os_muldif

大島 利雄

Aug 25, 2017

目次

os_muldif.rr for Risa/Asir

A library for computing (ordinary/partial) differential operators

by Toshio Oshima

1 List of Functions

1.1 Functions related to differential operators

以下の函数は module 化され，函数名の先頭に os_md. をつけて，os_md.muldo() のように呼び出す.

1.1.1 Fundamental functions

1. muldo(p_1,p_2,[x,∂_x]|lim=n) または muldo(p_1,p_2,x|lim=n)
   muldo(p_1,p_2,[[x_1,∂_{x_1}],[x_2,∂_{x_2}],...]|lim=n)
   :: 有理函数（初等函数でもよい）係数の常（または偏）微分作用素（の行列）の積 ($\Leftarrow [\partial_x, x] = 1$)
2. muledo(p_1,p_2,[x,∂_x]) または muledo(p_1,p_2,x)
   :: Euler 型常微分作用素（の行列）の積 ($\Leftarrow [\partial_x, x] = x$)
3. transpdo(p,[[x_1,∂_{x_1}],[x_2,∂_{x_2}],...],[[y_1,∂_{y_1}],[y_2,∂_{y_2}],...]|ex=1)
   :: 微分作用素の変換（$x_i \mapsto y_i = y_i(x)$, $\partial_{x_j} \mapsto \partial_{y_j} = c_j(x) + \sum_\nu a_{j\nu}(x)\,\partial_{x_\nu}$）
4. translpdo(p,[[x_1,∂_{x_1}],[x_2,∂_{x_2}],...],mat)
   :: 微分作用素の線形座標変換（$x_i \mapsto \sum_j (mat)_{ij} x_j$）
5. appldo(p,r,[x,∂_x]) または appldo(p,r,[[x_1,∂_{x_1}],[x_2,∂_{x_2}],...])
   :: 微分作用素（の行列）の有理式や初等函数（の行列）への作用の計算
6. adj(p,[x,∂_x]) または adj(p,[[x_1,∂_{x_1}],[x_2,∂_{x_2}],...])
   :: 微分作用素（の行列） p の formal adjoint
7. sftpexp(p,[x,∂_x],q,r) または sftpexp(p,[[x_1,∂_{x_1}],...],q,r])
   :: 微分作用素 p を $q^{-r} \circ p \circ q^r$ と変換する
8. appledo(p,r,[x,∂_x])
   :: Euler 型常微分作用素の有理式への作用の計算
9. divdo(p_1,p_2,[x,∂_x]|rev=1)
   :: 常微分作用素の割り算
10. mygcd(p_1,p_2,[x,∂_x]|rev=1,dviout=n) または 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)
    :: 有理函数係数の常微分作用素（または x の多項式，または正整数）p_1 と p_2 の GCD（最大公約元）
11. mylcm(p_1,p_2,[x,∂_x]|rev=1) または mylcm(p_1,p_2,[x]|rev=1)
    mylcm(p_1,p_2,x), mylcm(p_1,p_2,0)
    :: 有理函数係数の常微分作用素（または x の多項式，または正整数）p_1 と p_2 の LCM（最小公倍元）
12. m1div(m,n,[x,∂_x]) または m1div(m,n,[x]) または m1div(m,n,x)
    :: 有理函数係数の常微分作用素（or x の多項式）の正方行列 m と有理式（or x を含まない有理式）の正方行列 n に対し，$m = R[1](\partial_x - n) + R[0]$（or $m = R[1](x - n) + R[0]$）となるリスト $R = [R[0], R[1]]$ を返す．$R[0]$ は微分（or x）を含まない．
13. qdo(p_1,p_2,[x,∂_x])
    :: 常微分方程式 $p_1 u = 0$ に対し $q_1 p_2 u = 0$ となる微分作用素 q_1 と $q_2 p_2 u = u$ となる微分作用素 q_2 のリスト $[q_1, q_2]$ を返す
14. 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)
    :: 有理函数係数の常微分作用素/1 変数多項式や整数の行列の単因子を得る
15. sqrtdo(p,[x,∂_x]) ?
    :: $x \mapsto 1/x$ で（x のべき倍を除いて）不変な微分作用素 p に対する変数変換 $x \mapsto y = x + \sqrt{x^2 - 1}$
16. toeul(p,[x,∂_x],n)

:: 確定特異点型常微分作用素を $x=n$ で Euler 型に変換

17. `fromeul`$(p,[x,p_x],n)$
:: Euler 型常微分作用素を元に戻す（`toeul`$(p,[x,\partial_x],n)$ の逆変換）
18. `expat`$(p,[x,\partial_x],n)$
:: 確定特異点型常微分作用素の $x=n$ での特性指数を求める
19. `sftexp`$(p,[x,\partial_x],n,r)$
:: 常微分作用素 p を $(x-n)^{-r}\circ p\circ(x-n)^r$ に変換する
20. `fractrans`$(p,[x,\partial_x],n_0,n_1,n_2)$
:: 常微分作用素 p に $(n_0,n_1,n_2)\mapsto(0,1,\infty)$ という一次分数変換を行う
21. `chkexp`$(p,[x,\partial_x],n,r,m)$
:: $j=0,\dots,m-1$ の全てに対し，確定特異点型常微分作用素 p の解 u_j で $(x-n)^{-(r+j)}u_j$ が $x=n$ で正則でそこでの値が 1 となるものの存在条件
22. `soldif`$(p,[x,\partial_x],n,q,m)$
:: 常微分作用素 p の $x=n$ の近傍での $z^q(1+\sum_{j=1}^{\infty}c_jz^j)$ の形の形式解に対し，長さ $m+1$ のベクトル `[1` c_1 c_2 $\cdots$ c_m`]` を返す（$z=x-n$）.
23. `okuboetos`$(p,[x,\partial_x]$`|diag=[`$c_1,c_2,\dots$`])`
:: 単独 m 階 Okubo 型方程式 $pu=0$（n 階の項の係数が n 次以下の多項式）を Okubo 型の 1 階のシステムに変換する
24. `stoe`$(p,[x,dx],m)$
:: 1 階の常微分方程式系を単独高階に直す
25. `dform`$(\ell,x$`|dif=1)`
:: 変数 $x[0],x[1],\dots$ の 1 次微分形式 $\sum\ell[i][0]d(\ell[i][1])$，または 2 次微分形式 $\sum\ell[i][0]d(\ell[i][1])\wedge d(\ell[i][2])$ の計算．`dif=1` は 1 次微分形式の外微分の計算.
26. `solpokubo`$(p,[x,\partial_x],n)$
:: 単独 Okubo 型常微分作用素 p の n 次の固有多項式と固有値を求める

1.1.2 Fractional calculus

この項は，[O2] の主要結果（基本的部分は [O1] で解説）を Risa/Asir 上で実現したものとなっている.

27. `laplace`$(p,[x,\partial_x])$ または `laplace`$(p,[[x_1,\partial_{x_1}],[x_2,\partial_{x_2}],\dots])$
:: 微分作用素 p の（部分）Laplace 変換
28. `laplace1`$(p,[x,\partial_x])$ または `laplace1`$(p,[[x_1,\partial_{x_1}],[x_2,\partial_{x_2}],\dots])$
:: 微分作用素 p の（部分）逆 Laplace 変換
29. `mc`$(p,[x,\partial_x],r)$
:: 常微分作用素 p の middle convolution $mc_r(p)$
30. `mce`$(p,[x,\partial_x],n,r)$
:: 常微分作用素 p を $(\partial_x-n)^{-r}\circ p\circ(\partial_x-n)^r$ と変換
31. `rede`$(p,[x,\partial_x])$ または `rede`$(p,[[x_1,\partial_{x_1}],[x_2,\partial_{x_2}],\dots])$
:: 微分作用素 p の reduced representative を返す
32. `ad`$(p,[x,\partial_x],f)$
:: 常微分作用素 p の ∂_x を ∂_x-f に置き換える変換
33. `add`$(p,[x,\partial_x],f)$
:: 常微分作用素 p の ∂_x を ∂_x-f に置き換える addition，すなわち `rede(ad())`
34. `vadd`$(p,[x,\partial_x],[[c_0,r_0],[c_1,r_1],\dots])$
:: versal addition `add`$(p,[x,\partial_x],\sum_{j\geq 0}\frac{r_jx^j}{\prod_{\nu=0}^{j}(1-c_\nu x)})$
35. `addl`$(p,[x,\partial_x],f)$
:: 常微分作用素の addition の Laplace 変換 `laplace1(add(laplace()))`
36. `cotr`$(p,[x,\partial_x],f)$
:: 常微分作用素 p の $x\mapsto f(x)$ による座標変換

37. rcotr(*p*,[*x*,∂*x*],*f*)
:: 常微分作用素 *p* の $x \mapsto f(x)$ による座標変換の reduced representative
38. s2sp(*p*|num=1,std=k,short=1)
:: スペクトル型を表す文字列と “数のリストのリスト”との変換
39. chkspt(*m*|mat=1) または chkspt(*m*|opt=*t*) または fspt(*m*,*t*)
:: 分割の組 *m*（スペクトルタイプ）または generalized Riemann scheme（GRS）をチェックして [*pts*, *ord*, *idx*, *fuchs*, *rod*, *redsp*, *fspt*] を返す
opt="sp", "basic", "construct", "strip", "short", "long", "sort"
40. spgen(*n*|eq=1,str=1,std=fpt=[*k*,*ℓ*],sp=*m*,basic=1)
:: 階数 *n* 以下の rigid な分割の組（あるいは与えられたものの軌道）を得る
n が 0 や負のときは，rigidity index が *n* の basic なものを得る．
n は −10 以上に対応しているが，okubo [O5] が動作する環境にしておけば，この制限はない．
41. sproot(*p*,*t*|dviout=1,only=*k*,null=1)
:: スペクトル型を与えて構成やルートの情報を示す．*t*="base", "length", "type", "part", "pair", "pairs", *sp*
42. sp2grs(*m*,*a*,*ℓ*|mat=1)
:: spectral type から generalized Riemann scheme を生成する
43. ssubgrs (*m*,*ℓ*)
:: Generalized Riemann scheme *m* の *ℓ* に対する特性指数和
44. mcgrs(*m*,[$r_1, r_2, \ldots, r_n$]|mat=1)
:: middle convolution と addition を generalized Riemann scheme に施す
45. mcop(*p*,[$r_1, r_2, \ldots, r_n$],[$x, x_1, x_2, \ldots$])
:: middle convolution と addition を (偏）微分作用素 *p* に順に施していく
46. redgrs(*m*|mat=1)
:: 常微分作用素の generalized Riemann scheme の 1-step reduction
47. getbygrs(*m*,*t*|perm=*ℓ*,var=*v*,pt=[$p_1, \ldots$],mat=1) または
getbygrs(*m*,[*t*,$s_1, s_2, \ldots$]|perm=*ℓ*,ver=*v*,pt=[$p_1, \ldots$],mat=1)
:: generalized Riemann scheme (GRS) で定義される Fuchs 型常微分方程式の解析（GRS は短縮形またはスペクトルタイプでもよい）
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 は特異点の置換または互換（cf. mperm()），var は exponents の変数（cf. sp2grs()），$p_1, \ldots$ は特異点の位置（∞ は除く）．
48. shiftop(*ℓ*,*s*|zero=1,raw=*k*,all=*t*,dviout=1)
:: rigid なスペクトル型 *ℓ* と shift *s* から shift 作用素を求める
49. conf1sp(*m*|x2=±1,conf=0)
:: スペクトル型 *m* の微分作用素の Poincare rank 1 の合流過程を示す
50. anal2sp(*m*,*l*)
:: 同時スペクトル *m* の解析
51. mc2grs(*g*,*r*|dviout=*k*)
:: $\mathbb{P}^1$ 内の 5 点の配置に対する KZ 型方程式の同時スペクトル *g* の変換
52. 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*)
:: Pfaff 形式 $du = (A_0\frac{dx}{x} + A_y\frac{d(x-y)}{x-y} + A_1\frac{d(x-1)}{x-1} + B_0\frac{dy}{y} + B_1\frac{d(y-1)}{y-1})u$ の *x* 変数での addition+middle convolution を求める（$\ell = [A_0, A_y, A_1, B_0, B_1]$）．
ℓ がスペクトル型や Riemann scheme のとき，上の後者では *s* ="GRC", "GRSC", "Pfaff", "sp", "pairs", "irreducible", "All", "swap"

53. mcmgrs(g,r|dviout=k)
:: $\mathbb{P}^1$ 内の 5 点以上の点配置に対する KZ 型方程式の同時スペクトル g の変換
54. mmc(l,[μ,a_1,...,a_n]|mult=f)
:: Schlesinger 型常微分方程式および KZ 型方程式の addition+middle convolution
55. linfrac01(ℓ|over=1)
:: $x = 0, 1, \infty, y, z, \ldots$ の一次分数変換のリスト（$\ell = [x, y]$ etc.）
56. lft01(t, ℓ)
:: $\ell = (x_1, x_2, x_3, \ldots)$ に対する特殊一次分数変換

1.1.3 Some operators

57. 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])
:: 0, 1, ∞ に確定特異点を持つ $m+n$ 階の単独 Okubo 型微分作用素を求める
58. fuchs3e([$p_{0,1}, \ldots, p_{0,n}$],[$p_{1,1}, \ldots, p_{1,n}$],[$p_{2,1}, \ldots, p_{2,n}$])
:: 0, 1, ∞ に確定特異点を持つ n 階の Fuchs 型微分作用素を求める
59. ghg([$p_{1,1}, p_{1,2}, ..., p_{1,m}$],[$p_{2,1}, p_{2,2}, ..., p_{2,n}$])
:: 一般超幾何函数 ${}_mF_n(p_1; p_2; x)$（cf. seriesHG()）の満たす微分作用素
60. even4e([$p_{1,1}, p_{1,2}, p_{1,3}, p_{1,4}$],[$p_{2,1}, p_{2,2}$])
:: 4 階 even family (Rigid)
61. odd5e([$p_{1,1}, p_{1,2}, p_{1,3}, p_{1,4}, p_{1,5}$],[$p_{2,1}, p_{2,2}$])
:: 5 階 odd family (Rigid)
62. rigid211([$p_{0,1}, p_{0,2}$],[$p_{1,1}, p_{1,2}$],[q_0, q_1])
:: Type 211,211,211
63. 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)
64. eofamily([$p_{0,1}, p_{0,2}$],[$p_{1,1}$],[$p_{2,1}, \ldots, p_{2,n}$])
:: even/odd family (obsolete)
65. ev4s(p_1,p_2,p_3,p_4,p_5)
:: Heckman-Opdam 超幾何の (BC_2, BC_1) 型制限常微分 (Rigid)
66. b2e(p_1,p_2,p_3,p_4,p_5)
:: Heckman-Opdam 超幾何の (BC_2, A_1) 型制限常微分 (Non Rigid)
67. heun([a, b, c, d, e],p,r|)
:: Heun の微分方程式を与える．r はアクセサリパラメータ

1.2 Useful functions

以下の函数は module 化され，函数名の先頭に os_md. をつけて os_md.myhelp() のように呼び出す.

1.2.1 Extended function

68. myhelp(h)
:: os_muldif.rr のマニュアルを表示する
69. chkfun(f,s)
:: 函数 f（= 文字列）が定義済みかどうか調べ，未定義なら load(s) を実行
70. isMs()
:: Micorosoft Windows 環境かどうか調べる
71. isyes(p|set=ℓ)
:: 1 か 0 を返す函数を定義し，それを使う
72. isall(f,m)
:: m の要素に p 対して $f(p)$ が 0 となるものが存在すると 0，そうでなければ 1 を返す

73. ptype(p,ℓ)
:: ℓ は変数，または変数のリストで，それのみを変数とみなした type() を返す
74. keyin(s)
:: s を表示し，1 行のキー入力を待って，それを文字列として返す
75. showbyshell(s)
:: shell でコマンド s を実行した標準出力の結果を Risa/Asir で表示
76. getbyshell(s)
:: shell でコマンド s を実行した標準出力の結果をファイルとして得る
77. makev([ℓ_1,ℓ_2,...]|num=1)
:: $\ell_1, \ell_2 \ldots$ を合わせて一つの変数名を作る
78. shortv(p,[v_1,v_2,...]|top=v)
:: p の添字番号つき不定元 $v_1, \ldots$ を一文字の v 以下の変数名に変える
79. makenewv(ℓ|var=v,num=n)
:: ℓ に使われていない新しい不定元を生成する
80. isvar(p)
:: p が変数かどうか調べる
81. varargs(p,x|all=t)
:: 式 p に含まれる初等函数の函数子とその変数に現れる不定元のリストを返す
82. pfargs(p,x|level=t)
:: 式 p に現れる x 変数を含んだ初等函数と引数のリストを返す
83. isdif(p)
:: p が微分作用素と推測されるときはその変数と微分の組のリストを返し，そうでなければ 0 を返す
84. mysubst(r,[v_1,r_1]|inv=1) または mysubst(r,[[v_1,r_1],...]|inv=1)
:: subst(r,v_1,r_1,...) と同等．r が複雑で r_1 が有理式のときに特に有効.
85. myswap(p,[x_1,x_2,...,x_n])
:: 有理式またはそのリストなどの不定元の巡回置換 $(x_1, x_2, \ldots, x_n)$
86. mulsubst(r,[[$p_{1,0}, p_{1,1}$],[$p_{2,0}, p_{2,1}$],...]|inv=1)
:: 有理式またはそのリスト，ベクトル，行列 r に複数の代入 $p_{j,0} \mapsto p_{j,1}$ $(j = 1, 2, \ldots)$ を同時に行う
87. fmult(f,m,ℓ,n|$\cdots$)
:: $m_i \mapsto m_{i+1} = f(m_i, \ell[i], n[0], n[1], \ldots | \cdots)$ という変換（$m_0 = m$）の最終結果 $m_{length(l)}$ を返す
88. mtransbys(f,m,ℓ|$\cdots$)
:: スカラーに関する変換 $f()$ をリスト，ベクトルまたは行列 m に拡張する
89. mmulbys(f,m,n,ℓ|$\cdots$)
:: 和が定義された objects の 2 つに対して 1 つの object を与える演算 f を，objects を成分とするベクトルまたは行列 m と n の演算に拡張する
90. cmpsimple(p,q|comp=t)
:: 式 p と q の簡単さを比較
91. simplify(p,ℓ,t|var=[x_1,x_2,...])
:: $\ell = [\ell_0, \ell_1]$ のときは，p（の各要素毎）に subst(*,ℓ_0,ℓ_1) を調べてより簡単なら置き換える $(t = 1 \sim 7)$．$\ell = [\ell_1]$ で ℓ_1 が多項式のときは，ℓ_1 に一次に含まれる含まれる変数の線形関係式とみて簡単化する．複数調べるときは ℓ をリストや多項式のリストとする.
92. getel(m,i)
:: m がリスト，ベクトル，行列で i が非負整数なら $m[i]$ を返す
93. evalred(r|opt=[[s_1,t_1],[s_2,t_2]...])
:: sin(0), cos(0), exp(0) などを 0, 1, 1 などの整数に置き換える
94. evals(r|del=s,raw=1)
:: 文字列あるいは函数を評価する.
95. myeval([r,[x_1,f_1,v_1],[x_2,f_2,v_2],...])
:: os_md.myeval(subst([r,[x_2,f_2,v_2],...],x_1,f_1(os_md.myeval(v_1)))) を返す

os_myeval([r])=map(eval(r))

96. mydeval([r,[x_1,f_1,v_1],[x_2,f_2,v_2],...])
:: os_md.mydeval(subst([r,[x_2,f_2,v_2],...],x_1,f_1(os_md.mydeval(v_1)))) を返す
os_md.mydeval([r])=map(deval(r))
97. myval([r,[x_1,f_1,v_1],[x_2,f_2,v_2],...])
:: myeval() と同様な函数であるが，可能な限り正確な値を返す
98. f2df(f|opt=n)
:: 函数 f から sin などの函数子を除いて myeval() や mydeval() の引数の**リスト形式函数**に変換
99. todf(f,[v_1,...,v_n]) todf([f,[ops]],[v_1,...,v_n])
:: 函数子 f で変数が $v_1,\ldots,v_n$ のリスト形式函数を得る（n は f の引数の数）
100. df2big(f|inv=1)
:: リスト形式函数 f を倍精度浮動小数点計算から bigfloat 計算へ変更する
101. compdf(f,x,g) compdf(f,[x_1,x_2,...],[g_1,g_2,...])
:: リスト形式函数 f の変数 x にリスト形式函数 g を代入したリスト形式函数を返す
102. cutf(f,x,[[x_1,v_1],[x_2,v_2],...,[x_n,v_n]]) cutf(f,x,[t,[x_1,v_1],...,[x_n,v_n]])
:: 函数 f の変数が特定の範囲のとき函数値の変更を行い，その値またはリスト形式函数を返す
103. periodicf(f,[a,b],x) periodicf(ltov([g_1,g_2,...,g_n]),c,x)
:: x を変数とする函数 $f|_{[a,b)}$ を周期関数に拡張した函数にする
104. cmpf([f,[a,b]]|exp=c) cmpf(x)
:: 積分区間をコンパクト閉区間 $[0,1]$ に直した函数にする
105. myfeval(f,x) (f,a) myfeval(f,[x,a]) myfeval(f,[[x_1,a_1],...])
106. myfdeval(f,a) myfdeval(f,[x,a]) myfdeval(f,[[x_1,a_1],...])
:: myeval() や mydeval() の引数のリスト形式函数 f の変数に a を代入して値を得る
107. myf2eval(f,x,y)
108. myf2deval(f,x,y)
:: myeval() や mydeval() の引数のリスト形式 2 変数函数 f に代入して値を得る
109. myf3eval(f,x,y,z)
110. myf3deval(f,x,y,z)
:: myeval() や mydeval() の引数のリスト形式 3 変数函数 f に代入して値を得る
111. execproc(ℓ|all=1,var=k)
:: リスト形式手続きの実行
112. fsum(f,[m,n,d]|df=1,subst=1) fsum(f,[x,m,n,d]|df=1,subst=1)
:: 一般項が $f(x)$ で与えられる級数の和 $\sum_{k=0}^{[\frac{m-n}{d}]} f(m+kd)$ を返す
113. fint(f,n,[t_1,t_2]]|cpx=1,exp=c,int=k,prec=v) fint(f,n,ℓ|$\cdots$)
:: 複素積分を含む数値積分
114. fimag(f,x|inv=g)
:: 複素変数の指数関数を実変数の関数に変換
115. trig2exp(f,x|inv=g)
:: 三角関数と指数関数の変換と簡単化
116. isshortneg(f)
:: f と $-f$ を表現する文字列の長さの比較
117. fshorter(f,x)
:: x の三角関数の簡単化
118. fzero(f,[x,x_1,x_2]|mesh=m,dev=d,zero=1,trans=1,cont=1)
:: 実数値函数 f の零点を求める
119. fmmx(f,[x,x_1,x_2]|mesh=m,dev=d,mmx=1,zero=1,trans=1,cont=1,dif=1)
:: 実数値函数 f の極値を求める
120. flim(f,v|prec=c,init=t)
:: 変数 x の実数値関数 f の極限値を求める

121. fcont(f,[x,x_1,x_2]|mesh=m,dev=d,zero=1,trans=1,dif=1) ?
:: 実数値函数 f の不連続点や滑らかでない点を求める
122. fresidue(p,q|cond=[f_1,f_2,...],sum=1)
:: 多項式 q を分母とする有理式 p/q の特異点と留数の組のリスト（z が変数で，p は正則函数）

1.2.2 Numbers

123. abs(p)　abs([p,$prec$])
:: 整数または実数または複素数 p の絶対値を返す
124. sgn(n|val=1)
:: 数 n または置換 n の符号を返す
125. calc(p,[s,q])　calc(p,s)
:: 数や有理式に対して演算を施す
126. isint(p)
:: p が整数かどうか調べる
127. israt(p)
:: p が有理数かどうか調べる
128. iscrat(p)
:: p が数でその実部と虚部が共に有理数かどうか調べる
129. isalpha(n)
:: 整数 n がアルファベットの文字コードかどうか調べる
130. isnum(n)
:: 整数 n が数字 0〜9 の文字コードかどうか調べる
131. isalphanum(n)
:: 整数 n がアルファベットまたは数字の文字コードかどうか調べる
132. isdecimal(s)
:: 文字列 s が整数または小数を表しているかどうか調べる
133. nthmodp(a,n,p)
:: $a^n \mod p$ を返す（a は整数で n, p は自然数）
134. issquaremodp(a,p|power=n)
:: 平方剰余を調べる（ルジャンドルの平方剰余記号 $\left(\frac{a}{p}\right)$，n は平方を一般べきに）
135. rootmodp(a,p|power=n)
:: p を法として a の平方根（一般には n 乗根）を求める
136. primroot(p|all=1,ind=a)
:: 奇素数またはそのべき p の原始根やそれを底とする指数をもとめる
137. rabin(p,q)
:: p に対し q を底とするミラー・ラビンの素数判定を行う（素数，擬素数なら 1 を返す）
138. cfrac(x,n)
:: 有理数あるいは実数を連分数展開する（n は項数）
139. cfrac2n(l|loop=m)
::（循環）連分数を通常の形に直す
140. sqrtrat(r)
:: 有理数（または実部と虚部が有理数の複素数）の平方根を得る
141. sqrt2rat(r|mult=1)
:: 平方根や虚数を含んだ分数の有理化
142. sint(r,p|str=t,sqrt=1,zero=0)
:: 実数 r または複素数，リストや行列（ネスト対応）などの成分の実数を小数点以下 p 桁に丸める
143. frac2n(n|big=1)
:: 分数を実数になおす（r は複素数，有理式やそのリストや行列などでもよい）

1.2.3 Polynomials and rational functions

144. radd(*p*,*q*)
:: 有理式（の行列）p と q の和を既約有理式（の行列）の形で計算する
145. rmul(*p*,*q*)
:: 有理式（の行列）p と q の積を既約有理式（の行列）の形で計算する
146. polbyroot([$p_1,p_2,...,p_n$],*x*)
:: 多項式を根で与える
147. polbyvalue([[a_1,b_1],...,[a_n,b_n]],*x*)
:: x の $n-1$ 次多項式を n 個の点 $x=a_i$ での値 b_i で与える
148. pgen([[x_1,n_1],[x_2,n_2]...],*a*|sum=*n*,shift=*m*,sep=1,num=1)
:: 係数が a_* で x_i が n_i 次，全体で n 次以下の $x_1,\ldots$ の一般多項式を作る
149. rpdiv(*p*,*q*,*x*)
:: x の多項式の割り算
150. easierpol(*p*,*x*) または easierpol(*p*,[$x_1,x_2,\ldots$])
:: 有理式係数の x の多項式の係数に x を含まない有理式をかけて，係数の最大公約元が 1 の整数係数の多項式に変換
151. getroot(*p*,*x*|mult=1,cpx=1)
:: 多項式の根を有理式または有理数の平方根の範囲で求める
152. polroots(*p*,*x*|comp=*t*,err=*r*,lim=ℓ)
:: 変数 x の 1 変数多項式の根，多変数多項式の共通根（実根・虚根）の（近似）値を返す
153. fctri(*p*)
:: 実部と虚部が有理数係数の 1 変数多項式を，その範囲で既約分解する
154. polinsym(*p*,[$x_1,\ldots,x_n$],*s*)
:: $(x_1,\ldots,x_n)$ の対称有理式を基本対称式で表す
155. polinvsym(*p*,[$x_1,\ldots,x_n$],*s*)
:: polinsym(*p*,[$x_1,\ldots,x_n$],*s*) の逆函数
156. pol2sft(*p*,*x*|sft=*t*)
:: shifted power 多項式を与える
157. polinsft(*p*,*x*)
:: shifted power 多項式に直す（pol2sft() の逆変換）
158. sftpow(*p*,*n*)
159. sftpowext(*p*,*n*,*s*)
:: p の s shifted n power $\prod_{\nu=1}^{n}(p+(\nu-1)s)$ を返す
160. binom(*p*,*n*)
:: $p(p-1)(p-2)\cdots(p-n+1)/n!$ を返す
161. expower(*p*,*r*,*n*)
:: $(1+p)^r$ の展開を p^n まで求める
162. orthpoly(*n*|pol="*type*") orthpoly([*n*,*a*,...]|pol="*type*")
:: 変数 x の n 次直交多項式を返す
163. schurpoly ([m_1,m_2,...]|var=*v*)
:: $m_1\geq m_2\geq\cdots\geq m_n\geq 0$ のウェイトの Schur 多項式を返す
164. seriesMc(*f*,*k*,*v*|evalopt=[[s_1,t_1],[s_2,t_2]...])
:: 函数 f の変数または変数のリスト v に対する k 次の項までの Maclaurin 展開を求める
165. seriesHG([a_1,a_2,...],[b_1,b_2,...],*p*,*k*)
seriesHG([[a_1,...],[b_1,...],[c_1,...]],[[d_1,...],[e_1,...],[f_1,...]],[*x*,*y*],*k*)
:: 一般超幾何級数 ${}_mF_n(a_1,a_2,\ldots,a_m;b_1,b_2,\ldots,b_n;p)$ の p^k 次の項まで求める
または 2 変数の級数 $\sum_{0\leq i+j\leq k}\frac{(a_1)_{i+j}\cdots(b_1)_i\cdots(c_1)_j\cdots}{(d_1)_{i+j}\cdots(e_1)_i\cdots(f_1)_j\cdots}\frac{x^iy^j}{i!j!}$ を返す.
166. fctrtos(*r*|var=ℓ,rev=1,dic=1,TeX=*f*,dviout=1,lim=*n*,small=1,pages=1,add=*s*)

:: 有理式を因数分解した形の文字列に変換する

167. tohomog(r,[$x_1,x_2,\ldots$],y)
:: $(x_1,x_2,\ldots)$ の有理式に変数 y を導入して $(y,x_1,x_2,\ldots)$ の斉次式にする

168. substblock(p,x,q,y)
:: x の多項式 p,q に対し，$y=q$ とおいて p を x の次数が mydeg(q,x) 未満の (x,y) の多項式に直す

169. invf([$p_1,\ldots,p_n$],[$x_1,\ldots,x_n$],[$y_1,\ldots,y_n$])
:: $y_j=p_j(x)\quad(j=1,\ldots,n)$ を $x_j=q_j(y)\quad(j=1,\ldots,n)$ と解く

170. mydeg(p,x|opt=1)
:: deg(p,x) と同じ．p は行列や配列で係数は有理式でよい．

171. mymindeg(p,x|opt=1)
:: p がスカラーのときは mindeg(p,x) と同じ．係数は有理式でよいが，p が行列などのスカラーでないときは 0 以外の成分の最小次数を返す．

172. iscoef(p,f)
:: p の係数の全てが $f(*)\neq 0$ を満たすかどうかチェックする

173. mycoef(p,n,x)
:: coef(p,n,x) と同じ．n, l は同じ長さのリストでもよい．p は行列や配列で係数は有理式でよい．

174. pcoef(p,m,q)
pcoef(p,m,[[$x_1,..,x_n$],[$m_1,\ldots,m_n$]])
:: 多項式 p^m を展開したときの単項式 q に対する係数を返す

175. pfctr(p,x)
:: x の多項式または有理式 p の因数分解

176. cterm(p|var=[$x,y,\ldots$])
:: 多項式の定数項を返す．変数を指定可能．

177. terms(p,[$x,y,\ldots$]|rev=1,dic=1)
:: 多項式の存在する項の次数とべき指数のリストを返す

178. polcut (p,n,[$x,y,\ldots$]|top=m)
:: 変数のリスト [$x,y,\ldots$] の多項式 p から次数が（m 以上）n 以下でない項を削除

179. mydiff(p,x)
:: diff(p,x) と同じ．p は行列や配列で係数は有理式でよい．x もリストでよい．

180. myediff(p,x)
:: ediff(p,x) と同じ．p は行列や配列で係数は有理式でよい．

181. ptol(p,x|opt=0)
:: x の多項式 p の係数のリストを返す

182. pfrac(p,x|root=2,dviout=1,TeX=1)
:: x の有理式 p を部分分数展開し，分子，分母（多項式とべき）の組のリストを返す

183. lpgcd([$p_1,p_2,\ldots$])
:: 多項式 $p_1,p_2,\ldots$ の共通因子を返す

184. jacobian([$f_1,\ldots,f_n$],[$x_1,\ldots,x_n$]|mat=1)
:: 関数の組 $(f_1,\ldots,f_n)$ の変数 $(x_1,\ldots,x_n)$ についてのヤコビアンまたはヤコビ行列を返す．

185. hessian([f,[$x_1,\ldots,x_n$]|mat=1)
:: 関数の組 f の変数 $(x_1,\ldots,x_n)$ についてのヘシアンまたはヘッセ行列を返す．

186. wronskian[$f_1,\ldots,f_n$],x|mat=1)
:: 関数の組 $(f_1,\ldots,f_n)$ の変数 x についてのロンスキアンまたはロンスキー行列を返す．

187. prehombf(p,q|mem=$\pm$1)
:: 概均質ベクトル空間の相対不変式 p の b 函数を得る．q は双対多項式．

188. intpoly(p,x|exp=c,cos=c,sin=c)
:: 変数 x の多項式 p（またはそれと指数関数, 対数関数や三角関数の積）や有理式の原始函数を返す

189. integrate(f,x|dumb=k,dviout=p,log=1,frac=t,I=[a,b])
:: 関数 f を変数 x について不定積分する

190. powsum(n)
:: $1^n + 2^n + \cdots + m^n$ の m を x で置き換えた $n+1$ 次多項式を返す
191. bernoulli(n)
:: n 次の Bernoulli 多項式 $B_n(x)$ を返す

1.2.4 Functions with real/complex variables

以下の数値函数では変数（引数）を**不定変数**にすると対応するリスト形式函数を返す.

192. frac(x)
:: 実数 x の小数部分
193. erfc(x)　erfc([x,$prec$])
:: 相補誤差函数 $\mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt$
194. fouriers([a_0,a_1,...,a_m],[b_1,b_2,...,b_n],z|cpx=1,sum=f,y=t,const=c)
fouriers([f,m],[g,n],z|cpx=1,sum=f,y=t,const=c)
:: 有限 Fourier 級数 $a_0 + a_1 \cos z + a_2 \cos 2z + \cdots + a_m \cos mz + b_1 \sin x + b_2 \sin 2z + \cdots + b_n \sin nz$
195. myexp(z)
:: 指数関数 $\exp(z)$
196. mysin(z)
:: 三角関数 $\sin(z)$
197. mycos(z)
:: 三角関数 $\cos(z)$
198. mytan(z)
:: 三角関数 $\tan(z)$
199. myasin(z)
:: 逆三角関数 $\mathrm{asin}(z)$
200. myacos(z)
:: 逆三角関数 $\mathrm{acos}(z)$
201. myatan(z)
:: 逆三角関数 $\mathrm{atan}(z)$
202. mylog(z)
:: 対数函数 $\log(z)$
203. mypow(z,w)
:: 巾関数 z^w
204. myarg(z)
:: 偏角 $\arg(z)$
205. sqrt(z)　sqrt([z,$prec$])
:: z の平方根を与える
206. arg(z)　arg([z,$prec$])
:: z の偏角
207. gamma(z)　gamma([z,$prec$])
:: ガンマ函数 $\Gamma(z)$
208. digamma(z)　digamma([z,$prec$])
:: ディガンマ函数 $\psi(z) = \frac{\Gamma'(z)}{\Gamma(z)}$
209. lngamma(z)　lngamma([z,$prec$])
:: $\log(\Gamma(z))$
210. dilog(z)
:: ダイログ函数 $\mathrm{Li}_2(z) = \sum_{k=1}^\infty \frac{z^k}{k^2}$
211. zeta(s)　zeta([s,$prec$])
:: zeta 函数 $\zeta(s)$

212. eta(τ)　eta([τ,*prec*])
:: Dedekind の Eta 函数 $\eta(\tau) = e^{\frac{\pi i \tau}{12}} \prod_{m=1}^{\infty}(1 - e^{2\pi i m \tau})$
213. jell(τ)　jell([τ,*prec*])
:: Elliptic j-invariant $j(\tau)$

1.2.5 Lists and vectors

214. llsize([$m_1, m_2, \ldots$])
:: リスト m_j のリストに対し，成分 m_j の個数と m_j の要素の最大個数を返す
215. findin(m,[$\ell_0, \ell_1, \ldots$])
:: m に等しい要素を $\ell_0, \ell_1, \ldots$ から探す
216. countin(s,m,[$\ell_0, \ell_1, \ldots$]|step=k)
:: s 以上 m 以下の $\{\ell_0, \ell_1, \ldots\}$ の要素の個数，間隔 m での個数分布表を返す
217. delopt(ℓ,s|inv=1)　delopt(ℓ,[s_1,s_2,...]|inv=1)
:: オプションリスト（リストのリスト）からオプション s（複数指定可）を取り除く，または抜き出す
218. mycat([$\ell_1, \ldots, \ell_m$]|delim=s)
219. mycat0([$\ell_1, \ldots, \ell_m$],t|delim=s)
:: $\ell_1, \ldots, \ell_m$ を表示する
220. fcat(f,s|exe=1)
:: ファイル f に s を書き出す
221. vtozv(v)
:: 有理式のベクトルをスカラー倍して単純化する
222. l2p(l,v|size=s)
:: 係数のリスト l を与えて v の多項式を返す．またその逆変換を返す．
223. mulseries(v_1,v_2)
:: ２つのベクトルをべき級数とみなして，その積のベクトルを返す
224. pluspower(p,x,r,m)
:: $(1+p)^r$ の x に関するべき級数展開を第 m 項まで求める
225. average(ℓ|sint=k)
:: 実数のリストに対し，平均値や標準偏差を求める
226. vprod(v_1,v_2)
227. dvprod(v_1,v_2)
:: 2 つのベクトル（リストでもよい）の内積を返す
228. llbase(v,ℓ)
:: 変数 $\ell[0], \ell[1], \ldots$ の一次方程式のベクトル v の標準変換を行う（例は lsol() の項を参照）
229. lsol(v,l)
:: 変数 $\ell[0], \ell[1], \ldots$ に関する連立一次方程式を解く
230. lnsol(v,l)
:: 変数 $\ell[0], \ell[1], \ldots$ に関する連立一次方程式の有理数解を求める
231. lchange(ℓ,k,v|flat=1)
:: （多重）リスト ℓ の k で指定した位置の成分を v に置き換える
232. lmax([$m_1, m_2, \ldots$])
:: $m_1, m_2, \ldots$ の中の最大のものを返す
233. lmin([$m_1, m_2, \ldots$])
:: $m_1, m_2, \ldots$ の中の最小のものを返す
234. lgcd([$m_1, m_2, \ldots$]|poly=1])
:: $m_1, m_2, \ldots$ の最大公約数（元）を返す
235. llcm([$m_1, m_2, \ldots$]|poly=1])
:: $m_1, m_2, \ldots$ の最小公倍数（元）を返す
236. ldev(ℓ,s)

:: リスト s の整数倍をリスト ℓ に加えて，成分の絶対値を最小にする

237. lsort(ℓ_1,ℓ_2,t|c_1=[$c_{1,0}$,$c_{1,1}$,...],c_2=[$c_{2,0}$,$c_{2,1}$,...])
:: リスト ℓ_1 に対し，ℓ_2 との合併，共通部分，または ℓ_2 や共通部分を除く
その他，表形式データの操作
t = "cup", "setminus", "cap", "reduce", "sum", "subst"

238. msort(ℓ,s)
:: ベクトルやリストの成分を（多重）キー s に従ってソートする

239. addIL([a,b],[[a_1,b_1],[a_2,b_2],...]|in=t)
:: 有限区間の合併集合と有限区間の和集合の計算

240. vnext(v)
:: ベクトルの成分を並べ替え，辞書式順序で次のベクトルに変換する

241. vgen(v,w,m|opt=0)
:: 成分の和が m の非負成分のベクトル w を順に生成する

242. paracmpl(l,t|lim=1,low=1,para=[$v_1,v_2,\ldots$])
:: 有理的に t に依存したリスト l の成分（たとえば有理式）で張られる空間の正則完備化

1.2.6 Matrices

243. dupmat(m)
:: 成分が有理式の行列またはベクトル m の複製を作る

244. m2v(m)
:: 行列 m の成分を 1 行目から順に並べてベクトルに変換する

245. m2l(m|flat=1)
:: 有理式，ベクトルあるいは行列の m の成分を順に並べてリストにする

246. m2lv(m)
:: 行列 m の行ベクトルを並べてリストにする

247. m2ll(m)
:: 行列 m を行ベクトルをリストにかえたものを成分とするリストにする

248. lv2m(ℓ|fill=n)
:: 行ベクトル（行成分のリストでも可）のリスト ℓ から行列を作る

249. s2m(s)
:: 文字列での有理数成分の行列表現や，列のリストから行列を作る．必要最低サイズの行列になる

250. c2m(l,v|pow=t)
:: 基底の変換から係数行列を作る

251. mperm(m,[σ_0,σ_1...],[τ_0,τ_1,...])
mperm(m,[[σ_0,σ_1]],[[τ_0,τ_1]]), mperm(m,[σ,[m_1]],[τ,[m_2]])
:: 行列 m から小行列 $\left(m_{\sigma_i\tau_j}\right)$ を作る，または置換行列（または互換）で変換する

252. mtranspose(m)
:: 行列 m の転置行列を求める（m はリストのリストでもよい）

253. madjust(m,w|null=n)
:: 行列 m の列数を w に調整する

254. mpower(m,n)
:: 行列 m の n 乗を求める

255. mtoupper(m,n|opt=t,step=1,dviout=1,pages=1,tab=k,lim=w)
:: 行列 m に対し，行基本変形を行って，行の先頭からの 0 の成分の個数が下の行の方へ狭義単調増加となるようにする．最後の n 列は無視．

256. mytrace(m)
:: 行列の trace を返す．

257. mydet(m)
:: det(m) と同じ．ただし成分は有理式でもよい．

258. mydet2(m)
:: det(m) と同じ．ただし成分は有理式でもよい．
259. myrank(m)
:: 行列 m の rank を求める（例は mtoupper() の項）
260. mykernel(m|opt=1)
:: 行列 m の転置行列の kernel の基底を求める
261. myimage(m|opt=1)
:: 行列 m の転置行列の像の基底を求める
262. mymod(v,[$v_1,\ldots,v_k$]|opt=ℓ)
:: ベクトル $v_1,\ldots,v_k$ で張られる空間の商空間への v の射影を求める（例は myimage() の項）
263. mmod(m,[$v_1,\ldots,v_k$]|opt=1)
:: ベクトル $v_1,\ldots,v_k$ で張られる空間の商空間への線形写像 m の射影
264. myinv(m)
:: 正方行列 m の逆行列を求める
265. madj(g,m)
:: 行列 gmg^{-1} を計算する．m は行列のリストか行列のベクトルでもよい．
266. diagm(m,a)
:: 対角行列を作成する
267. mgen(m,n,a,s|sep=1)
:: size $m\times n$ の一般行列を作成する
268. unim(s|abs=m,num=n,both=1,int=1,rank=r,conj=1,res=1,wt=w,dviout=t,lim=w)
:: 行列式 1 でサイズ s の整数行列をランダムに生成する，行列 s をランダムに変換する
269. newbmat(m,n,[[$r_{00},r_{01},\ldots$],[$r_{10},\ldots$],$\ldots$]|null=t)
:: ブロック行列を作成する
270. meigen(m|mult=1)
:: 行列 m の固有値を返す
271. mdsimplify(m|show=1,type=ℓ)
:: 有理式正方行列 m，またはそのリストやベクトルを対角行列で簡単化
272. transm(m|dviout=1)
:: 行列 m の基本変形をインタラクティブに行う

1.2.7 Strings

273. str_char(s,n,t)
:: 文字列 s の n 文字以降に文字列 t の先頭文字が最初に現れる場所を返す（str_chr() の拡張）
274. str_pair(s,n,t_1,t_2|inv=1)
:: 文字列 s の n 文字以降で，文字（列）t_2 の出現回数が t_1 の出現回数を超える最初の位置を返す
275. str_str(s,t|top=n,end=m,sjis=1)
:: 文字列 s に部分文字列 t が最初に現れる場所を返す
276. str_cut(s,m,n)
:: 文字列 s の先頭から m 文字をスキップして，それ以降 $n-m+1$ 文字を返す
277. str_subst(s,s_0,s_1|sjis=1,raw=1) または
str_subst(s,[$s_{00},s_{01},\ldots$],[$s_{10},s_{11},\ldots$]|inv=1,sjis=1,raw=1)
str_subst(s,[[s_{00},s_{10}],[s_{01},s_{11}],$\ldots$],0|sjis=1)
:: 文字列 s に含まれる部分文字列 s_0 を s_1 で全て先頭から順に置き換える
278. str_times(s,n)
:: 文字列 s（またはリスト）を n 回繰り返した文字列（またはリスト）を返す
279. strip(s,t_1,t_2)
:: 文字列の外側の括弧を外す
280. s2os(s)

:: 文字列を入力可能な文字列に変換

281. sjis2jis(l)
:: 文字コードの整数のリストの先頭 2 つを ShiftJIS から JIS に変換

282. jis2sjis(l)
:: 文字コードの数字のリストの先頭 2 つを JIS から ShiftJIS に変換

283. s2euc(s)
:: ShiftJIS または JIS 文字列を EUC 文字列に変換

284. s2sjis(s)
:: EUC または JIS 文字列を ShiftJIS 文字列に変換

285. str_tb([$s_0, s_1, \ldots$],tb) または str_tb(0,tb)
:: テキスト用バッファを作成（$tb = 0$）し，そこ（戻り値 tb）に文字列 $s_0, \ldots$ を順に追加する．最初の引数を 0 として文字列を取り出す．

286. l2os(l)
:: リストを入力可能な文字列に変換

287. r2os(l)
:: 数式を eval_str() で入力可能な文字列に変換

288. r2ma(s)
:: Mathematica の形式の数式に変換

289. evalma(s|inv=1)
:: Mathematica の数式を読み込む

290. readcsv(s|eval=[$n_1, n_2, \ldots$],eval=n,sp=1,col=c,null=t,eq=1)
:: CSV 形式のデータをリスト形式で読み込む

291. tocsv(l|null=n,exe=f)
:: リストのリストまたは行列を CSV 形式のデータに変換

1.2.8 Permutations

292. ldict(n,m|opt=t)
:: $\{0, 1, 2, \ldots, m-1\}$ を並べ替えて辞書式順序で $n+1$ 番目のリストを返す

293. ndict(ℓ|opt=t)
:: $\{\ell_0, \ell_1, \ldots, \ell_{m-1}\}$ の並べ替えのリスト ℓ が何番目かを返す（最初は 0 番）

294. nextsub([$a_0, \ldots, a_{m-1}$],n)
:: $\{0, \ldots, n-1\}$ の m 個の部分集合を並べたとき，$\{a_0, \ldots, a_{m-1}\}$ の次の部分集合を返す．最後を与えた時は 0 を返す．

295. nextpart(ℓ)
:: 自然数の分割のリスト ℓ に対し，辞書式順序で次に大きなものを返す

296. transpart(ℓ)
:: 自然数の分割のリスト（Young 図式に対応）ℓ に対し，その双対を返す（例は nextpart() を参照）

297. trpos(a,b,n)
:: 互換 (a, b) にあたる n 次置換群の元を返す

298. sprod(s,t)
:: 置換群の積を返す

299. sinv(s)
:: 置換 s の逆元を返す

300. slen(s)
:: 置換 s の長さを返す

301. sord(s,t)
:: 置換を Bruhat order で比較する

1.2.9 TEX

302. my_tex_form(p|subst=[t_0, t_1],frac=f,root=r,ket=k)

:: print_tex_form(p) の戻り値から文字列置換や不要部分削除を行い，読みやすいソースに変換

303. show(p|opt=ℓ)
:: p を dviout で適切に表示する

304. dviout(p|clear=1,keep=1,delete=t,fctr=1,mult=1,subst=[$s0, s1$],eq=k,title=s)
:: p を dviout で表示する

305. dviout0(ℓ) または dviout0([$\ell_1, \ell_2, \ldots$]) または dviout0(ℓ|opt=s)
:: TEX での表示のための内容削除などの基本操作

306. verb_tex_form(p)
:: p を LATEX で表現可能な文字列にする

307. monotos(p)
:: 有理式を文字列に変換．単項式以外では (と) で囲む

308. monototex(p|minus=1)
:: 有理式を TEX の文字列に変換．単項式以外では (と) で囲む

309. rtotex(p)
:: 数式を TEX の文字列に変換．1 文字を越えるときは { と } で囲む

310. texsp(s)
:: TEX の文字列 s の最後が TEX のキーワードのとき，s の末尾に空白をつける．

311. texbegin(t,s|opt=u)
:: TEX の \begin{t}[u] s \end{t} というソースを出力する

312. texcr(k)
:: 127 以下の非負整数 k に応じて TEX における数式の改行の文字列を返す

313. texket(s|all=t)
:: TEX のソース s における括弧のサイズを可変にする

314. ltotex(l|opt=s,pre="string",cr="cr",small=1,lim=ℓ,var=v)
:: リストまたはベクトルを s = "spt" のとき重複度つきのリストとみて\left\{... または s = "GRS" のとき \begin{Bmatrix} ⋯ の形の Riemann scheme とみて TEX の文字列に変換など

315. mtotex(m|small=1,2, null=1,2, sp=1,2, idx=0,1, mat=s, var=ℓ, raw=1, lim=n)
:: 行列またはベクトルを TEX の文字列に変換するが，成分が有理式のときは因数分解した形にする

316. divmattex(s,l)
:: 行列の TEX のソース s の分割や列の並べ替えを行う

317. smallmattex(s)
:: TEX のソースで ( ) や { } で囲まれた行列を小サイズに変換する

318. texlen(s)
:: LATEX の数式の横幅を推測して文字数で返す

319. texlim(s, n|del=s_0,cut=s_1)
:: 長い TEX の数式を，複数行に分割する

1.2.10 Lines and curves

320. ladd(u,v,t)
:: ベクトルまたはリストの成分の和（$t = 1$）や差（$t = -1$）を成分とするリストを返す

321. dnorm(v)
:: ベクトルまたはリストのノルム，または 2 点間の距離を返す

322. mrot(θ|deg=1)　mrot([θ'_z,θ_y,θ_z]|deg=1,conj=1)
:: 角度 θ の回転行列を返す

323. dvangle(v_1,v_2)　dvangle([u_1, u_2, u_3],0)
:: ベクトルまたはリストの鋭む角度の余弦を返す

324. ptaffine(m,ℓ|org=v,shift=w,arg=θ,deg=θ,proc=1)
:: 実数の組（座標）のリストを結合する，またはアフィン変換（こちらは描画実行形式も可）を施す

325. ptpolygon(n,r|org=p,scale=t,arg=θ,deg=θ)

:: 半径 r の円に内接する正 n 多角形の頂点の平面座標

326. ptlattice(m,n,v_1,v_2|org=p,scale=t,cond=[f_1,f_2,...],line=1)
:: org を始点として v_1 方向に m 個まで，v_2 方向に n 個までの合計 $m \times n$ 個の格子点の座標

327. ptcopy(ℓ,v)
:: 座標のリスト ℓ を移動方向のリスト v に従って複製コピーする

328. ptcommon([s_1,s_2],[t_1,t_2]|in=k)
:: 直線や線分や円に対して共通点，接点や垂線の足，内分点，方向転換進行点，角度などを求める

329. ptwindow(ℓ,[x_1,x_2],[y_1,y_2]|scale=t)
:: 平面座標 (x,y) のリストで $x_1 \le x \le x_2$，$y_1 \le y \le y_2$ を満たさないものを 0 に変える

330. ptbbox([[x_1,y_1,...],[x_2,y_2,...],...]|box=b)
ptbbox([[[$x^{(1)}_{min}$,$x^{(1)}_{max}$],[$y^{(1)}_{min}$,$y^{(1)}_{max}$],...],[[$x^{(2)}_{min}$,$x^{(2)}_{maxS}$],...],...]|box=1)
:: 座標 $(x,y,\ldots)$ や箱のリストを囲む箱 [[x_{min},x_{max}],[y_{min},y_{max}],...] を返す

331. iscombox([[$x^{(1)}_{min}$,$x^{(1)}_{max}$],[$y^{(1)}_{min}$,$y^{(1)}_{max}$],...], [[$x^{(2)}_{min}$,$x^{(2)}_{max}$],[$y^{(2)}_{min}$,$y^{(2)}_{max}$],...])
:: 2 つの箱（区間の直積）の共通部分の有無を返す

332. lninbox([p_1,p_2],[[x_{min},x_{max}],[y_{min},y_{max}]]|in=1)
:: 平面内の 2 点 p_1 と p_2 を結ぶ直線（または線分）の箱内の部分（2 点を結ぶ線分）を返す

333. scale(ℓ|scale=[c_1,...],f=f,shift=[s_1,s_2],TeX=1,mes=[t,[a_1,b_1,w_1],...],line=[a,b],
prec=1,col=s)
:: 目盛や対数（函数）尺の作成

334. tobezier(ℓ|inv=[a,b,t],div=k)
:: 点のリスト ℓ で定まる Bézier 曲線のパラメータ表示とその逆変換と分割

335. lbezier(l|inv=t)
:: 区分 Bézier 曲線のデータ変換

336. velbezier(f,[a,b,t])
:: Bézier 曲線の最大速度ベクトル

337. ptbezier(l,[n,t])　ptbezier(l,s)
::（複合）Bézier 曲線上の点と速度ベクトルを求める

338. areabezier(l|rev=1,pt=[p_1,p_2,...],para=1,prec=v,int=k,exp=c,Acc=1,cpx=1)
:: Bézier 曲線を用いた領域の面積・数値積分の計算

339. ptcombezier(ℓ_1,ℓ_2,m)
:: 点のリスト ℓ_1, ℓ_2 で定まる 2 つの Bézier 曲線の交点を求める

340. ptcombz(b_1,b_2,m|red=t,prec=k)
:: 区分 Bézier 曲線 b_1, b_2 の交点を求める

1.2.11　Drawing curves and graphs

341. xyproc(f|dviout=1,opt=s,env=t)
:: XY-pic/TikZ や指定した環境の開始（$f=1$）と終了（$f=0$）や表示

342. xypos([x,y,s])　xypos([x,y,s,t])　xypos([x,y,s,t,u])　xypos([x,y])
:: XY-pic/TikZ での座標 (x,y) や式 s などの文字出力．t はラベルの文字，u はオプション文字列．

343. xyput([x,y,s]|scale=r)　xyput([x,y,s,t])　xyput([x,y,s,t,u])　xyput([x,y])
:: XY-pic/TikZ での座標 (x,y) や式 s などの文字出力．t はラベルの文字，u はオプション文字列．

344. xyline([x_1,y_1,s_1],[x_2,y_2,s_2]|opt=t)　xyline([x_1,y_1],[x_1,y_2]|opt=t)
:: XY-pic/TikZ で (x_1,y_1) と (x_2,y_2) を線で結ぶ（引数はベクトルでもよい）

345. xyarrow([x_1,y_1,s_1],[x_2,y_2,s_2]|opt=t,cmd=s)
:: XY-pic/TikZ で (x_1,y_1) と (x_2,y_2) を矢印等で結ぶ（引数はベクトルでもよい）

346. xyarrows([f_1,f_2],[v_1,v_2],[[x_1,x_2,m],[y_1,y_2,n]]|scale=r,abs=v)
:: XY-pic/TikZ で複数の矢印を描く

347. xybox([[x_1,y_1],[x_2,y_2],[x_3,y_3]]|opt=t,color=s...)　xybox([[x_1,y_1],[x_2,y_2]]|opt=t,color=s,...)
:: XY-pic/TikZ で (x_1,y_1) と (x_2,y_2) を対角点とし，(x_3,y_3) を頂点とする平行四辺形（あるいは水平

な辺をもつ長方形）を描く

348. xycirc([x,y,s],r|opt=t,arg=[θ1,θ2],deg=[θ1,θ2],close=1)
:: XY-pic/TikZ で (x,y) 中心の半径 r mm/cm の円を描く

349. xybezier([[x1,y1],...,[xn,yn]]|verb=k,opt=t,cmd=s,relative=1)
:: XY-pic/TikZ で区分 Bézier 曲線（複合 Bézier 曲線）を描く

350. draw_bezier(id,idx,b|col=c,opt=s,init=1)
:: キャンバス上に区分 Bézier 曲線を描く

351. xylines([[x1,y1,s1],[x2,y2,s2],...]|opt=t,close=1,curve=1,ratio=c,verb=1,scale=r,Acc=1,dviout=1,proc=p)
:: XY-pic/TikZ で s_j を (x_j,x_j) に置き，(x_1,y_1), (x_2,y_2),... を（Bézier 曲）線で結ぶ

352. xyang(r,p0,p1,p2|opt=t,scale=r,prec=1,ar=1,dviout=1,proc=1)
:: XY-pic/TikZ で角 $\angle p_1p_0p_2$ の記号や円弧や扇形や矢印を描く

353. xyoval(p,r,q|opt=t,arg=[t1,t2,t3],deg=[t1,t2,t3],scale=r,ar=1,prec=1,dviout=1,proc=1)
:: XY-pic/TikZ で p を中心として，軸の長さが r と qr の楕円またはその弧や扇形を描く

354. xygrid ([x,tx,ux,mx,sx],[y,ty,uy,my,sy]|raw=r,shift=[sx,sy])
::（対数）方眼紙の描画

355. xygraph(f,n,[t1,t2],[x1,x2],[y1,y2]|opt=t,rev=1,ax=[x0,y0,s,t,u],axopt=[h,w,o,z],scale=r,ratio=c,raw=1,org=[x0,y0],pt=[p1,p2,...],verb=1,para=1,prec=v,shift=[u,v],Acc=1,dviout=t,proc=p)
:: 変域の n 等分点での値で関数のグラフを描く（$t_1 \le \mathtt{x} \le t_2$，$(x_1,y_1)$–$(x_2,y_2)$ は表示窓の範囲）

356. xy2graph(f,n,[x1,x2],[y1,y2],[h1,h2],α,β|opt=t,scale=r,view=h,raw=1,trans=1,ax=[z1,z2,t],dev=m,acc=k,org=[x0,y0,z0],pt=[p1,p2,...],prec=v,title=s,dviout=k,ext=[a,b],shift=[u,v],cl=1,proc=p)
:: x, y 変数の区間を n 等分して曲面 $z=f(x,y)$ の 3D グラフを描く

357. xy2curve([f1,f2,f3],n,[t1,t2],[y1,y2],[z1,z2],α,β|scale=r,gap=g,opt=s,eq=q,raw=w,dviout=d)
:: 空間曲線の隠線処理つき描画

358. execdraw(ℓ,t|shift=[u,v],ext=[a,b],cl=1)
:: **描画実行形式** ℓ を出力形態 t に従って実行する

1.2.12 Applications

359. powprimroot(p,n|all=1,exp=1,log=f)
:: p 以上の素数 n 個とその原始根のリストを作る

360. ntable(f,[a,b],n|dif=1,str=[k1,k2,...],mult=m,title=t,top=[t1,...])
:: 区間 $[a,b]$ を n 等分，または n=[n1,n2] 等分した点での x 変数の函数 f の函数値の数表を作る

361. distpoint(l|div=5,opt=s,title=t,size=ℓ)
:: 100 点満点の点数表のデータ l を元に点数分布などを表にする

362. seriesTaylor(f,k,v|evalopt=opt,small=1,frac=0,dviout=n)
:: 函数 f の変数または変数のリスト v に対する k 次の項までの Taylor 展開を求める

1.2.13 Environments

363. Canvas
:: Risa/Asir の描画キャンバスのデフォルトサイズ

364. AMSTeX
:: この値が 1 は $\mathcal{AMS}$-LaTeX を意味する

365. TeXEq
:: デフォルトの LaTeX の数式環境の指定（dviout0(3) で値が分かる）

366. TeXLim
:: LaTeX で長い数式を行分割する際の 1 行の許容最大横幅文字数のデフォルト値

367. TikZ
:: グラフ表示に XY-pic を使うか TikZ を使うかを指定
368. XYPrec
:: グラフ表示の時の座標の小数点以下の丸め桁数
369. XYcm
:: XY-pic での単位を cm で表して，TikZ に合わせる
370. XYLim
:: XY-pic や TikZ での曲線描画で順に指定する点での改行の間隔
371. DVIOUTH
:: myhelp() で指定した函数の解説を示すためのプログラムの指定
372. DIROUT
:: 数式を LaTeX に変換したソースが格納されるディレクトリ（書き込み可能なことが要請される）
373. DVIOUTA
:: LaTeX の $\mathcal{AMS}$TeX 環境で risaout.tex を変換して表示するプログラムのパス名
374. DVIOUTB
:: LaTeX の $\mathcal{AMS}$TeX 環境で risaout10.tex （または risaout10.tex）を変換して表示するプログラムのパス名で，DVIOUTA と交換できる（cf. dvioutO(), risatex.bat）.
375. DVIOUTL
:: LaTeX 環境で riasout0.tex を変換して表示するプログラムのパス名
376. .muldif
:: os_muldif.rr をロードしたときに読み込まれるファイル
377. risatex.bat
:: os_muldif.rr が出力する LaTeX のソースファイルを変換して表示するプログラム

1.3 Some functions in the original library

以下の函数は，module 化されないので，函数名の先頭に os_md. をつけません.

1.3.1 数の演算

1. idiv(i_1,i_2)
:: 整数除算による商
2. irem (i_1,i_2)
:: 整数除算による剰余
3. fac(i)
:: 自然数 i の階乗
4. nm(p)
:: 有理数または有理式の分子
5. dn(p)
:: 有理数または有理式の分母
6. igcd(i_1,i_2)
:: 整数 i_1 と i_2 の GCD を求める
7. igcdcntl([i])
:: 整数 GCD のアルゴリズムの選択
8. ilcm(i_1,i_2)
:: 整数の最小公倍数を求める
9. isqrt(n)
:: n の平方根を越えない最大の整数を返す
10. inv(i,m)
:: m を法とする i の逆数

11. random([*seed*])
    :: 乱数を生成する
12. lrandom(*bit*)
    :: 多倍長乱数を生成する
13. mt_save(*fname*)
    :: 乱数生成器の現在の状態をファイルにセーブする
14. mt_load(*fname*)
    :: ファイルにセーブされた乱数生成器の状態をロードする
15. real(*comp*)
    :: *comp* の実数部分
16. imag(*comp*)
    :: *comp* の虚数部分
17. conj(*comp*)
    :: *comp* の共役複素数
18. eval(*obj* [, *prec*])
    :: *obj* の値の評価
19. deval(*obj*)
    :: *obj* の値の評価（倍精度浮動小数）
20. pari(*func*,*arg*,*prec*)
    :: PARI の函数 *func* を呼び出す.
21. setprec([n])
    :: bigfloat の桁数を n 桁に設定する
22. setmode([p])
    :: 有限体を $GF(p)$ に設定する
23. ntoint32(n)
    :: 非負整数と符号なし 32bit 整数の間の型変換
24. int32ton(*int*32)
    :: 非負整数と符号なし 32bit 整数の間の型変換
25. iand(m,n)
    :: 整数 m, n のビット毎の and
26. ior(m,n)
    :: 整数 m, n のビット毎の or
27. ixor(m,n)
    :: 整数 m, n のビット毎の xor
28. ishift(i,*count*)
    :: 整数 i の絶対値を bit 列とみて shift する
29. i2hex(i|cap=1,num=1,min=m)
    :: 非負整数 i の 16 進表示
30. pari(binary,n)
    :: 数 n の 2 進数表示
31. pari(factor,n)
    :: 整数 n または 1 変数多項式 n の素因数分解を求める
32. pari(issquare,n)
    :: 整数 n または多項式 n が平方数（元）かどうか調べる
33. pari(omega,n)
    :: 整数 n または 1 変数多項式 n の素因子の個数
34. pari(bigomega,n)
    :: 整数 n または 1 変数多項式 n の素因子の重複度込みの個数
35. pari(numdiv,n)

:: 整数 n を割り切る正整数の個数

36. pari(sigma,n)
:: n の正の約数の和
37. pari(isprime,num)
:: 自然数 num が素数かどうか調べる（合成数なら 0 を返す）
38. pari(ispsp,num)
:: 自然数 num が擬素数かどうか調べる（合成数なら 0 を返す）
39. pari(nextprime,num)
:: 自然数 num 以上の最小の擬素数を返す
40. pari(frac,num)
:: 実数 num の小数部分
41. pari(conj,num)
:: 複素数 num（行列でも可）の共役複素数を返す
42. pari(abs,num[,$prec$])
:: 複素数 num の絶対値を返す
43. pari(cf,num)
:: num の連分数展開を返す
44. pari(mu,n)
:: 自然数 n のメビウス函数 $\mu(n)$ の値（0 または ± 1）を返す
45. pari(phi,n)
:: 自然数 n に対する Euler の φ 函数の値（n 以下の n と素なものの個数）を返す

1.3.2 多項式，有理式の演算

46. var(rat)
:: rat の主変数を返す
47. vars(obj)
:: obj に含まれる変数のリスト
48. uc()
:: 未定係数法のための vtype が 1 の不定元を生成する．
49. coef($poly$,deg [,var])
:: $poly$ の var (省略時は主変数) に関する deg 次の係数を出力する
50. deg($poly$,var)
:: $poly$ の, 変数 var に関する最高次数
51. mindeg($poly$,var)
:: $poly$ の, 変数 var に関する最低次数
52. nmono(rat)
:: rat の単項式の項数
53. ord([$varlist$])
:: 変数順序の設定
54. sdiv($poly1$,$poly2$[,v])
:: $poly1$ を $poly2$ で割る除算が最後まで実行できる場合に商を求める
55. sdivm($poly1$,$poly2$,mod[,v])
:: $GF(mod)$ 上で $poly1$ を $poly2$ で割る除算が最後まで実行できる場合に商を求める
56. srem($poly1$,$poly2$[,v])
:: $poly1$ を $poly2$ で割る除算が最後まで実行できる場合に剰余を求める
57. sremm($poly1$,$poly2$,mod[,v])
:: $GF(mod)$ 上で $poly1$ を $poly2$ で割る除算が最後まで実行できる場合に剰余を求める
58. sqr($poly1$,$poly2$[,v])
:: $poly1$ を $poly2$ で割る除算が最後まで実行できる場合に商, 剰余を求める

59. sqrm(*poly1*,*poly2*,*mod*[,*v*])
:: *GF*(*mod*) 上で *poly1* を *poly2* で割る除算が最後まで実行できる場合に商, 剰余を求める
60. tdiv([*poly1*,*poly2*])
:: *poly1* が *poly2* で割れたら商を割れなければ 0 を返す
61. subst(*rat*[,*varn*,*ratn*]*)
:: 有理式の特定の不定元に, 定数あるいは多項式, 有理式などを代入する
62. psubst(*rat*[,*varn*,*ratn*]*)
:: 有理式の特定の不定元に, 定数あるいは多項式, 有理式などを代入する
63. diff(*rat*[,*varn*]*)　または　diff(*rat*,*varlist*)
:: *rat* を *varn* あるいは *varlist* の中の変数で順次微分する
64. res(*var*,*poly1*,*poly2* [,*mod*])
:: 変数 *var* に関する多項式 *poly1* と *poly2* の終結式を求める
65. fctr(*poly*)
:: *poly* を既約因子に分解する
66. sqfr(*poly*)
:: *poly* を無平方分解する
67. ufctrhint(*poly*,*hint*)
:: 次数情報を用いた有理数係数の 1 変数多項式の因数分解
68. modfctr(*poly*,*mod*)
:: 有限体上での多項式の因数分解
69. ptozp(*poly*|factor=1)
:: 有理係数多項式を有理数倍して整数係数で係数の GCD が 1 の多項式に直す
70. %　*poly* % *m*
:: *poly* の各係数を整数 *m* で割った剰余で置き換えた多項式を返す
71. prim(*poly*[,*v*])
:: 有理係数多項式 *poly* の原始的部分 (primitive part)
72. cont(*poly*[,*v*])
:: 有理係数多項式 *poly* の容量 (content)
73. gcd(*poly1*,*poly2*[,*mod*])
:: 二つの多項式の最大公約式 (GCD) を求める
74. gcdz(*poly1*,*poly2*)
:: 有限体上の二つの多項式の最大公約式 (GCD) を求める
75. red(*rat*)
:: *rat* を約分する
76. umul(p_1,p_2)
:: 整数係数一変数多項式の高速乗算
77. umul_ff(p_1,p_2)
:: 有限体係数一変数多項式の高速乗算
78. usquare(p_1)
:: 整数係数一変数多項式の高速 2 乗算
79. usquare_ff(p_1)
:: 有限体係数一変数多項式の高速 2 乗算
80. utmul(p_1,p_2,*d*)
:: 整数係数一変数多項式の高速乗算 (打ち切り次数指定)
81. utmul_ff(p_1,p_2,*d*)
:: 有限体係数一変数多項式の高速乗算 (打ち切り次数指定)
82. kmul(p_1,p_2)
:: 一変数多項式の乗算を Karatsuba 法で行う
83. ksquare(p_1)

:: 一変数多項式の高速 2 乗算を Karatsuba 法で行う

84. ktmul(p_1,p_2,d)
一変数多項式の高速乗算 (打ち切り次数指定) を Karatsuba 法で行う
85. set_upkara([*threshold*])
:: 1 変数多項式の積演算における N^2 , Karatsuba, FFT アルゴリズムの切替えの閾値
86. set_uptkara([*threshold*])
:: 1 変数多項式の積演算における N^2 , Karatsuba, FFT アルゴリズムの切替えの閾値
87. set_upfft([*threshold*])
:: 1 変数多項式の積演算における N^2 , Karatsuba, FFT アルゴリズムの切替えの閾値
88. utrunc(p,d)
:: 一変数多項式を次数で切る
89. udecomp(p,d)
:: 一変数多項式を次数で分ける
90. ureverse(p)
:: 一変数多項式の次数を逆にした多項式を作る
91. uinv_as_power_series(p,d)
:: 一変数多項式を冪級数とみて, 逆元計算
92. ureverse_inv_as_power_series(p,d)
:: 一変数多項式を次数を逆にして冪級数とみて, 逆元計算
93. udiv(p_1,p_2)
:: 一変数多項式 p_1, p_2 に対し, 商を返す
94. urem(p_1,p_2)
:: 一変数多項式 p_1, p_2 に対し, 剰余を返す
95. urembymul(p_1,p_2)
:: 一変数多項式 p_1, p_2 に対し, 剰余を返す
96. urembymul_precomp(p_1,p_2,*inv*)
:: 固定された多項式による剰余計算を多数行う場合などに用いる
97. ugcd(p_1,p_2)
:: 一変数多項式 p_1, p_2 に対し, GCD を返す
98. pari(content,p)
:: x 変数の多項式とみて，係数の最大公約元を返す
99. pari(round,p)
:: 多項式の各係数を近似整数に置き換える
100. pari(roots,p[,*prec*])
:: 多項式の根を求める
101. pari(disc,p)
:: x 変数の多項式 p の判別式を返す.

1.3.3 リスト，ベクトル，配列の演算

102. car(*list*)
:: 空でない *list* の先頭要素（空の場合は空のリスト）を返す
103. cdr(*list*)
:: 空でない *list* から先頭要素を除いた（空の場合は空の）リストを返す
104. cons(*obj*,*list*)
:: *list* の先頭に *obj* を加えたリストを返す
105. append(*list1*,*list2*)
:: *list1* の後に *list2* をつなげたリストを返す
106. reverse(*list*)
:: 逆順に並べ替えたリストを返す

107. length(*lv*)
:: リストまたはベクトルの長さを返す
108. newvect(*len*[,*list*])
:: 長さ *len* のベクトルを生成する
109. vector(*len*[,*list*])
:: 長さ *len* のベクトルを生成する
110. vect([*elements*])
:: 要素の並びからベクトルを生成する
111. ltov(*list*)
:: *list* をベクトルに変換する
112. vtol(*vect*)
:: ベクトルをリストに変換する
113. newbytearray(*len*,[*listorstring*])
:: newvect と同様にして byte array を生成する
114. size(*vect/mat*)
:: 行列のサイズまたはベクトルの長さを求める函数（戻り値はリスト）
115. qsort(*array* [,*func*])
:: 一次元配列 array をソートする.

1.3.4 行列の演算

116. newmat(*row*,*col* [,[[*a*,*b*,...],[*c*,*d*,...],...]])
117. matrix(*row*,*col* [,[[*a*,*b*,...],[*c*,*d*,...],...]])
:: *row* 行 *col* 列の行列を生成する
118. mat(*vector*[,...])
119. matr(*vector*[,...])
:: 行ベクトル（またはリスト）の並びから行列を生成する
120. matc(*vector*[,...])
:: 列ベクトル（またはリスト）の並びから行列を生成する
121. det(*mat*[,*mod*])
:: *mat* の行列式
122. nd_det(*mat*[,*mod*])
:: 有理数または有限体上の多項式行列 *mat* の行列式
123. invmat(*mat*)
:: 行列 *mat* の逆行列
124. rowx(*matrix*,*i*,*j*)
:: 第 *i* 行と第 *j* 行を交換する
125. rowm(*matrix*,*i*,*c*)
:: 第 *i* 行を *c* 倍する
126. rowa(*matrix*,*i*,*j*,*c*)
:: 第 *i* 行に第 *j* 行の *c* 倍を加える
127. colx(*matrix*,*i*,*j*)
:: 第 *i* 列と第 *j* 列を交換する
128. colm(*matrix*,*i*,*c*)
:: 第 *i* 列を *c* 倍する.
129. cola(*matrix*,*i*,*j*,*c*)
:: 第 *i* 列に第 *j* 列の *c* 倍を加える.
130. pari(adj,*mat*)
:: 行列 *mat* の余因子行列（元の行列との積が行列式となる）を返す
131. pari(trace,*mat*)

:: 行列 *mat* の trace を返す

132. pari(signat,*mat*)
:: 実対称行列の符号を返す

133. pari(indexrank,*mat*)
:: 行列 *mat* の階数に等しいサイズの正則小行列の一つの行と列の位置を返す

134. pari(supplement,*mat*)
:: 列ベクトルが一次独立な行列 *mat* の右に列を補って正則正方行列を作る

135. pari(hess,*mat*)
:: 正方行列をヘッセンベルグ行列に変換する

136. pari(eigen,*mat*[,*prec*])
:: 数を成分とする行列の固有ベクトルを返す

137. pari(jacobi,*mat*[,*prec*])
:: 実対称行列の固有値と固有ベクトルを返す

1.3.5 文字列に関する演算

138. rtostr(*obj*)
:: *obj* を文字列に変える

139. strtov(*str*)
:: *str* (文字列) を不定元に変える

140. eval_str(*str*)
:: *str* (文字列) を評価する.

141. strtoascii(*str*)
:: 文字列をアスキーコード（1 以上 255 以下の整数）のリストで表す

142. asciitostr(*list*)
:: アスキーコードの列を文字列に変換する

143. str_len(*str*)
:: 文字列の長さを返す

144. str_chr(*str*,*start*,*c*)
:: 文字が最初に現れる位置を返す

145. sub_str(*str*,*start*,*end*)
:: 部分文字列を返す

1.3.6 構造体に関する函数

146. newstruct(*name*)
:: 構造体名が name の構造体を生成する

147. arfreg(*name*,*add*,*sub*,*mul*,*div*,*pwr*,*chsgn*,*comp*)
:: 構造体に体する基本演算を登録する

148. str_type(*name*|*object*)
:: 構造体の識別番号を取得する

1.3.7 入出力

149. end

150. quit
:: 現在読み込み中のファイルを閉じる. トップレベルにおいてはセッションを終了することになる

151. load("*filename*")
:: *filename* を読み込む

152. which("*filename*")
:: 引数 filename に対し, load() が読み込むパス名を返す

153. output(["*filename*"])
:: 以降の出力先を *filename* または標準出力に切替える

154. bsave(*obj*,"*filename*")
:: *filename* に *obj* をバイナリ形式で書き込む
155. bload("*filename*")
:: *filename* から数式をバイナリ形式で読み込む
156. print(*obj*[,*nl*])
:: *obj* を評価して表示する
157. printf(*format*[,*args*])
158. fprintf(*fd*,*format*[,*args*])
159. sprintf(*format*[,*args*])
:: C に似たプリント関数（実験的仕様関数）
160. access(*file*)
:: *file* の存在をテストし，存在すれば 1，存在しなければ 0 を返す
161. remove_file(*file*)
:: file を消去する
162. open_file("*filename*" [,"*mode*"])
:: *filename* をオープンする
163. close_file(*num*)
:: 識別子 *num* のファイルをクローズする
164. get_line([*num*])
:: 識別子 *num* のファイルから 1 行読む
165. get_byte(*num*)
:: 識別子 *num* のファイルから 1 バイト読む
166. put_byte(*num*, *c*)
:: 識別子 *num* のファイルに 1 バイト *c* を書く
167. purge_stdin()
:: 標準入力のバッファをクリアする

1.3.8 型や函数，モジュールに関わる函数

168. type(*obj*)
:: *obj* **の型**を求める函数
169. ntype(*num*)
:: *num* **（数）の 型** (整数) を返す.
170. vtype(*var*)
:: *var* **（不定元）の型** (整数) を返す
171. call(*name*,*args*)
:: 函数 *name* を呼び出す
172. functor(*func*)
:: *func* の函数子を取り出す
173. args(*func*)
:: *func* の引数リストを取り出す
174. funargs(*func*)
:: cons(functor(*func*),args(*func*)) を返す
175. module_list()
:: 定義済みのモジュールのリストを得る
176. module_definedp(*name*)
:: モジュール *name* の存在をテストする
177. remove_module(*name*)
:: モジュール *name* を削除する

1.3.9 数値函数

178. dacos(*num*)
:: 函数値 Arccos(*num*) を倍精度浮動小数で求める
179. dasin(*num*)
:: 函数値 Arcsin(*num*) を倍精度浮動小数で求める
180. datan(*num*)
:: 函数値 Arctan(*num*) を倍精度浮動小数で求める
181. dcos(*num*)
:: 函数値 cos(*num*) を倍精度浮動小数で求める
182. dsin(*num*)
:: 函数値 sin(*num*) を倍精度浮動小数で求める
183. dtan(*num*)
:: 函数値 tan(*num*) を倍精度浮動小数で求める
184. dabs(*num*)
:: 絶対値 $|num|$ を倍精度浮動小数で求める
185. dexp(*num*)
:: 函数値 exp(*num*) を倍精度浮動小数で求める
186. dlog(*num*)
:: 対数値 log(*num*) を倍精度浮動小数で求める
187. dsqrt(*num*)
:: 平行根 $\sqrt{num}$ を倍精度浮動小数で求める
188. ceil(*num*)
:: *num* 以上の最小の整数を求める
189. dceil(*num*)
:: *num* 以上の最小の整数を求める（`ceil()` の別名）
190. floor(*num*)
:: *num* 以下の最大の整数を求める
191. dfloor(*num*)
:: *num* 以下の最大の整数を求める（`floor()` の別名）
192. rint(*num*)
:: *num* を整数にまるめる
193. drint(*num*)
:: *num* を整数にまるめる（`rint()` の別名）
194. pari(sqrt,*num*[,*prec*])
:: （複素）数 *num* の平方根を与える
195. pari(arg,*num*[,*prec*])
:: 複素数 *num* の偏角を与える（$-\pi <$ 戻り値 $\leq \pi$）
196. pari(gamma,*num*[,*prec*])
:: ガンマ関数（*num* は複素数でもよい）
197. pari(gamh,*num*[,*prec*])
:: $\Gamma(num + \frac{1}{2})$（*num* は複素数でもよい）
198. pari(lngamma,*num*[,*prec*])
:: $\log(\Gamma(x))$
199. pari(psi,*num*[,*prec*])
:: digamma 関数
200. pari(erfc,*num*[,*prec*])
:: 相補誤差関数 (complementary error function)
201. pari(dilog,*num*[,*prec*])

:: dilogarithm 関数

202. pari(eta,τ[,$prec$])
:: Dedekind の Eta 関数 $\eta(\tau)$ で τ は虚部が正の複素数

203. pari(wp,τ[,$prec$])
:: Weber 関数 $\mathfrak{f}(\tau)$ で τ は虚部が正の複素数

204. pari(wp2,τ[,$prec$])
:: Weber 関数 $\mathfrak{f}_2(\tau)$ で τ は虚部が正の複素数

205. pari(jell,τ[,$prec$])
:: Elliptic j-invariant $j(\tau)$ で τ は虚部が正の複素数

206. pari(zeta,s[,$prec$])
:: Riemann の ζ 関数

1.3.10 描画函数

207. ifplot(*func* [,*geometry*] [,*xrange*] [,*yrange*] [,*id*] [,*name*])
:: 2 変数関数の実数上での零点を表示する

208. conplot(*func* [,*geometry*] [,*xrange*] [,*yrange*] [,*zrange*] [,*id*] [,*name*])
:: 2 変数関数の実数上での等高線を表示する

209. plot(*func* [,*geometry*] [,*xrange*] [,*id*] [,*name*])
:: 1 変数関数のグラフを表示する

210. polarplo(*func* [,*geometry*] [,*thetarange*] [,*id*] [,*name*])
:: 極形式 r =*func*(θ) で与えられた曲線を表示する

211. plotover(*func*,*id*,*number*)
:: すでに存在しているウィンドウへ描画する

212. open_canvas(*id*[,*geometry*])
:: 描画用ウィンドウ (キャンバス) を生成する

213. clear_canvas(*id*,*index*)
:: キャンバスをクリアする

214. draw_obj(*id*,*index*,*pointorsegment* [,*color*])
:: キャンバス上に点または線分を描画する

215. draw_string(*id*,*index*,[x,y],*string* [,*color*])
:: キャンバス上に文字列を描画する

1.3.11 有限体に関する演算

216. setmod_ff([p|*defpoly2*]) setmod_ff([*defpolyp*,p]) setmod_ff([p,n])
:: 有限体の設定, 設定されている有限体の法, 定義多項式の表示

217. field_type_ff([p|*defpoly2*])
:: 設定されている基礎体の種類

218. field_order_ff()
:: 設定されている基礎体の位数

219. characteristic_ff()
:: 設定されている体の標数

220. extdeg_ff()
:: 設定されている基礎体の, 素体に対する拡大次数

221. simp_ff(*obj*)
:: 数, あるいは多項式の係数を有限体の元に変換

222. random_ff()
:: 有限体の元の乱数生成

223. lmptop(*obj*)
:: $GF(p)$ 係数多項式の係数を整数に変換

224. ntogf2n(m)

:: 自然数を $GF(2^n)$ の元に変換

225. gf2nton(m)
:: $GF(2^n)$ の元を自然数に変換

226. ptogf2n($poly$)
:: 一変数多項式を $GF(2^n)$ の元に変換

227. gf2ntop(m,[v])
:: $GF(2^n)$ の元を多項式に変換

228. ptosfp(p)
:: 小標数有限体への変換

229. sfptop(p)
:: 小標数有限体からの変換

230. defpoly_mod2(d)
:: $GF(2)$ 上既約な d 次の一変数多項式の生成

231. sffctr($poly$)
:: 多項式の小標数有限体上での既約分解

232. fctr_ff($poly$)
:: 1 変数多項式の有限体上での既約分解

233. irredcheck_ff($poly$)
:: 1 変数多項式の有限体上での既約判定

234. randpoly_ff(d,v)
:: 有限体上の乱数係数 1 変数多項式の生成

235. ecm_add_ff(p_1,p_2,ec)
:: 楕円曲線上の点の加算

236. ecm_sub_ff(p_1,p_2,ec)
:: 楕円曲線上の点の減算

237. ecm_chsgn_ff(p_1)
:: 楕円曲線上の点の逆元

1.3.12 その他の函数

238. ctrl("*switch*" [,*obj*])
:: "*switch*" で指定した環境の設定，設定値を返す.

239. debug
:: デバッグモードに入る

240. error(*message*)
:: エラーメッセージを表示してデバッグモードに入る

241. time()
:: セッション開始から現在までの CPU 時間および GC 時間を表示する

242. cputime(*onoff*)
:: 引数が 0 ならば cputime の表示を止める，それ以外ならば表示を行う

243. tstart()
:: CPU time 計測開始

244. tstop()
:: CPU time 計測終了および表示

245. timer(*interval*,*expr*,*val*)
:: 制限時間つきで計算を実行する

246. currenttime()
1970 年 1 月 1 日 0 時 0 分 0 秒からの経過秒数

247. sleep(*interval*)
:: プロセスの実行を $interval \times 10^{-3}$ 秒停止

248. heap()
:: 現在のヒープの大きさを返す (単位:バイト)
249. version()
:: Asir のバージョン（整数）を返す.
250. shell(*command*)
:: *command* をシェルコマンドとして実行する
251. map(*function*,*arg0*,*arg1*,...)
:: リスト, 配列の各要素に函数を適用する
252. flist()
:: 現在定義されている組み込み函数, ユーザ定義函数の函数名を文字列リストとして返す
253. delete_history([*index*])
:: ヒストリを消去する
254. get_rootdir()
:: Asir のルートディレクトリ名を取り出す
255. getopt([*key*])
:: オプションの値を返す
256. getenv(*name*)
:: 環境変数 *name* の値を返す

2 Risa/Asir

Risa/Asir はオープンソースの計算機代数 (数式処理) システムです.
神戸版は OpenXM コミッターによって開発されています. オリジナルの Risa/Asir は富士通研究所で開発されました.
現在の神戸版は Windows 用（32bit, 64bit), UNIX 用, MAC OS X 用が存在し
http://www.math.kobe-u.ac.jp/Asir/asir-ja.html
からダウンロードできます.
Risa/Asir の著作権, ライセンス同意事項についてはこちらを御覧ください. 要約すると「非商用の場合, 配布, 改変は自由」という形で提供されています.

以下の §2.1 – §2.12 および 3. Some function in the original library は, Risa/Asir の Help ファイル（chm 形式）の一部を, 若干の改変をほどこして取り込んだもので, myhelp() によって os_muldif.rr に含まれる函数とともに Risa/Asir から参照することが出来ます.

2.1 Risa および Asir

Risa の構成は次の通りである.

- 基本演算部
  これは, Risa の内部形式に変換されたオブジェクト (数, 多項式など) の間の演算を実行する部分であり, UNIX の 'libc.a' などと同様の, ライブラリとして存在する. エンジンは, C および アセンブラで記述され, 後述する言語インタフェース Asir の基本演算部として用いられている.
- メモリ管理部
  Risa では, メモリ管理部として, [Boehm-Weiser] によるフリーソフトウェア (gc-6.1alpha5) を用いている. これはガーベジコレクション (以下 GC と呼ぶ) を自動的に行うメモリ割り当て機構を持ち, Risa の各部分はすべてこれにより必要なメモリを得ている.
- Asir
  Asir は, Risa の計算エンジンの言語インタフェースである. Risa では, 比較的容易にユーザ用の言語インタフェースを作ることができる. Asir はその一つの例として作ったもので, C 言語に近い文法をもつ. また, C のデバッガとして広く用いられている dbx 風のデバッガも備えている.

2.2 Asir の特徴

Asir は, 前述の通り, 計算エンジンの言語インタフェースである. 通常 Asir という名前の実行可能ファイルとして提供される. 現在サポートされている機能は概ね次の通りである.

- C 言語風のユーザ言語
- 数, 多項式, 有理式の加減乗 (除)
- ベクトル, 行列の演算
- 最小限のリスト処理
- 組み込み函数 (因数分解, GCD, グレブナ基底など)
- ユーザ定義函数によるツール (代数体上の因数分解など)
- dbx 風のデバッガ
- 陰函数の描画
- PARI (see section pari()) による初等超越函数を含む式の評価
- UNIX 上での分散計算機能 (Open XM)

2.3 コマンドラインオプション

コマンドラインオプションは次の通り.

-heap *number* Risa/Asir では, 4KB のブロックをメモリ割り当ての単位として用いている. デフォルトでは, 初期 heap として, 16 ブロック (64KB) 割り当てるが, それを変更する場合, `-heap` を用いる. 単位はブロックである. heap の大きさは, `heap()` 函数で調べることができる (単位はバイト).

-adj *number* この値が大きいほど, 使用メモリ量は大きくなるが, GC 時間が少なくなる. *number* として 1 以上の整数が指定できる. デフォルトでは 3 である. この値が 1 以下になると GC をしない設定になるので要注意である. heap をなるべく伸ばさずに, GC を主体にしてメモリ管理したい場合には, この値を大きく (例えば 8) 設定する.

-norc 初期化ファイル `'$HOME/.asirrc'` を読まない.

-quiet 起動時の著作権表示を行わない.

-f *file* 標準入力の代わりに, *file* から入力を読み込んで実行する. エラーの際にはただちに終了する.

-paristack number `PARI` (see section `pari()`) 専用の領域の大きさを指定する. 単位はバイト. デフォルトでは 1 MB.

-maxheap *number* heap 領域の上限を指定する. 単位はバイト. デフォルトでは無制限. UNIX の場合, 実際には limit コマンドで表示される datasize の値に制限されているため, `-maxheap` の指定がなくても一定量以上に heap を獲得できない場合があるので注意.)

2.4 環境変数

`Asir` の実行に関するいくつかの環境変数が存在する. UNIX 上では環境変数は shell のコマンドラインから直接設定するか, shell の `rc` ファイルで設定する. Windows NT では, [設定]→[システム]→[環境] で設定する. Windows 95/98 では, `'c:\autoexec.bat'` に書いて reboot する.

ASIR_LIBDIR `Asir` のライブラリディレクトリ, すなわちユーザ言語で書かれたファイルなどがおかれるディレクトリ. 指定がない場合 UNIX 版では `'/usr/local/lib/asir'`, Windows 版では `Asir` メインディレクトリの下の `'lib'` ディレクトリが用いられる. この環境変数は `ASIRLOADPATH` に統合され廃止される予定.

ASIR_CONTRIB_DIR `Asir` の asir-contrib ディレクトリ, すなわち `OpenXM/asir-contrib` プロジェクトで書かれたパッケージやデータなどがおかれるディレクトリ. 指定がない場合 UNIX 版では `'/usr/local/lib/asir-contrib'`, Windows 版では `Asir` メインディレクトリの下の `'lib-asir-contrib'` ディレクトリが用いられる. この環境変数は `ASIRLOADPATH` に統合され廃止される予定.

ASIRLOADPATH ロードされるファイルがあるディレクトリを UNIX の場合 `':'`, Windows の場合 `';'` で区切って並べる. ディレクトリは左から順にサーチされる. この指定がない場合, および指定されたファイルが `ASIRLOADPATH` になかった場合, ライブラリディレクトリもサーチされる.

HOME `-norc` オプションつきで起動しない場合, `'$HOME/.asirrc'` があれば, 予めこのファイルを実行する. `HOME` が設定されていない場合, UNIX 版ではなにも読まないが, Windows 版では `Asir` メインディレクトリ (`get_rootdir()` で返されるディレクトリ) の `'.asirrc'` を探し, あればそれを実行する.

2.5 起動から終了まで

`Asir` を起動すると,

```
[0]
```

なるプロンプトが表示され, セッションが開始する. ‘$HOME/.asirrc’ (Windows 版の場合, HOME 設定されていない場合には get_rootdir() で返されるディレクトリにある‘.asirrc’) が存在している場合, このファイルを Asir ユーザ言語でかかれたファイルと見なし, 解釈実行する.

プロンプトは入力の番号を表す. セッションは, end; または quit; を入力することにより終了する. 入力は, ‘;’ または ‘$’ までを一区切りとして評価される. ‘;’ のとき結果は表示され, ‘$’ のとき表示されない.

```
% asir
[0] A;
0
[1] A=(x+y)^5;
x^5+5*y*x^4+10*y^2*x^3+10*y^3*x^2+5*y^4*x+y^5
[2] A;
x^5+5*y*x^4+10*y^2*x^3+10*y^3*x^2+5*y^4*x+y^5
[3] a=(x+y)^5;
evalpv : invalid assignment
return to toplevel
[3] a;
a
[4] fctr(A);
[[1,1],[x+y,5]]
[5] quit;
%
```

この例では, A, a, x, y なる文字が使用されている. A はプログラムにおける変数で, a, x, y は数学的な意味での不定元である. 一般にプログラム変数は大文字で始まり, 不定元は小文字で始まる. この例でわかるように, プログラム変数は, 数, 式などを格納しておくためのものであり, C 言語などにおける変数に対応する. 一方, 不定元はそれ自身で値を持つことはできず, 従って, 不定元に対する代入は許されない. 後に示すが, 不定元に対する代入は, 組み込み函数 subst() により明示的に行われる.

2.6 割り込み

計算を実行中に割り込みをかけたい場合, 割り込みキャラクタ (通常は C-c, Windows, DOS 版では C-x を入力する.

```
@ (x+y)^1000;
C-cinterrupt ?(q/t/c/d/u/w/?)
```

各選択肢の意味は次の通り.

- q　Asir を終了する. (確認あり)
- t　トップレベルに戻る. (確認あり)
- c　実行を継続する.
- d　デバッグモードに入る. デバッガに関しては See section デバッガ.
- u　register_handler() (see section ox_reset, ox_intr, register_handler) で登録された関数を実行後トップレベルに戻る. (確認あり)
- w　中断点までの函数の呼び出し列を表示する.
- ?　各選択肢の意味を説明する

2.7 エラー処理

組み込み函数に不正な型の引数を渡した場合などには実行が中断されるが, ユーザ函数の中でエラーが起きた場合にはトップレベルに戻る前に自動的にデバッグモードに入る. この状態でエラーの場所, 直前の引数の値などを調べることができる. 表示されるエラーメッセージはさまざまであり, 内部の函数名に引き続いてメッセージが表示される. これは, 呼び出された組み込み函数と必ずしも対応はしない.

その他, さまざまな原因により内部演算函数においてエラーが生ずることがある. UNIX 版の場合, これは次のいずれかの internal error として報告され, 通常のエラーと同様に扱って, デバッグモードに入る.

SEGV

BUS ERROR　組み込み函数によっては, 引数の型を厳密にチェックせずに演算ルーチンに引き渡してしまうものも存在している. このような状況において, 不正なポインタ, あるいは NULL ポインタによるアクセス違反があった場合, これらのエラーとなる.

BROKEN PIPE　プロセス間通信において, 相手先のプロセスとの間のストリームが既に存在していない場合 (例えば既に相手先のプロセスが終了している場合など) に, そのストリームに入出力しようとした場合にこのエラーとなる. これらは実際には, 組み込み函数の入口において, 引数を完全にチェックすることにより大部分は防げるが, 手間が多くかかることと, 場合によっては効率を落すことにもなるため, あえて引数チェックはユーザ任せにしてある.

2.8 計算結果, 特殊な数

@ はエスケープ文字として使用される. 現在次のような規定がある.

@n　n 番目の計算結果.
@@　直前の計算結果.
@i　虚数単位.
@pi　円周率.
@e　自然対数の底.
@　2 元体 GF(2) 上の一変数多項式の変数 (不定元)
@>, @<, @>=, @<=, @==, @&&, @||　quantifier elimination における, 一階述語論理演算子

```
[0] fctr(x^10-1);
[[1,1],[x-1,1],[x+1,1],[x^4+x^3+x^2+x+1,1],[x^4-x^3+x^2-x+1,1]]
[1] @@[3];
[x^4+x^3+x^2+x+1,1]
[2] eval(sin(@pi/2));
1.0000000000000000000000000000000000000000000000000000000000000000
[3] eval(log(@e),20);
0.99999999999999999999999999998
[4] @0[4][0];
x^4-x^3+x^2-x+1
[5] (1+@i)^5;
(-4-4*@i)
[6] eval(exp(@pi*@i));
-1.0000000000000000000000000000
[7] (@+1)^9;
(@^9+@^8+@+1)
```

トップレベルで計算された値はこのようにヒストリとして取り出し可能であるが, このことは, ガベージコレクタにとっては負担をもたらす可能性がある. 特に, 大きな式をトップレベルで計算した場合, その後の GC 時間が急速に増大する可能性がある. このような場合, delete_history() が有効である.

2.9 型

2.9.1 Asir で使用可能な型

Asir においては, 可読な形式で入力されたさまざまな対象は, パーザにより中間言語に変換され, インタプリタにより Risa の計算エンジンを呼び出しながら内部形式に変換される. 変換された対象は, 次のいずれかの型を持つ. 各番号は, 組み込み函数 type() により返される値に対応している. 各例は, Asir のプロンプトに対する入力が可能な形式のいくつかを示す.

0 0 実際には 0 を識別子にもつ対象は存在しない. 0 は, C における 0 ポインタにより表現されている. しかし, 便宜上 Asir の type(0) は値 0 を返す.

1 数 1 0x0d 2/3 14.5 3+2*@i
数は（0x0d は 16 進数, @i は虚数単位）, さらにいくつかの型に分けられる. これについては下で述べる.

2 多項式 (数でない) x afo (2.3*x+y)^10
多項式は, 全て展開され, その時点における変数順序に従って, 再帰的に 1 変数多項式として降冪の順に整理される. (See section 分散表現多項式). この時, その多項式に現れる順序最大の変数を 主変数と呼ぶ.

3 有理式 (多項式でない) (x+1)/(y^2-y-x) x/x
有理式は, 分母分子が約分可能でも, 明示的に red() が呼ばれない限り約分は行われない. これは, 多項式の GCD 演算が極めて重い演算であるためで, 有理式の演算は注意が必要である.

4 リスト [] [1,2,[3,4],[x,y]]
リストは読み出し専用である. [] は空リストを意味する. リストに対する操作としては, car(), cdr(), cons() などによる操作の他に, 読み出し専用の配列とみなして, [*index*] を必要なだけつけることにより要素の取り出しを行うことができる. 例えば

```
[0] L = [[1,2,3],[4,[5,6]],7]$
[1] L[1][1];
[5,6]
```

注意すべきことは, リスト, 配列 (行列, ベクトル) 共に, インデックスは 0 から始まることと, リストの要素の取り出しをインデックスで行うことは, 結局は先頭からポインタをたどることに相当するため, 配列に対する操作に比較して大きなリストでは時間がかかる場合があるということである.

5 ベクトル newvect(3) newvect(2,[a,1])
ベクトルは, newvect() で明示的に生成する必要がある. 前者の例では 2 成分の 0 ベクトルが生成され, 後者では, 第 0 成分が a, 第 1 成分が 1 のベクトルが生成される. 初期化のための 第 2 引数は, 第 1 引数以下の長さのリストを受け付ける. リストの要素は左から用いられ, 足りない分は 0 が補われる. 成分は [*index*] により取り出せる. 実際には, 各成分に, ベクトル, 行列, リストを含む任意の型の対象を代入できるので, 多次元配列をベクトルで表現することができる.

```
[0] A3 = newvect(3);
[ 0 0 0 ]
[1] for (I=0;I<3;I++)A3[I] = newvect(3);
[2] for (I=0;I<3;I++)for(J=0;J<3;J++)A3[I][J]=newvect(3);
[3] A3;
[ [ [ 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 ] ] ]
[4] A3[0];
[ [ 0 0 0 ] [ 0 0 0 ] [ 0 0 0 ] ]
[5] A3[0][0];
[ 0 0 0 ]
```

6 **行列** `newmat(2,2)` `newmat(2,3,[[x,y],[z]])`

行列の生成も `newmat()` により明示的に行われる. 初期化も, 引数がリストのリストとなることを除いてはベクトルと同様で, リストの各要素 (これはまたリストである) は, 各行の初期化に使われ, 足りない部分には 0 が埋められる. 行列も, 各要素には任意の対象を代入できる. 行列の各行は, ベクトルとして取り出すことができる.

```
[0] M=newmat(2,3);
[ 0 0 0 ]
[ 0 0 0 ]
[1] M[1];
[ 0 0 0 ]
[2] type(@@);
5
```

7 **文字列** `""` `"afo"`

文字列は, 主にファイル名などに用いられる. 文字列に対しては加算のみが定義されていて, 結果は 2 つの文字列の結合である.

```
[0] "afo"+"take";
afotake
```

8 **構造体** `newstruct(afo)`

`Asir` における構造体は, `C` における構造体を簡易化したものである. 固定長配列の各成分を名前でアクセスできるオブジェクトで, 構造体定義毎に名前をつける.

9 **分散表現多項式** `2*<<0,1,2,3>>-3*<<1,2,3,4>>`

これは, ほとんどグレブナ基底専用の型で, 通常の計算でこの型が必要となることはまずないが, グレブナ基底計算パッケージ自体がユーザ言語で書かれているため, ユーザが操作できるよう独立した型として `Asir` で使用できるようにしてある. これについては See section グレブナ基底の計算.

10 **符号なしマシン 32bit 整数**

11 **エラーオブジェクト**

以上二つは, Open XM において用いられる特殊オブジェクトである.

12 GF(2) **上の行列**

現在, 標数 2 の有限体における基底変換のためのオブジェクトとして用いられる.

13 MATHCAP **オブジェクト**

Open XM において, 実装されている機能を送受信するためのオブジェクトである.

14 first order formula

quantifier elimination で用いられる一階述語論理式.

15 matrix over $GF(p)$

小標数有限体上の行列.

16 byte array

符号なし byte の配列

-1 VOID **オブジェクト**

型識別子 -1 をもつオブジェクトは関数の戻り値などが無効であることを示す.

2.9.2 数の型

0 有理数

有理数は, 任意多倍長整数 (bignum) により実現されている. 有理数は常に既約分数で表現される.

1 倍精度浮動小数

マシンの提供する倍精度浮動小数である. `Asir` の起動時には, 通常の形式で入力された浮動小数はこの型に変換される. ただし, `ctrl()` により `bigfloat` が選択されている場合には bigfloat に変換される.

```
[0] 1.2;
1.2
[1] 1.2e-1000;
0
[2] ctrl("bigfloat",1);
1
[3] 1.2e-1000;
1.20000000000000000513 E-1000
```

倍精度浮動小数と有理数の演算は, 有理数が浮動小数に変換されて, 浮動小数として演算される.

2 代数的数 See section 代数的数に関する演算.

3 bigfloat

bigfloat は, `Asir` では `PARI` ライブラリにより実現されている. `PARI` においては, bigfloat は, 仮数部のみ任意多倍長で, 指数部は 1 ワード以内の整数に限られている. `ctrl()` で bigfloat を選択することにより, 以後の浮動小数の入力は bigfloat として扱われる. 精度はデフォルトでは 10 進 9 桁程度であるが, `setprec()` により指定可能である.

```
[0] ctrl("bigfloat",1);
1
[1] eval(2^(1/2));
1.41421356237309504876378807303 1
[2] setprec(100);
9
[3] eval(2^(1/2));
1.41421356237309504880168872420969807856967187537694807317...
```

`eval()` は, 引数に含まれる函数値を可能な限り数値化する函数である. `setprec()` で指定された桁数は, 結果の精度を保証するものではなく, `PARI` 内部で用いられる表現のサイズを示すことに注意すべきである. (See section `eval()`, `deval()`, `pari()`.)

4 複素数

複素数は, 有理数, 倍精度浮動小数, bigfloat を実部, 虚部として `a+b*@i` (`@i` は虚数単位) として与えられる数である. 実部, 虚部はそれぞれ `real()`, `imag()` で取り出せる.

5 小標数の有限素体の元

ここで言う小標数とは, 標数が 2^{27} 未満のもののことである. このような有限体は, 現在のところグレブナ基底計算において内部的に用いられ, 有限体係数の分散表現多項式の係数を取り出すことで得られる. それ自身は属する有限体に関する情報は持たず, `setmode()` で設定されている素数 p を用いて $GF(p)$ 上での演算が適用される.

6 大標数の有限素体の元

標数として任意の素数がとれる. この型の数は, 整数に対し `simp_ff` を適用することにより得られる.

7 標数 2 の有限体の元

標数 2 の任意の有限体の元を表現する. 標数 2 の有限体 F は, 拡大次数 [F:$GF(2)$] を n とすれば, $GF(2)$ 上既約な n 次多項式 $f(t)$ により $F = GF(2)[t]/(f(t))$ とあらわされる. さらに, $GF(2)[t]$ の元 g は, $f(t)$ も含めて自然な仕方でビット列とみなされるため, 形式上は, F の元 $g \mod f$ は, g, f をあらわす 2 つのビット列で表現することができる. F の元を入力するいくつかの方法が用意されている.

`@` `@` はその後ろに数字, 文字を伴って, ヒストリや特殊な数をあらわすが, 単独で現れた場合には, $F = GF(2)[t]/(f(t))$ における $t \mod f$ をあらわす. よって, `@` の多項式として F の元を入力できる. (`@^10+@+1` など)

ptogf2n 任意変数の 1 変数多項式を, ptogf2n() により対応する F の元に変換する.

ntogf2n 任意の自然数を, 自然な仕方で F の元とみなす. 自然数としては, 10 進, 16 進 (`0x` で始まる), 2 進 (`0b` で始まる) で入力が可能である.

その他 多項式の係数を丸ごと F の元に変換するような場合, simp_ff() により変換できる.

8 位数 p^n の有限体の元

位数が p^n (p は任意の素数, n は正整数) は, 標数 p および $GF(p)$ 上既約な n 次多項式 $m(x)$ を `setmod_ff` により指定することにより設定する. この体の元は $m(x)$ を法とする $GF(p)$ 上の多項式として表現される.

9 位数 p^n の有限体の元

(小位数) 位数が p^n の有限体 (p^n が 2^{29} 以下, p が 2^{14} 以上なら n は 1) は, 標数 p および拡大次数 n を setmod_ff() により指定することにより設定する. この体の 0 でない元は, p が 2^{14} 未満の場合, $GF(p^n)$ の乗法群の生成元を固定することにより, この元のべきとして表される. これにより, この体の 0 でない元は, このべき指数として表現される. p が 2^{14} 以上の場合は通常の剰余による表現となるが, 共通のプログラムで双方の場合を扱えるようにこのような仕様となっている.

10 位数 p^n の小位数有限体の代数拡大の元

前項の, 位数が p^n の小位数有限体の m 次拡大の元である. 標数 p および拡大次数 n, m を setmod_ff() により指定することにより設定する. 基礎体上の m 次既約多項式が自動生成され, その代数拡大の生成元の定義多項式として用いられる. 生成元は `@s` である.

11 分散表現の代数的数

See section 代数的数に関する演算. 小標数有限素体以外の有限体は setmod_ff() で設定する. 有限体の元どうしの演算では, 一方が有理数の場合には, その有理数は自動的に現在設定されている有限体の元に変換され, 演算が行われる.

2.9.3 不定元の型

多項式の変数となり得る対象を不定元とよぶ. `Asir` では, 英小文字で始まり, 任意個のアルファベット, 数字, '_' からなる文字列を不定元として扱うが, その他にもシステムにより不定元として扱われるものがいくつかある. `Asir` の内部形式としては, これらは全て多項式としての型を持つが, 数と同様, 不定元の型により区別される.

0 一般不定元

英小文字で始まる文字列. 多項式の変数として最も普通に用いられる.

```
[0] [vtype(a),vtype(aA_12)];
[0,0]
```

1 未定係数 uc() は, '_' で始まる文字列を名前とする不定元を生成する. これらは, ユーザが入力できないというだけで, 一般不定元と変わらないが, ユーザが入力した不定元と衝突しないという性質を利用して未定係数の自動生成などに用いることができる.

```
 [1] U=uc();
_0
```

```
[2] vtype(U);
1
```

2 函数形式

組み込み函数, ユーザ函数の呼び出しは, 評価されて何らかの Asir の内部形式に変換されるが, sin(x), cos(x+1) などは, 評価後もそのままの形で存在する. これは函数形式と呼ばれ, それ自身が 1 つの不定元として扱われる. またやや特殊な例として, 円周率 @pi や自然対数の底 @e も函数形式として扱われる.

```
[3] V=sin(x);
sin(x)
[4] vtype(V);
2
[5] vars(V^2+V+1);
[sin(x)]
```

3 函数子

函数呼び出しは, fname(*args*) という形で行なわれるが, fname の部分を函数子と呼ぶ. 函数子には, 函数の種類により組み込み函数子, ユーザ定義函数子, 初等函数子などがあるが, 函数子は単独で不定元として機能する.

```
[6] vtype(sin);
3
```

2.10 ユーザ言語 Asir

Asir の組み込み函数は, 因数分解, GCD などの計算を行うもの, ファイル入出力を行うもの, あるいは数式の一部を取り出すものなどさまざまなものが用意されているが, ユーザが実際に行いたいことを実行させるためには一般にはユーザ言語によるプログラムを書く必要がある. ユーザ言語も Asir と呼ばれる. 以下では, ユーザ言語の文法規則および実際のユーザ言語プログラムを例としたプログラムの書き方について述べる.

2.10.1 文法 (C 言語との違い)

Asir の文法は C 言語に準拠している. おもな相違点は次の通りである. 以下で, 変数とは Asir におけるプログラム用の変数, すなわち大文字で始まる文字列を意味することとする.

- 変数の型がない.
  既に説明したとおり, Asir で扱われる対象自身は全て何らかの型を持っている. しかし, プログラム変数自体は, どのような対象でも代入できるという意味で型がないのである.

```
[0] A = 1;
1
[1] type(A);
1
[2] A = [1,2,3];
[1,2,3]
[3] type(A);
4
```

- 函数内の変数は, デフォルトでは仮引数をこめてすべて局所変数. ただし, extern 宣言された変数は, トップレベルにおける大域変数となる. すなわち, 変数のスコープは大域変数と局所変数の 2 種類に単

純化されている．トップレベル, すなわちプロンプトに対して入力された変数は全て大域変数として登録される．また函数内では次のいずれかとなる．函数が定義されるファイルにおいて, その函数定義以前に, ある変数が extern 宣言されている場合, 函数内のその変数も大域変数として扱われる．extern 宣言されていない変数はその函数に局所的となる．

```
% cat afo
def afo() { return A;}
extern A$
def bfo() { return A;}
end$
% asir
[0] load("afo")$
[5] A = 1;
1
[6] afo();
0
[7] bfo();
1
```

- プログラム変数は大文字で始まり, 不定元, 函数は小文字で始まる.
  この点は, 既存の数式処理システムのほとんどと異なる点である．Asir がこの仕様を採用したのは, ユーザが不定元のつもりで使用した変数になんらかの値が代入されていた場合に混乱を招く, という, 既存のシステムにありがちな状況を避けるためである.
- switch 文, goto がない.
  goto がないため, 多重ループを一度に抜けるのがやや複雑になる場合がある.
- コンマ式は, for(A;B;C) または, while(A) の A, B, C にのみ使うことができる.
  これは, リストを正式なオブジェクトとして加えたことによる.

以上は制限であるが, 拡張としては次の点が挙げられる.

- 有理式に対する計算を, 通常の C における計算と同様にできる.
- リストが扱える．構造体を用いるまでもない要素の集合体を, リストで表すことができ, C で直接書く場合に比較してプログラムが短く, 読みやすく書ける.
- ユーザ定義函数における一行ヘルプ．Emacs-Lisp に類似した機能である．詳しくは, See section ユーザ定義函数 を見よ.
- ユーザ定義函数におけるオプション指定．これに関しては, See section オプション指定.

Asir では次の語句がキーワードとして定められている.

