os_muldif

Toshio Oshima

Aug 22, 2022

Contents

os_muldif.rr for Risa/Asir

A library for computing (ordinary/partial) differential operators

by Toshio Oshima

This note is a product of an on-going project translating the manual `os_muldif.pdf` of a library `os_muldif.rr` of `Risa\Asir` into in English. Currently this note contains only a small part of `os_muldif.pdf` related to the functions which analyze linear ordinary differential equations with polynomial coefficients.

The explanation of the functions without a short explanation starting with "::" is not yet translated. The number at the head of the name of a function coincides with the one in `os_muldif.pdf`. The number skipped in this note corresponds to a function which is not yet explained in this note.

1 List of Functions

1.1 Functions related to differential operators

The following functions are in a module and we call the functions by putting `os_md.` at the head such as `os_md.muldo()`.

1.1.1 Fundamental functions

1. `muldo`(p_1,p_2,[x,∂_x]|lim=n) or `muldo`(p_1,p_2,x|lim=n)
   `muldo`(p_1,p_2,[[x_1,∂_{x_1}],[x_2,∂_{x_2}],...]|lim=n)
   :: Returns the product of (a matrix of) ordinary (partial) differential operators whose coefficients are rational functions (or elementary functions) ($\Leftarrow [\partial_x, x] = 1$)
2. `caldo`([p_1,p_2,...],[x,∂_x])
   :: Returns the sum of products of differential operators
3. `muledo`(p_1,p_2,[x,∂_x]) or `muledo`(p_1,p_2,x)
   :: Returns the product (matrices) of differential operators of Euler type ($\Leftarrow [\partial_x, x] = x$)
4. `transpdo`(p,[[x_1,∂_{x_1}],[x_2,∂_{x_2}],...],[[y_1,∂_{y_1}],[y_2,∂_{y_2}],...]|ex=1,inv=f)
   :: Transformation of differential operator p by $x_i \mapsto y_i = y_i(x)$, $\partial_{x_j} \mapsto \partial_{y_j} = c_j(x) + \sum_\nu a_{j\nu}(x)\partial_{x_\nu}$
5. `translpdo`(p,[[x_1,∂_{x_1}],[x_2,∂_{x_2}],...],mat)
   :: Transforms a differential operator p by the coordinate transformation $x_i \mapsto \sum_j (mat)_{ij} x_j$
6. `transppow`([[x_1,∂_{x_1}],...,[x_n,∂_{x_n}]],m)
   :: Coordinate transformations by products of powers of coordinates
7. `appldo`(p,r,[x,∂_x]|Pfaff=1) or `appldo`(p,r,[[x_1,∂_{x_1}],[x_2,∂_{x_2}],...])
   :: Applies a (matrix of) differential operator to a (matrix of) rational function or elementary function or a Pfaffian system
8. `adj`(p,[x,∂_x]) or `adj`(p,[[x_1,∂_{x_1}],[x_2,∂_{x_2}],...])
   :: Returns the formal adjoint of a (matrix of) differential operator p
9. `sftpexp`(p,[x,∂_x],q,r) or `sftpexp`(p,[[x_1,∂_{x_1}],...],q,r])
   :: Returns $q^{-r} \circ p \circ q^r$
10. `appledo`(p,r,[x,∂_x])
    :: Applies a differential operator of Euler type to a rational function
11. `divdo`(p_1,p_2,[x,∂_x]|rev=1)
    :: Calculates divisions of ordinary differential operators
12. `mygcd`(p_1,p_2,[x,∂_x]|rev=1,dviout=n) or `mygcd`(p_1,p_2,[x]|rev=1,dviout=n)
    `mygcd`(p_1,p_2,x|dviout=n), `mygcd`(p_1,p_2,0|dviout=n)
    :: GCD of ordinary differential operators (or polynomials of x or positive integers) p_1 and p_2
13. `mylcm`(p_1,p_2,[x,∂_x]|rev=1) or `mylcm`(p_1,p_2,[x]|rev=1)

`mylcm(`p_1`,`p_2`,`x`)`, `mylcm(`p_1`,`p_2`,0)`
:: LCM of ordinary differential operators (or polynomials of x or positive integers) p_1 and p_2

14. `m1div(`m`,`n`,[`x`,`∂_x`])` or `m1div(`m`,`n`,[`x`])` or `m1div(`m`,`n`,`x`)`
:: Returns $R = [R[0], R[1]]$ such that $m = R[1](\partial_x - n) + R[0]$ (or $m = R[1](x - n) + R[0]$) for a square matrix m of differential operators with coefficients in rational functions (or polynomials of x)
15. `qdo(`p_1`,`p_2`,[`x`,`∂_x`])`
:: Returns the list $[q_1, q_2]$ of differential operators satisfying $q_1 p_2 u = 0$ and $q_2 p_2 u = u$ for a differential equation $p_1 u = 0$
16. `mdivisor(`m`,[`x`,`∂`]|trans=1,step=1,dviout=`t`)`
`mdivisor(`m`,`x`|trans=1,step=1,dviout=`t`)`, `mdivisor(`m`,0|trans=1,step=1,dviout=`t`)`
:: Elementary divisors of a matrix of ordinary differential operators/polynomials with coefficients in rational functions/integers
17. `sqrtdo(`p`,[`x`,`∂_x`])` ?
18. `toeul(`p`,[`x`,`∂_x`],`n`)`
:: Expresses a (Fuchsian) differential operator p in Euler type at $x = n$
19. `fromeul(`p`,[`x`,`p_x`],`n`)`
:: Transforms the differential operator p of Euler type into normal form (the above inverse)
20. `expat(`p`,[`x`,`∂_x`],`n`)`
:: Returns characteristic exponents of a differential operator p at a regular singular point $x = n$
21. `sftexp(`p`,[`x`,`∂_x`],`n`,`r`)`
:: Returns $(x - n)^{-r} \circ p \circ (x - n)^r$
22. `fractrans(`p`,[`x`,`∂_x`],`n_0`,`n_1`,`n_2`)`
:: Transforms p by the linear fractional transformation of x with $(n_0, n_1, n_2) \mapsto (0, 1, \infty)$
23. `chkexp(`p`,[`x`,`∂_x`],`n`,`r`,`m`)`
:: Returns condition so that p has the exponent $[r]_{(m)}$ at $x = n$
24. `soldif(`p`,[`x`,`∂_x`],`n`,`q`,`m`)`
25. `okuboetos(`p`,[`x`,`∂_x`]|diag=[`$c_1, c_2, \ldots$`])`
:: Transforms a single ODE of Okubo type to 1st order Okubo system
26. `stoe(`p`,[`x`,`dx`],`m`)`
:: Converts a first order system of ODE to a single ODE
27. `etos(`p`,[`x`,`dx`],`m`)`
:: Converts a single ODE to a first order system of ODE
28. `dform(`ℓ`,`x`|dif=1)`
:: Calculates differential 1-form $\sum \ell[i][0] d(\ell[i][1])$ or 2-form $\sum \ell[i][0] d(\ell[i][1]) \wedge d(\ell[i][2])$ with variables $x[0], x[1], \ldots$. Option `dif=1` means external derivative of 1-form
29. `solpokubo(`p`,[`x`,`∂_x`],`n`)`
:: Returns eigenvalues and eigenpolynomials of a single ODE of Okubo type

1.1.2 Fractional calculus

The functions in this section realize the results of [O3], [O5], [O7], [O9] and [O10].

30. `laplace(`p`,[`x, ∂_x`])` or `laplace(`p`,[[`x_1, ∂_{x_1}`],[`x_2, ∂_{x_2}`],...])`
:: (partial) Laplace transform of a differential operator p
31. `laplace1(`p`,[`x, ∂_x`])` or `laplace1(`p`,[[`x_1, ∂_{x_1}`],[`x_2, ∂_{x_2}`],...])`
:: Inverse of (partial) Laplace transform of a differential operator p
32. `mc(`p`,[`x, ∂_x`],`r`)`
:: Middle convolution $mc_r(p)$ of a differential operator p
33. `mce(`p`,[`x, ∂_x`],`n`,`r`)`
:: Transformation p into $(\partial_x - n)^{-r} \circ p \circ (\partial_x - n)^r$

34. rede(p,[x,∂_x]) or rede(p,[[x_1,∂_{x_1}],[x_2,∂_{x_2}],...])
:: Reduced representative of a differential operator p
35. ad(p,[x,∂_x],f)
:: Transform of a differential operator p defined by $\partial_x \to \partial_x - f$
36. add(p,[x,∂_x],f)
:: Addition of a differential operator p defined the map $\partial_x \to \partial_x - f$, namely, rede(ad())
37. vadd(p,[x,∂_x],[[c_0,r_0],[c_1,r_1],...])
:: Versal addition add(p,[x,∂_x], $\sum_{j\geq 0} \frac{r_j x^j}{\prod_{\nu=0}^{j}(1-c_\nu x)}$)
38. addl(p,[x,∂_x],f)
:: laplace1(add(laplace()))
39. cotr(p,[x,∂_x],f)
:: Transform of a differential operator p defined by $x \mapsto f(x)$
40. rcotr(p,[x,∂_x],f)
:: Reduced representative of the transformation P defined by $x \mapsto f(x)$
41. s2sp(p|num=1,std=k,short=1)
:: Converts a list of lists of numbers to and from its expression using a list of strings
42. s2csp(p|n=f)
:: Converts expressions of a spectral type with unramified irregular singularities
43. chkspt(m|mat=1,dumb=1) or chkspt(m|opt=t,dumb=1) or fspt(m,t)
:: Checks a tuple of partitions m (spectral type) or a generalized Riemann scheme (GRS) and returns [pts, ord, idx, $fuchs$, rod, $redsp$, $fspt$]
opt="sp", "basic", "construct", "strip", "short", "long", "sort", "idx"
44. spgen(n|eq=1,str=1,std=f,pt=[k,ℓ],sp=m,basic=1)
:: Gets rigid tuples of partitions with the rank $\leq n$ (or the orbit of a given tuple)
If $n \leq 0$, basic tuples with the index of the rigidity n are obtained.
45. sproot(p,t|dviout=1,only=k,sym=t,null=1)
:: Returns informations of the root corresponding to a spectral type t="base", "length", "type", "part", "pair", "pairs", sp
46. spbasic(k,d|str=1)
:: Returns the list of spectral types with rank d whose index of rigidity equals k
47. sp2grs(m,a,ℓ|mat=1)
:: Generates generalized Riemann scheme with a given spectral type
48. ssubgrs (m,ℓ)
49. mcgrs(m,[$r_1,r_2,\ldots,r_n$]|mat=1,slm=[[k_1,k_2,...],m'])
:: Applies middle convolutions and additions successively to a generalized Riemann scheme or a sum of residue matrices
50. mcop(p,[$r_1,r_2,\ldots,r_n$],[$x,x_1,x_2,\ldots$])
:: Applies middle convolutions and additions successively to a (partial) differential operator p
51. redgrs(m|mat=1)
:: Returns 1-step reduction of generalized Riemann scheme of Fuchsian differential equation
52. getbygrs(m,t|perm=ℓ,var=v,pt=[$p_1,\ldots$],mat=1) or
getbygrs(m,[$t,s_1,s_2,\ldots$]|perm=ℓ,var=v,pt=[$p_1,\ldots$],mat=1)
:: Analyzes Fuchsian ODE defined by generalized Riemann scheme (GRS) (GRS may be given in a short form or a spectral type)
t ="reduction", "construct", "connection", "operator", "series", "TeX", "Fuchs", "basic", "", "All", "irreducible", "recurrence"
s ="TeX", "dviout", "keep", "simplify", "short", "general", "operator", "irreducible", "sft", "top0", "x1", "x2"
ℓ is a permutation or transposition of singular points (cf. mperm()), var are variables of exponents

(cf. sp2grs()) , $p_1, \ldots$ are position of singular points (except for ∞).

53. spslm(m,[k_1,k_2,...])
:: Gets a semilocal monodromy of a rigid linear ordinary differential equation
54. shiftop(ℓ, s|zero=1,raw=k,all=t,dviout=1)
:: Gets the shift operator of rigid ODE with a spectral type ℓ corresponding to a shift$s$
55. shiftPfaff(a,b,g,x,[μ_1,μ_2])
:: Transformation of an adjacent relation of Pfaffian systems under a middle convolution
56. conf1sp(m|x2=± 1,conf=0)
:: Confluence (Poincare rank 1) of a differential operator with spectral type m
57. pf2kz(m|all=1)
:: Converts a Pfaffian system m of n variables to a KZ system of $n+2$ variables
58. confexp([[c_0,f_0],[c_1,f_1],...,[c_m,f_m]]) confexp([[f_1,...,f_n],[c_1,...,c_m]]|sym=k])
:: Confluence of characteristic exponents
59. mcvm(n|var=x,z=1,get=g) mcvm([n_1,n_2,...]|var=[a,b,...],z=1,get=g)
mcvm([r,k,i,j]|e=1,var=[a,b,...])
:: Middle convolution of versal unfolding
60. anal2sp(m,l)
:: Analyzes simultaneous spectral exponents m
61. mc2grs(g,r|top=0,dviout=k,div=ℓ,fig=s)
:: Transformation of simultaneous spectral exponents of KZ equation of 5 points in $\mathbb{P}^1$
62. m2mc(ℓ,[a_0,a_y,a_1,c]|swap=1,small=1,simplify=0,MC=1)
m2mc(ℓ,s|small=1,simplify=0,int=0,swap=t)
:: Addition+middle convolution of Pfaffian system
$du = (A_0\frac{dx}{x} + A_y\frac{d(x-y)}{x-y} + A_1\frac{d(x-1)}{x-1} + B_0\frac{dy}{y} + B_1\frac{d(y-1)}{y-1})u$ with respect to x-variable ($\ell = [A_0, A_y, A_1, B_0, B_1]$).
When ℓ is a spectral type or a Riemann scheme, s ="GRC", "GRSC", "extend", "Pfaff", "sp", "pairs", "irreducible", "All", "swap" in the latter case
63. mcmgrs(g,r|dviout=k)
:: Transform of simultaneous spectral exponents g of a KZ equation of several (≥ 5) points in $\mathbb{P}^1$
64. mmc(l,[μ,a_1,...,a_n]|full=1,homog=1,mult=f)
:: Addition and middle convolution of ODE of Schlesinger type and a KZ equation
65. linfrac01(ℓ|over=1)
:: List of linear fractional transformations of $x = 0, 1, \infty, y, z, \ldots$ ($\ell = [x, y]$ etc.)
66. lft01(ℓ, t|tr=τ)
:: Linear fractional transformations of ℓ = [x,y] or ℓ = [x,y_1,y_2,...,y_q]

1.1.3 Some operators

67. okubo3e([$p_{0,1}, \ldots, p_{0,m}$],[$p_{1,1}, \ldots, p_{1,n}$],[$p_{2,1}, \ldots, p_{2,m+n}$|opt=1])
68. fuchs3e([$p_{0,1}, \ldots, p_{0,n}$],[$p_{1,1}, \ldots, p_{1,n}$],[$p_{2,1}, \ldots, p_{2,n}$])
69. ghg([$p_{1,1}, p_{1,2}, ..., p_{1,m}$],[$p_{2,1}, p_{2,2}, ..., p_{2,n}$])
:: differential operator satisfied by generalized hypergeometric function ${}_mF_n(p_1; p_2; x)$
(cf. seriesHG())
70. even4e([$p_{1,1}, p_{1,2}, p_{1,3}, p_{1,4}$],[$p_{2,1}, p_{2,2}$])
:: Even family of order 4 (Rigid)
71. odd5e([$p_{1,1}, p_{1,2}, p_{1,3}, p_{1,4}, p_{1,5}$],[$p_{2,1}, p_{2,2}$])
:: Odd family of order 5 (Rigid)
72. rigid211([$p_{0,1}, p_{0,2}$],[$p_{1,1}, p_{1,2}$],[q_0, q_1])
:: Type 211,211,211
73. extra6e([$p_{1,1}, p_{1,2}, p_{1,3}, p_{1,4}, p_{1,5}, p_{1,6}$],[$p_{2,1}, p_{2,2}$])

:: Extra case (Rigid)

74. `eofamily([`$p_{0,1}, p_{0,2}$`],[`$p_{1,1}$`],[`$p_{2,1}, \ldots, p_{2,n}$`])`
:: Even/odd family (obsolete)
75. `ev4s(`p_1`,`p_2`,`p_3`,`p_4`,`p_5`)`
:: Rigid restriction of Heckman-Opdam's hypergeometric equation of type (BC_2, BC_1)
76. `b2e(`p_1`,`p_2`,`p_3`,`p_4`,`p_5`)`
:: Non-rigid restriction of Heckman-Opdam's hypergeometric equation of type (BC_2, A_1)
77. `heun([`a, b, c, d, e`],`p`,`r`|)`
:: Heun's equation. r is an accessory parameter

1.2 Useful functions

The following functions are in a module and we call the functions by putting `os_md.` at the head such as `os_md.myhelp()`.

1.2.1 Extended function

78. `myhelp(`h`)`
:: Displays the manual of `os_muldif.rr`
79. `chkfun(`f`,`s`)`
:: Checks whether f (= a string) is defined and `load(`s`)` if it is not
80. `isMs()`
:: Checks whether the operating system is Microsoft Windows
81. `isyes(`p`|set=`ℓ`)`
:: Defines a function returning 0 or 1
82. `isall(`f`,`m`)`
:: Returns 0 if m contains an element satisfying $f(p) = 0$ and 1 otherwise
83. `ptype(`p`,`ℓ`)`
:: Returns `type()` regarding ℓ as the variable or ℓ as the list of variables
84. `getline(`Id`|Max=`m`,CR=[`r_1`,`r_2`,..],LF=[`l_1`,`l_2`,...])`
:: An extension of `get_line()`
85. `keyin(`s`)`
:: Shows s and waits a line input from keyboard and returns it
86. `showbyshell(`s`)`
:: Executes s by shell and show the standard output in `Risa/Asir`
87. `getbyshell(`s`)`
:: Executes s by shell and puts the standard output to a file
88. `fcat(`f`,`s`|exe=1)`
:: Outputs s in a file f
89. `makev([`ℓ_1`,`$\ell_2, \ldots$`])|num=1)`
:: Makes a variable combining $\ell_1, \ell_2 \ldots$
90. `shortv(`p`,[`v_1`,`$v_2, \ldots$`])|top=`w`)`
:: Changes the variables $v_1, \ldots$ with indices contained in p into valiables with one letter
91. `makenewv(`ℓ`|var=`v`,num=`n`)`
:: Generates a new variable which is not used in ℓ
92. `isvar(`p`)`
:: Is p a variable?
93. `varargs(`p`|all=`t`)`
:: Returns elementary functions and variables in p
94. `pfargs(`p`,`x`|level=`t`)`
:: Returns all functions and variables containing variable x

95. `isdif`(p)
:: Returns a list of the pairs of variable and its derivatives
96. `mysubst`(r,[v_1,r_1]|`inv=1`) `mysubst`(r,[[v_1,r_1],...]|`inv=1`)
`mysubst`(r,[ℓ_1,ℓ_2]|`lpair=1,inv=1`)
:: Same as `subst`(r,v_1,r_1,...). Useful if r is complicated and r is a rational form
97. `myswap`(p,[x_1,x_2,...,x_n])
:: Cyclic permutation $(x_1, x_2, \ldots, x_n)$ of a (list of) rational form(s)
98. `mulsubst`(r,[[$p_{1,0}, p_{1,1}$],[$p_{2,0}, p_{2,1}$],...]|`inv=1,lpair=1`)
:: Simultaneous substitutions
99. `fmult`(f,m,ℓ,n|$\cdots$)
:: Returns $m_{length(l)}$ by $m_i \mapsto m_{i+1} = f$(`f`$(m_i$,$\ell[i]$,$n[0], n[1], \ldots$|$\cdots$) $(m_0 = m)$
100. `mtransbys`(f,m,ℓ|$\cdots$)
:: Extends a function $f(x)$ of scalar x to a function of a list, vector or matrix
101. `mmulbys`(f,m,n,ℓ|$\cdots$)
:: Extends a pair of arguments of f to a a pair of matrices
102. `cmpsimple`(p,q|`comp=`t)
:: Copmares the simplicities of p and q
103. `simplify`(p,ℓ,t|`var=`[$x_1, x_2, \ldots$])
:: Simplifies p by using linear relations indicated by ℓ
104. `getel`(m,i)
:: Returns $m[i]$ if m is a list/vector/matrix and i is non-negative integer
105. `evalred`(r|`opt=`[[s_1,t_1],[s_2,t_2]...])
:: Replaces `sin(0)`, `cos(0)`, `exp(0)` by 0, 1, 1 etc

1.2.2 Numbers
1.2.3 Polynomials and rational functions

191. `fctrtos`(r|`var=`ℓ,`rev=1,dic=1,TeX=`f,`dviout=1,lim=`n,`small=1,pages=1,add=`s)
:: Transforms a rational function r to factorized strings

1.2.4 Functions with real/complex variables
1.2.5 Lists and vectors
1.2.6 Matrices
1.2.7 Strings
1.2.8 Permutations
1.2.9 TEX

351. `show`(p|`opt=`ℓ,`raw=1`)
:: Displays p by using TEX in a plausible format
352. `dviout`(p|`clear=1,keep=1,delete=`t,`fctr=1,mult=1,subst=`[$s0, s1$],`eq=`k,`title=`s)
:: Displays p
353. `dviout0`(ℓ) or `dviout0`([$\ell_1, \ell_2, \ldots$]) or `dviout0`(ℓ|`opt=`s)
:: Fundamental operations to display equations using TEX
363. `ltotex`(l|`opt=`s,`pre="string",cr="cr",small=1,lim=`ℓ,`var=`v)
:: Transforms a list to plausible TEX source which represents GRS or a table or a list etc.

1.2.10 Lines and curves
1.2.11 Drawing curves and graphs
1.2.12 Applications
1.2.13 Environments

430. `Canvas`
:: Default size of the canvas of `Risa/Asir`

431. AMSTeX
:: TEX means $\mathcal{AMS}$LATEX if this value equals 1
432. TeXEq
:: Default display style of LATEX (dviout0(3) shows it)
433. TeXLim
:: Maximal width referred to divide a long equations into lines by LATEX
434. TeXPages
: Threshold value of the numbers of lines for a page in a display style of TEX (cf. fctrtos())
435. TikZ
:: Flag using XY-pic or Ti*k*Z in TEX
436. XYPrec
:: Number of figures after the decimal point in the expression of coordinates for graphic display
437. XYcm
:: Unit is cm even in XY-pic
438. XYLim
:: Maximal number of coordinates for a line in a source of XY-pic or Ti*k*Z
439. DVIOUTH
:: A program to show explanations of functions indicated by myhelp()
440. DIROUT
:: Directory where the source file of LATEX is put (it should be writable)
441. DVIOUTA
:: Pathname of a program which compiles a source risaout.tex of LATEX under $\mathcal{AMS}$TEX and displays it
442. DVIOUTB
:: Pathname of a program which compiles a source risaout10.tex (or risaout10.tex) of LATEX under $\mathcal{AMS}$TEX and displays it. We can choose this setting or DVIOUTA in Risa/Asir
443. DVIOUTL
Pathname of a program which compiles a source risaout0.tex of LATEX and displays it
444. .muldif
:: os_muldif.rr reads this file when it starts up
445. risatex.bat
:: Program compiling the LATEX file which os_muldif.rr outputs and displaying it on a screen

2 Risa/Asir

`Risa/Asir` is an open source general computer algebra system. Kobe distribution is being developed by OpenXM committers. The original Risa/Asir is developed at Fujitsu Labs LTD.

The Kobe distribution works on Windows (32bit, 64bit), UNIX and MAC OS X and can be obtained from

http://www.math.kobe-u.ac.jp/Asir/asir.html

2.1 Risa/Asir and os_muldif.rr

We should remark that the calculations of rational functions by `Risa/Asir` may give different resuls from what we have expected. In particular. they happen for calculations of polynomials or differential operators with coefficient rational functions and matrices whose entries are rational functions. Here we give some examples:

```
[0] 2/x-1/x+1/x-1/x;
(x^3)/(x^4)
[1] x/(x+y)+y/(x+y);
(x^2+2*y*x+y^2)/(x^2+2*y*x+y^2)
[2] x/y*y/x;
(y*x)/(y*x)
[3] 1/(1/x);
(x)/(1)
[4] deg((a/b)*x^2,x);
0
[5] diff((1/a)*x+1/b,x);
(b^2*a)/(b^2*a^2)
[6] diff((x+1)^(-3),x);
(-3*x^2-6*x-3)/(x^6+6*x^5+15*x^4+20*x^3+15*x^2+6*x+1)
[7] A = newmat(2,2,[[a,0],[0,1/a]]);
[ a 0 ]
[ 0 (1)/(a) ]
[8] det(A);
internal error (SEGV)
return to toplevel
[9] coef(x+1/a,1,x);
0
```

The library `os_muldif.rr` [O6] was first developped to avoid these inconvenient calculations and it is expected to give more reasonable results.

In this note some explanation of current functions in `os_muldif.rr` is given but it may be tentative and may be changed in future.

- If the command `which("os_muldif.rr")` shows the path to `os_muldif.rr`, `os_muldif.rr` can be loaded.
  If the path is not shown, put `os_muldif.rr` in the load path of `Risa/Asir` (cf. `ctrl("loadpath")`.

It is usually in `./lib/asir-contrib` where `names.rr` etc.exist. It is under `./lib/` where there are libraries of `Risa/Asir`. Note that the command `which("names.rr")` after stating up `Risa/Asir` shows the place. Then `os_muldif.rr` is loaded by the command `load("os_muldif.rr")$` and we can use the functions in `os_muldif.rr`.

- `Risa/Asir` loads `os_muldif.rr` as a module, we should put `os_md.` at the head of the name of a function to call a function.
- If `load("names.rr")` is performed (a package of `Risa/Asir` may perform this command at the start up), then `os_muldif.rr` can output results by TEX source which enables us to easily read and understand the results.
  If there is a path to `dviout` (a previewer of TEX) in Windows, the results can be shown by `dviout`. By a certain setting it is also possible under other environment according to its TEX system and the way to handle it. Namely a DIV file or a PDF file is created by the TEX system and it can be displayed on a screen.
- `chkfun(1,0)` shows Version of `Risa/Asir`.

Some of the characteristic functions of `os_muldif,rr` are `dviout()`, `show()`, `ltotex()`, `fctrtos()`, `mtotex()`, `myhelp()`, `xy2graph()`, `mtoupper()`, `mdivisor()`, `getbygrs()`, `shiftop()`, `m2mc()`, `mc2grs()`, `scale()` etc.

2.1.1 Install os_muldif.rr

- If `os_muldif.rr` exists in the load path of `Risa/Asir` which is shown by the command `ctrl("loadpath")`, the library `os_muldif.rr` is loaded by the command

```
[0] load("os_muldif.rr")$
Loaded muldif Ver. 00140330 (Toshio Oshima)
```

  and the functions defined by `os_muldif.rr` can be used.
- The function in `os_muldif.rr` is called by putting `os_md.` at the head of the name of the function. In the following `os_md.` is omitted except for examples.
- If a line

```
import("os_muldif.rr")$
```

  is added in the file `.asirrc` which exists in the directory indicated by `get_rootdir()`, then `os_muldif.rr` is automatically loaded when `Risa/Asir` starts up. An example of `.asirrc` is

```
import("contrib-setord.rr")$
import("gr")$
import("primdec")$
import("katsura")$
import("bfct")$
import("names.rr")$
import("oxrfc103.rr")$
import("os_muldif.rr")$
end$
```

- We can omit the head `os_md.` for functions which are frequently used by defining new corresponding functions. For example we prepare a file `mydef.rr` written as

```
def cat(X)
{return os_md.mycat(X|option_list=getopt());}
```

```
def myhelp(X)
{return os_md.myhelp(X|option_list=getopt());}
def show(X)
{return os_md.show(X|option_list=getopt());}
. . .
cat("mydef:\n cat, myhelp, show,...")$
```

and put it in the loadpath of `Risa/Asir` and write a line

```
import("mydef.rr")$
```

in `.asirrc`.

- A certain setting is necessary so that `os_muldif.rr` directly shows its results on the display screen using TEX if TEX is installed. The setting depends on the operating system and a command to compile a TEX source file and show a dvi file or PDF fle on a display screen. See `risatex.bat` and related subjects.
- The function `myhelp("`*fn*`")` in `os_muldif.rr` can show the explanation of a given function *fn* in a certain environment. See `DVIOUTH` and related explanation. Put `os_muldif.dvi` and `os_muldif.pdf` in `get_rootdir()/help`. Then if `dviout` works under Microsoft Windows, the setting of `DVIOUTH` may not be necessary.

3 Functions

3.1 Functions related to differential operators

The following functions are in a module and we call the functions by putting `os_md.` at the head such as `os_md.muldo()`.

3.1.1 Fundamental functions

1. `muldo(`p_1`,`p_2`,[`x`,`∂_x`]|lim=`n`)` or `muldo(`p_1`,`p_2`,`x`|lim=`n`)`
   `muldo(`p_1`,`p_2`,[[`x_1`,`∂_{x_1}`],[`x_2`,`∂_{x_2}`],...]|lim=`n`)` or `muldo(`p_1`,`p_2`,[[`x_1`],`x_2`,...]|lim=`n`)`
   :: Returns the product of ordinary (partial) differential operators whose coefficients are rational functions (or elementary functions) ($\Leftarrow [\partial_x, x] = 1$)
   - Rational functions are quotients of polynomials with coefficients in rational numbers (or complex numbers whose real and imaginary parts are rational numbers (cf. Type of numbers).
   - If ∂_x is standard `d`x, `[`x`,`∂_x`]` may be x in short.
     This is same for other following functions.
   - If p is a partial differential operators, the third parameter can be `[[x],[y,v],z]` for example. This is understood `[[x,dx],[y,v],[z,dz]]` adding `d` but the first element of the list should ne a list. Namely `[[x,dx],[y,dy]]` can be `[[x],y]` but `[x,y]` is understood p as an ordinary differential operator.
     Other following function the similar short expressions are allowed.
   - p_2 can be a vector of a matrix and in this case p_1 may be a matrix.
   - If x and x_i are 0, then the product is a usual product.
   - Differential operators of infinite order such as $\exp(\partial)$ or ∂^{-1} are allowed. (But they are not allowed for `appldo()` etc. If the calculation is not finite, then the calculation will be ignored after a finite terms).
   - Using the Leibniz formula, the order of differentiations are calculated up to 100 by default bue the number 100 is changed by the option parameter `lim=`n, which is only possible when p_1 and p_2 are scalars.
     For example, `muldo(dx^(-1),1/x,x)` and `muldo(exp(dx),1/x,x)` have infinite terms.

   Examples with $[\mathtt{dx}, \mathtt{x}] = 1$, $[\mathtt{dy}, \mathtt{y}] = 1$ are as follows ($\Leftarrow$ $\mathtt{d}x = \frac{\partial}{\partial x}$, $\mathtt{d}y = \frac{\partial}{\partial y}$)

```
[0] os_md.muldo((1+x)*dx+1, (1-x)*dx+1, [x,dx]);
(-x^2+1)*dx^2+(-x+1)*dx+1
[1] os_md.muldo(x*dx+1/(x-a),a*dx+1/x,x);
((a*x^3-a^2*x^2)*dx^2+x^2*dx-x+a+1)/(x^2-a*x)
[2] os_md.muldo((a+y)*dy, (b-y)*dy, y);
(-y^2+(-a+b)*y+b*a)*dy^2+(-y-a)*dy
[3] os_md.muldo((a+y)*dy, (b-y)*dy, [0,dy]);
(-y^2+(-a+b)*y+b*a)*dy^2
[4] os_md.muldo(dx+dy, x*dx+y*dy, [[x],y]);
x*dx^2+((x+y)*dy+1)*dx+y*dy^2+dy
[5] os_md.muldo(dx+dy,sin(x+y),[[x],y]);
sin(x+y)*dy+sin(x+y)*dx+2*cos(x+y)
[6] os_md.muldo(exp(dx),(x-1)^4,x);
exp(dx)*x^4
[7] subst(@@,exp(dx),1);
```

```
x^4
[8] os_md.muldo(x*dx^(-1),dx/x,x);
Over 100 derivations!
(x^100*dx^{100}+x^99*dx^99+2*x^98*dx^98+.........)/(x^100*dx^100)
[9] os_md.muldo(dx^(-1),dx/x,x|lim=5);
Over 5 derivations!
(x^5*dx^5+x^4*dx^4+2*x^3*dx^3+6*x^2*dx^2+24*x*dx+120)/(x^5*dx^5)
[10] os_md.muldo(x*exp(dx),1/x,x|lim=5);
Over 5 derivations!
(exp(dx)*x^5-exp(dx)*x^4+exp(dx)*x^3-exp(dx)*x^2+exp(dx)*x-exp(dx))/(x^5)
[11] deval(os_md.muldo(exp(dx),exp(x),x]);
Over 100 derivations!
2.71828*exp(1*dx)*exp(1*x)
```

`[0]` means

$$\big((1+x)\tfrac{d}{dx}+1\big)\circ\big((1-x)\tfrac{d}{dx}+1\big)=(1-x^2)\tfrac{d^2}{dx^2}+(1-x)\tfrac{d}{dx}+1$$

In `[6]`, `[8]`, `[9]`, `[10]`, `[11]` the results are understood by arranging them in sums of products or quotients of functions and differentials as in the usual expression. Namely
`[6]` means

$$e^{\frac{d}{dx}}\circ\tfrac{1}{(x-1)^4}=x^4e^{\frac{d}{dx}}=x^4(1+\tfrac{d}{dx}+\tfrac{1}{2}\tfrac{d^2}{dx^2}+\tfrac{1}{3!}\tfrac{d^3}{dx^3}+\cdots)$$

and `[8]` means

$$x\big(\tfrac{d}{dx}\big)^{-1}\circ\tfrac{1}{x}\tfrac{d}{dx}=1+\tfrac{1}{x}(\tfrac{d}{dx})^{-1}+\tfrac{2}{x^2}(\tfrac{d}{dx})^{-2}+\cdots$$

and `[10]` means

$$xe^{\frac{d}{dx}}\circ\tfrac{1}{x}=(1-\tfrac{1}{x}+\tfrac{1}{x^2}-\tfrac{1}{x^3}+\tfrac{1}{x^4}-\tfrac{1}{x^5})(1+\tfrac{d}{dx}+\tfrac{1}{2!}\tfrac{d^2}{dx^2}+\tfrac{1}{3!}\tfrac{d^3}{dx^3}+\cdots)$$

`[11]` corresponds to

$$e^{\frac{d}{dx}}\circ e^x=e^{x+1}e^{\frac{d}{dx}}$$

2. `caldo([`p_1`,`p_2`,...],[`x`,`∂_x`])`
   :: Returns the sum of products of differential operators
   Returns $p_1+p_2+\cdots$ if p_j are (partial) differential operators.
   If p_j are a list `[`$p_{j,1},\ldots$`]` of differential operators, then
   - $p_j=p_{j,1}p_{j,2}\cdots$ are products of differential operators.
   - Moreover if $p_{j,\nu}$ is `[`$q_{j,\nu},m_{j,\nu}$`]`, $p_{j,\nu}=q_{j,\nu}^{m_{j,\nu}}$. Here $m_{j,\nu}$ should be non-negative integer.

   ```
   [0] os_md.caldo([[dx-x,[dx+x,2]],y],x);
   dx^3+x*dx^2+(-x^2+3)*dx-x^3+x+y
   ```

3. `muledo(`p_1`,`p_2`,[`x`,`∂_x`])` or `muledo(`p_1`,`p_2`,`x`)`
   :: Returns the product of differential operators of Euler type ($\Leftarrow[\partial_x,x]=x$)
   p_2 can be a matrix of vector and in this case p_1 can be a matrix.
   `[dx,x]=x` $\Leftarrow$ `dx`$=x\frac{d}{dx}$.

   ```
   [0] os_md.muledo((1+x)*dx+1, (1-x)*dx+1, x);
   (-x^2+1)*dx^2+(-x+1)*dx+1
   ```

4. `transpdo(`p`,[[`x_1`,`∂_{x_1}`],[`x_2`,`∂_{x_2}`],...],[[`y_1`,`∂_{y_1}`],[`y_2`,`∂_{y_2}`],...]|ex=1,inv=`f`)`
   :: Transformation of differential operator p by $x_i\mapsto y_i=y_i(x)$, $\partial_{x_j}\mapsto\partial_{y_j}=c_j(x)+\sum_\nu a_{j\nu}(x)\partial_{x_\nu}$

- y_i, ∂_{y_j} are rational functions and differential operators
- $[x_i, \partial_{x_i}]$ may be written by x_i ($\Leftarrow\ \partial_{x_i} = \mathtt{d}x_i$).
- If length of the 3-rd list is shorter than the length of the 2-nd list, then the identity maps are added to the last of the 2-nd list.
- p can be a list or vector or matrix of differential operators.
- If the transformation corresponds to a birational transformation of coordinates, then the second list can be the list `[`y_1`,`y_2`,...]` with the same length as the first list. $\partial_{y_1}, \partial_{y_2}, \ldots$ are calculated.
  - `inv=1` : Returns an inverse transformation
  - `inv=2` : Returns `[`∂_{y_1}`,`∂_{y_2}`,...]`.
  - `inv=3` : Returns `[`∂_{y_1}`,`∂_{y_2}`,...]` corresponding to the inverse transformation.
- If the 3-rd parameter of this function is an invertible matrix $m = (m_{i,j})$ of integers or the corresponding list of lists, then the transformation is defined by the coordinate transformation `transppow()`:

$$x_i \mapsto \prod_{j=1}^{m} x_j^{m_{i,j}}.$$

```
[0] os_md.transpdo(x^2*dx^2,x,[[1/x,-x^2*dx]]);
x^2*dx^2+2*x*dx
[1] os_md.transpdo(x*dx+y*dy,[[x],y],[x+y,x-y]);
os_md.transpdo(x*dx+y*dy,[[x],y],[[x+y,(dx+dy)/2],[x-y,(dx-dy)/2]]);
y*dy+x*dx
[2] os_md.transpdo(4*dx^2,[[x,dx],[y,dy]],[x+y,x-y]);
dy^2+2*dx*dy+dx^2
[3] os_md.transpdo(x*dx+y*dy,[[x],y],[x+y,x-y]|inv=2);
[1/2*dy+1/2*dx,-1/2*dy+1/2*dx]
[4] os_md.transpdo(x*dx,x,[1/x]);
-x*dx
[5] os_md.transpdo(x*dx,x,[-dx+1/x,x]|ex=1);
-x*dx
```

- In `[0]`, `[1]` abbreviated form of `[[`x_1`,`∂_{x_1}`],[`x_2`,`∂_{x_2}`],...]` is used (cf. `muldo()`).
- In `[0]` and `[1]` the coordinate transformation $x \mapsto \frac{1}{x}$ and $(x, y) \mapsto (x+y, x-y)$ are considered. In these cases the transformations $\frac{d}{dx} \mapsto -\frac{1}{x^2}\frac{d}{dx}$ and $(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}) \mapsto (\frac{1}{2}\frac{\partial}{\partial x} + \frac{1}{2}\frac{\partial}{\partial y}, \frac{1}{2}\frac{\partial}{\partial x} - \frac{1}{2}\frac{\partial}{\partial y})$ are indicated but we may shortly indicate them as in `[2]`.
- We may indicated the transformation which induces an isomorphism of the ring of differential operators such as $(x, \frac{d}{dx}) \mapsto (x, \frac{d}{dx} + c)$.
- `ex=1` : Returns a transformation of p defined by $(x_i, \partial_{x_i}) \mapsto (S_i(x, \partial), T_i(x, \partial))$ which induces an isomorphism of the ring of differential operators. Here $[T_i(x, \partial), S_j(x, \partial)] = \delta_{i,j}$ and p is a polynomial of (x, ∂).
  `[5]` considers a transformation defined by $(x, \frac{d}{dx}) \mapsto (-\frac{d}{dx} + \frac{1}{x}, x)$.

5. `translpdo(`p`,[[`x_1`,`∂_{x_1}`],[`x_2`,`∂_{x_2}`],...],`mat`)`
   :: Transforms a differential operator p by the coordinate transformation $x_i \mapsto \sum_j (mat)_{ij} x_j$
   `[`x_i`,`∂_{x_i}`]` may be an abbreviated form x_i ($\Leftarrow\ \partial_{x_i} = \mathtt{d}x_i$).

```
[0] M=mat([1,1],[1,-1]);
[ 1 1 ]
[ 1 -1 ]
[1] os_md.translpdo(4*dx^2,[[x,dx],[y,dy]],M);
```

```
dy^2+2*dx*dy+dx^2
```

6. `transppow([[`x_1`,`∂_{x_1}`],...,[`x_n`,`∂_{x_n}`]],`m`)`
   :: Coordinate transformations by products of powers of coordinates
   Returns the transformation of `[[`x_1`,`∂_{x_1}`],...,[`x_n`,`∂_{x_n}`]]` byr the map

$$x_i \mapsto \prod_{j=1}^{n} x_j^{m_{i,j}} \quad (j = 1, \ldots, n)$$

```
[0] os_md.transppow([x,y],[[1,1],[0,-1]]);
[[x,(y*dy+x*dx)/(x)],[(x)/(y),(-y^2*dy)/(x)]]
[1] os_md.transppow([x,y],[[1,1],[0,1]]);
[[x,(-y*dy+x*dx)/(x)],[y*x,(dy)/(x)]]
```

7. `appldo(`p`,`r`,[`x`,`∂_x`]|Pfaff=1)` or `appldo(`p`,`r`,[[`x_1`,`∂_{x_1}`],[`x_2`,`∂_{x_2}`],...])`
   :: Applies a (matrix of) differential operator to a (matrix of) rational function or elementary function or a Pfaffian system
   r can be a vector or a matrix. In this case p can be a matrix.
   - `Pfaff=1` : For $\frac{du}{dx} = r(x)u$, $s(x)$ satisfying $p(x, \frac{d}{dx})u = s(x)u$ is returned (p, r is a matrix or a scalar).

```
[0] os_md.appldo(x*dx^2-2*dx, a*x^3+b*x^2, x);
-2*b*x
[1] os_md.appldo(x*dx^2+2*dx, x+y/x, x);
2
[2] V=newvect(2,[x/y,y/x]);
[ (x)/(y) (y)/(x) ]
[3] os_md.appldo(x*dx+1,V,x);
[ (2*x)/(y) 0 ]
[4] P=newmat(2,2,[[x*dx+1,x/y],[0,x*dx]]);
[ x*dx+1 (x)/(y) ]
[ 0 x*dx ]
[5] os_md.appldo(P,V,x);
[ (2*x+y)/(y) (-y)/(x) ]
[6] os_md.appldo(ddx,P,dx);
[ x 0 ]
[ 0 x ]
[7] A=newmat(2,2,[[cos(x+y),-sin(x+y)],[sin(x+y),cos(x+y)]]);
[ cos(x+y) -sin(x+y) ]
[ sin(x+y) cos(x+y) ]
[8] os_md.appldo(dx^2,A,x);
[ -cos(x+y) sin(x+y) ]
[ -sin(x+y) -cos(x+y) ]
[9] V=[lambda1,lambda2];C0=newmat(2,2,[V,V]);
[10] C=os_md.diagm(2,[1/x,1/(x-1)])*(C0+os_md.diagm(2,[mu]));
[11] P=os_md.diagm(2,[1,dx])*C0;
[12] U=os_md.appldo(P,C,[x,dx]|Pfaff=1);
[13] V=os_md.myinv(U);
```

In the above we have the matrices U and V of rational functions which satisfies $u = Uv$ and $v = Vu$ for $u_0 = \partial^{-\mu} x^{\lambda_1}(1-x)^{\lambda_2}$ (Gauss hypergeometric function), $v = \partial^{-\mu-1}\begin{pmatrix} x^{\lambda_1-1}(1-x)^{\lambda_2} \\ x^{\lambda_1}(1-x)^{\lambda_2-1}\end{pmatrix}$.

