OpenXM/Risa/Asir-Contrib

OpenXM/Risa/Asir-Contrib User's Manual (English Edition)
Edition 1.3.2-3 for OpenXM/Asir2000
March 2017 (minor update on 27 March 2025)

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 `OpenXM/rc/asirrc`. 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 mathematical and utilitiy functions are outcome of the Asir-contrib project.

The latest asir-contrib manual of the HEAD branch is at `http://www.math.kobe-u.ac.jp/OpenXM/Current/doc/index-doc.html`

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)

2 How to load Asir/Contrib

With loading OpenXM/rc/asirrc, we can use most functions in Asir/Contrib. The OpenXM/Risa/Asir reads this file, which is specified by the ASIR_CONFIG environmental variable, when it starts. The file names.rr is the top level file of the Asir/Contrib. Most other files are loaded from names.rr. Some packages are not loaded from names.rr and they must be loaded individually.

A sample of asirrc to use Asir/Contrib.

```
load("gr")$
load("primdec")$
load("katsura")$
load("bfct")$
load("names.rr")$
load("oxrfc103.rr")$
User_asirrc=which(getenv("HOME")+"/.asirrc")$
if (type(User_asirrc)!=0)
   if (!ctrl("quiet_mode")) print("Loading ~/.asirrc")$
   load(User_asirrc)$
else $
end$
```

3 Function Names in Asir Contrib

Not yet written.

Not yet written.

4 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.*`
- `mathematica.*`
- `phc.*`
- `print_dvi_form`
- `print_gif_form`
- `print_open_math_xml_form`
- `print_png_form`
- `print_xdvi_form`
- `print_xv_form`
- `tigers_xv_form`

5 Basic (Standard Functions)

5.0.1 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)]);
```

5.0.2 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 $\{1,2,3\}$

5.0.3 base_f_definedp

base_f_definedp(*Func*)

: returns 1 if the function Func is defined.

5.0.4 base_flatten

base_flatten(*S*)

: It flattens a nested list S.

Example:

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

5.0.5 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]]);
```

5.0.6 base_is_asir2018

base_is_asir2018()

: returns 1 if the system is asir2018.

5.0.7 base_is_equal

base_is_equal(*L1*,*L2*)

: returns 1 if the objects L1 and L2 are equal else return 0

5.0.8 base_ith

`base_ith(A,I)`
: It returns A[I].

Example:

```
R=[[x,10],[y,20]]; map(base_ith,R,0);
```

5.0.9 base_makelist

`base_makelist(Obj,K,B,T)`
: base_makelist generate a list from Obj where K runs in [B,T]. Options are qt=1 (keep quote data), step (step size). When B is a list, T is ignored and K runs in B.

Example 0:

```
base_makelist(k^2,k,1,10);
```

Example 1:

```
map(print_input_form,base_makelist(quote(x^2),x,1,10 | qt=1, step=0.5))
```

Example 2:

```
base_makelist(quote("the "+k),k,["cat","dog"],0);
```

5.0.10 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]);
```

5.0.11 base_permutation

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

Example:

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

5.0.12 base_position

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

Example:

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

5.0.13 base_preplace

`base_preplace(S,Rule)`
: It rewrites *S* by using the rule *Rule*. psubst is used instead of subst. The replacement is not performed for function arguments.

Example:

```
base_preplace(exp(x)+x^2,[[x,a+1],[exp(x),b]]);
```

x is replaced by a+1 and exp(x) is replaced by b in exp(x)+x^2.

5.0.14 base_product

`base_product(Obj,K,B,T)`
: base_product returns the product of Obj where K runs in [B,T]. Options are qt=1 (keep quote data), step (step size). When B is a list, K runs in B and T is ignored.

Example 0:

```
base_product(k^2,k,1,10);
```

Example 1:

```
base_product(quote(x^2),x,1,10 | qt=1, step=0.5);
```

Example 2:

```
base_product(quote(x^2),x,[a,b,c],0 | qt=1);
```

5.0.15 base_prune

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

Example:

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

5.0.16 base_range

`base_range(Start,End)`
: It returns a list numbers [Start, Start+Step, Start+2*Step, ..., Start+n*Step] where Start+n*Step < End <= Start+(n+1)*Step Default value of step is 1.

`base_range(Start,End | step=Step=key0)`
: This function allows optional variables *step=Step*

Example:

```
base_range(0,10);
```

5.0.17 base_rebuild_opt

`base_rebuild_opt(Opt)`
: It rebuilt the option list *Opt*

Example:

```
base_rebuild_opt([[key1,1],[key2,3]] | remove_keys=["key2"]);
```

it returns [[key1,1]]

5.0.18 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.

5.0.19 base_replace_n

`base_replace_n(S,Rule)`
: It rewrites S by using the rule *Rule*. It is used only for specializing variables to numbers and faster than base_replace.

Example:

