Asir OpenXM-Contrib

Asir OpenXM-Contrib User's Manual (English Edition)
Edition 1.0 for Asir2000
January 2000

by OpenXM Developing Team

0.1 Introduction

The computer algebra system `asir` can use servers, which support the `OpenXM` protocols, as plugin modules. The interface functions to call these servers are read by loading the file '`xm`'. This document explains these interface functions for `asir`.

As to technical details on the `OpenXM` protocols, see '`openxm-eg.tex`' at '`$(OpenXM_HOME)/doc/OpenXM-specs`'.

Enjoy mathematics on your computer.

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)

See `OpenXM/Copyright` for the copyright notice.

1 OpenXM-Contrib General Functions

1.1 Functions

1.1.1 ox_check_errors2

`ox_check_errors2(`*p*`)`
:: get a list of error objects on the statck of the server *p*.

return List

p Number

- It gets a list of error objects on the server stack.
- It does not pop the error objects.

```
[219] P=sm1_start();
0
[220] sm1(P," 0 get ");
0
[221] ox_check_errors2(P);
[error([7,4294967295,executeString: Usage:get])]
Type in ox_pops(P,N) to clean the stack of the ox server.
(debug)
```

2 GNUPLOT Functions

This chapter describes interface functions for GNUPLOT ox server `ox_sm1_gnuplot`. These interface functions are defined in the file `gnuplot`. The file '`gnuplot`' is at '`$(OpenXM_HOME)/lib/asir-contrib`'.

```
[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
```

The function `heat`(dt,*step*) demonstrates our gnuplot interface. It numerically solves the heat equation

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

with the initial condition

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

by the explicit scheme for $0 \leq t \leq dt * step$. The segment [0,1] is devided into `Heat_N` segments. If the celebrated Courant-Friedrichs-Levi number dt*`Heat_N`*`Heat_N` is less than or equal to 0.5, then the explicit scheme is numerically stable. One can observe the instability by changing CFL number.

```
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

2.1 Functions

2.1.1 gnuplot_start

`gnuplot_start()`
:: Start `ox_sm1_gnuplot` on the localhost.

return Integer

- Start `ox_sm1_gnuplot` on the localhost. It returns the descriptor of `ox_sm1_gnuplot`.
- Set `Xm_nox = 1` to start `ox_sm1_gnuplot` without a debug window.
- The descriptor is stored in `Gnuplot_proc`.

```
P = gnuplot_start();
```

`Reference`
`ox_launch`, `gnuplot`

2.1.2 gnuplot

`gnuplot(`*s*`|proc=`*p*`)`
:: Ask GNUPLOT to execute the command string *s*.

return Void

p Number

s String

- The server executes the gnuplot command *s*. When an error occurs, the gnuplot itself terminates and ox_sm1_gnuplot server automatically restarts gnuplot.
- gnuplot does not accept a long polynomial.
- gnuplot does not accept ^. Use ** instead.

```
[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)");
```

`Reference`
`ox_launch`, `gnuplot_start`, `rtostr`, `gnuplot_plotDots`

`Reference Book`
Yabuki Michiro, Otake Tuyoshi; Tukai konasu GNUPLOT, Techno Press, in Japansese, ISBN4-924998-11-7

2.1.3 gnuplot_plotDots

`gnuplot_plotDots(`*d*`,`*s*`|proc=`*p*`)`
:: Plot the dots *d* with the style *s*.

return Void

p Number

d List

s String or 0

- Plot the dots *d* with the style *s*. *s* is a string of the form "style color point". Here, style can be lines, points, linespoints, impulses, dots, steps, errorbars, boxes, boxerrorbars. color can be 1 (red), 2 (green), 3 (blue), 4, ... , 8. point can be a number from 1 to 8. The color and point field can be omitted.
- When *d* == `[ ]`, the screen will be cleared.

```
[239] P = gnuplot_start();
0
[240] gnuplot_plotDots([ ],0);
0
[241] for (I=0; I<10; I++) gnuplot_plotDots([[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_plotDots(A," lines ");
0
```

Reference

`gnuplot_start`, `plot "fileName" with options`(GNUPLOT command), `gnuplot_clean`, `gnuplot`

2.1.4 gnuplot_output

`gnuplot_output(|`*file*=*s*`)`

:: ask `GNUPLOT` to output graphic to the file *s* in the Postscript format.

return Void

s String

- ask `GNUPLOT` to output graphic to the file *s* in the Postscript format.
- When `s` is "x11" or this function is called without the argument, the output will be written to X11 display.

