# Asir OpenXM-Contrib

Asir OpenXM-Contrib User's Manual (日本語版)
Edition 1.1.2 for OpenXM/Asir2000
February 2000

**by OpenXM Developing Team**

## 0.1 はじめに

数式処理システム asir は OpenXM プロトコルをサポートしたサーバをプラグインモジュールとして利用できる. 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. このマニュアルでは asir 用のこれらの関数を説明する.

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.

OpenXM Committers:

- Maekawa, Masahide (Oct., 1999 – : CVS server)
- Noro, Masayuki (Jan., 1996 – : OpenXM Protocol, asir99)
- Ohara, Katsuyoshi (Jan., 1998 – : ox_math)
- Okutani, Yukio (Oct., 1999 – : asir contrib)
- Takayama, Nobuki (Jan., 1996 – : OpenXM Protocol, kan)
- Tamura, Yasushi (Nov., 1998 – : OpenMath proxy)

この Contrib パッケージについて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.

# 1 LONGNAME

### 1.0.1 longname

longname :: Asir/OpenXM (OpenXM 対応の asir) は OpenMath の Content Dictionaries に部分的に対応した, 以下の函数名をサポートする.

```
Basic_Pi = @pi $
Basic_true = 1$
Basic_false = 0$
Basic_e = @e$
def basic_plus(A,B) { return(A+B); }
def basic_times(A,B) { return(A*B); }
def basic_minus(A,B) { return(A-B); }
def basic_over(A,B) { return(A/B); }
def basic_power(A,B) { return(A^B); }
def basic_and(A,B)   { return(A && B); }
def basic_or(A,B)    { return(A || B); }
def basic_not(A)   { return(! A); }
def basic_IntegerQuotient(A,B) { return(idiv(A,B)); }
def basic_IntegerRemainder(A,B) { return(irem(A,B)); }
def basic_IntegerGcd(A,B)  { return(igcd(A,B)); }
def basic_exp(A)           { return(exp(A)); }
def basic_ln(A)            { return(log(A)); }
def basic_sin(A)           { return(sin(A)); }
def basic_cos(A)           { return(cos(A)); }
def basic_tan(A)           { return(tan(A)); }
def basic_diff(A,B)        { return(diff(A,B)); }
def basic_equal(A,B)       { return( A == B); }
def basic_unequal(A,B)     { return( A != B); }
def basic_less(A,B)        { return( A < B); }
def basic_lessequal(A,B)       { return( A <= B); }
def basic_greater(A,B)       { return( A > B); }
def basic_greaterequal(A,B)        { return( A >= B); }
def basic_select(A,B)         { return(A[B]); }
def basic_length(A)           { if (type(A)==4) return(length(A));
                                else return(size(A)[0]); }
Poly_lexicogrpahic = 2$
Poly_gradedLexicographic = 1$
Poly_gradedReverseLexicographic = 0$
def poly_degreeWrt(A,B)       { return(deg(A,B)); }
def poly_factor(A) { return(fctr(A)); }
def poly_gcd(A,B)  { return(gcd(A,B)); }
def poly_resultant(A,B,V) { return(res(V,A,B)); }
def poly_groebner(O,P,V)  { return(gr(P,V,O)); }
```

basic_ などがつかないつぎのような函数名も登録してある.

```
def factor(F) { return(poly_factor(F)); }
def cancel(F)  { return(red(F)); }
def numerator(F) { return(nm(F)); }
def denominator(F) { return(dn(F)); }
def hilbert_polynomial(G,V) {
     A=gr(G,V,0); B=map(dp_ht,map(dp_ptod,A,V));
     C=map(dp_dtop,B,V);
     return(sm1_hilbert([C,V]));
}
```

# 2 OpenXM-Contrib 一般函数

## 2.1 函数一覧

### 2.1.1 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(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;
```

# 3 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 doc]$ asir
This is Asir, Version 991116.
Copyright (C) FUJITSU LABORATORIES LIMITED.
3 March 1994. All rights reserved.
[155] load("xm");
1
xm Version 19991109. ox_help(0); ox_help("keyword"); for help message
[255] gnuplot_start();
0
[257] gnuplot("plot sin(x**2);");
0
```

