OpenXM/Risa/Asir-Contrib

OpenXM/Risa/Asir-Contrib User's Manual (日本語版)
Edition 1.2.3 for OpenXM/Asir2000
February 2005.

by OpenXM Developing Team

1 はじめに

数式処理システム `asir` は `OpenXM` プロトコル (Open message eXchange for Mathematics, http://www.openxm.org) をサポートしたサーバをコンポーネントとして利用できる．これらのサーバを呼ぶためのインタフェース関数はファイル '`xm`' をロードすることによりシステムに読み込まれる．Risa/Asir (OpenXM 配布版) では起動時に自動的に '`xm`' が読まれる．Risa/Asir (OpenXM 配布版) は, このマニュアルでは `OpenXM/Risa/Asir` と呼ぶ．このマニュアルでは `asir` 用のこれらの関数およびユーザ言語で書かれた数学関数およびユーティリティ関数を説明する．

`OpenXM` プロトコルの技術的詳細については, '`$(OpenXM_HOME)/doc/OpenXM-specs`' にあるファイル '`openxm-jp.tex`' を見て下さい．

それでは, あなたの計算機上で数学をお楽しみ下さい．

List of contributors:

- Maekawa, Masahide (Oct., 1999 – : CVS server)
- Noro, Masayuki (Jan., 1996 – : OpenXM Protocol OXRFC-100, asir2000)
- Ohara, Katsuyoshi (Jan., 1998 – : ox_math, oxc OXRFC-101)
- Takayama, Nobuki (Jan., 1996 – : OpenXM Protocol OXRFC-100, kan/sm1, asir-contrib)
- Tamura, Yasushi (Nov., 1998 – : OpenMath proxy, tfb)

- Fujimoto, Mitsushi (Windows)
- Iwane, Hidenao (Knapsack factorizer)
- Nakayama, Hiromasa (Gaussian elimination)
- Okutani, Yukio (Oct., 1999 – Feb., 2000 : matrix, diff, ...)
- Stillman, Mike (Macaulay 2 client and server)
- Tsai, Harrison (Macaulay 2 client and server)

この Contrib パッケージの著作権については, `OpenXM/Copyright` を見て下さい.

有用だとおもいますが無保証です.

2 Asir Contrib の函数名について

Asir Contrib には (1) 標準的な名前で定義された数学函数 (`names.rr` または `longname`) および (2) Asir 標準函数以外の有用なライブラリ函数および (3) OpenXM サーバを asir から呼ぶための函数が含まれている.

Asir Contrib の全ての函数名は次の形をしている: カテゴリ名_函数名

OpenXM サーバを呼び出す函数名はかならず, `OpenXM` サーバ名_函数名という形をしている. たとえば `sm1_hilbert` は OpenXM サーバ `sm1` の Hilbert 函数の計算函数を呼び出す函数である. 一方 `poly_hilbert_polynomial` は Asir Contrib の Hilbert 函数を計算するための (1) に属する標準的な函数名である. 標準函数 `poly_hilbert_polynomial` は, 現在 `sm1_hilbert` を呼び出して Hilbert 函数を計算しているが, これは将来変更されるかもしれない. たとえば, Asir 言語で記述された有用なライブラリ函数集 `commutativeRing.rr` が開発されて Hilbert 函数の計算函数 `commutativeRing_hilbert_polynomial` が含まれるようになったら, 標準函数 `poly_hilbert_polynomial` は, `commutativeRing_hilbert_polynomial` を呼び出して Hilbert 函数を計算するようになるかもしれない. したがって, ユーザプログラムは標準数学函数名を用いるのが望ましい.

標準数学函数名は, OpenXM project において, 全ての module で共通の仕様を持つように努力している. たとえば, kan/k0 も Asir Contrib と同様の標準数学函数名を持つ予定である. 現在実験的に数学函数のカテゴリ `complex` 複体 (複素数でない) のマニュアルを kan/k0, asir/contrib で共通化を試みている.

以下の章は, 標準数学函数の解説をおこない, それからライブラリ函数, それから, OpenXM サーバのインタフェースの説明をおこなう.

3 Windows 版 Asir-contrib

Windows でも不完全ながら asir-contrib が動作する. 現在, 外部コンポーネント sm1 および, 外部 コンポーネント を利用しない asir-contrib の関数が動作する. Cygwin 環境では外部コンポーネント sm1, phc が動作する. その他の外部コンポーネントは動作しない.

次の関数は Windows では動作しない. Windows での cygwin 環境では動作する場合がある.

- `gnuplot_*`
- `om_*`
- `m_*`
- `phc_*`
- `print_dvi_form`
- `print_gif_form`
- `print_open_math_xml_form`
- `print_png_form`
- `print_xdvi_form`
- `print_xv_form`
- `tigers_xv_form`

4 基礎 (標準函数)

4.0.1 base_cancel

base_cancel(S)
: It simplifies S by canceling the common factors of denominators and numerators.

Example:

```
base_cancel([(x-1)/(x^2-1), (x-1)/(x^3-1)]);
```

4.0.2 base_choose

base_choose(L,M)
: It returns the list of the order M subsets of L.

Example:

```
base_choose([1,2,3],2);
```

It outputs all the order 2 subsets of the set $\{1, 2, 3\}$

4.0.3 base_flatten

base_flatten(S)
: It flattens a nested list S.

Example:

```
base_flatten([[1,2,3],4]);
```

4.0.4 base_intersection

base_intersection(A,B)
: It returns the intersection of A and B as a set.

Example:

```
base_intersection([1,2,3],[2,3,5,[6,5]]);
```

4.0.5 base_memberq

base_memberq(A,S)
: It returns 1 if A is a member of the set S else returns 0.

Example:

```
base_memberq(2,[1,2,3]);
```

4.0.6 base_permutation

base_permutation(L)
: It outputs all permutations of L. BUG; it uses a slow algorithm.

Example:

```
base_permutation([1,2,3,4]);
```

4.0.7 base_position

base_position(A,S)
: It returns the position of A in S.

Example:

```
base_position("cat",["dog","cat","monkey"]);
```

4.0.8 base_prune

base_prune(A,S)
: It returns a list in which A is removed from S.

Example:

```
base_prune("cat",["dog","cat","monkey"]);
```

4.0.9 base_replace

base_replace(S,$Rule$)
: It rewrites S by using the rule $Rule$

Example:

```
base_replace(x^2+y^2,[[x,a+1],[y,b]]);
```

x is replaced by a+1 and y is replaced by b in x^2+y^2.

4.0.10 base_set_minus

base_set_minus(A,B)
: $A \setminus B$

Example:

```
base_set_minus([1,2,3],[3,4,5]);
```

4.0.11 base_set_union

base_set_union(A,B)
: $A \cup B$

Example:

```
base_set_union([1,2,3],[3,4,5]);
```

4.0.12 base_subsetq

base_subsetq(A,B)
: if $A \subseteq B$, then it returns 1 else 0.

Example:

```
base_subsetq([1,2],[1,2,3,4,5]);
```

4.0.13 base_subsets_of_size

base_subsets_of_size(K,S)
: It outputs all subsets of S of the size K. BUG; it uses a slow algorithm. Do not input a large S.

Example:

```
base_subsets_of_size(2,[3,5,3,2]);
```

5 数 (標準数学函数)

5.0.1 number_abs

number_abs(X)
:

Example:

```
number_abs(-3);
```

5.0.2 number_ceiling

number_ceiling(X)
:

Example:

```
number_abs(1.5);
```

5.0.3 number_factor

number_factor(X)
: It factors the given integer X.

Example:

```
number_factor(20);
```

5.0.4 number_floor

number_floor(X)
:

Example:

```
number_floor(1.5);
```

5.0.5 number_imaginary_part

number_imaginary_part(X)
:

Example:

```
number_imaginary_part(1+2*@i);
```

5.0.6 number_is_integer

number_is_integer(X)
:

Example:

```
number_is_integer(2/3);
```

5.0.7 number_real_part

number_real_part(X)
:

Example:

```
number_real_part(1+2*@i);
```

6 微積分 (標準数学函数)

7 級数 (標準数学函数)

8 超幾何函数 (標準数学函数)

まだ書いてない.

9 行列 (標準数学函数)

9.0.1 matrix_clone

`matrix_clone(`M`)`
: It generates the clone of the matrix M.

Example:

```
matrix_clone(matrix_list_to_matrix([[1,1],[0,1]]));
```

9.0.2 matrix_det

`matrix_det(`M`)`
: It returns the determinant of the matrix M.

Example:

```
poly_factor(matrix_det([[1,x,x^2],[1,y,y^2],[1,z,z^2]]));
```

9.0.3 matrix_diagonal_matrix

`matrix_diagonal_matrix(`L`)`
: It returns the diagonal matrix with diagonal entries L.

Example:

```
matrix_diagonal_matrix([1,2,3]);
```

References
: matrix_list_to_matrix

9.0.4 matrix_eigenavalues

`matrix_eigenavalues(`M`)`
: It returns the eigenvalues of the matrix M.

Example:

```
matrix_eigenvalues([[x,1],[0,y]]);
```

9.0.5 matrix_identity_matrix

matrix_identity_matrix(N)
: It returns the identity matrix of the size N.

Example:

```
matrix_identity_matrix(5);
```

References
matrix_diagonal_matrix

9.0.6 matrix_image

matrix_image(M)
: It computes the image of M. Redundant vectors are removed.

Example:

```
matrix_image([[1,2,3],[2,4,6],[1,0,0]]);
```

References
matrix_kernel

9.0.7 matrix_inner_product

matrix_inner_product(A,B)
: It returns the inner product of two vectors A and B.

Example:

```
matrix_inner_product([1,2],[x,y]);
```

9.0.8 matrix_inverse

matrix_inverse(M)
: It returns the inverse of the matrix M.

Example:

```
matrix_inverse([[1,2],[0,1]]);
```

9.0.9 matrix_kernel

matrix_kernel(M)
: It returns the basis of the kernel of the matrix M.

Example:

```
matrix_kernel([[1,1,1,1],[0,1,3,4]]);
```

9.0.10 matrix_list_to_matrix

matrix_list_to_matrix(M)
: It translates the list M to a matrix.

Example:

```
print_xdvi_form(matrix_list_to_matrix([[1,1],[0,2]]));
```

References
: matrix_matrix_to_list

9.0.11 matrix_matrix_to_list

matrix_matrix_to_list(M)
: It translates the matrix M to a list.

References
: matrix_list_to_matrix

9.0.12 matrix_rank

matrix_rank(M)
: It returns the rank of the matrix M.

Example:

```
matrix_rank([[1,1,1,1],[0,1,3,4]]);
```

9.0.13 matrix_solve_linear

matrix_solve_linear(M,X,B)
: It solves the system of linear equations M X = B

Example:

```
matrix_solve_linear([[1,2],[0,1]],[x,y],[1,2]);
```

9.0.14 matrix_submatrix

matrix_submatrix(*M*,*Ind*)
: It returns the submatrix of *M* defined by the index set *Ind*.

Example:

```
matrix_submatrix([[0,1],[2,3],[4,5]],[1,2]);
```

9.0.15 matrix_transpose

matrix_transpose(*M*)
: It returns the transpose of the matrix *M*.

References
: matrix_list_to_matrix

10 Graphic(標準数学函数)

10.0.16 print_dvi_form

print_dvi_form(*S*)
: It outputs *S* to a dvi file.

Example:

```
print_dvi_form(x^2-1);
```

References
print_xdvi_form , print_tex_form

10.0.17 print_em

print_em(*S*)
: It outputs *S* by a font to emphasize it.

Example:

```
print_em(x^2-1);
```

10.0.18 print_gif_form

print_gif_form(*S*)
: It outputs *S* to a file of the gif format.

print_gif_form(*S* | *table*=key0)
: This function allows optional variables *table*

Example:

```
print_gif_form(newmat(2,2,[[x^2,x],[y^2-1,x/(x-1)]]));
```

References
print_tex_form

10.0.19 print_input_form

`print_input_form(`*S*`)`
: It transforms *S* to a string which can be parsed by asir.

Example:

```
print_input_form(quote(x^3-1));
```

10.0.20 print_open_math_tfb_form

`print_open_math_tfb_form(`*S*`)`
: It transforms *S* to a tfb format of OpenMath XML.

It is experimental. You need to load taka_print_tfb.rr to call it.

Example:

```
print_open_math_tfb_form(quote(f(x,1/(y+1))+2));
```

10.0.21 print_open_math_xml_form

`print_open_math_xml_form(`*S*`)`
: It transforms *S* to a string which is compliant to OpenMath(1999).

Example:

```
print_open_math_xml_form(x^3-1);
```

References
: www.openmath.org

10.0.22 print_output

`print_output(`*Obj*`)`
: It outputs the object *Obj* to a file. If the optional variable *file* is set, then it outputs the *Obj* to the specified file, else it outputs it to "asir_output_tmp.txt". If the optional variable *mode* is set to "w", then the file is newly created. If the optional variable is not set, the *Obj* is appended to the file.

`print_output(`*Obj* `|` *file*`=key0,`*mode*`=key1)`
: This function allows optional variables *file*, *mode*

Example:

```
print_output("Hello"|file="test.txt");
```

References
glib_tops , (,)

10.0.23 print_ox_rfc100_xml_form

print_ox_rfc100_xml_form(*S*)
: It transforms *S* to a string which is compliant to OpenXM RFC 100.

Example:

```
print_ox_rfc100_xml_form(x^3-1);
```

References
www.openxm.org

10.0.24 print_png_form

print_png_form(*S*)
: It transforms *S* to a file of the format png.

Example:

```
print_png_form(x^3-1);
```

References
print_tex_form

10.0.25 print_terminal_form

print_terminal_form(*S*)
: It transforms *S* to the terminal form???

10.0.26 print_tex_form

print_tex_form(*S*)
: It transforms *S* to a string of the LaTeX format.

print_tex_form(*S* | *table*=key0)
: This function allows optional variables *table*

The global variable Print_tex_form_fraction_format takes the values "auto", "frac", or "/". The global variable Print_tex_form_no_automatic_subscript takes the values 0 or 1. BUG; A large input S cannot be translated.

Example:

```
print_tex_form(x*dx+1 | table=[["dx","\\partial_x"]]);
```

The optional variable table is used to give a translation table of asir symbols and tex symbols.

References

print_xdvi_form

10.0.27 print_tfb_form

print_tfb_form(*S*)

: It transforms *S* to the tfb format.

Example:

```
print_tfb_form(x+1);
```

10.0.28 print_xdvi_form

print_xdvi_form(*S*)

: It transforms *S* to a xdvi file and previews the file by xdvi.

Example 0:

```
print_xdvi_form(newmat(2,2,[[x^2,x],[y^2-1,x/(x-1)]]));
```

Example 1:

```
print_xdvi_form(print_tex_form(1/2));
```

References

print_tex_form , print_dvi_form

10.0.29 print_xv_form

print_xv_form(*S*)

: It transforms *S* to a gif file and previews the file by xv.

print_xv_form(*S* | *input*=key0,*format*=key1)

: This function allows optional variables *input*, *format*

Example 0:

```
print_xv_form(newmat(2,2,[[x^2,x],[y^2-1,x/(x-1)]]));
```

Example 1:

```
print_xv_form(x+y | format="png");
```

If the optional variable format="png" is set, png format will be used to generate an input for xv.

References

print_tex_form , print_gif_form

11 多項式 (標準数学函数)

11.0.1 poly_degree

poly_degree(F)
: It returns the degree of F with respect to the given weight vector.

poly_degree(F | *weight*=key0,v=key1)
: This function allows optional variables *weight*, v

The weight is given by the optional variable weight w. It returns $\mathrm{ord}_w(F)$

Example:

```
poly_degree(x^2+y^2-4 |weight=[100,1],v=[x,y]);
```

11.0.2 poly_elimination_ideal

poly_elimination_ideal(I, VV)
: It computes the ideal intersection of I and the monomial ideal generated by VV.

poly_elimination_ideal(I, VV | *grobner_basis*=key0,v=key1)
: This function allows optional variables *grobner_basis*, v

If grobner_basis is "yes", I is assumed to be a Grobner basis. The optional variable v is a list of variables which defines the ring of polynomials.