```
base_replace_n(x^2+y^2,[[x,1/2],[y,2.0+3*@i]]);
```

x is replaced by 1/2 and y is replaced by 2.0+3*in x^2+y^2.

5.0.20 base_rest

`base_rest(L)`
: It returns cdr(L).

Example:

```
R=[[x,10,30],[y,20,40]]; map(base_rest,R);
```

5.0.21 base_set_intersection

`base_set_intersection(A,B)`
: $A \cap B$

Example:

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

5.0.22 base_set_minus

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

Example:

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

5.0.23 base_set_union

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

Example:

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

5.0.24 base_subsequenceq

`base_subsequenceq(A,B)`
: if A is a subsequence B, then it returns 1 else 0.

Example:

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

5.0.25 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]);
```

5.0.26 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]);
```

5.0.27 base_sum

`base_sum(Obj,K,B,T)`
: base_sum returns the sum of Obj where K runs in [B,T]. Options are qt=1 (keep quote data), step (step size). When B is a list, K runs in B and T is ignored. When K is 0, then Obj is assumed to be a list or vector and Obj[B]+...+Obj[T] is returned.

Example 0:

```
base_sum(k^2,k,1,10);
```

Example 1:

```
base_sum(quote(x^2),x,1,10 | qt=1, step=0.5);
```

Example 2:

```
base_sum(quote(x^2),x,[a,b,c],0 | qt=1);
```

5.0.28 base_var_list

`base_var_list(Name,B,T)`
: base_var_list generate a list of variables Name+Index where Index runs on [B,T].

Example 0:

```
base_var_list(x,0,10);
```

Example 1:

```
base_var_list(x,1,4 | d = 1);
Options are d=1 (add d before the name).
```

6 Numbers (Standard Mathematical Functions)

6.0.1 number_abs

number_abs(*X*)
:

Example:

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

6.0.2 number_ceiling

number_ceiling(*X*)
:

Example:

```
number_abs(1.5);
```

6.0.3 number_eval

number_eval(*X*)
:

Example:

```
number_eval([1/10^10,@pi,exp(1)]);
```

6.0.4 number_factor

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

Example:

```
number_factor(20);
```

6.0.5 number_float_to_rational

number_float_to_rational(*X*)
:

Example:

```
number_float_to_rational(1.5234);
           number_setprec(30); //About 30 digits after the decimal point.  It also s
```

6.0.6 number_floor

number_floor(*X*)
:

Example:

```
number_floor(1.5);
```

6.0.7 number_imaginary_part

number_imaginary_part(*X*)
:

Example:

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

6.0.8 number_is_integer

number_is_integer(*X*)
:

Example:

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

6.0.9 number_real_part

number_real_part(*X*)
:

Example:

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

6.0.10 number_setprec

number_setprec(*X*)
: When X is 0, it returns the current value of precision.

Example:

```
 number_setprec(30);
  number_float_to_rational(F) returns
  an approximation of F by a rational number with the accuracy
  about 30 digits after the decimal point.
  It also calls setprec(30);
```

7 Calculus (Standard Mathematical Functions)

8 Series (Standard Mathematical Functions)

9 Special Functions (Standard Mathematical Functions)

Not yet written

10 Matrix (Standard Mathematical Functions)

10.0.1 `matrix_adjugate`

`matrix_adjugate(M)`

: It generates the adjugate matrix of the matrix M.

Example:

```
matrix_adjugate(matrix_list_to_matrix([[a,b],[c,d]]));
```

10.0.2 `matrix_clone`

`matrix_clone(M)`

: It generates the clone of the matrix M.

Example:

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

10.0.3 `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]]));
```

10.0.4 `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`

10.0.5 `matrix_eigenavalues`

`matrix_eigenavalues(M)`

: It returns the eigenvalues of the matrix M. if the option num=1, it returns the numerical approximate eigenvalues.

Example:

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

10.0.6 `matrix_gauge_transformation`

`matrix_gauge_transformation(M,T,V)`

: It returns T^(-1) M T - T^(-1) dT/dV

Example:

```
matrix_gauge_transformation([[0,x],[1,x]],[[x,0],[0,1]],x);
```

10.0.7 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`

10.0.8 matrix_ij

`matrix_ij(N,II,JJ)`
: It returns the matrix for exchanging II-th row(col) and JJ-th row(col).