8. `adj(`p`,[`x`,`∂_x`])` or `adj(`p`,[[`x_1`,`∂_{x_1}`],[`x_2`,`∂_{x_2}`],...])`
:: Returns the formal adjoint of a (matrix of) differential operator p

```
[0] os_md.adj(x*dx^2+x^3*dx+1,x);
x*dx^2+(-x^3+2)*dx-3*x^2+1
```

9. `sftpexp(`p`,[`x`,`∂_x`],`q`,`r`)` or `sftpexp(`p`,[[`x_1`,`∂_{x_1}`],...],`q`,`r`])`
:: Returns $q^{-r} \circ p \circ q^r$
Here q and r are rational function and r is a parameter.

```
[0] os_md.sftpexp(dx*dy,[[x,dx],[y,dy]],exp(x-y),a);
(dx+a)*dy-a*dx-a^2
[1] os_md.sftpexp(dx*dy,[[x,dx],[y,dy]],x-y,a);
((x^2-2*y*x+y^2)*dx+a*x-a*y)*dy+(-a*x+a*y)*dx-a^2+a
[2] os_md.show(@@);
```

$$(x-y)^2\partial_x\partial_y - a(x-y)\partial_x + a(x-y)\partial_y - a(a-1)$$

10. `appledo(`p`,`r`,[`x`,`∂_x`])`
:: Applies a differential operator of Euler type to a rational function

```
[0] os_md.appledo(dx^2,x^2+y/x,x);
(4*x^3+y)/(x)
```

11. `divdo(`p_1`,`p_2`,[`x`,`∂_x`]|rev=1)`
:: Calculates divisions of ordinary differential operators
- Returns $[q, r, m]$
  $\Rightarrow m * p_1 = q * p_2 + r \quad (\operatorname{ord} m = 0,\ \operatorname{ord} r < \operatorname{ord} p_2)$
- If `rev=1` is indicated
  $p_1 * m = p_2 * q + r \quad (\operatorname{ord} m = 0,\ \operatorname{ord} r < \operatorname{ord} p_2)$

```
[0] R=os_md.divdo(dx^2, x*dx+1, x);
[x*dx-2,2,x^2]
[1] os_md.muldo(R[0],x*dx+1,x)+R[1];
x^2*dx^2
[2] R=os_md.divdo(dx^2,x*dx+1,x|rev=1);
[x*dx+2,0,x^2]
[3] os_md.muldo(x*dx+1,R[0],x)+R[1];
x^2*dx^2+4*x*dx+2
[4] os_md.muldo(dx^2,R[2],x);
x^2*dx^2+4*x*dx+2
```

12. `mygcd(`p_1`,`p_2`,[`x, ∂_x`]|rev=1,dviout=`n`)` or `mygcd(`p_1`,`p_2`,[`x`]|rev=1,dviout=`n`)`
`mygcd(`p_1`,`p_2`,`x`|dviout=`n`)`, `mygcd(`p_1`,`p_2`,0|dviout=`n`)`
:: GCD of ordinary differential operators (or polynomials of x or positive integers) p_1 and p_2
- Let R be the result of the function. Then GCD equals $R[0] = R[1] * p_1 + R[2] * p_2$.
  $R[3] * p_1 + R[4] * p_2 = 0$ is valid and the matrix $\begin{pmatrix} R[1] & R[2] \\ R[3] & R[4]\end{pmatrix}$ is invertible.
  Moreover $R[3] * p_1 = -R[4] * p_2$ is LCM of p_1 and p_2,

- `rev=1` : The order of the products are inverted in the above.
- The calculation uses euclidean algorithm algorithm. If the size of p_1 and p_2 are same, p_1 is first divided by p_2.
- `dviout=0` : All steps in the euclidean algorithm returned as a list of lists.
  In the case of integers, each list consists of pair of integers and the pair of quotient and reminder. The first list contains `[`p_1`,`p_2`]`.
  In the case of polynomials or differential operators, each list consists of a pair of polynomials or differential operators, a pairs of quotient and reminder and a scalar to be multiplied (the list returned by `divdo()`).
- `dviout=1` : the euclidean algorithm is displayed using TEX.
- `dviout=2` : the euclidean algorithm is displayed using matrices and TEX.
- `dviout=-1,-2` : returns the TEX source of the above.

```
[0] P = os_md.muldo(x*dx+1,x*dx+1,x);
x^2*dx^2+3*x*dx+1
[1] Q = os_md.muldo(dx-1,x*dx+1,x);
x*dx^2+(-x+2)*dx-1
[2] os_md.mygcd(P,Q,[x]);
[x*dx+1,(1)/(x+1),(-x)/(x+1),((-x-1)*dx+x+2)/(x+1),((x^2+x)*dx+x+2)/(x+1)]
[3] os_md.mygcd(P,Q,[dx]);
[1,0,(1)/(x*dx^2+(-x+2)*dx-1),1,(-x^2*dx^2-3*x*dx-1)/(x*dx^2+(-x+2)*dx-1)]
[4] os_md.mygcd(234,111,0);
[3,-9,19,37,-78]
[5] os_md.mygcd(234,111,0|dviout=0);
[[234,111],[2,12],[9,3],[4,0]]
[6] os_md.mygcd(234,111,0|dviout=1);
```

$$
\begin{aligned}
234 &= 2\times 111+12\\
111 &= 9\times 12+3\\
12 &= 4\times 3
\end{aligned}
$$

```
[7] os_md.mygcd(234,111,0|dviout=-1);
234&=2\times111+12\allowdisplaybreaks\\
111&=9\times12+3\allowdisplaybreaks\\
12&=4\times3
[8] os_md.mygcd(234,111,0|dviout=2)$
```

$$
\begin{aligned}
\begin{pmatrix}234\\111\end{pmatrix} &= \begin{pmatrix}2&1\\1&0\end{pmatrix}\begin{pmatrix}111\\12\end{pmatrix}\\
&= \begin{pmatrix}2&1\\1&0\end{pmatrix}\begin{pmatrix}9&1\\1&0\end{pmatrix}\begin{pmatrix}12\\3\end{pmatrix} = \begin{pmatrix}19&2\\9&1\end{pmatrix}\begin{pmatrix}12\\3\end{pmatrix}\\
&= \begin{pmatrix}19&2\\9&1\end{pmatrix}\begin{pmatrix}4&1\\1&0\end{pmatrix}\begin{pmatrix}3\\0\end{pmatrix} = \begin{pmatrix}78&19\\37&9\end{pmatrix}\begin{pmatrix}3\\0\end{pmatrix},\\
\begin{pmatrix}3\\0\end{pmatrix} &= \begin{pmatrix}-9&19\\37&-78\end{pmatrix}\begin{pmatrix}234\\111\end{pmatrix}
\end{aligned}
$$

```
[9] os_md.mygcd(P,Q,[x,dx]|dviout=2)$
```

$$x^2\partial^2+3x\partial+1=(x)(x\partial^2-(x-2)\partial-1)+(x(x+1)\partial+(x+1))$$
$$(x^2+2x+1)(x\partial^2-(x-2)\partial-1)=((x+1)\partial-(x+2))(x(x+1)\partial+(x+1))$$

```
[10] os_md.mygcd(P,Q,[x,dx]|dviout=2);
```

$$\begin{aligned}\begin{pmatrix}x^2\partial^2+3x\partial+1\\ x\partial^2-(x-2)\partial-1\end{pmatrix}&=\begin{pmatrix}x&1\\1&0\end{pmatrix}\begin{pmatrix}x\partial^2-(x-2)\partial-1\\ x(x+1)\partial+(x+1)\end{pmatrix}\\&=\begin{pmatrix}x&1\\1&0\end{pmatrix}\begin{pmatrix}\frac{1}{x+1}\partial-\frac{x+2}{(x+1)^2}&1\\1&0\end{pmatrix}\begin{pmatrix}x(x+1)\partial+(x+1)\\0\end{pmatrix}\\&=\begin{pmatrix}\frac{x}{x+1}\partial+\frac{1}{(x+1)^2}&x\\\frac{1}{x+1}\partial-\frac{x+2}{(x+1)^2}&1\end{pmatrix}\begin{pmatrix}x(x+1)\partial+(x+1)\\0\end{pmatrix}\\&=\begin{pmatrix}x\partial+1&x\\\partial-1&1\end{pmatrix}\begin{pmatrix}x\partial+1\\0\end{pmatrix},\\\begin{pmatrix}x\partial+1\\0\end{pmatrix}&=\begin{pmatrix}\frac{1}{(x+1)^2}&-\frac{x}{(x+1)^2}\\-\frac{1}{x+1}\partial+\frac{x+2}{(x+1)^2}&\frac{x}{x+1}\partial+\frac{x+2}{(x+1)^2}\end{pmatrix}\begin{pmatrix}x^2\partial^2+3x\partial+1\\x\partial^2-(x-2)\partial-1\end{pmatrix}\end{aligned}$$

13. `mylcm(`p_1`,`p_2`,[`x,∂_x`]|rev=1)` or `mylcm(`p_1`,`p_2`,[`x`]|rev=1)`
`mylcm(`p_1`,`p_2`,`x`)`, `mylcm(`p_1`,`p_2`,0)`
:: LCM of ordinary differential operators (or polynomials of x or positive integers) p_1 and p_2

```
[0] P=os_md.mylcm((2-x)*dx-1,x*dx+1,[x,dx]);
(x^2-2*x)*dx^2+(4*x-4)*dx+2
[1] os_md.appldo(P,a/x+b/(2-x),[x,dx]);
0
[2] Q=os_md.mylcm((2-x)*dx-1,x*dx+1,x);
(x^2-2*x)*dx^2+(2*x-2)*dx+1
[3] fctr(Q);
[[1,1],[x*dx+1,1],[(x-2)*dx+1,1]]
```

14. `m1div(`m`,`n`,[`x,∂_x`])` or `m1div(`m`,`n`,[`x`])` or `m1div(`m`,`n`,`x`)`
:: Returns $R=[R[0],R[1]]$ such that $m=R[1](\partial_x-n)+R[0]$ (or $m=R[1](x-n)+R[0]$) for a square matrix m of differential operators with coefficients in rational functions (or polynomials of x).
Here $R[0]$ does not contain differentiation (ro x).
15. `qdo(`p_1`,`p_2`,[`x,∂_x`])`
:: Returns the list $[q_1,q_2]$ of differential operators satisfying $q_1p_2u=0$ and $q_2p_2u=u$ for a differential equation $p_1u=0$
The differential operators q_1 and q_2 satisfy the following.
$qp_2u=0 \Rightarrow \exists r$ such that $q=rq_1$
$qp_2u=u \Rightarrow \exists r$ such that $q=q_2+rq_1$ and $\operatorname{ord} q_2 \le \operatorname{ord} q$

```
[0] P=os_md.ghg([a,b],[c]);
(-x^2+x)*dx^2+((-a-b-1)*x+c)*dx-b*a
[1] os_md.qdo(P,dx,x);
[(x^2-x)*dx^2+((a+b+3)*x-c-1)*dx+(b+1)*a+b+1,((-x^2+x)*dx+(-a-b-1)*x+c)/(b*a)]
```

16. `mdivisor(`m`,[`x,∂`]|trans=1,step=1,dviout=`t`)`
`mdivisor(`m`,`x`|trans=1,step=1,dviout=`t`)`, `mdivisor(`m`,0|trans=1,step=1,dviout=`t`)`

:: Elementary divisors of a matrix of ordinary differential operators/polynomials with coefficients in rational functions/integers

- The algorithm in the case of a matrix of ordinary differential operators is given by [O3, Lemma 1.10].
- `step=1` : Gives intermediate process with fundamental transformations for lines and columns
- `trans=1` : Returns a list of the elementary divisors, the matrix for a transformation of lines acting from the left, that of columns acting from the right.
- `trans=2` : Returns the list with the above 3 elements + the inverse of the matrix transforming lines + the inverse of the matrix transforming columns
- m is invertible $\Leftrightarrow$ the elementary divisors are $[1, 1, \ldots]$. In this case $R[1]$ is the inverse. Here R is returned with the option `trans=1` and $R[2]$ is th identity matrix.
- `dviout=1` : Displays the intermediate steps by using TEX で表示.
- `dviout=-1` : Returns the above TEX source.
- `dviout=2` : Displays the intermediate steps and the matrices defining the transformations
- `dviout=-2` : Returns the above TEX source.
- `dviout=3` : Returns the result of elementary transformations and the matrices defining the transformations ans their inverses (corresponding to `trans=2`).
- `dviout=-3` : Returns the above TEX source.
- `unim()` gives problems for exercises of calculations of elementary divisors and diagonarizations of matrices.

```
[0] A=os_md.s2m("12-1,2-22,-121");
[ 1 2 -1 ]
[ 2 -2 2 ]
[ -1 2 1 ]
[1] os_md.mdivisor(os_md.mgen(3,0,[x],0)-A,x);
[1,x-2,x^2+2*x-8]
[2] os_md.mdivisor(os_md.mgen(3,0,[x],0)-A,x|step=1);
1: start
[ x-1 -2 1 ]
[ -2 x+2 -2 ]
[ 1 -2 x-1 ]
1: (1,2) -> (1,1)
[ -2 x-1 1 ]
[ x+2 -2 -2 ]
[ -2 1 x-1 ]
1: unit
[ 1 -1/2*x+1/2 -1/2 ]
[ 0 1/2*x^2+1/2*x-3 1/2*x-1 ]
[ 0 -x+2 x-2 ]
2: start
[ 1/2*x^2+1/2*x-3 1/2*x-1 ]
[ -x+2 x-2 ]
2: (1,2) -> (1,1)
[ 1/2*x-1 1/2*x^2+1/2*x-3 ]
[ x-2 -x+2 ]
 [ 2 0 ]*
```

```
 [ -4 2 ]
2: line 1 & 2
[ x-2 x^2+x-6 ]
[ 0 -2*x^2-4*x+16 ]
 *[ 1 -x-3 ]
  [ 0 1 ]
2: column 1 & 2
[ x-2 0 ]
[ 0 -2*x^2-4*x+16 ]
3: start
[ -2*x-8 ]
[1,x-2,x^2+2*x-8]
[3] os_md.mdivisor(os_md.mgen(2,0,[dx],0),[x,dx]|step=1);
1: start
[ dx 0 ]
[ 0 dx ]
1: column 1 += col2*x
[ dx 0 ]
[ x*dx+1 dx ]
 [ -x 1 ]*
 [ x*dx+2 -dx ]
1: line 1 & 2
[ 1 dx ]
[ 0 -dx^2 ]
1: unit
[ 1 dx ]
[ 0 -dx^2 ]
2: start
[ -dx^2 ]
[[1,dx^2],
[4] os_md.mdivisor(os_md.mgen(2,0,[dx],0),[x,dx]|trans=2);
[[1,dx^2],[ -x 1 ]
[ -x*dx-2 dx ],[ 1 -dx ]
[ x -x*dx+1 ],[ dx -1 ]
[ x*dx+1 -x ],[ -x*dx dx ]
[ -x 1 ]]
[5] os_md.mdivisor(os_md.mgen(2,0,[dx],0),[x,dx]|dviout=2);
```

$$\begin{pmatrix} \partial & 0 & 1 & 0 \\ 0 & \partial & 0 & 1 \\ 1 & 0 & & \\ 0 & 1 & & \end{pmatrix}$$
$$C1 \mathrel{+}= C2 \times (x)$$

$$\to \begin{pmatrix} \partial & 0 & 1 & 0 \\ x\partial+1 & \partial & 0 & 1 \\ 1 & 0 & & \\ x & 1 & & \end{pmatrix}$$

$$\begin{pmatrix} -x & 1 \\ x\partial+2 & -\partial \end{pmatrix} \begin{pmatrix} L1 \\ L2 \end{pmatrix}$$

$$\to \begin{pmatrix} 1 & \partial & -x & 1 \\ 0 & -\partial^2 & x\partial+2 & -\partial \\ 1 & 0 & & \\ x & 1 & & \end{pmatrix}$$

$$Cj \mathrel{-}= C1 \times \circ \quad (j>1)$$

$$\to \begin{pmatrix} 1 & 0 & -x & 1 \\ 0 & -\partial^2 & x\partial+2 & -\partial \\ 1 & -\partial & & \\ x & -x\partial+1 & & \end{pmatrix}$$

$$L2 \leftarrow (-1) \times L2$$

$$\to \begin{pmatrix} 1 & 0 & -x & 1 \\ 0 & \partial^2 & -x\partial-2 & \partial \\ 1 & -\partial & & \\ x & -x\partial+1 & & \end{pmatrix}$$

As a result,

$$\begin{pmatrix} 1 & 0 \\ 0 & \partial^2 \end{pmatrix} = \begin{pmatrix} -x & 1 \\ -x\partial-2 & \partial \end{pmatrix} \begin{pmatrix} \partial & 0 \\ 0 & \partial \end{pmatrix} \begin{pmatrix} 1 & -\partial \\ x & -x\partial+1 \end{pmatrix},$$

$$\begin{pmatrix} -x & 1 \\ -x\partial-2 & \partial \end{pmatrix}^{-1} = \begin{pmatrix} \partial & -1 \\ x\partial+1 & -x \end{pmatrix},$$

$$\begin{pmatrix} 1 & -\partial \\ x & -x\partial+1 \end{pmatrix}^{-1} = \begin{pmatrix} -x\partial & \partial \\ -x & 1 \end{pmatrix}.$$

The matrices defining the transformations are obtained by `mygcd()`.

The elements of the matrix m belong one of the following Euclidean rings Q.

(1) The ring of integers

(2) The polynomial ring of one variable whose coefficients are rational functions/numbers

(3) The ring of ordinary differential operators whose coefficients are rational functions

We use the following algorithm.

(a) If $A = (A_{i,j})$ is a zero matrix, we return `[0]`.

(b) We move the non-zero minimal (i,j) element of A to the $(1,1)$ element by permutations of lines and columns of A ((i,j) is chosen to be minimal by the lexicographic order). Here "minimal" means with respect to absolute value in (1) or degree in (2) or rank in (3).

(c) If there exists a non-zero element $A_{i,1}$ with $i>1$, we get $C \in GL(2,S)$ such that the second element of $C\binom{A_{1,1}}{A_{i,1}}$ vanishes (the second element of Cv vanishes ($\leftarrow$ euclidean algorithm). Here we can choose C so that the coefficients of elements of the matrix are polynomials in the cases (2) and (3). Then we apply C to the first and the i-th line of A and go to (b).

(d) If $A_{i,1}=0$ for $i>1$ and there exists a non-zero $A_{1,j}$ with $j>1$, we gets $C \in GL(2,S)$ such that the second element of $(A_{1,1}, A_{1,j})C$ vanishes. Then we apply C to the first column and the j-th column of A and go to (b).

- Hereafter we assume $A_{1,1} \neq 0$ and $A_{i,1} = A_{1,j} = 0$ for $i>1$ and $j>1$.

(e) If the number of lines of A or the number of columns of A is 1, we return `[`$A_{1,1}$`]`.

– Otherwise –

(f) If $A_{1,1}$ is invertible, we replace $A_{1,1}$ by 1 and go to (g) ii).

— In the case (1) or (2) —

(g) We check the existence of the element $A_{i,j}$ which is not divided by $A_{1,1}$.
i) If the above element $A_{i,j}$ exists, we add the j-th column to the first column. Then if $i \neq 2$, we swap 2-nd line and the i-th line. Then go to (b).
ii) If all the elements of A can be divided by $A_{1,1}$, we denote the matrix excluding the 1-st line and 1-st column of A by A' and call this function replacing A by $A_{1,1}^{-1}A'$. If we get $[R'_1, R'_2, \ldots]$ by this call, we return $[A_{1,1}R'_A, A_{1,1}R'_2, \ldots, A_{1,1}]$.

— In the case (3) —

(h) We choose a nonzero $A_{i,j}$ with $i > 1$ and $j > 1$ (the minimal (i, j) in the lexicographic order). We moreover choose a non-negative integer k such that $SA_{i,j}x^k + SA_{1,1} \not\subset SA_{1,1}$. Here we note that $0 \leq k <$ the rank of $A_{1,1}$. Then we change $A_{\nu,1}$ by $A_{\nu,1} + A_{\nu,j}x^k$ for $\nu = 1, 2, \ldots$ and go to (b).

`step=1` : the meaning of the comment displayed above a matrix is as follows.

- The first number id the depth of the nesting of called `mdivisor()`. If the depth increases by 1, the size of the matrices decreases by 1 (cf. (f), (g) ii)).
- `start` : The original matrix when this function is called.
- `(a,b) -> (1,1)` : Swaps the `a`-th line and the first line and moreover the `b`-column and the first column and moves the `(a,b)` element to the `(1,1)` element (cf. (b)).
- `line 1 & a` : Multiplies the first line and the `a`-th line by a matrix in $GL(2, S)$ from the left and changes the `(a,1)` element to 0. (cf. (c)).
- `column 1 & b` : Multiplies the first column and the `b`-th column by a matrix in $GL(2, S)$ from the right and changes the `(1,b)` element to 0. (cf. (d)).
- `unit` : `(1,1)` : The $(1, 1)$ element is invertible (cf. (f)).
- `column 1 += col b, line 2<->a` : Adds the `b`-th column to the first column and swaps 2-nd line and the a-th line (cf. (g) i)).
- `column 1 += col b*x^k` : Adds the column obtained by multiplying the `b`-column by `x^k` from the right to the first column (cf. (h)).

In TEX Li represents the i-th line and C_j represents the j-th column. For example,

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \begin{pmatrix} L3 \\ L5 \end{pmatrix}$$

means that the 3-rd line and the 5-th line are multiplied by $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ from the left.

```
[6] os_md.mdivisor(mat([3,5,7],[5,3,3]),0|dviout=2)
```

$$\begin{pmatrix} 3 & 5 & 7 & 1 & 0 \\ 5 & 3 & 3 & 0 & 1 \\ 1 & 0 & 0 & & \\ 0 & 1 & 0 & & \\ 0 & 0 & 1 & & \end{pmatrix}$$

$$\begin{pmatrix} 2 & -1 \\ -5 & 3 \end{pmatrix} \begin{pmatrix} L1 \\ L2 \end{pmatrix}$$

$$\to \begin{pmatrix} 1 & 7 & 11 & 2 & -1 \\ 0 & -16 & -26 & -5 & 3 \\ 1 & 0 & 0 & & \\ 0 & 1 & 0 & & \\ 0 & 0 & 1 & & \end{pmatrix}$$

$$Cj \mathrel{-}= C1 \times \circ \quad (j > 1)$$

$$\to \begin{pmatrix} 1 & 0 & 0 & 2 & -1 \\ 0 & -16 & -26 & -5 & 3 \\ 1 & -7 & -11 & & \\ 0 & 1 & 0 & & \\ 0 & 0 & 1 & & \end{pmatrix}$$

$$(C2 \quad C3)\begin{pmatrix} -5 & -13 \\ 3 & 8 \end{pmatrix}$$

$$\to \begin{pmatrix} 1 & 0 & 0 & 2 & -1 \\ 0 & 2 & 0 & -5 & 3 \\ 1 & 2 & 3 & & \\ 0 & -5 & -13 & & \\ 0 & 3 & 8 & & \end{pmatrix}$$

As a result,

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \end{pmatrix} = \begin{pmatrix} 2 & -1 \\ -5 & 3 \end{pmatrix}\begin{pmatrix} 3 & 5 & 7 \\ 5 & 3 & 3 \end{pmatrix}\begin{pmatrix} 1 & 2 & 3 \\ 0 & -5 & -13 \\ 0 & 3 & 8 \end{pmatrix},$$

$$\begin{pmatrix} 2 & -1 \\ -5 & 3 \end{pmatrix}^{-1} = \begin{pmatrix} 3 & 1 \\ 5 & 2 \end{pmatrix},$$

$$\begin{pmatrix} 1 & 2 & 3 \\ 0 & -5 & -13 \\ 0 & 3 & 8 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 7 & 11 \\ 0 & -8 & -13 \\ 0 & 3 & 5 \end{pmatrix}.$$

```
[7] os_md.mdivisor(mat([dx,0,0],[0,dx,0],[0,0,dx]),[x,dx]|dviout=3)
```

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \partial^3 \end{pmatrix} = P\begin{pmatrix} \partial & 0 & 0 \\ 0 & \partial & 0 \\ 0 & 0 & \partial \end{pmatrix} Q,$$

$$P = \begin{pmatrix} -x & 1 & 0 \\ -\frac{1}{2}x\partial - 1 & \frac{1}{2}\partial & -\frac{1}{2}x^2\partial - 2x \\ -x\partial^2 - 3\partial & \partial^2 & -x^2\partial^2 - 6x\partial - 6 \end{pmatrix} = \begin{pmatrix} \partial & -x^2\partial^2 - 4x\partial - 2 & \frac{1}{2}x^2\partial + 2x \\ x\partial + 1 & -x^3\partial^2 - 4x^2\partial - 2x & \frac{1}{2}x^3\partial + 2x^2 \\ 0 & \partial & \frac{-1}{2} \end{pmatrix}^{-1},$$

$$Q = \begin{pmatrix} 1 & -x^2\partial - 2x & \frac{1}{2}x^2\partial^3 + x\partial^2 - \partial \\ x & -x^3\partial - x^2 & \frac{1}{2}x^3\partial^3 + \frac{1}{2}x^2\partial^2 - x\partial + 1 \\ 0 & 1 & -\frac{1}{2}\partial^2 \end{pmatrix} = \begin{pmatrix} -x\partial & \partial & 0 \\ -\frac{1}{2}x\partial^2 - \partial & \frac{1}{2}\partial^2 & -\frac{1}{2}x^2\partial^2 - 2x\partial \\ -x & 1 & -x^2 \end{pmatrix}^{-1}.$$

```
[8] A=mat([2,-2,-2],[0,1,-1],[0,0,2])$
[9] os_md.mdivisor(os_md.mgen(3,0,[x],0)-A,x|dviout=2)$
```

$$\begin{pmatrix} x-2 & 2 & 2 & 1 & 0 & 0 \\ 0 & x-1 & 1 & 0 & 1 & 0 \\ 0 & 0 & x-2 & 0 & 0 & 1 \\ 1 & 0 & 0 & & & \\ 0 & 1 & 0 & & & \\ 0 & 0 & 1 & & & \end{pmatrix}$$

$C1 \leftrightarrow C2$

$$\to \begin{pmatrix} 2 & x-2 & 2 & 1 & 0 & 0 \\ x-1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & x-2 & 0 & 0 & 1 \\ 0 & 1 & 0 & & & \\ 1 & 0 & 0 & & & \\ 0 & 0 & 1 & & & \end{pmatrix}$$

$L1 \leftarrow (\frac{1}{2}) \times L1$

$$\to \begin{pmatrix} 1 & \frac{1}{2}(x-2) & 1 & \frac{1}{2} & 0 & 0 \\ x-1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & x-2 & 0 & 0 & 1 \\ 0 & 1 & 0 & & & \\ 1 & 0 & 0 & & & \\ 0 & 0 & 1 & & & \end{pmatrix}$$

$Li \mathrel{-}= \circ \times L1 \quad (i > 1)$

$$\to \begin{pmatrix} 1 & \frac{1}{2}(x-2) & 1 & \frac{1}{2} & 0 & 0 \\ 0 & -\frac{1}{2}(x-2)(x-1) & -(x-2) & -\frac{1}{2}(x-1) & 1 & 0 \\ 0 & 0 & x-2 & 0 & 0 & 1 \\ 0 & 1 & 0 & & & \\ 1 & 0 & 0 & & & \\ 0 & 0 & 1 & & & \end{pmatrix}$$

$Cj \mathrel{-}= C1 \times \circ \quad (j > 1)$

$$\to \begin{pmatrix} 1 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & -\frac{1}{2}(x-2)(x-1) & -(x-2) & -\frac{1}{2}(x-1) & 1 & 0 \\ 0 & 0 & x-2 & 0 & 0 & 1 \\ 0 & 1 & 0 & & & \\ 1 & -\frac{1}{2}(x-2) & -1 & & & \\ 0 & 0 & 1 & & & \end{pmatrix}$$

$L2 \leftrightarrow L3,\ C2 \leftrightarrow C3$

$$\to \begin{pmatrix} 1 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & x-2 & 0 & 0 & 0 & 1 \\ 0 & -(x-2) & -\frac{1}{2}(x-2)(x-1) & -\frac{1}{2}(x-1) & 1 & 0 \\ 0 & 0 & 1 & & & \\ 1 & -1 & -\frac{1}{2}(x-2) & & & \\ 0 & 1 & 0 & & & \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} L2 \\ L3 \end{pmatrix}$$

$$\to \begin{pmatrix} 1 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & x-2 & 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{1}{2}(x-2)(x-1) & -\frac{1}{2}(x-1) & 1 & 1 \\ 0 & 0 & 1 & & & \\ 1 & -1 & -\frac{1}{2}(x-2) & & & \\ 0 & 1 & 0 & & & \end{pmatrix}$$

$L3 \leftarrow (-2) \times L3$

$$\to \begin{pmatrix} 1 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & x-2 & 0 & 0 & 0 & 1 \\ 0 & 0 & (x-2)(x-1) & x-1 & -2 & -2 \\ 0 & 0 & 1 & & & \\ 1 & -1 & -\frac{1}{2}(x-2) & & & \\ 0 & 1 & 0 & & & \end{pmatrix}$$

As a result,

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & x-2 & 0 \\ 0 & 0 & (x-2)(x-1) \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 \\ x-1 & -2 & -2 \end{pmatrix} \begin{pmatrix} x-2 & 2 & 2 \\ 0 & x-1 & 1 \\ 0 & 0 & x-2 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 \\ 1 & -1 & -\frac{1}{2}(x-2) \\ 0 & 1 & 0 \end{pmatrix}$$

$$\begin{pmatrix} \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 \\ x-1 & -2 & -2 \end{pmatrix}^{-1} = \begin{pmatrix} 2 & 0 & 0 \\ x-1 & -1 & -\frac{1}{2} \\ 0 & 1 & 0 \end{pmatrix},$$

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & -1 & -\frac{1}{2}(x-2) \\ 0 & 1 & 0 \end{pmatrix}^{-1} = \begin{pmatrix} \frac{1}{2}(x-2) & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

17. `sqrtdo(`p`,[`x`,`∂_x`])` ?
18. `toeul(`p`,[`x`,`∂_x`],`n`)`
:: Expresses a (Fuchsian) differential operator p in Euler type at $x = n$
Transforms p by the coordinate transformation $x \mapsto x + n$ and replaces $x\partial_x$ by ∂_x and changes it so that the coefficients are polynomials
If n is the string `"infty"`, transforms p by the coordinate transformation $x \mapsto x + n$ and replaces $x\partial_x$ by ∂_x and multiplies it by a suitable power of x.

```
[0] os_md.toeul(os_md.ghg([a,b],[c]), x, 0);
(-x+1)*dx^2+((-a-b)*x+c-1)*dx-b*a*x
[1] os_md.toeul(os_md.ghg([a,b],[c]), x, "infty");
(x-1)*dx^2+((-c+1)*x+a+b)*dx-b*a
```

19. `fromeul(`p`,[`x`,`p_x`],`n`)`
:: Transforms the differential operator p of Euler type into normal form (the above inverse)
Multiplies by a power of $x - n$ so that the coefficient does not vanish at $x = n$.

```
[0] os_md.fromeul(dx,x,1);
dx
[1] os_md.fromeul(dx-a,x,1);
(x-1)*dx-a
```

20. `expat(`p`,[`x`,`∂_x`],`n`)`
:: Returns characteristic exponents of a differential operator p at a regular singular point $x = n$
- n may be a string `"infty"` (which means $n = \infty$).
- If p is Fuchsian and n is `"?"`, the exponents at all singular points are obtained.
- See (69) for `ghg()` in the following. It is the operator define Gauss hypergeometric equation.

```
[0] os_md.expat(os_md.ghg([a,b,c],[d,e]), x, 0);
[-d+1,-e+1,0]
[1] os_md.expat(os_md.ghg([a,b,c],[d,e]), x, 1);
[-a-b-c+d+e,1,0]
[2] os_md.expat(os_md.ghg([a,b,c],[d,e]), x, 2);
[1,2,0]
[3] os_md.expat(os_md.ghg([a,b,c],[d,e]), x, "infty");
[a,b,c]
[4] os_md.expat(os_md.ghg([a,b,c],[d,e]), x, "?");
[[0,[-d+1,-e+1,0]],[1,[-a-b-c+d+e,1,0]],[infty,[a,b,c]]]
```

21. `sftexp(`p`,[`x`,`∂_x`],`n`,`r`)`
:: Returns $(x - n)^{-r} \circ p \circ (x - n)^r$
Here $[\partial_x, r] = 0$ should valid.

```
[0] P=os_md.ghg([a,b],[c]);
(-x^2+x)*dx^2+((-a-b-1)*x+c)*dx-b*a
[1] os_md.sftexp(P,x,0,1-c);
(-x^2+x)*dx^2+((-a-b+2*c-3)*x-c+2)*dx+(-b+c-1)*a+(c-1)*b-c^2+2*c-1
```

```
[2] Q=os_md.sftexp(P,x,"infty",b);
(-x^3+x^2)*dx^2+((-a+b-1)*x^2+(-2*b+c)*x)*dx+b^2+(-c+1)*b
[3] os_md.expat(Q,x,"infty");
[a-b,0]
```

22. `fractrans(`p`,[`x,∂_x`],`n_0,n_1,n_2`)`
:: Transforms p by the linear fractional transformation of x with $(n_0, n_1, n_2) \mapsto (0, 1, \infty)$

```
[0] os_md.fractrans(os_md.ghg([a,b],[c]),x,1,0,"infty");
(-x^2+x)*dx^2+((-a-b-1)*x+a+b-c+1)*dx-b*a
[1] P=os_md.fractrans(os_md.ghg([a,b],[c]),x,"infty",1,0);
(x^3-x^2)*dx^2+((-c+2)*x^2+(a+b-1)*x)*dx-b*a
[2] os_md.expat(P,x,1);
[-a-b+c,0]
[3] os_md.expat(P,x,0);
[a,b]
```

23. `chkexp(`p`,[`x,∂_x`],`n,r,m`)`
:: Returns condition so that p has the exponent $[r]_{(m)}$ at $x = n$

```
[0] os_md.chkexp(os_md.ghg([a,b],[c]),x,0,0,1);
[]
[1] os_md.chkexp(os_md.ghg([a,b],[c]),x,"infty",0,2);
[a+b-1,-b*a]
[2] os_md.chkexp(os_md.ghg([a,b],[c]),x,"infty",0,1);
[-b*a]
```

24. `soldif(`p`,[`x,∂_x`],`n,q,m`)`

25. `okuboetos(`p`, [`x,∂_x`] | diag=[`$c_1, c_2, \ldots$`])`
:: Transforms a single ODE of Okubo type to 1st order Okubo system
The ODE $Pu = 0$ of order m with polynomial coefficients is of Okubo type if the degree of the coefficients of the n-th derivatives are of degree at most n $(n = 0, \ldots, m)$.
   - Returns `[[`$a_0, \ldots, a_{m-1}$`], `B, T`]`
   - `diag=[`$c_0, c_1, \ldots$`]` : indicates the differentt order from `[`$a_0, \ldots, a_{m-1}$`]`.
   - A single Fuchsian ODE is changed to a Fuchsian ODE of Okubo type by several transformations of $p \mapsto p * \partial_x + p'$.

$$(x - a_i)u_i' = \sum_{j=0}^{m-1} B_{ij} u_j \qquad (i = 0, \ldots, m-1),$$

$$u_i = \sum_{j=0}^{m-1} T_{i,j} u^{(j)}.$$

```
[0] P = os_md.ghg([a,b],[c]);
(-x^2+x)*dx^2+((-a-b-1)*x+c)*dx-b*a
[1] R = os_md.okuboetos(P,x)$
[2] R[0];
[ 0 1 ]
[3] R[1];
[ -c+1 1 ]
```

```
[ (-b+c-1)*a+(c-1)*b-c^2+2*c-1 -a-b+c-1 ]
[4] R[2];
[ 1 0 ]
[ c-1 x ]
[5] det(R[1]);
b*a
[6] os_md.fctrtos(R[1][1][0]);
-(b-c+1)*(a-c+1)
[7] R = os_md.okuboetos(P,x|diag=[1,0])$
[8] R[0];
[ 1 0 ]
[9] R[1];
[ -a-b+c 1 ]
[ (-b+c)*a+c*b-c^2 -c ]
[10] R[2];
[ 1 0 ]
[ a+b-c x-1 ]]
```

These mean

$$u = F(a,b,c;x),$$
$$\begin{pmatrix} u_0 \\ u_1 \end{pmatrix} = \begin{pmatrix} u \\ (c-1)u + xu' \end{pmatrix},$$
$$\begin{pmatrix} x & \\ & x-1 \end{pmatrix} \begin{pmatrix} u_0' \\ u_1' \end{pmatrix} = \begin{pmatrix} 1-c & 1 \\ -(b-c+1)(a-c+1) & -a-b+c-1 \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \end{pmatrix}.$$

In particular

$$v = \begin{pmatrix} v_0 \\ v_1 \end{pmatrix} = \begin{pmatrix} u \\ \frac{c-1}{a-c+1}u + \frac{x}{a-c+1}u' \end{pmatrix},$$
$$\begin{pmatrix} x & \\ & x-1 \end{pmatrix} \begin{pmatrix} v_0' \\ v_1' \end{pmatrix} = \begin{pmatrix} 1-c & a-c+1 \\ -(b-c+1) & -a-b+c-1 \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \end{pmatrix},$$
$$\frac{dv}{dx} = \frac{\begin{pmatrix} 1-c & a-c+1 \\ 0 & 0 \end{pmatrix}}{x} v + \frac{\begin{pmatrix} 0 & 0 \\ -b+c-1 & -a-b+c-1 \end{pmatrix}}{x-1} v$$

26. `stoe(`p`,[`x, dx`],`m`)`

:: Converts a first order system of ODE to a single ODE

- Returns the equation satisfied by u_m for the system $u_i' = \sum_{j \geq 0} p_{ij}(x) u_j$. Here p is a matrix with elements in rational functions.
- If m is a list $[m_1, m_2]$, then the expression of u_{m_2} in terms of u_{m_2} is returned, namely, the operator applying to u_{m_2} to get u_{m_1}.
- If m is negative, it implies $[-m, 0]$.
- See `baseODE()` for non-linear ODE.

```
[0] A=newmat(2,2,[[(-c+1)/(x),(a-c+1)/(x)],[(-b+c-1)/(x-1),(-a-b+c-1)/(x-1)]]);
[ (-c+1)/(x) (a-c+1)/(x) ]
[ (-b+c-1)/(x-1) (-a-b+c-1)/(x-1) ]
[1] os_md.stoe(A,x,0);
```

```
(x^2-x)*dx^2+((a+b+1)*x-c)*dx+b*a
[2] T=os_md.mgen(4,0,[x,x-1,x,x-1],0);
[ x 0 0 0 ]
[ 0 x-1 0 0 ]
[ 0 0 x 0 ]
[ 0 0 0 x-1 ]
[3] A=newmat(4,4,[[a1,1],[a21,a2,1],[a31,a32,a3,1],[a41,a42,a43,a4]]);
[ a1 1 0 0 ]
[ a21 a2 1 0 ]
[ a31 a32 a3 1 ]
[ a41 a42 a43 a4 ]
[4] C=os_md.myinv(T)*A;
[ (a1)/(x) (1)/(x) 0 0 ]
[ (a21)/(x-1) (a2)/(x-1) (1)/(x-1) 0 ]
[ (a31)/(x) (a32)/(x) (a3)/(x) (1)/(x) ]
[ (a41)/(x-1) (a42)/(x-1) (a43)/(x-1) (a4)/(x-1) ]
[5] P=os_md.stoe(C,x,0)$
[6] os_md.expat(P,x,"?");
[[0,[a3+1,a1,1,0]],[1,[a4+2,a2+1,1,0]],
[infty,[dx^4+(a4+a3+a1+a2)*dx^3+(-a21-a32-a43+(a3+a1+a2)*a4+(a1+a2)*a3+a2*a1)
*dx^2+((-a4-a3)*a21+(-a4-a1)*a32+(-a1-a2)*a43+a31+a42+((a1+a2)*a3+a2*a1)*a4+
a2*a1*a3)*dx+(a43-a3*a4)*a21-a1*a4*a32-a41-a2*a1*a43+a4*a31+a1*a42+a2*a1*a3*a4]]]
[7] os_md.mperm(A,[0,2,1,3],1);
[ a1 0 1 0 ]
[ a31 a3 a32 1 ]
[ a21 1 a2 0 ]
[ a41 a43 a42 a4 ]
[8] Q=E[2][1][0]-os_md.polbyroot([1,4],dx|var=a);
(-a21-a32-a43)*dx^2+((-a4-a3)*a21+(-a4-a1)*a32+(-a1-a2)*a43+a31+a42)*dx+(a43
-a3*a4)*a21-a1*a4*a32-a41-a2*a1*a43+a4*a31+a1*a42
```

27. `etos(`p`,[`x, dx`],`m`)`
:: Converts a single ODE to a first order system of ODE Converts a single equation $Pu = 0$ of the order n to a first-order system $\tilde{u}' = A\tilde{u}$ and returns A.
    - m is a matrix with elements in rational functions. Let m_{ij} be the (i, j)-element of m. Then the system is $u_i = \sum m_{ij} u^{(j)}$.
    - $m = 1$ means that m is the identity matrix.
    - If m is a list with length n, then m is identified with the diagonal matrix whose diagonal elements are given by m,

```
[0] P=(-x^2+x)*dx^2+((-a-b-1)*x+c)*dx-a*b$
[1] M=os_md.etos(P,x,[1,x]);
[ 0 1 ]
[  (-b*a)/(x-1) ((-a-b-1)*x+c)/(x^2-x) ]
[2] os_md.pfrac(M,x|dviout=1);
```

$$\frac{\begin{pmatrix}0 & 1\\ 0 & 0\end{pmatrix}}{x}+\frac{\begin{pmatrix}0 & 0\\ -ba & -(a+b-c+1)\end{pmatrix}}{x-1}$$

28. `dform(`ℓ, x`|dif=1)`
:: Calculates differential 1-form $\sum \ell[i][0]d(\ell[i][1])$ or 2-form $\sum \ell[i][0]d(\ell[i][1])\wedge d(\ell[i][2])$ with variables $x[0], x[1], \ldots$. Option `dif=1` means external derivative of 1-form
   - $\ell[i][j]$ can be matrices
   - `dif=1` : exterior derivative of differential 1-form

```
[0] os_md.dform([[a*y,x/y],[b*x,y/x]],[x,y]);
[(a*x-b*y)/(x),x],[(-a*x+b*y)/(y),y]]
[1] os_md.dform([[a*y,x/y,x],[b*x,y/x,y]],[x,y]);
[[(a*x^2-b*y^2)/(y*x),x,y]]
[2] os_md.dform([[x-y,y],[x-y+1,x]],[x,y]|dif=1);
[[2,x,y]]
```

Yhe above means

$$\begin{aligned} ay\cdot d(\tfrac{x}{y})+bx\cdot d(\tfrac{y}{x}) &= \tfrac{ax-by}{x}dx+\tfrac{-ax+by}{y}dy\\ ax\cdot d(\tfrac{x}{y})\wedge dx+bx\cdot d(\tfrac{y}{x})\wedge dy &= \tfrac{ax^2-by^2}{xy}dx\wedge dy\\ d\big((x-y)dy+(x-y-1)dx\big) &= 2dx\wedge dy \end{aligned}$$

29. `solpokubo(`p`,[`x, ∂_x`],`n`)`
:: Returns eigenvalues and eigenpolynomials of a single ODE of Okubo type
Here the differential operator

$$p=a_m(x)\tfrac{d^m}{dx^m}+\cdots+a_1(x)\tfrac{d}{dx}+a_0(x)$$

with polynomial coefficients is of Okubo type when $\deg_x a_m(x)=m$ and $\deg_x a_\nu(x)\le\nu$ for $\nu=0,\ldots,m-1$.

```
[0] P=x*(1-x)*dx^2+(c-a*x)*dx;
(-x^2+x)*dx^2+(-a*x+c)*dx
[1] os_md.solpokuboe(P,[x,dx],1);
[a*x-c,-a]
[2] os_md.solpokuboe(P,[x,dx],2);
[(a^2+3*a+2)*x^2+((-2*c-2)*a-2*c-2)*x+c^2+c,-2*a-2]
[3] os_md.fctrtos(@@[0]|var=x);
(a+1)*(a+2)*x^2-2*(c+1)*(a+1)*x+c*(c+1)
```

3.1.2 Fractional calculus

The functions in this section realize the results of [O3], [O5], [O7], [O9] and [O10].