Example 0:

```
poly_elimination_ideal([x^2+y^2-4,x*y-1],[x]);
```

Example 1:

```
A = poly_grobner_basis([x^2+y^2-4,x*y-1]|order=2,v=[y,x]);
      poly_elimination_ideal(A,[x]|grobner_basis="yes");
```

References
: gr , hgr , gr_mod , dp_*

11.0.3 poly_factor

poly_factor(F)
: It factorizes the polynomial F.

Example:

```
poly_factor(x^10-y^10);
```

11.0.4 poly_gcd

poly_gcd(*F*,*G*)
: It computes the polynomial GCD of *F* and *G*.

Example:

```
poly_gcd(x^10-y^10,x^25-y^25);
```

11.0.5 poly_grobner_basis

poly_grobner_basis(*I*)
: It returns the Grobner basis of *I*.

poly_grobner_basis(*I* | *order*=key0,*v*=key1)
: This function allows optional variables *order*, *v*

The optional variable v is a list of variables which defines the ring of polynomials.

Example:

```
A = poly_grobner_basis([x^2+y^2-4,x*y-1]|order=2,v=[y,x]);
```

11.0.6 poly_hilbert_polynomial

poly_hilbert_polynomial(*I*)
: It returns the Hilbert polynomial of the ideal I.

poly_hilbert_polynomial(*I* | *s*=key0,*v*=key1)
: This function allows optional variables *s*, *v*

The optional variable v is a list of variables.

Example:

```
poly_hilbert_polynomial([x1*y1,x1*y2,x2*y1,x2*y2]|s=k,v=[x1,x2,y1,y2]);
```

11.0.7 poly_initial

poly_initial(*I*)
: It returns the initial ideal of *I* with respect to the given order.

`poly_initial(`I `|` *order*`=key0,`v`=key1)`
: This function allows optional variables *order*, v

The optional variable v is a list of variables. This function computes $\mathrm{in}_{\prec}(I)$

Example:

```
poly_initial([x^2+y^2-4,x*y-1]|order=0,v=[x,y]);
```

11.0.8 poly_initial_coefficients

`poly_initial_coefficients(`I`)`
: It computes the coefficients of the initial ideal of I with respect to the given order.

`poly_initial_coefficients(`I `|` *order*`=key0,`v`=key1)`
: This function allows optional variables *order*, v

The optional variable v is a list of variables. The order is specified by the optional variable order

Example:

```
poly_initial_coefficients([x^2+y^2-4,x*y-1]|order=0,v=[x,y]);
```

11.0.9 poly_initial_term

`poly_initial_term(`F`)`
: It returns the initial term of a polynomial F with respect to the given weight vector.

`poly_initial_term(`F `|` *weight*`=key0,`*order*`=key1,`v`=key2)`
: This function allows optional variables *weight*, *order*, v

The weight is given by the optional variable weight w. It returns $\mathrm{in}_w(F)$

Example:

```
poly_initial_term( x^2+y^2-4 |weight=[100,1],v=[x,y]);
```

11.0.10 poly_solve_linear

`poly_solve_linear(`*Eqs*`,`V`)`
: It solves the system of linear equations *Eqs* with respect to the set of variables V.

Example:

```
poly_solve_linear([2*x+3*y-z-2, x+y+z-1], [x,y,z]);
```

12 複体 (標準数学函数)

13 OpenMath 函数 (1999 版)

'om'にこの節で定義されている関数が定義されている．Java の実行環境が設定されていることが必要である．

Author of OMproxy : Yasushi Tamura , tamura@math.kobe-u.ac.jp

13.0.1 om_start

om_start()
:: OMproxy をスタートする．このサーバは CMO と OpenMath XML (CD's in 1999) との間の変換をおこなう．

return 数

```
[155] load("om");
1
[160] om_start();
control: wait OX
Trying to connect to the server... Done.
0
[161] om_xml(<<1,0>>+2*<<0,1>>);
<OMOBJ><OMA><OMS name="DMP" cd="poly"/>
<OMA><OMS name="PolyRing" cd="poly"/>
 <OMI>2</OMI></OMA><OMA>
 <OMS name="SDMP" cd="poly"/>
 <OMA><OMS name="Monom" cd="poly"/><OMI>1</OMI><OMI>1</OMI><OMI>0</OMI></OMA>
 <OMA><OMS name="Monom" cd="poly"/><OMI>2</OMI><OMI>0</OMI><OMI>1</OMI></OMA>
</OMA></OMA></OMOBJ>
[162] om_xml_to_cmo(@);
(1)*<<1,0>>+(2)*<<0,1>>
```

13.0.2 om_xml

om_xml(*s*|proc=*p*)
:: *s* の CMO 表現を OpenMath の XML (CD's in 1999) 表現になおす．

return 文字列
p 数
s オブジェクト

- *s* の CMO 表現を OpenMath の XML (CD's in 1999) 表現になおす．

```
For (I=0; I<10; I++) {
   A = 2^I;
   B = om_xml(A);
```

```
        C = om_xml_to_cmo(B);
        print(A == C);
    }
```

13.0.3 om_xml_to_cmo

om_xml_to_cmo(s|proc=p)
:: OpenMath の XML (CD's in 1999) 表現 s を CMO になおす.

return	オブジェクト
p	数
s	文字列

- OpenMath の XML (CD's in 1999) 表現 s を CMO に変換する.

14 Differential equations (library by Okutani)

ファイル ‘gr’, ‘Matrix’, ‘Diff’ が必要です.

OpenXM/Risa/Asir での利用にあたっては,

```
load("Diff")$
```

が始めに必要.

Yukio Okutani 氏による Risa/Asir 言語で書かれた連立線形偏微分方程式用のライブラリです. すべての関数名は odiff_ で始まります.

この節で紹介される関数では微分作用素はリストまたは多項式で表現されます. リストによる表現は次のようになります.

$$[[f_\alpha, [\alpha_1, \ldots, \alpha_n]], \ldots]$$

これは

$$\sum_\alpha f_\alpha \partial^\alpha$$

という意味です. 線型偏微分方程式系

$$(\sum_{\alpha^{(i)}} f_{\alpha^{(i)}} \partial^{\alpha^{(i)}}) \bullet u = 0 \quad (i = 1, \ldots, s)$$

などのように複数の微分作用素を表現するときは微分作用素のリストを使います.

$$[[[f_{\alpha^{(1)}}, [\alpha_1^{(1)}, \ldots, \alpha_n^{(1)}]], \ldots], \ldots, [[f_{\alpha^{(s)}}, [\alpha_1^{(s)}, \ldots, \alpha_n^{(s)}]], \ldots]]$$

例えば微分作用素 $x\partial_x + y\partial_y + 1$ の場合は

$$[[x, [1, 0]], [y, [0, 1]], [1, [0, 0]]]$$

となります. また微分作用素のリストで $x\partial_x + y\partial_y + 1, {\partial_x}^2 + {\partial_y}^2$ を表すと

$$[[[x, [1, 0]], [y, [0, 1]], [1, [0, 0]]], [[1, [2, 0]], [1, [0, 2]]]]$$

となります. またこの表現法を使うときは変数リストを常に意識している必要があります. 次に多項式による表現について述べます. 変数 x に対する微分は dx で表現されます. 例えば $x\partial_x + y\partial_y + 1$ については

$$x * dx + y * dy + 1$$

と表現されます.

14.0.1 odiff_op_appell4

odiff_op_appell4(*a*,*b*,*c1*,*c2*,*V*)
:: appell の F_4 を零化する微分作用素を生成します.

return リスト

a, b, c1, c2
: 有理式

V
: リスト

- odiff_op_appell4の例.

```
[298] odiff_op_appell4(a,b,c1,c2,[x,y]);
[ [ [-x^2+x,[2,0]], [-2*y*x,[1,1]], [-y^2,[0,2]],
    [(-a-b-1)*x+c1,[1,0]], [(-a-b-1)*y,[0,1]], [-b*a,[0,0]] ],
  [ [-y^2+y,[0,2]], [-2*y*x,[1,1]], [-x^2,[2,0]],
    [(-a-b-1)*y+c2,[0,1]], [(-a-b-1)*x,[1,0]], [-b*a,[0,0]] ] ]
```

14.0.2 odiff_op_tosm1

odiff_op_tosm1(LL, V)
: :: リスト形式の微分作用素リストを sm1 形式に変換します.

return
: リスト

LL
: リスト

V
: リスト

- 微分作用素の係数は整数多項式に変換されます.
- odiff_op_tosm1の例

```
[299] odiff_op_tosm1([[[x,[2,0]],[-1,[0,0]]],
                      [[y,[0,2]],[-1,[0,0]]]],[x,y]);
[ + ( + (1) x) dx^2 + ( + (-1)), + ( + (1) y) dy^2 + ( + (-1))]

[300] odiff_op_tosm1([[[x,[1,0]],[y,[0,1]],[1,[0,0]]],
                      [[1,[2,0]],[1,[0,2]]]],[x,y]);
[ + ( + (1) x) dx + ( + (1) y) dy + ( + (1)), + ( + (1)) dx^2 + ( + (1)) dy^2]

[301] odiff_op_tosm1([[[1/2,[1,0]],[1,[0,0]]],
                      [[1/3,[0,1]],[1/4,[0,0]]]],[x,y]);
[ + ( + (6)) dx + ( + (12)), + ( + (4)) dy + ( + (3))]

[302] odiff_op_tosm1([[[1/2*x,[1,0]],[1,[0,0]]],
                      [[1/3*y,[0,1]],[1/4,[0,0]]]],[x,y]);
[ + ( + (6) x) dx + ( + (12)), + ( + (4) y) dy + ( + (3))]
```

14.0.3 odiff_op_toasir

odiff_op_toasir(LL, V)
: :: リスト形式の微分作用素リスト LL を asir の多項式に変換します.

return
: リスト

LL
: リスト

V リスト

- odiff_op_toasirの例

```
[303] odiff_op_toasir([[[1/2*x,[1,0]],[1,[0,0]]],
                       [[1/3*y,[0,1]],[1/4,[0,0]]]],[x,y]);
[1/2*x*dx+1,1/3*y*dy+1/4]

[304] odiff_op_toasir([[[x,[1,0]],[y,[0,1]],[1,[0,0]]],
                       [[1,[2,0]],[1,[0,2]]]],[x,y]);
[x*dx+y*dy+1,dx^2+dy^2]
```

14.0.4 odiff_op_fromasir

odiff_op_fromasir(*D_list*,*V*)
:: asir の多項式からリスト形式の微分作用素リストに変換します.

return リスト
D_list リスト
V リスト

- odiff_op_fromasirの例

```
[305] odiff_op_fromasir([1/2*x*dx+1,1/3*y*dy+1/4],[x,y]);
[[[1/2*x,[1,0]],[1,[0,0]]],[[1/3*y,[0,1]],[1/4,[0,0]]]]

[306] odiff_op_fromasir([x*dx+y*dy+1,dx^2+dy^2],[x,y]);
[[[x,[1,0]],[y,[0,1]],[1,[0,0]]],[[1,[2,0]],[1,[0,2]]]]
```

14.0.5 odiff_act

odiff_act(*L*,*F*,*V*)
:: 微分作用素 *L* を有理式 *F* に作用させる. *V* は変数リスト.

return 有理式
L リスト or 多項式
F 有理式
V リスト

- odiff_actの例

```
[302] odiff_act([[1,[2]]],x^3+x^2+x+1,[x]);
6*x+2

[303] odiff_act([[1,[1,0]],[1,[0,1]]],x^2+y^2,[x,y]);
2*x+2*y
```

```
[349] odiff_act(x*dx+y*dy, x^2+x*y+y^2, [x,y]);
2*x^2+2*y*x+2*y^2
```

14.0.6 odiff_act_appell4

odiff_act_appell4(a,b,$c1$,$c2$,F,V)
:: 微分作用素 odiff_op_appell4 を有理式 F に作用させる.

return リスト
a, b, c1, c2
有理式
F 有理式
V リスト

- odiff_act_appell4の例

```
[303] odiff_act_appell4(1,0,1,1,x^2+y^2,[x,y]);
[-6*x^2+4*x-6*y^2,-6*x^2-6*y^2+4*y]

[304] odiff_act_appell4(0,0,1,1,x^2+y^2,[x,y]);
[-4*x^2+4*x-4*y^2,-4*x^2-4*y^2+4*y]

[305] odiff_act_appell4(-2,-2,-1,-1,x^2+y^2,[x,y]);
[0,0]
```

14.0.7 odiff_poly_solve

odiff_poly_solve(LL,N,V)
:: 与えられた線型微分方程式系の N 次以下の多項式解を求める.

return リスト
LL リスト
N 整数
V リスト

- odiff_poly_solveの例.

```
[297] odiff_poly_solve([[[x,[1,0]],[-1,[0,0]]],[[y,[0,1]],[-1,[0,0]]]],5,[x,y]);
[_4*y*x,[_4]]

[298] odiff_poly_solve([[[x,[1,0]],[-2,[0,0]]],[[y,[0,1]],[-2,[0,0]]]],5,[x,y]);
[_33*y^2*x^2,[_33]]

[356] odiff_poly_solve([x*dx+y*dy-3,dx+dy],4,[x,y]);
[-_126*x^3+3*_126*y*x^2-3*_126*y^2*x+_126*y^3,[_126]]
```

14.0.8 odiff_poly_solve_hg1

odiff_poly_solve_hg1(*a*,*b*,*c*,*V*)
:: ガウスの超幾何微分方程式の多項式解を求める.

return リスト
a, *b*, *c* 有理式
V リスト

- odiff_poly_solve_hg1の例.

```
[334] odiff_poly_solve_hg1(-3,-6,-5,[x]);
[_1*x^6-2*_0*x^3+9/2*_0*x^2-18/5*_0*x+_0,[_0,_1]]

[335] odiff_poly_solve_hg1(-3,-6,-7,[x]);
[-4/7*_2*x^3+15/7*_2*x^2-18/7*_2*x+_2,[_2]]
```

14.0.9 odiff_poly_solve_appell4

odiff_poly_solve_appell4(*a*,*b*,*c1*,*c2*,*V*)
:: F_4 がみたす線型微分方程式系の多項式解を求める.

return リスト
a, *b*, *c1*, *c2*
有理式
V リスト

- odiff_poly_solve_appell4の例.

```
[299] odiff_poly_solve_appell4(-3,1,-1,-1,[x,y]);
[-_26*x^3+(3*_26*y+_26)*x^2+3*_24*y^2*x-_24*y^3+_24*y^2,[_24,_26]]

[300] odiff_poly_solve_appell4(-3,1,1,-1,[x,y]);
[-3*_45*y^2*x-_45*y^3+_45*y^2,[_45]]
```

14.0.10 odiff_rat_solve

odiff_rat_solve(*LL*,*Dn*,*N*,*V*)
:: 与えられた線型微分方程式系の分母が *Dn*, 分子が *N* 次以下の多項式であるような解を求める.

return リスト
LL リスト
Dn 有理式
N 整数
V リスト

- odiff_rat_solveの例.

```
[333] odiff_rat_solve([[[x,[1]],[1,[0]]]],x,1,[x]);
[(_8)/(x),[_8]]

[361] odiff_rat_solve([x*(1-x)*dx^2+(1-3*x)*dx-1],1-x,2,[x]);
[(_180)/(-x+1),[_180]]

[350] D = odiff_op_appell4(0,0,3,0,[x,y])$
[351] odiff_rat_solve(D,x^2,2,[x,y]);
[(_118*x^2-_114*y*x+1/2*_114*y^2+_114*y)/(x^2),[_114,_118]]
```

15 D-module (library by Okutani)

ファイル 'gr', 'xm', 'Matrix', 'Diff', 'Dmodule' が必要です.