Example:

```
matrix_ij(4,0,2);
```

10.0.9 matrix_image

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

Example:

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

References:

`matrix_kernel`

10.0.10 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]);
```

10.0.11 matrix_inverse

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

Example:

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

10.0.12 matrix_inverse_singular

`matrix_inverse_singular(Mat)`
: It returns a quasi-inverse matrix of Mat when it has 0-row and 0-column.

Example:

```
matrix_inverse_singular(newmat(3,3,[[1,0,2],[0,0,0],[3,0,4]]));
```

10.0.13 matrix_is_zero

`matrix_is_zero(A)`
: If it is 0 matrix or 0 vector or list consisting of 0, then it returns 1 else it returns 0.

Example:

```
matrix_is_zero(newmat(2,3));
```

10.0.14 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]]);
```

10.0.15 matrix_kronecker_product

`matrix_kronecker_product(A,B)`
: Kronecker product of the matrices A and B.

Example:

```
matrix_kronecker_product([[a11,a12],[a21,a22]],[[b11,b12],[b21,b22]]);
```

10.0.16 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`

10.0.17 matrix_matrix_to_list

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

References:

`matrix_list_to_matrix`

10.0.18 matrix_ones

`matrix_ones(N)`
: It returns the vector [1 1 ... 1] of length N. When one=m, it returns [m m ... m]. When size=[p,q] is given, N is ignored and returns p by q matrix with entries 1.

`matrix_ones(N | one=m=key0,size=[p=key1,q]=key2)`
: This function allows optional variables *one=m*, *size=[p, q]*

Example:

```
vtol(matrix_ones(3));  returns the list [1,1,1]
```

10.0.19 matrix_poly_to_matrix

`matrix_poly_to_matrix(`*Poly*`,`*Rule*`)`
: Replace variables in the polynomial Poly by matrices in the Rule.

Example:

```
matrix_poly_to_matrix(x^2-1,[[x,newmat(2,2,[[2,0],[0,3]])]]);
```

10.0.20 matrix_rank

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

Example:

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

10.0.21 matrix_rank_ff

`matrix_rank_ff(`*Mat*`,`*P*`)`
: It evaluates the rank of the matrix Mat by mod P. Entries may be rational numbers, and the inverse of the denominator D in F_P is properly computedd when P does not divide D, but the case P divides D does not raise an error.

10.0.22 matrix_row_matrix

`matrix_row_matrix(`*L*`)`
: It returns 1*n matrix [[L,L,...,L]] when L is a scalar. It returns 1*length(L) matrix [L].

`matrix_row_matrix(`*L* `|` *size*=*n*`=key0)`
: This function allows optional variables *size*=*n*

Example:

```
matrix_row_matrix(1 | size=5);
```

10.0.23 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]);
```

10.0.24 matrix_stack

`matrix_stack(`*A*`,`*B*`)`
: Stack the matrices A and B.

Example:

```
matrix_stack([[a11,a12],[a21,a22]],[[b11,b12],[b21,b22]]);
```

10.0.25 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]);
```

10.0.26 matrix_transpose

`matrix_transpose(`*M*`)`

: It returns the transpose of the matrix *M*.

References:

`matrix_list_to_matrix`

11 Graphic (Standard Mathematical Functions)

Not yet written.

12 Print (Standard Mathematical Functions)

12.0.1 print_c_form

`print_c_form(S)`

: It transforms S to the C format or python format string.

Example 0:

```
print_c_form(x^2+1);
```

Example 1:

```
print_c_form(x^2+1 | mode=python);
```

Example 2:

```
print_c_form(sin(x^2+1)/5 | mode=c);
```

12.0.2 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`

12.0.3 print_em

`print_em(S)`

: It outputs S by a font to emphasize it.

Example:

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

12.0.4 print_format

`print_format(S)`

: It changes the list format of S. Options are list, sep. Defaults are list=["",""], sep=",".

Example 0:

```
print_format([1,[x,y^2]]);
```

Example 1:

```
print_format([1,[x,y^2]] | list=["(",")"], sep=" ");
```

Example 2:

```
print_format(print_c_form([1,[x,y^2]]) );
```

12.0.5 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`

12.0.6 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));
```

12.0.7 print_open_math_tfb_form

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

Description:

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));
```

12.0.8 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`

12.0.9 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` , (,)

12.0.10 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`

12.0.11 print_pdf_form

`print_pdf_form(S)`
: It transforms S to a pdf file and previews the file.

Example 0:

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

Example 1:

```
print_pdf_form(poly_factor(x^10-1));
```

Optinal variabes: nopreview=1 does not preview the PDF file.

References:

`print_tex_form` , `print_xdvi_form`

12.0.12 print_png_form

`print_png_form(S)`
: It transforms S to a file of the format png. dvipng should be installed.

Example:

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

References:

`print_tex_form`

12.0.13 print_terminal_form

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

12.0.14 print_tex_form

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

`print_tex_form(S | table=key0,raw=key1)`
: This function allows optional variables *table*, *raw*

Description:

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. when AMSTeX = 1, "begin pmatrix" and "end pmatrix" will be used to output matrix.