- C 言語に由来: break, continue, do, else, extern, for, if, return, static, struct, while
- C 言語からの拡張: def, endmodule, function, global, local, localf, module
- 函数: car, cdr, getopt, newstruct, map, pari, quote, recmap, timer

2.10.2 ユーザ定義函数

ユーザによる函数の定義は ‘def’ 文で行う．文法エラーは読み込み時にある程度チェックされ, おおよその場所が表示される．既に (引数の個数に関係なく) 同名の函数が定義されている場合には, その函数は再定義される．ctrl() 函数により verbose フラグが on になっている場合,

```
afo() redefined.
```

というメッセージが表示される. ある函数の定義において, まだ未定義の函数を呼び出していても, 定義時にはエラーにならない. 実行時に未定義の函数を呼び出そうとした場合にエラーとなる.

```
def  f(X) {
    if ( !X )
        return 1;
    else
        return X * f(X-1);
}

def c(N)
{
    A = newvect(N+1); A[0] = B = newvect(1); B[0] = 1;
    for ( K = 1; K <= N; K++ ) {
        A[K] = B = newvect(K+1); B[0] = B[K] = 1;
        for ( P = A[K-1], J = 1; J < K; J++ )
            B[J] = P[J-1]+P[J];
        }
    return A;
}

def add(A,B)
"add two numbers."
{
    return A+B;
}
```

2 つ目の例では, 長さ N + 1 のベクトル (A とする) が返される. A[I] は長さ I+1 の配列であり, そのそれぞれの要素がを要素とする配列である.

3 つ目の例では, 引数並びのあとに文字列が置かれているが、これは Emacs-Lisp の関数定義に類似の機能で、ヘルプ用の文字列である。この例の場合、help(add) によってこの文字列が出力される。

参照 section help.

以下では, C によるプログラミングの経験がない人のために, Asir 言語によるプログラムの書き方を解説する.

2.10.3 変数および不定元

既に述べた通り, Asir においてはプログラム変数と不定元を明確に区別している.

変数 大文字で始まり, アルファベット, 数字, '_' からなる文字列 変数あるいはプログラム変数とは, Asir のさまざまな型の内部形式を格納するための箱であり, 格納された内部形式が, この変数の値である. 変数が式の要素として評価される時は, そこに収められた値に置き換えられる. すなわち, 内部形式の中にはプログラム変数は現れない. 変数は全て 0 で初期化されている.

```
[0] X^2+X+1;
1
[1] X=2;
2
[2] X^2+X+1;
```

7

不定元 小文字で始まり, アルファベット, 数字, '_' からなる文字列, またはシングルクオートで囲まれた文字列, もしくは函数形式. 不定元とは, 多項式環を構成する際に添加される変数をいう. Asir においては, 不定元は値をもたない超越的な元であり, 不定元への値の代入は許されない.

```
[3] X=x;
x
[4] X^2+X+1;
x^2+x+1
[5] A='Dx'*(x-1)+x*y-y;
(y+Dx)*x-y-Dx
[6] function foo(x,y);
[7] B=foo(x,y)*x^2-1;
foo(x,y)*x^2-1
```

2.10.4 引数

```
def sum(N) {
    for ( I = 1, S = 0; I <= N; I++ )
        S += I;
    return S;
}
```

これは, 1 から N までの自然数の和を求める函数 sum() の定義である. この例における sum(N) の N が引数である. この例は, 1 引数函数の例であるが, 一般に引数の個数は任意であり, 必要なだけの個数を ',' で区切って指定することができる. 引数は値が渡される. すなわち, 引数を受けとった側が, その引数の値を変更しても, 渡した側の変数は変化しない. ただし, 例外がある. それは, ベクトル, 行列を引数に渡した場合である. この場合も, 渡された変数そのものを書き替えることは, その函数に局所的な操作であるが, 要素を書き換えた場合, それは, 呼び出し側のベクトル, 行列の要素を書き換えることになる.

```
def clear_vector(M) {
    /* M is expected to be a vector */
    L = size(M)[0];
    for ( I = 0; I < L; I++ )
        M[I] = 0;
}
```

この函数は, 引数のベクトルを 0 ベクトルに初期化するための函数である. また, ベクトルを引数に渡すことにより, 複数の結果を引数のベクトルに収納して返すことができる. 実際には, このような場合には, 結果をリストにして返すこともできる. 状況に応じて使いわけすることが望ましい.

2.10.5 コメント

C と同様 '/*' と '*/' で囲まれた部分はコメントとして扱われる.

```
/*
 * This is a comment.
 */
```

```
def afo(X) {
```

コメントは複数行に渡っても構わないが, 入れ子にすることはできない. ‘/*’ がいくつあっても最初のもののみが有効となり, 最初に現れた ‘*/’ でコメントは終了したと見なされる. プログラムなどで, コメントを含む可能性がある部分をコメントアウトした場合には, #if 0, #endif を使えばよい. (See section プリプロセッサ.)

```
#if 0
def bfo(X) {
/* empty */
}
#endif
```

2.10.6 文

Asir のユーザ函数は,

```
def 名前(引数,引数,...,引数) {
    文
    文
    ...
    文
}
```

という形で定義される. このように, 文は函数の基本的構成要素であり, プログラムを書くためには, 文がどのようなものであるか知らなければならない. 最も単純な文として, 単文がある. これは,

```
S = sum(N);
```

のように, 式に終端記号 (‘;’ または \tt ‘$’) をつけたものである. この単文及び類似の return 文, break 文などが文の最小構成単位となる. if 文や for 文の定義 (section 文法の詳細) を見ればわかる通り, それらの本体は, 単なる一つの文として定義されている. 通常は, 本体には複数の文が書けることが必要となる. このような場合, ‘{’ と ‘}’ で文の並びを括って, 一つの文として扱うことができる. これを複文と呼ぶ.

```
if ( I == 0 ) {
    J = 1;
    K = 2;
    L = 3;
}
```

‘}’ の後ろには終端記号は必要ない. なぜなら, ‘’ **文並び** ‘’ が既に文となっていて, if 文の要請を満たしているからである.

2.10.7 return 文

return 文は,

return 式;

return;

の 2 つの形式がある. いずれも函数から抜けるための文である. 前者は函数の値として 式 を返す. 後者では, 函数の値として何が返されるかはわからない.

2.10.8 if 文

if 文には

```
if ( 式 )                    if ( 式 )
    文          及び             文
else
    文
```

の 2 種類がある. これらの動作は明らかであるが, 文の位置に if 文が来た場合に注意を要する. 次の例を考えてみよう.

```
if ( 式 )
    if ( 式 ) 文
else
    文
```

この場合, 字下げからは, else 以下は, 最初の if に対応するように見えるが, パーザは, 自動的に 2 番目の if に対応すると判断する. すなわち, 2 種類の if 文を許したために, 文法に曖昧性が現れ, それを解消するために,else 以下は, 最も近い if に対応するという規則が適用されるのである. 従って, この例は,

```
if ( 式 ) {
    if ( 式 ) 文 else 文
}
```

という意味となる. 字下げに対応させるためには,

```
if ( 式 ) {
    if ( 式 ) 文
} else
    文
```

としなければならない.

関数の中でなく, top level で if 文を用いるときは $ または ; で終了する必要がある. これらがないと次の文がよみとばされる.

2.10.9 ループ

ループを構成する文は, while 文, for 文, do 文の 3 種類がある.

- while 文
  形式は,

  ```
  while ( 式 ) 文
  ```

  で, これは, 式 を評価して, その値が 0 でない限り 文 を実行するという意味となる. たとえば 式 が 1 ならば, 単純な無限ループとなる.
- for 文
  形式は,

  ```
  for ( 式並び-1; 式; 式並び-2 ) 文
  ```

  で, これは式並び −1 (を単文並びにしたもの)

```
while ( 式 ) {
    文
    式並び$-2$（を単文並びにしたもの）
}
```

と等価である.

- do 文

```
do {
    文
} while ( 式 )
```

は, 先に 文を実行してから条件式による判定を行う所が while 文と異なっている. ループを抜け出す手段として, break 文及び return 文がある. また, ループの制御をある位置に移す手段として continue 文がある.

- break
  break 文は, それを囲むループを一つだけ抜ける.
- return
  return 文は, 一般に函数から抜けるための文であり, ループの中からでも有効である.
- continue
  continue 文は, ループの本体の文の末端に制御を移す. 例えば for 文では, 最後の式並びの実行を行い, while 文では条件式の判定に移る.

2.10.10 構造体定義

構造体とは, 各成分の要素が名前でアクセスできる固定長配列と思ってよい. 各構造体は名前で区別される. 構造体は, struct 文により宣言される. 構造体が宣言されるとき, asir は内部で構造体のそれぞれの型に固有の識別番号をつける. この番号は, 組み込み関数 str_type() により取得できる. ある型の構造体は, 組み込み関数 newstruct() により生成される. 構造体の各メンバは, 演算子 -> によりアクセスする. メンバが構造体の場合, -> による指定は入れ子にできる.

```
[1] struct rat {num,denom};
0
[2] A = newstruct(rat);
{0,0}
[3] A->num = 1;
1
[4] A->den = 2;
2
[5] A;
{1,2}
[6] struct_type(A);
1
```

2.10.11 さまざまな式

主な式の構成要素としては, 次のようなものがある.

- 加減乗除, 冪
  冪は, '^' により表す. 除算 '/' は, 体としての演算に用いる. 例えば, 2/3 は有理数の 2/3 を表す. 整

数除算, 多項式除算 (剰余を含む演算) には別途組み込み函数が用意されている.

```
x+1  A^2*B*afo X/3
```

- インデックスつきの変数
  ベクトル, 行列, リストの要素はインデックスを用いることにより取り出せる. インデックスは 0 から始まることに注意する. 取り出した要素がベクトル, 行列, リストなら, さらにインデックスをつけることも有効である.

```
V[0] M[1][2]
```

- 比較演算
  等しい (‘==’), 等しくない (‘!=’), 大小 (‘>’, ‘<’, ‘>=’, ‘<=’) の 2 項演算がある. 真ならば有理数の 1, 偽ならば 0 を値に持つ.
- 論理式
  論理積 (‘&&’), 論理和 (‘||’) の 2 項演算と, 否定 (‘!’) が用意されている. 値はやはり 1, 0 である.
- 代入
  通常の代入は ‘=’ で行う. このほか, 算術演算子と組み合わせて特殊な代入を行うこともできる (‘+=’, ‘-=’, ‘*=’, ‘/=’, ‘^=’)

```
A = 2  A *= 3（これは A = A*3 と同じ; その他の演算子も同様）
```

- 函数呼び出し
  函数呼び出しも式の一種である.
- ‘++’, ‘--’
  これらは, 変数の前後について, それぞれ次のような操作, 値を表す.

```
A++  値は元の A の値, A = A+1
A--  値は元の A の値, A = A-1
++A  A = A+1, 値は変化後の値
--A  A = A-1, 値は変化後の値
```

2.10.12 プリプロセッサ

Asir のユーザ言語は C 言語を模したものである. C の特徴として, プリプロセッサ cpp によるマクロ展開, ファイルのインクルードがあるが, Asir においてもユーザ言語ファイルの読み込みの際 cpp を通してから読み込むこととした. これによりユーザ言語ファイル中で #include, #define, #if などが使える.

- #include
  UNIX では インクルードファイルは, Asir のライブラリディレクトリ (環境変数 ASIR_LIBDIR で指定されたディレクトリ) と #include が書かれているファイルと同じディレクトリをサーチする. UNIX 以外では cpp に特に引数を渡さないため, #include が書かれているファイルと同じディレクトリのみをサーチする.
- #define
  これは, C におけるのと全く同様に用いることができる.
- #if
  /*, */ によるコメントは入れ子にできないので, プログラムの大きな部分をコメントアウトする際に,#if 0, #endif を使うと便利である.

次の例は, ‘defs.h’ にあるマクロ定義である.

```
#define ZERO 0
#define NUM 1
```

```
#define POLY 2
#define RAT 3
#define LIST 4
#define VECT 5
#define MAT 6
#define STR 7
#define N_Q 0
#define N_R 1
#define N_A 2
#define N_B 3
#define N_C 4
#define V_IND 0
#define V_UC 1
#define V_PF 2
#define V_SR 3
#define isnum(a) (type(a)==NUM)
#define ispoly(a) (type(a)==POLY)
#define israt(a) (type(a)==RAT)
#define islist(a) (type(a)==LIST)
#define isvect(a) (type(a)==VECT)
#define ismat(a) (type(a)==MAT)
#define isstr(a) (type(a)==STR)
#define FIRST(L) (car(L))
#define SECOND(L) (car(cdr(L)))
#define THIRD(L) (car(cdr(cdr(L))))
#define FOURTH(L) (car(cdr(cdr(cdr(L)))))
#define DEG(a) deg(a,var(a))
#define LCOEF(a) coef(a,deg(a,var(a)))
#define LTERM(a) coef(a,deg(a,var(a)))*var(a)^deg(a,var(a))
#define TT(a) car(car(a))
#define TS(a) car(cdr(car(a)))
#define MAX(a,b) ((a)>(b)?(a):(b))
```

C のプリプロセッサを流用しているため, プリプロセッサは $ を正しく処理できない. たとえば LIST が定義されていても LIST$は置換されない. $ の前に空白をおいて LIST $ と書かないといけない.

2.10.13 オプション指定

ユーザ定義関数が N 変数で宣言された場合, その関数は, N 変数での呼び出しのみが許される.

```
[0] def factor(A) { return fctr(A); }
[1] factor(x^5-1,3);
evalf : argument mismatch in factor()
return to toplevel
```

不定個引数の関数をユーザ言語で記述したい場合, リスト, 配列を用いることで可能となるが, 次のようなより分かりやすい方法も可能である.

```
% cat factor
def factor(F)
{
    Mod = getopt(mod);
    ModType = type(Mod);
    if ( ModType == -1 ) /* 'mod' is not specified. */
        return fctr(F);
    else if ( ModType == 0 ) /* 'mod' is a number */
        return modfctr(F,Mod);
}

[0] load("factor")$
[1] factor(x^5-1);
[[1,1],[x-1,1],[x^4+x^3+x^2+x+1,1]]
[2] factor(x^5-1|mod=11);
[[1,1],[x+6,1],[x+2,1],[x+10,1],[x+7,1],[x+8,1]]
```

2 番目の `factor()` の呼び出しにおいて, 関数定義の際に宣言された引数 `x^5-1` の後ろに `mod=11` が置かれている. これは, 関数実行時に, `mod` という keyword に対して 11 という値を割り当てることを指定している. これをオプション指定と呼ぶことにする. この値は `getopt(mod)` で取り出すことができる. 1 番目の呼び出しのように `mod` に対するオプション指定がない場合には, `getopt(mod)` は型識別子 -1 のオブジェクトを返す. これにより, 指定がない場合の動作を if 文により記述できる. '`|`' の後ろには, 任意個のオプションを, '`,`' で区切って指定することができる.

```
[100] xxx(1,2,x^2-1,[1,2,3]|proc=1,index=5);
```

さらに, オプションを `key1=value1,key2=value2,...` のように '`,`' で区切って渡す代わりに, 特別なキーワード `option_list` とオプションリスト `[["key1",value1],["key2",value2],...]` を用いて渡すことも可能である.

```
[101] dp_gr_main([x^2+y^2-1,x*y-1]|option_list=[["v",[x,y]],["order",[[x,5,y,1]]]]);
```

特に, 引数なしの `getopt()` はオプションリストを返すので, オプションをとる関数から, オプションをとる関数を呼び出すときには有用である.

```
% cat foo.rr
def foo(F)
{
    OPTS=getopt();
    return factor(F|option_list=OPTS);
}

[3] load("foo.rr")$
[4] foo(x^5-1|mod=11);
[[1,1],[x+6,1],[x+2,1],[x+10,1],[x+7,1],[x+8,1]]
```

2.11 モジュール

ライブラリで定義されている関数, 変数をカプセル化する仕組みがモジュール (module) である. はじめにモジュールを用いたプログラムの例をあげよう.

```
module stack;

static Sp $
Sp = 0$
static Ssize$
Ssize = 100$
static Stack $
Stack = newvect(Ssize)$
localf push $
localf pop $

def push(A) {
  if (Sp >= Ssize) {print("Warning: Stack overflow\nDiscard the top"); pop();}
  Stack[Sp] = A;
  Sp++;
}
def pop() {
  local A;
  if (Sp <= 0) {print("Stack underflow"); return 0;}
  Sp--;
  A = Stack[Sp];
  return A;
}
endmodule;

def demo() {
  stack.push(1);
  stack.push(2);
  print(stack.pop());
  print(stack.pop());
}
```

モジュールは module モジュール名 ～ endmodule で囲む. モジュールは入れ子にはできない. モジュールの中だけで使う大域変数は static で宣言する. この変数はモジュールの外からは参照もできないし変更もできない. モジュールの外の大域変数は extern で宣言する.

モジュール内部で定義する関数は localf を用いて宣言しないといけない. 上の例では push と pop を宣言している. この宣言は必須である.

モジュール moduleName で定義された関数 functionName をモジュールの外から呼ぶには moduleName.functionName(引数 1, 引数 2, ...) なる形式でよぶ. モジュールの中からは, 関数名のみでよい. 次の例では, モジュールの外からモジュール stack で定義された関数 push, pop を呼んでいる.

```
stack.push(2);
print( stack.pop() );
2
```

モジュールで用いる関数名は局所的である．つまりモジュールの外や別のモジュールで定義されている関数名と同じ名前が利用できる．

モジュール機能は大規模ライブラリの開発を想定している．ライブラリを必要に応じて分割ロードするには，関数 module_definedp() を用いるのが便利である．デマンドロードはたとえば次のように行なえば良い．

```
if (!module_definedp("stack")) load("stack.rr") $
```

asir では局所変数の宣言は不要であった．しかしモジュール stack の例を見れば分かるように，local A; なる形式で局所変数を宣言できる．キーワード local を用いると，宣言機能が有効となる．宣言機能を有効にすると，宣言されてない変数はロードの段階でエラーを起こす．変数名のタイプミスによる予期しないトラブルを防ぐには，宣言機能を有効にしてプログラムするのがよい．

モジュール内の関数をそのモジュールが定義される前に呼び出すような関数を書くときには，その関数の前でモジュールを次のようにプロトタイプ宣言しておく必要がある．

```
/* Prototype declaration of the module stack */
module stack;
localf push $
localf pop $
endmodule;

def demo() {
  print("-----------------");
  stack.push(1);
  print(stack.pop());
  print("---------------");
}

module stack;
  /* The body of the module stack */
endmodule;
```

モジュールの中からトップレベルで定義されている関数を呼ぶには，下の例のように :: を用いる．

```
def afo() {
  S = "afo, afo";
  return S;
}
module abc;
localf foo,afo $

def foo() {
  G = ::afo();
  return G;
}
```

```
def afo() {
  return "afo, afo in abc";
}
endmodule;
end$

[1200] abc.foo();
afo, afo
[1201] abc.afo();
afo, afo in abc
```

2.12 デバッガ

2.12.1 デバッガとは

C 言語で書かれたプログラムのためのデバッガ dbx は, ソースレベルでのブレークポイントの設定, ステップ実行, 変数の参照などが可能な強力なデバッガである. Asir では, dbx 風のデバッガを用意している. デバッグモードに入るには, トップレベルで debug; と入力する.

```
[10] debug;
(debug)
```

その他, 次の方法, あるいは状況でデバッグモードに入る.

- 実行中ブレークポイントに達した場合
- 割り込みで 'd' を選択した場合
- 実行中エラーを起こした場合 この場合, 実行の継続は不可能であるが, 直接のエラーの原因となった
- ユーザ定義函数の文を表示してデバッグモードに入るため, エラー時における変数の値を参照でき, デバッグに役立たせることができる.
- error() が呼び出された場合

2.12.2 コマンドの解説

コマンドは dbx のコマンドの内必要最小限のものを採用した. 更に, gdb のコマンドからもいくつか便利なものを採用した. 実際の機能は dbx とほぼ同様であるが, step, next は, 次の行ではなく次の文を実行する. 従って, 1 行に複数の文がある場合は, その文の数だけ next を実行しなければ次の行に進めない. また, dbx と同様 '.dbxinit' を読み込むので, dbx と同じ alias を使うことができる.

step
: 次の文を実行する. 次の文が函数を含むとき, その函数に入る.

next
: 次の文を実行する.

finish
: 現在実行中の函数の実行が終了した時点で再びデバッグモードに入る. 誤って step を実行した場合に有効である.

cont

quit
: デバッグモードから抜け, 実行を継続する.

up [n]

スタックフレームを 1 段 (引数 n がある時は n 段) 上がる. これにより, そのスタックフレームに属する変数の値の参照, 変更ができる.

down [n]
: スタックフレームを 1 段 (引数 n がある時は n 段) 下がる.

frame [n]
: 引数がないとき, 現在実行中の函数を表示する. 引数があるとき, スタックフレームを番号 n のものに設定する. ここでスタックフレームの番号とは where により表示される呼び出し列において, 先頭に表示される番号のことである.

list [startline]
list function
: 現在行, または startline, または function の先頭から 10 行ソースファイルを表示する.

print expr
: expr を表示する.

func function
: 対象函数を function に設定する.

stop at sourceline [if cond]
stop in function
: sourceline 行目, または function の先頭にブレークポイントを設定する. ブレークポイントは, 函数が再定義された場合自動的に取り消される. if が続く場合, cond が評価され, それが 0 でない場合に実行が中断し, デバッグモードに入る.

trace expr at sourceline [if cond]
trace expr in function
: stop と同様であるが, trace では単に expr を表示するのみで, デバッグモードには入らない.

delete n
: ブレークポイント n を取り消す.

status
: ブレークポイントの一覧を表示する.

where
: 現在の停止点までの呼び出し列を表示する.

alias alias command
: command に alias の別名を与える.

print の引数として, トップレベルにおけるほとんどすべての式がとれる. 通常は, 変数の内容の表示が主であるが, 必要に応じて次のような使い方ができる.

- **変数の書き換え**　実行中のブレークポイントにおいて, 変数の値を変更して実行を継続させたい場合, 次のような操作を行えばよい.

```
(debug) print A
A = 2
(debug) print A=1
A=1 = 1
(debug) print A
A = 1
```

- **函数の呼び出し**　函数呼び出しも式であるから, print の引数としてとれる.

```
(debug) print length(List)
length(List) = 14
```

この例では, 変数 List に格納されているリストの長さを length() により調べている.

```
(debug) print ctrl("cputime",1)
ctrl("cputime",1) = 1
```

この例は, 計算開始時に CPU 時間の表示の指定をし忘れた場合などに, 計算途中でデバッグモードから指定を行えることを示している. また, 止むを得ず計算を中断しなければならない場合, デバッグモードから bsave() などのコマンドにより途中結果をファイルに保存することもできる.

```
(debug) print bsave(A,"savefile")
bsave(A,"savefile") = 1
```

デバッグモードからの函数呼び出しで注意すべきことは, print の引数がユーザ定義函数の呼び出しを含む場合, その函数呼び出しでエラーが起こった場合に元の函数の実行継続が不可能になる場合があるということである.

2.12.3 デバッガの使用例

ここでは, 階乗を再帰的に計算させるユーザ定義函数を例として, デバッガの実際の使用法を示す.

```
% asir
[0] load("fac")$
[3] debug$
(debug) list factorial
1   def factorial(X) {
2       if ( !X )
3           return 1;
4       else
5           return X * factorial(X - 1);
6   }
7   end$
(debug) stop at 5                    <-- ブレークポイントの設定
(0) stop at "./fac":5
(debug) quit                         <-- デバッグモードを抜ける
[4] factorial(6);                    <-- factorial(6) の呼び出し
stopped in factorial at line 5 in file "./fac"
5           return X * factorial(X - 1);
(debug) where                        <-- ブレークポイントまでの呼び出し列の表示
factorial(), line 5 in "./fac"
(debug) print X                      <-- X の値の表示
X = 6
(debug) step                         <-- ステップ実行（函数に入る）
stopped in factorial at line 2 in file "./fac"
2 if ( !X )
(debug) where
factorial(), line 2 in "./fac"
factorial(), line 5 in "./fac"
(debug) print X
X = 5
(debug) delete 0                     <-- ブレークポイント 0 の消去
```

```
(debug) cont                          <-- 実行継続
720                                   <-- 結果 = 6!
[5] quit;
```

2.12.4 デバッガの初期化ファイルの例

前に述べた通り, Asir は, 起動時に ‘$HOME/.dbxinit’ を読み込む. このファイルは, dbx のさまざまな初期設定用のコマンドを記述しておくファイルであるが, Asir は, alias 行のみを認識する. 例えば,

```
% cat ~/.dbxinit
alias n next
alias c cont
alias p print
alias s step
alias d delete
alias r run
alias l list
alias q quit
```

なる設定により, print, cont など, デバッグモードにおいて頻繁に用いられるコマンドが, それぞれ p, c など, 短い文字列で代用できる. また, デバッグモードにおいて, alias コマンドにより alias の追加ができる.

```
lex_hensel(La,[a,b,c],0,[a,b,c],0);
stopped in gennf at line 226 in file "/home/usr3/noro/asir/gr"
226        N = length(V); Len = length(G); dp_ord(0); PS = newvect(Len);
(debug) p V
V = [a,b,c]
(debug) c
...
```

2.12.5 文法の詳細

<式>:

- ‘(’<式>‘)’
- <式> <二項演算子> <式>
- ‘+’ <式>
- ‘-’ <式>
- <左辺値>
- <左辺値> <代入演算子> <式>
- <左辺値> ‘++’
- <左辺値> ‘--’
- ‘++’ <左辺値>
- ‘--’ <左辺値>
- ‘!’ <式>
- <式> ‘?’ <式> ‘:’ <式>
- <函数> ‘(’ <式並び> ‘)’
- <函数> ‘(’ <式並び> ‘|’ <オプション並び> ‘)’

<文字列>
<指数ベクトル>
<アトム>
<リスト>

(See section さまざまな式.)

<左辺値>:
<変数> [‘[’<式>‘]’]*

<二項演算子>:
‘+’ ‘-’ ‘*’ ‘/’ ‘%’ ‘^’(冪)
‘==’ ‘!=’ ‘<’ ‘>’ ‘<=’ ‘>=’ ‘&&’ ‘||’

<代入演算子>:
‘=’ ‘+=’ ‘-=’ ‘*=’ ‘/=’ ‘%=’ ‘^=’

<式並び>:
<空>
<式> [‘,’ <式>]*

<オプション>:
alphabet で始まる文字列 ‘=’ <式>

<オプション並び>:
<オプション>
<オプション> [‘,’ <オプション>]*

<リスト>:
‘[’ <式並び> ‘]’

<変数>:
大文字で始まる文字列 (X,Y,Japan など)

(See section 変数と不定元.)

<函数>:
小文字で始まる文字列 (fctr,gcd など)

<アトム>:
<不定元>
<数>

<不定元>:
小文字で始まる文字列 (a,bCD,c1_2 など)

(See section 変数と不定元.)

<数>:
  <有理数>
  <浮動小数>
  <代数的数>
  <複素数>

(See section 数の型.)

<有理数>:
  0, 1, -2, 3/4

<浮動小数>:
  0.0, 1.2e10

<代数的数>:
  newalg(x^2+1), alg(0)^2+1

(See section 代数的数に関する演算.)

<複素数>:
  1+@i, 2.3*@i

<文字列>:
  ‘"’ で囲まれた文字列

<指数ベクトル>:
  ‘<<’ <式並び> ‘>>’

(See section グレブナ基底の計算.)

<文>:
  <式> <終端>
  <複文>
  ‘break’ <終端>
  ‘continue’ <終端>
  ‘return’ <終端>
  ‘return’ <式> <終端>
  ‘if’ ‘(’ <式並び> ‘)’ <文>
  ‘if’ ‘(’ <式並び> ‘)’ <文> ‘else’ <文>
  ‘for’ ‘(’ <式並び> ‘;’ <式並び> ‘;’ <式並び> ‘)’ <文>
  ‘do’ <文> ‘while’ ‘(’ <式並び> ‘)’ <終端>
  ‘while’ ‘(’ <式並び> ‘)’ <文>
  ‘def’ <函数> ‘(’ <式並び> ‘)’ ‘{’ <変数宣言> <文並び> ‘}’

‘end(quit)’ <終端>

(See section 文.)

<終端>:
‘;’ ‘$’

<変数宣言>:
[‘extern’ <変数> [‘,’ <変数>]* <終端>]*

<複文>:
‘{’ <文並び> ‘}’

<文並び>:
[<文>]*

2.13 有限体における演算

2.13.1 有限体の表現および演算

Asir においては, 有限体は, 正標数素体 $GF(p)$, 標数 2 の有限体 $GF(2^n)$, $GF(p)$ の n 次拡大 $GF(p^n)$ が定義できる. これらは全て, setmod_ff() により定義される.

```
[0] P=pari(nextprime,2^50);
1125899906842679
[1] setmod_ff(P);
1125899906842679
[2] field_type_ff();
1
[3] load("fff");
1
[4] F=defpoly_mod2(50);
x^50+x^4+x^3+x^2+1
[5] setmod_ff(F);
x^50+x^4+x^3+x^2+1
[6] field_type_ff();
2
[7] setmod_ff(x^3+x+1,1125899906842679);
[1*x^3+1*x+1,1125899906842679]
[8] field_type_ff();
3
[9] setmod_ff(3,5);
[3,x^5+2*x+1,x]
[10] field_type_ff();
4
```

setmod_ff() は, さまざまなタイプの有限体を基礎体としてセットする. 引数が正整数 p の場合 $GF(p)$, n 次多項式 $f(x)$ の場合, $f(x) \mod 2$ を定義多項式とする $GF(2^n)$ をそれぞれ基礎体としてセットする. また, 有限素体の有限次拡大も定義できる. 詳しくは **数の型**を参照. setmod_ff() においては引数の既約チェックは行わず, 呼び出し側が責任を持つ.

基礎体とは, あくまで有限体の元として宣言あるいは定義されたオブジェクトが, セットされた基礎体の演算に従うという意味である. 即ち, 有理数どうしの演算の結果は有理数となる. 但し, 四則演算において一方のオペランドが有限体の元の場合には, 他の元も自動的に同じ有限体の元と見なされ, 演算結果も同様になる.

0 でない有限体の元は, 数オブジェクトであり, 識別子の値は 1 である. さらに, 0 でない有限体の元の数識別子は, $GF(p)$ の場合 6, $GF(2^n)$ の場合 7 となる.

有限体の元の入力方法は, 有限体の種類により様々である. $GF(p)$ の場合, simp_ff() による.

```
[0] P=pari(nextprime,2^50);
1125899906842679
[1] setmod_ff(P);
1125899906842679
[2] A=simp_ff(2^100);
3025
[3] ntype(@@);
6
```

また, $GF(2^n)$ の場合いくつかの方法がある.

```
[0] setmod_ff(x^50+x^4+x^3+x^2+1);
x^50+x^4+x^3+x^2+1
[1] A=@;
(@)
[2] ptogf2n(x^50+1);
(@^50+1)
[3] simp_ff(@@);
(@^4+@^3+@^2)
[4] ntogf2n(2^10-1);
(@^9+@^8+@^7+@^6+@^5+@^4+@^3+@^2+@+1)
```

有限体の元は数であり, 体演算が可能である. @ は $GF(2^n)$ の, $GF(2)$ 上の生成元である. 詳しくは **数の型**を参照.

2.13.2 有限体上での 1 変数多項式環の演算

fff では, 有限体上の 1 変数多項式に対し, 無平方分解, DDF, 因数分解, 多項式の既約判定などの関数が定義されている.

いずれも, 結果は [**因子**, **重複度**] のリストとなるが, 因子は monic となり, 入力多項式の主係数は捨てられる. 無平方分解は, 多項式とその微分との GCD の計算から始まるもっとも一般的なアルゴリズムを採用している.

有限体上での因数分解は, DDF の後, 次数別因子の分解の際に, Berlekamp アルゴリズムで零空間を求め, 基底ベクトルの最小多項式を求め, その根を Cantor-Zassenhaus アルゴリズムにより求める, という方法を実装している.

2.13.3 小標数有限体上での多項式環の演算

小標数有限体係数の多項式に限り, 多変数多項式の因数分解が組み込み関数として実装されている. 関数は sffctr() である. また, modfctr() も, 有限素体上で多変数多項式の因数分解を行うが, 実際には, 内部で十

分大きな拡大体を設定し, sffctr() を呼び出して, 最終的に素体上の因子を構成する, という方法で計算している.

2.13.4 有限体上の楕円曲線に関する演算

有限体上の楕円曲線に関するいくつかの基本的な演算が, 組み込み関数として提供されている.

楕円曲線の指定は, 長さ 2 のベクトル [a b] で行う. a, b は有限体の元で, setmod_ff() で定義されている有限体が素体の場合, $y^2 = x^3 + ax + b$, 標数 2 の体の場合 $y^2 + xy = x^3 + ax^2 + b$ を表す.

楕円曲線上の点は, 無限遠点も込めて加法群をなす. この演算に関して, 加算 (ecm_add_ff()), 減算 (ecm_sub_ff()) および逆元計算のための関数 (ecm_chsgn_ff()) が提供されている. 注意すべきは, 演算の対象となる点の表現が,

無限遠点は 0. それ以外の点は, 長さ 3 のベクトル [x y z]. ただし, z は 0 でない. という点である. [x y z] は斉次座標による表現であり, アフィン座標では [x/z y/z] なる点を表す. よって, アフィン座標 [x y] で表現された点を演算対象とするには, [x y 1] なるベクトルを生成する必要がある. 演算結果も斉次座標で得られるが, z 座標が 1 とは限らないため, アフィン座標を求めるためには x, y 座標を z 座標で割る必要がある.

2.14 Risa/Asir と os_muldif.rr

Risa/Asir は有理関数の計算を行う場合に注意が必要で, 特に有理関数係数の多項式や微分作用素, 成分が有理関数の行列の演算が, そのままでは思い通りにならないことがある. いくつかの例を挙げてみよう.

```
[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
```

このような場合も、より望まれる形で結果が得られるように作ったライブラリが os_muldif.rr [O6] である.

以下, 現状の関数の解説があるが, 流動的かつ暫定的なものである.

- Risa/Asir のサーチパス（標準的には，RisaAsir—のライブラリのある./lib/の下の./lib/asir-contrib で，names.rr などがインストールされている場所に置く．その場所は Risa/Asir を起動して which("names.rr") とすれば表示される）に os_muldif.rr を入れると Risa/Asir において load("os_muldif.rr")$ とすることにより，os_muldif.rr の中の函数が使用できるようになる．
- デフォルトではモジュール化して読み込まれるので，函数名の先頭に os_md. をつけて呼び出さなくてはなくてはならない．
- load("names.rr") も実行しておけば（最近の Risa/Asir のパッケージでは動時に読み込まれていることがあり，そのときはこれは不要），数式や結果を読みやすい TEX のソースで出力可能できる．さらに MS Windows においては，dviout にパスを通しておけば，dviout で表示することが出来る．他のシステムでも簡単な変更で，TEX のプレビュ機能を使って結果が表示可能になる．
- chkfun(1,0) によって Version 番号が分かる（0 から始まるものは暫定版）．
- 使用例が書いていないものは未テストなので，正しく動作しない可能性が高い．
- また？の印が付いているものも同様である．
- 不具合は知らせていただけと有り難い．

このライブラリの特徴的な関数としては，dviout(), show(), ltotex(), fctrtos(), mtotex(), myhelp(), xy2graph(), mtoupper(), mdivisor(), getbygrs(), shiftop(), m2mc() などがある．

2.14.1 os_muldif.rr のインストール

- get_rootdir() で示される Risa/Asir のライブラリがインストールされたディレクトリの下の（Risa/Asir のライブラリ・サーチパス）lib\asir-contrib に os_muldif.rr を入れる[*1]．
  Risa/Asir から

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

  として読み込むと，os_muldif.rr で定義された函数が使えるようになる．
- デフォルトではモジュール化して読み込まれるので，以下に説明される函数名の先頭に os_md. をつけて呼び出さなくてはなくてはならない．
  以下では，例を除いて先頭の os_md. が全て省略されているので注意．
- Risa/Asir の起動時に自動的に読み込むには，get_rootdir() にある .asirrc に

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

  の 1 行を追加しておけばよい．そのときの .asirrc は例えば以下のようである．

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

- os_muldif.rr の冒頭の

[*1] names.rr などが置かれている場所なので，Risa/Asir のコマンドラインから which("names.rr") などとすれば，そのディレクトリが分かる．

```
#define USEMODULE 1
```

を

```
#undef USEMODULE
```

と書き換えると，モジュール化されずに読み込まれるので，函数の先頭に os_md. をつける必要はなくなる．しかしながら，他の函数とのバッティングが起こる可能性が生じるので，それは推奨されない．

- よく使う関数のみ os_md. を略した形や簡単な名前にするには，その関数を呼び出す新たな関数を再定義しておくとよい．たとえば

```
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,...")$
```

というような内容を mydef.rr というファイルに書いてライブラリに入れて.asirrc に以下の一行を加えておけばよい．

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

- TEX を利用して数式を綺麗に表示するには，OS や動作環境に依存するプレビューアの指示などの設定が必要となる．これについては risatex.bat の項の前後を参照してください．
- myhelp("*fn*") によって函数 *fn* の説明を表示させる機能の有効化については，DVIOUTH の項を参照してください．また，os_muldif.dvi と os_muldif.pdf は get_rootdir()/help に入れてください．こうすれば dviout にパスが通っていると，DVIOUTH の設定は不要です．

3 Functions

3.1 Functions related to differential operators

以下の函数は module 化され，函数名の先頭に `os_md.` をつけて，`os_md.muldo()` のように呼び出す.

3.1.1 Fundamental functions

1. `muldo(`p_1`,`p_2`,[`x`,`∂_x`]|lim=`n`)` または `muldo(`p_1`,`p_2`,`x`|lim=`n`)`
   `muldo(`p_1`,`p_2`,[[`x_1`,`∂_{x_1}`],[`x_2`,`∂_{x_2}`],...]|lim=`n`)` または `muldo(`p_1`,`p_2`,[[`x_1`],`x_2`,...]|lim=`n`)`
   :: 有理函数（初等函数でもよい）係数の常（または偏）微分作用素の積 ($\Leftarrow [\partial_x, x] = 1$)
   - 有理函数は多項式の商で表される．多項式の係数は普通は有理数とするが，実部と虚部が有理数の複素数でもよい（cf. **数の型**）.
   - ∂_x が標準的な `d`x のときは，`[`x`,`∂_x`]` の代わりに x としてよい.
     これは以下の函数でも同様とする.
   - 偏微分作用素のときは，3 番目の引数が `[[x],[y,v],z]` などでもよい．これは `d` を付加して `[[x,dx],[y,v],[z,dz]]` と解釈される．ただしリストの最初はリストでなくてはならない．すなわち `[[x,dx],[y,dy]]` は `[[x],y]` と書いてもよいが，`[x,y]` は常微分作用素と解釈される.
     以下の函数で微分作用素とある場合は，同様な省略が可能である.
   - p_2 はベクトルまたは行列でもよい．このときは p_1 は行列でもよい.
   - x や x_i が 0 のときは，単なる積を返す.
   - $\exp(\partial)$ などの無限階微分作用素や ∂^{-1} などの擬微分作用素も計算可能（ただし，`appldo()` などでは不可，また計算が有限回で終わらないこともあり，そのときは途中で打ち切る）.
   - Leibniz 公式での積の計算で，各変数の微分の回数が 100 回以上になるとそこで打ち切るのがデフォルトであるが `lim=`n のオプションで変更可能（これは p_1, p_2 がスカラーのときのみ有効）.
     たとえば，`muldo(dx^(-1),1/x,x)` や `muldo(exp(dx),1/x,x)` は有限回で計算が終わらない.

   $[\mathtt{dx}, \mathtt{x}] = 1$, $[\mathtt{dy}, \mathtt{y}] = 1$ という交換関係のとき（$\Leftarrow$ `d`$x = \frac{\partial}{\partial x}$, `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] は

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

を意味する．なお，[6], [8], [9], [10], [11] の結果は，微分と函数とが可換と考えて割り算をして並べ直したものが通常の表記での結果となる．すなわち

[6] は

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

[8] は

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

[10] は

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

と解釈する．[11] は，実際は

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

である．

2. muledo(p_1,p_2,[x,∂_x]) または muledo(p_1,p_2,x)
   :: Euler 型常微分作用素の積 ($\Leftarrow [\partial_x, x]=x$)
   p_2 は行列またはベクトルでもよい．このとき，p_1 は行列でもよい．
   [dx, x] = x という交換関係のとき（$\Leftarrow$ d$x=x\frac{d}{dx}$）

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

3. transpdo(p,[[x_1,∂_{x_1}],[x_2,∂_{x_2}],...],[[y_1,∂_{y_1}],[y_2,∂_{y_2}],...]|ex=1)
   :: 微分作用素の変換（$x_i\mapsto y_i=y_i(x)$, $\partial_{x_j}\mapsto\partial_{y_j}=c_j(x)+\sum_\nu a_{j\nu}(x)\,\partial_{x_\nu}$）
   - y_i, ∂_{y_j} は $(x_1,\ldots,x_n)$ の有理式および有理式係数の微分作用素
   - $[x_i,\partial_{x_i}]$ の方は x_i の形の省略形も可（$\Leftarrow$ $\partial_{x_i}=$ dx_i）
   - 引数のリストの 2 番目が 3 番目より長い場合，残りは恒等変換とみなす．

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

- 上の [0], [1], [3] では，[[x_1,∂_{x_1}],[x_2,∂_{x_2}],...] について省略形を用いている (cf. muldo()).
- [0] や [1], [2] では，$x \mapsto \frac{1}{x}$ や $(x,y) \mapsto (x+y, x-y)$ という座標変換による微分作用素の変換を求めている．このとき，対応する変換 $\frac{d}{dx} \mapsto -\frac{1}{x^2}\frac{d}{dx}$ や $(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}) \mapsto (\frac{1}{2}\frac{\partial}{\partial x} + \frac{1}{2}\frac{\partial}{\partial y}, \frac{1}{2}\frac{\partial}{\partial x} - \frac{1}{2}\frac{\partial}{\partial y})$ も指定する必要がある．
- 座標変換ではないが $(x, \frac{d}{dx}) \mapsto (x, \frac{d}{dx} + c)$ のような微分作用素環の同型変換となるものでもよい．
- ex=1 : $(x_i, \partial_{x_i}) \mapsto (S_i(x,\partial), T_i(x,\partial))$ という微分作用素環の準同型写像による変換を計算する．すなわち $[T_i(x,\partial), S_j(x,\partial)] = \delta_{i,j}$ であって，p は (x,∂) について多項式である必要がある．
  [3] は，$(x, \frac{d}{dx}) \mapsto (-\frac{d}{dx} + \frac{1}{x}, x)$ という変換などを考えている．

4. translpdo(p,[[x_1,∂_{x_1}],[x_2,∂_{x_2}],...],mat)
   :: 微分作用素の線形座標変換（$x_i \mapsto \sum_j (mat)_{ij} x_j$）
   [x_i,∂_{x_i}] は x_i の形の省略形も可（$\Leftarrow$ $\partial_{x_i} = \mathtt{d}x_i$）.

```
[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
```

5. appldo(p,r,[x,∂_x]) または appldo(p,r,[[x_1,∂_{x_1}],[x_2,∂_{x_2}],...])
   :: 微分作用素（の行列）の有理式や初等函数（の行列）への作用の計算
   r はベクトルまたは行列でもよい．この場合は p は行列でもよい．

```
[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) ]
```

6. adj(p,[x,∂_x]) または adj(p,[[x_1,∂_{x_1}],[x_2,∂_{x_2}],...])
   :: 微分作用素（の行列） p の formal adjoint

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

7. `sftpexp(`p`,[`x`,`∂_x`],`q`,`$r`$`)` または `sftpexp(`$p$`,[[`$x_1$`,`$\partial_{x_1}$`],...],`$q$`,`$r`$`])`
:: 微分作用素 p を $q^{-r}\circ p\circ q^r$ と変換する
q, r は有理式．ただし，r は（x_ν を含まない）パラメータ．

```
[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] show(@@);
```

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

8. `appledo(`p`,`r`,[`x`,`∂_x`])`
:: Euler 型常微分作用素の有理式への作用の計算

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

9. `divdo(`p_1`,`p_2`,[`x`,`∂_x`]|rev=1)`
:: 常微分作用素の割り算
   - 戻り値 $[q, r, m]$
     $\Rightarrow m * p_1 = q * p_2 + r \quad (\mathrm{ord}\, m = 0,\ \mathrm{ord}\, r < \mathrm{ord}\, p_2)$
   - `rev=1` を指定すると
     $p_1 * m = p_2 * q + r \quad (\mathrm{ord}\, m = 0,\ \mathrm{ord}\, r < \mathrm{ord}\, p_2)$

```
[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
```

10. `mygcd(`p_1`,`p_2`,[`x, ∂_x`]|rev=1,dviout=`n`)` または `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`)`
:: 有理函数係数の常微分作用素（または x の多項式，または正整数）p_1 と p_2 の GCD（最大公約元）
   - 戻り値を R とおくと GCD は $R[0] = R[1] * p_1 + R[2] * p_2$ となる．
     $R[3] * p_1 + R[4] * p_2 = 0$ が成立し，行列 $\begin{pmatrix} R[1] & R[2] \\ R[3] & R[4] \end{pmatrix}$ は可逆．
     また $R[3] * p_1 = -R[4] * p_2$ が LCM となる．
   - `rev=1` を指定すると，上で積の順序が全て逆になる．
   - ユークリッドの互除法による計算であるが，p_1 と p_2 の大きさが同じ場合は p_1 を p_2 で割ることから始める．
   - `dviout=0`：ユークリッド互除法での各ステップでのデータがリストのリストで返される．
     整数のときは，リストの最初は，最初の整数の組，その後は商と余りの組のリストが，余りが 0 に

なるまで続く．
多項式と常微分作用素の時は，最初の式の組，その後は商と余りとスカラー倍の 3 つのデータの組（divdo() の返す値）が続くリスト．

- dviout=1：ユークリッドの互除法の割り算が TEX を使って示される．
- dviout=2：ユークリッドの互除法が行列の形で TEX を使って示される．
- dviout=-1,-2：上の TEX のソースが返される．

```
[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)$
```

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

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

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

11. `mylcm(`p_1`,`p_2`,[`x,∂_x`]|rev=1)` または `mylcm(`p_1`,`p_2`,[`x`]|rev=1)`
`mylcm(`p_1`,`p_2`,`x`)`, `mylcm(`p_1`,`p_2`,0)`
:: 有理函数係数の常微分作用素（または x の多項式，または正整数）p_1 と p_2 の LCM（最小公倍元）

```
[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]]
```

12. `m1div(`m`,`n`,[`x,∂_x`])` または `m1div(`m`,`n`,[`x`])` または `m1div(`m`,`n`,`x`)`
:: 有理函数係数の常微分作用素（or x の多項式）の正方行列 m と有理式（or x を含まない有理式）の正方行列 n に対し，$m = R[1](\partial_x - n) + R[0]$（or $m = R[1](x-n)+R[0]$）となるリスト $R = [R[0], R[1]]$ を返す．$R[0]$ は微分（or x）を含まない．
13. `qdo(`p_1`,`p_2`,[`x,∂_x`])`
:: 常微分方程式 $p_1 u = 0$ に対し $q_1 p_2 u = 0$ となる微分作用素 q_1 と $q_2 p_2 u = u$ となる微分作用素 q_2 のリスト $[q_1, q_2]$ を返す
q_1, q_2 は以下の以下の性質をもつ．
$qp_2u = 0 \ \Rightarrow \ \exists r$ 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)]
```

14. `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`)`
:: 有理函数係数の常微分作用素/1 変数多項式や整数の行列の単因子を得る
- 有理函数係数の常微分作用素の行列の場合は [O2, Lemma 1.10] に基づく．
- `step=1`：行と列の基本変形の途中過程を示す
- `trans=1`：単因子，左からかける行変形行列，右からかける列変形行列の 3 成分のリストを返す．
  `trans=2`：上の 3 成分の後に，行変形行列の逆行列，列変形行列の逆行列を加えた 5 成分のリストを返す．
- m が可逆 $\Leftrightarrow$ 単因子が $[1, 1, \ldots]$．この場合は `trans=1` を指定したときの戻り値を R とおくと，

$R[1]$ が逆行列，$R[2]$ は単位行列となる．

- dviout=1：基本変形の途中過程を TEX で表示．
- dviout=-1：dviout=1 の TEX のソースを返すが，表示はしない．
- dviout=2：基本変形の途中過程を TEX で表示．左右からかける行列も表示．
- dviout=-2：dviout=2 の TEX のソースを返すが，表示はしない．
- dviout=3：基本変形の結果と変換行列とその逆行列を表示（trans=2 の情報）．
- dviout=-3：dviout=3 の TEX のソースを返すが，表示はしない．
- unim() によって単因子計算や対角化の演習のための正方整数行列の生成ができる．

```
[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,

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

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

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

上の `dviout=2` のオプションの結果から分かるように，以下のアルゴリズムで計算を行う．なお，以下にある行列 C は `mygcd()` によって求めている．

行列 A の成分はユークリッド環 R で、以下のいずれかとします.

(1) 整数環

(2) 有理函数係数の 1 変数多項式環

(3) 有理函数係数の常微分作用素環

アルゴリズムは以下の通り

(a) A が零行列なら [0] を返す

(b) A の零でない最小の (i,j) 成分を，行の交換，列の交換で $(1,1)$ 成分に移す（(i,j) は辞書式順序で最小なもの）．ここで最小とは，(1) 絶対値, (2) 次数, (3) 階数 が基準

(c) A の 1 列目の 2 行目以下に零でない成分があれば，$(1,1)$ 成分とその最初の零でない成分を並べたサイズ 2 の列ベクトルに適当な $GL(2,R)$ の元 C を左からかけて，ベクトルの第 2 成分を零にするものを求める（← ユークリッドの互除法）．$SL(2,R)$ でなくて $GL(2,R)$ にしたのは，変換後の第 1 成分の係数が (2), (3) の時に多項式となるようにするため．1 行目と対応する行に左から C をかけて行変形する．そのあと (b) へ.

(d) A の 1 列目が第 1 成分を除いて零のとき．第 1 行の第 2 成分以降で零でないものがあれば，3. と同様にサイズ 2 の行ベクトルとそれに右からかける $GL(2,R)$ の元 C を求めて列変形を行い，(b) へ.

- 以下，$(1,1)$ 成分を除いて A の 1 行目と 1 列目の成分が全て 0 とする -

(e) A の行または列のサイズが 1 のとき，[$(1,1)$ 成分] を返す.

– 以下はそれ以外 –

(f) A の $(1,1)$ 成分が可逆のとき，それを 1 と置き直して，以下の (g) ii) と同様なことを行う.

— (1) または (2) のとき —

(g) $(1,1)$ 成分で割り切れない成分があるかどうか調べる.

i) 割り切れない成分をもつ列があれば，その列を 1 列目に加え，その成分が 2 行目でなければ 2 行目と行を交換．その後 (b) へ

ii) すべて割り切れれば，1 行目と 1 列目を除いた行列を $(1,1)$ 成分で割ってできる，行と列のサイズが 1 ずつ小さな行列，を引数として，この函数を呼び，戻り値の各成分を $(1,1)$ 成分倍したリストに $(1,1)$ 成分を追加したリストを返す.

— (3) のとき —

(h) 零でない $(0,0)$ 成分以外の (i,j) 成分を選んで（辞書式順序で最小な (i,j) を選ぶ）それを P とし，Px^k と $(1,1)$ 成分の作る左イデアルが $(1,1)$ 成分の作る左イデアルに入らない k を求める（k

は (i,j) 成分の階数以下の非負整数となる)．なお x は微分との交換子が恒等写像となるかけ算作用素．j 列目に x^k を右からかけたものを 1 列目に加えて (b) へ.

`step=1` を指定したときの表示される行列の上のコメントの意味は以下の通り.

- 最初の数字は，`mdivisor()` が呼ばれたネスティングの深さで，1 段深まると行列サイズが 1 減る (cf. (f), (g) ii))．
- `start`：この函数が呼び出されたときの行列.
- `(a,b) -> (1,1)`：`a` 行目と 1 行目，`b` 列目と 1 列目を交換して，`(a,b)` 成分を `(1,1)` に移動する（cf. (b)）.
- `line 1 & a`：1 行目と `a` 行目に左から $GL(2,R)$ をかけて `(a,1)` 成分を 0 にする（cf. (c)）.
- `column 1 & b`：1 列目と `b` 列目に右から $GL(2,R)$ をかけて `(1,b)` 成分を 0 にする（cf. (d)）.
- `unit`：`(1,1)` 成分が可逆元のときの処理（cf. (f)）.
- `column 1 += col b, line 2<->a`：1 列目に `b` 列目を加えて，2 行目と `a` 行目を交換（cf. (g) i)）.
- `column 1 += col b*x^k`：`b` 列目に右から `x^k` をかけたものを 1 列目に加える（cf. (h)）.

一方，TEX においては，i 行目を Li と表し, j 列目を Cj と表して，たとえば $\begin{pmatrix}1&2\\3&4\end{pmatrix}$ を 3 行目と 5 行目にかけるのは

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

と表す.

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

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

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

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

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

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

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

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

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)$
```

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

$C1\leftrightarrow C2$

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

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

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

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

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

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

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

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

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

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

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

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

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

As a result,

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

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

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

15. `sqrtdo(`p`,[`x`,`∂_x`])` ?
:: $x \mapsto 1/x$ で（x のべき倍を除いて）不変な微分作用素 p に対する変数変換 $x \mapsto y = x + \sqrt{x^2-1}$ $x = \frac{1}{2}(y + y^{-1})$ に注意
16. `toeul(`p`,[`x`,`∂_x`],`n`)`
:: 確定特異点型常微分作用素を $x = n$ で Euler 型に変換
変数変換 $x \mapsto x + n$ のあと $x\partial_x$ を ∂_x で置き換え，多項式係数にする．
ただし，n が文字列 “infty” のときは，変数変換 $x \mapsto 1/x$ のあと $x\partial_x$ を ∂_x で置き換える．
最後に x の巾を掛けて $x = 0$ で消えない作用素に調整する．

```
[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
```

17. `fromeul(`p`,[`x`,`p_x`],`n`)`
:: Euler 型常微分作用素を元に戻す（`toeul(`$p,[x,\partial_x],n$`)` の逆変換）
最後に $x-n$ の巾をかけて $x=n$ で消えない多項式係数とする．

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

18. `expat(`p`,[`x`,`∂_x`],`n`)`
:: 確定特異点型常微分作用素の $x=n$ での特性指数を求める
- n は文字列 “infty” でもよい（$n=\infty$ に対応）．
- Fuchs 型常微分作用素で n が文字列 “?” のときは，全ての特異点とそこでの特性指数とを求める．
- 以下，例で現れる `ghg()` については，(59) を参照

```
[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]]]
```

19. `sftexp(`p`,[`x`,`∂_x`],`n`,`r`)`
:: 常微分作用素 p を $(x-n)^{-r}\circ p\circ(x-n)^r$ に変換する
ただし，$[\partial_x,r]=0$ とする（以下同様）．

```
[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]
```

20. `fractrans(`p`,[`x`,`∂_x`],`n_0`,`n_1`,`n_2`)`
:: 常微分作用素 p に $(n_0,n_1,n_2)\mapsto(0,1,\infty)$ という一次分数変換を行う

```
[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]
```

21. chkexp(p,[x,∂_x],n,r,m)
:: $j=0,\dots,m-1$ の全てに対し，確定特異点型常微分作用素 p の解 u_j で $(x-n)^{-(r+j)}u_j$ が $x=n$ で正則でそこでの値が 1 となるものが存在する条件を求める．

```
[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]
```

22. soldif(p,[x,∂_x],n,q,m)
:: 常微分作用素 p の $x=n$ の近傍での $z^q(1+\sum_{j=1}^{\infty}c_jz^j)$ の形の形式解に対し，長さ $m+1$ のベクトル [1 c_1 c_2 $\cdots$ c_m] を返す（$z=x-n$）．
$z^j\log z$ の項が現れる解も構成される．？
n ="infty" のときは，$x=\infty$, $z=1/x$ とする．

```
[0] R=os_md.soldif(os_md.ghg([a,b],[c]),x,"infty",a,4);
[ 1 (a^2+(-c+1)*a)/(a-b+1) (1/2*a^4+(-c+2)*a^3+(1/2*c^2-5/2*c+
...
[1] fctr(dn(R[4]));
[[1,1],[a-b+1,1],[a-b+2,1],[a-b+3,1],[a-b+4,1]]
[2] fctr(nm(R[4]));
[[1/24,1],[a,1],[a+1,1],[a+2,1],[a+3,1],[a-c+1,1],[a-c+2,1],
[a-c+3,1],[a-c+4,1]]
```

23. okuboetos(p, [x,∂_x] | diag=[$c_1,c_2,\dots$])
:: 単独 m 階 Okubo 型方程式 $pu=0$（n 階の項の係数が n 次以下の多項式）を Okubo 型の 1 階のシステムに変換する
- 戻り値は [[$a_0,\dots,a_{m-1}$], B, T]
- オプション diag=[$c_0,c_1,\dots$] によって，[$a_0,\dots,a_{m-1}$] の順序を変えたものを指定することができる．
- 単独 Fuchs 型のとき，$p\mapsto p*\partial_x+p'$ の変換を何度か施すと Okubo 型になる

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

```
[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 ]]
```

以上から

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

特に

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

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

:: 1 階の常微分方程式系を単独高階に直す

- 方程式系 $u_i' = \sum_{j\geq 0} p_{ij}(x) u_j$ の u_m の満たす方程式を返す．p は有理式を成分とする正方行列.
- m がリスト $[m_1, m_2]$ のときは，u_{m_1} を u_{m_2} を用いて表す (u_{m_2} に施す微分作用素)．
- m が負数のときは，$[-m, 0]$ を表す.

```
[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([a1,a2,a3,a4],dx);
(-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
```

25. `dform(`ℓ, x`|dif=1)`
:: 変数 $x[0], x[1], \ldots$ の 1 次微分形式 $\sum \ell[i][0]d(\ell[i][1])$, または 2 次微分形式 $\sum \ell[i][0]d(\ell[i][1]) \wedge d(\ell[i][2])$ の計算. `dif=1` は 1 次微分形式の外微分の計算.
- $\ell[i][j]$ は行列でもよい
- `dif=1` は，1 次形式の外微分を計算する

```
[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]]
```

上の計算は，以下を意味する．

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

26. `solpokubo(`p`,[`x,∂_x`],`n`)`
:: 単独 Okubo 型常微分作用素 p の n 次の固有多項式と固有値を求める
Okubo 型常微分作用素とは，k 階微分の係数が k 次以下の多項式で，最高階は階数と次数が等しい作用素

```
[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

この項は，[O2] の主要結果（基本的部分は [O1] で解説）を Risa/Asir 上で実現したものとなっている．

27. `laplace(`p`,[`x,∂_x`])`
`laplace(`p`,[[`x_1,∂_{x_1}`],[`x_2,∂_{x_2}`],...])`
:: 微分作用素 p の（部分）Laplace 変換
$(x,\partial_x)\mapsto(-\partial_x,x)$

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

28. `laplace1(`p`,[`x,∂_x`])`
`laplace1(`p`,[[`x_1,∂_{x_1}`],[`x_2,∂_{x_2}`],...])`
:: 微分作用素 p の（部分）逆 Laplace 変換
$(x,\partial_x)\mapsto(\partial_x,-x)$

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

29. `mc(`p`,[`x,∂_x`],`r`)`
:: 常微分作用素 p の middle convolution $mc_r(p)$

```
[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]]]
```

30. `mce(`p`,[`x,∂_x`],`n`,`r`)`
:: 常微分作用素 p を $(\partial_x-n)^{-r}\circ p\circ(\partial_x-n)^r$ と変換して常微分作用素に戻す．
ただし n は，∂_x と可換とする．$n=0$ のときは middle convolution $mc_r(p)$.

31. rede(p,[x,∂_x])
rede(p,[[x_1,∂_{x_1}],[x_2,∂_{x_2}],...])
:: 微分作用素 p の reduced representative を返す
32. ad(p,[x,∂_x],f)
:: 常微分作用素 p の ∂_x を $\partial_x - f$ に置き換える変換
33. add(p,[x,∂_x],f)
:: 常微分作用素 p の ∂_x を $\partial_x - f$ に置き換える addition，すなわち 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
```

34. vadd(p,[x,∂_x],[[c_0,r_0],[c_1,r_1],...])
:: versal addition add(p,[x,∂_x], $\sum_{j\geq 0}\frac{r_j x^j}{\prod_{\nu=0}^{j}(1-c_\nu x)}$)
35. addl(p,[x,∂_x],f)
:: 常微分作用素の addition の Laplace 変換 laplace1(add(laplace()))

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

36. cotr(p,[x,∂_x],f)
:: 常微分作用素 p の $x \mapsto f(x)$ による座標変換
37. rcotr(p,[x,∂_x],f)
:: 常微分作用素 p の $x \mapsto f(x)$ による座標変換の reduced representative
38. s2sp(p|num=1,std=k,short=1)
:: スペクトル型を表す文字列と “数のリストのリスト” との変換
- $10,11,12,\ldots,35$ は，a,b,c,...,z で表す．負の数や ^ や分数も有効.
- ^ は（べき乗ではなく）同じ数の繰り返しを表す.
- 10 以上の数を ( ) で括って表してもよいが，括弧のネスティングは不可.
- num=1 を指定すると，a,b,c,... は戻り値に使わずに括弧で括って表す.
- 引数においては，$36,37,\ldots,60$ を $A,B,\ldots,Z$ と表してもよい.
- 有理数以外は< >で囲った文字列に変換される.
- std=k を指定すると戻り値を, 各特異点での重複度を大きい順にして
  $k=1$ の時は各特異点を重複度の辞書式順序を小さい順に
  $k=-1$ のときは各特異点を重複度の辞書式順序を大きい順に
  並べたスペクトル型とする.
- short=1 を指定すると，同じ重複度が 4 個以上続くときに ^* を使う短縮形の文字列に変換する.
  このとき std=k のオプションも同次指定可能.

```
[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
```

39. `chkspt(`m`|mat=1)` または `chkspt(`m`|opt=`t`)` または `fspt(`m`,`t`)`
:: 分割の組 m（スペクトルタイプ）または generalized Riemann scheme（GRS）をチェックして $[pts, ord, idx, fuchs, rod, redsp, fspt]$ を返す
pts 特異点の数（分割の組の数）
ord 階数
idx rigidity index
$fuchs$ Fuchs の関係式
rod reduction での階数減
$redsp$ reduction exponents のリスト
$fspt$ 対応する basic なスペクトルタイプ

`mat=1` は，Schleginger 型を意味する.

以下順に $t = 0, 1, \ldots$ であり `fspt(`m`,`t`)` としてもよいが，文字列により opt を指定した場合 `chkspt()` のみ可能で

- `opt="sp"` or 0: 与えられた GRS のスペクトルタイプを返す.
- `opt="basic"` or 1: `chkspt(`m`)` の戻り値のリストの 7 番目の $fspt$ のみを返す．realizable でないときは 0 を返す.
- `opt="constuct"` or 2: basic なスペクトルタイプからの構成を示す．realizable でないときは 0 を返す.
- `opt="strip"` or 3: generalized Riemann scheme またはスペクトルタイプから重複度 0 の項を削る.
- `opt="short"` or 4: GRS を短縮形に
- `opt="long"` or 5: GRS を短縮形から戻す
- `opt="sort"` or 6: スペクトル型を各点でソートして返す
- `opt="root"` or 7: Kac-Moody Weyl 群での構成を示す.
  戻り値のリストは 3 成分で，最初は base，次は与えたもの，最後が鏡映のリスト．鏡映は 3 成分で最初が内積の値，次とその次で，枝の番号と頂点からの番号.

短縮形とは，重複度が 1 の exponent は，重複度込みのリストから exponent のみ短くしたもの.

```
[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]]]
```

40. `spgen(`n`|eq=1,str=1,std=fpt=[`k,ℓ`],sp=`m`,basic=1)`
:: 階数 n 以下の rigid な分割の組（あるいは与えられたものの軌道）を得る
n が 0 や負のときは，rigidity index が n の basic なものを得る．
n は -10 以上に対応しているが，`okubo` [O5] が動作する環境にしておけば，この制限はない．
- `eq=1`：位数が丁度 n のもの（$n>1$ のとき）
- `str=1`：文字列で得る
- `pt=[`k,ℓ`]`：分割が k 組以上 ℓ 組以下のもの
- `sp=`m：スペクトル型が m のものから階数増加の方向で得られるもの
さらにここで `basic=1` を指定すると，階数増加の方向に限らないもの
- `std=`f：各特異点の重複度は大きい順に，特異点間では $f=1$ 小さい（resp. 大きい）順に並べる．

```
[0] os_md.spgen(4|eq=1,pt=[4,4]);
[[[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=[4,4],str=1));
```

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

[0] 階数が 4 で特異点が 4 個の rigid なスペクトル型のリストを得る
[1] 同じスペクトル型を文字列形式で得て表示
[2] rigid 指数が -2 で特異点数 4 のスペクトル型を文字列形式で得て表示
プログラムの例は

```
/* 8 階で特異点が 4 個のリジッドなスペクトル型を辞書式順序の大きい順に
   並べたリストを得る（文字列で）*/
Rank=8; /* give rank */
G=os_md.spgen(Rank|eq=1,pt=[4,4]); /* get spectral types */
for(L=[];G!=[];G=cdr(G))
  L=cons(os_md.s2sp(os_md.s2sp(car(G)|std=-1)),L);
L=msort(L,[-1,0]);
```

41. `sproot(`p`,`t`|dviout=1,only=`k`,null=1)`
:: スペクトル型を与えて構成やルートの情報を示す．t=`"base"`, `"length"`, `"type"`, `"part"`, `"pair"`, `"pairs"`, sp
- `length`：basic に変換する Weyl 群の長さ（$= \#\Delta(\mathbf{m})$）
- `base`：`chksp(`p`|opt="root")` と同じ．単純鏡映での最短積
- `type`：上に対応する型（$= [\Delta(\mathbf{m})]$，cf. [O2, (7.40)]）.
- `height`：root の高さ（単純ルートの一次結合で表示したときの係数の和．cf. [O2, (7.35)]）
- `part`：上の Weyl 群の元で正負が変わるルート（$= \Delta(\mathbf{m})$：既約条件に対応．cf. [O2, (7.30)]）の情報を得る．最初が対応する basic なルート，2 項目が与えられたルート，3 項目が求めるルートおよび与えられたルートとの内積の組のリスト．
- `pair`, `pairs`：既約分解に対応するルートの分解
  `dviout=1` と `only=`k が指定可能．ルートは対応する分割で示す．
  `iand`$(k,1)=1$：分割の相手が分割に対応するもの
  `iand`$(k,2)=2$：分割の相手の位数が 0 のルートとなるもの
  `iand`$(k,4)=4$：分割の相手が負ルートとなるもの
  `null=1`：該当のものがなければ出力しない
- sp：スペクトル型を与えた場合は，対応するルートの内積を与える

```
[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);
```

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

上の右辺の最初の項は $\Delta(\mathbf{m})$ に属する実正ルートとなる．また，`"pair"` のときは，上の最後の 2 行が異なり，$20,20,20,200 \oplus 11,11,02,011$ などとなる．

[O1, 第 8 章] や [O2, Chapter 7] にスペクトル型 $\mathbf{m} = (m_{j,\nu})$ と Kac-Moody ルート系のルート α との対応が解説されている．

$$\alpha_{\mathbf{m}} = n\alpha_o + \sum_{j\geq 0}\Big(\sum_{\nu>i} m_{j,\nu}\Big)\alpha_{j,i}, \quad n = m_{j,1} + m_{j,2} + \cdots \quad (j \text{ に依らない})$$

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

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

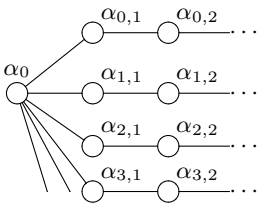
$$\texttt{"31,31,22,211"} \leftrightarrow \alpha = 4\alpha_0 + \alpha_{0,1} + \alpha_{1,1} + 2\alpha_{2,1} + 2\alpha_{3,1} + \alpha_{3,2}$$

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

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

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

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



Gauss の超幾何に対応する"11,11,11"のときは

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

であって，height は $2+1+1+1=5$ で，rigid な場合は $[\Delta(\mathbf{m})]$ は (height で与えられる数 -1) という自然数の分割となるが，今の場合は $4=1+1+1+1$ である．また base で与えられた 3 番目の項のリスト

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

は，3 つの数字の 2 項目と 3 項目によって

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

を意味し，最初の数字は対応する鏡映による height の変化の数を示している．
なお，一般に $w_{\mathbf{m}} = s_{\alpha_{i_0}}s_{\alpha_{i_1}}\cdots s_{\alpha_{i_K}}$ のときは

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

であり，$\mathbf{m}$ が rigid なときは $w_{\mathbf{m}}\alpha_{\mathbf{m}} = \alpha_0$ となる長さ $(=K)$ 最小の Weyl 群 (鏡映 s_α $(\alpha\in\Pi)$ で生成される群) の元．なお $m_{j,\nu}$ は $m_{j,1}\geq m_{j,2}\geq\cdots$ となるように正規化して考えるので，$i_0=0$ である．

```
/* 8 階で特異点が 4 個以上に対応するリジッドなスペクトル型の分解で,
   Type 2 と Type 3 のものを全て得る */
[10] Rank=8; /* give rank */
[11] G=os_md.spgen(Rank|eq=1,str=1,pt=[4,100]); /* get spectral types */
[12] 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);
```

42. sp2grs(m,a,ℓ|mat=1)
:: spectral type から generalized Riemann scheme を生成する
- 逆の変換は，chkspt() で opt="sp"を指定したもの．
- $i+1$ 番目の特異点の $j+1$ 番目のスペクトルパラメータは a が変数の時 aij で、リストの時 $a[i]j$ で表される．a や $a[i]$ は文字列でもよい．
- ℓ が $[\ell_1,\ell_2]$ というリストの時，ℓ_1 番目の特異点における ℓ_2 番目の exponent は，Fuchs の関係式を満たすように決められる．その exponent が 0 のときは，別の exponent で調整される．
- ℓ が正整数 k のとき，aij は $aij+k$ または，$a[i]j+k$ となる．前者と併用するときは，$\ell=[\ell_1,\ell_2,k]$ というリストにする．
- ℓ が負整数 $-1-k$ のときは，標準的にいくつかの exponents を 0 とし，上の k シフトを行う．
- mat=1 は，Schleginger 型を意味する（Fuchs の関係式が異なる）．

```
[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
```

43. `ssubgrs` (m, ℓ)
:: Generalized Riemann scheme m の ℓ に対する特性指数和

```
[0] GRS=os_md.sp2grs("111,111,21",[a,b,c],[1,3,-2]);
[[[1,a1],[1,a2],[1,-c-b1-b2-a2-a1+2]],[[1,b1],[1,b2],[1,0]],
 [[2,0],[1,c]]]
[1] os_md.ssubgrs(GRS,"010,100,10");
b1+a2
[2] os_md.ssubgrs(GRS,"011,110,11");
-a1+2
```

44. `mcgrs(`m`,[`$r_1, r_2, \ldots, r_n$`]|mat=1)`
:: middle convolution と addition を generalized Riemann scheme に施す
r_j がスカラーの時は mc_{r_j} を，r_j がリスト $[r_{j,0}, \ldots, r_{j,p}]$ のときは addition を意味する（$r_{j,0}$ は無視される）．施す順序は，最初が r_n，次に r_{n-1} という順で，最後が r_1.

45. `mcop(`p`,[`$r_1, r_2, \ldots, r_n$`],[`$x, x_1, x_2, \ldots$`])`
:: middle convolution と addition を (偏）微分作用素 p に順に施していく
   - `[`$x, x_1, x_2, \ldots$`]` によって，x は変数，$x_1, x_2, \ldots$ は有限の特異点の位置を示す.
   - `[`$r_1, r_2, \ldots$`]` において，$r_j = [r_{j,0}, r_{j,1}, \ldots]$ は middle convolution と additions を表す.
     - まず middle convolution $mc_{r_{j,0}}$ を施し、その後 $x = x_\nu$ における additions（関数 u の $(x - x_\nu)^{r_{j,\nu}} u$ へのゲージ変換に対応）を行う（$\nu = 1, 2, \ldots$）.
     - p に対して r_j に対応する変換を，r_1, $r_2, \ldots$ の順に行う.

```
[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] show(@@)$
```

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

46. `redgrs(`m`|mat=1)`
:: 常微分作用素の generalized Riemann scheme の 1-step reduction
結果のデータ $[redsp, m']$ を返す.
reduction 出来ないときは非負整数（basic, 0 は rigid) または負の値（非存在）を返す.

47. `getbygrs(`m`,`t`|perm=`ℓ`,var=`v`,pt=[`$p_1,\ldots$`],mat=1)` または
`getbygrs(`m`,[`$t,s_1,s_2,\ldots$`]|perm=`ℓ`,ver=`v`,pt=[`$p_1,\ldots$`],mat=1)`
:: generalized Riemann scheme (GRS) で定義される Fuchs 型常微分方程式の解析（GRS は短縮形またはスペクトルタイプでもよい）
   - `mat=1` は，Schleginger 型を意味する.
   - `getbygrs(0,0)` で渡されるパラメータが示される.
   - m の要素の数を $p+1$ 個とすると，それは $p+1$ 個の特異点に対応する．それを順に $x_0=\infty, x_1=0, x_2=1, x_3,\ldots,x_p$ とする.
   - Fuchs 型方程式の特異点 $x=x_j$ における generalized exponents（一般特性指数）が $\{[\lambda_0]_{(n_0)},\ldots,[\lambda_k]_{(n_k)}\}$ であるとは exponents が重複度を込めて $\{\lambda_j+\nu;\ 0\le\nu<n_j,\ j=0,\ldots,k\}$ であって，$\{\lambda_0,\ldots,\lambda_k\}$ に互いに差が整数となる要素が含まれない場合（そうでない場合の定義は略すが，たとえば λ_j が全て等しいなら対応する局所モノドロミー群の Jordan ブロックへの分解サイズが，$n=n_0+\cdots+n_k$ の分割の双対分割に対応すること），$x=x_j$ での局所モノドロミー群が半単純（対角化可能）となるとき，すなわち $(x-x_j)^{\lambda_j+\nu}\phi_{j,\nu}(x)$ $(0\le\nu<n_\nu)$ という形の局所解が存在することをいう（$\phi_{j,\nu}(x)$ は，$x=x_j$ で正則で，$\phi_{j,\nu}(x_j)=1$).
   - 各特異点での generalized exponents を並べて表にしたものを generalized Riemann scheme (GRS) という（cf. [O1, 第 5 章]，[O2, Definition 4.6]）：

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

   - `perm=[[`i,j`]]`：i 番目の特異点と j 番目の特異点の入れ替え
   - `perm=[`$j_0,\ldots,j_p$`]`：特異点の並べ替え（x_{j_0} の特性指数が ∞ に移る).
   - `var=`v：exponets の変数の指定．`sp2grs()` の第 2 変数の指定にあたる.
   - `pt=[`$p_1,p_2,\ldots$`]`：特異点の位置を ∞, p_1, p_2, ... とする.
   - Generalize Riemann scheme m の特性指数の一つを指定せずに，それを `"?"` で表してもよい．その特性指数は Fuchs の関係式から決定される.
   - m がスペクトル型（分割の組）のとき，対応する GRS に変換される．このとき，$s=$`"general"` を指定できる（指定しないと，いくつかの exponents が 0 として，Okubo 型またはそれに近いものにする).

$R=$`getbygrs(`m`,`t`)` とする.
オプション・パラメータ s は一つの文字列，あるいは，複数の文字列を指定する．`"TeX"`, `"dviout"`, `"keep"` は，以下のどの機能においても指定できる.
   - $t=$`"reduction"` or 0：basic な方程式への reduction
     $s=$`"TeX"` を指定したときは，さらに `"top0"` も指定可能
     $R[i][0]$：i 番目の reduction の x_j での exponents の位置 $R[i][0][\nu]$

($\nu = 0, \ldots, p$, $i = 1, \ldots$)
$R[i][0][\nu]$ ($\nu = 1, \ldots, p$) に対応する addition を行い，$R[i][0][0]$ に対する middle convolution を行う．
$R[i][1]$：上の reduction の結果の GRS
$R[0][1]$：元の GRS

- $t =$"construct" or 1：方程式の構成法と解の積分表示
  $R[0][1]$: reduction された basic な方程式の GRS ($\mathbf{m}'$ とおく)．
  $\mathrm{RAd}\big(\prod_{\nu=1}^{p}(x-x_\nu)^{R[i][0][\nu]}\big) \circ mc_{R[i][0][0]}$ によって GRS が $R[i-1][1]$ から $R[i][1]$ に変わり ($i = 1, 2, \ldots$)，$\mathbf{m}'$ から $\mathbf{m}$ への変換が分かる．
  $s =$"short" を指定可能
  $s =$"TeX" を指定すると，解の積分表示が得られる（局所モノドロミーの固有値の重複度が 1 の局所解）
- $t =$"connection" or 2：接続公式を求める
  $s =$"simplify" を指定可能
  特異点が $p+1$ 点のときは，特異点の位置が $x_0 = \infty, x_1 = 0, x_2 = 1, x_3, x_4, \ldots, x_p$ とする上記 GRS であって，$x = 0$ と 1 での最後の exponent の重複度が共に 1 であるとき，対応する正規化した局所解の 0 から 1 への接続係数は

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

  で与えられる．c_B は対応する basic 方程式の接続係数であるが，rigid なとき（すなわちアクセサリ・パラメータの無いとき）は $c_B = 1$ となる．特異点が 3 点のときは、$(1-x_j^{-1})^{R[2][j]}$ の項はないことに注意．
- $t =$"operator" or 3：方程式を求める
  $s =$"simplify" を指定可能．
  結果は，x が変数 x，dx が $\frac{d}{dx}$ を表す．
  アクセサリ・パラメータは ri_j と表され，それは帰着 basic 方程式の $x^j \frac{d^i}{dx^i}$ の係数にあたる．
  この計算は、他の計算に比べて重いので注意．
- $t =$"series" or 4：$x = 0$ での最後の exponent（重複度が 1 の必要あり）に対応する局所解のベキ級数表示を求める．
  $R = [R_0, R_1, R_2]$
  $R_0 = [[R_{00}, [R_{01}, T_1], [R_{02}, T_2], \ldots], [P_1, Q_1], [P_2, Q_2], \ldots]$
  $R_1 = [[R_{100}, P_{101}, P_{102}, \ldots], [R_{110}, P_{111}, P_{112}, \ldots], \ldots]$
  $R_2 = [[R_{200}, P_{201}, P_{202}, \ldots], [R_{210}, P_{211}, P_{212}, \ldots], \ldots]$
  $0 \leq P_1 < P_2 < P_3 < \cdots$

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

- $t =$ "TeX" or 5：GRS を LaTeX で表示
  $s =$ "top0" を指定すると，GRS で ∞ を最後に表示する．
- $t =$ "Fuchs" or 6：Fuchs の関係式を得る
- $t =$ "All" or 7：s に"dviout"を指定したときのみ有効で，dviout での関連する結果の表示．s に"irreducible"を指定すると既約性の判定条件の非表示，"operator"を指定すると方程式の表示も行う．

- t = "basic" or 8：帰着される basic な GRS を得る
  s ="short" を指定可能
- t = "" or 9：GRS を短縮形でない形式で示す
  s ="short" を指定すると，短縮形で示す
- t = "irreducible" or 10：既約条件を求める
  s = "simplify を指定可能
  リストの各 liner form が整数値とならないことが既約性の必要十分条件．rigid でないならば，さらに帰着された方程式の既約条件も課される．
- t = "recurrence" or 11：3 項間漸化式を示す
  ∞, 0, 1 の最後の exponents の重複度がすべて 1 で rigid のときのみ可．
  原点の規格化された局所解について上の exponents をシフトしたもの．

s は文字列または文字列のリスト．文字列は
- "simplify"：Fuchs の関係式を使って結果を簡単にする
- "TeX"：LaTeX の数式で出力する
- "dviout"：dviout で表示する
  このときオプション・パラメータ title= で文字列を指定できる
- "keep"：上と同様だが，表示はしない
- "short"：GRS を短縮形で表示
- "general"：スペクトルタイプを与えたとき，exponents を完全に generic にとる
- "x1"：特異点を，∞, x_1, x_2,... とする（GRS の TeX 表示、および作用素を得るときのみ有効）
- "x2"：特異点を，∞, 0, x_2,... とする（GRS の TeX 表示、および作用素を得るときのみ有効）
- "top0"：GRS の TeX での表示で，∞ を最後とする
- "sht"：自動生成される exponents の番号を 1（デフォルトは 0）から始める

```
[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(HO,"Fuchs");
b1+c1+a1+a0+d1+a2-3
[27] P=os_md.getbygrs(HO,"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 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

$$\begin{aligned} & P\left\{\begin{array}{cccc} x=\infty & 0 & 1 & \\ 0 & 0 & 0 & ;\ x \end{array}\right\} \\ & \quad \to \mathrm{RAd}\big(x^{c_0+a_1+b_2-1}(x-1)^{-c_0-b_2-a_2}\big) \end{aligned}$$

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

Hence an integral representation of the solution is

$$
\begin{aligned}
u(x) = x^{b_0}(1-x)^{c_0}\int_c^x\int_c^{s_0}&(x-s_0)^{a_0+b_0+c_0}s_0^{a_0+b_1+c_0-1} \\
&\cdot(s_0-s_1)^{a_1+b_1+c_0}s_1^{a_1+b_2+c_0-1}(1-s_1)^{-a_2-b_2-c_0}ds_1ds_0.
\end{aligned}
$$

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

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

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

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

corresponding to the Riemann scheme

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

The equation is irreducible if and only if

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

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

とすると，以下が表示される．
Riemann scheme

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

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

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

Irreduciblity $\Leftrightarrow$ 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 coorespond 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

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

```
[35] 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]]]
[36] 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]]
```

48. `shiftop(`ℓ, s`|zero=1,raw=`k`,all=`t`,dviout=1)`
:: rigid なスペクトル型 ℓ と shift s から shift 作用素を求める
ℓ が Riemann scheme, s がシフトするパラメータとそれの変換後の組のリストでもよい.
shift 作用素，微分作用素，Riemann scheme の 3 項のリストを返す.
   - `zero=1`：いくつかの特性指数を 0 に正規化
   - `raw=1`：定数倍も含めて正規化
     `raw=2`：左逆の shift 作用素との合成のスカラーを返す
     このスカラーが 0 となる $\Leftrightarrow$ shift 作用素が同型写像でない
     `raw=3`：shift 作用素を上のスカラーとのリストとして返す
     `raw=4`：さらに左逆 shift 作用素を上のリストの最後に加える
   - `all=0`：shift 作用素のみを返す
     `all=1`： shift 作用素，微分方程式，Riemann scheme, shift された微分作用素，shift された Riemann scheme の 5 項のリストを返す
   - `dviout=1`：結果を `dviout` によって画面表示する．オプション `all` を指定可能（`all=1` によって微分方程式も表示）.

なお，shift でずれる特性指数は，各特異点でなるべく後に指定した方が高速に計算できる.

```
[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

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

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

$$Q_1 = (2a+b+c)x(x-1)\partial + a((2a+b+c)x - a - c)$$

$$Q_2 = -(2a+b+c)x(x-1)\partial + (a+b+c)((2a+b+c)x - a - b)$$

$$Q_2Q_1 \equiv a(a+c)(a+b)(a+b+c) \mod W(x)P$$

$$P = -x(x-1)\partial^2 + ((b+c-2)x - b + 1)\partial + a(a+b+c-1)$$

49. conf1sp(m|x2=± 1,conf=0)
:: スペクトル型 m の微分作用素の Poincare rank 1 の合流過程を示す
スペクトル型を与える各分割は，順に特異点 ∞, $\frac{1}{c}$, $x_2, \ldots$ に対応するものとし，∞ に対応する分割は $\frac{1}{c}$ に対応する分割の細分を指定する（$c=0$ が合流）．ただし，x2=1 を指定しないと $x_2 = 0$ とする．x2=-1 のときは，$x_2, \ldots$ を $\frac{1}{c2}, \ldots$ とする．
conf=0 を指定すると，特異点を ∞, 0, c, $x_3, \ldots$ とし，原点と点 c の特異点の合流を示す．

```
[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)
```

50. anal2sp(m,l)
:: 同時スペクトル m の解析
$m = [[m_1, \lambda_1, \mu_1], [m_2, \lambda_2, \mu_2], \ldots]$ は重複度と同時固有値

l は以下の意味を持つ.

0	同じスペクトル型をまとめる（同時スペクトルでなくても可）.
["add",n]	スペクトル型 n を加える（同時スペクトルでなくても可）.
["sub",n]	スペクトル型 n を引く（同時スペクトルでなくても可）.
["swap"]	同時スペクト型の順序を入れ替える.
["+",c_1,c_2]	固有値をシフトする.
["*",c_1,c_2]	固有値の一次結合を返す.
["mult",m]	同次固有値の重複度を全て m 倍する.
["get",f,c]	f 番目の固有値が c となるものを抜き出す.
["put",f,c]	f 番目の固有値を c に置き換える.
["get1",f]	f 番目の固有値のみを抜き出す（結果は同時スペクトルでない）.
["get1",f,c]	f 番目の固有値が c のものを抜き出す（結果は他方のスペクトル）.
["put1",f,c]	同時でないスペクトルに，f 番目の固有値に c を追加する.
["put1"]	同じスペクトル型をコピーする.
["val",f]	f 番目のスペクトルの階数と固有値の和を返す.
[[$\cdots$],[$\cdots$],...]	複数のコマンドを続けて実行する.

```
[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]]  /* R の定義 */
[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"]);    /* 順番を逆に */
[[2,b,a],[3,c,a],[-1,c,a],[-2,b,a],[1,c,b]]
[3] os_md.anal2sp(R,0);           /* まとめる */
[[2,a,c],[1,b,c]]
[4] os_md.anal2sp(R,["*",f,-g]);  /* 固有値のシフト */
[[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,["x",f,-g]);  /* 固有値の一次結合 */
[[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([[2,a,b],[3,a,c],[-1,a,c]],["mult",2]); /* 重複度 *= m */
[[4,a,b],[6,a,c],[-2,a,c]]
[7] os_md.anal2sp(R,["get",2,c]); /* 指定した固有値のみ抽出 */
[[3,a,c],[-1,a,c],[1,b,c]]
[8] os_md.anal2sp(R,["put",1,f]); /* 固有値を置き換える */
[[2,f,b],[3,f,c],[-1,f,c],[-2,f,b],[1,f,c]]
[9] os_md.anal2sp(R,["get",1,a]); /* 指定した固有値のみ抽出 */
[[2,a,b],[3,a,c],[-1,a,c],[-2,a,b]]
[10] R1=os_md.anal2sp(R,["get1",1,a]); /* 指定した固有値の他方を抽出 */
[[2,b],[3,c],[-1,c],[-2,b]]
[11] os_md.anal2sp(R1,["put1",1,a]);
[[2,a,b],[3,a,c],[-1,a,c],[-2,a,b]]
[12] os_md.anal2sp(R1,["put1"]);
[[2,b,b],[3,c,c],[-1,c,c],[-2,b,b]]
[13] os_md.anal2sp(R,["val",1]); /* 固有値の和 */
[3,2*a+b]
[14] os_md.anal2sp(R,[["get1",1],["put1"],0]);
```

`[[2,a,a],[1,b,b]]`

51. `mc2grs(`g`,`r`|dviout=`k`)`
:: $\mathbb{P}^1$ 内の 5 点の配置に対する KZ 型方程式の同時スペクトル g の変換
KZ 型方程式

$$du = \sum_{0\le i<j\le 3} A_{i,j}\frac{d(x_i-x_j)}{x_i-x_j}u,$$
$$A_{i,4} = -\sum_{j=0}^{3} A_{i,j},\ A_{j,i} = A_{i,j},\ A_{i,i} = 0$$

において，I, $J \subset \{0,1,2,3\}$, $\#I = \#J = 2$, $I\cap J = \emptyset$ のとき

$$[A_I, A_J] = 0$$

が満たされるので，A_I と A_J の同時固有値分解が出来る．このような 15 組の $[I,J]$ に対する同時固有値（重複度と各固有値）のリスト（ヘッダは `[`I`,`J`]` で，`I=[`i_i`,`i_2`]`, `J=[`j_i`,`j_2`]` とすると，$i_1 < i_2$, $j_1 < j_2$, $i_1 < j_1$ となる）のリストを g とする．

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

のときは同時固有値のリストは `[[[0,1],[2,3]],[2,`a`,`c`],[1,`b`,`d`]]` となる．
なお，既約性から $\sum_{0\le i<j\le 3} A_{i,j}$ はスカラー行列としてよいので，そのスカラーを κ とする．
同時固有値のリストのリストを g で与えるが，それ以外の指定は

- $g = 0$ のとき，自明な `[[[[0,1],[2,3]],[1,0,0]],[[[0,1],[2,4]],[1,0,0]],...]` を意味する．
- g を，4 行列 $A_{0,4}$, $A_{0,1}$, $A_{0,2}$, $A_{0,3}$ のスペクトル型（文字列あるいは，4 点の GRS）を，この順で指定する．このときは，m2mc() におけると同様なオプション（`dep=[`m`,`n`]`，`int=0`）などの指定が可能）．なお，`top=0` を指定した場合は，$A_{0,1}$, $A_{0,2}$, $A_{0,3}$, $A_{0,4}$ の順と解釈される．
  スペクトル型や文字列で指定した場合は，以下の r が特別の意味を持つ．
  - $r = 1$：スペクトル型を標準的な順序にし，同時固有値のリストのリストをそのまま返す．
  - $r = 3$：同時固有値のリストのパラメータを `a,b,c,...` で表し，Fuchs 条件と同次固有値のリストのリストとの組をリストとして返す．

r によって，以下の機能になる．

- `0`：同時固有値のリストのリストをそのまま返す．
- `["sort"]`：ヘッダが標準的な順序になるよう，リストを並び替える（`[ ]` は省略可）．
- `["deg"]`：κ を返す（`[ ]` は省略可）．
- `[[`i`,`j`],`λ`]`：$A_{i,j} \mapsto A_{i,j}+\lambda$, $A_{i,4} \mapsto A_{i,4}-\lambda$, $A_{j,4} \mapsto A_{j,4}-\lambda$ という addition を返す $(0 \le i < j \le 3)$．
- `["homog"]`：$r =$`[[2,3],`$-\kappa$`]` に対応する addition によって $\kappa = 0$ とする（`[ ]` は省略可）．
- `["swap",[`i`,`j`]]`：添え字の i と j を交換する．i または j が 4 のときは，$r =$`["homog"]` を行ってから添え字を交換する．
- `["perm",[`$i_0,\dots,i_4$`]]`：添え字の置換 $\begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ i_0 & i_1 & i_2 & i_3 & i_4 \end{pmatrix}$ を行う．添え字 4 を変更するときは，$r =$`["homog"]` を行ってから置換する．
- `[`μ`]`：x_0 変数に対する middle convolution $\mathrm{mc}_{x_0,\mu}$ を行う（アルゴリズムは [O3] によって与えられた）．
- `[`i`,`μ`]`：x_i 変数に対する middle convolution $\mathrm{mc}_{x_i,\mu}$ を行う．ただし，$i = 4$ のときは，$r =$`["swap",[0,4]]` を行ってから $\mathrm{mc}_{x_0,\mu}$ を施し，再度 $r =$`["swap",[0,4]]` で戻す．
- `[[[`i_1`,`j_1`],`λ_1`],...,[`i_k`,`μ_k`],...]`：additions と middle convolutions を連続して行う．

- [[a,b,c]] : [[[0,1],a],[[0,2],b],[[0,3],c]] と同じ.
- [[a,b,c,μ]] : [[[0,1],a],[[0,2],b],[[0,3],c],[μ]] と同じ.
- ["get",[I,J]] : $[A_I, A_J] = 0$ に対する同時固有値を返す（ヘッダ $[I, J]$ のついたリスト）.
- ["get",I] : A_I に対する固有値を返す（ヘッダ I のついたリスト）.
- ["get",i] $(0 \le i \le 4)$: $i \in I$ を満たす A_I に対する GRS を返す（変数 x_i についての常微分方程式の GRS）.
  dviout=1 を指定すると TEX を使って表示する. dviout=-1 では TEX のソースを返す. dviout=2 を指定すると，TEX のソースを書き出すが表示はしない（次回の表示コマンドでまとめて表示する）.
- ["get"] : 10 個全ての A_I の固有値のリストを返す.
  dviout=k の指定が可能.
- ["get0",[I,J]], ["get0",I], ["get0",i] $(0 \le i \le 4)$, ["get0"] : "get0" を "get" に変えたものと同じであるが，ヘッダはつけない.
- ["get0",I,J] : $A_I + A_J$ のスペクトルを返す.
- ["show"] : 同時スペクトルを TEX を使って表示する.
  dviout=-1 の指定が可能.
- ["show0"] : 15 組の同時スペクトルの固有値の重複度を示す（15 組の順序は，"show"におけると同じ）.
  dviout=1,-1 の指定が可能.
- ["spct"] : スペクトル型を返す. 5×5 行列で (i, j) 成分は，$i \ne j$ のとき $A_{i,j}$ のスペクトル型，$i = j$ のとき x_i についての index of rigidity $(0 \le i, j \le 4)$.
  5×5 行列で (i, j) 成分は，$i \ne j$ のとき $A_{i,j}$ のスペクトル型，$i = j$ のとき x_i についての index of rigidity $(0 \le i, j \le 4)$.
  dviout=k の指定が可能.
- ["mult",ℓ_1,...,ℓ_m] : $g_0 = g$ として g_i =mc2grs(g_{i-1},ℓ_i) を $i = 1, \ldots, m$ に対して順に実行して g_m を返す. オプションも有効.

```
[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");
a+b+c
[3] os_md.mc2grs(M1,"homog");
[[[[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");
[[[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;
1
[7] os_md.mc2grs(F1,["get",0]);
```

```
[[[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);
```

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

```
[10] os_md.mc2grs(F1,"get"|dviout=1)$
```

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

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

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

```
[12] os_md.mc2grs("322,52,52,43","show0"|dviout=1)$
```

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

```
[13] os_md.mc2grs(F1,"spct"|dviout=1)$
```

	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)$
```

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

```
[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]]]
```

mc2grs() を使った関数の例を挙げる.

```
/* KZ 方程式への拡張が同一となるリジッドスペクトル型のサーチ
   S:スペクトル型 同一のものがあれば，スペクトル型の組を返す.
   S:数           S 階で 2 変数超幾何に対し，上のリストを返す. */
def sameKZ(S)
{
    if(type(S)==1){
        SS=os_md.spgen(S|eq=1,pt=[4,4]);
        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);
}

/* KZ 方程式で N 階で idx が K のものが現れるもののリストを得る */
def idxKZ(K,N)
{
    L=os_md.spgen(N|eq=1,pt=[4,4],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);
}
```

52. 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)
:: Pfaff 形式 $du = (A_0\frac{dx}{x} + A_y\frac{d(x-y)}{x-y} + A_1\frac{d(x-1)}{x-1} + B_0\frac{dy}{y} + B_1\frac{d(y-1)}{y-1})u$ の x 変数での addition+middle convolution を求める（$\ell = [A_0, A_y, A_1, B_0, B_1]$）.
ℓ がスペクトル型や Riemann scheme のとき，上の後者では s ="GRC", "GRSC", "Pfaff", "sp", "pairs", "irreducible", "All", "swap"
  - m2mc(0,0) で使い方が簡略化して表示される.
  - ℓ は $[A_0, A_y, A_1, B_0, B_1]$ という 5 個の正方行列のリストまたはベクトル.
    A_0, A_y, A_1 にそれぞれ a_0, a_y, a_1 による addition を施したあと，パラメータ c による middle convolution（MC=1 を指定した場合は convolution ）を行った 5 個の行列のベクトルを返す.
    addition を行わないときは，[a_0,a_y,a_1,c] の代わりに単に c としてよい.
    addition のみのときは，c を省いて [a_0,a_y,a_1] と指定する.
  - ℓ は x 変数のスペクトル型や Riemann scheme でもよい．スペクトル型や Riemann scheme は $x = \infty$, $x = 0$, $x = y$, $x = 1$ の順に与える．このときは a_0 や s は 0（[A_0,A_y,A_1,B_0,B_1] を返す）または次々項に挙げる文字列とする.
  - ℓ をスペクトル型で与えるときは，各特異点で重複度が大きな順に並べると結果が簡単になることが多い（オプション dep=[m,n] の項参照）.
  - a_0 や s が以下の文字列の場合は，特別の意味を持つ
    - GRS : Generelized Riemann scheme を返す．a_y に "dviout" を指定できる.
    - GRSC : 上と同じであるが，さらに $x = y = 0$, $x = y = 1$ の一般化特性指数も含める.
    - Pfaff : 文字列で Pfaff 形式を返す．a_y に "dviout" を指定できる.
    - sp : x 変数と y 変数のスペクトル型を $x = y$ から順に表す.
    - pairs, pair : 可約なときの分解を与える分割を示す.
    - irreducible : 既約条件を与えるパラメータを示す.
    - All : 上の全てを指定し，さらに．a_1 に "dviout" を指定したと見なされる．さらに GRSC を指定すると, GRS でなくて GRSC を指定したと見なされる．operator=0 のオプションで，Pfaff 形式の方程式を示さない.
    - swap : x 変数と y 変数を入れ替えた変換結果を返す.
      さらにオプション swap=t を指定すると別の変換の意味になる.
      $t = 1$: x 変数と y 変数の入れ替え（指定しない場合と同じ）.
      $t = 2$: 座標変換 $(x, y) \mapsto (x, \frac{x}{y})$ を行う.
      t = [t_0,t_1,t_2] : (t_0, t_1, t_2) は $(0, 1, 2)$ の並べ替えで，$0, 1, 2$ は順に特異点 $0, 1, \infty$ を意味する.

また，以下のオプションがある.
  - swap=1 : x と y を交換して middle convolution を行う
  - small=1 : TEX での行列を小さなサイズにする
  - simplify=0 : 対角行列による共役変換による行列の簡単化をしない
  - simplify=1,3 : 対角行列による共役変換による別の簡単化をする
  - dep=[m,n] : スペクトル型からの特性指数の自動生成で，m 番目の特異点の n 番目の特性指数を Fuchs 条件から定める．デフォルトは $m = 1$（特異点 $x = \infty$）で最後の特性指数とする.
    なお，$x = \infty$ 以外の特異点の最初の特性指数は 0 に正規化される.
  - int=0 : スペクトル型からの特性指数の自動生成で，パラメータの分数倍が現れることを許す.

```
[0] Z=newmat(1,1,[[0]]);
[ 0 ]
[2] F1=os_md.m2mc([Z,Z,Z,Z,Z],[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 ] ]
[3] os_md.m2mc(F1,["GRS","dviout"]);
```

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

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

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

```
[5] 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 ] ]
[6] os_md.m2mc(F2,["GRS","dviout"]);
```

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

```
[7] 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 ] ]
[8] os_md.m2mc("21,21,21,21",["GRSC","dviout"]|small=1);
```

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

```
[9] 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, sp, irreducible, pair, pairs, Pfaff, All
  s : TeX, dviout, GRSC
  option : swap, small, simplify, operator
  Ex: m2mc("21,21,21,21","All")

[10] 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

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

53. `mcmgrs(`g`,`r`|dviout=`k`)`

:: $\mathbb{P}^1$ 内の 5 点以上の点配置に対する KZ 型方程式の同時スペクトル g の変換

KZ 型方程式で `mc2grs()` にある正方行列 $A_{i,j}$ のインデックスが 0 から一般の $N-1$ までの場合を扱う（解を多変数超幾何関数と考えたときの変数の個数は $N-2$ 個となる）．すなわち

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

可換な組 $(A_{0,i}, A_{j,k})$ と $(A_{0,i}, A_{0,i,j})$ に対する同時固有値（重複度と各固有値）のリストを g として与えることができる（i, j, k は $\{1, \dots, N\}$ の中の互いに異なるインデックス）．前者は $N \times {}_{N-1}C_2$ 組，後者は $N \times (N-1)$ 組で，合計 $\frac{N^2(N-1)}{2}$ 組のデータとなる．スペクトル型や文字列や数で g を指定した場合は, 以下のようになる．

- g が 2 以上の整数の時，自明な g 変数の超幾何に対応する同時固有値のデータを返す ($N = g+2$)．
- g を，行列 $A_{0,N}$, $A_{0,1}, A_{0,2}, \dots, A_{0,N-1}$ のスペクトル型（文字列あるいは，4 点の GRS）で，この順で指定する．
  このとき，`short=0` を指定すると，インデックス付きのパラメータを使う．

r によって，以下の機能になる．

- `0`：同時固有値のリストのリストをそのまま返す．
- `["deg"]`：κ を返す（`[ ]` は省略可）．
- `[[`i`,`j`],`λ`]`：$A_{i,j} \mapsto A_{i,j} + \lambda$, $A_{i,N} \mapsto A_{i,N} - \lambda$, $A_{j,N} \mapsto A_{j,N} - \lambda$ という addition を返す ($0 \le i < j < N$)．
- `[`μ`]`：x_0 変数に対する middle convolution $\mathrm{mc}_{x_0,\mu}$ を行う（アルゴリズムは [O3] によって与えられた）．
- `[[[`i_1`,`j_1`],`λ_1`],...,[`μ_k`],...]`：additions と middle convolutions を連続して行う．
- `[[`a_1`,...,`a_{N-1}`]]`：`[[[0,1],`a_1`],...,[[0,N-1],`a_{N-1}`]]` と同じ．
- `[[`a_1`,...,`a_{N-1}`,`μ`]]`：`[[[0,1],`a_1`],...,[[0,N-1],`a_{N-1}`],[`μ`]]` と同じ．
- `["get",[`I`,`J`]]`：$[A_I, A_J] = 0$ に対する同時固有値を返す（ヘッダ $[I, J]$ のついたリスト）．
- `["get",`I`]`：A_I に対する固有値を返す（ヘッダ I のついたリスト）．
- `["get",`i`]` ($0 \le i \le N$)：$i \in I$ を満たす A_I に対する GRS を返す（変数 x_i についての常微分方程式の GRS）．
  `dviout=1` を指定すると TEX を使って表示する．`dviout=-1` では TEX のソースを返す．

dviout=2 を指定すると，TEX のソースを書き出すが表示はしない（次回の表示コマンドでまとめて表示する）.

- ["get"]：$\frac{(N+1)N}{2}$ 個全ての $A_{i,j}$ の固有値のリストを返す.
  dviout=k の指定が可能.
- ["get0",[I,J]], ["get0",I], ["get0",i] $(0 \le i \le N)$, ["get0"]："get0" を "get" に変えたものと同じであるが，ヘッダはつけない.
- ["show"]：同時スペクトルを TEX を使って表示する.
  dviout=-1 の指定が可能.
- ["show0"]：$\frac{N^2(N-1)}{2}$ 組の同時スペクトルの固有値の重複度を示す（順序は，"show"におけると同じ）.
  dviout=1,-1 の指定が可能.
- ["spct"]：スペクトル型を返す．$(N+1)\times(N+1)$ 行列で (i,j) 成分は，$i \ne j$ のとき $A_{i,j}$ のスペクトル型，$i=j$ のとき x_i についての index of rigidity（$0 \le i,j \le N+1$）.
  $(N+1)\times(N+1)$ 行列で (i,j) 成分は，$i \ne j$ のとき $A_{i,j}$ のスペクトル型，$i=j$ のとき x_i についての index of rigidity（$0 \le i,j \le N+1$）.
  dviout=k の指定が可能.
- ["mult",ℓ_1,...,ℓ_m]：$g_0=g$ として $g_i=$mcmgrs(g_{i-1},ℓ_i) を $i=1,\dots,m$ に対して順に実行して g_m を返す．オプションも有効.

```
[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)$
```

の実行結果は以下のようになる.

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

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

	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

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

54. mmc(l,[μ,a_1,...,a_n]|mult=f)
:: Schlesinger 型常微分方程式および KZ 型方程式の addition+middle convolution
Schlesinger 型方程式

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

または KZ 型方程式

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

の addition または middle convolution を行う．

前者の場合は ℓ に `[`$A_1, \dots, A_n$`]`，後者の場合は `[`$A_{0,1}, \dots, A_{0,n}, A_{1,2}, \dots, A_{1,n}, \dots, A_{n-1,n}$`]` ($\frac{n(n+1)}{2}$ 個の成分）を指定する．あるいはスペクトル型や GRS を指定する（後者の場合は，変数 x_0 の常微分方程式について指定する．このときはリジッドである必要がある)．

- `[`$a_1, \dots, a_n, \mu$`]` : A_j（または $A_{0,j}$）に対して a_j を加えた後，x（または x_0）変数の middle convolution mc_μ を行う，
  一方のみの指定は，`[`$a_1, \dots, a_n$`]` または `[`μ`]` とする．
- `mult=`f : $f = 0$ は Schlesinger 型，$f = 1$ は KZ 型を意味する．
  デフォルトでは，ℓ の成分の数が 6 以上の時に KZ 型，5 以下では Schlesinger 型とみなす．また，スペクトル型で指定したときは KZ 型とみなす．

55. `linfrac01(`ℓ`|over=1)`
:: $x = 0, 1, \infty, y, z, \dots$ の一次分数変換のリスト（$\ell = [x, y]$ etc.）
`linfrac01([`x, y`])` あるいは `linfrac01(newvect(10,[`$\cdots$`])` の形
後者の場合の特異点は順に
$0 : x = 0,\ 1 : x = y,\ 2 : x = 1,\ 3 : y = 0,\ 4 : y = 1,\ 5 : x = \infty,\ 6 : y = \infty,$
$7 : x = y = \cdots = 0,\ 8 : x = y = \cdots = 1,\ 9 : x = y = \cdots = \infty$
$10 : y_2 = 0,\ 11 : y_2 = x,\ 12 : y_2 = y,\ 13 : y_2 = 1,\ 14 : y_2 = \infty$
$15 : y_3 = 0,\ 16 : y_3 = x,\ 17 : y_3 = y,\ 18 : y_3 = y_2,\ 19 : y_3 = 1,\ 20 : y_3 = \infty$
$21 : y_4 = 0, \cdots$
で，一次分数変換によるそれらの置換が得られる．
ベクトルのサイズは，$10, 15, 21, 28, \dots$.
y_4 以上（リストのサイズが 5 以上）のときは `over=1` を指定．
y_k までのときは，$(k + 4)!$ 個の元が得られる．

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

56. `lft01(`t, ℓ`)`
:: $\ell = (x_1, x_2, x_3, \dots)$ に対する特殊一次分数変換
以下の座標変換．`linfrac01(`ℓ`)` の ℓ と同様なベクトルでも可

$t = 0$	$(x_2, x_1, x_3, x_4, ...)$
$t = -j$	$(x_1, x_2, ..., x_j, x_{j+2}, x_{j+1}, x_{j+3}, ...)$
$t = 1$	$(1 - x_1, 1 - x_2, 1 - x_3, 1 - x_4, ...)$
$t = 2$	$(1/x_1, 1/x_2, 1/x_3, 1/x_4, ...)$
$t = 3$	$(x_1, x_1/x_2, x_1/x_3, x_1/x_4, ...)$

3.1.3 Some operators

57. `okubo3e([`$p_{0,1}, \dots, p_{0,m}$`],[`$p_{1,1}, \dots, p_{1,n}$`],[`$p_{2,1}, \dots, p_{2,m+n}$`|opt=1])`
:: 0, 1, ∞ に確定特異点を持つ $m + n$ 階の単独 Okubo 型微分作用素を求める
0 で n 次元，1 で m 次元の正則解をもつ．Riemann scheme の記号では

$$\wp\left\{\begin{array}{lccccccccc} 0: & 0 & 1 & \cdots & n-1 & p_{0,1} & \cdots\cdots\cdots & & p_{0,m} & \\ 1: & 0 & 1 & \cdots\cdots\cdots & & m-1 & p_{1,1} & \cdots & p_{1,n} & ;x \\ \infty: & p_{2,1} & \cdots\cdots\cdots & & & & & & p_{2,m+n} & \end{array}\right\}$$

Fuchs の関係式：$\sum p_{i,j} = mn$

アクセサリパラメータが $(m-1)(n-1)$ 個あり，`r1,r2,...` で表される．

$p_{i,j}$ のうちの一つは `"?"` でもよい（値は Fuchs の関係式から計算される）．

- `opt=1` を指定すると `getbygrs()` を用いる．

```
[0] P = os_md.okubo3e([1-e,1-f],[2-g,"?"],[a,b,c,d]);
1 accessory parameters: r1,r2,...
(x^4-2*x^3+x^2)*dx^4+((a+b+c+d+6)*x^3+(-a-b-c-d-e-f-9)*x^2+(e+f+3)*x)*dx^3
+(((b+c+d+3)*a+(c+d+3)*b+(d+3)*c+3*d+7)*x^2+((-b-c-d+g-3)*a+(-c-d+g-3)*b
+(-d+g-3)*c+(g-3)*d+(-f-g-1)*e+(-g-1)*f-g^2+2*g-8)*x+(f+1)*e+f+1)*dx^2+
((((c+d+1)*b+(d+1)*c+d+1)*a+((d+1)*c+d+1)*b+(d+1)*c+d+1)*x-r1)*dx+d*c*b*a
[1] os_md.expat(P,x,1);
[-a-b-c-d+e+f+g,-g+2,1,0]
```

58. `fuchs3e([`$p_{0,1},\ldots,p_{0,n}$`],[`$p_{1,1},\ldots,p_{1,n}$`],[`$p_{2,1},\ldots,p_{2,n}$`])`
:: 0, 1, ∞ に確定特異点を持つ n 階の Fuchs 型微分作用素を求める

$$\wp\left\{\begin{array}{lcccc} 0: & p_{0,1} & p_{0,2} & \cdots & p_{0,n} \\ 1: & p_{1,1} & p_{1,2} & \cdots & p_{1,n} \quad ;x \\ \infty: & p_{2,1} & p_{2,2} & \cdots & p_{2,n} \end{array}\right\}$$

Fuchs の関係式：$\sum p_{i,j} = \frac{n(n-1)}{2}$

アクセサリパラメータが $\frac{(n-1)(n-2)}{2}$ 個あり，`ri_j` で表される（cf. `getbygrs()`）．

$p_{i,j}$ のうちの一つは `"?"` でもよい（値は Fuchs の関係式から計算される）．

59. `ghg([`$p_{1,1},p_{1,2},...,p_{1,m}$`],[`$p_{2,1},p_{2,2},...,p_{2,n}$`])`
:: 一般超幾何函数 ${}_mF_n(p_1;p_2;x)$（cf. `seriesHG()`）の満たす微分作用素

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

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

変数は `x`，微分は `dx` で表記される．

$m = n+1$ のときは Rigid な Fuchs 型で

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

Fuchs の関係式：$\gamma = \sum p_{1,\nu} - \sum p_{2,\nu}$

$m=2$, $n=1$ のときが Gauss の超幾何

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

60. `even4e([`$p_{1,1},p_{1,2},p_{1,3},p_{1,4}$`],[`$p_{2,1},p_{2,2}$`])`
:: 4 階 even family (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 の関係式: $2p_0 + p_{1,1} + p_{1,2} + p_{1,3} + p_{1,4} + p_{2,1} + p_{1,2} = 3$

61. `odd5e([`$p_{1,1},p_{1,2},p_{1,3},p_{1,4},p_{1,5}$`],[`$p_{2,1},p_{2,2}$`])`
:: 5 階 odd family (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 の関係式 $2p_0+p_{2,1}+2p_{2,2}+\sum p_{1,j}=4$

62. `rigid211([`$p_{0,1},p_{0,2}$`],[`$p_{1,1},p_{1,2}$`],[`q_0,q_1`])`
:: Type 211,211,211
$$\wp\left\{\begin{array}{lcccc} 0: & 0 & 1 & p_{0,1} & p_{0,2} \\ 1: & 0 & 1 & p_{1,1} & p_{1,2} \\ \infty: & q_0 & q_0+1 & q_1 & q_2 \end{array} ; x\right\}$$
Fuchs の関係式 $p_{0,1}+p_{0,2}+p_{1,1}+p_{1,2}+2q_0+q_1+q_2=4$

63. `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)
$$\wp\left\{\begin{array}{lcccccc} 0: & 0 & 1 & p_{2,1} & p_{2,1}+1 & p_{2,2} & p_{2,2}+1 \\ 1: & 0 & 1 & 2 & 3 & p_0 & p_0+1 \\ \infty: & p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} & p_{1,5} & p_{2,6} \end{array} ; x\right\}$$
Fuchs の関係式 $2p_0+2p_{2,1}+2p_{2,2}+\sum p_{1,j}=5$

64. `eofamily([`$p_{0,1},p_{0,2}$`],[`$p_{1,1}$`],[`$p_{2,1},\ldots,p_{2,n}$`])`
:: even/odd family (obsolete)
Even family (Rigid)
Fuchs の関係式: $p_{0,1}+(m-1)p_{0,2}+mp_{1,2}+\sum p_{2,\nu}=n-1$
$$\wp\left\{\begin{array}{lcccccccc} 0: & 0 & 1 & \cdots & m-1 & p_{0,1} & p_{0,2} & \cdots & p_{0,2}+m-2 \\ 1: & 0 & 1 & \cdots & m-1 & p_{1,2} & p_{1,2}+1 & \cdots & p_{1,2}+m-1 \\ \infty: & p_{2,1} & \cdots & p_{2,m} & & p_{2,m+1} & p_{2,m+2} & \cdots & p_{2,2m} \end{array} ; x\right\}$$
Odd family (Rigid)
Fuchs の関係式: $p_{0,1}+mp_{0,2}+mp_{1,2}+\sum p_{2,\nu}=n-1$
$$\wp\left\{\begin{array}{lcccccccc} 0: & 0 & 1 & \cdots & m-1 & p_{0,1} & p_{0,2} & \cdots & p_{0,2}+m-1 \\ 1: & 0 & 1 & \cdots & m-1 & m & p_{1,2} & \cdots & p_{1,2}+m-1 \\ \infty: & p_{2,1} & \cdots & p_{2,m} & & p_{2,m+1} & p_{2,m+2} & \cdots & p_{2,2m+1} \end{array} ; x\right\}$$

65. `ev4s(`p_1`,`p_2`,`p_3`,`p_4`,`p_5`)`
:: Heckman-Opdam 超幾何の (BC_2,BC_1) 型制限常微分 (Rigid)

```
[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]
```

66. `b2e(`p_1`,`p_2`,`p_3`,`p_4`,`p_5`)`
:: Heckman-Opdam 超幾何の (BC_2,A_1) 型制限常微分 (Non Rigid)

```
[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]
```

67. `heun([`a,b,c,d,e`],`p`,`r`|)`
:: Heun の微分方程式を与える．r はアクセサリパラメータ

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

```
[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

以下の函数は module 化され，函数名の先頭に `os_md.` をつけて `os_md.myhelp()` のように呼び出す．

3.2.1 Extended function

68. `myhelp(`h`)`
:: `os_muldif.rr` のマニュアルを表示する
- `os_muldif.pdf`, `os_muldif.dvi` を `get_rootdir()\help` に入れておく．それぞれ，$h=1,-1$ で表示される．
- DVIOUTH が正しく設定されていれば，`myhelp("m2mc")` のようにして `myhelp(`h`)` の h に函数名を文字列で入れると，その函数の解説が（`os_muldif.dvi` に含まれていれば）表示される．文字列 h には函数の先頭の `os_md.` は省略できる．
- $h=0$, `os_md.getbygrs`, `os_md.m2mc`, `os_md.mgen` でも対応するものが表示される．
- $h=[dviout,n]$: $dviout$ で `dviout` のパス名，n で表示する `dviout` の番号を設定しておく．なお，`Risa/Asir` の数式などの結果の表示に最初の `dviout` が使われるので，$n=2$ などが適当．
- 他の環境やより一般的な場合は DVIOUTH で設定する．

69. `chkfun(`f`,`s`)`
:: 函数 f（= 文字列）が定義済みかどうか調べ，未定義なら `load(`s`)` を実行
- $f=0$: `Risa/Asir` の version 表示
- $f=1$: `os_muldif.rr` の version 表示
- $s=0$: f が文字列の時，その函数名の函数が定義済みかどうかのみを調べる
- f はモジュール化された函数には対応していない

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

70. `isMs()`

:: Micorosoft Windows 環境かどうか調べる
環境変数 temp, tmp を調べて 3 文字目が ¥(あるいは \ ) であれば 1 を返す

71. isyes(*p*|set=*ℓ*)
:: 1 か 0 を返す函数を定義し，それを使う
- set=1 によって定義する．このとき *p* は函数とそれに渡すパラメータリストと判断に使うその函数の値をチェックする範囲との 3 つからなるリスト.
  すなわち $p=$[*fn*,[t_1,t_2,...],[*a*,*b*]] ならば，isyes(X|set=*p*) は
  ```
  def foo(X){
    R=fn(X,t1,t2,...);
    return (R>=a && R<=b)?1:0;
  }
  ```
  という関数に対応する.
  - *p* =[type,[],[0,1]] のときは，type(*p*) の値が 0 以上 1 以下の時は 1(yes) を，そうでないときは 0(no) を返す函数の定義．$-\infty$ や ∞ は "" で表す.
  - 範囲が 1 つの値ならば，その値でよい．よって [type,[],1] は [type,[],[1,1]] と同じ.
  - 複数の条件を課すときは，上の 3 つ組をさらにリストにする．この場合は全てが yes のときのみ 1 を返す函数を定義する.
  - *p*=[[os_md.isint,[],1],[os_md.calc,[[">",0]],1]]] は，正の整数のときのみ 1 を返す函数の定義.
  - 新たに set=1 で定義するまでは前回の定義が有効.
- set=0 では，現在定義されているものを返す（以下の例の [11] を参照）.
- set= の右辺に上の *p* で与える設定を指定し，*p* にはその設定で判定すべきものを渡す．このときは，定義されていた設定は破壊されない.
- set= を指定しないときは，定義に従って *p* を判定して 1 か 0 を返す.

```
[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
```

上の [11] は以下と同じ.