```
[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
```

Reference

`gnuplot`

3 Mathematica Functions

This chapter describes interface functions for Mathematica ox server `ox_math`. These interface functions are defined in the file 'm'. You need to load the file before using the interface functions. by the command `load("m")$`. The file 'm' is at '`$(OpenXM_HOME)/lib/asir-contrib`'.

Note: `ox_reset` does not work.

```
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`.

3.1 Functions

3.1.1 m_start

`m_start()`
:: Start `ox_math` on the localhost.

return Integer

- Start `ox_math` on the localhost. It returns the descriptor of `ox_math`.
- Set `Xm_nox = 1` to start `ox_math` without a debug window.
- The descriptor is stored in the variable `M_proc`.

```
P = m_start()
```

`Reference`
`ox_launch`

3.1.2 m_tree_to_string

`m_tree_to_string`(t)
:: translates Mathematica tree data t into a string that can be understandable by `asir` as far as possible.

return String

t List

- t is a Mathematica tree data which is generated by `ox_math`.
- This function translates Mathematica tree data t into a string that may be understandable by `asir`.
- This function translates t into a prefix or infix expression that may be understantable by `asir`. The first element of the list t is a key word string of the Mathematica object. If this function recognizes the key word, it translates `t` into the form that can be understandable by `asir`. If it cannot recognizes the key word, it translates `t` into a function call with the function name `m_(the key word)`.

```
[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*)
```

Reference
`ox_pop_cmo`, `eval_str`, `m_rtostr`

3.1.3 m_rtostr

`m_rtostr`(t)
:: translate the object t into a string that can be understandable by Mathematica as far as possible.

return String

t Object

- It translates the object t into a string that can be understandable by Mathematica as far as possible. For example, `asir` uses `[, ]` to express a list, but `Mathematica` uses `{, }`. This function makes this sort of translations.

```
[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}}
```

Let us see one more example. The following function `m_Inverse(M)` outputs the inverse matrix of the matrix `M` by calling `ox_math`. It translates `asir` matrix `M` into a Mathematica expression by `r_tostr(M)` and makes Mathematica compute the inverse matrix of `M` by `ox_execute_string`.

```
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)
```

Reference

`ox_execute_string`, `ToExpression`(Mathematica), `m_tree_to_string`

4 OpenMath Functions

In order to use functions in this section, you need to load `om` by the command `load("om");`

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

4.0.1 om_start

`om_start()`
:: Start `OMproxy` server to make a translation between CMO and OpenMath XML expressions.

return Number

```
[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>>
```

4.0.2 om_xml

`om_xml(`s`|proc=`p`)`
:: Translate CMO expression of s to a XML expression of OpenMath.

return String

p Number

s Object

- Translate CMO s to a XML expression of OpenMath.

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

4.0.3 om_xml_to_cmo

`om_xml_to_cmo(`*s*`|proc=`*p*`)`
:: Translate XML expression *s* of OpenMath to a CMO.

return Object

p Number

s String

- Translate XML expression *s* of OpenMath to a CMO.

5 PHC Functions

This chapter describes interface functions for PHC pack ox server `ox_sm1_phc`. These interface functions are defined in the file 'phc'. The file `phc` is at '`$(OpenXM_HOME)/lib/asir-contrib`'.

```
[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.

Reference 1: Jan Verschelde, PHCpack: A general-purpose solver for polynomial systems by homotopy continuation". ACM Transaction on Mathematical Softwares, 25(2): 251-276, 1999.

Reference 2: Cox, D., O'Shea, Little, J., Using Algebraic Geometry, Springer. See the chapter on mixed volumes.

5.1 Functions

5.1.1 phc_start

`phc_start()`
:: Start `ox_sm1_phc` on the localhost.

return Integer

- Start `ox_sm1_phc` on the localhost. It returns the descriptor of `ox_sm1_phc`.
- Set `Xm_nox = 1` to start `ox_sm1_phc` without a debug window.
- The descriptor is stored in `Phc_proc`.