OpenXM/Risa/Asir での利用にあたっては,

```
load("Diff")$ load("Dmodule")$
```

が始めに必要.

Yukio Okutani 氏による D-加群計算用の sm1 サーバとのインタフェースライブラリです. すべての関数名は odmodule_ で始まります.

15.0.1 odmodule_d_op_tosm1

odmodule_d_op_tosm1(LL, V)
:: リスト形式の微分作用素リストを sm1 形式に変換します.

return	リスト
LL	リスト
V	リスト

- 微分作用素の係数は整数多項式に変換されます.
- この関数は diff_op_tosm1と等価です.
- odmodule_d_op_tosm1の例

```
[299] odmodule_d_op_tosm1([[[x,[2,0]],[-1,[0,0]]],
                           [[y,[0,2]],[-1,[0,0]]]],[x,y]);
[ + ( + (1) x) dx^2 + ( + (-1)), + ( + (1) y) dy^2 + ( + (-1))]

[300] odmodule_d_op_tosm1([[[x,[1,0]],[y,[0,1]],[1,[0,0]]],
                           [[1,[2,0]],[1,[0,2]]]],[x,y]);
[ + ( + (1) x) dx + ( + (1) y) dy + ( + (1)), + ( + (1)) dx^2 + ( + (1)) dy^2]

[301] odmodule_d_op_tosm1([[[1/2,[1,0]],[1,[0,0]]],
                           [[1/3,[0,1]],[1/4,[0,0]]]],[x,y]);
[ + ( + (6)) dx + ( + (12)), + ( + (4)) dy + ( + (3))]

[302] odmodule_d_op_tosm1([[[1/2*x,[1,0]],[1,[0,0]]],
                           [[1/3*y,[0,1]],[1/4,[0,0]]]],[x,y]);
[ + ( + (6) x) dx + ( + (12)), + ( + (4) y) dy + ( + (3))]
```

15.0.2 odmodule_d_op_toasir

odmodule_d_op_toasir(LL, V)
:: リスト形式の微分作用素リスト LL を asir の多項式に変換します.

return リスト
LL リスト
V リスト

- この関数は diff_op_toasirと等価です.
- odmodule_d_op_toasirの例

```
[303] odmodule_d_op_toasir([[[1/2*x,[1,0]],[1,[0,0]]],
                            [[1/3*y,[0,1]],[1/4,[0,0]]]],[x,y]);
[1/2*x*dx+1,1/3*y*dy+1/4]

[304] odmodule_d_op_toasir([[[x,[1,0]],[y,[0,1]],[1,[0,0]]],
                            [[1,[2,0]],[1,[0,2]]]],[x,y]);
[x*dx+y*dy+1,dx^2+dy^2]
```

15.0.3 odmodule_d_op_fromasir

odmodule_d_op_fromasir(*D_list*, *V*)
:: asir の多項式からリスト形式の微分作用素リストに変換します.

return リスト
D_list リスト
V リスト

- この関数は diff_op_fromasirと等価です.
- odmodule_d_op_fromasirの例

```
[305] odmodule_d_op_fromasir([1/2*x*dx+1,1/3*y*dy+1/4],[x,y]);
[[[1/2*x,[1,0]],[1,[0,0]]],[[1/3*y,[0,1]],[1/4,[0,0]]]]

[306] odmodule_d_op_fromasir([x*dx+y*dy+1,dx^2+dy^2],[x,y]);
[[[x,[1,0]],[y,[0,1]],[1,[0,0]]],[[1,[2,0]],[1,[0,2]]]]
```

15.0.4 odmodule_ch_ideal

odmodule_ch_ideal(*D_ideal*, *V*)
:: *D_ideal* の characteristic ideal を求めます.

return リスト
D_ideal リスト
V リスト

- *D_ideal* は generic parameter を含むことができます.
- odmodule_ch_idealの例

```
[344] odmodule_ch_ideal([x*dx+y*dy+a,dx^2+dy^2],[x,y]);
[x*dx+y*dy,dx^2+dy^2,y*dy*dx-x*dy^2,(x^2+y^2)*dy^2]

[348] odmodule_ch_ideal(diff_op_appell4(a,b,c1,c2,[x,y]),[x,y]);
[-x*dx^2+y*dy^2,2*y*x*dy*dx+(y*x+y^2-y)*dy^2,
 (2*y^2-2*y)*dy^2*dx+(-y*x+3*y^2+y)*dy^3,
 2*y*x*dy^2*dx+(y*x^2+(-2*y^2-y)*x+y^3-y^2)*dy^3]
```

15.0.5 odmodule_singular_locus

odmodule_singular_locus(*D_ideal*, *V*)

:: *D_ideal* の singular locus を求めます.

return リスト
D_ideal リスト
V リスト

- *D_ideal* は generic parameter を含むことができます.
- odmodule_singular_locusの例

```
[356] D = diff_op_appell4(a,b,c1,c2,[x,y])$
[357] odmodule_singular_locus(D,[x,y]);
[-y*x^3+(2*y^2+2*y)*x^2+(-y^3+2*y^2-y)*x]

[358] D = diff_op_hg1(a,b,c,[x])$
[359] odmodule_singular_locus(D,[x]);
[x^2-x]
```

15.0.6 odmodule_restriction

odmodule_restriction(*D_ideal*, *V*, *Rest*)

:: *D_ideal* の 0 次の restriction を求めます.

return リスト
D_ideal リスト
V リスト
Rest リスト

- *D_ideal* は generic parameter を含むことができます.
- odmodule_restrictionの例.

```
[345] odmodule_restriction([x*dx+y*dy+a,dx^2+dy^2],[x,y],[y]);
[[2,[-x*dx-a,-e0*x*dx-e0*a-e0]]]
```

15.0.7 odmodule_elimination

odmodule_elimination(*D_ideal*, *V*, *Elim*)
:: *D_ideal* の elimination ideal を求めます.

return リスト
D_ideal リスト
V リスト
Elim リスト

- *D_ideal* は generic parameter を含むことができます.
- odmodule_eliminationの例.

```
[346] odmodule_elimination([x*dx+y*dy+a,dx^2+dy^2],[x,y],[[y],[0]]);
[x^2*dx^2+(2*a+2)*x*dx+a^2+a]

[347] odmodule_elimination([x*dx+y*dy+a,dx^2+dy^2],[x,y],[[y],[b]]);
[(x^2+b^2)*dx^2+(2*a+2)*x*dx+a^2+a]
```

16 DSOLV 函数

この節は正則ホロノミック系を級数で解くための函数をあつめてある. アルゴリズムについては [SST] に説明がある. このパッケージは次のコマンド `load("dsolv");` でロードできる. このパッケージは `Diff` および `Dmodule` を使用する.

OpenXM/Risa/Asir での利用にあたっては,

```
load("dsolv");$
```

が始めに必要.

このパッケージは `ox_sm1` を利用している. したがって使用できる変数は `sm1` パッケージと同様の変数しかつかえない.

16.1 函数一覧

16.1.1 dsolv_dual

`dsolv_dual(`*f*,*v*`)`
:: *f* のグレブナ双対

戻り値　　リスト
f, *v*　　リスト

- 変数 *v* 上の多項式環において, *f* のグレブナ双対を求める.
- *f* で生成されるイデアルは, *v* で生成される極大イデアルに対して, primary でないといけない. primary でない場合, この函数は無限ループにおちいる.

Algorithm: この函数は本 [SST] の Algorithm 2.3.14 の実装である. 出力中の変数 x, y, ... をそれぞれ log(x), log(y), ..., でおきかえると, これらの log 多項式は, `f_(x->x*dx, y->y*dy, ...)` で生成される微分方程式系の解となっている.

```
[435] dsolv_dual([y-x^2,y+x^2],[x,y]);
[x,1]
[436] dsolv_act(y*dy-sm1_mul(x*dx,x*dx,[x,y]),log(x),[x,y]);
0
[437] dsolv_act(y*dy+sm1_mul(x*dx,x*dx,[x,y]),log(x),[x,y]);
0

[439] primadec([y^2-x^3,x^2*y^2],[x,y]);
[[[y^2-x^3,y^4,x^2*y^2],[y,x]]]
[440] dsolv_dual([y^2-x^3,x^2*y^2],[x,y]);
[x*y^3+1/4*x^4*y, x^2*y, x*y^2+1/12*x^4, y^3+x^3*y,
```

```
 x^2, x*y, y^2+1/3*x^3, x, y, 1]

[441] dsolv_test_dual();
  Output is  omitted.
```

16.1.2 dsolv_starting_term

dsolv_starting_term(*f*,*v*,*w*)
:: 正則ホロノミック系 *f* の方向 *w* での級数解の Staring terms を計算する. ここで, *v* は変数の集合.

戻り値　リスト
f, *v*, *w*　リスト

- 正則ホロノミック系 *f* の方向 *w* での級数解の Staring terms を計算する. ここで, *v* は変数の集合.
- 戻り値は次の形をしている: [[*e1*, *e2*, ...], [*s1*, *s2*, ...]] ここで *e1* は exponent ベクトルであり *s1* はこのベクトルに対応する解の集合, 以下同様.
- 変数 Dsolv_message_starting_term を 1 にしておくと, この函数は計算の途中にいろいろとメッセージを出力する.

Algorithm: Saito, Sturmfels, Takayama, Grobner Deformations of Hypergeometric Differential Equations ([SST]), Chapter 2.

```
[1076]   F = sm1_gkz( [ [[1,1,1,1,1],[1,1,0,-1,0],[0,1,1,-1,0]], [1,0,0]]);
[[x5*dx5+x4*dx4+x3*dx3+x2*dx2+x1*dx1-1,-x4*dx4+x2*dx2+x1*dx1,
  -x4*dx4+x3*dx3+x2*dx2,
  -dx2*dx5+dx1*dx3,dx5^2-dx2*dx4],[x1,x2,x3,x4,x5]]
[1077]  A= dsolv_starting_term(F[0],F[1],[1,1,1,1,0])$
Computing the initial ideal.
Done.
Computing a primary ideal decomposition.
Primary ideal decomposition of the initial Frobenius ideal
to the direction [1,1,1,1,0] is
[[[x5+2*x4+x3-1,x5+3*x4-x2-1,x5+2*x4+x1-1,3*x5^2+(8*x4-6)*x5-8*x4+3,
   x5^2-2*x5-8*x4^2+1,x5^3-3*x5^2+3*x5-1],
 [x5-1,x4,x3,x2,x1]]]

----------- root is [ 0 0 0 0 1 ]
----------- dual system is
[x5^2+(-3/4*x4-1/2*x3-1/4*x2-1/2*x1)*x5+1/8*x4^2
 +(1/4*x3+1/4*x1)*x4+1/4*x2*x3-1/8*x2^2+1/4*x1*x2,
 x4-2*x3+3*x2-2*x1,x5-x3+x2-x1,1]

[1078] A[0];
[[ 0 0 0 0 1 ]]
[1079] map(fctr,A[1][0]);
[[[1/8,1],[x5,1],[log(x2)+log(x4)-2*log(x5),1],
          [2*log(x1)-log(x2)+2*log(x3)+log(x4)-4*log(x5),1]],
 [[1,1],[x5,1],[-2*log(x1)+3*log(x2)-2*log(x3)+log(x4),1]],
```

```
[[1,1],[x5,1],[-log(x1)+log(x2)-log(x3)+log(x5),1]],
[[1,1],[x5,1]]]
```

17 GNUPLOT 函数

この節では GNUPLOT の ox サーバ ox_sm1_gnuplot とのインタフェース関数を解説する. これらの関数はファイル 'gnuplot' で定義されている. gnuplot は '$(OpenXM_HOME)/lib/asir-contrib/' にある.

```
nobuki@yama:~$ asir
This is Risa/Asir, Version 20020802 (Kobe Distribution).
Copyright (C) 1994-2000, all rights reserved, FUJITSU LABORATORIES LIMITED.
Copyright 2000,2001, Risa/Asir committers, http://www.openxm.org/.
GC 6.1(alpha5) copyright 2001, H-J. Boehm, A. J. Demers, Xerox, SGI, HP.
PARI 2.2.1(alpha), copyright (C) 2000,
     C. Batut, K. Belabas, D. Bernardi, H. Cohen and M. Olivier.
OpenXM/Risa/Asir-Contrib(20020804), Copyright 2000-2002, OpenXM.org
help("keyword"); ox_help(0); ox_help("keyword"); ox_grep("keyword");
     for help messages (unix version only).
[255] gnuplot.start();
0
[257] gnuplot.gnuplot("plot sin(x**2);");
0
```

関数 gnuplot.heat(dt,$step$) はわれわれの GNUPLOT インタフェース関数のデモである. この関数は熱伝導方程式

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \quad u(t,0) = u(t,1) = 1$$

を初期条件

$$u(0,x) = x,\ (0 \le x \le 0.5), \quad u(1,x) = 1 - x,\ (0.5 \le x \le 1)$$

で陽的差分法を用いて時間が $0 \le t \le dt * step$ の範囲で解く. 区間 [0,1] は Heat_N 個に分割される. static 変数 Heat_N は関数 gnuplot.set_heat_N で設定する. 有名な Courant-Friedrichs-Levi 数 dt*Heat_N*Heat_N が 0.5 以下であれば, 陽的差分スキームは安定である. CFL を変えることにより, 不安定性が生じるのを観察できる.

```
gnuplot.set_heat_N(20); gnuplot.heat(0.001,30);   (CFL number is 0.4)
gnuplot.set_heat_N(20); gnuplot.heat(0.003,30);   (CFL > 0.5 unstable)
```

Author of GNUPLOT: Thomas Williams, Colin Kelley

17.1 函数一覧

17.1.1 gnuplot.start

gnuplot.start()
:: Localhost で ox_sm1_gnuplot を起動する.

return 整数

- Localhost で ox_sm1_gnuplot を起動する. 起動された ox_sm1_gnuplot の識別番号を戻す.
- Xm_noX =1 としておくと, ox_sm1_gnuplot 用の debug window が開かない.
- 識別番号は, Gnuplot_proc に格納される.

```
P = gnuplot.start();
```

参照 ox_launch, gnuplot

17.1.2 gnuplot

gnuplot.gnuplot(*s*|proc=*p*)
:: GNUPLOT にコマンド *s* を実行してもらう.

return なし
p 数
s 文字列

- サーバは GNUPLOT のコマンド *s* を実行する. エラーがおきた場合 GNUPLOT 本体は終了してしまうが, ox_sm1_gnuplot は自動的に GNUPLOT 本体をリスタートする.
- GNUPLOT は長い多項式をただしくうけつけない.
- GNUPLOT は ^ をうけつけない. かわりに, ** を使う.

```
[232] P = gnuplot.start();
0
*Plot 3 dimensional graph.
[233] gnuplot.gnuplot("splot x**2-y**2;"|proc=P);
0
*Plot 2 dimensional graph.
[234] gnuplot.gnuplot("plot [-pi:pi] [-2:2] cos(x);");
0
*Output a graph as a postscript figure.
[235] gnuplot.output(|file="hoge.eps");
0
[236] gnuplot.gnuplot("plot sin(x)*cos(x);");
0
[237] gnuplot.gnuplot(|file="x11");
0

*Plot 3 dimensional graph hiding unvisible lines.
[236] gnuplot.gnuplot("set hidden3d");
0
[237] gnuplot.gnuplot("splot (x**2+y**2)*sin(x**2+y**2)");
0
[238] gnuplot.gnuplot("set isosamples 50");
0
[239] gnuplot.gnuplot("splot (x**2+y**2)*sin(x**2+y**2)");
```

参照 ox_launch, gnuplot.start, rtostr, gnuplot.plot_dots

参考書　矢吹道郎, 大竹つよし; 使いこなす GNUPLOT, テクノプレス, ISBN4-924998-11-7

17.1.3 gnuplot.plot_dots

gnuplot.plot_dots(*d*,*s*|proc=*p*)
:: 点の集合 *d* をスタイル *s* でプロットする.

return　なし
p　数
d　リスト
s　文字列 または 0

- 点集合 *d* をスタイル *s* でプロットする. *s* は次のような文字列: "style color point". ここで style には lines, points, linespoints, impulses, dots, steps, errorbars, boxes, boxerrorbars を選べる. color には 1 (red), 2 (green), 3 (blue), 4, ... , 8 を選べる. point は 1 から 8 の数を入れる. color, point は省略してよい.
- *d* == [ ] のときはスクリーンがまず消去される.

```
[239] P = gnuplot.start();
0
[240] gnuplot.plot_dots([ ],0);
0
[241] for (I=0; I<10; I++) gnuplot.plot_dots([[I,I^2]]," lines ");
[242] A = [ ];
[]
[243] for (I=0; I<10; I++) A = append(A,[ [I,I^2]]);
[244] A;
[[0,0],[1,1],[2,4],[3,9],[4,16],[5,25],[6,36],[7,49],[8,64],[9,81]]
[245] gnuplot.plot_dots(A," lines ");
0
```

参照　gnuplot.start, plot "fileName" with options(GNUPLOT command), gnuplot.clean, gnuplot

17.1.4 gnuplot.heat

gnuplot.heat(*dt*,*step*)
:: 熱伝導方程式を数値的に解く.

return　なし
dt　浮動小数点数
step　整数

- 熱伝導方程式 du/dt = d^2 u/dx^2, u(t,0) = u(t,1) = 0 を初期条件 u(0,x) = x (0 <= x <= 0.5), u(0,x) = 1-x (0.5 <= x <= 1.0) で解く.
- Heat_N は空間方向でのメッシュの数.