References:

`print_xdvi_form`

12.0.15 print_tfb_form

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

Example:

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

12.0.16 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) | texstr=1);
```

References:

`print_tex_form` , `print_dvi_form`

12.0.17 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`

13 Polynomials (Standard Mathematical Functions)

13.0.1 poly_coefficient

poly_coefficient(*F*,*Deg*,*V*)
: It returns the coefficient of V^Deg in F. F may be rational or list or vector.

Example:

```
F=[(x+y+z)^10/z^2,(x-y+z)^10/z^3]$
poly_coefficient(F,10,x);
```

13.0.2 poly_coefficients_list

poly_coefficients_list(*F*,*V*)
: It returns the list of coefficients of F with respect to the variable list V. F may be rational or list or vector.

Example:

```
F=[(x+y+c*z)^2/c^2,(x-y+c*z)^2/c^3]$
poly_coefficients_list(F,[x,y,z]);
```

13.0.3 poly_coefficients_of_monomial_list

poly_coefficients_of_monomial_list(*F*,*VV*)
: It returns the list of coefficients of F with respect to a list of monomials VV.

Example:

```
poly_coefficients_of_monomial_list(2+3*x+4*z,[1,x,y,z]);
poly_coefficients_of_monomial_list((x+z)^3+5*y,[1,x,y,z,x^2*z]);
poly_coefficients_of_monomial_list([(x+y)^3,x+y],[x,x^2,x^3,x^2*y,x*y^2,y^3]);
```

References:

poly_construct_from_coefficients_of_monomial_list

13.0.4 poly_construct_from_coefficients_of_monomial_list

poly_construct_from_coefficients_of_monomial_list(*L*,*VV*)
: It returns the inner product of L and VV.

Example:

```
L=tk_poly_coefficients_of_monomial_list((x+y)^3,VV=[x,x^2,x^3,x^2*y,x*y^2,y^3]);
poly_construct_from_coefficients_of_monomial_list(L,VV);
```

References:

poly_coefficients_of_monomial_list

13.0.5 poly_dact

`poly_dact(Op,F,XL)`
: Act the differential operator Op to F. XL is a list of x variables.

Example:

```
poly_dact( x*dx+y*dy+a, x^(-3)*y^(-2), [x,y]);
```

13.0.6 poly_decompose_by_weight

`poly_decompose_by_weight(F,V,W)`
: decompose F into homogeneous components with respect to the variable V with the weight W. The return value is [[Max_ord,Min_ord],[component of Max_ord, ..., component of Min_ord]];

Example:

```
poly_decompose_by_weight(x^2*dx^2-x*(x*dx+y*dy+a),[x,y,dx,dy],[-1,-1,1,1]);
```

13.0.7 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*

Description:

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]);
```

13.0.8 poly_denominator

`poly_denominator(L)`
: It returns the denominator of L. L may be a list.

Example:

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

13.0.9 poly_diff2euler

`poly_diff2euler(Op,XL)`
: Express the differential operator Op by the euler operators. XL is a list of x variables. When XL=[x,y], dx,dy are differential operators and tx,ty are Euler operators (tx=x*dx, ty=y*dy). t stands for theta. When the return value is R, R[0]*R[1]=Op.

Example:

```
poly_diff2euler(dx^2-a*x,[x]);
```

13.0.10 poly_dmul

`poly_dmul(Op1,Op2,XL)`

: Multiply Op1 and Op2 in the Weyl algebra (the ring of differential operators). XL is a list of x variables.

Example:

```
poly_dmul( x*dx+y*dy+a*x, x*y*dx*dy, [x,y]);
```

13.0.11 poly_dvar

`poly_dvar(V)`

: Add d to the variable name *V*.

Example:

```
poly_dvar([x1,x2,x3]);
        poly_dvar([x1,x2,x3] | d=t);
```

13.0.12 poly_elimination_ideal

`poly_elimination_ideal(I,VV)`

: It computes the intersection of the ideal *I* and the subring K[*VV*].

`poly_elimination_ideal(I,VV | grobner_basis=key0,gb=key1,v=key2,homo=key3,grace=key4,strategy=key5)`

: This function allows optional variables *grobner_basis*, *gb*, *v*, *homo*, *grace*, *strategy*

Description:

If grobner_basis is "yes" or gb=1, 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");
When strategy=1(default),
  nd_gr is used when trace=0(defauult),
  nd_gr_trace is used when trace=1.
```

References:

`gr` , `hgr` , `gr_mod` , `dp_*`

13.0.13 poly_euler2diff

`poly_euler2diff(Op,XL)`

: Translate the differential operator Op expressed in terms of euler operators into the operators in terms of d. XL is a list of x variables. When XL=[x,y], dx,dy are differential operators and tx,ty are Euler operators (tx=x*dx, ty=y*dy). t stands for theta.

Example:

```
poly_euler2diff(tx^2-x*(tx+1/2)^2,[x]);
```

13.0.14 poly_expand

`poly_expand(F)`
: This is an alias of poly_sort.

References:

`poly_sort`

13.0.15 poly_factor

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

Example:

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

13.0.16 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);
```

13.0.17 poly_gr_w

`poly_gr_w(F,V,W)`
: It returns the Grobner basis of F for the weight vector W. It is the second interface for poly_grobner_basis.