関数 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. 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.

```
Heat_N=20; heat(0.001,30);   (CFL number is 0.4)
Heat_N=20; heat(0.003,30);   (CFL > 0.5 unstable)
```

Author of GNUPLOT: Thomas Williams, Colin Kelley

## 3.1 函数一覧

### 3.1.1 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

### 3.1.2 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("splot x**2-y**2;"|proc=P);
0
*Plot 2 dimensional graph.
[234] 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("plot sin(x)*cos(x);");
0
[237] gnuplot(|file="x11");
0

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

### 3.1.3 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

### 3.1.4 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("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

### 3.1.5 gnuplot_plot_function

gnuplot($f$|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

# 4 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.

```
bash$ asir
This is Asir, Version 991110.
Copyright (C) FUJITSU LABORATORIES LIMITED.
3 March 1994. All rights reserved.
[155] load("m");
1
1
xm Version 19991109. ox_help(0); ox_help("keyword"); for help message
m Version 19991113. m_start, m_tree_to_string, m_N_Eigenvalues
[259] m_start();
ox_math has started.
ox_math: Portions copyright 2000 Wolfram Research, Inc.
See OpenXM/Copyright/Copyright.mathlink  for details.
0
[260] m_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 to buy Mathematica from Wolfram Research to make this package work. See http://www.wolfram.com

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

## 4.1 函数一覧

### 4.1.1 m_start

m_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 = m_start()
```

参照 ox_launch

### 4.1.2 m_tree_to_string

m_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] m_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] m_tree_to_string(A);
(1)+((-2)*(x))+((x)^(2))
[271] eval_str(@);
x^2-2*x+1

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

参照 ox_pop_cmo, eval_str, m_rtostr

### 4.1.3 m_rtostr

m_rtostr($t$)
:: $t$ 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] m_rtostr([1,2,3]);
1,2,3
[260] m_rtostr([[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 m_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. m_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 m_Inverse(M) {
  P = 0;
  A = m_rtostr(M);
  ox_execute_string(P,"Inverse["+A+"]");
  B = ox_pop_cmo(B);
  C = m_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=m_Inverse(M)$
[271] red(A[0][0]);
(z*y)/(x^2+(-y-z)*x+z*y)
```

参照 ox_execute_string, ToExpression(Mathematica), m_tree_to_string

# 5 OpenMath 函数

この節の関数を使用するには, om をロードする必要がある. JAVA の実行環境が適切に設定されている必要がある.

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

### 5.0.1 om_start

om_start()
:: OMproxy をスタートする. このサーバは CMO と OpenMath XML 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>>
```

### 5.0.2 om_xml

om_xml(s|proc=$p$)
:: s の CMO 表現を OpenMath の XML expression 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 expression 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);
```

### 5.0.3 om_xml_to_cmo

om_xml_to_cmo(*s*|proc=*p*)
:: OpenMath の XML 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 表現 s を CMO に変換する.

# 6 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 doc]$ asir
This is Asir, Version 991116.
Copyright (C) FUJITSU LABORATORIES LIMITED.
3 March 1994. All rights reserved.
[155] load("xm");
1
xm Version 19991109. ox_help(0); ox_help("keyword"); for help message
[255] phc_start();
0
[257] 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.

## 6.1 函数一覧

### 6.1.1 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

### 6.1.2 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.

```
[232] P = phc_start();
0
[233] 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)

# 7 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.

```
This is Risa/Asir, Version 20000126.
Copyright (C) FUJITSU LABORATORIES LIMITED.
1994-1999. All rights reserved.
xm version 20000202. Copyright (C) OpenXM Developing Team. 2000.
ox_help(0); ox_help("keyword"); ox_grep("keyword"); for help message
Loading ~/.asirrc

[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.

## 7.1 ox_sm1_forAsir サーバ

### 7.1.1 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'

## 7.2 函数一覧

### 7.2.1 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 Sm1_proc に格納される.

```
[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

### 7.2.2 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.

```
[261] sm1(0," ( (x-1)^2 ) . ");
0
```

```
[262] ox_pop_string(0);
x^2-2*x+1
[263] 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.

### 7.2.3 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.

Reference
ox_push_cmo

### 7.2.4 sm1_gb

sm1_gb([$f$,$v$,$w$]|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.
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
$p$　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 イニシァル多項式 ( $w$ 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.