- この関数は将来 pde_heat_demo と呼ばれる予定.

```
[232] Heat_N = 20$
[233] gnuplot.heat(0.001,30)$
```

17.1.5 gnuplot.output

gnuplot.output(|*file*=*s*)
:: GNUPLOT にファイル *s* へポストスクリプトで出力するように頼む.

return Void
s String

- GNUPLOT にファイル *s* へポストスクリプトで出力するように頼む.
- s が "x11" または, この関数を引数無しでよぶと, 以後, X11 の display に graphics が出力される.

```
[273] gnuplot.output(|file="hoge.eps");
Graphic output of GNUPLOT will be written to hoge.eps as a Poscript file.
0
[274] gnuplot.gnuplot("plot tan(x)+sin(x);");
0
[275] gnuplot.output();
Usage of gnuplot.output: gnuplot.output(|file="string")
                         gnuplot.output(|file="x11")
Output device is set to X11
```

参照 gnuplot

17.1.6 gnuplot.plot_function

gnuplot.plot_function(*f*|proc=*p*)
:: gnuplot サーバに *f* のグラフを書くように頼む.

戻り値 なし
p 数
f 多項式または多項式のリスト

- gnuplot サーバに *f* のグラフを書くように頼む.

```
[290] gnuplot.plot_function((x+sin(x))^2);
0
[291] gnuplot.plot_function([x,x^2,x^3]);
0
```

参照 gnuplot.to_gnuplot_format

17.1.7 gnuplot.stop

gnuplot.stop()
:: GNUPLOT を停止し, 通信用の fifo ファイルを消す.

return Void
s String

- GNUPLOT を停止し, 一時ディレクトリの下に作成された通信用の fifo ファイルを消す.
- 通信用の fifo ファイル名は gnuplot で始まる.

```
[273] gnuplot.stop()
```

参照 gnuplot.start

17.1.8 gnuplot.setenv

gnuplot.setenv(*key*,*value*)
::

return Void
key String
value Object

- key は "gnuplot.callingMethod" または "plot.gnuplotexec".

```
   Use the old method to communicate with gnuplot (version 3).
   This method does not use mkfifo, but we need a patched version of gnuplot.
[273] gnuplot.setenv("gnuplot.callingMethod",0);
[274] gnuplot.setenv("plot.gnuplotexec",getenv("OpenXM_HOME")+"/bin/gnuplot4ox");

   Calling your own gnuplot binary.
[274] gnuplot.setenv("plot.gnuplotexec","/cygdrive/c/program files/gnuplot/pgnuplot.ex
```

参照 gnuplot.start

18 グラフィックライブラリ (2 次元)

19 Graphic Library (2 dimensional)

ライブラリ glib は, Risa/Asir の グラフィック基本関数 (draw_obj) に対する, 昔の BASIC のような単純なインタフェースを提供する.

19.0.1 glib_line

`glib_line(`*X0*`,`*Y0*`,`*X1*`,`*Y1*`)`
: It draws the line [*X0*,*Y0*]– [*X1*,*Y1*] with *color*

`glib_line(`*X0*`,`*Y0*`,`*X1*`,`*Y1* `|` *color*`=key0)`
: This function allows optional variables *color*

Example:

```
glib_line(0,0,5,3/2 | color=0xff00ff);
```

19.0.2 glib_open

`glib_open()`
: It starts the ox_plot server and opens a canvas. The canvas size is set to *Glib_canvas_x* X *Glib_canvas_y* (the default value is 400). This function is automatically called when the user calls glib functions.

19.0.3 glib_plot

`glib_plot(`*F*`)`
: It plots an object *F* on the glib canvas.

Example 0:

```
glib_plot([[0,1],[0.1,0.9],[0.2,0.7],[0.3,0.5],[0.4,0.8]]);
```

Example 1:

```
glib_plot(tan(x));
```

19.0.4 glib_print

`glib_print(`*X*`,`*Y*`,`*Text*`)`
: It put a string *Text* at [*X*,*Y*] on the glib canvas.

glib_print(*X*,*Y*,*Text* | *color*=key0)
: This function allows optional variables *color*

Example:

```
glib_print(100,100,"Hello Worlds" | color=0xff0000);
```

19.0.5 glib_ps_form

glib_ps_form(*S*)
: It returns the PS code generated by executing *S* (experimental).

Example 0:

```
glib_ps_form(quote( glib_line(0,0,100,100) ));
```

Example 1:

```
glib_ps_form(quote([glib_line(0,0,100,100),glib_line(100,0,0,100)]));
```

References
: glib_tops

19.0.6 glib_putpixel

glib_putpixel(*X*,*Y*)
: It puts a pixel at [*X*,*Y*] with *color*

glib_putpixel(*X*,*Y* | *color*=key0)
: This function allows optional variables *color*

Example:

```
glib_putpixel(1,2 | color=0xffff00);
```

19.0.7 glib_tops

glib_tops()
: If Glib_ps is set to 1, it returns a postscript program to draw the picture on the canvas.

References
: print_output

19.0.8 `glib_window`

`glib_window(`*Xmin*`,`*Ymin*`,`*Xmax*`,`*Ymax*`)`

: It generates a window with the left top corner [*Xmin*,*Ymin*] and the right bottom corner [*Xmax*,*Ymax*]. If the global variable *Glib_math_coordinate* is set to 1, mathematical coordinate system will be employed, i.e., the left top corner will have the coordinate [*Xmin*,*Ymax*].

Example:

```
glib_window(-1,-1,10,10);
```

20 Mathematica 函数

この節では Mathematica の ox サーバ ox_math とのインタフェース関数を解説する. これらの関数はファイル 'm' で定義されているのでこのファイルを load("m")$ でこのファイルをロードしてから使用しないといけない. 'm' は '$(OpenXM_HOME)/lib/asir-contrib' にある.

注意: ox_reset は動かない.

```
nobuki@yama:~$ asir
This is Risa/Asir, Version 20020802 (Kobe Distribution).
Copyright (C) 1994-2000, all rights reserved, FUJITSU LABORATORIES LIMITED.
Copyright 2000,2001, Risa/Asir committers, http://www.openxm.org/.
GC 6.1(alpha5) copyright 2001, H-J. Boehm, A. J. Demers, Xerox, SGI, HP.
PARI 2.2.1(alpha), copyright (C) 2000,
     C. Batut, K. Belabas, D. Bernardi, H. Cohen and M. Olivier.
OpenXM/Risa/Asir-Contrib(20020804), Copyright 2000-2002, OpenXM.org
help("keyword"); ox_help(0); ox_help("keyword"); ox_grep("keyword");
     for help messages (unix version only).
[258] load("m")$
m Version 19991113. mathematica.start, mathematica.tree_to_string, mathematica.n_Eigen
[259] mathematica.start();
ox_math has started.
ox_math: Portions copyright 2000 Wolfram Research, Inc.
See OpenXM/Copyright/Copyright.mathlink  for details.
0
[260] mathematica.n_Eigenvalues([[1,2],[4,5]]);
[-0.464102,6.4641]
```

Mathematica is the trade mark of Wolfram Research Inc. This package requires Mathmatica Version 3.0, so you need Mathematica to make this package work. See http://www.wolfram.com. The copyright and license agreement of the mathlink is put at OpenXM/Copyright/Copyright.mathlink Note that the licence prohibits to connect to a mathematica kernel via the internet.

Author of ox_math: Katsuyoshi Ohara, ohara@air.s.kanazawa-u.ac.jp.

20.1 函数一覧

20.1.1 mathematica.start

mathematica.start()

:: Localhost で ox_math を起動する.

return 整数

- Localhost で ox_math を起動する. 起動された ox_math の識別番号を戻す.
- Xm_noX =1 としておくと, ox_math 用の debug window が開かない.

- 識別番号は M_proc に格納される.

```
P = mathematica.start()
```

参照 ox_launch

20.1.2 mathematica.tree_to_string

mathematica.tree_to_string(t)
:: ox_math の戻す Mathematica の木構造データ t を asir 形式になおす.

return 文字列
t リスト

- t は ox_math の戻す Mathematica の木構造データ.
- ox_math の戻す Mathematica の木構造データ t を asir 形式になおす.
- t をなるべく asir が理解できる形での, 前置または中置記法の文字列に変換する. t の先頭要素の文字列がキーワードであるが, その文字が変換テーブルにないときは, m_ をキーワードの先頭につけて, 関数呼出形式の文字列へかえる.

```
[267] mathematica.start();
0
[268] ox_execute_string(0,"Expand[(x-1)^2]");
0
[269] A=ox_pop_cmo(0);
[Plus,1,[Times,-2,x],[Power,x,2]]
[270] mathematica.tree_to_string(A);
(1)+((-2)*(x))+((x)^(2))
[271] eval_str(@);
x^2-2*x+1

[259] mathematica.tree_to_string(["List",1,2]);
[1 , 2]
[260] mathematica.tree_to_string(["Plus",2,3]);
(2)+(3)
[261] mathematica.tree_to_string(["Complex",2.3,4.55]);
mathematica.complex(2.3 , 4.55)
[362] mathematica.tree_to_string(["Plus",["Complex",1.2,3.5],1/2]);
(mathematica.complex(1.2 , 3.5))+(1/2)
[380] eval_str(@);
(1.7+3.5*@i)
```

参照 ox_pop_cmo, eval_str, mathematica.rtomstr

20.1.3 mathematica.rtomstr

mathematica.rtomstr(t)
:: t をなるべく Mathematica の理解可能な文字列に変える.

return 文字列
t オブジェクト

- *t* をなるべく Mathematica が理解できる形の文字列に変換する. たとえば, asir ではリストを [,] で囲むが, Mathematica では {, } で囲む. この関数はこの変換をおこなう.

```
[259] mathematica.rtomstr([1,2,3]);
{1,2,3}
[260] mathematica.rtomstr([[1,x,x^2],[1,y,y^2]]);
{{1,x,x^2},{1,y,y^2}}
```

もう一つ例をあげよう. 次の関数 mathematica.inverse(M) は ox_math をよんで行列 M の逆行列を計算する関数である. mathematica.inverse(M) は次のように r_tostr(M) を用いて asir の行列を Mathematica 形式に変換してから ox_execute_string で Mathematica に逆行列を計算させている.

```
def inverse(M) {
  P = 0;
  A = mathematica.rtomstr(M);
  ox_execute_string(P,"Inverse["+A+"]");
  B = ox_pop_cmo(B);
  C = mathematica.tree_to_string(B);
  return(eval_str(C));
}

[269] M=[[1,x,x^2],[1,y,y^2],[1,z,z^2]];
[[1,x,x^2],[1,y,y^2],[1,z,z^2]]
[270] A=mathematica.inverse(M)$
[271] red(A[0][0]);
(z*y)/(x^2+(-y-z)*x+z*y)
```

参照 ox_execute_string, ToExpression(Mathematica), mathematica.tree_to_string

21 OpenXM-Contrib 一般函数

21.1 函数一覧

21.1.1 ox_check_errors2

ox_check_errors2(p)
:: 識別番号 p のサーバのスタック上にあるエラーオブジェクトをリストで戻す.

return リスト
p 数

- 識別番号 p のサーバのスタック上にあるエラーオブジェクトをリストで戻す.
- エラーオブジェクトのポップはしない.

```
[219] P=sm1.start();
0
[220] sm1.sm1(P," 0 get ");
0
[221] ox_check_errors2(P);
[error([7,4294967295,executeString: Usage:get])]
Error on the server of the process number = 1
To clean the stack of the ox server,
type in ox_pops(P,N) (P: process number, N: the number of data you need to
pop)
out of the debug mode.
If you like to automatically clean data on the server stack,
set XM_debug=0;
```

22 OXshell の関数

OXshell はシステムのコマンドを ox server より実行する仕組みである. 詳しくは OpenXM/src/kan96xx/Doc/oxshe
および OpenXM/doc/Papers/rims-2003-12-16-ja.tex を見よ.

22.0.1 oxshell.get_value

oxshell.get_value(*NAME*, *V*)
: It get the value of the variable *NAME* on the server ox_shell.

Example:

```
oxshell.set_value("abc","Hello world!");
          oxshell.oxshell(["cp", "stringIn://abc", "stringOut://result"]);
          oxshell.get_value("result");
```

References
: oxshell.oxshell , oxshell.set_value

22.0.2 oxshell.oxshell

oxshell.oxshell(*L*)
: It executes command *L* on a ox_shell server. *L* must be an array. The result is the outputs to stdout and stderr.

Example:

```
oxshell.oxshell(["ls"]);
```

References
: ox_shell , oxshell.set_value , oxshell.get_value

22.0.3 oxshell.set_value

oxshell.set_value(*NAME*, *V*)
: It set the value *V* to the variable *Name* on the server ox_shell.

Example:

```
oxshell.set_value("abc","Hello world!");
          oxshell.oxshell(["cat", "stringIn://abc"]);
```

References
: oxshell.oxshell , oxshell.get_value

23 pFq に関する (コ) ホモロジー

この節では超幾何関数 pFq ($_pF_q$) の (コ) ホモロジ群に関連した不変量を計算する関数を解説する.

OpenXM/Risa/Asir での利用にあたっては,

```
load("pfpcoh.rr")$ load("pfphom.rr")$
```

が始めに必要.

23.0.1 pfp_omega

pfp_omega(P)

: It returns the Gauss-Manin connection Omega for the generalized hypergeometric function P F P-1 (aa1,aa2, ...; cc1, cc2, ...;x) .

Define a vector valued function Y of which elements are generalized hypergeometric function f_1=F and f_2=xdf_1/dx, f3=xd f_2/dx, ... It satisfies dY/dx= Omega Y. Generalized hypergeometric function is defined by the series p F p-1(aa1,aa2, ...; cc1, cc2, ...;x) = sum(k=0,infty; (aa1)_k (aa2)_k .../((1)_k (cc1)_k (cc2)_k ...) x^k)

Example:

```
pfp_omega(3);
```

23.0.2 pfpcoh_intersection

pfpcoh_intersection(P)

: pfpcoh_intersection(P) returns an intersection matrix for cocycles associated to the generalized hypergeometric function p F_(p-1).

This program pfpcoh.rr computes an intersection matrix S of cocycles of p F p-1 and compares it with the matrix obtained by solving a differential equation for intersection matrix.