30. `laplace(`p`,[`x, ∂_x`])`
`laplace(`p`,[[`x_1, ∂_{x_1}`],[`x_2, ∂_{x_2}`],...])`
:: (partial) Laplace transform of a differential operator p $\quad (x,\partial_x)\mapsto(-\partial_x, x)$

```
[0] os_md.laplace(x^2*dx+1,x);
x*dx^2+2*dx+1
```

31. `laplace1(`p`,[`x,∂_x`])`
    `laplace1(`p`,[[`x_1,∂_{x_1}`],[`x_2,∂_{x_2}`],...])`
    :: Inverse of (partial) Laplace transform of a differential operator p
    $(x,\partial_x)\mapsto(\partial_x,-x)$

```
[0] os_md.laplace1(x^2*dx+1,x);
-x*dx^2-2*dx+1
```

32. `mc(`p`,[`x,∂_x`],`r`)`
    :: Middle convolution $mc_r(p)$ of a differential operator p

```
[0] P=os_md.mc(x*(1-x)*dx-a-b*x,x,r);
(x^2-x)*dx^2+((-2*r+b+2)*x+r+a-1)*dx+r^2+(-b-1)*r+b
[1] os_md.expat(P,x,"?");
[[0,[r+a,0]],[1,[r-a-b,0]],[infty,[-r+b,-r+1]]]
```

33. `mce(`p`,[`x,∂_x`],`n`,`r`)`
    :: Transformation p into $(\partial_x-n)^{-r}\circ p\circ(\partial_x-n)^r$and returns corresponding differential operator
    Here n should commute with ∂_x. If $n=0$, this corresponds to the middle convolution $mc_r(p)$.
34. `rede(`p`,[`x,∂_x`])`
    `rede(`p`,[[`x_1,∂_{x_1}`],[`x_2,∂_{x_2}`],...])`
    :: Reduced representative of a differential operator p

```
[0] os_md.rede(x*(y^2-1)*dx+(y+1)/x*dy,[[x],y]);
dy+(y-1)*x^2*dx
```

35. `ad(`p`,[`x,∂_x`],`f`)`
    :: Transform of a differential operator p defined by $\partial_x\to\partial_x-f$
36. `add(`p`,[`x,∂_x`],`f`)`
    :: Addition of a differential operator p defined the map $\partial_x\to\partial_x-f$, namely, `rede(ad())`

```
[0] os_md.add(dx^2+x,x,1/x^2);
x^4*dx^2-2*x^2*dx+x^5+2*x+1
```

37. `vadd(`p`,[`x,∂_x`],[[`c_0,r_0`],[`c_1,r_1`],...])`
    :: Versal addition `add(`p`,[`x,∂_x`],`$\sum_{j\geq 0}\frac{r_jx^j}{\prod_{\nu=0}^{j}(1-c_\nu x)}$`)`
38. `addl(`p`,[`x,∂_x`],`f`)`
    :: `laplace1(add(laplace()))`

```
[0] os_md.add(x,x,a/(x-c));
-x*dx+c*x-a-1
```

39. `cotr(`p`,[`x,∂_x`],`f`)`
    :: Transform of a differential operator p defined by $x\mapsto f(x)$
40. `rcotr(`p`,[`x,∂_x`],`f`)`
    :: Reduced representative of the transformation P defined by $x\mapsto f(x)$
41. `s2sp(`p`|num=1,std=k,short=1)`
    :: Converts a list of lists of numbers to and from its expression using a list of strings
    - $10,11,12,\ldots,35$ are written by `a,b,c,...,z`, respectively. で表す. A negative number and `^` and a fractional number is allowed.
    - `^` means the repetition of a same number (does not mean a power)
    - a number larger than 9 can be expressed with "(" and ")" but its nesting is not allowed.

- `num=1` : `a,b,c,...` is not used for the return but use "(" and ")".
- For the parameters p, $36, 37, \ldots, 60$ may be expressed by $A, B, \ldots, Z$.
- The element in p which is not a rational number is expressed by a string with `< >`.
- `std=`k : the spectral types are arranged with the numbers according to the multiplicities at each singular points ns and the singular points are arranged in the lexicographic order
  $k = 1$: started from smaller ones
  $k = -1$: started from larger ones
- `short=1` : if the same number repeats more that 3 times, the expression with `^*` is used. In this case the option `std=`k is also allowed.

```
[0] os_md.s2sp("121,22,211");
[[1,2,1],[2,2],[2,1,1]]
[1] os_md.s2sp(@@);
121,22,211
[2] os_md.s2sp("5^2,541,1^a");
[[5,5],[5,4,1],[1,1,1,1,1,1,1,1,1,1]]
[3] os_md.s2sp("2-3a,1^9");
[[2,-3,10],[1,1,1,1,1,1,1,1,1]]
[4] newmat(4,4,os_md.s2sp("1,01,001,0001"));
[ 1 0 0 0 ]
[ 0 1 0 0 ]
[ 0 0 1 0 ]
[ 0 0 0 1 ]
[5] S=os_md.s2sp("1(-15)a-b,fg-7/(80)");
[[1,-15,10,-11],[15,16,-7/80]]
[6] os_md.s2sp(S);
1(-15)a(-11),fg(-7/80)
[7] os_md.s2sp(S|num=1);
1(-15)(10)(-11),(15)(16)(-7/80)
[8] os_md.s2sp([[1,2],[a-b,2.5]]);
12,<a-b><2.5>
[9] os_md.s2sp(@@);
[[1,2],[a-b,2.5]]
[10] os_md.s2sp(os_md.s2sp("32,2111,41,23"|std=1));
2111,32,32,41
[11] os_md.s2sp(os_md.s2sp("32,2111,41,23"|std=-1));
41,32,32,2111
[12] os_md.s2sp("111111,33,222"|short=1,std=-1);
33,222,1^6
```

42. `s2csp(`p`|n=`f`)`
:: Converts expressions of a spectral type with unramified irregular singularities
   - A spectral type with confluent singularities is a list of singularities at each singular points
   - Spectral type of an irregular singular point is nested lists of multiplicities according to the levels of irregularities
   - A spectral type with unramified irregular singularities is expressed a string according to its versal unfolding (cf. [O10]).

- $n=1$: Supports an expression with "(" and ")" (cf. [Hiroe-Kawakami-Nakamura-Sakai]).
- $n=-1$: Supports the above expression both for input and output.

```
[0] L=os_md.s2csp("111,21,111");
[[1,1,1],[2,1],[1,1,1]]
[1] os_md.s2csp(L);
111,21,111
[2] L=os_md.s2csp("111|21,111");
[[[2,[1,1]],[1,[1]]],[1,1,1]]
[3] os_md.s2csp(L);
111|21,111
[4] os_md.s2csp(L|n=1);
(1 1) (1),1 1 1
[5] L=os_md.s2csp("111|111,21");
[[[1,[1]],[1,[1]],[1,[1]]],[2,1]]
[6] os_md.s2csp(L);
111|111,21
[7] os_md.s2csp(L|n=1);
(1) (1) (1),2 1
[8] L=os_md.s2csp("111|111|21");
[[[2,[[1,[1]],[1,[1]]]],[1,[[1,[1]]]]]]
[9] os_md.s2csp(L);
111|111|21
[10] os_md.s2csp(L|n=1);
((1) (1)) ((1))
[11] os_md.s2csp("((1)(1))((1))"|n=1);
111|111|21
[12] os_md.s2csp("(1)(1)(1),21"|n=-1);
[[[1],[1],[1]],[2,1]]
[13] os_md.s2csp( [[[1],[1],[1]],[2,1]]|n=-1);
(1) (1) (1),2 1
```

43. `chkspt(`m`|mat=1,dumb=1)` or `chkspt(`m`|opt=`t`,dumb=1)` or `fspt(`m`,`t`)`
:: Checks a tuple of partitions m (spectral type) or a generalized Riemann scheme (GRS) and returns $[pts, ord, idx, fuchs, rod, redsp, fspt]$
pts : number of singular points
ord : rank
idx : index of rigidity
$fuchs$: Fuchs relation
rod : rank reduced by one-step reduction
$redsp$: the positions of exponents at singular points specified by one-step reduction
$fspt$: corresponding basic spectral type
  - `mat=1` : Schleginger type
  - `dumb=1` : stop showing errors
  - return : -1 means the tuple is not correct, 0 means that the spectral type is not realizable
  - `opt="sp"` or 0: return spectral type of the given GRS
  - `opt="basic"` or 1: `chkspt(`m`)` : return $fspt$ (return 0 if m is not realizable)

- `opt="construct"` or 2: return the construction from the basic spectral type (return 0 if m is not realizable)
- `opt="strip"` or 3: delete the exponents with the multiplicity 0 from the generalized Riemann scheme or the spectral type
- `opt="short"` or 4: give GRS in a short form
- `opt="long"` or 5: give GRS in a normal form
- `opt="sort"` or 6: return a sorted spectral type at each singular point
- `opt="root"` or 7: return the construction by Kac-Moody Weyl group
  The returned list has 3 elements which are base spectral type, given spectral type and the list of reflections. The reflection has 3 elements, namely, the value of inner product, the branch and the position in the star-shaped Dynkin diagram which specified the root in the Dynkin diagram.
- `opt="idx"` : return index of rigidity

The short form means that the exponents with free multiplicity are given the values without the multiplicity.

```
[0] os_md.chkspt([[1,2,1],[2,2],[1,1,1,1]]);
[3,4,2,0,1,[ 1 0 0 ],[[1],[1],[1]]]]
[1] os_md.chkspt("121,22,1111");
[3,4,2,0,1,[ 1 0 0 ],[[1],[1],[1]]]]
[2] M=os_md.sp2grs([[1,1,1,1],[2,1,1],[2,2]],[a,b,c],[1,1]);
[[[1,-2*c1-2*c0-b1-2*b0-a1-b2-a3-a2+3],[1,a1],[1,a2],[1,a3]],
[[2,b0],[1,b1],[1,b2]],[[2,c0],[2,c1]]]
[3] os_md.sp2grs([[1,1,1,1],[2,1,1],[2,2]],[a,b,c],[1,1]|mat=1);
[[[1,-2*c1-2*c0-b1-2*b0-a1-b2-a3-a2],[1,a1],[1,a2],[1,a3]],
[[2,b0],[1,b1],[1,b2]],[[2,c0],[2,c1]]]
[4] os_md.chkspt(M|opt="sp");
[[1,1,1,1],[2,1,1],[2,2]]
[5] os_md.chkspt(M|opt="basic");
[[1],[1],[1]]
[6] os_md.chkspt(M|opt="construct");
[
 [[1],[1],[1]],
 [[1,1],[1,1],[1,1]],
 [[1,1,1],[1,1,1],[2,1]],
 [[1,1,1,1],[2,1,1],[2,2]]
]
[7] os_md.chkspt(M|opt="short");
[[-2*c1-2*c0-b2-b1-2*b0-a2-a1-a3+3,a1,a2,a3],
[[2,b0],b1,b2],[[2,c0],[2,c1]]]
[8] os_md.chkspt([[0,1,2,1],[2,0,2,0],[1,1,1,1]]|opt="strip");
[[2,1,1],[2,2],[1,1,1,1]]
[9] os_md.chkspt([[0,1,2,1],[2,0,2,0],[1,1,1,1]]|opt="sort");
[[2,1,1,0],[2,2,0,0],[1,1,1,1]]
[10] os_md.chkspt("21,21,21,21"|opt="root");
[[[1],[1],[1],[1]],[[2,1],[2,1],[2,1],[2,1]],
```

`[[1,3,1],[1,2,1],[1,1,1],[1,0,1],[2,0,0]]]`

44. `spgen(`n`|eq=1,str=1,std=`f`,pt=[`k,ℓ`],sp=`m`,basic=1)`
:: Gets rigid tuples of partitions with the rank $\leq n$ (or the orbit of a given tuple)
If $n \leq 0$, basic tuples with the index of the rigidity n are obtained.
- `eq=1` : the rank is exactly n (when $n > 1$)
- `str=1` : returns in string form
- `pt=[`k,ℓ`]` : the number of the partition is in $[k,\ell]$. `pt=`k means $k = \ell$
- `sp=`m : returns tuples in the orbit of a tuple (spectral type) m only to the direction with higher rank.
If `basic=1` is indicated, the direction is not specified.
- `std=`f : The multiplicities are arranged from larger ones at each partition and the partitions are arranged from smaller (resp. larger) tuples if $f = 1$ (resp. $f = -1$).

```
[0] os_md.spgen(4|eq=1,pt=4);
[[[2,2],[2,2],[2,2],[3,1]],[[2,2],[3,1],[3,1],[2,1,1]]]
[1] ltov(os_md.spgen(4|eq=1,pt=4,str=1));
[ 22,22,22,31 22,31,31,211 ]
[2] ltov(os_md.spgen(4|eq=1,pt=4,str=1,std=-1));
[ 31,22,22,22 31,31,22,211 ]
[3] ltov(os_md.msort(os_md.spgen(4|eq=1,pt=4,str=1,std=-1),[-1,0]));
[ 31,31,22,211 31,22,22,22 ]
[4] ltov(os_md.spgen(-2|pt=4,str=1));
[ 21,21,111,111 22,22,22,211 31,22,22,1111 ]
```

[0] gets the list of rigid spectral types with 4 singular points whose rank equals 4.
[1] gets the same list as above in the string form
[2] gets the list of basic spectral types with 4 singular points whose index of rigidity equals -2

An example of a program :

```
/* Get the list of rigid spectral types with rank 8 which have 4 singular points
   in the lexicographic order from the larger ones (in strings format) */
Rank=8; /* give rank */
[5] G=os_md.spgen(Rank|eq=1,pt=4); /* get spectral types */
[6] for(L=[];G!=[];G=cdr(G))
      L=cons(os_md.s2sp(os_md.s2sp(car(G)|std=-1)),L);
[7] L=msort(L,[-1,0]);
```

45. `sproot(`p`,`t`|dviout=1,only=`k`,sym=`t`,null=1)`
:: Returns informations of the root corresponding to a spectral type t=`"base"`, `"length"`, `"type"`, `"part"`, `"pair"`, `"pairs"`, sp
- `length` : the length of the element of the Weyl group which changes p basic.
- `base` : same as `chksp(`p`|opt="root")` . The minimal expression of the element.
- `type` : the type of the element (= $[\Delta(\mathbf{m})]$, cf. [O3, (7.40)])
- `height` : the hight of the corresponding root (the sum of the coefficients in the linear expression of the root in terms of simple roots cf. [O3, (7.35)])
- `part` : data for the set of real roots changed from positive to negative under the element (= $\Delta(\mathbf{m})$: corresponds to the condition of irreducibility. cf. [O3, (7.30)])
See `chkspt(`m`|opt=root)` for the data.

- `pair`, `pairs`:decompositions of root corresponding to the reducibility. See [9] for the difference between `pair` and `pairs`.
  `dviout=1` and `only=`k can be indicated. roots are indicated by the corresponding partitions
  `iand`$(k,1)=1$: the decomposition whose paired one is also a partition
  `iand`$(k,2)=2$: the decomposition whose paired one is a root with order 0
  `iand`$(k,4)=4$: the decomposition whose paired one is a negative root
  `sym=1` : identifies the decompositions under symmetries
  `sym=2` : identifies the decompositions under symmetries only at each singular point
  `null=1` : return nothing if there is no corresponding decompositions
- sp : If spectral type is indicated, the inner product with p as roots is returned

```
[0] os_md.sproot("11,11,11","height");
5
[1] os_md.sproot("11,11,11","length");
4
[2] os_md.sproot("11,11,11","type");
[[4,1]]
[3] os_md.sproot("11,11,11","base");
[[[1],[1],[1]],[[1,1],[1,1],[1,1]],
 [[1,2,1],[1,1,1],[1,0,1],[1,0,0]]]
[4] os_md.sproot("11,11,11","part");
[[[1],[1],[1]],[[1,1],[1,1],[1,1]],
[[1,[[1,0],[1,0],[0,1]]],[1,[[1,0],[0,1],[1,0]]],
[1,[[0,1],[1,0],[1,0]]],[1,[[1,0],[1,0],[1,0]]]]]
[4] os_md.sproot("11,11,11","11,11,11");
2
[5] os_md.sproot("11,11,11","10,10,10");
1
[6] os_md.sproot("31,31,22,211","length");
8
[7] os_md.sproot("31,31,22,211","type");
[[6,1],[2,2]]
[8] os_md.sproot("31,31,22,211","height");
11
[9] os_md.sproot("31,31,22,211","pairs"|dviout=1);
```

$$\begin{aligned}
31,31,22,211 &= 10,10,01,001 \oplus 21,21,21,210\\
&= 20,11,11,110 \oplus 11,20,11,101\\
&= 11,20,11,110 \oplus 20,11,11,101\\
&= 10,10,01,010 \oplus 21,21,21,201\\
&= 10,10,10,001 \oplus 21,21,12,210\\
&= 10,10,10,010 \oplus 21,21,12,201\\
&= 2(10,10,01,100) \oplus 11,11,20,011\\
&= 2(10,10,10,100) \oplus 11,11,02,011
\end{aligned}$$

The first terms in the left hand side in the above correspond to the positive real roots in $\Delta(\mathbf{m})$. When `"pair"` indicated, the last two lines in the above are different as $20,20,20,200 \oplus 11,11,02,011$

etc.

```
[10] os_md.sproot("31,31,22,211","pairs"|dviout=1,sym=1);
[11] os_md.sproot("31,31,22,211","pairs"|dviout=1,sym=2);
```

$$\begin{aligned}
31,31,22,211 &= 20,11,11,110 \oplus 11,20,11,101\\
&= 10,10,10,010 \oplus 21,21,12,201\\
&= 2(10,10,10,100) \oplus 11,11,02,011\\
31,31,22,211 &= 20,11,11,110 \oplus 11,20,11,101\\
&= 11,20,11,110 \oplus 20,11,11,101\\
&= 10,10,10,010 \oplus 21,21,12,201\\
&= 2(10,10,10,100) \oplus 11,11,02,011
\end{aligned}$$

In [O2, Chapter 8] and [O3, Chapter 7] the correspondence between the spectral type $\mathbf{m} = (m_{j,\nu})$ and the roots α of a star shaped Kac-Moody root system.

$$\alpha_{\mathbf{m}} = n\alpha_o + \sum_{j\geq 0}\Big(\sum_{\nu>i} m_{j,\nu}\Big)\alpha_{j,i},\quad n = m_{j,1} + m_{j,2} + \cdots \quad \text{(which do not depend on } j\text{)}$$

$$\texttt{"11,11,11"} \leftrightarrow \alpha = 2\alpha_0 + \alpha_{0,1} + \alpha_{1,1} + \alpha_{2,1}$$

$$\texttt{"21,21,21,21"} \leftrightarrow \alpha = 2\alpha_0 + \alpha_{0,1} + \alpha_{1,1} + \alpha_{2,1} + \alpha_{3,1}$$

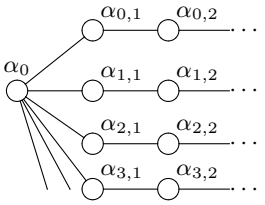
$$\texttt{"31,31,22,211"} \leftrightarrow \alpha = 4\alpha_0 + \alpha_{0,1} + \alpha_{1,1} + 2\alpha_{2,1} + 2\alpha_{3,1} + \alpha_{3,2}$$

$$(\alpha|\alpha) = 2 \quad (\alpha \in \Pi := \{\alpha_0,\, \alpha_{j,\nu} \mid j \geq 0,\, \nu > 0\})$$

$$(\alpha_0|\alpha_{j,\nu}) = -\delta_{\nu,1}$$

$$(\alpha_{i,\mu}|\alpha_{j,\nu}) = \begin{cases} 0 & (i \neq j \quad \text{or} \quad |\mu-\nu| > 1)\\ -1 & (i = j \quad \text{and} \quad |\mu-\nu| = 1)\end{cases}$$

$$s_\alpha \,:\, x \mapsto x - (x|\alpha)\alpha \qquad (\alpha \in \Pi)$$



In the case of the spectral type `"11,11,11"` of Gauss hypergeometric equation we have

$$\begin{aligned}
\alpha_{\mathbf{m}} = 2\alpha_0 + \alpha_{0,1} + \alpha_{1,1} + \alpha_{1,2} &= s_{\alpha_0}s_{\alpha_{2,1}}s_{\alpha_{1,1}}s_{\alpha_{0,1}}(\alpha_0) = w_{\mathbf{m}}^{-1}(\alpha_0),\\
w_{\mathbf{m}} &= s_{\alpha_{0,1}}s_{\alpha_{1,1}}s_{\alpha_{2,1}}s_{\alpha_0},\\
\Delta(\mathbf{m}) &= \{\alpha_0 + \alpha_{0,1},\, \alpha_0 + \alpha_{1,1},\, \alpha_0 + \alpha_{2,1},\, \alpha_0\},\\
[\Delta(\mathbf{m})] &:= \{(\alpha|\alpha_{\mathbf{m}}) \mid \alpha \in \Delta(\mathbf{m})\} = \{1,1,1,1\}\\
&\leftrightarrow 4 = 1+1+1+1 : 1^4
\end{aligned}$$

The `height` equals $2+1+1+1 = 5$. In the rigid case $[\Delta(\mathbf{m})]$ is the partition of the natural number `height` -1 and in this case it is $4 = 1+1+1+1$. Moreover the second and third numbers of the 3rd element of the list

```
[[1,2,1],[1,1,1],[1,0,1],[1,0,0]]
```

given by `base` mean

$$w_{\mathbf{m}}^{-1} = s_{\alpha_{2,1}}s_{\alpha_{1,1}}s_{\alpha_{0,1}}s_{\alpha_0}$$

and the first numbers show the differences of `height` under the simple reflections.

In general if $w_{\mathbf{m}} = s_{\alpha_{i_0}}s_{\alpha_{i_1}}\cdots s_{\alpha_{i_K}}$, we have

$$\Delta(\mathbf{m}) = \{s_{\alpha_{i_0}}\cdots s_{\alpha_{i_{\nu-1}}}(\alpha_{i_\nu}) \mid \nu = 0,\ldots,K\}$$

and moreover if $\mathbf{m}$ is rigid, $w_{\mathbf{m}}\alpha_{\mathbf{m}} = \alpha_0$. Here $w_{\gg}$ is the unique element with the minimal length ($= K$) of the Weyl group (which is generated by simple reflections s_α ($\alpha \in \Pi$)). Mote that $m_{j,\nu}$ is normalized by the condition $m_{j,1} \geq m_{j,2} \geq \cdots$ and $i_0 = 0$.

```
/* Get all rhe decompositions of Type 2 and Type 3 of spectral types with rank 8
 which have more than three singular points  */
[12] Rank=8; /* give rank */
[13] G=os_md.spgen(Rank|eq=1,str=1,pt=[4,100]); /* get spectral types */
[14] for(T=G;T!=[];T=cdr(T))
       if((os_md.sproot(car(T),"pairs"|only=6))!=[]) /* only type 2, 3 */
         os_md.sproot(os_md.s2sp(car(T)|std=-1),  /* in standard order */
         pairs"|only=6,dviout=1);
```

46. `spbasic(`k`,`d`|str=1)`
:: Returns the list of spectral types with rank d whose index of rigidity equals k
- k : 0 or a negative integer
- $d = 0$ means the list without the restriction of the rank
  The required spectral types correspond to $\{m_{j,\nu}\}$ in the identity

$$\Big(\sum_{j=0}^{p}(\operatorname{ord}\mathbf{m} - m_{j,1}) - 2\cdot\operatorname{ord}\mathbf{m}\Big)\operatorname{ord}\mathbf{m} + \sum_{j=0}^{p}\Big(\sum_{\nu=1}^{n_j}(m_{j,1} - m_{j,\nu})m_{j,\nu}\Big) = -\operatorname{idx}\mathbf{m}$$

  with the condition $m_{j,1} \geq m_{j,2} \geq \cdots$, $\mathbf{m}_1 \geq \mathbf{m}_2 \geq \cdots$ and the first terms of the left hand side of this identity is non-negative.

The list is arranged in the lexicographic order from the larger ones. Here we note

$$0 \leq \Big(\sum_{j=0}^{p}(\operatorname{ord}\mathbf{m} - m_{j,1}) - 2\operatorname{ord}\mathbf{m}\Big)\operatorname{ord}\mathbf{m} \leq |\operatorname{idx}\mathbf{m}|$$

If $\operatorname{ord}\mathbf{m} > |\operatorname{idx}\mathbf{m}|$, then $\sum_{j=0}^{p}(\operatorname{ord}\mathbf{m} - m_{j,1}) = 2\cdot\operatorname{ord}\mathbf{m}$.
In particular

$$p+1 \leq \tfrac{1}{2}|\operatorname{idx}\mathbf{m}| + 4, \quad \operatorname{ord}\mathbf{m} \leq 3|\operatorname{idx}\mathbf{m}| + 6, \quad \operatorname{ord}\mathbf{m} \leq |\operatorname{idx}\mathbf{m}| + 2 \quad (p+1 > 3).$$

Moreover if $\operatorname{ord}\mathbf{m} > |\operatorname{idx}\mathbf{m}| + 2$, then $p = 2$ and $m_{0,1} + m_{1,1} + m_{2,1} = \operatorname{ord}\mathbf{m}$.

rigidity index	0	−2	−4	−6	−8	−10	−12	−14	−16	−18	−20	−22
time(sec)			0.02	0.03	0.11	0.22	0.64	1.13	2.25	4.03	7.50	12.92
total number	4	13	37	69	113	198	291	415	647	883	1186	1682
3 singular pts.	3	9	25	46	73	127	182	249	395	521	680	963
4 singular pts.	1	3	9	17	29	50	76	115	172	243	345	478

```
[0] os_md.spbasic(-2,0);
[[[1,1],[1,1],[1,1],[1,1],[1,1]],[[2,1],[2,1],[1,1,1],[1,1,1]],...
[1] tstart$os_md.mycat(length(os_md.spbasic(-10,0)))$tstop$
198
[2] 0.1875sec + gc : 0.03125sec(0.219sec)
```

The time for the calculation to get the list is given under Let's Note CF-SZ5 (i7-6500U).

47. `sp2grs(`m`,`a`,`ℓ`|mat=1)`
:: Generates generalized Riemann scheme with a given spectral type
- The inverse transformation is given by `chkspt()` with the option `opt="sp"`.

- The $j+1$-th spectral parameter in the $i+1$-th singular point is aij (resp. $a[i]j$) if a is a variable (resp. a list), where a or $a[i]$ may be a string.
- When ℓ is a list $[\ell_1, \ell_2]$, ℓ_2-th exponent of ℓ_1-th singular point is determined by Fuchs relation. If the exponent is zero, another exponent is arranged.
- When ℓ is a positive integer, aij is changed into $aij+k$ or $a[i]j+k$. If the above indication is also necessary, we indicate $\ell = [\ell_1, \ell_2, k]$.
- When ℓ is a negative integer $-1-k$, some exponents are set to 0 in a standard way together with the above k shift of indices.
- `mat=1` means the Schleginger type (which has a different Fuchs relation)

```
[0] os_md.sp2grs([[1,1],[1,1],[1,1]],a,0);
[[[1,a00],[1,a01]],[[1,a10],[1,a11]],[[1,a20],[1,a21]]]
[1] M=os_md.sp2grs("111,111,21",[a,b,c],[3,2]);
[[[1,a0],[1,a1],[1,a2]],[[1,b0],[1,b1],[1,b2]],[[2,c0],
[1,-b1-b2-a2-a1-2*c0-b0-a0+2]]]
[2] os_md.sp2grs([[1,1,1],[1,1,1],[2,1]],[a,b,c],[3,2,-2]);
[[[1,a1],[1,a2],[1,a3]],[[1,b1],[1,b2],[1,0]],[[2,0],
[1,-b1-b2-a3-a2-a1+2]]]
[3] subst(M,b0,0,b1,1-b1,b2,1-b2,c0,0);
[[[1,a0],[1,a1],[1,a2]],[[1,0],[1,-b1+1],[1,-b2+1]],[[2,0],
[1,b1+b2-a2-a1-a0]]]
[4] os_md.sp2grs([[1,1],[1,1],[1,1]],a,1);
[[[1,a01],[1,a02]],[[1,a11],[1,a12]],[[1,a21],[1,a22]]]
[5] os_md.ssubgrs(M,"110,110,11");
-b2-a2-c0+2
[6] os_md.ssubgrs(M,[[1,1,0],[1,1,0],[1,1]]);
-b2-a2-c0+2
```

48. `ssubgrs` (m, ℓ)
49. `mcgrs(`m`,[`$r_1, r_2, \ldots, r_n$`]|mat=1,slm=[[`k_1`,`k_2`,...],`m'`])`
:: Applies middle convolutions and additions successively to a generalized Riemann scheme or a sum of residue matrices
    - If r_j is a scalar, it means a middle convolution mc_{r_j}. If r_j is a list $[r_{j,1}, \ldots, r_{j,n}]$, it means additions. If r_j is a list $[r_{j,0}, r_{j,1}, \ldots, r_{j,n}]$, it means that a middle convolution $mc_{r_{j,0}}$ is applied first and then the additions corresponding to $[r_{j,1}, \ldots, r_{j,n}]$ are applied. Firstly r_n is applied to p and then ar_{n-1} to its result,..., and r_1 is lastly applied.
    - We indicate `mat=1` if the Riemann scheme is of Schlesinger type.
    If `sm1=` is indicated, the transformation of the spectral type of sum m' of $k_1, k_2, \ldots$-th residue matrices. This enables us to know the semi-local monodromy and the monodromy of the corresponding confluent singular point (cf. [O7]).

```
[1] M0=[[1,0]]$M=[M0,M0,M0,M0]$S=[m,[a,b,c]]$
[2] os_md.mcgrs(M,S);
[[[2,-m+1],[1,-a-b-c-m]],[[2,0],[1,a+m]],[[2,0],[1,b+m]],[[2,0],[1,c+m]]]
[3] os_md.mcgrs(M,S|mat=1);
[[[2,-m],[1,-a-b-c-m]],[[2,0],[1,a+m]],[[2,0],[1,b+m]],[[2,0],[1,c+m]]]
[4] os_md.mcgrs(M,S|mat=1,slm=[[0,1],M0]);
[[[[2,-m],[1,-a-b-c-m]],[[2,0],[1,a+m]],[[2,0],[1,b+m]],[[2,0],[1,c+m]]],
```

`[[1,0],[1,-m],[1,-b-c-m]]]`

50. `mcop(`p`,[`$r_1, r_2, \ldots, r_n$`],[`$x, x_1, x_2, \ldots$`])`
:: Applies middle convolutions and additions successively to a (partial) differential operator p
- `[`$x, x_1, x_2, \ldots$`]` : x is the variable, $x_1, x_2, \ldots$ are singular points.
- `[`$r_1, r_2, \ldots$`]` : $r_j = [r_{j,0}, r_{j,1}, \ldots, r_{j,\nu}, \ldots]$ is the j-th operation of a middle convolution and additions at x_ν
  - Apply the middle convolution $mc_{r_{j,0}}$ and after that apply the additions at $x = x_\nu$ (corresponding to the gage transformation of a function u to $(x - x_\nu)^{r_{j,\nu}} u$ $(\nu = 1, 2, \ldots)$.
  - Apply the transformation described by r_j to p in the order r_1, $r_2, \ldots$.
  - `d`x : represents $\frac{d}{dx}$ (or $\frac{\partial}{\partial x}$).

```
[0] R=os_md.getbygrs("21,21,21,21","construct")$
[1] R[0][1];
[[[1,b+c+d+a1+2*a0-2]],[[1,0]],[[1,0]],[[1,0]]]
[2] S=os_md.lsol(os_md.getbygrs("21,21,21,21","Fuchs"),a1);
-b-c-d-2*a0+2
[3] R=subst(R,a1,S,a0,a);
[[0,[[[1,0]],[[1,0]],[[1,0]],[[1,0]]]],
[[0,a+b-1,a+c-1,a+d-1],[[[1,-3*a-b-c-d+3]],[[1,a+b-1]],[[1,a+c-1]],[[1,a+d-1]]]],
[[-a+1,0,0,0],[[[2,a],[1,-2*a-b-c-d+2]],[[2,0],[1,b]],[[2,0],[1,c]],[[2,0],[1,d]]]]]
[4] RR=[R[1][0],R[2][0]];
[[0,a+b-1,a+c-1,a+d-1],[-a+1,0,0,0]]
[5] os_md.mcop(dy,RR,[x,0,1,y]);
((-x+y)*dx-a)*dy+(-a-d+1)*dx
[6] os_md.show(@@)$
```

$$-(x-y)\partial_x\partial_y - (a+d-1)\partial_x - a\partial_y$$

51. `redgrs(`m`|mat=1)`
:: Returns 1-step reduction of generalized Riemann scheme of Fuchsian differential equation
Returns [*redsp*, m'].
If the reduction is not possible, it returns non-negative integer (basic, 0 means rigid) or negative number (in the case of the non-existence of the operator)

52. `getbygrs(`m`,`t`|perm=`ℓ`,var=`v`,pt=[`$p_1, \ldots$`],mat=1)` or
`getbygrs(`m`,[`$t, s_1, s_2, \ldots$`]|perm=`ℓ`,var=`v`,pt=[`$p_1, \ldots$`],mat=1)`
:: Analyzes Fuchsian ODE defined by generalized Riemann scheme (GRS) (GRS may be given in a short form or a spectral type)
- `mat=1` : Schleginger form
- `getbygrs(0,0)` : parameter is shown (for help)
- The number of the elements of m which we put $p+1$ corresponds to the number of the singular points denoted by $x_0 = \infty, x_1 = 0, x_2 = 1, x_3, \ldots, x_p$.
- The generalized exponents of a Fuchsian equation at $x = x_j$ are $\{[\lambda_0]_{(n_0)}, \ldots, [\lambda_k]_{(n_k)}\}$ if the exponents are $\{\lambda_j + \nu;\ 0 \le \nu < n_j,\ j = 0, \ldots, k\}$ including their multiplicities and if any difference of the elements $\{\lambda_0, \ldots, \lambda_k\}$ is not an integer (otherwise we will not give the definition here but for example when all λ_j are same, then the sizes of Jordan blocks of the corresponding local monodromy matrix correspond to the dual partition of $n = n_0 + \cdots + n_k$ $x = x_j$ is semisimple (diagonalizable), namely, there exist local solutions $(x - x_j)^{\lambda_j + \nu}\phi_{j,\nu}(x)$ $(0 \le \nu < n_\nu)$ ($\phi_{j,\nu}(x)$ are holomorphic at $x = x_j$ and $\phi_{j,\nu}(x_j) = 1$).
- Generalized Riemann scheme (GRS) is the table of generalized exponents at the $p+1$ singular

points (cf. [O2, Chapter 5], [O3, Definition 4.6]) :

$$\left\{\begin{array}{cc} x=\infty & x_j\ (j=1,\dots,p) \\ [m[0][0][1]]_{(m[0][0][0])} & [m[j][0][1]]_{(m[j][0][0])} \\ [m[0][1][1]]_{(m[0][1][0])} & [m[j][1][1]]_{(m[j][1][0])} \\ \vdots & \vdots \\ [m[0][i][1]]_{(m[0][i][0])} & [m[j][i][1]]_{(m[j][i][0])} \\ \vdots & \vdots \end{array}\ ;\ x\right\}$$

- `perm=[[`i,j`]]` : transposition of i-th singular point and j-th singular point
- `perm=[`$j_0,\dots,j_p$`]` : permutation of singular points (x_{j_0} corresponds to ∞).
- `var=`v : return the variable of exponents. (cf. the second parameter of `sp2grs()`).
- `pt=[`$p_1,p_2,\dots$`]` : the singular points are ∞, p_1, p_2, ...
- One of the exponent of Generalize Riemann scheme m may not be indicated by `"?"`. The is is determined by Fuchs relation.
- When m is a spectral types (a tuple of partitions), it is replaced to the corresponding GRS. If $s=$ `"general"` is indicated in the above case, the exponents are general. Otherwise some exponents are 0 and the GRS may be Okubo type or a close one.

Put $R=$ `getbygrs(`m`,`t`)`.
The option parameter s is a string of a list of strings. `"TeX"`, `"dviout"`, `"keep"` can be indicated in the following cases.

- $t=$`"reduction"` or 0 : reduction to the basic equation
  `"top0"` can be indicated when $s=$`"TeX"` is indicated.
  $R[i][0]$: the i-th reduction are as follows ($i=1,2,\dots$)
  the additions at x_ν with the parameters $R[i][0][\nu]$ for $\nu=1,\dots,p$ and then the middle convolution with the parameter $R[i][0][0]$.
  $R[i][1]$: the GRS obtained by the above reduction
  $R[0][1]$: the original GRS
- $t=$`"construct"` or 1 : construction of the equation and integral representation of the solution
  $R[0][1]$: GRS of the basic equation obtained by the reduction (we denote it by $\mathbf{m}'$).
  By $\mathrm{RAd}\left(\prod_{\nu=1}^{p}(x-x_\nu)^{R[i][0][\nu]}\right)\circ mc_{R[i][0][0]}$ the GRS is changed from $R[i-1][1]$ to $R[i][1]$ ($i=1,2,\dots$) and the corresponding transformation of $\mathbf{m}'$ into $\mathbf{m}$ is given.
  $s=$`"short"` can be indicated
  $s=$`"TeX"` means the integral representation of the solution (in TEX) corresponding to a multiplicity one exponent of the local monodromy at $x=x_1$ ($x_1=0$ in default).
- $t=$`"connection"` or 2 : the connection coefficient
  $s=$`"simplify"` can be indicated
  When the number of singular points is $p+1$, they are $x_0=\infty, x_1=0, x_2=1, x_3, x_4,\dots,x_p$ by default. When the GRS is as given and the multiplicities of the last exponents at $x=0$ and $x=1$ are 1, the connection coefficient of the corresponding normalized local solution at $x=0$ to that of $x=1$ is given by

$$c_B\frac{\prod_\nu\Gamma(R[0][\nu])}{\prod_\nu\Gamma(R[1][\nu])}\prod_{j\geq 3}(1-x_j^{-1})^{R[2][j]}$$

  Here c_B is the corresponding connection coefficient of the basic equation. When the equation is rigid (it has no accessory parameter), then $c_B=0$. Note that the terms $(1-x_j^{-1})^{R[2][j]}$ do not exist when the number of the singular points is 3.
- $t=$`"operator"` or 3 : get the equation
  $s=$`"simplify"` may be indicated

`x` is the variable x and `dx` means $\frac{d}{dx}$ in the result
The accessory parameters are expressed by `ri_j` and they correspond to coefficient of certain $x^j \frac{d^i}{dx^i}$.
This calculation is heavier than other calculations and the result may be long.