```
P = phc_start()
```

`Reference`
`ox_launch`, `phc`

5.1.2 phc

`phc(`*s*`|proc=`*p*`)`

:: Ask PHC pack to find all the roots in the complex torus of the given systems of polynomials *s*

return Void

p Number

s List

- The server calls PHC pack to solve a system of algebraic equations S by homotopy methods. PHC pack has been developed by Jan Verschelde. See `www.mth.msu.edu/~jan` for the original distribution. The original PHC pack can choose several strategies to solve, but our phc interface uses only black-box solver, which is general and automatic but is not efficient. So, if you fails by our interface, try the other strategies via the original user interface.
- phc generates working files tmp.phc.out.pid, tmp.input.*, tmp.output.*. Here, pid the process number of the server. The file `tmp.output.*` contains details informations on how PCH pack solves the system.
- The number of variables and the number of equations `length(s)` must agree.

```
[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
  ...
```

`Reference`

`ox_launch`, `phc_start`, '`$(OpenXM_HOME)/bin/lin_phcv2`'(original PHC pack binary for linux)

6 SM1 Functions

This chapter describes interface functions for sm1 ox server `ox_sm1_forAsir`. These interface functions are defined in the file '`sm1`'. The file '`sm1`' is at '`$(OpenXM_HOME)/lib/asir/contrib-packages`'. The system `sm1` is a system to compute in the ring of differential operators. Many constructions of invariants in the computational algebraic geometry reduce to constructions in the ring of differential operators. Documents on `sm1` are in the directory `OpenXM/doc/kan96xx`.

Let us evaluate the dimensions of the de Rham cohomology groups of $X := \mathbf{C} \setminus \{0, 1\} = \mathbf{C} \setminus V(x(x-1))$. The space X is a two punctured plane, so two loops that encircles the points $x = 0$ and $x = 1$ respectively spans the first homology group. Hence, the dimension of the first de Rham cohomology group is 2. `sm1` answers the dimensions of the 0th and the first cohomology groups.

```
[155] load("xm");
1
xm Version 19991118. ox_help(0); ox_help("keyword"); for help message
[255] sm1_start();
0
[256] 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: Saito, M., Sturmfels, B., Takayama, N., Grobner Deformations of Hypergeometric Differential Equations, 1999, Springer. See the appendix.

6.1 ox_sm1_forAsir Server

6.1.1 ox_sm1_forAsir

`ox_sm1_forAsir`
:: `sm1` server for `asir`.

- `ox_sm1_forAsir` is the `sm1` server started from asir by the command `sm1_start`. In the standard setting,
  `ox_sm1_forAsir` = '`$(OpenXM_HOME)/lib/sm1/bin/ox_sm1`' + '`$(OpenXM_HOME)/lib/sm1/callsm1.sm1`' (macro file)
  + '`$(OpenXM_HOME)/lib/sm1/callsm1b.sm1`' (macro file)
  The macro files '`callsm1.sm1`' and '`callsm1b.sm1`' are searched from current directory, `$(LOAD_SM1_PATH)`, `$(OpenXM_HOME)/lib/sm1`, `/usr/local/lib/sm1` in this order.
- Note for programmers: See the files '`$(OpenXM_HOME)/src/kxx/oxserver00.c`', '`$(OpenXM_HOME)/src/kxx/sm1stackmachine.c`' to build your own server by reading `sm1` macros.

6.2 Functions

6.2.1 sm1_start

`sm1_start()`
:: Start `ox_sm1_forAsir` on the localhost.

return Integer

- Start `ox_sm1_forAsir` on the localhost. It returns the descriptor of `ox_sm1_forAsir`.
- Set `Xm_nox = 1` to start `ox_sm1_forAsir` without a debug window.
- You should set suitable orders of variable by the command `ord`. For example, when you are working in the ring of differential operators on the variable `x` and `dx` (`dx` stands for $\partial/\partial x$), `sm1` server assumes that the variable `dx` is collected to the right and the variable `x` is collected to the left in the printed expression. In the example below, you must not use the variable `c` for computation in `sm1`.
- The descriptor is stored in `Sm1_proc`.

```
[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] c*dc;
dc*c
[263] sm1_mul(da,a,[a]);
a*da+1
[264] sm1_mul(a,da,[a]);
a*da
```

