OpenXM/Risa/Asir-Contrib

OpenXM/Risa/Asir-Contrib User's Manual (English Edition)
Edition 1.2.2 for OpenXM/Asir2000
August 2002

by OpenXM Developing Team

1 Introduction

The computer algebra system `asir` can use servers, which support the `OpenXM` protocols (Open message eXchange for Mathematics, http://www.openxm.org), as components. The interface functions to call these servers are loaded by loading the file 'xm'. This file is automatically loaded in "Risa/Asir(OpenXM distribution)", which we call `OpenXM/Risa/Asir` in this document. This document explains these interface functions for `asir` and several mathematical and utility functions written in the user languages of Risa/Asir. These mathematica and utilitiy functions are outputs of Asir-contrib project.

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

Enjoy mathematics on your computer.

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)

See `OpenXM/Copyright` for the copyright notice.

2 Function Names in Asir Contrib

Not yet written.

Not yet written.

3 Asir-contrib for Windows

A part of Asir-contrib works on Windows. The following functions and components work on windows; the outer component sm1 and functions in asir-contrib which do not call outer components. In the cygwin environement, the outer components sm1, phc work. The other outer components do not work.

The following functions do not work on Windows. Some of them work in the cygwin environment of Windows.

- `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 Basic (Standard Functions)

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 Numbers (Standard Mathematical Functions)

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_floor

number_floor(*X*)
:

Example:
```
number_floor(1.5);
```

5.0.4 number_imaginary_part

number_imaginary_part(*X*)
:

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

5.0.5 number_is_integer

number_is_integer(*X*)
:

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

5.0.6 number_real_part

number_real_part(*X*)
:

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

6 Calculus (Standard Mathematical Functions)

7 Series (Standard Mathematical Functions)

8 Hypergeometric Functions (Standard Mathematical Functions)

Not yet written

9 Matrix (Standard Mathematical Functions)

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_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]);
```

9.0.7 matrix_inverse

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

Example:

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

9.0.8 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]]);
```

9.0.9 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`

9.0.10 matrix_matrix_to_list

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

`References`
`matrix_list_to_matrix`

9.0.11 matrix_rank

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

Example:

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

9.0.12 `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]);
```

9.0.13 `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]);
```

9.0.14 `matrix_transpose`

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

`References`
`matrix_list_to_matrix`

10 Graphic (Standard Mathematical Functions)

11 Print (Standard Mathematical Functions)

11.0.1 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

11.0.2 print_em

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

Example:

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

11.0.3 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

11.0.4 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));
```

11.0.5 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));
```

11.0.6 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`

11.0.7 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` , (,)

11.0.8 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`

11.0.9 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`

11.0.10 print_terminal_form

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

11.0.11 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`

11.0.12 print_tfb_form

`print_tfb_form(S)`
: It transforms S to the tfb format.

Example:

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

11.0.13 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`

11.0.14 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`

12 Polynomials (Standard Mathematical Functions)

12.0.1 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]);
```

12.0.2 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_*`

12.0.3 poly_factor

`poly_factor(F)`
: It factorizes the polynomial F.

Example:

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

12.0.4 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);
```

12.0.5 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]);
```

12.0.6 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]);
```

12.0.7 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]);
```

12.0.8 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]);
```

12.0.9 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]);
```

12.0.10 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]);
```

13 Complex (Standard Mathematical Functions)

14 OpenXM-Contrib General Functions

14.1 Functions

14.1.1 `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])]
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;
```

15 DSOLV Functions

This section is a collection of functions to solve regular holonomic systems in terms of series. Algorithms are explained in the book [SST]. You can load this package by the command `load("dsolv")$` This package requires `Diff` and `dmodule`.

To use the functions of the package `dsolv` in OpenXM/Risa/Asir, executing the command `load("dsolv")$` is necessary at first.

This package uses `ox_sm1`, so the variables you can use is as same as those you can use in the package `sm1`.

15.1 Functions

15.1.1 dsolv_dual

`dsolv_dual(`*f*`,`*v*`)`
:: Grobner dual of *f*.

return List

f, *v* List

- It returns the Grobner dual of *f* in the ring of polynomials with variables *v*.
- The ideal generated by *f* must be primary to the maximal ideal generated by *v*. If it is not primary to the maximal ideal, then this function falls into an infinite loop.

Algorithm: This is an implementation of Algorithm 2.3.14 of the book [SST]. If we replace variables x, y, ... in the output by log(x), log(y), ..., then these polynomials in log are solutions of the system of differential equations *f*`_(x->x*dx, y->y*dy, ...)`.

