight_vector) defines a ring of polynomials
with the order defined by the weight_vector
(string names, array of array weight_vector).
RingPoly(names,weight_vector,characteristic)
Example: R=RingPoly("x,y",[["x",10,"y",1]]);
         (x+y)^10:
```

`See also  GetRing, RingD, SetRing`

1.1.74 Sannfs2

`Sannfs2 (?)`
`::`

- Sannfs2(f) constructs the V-minimal free resolution for the weight (-1,1) of the Laplace transform of the annihilating ideal of the polynomial f in x,y. See also Sminimal, Sannfs3.

```
Example: a=Sannfs2("x^3-y^2");
         b=a[0]; sm1_pmat(b);
         b[1]*b[0]:
Example: a=Sannfs2("x*y*(x-y)*(x+y)");
         b=a[0]; sm1_pmat(b);
         b[1]*b[0]:
```

See also Not yet written.

1.1.75 Sannfs3

Sannfs3 (?)
::

- Sannfs3(f) constructs the V-minimal free resolution for the weight (-1,1) of the Laplace transform of the annihilating ideal of the polynomial f in x,y,z. See also Sminimal, Sannfs2.

```
Example: a=Sannfs3("x^3-y^2*z^2");
         b=a[0]; sm1_pmat(b);
         b[1]*b[0]: b[2]*b[1]:
```

See also Not yet written.

1.1.76 Save

Save(*obj*):: write a given object to a file.

- Save(*obj*) appends *obj* to the file sm1out.txt (object *obj*).

1.1.77 SetRingVariables

SetRingVariables()::

- Set the generators of the current ring as global variables. You do not need explicitly call this function which is called from RingD. cf. RingD(), Poly(), PolyR()

See also RingD

1.1.78 SgetShift

SgetShift (?)
::

- SgetShift(mat,w,m) returns the shift vector of mat with respect to w with the shift m. Note that the order of the ring and the weight w must be the same.

```
Example:  Sweyl("x,y",[["x",-1,"Dx",1]]);
          SgetShift([[x*Dx+1,Dx^2+x^5],[Poly("0"),x],[x,x]],["x",-1,"Dx",1],[2,0]):
```

See also Not yet written.

1.1.79 SgetShifts

`SgetShifts (?)`
::

- SgetShifts(resmat,w) returns the shift vectors of the resolution resmat with respect to w with the shift m. Note that the order of the ring and the weight w must be the same. Zero row is not allowed.

```
Example:   a=Sannfs2("x^3-y^2");
           b=a[0]; w = ["x",-1,"y",-1,"Dx",1,"Dy",1];
           Sweyl("x,y",[w]); b = Reparse(b);
           SgetShifts(b,w):
```

`See also` Not yet written.

1.1.80 Sinit_w

`Sinit_w (?)`
::

- Sinit_w(resmat,w) returns the initial of the complex resmat with respect to the weight w.

```
Example:   a=Sannfs2("x^3-y^2");
           b=a[0]; w = ["x",-1,"y",-1,"Dx",1,"Dy",1];
           Sweyl("x,y",[w]); b = Reparse(b);
           c=Sinit_w(b,w); c:
```

`See also` Not yet written.

1.1.81 Sintegration

`Sintegration (?)`
::

- Ans=Sintegration(gg,v,intv) gives the cohomology group of the integration (complex) of gg along the list of variables intv. Here, v is a list of variables. Interation variables intv must be in the first part of v. Ans[1] gives the cohomology groups. When v=intv, Ans[0] gives the dimension of the cohomology groups. Sintegration uses the function Srestall. cf. Srestriction

```
Example: RingD("x,t"); II=Sintegration([Dt-(3*t^2-x),Dx+t],[t,x],[t]); II:
```

`See also` Not yet written.

1.1.82 Sminimal

`Sminimal (?)`
::

- It constructs the V-minimal free resolution by LaScala's algorithm option: "homogenized" (no automatic homogenization) : "Sordinary" (no (u,v)-minimal resolution) Options should be given as an array.