`Reference`
`ox_launch`, `sm1_push_int0`, `sm1_push_poly0`, `ord`

6.2.2 sm1

`sm1(`p`,`s`)` :: ask the `sm1` server to execute the command string s.

return Void

p Number

s String

- It asks the `sm1` server of the descriptor number p to execute the command string s.

```
[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]
```

`Reference`
`sm1_start`, `ox_push_int0`, `sm1_push_poly0`.

6.2.3 sm1_push_int0

`sm1_push_int0(`p`,`f`)`
:: push the object f to the server with the descriptor number p.

return Void

p Number

f Object

- When `type(`f`)` is 2 (recursive polynomial), f is converted to a string (type == 7) and is sent to the server by `ox_push_cmo`.
- When `type(`f`)` is 0 (zero), it is translated to the 32 bit integer zero on the server. Note that `ox_push_cmo(`p`,0)` sends `CMO_NULL` to the server. In other words, the server does not get the 32 bit integer 0 nor the bignum 0.
- `sm1` integers are classfied into the 32 bit integer and the bignum. When `type(`f`)` is 1 (number), it is translated to the 32 bit integer on the server. Note that `ox_push_cmo(`p`,1234)` send the bignum 1234 to the `sm1` server.
- In other cases, `ox_push_cmo` is called without data conversion.

`Reference`
`ox_push_cmo`

6.2.4 sm1_gb

`sm1_gb([`f`,`v`,`w`]|proc=`p`)`
:: computes the Grobner basis of f in the ring of differential operators with the variable v.

`sm1_gb_d([`f`,`v`,`w`]|proc=`p`)`
:: computes the Grobner basis of f in the ring of differential operators with the variable v. The result will be returned as a list of distributed polynomials.

return List

p Number

f, v, w List

- It returns the Grobner basis of the set of polynomials f in the ring of deferential operators with the variables v.
- The weight vectors are given by w, which can be omitted. If w is not given, the graded reverse lexicographic order will be used to compute Grobner basis.
- The return value of `sm1_gb` is the list of the Grobner basis of f and the initial terms (when w is not given) or initial ideal (when w is given).
- `sm1_gb_d` returns the results by a list of distributed polynomials. Monomials in each distributed polynomial are ordered in the given order. The return value consists of [variable names, order matrix, grobner basis in districuted polynomials, initial monomials or initial polynomials].

- When a non-term order is given, the Grobner basis is computed in the homogenized Weyl algebra (See Section 1.2 of the book of SST). The homogenization variable h is automatically added.

```
[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]]
```

In the example above, the set $\{x\partial_x + y\partial_y - 1, y^2\partial_y^2 + 2\}$ is the Gröbner basis of the input with respect to the graded reverse lexicographic order such that $1 \leq \partial_y \leq \partial_x \leq y \leq x \leq \cdots$. The set $\{x\partial_x, y^2\partial_y\}$ is the leading monomials (the initial monominals) of the Gröbner basis.

```
[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]]
```

In the example above, two monomials $m = x^a y^b \partial_x^c \partial_y^d$ and $m' = x^{a'} y^{b'} \partial_x^{c'} \partial_y^{d'}$ are firstly compared by the weight vector `(dx,dy,x,y) = (50,2,1,0)` (i.e., m is larger than m' if $50c + 2d + a > 50c' + 2d' + a'$) and when the comparison is tie, then these are compared by the reverse lexicographic order (i.e., if $50c + 2d + a = 50c' + 2d' + a'$, then use the reverse lexicogrpahic order).

```
[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>>]]]
```

6.2.5 sm1_deRham

`sm1_deRham([`f, v`]|proc=`p`)`

:: ask the server to evaluate the dimensions of the de Rham cohomology groups of C^n - (the zero set of f=0).