- t =`"series"` or 4 : The local solution in the power series corresponding to the last exponent at $x = 0$. The multiplicity of the exponent should be 1.
  $R = [R_0, R_1, R_2]$
  $R_0 = [[R_{00}, [R_{01}, T_1], [R_{02}, T_2], \ldots], [P_1, Q_1], [P_2, Q_2], \ldots]$
  $R_1 = [[R_{100}, P_{101}, P_{102}, \ldots], [R_{110}, P_{111}, P_{112}, \ldots], \ldots]$
  $R_2 = [[R_{200}, P_{201}, P_{202}, \ldots], [R_{210}, P_{211}, P_{212}, \ldots], \ldots]$
  $0 \le P_1 < P_2 < P_3 < \cdots$

$$\sum_{n_{P_1}, n_{P_2}, \ldots} C_{n_0} x^{R_{00}+n_0} \prod_{j\ge 0,\ Pj\ne 0} \Big(\frac{x}{x_{Q_j}}\Big)^{n_{Pj}}$$
$$\cdot \prod_{j\ge 1} \Big(1 - \frac{x}{x_{T_j}}\Big)^{R_{0j}} \frac{\prod_{j\ge 0} (R_{1j0})_{n_{P_{1j1}}+n_{P_{1j2}}+\cdots}}{\prod_{j\ge 0} (R_{2j0})_{n_{P_{2j1}}+n_{P_{2j2}}+\cdots}},$$
$$x_1 = x_2 = 1,$$
$$P_1 \ne 0 \Rightarrow n_0 = 0 \ \text{ and } \ C_0 = 1.$$

- t = `"TeX"` or 5 : give GRS in LaTeX
  s = `"top0"` : indicate that $x = \infty$ is the last in GRS
- t = `"Fuchs"` or 6 : get Fuchs relation
- t = `"All"` or 7 : only valid when `"dviout"` is indicated in s. Several related results are displayed using `dviout()`. If `"irreducible"` is indicated in s, the condition for the irreducibility is omitted. If `"operator"` is indicated in s, the differential operator is shown.
- t = `"basic"` or 8 : get the corresponding basic GRS
  s =`"short"` may be indicated
- t = `""` or 9 : get GRS
  s =`"short"` indicates the short form
- t = `"irreducible"` or 10 : get the condition for the irreducibility
  s = `"simplify` may be indicated
  The condition for the irreducibility equals that the any value of the linear form in the resulting list is not an integer (with the condition for the irreducibility of the resulting basic equation if the equation is not rigid).
- t = `"recurrence"` or 11 : adjacent relation in three terms This is only valid when the number of singular points are three (0, 1, ∞) and the multiplicities of the last exponent of the three singularities are 1.
  The result is the shift of the local normalized solution with the last exponent at the origin.

s is a string or a list of string. The string are

- `"simplify"` : simplify the result using the Fuchs relation
- `"TeX"` : the result is given in LaTeX
- `"dviout"` : the result is displayed by `dviout`
  In this case we can indicate a title by the option parameter `title=`
- `"keep"` : same as above but the result is not displayed (only a file in TeXis created).
- `"short"` : GRS is a short format
- `"general"` : exponents are fully generic according to the spectral type
- `"x1"` : the singular points are ∞, x_1, x_2, ... (only valid for GRS in TeXand the differential operator)

- `"x2"` : the singular points are ∞, 0, $x_2, \ldots$. (only valid for GRS in TEXand the differential operator)
- `"top0"` : ∞ is the last in GRS in TEX.
- `"sht"` : the suffices of exponents start from 1 (0 by default) automatically generated from the given spectral type

```
[1] M=os_md.sp2grs([[1,1,1],[1,1,1],[2,1]],[a,b,c],[3,2]);
[[[1,a0],[1,a1],[1,a2]],[[1,b0],[1,b1],[1,b2]],[[2,c0],
[1,-2*c0-b2-b1-b0-a2-a1-a0+2]]]
[2] os_md.getbygrs(M,"reduction");
[
 [0,
  [ [[1,a0],[1,a1],[1,a2]],
    [[1,b0],[1,b1],[1,b2]],
    [[2,c0],[1,-2*c0-b2-b1-b0-a2-a1-a0+2]]
  ]
 ],
 [[ 0 0 0 ],
  [ [[0,-c0-b0-a0+2],[1,a1-a0+1],[1,a2-a0+1]],
    [[0,0],[1,c0+b1+a0-1],[1,c0+b2+a0-1]],
    [[1,0],[1,-2*c0-b2-b1-a2-a1+1]]
  ]
 ],
 [[ 1 1 0 ],
  [ [[0,-c0-b0-a1+2],[0,-c0-b1-a1+2],[1,a2-a1+1]],
    [[0,a1-a0],[0,0],[1,c0+b2+a1-1]],
    [[0,0],[1,-c0-b2-a2]]
  ]
 ]
]
[3] os_md.getbygrs(M,"construct");
[
 [[0,
  [ [[1,0]],
    [[1,0]],
    [[1,0]]
  ]
 ]
 [[0,c0+b2+a1-1,-c0-b2-a2]
  [ [[1,a2-a1+1]],
    [[1,c0+b2+a1-1]],
    [[1,-c0-b2-a2]]
  ]
 ],
 [[-c0-b1-a1+1,c0+b1+a0-1,0],
```

```
  [ [[1,a1-a0+1],[1,a2-a0+1]],
    [[1,c0+b1+a0-1],[1,c0+b2+a0-1]],
    [[1,0],[1,-2*c0-b2-b1-a2-a1+1]]
  ]
 ],
 [[-c0-b0-a0+1,b0,c0],
  [ [[1,a0],[1,a1],[1,a2]],
    [[1,b0],[1,b1],[1,b2]],
    [[2,c0],[1,-2*c0-b2-b1-b0-a2-a1-a0+2]]
  ]
 ]
]
[4] os_md.getbygrs(M,"connection");
[[3*c0+b2+b1+b0+a2+a1+a0-2,b2-b1+1,b2-b0+1],
 [c0+b2+a2,c0+b2+a1,c0+b2+a0],
 [ 0 2*c0+b2+b1+b0+a1+a0-2 -b2-a2 ]]
[5] os_md.getbygrs(M,["construct","TeX"])$
x^{b_{0}}(1-x)^{c_0}
\int_c^x(x-s_0)^{c_0+b_0+a_0}ds_0
s_0^{c_0+b_1+a_0-1}
\int_c^{s_0}(s_0-s_1)^{c_0+b_1+a_1}
s_1^{c_0+b_2+a_1-1}
(1-s_1)^{-c_0-b_2-a_2}ds_1
[6] M0=subst(M,b2,0,c0,0,b1,1-b1,b0,1-b0);
[
 [[1,a0],[1,a1],[1,a2]],
 [[1,-b0+1],[1,-b1+1],[1,0]],
 [[2,0],[1,b1+b0-a2-a1-a0]]
]
[7] os_md.getbygrs(M0,"connection");
[[b1,-b1-b0+a2+a1+a0,b0],
 [a2,a1,a0],
 [ 0 -b1-b0+a1+a0 -a2 ]]
[8] P=os_md.getbygrs(M0,"operator");
(x^3-x^2)*dx^3+((a1+a0+a2+3)*x^2+(-b1-b0-1)*x)*dx^2+
(((a0+a2+1)*a1+(a2+1)*a0+a2+1)*x-b0*b1)*dx+a2*a0*a1
[9] os_md.expat(P,x,"?");
[[0,[-b1+1,-b0+1,0]],[1,[b1+b0-a1-a0-a2,1,0]],[infty,[a1,a0,a2]]]
[10] os_md.getbygrs(M0,"irreducible");
[b1-a1,b1-a2,a1,a2,b0-a0,b0-a2,b0-a1,a0,b1-a0]
[11] M1=os_md.sp2grs([[1,1,1],[1,1,1],[2,1]],[a,b,c],[]);
[[[1,a0],[1,a1],[1,a2]],[[1,b0],[1,b1],[1,b2]],[[2,c0],[1,c1]]]
[12] M2 = subst(M1,b0,1-b0,b1,1-b1,b2,0,c0,0,c1,-c1);
[[[1,a0],[1,a1],[1,a2]],[[1,-b0+1],[1,-b1+1],[1,0]],[[2,0],[1,-c1]]]
```

```
[13] os_md.getbygrs(M2,"connection");
[[c1,b1,b0],[c1+b1+b0-a1-a0,a1,a0],[ 0 -b1-b0+a1+a0 -c1-b1-b0+a1+a0 ]]
[14] os_md.getbygrs(M2,["connection","simplify"]);
[[c1,b1,b0],[a2,a1,a0],[ 0 c1-a2 -a2 ]]
[15] os_md.getbygrs(M2, "Fuchs");
-c1-b1-b0+a2+a1+a0
[16] os_md.getbygrs(M2, "TeX")$
P\begin{Bmatrix}
x=\infty & 0 & 1\\
  a_0 &   -b_0+ 1 & [0]_{(2)} & \!\!;x\\
  a_1 &   -b_1+ 1 &  -c_1\\
  a_2 & 0 & \\
\end{Bmatrix}
[17] os_md.getbygrs(M2,["connection","simplify","TeX"])$
c(0:0 \rightsquigarrow 1: -c_1)=\frac{
  \Gamma(c_1)
  \Gamma(b_1)
  \Gamma(b_0)
 }{
  \Gamma(a_2)
  \Gamma(a_1)
  \Gamma(a_0)
 }
[18] os_md.getbygrs(M2, ["Fuchs","TeX"]);
 -c_1-b_1-b_0+a_2+a_1+a_0
[19] os_md.getbygrs([[1,1],[1,1],[1,1]],"operator");
(-x^2+x)*dx^2+((c1+b0-2)*x-b0+1)*dx+a0*c1+a0*b0+a0^2-a0
[20] HO=[[1,1,1],[2,1],[2,1],[2,1]]$
[21] os_md.getbygrs(HO,["","short"]);
[[a0,a1,a2],[[2,0],b1],[[2,0],c1],[[2,0],d1]]
[22] os_md.getbygrs(HO,"TeX");
P\begin{Bmatrix}
x=\infty & 0 & 1& x_3\\
 a_0 & [0]_{(2)} & [0]_{(2)} & [0]_{(2)}&\!\!;x\\
 a_{1} & b_1 & c_1 & d_1\\
 a_{2} &  &  & \\
\end{Bmatrix}
[23] os_md.getbygrs(HO,["basic","short"]);
[[b1+a1,b1+a2],[-b1-a0+1,0],[0,c1+a0-1],[0,a0+d1-1]]
[24] os_md.getbygrs(HO,["","short","general"]);
[[a0,a1,a2],[[2,b0],b1],[[2,c0],c1],[[2,d0],d1]]
[25] os_md.getbygrs(HO,["basic","TeX"]);
P\begin{Bmatrix}
x=\infty & 0 & 1& x_3\\
```

```
 b_1+a_1 & -b_1-a_0+1 & 0 & 0&\!\!;x\\
 b_1+a_2 & 0 & c_1+a_0-1 & a_0+d_1-1\\
\end{Bmatrix}
[26] os_md.getbygrs(H0,"Fuchs");
b1+c1+a1+a0+d1+a2-3
[27] P=os_md.getbygrs(H0,"operator");
(-x^3+(x_3+1)*x^2-x_3*x)*dx^3+((-a1-a0-a2-3)*x^2+(-x_3*b1+(-x_3+1)*c1+a1+a0+a2
+4*x_3+1)*x+x_3*b1-2*x_3)*dx^2+(((-a0-a2-1)*a1+(-a2-1)*a0-a2-1)*x-b1^2+((x_3-1)*c1
-a1-a0-a2-2*x_3+2)*b1+(-x_3+1)*c1+a1+a0+a2+2*x_3-r0_0-1)*dx-a2*a0*a1
[28] os_md.expat(P,x,"?");
[[0,[b1,1,0]],[1,[c1,1,0]],[x_3,[-b1-c1-a1-a0-a2+3,1,0]],
[infty,[a1,a0,a2]]]
```

In the above example ([1]–[5]) we examine the generalized Riemann scheme

$$\mathcal{R} = P\left\{\begin{array}{cccc} x=\infty & 0 & 1 & \\ a_0 & b_0 & [c_0]_{(2)} & ;\ x \\ a_1 & b_1 & c_1 & \\ a_2 & b_2 & & \end{array}\right\}$$
$$\text{with}\quad c_1 = -a_0-a_1-a_2-b_0-b_1-b_2-2c_0+2$$

whose spectral type is $\mathbf{m} = 111, 111, 21$. Then `getbygrs`$(m,1)$ shows that the equation $Pu = 0$ with the GRS $\mathcal{R}$ is given by

$$\begin{aligned} P = {} & \mathrm{RAd}\big(x^{b_0}(x-1)^{c_0}\big) \circ mc_{-a_0-b_0-c_0+1} \circ \mathrm{RAd}\big(x^{a_0+b_1+c_0-1}\big) \\ & \circ mc_{-a_1-b_1-c_0+1} \circ \mathrm{RAd}\big(x^{a_1+b_2+c_0-1}(x-1)^{-a_2-b_2-c_0}\big)\partial_x \end{aligned}$$

and the generalized Riemann scheme changes as follows

$$\begin{aligned} & P\left\{\begin{array}{ccc} x=\infty & 0 & 1 \\ 0 & 0 & 0\ ;\ x \end{array}\right\} \\ & \quad \to \mathrm{RAd}\big(x^{c_0+a_1+b_2-1}(x-1)^{-c_0-b_2-a_2}\big) \\ & P\left\{\begin{array}{ccc} x=\infty & 0 & 1 \\ a_2-a_1+1 & c_0+b_2+a_1-1 & -c_0-b_2-a_2\ ;\ x \end{array}\right\} \\ & \quad \to \mathrm{RAd}\big(x^{a_0+b_1+c_0-1}\big) \circ mc_{-a_1-b_1-c_0+1} \\ & P\left\{\begin{array}{ccc} x=\infty & 0 & 1 \\ a_1-a_0+1 & c_0+b_1+a_0-1 & 0 \\ a_2-a_0+1 & c_0+b_2+a_0-1 & -2c_0-b_2-b_1-a_2-a_1+1\ ;\ x \end{array}\right\} \\ & \quad \to \mathrm{RAd}\big(x^{b_0}(x-1)^{c_0}\big) \circ mc_{-a_0-b_0-c_0+1} \\ & P\left\{\begin{array}{cccc} x=\infty & 0 & 1 & \\ a_0 & b_0 & [c_0]_{(2)} & \\ a_1 & b_1 & c_1 := -a_0-a_1-a_2-b_0-b_1-b_2-2c_0+2\ ;\ x \\ a_2 & b_2 & & \end{array}\right\}. \end{aligned}$$

Hence an integral representation of the solution is

$$\begin{aligned} u(x) = {} & x^{b_0}(1-x)^{c_0} \int_c^x \int_c^{s_0} (x-s_0)^{a_0+b_0+c_0} s_0^{a_0+b_1+c_0-1} \\ & \cdot (s_0-s_1)^{a_1+b_1+c_0} s_1^{a_1+b_2+c_0-1} (1-s_1)^{-a_2-b_2-c_0} ds_1 ds_0. \end{aligned}$$

When $c=0$ or $c=1$ or $c=\infty$, $u(x)$ is a local solution at $x=0$ or $x=1$ or $x=\infty$ corresponding to the exponent b_2 or c_1 or a_2, respectively. It corresponds to the local solution for *the last exponent with free multiplicity* at each singular point.
By `getbygrs(m,"connection")` we get the connection coefficient

$$\begin{aligned} c(0:b_2\rightsquigarrow 1:c_1) &= \frac{\Gamma(a_0+a_1+a_2+b_0+b_1+b_2+3c_0-2)\prod_{\nu=0}^{1}\Gamma(b_2-b_\nu+1)}{\prod_{\nu=0}^{2}\Gamma(a_\nu+b_2+c_0)} \\ &= \frac{\Gamma(c_0-c_1)\prod_{\nu=0}^{1}\Gamma(b_2-b_\nu+1)}{\prod_{\nu=0}^{2}\Gamma(a_\nu+b_2+c_0)}, \end{aligned}$$

which corresponds to the local solution

$$x^{b_2}(1-x)^{c_0}{}_3F_2(a_0+b_2+c_0, a_1+b_2+c_0, a_2+b_2+c_0, 1-b_0+b_2, 1-b_1+b_2; x).$$

The calculation

```
[29] G=[[a,b],[1-c,0],[c-a-b,0]];
[
 [a,     b],
 [-c+1,  0],
 [-a-b+c,0]
]
[30] os_md.getbygrs(G,"operator");
(-x^2+x)*dx^2+((-a-b-1)*x+c)*dx-b*a
[31] os_md.getbygrs(G,"connection");
[[-a-b+c,c],[-a+c,-b+c],[ 0 -b -b ]]
[32] os_md.getbygrs(G,["construct","TeX"])$
x^{-c+1}(1-x)^{-a-b+c}\int_c^x(x-s_0)^{-b+1}
s_0^{-b+c-1}(1-s_0)^{a-c}ds_0
[33] os_md.getbygrs(G,"irreducible")
[b,a,b-c,a-c]
```

shows Gauss summation formula (cf. [31])

$$F(a,b,c;1) = \frac{\Gamma(c-a-b)\Gamma(c)}{\Gamma(c-a)\Gamma(c-b)}$$

for the Gauss hypergeometric series

$$F(a,b,c;x) = \sum_{k=0}^{\infty} \frac{(a)(a+1)\cdots(a+k-1)\cdot b(b+1)\cdots(b+k-1)}{(c)(c+1)\cdots(c+k-1)k!}x^k,$$

which is a solution of the hypergeometric equation (cf. [30])

$$x(1-x)u'' + \big(c-(a+b+1)x\big)u' - abu = 0$$

corresponding to the Riemann scheme (cf. [29])

$$P\left\{\begin{array}{ccc} x=\infty & 0 & 1 \\ a & 0 & 0 \\ b & 1-c & c-a-b \end{array}\ ;\ x\right\}.$$

The equation is irreducible if and only if (cf. [33])

$$a,\ b,\ a-c,\ b-c \notin \mathbb{Z}.$$

```
[34] os_md.getbygrs([[a1,a2,a3],[0,b1,b2],[[2,0],"?"]],["reduction","dviout","top0"]);
```

gives the following with its source file in LaTeX:

$$P\left\{\begin{matrix} x=0 & 1 & \infty & \\ 0 & [0]_{(2)} & a_1 & ;x \\ b_1 & -b_1-b_2-a_3-a_2-a_1+2 & a_2 & \\ b_2 & & a_3 & \end{matrix}\right\}$$

$$\leftarrow \mathrm{mc}_{-a_1+1}:\ P\left\{\begin{matrix} x=0 & 1 & \infty & \\ b_1+a_1-1 & 0 & a_2-a_1+1 & ;x \\ b_2+a_1-1 & -b_1-b_2-a_3-a_2+1 & a_3-a_1+1 & \end{matrix}\right\}$$

$$\leftarrow \mathrm{Ad}\left(x^{b_1+a_1-1}\right)\mathrm{mc}_{-b_1-a_2+1}:\ P\left\{\begin{matrix} x=0 & 1 & \infty & \\ b_2+a_2-1 & -b_2-a_3 & a_3-a_2+1 & ;x \end{matrix}\right\}$$

$$\leftarrow \mathrm{Ad}\left(x^{b_2+a_2-1}(1-x)^{-b_2-a_3}\right):\ P\left\{\begin{matrix} x=0 & 1 & \infty & \\ 0 & 0 & 0 & ;x \end{matrix}\right\}$$

```
[35] os_md.getbygrs("11,11,11",["All","dviout","operator"]);
```

gives the following with its source file in LaTeX:

Riemann scheme

$$P\left\{\begin{matrix} x=\infty & 0 & 1 & \\ a_0 & 0 & 0 & ;x \\ a_1 & b & c & \end{matrix}\right\}$$

Fuchs condition

$$b+c+a_1+a_0-1$$

Connection formula

$$c(0:b\rightsquigarrow 1:c)=\frac{\Gamma(-c)\Gamma(b+1)}{\Gamma(b+a_0)\Gamma(b+a_1)}$$

Recurrence relation shifting the last exponents at ∞, 0, 1

$$u_{0,0,0}-u_{+1,0,-1}=\frac{(b+1)}{(b+a_0)}u_{0,+1,-1}$$

Integral representation

$$\int_p^x (x-s_0)^{-a_0}s_0^{-c-a_1}(1-s_0)^{-b-a_1}ds_0$$
$$\sim \frac{\Gamma(-a_0+1)\Gamma(b+a_0)}{\Gamma(b+1)}x^b\quad (p=0,\ x\to 0)$$

Series expansion

$$\sum_{n\geq 0}\frac{(b+a_0)_n(b+a_1)_n}{(b+1)_n n!}x^{b+n}$$

Irreducibility $\Leftrightarrow$ any value of the following linear forms $\notin \mathbb{Z}$

$$\begin{matrix} a_0 & a_1 \\ c+a_0 & b+a_0 \end{matrix}$$

which correspond to the decompositions

$$\begin{aligned} 11,11,11 &= 10,10,01 \oplus 01,01,10 \\ &= 10,01,10 \oplus 01,10,01 \\ &= 01,10,10 \oplus 10,01,01 \\ &= 10,10,10 \oplus 01,01,01 \end{aligned}$$

Operator

$$-x(x-1)\partial^2 + ((b+c-2)x - b + 1)\partial - a_0 a_1$$

```
[36] os_md.getbygrs([[[2,e],f,g],[[2,0],a,b],[[2,0],c,d]],["All","top0","dviout"]);
```

gives

Riemann scheme

$$P\left\{\begin{matrix} x=0 & 1 & \infty & \\ [0]_{(2)} & [0]_{(2)} & [e]_{(2)} & ;x \\ a & c & f & \\ b & d & g & \end{matrix}\right\}$$

Fuchs condition

$$a+b+c+d+2e+f+g-3$$

Connection formula

$$c(0:b \rightsquigarrow 1:d) = \frac{\Gamma(-d)\Gamma(c-d)\Gamma(b+1)\Gamma(-a+b+1)}{\Gamma(-d-e+1)\Gamma(b+e)\Gamma(b+c+e+g-1)\Gamma(b+c+e+f-1)}$$

Recurrence relation shifting the last exponents at ∞, 0, 1

$$u_{0,0,0} - u_{+1,0,-1} = \frac{(b+1)(-a+b+1)}{(b+e)(b+c+e+f-1)} u_{0,+1,-1}$$

Integral representation

$$\begin{aligned} &\int_p^x \int_p^{s_0} (x-s_0)^{-e} s_0^{a+e-1} (1-s_0)^{c+e-1} (s_0-s_1)^{b+d+e+g-2} s_1^{b+c+e+f-2} (1-s_1)^{a+d+e+f-2} ds_1 ds_0 \\ &\sim \frac{\Gamma(-e+1)\Gamma(b+e)\Gamma(b+d+e+g-1)\Gamma(b+c+e+f-1)}{\Gamma(b+1)\Gamma(-a+b+1)} x^b \quad (p=0,\ x \to 0) \end{aligned}$$

Series expansion

$$\sum_{n_1 \geq 0,\ n_2 \geq 0} \frac{(-c-e+1)_{n_2} (b+c+e+g-1)_{n_1} (b+c+e+f-1)_{n_1} (b+e)_{n_1+n_2}}{(-a+b+1)_{n_1} (b+1)_{n_1+n_2} n_1! n_2!} x^{b+n_1+n_2}$$

Irreducibility $\Leftrightarrow$ any value of the following linear forms $\notin \mathbb{Z}$

$$\begin{array}{llll} g & f & e & \\ d+e & c+e & b+e & a+e \\ b+c+e+f & a+d+e+f & a+c+e+g & a+c+e+f \end{array}$$

which correspond to the decompositions

$$\begin{aligned}
211, 211, 211 &= 110, 110, 101 \oplus 101, 101, 110 \\
&= 110, 101, 110 \oplus 101, 110, 101 \\
&= 101, 110, 110 \oplus 110, 101, 101 \\
&= 110, 110, 110 \oplus 101, 101, 101 \\
&= 100, 100, 001 \oplus 111, 111, 210 \\
&= 100, 100, 010 \oplus 111, 111, 201 \\
&= 100, 001, 100 \oplus 111, 210, 111 \\
&= 100, 010, 100 \oplus 111, 201, 111 \\
&= 001, 100, 100 \oplus 210, 111, 111 \\
&= 010, 100, 100 \oplus 201, 111, 111 \\
&= 2(100, 100, 100) \oplus 011, 011, 011
\end{aligned}$$

A non-rigid case:

```
[37] os_md.getbygrs("3111,3111,3111",["All","dviout"]);
```

shows

Riemann Scheme

$$P\left\{\begin{array}{ccc} x=\infty & 0 & 1 \\ [a_0]_{(3)} & [0]_{(3)} & [0]_{(3)} \\ a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{array} ; x\right\}$$

Basic Riemann Scheme

$$P\left\{\begin{array}{ccc} x=\infty & 0 & 1 \\ b_3+c_1+a_1+a_0-1 & b_1-b_3 & 0 \\ b_3+c_1+a_2+a_0-1 & b_2-b_3 & c_2-c_1 \\ b_3+c_1+a_3+a_0-1 & 0 & c_3-c_1 \end{array} ; x\right\}$$

$$[[1,1,1],[1,1,1],[1,1,1]]$$

Fuchs condition

$$b_1+b_2+b_3+c_3+c_2+c_1+a_3+a_2+a_1+3a_0-6$$

Connection formula

$$\begin{aligned}
&c(0\,{:}\,b_3 \rightsquigarrow 1\,{:}\,c_3) \\
&= \frac{\Gamma(-c_3)\Gamma(b_3+1)}{\Gamma(-c_3-a_0+1)\Gamma(b_3+a_0)} c_B(0\,{:}\,0 \rightsquigarrow 1\,{:}\,c_3-c_1)
\end{aligned}$$

Integral representation

$$\begin{aligned}
&\int_p^x (x-s_0)^{-a_0} s_0^{b_3+a_0-1} (1-s_0)^{c_1+a_0-1} u_B(s_0) ds_0 \\
&\sim \frac{\Gamma(-a_0+1)\Gamma(b_3+a_0)}{\Gamma(b_3+1)} C_0 x^{b_3} \quad (p=0,\ x\to 0)
\end{aligned}$$

Series expansion

$$\sum_{n_0\geq 0,\ n_1\geq 0} \frac{(-c_1-a_0+1)_{n_1}(b_3+a_0)_{n_0+n_1}}{(b_3+1)_{n_0+n_1}n_1!} C_{n_0} x^{b_3+n_0+n_1}$$

Irreducibility $\Leftrightarrow$ any value of the following linear forms $\notin \mathbb{Z}$ + fundamental irreducibility

$$\begin{array}{cccccc} a_0 & a_1 & a_2 & a_3 & & \\ c_1+a_0 & c_2+a_0 & c_3+a_0 & b_3+a_0 & b_2+a_0 & b_1+a_0 \end{array}$$

which correspond to the decompositions

$$\begin{aligned} 3111,3111,3111 &= 1000,1000,0001 \oplus 2111,2111,3110 \\ &= 1000,1000,0010 \oplus 2111,2111,3101 \\ &= 1000,1000,0100 \oplus 2111,2111,3011 \\ &= 1000,0001,1000 \oplus 2111,3110,2111 \\ &= 1000,0010,1000 \oplus 2111,3101,2111 \\ &= 1000,0100,1000 \oplus 2111,3011,2111 \\ &= 0001,1000,1000 \oplus 3110,2111,2111 \\ &= 0010,1000,1000 \oplus 3101,2111,2111 \\ &= 0100,1000,1000 \oplus 3011,2111,2111 \\ &= 3(1000,1000,1000) \oplus 0111,0111,0111 \end{aligned}$$

Here the equation $Pu = 0$ with a non-rigid Riemann scheme is reduced to a basic equation $P_B u_B = 0$ by middle convolutions and additions.
The connection coefficients, an integral representation, a local solution in a power series of the equation are expressed as above by the connection coefficients c_B, an solution $\sum_{n_0=0}^{\infty} C_{n_0} x^{b_3+n_0}$vof the solution $u_B(x)$ of $P_b u_B = 0$. The fundamental irreducibility is the condition of irreducibility of the equation $P_B u_B = 0$.
P_B is calculated by `mcop()` for a given differential operator P.

```
[38] os_md.getbygrs([[0,a,b],[[2,"?"],0],[d,e,f]],[""]|mat=1);
[[[1,0],[1,a],[1,b]],[[2,-1/2*a-1/2*b-1/2*d-1/2*e-1/2*f],[1,0]],[[1,d],[1,e],[1,f]]]
[39] os_md.getbygrs([[0,a,b],[[2,"?"],0],[d,e,f]],["","short"]);
[[0,a,b],[[2,-1/2*a-1/2*b-1/2*d-1/2*e-1/2*f+1],0],[d,e,f]]
```

53. `spslm(`m`,[`k_1`,`k_2`,...])`
:: Gets a semilocal monodromy of a rigid linear ordinary differential equation
- m : spectral type of a generalized Riemann scheme.
- calculate the semilocal monodromy corresponding to the simple curve (path) around singular points indexed by $k_1, k_2, \ldots$ (cf. [O7]).
- return the list of generalized Riemann scheme and the logarithm of the semilocal monodromy

```
[0] os_md.spslm("21,21,21,21",[0,3]);
[1] [[[[2,a0],[1,a1]],[[2,0],[1,-c-d-a1-2*a0]],[[2,0],[1,c]],[[2,0],[1,d]]],
     [[1,0],[1,a0],[1,d+a1+a0]]]
```

This means that the Jordan Pochhammer equation (of Schlesinger type) with the generalized Riemann scheme

$$\begin{Bmatrix} x=\infty & 0 & 1 & y \\ [a_0]_2 & [0]_2 & [0]_2 & [0]_2 \\ a_1 & -2a_0-a_1-c-d & c & d \end{Bmatrix}$$

is studied and the eigenvalues of the monodromy matrix (which is semisimple if the parameters are generic) corresponding to the simple curve which contains ∞ and y in the region surrounded by the curve are 1, $e^{2\pi i a_0}$, $e^{2\pi i(a_0+a_1+d)}$.

54. `shiftop(`ℓ, s`|zero=1,raw=`k`,all=`t`,dviout=1)`

:: Gets the shift operator of rigid ODE with a spectral type ℓ corresponding to a shift$s$
ℓ may be a Riemann scheme and s may be a list of the parameter and the shifted parameter.
Gets the list of the three elements, shift operator, differential operator and the Riemann scheme

- `zero=1` : some of the exponents are normalized to 0
- `raw=1` : normalize including the constant multiple
  `raw=2` : give the constant with the composition with its left inverse shift operator
  The constant vanishes $\Leftrightarrow$ the shift operator is not isomorphic
  `raw=3` : give the list of the shift operator and the above constant
  `raw=4` : give the list of the shift operator, the above constant and the left inverse shift operator
- `all=0` : give only the shift operator
  `all=1` : give the list of shift operator, differential operator, Riemann scheme, shifted differential operator, shifted Riemann scheme
- `dviout=1` : the result is displayed by `dviout()`. the option `all` is possible. `all=1` means to display including the differential operator.

The exponents to be shifted are better to be the latter ones in each singular points for a faster calculation.

```
[0] os_md.shiftop("11,11,11","00,01,0-1"|zero=1);
[1] [x*dx-b,(-x^2+x)*dx^2+((b+c-2)*x-b+1)*dx+a^2+(b+c-1)*a,
 [[[1,a],[1,-a-b-c+1]],[[1,0],[1,b]],[[1,0],[1,c]]]]
[2] R=os_md.shiftop("11,11,11","00,01,0-1"|zero=1,raw=2,all=0);
a^2+(b+c-1)*a+(c-1)*b
[3] fctr(R);
[[1,1],[a+c-1,1],[a+b,1]]
[4] os_md.shiftop("11,11,11","00,01,0-1"|zero=1,raw=4,all=0);
[x*dx-b,a^2+(b+c-1)*a+(c-1)*b,(x-1)*dx-c+1]
[5] os_md.shiftop("11,11,11","10,00,0-1"|zero=1,raw=2,all=0);
a^2+b*a
[6] fctr(@@);
[[1,1],[a,1],[a+b,1]]
[7] os_md.shiftop("11,11,11","10,00,0-1"|zero=1,raw=3,all=0);
[x*dx+a,a^2+b*a]
[8] os_md.shiftop("11,11,11","1-1,00,00"|zero=1,dviout=1,all=1)$
```

Shift Operator

$$\begin{Bmatrix} x=\infty & 0 & 1 \\ a & 0 & 0 \\ -a-b-c+1 & b & c \end{Bmatrix} = \{u \mid Pu = 0\}$$

$$\overset{Q_1}{\underset{Q_2}{\rightleftarrows}} \begin{Bmatrix} x=\infty & 0 & 1 \\ a+1 & 0 & 0 \\ -a-b-c & b & c \end{Bmatrix}$$

$$
\begin{aligned}
Q_1 &= (2a+b+c)x(x-1)\partial + a((2a+b+c)x-a-c)\\
Q_2 &= -(2a+b+c)x(x-1)\partial + (a+b+c)((2a+b+c)x-a-b)\\
Q_2Q_1 &\equiv a(a+c)(a+b)(a+b+c) \mod W(x)P\\
P &= -x(x-1)\partial^2 + ((b+c-2)x-b+1)\partial + a(a+b+c-1)
\end{aligned}
$$

55. `shiftPfaff(`a`,`b`,`g`,`x`,[`μ_1`,`μ_2`])`
:: Transformation of an adjacent relation of Pfaffian systems under a middle convolution
Here $a(x)$, $b(x)$ and $g(x)$ are square matrices of rational functions and $\mu_1-\mu_2\in\mathbb{Z}$ should be valid. When

$$
\begin{aligned}
u'(x) &= a(x)u, \quad u(x) = \frac{1}{\Gamma(\mu_1)}\int_{x_0}^{x}\bar{u}(t)(x-t)^{\mu_1-1}dt,\\
v'(x) &= b(x)v, \quad v(x) = \frac{1}{\Gamma(\mu_2)}\int_{x_0}^{x}\bar{v}(t)(x-t)^{\mu_2-1}dt,\\
\bar{v}(x) &= g(x)\bar{u}(x),
\end{aligned}
$$

this function returns the matrix $G(x)$ satisfying $u(x)=G(x)v(x)$.
- If $\mu_1=\mu_2$, [μ_1,μ_2] may be simply replaced by μ_1.

56. `conf1sp(`m`|x2=`± 1`,conf=0)`
:: Confluence (Poincare rank 1) of a differential operator with spectral type m
Partitions corresponding to a spectral type are assumed to the singularities ∞, $\frac{1}{c}$, $x_2,\dots$ in this order and the partition corresponding to ∞ should be a refinement of the partition corresponding to $\frac{1}{c}$.
If `x2=-1` is indicated, $\frac{1}{c2},\dots$ are replaced by $x_2,\dots$.
If `conf=0` is indicated, the singular points are ∞, 0, c, $x_3,\dots$ and the confluence of the points 0 and c is calculated.

```
[0] P=os_md.conf1sp([[1,1],[1,1],[1,1]]|x2=1);
(-c*x^2+(x_2*c+1)*x-x_2)*dx^2+(((-a01-a00-1)*c+a11)*x+x_2*c-x_2*a11+a01+a00)*dx
-a00*a01*c+a00*a11
[1] os_md.fctrtos(P|TeX=1,var=dx);
-(x-x_2)(cx-1){dx}^2-(((a_{01}+a_{00}+1)c-a_{11})x-x_2c+x_2a_{11}-a_{01}-a_{00})
[2] os_md.expat(P,x,"?");
[[x_2,[-a01-a00+1,0]],[(1)/(c),[(a11)/(c),0]],
 [infty,[(a01*c-a11)/(c),a00]]]
[3] Q=os_md.conf1sp([[1,1],[1,1],[1,1]]);
(-c*x^2+x)*dx^2+(((-a01-a00-1)*c+a11)*x+a01+a00)*dx-a00*a01*c+a00*a11
[4] subst(Q,c,0);
x*dx^2+(a11*x+a01+a00)*dx+a00*a11
[5] P=os_md.conf1sp([[1,1],[1,1],[1,1]]|conf=0);
(-x^2+c*x)*dx^2+((a01-2)*x+(-a01+1)*c+a11)*dx+a20^2+(a01-1)*a20
[6] os_md.expat(P,x,"?");
[[0,[(a01*c-a11)/(c),0]],[c,[(a11)/(c),0]],[infty,[-a20-a01+1,a20]]]
[7] subst(-P,c,0);
-x^2*dx^2+((a01-2)*x+a11)*dx+a20^2+(a01-1)*a20
[8] os_md.fctrtos(-@@|TeX=1,var=dx);
x^2{dx}^2-((a_{01}-2)x+a_{11}){dx}-a_{20}(a_{20}+a_{01}-1)
```

57. `pf2kz(`m`|all=1)`
:: Converts a Pfaffian system m of n variables to a KZ system of $n+2$ variables
When $n=2$, the Pfaffian system

$$\frac{\partial u}{\partial x} = \frac{A_1 u}{x} + \frac{A_2 u}{x-1} + \frac{A_0 u}{y}$$
$$\frac{\partial u}{\partial y} = \frac{A_3 u}{y} + \frac{A_4 u}{y-1} + \frac{A_0 u}{x}$$

with $m = $ `[`A_0, A_1, A_2, A_3, A_4`,]` is transformed into

$$\frac{\partial u}{\partial x_i} = \sum_{0\le j\le 3,\ j\ne i} \frac{A_{i,j} u}{x_i - x_j} \quad (0 \le i \le 3).$$

Here

$$(x_0, x_1, x_2, x_3, x_4) = (x, y, 0, 1, \infty)$$
$$A_{i,j} = A_{j,i},\ A_{0,j} = A_j,\ A_{1,2} = A_3,\ A_{1,3} = A_4,$$
$$\sum_{0\le i<j\le 3} A_{i,j} = \sum_{0\le \nu\le 4,\ \nu\ne i} A_{i,j} = 0.$$

The function returns the list of $n+3$ elements `[`A_{23}`,` A_{04}`,` A_{14}`,` A_{24}`,` A_{34}`]`.
In general, the coordinates of a Pfaffian system and those of a KZ equation are given by $(x_0, x_1, \dots, x_{n+2}) = (x, y_1, \dots, y_{n-1}, 0, 1, \infty)$.
If `all=1` is indicated, this function returns the list of A_{ij} arranged in the lexicographic order of (i,j), which has $\frac{(n+2)(n+3)}{2}$ elements.

```
[0] N=5$P=[1,2,3,4,5]$MM=os_md.pf2kz(P|all=1)$M=newmat(N,N)$
[1] for(I=0;I<N-1;I++) for(J=I+1;J<N;J++)
      M[I][J]=M[J][I]=MM[os_md.lext2([I,J,N])[0]]
[2] dviout(M)$
```

$$\begin{pmatrix} 0 & 1 & 2 & 3 & -6 \\ 1 & 0 & 4 & 5 & -10 \\ 2 & 4 & 0 & -15 & 9 \\ 3 & 5 & -15 & 0 & 7 \\ -6 & -10 & 9 & 7 & 0 \end{pmatrix}$$: the sum of diagonal elements and that in each line are 0

58. `confexp([[`c_0`,`f_0`],[`c_1`,`f_1`],...,[`c_m`,`f_m`]])` `confexp([[`f_1`,...,`f_n`],[`c_1`,...,`c_m`]]|sym=`k`)`
:: Confluence of characteristic exponents

- A sum of constant multiples of $\frac{1}{x-c_j}$ $(j=0,\dots,m)$ is expressed by a sum of constant multiples of $\frac{1}{\prod_{\nu=0}^{j}(x-c_\nu)}$ $(j=0,\dots,m)$. Namely, this function returns the list `[`λ_0`,`λ_1`,...,`λ_m`]` of the coefficients given by

$$\sum_{j=0}^{m} \frac{f_j}{x-c_j} = \sum_{k=0}^{m} \frac{\lambda_k}{\prod_{\nu=0}^{k}(x-c_\nu)}$$

(cf. `paracmpl()`). Here f_j are matrices without the variable x.
The parameter of this function may be a list of the parameters as above. Then this function returns the list of the corresponding results.
Note that we have

$$\sum_{j=0}^{m} \frac{f_j}{1-c_j y} = \sum_{k=0}^{m} \frac{\lambda_k y^k}{\prod_{\nu=0}^{k}(1-c_\nu y)}.$$

by dividing the above equality by x and putting $x=\frac{1}{y}$.

- If `sym=1` is indicated, this function returns the $n\times m$ matrix M satisfying

$$\begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} = M \begin{pmatrix} \frac{x^{m-1}}{(x-c_1)\cdots(x-c_m)} \\ \frac{x^{m-2}}{(x-c_1)\cdots(x-c_m)} \\ \vdots \\ \frac{1}{(x-c_1)\cdots(x-c_m)} \end{pmatrix}.$$

  Here $(x-c_1)\cdots(x-c_m)f_j$ should be polynomials of x with degree $\leq (m-1)$.
- If `sym=2` is indicated, this function returns the list `[`hf_1`,...,`hf_n`]` of polynomials of x with $h=(x-c_1)\dots(x-c_m)$.
- If `sym=3` is indicated, this function returns the $n\times m$ matrix M satisfying

$$\begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} = M \begin{pmatrix} \frac{1}{x-c_1} \\ \frac{1}{(x-c_1)(x-c_2)} \\ \vdots \\ \frac{1}{(x-c_1)\cdots(x-c_m)} \end{pmatrix}.$$

```
[0] F=[[c0,((c0-c1)*a0+a1)/(c0-c1)],[c1,(a1)/(-c0+c1)]];
[1] os_md.confexp(F);
[a0,a1]
[2] G=os_md.mysubst(F,[c0,0]);
[[0,(a0*c1-a1)/(c1)],[c1,(a1)/(c1)]]
[3] os_md.confexp(G);
[a0,a1]
[4] os_md.confexp([[1/(x-a1),1/(x-a2)],[a1,a2]]|sym=1);
[ 1 -a2 ]
[ 1 -a1 ]
[5] os_md.confexp([[1/(x-a1),1/(x-a1)/(x-a2)],[a1,a2]]|sym=1);
[ 1 -a2 ]
[ 0 1 ]
[6] os_md.confexp([[1/(x-a1),1/(x-a2)],[a1,a2]]|sym=2);
[x-a2,x-a1]
```

59. `mcvm(`n`|var=`x`,z=1,get=`g`)` `mcvm([`n_1`,`n_2`,...]|var=[`a`,`b`,...],z=1,get=`g`)`
`mcvm([`r`,`k`,`i`,`j`]|e=1,var=[`a`,`b`,...])`
:: Middle convolution of versal unfolding
Returns the middle convolution of versal Schlesinger systems

$$\frac{du}{dx}=\sum_{j=1}^{n}\frac{A_j}{(x-a_1)(x-a_2)\cdots(x-a_j)}u,$$

$$\frac{du}{dx}=-\sum_{j=1}^{n}\frac{A_j x^{j-1}}{(1-a_1x)(1-a_2x)\cdots(1-a_jx)}u,$$

$$\frac{du}{dx}=\sum_{i=1}^{n}\sum_{j=1}^{r_i}\frac{A_{ij}}{(x-a_{i1})(x-a_{i2})\cdots(x-a_{ij})}u,$$

$$\frac{du}{dx}=\sum_{i=1}^{n}\sum_{j=1}^{r_i}\frac{A_{ij}}{(x-a_{i1})(x-a_{i2})\cdots(x-a_{ij})}u-\sum_{j=1}^{r_0}\frac{A_{0j}x^{j-1}}{(1-a_{01}x)(1-a_{02}x)\cdots(1-a_{0j}x)}u.$$

Fuchsian system

$$\frac{du}{dx} = \sum_{j=a}^{p} \frac{C_j}{x - a_j} u$$

with square matrices C_j of size N is expressed by

$$u' = (C_1, \ldots, C_p)\, {}^t(p_1, \ldots, p_n) u$$

and then the function

$$\tilde{u}(x) = \begin{pmatrix} \Gamma(\mu+1)^{-1} \displaystyle\int_c^x u(t)(x-a_1)^{-1}(x-t)^{\mu} dt \\ \vdots \\ \Gamma(\mu+1)^{-1} \displaystyle\int_c^x u(t)(x-a_n)^{-1}(x-t)^{\mu} dt \end{pmatrix}$$

defined by the solution $u(x)$ (c is a_j or ∞) satisfies

$$\begin{aligned} \tilde{u}' &= \operatorname{diag}(p_1 I_N, \ldots, p_n I_N)\big({}^t(1_N, \ldots, I_N)(C_1, \ldots, C_n) + \mu I_{pN}\big)\tilde{u} \\ &= \sum_{j=1}^{n} \frac{\tilde{C}_j}{x - a_j} \tilde{u}, \quad \tilde{C}_j = \begin{matrix} \\ \\ j) \end{matrix} \overset{\overset{j}{\smile}}{\begin{pmatrix} 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ C_1 & \cdots & C_j + \mu I_N & \cdots & C_n \\ 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \cdots & \vdots & \cdots & \vdots \end{pmatrix}}. \end{aligned}$$

This follows from

$$\begin{pmatrix} (x-a_1)\frac{d}{dx}\frac{u}{x-a_1} \\ (x-a_2)\frac{d}{dx}\frac{u}{x-a_2} \end{pmatrix} = \begin{pmatrix} \frac{(C_1-1)u}{x-a_1} + \frac{C_2 u}{x-a_2} \\ \frac{C_1 u}{x-a_1} + \frac{C_2 - 1 u}{x-a_2} \end{pmatrix} = \begin{pmatrix} C_1 - 1 & C_2 \\ C_1 & C_2 - 1 \end{pmatrix} \begin{pmatrix} \frac{u}{x-a_1} \\ \frac{u}{x-a_2} \end{pmatrix}$$

because the integral transformation $v(x) \mapsto \int_c^x v(t)(x-t)^{\mu} dt$ transforms $x\frac{d}{dx}$ and $\frac{d}{dx}$ to $x\frac{d}{dx} - \mu - 1$ and $\frac{d}{dx}$, respectively.

- Consider the first equation. Change the base by

$$q_j(x) = \frac{1}{(x-a_1)\cdots(x-a_j)} \qquad (j = 1, \ldots, n),$$

$$\hat{u}(x) = \begin{pmatrix} \Gamma(\mu+1)^{-1} \displaystyle\int_c^x u(t) q_1(x)(x-t)^{\mu} dt \\ \vdots \\ \Gamma(\mu+1)^{-1} \displaystyle\int_c^x u(t) q_n(x)(x-t)^{\mu} dt \end{pmatrix}$$

  and suppose that A_k of the first equation are changed to $\hat{A}_k$. In this case, this function returns $[A'_1, \ldots, A'_p]$ with `mcvm(`n`|get=1)` such that

$$\hat{A}_k = \big(\delta_{i,k} A_j\big)_{\substack{1 \le i \le n \\ 1 \le j \le n}} + \mu A'_k$$

  Here $A'_k = \big(D_{n,k,i,j} I_N\big)_{\substack{1 \le i \le n \\ 1 \le j \le n}}$ and $D_{n,k,i,j}$ is obtained by the option `e=1`, which will be mentioned later.