Algorithm: Ohara, Sugiki, Takayama, Quadratic Relations for Generalized Hypergeometric Functions p F p-1

Example:

```
load("pfpcoh.rr")$
S=pfpcoh_intersection(3);
```

Author : K.Ohara

23.0.3 pfphom_intersection

pfphom_intersection(P)
: intersection matrix of homology cycles.

Computing intersection matrix of cycles associated to p F_(p-1). As to the meaning of parameters c1, c2, c3, ..., see the paper Ohara, Kyushu J. Math. Vol. 51 PP.123.

Algorithm: Ohara, Sugiki, Takayama, Quadratic Relations for Generalized Hypergeometric Functions p F p-1

Example:

```
SS = pfphom_intersection(3)$
```

You get the intersection matrix of homologies for 3 F 2.

Author : K.Ohara

23.0.4 pfphom_monodromy_pair_kyushu

pfphom_monodromy_pair_kyushu(P)
:

It returns the pair of monodromy matrices.

Algorithm: Ohara, Kyushu J. Math. Vol.51 PP.123 (1997)

Example:

```
MP = pfphom_monodromy_pair_kyushu(3)$
```

You get a pair of monodromy matricies for 3F2 standing for two paths encircling 0 and 1.

24 PHC 函数

この節では PHC pack の ox サーバ ox_sm1_phc とのインタフェース関数を解説する. これらの関数はファイル 'phc' で定義されている. phc は '$(OpenXM_HOME)/lib/asir-contrib' にある.

```
nobuki@yama:~$ asir
This is Risa/Asir, Version 20020802 (Kobe Distribution).
Copyright (C) 1994-2000, all rights reserved, FUJITSU LABORATORIES LIMITED.
Copyright 2000,2001, Risa/Asir committers, http://www.openxm.org/.
GC 6.1(alpha5) copyright 2001, H-J. Boehm, A. J. Demers, Xerox, SGI, HP.
PARI 2.2.1(alpha), copyright (C) 2000,
     C. Batut, K. Belabas, D. Bernardi, H. Cohen and M. Olivier.
OpenXM/Risa/Asir-Contrib(20020804), Copyright 2000-2002, OpenXM.org
help("keyword"); ox_help(0); ox_help("keyword"); ox_grep("keyword");
     for help messages (unix version only).
[255] phc.start();
0
[257] phc.phc([x^2+y^2-4,x*y-1]);
The detailed output is in the file tmp.output.*
The answer is in the variable Phc.
0
[260] Phc ;
[[[-0.517638,0],[-1.93185,0]],
[[1.93185,0],[0.517638,0]],
[[-1.93185,0],[-0.517638,0]],
[[0.517638,0],[1.93185,0]]]
[261]
```

Author of PHC pack: Jan Verschelde.
参考書 1: Jan Verschelde, PHCpack: A general-purpose solver for polynomial systems by homotopy continuation". ACM Transaction on Mathematical Softwares, 25(2): 251-276, 1999.
参考書 2: Cox, D., O'Shea, Little, J., Using Algebraic Geometry, Springer. Mixed volumes についての章を見よ.

24.1 函数一覧

24.1.1 phc.start

phc.start()
:: Localhost で ox_sm1_phc を起動する.

return 整数

- Localhost で ox_sm1_phc を起動する. 起動された ox_sm1_phc の識別番号を戻す.
- Xm_noX =1 としておくと, ox_sm1_phc 用の debug window が開かない.
- 識別番号は Phc_proc に格納される.

```
P = phc.start()
```

参照 ox_launch, phc

24.1.2 phc.phc

phc.phc(*s*|proc=*p*)
:: PHC pack に代数方程式系 *s* の解をすべてもとめてくれるように頼む.

return なし
p 数
s リスト

- 代数方程式系 *S* をホモトピー法で解くために PHC pack を呼ぶ. PHC pack を開発したのは Jan Verschelde である. オリジナルの配布元は www.mth.msu.edu/~jan である. PHC pack は代数方程式系を解くためにいろいろな戦略をえらぶことができるが, このインタフェース関数では, black-box solver しか用いていない. black-box solver は一般的な戦略ではあるが, 能率的ではない. この関数で代数方程式を解くのに失敗したら, オリジナルの PHC pack を用い, ほかの戦略を試してみるとよい.
- PHC は作業ファイル tmp.phc.out.pid, tmp.input.*, tmp.output.* を生成する. ここで pid は ox_sm1_phc のプロセス番号である. ファイル tmp.output.* には PHC pack がどのように方程式系を解いたのかの詳しい情報がはいっている.
- 変数の数と方程式の数 length(*s*) は等しくないといけない.

Algorithm: Jan Verschelde, PHCpack: A general-purpose solver for polynomial systems by homotopy continuation". ACM Transaction on Mathematical Softwares, 25(2): 251-276, 1999.

```
[232] P = phc.start();
0
[233] phc.phc([x^2+y^2-4,x*y-1]|proc=P);
The detailed output is in the file tmp.output.*
The answer is in the variable Phc.
0
[234] Phc;
[[[-1.93185,0],[-0.517638,0]],
 [[0.517638,0],[1.93185,0]],
 [[-0.517638,0],[-1.93185,0]],
 [[1.93185,0],[0.517638,0]]]

 [[x=[real, imaginary], y=[real,imaginary]],  the first solution
  [x=[real, imaginary], y=[real,imaginary]],  the second solution
  ...
```

参照 ox_launch, phc.start, '$(OpenXM_HOME)/bin/lin_phcv2'(original PHC pack binary for linux)

25 Plucker 関係式

25.0.1 plucker

$(m+1) \times n$ 行列を考える. $i_1, \ldots, i_m, j_k$ 列をならべてつくった正方行列式を $p_{i_1 \cdots i_m j_k}$ と書くとき, Plücker の関係式は

$$\sum_{k=0}^{m+1} (-1)^k p_{i_1 \cdots i_m j_k} p_{j_0 \cdots \hat{} j_k \cdots j_{m+1}} = 0$$

と書ける. このパッケージでは, この Plücker の関係式を扱うための関数を提供する.

25.0.2 plucker_relation

plucker_relation(L,M)
:: Index 集合 L, M に対応する Plucker 関係式を戻す.

return	quote
L	リスト
M	リスト

- L には, Plucker 関係式の $i_1, \ldots, i_m$ を, M には, Plucker 関係式の $j_0, \ldots, j_{m+1}$ を与える.

```
[297] A = plucker_relation([1,2],[3,4,5,6]);
quote(y_1_2_3*y_4_5_6-y_1_2_4*y_3_5_6+y_1_2_5*y_3_4_6-y_1_2_6*y_3_4_5)
[298] eval_str(print_terminal_form(A));
y_4_5_6*y_1_2_3-y_3_5_6*y_1_2_4+y_3_4_6*y_1_2_5-y_3_4_5*y_1_2_6
```

25.0.3 plucker_y

plucker_y(L)
:: Index 集合 L に対応する変数を戻す.

return	変数
L	リスト

- Index 集合 L は小さい順にソートされる. このとき符号もともに計算される.

```
[297] plucker_y([1,2,3]);
y_1_2_3

[298] plucker_y([2,1,3]);
-y_1_2_3
```

25.0.4 plucker_index

`plucker_index(`V`)`
: It gets the index of the variable V.

Example:

```
plucker_index(plucker_y([1,2,3]));
```

26 SM1 函数

この節では sm1 の ox サーバ ox_sm1_forAsir とのインタフェース関数を解説する. これらの関数はファイル 'sm1' で定義されている. 'sm1' は '$(OpenXM_HOME)/lib/asir-contrib' にある. システム sm1 は微分作用素環で計算するためのシステムである. 計算代数幾何のいろいろな不変量の計算が微分作用素の計算に帰着する. sm1 についての文書は OpenXM/doc/kan96xx にある.

とこに断りがないかぎりこの節のすべての関数は, 有理数係数の式を入力としてうけつけない. すべての多項式の係数は整数でないといけない.

空間 $X := \mathbf{C} \setminus \{0,1\} = \mathbf{C} \setminus V(x(x-1))$ のドラームコホモロジ群達の次元を計算してみよう. X は平面に二つの穴をあけた空間であるので, 点 $x=0$, $x=1$ のまわりをまわる二つのループが 1 次元のホモロジー群の空間をはる. したがって, 1 次元ドラームコホモロジ群の次元は 2 である. sm1 は 0 次元のコホモロジ群の次元および 1 次元のコホモロジ群の次元を答える.

```
nobuki@yama:~$ asir
This is Risa/Asir, Version 20020802 (Kobe Distribution).
Copyright (C) 1994-2000, all rights reserved, FUJITSU LABORATORIES LIMITED.
Copyright 2000,2001, Risa/Asir committers, http://www.openxm.org/.
GC 6.1(alpha5) copyright 2001, H-J. Boehm, A. J. Demers, Xerox, SGI, HP.
PARI 2.2.1(alpha), copyright (C) 2000,
     C. Batut, K. Belabas, D. Bernardi, H. Cohen and M. Olivier.
OpenXM/Risa/Asir-Contrib(20020804), Copyright 2000-2002, OpenXM.org
help("keyword"); ox_help(0); ox_help("keyword"); ox_grep("keyword");
     for help messages (unix version only).

[283] sm1.deRham([x*(x-1),[x]]);
[1,2]
```

The author of sm1 : Nobuki Takayama, takayama@math.sci.kobe-u.ac.jp
The author of sm1 packages : Toshinori Oaku, oaku@twcu.ac.jp
Reference: [SST] Saito, M., Sturmfels, B., Takayama, N., Grobner Deformations of Hypergeometric Differential Equations, 1999, Springer. See the appendix.

26.1 ox_sm1_forAsir サーバ

26.1.1 ox_sm1_forAsir

ox_sm1_forAsir
:: asir のための sm1 サーバ.

- サーバ ox_sm1_forAsir は asir よりコマンド sm1.start で起動される sm1 サーバである.
  標準的設定では,
  ox_sm1_forAsir = '$(OpenXM_HOME)/lib/sm1/bin/ox_sm1' + '$(OpenXM_HOME)/lib/sm1/callsm1.sm1' (macro file)
  + '$(OpenXM_HOME)/lib/sm1/callsm1b.sm1' (macro file)

であり, これらのマクロファイルは, 一般には current directory, $(LOAD_SM1_PATH), $(OpenXM_HOME)/lib/sm1, /usr/local/lib/sm1 の順番でさがされる.

- プログラマーのためのノート: sm1 マクロを読み込んで自分独自のサーバを作るには次のファイルも見よ '$(OpenXM_HOME)/src/kxx/oxserver00.c', '$(OpenXM_HOME)/src/kxx/sm1stackmachine.c'

26.2 函数一览

26.2.1 sm1.start

sm1.start()

:: localhost で ox_sm1_forAsir をスタートする.

return 整数

- localhost で ox_sm1_forAsir をスタートする. サーバ ox_sm1_forAsir の識別番号を戻す.
- Xm_noX = 1 とおくとサーバ ox_sm1_forAsir をデバッグ用のウィンドウなしに起動できる.
- コマンド ord を用いて変数順序を正しく設定しておく必要がある. たとえば, 変数 x と dx 上の微分作用素環 (dx は $\partial/\partial x$ に対応) で計算しているとき, sm1 サーバは式を印刷したとき, 変数 dx は右側に集めれ変数 x は左側にあつめられていると仮定している. 次の例では, 変数 cc を sm1 での計算のために用いてはいけない.
- a より z のなかで, d と o を除いたもの, それから, x0, ..., x20, y0, ..., y20, z0, ..., z20 は, デフォールトで微分作用素環の変数として使える (cf. Sm1_ord_list in sm1).
- 識別番号は static Sm1_proc に格納される. この識別番号は関数 sm1.get_Sm1_proc() でとりだすことができる.

```
[260] ord([da,a,db,b]);
[da,a,db,b,dx,dy,dz,x,y,z,dt,ds,t,s,u,v,w,
......... omit ..................
]
[261] a*da;
a*da
[262] cc*dcc;
dcc*cc
[263] sm1.mul(da,a,[a]);
a*da+1
[264] sm1.mul(a,da,[a]);
a*da
```

参照 ox_launch, sm1.push_int0, sm1.push_poly0, ord

26.2.2 sm1.sm1

sm1.sm1(p,s)

:: サーバ sm1 にコマンド列 s を実行してくれるようにたのむ.

return なし
p 数
s 文字列

- 識別番号 p の sm1 サーバにコマンド列 s を実行してくれるように頼む. (次の例では, 識別番号 0)

```
[261] sm1.sm1(0," ( (x-1)^2 ) . ");
0
[262] ox_pop_string(0);
x^2-2*x+1
[263] sm1.sm1(0," [(x*(x-1))  [(x)]] deRham ");
0
[264] ox_pop_string(0);
[1 , 2]
```

参照 sm1.start, ox_push_int0, sm1.push_poly0, sm1.get_Sm1_proc().

26.2.3 sm1.push_int0

sm1.push_int0(p,f)
:: オブジェクト f を識別子 p のサーバへ送る.

return なし
p 数
f オブジェクト

- type(f) が 2 (再帰多項式) のとき, f は文字列 (type == 7) に変換されて, ox_push_cmo を用いてサーバへ送られる.
- type(f) が 0 (zero) のときは, サーバ上では, 32 bit 整数と解釈される. なお ox_push_cmo(P,0) はサーバに対して CMO_NULL をおくるので, サーバ側では, 32 bit 整数を受け取るわけではない.
- sm1 の整数は, 32 bit 整数と bignum にわけることができる. type(f) が 1 (数) のとき, この関数は 32 bit integer をサーバにおくる. ox_push_cmo(p,1234) は bignum の 1234 を sm1 サーバにおくることに注意しよう.
- その他の場合には ox_push_cmo をデータ型の変換なしに呼び出す.

```
[219] P=sm1.start();
0
[220] sm1.push_int0(P,x*dx+1);
0
[221] A=ox_pop_cmo(P);
x*dx+1
[223] type(A);
7   (string)

[271] sm1.push_int0(0,[x*(x-1),[x]]);
0
[272] ox_execute_string(0," deRham ");
0
```

```
[273] ox_pop_cmo(0);
[1,2]
```

Reference

ox_push_cmo

26.2.4 sm1.gb

sm1.gb([f,v,w]|proc=p,sorted=q,dehomogenize=r)

:: v 上の微分作用素環において f のグレブナ基底を計算する.

sm1.gb_d([f,v,w]|proc=p)

:: v 上の微分作用素環において f のグレブナ基底を計算する. 結果を分散多項式のリストで戻す.

return リスト
p, q, r 数
f, v, w リスト

- v 上の微分作用素環において f のグレブナ基底を計算する.
- Weight ベクトル w は省略してよい. 省略した場合, graded reverse lexicographic order をつかってブレブナ基底を計算する.
- sm1.gb の戻り値は f のグレブナ基底およびイニシャルモノミアル (w がないとき) または イニシャル多項式 (w が与えらたとき) のリストである.
- sm1.gb_d は結果を分散多項式のリストで戻す. 多項式の中に現れるモノミアルはグレブナ基底を計算するときに与えらた順序でソートされている. 戻り値は [変数名のリスト, 順序をきめる行列, グレブナ基底, イニシャルモノミアルまたはイニシァル多項式] である.
- Term order でない順序が与えられた場合は, 同次化ワイル代数でグレブナ基底が計算される (SST の本の Section 1.2 を見よ). 同次化変数 h が結果に加わる.
- オプショナル変数 q がセットされているときは, 3 番目の戻り値として, グレブナ基底およびイニシァルのリストが与えられた順序でソートされたモノミアルの和として戻される. いまのところこの多項式は, 文字列で表現される. オプショナル変数 r がセットされているときは, 戻り多項式は dehomogenize される (すなわち h に 1 が代入される).

```
[293] sm1.gb([[x*dx+y*dy-1,x*y*dx*dy-2],[x,y]]);
[[x*dx+y*dy-1,y^2*dy^2+2],[x*dx,y^2*dy^2]]
```