return List

p Number

f String or polynomial

v List

- It returns the dimensions of the de Rham cohomology groups of X = C^n \ V(f). In other words, it returns [dim H^0(X,C), dim H^1(X,C), dim H^2(X,C), ..., dim H^n(X,C)].
- *v* is a list of variables. n = `length(`*v*`)`.
- `sm1_deRham` requires huge computer resources. For example, `sm1_deRham(0,[x*y*z*(x+y+z-1)*(x-y),[x,y,z]])` is already very hard.
- To efficiently analyze the roots of b-function, `ox_asir` should be used from `ox_sm1_forAsir`. It is recommended to load the communication module for `ox_asir` by the command
  `sm1(0,"[(parse) (oxasir.sm1) pushfile] extension");` This command is automatically executed when `ox_sm1_forAsir` is started.
- If you make an interruption to the function `sm1_deRham` by `ox_reset(Sm1_proc);`, the server might get out of the standard mode. So, it is strongly recommended to execute the command `ox_shutdown(Sm1_proc);` to interrupt and restart the server.

```
[332] sm1_deRham([x^3-y^2,[x,y]]);
[1,1,0]
[333] sm1_deRham([x*(x-1),[x]]);
[1,2]
```

`Reference`
`sm1_start`, `deRham` (sm1 command)

`Reference paper`
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.

6.2.6 sm1_hilbert

`sm1_hilbert([`*f*`,`*v*`]|proc=`*p*`)`
:: ask the server to compute the Hilbert polynomial for the set of polynomials *f*.

return Polynomial

p Number

f, v List

- It returns the Hilbert polynomial h(k) of the set of polynomials *f* with respect to the set of variables *v*.
- h(k) = dim_Q F_k/I \cap F_k where F_k the set of polynomials of which degree is less than or equal to k and I is the ideal generated by the set of polynomials *f*.
- For an efficient computation, it is preferable that the set of polynomials *f* is a set of monomials. In fact, this function firstly compute a Grobner basis of *f*, and then compute the Hilbert polynomial of the initial monomials of the basis. If the input *f* is already a Grobner basis, a Grobner basis is recomputed in this function, which is a waste of time and Grobner basis computation in the ring of polynomials in `sm1` is slower than in `asir`.

```
[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
```

Reference

sm1_start, sm1_gb

6.2.7 sm1_genericAnn

sm1_genericAnn([*f*,*v*]|proc=*p*)

:: It computes the annihilating ideal for *f*^s. *v* is the list of variables. Here, s is *v*[0] and *f* is a polynomial in the variables rest(*v*).

return List

p Number

f Polynomial

v List

- This function computes the annihilating ideal for *f*^s. *v* is the list of variables. Here, s is *v*[0] and *f* is a polynomial in the variables rest(*v*).

```
[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]
```

Reference

sm1_start

6.2.8 sm1_wTensor0

sm1_wTensor0([*f*,*g*,*v*,*w*]|proc=*p*)

:: It computes the D-module theoretic 0-th tensor product of *f* and *g*.

return List

p Number

f, *g*, *v*, *w* List

- It returns the D-module theoretic 0-th tensor product of f and g.
- v is a list of variables. w is a list of weights. The integer w[i] is the weight of the variable v[i].
- `sm1_wTensor0` calls `wRestriction0` of `ox_sm1`, which requires a generic weight vector w to compute the restriction. If w is not generic, the computation fails.
- Let F and G be solutions of f and g respectively. Intuitively speaking, the 0-th tensor product is a system of differential equations which annihilates the function FG.
- The answer is a submodule of a free module D^r in general even if the inputs f and g are left ideals of D.

```
[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]]
```

6.2.9 sm1_reduction

`sm1_reduction([`f`,`g`,`v`,`w`]|proc=`p`)`

::

return List

f Polynomial

g, v, w List

p Number (the process number of ox_sm1)

- It reduces f by the set of polynomial g in the homogenized Weyl algebra; it applies the division algorithm to f. The set of variables is v and w is weight vectors to determine the order, which can be ommited.
- The return value is of the form [r,c0,[c1,...,cm],[g1,...gm]] where g=[g1, ..., gm] and r/c0 + c1 g1 + ... + cm gm = 0. r/c0 is the normal form.
- The function reduction reduces reducible terms that appear in lower order terms.
- The functions sm1_reduction_d(P,F,G) and sm1_reduction_noH_d(P,F,G) are for distributed polynomials.

```
[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]]
```