- If the option `get` is not indicated, the function returns thw matrix S given by

$$\begin{pmatrix} p_1(x) \\ \vdots \\ p_n(x) \end{pmatrix} = S \begin{pmatrix} q_1(x) \\ \vdots \\ q_n(x) \end{pmatrix}$$

where $\tilde{u} = S^{-1}\hat{u}$ and $(C_1, \ldots, C_n)S = (A_1, \ldots, A_n)$.

$$\begin{aligned} \hat{u}' = S^{-1}\tilde{u}' &= S^{-1} \begin{pmatrix} p_1 \\ \vdots \end{pmatrix} (C_1, \ldots)\, \tilde{u} + S^{-1}\mu \begin{pmatrix} p_1 & & \\ & \ddots & \end{pmatrix} \tilde{u} \\ &= \mathbf{q}\,(A_1, A_2, \ldots)\, \hat{u} + \mu S^{-1} \begin{pmatrix} S_1\mathbf{q} & & \\ & S_2\mathbf{q} & \\ & & \ddots \end{pmatrix} S\hat{u}, \end{aligned}$$

$$\sum A_k q_k = S^{-1} \begin{pmatrix} S_1\mathbf{q} & & \\ & S_2\mathbf{q} & \\ & & \ddots \end{pmatrix} S, \quad S = \begin{pmatrix} S_1 \\ S_2 \\ \vdots \end{pmatrix}, \quad \mathbf{q} = \begin{pmatrix} q_1 \\ q_2 \\ \vdots \end{pmatrix}.$$

- Consider the second equation. Then $(x - a_i)^{-1}$ and $q_i(x)$ are replaced by $(\frac{1}{a_i} - x)^{-1}$ and $x^{i-1}(1 - a_1x)^{-1} \cdots (1 - a_ix)^{-1}$, respectively, ahd this functions returns the corresponding result.
- By the option `var=[`a`,`b`,`c`,...]` the points a_i representing the singular points are replaced by a, b, c, ...
- In rhe case of the 3-rd equation or the 4-th equation, ℓ is `[`$r_1, \ldots, r_n$`]` or `[`$r_0, r_1, \ldots, r_n$`]`, respectively, and the option `z=1` should be indicated in the latter case.
- Suppose the above Fuchs system can be extended to the KZ system

$$\begin{aligned} &\frac{\partial u}{\partial x_i} = \sum_{\nu \in \{0,\ldots,i\} \setminus \{i\}}^{p} \frac{C_{i,\nu}}{x_i - x_\nu} \quad (i = 0, \ldots, p), \\ &C_{i,j} = C_{j,i},\ C_{0,j} = C_j \end{aligned}$$

by putting $x_0 = x$. Then the convolution transforms $C_{0,j} = C_j$ to $\hat{C}_{0,j} = \tilde{C}_j$ and it is extended to the KZ system and the resulting system is

$$\tilde{C}_{i,j} = \begin{array}{c} \\ i) \\ \\ \\ \\ j) \\ \\ \end{array} \begin{array}{c} \overset{\smile}{i} \qquad\qquad\qquad\qquad \overset{\smile}{j} \\ \begin{pmatrix} C_{i,j} & & & & & & & \\ & \ddots & & & & & & \\ & & C_{i,j} + C_{0,j} & & & -C_{0,j} & & \\ & & & C_{i,j} & & & & \\ & & & & \ddots & & & \\ & & -C_{0,i} & & & C_{i,j} + C_{0,i} & & \\ & & & & & & \ddots & \\ & & & & & & & C_{i,j} \end{pmatrix} \end{array}$$

- Suppose `get=2` is indicated.
  The function returns the list of matrices whose elements are the transformation of $\tilde{C}_{i,j}$ $(1 \leq i < j \leq p)$ excluded the diagonal elements $C_{i,j}$ and expressed by $A_{0,\nu}$. Here A_j or $A_{i,j}$ are used for the versal unfolding and $A_{0,\nu}$ is expressed as `a0`ν by default. This gives the a convolution of KZ equation with unramified irregular singularities (cf. [O9]).

- Suppose `get=3` is indicated.
  The matrix $A_1(x)$ of the equation $\frac{\partial \hat{u}}{\partial x_1} = A_1(x)\hat{u}$ corresponding to the 3-rd equation with $n = 2$ and $x_0 = x$ and $a_{1,j} = x_1 - a_j$ and $a_{2,j} = x_2 - b_j$ is expressed by using a complete base

$$\Big\{\frac{1}{(x_1-x_0)^\nu} \mid \nu = 1,\dots,r_1\Big\} \cup \Big\{\frac{1}{x_1-x_2+a_1-b_1},\dots,$$
$$\frac{1}{(x_1-x_2+a_1-b_1)\cdots(x_1-x_2+a_1-b_{r_2})},\dots,\frac{1}{(x_1-x_2+a_1-b_1)\cdots(x_1-x_2+a_{r_1}-b_{r_2})}\Big\}.$$

  This function returns a list whose first part is the list of the latter r_1r_2 coefficients with respect the base. Here $a_1 = b_1 = 0$ is substituted and the diagonal parts $A_{i,j}$ are omitted. The latter part of the list is the substitution $a_2 = \cdots = a_{r_1} = b_1 = \cdots = b_{r_2} = 0$ of the first part.

  The result in the case when $n > 2$ is easily obtained by that in the case when $n = 2$ (cf. [O9]).

$$I = \{i_0, i_0+1, \dots, i_0+r_I-1\}, \quad J = \{j_0, j_0+1, \dots, j_0+r_J-1\},$$
$$du = \sum_\nu \frac{C_{0,\nu}}{x-x_\nu}dx = \sum_{i\notin I}\frac{C_{0,i}}{x-x_\nu}du + \sum_{i\in I}\frac{A_{0,i}}{(x-x_{i_0})\cdots(x-x_i)}du,$$
$$\frac{\partial \hat{u}}{\partial x_{i_0}} = \sum_{i\in I}\Big(\frac{\tilde{A}_{i,0}}{(x_{i_0}+e_{i_0}-x_0)\cdots(x_{i_0}+e_i-x_0)} + S^{-1}\sum_{\nu\notin I,\,\nu\neq 0}\frac{\tilde{C}_{i,\nu}}{x_{i_0}+e_i-x_\nu}S\Big)\hat{u},$$
$$\tilde{C}_{i,\nu} = \Big(C_{i,\nu}\delta_{p,q} + \delta_{p,i}(\delta_{q,i}-\delta_{q,\nu})C_{0,\nu} + \delta_{p,\nu}(\delta_{q,\nu}-\delta_{q,i})C_{0,i}\Big)_{p,q},$$

$$\frac{1}{x_{i_0}+e_i-x_{j_0}-e_j} = \sum_{k\in I,\,\ell\in J}U_{(i,j),(k,\ell)}q_{k,\ell},$$
$$S^{-1}\sum_{i\in I,\ j\in J}\Big(C_{i,j}\delta_{p,q} + \delta_{p,i}(\delta_{q,i}-\delta_{q,j})C_{0,j} + \delta_{p,j}(\delta_{q,j}-\delta_{q,i})C_{0,i}\Big)_{p,q}(x_{i_0}+e_i-x_{j_0}-e_j)^{-1}S$$
$$= \sum_{i\in I,\,j\in J}\sum_{k\in I,\,\ell\in J}S^{-1}\Big(C_{i,j}\delta_{p,q} + \delta_{p,i}(\delta_{q,i}-\delta_{q,j})C_{0,j} + \delta_{p,j}(\delta_{q,j}-\delta_{q,i})C_{0,i}\Big)_{p,q}SU_{(i,j),(k,\ell)}q_{k,\ell}$$
$$= \sum_{k\in I,\,\ell\in J}\Big(A_{k,\ell}\delta_{p,q}\Big)_{p,q}q_{k,\ell}$$
$$+ \sum_{i\in I,\,j\in J}\sum_{k\in I,\,\ell\in J}S^{-1}\Big(\delta_{p,i}(\delta_{q,i}-\delta_{q,j})\sum_{s=i_0}^{i}A_{0,s}q_{s-i_0+1,\{1,\dots,s-i_0\}}(x_{i_0},\dots)\Big)_{p,q}SU_{(i,j),(k,\ell)}q_{k,\ell}$$
$$+ \sum_{i\in I,\,j\in J}\sum_{k\in I,\,\ell\in J}S^{-1}\Big(\delta_{p,i}(\delta_{q,j}-\delta_{q,i})\sum_{s=j_0}^{j}A_{0,s}q_{s-j_0+1,\{1,\dots,s-j_0\}}(x_{j_0},\dots)\Big)_{p,q}SU_{(i,j),(k,\ell)}q_{k,\ell}.$$

- If `get=4` is indicated, the latter element of the list given by `get=3` is returned.
- If `e=1` is indicated and the first parameter is `[r,k,i,j]`, then the polynomial $D_{r,k,i,j}$ of degree $j+k-i-1$ is returned (r is ignored).

$$S = \Big(S_{i,j}\Big)_{i,j}, \quad S_{i,j} = \begin{cases} 0 & (i<j) \\ s_{i,\{1,\dots,j\}} & (i\geq j)\end{cases} \quad (\text{最初の方程式}),$$
$$s_{i,\{j_1,\dots,j_k\}}(\mathbf{a}) := \prod_{j=1}^{k}(a_i - a_{j_\nu}), \quad p_i = \sum_{j=1}^{i}s_{i,\{1,\dots,j-1\}}q_j,$$
$$D_{r,k,i,j}(\mathbf{a}) := \sum_{\nu=\max\{k,j\}}^{i}\frac{s_{\nu,\{1,\dots,j-1\}}}{s_{\nu,\{k,k+1,\dots,i\}\setminus\{\nu\}}}.$$

- If `e=2` is indicated and the first parameter is `[r,k,i,j]`, then the polynomial $D'_{r,k,i,j}$ of degree $j+k-i-1$ is returned.

$$D'_{r,k,i,j}(\mathbf{a}') := \sum_{\nu=\max\{k,j\}}^{i} \frac{a_{\nu+1} s_{\nu+1,\{2,\ldots,j\}}}{s_{\nu+1,\{k,k+1,\ldots,i+1\}\setminus\{\nu+1\}}} = D_{r,k,i+1,j+1}(\mathbf{a})|_{a_1=0}.$$

```
[0] os_md.mcvm([2,1]);
[ 1 0 0 ]
[ 1 -a1+a2 0 ]
[ 0 0 1 ]
[1] os_md.mcvm([2,2]|var=[a,b]);
[ 1 0 0 0 ]
[ 1 -a1+a2 0 0 ]
[ 0 0 1 0 ]
[ 0 0 1 -b1+b2 ]
[2] os_md.mcvm([2,1]|get=1);
[[ 1 0 0 ]
 [ 0 1 0 ]
 [ 0 0 0 ],
 [ 0 0 0 ]
 [ 1 -a1+a2 0 ]
 [ 0 0 0 ],
 [ 0 0 0 ]
 [ 0 0 0 ]
 [ 0 0 1 ]]
```

This is the 3-rd equation with $n=2$, $r_1=2$, $r_2=1$ and we have

$$\hat{A}_{1,1} = \begin{pmatrix} A_1+\mu & A_2 & A_3 \\ 0 & \mu & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \hat{A}_{1,2} = \begin{pmatrix} 0 & 0 & 0 \\ A_1+\mu & A_2+(a_2-a_1)\mu & A_3 \\ 0 & 0 & 0 \end{pmatrix},$$

$$\hat{A}_{2,1} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ A_1 & A_2 & A_3+\mu \end{pmatrix}$$

```
[3] subst(os_md.mcvm([1,3]|get=1),b1,0);
[[ 1 0 0 0 ]
 [ 0 0 0 0 ]
 [ 0 0 0 0 ]
 [ 0 0 0 0 ],
 [ 0 0 0 0 ]
 [ 0 1 0 0 ]
 [ 0 0 1 0 ]
 [ 0 0 0 1 ],
 [ 0 0 0 0 ]
 [ 0 0 0 0 ]
 [ 0 1 b2 0 ]
 [ 0 0 1 b3 ],
```

```
 [ 0 0 0 0 ]
 [ 0 0 0 0 ]
 [ 0 0 0 0 ]
 [ 0 1 b3 -b3*b2+b3^2 ]]
[4] M=os_md.mcvm([1,3])$
[5] A=os_md.mcvm([1,3]|get=2)$
[6] M=subst(M,b1,0)$ A=subst(A,b1,0)$
[7] map(red,M[1][1]*A[0]+M[2][1]*A[1]+M[3][1]*A[2]); /* A12 */
[ a02 -a02 -a03 -a04 ]
[ -a01 a01 0 0 ]
[ 0 0 a01 0 ]
[ 0 0 0 a01 ]
[8] map(red,M[1][2]*A[0]+M[2][2]*A[1]+M[3][2]*A[2]); /* A13 */
[ a03 -a03 -a03*b2-a04 -a04*b3 ]
[ 0 0 0 0 ]
[ -a01 a01 a01*b2 0 ]
[ 0 0 a01 a01*b3 ]
[9] map(red,M[1][3]*A[0]+M[2][3]*A[1]+M[3][2]*A[3]); /* A14 */
[ a04 -a04 -a04*b3 a04*b3*b2-a04*b3^2 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[ -a01 a01 a01*b3 -a01*b3*b2+a01*b3^2 ]
```

This shows that the convolution of the system

$$
\begin{aligned}
\frac{\partial u}{\partial x_0} &= \Big(\frac{A_{01}}{x_0-x_1}+\sum_{j=2}^{4}\frac{A_{0j}}{(x_0-x_2)(x_0-x_2-b_2)\cdots(x_0-x_2-b_{j-1})}\Big)u,\\
\frac{\partial u}{\partial x_1} &= \Big(\frac{A_{10}}{x_1-x_0}+\sum_{j=2}^{4}\frac{A_{1j}}{(x_1-x_2)(x_1-x_2-b_2)\cdots(x_1-x_2-b_{j-1})}\Big)u,\\
\frac{\partial u}{\partial x_2} &= \sum_{i=0}^{1}\sum_{j=2}^{4}\Big(\frac{(-1)^j A_{ji}}{(x_2-x_i)(x_2-x_i+b_2)\cdots(x_2-x_1+b_{j-1})}\Big)u,\\
A_{i,j} &= A_{j,i}
\end{aligned}
$$

is given by

$$
\hat{A}_{01}=\begin{pmatrix} A_{01}+\mu & A_{02} & A_{03} & A_{04}\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\end{pmatrix},\quad
\hat{A}_{02}=\begin{pmatrix} 0 & 0 & 0 & 0\\ A_{01} & A_{02}+\mu & A_{03} & A_{04}\\ 0 & 0 & \mu & 0\\ 0 & 0 & 0 & \mu\end{pmatrix},
$$

$$
\hat{A}_{03}=\begin{pmatrix} 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ a_{01} & A_{02}+\mu & A_{03}+b_2\mu & A_{04}\\ 0 & 0 & \mu & b_3\mu\end{pmatrix},\quad
\hat{A}_{04}=\begin{pmatrix} 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ A_{01} & A_{02}+\mu & A_{03}+b_3\mu & A_{04}+b_3(b_3-b_2)\mu\end{pmatrix},
$$

$$
\hat{A}_{12}=\begin{pmatrix} A_{12}+A_{02} & -A_{02} & -A_{03} & -A_{04}\\ -A_{01} & A_{12}+A_{01} & 0 & 0\\ 0 & 0 & A_{12}+A_{01} & 0\\ 0 & 0 & 0 & A_{12}+A_{01}\end{pmatrix},
$$

$$\hat{A}_{13} = \begin{pmatrix} A_{13}+A_{03} & -A_{03} & -b_2A_{03}-A_{04} & -b_3A_{04} \\ 0 & A_{13} & 0 & 0 \\ -A_{01} & A_{01} & A_{13}+A_{01} & b_3A_{01} \\ 0 & 0 & A_{01} & A_{13}+b_3A_{01} \end{pmatrix},$$

$$\hat{A}_{14} = \begin{pmatrix} A_{14}+A_{04} & -A_{04} & -b_3A_{04} & -b_3(b_3-b_2)A_{04} \\ 0 & A_{14} & 0 & 0 \\ 0 & 0 & A_{14} & 0 \\ -A_{01} & A_{01} & b_3A_{01} & A_{14}+b_3(b_3-b_2)A_{01} \end{pmatrix}$$

with $\tilde{A}_{ij} = \tilde{A}_{ji}$.

```
[10]  os_md.mcvm([2,3]|get=1);
[[ 1 0 0 0 0 ]
 [ 0 1 0 0 0 ]
 [ 0 0 0 0 0 ]
 [ 0 0 0 0 0 ]
 [ 0 0 0 0 0 ],
 [ 0 0 0 0 0 ]
 [ 1 -a1+a2 0 0 0 ]
 [ 0 0 0 0 0 ]
 [ 0 0 0 0 0 ]
 [ 0 0 0 0 0 ],
 [ 0 0 0 0 0 ]
 [ 0 0 0 0 0 ]
 [ 0 0 1 0 0 ]
 [ 0 0 0 1 0 ]
 [ 0 0 0 0 1 ],
 [ 0 0 0 0 0 ]
 [ 0 0 0 0 0 ]
 [ 0 0 0 0 0 ]
 [ 0 0 1 -b1+b2 0 ]
 [ 0 0 0 1 -b1+b3 ],[ 0 0 0 0 0 ]
 [ 0 0 0 0 0 ]
 [ 0 0 0 0 0 ]
 [ 0 0 0 0 0 ]
 [ 0 0 1 -b1+b3 (b2-b3)*b1-b3*b2+b3^2 ]]
[11]  os_md.mcvm([2,3]|get=4);
 [[ a03 0 -a03 -a04 -a05 ]
 [ 0 a03 0 0 0 ]
 [ -a01 -a02 a01 0 0 ]
 [ 0 0 0 a01 0 ]
 [ 0 0 0 0 a01 ],
 [ a04 0 -a04 -a05 0 ]
 [ -a03 a04 a03 a04 a05 ]
 [ a02 0 -a02 0 0 ]
 [ -a01 -a02 a01 -a02 0 ]
 [ 0 0 0 a01 -a02 ],
```

```
[ a05 0 -a05 0 0 ]
[ -2*a04 a05 2*a04 2*a05 0 ]
[ 0 0 0 0 0 ]
[ 2*a02 0 -2*a02 0 0 ]
[ -a01 -a02 a01 -2*a02 0 ],
[ 0 0 0 0 0 ]
[ -3*a05 0 3*a05 0 0 ]
[ 0 0 0 0 0 ]
[ 0 0 0 0 0 ]
[ 3*a02 0 -3*a02 0 0 ],
[ 0 0 0 0 0 ]
[ 0 0 0 0 0 ]
[ 0 0 0 0 0 ]
[ 0 0 0 0 0 ]
[ 0 0 0 0 0 ],
[ 0 0 0 0 0 ]
[ 0 0 0 0 0 ]
[ 0 0 0 0 0 ]
[ 0 0 0 0 0 ]
[ 0 0 0 0 0 ]]
```

This result shows that the convolution of the KZ equation

$$\begin{aligned}
\frac{\partial u}{\partial x_0} &= \Big(\sum_{\nu=1}^{2} \frac{A_{0,\nu}}{(x_0-x_1)^\nu} + \sum_{\nu=1}^{3} \frac{A_{0,\nu+2}}{(x_0-x_2)^\nu}\Big)u,\\
\frac{\partial u}{\partial x_1} &= \Big(\sum_{\nu=1}^{2} \frac{(-1)^{\nu-1}A_{\nu,0}}{(x_1-x_0)^\nu} + \sum_{\nu=1}^{4} \frac{A_{1,\nu+2}}{(x_1-x_2)^\nu}\Big)u,\\
\frac{\partial u}{\partial x_2} &= \Big(\sum_{\nu=1}^{3} \frac{(-1)^{\nu-1}A_{\nu+2,0}}{(x_2-x_0)^\nu} + \sum_{\nu=1}^{4} \frac{(-1)^{\nu-1}A_{\nu+2,1}}{(x_2-x_1)^\nu}\Big)u,\\
A_{i,j} &= A_{j,i}
\end{aligned}$$

is given by (cf. [O9])

$$\hat{A}_{01} = \begin{pmatrix} A_{01}+\mu & A_{02} & A_{03} & A_{04} & A_{05} \\ 0 & \mu & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \quad \hat{A}_{02} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ A_{01}+\mu & A_{02} & A_{03} & A_{04} & A_{05} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\hat{A}_{03} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ A_{01} & A_{02} & A_{03}+\mu & A_{04} & A_{05} \\ 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & \mu \end{pmatrix}, \quad \hat{A}_{04} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ A_{01} & A_{02} & A_{03}+\mu & A_{04} & A_{05} \\ 0 & 0 & 0 & \mu & 0 \end{pmatrix},$$

$$\hat{A}_{05} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ A_{01} & A_{02} & A_{03}+\mu & A_{04} & A_{05} \end{pmatrix},$$

$$\hat{A}_{13}=\begin{pmatrix} A_{13}+A_{03} & 0 & -A_{03} & -A_{0,4} & -A_{05} \\ 0 & A_{13}+A_{03} & 0 & 0 & 0 \\ -A_{01} & -A_{02} & A_{13}+A_{01} & 0 & 0 \\ 0 & 0 & 0 & A_{13}+A_{01} & A_{02} \\ 0 & 0 & 0 & 0 & A_{13}+A_{01} \end{pmatrix},$$

$$\hat{A}_{14}=\begin{pmatrix} A_{14}+A_{04} & 0 & -A_{04} & -A_{05} & 0 \\ -A_{03} & A_{14}+A_{04} & A_{03} & A_{04} & A_{05} \\ A_{02} & 0 & A_{14}-A_{02} & 0 & 0 \\ -A_{01} & -A_{02} & A_{14}+A_{01} & -A_{02} & 0 \\ 0 & 0 & 0 & A_{01} & A_{14}-A_{02} \end{pmatrix},$$

$$\hat{A}_{15}=\begin{pmatrix} A_{15}+A_{05} & 0 & -A_{05} & 0 & 0 \\ -2A_{04} & A_{15}+A_{05} & 2A_{04} & 2A_{05} & 0 \\ 0 & 0 & A_{15} & 0 & 0 \\ 2A_{02} & 0 & -2A_{02} & A_{15} & 0 \\ -A_{01} & -A_{02} & A_{01} & -2A_{02} & A_{15} \end{pmatrix},$$

$$\hat{A}_{16}=\begin{pmatrix} A_{16} & 0 & 0 & 0 & 0 \\ -3A_{05} & A_{16} & 3A_{05} & 0 & 0 \\ 0 & 0 & A_{16} & 0 & 0 \\ 0 & 0 & 0 & A_{16} & 0 \\ 3A_{02} & 0 & -3A_{02} & 0 & A_{16} \end{pmatrix}$$

60. `anal2sp(`m`,`l`)`

:: Analyzes simultaneous spectral exponents m

$m=[[m_1,\lambda_1,\mu_1],[m_2,\lambda_2,\mu_2],\ldots]$ are the list of lists of multiplicity and a pair of simultaneous eigenvalues

l has the following meaning. Here a spectral type means a pair of eigenvalues.

`0` unifies the same spectral types (works for single eigenvalues).

`["add",`n`]` add spectral type n (works for single eigenvalues).

`["sub",`n`]` delete spectral type n (works for single eigenvalues).

`["swap"]` swap eigenvalues

`["+",`c_1`,`c_2`]` shifts eigenvalues

`["*",`c_1`,`c_2`]` linear combinations of eigenvalues

`["+",`c_1`,`c_2`,...]` shift eigenvalues (number of eigenvalues are arbitrary)

`["*",`c_1`,`c_2`,...]` linear combinations of eigenvalues (number of eigenvalues are arbitrary)

`["mult",`m`]` multiplies the multiplicities by m.

`["get",`f`,`c`]` extracts the elements whose f-th eigenvalue $=c$.

`["put",`f`,`c`]` replaces the f-th eigenvalue by c.

`["get1",`f`]` extracts only f-th eigenvalues (the result is not a pair of eigenvalues)

`["get1",`f`,`c`]` extracts elements f-th eigenvalue equals c (the result is deleted th eigenvalue c)

`["put1",`f`,`c`]` inserts f-th eigenvalue c

`["put1"]` copies eigenvalues

`["max"]` returns the position with the maximal multiplicity with the corresponding element

`["max",`f`,`c`]` returns the position with the maximal multiplicity whose f-th eigenvalue is c and the corresponding element. returns `[-1]` if there are no such element

`["val",`f`]` returns the multiplicity and the sum of eigenvalues of the f-th element.

`[[`$\cdots$`],[`$\cdots$`],...]` executes several commands

```
[0] R=os_md.anal2sp([[2,a,b],[3,a,c],[-1,a,c]],["add",[[-2,a,b],[1,b,c]]]);
[[2,a,b],[3,a,c],[-1,a,c],[-2,a,b],[1,b,c]]  /* define R */
[1] os_md.anal2sp([[2,a,b],[3,a,c],[-1,a,c]],["sub",[[2,a,b],[-1,b,c]]]);
[[2,a,b],[3,a,c],[-1,a,c],[-2,a,b],[1,b,c]]
[2] os_md.anal2sp(R,["swap"]);    /* swap two eigenvalues */
[[2,b,a],[3,c,a],[-1,c,a],[-2,b,a],[1,c,b]]
```

```
[3] os_md.anal2sp(R,0);           /* unify */
[[2,a,c],[1,b,c]]
[4] os_md.anal2sp(R,["+",f,-g]);  /* shift eigenvalues */
[[2,a+f,b-g],[3,a+f,c-g],[-1,a+f,c-g],[-2,a+f,b-g],[1,b+f,c-g]]
[5] os_md.anal2sp(R,["*",f,-g]);  /* linear combination of eigenvalues */
[[2,f*a-g*b],[3,f*a-g*c],[-1,f*a-g*c],[-2,f*a-g*b],[1,f*b-g*c]]
[6] os_md.anal2sp(R,["*",f]);
[[2,f*a],[3,f*a],[-1,f*a],[-2,f*a],[1,f*b]]
[7] os_md.anal2sp([[2,a,b],[3,a,c],[-1,a,c]],["mult",2]); /* multiplicity *= m */
[[4,a,b],[6,a,c],[-2,a,c]]
[8] os_md.anal2sp(R,["get",2,c]); /* extract elements with a given eigenvalue */
[[3,a,c],[-1,a,c],[1,b,c]]
[9] os_md.anal2sp(R,["put",1,f]); /* replace eigenvalues */
[[2,f,b],[3,f,c],[-1,f,c],[-2,f,b],[1,f,c]]
[10] os_md.anal2sp(R,["get",1,a]); /* extract elements with a given eigenvalue */
[[2,a,b],[3,a,c],[-1,a,c],[-2,a,b]]
[11] R1=os_md.anal2sp(R,["get1",1,a]); /* extract elements by a given eigenvalue */
[[2,b],[3,c],[-1,c],[-2,b]]
[12] os_md.anal2sp(R1,["put1",1,a]); /* insert an eigenvalue */
[[2,a,b],[3,a,c],[-1,a,c],[-2,a,b]]
[13] os_md.anal2sp(R1,["put1"]); /* copies eigenvalues */
[[2,b,b],[3,c,c],[-1,c,c],[-2,b,b]]
[14] os_md.anal2sp(R,["val",1]); /* sum of 1-st eigenvalues */
[3,2*a+b]
[15] os_md.anal2sp(R,[["get1",1],["put1"],0]);
[[2,a,a],[1,b,b]]
[16] os_md.anal2sp(R,["max"]);   /* maximal multiplicity */
[1,[3,a,c]]
[17] os_md.anal2sp(R,["max",1,b]);
[4,[1,b,c]]
[18] os_md.anal2sp(R,["max",1,c]);
[-1]
```

61. `mc2grs(`g`,`r`|top=0,dviout=`k`,div=`ℓ`,fig=`s`)`
:: Transformation of simultaneous spectral exponents of KZ equation of 5 points in $\mathbb{P}^1$
The KZ equation

$$du = \sum_{0\le i<j\le 3} A_{i,j}\frac{d(x_i-x_j)}{x_i-x_j}u,$$

$$A_{i,4} = -\sum_{j=0}^{3} A_{i,j},\ A_{j,i} = A_{i,j},\ A_{i,i} = 0$$

satisfies

$$[A_I, A_J] = 0 \quad (I,\ J \subset \{0,1,2,3\},\ \#I = \#J = 2,\ I\cap J = \emptyset)$$

and the simultaneous eigenspace decompositions are possible for the 15 pairs of A_I and A_J satisfying $[A_I, A_J] = 0$. Then g indicates a list of 15 lists each of which consists of `[`I`,`J`]` and the

corresponding dimension of simultaneous eigenspace and two eigenvalues for the pairs A_I and A_J. Here `I=[`i_i`,`i_2`]`, `J=[`j_i`,`j_2`]` with $i_1 < i_2$, $j_1 < j_2$, $i_1 < j_1$. For example, the list corresponding to

$$A_{0,1} = \begin{pmatrix} a & & \\ & a & \\ & & b \end{pmatrix}, \ A_{2,3} = \begin{pmatrix} c & & \\ & c & \\ & & d \end{pmatrix}$$

is `[[[0,1],[2,3]],[2,`a`,`c`],[1,`b`,`d`]]`.
We may assume that $\sum_{0\le i<j\le 3} A_{i,j}$ is a scalar matrix because of the generic irreducibility and we denote the scalar by κ.
Other parameters mean as follows.

- $g = 0$ means trivial `[[[[0,1],[2,3]],[1,0,0]],[[[0,1],[2,4]],[1,0,0]],...]`.
- g may be indicated by spectral types of 4 matrices $A_{0,4}$, $A_{0,1}$, $A_{0,2}$, $A_{0,3}$ (string or GRS for 4 points) in this order. Then the options `dep=[`m`,`n`]`, `int=0` etc. are possible as for the function `m2mc()`.
  If `top=0` is indicated, the order is changed to $A_{0,1}$, $A_{0,2}$, $A_{0,3}$, $A_{0,4}$.
  If g is indicated by spectral type or string, the parameter r has the meaning as follows.
  - $r = 1$: Changes the spectral type in standard order and returns the list of 15 lists of eigenspace decompositions.
  - $r = 3$: Define the parameters of simultaneous eigenvalues by `a,b,c,...` and returns the Fuchs condition and the list of 15 lists.

r determines the following functions

- `0` : returns the list of 15 lists.
- `["sort"]` : arranges the list so that its elements are in standard order (`[ ]` may be omitted).
- `["deg"]` : returns κ (`[ ]` may be omitted).
  Note that $A_{0,1} + A_{0,2} + A_{0,3} + A_{1,2} + A_{1,3} + A_{2,3} = \kappa$ and
  $A_{i,j} = A_{\{0,1,2,3,4\}\setminus\{i,j\}} \begin{cases} +\kappa & (j < 4) \\ -\kappa & (j = 4) \end{cases}$ $(0 \le i < j \le 4)$.
- `[[`i`,`j`],`λ`]` : transforms by the addition $A_{i,j} \mapsto A_{i,j} + \lambda$, $A_{i,4} \mapsto A_{i,4} - \lambda$, $A_{j,4} \mapsto A_{j,4} - \lambda$. $(0 \le i < j \le 3)$.
- `["homog"]` : transforms by the addition $r =$`[[1,3],`$-\kappa$`]` to change $\kappa = 0$ (`[ ]` may be omitted).
- `["homog",[m,n]]` : transforms by the addition $r =$`[[`m`,`n`],`$-\kappa$`]` $(0 \le m < n < 4)$ to change $\kappa = 0$ (`[ ]` may be omitted).
- `["swap",[`i`,`j`]]` : swaps the indices i and j. If i or j is 4, swap them after the homogenization ny $r =$`["homog"]`.
- `["perm",[`$i_0,\dots,i_4$`]]` : permutation of indices $\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ i_0 & i_1 & i_2 & i_3 & i_4 \end{pmatrix}$. If the index 4 is changed, the permutation is applied after the homogenization.
- `[`μ`]` : middle convolution $\mathrm{mc}_{x_0,\mu}$ for the variable x_0 (the algorithm is given by [O5]).
- `[`i`,`μ`]` : middle convolution $\mathrm{mc}_{x_i,\mu}$ for the variable x_i If $i = 4$, this function first transforms by $r =$`["swap",[0,4]]` and then by $\mathrm{mc}_{x_0,\mu}$ and lastly by $r =$`["swap",[0,4]]`.
- `[[[`i_1`,`j_1`],`λ_1`],...,[`i_k`,`μ_k`],...]` : transforms by successive additions middle convolutions.
- `[[`a`,`b`,`c`]]` : means `[[[0,1],`a`],[[0,2],`b`],[[0,3],`c`]]`.
- `[[`a`,`b`,`c`,`μ`]]` : means `[[[0,1],`a`],[[0,2],`b`],[[0,3],`c`],[`μ`]]`.
- `["get",[`I`,`J`]]` : returns the list of simultaneous eigenvalues corresponding to $[A_I, A_J] = 0$ with the header $[I, J]$.
- `["get",`I`]` : returns the list of eigenvalues of A_I with the header I.
- `["get",[`i`,`j`,`k`]]` : returns the spectral data of $A_{i,j} + A_{i,k} + A_{j,k}$ with the header I.
- `["get",`i`]` $(0 \le i \le 4)$: returns the GRS corresponding to $i \in I$, which is the GRS of ODE of the variable x_i.

If `dviout=1` is indicated, the result is displayed tby using TEX.
If `dviout=-1` is indicated, the result is given by the TEX source.
If `dviout=2` is indicated, the result is written in the TEX source file without its display. The result can be displayed later by a command.

- `["get"]` : returns the list of eigenvalues of 10 matrices A_I.
  `dviout=`k can be indicated. In this case the Riemann Scheme is divided to lines by the option `div=`ℓ (cf. `divmattex()`).
- `["get0",[`I`,`J`]]`, `["get0",`I`]`, `["get0",`i`]` $(0 \leq i \leq 4)$, `["get0",[`i`,`j`,`k`]]`, `["get0"]` : same as in the case where `"get0"` is replaced by `"get"` except that the headers are omitted.
- `["get0",`I`,`J`]` : returns the spectral values of $A_I + A_J$.
- `["show"]` : displays the list g by using TEX.
  The option `dviout=-1` is possible.
- `["show0"]` : shows 15 spectral types. The order of 15 types is same as in the case `"show"`.
  The option `dviout=1,-1,2` is possible. Moreover
  - `fig=1` indicates to show the result in a figure by using TikZ.
  - `fig=[`k`]` : the size of the figure is about k cm (or a little larger). $k = 12$ is the default setting.
  - `fig=[`k`,`s`]` or `fig=[`s`]` : s is a string to indicates 10 colors.
    `[`s`]` is `["black,dashed","green,dashed","red,dashed","blue,dashed", "black", "cyan","green","blue","red","magenta"]` by default.
- `["spct"]` : returns the spectral types as a 5×5 matrix
  The (i, j) element id the spectral type of $A_{i,j}$ if $i \neq j$ and the index of rigidity with respect to s_i if $i = j$ $(0 \leq i, j \leq 4)$.
  `dviout=`k is possible.
- `["spct1"]` : returns a 5×6 matrix describing the spectral types together with the spectral type after one-step reduction (if possible). The result is same as in the case `["spct"]` but the 6-th column is added which is formed by the spectral types after one-step reduction.
- `["mult",`ℓ_1`,...,`ℓ_m`]` : Puts $g_0 = g$ and executes this function with $g_i =$`mc2grs(`g_{i-1}`,`ℓ_i`)` for $i = 1, \ldots, m$ and returns g_m. Other options are valid.
- `["eigen",`I`]` : returns the eigenspace decompositions of each eigenspace of A_I by the action of 3 residue matrices commuting with A_I (the list of eigenvalues of A_I and eigenspace decompositions. The eigenspace decomposition is the pair of multiplicities and the eigenvalue of the matrix commuting with A_I).
  The matrices which commutes with A_I are A_J satisfying $J \subset \{0, 1, 2, 3, 4\} \setminus I$ and the result is returned in the lexicographic order of$J$. For example, J are $[1, 3]$, $[1, 4]$ and $[3, 4]$ in the lexicographic order if $I = [0, 2]$.
- `["rest",`I`]` : returns GRS of the boundary equations of x_0 corresponding to the exponents of the singular line $x_i = x_j$ with $I = [i, j]$. The singularities $x_0 = x_k$ are arranged from a smaller k.
- `["rest0",`I`]` : returns spectral types (strings) of the boundary equations of x_0 corresponding to the exponent of the singular line $x_i = x_j$ with $I = [i, j]$.
- `["rest1",`I`]` : returns non-rigid spectral types of the boundary equation of x_0 corresponding to the exponent of the singular line $x_i = x_j$ with $I = [i, j]$.
- When `["eigen"]`, `["rest"]`, `["rest0"]` or `["rest1"]` is indicated, the function returns the list of the above results of all 10 I with I on the each top.
  In this case the option `dviout=1,-1,2` is possible.

```
[0] M=os_md.mc2grs(0,0);
[[[[0,1],[2,3]],[1,0,0]],[[[0,1],[2,4]],[1,0,0]],...
```

```
[1] M1=os_md.mc2grs(M,[[[0,1],a],[[0,2],b],[[0,3],c]]);
[[[[0,1],[2,3]],[1,a,0]],[[[0,1],[2,4]],[1,a,-b]],...
[2] os_md.mc2grs(M1,"deg");     /* Get kappa */
a+b+c
[3] os_md.mc2grs(M1,["homog",[2,3]]);   /* homogenize */
[[[[0,1],[2,3]],[1,a,0]],[[[0,1],[2,4]],[1,a,-b]],...,
 [[[1,4],[2,3]],[1,-a,-a-b-c]]]
[4] F1=os_md.mc2grs(M1,[d]);
[[[[0,1],[2,3]],[1,a+d,0],[1,0,0],[1,0,b+c]],[[[0,1],[2,4]],[1,a+d,-b],[1,0,-b],
[1,0,-a-b-c-d]],...
[5] os_md.mc2grs(F1,"get");   /* All residue matrices */
[[[0,1],[1,a+d],[2,0]],[[0,2],[1,b+d],[2,0]],[[0,3],[1,c+d],[2,0]],[[0,4],
[1,-a-b-c-d],[2,-d]],[[1,2],[2,0],[1,a+b]],[[1,3],[2,0],[1,a+c]],[[1,4],[2,-a],
[1,-a-b-c-d]],[[2,3],[2,0],[1,b+c]],[[2,4],[2,-b],[1,-a-b-c-d]],[[3,4],[2,-c],
[1,-a-b-c-d]]]
[6] os_md.mc2grs(F1,[-d])==M1;  /* middle convolution */
1
[7] os_md.mc2grs(F1,["get",0]); /* GRS for the variable x0 */
[[[0,1],[2,0],[1,a+d]],[[0,2],[2,0],[1,b+d]],[[0,3],[2,0],[1,c+d]],
[[0,4],[2,-d],[1,-a-b-c-d]]]
[8] os_md.mc2grs(F1,["get",0]|dviout=-1);
\begin{Bmatrix}
 A_{01}&A_{02}&A_{03}&A_{04}\\
 [0]_2 & [0]_2 & [0]_2 & [-d]_2\\
 a+d & b+d & c+d & -a-b-c-d
\end{Bmatrix}
[9] os_md.mc2grs(F1,["get",0]|dviout=1); /* GRS for the variable x0 */
```

$$\begin{Bmatrix} A_{01} & A_{02} & A_{03} & A_{04} \\ [0]_2 & [0]_2 & [0]_2 & [-d]_2 \\ a+d & b+d & c+d & -a-b-c-d \end{Bmatrix}$$

```
[10] os_md.mc2grs(F1,"get"|dviout=1)$  /* GRS of KZ equation */
```

$$\begin{Bmatrix} A_{01} & A_{02} & A_{03} & A_{04} & A_{12} & A_{13} & A_{23} & A_{14} & A_{24} & A_{34} \\ [0]_2 & [0]_2 & [0]_2 & [-d]_2 & [0]_2 & [0]_2 & [0]_2 & [-a]_2 & [-b]_2 & [-c]_2 \\ a+d & b+d & c+d & -a-b-c-d & a+b & a+c & b+c & -a-b-c-d & -a-b-c-d & -a-b-c-d \end{Bmatrix}$$

```
[11] os_md.mc2grs(F1,"show"|dviout=1)$  /* simultaneous eigenspaces */
```

$$\begin{aligned}
&[A_{01} : A_{23}] = \{[a+d : 0],\ [0 : 0],\ [0 : b+c]\},\\
&[A_{01} : A_{24}] = \{[a+d : -b],\ [0 : -b],\ [0 : -a-b-c-d]\},\\
&[A_{01} : A_{34}] = \{[a+d : -c],\ [0 : -c],\ [0 : -a-b-c-d]\},\\
&[A_{02} : A_{13}] = \{[b+d : 0],\ [0 : 0],\ [0 : a+c]\},\\
&[A_{02} : A_{14}] = \{[b+d : -a],\ [0 : -a],\ [0 : -a-b-c-d]\},\\
&[A_{02} : A_{34}] = \{[b+d : -c],\ [0 : -c],\ [0 : -a-b-c-d]\},\\
&[A_{03} : A_{12}] = \{[c+d : 0],\ [0 : 0],\ [0 : a+b]\},
\end{aligned}$$

$$[A_{03} : A_{14}] = \{[c+d : -a],\ [0 : -a],\ [0 : -a-b-c-d]\},$$
$$[A_{03} : A_{24}] = \{[c+d : -b],\ [0 : -b],\ [0 : -a-b-c-d]\},$$
$$[A_{04} : A_{12}] = \{[-a-b-c-d : 0],\ [-d : 0],\ [-d : a+b]\},$$
$$[A_{04} : A_{13}] = \{[-a-b-c-d : 0],\ [-d : 0],\ [-d : a+c]\},$$
$$[A_{04} : A_{23}] = \{[-a-b-c-d : 0],\ [-d : 0],\ [-d : b+c]\},$$
$$[A_{12} : A_{34}] = \{[0 : -c],\ [0 : -a-b-c-d],\ [a+b : -c]\},$$
$$[A_{13} : A_{24}] = \{[0 : -b],\ [0 : -a-b-c-d],\ [a+c : -b]\},$$
$$[A_{14} : A_{23}] = \{[-a : 0],\ [-a-b-c-d : 0],\ [-a : b+c]\}$$

```
[12] os_md.mc2grs("322,52,52,43","show0"|dviout=1)$
     /* spectral type of simultaneous eigenspace decomposition */
```

$$2^2 1^3, 1^7, 1^7, 2^2 1^3, 1^7, 1^7, 21^5, 1^7, 1^7, 1^7, 1^7, 1^7, 1^7, 1^7$$

```
[13] os_md.mc2grs(F1,"spct"|dviout=1)$   /* spectral type of KZ equation */
```

	x_0	x_1	x_2	x_3	x_4	idx
x_0		21	21	21	21	2
x_1	21		21	21	21	2
x_2	21	21		21	21	2
x_3	21	21	21		21	2
x_4	21	21	21	21		2

```
[14] os_md.mc2grs("21,21,21,21","get"|dviout=1)$  /* GRS of KZ equation */
```

$$\left\{\begin{matrix} A_{01} & A_{02} & A_{03} & A_{04} & A_{12} & A_{13} & A_{23} & A_{14} & A_{24} & A_{34} \\ [0]_2 & [0]_2 & [0]_2 & [d]_2 & [0]_2 & [0]_2 & [0]_2 & [-a-d]_2 & [-b-d]_2 & [-c-d]_2 \\ a & b & c & -a-b-c-2d & a+b+2d & a+c+2d & b+c+2d & -a-b-c-2d & -a-b-c-2d & -a-b-c-2d \end{matrix}\right\}$$

```
[15] os_md.mc2grs([[[[2,3],[0,4]],[1,a,b]],[[[0,1],[2,3]],[1,c,d]]],"sort");
[[[[0,1],[2,3]],[1,c,d]],[[[0,4],[2,3]],[1,b,a]]]
[16] os_md.mc2grs(F1,["eigen",[0,1]]);  /* decomp. of eigenspaces */
[[a+d,[[1,0]],[[1,-b]],[[1,-c]]],[0,[[1,0],[1,b+c]],[[1,-b],[1,-a-b-c-d]],
[[1,-c],[1,-a-b-c-d]]]]
[17] os_md.mc2grs(F1,["rest",[0,1]]);   /* bry GRS for a sing. line */
[[a+d,[[1,a+b+d]],[[1,a+c+d]],[[1,-2*a-b-c-2*d]]],[0,[[1,a+b+d],[1,0]],[[1,a+c+d],
[1,0]],[[1,-a-b-c-d],[1,-a-d]]]]
[18] os_md.mc2grs(F1,["rest0",[0,1]]);  /* bry spectral types for sing. line */
[[a+d,1,1,1],[0,11,11,11]]
[19] os_md.mc2grs("322,52,52,43","rest0"); /* all bry spectral types  */
[[[0,1],[a,11,11,11],[0,11111,11111,221]],[[0,2],[b,11,11,11],[0,11111,11111,221]],
...
[20] os_md.mc2grs("322,52,52,43","rest1"); /* non-rigid bry spectral types でな
い */
[[[0,1],[0,11111,11111,221]],[[0,2],[0,11111,11111,221]],[[0,3],[2*c,111,111,111],
[0,1111,1111,211]],[[0,4],[2*d1,111,111,111]]]
[21] os_md.mc2grs("322,52,52,43","rest1"|dviout=1); /* non-rigid bry spectral types */
```

$$
\begin{aligned}
&[01] : 221, 1^5, 1^5\\
&[02] : 221, 1^5, 1^5\\
&[03] : 111, 111, 111 \quad 211, 1^4, 1^4\\
&[04] : 111, 111, 111
\end{aligned}
$$

```
[22] os_md.mc2grs("322,52,52,43","rest0"|dviout=1);  /* bry spectral type */
```

$$
\begin{aligned}
&[01] : 11, 11, 11 \quad 221, 1^5, 1^5\\
&[02] : 11, 11, 11 \quad 221, 1^5, 1^5\\
&[03] : 111, 111, 111 \quad 211, 1^4, 1^4\\
&[04] : 11, 11, 11 \quad 11, 11, 11 \quad 111, 111, 111\\
&[12] : 11, 11, 11 \quad 11, 11, 11 \quad 21, 111, 111\\
&[13] : 1, 1, 1 \quad 21, 111, 111 \quad 21, 111, 111\\
&[14] : 1, 1, 1 \quad 1, 1, 1 \quad 1, 1, 1 \quad 11, 11, 11 \quad 11, 11, 11\\
&[23] : 1, 1, 1 \quad 21, 111, 111 \quad 21, 111, 111\\
&[24] : 1, 1, 1 \quad 1, 1, 1 \quad 1, 1, 1 \quad 11, 11, 11 \quad 11, 11, 11\\
&[34] : 1, 1, 1 \quad 1, 1, 1 \quad 1, 1, 1 \quad 11, 11, 11 \quad 11, 11, 11
\end{aligned}
$$