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

72. isall(f,m)
:: m の要素に p 対して $f(p)$ が 0 となるものが存在すると 0，そうでなければ 1 を返す
m がリスト，ベクトル，行列でなければ m 自身を m の要素とみなす（isyes() の例を参照）.

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

73. ptype(p,ℓ)
:: ℓ は変数，または変数のリストで，それのみを変数とみなした type() を返す
- ptype(p,vars(p)) は type(p) と同じ.
- 有理式であって，その分母に与えられた変数の（多項式以外の）初等関数を含む場合は 128 を，分子に与えられた変数の初等関数を含む場合は 64 を戻り値に加えて返す.

```
[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
```

74. keyin(s)
:: s を表示し，1 行のキー入力を待って，それを文字列として返す
行末の改行記号を削除した文字列を返す.

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

75. showbyshell(s)
:: shell でコマンド s を実行した標準出力の結果を Risa/Asir で表示

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

76. getbyshell(s)
:: shell でコマンド s を実行した標準出力の結果をファイルとして得る
- 戻り値を Id とおくと，Id ≥ 0 なら出力結果が得られたことを意味し，結果は get_line(Id) で一行ずつ文字列として読み込める．get_byte(Id) で 1 バイトずつ読み込むことも出来る.
- 最終行の後は，getline(Id) は 0 を返す．終了後は getbyshell(Id) または close_file(Id) が（少なくとも再度この函数を呼ぶ前に）必要.
- テンポラリファイルが環境変数 %temp% または %tmp% または DIROUT で定義されたディレクトリに作成される．前回用いられたファイルを消去してから新たにファイルが作られる.

77. makev([ℓ_1,ℓ_2,...]|num=1)
:: $\ell_1, \ell_2 \ldots$ を合わせて一つの変数名を作る
- ℓ_i には，変数，文字列，非負整数などが可能.
- $10, 11, \ldots$ という数字は num=1 を指定しないと，$a, b, \ldots$ という文字になる.

```
[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.make([a,10,"_",3]|num=1);
a10_3
```

```
[5] os_md.my_tex_form(@@);
a_{10,3}
```

78. shortv(p,[v_1,v_2,...]|top=v)
:: p の添字番号つき不定元 $v_1,\ldots$ を一文字の v 以下の変数名に変える
不定元 v_i は一文字の不定元で，それに添え字番号は 0 または 1 から始まって連続している変数名の不定元に対し，v（デフォルトは a）から始まって順に使われていない変数名（z まで）に変更する．

```
[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]
```

79. makenewv(ℓ|var=v,num=n)
:: ℓ に使われていない新しい不定元を生成する
- 不定元は z_0, z_1,...,z_9, z_10,... という順で ℓ に使われていない最初のものを返す．
- 不定元のデフォルトの z_の部分は var=v と指定して v で置き換えられる．
  ただし v は不定元または文字列．
- ℓ の中の関数の引数に含まれる不定元も考慮される．
- num=n で，n 個の新しい不定元のリストが得られる．

```
[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
```

80. isvar(p)
:: p が変数かどうか調べる
p が変数の時 1, そうでないとき 0 を返す

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

81. varargs(p,x|all=t)
:: 式 p に含まれる初等函数の函数子とその変数に現れる不定元のリストを返す
- p は行列やリストでもよい．
- 現れる初等函数の函数子のリストとそれらで使われる変数のリスト．

- 函数の引数に含まれる初等函数についても考慮される.
- all=1 を指定すると含まれる全ての初等函数の函数子と全ての変数を返す.
- all=2 を指定すると含まれる全ての変数を返す.

```
[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]]
```

82. pfargs(p,x|level=t)
:: 式 p に現れる x 変数を含んだ初等函数と引数のリストを返す
- p は行列やリストでもよい.
- 現れる初等函数とその函数子と引数の組のリストをリストにして返す.
- 函数の引数に含まれる初等函数についても考慮される.
- level=t : $t = 1$ は函数の引数は無視，$t = 2$ では，函数の引数のみで，その深さを t で表す（デフォルトは $t = 0$）.

```
[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]]
```

83. isdif(p)
:: p が微分作用素と推測されるときはその変数と微分の組のリストを返し，そうでなければ 0 を返す
p が多項式または有理式であって，d で始まる変数で 2 文字目が小文字のアルファベットとなっているものが p に存在し，そのような変数全てに対して p が多項式になっているとき，微分作用素と推測する.

```
[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
```

84. mysubst(r,[v_1,r_1]|inv=1) または mysubst(r,[[v_1,r_1],...]|inv=1)
:: subst(r,v_1,r_1,...) と同等. r が複雑で r_1 が有理式のときに特に有効.
- r は有理式やそれを成分とするリスト，ベクトル，行列（再帰的）とするが，subst() と異なり，文字列の成分があってもよい（文字列の成分は変換されない）.
- r_j も不定元のとき inv=1 が指定可能で，逆向きの置き換え（v_j と r_j を取り替えた変換）を行う.

85. myswap(p,[x_1,x_2,...,x_n])
:: 有理式またはそのリストなどの不定元の巡回置換 $(x_1, x_2, \ldots, x_n)$
n が 2 のときは，不定元の互換.

```
[0] os_md.myswap(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]
```

86. mulsubst(r,[[$p_{1,0},p_{1,1}$],[$p_{2,0},p_{2,1}$],...]|inv=1)
:: 有理式またはそのリスト，ベクトル，行列 r に複数の代入 $p_{j,0} \mapsto p_{j,1}$ $(j=1,2,\ldots)$ を同時に行う
[[x,y],[y,x]] で x と y を交換.

```
[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]
```

87. fmult(f,m,ℓ,n|$\cdots$)
:: $m_i \mapsto m_{i+1} = f(m_i,\ell[i],n[0],n[1],\ldots|\cdots)$ という変換（$m_0 = m$）の最終結果 $m_{length(l)}$ を返す
- ℓ と n はリスト.
- オプション指定はそのまま渡される.

88. mtransbys(f,m,ℓ|$\cdots$)
:: スカラーに関する変換 $f()$ をリスト，ベクトルまたは行列 m に拡張する
- m の各成分 m_ν を $f(m_\nu,l[0],l[1],\ldots|\cdots)$ と変換する.
- リスト，ベクトルまたは行列はネストしていてもよい（行列のリストなど）.
- 引数はリスト ℓ にしてまとめる.
- オプション指定はそのまま渡される.
- map(f,m,$l[0]$,...) と同じだが，map() はネストやオプション指定が不可.

```
[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]]
```

89. mmulbys(f,m,n,ℓ|$\cdots$)

:: 和が定義された objects の 2 つに対して 1 つの object を与える演算 f を，objects を成分とするベクトルまたは行列 m と n の演算に拡張する

- 引数はリスト ℓ にしてまとめる．m は object または行列.
- m と n が行列のときは以下の行列を返す.
  $\left(\sum_\nu f(m_{i,\nu}, n_{\nu,j}, \ell[0], \ell[1], \ldots)\right)_{i,j}$
- m または n がスカラーのときは，他方（の成分）にそのスカラーを掛けた結果を返す.
- m がベクトルのときは行ベクトル，n がベクトルのときは列ベクトルとみなし，同様の計算をして（上で，i または j のインデックスがないと考える），結果をベクトルで返す.
  ただし m と n が共に（同じサイズの）ベクトルのときは，結果はスカラーで返す.
- オプション指定はそのまま渡される.

90. cmpsimple(p,q|comp=t)
:: 式 p と q の簡単さを比較
p の方が q より簡単な場合負の整数を，逆の場合正の整数を返す.
iand(t,1)!=0 のときは変数の個数を，iand(t,2)!=0 のとき単項式の個数を，iand(t,4)!=0 のときは表示の長さを基準に，最後に P と Q の大小関係を，この順で比較し，差があった時点で簡単とみなす．デフォルトは，$t = 7$.

```
[0] os_md.cmpsimple((x+y)^2,(x+1)^3|comp=1);  /* 変数の個数 */
1
[1] os_md.cmpsimple((x+y)^2,(x+1)^3|comp=2);  /* 単項式の個数 */
-1
[2] os_md.cmpsimple((x+y+z)^2,(sin(x)+cos(x))^2)|comp=4);  /* 表示の長さ */
-4                                                  /* 表示の長さの差 */
[3] os_md.cmpsimple(1,-1);
-1                                                  /* 表示の長さの差 */
[4] os_md.cmpsimple(4,2);
1                                                   /* 大小比較 */
```

91. simplify(p,ℓ,t|var=[$x_1, x_2, \ldots$])
:: $\ell = [\ell_0, \ell_1]$ のときは，p（の各要素毎）に subst(*,ℓ_0,ℓ_1) を調べてより簡単なら置き換える（$t = 1 \sim 7$）．$\ell = [\ell_1]$ で ℓ_1 が多項式のときは，ℓ_1 に一次に含まれる含まれる変数の線形関係式とみて簡単化する．複数調べるときは ℓ をリストや多項式のリストとする.

- iand(t,1)!=0 のときは変数の個数を，iand(t,2)!=0 のとき単項式の個数を，iand(t,4)!=0 のときは表示の長さを基準に，この順で比較し，差があった時点で簡単とみなす.
- この比較で等しいときは，両者を直接比べて小さい方とする.
- 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=[$x_1, x_2, \ldots$] を指定すると，p を $x_1, x_2, \ldots$ の有理式とみて，その分母分子の多項式の係数毎に単純化する.

```
[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
```

92. getel(m,i)

:: m がリスト，ベクトル，行列で i が非負整数なら $m[i]$ を返す
i が $[i_1, i_2]$ というリストで m が行列ならば $m[i_1][i_2]$ を返す
それ以外では m をそのまま返す

93. evalred(r|opt=[[s_1,t_1],[s_2,t_2]...])
:: sin(0), cos(0), exp(0) などを 0, 1, 1 などの整数に置き換える
- 置き換えは，以下のものの値
sin(k@pi), cos(k@pi) で値が $0, \pm 1, \pm\frac{1}{2}$ のいずれかになるもの，tan(0), asin(0), atan(0), sinh(0), cosh(0), tanh(0), exp(0), log(1), pow(1,x), 1^x
- s_j を t_j で置き換える（$j = 1, 2, \ldots$），という規則を付加する．付加する規則が 1 つのみのときは opt=[s,t] としてもよい．

```
[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
```

94. evals(r|del=s,raw=1)
:: 文字列あるいは函数を評価する．
- r が文字列のときは，eval_str(s) を返す．さらに del=s が指定されたときは，文字列 s を区切り記号として区切られた文字列を評価して，順にリストとして返す．ただし，raw=1 が指定してあると，評価せずに文字列のリストに分割して返す．
- r がリスト [r_0,r_1,...] のとき，r_0 が文字列でなければ r_0 を函数とみなして
r_0(r_1,r_2,...) を返す．
r_0 が文字列の時も同様なものを返す．引数 $r_1, \ldots$ の中に文字列があれば，それは eval_str() によって得られる値として変換される（execproc() も似た機能の函数）．

```
[0] os_md.evals("(x+1)^2");
x^2+2*x+1
[1] os_md.evals("(x+y)^2,2^3,10/20"|del=",");
[x^2+2*y*x+y^2,8,1/2]
[2] os_md.evals("(x+y)^2,2^3,10/20"|del=",",raw=1);
[(x+y)^2,2^3,10/20]
[3] os_md.evals([dexp,1]);
2.71828
[4] os_md.evals(["pari","gamma",3+@i]);
(0.96286515302378809804+1.3390971760532574429 8*@i)
```

95. myeval([r,[x_1,f_1,v_1],[x_2,f_2,v_2],...])
:: os_md.myeval(subst([r,[x_2,f_2,v_2],...],x_1,f_1(os_md.myeval(v_1)))) を返す
os_myeval([r])=map(eval(r))

96. mydeval([r,[x_1,f_1,v_1],[x_2,f_2,v_2],...])
:: os_md.mydeval(subst([r,[x_2,f_2,v_2],...],x_1,f_1(os_md.mydeval(v_1)))) を返す
os_md.mydeval([r])=map(deval(r))
- r は有理式成分のベクトルや行列やリストなどでもよい．
- 引数が多項式や有理式 r のときは，eval(r) または deval(r) に置き換えられる．

- 引数がベクトルや行列 r のときは，map(eval,r) または map(deval,r) に置き換えられる．
- $v_1, v_2, \ldots$ は，数でなくてそれぞれ myeval() で値が求まるものでもよい（ネスティング可能で，合成函数に対応する）．
- 函数 $f_1, \ldots$ は，dsin などの函数のほか，引数を数とし，数を返す函数で（ユーザが定義したもので）もよい．より一般に定義されたものが解釈可能であればよい．
- $f_1 = 0$ のときは，恒等写像と解釈される．
- 函数 f_1 の引数が複数あるときは [x_1,f_1,$v_{1,1}$,$v_{1,2}$,...] と指定すると f_1(os_md.myeval(v_1)) は f_1(os_md.myeval($v_{1,1}$),os_md.myeval($v_{1,2}$),...)
- $v_1, v_2, \ldots$ や $v_{1,1}, v_{1,2}, \ldots$ がリストのときは，それらを [v_1], [$v_{1,2}$] のように指定する（mydeval() や myeval() でなくて deval() や eval() で評価される）．
- 函数 f_1 にオプションパラメータを付加して用いるときは，f_1 の代わりに [f_1,opt] のように函数とそのオプションリストの組のリストとして指定する．ただし opt は [[s_1,v_1],[s_2,v_2],...]] の形で，s_j はオプション文字列，v_j はその値である（getopt() で得られる形式）．
- f_2 以下についても同様．
- ```
  F=sin(x)$
  for(I=V=0;I<1000;I++) V+=deval(subst(F,x,I/1000));
  ```
  とすると不定元が 1000 個定義され，以降も含めて Risa/Asir の動作が遅くなるが
  ```
  F=[y,[y,dsin,x]]$
  for(I=V=0;I<1000;I++) V+=os_md.myeval(subst(F,x,I/1000));
  ```
  とすればそのような不定元の生成が起こらない．これはより簡明に
  ```
  F=f2df(sin(x))$for(I=V=0;I<1000;I++) V+=os_md.myfeval(F,I/1000));
  ```
  とすればよい（cf. f2df(), myfeval()）
- 函数がこの引数の形に変換されたものは，xygraph() や xy2graph() でサポートされている．

```
[0] F=os_md.f2df(exp(-x^2-y^2));
[z__,[z__,dexp,-x^2-y^2]]
[1] for(V=I=0;I<1000;I++) for(J=0;J<1000;J++)
    V+=os_md.mydeval(subst(F,x,I/1000,y,J/1000));
[2] V/1000^2;
0.558218
[3] F=[w,[z,0,x+y*@i],[w,os_md.abs,z^2+1]]$
[4] os_md.myeval(subst(F,x,1,y,1));
2.23606797749978969619
[5] def mkC(X,Y){return X+Y*@i;}
[6] G=[w,[z,mkC,x,y],[w,os_md.abs,z^2+1]]$
[7] os_md.myeval(subst(G,x,1,y,1));
2.23606797749978969619
```

97. myval([r,[x_1,f_1,v_1],[x_2,f_2,v_2],...])
:: myeval() と同様な函数であるが，可能な限り正確な値を返す
- 三角関数は，その引数が r*@pi で $4r$ または $6r$ が整数のときに正確な値を返す．
- 指数関数は，その引数が整数または指数または r*@i*@pi で $4r$ または $6r$ が整数のときに正確な値を返す．
- 対数関数は，その引数が@e または 1 のとき正確な値を返す．
- べき函数 x^y において，x が 0 または 1 あるいは y が整数のとき，あるいは $y = \frac{1}{2}$ で x が有理数のときは正確な値を返す．
- 実数の絶対値は正確な値を返す．
- 正確な戻り値には，有理数の他，$\sqrt{-1}$, $\sqrt{m}$（m は正整数）が含まれることがある．
```

```
[0] os_md.myval(sin(@pi));
0
[1] os_md.myval(sin(@pi/6));
1/2
[2] deval(sin(@pi/6));
0.5
[3] eval(sin(@pi/6));
0.49999999999999999
[4] os_md.myval(4^sin(@pi/6));
2
[5] os_md.myval(2^exp(@pi*@i));
1/2
[6] os_md.myval(sin(@pi/3));
1/2*((3)^(1/2))
[7] os_md.myval(2*exp(@pi/3*@i));
(1*@i)*((3)^(1/2))+1
[8] eval(@@);
(1+1.7320508075688772934653050783 4*@i)
[9] os_md.myval((2/3)^sin(@pi/6));
1/3*((6)^(1/2))
[10] os_md.myval(os_md.abs(1-2^(1/2)));
((2)^(1/2))-1
[11] os_md.myval((2^(1/2))^(-2)+(3^(3/2))+sin(@pi/6));
3*((3)^(1/2))+1
```

以上は，次のような正確な値を得ている．

$$\sin\pi = 0, \qquad 4^{\sin\frac{\pi}{6}} = 2, \qquad 2^{\exp\pi i} = \tfrac{1}{2},$$
$$2\exp\tfrac{\pi}{3}i = 1+\sqrt{3}i, \qquad (\tfrac{2}{3})^{\sin\frac{\pi}{6}} = \tfrac{\sqrt{6}}{3}, \qquad \left|1-\sqrt{2}\right| = \sqrt{2}-1.$$

98. `f2df(`f`|opt=`n`)`
:: 函数 f から `sin` などの函数子を除いて `myeval()` や `mydeval()` の引数の**リスト形式函数**に変換
- 函数子 `sin,cos,tan,asin,acos,atan,asin,acos,sinh,cosh,tanh,exp,log,pow` を含んだ函数，および x^y の形のべきの函数を含んだ函数から，それらを除いたリスト形式函数に変換する．
- f は有理式のほか，有理式成分のベクトルや行列でもよい．
- `opt=1,3` : `map(eval,`f`)` に対して変換を行う（`@pi` や `cos(0)` などが含まれている場合の効率化）
  `opt=-1,-3` : `map(deval,`f`)` に対して変換を行う
  `opt=-3,2,3` : f がリストの場合も変換を行う（通常は変換せずに返す）

```
[0] os_md.f2df(exp(-x^2-y^2));
[z__,[z__,os_md.myexp,-x^2-y^2]]
[1] os_md.f2df(x^(y+1));
[z__,[z__,os_md.mypow,x,y+1]]
[2] os_md.f2df(x^0.5);
[z__,[z__,dsqrt,x]]
[3] os_md.f2df(exp(-x)*(sin(x)+cos(y)));
[(z__1+z__2)*z__,[z__,os_md.myexp,-x],[z__1,os_md.mysin,x],[z__2,os_md.mycos,y]]
```

```
[4] os_md.f2df(exp(t*sin(x)+cos(y)));
[z__,[z__,dexp,[z1__*t+z1__1,[z1__,dsin,x],[z1__1,dcos,y]]]]
[5] os_md.f2df(exp(t*sin(x)+cos(y))/(sin(x)+2));
[(z__1)/(z__+2),[z__,os_md.mysin,x],[z__1,os_md.myexp,[z__*t+z1__,[z1__,
os_md.mycos,y]]]]
[6] os_md.f2df(tanh(x));
[z__,[z__,0,(z__1^2-1)/(z__1^2+1)],[z__1,os_md.myexp,x]]
[7] os_md.f2df(exp(x+y^2*@pi));
[z__,[z__,os_md.myexp,x+@pi*y^2]]
[8] os_md.f2df(exp(x+y^2*@pi)|opt=1);
[z__,[z__,os_md.myexp,x+3.14159265358979323829*y^2]]
[9] os_md.f2df(exp(x+y^2*@pi)|opt=-1);
[1*z__,[z__,os_md.myexp,1*x+3.14159*y^2]]
[10] M=mat([cos(x),sin(x)],[-sin(x),cos(x)]);
[ cos(x) sin(x) ]
[ -sin(x) cos(x) ]
[11] F=os_md.f2df(M);
[[ z__1 z__ ]
[ -z__ z__1 ],[z__,os_md.mysin,x],[z__1,os_md.mycos,x]]
[12] os_md.myeval(subst(F,x,0));
[ 1 0 ]
[ 0 1 ]
```

99. `todf(`f`,[`v_1`,...,`v_n`])` `todf([`f`,[`ops`]],[`v_1`,...,`v_n`])`
:: 函数子 f で変数が $v_1,\ldots,v_n$ のリスト形式函数を得る（n は f の引数の数）
- `[`ops`]` は f に渡すオプションリスト（`getopt()` で得られる形式）.
- $n=1$ で v_1 がリストでないときは，`[`v_1`,...,`v_n`]` は単に v_1 としていしてもよい.
- v_j は数値，不定元，有理式，リスト形式函数など `f2df()` で解釈できる函数でよい.
  以下では $|z|$, $[2\exp\frac{x}{2}]$, $\zeta(\frac{1}{2}+\sqrt{-1}x)$, $|\zeta(\frac{1}{2}+\sqrt{-1}x)|$ に対応するリスト形式函数を得ている.

```
[0] os_md.todf(os_md.abs,z);
[z_0,[z_0,os_md.abs,z]]
[1] os_md.abs(z);
[z_0,[z_0,os_md.abs,z]]
[2] os_md.todf(floor,2*exp(x/2));
[z_0,[z_0,floor,[2*z__,[z__,os_md.myexp,1/2*x]]]]
[3] os_md.todf(os_md.zeta,1/2+x*@i)
[z_0,[z_0,os_md.zeta,(1*@i)*x+1/2]]
[4] F=os_md.abs(os_md.zeta(1/2+x*@i));
[z_1,[z_1,os_md.abs,[z_0,[z_0,os_md.zeta,(1*@i)*x+1/2]]]]
[5] os_md.myfeval(F,98.831194218);
0.00000000068100083539303687859
```

100. `df2big(`f`|inv=1)`
:: リスト形式函数 f を倍精度浮動小数点計算から `bigfloat` 計算へ変更する
- `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `exp`, `log` に対応.
- `inv=1` の指定は逆変換を意味する.

```
[0] F=os_md.f2df(sin(x));
[z__,[z__,os_md.mysin,x]]
[1] F=os_md.df2big(F);
[z__,[z__,sin,x]]
[2] F=os_md.df2big(F|inv=1);
[z__,[z__,os_md.mysin,x]]
```

101. compdf(f,x,g)　compdf(f,[x_1,x_2,...],[g_1,g_2,...])
:: リスト形式函数 f の変数 x にリスト形式函数 g を代入したリスト形式函数を返す
- f は多項式や有理函数でもよい．たとえばリスト形式函数 f と g に対して $f\cdot g$ を得るには compdf(x*y,[x,y],[f,g]) とすればよい．
- f や g は f2df() で解釈可能な式でもよい．
- f が文字列のときは，|p|，[p]，abs(p)，floor(p)，rint(p)，zeta(p)，gamma(p)，arg(p)，real(p)，imag(p)，conj(p) が可能．ただし p は f2df() で解釈可能な式．なお (p) を略した場合は (x) を指定したとみなす．
- 変数 x_1，x_2,... にそれぞれリスト形式函数 g_1，g_2,... を代入するときは，$x=[x_1,x_2,...]$，$g=[g_1,g_2,...]$ とすればよい．

```
[0] Sin_x=os_md.f2df(sin(x))$
[1] Exp_x=os_md.f2df(exp(x))$
[2] ExpSin_x=os_md.compdf(Exp_x,x,Sin_x)$
[3] os_md.myfeval(ExpSin_x,0);
1
[4] os_md.compdf("|2*sin(x^2)|",x,x);
[z__,[z__,os_md.abs,[2*z__,[z__,os_md.mysin,x^2]]]]
```

102. cutf(f,x,[[x_1,v_1],[x_2,v_2],...,[x_n,v_n]])　cutf(f,x,[t,[x_1,v_1],...,[x_n,v_n]])
:: 函数 f の変数が特定の範囲のとき函数値の変更を行い，その値またはリスト形式函数を返す
- 第 3 引数のリストの個数 n は 2 以上で，$x_1\le x_2<x_2<\cdots<x_{n-1}\le x_n$ を満たす必要がある．
- $x<x_1$ の場合は v_1 を返す，v_1 は（リスト形式函数）でもよい（myfeval(v_1,x) を返す）．第 3 引数の最初の要素が [] のときは，その項は無視される．
- $x>x_n$ の場合は v_n を返す．第 3 引数の最後の要素が [] のときは，その項は無視される．
- $1<j<n$ で $x=x_j$ のときは v_j を返す．v_j はリスト形式函数でもよい．
- 上記以外の時は $f(x)$ を返す．
- f の変数が x でないときは，それを第 3 引数の最初の要素として指定する．たとえば [t,[x_1,v_1],...] などとする．
- 2 番目の引数が不定元のときは，その不定元を変数とするリスト形式函数にして返す．

```
[0] os_md.cutf(1/x,3,[[-1,0],[0,0],[1,0]]);
0
[1] os_md.cutf(1/x,0.5,[[-1,0],[0,0],[1,0]]);
2
[3] os_md.cutf(1/x,0,[[-1,0],[0,0],[1,0]]);
0
[4] F=os_md.f2df(sin(x)/x)$
[5] os_md.cutf(F,1.5,[[],[0,1],[]]);
0.664997
[6] os_md.cutf(F,0,[[],[0,1],[]]);
```

```
1
[7] os_md.cutf(1,-2,[[0,-1],[0,0],[]]);
-1
[8] os_md.cutf(1,0,[[0,-1],[0,0],[]]);
0
[9] os_md.cutf(1,0.5,[[0,-1],[0,0],[]]);
1
[10] os_md.cutf(x/y,0.5,[y,[-1,0],[0,0],[1,0]]);
2*x
[11] F=os_md.cutf(sin(x)/x,x,[[],[0,1],[]]);   /* x=0 のとき 1 と拡張した函数 */
[z_0,[z_0,os_md.cutf,[(sin(z_1))/(z_1)],x,[[z_1,[],[0,1],[]]]]]
[12] os_md.myfdeval(F,0);
1
[13] os_md.myfdeval(F,1)-dsin(1);
0
```

103. `periodicf(`f`,[`a`,`b`],`x`)`　`periodicf(ltov([`g_1`,`g_2`,...,`g_n`]),`c`,`x`)`
:: x を変数とする函数 $f|_{[a,b)}$ を周期関数に拡張した函数にする
- $x-k(b-a)$ の値 y が $a \le y < b$ となる整数 k を選んで $f(y)$ を返す.
- $[\frac{x}{c}]-kn$ の値が 0 以上 n 未満の数となる整数 k と，$x-(kn+\ell)c$ の値 y が 0 以上 c 未満にある ℓ を選んで $g_{\ell+1}(y)$ を返す.
  g_k は数値，あるいは（リスト形式）函数.
- x が不定元のとき，x を変数とする上のようなリスト形式函数にして返す.

```
[0] os_md.periodicf(x^2,[-1,1],0.2);
0.04
[1] os_md.periodicf(x^2,[-1,1],1.2);
0.64
[2] os_md.periodicf(x^2,[-1,1],2.2);
0.04
[3] F=os_md.periodicf(x^2,[-1,1],x)$
[4] os_md.myfeval(F,2.2);
0.04
[5] os_md.periodicf(ltov([1,2,3,4]),1,2);
3
[6] G=os_md.periodicf(ltov([1,2,3,4]),1,x)$
[7] os_md.myfeval(G,2.2);
3
[7] os_md.myfeval(G,4);
1
[8] G=os_md.periodicf(ltov([x,1-x,-x,x-1]),1,x);
[z_1,[z_1,os_md.periodicf,[[ z_0 -z_0+1 -z_0 z_0-1 ]],1,[[z_0,x]]]]
```

104. `cmpf([`f`,[`a`,`b`]|exp=`c`)`　`cmpf(`x`)`
:: 積分区間をコンパクト閉区間 $[0,1]$ に直した函数にする
- `[`f`,[`a`,`b`]]` によって $\int_a^b f(x)dx = \int_0^1 g(x)dx$ となる函数 $g(x)$ を定義し，それをリスト形式函数

として返す.
新たな定義をするまでは，cmpf(x) は $g(x)$ を返す.

- a ="-infty"は $a=-\infty$ を，b ="infty"は $b=\infty$ を意味し，無限区間の積分を有限区間の積分に変換する．なお，"-infty"，"infty"は任意の文字列でよい.
- exp のオプションを指定しない場合は以下で $c=-1$ とする.
  exp=c と負数 c を指定した場合も含めて
  [a,∞] で a が有限のときは $g(x)=\frac{f(a+\frac{x}{C(1-x)})}{C(1-x)^2}$ $(C=\frac{|c|}{4})$.
  [−∞,b] で b が有限のときは $g(x)=\frac{f(b+1-\frac{1}{Cx})}{Cx^2}$ $(C=\frac{|c|}{4})$.
  [−∞,∞] のときは $g(x)=(\frac{1}{C(1-x)^2}+\frac{1}{Cx^2})f(\frac{1}{C(1-x)}-\frac{1}{Cx})$ $(C=2|c|)$
- exp=c と正数 c を指定すると
  [a,∞] で a が有限のときは $g(x)=\frac{e^{\frac{x}{1-x}}}{C(1-x)^2}f(\frac{e^{\frac{x}{1-x}}-1}{C}+a)$ $(C=\frac{c}{6})$.
  [−∞,b] で b が有限のときは $g(x)=\frac{e^{1-\frac{1}{x}}}{Cx^2}f(\frac{1-e^{\frac{1}{x}}}{C}+b)$ $(C=\frac{c}{6})$.
  [−∞,∞] のときは $g(x)=\left(\frac{e^{\frac{1}{x}}}{Cx^2}+\frac{e^{\frac{1}{1-x}}}{C(1-x)^2}\right)f(\frac{e^{\frac{1}{1-x}}-e^{\frac{1}{x}}}{C})$ $(C=12c)$.
- f はリスト形式函数でもよい.
- cmpf([1/(1+x^2),["",""]]) で得られる関数は

$$g(x)=\texttt{cmpf}(x)=\begin{cases}\frac{2}{2x^2-2x+1} & x\in[0,1],\\ 2 & x\notin[0,1].\end{cases}$$

- areabezier() で使われている（例にあるグラフを参照）.

105. myfeval(f,x) (f,a)　myfeval(f,[x,a])　myfeval(f,[[x_1,a_1],...])
106. myfdeval(f,a)　myfdeval(f,[x,a])　myfdeval(f,[[x_1,a_1],...])
:: myeval() や mydeval() の引数のリスト形式函数 f の変数に a を代入して値を得る
- f の変数がデフォルトの x でなくて v のときは x を [v,x] とする.
- f の複数の変数に代入するときは第 2 引数を [[x,0.5],[y,3]] などのようにする.

107. myf2eval(f,x,y)
108. myf2deval(f,x,y)
:: myeval() や mydeval() の引数のリスト形式 2 変数函数 f に代入して値を得る
109. myf3eval(f,x,y,z)
110. myf3deval(f,x,y,z)
:: myeval() や mydeval() の引数のリスト形式 3 変数函数 f に代入して値を得る
- 1 変数関数 f は変数 x，2 変数関数 f は変数 x, y，3 変数函数 f は変数 x, y, z を用いて表す.
- myfeval(f,x) と myeval(mysubst(f,[x,x])) とは同等である．同様に myf2eval(f,x,y) と myeval(mysubst(f,[[x,x],[y,y]])) とは同等であり，さらに myf3eval(f,x,y,z) と myeval(subst(f,[[x,x],[y,y],[z,z]])) は同等である（eval を deval に置き換えたものも同様）.
- myf2eval(f,a,b) は myfeval(f,[[x,a],[y,b]]) と同等である.

```
[0] F=[[xx,yy],[xx,dcos,x],[yy,dsin,x]]$
[1] os_md.myfdeval(F,0);
[1,0]
[2] os_md.myfdeval(F,@pi/3);
[0.5,0.866025]
```

上は

```
[3] os_md.mydeval(subst(F,x,@pi/3));
[0.5,0.866025]
[4] def afo(X){ return  [dcos(X),dsin(X)];}
```

```
[5] afo(deval(@pi/3));
[0.5,0.866025]
```

などと同様な結果を得る.

111. `execproc(`ℓ`|all=1,var=`k`)`
:: リスト形式手続きの実行
リスト形式手続きとは，函数，引数リスト，オプションリストの 3 つ組のリストを並べたリスト.
すなわち ℓ=[ℓ_1,ℓ_2,$\cdots$] で，ℓ_j=[f_j,v_j,p_j] となっているとき `call(`f_j`,`v_j`|option_list=`p_j`)` を $j = 1, 2, \ldots$ と実行する.
- オプションがない場合は，略して ℓ_j=[f_j,v_j] としてよい.
- 最後の函数の戻り値が返される.
- 不定元 v1, v2,...,vk は特別の意味を持ち，それぞれ直前，2 つ前 ,..., k 個前の函数の実行結果の戻り値で置き換えられる.
- 上において，$k = 2$ がデフォルトであるが，`var=`k によって変更できる.
- `all=1`：これを指定すると，全ての函数の実行結果の戻り値が返される（最後の実行結果の戻り値を先頭とするリスト）.
- `evals()` も似た機能の函数で，`pari()` も扱える.

```
[1] L=[[os_md.binom,[x,2]],[os_md.binom,[v1,2]]]$
[2] os_md.execproc(subst(L,x,6)|all=1);
[105,15]
```

112. `fsum(`f`,[`m`,`n`,`d`]|df=1,subst=1)`　`fsum(`f`,[`x`,`m`,`n`,`d`]|df=1,subst=1)`
:: 一般項が $f(x)$ で与えられる級数の和 $\sum_{k=0}^{[\frac{m-n}{d}]} f(m+kd)$ を返す
- f の変数がデフォルトの x のときは，変数 x の指定は不要.
- d が指定されていないときは，$d = 1$ と解釈される.
- `df=1` が指定されていると，f を `f2df(`f`)` に置き換えてから処理する.
- `subst=1` が指定されていると，単純な代入で処理する（たとえば，sin(0.5) なども実数に変えない）.

```
[1] os_md.fsum(x,[1,10]);
55
[2] os_md.fsum(x^y,[y,0,5]);
x^5+x^4+x^3+x^2+x+1
[3] os_md.fsum(x^y,[y,0,5,2]);
x^4+x^2+1
[4] P=eval(@pi);
3.14159265358979
[5] s_md.fsum(sin(x)/10000,[0,P,P/10000]|df=1)*P;
1.99999998355037
[6] An=2*((x)^(2*n-1))/(2*n-1)$        /* term */
[7] Un=2*x^(2*n+1)/(2*n+1)/(1-x^2)$  /* error */
[8] P4=os_md.fsum(An,[n,1,4,1]|subst=1);
2/7*x^7+2/5*x^5+2/3*x^3+2*x      /* log((1+x)/(1-x)) */
[9] os_md.fctrtos(@@|dviout=1,var=x,rev=1,small=1);
```

$$2x + \frac{2}{3}x^3 + \frac{2}{5}x^5 + \frac{2}{7}x^7$$

```
[10] subst(P4,x,1/3);             /* log 2 */
53056/76545
[11] deval(@@);
```

```
0.693135
[12] deval(log(2));
0.693147
[13] deval(subst(Un,x,1/3,n,4));
1.27013e-005
[14] P5=os_md.fsum(An,[n,1,5,1]|subst=1)$
[15] subst(P5,x,1/3);
4297606/6200145
[17] deval(@@);
0.693146
[18] deval(subst(Un,x,1/3,n,5));
1.15467e-006
```

113. `fint(`f`,`n`,[`t_1`,`t_2`]]|cpx=1,exp=`c`,int=`k`,prec=`v`)` `fint(`f`,`n`,`ℓ`|···)`

:: 複素積分を含む数値積分

- f は披積分函数で，第 3 引数は積分範囲を表す.
- n は分割の数であるが，積分範囲の外でも f が定義されているときは $-n$ を与えると精度が上がることが多い.
- $n=$`0` は，$n=32$ と解釈される.

実変数の積分

- 実変数函数 f のとき，デフォルトの変数は `x` で，n は分割の数，`[`t_1`,`t_2`]` は積分区間を表す.
- 積分区間で，$t_1=$`"-"` は $-\infty$ を，$t_2=$`"+"` は $+\infty$ を表す.
- 変数が `x` でなくて t のとき，積分区間と合わせて第 3 引数に `[`t`,`t_1`,`t_2`]` と指定する.
- 実変数関数 f が複素数値のとき，`cpx=1` を指定する（たとえば `1/(x^2+@i)` のとき）.

複素積分

- 複素積分のとき，函数 f の変数は `x`, `y`, `z` を使うことができる（同時使用可）.
  なお，`x` は複素変数 `z` の実部，`y` は虚部を表す.
- 積分路は，積分路の実部，虚部を表す函数の組と積分区間とのリスト ℓ で表す．函数のパラメータは `t` とする．たとえば
  ```
  [[(cos(t),sin(t)],[0,2*@pi]]
  ```
  は単位円を表す.
- 積分路の曲線を表す函数は微分可能な函数でなければならないが，それらの合併でもよい．各曲線毎にリストにする．たとえば
  ```
  [ [[t,0],[-1,1]], [[(cos(t),sin(t)],[0,2*@pi]] ]
  ```
  は単位円の上半平面の部分の周を表す曲線
- 折線を積分路とする場合は，通過点の座標を指定する．たとえば
  ```
  [[1,-1],[1,1],[1,-1],[-1,-1],[1,-1]]
  ```
- 始点に戻る閉曲線の場合は，最後に `-1` と指定すればよい．上の長方形の周は
  ```
  [[1,-1],[1,1],[1,-1],[-1,-1],-1]
  ```
  としてもよい.

オプション

- デフォルトではベジェ曲線で積分を近似するが
  - `int=1` を指定すると，$|n|$ 分割して台形公式で近似計算する.
  - `int=2` を指定すると，$|n|$ 分割してシンプソンの公式で近似計算する．このとき，n が奇数なら $|n|+1$ 分割する.
- 積分変数を t とすると，無限区間での積分において無限遠で $o(|t|^{-2})$（有理函数のときは $O(|t|^{-2})$）を満たさないと誤差が大きくなるので，`prec=16` または `exp=1` などと指定すると改善されることが多い.

- 函数に滑らかでない点や不連続点がある場合は，prec=16 などと指定すると計算誤差が少なくなる．
- Acc=1 を指定すると pari() による高精度計算を行う．
  ただし ctrl("bigfloat",1) および setprec(*prec*) による精度の設定が必要．また，有理関数でなくて $\sin(x)$ などの初等関数を扱う場合は注意が必要で，不定元が大量に生成される可能性がある（myeval() の項を参照．ord() で分かる）．

以下の [0], [1], [2], [3] はそれぞれ次の積分を表す．

$$\int_{-\infty}^{\infty}\frac{dx}{x^2+1}=\pi,\quad \int_{-\infty}^{\infty}\frac{\sqrt{2}\,dx}{x^2+\sqrt{-1}}=\pi-\pi\sqrt{-1},\quad \int_{|z|=1}\frac{dz}{2z}=\pi\sqrt{-1},\quad \int_{|z|=1}\frac{e^{\frac{1}{z}}}{2}dz=\pi\sqrt{-1}.$$

```
[0] os_md.fint(1/(x^2+1),32,["-","+"]);
3.14159
[1] os_md.fint(dsqrt(2)/(x^2+@i),96,["-","+"]|cpx=1);
(3.1416-3.14158*@i)
[2] os_md.fint(1/(2*z),-96,[[cos(t),sin(t)],[0,2*@pi]]);
(3.14159*@i)
[3] os_md.fint(exp(1/z)/2,-96,[[cos(t),sin(t)],[0,2*@pi]]);
(-4.96894e-016+3.14159*@i)
```

Bézier 曲線を使った数値積分の相対誤差

非積分関数	積分範囲	指定	16 分割	32 分割	96 分割	384 分割	1536 分割
$\frac{1}{x^2+1}$	$-\infty<x<\infty$	$(n,-)$	1.3×10^{-5}	1.3×10^{-7}	8.5×10^{-10}	4.7×10^{-12}	2.1×10^{-14}
$\frac{1}{x^2+\sqrt{-1}}$	$-\infty<x<\infty$	$(n,-)$	2.3×10^{-3}	1.7×10^{-4}	2.6×10^{-6}	1.9×10^{-8}	8.1×10^{-11}
$e^{\frac{1}{z}}$	$\lvert z\rvert=1$	$(-n,-)$	7.6×10^{-5}	4.1×10^{-6}	4.8×10^{-8}	1.9×10^{-10}	7.3×10^{-13}

114. fimag(*f*,*x*|inv=*g*)
:: 複素変数の指数関数を実変数の関数に変換

```
[1] os_md.fimag(exp(r+x*@i));
   (1*@i)*exp(r)*sin(x)+exp(r)*cos(x)
[2] os_md.fimag(2^(r+x*@i));
   ((2)^(r))*cos(log(2)*x)+(1*@i)*((2)^(r))*sin(log(2)*x)
```

115. trig2exp(*f*,*x*|inv=*g*)
:: 三角関数と指数関数の変換と簡単化
- デフォルトでは，三角函数を複素変数の指数函数で置き換え，指数関数の積は一つにまとめる．
- inv=1：複素変数の指数函数の虚部は，三角函数を使って実変数の函数に置き換える．
  三角関数の積は，三角関数の和にまとめられる．
- inv=sin(*y*), inv=cos(*y*), inv=tan(*y*)：*x* が trig2exp( ) の第 2 引数のとき，$\sin x$ または $\cos x$ または $\tan x$ の有理式に可能な限り直す（$y=x$ のとき）．
  より一般に inv=sin($ax+b$), inv=cos($ax+b$) のように，*y* は第 2 変数 *x* の 1 次式でもよい（たとえば，inv=cos(x/2), inv=sin(x+@pi/3) など）．
- 合成函数の変数中の三角関数や指数関数は変更されない．

```
[0] os_md.trig2exp(sin(x),x);
(-1/2*@i)*exp((1*@i)*x)+(1/2*@i)*exp((-1*@i)*x)
[1] os_md.trig2exp(@@,x|inv=1);
sin(x)
[2] os_md.trig2exp(16*sin(x)^5,x|inv=1);
```

```
sin(5*x)+10*sin(x)-5*sin(3*x)
[3] os_md.trig2exp(4*cos(x)^2*exp(x)^2,x);
2*exp(2*x)+exp((2-2*@i)*x)+exp((2+2*@i)*x)
[4] os_md.trig2exp(4*x*sin(x)^2*cos(x),x|inv=1);
(cos(x)-cos(3*x))*x
[5] os_md.trig2exp(sin(5*x),x|inv=sin(x));
16*sin(x)^5-20*sin(x)^3+5sin(x)
[6] os_md.trig2exp(cos(x+y),x|inv=cos(x));
cos(y)*cos(x)-sin(x)*sin(y)
[7] os_md.trig2exp(cos(2*x+y),x|inv=cos(x));
-2*sin(y)*cos(x)*sin(x)+2*cos(y)*cos(x)^2-cos(y)
[8] os_md.trig2exp(cos(2*x+y),x|inv=sin(x));
-2*cos(y)*sin(x)^2-2*sin(y)*cos(x)*sin(x)+cos(y)
[9] os_md.trig2exp(cos(2*x+3*@pi/2),x|inv=cos(x));
2*sin(x)*cos(x)
[10] os_md.trig2exp(2*cos(2*x+@pi/6),x|inv=cos(x));
2*((3)^(1/2))*cos(x)^2-sin(x)*cos(x)-((3)^(1/2))
[11] os_md.trig2exp(cos(x),x|inv=cos(x/3));
4*cos(1/3*x)^3-3*cos(1/3*x)
[12] os_md.trig2exp(2*cos(x),x|inv=cos(x+@pi/3));
cos(x+1/3*@pi)+sin(x+1/3*@pi)*((3)^(1/2))
[13] os_md.trig2exp(tan(3*x),x|inv=tan(x));
(tan(x)^3-3*tan(x))/(3*tan(x)^2-1)
```

上は以下の等式を示している

$$\begin{aligned}\sin x &= -\tfrac{i}{2}\exp(xi) + \tfrac{i}{2}\exp(-xi),\\ 4\cos^2 x\cdot\exp^2 x &= 2\exp 2x + \exp\bigl((2-2i)x\bigr) + \exp\bigl((2+2i)x\bigr).\end{aligned}$$

`inv=1` を指定した上の例は

$$\begin{aligned}16\sin^5 x &= \sin 5x - 5\sin 3x + 10\sin x,\\ 4x\sin^2 x\cdot\cos x &= x(\cos x - \cos 3x).\end{aligned}$$

`inv=sin(x)` または `inv=cos(x)` を指定した上の例は

$$\begin{aligned}\sin 5x &= 16\sin^5 x - 20\sin^3 x + 5\sin x,\\ \cos(x+y) &= \cos x\cdot\cos y - \sin x\cdot\sin y,\\ \cos(2x+y) &= -2\sin y\cdot\cos x\cdot\sin x + 2\cos y\cdot\cos^2 x - \cos y\\ &= -2\cos y\cdot\sin^2 x - 2\sin y\cdot\cos x\cdot\sin x + \cos y,\\ \cos(2x+\tfrac{\pi}{2}) &= -2\sin x\cdot\cos x,\\ 2\cos(2x+\tfrac{\pi}{6}) &= 2\sqrt{3}\cos^2 x - \sin x\cdot\cos x - \sqrt{3}.\end{aligned}$$

`inv=cos(x/3)` を指定すると

$$\cos x = 4\cos^3\tfrac{x}{3} - 3\cos\tfrac{x}{3}.$$

`inv=cos(x+@pi/3)` を指定すると

$$2\cos x = \cos(x+\tfrac{\pi}{3}) + \sqrt{3}\sin(x+\tfrac{\pi}{3}).$$

`inv=tan(x)` を指定すると

$$\tan 3x = \frac{\tan^3 x - 3\tan x}{3\tan^2 x - 1}.$$

116. `isshortneg(`f`)`
:: f と $-f$ を表現する文字列の長さの比較
$-f$ の表示文字列の長さが f の表示文字列の長さより短いかどうか判定する.

117. `fshorter(`f`,`x`)`
:: x の三角関数の簡単化
- 三角関数の有理函数を自動的に簡単化する.
- 現れる三角関数の変数の比は有理数になっている必要がある.

```
[0] os_md.fshorter(8*sin(x)^4-8*sin(x)^2+sin(x)+1,x);
sin(x)+cos(4*x)
[1] os_md.fshorter((sin(x)-3*sin(3*x))/(3*cos(x)+cos(3*x)),x);
tan(x)^3-2*tan(x)
[2] os_md.fshorter((1-cos(2*x))/(1+cos(2*x)),x);
tan(x)^2
```

118. `fzero(`f`,[`x`,`x_1`,`x_2`]|mesh=`m`,dev=`d`,zero=1,trans=1,cont=1)`
:: 実数値函数 f の零点を求める
- 函数 f は `myeval()` の引数の形でもよい.
- x は変数で，区間 $[x_1, x_2]$ での零点を求める.
変数 x がデフォルトの `x` のときはそれを省略して，二番目の引数は `[`x_1`,`x_2`]` でよい.
- 戻り値は，零点の近似値とそこでの f の値の組のリスト
- f が多項式や有理式のときは分子の多項式の零点を調べる．その次数が 2 以下の時は根の公式を，それ以上の次数の時は `polroots()` を使う.
- `mesh=`m：区間を m 等分してその小区間の両端で f の符号が変わるときに，その間の零点をひとつずつ求める．デフォルトは $m = 1024$.
- `dev=`d：小区間の両端の値で函数のグラフを線分で近似したときの零点または両端の中点での函数の値を順に調べて（前者は奇数回目，後者は偶数回目）零点の近似計算をするが，その繰り返しの最大回数．デフォルトは $d = 16$.
- `zero=1`：区間 $[x_1, x_2]$ の両端で f の符号が変わるときに，その間の零点をひとつ求める.
- `trans=1`：`myeval()` で値が計算できる函数であることを示す（設定しなくてもよい）.
- `cont=1`：f が連続関数であることを示す（設定しなくてもよい）.

```
[0] os_md.fzero(3-x^2,[-2.0,2.0]);
[[-1.73205,-4.44089e-016],[1.73205,-4.44089e-016]]
[1] os_md.fzero(exp(x)*sin(x)-cos(x),[0.0,10.0]);
[[0.531391,1.11022e-016],[3.18303,-4.44089e-016],[6.28505,3.28626e-013],
[9.42486,1.23013e-011]]
[2] os_md.fzero(2/x-x,[-1.0,10.0]);
[[1.41421,3.14018e-016]]
```

119. `fmmx(`f`,[`x`,`x_1`,`x_2`]|mesh=`m`,dev=`d`,mmx=1,zero=1,trans=1,cont=1,dif=1)`
:: 実数値函数 f の極値を求める

- 函数 f は myeval() の引数の形でもよい.
- x は変数で，区間 $[x_1, x_2]$ での極値を求める.
  変数 x がデフォルトの x のときはそれを省略して，二番目の引数は [x_1,x_2] でよい.
- 戻り値は，極値をとる x の近似値とそこでの f の値の組のリスト
- mmx=1：極値でなくて最大値と最小値を求める.
  [[$x_{\min}$,$f(x_{\min})$],[$x_{\max}$,$f(x_{\max})$]] が戻り値で，$f(x_{\min})$ が最小値，$f(x_{\max})$ が最大値.
- mesh=m：区間を m 等分してその隣接した 2 つの小区間の隣接点での函数の値が両端での値の間にないときその間の極値をひとつずつ求める．デフォルトは $m = 1024$.
- dev=d：上で調べる隣接区間をそれぞれ 2 等分してできる 3 組の隣接小区間を順に調べて，極値のある隣接小区間を選ぶ，ということの繰り返しの最大回数．デフォルトは $d = 16$.
- dif=1：f が微分できる函数のとき，f' の零点を極値を取る点とみなす
- dif=g：函数 g の零点を極値を取る点とみなす
- zero=1：dif を指定したとき，区間の両端で f'（または g）の符号が変わるときに，その間の f'（または g）の零点のひとつを極値をとる点とする.
- cont=1：dif を指定したとき，f が連続関数であることを示す（設定しなくてもよい）.
- trans=1：myeval() で値が計算できる函数であることを示す（設定しなくてもよい）.

```
[0] os_md.fmmx(sin(x),[0.0,6.0]);
[[0,0],[1.5708,1],[4.71239,-1],[6,-0.279415]]
[1] os_md.fmmx(sin(x),[0.0,6.0]|mmx=1);
[[4.71239,-1],[1.5708,1]]
[2] os_md.fmmx(cos(x),[0.0,6.0]|dif=1);
[[0,1],[3.14159,-1],[6,0.96017]]
```

120. flim(f,v|prec=c,init=t)
:: 変数 x の実数値関数 f の極限値を求める
- 戻り値は極限値．ただし空文字列は極限値なし，あるいは判定できないこと，また"+", "-"はそれぞれ $+\infty$, $-\infty$ を表す.
- $v = a$：$\lim_{x \to a} f(x)$
- v = ["+",a]：$\lim_{x \to a+0} f(x)$
- v = ["-",a]：$\lim_{x \to a-0} f(x)$
- v = "+"：$\lim_{x \to \infty} f(x)$
- v = "-"：$\lim_{x \to -\infty} f(x)$
- prec=c：c は -1 以上 30 以下の数で，c を増やすと極限値の正確性が増す（$c = 0$ がデフォルト）.
- $f(x)$ のとる値の範囲を，v = ["+",a] のときは，$[a + 8^{-k-1}, a + 8^{-k}]$ で，v = "+" のときは $[8^k, 8^{k+1}]$ で，$k = 4, 5, \ldots, 12$ に対して fmmx(|mmx=1,mesh=16,dev=4) で得て，それによって極限の存在を判定している.
- init=t：t は正の実数で，デフォルトと比べて v に t 倍近い値に限って調べる.

```
[0] os_md.flim(sin(x)/x,0);
1
[1] os_md.flim(sin(x)/x,"+");
0
[2] os_md.flim((1+2*x^2)/(x+x^2),"+");
2
[3] os_md.flim((1+x^4)/(1-x),"+");
-
```

121. fcont(f,[x,x_1,x_2]|mesh=m,dev=d,zero=1,trans=1,dif=1) ?
:: 実数値函数 f の不連続点や滑らかでない点を求める
- 函数 f は myeval() の引数の形でもよい.
- x は変数で，区間 $[x_1, x_2]$ での極値を求める.
  変数 x がデフォルトの x のときはそれを省略して，二番目の引数は [x_1,x_2] でよい.
- 戻り値は，特異点を鋏む区間とその 2 点でのギャップの組のリスト

122. fresidue(p,q|cond=[f_1,f_2,...],sum=1)
:: 多項式 q を分母とする有理式 p/q の特異点と留数の組のリスト（z が変数で，p は正則函数）
- $f_1, \ldots$ は z, x, y を変数とする函数で（z=x+yi）これらが全て正となる留数のみを選ぶ
- sum=1：条件を満たす留数の和を得る
- sum=2：条件を満たす留数の和の $2\pi i$ 倍を得る（複素積分の計算に用いる）

```
[0] os_md.fresidue(z^2,(z^2+1)^2);
[[(1*@i),(-1/4*@i)],[(-1*@i),(1/4*@i)]]
[1] os_md.fresidue(16,z^4+4);
[[(-1+1*@i),(1-1*@i)],[(-1-1*@i),(1+1*@i)],[(1+1*@i),(-1-1*@i)],
[(1-1*@i),(-1+1*@i)]]
[2] os_md.fresidue(16,z^4+4|cond=[y]);
[[(-1+1*@i),(1-1*@i)],[(1+1*@i),(-1-1*@i)]]
[3] os_md.fresidue(16,z^4+4|cond=[y],sum=1);
(-2*@i)
[4] os_md.fresidue(16,z^4+4|cond=[y],sum=2);
4*@pi
[5] os_md.fresidue(3*z^4,z^6+1|cond=[y],sum=2);
2*@pi
[6] os_md.fresidue(3*z,z^3+@i|cond=[y],sum=2);
2*@pi
[7] os_md.fresidue(@e^(@i*z*xi),z^2+1|cond=[y],sum=2);
((@e)^(-xi))*@pi
```

[4], [5], [6], [7] （$\xi \geq 0$）は，留数計算により以下の積分計算を得ている.

$$\int_{-\infty}^{\infty} \frac{16}{x^4+4}dx = 4\pi, \quad \int_{-\infty}^{\infty} \frac{3x^4}{x^6+1}dx = 2\pi, \quad \int_{-\infty}^{\infty} \frac{3x}{x^3+\sqrt{-1}}dx = 2\pi, \quad \int_{-\infty}^{\infty} \frac{e^{ix\xi}}{x^2+1}dx = e^{-\xi}\pi.$$

3.2.2 Numbers

123. abs(p)　abs([p,$prec$])
:: 整数または実数または複素数 p の絶対値を返す
- p が複素数のとき [p,$prec$] として $prec$ で精度の桁数を指定できる.
- 不定変数を引数とすると対応するリスト形式函数が得られる

124. sgn(n|val=1)
:: 数 n または置換 n の符号を返す
- n が数の時は，正負に応じて 1, 0, -1 を返す.
- n がリストまたはベクトルの時，小さい順に並んだものとの転倒数が偶のとき 1, 奇のとき -1 を返す.
  val=1 を指定すると，転倒数を返す.

```
[0] os_md.sgn(-1.2);
```

```
-1
[1] os_md.sgn(4/3);
1
[2] os_md.sgn([4,3,2,1]);
1
[3] os_md.sgn([4,3,2,1]|val=1);
6
```

125. calc(p,[s,q])　calc(p,s)
:: 数や有理式に対して演算を施す
前者は s="+", "-", "*", "/", "^", ">", "<", "=", ">=", "<=", "!=", 後者は s= "abs", "neg", "sqr", "inv", "sgn" が有効.
map() や mtransbys() などと組み合わせて使うと便利.

```
[0] os_md.calc(x,["-",y]);
x-y
[1] os_md.calc(3,["^",4]);
81
[2] os_md.calc(x/y,"inv");
(y)/(x)
[3] map(os_md.calc,[3,-3,0,2],"sgn");
[1,-1,0,1]
```

126. isint(p)
:: p が整数かどうか調べる
p が整数の時 1, そうでないとき 0 を返す
127. israt(p)
:: p が有理数かどうか調べる
p が有理数の時 1, そうでないとき 0 を返す
128. iscrat(p)
:: p が数でその実部と虚部が共に有理数かどうか調べる
p の実部と虚部が共に有理数の数の時 1, そうでないとき 0 を返す
129. isalpha(n)
:: 整数 n がアルファベットの文字コードかどうか調べる

```
[0] strtoascii("1")[0];
49
[1] os_md.isalpha(49);
0
[2] os_md.isalpha(strtoascii("a")[0]);
1
```

130. isnum(n)
:: 整数 n が数字 0〜9 の文字コードかどうか調べる
131. isalphanum(n)
:: 整数 n がアルファベットまたは数字の文字コードかどうか調べる
132. isdecimal(s)
:: 文字列 s が整数または小数を表しているかどうか調べる
- s が文字列でないときは 0 を返す.

- 前後の空白文字列は無視して判定する.

```
[0]  os_md.isdecimal("0");
1
[2] os_md.isdecimal(" -2.5 ");
1
[3] os_md.isdecimal(" -2.5/2 ");
0
[4] os_md.isdecimal(-2.5);
0
```

133. nthmodp(a,n,p)
:: $a^n \mod p$ を返す（a は整数で n, p は自然数）

```
[0] os_md.nthmodp(107,10006,10007);
1
[1] os_md.nthmodp(107,1006,1007);
425
```

134. issquaremodp(a,p|power=n)
:: 平方剰余を調べる（ルジャンドルの平方剰余記号 $\left(\frac{a}{p}\right)$，n は平方を一般べきに）
- p は素数，n は正整数.
- $a \equiv 0 \mod p$ のときは 0 を返す．そうでないときは以下の値を返す.
- $x^2 \equiv a \mod p$ が解をもつとき 1, もたないとき -1.
- power=n を指定したときは，$x^n \equiv a \mod p$ の解の存在問題に置き換える.

```
[0] os_md.rootmodp(12,13);
[5,8]
[1] os_md.rootmodp(11,13);
[0]
[2] os_md.rootmodp(5,13|power=3);
[7,8,11]
```

135. rootmodp(a,p|power=n)
:: p を法として a の平方根（一般には n 乗根）を求める
- p を法とした p 未満のべき乗根を小さい順にリストで返す.
- p が奇素数のべきのときは原始根を用いて計算する.

```
[0] os_md.rootmodp(12,13);
[5,8]
[1] os_md.issquaremodp(11,13);
[]
[2] os_md.rootmodp(5,13^2|power=3);
[7,8,154]
[3] 154^3%13^2;
5
[4] os_md.rootmodp(17,32);
[7,9]
```

136. primroot(p|all=1,ind=a)

:: 奇素数またはそのべき p の原始根やそれを底とする指数をもとめる
- p が素数のときは自然数の最小の原始根を返す.
- `all=1`：原始根を全て小さい順にリストで返す.
- `ind=`a：最小の原始根 ζ を底とする a の指数 n を返す（$\zeta^n \equiv a \mod p$）.

```
[0] os_md.primroot(7);
3
[1] os_md.primroot(17|ind=2);
14
[2] os_md.nthmodp(3,14,17);
2
[3] os_md.primroot(7^2|all=1);
[3,5,10,12,17,24,26,33,38,40,45,47]
[4] for(P=3,V=0;;P=pari(nextprime,P+2))
  if((Q=os_md.primroot(P))>V) os_md.mycat([Q,P]);
2 3
3 7
5 23
6 41
7 71
19 191
21 409
23 2161
......
```

137. `rabin(`p`,`q`)`
:: p に対し q を底とするミラー・ラビンの素数判定を行う（素数，擬素数なら 1 を返す）

```
[0] os_md.rabin(10007,2);
1
[1] os_md.rabin(1007,2);
0
```

138. `cfrac(`x`,`n`)`
:: 有理数あるいは実数を連分数展開する（n は項数）
- `pari(cf,)` を参照.
- n が負のときは，分母が $|n|$ 以下の近似分数を返す（近似値から分数に戻す）.
  たとえば，分母が 3 桁以下の分数は，近似実数の誤差が 10^{-6} 以下ならそれから定まる.

```
[0] os_md.cfrac(3.14159265358979,10);
[3,7,15,1,292,1,1,1,2,1,3]
[1] V=1237.0/4123;
0.300024
[2] os_md.cfrac(V,-10000);
1237/4123
```

139. `cfrac2n(`l`|loop=`m`)`
:: （循環）連分数を通常の形に直す
- `loop=m`：m 項目から循環連している連分数を表す.

```
[0] os_md.cfrac2n([2,3]);
7/3
[1] os_md.cfrac2n([2,3]|loop=1);
1/3*((15)^(1/2))+1
[2] eval(@@);
2.29099444873580562818
[3] os_md.cfrac(@@,11);
[2,3,2,3,2,3,2,3,2,3,2,3]
[4] os_md.cfrac2n([1,2,3]|loop=2);
1/3*((15)^(1/2))+2
[5] eval(@@);
1.43649167310370844239
[6] os_md.cfrac(@@,11);
[1,2,3,2,3,2,3,2,3,2,3,2]
```

140. `sqrtrat(`r`)`

:: 有理数（または実部と虚部が有理数の複素数）の平方根を得る

- r が有理数でも実部と虚部が有理数の複素数でもないときは `[]` を返す.
- r が有理数のとき，$\frac{m}{n}$, $\frac{m}{n}\sqrt{-1}$, $\frac{m\sqrt{p}}{n}$, $\frac{m\sqrt{-p}}{n}$ のいずれかを返す．ただし，m, n, p は整数（$p>1$）である.
  $\sqrt{p}$ は `((`p`)^(1/2))` ，$\sqrt{-p}$ は `@i*((`p`)^(1/2))` と表示される.

```
[0] os_md.sqrtrat(4/9);
2/3
[1] os_md.sqrtrat(-4/9);
(2/3*@i)
[2] os_md.sqrtrat(9/8);
3/4*((2)^(1/2))
[3] os_md.sqrtrat(-9/8);
(3/4*@i)*((2)^(1/2))
[4]  os_md.sqrtrat(-1);
(1*@i)
[5] os_md.sqrtrat(os_md.sqrtrat(4*@i-3));
(1+2*@i)
[6] os_md.sqrtrat((4*@i-3)*2);
(1+2*@i)*((2)^(1/2))
[7] os_md.sqrtrat(1+@i);
((1/2*((2)^(1/2))+1/2)^(1/2))+(1*@i)*((1/2*((2)^(1/2))-1/2)^(1/2))
[8] os_md.sqrtrat(-a^2+2*a-1);
(1*@i)*a+(-1*@i)
```

上は以下を示している

$$\sqrt{\frac{4}{9}}=\frac{2}{3},\ \sqrt{-\frac{4}{9}}=\frac{2}{3}i,\ \sqrt{\frac{9}{8}}=\frac{3\sqrt{2}}{4},\ \sqrt{-\frac{9}{8}}=\frac{3\sqrt{2}i}{4},\ \sqrt{-1}=i,\ \sqrt{4i-3}=1+2i,$$

$$\sqrt{2(4i-3)}=\sqrt{2}(1+2i),\ \sqrt{1+i}=\sqrt{\frac{\sqrt{2}}{2}+\frac{1}{2}}+\sqrt{\frac{\sqrt{2}}{2}-\frac{1}{2}}i,\ \sqrt{-a^2+2a-1}=ai-i.$$

141. sqrt2rat(r|mult=1)

:: 平方根や虚数を含んだ分数の有理化

- $p(x)^{n+\frac{1}{2}}$ は，$p(x)^n \cdot p(x)^{\frac{1}{2}}$ の形に直される（n は整数）
- 分子や分母に表れる平方根は高々一つで，共に a^(1/2) の形をしている場合が特に有効.
- a^(1/2) の有理数係数の有理式は, 有理数 r_1, r_2 によって $r_1 + r_2 \cdot$ a^(1/2) の形に標準化される.
- mult=1 を指定すると，函数の引数についてもこの有理化変換を行う.

```
[0] os_md.sqrt2rat((x+@i)/(y+@i));
((y+(-1*@i))*x+(1*@i)*y+1)/(y^2+1)
[1] os_md.sqrt2rat((2+2^(1/2))/(2-2^(1/2)));
2*((2)^(1/2))+3
[2] P=2^(1/2);
((2)^(1/2))
[3] os_md.sqrt2rat(P^4);
4
[4] os_md.sqrt2rat((P+1)^3/(P-1));
4*((2)^(1/2))+13
[5] -((x-1)^(-1/2))*x+((x-1)^(-1/2))+((x-1)^(1/2))
0
```

以上は，以下の等式を示している.

$$\frac{x+i}{y+i} = \frac{xy - xi + yi + 1}{y^2+1}, \quad \frac{2+\sqrt{2}}{2-\sqrt{2}} = 2\sqrt{2}+3, \quad \left(\sqrt{2}\right)^4 = 4, \quad \frac{(\sqrt{2}+1)^3}{\sqrt{2}-1} = 4\sqrt{2}+13.$$

142. sint(r,p|str=t,sqrt=1,zero=0)

:: 実数 r または複素数，リストや行列（ネスト対応）などの成分の実数を小数点以下 p 桁に丸める

- p が負の時は，四捨五入して下 $|p|$ 桁を 0 にする.
- r やその成分が数でない場合はそのまま返す.
- 円周率@pi, ネピア数@e は評価される．複素数は実部と虚部に分けて処理される.
- str=0 ：$p > 0$ のとき，p を有効数字の桁数として丸めた概数を返す.
- str=1 ：文字列で返す.
- str=2 ：1.034*10^(−2) のような文字列で返す.
- str=3 ：10.34 のような TEX の文字列で返す.
- str=4 ：1.034\times10^{−2} のような TEX の文字列で返す.
- sqrt=1：TEX の文字列で返すとき，虚数単位の i を \sqrt{-1} に変更する.
- zero=0：0.··· となるとき，最初の 0 を省いて，.··· とする.

```
[0] V=eval(exp(1));
2.71828182845904523521
[1] os_md.sint(V,4);
2.7183
[2] os_md.sint(1000*V,-2);
2700
[3] os_md.sint([2/3,1/7],5);
[0.66667,0.14286]
[4] os_md.sint([2/3,1/7],5|str=3,zero=0);
[.66667,.14286]
[5] os_md.sint(@pi+@e*@i,5);
```

```
3.14159+2.71828*@i
[6] os_md.sint(@pi+@e*@i,5|str=3);
3.14159+2.71828i
[7] os_md.sint(@pi+@e*@i,5|str=3,sqrt=1);
3.14159+2.71828\sqrt{-1}
[8] os_md.sint(-123.0202,3);
-123.02
[9] os_md.sint(-123.0202,3|str=1);
-123.020
[10] os_md.sint(-123.0202,3|str=2);
-1.230*10^(2)
[11] os_md.sint(-123.0202,3|str=4);
-1.230\times10^{2}
[12] os_md.sint(-12.30202,3|str=4);
-1.230\times10
[13] os_md.sint(-123.0202+@i,3|str=4);
(-1.230+0.010i)\times10^{2}
[14] os_md.sint(-123.0202+@i,3|str=4,sqrt=1);
(-1.230+0.010\sqrt{-1})\times10^{2}
[15] os_md.sint(1234.567,3|str=0);
1230
[16] os_md.sint(1234.567+11.1*@i,3|str=0);
(1230+10*@i)
```

143. `frac2n(`n`|big=1)`
:: 分数を実数になおす（r は複素数，有理式やそのリストや行列などでもよい）
- `big=1` を指定すると `bigfloat` で返す.

```
[0] os_md.frac2n(1/3);
0.333333
[1] os_md.frac2n([1/3,1/4]);
[0.333333,0.25]
[2] os_md.frac2n([1/3,1/4]|big=1);
[0.33333333333333333304,0.25000000000000000000]
[3] os_md.frac2n([1/3,1/4+1/6*@i+3/4*x]);
[0.333333,0.75*x+(0.25+0.166667*@i)]
```

3.2.3 Polynomials and rational functions

144. `radd(`p`,`q`)`
:: 有理式 p と q の和を既約有理式の形で計算する
p と q はベクトルまたは行列でもよい．一方がスカラーで他方がベクトルまたは正方行列でもよい.

```
[0] P=y^3/(x*y+y^2);
(y^3)/(y*x+y^2)
[1] Q=(x^2*y-y^3)/(x*y-y^2);
(y*x^2-y^3)/(y*x-y^2)
```

```
[2] P+Q;
(y^2*x^3+y^3*x^2-2*y^5)/(y^2*x^2-y^4)
[3] os_md.radd(P,Q);
(x^2+2*y*x+2*y^2)/(x+y)
[4] P*Q;
(y^4*x^2-y^6)/(y^2*x^2-y^4)
[5] os_md.rmul(P,Q);
y^2
[6] M = newmat(3,3,[[x,x,x],[x,x,x],[x,x,x]]);
[ x x x ]
[ x x x ]
[ x x x ]
[7] M1 = subst(M,x,1/x);
[ (1)/(x) (1)/(x) (1)/(x) ]
[ (1)/(x) (1)/(x) (1)/(x) ]
[ (1)/(x) (1)/(x) (1)/(x) ]
[8] M2 = subst(M,x,1/(x+y));
[ (1)/(x+y) (1)/(x+y) (1)/(x+y) ]
[ (1)/(x+y) (1)/(x+y) (1)/(x+y) ]
[ (1)/(x+y) (1)/(x+y) (1)/(x+y) ]
[9] M1*M2;
[ (3*x^4+6*y*x^3+3*y^2*x^2)/(x^6+3*y*x^5+3*y^2*x^4+y^3*x^3)
...
[10] os_md.rmul(M1,M2);
[ (3)/(x^2+y*x) (3)/(x^2+y*x) (3)/(x^2+y*x) ]
[ (3)/(x^2+y*x) (3)/(x^2+y*x) (3)/(x^2+y*x) ]
[ (3)/(x^2+y*x) (3)/(x^2+y*x) (3)/(x^2+y*x) ]
[11] N=newmat(2,2,[[a,b],[c,d]]);
[ a b ]
[ c d ]
[12] os_md.radd(x,-N);
[ x-a -b ]
[ -c x-d ]
[13] os_md.radd(x,N[0]);
[ x+a x+b ]
[14] os_md.rmul(x,N);
[ a*x b*x ]
[ c*x d*x ]
[15] V=ltov([u,v]);
[ u v ]
[16] os_md.rmul(N,V);
[ a*u+b*v c*u+d*v ]
[17] os_md.rmul(V,N);
[ a*u+c*v b*u+d*v ]
```

```
[18] os_md.rmul(V,2);
[ 2*u 2*v ]
[19] os_md.rmul(V,V);
u^2+v^2
```

145. rmul(p,q)
:: 有理式（の行列）p と q の積を既約有理式（の行列）の形で計算する
q はベクトルまたは行列でもよい．このとき p は行列やベクトルでもよい（例は radd() の項を参照）.

146. polbyroot([$p_1, p_2, ..., p_n$],x)
:: 多項式を根で与える
戻り値: $\prod_j (x - p_j)$

```
[0] os_md.polbyroot([a,b,c],t);
t^3+(-a-b-c)*t^2+((b+c)*a+c*b)*t-c*b*a
```

147. polbyvalue([[a_1, b_1],...,[a_n, b_n]],x)
:: x の $n-1$ 次多項式を n 個の点 $x = a_i$ での値 b_i で与える
戻り値: $n-1$ 次多項式

```
[0] os_md.polbyvalue([[0,1],[2,3],[3,6]],t);
2/3*t^2-1/3*t+1
```

148. pgen([[x_1, n_1],[x_2, n_2]...],a|sum=n,shift=m,sep=1,num=1)
:: 係数が a_* で x_i が n_i 次，全体で n 次以下の $x_1, \ldots$ の一般多項式を作る
- shift=1：suffix を 1 から始める
- num=1：$10, 11, \ldots$ を $a, b, \ldots$ でなくて数字のままで表す.
- sep=1：複数の suffices のとき _ で区切る

```
[0] os_md.pgen([[x,2],[y,1]],a_);
(a_21*y+a_20)*x^2+(a_11*y+a_10)*x+a_01*y+a_00
[1] os_md.pgen([[x,2],[y,2]],a|sum=3,shift=1);
(a32*y+a31)*x^2+(a23*y^2+a22*y+a21)*x+a13*y^2+a12*y+a11
[2] os_md.pgen([x,6],a);
a6*x^6+a5*x^5+a4*x^4+a3*x^3+a2*x^2+a1*x+a0
[3] os_md.pgen([[x,2],[y,2]],a|num=1,sep=1);
(a2_2*y^2+a2_1*y+a2_0)*x^2+(a1_2*y^2+a1_1*y+a1_0)*x+a0_2*y^2+a0_1*y+a0_0
[4] os_md.my_tex_form(@@);
(a_{2,2}y^2+a_{2,1}y+a_{2,0})x^2+(a_{1,2}y^2+a_{1,1}y+a_{1,0})x
+a_{0,2}y^2+a_{0,1}y+a_{0,0}
```

149. rpdiv(p,q,x)
:: x の多項式の割り算
戻り値 $[r, m, s]$: $r = m * p - s * q$,
mydeg(r,x) < mydeg(q,x), mydeg(m,x) = 0

```
[0] R=os_md.rpdiv(3*x^3+x^2+5,a*x^2+x+1,x);
[(-4*a+3)*x+5*a^2-a+3,a^2,3*a*x+a-3]
[1] R[1]*(3*x^3+x^2+5)-R[2]*(a*x^2+x+1) - R[0];
0
```

150. easierpol(p,x) または easierpol(p,[$x_1, x_2, \ldots$])

:: 有理式係数の x の多項式の係数に x を含まない有理式をかけて，係数の最大公約元が 1 の整数係数の多項式に変換

```
[0] os_md.easierpol(6*r*y/a-3*r*x^2,x);
a*x^2-2*y
```

151. getroot(p,x|mult=1,cpx=1)

:: 多項式の根を有理式または有理数の平方根の範囲で求める

- mult=1 を指定すると，重複度と根の組のリストを返す.
- 係数が有理数の多項式の場合に cpx=1 を指定すると，実部と虚部が有理数の根も合わせて返す.
- 係数が有理数の多項式の場合に cpx=2 を指定すると，実部と虚部が有理数の平方根を使って表せる根も合わせて返す.
- 係数が有理数の多項式の場合に cpx=3 を指定すると，より複雑な根も合わせて返す.
- これら以外の根は，定義方程式で返す.

```
[0] os_md.getroot(os_md.polbyroot([a,b,a,c+d],t),t);
[a,a,b,c+d]
[1] os_md.getroot(os_md.polbyroot([a,b,(b+c)^2],t^2),t);
[t^2-a,t^2-b,-b-c,b+c]
[2] os_md.getroot(os_md.polbyroot([a^4,(b+c)^2],t^2),t|mult=1);
[[1,-a^2],[1,a^2],[1,-b-c],[1,b+c]]
[3] os_md.getroot(2*x^3-x^2-4*x+2,x);
[x^2-2,1/2]
[4] os_md.getroot((2*x^3-x^2-4*x+2)^2,x|mult=1);
[[2,x^2-2],[2,1/2]]
[5] os_md.getroot(2*x^3+x^2+4*x+2,x|cpx=2);
[(-1*@i)*((2)^(1/2)),(1*@i)*((2)^(1/2)),-1/2]
```

152. polroots(p,x|comp=t,err=r,lim=ℓ)

:: 変数 x の 1 変数多項式の根，多変数多項式の共通根（実根・虚根）の（近似）値を返す

- デフォルトでは実根を小さい順に返す.
- comp=1 を指定すると実根と虚根とを重複を込めて返す.
- comp=-1 とすると，実根と虚根を返すが，多項式の係数が有理数のとき有理根は有理数で返す.
- comp=-2 とすると，実根を全て返すが，多項式の係数が有理数のとき有理根は有理数で返す.
- comp=2 を指定すると有理根のみを返す（全ての係数が有理数の時のみ有効）.
  係数に変数以外のパラメータが含まれていてもよい（パラメータについての有理解が得られる）.
- 多変数多項式の場合は，x に変数 n 個のリスト，p に多項式の n 個のリストを指定することによって共通根が得られる.
  - 0 以外の数の戻りは，共通根がみつからなかったことを示す.（非存在と推定される）.
  - 0 の場合は，一意に定まらなかったことを示す（乱数を用いているので，再度試みると求まる可能性は皆無ではない）.
  - 近似計算をしているので，デフォルトでは 2^{-32} 程度の途中計算の誤差を無視して，共通根かどうかを判断している（求めた根の精度とは異なる）.
    この値は，err=r によって変更できる.
- lim=ℓ によって根の範囲を制限できる.
  - 実根または実部に対し，変数 x の範囲を $[a,b]$ に限るには，ℓ の要素に [x,[a,b]] を入れる.
  - 虚部も $[c,d]$ に限るには，[x,[a,b],[c,d]] を入れる.
  - 後者で実部を制限しないときは，[x,[],[c,d]] とする.
  - たとえば x と y の実部を $[0,1]$ に制限するには，lim=[[x,[0,1]],[y,[0,1]]] とする.
  - 一つの変数のみの指定のときは，たとえば lim=[x,[0,1]] または lim=[0,1] としてもよい.

```
[0] os_md.polroots(x^4-x-2,x);
[-1.0000000000000000000,1.35320996419932442942]
[1] os_md.polroots(x^4-x-2,x|comp=1);
[-1.0000000000000000000,1.35320996419932442942,
(-0.17660498209966221474-1.202820819285478805 91*@i),
(-0.17660498209966221474+1.202820819285478805 91*@i)]
[2] os_md.polroots(x^4-x-2,x|comp=2);
[-1]
[3] os_md.polroots(x^4-x-2,x|comp=-1);
[-1,(-0.17660498209966221474-1.202820819285478805 91*@i),
(-0.17660498209966221474+1.202820819285478805 91*@i),
1.35320996419932442942]
[4] os_md.polroots(x^4-x-2,x|comp=-2);
[-1,1.35320996419932442942]
[5] os_md.polroots([x^2+y^2-2,y^2+z^2-2,z^2+x^2-2],[x,y,z]|comp=-1);
[[-1,-1,-1],[1,-1,-1],[-1,1,-1],[1,1,-1],[-1,-1,1],[1,-1,1],[-1,1,1],[1,1,1]]
[6] os_md.polroots([x^2+y^2-2,y^2+z^2-2,z^2+x^2-2],[x,y,z]|comp=2,lim=[x,[0,1]]);
[[1,-1,-1],[1,1,-1],[1,-1,1],[1,1,1]]
[7] os_md.polroots([x^2+y^2-2,y^2+z^2-2,x^2+2*y^2+z^2-2],[x,y,z]|comp=-1);
100
[8] os_md.polroots([x^2+y^2-2,y^2+z^2-2,x^2+2*y^2+z^2-4],[x,y,z]|comp=-1);
0
[9] os_md.polroots([x^3+y^2-2,x^2+y^3-4],[x,y]);
[[-0.68364887206699437088,1.52299734898613986185]]
[10] os_md.polroots([x^3+y^2-2,x^2+y^3-4],[x,y]|comp=1);
[[-0.68364887206699437088,1.52299734898613986185],[(0.55290983195525726542
+0.67794790320397197786*@i),(1.61325322177512775522-0.09613214565586364602 4*@i)],
 ...                              /* 1 個の実根と 8 個の複素根 */
```

上の [9] の計算には，0.03 sec 程度かかった.

153. fctri(p)
:: 実部と虚部が有理数係数の 1 変数多項式を，その範囲で既約分解する
af_noalg() を使うので，load("sp") として関数をロードしておくことが必要.
戻り値は fctr() と同じ形であるが，定数倍は無視される.

```
[1] os_md.fctri(4*x^2+1);
[[1,1],[2*x+(-1*@i),1],[2*x+(1*@i),1],[1,1]]
[2] os_md.fctri(@i*x^3+1);
[[1,1],[x+(1*@i),1],[x^2+(-1*@i)*x-1,1]]
```

154. polinsym(p,[$x_1,\ldots,x_n$],s)
:: $(x_1,\ldots,x_n)$ の対称有理式を基本対称式で表す
k 次の基本対称式を sk と表す（$k=1,\ldots,n$）.

```
[0] os_md.polinsym((a^2+b^2+c^2+d^2)^2,[a,b,c],s);
d^4+(2*s1^2-4*s2)*d^2+s1^4-4*s2*s1^2+4*s2^2
[1] P=os_md.polinsym(x^2+y^2+a/(x+y+b),[x,y],s);
```

```
(a+(s1^2-2*s2)*b+s1^3-2*s2*s1)/(b+s1)
[2] os_md.polinvsym(P,[u,v],s);
(u^3+(v+b)*u^2+v^2*u+v^3+b*v^2+a)/(u+v+b)
```

155. `polinvsym(`p`,[`$x_1,\ldots,x_n$`],`s`)`
:: `polinsym(`p`,[`$x_1,\ldots,x_n$`],`s`)` の逆函数

156. `pol2sft(`p`,`x`|sft=`t`)`
:: shifted power 多項式を与える
- $p=\sum c_n x^n$ ⇒ 戻り値: $\sum c_n x^{\underline{n}}$ $(x^{\underline{n}} := x(x-1)\cdots(x-n+1))$
- `sft=`t を指定すると $x^n \mapsto x(x-t)(x-2t)\cdots(x-(n-1)t)$ というずらしによる多項式の変換

```
[0] os_md.pol2sft(x^3, x);
x^3-3*x^2+2*x
[1] os_md.pol2sft(x^3,x|sft=-1);
x^3+3*x^2+2*x
[2] os_md.pol2sft(x^3,x|sft=e);
x^3-3*e*x^2+2*e^2*x
```

157. `polinsft(`p`,`x`)`
:: shifted power 多項式に直す（`pol2sft()` の逆変換）
$p=\sum c_n x^n = \sum a_n x^{\underline{n}}$ ⇒ 戻り値: $\sum a_n x^n$ $(x^{\underline{n}} := x(x-1)\cdots(x-n+1))$

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

158. `sftpow(`p`,`n`)`

159. `sftpowext(`p`,`n`,`s`)`
:: p の s shifted n power $\prod_{\nu=1}^{n}(p+(\nu-1)s)$ を返す
$\displaystyle\prod_{\nu=1}^{n}\bigl(p+(\nu-1)s\bigr)$ を返す．`sftpow( )` では，$s=1$ と解釈される．

160. `binom(`p`,`n`)`
:: $p(p-1)(p-2)\cdots(p-n+1)/n!$ を返す

161. `expower(`p`,`r`,`n`)`
:: $(1+p)^r$ の展開を p^n まで求める

```
[0] os_md.expower(x,r,3);
(1/6*r^3-1/2*r^2+1/3*r)*x^3+(1/2*r^2-1/2*r)*x^2+r*x+1
[2] os_md.expower(0.5,1/3,10);
1.14471
```

`[2]` は $\sqrt[3]{1.5}$ の近似値を求めている．

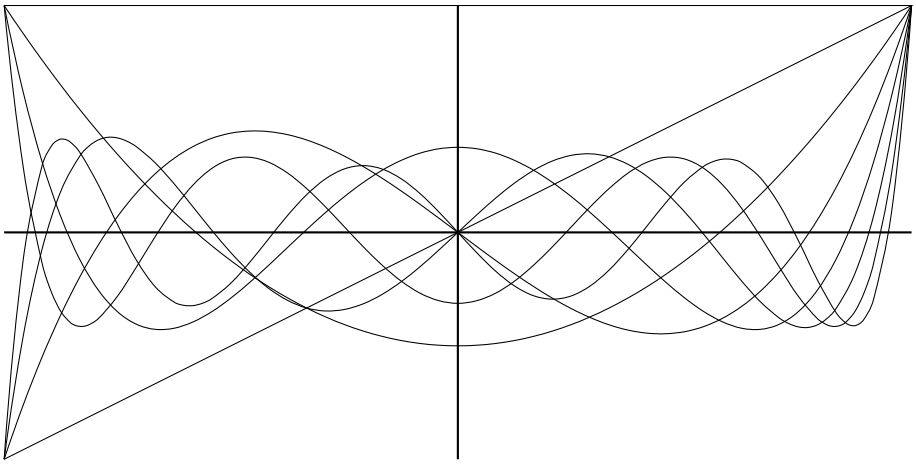
162. `orthpoly(`n`|pol="`*type*`")`　　`orthpoly([`n`,`a`,...]|pol="`*type*`")`
:: 変数 `x` の n 次直交多項式を返す
- デフォルトでは n 次の Legendre 多項式 $P_n(x)$ を返す
- *type* は，`Legendre`, `Gegenbauer`, `Tchebycheff`, `2Tchebycheff`, `Jacobi`, `Laguerre`, `Hermite`, `Selected` のいずれかの文字列が指定可能．ただし，先頭から 2 文字のみで判断される（3 文字目以降は省略可）．
- *type* が，`Gebenbauer` のときの引数は `[`n`,`a`]` で，Gegenbauer 多項式 $C_n^a(x)$ を返す．
- *type* が，`Laguarre` のときは，`[`n`,`a`]` を引数として指定すると Laguarre 多項式 $L_n^{(a)}(x)$ を返し，n を指定すると，$L_n^{(0)}(x)$ を返す．
- *type* が `Jacobi` のときは，`[`n`,`a`,`c`]` を指定すると Jacobi 多項式 $G_n(a,b;x) = F(-n,a+n,c;x)$

を返す.

- *type* が Tchebycheff のときは，戻り値は Tchebycheff 多項式 $T_n(x)$ で，$\cos(n\theta) = T_n(\cos\theta)$.
- *type* が 2Tchebycheff のときは，戻り値 $\bar{U}_n(x)$（この時のみ $n-1$ 次多項式）に $\sqrt{1-x^2}$ を掛けたものが第 2 種の Tchebycheff 関数 $U_n(x)$ となる．このとき $\sin n\theta = \bar{U}_n(\cos\theta)\sin\theta$.
- *type* が Selected のときは，引数 [n,a] に対し，選点直交多項式 $P_{n,a}(x)$ を返す.

```
[0] os_md.orthpoly(5);
63/8*x^5-35/4*x^3+15/8*x
[1] os_md.orthpoly(5|pol="Tc");
16*x^5-20*x^3+5*x
[2] os_md.orthpoly(5|pol="He");
x^5-10*x^3+15*x
[3] os_md.orthpoly(1|pol="La");
-x+1
[4] os_md.orthpoly([1,a]|pol="La");
-x+a+1
[5] T=os_md.xylines([[-1,0],[1,0]]|scale=[6,3],opt="thick")$
[6] T+=os_md.xylines([[0,-1],[0,1]]|scale=[6,3],opt="thick")$
[6] T+=os_md.xylines([[-1,1],[1,1],[-1,-1]]|scale=[6,3])$
[7] for(I=2;I<=7;I++) T+=os_md.xygraph(os_md.orthpoly(I),
    -48,[-1,1],0,[-1,1]|scale=[6,3],prec=6,dviout=-1)$
[8] os_md.xyproc(T|dviout=1)$
```

ルジャンドル多項式 $P_0(x) \sim P_7(x)$ のグラフ



163. schurpoly ([m_1,m_2,...]|var=v)
:: $m_1 \geq m_2 \geq \cdots \geq m_n \geq 0$ のウェイトの Schur 多項式を返す
- デフォルトの変数は x1,x2,...,xn.
- var=y とすると，変数は y1,y2,...,yn
- var=[x,y,...] とすると，変数は $x, y, \ldots$.

```
[0] os_md.schurpoly([2,1,0]);
(x2+x1)*x3^2+(x2^2+2*x1*x2+x1^2)*x3+x1*x2^2+x1^2*x2
[1] os_md.polinsym(@@,[x1,x2,x3],s);
```

```
s2*s1-s3
[2] os_md.schurpoly([2,2,0]|var=[x,y,z]);
(y^2+z*y+z^2)*x^2+(z*y^2+z^2*y)*x+z^2*y^2
[3] os_md.polinsym(@@,[x,y,z],s);
-s3*s1+s2^2
```

164. `seriesMc(`f`,`k`,`v`|evalopt=[[`s_1`,`t_1`],[`s_2`,`t_2`]...])`
:: 函数 f の変数または変数のリスト v に対する k 次の項までの Maclaurin 展開を求める
`evalopt=` により，展開係数に対し `evalred()` を使うときのオプションを設定する

```
[0] os_md.fctrtos(os_md.seriesMc(exp(sin(x)),8,x)|var=x,rev=1);
1+x+1/2*x^2-1/8*x^4-1/15*x^5-1/240*x^6+1/90*x^7+31/5760*x^8
[1] os_md.fctrtos(os_md.seriesMc(log(cos(x)+y),4,[x,y])|var=[x,y],rev=1);
y-1/2*y^2-1/2*x^2+1/3*y^3+1/2*x^2*y-1/4*y^4-1/2*x^2*y^2+1/24*x^4
[2] S=os_md.seriesMc(subst(1/x,x,x+a),4,x))$
[3] T=os_md.fctrtos(S|var=[x,"(x-a)"],rev=1,TeX=1);
\frac{1}{a}-\frac{1}{a^2}(x-a)+\frac{1}{a^3}(x-a)^2-\frac{1}{a^4}(x-a)^3
+\frac{1}{a^5}(x-a)^4
[4] os_md.dviout("$\\frac1x="+T+"+\\cdots$")$
```

$$\frac{1}{x}=\frac{1}{a}-\frac{1}{a^2}(x-a)+\frac{1}{a^3}(x-a)^2-\frac{1}{a^4}(x-a)^3+\frac{1}{a^5}(x-a)^4+\cdots$$

上の `[2],[3]` では $x=a$ での 4 次までの Taylor 展開を求めて，TEX のソースの形で出力している．

165. `seriesHG([`a_1`,`a_2`,...],[`b_1`,`b_2`,...],`p`,`k`)`
`seriesHG([[`a_1`,...],[`b_1`,...],[`c_1`,...]],[[`d_1`,...],[`e_1`,...],[`f_1`,...]],[`x`,`y`],`k`)`
:: 一般超幾何級数 ${}_mF_n(a_1,a_2,\dots,a_m;b_1,b_2,\dots,b_n;p)$ の p^k 次の項まで求める
または 2 変数の級数 $\sum_{0\le i+j\le k}\frac{(a_1)_{i+j}\cdots(b_1)_i\cdots(c_1)_j\cdots}{(d_1)_{i+j}\cdots(e_1)_i\cdots(f_1)_j\cdots}\frac{x^iy^j}{i!j!}$ を返す．
一般超幾何級数は

$$ {}_mF_n(a_1,\dots,a_m;b_1,\dots,b_n;x)=\sum_{k=0}^{\infty}\left(\frac{\prod_{i=1}^m a_i(a_i+1)\cdots(a_i+k-1)}{\prod_{j=1}^n b_j(b_j+1)\cdots(b_j+k-1)}\right)\frac{x^k}{k!}$$

で与えられる．Gauss の超幾何級数は $F(a,b,c;x)=$`seriesHG([`a`,`b`],[`c`],`x`,*)` となる．
微分方程式は `ghg()` を，例は `fctrtos()` の `[18]` の項を参照．
2 変数の級数

$$\sum_{0\le i+j\le k}\frac{\prod_k\prod_\nu^{i+j}(a_k+\nu-1)\cdot\prod_k\prod_\nu^{i}(b_k+\nu-1)\cdot\prod_k\prod_\nu^{j}(c_k+\nu-1)}{\prod_k\prod_\nu^{i+j}(d_k+\nu-1)\cdot\prod_k\prod_\nu^{i}(e_k+\nu-1)\cdot\prod_k\prod_\nu^{j}(f_k+\nu-1)}\frac{x^i}{i!}\frac{y^j}{j!}$$

を得ることが出来る．
Appell の超幾何級数は 2 変数の級数で

$$\begin{aligned}
F_1(a;b,b';c;x,y)&=\texttt{seriesHG([[}a\texttt{],[}b\texttt{],[}b'\texttt{]],[[}c\texttt{],[],[]],[}x,y\texttt{],*)},\\
F_2(a;b,b';c,c';x,y)&=\texttt{seriesHG([[}a\texttt{],[}b\texttt{],[}b'\texttt{]],[[],[}c\texttt{],[}c'\texttt{]],[}x,y\texttt{],*)},\\
F_3(a,a';b,b';c;x,y)&=\texttt{seriesHG([[],[}a,b\texttt{],[}a',b'\texttt{]],[[}c\texttt{],[],[]],[}x,y\texttt{],*)},\\
F_4(a;b;c,c';x,y)&=\texttt{seriesHG([[}a,b\texttt{],[],[]],[[],[}c\texttt{],[}c'\texttt{]],[}x,y\texttt{],*)}.
\end{aligned}$$

166. `fctrtos(`r`|var=`ℓ`,rev=1,dic=1,TeX=`f`,dviout=1,lim=`n`,small=1,pages=1,add=`s`)`
:: 有理式を因数分解した形の文字列に変換する
- `TeX=1`：LATEX のソースを出力する．このとき，多項式でなく有理式の場合は，分子と分母がリストで出力される．

- TeX=2, 3：LaTeX のソースを出力する．有理式のときは通常の \frac{ }{ } の文字列となるが，それが行幅制限を超えた場合は，\Bigl(\Bigr)\bigm/\Bigl(\Bigr) の形の文字列と途中に以下の項に述べる必要な改行が出力される．
- var=x：x の多項式として係数（有理式でもよい）を因数分解して表す．さらに，TeX=2（resp. TeX=3）とすると，1 行の制限を越えないように，指定した変数のベキの後などに \\（resp. \\ &）が入る．より詳しくは，各項が 1 行の文字幅以内ならば，改行は項と項の間のみ．1 行の文字幅を超える項があれば，その前の項の後で改行し，さらに項の途中と（次の項があれば）その項の後に改行を入れる．
- var=[x,s]：変数 x を文字列 s で置き換える．
- var=[$x,y,\ldots$]：$(x,y,\ldots)$ の多項式とみなす．
- var=[[x,s],[y,t],...]：上で，さらに $x,y,\ldots$ をそれぞれ文字列 $s,t,\ldots$ で置き換える．
- rev=1：変数を指定した場合, 通常は次数の高い単項式の順に表示されるが，それを逆順にする．
- dic=1：変数を指定した場合, 単項式のべき指数の辞書式順序で大きいものから表示する．
- var="dif0"：先頭が d で，その次が小文字のアルファベットで表されている変数を，先頭の d を除いた変数の微分作用素とみなす．オプション TeX を指定した場合，微分は ∂ を用いて，∂_{x_1} のように表す．
- var="dif"：上と同じであるが，TeX が指定されていて，微分する変数が 1 個のみで変数が 1 文字で表されているときは，微分を単に ∂ で表す．また微分する変数が 2 個以上あってもその変数名が全て辞書式順序で x0 と x99 の間にあるならば，微分は ∂_0 や ∂_{99} などと表す．
- var="dif1"：上と同様であるが，TeX が指定されると微分は $\frac{d^2}{dx^2}$ または $\frac{\partial^3}{\partial x^2\partial y}$ の様に表される．
- var="dif2"：上と同様であるが，1 変数でも常微分の記号は用いず，偏微分の記号を用いる．
- lim=n：r が多項式のとき，LaTeX のソースへの変換で 1 行が n 文字幅を越えないように（TeX=2 ならば）\\ または （TeX=1,3 ならば）\\ & で改行．ただし，後者でも先頭には & を付加しない．なお，n が 0 以外の 30 以下の正数ならデフォルトの n =TeXLim と解釈される．$n=0$ のときは文字幅制限は考慮しない．
- small=1: LaTeX のソースへの変換で分数をテキストスタイルにする．
  ただし，これによる 1 行の文字数制限の変化は考慮されない．
- dviout=1：dviout を用いて表示する．TeX=3 が指定されたとみなされる．
  なお，画面表示するときは，show() が便利（オプションパラメータがそのまま有効）．
- pages=1：TeX 出力で 1 ページに入らないような長大な式のときは，\begin{align*}... の数式環境を使って改ページを許す．
- add=s：各項の最後に文字列 s を入れる．s は式でもよい．

```
[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)$
```

上の [18] の画面表示は

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

上の [19] の画面表示は，改行位置も含めて次のようになる．

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

var= を指定して微分作用素として扱うと

```
[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)$
```

はそれぞれ以下のような表示となる．

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

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

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

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

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

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

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

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

167. `tohomog(`r`,[`$x_1,x_2,\ldots$`],`y`)`
:: $(x_1,x_2,\ldots)$ の有理式に変数 y を導入して $(y,x_1,x_2,\ldots)$ の斉次式にする

```
[0] os_md.tohomog((x^2+2*x+a*y)/(y+1), [x,y], t);
(x^2+2*t*x+a*t*y)/(y+t)
[1] P = subst(os_md.tohomog(x^2+a*x+b, [x], y), y, x-y);
(a+b+1)*x^2+(-a-2*b)*y*x+b*y^2
[2] subst(P, y, x-1);
x^2+a*x+b
```

x の n 次多項式 $p(x)$ に対し $p(x)=\sum a_jx^j(x-c)^{n-j}$ を満たす多項式 $\sum a_jx^jy^{n-j}$ は，`subst(tohomog(`p`,[`x`],`y`),`y`,`$(x-y)/c$`)` で求められる．

168. `substblock(`p`,`x`,`q`,`y`)`
:: x の多項式 p,q に対し，$y=q$ とおいて p を x の次数が `mydeg(`q`,`x`)` 未満の (x,y) の多項式に直す
p, y は有理式でもよい．y は x を含んではならない．

```
[0] os_md.substblock(x^4+a*x^3+2*x+1,x,x^2+1,y);
(a*y-a+2)*x+y^2-2*y+2
[1] subst(P,y,x^2+1);
x^4+a*x^3+2*x+1
```

169. `invf([`$p_1,\ldots,p_n$`],[`$x_1,\ldots,x_n$`],[`$y_1,\ldots,y_n$`])`
:: $y_j=p_j(x)\quad(j=1,\ldots,n)$ を $x_j=q_j(y)\quad(j=1,\ldots,n)$ と解く（逆函数）．
ただし，`deg(`p_i,x_j`)` は，$i=j$ のとき 1 で $i<j$ のとき 0 でなければならない．
戻り値：`[`$q_1(y_1),\ldots,q_n(y_n)$`]`

170. `mydeg(`p`,`x`|opt=1)`
:: `deg(`p`,`x`)` と同じ．p は行列や配列で係数は有理式でよい．
- 有理式 p が x の多項式でなければ -2 を返す．
- p が行列のとき，オプション `opt=1` により，戻り値 R において $R[0]$ が次数で $R[1]$ はその位置（の一つ）．
- $p=0$ のときは戻り値が -1．

```
[0] os_md.mydeg((a*x+b*x^2)^2/b^2,x);
4
[1] os_md.mydeg(0,x);
-1
```

171. mymindeg(p,x|opt=1)

:: p がスカラーのときは mindeg(p,x) と同じ．係数は有理式でよいが，p が行列などのスカラーでないときは 0 以外の成分の最小次数を返す．

- 有理式 p が x の多項式でなければ -2 を返す．
- p が行列やベクトルのとき，オプション opt=1 により，戻り値 R において $R[0]$ が次数で $R[1]$ はその位置（先頭から見て最初のもの．行列の場合は，1 行目の次が 2 行目という順）．
  0 ではないが次数が 0 の項が複数あれば，その型が最小の位置を返す．
- 戻り値が
  -2：x の多項式でなく，有理式である
  -3：零行列またはベクトルである

```
[0] os_md.mymindeg((a*x+b*x^2)^2/b^2,x);
2
[1] A=newmat(2,2,[[x^2,0],[x,x]]);
[ x^2 0 ]
[ x x ]
[2] os_md.mymindeg(A,x|opt=1);
[1,[1,0]]
[3] A=newmat(2,2,[[x^2,y^2],[x,x]]);
[ x^2 y^2 ]
[ x x ]
[4] os_md.mymindeg(A,x|opt=1);
[0,[0,1]]
[5] os_md.mymindeg(0,x);
0
[6] os_md.mymindeg(x/(x+1)+a,x);
-2
[7] os_md.mymindeg(newmat(2,2),x);
-3
```

172. iscoef(p,f)

:: p の係数の全てが $f(*) \neq 0$ を満たすかどうかチェックする

- 多項式の係数が条件を満たすかどうかのチェックをして，0 または 1 を返す．
- P が有理式のときは，その分母と分子について，リストやベクトルや行列のときは，その成分全てが条件を満たすかどうかをチェックする．

```
[0] os_md.iscoef(1+1/2*x*y,os_md.isint); /* 整数係数？ */
0
[1] os_md.iscoef(1+1/2*x*y,os_md.israt); /* 有理数係数？ */
1
```

173. mycoef(p,n,x)

:: coef(p,n,x) と同じ．n, l は同じ長さのリストでもよい．p は行列や配列で係数は有理式でよい．

- 戻り値は（有理式なら）既約に直される．
- p が x の有理式でなければ 0 を返す．

```
[0] os_md.mycoef((a+b*x)^2/b^2,1,x);
(2*a)/(b)
[1] os_md.mycoef((a+b*x)^2/b^2,2,x);
```

```
1
[2] coef(x+a/b,1,x);
0
[3] os_md.mycoef(x+a/b,1,x);
1
[4] os_md.mycoef(A,0,x);
[ 0 1 ]
[ 0 0 ]
[5] os_md.mycoef(A,1,x);
[ 1 0 ]
[ 0 1 ]
[6] os_md.mycoef((x/a+y+1)^3,2,x);
(3*y+3)/(a^2)
[7]  os_md.mycoef((x/a+y+1)^3,[1,1],[x,y]);
(6)/(a)
```

174. pcoef(p,m,q)
pcoef(p,m,[[x_1,..,x_n],[m_1,...,m_n]])
:: 多項式 p^m を展開したときの単項式 q に対する係数を返す
- p は多項式，m は非負整数.
- q は変数とベキのリストでもよい.
- p の変数が多く，さらに m が大きくて p^m が計算不能な場合にも有効.

```
[0] P=(x+2*y+1)^2+w^2+z$
[1] os_md.pcoef(P,8,x^2*y);
3360
[2] os_md.pcoef(P,8,[[x,y,z],[7,2,0]]);
80640*w^3+887040*w^2+2306304*w+1647360
```

175. pfctr(p,x)
:: x の多項式または有理式 p の因数分解

```
[0] os_md.pfctr((x^2-y^4)^3*(x-y^2/z)*y^2/((x*z-1)*z),x);
[[(y^2)/(z^2),1],[x-y^2,3],[x+y^2,3],[z*x-y^2,1],[z*x-1,-1]]
```

176. cterm(p|var=[x,y,...])
:: 多項式の定数項を返す．変数を指定可能.