```
[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.
```

15.1.2 dsolv_starting_term

`dsolv_starting_term(f,v,w)`
: :: Find the starting term of the solutions of the regular holonomic system *f* to the direction *w*.

return List

f, *v*, *w* List

- Find the starting term of the solutions of the regular holonomic system *f* to the direction *w*.
- The return value is of the form [[*e1*, *e2*, ...], [*s1*, *s2*, ...]] where *e1* is an exponent vector and *s1* is the corresponding solution set, and so on.
- If you set `Dsolv_message_starting_term` to 1, then this function outputs messages during the computation.

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]]]
```

16 Graphic Library (2 dimensional)

The library glib provides a simple interface like old BASIC to the graphic primitive (draw_obj) of Risa/Asir.

16.0.1 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);
```

16.0.2 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.

16.0.3 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));
```

16.0.4 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);
```

16.0.5 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);
```

16.0.6 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`

16.0.7 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);
```

17 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:~$ 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("plot sin(x**2);");
0
```

The function `gnuplot_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

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

by the explicit scheme for $0 \le t \le dt * step$. 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; gnuplot_heat(0.001,30);   (CFL number is 0.4)
Heat_N=20; gnuplot_heat(0.003,30);   (CFL > 0.5 unstable)
```

Author of `GNUPLOT`: Thomas Williams, Colin Kelley

17.1 Functions

17.1.1 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`

17.1.2 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_plot_dots`

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

17.1.3 gnuplot_plot_dots

`gnuplot_plot_dots(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_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
```

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

17.1.4 gnuplot_heat

`gnuplot_heat(dt,step)`
:: It solves the heat equation numerical and plots solutions

return Void

dt floating point number

step Integer

- It solves the heat equation du/dt = d^2 u/dx^2, u(t,0) = u(t,1) = 0 with the initial condition u(0,x) = x (0 <= x <= 0.5), u(0,x) = 1-x (0.5 <= x <= 1.0).

- Heat_N is the number of the meshes in the space.
- This function will be called `pde_heat_demo` in a future.

Algorithm: NOT Written. (Difference scheme. Courant-Levi-Friedrichs conditions.)

```
[232] Heat_N = 20$
[233] gnuplot_heat(0.001,30)$
```

17.1.5 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`

17.1.6 gnuplot_plot_function

`gnuplot(`*`f`*`|proc=`*`p`*`)`
:: ask the `gnuplot` server to draw a graph of *f*

return Void

p Number

f Polynomial or a list of polynomials

- ask the `gnuplot` server to draw a graph of *f*

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

`Reference`
`gnuplot_to_gnuplot_format`

18 OpenMath Function Names (Version in 1999)

18.0.1 OpenMath CD Basic

OpenMath Function Names
:: Asir/OpenXM supports function names in function names following OpenMath Content Dictionaries (Version in 1999).

```
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_resultant(A,B,V) { return(res(V,A,B)); }
def poly_groebner(O,P,V)  { return(gr(P,V,O)); }
def poly_hilbert_polynomial(I)  { return(taka_poly_hilbert_polynomial(I)); }
```

19 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.

```
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. 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 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`.

19.1 Functions

19.1.1 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

19.1.2 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`

19.1.3 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`

20 OpenMath Functions(Version 1999)

The functions in this section is defined in the file'`om`'.

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

20.0.1 `om_start`

`om_start()`
:: Start `OMproxy` server to make a translation between CMO and OpenMath XML (CD's in 1999) 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>>
```

20.0.2 `om_xml`

`om_xml(`*s*`|proc=`*p*`)`
:: Translate CMO expression of s to a XML expression of OpenMath(CD's in 1999).

return String

p Number

s Object

- Translate CMO s to a XML expression of OpenMath(CD's in 1999).

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

20.0.3 `om_xml_to_cmo`

`om_xml_to_cmo(`*s*`|proc=`*p*`)`
:: Translate XML expression (CD's in 1999) *s* of OpenMath to a CMO.

return Object

p Number

s String

- Translate XML expression (CD's in 1999) *s* of OpenMath to a CMO.

21 Cohomology group associated to pFq

This section describes functions to evaluate invariants associated to (co)homology groups of the hypergeometric functions pFq ($_pF_q$)

In order to use the functions in this section in OpenXM/Risa/Asir, executing the commands

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

is necessary at first.