Example:

```
poly_gr_w([x^2+y^2-1,x*y-1],[x,y],[1,0]);
```

References:

`poly_in_w` , `poly_grobner_basis`

13.0.18 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*

Description:

The optional variable v is a list of variables which defines the ring of polynomials. Other Options; p (characteristic), homo, method (nd_gr_trace(default), nd_gr, nd_weyl_gr, nd_weyl_gr_trace, nd_f4, nd_f4_trace), order_matrix, order. See also asir manual. alias; poly_groebner_basis

Example:

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

```
A->Ring->Variables;
A->Ring->Order;
B = poly_grobner_basis([x^2+y^2-4,x*y-1]|order=[[10,1]],v=[y,x]);
C = poly_grobner_basis([x^2+y^2-4,x*y-1]|order=[block,[0,1],[0,1]],v=[y,x]);
```

13.0.19 poly_hilbert_polynomial

`poly_hilbert_polynomial(I)`
: It returns the Hilbert polynomial of the poly_init(I).

`poly_hilbert_polynomial(I | s=key0,v=key1,sm1=key2)`
: This function allows optional variables *s*, *v*, *sm1*

Description:

The optional variable v is a list of variables. sm1=1 forces to call sm1. [sum(H(k),k,0,h), H(h)] where H(h) is the number of degree h monomials when h>>0. On asir2018, it returns [sum(H(k),k,0,h), H(h),[H[0],H[1],...],F,d] where F/(1-h)^d is the Poincare series.

Example:

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

13.0.20 poly_ideal_colon

`poly_ideal_colon(I,J,V)`
: It computes the colon ideal of *I* by *J* *V* is the list of variables.

Example:

```
B=[(x+y+z)^50,(x-y+z)^50]$
V=[x,y,z]$
B=poly_ideal_colon(B,[(x+y+z)^49,(x-y+z)^49],V);
```

13.0.21 poly_ideal_intersection

`poly_ideal_intersection(I,J,V,Ord)`
: It computes the intersection of the ideal *I* and *J* *V* is the list of variables. *Ord* is the order.

Example:

```
A=[j*h*g*f*e*d*b,j*i*g*d*c*b,j*i*h*g*d*b,j*i*h*e*b,i*e*c*b,z]$
B=[a*d-j*c,b*c,d*e-f*g*h]$
V=[a,b,c,d,e,f,g,h,i,j,z]$
poly_ideal_intersection(A,B,V,0);
```

13.0.22 poly_ideal_saturation

`poly_ideal_saturation(I,J,V)`
: It computes the saturation ideal of I by J. *V* is the list of variables.

Example:

```
B=[(x+y+z)^50,(x-y+z)^50]$
V=[x,y,z]$
B=poly_ideal_saturation(B,[(x+y+z)^49,(x-y+z)^49],V);
```

13.0.23 poly_in

poly_in(*I*)
: It is an alias of poly_initial().

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

Example:

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

13.0.24 poly_in_w

poly_in_w(*F*,*V*,*W*)
: It returns the initial term or the initial ideal in_w(F) for the weight vector given by order. F is s single polynomial or a list of polynomials.

poly_in_w(*F*,*V*,*W* | *gb*=key0)
: This function allows optional variables *gb*

Example:

```
poly_in_w([x^2+y^2-1,x*y-x] , [x,y], [1,0]);
```

References:

poly_weight_to_omatrix , poly_grobner_basis , poly_gr_w , poly_in_w_

13.0.25 poly_in_w_

poly_in_w_(*F*)
: It returns the initial term or the initial ideal in_w(F) for the weight vector given by order. F is s single polynomial or a list of polynomials. This is a new interface of poly_in_w with shorter args.

poly_in_w_(*F* | *v*=key0,*weight*=key1,*gb*=key2)
: This function allows optional variables *v*, *weight*, *gb*

Example:

```
poly_in_w_([x^2+y^2-1,x*y-x] | v=[x,y],weight=[1,0]);
```

References:

poly_weight_to_omatrix , poly_grobner_basis , poly_gr_w

13.0.26 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*

Description:

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

13.0.27 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*

Description:

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]);
```

13.0.28 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*

Description:

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]);
```

13.0.29 poly_lcm

`poly_lcm(L)`
: It returns the LCM of L[0], L[1], ...

Example:

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

13.0.30 poly_numerator

`poly_numerator(L)`
: It returns the numerator of L. L may be a list.

Example:

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

13.0.31 poly_ord_w

`poly_ord_w(F,V,W)`
: It returns the order with respect to W of F.