```
[0] os_md.cterm((x+y+a+2)^3);
8
[1] os_md.cterm((x+y+a+2)^3|var=[x,y]);
a^3+6*a^2+12*a+8
[2] os_md.cterm((x+y+a+2)^3/(a+2)^4|var=[x,y]);
(1)/(a+2)
```

177. terms(p,[x,y,...]|rev=1,dic=1)
:: 多項式の存在する項の次数とべき指数のリストを返す
次数と各べき指数のリストの辞書式順序で大きい順に並べて返す.
- rev=1：小さい順に並べて返す

- dic=1：最初の次数は無視してべき指数のみで比較する

```
[0] os_md.terms((x^3+y^2+z)^2,[x,y]);
[[6,6,0],[5,3,2],[4,0,4],[3,3,0],[2,0,2],[0,0,0]]
[1] os_md.terms((x^3+y^2+z)^2,[x,y]|dic=1);
[[6,6,0],[5,3,2],[3,3,0],[4,0,4],[2,0,2],[0,0,0]]
[2] os_md.terms((x^2+y^1+z)^2,[x,y]|rev=1);
[[0,0,0],[1,0,1],[2,2,0],[2,0,2],[3,2,1],[4,4,0]]
```

178. polcut (p,n,[$x,y,\ldots$]|top=m)
:: 変数のリスト [$x,y,\ldots$] の多項式 p から次数が（m 以上）n 以下でない項を削除

```
[0] os_md.polcut((x+1)^5,3,x);
10*x^3+10*x^2+5*x+1
[1] os_md.polcut((x+y+1)^5,3,[x,y]|top=3);
10*x^3+30*y*x^2+30*y^2*x+10*y^3
```

179. mydiff(p,x)
:: diff(p,x) と同じ．p は行列や配列で係数は有理式でよい．x もリストでよい．
$x=0$ のときは 0 を返す．

```
[0] os_md.diff(x+a/b,x);
(b^2)/(b^2)
[1] os_md.mydiff(x+a/b,x);
1
[2] os_md.mydiff((x/a+y+1)^3,[x,y]);
(6*x+6*a*y+6*a)/(a^2)
```

180. myediff(p,x)
:: ediff(p,x) と同じ．p は行列や配列で係数は有理式でよい．
$x=0$ のときは 0 を返す．
181. ptol(p,x|opt=0)
:: x の多項式 p の係数のリストを返す
- p は多項式のリスト $[p_1,p_2,\ldots]$ でもよい．
- x が多変数のときは $[x_1,x_2]$ のように変数のリストを渡す．
- opt=0 は，リストから 0 の項が省かれる．

```
[0] os_md.ptol((x-1)*(x+1),x);
[-1,0,1]
[1] os_md.ptol((a*x+b*y+c)^2,y);
[a^2*x^2+2*c*a*x+c^2,2*b*a*x+2*c*b,b^2]
[2] os_md.ptol([a*x+b*y,(a*x+b*y+c*z)^2],[x,y]);
[0,b,a,c^2*z^2,2*c*b*z,b^2,2*c*a*z,2*b*a,a^2]
[3] os_md.ptol([a*x+b*y,(a*x+b*y+c*z)^2],[x,y]|opt=0);
[b,a,c^2*z^2,2*c*b*z,b^2,2*c*a*z,2*b*a,a^2]
```

182. pfrac(p,x|root=2,dviout=1,TeX=1)
:: x の有理式 p を部分分数展開し，分子，分母（多項式とべき）の組のリストを返す
- dviout=1 を指定すると，結果を dviout で表示する．
- TeX=1 を指定すると，上の LaTeX のソースが得られる．

- root=2 を指定すると，$x^4+(2c-r)x^2+c^2=(x^2+\sqrt{r}x+c)(x^2-\sqrt{r}x+c)$ のような分母の因数分解を許す.

```
[0] os_md.pfrac((x^2+1)/((x-1)*(x-2)),x);
[[1,1,1],[-2,x-1,1],[5,x-2,1]] % 1/1^1+(-2)/(x-1)^1+5/(x-2)^1
[1] os_md.pfrac(1/((x-a)^2*(x-b)),x);
[ [(-1)/(a^2-2*b*a+b^2),x-a,1],[(1)/(a-b),x-a,2],
  [(1)/(a^2-2*b*a+b^2),x-b,1] ]
```

pfrac(1/((x-a)^2*(x-b)),x|dviout=1) とすると，以下の表示を得る.

$$-\frac{1}{(a-b)^2(x-a)}+\frac{1}{(a-b)(x-a)^2}+\frac{1}{(a-b)^2(x-b)}$$

pfrac(1/(x^4+1),x|dviout=1,root=2), pfrac(1/(x^6+1),x|dviout=1,root=2), とすると，次の = 以下の表示を得る.

$$\frac{1}{x^4+1}=\frac{x+\sqrt{2}}{2\sqrt{2}(x^2+\sqrt{2}x+1)}-\frac{x-\sqrt{2}}{2\sqrt{2}(x^2-\sqrt{2}x+1)},$$
$$\frac{1}{x^6+1}=\frac{\sqrt{3}x+2}{6(x^2+\sqrt{3}x+1)}-\frac{\sqrt{3}x-2}{6(x^2-\sqrt{3}x+1)}+\frac{1}{3(x^2+1)}.$$

183. lpgcd([$p_1,p_2,\ldots$])
:: 多項式 $p_1,p_2,\ldots$ の共通因子を返す

```
[0] os_md.lpgcd([(x+y)^3*(x-y),(x+y)^2*x^4,(x+y)^2*(x+y+z)]);
x^2+2*y*x+y^2
[1] os_md.lpgcd([0,0]);
0
[2] os_md.lpgcd([15/4,9/7,27/2]);
3
[3] os_md.lpgcd([4/10,8]);
2
```

184. jacobian([$f_1,\ldots,f_n$],[$x_1,\ldots,x_n$]|mat=1)
:: 関数の組 $(f_1,\ldots,f_n)$ の変数 $(x_1,\ldots,x_n)$ についてのヤコビアンまたはヤコビ行列を返す.

```
[0] os_md.jacobian([x+y+z,x*y+y*z+z*x,x*y*z],[x,y,z]);
(y-z)*x^2+(-y^2+z^2)*x+z*y^2-z^2*y
[1] fctr(@@);
[[1,1],[y-z,1],[x-z,1],[x-y,1]]
[2] os_md.jacobian([x+y+z,x*y+y*z+z*x,x*y*z],[x,y,z]|mat=1);
[ 1 1 1 ]
[ y+z x+z x+y ]
[ z*y z*x y*x ]
```

185. hessian([f,[$x_1,\ldots,x_n$]|mat=1)
:: 関数の組 f の変数 $(x_1,\ldots,x_n)$ についてのヘシアンまたはヘッセ行列を返す.

```
[0] os_md.hessian(x^3+x*y+y^3,[x,y]);
36*y*x-1
```

```
[1]  os_md.hessian(x^3+x*y+y^3,[x,y]|mat=1);
[ 6*x 1 ]
[ 1 6*y ]
```

186. `wronskian[`$f_1,\dots,f_n$`],`x`|mat=1)`
:: 関数の組 $(f_1,\dots,f_n)$ の変数 x についてのロンスキアンまたはロンスキー行列を返す.

```
[0] os_md.wronskian([cos(x),sin(x)],x);
cos(x)^2+sin(x)^2
[1] os_md.trig2exp(@@,x|inv=1);
1
[2] os_md.wronskian([cos(x),sin(x)],x|mat=1);
[ cos(x) -sin(x) ]
[ sin(x) cos(x) ]
```

187. `prehombf(`p`,`q`|mem=`±1`)`
:: 概均質ベクトル空間の相対不変式 p の b 函数を得る．q は双対多項式.
- $p=q$ のときは，$q=0$ としてよい.
- $q(\partial_x)p(x)^{s+1}=b(s)p(x)^s$ に対し，`fctr`$(b(s))$ を返す.
- `mem=1` を指定すると，速度よりメモリー使用量を優先する.
- `mem=-1` を指定すると，上と同じだが，計算途中の進行状態も示す.

```
[0] os_md.prehombf(os_md.mydet(os_md.mgen(5,5,x,1)),0);
[[1,1],[s+1,1],[s+2,1],[s+3,1],[s+4,1][s+5,1],]
[1] A=mgen(2,4,x,1);
[ x11 x12 x13 x14 ]
[ x21 x22 x23 x24 ]
[2] os_md.prehombf(os_md.mydet(A*os_md.mtranspose(A)),0);
[[4,1],[s+1,1],[s+2,1],[2*s+3,2]]
[3] os_md.prehombf(os_md.mydet(os_md.mgen(6,6,x,1)),0|mem=-1);
```

188. `intpoly(`p`,`x`|exp=`c`,cos=`c`,sin=`c`)`
:: 変数 x の多項式 p（またはそれと指数関数, 対数関数や三角関数の積）や有理式の原始函数を返す
- p が多項式のときは，定数項のない多項式 $\int_0^x p(x)\,dx$ を返す
- `exp=c` を指定したとき，$\int p(x)e^{cx}dx=q(x)e^{cx}$ となる多項式 $q(x)$ を返す
- `cos=c` を指定したとき，$\int p(x)\cos cx\,dx=q(x)\cos cx+r(x)\sin cx$ となる多項式の組 `[`$q(x),r(x)$`]` を返す
- `sin=c` を指定したとき，$\int p(x)\sin cx\,dx=q(x)\cos cx+r(x)\sin cx$ となる多項式の組 `[`$q(x),r(x)$`]` を返す
- `pow=c` を指定したとき，$\int p(x)x^c\,dx=q(x)x^c$ となる多項式 $q(x)$ を返す
- `log=[`c`,`d`]` を指定したとき，$\int p(x)\log(cx+d)\,dx=q(x)\log(cx+d)+r(x)$ となる多項式の組 `[`$q(x),r(x)$`]` を返す
- `log=[`c`,`d`,`m`]` を指定したとき，$\int p(x)\log^m(cx+d)\,dx=\sum_{j=0}^m q_j(x)\log^{m-j}(cx+d)$ となる多項式のリスト `[`$q_0(x),q_1(x),\dots$`]` を返す (m は正整数).
- c, d は複素数や x を含まない有理式でもよい.
- p が x の有理式の時，その原始関数を返す．ただし分母は x の 2 次以下の多項式の積に因数分解可能され，2 次多項式の判別式の定数倍は完全平方とする.

```
[0] os_md.intpoly(x^2/c+a/b,x);
(b*x^3+3*c*a*x)/(3*c*b)
```

```
[1] os_md.intpoly(x^2,x|exp=1);
x^2-2*x+2
[2] os_md.intpoly(x^2,x|sin=c);
[(-c^2*x^2+2)/(c^3),(2*x)/(c^2)]
[3] os_md.intpoly(4*x,x|log=[1,1,2]);
[2*x^2-2,-2*x^2+4*x+6,x^2-6*x-7]
[4] s_md.intpoly(2/(x^2+a^2)^2,x);
(atan((x)/(a))*x^2+a*x+atan((x)/(a))*a^2)/(a^3*x^2+a^5)
[5] os_md.intpoly(4/(x^2-a^2)^2,x);
((log(x+a)-log(x-a))*x^2-2*a*x+(-log(x+a)+log(x-a))*a^2)/(a^3*x^2-a^5)
[6] os_md.mycoef(R,1,log(x+1));
1
[7] R=os_md.intpoly(1/(x^2-2),x);
1/4*log((x-((2)^(1/2)))/(x+((2)^(1/2))))*((2)^(1/2))
[8] os_md.sqrt2rat(diff(R,x));
(1)/(x^2-2)
```

189. `integrate(`f`,`x`|dumb=`k`,dviout=`p`,log=1,frac=`t`,I=[`a`,`b`])`
:: 関数 f を変数 x について不定積分する
- `I=[`a`,`b`]`：定積分 $\int_a^b f\,dx$ を返す.
- 不定積分が求められなかった場合は，`[]` を返す.
- `dumb=1`：メッセージを表示しない.
- `dumb=-1`：変数変換の過程を示す.
- `dviout=1`：結果の等式を TEX を使って表示する．変形過程を示す `dumb=-1` の指定（以下と等価）が可能.
- `dviout=2`：結果を途中過程も含めて TEX を使って表示する.
  - `log=1` を指定すると，$\log|\cdots|$ でなくて $\log(\cdots)$ とする.
  - `frac=`t：を指定すると，函数で整理して分けての最終表示の仕方を変える.
    `iand(`t`, 1)`：使われる函数が一つの時のみ（函数の変数中の函数は無視）.
    `iand(`t`, 2)`：より表示が短くなるときのみ.
    `iand(`t`, 4)`：もとのまま.
    `iand(`t`, 8)`：函数の有理式のときも（デフォルトは多項式のときのみ）.
- `dviout=-1`：f も含めた結果の TEX のソースを返す．変形過程を示す.
  `dumb=-1` の指定が可能.
- `dviout=-2`：結果の函数の TEX のソースを返す.
  `dumb=-1` を指定すると，変数変換の過程を含めてリストで返す.
  リストの成分は途中結果を表す以下のリスト（その和が途中結果）のリストとなる.
  - `[`x`,`$f(x)$`,...]`：x を積分変数とする不定積分を表す（$f(x),\ldots$ の和）.
  - `[0,`y`,`$p(y)$`,`$q(x)$`]`：変数 y を導入して $p(y)=q(x)$ と置く.
    なお，$y=p(y)$ のときは，$p(y)$ の項を省略可能.
  - `[1:`$f(x)$`,...]`：積分結果を表す（$f(x),\ldots$ の和）.

以下ような関数の不定積分に対応（以下の有理式の積分に関連して，求まらない場合もある）.
- x の有理関数（分母が 2 次以下の多項式の積に因数分解される必要がある）
- x および x の（複数個でもよい）1 次関数の sin, cos, exp を不定元とする多項式
- 多項式と x^c や a^x との積，x と x の 1 次関数の log の多項式，さらにはそれらと前項の式との和
- $\sin x$, $\cos x$, $\tan x$ の有理式．有理数 k を用いた $\sin kx$ や $\cos kx$ などが混ざっていてもよい

$$\int r(\cos x,\sin x)dx=\int R\Big(\frac{1-t^2}{1+t^2},\frac{2t}{1+t^2}\Big)\frac{2}{1+t^2}dt\quad(t=\tan\tfrac{x}{2})$$

- $r(x)\arctan p(x)$ の形

$$\int r(x)\arctan^m p(x)\,dx = \int r(x)dx\cdot\arctan^m p(x) - m\int \frac{p'(x)\int r(x)dx}{1+p(x)^2}\arctan^{m-1} p(x)\,dx$$

- arcsin, arccos, log においても上と同様な部分積分を行う.
- $\exp x$ や e^x の有理式. 有理数 k を用いた $\exp kx$ や e^{kx} などが混ざっていてもよい

$$\int r(e^{kx})dx = \int r(y)\frac{dy}{ky} \qquad (y=e^{kx})$$

- $r\big(x,\sqrt{c^2-(ax-b)^2}\big)$ で，$r(x,y)$ が有理式

$$\int r\big(x,\sqrt{c^2-(ax-b)^2}\big)\,dx = \int r\big(\tfrac{c}{a}\sin t+\tfrac{b}{a},c\cos t\big)\tfrac{c}{a}\cos t\,dt \quad (x=\tfrac{c}{a}\sin t+\tfrac{b}{a})$$

- $r\big(x,\sqrt{(ax-b)^2-c^2}\big)$ で，$r(x,y)$ が有理式

$$\int r\big(x,\sqrt{(ax-b)^2-c^2}\big)\,dx = \int r\Big(\frac{c}{a\cos t}+\frac{b}{a},\frac{c\sin t}{\cos t}\Big)\frac{c\sin t}{a\cos^2 t}dt \quad (x=\frac{c}{a\cos t}+\frac{b}{a})$$

- $r\big(x,\sqrt{(ax-b)^2+c^2}\big)$ で，$r(x,y)$ が有理式

$$\int r\big(x,\sqrt{c^2+(ax-b)^2}\big)\,dx = \int r\Big(\frac{c\sin t}{a\cos t}+\frac{b}{a},\frac{c}{\cos t}\Big)\frac{c}{a\cos^2 t}dt \quad (x=\frac{c\sin t}{a\cos t}+\frac{b}{a})$$

- $r\big(x,(\frac{\alpha x+\beta}{\gamma x+\delta})^{\frac{m}{n}}\big)$ で，$r(x,y)$ が有理式

$$\int r\big(x,(\tfrac{\alpha x+\beta}{\gamma x+\delta})^{\frac{m}{n}}\big)\,dx = \int r\Big(-\frac{\delta t^n-\beta}{\gamma t^n-\alpha},t^m\Big)\Big(-\frac{d}{dt}\frac{\delta t^n-\beta}{\gamma t^n-\alpha}\Big)\,dt \quad (t=(\tfrac{\alpha x+\beta}{\gamma x+\delta})^{\frac{1}{n}})$$

- $r\big(x,\sqrt{\alpha x+\beta},\sqrt{\gamma x+\delta}\big)$ で，$r(x,y,z)$ が有理式

$$\int r\big(x,\sqrt{\alpha x+\beta},\sqrt{\gamma x+\delta}\big)dx = \int r\left(\frac{t^2-\delta}{\gamma},\sqrt{\frac{\alpha t^2-\alpha\delta+\beta\gamma}{\gamma}},t\right)\frac{2t}{\gamma}dt \quad (t=\sqrt{\gamma x+\delta})$$

```
[0] os_md.integrate(x^2/c+a/b,x);
(b*x^3+3*c*a*x)/(3*c*b)
[1] os_md.integrate(x^2*exp(x),x);
exp(x)*x^2-2*exp(x)*x+2*exp(x)
[2] os_md.integrate(x*a^x,x);
(((a)^(x))*log(a)^2*x-((a)^(x))*log(a))/(log(a)^3)
[3] os_md.integrate(x^2*sin(c*x),x);
(-cos(c*x)*c^4*x^2+2*sin(c*x)*c^3*x+2*cos(c*x)*c^2)/(c^5)
[4] os_md.integrate(4*x*log(x+1)^2,x);
(2*log(x+1)^2-2*log(x+1)+1)*x^2+(4*log(x+1)-6)*x-2*log(x+1)^2+6*log(x+1)
[5] os_md.integrate(2/(x^2+a^2)^2,x);
(atan((x)/(a))*x^2+a*x+atan((x)/(a))*a^2)/(a^3*x^2+a^5)
[6] os_md.integrate(4/(x^2-a^2)^2,x);
((log(x+a)-log(x-a))*x^2-2*a*x+(-log(x+a)+log(x-a))*a^2)/(a^3*x^2-a^5)
```

```
[7] os_md.integrate(1/(x^2-2),x);
-1/4*log((x+((2)^(1/2)))/(x-((2)^(1/2))))*((2)^(1/2))
[8] os_md.integrate(1/(x^4+1),x);
(1/8*log(x^2+((2)^(1/2))*x+1)-1/8*log(x^2-((2)^(1/2))*x+1)+1/4
*atan(((2)^(1/2))*x+1)+1/4*atan(((2)^(1/2))*x-1))*((2)^(1/2))
[9] os_md.integrate(1/(x^6+1),x);
(1/12*log(x^2+((3)^(1/2))*x+1)-1/12*log(x^2-((3)^(1/2))*x+1))*((3)^(1/2))
+1/6*atan(2*x+((3)^(1/2)))+1/6*atan(2*x-((3)^(1/2)))+1/3*atan(x)
[10] sqrt2rat(diff(@@,x));
(1)/(x^6+1)
[11] os_md.integrate(2*x*sin(x)*exp(x),x);
(sin(x)-cos(x))*exp(x)*x+cos(x)*exp(x)
[12] os_md.integrate(tan(x),x);
-log(cos(x))
[13] os_md.integrate(2*sin(2*x)*tan(2*x),x);
-sin(2*x)+log((sin(x)+cos(x))/(sin(x)-cos(x)))
[14] os_md.integrate(tan(x)^2,x);
(-cos(x)*x+sin(x))/(cos(x))
[15] os_md.integrate(1/(3*cos(x)+4*sin(x)),x);
1/5*log((3*tan(1/2*x)+1)/(tan(1/2*x)-3))
[16] os_md.integrate(atan(x),x);
atan(x)*x-1/2*log(x^2+1)
[17] os_md.integrate(6*(x^2+1)*atan(1/x),x);
2*atan((1)/(x))*x^3+x^2+6*atan((1)/(x))*x+2*log(x^2+1)
[18]  os_md.integrate((2*b*x-a^2*x^2)^(-1/2),x);
(asin((a^2*x-b*a)/(b)))/(a)
[19] os_md.integrate((a^2-x^2)^(1/2),x);
1/2*((-x^2+a^2)^(1/2))*x+1/2*asin((x)/(a))*a^2
[20] os_md.integrate((x^2-1)^(-1/2),x);
log(x+((x^2-1)^(1/2)))
[21] os_md.integrate((x^2-1)^(1/2),x);
1/2*((x^2-1)^(1/2))*x-1/2*log(x+((x^2-1)^(1/2)))
[22] os_md.integrate((x^2+a^2)^(-1/2),x);
log(x+((x^2+a^2)^(1/2)))
[23] os_md.integrate(2*(x^2+a^2)^(1/2),x);
((x^2+a^2)^(1/2))*x+log(x+((x^2+a^2)^(1/2)))*a^2
[24] os_md.sqrt2rat(diff(@@,x));
2*((x^2+a^2)^(1/2))
[25] os_md.integrate(x^(1/2)/(x+1),x);
2*((x)^(1/2))-2*atan(((x)^(1/2)))
[26] os_md.integrate(x^(1/3)/(x+1),x);
3*((x)^(1/3))-atan(2/3*((3)^(1/2))*((x)^(1/3))-1/3*((3)^(1/2)))*((3)^(1/2))
+1/2*log(((x)^(1/3))^2-((x)^(1/3))+1)-log(((x)^(1/3))+1)
[27] os_md.sqrt2rat(diff(@@,x));
```

```
(((x)^(-2/3))*x)/(x+1)
[28] os_md.integrate(((2-x)/x)^(1/2),x);
(((-x+2)/(x))^(1/2))*x-2*atan((((-x+2)/(x))^(1/2)))
[29] os_md.integrate(1/(@e^x+@e^(-x)),x);
atan(exp(x))
[30] os_md.integrate((sin(x)*x+cos(x))/(cos(x)*x),x);
log|x|-log|cos(x)|
[31] os_md.integrate(asin(x)^2,x);
(asin(x)^2-2)*x+2*asin(x)*((-x^2+1)^(1/2))
[32] os_md.integrate(1/(x*log(x)),x);
log(log(x))
```

上の [0]〜[32] では

$$\int_0^x (\tfrac{1}{c}x^2 + \tfrac{a}{b})\,dx = \tfrac{1}{3c}x^3 + \tfrac{a}{b}x,$$

$$\int x^2 e^x\,dx = (x^2 - 2x + 2)e^x,$$

$$\int x a^x\,dx = \frac{(x\log a - 1)a^x}{(\log a)^2},$$

$$\int x^2 \sin cx\,dx = (-\tfrac{1}{c}x^2 + \tfrac{1}{c^3})\cos cx + \tfrac{2}{c^2}x\sin cx,$$

$$\int 4x\log^2|x+1|\,dx = (2x^2-2)\log^2|x+1| - (2x^2-4x-6)\log|x+1| + x^2 - 6x,$$

$$\int \frac{2\,dx}{(x^2+a^2)^2} = \frac{1}{a^3}\arctan\frac{x}{a} + \frac{x}{a^2(x^2+a^2)},$$

$$\int \frac{4\,dx}{(x^2-a^2)^2} = \frac{1}{a^3}\big(\log|x+a| - \log|x-a|\big) - \frac{2x}{a^2(x^2-a^2)},$$

$$\int \frac{dx}{x^2-2} = \frac{\sqrt{2}}{4}\log\left|\frac{x-\sqrt{2}}{x+\sqrt{2}}\right|,$$

$$\int \frac{dx}{x^4+1} = \frac{\sqrt{2}}{8}\log\left|\frac{x^2+\sqrt{2}x+1}{x^2-\sqrt{2}x+1}\right| + \frac{\sqrt{2}}{4}\big(\arctan(\sqrt{2}x+1) + \arctan(\sqrt{2}x-1)\big),$$

$$\int \frac{dx}{x^6+1} = \tfrac{\sqrt{3}}{12}\Big(\log\left|x^2+\sqrt{3}x+1\right| - \log\left|x^2-\sqrt{3}x+1\right|\Big) + \tfrac{1}{6}\arctan(2x+\sqrt{3}) + \tfrac{1}{6}\arctan(2x-\sqrt{3}) + \tfrac{1}{3}\arctan x,$$

$$\int x\sin x\cdot e^x\,dx = \big(x(\sin x - \cos x) + \cos x\big)e^x,$$

$$\int \tan x\,dx = -\log|\cos x|,$$

$$\int \tan^2 x\,dx = \frac{\sin x}{\cos x} - x,$$

$$\int \frac{dx}{3\cos x + 4\sin x} = \frac{1}{5}\log\left|\frac{3\tan\frac{x}{2}+1}{\tan\frac{x}{2}-3}\right|,$$

$$\int \arctan x\,dx = x\arctan x - \tfrac{1}{2}\log(x^2+1),$$

$$\int 6(x^2+1)\arctan\tfrac{1}{x}\,dx = (2x^3+6x)\arctan\tfrac{1}{x} + x^2 + 2\log(x^2+1),$$

$$\int (2bx-a^2x^2)^{-\frac{1}{2}}\,dx = \frac{1}{a}\arcsin\frac{a^2x-ab}{b},$$
$$\int \sqrt{a^2-x^2}\,dx = \frac{1}{2}\Big(x\sqrt{a^2-x^2}+a^2\arcsin\frac{x}{a}\Big),$$
$$\int \frac{dx}{\sqrt{x^2-1}} = \log\big|x+\sqrt{x^2-1}\big|,$$
$$\int \sqrt{x^2-1}\,dx = \frac{1}{2}\Big(x\sqrt{x^2-1}-\log\big(x+\sqrt{x^2-1}\big)\Big),$$
$$\int \frac{dx}{\sqrt{x^2+a^2}} = \log\big|x+\sqrt{x^2+a^2}\big|,$$
$$\int 2\sqrt{x^2+a^2}\,dx = x\sqrt{x^2+a^2}+a^2\log\big|x+\sqrt{x^2+a^2}\big|,$$
$$\int \frac{\sqrt{x}\,dx}{x+1} = 2\sqrt{x}-2\arctan\sqrt{x},$$
$$\int \frac{\sqrt[3]{x}\,dx}{x+1} = 3\sqrt[3]{x}-\sqrt{3}\arctan(\tfrac{2\sqrt{3}}{3}\sqrt[3]{x}-\tfrac{\sqrt{3}}{3})+\tfrac{1}{2}\log|\sqrt[3]{x^2}-\sqrt[3]{x}+1|-\log|\sqrt[3]{x}+1|,$$
$$\int \sqrt{\frac{2-x}{x}}\,dx = x\sqrt{\frac{2-x}{x}}-2\arctan\sqrt{\frac{2-x}{x}},$$
$$\int \frac{dx}{e^x+e^{-x}} = \arctan\exp(x),$$
$$\int \frac{x\sin x+\cos x}{x\cos x}\,dx = \log|x|-\log|\cos(x)|,$$
$$\int \arcsin^2 x\,dx = x\arcsin^2 x+2\sqrt{1-x^2}\arcsin x-2x,$$
$$\int \frac{dx}{x\log x} = \log(\log x)$$

を得ている.

```
[33] os_md.integrate(1/(x^4+4),x|dviout=1,log=1)$
```

$$\int \frac{dx}{x^4+4} = \frac{1}{8}\arctan(x+1)+\frac{1}{8}\arctan(x-1)+\frac{1}{16}\log\left(\frac{x^2+2x+2}{x^2-2x+2}\right)$$

```
[34] os_md.integrate(((1-cos(x))/(1/2-cos(x)))^(1/2),x|dviout=2)$
```

$$\begin{aligned}
\int \left(\frac{-\cos(x)+1}{-\cos(x)+\frac{1}{2}}\right)^{\frac{1}{2}} dx \qquad & \left(t=\tan(\tfrac{1}{2}x)\right)\\
&= \int \frac{4\sqrt{\frac{1}{3t^2-1}}t}{t^2+1}\,dt\\
&\qquad \left(\sqrt{3}t=\frac{1}{\cos(s)}\right)\\
&= \int \frac{-4}{3\left(\sin(s)\right)^2-4}\,dx_1\\
&\qquad (u=\tan(s))\\
&= \int \frac{4}{u^2+4}\,du
\end{aligned}$$

$$
\begin{aligned}
&= 2\arctan(\tfrac{1}{2}u)\\
&= 2\arctan\left(\frac{\frac{1}{2}\sin(s)}{\cos(s)}\right)\\
&= 2\arctan(\tfrac{1}{2}\sqrt{3t^2-1})\\
&= 2\arctan\left(\tfrac{1}{2}\sqrt{3\left(\tan(\tfrac{1}{2}x)\right)^2-1}\right)
\end{aligned}
$$

```
[35] os_md.integrate(x*sin(x),x|I=[0,@pi]);
@pi
[36] os_md.integrate(x^2*exp(-x),x|I=[0,"infty"]);
2
[37] os_md.integrate(log(x),x|I=[["+",0],1]);
-1
```

上の [35]〜[37] でオプション dviout=1 を指定すると以下が得られる.

$$\int_0^{\pi}\sin(x)x\,dx=\pi,\quad \int_0^{\infty}\exp(-x)x^2\,dx=2,\quad \int_0^{1}\log(x)\,dx=1.$$

190. powsum(n)
:: $1^n+2^n+\cdots+m^n$ の m を x で置き換えた $n+1$ 次多項式を返す

```
[0] os_md.fctrtos(os_md.powsum(3));
1/4*x^2*(x+1)^2
```

191. bernoulli(n)
:: n 次の Bernoulli 多項式 $B_n(x)$ を返す

```
[0] os_md.fctrtos(os_md.bernoulli(3));
1/2*x*(x-1)*(2*x-1)
```

3.2.4 Functions with real/complex variables

以下の数値函数では変数（引数）を**不定変数**にすると対応するリスト形式函数を返す.

192. frac(x)
:: 実数 x の小数部分
pari(frac,x) と同等.

193. erfc(x)　erfc([x,$prec$])
:: 相補誤差函数 $\mathrm{erfc}(x)=\frac{2}{\sqrt{\pi}}\int_x^{\infty}e^{-t^2}dt$
pari(erfc,x) または pari(erfc,x,$prec$) と同等.

194. fouriers([a_0,a_1,...,a_m],[b_1,b_2,...,b_n],z|cpx=1,sum=f,y=t,const=c)
fouriers([f,m],[g,n],z|cpx=1,sum=f,y=t,const=c)
:: 有限 Fourier 級数 $a_0+a_1\cos z+a_2\cos 2z+\cdots+a_m\cos mz+b_1\sin x+b_2\sin 2z+\cdots+b_n\sin nz$
- cpx=1 のときは，$a_0+a_1e^{iz}+a_2e^{2iz}+\cdots+a_me^{imz}+b_1e^{-iz}+b_2e^{-2iz}+\cdots+b_ne^{-niz}$
- [a_0,a_1,...,a_m] の代わりに [f,m] と指定することができ，このときは a_k =myf2eval(f,k,t) と解釈される（$0\le k\le m$）.
  ただし const=c を指定すると，a_0 は c で与えられる．また，f の変数に y が含まれないときは，y=t の指定は無視される.
- [b_1,...,b_n] の代わりに [g,n] と指定することができ，このときは b_k =myf2eval(g,k,t) と解釈される（$1\le k\le n$）.
- sum=1：チェザロ総和法を用いる.

- sum=2：シグマ総和法を用いる．
- sum=s(x,y)：フーリエ級数の有限和

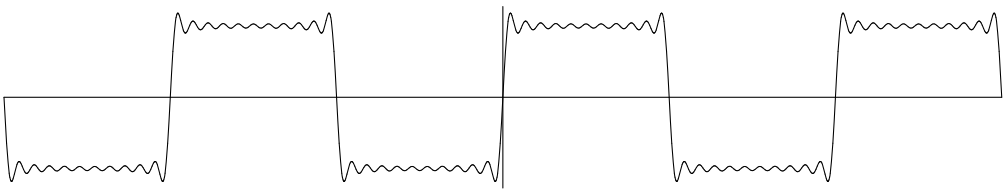
$$a_0+\sum_{k=1}^{n-1} s(\tfrac{k}{n},n)(a_k\cos kx+b_k\sin kx)$$

において 1 とは限らない項 $s(\frac{k}{n},n)$ で調整して総和を取る総和法を行う．チェザロ総和法では，s(x,y) は $1-$x，シグマ総和法では $\frac{\sin \pi x}{\pi x}$ を用いる．
sum=@e^(-t*x^2*y^2)：熱方程式を満たすフーリエ級数の t 秒後の温度分布を示す．

```
[0] for(R=[],I=21;I>0;I--) R=cons((1-(-1)^I)/(2*I),R);
[1] R;
[1,0,1/3,0,1/5,0,1/7,0,1/9,0,1/11,0,1/13,0,1/15,0,1/17,0,1/19,0,1/21]
[2] F=os_md.fouriers([],R,x)$
[3] os_md.xygraph(F,-192,[-3*@pi,3*@pi],0,[-1,1]|scale=[0.7,1.2],ax=[0,0],prec=6,
dviout=1);
```

上の [2] は，次のようにしてもよい（$\sum_{m=0}^{10}\sin(2m+1)x$）．

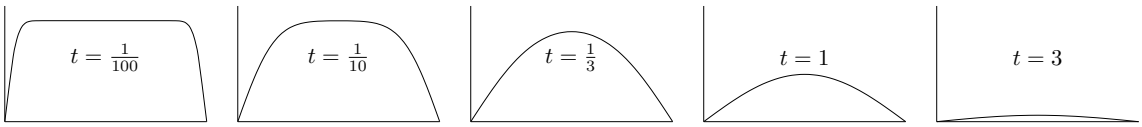
```
[2] F=os_md.fouriers([],[(1-(-1)^x)/2/x,21],x)$
```



fouriers([],[(1-(-1)^x)/x,n],x)：方形波（最大値は $\frac{\pi}{4}$）
fouriers([],[-(-1)^x/x,n],x)：鋸波（最大値は $\frac{\pi}{2}$）
fouriers([(1-(-1)^x)/x,n],[],x)：三角波（最大値と最小値は $\pm\frac{\pi^2}{4}$）
fouriers([sin(x*y)/x/y,n],[],x|const=1/2,y=t)：幅 $2t$ の方形パルス波（$t>0$ は十分小）
fouriers([2*(1-cos(x*y))/x^2/y^2,n],[],x|const=1/2,y=t)：幅 $2t$ の三角パルス波
fouriers([1,n],[],x|const=1/2)：ディリクレ核
fouriers([1,n],[],x|const=1/2,sum=1)：フェイエール核
fouriers([e^(-x^2*y),n],[],x|const=1/2,y=t)：熱方程式の基本解

```
[4] P=os_md.fouriers([],[(1-(-1)^x)/x,41],x|sum=@e^(-x^2*y^2/100))$
[5] os_md.xygraph(P,-24,[0,@pi],0,[0,1.8]|ax=[0,0],prec=6,dviout=1)
```

上の /100 を /10, /3, /1, *3 とすると，順に以下が表示される．これは有限の長さの一定温度（たとえば 1000°）の棒の両端を時刻 0 以降 0° に冷やしたときの温度分布の時間経過とみることができる．



なお，片端のみ冷やして他端が断熱状態の時は，上の図を横に二分したものが温度分布の時間経過を与える．図示するのは，[5] の [0,@pi] を [0,@pi/2] に変更すればよい．

195. myexp(z)
:: 指数関数 $\exp(z)$
196. mysin(z)
:: 三角関数 $\sin(z)$

197. mycos(z)
:: 三角関数 cos(z)
198. mytan(z)
:: 三角関数 tan(z)
199. myasin(z)
:: 逆三角関数 asin(z)
200. myacos(z)
:: 逆三角関数 acos(z)
201. myatan(z)
:: 逆三角関数 atan(z)
202. mylog(z)
:: 対数函数 log(z)
203. mypow(z,w)
:: 巾関数 z^w
mypow(0,w) は w に依らずに 0 を，mypow(z,0) は $z \neq 0$ のとき 1 を返す.
204. myarg(z)
:: 偏角 arg(z)
偏角は, $(-\pi, \pi]$ の範囲が返される.
z は平面の座標 (2 成分のリストかベクトル) でもよい.

```
[0] os_md.myarg(2-@i);
-0.463648
[1] os_md.myarg([2,-1]);
-0.463648
```

指数関数, 対数関数, 三角関数, 逆三角函数, mypow(), myarg() は, dexp(), dsin(), dcos(), dtan(), dacos(), dsqrt(), datan(), real(), imag() を組み合わせて実現されている.
205. sqrt(z)　sqrt([z,$prec$])
:: z の平方根を与える
- pari(sqrt,z) または pari(sqrt,z,$prec$) と同等.
- $prec$ は精度（桁数）を与える.
- $\sqrt{z}$ または $\sqrt{-z}$ が有理数 r となるときは，r あるいは r*@i を返す.

```
[0] os_md.sqrt(2);
1.41421356237309504876
[1] os_md.sqrt([2,30]);
1.41421356237309504880168872420969807856 8
[2] os_md.sqrt(-1);
(1.000000000000000000*@i)
[3] os_md.sqrt(@i);
(0.707106781186547524 38+0.707106781186547524 38*@i)
```

206. arg(z)　arg([z,$prec$])
:: z の偏角
pari(gamma,z) または pari(arg,z,$prec$) と同等.

```
[0] os_md.gamma(1);
0.99999999999999999994
[1] os_md.gamma(100000);
2.82422940796034787421 E456568
```

```
[2] os_md.gamma(@i);
(-0.15494982830181068512-0.49801566811835604271*@i)
```

207. `gamma(`z`)`　`gamma([`z`,`$prec$`])`
:: ガンマ函数 $\Gamma(z)$
`pari(gamma,`z`)` または `pari(gamma,`z`,`$prec$`)` と同等.

```
[0] os_md.gamma(1);
0.99999999999999999994
[1] os_md.gamma(100000);
2.82422940796034787421 E456568
[2] os_md.gamma(@i);
(-0.15494982830181068512-0.49801566811835604271*@i)
```

208. `digamma(`z`)`　`digamma([`z`,`$prec$`])`
:: ディガンマ函数 $\psi(z) = \frac{\Gamma'(z)}{\Gamma(z)}$
`pari(psi,`z`)` または `pari(digamma,`z`,`$prec$`)` と同等.
209. `lngamma(`z`)`　`lngamma([`z`,`$prec$`])`
:: $\log(\Gamma(z))$
`pari(lngamma,`z`)` または `pari(lngamma,`z`,`$prec$`)` と同等.
210. `dilog(`z`)`
:: ダイログ函数 $\mathrm{Li}_2(z) = \sum_{k=1}^{\infty} \frac{z^k}{k^2}$
`pari(dilog,`z`)` と同等.
211. `zeta(`s`)`　`zeta([`s`,`$prec$`])`
:: zeta 函数 $\zeta(s)$
`pari(zeta,`s`)` または `pari(zeta,`s`,`$prec$`)` と同等.
212. `eta(`τ`)`　`eta([`τ`,`$prec$`])`
:: Dedekind の Eta 函数 $\eta(\tau) = e^{\frac{\pi i \tau}{12}} \prod_{m=1}^{\infty}(1 - e^{2\pi i m \tau})$
　τ は虚部が正の複素数.
`pari(eta,`τ`)` または `pari(eta,`τ`,`$prec$`)` と同等.
213. `jell(`τ`)`　`jell([`τ`,`$prec$`])`
:: Elliptic j-invariant $j(\tau)$
　τ は虚部が正の複素数.
`pari(jell,`s`)` または `pari(jell,`τ`,`$prec$`)` と同等.

3.2.5 Lists and vectors

214. `llsize([`$m_1, m_2, \ldots$`])`
:: リスト m_j のリストに対し，成分 m_j の個数と m_j の要素の最大個数を返す

```
[0] os_md.llsize([[1,2],[0,1,2,3],[1]]);
[3,4]
```

215. `findin(`m`,[`$\ell_0, \ell_1, \ldots$`])`
:: m に等しい要素を $\ell_0, \ell_1, \ldots$ から探す
- 最初に見つかった番号を返す．存在しなければ -1 を返す.
- 2 番目の引数はリストでなくてベクトルでも可.

```
[0] os_md.findin("cat",["man","dog","cat","bird"]);
[1] 2
```

216. `countin(`s`,`m`,[`$\ell_0, \ell_1, \ldots$`]|step=`k`)`

:: s 以上 m 以下の $\{\ell_0,\ell_1,\ldots\}$ の要素の個数，間隔 m での個数分布表を返す
最後の引数はリストでなくてベクトルでも可.
step=k を指定すると，s から等間隔 m 毎に区切った $|k|$ 個の区間に入る元の個数のリストを返す ($m>0$ でなければならない).
間隔の区切りと等しいものは，k が正の時は小さい方に入れて数え，負の時は大きい方に数える．ただし $|k|=1$ のときは，データの最大のものを含む区間まで数える.

```
[0] os_md.countin(2,4,[1,2,3,4,5,2,0]);
4
[1] os_md.countin(0,0.5,[1.1,1.2,1.4,1.6,3.1,3.4,3.4,3.7]|step=1);
[0,0,3,1,0,0,3,1]
[2] os_md.countin(0,0.5,[1.1,1.2,1.4,1.6,3.1,3.5,3.5,3.7]|step=-1);
[0,0,3,1,0,0,1,3]
[3] os_md.countin(0,0.5,[1.1,1.2,1.4,1.6,3.1,3.4,3.4,3.7]|step=5);
[0,0,3,1,0]
```

217. delopt(ℓ,s|inv=1)　delopt(ℓ,[s_1,s_2,...]|inv=1)
:: オプションリスト（リストのリスト）からオプション s（複数指定可）を取り除く，または抜き出す
ℓ をリスト $[\ell_1,\ell_2,\ldots]$ とするとき，成分 ℓ_j がリストであって，ℓ_j の最初の成分が s に等しいときに ℓ_j を除いたリストを返す.
複数のオプション s を指定するときは，それらをリストにまとめて指定する.
- 抜き出された結果の並びは，ℓ 内の順序を保つ.
- inv=1：これを指定すると，ℓ_j の最初の成分が s に（s がリストの時は，そのいずれかに）等しいもののみのリストを返す.

```
[0] Opt=[["opt",1],["inv",0],["s",["title",2]]];
[[opt,1],[inv,0],[s,[title,2]]]
[1] os_md.delopt(Opt,"opt");
[[inv,0],[s,[title,2]]]
[2] os_md.delopt(Opt,"inv");
[[opt,1],[s,[title,2]]]
[3] os_md.delopt(Opt,"inv"|inv=1);
[[inv,0]]
[4] L=[[0,1,2],[1,2,3],[3,4,5],[4,5,6],[0,2,4]];
[[0,1,2],[1,2,3],[3,4,5],[4,5,6],[0,2,4]]
[5] [2590] os_md.delopt(L,0|inv=1);
[[0,1,2],[0,2,4]]
[6] os_md.delopt(L,[0,2]|inv=1);
[[0,1,2],[0,2,4]]
[7] os_md.delopt(L,[0,3]|inv=1);
[[0,1,2],[3,4,5],[0,2,4]]
[8] os_md.delopt(L,[0,3]);
[[1,2,3],[4,5,6]]
```

218. mycat([$\ell_1,\ldots,\ell_m$]|delim=s)
219. mycat0([$\ell_1,\ldots,\ell_m$],t|delim=s)
:: $\ell_1,\ldots,\ell_m$ を表示する
- delim で区切り記号（デフォルトは mycat() では空白，mycat0() では無し）を設定できる.

- $t=0$ で非改行，$t=1$ で改行.

```
[0] V=100$
[1] os_md.mycat([V, (V>10)?"is":"is not","larger than 10."])$
100 is larger than 10.
[2] V=5$
[3] os_md.mycat([V, (V>10)?"is":"is not","larger than 10."])$
5 is not larger than 10.
```

220. fcat(f,s|exe=1)
:: ファイル f に s を書き出す
- ファイル f に print(s) を書き出す．ファイル f が存在していれば追記.
  なお，ファイル f を削除するには remove_file(f) とすればよい.
  以下では，DIROUT で設定されたディレクトリのファイルとなる.
- $f=0$：デフォルトのファイル fcat.txt に追記する.
- $f=1$：デフォルトのファイルに上書きする.
- $f=2,\ldots,9$：ファイル fcat$f$.txt に上書きする.
- $f=-1$：デフォルトのファイル名を返す.
- exe=1：出力した後に関連づけされたプログラム（通常はエディター）に渡す.

221. vtozv(v)
:: 有理式のベクトルをスカラー倍して単純化する
- v が有理式のときは，それを長さ 1 のベクトルに自動的に直す
- 各成分は整数係数で共通因子のない多項式になる
- 戻り値：単純化されたベクトルとスカラー倍したスカラーのリスト

```
[0] V=newvect(2,[-x*z/(2*y),-3*y*z^2/(5*x)]);
[ (-z*x)/(2*y) (-3*z^2*y)/(5*x) ]
[1] os_md.vtozv(V);
[[ -5*x^2 -6*z*y^2 ],(10*y*x)/(z)]
```

222. l2p(l,v|size=s)
:: 係数のリスト l を与えて v の多項式を返す．またその逆変換を返す.
- l がリストやベクトルの時は，$\sum_j l[j]v^j$ を返す.
- l は v の多項式の時は，係数のリストを返す．この時は，size=s でリストのサイズを指定できる.

```
[0] P=os_md.l2p([a,b,c],x);
c*x^2+b*x+a
[1] os_md.l2p(P,x);
[a,b,c]
[2] os_md.l2p(P,x|size=5);
[a,b,c,0,0]
[3] os_md.l2p([1,1,1,1,1,1,1,1,1,1],1/2);
1023/512
```

223. mulseries(v_1,v_2)
:: ２つのベクトルをべき級数とみなして，その積のベクトルを返す
戻り値の長さは，両者のベクトルの長さの小さい方とする．(cf. vprod().)

224. pluspower(p,x,r,m)
:: $(1+p)^r$ の x に関するべき級数展開を第 m 項まで求める
- p は定数項のない x の多項式.

- 戻り値は長さ m のベクトルで最初の成分の値は 1.

225. average(ℓ|sint=k)
:: 実数のリストに対し，平均値や標準偏差を求める
戻り値は [**平均値**，**標準偏差**，**個数**，**最小値**，**最大値**].
sint=k : k が自然数の時は，値を小数点以下 k 桁に丸める（sint() の第 2 引数）.

```
[0] os_md.average([11,12,34,53,23,12,24,68,55,57,32,20]);
[33.4167,19.1505,12,11,68]
[1] os_md.average([11,12,34,53,23,12,24,68,55,57,32,20]|sint=1);
[33.4,19.2,12,11,68]
```

226. vprod(v_1,v_2)
227. dvprod(v_1,v_2)
:: 2 つのベクトル（リストでもよい）の内積を返す
- dvprod( ) は, 成分が数値のときのみ用いることができる

```
[0] V1=newvect(3,[1,2,3]);
[ 1 2 3 ]
[1] V2=newvect(3,[a,b,c]);
[ a b c ]
[2] os_md.vprod(V1,V2);
a+2*b+3*c
[3] os_md.mulseries(V1,V2);
[ a 2*a+b 3*a+2*b+c ]
```

228. llbase(v,ℓ)
:: 変数 $\ell[0], \ell[1], \ldots$ の一次方程式のベクトル v の標準変換を行う（例は lsol() の項を参照）
229. lsol(v,l)
:: 変数 $\ell[0], \ell[1], \ldots$ に関する連立一次方程式を解く
変数は，x^3 のような変数の正整数ベキでもよい．ベキが異なるとき異なる変数とみなす.
- ℓ が変数で v が ℓ の一次式でもよい．そのときは式を返す.

```
[0] V = newvect(3,[x+2*y-2,3*x+4*y-1,x+y-k]);
[ x+2*y-2 3*x+4*y-1 x+y-k ]
[1] os_md.llbase(V,[x,y]);
[ -x-3 -2*y+5 2*k+1 ]
[2] os_md.lsol(V,[x,y]);
[ [x,-3] [y,5/2] 2*k+1 ]
[3] V = newvect(2, [a*x+b*y-e, c*x+d*y-d]);
[ a*x+b*y-e c*x+d*y-d ]
[4] os_md.lsol(V,[x,y]);
[ [x,(-d*b+e*d)/(d*a-c*b)] [y,(d*a-e*c)/(d*a-c*b)] ]
[5] VV = subst(V,y,x^2);
[ b*x^2+a*x-e d*x^2+c*x-f ]
[6] os_md.lsol(VV,[x^2,x]);
[ [x^2,(d*a-e*c)/(d*a-c*b)] [x,(-d*b+e*d)/(d*a-c*b)] ]
[7] os_md.lsol(a*x+b*y+c,x);
(-b*y-c)/(a)
```

230. `lnsol(`v`,`l`)`
:: 変数 $\ell[0],\ell[1],\ldots$ に関する連立一次方程式の有理数解を求める
変数は，x^3 のような変数の正整数ベキでもよい．ベキが異なるとき異なる変数とみなす．なお v は多項式のリストとする．

```
[0] P=(a-1)*x+(a+1)*y-2*a+3;
(a-1)*x+(a+1)*y-2*a+3
[1] os_md.lnsol([P],[x,y]);
[ [x,5/2] [y,-1/2] ]
```

231. `lchange(`ℓ`,`k`,`v`|flat=1)`
::（多重）リスト ℓ の k で指定した位置の成分を v に置き換える
- k はリストで，重複の深さの成分がある．k が数のときは，$[k]$ とみなす．なお，0 は最初の成分．
- `flat=1` を指定したとき，$k=[k_1,k_2,\ldots]$, $v=[v_1,v_2,\ldots]$ ならば，$\ell[k_\nu]$ を v_ν （$\nu=1,2,\ldots$）で置き換えたリストを返す．

```
[0] L=[[1,2,3],[4,5,6],[7,8]]$
[1] os_md.lchange(L,[1,0],0);
[[1,2,3],[0,5,6],[7,8]]
[2] os_md.lchange(L,[2],[7,8,9]);
[[1,2,3],[4,5,6],[7,8,9]]
[3] os_md.lchange(L,[1,0],[4,5]|flat=1);
[5,4,[7,8]]
```

232. `lmax([`$m_1,m_2,\ldots$`])`
:: $m_1,m_2,\ldots$ の中の最大のものを返す
233. `lmin([`$m_1,m_2,\ldots$`])`
:: $m_1,m_2,\ldots$ の中の最小のものを返す
- 引数はリストでなく，ベクトルや行列でもよい

```
[0] os_md.lmax([1,5,4,2,3]);
5
[1] os_md.lmin([1,5,4,2,3]);
2
[3] A=mat([1,3],[5,2]);
[ 1 3 ]
[ 5 2 ]
[4] os_md.lmax(A);
5
[5] os_md.lmin(A[1]);
2
```

234. `lgcd([`$m_1,m_2,\ldots$`]|poly=1])`
:: $m_1,m_2,\ldots$ の最大公約数（元）を返す
235. `llcm([`$m_1,m_2,\ldots$`]|poly=1])`
:: $m_1,m_2,\ldots$ の最小公倍数（元）を返す
- `poly=1` を指定すると，要素を多項式とみなす．
- 引数はリストでなく，ベクトルや行列でもよい．
- `llcm()` において，成分が 0 の元は無視される（`ilcm()` と異なる）．

```
[0] os_md.lgcd([-6,9,-24]);
3
[1] os_md.llcm([-6,9,-24]);
72
[2] os_md.lgcd([(x+y)^2,x^2-y^2]|poly=1);
x+y
[3] os_md.llcm([(x+y)^2,x^2-y^2]|poly=1);
x^3+y*x^2-y^2*x-y^3
```

236. `ldev(`ℓ`,`s`)`
:: リスト s の整数倍をリスト ℓ に加えて，成分の絶対値を最小にする
- $\ell + ms$ が求めるものとするとき，$[m, \ell + ms]$ 返す.

```
[0] os_md.ldev([1,3,5,9],[1,-1,1,-1]);
[1] [2,[3,1,7,7]]
```

237. `lsort(`ℓ_1`,`ℓ_2`,`t`|c_1=[`$c_{1,0}$`,`$c_{1,1}$`,...],c_2=[`$c_{2,0}$`,`$c_{2,1}$`,...])`
:: リスト ℓ_1 に対し，ℓ_2 との合併，共通部分，または ℓ_2 や共通部分を除く
その他，表形式データの操作
· ℓ がリストで，t が数または文字列のとき.
- 小さい順にソートされたリストを返す.
- $\ell_2 =$ `[]` で $t = 0$ または `"cup"` のときは l_1 をソート，
  $t = 1$ または`"setminus"`のときは，さらに重複を除く.
  $t = 7$ または`"cut"`のときは `c1=`k で ℓ_1 を最初の k 個の要素と残りの要素の 2 つのリストに分割する．ただし k が負の数のときは，最後の $|k|$ 個を除いた要素と最後の $|k|$ 個の要素との 2 つに分割する.
  $t = 8$ または`"count"`のときは，ℓ_1 の要素数と，それの重複を除いた要素数の組のリストを返す.
- $t = 0, 4$, `"cup"` または `"sum"` のときは ℓ_2 の要素を合わせ（て重複を除い）たリスト
- $t = 1$, `"setminus"` のときは ℓ_2 に含まれない ℓ_1 の要素の（重複を除いた）リスト
- $t = 2$, `"cap"`のときは，共通の要素の（重複を除いた）リスト
- $t = 3$, `"reduce"` のときは，同じものを同数消した最小リストの組を返す.
- $t = 8$, `"count"` のときは，ℓ_1, ℓ_2 の要素数，それぞれで重複を除いた要素数，共通の要素の要素数，同じものを同数消した最初リストの要素数の 8 個の数のリストを返す.
- $t = 9$ または`"cons"`で，ℓ_1 がリストのリストのときは，ℓ_1 の各要素の先頭に ℓ_2 の要素を順に付加する．ℓ_2 の要素の数が ℓ_1 の要素より少ないときは，ℓ_2 の要素の不足部分は，その最後の要素が続いているとみなす．ℓ_2 がリストでないときは，それのみを要素とするリストとみなす.
- $t = 10$ または`"cmp"`で，ℓ_1 と ℓ_2 の違い（その位置と内容の 3 つ組）のリストを出力する．ℓ_1, ℓ_2 はリストまたはリストのリスト.
  $t = 11$ または`"append"`で，ℓ_1, ℓ_2 がリストのリストのとき，要素を append したリストを返す.

· $t =$`[`t_0`,`t_1`,...]` がリストで，ℓ_1 がリストのリストのとき
t_0 については，上の t と同様の指定が可能で，以下のようになる．（エクセルの表のようなものの操作. cf. `readcsv()`）．また $t =$`[]` は，`[0]` と解釈される.
- $\ell_2 =$`[]` または `"sort"` のとき
  - $t_0 = 0$ または `"put"` のときはリストの要素の t_1 番目の項をキーとしてソートする（逆順ソートでは，$-t_1 - 1$ を指定する）
  - $t_0 = 3$ または `"reduce"` のときは，キーが同じものは重複を省いて一つを選ぶ.
- $\ell_2 = 0$ または `"col"` のとき：オプション `c1=`を指定すると
  - $t_0 = 0$ または `"put"` のとき, $c_{1,1}, c_{1,2}, \ldots$ 番目の項目を抜き出したリストのリストを返す.
  - $t_0 = 1$ または `"setminus"` のとき, $c_{1,1}, c_{1,2} \ldots$ 番目の項目を削除したリストのリストを返す.
- $\ell_2 = 1$ または `"num"` のとき

t_1 が存在しないときは，$t_1 = 0$ とみなす.
- t_0 ="col"：項目番号のリストを最初の要素として挿入.
- t_1 が数のとき
  - $t_0 = 0$ または "put"：各要素の先頭に順に t_1 から番号を挿入する.
  - $t_0 = 1$ または "get"：c_1 で指定された番号 $c_{1,0}, c_{1,1}, \cdots$ の要素をこの順に抜き出す.
  - $t_0 = 2$ または "sub"：c_1 で指定された番号の要素を削除.
  - $t_0 = 3$ または "sum"：要素の個数を返す
- それ以外のとき，以下のように i 番目の要素について値 V_i を得る.
  - t_1 が函数子のとき，その要素（ただし，t_2 が指定されていれば，要素の t_2 番目の項目）を引数として返された値．isyes() や calc() などが利用できる.
  - t_1 がリスト [s,v] のときは，函数 calc($\cdot$,[s,v]) という函数にその要素（ただし，t_2 が指定されていれば，要素の t_2 番目の項目）を入れて返された値
    s は ">", "<", "=", ">=", "<=", "!=" などが可能.
  - t_1 が文字列 "+" のとき，各要素の $c_{1,1}, c_{1,2} \ldots$ 番目の項目の和（数でないものは 0 とみなして足す）.

  この値に V_i 対し
  - $t_0 = 0$ または "put"：各要素の先頭に V_i を挿入する.
  - $t_0 = 1$ または "get"：V_i が 0 でない要素のみ抜き出す.
  - $t_0 = 2$ または "sub"：V_i が 0 の要素のみ抜き出す.
  - $t_0 = 3$ または "sum"：V_i の和を返す

- ℓ_2 ="transpose" で t_0 ="put" のとき, 行と列を逆にする（転置）.
- ℓ_2 ="adjust" で t_0 ="put" のとき, 要素の項目の個数が異なっているとき，最大の個数に揃え，後ろには空文字列 "" を詰める．なお，t_1 で空きに詰めるものを指定できる.
- ℓ_2 ="subst" で t_0 ="put" のとき t_j=[a_j,b_j,c_j]：a_j 番目の要素の b_j 番目の項を c_j で置き換える.
- ℓ_1, ℓ_2 がリストのリストのとき，ℓ_1 の要素の t_1 番目の項を ℓ_1 のキー項目，ℓ_2 の要素の t_2 番目の項を ℓ_2 のキー項目と指定して，以下を行う.
  - t_1 や t_2 が省略された場合は，それらは 0 とみなす.
  - ℓ_2 の各要素がリストではない 1 項目からなっているとき，その項目のみを並べたリストを ℓ_2 としてもよい.
  - $t_0 = 0$ または "cup" のとき, ℓ_1 の要素に同じキーの ℓ_2 の要素を付け加える.
  - $t_0 = 2$ または "cap" のとき, 上と同様であるが，同じキーをもつ ℓ_2 の要素がないものは削除する.
  - $t_0 = 1$ または "setminus" のときは, 同じキーをもつ ℓ_2 の要素があるものを ℓ_1 から削除する（ℓ_2 はキー以外の項は無視される）.
  - $t_0 = 3$ または "sum" のときは, 同じキーをもつ ℓ_1 の要素と ℓ_2 の要素はつにまとめ，ℓ_1 と ℓ_2 を合併する.
  - $t_0 = 4$ または "over" のときは, 同じキーをもつ ℓ_1 の要素と ℓ_2 の要素に対し，ℓ_1 の $c_{1,\nu}$ 番目の項目を ℓ_2 の要素の $c_{2,\nu}$ 番目の項目によって置き換えて返す.
    ℓ_1 と ℓ_2 の要素が 1 つのときは，リストにせずにそのまま指定してもよい．次においても同様.
  - $t_0 = 5$ または "subst" のときは, 同じキーをもつ ℓ_1 の要素と ℓ_2 の要素に対し，ℓ_2 の要素の $c_{2,\nu}$ 番目の項目が空（文字列）でなければ，ℓ_1 の $c_{1,\nu}$ 番目の項目をそれで置き換えて返す.

  上を行って，$t_0 = 4, 5$ 以外では，オプション c1 と c2 で指定した ℓ_1 と ℓ_2 の要素の項目を抜き出した（指定しないときは（$t_0 = 3$ 以外では ℓ_2 のキーを除いた）全項目のリストのリストを返す.

```
[0] os_md.lsort([3,2,0,a,1,2,b],[],0);
[0,1,2,2,3,b,a]
[1] os_md.lsort([3,2,0,a,1,2,b],[],1);
[0,1,2,3,b,a]
```

```
[2] os_md.lsort([3,2,0,a,1,2,b],[a,c,0],"cup");
[0,1,2,3,c,b,a]
[3] os_md.lsort([3,2,0,1,2,a,b],[a,c,0],"setmins");
[1,2,3,b]
[4] os_md.lsort([3,2,0,a,1,2,b],[a,c,0],"cap");
[0,a]
[5] os_md.lsort([3,2,0,a,1,2,b],[a,c,2],"reduce");
[[0,1,2,3,b],[c]]
[6] os_md.lsort([3,2,0,a,1,2,b],[],"cut"|c1=2);
[[3,2],[0,a,1,2,b]]
[7] os_md.lsort([3,2,0,a,1,2,b],[],"cut"|c1=-2);
[[3,2,0,a,1],[2,b]]
[8] os_md.lsort([3,2,0,1,2,a,b],[a,c,0],"count");
[7,3,6,3,2,5]
[9] L=[[5,"A","P"],[2,"C","Q"],[3,"B","P"]]$
[10] os_md.lsort(L,"sort",["put",0]);     /* sort by key 0 */
[[2,C,Q],[3,B,P],[5,A,P]]
[11] os_md.lsort(L1,"sort",["put",-2]);   /* reverse sort by key 1 */
[[2,C,Q],[3,B,P],[5,A,P]]
[12] os_md.lsort(L,"sort",["reduce",2]);  /* reduce duplication by key 2 */
[[3,B,P],[2,C,Q]]
[13] os_md.lsort(L,"col",["put"]|c1=[2,1]);    /* extract column */
[[P,A],[Q,C],[P,B]]
[14] os_md.lsort(L,"col",["setminus"]|c1=[0]); /* delete column */
[[A,P],[C,Q],[B,P]]
[15] os_md.lsort(L,"num",["put",1000]);   /* put number */
[[1000,5,A,P],[1001,2,C,Q],[1002,3,B,P]]
[16] os_md.lsort(L,"num",["col"]);        /* add column numbers */
[[0,1,2],[5,A,P],[2,C,Q],[3,B,P]]
[17] os_md.lsort(L,"transpose",[]);   /* transpose */
[[5,2,3],[A,C,B],[P,Q,P]]
[18] os_md.lsort(L,"num",["get"]|c1=[2,1]);  /* get lines */
[[3,B,P],[2,C,Q]]
[19] os_md.lsort(L,"num",["sub"]|c1=[2,1]); /* delete lines */
[[5,A,P]]
[20] os_md.lsort(L,"num",["sum"]);    /* num. of elements */
3
[21] os_md.lsort(L,"num",["put",[">","B"],1]);  /* put number */
[[0,5,A,P],[1,2,C,Q],[0,3,B,P]]
[22] os_md.lsort(L,"num",["get",["=","B"],1]);  /* get elements */
[[3,B,P]]
[23] os_md.lsort(L,"num",["get",[">=","B"],1]); /* get elements */
[[2,C,Q],[3,B,P]]
[24] os_md.lsort(L,"num",["get",["!=","B"],1]); /* get elements */
```

```
[[5,A,P],[2,C,Q]]
[25] os_md.lsort(L,"num",["sub",["=","B"],1]);  /* substract elements */
[[5,A,P],[2,C,Q]]
[26] os_md.lsort(L,"num",["sum",["!=","B"],1]); /* mum. of elements */
2
[27] os_md.lsort(L,"num",["sum","+"]|c1=[0]);   /* sum of column entries */
10
[28] os_md.lsort(L,"subst",[]|c1=[[0,1,"#"],[1,2,"?"]]); /* substitute */
[[5,#,P],[2,C,?],[3,B,P]]
[29] os_md.lsort(L,"","cons");                  /* cons "" */
[[,5,A,P],[,2,C,Q],[,3,B,P]]
[30] os_md.lsort(L,["a","b"],"cons");           /* cons elements */
[[a,5,A,P],[b,2,C,Q],[b,3,B,P]]
[31] L2=[["a",3,"r"],["c",2,"p"],["b",7,"q"]]$
[32] os_md.lsort(L,L2,["cup",0,1]);  /* cup two ll */
[[5,A,P,,],[2,C,Q,c,p],[3,B,P,a,r]]
[33] os_md.lsort(L,L2,["cap",0,1]);  /* cap two ll */
[[2,C,Q,c,p],[3,B,P,a,r]]
[34] os_md.lsort(L,L2,["sum",0,1]);  /* sum two ll */
[[5,A,P,,,],[2,C,Q,c,2,p],[3,B,P,a,3,r],[,,,b,7,q]]
[35] os_md.lsort(L,L2,"append");                /* append two ll */
[[5,A,P,a,3,r],[2,C,Q,c,2,p],[3,B,P,b,7,q]]
[36] os_md.lsort(L,[2,5],["cap",0]);  /* get lines by list */
[[5,A,P],[2,C,Q]]
[37] os_md.lsort(L,[2,5],["setminus",0]);  /* delete lines by list */
[[3,B,P]]
[38] os_md.lsort(L,L2,["subst",0,1]|c1=1,c2=0); /* substitute */
[[5,A,P],[2,c,Q],[3,a,P]]
[39] os_md.lsort(L,L2,["subst",0,1]|c1=[1,2],c2=[0,1]); /* substitute */
[[5,A,P],[2,c,2],[3,a,3]]
[40] os_md.lsort([[a,b,c],[d,e,f]],[[a,b,c],[d,h,f]],"cmp");
[[[[1,1],e,h]]]
[41] os_md.lsort([[a,b,c],[d,e,f]],[[a,b,c],[d,e]],"cmp")$
Different size :line  1
[42] os_md.lsort([[a,b,c],[d,e,f]],"transpose",[]);
[[a,d],[b,e],[c,f]]
```

238. msort(ℓ,s)

:: ベクトルやリストの成分を（多重）キー s に従ってソートする

- s =[f,m]

  $f=1$：小さい順，$f=-1$：大きい順

  キーの m は

  - 0：成分の大小
  - 1：成分がベクトルまたはリストのとき，その長さ
  - 2：成分がベクトルまたはリストのとき，先頭からの辞書式順序

- 3：成分のタイプ
- 4：成分の絶対値
- 5：表示の長さ
- fn：引数が 2 個の比較の函数子 fn を用いる
- [fn]：引数が 1 個の函数 fn に代入して返される値の大小

- s=[f,m,k]：成分がベクトルまたはリストのとき，$(k+1)$ 番目の成分で比較する
- s=[f,m,k_1,k_2]：成分がベクトルまたはリストのベクトルまたはリストのとき，(k_1+1) 番目の成分の (k_2+1) 番目の成分で比較する（さらに s=[f,m,k_1,k_2,k_3] なども可能）.
- s=[s_1,s_2,...]：s_i は上のようなキーで，s_1 のキーでの大小比較で同一となったときは s_2 キーで比較というようにキーの辞書式順序で比較する.

```
[0] os_md.msort([3,2,1,5],[-1,0]);  /* 逆順 */
[5,3,2,1]
[1] os_md.msort([[1,4],[3,2,1],[4,5]],[1,0,1]); /* 2 番目の成分 */
[[3,2,1],[1,4],[4,5]]
[2] os_md.msort([[1,4],[3,2,1],[4,5]],[-1,1]);  /* 長い順 */
[[3,2,1],[4,5],[1,4]]
[3] os_md.msort([[2,2],[3,1],[2,1,1]],[-1,2]);  /* 辞書式順序の大きい順 */
[[3,1],[2,2],[2,1,1]]
[4] os_md.msort([[1,4],[3,2,1],[4,5]],[[-1,1],[1,0,0]]); /* マルチキー */
[[3,2,1],[1,4],[4,5]]
[5] os_md.msort(["This","is","not","an","ant"],[[1,[str_len]],[1,0]]);
[an,is,ant,not,This]
```

239. addIL([a,b],[[a_1,b_1],[a_2,b_2],...]|in=t)
:: 有限区間の合併集合と有限区間の和集合の計算

- $a_1 < b_1 < a_2 < b_2 < \ldots$ となっている必要がある.
- [a,b] の代わりに最初の引数が 0 のときは，第 2 引数を上の形に直す.
- in=-1 を指定すると，[a,b] から [[a_1,b_1],[a_2,b_2],...] を除いた集合を返す.
- in=1 を指定していて最初の引数が数 a とすると，a が集合に含まれているかどうかを返す.
- in=2 を指定していて最初の引数が数 a とすると，a が集合に含まれているかどうかを返す．ただし区間の端になるときは 2 を返す.
- in=2 を指定していて最初の引数が数 a とすると，a が含まれている区間を返す．それがないときは 0 を返す.

```
[0] os_md.addIL([6,7],[[2,5],[8,11]]);
[[2,5],[6,7],[8,11]]
[1] os_md.addIL([4,6],[[2,5],[8,11]]);
[[2,6],[8,11]]
[2] os_md.addIL([1,6],[[2,5],[8,11]]);
[[1,6],[8,11]]
[3] os_md.addIL([9,10],[[2,5],[8,11]]);
[[2,5],[8,11]]
[4] os_md.addIL([4,8],[[2,5],[8,11]]);
[[2,11]]
[5] os_md.addIL(0,[[4,8],[2,5],[8,11]]);
[[2,11]]
[6] os_md.addIL([0,12],[[2,5],[8,11]]|in=-1);
```

```
[[0,2],[5,8],[11,12]]
[7] os_md.addIL([6,9],[[2,5],[8,11]]|in=-1);
[[6,8]]
[8] os_md.addIL(3,[[2,5],[8,11]]|in=1);
1
[9] os_md.addIL(6,[[2,5],[8,11]]|in=1);
0
[10] os_md.addIL(5,[[2,5],[8,11]]|in=1);
1
[11] os_md.addIL(5,[[2,5],[8,11]]|in=2);
2
[12] os_md.addIL(3,[[2,5],[8,11]]|in=3);
[2,5]
[13] os_md.addIL(6,[[2,5],[8,11]]|in=3);
0
```

240. vnext(v)
:: ベクトルの成分を並べ替え，辞書式順序で次のベクトルに変換する
- 成分に等しいものがあってもよい.
- 最後のベクトルを与えたときは 0，それ以外では 1 を返す.

```
[0] V = newvect(7, [3,1,5,7,6,4,2]);
[ 3 1 5 7 6 4 2 ]
[1] os_md.vnext(V); V;
1
[2] [ 3 1 6 2 4 5 7 ]
[3] V = newvect(4, [4,3,2,1]);
[ 4 3 2 1 ]
[4] os_md.vnext(V); V;
0
[5] [ 4 3 2 1 ]
[6] V = newvect(7, [2,1,5,5,6,5,2]);
[ 2 1 5 5 6 5 2 ]
[7] os_md.vnext(V); V;
1
[8] [ 2 1 5 6 2 5 5 ]
```

241. vgen(v,w,m|opt=0)
:: 成分の和が m の非負成分のベクトル w を順に生成する
- v, w は同じサイズのベクトルで，$w[i] \leq v[i]$ の制限つき.
- opt=0 は，初期化でそのときの戻り値は 0．それ以外での戻り値は書き換えられた最高桁
- 最後には 0 を返して，初期化される.

```
[0] V=newvect(3,[2,1,2]);
[ 2 1 2 ]
[1] W=newvect(3);
[ 0 0 0 ]
```

```
[2] os_md.vgen(V,W,2|opt=0);
0
[3] W;
[ 2 0 0 ]
[4] os_md.vgen(V,W,2);
1
[5] W;
[ 1 1 0 ]
[6] os_md.vgen(V,W,2);
2
[7] W;
[ 1 0 1 ]
[8] os_md.vgen(V,W,2);
1
[9] W;
[ 0 1 1 ]
[10] os_md.vgen(V,W,2);
2
[11] W;
[ 0 0 2 ]
[12] os_md.vgen(V,W,2);
0
[13] W;
[ 2 0 0 ]
```

242. `paracmpl(`l`,`t`|lim=1,low=1,para=[`$v_1, v_2, \ldots$`])`
:: 有理的に t に依存したリスト l の成分（たとえば有理式）で張られる空間の正則完備化
l の成分が有理式 $f_1(x,t),\ldots,f_m(x,t)$ のとき（x は多変数でよい），generic に $t=t_0$ と特殊化した $f_1(x,t_0),\ldots,f_m(x,t_0)$ の一次独立な個数を $\bar{m}$ とする．1 変数 t の有理式 $g_{i,j}(t)$ を適当に取って

$$h_j(x,t)=\sum_{i=1}^{m} f_i(x,t)g_{i,j}(t)$$

とおくと，0 でない $h_j(x,t)$ の個数は $\bar{m}$ で，しかも $h_j(x,t)$ は $t=0$ で極を持たず，$h_1(x,0),\ldots,h_m(x,0)$ の一次独立な個数は $\bar{m}$ とできる（cf. [OSe, Proposition 2.21], [O4, Lemma 6.3]）．
`paracmpl([`$f_1(x,t),\ldots,f_m(x,t)$`],t)` は，この `[[`$h_1(x,t),\ldots,h_m(x,t)$`],(`$g_{i,j}(t)$`)]` を返す．

- l はベクトルでもよいが，リストで返される．
- $[h_1(x,t),\ldots,h_m(x,t)]$ は，t の有理式を成分とする可逆行列 $A=\left(a_{i,j}(t)\right)_{\substack{1\le i\le m\\ 1\le j\le m}}$ で，A と A^{-1} の各成分が $t=0$ に極を持たないものによる変換を除いて一意となる．
- 返される $G=\left(g_{i,j}(t)\right)_{\substack{1\le i\le m\\ 1\le j\le m}}$ は上三角行列となる．
- `low=1` を指定すると，x の次数が小さい項に注目する．
- `lim=1` を指定すると `[`$h_1(x,0),\ldots,h_m(x,0)$`]` を返す．
- $f_i(x,t)$ は有理式でなく，有理式を成分とするリストやベクトルでもよい．
- `para=[`$v_1,\ldots,v_k$`]` を指定すると，数係数でなくて不定元 $v_1,\ldots,v_k$ 係数の有理関数体係数の有理式とみなす．
- $\bar{m}=m$ のとき，$f_i(x,t)$ が有理式でなくて $\exp((1+t)x)$ のような式の場合は，適当な次数まで Taylor 展開した式で $f_i(t,x)$（の分母分子）を置き換え，`low=1` を指定して求めた G で変換すれば

よい.

```
[0] os_md.paracmpl([1/(x-1),1/((t+1)*x-1)],t);
[[(1)/(x-1),(1)/((t+1)*x^2+(-t-2)*x+1)],[ 1 (1)/(t) ]
[ 0 (-t-1)/(t) ]]
[1] os_md.paracmpl([1/(1-x),1/(1-(t+1)*x)],t|lim=1);
[ (1)/(x-1) (1)/(x^2-2*x+1) ]
[2] os_md.paracmpl([1/(x-1),1/((t+1)*x-1)],t|low=1);
[[(-1)/(x-1),(x)/((t+1)*x^2+(-t-2)*x+1)],[ -1 (1)/(t) ]
[ 0 (-1)/(t) ]]
[3] os_md.paracmpl([1/(x-1),1/((t+1)*x-1)],t|low=1,lim=1);
[ (-1)/(x-1) (x)/(x^2-2*x+1) ]
[4] os_md.paracmpl([[1/t,1,1],[1+t,0,-t]],t);
[[[1,t,t],[0,t+1,t+2]],[ t t+1 ]
[ 0 (-1)/(t) ]]
[5] os_md.paracmpl([[1/t,1,1],[1+t,0,-t]],t|lim=1);
[[1,0,0],[0,1,2]]
```

上の [0]〜[5] は以下を示している.

$$\left(\frac{1}{x-1},\ \frac{1}{(t+1)x^2+(-t-2)x+1}\right)=\left(\frac{1}{x-1},\ \frac{1}{(t+1)x-1}\right)\begin{pmatrix}1 & \frac{1}{t}\\ 0 & \frac{-t-1}{t}\end{pmatrix}$$
$$=\left(\frac{1}{x-1},\frac{1}{(x-1)^2}\right)\qquad \text{if } t=0,$$
$$\left(\frac{1}{1-x},\ \frac{x}{(t+1)x^2+(-t-2)x+1}\right)=\left(\frac{1}{x-1},\ \frac{1}{(t+1)x-1}\right)\begin{pmatrix}-1 & \frac{1}{t}\\ 0 & -\frac{1}{t}\end{pmatrix}$$
$$=\left(\frac{1}{1-x},\frac{x}{(1-x)^2}\right)\qquad \text{if } t=0,$$
$$\left(\begin{pmatrix}1\\ t\\ t\end{pmatrix},\begin{pmatrix}0\\ t+1\\ t+2\end{pmatrix}\right)=\left(\begin{pmatrix}\frac{1}{t}\\ 1\\ 1\end{pmatrix},\begin{pmatrix}t+1\\ 0\\ t\end{pmatrix}\right)\begin{pmatrix}t & t+1\\ 0 & -\frac{1}{t}\end{pmatrix}$$
$$=\left(\begin{pmatrix}1\\ 0\\ 0\end{pmatrix},\begin{pmatrix}0\\ 1\\ 2\end{pmatrix}\right)\qquad \text{if } t=0.$$

ここで以下に注意

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

微分方程式

$$\big((t+1)x^2-(t+2)x+1\big)\frac{d^2u}{dx^2}+2\big(2(t+1)x-t-2\big)\frac{du}{dx}+2(t+1)u=0$$

の解空間は $t\neq 0$ のとき $\{\frac{C_1}{x-1}+\frac{C_2}{(t+1)x-1}\mid C_1,C_2\in\mathbb{C}\}$ である. このことから $t=0$ のときの解空間は $\{\frac{C_1}{x-1}+\frac{C_2}{(x-1)^2}\mid C_1,C_2\in\mathbb{C}\}$ となることが分かる. より一般に, 任意の t で解空間は $\{\frac{C_1}{x-1}+\frac{C_2x}{(t+1)^2x^2-(t+2)x+1}\mid C_1,C_2\in\mathbb{C}\}$ で与えられる.

$x = 0$ の近くで考えるときは，low=1 を指定する方が便利なことが多い．

3.2.6 Matrices

243. dupmat(m)
:: 成分が有理式の行列またはベクトル m の複製を作る

```
[0] A=B=newmat(2,2,[[1,2],[3,4]]);
[ 1 2 ]
[ 3 4 ]
[1] C=os_md.dupmat(A);
[ 1 2 ]
[ 3 4 ]
[2] C[0][0]=0;
0
[3] C;
[ 0 2 ]
[ 3 4 ]
[4] A;
[ 1 2 ]
[ 3 4 ]
[5] B[0][0]=0;
0
[6] B;
[ 0 2 ]
[ 3 4 ]
[7] A;
[ 0 2 ]
[ 3 4 ]
```

244. m2v(m)
:: 行列 m の成分を 1 行目から順に並べてベクトルに変換する

```
[0] M=newmat(2,2,[[a,b],[c,d]]);
[ a b ]
[ c d ]
[1] os_md.m2v(M);
[ a b c d ]
```

245. m2l(m|flat=1)
:: 有理式，ベクトルあるいは行列の m の成分を順に並べてリストにする
flat=1：リストのリストのときは，一つ内側のリストを外して並べる．
たとえば，リスト $\ell_1, \ell_2, \ell_3, \ell_4$ をこの順に連結したリストを作るには m2l([ℓ_1,ℓ_2,ℓ_3,ℓ_4]|flat=1) とすればよい．

```
[0] M=newmat(2,2,[[a,b],[c,d]]);
[ a b ]
[ c d ]
[1] os_md.m2l(M);
```

```
[a,b,c,d]
[2] os_md.m2lv(M);
[[ a b ],[ c d ]]
[3] os_md.m2l(a);
[a]
[4] os_md.m2l([a,b]);
[a,b]
[5] os_md.m2l([[1,2],[3,4,[5,6],7],[8,9]]|flat=1);
[1,2,3,4,[5,6],7,8,9];
```

246. `m2lv(m)`
:: 行列 m の行ベクトルを並べてリストにする（cf. 245, 248）
247. `m2ll(m)`
:: 行列 m を行ベクトルをリストにかえたものを成分とするリストにする（cf. 248）
248. `lv2m(`ℓ`|fill=`n`)`
:: 行ベクトル（行成分のリストでも可）のリスト ℓ から行列を作る
   - 行列の列の大きさは，行ベクトルの最大長で決まり，行ベクトルのサイズが小さい時は，後ろに 0 が補充される（cf. `mat()`）.
   - `fill=`n：行ベクトルのサイズが小さいとき，0 でなくて n を補充する.

```
[0] A=newmat(3,2,[[1,2],[3,4],[5,6]]);
[ 1 2 ]
[ 3 4 ]
[ 5 6 ]
[1] LV=os_md.m2lv(A);
[[ 1 2 ],[ 3 4 ],[ 5 6 ]]
[2] LL=os_md.m2ll(A);
[[1,2],[3,4],[5,6]]
[3] os_md.lv2m(LV);
[ 1 2 ]
[ 3 4 ]
[ 5 6 ]
[4] os_md.lv2m(LL);
[ 1 2 ]
[ 3 4 ]
[ 5 6 ]
[5] A0=newvect(2,[1,2]);
[ 1 2 ]
[6] A1=newvect(3,[1,2,3]);
[ 1 2 3 ]
[7] os_md.lv2m([A0,A1]);
[ 1 2 0 ]
[ 1 2 3 ]
[8] os_md.lv2m([[1,2],[1,2,3]]);
[ 1 2 0 ]
[ 1 2 3 ]
```

```
[9] os_md.lv2m([[1,2],[1,2,3]]|fill=" ");
[ 1 2   ]
[ 1 2 3 ]
```

249. s2m(s)

:: 文字列での有理数成分の行列表現や，列のリストから行列を作る．必要最低サイズの行列になる

- 整数や有理数成分の行列を，行の区切りをコンマで示した文字列で指定する．成分が一桁の整数のときに便利（以下の [0], [1], [4], [5] を参照）．
  - 同じ数が例えば 3 回続くときは，^3 をつけて表してもよい．
  - a,b,... は 10, 11, ... を表す．
  - 10 以上の数は (10) のように括弧で囲んで表してもよい．たとえば，2/b は $\frac{2}{11}$．
  - 分数は / を使って表す．
  - たとえば 12(-2)/3^2 は，$1, 2, -\frac{2}{3}, -\frac{2}{3}$ と 4 つの数字が並んでいると解釈される．
- s2sp() あるいは Risa/Asir の表示の文字列によって行列成分を表すリストから行列に戻す機能をもつ（以下の [0], [6] の例を参照）．
  すなわち，eval_str() を式でなくて，式が成分のリストや行列に拡張した機能．
- 行列を作成する newmat(m, n, l) の ℓ を [ ] で囲って渡してもよい（以下の [2], [3] の例）．
- 行列のサイズは自動判定されるが mat() と異なり，列サイズは ℓ の成分の最大の長さとなる（mat() は成分の最初のリストの長さ）．
- s が行列の時は，そのまま返す．

```
[0] os_md.s2m("21,111,00a");
[ 2 1 0 ]
[ 1 1 1 ]
[ 0 0 10 ]
[1] os_md.s2m("-1,0-1,0^2-1,0^3-1");
[ -1 0 0 0 ]
[ 0 -1 0 0 ]
[ 0 0 -1 0 ]
[ 0 0 0 -1 ]
[2] os_md.s2m([[0,a],[0,0,a],[0]]);
[ 0 a 0 ]
[ 0 0 a ]
[ 0 0 0 ]
[3] mat([0,a],[0,0,a],[0]);
[ 0 a ]
[ 0 0 ]
[ 0 0 ]
[4] os_md.s2m("123,32-2/3^2");
[ 1 2 3 0 ]
[ 3 2 -2/3 -2/3 ]
[5] os_md.s2m("123,32(-3/b)^2");
[ 1 2 3 0 ]
[ 3 2 -3/11 -3/11 ]
[6] os_md.s2m("[x*x 1+2][z w+1]");
[ x^2 3 ]
[ z w+1 ]
```

250. c2m(l,v|pow=t)

:: 基底の変換から係数行列を作る

- l は一次変換した結果のリスト
- t は基底（変数）のリスト
- pow=t を指定したときは，v は変数で v^j の $j=0,1,\ldots,t$ を基底と考えた係数の行列（v の t 次多項式の変換）.

```
[0] os_md.c2m([a*x+b*y,c*x+d*y],[x,y]);
[ a b ]
[ c d ]
[1] os_md.c2m([1+2*x+3*x^2,2*x+4*x^2,1+3*x+5*x^2],x|pow=2);
[ 1 2 3 ]
[ 0 2 4 ]
[ 1 3 5 ]
[2] os_md.c2m([1+2*x+3*x^2,2*x+4*x^2,1+3*x+5*x^2],x);  /* 次数は自動判断 */
[ 1 2 3 ]
[ 0 2 4 ]
[ 1 3 5 ]
```

251. mperm(m,[σ_0,$\sigma_1\ldots$],[τ_0,$\tau_1,\ldots$])
mperm(m,[[σ_0,σ_1]],[[τ_0,τ_1]]), mperm(m,[σ,[m_1]],[τ,[m_2]])

:: 行列 m から小行列 $\left(m_{\sigma_i\tau_j}\right)$ を作る，または置換行列（または互換）で変換する

$M=\left(m_{ij}\right)_{\substack{0\le i<m\\ 0\le j<n}}$ は $\left(m_{\sigma_i\tau_j}\right)$ に変わる.

- 元の行列は破壊されず，新たな行列が返される.
- 第 2 引数が [[σ_0,σ_1]] となっていると，σ_0 行目と σ_1 行目の入れ替えを意味する.
- 第 3 引数が 1 のときは，第 3 引数が第 2 引数に等しいことを意味する.
- 第 2 引数，または第 3 引数が 0 のときは，それらが恒等変換であることを意味する.
- [σ,[k]] は [$\sigma,\sigma+1,\ldots,\sigma+k-1$] と解釈される.
- m はベクトルまたはリストでもよい．このとき第 3 引数は無視される.
- たとえば，$\sigma\le i<\sigma+k$, $\sigma\le j<\sigma+k$ を満たす m の成分 $m_{i,j}$ からなる k 次の正方行列 $\left(m_{i,j}\right)_{\substack{\sigma\le i<\sigma+k\\ \sigma\le j<\sigma+k}}$ は mperm(m,[σ,[k]],1) によって，1, 3, 5 行目のみを抜き出して得られる行列は mperm(m,[1,3,5],0) によって得られる.

```
[0] A=newmat(3,3,[[a,b,c],[d,e,f],[g,h,i]]);
[ a b c ]
[ d e f ]
[ g h i ]
[1] os_md.mperm(A,1,[1,2,0]);
[ e f d ]
[ h i g ]
[ b c a ]
[2] os_md.mperm(A,[1,2],[0,1]);
[ d e ]
[ g h ]
[3] os_md.mperm(A,[1,[2]],[0,[3]]);
[ d e f ]
[ g h i ]
```

```
[4] os_md.mperm(A,[1,[2]],1);
[ e f ]
[ h i ]
[5] os_md.mperm(A,[[0,1]],1);
[ e d f ]
[ b a c ]
[ h g i ]
[6] os_md.mperm([a,b,c,d],[1,3,0,2],0);
[b,d,a,c]
[7] os_md.mperm(ltov([a,b,c,d]),[1,3,0,2],0);
[ b d a c ]
```

252. mtranspose(m)
:: 行列 m の転置行列を求める（m はリストのリストでもよい）
リストのリストの時は，内部のリストの長さが順に変わらないか減少していれば転置となる．
（行列のときの例は mtoupper() の項）

```
[0] L=[[1,2,3],[4,5],[6,7]]$
[1] os_md.mtranspose(L);
[[1,4,6],[2,5,7],[3]]
```

253. madjust(m,w|null=n)
:: 行列 m の列数を w に調整する
- $w > 0$ のときは，列数が w になるように分割し，分割されたブロックは行を増やして繋げる．
- $w < 0$ のときは，列数を $|w|$ 倍し，$w > 0$ のときの逆操作で行数を減らす．
- null=n：行列の列数調整で定義されない成分を n とする．$n = 0$ がデフォルト．
- m はリストのリストでもよい．

```
[0] M=os_md.mgen(3,5,a,1);
[ a11 a12 a13 a14 a15 ]
[ a21 a22 a23 a24 a25 ]
[ a31 a32 a33 a34 a35 ]
[1] os_md.madjust(M,3);
[ a11 a12 a13 ]
[ a21 a22 a23 ]
[ a31 a32 a33 ]
[ a14 a15 0 ]
[ a24 a25 0 ]
[ a34 a35 0 ]
[2] os_md.madjust(M,-2|null="?");
[ a11 a12 a13 a14 a15 a31 a32 a33 a34 a35 ]
[ a21 a22 a23 a24 a25 ? ? ? ? ? ]
```

254. mpower(m,n)
:: 行列 m の n 乗を求める
n は整数．

```
[0] M=newmat(2,2,[[a,b],[c,d]]);
[ a b ]
```

```
[ c d ]
[1] os_md.mpower(M,0);
[ 1 0 ]
[ 0 1 ]
[2] os_md.mpower(M,3);
[ a^3+2*c*b*a+d*c*b b*a^2+d*b*a+c*b^2+d^2*b ]
[ c*a^2+d*c*a+c^2*b+d^2*c c*b*a+2*d*c*b+d^3 ]
[3] os_md.mpower(M,-2);
[ (c*b+d^2)/(d^2*a^2-2*d*c*b*a+c^2*b^2) (-b*a-d*b)/(d^2*a^2-2*d*c*b*a+c^2*b^2) ]
[ (-c*a-d*c)/(d^2*a^2-2*d*c*b*a+c^2*b^2) (a^2+c*b)/(d^2*a^2-2*d*c*b*a+c^2*b^2) ]
```

n が正の時は，$m, m^2, m^4 = (m^2)^2, \ldots$ と計算して積を求めている (n を 2 進法で表した 1 の部分に対応する積)．負の時は m^{-1} に対して同様な計算．

255. `mtoupper(`m`,`n`|opt=`t`,step=1,dviout=1,pages=1,tab=`k`,lim=`w`)`
:: 行列 m に対し，以下の行基本変形を行って，行の先頭からの 0 の成分の個数が下の行の方へ狭義単調増加となるようにする．最後の n 列は無視．途中の行から全て 0 になることもある．
- 行頭からの 0 の成分の個数の下方への単調性を保って，以下の操作を 1 行目から順に行う．
  - m のある行のスカラー倍を別の行に加える
  - 行を入れ替え，(`opt>1` でなければ) 後ろに移動した行を -1 倍する．
- `opt=1`：さらに行の 0 でない先頭の要素のある列は，行変形でその要素以外を 0 にする．
- `opt=2`：さらに行に 0 でないスカラーをかけて 0 でない行の先頭の成分を 1 にする．
- `opt=3`：`opt=2` とほぼ同じだが，より簡単な計算になるように行交換を増やす (以下の (b), (c) のみ考慮)．
- `opt=4`：`opt=3` とほぼ同じだが，さらになるべく分数が現れない簡単な行変形にする．
- `opt=5`：`opt=4` とほぼ同じだが，行列が一つのパラメータを多項式として含むとき，そのパラメータによって必要に応じて場合分けを行う．ただし，パラメータの値が有理数で場合分け可能なときに限る．
- `opt=6`, `opt=7`：`opt=5` とほぼ同じだが，行列に含まれるパラメータが 2 個以上でも出来るだけ場合分けを用いて基本変形する．
- 最後の n 列は無視する．よって最後の n 列のみに現れる変数は，上のスカラーには含まれない．通常 n は 0 または 1 または -1.
  - $-n$ が m の行数に等しいときは，m の右側にサイズ $|n|$ の単位行列を付加して列サイズを $|n|$ だけ増やした行列に対し，最後の $|n|$ 列を無視して行変形を行う．
    $N =$`size(`m`)[0]` とおくと `mperm(`m`,-N|opt=2),0,[N,[N]])` が m の変換行列となる．特に m が可逆行列であるならば，これは m の逆行列となる．
  - $n = -1$ のときは，第 k 番目の成分が $(\mathtt{zz})^{k-1}$ となる縦ベクトルを列の最後に付加して列サイズを一つ増やした行列 m' に対して，`mtoupper(`m'`,1)` を行う．
  - A が変数 `zz` を含まない size $S \times S'$ の行列ならば，$B =$ `mtoupper(`A`,-1)` とおくと，変換後の行列は `mperm(`B`,1,[0,[`S'`]])` となり，`mycoef(`B`[I-1][`S'`],J-1,zz)` が左からかける変換行列の (I, J) 成分となる．
- `step=1` を指定したときは，途中の行変形過程を表示する．
- `dviout=1` は `step=1` を指定したときのみ有効で，途中の行変形過程を `dviout` で表示する．
  - このとき `cr=` のオプションで改行の LaTeX コードを指定可能 (デフォルトは `cr="\\\\\n & "`で，`cr=7` と同じ)．
    たとえば，`dviout` で表示する場合は，`risaout.tex` の `begin{document}`の直前に
    `\def\pause{\special{dviout '+M-}}`
    の一行を挿入しておき，`cr="\\\\ \\pause\n &"` (`cr=23` と同じ) とすれば，表示の際に space キーを押す毎に 1 ステップずつ進む．
  - 通常は 1 ステップ毎に改行されるが，`lim=`w を指定すると，行幅が w 文字程度と見なして 1

行に複数ステップを入れる．

- `dviout=2` も同様であるが，表示は行わない．
- `dviout=-1` では，TEX のソースを返す．
- `dviout=-2` では，TEX のソースの元となるリストを返す．返されたものを L とすると
  `ltotex(reverse(L)|opt=["cr","spts0"],str=1,cr=7)`
  によって TEX のソースが得られる．
- `tab=`k：`dviout` を指定し `step` に 5 以上を指定したときの場合分けのインデントの幅を k mm とする（デフォルトは $k=2$）．
- `pages=1`：`dviout` を指定したとき，途中で改ページを許す（結果が長くなる場合などに指定）
- `unim()` によって行基本変形の演習用の行列生成ができる．
- 行基本変形の手順は以下のようになっている．
  (a) 行列 $M=(m_{j,\ell})$ は $k-1$ 列まで基本変形が終了した行列で，それの j_0-1 行目までの基本変形が完了しているとする．このとき $k-1$ 列までの成分は j_0 行目以降は 0 となっていて，j_0-1 行目までの行の 0 でない先頭成分は 1 で，それは $k-1$ 列目以下で，その列の他の成分は 0 になっている．また k 列目には j_0 行目またはそれ以降に初めて 0 でない成分 $m_{j,k}$ があるとする．
  以下のように j_0 行目またはそれ以降の 0 でない t 行目の成分 $m_{t,k}$ を調べて基準の j_1 行目を決め，（必要なら）行交換してそれを j_0 行目に移動する．得られた行列 $M=(m_{j,\ell})$ において $m_{j_0,k}$ で j_0 行目を割り，j_0 行目のスカラー倍を他の行に加えて少なくとも k 列目までの基本変形を終了させる．
  (b) j 行目またはそれ以降の成分 $m_{t,k}$ で 1 となるものがあればその最初の t を j_1 行目とおく．
  (c) そうでなくて j 行目またはそれ以降の成分 $m_{t,k}$ で -1 となるものがあればその最初を j_1 行目とおく．
  (d) そうでなくて j 行目またはそれ以降の成分 $m_{t,k}$ が正整数で，その行を $m_{t,k}$ で割ったものが整数行ベクトルになるものがあればその最初の行を j_1 行目とする．
  (e) そうでなくて j 行目またはそれ以降の成分 $m_{t,k}$ が負整数で，その行を $m_{t,k}$ で割ったものが整数行ベクトルになるものがあればその最初の行を j_1 行目とする．
  (f) そうでなくて j 行目またはそれ以降の成分 $m_{t,k}$ が 0 でなくて，その行を $m_{t,k}$ で割ったものが整数行ベクトルになるものがあればその最初の行を j_1 行目とする．
  (g) そうでなければ，t 行目の成分 $m_{t,k}$ が整数で，k 行目以降の t 行目と異なる j 行目に t 行目の整数倍を足して (j,k) 成分を 0 に出来るか調べ，可能ならそれを行う．
  (h) そうでなければ，t 行目の成分 $m_{t,k}$ が整数で，j 行目以降のある行を t 行目に足すか引くかして (t,k) 成分を 1 に出来るか調べ，可能ならそれを行って $j_1=t$ とする．ただし，j 行目と t 行目に変数を含まないものを優先する．
  (i) そうでなければ，t 行目の成分 $m_{t,k}$ が整数で，j 行目以降のある行目に j 行目以降のある行の整数倍を足すか引くかして (t,k) 成分を 1 に出来るか調べ，可能ならそれを行ってその行を $j_1=t$ とする．ただし，j 行目と t 行目に変数を含まないものを優先する．
  (j) そうでなければ，t 行目の成分 $m_{t,k}$ が整数で，j 行目以降のある行に j 行目以降のある行を足すか引くかして (t,k) 成分を -1 に出来るか調べ，可能ならそれを行って $j_1=t$ とする．ただし，j 行目と t 行目に変数を含まないものを優先する．
  (k) そうでなければ，t 行目に j 行目以降のある行の整数倍を足すか引くかして (t,k) 成分を -1 に出来るか調べ，可能ならそれを行って $j_1=t$ とする．ただし，j 行目と t 行目に変数を含まないものを優先する．
  (l) そうでなければ最初に現れた整数成分 $m_{t,k}$ の行を，整数の行がなければパラメータを含まない数 $m_{t,k}$ の最初の行を j_1 行とする．
  (m) `opt>4` のとき，k 列目の有理式 $m_{t,k}$ の分子がある一つパラメータの多項式となっているとき，行変形によって k 列目の成分の分子の多項式の最低次数を下げることができればそれを行うことを続ける．
  (n) `opt>4` のとき，k 列目の成分で分子が数となる $m_{t,k}$ があれば，その最初の行 t によって $j_1=t$ とする．
  (o) `opt>4` のとき，有理式 $m_{t,k}$ で分子の最大公約元の多項式の根が有理数で与えられるときは，その項が 0 となる場合をまず扱って行変形の最終形まで求める．そのあと $j_1=t$ とする．
  (p) `opt>4` のとき，$m_{t,k}$ が 0 になるための条件が，ある変数がその変数を含まない変数の多項式や分母が 0 でない有理式（ただし `opt=5` のときは有理数）で与えられるものがあるかどうか調べ，そのような t が存在すれば 0 になる場合を最終形まで求め，次に 0 にならない場合を調べるため $j_1=t$ とする．
  (q) 以上に該当しなければ，成分 $m_{t,k}$ の型が最小となるものが最初に現れる行 t に対して $j_1=t$ とする．

```
[0] M=newmat(2,3,[[a,b,1],[c,d,x]]);
[ a b 1 ]
```

```
[ c d x ]
[1] os_md.mtoupper(M,1);
[ a b 1 ]
[ 0 (d*a-c*b)/(a) (a*x-c)/(a) ]
[2] os_md.mtoupper(M,0);
[ a b 1 ]
[ 0 (d*a-c*b)/(a) (a*x-c)/(a) ]
[3] os_md.mtoupper(M,1|opt=1);
[ a 0 (-b*a*x+d*a)/(d*a-c*b) ]
[ 0 (d*a-c*b)/(a) (a*x-c)/(a) ]
[4] M=os_md.mtranspose(M);
[ a c ]
[ b d ]
[ 1 x ]
[5] os_md.mtoupper(M,0);
[ a c ]
[ 0 (d*a-c*b)/(a) ]
[ 0 0 ]
[6] M=newmat(3,3,[[0,0,1],[1,1,1],[1,1,2]]);
[ 0 0 1 ]
[ 1 1 1 ]
[ 1 1 2 ]
[7] os_md.myrank(M);
2
[8] os_md.mtoupper(M,0|opt=1,step=1)$
[ 0 0 1 ]
[ 1 1 1 ]
[ 1 1 2 ]

line1 <-> line2
[ 1 1 1 ]
[ 0 0 -1 ]
[ 1 1 2 ]

line3 -= line1
[ 1 1 1 ]
[ 0 0 -1 ]
[ 0 0 1 ]

line1 += line2
[ 1 1 0 ]
[ 0 0 -1 ]
[ 0 0 1 ]
```

```
line3 += line2
[ 1 1 0 ]
[ 0 0 -1 ]
[ 0 0 0 ]
[9] A=mat([2,1],[1,2]);
[ 2 1 ]
[ 1 2 ]
[10] os_md.mtoupper(A,-2|opt=2,step=1)$
[ 2 1 1 0 ]
[ 1 2 0 1 ]

line1 * (1/2)
[ 1 1/2 1/2 0 ]
[ 1 2 0 1 ]

line2 -= line1
[ 1 1/2 1/2 0 ]
[ 0 3/2 -1/2 1 ]

line2 * (2/3)
[ 1 1/2 1/2 0 ]
[ 0 1 -1/3 2/3 ]

line1 += line2 * (-1/2)
[ 1 0 2/3 -1/3 ]
[ 0 1 -1/3 2/3 ]

[11] os_md.mtoupper(A,-2|opt=3,step=1)$
[ 2 1 1 0 ]
[ 1 2 0 1 ]

line1 <-> line2
[ 1 2 0 1 ]
[ 2 1 1 0 ]

line2 += line1 * (-2)
[ 1 2 0 1 ]
[ 0 -3 1 -2 ]

line2 * (-1/3)
[ 1 2 0 1 ]
[ 0 1 -1/3 2/3 ]

line1 += line2 * (-2)
```

```
[ 1 0 2/3 -1/3 ]
[ 0 1 -1/3 2/3 ]

[12] os_md.myinv(A);
[ 2/3 -1/3 ]
[ -1/3 2/3 ]
[13] os_md.mtoupper(A,-1|opt=2);
[ 1 0 -1/3*zz+2/3 ]
[ 0 1 2/3*zz-1/3 ]
[14] os_md.mtoupper(mat([2,1,3],[1,2,3]),-2|opt=3,step=1,dviout=1)$
```

$$\begin{pmatrix}2&1&3&1&0\\1&2&3&0&1\end{pmatrix}$$
$$\xrightarrow{\text{line1} \leftrightarrow \text{line2}} \begin{pmatrix}1&2&3&0&1\\2&1&3&1&0\end{pmatrix}$$
$$\xrightarrow{\text{line2} -= \text{line1}\times(2)} \begin{pmatrix}1&2&3&0&1\\0&-3&-3&1&-2\end{pmatrix}$$
$$\xrightarrow{\text{line2} \times= \left(\frac{-1}{3}\right)} \begin{pmatrix}1&2&3&0&1\\0&1&1&\frac{-1}{3}&\frac{2}{3}\end{pmatrix}$$
$$\xrightarrow{\text{line1} -= \text{line2}\times(2)} \begin{pmatrix}1&0&1&\frac{2}{3}&\frac{-1}{3}\\0&1&1&\frac{-1}{3}&\frac{2}{3}\end{pmatrix}$$

```
[15] os_md.mtoupper(os_md.s2m("32,53"),-2|opt=3,step=1,dviout=1)$
```

$$\begin{pmatrix}3&2&1&0\\5&3&0&1\end{pmatrix}$$
$$\xrightarrow{\text{line1} \times= \left(\frac{1}{3}\right)} \begin{pmatrix}1&\frac{2}{3}&\frac{1}{3}&0\\5&3&0&1\end{pmatrix}$$
$$\xrightarrow{\text{line2} -= \text{line1}\times(5)} \begin{pmatrix}1&\frac{2}{3}&\frac{1}{3}&0\\0&\frac{-1}{3}&\frac{-5}{3}&1\end{pmatrix}$$
$$\xrightarrow{\text{line2} \times= (-3)} \begin{pmatrix}1&\frac{2}{3}&\frac{1}{3}&0\\0&1&5&-3\end{pmatrix}$$
$$\xrightarrow{\text{line1} -= \text{line2}\times\left(\frac{2}{3}\right)} \begin{pmatrix}1&0&-3&2\\0&1&5&-3\end{pmatrix}$$

```
[16] os_md.mtoupper(os_md.s2m("32,53"),-2|opt=4,step=1,dviout=1)$
```

$$\begin{pmatrix}3&2&1&0\\5&3&0&1\end{pmatrix}$$
$$\xrightarrow{\text{line2} -= \text{line1}\times(2)} \begin{pmatrix}3&2&1&0\\-1&-1&-2&1\end{pmatrix}$$
$$\xrightarrow{\text{line1} \leftrightarrow \text{line2}} \begin{pmatrix}-1&-1&-2&1\\3&2&1&0\end{pmatrix}$$
$$\xrightarrow{\text{line1} \times= (-1)} \begin{pmatrix}1&1&2&-1\\3&2&1&0\end{pmatrix}$$
$$\xrightarrow{\text{line2} -= \text{line1}\times(3)} \begin{pmatrix}1&1&2&-1\\0&-1&-5&3\end{pmatrix}$$

$$\xrightarrow{\text{line2 } \times= (-1)} \begin{pmatrix} 1 & 1 & 2 & -1 \\ 0 & 1 & 5 & -3 \end{pmatrix}$$
$$\xrightarrow{\text{line1 } -= \text{ line2}} \begin{pmatrix} 1 & 0 & -3 & 2 \\ 0 & 1 & 5 & -3 \end{pmatrix}$$

```
[16] os_md.mtoupper(mat([a,2],[a+2,4*a+2]),0|step=1,opt=5,dviout=1)$
```

$$\begin{pmatrix} a & 2 \\ a+2 & 4a+2 \end{pmatrix}$$
$$\xrightarrow{\text{line2 } += \text{ line1}\times(-1)} \begin{pmatrix} a & 2 \\ 2 & 4a \end{pmatrix}$$
$$\xrightarrow{\text{line1 } \leftrightarrow \text{ line2}} \begin{pmatrix} 2 & 4a \\ a & 2 \end{pmatrix}$$
$$\xrightarrow{\text{line1 } \times= \left(\frac{1}{2}\right)} \begin{pmatrix} 1 & 2a \\ a & 2 \end{pmatrix}$$
$$\xrightarrow{\text{line2 } += \text{ line1}\times(-a)} \begin{pmatrix} 1 & 2a \\ 0 & -2a^2+2 \end{pmatrix}$$

If $a = 1$,

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

If $a = -1$,

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

If $a^2 - 1 \neq 0$,

$$\xrightarrow{\text{line2 } \times= \left(\frac{-1}{a^2-1}\right)} \begin{pmatrix} 1 & 2a \\ 0 & 1 \end{pmatrix}$$
$$\xrightarrow{\text{line1 } += \text{ line2}\times(-2a)} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

```
[17] os_md.mtoupper(mat([a,b],[c,d]),-2|step=1,opt=6,dviout=1)$
```

$$\begin{pmatrix} a & b & 1 & 0 \\ c & d & 0 & 1 \end{pmatrix}$$

If $a = 0$,

$$\begin{pmatrix} 0 & b & 1 & 0 \\ c & d & 0 & 1 \end{pmatrix}$$

If $c = 0$,

$$\begin{pmatrix} 0 & b & 1 & 0 \\ 0 & d & 0 & 1 \end{pmatrix}$$

If $b = 0$,

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & d & 0 & 1 \end{pmatrix}$$

If $d = 0$,

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

If $d \neq 0$,

$$\xrightarrow{\text{line1} \leftrightarrow \text{line2}} \begin{pmatrix} 0 & d & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$\xrightarrow{\text{line1} \times= \left(\frac{1}{d}\right)} \begin{pmatrix} 0 & 1 & 0 & \frac{1}{d} \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

If $b \neq 0$,

$$\xrightarrow{\text{line1} \times= \left(\frac{1}{b}\right)} \begin{pmatrix} 0 & 1 & \frac{1}{b} & 0 \\ 0 & d & 0 & 1 \end{pmatrix}$$

$$\xrightarrow{\text{line2} \mathrel{+}= \text{line1} \times (-d)} \begin{pmatrix} 0 & 1 & \frac{1}{b} & 0 \\ 0 & 0 & \frac{-d}{b} & 1 \end{pmatrix}$$

If $c \neq 0$,

$$\xrightarrow{\text{line1} \leftrightarrow \text{line2}} \begin{pmatrix} c & d & 0 & 1 \\ 0 & b & 1 & 0 \end{pmatrix}$$

$$\xrightarrow{\text{line1} \times= \left(\frac{1}{c}\right)} \begin{pmatrix} 1 & \frac{d}{c} & 0 & \frac{1}{c} \\ 0 & b & 1 & 0 \end{pmatrix}$$

If $b = 0$,

$$\begin{pmatrix} 1 & \frac{d}{c} & 0 & \frac{1}{c} \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

If $b \neq 0$,

$$\xrightarrow{\text{line2} \times= \left(\frac{1}{b}\right)} \begin{pmatrix} 1 & \frac{d}{c} & 0 & \frac{1}{c} \\ 0 & 1 & \frac{1}{b} & 0 \end{pmatrix}$$

$$\xrightarrow{\text{line1} \mathrel{+}= \text{line2} \times \left(\frac{-d}{c}\right)} \begin{pmatrix} 1 & 0 & \frac{-d}{cb} & \frac{1}{c} \\ 0 & 1 & \frac{1}{b} & 0 \end{pmatrix}$$

If $a \neq 0$,

$$\xrightarrow{\text{line1} \times= \left(\frac{1}{a}\right)} \begin{pmatrix} 1 & \frac{b}{a} & \frac{1}{a} & 0 \\ c & d & 0 & 1 \end{pmatrix}$$

$$\xrightarrow{\text{line2} \mathrel{+}= \text{line1} \times (-c)} \begin{pmatrix} 1 & \frac{b}{a} & \frac{1}{a} & 0 \\ 0 & \frac{da-cb}{a} & \frac{-c}{a} & 1 \end{pmatrix}$$

If $d = \dfrac{cb}{a}$,

$$\begin{pmatrix} 1 & \frac{b}{a} & \frac{1}{a} & 0 \\ 0 & 0 & \frac{-c}{a} & 1 \end{pmatrix}$$

If $da - cb \neq 0$,

$$\xrightarrow{\text{line2} \times= \left(\frac{a}{da-cb}\right)} \begin{pmatrix} 1 & \frac{b}{a} & \frac{1}{a} & 0 \\ 0 & 1 & \frac{-c}{da-cb} & \frac{a}{da-cb} \end{pmatrix}$$

$$\xrightarrow{\text{line1} \mathrel{+}= \text{line2} \times \left(\frac{-b}{a}\right)} \begin{pmatrix} 1 & 0 & \frac{d}{da-cb} & \frac{-b}{da-cb} \\ 0 & 1 & \frac{-c}{da-cb} & \frac{a}{da-cb} \end{pmatrix}$$

```
[18] os_md.mtoupper(mat([a^4-b^2,b]),0|step=1,opt=6,dviout=1)$
```

$$\begin{pmatrix} a^4 - b^2 & b \end{pmatrix}$$

If $b = a^2$,

$$\begin{pmatrix} 0 & a^2 \end{pmatrix}$$

If $a = 0$,

$$\begin{pmatrix} 0 & 0 \end{pmatrix}$$

$$
\begin{array}{l}
\text{If } a \neq 0, \\
\qquad \xrightarrow{\text{line1 } \times = \left(\frac{1}{a^2}\right)} (0 \quad 1) \\
\text{If } b = -a^2, \\
\quad (0 \quad -a^2) \\
\text{If } a = 0, \\
\quad (0 \quad 0) \\
\text{If } a \neq 0, \\
\qquad \xrightarrow{\text{line1 } \times = \left(\frac{-1}{a^2}\right)} (0 \quad 1) \\
\text{If } a^4 - b^2 \neq 0, \\
\qquad \xrightarrow{\text{line1 } \times = \left(\frac{1}{a^4-b^2}\right)} \left(1 \quad \frac{b}{a^4-b^2}\right)
\end{array}
$$

```
[19] os_md.mtoupper(mat([a^4+b^2,b]),0|step=1,opt=6,dviout=1)$
```

$$
\begin{array}{l}
\left(a^4+b^2 \quad b\right) \\
\text{Assume } a^4+b^2 \neq 0, \\
\qquad \xrightarrow{\text{line1 } \times = \left(\frac{1}{a^4+b^2}\right)} \left(1 \quad \frac{b}{a^4+b^2}\right)
\end{array}
$$

256. `mytrace(`m`)`
:: 行列の trace を返す.
257. `mydet(`m`)`
:: `det(`m`)` と同じ．ただし成分は有理式でもよい.
258. `mydet2(`m`)`
:: `det(`m`)` と同じ．ただし成分は有理式でもよい.
`mydet()` は `det()` を，また `mydet2()` は `mtoupper()` を利用する.