```
Example:  Sweyl("x,y",[["x",-1,"y",-1,"Dx",1,"Dy",1]]);
          v=[[2*x*Dx + 3*y*Dy+6, 0],
             [3*x^2*Dy + 2*y*Dx, 0],
             [0,  x^2+y^2],
             [0,  x*y]];
          a=Sminimal(v);
          Sweyl("x,y",[["x",-1,"y",-1,"Dx",1,"Dy",1]]);
          b = ReParse(a[0]); sm1_pmat(b);
          IsExact_h(b,[x,y]):
Note:  a[0] is the V-minimal resolution. a[3] is the Schreyer resolution.
 ---> D^{m_3} --b[2]--> D^{m_2} --b[1]--> D^{m_1} --b[0]--> D^{m_0}
  Here D^{m_i} are the set of row vectors.
```

See also Not yet written.

1.1.83 Sord_w

Sord_w (?)

::

- Sord_w(f,w) returns the w-order of f

```
Example: Sord_w(x^2*Dx*Dy,[x,-1,Dx,1]):
```

See also Not yet written.

1.1.84 SresolutionFrameWithTower

SresolutionFrameWithTower (?)

::

- It returs [resolution of the initial, gbTower, skelton, gbasis] option: "homogenized" (no automatic homogenization)

```
Example: Sweyl("x,y");
         a=SresolutionFrameWithTower([x^3,x*y,y^3-1]);
```

See also Not yet written.

1.1.85 Srestall

Srestall (?)

::

- Srestall(gg,v,rv,k1) evaluates the dimensions of all restictions of gg along the list of variables rv. Here, v is a list of variables and k1 is the maximal integral root of the b-function of gg. Srestall uses the function Sminimal to get a (-w,w)-minimal free resolution. cf. Bfroots1(ii,vv)

See also Not yet written.

1.1.86 Srestriction

`Srestriction (?)`
::

- Ans=Srestriction(gg,v,restv) gives the cohomology group of the restriction (complex) of gg along the list of variables restv. Here, v is a list of variables. Restriction variables restv must be in the first part of v. Ans[1] gives the cohomology groups. Eliminate module base 1,e_,e_^2, ... in Ans[1,0,1] by the order 1>e_>e_^2>..., then the restriction ideal is obtained. When v=restv, Ans[0] gives the dimension of the cohomology groups. Srestriction uses the function Sintegration.

```
Example: RingD("x,y"); R=Srestriction([x*Dx-1,y*Dy-1],[x,y],[x]); R:
```

`See also` Not yet written.

1.1.87 Stderr.

`Stderr.` :: class Stderr

- Stderr is a class which supports printing to stderr. Methods are Print, Println, Ln, Flush. Stderr_fd is a global variable to save a file descriptor.

1.1.88 StringToAsciiArray

`StringToAsciiArray(`*s*`):: translate a string to an array of ascii codes.`

- StringToAsciiArray(*s*) decomposes the string *s* into an array of ascii codes of *s*. cf. AsciiToString.

`See also` `AsciiToString`

1.1.89 Substitute

`Substitute(`*f*`,`*xx*`,`*g*`)::`

- Substitute(*f*,*xx*,*g*) replaces *xx* in *f* by *g* . This function takes coeffients of *f* with respect to *xx* and returns the inner product of the vector of coefficients and the vector of which elements are g^(corresponding exponent). Note that it may cause an unexpected result in non-commutative rings.

`See also` `Replace`

1.1.90 System

`System(`*comm*`):: call the unix shell.`

- System(*comm*) executes the unix system command *comm* (string *comm*)

```
System("ls");
```

1.1.91 Tag

`Tag(`*f*`):: return the tag of a given object.`

- Tag(*f*) returns the datatype tag of *f* where 0: null, 5: string, 6: array, 9: polynomial, 15: integer(big-num), 16: rational, 18:double, 257: Error

```
Tag([Poly("0"), 0]):
```

See also Is*

1.1.92 ToDouble

ToDouble(*f*):: translate a given object to double.

- ToDouble(f) translates f into double when it is possible object f.

```
ToDouble([1,1/2,[5]]):
```

See also DC, Is*, Tag

1.1.93 ToString

ToString(*obj*):: translate a given object to a string.

- ToString(*obj*) transforms the *obj* to a string.

See also DC

1.1.94 ToricIdeal

ToricIdeal (?)
::

- ToricIdeal(a) returns the affine toric ideal associated to the matrix a The answer is given by a list of strings.

```
Example: ToricIdeal([[1,1,1,1],[0,1,3,4]]);
```

See also Not yet written.

1.1.95 Translate.to_int0

Translate.to_int0 (?)
::

- to_int0(a) : as same as sm1_push_int0.

See also Not yet written.

1.1.96 Transpose

Transpose(*m*):: transposition

- Transpose(m.) return the transpose of the matrix m (array of array m).

See also NewMatrix

1.1.97 UseSmallD

UseSmallD (?)
::

- UseSmallD() changes the Dsymbol to d and E,H symbols to ee0,hh @.Dsymbol can be refered by aaaDsymbol.

See also Not yet written.

1.1.98 Weyl

`Weyl (?)` ::

- Weyl(v,w) defines the Weyl algebra (the ring of differential operators) with the weight vector w.