Example:

```
poly_ord_w(x^2+y^2-1,[x,y],[1,3]);
```

References:

`poly_in_w`

13.0.32 poly_prime_dec

`poly_prime_dec(I,V)`
: It computes the prime ideal decomposition of the radical of I. *V* is a list of variables.

Example:

```
B=[x00*x11-x01*x10,x01*x12-x02*x11,x02*x13-x03*x12,x03*x14-x04*x13,
        -x11*x20+x21*x10,-x21*x12+x22*x11,-x22*x13+x23*x12,-x23*x14+x24*x13];
        V=[x00,x01,x02,x03,x04,x10,x11,x12,x13,x14,x20,x21,x22,x23,x24];
        poly_prime_dec(B,V | radical=1);
```

13.0.33 poly_r_omatrix

`poly_r_omatrix(N)`
: It gives a weight matrix, which is used to compute a Grobner basis in K(x)<dx>, |x|=|dx|=N.

Example:

```
poly_r_omatrix(3);
When the option lex is given, the last lex variables are
compared firstly by the lexicographic order, e.g.,
poly_r_omatrix(4 | lex=2) is compared by the matrix
 0 0 0 0   0 0 0 1
 0 0 0 0   0 0 1 0
 0 0 0 0   1 1 0 0
 ....
```

References:

`poly_weight_to_omatrix`

13.0.34 poly_replace_factor

poly_replace_factor(*F*,*Rule*)
: It factorizes F and replaces factors by the Rule.

Example:

```
poly_replace_factor(2*x/((x-y)^3*y), [[x-y,s]]);
  It returns 2*x/(s^3*y).
```

13.0.35 poly_solve_linear

poly_solve_linear(*Eqs*,*V*)
: It solves the system of linear equations *Eqs* with respect to the set of variables *V*. When the option p=P is given, it solves the system by mod P. When the option reverse=1 is given, the lex order of reverse(V) is used.

Example:

```
poly_solve_linear([2*x+3*y-z-2, x+y+z-1], [x,y,z]);
         poly_solve_linear([2*x+3*y-z-2, x+y+z-1], [x,y,z] | p=13);
```

13.0.36 poly_sort

poly_sort(*F*)
: It expands *F* with a given variables *v*=V and a given weight *w*=W. It returns a quote object. If *truncate* option is set, the expansion is truncated at the given degree.

poly_sort(*F* | *v*=key0,*w*=key1,*truncate*=key2)
: This function allows optional variables *v*, *w*, *truncate*

Example:

```
poly_sort((x-y-a)^3 | v=[x,y], w=[-1,-1])
   returns a series expansion in terms of x and y.
```

13.0.37 poly_subsetq

poly_subsetq(*II*,*JJ*,*V*)
: If the ideal II is contained in the ideal JJ, it returns 1, else 0.

Example:

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

Optinal variabes: gb=1 (if JJ is already a GB). verbose=1 Note that when gb=1, the order must not be changed since the GB of JJ was computed. Otherwise, this function does not give correct answer or stucks. If gb=1 is not given, dp_ord(0) is executed in this function.

13.0.38 poly_toric_ideal

poly_toric_ideal(*A*,*V*)
: It returns generators of the affine toric ideal defined by the matrix(list) A. V is the list of variables.

Example:

```
poly_toric_ideal([[1,1,1,1],[0,1,2,3]],base_var_list(x,0,3));
```

Optinal variabes: nk_toric=1 (disable 4ti2)

13.0.39 poly_w_marking

`poly_w_marking(Id,V,W)`
: The monomials x^a in Id is rewritten to x^a*t_w^(<a,w>+b). <a,w> is the inner product and b is an integer to avoid negative powers of t_w. Return value is [w-marked polynomial, b]

Example:

```
poly_w_marking(x*dx^2+y*dy+a,[x,y,dx,dy],[-1,-1,1,1]);
  [t_w*x*dx^2+y*dy+a,0]
```

Optinal variabes: specify a name of homogenization variable by the option hvar. The default is t_w.

13.0.40 poly_weight_to_omatrix

`poly_weight_to_omatrix(W,V)`
: [obsoleted] It translates the weight vector *W* into a matrix, which is used to set the order in asir Grobner basis functions. *V* is the list of variables.