```
[0] M=newmat(3,3,[[1,x,x^2],[1,y,y^2],[1,z,z^2]]);
[ 1 x x^2 ]
[ 1 y y^2 ]
[ 1 z z^2 ]
[1] fctr(os_md.mydet2(M));
[[-1,1],[y-z,1],[x-z,1],[x-y,1]]
[2] N=newmat(3,3,[[1,x/y,x^2/y^2],[1,y/z,y^2/z^2],[1,z/x,z^2/x^2]]);
[ 1 (x)/(y) (x^2)/(y^2) ]
[ 1 (y)/(z) (y^2)/(z^2) ]
[ 1 (z)/(x) (z^2)/(x^2) ]
[3] os_md.fctrtos(os_md.mydet2(N));
-(z*x-y^2)*(y*x-z^2)*(x^2-z*y)/(z^2*y^2*x^2)
[4] det(N);
internal error (SEGV)
return to toplevel
```

259. `myrank(`m`)`
:: 行列 m の rank を求める（例は `mtoupper()` の項）

260. mykernel(m|opt=1)
:: 行列 m の転置行列の kernel の基底を求める
オプション opt=1 を指定したときは，転置しない行列とする．

```
[0] A=newmat(4,3,[[x],[y,1,1],[0,1,1],[1,1,1]]);
[ x 0 0 ]
[ y 1 1 ]
[ 0 1 1 ]
[ 1 1 1 ]
[1] os_md.mykernel(A);
[[ y -x x 0 ],[ y-1 -x 0 x ]]
```

261. myimage(m|opt=1)
:: 行列 m の転置行列の像の基底を求める
- オプション opt=1 を指定したときは，転置しない像とする．
- 基底のベクトルの最初の 0 でない成分の位置は狭義単調増加で，他のベクトルのその位置の成分は 0 となっている．

```
[0] A=newmat(4,3,[[x,-x],[y,1-y,1],[0,1,1],[0,1,2]]);
[ x -x 0 ]
[ y -y+1 1 ]
[ 0 1 1 ]
[ 0 1 2 ]
[1] Im=os_md.myimage(A|opt=1);
[[ x 0 -y 0 ],[ 0 1 1 0 ],[ 0 0 0 1 ]]
[2] V=newvect(4,[1,1,1,1]);
[ 1 1 1 1 ]
[3] os_md.mymod(V,Im);
[ 0 0 (y)/(x) 0 ]
[4] os_md.mymod(V,Im|opt=1);
1
[5] os_md.mymod(V,Im|opt=2);
[ (y)/(x) ]
```

262. mymod(v,[$v_1,\ldots,v_k$]|opt=ℓ)
:: ベクトル $v_1,\ldots,v_k$ で張られる空間の商空間への v の射影を求める（例は myimage() の項）
- $v_1,\ldots,v_k$ の 0 でない要素の位置は，順に真に増加している必要がある．
- オプション opt=1 を指定した場合，射影が 0 のとき 0 を，それ以外では 1 を返す．
- opt=2 を指定したときは，商空間の標準基底を取った射影を表す．

263. mmod(m,[$v_1,\ldots,v_k$]|opt=1)
:: ベクトル $v_1,\ldots,v_k$ で張られる空間の商空間への線形写像 m の射影
- $v_1,\ldots,v_k$ の 0 でない要素の位置は，順に真に増加している必要がある．また，opt=1 を指定しないときは，これらで張られる空間は m で不変でなければならない．
- opt=1 のときは，商空間上の線形変換でなくて，商空間への線形写像とみなす（このときは m は正方行列でなくてよい）．

264. myinv(m)
:: 正方行列 m の逆行列を求める
成分は有理式でもよい．m が可逆でないとき，0 を返す．

```
[0] M=newmat(2,2,[[a,b],[c,d]]);
[ a b ]
[ c d ]
[1] os_md.myinv(M);
[ (d)/(d*a-c*b) (-b)/(d*a-c*b) ]
[ (-c)/(d*a-c*b) (a)/(d*a-c*b) ]
```

265. madj(g,m)
:: 行列 gmg^{-1} を計算する．m は行列のリストか行列のベクトルでもよい．
266. diagm(m,a)
:: 対角行列を作成する
mgen(m,0,a,1) と同じ
267. mgen(m,n,a,s|sep=1)
:: size $m \times n$ の一般行列を作成する
- (i,j)-成分は a が変数の時 $ai'j'$ で，リストの時 $a[i]j'$ で与えられる（$i' = i+s$, $j' = j+s$）
- s は 0 または 1 で，成分の index は s から始まるものとする．
- a や $a[i]$ は文字列でもよい．また，以下の 3 つの場合（$n = 0, -1, -2$）のときは数字でもよい．
- $n = 0$ または diag のときは size m の対角行列を返す．
- $n = -1$ または highdiag のときは対角成分の一つ上の成分のみ 0 でない size m の正方行列を返す．
- $n = -2$ または lowdiag のときは対角成分の一つ下の成分のみ 0 でない size m の正方行列を返す．
- $n = -3$ または skew のときは size m の歪対称正方行列を返す．
- $n = -4$ または symmetric のときは size m の対称正方行列を返す．
- $n = -5$ または perm のときは size m の置換行列を返す．この行列 G で縦ベクトルを変換すると，i 番目に a_i 番目の成分が入る．
- sep=1 を指定すると，二重インデックスの間に _ を入れる．
- リストの要素の個数が足りないときは，最後の要素がそれ以降も続いているとみなされる．

mgen(0,0,0,0) で簡単な仕様が表示される．

```
[0] os_md.mgen(2,2,c,0);
[ c00 c01 ]
[ c10 c11 ]
[1] os_md.mgen(2,3,[a,b],1);
[ a1 a2 a3 ]
[ b1 b2 b3 ]
[3] os_md.mgen(3,"diag","a_",1);
[ a_1 0 0 ]
[ 0 a_2 0 ]
[ 0 0 a_3 ]
[4] os_md.mgen(2,"diag",[a,b],0);
[ a 0 ]
[ 0 b ]
[5] os_md.mgen(3,"diag",[1,2,3],0);
[ 1 0 0 ]
[ 0 2 0 ]
[ 0 0 3 ]
[6] os_md.mgen(3,"diag",[x],0);
```

```
[ x 0 0 ]
[ 0 x 0 ]
[ 0 0 x ]
[7] os_md.mgen(3,"highdiag","e_", 1);
[ 0 e_1 0 ]
[ 0 0 e_2 ]
[ 0 0 0 ]
[8] os_md.mgen(3,"skew",[a,b,c],0);
[ 0 a1 a2 ]
[ -a1 0 b2 ]
[ -a2 -b2 0 ]
[9] os_md.mgen(3,"symmetric",[a,b,c],1);
[ a1 a2 a3 ]
[ a2 b2 b3 ]
[ a3 b3 c3 ]
[10] os_md.mgen(4,"diag",[a],0)+os_md.mgen(4,"lowdiag",[1],0);
[ a 0 0 0 ]
[ 1 a 0 0 ]
[ 0 1 a 0 ]
[ 0 0 1 a ]
[10] os_md.mgen(3,"perm",[3,1,2],1);
[ 0 0 1 ]
[ 1 0 0 ]
[ 0 1 0 ]
```

268. `unim(`s`|abs=`m`,num=`n`,both=1,int=1,rank=`r`,conj=1,res=1,wt=`w`,dviout=`t`,lim=`w`)`
:: 行列式 1 でサイズ s の整数行列をランダムに生成する，行列 s をランダムに変換する
- s が正整数の時は，サイズ s で行列式が 1 の行列をランダムに生成する．
- s が正整数で `conj=1` を指定したときは，サイズ s の行列とその Jordan 標準型の組をランダムに生成する．
  - `wt=`w によって固有値の広がり（サイズ $+w$）を指定できる（$w=2$ がデフォルト）．
  - `diag=1` によって，対角化可能行列に制限する．
  - `res=1` によって，変換行列とその逆行列も返す．すなわち戻り値が $[A,B,P,Q]$ ならば $P=Q^{-1}$, $B=PAQ$ で，B が A の Jordan 標準形．
    - `int=1` を指定すると自明な部分があって不適切な問題が減らされる．
    - `int=2` を指定すると，上に加えて整数行列 Q がユニモジュラーではない例も含める（このとき P の成分は整数でない）．ただし，`int` の値をより大きくすると，その例の割合は少なくなる．
    - `dviout=1` を指定すると，TEX を使って結果を画面表示する．
    - `dviout=-1` を指定すると，TEX のソースを返す．
- s が行列の時は，ある行の整数倍を別の行に加える変換を何度かランダムに行って返す．
- s が行列で `both=1` を指定したときは，列に関しても同様な変換を行う．
- s が正方行列で `conj=1` を指定したときは，与えられた行列に共役な行列をランダムに返す．
- s が `[`p`,`q`]` という正整数のリストの時は，行基本変形のための $p\times q$ 行列をランダムに生成する．行列の階数が 2 以上で p と q を越えないものが，各階数に対してほぼ等確率で得られる．
- s が `[`p`,`q`,`k_1`,`k_2`,...]` というリストの時は，ある行の k_ν 列目が先頭の 1 となる簡約行列に行基本変形で変換される $p\times q$ 行列をランダムに生成する．

- s がリストの時，rank=r で行列の階数を指定する．
- s がリストの時，int=1 で行基本変形による簡約行列が整数行列になるものに制限する（一般の場合も分母は 1 桁の整数に抑えられている）．
- s がリストの時，生成される行列の 1 列目と 2 列目が線形独立となる確率は高めに設定してあるが，wt=w によってそれを変更する（w は非負整数で，デフォルトは $w = 2$ である．この値が大きいほど確率は高くなる）．
- s がリストの時，res=1 である行の整数倍を別の行に加える変換で得られる上三角行列も返す．
  - さらに dviout=1 を指定すると，TEX を使って基本変形の過程を含めて結果を画面表示する．このとき mtoupper() における lim=w の指定が可能．
  - 前項で dviout=1 と指定した場合は，TEX のソースを返す．
- 現れる行列の成分の絶対値が許される範囲内に留まる整数行列内の変形のみが使われる．
- abs=m：成分の絶対値の最大値を m 以下とする（デフォルトは $m = 9$）．
- num=n：ランダム化の程度（$n = 100$ または 200 がデフォルト）．

```
[0] os_md.unim(4);                  /* 行列式 1 の 4x4 行列 */
[ 3 -2 -8 3 ]
[ -7 -7 -6 7 ]
[ 8 7 5 -7 ]
[ -3 1 6 -2 ]
[1] os_md.unim([4,5]);              /* 行基本変形用 4x5 行列 */
[ 2 -2 2 6 9 ]
[ 3 2 -7 -1 -1 ]
[ -6 -1 8 -4 -2 ]
[ 0 4 -8 -8 -5 ]
[2] os_md.unim([3,3]|rank=2);       /* 階数が 2 の行基本変形用 3x3 行列 */
[ -9 7 5 ]
[ 5 -4 -3 ]
[ 7 1 9 ]
[3] os_md.unim([3,3]|rank=2,abs=99);  /* 階数が 2 の行基本変形用 3x3 行列 */
[ 28 -55 24 ]
[ -15 49 -91 ]
[ -33 64 -25 ]
[4] os_md.unim([4,5]|abs=3,res=1); /* 成分の絶対値が 3 以下 + 基本変形結果 */
[[ -1 1 3 -3 3 ]
[ -2 -1 0 1 3 ]
[ -3 -1 1 -1 2 ]
[ 2 1 0 1 1 ],[ 1 0 -1 0 0 ]
[ 0 -1 -2 1 0 ]
[ 0 0 0 2 0 ]
[ 0 0 0 0 -1 ]]
[5] os_md.unim([4,5,0,1,3]);
    /* 0,1,3 列目が先頭の 1 となる行をもつ簡約型に変換できる行列 */
[ 4 5 -6 2 -7 ]
[ -4 -6 8 1 -1 ]
[ 3 6 -9 -2 3 ]
[ 6 5 -4 -1 4 ]
```

```
[6] os_md.unim(os_md.diagm(3,[0,-1,1])|conj=1); /* 固有値が 0,-1,1 の行列 */
[ -4 -5 -8 ]
[ -6 -1 -6 ]
[ 7 -1 5 ]
[7] A=os_md.mgen(3,"highdiag",[1,0],0)+os_md.diagm(3,[0,0,1]);
[ 0 1 0 ]
[ 0 0 0 ]
[ 0 0 1 ]
[8] os_md.unim(A|conj=1);           /* A と共役な行列 */
[ 3 -3 1 ]
[ 5 -1 -3 ]
[ 4 -2 -1 ]
[9] os_md.unim(4|conj=1);           /* 正方行列と Jordan 標準形の組 */
[[ 8 9 -1 -9 ]
 [ -5 -5 6 4 ]
 [ 2 2 -1 -2 ]
 [ 2 3 5 -4 ],
 [ -1 1 0 0 ]
 [ 0 -1 1 0 ]
 [ 0 0 -1 0 ]
 [ 0 0 0 1 ]]
[10] for(I=0,M=[];I<100;I++) M=cons(os_md.unim([3,4]),M)$
[11] for(N=[];M!=[];M=cdr(M)) N=cons([length(os_md.mtoupper(car(M),0|step=1,
opt=7,dviout=-2)),car(M)],N)$
[12] for(M=qsort(N);M!=[];M=cdr(M)) os_md.mtoupper(car(M)[1],0|step=1,opt=7,
dviout=2,lim=80)$          /* 3x4 行列の行基本変形の問題 100 題 */
[13] dviout(" ");
[14] for(I=0,R=[];I<100;I++) R=cons(os_md.unim(4|conj=1,res=1,int=1),R);
[15] S="&"+os_md.ltotex(R|opt="cr",cr=15)$
[16] dviout(S|eq="align")$  /* 4 次正方行列の Jordan 標準形と変換行列 100 題 */
```

上の [15] 以降を以下のようにしてもよい.

```
[15] for(T=[],S=R;S!=[];S=cdr(S))
      T=cons([S[0][1],"&=",S[0][2],S[0][0],S[0][3]],T);
[16] S=os_md.ltotex(T|opt=["cr","spts0"],str=1,cr=11)$
[17] dviout(S|eq="align")$  /* 4 次正方行列の Jordan 標準形と変換行列 100 題 */
```

上の [17] は，以下のようになる.

$$\begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 3 \end{pmatrix} = \begin{pmatrix} 0 & -1 & 1 & 0 \\ -1 & 5 & -3 & -1 \\ 1 & -2 & 1 & 1 \\ 0 & 1 & -1 & -1 \end{pmatrix} \begin{pmatrix} 5 & 0 & -4 & 0 \\ 2 & 2 & -3 & 1 \\ 1 & 8 & -7 & 0 \\ 1 & -9 & 7 & 4 \end{pmatrix} \begin{pmatrix} 2 & 1 & 2 & 1 \\ 2 & 1 & 1 & 0 \\ 3 & 1 & 1 & 0 \\ -1 & 0 & 0 & -1 \end{pmatrix}$$

$$\begin{pmatrix} -2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix} = \begin{pmatrix} 0 & -2 & -1 & 1 \\ 1 & 1 & 0 & -1 \\ 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1 & 6 & 5 & -1 \\ -4 & -1 & 0 & 4 \\ 4 & 4 & 3 & -4 \\ -4 & 6 & 5 & 2 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

```
[18] os_md.unim(4|int=2,conj=1,res=1,dviout=1);
```

$$\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix} = \begin{pmatrix} \frac{5}{2} & -\frac{5}{4} & \frac{1}{4} & -\frac{3}{4} \\ -5 & 2 & -1 & 2 \\ 2 & 1 & 1 & -\frac{3}{2} \\ \frac{1}{2} & -\frac{1}{4} & \frac{1}{4} & -\frac{1}{4} \end{pmatrix} \begin{pmatrix} -6 & 5 & 2 & 0 \\ -4 & 4 & 1 & 0 \\ 6 & -1 & 8 & -6 \\ -8 & 8 & 8 & -3 \end{pmatrix} \begin{pmatrix} 5 & 3 & 1 & 3 \\ 3 & 2 & 1 & 1 \\ 5 & 4 & 1 & 11 \\ 12 & 8 & 2 & 12 \end{pmatrix}$$

269. `newbmat(`m`,`n`,[[`$r_{00}, r_{01}, \ldots$`],[`$r_{10}, \ldots$`],...]|null=`t`)`

:: ブロック行列を作成する

- r_{ij} の行または列のサイズが 1 のときは，ベクトルまたはリストで表してもよい.
- r_{ij} がサイズ 1 の正方行列のときは，スカラーで表してもよい.
- r_{ij} が 0 行列の時は 0 で表してもよい.
  このとき `null=`t が指定してあれば，0 行列でなく，各成分が t の行列と解釈される.
- ブロックに分けた各行や各列には少なくとも 1 つの行列があればよい.

```
[0] A=os_md.newbmat(2,2,[[a,b],[c,d]]);
[ a b ]
[ c d ]
[1] os_md.newbmat(2,2,[[0,A],[2*A]]);
[ 0 0 a b ]
[ 0 0 c d ]
[ 2*a 2*b 0 0 ]
[ 2*c 2*d 0 0 ]
[2] os_md.newbmat(2,2,[[A],[0,1]]);
[ a b 0 ]
[ c d 0 ]
[ 0 0 1 ]
[3] os_md.newbmat(2,3,[[A]]);
[ a b 0 0 ]
[ c d 0 0 ]
[ 0 0 0 0 ]
[4] os_md.newbmat(1,2,[[A,[e,f]]]);
[ a b e ]
[ c d f ]
[5] os_md.newbmat(2,2,[[A],[0,0]]|null="A");
[ a b A ]
[ c d A ]
[ A A A ]
[6] os_md.newbmat(2,2,[[A],[0,[0]]]|null="A");
[ a b A ]
[ c d A ]
[ A A 0 ]
```

270. `meigen(`m`|mult=1)`

:: 行列 m の固有値を返す

- `mult=1` のときはスペクトル型の形で，重複度が大きい順に返す.
- m は行列のリストやベクトルでもよい.
- 有理式でない固有値は，`zz` の多項式の根で表される.

- unim() によって固有値計算練習用の行列の生成ができる.

```
[0] A=newmat(3,3,[[1,-1,-1],[4,6,5],[-2,-2,-1]]);
[ 1 -1 -1 ]
[ 4 6 5 ]
[ -2 -2 -1 ]
[1] os_md.meigen(A);
[1,2,3]
[2] B=newmat(3,3,[[1,-1,-1],[4,6,5],[-2,-2,1]]);
[3] os_md.meigen(B);
[zz^2-6*zz+13,2]
[4] C=newmat(3,3,[[a],[0,b,2],[0,0,a]]);
[ a 0 0 ]
[ 0 b 2 ]
[ 0 0 a ]
[5] os_md.meigen(A);
[a,a,b]
[6] os_md.meigen(A|mult=1);
[[2,a],[1,b]]
[7] os_md.meigen([A,C]);
[[1,2,3],[a,a,b]]
[8] os_md.meigen([A,C]|mult=1);
[[[1,1],[1,2],[1,3]],[[2,a],[1,b]]]
```

271. mdsimplify(m|show=1,type=ℓ)
:: 有理式正方行列 m，またはそのリストやベクトルを対角行列で簡単化
- 元の行列は保たれない.
- show=1 を指定すると，用いる対角行列も示す.
- type=1 とすると，行と列の分母を列と行に変換する.
- type=2 (resp. 3) のときは行（resp. 列）の分母を列（resp. 行）に変換する.

```
[0] A=newmat(2,2,[[a,b/c^2],[c/b,d]]);
[ a (b)/(c^2) ]
[ (c)/(b) d ]
[1] os_md.mdsimplify(A|show=1);
[[ a (1)/(c) ]
[ 1 d ],[ (c)/(b) 1 ]]
```

272. transm(m|dvioout=1)
:: 行列 m の基本変形をインタラクティブに行う
- dvioout=1 を指定すると，TEX を用いて結果の画面表示ができる.
- m は s2m() の引数の形式でもよい.

表示される ? の後に 1 行キー入力して指示することによって基本変形する．例えば
空　　：コマンドの表示
2,5　　：2 行目と 5 行目の入れ替え
2,5,-2 ：2 行目に 5 行目の -2 倍を加える
2,2,-2 ：2 行目を -2 倍する
2,5,0　：2 行目に 5 行目のスカラー倍を加えて，2 行目における 5 行目の先頭の 0 でない列を 0 に

r,2,5 ：2 列目と 5 列目の入れ替え（列変形を行うには先頭に r をつける）
s,x,2 ：x に 2 を代入する
t ：転置する
0 ：最初の行列に戻る
f ：一つ前の行列に戻る
g ：次の行列へ（f の直後のみ可能）
a ：以降，自動的に行基本変形を行う
A ：上と同じだが dviout=1 のときは，TEX を用いた画面表示となる
q ：終わり（変形の過程の行列をリストにして返す）

```
[0] L=os_md.transm(mat([2,3,1,-1,a],[3,2,2,1,1],[4,0,3,4,1],[5,4,4,1,1]))$
[ 2 3 1 -1 a ]
[ 3 2 2 1 1 ]
[ 4 0 3 4 1 ]
[ 5 4 4 1 1 ]
?
2,5    : line2 <-> line5
2,5,-2 ; line2 += (-2)*line5
2,2,-2 : line2 *= -2
2,5,0  : line2 += (?)*line5 for reduction
....
? 3,2,-1
[ 2 3 1 -1 a ]
[ 3 2 2 1 1 ]
[ 1 -2 1 3 0 ]
[ 5 4 4 1 1 ]
? 1,3
[ 1 -2 1 3 0 ]
[ 3 2 2 1 1 ]
[ 2 3 1 -1 a ]
[ 5 4 4 1 1 ]
? 2,1,0
[ 1 -2 1 3 0 ]
[ 0 8 -1 -8 1 ]
[ 2 3 1 -1 a ]
[ 5 4 4 1 1 ]
? f
[ 1 -2 1 3 0 ]
[ 3 2 2 1 1 ]
[ 2 3 1 -1 a ]
[ 5 4 4 1 1 ]
? s,a,1
[ 2 3 1 -1 1 ]
[ 3 2 2 1 1 ]
[ 4 0 3 4 1 ]
```

```
[ 5 4 4 1 1 ]
? q
[1]
```

3.2.7 Strings

273. `str_char(s,`n`,`t`)`
:: 文字列 s の n 文字以降に文字列 t の先頭文字が最初に現れる場所を返す（`str_chr()` の拡張）
- 見つからないときは -1 を返す.
- s は，（たとえば `strtoascii()` で得られる整数の）リストやベクトルなどでもよい.
- s がリストやベクトルの時は，t は文字列以外（たとえば文字コード）でもよい.
- s がリストまたはベクトルの場合，型返還せずにそのまま扱うのでメモリー効率がよい.
- `str_str()` の方が高機能.

274. `str_pair(s,`n`,`t_1`,`t_2`|inv=1)`
:: 文字列 s の n 文字以降で，文字（列）t_2 の出現回数が t_1 の出現回数を超える最初の位置を返す
- 見つからないときは -1 を返す.
- s は，（たとえば `strtoascii()` で得られる整数の）リストやベクトルなどでもよい.
- t_1, t_2 は文字コードでもよい．あるいは文字列でもよい.
- `sjis=1` が指定されると，シフト JIS の文字列とみなして処理する.
- `inv=1` が指定されると，文字列 s の n 文字以前で，文字（列）t_1 の出現回数が t_2 の出現回数を超える最後の位置を返す．ただし，`sjis=1` の指定は無視される.
- s がリストまたはベクトルの場合，型変換せずにそのまま扱うのでメモリー効率がよい.

```
[0] os_md.str_pair("(1+((2+3)*5)/4)(6+",1,"(",")");
14
[1] os_md.str_cut("(1+((2+3)*5)/4)(6+",0,14);
(1+((2+3)*5)/4)
[2] os_md.str_pair("(1+((2+3)*5)/4)(6+",13,"(",")"|inv=1);
0
[3] os_md.str_pair("(1+((2+3)*5)/4)(6+",9,"(",")"|inv=1);
3
```

275. `str_str(`s`,`t`|top=`n`,end=`m`,sjis=1)`
:: 文字列 s に部分文字列 t が最初に現れる場所を返す
- 部分文字列 t が文字列 s に含まれないとき，-1 を返す.
- `top=`n が指定されると，最初の n 文字より後を探す（$n=0$ がデフォルト）.
- `end=`m が指定されると，最初から $m+1$ 文字を除いた後から現れるものは探さない.
- `sjis=1` が指定されると，シフト JIS の文字列とみなして処理する.
- s および t は（`strtoascii()` によって変換されたような）文字コードのリスト，あるいはベクトルでもよい.
- t の長さが 1 のときは文字コードでもよい.
- s が文字コードのリストまたはベクトルの場合，型変換しないのでメモリー効率がよくて高速.
- t がリストで，その成分が文字列（あるいはそれを文字コードのリストやベクトルとしたもの）であった場合, t の何番目（最初が 0 番目と数える）の文字列が s に最初に現れるかと，その位置の組とのリストを返す．いずれもが現れない場合は $[-1,-1]$ を返す.
  - t には頻度の多いものを先に並べる方が効率がよい.
  - 同じ位置から t の複数の文字列が現れた場合, 先に並べたものを返す．すなわち $s=$`"aabc..."` に対して $t=$`["abc","ab"]` とした場合，文字列 `"abc"` が見つけられる.

```
[0] os_md.str_str("abcdefghiabc", "bc");
```

```
1
[1] os_md.str_str("abcdefghiabc", "bc"|top=2);
10
[2] os_md.str_str("abcdefghiabc", "ac");
-1
[3] os_md.str_str("abcdefghiabc", ["ab","gh"]|top=1);
[1,6]
[4] os_md.str_str("abcdefghiabc", ["ab","gh"]|top=1,end=5);
[-1,0]
[5] os_md.str_str("abcdefghiabc", ["bcd","bc"]);
[0,1]
[6] os_md.str_str("abcdefghiabc", ["bcd","bc"]|top=2);
[1,10]
```

276. `str_cut(`s`,`m`,`n`)`
:: 文字列 s の先頭から m 文字をスキップして，それ以降 $n-m+1$ 文字を返す
- s が文字列のときは `sub_str(`s`,`m`,`n`)` と同じ（例は `str_pair()` を参照）.
- s は文字コードのリストやベクトルでもよい（こちらの方がオーバーヘッドが少ない）．ただし，戻り値は文字列.
- n が十分大きいときは，s の $m+1$ 文字目以降を返す.

277. `str_subst(`s`,`s_0`,`s_1`|sjis=1,raw=1)` または
`str_subst(`s`,[`$s_{00}, s_{01}, \ldots$`],[`$s_{10}, s_{11}, \ldots$`]|inv=1,sjis=1,raw=1)`
`str_subst(`s`,[[`s_{00}, s_{10}`],[`s_{01}, s_{11}`],...],0|sjis=1)`
:: 文字列 s に含まれる部分文字列 s_0 を s_1 で全て先頭から順に置き換える
- 複数種の置き換え，すなわち部分文字列 s_{0j} を s_{1j} へ $j=0,1,\ldots$ の順に s の置き換えができる.
- 複数種置き換えで，置き換えられた部分文字列の再置き換えは行わない.
- s は `strtoascii()` によって置き換えられた整数のリストやベクトルでもよい（こちらの方がオーバーヘッドが少ない．ただし，`raw=1` を設定しないと，戻り値は文字列.
- `inv=1` によって，逆に s_{1j} を s_{0j} へ $j=0,1,\ldots$ の順の置き換えが行われる.
- `sjis=1` が指定されると，シフト JIS の文字列とみなして処理する.
- `raw=1` が指定されていると，`strtoascii()` で置き換えられた整数のリストを返す（こちらの方がオーバーヘッドが少ない）.

```
[0] os_md.str_subst("abc", ["a","b"],["b","a"]);
bac
[1] os_md.str_subst("abcdefghi", ["bc","cd","ef"],["cd","ab","ef"]);
acddefghi
[2] os_md.str_subst("abcdefghi", [["bc","cd"],["cd","ab"],["de","ef"]],0);
acdeffghi
```

278. `str_times(`s`,`n`)`
:: 文字列 s（またはリスト）を n 回繰り返した文字列（またはリスト）を返す
- リストのときは，$s=$`[[`$s_1, s_2, \ldots, s_k$`]]` とすると，`[`$s_1, s_2, \ldots, s_k, s_1, s_2, \ldots, s_k, s_1, s_2, \ldots$`]` という長さ n のリストを返す.
- $s=$`[`$t_1, \ldots, t_k$`,[`$s_1, s_2, \ldots, s_k$`]]` のときは，`[`$t_1, \ldots, t_k, s_1, s_2, \ldots, s_k, s_1, s_2, \ldots, s_k, s_1, s_2, \ldots$`]` という長さ n のリストを返す.

```
[0] os_md.str_times("Abc",5);
AbcAbcAbcAbcAbc
```

```
[1] os_md.str_times([["Abc"]],5);
[Abc,Abc,Abc,Abc,Abc]
[2] os_md.str_times([["Abc","Def"]],5)
[Abc,Def,Abc,Def,Abc]
[3] os_md.str_times(["012",["Abc","Def"]],6);
[012,Abc,Def,Abc,Def,Abc]
```

279. strip(s,t_1,t_2)
:: 文字列の外側の括弧を外す
- s が t_1 と t_2 のペアの括弧で全体が囲まれているとき，その括弧を外した文字列を返す.

```
[0] os_md.strip("((x+y)^2+(y+z))","(",")");
(x+y)^2+(y+z)
[1] os_md.strip("(x+y)^2+(y+z)","(",")");
(x+y)^2+(y+z)
```

280. s2os(s)
:: 文字列を入力可能な文字列に変換

```
[0] S="\"表\計算\"";
"表計算"
[1] os_md.s2os(S);
\"表\計算\"
```

281. sjis2jis(l)
:: 文字コードの整数のリストの先頭 2 つを ShiftJIS から JIS に変換
変換後の長さ 2 の整数のリストを返す
282. jis2sjis(l)
:: 文字コードの数字のリストの先頭 2 つを JIS から ShiftJIS に変換
変換後の長さ 2 の整数のリストを返す

```
[0] P="整数"$
[1] L=strtoascii(P);
[144,174,144,148]
[2] i2hex(L);
[90,ae,90,94]
[3] LL=os_md.sjis2jis(L);
[64,48]
[4] os_md.jis2sjis(LL);
[144,174]
```

283. s2euc(s)
:: ShiftJIS または JIS 文字列を EUC 文字列に変換
0x0d のコードは削除して 0x0a のコードに変換する.
284. s2sjis(s)
:: EUC または JIS 文字列を ShiftJIS 文字列に変換
0x0a のコードは 0x0d 0x0a のコードに変換する.
285. str_tb([$s_0, s_1, \ldots$],tb) または str_tb(0,tb)
:: テキスト用バッファを作成（$tb = 0$）し，そこ（戻り値 tb）に文字列 $s_0, \ldots$ を順に追加する．最初の引数を 0 として文字列を取り出す.

- $tb = 0$ のときは，tb は空文字列と解釈される．
- tb が文字列ならばそれで初期化されたテキスト用バッファを作成し，さらに第 1 引数で指定された文字列を順に追加したテキスト用バッファを返す．ただしここで第 1 引数が 0 のときは，それは空文字列と解釈される．
- tb がテキスト用バッファの場合は，第 1 引数が文字列またはそのリストのとき，それが順にバッファに追加される．ただし第 1 引数が 0 のときはテキスト用バッファの文字列を取り出して返す．
- 文字列が一つなら，[s_0] は単に s_0 でよい．
- この函数は，細かく文字列を追加していった場合のメモリー消費の非効率性を避けるために用いられる．また，str_len(tb) でバッファに格納された文字列サイズが分かる．

```
[0] Buf=os_md.str_tb("This is ",0);
This is
[1] os_md.str_tb("a pen.",Buf)$
[2] str_len(Buf);
14
[3] Str=os_md.str_tb(0,Buf);
This is a pen.
[5] Buf=os_md.str_tb(" a pen.","This is");
This is a pen.
[6] type(Buf);
21
[7] type(Str);
7
[8] Buf=os_md.str_tb(0,0);

[9] X=3$
[10] os_md.str_tb([rtostr(X),"^2=",rtostr(X^2)],Buf);
3^2=9
```

286. l2os(l)
:: リストを入力可能な文字列に変換
- 行列やベクトルはリストに直す．これらはネスティング可能．
- 成分の文字列は " で囲って表示する．

```
[0] P=mat([x+y,"This is"],[(x+y)^2,"\"square\""]);
[ x+y This is ]
[ x^2+2*y*x+y^2 "square" ]
[1] R=os_md.l2os(P);
[[x+y, "This is"], [x^2+2*y*x+y^2, "\"square\""]]
[2] eval_str(R);
[[x+y,This is],[x^2+2*y*x+y^2,"square"]]
```

287. r2os(l)
:: 数式を eval_str() で入力可能な文字列に変換
- 行列，ベクトル，リストなども許される．これらはネスティング可能．
- 文字列は " で囲って表示する．

```
[0] P=mat([x+y,"This is"],[(x+y)^2,"\"square\""]);
```

```
[ x+y This is ]
[ x^2+2*y*x+y^2 "square" ]
[1] R=os_md.r2os(P);
mat([x+y,"This is"],[x^2+2*y*x+y^2,"\"square\""])

[2] eval_str(R);
[ x+y This is ]
[ x^2+2*y*x+y^2 "square" ]
```

288. r2ma(s)
:: Mathematica の形式の数式に変換
evalma(s|inv=1) と同じ

289. evalma(s|inv=1)
:: Mathematica の数式を読み込む
- Mathematica の函数 Sin, Cos, Tan, ArcSin, ArcCos, ArcTan, Exp, Log, Sinh, Cosh, Tanh に対応している.
- $s=0$ とすると，Mathematica 形式の文字列を（コピー・ペーストなどにより）直接入力できる. 複数行に渡ってもよい. また入力終了は文字 ; の後の改行で示す.
- inv=1 とすると，逆変換，すなわち数式から Mathematica 形式の文字列に変換する.

```
[0] P=os_md.evalma(0)$
Mathematica text (terminated by ;) ?
{(x+y)^2,
y}
;
[1] P;
[ x^2+2*y*x+y^2 y ]
[2] os_md.r2ma(P);
{x^2+2*y*x+y^2,y}
[3] os_md.evalma(0)$
Mathematica text (terminated by ;) ?
{x+y)^2,
y};
exprparse : syntax error
[4] os_md.evalma("{{1,2},{3,4}};");
[ 1 2 ]
[ 3 4 ]
[5] os_md.my_tex_form(@@);
\begin{pmatrix}
 1&2 \\
 3&4 \\
\end{pmatrix}
```

290. readcsv(s|eval=[$n_1,n_2,\ldots$],eval=n,sp=1,col=c,null=t,eq=1)
:: CSV 形式のデータをリスト形式で読み込む
- CSV 形式とは，複数のデータをコンマと改行で区切ったデータ形式であるが，それを 1 行のデータをリストにし，さらにそのリストを行順に並べたリストとして読み込む.

- 通常は文字列として読み込むが，各行で $n_1, n_2, \ldots$ で指定した項目は式として読み込む（n_1 は n_1 番目の項目 – 最初は 1 番目 – を表す）．
  n 項目のみの指定のときは，eval=n としてもよい．
- eval="all"というオプションでは，可能な項目は全て（小）数として読み込む．
- sp=1 というオプションでは，空白やタブやそれらが連続したものをデータの区切りとみなす．
- col=c というオプションでは，各行の第 c 項目のみを読んだリストとする．
- null=t というオプションでは，データの無い項目を t とする（デフォルトは空文字列）．
- eq=1 というオプションは，数でなくて式として読み込み，t のデフォルトは 0.
- CSV 形式では，データにコンマを含む場合は両端を ” で囲って表す．さらにその中に ” があれば，それを二つ重ねて一つの ” を表す．
- ファイルが見つからない場合は，負の数字を返す．
- MS Windows のときは，ShiftJIS の文字列に対応している．

```
[0] os_md.showbyshell("type z:\\test.txt")$
12,abc,(x+y)^2
0,2/3,2/4
2,"a ""b",4/4
[1] os_md.readcsv("z:\\test.txt");
[[12,abc,(x+y)^2],[0,2/3,2/4],[2,a "b,4/4]]
[2] os_md.readcsv("z:\\test.txt"|eval=[3],eq=1);
[[12,abc,x^2+2*y*x+y^2],[0,2/3,1/2],[2,a "b,1]]
[3] os_md.readcsv("z:\\test.txt"|col=3,eval=3,eq=1);
[x^2+2*y*x+y^2,1/2,1]
```

291. tocsv(l|null=n,exe=f)
:: リストのリストまたは行列を CSV 形式のデータに変換
- null=t というオプションでは，t というデータを空の項目とみなす．
- exe=0, 1 : DIROUT で指定したディレクトリの risaout.csv というファイルに（存在すれば 0 では追加，1 では上書き）出力し，関連づけされた Excel を呼び出す．
- exe=2,...,9 : DIROUT で指定したディレクトリの risaout2.csv,...,risaout9.csv というファイルに上書き出力し，関連づけされた Excel を呼び出す．
- exe=f : f が拡張子.csv というファイル名のときは，そのファイル名のファイルとして出力し，関連づけされた Excel を呼び出す．

3.2.8 Permutations

292. ldict(n,m|opt=t)
:: $\{0, 1, 2, \ldots, m-1\}$ を並べ替えて辞書式順序で $n+1$ 番目のリストを返す
存在しないときは 0 を返す．
ただし，数字を逆に並べて大きい順の辞書式順序とする．
- $m = 0$ のときは $m = \infty$ とみなし，後ろの並べ替えない部分を省いたリストを返す．
- $m > 0$ でオプション opt を指定した場合の意味は，次項の ndict() を参照．

```
[0] os_md.ldict(0,3);
[0,1,2]
[1] os_md.ldict(1,3);
[1,0,2]
[2] os_md.ldict(2,3);
[0,2,1]
[3] os_md.ldict(3,3);
```

```
[2,0,1]
[3] os_md.ldict(4,3);
[1,2,0]
[4] os_md.ldict(5,3);
[2,1,0]
[5] os_md.ldict(6,3);
too small size
0
[6] os_md.ldict(6,4);
[0,1,2,3]
[7] os_md.ldict(7,0);
[1,0,3,2]
[8] os_md.ldict(10000,0);
[4,7,2,3,5,1,0,6]
[9] os_md.ldict(3,4|opt=0);
[2,0,1,3]
[10] os_md.ldict(3,4|opt=1);
[0,2,3,1]
[11] os_md.ldict(3,4|opt=2);
[3,1,0,2]
[12] os_md.ldict(3,4|opt=3);
[1,3,2,0]
```

293. ndict(ℓ|opt=t)
:: $\{\ell_0, \ell_1, \ldots, \ell_{m-1}\}$ の並べ替えのリスト ℓ が何番目かを返す（最初は 0 番）
ℓ の要素は，互いに異なって比較できるものであればよい.
オプション opt を指定した場合は
- $t = 0$：末尾からみて大きい辞書式順序（デフォルト）
- $t = 1$：先頭からみて小さい辞書式順序
- $t = 2$：先頭からみて大きい辞書式順序
- $t = 3$：末尾からみて小さい辞書式順序

```
[0] os_md.ndict([4,7,2,3,5,1,0,6]);
10000
[1] os_md.ndict( [0,2,1] | opt=0);
2
[2] os_md.ndict( [0,2,1] | opt=1);
1
[3] os_md.ndict( [0,2,1] | opt=2);
4
[4] os_md.ndict( [0,2,1] | opt=3);
3
```

294. nextsub([$a_0, \ldots, a_{m-1}$],n)
:: $\{0, \ldots, n-1\}$ の m 個の部分集合を並べたとき，$\{a_0, \ldots, a_{m-1}\}$ の次の部分集合を返す．最後を与えた時は 0 を返す.
- $0 \le a_0 < \cdots < a_{m-1} \le n$ の必要がある.

- 先頭からの辞書式順序.
- 最初の引数が非負整数 m のとき，$[0,1,\ldots,m-1]$ を返す.

```
[0] S = os_md.nextsub(3,5);
[0,1,2];
[1] while(S != 0){S = os_md.nextsub(S,5); print(S);}$
[0,1,3]
[0,1,4]
[0,2,3]
[0,2,4]
[0,3,4]
[1,2,3]
[1,2,4]
[1,3,4]
[2,3,4]
```

295. `nextpart(`ℓ`)`
:: 自然数の分割のリスト ℓ に対し，辞書式順序で次に大きなものを返す
$\ell=[\ell_0,\ell_1,\ldots]$ は大きさの順に並んでいなければならない

```
[0] S=[5];
[5]
[1] while((S = os_md.nextpart(S)) != 0) os_md.mycat([S,os_md.transpart(S)]);
[4,1] [2,1,1,1]
[3,2] [2,2,1]
[3,1,1] [3,1,1]
[2,2,1] [3,2]
[2,1,1,1] [4,1]
[1,1,1,1,1] [5]
```

296. `transpart(`ℓ`)`
:: 自然数の分割のリスト（Young 図式に対応）ℓ に対し，その双対を返す（例は `nextpart()` を参照）

n 次の置換群の元を $\{0,1,2,\ldots,n-1\}$ の並べ替えと考えて，ベクトル $s=[s_0\ \ldots\ s_{n-1}]$ で表す.
以下，s, t はそのような置換群の元とする.

297. `trpos(`a`,`b`,`n`)`
:: 互換 (a,b) にあたる n 次置換群の元を返す

```
[0] os_md.trpos(2,4,6);
[ 0 1 4 3 2 5 ]
[1] os_md.trpos(0,0,6);
[ 0 1 2 3 4 5 ]
```

298. `sprod(`s`,`t`)`
:: 置換群の積を返す
$s=[s_0\ \ldots\ s_{n-1}]$, $t=[t_0\ \ldots\ t_{n-1}]$ のときは積は $[s_{t_0}\ \ldots\ s_{t_{n-1}}]$ となる.
299. `sinv(`s`)`
:: 置換 s の逆元を返す
300. `slen(`s`)`
:: 置換 s の長さを返す

301. sord(s,t)
:: 置換を Bruhat order で比較する
戻り値は
0: equal　1: $s > t$　-1: $s < t$　2: no order

3.2.9 TEX

以下の函数の他にも，fctrtos(), getbygrs() などが LaTeX の出力に対応している．

302. my_tex_form(p|subst=[t_0,t_1],frac=f,root=r,ket=k)
:: print_tex_form(p) の戻り値から文字列置換や不要部分削除を行い，読みやすいソースに変換

- 置き換えは str_subst(s,t_0,t_1) でなされる．
- frac=0：係数が分数のとき，2/3 のように出力する
- frac=1：係数が分数のとき，\frac{2}{3} のように出力する．ただし，AMSTeX=0 のときは {2\over 3} のように出力．
- frac=2：係数が分数のとき，\tfrac{2}{3} のように出力する（デフォルト）．ただし，AMSTeX=0 のときは {2\over 3} のように出力．
  このとき 2^(1/2) は $2^{\frac{1}{2}}$ でなく，$\sqrt{2}$ となるよう\sqrt{2}が出力される（デフォルトでは $\sqrt{\cdots}$ の $\cdots$ が 32 字以内の場合）．
  ただし，root=r の指定で $r = 0$ のときは $(\cdots)^{\frac{1}{2}}$ の形に，$r = 2$ のときは $\sqrt{\cdots}$ の形になるよう出力される．
  同様に，2^(1/3) は $2^{\frac{1}{3}}$ でなく，$\sqrt[3]{2}$ と出力される．
- ket=1：一番内側の ( ) を除いて \left(\right\) とするする．
- ket=2：一番内側の ( ) も含めて \left(\right\) に変換する．

```
[0] AMSTeX=1$
[1] print_tex_form(x_1+x_2^2/y);
\frac{   {x}_{1}  {y}+  {x}_{2}^{ 2} }{ {y}}
[2] os_md.my_tex_form(x_1+x_2^2/y);
\frac{x_1y+x_2^2}y
[3] print_tex_form((1+x)^(1/2));
( (  {x}+ 1))^{ 1/2}
[4] os_md.my_tex_form((1+x)^(1/2));
\sqrt{x+1}
[5] print_tex_form(x/2*(1+x)^(1/2));
 1/2  ( (  {x}+ 1))^{ 1/2}   {x}
[6] os_md.my_tex_form(x/2*(1+x)^(1/2)|frac=0);
1/2(x+1)^{1/2}x
[7] os_md.my_tex_form(x/2*(1+x)^(1/2)|frac=1);
\frac{1}2(x+1)^{\frac{1}2}x
[8] os_md.my_tex_form(x/2*(1+x)^(1/2)|frac=2);
\tfrac{1}2\sqrt{x+1}x
[9]  print_tex_form(atan(x));
 {atan}( {x})
[10] os_md.my_tex_form(atan(x));
{\arctan}(x)
[11] os_md.my_tex_form(2^(1/3));
\sqrt[3]{2}
```

303. show(*p*|opt=*ℓ*)
:: *p* を dviout で適切に表示する
微分作用素，重複度つき固有値形式のリスト，GRS 形式などを自動判定して dviout で適切に表示する．期待する表示と異なる場合は，オプションを付けるか，別の細かな指定が出来る函数を使う．
- opt="verb"が指定されると，Risa/Asir での表示と同じ形で dviout で表示される．
  以下はそれ以外の場合
- *p* が有理式や多項式のとき
  - fctrtos(*p*) を使って因数分解して表示する．
  - opt の指定がなかったときは，*p* が微分作用素かどうかは isdif(*p*) で判定し，微分作用素と判断された場合は var="dif" のオプションで表示する．
  - opt="pfrac"と指定すると，*p* の主変数について部分分数展開して表示する．
  - それ以外の場合は，fctrtos() に全てのオプションパラメータを引き渡して表示する．
  - 1 行の横幅を超えてしまう数式は，行分割されて表示される（単項式や数は除く）．
- *p* が行列の時は，mtotex(*p*|lim=1,small=2) を使って表示する．
  - サイズの大きな行列も考慮されている．
  - opt の指定は，mtotex() に var の指定として渡されて表示する．
- *p* がリストの時は，opt が文字列かリストで指定されていれば，全てのオプションパラメータを ltotex(*p*) に渡して結果を表示する．それ以外では
  - GRS 形式，重複度つきの固有値の形式あるいは文字列のリストなら，それに合わせた形で出力する．なお，重複度つきの固有値の形式とは，最初が非負整数，次が式や数の 2 つの要素からなるリストを集めたリスト，GRS 形式とは，その要素が重複度つきの固有値の形式となっているリスト．
  - リスト *p* の各要素がリストとし，各要素も文字列かあるいは，式（ここでは有理式，ベクトル，行列のいすれかとする）からなるリストとする．文字列も式も現れているとすると，ltotex(*p*|opt=["cr","spts"]) で変換して表示する．ただしいずれかの文字列に \ が含まれていれば，opt=["cr","spts0"] とし，さらに str=1 のオプションをつける．
- *p* が文字列の時は，opt="raw"が指定されているとそのまま TEX の文字列とみなして表示し，opt="eq"が指定されていると数式を表すとみなして\begin{equation} \end{equation} で鋏んで display style で表示する．それ以外では，文字列に \ ^ _ のいずれの文字も含まれていなければそのまま，含まれていれば display style で表示する．
- 上記以外では dviout(*p*) によって表示される．

304. dviout(*p*|clear=1,keep=1,delete=*t*,fctr=1,mult=1,subst=[*s*0, *s*1],eq=*k*,title=*s*)
:: *p* を dviout で表示する
- dviout(@@) とすれば，直前の結果が dviout で表示される．
- dviout.exe にはパスを通しておく（あるいは，risatex.bat を書き換える．書き換えによって dviout 以外や Windows 以外の UNIX などの OS にも対応可）．
- この函数によって LATEX のソースが DIROUT/out.tex に出力され，それが DIROUT/risaout.tex から読み込まれる．
- *p* が文字列の場合，通常は LATEX のソースとみなして表示するが
  eq=0 のときは，デフォルト数式環境
  eq=1 のときは \[と\]
  eq=2 のときは \begin{align} と \end{align}
  eq=3 のときは \begin{gather} と \end{gather}
  eq=4 のときは \begin{multline} と \end{multline}
  eq=5 のときは \begin{align}\begin{split} & と \end{split}\end{align}
  eq=6 のときは \begin{align*} & と \end{align*}
  eq=7 のときは \begin{gather*} と \end{gather*}
  eq=8 のときは \begin{equation} と \end{equation}
  eq=*s* と文字列 *s* で与えられたときは，\begin{*s*} と\end{*s*}

で挟んで数式とみなして LaTeX を使って表示する．

- s_0, s_1 で部分文字列の（複数）置き換えをする（cf. str_subst()）．
- p=" "（1 文字の空白）は，再表示．
- clear=1 によって，追加でなくて新規の表示となる．
- keep=1 のときは，LaTeX のソースのみの変更で，表示はしない（後から表示）．
- delete=t によって，最後の t 個の式番号の付いた式を削除する（$0 \le t \le 10$）．
- mult=1 のときは，p がリストなら，各項を別に表示する．
- fcrt=1 のときは，p が多項式または有理式なら因数分解して表示する．
- tilte=s で，s が文字列ならそれのあとに p が表示される．
  dviout で source specials が有効になっていれば，dviout の画面で表示された文字の部分のダブルクリックによりエディタが起動され，LaTeX のソースの該当箇所が編集可能になる．
  この編集処理後の表示（たとえば dviout(" ")）でこの修正が有効になる．
- risatex.bat や risaout.tex が存在しない場合，それらが（ディレクトリが書き込み可能ならば）自動的に作成される．これらは必要に応じて書き換えてよい．
  risatex.bat を起動可能なように（デフォルトでは get_rootdir()/bin に置かれる）設定し，また risaout.tex を DIROUT で指定されるディレクトリに入れておく必要がある（cf. DVIOUTA）．

305. dviout0(ℓ) または dviout0([$\ell_1, \ell_2, \ldots$]) または dviout0(ℓ|opt=s)
:: TeX での表示のための内容削除などの基本操作
- $\ell = 0$: dviout(" "|keep=1,clear=1) と同じで内容の消去
- $\ell = 1$: dviout(" ") と同じで再表示
- $\ell = 2$: dviout(" "|clear=1) と同じで，内容を消去して再表示
- $\ell = 3$: DIROUT, DVIOUTH, DVIOUTA, DVIOUTB, DVIOUTL, Canvas, TeXLim, TeXEq, AMSTeX, TikZ, XYPrec, XYcm の値が示される．
- $\ell = 4$: DVIOUTA と DVIOUTB の値を交換し，DVIOUTA の値を示す．
  実行例は DVIOUTL の項を参照．
- $\ell = 5$: DVIOUTA と DVIOUTB を初期状態と逆にする．
- $\ell = 6$: TikZ を 1 にする（グラフ表示に TikZ を使う）．
- $\ell = 7$: TikZ を 0 にする（グラフ表示に XY-pic を使う）．
- $\ell > 10$: $\mathcal{AMS}$TeX における行列サイズを ℓ まで許す．
- ℓ が負の整数 : dviout(" "|keep=1,delete=$-\ell$) と同じで，最後の ℓ 個の式の消去
- ℓ が文字列 : 先頭に \ を付加した文字列を TeX のソースに付加して改行する．
  たとえば dviout0("newpage") は改ページを意味する．
  ただし，以下の文字列の時は別の意味を持つ．
  - "" : 空白一つと改行を付加する．
  - "cls" : ソースを空にする．
  - "show" : 表示する．
  - "?" : 設定の表示．
- 複数の操作を与えるときは，リストにする．
- opt=s を指定したときは s に ℓ の値が設定される（s は文字列）．
  ただし，$\ell = -1$ のときは現在の設定値が返される．
  s には TikZ, TeXEq, TeXLim, XYPrec, XYcm, XYLim, DVIOUT, Canvas が可能．
  なお，DVIOUT のときは，DVIOUTA, DVIOUTB の設定が $\ell = 0$ で初期状態，$\ell = 1$ で逆の設定となる．

306. verb_tex_form(p)
:: p を LaTeX で表現可能な文字列にする
verb_tex_form(p) は rtostr(p) と同等であるが，\begin{align}... \end{align} などの数式環境中でも使用できる．

307. monotos(p)
:: 有理式を文字列に変換．単項式以外では (と) で囲む

308. `monototex(`p`|minus=1)`
:: 有理式を TEX の文字列に変換．単項式以外では (と) で囲む
- `minus=1` を指定すると，− で始まる場合も (と) で囲む

309. `rtotex(`p`)`
:: 数式を TEX の文字列に変換．1 文字を越えるときは { と } で囲む

```
[0] os_md.monotos(a2);
a2
[1] os_md.monototex(a2);
a_2
[2] os_md.rtotex(a2);
{a_2}
[3] os_md.monotos(a+1);
(a+1)
[4] os_md.monototex(a+1);
(a+1)
[5] os_md.rtotex(a+1);
{a+1}
[6] P=(a^(12)+b)/(2*c+d)+alpha2;
(a^12+b+2*alpha_2*c+alpha2*d)/(2*c+d)
[7] os_md.monotos(P);
((a^12+b+2*alpha_2*c+alpha2*d)/(2*c+d))
[8] os_md.monototex(P);
(\frac{a^{12}+b+2{\alpha}_2c+{\alpha}_2d}{2c+d})
[9] os_md.rtotex(P);
{\frac{a^{12}+b+2{\alpha}_2c+{\alpha}_2d}{2c+d}}
```

310. `texsp(`s`)`
:: TEX の文字列 s の最後が TEX のキーワードのとき，s の末尾に空白をつける．

```
[0] os_md.my_tex_form(alpha*x);
 \alpha{x}
[1] os_md.my_tex_form(alpha)+"x";
 \alphax
[2]  os_md.texsp(os_md.my_tex_form(alpha))+"x";
 \alpha x
[3] os_md.texsp(os_md.my_tex_form(alpha*x))+"y";
 \alpha{x}y
```

311. `texbegin(`t`,`s`|opt=`u`)`
:: TEX の `\begin{`t`}[`u`]` s `\end{`t`}` というソースを出力する
312. `texcr(`k`)`
:: 127 以下の非負整数 k に応じて TEX における数式の改行の文字列を返す
$k = 127$ のときは，改行のコードは

```
,\allowdisplaybreaks\\\pause\n& =
```

となる（なお，TEX のソースでは，上の `\\` は `\` に，`\n` は改行になる）．
`iand(cr,`k`)`=0 のときは上の文字列から k に応じて順に以下の文字列が削られる．

- 32 : ,
- 8 : \allowdisplaybreaks
- 2 : \\
- 16 : \pause
- 1 : \n
- 4 : &
- 64 : =
- $k=0$ のときは空白 1 文字となる.
- k が 0 から 127 までの整数でない場合は，k がそのまま返される.

```
[0] os_md.texcr(31);
\allowdisplaybreaks\\\pause
&
[1] os_md.texcr(7);
\\
&
[2] os_md.texcr(15);
\allowdisplaybreaks\\
&
```

313. texket(s|all=t)
:: TEX のソース s における括弧のサイズを可変にする
- 一番内部の ( ) 以外を \left(\right) に変更する.
  一番内部は，その中身が空白や英数字や { } や _ 以外の文字が含まれる場合は同様に変更する.
  既に \left(\right) となっている箇所は変更しない.
- all=1 : 全ての括弧を \left(\right) と変更する.
- all=-1 : 一番内部の ( ) は変更しない.

```
[0]  S=os_md.my_tex_form((sin(x+x0)/(cos(x-x0)))^(1/2));
( \frac{ \sin(x+x_0)}{ \cos(x-x_0)})^{\tfrac{1}2}
[1] os_md.texket(S);
\left( \frac{ \sin(x+x_0)}{ \cos(x-x_0)}\right)^{\tfrac{1}2}
[2] os_md.texket(S|all=1);
\left( \frac{ \sin\left(x+x_0\right)}{ \cos\left(x-x_0\right)}\right)^{\tfrac{1}2}
```

314. ltotex(l|opt=s,pre="string",cr="cr",small=1,lim=ℓ,var=v)
:: リストまたはベクトルを $s=$ "spt" のとき重複度つきのリストとみて\left\{... または $s=$ "GRS" のとき \begin{Bmatrix} ⋯ の形の Riemann scheme とみて TEX の文字列に変換など
- 画面表示する場合は，show() を参照のこと（以下のオプションパラメータはそのまま有効）.
- $s=$ "spt" の場合は meigen(|mult=1) のようなリスト形式をスペクトル型にする.
- $s=$ "GRS" の場合は sp2grs() の結果のようなリスト形式を Riemann scheme 形式にする．この場合は，pre が有効.
- $s=$ "coord" の場合はリストを座標とみなしてその形式 (, ,...) にする.
  - cpx=1 : 複素数 $a+b\sqrt{-1}$ のとき，$a+bi$ と表示する（デフォルト）.
  - cpx=2 : 複素数 $a+b\sqrt{-1}$ のとき，$a+b\sqrt{-1}$ と表示する.
- $s=$ ["Pfaff",u,x,x-y,...] の場合は行列のリストを

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

のような Pfaff 形式にする.

- s = ["Fuchs",u,x,x,x-y,x-1,...] の場合は行列のリストを

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

のような Fuchs 形式にする.
- s = "dform" のときは，dform() で扱ったような微分形式を表すリストとみなす.
ℓ =[[(a*x-b*y)/(x),x,z],[(-a*x+b*y)/(y),y,z]] ならば

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

となる．係数は，有理式や有理式の行列などでよい.
- s = "vect" のときは，縦ベクトルとしての表示形態にする.
- s = "cr" のときは，リストの 1 項目毎に改行する.
次項と同様，cr= や var= のオプション指定が可能
- s = "spts" のときはリストの各項目を空白を挟んで，s = "spts0" のときは空白を挟まずに，1 行あたり出来るだけ多く入れる.
  - lim=ℓ でデフォルトの 1 行の文字幅を ℓ に変更可能で，$\ell = 0$ のときは文字幅制限なし.
  - str=1 で，文字列はそのまま TEX の文字列と解釈して挿入する.
  - cr= の指定で改行のコードを指定可能．デフォルトは cr="\\\\\n & ".
  0 以上 32 未満の数字 k を指定することもできて，改行のコードは texcr(k) となる．すなわちデフォルトは $k = 7 = 2+1+4$ で，$k = 0$ のときは空白の 1 文字となる．たとえば $k = 15$ のときは \\allowdisplaybreaks\\\\\n&
  - var= による変数の指定が行列（cf. mtotex()）や多項式，有理式（cf. ltotex()）に適用される.
- s =["cr","spts"] または s =["cr","spts0"] のときは，リストの 1 項目毎に改行するが（cr= や var= のオプション指定が可能），リストの項目がリストのときはその項目を s ="spts" または s ="spts0" の形式にし，それにはさらに str=1 のオプションが有効.
ℓ のリスト（改行文字列は s_1）の項目がリストの場合，そこでの改行文字列 s_2 を指定するときは，cr=[s_1,s_2] とする．s_1, s_2 は数字での指定でもよい.
- s = "text" のときは，s = "spts" のときと同じだがテキストスタイルとする．このときも str=1 や cr= のオプションが可能.
- s = "tab" で ℓ がリストのリストのときは，\begin{tabular}...\end{tabular}の形式でテキストスタイルで出力する.
  - title=s で文字列 s を指定すると，タイトルがつけられる.
  - left=[s_1,s_2,...] を指定すると，それが 1 列目として付加される.
  途中の s_j が [$s_{j,1}$,$s_{j,2}$,...] となっていると，それが最終の行まで繰り返される（以下も同様）.
  - right=[s_1,s_2,...] を指定すると，それが最終列として付加される.
  - top=[s_1,s_2,...] を指定すると，それが先頭行として付加される（上の right=などがあればそれの処理の後）.
  - last=[s_1,s_2,...] を指定すると，それが最終行として付加される.
  - null=n を指定すると，n に等しい要素は表では空白とする（デフォルトは n =""）.
  - hline=[$h_1, h_2, \ldots$] を指定すると，h_i 行の後に横線を引く.
  （タイトルを除いた）表の始まりが 0，最初が 1 行目と数え，重複しての指定は多重線を表す.
    * z は最終行（の後）を示す．z-1 のような指定も可能.
    * v_j が，[k_j,n_j] という正整数 n_j と非負整数 k_j の組のリストときは，n_j で割った余りが k_j となる行全ての指定を意味する.
  - vline=[$v_1, v_2, \ldots$] を指定すると，v_j 列の後に縦線を引く．表の両端の縦線以外を指定する.
  最初が 1 列目で，重複しての指定は多重線を表す.
    * z は最終列（の後）を示す．z-1 のような指定も可能.

* v_j が，[k_j,n_j] という正整数 n_j と非負整数 k_j の組のリストのときは，n_j で割った余りが k_j となる列全ての指定を意味する．

– align=s：\begin{tabular} に続いて配置や縦線を指定する { } の中の文字列を指定する．このオプションがあるときは，vline= の指定は無視される．
ただし s が 1 文字の時は，各項の配置を指定する文字で vline= の指定は有効．"r" はデフォルトの右寄せ．"c" は中央配置，"l" は右寄せである．

– vert=1：縦横を転置して表にする．

– width=w：w が正のときは，横を w 項にして縦に積み重ねた表にする．
w が負の時は，横の項数を $|w|$ 倍して行数の少ない表にする．

* vert=1 が指定してあれば，縦横を転置してから横の項数調整を行う．

オプションの使用例は powprimroot() の項にある．
リストの調整は lv2m(|null="") で行列に直し，madjust()，mtranspose()，mperm()，newbmat() などを使って変更後，m2ll() でリストに戻すのが便利．

- s = "graph" で ℓ が数字のリストのときは，XY-pic または TikZ を使って棒グラフを出力する．棒グラフの下に文字列や数字を入れるときは，それをリストにして数字のリストの後ろにつける．
そのリストの個数がデータの個数より 1 つ少ないときは，棒グラフの間に文字列や数字を入れる．

– size=ℓ：このときオプションでサイズ指定ができる．ℓ は数のリストで，全体の横幅，最も長いグラフの高さ，棒幅の（隣との幅との）比率（折れ線グラフの時は無視される），文字列との空きの高さ，グラフと文字列との秋の高さ（指定しないときは前者と同じ値），を順に指定できる．最初の 2 つ以外は省略可能．
ℓ の 2 番目の成分が負の数 c の時は，棒グラフの高さをデータの数値の c 倍とする．
TikZ=0，1 に応じてデフォルトは $\ell = [80, 30, 1/2, 2]$，$[8, 3, 1/2, 0.2]$．（ただし，horiz=1 を指定したときは，4 番目値の 2 または 0.2 のデフォルトは，3 または 0.3 になる）．

– max=m：上限値を m とする（デフォルトは自動判定，上限値がグラフの高さになる）．

– shift=n：底の基準線の値を n にずらす（デフォルトは 0）．

– horiz=1：棒グラフを縦でなくて横に描く．

– line=1：折れ線グラフで描く．
line=2：上と同じだが，ポイントを • で明示する．

– line=[n,t]：n は 2 または 1 で折れ線グラフでのポイント有無．
t は XY-pic/TikZ で使われる線種などの指定文字列．たとえば s="@{.}"/dotted は点線（cf. xyarrow()）．

– line=[-1,r]：r mm を半径とする円グラフで表示する．

– value=0：値を表示しない．
value=1：棒グラフが底の基準線の位置につぶれても値を表示する．

– strip=1：\begin{xy} ··· \end{xy}や\begin{tikzpicture} ··· \end{tikzpicture} などの始まりと終わりの部分を出力しない．
strip=2：さらに下の基準線や目盛を描かない．
strip=3：値のみを表示する（value=0 で抜ける部分のみの表示）

– color=s：というオプション指定をすると，TikZ を用いたグラフの場合には色付けや塗りつぶしができる（xybox() のときの指定と同じ）．円グラフでは，指定文字列のデータの個数だけ並べたリストとする．

– mult=1：複数の棒グラフまたは複数の折れ線グラフを重ねて書くことができる．この場合，ℓ は，[[ℓ_1,ℓ_2,...],m] の形で，color または line も各グラフごとにリストにして指定する必要がある（無指定は不可）．
relative=1：さらにこれを指定すると，ℓ_1, ℓ_2 の値は，順に成分の値を加えたデータとしたグラフを描く．これを指定しない場合は，棒グラフの時は ℓ_j と ℓ_{j+1} の各成分は，後者の方が大きいとしている．

- 返された文字列 s は，s ="text"，"tab"，"graph" の場合は dviout(s) で，それ以外では dviout(s|eq=5) で表示できる．

- small=1 のときは行列を小サイズにする.

```
[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.

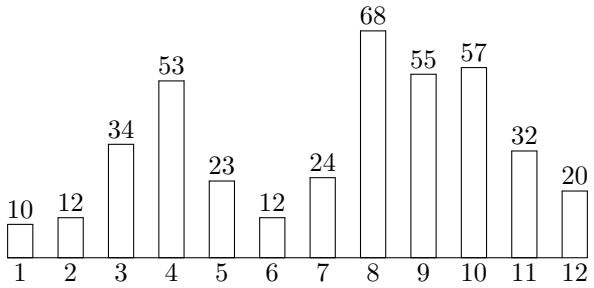
```
[23] L=[10,12,34,53,23,12,24,68,55,57,32,20]$
[24] M=[1,2,3,4,5,6,7,8,9,10,11,12]$
[25] os_md.dviout(os_md.ltotex([M,L]|opt="tab",title="Year 2014"));
```

上で得られた TEX のソースを表示または印刷すると（以下の例でも同様）.

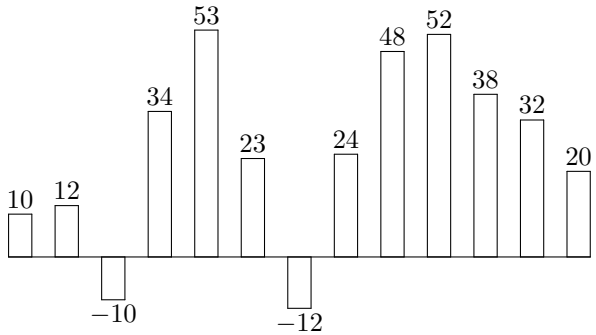
Year 2014

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

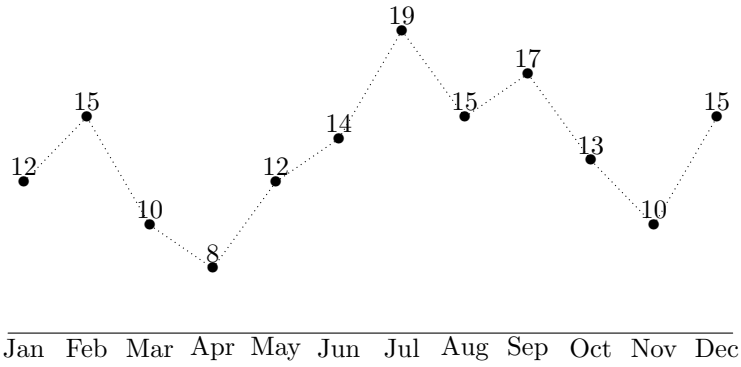
```
[26] os_md.dviout(os_md.ltotex([L,M]|opt="graph"));
```




```
[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]);
```

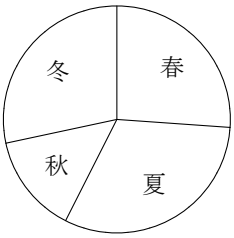



```
[32] L=cons("number",L)$
[33] M=cons("Month",M)$
[34] os_md.ltotex([M,L]|opt="tab",hline=[0,1,2],vline=[0,1,1,13],
title="Year 2014");
```

Year 2014

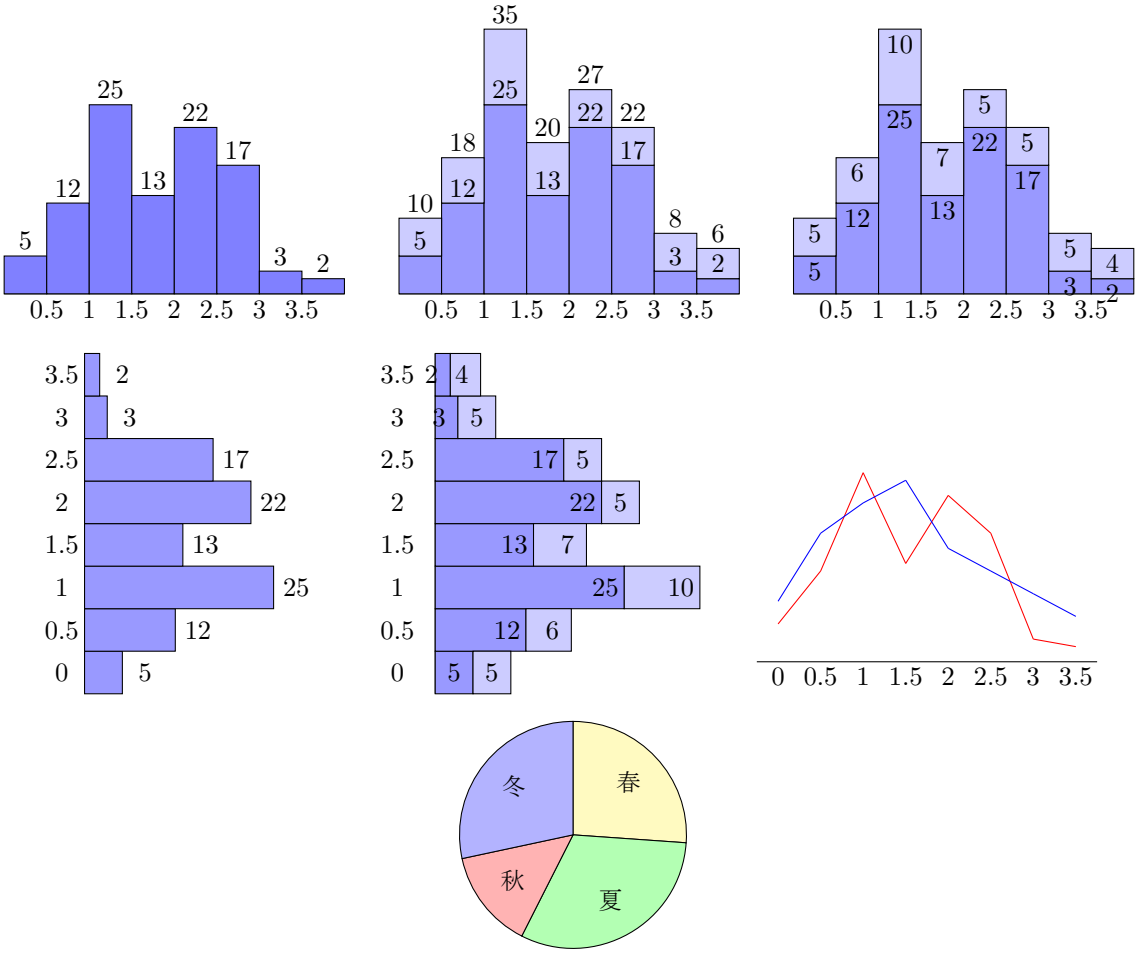
Month	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
number	12	15	10	8	12	14	19	15	17	13	10	15

```
[35] L=[35,42,19,38]$
[36] LL=["春","夏","秋","冬"]$
[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]);
\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=ladd(L2,L,-1);
[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]$LL=["春","夏","秋","冬"]$
[53] os_md.ltotex([L,LL]|opt="graph",line=[-1,15],color=["fill=yellow!30",
     "fill=green!30","fill=red!30","fill=blue!30"]);
```



315. mtotex(m|small=1,2, null=1,2, sp=1,2, idx=0,1, mat=s, var=ℓ, raw=1, lim=n)
:: 行列またはベクトルを TEX の文字列に変換するが，成分が有理式のときは因数分解した形にする

- m がベクトルのときは，1 行の行列とみなす．
- `null=1` では，0 の成分は空白にする．`null=2` では対角成分は残す．
- `var=`ℓ　多項式や有理式の成分を `fctrtos()` で変換するときのオプション．
- `small=1` とすると，小さな行列にする．行列の文字数換算の幅は小さくなる．
- `small=2` とすると，`lim` を設定したとき，横幅が大きすぎる行列を小さな行列にする．
- `raw=1` を指定しないと，成分が文字列のとき `\text{ }` の形で，指定しない場合は LaTeX の文字列としてそのまま用いる．
- `sp=1` 成分が 2 個の元からなるリスト `[s,t]` のとき $[t]_s$ と，`sp=2` のとき $[t]_{(s)}$ となるようにする．
- `idx=1` とすると，先頭に行を追加して，列の番号を (1), (2),... のようにつける．`idx=0` のときは (0) から始める．
- `idx=`ℓ でリストやベクトルを指定したときは，先頭に行を追加してそれを各列に順に並べる．個数が不足したら，それ以降は最後のものを並べる．
- `lim=`n とすると，行列がその文字数幅に入らないと推測されるときは，行列を分割して表示する．ただし，$0 < n < 30$ のときはデフォルトの横幅サイズとする．`lim=0` は横幅制限なし．
- `mat=`s において，s=`"p"`, `"b"`, `"B"`, `"v"`, `"V"` のとき行列は ( ), [ ], { }, | |, ‖ ‖ の形になる．s =`""` のときは，括弧は描かれない．
- `len=1` を指定すると，LaTeX の文字列と推測された横幅の文字数を 2 項目に加えたリストを返す．

```
[0] A=newmat(2,2,[[y/(x+a),z],[0,1]]);
[ (y)/(x+a) z ]
[ 0 1 ]
[1] os_md.mtotex(A^2);
\begin{pmatrix}
 \frac{y^2}{(x+a)^2} & \frac{z(x+y+a)}{x+a}\\
 0 & 1
\end{pmatrix}
[2] os_md.mtotex(A^2|small=1);
\left(\begin{smallmatrix}
 \frac{y^2}{(x+a)^2} & \frac{z(x+y+a)}{x+a}\\
 0 & 1
\end{smallmatrix}\right)
[3] B=os_md.mgen(3,"highdiag",a,1);
[ 0 a1 0 ]
[ 0 0 a2 ]
[ 0 0 0 ]
[4] os_md.mtotex(B);
\begin{pmatrix}
 0 & a_1 & 0\\
 0 & 0 & a_2\\
 0 & 0 & 0
\end{pmatrix}

[5] os_md.mtotex(B|null=1);
\begin{pmatrix}
  & a_1 & \\
  &  & a_2\\
```

```
  &  &
\end{pmatrix}

[6] os_md.mtotex(B|null=2);
\begin{pmatrix}
 0 & a_1 & \\
  & 0 & a_2\\
  &  & 0
\end{pmatrix}
[7] os_md.mtotex(A^3|len=1)$
[8] @@[1];
34
[8] os_md.mtotex(A^3|len=1,small=1)$
[9] @@[1];
27
```

316. divmattex(s,l)

:: 行列の TEX のソース s の分割や列の並べ替えを行う
l によって以下のように変換される.

- [[k_0,k_1,...]：ν 列目をもとの行列の k_ν 列目とする（$\nu = 0, 1, \ldots$）.
- [[k_0^1,k_1^1,...],[k_0^2,k_0^2,...],...]：横長の行列を分割して複数に分ける.
  $[k_0^1$,k_1^1,...] は最初の分割された行列.
- [m_1,...]：[[0,1,...$m_1 - 1$],[m_1,...],...]：を表す.
  特に [m]：とすると，行列を最初の m 列とそれ以降との 2 つに分割する.
- 0：成分が文字列の Risa/Asir の行列の形で返す．ただし，その成分がない場合は 0 となる.

```
[0] S=os_md.my_tex_form(os_md.mgen(3,9,a,0));
\begin{pmatrix}
 a_{00} & a_{01} & a_{02} & a_{03} & a_{04} & a_{05} & a_{06} & a_{07} & a_{08}\\
 a_{10} & a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} & a_{18}\\
 a_{20} & a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} & a_{27} & a_{28}
\end{pmatrix}
[1] os_md.divmattex(S,[[2,4,1]]);
\begin{pmatrix}
a_{02} & a_{04} & a_{01} \\
a_{12} & a_{14} & a_{11} \\
a_{22} & a_{24} & a_{21}
\end{pmatrix}
[2] os_md.divmattex(S,[4]);
\begin{align*}
&\left(\begin{matrix}
a_{00} & a_{01} & a_{02} & a_{03} \\
a_{10} & a_{11} & a_{12} & a_{13} \\
a_{20} & a_{21} & a_{22} & a_{23}
\end{matrix}\right.\\
&\quad\left.\begin{matrix}
```

```
 a_{04} & a_{05} & a_{06} & a_{07} & a_{08}\\
 a_{14} & a_{15} & a_{16} & a_{17} & a_{18}\\
 a_{24} & a_{25} & a_{26} & a_{27} & a_{28}
\end{matrix}\right)
%
\end{align*}
```

317. `smallmattex(`s`)`
:: TEX のソースで ( ) や { } で囲まれた行列を小サイズに変換する（cf. `mtotex()`）

318. `texlen(`s`)`
:: LATEX の数式の横幅を推測して文字数で返す
`Risa/Asir` の TEX 出力の文字列で，多項式や有理式に対応する．改行記号や行列には対応していない．
より具体的には以下の通り．
- `\frac{...}{...}` は，分母と分子の推測横幅文字数の大きい方と数える（ネスティング可）
- 上を除いて `\` で始まりアルファベットか続く部分はまとめて 1 文字幅とみなす．
- 空白や改行文字，`& ^ _ { }` は文字数に数えない．
- `_*^` が存在して `*` が任意の 1 文字の場合は `_*` を文字数に含めない．
- `_{**}^` または `_{**}}^` に対しては（`*` は任意文字），`_{**}` を 1 文字とみなして数える．

319. `texlim(`s,n`|del=`s_0`,cut=`s_1`)`
:: 長い TEX の数式を，複数行に分割する
s は多項式に対応する（`\frac` などが含まれていない）TEX の数式．
`texlen()` で計った文字数幅が n を越えないよう分割する．ただし $n < 30$ のときは，$n =$`TeXLim` とみなされる．
- 改行は `+ - (` の前．ただし `\Bigl(` や `\biggl(` などには正しく対応している．
- s_0 は元の改行の文字列（これは残される）で s_1 は新たに挿入される改行の文字列．
- デフォルトは，$s_1 =$`"\\\\\n&"` かつ $s_0 = s_1$．
- $s = 1$ のときは，`TeXLim` を n に設定するコマンドとなる．

3.2.10 Lines and curves

320. `ladd(`u`,`v`,`t`)`
:: ベクトルまたはリストの成分の和（$t = 1$）や差（$t = -1$）を成分とするリストを返す
u に v を t 倍して加えた座標を返す．

```
[0] os_md.ladd([1,2],[3,4],1);
[4,6]
[1] os_md.ladd(ltov([1,2]),[3,4],1);
[4,6]
[2] os_md.ladd(P,Q,-1);
[-1,-2]
```

321. `dnorm(`v`)`
:: ベクトルまたはリストのノルム，または 2 点間の距離を返す
v が $[v_1, v_2]$ という 2 つのリストまたはベクトルの時は，その差のノルムを返す．

```
[0] os_md.dnorm([1,2]);
2.23607
[1] os_md.dnorm([[1,2],[2,3]]);
1.41421
```

322. `mrot(`θ`|deg=1)` `mrot([`θ'_z`,`θ_y`,`θ_z`]|deg=1,conj=1)`

:: 角度 θ の回転行列を返す

- deg=1 を指定すると，度単位となる．
- 引数が [θ'_z,θ_y,θ_z] のときは，z 軸の周りに θ_z, x 軸の周りに θ_y 最後に z 軸の周りに θ'_z 回転した 3 次元の回転行列を返す．
  このとき conj=1 を指定すると，z 軸の周りに θ_z, y 軸の周りに θ_y 回転したときの z 軸にあたる軸に対する θ'_z の回転を返す．

```
[0] os_md.mrot(@pi/6);
[ 0.866025 -0.5 ]
[ 0.5 0.866025 ]
[1] os_md.mrot(45|deg=1);
[ 0.707107 -0.707107 ]
[ 0.707107 0.707107 ]
[2] os_md.mrot([30,0,0]|deg=1);
[ 0.866025 -0.5 0 ]
[ 0.5 0.866025 0 ]
[ 0 0 1 ]
[3] os_md.mrot([0,30,0]|deg=1);
[ 0.866025 0 -0.5 ]
[ 0 1 0 ]
[ 0.5 0 0.866025 ]
[4] os_md.mrot([1,30,45]|deg=1,conj=1);
[ 0.999867 -0.0151333 -0.00612372 ]
[ 0.0150952 0.999867 -0.00621699 ]
[ 0.00621699 0.00612372 0.999962 ]
```

323. dvangle(v_1,v_2)　dvangle($[u_1,u_2,u_3]$,0)

:: ベクトルまたはリストの鋭む角度の余弦を返す

- dvangle($[u_1,u_2,u_3]$,0) は dvangle(u_2-u_1,u_3-u_2) と解釈される．
- v_1 または v_2 が 0 のときや u_1, u_2, u_3 のいずれかが 0 のときは 1 を返す．
- 2 次元ベクトル V の偏角（π 以下で $-\pi$ より大）は myarg(V[0]+V[1]*@i) で得られる．

324. ptaffine(m,ℓ|org=v,shift=w,arg=θ,deg=θ,proc=1)

:: 実数の組（座標）のリストを結合する，またはアフィン変換（こちらは**描画実行形式**も可）を施す

- 曲線や折れ線の通過点の座標を表す複数のデータのリストを想定している．
- m は行列またはスカラー
- org=v：v は座標で，ここを原点とした線型変換とする（デフォルトは原点中心）．
- arg=θ：平面座標のリストのとき指定可能．角度 θ の反時計回り回転を行った後，M による線型変換を行う．θ は@pi/2 など deval() によって実数が得られるものでもよい．
- deg=θ：上と同じだが，角度をラジアンでなく度で与える．
- shift=w：w は座標で，変換を行ったあと，w だけ平行移動する．
- proc=1：**描画実行形式**（cf. execdraw()）ℓ のアフィン変換の場合に指定．
- ℓ の成分に数や文字列などがあってもよい．それはそのままに保たれる（xybezier() の引数など）．
- ℓ は単に座標であってもよい．
- 座標はリストでなくてベクトルで与えてもよい．
- m が以下の文字列のときは特別の意味をもつ（connect などで，**曲線を繋げる**ことができる）．
  - reverse：ℓ を xybezier() のデータとみなして，描く順序を逆にしたデータに変換
  - union：ℓ を xybezier() のデータのリストとみなして，合わせて一つの曲線データとする．

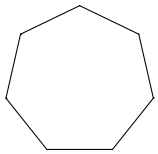
– connect : ℓ を xybezier() のデータのリストとみなして，順につなげた一つの曲線データとする．
– close : ℓ を xybezier() のデータまたはそのリストとみなして，順につなげ，始点と終点をつなげて閉曲線データとする．
– loop : ℓ を xybezier() のデータまたはそのリストとみなして，順につなげ，終点を始点に変えて閉曲線データとする．

```
[0] L=[[1,0],[0,1],[1,1]]$
[1] os_md.ptaffine(2,L|arg=3.1416/8);
[[1.84776,0.765369],[-0.765369,1.84776],[1.08239,2.61313]]
[2] os_md.ptaffine(2,L|arg=3.1416/8,org=[1,1]);
[[1.76537,-0.847758],[-0.847758,0.234631],[1,1]]
[3] os_md.ptaffine(2,L[0]|arg=3.1416/8,org=[1,1]);
[1.76537,-0.847758]
[4] L1=[[2,0],[1,2],[2,1]]$
[5] os_md.ptaffine("union",[L,L1]);
[[1,0],[0,1],[1,1],0,[2,0],[1,2],[2,1]]
[6] os_md.ptaffine("connect",[L,L1]);
[[1,0],[0,1],[1,1],1,[2,0],1,[1,2],[2,1]]
[7] os_md.ptaffine("close",[L,L1]);
[[1,0],[0,1],[1,1],1,[2,0],1,[1,2],[2,1],1,-1]
[8] L3=[[1,1],[2,0],[1,2],[1,0]]$
[9] os_md.ptaffine("connect",[L,L3]);
[[1,0],[0,1],[1,1],1,[2,0],[1,2],[1,0]]
[10] os_md.ptaffine("close",[L,L3]);
[[1,0],[0,1],[1,1],1,[2,0],[1,2],-1]
[11] L2=[2,1],[1,2],[2,1]]$
[12] os_md.ptaffine("close",[L,L2]);
[[1,0],[0,1],[1,1],1,[2,1],1,[1,2],[2,1],1,-1]
[13] os_md.ptaffine("loop",[L,L2]);
[[1,0],[0,1],[1,1],1,[2,1],1,[1,2],-1]
[14] os_md.ptaffine("reverse",[[1,0],[0,1],[1,1],1,[2,0],[1,2],[2,1]]);
[[2,1],[1,2],[2,0],[1,1],1,[0,1],[1,0]]
```

325. ptpolygon(n,r|org=p,scale=t,arg=θ,deg=θ)
:: 半径 r の円に内接する正 n 多角形の頂点の平面座標
- 半径 r の円に内接する正多角形の座標のリストを返す
- デフォルトでは下辺が水平となる．
- org=p : 中心の座標（デフォルトは $(0,0)$）．
- arg=θ : 多角形を反時計回りに θ 回転．@pi/8 などの指定も可能．
- deg=θ : 上と同じだが，角度の単位が度
- xylines() の例を参照．

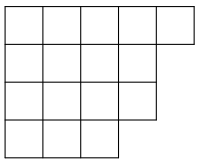
```
[0] os_md.ptpolygon(5,2);
[[-1.17557,-1.61803],[1.17557,-1.61803],[1.90211,0.618034],[-2.07711e-013,2],
[-1.90211,0.618034]]
[1] os_md.sint(os_md.ptpolygon(5,2),4);
```

```
os_md.sint(os_md.ptpolygon(5,2),4);
[[-1.1756,-1.618],[1.1756,-1.618],[1.9021,0.618],[0,2],[-1.9021,0.618]]
[2] os_md.ptpolygon(4,2);
[[-1.41421,-1.41421],[1.41421,-1.41421],[1.41421,1.41421],[-1.41421,1.41421]]
[3] os_md.sint(os_md.ptpolygon(4,1|deg=45,scale=2),5);
[[0,-2],[2,0],[0,2],[-2,0]]
[4] os_md.xylines(os_md.ptpolygon(7,10)|close=1,dviout=1);
```



326. ptlattice(m,n,v_1,v_2|org=p,scale=t,cond=[f_1,f_2,...],line=1)
:: org を始点として v_1 方向に m 個まで，v_2 方向に n 個までの合計 $m \times n$ 個の格子点の座標
- scale=t：得られた座標を全て t 倍にして返す（org も同様）.
- line=1：格子点でなくて，格子を描く線分のデータを返す xylines() の例を参照）.
- $f_1, f_2, \ldots$ は (x, y) の函数で，これらが全て非負の点のみ残す（平面座標の時のみ指定可能）.

```
[0] os_md.ptlattice(2,3,[1,0],[0,1]);
[[0,0],[0,1],[0,2],[1,0],[1,1],[1,2]]
[1] os_md.ptlattice(2,3,[1,0],[0,1]|line=1);
[[0,0],[1,0],0,[0,1],[1,1],0,[0,2],[1,2],0,[0,0],[0,2],0,[1,0],[1,2]]
[2] os_md.xylines(os_md.ptlattice(10,5,[5,0],[0,5]|line=1,cond=[35-x*2+y])|dviout=1);
```



327. ptcopy(ℓ,v)
:: 座標のリスト ℓ を移動方向のリスト v に従って複製コピーする
- 移動方向のリストの成分で 0 はそのままコピーされる.
- 各移動の間には 0 が挿入される.
- 移動方向が一つなら，それを v で指定してよい.

```
[0] os_md.ptcopy([[0,1],[1,2]],[[0.1,0],[0,0.1],[0.1,0.1]]);
[[0.1,1],[1.1,2],0,[0,1.1],[1,2.1],0,[0.1,1.1],[1.1,2.1]]
```

328. ptcommon([s_1,s_2],[t_1,t_2]|in=k)
:: 直線や線分や円に対して共通点，接点や垂線の足，内分点，方向転換進行点，角度などを求める
- $s_1 = [x_1, y_1]$, $s_2 = [x_2, y_2]$, $t_1 = [u_1, v_1]$, $t_2 = [u_2, v_2]$ のとき，s_1 と s_2 とを結ぶ直線と，t_1 と t_2 を結ぶ直線の交点の座標（並行の時は 0 を一致の時は 1）を返す.
- $s_1 = [x_1, y_1]$, $s_2 = [x_2, y_2]$, $t_1 = [u_1, v_1]$, $t_2 = 0$ のとき，s_1 と s_2 とを結ぶ直線に点 t_1 から下ろした垂線の足の座標を返す,
- $s_1 = [x_1, y_1]$, $s_2 = [x_2, y_2]$, $t_1 = [u_1, v_1]$, $t_2 > 0$ のとき，s_1 と s_2 とを結ぶ直線と，点 t_1 を中心とする半径 t_2 の円との交点を返す（交点をリストで，交点がないときは 0 を返す）.
- $s_1 = [x_1, y_1]$, $s_2 > 0$, $t_1 = [u_1, v_1]$, $t_2 > 0$ のとき，点 s_1 を中心とする半径 s_2 の円と，点 t_1 を中心とする半径 t_2 の円との交点を返す（交点をリストで，交点がないときは 0 を返す）.
- $s_1 = [x_1, y_1]$, $s_2 > 0$, $t_1 = [u_1, v_1]$, $t_2 = 0$ のとき，点 s_1 を中心とする半径 s_2 の円に t_1 を通る接線を引いたときの接点の座標を返す（接点をリストで，接点がないときは 0 を返す）.

- $s_1 = [x_1, y_1]$, $s_2 = [x_2, y_2]$, t_1 が数のとき，$\overrightarrow{s_1 s_2}$ を t_2 だけ回転した向きの単位ベクトルを t_1 倍したベクトルだけ s_2 から移動した点を返す（方向転換進行点）.
  ただし in=1 を指定すると，線分 $s_1 s_2$ の比 $t_1 : t_2$ での内分点を返す.
- in=1 を指定すると，2 点を結ぶ直線が 2 点を結ぶ線分で置き換えられる.
- in=-1 を指定すると，2 点を結ぶ直線が 2 点を結ぶ線分の垂直二等分線で置き換えられる.
- in=-2 を指定すると，s_1 と s_2 とを結ぶ直線が，その 2 点を結ぶ線分の垂直二等分線で置き換えられる.
- in=-3 を指定すると，s_1 と s_2 とを結ぶ線分の垂直二等分線と，t_1 と t_2 を結ぶ線分の交点があればそれを返す.
- 直線と直線の交点を求める場合は，座標に不定元が含まれていてもよい．また，全ての座標が有理数で指定してあれば，直線の交点の座標も有理数で返される.
- in=2 を指定すると，$\overrightarrow{s_1 s_2}$ と $\overrightarrow{t_1 t_2}$ のなす角度を返す（$-\pi$ を越えて π 以下で，$\overrightarrow{s_1 s_2}$ を基準として反時計回りが正の向き）.
- in=3 を指定すると，上と同じだが度単位で返す.

```
[0] os_md.ptcommon([[0,0],[1,2]],[[0,4],[4,0]]);  /* 二直線 */
[4/3,8/3]
[1] os_md.ptcommon([[0,0],[1,2]],[[0,4],[x,y]]);
[(4*x)/(2*x-y+4),(8*x)/(2*x-y+4)]
[2] os_md.ptcommon([[0,0],[1,2]],[[0,4],[4,0]|in=1);  /* 二線分 */
0
[3] os_md.ptcommon([[0,0],[2,2]],[[0,0],[2,3]]|in=-2);  /* 垂直二等分線と直線 */
[4/5,6/5]
[4] os_md.ptcommon([[0,0],[2,2]],[[5,4],0]);       /* 垂線の足を求める */
[9/2,9/2]
[5] os_md.ptcommon([[0,a],[1,b]],[[0,c],[1,d]]);
[(a-c)/(a-b-c+d),(d*a-c*b)/(a-b-c+d)]
[6] os_md.ptcommon([[0,0],[4,8]],[[4,8],0.5]|in=1); /* 線分と円 */
[[3.77639,7.55279]]
[7] os_md.ptcommon([[0,0],1],[[1,2],3]);            /* 二円 */
[[-0.96332495807107996983525630141,-0.26833752096446001508],
[0.36332495807107996975,-0.931662479035539984 87]]
[8] os_md.ptcommon([[0,0],1],[[1,4],0]);            /* 円の接点を求める */
[[-0.88235294117647058813,0.47058823529411764725015332 4556],[1,0]]
[9] os_md.ptcommon([[0,1],[2,3]],[1,2]|in=1);       /* 内分点 */
[2/3,5/3]
[10] os_md.ptcommon([[0,1],[2,3]],[t,1-t]|in=1);
[2*t,2*t+1]
[11] os_md.ptcommon([[0,1],[2,3]],[2,-1]|in=1);      /* 外分点 */
[4,5]
[12] os_md.ptcommon([[0,0],[2,3]],[1,@pi/2]);       /* 方向転換進行点 */
[1.16795,3.5547]
[13] os_md.ptcommon([[0,0],[1,1]],[[0,0],[-2,2]]|in=2);  /* 角度（ラジアン)*/
1.5708
[14] os_md.ptcommon([[0,0],[1,1]],[[0,0],[-2,2]]|in=3);  /* 角度（度）*/
90
```

329. ptwindow(ℓ,[x_1,x_2],[y_1,y_2]|scale=t)
:: 平面座標 (x,y) のリストで $x_1 \le x \le x_2$，$y_1 \le y \le y_2$ を満たさないものを 0 に変える
- scale=t : リストで与えられた平面座標は t 倍されているとみなす.

```
[0] os_md.ptwindow([[0,3],[2,1],[4,2],[5,6],[1,5]],[0,3],[1,6]);
[[0,3],[2,1],0,0,[1,5]]
```

330. ptbbox([[x_1,y_1,...],[x_2,y_2,...],...]|box=b)
ptbbox([[[$x^{(1)}_{min}$,$x^{(1)}_{max}$],[$y^{(1)}_{min}$,$y^{(1)}_{max}$],...],[[$x^{(2)}_{min}$,$x^{(2)}_{maxS}$],...],...]|box=1)
:: 座標 $(x,y,\ldots)$ や箱のリストを囲む箱 [[x_{min},x_{max}],[y_{min},y_{max}],...] を返す
- 各座標はリストでもベクトルでもよい．xybezier() の引数の形式でもよい.
- box=1 を指定すると，箱のリストを与えて，それらを囲む箱を返す.
- box=b として箱 b を指定すると，ptbbox([ptbbox(ℓ),b]|box=1) と解釈される.

```
[0] os_md.ptbbox([[1,2],[3,4],[0,8],[4,1]]);
[[0,4],[1,8]]
[1] os_md.ptbbox([[1,2],[3,4],[0,8],[4,1],-1]);
[[0,4],[1,8]]
[2] os_md.ptbbox([[[1,2],[3,4]],[[0,8],[1,3]]]|box=1);
[[0,8],[1,4]]
[3] os_md.ptbbox([[1,2],[3,4],[0,8],[4,1]]|box=[[1,5],[0,7]]);
[[0,5],[0,8]]
```

331. iscombox([[$x^{(1)}_{min}$,$x^{(1)}_{max}$],[$y^{(1)}_{min}$,$y^{(1)}_{max}$],...], [[$x^{(2)}_{min}$,$x^{(2)}_{max}$],[$y^{(2)}_{min}$,$y^{(2)}_{max}$],...])
:: 2 つの箱（区間の直積）の共通部分の有無を返す

```
[0] S=[[1,2],[4,6]]$
[1] os_md.iscombox(S,[[1.5,3],[4,7]]);
1
[2] os_md.iscombox(S,[[3,3],[4,6]]);
0
```

332. lninbox([p_1,p_2],[[x_{min},x_{max}],[y_{min},y_{max}]]|in=1)
:: 平面内の 2 点 p_1 と p_2 を結ぶ直線（または線分）の箱内の部分（2 点を結ぶ線分）を返す
- 戻り値は，線分を表す両端点の座標のリスト.
- in=1 : これを指定すると，2 点を結ぶ線分となる.

2 点 $(0.2,0.3)$ と $(-1,3)$ とを結ぶ直線（または線分）の $0 \le x \le 1$, $0 \le y \le 1$ 内の部分は,

```
[0] os_md.lninbox([[0.2,0.3],[-1,3]],[[0,1],[0,1]]);
[[0.333333,0],[0,0.75]]
[1] os_md.lninbox([[0.2,0.3],[-1,3]],[[0,1],[0,1]]|in=1);
[[0.2,0.3],[0,0.75]]
```

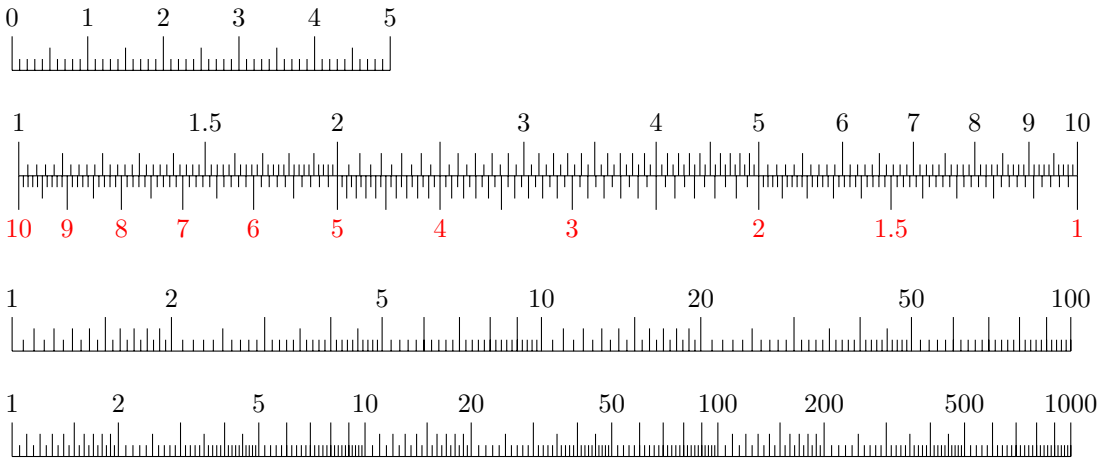
333. scale(ℓ|scale=[c_1,...],f=f,shift=[s_1,s_2],TeX=1,mes=[t,[a_1,b_1,w_1],...],line=[a,b],prec=1,col=s)
:: 目盛や対数（函数）尺の作成
[a,b,w] は a から b までの等幅 w の目盛を表す. これらの目盛を表すリスト（3 つの数字のリスト）$\ell_{i,j}$ に対し，ℓ =[[$\ell_{1,1}$,$\ell_{1,2}$,...],[$\ell_{1,1}$,$\ell_{1,2}$,...],...] が一般的な目盛の指定で，$\ell_{1,\nu}$ が第 1 段階，$\ell_{2,\nu}$ が第 2 段階の目盛，というように表す（より多段階も可能）.
なお，1 段階の目盛のみの時は単純に [a_1,a_2,a_3,...] と表してもよい.
scale(ℓ) は，目盛リスト m =[[$m_{1,1}$,$m_{1,2}$,...],[$m_{2,1}$,$m_{2,2}$,...],...] を返す. ここで，後ろの段

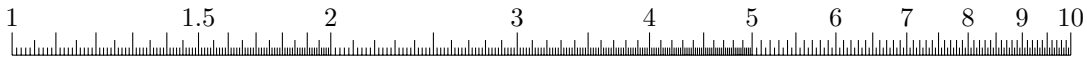
階に現れる目盛は省かれる．

なお，対数尺の目盛のデフォルトは 3 段階で
`[[[1,2,1/50],[5,10,1/2],[5,10,1/10]],[[1,5,1/10],[5,10,1/2]],[[1,5,1/2],[5,10,1]]]`
であるが，`prec=1` を指定すると，倍程度細かくなる． $\ell=0$ または 1 のとき，$\log_{10} x$ $(1\le x\le 10)$ に対する目盛（C 尺に対応）を生成するが，$\ell=2$ のときは $\frac{1}{2}\log_{10} x$ $(1\le x\le 100)$ に対する目盛 (A 尺に対応）を生成し，$\ell=3$ のときは $\frac{1}{3}\log_{10} x$ $(1\le x\le 1000)$ に対する目盛 (K 尺に対応）を生成する．また，CI 尺の目盛を生成するには (c_1, s_1) を $(-c_1, s_1+c_1)$ に変えればよい．

- `scale=`c ： 目盛の位置 `[[`$cf(m_{1,1})+s_1$`,`$cf(m_{1,2})+s_1,\ldots$`],[`$cf(m_{2,1})+s_1$`,`$cf(m_{2,2})+s_1,\ldots$`],...]` を返す．ここで $f(x)$ のデフォルトは自然対数 $\log x/\log 10$.
- `scale=[`c_1`,`$c_2,\ldots$`]`：以下の目盛描画用パラメータを返す（`xylines()` の引数）．
  なお，`shift=[`s_1`,`s_2`]` によって描画の相対位置を指定する．
  `[[`$c_1f(m_{1,1})+s_1$`,`s_2`],[`$c_1f(m_{1,1})+s_1$`,`s_2`+`c_2`],0,` `[`$c_1f(m_{1,2})+s_1$`,`s_2`],[`$c_1f(m_{1,2})+s_1$`,`s_2`+`c_2`],0,...],` `[`$c_1f(m_{2,1})+s_1$`,`s_2`],[`$c_1f(m_{2,1})+s_1$`,`s_2`+`c_3`],0,` `[`$c_1f(m_{2,2})+s_1$`,`s_2`],[`$c_1f(m_{2,2})+s_1$`,`s_2`+`c_3`],0,...],...]`
  $c_1, c_2, \ldots$ は負の値も設定可能．なお c_3 以降がないときは，c_n のデフォルトは $(n-1)c_2$.
- `TeX=1`：目盛描画の TEX のソースを返す（戻り値で `xyproc()` を呼ぶ）．
  目盛線の色づけなどを行うときは，`col="red"`などとオプションで指定する（Ti*k*Z を用いるとき）．
  このとき，`mes=[`t`,[`a_1`,`b_1`,`w_1`],...]`，あるいは `mes=[`t`,`c_1`,`c_2`,`c_3`,...]` により，それ以降で指定した目盛の数字を，高さ t の位置に描く．なお，目盛の数字の色付けなどを指定するときは，t を `[`t`,"red"]` などと，オプション文字列を指定する．
  `line=[`a`,`b`]` によって基準線を a から b まで描く．

```
[0] S=os_md.scale([[[0,5,1/10]],[[0,5,1/2]],[[0,5,1]]]|scale=[1,0.15],f=x,
    TeX=1,line=[0,5])$
[1] S=os_md.scale(0|scale=[14,0.15],TeX=1,mes=[0.7,[1,2,1/2],[2,10,1]])$
[2] S+=os_md.scale(0|scale=[-14,-0.15],TeX=1,shift=[14,0],line=[-14,0],
    mes=[[-0.7,"red"],[1,2,1/2],[2,10,1]])$
[3] os_md.xyproc(S|dviout=1)$
[4] S=os_md.scale(2|scale=[14,0.15],TeX=1,mes=[0.7,1,2,5,10,20,50,100],
    line=[0,14])$
[5] S=os_md.scale(3|scale=[14,0.15],TeX=1,mes=[0.7,1,2,5,10,20,50,100,200,
    500,1000],line=[0,14])$
[6] S=os_md.scale(0|scale=[14,0.1],TeX=1,mes=[0.5,[1,2,1/2],[2,10,1]],
    line=[0,14],prec=1)$
```





334. `tobezier(`ℓ`|inv=[`a`,`b`,`t`],div=`k`)`

:: 点のリスト ℓ で定まる Bézier 曲線のパラメータ表示とその逆変換と分割

- $\ell=[p_0,\dots,p_n]$ で，p_j が座標を表すリストまたはベクトルのとき，これらによって定まる Bézier 曲線（パラメータは `t` $\in[0,1]$）を，`t` の n 次多項式を成分とするベクトル）を返す.

  Bézier 曲線とは

$$\gamma(t)=\gamma(p_1,\dots,p_n;t)=\sum_{j=0}^{n}\binom{n}{j}t^j(1-t)^{n-j}p_j\quad(t\in[0,1]),\quad \binom{n}{j}=\frac{n!}{j!(n-j)!}$$

  で定まる曲線である（このとき n 次 Bézier 曲線という）.

  $\gamma(p_0,p_1;t)=(1-t)p_0+tp_1$ は線分 p_0p_1 を $t:(1-t)$ に内分する点で，1 次 Bézier 曲線は p_0 と p_1 を結ぶ線分となる．また平面内の 2 次 Bézier 曲線は，x 座標と y 座標が t の 2 次（以下）の多項式となるが，平面を回転すると x 座標は t の 1 次式にできる．よって，2 次 Bézier 曲線は放物線の一部（または線分）を回転したものとなる.

  一方，$\binom{n}{j}=\binom{n-1}{j}+\binom{n-1}{j-1}$ であるから

$$\tbinom{n}{j}t^j(1-t)^{n-j}=(1-t)\cdot\tbinom{n-1}{j}t^j(1-t)^{n-1-j}+t\cdot\tbinom{n-1}{j-1}t^{j-1}(1-t)^{n-1},\quad \tbinom{n-1}{-1}=\tbinom{n-1}{n}=0$$

  となり，p_j にこれを掛けて $j=0,\dots,n$ について加えると

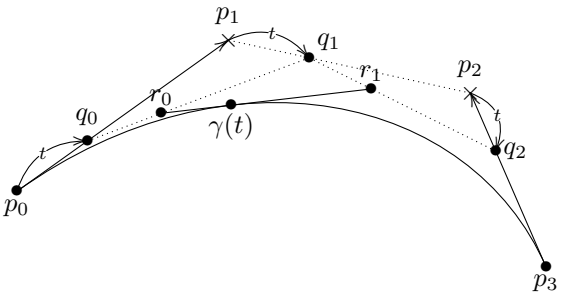
$$\gamma(p_0,\dots,t_n:t)=\gamma\big(\gamma(p_0,\dots,p_{n-1};t),\gamma(p_1,\dots,p_n;t);t\big)$$

  がわかる．これを遡れば，元の点 $\{p_0,\dots,p_n\}$ の $t:(1-t)$ の内分点を取ることを繰り返すことにより $\gamma(p_0,\dots,p_n:t)$ が得られる.

  たとえば 3 次 (cubic)Bézier 曲線は

$$\begin{aligned}\gamma(p_0,p_1,p_2,p_3;t)&=\gamma\big(\gamma(p_0,p_1,p_2;t),\gamma(p_1,p_2,p_3;t);t\big)\\&=\gamma\Big(\gamma\big(\gamma(p_0,p_1;t),\gamma(p_1,p_2;t);t\big),\gamma\big(\gamma(p_1,p_2;t),\gamma(p_2,p_3;t),t\big);t\Big)\end{aligned}$$

  となる．よって p_0p_1, p_1p_2, p_3p_4 の $t:(1-t)$ の内分点を順に q_0, q_1, q_2 とし，さらに q_0q_1 と q_1q_2 の $t:(1-t)$ の内分点を r_0, r_1 とすると r_0r_1 の $t:(1-t)$ の内分点が $\gamma(p_0,p_1,p_2,p_3;t)$ となる.



- `inv=[`a`,`b`]`：パラメータが `t` $\in[a,b]$ の部分の Bézier 曲線に対応する座標（ベクトルで表す）のリストを返す.
- `div=1`：Bézier 曲線を 2 分割して，それらを定義する座標の組のリストを返す（3 次のときは，ベクトルの 4 個のリストの 2 つのリストで，前者が $[0,\frac{1}{2}]$ の部分，後者が $[\frac{1}{2},1]$ の部分）．3 次のみ高速化対応.
- `div=`k：$2\le k\le 256$ のとき，Bézier 曲線を k 分割して，それらを定義する座標の組のリスト k 個のリストを返す.