```
Example:  Weyl("x,y",[["x",-1,"Dx",1]]);
```

`See also` Not yet written.

1.1.99 false

`false` ::

- false returns sm1 integer 0.

1.1.100 generic_bfct

`generic_bfct(`*ii*`,`*vv*`,`*dd*`,`*ww*`):: b-function for a weight vector`

- generic_bfct(ii,vv,dd,ww) gives the b-function of the ideal ii for the weight vector ww. vv is a list of variables and dd is a list of D-variables. It calls ox_asir and returns a string

```
Example: RingD("x,y"); f=generic_bfct([Dx^2+Dy^2-1,Dx*Dy-4],[x,y],[Dx,Dy],[1,1]); Prin
      RingD("s"); ff=ReParse(f); Factor(ff);
```

`See also` `Factor`, `Srestriction`, `Sintegration`

1.1.101 load

`load(`*fname*`):: load a given file.`

- load(*fname*) loads the file *fname* (string *fname*). load *fname* loads the file *fname* . load[*fname*] loads the file *fname* with the preprocessing by /lib/cpp.

`See also` `Load_sm1`

1.1.102 sm1

`sm1(`*arg1*`,`*arg2*`):: execute sm1 commands`

- sm1(*arg1*,*arg2*,...) is used to embed sm1 native code in the kxx program.

```
sm1( 2, 2, " add print ");
def myadd(a,b) { sm1(" a b add /FunctionValue set "); }
```

`See also` `usage (sm1)`

1.1.103 sm1_deRham

`sm1_deRham (?)`
::

- sm1_deRham(f,v) computes the dimension of the deRham cohomology groups of C^n - V(f) This function does not use (-w,w)-minimal free resolution.

```
Example:  sm1_deRham("x^3-y^2","x,y");
```

`See also` Not yet written.

1.1.104 `true`

`true` ::

- true returns sm1 integer 1.

2 COMPLEX function

2.1 A list of functions

2.1.1 Res_solv

`Res_solv(`*m*`,`*d*`)`
:: Find a solution u of the linear indefinite equation u $m = d$.

`Res_solv(`*m*`,`*d*`,`*r*`)`
:: Find a solution u of the linear indefinite equation u $m = d$. r is a ring object.

return When [c,r] is the return value, c/r is the solution u.

m Matrix or vector

d Vector or scalar

- Find a solution u of the linear indefinite equation u $m = d$.
- It solves the linear indefinite equation in the ring of differential operators (with graded reverse lexicographic order) of the same set of variables of the ring to which m or d belongs. When the ring r is given, it solves the linear indefinite equation in the ring of differential operators (with graded reverse lexicographic order) of the same set of variables of the ring r.
- When m and d consist of constants, a ring r should be given.

```
In(16)= RingD("x,y");
In(17)= mm=[Dx,Dy,x];
In(18)= Res_solv(mm,1):
[    [    x , 0 , -Dx ]  , -1 ]
```

The output means that -x*Dx + 0*Dy+Dx*x = 1.

```
In(4)=RingD("x");
     m=[ [x*Dx+2, 0],[Dx+3,x^3],[3,x],[Dx*(x*Dx+3)-(x*Dx+2)*(x*Dx-4),0]];
     d=[1,0];
     Res_solv(m,d):

[    [    x^2*Dx-x*Dx-4*x-1 , 0 , 0 , x ]  , -2 ]
```

The output implies that -(1/2)*(x^2*Dx-x*Dx-4*x-1)*[x*Dx+2, 0]-(1/2)*[Dx*(x*Dx+3)-(x*Dx+2)*(x*Dx-4),0] = [1,0]

```
In(4)= r=RingD("x,y");
In(5)= Res_solv([[1,2],[3,4]],[5,0],r):
[    [    10 , -5 ]  , -1 ]
```

See also Res_solv_h, Kernel, GetRing, SetRing.

Files lib/restriction/complex.k

2.1.2 Res_solv2

`Res_solv2(m,v,j)`
:: Find a solution u of the linear indefinite equation u $m = v$ mod j.

`Res_solv2(m,v,j,r)`
:: Find a solution u of the linear indefinite equation u $m = v$ mod j. r is a ring object.

return When [c,r] is the return value, c/r is the solution u.

m Matrix or vector

v j Vector or scalar

- Find a solution u of the linear indefinite equation u $m = v$ mod j.
- Let m be the left D-homomorphism m : D^p —> D^q/j. The function returns an element in m^(-1)(v).
- It solves the linear indefinite equation in the ring of differential operators (with graded reverse lexicographic order) of the same set of variables of the ring to which m or v belongs. When the ring r is given, it solves the linear indefinite equation in the ring of differential operators (with graded reverse lexicographic order) of the same set of variables of the ring r.
- When m and v consist of constants, a ring r should be given.