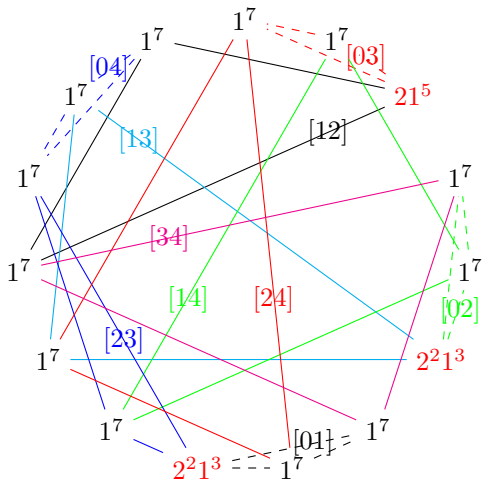
```
[23] os_md.mc2grs("322,52,52,43","rest"|dviout=-1); /* bry Riemann Scheme */
```

$$
\begin{aligned}
[01] : a \quad &\to \begin{Bmatrix} -c-d_1-d_2 & c-d_1 & -a-2c \\ a+b+2c+2d_1+d_2 & a+2c+2d_1 & -a-b-2c-2d_1 \end{Bmatrix}\\
0 \quad &\to \begin{Bmatrix} -c-d_1-d_2 & c-d_1 & [-2c]_2 \\ a+b+2c+2d_1+d_2 & a+2c+2d_1 & [-b-2c-2d_1]_2 \\ 0 & -a+2c-2d_1 & b+4d_1 \\ a+b+4c+4d_1+2d_2 & 0 & \\ -a-b-2c-2d_1-2d_2 & -a-2d_1 & \end{Bmatrix}\\
[02] : b \quad &\to \begin{Bmatrix} -c-d_1-d_2 & -c-d_1 & -b \\ a+b+2c+2d_1+d_2 & b+2d_1 & -a-b-2d_1 \end{Bmatrix}\\
&\dots
\end{aligned}
$$

```
[24] os_md.mc2grs("322,52,52,43","spct1"|dviout=1)$ /* all spectral types */
```

	x_0	x_1	x_2	x_3	x_4	idx	reduction
x_0		52	52	43	322	2	22,22,31,22
x_1	52		322	331	22111	-22	
x_2	52	322		331	22111	-22	
x_3	43	331	331		22111	-24	
x_4	322	22111	22111	22111		-48	

```
[25] os_md.mc2grs("322,52,52,43","show0"|dviout=1,fig=[6]);
```




```
/* Search rigid spectral types whose extension to KZ eq. are identical
   S:spectral type  the spectral types with the same extension
   S:number         the list of all the above spectral types with rank S */
def sameKZ(S)
{
    if(type(S)==1){
        SS=os_md.spgen(S|eq=1,pt=4);
        for(L=[];SS!=[];SS=cdr(SS))
        if((TL=sameKZ(car(SS)))!=0 && TL[0]>TL[1]) L=cons(TL,L);
        return L;
    }
    S=os_md.s2sp(S|std=-1);
    KZ=os_md.mc2grs(S,0);
    Sp=os_md.mc2grs(KZ,"spct");
    for(L=[],I=1;I<5;I++){
        if(Sp[I][I]==2){
            for(S2=[],J=0;J<5;J++) if(I!=J) S2=cons(Sp[I][J],S2);
            S2=os_md.s2sp(S2|std=-1);
            if(S!=S2) L=cons(S2,L);
        }
    }
    if(L==[]) return 0;
    for(LS=[];L!=[];L=cdr(L)) LS=cons(os_md.s2sp(car(L)),LS);
    LS=os_md.lsort(LS,[],"setminus");
    return cons(os_md.s2sp(S),LS);
}

/*  spectral types of rank N whose KZ ext. have idx=K for a variable */
def idxKZ(K,N)
{
    L=os_md.spgen(N|eq=1,pt=4,std=-1);
```

```
    for(LL=[];L!=[];L=cdr(L)){
        KZ=os_md.mc2grs(car(L),0);
        Sp=os_md.mc2grs(KZ,"spct");
        for(I=1;I<5;I++){
            if(Sp[I][I]==K){
                LL=cons(os_md.s2sp(car(L)),LL);
                break;
            }
        }
    }
    return reverse(LL);
}

/* the list with KZ eq. of rank N with a non-rigid restriction on a
   singular hypersurfaces */
def rest1KZ(N)
{
    L=os_md.spgen(N|eq=1,pt=4,std=-1);
    for(LL=[];L!=[];L=cdr(L)){
        KZ=os_md.mc2grs(car(L),0);
        R=os_md.mc2grs(KZ,"rest1");
        for(S=[];R!=[];R=cdr(R)){
            for(T=cdr(car(R));T!=[];T=cdr(T))
                S=cons(os_md.s2sp(car(T)[1]|short=1,std=-1),S);
        }
        if(S!=[]){
            S=os_md.lsort(S,[],1);
            LL=cons([os_md.s2sp(car(L)|short=1),S],LL);
        }
    }
    return reverse(LL);
}

/* non-rigid restricted eq. in the above (opt="basic") */
def rest2KZ(N)
{
    F=(getopt(opt)=="basic")?1:0;
    for(S=[],R=rest1KZ(N);R!=[];R=cdr(R)){
        TR=car(R)[1];
        if(F==1){
                T=os_md.chkspt(TR|opt="basic");
                if(type(T)==4) TR=os_md.s2sp(T|short=1,std=-1);
        }
        S=append(TR,S);
```

```
    return os_md.lsort(S,[],1);
}
/* sorted but same as above with index of rigidity (opt="basic") */
def rest3KZ(N)
{
    R=rest2KZ(N|opt=getopt());
    else R=rest2KZ(N);
    for(S=[];R!=[];R=cdr(R))
      S=cons([os_md.chkspt(car(R)|opt="idx"),car(R)],S);
    return reverse(qsort(S));
}
```

62. `m2mc(`ℓ`,[`a_0`,`a_y`,`a_1`,`c`]|swap=1,small=1,simplify=0,MC=1)`
`m2mc(`ℓ`,`s`|small=1,simplify=0,int=0,swap=`t`)`
:: Addition+middle convolution of Pfaffian system
$du = (A_0\frac{dx}{x} + A_y\frac{d(x-y)}{x-y} + A_1\frac{d(x-1)}{x-1} + B_0\frac{dy}{y} + B_1\frac{d(y-1)}{y-1})u$ with respect to x-variable ($\ell = [A_0, A_y, A_1, B_0, B_1]$).
When ℓ is a spectral type or a Riemann scheme, s =`"GRC"`, `"GRSC"`, `"extend"`, `"Pfaff"`, `"sp"`, `"pairs"`, `"irreducible"`, `"All"`, `"swap"` in the latter case
- `m2mc(0,0)` : Shows usage
- ℓ is a list $[A_0, A_y, A_1, B_0, B_1]$ or vector of 5 square matrices. If the size of the matrix is 1, ℓ can be a list or vector of 5 scalars. Transforms ℓ by additions A_0, A_y, A_1 with the parameters a_0, a_y, a_1, respectively and then transforms by a middle convolution (convolution if `MC=1` is indicated) with the parameter c and returns the vector of the resulting 5 matrices.
  If additions are omitted, `[`a_0`,`a_y`,`a_1`,`c`]` can be simply c.
  If the middle convolution is omitted, `[`a_0`,`a_y`,`a_1`]` can be allowed.
- ℓ can be spectral type of x-variable or Riemann scheme. Spectral type or Riemann scheme is arranged in the order $x=\infty$, $x=0$, $x=y$, $x=1$. In this case a_0 and s are ≠ 0 (returns `[`A_0`,`A_y`,`A_1`,`B_0`,`B_1`]`) or the string given below.
- If ℓ is a spectral type, it may be better to arrange the multiplicities from bigger ones (cf. the option `dep=[`m`,`n`]`).
- If a_0 or s is a string, ite has a special meaning.
  - `GRS` : returns Generalized Riemann scheme. a_y can be `"dviout"`.
  - `GRSC` : same as above but containing generalized exponents for $x=y=0$ and $x=y=1$.
  - `extend` : returns 10 residue matrices corresponding to
    $[A_{x,y}, A_{x,0}, A_{x,1}, A_{x,\infty}, A_{y,0}, A_{y,1}, A_{y,\infty}, A_{0,1}, A_{0,\infty}, A_{1,\infty}]$
    This list of this 10 matrices can be the parameter ℓ.
    If a_0 =`"extend"` and a_1 =`"eigen"`, we have a TEX source of residue matrices, their eigenvalues and eigenvectors.
  - `Pfaff` : returns Pfaffian system by string. a_y can be `"dviout"`.
  - `sp` : spectral types of residue matrices for x-variable and y-variables starting from $x=y$
  - `pairs`, `pair` : returns decompositions in the case when the system is reducible
  - `irreducible` : returns parameters which give the conditions of irreducibility
  - `All` : means that the above all parameters are indicated together with a_1 =`"dviout"`. Moreover if `GRSC` is indicated, this indication is valid in place of `GRS`.
    `operator=0` : do not show the system of Pfaffian form
  - `swap` : returns the result under swapping x-variable and y-variable.
    If `swap=`t is indicated, it means another transformation:.

$t=1$: swaps x-variable and y-variable.
$t=2$: transforms by the correspondence $(x,y)\mapsto(x,\frac{x}{y})$.
$t=$[t_0,t_1,t_2] : $(t_0,t_1,t_2)$is a permutation of the numbers $(0,1,2)$ and $0,1,2$ means singular points $0,1,\infty$, respectively.

Here the residue matrices of the extended KZ equation are

	x	y	0	1	∞
x		$A_{x,y}$	$A_{x,0}$	$A_{x,1}$	$A_{x,\infty}$
y	$A_{x,y}$		$A_{y,0}$	$A_{y,1}$	$A_{y,\infty}$
0	$A_{x,0}$	$A_{y,0}$		$A_{0,1}$	$A_{0,\infty}$
1	$A_{x,1}$	$A_{y,1}$	$A_{0,1}$		$A_{1,\infty}$
∞	$A_{x,\infty}$	$A_{y,\infty}$	$A_{0,\infty}$	$A_{1,\infty}$	

$$\begin{cases} A_0=A_{x,0},\ A_1=A_{x,1},\ A_y=A_{x,y}, \\ B_0=A_{y,0},\ B_1=A_{y,1}, \\ A_0+A_1+A_y+B_0+B_1+A_{0,1}=0, \\ \text{the sum of the elements of a line is 0} \end{cases}$$

$$\begin{cases} A_{x,\infty}=-(A_{x,y}+A_{x,0}+A_{x,1}), \quad A_{y,\infty}=-(A_{x,y}+A_{y,0}+A_{y,1}), \\ A_{0,1}=A_{x,\infty}+A_{y,\infty}+A_{x,y}, \quad A_{0,\infty}=A_{x,y}+A_{x,1}+A_{y,1}, \quad A_{1,\infty}=A_{x,y}+A_{x,0}+A_{y,0}. \end{cases}$$

There are following options.

- `swap=1` : middle convolution after with swapping x and y.
- `small=1` : produces matrices of smaller size by TEX.
- `simplify=0` : does not simplify by an adjoint transformation using a diagonal matrix
- `simplify=1,3` : another simplification by an adjoint transformation using a diagonal matrix
- `dep=`[m,n] : For an automatic generation of spectral exponents from a spectral type the n-th exponent of m-th singular point is determined by Fuchs condition. $m=1$ (and $x=\infty$) and the last exponent
  The first exponents except for the point $x=\infty$ are normalized to 0.
- `int=0` : Allows a fractional multiple of parameters in the automatic generation of characteristic exponents.

```
[0] F1=os_md.m2mc([0,0,0,0,0],[a,b,c,d]);
[ [ a+d b c ]
[ 0 0 0 ]
[ 0 0 0 ] [ 0 0 0 ]
[ a b+d c ]
[ 0 0 0 ] [ 0 0 0 ]
[ 0 0 0 ]
[ a b c+d ] [ b -b 0 ]
[ -a a 0 ]
[ 0 0 0 ] [ 0 0 0 ]
[ 0 c -c ]
[ 0 -b b ] ]
[1] os_md.m2mc(F1,["GRS","dviout"]);
```

$$\left\{\begin{matrix} x=0 & x=y & x=1 & y=0 & y=1 & x=\infty & y=\infty & x=y=\infty \\ a+d & b+d & c+d & a+b & b+c & -a-b-c-d & -a-b-c-d & [-a-b-c-d]_2 \\ [0]_2 & [0]_2 & [0]_2 & [0]_2 & [0]_2 & [-d]_2 & [-b]_2 & -b-d \end{matrix}\right\}$$

```
[2] os_md.m2mc(F1,["Pfaff","dviout"]);
```

$$du = \left(\begin{pmatrix} a+d & b & c \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \frac{dx}{x} + \begin{pmatrix} 0 & 0 & 0 \\ a & b+d & c \\ 0 & 0 & 0 \end{pmatrix} \frac{d(x-y)}{x-y} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ a & b & c+d \end{pmatrix} \frac{d(x-1)}{x-1} \right.$$
$$\left. + \begin{pmatrix} b & -b & 0 \\ -a & a & 0 \\ 0 & 0 & 0 \end{pmatrix} \frac{dy}{y} + \begin{pmatrix} 0 & 0 & 0 \\ 0 & c & -c \\ 0 & -b & b \end{pmatrix} \frac{d(y-1)}{y-1} \right) u$$

```
[3] F2=os_md.m2mc(F1,[-a-d,0,0,e]);
[ [ -a-d+e 0 c 0 ]
[ 0 -a-d+e 0 c+d ]
[ 0 0 0 0 ]
[ 0 0 0 0 ] [ 0 0 0 0 ]
[ 0 0 0 0 ]
[ (-d*a-d*b-d^2)/(c) -d b+d+e c+d ]
[ 0 0 0 0 ] [ 0 0 0 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[ (-d*b)/(c+d) (-d*a-d*c-d^2)/(c+d) (c*b)/(c+d) c+d+e ] [ a+b+d c -c 0 ]
[ 0 0 0 0 ]
[ (d*a+d*b+d^2)/(c) d -d 0 ]
[ 0 0 0 0 ] [ c -c 0 0 ]
[ -b b 0 0 ]
[ 0 0 c -c-d ]
[ 0 0 (-c*b)/(c+d) b ] ]
[4] os_md.m2mc(F2,["GRS","dviout"]);
```

$$\left\{ \begin{matrix} x=0 & x=y & x=1 & y=0 & y=1 & x=\infty & y=\infty & x=y=\infty \\ [-a-d+e]_2 & b+d+e & c+d+e & a+b & [b+c]_2 & [a-e]_2 & -a-b-c-d & a-b-e \\ [0]_2 & [0]_3 & [0]_3 & [0]_3 & [0]_2 & -b-c-e & -b-c-e & [-b-c-e]_3 \\ & & & & & -e & [-b]_2 & \end{matrix} \right\}$$

```
[5] os_md.m2mc(F1,[-a-d,0,0,e]|simplify=1,small=1);
[ [ -a-d+e 0 1 0 ]
[ 0 -a-d+e 0 1 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ] [ 0 0 0 0 ]
[ 0 0 0 0 ]
[ -d*a-d*b-d^2 -d b+d+e 1 ]
[ 0 0 0 0 ] [ 0 0 0 0 ]
[ 0 0 0 0 ]
[ 0 0 0 0 ]
[ -d*c*b -d*a-d*c-d^2 c*b c+d+e ] [ a+b+d 1 -1 0 ]
[ 0 0 0 0 ]
[ d*a+d*b+d^2 d -d 0 ]
[ 0 0 0 0 ] [ c -1 0 0 ]
[ -c*b b 0 0 ]
[ 0 0 c -1 ]
[ 0 0 -c*b b ] ]
[6] os_md.m2mc("21,21,21,21",["GRSC","dviout"]|small=1);
```

$$\left\{\begin{array}{cccccccccc} x=0 & x=y & x=1 & y=0 & y=1 & x=\infty & y=\infty & x=y=\infty & x=y=0 & x=y=1 \\ a & b & c & a+b+2d & b+c+2d & -a-b-c-2d_1 & -a-b-c-2d & [-a-b-c-2d]_2 & [a+b+d]_2 & [b+c+d]_2 \\ [0]_2 & [0]_2 & [0]_2 & [0]_2 & [0]_2 & [d]_2 & [-b-d]_2 & -b & 0 & 0 \end{array}\right\}$$

```
[7] os_md.m2mc(0,0)$
m2mc(m,t) or m2mc(m,[t,s])  Calculation of Pfaff system of two variables
  m : list of 5 residue mat. or GRS/spc for rigid 4 singular points
  t : [a0,ay,a1,c], swap, GRS, GRSC, extend (eigen), sp, irreducible, pair, pairs, Pfaff, All
  s : TeX, dviout, GRSC
  option : swap, small, simplify, operator
  Ex: m2mc("21,21,21,21","All")

[8] os_md.m2mc("21,21,21,21", "All")$
```

Riemann scheme

$$\left\{\begin{array}{cccccccc} x=0 & x=y & x=1 & y=0 & y=1 & x=\infty & y=\infty & x=y=\infty \\ a & b & c & a+b+2d & b+c+2d & -a-b-c-2d & -a-b-c-2d & [-a-b-c-2d]_2 \\ [0]_2 & [0]_2 & [0]_2 & [0]_2 & [0]_2 & [d]_2 & [-b-d]_2 & -b \end{array}\right\}$$

Spectre types : 12,12,12,12 : 12,12,12,12
By the decompositions

$$\begin{aligned} 21,21,21,21 &= 10,10,10,01 \oplus 11,11,11,20 \\ &= 10,10,01,10 \oplus 11,11,20,11 \\ &= 10,01,10,10 \oplus 11,20,11,11 \\ &= 01,10,10,10 \oplus 20,11,11,11 \\ &= 2(10,10,10,10) \oplus 01,01,01,01 \end{aligned}$$

irreducibility $\Leftrightarrow$ $\emptyset = \mathbb{Z} \cap$

$$\{d,\ c+d,\ b+d,\ a+d,\ a+b+c+2d\}$$

The equation in a Pfaff form is

$$\begin{aligned} du = &\left(\begin{pmatrix} a & b+d & c+d \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}\frac{dx}{x}\right. \\ &+\begin{pmatrix} 0 & 0 & 0 \\ a+d & b & c+d \\ 0 & 0 & 0 \end{pmatrix}\frac{d(x-y)}{x-y} \\ &+\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ a+d & b+d & c \end{pmatrix}\frac{d(x-1)}{x-1} \\ &+\begin{pmatrix} b+d & -(b+d) & 0 \\ -(a+d) & a+d & 0 \\ 0 & 0 & 0 \end{pmatrix}\frac{dy}{y} \\ &\left.+\begin{pmatrix} 0 & 0 & 0 \\ 0 & c+d & -(c+d) \\ 0 & -(b+d) & b+d \end{pmatrix}\frac{d(y-1)}{y-1}\right)u \end{aligned}$$

63. `mcmgrs(`g`,`r`|dviout=`k`)`
:: Transform of simultaneous spectral exponents g of a KZ equation of several (≥ 5) points in $\mathbb{P}^1$
The function handles a KZ equation such that the indices of the square matrices $A_{i,j}$ in `mc2grs()` runs from 0 to $N-1$. If we regards the solution of hypergeometric function with several variables,

the number of variables equals $N-2$. The system is

$$\frac{\partial u}{\partial x_i} = \sum_{j\in\{0,1,\dots,N-1\}\setminus\{i\}} \frac{A_{i,j}}{x_i - x_j} u \qquad (i = 0, 1, \dots, N-1),$$
$$A_{i,i} = 0,\ A_{i,j} = A_{j,i},\ A_{i,N} := -(A_{i,0} + A_{i,1} + \cdots + A_{i,N-1}),$$
$$A_{0,i,j} := A_{0,i} + A_{0,j} + A_{i,j}.$$

The parameter g can be the list of simultaneous eigenvalues (multiplicities and eigenvalues) for the commutative pairs $(A_{0,i}, A_{j,k})$ and $(A_{0,i}, A_{0,i,j})$. Here i, j, $k$are different indices in $\{1, \dots, N\}$. The former are $N \times {}_{N-1}C_2$ pairs and the latter are $N \times (N-1)$ pairs and there are $\frac{N^2(N-1)}{2}$ pairs. If g is indicated by a spectral type or the corresponding string, then we have the following.

- If g is a positive integer different from 1, the trivial data of simultaneous eigenvalues of trivial hypergeometric function of g variables. $(N = g + 2)$.
- Suppose g is defined by the spectral types (GRS or a string) of matrices 行列 $A_{0,N}$, $A_{0,1}, A_{0,2}, \dots, A_{0,N-1}$. In this case the parameters with indices are used if `short=0` is indicated.

r determines the following functions.

- `0` : returns the list of lists of simultaneous eigenvalues.
- `["deg"]` : returns κ (`[ ]` can be omitted).
- `[[`i`,`j`],`λ`]` : transforms by the addition $A_{i,j} \mapsto A_{i,j} + \lambda$, $A_{i,N} \mapsto A_{i,N} - \lambda$, $A_{j,N} \mapsto A_{j,N} - \lambda$ $(0 \le i < j < N)$.
- `[`μ`]` : transforms by middle convolution $\mathrm{mc}_{x_0,\mu}$ of x_0 variable (the algorithm is given by [O5]).
- `[[[`i_1`,`j_1`],`λ_1`],...,[`μ_k`],...]` : transforms by successive additions and middle convolutions.
- `[[`a_1`,...,`a_{N-1}`]]` : same as `[[[0,1],`a_1`],...,[[0,N-1],`a_{N-1}`]]`.
- `[[`a_1`,...,`a_{N-1}`,`μ`]]` : same as `[[[0,1],`a_1`],...,[[0,N-1],`a_{N-1}`],[`μ`]]`.
- `["get",[`I`,`J`]]` : returns the list of simultaneous eigenvalues for $[A_I, A_J] = 0$ with the header $[I, J]$.
- `["get",`I`]` : returns eigenvalues of A_I with the header I.
- `["get",`i`]` $(0 \le i \le N)$: returns GRS for A_I with $i \in I$, which is GRS for the variable x_i.
  `dviout=1` : displays the result by using TEX.
  `dviout=-1` : returns the corresponding source file of TEX.
  `dviout=2` : outputs the corresponding source file of TEXwithout display.
- `["get"]` : returns the list of eigenvalues of $\frac{(N+1)N}{2}$ matrices.
  `dviout=`k can be indicated.
- `["get0",[`I`,`J`]]`, `["get0",`I`]`, `["get0",`i`]` $(0 \le i \le N)$, `["get0"]` : `"get0"` are same as `"get"` but the header is omitted.
- `["show"]` : displays simultaneous eigenvalues by using TEX.
  `dviout=-1` can be indicated.
- `["show0"]` : shows the multiplicities of simultaneous eigenvalues of $\frac{N^2(N-1)}{2}$ pairs (the order is same as in `"show"`).
  `dviout=1,-1` can be indicated.
- `["spct"]` : returns spectral types.
  Returns $(N+1) \times (N+1)$ matrix whose (i, j) element is the spectral type of $A_{i,j}$ if $i \ne j$ and the index of rigidity for the variable x_i if $i = j$. のとき $A_{i,j}$ $(0 \le i, j \le N+1)$.
  `dviout=`k can be indicated.
- `["mult",`ℓ_1`,...,`ℓ_m`]` : Puts $g_0 = g$ and defines $g_i =$`mcmgrs(`g_{i-1}`,`ℓ_i`)` for $i = 1, \dots, m$ and returns g_m. Options are possible.

```
[0] os_md.mcmgrs("31,31,31,31,31",["get"]|dviout=1)$
[1] os_md.mcmgrs("31,31,31,31,31",["spct"]|dviout=1)$
[2] os_md.mcmgrs("31,31,31,31,31",["show"]|dviout=1)$
```

give

$$\left\{\begin{array}{cccccccccc} A_{01} & A_{02} & A_{03} & A_{04} & A_{05} & A_{12} & A_{13} & A_{14} & A_{15} \\ [0]_3 & [0]_3 & [0]_3 & [0]_3 & [e]_3 & [0]_3 & [0]_3 & [0]_3 & [-a-e]_3 \\ a & b & c & d & -a-b-c-d-3e & a+b+2e & a+c+2e & a+d+2e & -a-b-c-d-3e \end{array}\right.$$

$$\left.\begin{array}{ccccccc} A_{23} & A_{24} & A_{25} & A_{34} & A_{35} & A_{45} \\ [0]_3 & [0]_3 & [-b-e]_3 & [0]_3 & [-c-e]_3 & [-d-e]_3 \\ b+c+2e & b+d+2e & -a-b-c-d-3e & c+d+2e & -a-b-c-d-3e & -a-b-c-d-3e \end{array}\right\}$$

	x_0	x_1	x_2	x_3	x_4	x_5	idx
x_0		31	31	31	31	31	2
x_1	31		31	31	31	31	2
x_2	31	31		31	31	31	2
x_3	31	31	31		31	31	2
x_4	31	31	31	31		31	2
x_5	31	31	31	31	31		2

$$\begin{aligned} [A_{01} : A_{23}] &= \{[a:0],\, [0:0]_2,\, [0:b+c+2e]\}, \\ [A_{01} : A_{24}] &= \{[a:0],\, [0:0]_2,\, [0:b+d+2e]\}, \\ &\cdots\cdots \\ [A_{01} : A_{012}] &= \{[a:a+b+e],\, [0:0]_2,\, [0:a+b+e]\}, \\ [A_{01} : A_{013}] &= \{[a:a+c+e],\, [0:0]_2,\, [0:a+c+e]\}, \\ &\cdots\cdots \end{aligned}$$

64. `mmc(`l`,[`μ`,`a_1`,...,`a_n`]|full=1,homog=1,mult=`f`)`
:: Addition and middle convolution of ODE of Schlesinger type and a KZ equation
Transforms of the Fuchsian system of Schlesinger type

$$\frac{du}{dx} = \sum_{j=1}^{n} \frac{A_j}{x - x_j} u$$

and the KZ type equation

$$\frac{\partial u}{\partial x_i} = \sum_{j \in \{0,\ldots,n\} \setminus \{i\}} \frac{A_{i,j}}{x_i - x_j} u \qquad (i = 0, \ldots, n)$$

by additions and middle convolutions.

We put $\ell =$`[`$A_1, \ldots, A_n$`]` or $\ell =$`[`$A_{0,1}, \ldots, A_{0,n}, A_{1,2}, \ldots, A_{1,n}, \ldots, A_{n-1,n}$`]` ($\frac{n(n+1)}{2}$ elements) for the former system or latter equation, respectively. If g is a list of rational numbers or functions, they are regarded as matrices of size 1×1.
Moreover ℓ can be a spectral type or GRS. Then for the latter KZ equation ℓ should be rigid and correspond to the differential equation of the variable x_0.

- `[`$\mu, a_1, \ldots, a_n$`]` : Adds a_j to A_j (or $A_{0,j}$) and then transforms the resulting matrices by the middle convolution mc_μ with respect to the variable x (or x_0).
  We may indicate `[`$a_1, \ldots, a_n$`]` or `[`μ`]` by one of the above transformation.
- `mult=`f : $f = 0$ means the former system of Schlesinger type and $f = 1$ means a KZ equation. By default, ℓ corresponds to a system of Schleginger type if the number of elements of ℓ is smaller than 6 and to a KZ system otherwise. If ℓ is a spectral type, ℓ corresponds to a KZ system by default.
- `homog=1` : transforms by the homogeneous middle convolution for a KZ system, namely, middle convolution with respect to μ together with an addition of the last residue matrix by $-\mu$.

65. `linfrac01(`ℓ`|over=1)`

:: List of linear fractional transformations of $x = 0, 1, \infty, y, z, \ldots$ ($\ell = [x, y]$ etc.)
$\ell =$ `[`x`,`y`]` or $\ell =$ `[`x`,`y_1`,...,`y_q`]` and put $\mathbf{x} = (x_0, x_1, \ldots, x_{q+3}) = (x, 0, 1, y_1, \ldots, y_q, \infty)$
($x_0 = x$, $x_1 = 0$, $x_2 = 1$, $x_3 = y_1, \ldots, x_{q+2} = y_q$, $x_{q+3} = \infty$) と置く.

- A linear fractional transformation corresponds to a permutation of $\{0, 1, \ldots, q+3\}$. If $q > 1$ there are $(q+4)!$ permutations. But if $q = 1$, a linear transformation corresponds to a permutation of $\{1, 2, 3\}$. cf. `lft01()`).
  This function returns the list of all linear fractional transformations.
- $\ell =$ `[[`x`,`y`]]` or $\ell =$ `[[`x`,`y_1`,...,`y_q`]]`: returns the list of pairs of linear transformations with the corresponding permutations.

```
[0] os_md.linfrac01([x]);
[[x],[-x+1],[(1)/(x)],[(x-1)/(x)],[(-1)/(x-1)],[(x)/(x-1)]]
[1] os_md.linfrac01([[x]]);
[[[x],[0,1,2,3]],[[-x+1],[0,2,1,3]],[[(1)/(x)],[0,3,2,1]],[[(x-1)/(x)],[0,2,3,1]],
 [[(-1)/(x-1)],[0,3,1,2]],[[(x)/(x-1)],[0,1,3,2]]]
[2] os_md.linfrac([x,y]);
[[x,y],[y,x],[-x+1,-y+1],[(1)/(x),(1)/(y)],[(1)/(x),(y)/(x)],[(y)/(x),(1)/(x)],
 ...
 [(y*x-y)/((y-1)*x),(x-1)/(y-1)],[((y-1)*x)/(y*x-y),(y-1)/(x-1)]]
```

66. `lft01(`ℓ, t`|tr=`τ`)`

:: Linear fractional transformations of $\ell =$ `[x,y]` or $\ell =$ `[`x`,`y_1`,`y_2`,...,`y_q`]`
Let $(0, 1, y_1, y_2, \ldots, y_q, \infty)$ be singular points with respect to x-variable and put $\mathbf{x} = (x_0, x_1, \ldots, x_{q+3}) = (x, 0, 1, y_1, \ldots, y_q, \infty)$
($x_0 = x$, $x_1 = 0$, $x_2 = 1$, $x_3 = y_1, \ldots, x_{q+2} = y_q$, $x_{q+3} = \infty$).

- We consider the KZ equation

$$\frac{\partial u}{\partial x_i} = \sum_{j \in \{0,\ldots,q+2\} \setminus \{i\}} \frac{A_{i,j}}{x_i - x_j} u \quad (0 \le i \le q+2),$$
$$A_{i,j} = A_{j,i}, \qquad \sum_{j \in \{0,\ldots,q+3\} \setminus \{i\}} A_{i,j} = 0 \qquad (0 \le i \le q+3)$$

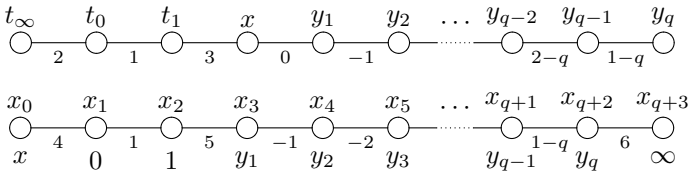
  and its restriction to $x_1 = 0$ and $x_2 = 1$ with variables $(x, y_1, \ldots, y_q)$.
- A linear fractional transformation corresponds to a permutation of$\{0, 1, \ldots, q+3\}$.
- If t is a permutation σ, then the corresponding linear fractional transformation is returned.
- If $\sigma =$ `[`$\sigma_0, \ldots, \sigma_{q+3}$`]`, the singular hypersurface $x_i = x_j$ changes into $x_{\sigma_i} = x_{\sigma_j}$. namely,

$$\frac{\partial u}{\partial x_i} = \sum_{j \in \{0,\ldots,q+2\} \setminus \{i\}} \frac{A_{i,j}}{x_i - x_j} u \quad (0 \le i \le q+2)$$
$$\to \quad \frac{\partial u}{\partial x_i} = \sum_{j \in \{0,\ldots,q+2\} \setminus \{i\}} \frac{A_{i,j}}{x_{\sigma(i)} - x_{\sigma(j)}} u = \sum_{j \in \{0,\ldots,q+2\} \setminus \{i\}} \frac{A_{\sigma^{-1}(i),\sigma^{-1}(j)}}{x_i - x_j} u$$

  σ can be a transposition `[`i`,`j`]` or an identity `1`.
- If $\ell =$ `[[`x`,`y`],1]`, then the corresponding permutation is also returned. Here `1` can be a permutation. Namely,
  `os_md.lft01([[x,y],1],`$\sigma \circ \sigma'$`)` = `os_md.lft01(os_md.lft01([[x,y],1],`σ`),`σ'`)`
- If t is an integer, it corresponds to the following permutation

t	$0<j<q$	transposition	t_∞	t_0	t_1	t_x	t_y	$\frac{(t_x-t_0)(t_1-t_\infty)}{(t_x-t_\infty)(t_1-t_0)}$	$\frac{(t_y-t_0)(t_1-t_\infty)}{(t_y-t_\infty)(t_1-t_0)}$
			∞	0	1	x	y	x	y
$-j$	$y_j \leftrightarrow y_{j+1}$	$(j+2, j+3)$	∞	0	1	x		x	
0	$x \leftrightarrow y_1$	(0,3)	∞	0	1	y	x	y	x
1	$0 \leftrightarrow 1$	(1,2)	∞	1	0	x	y	$1-x$	$1-y$
2	$\infty \leftrightarrow 0$	$(1, q+3)$	0	∞	1	x	y	$\frac{1}{x}$	$\frac{1}{y}$
3	$1 \leftrightarrow x$	(0,2)	∞	x	0	1	y	$\frac{1}{x}$	$\frac{y}{x}$
4	$0 \leftrightarrow x$	(0,1)	∞	0	x	1	y	$\frac{x}{x-1}$	$\frac{x-y}{x-1}$
5	$1 \leftrightarrow y$	(2,3)	∞	0	x	y	1	$\frac{x}{y}$	$\frac{1}{y}$
6	$\infty \leftrightarrow y_q$	$(q+2, q+3)$	y_q	0	x	1		$\frac{x(1-y_q)}{x-y_q}$	$1-y$ $(q=1)$
7	$1 \leftrightarrow \infty$	$(2, q+3)$	1	0	∞	x	y	$\frac{x}{x-1}$	$\frac{y}{y-1}$
8	$x \leftrightarrow \infty$	$(0, q+3)$	x	0	1	∞	y	$1-x$	$\frac{y(x-1)}{x-y}$

$t=0$	$(y_1, x, y_2, y_3, ...)$	$x_0 = x \leftrightarrow x_3 = y_1$
$t=-j$	$(x, y_1, ..., y_{j-1}, y_{j+1}, y_j, y_{j+2}, ...)$	$y_j \leftrightarrow y_{j+1} \quad (1 \le j \le q-1)$
$t=1$	$(1-x, 1-y_1, 1-y_2, 1-y_3, ...)$	$x_1 = t_0 \leftrightarrow x_2 = t_1$
$t=2$	$(1/x, 1/y_1, 1/y_2, 1/y_3, ...)$	$x_1 = t_0 \leftrightarrow x_{q+3} = t_\infty$
$t=3$	$(1/x, y_1/x, y_2/x, y_3/x, ...)$	$x_0 = x \leftrightarrow x_2 = t_1$
$t=4$	$(\frac{x}{x-1}, \frac{x-y_1}{x-1}, \frac{x-y_2}{x-1}, \frac{x-y_3}{x-1}, ...)$	$x_0 = x \leftrightarrow x_1 = t_0$
$t=7$	$(\frac{x}{x-1}, \frac{y_1}{y_1-1}, \frac{y_2}{y_2-1}, \frac{y_3}{y_3-1}, ...)$	$x_2 = t_1 \leftrightarrow x_{q+3} = t_\infty$
$t=8$	$(1-x, \frac{y(x-1)}{x-y}, \frac{y_2(x-1)}{x-y_2}, \frac{y_3(x-1)}{x-y_3}, ...)$	$x_0 = x \leftrightarrow x_{q+3} = t_\infty$




```
[0] for(I=-1;I<9;I++) os_md.mycat([I,os_md.lft01([x,y,z],I|tr=1)]);
-1 [[x,z,y],                                        [0,1,2,4,3,5]]
 0 [[y,x,z],                                        [3,1,2,0,4,5]]
 1 [[-x+1,-y+1,-z+1],                               [0,2,1,3,4,5]]
 2 [[(1)/(x),(1)/(y),(1)/(z)],                      [0,5,2,3,4,1]]
 3 [[(1)/(x),(y)/(x),(z)/(x)],                      [2,1,0,3,4,5]]
 4 [[(x)/(x-1),(x-y)/(x-1),(x-z)/(x-1)],            [1,0,2,3,4,5]]
 5 [[(x)/(y),(1)/(y),(z)/(y)],                      [0,1,3,2,4,5]]
 6 [[((-z+1)*x)/(x-z),((-z+1)*y)/(y-z),-z+1],       [0,1,2,3,5,4]]
 7 [[(x)/(x-1),(y)/(y-1),(z)/(z-1)],                [0,1,5,3,4,2]]
 8 [[-x+1,(y*x-y)/(x-y),(z*x-z)/(x-z)],             [5,1,2,3,4,0]]
[1] os_md.lft01([[x,y],1],1);
[[-x+1,-y+1],[0,2,1,3,4]]
[2] os_md.lft01(@@,2);
[[(-1)/(x-1),(-1)/(y-1)],[0,4,1,3,2]]  /* (0,1,infty) -> (∞,0,infty) */
[3] os_md.lft01([x,y],@@[1]);
[(-1)/(x-1),(-1)/(y-1)]
```

```
[4] os_md.lft01([x,y],[2,4,0,3,1]);
[x,(x)/(y)]
[5] os_md.lft01([x,y],[1,0,4,3,2]);
[x,(-x+y)/(y-1)]
[6] os_md.lft01([x,y],[4,2,1,3,0]);
[x,((-y+1)*x)/(x-y)]
```

3.1.3 Some operators

67. `okubo3e(`$[p_{0,1},\ldots,p_{0,m}]$`,`$[p_{1,1},\ldots,p_{1,n}]$`,`$[p_{2,1},\ldots,p_{2,m+n}]$`|opt=1])`
68. `fuchs3e(`$[p_{0,1},\ldots,p_{0,n}]$`,`$[p_{1,1},\ldots,p_{1,n}]$`,`$[p_{2,1},\ldots,p_{2,n}]$`)`
69. `ghg(`$[p_{1,1},p_{1,2},...,p_{1,m}]$`,`$[p_{2,1},p_{2,2},...,p_{2,n}]$`)`
:: differential operator satisfied by generalized hypergeometric function ${}_mF_n(p_1;p_2;x)$
(cf. `seriesHG()`)

$$P(x,\tfrac{d}{dx}) = \prod_{j=1}^{n}(x\tfrac{d}{dx}+p_{2,j})\cdot\tfrac{d}{dx} - \prod_{j=1}^{m}(x\tfrac{d}{dx}+p_{1,j})$$

$${}_mF_n(p_{1,1},\ldots,p_{1.m};p_{2,1},\ldots,p_{2,n};x) = \sum_{n=0}^{\infty}\frac{\prod_{\nu=1}^{m}(p_{1,\nu})_n}{\prod_{\nu=1}^{n}(p_{2,\nu})_n}\frac{x^n}{n!}$$

`x` is the variable and `dx` is differential
If $m=n+1$, the operator is rigid and Fuchsian and

$$\wp\left\{\begin{array}{lccccc} 0: & 0 & 1-p_{2,1} & \cdots & 1-p_{2,m-2} & 1-p_{2,m-1} \\ 1: & 0 & 1 & \cdots & m-2 & -\gamma \\ \infty: & p_{1,1} & p_{1,2} & \ldots\ldots\ldots\ldots & & p_{1,m} \end{array};x\right\}$$

Fuchs relation : $\gamma = \sum p_{1,\nu} - \sum p_{2,\nu}$
When $m=2$ and $n=1$, the operator corresponds to Gauss hypergeometric equation

```
[0] os_md.ghg([a,b],[c]);
(-x^2+x)*dx^2+((-a-b-1)*x+c)*dx-b*a
```

70. `even4e(`$[p_{1,1},p_{1,2},p_{1,3},p_{1,4}]$`,`$[p_{2,1},p_{2,2}]$`)`
:: Even family of order 4 (Rigid)

$$\wp\left\{\begin{array}{lccccc} 0: & 0 & 1 & p_{2,1} & p_{2,2} & \\ 1: & 0 & 1 & p_0 & p_0+1 & ;x \\ \infty: & p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} & \end{array}\right\}$$

Fuchs relation : $2p_0+p_{1,1}+p_{1,2}+p_{1,3}+p_{1,4}+p_{2,1}+p_{1,2}=3$
71. `odd5e(`$[p_{1,1},p_{1,2},p_{1,3},p_{1,4},p_{1,5}]$`,`$[p_{2,1},p_{2,2}]$`)`
:: Odd family of order 5 (Rigid)

$$\wp\left\{\begin{array}{lccccc} 0: & 0 & 1 & p_{2,1} & p_{2,2} & p_{2,2}+1 \\ 1: & 0 & 1 & 2 & p_0 & p_0+1 \\ \infty: & p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} & p_{1,5} \end{array};x\right\}$$

Fuchs relation $2p_0+p_{2,1}+2p_{2,2}+\sum p_{1,j}=4$
72. `rigid211(`$[p_{0,1},p_{0,2}]$`,`$[p_{1,1},p_{1,2}]$`,`$[q_0,q_1]$`)`
:: Type 211,211,211

$$\wp\left\{\begin{array}{lccccc} 0: & 0 & 1 & p_{0,1} & p_{0,2} & \\ 1: & 0 & 1 & p_{1,1} & p_{1,2} & ;x \\ \infty: & q_0 & q_0+1 & q_1 & q_2 & \end{array}\right\}$$

Fuchs relation : $p_{0,1}+p_{0,2}+p_{1,1}+p_{1,2}+2q_0+q_1+q_2=4$
73. `extra6e(`$[p_{1,1},p_{1,2},p_{1,3},p_{1,4},p_{1,5},p_{1,6}]$`,`$[p_{2,1},p_{2,2}]$`)`

:: Extra case (Rigid)

$$\wp\left\{\begin{array}{lccccccc} 0: & 0 & 1 & p_{2,1} & p_{2,1}+1 & p_{2,2} & p_{2,2}+1 & \\ 1: & 0 & 1 & 2 & 3 & p_0 & p_0+1 & ;x \\ \infty: & p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} & p_{1,5} & p_{2,6} & \end{array}\right\}$$

Fuchs relation : $2p_0 + 2p_{2,1} + 2p_{2,2} + \sum p_{1,j} = 5$

74. `eofamily([`$p_{0,1}, p_{0,2}$`],[`$p_{1,1}$`],[`$p_{2,1}, \ldots, p_{2,n}$`])`
:: Even/odd family (obsolete)
75. `ev4s(`p_1`,`p_2`,`p_3`,`p_4`,`p_5`)`
:: Rigid restriction of Heckman-Opdam's hypergeometric equation of type (BC_2, BC_1)

```
[0] P=os_md.ev4s(a,b,c,d,e);
(x^4-2*x^3+x^2)*dx^4+((-2*a+2*b+8)*x^3+(4*a-2*b-13)*x^2+(-2*
...
[1] os_md.expat(P,x,0);
[a-c+1/2,a+c-1/2,1,0]
[2] os_md.expat(P,x,1);
[-b+1/2,-b+3/2,1,0]
[3] os_md.chkexp(P,x,1,-b+1/2,2);
[]
[4] os_md.expat(P,x,"infty");
[-1/2*a+1/2*b-1/2*d+1/2,-1/2*a+1/2*b+1/2*d+1/2,-1/2*a+1/2*b-1/2*e+1/2,
-1/2*a+1/2*b+1/2*e+1/2]
```

76. `b2e(`p_1`,`p_2`,`p_3`,`p_4`,`p_5`)`
:: Non-rigid restriction of Heckman-Opdam's hypergeometric equation of type (BC_2, A_1)

```
[0] P = os_md.b2e(a,b,c,d,e);
(x^4-2*x^3+x^2)*dx^4+((-4*c+10)*x^3+(6*c-15)*x^2+(-2*c+5)*x
...
[1] os_md.expat(P,x,0);
[-a+c+1/2,a+c-1/2,1,0]
[2] os_md.expat(P,x,1);
[-b+c+1/2,b+c-1/2,1,0]
[3] os_md.expat(P,x,"infty");
[-c-1/2*d-1/2*e+1,-c-1/2*d+1/2*e+1,-c+1/2*d-1/2*e+1,-c+1/2*d+1/2*e+1]
```

77. `heun([`a, b, c, d, e`],`p`,`r`|)`
:: Heun's equation. r is an accessory parameter

$$\wp\left\{\begin{array}{lccc} 0: & 0 & c & \\ 1: & 0 & d & ;\ x \\ p: & 0 & e & \\ \infty: & a & b & \end{array}\right\} \qquad \text{Fuchs relation : } a+b+1 = c+d+e$$

```
[0] os_md.heun([a,b,c,d,"?"],p,r);
(x^3+(-p-1)*x^2+p*x)*dx^2+((a+b+1)*x^2+((-c-d)*p-a-b+d-1)*x+c*p)*dx+b*a*x-b*a*r
```

3.2 Useful functions

The following functions are in a module and we call the functions by putting `os_md.` at the head such as `os_md.myhelp()`.