Example:

```
M=poly_weight_to_omatrix([2,1,0],[x,y,z]);
         nd_gr([x^3+z^3-1,x*y*z-1,y^2+z^2-1,[x,y,z],0,M);
```

13.0.41 poly_weight_to_ord_matrix

`poly_weight_to_ord_matrix(W)`
: Weight vector W is transformed to a matrix defined order for dp_ord, nd_gr, ... It is a new version of poly_weight_to_omatrix(W,V) [obsoleted]

Example:

```
Mat=poly_weight_to_ord_matrix([1,1,1,1,0,1,1,1,1,0]);
Mat=poly_weight_to_ord_matrix([]|tie_breaker=[lex,0,1,2,3,5,6,7,8,4,9]);
```

Optinal variabes: tie_breaker=[lex,n1,n2,n3,...] defines the lexcographic order x_n1,x_n2, x_n3, ... when variables are x_*

13.0.42 poly_weyl_subsetq

`poly_weyl_subsetq(II,JJ,V)`
: If the ideal II in the Weyl algebra is contained in the ideal JJ, it returns 1, else 0.

Example:

```
poly_weyl_subsetq([x*dx^2],[x*dx-1],[x,dx]);
```

Optinal variabes: gb=1 (if JJ is already a GB). verbose=1. Note that when gb=1, the order must not be changed since the GB of JJ was computed. Otherwise, this function does not give correct answer or stucks. If gb=1 is not given, dp_ord(0) is executed in this function.

14 Complex (Standard Mathematical Functions)

15 Graphic Library (2 dimensional)

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

15.0.1 glib_clear

`glib_clear()`
: Clear the screen.

15.0.2 glib_flush

`glib_flush()`
: ; Flush the output. (Cfep only. It also set initGL to 1.).

15.0.3 glib_line

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

`glib_line(X0,Y0,X1,Y1 | color=key0,shape=key1)`
: This function allows optional variables *color*, *shape*

Example:

```
glib_line(0,0,5,3/2 | color=0xff00ff);
          glib_line(0,0,10,0 | shape=arrow);
```

15.0.4 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.

15.0.5 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));
```

15.0.6 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);
```

15.0.7 glib_ps_form

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

Example 0:

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

Example 1:

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

References:

`glib_tops`

15.0.8 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);
```

15.0.9 glib_remove_last

`glib_remove_last()`
: Remove the last object. glib_flush() should also be called to remove the last object. (cfep only).

15.0.10 glib_set_pixel_size

`glib_set_pixel_size(P)`
: Set the size of putpixel to P. 1.0 is the default. (cfep only).

15.0.11 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`

15.0.12 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);
```

16 OpenXM-Contrib General Functions

16.1 Functions

16.1.1 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.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;
```

17 OXshell Functions

OXshell is a system to execute system commands from ox servers. As to details, see the files OpenXM/src/kan96xx/Doc/oxshell.oxw and OpenXM/doc/Papers/rims-2003-12-16-ja.tex.

17.0.1 oxshell.get_value

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

Example:

```
oxshell.set_value("abc","Hello world!");
         oxshell.oxshell(["cp", "stringIn://abc", "stringOut://result"]);
         oxshell.get_value("result");
  What we do is a file $TMP/abc* is generated with the contents Hello world! and copi
  The contents of the file is stored in the variable result on ox_sm1.
```

References:

`oxshell.oxshell` , `oxshell.set_value`

17.0.2 oxshell.oxshell

`oxshell.oxshell(L)`
: It executes command *L* on a ox_shell server. *L* must be an array. The result is the outputs to stdout and stderr. A temporary file will be generated under $TMP. cf. oxshell.keep_tmp()

Example:

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

References:

`ox_shell` , `oxshell.set_value` , `oxshell.get_value` , `oxshell` , `of` , `sm1.`

17.0.3 oxshell.set_value

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

Example:

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

References:

`oxshell.oxshell` , `oxshell.get_value`

18 Asir System Utility Functions

18.0.1 asir_contrib_update

`asir_contrib_update()`

: It updates the asir-contrib library and/or some other files to the HEAD branch. The usage will be shown by asir_contrib_update() without the option update. Options are update, clean, url, install_dir, zip_files, tmp. Default values update=0, clean=0, url="http://www.math.kobe-u.ac.jp/OpenXM/Current", install_dir=%APPDATA%/OpenXM (win) or install_dir=$OpenXM_tmp/OpenXM (others) zip_files=["lib-asir-contrib.zip"]

Example:

```
asir_contrib_update();
asir_contrib_update(|update=1);    update the library
asir_contrib_update(|update=3);    update the library and the documents
asir_contrib_update(|clean=1);
asir_contrib_update(|zip_files=["lib-asir-contrib.zip","doc-asir2000.zip","doc-asir
```

19 Utility Functions

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

19.0.1 `util_damepathq`

`util_damepathq(`*S*`)`

: When *S* is a string by the ShiftJIS code and *S* contains dame-moji with respect to \, it returns [a non-zero number, the string].

Example:

```
T = [0x5c,0xe4,0x5c,0x41,0x42]$
T2=asciitostr(T)$
util_damepathq(T2);
```

19.0.2 `util_file_exists`

`util_file_exists(`*Fname*`)`

: It returns 1 when *Fname* exists. It returns 0 when *Fname* does not exist.

19.0.3 `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");
```

19.0.4 `util_find_and_replace`

`util_find_and_replace(`*W*`,`*S*`,`*Wnew*`)`

: It replaces *W* in *S* by *Wnew*. Arguments must be lists of ascii codes or strings.