```
In(28)= r=RingD("x,y");
In(29)= Res_solv2([x,y],[x^2+y^2],[x]):
[    [    0 , y ]  , 1 ]
```

The output means that 0*x + y*y = x^2+y^2 mod x

```
In(32)= Res_solv2([x,y],[x^2+y^2],[],r):
[    [    x , y ]  , 1 ]
```

The output implies that x*x + y*y = x^2+y^2.

See also `Res_solv2_h`, `Kernel2`, `GetRing`, `SetRing`.

Files `lib/restriction/complex.k`

2.1.3 Kernel

`Kernel(m)`
:: Find solution basis of the linear indefinite equation u $m = 0$.

`Kernel(m,r)`
:: Find solution basis of the linear indefinite equation u $m = 0$. r is a ring object.

return List

m Matrix or vector

- Find solution basis of the linear indefinite equation u $m = 0$.

- When the return value is k, k[0] is a set of generators of the kernel. k[1] is [gb, backward transformation, syzygy without dehomogenization].
- It finds the kernel in the ring to which m belongs. When the ring r is given, it finds the kernel in the ring r.
- When m consists of constants, a ring r should be given.

```
In(16)= RingD("x,y");
In(17)= mm=[[Dx],[Dy],[x]];
In(18)= Pmat(Kernel(mm));
 [
  [
    [    -x*Dx-2 , 0 , Dx^2 ]
    [    -x*Dy , -1 , Dx*Dy ]
    [    -x^2 , 0 , x*Dx-1 ]
  ]
  [
   [
     [    -1 ]
   ]
   [
     [    x , 0 , -Dx ]
   ]
   [
     [    -x*Dx-2 , 0 , Dx^2 ]
     [    -x*Dy , -1 , Dx*Dy ]
     [    -x^2 , 0 , x*Dx-1 ]
   ]
  ]
 ]



In(4)= r=RingD("x,y");
In(5)= k=Kernel([[1,2],[2,4]],r); k[0]:
[    [    2 , -1 ]  ]
```

See also Kernel_h, Res_solv, GetRing, SetRing.

Files lib/restriction/complex.k

2.1.4 Kernel2

Kernel2(*m*)
:: Get the kernel of m : D^p —> D^q/j.

Kernel2(*m*,*r*)
:: Get the kernel of m : D^p —> D^q/j. r is a ring object.

return List

m j Matrix or vector

- Get a set of generators of the the kernel of *m* : D^p —> D^q/*j*.
- D^p is a set of row vectors. When u is an element of D^p, define the map from D^p to D^q/*j* by u *m*.
- It finds the kernel in the ring to which *m* belongs. When the ring *r* is given, it finds the kernel in the ring *r*.
- When *m* consists of constants, a ring *r* should be given.

```
In(27)= r=RingD("x,y");
In(28)= Kernel2([[x,y],[x^2,x*y]],[]):
[    [    -x , 1 ]  ]
In(29)=Kernel2([[x,y],[x^2,x*y]],[[x,y]]):
[    [    1 , 0 ]  , [    0 , 1 ]  ]

In(41)=Kernel2([0],[0],r):
[    [    1 ]  , [    0 ]  ]
In(42)=Kernel2([[0,0],[0,0]],[[0,0]],r):
[    [    1 , 0 ]  , [    0 , 1 ]  , [    0 , 0 ]  ]
In(43)=Kernel2([[0,0,0],[0,0,0]],[],r):
[    [    1 , 0 ]  , [    0 , 1 ]  ]
```

See also Kernel2_h, Res_solv2, GetRing, SetRing, Kernel

Files lib/restriction/complex.k

2.1.5 Gb

Gb(*f*) :: It computes the Grobner basis of *f*.

Gb(*m*,*r*) :: It computes the Grobner basis of *f*. *r* is a ring object.

Gb_h(*f*) :: It computes the Grobner basis of *f*.

Gb_h(*m*,*r*)
:: It computes the Grobner basis of *f*. *r* is a ring object.

return List

f Matrix or vector

- Compute the Grobner basis of *f*.
- Functions with _h computes Grobner bases in the homogenized Weyl algebra.
- When the return value is k, k[0] is the Grobner basis. k[1] is the initial ideal or the initial module of *f*, when the ring is defined with a weight vector.
- It computes the Grobner basis in the ring to which *f* belongs. When the ring *r* is given, it computes the Grobner basis in the ring *r*.
- When *f* consists of constants, a ring *r* should be given.