```
[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).

```
[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

### 7.2.5 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_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_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)
参考論文 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.

### 7.2.6 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.

*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.
- 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.

```
[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

### 7.2.7 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

### 7.2.8 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]]
```

### 7.2.9 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],[g1,...gm]] ここで $g$=[g1, ..., gm] であり, r/c0 + c1 g1 + ... + cm gm = 0 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],[x+y^3-4*y,y^4-4*y^2+1]]
[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],[x+y^3-4*y,y^4-4*y^2+1]]
```

参照 sm1_start, sm1_find_proc, d_true_nf

### 7.2.10 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

### 7.2.11 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]]]]
```

### 7.2.12 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.

```
[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]]
```

### 7.2.13 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),

### 7.2.14 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]]
```

### 7.2.15 sm1_appell1

sm1_appell1(a|proc=$p$)

:: F_1 または F_D に対応すう, 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.

```
[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]));
1

[285] Mu=2$ Beta = 1/3$
[287] sm1_rank(sm1_appell1([Mu+Beta,Mu+1,Beta,Beta,Beta]));
4
```

### 7.2.16 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
```

### 7.2.17 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.

# 8 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.

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.

## 8.1 函数一覧

### 8.1.1 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.
- 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.
```

### 8.1.2 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.

```
[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]]]
```

# 9 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.

### 9.0.1 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(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.

# 10 その他の函数

この章ではいろいろな関数を紹介します.

## 10.1 Matrix operations

ファイル 'gr' が必要です.

### 10.1.1 matrix_1

matrix_1($N$)
:: $N$ 次単位行列を生成します.

*return* 行列
$N$ 整数

- matrix_1の例.

```
[297] matrix_1(1);
[ 1 ]

[298] matrix_1(2);
[ 1 0 ]
[ 0 1 ]

[299] matrix_1(3);
[ 1 0 0 ]
[ 0 1 0 ]
[ 0 0 1 ]
```

### 10.1.2 matrix_diag

matrix_diag($L$)
:: リスト $L$ の要素を対角成分とする対角行列を生成します.

*return* 行列
$L$ リスト

- matrix_diagの例.

```
[301] matrix_diag([1,2,3]);
[ 1 0 0 ]
[ 0 2 0 ]
[ 0 0 3 ]
```

```
[302] matrix_diag([a,b,c]);
[ a 0 0 ]
[ 0 b 0 ]
[ 0 0 c ]
```

### 10.1.3 matrix_trans

matrix_trans($A$)
:: 行列 $A$ の転置行列を求めます.

*return* 行列
$A$ 行列 or リスト

- matrix_transの例.

```
[303] matrix_trans([[a,b],[c,d]]);
[ a c ]
[ b d ]

[304] matrix_trans([[1,2,3],[4,5,6],[7,8,9]]);
[ 1 4 7 ]
[ 2 5 8 ]
[ 3 6 9 ]
```

### 10.1.4 matrix_inverse

matrix_inverse($A$)
:: 行列 $A$ の逆行列を求めます.

*return* 行列
$A$ 行列 or リスト

- 行列 $A$ の成分は有理数 or 多項式.
- matrix_inverseの例.

```
[309] matrix_inverse([[1,1],[1,2]]);
[ 2 -1 ]
[ -1 1 ]

[310] matrix_inverse([[a,b],[c,d]]);
[ (d)/(d*a-c*b) (-b)/(d*a-c*b) ]
[ (-c)/(d*a-c*b) (a)/(d*a-c*b) ]
```

### 10.1.5 matrix_solve

matrix_solve($A$,$X$,$Y$)
:: matrix_solveは方程式 A*X=Y の解を求めます.

| | |
|---|---|
| *return* | リスト |
| $A$ | 行列 or リスト |
| $X$ | ベクトル or リスト |
| $Y$ | ベクトル or リスト |

- 行列 $A$ の成分は有理数 or 多項式.
- matrix_solveの例.

```
[376] matrix_solve([[1,-1],[2,3]],[x,y],[1,1]);
[[x,4/5],[y,-1/5]]

[377] matrix_solve([[a,b],[c,d]],[x,y],[1,1]);
[[x,(-b+d)/(d*a-c*b)],[y,(a-c)/(d*a-c*b)]]
```