上の例において, 集合 $\{x\partial_x + y\partial_y - 1, y^2\partial_y^2 + 2\}$ は $1 \leq \partial_y \leq \partial_x \leq y \leq x \leq \cdots$ であるような graded reverse lexicographic order に関するグレブナ基底である. 集合 $\{x\partial_x, y^2\partial_y\}$ はグレブナ基底の各元に対する leading monomial (initial monomial) である.

```
[294] sm1.gb([[dx^2+dy^2-4,dx*dy-1],[x,y],[[dx,50,dy,2,x,1]]]);
[[dx+dy^3-4*dy,-dy^4+4*dy^2-1],[dx,-dy^4]]
```

上の例において二つのモノミアル $m = x^a y^b \partial_x^c \partial_y^d$ および $m' = x^{a'} y^{b'} \partial_x^{c'} \partial_y^{d'}$ は最初に weight vector (dx,dy,x,y) = (50,2,1,0) を用いて比較される (つまり m は $50c + 2d + a > 50c' + 2d' + a'$ のとき m' より大きい) 次にこの比較で勝負がつかないときは reverse lexicographic order で比較される (つまり $50c + 2d + a = 50c' + 2d' + a'$ のとき reverse lexicographic order で比較される).

```
[294] F=sm1.gb([[dx^2+dy^2-4,dx*dy-1],[x,y],[[dx,50,dy,2,x,1]]]|sorted=1);
      map(print,F[2][0])$
      map(print,F[2][1])$

[595]
   sm1.gb([["dx*(x*dx +y*dy-2)-1","dy*(x*dx + y*dy -2)-1"],
            [x,y],[[dx,1,x,-1],[dy,1]]]);

[[x*dx^2+(y*dy-h^2)*dx-h^3,x*dy*dx+y*dy^2-h^2*dy-h^3,h^3*dx-h^3*dy],
 [x*dx^2+(y*dy-h^2)*dx,x*dy*dx+y*dy^2-h^2*dy-h^3,h^3*dx]]

[596]
   sm1.gb_d([["dx (x dx +y dy-2)-1","dy (x dx + y dy -2)-1"],
             "x,y",[[dx,1,x,-1],[dy,1]]]);
[[[e0,x,y,H,E,dx,dy,h],
 [[0,-1,0,0,0,1,0,0],[0,0,0,0,0,0,1,0],[1,0,0,0,0,0,0,0],
  [0,1,1,1,1,1,1,0],[0,0,0,0,0,0,-1,0],[0,0,0,0,0,-1,0,0],
  [0,0,0,0,-1,0,0,0],[0,0,0,-1,0,0,0,0],[0,0,-1,0,0,0,0,0],
  [0,0,0,0,0,0,0,1]]],
[[(1)*<<0,0,1,0,0,1,1,0>>+(1)*<<0,1,0,0,0,2,0,0>>+(-1)*<<0,0,0,0,0,1,0,2>>+(-
1)*
<<0,0,0,0,0,0,0,3>>,(1)*<<0,0,1,0,0,0,2,0>>+(1)*<<0,1,0,0,0,1,1,0>>+(-1)*<<0,0,0
,0,0,0,1,2>>+(-1)*<<0,0,0,0,0,0,0,3>>,(1)*<<0,0,0,0,0,1,0,3>>+(-1)*<<0,0,0,0,0,0
,1,3>>],
 [(1)*<<0,0,1,0,0,1,1,0>>+(1)*<<0,1,0,0,0,2,0,0>>+(-1)*<<0,0,0,0,0,1,0,2>>,(1)*<
<0,0,1,0,0,0,2,0>>+(1)*<<0,1,0,0,0,1,1,0>>+(-1)*<<0,0,0,0,0,0,1,2>>+(-1)*<<0,0,0
,0,0,0,0,3>>,(1)*<<0,0,0,0,0,1,0,3>>]]]
```

参照　　sm1.reduction, sm1.rat_to_p

26.2.5 sm1.deRham

sm1.deRham([f, v]|proc=p)

:: 空間 C^n - (the zero set of f=0) のドラームコホモロジ群の次元を計算してくれるようにサーバに頼む.

return　　リスト
p　　数
f　　文字列 または 多項式
v　　リスト

- この函数は空間 X = C^n \ V(f) のドラームコホモロジ群の次元を計算する. すなわち, [dim H^0(X,C), dim H^1(X,C), dim H^2(X,C), ..., dim H^n(X,C)] を戻す.
- v は変数のリスト. n = length(v) である.
- sm1.deRham は計算機の資源を大量に使用する. たとえば sm1.deRham(0,[x*y*z*(x+y+z-1)*(x-y),[x,y,z]]) の計算すらすでに非常に大変である.
- b-関数の根を効率よく解析するには, ox_asir が ox_sm1_forAsir より使用されるべきである. コマンド
  sm1(0,"[(parse) (oxasir.sm1) pushfile] extension"); を用いて, ox_asir との通信モジュー

ルをあらかじめロードしておくとよい. このコマンドは ox_asir_forAsir のスタート時に自動的に実行されている.

- sm1.deRham を ox_reset(sm1.get_Sm1_proc()); で中断すると, 以後 sm1 サーバが非標準モードに入り予期しない動作をする場合があるので, コマンド ox_shutdown(sm1.get_Sm1_proc()); で, ox_sm1_forAsir を一時 shutdown してリスタートした方が安全である.

```
[332] sm1.deRham([x^3-y^2,[x,y]]);
[1,1,0]
[333] sm1.deRham([x*(x-1),[x]]);
[1,2]
```

参照 sm1.start, deRham (sm1 command)

Algorithm:

Oaku, Takayama, An algorithm for de Rham cohomology groups of the complement of an affine variety via D-module computation, Journal of pure and applied algebra 139 (1999), 201–233.

26.2.6 sm1.hilbert

sm1.hilbert([*f*,*v*]|proc=*p*)

:: 多項式の集合 *f* のヒルベルト多項式を計算する.

hilbert_polynomial(*f*,*v*)

:: 多項式の集合 *f* のヒルベルト多項式を計算する.

return 多項式

p 数

f, *v* リスト

- 多項式の集合 *f* の変数 *v* にかんするヒルベルト多項式 h(k) を計算する.
- h(k) = dim_Q F_k/I \cap F_k ここで F_k は次数が k 以下であるような多項式の集合である. I は多項式の集合 *f* で生成されるイデアルである.
- sm1.hilbert にかんするノート: 効率よく計算するには *f* はモノミアルの集合にした方がいい. 実際, この函数はまず *f* のグレブナ基底を計算し, それからその initial monomial 達のヒルベルト多項式を計算する. したがって, 入力 *f* がすでにグレブナ基底だとこの函数のなかでもう一度グレブナ基底の計算がおこなわれる. これは時間の無駄であるし, sm1 の多項式グレブナ基底計算は asir より遅い.

```
[346] load("katsura")$
[351] A=hilbert_polynomial(katsura(5),[u0,u1,u2,u3,u4,u5]);
32
```

```
[279] load("katsura")$
[280] A=gr(katsura(5),[u0,u1,u2,u3,u4,u5],0)$
[281] dp_ord();
0
[282] B=map(dp_ht,map(dp_ptod,A,[u0,u1,u2,u3,u4,u5]));
[(1)*<<1,0,0,0,0,0>>,(1)*<<0,0,0,2,0,0>>,(1)*<<0,0,1,1,0,0>>,(1)*<<0,0,2,0,0,0>>,
```

```
 (1)*<<0,1,1,0,0,0>>,(1)*<<0,2,0,0,0,0>>,(1)*<<0,0,0,1,1,1>>,(1)*<<0,0,0,1,2,0>>,
 (1)*<<0,0,1,0,2,0>>,(1)*<<0,1,0,0,2,0>>,(1)*<<0,1,0,1,1,0>>,(1)*<<0,0,0,0,2,2>>,
  (1)*<<0,0,1,0,1,2>>,(1)*<<0,1,0,0,1,2>>,(1)*<<0,1,0,1,0,2>>,(1)*<<0,0,0,0,3,1>>,
  (1)*<<0,0,0,0,4,0>>,(1)*<<0,0,0,0,1,4>>,(1)*<<0,0,0,1,0,4>>,(1)*<<0,0,1,0,0,4>>,
 (1)*<<0,1,0,0,0,4>>,(1)*<<0,0,0,0,0,6>>]
[283] C=map(dp_dtop,B,[u0,u1,u2,u3,u4,u5]);
[u0,u3^2,u3*u2,u2^2,u2*u1,u1^2,u5*u4*u3,u4^2*u3,u4^2*u2,u4^2*u1,u4*u3*u1,
 u5^2*u4^2,u5^2*u4*u2,u5^2*u4*u1,u5^2*u3*u1,u5*u4^3,u4^4,u5^4*u4,u5^4*u3,
 u5^4*u2,u5^4*u1,u5^6]
[284] sm1.hilbert([C,[u0,u1,u2,u3,u4,u5]]);
32
```

参照 sm1.start, sm1.gb, longname

26.2.7 sm1.genericAnn

sm1.genericAnn([*f*,*v*]|proc=*p*)

:: *f*^s のみたす微分方程式全体をもとめる. *v* は変数のリストである. ここで, s は *v*[0] であり, *f* は変数 rest(*v*) 上の多項式である.

return リスト
p 数
f 多項式
v リスト

- この函数は, *f*^s のみたす微分方程式全体をもとめる. *v* は変数のリストである. ここで, s は *v*[0] であり, *f* は変数 rest(*v*) 上の多項式である.

```
[595] sm1.genericAnn([x^3+y^3+z^3,[s,x,y,z]]);
[-x*dx-y*dy-z*dz+3*s,z^2*dy-y^2*dz,z^2*dx-x^2*dz,y^2*dx-x^2*dy]
```

参照 sm1.start

26.2.8 sm1.wTensor0

sm1.wTensor0([*f*,*g*,*v*,*w*]|proc=*p*)

:: *f* と *g* の D-module としての 0 次テンソル積を計算する.

return リスト
p 数
f, g, v, w リスト

- *f* と *g* の D-加群としての 0 次テンソル積を計算する.
- *v* は変数のリストである. *w* は weight のリストである. 整数 *w*[i] は変数 *v*[i] の weight である.
- sm1.wTensor0 は ox_sm1 の wRestriction0 をよんでいる. wRestriction0 は, generic な weight ベクトル *w* をもとにして制限を計算している. Weight ベクトル *w* が generic でないと計算がエラーで停止する.

- F および G を f と g それぞれの解とする. 直観的にいえば, 0 次のテンソル積は 関数 FG のみたす微分方程式系である.
- 入力 f, g が D の左イデアルであっても, 一般に, 出力は自由加群 D^r の部分加群である.

```
[258]  sm1.wTensor0([[x*dx -1, y*dy -4],[dx+dy,dx-dy^2],[x,y],[1,2]]);
[[-y*x*dx-y*x*dy+4*x+y],[5*x*dx^2+5*x*dx+2*y*dy^2+(-2*y-6)*dy+3],
 [-25*x*dx+(-5*y*x-2*y^2)*dy^2+((5*y+15)*x+2*y^2+16*y)*dy-20*x-8*y-15],
 [y^2*dy^2+(-y^2-8*y)*dy+4*y+20]]
```

26.2.9 sm1.reduction

`sm1.reduction([`f`,`g`,`v`,`w`]|proc=`p`)`
::

return リスト
f 多項式
g, v, w リスト
p 数 (ox_sm1 のプロセス番号)

- この函数は f を homogenized ワイル代数において, 多項式集合 g で簡単化 (reduce) する; つまり, この函数は, f に割算アルゴリズムを適用する. 変数集合は v で指定する. w は順序を指定するための ウエイトベクトルであり, 省略してもよい. `sm1.reduction_noH` は, Weyl algebra 用.
- 戻り値は次の形をしている: [r,c0,[c1,...,cm],g] ここで g=[g1, ..., gm] であり, c0 f + c1 g1 + ... + cm gm = r がなりたつ. r/c0 が normal form である.
- この函数は, 低次項にあらわれる reducible な項も簡単化する.
- 函数 sm1.reduction_d(P,F,G) および sm1.reduction_noH_d(P,F,G) は, 分散多項式用である.

```
[259] sm1.reduction([x^2+y^2-4,[y^4-4*y^2+1,x+y^3-4*y],[x,y]]);
[x^2+y^2-4,1,[0,0],[y^4-4*y^2+1,x+y^3-4*y]]
[260] sm1.reduction([x^2+y^2-4,[y^4-4*y^2+1,x+y^3-4*y],[x,y],[[x,1]]]);
[0,1,[-y^2+4,-x+y^3-4*y],[y^4-4*y^2+1,x+y^3-4*y]]
```

参照 `sm1.start`, `d_true_nf`

26.2.10 sm1.xml_tree_to_prefix_string

`sm1.xml_tree_to_prefix_string(`s`|proc=`p`)`
:: XML で書かれた OpenMath の木表現 s を前置記法になおす.

return String
p Number
s String

- XML で書かれた OpenMath の木表現 s を前置記法になおす.
- この函数は om_* に将来移すべきである.

- om_xml_to_cmo(OpenMath Tree Expression) は CMO_TREE を戻す. asir はこの CMO をまだサポートしていない.
- java の実行環境が必要. (たとえば, /usr/local/jdk1.1.8/bin をコマンドサーチパスに入れるなど.)

```
[263] load("om");
1
[270] F=om_xml(x^4-1);
control: wait OX
Trying to connect to the server... Done.
<OMOBJ><OMA><OMS name="plus" cd="basic"/><OMA>
<OMS name="times" cd="basic"/><OMA>
<OMS name="power" cd="basic"/><OMV name="x"/><OMI>4</OMI></OMA>
<OMI>1</OMI></OMA><OMA><OMS name="times" cd="basic"/><OMA>
<OMS name="power" cd="basic"/><OMV name="x"/><OMI>0</OMI></OMA>
<OMI>-1</OMI></OMA></OMA></OMOBJ>
[271] sm1.xml_tree_to_prefix_string(F);
basic_plus(basic_times(basic_power(x,4),1),basic_times(basic_power(x,0),-1))
```

参照 om_*, OpenXM/src/OpenMath, eval_str

26.2.11 sm1.syz

sm1.syz([*f*,*v*,*w*]|proc=*p*)
:: *v* 上の微分作用素環において *f* の syzygy を計算する.

return リスト
p 数
f, *v*, *w* リスト

- 戻り値は次の形をしている: [*s*,[*g*, *m*, *t*]]. ここで *s* は *f* の *v* を変数とする微分作用素環における syzygy である. *g* は *f* の weight vector *w* に関するグレブナ基底である. *m* は入力行列 *f* をグレブナ基底 *g* へ変換する行列である. *t* はグレブナ基底 *g* の syzygy である. まとめると, 次の等式がなりたつ: $g = m\,f$, $s\,f = 0$.
- Weight ベクトル *w* は省略してよい. 省略した場合, graded reverse lexicographic order をつかってブレブナ基底を計算する.
- Term order でない順序が与えられた場合は, 同次化ワイル代数でグレブナ基底が計算される (SST の本の Section 1.2 を見よ). 同次化変数 h が結果に加わる.

```
[293] sm1.syz([[x*dx+y*dy-1,x*y*dx*dy-2],[x,y]]);
[[[y*x*dy*dx-2,-x*dx-y*dy+1]],    generators of the syzygy
 [[[x*dx+y*dy-1],[y^2*dy^2+2]],   grobner basis
  [[1,0],[y*dy,-1]],              transformation matrix
 [[y*x*dy*dx-2,-x*dx-y*dy+1]]]]

[294]sm1.syz([[x^2*dx^2+x*dx+y^2*dy^2+y*dy-4,x*y*dx*dy-1],[x,y],[[dx,-1,x,1]]]);
[[[y*x*dy*dx-1,-x^2*dx^2-x*dx-y^2*dy^2-y*dy+4]], generators of the syzygy
 [[[x^2*dx^2+h^2*x*dx+y^2*dy^2+h^2*y*dy-4*h^4],[y*x*dy*dx-h^4], GB
  [h^4*x*dx+y^3*dy^3+3*h^2*y^2*dy^2-3*h^4*y*dy]],
 [[1,0],[0,1],[y*dy,-x*dx]],     transformation matrix
```

```
[[y*x*dy*dx-h^4,-x^2*dx^2-h^2*x*dx-y^2*dy^2-h^2*y*dy+4*h^4]]]
```