```
In(5)= r=RingD("x,y");
In(6)= m=[[x^2+y^2-1],[x*y-1]];
In(7)= Gb(m):
[    [    [    x^2+y^2-1 ]  , [    x*y-1 ]  , [    y^3+x-y ]  ]  ,
```

```
[    [    x^2+y^2-1 ]  , [    x*y-1 ]  , [    y^3+x-y ]  ]  ]

In(11)= RingD("x,y",[["x",1]]);
In(12)= r=RingD("x,y",[["x",1]]);
In(13)= Gb(m,r):
[    [    [    x+y^3-y ]  , [    -y^4+y^2-1 ]  ]  ,
[    [    x ]  , [    -y^4+y^2-1 ]  ]  ]
```

See also Gb_h, Kernel, Res_solv, RingD, GetRing, SetRing.

Files lib/restriction/complex.k

2.1.6 Res_shiftMatrix

Res_shiftMatrix(*m*,*v*)
:: Generate a matrix associated to a degree shift vector *m*

Res_shiftMatrix(*m*,*v*,*r*)
:: Generate a matrix associated to a degree shift vector *m* *r* is a ring object.

return Matrix

m Vector

v 多項式変数または文字列

- Returns n by n matrix diag(*v*^(*m*1), ..., *v*^(*m*n))

```
In(5)= r=RingD("x,y");
In(6)= Res_shiftMatrix([-1,0,3],x):
[    [    x^(-1) , 0 , 0 ]  , [    0 , 1 , 0 ]  , [    0 , 0 , x^3 ]  ]

In(9)=  rrr = RingD("t,x,y",[["t",1,"x",-1,"y",-1,"Dx",1,"Dy",1]]);
In(10)=  m=[Dx-(x*Dx+y*Dy+2),Dy-(x*Dx+y*Dy+2)];
In(12)=  m=Gb(m);
In(13)=  k = Kernel_h(m[0]);
In(14)=  Pmat(k[0]);
 [
   [    -Dy+3*h , Dx-3*h , 1 ]
   [    -x*Dx+x*Dy-y*Dy-3*x*h , y*Dy+3*x*h , h-x ]
 ]

In(15)=Pmat(m[0]);
  [    Dx*h-x*Dx-y*Dy-2*h^2 , Dy*h-x*Dx-y*Dy-2*h^2 ,
       x*Dx^2-x*Dx*Dy+y*Dx*Dy-y*Dy^2 ]

In(18)=k2 = Gb_h(k[0]*Res_shiftMatrix([1,1,1],t));
In(19)=Pmat(Substitute(k2[0],t,1));
 [
   [    -Dy+3*h , Dx-3*h , 1 ]
   [    -x*Dx+x*Dy-y*Dy-3*x*h , y*Dy+3*x*h , h-x ]
```

```
]
```

See also `Gb`, (m,(u,v))-Grobner basis

Files `lib/restriction/complex.k`

3 Primitive function

3.1 A list of functions

3.1.1 Getxvars

`Getxvars()`
:: Return x variables

return [x_list, x_str] x_list is a list of x variables, x_str is a string consisting of x variables separated by commas.

```
In(4)=RingD("x,y");
In(5)=Getxvars():
[    [    y , x ]  , y,x, ]
```

See also

Files lib/restriction/complex.k

3.1.2 Firstn

`Firstn(`*m*`,`*n*`)`
:: Return the first *n* elements of *m*.

return Matrix or vector

m Matrix or vector

n Number

- The first n elements of m. When m is a matrix, it returns the matrix consisting of first n elements of rows of m.

```
In(16)= mm = [[1,2,3],[4,5,6]];
In(17)= Firstn(mm,2):
    [[1,2],
     [4,5]]
```

See also

Files lib/restriction/complex.k

index

(Index is nonexistent)

(Index is nonexistent)

Short Contents

Table of Contents