Reference

`sm1_start`, `sm1_find_proc`, `d_true_nf`

6.2.10 sm1_xml_tree_to_prefix_string

`sm1_xml_tree_to_prefix_string(`s`|proc=`p`)`

:: Translate OpenMath Tree Expression s in XML to a prefix notation.

return String

p Number

s String

- It translate OpenMath Tree Expression *s* in XML to a prefix notation.
- This function should be moved to om_* in a future.
- `om_xml_to_cmo(OpenMath Tree Expression)` returns CMO_TREE. asir has not yet understood this CMO.
- `java` execution environment is required. (For example, `/usr/local/jdk1.1.8/bin` should be in the command search path.)

```
[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);
plus_Basic(times_Basic(power_Basic(x,4),1),times_Basic(power_Basic(x,0),-1))
```

Reference

`om_*`, `OpenXM/src/OpenMath`, `eval_str`

6.2.11 sm1_syz

`sm1_syz([`*f*`,`*v*`,`*w*`]|proc=`*p*`)`

:: computes the syzygy of *f* in the ring of differential operators with the variable *v*.

return List

p Number

f, *v*, *w* List

- The return values is of the form [s,[*g*, *m*, *t*]]. Here *s* is the syzygy of *f* in the ring of differential operators with the variable *v*. *g* is a Groebner basis of *f* with the weight vector *w*, and *m* is a matrix that translates the input matrix *f* to the Gr\"obner basis *g*. *t* is the syzygy of the Gr\"obner basis *g*. In summary, $g = m\ f$ and $s\ f = 0$ hold as matrices.
- The weight vectors are given by *w*, which can be omitted. If *w* is not given, the graded reverse lexicographic order will be used to compute Grobner basis.
- When a non-term order is given, the Grobner basis is computed in the homogenized Weyl algebra (See Section 1.2 of the book of SST). The homogenization variable h is automatically added.

```
[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]]]]
```

6.2.12 sm1_mul

`sm1_mul`(*f*,*g*,*v*|`proc`=*p*)

:: ask the sm1 server to multiply *f* and *g* in the ring of differential operators over *v*.

return Polynomial or List

p Number

f, *g* Polynomial or List

v List

- Ask the sm1 server to multiply *f* and *g* in the ring of differential operators over *v*.
- `sm1_mul_h` is for homogenized Weyl algebra.

```
[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]]
```

Index

(Index is nonexistent)

(Index is nonexistent)

Short Contents

1 OpenXM-Contrib General Functions 2
2 GNUPLOT Functions 3
3 Mathematica Functions 6
4 OpenMath Functions 9
5 PHC Functions 11
6 SM1 Functions 13
Index 22

Table of Contents

0.1 Introduction 1

1 OpenXM-Contrib General Functions 2

1.1 Functions 2
1.1.1 ox_check_errors2 2

2 GNUPLOT Functions 3

2.1 Functions 3
2.1.1 gnuplot_start 3
2.1.2 gnuplot 4
2.1.3 gnuplot_plotDots 4
2.1.4 gnuplot_output 5

3 Mathematica Functions 6

3.1 Functions 6
3.1.1 m_start 6
3.1.2 m_tree_to_string 7
3.1.3 m_rtostr 7

4 OpenMath Functions 9

4.0.1 om_start 9
4.0.2 om_xml 9
4.0.3 om_xml_to_cmo 10

5 PHC Functions 11

5.1 Functions 11
5.1.1 phc_start 11
5.1.2 phc 12

6 SM1 Functions 13

6.1 ox_sm1_forAsir Server 13
6.1.1 ox_sm1_forAsir 13
6.2 Functions 14
6.2.1 sm1_start 14
6.2.2 sm1 14
6.2.3 sm1_push_int0 15
6.2.4 sm1_gb 15
6.2.5 sm1_deRham 16
6.2.6 sm1_hilbert 17
6.2.7 sm1_genericAnn 18
6.2.8 sm1_wTensor0 18
6.2.9 sm1_reduction 19

6.2.10 `sm1_xml_tree_to_prefix_string` 19
6.2.11 `sm1_syz` 20
6.2.12 `sm1_mul` 21

Index 22