3.2.1 Extended function

78. `myhelp(`h`)`
:: Displays the manual of `os_muldif.rr`
- `os_muldif.pdf`, `os_muldif.dvi` should be in `get_rootdir()\help`. They are displayed by $h = 1$ or -1, respectively.
- Suppose DVIOUTH is correctly set. Then the explanation of the function is shown by indicating its name h by `myhelp(`h`)` (if it is contained in `os_muldif.dvi`). Here h is a string such as `myhelp("m2mc")` removed the top `os_md.`.
- When $h = 0$, `os_md.getbygrs`, `os_md.m2mc` or `os_md.mgen`, the corresponding explanation is displayed.
- $h = [dviout, n]$: Sets the environment to this display. *dviout* is the pathname of `dviout` and n is the number of `dviout`. Here $n = 2$ is recommended since the first `dviout` is used to display equations by `os_muldif.rr`.
- DVIOUTH may be used under other environment or more general settings.

79. `chkfun(`f`,`s`)`
:: Checks whether f (= a string) is defined and `load(`s`)` if it is not
- $f = 0$: Returns version of `Risa/Asir`
- $f = 1$: Returns version of `os_muldif.rr`
- $s = 0$: Checks whether the function with the name f is defined when f is a string.
- f can not be a function in a module

```
[0] os_md.chkfun(0,0)$
Risa/Asir Ver. 20121217
[1] os_md.chkfun(1,0)$
Loaded os_muldif Ver. 00140401 (Toshio Oshima)
```

80. `isMs()`
:: Checks whether the operating system is Microsoft Windows
Checks the environment variable `temp` or `tmp` and returns 1 if the 3rd character of the value is ¥ or \ and 0 otherwise.

81. `isyes(`p`|set=`ℓ`)`
:: Defines a function returning 0 or 1
- `set=1` : Defines a function. p is a list whose elements are a function and the list of parameters of the functions and the range of the value returned by the function which means as follows.
  Namely, when $p = [fn, [t_1, t_2, \ldots], [a, b]]$. `isyes(X|set=`$p$`)` means the function
  ```
  def foo(X){
    R=fn(X,t1,t2,...);
    return (R>=a && R<=b)?1:0;
  }
  ```
  – Suppose $p =$`[type,[],[0,1]]`. Then the function is defined which returns 1 (yes) or 0 (no) if `type(`p`)` $\in [0, 1]$ and 0 otherwise. Here $-\infty$ or ∞ is indicated by `""`.
  – If the range is a value, then the value can be indicated. Namely, `[type,[],1]` is equivalent to `[type,[],[1,1]]`.
  – If several conditions are indicated by a list of lists reoresenting a condition, then the

function defined returns 1 only if all the conditions are "yes".
  - p=`[[os_md.isint,[],1],[os_md.calc,[[">",0]],1]]` means that the function defined returns 1 only if the given parameter is a positive integer.
  - The function defined is valid until a new function is defined by `set=1`.
- `set=0` : Returns the function defined (see the following example `[11]`).
- `set=` : If the list for judge is indicated by `set=` and p is an object to be judged, then the value 1 or 0 is returned with keeping the already defined function.
- If `set=` is not indicated, p is judged and 1 or 0 is returned according to the judge.

```
[0] os_md.isyes([[os_md.isint,[],1],[os_md.calc,[[">",0]],1]]|set=1)$
[1] os_md.isyes(0|set=0);
[[os_md.isint,[],1],[os_md.calc,[[>,0]],1]]
[2] os_md.isyes(3/2);
0
[3] os_md.isyes(@i);
0
[4] os_md.isyes(-1);
0
[5] os_md.isyes(3/2);
0
[6] os_md.isyes(3);
1
[7] A=mat([1,2],[3,4]);
[ 1 2 ]
[ 3 4 ]
[8] os_md.isall(os_md.isyes,A);
1
[9] A[1][1]=0$
[10] os_md.isall(os_md.isyes,A);
0
[11] def isPositiveInt(X){
  P=os_md.isyes(0|set=0);
  os_md.isyes([[os_md.isint,[],1],[os_md.calc,[[">",0]],1]]|set=1);
  V=os_md.isyes(X);
  os_md.isyes(P|set=1);
  return V;
}$
[12] isPositiveInt(3/2);
0
```

The above `[11]` is same as the following function

```
def isPositiveInt(X){
 return os_md.isyes(X|set=[[os_md.isint,[],1],[os_md.calc,[[">",0]],1]]);
}$
```

82. `isall(`f`,`m`)`

:: Returns 0 if m contains an element satisfying $f(p) = 0$ and 1 otherwise
If m is not a list, a vector nor a matrix, m is replaced by `[`m`]`.
See an example of `isyes()`.

```
[0] os_md.isall(os_md.isint,[0,2,3,1/2]);
0
[1] os_md.isall(os_md.isint,[0,2,3,4]);
1
[3] os_md.isall(os_md.isint,[0,2,3,[1]]);
0
[4] os_md.isall(os_md.isint,mat([0,2,3,4]));
1
```

83. `ptype(`p`,`ℓ`)`
:: Returns `type()` regarding ℓ as the variable or ℓ as the list of variables
- `ptype(`p`,vars(`p`))` corresponds to `type(`p`)`.
- If p is a rational function and its denominator (resp. numerator) contains an elementary function which is not a polynomial, then 128 (resp. 64) is added to the number returned.

```
[0] P=(x+y)^2/a;
(x^2+2*y*x+y^2)/(a)
[1] os_md.ptype(P,x);
2
[2] os_md.ptype(P,a);
3
[3] os_md.ptype(P,z);
1
[4] os_md.ptype(P,[x,y]);
2
[5] os_md.ptype(P,[x,y,a]);
3
[6] os_md.ptype([1,2],[x,y]);
4
[7] os_md.ptype(sin(x)+sin(y)+@pi,x);
65
[8] os_md.ptype(sin(x)+x,x);
66
[8] os_md.ptype(sin(x)/x,x);
67
[9] os_md.ptype(1/sin(x),x);
129
[10] os_md.ptype(sin(x)/(x+cos(x)),x);
195
```

84. `getline(`Id`|Max=`m`,CR=[`r_1`,`r_2`,..],LF=[`l_1`,`l_2`,...])`
:: An extension of `get_line()`
Gets a line form a text file (deleted the last line feed).

- m : maximal length in bytes for a line (1023 by default).
- r_1, r_2,...: codes to be ignored (`[13]` by default),
- l_1, l_2,...: codes indicating the end of a line (`[10]` by default).

85. `keyin(`s`)`
:: Shows s and waits a line input from keyboard and returns it
Returns a line input deleting the last linefeed
`mycat0()` is used to show s and therefore numbers or a list is allowed.

```
[0] S=os_md.keyin("Input? ")$
Input? This is a pen.
[1] S;
This is a pen.
```

86. `showbyshell(`s`)`
:: Executes s by shell and show the standard output in `Risa/Asir`

```
[0] os_md.showbyshell("echo %temp%")$
[1] os_md.showbyshell("dir c:\\")$
```

87. `getbyshell(`s`)`
:: Executes s by shell and puts the standard output to a file
- If `Id`is returned, `Id` ≥ 0 means that the output is obtained and it can be read `get_line(Id)` by lines. If we use `get_byte(Id)`, we get by bites.
- We get 0 by `getline(Id)` after tle last line. We need `getbyshell(Id)` or `close_file(Id)` before we call this function again.
- A temporary file is made in the directory indicated by the environment variable `%temp%` or `%tmp%` or the value `DIROUT`. A new file is created after erasing the former file.

88. `fcat(`f`,`s`|exe=1)`
:: Outputs s in a file f
- Outputs `print(`s`)` in the file f. If f exists, the output is appended in the file by default.
The command `remove_file(`f`)` means to delete the file f.
In the followings, the file is in the directory indicated by `DIROUT`.
- $f = 0$: Appends `print(`s`)` in the file `fcat.txt`.
- $f = 1$: Outputs `print(`s`)` to the file `fcat.txt`.
- $f = 2, \ldots, 9$: Outputs to the file `fcat`f`.txt`.
- $f = -1$: Returns the default filename.
- `exe=1` : Outputs `print(`s`)` and executes the related program (usually an editor).

89. `makev([`ℓ_1`,`ℓ_2`,...]|num=1)`
:: Makes a variable combining $\ell_1, \ell_2 \ldots$
- ℓ_i may be a variable, a string or a non-negative integer
- The integers $10, 11, \ldots$ are understood as a letter $a, b, \ldots$ if `num=1` is not indicated.

```
[0] os_md.makev(["a_", 0, 1]);
a_01
[1] os_md.makev([a, 0, 1]);
a01
[2] os_md.makev([a, 0, b]);
a0b
[3] os_md.makev([a, 10, b]);
aab
```

```
[4] os_md.makev([a,10,"_",3]|num=1);
a10_3
[5] os_md.my_tex_form(@@);
a_{10,3}
```

90. `shortv(`p`,[`v_1`,`v_2`,...]|top=`w`)`
:: Changes the variables $v_1, \ldots$ with indices contained in p into valiables with one letter
When the name v_i is a letter followed by numbers starting from 0 or 1, then v_i are changed to variables starting from w (`a` by default) up to `z` excluded the variables already used.

```
[0] P=[a1,a2,b,c0,c1]$
[1] os_md.shortv(P,[a,c]);
[a,c,b,d,e]
[2] os_md.shortv(P,[c,a]);
[d,e,b,a,c]
[3] os_md.shortv(P,[a,c]|top=x);
[x,y,b,z,c1]
```

91. `makenewv(`ℓ`|var=`v`,num=`n`)`
:: Generates a new variable which is not used in ℓ
- Returns a variable from `z_0`, `z_1`, ...,`z_9`, `z_10`, ... which is not contained in ℓ.
- The variable `z_` to be returned is changed by `var=`v. Here v is a variable or a string.
- The variables contained in functions in ℓ are checked.
- `num=`n : New n variables are returned.

```
[0] os_md.makenewv(0);
z_0
[1] os_md.makenewv([z_0+z_1,z_2]);
z_3
[2] os_md.makenewv([z_0+z_1,z_2]|var=x);
x0
[3] os_md.makenewv([z_0,z_1]|num=3);
[z_2,z_3,z_4]
[4] os_md.makenewv(z_2*sin(cos(z_0+z_1))+z_3);
z_4
```

92. `isvar(`p`)`
:: Is p a variable?
Returns 1 or 0.

```
[0] os_md.isvar(dx);
1
[1] os_md.isvar(-dx);
0
[2] os_md.isvar(1);
0
```

93. `varargs(`p`|all=`t`)`
:: Returns elementary functions and variables in p
- p can be matrix or list.

- Returns the list of elementary functions and their variables in p.
- The variables contained in elementary functions appeared in variables of functions in p are considered.
- `all=1` : Returns all elementary functions and variables
- `all=2` : Returns all variables.

```
[0] os_md.varargs(x*sin(y)+z+cos(s-t));
[[sin,cos],[y,s,t]]
[1] os_md.varargs(x*sin(y)+z+cos(s-t)|all=1);
[[sin,cos],[x,z,y,s,t]]
[2] os_md.varargs(x*sin(y)+z+cos(s-t)|all=2);
[x,z,y,s,t]
[3] os_md.varargs(sin(cos(x)*sin(y)+@pi+z));
[[sin,cos],[x,y,z]]
```

94. `pfargs(`p`,`x`|level=`t`)`
:: Returns all functions and variables containing variable x
- p can be a matrix or a list.
- Returns the list of elementary functions and their variables in p.
- Elementary functions in an argument of elementary function are considered.
- `level=`t : $t = 1$ means that the argument of a function is ignored. $t = 2$ means only the variables in the argument of a function. t means the depth of the function.

```
[0] os_md.pfargs(log(log(log(x))),x);
[[log(log(log(x))),log,log(log(x))],[log(x),log,x],[log(log(x)),log,log(x)]]
[1] os_md.pfargs(sin(cos(x)*cos(y)+@pi+z),x);
[[sin(z+cos(y)*cos(x)+@pi),sin,z+cos(y)*cos(x)+@pi],[cos(x),cos,x]]
[2] os_md.pfargs(log(log(log(x))),x|level=1);
[[log(log(log(x))),log,log(log(x))]]
[3] os_md.pfargs(log(log(log(x))),x|level=2);
[[log(log(x)),log,log(x)]]
[4] os_md.pfargs(log(log(log(x))),x|level=3);
[[log(x),log,x]]
```

95. `isdif(`p`)`
:: Returns a list of the pairs of variable and its derivatives
p is considered a differential operator if
- p is a polynimial or a rational form
- p contains a variable whose first character is 'd' and 2nd chracter is a small alphabetical letter and p is a polynomial with respect to the variable.

```
[0] os_md.isdif((x+dx+dx1+dy+dz)^2/z);
[[x1,dx1],[x,dx],[y,dy],[z,dz]]
[1] os_md.isdif((x+dx+dx1+dy+dz)^2/dz);
0
```

96. `mysubst(`r`,[`v_1`,`r_1`]|inv=1)` `mysubst(`r`,[[`v_1`,`r_1`],...]|inv=1)`
`mysubst(`r`,[`ℓ_1`,`ℓ_2`]|lpair=1,inv=1)`
:: Same as `subst(`r`,`v_1`,`r_1`,...)`. Useful if r is complicated and r is a rational form
- r may be a rational form, a list, a vector or a matrix containing rational forms. It may contains

a strings as an element, which is not changed.

- When r_j are variables, `inv=1` can be indicated which means the inverse substitution (by the transposition of v_j and r_j).
- `inv=1` : inverse substitution
- If the number of elements of ℓ_1 is larger that 2, `lpair=1` is indicated.

97. `myswap(`p`,[`x_1`,`x_2`,...,`x_n`])`
:: Cyclic permutation $(x_1, x_2, \ldots, x_n)$ of a (list of) rational form(s)
$n = 2$ means the transposition of two variables.

```
[0] os_md.myswap([x,y,z,w],[x,y,z]);
[y,z,x,w]
[1] os_md.myswap([x,y,z,w],[x,z]);
[z,y,x,w]
```

98. `mulsubst(`r`,[[`$p_{1,0}$`,`$p_{1,1}$`],[`$p_{2,0}$`,`$p_{2,1}$`],...]|inv=1,lpair=1)`
:: Simultaneous substitutions
    - Retruns the imultaneous substitution $p_{j,0} \mapsto p_{j,1} \quad (j = 1, 2, \ldots)$.
    - `[[x,y],[y,x]]` : transposition of `x` and `y`.
    - `lpair=1` : the variables in ℓ_1 are simultaneously replaced by the elements of ℓ_2, respectively.

```
[0] os_md.mulsubst(x+y^2+z^3,[[x,y],[y,x]]);
x^2+y+z^3
[1] os_md.mysubst(x+y^2+z^3,[[x,y],[y,x]]);
x^2+x+z^3
[2] os_md.mysubst(["top",x,y],[x,2]);
[top,2,y]
[3] subst(["top",x,y],[x,2]);
subst : invalid argument
return to toplevel
[4] os_md.mulsubst([x,y,z],[x,w]);
[w,y,z]
[5] os_md.mulsubst([x,y,z],[x,w]|inv=1);
[x,y,z]
```

99. `fmult(`f`,`m`,`ℓ`,`n`|`$\cdots$`)`
:: Returns $m_{length(l)}$ by $m_i \mapsto m_{i+1} = f($`(`m_i`,`$\ell[i]$`,`$n[0], n[1], \ldots$`|`$\cdots$`)` $(m_0 = m)$
    - ℓ and n are lists.
    - Options are indicated to functions.

100. `mtransbys(`f`,`m`,`ℓ`|`$\cdots$`)`
:: Extends a function $f(x)$ of scalar x to a function of a list, vector or matrix
    - The elements $m_n u$ of m are transformed to $f(m_\nu, l[0], l[1], \ldots |\cdots)$.
    - The list/vector/matrix mae be nested.
    - The arguments are indicated by the list ℓ.
    - Option are put to the function.
    - This function is similar to `map(`f`,`m`,`$l[0], \ldots$`)` but `map()` cannot be nested and options are not indicated.

```
[0] A=newmat(2,2,[[(x^2-y^2)/(x+y),x/y],
    [yx/x^2,(x^2-y^2)/(x-y)]]);
```

```
[ (x^2-y^2)/(x+y) (x)/(y) ]
[ (y*x)/(x^2) (x^2-y^2)/(x-y) ]
[1] os_md.mtransbys(red,A,[]);
[ x-y (x)/(y) ]
[ (y)/(x) x+y ]
[3] os_md.mtransbys(os_md.abs,[[1,-2],[3,-4]],[]);
[[1,2],[3,4]]
[4] map(os_md.abs,[[1,-2],[3,-4]]);
[[1,-2],[3,-4]]
```

101. `mmulbys(`f`,`m`,`n`,`ℓ`|`$\cdots$`)`
:: Extends a pair of arguments of f to a a pair of matrices
   - ℓ is a list of options. m is a argument of f or a matrices of the rarguments of f.
   - If m and n are matrices, the natrices
     $\left(\sum_\nu f(m_{i,\nu}, n_{\nu,j}, \ell[0], \ell[1], \ldots)\right)_{i,j}$
     is returned.
   - If m or n is matrices, it is considered as a scalar matrix.
   - If m (redp. n) is a vector, it is considered as a line (resp. column) vector.
     If m and n are vectors, a scalar is returned.
   - Options are valid,

102. `cmpsimple(`p`,`q`|comp=`t`)`
:: Copmares the simplicities of p and q
If p is simpler than q, a negative integer is returned and if q is simpler than p, a posirive number is returnd.
Compare
   - `iand(`t`,1)!=0` : numbers of variables
   - `iand(`t`,2)!=0` : numbers of monomials
   - `iand(`t`,4)!=0` : length of their expression
   - compares p and p in `Risa/Asir`

in this order. Here $t = 7$ by default.

```
[0] os_md.cmpsimple((x+y)^2,(x+1)^3|comp=1);  /* # of variables */
1
[1] os_md.cmpsimple((x+y)^2,(x+1)^3|comp=2);  /* # of monomials */
-1
[2] os_md.cmpsimple((x+y+z)^2,(sin(x)+cos(x))^2)|comp=4); /* length of expression */
-4                                            /* length of expression */
[3] os_md.cmpsimple(1,-1);
-1                                            /* length of expression */
[4] os_md.cmpsimple(4,2);
1                                             /* compare in Risa/Asir */
```

103. `simplify(`p`,`ℓ`,`t`|var=[`$x_1, x_2, \ldots$`])`
:: Simplifies p by using linear relations indicated by ℓ
   - $\ell = [\ell_0, \ell_1]$: Applies `subst(*,`ℓ_0, ℓ_1`)` to (each element of) p and uses this subsititution if the the result is simpler ($t = 1 \sim 7$).
   - $\ell = [\ell_1]$; If ℓ_1 is a polynomial and ℓ_1 contains a variable with just degree 1, we applies the above simplification by this linear relation.

- If there many linear relations, ℓ is indicated the list of linear relations.
- Compares the simplicity of the substitution checking
  `iand`(t,`1)!=0` : number of variables
  `iand(`t`,2)!=0` : number of monomials , `iand(`t`,4)!=0` : length of expression
  in this order and if the subsitution is simpler, returns the substitution
- If the substitution is same as original in the above check, check the order in `|`Risa/Asir—.
- 複数の置き換えを試すときは, $\ell = [[\ell_{00}, \ell_{01}], [\ell_{10}, \ell_{11}], \ldots]$ とする.
- `var=[`$x_1, x_2, \ldots$`]` : If p is a rational form of variables $x_1, x_2, \ldots$, the numerator and denominator are simplified as polynomials.

```
[0] os_md.simplify([x+y+z+a+b,x+z+a+b],[a,-x-y-w/2-c],1);
[z-1/2*w+b-c,x+z+a+b]
[1] os_md.simplify([x+y+z+a+b,x+z+a+b],[a,-x-y-w/2-c],4);
[x+y+z+a+b,x+z+a+b]
[1] os_md.simplify(x+3*y+4*z,[x+y+z],1);
2*y+3*z
[2] os_md.simplify(x+3*y+4*z,[x+y+z],4);
-3*x-y
```

104. `getel(`m`,`i`)`
:: Returns $m[i]$ if m is a list/vector/matrix and i is non-negative integer
If i is a list $[i_1, i_2]$ and m is a matrix, then $m[i_1][i_2]$ is returned

Returns m in the remaining cases.

105. `evalred(`r`|opt=[[`s_1`,`t_1`],[`s_2`,`t_2`]...])`
:: Replaces `sin(0)`, `cos(0)`, `exp(0)` by 0, 1, 1 etc
- Replace the following expression by exact values
  `sin(`k`@pi)`, `cos(`k`@pi)` etc. with the value in $0, \pm 1, \pm\frac{1}{2}$, `tan(0)`, `asin(0)`, `atan(0)`, `sinh(0)`, `cosh(0)`, `tanh(0)`, `exp(0)`, `log(1)`, `pow(1,`x`)`, `1^`x
- Adds the rule which means s_j are replaced by t_j $(j = 1, 2, \ldots)$. We may indicate `opt=[`s`,`t`]` if there is only one rule to be added.

```
[0] evalred(exp(sin(0)));
1
[1] eval(exp(sin(0)));
1.0000000000000000000000000000
[2] os_md.evalred(exp(sin(@pi)));
1
[3] eval(exp(sin(@pi)));
1.0000000000000000000000000000
```

3.2.2 Numbers

3.2.3 Polynomials and rational functions

191. `fctrtos(`r`|var=`ℓ`,rev=1,dic=1,TeX=`f`,dviout=1,lim=`n`,small=1,pages=1,add=`s`)`
:: Transforms a rational function r to factorized strings
- `TeX=1` : Returns a source in LaTeX. If s is not a polynomial, the list of the numerator and denominator is returned.
- `TeX=2, 3` : Returns a source in LaTeX. If s is not a polynomial, `\frac{ }{ }` is used but it is

too long for a line `\Bigl( \Bigr)\bigm/\Bigl( \Bigr)` is used with necessary new lines as follows.

- `var=`x : Returns as polynomials of x (for numerator/denominator) with the factorized coefficients. If `TeX=2` (resp. `TeX=3`) is indicated, `\\` (resp. `\\ &`) is used after the powers of x etc. considering the width of a line.
- `var=[`x,s`]` : Transforms the variable x to a string s.
- `var=[`$x,y,\ldots$`]` : Regards r as a polynomial of $(x, y, \ldots)$.
- `var=[[`x,s`],[`y,t`],...]` : As above and $x, y, \ldots$ are transformed by strings $s, t, \ldots$, respectively.
- `rev=1` : Monomials are arranged from smaller degree.
- `dic=1` : Monomials are arranged from bigger degree in the lexicographic order.
- `dic=2` : Same as above but the product of variables is arranged in the reverse order.
- `var="dif0"` : The variable starting `d` followed by a lower case alphabet is regarded as a differential of the variable omitted the top `d`. The option `TeX` is indicted, the differential is expressed by ∂_{x_1} using ∂.
- `var="dif"` : Same as above. But if there is only one variable to be differentiated, the differential is simply expressed by ∂. If there are many such variables and all the names of variables to be differentiated are between `x0` と `x99`, then the differentials are expressed as $\partial_0, \ldots, \partial_{99}$.
- `var="dif1"` : Same as above but differentials are expressed as $\frac{d^2}{dx^2}$ or $\frac{\partial^3}{\partial x^2 \partial y}$.
- `var="dif2"` : Same as above but the partial differential is used even if there is only one variable to be differentiated.
- `lim=`n : If r is a polynomial, the width of each line is arranged so that it does not exceed the width of n characters. The string `\\` or `\\ &` are used in cases `TeX=2` or `TeX=1,3`, respectively, but `&` is not put at the top. If n is smaller than 30, it is considered to n =`TeXLim`. If n is 0, the width of a line is not considered.
- `small=1`: Uses the text style in LaTeX for fractions.
  The change of width by this indication is ignored.
- `dviout=1` : Displays r by `dviout`. `TeX=3` is automatically indicated.
  `show()` is useful for the display. The same option parameters are valid.
- `pages=1` : r is too long to put it in a page, `\begin{align*}...` is used and multiple pages are allowed. `pages=2` : Same as above but `\allowdisplaybreaks` are used. The number of lines in a page is indicated by `TeXPages`.
- `add=`s : Puts string s after each term. s may be a variable or a equation.

```
[0] S = os_md.fctrtos(1/(x-y)^2-1/(x+y)^2);
4*y*x/((x-y)^2*(x+y)^2);
[1] eval_str(S);
(4*y*x)/(x^4-2*y^2*x^2+y^4)
[2] os_md.fctrtos(1/(x-y)^2-1/(x+y)^2|TeX=1);
[4yx,(x-y)^2(x+y)^2]
[3] os_md.fctrtos(1/(x-y)^2-1/(x+y)^2|TeX=2);
\frac{4yx}{(x-y)^2(x+y)^2}
[4] os_md.fctrtos((x-a^4+1)^2|var=x);
x^2-2*(a-1)*(a+1)*(a^2+1)*x+(a-1)^2*(a+1)^2*(a^2+1)^2
[5] os_md.fctrtos((x-a^4+1)^2|var=x, TeX=1);
x^2-2(a-1)(a+1)(a^2+1)x+(a-1)^2(a+1)^2(a^2+1)^2
[6] os_md.fctrtos((x+y+1/a)^2|var=[x,y],TeX=1);
x^2+2xy+y^2+\frac{2}{a}x+\frac{2}{a}y+\frac{1}{a^2}
```

```
[7] os_md.fctrtos((x+y+a+b)^2|var=[x,y],TeX=1);
x^2+2xy+y^2+2(a+b)x+2(a+b)y+(a+b)^2
[8] os_md.fctrtos((x+y+a+b)^2|var=[x,y],TeX=1,dic=1);
x^2+2xy+2(a+b)x+y^2+2(a+b)y+(a+b)^2
[9] os_md.fctrtos((x+y+a+b)^2|var=[x,y],TeX=1,rev=1);
(a+b)^2+2(a+b)y+2(a+b)x+y^2+2xy+x^2
[10] os_md.fctrtos((a+b+dx)^2|var="dif",TeX=1);
\partial^2+2(a+b)\partial+(a+b)^2
[11] os_md.fctrtos((a+b+dx1)^2|var="dif",TeX=1);
\partial_{x_1}^2+2(a+b)\partial_{x_1}+(a+b)^2
[12] os_md.fctrtos((a+b+dx)^2|var="dif1",TeX=1);
\frac{d^2}{dx^2}+2(a+b)\frac{d}{dx}+(a+b)^2
[13] os_md.fctrtos((a+b+dx)^2|var="dif2",TeX=1);
\frac{\partial^2}{\partial x^2}+2(a+b)\frac{\partial}{\partial x}+(a+b)^2
[14] os_md.fctrtos((a+dx+dy)^2|var="dif1",TeX=1);
\partial_x^2+2\partial_x\partial_y+\partial_y^2+2a\partial_x+2a\partial_y+a^2
[15] os_md.fctrtos((dx+dy)^2|var="dif1",TeX=1);
\frac{\partial^2}{\partial x^2}+2\frac{\partial^2}{\partial x\partial y}
+\frac{\partial^2}{\partial y^2}
[16] os_md.fctrtos((x+2*y-z)^(20)+1|dviout=1)$
[17] os_md.fctrtos((x+2*y-z)^(20)+1|var=z,dviout=1)$
[18] os_md.fctrtos(os_md.seriesHG([a,b],[c],x,3)|var=x,rev=1,dviout=1)$
[19] os_md.fctrtos((1/(alpha+beta)+dx+d)^4+1|var="dif1",dviout=1)$
```

The result of `[18]` is the display

$$1+\frac{ba}{c}x+\frac{b(b+1)a(a+1)}{2c(c+1)}x^2+\frac{b(b+1)(b+2)a(a+1)(a+2)}{6c(c+1)(c+2)}x^3$$

The result of `[18]` is the following including the break of each line.

$$\frac{d^4}{dx^4}+\frac{4((\alpha+\beta)d+1)}{\alpha+\beta}\frac{d^3}{dx^3}+\frac{6((\alpha+\beta)d+1)^2}{(\alpha+\beta)^2}\frac{d^2}{dx^2}+\frac{4((\alpha+\beta)d+1)^3}{(\alpha+\beta)^3}\frac{d}{dx}$$
$$+\Big((\alpha^4+4\beta\alpha^3+6\beta^2\alpha^2+4\beta^3\alpha+\beta^4)d^4+(4\alpha^3+12\beta\alpha^2+12\beta^2\alpha+4\beta^3)d^3+(6\alpha^2+12\beta\alpha+6\beta^2)d^2+(4\alpha$$
$$+4\beta)d+\alpha^4+4\beta\alpha^3+6\beta^2\alpha^2+4\beta^3\alpha+\beta^4+1\Big)\Big/\Big((\alpha+\beta)^4\Big)$$

If `var=` is indicated and r is considered as a differential operator by

```
[20] os_md.fctrtos((1/(a+b)+dx1+dx2)^2|var="dif0",dviout=1)$
[21] os_md.fctrtos((1/(a+b)+dx1+dx2)^2|var="dif",dviout=1,small=1)$
[22] os_md.fctrtos((1/(b-a)+dx1+dx2)^2|var="dif2",dviout=1)$
[23] os_md.fctrtos((1/(a+b)+dx+d)^2|var="dif",dviout=1)$
[24] os_md.fctrtos((1/(a+b)+dx+d)^2|var="dif0",dviout=1)$
[25] os_md.fctrtos((1/(a+b)+dx+d)^2|var="dif1",dviout=1)$
[26] os_md.fctrtos((1/(a+b)+dx+d)^2|var="dif2",dviout=1,add="u")$
[27] os_md.fctrtos((1/(a+b)+dx+d)^2|var="dif1",dviout=1,small=1)$
```

The results are as follows, respectively.

$$\partial_{x_1}^2+2\partial_{x_1}\partial_{x_2}+\partial_{x_2}^2+\frac{2}{a+b}\partial_{x_1}+\frac{2}{a+b}\partial_{x_2}+\frac{1}{(a+b)^2} \tag{20}$$

$$\partial_1^2+2\partial_1\partial_2+\partial_2^2+\tfrac{2}{a+b}\partial_1+\tfrac{2}{a+b}\partial_2+\tfrac{1}{(a+b)^2} \tag{21}$$

$$\frac{\partial^2}{\partial x_1^2}+2\frac{\partial^2}{\partial x_1\partial x_2}+\frac{\partial^2}{\partial x_2^2}-\frac{2}{a-b}\frac{\partial}{\partial x_1}-\frac{2}{a-b}\frac{\partial}{\partial x_2}+\frac{1}{(a-b)^2} \tag{22}$$

$$\partial^2+\frac{2(da+db+1)}{a+b}\partial+\frac{(da+db+1)^2}{(a+b)^2} \tag{23}$$

$$\partial_x^2+\frac{2(da+db+1)}{a+b}\partial_x+\frac{(da+db+1)^2}{(a+b)^2} \tag{24}$$

$$\frac{d^2}{dx^2}+\frac{2(da+db+1)}{a+b}\frac{d}{dx}+\frac{(da+db+1)^2}{(a+b)^2} \tag{25}$$

$$\frac{\partial^2 u}{\partial x^2}+\frac{2(da+db+1)}{a+b}\frac{\partial u}{\partial x}+\frac{(da+db+1)^2}{(a+b)^2}u \tag{26}$$

$$\tfrac{d^2}{dx^2}+\tfrac{2(da+db+1)}{a+b}\tfrac{d}{dx}+\tfrac{(da+db+1)^2}{(a+b)^2} \tag{27}$$

```
[28] os_md.fctrtos((a+b+x+y)^2|var=[x,y],TeX=2);
x^2+2xy+y^2+2(a+b)x+2(a+b)y+(a+b)^2   /* order by degree of (x,y) */
[29] os_md.fctrtos((a+b+x+y)^2|var=[x,y],dic=1,TeX=2);
x^2+2xy+2(a+b)x+y^2+2(a+b)y+(a+b)^2  /* order by degree of x */
[30] os_md.fctrtos((a+b+x+y)^2|var=[x,y],dic=2,TeX=2);
x^2+2yx+2(a+b)x+y^2+2(a+b)y+(a+b)^2  /* inverse order by degree of x */
[31] os_md.fctrtos((a+b+x+y)^2|var=[x],TeX=2);
x^2+2(y+a+b)x+(y+a+b)^2
```

3.2.4 Functions with real/complex variables

3.2.5 Lists and vectors

3.2.6 Matrices

3.2.7 Strings

3.2.8 Permutations

3.2.9 TEX

351. `show(`p`|opt=`ℓ`,raw=1)`

:: Displays p by using TEX in a plausible format

Display properly a differential operator, a list of eigenvalues with multiplicities, GRS etc. If the display is not good, we may put a option or use other functions with precise options.

- `opt="verb"` : Displays p by `dviout` as in the case `Risa/Asir`.
- `raw=1` : Returns the source in TEX.
  The following are other cases.
- p is a polynomial or a rational function:
  - Displays of the resolution p into factors by using `fctrtos`(p).
  - If the option `opt` is not indicated and p is considered to be a differential operator by `isdif`(p), then p is displayed with the option `var="dif"`.
  - `opt="pfrac"`: Displays decomposition of p into partial fractions.
  - In the other case p is displayed by `fctrtos()` with all the options.

– If p is very long, it is displayed by dividing lines (except the case when p is a monomial or a number).

- If p is a matrix, p is displayed by `mtotex(`p`|lim=1,small=2)`.
  – If the size of the matrix is very large, it is displayed by dividing into lines.
  – `opt=`ℓ : Uses `mtotex()` with the option `var=`ℓ.
- If p is list and all the elements of p are polynomials and p is long, it may be displayed by `eqs2tex()`. In this case the option `var=`V is valid, where V is a variable or the second parameter of `eq2tex()`.
  In the other case, if `opt` is indicated by a string or a list, then p is displayed by `ltotex(`p`)` with all option parameters.
  Otherwise:
  – If p is a list corresponding to GRS or eigenvalues with multiplicities or list of strings, then p may be properly displayed. Here a list of eigenvalues with multiplicities means a list of the pairs of a non-negative integer and an equation or an integer. A list corresponding to GRS is a list of lists of eigenvalues with multiplicities.
  – Suppose p is a list and the element of p is a list and moreover the element is a list of strings or equations (rational functions, vectors, matrices). If the list contains a string and an equation, p is displayed by using `ltotex(`p`|opt=["cr","spts"])`. Here there is a string containing `\`, we put the options `opt=["cr","spts0"]` and `str=1`.
- p is a string:
  – `opt="raw"` : p is displayed as a string of TEX.
  – `opt="eq"` : p is considered as an equation and displayed with `\begin{equation}` p `\end{equation}`.
  – If the above options are not indicated and `\begin{` is not contained and `\draw` or `\ar@` is contained in the first 128 characters, p is displayed by using `xyproc()` considering the setting of `TiKZ`.
  – In the other case, if p contains any of `\begin{equation`, `\begin{align` and `\[` and p does not contain any of `&` `^` and `_`, then p is directly displayed. Otherwise p is displayed by display style.
- In the other case, p is displayed by `dviout(`p`)`.

```
[0] os_md.show(2*2^x|raw=1);
2\cdot{2}^x
[1] os_md.show(1.2*2^x|raw=1);
1.2\cdot{2}^x
[2] os_md.show((x+@i)^2|raw=1);
x^2+(2\sqrt{-1})x-1
[3] os_md.show(2*(x+1)^2/y|raw=1);
\frac{2(x+1)^2}{y}
```

352. `dviout(`p`|clear=1,keep=1,delete=`t`,fctr=1,mult=1,subst=[`$s0, s1$`],eq=`k`,title=`s`)`
:: Displays p
- `dviout(@@)` means to display the last result.
- The command `dviout` should be executable in Windows. If it is not so, `risatex.bat` should be edited and another program (under other operating system such as Unix or Mac) can be used.
- Outputs a LATEX source file to `DIROUT/out.tex` and `DIROUT/risaout.tex` read it.
- If p is a string, it is regarded as a LATEX source by default but the following option indicates it a source of equations in LATEX and use

eq=0 : default display style
eq=1 : \[and \]
eq=2 : \begin{align} and \end{align}
eq=3 : \begin{gather} and \end{gather}
eq=4 : \begin{multline} and \end{multline}
eq=5 : \begin{align}\begin{split} & and \end{split}\end{align}
eq=6 : \begin{align*} & and \end{align*}
eq=7 : \begin{gather*} and \end{gather*}
eq=8 : \begin{equation} and \end{equation}
eq=s with a string s : \begin{s} and \end{s}
to display it in a display style.

- Substitute substrings s_0 by s_1 (multiple substitutions are possible. cf. str_subst()).
- p=" " (a space) : displays again
- clear=1 : displays equations with erasing the last display
- keep=1 : rewrites the LaTeX source file without a display
- delete=t : erases the last t equations with labels of numbers ($0 \leq t \leq 10$).
- mult=1 : displays p in different lines for the elements when p is a list
- fcrt=1 : displays p after factorizing it when p is a polynomial or a rational function.
- tilte=s : displays p with s when p is a string.

  If source special is valid in dviout, a double click at the preview screen given by dviout starts up an editor and moves to the indicated point of the LaTeX source file.

  If the source file is changed, it will be valid in the next display (for example, dviout(" ")) .
- If risatex.bat or risaout.tex does not exists, they are made by os_muldif.rr if the corresponding directory is writable. They can be edited according to the environment.
  risatex.bat should be executable (whose location is get_rootdir()/bin by default) and the file risaout.tex should be in the directory defined by DIROUT (cf. DVIOUTA).

353. dviout0(ℓ) or dviout0([$\ell_1, \ell_2, \ldots$]) or dviout0(ℓ|opt=s)
:: Fundamental operations to display equations using TeX
- $\ell = 0$: Erase display (dviout(" "|keep=1,clear=1)).
- $\ell = 1$: Display (dviout(" ")).
- $\ell = 2$: Erase and Display (dviout(" "|clear=1)).
- $\ell = 3$: Shows values of DIROUT, DVIOUTH, DVIOUTA, DVIOUTB, DVIOUTL, Canvas, TeXLim, TeXEq, AMSTeX, TikZ, XYPrec, XYcm.
- $\ell = 4$: Swaps DVIOUTA and DVIOUTB and shows DVIOUTA.
  See DVIOUTL for an example of the result.
- $\ell = 5$: Converts DVIOUTA and DVIOUTB form the default setting.
- $\ell = 6$: Sets TikZ 1 (to use TikZ).
- $\ell = 7$: Sets TikZ 1 (to use Xy-pic).
- $\ell > 10$: Allows the size of matrix $\mathcal{AMS}$TeX up to ℓ.
- When ℓ is a negative integer, it means dviout(" "|keep=1,delete=$-\ell$), namely, the last ℓ equations are erased.
- If ℓ is a string, \ followed by the string is put in the TeX file and begins a newline
  For example, dviout0("newpage") means to start a new line.
  The following string has an different meaning.
  - "" : A space and a line feeds
  - "cls" : Erase
  - "show" : Display
  - "?" : Shows the setting

- Several commands can be written as a list.
- If `opt=`s is indicated, s is replaced by ℓ (s is a string).
  In this case, if $\ell = -1$, the current value is returned,
  s can be `TikZ`, `TeXEq`, `TeXLim`, `XYPrec`, `XYcm`, `XYLim`, `DVIOUT`, `Canvas` or `TeXPages`.
  In the case when s is `DVIOUT`, the setting of `DVIOUTA` and `DVIOUTB` are set to the initial values if $s = 1$ and its converse if $\ell = 1$.