- inv=[a,b] : ℓ が Bézier 曲線のとき，それを t $\in [a,b]$ をパラメータとする Bézier 曲線とみなして，それを定義する点の座標（ベクトルで表す）のリストで返す．パラメータが t でないときは，それを inv=[a,b,t] のようにして明示することができる．
- inv=1 : inv=[0,1] とみなす．tobezier( ) の逆変換となる．

```
[0] S=os_md.tobezier([[p0,q0],[p1,q1],[p2,q2],[p3,q3]]);
[ (-p0+3*p1-3*p2+p3)*t^3+(3*p0-6*p1+3*p2)*t^2+(-3*p0+3*p1)*t+p0
  (-q0+3*q1-3*q2+q3)*t^3+(3*q0-6*q1+3*q2)*t^2+(-3*q0+3*q1)*t+q0 ]
[1] T=os_md.tobezier(@@|inv=1);
[[ p0 q0 ],[ p1 q1 ],[ p2 q2 ],[ p3 q3 ]]
[2] os_md.tobezier(T|inv=[0,1/2]);          /* Bezier 曲線の 2 分割の前半 */
[[ p0 q0 ],[ 1/2*p0+1/2*p1 1/2*q0+1/2*q1 ],
[ 1/4*p0+1/2*p1+1/4*p2 1/4*q0+1/2*q1+1/4*q2 ],
[ 1/8*p0+3/8*p1+3/8*p2+1/8*p3 1/8*q0+3/8*q1+3/8*q2+1/8*q3 ]]
```

335. lbezier(l|inv=t)
:: 区分 Bézier 曲線のデータ変換
xybezier() のデータ形式（座標と，整数 0, 1, -1 のリスト）を，区分 Bézier 曲線を与える座標のリストを成分とするリストに変換する．
- inv=1 : 上の変換の逆変換を行う．区分 Bézier 曲線をつなげる指定 1 や，閉曲線を描く指定 -1 を可能な限り使う．
- inv=2 : 上と同様だが，閉曲線を描く指定を使わない．
- inv=3 : 上と同様だが，閉曲線を描く指定は，全体として一つの閉曲線となる場合のみ使う．

```
[0] P=os_md.xyoval([1,3.5],2.5,2|opt=0);  /* 中心 (1,3.5) 半経 2.5x5 の楕円 */
[[3.5,3.5],[3.5,7.349],[1.41667,9.75463],[-0.25,7.83013],1,[-1.91667,5.90563],
[-1.91667,1.09437],[-0.25,-0.830127],1,[1.41667,-2.75463],[3.5,-0.349002],-1]
[1] Q=os_md.lbezier(P);
[[[3.5,3.5],[3.5,7.349],[1.41667,9.75463],[-0.25,7.83013]],
[[-0.25,7.83013],[-1.91667,5.90563],[-1.91667,1.09437],[-0.25,-0.830127]],
[[-0.25,-0.830127],[1.41667,-2.75463],[3.5,-0.349002],[3.5,3.5]]]
[2] os_md.lbezier(Q|inv=1);
[[3.5,3.5],[3.5,7.349],[1.41667,9.75463],[-0.25,7.83013],1,[-1.91667,5.90563],
[-1.91667,1.09437],[-0.25,-0.830127],1,[1.41667,-2.75463],[3.5,-0.349002],-1]
```

336. velbezier(f,[a,b,t])
:: Bézier 曲線の最大速度ベクトル
- $t \in [a,b]$ をパラメータとする Bézier 曲線 f の成分毎の最大速度をリストで返す．
- 2 番目の引数が 0 のときは，デフォルトの [0,1,t] を意味する．
- パラメータが t のときは，2 番目の引数を [a,b] としてよい．

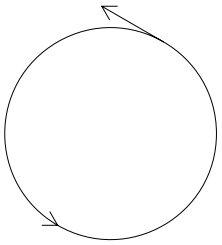
```
[0] B=os_md.tobezier([[0,0],[1.2,0.7],[2.1,0.5],[3,0]]);
[ 0.3*t^3-0.9*t^2+3.6*t 0.6*t^3-2.7*t^2+2.1*t ];
[1] os_md.velbezier(B,0);
[3.6,2.1]
[2] os_md.velbezier(B,[0.5,1]);
[2.925,1.5]
```

337. ptbezier(l,[n,t]) ptbezier(l,s)

:: （複合）Bézier 曲線上の点と速度ベクトルを求める

- l は，xybezier() の引数で与える（複合）Bézier 曲線のデータ，またはそれを lbezier() で変換した形
- n 番目の Bézier 曲線の t $= x$ の点の座標と速度ベクトルの組を返す.
- $n = 0$ は最初の Bézier 曲線，$n = 1$ は 2 番目，$n = -1$ は最後の，$n = -2$ は最後から 2 番目の Bézier 曲線を表す.
- 2 番目の引数 s が非負実数の時は，n は s の整数部分 $+1$ を，t は s の小数部分を表す.
  ただし x が Bésier 曲線の個数より大きいときは，（複合）Bézier 曲線の終点を表す.
- 2 番目の引数 x が -1 のときは，Bésier 曲線の個数を返す.

```
[0] R=os_md.xyang(2,[1,1],0,0|opt=0)$  /* (1,1) 中心の半径 2 の円のデータ */
[1] os_md.ptbezier(R,-1);
3                                      /* 3 つのベジェ曲線からなる , 0 <= s <= 3 */
[2] S=os_md.ptbezier(R,0.5);           /* s=0.5 での座標と速度ベクトル     */
[[ 2 2.73205 ],[-3.5,2.02073]]
[3] T=os_md.ladd(S[0],S[1],1/3);       /* 速度ベクトルの頂点（長さ 1/3） */
[0.833333,3.40563]
[4] R1=os_md.xyang(0.3,T,S[0],2|ar=1,opt=0)$   /* s=0.5 での速度ベクトルの描画  */
[5] U=os_md.ptbezier(R,2)$ /* s=2 での座標と速度ベクトル */
[6] V=[U[0][0]-U[1][0],U[0][1]-U[1][1]]$
[7] R2=os_md.xyang(0.3,U[0],V,3|opt=0)$        /* s=1.5 に曲線上の矢印の描画     */
[8] Out=os_md.ptaffine("union",[R,R1,R2])$
[9] os_md.dviout(os_md.xyproc(Out));
```



338. areabezier(l|rev=1,pt=[p_1,p_2,...],para=1,prec=v,int=k,exp=c,Acc=1,cpx=1)

:: Bézier 曲線を用いた領域の面積・数値積分の計算

- l が [f,n,[t_1,t_2]] という xygraph() の最初の 3 つの引数であって，f が変数 x の函数のときは，Bézier 曲線で近似することによって積分 $\int_{t_1}^{t_2} f(x)\,dx$ を近似計算する.
  - int=1 を指定すると，$|n|$ 分割して台形公式で近似計算する.
  - int=2 を指定すると，$|n|$ 分割してシンプソンの公式で近似計算する．このとき，n が奇数なら $|n|+1$ 分割する。
  - cpx=1：非積分関数が複素数値函数であるときに指定する.
  - 変数が x でなくて，たとえば t のときは，[t,t_1,t_2] と明示する.
  - 等分割の個数は $|n|$ となる（$n = 0$ は，32 と解釈される）.
  - 積分区間外まで函数が定義されているときは，n を負の数にする（n が分割の数）.
  - t_1 ="-infty"（""でも可）とすると $a = -\infty$ と，t_2 ="infty"（""でも可）とすると $b = \infty$ と解釈され，無限区間での積分となる.
    無限区間での積分のとき exp=c の指定が有効（cf. cmpf()）.
  - 無限区間での積分において無限遠で $o(|x|^{-2})$（有理函数のときは $O(|x|^{-2})$ を満たさないとき）は誤差が大きくなるので，prec=16 または exp=1 などと指定すると改善される.
- 函数に滑らかでない点や不連続点がある場合は，prec=16 などと指定すると計算誤差が少なく

なる．

- f がリストで f=[f_1,f_2] というパラメータ表示の曲線のときは，3 番目の要素が [t,t_1,t_2] ならば，その曲線を Bézier 曲線で近似した曲線に沿って，積分 $\int_{t_1}^{t_2} f_2(t)\,df_1(t)$ を計算する．
  特にこの Bézier 曲線が閉曲線のときは，その曲線で囲まれた領域の面積の計算となる．このとき，時計回りかどうかで正負が決まり，反時計回りに 1 回囲まれた部分の面積は -1 倍される．
- Acc=1 を指定すると pari() による高精度計算を行う．
  ただし ctrl("bigfloat",1) および setprec(*prec*) による精度の設定が必要．また，有理関数でなくて $\sin(x)$ などの初等関数を扱う場合は注意が必要で，不定元が大量に生成される可能性がある（myeval() の項を参照．ord() で分かる）．
- 以上の時のオプションパラメータは xygraph() のときと同じ意味に解釈される．
- l が lbezier() の引数となる Bézier 曲線 C のデータ（2 形式のいずれでも可）ならば，線積分 $\int_C ydx$ の値を返す．
  閉曲線のデータであったなら，その曲線で囲まれた領域の面積となる．

```
[0] os_md.areabezier([x^3,-4,[0,2]]);
4.01564
[1] os_md.areabezier([x^3,-8,[0,2]]);
4.00105
[2] os_md.areabezier([x^3,-16,[0,2]]);
4.00007
[3] os_md.areabezier([x^3,-32,[0,2]]);
4
[4] os_md.areabezier([sin(x),-4,[0,@pi]]);
1.98989
[5] os_md.areabezier([sin(x),-8,[0,@pi]]);
1.99935
[6] os_md.areabezier([sin(x),-16,[0,@pi]]);
1.99996
[7] os_md.areabezier([sin(x),-32,[0,@pi]]);
2
[8] os_md.areabezier([[cos(x),-sin(x)],-16,[0,2*@pi]]);
3.14159
[9] os_md.areabezier([exp(-x),16,[0,"infty"]]);
0.999982
[10] tstart$os_md.areabezier([exp(-x),64,[0,"infty"]]);tstop$
[11] 1
[12] 0.0156sec(0.016sec)
[13] os_md.areabezier([exp(-x^2),32,["-infty","infty"]]);
1.77268
[14] tstart$R=os_md.areabezier([exp(-x^2),512,["",""]]);tstop$
[15] 1.77245
[16] 0.0312sec + gc : 0.0312sec(0.063sec)
[17] V=eval(@pi^(1/2));
1.77245385090551602720251866962
[18] R-V;
0.00000000610225522239618567 43
```

```
[19] os_md.areabezier([x^(-3/2),0,[1,""]]);
1.90184
[20] os_md.areabezier([x^(-3/2),0,[1,""]]|prec=16);
1.99991
[21] os_md.areabezier([x^(-3/2),0,[1,""]]|exp=1);
2.00008
[22] os_md.areabezier([1/(1+x^2),0,["",""]]);
3.14159
[23] os_md.areabezier([1/(1+x^2),0,["",""]])-eval(@pi);
0.00000040239333897167800513
[24] os_md.areabezier([1/(1+x^2),1000,["",""]])-eval(@pi);
-3.3363329451 E-13
[25] tstart$os_md.areabezier([1/(1+x^2),5000,["",""]])-eval(@pi);tstop;
[26] -5.6638721488 E-16
[27]  0.2964sec + gc : 0.2496sec(0.546sec)
[28] os_md.areabezier([1/(1+x^2),0,["",""]]|exp=1);
3.14151
[29] os_md.areabezier([1/(1+x^2),-10,["",""]]|exp=1);
3.1417
[30] tstart$os_md.areabezier([1/(1+x^2),10000,["",""]]|exp=1)-eval(@pi);
tstop$
[31] -9.07164967805 E-14
[32] 0.8268sec + gc : 0.4056sec(1.248sec)
[33] V=eval(@pi*2^(-1/2));
2.22144146907918312331
[34] os_md.areabezier([1/(1+x^4),0,["",""]]);
2.22151
[35] os_md.areabezier([1/(1+x^4),0,["",""]]|exp=1);
2.22192
[36] os_md.areabezier([1/(1+x^4),10000,["",""]])-V;
-8.6669820621 E-14
[37] os_md.areabezier([1/(1+x^4),10000,["",""]]|exp=1)-V;
-4.6270379014 E-12
[38] os_md.areabezier([dsqrt(2)/(x^2+@i),0,["",""]]|cpx=1);
(3.14247-3.1418*@i)
[39] os_md.areabezier([dsqrt(2)/(x^2+@i),1000,["",""]]|cpx=1);
(3.14159-3.14159*@i)
[40] ctrl("bigfloat",1)$setprec(30);
19
[41] F=[cos(x),-sin(x)]$
[42] tstart$os_md.areabezier([F,-1536,[0,2*@pi]])/eval(@pi)-1;tstop$
[43] -3.17671236538252799164 E-17
[44] 0.2496sec + gc : 0.0936sec(0.343sec)
[45] tstart$os_md.areabezier([F,-1536,[0,2*@pi]]|Acc=1)/eval(@pi)-1;tstop$
```

```
[46] 8.71510564319365765590 E-20
[47] 0.4212sec + gc : 0.0624sec(0.5sec)
```

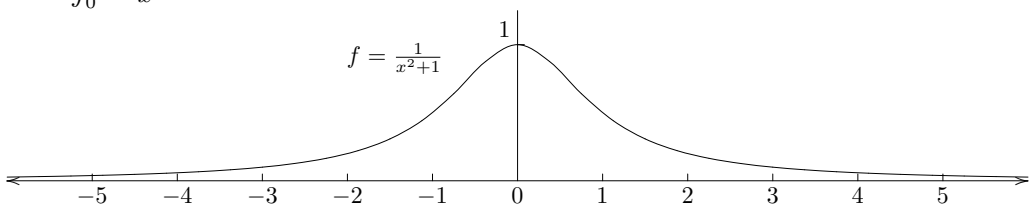
Bézier 曲線を使った数値積分の相対誤差

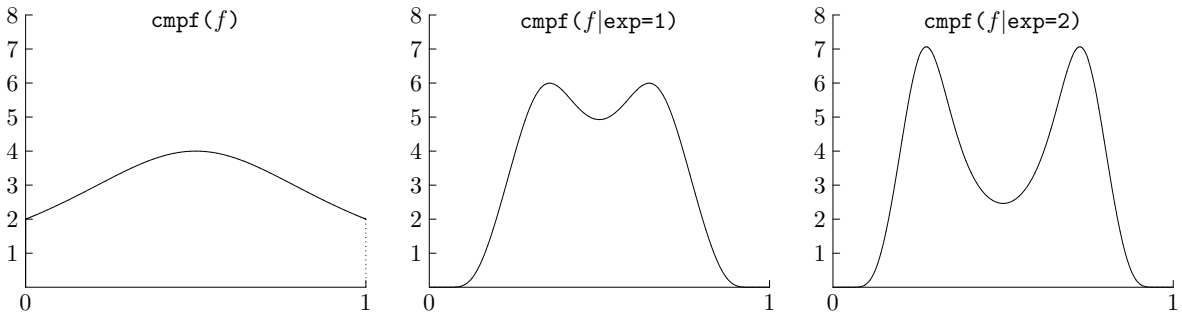
非積分関数	積分範囲	指定	16 分割	32 分割	96 分割	384 分割	1536 分割
$(\cos\theta, \sin\theta)$	$0 \le \theta \le 2\pi$	$(-n, -)$	6.8×10^{-8}	1.1×10^{-9}	1.5×10^{-12}	3.2×10^{-17}	8.7×10^{-20}
Cardioid	$-\pi \le \theta \le \pi$	$(n, -)$	5.4×10^{-4}	3.1×10^{-5}	3.8×10^{-7}	1.5×10^{-9}	5.8×10^{-12}
$x \sin x$	$0 \le x \le \pi$	$(-n, -)$	2.9×10^{-4}	1.8×10^{-6}	2.2×10^{-8}	8.7×10^{-11}	3.4×10^{-13}
$\frac{\sin x}{x}$	$0 < x \le \pi$	$(-n, -)$	1.5×10^{-6}	9.5×10^{-8}	1.2×10^{-9}	4.6×10^{-12}	1.7×10^{-14}
$\frac{1}{x^2+1}$	$-\infty < x < \infty$	$(n, -)$	1.3×10^{-5}	1.3×10^{-7}	8.5×10^{-10}	4.7×10^{-12}	2.1×10^{-14}
e^{-x^2}	$-\infty < x < \infty$	$(n, -)$	7.1×10^{-4}	1.3×10^{-4}	2.6×10^{-6}	1.1×10^{-8}	4.3×10^{-11}
$x^{-\frac{3}{2}}$	$1 \le x < \infty$	$(n, -)$	7.1×10^{-2}	4.9×10^{-2}	2.8×10^{-2}	1.4×10^{-2}	7.0×10^{-3}
$x^{-\frac{3}{2}}$	$1 \le x < \infty$	$(n, -)^{16}$	5.9×10^{-6}	4.5×10^{-5}	2.8×10^{-5}	1.4×10^{-5}	6.9×10^{-6}
$x^{-\frac{3}{2}}$	$1 \le x < \infty$	$(n, 1)$	3.0×10^{-4}	3.8×10^{-5}	1.4×10^{-6}	6.6×10^{-9}	2.6×10^{-11}

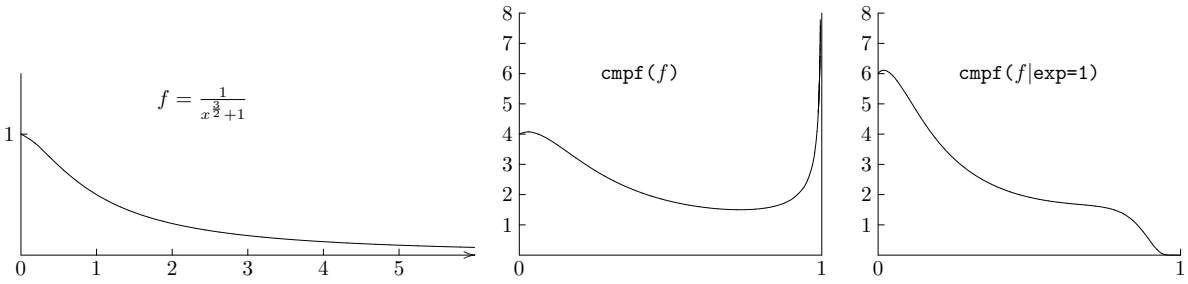
- 上表の最初は円の面積の数値計算例で，2 番目の Cardioid は $\big((1+\cos\theta)\cos\theta, (1+\cos\theta)\sin\theta\big)$ でカージオイド曲線で囲まれた領域の面積の数値計算例である．
- n 分割による計算で，上表の指定の欄の $\pm n$ は `areabezier()` の引数の 2 番目を意味する．
- $(n, -)^{16}$ は `prec=16` のオプションの指定を意味する．
- $(n, 1)$ は `exp=1` のオプションを指定したことを意味する．
  無限区間での積分では，よい近似には `exp=`c のオプションパラメータ c の指定が必要だが，最後の例（$x \to \infty$ で $o(x^{-2})$ とならない例）のみ標準的な指定をした以外は指定をしていない．
- 上において分割の数が大きくて最も時間がかかる場合の計算時間は，0.1 sec 程度である．
- $\frac{\sin x}{x}$ の $(0, \pi]$ での数値積分は以下のように行った．

```
[0] F=os_md.f2df(sin(x)/x)$
[1] F=os_md.cutf(F,x,[[],[0,1],[]])$      /* x=0 のとき 1 となるように拡張 */
[2] os_md.areabezier([F,-32,[0,@pi]]);
1.85194
```

なお $\displaystyle\int_0^{\pi} \frac{\sin x}{x}\,dx = 1.8519370519824661706\cdots$







上のグラフは，次のようにして得られる TEX のソースで描かれている：

```
os_md.xygraph(1/(1+x^2),-32,[-6,6],0,[0,1.5]|ax=[0,0,1,1,1],scale=[1.25,2]);
R=os_md.cmpf([1/(1+x^2),["",""]]|exp=1);
os_md.xygraph(R,-64,[0,1],0,[0,8]|ax=[0,0,1,1,1],scale=[5,0.5],prec=6);
```

scale=の指定で $1.25\times 2=5\times 0.5$ であるから，図の面積が保たれることに注意.

339. ptcombezier(ℓ_1,ℓ_2,m)

:: 点のリスト ℓ_1, ℓ_2 で定まる 2 つの Bézier 曲線の交点を求める

- ℓ_1, ℓ_2 は，3 次 Bézier 曲線を想定している（それ以外でも可）
- m は交点の正確さ（パラメータを 2^m 分割する）．$m=0$ はデフォルトで，$m=20$ を意味する.
- 交点は，対応する ℓ_1 のパラメータ値，ℓ_2 のパラメータ値，交点の座標の 3 つ組のリストで表され，交点の数だけのそれのリストを返す.

```
[0] L1=[[0,0],[1,1],[2,2],[3.0,3.0]]$
[1] L2=[[3.0,0],[2,1],[1,2],[0,3.0]]$
[2] os_md.ptcombezier(L1,L2,24);
[[0.5,0.5,[1.5,1.5]]]
```

340. ptcombz(b_1,b_2,m|red=t,prec=k)

:: 区分 Bézier 曲線 b_1, b_2 の交点を求める

- b_1, b_2 は，xybezier() の引数の形のデータ，あるいは lbezier で変換された区分 Bézier 曲線を与える座標のリストのリストのいずれか.
- $b_2=0$ のときは，b_1 で与えられた曲線の自己交叉点を返す．ただし各区分 Bézier 曲線自身や隣り合った区分 Bézier 曲線の交点は除く.
- b_1 と b_2 のそれぞれ何番目の区分 Bézier 曲線かのリスト（最初を 0 番目と数える）を先頭に付加した ptcombezier() の戻り値をまとめてリストの成分としたリストを返す.
- m は交点の正確さ（パラメータを 2^m 分割する）．$m=0$ はデフォルトで，$m=20$ を意味する.
- red=1：ほぼ等しい座標の点が複数あるときは，そのうちの一つデータのみ返す.
- red=2：交点の座標のみ返す（ほぼ等しい座標の点は重複して表示しない）.
- prec=k：オプション red を指定したとき，x 座標および y 座標の差が曲線の幅（現れる座標の x 座標の最大値と最小値の差，および y 座標の最大値と最小値の差）の $1/2^k$ 程度以下ならば，同じ点と見なす．デフォルトは $k=12$.

```
[0] P=os_md.xyoval([0,0],1,1|opt=0)$        /* 中心 (0,0) 半径 1 の円        */
[1] Q=os_md.xyoval([1,1],1.2,0.5|opt=0)$    /* 中心 (1,1) 半経 1.2x0.6 の楕円 */
[2] os_md.ptcombz(P,Q,0);
[[[0,1],[0.853506,0.515474,[-0.199425,0.981241]]],  /* 交点 (-0.200,0.981) */
 [[0,2],[0.188392,0.209443,[0.917957,0.400476]]]]   /* 交点  (0.918,0.400) */
[3] F=[sin(2*x),sin(3*x)]$
```

```
[4] LS=os_md.xygraph(F,-24,[0,2*@pi],[-2,2],[-2,2]|opt=0)$   /* リサージュ図形 */
[5] tstart$os_md.sint(os_md.ptcombz(LT,0,0),4);tstop;
[[[12,0],[0,0,[0,0]]],[[13,4],[0,1,[0.5,-0.7071]]],[[13,5],[0,0,[0.5,-0.7071]]],
 [[16,0],[1,1,[0.5,0.7071]]],[[16,1],[1,0,[0.5,0.7071]]],[[16,3],[0,1,[0.866,0]]],
 [[16,4],[0,0,[0.866,0]]],[[19,8],[1,0,[-0.866,0]]],[[19,10],[0,1,[-0.5,0.7071]]],
 [[19,11],[0,0,[-0.5,0.7071]]],[[20,8],[0,0,[-0.866,0]]],
 [[22,6],[1,1,[-0.5,-0.7071]]],[[22,7],[1,0,[-0.5,-0.7071]]]]
[6] 0.0312sec(0.047sec)
[7] LT=os_md.xygraph(F,-96,[0,2*@pi],[-2,2],[-2,2]|opt=0)$
[8] tstart$os_md.ptcombz(LT,0,0)$tstop;
[9] [10] 0.2028sec + gc : 0.078sec(0.281sec)
[11] os_md.sint(os_md.ptcombz(LS,0,0|red=1),4);
[[[12,0],[0,0,[0,0]]],[[13,5],[0,0,[0.5,-0.7071]]],[[16,0],[1,1,[0.5,0.7071]]],
 [[16,4],[0,0,[0.866,0]]],[[19,10],[0,1,[-0.5,0.7071]]],[[20,8],[0,0,[-0.866,0]]],
 [[22,7],[1,0,[-0.5,-0.7071]]]]
[12] os_md.sint(os_md.ptcombz(LS,0,0|red=2),4);
[[0,0],[0.5,-0.7071],[0.5,0.7071],[0.866,0],[-0.5,0.7071],[-0.866,0],
[-0.5,-0.7071]]
```

3.2.11 Drawing curves and graphs

341. xyproc(f|dviout=1,opt=s,env=t)
:: XY-pic/TikZ や指定した環境の開始（$f=1$）と終了（$f=0$）や表示
- f が文字列の時は，TikZ に応じて\begin{xy}と\end{xy}または\begin{tikzpicture}と\end{tikzpicture}で鋏む．
  さらに dviout=1 が指定してあれば dviout() を使って画面表示する．
- opt=s で文字列 s を指定してあれば\begin{tikzpicture}[s] などのようにオプションをつける．
- env=t で環境 xy などを別の環境に変更できる（たとえば，env="scope"）．
- dviout=1 で画面表示などがなされる．
- xyput(), xyline(), xyarrow(), xybox(), xycirc(), xylines(), xygraph() などで得た文字列を（str_tb() などを使って）足し合わせてこの函数を呼ぶとよい．

```
[0] os_md.xyproc(1);
\begin{xy}

[1] os_md.xyproc(0);
\end{xy}
```

342. xypos([x,y,s]) xypos([x,y,s,t]) xypos([x,y,s,t,u]) xypos([x,y])
:: XY-pic/TikZ での座標 (x,y) や式 s などの文字出力．t はラベルの文字，u はオプション文字列．
- 引数はベクトルでもよい
- xypos([x,y]) のとき，以下の文字列が返される
  (x,y)
  ただし数字 x, y は小数点以下 XYPrec 桁に丸められる．
- s が文字列なら，テキストとして表示する．
- x が文字列ならばラベルとみなし，y は無視される．このラベルは TikZ=0 のときは"x"の形で，TikZ=1 のときは (x) の形に変換されるが，x が空文字列なら空文字列でラベル無しとなる．
- TikZ=0 の場合

– t は一文字．空文字列 "" のとき出力しない．
– u はオプション文字列でラベルの後につく．0 などとすると出力しない．
– $s = [s_1, s_2]$ とすると，s_1 は前置されるオプション文字列，s_2 は出力文字列を表す．
  * s_1 ="[F]"：枠で囲む
  * s_1 ="[F.]"：囲む枠が点線（. を = とすると二重線）
  * s_1 ="+[F]"：余裕のある枠で囲む
  * s_1 ="[Fo]"：丸く囲む（o を oo とすると二重丸）
  * s_1 ="+!U"：下に置く．U を D, L, R とすると上，右，左に変わる．
    さらに UL などとすると，右下などとなる．
    + を ++ とすると，位置のずらしが大きくなる．

• TikZ=1 の場合
– xypos([x,y,[s_1,s_2],s_t,s_u]) で，x, y が数，s_1, s_2, s_t, s_u が文字列で s_2 が空でないとき
  node[s_1](s_t) at(x,y){s_2}s_u
– xypos([x,y,[s_1,""],s_t,s_u]) のとき
  coordinate[s_1](s_t) at (x,y)s_u
– xypos([x,y,[s_1,s_2],s_t]) または xypos([x,y,s,s_t]) のときは，上の s_u の項は現れない．
– s_t が数字 1 のときは，s_t ="_" と解釈される．
– s_t が空文字列の時，または xypos([x,y,[s_1,s_2]]) や xypos([x,y,s]) のとき，上の (s_t) の項は現れない．
– s や s_1 が文字列でないときは数式として TEX の文字列に変換される．
– s_1 にはたとえば以下のオプション文字が可能（複数の指定は , で区切る）．
  * 位置指定：above, below, right, left, above right,...
    below=1pt,...
    anchor=west, anchor=south east,...
  * 形状指定：circle, rectangle,...
  * 指定した形状を描く：draw
  * テキストボックスの幅指定 ：width=3cm
  * テキストボックスの角を丸める ：rounded corners
  * テキストボックス内の文字位置指定 ：text centered
  * 色指定 ：red, green, blue, green!20!white,...
  * 塗りつぶし：fill=red, fill=green,...

```
[0] os_md.xypos([2/7,5/3]);
(0.285714,1.66667)
[1] os_md.dviout0(0|opt="TikZ")$
TikZ=0
[2] os_md.xypos([2/7,5/3,4/5]);
(0.285714,1.66667) *{\frac{4}5}
[3] os_md.xypos([2.5,3.1,"$\\bullet$"]);
(2.5,3.1) *{\bullet}
[4] os_md.xypos([2.5,3.1,"This is"]);
(2.5,3.1) *\txt{This is}
[5] os_md.xypos([2,3,"$\\times$","A"]);
(2,3) *{\times}="A"
[6] os_md.xypos([2,3,"","A"]);
(2,3)="A"
[7] os_md.xypos([1,2,["+[F]","Sum"],"S"]);
(1,2) *+[F]\txt{Sum}="S"
```

```
[8] os_md.xypos(["A","B",["+[F]","Sum"],"S"]);
"A" *+[F]\txt{Sum}="S"
[9] os_md.dvioutO(1|opt="TikZ")$
TikZ=1
[10] os_md.xypos([2/7,5/3,4/5]);
node at(0.2857,1.6667){$\frac{4}5$}
[11] os_md.xypos([2.5,3.1,"$\\bullet$"]);
node at(2.5,3.1){$\bullet$}
[12] os_md.xypos([2.5,3.1,"This is"]);
node at(2.5,3.1){This is}
[13] os_md.xypos([2,3,"","A"]);
coordinate(A) at(2,3)
[14] os_md.xypos([1,2,["red","Sum"],"S"]);
node[red](S) at(1,2){Sum}
[15] os_md.xypos(["A","B",["draw,rectangle","Sum"],"S"]);
node[draw,rectangle](S) at(A){Sum}
[16] os_md.dvioutO(3|opt="XYPrec")$
XYPrec=3
[17] os_md.xypos([@pi,5/3,4/5]);
node at(3.142,1.667){$\frac{4}5$}
```

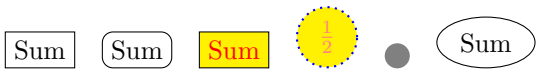
343. xyput([x,y,s]|scale=r)　xyput([x,y,s,t])　xyput([x,y,s,t,u])　xyput([x,y])
:: XY-pic/TikZ での座標 (x,y) や式 s などの文字出力．t はラベルの文字，u はオプション文字列．

- xypos() と同じだが，その結果の前後に
  TikZ=0 のときは "{" および "};\n"
  を
  TikZ=1 のときは "\\" および ";\n"
  を，付加して返す．
- scale=r：XY-pic/TikZ の座標に直すときに r 倍する．
- scale=[r_1,r_2]：XY-pic/TikZ の座標に直すときに x 座標を r_1 倍，y 座標を r_2 倍する．

```
[0] os_md.dvioutO([4,6])$
DVIOUTA="%ASIRROOT%\bin\risatex.bat"
TikZ=1
[1] os_md.xyput([0,0,["draw,rectangle","Sum"]]);
\node[draw,rectangle] at(0,0){Sum};
[2] os_md.xyput([0,0,["draw,rectangle,rounded corners","Sum"]]);
\node[draw,rectangle,rounded corners] at(0,0){Sum};
[3] os_md.xyput([0,0,["draw,rectangle,fill=yellow,text=red","Sum"]]);
\node[draw,rectangle,fill=yellow,text=red] at(0,0){Sum};
[4] os_md.xyput([0,0,["draw,circle,thick,dotted,blue,fill=yellow,text=red!50",
    "$\\tfrac12$"]]);
\node[draw,circle,thick,dotted,blue,fill=yellow,text=red!50] at(0,0){$\tfrac12$};
[5] os_md.xyput([0,0,["circle,radius=2pt,fill=gray",""]]);
\node[circle,radius=2pt,fill=gray] at(0,0){};
[6] os_md.xyput([0,0,["draw,ellipse","Sum"]]);
```

```
\node[draw,ellipse] at(0,0){Sum};
```

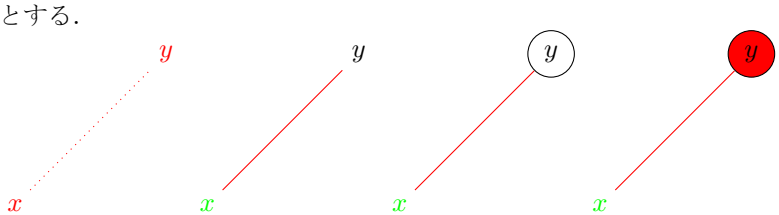
上の [6] は, TEX ファイルの初めに
```
\usetikzlibrary{shapes}
```
が必要である.
上の結果を S とおいて，os_md.xyproc(S|dviout=1) とすると以下の表示が得られる.



344. xyline([x_1,y_1,s_1],[x_2,y_2,s_2]|opt=t)　xyline([x_1,y_1],[x_1,y_2]|opt=t)
:: Xy-pic/TikZ で (x_1, y_1) と (x_2, y_2) を線で結ぶ（引数はベクトルでもよい）
[x_1,y_1,s_1,t_1,u_1] などとするとラベル t_1 がつく（cf. xypos()）.
TikZ のときは，opt=t によって色や線種などのオプション指定文字が有効.

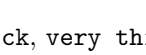
```
[0] os_md.dviout0(0|opt="TikZ")$
TikZ=0
[1] os_md.xyline([1,2,x],[3,4,y]);
{(1,2) *{x} \ar@{-} (3,4) *{y}};
[2] os_md.xyline([1,2,x,"A"],[3,4,y,"B"]);
{(1,2) *{x}="A" \ar@{-} (3,4) *{y}="B"};
[3] os_md.dviout0(1|opt="TikZ");
TikZ=1
[4] os_md.xyline([1,2,x],[3,4,y]);
\draw node(_0) at(1,2){$x$} node(_1) at(3,4){$y$}(_0)--(_1);
[5] os_md.xyline([1,2,x,"A"],[3,4,y,"B"]);
\draw node(A) at(1,2){$x$} node(B) at(3,4){$y$}(A)--(B);
```

上の [5] において，全体を赤にして，線を点線とするには
```
os_md.xyline([1,2,x],[3,4,y]|opt="dotted,red");
```
とする．また，[5] において，x を緑に，y を黒に，線を赤にするには
```
os_md.xyline([1,2,["green",x]],[3,4,["black",y]]|opt="red");
```
とする．さらに y を丸で囲むには
```
os_md.xyline([1,2,["green",x]],[3,4,["black,draw,circle",y]]|opt="red");
```
とし，さらに y を囲む円内を赤で塗りつぶすには
```
os_md.xyline([1,2,["green",x]],[3,4,["black,draw,circle,fill=red",y]]|opt="red");
```
とする.



345. xyarrow([x_1,y_1,s_1],[x_2,y_2,s_2]|opt=t,cmd=s)
:: Xy-pic/TikZ で (x_1, y_1) と (x_2, y_2) を矢印等で結ぶ（引数はベクトルでもよい）
- 出力する 3 番目の要素 s_1, s_2 は省略可.
- xypos() にあるように，[x_1,y_1,s_1,t_1,u_1] とするとラベル t_1 がつき，$s_1 = [s_{1,1}, s_{1,2}]$ の形もサポートされる.
- 最初の引数が数字のときは "%\n" を，また Xy-pic の場合は最初の引数が 0 でなくて 2 番目の引数が数字のときは空文字列を返す.
- オプションの t は線種などを指定する Xy-pic/TikZ の文字列.

まず Xy-pic の場合のいくつかの例を挙げる．

`"@{->}"`	矢印（デフォルト）	
`"@{<->}"`	両側矢印	
`"@{-}"`	実線	
`"@{.}"`	点線	
`"@{~}"`	波線	
`"@{=}"`	二重線	
`"@{--}"`	破線	
`"@2{.}"`	二重点線	
`"@{{*}->}"`	矢印の例	
`"@{x.o}"`	矢印の例	
`"@2{|.>>}"`	矢印の例	
`"@/^1.5mm/"`	進む方向の左側に 1.5 mm 曲げる	
`"@(ru,ld)"`	右上に出て左下に入る	

TikZ の場合は，オプション文字列 t で指定する．

線の太さ	`very thin`, `thin`, `semithick`, `thick`, `very thick`, `ultra thick`, `width=2pt`,...
点線	`dashed`, `dotted`, `dashe dot`, `dashe dot dot`
点の密度	`loosely dashed`, `densely dashed`, `loosely dotted`, `densly dotted`
矢印	`-`, `->`, `<->`, `<-`, `->>`
矢印の形状	`>=stealth`, `>=latex` など
二重線	`double`
色づけ	`red`, `blue`, `green`, `cyan`, `magenta`, `yellow`, `black`, `gray`, `white`, `darkgray`, `lightgray`, `brown`, `lime`, `olive`, `orange`, `pink`, `purple`, `teal`, `violet`, `green!30!white`（緑 30% 白 70%）, `cyan!10` など
曲線を曲げる	`bend right,distance=0.2cm` など

$t = [t_0, t_1]$ のときは，t_0 が上のように解釈され（$t_0 = 0$ はデフォルト），文字列 t_1 は

`->`	デフォルト
`to[out=60,in=120]`	出る線と入る線の角度指定
`to[out=60,in=120,relative]`	出る線と入る線の相対角度指定
`|-`	縦線，次に横線でつなぐ
`-|`	横線，次に縦線でつなぐ
t_1 の末尾が `"+"`	相対位置指定

- TikZ の場合は，オプション `opt` や `cmd` などを利用して，より広い描画コマンドに使うことができる．
  - `os_md.xyarrow([`x_1`,`y_1`],[`x_2`,`y_2`])` で，x_1, x_2, y_1, y_2 が数字の時

    `\draw[->](`x_1`,`y_1`) -- (`x_2`,`y_2`)`
  - `os_md.xyarrow([`x_1`,`y_1`,`s_1`],[`x_2`,`y_2`])` で，x_1, x_2, y_1, y_2 が数字, s_1 が文字列の時

    `\draw[->]node(_0) at(`x_1`,`y_1`){`s_1`} coordinate(_1) at(`x_2`,`y_2`)(_0) -- (_1);`
  - `os_md.xyarrow([`x_1`,`y_1`,`s_1`],[`x_2`,`y_2`,`s_2`])` で，x_1, x_2, y_1, y_2 が数字, s_1, s_2 が文字列

    `\draw[->]node(_0) at(`x_1`,`y_1`){`s_1`} node(_1) at(`x_2`,`y_2`){`s_2`}(_0) -- (_1);`

    s_1 や s_2 が数式の時は，それを TeX のソースに変換して `$` で鋏んだものに置き換えられる．

– os_md.xyarrow([x_1,y_1,[s_1',s_1],t_1,u_1],[x_2,y_2,[s_2',s_2],t_2,u_2]|opt=[t,t',t''],cmd=s) で，x_1, x_2, y_1, y_2 が数字, 他が文字列

\$s[t]node[s_1'](t_1) at(x_1,y_1){s_1}u_1 node[t_2'](t_2) at(x_2,y_3){s_1}u_2(t_1)t'(t_2)t'';

なお，s のデフォルトは draw, t' のデフォルトは --，t のデフォルトは -> となっている．また t', t'' を指定しないときは，opt=t としてよい．

– x_1 や x_2 が文字列のときは，それはラベルとみなされて y_1 や y_2 は無視され，( ) で囲まれて出力される．

第 2 引数が数字のときは別の解釈が成される．第 2 引数の値を n とすると, n の値に応じてデフォルトのコマンド \draw が次のように置き換えられる（デフォルト以外の指定をしたときは，オプションの cmd= の設定より優先される）.

n の値	コマンド	
0	\draw	\path[draw] と同等
1	\fill	\path[fill] と同等
2	\filldraw	\path[fill,draw] と同等
3	\shade	\path[shade] と同等
4	\shadedraw	\path[shade,draw] と同等
5	\clip	\path[clip] と同等
6	\pettern	\path[pattern] と同等
7	\path	
8	\node	\path node と同等
9	\coordinate	\path coordinate と同等

これで定まるコマンドを \cmd としたとき

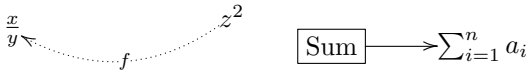
os_md.xyarrow([x,y],n|opt=[t,t',t''])

で x, y, n が数字で他が文字列のとき

\cmd[t](x,y)t'[t'']

となる．なお，t'' の項がない場合は上の [t''] の項は現れない.

```
[0] os_md.dviout0(7)$
TikZ=0
[1] os_md.xyarrow([1,2,x/y],[30,4,z^2,0,"|f"]|opt="@{<.} @/_5mm/");
{(1,2) *{\frac{x}y} \ar@{<.} @/_5mm/ (30,4) *{z^2 }|f};
[2] os_md.xyproc(@@|dviout=1);
[3] os_md.xyarrow([0,0,["+[F]","Sum"]],[20,0,"$\\sum_{i=1}^na_i$"]);
{(0,0) *+[F]\txt{Sum} \ar (20,0) *{\sum_{i=1}^na_i}};
[4] os_md.xyproc(@@|dviout=1);
```



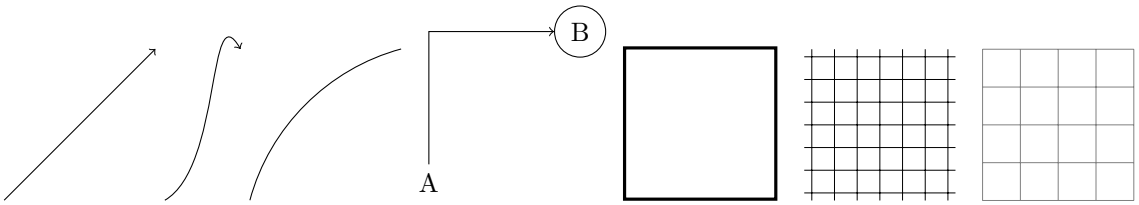

```
[0] os_md.dviout0(6)$
TikZ=1
[1] os_md.xyarrow([0,1],[2,3]);
\draw[->](0,1) -- (2,3);
[2] os_md.xyarrow([0,1],[1,2]|opt=["->,thin","to[out=60,in=120]+"]);
\draw[->,thin](0,1)to[out=30,in=120]+(2,3);
[3] os_md.xyarrow([0,1],[2,3]|opt=["bent left, distance=1cm","to"]);
```

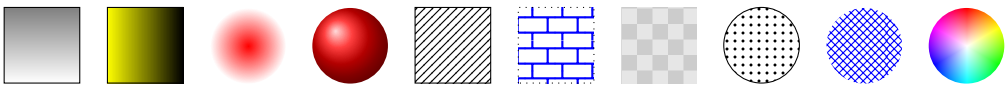
```
\draw[bent left, distance=1cm](0,1)to(2,3);
[4] os_md.xyarrow([0,1,"A"],[2,3,["draw,circle","B"]]|opt=[0,"|-"]);
\draw[->]node(_0) at(0,0){A} node[draw,circle](_1) at(1,1){B}(_0)|-(_1);
[5] os_md.xyarrow([0,1],[2,3]|opt=["very thick","rectangle"]);
\draw[very thick](0,1)rectangle(2,3);
[6] os_md.xyarrow([-1,-1],[1,1]|opt=["very thin,step=0.3cm","grid"]);
\draw[very thin,step=0.3cm](-1,-1)grid(1,1);
[7] os_md.xyarrow([-1,-1],[1,1]|opt=["help lines,step=0.5cm","grid"]);
\draw[help lines,step=0.5cm](-1,-1)grid(1,1);
[8] os_md.xyarrow([0,0],[1,1]|opt=["shade","rectangle"]);
\draw[shade](0,0)rectangle(1,1);
[9] os_md.xyarrow([0,0],[1,1]|opt=["shade,left color=yellow,right color=black",
    "rectangle"]);
\draw[shade,left color=yellow,right color=black](0,0)rectangle(1,1);
[10] os_md.xyarrow([0,0],[".5cm"]|opt=["inner color=red","circle"],cmd="shade");
\shade[inner color=red](0,0)circle(.5cm);
[11] os_md.xyarrow([0,0],[".5cm"]|opt=["ball color=red","circle"],cmd="shade");
\shade[ball color=red](0,0)circle(.5cm);
[12] os_md.xyarrow([0,0],[1,1]|opt=["pattern=north east lines","rectangle"]);
\draw[pattern=north east lines](0,0)rectangle(1,1);
[13] os_md.xyarrow([0,0],[1,1]|opt=["dotted,pattern=bricks,pattern color=blue",
     "rectangle"]);
\draw[dotted,pattern=bricks,pattern color=blue](0,0)rectangle(1,1);
[14] os_md.xyarrow([0,0],[1,1]|opt=["pattern=checkboard light gray","rectangle"],
     cmd="fill");
\fill[pattern=checkboard light gray](0,0)rectangle(1,1);
[15] os_md.xyarrow([0,0],0|opt=["pattern=dots","circle","radius=.5cm"]);
\draw[pattern=dots](0,0)circle[radius=.5cm];
[16] os_md.xyarrow([0,0],1|opt=["pattern=crosshatch,pattern color=blue",
    "circle","radius=.5cm"]);
\fill[pattern=crosshatch,pattern color=blue](0,0)circle[radius=.5cm];
[17] os_md.xyarrow([0,0],3|opt=["shading=color wheel white center","circle",
     "radius=.5cm"]);
\shade[shading=color wheel white center](0,0)circle[radius=.5cm];
```

上の [12] 以降の pattern の指定では, TEX ファイルの初めに

```
\usetikzlibrary{patterns}
```

が必要である. 上の結果を S として xyproc(S|dviout=1) とすると順に以下が得られる.





\shade で指定される塗りつぶしの色変化は

top color=, bottom color=, midddle color=
left color=, right color=, midddle color=
inner color=, outer color=
ball color=
upper left=, upper right=, lower left=, lower right=
shading=color wheel, color with black center, color wheel white center

pattern= で設定できる塗りつぶしパターンは

horizontal lines, vertical lines, north east lines, north west lines, grid, crosshatch, dots, crosshatch dots, fivepointed stars, sixpointed stars, bricks, checkerboard

がある．そのバリエーションで以下のようなものも可能である．

checkerboard light gray, horizontal lines light gray, horizontal lines gray, horizontal lines dark gray, horizontal lines light blue, horizontal lines dark blue, crosshatch dots gray, crosshatch dots light steel blue

346. xyarrows([f_1,f_2],[v_1,v_2],[[x_1,x_2,m],[y_1,y_2,n]]|scale=r,abs=v)
:: XY-pic/TikZ で複数の矢印を描く
x 変数の区間 $[x_1, x_2]$ を m 等分，y 変数の区間 $[y_1, y_2]$ を n 等分し，それぞれの終点を除いた $m \times n$ の各格子点 (x, y) に対して，点 $(f_1(x, y), f_2(x, y))$ を始点として，$(v_1(x, y), v_2(x, y))$ というベクトル（矢印）をXY-pic/TikZ で描く．

- y 変数がないときは，第 3 引数を [x_1,x_2,m] としてよい．
- scale=r：ベクトルを r 倍して描く．
- abs=v：ベクトルの長さを v にスケール変換する．
- opt=s：xyarrow( ) にそのまま渡されるオプション・パラメータ

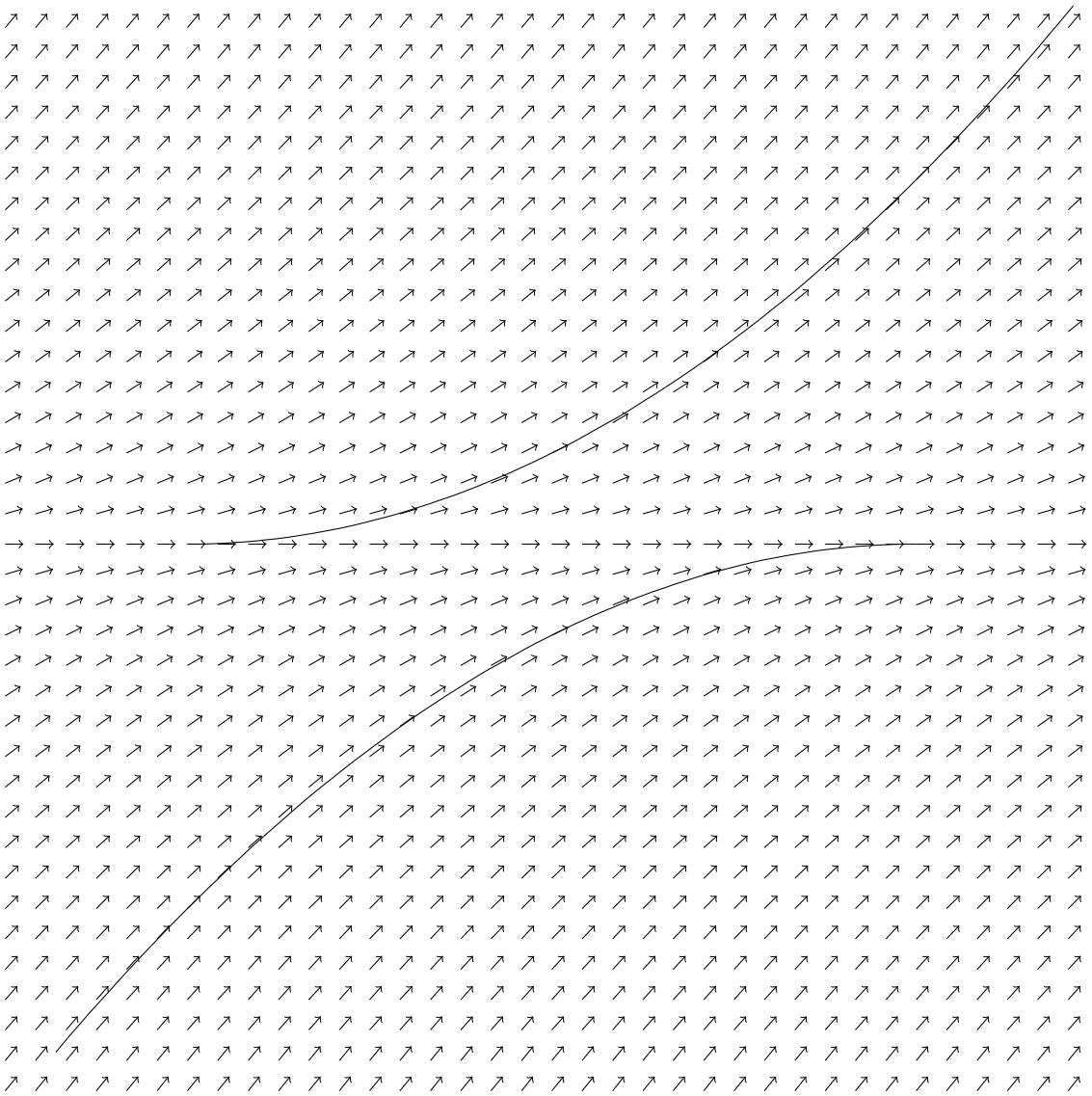
```
[0] os_md.dvi([0,4,6])$
[1] Abs=os_md.abs(y)$Sqrt=os_md.sqrt(x)$
[2] SqrtAbs=os_md.compdf(Sqrt,x,Abs)$
[3] P=[x,y]$Q=[1,SqrtAbs]$
[4] S=os_md.xyarrows(P,Q,[[-1.5,1.5,36],[-1.5,1.5,36]]|abs=0.05)$
[5] S+=os_md.xygraph((x+1)^2/4,-36,[-1,1.5],[-1,1.5],[-1.5,1.5])$
[6] S+=os_md.xygraph(-(x-1)^2/4,-36,[-1.5,1],[-1.5,1],[-1.5,1.5])$
[7] os_md.xyproc(S|opt="scale=5"|dviout=1)$
```

これの実行結果は図 1 となる．

347. xybox([[x_1,y_1],[x_2,y_2],[x_3,y_3]]|opt=t,color=s...)　xybox([[x_1,y_1],[x_2,y_2]]|opt=t,color=s,...)
:: XY-pic/TikZ で (x_1, y_1) と (x_2, y_2) を対角点とし，(x_3, y_3) を頂点とする平行四辺形（あるいは水平な辺をもつ長方形）を描く

- [x_j,y_j] はベクトルでもよい．
- xylines() のオプションと同じオプションが有効．
- TikZ での長方形描画の場合は，color=s のオプションが指定でき，色付けや塗りつぶしなどが可能．このとき，他のオプションは無視される．
  - color=s として s が文字列の時，\draw[s]··· となる．
    color="red"：赤で箱を描く
    color="fill=red"：赤で塗りつぶす．枠は（指定しなければ）黒
  - color=[s,cmd]：とすると \$cmd\draw[s]··· となる．

図 1　微分方程式 $\frac{dy}{dx} = \sqrt{|y|}$



```
[0] os_md.dviout0(0|opt="TikZ")$
TikZ=0
[1] os_md.xybox([[1,2],[30,40]]);
{(1,2) \ar@{-} (1,40)};
{(1,40) \ar@{-} (30,40)};
{(30,40) \ar@{-} (30,2)};
{(30,2) \ar@{-} (1,2)};
[2] os_md.xybox([[1,2],[3,4],[0,0]]);
{(1,2) \ar@{-} (0,0)};
{(0,0) \ar@{-} (3,4)};
{(3,4) \ar@{-} (4,6)};
```

```
{(4,6) \ar@{-} (1,2)};
[3] os_md.dviout0(1|opt="TikZ")$
TikZ=1
[4] os_md.xybox([[1,2],[3,4]]);
\draw(1,2)rectangle(3,4);
[5] os_md.xybox([[1,2],[3,4]]|color="red");
\draw[red](1,2)rectangle(30,40);
[6] os_md.xybox([[1,2],[3,4]]|color="red,fill=yellow");
\draw[red,fill=yellow](1,2)rectangle(30,40);
[7] os_md.xybox([[1,2],[3,4]]|color=["yellow","fill"]);
\fill[red](1,2)rectangle(30,40);
[8] os_md.xybox([[1,2],[3,4],[0,0]]);
\draw (1,2) -- (0,0) -- (3,4) -- (4,6) -- cycle;
```

348. `xycirc([`x`,`y`,`s`],`r`|opt=`t`,arg=[`θ_1`,`θ_2`],deg=[`θ_1`,`θ_2`],close=1)`
:: XY-pic/TikZ で (x,y) 中心の半径 r mm/cm の円を描く
- 出力する 3 番目の s は省略可
- `opt=`t：オプション文字列
- 大きな円が描けない場合は，`arg=[-@pi,@pi]` を用いる．なお `usepackage[pdf,all]{xy}` によって `pdf` ファイルを作成する場合は，このエラーは起きない．
- s があって $r=0$ のときは，s を円で囲む．
- `arg` を指定したときは意味が異なり，偏角 θ_1 から θ_2 までの円弧を描く（$0<\theta_2-\theta_1\le 2\pi$）．このときオプション `opt` は線種指定文字と解釈される（cf. `xylines( |curve=1)`）．また `close=1` で中心と線分で繋いで扇型とする．
- より汎用性のある `xyang()` を用いる方がよい．

```
[0] os_md.xycirc([2,3],5|opt="l^d");
{(2,3) *\cir<5mm>{l^d}};
[1] os_md.xycirc([2,3,x],0);
{(2,3) *+{x} *\cir{}};
```

349. `xybezier([[`x_1`,`y_1`],...,[`x_n`,`y_n`]]|verb=`k`,opt=`t`,cmd=`s`,relative=1)`
:: XY-pic/TikZ で区分 Bézier 曲線（複合 Bézier 曲線）を描く
始点を (x_1,y_1), 終点を (x_n,y_n) とし，途中の座標を制御点とする Bézier 曲線を描く．
- `[`x_j`,`y_j`]` はベクトルでもよい．
- 途中の `[`x_i`,`y_i`]` を数字の 0, ±1 にすると以下のような特別の意味を持つ．
  - `0`：前後で切り離して別の Bézier 曲線とする．
  - `1`：直前の点を途中の通過点とする Bézier 曲線とする．
  - `-1`：始点を終点とする閉じた Bésizer 曲線とし，前後で切り離す．
- `verb=1`：始点と終点を •，制御点を × で表示する．単独の Bézier 曲線となる場合のみ有効．
- `verb=2`：上と同様だが，終点は表示しない．
- `verb=[`k`,`t_1`,`t_2`]`：始点と終点，制御点の描き方を指定する．
  - $k=1,2$ は省略できて，省略したときは $k=1$ （始点と終点を描く）と解釈される．
  - t_1 は始点と終点，t_2 は制御点の示す記号を表し，`xyput()` の引数のリストのの成分 s に対応する．
  - デフォルトは，t_1 =`"$\\bullet$"`, t_2 =`"$\\times$"`

  - t_2 を省略して verb=[t_1] とすると，始点と終点のみ描いて，制御点は描かない．
  - TikZ のときは，たとえば verb=[["red","$\\bullet$"],"$\\times$"] とすると，始点と終点を赤で描く．
- relative=1：TikZ のときに，始点以外を座標の相対指定とする．
- cmd=s：TikZ におけるコマンド \draw の代わりに \s を使う．
- XYLim：TEX のソースで，XYLim 個の点毎に見やすくするため改行を入れる．
  dviout0(n|opt="XYLim") によって XYLim を n に変更できる（デフォルトは 4）．
- opt=t：コマンドの後にオプション t をつける．

TikZ の場合は，xyarrow() の項に書かれた線種指定の他，以下のような塗りつぶしコマンドが使える．

fill	塗りつぶし
fill=yellow,draw=blue	赤で線を描き，内部を黄色で塗りつぶす
fill=red,opacity=.5	赤で透明度 0.5 で塗りつぶす
fill=red,even odd rule	winding number が奇数（デフォルトは 0 以外）の部分を赤で塗りつぶす

```
[0] A=newvect(8)$
[1] for(I=0;I<8;I++) A[I]=[I,I/2];$
[2] os_md.dviout0(7)$
TikZ=0
[3] os_md.xybezier([A[0],A[1],A[2],A[3],A[4]]);
{(0,0);(4,2)
**\crv{(1,0.5)&(2,1)&(3,1.5)}};
[4] os_md.xybezier([A[0],A[1],A[2],A[3],0,A[4],A[5],A[6]]);
{(0,0);(3,1.5)
**\crv{(1,0.5)&(2,1)}};
{(4,2);(6,3)
**\crv{(5,2.5)}};
[5] os_md.xybezier([A[0],A[1],A[2],A[3],1,A[4],A[5],A[6]]);
{(0,0);(3,1.5)
**\crv{(1,0.5)&(2,1)}};
{(3,1.5);(6,3)
**\crv{(4,2)&(5,2.5)}};
[6] os_md.dviout0(6)$
TikZ=1
[7] os_md.xybezier([A[0],A[1],A[2],A[3]]);
\draw (0,0) .. controls (1,0.5) and (2,1) .. (3,1.5);
[8] os_md.xybezier([A[0],A[1],A[2],A[3],0,A[4],A[5],A[6]]);
\draw (0,0) .. controls (1,0.5) and (2,1) .. (3,1.5)
 (4,2) .. controls (5,2.5) .. (6,3);
[9] os_md.xybezier([A[0],A[1],A[2],A[3],1,A[4],A[5],A[6]]);
\draw (0,0) .. controls (1,0.5) and (2,1) .. (3,1.5)
  .. controls (4,2) and (5,2.5) .. (6,3);
[10] os_md.xybezier([A[0],A[1],A[2],A[3],-1,A[4],A[5],A[6],-1]);
\draw (0,0) .. controls (1,0.5) and (2,1) and (3,1.5) .. cycle
 (4,2) .. controls (5,2.5) and (6,3) .. cycle;
[11] os_md.xybezier([A[0],A[1],A[2],A[3],0,A[4],A[5],A[6],-1]|relative=1);
```

```
\draw (0,0) .. controls +(1,0.5) and +(2,1) .. +(3,1.5)
 (4,2) .. controls +(5,2.5) and +(6,3) .. cycle;
[11] os_md.xybezier([[0,0],[2,0],1,[1,1],1,-1]|cmd="fill",opt="green");
\fill[green] (0,0) -- (2,0) -- (1,1) -- cycle;
[12] os_md.xybezier([[0,0],[2,0],1,[1,1],1,-1]|opt="fill,red");
\draw[fill,red](0,0) -- (2,0) -- (1,1) -- cycle;
[13] os_md.xybezier([[0,0],[2,0],1,[1,1],1,-1]|opt="fill=yellow,draw=red");
\draw[fill=yellow,draw=red](0,0) -- (2,0) -- (1,1) -- cycle;
```

350. `draw_bezier(`*id*`,`*idx*`,`*b*`|col=`*c*`,opt=`*s*`,init=1)`
:: キャンバス上に区分 Bézier 曲線を描く
サーバー id が *id* でキャンバス id が *idx* のキャンバスに Bézier 曲線を描く（cf. `draw_obj()`）.
- *b* は `xybezier()` の引数の形式, あるいはそれを `lbezier()` で変換したもの（よって区分 Bézier 曲線でよい）. または, `tobezier()` の戻り値の `t` 変数の Bézier 曲線でもよい.
- `col=`*c*：色指定（`0xffffff` が白）.
- 文字列による指定で, `red`, `blue`, `green`, `yellow`, `cyan`, `magenta`, `black`, `white`, `gray`, `thin`, `very thin`, `dotted`, `dashed` が可能. `opt="red,thin,dotted"` などという指定ができる.
- `init=1`：点線描画の初期化（他の引数はすべて無視）.

```
[0] Id=ox_launch_nox(0,"ox_plot")$
[1] open_canvas(Id)$
[2] Idx=ox_pop_cmo(Id)$
[3] B=os_md.xyoval([150,150],100,1|opt=0)$   /* 中心 (150,150) 半径 100 の円 */
[4] tstart$os_md.draw_bezier(Id,Ind,B|col=0xff00ff)$tstop;
[5] [6] 0.0156sec(0.015sec)
```

351. `xylines([[`x_1`,`y_1`,`s_1`],[`x_2`,`y_2`,`s_2`],...]|opt=`*t*`,close=1,curve=1,ratio=`*c*`,verb=1,scale=`*r*`,Acc=1,dviout=1,proc=`*p*`)`
:: XY-pic/Ti*k*Z で s_j を (x_j, x_j) に置き，(x_1, y_1), (x_2, y_2), ... を（Bézier 曲）線で結ぶ
- `close=1`：最後の点を最初の点と結んで，多角形または滑らかな閉曲線を作る.
- `close=-1`：曲線で結ぶ場合に，最初と最後の点は制御点とし，残りの点を滑らかに結ぶ.
- `curve=1`：曲線（Bézier 曲線を使う）で滑らかに結ぶ.
  P_0, P_1, P_2, P_3 を通る曲線で P_1 と P_2 を結ぶには，$\overrightarrow{P_0P_2}$ の方向に P_1 から $c_1\overline{P_1P_2}$ の距離進んだ点 Q と，$\overrightarrow{P_3P_1}$ の方向に P_2 から $c_2\overline{P_1P_2}$ 進んだ点 R の 2 点を制御点とする 3 次の Bézier 曲線を用いる．P_0 が存在しないときは Q を，P_3 が存在しないときは R を省く．$\overrightarrow{P_0P_2}$ と $\overrightarrow{P_1P_3}$ のなす角を θ $(0 \le \theta \le \pi)$ とするとき，デフォルトでは

$$\frac{c_1+c_2}{2} = \frac{2}{3(1+\cos\frac{\theta}{2})}, \quad c_1 : c_2 = \overline{P_0P_2} : \overline{P_3P_1}$$

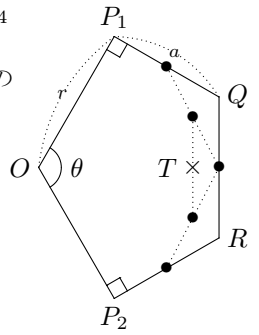
  としている.
  正 n 角形の頂点を指定し，`close=1` を指定すると，デフォルトでは外接円に近い曲線になる. 実際, 外接円の半径を 1 としたとき，得られた曲線の中心からの距離と 1 との差の最大値は，$n=3$ のとき 0.0015, $n=4$ のとき 0.00027, $n=6$ のとき 0.000024, $n=8$ のとき 0.000004 という程度の小さな値となる.
  なお，`ratio=`*c* を指定したときは，以下のように定める.

$$\overrightarrow{P_1Q} = c\overrightarrow{P_0P_2}, \quad \overrightarrow{P_2R} = c\overrightarrow{P_3P_1}$$

- `acc=1`：`curve=1` を指定したとき，`pari()` の高精度で制御点を計算する.

- `curve=2`：B-spline 曲線で結ぶ（途中の点は制御点で通らない）．
  このときは s_j は最初と最後のみ指定可能．Ti*k*Z ではサポートされない（3 次以下の Bézier 曲線はサポート）．
- $s_1, s_2, \ldots$ は省略可能．また s_j がないときは，`[`x_j`,`y_j`]` はリストでなくてベクトルでもよい．
- 通過点のデータに 0 があるときは，その前後で直線または曲線を切る．
- `scale=`r：Xy-pic/Ti*k*Z の座標に直すときに r 倍する．
- `scale=[`r_1`,`r_2`]`：Xy-pic/Ti*k*Z の座標に直すときに x 座標を r_1 倍，y 座標を r_2 倍する．
  Xy-pic は数字が mm 単位，Ti*k*Z は cm 単位である．
- `opt=`t：文字列 t による線種の指定．Xy-pic でオプション `curve` を指定せずに線分で結ぶときの線種指定は `xyarrow()` を参照．`curve`= 1, 2 のときは，たとえば Xy-pic ならば `"~*=<3pt>{.}"` は 3pt 間隔の点線．
  Ti*k*Z のときの線種指定は，`xyarrow()` の項を参照（t がオプション `opt` の値として渡される）．また $t = [t_0, t_1]$ と 2 つの文字列のリストのときは，t_0 と t_1 とが `xybezier()` の `opt` と `cmd` のパラメータに設定される．
- `opt=0`：`xybezier()` に渡す引数のリストを返す．
- `dviout=1`：画面で表示する．
- `proc=1`：描画実行形式の描画成分を返す（`proc=2` も同様）．
- `verb=1`：通過点を •，制御点を × で表示する．
- `verb=[`t_1`,`t_2`]`：通過点を t_1，制御点を t_2 で表す（cf. `xybezier()`）．
  デフォルトは，t_1 =`"\\bullet"`, t_2 =`"\\times"` である．
- `verb=[`t_1`]`：通過点を t_1 で表す．

円の中心 O があって，$OP_0 = OP_1 = OP_2 = OP_3$ かつ $P_0P_1 = P_1P_2 = P_3P_4$ となっている場合を考える．$\angle P_1OP_2 = \theta$, $OP_0 = r$, $P_1Q = P_2R = a$ とおいて座標を適当に選ぶと，Bézier 曲線上の $t = \frac{1}{2}$ に対応する点 T は以下のようになる．



$$
\begin{aligned}
&O : (0,0),\ P_1 : (r\cos\tfrac{\theta}{2}, r\sin\tfrac{\theta}{2}),\ P_1 : (r\cos\tfrac{\theta}{2}, -r\sin\tfrac{\theta}{2})\\
&Q : (r\cos\tfrac{\theta}{2} + a\cos\tfrac{\theta-\pi}{2}, r\sin\tfrac{\theta}{2} + a\sin\tfrac{\theta-\pi}{2})\\
&\quad = (r\cos\tfrac{\theta}{2} + a\sin\tfrac{\theta}{2}, r\sin\tfrac{\theta}{2} - a\cos\tfrac{\theta}{2})\\
&R : (r\cos\tfrac{\theta}{2} + a\sin\tfrac{\theta}{2}, -r\sin\tfrac{\theta}{2} + a\cos\tfrac{\theta}{2})\\
&T : (r\cos\tfrac{\theta}{2} + \tfrac{3}{4}a\sin\tfrac{\theta}{2}, 0)
\end{aligned}
$$

条件 $OT = OP_1$ が成り立つとすると

$$r\cos\tfrac{\theta}{2} + \tfrac{3}{4}a\sin\tfrac{\theta}{2} = r$$

より

$$a = \frac{4}{3}\frac{1-\cos\frac{\theta}{2}}{\sin\frac{\theta}{2}}r = \tfrac{4}{3}\tan\tfrac{\theta}{4}r = \frac{4}{3}\frac{\sin\frac{\theta}{2}}{1+\cos\frac{\theta}{2}}r.$$

このとき

$$
\begin{aligned}
Q &: \Big(r\cos\tfrac{\theta}{2} + \frac{4}{3}\frac{1-\cos\frac{\theta}{2}}{\sin\frac{\theta}{2}}\sin\tfrac{\theta}{2}r, r\sin\tfrac{\theta}{2} - \frac{4}{3}\frac{\sin\frac{\theta}{2}}{1+\cos\frac{\theta}{2}}\cos\tfrac{\theta}{2}r\Big)\\
&= \Big((\tfrac{4}{3} - \tfrac{1}{3}\cos\tfrac{\theta}{2})r, (1 - \tfrac{1}{3}\cos\tfrac{\theta}{2})\frac{\sin\frac{\theta}{2}}{1+\cos\frac{\theta}{2}}r\Big),\\
\frac{P_1Q}{P_1P_2} &= \frac{4}{3}\frac{\sin\frac{\theta}{2}}{1+\cos\frac{\theta}{2}}\frac{1}{2\sin\frac{\theta}{2}} = \frac{2}{3(1+\cos\frac{\theta}{2})}
\end{aligned}
$$

となる．

$r=1$ として，Bézier 曲線を $B(t)=\big(x(t),y(t)\big)=P_1(1-t)^3+3Qt(1-t)^2+3Rt^2(1-t)+P_2t^3$，$c=\cos\frac{\theta}{2}$ とおくと

$$\begin{aligned}L(s)&:=x(s+\tfrac12)^2+y(s+\tfrac12)^2\\&=\frac{16(1-c)^3}{1+c}s^6-\frac{8(1-c)^3}{1+c}s^4+\frac{(1-c)^3}{1+c}s^2+1\\&=\tfrac{(1-c)^3}{1+c}s^2(4s^2-1)^2+1\end{aligned}$$

となるので，$0\le s\le\frac12$ の範囲で

$$\sqrt[3]{8s^2(1-4s^2)^2}\ge\frac{8s^2+(1-4s^2)+(1-4s^2)}{3}=\frac23$$

となり，等号成立は $8s^2=1-4s^2$，すなわち $s^2=\frac{1}{12}$ のときである．よって $|s|\le\frac12$ の範囲で $L(s)$ は $s=0,\pm\frac12$ で最小値 1 をとり，$s=\pm\frac{1}{2\sqrt3}$ で最大値をとる．

$$L(\pm\tfrac{1}{2\sqrt3})-1=\frac{1}{27}\frac{(1-c)^3}{1+c},$$

$$\sqrt{L(\pm\tfrac{1}{2\sqrt3})}-1\fallingdotseq\frac{1}{54}\frac{(1-\cos\frac{\theta}{2})^3}{1+\cos\frac{\theta}{2}}\fallingdotseq\begin{cases}\frac{1}{648} & (\theta=\frac{2\pi}{3}=120^\circ)\\ \frac{1}{3668} & (\theta=\frac{\pi}{2}=90^\circ)\\ \frac{1}{41900} & (\theta=\frac{\pi}{3}=60^\circ)\\ \frac{1}{235541} & (\theta=\frac{\pi}{4}=45^\circ)\\ \frac{1}{2683400} & (\theta=\frac{\pi}{6}=30^\circ)\end{cases}$$

デフォルトでは P_0,P_1,P_2,P_3 を通る滑らかな曲線の P_1 と P_2 を結ぶ部分は

$$\cos\theta=\frac{(\overrightarrow{P_0P_2},\overrightarrow{P_3P_1})}{\overline{P_0P_2}\cdot\overline{P_1P_3}},$$

$$\cos\frac{\theta}{2}=\sqrt{\frac{1+\cos\theta}{2}},$$

$$c_1\frac{\overline{P_1P_3}}{\overline{P_0P_2}}\cdot\overrightarrow{P_0P_2}=\frac{2}{3+3\sqrt{\frac{1+\frac{(\overrightarrow{P_0P_2},\overrightarrow{P_1P_3})}{\overline{P_0P_2}\cdot\overline{P_1P_3}}}{2}}}\frac{2\overline{P_1P_2}}{\overline{P_0P_2}+\overline{P_1P_3}}\cdot\overrightarrow{P_0P_2}$$

より

$$c:=\frac{4\overline{P_1P_2}}{3(\overline{P_0P_2}+\overline{P_1P_3})}\frac{1}{1+\sqrt{\frac{1+\frac{(\overrightarrow{P_0P_2},\overrightarrow{P_1P_3})}{\overline{P_0P_2}\cdot\overline{P_1P_3}}}{2}}}$$

とおくと

$$\overrightarrow{P_1Q}=c\overrightarrow{P_0P_2},\ \overrightarrow{P_2R}=c\overrightarrow{P_3P_1}$$

によって，P_1 を始点，P_2 を終点，Q と R の 2 点をこの順に制御点とする 3 次 Bézier 曲線となる．

Q_0,Q_1,Q_2,Q_3 と順に与えたとき，Q_0 を始点，Q_1 と Q_2 を制御点，Q_3 を終点とする（3 次）Bézier 曲線は

$$\begin{aligned}B(t)&=Q_0(1-t)^3+3Q_1t(1-t)^2+3Q_2t^2(1-t)+Q_3t^3\\&=(-Q_0+3Q_1-3Q_2+Q_3)t^3+(3Q_0-6Q_1+3Q_2)t^2+(-3Q_0+3Q_1)t+Q_0\end{aligned}$$

で与えられる．

なお，P_0, P_1, P_2, P_3 と順に与えたとき，P_1 を始点，P_2 を終点とする Catmull-Rom スプライン曲線は

$$C(t) = (-\tfrac{1}{2}P_0 + \tfrac{3}{2}P_1 - \tfrac{3}{2}P_2 + \tfrac{1}{2}P_3)t^3 + (P_0 - \tfrac{5}{2}P_1 + 2P_2 + \tfrac{1}{2}P_3)t^2 + (-\tfrac{1}{2}P_0 + \tfrac{1}{2}P_2)t + P_1$$

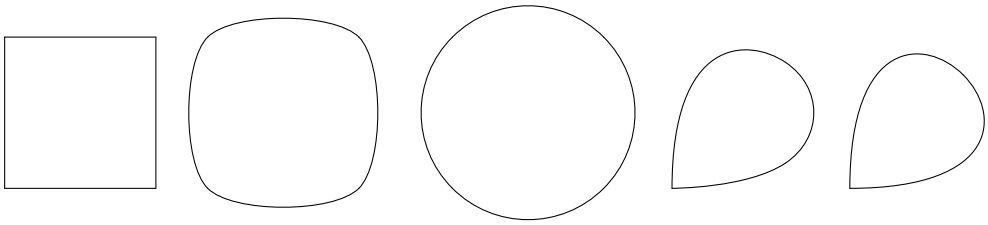
である．両者の関係は

$$\begin{cases} Q_0 = P_1, \\ Q_1 = P_1 + \frac{1}{6}(P_2 - P_0), \\ Q_2 = P_2 + \frac{1}{6}(P_3 - P_1), \\ Q_3 = P_2. \end{cases}$$

よって $c = \frac{1}{6}$ と固定したものとみなせる．

```
[0] L=[[0,0],[20,0],[20,20],[0,20]]$
[1] L0=os_md.xylines(L|close=1);
{(0,0) \ar@{-} (20,0)};
{(20,0) \ar@{-} (20,20)};
{(20,20) \ar@{-} (0,20)};
{(0,20) \ar@{-} (0,0)};
[2] L1=os_md.xylines(L|close=1,curve=1,ratio=1/6)$
[3] L2=os_md.xylines(L|close=1,curve=1)$
[4] L3=os_md.xylines(L|close=1,curve=2)$
[5] L4=os_md.xybezier(append(L,[[0,0]]))$
```

として，L0, L1, L2, L3, L4 を順に表示すると



これらは Xy-pic の例であるが，Ti*k*Z の場合は座標のスケールの違いから，scale=0.1 を指定するとサイズが同じになる．たとえば，上の [3] の L2 は

```
{(0,0);(20,0)
**\crv{(5.523,-5.523)&(14.477,-5.523)}};
{(20,0);(20,20)
**\crv{(25.523,5.523)&(25.523,14.477)}};
{(20,20);(0,20)
**\crv{(14.477,25.523)&(5.523,25.523)}};
{(0,20);(0,0)
**\crv{(-5.523,14.477)&(-5.523,5.523)}};
```

であるが，os_md.dviout(6) などとして Ti*k*Z 出力になるようにして

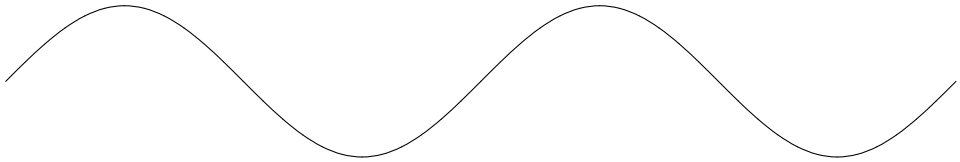
```
os_md.xylines(L|close=1,curve=1,scale=0.1);
```

を実行したときは

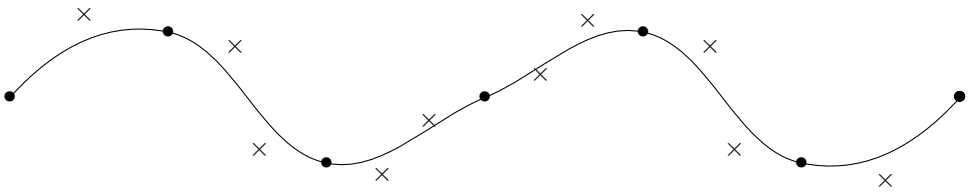
```
\draw (0,0) .. controls (0.552,-0.552) and (1.448,-0.552) .. (2,0) ..
 controls (2.552,0.552) and (2.552,1.448) .. (2,2) .. controls (1.448,2.552)
 and (0.552,2.552) .. (0,2) .. controls (-0.552,1.448) and (-0.552,0.552)
 .. cycle;
```

となる．これは以下でも同様である．

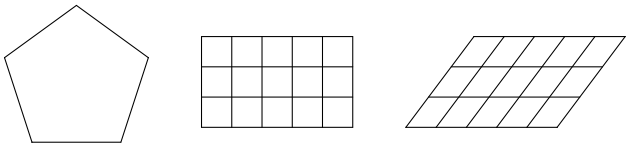
```
[6] Pi=3.14159265$
[7] for(V=[],I=0;I<=48;I++) V=cons([10*Pi*I/12,dsin(Pi*I/12)*10],V);
[8] os_md.xyproc(os_md.xylines(V|curve=1)|dviout=1)$
```



```
[9] for(V=[],I=0;I<=6;I++) V=cons([10*Pi*I*2/3,dsin(Pi*2*I/3)*10],V);
[10] os_md.xylines(V|curve=1,dviout=1,verb=1);
```



```
[11] os_md.xylines(os_md.ptpolygon(5,10)|close=1,dviout=1)$
[12] os_md.xylines(os_md.ptlattice(6,4,[4,0],[0,4]|line=1)|dviout=1)$
[13] os_md.xylines(os_md.ptlattice(6,4,[4,0],[3,4]|line=1)|dviout=1)$
```



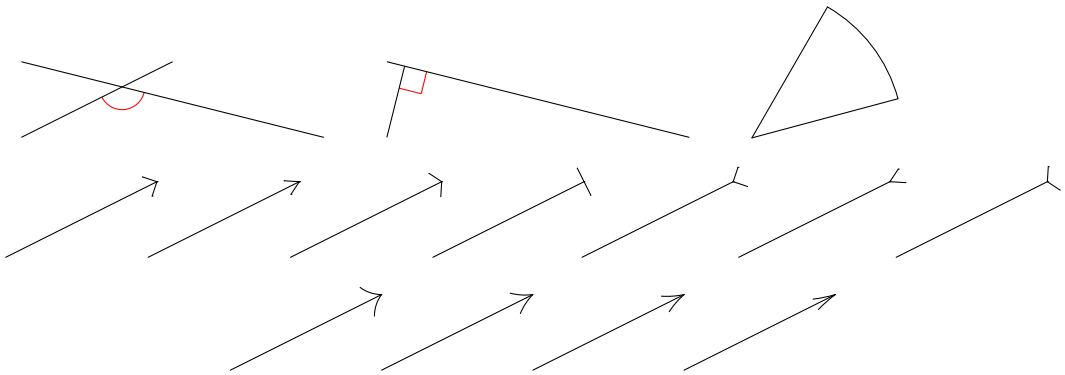
352. `xyang(`r`,`p_0`,`p_1`,`p_2`|opt=`t`,scale=`r`,prec=1,ar=1,dviout=1,proc=1)`
:: XY-pic/Ti*k*Z で角 $\angle p_1p_0p_2$ の記号や円弧や扇形や矢印を描く

- $p_0 = [x_0, y_0]$, $p_1 = [x_1, y_1]$, $p_2 = [x_2, y_2]$ のときは，$\angle p_1p_0p_2$ の角の記号を半径 r の弧で描く（$\overrightarrow{p_0p_1}$ から反時計回り）．
- $p_0 = [x_0, y_0]$, $p_1 = [x_1, y_1]$, $p_2 = \pm 1$ のときは，p_0 を頂点する $\overrightarrow{p_0p_1}$ 方向とそれと直交する方向に一辺 r の直角記号を描く（$p_2 = 1$ のときは反時計回り，$p_2 = -1$ のときは時計回り）．
- $p_0 = [x_0, y_0]$, $p_1 = [x_1, y_1]$, $p_2 = \pm 2, \pm 3, \pm 4, 0$ のときは，p_0 を頂点とする矢印の先を描く．r は先端からの長さ．線分との角度は，順に 45°, 30°, 60°, 90° である．ただし，$p_2 < 0$ の時は矢印の先でなくて尾を示す．
  $p_2 = 5, 6, 7, 8$ のときは，矢先が曲がっていて，開きは順に 45°, 30°, 22.5°, 15° となる．
  `ar=1` を指定すると，p_0 と p_1 を結ぶ線分も描いて，矢印となる．
- $p_0 = [x_0, y_0]$ で，p_1 と p_2 が数で $p_1 \le p_2 \le p_1 + 2\pi$ の時，p_0 を中心に半径 r の円弧を，偏角が p_1 から p_2 まで Bézier 曲線を使って描く．ただし，$p_1 = p_2$ のときは円とする．
  円弧の場合，`ar=1` を指定すると扇形を描く．
- `prec=1`：真の円弧により近くする（デフォルトでの円の中心からの距離の誤差は，0.16% 以下であるが，それを $\frac{1}{3600} \fallingdotseq 0.03\%$ 以下とする）．
  `prec=2`, 3 に応じて，誤差は $\frac{1}{41900}$，$\frac{1}{235500}$ 以下になる．

- proc=1：描画実行形式の要素を返す.
- 指定したオプションパラメータは，上のものを除いて全て xylines() に渡される.

TikZ のときは

```
[0] os_md.xyang(0.3,[1,1],[6,1],[5,5]);  /* 角度の弧 */
\draw (1.3,1) .. controls (1.3,1.08) and (1.268,1.156) .. (1.212,1.212);
[1] os_md.xyang(0.3,[1,1],[6,2],1);      /* 直角記号（反時計回り）*/
\draw (1.294,1.059) -- (1.235,1.353) -- (0.941,1.294);
[2] os_md.xyang(0.3,[1,1],[6,2],-1);     /* 直角記号（時計回り）*/
\draw (1.294,1.059) -- (1.235,1.353) -- (0.941,1.294);
[3] os_md.xyang(2,[1,1],0,@pi/2|opt="dotted,red");  /* 円弧 */
\draw[dotted,red] (3,1) .. controls (3,2.105) and (2.105,3) .. (1,3);
[4] os_md.xyang(2,[1,1],@pi/12,@pi/3|ar=1,scale=2); /* 扇形 */
\draw (5.864,3.035) .. controls (5.589,4.06) and (4.919,4.934) .. (4,5.464)
-- (2,2) -- cycle;
[5] os_md.xyang(2,[1,1],0,0);            /* 円 ：誤差 1/640 以下 */
\draw (3,1) .. controls (3,2.54) and (1.333,3.502) .. (0,2.732)
.. controls (-1.333,1.962) and (-1.333,0.038) .. (0,-0.732) ..
controls (1.333,-1.502) and (3,-0.54) .. cycle;
[6] os_md.xyang(2,[1,1],0,0|prec=1);     /* 円 ：誤差 1/3600 以下 */
\draw (3,1) .. controls (3,2.105) and (2.105,3) .. (1,3)
.. controls (-0.105,3) and (-1,2.105) .. (-1,1) .. controls (-1,-0.105)
and (-0.105,-1) .. (1,-1) .. controls (2.105,-1) and (3,-0.105) .. cycle;
[7] P=[0,0]$Q=[2,1]$R=[0,1]$S=[4,0]$    /* 4 点 P, Q, R, S を定義 */
[8][9][10][11]
[12] T=os_md.ptcommon([P,Q],[R,S]);     /* T : PQ と RS の交点 */
[4/3,2/3]
[13] SS=os_md.xyang(0.3,T,P,S|opt="red")+os_md.xyline(P,Q)+os_md.xyline(R,S)$
[14] os_md.xyproc(SS|dviout=1)$         /* PQ, RS, 角 PTQ を描く */
[15] U=os_md.ptcommon([R,S],[P,0])$     /* U : P から RS に下ろした垂線の足 */
[16] SS=os_md.xang(0.3,T,P,1)+os_md.xyline(P,T)+os_md.xyline(R,S)$
[17] os_md.xyproc(SS|dviout=1)$         /* P から RS に下ろした垂線を描く  */
[18] os_md.xyproc(os_md.xyang(0.2,Q,P,2|ar=1)|dviout=1)$         /* 矢印 */
[19] os_md.xyproc(os_md.xyang(0.2,Q,P,3|ar=1)|dviout=1)$
[20] os_md.xyproc(os_md.xyang(0.2,Q,P,4|ar=1)|dviout=1)$
[21] os_md.xyproc(os_md.xyang(0.2,Q,P,0|ar=1)|dviout=1)$
[22] os_md.xyproc(os_md.xyang(0.2,Q,P,-2|ar=1)|dviout=1)$
[23] os_md.xyproc(os_md.xyang(0.2,Q,P,-3|ar=1)|dviout=1)$
[24] os_md.xyproc(os_md.xyang(0.2,Q,P,-4|ar=1)|dviout=1)$
[25] os_md.xyproc(os_md.xyang(0.3,Q,P,5|ar=1)|dviout=1)$ /* 曲がった矢先 */
[26] os_md.xyproc(os_md.xyang(0.3,Q,P,6|ar=1)|dviout=1)$
[27] os_md.xyproc(os_md.xyang(0.3,Q,P,7|ar=1)|dviout=1)$
[28] os_md.xyproc(os_md.xyang(0.3,Q,P,8|ar=1)|dviout=1)$
```



353. xyoval(*p*,*r*,*q*|opt=*t*,arg=[t_1,t_2,t_3],deg=[t_1,t_2,t_3],scale=*r*,ar=1,prec=1,dviout=1,proc=1)
:: XY-pic/Ti*k*Z で p を中心として，軸の長さが r と qr の楕円またはその弧や扇形を描く
- 点 p を中心に，基準軸の半径が r, それに直交する軸の半径が qr の楕円を描く
- 楕円の基準軸の偏角は t_3 で指定する．この指定がないときは $t_3 = 0$ とみなされる．
- 楕円の基準軸方向から，パラメータが t_1 と t_2 の間の部分の弧を描く．
  この場合，ar=1 を指定すると扇形が描かれる．
  $t_1 = t_2$ または指定がないときは，楕円全体が描かれる．

$$p + r\begin{pmatrix}\cos t_3 & \sin t_3\\ -\sin t_3 & \cos t_3\end{pmatrix}\begin{pmatrix}\cos t\\ q\sin t\end{pmatrix}\qquad (t\in[t_1,t_2])$$

- t_1, t_2, t_3 の指定はラジアン (Ang) または角度（deg）.
- opt=*t*：xylines() におけるオプションと同じ意味.
- proc=1：描画実行形式の要素を返す.

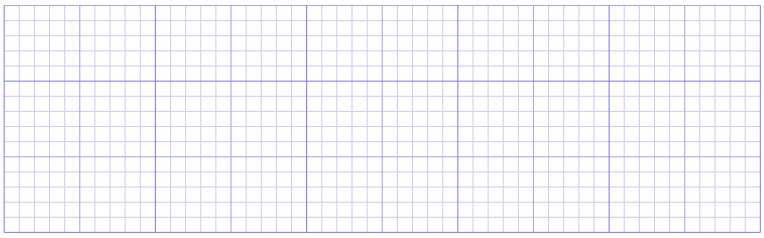
```
[0] os_md.xyoval([0,0],1,1/2|arg=[2*@pi/3,7*@pi/3,@pi/8],scale=10,opt=0);
[[-6.27647,2.0871],[-10.6562,-1.40948],[-10.451,-5.23007],[-5.84741,-5.90082],
1,[-1.2438,-6.57157],[5.5263,-3.7673],[8.30725,-0.0377664],1,[11.0882,3.69177],
[8.53175,6.53844],[2.96233,5.91393]]
[1] os_md.xyoval([0,0],1,1/2|arg=[2*@pi/3,7*@pi/3,@pi/8],scale=10);
{(-6.276,2.087);(-5.847,-5.901)
**\crv{(-10.656,-1.409)&(-10.451,-5.23)}};
{(-5.847,-5.901);(8.307,-0.038)
**\crv{(-1.244,-6.572)&(5.526,-3.767)}};
{(8.307,-0.038);(2.962,5.914)
**\crv{(11.088,3.692)&(8.532,6.538)}};
[3] os_md.dviout0([0,6])$
[4] os_md.xyoval([0,0],1,1/2|dviout=1)$
[5] os_md.xyoval([0,0],1,1/2|arg=[2*@pi/3,7*@pi/3,@pi/8],dviout=1)$
[6] os_md.xyoval([0,0],1,1/2|arg=[2*@pi/3,7*@pi/3,@pi/8],ar=1,dviout=1)$
[7] os_md.xyoval([0,0],1,1/2|opt="red",dviout=1)$
```

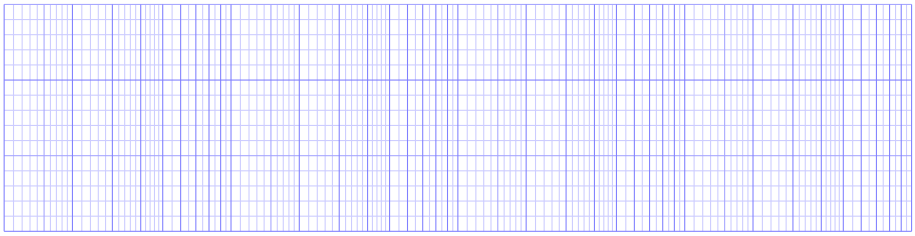
354. xygrid ([*x*,t_x,u_x,m_x,s_x],[*y*,t_y,u_y,m_y,s_y]|raw=*r*,shift=[s_x,s_y])
:: （対数）方眼紙の描画

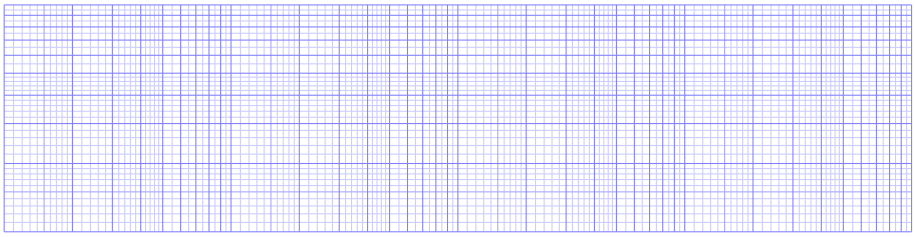
- 横 x，縦 y の方眼紙を作成する（xyproc() の引数の形で返す）．t_x, t_y は，0 のときは通常の目盛，1 のときは対数目盛，-1 のときは逆対数目盛とし，u_x, u_y は，単位の長さ，m_x, m_y は最小の目盛線の間隔とする．
  色付けなどを考慮すると TikZ を利用するのが好ましいので，以下ではそれを念頭に例を述べる（長さの単位は cm となる．dviout0(1|opt="TikZ") によってXy-pic でなくて，TikZ 対応となる）．
- t_x, t_y は直接 0 からの数字のリストとして指定してもよい（m_x, m_y は無視される）．scale() の返り値の形でもよい．
- 目盛線は，横，縦共に 3 段階で，線種はそれぞれ s_x, s_y で文字列指定する．3 段階で変えるときは，文字列 3 つのリストで，小さい目盛線，中間の目盛線，大きい目盛線の順に指定する．3 段階以外も可能．
- 通常の目盛のときは，単位の長さを，細分なし，2 等分，または 5 等分，または 10 等分まで，最小間隔以上という条件で細かく取る．10 等分のときは，2 等分を 2 段階目の線種とする．あるいは，m_x, m_y を scale() の引数の形（リスト）で指定してもよい．
- 対数目盛のときは，1 から 2 まで，2 から 5 まで，5 から 10 までに分けて（1 から 10 までの間隔が単位の長さ），それぞれ細分なし，2 等分，5 等分，10 等分のいずれかで，最小間隔以上という条件で，細かく取る．10 等分のときは，2 等分を中間の線種とする．あるいは，m_x, m_y を scale() の引数の形（リスト）で指定してもよい．
- shift=[s_x,s_y] によって描画位置の原点を移動できる（左下隅が原点）．
- たとえば，五線譜の用紙を描くには y =22.4, t_y =[0,0.2,0.4,0.6,0.8], u_y =2.5, t_x =[] などとすればよい．
- raw=1：縦線の位置（3 段階），横線の位置（3 段階）の組のリストを返す．
- raw=2：上を，xylines() の引数のリストの組として返す．

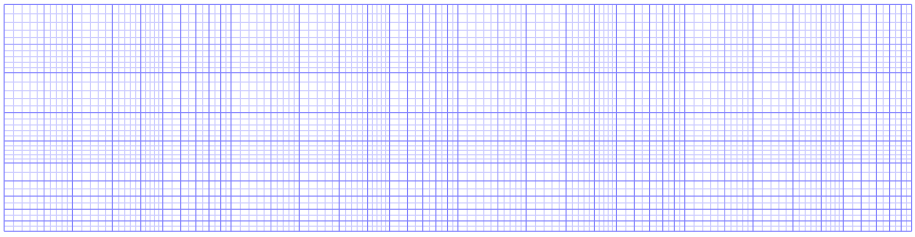
```
[0] L=["blue!20","blue!35","blue!50"]$
   /* [最小目盛の線種，中間目盛の線種，通常目盛の線種] */
[1] X=[10,0,2,0.15,L]]$ Y=[3,0,2,0.15,L]]$
   /* [方眼紙のサイズ（横/縦 cm），0:等間隔か 1:対数目盛か，一つのユニットの大きさ（cm），
      細分したときの最小幅（cm），線種] */
[2] S0=xyproc(os_md.xygrid(X,Y))$   /* 2mm 方眼紙 */
[3] U=[12,1,3,0.03,L]$ V=[3,1,3,0.03,L]$ W=[3,-1,3,0.03,L]$
[4] S1=os_md.xyproc(os_md.xygrid(U,Y))$   /* 片対数方眼紙 */
[5] S2=os_md.xyproc(os_md.xygrid(U,V))$   /* 両対数方眼紙 */
[6] S3=os_md.xyproc(os_md.xygrid(U,W))$   /* 左上原点両対数方眼紙 */
```

S0, S1, S2, S3 は方眼紙の TEX のソースとなる．









355. xygraph(*f*,*n*,[t_1,t_2],[x_1,x_2],[y_1,y_2]|opt=*t*,rev=1,ax=[x_0,y_0,*s*,*t*,*u*],axopt=[*h*,*w*,*o*,*z*],scale=*r*,ratio=*c*,raw=1,org=[x_0,y_0],pt=[p_1,p_2,...],verb=1,para=1,prec=*v*,shift=[*u*,*v*],Acc=1,dviout=*t*,proc=*p*)

:: 変域の n 等分点での値で関数のグラフを描く（$t_1 \le$ x $\le t_2$，(x_1,y_1)–(x_2,y_2) は表示窓の範囲）

- 点を繋ぐのは xylines() を curve=1 のオプションで用いる．
- 変数は x で，さらに $x_1 < x_2$，$y_1 < y_2$ でなければならない．
- f は [f_1,f_2] の形のパラメータ表示でもよい．
- para=1：f が [f_1,f_2] のパラメータ表示であることを明示する．
- f や前項の f_1, f_2 は有理函数や $\sin(x)$ などの初等関数に限らず，mydeval() が解釈できる函数ならばよい．よってユーザが定義した函数や，Γ 函数などのように pari() 経由でサポートされた函数でもよい．
- f_1 が有理式で f_2 がリスト形式の函数のときは，para=1 としてパラメータ表示であることを明示する必要がある．
- 変数を x でなくて t とするときは，[t_1,t_2] でなくて [t,t_1,t_2] と指定する．
- $x_1 = t_1$, $x_2 = t_2$ のときは，第 4 引数 [x_1,x_2] を 0 と指定してもよい．
- rev=1：$x = f(y)$ のグラフとみなす．
- (x_1,y_1) と (x_2,y_2) で定まる窓から外れる点は除く．
- $n = 0$ と指定したときは，$n = 32$ とみなす．
- $n < 0$ のときは，$|n|$ 等分した区間の両側の 1 区間外に制御点をとる．
- n 等分点でなくて点を直接指定するときは，[t_1,t_2] でなくて [t_1,t_2,...,t_m] のように指定する（なお，$t_1,\ldots,t_m$ は単調増加な実数列．さらに最初を実数でなく，変数の指定としてもよい）．n が負の時は，最初と最後の点 t_1, t_m は制御点とする．
- prec=[v_1,v_2,v_3]：折れ線で繋ぐと角度が v_2° 以上の分割点の両側を 2 つに細分すること（さらに $v_3 > 0$ のときは窓の対角線の長さの v_3 分の 1 以上の跳びがある区間の細分）を v_1 ステップ行い，さらに $v_3 > 0$ のとき，窓の対角線の長さの v_3 分の 1 以上の跳びを不連続点とみなす．
  - $v_2 = 1$ のときは $v_2 = 30$ と解釈され，$1 < v_2 < 10$ のときは $v_2 = 10$ と解釈され，$v_2 > 120$ のときは $v_2 = 120$ と解釈される．

  - $v_3 = 0$ のときは，不連続点はないものと解釈する．
  - $v_3 = 1$ のときは $v_3 = 16$ と解釈され，$v_3 > 512$ のときは $v_3 = 512$ と解釈される．
  - `prec=[`v_1`,`v_2`]` は，`prec=[`v_1`,`v_2`,0]` と解釈される．
  - `prec=`v_1 は，$v_1 = 0$ のとき `prec=[4,30,0]` と，$v_1 > 0$ のとき `prec=[`v_1`,30,0]` と，$v_1 < 0$ のとき `prec=[`$|v_1|$`,30,16]` と解釈される．
- `opt=`t：線種の指定など（`xylines()` に渡される）．
- `opt=0`: 曲線を `xybezier()` の引数のデータ形式で返す．
- `ratio=`c：Bézier 曲線を使うときのパラメータ（`xylines()` に渡される）．
- `ax=[`x_0`,`y_0`]`：(x_0, y_0) を原点として x 軸と y 軸を窓内に描く．
  窓内に現れない軸は描かないので，それを利用して一方の軸のみの描画ができる．
- `ax=[`x_0`,`y_0`,`s`,`t`]`：s が正数なら，x 軸に原点から s 毎に目盛をつける．
  目盛をつける位置を指定するときは $s =$ `[`$s_1, s_2, \ldots$`]` と原点からの x 座標の位置を書く．
  $s_j =$`[`$s_{j,0}$`,`$s_{j,1}$`]` となっていると $s_{j,0}$ の位置に目盛をつけて，そこに $s_{j,1}$ を書く．
  t で同様に y 軸の目盛の指定ができる
- `ax=[`x_0`,`y_0`,`s`,`t`,`u`]`：s, t が数字で u が 1 と 2 とすると，目盛が `ax=[`x_0`,`y_0`,`s`,`t`]` に従ってつけられ，u に従ってそこに数字が書かれる．$u = 0$ のとき k 番目の位置の x 軸には $ks + x_0$ が y 軸には $kt + y_0$ が書かれ，$u = 1$ のとき k 番目の位置の x 軸には ks が y 軸には kt が書かれ，$u = 2$ のとき k 番目の位置に k が書かれる．
- `axopt=z`：z が文字列の時は x 軸と y 軸の線種指定文字列（デフォルトは実線で Xy-pic のときは `"@-"`）．
- `axopt=h`：h が 0 以外の数字の時は，目盛の軸からの長さを与える．
  値が負の時は軸から負方向（x 軸なら下，y 軸なら左）の目盛．
- `axopt=[`h`,`w`,`o`,`z`]`：軸や目盛の描き方の指定
  - $h =$`[`h_0`,`h_1`]` または $h =$`[`h_0`,`h_1`,`h_2`]` のとき x 軸の目盛の高さを h_0 から h_1 までとする（Xy-pic/TikZ の mm/cm 単位）．
    h_2 は文字列で，目盛位置に文字などを置く位置指定（デフォルトは下 `"+!U"`/`"below"`）．
    h が 0 でない数字なら，$h_0 = 0$, $h_1 = 1$ と解釈される（$h = 0$ のときはデフォルトのまま）．
  - $w =$`[`w_0`,`w_1`]` または $w =$`[`w_0`,`w_1`,`w_2`]` のとき y 軸の目盛の幅を w_0 から w_1 までとする（Xy-pic/TikZ の mm/cm 単位）．
    w_3 は文字列で，目盛位置に文字などを置く位置指定（デフォルトは左 `"+!R"`/`"left"`）．
    w が数字なら，$w_0 = 0$, $w_1 = 1$ と解釈される（$w = 0$ のときはデフォルトのまま）．
  - o が文字列のとき，x 軸の原点に文字を置く位置指定（デフォルトは左下 `"+!UR"`/`"below left"`）．
    o が文字列でなければ無視される．
  - z：x 軸と y 軸の線種指定文字列（デフォルトは実線で Xy-pic のときは `"@-"`）．
  - z または z と o を省略可能．
- `pt=[`p_1`,`p_2`,...]`：（複数の）点を明示するまたは線を描く．仕様は `xy2graph()` の同様なオプションに習う．
- `scale=`r：Xy-pic/TikZ の座標に直すときに r 倍する．
- `scale=[`r_1`,`r_2`]`：Xy-pic/TikZ の座標に直すときに x 座標を r_1 倍，y 座標を r_2 倍する．
- `org=[`x_0`,`y_0`]`：Xy-pic/TikZ の座標に直すときに (x_0, y_0) をXy-pic/TikZ の原点に対応させる．
  上の両者が指定されたとき，座標 (x, y) は Xy-pic/TikZ の座標 $(r_1(x - x_0), r_2(y - y_0))$ に変換される．
- `raw=1`：通過点のリストを返す (他のオプションは無視される)．
  これは `xylines()` のデータとなり，また `ptaffine()` でアフィン変換が計算できる．
  なお，範囲外の点は 0 というデータに変換される．
- $f =$`[0,0]` のときはグラフを描かない．座標軸のみを描くときに用いる．
- `err=c`：函数の定義域を外れるエラーが生じるときは，$c = 1$, $c = -1$ などとしてこのオプションを指定すると，エラーが解消される可能性がある（c は絶対値があまり大きくない実数）．

- verb=1：グラフ上の通過点を ●，制御点を × で表示する．
- verb=[t_1,t_2] , verb=[t_1]: グラフ上の通過点や制御点の描き方を指定する（cf. xylines()）．
- proc=1：execdraw() の描画実行形式を返す．
- proc=2：上と同様であるが，Windows サイズの情報は含めない．
- proc=3：描画実行形式の関数描画成分を返す．
- dvioout=1：グラフを TEX を用いて画面表示する．
- dvioout=t：t がリストのとき，execdraw() を用いて処理する．t は execdraw(ℓ,t) の 2 番目の引数に対応する．このとき ext=[a,b],cl=1 の execdraw() のオプションも指定可能．
  最も簡単な指定は dvioout=[0]，あるいは dvioout=[[0,[500,400]],0] など．

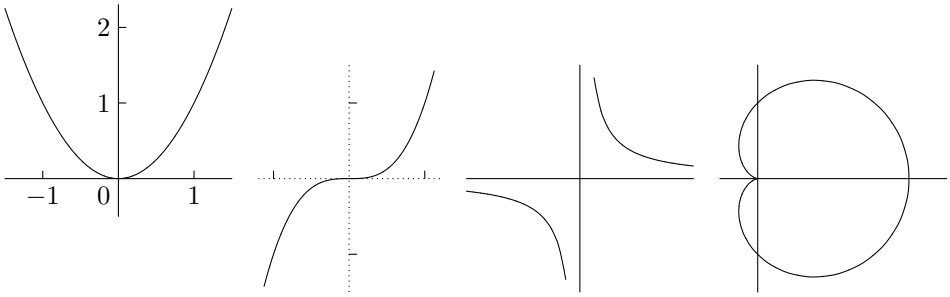
XY-pic のときは

```
[0] os_md.xygraph(x^2,0,[-1.5,1.5],[-1.5,1.5],[-0.5,2.3]|dvioout=1,ax
    =[0,0,1,1,1],scale=10);
[1] os_md.xygraph(x^3,0,[-1.2,1.2],[-1.2,1.2],[-1.5,1.5]|dvioout=1,ax
    =[0,0,1,1],axopt="@{.}",scale=10);
[2] os_md.xygraph(1/x,0,[-3,3],[-3,3],[-3,3]|dvioout=1,ax=[0,0],scale=5);
[3] F=[(1+cos(x))*cos(x),(1+cos(x))*sin(x)]$
[4] os_md.xygraph(F,0,[-@pi,@pi],[-0.5,2.5],[-1.5,1.5]|dvioout=1,scale=10,
    ax=[0,0]);
```

一方，Ti*k*Z のときは，上は

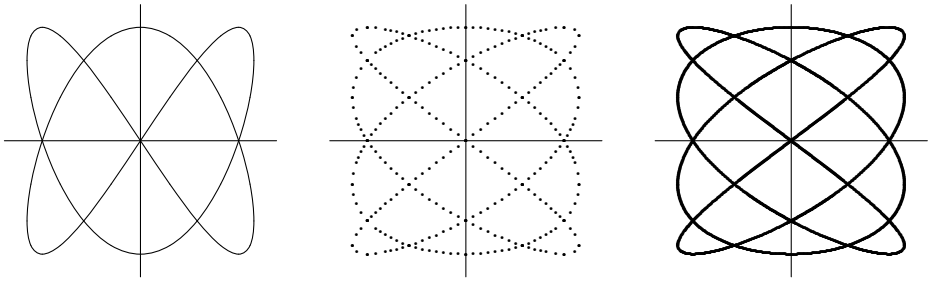
```
[0] os_md.xygraph(x^2,0,[-1.5,1.5],[-1.5,1.5],[-0.5,2.3]|dvioout=1,ax
    =[0,0,1,1,1]);
[1] os_md.xygraph(x^3,0,[-1.2,1.2],[-1.2,1.2],[-1.5,1.5]|dvioout=1,ax
    =[0,0,1,1],axopt="dotted");
[2] os_md.xygraph(1/x,0,[-3,3],[-3,3],[-3,3]|dvioout=1,ax=[0,0],scale=0.5);
[3] F=[(1+cos(x))*cos(x),(1+cos(x))*sin(x)]$
[4] os_md.xygraph(F,0,[-@pi,@pi],[-0.5,2.5],[-1.5,1.5]|dvioout=1,ax=[0,0]);
```

となる．
以下，XY-pic の場合の例であるが，Ti*k*Z の場合は（前者は mm 単位，後者は cm 単位なので）scale= の値を十分の一に，[9] の opt="~*=<3pt>{.}" を opt="dotted" に，[10] の opt="~*={.}" を opt="very thick" などに変更する．



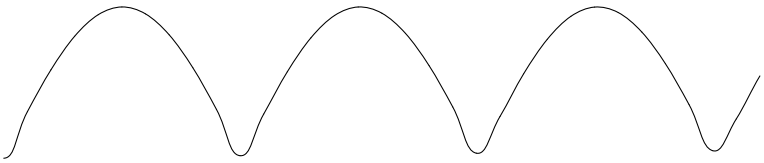

```
[5] F1=[sin(2*x),sin(3*x)]$
[6] os_md.xygraph(F1,-48,[-@pi,@pi],[-1.2,1.2],[-1.2,1.2]|dvioout=1,scale=15,
    ax=[0,0])$
[7] F2=[sin(4*x),sin(3*x)]$
[8] os_md.xygraph(F2,-48,[-@pi,@pi],[-1.2,1.2],[-1.2,1.2]|dvioout=1,scale=15,
```

```
    ax=[0,0],opt="~*=<3pt>{.}")$
[9] os_md.xygraph(F2,-48,[-@pi,@pi],[-1.2,1.2],[-1.2,1.2]|dviout=1,scale=15,
    ax=[0,0],opt="~*={.}")$
```

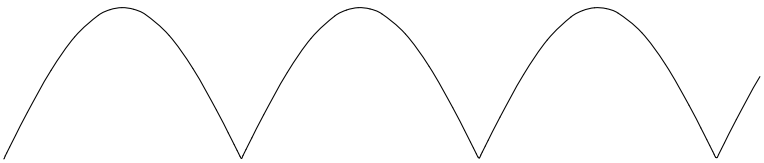


$y = |\sin(x)|$ $(0 \le x \le 10)$ のような微分不可能点をもつ曲線を描くにはオプション `prec` を用いる.

```
[10] F=os_md.abs(sin(x))$
[11] os_md.xygraph(F,-32,[0,10],[0,10],[0,1]|dviout=1,scale=[15,25])$
[12] os_md.xygraph(F,-32,[0,10],[0,10],[0,1]|dviout=1,scale=[15,25],prec=0)$
```



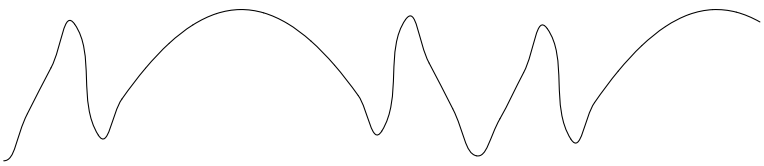
微分可能でない点の近くでは不正確であるが，`prec=0` を指定すると以下のようにより正確になる.