26.2.12 sm1.mul

sm1.mul(*f*,*g*,*v*|proc=*p*)
:: sm1 サーバ に *f* かける *g* を *v* 上の微分作用素環でやってくれるように頼む.

return	多項式またはリスト
p	数
f, g	多項式またはリスト
v	リスト

- sm1 サーバ に *f* かける *g* を *v* 上の微分作用素環でやってくれるように頼む.
- sm1.mul_h は homogenized Weyl 代数用.
- BUG: sm1.mul(p0*dp0,1,[p0]) は dp0*p0+1 を戻す. d 変数が後ろにくるような変数順序がはいっていないと, この関数は正しい答えを戻さない.

```
[277] sm1.mul(dx,x,[x]);
x*dx+1
[278] sm1.mul([x,y],[1,2],[x,y]);
x+2*y
[279] sm1.mul([[1,2],[3,4]],[[x,y],[1,2]],[x,y]);
[[x+2,y+4],[3*x+4,3*y+8]]
```

26.2.13 sm1.distraction

sm1.distraction([*f*,*v*,*x*,*d*,*s*]|proc=*p*)
:: sm1 に *f* の distraction を計算してもらう.

return	リスト
p	数
f	多項式
v,*x*,*d*,*s*	リスト

- 識別子 *p* の sm1 サーバに, *f* の distraction を *v* 上の微分作用素環で計算してもらう.
- *x* , *d* は, それぞれ, distract すべき x 変数, d 変数のリスト. Distraction したら, *s* を変数として結果を表す.
- Distraction というのは x*dx を x で置き換えることである. 詳しくは Saito, Sturmfels, Takayama : Grobner Deformations of Hypergeometric Differential Equations の page 68 を見よ.

```
[280] sm1.distraction([x*dx,[x],[x],[dx],[x]]);
x
[281] sm1.distraction([dx^2,[x],[x],[dx],[x]]);
x^2-x
[282] sm1.distraction([x^2,[x],[x],[dx],[x]]);
x^2+3*x+2
```

```
[283] fctr(@);
[[1,1],[x+1,1],[x+2,1]]
[284] sm1.distraction([x*dx*y+x^2*dx^2*dy,[x,y],[x],[dx],[x]]);
(x^2-x)*dy+x*y
```

参照 distraction2(sm1),

26.2.14 sm1.gkz

sm1.gkz([A,B]|proc=p)
:: 行列 A とパラメータ B に付随した GKZ 系 (A-hypergeometric system) をもどす.

return リスト
p 数
A, B リスト

- 行列 A とパラメータ B に付随した GKZ 系 (A-hypergeometric system) をもどす.

```
[280] sm1.gkz([  [[1,1,1,1],[0,1,3,4]],  [0,2] ]);
[[x4*dx4+x3*dx3+x2*dx2+x1*dx1,4*x4*dx4+3*x3*dx3+x2*dx2-2,
 -dx1*dx4+dx2*dx3,-dx2^2*dx4+dx1*dx3^2,dx1^2*dx3-dx2^3,-dx2*dx4^2+dx3^3],
 [x1,x2,x3,x4]]
```

26.2.15 sm1.appell1

sm1.appell1(a|proc=p)
:: F_1 または F_D に対応する方程式系を戻す.

return リスト
p 数
a リスト

- Appell の関数 F_1 および その n 変数化である Lauricella の関数 F_D(a,b1,b2,...,bn,c;x1,...,xn) のみたす微分方程式系を戻す. ここで, a =(a,c,b1,...,bn). パラメータは有理数でもよい.
- sm1 の関数 appell1 をよぶわけでないので, パラメータが有理数や文字式の場合も正しく動く.

```
[281] sm1.appell1([1,2,3,4]);
[[((-x1+1)*x2*dx1-3*x2)*dx2+(-x1^2+x1)*dx1^2+(-5*x1+2)*dx1-3,
  (-x2^2+x2)*dx2^2+((-x1*x2+x1)*dx1-6*x2+2)*dx2-4*x1*dx1-4,
  ((-x2+x1)*dx1+3)*dx2-4*dx1],       equations
 [x1,x2]]                            the list of variables

[282] sm1.gb(@);
```

```
[[((-x2+x1)*dx1+3)*dx2-4*dx1,((-x1+1)*x2*dx1-3*x2)*dx2+(-x1^2+x1)*dx1^2
  +(-5*x1+2)*dx1-3,(-x2^2+x2)*dx2^2+((-x2^2+x1)*dx1-3*x2+2)*dx2
  +(-4*x2-4*x1)*dx1-4,
  (x2^3+(-x1-1)*x2^2+x1*x2)*dx2^2+((-x1*x2+x1^2)*dx1+6*x2^2
 +(-3*x1-2)*x2+2*x1)*dx2-4*x1^2*dx1+4*x2-4*x1],
 [x1*dx1*dx2,-x1^2*dx1^2,-x2^2*dx1*dx2,-x1*x2^2*dx2^2]]

[283] sm1.rank(sm1.appell1([1/2,3,5,-1/3]));
3

[285] Mu=2$ Beta = 1/3$
[287] sm1.rank(sm1.appell1([Mu+Beta,Mu+1,Beta,Beta,Beta]));
4
```

26.2.16 sm1.appell4

sm1.appell4(*a*|proc=*p*)

:: F_4 または F_C に対応する方程式系を戻す.

return リスト
p 数
a リスト

- Appell の関数 F_4 および その n 変数化である Lauricella の関数 F_C(a,b,c1,c2,...,cn;x1,...,xn) のみたす微分方程式系を戻す. ここで, *a* =(a,b,c1,...,cn). パラメータは有理数でもよい.
- sm1 の関数 appell1 をよぶわけでないので, パラメータが有理数や文字式の場合も正しく動く.

```
[281] sm1.appell4([1,2,3,4]);
  [[-x2^2*dx2^2+(-2*x1*x2*dx1-4*x2)*dx2+(-x1^2+x1)*dx1^2+(-4*x1+3)*dx1-2,
  (-x2^2+x2)*dx2^2+(-2*x1*x2*dx1-4*x2+4)*dx2-x1^2*dx1^2-4*x1*dx1-2],
                                                          equations
    [x1,x2]]                                  the list of variables

[282] sm1.rank(@);
4
```

26.2.17 sm1.rank

sm1.rank(*a*|proc=*p*)

:: 微分方程式系 *a* の holonomic rank を戻す.

return 数
p 数

a リスト

- 微分方程式系 *a* の, generic point での正則解の次元を戻す. この次元を, holonomic rank と呼ぶ.
- *a* は微分作用素のリストと変数のリストよりなる.
- *a* が regular holonomic のときは `sm1.rrank` も holonomic rank を戻す. いっぱんにこの関数の方が `sm1.rank` より早い.

```
[284]  sm1.gkz([  [[1,1,1,1],[0,1,3,4]],  [0,2] ]);
[[x4*dx4+x3*dx3+x2*dx2+x1*dx1,4*x4*dx4+3*x3*dx3+x2*dx2-2,
  -dx1*dx4+dx2*dx3, -dx2^2*dx4+dx1*dx3^2,dx1^2*dx3-dx2^3,-dx2*dx4^2+dx3^3],
 [x1,x2,x3,x4]]
[285] sm1.rrank(@);
4

[286]  sm1.gkz([  [[1,1,1,1],[0,1,3,4]],  [1,2]]);
[[x4*dx4+x3*dx3+x2*dx2+x1*dx1-1,4*x4*dx4+3*x3*dx3+x2*dx2-2,
 -dx1*dx4+dx2*dx3,-dx2^2*dx4+dx1*dx3^2,dx1^2*dx3-dx2^3,-dx2*dx4^2+dx3^3],
 [x1,x2,x3,x4]]
[287] sm1.rrank(@);
5
```

26.2.18 `sm1.auto_reduce`

`sm1.auto_reduce(`*s*`|proc=`*p*`)`
:: フラグ "AutoReduce" を *s* に設定.

戻り値 数
p 数
s 数

- *s* が 1 のとき, 以後計算されるグレブナ基底はすべて, reduced グレブナ基底となる.
- *s* が 0 のとき, 計算されるグレブナ基底は reduced グレブナ基底とはかぎらない. こちらがデフォールト.

26.2.19 `sm1.slope`

`sm1.slope(`*ii*`,`*v*`,`*f_filtration*`,`*v_filtration*`|proc=`*p*`)`
:: 微分方程式系 *ii* の slope を戻す.

return 数
p 数
ii リスト (方程式)
v リスト (変数)
f_filtration リスト (weight vector)

v_filtration
: リスト (weight vector)

- `sm1.slope` は微分方程式系 *ii* の V filtration *v_filtration* で指定する超平面に沿っての (geomeric) slope を計算する.
- *v* は変数のリスト.
- 戻り値は, リストを成分とするリストである. 成分リストの第 1 要素が slope, 第 2 要素は, その weight vector に対応する microcharacteristic variety が bihomogeneous でない.

Algorithm: "A.Assi, F.J.Castro-Jimenez and J.M.Granger, How to calculate the slopes of a D-module, Compositio Math, 104, 1-17, 1996" をみよ. Slope の本来の定義では, 符号が負となるが, このプログラムは, Slope の絶対値を戻す.

```
[284] A= sm1.gkz([  [[1,2,3]],  [-3] ]);


[285] sm1.slope(A[0],A[1],[0,0,0,1,1,1],[0,0,-1,0,0,1]);

[286] A2 = sm1.gkz([ [[1,1,1,0],[2,-3,1,-3]], [1,0]]);
     (* This is an interesting example given by Laura Matusevich,
        June 9, 2001 *)

[287] sm1.slope(A2[0],A2[1],[0,0,0,0,1,1,1,1],[0,0,0,-1,0,0,0,1]);
```

参照 `sm.gb`

26.2.20 `sm1.ahg`

`sm1.ahg`(*A*)
: It idential with sm1.gkz(*A*).

26.2.21 `sm1.bfunction`

`sm1.bfunction`(*F*)
: It computes the global b-function of *F*.

It no longer calls sm1's original bfunction. Instead, it calls asir "bfct".

Algorithm: M.Noro, Mathematical Software, icms 2002, pp.147–157.

Example:

```
sm1.bfunction(x^2-y^3);
```

26.2.22 sm1.call_sm1

sm1.call_sm1(F)
: It executes F on the sm1 server. See also sm1.

26.2.23 sm1.ecart_homogenize01Ideal

sm1.ecart_homogenize01Ideal(A)
: It (0,1)-homogenizes the ideal $A[0]$. Note that it is not an elementwise homogenization.

Example:

```
input1
  F=[(1-x)*dx+1]$ FF=[F,"x,y"]$
  sm1.ecart_homogenize01Ideal(FF);
intput2
  F=sm1.appell1([1,2,3,4]);
  sm1.ecart_homogenize01Ideal(F);
```

26.2.24 sm1.ecartd_gb

sm1.ecartd_gb(A)
: It returns a standard basis of A by using a tangent cone algorithm. h[0,1](D)-homogenization is used. If the option rv="dp" (return_value="dp") is given, the answer is returned in distributed polynomials.

Example:

```
input1
  F=[2*(1-x-y)*dx+1,2*(1-x-y)*dy+1]$
  FF=[F,"x,y",[[dx,1,dy,1],[x,-1,y,-1]]]$
  sm1.ecartd_gb(FF);
output1
  [[(-2*x-2*y+2)*dx+h,(-2*x-2*y+2)*dy+h],[(-2*x-2*y+2)*dx,(-2*x-2*y+2)*dy]]
input2
  F=[2*(1-x-y)*dx+h,2*(1-x-y)*dy+h]$
  FF=[F,"x,y",[[dx,1,dy,1],[x,-1,y,-1,dx,1,dy,1]],["noAutoHomogenize",1]]$
  sm1.ecartd_gb(FF);
```

26.2.25 sm1.ecartd_gb_oxRingStructure

sm1.ecartd_gb_oxRingStructure()
: It returns the oxRingStructure of the most recent ecartd_gb computation.

26.2.26 sm1.ecartd_isSameIdeal_h

sm1.ecartd_isSameIdeal_h(*F*)

: Here, *F*=[*II*,*JJ*,*V*]. It compares two ideals *II* and *JJ* in h[0,1](D)_alg.

Example:

```
input
  II=[(1-x)^2*dx+h*(1-x)]$ JJ = [(1-x)*dx+h]$
  V=[x]$
  sm1.ecartd_isSameIdeal_h([II,JJ,V]);
```

26.2.27 sm1.ecartd_reduction

sm1.ecartd_reduction(*F*,*A*)

: It returns a reduced form of *F* in terms of *A* by using a tangent cone algorithm. h[0,1](D)-homogenization is used.

Example:

```
input
  F=[2*(1-x-y)*dx+h,2*(1-x-y)*dy+h]$
  FF=[F,"x,y",[[dx,1,dy,1],[x,-1,y,-1]]]$
  sm1.ecartd_reduction(dx+dy,FF);
```

26.2.28 sm1.ecartd_reduction_noh

sm1.ecartd_reduction_noh(*F*,*A*)

: It returns a reduced form of *F* in terms of *A* by using a tangent cone algorithm. h[0,1](D)-homogenization is NOT used. *A*[0] must not contain the variable h.

Example:

```
F=[2*(1-x-y)*dx+1,2*(1-x-y)*dy+1]$
  FF=[F,"x,y",[[dx,1,dy,1],[x,-1,y,-1]]]$
  sm1.ecartd_reduction_noh(dx+dy,FF);
```

26.2.29 sm1.ecartd_syz

sm1.ecartd_syz(*A*)

: It returns a syzygy of *A* by using a tangent cone algorithm. h[0,1](D)-homogenization is used. If the option rv="dp" (return_value="dp") is given, the answer is returned in

distributed polynomials. The return value is in the format [s,[g,m,t]]. s is the generator of the syzygies, g is the Grobner basis, m is the translation matrix from the generators to g. t is the syzygy of g.

Example:

```
input1
  F=[2*(1-x-y)*dx+1,2*(1-x-y)*dy+1]$
  FF=[F,"x,y",[[dx,1,dy,1],[x,-1,y,-1]]]$
  sm1.ecartd_syz(FF);
 input2
  F=[2*(1-x-y)*dx+h,2*(1-x-y)*dy+h]$
  FF=[F,"x,y",[[dx,1,dy,1],[x,-1,y,-1,dx,1,dy,1]],["noAutoHomogenize",1]]$
  sm1.ecartd_syz(FF);
```

26.2.30 sm1.gb_oxRingStructure

`sm1.gb_oxRingStructure()`
: It returns the oxRingStructure of the most recent gb computation.

26.2.31 sm1.gb_reduction

`sm1.gb_reduction(`F`,`A`)`
: It returns a reduced form of F in terms of A by using a normal form algorithm. h[1,1](D)-homogenization is used.

Example:

```
input
  F=[2*(h-x-y)*dx+h^2,2*(h-x-y)*dy+h^2]$
  FF=[F,"x,y",[[dx,1,dy,1],[x,-1,y,-1,dx,1,dy,1]]]$
  sm1.gb_reduction((h-x-y)^2*dx*dy,FF);
```

26.2.32 sm1.gb_reduction_noh

`sm1.gb_reduction_noh(`F`,`A`)`
: It returns a reduced form of F in terms of A by using a normal form algorithm.

Example:

```
input
  F=[2*dx+1,2*dy+1]$
  FF=[F,"x,y",[[dx,1,dy,1]]]$
  sm1.gb_reduction_noh((1-x-y)^2*dx*dy,FF);
```

26.2.33 sm1.generalized_bfunction

`sm1.generalized_bfunction(`I, V, VD, W`)`

: It computes the generalized b-function (indicial equation) of I with respect to the weight W.