363. `ltotex(`l`|opt=`s`,pre="string",cr="cr",small=1,lim=`ℓ`,var=`v`)`
:: Transforms a list to plausible TEX source which represents GRS or a table or a list etc.
- `show()` can be used to display the result (the same parameters are valid).
- $s =$ `"spt"` : Transforms the list (given by `meigen( |mult=1)`) as a spectral type..
- $s =$ `"GRS"` : Transforms the list (given by `sp2grs()`) as a GRS. In this case `pre` is valid.
- $s =$ `"coord"` : Transforms the list to (, ,...) as a coordinate.
  - `cpx=1` : Complex number $a + b\sqrt{-1}$ is transformed to $a + bi$ (by default).
  - `cpx=2` : Complex number $a + b\sqrt{-1}$ is transformed to $a + b\sqrt{-1}$.
- $s =$ `["Pfaff",u,x,x-y,...]` : A list of matrices is transformed to a Pfaff form such as

$$du = \left(A_0 \frac{dx}{x} + A_1 \frac{d(x-y)}{x-y} + \cdots \right) u.$$

- $s =$ `["Fuchs",u,x,x,x-y,x-1,...]` : A list of matrices is transformed to a Fuchsian form such as

$$\frac{du}{dx} = \left(\frac{A_0}{x} + \frac{A_1}{x-y} + \frac{A_2}{x-1} + \cdots \right) u.$$

- $s =$ `"dform"` : A list in the format given by `dform()` is considered as a differential form.
  If $\ell =$`[[(a*x-b*y)/(x),x,z],[(-a*x+b*y)/(y),y,z]]`, we have

$$(\frac{ax-by}{x})\, dx \wedge dz + (\frac{-ax+by}{y})\, dy \wedge dz.$$

  Here the coefficients are (matrices of) rational functions.
- $s =$ `"vect"` : Transforms the list as a column vector,
- $s =$ `"cr"` : Element are put in different lines.
  As in the following, `cr=` and `var=` are possible.
- $s =$ `"spts"` : Puts a space between the elements of the list and arranges elements as many as possible in each line.
- $s =$ `"spts0"` : Same as above but does not put a apace.
  - `lim=`ℓ : The line width is considered to be ℓ as a number of characters. If $\ell = 0$, the limit is ignored.
  - `str=1` : Strings are considered usual strings in TEX
  - `cr=` : The code of a new line is indicated. It is `cr="\\\\\n & "` by default.
    If it is an integer k with $0 \leq k < 32$, then the code of a new line is `texcr(`k`)`. Then $k = 7 = 2 + 1 + 4$ by default and if $k = 0$, it is a space and if $k = 15$, it is `\\allowdisplaybreaks\\\\\n&` .
  - `var=` : Defines a variable for matrices (cf. `mtotex()`), polynomials and rational functions (cf. `ltotex()`).
- $s =$`["cr","spts"]` or $s =$`["cr","spts0"]` : Puts a new line between elements (`cr=` and `var=` can be indicated). If the element of the list is a list, the element follows the indication $s =$`"spts"` or $s =$`"spts0"` and moreover `str=1` can be indicated.
  If the elements of ℓ is lists and we put `cr=[`s_1`,`s_2`]`, the elements of ℓ will be separated by the code s_1 and the elements of the lists will be separated by the code s_2. s_1 and s_2 can be strings or integers as above.

- s = "text" : It is same as in the case s = "spts" but l is returns as in a text style. In this case str=1 and cr= can be indicated.
- s = "tab" and ℓ is a list of lists : Returns a table \begin{tabular}...\end{tabular} in a test style.
  - title=s : The string s is a title of the table.
  - left=[s_1,s_2,...] : Inserts the fist column of the table.
    If s_j=[$s_{j,1}$,$s_{j,2}$,...] for a suitable j, it will be repeated up to the last line, which is same in the following.
  - right=[s_1,s_2,...] : Inserts as the last column.
  - top=[s_1,s_2,...] : Inserts as the first line. If right= is indicated, the indication right= is processed before top=.
  - last=[s_1,s_2,...] : Inserts as the last line.
  - null=n : The element equals n is transformed into a space (n = "" by default).
  - hline=[$h_1, h_2, \ldots$] : Inserts horizontal lines after h_i-th line. Here the top of the table without a title is indicated by 0 and the first line of the table is indicate by 1 and a multiple indication of the same number means a horizontal multiplet.
    - z : means (after) the last line. The indication z-1 is possible.
    - v_j =[k_j,n_j] with a positive integer n_j and a non-negative integer n_j : Indicates k-th lines of the table with $k = \nu n_j + k_j$ for $\nu = 0, 1, 2, \ldots$.
  - vline=[$v_1, v_2, \ldots$] : Indicates horizontal lines after v_j-th lines of the table.
    The first column is indicated by 1 and a multiple indication by a same number means a horizontal multiplet.
    - z means the last column of the table. The indication z-1 is possible.
    - v_j =[k_j,n_j] with a positive integer n_j and a non-negative integer n_j : Indicates k-th columns with $k = \nu n_j + k_j$ for $\nu = 0, 1, 2, \ldots$.
  - align=s : s is the string to define the arrangement and vertical lines indicated as {s} after \begin{tabular}. If this is indicated, vline= is ignored.
    But if s is a character, it is considered as a character defining the arrangement and the indication vline= is valid.
    s="r" : right, s="c" : center, d="l" : left.
  - vert=1 : Transposes the table.
  - width=w : If w is a positive integer, the number of the columns of the table is limited by w and the table is changed int a piled table.
    If w is a negative integer, the number of the columns of the table is multiplied by $|w|$ and the table is changed into a table with a smaller number of lines.
    - If vert=1 is indicated, the above transformation is processed after the transposition.

  See examples powprimroot().
  To arrange the table it is convenient to so as follows. We transform the corresponding list to a matrix by lv2m(|null="") and arrange the matrix by madjust(), mtranspose(), mperm(), newbmat() etc. and then transforms the matrix to a list by m2ll().
- s = "graph" and ℓ is a list of numbers : Returns a bar graph by using XY-pic or TikZ. To put strings or numbers below the bar graph, we insert a list of the strings or the numbers after the last list of numbers.
  If the number of the element of the list is smaller than the number of elements of the data in a line by 1, the strings or numbers are put between the bars.
  - size=ℓ : Indicates the size of the graph.
    ℓ is a list of numbers formed by the width of the graph, the hight of the graph, the ratio of the gap between bars and the width of a bar, which is ignored in the case of a line graph, the height between the graph and the strings, which is the same value as the former if it

is not indicated.

They are omitted except the first and the second numbers.

The second element of ℓ is a negative number $-c$, the height of the bar is indicated to be the number of the table $\times$ c.

`TikZ=0`, `1` : $\ell = [80, 30, 1/2, 2]$, $[8, 3, 1/2, 0.2]$ respectively, by default.

If `horiz=1` is indicated, the 4-th values 4 and 0.2 are changed to 2 and 0.2, respectively, by default.

- `max=`m : m is a maximal value (which is automatically determined by default. This value is the height of the graph).
- `shift=`n : The bottom reference line is shifted by n (0 by default).
- `horiz=1` : A bar graph with horizontal bars.
- `line=1` : A line graph.
  `line=2` : Same as above with indicated the points by $\bullet$.
- `line=[`n`,`t`]` : n is 2 or 1 according to the use of points in a line graph.
  t indicates the style of lines which corresponds to the indication in XY-pic/TikZ. For example, `s="@{.}"/dotted` is a dotted line (cf. `xyarrow()`).
- `line=[-1,`r`]` : A pie chart with a circle of radius r mm.
- `value=0` : Does not show values.
  `value=1` : Shows the corresponding value even if the bar is shrunk to the bottom reference line.
- `strip=1` : The strings `\begin{xy}` $\cdots$ `\end{xy}` or `\begin{tikzpicture}` $\cdots$ `\end{tikzpicture}` are omitted.
  `strip=2` : The bottom reference line is omitted.
  `strip=3` : Only the values are shown (the part omitted by `value=0`).
- `color=`s : Colors and fills the graph if TikZis used (same as in the case of `xybox()`). In the case of a pie graph s is a list of the indications to the data.
- `mult=1` : Draw several bar graphs or line graphs in one graph. In this case ℓ `=[[`ℓ_1`,`ℓ_2`,...],`m`]` and `color` or `line` should be indicated by the corresponding list.

  `relative=1` : The values of ℓ_1, ℓ_2 are transformed to the values added successively.

  If this is not indicated in the case of a bar graph, the element of ℓ_{j+1} should be larger than or equals to the corresponding element of ℓ_j.

- If R is returned and s =`"text"`, `"tab"` or `"graph"`, R is displayed by `dviout(`R`)` and by `dviout(`R`|eq=5)` otherwise.
- `small=1` : Uses `\begin{smallmatrix}`.

```
[0] os_md.ltotex([a+b, c/d,[2,3]]);
\left\{
 a+b,\, \frac{c}d,\, [2,3]
\right\}

[1] os_md.dviout(@@|eq=5)$
```

$$[a+b, \frac{c}{d}, [2,3]]$$

```
[2] L=[[12,a+b],[3,c],[1,d]];
[[12,a+b],[3,c],[1,d]]
[3] os_md.ltotex(L|opt="spt");
\left\{
```

```
 [a+b]_{12},\, [c]_3,\, d
\right\}

[4] os_md.dviout(@@|eq=5)$
```

$$\{[a+b]_{12},\, [c]_3,\, d\}$$

```
[5] LL=[L,[[3,3*b],[2,f]]];
[[[12,a+b],[3,c],[1,d]],[[3,3*b],[2,f]]]
[6] os_md.ltotex(LL|opt="GRS",pre=" 0 & 1\\\\\n");
\begin{Bmatrix}
 0 & 1\\
 [a+b]_{12} & [3b]_3\\
 [c]_3 & [f]_2\\
 d &
\end{Bmatrix}
[7] os_md.dviout(@@|eq=5)$
```

$$\left\{\begin{matrix} 0 & 1 \\ [a+b]_{12} & [3b]_3 \\ [c]_3 & [f]_2 \\ d & \end{matrix}\right\}$$

```
[8] A=newmat(2,2,[[a,b],[c,d]]);
[ a b ]
[ c d ]
[9] B=newmat(2,2,[[p,q],[r,s]]);
[ p q ]
[ r s ]
[10] os_md.ltotex([A,B]|opt=["Pfaff",u,x-1,y]);
du= \Biggl(\begin{pmatrix}
 a&b \\
 c&d
\end{pmatrix}\frac{d(x-1)}{x-1}
\\&
 + \begin{pmatrix}
 p&q \\
 r&s
\end{pmatrix}\frac{dy}{y}
\Biggr)u

[11] os_md.dviout(@@|eq=5,subst=["\\\\&","%"])$
```

$$du = \left(\begin{pmatrix} a & b \\ c & d \end{pmatrix}\frac{d(x-1)}{x-1} + \begin{pmatrix} p & q \\ r & s \end{pmatrix}\frac{dy}{y}\right)u$$

```
[12] os_md.ltotex([A,B]|opt=["Fuchs",u,x,x,x-1]);
\frac{du}{dx}= \Biggl(\frac{\begin{pmatrix}
 a&b \\
```

```
 c&d
\end{pmatrix}}{x}
 + \frac{\begin{pmatrix}
 p&q \\
 r&s
\end{pmatrix}}{x-1}
\Biggr)u
```

```
[13] os_md.dviout(os_md.smallmattex(@@)|eq=5);
```

$$\frac{du}{dx} = \left(\frac{\left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)}{x} + \frac{\left(\begin{smallmatrix} p & q \\ r & s \end{smallmatrix}\right)}{x-1} \right) u$$

```
[14] P=[[2*a,x,y],[(a+b)^2,y,z],[-2,x,z/y]]$
[15] os_md.ltotex(P|opt="dform");
2a\,dx\wedge dy+(a^2+2ba+b^2 )\,dy\wedge dz-2\,dx\wedge d(\frac{z}y)
[16] os_md.dviout(@@|eq=5,subst=["\\frac","\\tfrac"]);
```

$$2a\,dx \wedge dy + (a^2 + 2ba + b^2)\,dy \wedge dz - 2\,dx \wedge d(\tfrac{z}{y})$$

```
[17] os_md.ltotex(["There are",(n+1)^2,"points."]);
[\texttt{There are},n^2+2n+1,\texttt{points.}]
[18] os_md.dviout(@@|eq=5)$
```

$$[\texttt{There are}, n^2 + 2n + 1, \texttt{points.}]$$

```
[19] os_md.ltotex(["There are",(n+1)^2,"points."]|opt="text");
$\texttt{There are}$
$n^2+2n+1$
$\texttt{points.}$
[20] os_md.dviout(@@)$
```

There are $n^2 + 2n + 1$ points.

```
[21] os_md.ltotex(["There are",(n+1)^2,"points."]|opt="text",str=1);
There are $n^2+2n+1$
points.
[22] os_md.dviout(@@)$
```

There are $n^2 + 2n + 1$ points.

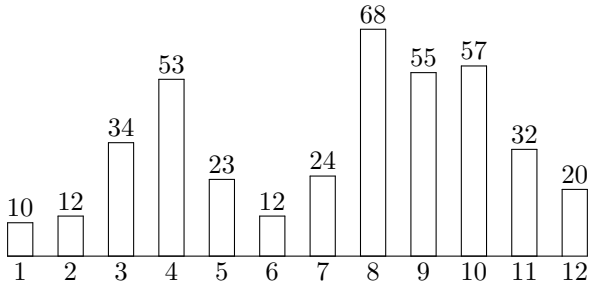
```
[23] L=[10,12,34,53,23,12,24,68,55,57,32,20]$
[24] M=[1,2,3,4,5,6,7,8,9,10,11,12]$
[25] os_md.dviout(os_md.ltotex([M,L]|opt="tab",title="Year 2014"));
```

We display the above TEX source.

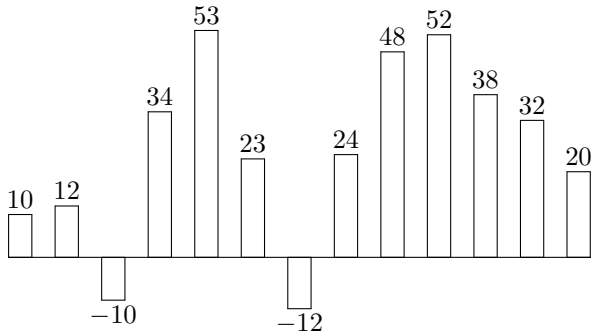
Year 2014

1	2	3	4	5	6	7	8	9	10	11	12
10	12	34	53	23	12	24	68	55	57	32	20

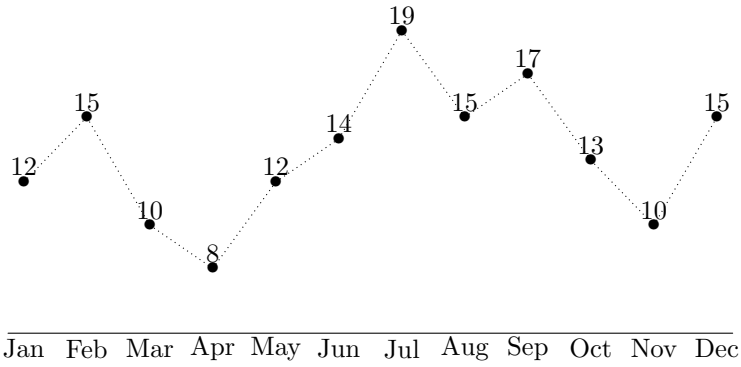
```
[26] os_md.dviout(os_md.ltotex([L,M]|opt="graph"));
```




```
[27] L=[10,12,-10,34,53,23,-12,24,48,52,38,32,20]$
[28] os_md.ltotex(L|opt="graph");
```




```
[29] L=[12,15,10,8,12,14,19,15,17,13,10,15]$
[30] M=["Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov",
     "Dec"]$
[31] os_md.ltotex([L,LL]|opt="graph",line=[2,"@{.}"],shift=5,size=[100,40]);
```

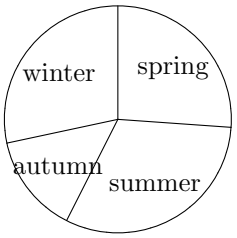



```
[32] L=cons("number",L)$
[33] M=cons("Month",M)$
[34] os_md.ltotex([M,L]|opt="tab",hline=[0,1,2],vline=[0,1,1,13],
title="Year 2014");
```

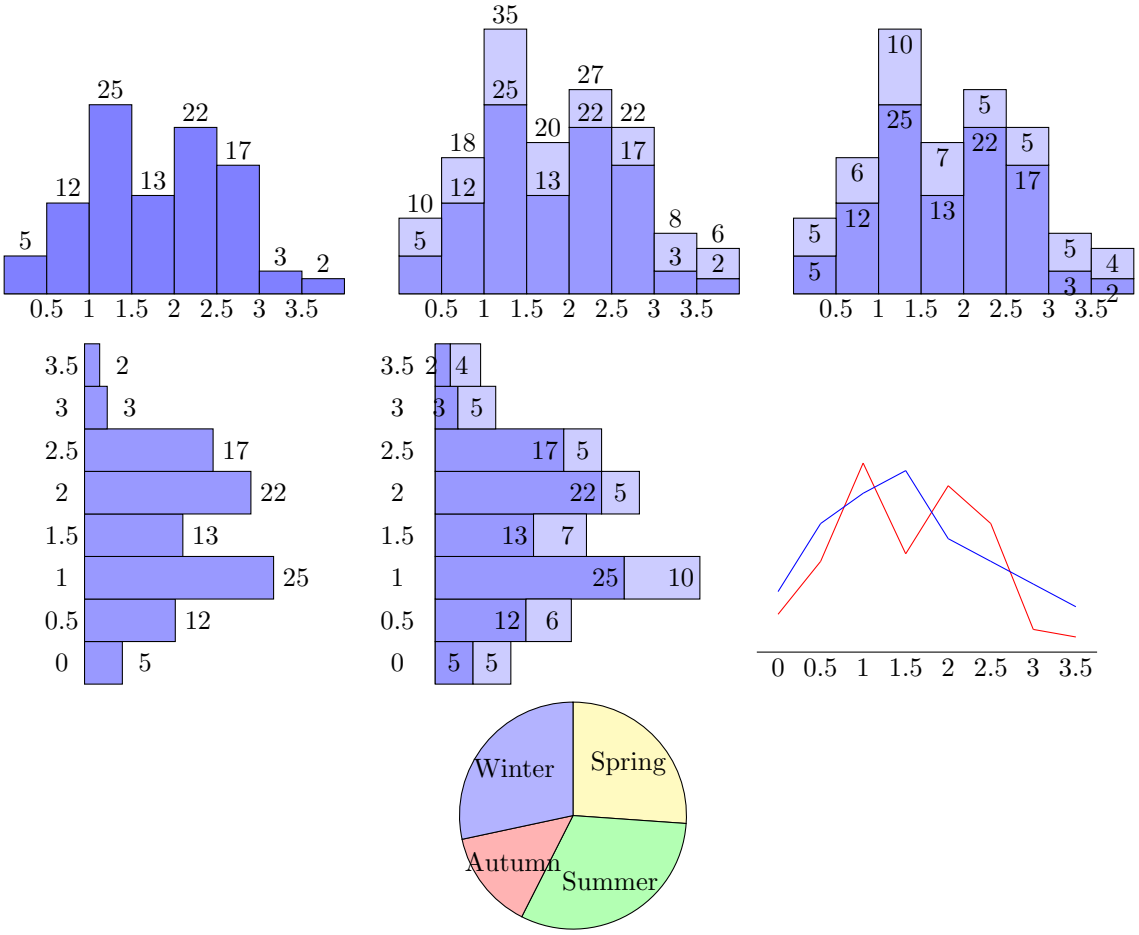
Year 2014

Month	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
number	12	15	10	8	12	14	19	15	17	13	10	15

```
[35] L=[35,42,19,38]$
[36] LL=["spring","summer","autumn","winter"]$
[37] os_md.ltotex([L,LL]|opt="graph",line=[-1,15]);
```




```
[38] L=[5,12,25,13,22,17,3,2]$
[39] M=[0.5,1,1.5,2,2.5,3,3.5]$
[40] os_md.dviout0(1|opt="TikZ")$
[41] os_md.ltotex([L,M]|opt="graph",color="fill=blue!50",size=[4.5,-0.1,1]);
/* width 4.5cm, height x 0.1, ratio (width) 1 */
\begin{tikzpicture}
\draw(0,0)--(4.5,0);
\draw[fill=blue!50](0,0)rectangle(0.563,0.5);
\node at(0.281,0.7){$5$};
\node at(0.563,-0.2){$ 0.5$};
\draw[fill=blue!50](0.563,0)rectangle(1.125,1.2);
...
[42] L2=[10,18,35,20,27,22,8,6]$
[43] os_md.ltotex([[L,L2],M]|opt="graph",color=["fill=blue!40","fill=blue!20"],
size=[4.5,-0.1,1],mult=1);
[44] L3=lsub([L2,L]);
[5,6,10,7,5,5,5,4]
[45] os_md.ltotex([[L,L3],M]|opt="graph",color=["fill=blue!40","fill=blue!20"],
size=[4.5,-0.1,1],mult=1,relative=1);
[46] N=cons(0,M);
[0,0.5,1,1.5,2,2.5,3,3.5];
[47] os_md.ltotex([L,N]|opt="graph",color="fill=blue!40",
size=[4.5,-0.1,1],horiz=1);
[48] os_md.ltotex([[L,L3],N]|opt="graph",color=["fill=blue!40",
"fill=blue!20"],size=[4.5,-0.1,1,0.5,0.25],mult=1,horiz=1,relative=1);
[49] os_md.ltotex([L,N]|opt="graph",line=[2,"red"],size=[4.5,-0.1,1]);
[50] L3=[8,17,21,24,15,12,9,6]$
[51] os_md.ltotex([[L,L3],N]|opt="graph",line=[[1,"red"],[1,"blue"]],mult=1,
size=[4.5,-0.1,1],value=0);
[52] L=[35,42,19,38]$LL=["spring","summer","autumn","winter"]$
[53] os_md.ltotex([L,LL]|opt="graph",line=[-1,15],color=["fill=yellow!30",
     "fill=green!30","fill=red!30","fill=blue!30"]);
```



3.2.10 Lines and curves

3.2.11 Drawing curves and graphs

3.2.12 Applications

3.2.12.1 Make tables

3.2.12.2 Make tables and matrices

3.2.12.3 Mathematical tables

3.2.12.4 Table of score distribution

3.2.12.5 Tayler's expansion

3.2.12.6 Slide Scale

3.2.13 Environments

430. `Canvas`
:: Default size of the canvas of `Risa/Asir`

431. `AMSTeX`
:: TEX means $\mathcal{AMS}$LaTeX if this value equals 1
In `os_muldif.rr` `AMSTeX=1` by default. If otherwise, the TEX source created some functions may not be correct.

```
[0] M=newmat(2,2,[[a,b],[c,d]]);
[ a b ]
[ c d ]
[1] AMSTeX=0$
[2] print_tex_form(M);
\pmatrix{
 {a}&  {b} \cr
 {c}&  {d} \cr
}

[3] AMSTeX=1$
[4] print_tex_form(M);
\begin{pmatrix}
 {a}&  {b} \\
 {c}&  {d}
\end{pmatrix}

[5] os_md.my_tex_form(M);
\begin{pmatrix}
 a&b \\
 c&d
\end{pmatrix}
[6] print_tex_form(x_1+x_2^2/y);
\frac{   {x}_{1}  {y}+  {x}_{2}^{ 2} }{ {y}}
[7] os_md.my_tex_form(x_1+x_2^2/y);
\frac{x_1y+x_2^2}y
```

432. `TeXEq`
:: Default display style of LaTeX (`dviout0(3)` shows it)
The value `TeXEq` is one of 1, 2, 3, 4, 5, 6, 7 and it corresponds to the value of the option parameter `eq=` of `dviout()`.
When `AMSTeX=1`, the default is 5 and when `AMSTeX=0` it is `TeXEq=1`.
It can be changed by `dviout0(`n`|opt="TexEq")`.

433. `TeXLim`
:: Maximal width referred to divide a long equations into lines by LaTeX
- The number means an estimated length measured by the number of characters.
- It can be defined in `.muldif` and `dviout0("?")` shows the value.
- `texlim(1,`n`)` or `dviout0(`n`|opt="TexLim")` can change the value (the default value is 80).

434. `TeXPages`
: Threshold value of the numbers of lines for a page in a display style of TeX (cf. `fctrtos()`)
- `dviout0(`n`|opt="TexPages")` can change the value (80 by default).

435. `TikZ`
:: Flag using XY-pic or TikZ in TeX
`TikZ=1` : TikZand , `TikZ=0` : XY-pic
The flag is changed by `dviout0(6)`, `dviout0(7)` or `dviout0(1|opt="TikZ")`, `dviout0(0|opt="TikZ")`.

436. `XYPrec`
:: Number of figures after the decimal point in the expression of coordinates for graphic display

It can be changed by dviout0(*n*|opt="XYPrec").

437. XYcm
:: Unit is cm even in XY-pic
It can be changed by dviout0(0|opt="XYcm") or dviout0(1|opt="XYcm").

438. XYLim
:: Maximal number of coordinates for a line in a source of XY-pic or Ti*k*Z
It can be changed by dviout0(*n*|opt="XYLim").

439. DVIOUTH
:: A program to show explanations of functions indicated by myhelp()
- This can be define in .muldif and dviout0("?") shows the definition.
- The default setting when the command dviout works in Windows is

  start dviout -2 -hyper:0x90 "%ASIRROOT%\help\os_muldif.dvi" #%LABEL%

  – This means that 2-nd dviout shows the place labeled by %LABEL% (which myhelp(*s*) replaces it by *s*) of get_rootdir()\help\os_muldif.dvi.
  – If dviout is installed in Windows and the command dviout does not work, the full pathname can be indicated.
  – The indication -hyper:0x90 means to show the hot spot in blue (which is similar in ox_muldif.pdf). If it is not indicated, the underline is also shown,
  In dviout this setting can be changed by Option → Setup parameters... → HyperTeX → Color and the button OK. If the value -hyper: is set by the value defined by hyper= in Option → Non-default Parameters, the hot spot is shown by the setting.

  The explanation of *fn* exists in the place with the label r:*fn*. If it contains the underscore "_", it is omitted in the label.
- myhelp() shows the explanation of an indicated function %ASIRROOT% is replaced by get_rootdir() and %LABEL% is replaced by the corresponding label as above and then myhelp() shows the explanation of the indicated function

440. DIROUT
:: Directory where the source file of LaTeX is put (it should be writable)

441. DVIOUTA
:: Pathname of a program which compiles a source risaout.tex of LaTeX under $\mathcal{AMS}$TeX and displays it

442. DVIOUTB
:: Pathname of a program which compiles a source risaout10.tex (or risaout10.tex) of LaTeX under $\mathcal{AMS}$TeX and displays it. We can choose this setting or DVIOUTA in Risa/Asir

443. DVIOUTL
Pathname of a program which compiles a source risaout0.tex of LaTeX and displays it
We can set it in .muldif, and the values are obtained by the command dviout0("?").
The default under Windows is

```
[0] os_md.dviout0(3)$
DIROUT ="%HOME%\tex"
DVIOUTH="start dviout -2 -hyper:0x90 "%ASIRROOT%\help\os_muldif.dvi" #%LABEL%"
DVIOUTA="%ASIRROOT%\bin\risatex.bat"
DVIOUTB="%ASIRROOT%\bin\risatex1%TikZ%.bat"
DVIOUTL="%ASIRROOT%\bin\risatex0.bat"
Canvas = [400,400]
TeXPages = 20
TeXLim = 80
```

```
TeXEq  = 5
AMSTeX = 1
TikZ   = 0
XYPrec = 3
XYcm   = 0
XYLim  = 4
```

The default setting under other platforms is

```
[0] os_md.dviout0(3)$
DIROUT ="%HOME%/asir/tex"
DVIOUTH="%ASIRROOT%/help/os_muldif.pdf"
DVIOUTA="%ASIRROOT%/bin/risaout.sh"
DVIOUTB="%ASIRROOT%/bin/risaout1%TikZ%.sh"
DVIOUTL="%ASIRROOT%/bin/risaout0.sh"
DVIOUTL="%ASIRROOT%/bin\risatex0.bat"
TeXPages = 20
Canvas = [400,400]
TeXLim = 80
TeXEq  = 5
AMSTeX = 1
TikZ   = 0
XYPrec = 3
XYcm   = 0
XYLim  = 4
```

Here %TikZ%is the value of Ti*k*Z, %ASIRROOT% is the install directory obtained by get_rootdir(), %HOME% means the value of environment variable HOME but the latter equals %ASIRROOT% buy default in Windows.
In .muldif, %ASIRROOT% and %HOME% are not allowed to contain spaces and they should be indicated by using " as

```
DIROUT ="\"%HOME%\asir\tex\""
DVIOUTA="\"%ASIRROOT%\\bin\risaout.bat\""
DVIOUTB="\"%ASIRROOT%\bin\risaout1%TikZ%.bat\""
DVIOUTL="\"%ASIRROOT%\bin\risaout0.bat\""
```

if necessary.

444. .muldif

:: os_muldif.rr reads this file when it starts up

- If .muldif is in %HOME%, it is read when os_muldif.rr is loaded. It is also searched in %HOME%/asir, %ASIRROOT%, %ASIRROOT%/bin, %ASIRROOT%/lib-asir-contrib in this order. Here %ASIRROOT% is equal to get_rootdir(). Moreover %HOME% coincides with the environment variable HOME and it is also coincides with %ASIRROOT% under Windows.
- The values TeXLim, TeXPages, TeXEq, DIROUT, DVIOUTA, DVIOUTB, DVIOUTL, DVIOUTH, TikZ, XYPrec, XYcm, XYLim, Canvas can be changed from the default values.
- The current setting can be obtained by dviout0(3).

An example of .muldif is

```
DVIOUTH="start c:\\dviout\\dviout -2 \"%ASIRROOT%\\help\\os_muldif.dvi\" #%LABEL%"
end$
```

Following the format of `C`, the letter `\` and `"` in a word should be written by `\\` and `\"` etc. (It may be unnecessary in many cases). Beginning of a sequence of strings and its end can be clarified by `"`.

445. `risatex.bat`

:: Program compiling the LaTeX file which `os_muldif.rr` outputs and displaying it on a screen

- `os_muldif.rr` outputs a source file `out.tex` (or `out0.tex`) of LaTeX to get a nice display of resulting results and equations on a screen and then `risatex.bat` called by `os_muldif.rr` reads it from `risaout.tex` (or `risaout1.tex`, `risaout0.tex`) and compiles it into a `dvi` file or a `pdf` file and display them. Here `risaout1.tex` or `risaout0.tex` is called by `risaout0.bat` as in the case of `risatex.bat`.
- DVIOUTA (or DVIOUTB, DVIOUTL) indicates the above. We can define three different ways to process the LaTeX file using `DVIOUTA`, `DVIOUTB` and `DVIOUTL`.

Examples of settings to display results and equations using TeXare as follows.

- The default setting of `risatex.bat` which is located in `get_rootdir()\bin` is (cf. DVIOUTA) :

  ```
  cd "c:\Program Files\asir\tex"
  platex -src=cr,display,hbox,math,par risaout
  start dviout -1 "c:\Program Files\asir\tex\risaout" 1000
  ```

  – `c:\Program Files\asir` in the above is replaced by `get_rootdir()`. In the first line `cd` indicates the directory given by DIROUT where `os_muldif.rr` outputs a LaTeX file. The directory should be writable.
  – If the command `dviout` does not work, it should be indicated by its full pathname or it should be replaced by another program to display the compiled file processed by TeX system.
  – The last parameter `-1` in the last line indicates to the command to the 1-st `dviout`. It will start up if it is not so.
  – The last parameter 1000 indicated the page to be displayed. If it is sufficiently large, it means the last page. The equations to be displayed will be added at the end.

  An example of `risaout.tex` whose location is given by DIROUT:

  ```
  \documentclass[a4paper]{amsart}
  \usepackage{amsmath, amssymb, amsfonts}
  \pagestyle{empty}
  \begin{document}
  \thispagestyle{empty}
  \input{out}
  \end{document}
  ```

  The following example allows longer equations in a line. Moreover Xy-pic (or Ti*k*Z) will be loaded to handle several graphics.

  ```
  \documentclass[a4paper]{amsart}
  \usepackage{amsmath,amssymb,amsfonts}
  \AtBeginDvi{\special{dviout -y=A3L}}
  \usepackage[all]{xypic}
  \pagestyle{empty}
  \textwidth=7.6in
  \textheight=11in
  ```

```
\voffset=-1.4in
\hoffset=-1.4in
\begin{document}
\thispagestyle{empty}
\input{out}
\end{document}
```

If a pdf file is created by TEX by dvipdfmx or pdfTeX, it is better to change the above \usepackage[all]{xypic} by \usepackage[pdf,all]{xypic} so that a better pdf is created.
To set it by DVIOUTB (which os_muldif.rr uses by the command dviout0(4)), we put the following risaoutpdf.tex in c:\Program Files\asir\tex

```
\documentclass[a4paper]{amsart}
\usepackage{amsmath,amssymb,amsfonts}
\usepackage{pdf,all]{xypic}
\pagestyle{empty}
\begin{document}
\thispagestyle{empty}
\input{out}
\end{document}
```

and moreover risatex1.bat may be

```
cd "c:\Program Files\asir\tex"
platex risaoutpdf
dvipdfmx risaoutpdf
risaoutpdf.pdf
```

and risatex1.bat is indicated by DVIOUTB with its full pathname.

To display the pdf by SumatraPDF.exe, the last line in the above may be replaced by

```
 "c:\Program Files\SumatraPDF\SumatraPDF.exe" -reuse-instance risaoutpdf.pdf
```

If we use Ti*k*Z in place of XY-pic, which is more desirable, risaoutpdf.tex may be as follows:

```
\documentclass[dvipdfmx,a4paper]{amsart}
\usepackage{amsmath,amssymb,amsfonts}
\usepackage{tikz}
\pagestyle{empty}
\begin{document}
\thispagestyle{empty}
\input{out}
\end{document}
```

To use graphics with Ti*k*Z, we put a line

```
TikZ=1
```

in .muldif. This can be changed by dviout0(6) or dviout0(7) after loading os_muldif.rr.

To use both of XY-pic and Ti*k*Z, we set as follows.
Since %TiKZ% in DVIOUTB is replaced by the value %TiKZ%, we change the filenames of risaoutpdf.tex and risaoutpdf0.tex to risaoutpdf0.tex and risaoutpdf1.tex which correspond to XY-pic and Ti*k*Z respectively, and risatex10.bat are risatex11.bat

```
cd "c:\Program Files\asir\tex"
platex risaoutpdf0
dvipdfmx risaoutpdf0
"c:\Program Files\SumatraPDF\SumatraPDF.exe" -reuse-instance risaoutpdf0.pdf
```

and

```
cd "c:\Program Files\asir\tex"
platex risaoutpdf1
dvipdfmx risaoutpdf1
"c:\Program Files\SumatraPDF\SumatraPDF.exe" -reuse-instance risaoutpdf1.pdf
```

We may use XY-pic and Ti*k*Z at a time and then its example is

```
\documentclass[dvipdfmx,a4paper]{jsarticle}
\usepackage{amsmath,amssymb,amsthm,amscd,mathrsfs}
\usepackage{tikz}
%\usetikzlibrary{patterns,shapes} % if necessary
\usepackage[pdf,all]{xy}
\begin{document}
\thispagestyle{empty}
\input{out}
\end{document}
```

- In the case `AMSTeX=0`, `risatex0.bat` and `risaout0.tex` are used and they are as follows by default.

```
cd "c:\Program Files\asir\tex"
platex -src=cr,display,hbox,math,par risaout0
start dviout -1 "c:\Program Files\asir\tex\risaout0" 1000
```

```
\documentclass{article}
\pagestyle{empty}
\begin{document}
\thispagestyle{empty}
\input{out0}
\end{document}
```

- Under `Unix` or `Mac`, `risatex.bat` may be replaced by `risatex.sh` which is the default name and we set it executable (`chmod 755`). For example, it is

```
#!/bin/sh
cd ${HOME}/asir/tex
platex -src=cr,display,hbox,math,par risaout
dvipdfmx risaout
evince risaout.pdf &
```

or

```
#!/bin/sh
cd $HOME/asir/tex
/usr/local/texlive/2014/bin/x86_64-darwin/platex risaout
/usr/local/texlive/2014/bin/x86_64-darwin/dvipdfmx risaout
open -a Preview risaout.pdf
```

In this case `risaout.tex` may be

```
\documentclass[a4paper]{amsart}
\usepackage{amsmath,amssymb,amsfonts}
\usepackage[pdf,all]{xy}
\pagestyle{empty}
\begin{document}
\thispagestyle{empty}
\input{out}
\end{document}
```

Here `risatex0.bat` and `risatex1.bat` may be `risatex0.sh` and `risatex1.sh`.

- `risaout.tex` and `risaout0.tex` do not exist in `DIROUT` and the directory is writable, they are automatically created if necessary.

3.2.14 Suppliment

3.2.14.1 Input matrices

3.2.14.2 Plane figure

3.2.14.3 Function by list

3.2.14.4 Analyze functions with values in real/complex numbers

3.2.14.5 Data by table

Bibliography

[Apéri] R. Apéry, Irrationalité de $\zeta(2)$ et $\zeta(3)$, Journées arithmétiques de Luminy, Astérisque no.61, (1979), 11–13.

[Batut et al.] C. Batut, D. Bernardi, H. Cohen, M. Olivier, *User's Guide to PARI–GP*, 1993.

[Becker-Weispfenning] T. Becker, V. Weispfenning, *Groebner Bases*, Graduate Texts in Math. **141**, Springer-Verlag, 1993.

[Boehm-Weiser] H. Boehm, M. Weiser, *Garbage Collection in an Uncooperative Environment*, Software Practice & Experience, September 1988, 807–820.

[DR] M. Dettweiler and S. Reiter, *An algorithm of Katz and its applications to the inverse Galois problems*, J. Symbolic Comput. **30**(2000), 761–798.

[Gebaue-Moeller] R. Gebauer, H. H. Moeller, *An installation of Buchberger's algorithm*, J. of Symbolic Computation **6**, 275–286.

[Giovini et al.] A. Giovini, T. Mora, G. Niesi, L. Robbiano, C. Traverso, *"One sugar cube, please" OR Selection strategies in the Buchberger algorithm*, Proc. ISSAC'91, 49–54.

[Ha] Y. Haraoka, *Middle convolution for completely integrable systems with logarithmic singularities along hyperplane arrangements*, Adv. Studies in Pure Math. **62**(2012), 109–136.

[Hiroe-Kawakami-Nakamura-Sakai] K. Hiroe, H. Kawakami, A. Nakamura and H. Sakai, *4-dimensional Painlevé-type equations*, MSJ Memoirs vol.37, Mathematical Society of Japan, 2018.

[Hiroe-Oshima] K. Hiroe and T. Oshima, *A classification of roots of symmetric Kac-Moody root systems and its application*, Symmetries, Integrable Systems and Representations, Springer Proceedings in Mathematics and Statistics **40** (2012), 195–241.

[Morier-Genoud and Ovsienko] S. Morier-Genoud and V. Ovsienko, q-deformed rationals and q-continued fractions, arXiv:1812.00170, 2020.

[Noro-Takeshima] M. Noro, T. Takeshima, *Risa/Asir – A Computer Algebra System*, Proc. ISSAC'92, 387–396.

[Noro-Yokoyama] M. Noro, K. Yokoyama, *A Modular Method to Compute the Rational Univariate Representation of Zero-Dimensional Ideals*, J. Symb. Comp. **28/1** (1999), 243–263.

[O1] T. Oshima, *Annihilators of generalized Verma modules of the scalar type for classical Lie algebras*, "Harmonic Analysis, Group Representations, Automorphic forms and Invariant Theory", in honor of Roger Howe, Vol. 12, Lecture Notes Series, 2007, 277-319, National University of Singapore.

[O2] T. Oshima, *special functions and algebraic linear ordinary differential equations*, in Japanese, Lecture note **11**, Department of Mathematics Univ. of Tokyo 2011, http://www.ms.u-tokyo.ac.jp/publication/documents/spfct3.pdf.

[O3] T. Oshima, *Fractional calculus of Weyl algebra and Fuchsian differential equations*, MSJ Memoirs **28**, Mathematical Society of Japan, Tokyo, 2012.

[O4] T. Oshima, *Drawing Curves*, Mathematical Progress in Expressive Image Synthesis III, edited by Y. Dobashi and H. Ochiai, Mathematics for Industry, **24** (2016), 95–106, Springer.

[O5] T. Oshima, *Transformation of KZ type equations*, Microlocal Analysis and Singular Perterbation Theory, RIMS Kôkyûroku Bessatsu **B61** (2017), 141–162.

[O6] T. Oshima, `os_muldif.rr`, a library of the calculation of differential operators for computer algebra Risa/Asir, 2007–2022, https://www.ms.u-tokyo.ac.jp/~oshima

[O7] T. Oshima, Semilocal monodromy of rigid local systems, Formal and Analytic Solutions of Diff. Equations, Springer Proceedings in Mathematics and Statistics **256**(2018) 189-199,

[O8] T. Oshima, Counting numbers, in Japanese, Sûgaku Shobo, 2019, 226 pp.

[O9] T. Oshima, *Confluence and versal unfolding of Pfaffian systems*, Josai Mathematical Journal **12**(2020), 117–151.

[O10] T. Oshima, *Versal unfolding of irregular singularities of a linear differential equation on the Riemann sphere*, Publ. RIMS Kyoto Univ. **57** (2021), 893–920.

[OSe] T. Oshima and J. Sekiguchi, *Eigenspaces of invariant differential operators on an affine symmetric spaces*, Invent. Math. **57**(1980), 1–81.

[OSh] T. Oshima and N. Shimeno, *Heckman-Opdam hypergeometric functions and their specializations*, RIMS Kôkyûroku Bessatsu **B20** (2010), 129–162.

[Saito-Sturmfels-Takayama] M. Saito, B. Sturmfels, N. Takayama, *Groebner deformations of hypergeometric differential equations*, Algorithms and Computation in Mathematics **6**, Springer-Verlag (2000).

[Shimoyama-Yokoyama] T. Shimoyama, K. Yokoyama, *Localization and primary decomposition of polynomial ideals*, J. Symb. Comp. **22** (1996), 247–277.

[Shoup] V. Shoup, *A new polynomial factorization algorithm and its implementation*, J. Symb. Comp. **20**(1995), 364–397.

[Traverso] C. Traverso, *Groebner trace algorithms*, Proc. ISSAC '88(LNCS 358), 125-138.

[Yokoyama] K. Yokoyama, *Prime decomposition of polynomial ideals over finite fields*, Proc. ICMS, (2002), 217–227.

[Weber] K. Weber, *The accelerated Integer GCD Algorithm*, ACM TOMS, **21, 1**(1995), 111–122.