21.0.1 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);
```

21.0.2 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 pF_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

21.0.3 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

21.0.4 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.

22 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:~$ 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([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.

22.1 Functions

22.1.1 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`

22.1.2 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.

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([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)

23 Plucker Relations

23.0.1 plucker

Consider $(m+1) \times n$ matrix. The subsquare matrix consisting of $i_1, \ldots, i_m, j_k$ columns is denoted by $p_{i_1 \cdots i_m j_k}$. The Plucker relation is

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

This package provides functions for Plucker relations.

23.0.2 plucker_relation

plucker_relation(*L*,*M*)
:: Returns the plucker relation defined by the index sets L and M.

return quote

L List

M List

- L is the index set $i_1, \ldots, i_m$ of the plucker relations and M is the index set $j_0, \ldots, j_{m+1}$ of the plucker relations.

```
[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
```

23.0.3 plucker_y

plucker_y(*L*)
:: Returns the variable standing for the index L.

return Variable

L List

- Index set L is sorted and the sign is evaluated by the sorting.

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

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

23.0.4 plucker_index

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

Example:

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

24 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`.

All the coefficients of input polynomials should be integers for most functions in this section. Other functions accept rational numbers as inputs and it will be explicitely noted in each explanation of these functions.

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.

```
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.

24.1 `ox_sm1_forAsir` Server

24.1.1 `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.

24.2 Functions

24.2.1 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 might have to 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 cc for computation in sm1.
- The variables from a to z except d and o and x0, ..., x20, y0, ..., y20, z0, ..., z20 can be used as variables for ring of differential operators in default. (cf. Sm1_ord_list 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] cc*dcc;
dcc*cc
[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

24.2.2 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`, `Sm1_proc`.

24.2.3 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`

24.2.4 sm1_gb

`sm1_gb([f,v,w]|proc=p,sorted=q,dehomogenize=r)`
:: 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, q, r 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.
- When the optional variable *q* is set, `sm1_gb` returns, as the third return value, a list of the Grobner basis and the initial ideal with sums of monomials sorted by the given order. Each polynomial is expressed as a string temporally for now. When the optional variable *r* is set to one, the polynomials are dehomogenized (,i.e., h is set to 1).

```
[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 $1 \le \partial_y \le \partial_x \le y \le x \le \cdots$. 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).

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

Reference
    `sm1_reduction`, `sm1_rat_to_p`

24.2.5 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)

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.

24.2.6 sm1_hilbert

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

`hilbert_polynomial(`*f*`,`*v*`)`
: :: 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*.
- Note for sm1_hilbert: 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`.

```
[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
```

Reference
: sm1_start, sm1_gb, longname

24.2.7 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

24.2.8 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]]
```

24.2.9 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. `sm1_reduction_noH` is for the Weyl algebra.
- The return value is of the form [r,c0,[c1,...,cm],[g1,...gm]] where *g*=[g1, ..., gm] and c0 f + c1 g1 + ... + cm gm = r. 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],[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]]
```

`Reference`
  `sm1_start`, `sm1_find_proc`, `d_true_nf`

24.2.10 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);
basic_plus(basic_times(basic_power(x,4),1),basic_times(basic_power(x,0),-1))
```

`Reference`

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

24.2.11 `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]]]]
```

24.2.12 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]]
```

24.2.13 sm1_distraction

sm1_distraction([*f*,*v*,*x*,*d*,*s*] |proc=*p*)
:: ask the sm1 server to compute the distraction of *f*.

return List

p Number

f Polynomial

v,*x*,*d*,*s* List

- It asks the sm1 server of the descriptor number *p* to compute the distraction of *f* in the ring of differential operators with variables *v*.
- *x* is a list of x-variables and *d* is that of d-variables to be distracted. *s* is a list of variables to express the distracted *f*.
- Distraction is roughly speaking to replace x*dx by a single variable x. See Saito, Sturmfels, Takayama : Grobner Deformations of Hypergeometric Differential Equations at page 68 for details.

```
[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
```

`Reference`
    `distraction2(sm1)`,

24.2.14 sm1_gkz

`sm1_gkz([`*A*`,`*B*`]|proc=`*p*`)`
:: Returns the GKZ system (A-hypergeometric system) associated to the matrix *A* with the parameter vector *B*.

return List

p Number

A, *B* List

- Returns the GKZ hypergeometric system (A-hypergeometric system) associated to the matrix

```
[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]]
```

24.2.15 sm1_appell1

`sm1_appell1(a|proc=`*p*`)`
:: Returns the Appell hypergeometric system F_1 or F_D.

return List

p Number

a List

- Returns the hypergeometric system for the Lauricella function F_D(a,b1,b2,...,bn,c;x1,...,xn) where *a* =(a,c,b1,...,bn). When n=2, the Lauricella function is called the Appell function F_1. The parameters a, c, b1, ..., bn may be rational numbers.

```
[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
```