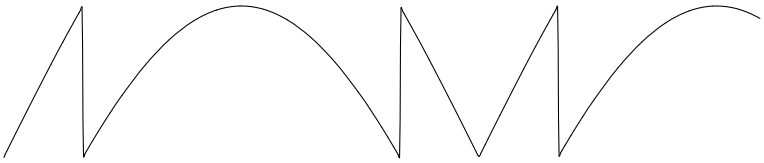
この例では，$[0, 10]$ を 32 等分した関数の値から 32 本の cubic Bézier 曲線を繋げた近似から，不連続点の近くで自動的に細分して 63 本の cubic Bézier 曲線を繋げたものになった.
$y = |2\sin(x)| - \texttt{floor}(|2\sin(x)|)$ $(0 \le x \le 5)$ のような不連続点をもつ曲線を描くにもオプション `prec` を用いる．ここで `floor(`t`)` は t を越えない最大整数を表す.

```
[13] G=[u,[v,dsin,x],[w,os_md.abs,2*v],[z,dfloor,w],[u,0,-z+w]]$
[14] os_md.xygraph(G,-32,[0,5],[0,5],[0,1]|dviout=1,scale=20)$
[15] os_md.xygraph(G,-32,[0,5],[0,5],[0,1]|dviout=1,scale=20,prec=0)$
[16] os_md.xygraph(G,-32,[0,5],[0,5],[0,1]|dviout=1,scale=20,prec=[4,0,1])$
```

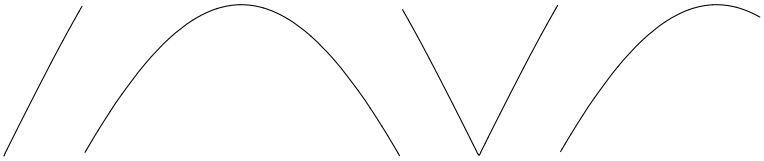
の実行結果は以下のようになる．すなわちオプション `prec` を指定しない場合 `[14]` は



となり，prec=0 のとき [15] は



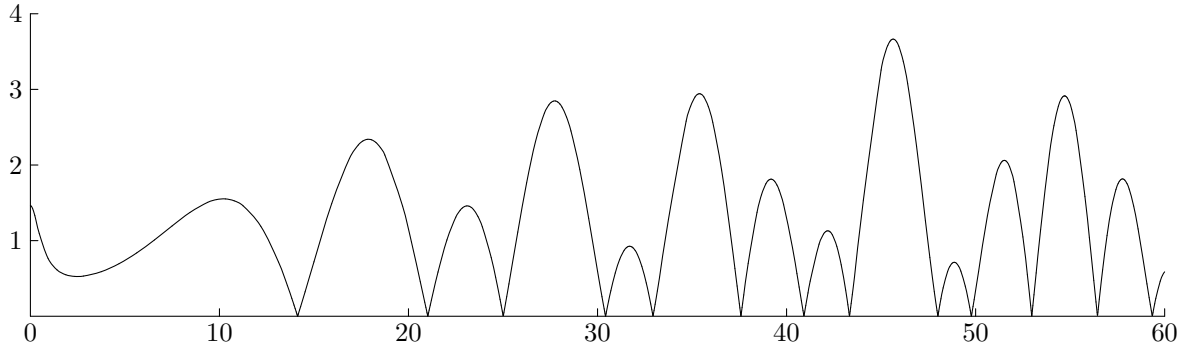
となり，prec=-4 または prec=[4,0,1] のとき [16] は以下のようになる．



この例では，最初 32 本の cubic Bézier 曲線で描かれていたものが，最終的には 70 本の cubic Bezier に分割された．

```
[17] H=[w,[z,os_md.zeta,1/2+@i*x],[w,os_md.abs,z]]$
[18] os_md.xygraph(H,-64,[0,60],[0,60],[0,4]|dviout=1,scale=[2.5,10],prec=6,
     ax=[0,0,10,1,1])$
```

とすると $|\zeta(\frac{1}{2}+x\sqrt{-1})|$ $(0 \leq x \leq 60)$ のグラフが得られる．



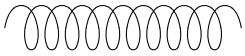
この例では，$[0, 60]$ の 96 等分割から自動的に 355 分割に細分した関数値を得て描かれている．

Risa/Asir のキャンバスでの表示は，たとえば次のようにする（以下の [22] では，重ね書きをしている）．

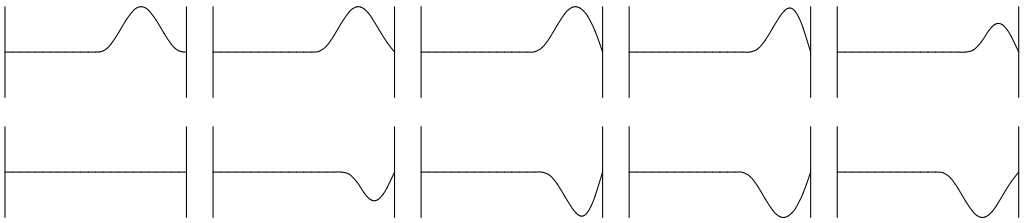
```
[19] dviout0([0,5,6])$
DVIOUTA="%ASIRROOT%\bin\risatex1%TikZ%.bat"
TikZ=1
[20] W=[-1.2,1.2]$
[21] P=os_md.xygraph(F1,-48,[-@pi,@pi],W,W|dviout=[0],ax=[0,0])$
[22] os_md.xygraph(F2,-48,[-@pi,@pi],W,W|dviout=[P],opt="red,dotted")$
[23] S=600$                              /* 窓のサイズ */
[24] os_md.xygraph(F2,-48,[-@pi,@pi],W,W|dviout=[[0,[S]],ax=[0,0])$
```

その他の例を挙げる．

```
[25] G=[1/7*x-1/2*cos(x),sin(x)]$   /* バネ・コイル */
[26] os_md.xygraph(G,-63,[0,21*@pi],[-1,10],[-2,2]|scale=0.3,dviout=1)$
```



```
[27] F=os_md.cutf((1-x^2)^3,x,[[-1,0],[],[1,0]])$  /* 1 つの波 */
[28] G=os_md.periodicf(F,[-4,4],x)$                /* 周期的拡張 */
[29] G1=subst(G,x,x-t-2)$                          /* 進行波 */
[30] G2=subst(G,x,x+t+2)$                          /* 逆の波 */
[31] H=os_md.compdf(x-y,[x,y],[G1,G2]); /* 区間 [0,4] の波，周期 8 */
[32] E="\\draw (2.4,-0.6)--(2.4,-0.6);\n"$         /* 右端 */
[33] for(I=1;I<2;I+=0.2) os_md.xyproc(os_md.xygraph(os_md.mysubst(H,[t,I]),
     -24,[0,4],0,[-1,1]|scale=[0.6,0.6],prec=6,ax=[0,10])+E|dviout=1);
```



これをパラパラ漫画風にする（160 ページ）.

```
[34] T0="\\newpage\n\\begin{tikzpicture}\n"$
[35] T1="\\draw (0,-3) -- (0,3) (16,-3) -- (16,3)\n\\end{tikzpicture}\n"$
[36] Tb=os_md.str_tb(0,0)$
[37] for(I=0;I<8;I+=1/20) os_md.str_tb(T0+os_md.xygraph(os_md.mysubst(H,[t,I]),
     -24,[0,4],0,[-1,1]|scale=[4,3],prec=6)+T1,Tb);
[38] dviout(os_md.str_tb(0,Tb))$
```

356. xy2graph(f,n,[x_1,x_2],[y_1,y_2],[h_1,h_2],α,β|opt=t,scale=r,view=h,raw=1,trans=1,
ax=[z_1,z_2,t],dev=m,acc=k,org=[x_0,y_0,z_0],pt=[p_1,p_2,...],prec=v,title=s,dviout=k,
ext=[a,b],shift=[u,v],cl=1,proc=p)
:: x, y 変数の区間を n 等分して曲面 $z=f(x,y)$ の 3D グラフを描く
- x 軸の正方向から y 軸の正方向に α 度だけ回転した（無限）遠方から β 度 $(-90<\beta<90)$ の角度で見下ろす方向に見た曲面 $z=f(x,y)$ $(x_1\le x\le x_2,\ y_1\le y\le y_2)$ の 3D グラフを描く．なお，投影した高さ方向の座標（y 座標）が $[h_1,h_2]$ に入る範囲のみ描く．
  より具体的には 3 次元の点 (x,y,z) は以下のように平面の点に投影される．

$$(x,y,z)\mapsto(-x\sin\alpha^\circ+y\cos\alpha^\circ,\ z\cos\beta^\circ-x\cos\alpha^\circ\sin\beta^\circ-y\sin\alpha^\circ\sin\beta^\circ)$$

  $\alpha=0$ のときは $\alpha=60$, $\beta=0$ のときは $\beta=15$ と解釈される．
  α は 0 であるか，または 90 の整数倍とは 5 以上離れていることが必要．
  通常は h_1 を十分小さく，h_2 を十分大きく取っておけばよい（指定した範囲の曲面全体の表示）．
- 曲面上で x 座標が定数，あるいは y 座標が定数で定まる曲線を（曲面で隠れる部分，すなわち隠線を消して）描くことで曲面を表す．定数は座標の n 等分点と定める．
  $n<0$ のときも $|n|$ 等分点をとるが，その一つ外側を制御点にとる．
- f は有理函数や $\sin(x)$ などの初等関数に限らず，mydeval() が解釈できる函数ならばよい．ユーザが定義した函数でもよい．
- cpx=1,2,3 を指定するか f に @i が含まれていれば，mydeval() でなくて myeval() が使われる．

- f が 1 変数 z の有理函数ならば，$z = x + yi$ 変数の複素函数と考え $z = |f(x + iy)|$ のグラフを描く．この場合 f は

  ```
  [w,[z,0,x+y@i],[w,os_md.abs,f]]
  ```

  で置き換えられる．なお，複素変数の $\sin(z)$, $\cos(z)$, $\tan(z)$, $\mathrm{atan}(z)$, $\mathrm{asin}(z)$, $\mathrm{acos}(z)$, $\sinh(z)$, $\cosh(z)$, $\tanh(z)$, $\exp(z)$, $\log(z)$, z^w や，それらを含む合成関数や有理関数がサポートされている．

  f が `sin(z^2)+1` のときは

  ```
  [w,[z,0,x+y*@i],[w,os_md.abs,[z__+1,[z__,os_md.sin,z^2]]]]
  ```

  のように置き換えられる．
  $|\Gamma(z)|$ のグラフを描くときは，たとえば f を以下のように取ればよい．

  ```
  [w,[z,0,x+y*@i],[u,os_md.gamma,z],[w,os_md.abs,u]]
  ```

  このときタイトルの函数は `title="\\Gamma(z)"` というオプションで表示できる．
- `scale=`r：表示するために XY-pic/TikZ の座標に直すときに r 倍する．
- `scale=[`r_1`,`r_2`]`：元の曲面の z 座標を $\frac{r_2}{r_1}$ 倍したものを平面に投影したあと，表示するため XY-pic/TikZ の座標に直すときに座標の単位を r_1 倍する．
- `scale=[`r_1`,`r_2`,`r_3`]`：元の曲面の z 座標を $\frac{r_2}{r_1}$ 倍，y 座標を $\frac{r_3}{r_1}$ したものを平面に投影したあと，表示するため XY-pic/TikZ の座標に直すときに座標の単位を r_1 倍する．
- `org=[`x_0`,`y_0`,`z_0`]`：元の座標の (x_0, y_0, z_0) を XY-pic/TikZ の座標での原点にする（デフォルトでは原点が原点に対応）．
- $|n|$ が大きいと TEX のソース・ファイルが長大になり，ソース・ファイルから dvi ファイルや pdf ファイルへの変換に時間がかかることがあるので注意．
  $n = -16$ がデフォルト（$n = 0$ とするとデフォルト値と解釈される）．
- `view=1`：陰線消去を行わない．
  `view=-1`：陰線は点線で表示する．
- `raw=1`：通過点のリストを返す．
- `dev=`m：陰線消去のためのメッシュを，x 変数, y 変数とも $m \times |n|$ 等分したものとする（デフォルトは $m = 16$）．ただし `dev=[`m_1, m_2`]` とすると，m_2 は出力される x 座標のメッシュの細かさ，m_1 は曲線間にとるメッシュの細かさ，と別に指定することができる．
- $|n| \times m$ を増やすと処理時間が増える．特に f が多項式や有理函数でなくて三角関数や指数関数，対数関数，べき函数などを含むときは注意．なお処理時間は，ほぼ $|n|^2 \times m^2$ に比例．
- `acc=`k：描く曲線の等分点の個数を約 k 倍にする（k は実数．曲線の本数は不変）．$k = 2$ なら n を 2 倍にして `dev` を半分にした場合の曲線を一本おきに描くことにほぼ等しい（結果のファイル・サイズは，$n^2 \times k$ にほぼ比例）．
- `err=c`：有理函数の分母が 0 になるなどの，函数の定義域を外れるエラーが生じるときは，$c = 1$, $c = -1$ などとしてこのオプションを指定すると，エラーが解消される可能性が大きい（c は絶対値があまり大きくない実数）．
- `prec=`v：`xygraph()` の同様のパラメータと同じ．
- `ax=[`z_1`,`z_2`]`：x, y, z の座標が (x_2, y_2, z_1), (x_1, y_1, z_2) を対角線の頂点とする直方体の枠を書く，
- `ax=[`z_1`,`z_2`,`t`]` ：上に加え，一部の頂点の座標を入れる．
  - $t = \pm 1, \pm 5$：2 頂点の座標を入れる．
  - $t = \pm 2, \pm 6$：4 頂点の座標を入れる．
  - $t = \pm 5, \pm 6$：座標の文字を小さくする．
  - $t = 0, -8$：頂点の座標を入れない．
  - $t < 0$：直方体の後ろ側の枠を点線で描く（デフォルトは実線）
- 複素変数の実数値函数のときに座標を入れる形式は，$(x, y, z) = (1, 2, 3)$ の場合に
  - `cpx=1`：$(1 + 2i, 3)$ （デフォルト）
  - `cpx=2`：$(1 + 2\sqrt{-1}, 3)$

– cpx=3：$(1,2,3)$

- opt=u：曲面内の曲線を描くときのオプション文字列を指定する．表側と裏側で変更するときは opt=[u_1,u_2] と，表側，裏側の順に指定する．
  TikZ のときは，さらに直方体の枠を描くときのオプション文字列が指定できる．
  たとえば TikZ で opt=["black","red","blue"] とすれば，表側を黒，裏側を赤，枠を青で描く．
- title=s：画面表示するとき，函数名が s で表示される（s は TEX の数式モードでの文字列 ）．函数がリスト形式のときの函数名表示に役立つ．
- pt=[p_1,p_2,...]：（複数の）点を明示する，あるいは線を引く．
  – p_i=[x_i,y_i,z_i] とすると (x_i,y_i,z_i) を ● で明示する，
  – p_i=[[x_i,y_i,z_i],s_i] とすると，● （すなわち\bullet）でなくて s_1 で明示される（s_i は数式モードの文字列で，\ は \\ で表す）．
  – p_i=[[x_i,y_i,z_i],s_i,t_i] とすると，● でなくて s_i で明示され（s_i は文字列），XY-pic のときは，そのあとに文字列 t_i を置く（s_i が文字列でないときは ● のまま）．s_i が不要なときは，s_i = "" とする．
  TikZ のときは，t_i はラベル付け文字で，s_i は $s_i=[s_{i,0},s_{i,2}]$ というリストでよい．後に置く文字があるときは，t_i の後に t_i,u_i と続ける．
  t_i=1 は "_" というラベルを表し，その後で座標の代わり 1 または"_" で参照できる．
  – 点が一つの時は，pt=p_1 としてよい．
  – pt=[0,0,1]：$(0,0,1)$ を ● で明示する．
  – pt=[[0,1,2],"$\\times$"]：$(0,1,2)$ を × で明示する．

  XY-pic のときは
  * pt=[[0,0,1],0," *+!D{(0,0,1)}"]：$(0,0,1)$ を ● で明示し，その上に $(0,0,1)$ と表示する．
  * pt=[[[0,0,1],0," *+!D{(0,0,1)}"],[[0,1,2],"$\\times$"]]]：上の両方を行う．
  * pt=[[0,0,1],0,"=\"A\""]：点 $(0,0,1)$ を ● で明示し，ラベル "A" をつける．

  TikZ のときは上と同様なことは以下のように書ける
  * pt=[[[0,0,1],0,1],[1,["below","$(0,0,1)$"]]]：$(0,0,1)$ を ● で明示し，その上に $(0,0,1)$ と表示する．
  * pt=[[[0,0,1],0,1],[1,["below","$(0,0,1)$"]],[[0,1,2],"$\\times$"]]]：上の両方を行う．
  * pt=[[0,0,1],0,"A"]：点 $(0,0,1)$ を ● で明示し，ラベル "A" をつける．

  – p_i=[[x_i,y_i,z_i],[x_i',y_i',z_i']] とすると，2 点 (x_i,y_i,z_i) と (x_i',y_i',z_i') を線で結ぶ．
  – p_i=[[x_i,y_i,z_i],[x_i',y_i',z_i'],t_i] とすると，2 点 (x_i,y_i,z_i) と (x_i',y_i',z_i') を結ぶが
  * $t_i=0,1$：実線で結ぶ
  * $t_i=2$：点線で結ぶ
  * $t_i=-1$：曲面のあるところは除いて実線で結ぶ
  * $t_i=-2$：曲面のあるところは除いて点線で結ぶ

  さらに t_i の後ろにオプション文字列 u_i を書いて p_i=[[x_i,y_i,z_i],[x_i',y_i',z_i'],t_i,u_i] とし，TikZ では色や線の太さ指定などの指定が可能．
- proc=1：execdraw() の描画実行形式を返す．
- proc=2：上と同様であるが，Windows サイズの情報は含めない．
- dviout=1：画面表示する．
- dviout=2：ディスプレイスタイルで函数式も含めて画面表示する．
- dviout=3：さらに視点の角度と，もとの曲面の高さや y 座標のスケールを変えたとき，その倍率を表示する．
- k が -1, -2, -3 の時は，dviout=$|k|$ に対応する TEX のコードがリスト形式で　出力される．
  リストの最後の成分は xyproc() によってグラフの TEX のソースとなる．その前の成分は函数式

などを表す TEX のソース，tarns=1 を指定した場合はその結果が先頭につく．

- dviout=k：k がリストのとき，execdraw() を用いて処理する．k は execdraw(ℓ,k) の 2 番目の引数に対応する．このとき ext=[a,b],shift=[u,v],cl=1 の execdraw() のオプションも指定可能．
- dviout=k：この指定が無いときは，戻り値を S したとき，xyproc(S) がグラフの TEX のコードとなる．
- trans=1：もとの (x,y,z) 座標に対応する XY-pic/TikZ の座標 $[X,Y]$ を返す．具体的には

$$X = -r_1(x-x_0)\sin\alpha^\circ + r_2(y-y_0)\cos\alpha^\circ,$$
$$Y = r_3(z-z_0)\cos\beta^\circ - r_1(x-x_0)\cos\alpha^\circ\sin\beta^\circ - r_2(y-y_0)\sin\alpha^\circ\sin\beta^\circ.$$

ただし，これはオプションなどの指定が
  - xy2graph の第 6, 7 引数：(α,β)　($\alpha=0$ のときは $\alpha=60$, $\beta=0$ のときは 15 と解釈される)
  - scale=[r_1,r_3,r_2]　デフォルトは [1,1,1] となり，r_2 を指定しないときは $r_2=r_1$ で，r_3 を指定しないときは $r_3=r_1$ と解釈される．
  - Org=[x_0,y_0,z_0]　デフォルトは [0,0,0]

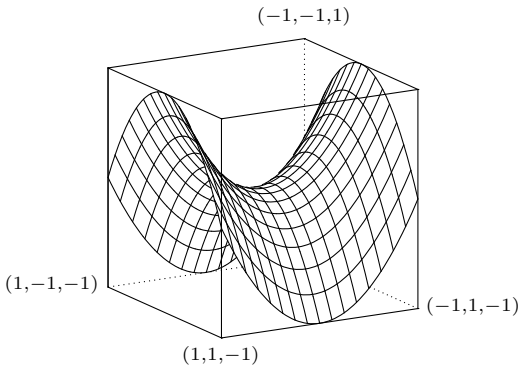
の場合である．
F=os_md.xy2graph(⋯|trans=1) とおくと，(x,y,z) に対応する XY-pic/TikZ での座標は

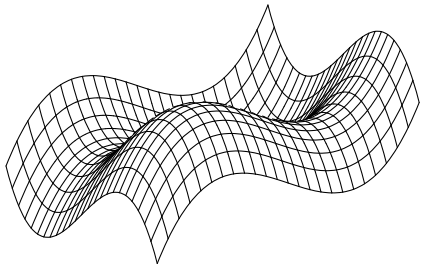
os_md.myf3deval([F],x,y,z)

によって得られる．
- pdf ファイルを作成するときは，\usepackage[pdf,all]{xy}と指定して dvipdfmx などを用いるとよい．ジャギーがなく，サイズの小さな pdf ファイルが，より短い時間で作成できる．また，複雑な画像でも処理が出来る．
  pdf ファイル作成と dvi ファイル作成とを切り替えるなど，異なるやり方で TEX ファイルを処理するには DVIOUTB および dviout0(4) を利用するとよい．

```
[0] os_md.xy2graph(x^2-y^2,0,[-1,1],[-1,1],[-2,2],0,0|ax=[-1,1,-6],scale=15,
dev=64,dviout=3)$
[1] os_md.xy2graph(-x^3-y^3,-24,[-1,1],[-1,1],[-2,2],60,-35|scale=20,dev=64,
dviout=2)$
```

$z=x^2-y^2$　$(-1\le x\le 1,\ -1\le y\le 1)$　　$z=-x^3-y^3$　$(-1\le x\le 1,\ -1\le y\le 1)$





上は XY-pic のときで，TikZ を用いる場合の scale の値は $\frac{1}{10}$ 倍にする．

```
[2] S0=[[3.1416,0,0],0,"*+!U{(\\pi,0,0)}"]$
[3] S1=[[0,0,0],0,"*+!U{(0,0,0)}"]$
```

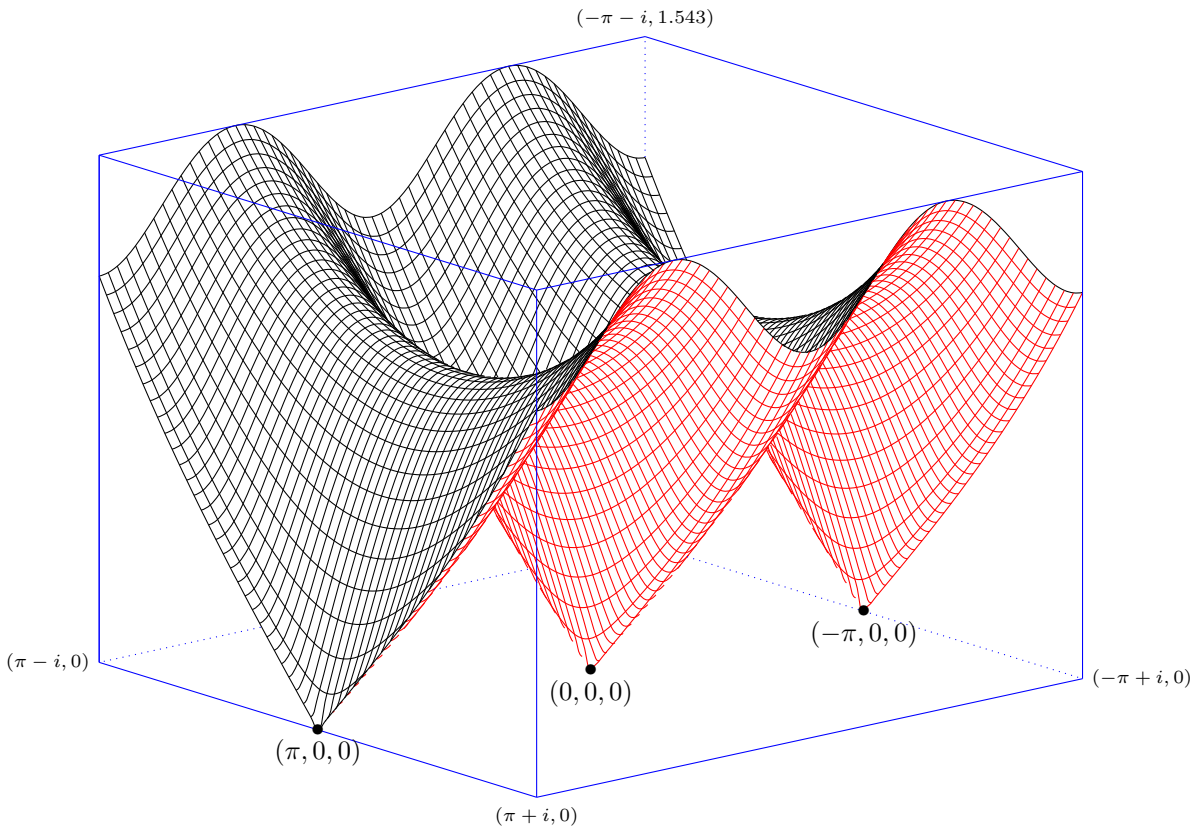
```
[4] S2=[[-3.1416,0,0],0,"*+!U{(-\\pi,0,0)}"]$
[5] S3=[[3.1416,0,0],[-3.1416,0,0],2]$
[6] os_md.xy2graph(sin(z),-60,[-@pi,@pi],[-1,1],[-5,8],50,0|
    scale=[15,45,45],ax=[0,1.543,-6],dviout=3,pt=[S0,S1,S2,S3])$
```

上の入力から次のグラフの描画までに，30 秒程度かかる（2014 年におけるメジャーなパソコン）．なお，有理函数のグラフなら 10 秒程度．
TikZ のときは，上の `[2]`，`[3]`，`[4]` `[6]` は次のようになる．

```
[2] S=[[[3.1416,0,0],0,1], [1,["below","$(\\pi,0,0)$"]]]$
[3] S=append([[[0,0,0],0,1],[1,["below","$(0,0,0)$"]]],S)$
[4] S=append([[[-3.1416,0,0],0,1],[1,["below","$(-\\pi,0,0)$"]]],S)$
[6] os_md.xy2graph(sin(z),-60,[-@pi,@pi],[-1,1],[-5,8],50,0|dviout=3,
    scale=[1.5,4.5,4.5],ax=[0,1.543,-6],pt=S,opt=["black","red","blue"])$
```

`append( )` を使わずに，`os_md.m2l([S1,S2,S3]|list=1)` などとする方法もある．この結果は図 2 である．

図 2　`xy2graph()` の例

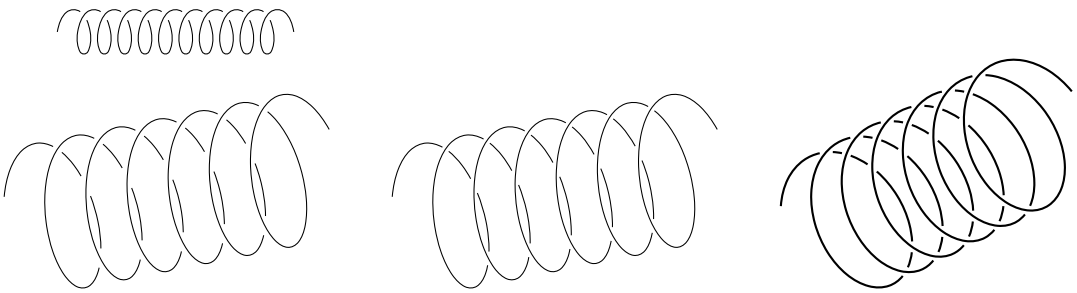


357. xy2curve([f_1,f_2,f_3],n,[t_1,t_2],[y_1,y_2],[z_1,z_2],α,β|scale=r,gap=g,opt=s,eq=q,raw=w,dviout=d)

:: 空間曲線の隠線処理つき描画

- $\big(f_1(x),f_2(x),f_3(x)\big)$　$(x\in[t_1,t_2])$ で定義される空間曲線を x 軸の正方向を y 軸の正方向に α 度だけ回転した無限遠方から，β 度 $(-90<\beta<90)$ 見下ろす方向に回転して，yz 平面に射影した曲線を描く．
  ただし変数が x でなくて t ときは，[t_1,t_2] の代わりに [t,t_1,t_2] と書く．
  描画範囲は y 座標，z 座標が区間 $[y_1,y_2]$，$[z_1,z_2]$ の範囲（回転や以下のスケール変換後）．
- gap=g：隠線消去の幅を g mm を基準とする．$g=0$ のときは，隠線消去は行わない．デフォルトは $g=0.7$.
- gap=[g,m]：隠線消去の幅の最大値を $m\times g$ mm とする（デフォルトは $m=3$）.
- scale=r：各座標を r 倍する．
- sacle=[r_1,r_2]：x 座標を r_1 倍，y 座標と z 座標を r_2 倍する．
- sacle=[r_1,r_2,r_3]：x 座標を r_1 倍，y 座標を r_2 倍，z 座標を r_3 倍する．
- α が 3 行 3 列の行列の場合は，これによる線形変換に置き換える（回転やスケール変換は無視）.
- opt=s：線種などの指定文字列（xygraph() のときと同様）．ただし gap=0 を指定しないときは，xybezier() に渡すパラメータの形式に拡張される．
- eq=q：空間内で q mm 以下の距離の 2 点は同じ点とみなす（デフォルトは $q=0.01$）.
- dviout=1：画面表示を行う．
- raw=1：TEX のソースを返す．
- raw=2：隠線処理後の区分 Bézier 曲線を与える座標のリストのリストを返す．
- raw=3：隠線処理前の区分 Bézier 曲線を与える座標のリストのリストと，交点の情報を返す．
- raw=4：隠線処理前の区分 Bézier 曲線を与える座標のリストのリストと，区分点などのデータを返す．
- raw=5：隠線処理前の回転やスケール変換後の曲線を返す．

```
[0] F=[cos(x),1/7*x-1/2*cos(x),sin(x)]$      /* バネ */
[1] os_md.xy2curve(H,-63,[0,21*@pi],[-1,10],[-2,2],0,0|scale=0.3);
[2] G=[cos(t),t/10,sin(t)]$V=[t,0,40]$
[3] os_md.xy2curve(G,-96,V,[-2,6],[-4,4],30,-20|dviout=1);
[4] os_md.xy2curve(G,-96,V,[-2,6],[-4,4],30,-20|dviout=1,gap=0.5);
[5] os_md.xy2curve(G,-96,V,[-2,6],[-4,4],50,-25|dviout=1,opt="thick",gap=0.6);
```



358. execdraw(ℓ,t|shift=[u,v],ext=[a,b],cl=1)

:: 描画実行形式 ℓ を出力形態 t に従って実行する

ℓ や t はリストであるが，成分が一つの時は成分のみを指定してもよい．

$t=$[t_0,t_1,t_2] は，t_0 により出力先を，t_1 により描画実行形式と出力先との座標との対応を，t_2 により描画実行形式の原点の移動を指定する．

- t_2 を指定しないときは，$t=$[t_0,t_1] のように，また t_1 と t_2 を共に指定しないときは，$t=$[t_0] のように略してよい．
- t_0 が数で t_1, t_2 を指定しないときは，$t=t_0$ としてよい．

- `cl=1`：表示画面をクリアーしてから描画する．

出力先を指定する t_0 は

- 0：`Risa/Asir` におけるキャンバスを用いた描画．
  リスト形式で `[0]` あるいは `[0,[`*width*`,`*height*`]]` によって新規キャンバス，
  または `[0,`*index*`]`，あるいは `[0,`*id*`,`*index*`]`，あるいは `[0,`*id*`,`*index*`,`*width*`]`，`[0,`*id*`,`*index*`,`*width*`,`*height*`]`，によって描画キャンバスを指定する．
  - *id* はサーバ Id，*index* はキャンバス Id，*width*, *height* は pixel 単位のサイズ（cf. `draw_obj()`）
  - `[0,[`*width*`,`*height*`]]` において *width* = *height* のときは `[0,[`*width*`]]` としてよい．ただし
  - *t*`=[[0,[`*x*`]]]` は，*t*`=[[0,[`*x*`,`*x*`]],`*x*`]` と解釈される（すなわち $t_1 = x$）．
  - `execdraw([],0)` は，描画画面を新規に開く．
  - `shift=[`*u*`,`*v*`]`：右方向に *u*，上方向に *v* だけ pixel 単位で移動する．
  - `ext=[`*a*`,`*b*`]`：左側から *a*，右側から *b* だけ pixel 単位で縮めて描く．
  - `ext` や `shift` は，キャンバスに応じて描画サイズが変更されても文字列の大きさは不変であることに対処するのが主目的のオプションである．
  - ℓ 中の Window の x 座標と y 座標の範囲指定が同一で，横幅 t_1 と `shift` と `ext` の設定が同じであれば（t_2 は設定していないか同じとする），同一キャンバスに同じ座標系で上書き表示される．
  - t = `[`t_0`,`t_1`,`t_2`,`ℓ'`]` により，別の描画実行形式 ℓ' のものを用いることができる（t_1 はキャンバスの横幅とする．それが t_0 で指定される場合は $t_1 = 0$ としてもよい）．
    なお，ℓ' は，キャンバスサイズとは独立した情報である．
  - 戻り値は `[0,`*id*`,`*index*`]`，または `[0,`*id*`,`*index*`,`*width*`,` *height*`]`．
    これは，同じキャンバスへの描画のときの t_0 として用いることができる．
    $t_1 = 0$, $t_2 = -1$ のときは，さらに 2 つの 0 と変換情報とを付加して組にしたリストが戻される．最後（4 番目）の成分は上の ℓ' として用いることができる．また戻り値は，そのまま t として用いることにより，同じ座標でキャンバス上に他の描画実行形式の**重ね描き**ができる．

  たとえば t は以下のようになる
  - 0：`Canvas` で設定されたサイズのキャンバスを新規に開いて，それに合わせた横幅で表示．
    `[[0,[300,300]],300]` や `[[0,[300]]]` と同じ．
  - `[[0,[300,400]],280]`：300 × 400 pixel のキャンバスを開き，横幅 280 pixel で表示．すなわち，t_0 = `[0,[300,300]]`，t_1 = `280`.
  - `[[0,[500]],0]`：500 × 500 pixel のキャンバスを開き，スケールと位置を自動調整して描画．
  - `[[0,[500]],0,-1]`：上と同様で，さらに調整情報も含めて返す．
  - `[[0,2,3],400]`：サーバ Id が 2，キャンバス Id が 3 の窓に，横幅 400 pixel で表示
  - `[[0,2,3,400,400],0]`：サーバ Id が 2，キャンバス Id が 3, 400 × 400 pixel のキャンバスに，描画スケールと位置を自動調整して描画．
  - `[[0,2,3,400,400],0,0,T]`：上と同じであるが，調整は `T` に従う．
- 1：TEX のソースを返す．
- 2：`dviout()` による画面表示．
- −1：窓のサイズを表示してその情報を返す．
  t`=[-1,[`x_0`,`x_1`],[`y_0`,`y_1`],`*k*`]`：窓のサイズを再定義した ℓ を返す．
  t`=[-1,[`x_0`,`x_1`],[`y_0`,`y_1`],`*k*`,[`*a*`,`*b*`],[`*u*`,`*v*`]]` も可能．
  後者の代わりにオプション `shift=[`*u*`,`*v*`],ext=[`*a*`,`*b*`]` でも指定可能．
- −2：表示はせずに窓のサイズを返す．
- −3：中身を読んで描画の箱サイズを返す．2 項目は `Risa/Asir` で表示の際の文字列の最大長で，その後は文字表示の箱サイズ（文字列表示があると不正確）．
- −4：含まれる位置情報の個数を返す．
- −5：含まれる実行形式の函数を返す（ℓ の第 1 成分から重複と負の数を除いたリスト）．
  たとえば，`delopt(`ℓ`,[0,1]|inv=1)` とすると，描画実行形式 ℓ から第 1 成分が 0（描画範囲）と 1（Bézier 曲線描画）のもののみを抜き出した描画実行形式を得ることができる．

Risa/Asir の描画キャンバスでの表示の場合

- $t_1 = 0$: ℓ の描画範囲の横幅と左上端の位置をキャンバスに合わせる.
  さらに微調整するには，キャンバスの縦横比率を変更したり，オプションパラメータの ext と shift を用いるとよい.
  このとき
  - $t_2 = 1$: 描画範囲の箱サイズの他，自動的に得た ext と shift の値が表示される（オプションパラメータは加味されない）.
  - $t_2 = -1$: 上の情報が戻り値に付加されたリストを返す（オプションパラメータも加味される）. この戻り値が R のとき，execdraw($\tilde{\ell}$,R) とすると，同じキャンバスに同じ座標で, $\tilde{\ell}$ の**重ね書き**ができる. また，execdraw(ℓ,[t_0,0,0,R[3]]) や execdraw(ℓ,[t_0,0,-1,R[3]]) のようにして（新規，または既存の）別のキャンバスに描くことも出来る.

    たとえば，戻り値は

    ```
    [[0,2,3,500,400],0,0,[0,[-1.2,1],[-1.2,1],0,[10,10],[0,-10]]]
    ```

    のような形になる.

    最初の [0,2,3,500,400] は t_0 に対応する部分で，順に，0 は Risa/Asir のキャンバスへの描画，そのサーバ Id は 2，キャンバス Id は 3，サイズが 500×400 pixel であることを意味する.

    次の成分の 0 は表示の際の調整を意味し，その次の 0 は特に意味はなく，最後の成分の

    ```
    [0,[-1.2,1],[-1.2,1],0,[10,10],[5,-15]]
    ```

    は描画実行形式の成分の形である. すなわち

    描画実行形式の箱がその座標の単位で [-1.2,1]×[-1.2,1] であって，次の 0 は単位あたりの実長が無定義を意味し，ext=[10,10], shift=[5,-15] で Risa/Asir のキャンバスに対応させることを意味している. より具体的には

    ext によってキャンバスの横幅 500 pixel の左右に 10 pixel ずつ余裕を持たせるので，座標の単位は描画の際に $\frac{500-10-10}{1-(-1.2)}$ 倍される. また，描画実行形式の箱の左上端が，キャンバスの左上端から右に $(10+5)$ pixel, 下に 15 pixel の位置になる.

    なお, 画実行形式の箱の外に描画要素があっても，それがキャンバス内に入れば描画される.
- キャンバスへの描画は，まず描画実行形式から描画範囲の箱を読み取ってそれによりキャンバスとの座標対応を定め，次に描画実行種別番号 2 の文字列表示を先頭から順に実行し，最後に残りを先頭から順に実行する. これはキャンバスにおける文字列表示において，文字でなくて文字の箱が上書きされることに対応するため.
- $t_1 > 0$ のときは Windows の横幅が t_1 pixel になるようにスケール変換される（$t_1 = 1$ は設定に従う）.
- t_1 がリストや行列の時は，以下のキャンバス表示でない場合と同じ.

Risa/Asir の描画キャンバス以外の場合は, t_1 はスケール変換を表す (TEX 利用などの場合に有益).

- t_1 が 2×2 の行列の時，座標を t_1 によって線型変換する.
- $t_1 =$ [t_{10}, t_{11}] とすると x 座標が t_{10} 倍，y 座標が t_{11} 倍される（負の値も可能）.

$t_2 =$ [x, y] は，座標 (x, y) を原点に移動する平行移動.
t_1 によるスケール変換の前に行う. Risa/Asir の描画キャンバスでは無効.

以下，**描画実行形式** ℓ について説明する.

- ℓ=[ℓ_1,ℓ_2,...], ℓ_j=[f_j,...] は，execproc() の形式に従う. ただし，f_j が整数の ℓ_j（**描画実行識別番号**）は次のような画像描画の特別形式である.
  - [0,[x_0,x_1],[y_0,y_1],k] : 窓の x 座標と y 座標の範囲. 座標の 1 が k cm に対応する. k が省略されたり, $k = 0$ の場合は，$k = 1$ と解釈される.
    ℓ にこの成分が複数あるときは，最初のものが有効. ℓ の初めの方に書かれるのが望ましい.
  - [0,[x_0,x_1],[y_0,y_1],k,[a,b],[u,v]] : 上の拡張としてこの形式がある. これは Risa/Asir での表示の際のみ意味を持ち，デフォルトで ext=[a,b], shift=[u,v] というオ

プションが設定される．
  - `[1,`p_j`,`ℓ_{j1}`,`ℓ_{j2}`,...]`：Bézier 曲線描画，p_j は `xybezier()` のオプション．$\ell_{j\nu}$ はその引数．すなわち `xybezier(`$\ell_{j\nu}$`|optilon_list=`p_j`)` $(i = 1, 2, \ldots)$ に対応する描画を意味する．あるいは，$\ell_{j\nu}$ は `lbezier()` で変換されたデータでもよい．
  - `[2,`p_j`,[`x_j`,`y_j`],[`s_j`],`t_j`]`：文字列描画函数 `xyput([`x_j`,`y_j`,`s_j`])` に対応．
    t_j は `Risa/Asir` のキャンバス表示での代替文字列（設定しなくてもよい）．
    s_j が文字列の時は，`[2,`p_j`,[`x_j`,`y_j`],`s_j`,`t_j`]` としてもよい．
  - `[3,`p_j`,[`x_{j1}`,`y_{j1}`],[`x_{j2}`,`y_{j2}`],...]`：線分描画函数 `xyarrow([`x_{j1}`,`y_{j1}`],[`x_{j2}`,`y_{j2}`])` に対応．
  - `[4,`p_j`,[`x_{j1}`,`y_{j1}`],[`x_{j2}`,`y_{j2}`],...]`：線分描画函数 `xyline([`x_{j1}`,`y_{j1}`],[`x_{j2}`,`y_{j2}`])` に対応．
  - `[5,`p_j`,`s_j`]`：TEX の文字列描画の `dviout(`s_j`)` に対応．
  - `[-1,...]`：この項は無視される（コメントなど）．
  - `[-2,`s`]`：この項の s は文字列で，TEX のソースには，先頭に % を付加してコメントとして記され，そのあと改行される．
- p_j はオプションリストに対応する（f_j が函数子の場合は，ℓ_j の 3 番目の成分にあたる）．
- 複数の描画実行形式は `append()`，または `m2l( |list=1)` によって一つにまとめることが出来る（グラフの**重ね合わせ**に対応）．
- `ptaffine(`m`,`ℓ`|proc=1,shift=`w`,...)` で affine 変換（平行移動と線形変換）ができる．
- `[1,`p_j`,`ℓ_{j1}`,`ℓ_{j2}`,...]` における $\ell_{j\nu}$ はリストで，標準の形の場合，各成分は座標（すなわち実数の 2 つの組のリスト）または，区切り記号の数字の 0, 1, −1 からなり，複数の Bézier 曲線を表わす．

  一つの Bézier 曲線は始点，いくつかの制御点，終点とが連続して並んで定義されている．$n+1$ 個並んでいれば，n 次の Bézier 曲線となる．PDF 化などから 3 次までの Bézier 曲線が望ましい．なお，1 次の Bézier 曲線は制御点がなく，線分を表す．

  0 は Bézier 曲線の区切りで，1 は区切った後に直前の終点の座標を次の Bézier 曲線の始点とみなす（座標を挿入する）ことを意味する．1 で区切られていると連続曲線となるが，−1 は連続曲線の先頭を終点として閉曲線とすることを意味する．たとえば

  `[[`x_1`,`y_1`],[`x_2`,`y_2`],1,[`x_3`,`y_3`],1,-1]`

  は (x_1, y_1), (x_2, y_2), (x_3, y_3) を頂点とする三角形を意味する．

  これが標準の形で，TEX で Ti*k*Z のソースを作成するときは，一つの $\ell_{j\nu}$ が一つの `\draw` 文に変換される．

  一方，$\ell_{j\nu}$ は一つの Bézier 曲線を表す座標のリストを成分とするリストの形のものも許される．この場合は，区切り記号の数字は存在しない．TEX で Ti*k*Z のソースを作成する場合は，一つの Bézier 曲線毎に `\draw` 文が生成される．上の三角形は，次のように表せる．

  `[[[`x_1`,`y_1`],[`x_2`,`y_2`]],[[`x_2`,`y_2`],[`x_3`,`y_3`]],[[`x_3`,`y_3`],[`x_1`,`y_1`]]]`

  $\ell_{j\nu}$ にはこの二つの形式が許されるが，両者は `lbezier()` によって相互変換できる．

`R` が描画実行形式の時，たとえば
- `execdraw(R,[[0,[600]],0])`：600 × 600 の `Risa/Asir` のキャンバスを開いて表示する
- `execdraw(R,[[0,[600,700]],0]|ext=[30,20],shift=[10,-40])`：600 × 600 の `Risa/Asir` のキャンバスを開いて表示する．
  その際，左側を 30 pixel, 左側を 20 pixel 内側になるように描画を縮小する（負の値で拡大も可）．さらに右に 10 pixel，上に 40 pxel ずらす．
- `execdraw(R,[[0,1,5,600],0]|ext=[30,20],shift=[10,-40])`：サーバ ID が 1，キャンバス Id が 5 のキャンバスに，横幅が 600 pixel として上と同様に表示する．
- `execdraw(R,[1,1.5,[1,1]])`：(1, 1) を原点に並行移動して，さらに画像を 1.5 倍に拡大した TEX のソースを出力する．

```
[0] os_md.dviout0([0,5,6])$                              /* TikZ 形式に     */
[1] F1=[sin(2*x),sin(3*x)]$
[2] W=[-1.2,1,2]$
```

```
[3] R=os_md.xygraph(F1,-48,[-@pi,@pi],W,W|proc=1,ax=[0,0])$  /* リサージュ曲線 */
[4] P=os_md.execdraw(R,0)$          /* Risa/Asir のキャンパスで表示      */
[5] S=os_md.execdraw(R,[1,1.5])$  /* 画像を 1.5 倍した TeX のソースを出力 */
[6] os_md.execdraw(R,[2,1.5])$    /* 1.5 倍した画像を TeX を用いて表示    */
[7] F2=[sin(3*x),sin(4*x)]$
[8] R2=os_md.xygraph(F2,-48,[-@pi,@pi],W,W|proc=2,opt="red")$
[9] R2=append(R,R2)$                                    /* グラフの重ね合わせ */
[10] os_md.execdraw(R2,[2,1.5])$
```

上の [4], [5], [6] では，描画実行形式 R に対して，それぞれ，Risa/Asir のキャンバスで表示，TEX のソースに変換，TEX を使って PDF に変換して画面表示を行っている．[5], [6] では，グラフを 1.5 倍に拡大している．
R2 は 2 つのグラフを**重ねた描画実行形式**で，[10] ではそれを表示している．

```
[11] S=os_md.xy2graph(x^2-y^2,0,[-1,1],[-1,1],[-2,2],0,0|ax=[-1,1,-6],scale=1.5,
 dviout=3,proc=1)$
[12] os_md.execdraw(S,[[0,[500,400]],500]|ext=[80,70],shift=[0,90])$
[13] os_md.execdraw(S,[[0,[500]],0])$
[14] os_md.execdraw(S,-1);
Windows :  -2.04904 < x < 2.04904 ,  -3.42811 < y < 3.42811   by 1 cm
[[-2.04904,2.04904],[-3.42811,3.42811],1]
[15] os_md.execdraw(S,-3);
[[-2.04904,2.04904],[-1.97922,1.97922],9,[[-2.04904,2.04904],
 [-1.97922,1.97922]]]
[16] os_md.execdraw(S,-4);
1366
[17] os_md.execdraw(S,-5);
[0,1,2,5]
```

3.2.12 Applications

いくつかの函数を応用した例を述べる．

3.2.12.1 表の作成

359. powprimroot(p,n|all=1,exp=1,log=f)
:: p 以上の素数 n 個とその原始根のリストを作る
- all=1 各素数に対し最小原始根のみでなく，全ての原始根をリストする
- exp=1 最小原始根とそのべき（指数関数の値）のリストを作る
- log=1 最小原始根に対する対数関数の値のリストを作る
- log=2 上の対数関数のリストにおいて，べき 0 を使わない
- log=3 最小原始根に対する対数関数の素数のときの値のリストを作る

```
[0] os_md.powprimroot(3,4);
[[3,2],[5,2],[7,3],[11,2]]
[1] os_md.powprimroot(3,4|all=1);
[[3,2],[5,2,3],[7,3,5],[11,2,6,7,8]]
[2] os_md.powprimroot(3,4|exp=1);
[[$p$,1,2,3,4,5,6,7,8,9,10],[3,2,1],[5,2,4,3,1],[7,3,2,6,4,5,1],
```

```
[11,2,4,8,5,10,9,7,3,6,1]]
[3] os_md.powprimroot(3,4|log=1);
[[$p$,1,2,3,4,5,6,7,8,9,10],[3,0,1],[5,0,1,3,2],[7,0,2,1,4,5,3],
[11,0,1,8,2,4,9,7,3,6,5]]
[4] os_md.powprimroot(3,4|log=2);
[[$p$,1,2,3,4,5,6,7,8,9,10],[3,2,1],[5,4,1,3,2],[7,6,2,1,4,5,3],
[11,10,1,8,2,4,9,7,3,6,5]]
[5] os_md.powprimroot(3,4|log=3);
[[3,2,1],[5,2,1,3],[7,3,2,1,5],[11,2,1,8,4,7],[$p/a$,$r$,2,3,5,7]]
```

実際に ltotex(|opt="tab") によってリストを TEX の表にして表示してみよう.

```
[6] T="素数$p$の原始根とそのべきを$p$で割った余り"$
[7] L=os_md.powprimroot(3,8|exp=1)$
[8] Out=os_md.ltotex(L|opt="tab",hline=[0,1,z],vline=[0,1,z],title=T)$
[9] os_md.dviout(Out)$
```

素数 p の原始根とそのべきを p で割った余り

p	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
3	2	1																				
5	2	4	3	1																		
7	3	2	6	4	5	1																
11	2	4	8	5	10	9	7	3	6	1												
13	2	4	8	3	6	12	11	9	5	10	7	1										
17	3	9	10	13	5	15	11	16	14	8	7	4	12	2	6	1						
19	2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1				
23	5	2	10	4	20	8	17	16	11	9	22	18	21	13	19	3	15	6	7	12	14	1

上の hline=[0,1,z] で先頭と最後の行の他（z は最後を表す），1 行目の終わりに横線を引いている．縦線も同様．また title=T でタイトルをつけている．以下も同様の例である．

```
[10] T="Table of $k$ satisfying $r^k\\equiv a\\pmod{p}$"$
[11] L=os_md.powprimroot(11,10|log=3);
[[11,2,1,8,4,7],[13,2,1,4,9,11,7],[17,3,14,1,5,11,7,4],[19,2,1,13,16,6,12,5,10],
...
[12] S=os_md.ltotex(L|opt="tab",hline=[0,z-1,z],vline=[0,1,2,z],title=T)$
[13] os_md.dviout(S)$
```

Table of k satisfying $r^k \equiv a \pmod{p}$

11	2	1	8	4	7									
13	2	1	4	9	11	7								
17	3	14	1	5	11	7	4							
19	2	1	13	16	6	12	5	10						
23	5	2	16	1	19	9	14	7	15					
29	2	1	5	22	12	25	18	21	9	20				
31	3	24	1	20	28	23	11	7	4	27	9			
37	2	1	26	23	32	30	11	7	35	15	21	9		
41	6	26	15	22	39	3	31	33	9	36	7	28	32	
43	3	27	1	25	35	30	32	38	19	16	41	34	7	6
p/a	r	2	3	5	7	11	13	17	19	23	29	31	37	41

次に，多くのリストをたたんで表示することを考える．
70 個の素数と原始根の組を表示する例を示そう．

```
[13] L=os_md.powprimroot(3,70);
[[3,2],[5,2],[7,3],[11,2],[13,2],[17,3],[19,2],[23,5],[29,2],[31,3],[37,2],[41,6],
....
[14] S=os_md.ltotex(L|opt="tab",width=-10,vline=[[0,2]],hline=[0,1,z],top=
[["$p$","$\\zeta$"]])$
[15] os_md.dviout(S)$
```

横 2 列で縦に 70 行の縦長の表を [14] の width=-10 によって行を 10 分割して横に並べる．vline=[[0,2]] は，2 で割った余りが 0 の列，すなわち先頭から 2 列毎の罫線を意味している，top=[["p","$\\zeta$"]] によって，1 行目に p と ζ の繰り返しを付加している．

p	ζ	p	ζ	p	ζ	p	ζ	p	ζ	p	ζ	p	ζ	p	ζ	p	ζ	p	ζ
3	2	23	5	53	2	83	2	113	3	157	5	193	5	233	3	271	6	313	10
5	2	29	2	59	2	89	3	127	3	163	2	197	2	239	7	277	5	317	2
7	3	31	3	61	2	97	5	131	2	167	5	199	3	241	7	281	3	331	3
11	2	37	2	67	2	101	2	137	3	173	2	211	2	251	6	283	3	337	10
13	2	41	6	71	7	103	5	139	2	179	2	223	3	257	3	293	2	347	2
17	3	43	3	73	5	107	2	149	2	181	2	227	2	263	5	307	5	349	2
19	2	47	5	79	3	109	6	151	6	191	19	229	6	269	2	311	17	353	3

```
[16] S=os_md.ltotex(L|opt="tab",width=16,hline=[[0,2]],vert=1);
[17] os_md.dviout(S)$
```

横 2 列で縦に 70 行の表を [16] の vert=1 で転置して横 70 列，縦 2 列に変換し，さらに width=16 によって横幅 16 列で切って折り返して並べる．

3	5	7	11	13	17	19	23	29	31	37	41	43	47	53	59
2	2	3	2	2	3	2	5	2	3	2	6	3	5	2	2
61	67	71	73	79	83	89	97	101	103	107	109	113	127	131	137
2	2	7	5	3	2	3	5	2	5	2	6	3	3	2	3
139	149	151	157	163	167	173	179	181	191	193	197	199	211	223	227
2	2	6	5	2	5	2	2	2	19	5	2	3	2	3	2
229	233	239	241	251	257	263	269	271	277	281	283	293	307	311	313
6	3	7	7	6	3	5	2	6	5	3	3	2	5	17	10
317	331	337	347	349	353										
2	3	10	2	2	3										

3.2.12.2 表や行列の作成

- 表とは，Excel の表のように，何項かのデータを縦横に並べたものと考えます．Risa/Asir では，ベクトルのリストか，リストのリストの型のものを意味することとします．
- 表を変形するときは，表を行列に変換すると便利で，様々な変形の機能が使えます．以下では，そのように変換したものも「表」と呼ぶことにします．
- **リストのリスト，あるいはベクトルのリストを行列に直す**には，lv2m() を用います．データのないところに対応する行列の成分はデフォルトでは 0 になりますが，オプション null=t で t に置き換えられます．
- 逆の変換は，m2ll() または m2ll() を用います．

```
[0] LL=[[1,2,3],[4,5]];
[[1,2,3],[4,5]]
[1] os_md.lv2m(LL);
[ 1 2 3 ]
[ 4 5 0 ]
[2] os_md.m2ll(@@);
```

```
[[1,2,3],[4,5,0]]
[3] LV=[newvect(3,[1,2,3]),newvect(2,[4,5])];
[[ 1 2 3 ],[ 4 5 ]]
[4] os_md.lv2m(LV|null="");
[ 1 2 3 ]
[ 4 5  ]
[5] os_md.m2ll(@@);
[[1,2,3],[4,5,]]
```

行列（表）から**行や列を抜き出したり入れ替えて，新しい行列（表）を作る**には mperm() を用います（第 2 引数で行を，第 3 引数で列を指定します）.

```
[6] A=os_md.mgen(3,4,a,0);
[ a00 a01 a02 a03 ]
[ a10 a11 a12 a13 ]
[ a20 a21 a22 a23 ]
[7] os_md.mperm(A,[2,0],[1,2,3]);
[ a21 a22 a23 ]
[ a01 a02 a03 ]
```

上で A から 2 行目と 0 行目をこの順に抜き出し，さらにそれの 1 列目，2 列目，3 列目を抜き出して作られる表を得ています．なお，行と列の番号は 0 から始まることに注意.
順に並んだ数字は，先頭の数字の後に並ぶ個数をリストとして指定することも可能です.

```
[8] os_md.mperm(A,[2,0],[1,[3]]);
[ a21 a22 a23 ]
[ a01 a02 a03 ]
```

また順に並んだ全ての行や列を表すには，数字 0 を用いることができます．よって以下のようにして最後の列を削除することができます.

```
[9] os_md.mperm(A,0,[0,[3]]);
[ a00 a01 a02 ]
[ a10 a11 a12 ]
[ a20 a21 a22 ]
```

2 つの行の入れ替えや 2 つの列の入れ替えは，[[m_1,m_2]] で指定します．たとえば，0 行目と 1 行目の入れ替えは

```
[10] os_md.mperm(A,[[0,1]],0);
[ a10 a11 a12 a13 ]
[ a00 a01 a02 a03 ]
[ a20 a21 a22 a23 ]
```

なお行列は**成分を直接指定して変更**できます.

```
[11] AA=os_md.dupmat(A);
[ a00 a01 a02 a03 ]
[ a10 a11 a12 a13 ]
[ a20 a21 a22 a23 ]
[12] AA[0][0]=b00$
```

```
[13] AA;
[ b00 a01 a02 a03 ]
[ a10 a11 a12 a13 ]
[ a20 a21 a22 a23 ]
```

行と列を入れ替えて**転置した表**にするには mtranspose() を用います.

```
[14] AT=os_md.mtranspose(A);
[ a00 a10 a20 ]
[ a01 a11 a21 ]
[ a02 a12 a22 ]
[ a03 a13 a23 ]
```

2 つ（またはそれ以上）の**表を結合する**には，newbmat() を用います.

```
[15] B=os_md.mgen(3,3,b,0);
[ b00 b01 b02 ]
[ b10 b11 b12 ]
[ b20 b21 b22 ]
[16] C=os_md.newbmat(1,2,[[A,B]]);
[ a00 a01 a02 a03 b00 b01 b02 ]
[ a10 a11 a12 a13 b10 b11 b12 ]
[ a20 a21 a22 a23 b20 b21 b22 ]
[17] D=os_md.newbmat(2,1,[[AT],[B]]);
[ a00 a10 a20 ]
[ a01 a11 a21 ]
[ a02 a12 a22 ]
[ a03 a13 a23 ]
[ b00 b01 b02 ]
[ b10 b11 b12 ]
[ b20 b21 b22 ]
```

横長の表を分割して縦に繋げるには，madjust() を用います．列の個数を 2 番目の引数で指定します.

```
[18] os_md.newbmat(C,3|null="");
[ a00 a01 a02 ]
[ a10 a11 a12 ]
[ a20 a21 a22 ]
[ a03 b00 b01 ]
[ a13 b10 b11 ]
[ a23 b20 b21 ]
[ b02   ]
[ b12   ]
[ b22   ]
```

縦長の表を折り返して横に並べるには，madjust() を用います．いくつ折り返すかの数の −1 倍を 2 番目の引数で指定します.

```
[19] os_md.newbmat(D,-4|null="");
```

```
[ a00 a10 a20 a02 a12 a22 b00 b01 b02 b20 b21 b22 ]
[ a01 a11 a21 a03 a13 a23 b10 b11 b12     ]
```

3.2.12.3 函数値の数表

360. ntable(f,[a,b],n|dif=1,str=[k_1,k_2,...],mult=m,title=t,top=[t_1,...])
:: 区間 $[a,b]$ を n 等分，または n=[n_1,n_2] 等分した点での x 変数の函数 f の函数値の数表を作る

n が数字のとき

- f は sin(x*@pi/180) のような通常の関数の他，リスト形式函数でもよい．
- dif=1 を指定すると，次の函数値との差の絶対値も表に加える．
- オプション str を指定すると，文字列の表となる．
  k_1 は変数の小数点以下の桁数．k_2 は値の小数点以下の桁数．
  k_3 は dif=1 で加わった値の差の桁数に対応する．k_3 の指定がないときは，k_2 と等しいとみなされる．
- mult=m を指定すると，分割して横に m 列並べた TEX 形式の表に変換される．このとき
  - title=t を指定すると，表にタイトル t がつく．
  - top=[t_1,...] を指定すると，それが表の 1 行目に入る．

n が数字のリスト [n_1,n_2] のとき, 区間 [a,b] を n_1 等分した小区間 [a_i,b_i] をさらに n_2 等分した点での函数値の表を作成する．このとき

- オプション str を指定すると，表の TEX ソースが出力される．
  k_1 は変数の小数点以下の桁数．k_2 は値の小数点以下の桁数．
  k_3 は dif=1 で加わった値の差の桁数に対応する．k_3 の指定がないときは，k_2 と等しいとみなされる．
  さらにこのとき，以下のオプションが指定可能
  - dif=1 を指定すると，次の函数値との差の各小区間での最小値と最大値を表に加える．
  - hline=h ： 横線の指定（ltotex(ℓ|opt="tab",hline=h) を参照）．デフォルトは h =[0,1,z].
  - vline=v：縦の指定（ltotex(ℓ|opt="tab",vline=v) を参照）．デフォルトは v =[0,1,z]（dif=1 を指定した場合は [0,1,z-2].
  - top=[t_1,...,t_m] を指定すると，それが表の 1 行目に入る．m は，n_2, n_2+1, n_2+2, n_2+3 のいずれか．
  - title=s：タイトルを文字列で指定．

2 番目の引数が [a,b] でなくて，[[a_1,b_1],[a_2,b_2]] のときは，2 変数函数 $f(x,y)$ の数値表になる．各行は x の値が [a_1,,b_1] を n_1 等分した値を取り，そのときの y の値は [a_2,b_2] の n_2 等分した値を取ったとき $f(x,y)$ の票になる

例えば 4 桁常用対数表は，以下の様に作成できる．

```
[0] os_md.ntable(log(x)/log(10),[1,5.5],[45,10]|str=[1,4],top=[0,1,2,3,4,5,6,7,
    8,9],hline=[0,[1,5]],vline=[0,6,z]);
  /* [1,5.5] を 45 等分し，各行はそれを 10 等分した常用対数（0.01 刻み）．
     左端は小数点以下 1 桁，函数値は小数点以下 4 桁で丸める
     横線は最初と 1 行目の後から 5 行目毎，縦線は最初と 6 列目の後と最後 */
```

4 桁常用対数表

	0	1	2	3	4	5	6	7	8	9
1.0	.0000	.0043	.0086	.0128	.0170	.0212	.0253	.0294	.0334	.0374
1.1	.0414	.0453	.0492	.0531	.0569	.0607	.0645	.0682	.0719	.0755
1.2	.0792	.0828	.0864	.0899	.0934	.0969	.1004	.1038	.1072	.1106
1.3	.1139	.1173	.1206	.1239	.1271	.1303	.1335	.1367	.1399	.1430
1.4	.1461	.1492	.1523	.1553	.1584	.1614	.1644	.1673	.1703	.1732
1.5	.1761	.1790	.1818	.1847	.1875	.1903	.1931	.1959	.1987	.2014

以下，5.4 の項まで続く．

4 桁目は線形近似で得るので，例えば 30 から 43 までの計算表は

```
[1] os_md.ntable(x*y/10,[[1,11],[30,44]],[10,14]|hline=[0,z-1,z],str=[0,0],TeX=1);
   /* x=1,2,...,10 (10 個)  y=30,31,...,43 (14 個) */
```

とすれば得られる．

1	3	3	3	3	3	4	4	4	4	4	4	4	4	4
2	6	6	6	7	7	7	7	7	8	8	8	8	8	9
3	9	9	10	10	10	11	11	11	11	12	12	12	13	13
4	12	12	13	13	14	14	14	15	15	16	16	16	17	17
5	15	16	16	17	17	18	18	19	19	20	20	21	21	22
6	18	19	19	20	20	21	22	22	23	23	24	25	25	26
7	21	22	22	23	24	25	25	26	27	27	28	29	29	30
8	24	25	26	26	27	28	29	30	30	31	32	33	34	34
9	27	28	29	30	31	32	32	33	34	35	36	37	38	39
10	30	31	32	33	34	35	36	37	38	39	40	41	42	43

各行の函数値の差分の最小と最大の数の組を加えた函数値表を作成するには，以下のようにすればよい．

```
[2] os_md.ntable(log(x)/log(10),[1,5.5],[45,10]|str=[1,4],top=[0,1,2,3,4,5,6,7,
    8,9],hline=[0,[1,5]],vline=[0,1,6,z-2],dif=1);
```

	0	1	2	3	4	5	6	7	8	9		
1.0	.0000	.0043	.0086	.0128	.0170	.0212	.0253	.0294	.0334	.0374	40	43
1.1	.0414	.0453	.0492	.0531	.0569	.0607	.0645	.0682	.0719	.0755	36	39
1.2	.0792	.0828	.0864	.0899	.0934	.0969	.1004	.1038	.1072	.1106	33	36

その他，たとえば 0° から 45° までの 10′ 刻みの三角関数表（正弦）は

```
[3] T=["$0'$","$10'$","$20'$","$30'$","$40'$","$50'$"]$
[4] os_md.ntable(sin(@pi*x/180),[0,45],[45,6]|str=[0,4],top=T,hline=[0,[1,5]],
    vline=[0,1,z-2],dif=1);
```

とすると得られる（sin を tan で置き換えると，正接の表になる）．

	0′	10′	20′	30′	40′	50′		
0	.0000	.0029	.0058	.0087	.0116	.0145	29	30
1	.0175	.0204	.0233	.0262	.0291	.0320	29	29
2	.0349	.0378	.0407	.0436	.0465	.0494	29	29
3	.0523	.0552	.0581	.0610	.0640	.0669	29	30
4	.0698	.0727	.0756	.0785	.0814	.0843	29	29
5	.0872	.0901	.0929	.0958	.0987	.1016	28	29

90° までの表を 1 ページ内に収めるには

```
[5] T=append(T,["$60'$","$70'$","$80'$","$90'$","$100'$","$110'$"])$
[6] os_md.ntable(sin(@pi*x/180),[0,90],[45,12]|str=[0,4],top=T,hline=[0,[1,5]],
    vline=[0,1,7,z-2],dif=1);
```

とすればよい.

	0′	10′	20′	30′	40′	50′	60′	70′	80′	90′	100′	110′		
0	.0000	.0029	.0058	.0087	.0116	.0145	.0175	.0204	.0233	.0262	.0291	.0320	29	30
2	.0349	.0378	.0407	.0436	.0465	.0494	.0523	.0552	.0581	.0610	.0640	.0669	29	30
4	.0698	.0727	.0756	.0785	.0814	.0843	.0872	.0901	.0929	.0958	.0987	.1016	28	29
6	.1045	.1074	.1103	.1132	.1161	.1190	.1219	.1248	.1276	.1305	.1334	.1363	28	29
8	.1392	.1421	.1449	.1478	.1507	.1536	.1564	.1593	.1622	.1650	.1679	.1708	28	29
10	.1736	.1765	.1794	.1822	.1851	.1880	.1908	.1937	.1965	.1994	.2022	.2051	28	29

3.2.12.4 点数分布表

361. distpoint(l|div=5,opt=s,title=t,size=ℓ)
:: 100 点満点の点数表のデータ l を元に点数分布などを表にする
- 10 点刻みの人数の分布を TEX の表にする
- s="data" 10 点刻みの人数の分布をリストにする
- s="graph" 10 点刻みの人数の分布を棒グラフにする
  - size=ℓ でグラフのサイズを指定できる（cf. ltotex(|opt="graph")）
- div=5 上で 10 点刻みを 5 点刻みとする
- s="average" 平均点，標準偏差，受験人数，最低点，最高点を TEX の表にする
- l は，数のリストまたは数のリストのリストとするが，そのなかに非負整数でないものがあれば，それは欠席とみなされ，s="average"でその数が示される

```
[0] M=[[74,71,47,80,66],[54,71,50,76,45],[75,70,78,x,69],[35,44,74,35,37],
[59,x,85,75,52],[92,55,70,61,45],[70,79,77,76,55]]$
[1] os_md.distpoint(M|opt="data");
[0,0,0,3,4,6,3,14,2,1]
[2] os_md.distpoint(M|opt="data",div=5);
[3] [0,0,0,0,0,0,0,3,1,3,3,3,1,2,7,7,1,1,1,0]
[4] S=os_md.distpoint(M|opt="average",title="線形代数（中間試験）");
\begin{tabular}{|cccccc|}
\multicolumn{6}{c}{線形代数（中間試験）}\\ \hline
平均点& 標準偏差& 最低点& 最高点& 受験人数& 欠席者\\
$63.7$& $15.1$& $35$& $92$& $33$& $2$\\ \hline
\end{tabular}
[5] os_md.dviout(S)$
```

[1] で M をリストのリストにしているのは，5 人ずつ目印のため．また x は欠席者を表している（アルファベットの任意の文字でよい）.

なお，Excel などに点数のデータが入っているときにそれを読み込むには，それを例えば *fname* というファイルに CSV 形式で出力する．その n 項目が点数のデータとすると

M=os_md.readcsv(*fname*|col=n,eval=n)

によって点数のデータが M にリストとして読み込まれる．
なお，上の [5] の結果は：

線形代数（中間試験）

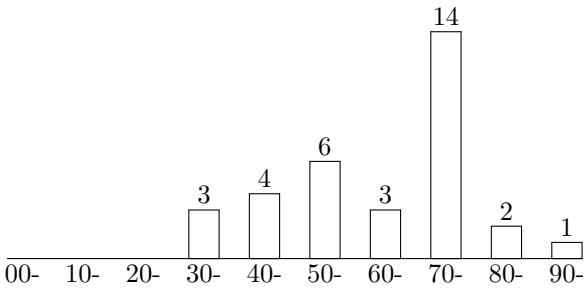
平均点	標準偏差	最低点	最高点	受験人数	欠席者
63.7	15.1	35	92	33	2

```
[6] S=os_md.distpoint(M|title="点数分布");
\begin{tabular}{|rrrrrrrrrr|}
\multicolumn{10}{c}{点数分布}\\ \hline
00--09& 10--19& 20--29& 30--39& 40--49& 50--59& 60--69& 70--79& 80--89& 90--100\\
$0$& $0$& $0$& $3$& $4$& $6$& $3$& $14$& $2$& $1$\\ \hline
\end{tabular}
[7] os_md.dviout(S)$
```

点数分布

00–09	10–19	20–29	30–39	40–49	50–59	60–69	70–79	80–89	90–100
0	0	0	3	4	6	3	14	2	1

```
[8] S=os_md.distpoint(M|opt="graph")$
[9] os_md.dviout(S)$
```



3.2.12.5 Taylor 展開

362. seriesTaylor(f,k,v|evalopt=*opt*,small=1,frac=0,dviout=n)
:: 函数 f の変数または変数のリスト v に対する k 次の項までの Taylor 展開を求める
- Taylor 展開を TEX の形に変換して返す．式が長いときは，改行が入る．
- v=x：変数 x について，原点での Taylor 展開を与える．
- v=[x,a]：変数 x について，$x = a$ での Taylor 展開を与える．ただし a も変数のときは，v=[[x,a]] としなくてはならない（次項と区別するため）．
- v=[x,y,...]：変数 $(x, y, \ldots)$ についての原点での Taylor 展開を与える．
- v =[[x,a],[y,b],...]：変数 $(x, y, \ldots)$ について，点 $(a, b, \ldots)$ を中心とする Taylor 展開を与える．ここで，たとえば $a = 0$ のときは，上の [x,a] の部分は，単に x と表記してよい．
- evalopt=[[s_1,t_1],[s_2,t_2]...]：は evalred() を使うときのオプション（cf. seriesMc()）．
- small=1：分数が現れるとき \frac でなくて \tfrac を使う．
- frac=0：有理数が現れるとき，それを（近似）実数に変換する．
  このときは，ctrl(double_output,1) を指定するのがよい．
- dviout=1：Taylor 展開の式を TEX に変換したものを表示する（元の f も表示）．
- dviout=2：Taylor 展開の部分のみを TEX に変換したものを表示する（元の f は表示せず）．
- dviout=0：デフォルトで，dviout=2 の TEX ソースを返す．

- `dviout=-1`：`dviout=2` のときのソースを返す.

まず $\sin x$ の Taylor 展開を求めてみる.

```
[0] os_md.seriesTaylor(sin(x),5,x);
x-\frac{1}6x^3+\frac{1}{120}x^5
[1] os_md.seriesTaylor(sin(x),5,x|small=1,dviout=-1);
\begin{align}\begin{split}
 \sin(x)&=x-\tfrac{1}6x^3+\tfrac{1}{120}x^5+\cdots
\end{split}\end{align}

[2] os_md.seriesTaylor(sin(x),7,x|small=1)$
```

$$\sin(x) = x - \tfrac{1}{6}x^3 + \tfrac{1}{120}x^5 - \tfrac{1}{5040}x^7 + \cdots$$

ここでは $\sin x$ の 7 次の項までの Taylor 展開を求め，係数は小さなサイズの分数で表示している.
また，展開の係数を近似小数で表した表示を得るには，以下のようにすればよい.

```
[3] os_md.seriesTaylor(sin(x),5,x|frac=0,dviout=-1);
\begin{align}\begin{split}
 \sin(x)&=x- 0.166667x^3+ 0.00833333x^5+\cdots
\end{split}\end{align}

[4] os_md.dviout(@@)$
```

$$\sin(x) = x - 0.166667x^3 + 0.00833333x^5 + \cdots$$

```
[5] os_md.seriesTaylor(sin(x),5,[x,@pi/2]|dviout=-1,small=1,evalopt
 =[[sin(@pi/2),1],[cos(@pi/2),0]],dviout=-1);
\begin{align}\begin{split}
 \sin(x)&=1-\tfrac{1}2(x-\tfrac{1}2 \pi)^2+\tfrac{1}{24}(x-\tfrac{1}2 \pi)^4
 +\cdots
\end{split}\end{align}

[6] os_md.dviout(@@)$
```

$$\sin(x) = 1 - \tfrac{1}{2}(x - \tfrac{1}{2}\pi)^2 + \tfrac{1}{24}(x - \tfrac{1}{2}\pi)^4 + \cdots$$

`[5]` において $\sin(x)$ の $x = \frac{\pi}{2}$ での Taylor 展開を 5 次の項まで求めているが, $\sin(\frac{\pi}{2}) = 1$, $\cos(\frac{\pi}{2}) = 0$ という情報を与えている．これを与えないと，係数は近似小数になる（$\sin(0) = 0$, $\cos(0) = 1$ などの情報は不要（cf. `evalred()`））.

```
[7] os_md.seriesTaylor((x+1)^(1/2),10,x|frac=0,dviout=1);
```

$$\sqrt{x+1} = 1 + 0.5x - 0.125x^2 + 0.0625x^3 - 0.0390625x^4 + 0.0273438x^5 - 0.0205078x^6 + 0.0161133x^7 - 0.013092x^8 + 0.01091x^9 - 0.00927353x^{10} + \cdots$$

$\sqrt{x+1}$ の $x = 0$ での Taylor 展開を求めたが，上のように，長い式は改行して表示される（cf. `TeXLim`）.
最後に 2 変数函数の原点での Taylor 展開の例を挙げる.

```
[8] os_md.seriesTaylor(exp(sin((x-y)/(x^2+2*y^2+2))),3,[x,y]|small=1,dvioout=1)$
```

$$\exp(\sin(\frac{x-y}{x^2+2y^2+2})) = 1+\frac{1}{2}x-\frac{1}{2}y+\frac{1}{8}x^2-\frac{1}{4}xy+\frac{1}{8}y^2-\frac{1}{4}x^3+\frac{1}{4}x^2y-\frac{1}{2}xy^2+\frac{1}{2}y^3+\cdots$$

3.2.13 Environments

363. Canvas
:: Risa/Asir の描画キャンバスのデフォルトサイズ
execdraw() などで Risa/Asir の描画キャンバスを開くときのデフォルトサイズ
- 横と縦の pixel サイズのリストで，デフォルトは [400,400] となっている．
- dvioout0([800,400]|opt="Canvas") のようにして変更可能
- 初期化ファイル.muldif でデフォルト値の設定変更可能

364. AMSTeX
:: この値が 1 は $\mathcal{A}\mathcal{M}\mathcal{S}$LaTeX を意味する
os_muldif.rr では，AMSTeX=1 がデフォルトで，これが想定されている．それ以外では，函数によっては作成される TeX のソースファイルが正しくないことがある．

```
[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
```

365. TeXEq
:: デフォルトの LaTeX の数式環境の指定（dvioout0(3) で値が分かる）
TeXEq の値は，1, 2, 3, 4, 5, 6, 7 のいずれかで，それは dvioout() のオプションパラメータの eq= の

値に対応する．
AMSTeX=1 のときのデフォルトは 5 で，AMSTeX=0 のときは TeXEq=1 とみなされる．
dviout0(*n*|opt="TexEq") で変更できる．

366. TeXLim
:: LaTeX で長い数式を行分割する際の 1 行の許容最大横幅文字数のデフォルト値
- .muldif で設定でき，dviout0("?") で値が分かる．
- texlim(1,*n*) または dviout0(*n*|opt="TexLim") で変更できる（デフォルトは 80）．

367. TikZ
:: グラフ表示に Xy-pic を使うか Ti*k*Z を使うかを指定
TikZ=1 のときは Ti*k*Z を，TikZ=0 のときは Xy-pic を使う．
切り替えは dviout0(6), dviout0(7) または dviout0(1|opt="TikZ"), dviout0(0|opt="TikZ").

368. XYPrec
:: グラフ表示の時の座標の小数点以下の丸め桁数
切り替えは dviout0(*n*|opt="XYPrec").

369. XYcm
:: Xy-pic での単位を cm で表して，Ti*k*Z に合わせる
切り替えは dviout0(0|opt="XYcm"), dviout0(1|opt="XYcm").

370. XYLim
:: Xy-pic や Ti*k*Z での曲線描画で順に指定する点での改行の間隔
切り替えは dviout0(0|opt="XYLim"), dviout0(1|opt="XYLim").

371. DVIOUTH
:: myhelp() で指定した函数の解説を示すためのプログラムの指定
- .muldif で設定でき，dviout0("?") で設定が分かる．
- デフォルトは Windows 環境の場合の dviout にパスが通っているときの設定で
  ```
  start dviout -2 -hyper:0x90 "%ASIRROOT%\help\os_muldif.dvi" #%LABEL%
  ```
  - これは 2 番目の dviout に，get_rootdir()\help\os_muldif.dvi のラベル %LABEL%（ここは myhelp() がその引数を使って置き換える）がついた箇所を表示させることを意味します．
  - 上において，dviout にパスが通っていないときは，それをフルパス名にする．
  - -hyper:0x90 はホットスポットを青の文字で示すことを意味します（ox_muldif.pdf と同じ表示となる設定）．指定しないと，さらにアンダーラインが引かれます．
    dviout の Option → Setup parameters... → HyperTeX → Color の項を変更して OK ボタンを押すとこの設定が変えられる．Option → Non-default Parameters で表示される hyper= の値を上の -hyper: に設定すれば，その設定でホットスポットが表示されるようになる．
- 函数名 *fn* の説明は，os_muldif.dvi および os_muldif.pdf の中のラベル r:*fn* がつけられた場所にある．ただし *fn* にアンダースコアー "_" が含まれるときは，_ を削ってラベルになっている．
- %ASIRROOT% が get_rootdir() に，%LABEL% が函数名に対応する（上で述べた）ラベルに置き換えられて shell() の引数として myhelp() で使われ，函数に対応するヘルプが表示される．

372. DIROUT
:: 数式を LaTeX に変換したソースが格納されるディレクトリ（書き込み可能なことが要請される）

373. DVIOUTA
:: LaTeX の $\mathcal{AMS}$TeX 環境で risaout.tex を変換して表示するプログラムのパス名

374. DVIOUTB
:: LaTeX の $\mathcal{AMS}$TeX 環境で risaout10.tex （または risaout10.tex）を変換して表示するプログラムのパス名で，DVIOUTA と交換できる（cf. dviout0(), risatex.bat）．

375. DVIOUTL
:: LaTeX 環境で riasout0.tex を変換して表示するプログラムのパス名
.muldif で設定でき，設定された値は dviout0("?") で分かる．
Windows 環境でのデフォルトは

```
[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]
TeXLim = 80
TeXEq  = 5
AMSTeX = 1
TikZ   = 0
XYPrec = 3
XYcm   = 0
XYLim  = 4
```

それ以外でのデフォルトは

```
[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"
Canvas = [400,400]
TeXLim = 80
TeXEq  = 5
AMSTeX = 1
TikZ   = 0
XYPrec = 3
XYcm   = 0
XYLim  = 4
```

となっている.
なお，%TikZ% は Ti*k*Z の値，%ASIRROOT% は get_rootdir() で得られる Risa/Asir のインストールディレクトリ，%HOME% は環境変数 HOME の値を意味するが，Windows のときの後者のデフォルトは %ASIRROOT% に等しい.
%ASIRROOT% や %HOME% に空白が含まれていると正しく動作しないので，次の .muldif によって " で二重に囲んで

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

のように設定し直して下さい.

376. .muldif
:: os_muldif.rr をロードしたときに読み込まれるファイル

- ファイル.muldif を %HOME% に入れておくと，自動的に読み込まれる．そのほか，順に %HOME%/asir，%ASIRROOT%，%ASIRROOT%/bin, %ASIRROOT%/lib-asir-contrib が探される．なお %ASIRROOT% は get_rootdir() と同じ．また %HOME% は環境変数 HOME の値であるが，Windows では通常 %ASIRROOT% と等しい．
- TeXLim, TeXEq, DIROUT, DVIOUTA, DVIOUTB, DVIOUTL, DVIOUTH, TikZ, XYPrec, XYcm, XYLim, Canvas のデフォルト設定の変更が可能．
- 設定状態は dvioutO(3) で表示される．

たとえば，.muldif の例は

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

なお，C の文法に準じて，文字列中の文字 \ や " は \\ や \" と書く（そのように書かなくても多くの場合は大丈夫であろう）．さらに文字列の始まりと終わりは " で示す（それらは，行で最初にある " と最後にある " とみなしている）．

377. risatex.bat

:: os_muldif.rr が出力する LaTeX のソースファイルを変換して表示するプログラム

- os_muldif.rr が TeX を使って結果や数式を綺麗に表示する際には，DIROUT 内の LaTeX ファイル out.tex（または out0.tex）に LaTeX のソースを書き出しますが，それらを risaout.tex（または risaout1.tex，risaout0.tex）から読み込んで dvi ファイルや pdf に変換して表示するデフォルトのプログラムです（後者は risaout0.bat）．
- DVIOUTA（または DVIOUTB，DVIOUTL）で指定されます．

TeX を使って数式や結果を表示するための設定は以下の通りです．

- Windows 環境におけるデフォルト設定で get_rootdir()\bin に入れる risatex.bat は，例えば以下のような内容です（cf. DVIOUTA）：

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

  - c:\Program Files\asir の部分は，デフォルトでは get_rootdir() となります．cd の後には，DIROUT で示されるディレクトリを示しますが，数式を LaTeX に直した out.tex を書き込むディレクトリであって書き込み可能である必要があるので，デフォルトを変更するのがよいでしょう．
  - また dvioutt にパスが通っていないときは，上の 3 行目の dvioutt をフルパス名に変更します．
  - 最後の行のパラメータ −1 は，1 番目の dvioutt に対する指示を意味し，それが起動していないと起動することも意味します．
  - 最後の 1000 は表示するページを表し，十分大きな数にしておけば最終ページを表示することを意味します．表示する数式が最後に追加されていくので，このようにしています．

  DIROUT に置かれる risaout.tex は以下のような内容のファイルです：

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

  以下のように変更すると，より長い式が表示できます．またグラフ表示などで XY-pic または TikZ を使うことがあるので，それを読み込んでいます．この XY-pic はインタネットなどから情報を得て，インストールしてください．

```
\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}
```

dvi ファイルは使わず，dvipdfmx や pdfTeX によって pdf ファイルを生成してそれのみを使う場合は，上の \usepackage[all]{xypic} を\usepackage[pdf,all]{xypic} に変えた方が，良質の pdf ファイルが得られます．
それを（dviout0(4) での切り替えに対応の）DVIOUTB に設定するには，たとえば以下の risaoutpdf.tex を c:\Program Files\asir\tex におき

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

さらに risatex1.bat を

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

とし，risatex1.bat はフルパスで DVIOUTB に設定します．
pdf ファイルを更新表示可能な SumatraPDF.exe で表示するには，上の最終行の risaoutpdf.pdf をたとえば

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

などのように書き換えます．

Xy-pic でなくてより高機能の Ti*k*Z（ただし速度はより遅い）を使う場合は，risaoutpdf.tex をたとえば以下のように変更します．

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

```
\end{document}
```
また上に合わせたグラフィック出力にするには，初期設定ファイル .muldif に
```
TikZ=1
```
という一行を加えます．これは dviout0(6), dviout0(7) で起動後に変更もできます．

XY-pic と TikZ の両方に自動対応するには，たとえば以下のようにします．
DVIOUTB などの中の %TiKZ% には TikZ の値に置き換えられるのでデフォルトの DVIOUTB の場合には，XY-pic 対応と TikZ 対応の risaoutpdf.tex をそれぞれ risaoutpdf0.tex, risaoutpdf1.tex という異なったファイル名にし，バッチファイル risatex10.bat, risatex11.bat をそれぞれ
```
cd "c:\Program Files\asir\tex"
platex risaoutpdf0
dvipdfmx risaoutpdf0
"c:\Program Files\SumatraPDF\SumatraPDF.exe" -reuse-instance risaoutpdf0.pdf
```
および
```
cd "c:\Program Files\asir\tex"
platex risaoutpdf1
dvipdfmx risaoutpdf1
"c:\Program Files\SumatraPDF\SumatraPDF.exe" -reuse-instance risaoutpdf1.pdf
```
とします．
XY-pic と TikZ を同時に対応することも出来て，例えば以下のようにします．
```
\documentclass[dvipdfmx,a4paper]{jsarticle}
\usepackage{amsmath,amssymb,amsthm,amscd,mathrsfs}
\usepackage{tikz}
%\usetikzlibrary{patterns,shapes} % 必要に応じて
\usepackage[pdf,all]{xy}
\begin{document}
\thispagestyle{empty}
\input{out}
\end{document}
```

- なお，AMSTeX=0 のときは，risatex0.bat と risaout0.tex が使われ，それぞれ以下のようです．
```
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}
```
- Unix や Mac の場合は，risatex.bat でなくて risatex.sh がデフォルトのファイル名で，例えば以下のようであって，それに実行権限を付加しておきます（chmod 755）．
```
#!/bin/sh
cd ${HOME}/asir/tex
platex -src=cr,display,hbox,math,par risaout
dvipdfmx risaout
```

```
evince risaout.pdf &
```
あるいは，環境によっては
```
#!/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
```
のようにフルパスで実行ファイルを指定します．このとき risaout.tex は，たとえば以下のようになります．
```
\documentclass[a4paper]{amsart}
\usepackage{amsmath,amssymb,amsfonts}
\usepackage[pdf,all]{xy}
\pagestyle{empty}
\begin{document}
\thispagestyle{empty}
\input{out}
\end{document}
```
なお，risatex0.bat や risatex1.bat も，デフォルトのファイル名は risatex0.sh や risatex1.sh となります．

- risaout.tex および risaout0.tex が DIROUT に存在しないとき（かつ DIROUT が書き込み可能なとき）は，必要に応じてデフォルトのものが自動的に作成されます．
- get_rootdir()\lib-asir-contrib にある noro_print.rr を以下のように変更し，$\mathcal{A}MS$LaTeX にも対応するようにします．
  なお，この変更は配布版に取り入れられています．

```
*** noro_print_org.rr Wed May 24 17:54:34 2006
--- noro_print.rr Mon Dec 14 00:26:32 2009
***************
*** 4,7 ****
--- 4,8 ----

  extern Taka_png_form_res$
+ extern AMSTeX$
  Taka_png_form_res=150$

*** 78,84 ****

  def taka_tex_form_matrix(A,Tb) {
    N = size(A)[0];
    M = size(A)[1];
!   write_to_tb("\\pmatrix{\n",Tb);
    for (I=0; I<N; I++) {
      for (J=0; J<M; J++) {
--- 79,86 ----

  def taka_tex_form_matrix(A,Tb) {
```

```
+   extern AMSTeX;
    N = size(A)[0];
    M = size(A)[1];
!   write_to_tb(AMSTeX?"\\begin{pmatrix}\n":"\\pmatrix{\n",Tb);
    for (I=0; I<N; I++) {
      for (J=0; J<M; J++) {
***************
*** 86,102 ****
        if (J != M-1) write_to_tb("& ",Tb);
      }
!     write_to_tb(" \\cr\n",Tb);
    }
!   write_to_tb("}\n",Tb);
  }

  def taka_tex_form_vector(A,Tb) {
    N = size(A)[0];
!   write_to_tb("\\pmatrix{\n",Tb);
    for (I=0; I<N; I++) {
      taka_tex_form(A[I],Tb);
      if (I != N-1) write_to_tb("& ",Tb);
    }
!   write_to_tb("\\cr}\n",Tb);
  }

--- 88,105 ----
        if (J != M-1) write_to_tb("& ",Tb);
      }
!     write_to_tb(AMSTeX?((I==N-1)?"\n":" \\\\\n"):" \\cr\n",Tb);
    }
!   write_to_tb(AMSTeX?"\\end{pmatrix}\n":"}\n",Tb);
  }

  def taka_tex_form_vector(A,Tb) {
+   extern AMSTeX;
    N = size(A)[0];
!   write_to_tb(AMSTeX?"\\begin{pmatrix}\n":"\\pmatrix{\n",Tb);
    for (I=0; I<N; I++) {
      taka_tex_form(A[I],Tb);
      if (I != N-1) write_to_tb("& ",Tb);
    }
!   write_to_tb(AMSTeX?"\n\\end{pmatrix}\n":"\\cr}\n",Tb);
  }
```

3.2.14 補足

3.2.14.1 行列の入力

行列の入力が容易になるよう様々な方法が提供されている.

- 行列 $\begin{pmatrix} 2 & 1 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & -1 \end{pmatrix}$ の入力は
  - `mat([2,1,0],[0,2,0],[0,0,-1])`：各行をリストで列の数だけ並べる.
    `mat([2,1,0],[0,2],[0,0,-1])` でもよい.
    最初の行の成分の個数で列の数が決まるので，最初の行のみ後方の 0 は省略できない.
  - `s2m("210,020,0^2-1")`：行成分を並べて行を“`,`”で区切る.
    `s2m("21,02,0^2-1")` でもよい.
    成分があまり大きくない整数や有理数のとき可能で，入力が短いので便利（詳しくは，`s2m()`）.
    行成分の最大個数で列のサイズが決まり，それに足らない成分には 0 が入る.
    たとえば数 0 が 4 個続くときは，`0^4` と書いてもよい.
  - `s2m([[2,1,0],[0,2,0],[0,0,-1]])`：`mat()` とほぼ同じ，全体を `[ ]` で囲う.
    `s2m([[2,1],[0,2],[0,0,-1]])` と省略できる.
  - `s2m("[2 1 0][0 2 0][0 0 -1]")`：`Risa/Asir` の行列の画面出力を文字列として指定.
    `s2m("[2 1][0 2][0 0 -1]")` と省略できる.

  などのいずれもが可能である.
  整数成分の 2 が不定元 `a` のときは，上の 2 番目方法のみ不可（`a` は整数 10 と解釈される）.

- 対角行列
  $\begin{pmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{pmatrix}$ ：`diagm(3,[a,b,c])`（対角成分を与える）
  $\begin{pmatrix} \lambda & 0 & 0 \\ 0 & \lambda & 0 \\ 0 & 0 & \lambda \end{pmatrix}$ ：`diagm(3,[lambda])`（スカラー行列）
  $\begin{pmatrix} a_1 & 0 & 0 \\ 0 & a_2 & 0 \\ 0 & 0 & a_3 \end{pmatrix}$ ：`diagm(3,a)`（対角成分の自動添え字）

- 一般行列
  $\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$ ：`mgen(3,3,a,1)`　（二重添え字の一般行列）
  $\begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix}$ ：`mgen(3,3,a,0)`　（二重添え字の一般行列）
  $\begin{pmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{pmatrix}$ ：`mgen(3,3,[a,b,c],1)`　（一重添え字の一般行列）
  $\begin{pmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{pmatrix}$ ：`mtanspose(mgen(3,3,[a,b,c],1))`（一重添え字の一般行列）
  $\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{12} & a_{22} & a_{23} \\ a_{13} & a_{23} & a_{33} \end{pmatrix}$ ：`mgen(3,"symmetric",a,1)`　（二重添え字の一般対称行列）

$\begin{pmatrix} a_1 & a_2 & a_3 \\ a_2 & b_2 & b_3 \\ a_3 & b_3 & c_3 \end{pmatrix}$: `mgen(3,"symmetric",[a,b,c],1)` （一重添え字の一般対称行列）

$\begin{pmatrix} 0 & a_{12} & a_{13} \\ -a_{12} & 0 & a_{23} \\ -a_{13} & -a_{23} & 0 \end{pmatrix}$: `mgen(3,"skew",a,1)`： （二重添え字の一般歪対称行列）

$\begin{pmatrix} 0 & a & 0 \\ 0 & 0 & a \\ 0 & 0 & 0 \end{pmatrix}$: `mgen(3,"highdiag",[a],1)` （対角成分の一つ上）

$\begin{pmatrix} \lambda & 1 & 0 \\ 0 & \lambda & 1 \\ 0 & 0 & \lambda \end{pmatrix}$: `diagm(3,[lambda])+mgen(3,"highdiag",[1],1)`

$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$: `mgen(3,"perm",[3,1,2],1);` （置換行列）

より詳しくは，`mgen()` を参照.

- 行列をつなげる

$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $B = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$ とすると

$\begin{pmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \end{pmatrix}$: `newbmat(1,2,[[A,B]])` （横につなげる）

$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 2 & 0 \\ 0 & 2 \end{pmatrix}$: `newbmat(2,1,[[A],[B]])` （縦につなげる）

$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix}$: `newbmat(2,2,[[A],[0,B]])` （対角につなげる）

$\begin{pmatrix} 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \\ 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \end{pmatrix}$: `newbmat(2,2,[[A,B],[B,A]])`

`newbmat(`m`,`n`,[[`A_{11}`,`A_{12}`,...],[`A_{21}`,`A_{22}`,...],...])` は，$m \times n$ のブロック行列を作る函数である.

- 行列を抜き出す

$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix}$ とする．ここでは，行や列の番号は 0 から始まると考える.

$\begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{20} & a_{21} & a_{22} \end{pmatrix}$: `mperm(A,[0,2],[0,1,2])` （0 行目と 2 行目，0 列目と 1 列目と 2 列目をこの順に抽出）

$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{20} & a_{21} & a_{22} & a_{23} \end{pmatrix}$: `mperm(A,[0,2],0)` （0 行目と 2 行目をこの順に抽出）

$\begin{pmatrix} a_{02} & a_{00} \\ a_{12} & a_{10} \\ a_{22} & a_{20} \\ a_{32} & a_{30} \end{pmatrix}$: `mperm(A,0,[2,0])` （2 列目と 0 列目をこの順に抽出）

$\begin{pmatrix} a_{00} & a_{02} \\ a_{20} & a_{22} \end{pmatrix}$ ：`mperm(A,[0,2],1)` （0 行目と 2 行目から対角行列を作る）

$\begin{pmatrix} a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \\ a_{00} & a_{01} & a_{02} & a_{03} \end{pmatrix}$ ：`mperm(A,[1,2,3,0],0)` （0 行目を最後の行に繰り下げる）

2 番目は行の指定を，3 番目は列の指定をリストで表す.
ただし 0 は変えないこと, 3 番目の引数 1 は 2 番目の引数に等しいことを表す.
より詳しくは mperm() を参照.

- 行列の列数の調整：madjust() を参照

3.2.14.2 平面図形

P, Q, R, S, T は座標，s, t は実数とする（平面座標は，$Q = [3, 4]$ のようにリストで与える）.

交点や接点を求める

- $\overrightarrow{OX} = \overrightarrow{OP} + \overrightarrow{OQ}$ となる X を返す（O は原点）：
  ladd(P,Q,1)
- $\overrightarrow{OX} = \overrightarrow{OP} - \overrightarrow{OQ}$ となる X を返す：
  ladd(P,Q,-1)
- $\overrightarrow{OX} = \overrightarrow{OP} + t \cdot \overrightarrow{OQ}$ となる X を返す：
  ladd(P,Q,t)
- 直線 PQ と直線 RS との交点を返す （並行なら 0 を，一致するなら 2 を返す）：
  ptcommon([P,Q],[R,S])
- 線分 PQ と線分 RS との交点を返す：
  ptcommon([P,Q],[R,S]|in=1) （交わらないときは 0 を返す）
- 点 R から直線 PQ に下ろした垂線の足を返す：
  ptcommon([P,Q],[R,0])
- 点 R から線分 PQ に下ろした垂線の足を返す（線分 PQ 上に垂線の足がないときは 0 を返す）：
  ptcommon([P,Q],[R,0]|in=1)
- 直線 PQ の垂直二等分線と直線 RS の垂直二等分線の交点を返す：
  ptcommon([P,Q],[R,0]|in=-1)
- 直線 PQ の垂直二等分線と直線 RS の交点を返す：
  ptcommon([P,Q],[R,0]|in=-2)
- 直線 PQ の垂直二等分線と線分 RS の交点を返す （線分 RS と交わらないときは 0 を返す）：
  ptcommon([P,Q],[R,0]|in=-3)
- $\overrightarrow{PQ}$ を s 倍して θ 回転したベクトル $\mathbf{v}$ とするとき，$\overrightarrow{OX} = \overrightarrow{OQ} + \mathbf{v}$ となる X を返す：
  ptcommon([P,Q],[s,θ])
- 線分 PQ の $s : t$ の内分点を返す：
  ptcommon([P,Q],[s,t]|in=1)
- 線分 PQ の $s : t$ の外分点を返す：
  ptcommon([P,Q],[s,$-t$]|in=1)
- 直線 PQ と中心が R で半径が r の円との交点を返す（交点がないときは 0 を返す）：
  ptcommon([P,Q],[R,r])
- 線分 PQ と中心が R で半径が r の円との交点を返す（交点がないときは 0 を返す）：
  ptcommon([P,Q],[R,r]|in=1)

- 中心が P で半径 s の円と中心が R で半径が r の円との交点（ないときは 0）を返す：
  `ptcommon([P,s],[R,r]|in=1)`
- 中心が P で半径 s の円に点 R から引いた接線の接点を返す：
  `ptcommon([P,s],[R,0]|in=1)`
- 2 つの Bézier 曲線 ℓ_1 と ℓ_2 の交点を返す：
  `ptcombezier(`ℓ_1`,`ℓ_2`,`m`)`
- 2 つの複合 Bézier 曲線 b_1 と b_2 の交点を返す：
  `ptcombz(`ℓ_1`,`ℓ_2`,`m`)`
- 直線 PQ の $\{(x,y) \mid x_{min} \le x \le x_{max},\ y_{min} \le y \le y_{max}\}$ の部分の両端を返す：
  `lninbox([P,Q],[[`x_{min}`,`x_{max}`],[`y_{min}`,`y_{max}`]])`
- 線分 PQ の $\{(x,y) \mid x_{min} \le x \le x_{max},\ y_{min} \le y \le y_{max}\}$ の部分の両端を返す：
  `lninbox([P,Q],[[`x_{min}`,`x_{max}`],[`y_{min}`,`y_{max}`]]|in=1)`

条件を満たす点のリストを求める

- 点 $P, Q, R, \ldots$ の $\{(x,y) \mid x_{min} \le x \le x_{max},\ y_{min} \le y \le y_{max}\}$ 外のものを数 0 に変える：
  `ptwindow([P,Q,R,...],[`x_{min}`,`x_{max}`],[`y_{min}`,`y_{max}`])`
- 点 $P, Q, R, \ldots$ にアフィン変換を施す：
  V を中心に θ 回転して行列（またはスカラー）M を掛け，さらに W 平行移動する．
  座標でなくて数の成分があれば，それは変換しない（`xybezier()` の引数など）．
  `ptaffine(`M`,[P,Q,R,...]|org=`V`,arg=`θ`,deg=`θ`,shift=`W`)`
- 半径 r で中心 P の円に内接する（始点の偏角が θ の）正 n 多角形の頂点の平面座標を返す：
  `ptpolygon(`n`,`r`|org=`P`,arg=`θ`,deg=`θ`)`
- $\overrightarrow{OX} = \overrightarrow{OS} + k \cdot \overrightarrow{OQ} + \ell \cdot \overrightarrow{OR}$ （$0 \le k < m$, $0 \le \ell < n$）の格子点 X の座標のリストを返す：
  $f_1(X[0], X[1]) \ge 0$, $f_2(X[0], X[1]) \ge 0, \ldots$ を満たさない点は除く．
  `ptlattice(`m`,`n`,`P`,`Q`|org=`S`,scale=`t`,cond=[`f_1`,`f_2`,...])`

長さや角度を求める

- $\overrightarrow{OP}$ の長さを返す：
  `dnorm(`P`)`
- $\overrightarrow{PQ}$ の長さを返す：
  `dnorm([`P`,`Q`])`
- $\overrightarrow{OP}$ と $\overrightarrow{OQ}$ の内積を返す：
  `dvprod(`P`,`Q`)`
- $\overrightarrow{PQ}$ と $\overrightarrow{RS}$ のなす角 θ（$-\pi < \theta \le \pi$）をラジアンで返す：
  `ptcommon([P,Q],[R,S]|in=2)`
- $\overrightarrow{PQ}$ と $\overrightarrow{RS}$ のなす角 θ（$-180 < \theta \le 180$）を度で返す：
  `ptcommon([P,Q],[R,S]|in=3)`
- $\angle POQ$ の余弦を返す（$P = O$ または $Q = O$ のときは 1 を返す）：
  `dvangle(`P`,`Q`)`
- $\angle PQR$ の余弦を返す（$P = Q$ または $Q = R$ のときは 1 を返す）：
  `dvangle([`P`,`Q`,`R`],0)`

基本的図形の描画

以下はデフォルトでは TEX のソースを返すが，オプション `proc=1` で描画実行形式を返し，`dviout=1` で画面表示する．

- 点 P, Q, ... を**折れ線**で順に繋ぐ：
  xylines([P,Q,...]|opt=s)
  TikZ の場合は，s に以下のような指定が可能（重複指定は "red,dotted" のように指定）.
  線の太さ指定 "thick", "thin", "width=2pt",...
  線種指定 "dotted", "densely dotted", "loosely dotted", "dashed",...
  二重線 "double", "double distance=2pt",...
  色指定 "red", "blue", "lightgray", "green!30!white",...
- 点 P, Q, ... を順に折れ線で繋ぎ，最後の点と最初の点も繋ぐ：
  xylines([P,Q,...]|close=1,opt=s)
  TikZ の場合は，さらに s に以下のような塗りつぶし指定も可能
  塗りつぶしの色 "fill=red","fill=lightgray",...
  パターン塗りつぶし "pattern=northeastline", "pattern=grid", "pattern=dots",...
  パターンの色 "pattern color=red",...
  塗りつぶしの透明度 "opacity=.5"
  塗りつぶし部分 "even odd rule"
  塗りつぶしのみで境界線を描かないときは，opt=[s,"fill"] と指定する.
- 点 P, Q, ... を順に通る**滑らかな曲線**：
  xylines([P,Q,...]|curve=1,opt=s)
- 点 P, Q, ... を順に通る**滑らかな閉曲線**：
  xylines([P,Q,...]|curve=1,close=1,opt=s)
- $y=f(x)$ $(x_1 \le x \le x_2)$ で定まる**函数のグラフ**の $y_1 \le y \le y_2$ の部分：
  xygraph(f,n,[x_1,x_2],0,[y_1,y_2]|opt=s,org=P,scale=r,prec=v)
- $\big(f_1(t), f_2(t)\big)$ $(t_1 \le t \le t_2)$ で定まる**平面曲線**の $x_1 \le x \le x_2$, $y_1 \le y \le y_2$ の部分：
  xygraph([f_1,f_2],n,[t,t_1,t_2],[x_1,x_2],[y_1,y_2]|opt=s,org=P,scale=r,prec=v)
- 点 P, Q を対角の頂点とする**水平な長方形**：
  xybox([P,Q])
- 点 P, Q を対角の頂点とする**水平な長方形**の TikZ での色付けと塗りつぶし：
  xybox([P,Q]|color=s,fill=t)
- 点 P, Q を対角の頂点とし，R をもう一つの頂点とする**平行四辺形**：
  xybox([P,Q,R])
- P を中心とする半径 r の円に内接し，一つの頂点の偏角が θ となる**正 n 角形**：
  xylines(ptpolygon(n,r|org=P,arg=θ))
- 点 P を中心とする半径 r の**円**：
  xyang(r,P,0,0)
- 中心 P，半径 r の円の偏角が $[\alpha,\beta]$ の部分の**円弧**：
  xyang(r,P,α,β)
- 中心 P，半径 r の円の偏角が $[\alpha,\beta]$ の部分の**扇形**：
  xyang(r,P,α,β|ar=1)
- 中心が P で基準軸の半径が r と qr の**楕円**：
  xyoval(P,r,q)
- 中心が P で基準軸の半径が r と qr で，基準軸の偏角が γ の**楕円**：
  xyoval(P,r,q|ang=[0,0,γ],deg=[0,0,γ])
- 中心が点 P で基準軸の偏角が γ，半径が r と qr で，偏角が $[\alpha,\beta]$ の部分の**楕円弧**：
  xyoval(P,r,q|ang=[α,β,γ],deg=[α,β,γ])
- 点 P が中心，基準軸の偏角が γ，半径が r と qr で，偏角が $[\alpha,\beta]$ の部分の**楕円扇形**：
  xyoval(P,r,q|ar=1,ang=[0,0,γ],deg=[0,0,γ])

- 格子点を（$f_j \geq 0$ を満たす）$\overrightarrow{OS}+k\cdot\overrightarrow{OQ}+\ell\cdot\overrightarrow{OR}$（$0\leq k<m$, $0\leq \ell<n$）とする**格子**：
  `xylines(ptlattice(`m`,`n`,`P`,`Q`|line=1,org=`S`,scale=`t`,cond=[`f_1`,`f_2`,...]))`
- $\angle QPR$ に半径 r の**角度記号**（$\overrightarrow{PQ}$ から反時計回りに $\overrightarrow{PR}$ まで）：
  `xyang(`r`,`P`,`Q`,`R`)`
- $\overrightarrow{PQ}$ の始点 P に一辺 r で反時計回りに**直角記号**を描く：
  `xyang(`r`,`P`,`Q`,1)`
- $\overrightarrow{PQ}$ の始点 P に一辺 r で時計回りに**直角記号**を描く：
  `xyang(`r`,`P`,`Q`,-1)`
- $\overrightarrow{QP}$ の終点 P に種々の矢印記号の**矢先**を長さ r で描く（$-4\leq t\leq -2,\ t=0,\ 2\leq t\leq 8$）:
  $t=-4(135°),\ -3(120°),\ -2(150°),\ 0(90°),\ 2,5(45°),\ 3,6(30°),\ 4(60°),\ 7(22.5°),\ 8(15°)$
  $t\geq 5$ のときは曲線
  `xyang(`r`,`P`,`Q`,`t`)`
- $\overrightarrow{QP}$ の終点 P に種々の**矢印**を長さ r で描く（$-4\leq t\leq -2,\ t=0,\ 2\leq t\leq 8$）:
  `xyang(`r`,`P`,`Q`,`t`|ar=1)`

3.2.14.3　リスト形式函数

数値に対して値を取る函数には，`dsin(`x`)` のような函数と `sin(`x`)` のような函数がある．

前者の x は `dsin(0.1)` のように実数のみが可能で不定元の `x` を用いて `dsin(x)` とするとエラーとなる．

後者は不定元 `x` を用いた `sin(x)` という形が可能で，たとえばそれの導関数は `diff(sin(x),x)` によって `cos(x)` が得られる．これらの函数を代入した `exp(sin(x)+y)` のような函数を定義することも可能である．

一方，`subst(sin(x),x,0.1)` としても $\sin 0.1$ の値が得られるのでなく不定元 `sin(0.1)` が得られ，実際の $\sin 0.1$ の値を得るには `eval(sin(0.1))` としなければならない．ところが `sin(0.1)` や `sin(0.2)` は異なる不定元として `Risa/Asir` のメモリー・エリアに登録されたままになるので，いろいろな数値を代入することを何度も行うと大量の不定元が登録されて，`Risa/Asir` の動作が極端に遅くなったり不安定になったりする[*2]．（3D グラフ描画の際に生じた）このような不具合への対処のため**リスト形式函数**を定義した．

最も簡単なものは

[返す式，[不定元，函数子，引数]]

というリストの形で，上の**引数**に不定元 `x` が含まれているとすると，リストの `x` に数を代入し，次に（代入を行った）**引数**を与えて**函数**を呼んだ戻り値を**返す式**の**不定元**に代入して得られた値を返す，という形でリスト形式函数の値が求められる．

`sin(x)` をリスト形式函数で表現すると `[z,[z,sin,x]]` となり[*3]，リスト形式函数に対する `eval()` や `deval()` に対応するものとして `myeval()` と `mydeval()` が定義されている．よってたとえば $\sin^2(0.1)$ は

```
[0] F=sin(x)^2$
[1] FL=[z^2,[z,sin,x]]$
[2] eval(subst(F,x,0.1));
0.009966711079379185537 1
[3] myeval(subst(FL,x,0.1));
0.009966711079379185537 1
[4] dsin(0.1)^2;
0.00996671
[5] deval(sin(0.1)^2)-dsin(0.1)^2;
0
```

*2 現時点では，`Risa/Asir` の再起動以外には，このように発生した不定元をメモリーから消す手段が提供されていない．

*3 この `z` は任意の不定元でよい．

```
[6] os_md.myfdeval([z^2,[z,dsin,x]],0.1);
0.00996671
```

のようになる．

ここで myfdeval() は，subst() と mydeval() とを同事に行う函数である（同様な myfeval() もある）．
より複雑なリスト形式函数の仕様については myeval() の項を参照して下さい．
なお，次のようなものはリスト形式函数である．

- sin などの函数子を含まない x^2+1 や (x+1)/(y-1) のような多項式や有理式
- リスト形式函数にリスト形式函数を代入したものは，リスト形式函数となる（compdf() で代入が可能）．

f2df() は，函数子を含む通常の函数をリスト形式函数に変換する．変換で sin のような函数は dsin のような数値のみを引数とする函数で置き換えるので，多くの値を代入しても不定元の大量発生は起こらない．一方 sin(0.1)^2 の計算は dsin(0.1)^2 に置き換えられるので，任意精度（bigfloat）浮動小数の計算ではなく倍精度浮動小数計算となる（cf. §2.9.2）．

```
[0] deval(exp(sin(1)));
2.31978
[1] Exp_x=os_md.f2df(exp(x));
[z__,[z__,os_md.myexp,x]]
[2] Sin_x=os_md.f2df(sin(x));
[z__,[z__,os_md.mysin,x]]
[3] F=os_md.compdf(Exp_x,x,Sin_x);
[z__,[z__00,os_md.mysin,x],[z__,os_md.myexp,z__00]]
[4] os_md.myfdeval(F,1);
2.31978
[5] G=os_md.f2df(exp(sin(x)));
[z__,[z__,os_md.myexp,[z1__,[z1__,os_md.mysin,x]]]]
[6] os_md.myfdeval(G,1);
2.31978
[7] os_md.compdf(y/(x^2+1),[x,y],[Exp_x,Sin_x]);
[(z__01)/(z__00^2+1),[z__01,os_md.mysin,x],[z__00,os_md.myexp,x]]
```

なお，上の myexp() や mysin() は dexp() や dsin() と違って複素変数も許す拡張された数値函数である．

- 数値函数で，**不定変数**，すなわち，不定元，有理式，函数やリスト形式函数（など f2df() で扱える函数）を引数としたとき，対応するリスト形式函数を返す函数に次のようなものがある
  - 倍精度浮動小数点計算の数値函数
    myexp(), mycos(), mysin(), mytan(), myacos(), myasin(), myatan(), mylog(), mypow(), myarg(), fouriers()
  - bigfloat にも対応した
    abs(), sqrt(), frac(), arg(), gamma(), lngamma(), digamma(), dilog(), erfc(), zeta(), eta(), jell()
- 函数 f をリスト形式函数に変換する
  f2df(f)
- 函数 f の不定元 x に函数 g を代入したリスト形式函数を得る（f, g はリスト形式函数でもよい）
  compdf(f,x,g)
- 函数 f の不定元 $x_1, x_2, \ldots$ のそれぞれに函数 $g_1, g_2, \ldots$ を代入したリスト形式函数を得る
  compdf(f,[x_1,x_2,...],[g_1,g_2,...])