### 10.1.6 matrix_kernel

matrix_kernel($A$)
:: matrix_kernelは行列 A の kernel の基底を求めます.

| | |
|---|---|
| *return* | リスト |
| $A$ | 行列 or リスト |

- 行列 $A$ の成分は有理数 or 多項式.
- matrix_kernelの例.

```
[383] matrix_kernel([[1,2,3,4],[1,0,1,0]]);
[2,[[1,0,-1,1/2],[0,1,0,-1/2]]]

[384] matrix_kernel([[1,2,3,4],[1,0,1,0],[0,0,0,1]]);
[1,[[1,1,-1,0]]]
```

## 10.2 Differential equations

ファイル 'gr', 'Matrix' が必要です.

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.

### 10.2.1 diff_op_appell4

diff_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

- diff_op_appell4の例.

```
[298] diff_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]] ] ]
```

### 10.2.2 diff_op_tosm1

diff_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.
- diff_op_tosm1の例

```
[299] diff_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] diff_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] diff_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] diff_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))]
```

### 10.2.3 diff_op_toasir

diff_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

- diff_op_toasirの例

```
[303] diff_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] diff_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]
```

### 10.2.4 diff_op_fromasir

diff_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

- diff_op_fromasirの例

```
[305] diff_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] diff_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]]]]
```

### 10.2.5 diff_act

diff_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

- diff_actの例

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

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

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

### 10.2.6 diff_act_appell4

diff_act_appell4($a$,$b$,$c1$,$c2$,$F$,$V$)
:: 微分作用素 diff_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

- diff_act_appell4の例

```
[303] diff_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] diff_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] diff_act_appell4(-2,-2,-1,-1,x^2+y^2,[x,y]);
[0,0]
```

### 10.2.7 diff_poly_solve

diff_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 |

- diff_poly_solveの例.

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

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

[356] diff_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]]
```

### 10.2.8 diff_poly_solve_hg1

diff_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 |

- diff_poly_solve_hg1の例.

```
[334] diff_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] diff_poly_solve_hg1(-3,-6,-7,[x]);
[-4/7*_2*x^3+15/7*_2*x^2-18/7*_2*x+_2,[_2]]
```

### 10.2.9 diff_poly_solve_appell4

diff_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

- diff_poly_solve_appell4の例.

```
[299] diff_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] diff_poly_solve_appell4(-3,1,1,-1,[x,y]);
[-3*_45*y^2*x-_45*y^3+_45*y^2,[_45]]
```

### 10.2.10 diff_rat_solve

diff_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

- diff_rat_solveの例.

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

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

[350] D = diff_op_appell4(0,0,3,0,[x,y])$
[351] diff_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]]
```

## 10.3 D-module

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

### 10.3.1 dmodule_d_op_tosm1

dmodule_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.
- dmodule_d_op_tosm1の例

```
[299] dmodule_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] dmodule_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] dmodule_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] dmodule_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))]
```

### 10.3.2 dmodule_d_op_toasir

dmodule_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.
- dmodule_d_op_toasirの例

```
[303] dmodule_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] dmodule_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]
```

### 10.3.3 dmodule_d_op_fromasir

dmodule_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.
- dmodule_d_op_fromasirの例

```
[305] dmodule_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] dmodule_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]]]]
```

### 10.3.4 dmodule_ch_ideal

dmodule_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.
- dmodule_ch_idealの例

```
[344] dmodule_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] dmodule_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]
```

### 10.3.5 dmodule_singular_locus

dmodule_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.
- dmodule_singular_locusの例

```
[356] D = diff_op_appell4(a,b,c1,c2,[x,y])$
[357] dmodule_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] dmodule_singular_locus(D,[x]);
[x^2-x]
```

### 10.3.6 dmodule_restriction

dmodule_restriction(*D_ideal*, *V*, *Rest*)
:: *D_ideal* 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.
- dmodule_restrictionの例.