24.2.16 sm1_appell4

sm1_appell4(a|proc=*p*)
  :: Returns the Appell hypergeometric system F_4 or F_C.

return List

p Number

a List

- Returns the hypergeometric system for the Lauricella function F_4(a,b,c1,c2,...,cn;x1,...,xn) where a =(a,b,c1,...,cn). When n=2, the Lauricella function is called the Appell function F_4. The parameters a, b, c1, ..., cn may be rational numbers.

```
[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
```

24.2.17 sm1_rank

`sm1_rank(a |proc=p)`
:: Returns the holonomic rank of the system of differential equations a.

return Number

p Number

a List

- It evaluates the dimension of the space of holomorphic solutions at a generic point of the system of differential equations a. The dimension is called the holonomic rank.
- a is a list consisting of a list of differential equations and a list of variables.
- `sm1_rrank` returns the holonomic rank when a is regular holonomic. It is generally faster than `sm1_rank`.

```
[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
```

24.2.18 sm1_auto_reduce

`sm1_auto_reduce(s |proc=p)`
:: Set the flag "AutoReduce" to s.

return Number

p Number

s Number

- If *s* is 1, then all Grobner bases to be computed will be the reduced Grobner bases.
- If *s* is 0, then Grobner bases are not necessarily the reduced Grobner bases. This is the default.

24.2.19 sm1_slope

`sm1_slope(ii,v,f_filtration,v_filtration |proc=p)`
:: Returns the slopes of differential equations *ii*.

return List

p Number

ii List (equations)

v List (variables)

f_filtration List (weight vector)

v_filtration
List (weight vector)

- `sm1_slope` returns the (geometric) slopes of the system of differential equations *ii* along the hyperplane specified by the V filtration *v_filtration*.
- *v* is a list of variables.
- The return value is a list of lists. The first entry of each list is the slope and the second entry is the weight vector for which the microcharacteristic variety is not bihomogeneous.

Algorithm: see "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" Note that the signs of the slopes are negative, but the absolute values of the slopes are returned.

```
[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]);
```

`Reference`
`sm_gb`

24.2.20 `sm1_ahg`

`sm1_ahg(`*A*`)`
: It idential with sm1_gkz(A).

24.2.21 `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);
```

24.2.22 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]);
```

24.2.23 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]);
```

24.2.24 sm1_saturation

`sm1_saturation([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]]);
```

25 TIGERS Functions

This chapter describes interface functions for tigers ox server `ox_sm1_tigers`.

25.0.1 `tigers`

`tigers(a|proc=a)`

:: It asks the `tigers` server of the descriptor number *p* to enumerate all Grobner bases associated to the toric variaty defined by the matrix *a*.

return List

p Number

a List

- It asks the `tigers` server of the descriptor number *p* to enumerate all Grobner bases associated to the toric variaty defined by the matrix *a*.
- The system tigers is an expert system to enumerate all Gr\"obner bases of affine toric ideals. In other words, it can be used to determine the state polytope of a given affine toric ideal. As to a theoretical background, see the book
  B.Sturmfels, Grobner bases and Convex Polytopes.
  The original tigers is written by Birk Hubert. The algorithm used in explained in the paper
  B.Huber and R.Thomas, Computing Grobner Fans of Toric Ideals.

```
[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]]]
```

In this example, all reduced Grobner bases for the toric ideal associated to the matrix *A* are stored in *S*. There are eight distinct Grobner bases of *A*. [[i_1, i_2, ...],[j_1, j_2, ...]] is a set of exponents of two monomials and stands for a binomial. For example, the S[0] consists of
x1 x3 - x2^2, x1 x4 - x2 x3, x2 x4 - x3^2.
<x1 x3, x1 x4, x2 x4> is the initial ideal of S[0].

26 Utility Functions

Utility functions provide some usuful functions to access to the system and to process strings.

26.0.1 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");
```

26.0.2 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.

26.0.3 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.

26.0.4 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`

26.0.5 util_load_file_as_a_string

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

26.0.6 util_part

`util_part(`*S*`,`*P*`,`*Q*`)`
: It returns from *P*th element to *Q*th element of *S*.

26.0.7 util_read_file_as_a_string

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

26.0.8 util_remove_cr

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

26.0.9 util_v

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

Example:

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

References
util_index

26.0.10 util_write_string_to_a_file

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

27 Misc

This section describes functions that have not yet been classifed. These will be moved to independent sections in a future.

Index

(Index is nonexistent)

(Index is nonexistent)

Short Contents

Table of Contents