19.0.5 `util_find_start`

`util_find_start()`

: It tries to find the gnome-open command or an installed browser in unix systems. It returns "open" on MacOS X and returns "start" on Windows.

`util_find_start( |` *browser*`=key0)`

: This function allows optional variables *browser*

19.0.6 `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 lists of ascii codes or strings.

19.0.7 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`

19.0.8 util_load_file_as_a_string

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

19.0.9 util_part

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

19.0.10 util_read_file_as_a_string

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

19.0.11 util_remove_cr

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

19.0.12 util_timing

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

Example:

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

19.0.13 util_v

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

Example:

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

References:

`util_index`

19.0.14 util_write_string_to_a_file

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

20 Other Manuals

This section introduces other manuals in the asir-contrib project.

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

20.0.1 dsolv (Solving the initial ideal for holonomic systems)

../dsolv-html/dsolv-en.html

20.0.2 gtt_ekn (Two way contingency tables by HGM)

../gtt_ekn-html/gtt_ekn-en.html

20.0.3 f_res (Comuting resultant)

../f_res-html/f_res-en.html

20.0.4 (gnuplot ox server for graphics)

../gnuplot-html/gnuplot-en.html

20.0.5 mathematica (Mathematica (TM) ox server)

../mathematica-html/mathematica-en.html

20.0.6 mt_graph (3D grapher)

../mk_graph-html/mk_graph-en.html

20.0.7 mt_gkz (Intersection matrix of GKZ systems)

../mt_gkz-html/mt_gkz-en.html

20.0.8 mt_mm (Macaulay matrix method)

../mt_mm-html/mt_mm-en.html

20.0.9 n_wishartd (restriction of matrix 1F1)

../n_wishartd-html/n_wishartd-en.html

20.0.10 nn_ndbf (local b-function)

../nn_ndbf-html/nn_ndbf-en.html

20.0.11 noro_mwl (Mordel Weil Lattice)

../noro_mwl-html/noro_mwl-en.html

20.0.12 noro_module_syz (syzygies for modules)

../noro_module_syz-html/noro_module_syz-en.html

20.0.13 ns_twistedlog (twisted logarithmic cohomology group)

../ns_twistedlog-html/ns_twistedlog-en.html

20.0.14 nk_fb_gen_c (Fisher Bingham MLE)

`../nk_fb_gen_c-html/nk_fb_gen_c-en.html`

20.0.15 ok_diff (Okutani's library for differential operators)

`../ok_diff-html/ok_diff-en.html`

20.0.16 ok_dmodule (Okutani's library for D-modules)

`../ok_dmodule-html/ok_dmodule-en.html`

20.0.17 om (om (java) ox server for translating CMO and OpenMath)

`../om-html/om-en.html`

20.0.18 ox_pari (OpenXM pari server)

`../ox_pari-html/ox_pari-en.html`

20.0.19 (Plucker relations)

`../plucker-html/plucker-en.html`

20.0.20 pfpcoh (Ohara's library for homology/cohomology groups for p F q)

`../pfpcoh-html/pfpcoh-en.html`

20.0.21 phc (PHC ox server for solving systems of algebraic equations by the homotopy method)

`../phc-html/phc-en.html`

20.0.22 sm1 (Kan/sm1 ox server for the ring of differential operators)

`../sm1-html/sm1-en.html`

20.0.23 tigers (tigers ox server for toric universal Grobner bases)

`../tigers-html/tigers-en.html`

20.0.24 tk_ode_by_mpfr (Generating C codes for numerical analysis of ODE by MPFR, document in Japanese)

`../tk_ode_by_mpfr-html/tk_ode_by_mpfr-ja.html`

20.0.25 [[todo_parametrize]]

todo_parametrize/todo_parametrize_toc

With loading the file `todo_parametrize/todo_parametrize.rr` the function `paramerize` is installed. The function finds a parametric expression of a given rational curve. As to details, see See *A package for algebraic curves* (in Japanese).

```
[1205] load("todo_parametrize/todo_parametrize.rr");
```

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

20.0.26 taji_alc

With loading the file `taji_alc.rr` functions for algebraic local cohomology groups in one variable are imported.

```
import("taji_alc.rr");
taji_alc.laurent_expansion(x,(x-1)^3);
```

20.0.27 Manual and papers which are not written in texinfo.

Links to manuals and papers related to files and commands in asir-contrib are at OpenXM documents (`http://www.math.kobe-u.ac.jp/OpenXM/Current/doc/index-doc-en.html`).

Index

(Index is nonexistent)

(Index is nonexistent)

Short Contents

Table of Contents