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

### 10.3.7 dmodule_elimination

dmodule_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.
- dmodule_eliminationの例.

```
[346] dmodule_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] dmodule_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]
```

# 索引

(Index is nonexistent)

(Index is nonexistent)

# 目次(抄)

1 LONGNAME . . . 3
2 OpenXM-Contrib 一般函数 . . . 5
3 GNUPLOT 函数 . . . 6
4 Mathematica 函数 . . . 10
5 OpenMath 函数 . . . 13
6 PHC 函数 . . . 15
7 SM1 函数 . . . 17
8 DSOLV 函数 . . . 29
9 TIGERS 函数 . . . 31
10 その他の函数 . . . 32
索引 . . . 44

# 目次

0.1 はじめに ..... 1

**1 LONGNAME ..... 3**

1.0.1 longname ..... 3

**2 OpenXM-Contrib 一般函数 ..... 5**

2.1 函数一覧 ..... 5

2.1.1 ox_check_errors2 ..... 5

**3 GNUPLOT 函数 ..... 6**

3.1 函数一覧 ..... 6

3.1.1 gnuplot_start ..... 6

3.1.2 gnuplot ..... 7

3.1.3 gnuplot_plot_dots ..... 7

3.1.4 gnuplot_output ..... 8

3.1.5 gnuplot_plot_function ..... 9

**4 Mathematica 函数 ..... 10**

4.1 函数一覧 ..... 10

4.1.1 m_start ..... 10

4.1.2 m_tree_to_string ..... 11

4.1.3 m_rtostr ..... 11

**5 OpenMath 函数 ..... 13**

5.0.1 om_start ..... 13

5.0.2 om_xml ..... 13

5.0.3 om_xml_to_cmo ..... 14

**6 PHC 函数 ..... 15**

6.1 函数一覧 ..... 15

6.1.1 phc_start ..... 15

6.1.2 phc ..... 16

**7 SM1 函数 ..... 17**

7.1 ox_sm1_forAsir サーバ ..... 17

7.1.1 ox_sm1_forAsir ..... 17

7.2 函数一覧 ..... 18

7.2.1 sm1_start ..... 18

7.2.2 sm1 ..... 18

7.2.3 sm1_push_int0 ..... 19

7.2.4 sm1_gb ..... 19

7.2.5 sm1_deRham ..... 20

7.2.6 sm1_hilbert ..... 21

7.2.7 sm1_genericAnn ..... 22

7.2.8 sm1_wTensor0 ..... 22

7.2.9 sm1_reduction ..... 23

7.2.10 sm1_xml_tree_to_prefix_string ..... 23

7.2.11 sm1_syz ..... 24

7.2.12 sm1_mul ..... 25
7.2.13 sm1_distraction ..... 25
7.2.14 sm1_gkz ..... 26
7.2.15 sm1_appell1 ..... 26
7.2.16 sm1_rank ..... 27
7.2.17 sm1_auto_reduce ..... 28

**8 DSOLV 函数 ..... 29**
8.1 函数一覧 ..... 29
8.1.1 dsolv_dual ..... 29
8.1.2 dsolv_starting_term ..... 30

**9 TIGERS 函数 ..... 31**
9.0.1 tigers ..... 31

**10 その他の函数 ..... 32**
10.1 Matrix operations ..... 32
10.1.1 matrix_1 ..... 32
10.1.2 matrix_diag ..... 32
10.1.3 matrix_trans ..... 33
10.1.4 matrix_inverse ..... 33
10.1.5 matrix_solve ..... 33
10.1.6 matrix_kernel ..... 34
10.2 Differential equations ..... 34
10.2.1 diff_op_appell4 ..... 35
10.2.2 diff_op_tosm1 ..... 35
10.2.3 diff_op_toasir ..... 36
10.2.4 diff_op_fromasir ..... 36
10.2.5 diff_act ..... 37
10.2.6 diff_act_appell4 ..... 37
10.2.7 diff_poly_solve ..... 38
10.2.8 diff_poly_solve_hg1 ..... 38
10.2.9 diff_poly_solve_appell4 ..... 39
10.2.10 diff_rat_solve ..... 39
10.3 D-module ..... 39
10.3.1 dmodule_d_op_tosm1 ..... 40
10.3.2 dmodule_d_op_toasir ..... 40
10.3.3 dmodule_d_op_fromasir ..... 41
10.3.4 dmodule_ch_ideal ..... 41
10.3.5 dmodule_singular_locus ..... 41
10.3.6 dmodule_restriction ..... 42
10.3.7 dmodule_elimination ..... 42

**索引 ..... 44**