It no longer calls sm1's original function. Instead, it calls asir "generic_bfct".

Example:

```
sm1.generalized_bfunction([x^2*dx^2-1/2,dy^2],[x,y],[dx,dy],[-1,0,1,0]);
```

26.2.34 sm1.isSameIdeal_in_Dalg

`sm1.isSameIdeal_in_Dalg(`I, J, V`)`

: It compares two ideals I and J in D_alg (algebraic D with variables V, no homogenization).

Example:

```
Input1
  II=[(1-x)^2*dx+(1-x)]$ JJ = [(1-x)*dx+1]$ V=[x]$
  sm1.isSameIdeal_in_Dalg(II,JJ,V);
```

26.2.35 sm1.restriction

`sm1.restriction(`I, V, R`)`

: It computes the restriction of I as a D-module to the set defined by R. V is the list of variables. When the optional variable *degree*=d is given, only the restrictions from 0 to d are computed. Note that, in case of vector input, RESTRICTION VARIABLES MUST APPEAR FIRST in the list of variable V. We are using wbfRoots to get the roots of b-functions, so we can use only generic weight vector for now.

`sm1.restriction(`I, V, R `| degree=key0)`

: This function allows optional variables *degree*

Algorithm: T.Oaku and N.Takayama, math.AG/9805006, http://xxx.langl.gov

Example:

```
sm1.restriction([dx^2-x,dy^2-1],[x,y],[y]);
```

26.2.36 sm1.saturation

`sm1.saturation(`T`)`

: $T = [I,J,V]$. It returns saturation of I with respect to J^infty. V is a list of variables.

Example:

```
sm1.saturation([[x2^2,x2*x4, x2, x4^2], [x2,x4], [x2,x4]]);
```

27 TIGERS 函数

この章では, tigers ox server `ox_sm1_tigers` にたいするインタフェース函数を説明する.

27.0.1 `tigers.tigers`

`tigers.tigers(`*a*`|proc=`*a*`)`

:: この函数は識別子 p の tigers サーバに行列 a に付随したトーリックイデアルのすべてのグレブナ基底を計算してくれるようにたのむ.

戻り値	リスト
p	数
a	リスト

- この函数は識別子 p の tigers サーバに行列 a に付随したトーリックイデアルのすべてのグレブナ基底を計算してくれるようにたのむ.
- Tigers は アフィントーリックイデアルの reduced グレブナ基底をすべて数えあげるための専用のプログラムである. このプログラムは, アフィントーリックイデアルの state polytope をきめるために使える. 理論的なバックグラウンドについては, 本
  B.Sturmfels, Grobner bases and Convex Polytopes
  を見よ. Tigers は Birk Hubert が作者である. このプログラムの利用しているアルゴリズムは
  B.Huber and R.Thomas, Computing Grobner Fans of Toric Ideals
  に説明されている.

```
[395] A=[[1,1,1,1],[0,1,2,3]]$
[306] S=tigers.tigers(A)$
[307] length(S);
8
[308] S[0];
[[[1,0,1,0],[0,2,0,0]],[[1,0,0,1],[0,1,1,0]],[[0,1,0,1],[0,0,2,0]]]
[309] S[1];
[[[1,0,0,1],[0,1,1,0]],[[0,2,0,0],[1,0,1,0]],[[0,1,0,1],[0,0,2,0]]]
```

この例では, A に付随したアフィントーリックイデアルのすべてのグレブナ基底が S に格納される. この例では, 8 個のグレブナ基底がある. [[i_1, i_2, ...],[j_1, j_2, ...]] は二つのモノミアルの exponent をならべたものであり, 2 項式をあらわす. たとえば, S[0] は次の多項式の集合
x1 x3 - x2^2, x1 x4 - x2 x3, x2 x4 - x3^2
であり, <x1 x3, x1 x4, x2 x4> がその initial ideal である.

28 便利な関数

システムの資源にアクセスするためおよび文字列処理の便利な関数を集めてある.

28.0.1 util_filter

util_filter(*Command*,*Input*)
: It executes the filter program *Command* with the *Input* and returns the output of the filter as a string.

util_filter(*Command*,*Input* | *env*=key0)
: This function allows optional variables *env*

Example:

```
util_filter("sort","cat\ndog\ncentipede\n");
```

28.0.2 util_find_and_replace

util_find_and_replace(*W*,*S*,*Wnew*)
: It replaces *W* in *S* by *Wnew*. Arguments must be a list of ascii codes.

28.0.3 util_find_substr

util_find_substr(*W*,*S*)
: It returns the position of *W* in *S*. If *W* cannot be found, it returns -1. Arguments must be a list of ascii codes.

28.0.4 util_index

util_index(*V*)
: It returns the name part and the index part of *V*.

Example:

```
util_index(x_2_3)
```

References
: util_v

28.0.5 util_load_file_as_a_string

util_load_file_as_a_string(F)
: It reads a file F as a string.

28.0.6 util_part

util_part(S,P,Q)
: It returns from Pth element to Qth element of S.

28.0.7 util_read_file_as_a_string

util_read_file_as_a_string(F)
: It reads a file F as a string.

28.0.8 util_remove_cr

util_remove_cr(S)
: It removes cr/lf/tabs from S. Arguments must be a list of ascii codes.

28.0.9 util_timing

util_timing(Q)
: Show the timing data to execute Q.

Example:

```
util_timing( quote( fctr(x^50-y^50) ));
```

28.0.10 util_v

util_v(V,L)
: It returns a variable indexed by L.

Example:

```
util_v("x",[1,3]);
```

References
: util_index

28.0.11 util_write_string_to_a_file

util_write_string_to_a_file(*Fname*,*S*)
: It writes a string *S* to a file *Fname*.

29 その他

この節ではまだ分類がおわっていない関数を解説する．この節の関数は将来は別の独立した節へ移す予定である．

29.0.1 todo_parametrize

パッケージ todo_parametrize/todo_parametrize.rr をロードすることにより, 有理曲線のパラメータ表示を見付ける関数である, paramerize が利用できるようになる．詳しくは todo_parametrize_ja.texi を見よ．このパッケージのマニュアルへの統合はまだできていない．このパッケージはまだ module 構造を利用していないので, 既存のライブラリと名前の衝突の可能性がある．

```
[1205] load("todo_parametrize/todo_parametrize.rr");
1
[1425] parametrize(y^2-x^3);
[155*t^2+20*t+1,720*t^4+1044*t^3+580*t^2,155*t^4+20*t^3+t^2,(-x)/(y)]
[1426] parametrize(y^2+x^3);
[-t,1,t^3,(-x)/(y)]
```

索引

(Index is nonexistent)

(Index is nonexistent)

目次（抄）

1 はじめに 1
2 Asir Contrib の函数名について 3
3 Windows 版 Asir-contrib 4
4 基礎 (標準函数) 5
5 数 (標準数学函数) 8
6 微積分 (標準数学函数) 10
7 級数 (標準数学函数) 11
8 超幾何函数 (標準数学函数) 12
9 行列 (標準数学函数) 13
10 Graphic(標準数学函数) 17
11 多項式 (標準数学函数) 22
12 複体 (標準数学函数) 25
13 OpenMath 函数 (1999 版) 26
14 Differential equations (library by Okutani) 28
15 D-module (library by Okutani) 34
16 DSOLV 函数 38
17 GNUPLOT 函数 41
18 グラフィックライブラリ (2 次元) 46
19 Graphic Library (2 dimensional) 47
20 Mathematica 函数 50
21 OpenXM-Contrib 一般函数 53
22 OXshell の関数 54
23 pFq に関する (コ) ホモロジー 55
24 PHC 函数 57
25 Plucker 関係式 59
26 SM1 函数 61
27 TIGERS 函数 80
28 便利な関数 81
29 その他 84
索引 85

目次

1 はじめに 1

2 Asir Contrib の函数名について 3

3 Windows 版 Asir-contrib 4

4 基礎 (標準函数) 5

4.0.1 base_cancel 5
4.0.2 base_choose 5
4.0.3 base_flatten 5
4.0.4 base_intersection 5
4.0.5 base_memberq 6
4.0.6 base_permutation 6
4.0.7 base_position 6
4.0.8 base_prune 6
4.0.9 base_replace 6
4.0.10 base_set_minus 7
4.0.11 base_set_union 7
4.0.12 base_subsetq 7
4.0.13 base_subsets_of_size 7

5 数 (標準数学函数) 8

5.0.1 number_abs 8
5.0.2 number_ceiling 8
5.0.3 number_factor 8
5.0.4 number_floor 8
5.0.5 number_imaginary_part 9
5.0.6 number_is_integer 9
5.0.7 number_real_part 9

6 微積分 (標準数学函数) 10

7 級数 (標準数学函数) 11

8 超幾何函数 (標準数学函数) 12

9 行列 (標準数学函数) 13

9.0.1 matrix_clone 13
9.0.2 matrix_det 13
9.0.3 matrix_diagonal_matrix 13
9.0.4 matrix_eigenavalues 13
9.0.5 matrix_identity_matrix 14
9.0.6 matrix_image 14
9.0.7 matrix_inner_product 14
9.0.8 matrix_inverse 14
9.0.9 matrix_kernel 15

9.0.10 matrix_list_to_matrix 15
9.0.11 matrix_matrix_to_list 15
9.0.12 matrix_rank 15
9.0.13 matrix_solve_linear 15
9.0.14 matrix_submatrix 16
9.0.15 matrix_transpose 16

10 Graphic(標準数学函数) 17

10.0.16 print_dvi_form 17
10.0.17 print_em 17
10.0.18 print_gif_form 17
10.0.19 print_input_form 17
10.0.20 print_open_math_tfb_form 18
10.0.21 print_open_math_xml_form 18
10.0.22 print_output 18
10.0.23 print_ox_rfc100_xml_form 19
10.0.24 print_png_form 19
10.0.25 print_terminal_form 19
10.0.26 print_tex_form 19
10.0.27 print_tfb_form 20
10.0.28 print_xdvi_form 20
10.0.29 print_xv_form 20

11 多項式 (標準数学函数) 22

11.0.1 poly_degree 22
11.0.2 poly_elimination_ideal 22
11.0.3 poly_factor 22
11.0.4 poly_gcd 23
11.0.5 poly_grobner_basis 23
11.0.6 poly_hilbert_polynomial 23
11.0.7 poly_initial 23
11.0.8 poly_initial_coefficients 24
11.0.9 poly_initial_term 24
11.0.10 poly_solve_linear 24

12 複体 (標準数学函数) 25

13 OpenMath 函数 (1999 版) 26

13.0.1 om_start 26
13.0.2 om_xml 26
13.0.3 om_xml_to_cmo 27

14 Differential equations (library by Okutani) 28

14.0.1 odiff_op_appell4 28
14.0.2 odiff_op_tosm1 29
14.0.3 odiff_op_toasir 29
14.0.4 odiff_op_fromasir 30
14.0.5 odiff_act 30
14.0.6 odiff_act_appell4 31
14.0.7 odiff_poly_solve 31
14.0.8 odiff_poly_solve_hg1 32
14.0.9 odiff_poly_solve_appell4 32
14.0.10 odiff_rat_solve 32

15 D-module (library by Okutani) 34
15.0.1 odmodule_d_op_tosm1 34
15.0.2 odmodule_d_op_toasir 34
15.0.3 odmodule_d_op_fromasir 35
15.0.4 odmodule_ch_ideal 35
15.0.5 odmodule_singular_locus 36
15.0.6 odmodule_restriction 36
15.0.7 odmodule_elimination 37

16 DSOLV 函数 38
16.1 函数一覧 38
16.1.1 dsolv_dual 38
16.1.2 dsolv_starting_term 39

17 GNUPLOT 函数 41
17.1 函数一覧 41
17.1.1 gnuplot.start 41
17.1.2 gnuplot 42
17.1.3 gnuplot.plot_dots 43
17.1.4 gnuplot.heat 43
17.1.5 gnuplot.output 44
17.1.6 gnuplot.plot_function 44
17.1.7 gnuplot.stop 45
17.1.8 gnuplot.setenv 45

18 グラフィックライブラリ (2 次元) 46

19 Graphic Library (2 dimensional) 47
19.0.1 glib_line 47
19.0.2 glib_open 47
19.0.3 glib_plot 47
19.0.4 glib_print 47
19.0.5 glib_ps_form 48
19.0.6 glib_putpixel 48
19.0.7 glib_tops 48
19.0.8 glib_window 49

20 Mathematica 函数 50
20.1 函数一覧 50
20.1.1 mathematica.start 50
20.1.2 mathematica.tree_to_string 51
20.1.3 mathematica.rtomstr 51

21 OpenXM-Contrib 一般函数 53
21.1 函数一覧 53
21.1.1 ox_check_errors2 53

22 OXshell の関数 54
22.0.1 oxshell.get_value 54
22.0.2 oxshell.oxshell 54
22.0.3 oxshell.set_value 54

23 pFq に関する (コ) ホモロジー 55

23.0.1 pfp_omega 55
23.0.2 pfpcoh_intersection 55
23.0.3 pfphom_intersection 56
23.0.4 pfphom_monodromy_pair_kyushu 56

24 PHC 函数 57

24.1 函数一覧 57
24.1.1 phc.start 57
24.1.2 phc.phc 58

25 Plucker 関係式 59

25.0.1 plucker 59
25.0.2 plucker_relation 59
25.0.3 plucker_y 59
25.0.4 plucker_index 60

26 SM1 函数 61

26.1 ox_sm1_forAsir サーバ 61
26.1.1 ox_sm1_forAsir 61
26.2 函数一覧 62
26.2.1 sm1.start 62
26.2.2 sm1.sm1 62
26.2.3 sm1.push_int0 63
26.2.4 sm1.gb 64
26.2.5 sm1.deRham 65
26.2.6 sm1.hilbert 66
26.2.7 sm1.genericAnn 67
26.2.8 sm1.wTensor0 67
26.2.9 sm1.reduction 68
26.2.10 sm1.xml_tree_to_prefix_string 68
26.2.11 sm1.syz 69
26.2.12 sm1.mul 70
26.2.13 sm1.distraction 70
26.2.14 sm1.gkz 71
26.2.15 sm1.appell1 71
26.2.16 sm1.appell4 72
26.2.17 sm1.rank 72
26.2.18 sm1.auto_reduce 73
26.2.19 sm1.slope 73
26.2.20 sm1.ahg 74
26.2.21 sm1.bfunction 74
26.2.22 sm1.call_sm1 75
26.2.23 sm1.ecart_homogenize01Ideal 75
26.2.24 sm1.ecartd_gb 75
26.2.25 sm1.ecartd_gb_oxRingStructure 75
26.2.26 sm1.ecartd_isSameIdeal_h 76
26.2.27 sm1.ecartd_reduction 76
26.2.28 sm1.ecartd_reduction_noh 76
26.2.29 sm1.ecartd_syz 76
26.2.30 sm1.gb_oxRingStructure 77
26.2.31 sm1.gb_reduction 77
26.2.32 sm1.gb_reduction_noh 77
26.2.33 sm1.generalized_bfunction 78

26.2.34 sm1.isSameIdeal_in_Dalg 78
26.2.35 sm1.restriction 78
26.2.36 sm1.saturation 78

27 TIGERS 函数 80
27.0.1 tigers.tigers 80

28 便利な関数 81
28.0.1 util_filter 81
28.0.2 util_find_and_replace 81
28.0.3 util_find_substr 81
28.0.4 util_index 81
28.0.5 util_load_file_as_a_string 82
28.0.6 util_part 82
28.0.7 util_read_file_as_a_string 82
28.0.8 util_remove_cr 82
28.0.9 util_timing 82
28.0.10 util_v 82
28.0.11 util_write_string_to_a_file 83

29 その他 84
29.0.1 todo_parametrize 84

索引 85