- 函数子 f で変数が $v_1,\ldots,v_n$ のリスト形式函数を得る（n は f の引数の数）
  v_j は数値，不定元，有理式，リスト形式函数など f2df() で解釈できる函数でよい．
  todf(f,[v_1,...,v_n])
- 函数子 f のオプションリストが [ops] で変数が $v_1,\ldots,v_n$ のリスト形式函数を得る．
  v_j は数値，不定元，有理式，リスト形式函数など f2df() で解釈できる函数でよい．
  todf([f,[ops]],[v_1,...,v_n])
- 変数への代入などによって値を評価できる（リスト形式）函数 f を評価して函数値を得る
  myeval(f)　mydeval(f)　myval(f)
  myeval() は eval() にあたり（可能なら bigfloat），mydeval() は deval() にあたり（複素数値には非対応），myval() は可能な限り浮動小数点を用いない値を得る．
- （リスト形式）函数 f の不定元 x に値 a を代入した函数値を得る
  myfeval(f,a)　myfdeval(f,a)
  前者は eval() に対応（可能なら bigfloat），後者は deval() に対応（複素数値には非対応）．
- （リスト形式）函数 f の不定元 x に値 a を代入した函数値を得る
  myfeval(f,[x,a])　myfdeval(f,[x,a])
- （リスト形式）函数 f の不定元 $x_1,x_2,\ldots$ に値 $a_1,a_2,\ldots$ を代入した函数値を得る
  myfeval(f,[[x_1,a_1],[x_2,a_2],...])　myfdeval()(f,[[x_1,a_1],[x_2,a_2],...])
- （リスト形式）函数 f の不定元 x, y に値 a, b を代入した函数値を得る
  myf2eval(f,a,b)　myf2deval(f,a,b)
- リスト形式函数 f を倍精度浮動小数点計算から bigfloat 計算へ変更する
  df2big(f)
- リスト形式函数 f を bigfloat 計算から倍精度浮動小数点計算へ変更する
  df2big(f|inv=1)
- 函数 f の変数 x の a_j での値を b_j に変更した（$j=1,\ldots,n-1$）変数 x のリスト形式函数を得る
  cutf(f,x,[[],[a_1,b_1],,...,[a_{n-1},b_{n-1}],[]])
- $a\le x<b$ のとき $f(x)$ となる周期 $b-a$ の x 変数のリスト形式函数を得る（f の変数は x とする）．
  periodicf(f,[a,b],x)
- $k=1,\ldots,n$ に対して $(k-1)c\le x<kc$ のとき $h_k(y)$ $(0\le y<c,\ \frac{x-y}{c}\in\mathbb{Z})$ となる周期 nc の x 変数のリスト形式函数を得る
  periodicf(ltov([h_1,h_2,...,h_n]),c,x)
- 変数の範囲によって異なる函数の値を取るリスト形式函数を得る

$$g(x)=\begin{cases} v_1(x) & (x<a_0),\\ v_k & (x=a_k,\ k=1,\ldots,n-1),\\ v_n(x) & (x>a_n),\\ f(x) & (\text{上以外の } x)\end{cases}$$

  という x を変数とするリスト形式函数 $g(x)$ は次のようにして得られる．
  ただし，f, v_1, v_n の変数は x で（v_1, v_n は数でもよい），$a_0\le a_1<a_2<\cdots<a_{n-1}\le a_n$ となっている必要がある（以下で [t,[a_0,v_0],[a_1,v_1],...,[a_n,v_n]] とすると，f の変数の t に対応する）．
  cutf(f,x,[[a_0,v_1],[a_1,v_2],...,[a_n,v_n]])
- pari($funct$,) で呼び出される函数 $funct$ を変数 x のリスト形式函数にする
  [z_,[z_,os_md.evals,["pari","$funct$",x]]]

3.2.14.4　実数値／複素数値函数の解析

以下の函数には sin(x) のような函数のほか，リスト形式函数も含まれる．

零点

- 実数値函数 f の区間 $[x_1, x_2]$ における（近似）零点とそこでの f の値を求める
  fzero(f,[x_1,x_2]|mesh=m,dev=d,dif=1,zero=1)
- x 変数の多項式 p の実根を求める
  polroots(p,x)
- x 変数の多項式 p の実根を $a \le x \le b$ の範囲で求める
  polroots(p,x|lim=[a,b])
- x 変数の多項式 p の（指定した範囲にある）有理根を求める
  polroots(p,x|comp=2,lim=[a,b])
- z 変数の多項式 p の（指定した範囲 $a \le \mathrm{Re}\, z \le b,\ b \le \mathrm{Im}\, z \le d$）内の）複素根をすべて求める
  polroots(p,z|comp=1,lim=[z,[a,b],[c,d]])
  実部の制限がないときは [z,[],[c,d]], 虚部の制限がないときは [z,[a,b]] とする.
- z 変数の多項式 p の（指定した範囲内の）複素根をすべて求め，有理根は近似値でなく有理数を返す
  polroots(p,z|comp=-1,lim=[z,[a,b],[c,d]])
- $x_1, \ldots, x_n$ 変数の n 個の多項式 $p_1, \ldots, p_n$ の共通実根をもとめる
  polroots([p_1,...,p_n],[x_1,...,x_n]|err=r)
- $z_1, \ldots, z_n$ 変数の n 個の多項式 $p_1, \ldots, p_n$ の（範囲など）指定した条件を満たす共通根もとめる
  polroots([p_1,...],[z_1,...]|comp=t,lim=[[z_1,[a_1,b_1],[c_1,d_1]],...],err=r)

極値

- 実数値函数 f の点 a での極限値を求める（$a = -\infty$ を "-" で, $a = \infty$ を "+" で表す）
  flim(f,a|prec=c,init=t)
- 実数値函数 f の点 a での右極限値や左極限値を求める
  flim(f,["±",a]|prec=c,init=t)
- 実数値函数 f の区間 $[x_1, x_2]$ の極値とそのときの変数の値を求める
  fmmx(f,[x_1,x_2]|mesh=m,dev=d,dif=1,zero=1)
- 実数値函数 g の区間 $[x_1, x_2]$ 内の零点とその点における函数 f の値を求める
  fmmx(f,[x_1,x_2]|mesh=m,dev=d,dif=g,zero=1)
- 実数値函数 f の区間 $[x_1, x_2]$ での最小値と最大値とそのときの変数の値を求める
  fmmx(f,[x_1,x_2]|mmx=1,mesh=m,dev=d,dif=1,zero=1)

数値積分

- 区間 $[a, b]$ での実数値函数 f の数値積分（n は分割点の個数）
  a = "-" は $-\infty$ を，b = "+" は $+\infty$ を意味する.
  変数が x でなくて t のときは，[a,b] を [t,a,b] とする.
  $[a, b]$ の外でも f が滑らかに定義されているときは，分割点の個数の -1 倍を n とする.
  fint(f,n,[a,b]|exp=c,int=k,prec=v)
- 区間 $[a, b]$ での複素数値函数 f の数値積分（n は分割点の個数）
  fint(f,n,[a,b]|cpx=1,exp=c,int=k,prec=v)
- 複素数値函数 f の C^1 級曲線 $[a, b] \ni \mathtt{t} \mapsto \big(\phi(\mathtt{t}), \psi(\mathtt{t})\big)$ に沿った複素積分
  f の変数は複素数 z とその実部 x と虚部 y を用いることができる.
  fint(f,n,[[ϕ,ψ],[a,b]]|exp=c,int=k,prec=v)
- 複素数値函数 f の区分的 C^1 級曲線に沿った複素積分（f の変数は z, x, y）
  fint(f,n,[[[ϕ_1,ψ_1],[a_1,b_1]],[[ϕ_2,ψ_2],[a_2,b_2]]...]|exp=c,int=k,prec=v)
- 点 $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$ を順に繋ぐ折線に沿った複素数値函数 f の複素積分
  fint(f,n,[[a_1,b_1],[a_2,b_2],...,[a_n,b_n]]|exp=c,int=k,prec=v)

- 点 $(a_1,b_1),\ldots,(a_n,b_n),(a_1,b_1)$ を順に繋ぐ閉じた折線に沿った複素数値函数 f の複素積分
  `fint(`f`,`n`,[[`a_1`,`b_1`],[`a_2`,`b_2`],...,[`a_n`,`b_n`],-1]|exp=`c`,int=`k`,prec=`v`)`
- 曲線 $C:[a,b]\ni \mathtt{t}\mapsto \big(\phi(\mathtt{t}),\psi(\mathtt{t})\big)$ に沿った線積分 $\int_C y\,dx$
  無限区間，すなわち $a=$ `"-"`（$-\infty$ を意味する），$b=$ `"+"`（$+\infty$ を意味する）も可
  `areabezier([[`ϕ`,`ψ`],`n`,[t,`a`,`b`]]|exp=`c`,int=`k`,prec=`v`)`
- `z` 変数の有理式 $\frac{p}{q}$（q は多項式）の $\forall f_j>0$ で表される領域の周での複素積分（留数計算）
  `fresidue(`p`,`q`|cond=[`f_1`,`f_2`,...],sum=2)`
- 滑らかな閉曲線 $C:[a,b]\ni \mathtt{t}\mapsto \big(\phi(\mathtt{t}),\psi(\mathtt{t})\big)$ で囲まれた領域の面積（曲線が時計回りならその -1 倍）
  `areabezier([[`ϕ`,`ψ`],`n`,[t,`a`,`b`]]|int=`k`,prec=`v`)`
- 点 $(a_1,b_1),(a_2,b_2),\ldots,(a_n,b_n)$ を順に通り始点にもどる折線で囲まれた多角形の面積
  `areabezier(xylines([[`a_1`,`b_1`],...,[`a_n`,`b_n`]]|close=1))`
- 点 $(a_1,b_1),(a_2,b_2),\ldots,(a_n,b_n)$ を順に通り始点にもどる滑らかな閉曲線で囲まれた部分の面積
  `areabezier(xylines([[`a_1`,`b_1`],...,[`a_n`,`b_n`]]|close=1,curve=1))`
- 区分的 Bézier 曲線 ℓ で囲まれた部分の面積（ℓ は `lbezier()` で扱われるデータ形式のいずれか）
  `areabezier(`ℓ`)`

不定積分（原始関数）

- x を変数とする函数 f の不定積分 $\int f(x)\,dx$
  `integrate(`f`,`x`)`
- x を変数とする多項式 p の $[0,x]$ での積分 $\int_0^x p(x)\,dx$
  `intpoly(`p`,`x`)`
- 原始関数を求めることによる定積分 $\int_a^b f(x)\,dx$ の計算
  `integrate(`f`,`x`|I=[`a`,`b`])`
- x を変数とする有理函数 p の原始関数 $\int p(x)\,dx$
  `intpoly(`p`,`x`)`
- 多項式 p と三角関数 $\sin\lambda x$ の積の原始関数
  `intpoly(`p`,`x`|sin=`λ`)`
  戻り値 `[`$g(x)$`,`$h(x)$`]` は $\int p(x)\sin\lambda x\,dx=g(x)\cos\lambda x+h(x)\sin\lambda x$ を意味する．
- 多項式 p と三角関数 $\cos\lambda x$ の積の原始関数
  `intpoly(`p`,`x`|cos=`λ`)`
  戻り値 `[`$g(x)$`,`$h(x)$`]` は $\int p(x)\cos\lambda x\,dx=g(x)\cos\lambda x+h(x)\sin\lambda x$ を意味する．
- 多項式 p とべき函数 x^λ の積の原始関数
  `intpoly(`p`,`x`|pow=`λ`)`
  戻り値 $g(x)$ は $\int p(x)x^{\lambda x}dx=g(x)x^\lambda$ を意味する．
- 多項式 p と対数関数 $\log(\lambda x+c)$ の積の原始関数
  `intpoly(`p`,`x`|log=[`λ`,`c`])`
  戻り値 `[`$g(x)$`,`$h(x)$`]` は $\int p(x)\log(\lambda x+c)dx\,dx=g(x)\log(\lambda x+c)+h(x)$ を意味する．
- 多項式 p と対数関数の自然数べき $\log^m(\lambda x+c)$ の積の原始関数
  `intpoly(`p`,`x`|log=[`λ`,`c`,`m`])`
  戻り値 `[`$g_0(x)$`,`$g_1(x)$`,...]` は $\int p(x)\log^m(\lambda x+c)dx=\sum_{j=0}^m g_j(x)\log^{m-j}(\lambda x+c)$ を意味する．

留数

- `z` 変数の多項式 q を分母とする有理式 $\frac{p}{q}$ の特異点と留数のリストを得る
  `fresidue(`p`,`q`)`
- $f_1>0$, $f_2>0,\ldots$ を満たす領域での有理式 $\frac{p}{q}$ の特異点と留数のリストを得る
  f_j は，`z`, `x`, `y` の（リスト形式）函数（`z` $=$ `x` $+$ `y`i）
  `fresidue(`p`,`q`|cond=[`f_1`,`f_2`,...])`

- 条件を満たす留数の和を求める
  `fresidue(`p`,`q`|cond=[`f_1`,`f_2`,...],sum=1)`
- `load("sp")` によって `af_noalg()` を使えるようにおくと，正確な値が求められる範囲が拡がる (cf. `fctri()`).

初等関数の変換

- x 変数の三角関数を，複素変数の指数関数に変更し，指数関数の積は一つにまとめる
  `trig2exp(`f`,`x`)`
- x 変数の三角関数や複素変数の指数関数を実変数の三角関数と指数関数に変換し，三角関数の積は和にまとめる
  `trig2exp(`f`,`x`|inv=1)`
- x 変数の三角関数を，可能な限り $\sin x$ の多項式や $\sin x$ の多項式と $\cos x$ との積の和に変換する
  `trig2exp(`f`,`x`|inv=sin(`x`))`
- x 変数の三角関数を，可能な限り $\cos x$ の多項式や $\cos x$ の多項式と $\sin x$ との積の和に変換する
  `trig2exp(`f`,`x`|inv=cos(`x`))`

3.2.14.5 表形式データ

エクセルの表形式データのような表形式データを，Risa/Asir で処理するための函数について記す．そこでは，各行をリストとし，それを集めたリストを**リスト形式データ**と呼ぶことにする．また，行をリスト形式データの要素，縦の列を「項目」と呼ぶことにする．

表形式データの変換

- エクセルの CSV 形式データをリスト形式データに変換して読み込む（`eval="all"`とすると，整数または小数を数として読み込む．デフォルトは文字列として読み込み，空の要素の項目は空文字列として読み込む）
  `readcsv(`s`|eval=[`n_1`,`n_2`,...],eval=`n`,null=`t`)`
- リスト形式データを CSV 形式に変換（ファイルとして出力するには，`fcat()` を用いるとよい）
  `tocsv(`ℓ`)`
- リスト形式データを CSV 形式に変換して，それを `Excel` に渡す
  `tocsv(`ℓ`|exe=`f`)`
- リスト形式データを行列に変換
  `lv2m(`ℓ`)`
- 行列をリスト形式データに変換
  `m2ll(`m`)`
- リスト形式データを TEX の表に変換
  `ltotex(`ℓ`|opt="tab",title=`s`)`
- 最初の n 個とそれ以降の要素との 2 つの表形式リストに分割する
  `lsort(`ℓ`,[],"cut"|c1=`n`)`
- 最初の方の要素とそれ以降の n 個の要素の 2 つの表形式リストに分割する
  `lsort(`ℓ`,[],"cut"|c1=-`n`)`
- 要素の項目数が異なっているとき，後ろに s（デフォルトは空文字）を詰めて揃える
  `lsort(`ℓ`,"adjust",["put",s])`
- 表を転置する（行と列をひっくり返す）
  `lsort(`ℓ`,"transpose",[])`

データのソート

- t_1 番目の項目でソート
  `lsort(`ℓ`,"sort",["put",`t_1`])`
- t_1 番目の項目で逆順ソート

lsort(ℓ,"sort",["put",$-t_1-1$])
- 多重キーを用いた要素のソート
  msort(ℓ,s)
- t_1 番目が同じものは一つを除いて削除（重複を省く）
  lsort(ℓ,"sort",["reduce"])

項目を付加

- 要素の先頭に，中身が s の項目を付加
  lsort(ℓ,s,"cons")
- 要素の先頭に，$s_1, s_2, \ldots$ を順に付加
  lsort(ℓ,[s_1,s_2,...],"cons")
- 項目番号を順に入れたリストを最初の要素として付加
  lsort(ℓ,"num",["col"])
- 各要素の先頭に要素番号を n から順に入れた項目を付加
  lsort(ℓ,"num",["put",n])

要素や項目を抜き出す

- 項目 $c_{1,1}, c_{1,2}, \cdots$ を抜き出す
  lsort(ℓ,"col",["put"]|c_1=[$c_{1,1}$,$c_{1,2}$,...])
- 項目 $c_{1,1}, c_{1,2}, \cdots$ を除く
  lsort(ℓ,"col",["setminus"]|c_1=[$c_{1,1}$,$c_{1,2}$,...])
- $c_{1,1}$ 番目，$c_{1,2}$ 番目 ... の要素をこの順に抜き出す
  lsort(ℓ,"num",["get"]|c_1=[$c_{1,1}$,$c_{1,2}$,...])
- $c_{1,1}$ 番目，$c_{1,2}$ 番目,... の要素を削除
  lsort(ℓ,"num",["sub"]|c_1=[$c_{1,1}$,$c_{1,2}$,...])
- n 番目の項目が s と等しいものを抜きだす
  lsort(ℓ,"num",["get",["=",s],n])
- n 番目の項目が s と等しいものを除く
  lsort(ℓ,"num",["sub",["=",s],n])
- n 番目の項目が s より大きい（小さい etc.）ものを抜きだす
  （次とその次の ">" は "<", ">=", "<=", "!=" などが可能）
  lsort(ℓ,"num",["get",[">",s],n])
- n 番目の項目が s より大きい（小さい etc.）ものを削除する
  lsort(ℓ,"num",["sub",[">",s],n])
- n 番目の項目（n がなければ要素）を函数 f に入れた戻り値が 0 でない要素を抜き出す
  lsort(ℓ,"num",["get",f,n])
- n 番目の項目（n がなければ要素）を函数 f に入れた戻り値が 0 でない要素を削除
  lsort(ℓ,"num",["sub",f,n])

数を数える

- 要素の個数を返す
  lsort(ℓ,"num",["sum"])
- n 番目の項目が s に等しいもの個数を返す（以下の "=" は，"!=", ">", "<", ">=", <= が可能）
  lsort(ℓ,"num",["sum",["=",s],n])
- c_1 番目の項目，c_1 番目の項目，... の和（数でない項目は 0）を全ての要素に渡って足し合わせて返す
  lsort(ℓ,"num",["sum","+"]|c1=[$c_{1,1}$,$c_{1,2}$,...])
- c_1 番目の項目，c_1 番目の項目,... の和を足し合わせたものを要素の最初の項目に追加
  lsort(ℓ,"num",["put","+"]|c1=[$c_{1,1}$,$c_{1,2}$,...])
- n 番目の項目（n がなければ要素）を函数 f に入れた戻り値を要素の最初の項目に追加

lsort(ℓ,"num",["put",f,n])
- n 番目の項目（n がなければ要素）を函数 f に入れた戻り値を全ての要素について足した和を返す
lsort(ℓ,"num",["sum",f,n])

項目の書き換え

- m_ν 番目の要素の n_ν 番目の項目を s_ν に置き換える
lsort(ℓ,"subst",[]|c1=[[m_1,n_1,s_1],[m_2,n_2,s_2],...)

2 つのリスト型式データ ℓ, ℓ' に対し，指定したキー項目が同じものを対応する要素とみなして ℓ を変換

- ℓ の要素に対応する ℓ' の要素があれば，項目を合わせた要素にする
lsort(ℓ_1,ℓ_2,["cup",m,n]|c1=[$c_{1,1}$,$c_{1,2}$,...],c2=[$c_{2,1}$,$c_{2,2}$,...])
- ℓ の要素に対応する ℓ' の要素があれば，項目を合わせた要素にして抜き出す
lsort(ℓ_1,ℓ_2,["cap",m,n]|c1=[$c_{1,1}$,$c_{1,2}$,...],c2=[$c_{2,1}$,$c_{2,2}$,...])
- ℓ の要素に対応する ℓ' の要素があれば，ℓ から削除する
lsort(ℓ_1,ℓ_2,["setminus",m,n]|c1=[$c_{1,1}$,$c_{1,2}$,...])
- ℓ の要素に対応する ℓ' の要素があれば，対応する要素で ℓ' の指定した項目を ℓ の指定した項目を上書きする（複数あればリストにする）
lsort(ℓ_1,ℓ_2,["over",m,n]|c1=c_1,c2=c_2])
- ℓ の要素に対応する ℓ' の要素がり，対応する要素で ℓ' の指定した項目が空文字列でなければ，それで ℓ の指定した項目を上書きする
lsort(ℓ_1,ℓ_2,["subst",m,n]|c1=c_1,c2=c_2])

その他

- 2 つのリスト型式データの中身の違いをリストで返す
lsort(ℓ_1,ℓ_2,"cmp")
- 要素数が同じ 2 つのリスト型式データの各要素を繋げて返す
lsort(ℓ_1,ℓ_2,"append")

3.3 Some functions in the original library

以下の函数は，module 化されないので，函数名の先頭に os_md. をつけません.

3.3.1 数の演算

1. idiv(i_1,i_2)
:: 整数除算による商
2. irem (i_1,i_2)
:: 整数除算による剰余
   - i_1 の i_2 による整数除算による商, 剰余を求める.
   - i_2 は 0 であってはならない.
   - 被除数が負の場合, 絶対値に対する値にマイナスをつけた値を返す.
   - i_1 % i_2 は, 結果が正に正規化されることを除けば irem() の代わりに用いることができる.
   - 多項式の場合は sdiv(), srem() を用いる.

```
[0] idiv(100,7);
14
[0] idiv(-100,7);
-14
[1] irem(100,7);
2
```

```
[2] irem(-100,7);
-2
```

3. fac(i)
   :: 自然数 i の階乗
   - i が負の場合は 0 を返す.
4. nm(p)
   :: 有理数または有理式の分子
5. dn(p)
   :: 有理数または有理式の分母
   - 与えられた有理数また有理式の分子及び分母を返す.
   - 有理数の場合, 分母は常に正で, 符号は分子が持つ.
   - 有理式の場合, 単に分母, 分子を取り出すだけである. 有理式に対しては, 約分は自動的には行われない. red() を明示的に呼び出す必要がある.

```
[0] [nm(-43/8),dn(-43/8)];
[-43,8]
[1] dn((x*z)/(x*y));
y*x
[2] dn(red((x*z)/(x*y)));
y
```

6. igcd(i_1,i_2)
   :: 整数 i_1 と i_2 の GCD を求める
   - 引数が整数でない場合は, エラーまたは無意味な結果を返す.
   - 多項式の場合は, gcd(), gcdz() を用いる.
   - 整数 GCD にはさまざまな方法があり, igcdcntl() で設定できる.
7. igcdcntl([i])
   :: 整数 GCD のアルゴリズムの選択

   0 Euclid 互除法 (default)
   1 binary GCD
   2 bmod GCD
   3 accelerated integer GCD

   2, 3 は [Weber] による. おおむね 3 が高速だが, 例外もある.

```
[0] A=lrandom(10^4)$
[1] B=lrandom(10^4)$
[2] C=lrandom(10^4)$
[3] D=A*C$
[4] E=A*B$
[5] cputime(1)$
[6] igcd(D,E)$
0.6sec + gc : 1.93sec(2.531sec)
[7] igcdcntl(1)$
[8] igcd(D,E)$
0.27sec(0.2635sec)
[9] igcdcntl(2)$
[10] igcd(D,E)$
0.19sec(0.1928sec)
```

```
[11] igcdcntl(3)$
[12] igcd(D,E)$
0.08sec(0.08023sec)
```

8. `ilcm(`i_1`,`i_2`)`
   :: 整数の最小公倍数を求める
   一方が 0 のとき 0 を返す.
9. `isqrt(`n`)`
   :: n の平方根を越えない最大の整数を返す
10. `inv(`i`,`m`)`
    :: m を法とする i の逆数
    - $ia = 1 \mod (m)$ なる整数 a を求める.
    - i と m は互いに素でなければならないが, `inv()` はそのチェックは行わない

```
[0] igcd(1234,4321);
1
[1] inv(1234,4321);
3239
[2] irem(3239*1234,4321);
1
```

11. `random(`[*seed*]`)`
    :: 乱数を生成する
    - 最大 $2^{32}-1$ の非負整数の乱数を生成する.
    - 0 でない引数がある時, その値を seed として設定してから, 乱数を生成する.
    - default の seed は固定のため, 種を設定しなければ, 生成される乱数の系列は起動毎に一定である.
    - 松本眞–西村拓士による Mersenne Twister アルゴリズムの, 彼ら自身による実装を用いている. (http://www.math.keio.ac.jp/matsumoto/mt.html)
    - 周期は $2^{19937}-1$ と非常に長い. `mt_save` により state をファイルに save できる. これを `mt_load` で読み込むことにより, 異る `Asir` セッション間で一つの乱数の系列を辿ることができる
12. `lrandom(`*bit*`)`
    :: 多倍長乱数を生成する
    - 高々 bit の非負整数の乱数を生成する.
    - random を複数回呼び出して結合し, 指定の bit 長にマスクしている
13. `mt_save(`*fname*`)`
    :: 乱数生成器の現在の状態をファイルにセーブする
14. `mt_load(`*fname*`)`
    :: ファイルにセーブされた乱数生成器の状態をロードする
    - ある状態をセーブし, その状態をロードすることで, 一つの疑似乱数系列を, 新規の `Asir` セッションで続けてたどることができる.

```
[0] random();
3510405877
[1] mt_save("/tmp/mt_state");
1
[2] random();
4290933890
[3] quit;
% asir
```

```
This is Asir, Version 991108.
Copyright (C) FUJITSU LABORATORIES LIMITED.
3 March 1994. All rights reserved.
[4] mt_load("/tmp/mt_state");
1
[5] random();
4290933890
```

15. `real(`*comp*`)`
    :: *comp* の実数部分
16. `imag(`*comp*`)`
    :: *comp* の虚数部分
17. `conj(`*comp*`)`
    :: *comp* の共役複素数
    - 複素数に対し, 実部, 虚部, 共役を求める.
    - これらは, 多項式に対しても働く.

```
[0] A=(2+@i)^3;
(2+11*@i)
[1] [real(A),imag(A),conj(A)];
[2,11,(2-11*@i)]
```

18. `eval(`*obj* [, *prec*]`)`
    :: *obj* の値の評価
19. `deval(`*obj*`)`
    :: *obj* の値の評価（倍精度浮動小数）
    - *obj* に含まれる函数の値を可能な限り評価する.
    - `deval` は倍精度浮動小数を結果として返すが, `eval` は有理数はそのまま残す.
    - `eval` においては, 計算は `PARI` (section pari) が行う. `deval` においては, 計算は `C` 数学ライブラリの函数を用いて行う.
    - `deval` は複素数は扱えない.
    - `eval` においては, `prec` を指定した場合, 計算は, 10 進 `prec` 桁程度で行われる. `prec` の指定がない場合, 現在設定されている精度で行われる. (See section `setprec()`.)
    - 扱える函数は, 次の通り. `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `sinh`, `cosh`, `tanh`, `asinh`, `acosh`, `atanh`, `exp`, `log`, `pow(a,b)`, `(a^b)`
    - 以下の記号を数として評価できる. ただし `@i` を扱えるのは `eval()` のみである.
      `@i` 虚数単位
      `@pi` 円周率
      `@e` 自然対数の底

```
[0] eval(exp(@pi*@i));
-1.0000000000000000000000000000
[1] eval(2^(1/2));
1.4142135623730950488763788073031
[2] eval(sin(@pi/3));
0.86602540378443864674620506632
[3] eval(sin(@pi/3)-3^(1/2)/2,50);
-2.78791084448179148471 E-58
[4] eval(1/2);
```

```
1/2
[5] deval(sin(1)^2+cos(1)^2);
1
```

20. pari(*func*,*arg*,*prec*)
:: PARI の函数 *func* を呼び出す.
- PARI [Batut et al.] は Bordeaux 大学で開発されフリーソフトウェアとして公開されている. PARI は数式処理的な機能を有してはいるが, 主なターゲットは整数論に関連した数 (bignum, bigfloat) の演算で, 四則演算に限らず bigfloat によるさまざまな函数値の評価を高速に行うことができる. PARI は他のプログラムからサブルーチンライブラリとして用いることができ, また, gp という PARI ライブラリのインタフェースにより UNIX のアプリケーションとして利用することもできる. 現在のバージョンは 2.0.17beta でいくつかの ftp site (たとえば http://pari.math.u-bordeaux.fr/) から anonymous ftp できる.
- 最後の引数 *prec* で計算精度を指定できる. *prec* を省略した場合 setprec() で指定した精度となる. 現時点で実行できる PARI の函数は次の通りである. いずれも 1 引数で Asir が対応できる型の引数をとる函数である. なお各々の機能については PARI のマニュアルを参照のこと.
  abs, adj, arg, bigomega, binary, ceil, centerlift, cf, classno, classno2, conj, content, denom, det, det2, detr, dilog, disc, discf, divisors, eigen, eintg1, erfc, eta, factor, floor, frac, galois, galoisconj, gamh, gamma, hclassno, hermite, hess, imag, image, image2, indexrank, indsort, initalg, isfund, isprime, ispsp, isqrt, issqfree, issquare, jacobi, jell, ker, keri, kerint, kerintg1, kerint2, kerr, length, lexsort, lift, lindep, lll, lllg1, lllgen, lllgram, lllgramg1, lllgramgen, lllgramint, lllgramkerim, lllgramkerimgen, lllint, lllkerim, lllkerimgen, lllrat, lngamma, logagm, mat, matrixqz2, matrixqz3, matsize, modreverse, mu, nextprime, norm, norml2, numdiv, numer, omega, order, ordred, phi, pnqn, polred, polred2, primroot, psi, quadgen, quadpoly, real, recip, redcomp, redreal, regula, reorder, reverse, rhoreal, roots, rootslong, round, sigma, signat, simplify, smalldiscf, smallfact, smallpolred, smallpolred2, smith, smith2, sort, sqr, sqred, sqrt, supplement, trace, trans, trunc, type, unit, vec, wp, wp2, zeta
- Asir で用いているのは PARI のほんの一部の機能であるが, 今後より多くの機能が利用できるよう改良する予定である.

```
/* 行列の固有ベクトルを求める. */
[0] pari(eigen,newmat(2,2,[[1,1],[1,2]]));
[ -1.6180339887498948481977192199 0.61803398874989484826 ]
[ 1 1 ]
/* 1 変数多項式の根を求める. */
[1] pari(roots,t^2-2);
[ -1.41421356237309504876 1.41421356237309504876 ]
```

21. setprec([*n*])
:: bigfloat の桁数を *n* 桁に設定する
- 引数がある場合, bigfloat の桁数を *n* 桁に設定する. 引数のあるなしにかかわらず, 以前に設定されていた値を返す.
- bigfloat の計算は PARI によって行われる.
- bigfloat での計算に対し有効である. bigfloat の flag を on にする方法は, ctrl() を参照. 設定できる桁数に上限はないが, 指定した桁数に設定されるとは限らない. 大きめの値を設定するのが安全である.

```
[0] setprec();
```

```
9
[1] setprec(100);
9
[3] setprec(100);
96
```

22. setmode([p])
:: 有限体を $GF(p)$ に設定する
- p は 2^{27} 未満の素数
- 有限体を $GF(p)$ に設定する. 設定値を返す.
- 有限体の元の型を持つ数は, それ自身はどの有限体に属するかの情報を持たず, 現在設定されている素数 p により $GF(p)$ 上での演算が適用される.
- 位数の大きな有限体に関しては有限体に関する演算を参照.

```
[0] A=dp_mod(dp_ptod(2*x,[x]),3,[]);
(2)*<<1>>
[1] A+A;
addmi : invalid modulus
return to toplevel
[2] setmod(3);
3
[3] A+A;
(1)*<<1>>
```

23. ntoint32(n)
:: 非負整数と符号なし 32bit 整数の間の型変換
24. int32ton($int32$)
:: 非負整数と符号なし 32bit 整数の間の型変換
- n は 2^{32} 未満の非負整数, $int32$ は符号なし 32bit 整数
- 非負整数 (識別子 1) の符号なし 32bit 整数 (識別子 10) への変換, またはその逆変換を行う.
- 32bit 整数は OpenXM の基本構成要素であり, 整数をその型で送信する必要がある場合に用いる.

25. iand(m,n)
:: 整数 m, n のビット毎の and
26. ior(m,n)
:: 整数 m, n のビット毎の or
27. ixor(m,n)
:: 整数 m, n のビット毎の xor
- 整数 m, n の絶対値を bit 列とみて演算する.
- 引数の符号は無視し, 非負の値を返す.

```
[0] ctrl("hex",1);
0x1
[1] iand(0xeeeeeeeeeeeeeeee,0x2984723234812312312);
0x4622224802202202
[2] ior(0xa0a0a0a0a0a0a0a0,0xb0c0b0b0b0b0b0b0);
0xabacababababababab
[3] ixor(0xfffffffffff,0x234234234234);
0x2cbdcbdcbdcb
```

28. ishift(i,$count$)
:: 整数 i の絶対値を bit 列とみて shift する
- i の符号は無視し, 非負の値を返す.
- $count$ が正ならば右 shift, 負ならば左 shift を行う

```
[0] ctrl("hex",1);
0x1
[1] ishift(0x1000000,12);
0x1000
[2] ishift(0x1000,-12);
0x1000000
[3] ixor(0x1248,ishift(1,-16)-1);
0xedb7
```

29. i2hex(i|cap=1,num=1,min=m)
:: 非負整数 i の 16 進表示
- cap=1 16 進数表示に大文字を使う
- num=1 16 進数表示の先頭に 0x をつける
- min=m 必要なら先頭に 0 をつけて m 桁以上にする（デフォルトは $m = 2$）
- i が非負整数のリストのときは, 各成分を 16 進表示して返す
- i が非負整数やリストでないときは整数 0 を返す

```
[0] os_md.i2hex(123);
7b
[1] os_md.i2hex(123|cap=1);
7B
[2] os_md.i2hex(123|cap=1,num=1);
0x7B
[3] os_md.i2hex([123,0,231]);
[7b,00,e7]
```

30. pari(binary,n)
:: 数 n の 2 進数表示

```
[0]  pari(binary,6);
[ 1 1 0 ]
[1] pari(binary,6.3);
[ [ 1 1 0 ] [ 0 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 0 1 1 0 0 1 1 0 0
0 0 0 0 0 0 0 0 0 ] ]
```

31. pari(factor,n)
:: 整数 n または 1 変数多項式 n の素因数分解を求める

```
[0] pari(factor,1239321);
[ 3 1 ]
[ 73 1 ]
[ 5659 1 ]
[1] pari(factor,-24);
[ -1 1 ]
```

```
[ 2 3 ]
[ 3 1 ]
[2] pari(factor,2*x^3-2);
[ x-1 1 ]
[ x^2+x+1 1 ]
[3] pari(factor,x^2-y^2);
 ***   sorry, factor for general polynomials is not yet implemented.
```

32. pari(issquare,*n*)
:: 整数 *n* または多項式 *n* が平方数（元）かどうか調べる

```
[0] pari(issquare,36);
1
[1] pari(issquare,8);
0
[2] pari(issquare,16/9);
1
[3] pari(issquare,9*x^2+12*x*y+4*y^2);
1
[4] pari(issquare,2*x^2);
0
```

33. pari(omega,*n*)
:: 整数 *n* または 1 変数多項式 *n* の素因子の個数
34. pari(bigomega,*n*)
:: 整数 *n* または 1 変数多項式 *n* の素因子の重複度込みの個数

```
[0] pari(omega,-24);
2
[1] pari(bigomega,-24);
4
```

35. pari(numdiv,*n*)
:: 整数 *n* を割り切る正整数の個数

```
[0] pari(numdiv,-12);
12
```

36. pari(sigma,*n*)
:: *n* の正の約数の和

```
[0] pari(sigma,-12);
28
```

37. pari(isprime,*num*)
:: 自然数 *num* が素数かどうか調べる（合成数なら 0 を返す）

```
[0] pari(isprime,113);
1
```

38. pari(ispsp,*num*)
:: 自然数 *num* が擬素数かどうか調べる（合成数なら 0 を返す）

Baillie-Pomerance-Selfridge-Wagstaff の擬素数かどうかを調べる.
少なくとも $2^{64} = 1.84\cdots \times 10^{20}$ 以下では素数と一致.

```
[0] pari(ispsp,113);
1
```

39. pari(nextprime,*num*)
:: 自然数 *num* 以上の最小の擬素数を返す

```
[0] pari(nextprime,10^30);
1000000000000000000000000000057
```

40. pari(frac,*num*)
:: 実数 *num* の小数部分

```
[0] pari(frac,2.25);
0.25000000000000000000
```

41. pari(conj,*num*)
:: 複素数 *num*（行列でも可）の共役複素数を返す

```
[0] pari(conj,2+3*@i);
(2-3*@i)
```

42. pari(abs,*num*[,*prec*])
:: 複素数 *num* の絶対値を返す

```
[0] pari(abs,1+@i);
1.4142135623730950488016887242096980785696718753769480731766
[1] pari(abs,1+@i,50);
1.41421356237309504880168872420969807856967187537694807317638
```

43. pari(cf,*num*)
:: *num* の連分数展開を返す

```
[0] pari(cf,1234567/678901);
[ 1 1 4 1 1 27 3 2 2 3 3 5 2 ]
[1] pari(cf,eval(@pi));
[ 3 7 15 1 292 1 1 1 2 1 3 1 14 2 1 1 2 2 2 ]
```

44. pari(mu,n)
:: 自然数 n のメビウス函数 $\mu(n)$ の値（0 または ±1）を返す

```
[0] pari(mu,230);
-1
```

45. pari(phi,n)
:: 自然数 n に対する Euler の φ 函数の値（n 以下の n と素なものの個数）を返す

```
[0] pari(phi,12);
4
```

3.3.2 多項式，有理式の演算

46. var(*rat*)

:: *rat* の主変数を返す
- 主変数に関しては, Asir で使用可能な型の項を参照
- デフォルトの変数順序は次のようになっている. x, y, z, u, v, w, p, q, r, s, t, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, 以後は変数の現れた順.

```
[0] var(x^2+y^2+a^2);
x
[1] var(a*b*c*d*e);
a
[2] var(3/abc+2*xy/efg);
```

47. vars(*obj*)
:: *obj* に含まれる変数のリスト
- 与えられた式に含まれる変数のリストを返す.
- 変数順序の高いものから順に並べる.
- 関数の引数に現れる変数（たとえば exp(x*sin(y+z))）もリストするには varargs() を用いる.

```
[0] vars(x^2+y^2+a^2);
[x,y,a]
[1] vars(3/abc+2*xy/efg);
[abc,xy,efg]
[2] vars([x,y,z]);
[x,y,z]
```

48. uc()
:: 未定係数法のための vtype が 1 の不定元を生成する.
- uc() を実行するたびに, _0, _1, _2,... という不定元を生成する.
- uc() で生成された不定元は, 直接キーボードから入力することができない. これは, プログラム中で未定係数を自動生成する場合, 入力などに含まれる不定元と同一のものが生成されることを防ぐためである.
- 通常の不定元 (vtype が 0) の自動生成には rtostr(), strtov() を用いる.
- uc() で生成された不定元の不定元としての型 (vtype) は 1 である. (See section 不定元の型.)
- makev() では不定元の生成が容易に出来る.

```
[0] A=uc();
_0
[1] B=uc();
_1
[2] (uc()+uc())^2;
_2^2+2*_3*_2+_3^2
[3] (A+B)^2;
_0^2+2*_1*_0+_1^2
```

49. coef(*poly*,*deg* [,*var*])
:: *poly* の *var* (省略時は主変数) に関する *deg* 次の係数を出力する
- *var* は, 省略すると主変数 var(*poly*) だとみなされる.
- *var* が主変数でない時, *var* が主変数の場合に比較して効率が落ちる.
- mycoef() は, 初等函数係数や行列係数の多項式にも対応する.

```
[0] A = (x+y+z)^3;
```

```
x^3+(3*y+3*z)*x^2+(3*y^2+6*z*y+3*z^2)*x+y^3+3*z*y^2+3*z^2*y+z^3
[1] coef(A,1,y);
3*x^2+6*z*x+3*z^2
[2] coef(A,0);
y^3+3*z*y^2+3*z^2*y+z^3
```

50. deg(*poly*,*var*)
:: *poly* の, 変数 *var* に関する最高次数
51. mindeg(*poly*,*var*)
:: *poly* の, 変数 *var* に関する最低次数
- 与えられた多項式の変数 *var* に関する最高次数, 最低次数を出力する.
- 変数 *var* を省略することは出来ない.
- 初等函数成分の行列係数多項式などにも対応した上位互換函数 mydeg(), mymindeg() がある.

```
[0] deg((x+y+z)^10,x);
10
[1] deg((x+y+z)^10,w);
0
[75] mindeg(x^2+3*x*y,x);
1
```

52. nmono(*rat*)
:: rat の単項式の項数
- 多項式を展開した状態での 0 でない係数を持つ単項式の項数を求める.
- 有理式の場合は, 分子と分母の項数の和が返される.
- 函数形式 (section 不定元の型) は, 引数が何であっても単項とみなされる.(1 個の不定元と同じ).

```
[0] nmono((x+y)^10);
11
[1] nmono((x+y)^10/(x+z)^10);
22
[2] nmono(sin((x+y)^10));
1
```

53. ord([*varlist*])
:: 変数順序の設定
- 引数があるとき, 引数の変数リストを先頭に出し, 残りの変数がその後に続くように変数順序を設定する. 引数のあるなしに関わらず, ord() の終了時における変数順序リストを返す.
- この函数による変数順序の変更を行っても, 既にプログラム変数などに代入されている式の内部形式は新しい順序に従っては変更されない. 従って, この函数による順序の変更は, Asir の起動直後, あるいは, 新たな変数が現れた時点に行われるべきである. 異なる変数順序のもとで生成された式どうしの演算が行われた場合, 予期せぬ結果が生ずることもあり得る.

```
[0] ord();
[x,y,z,u,v,w,p,q,r,s,t,a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,_x,_y,_z,_u,_v,
_w,_p,_q,_r,_s,_t,_a,_b,_c,_d,_e,_f,_g,_h,_i,_j,_k,_l,_m,_n,_o,
exp(_x),(_x)^(_y),log(_x),(_x)^(_y-1),cos(_x),sin(_x),tan(_x),
(-_x^2+1)^(-1/2),cosh(_x),sinh(_x),tanh(_x),
(_x^2+1)^(-1/2),(_x^2-1)^(-1/2)]
```

```
[1] ord([dx,dy,dz,a,b,c]);
[dx,dy,dz,a,b,c,x,y,z,u,v,w,p,q,r,s,t,d,e,f,g,h,i,j,k,l,m,n,o,_x,_y,
_z,_u,_v,_w,_p,_q,_r,_s,_t,_a,_b,_c,_d,_e,_f,_g,_h,_i,_j,_k,_l,_m,_n,
_o,exp(_x),(_x)^(_y),log(_x),(_x)^(_y-1),cos(_x),sin(_x),tan(_x),
(-_x^2+1)^(-1/2),cosh(_x),sinh(_x),tanh(_x),
(_x^2+1)^(-1/2),(_x^2-1)^(-1/2)]
```

54. sdiv(*poly1*,*poly2*[,*v*])
:: *poly1* を *poly2* で割る除算が最後まで実行できる場合に商を求める
55. sdivm(*poly1*,*poly2*,*mod*[,*v*])
:: $GF(mod)$ 上で *poly1* を *poly2* で割る除算が最後まで実行できる場合に商を求める
56. srem(*poly1*,*poly2*[,*v*])
:: *poly1* を *poly2* で割る除算が最後まで実行できる場合に剰余を求める
57. sremm(*poly1*,*poly2*,*mod*[,*v*])
:: $GF(mod)$ 上で *poly1* を *poly2* で割る除算が最後まで実行できる場合に剰余を求める
58. sqr(*poly1*,*poly2*[,*v*])
:: *poly1* を *poly2* で割る除算が最後まで実行できる場合に商, 剰余を求める
59. sqrm(*poly1*,*poly2*,*mod*[,*v*])
:: $GF(mod)$ 上で *poly1* を *poly2* で割る除算が最後まで実行できる場合に商, 剰余を求める
    - *poly1* を *poly2* の主変数 var(*poly2*) (引数 v がある場合には v) に関する多項式と見て, *poly2* で, 割り算を行う.
    - sdivm(), sremm(), sqrm() は $GF(mod)$ 上で計算する.
    - 多項式の除算は, 主係数どうしの割算により得られた商と, 主変数の適当な冪の積を *poly2* に掛けて, *poly1* から引くという操作を *poly1* の次数が *poly2* の次数より小さくなるまで繰り返して行う. この操作が, 多項式の範囲内で行われるためには, 各ステップにおいて主係数どうしの除算が, 多項式としての整除である必要がある. これが, 「除算が最後まで実行できる」ことの意味である. 典型的な場合として, *poly2* の主係数が, 有理数である場合, あるいは, *poly2* が *poly1* の因子であることがわかっている場合などがある.
    - sqr() は商と剰余を同時に求めたい時に用いる.
    - 整数除算の商, 剰余は idiv, irem を用いる.
    - 係数に対する剰余演算は % を用いる.
    - 上の仮定を満たさない, より一般の場合の割り算を行う函数 rpdiv() がある.

```
[0] sdiv((x+y+z)^3,x^2+y+a);
x+3*y+3*z
[1] srem((x+y+z)^2,x^2+y+a);
(2*y+2*z)*x+y^2+(2*z-1)*y+z^2-a
[2] X=(x+y+z)*(x-y-z)^2;
x^3+(-y-z)*x^2+(-y^2-2*z*y-z^2)*x+y^3+3*z*y^2+3*z^2*y+z^3
[3] Y=(x+y+z)^2*(x-y-z);
x^3+(y+z)*x^2+(-y^2-2*z*y-z^2)*x-y^3-3*z*y^2-3*z^2*y-z^3
[4] G=gcd(X,Y);
x^2-y^2-2*z*y-z^2
[5] sqr(X,G);
[x-y-z,0]
[6] sqr(Y,G);
[x+y+z,0]
[7] sdiv(y*x^3+x+1,y*x+1);
```

```
divsp: cannot happen
return to toplevel
```

60. `tdiv([`*poly1*`,`*poly2*`])`
:: *poly1* が *poly2* で割れたら商を割れなければ 0 を返す
- ある多項式が既約因子であることはわかっているが, その重複度がわからない場合に, `tdiv()` を繰り返し呼ぶことにより重複度がわかる.

```
[1] Y=(x+y+z)^5*(x-y-z)^3;
x^8+(2*y+2*z)*x^7+(-2*y^2-4*z*y-2*z^2)*x^6
+(-6*y^3-18*z*y^2-18*z^2*y-6*z^3)*x^5
+(6*y^5+30*z*y^4+60*z^2*y^3+60*z^3*y^2+30*z^4*y+6*z^5)*x^3
+(2*y^6+12*z*y^5+30*z^2*y^4+40*z^3*y^3+30*z^4*y^2+12*z^5*y+2*z^6)*x^2
+(-2*y^7-14*z*y^6-42*z^2*y^5-70*z^3*y^4-70*z^4*y^3-42*z^5*y^2
-14*z^6*y-2*z^7)*x-y^8-8*z*y^7-28*z^2*y^6-56*z^3*y^5-70*z^4*y^4
-56*z^5*y^3-28*z^6*y^2-8*z^7*y-z^8
[2] for(I=0,F=x+y+z,T=Y; T=tdiv(T,F); I++);
[3] I;
5
```

61. `subst(`*rat*`[,`*varn*`,`*ratn* `]*)`
:: 有理式の特定の不定元に, 定数あるいは多項式, 有理式などを代入する
62. `psubst(`*rat*`[,`*varn*`,`*ratn* `]*)`
:: 有理式の特定の不定元に, 定数あるいは多項式, 有理式などを代入する
- `subst(`*rat*`,`*var1*`,`*rat1*`,`*var2*`,`*rat2*`,...)` は, `subst(subst(`*rat*`,`*var1*`,`*rat1*`),`*var2*`,`*rat2*`,...)` と同じ意味である.
- 入力の左側から順に代入を繰り返すために, 入力の順によって結果が変わることがある.
- `subst()` は, `sin()` などの函数の引数に対しても代入を行う. `psubst()` は, このような函数を一つの独立した不定元と見なして, その引数には代入は行わない (partial substitution のつもり)
- `Asir` では, 有理式の約分は自動的には行わないため, 有理式の代入は, 思わぬ計算時間の増大を引き起こす場合がある. 有理式を代入する場合には, 問題に応じた独自の函数を書いて, なるべく分母, 分子が大きくならないように配慮することもしばしば必要なる. 分数を代入する場合も同様である.
- 約分しながら代入を行う函数 `mysubst()` がある. また `mulsubst()` は, 複数の代入の順序にはよらない.
- `subst()` の引数 *rat* が文字列ならばエラーとなるが, `mysubst()` や `mulsubst()` はそのまま *rat* を返す.
- `subst` の引数 *rat* がリスト, 配列, 行列, あるいは分散表現多項式であった場合には, それぞれの要素または係数に対して再帰的に `subst` を行う.

```
[0] subst(x^3-3*y*x^2+3*y^2*x-y^3,y,2);
x^3-6*x^2+12*x-8
[1] subst(@@,x,-1);
-27
[2] subst(x^3-3*y*x^2+3*y^2*x-y^3,y,2,x,-1);
-27
[3] subst(x*y^3,x,y,y,x);
x^4
[4] subst(x*y^3,y,x,x,y);
```

```
y^4
[5] subst(x*y^3,x,t,y,x,t,y);
y*x^3
[6] mulsubst(x*y^3,[[x,y],[y,x]]);
y*x^3
[7] subst(x*sin(x),x,t);
sin(t)*t
[8] psubst(x*sin(x),x,t);
sin(x)*t
[9] subst(["afo",x],x,2);
subst : invalid argument
return to toplevel
[10] os_md.mysubst(["afo",x],[x,2]);
[afo,2]
```

63. `diff(`*rat*`[,`*varn*`]*)` または `diff(`*rat*`,`*varlist*`)`
:: *rat* を *varn* あるいは *varlist* の中の変数で順次微分する
- 与えられた初等函数を *varn* あるいは *varlist* の中の変数で順次微分する.
- 左側の不定元より, 順に微分していく. つまり, `diff(`*rat*`,x,y)` は, `diff(diff(`*rat*`,x),y)` と同じである.
- 有理式を微分する場合は約分されないので分母が膨らんで計算が複雑になるのを避ける `appldo()` という函数がある. これは行列やベクトルにも対応している.

```
[0] diff((x+2*y)^2,x);
2*x+4*y
[1] diff((x+2*y)^2,x,y);
4
[2] diff(x/sin(log(x)+1),x);
(sin(log(x)+1)-cos(log(x)+1))/(sin(log(x)+1)^2)
[3] diff(sin(x),[x,x,x,x]);
sin(x)
[4] diff(sin(x)+1/(a+1),x,x);
(-sin(x)*a^4-4*sin(x)*a^3-6*sin(x)*a^2-4*sin(x)*a-sin(x))/(a^4+4*a^3+6*a^2+4*a+1)
[5] appldo(dx^2,sin(x)+1/(a+1),x);
-sin(x)
```

64. `res(`*var*`,`*poly1*`,`*poly2* `[,`*mod*`])`
:: 変数 *var* に関する多項式 *poly1* と *poly2* の終結式を求める
- 部分終結式アルゴリズムによる.
- 引数 *mod* がある時, $GF(mod)$ 上での計算を行う.

```
[0] res(t,(t^3+1)*x+1,(t^3+1)*y+t);
-x^3-x^2-y^3
```

65. `fctr(`*poly*`)`
:: *poly* を既約因子に分解する
66. `sqfr(`*poly*`)`
:: *poly* を無平方分解する

- 有理数係数の多項式 poly を因数分解する. `fctr()` は既約因子分解, `sqfr()` は無平方因子分解.
- 結果は [[数係数,1],[因子, 重複度],...] なるリスト.
- 数係数と全ての因子^重複度 の積が *poly* と等しい.
- 数係数 は, (*poly*/数係数) が, 整数係数で, 係数の GCD が 1 となるような多項式になるように選ばれている (cf. `ptozp()`).

```
[0] fctr(x^10-1);
[[1,1],[x-1,1],[x+1,1],[x^4+x^3+x^2+x+1,1],[x^4-x^3+x^2-x+1,1]]
[1] fctr(x^3+y^3+(z/3)^3-x*y*z);
[[1/27,1],[9*x^2+(-9*y-3*z)*x+9*y^2-3*z*y+z^2,1],[3*x+3*y+z,1]]
[2] A=(a+b+c+d)^2;
a^2+(2*b+2*c+2*d)*a+b^2+(2*c+2*d)*b+c^2+2*d*c+d^2
[3] fctr(A);
[[1,1],[a+b+c+d,2]]
[4] A=(x+1)*(x^2-y^2)^2;
x^5+x^4-2*y^2*x^3-2*y^2*x^2+y^4*x+y^4
[5] sqfr(A);
[[1,1],[x+1,1],[-x^2+y^2,2]]
[6] fctr(A);
[[1,1],[x+1,1],[-x-y,2],[x-y,2]]
```

67. `ufctrhint(`*poly*`,`*hint*`)`
:: 次数情報を用いた有理数係数の 1 変数多項式の因数分解
- 各既約因子の次数が自然数 *hint* の倍数であることがわかっている場合に *poly* の既約因子分解を `fctr()` より効率良く行う. *poly* が, *d* 次の拡大体上におけるある多項式のノルム (section 代数的数に関する演算) で無平方である場合, 各既約因子の次数は *d* の倍数となる. このような場合に用いられる

```
[0] A=t^9-15*t^6-87*t^3-125;
t^9-15*t^6-87*t^3-125
0msec
[1] N=res(t,subst(A,t,x-2*t),A);
-x^81+1215*x^78-567405*x^75+139519665*x^72-19360343142*x^69
+1720634125410*x^66-88249977024390*x^63-4856095669551930*x^60
+199938524524057142 1*x^57-1557968995259025 1515*x^54
+15956967531741971462865*x^51
...
+14039558872035397353552612361266144455065 9875*x^6
+10122324287343155430042768923500799484375*x^3
+13926274344440731013345902118273331445312 5
980msec + gc : 250msec
[2] sqfr(N);
[[-1,1],[x^81-1215*x^78+567405*x^75-139519665*x^72+19360343142*x^69
-1720634125410*x^66+88249977024390*x^63+4856095669551930*x^60
-199938524524057142 1*x^57+1557968995259025 1515*x^54
...
-10122324287343155430042768923500799484375*x^3
```

```
-139262743444407310133459021182733314453125,1]]
20msec
[3] fctr(N);
[[-1,1],[x^9-405*x^6-63423*x^3-2460375,1],
[x^18-486*x^15+98739*x^12-9316620*x^9+945468531*x^6-12368049246*x^3
+296607516309,1],[x^18-8667*x^12+19842651*x^6+19683,1],
[x^18-324*x^15+44469*x^12-1180980*x^9+427455711*x^6+2793253896*x^3
+31524548679,1],
[x^18+10773*x^12+2784051*x^6+307546875,1]]
167.050sec + gc : 1.890sec
[4] ufctrhint(N,9);
[[-1,1],[x^9-405*x^6-63423*x^3-2460375,1],
[x^18-486*x^15+98739*x^12-9316620*x^9+945468531*x^6-12368049246*x^3
+296607516309,1],[x^18-8667*x^12+19842651*x^6+19683,1],
[x^18-324*x^15+44469*x^12-1180980*x^9+427455711*x^6+2793253896*x^3
+31524548679,1],
[x^18+10773*x^12+2784051*x^6+307546875,1]]
119.340sec + gc : 1.300sec
```

68. `modfctr(`*poly*`,`*mod*`)`
:: 有限体上での多項式の因数分解
- 2^29 未満の自然数 *mod* を標数とする素体上で多項式 *poly* を既約因子に分解する.
- 結果は **`[[数係数,1],[因子, 重複度],...]`** なるリスト.
- 数係数 と 全ての 因子 重複度 の積が *poly* と等しい.
- 大きな位数を持つ有限体上の因数分解には `fctr_ff()` を用いる. (cf. 有限体に関する演算)

```
[0] modfctr(x^10+x^2+1,2147483647);
[[1,1],[x+1513477736,1],[x+2055628767,1],[x+91854880,1],
[x+634005911,1],[x+1513477735,1],[x+634005912,1],
[x^4+1759639395*x^2+2045307031,1]]
[1] modfctr(2*x^6+(y^2+z*y)*x^4+2*z*y^3*x^2+(2*z^2*y^2+z^3*y)*x+z^4,3);
[[2,1],[2*x^3+z*y*x+z^2,1],[2*x^3+y^2*x+2*z^2,1]]
```

69. `ptozp(`*poly*`|factor=1)`
:: 有理係数多項式を有理数倍して整数係数で係数の GCD が 1 の多項式に直す

- 与えられた多項式 *poly* に適当な有理数を掛けて, 整数係数かつ係数の GCD が 1 になるようにする.
- 分数の四則演算は, 整数の演算に比較して遅いため, 種々の多項式演算の前に, 多項式を整数係数にしておくことが望ましい.
- 有理式を約分する `red()` で分数係数有理式を約分しても, 分子多項式の係数は有理数のままであり, 有理式の分子を求める `nm()` では, 分数係数多項式は, 分数係数のままの形で出力されるため, 直ちに整数係数多項式を得る事は出来ない.
- オプション `factor` が設定された場合の戻り値はリスト `[`g`,`c`]` である. ここで c は有理数であり, g がオプションのない場合の戻り値であり, $poly = c * g$ となる.

```
[0] ptozp(2*x+5/3);
6*x+5
```

```
[1] nm(2*x+5/3);
2*x+5/3
```

70. % *poly* % *m*
:: *poly* の各係数を整数 *m* で割った剰余で置き換えた多項式を返す
- 結果の係数は全て正の整数となる.
- *poly* は整数でもよい. この場合, 結果が正に正規化されることを除けば irem() と同様に用いることができる.
- *poly* の係数, *m* とも整数である必要があるが, チェックは行なわれない.

```
[0] (x+2)^5 % 3;
x^5+x^4+x^3+2*x^2+2*x+2
[1] (x-2)^5 % 3;
x^5+2*x^4+x^3+x^2+2*x+1
[2] (-5) % 4;
3
[3] irem(-5,4);
-1
```

71. prim(*poly*[,*v*])
:: 有理係数多項式 *poly* の原始的部分 (primitive part)
72. cont(*poly*[,*v*])
:: 有理係数多項式 *poly* の容量 (content)
- *poly* の主変数 (引数 *v* がある場合には *v*) に関する原始的部分, 容量を求める

```
[0] E=(y-z)*(x+y)*(x-z)*(2*x-y);
(2*y-2*z)*x^3+(y^2-3*z*y+2*z^2)*x^2+(-y^3+z^2*y)*x+z*y^3-z^2*y^2
[1] prim(E);
2*x^3+(y-2*z)*x^2+(-y^2-z*y)*x+z*y^2
[2] cont(E);
y-z
[3] prim(E,z);
(y-z)*x-z*y+z^2
```

73. gcd(*poly1*,*poly2*[,*mod*])
:: 二つの多項式の最大公約式 (GCD) を求める
74. gcdz(*poly1*,*poly2*)
:: 有限体上の二つの多項式の最大公約式 (GCD) を求める
- gcd() は有理数体上の多項式としての GCD を返す. すなわち, 結果は整数係数で, かつ係数の GCD が 1 になるような多項式, または, 互いに素の場合は 1 を返す.
- gcdz() は *poly1*, *poly2* ともに整数係数の場合に, 整数環上の多項式としての GCD を返す. すなわち, gcd() の値に, 係数全体の整数 GCD の値を掛けたものを返す.
- 引数 *mod* がある時, gcd() は $GF(mod)$ 上での GCD を返す.
- gcd(), gcdz() は Extended Zassenhaus アルゴリズムによる. 有限体上の GCD は PRS アルゴリズムによっているため, 大きな問題, GCD が 1 の場合などにおいて効率が悪い.
- 有理函数体係数の多項式環や常微分作用素環, さらには整数環に汎用的に対応している mygcd() がある.

```
[0] gcd(12*(x^2+2*x+1)^2,18*(x^2+(y+1)*x+y)^3);
x^3+3*x^2+3*x+1
```

```
[1] gcdz(12*(x^2+2*x+1)^2,18*(x^2+(y+1)*x+y)^3);
6*x^3+18*x^2+18*x+6
[2] gcd((x+y)*(x-y)^2,(x+y)^2*(x-y));
x^2-y^2
[3] gcd((x+y)*(x-y)^2,(x+y)^2*(x-y),2);
x^3+y*x^2+y^2*x+y^3
```

75. red(*rat*)
:: *rat* を約分する
   - Asir は有理数の約分を常に自動的に行う. しかし, 有理式については通分は行うが, 約分はユーザーが指定しない限り行わない. この約分を行うコマンドが red である.
   - EZGCD により *rat* の分子, 分母を約分する.
   - 出力される有理式の分母の多項式は, 各係数の GCD が 1 の整数係数多項式である. 分子については整数係数多項式となるとは限らない.
   - GCD は大変重い演算なので, 他の方法で除ける共通因子は可能な限り除くのが望ましい. また, 分母, 分子が大きくなってからのこの函数の呼び出しは, 非常に時間が掛かる場合が多い. 有理式演算を行う場合は, ある程度頻繁に, 約分を行う必要がある.

```
[0] (x^3-1)/(x-1);
(x^3-1)/(x-1)
[1] red((x^3-1)/(x-1));
x^2+x+1
[2] red((x^3+y^3+z^3-3*x*y*z)/(x+y+z));
x^2+(-y-z)*x+y^2-z*y+z^2
[3] red((3*x*y)/(12*x^2+21*y^3*x));
(y)/(4*x+7*y^3)
[4] red((3/4*x^2+5/6*x)/(2*y*x+4/3*x));
(9/8*x+5/4)/(3*y+2)
```

76. umul(p_1,p_2)
:: 整数係数一変数多項式の高速乗算
77. umul_ff(p_1,p_2)
:: 有限体係数一変数多項式の高速乗算
78. usquare(p_1)
:: 整数係数一変数多項式の高速 2 乗算
79. usquare_ff(p_1)
:: 有限体係数一変数多項式の高速 2 乗算
80. utmul(p_1,p_2,d)
:: 整数係数一変数多項式の高速乗算 (打ち切り次数指定)
81. utmul_ff(p_1,p_2,d)
:: 有限体係数一変数多項式の高速乗算 (打ち切り次数指定)
   - 一変数多項式の乗算を, 次数に応じて決まるアルゴリズムを用いて高速に行う.
   - umul(), usquare(), utmul() は係数を整数と見なして, 整数係数の多項式として積を求める. 係数が有限体 $GF(p)$ の元の場合には, 係数は 0 以上 p 未満の整数と見なされる.
   - umul_ff(), usquare_ff(), utmul_ff() は, 係数を有限体の元と見なして, 有限体上の多項式として積を求める. ただし, 引数の係数が整数の場合, 整数係数の多項式を返す場合もあるので, これらを呼び出した結果が有限体係数であることを保証するためにはあらかじめ simp_ff() で係数を有限体の元に変換しておくとよい.
   - umul_ff(), usquare_ff(), utmul_ff() は, $GF(2^n)$ 係数の多項式を引数に取れない.

- umul(), umul_ff() の結果は p_1, p_2 の積, usquare(), usquare_ff() の結果は p_1 の 2 乗, utmul(), utmul_ff() の結果は p_1, p_2 の積の, d 次以下の部分となる.
- いずれも, set_upkara() (utmul, utmul_ff については set_uptkara()) で返される値以下の次数に対しては通常の筆算形式の方法, set_upfft() で返される値以下の次数に対しては Karatsuba 法, それ以上では FFT および中国剰余定理が用いられる. すなわち, 整数に対する FFT ではなく, 十分多くの 1 ワード以内の法 mi を用意し, p_1, p_2 の係数を mi で割った余りとしたものの積を, FFT で計算し, 最後に中国剰余定理で合成する. その際, 有限体版の函数においては, 最後に基礎体を表す法で各係数の剰余を計算するが, ここでは Shoup によるトリック [Shoup] を用いて高速化してある

```
[0] load("fff")$
[1] cputime(1)$
0sec(1.407e-05sec)
[2] setmod_ff(2^160-47);
1461501637330902918203684832716283019655932542929
0sec(0.00028sec)
[3] A=randpoly_ff(100,x)$
0sec(0.001422sec)
[4] B=randpoly_ff(100,x)$
0sec(0.00107sec)
[5] for(I=0;I<100;I++)A*B;
7.77sec + gc : 8.38sec(16.15sec)
[6] for(I=0;I<100;I++)umul(A,B);
2.24sec + gc : 1.52sec(3.767sec)
[7] for(I=0;I<100;I++)umul_ff(A,B);
1.42sec + gc : 0.24sec(1.653sec)
[8] for(I=0;I<100;I++)usquare_ff(A);
1.08sec + gc : 0.21sec(1.297sec)
[9] for(I=0;I<100;I++)utmul_ff(A,B,100);
1.2sec + gc : 0.17sec(1.366sec)
[10] deg(utmul_ff(A,B,100),x);
100
```

82. kmul(p_1,p_2)
:: 一変数多項式の乗算を Karatsuba 法で行う
83. ksquare(p_1)
:: 一変数多項式の高速 2 乗算を Karatsuba 法で行う
84. ktmul(p_1,p_2,d)
一変数多項式の高速乗算 (打ち切り次数指定) を Karatsuba 法で行う
- 一変数多項式の乗算を Karatsuba 法で行う.
- 基本的には umul() と同様だが, 次数が大きくなっても FFT を用いた高速化は行わない.
- $GF(2^n)$ 係数の多項式にも用いることができる.

```
[0] load("code/fff");
1
[1] setmod_ff(defpoly_mod2(160));
x^160+x^5+x^3+x^2+1
```

```
A=randpoly_ff(100,x)$
B=randpoly_ff(100,x)$
[2] umul(A,B)$
umul : invalid argument
return to toplevel
[3] kmul(A,B)$
```

85. set_upkara([*threshold*])
:: 1 変数多項式の積演算における N^2 , Karatsuba, FFT アルゴリズムの切替えの閾値
86. set_uptkara([*threshold*])
:: 1 変数多項式の積演算における N^2 , Karatsuba, FFT アルゴリズムの切替えの閾値
87. set_upfft([*threshold*])
:: 1 変数多項式の積演算における N^2 , Karatsuba, FFT アルゴリズムの切替えの閾値
   - いずれも, 一変数多項式の積の計算における, アルゴリズム切替えの閾値を設定する.
   - 一変数多項式の積は, 次数 N が小さい範囲では通常の N^2 アルゴリズム, 中程度の場合 Karatsuba アルゴリズム, 大きい場合には FFT アルゴリズムで計算される. この切替えの次数を設定する.
   - 詳細は, それぞれの積函数の項を参照のこと.
88. utrunc(p,d)
:: 一変数多項式を次数で切る
89. udecomp(p,d)
:: 一変数多項式を次数で分ける
90. ureverse(p)
:: 一変数多項式の次数を逆にした多項式を作る
   - p の変数を x とする. このとき $p = p_1 + x^{d+1}p_2$ (p_1 の次数は d 以下) と分解できる. utrunc() は p_1 を返し, udecomp() は [p_1,p_2] を返す.
   - p の次数を e とし, i 次の係数を $p[i]$ とすれば, ureverse() は $p[e] + p[e-1]x + \cdots$ を返す

```
[0] utrunc((x+1)^10,5);
252*x^5+210*x^4+120*x^3+45*x^2+10*x+1
[1] udecomp((x+1)^10,5);
[252*x^5+210*x^4+120*x^3+45*x^2+10*x+1,x^4+10*x^3+45*x^2+120*x+210]
[2] ureverse(3*x^3+x^2+2*x);
2*x^2+x+3
```

91. uinv_as_power_series(p,d)
:: 一変数多項式を冪級数とみて, 逆元計算
92. ureverse_inv_as_power_series(p,d)
:: 一変数多項式を次数を逆にして冪級数とみて, 逆元計算
   - uinv_as_power_series(p,d) は, 定数項が 0 でない多項式 p に対し, $pr-1$ の最低次数が $d+1$ 以上になるような 高々 d 次の多項式 r を求める.
   - ureverse_inv_as_power_series(p,d) は p の次数を e とするとき, p_1 =ureverse(p,e) に対して uinv_as_power_series(p_1,d) を計算する.
   - rembymul_precomp() の引数として用いる場合, ureverse_inv_as_power_series() の結果をそのまま用いることができる.

```
[123] A=(x+1)^5;
x^5+5*x^4+10*x^3+10*x^2+5*x+1
[124] uinv_as_power_series(A,5);
-126*x^5+70*x^4-35*x^3+15*x^2-5*x+1
```

```
[126] A*R;
-126*x^10-560*x^9-945*x^8-720*x^7-210*x^6+1
[127] A=x^10+x^9;
x^10+x^9
[128] R=ureverse_inv_as_power_series(A,5);
-x^5+x^4-x^3+x^2-x+1
[129] ureverse(A)*R;
-x^6+1
```

93. udiv(p_1,p_2)
:: 一変数多項式 p_1, p_2 に対し, 商を返す
94. urem(p_1,p_2)
:: 一変数多項式 p_1, p_2 に対し, 剰余を返す
95. urembymul(p_1,p_2)
:: 一変数多項式 p_1, p_2 に対し, 剰余を返す
96. urembymul_precomp(p_1,p_2,inv)
:: 固定された多項式による剰余計算を多数行う場合などに用いる
97. ugcd(p_1,p_2)
:: 一変数多項式 p_1, p_2 に対し, GCD を返す
    - 一変数多項式 p_1, p_2 に対し, udiv は商, urem, urembymul は剰余, ugcd は GCD を返す. これらは, 密な一変数多項式に対する高速化を図ったものである. urembymul は, p_2 による剰余計算を, p_2 の冪級数としての逆元計算および, 乗算 2 回に置き換えたもので, 次数が大きい場合に有効である.
    - urembymul_precomp は, 固定された多項式による剰余計算を多数行う場合などに効果を発揮する. 第 3 引数は, あらかじめ ureverse_inv_as_power_series() により計算しておく.

```
[0] setmod_ff(2^160-47);
1461501637330902918203684832716283019655932542929
[1] A=randpoly_ff(200,x)$
[2] B=randpoly_ff(101,x)$
[3] cputime(1)$
0sec(1.597e-05sec)
[4] srem(A,B)$
0.15sec + gc : 0.15sec(0.3035sec)
[5] urem(A,B)$
0.11sec + gc : 0.12sec(0.2347sec)
[6] urembymul(A,B)$
0.08sec + gc : 0.09sec(0.1651sec)
[7] R=ureverse_inv_as_power_series(B,101)$
0.04sec + gc : 0.03sec(0.063sec)
[8] urembymul_precomp(A,B,R)$
0.03sec(0.02501sec)
```

98. pari(content,p)
:: x 変数の多項式とみて, 係数の最大公約元を返す

```
[0] pari(content,12*x^2-9*x+15);
3
```

```
[1] pari(content,3*x^2-2*x+2/3);
1/3
[2]  pari(content,3*y^2*x^2-2*x*y+2/3*y);
1/3*y
```

99. pari(round,*p*)
:: 多項式の各係数を近似整数に置き換える

```
[0] pari(round,(1.3+x+3*y)^2);
x^2+(6*y+3)*x+9*y^2+8*y+2
```

100. pari(roots,*p*[,*prec*])
:: 多項式の根を求める

```
[0] pari(roots,x^3+2);
[ -1.25992104989487316475
 (0.62996052494743658237-1.09112363597172140345 7*@i)
 (0.62996052494743658237+1.09112363597172140345 7*@i) ]
```

101. pari(disc,*p*)
:: x 変数の多項式 *p* の判別式を返す.

```
[0] pari(disc,a*x^2+b*x+c);
-4*c*a+b^2
[1] pari(disc,x^4+a*x^2+b*x+c);
16*c*a^4-4*b^2*a^3-128*c^2*a^2+144*c*b^2*a-27*b^4+256*c^3
```

3.3.3 リスト，ベクトル，配列の演算

102. car(*list*)
:: 空でない *list* の先頭要素（空の場合は空のリスト）を返す
103. cdr(*list*)
:: 空でない *list* から先頭要素を除いた（空の場合は空の）リストを返す
104. cons(*obj*,*list*)
:: *list* の先頭に *obj* を加えたリストを返す
105. append(*list1*,*list2*)
:: *list1* の後に *list2* をつなげたリストを返す
m2l(|list=1) を使うと，一度に多くのリストをつなげることができる.
106. reverse(*list*)
:: 逆順に並べ替えたリストを返す
107. length(*lv*)
:: リストまたはベクトルの長さを返す

- リストは [*obj1*,*obj2*,...] と表される. *obj1* が先頭要素である.
- car() は, 空でない *list* の先頭要素を出力する. 空リストが入力された場合は, 空リストが出力される.
- cdr() は, 空でない *list* から先頭要素を取り除いたリストを出力する. 空リストが入力された場合は, 空リストが出力される.
- cons() は, *list* の先頭に *obj* を付け加えたリストを出力する.
- append() は, *list1* の要素と *list2* のすべての要素を結合させたリスト [*list1* **の要素の並び**,*list2* **の要素の並び**] を出力する.
- reverse() は, *list* を逆順にしたリストを出力する.

- `length()` は, *list* または *vect* の長さを出力する. 行列の要素の個数は, `size()` を用いる.
- リストは読み出し専用で, 要素の入れ替えはできない. リストの n 番目の要素の取り出しは, `cdr()` を n 回適用した後 `car()` を適用することにより可能であるが, 便法として, ベクトル, 行列などの配列と同様, インデックス `[`n`]` を後ろに付けることにより取り出すことができる. ただし, システム内部では, 実際にポインタを n 回たどるので, 後ろの要素ほど取り出しに時間がかかる.
- `cdr()` は新しいセルを生成しないが, `append()` は, 実際には第 1 引数のリストの長さだけの `cons()` の繰り返しとなるため, 第 1 引数のリストが長い場合には多くのメモリを消費することになる. `reverse()` に関しても同様である.

```
[0] L = [[1,2,3],4,[5,6]];
[[1,2,3],4,[5,6]]
[1] car(L);
[1,2,3]
[2] cdr(L);
[4,[5,6]]
[3] cons(x*y,L);
[y*x,[1,2,3],4,[5,6]]
[4] append([a,b,c],[d]);
[a,b,c,d]
[5] reverse([a,b,c,d]);
[d,c,b,a]
[6] length(L);
3
[7] length(ltov(L));
3
[8] L[2][0];
5
```

108. `newvect(`*len*`[,`*list*`])`
:: 長さ *len* のベクトルを生成する
109. `vector(`*len*`[,`*list*`])`
:: 長さ *len* のベクトルを生成する
110. `vect([`*elements*`])`
:: 要素の並びからベクトルを生成する
    - `vect()` は要素の並びからベクトルを生成する.
    - `vector()` は `newvect()` の別名である.
    - `newvect()` は長さ *len* のベクトルを生成する. 第 2 引数がない場合, 各成分は 0 に初期化される. 第 2 引数がある場合, インデックスの小さい成分から, リストの各要素により初期化される. 各要素は, 先頭から順に使われ, 足りない分は 0 が埋められる.
    - ベクトルの成分は, 第 0 成分から第 *len*−1 成分となる. (第 1 成分からではない事に注意.)
    - リストは各成分が, ポインタを辿る事によってシーケンシャルに呼び出されるのに対し, ベクトルは各成分が第一成分からのメモリ上の displacement (変位) によってランダムアクセスで呼び出され, その結果, 成分のアクセス時間に大きな差が出てくる. 成分アクセスは, リストでは, 成分の量が増えるに従って時間がかかるようになるが, ベクトルでは, 成分の量に依存せずほぼ一定である.
    - `Asir` では, 縦ベクトル, 横ベクトルの区別はない. 行列を左から掛ければ縦ベクトルとみなされるし, 右から掛ければ横ベクトルとみなされる.
    - ベクトルの長さは `size()` によって得られる. 函数の引数としてベクトルを渡した場合, 渡された函数は, そのベクトルの成分を書き換えることができる.

```
[0] A=newvect(5);
[ 0 0 0 0 0 ]
[1] A=newvect(5,[1,2,3,4,[5,6]]);
[ 1 2 3 4 [5,6] ]
[2] A[0];
1
[3] A[4];
[5,6]
[4] size(A);
[5]
[5] length(A);
5
[6] vect(1,2,3,4,[5,6]);
[ 1 2 3 4 [5,6] ]
[7] def afo(V) { V[0] = x; }
[8] afo(A)$
[9] A;
[ x 2 3 4 [5,6] ]
```

111. `ltov(`*list*`)`
:: *list* をベクトルに変換する
- リスト *list* を同じ長さのベクトルに変換する.
- この函数は `newvect(length(`*list*`),`*list*`)` に等しい

```
[0] A=[1,2,3];
[1] ltov(A);
[ 1 2 3 ]
```

112. `vtol(`*vect*`)`
:: ベクトルをリストに変換する
- 長さ n のベクトル *vect* を `[`*vect*`[0],...,`*vect*`[n-1]]` なるリストに変換する.
- リストからベクトルへの変換は `newvect()` で行う.

```
[0] A=newvect(3,[1,2,3]);
[ 1 2 3 ]
[1] vtol(A);
[1,2,3]
```

113. `newbytearray(`*len*`,[`*listorstring*`])`
:: newvect と同様にして byte array を生成する
- 文字列で初期値を指定することも可能である.
- byte array の要素のアクセスは配列と同様である.

```
[0] A=newbytearray(3);
|00 00 00|
[1] A=newbytearray(3,[1,2,3]);
|01 02 03|
[2] A=newbytearray(3,"abc");
|61 62 63|
```

```
[3] A[0];
97
[4] A[1]=123;
123
[5] A;
|61 7b 63|
```

114. `size(`*vect/mat*`)`
:: 行列のサイズまたはベクトルの長さを求める函数（戻り値はリスト）
- *vect* の長さ, または *mat* の大きさをリスト [*vect* の長さ], [*mat* の行数, *mat* の列数] で出力する.
- *vect* の長さは `length()` で求めることもできる.
- *list* の長さは `length()` を, 有理式に現れる単項式の数は `nmono()` を用いる.

```
[0] A = newvect(4);
[ 0 0 0 0 ]
[1] size(A);
[4]
[2] length(A);
4
[3] B = newmat(2,3,[[1,2,3],[4,5,6]]);
[ 1 2 3 ]
[ 4 5 6 ]
[4] size(B);
[2,3]
```

115. `qsort(`*array* [`,`*func*]`)`
:: 一次元配列 array をソートする.
- 一次元配列（リストまたはベクトル）を quick sort でソートする.
- 比較用函数が指定されていない場合, オブジェクトどうしの比較結果で順序が下のものから順に並べ換えられる.
- 0, 1, −1 を返す 2 引数函数が *func* として与えられた場合, *func*(A,B)=1 の場合に A<B として, 順序が下のものから順に並べ換えられる.
- 配列は新たに生成されず, 引数の配列の要素のみ入れ替わる.

```
[0] qsort(newvect(10,[1,4,6,7,3,2,9,6,0,-1]));
[ -1 0 1 2 3 4 6 6 7 9 ]
[1] def rev(A,B) { return A>B?-1:(A<B?1:0); }
[2] qsort(newvect(10,[1,4,6,7,3,2,9,6,0,-1]),rev);
[ 9 7 6 6 4 3 2 1 0 -1 ]
[3] qsort([1,4,6,7,3,2,9,6,0,-1]);
[-1,0,1,2,3,4,6,6,7,9]
```

3.3.4 行列の演算

116. `newmat(`*row*`,`*col* [`,[[`*a*`,`*b*`,...],[`*c*`,`*d*`,...],...]`] `)`
117. `matrix(`*row*`,`*col* [`,[[`*a*`,`*b*`,...],[`*c*`,`*d*`,...],...]`] `)`
:: *row* 行 *col* 列の行列を生成する
- `matrix` は `newmat` の別名である.

- *row* 行 *col* 列の行列を生成する. 第 3 引数がない場合, 各成分は 0 に初期化される. 第 3 引数がある場合, インデックスの小さい成分から, 各行が, リストの各要素 (これはまたリストである) により初期化される. 各要素は, 先頭から順に使われ, 足りない分は 0 が埋められる.
- 行列のサイズは size() で得られる.
- M が行列のとき, M[I] により第 I 行をベクトルとして取り出すことができる. このベクトルは, もとの行列と成分を共有しており, いずれかの成分を書き換えれば, 他の対応する成分も書き換わることになる.
- 函数の引数として行列を渡した場合, 渡された函数は, その行列の成分を書き換えることができる.

```
[0] A = newmat(3,3,[[1,1,1],[x,y],[x^2]]);
[ 1 1 1 ]
[ x y 0 ]
[ x^2 0 0 ]
[1] det(A);
-y*x^2
[2] size(A);
[3,3]
[3] A[1];
[ x y 0 ]
[4] A[1][3];
getarray : Out of range
return to toplevel
```

118. mat(*vector*[,...])
119. matr(*vector*[,...])
:: 行ベクトル（またはリスト）の並びから行列を生成する
120. matc(*vector*[,...])
:: 列ベクトル（またはリスト）の並びから行列を生成する
- mat() は matr() の別名.
- 行列のサイズは，最初のベクトル（リスト）の長さと引数の個数で決まり，足りない部分は 0 で埋められる.
- s2m() の方が高機能

```
[0] matr([1,2,3],[4,5,6],[7,8]);
[ 1 2 3 ]
[ 4 5 6 ]
[ 7 8 0 ]
[1] matr([1,2],[4,5,6],[7,8,0]);
[ 1 2 ]
[ 4 5 ]
[ 7 8 ]
[2] matc([1,2,3],[4,5,6],[7,8]);
[ 1 4 7 ]
[ 2 5 8 ]
[ 3 6 0 ]
[3] mat(ltov([1,2]),[4,5,6]);
[ 1 2 ]
```

```
[ 4 5 ]
```

121. det(*mat*[,*mod*])
:: *mat* の行列式

122. nd_det(*mat*[,*mod*])
:: 有理数または有限体上の多項式行列 *mat* の行列式

- det() および nd_det() は行列 *mat* の行列式を求める.
- 引数 *mod* がある時, $GF(mod)$ 上での行列式を求める. 分数なしのガウス消去法によっているため, 多変数多項式を成分とする行列に対しては小行列式展開による方法のほうが効率がよい場合もある.
- nd_det() は有理数または有限体上の多項式行列の行列式計算専用である. アルゴリズムはやはり分数なしのガウス消去法だが, データ構造および乗除算の工夫により, 一般に det() より高速に計算できる.
- 有理式を成分とする行列には非対応なので, mydet() または mydet2() を用いる.

```
[0] A=newmat(5,5)$
[1] V=[x,y,z,u,v];
[x,y,z,u,v]
[2] for(I=0;I<5;I++)for(J=0,B=A[I],W=V[I];J<5;J++)B[J]=W^J;
[3] A;
[ 1 x x^2 x^3 x^4 ]
[ 1 y y^2 y^3 y^4 ]
[ 1 z z^2 z^3 z^4 ]
[ 1 u u^2 u^3 u^4 ]
[ 1 v v^2 v^3 v^4 ]
[4] fctr(det(A));
[[1,1],[u-v,1],[-z+v,1],[-z+u,1],[-y+u,1],[y-v,1],[-y+z,1],[-x+u,1],
[-x+z,1],[-x+v,1],[-x+y,1]]
B=mat([1,0],[0,1/x]);
[ 1 0 ]
[ 0 (1)/(x) ]
[5] det(B);
internal error (SEGV)
return to toplevel
[6] os_md.mydet(B);
(1)/(x)
```

123. invmat(*mat*)
:: 行列 *mat* の逆行列

- 逆行列は [分母, 分子] の形で返され, 分母が行列, 分母/分子 が逆行列となる.
- 有理式を成分とする行列には非対応なので, そのときは myinv() を用いる.

```
[0] A = newmat(3,3)$
[1] for(I=0;I<3;I++)for(J=0,B=A[I],W=V[I];J<3;J++)B[J]=W^J;
[2] A;
[ 1 x x^2 ]
[ 1 y y^2 ]
[ 1 z z^2 ]
```

```
[3] invmat(A);
[[ -z*y^2+z^2*y z*x^2-z^2*x -y*x^2+y^2*x ]
[ y^2-z^2 -x^2+z^2 x^2-y^2 ]
[ -y+z x-z -x+y ],(-y+z)*x^2+(y^2-z^2)*x-z*y^2+z^2*y]
[4] A*B[0];
[ (-y+z)*x^2+(y^2-z^2)*x-z*y^2+z^2*y 0 0 ]
[ 0 (-y+z)*x^2+(y^2-z^2)*x-z*y^2+z^2*y 0 ]
[ 0 0 (-y+z)*x^2+(y^2-z^2)*x-z*y^2+z^2*y ]
[5] map(red,A*B[0]/B[1]);
[ 1 0 0 ]
[ 0 1 0 ]
[ 0 0 1 ]
[6] C=mat([1,0],[0,1/x]);
[ 1 0 ]
[ 0 (1)/(x) ]
[7] invmat(C);
internal error (SEGV)
return to toplevel
[8] os_md.myinv(C);
[ 1 0 ]
[ 0 x ]
[9] M=mat([2,0],[0,1/2]);
[ 2 0 ]
[ 0 1/2 ]
[10] invmat(M);
[[ 1/2 0 ]
[ 0 2 ],1]
[11] N=mat([2,0],[0,1]);
[ 2 0 ]
[ 0 1 ]
[12] invmat(N);
[[ 1 0 ]
[ 0 2 ],2]
```

124. `rowx(`*matrix*`,`i`,`j`)`
:: 第 i 行と第 j 行を交換する
125. `rowm(`*matrix*`,`i`,`c`)`
:: 第 i 行を c 倍する
126. `rowa(`*matrix*`,`i`,`j`,`c`)`
:: 第 i 行に第 j 行の c 倍を加える
127. `colx(`*matrix*`,`i`,`j`)`
:: 第 i 列と第 j 列を交換する
128. `colm(`*matrix*`,`i`,`c`)`
:: 第 i 列を c 倍する.
129. `cola(`*matrix*`,`i`,`j`,`c`)`
:: 第 i 列に第 j 列の c 倍を加える.

- 行列の基本変形を行うための函数である.
- 行列が破壊されることに注意する.

```
[0] A=newmat(3,3,[[1,2,3],[4,5,6],[7,8,9]]);
[ 1 2 3 ]
[ 4 5 6 ]
[ 7 8 9 ]
[1] rowx(A,1,2)$
[2] A;
[ 1 2 3 ]
[ 7 8 9 ]
[ 4 5 6 ]
[3] rowm(A,2,x);
[ 1 2 3 ]
[ 7 8 9 ]
[ 4*x 5*x 6*x ]
[4] rowa(A,0,1,z);
[ 7*z+1 8*z+2 9*z+3 ]
[ 7 8 9 ]
[ 4*x 5*x 6*x ]
```

130. pari(adj,*mat*)
:: 行列 *mat* の余因子行列（元の行列との積が行列式となる）を返す

```
[0] pari(adj,mat([1,1,1],[x,1,1],[1,y,1]));
[ -y+1 y-1 0 ]
[ -x+1 0 x-1 ]
[ y*x-1 -y+1 -x+1 ]
```

131. pari(trace,*mat*)
:: 行列 *mat* の trace を返す

```
[0] pari(trace,mat([1,1,1],[x,1,1],[1,y,1]));
3
```

132. pari(signat,*mat*)
:: 実対称行列の符号を返す

```
[0] pari(signat,mat([1,2,3],[1,1,1],[3,2,1]));
[ 2 1 ]
```

133. pari(indexrank,*mat*)
:: 行列 *mat* の階数に等しいサイズの正則小行列の一つの行と列の位置を返す

```
[0] pari(indexrank,mat([1,2,3],[2,4,6],[1,3,2],[2,5,5],[0,1,-1]));
[ [ 1 3 ] [ 1 2 ] ]
```

134. pari(supplement,*mat*)
:: 列ベクトルが一次独立な行列 *mat* の右に列を補って正則正方行列を作る

```
[0] A=mat([1,1],[1,1],[1,0],[0,1]);
```

```
[ 1 1 ]
[ 1 1 ]
[ 1 0 ]
[ 0 1 ]
[1] pari(supplement,A);
[ 1 1 0 0 ]
[ 1 1 1 0 ]
[ 1 0 0 0 ]
[ 0 1 0 1 ]
```

135. pari(hess,*mat*)
:: 正方行列をヘッセンベルグ行列に変換する
ヘッセンベルグ行列（対角成分の 2 つより下の成分が 0 の行列）に相似変換する.

```
[0]pari(hess,mat([1,2,3],[4,5,6],[7,8,9]));
[ 1 29/7 2 ]
[ 7 95/7 8 ]
[ 0 54/49 3/7 ]
```

136. pari(eigen,*mat*[,*prec*])
:: 数を成分とする行列の固有ベクトルを返す

```
[0] A=os_md.s2m("(13)2,(10)(14)");
[ 13 2 ]
[ 10 14 ]
[1] pari(eigen,A);
[ -1/2 2/5 ]
[ 1 1 ]
```

137. pari(jacobi,*mat*[,*prec*])
:: 実対称行列の固有値と固有ベクトルを返す

```
[0] pari(jacobi,mat([1,2],[2,1]));
[ [ 3.000000000000000000 -1.000000000000000000 ]
  [ 0.70710678118654752449 -0.70710678118654752438 ]
  [ 0.70710678118654752438 0.70710678118654752449 ] ]
```

3.3.5 文字列に関する演算

138. rtostr(*obj*)
:: *obj* を文字列に変える
- 任意のオブジェクト *obj* を文字列に変える.
- 整数などを文字列に変換して変数名と結合することにより, 添字付きの不定元を生成する場合に多く用いられる.
- 逆に, 文字列を不定元に変換する時には, strtov() (cf. makev()) を用いる.

```
[0] A=afo;
afo
[1] type(A);
2
```

```
[2] B=rtostr(A);
afo
[3] type(B);
7
[4] B+"1";
afo1
```

139. strtov(*str*)
:: *str* (文字列) を不定元に変える
- 不定元として変換可能な文字列とは, 英小文字で始まり, 英字, 数字および記号 _ で作られる文字列である.
- rtostr() と組合せて, プログラム中で自動的に不定元を生成したい時に用いられる.

```
[0] A="afo";
afo
[1] for (I=0;I<3;I++) {B=strtov(A+rtostr(I)); print([B,type(B)]);}
[afo0,2]
[afo1,2]
```

140. eval_str(*str*)
:: *str* (文字列) を評価する.
- Asir の parser が受理可能な文字列を評価してその結果を返す.
- 評価可能な文字列は, 式を表すものに限る.
- 論理的には rtostr() の逆函数となる.
- Mathematica の文字列の評価には evalma() を用いる.

```
[0] eval_str("1+2");
3
[1] fctr(eval_str(rtostr((x+y)^10)));
[[1,1],[x+y,10]]
```

141. strtoascii(*str*)
:: 文字列をアスキーコード（1 以上 255 以下の整数）のリストで表す
strtoascii() は文字列を整数のリストに変換する. 各整数は文字列のアスキーコードを表す.
142. asciitostr(*list*)
:: アスキーコードの列を文字列に変換する
asciitostr() は asciitostr() の逆函数である.

```
[0] strtoascii("abcxyz");
[97,98,99,120,121,122]
[1] asciitostr(@);
abcxyz
[2] asciitostr([256]);
asciitostr : argument out of range
return to toplevel
```

143. str_len(*str*)
:: 文字列の長さを返す
144. str_chr(*str*,*start*,*c*)
:: 文字が最初に現れる位置を返す

- *str* の start 番目の文字からスキャンして 最初に *c* の最初の文字が現れた位置を返す. 文字列の先頭は 0 番目とする
- str_char() が上位互換函数. str_str() はより多機能な函数.

```
[0] Line="123 456 (x+y)^3";
123 456 (x+y)^3
[1] Sp1 = str_chr(Line,0," ");
3
[2] D0 = eval_str(sub_str(Line,0,Sp1-1));
123
[3] Sp2 = str_chr(Line,Sp1+1," ");
7
[4] D1 = eval_str(sub_str(Line,Sp1+1,Sp2-1));
456
[5] C = eval_str(sub_str(Line,Sp2+1,str_len(Line)-1));
x^3+3*y*x^2+3*y^2*x+y^3
```

145. sub_str(*str*,*start*,*end*)
:: 部分文字列を返す
- str_cut() はより多機能な函数.

3.3.6 構造体に関する函数

146. newstruct(*name*)
:: 構造体名が name の構造体を生成する
- あらかじめ, *name* なる構造体が定義されていなければならない.
- 構造体の各メンバは演算子 -> により名前でアクセスする. メンバが構造体の場合, 更に -> による指定を続けることができる

```
[0] struct list {h,t};
0
[1] A=newstruct(list);
{0,0}
[2] A->t = newstruct(list);
{0,0}
[3] A;
{0,{0,0}}
[4] A->h = 1;
1
[5] A->t->h = 2;
2
[6] A->t->t = 3;
3
[7] A;
{1,{2,3}}
```

147. arfreg(*name*,*add*,*sub*,*mul*,*div*,*pwr*,*chsgn*,*comp*)
:: 構造体に体する基本演算を登録する
- 登録したくない基本演算に対しては引数に 0 を与える. これによって一部の演算のみを利用するこ

とができる.

- それぞれの函数の仕様は次の通りである.

add(A,B)	A+B
sub(A,B)	A-B
mul(A,B)	A*B
div(A,B)	A/B
pwr(A,B)	A^B
chsgn(A)	-A
comp(A,B)	1,0,-1 according to the result of a comparison between A and B.

```
% cat test
struct a {id,body}\$
def add(A,B)
{
  C = newstruct(a);
  C->id = A->id; C->body = A->body+B->body;
  return C;
}

def sub(A,B)
{
  C = newstruct(a);
  C->id = A->id; C->body = A->body-B->body;
  return C;
}

def mul(A,B)
{
  C = newstruct(a);
  C->id = A->id; C->body = A->body*B->body;
  return C;
}

def div(A,B)
{
  C = newstruct(a);
  C->id = A->id; C->body = A->body/B->body;
  return C;
}

def pwr(A,B)
{
  C = newstruct(a);
  C->id = A->id; C->body = A->body^B;
  return C;
```

```
}

def chsgn(A)
{
  C = newstruct(a);
  C->id = A->id; C->body = -A->body;
  return C;
}

def comp(A,B)
{
  if ( A->body > B->body )
    return 1;
  else if ( A->body < B->body )
    return -1;
  else
    return 0;
}

arfreg("a",add,sub,mul,div,pwr,chsgn,comp)$
end$
% asir
This is Risa/Asir, Version 20000908.
Copyright (C) FUJITSU LABORATORIES LIMITED.
1994-2000. All rights reserved.
[0] load("./test")$
[11] A=newstruct(a);
{0,0}
[12] B=newstruct(a);
{0,0}
[13] A->body = 3;
3
[14] B->body = 4;
4
[15] A*B;
{0,12}
```

148. str_type(*name*|*object*)

:: 構造体の識別番号を取得する

- 名前が *name* である構造体, または *object* の指す構造体の識別番号を取得する. エラーのときは -1 を返す

```
[0] struct list {h,t};
0
[1] A=newstruct(list);
```

```
{0,0}
[2] struct_type(A);
3
[3] struct_type("list");
3
```

3.3.7 入出力

149. `end`
150. `quit`
:: 現在読み込み中のファイルを閉じる．トップレベルにおいてはセッションを終了することになる
- `end`, `quit` ともに無引数の函数であるが，‘`()`’ なしで呼び出すことができる．いずれも現在読み込み中のファイルを閉じる．これは，トップレベルにおいてはセッションを終了させることになる．
- ファイルの場合，ファイルの終端まで読めば，自動的にファイルは閉じられるが，トップレベルの場合プロンプトが出ないまま，入力待ちになるので，ファイルの終端には `end$` を書くのが望ましい．

```
[0] quit;
%
```

151. `load("`*filename*`")`
:: *filename* を読み込む
- 実際のプログラムの書き方は，**ユーザ言語 Asir** を参照．テキストファイルを読み込む場合，cpp を通すので，`C` のプログラム同様 `#include`, `#define` を使うことができる．
- 指定したファイルが存在した時には 1 を返し，存在しなかった時は 0 を返す．
- ファイル名が ‘/’ で始まる場合は絶対パス，‘.’ で始まる場合はカレントディレクトリからの相対パスと見なされる．それ以外の場合，環境変数 `ASIRLOADPATH` に設定されているディレクトリを左から順にサーチする．それらに該当するファイルが存在しない場合，標準ライブラリディレクトリ (あるいは環境変数 `ASIR_LIBDIR` に設定されているディレクトリ) もサーチする．Windows 版の場合，`ASIR_LIBDIR` が設定されていない場合には，`get_rootdir()/lib` をサーチする．
- 読み込むファイルの最後に，`end$` がないと `load()` 終了後にプロンプトがでないが，実際には入力を受け付ける．しかし，混乱を招くおそれがあるのでファイルの最後に `end$` を書いておくことが望ましい．(`end;` でもよいが，`end` が返す値 0 が表示されるため，`end$` をお勧めする．)
- Windows 版もディレクトリのセパレータとして ‘`/`’ を用いる

152. `which("`*filename*`")`
:: 引数 filename に対し，`load()` が読み込むパス名を返す
- `load()` がファイルをサーチする手順に従ってサーチし，ファイルが存在する場合にはパス名を文字列として，存在しない場合には 0 を返す．
- サーチの手順については `load()` を参照．
- Windows 版もディレクトリのセパレータとして ‘`/`’ を用いる

```
[0] which("gr");
./gb/gr
[1] which("/usr/local/lib/gr");
0
[2] which("/usr/local/lib/asir/gr");
/usr/local/lib/asir/gr
```

153. `output([")`*filename*`"])`
:: 以降の出力先を *filename* または標準出力に切替える
- `Asir` の出力を標準出力から，ファイルへの出力に切替える．なお，ファイル出力の間は，標準出力

にはキーボードからの入力以外, 出力されない.

- 別のファイル出力に切替える時には, 再び output("*filename*") を実行する. 又, ファイル出力を終了し標準出力に戻りたい時には, 引数なしで output() を実行する.
- 指定したファイル *filename* が存在した時は, そのファイルの末尾に追書きされ, 存在しなかった時には, 新たにファイルを作成し, そこに書き込まれる.
- ファイルネームを " " ダブルクォートなしで指定をしたり, ユーザが, 書き込めないファイルを指定したりすると, エラーによりトップレベルに戻る.
- 入力したものも込めてファイルに出力したい場合には, ctrl("*echo*",1) を実行した後でファイル出力に切替えれば良い. 計算時間など, 標準エラー出力に書き出されるものはファイルには書き出されない. 函数形式, 未定係数 (vtype() 参照) を含まない数式のファイルへの読み書きは, bload(), bsave() を使うのが, 時間, 空間ともに効率がよい.
- Windows 版もディレクトリのセパレータとして '/' を用いる.

```
[0] output("afo");
fctr(x^2-y^2);
print("afo");
output();
1
[1] quit;
% cat afo
1
[2] [[1,1],[x+y,1],[x-y,1]]
[3] afo
0
[4]
```

154. bsave(*obj*,"*filename*")
:: *filename* に *obj* をバイナリ形式で書き込む

155. bload("*filename*")
:: *filename* から数式をバイナリ形式で読み込む

- *obj* は 函数形式, 未定係数を含まない任意の数式
- bsave() は内部形式をほぼそのままバイナリ形式でファイルに書き込む. bload() は, bsave() で書き込んだ数式を読み込んで内部形式に変換する. 現在のインプリメンテーションの制限により, 函数形式, 未定係数 (vtype()) を含まないリスト, 配列などを含む任意の数式をファイルに保存することができる. output などで保存した場合, 読み込み時にパーザが起動されるが, bsave() で保存したものを bload() で読む場合, 直接内部形式が構成できるため, 時間的, 空間的に効率がよい.
- 多項式の場合, 書き込み時と読み込み時で変数順序が異なる場合があるが, その場合には, 自動的に現在の変数順序における内部形式に変換される.
- Windows 版もディレクトリのセパレータとして '/' を用いる.

```
[0] A=(x+y+z+u+v+w)^20$
[1] bsave(A,"afo");
1
[2] B = bload("afo")$
[3] A == B;
1
[4] X=(x+y)^2;
x^2+2*y*x+y^2
```

```
[5] bsave(X,"afo")$
[6] quit;
% asir
[0] ord([y,x])$
[1] bload("afo");
y^2+2*x*y+x^2
```

156. print(*obj*[,*nl*])
:: *obj* を評価して表示する
   - 第 2 引数がないか, または 0, 2 以外の場合, 改行する. 第 2 引数が 0 の場合, 改行せず, 出力はバッファに書き込まれ, バッファはフラッシュされない. 第 2 引数が 2 の場合, 改行しないがバッファはフラッシュされる. この函数の戻り値は 0 であるから, print(); で実行すると, 出力の後に 0 が返される. print()$ とすれば, 最後の 0 は出力されない.
   - 複数の *obj* を同時に出力したい時は *obj* をリストにするとよい. mycat(), mycat0() を参照.

```
[0] def cat(L) { while ( L != [] ) { print(car(L),0); L = cdr(L);}
print(""); }
[1] cat([xyz,123,"gahaha"])$
xyz123gahaha
```

157. printf(*format*[,*args*])
158. fprintf(*fd*,*format*[,*args*])
159. sprintf(*format*[,*args*])
:: C に似たプリント関数（実験的仕様関数）
   - printf は書式文字列 *format* にしたがい, オブジェクト *args* を標準出力に書き出す.
   - fprintf は結果を, ファイル記述子 *fd* の指すファイルに書き出す.
   - sprintf は結果を文字列で返し, 標準出力には書き出さない.
   - 書式文字列の中で %a (any) が利用可能. *args* の個数は書式文字列の中の %a の個数に等しくすること. ファイル記述子は openfile() 関数を用いて得ること.

```
[0] printf("%a: rat = %a\n",10,x^2-1)$
10: rat = x^2-1
[1] S=sprintf("%a: rat = %a",20,x^2-1)$
[2] S;
20: rat = x^2-1
[3] Fd=open_file("hoge.txt","w");
0
[4] fprintf(Fd,"Poly=%a\n",(x-1)^3)$
[5] close_file(Fd)$
[6] quit;

$ cat hoge.txt
Poly=x^3-3*x^2+3*x-1
```

160. access(*file*)
:: *file* の存在をテストし，存在すれば 1，存在しなければ 0 を返す
161. remove_file(*file*)
:: file を消去する
162. open_file("*filename*" [,"*mode*"])

:: *filename* をオープンする

163. `close_file(`*num*`)`
:: 識別子 *num* のファイルをクローズする

164. `get_line(`[*num*]`)`
:: 識別子 *num* のファイルから 1 行読む

165. `get_byte(`*num*`)`
:: 識別子 *num* のファイルから 1 バイト読む

166. `put_byte(`*num*`, `*c*`)`
:: 識別子 *num* のファイルに 1 バイト *c* を書く

167. `purge_stdin()`
:: 標準入力のバッファをクリアする

- `open_file()` はファイルをオープンする. *mode* 指定がない場合は読み出し用, *mode* 指定がある場合には, *C* の標準入出力関数 `fopen()` に対するモード指定とみなす. たとえば新規書き込み用の場合 `"w"`, 末尾追加の場合 `"a"` など. 成功した場合, ファイル識別子として非負整数を返す. 失敗の場合エラーとなる. 不要になったファイルは `close_file()` でクローズする. 特別なファイル名 `unix://stdin`, `unix://stdout`, `unix://stderr` を与えるとそれぞれ標準入力, 標準出力, 標準エラー出力をオープンする. この場合モード指定は無視される.
- `get_line()` は現在オープンしているファイルから 1 行読み, 文字列として返す. 引数がない場合, 標準入力から 1 行読む.
- ファイルの終りまで読んだ後に `get_line()` が呼ばれた場合, 整数の 0 を返す.
- `get_byte()` は現在オープンしているファイルから 1 バイト読み 整数として返す.
- `put_byte()` は現在オープンしているファイルに 1 バイト書き, そのバイトを整数として返す.
- 読み出した文字列は, 必要があれば `sub_str()` などの文字列処理関数で加工したのち `eval_str()` により内部形式に変換できる.
- `purge_stdin()` は, 標準入力バッファを空にする. 関数内で `get_line()` により標準入力から文字列を受け取る場合, 既にバッファ内に存在する文字列による誤動作を防ぐためにあらかじめ呼び出す.

```
[0] Id = open_file("test");
0
[1] get_line(Id);
12345

[2] get_line(Id);
67890

[3] get_line(Id);
0
[4] type(@@);
0
[5] close_file(Id);
1
[6] open_file("test");
1
[7] get_line(1);
12345
```

```
[8] get_byte(1);
54                    /* the ASCII code of '6' */
[9] get_line(1);
7890                  /* the rest of the last line */
[10] def test() { return get_line(); }
[11] def test1() { purge_stdin(); return get_line(); }
[12] test();
                      /* a remaining newline character has been read */
                      /* returns immediately */
[13] test1();
123;                  /* input from a keyboard */
123;                  /* returned value */
```

3.3.8 型や函数，モジュールに関わる函数

168. `type(`*obj*`)`
:: *obj* **の型**を求める函数
−1: 空　0:0　1: 数　2: 多項式　3: 有理式　4: リスト　5: ベクトル　6: 行列　7: 文字列　8: 構造体
9: 分散表現多項式　10: 符号なしマシン 32bit 整数　11: エラーオブジェクト　12: $GF(2)$ 上の行列
13: MATHCAP オブジェクト　14: first order formula　15: matrix over $GF(p)$　16: ビット列
- 数の型を求めるには, `ntype()` を用いる. 不定元の型を求めるには, `vtype()` を用いる.

169. `ntype(`*num*`)`
:: *num*（数）**の 型** (整数) を返す. 数の型の値は
0: 有理数　1: 倍精度浮動小数　2: 代数的数　3: 任意精度浮動小数 (bigfloat)　4: 複素数
5: 有限体の元　6: 大標数素体の元　7: 標数 2 の有限体の元
- `newalg(x^2+1)` で生成した数と, 虚数単位 `@i` は, 異なるものとして扱われる.
- 代数的数に関しては, See section 代数的数に関する演算.

```
[0] [10/37,ntype(10/37)];
[10/37,0]
[1] [10.0/37.0,ntype(10.0/37.0)];
[0.27027,1]
[2] [newalg(x^2+1)+1,ntype(newalg(x^2+1)+1)];
[(#0+1),2]
[3] [eval(sin(@pi/6)),ntype(eval(sin(@pi/6)))];
[0.49999999999999999991,3]
[4] [@i+1,ntype(@i+1)];
[(1+1*@i),4]
```

170. `vtype(`*var*`)`
:: *var*（不定元）**の型** (整数) を返す
- *var* (不定元) の型の値は次の通り.
  0　一般不定元 (`a,b,x,afo,bfo,`..., etc)
  1　`uc()` で生成された不定元 (`_0, _1, _2,` ... etc)
  2　函数形式 (`sin(x), log(a+1), acosh(1), @pi, @e,` ... etc)
  3　函数子 (組み込み函数子, ユーザ定義函数子, 初等函数子：`sin, log,` ... etc)

171. `call(`*name*`,`*args*`)`
:: 函数 *name* を呼び出す

- 函数子に関しては, vtype() を参照.
- 引数の個数があらかじめ分かっているときは *(*name*)() の形式が使える.

```
[0] A=igcd;
igcd
[1] call(A,[4,6]);
2
[2] (*A)(4,6);
2
```

172. functor(*func*)
:: *func* の函数子を取り出す
173. args(*func*)
:: *func* の引数リストを取り出す
174. funargs(*func*)
:: cons(functor(*func*),args(*func*)) を返す
- 函数形式に関しては, vtype() を参照.
- 函数形式 func の函数子, 引数リストを取り出す.
- 逆に, 取り出した函数子を値に持つプログラム変数を F とすれば (*F)(*x*) で x を引数とする函数呼び出しまたは函数形式が入力できる

```
[0] functor(sin(x));
sin
[1] args(sin(x));
[x]
[2] funargs(sin(3*cos(y)));
[sin,3*cos(y)]
[3] for (L=[sin,cos,tan];L!=[];L=cdr(L)) {A=car(L);
print(eval((*A)(@pi/3)));}
0.86602540349122136831
0.5000000002
1.7320508058
```

175. module_list()
:: 定義済みのモジュールのリストを得る

```
[0] module_list();
[gr,primdec,bfct,sm1,gnuplot,tigers,phc]
```

176. module_definedp(*name*)
:: モジュール *name* の存在をテストする
- モジュール *name* が存在すれば 1, 存在しなければ 0 を返す

```
[0] module_definedp("gr");
1
```

177. remove_module(*name*)
:: モジュール *name* を削除する
- 削除に成功すれば 1, 失敗すれば 0 を返す.

```
[0] remove_module("gr");
```

```
1
```

3.3.9 数値函数

178. `dacos(`*num*`)`
:: 函数値 Arccos(*num*) を倍精度浮動小数で求める
179. `dasin(`*num*`)`
:: 函数値 Arcsin(*num*) を倍精度浮動小数で求める
180. `datan(`*num*`)`
:: 函数値 Arctan(*num*) を倍精度浮動小数で求める
181. `dcos(`*num*`)`
:: 函数値 cos(*num*) を倍精度浮動小数で求める
182. `dsin(`*num*`)`
:: 函数値 sin(*num*) を倍精度浮動小数で求める
183. `dtan(`*num*`)`
:: 函数値 tan(*num*) を倍精度浮動小数で求める
    - これらの函数は C 言語の標準数学ライブラリを用いる．したがって, 計算結果はオペレーティングシステムとコンパイラに依存する．
    - 複素数に対しては正しくない結果を返すので注意しなければならない．
    - `@pi` などのシンボルを引数に与えることはできない．

```
[0] 4*datan(1);
3.14159
```

184. `dabs(`*num*`)`
:: 絶対値 $|num|$ を倍精度浮動小数で求める
185. `dexp(`*num*`)`
:: 函数値 exp(*num*) を倍精度浮動小数で求める
186. `dlog(`*num*`)`
:: 対数値 log(*num*) を倍精度浮動小数で求める
187. `dsqrt(`*num*`)`
:: 平行根 $\sqrt{num}$ を倍精度浮動小数で求める
    - これらの函数は C 言語の標準数学ライブラリを用いる．したがって, 計算結果はオペレーティングシステムとコンパイラに依存する．
    - `dabs()` と `dsqrt()` を除き、複素数に対しては正しくない結果を返すので注意しなければならない．
    - `@pi` などのシンボルを引数に与えることはできない

```
[0] dexp(1);
2.71828
```

188. `ceil(`*num*`)`
:: *num* 以上の最小の整数を求める

```
[0] dceil(1.1);
1
```

189. `dceil(`*num*`)`
:: *num* 以上の最小の整数を求める（`ceil()` の別名）
190. `floor(`*num*`)`
:: *num* 以下の最大の整数を求める
191. `dfloor(`*num*`)`
:: *num* 以下の最大の整数を求める（`floor()` の別名）

192. `rint(`*num*`)`
:: *num* を整数にまるめる
193. `drint(`*num*`)`
:: *num* を整数にまるめる（`rint()` の別名）
194. `pari(sqrt,`*num*`[,`*prec*`])`
:: （複素）数 *num* の平方根を与える

```
[0] pari(sqrt,2);
1.4142135623730950488
[1] pari(sqrt,1+@i);
(1.0986841134678099659570+0.45508986056222734133*@i)
```

195. `pari(arg,`*num*`[,`*prec*`])`
:: 複素数 *num* の偏角を与える（$-\pi <$ 戻り値 $\leq \pi$）

```
[0] pari(arg,1+@i);
0.78539816339744830961566084581
```

196. `pari(gamma,`*num*`[,`*prec*`])`
:: ガンマ関数（*num* は複素数でもよい）
197. `pari(gamh,`*num*`[,`*prec*`])`
:: $\Gamma(num + \frac{1}{2})$（*num* は複素数でもよい）

```
[0] pari(gamma,5);
23.999999999999999826
[1] pari(gamma,1+@i);
(0.49801566811835604271-0.15494982830181068512*@i)
```

198. `pari(lngamma,`*num*`[,`*prec*`])`
:: $\log(\Gamma(x))$

```
[0] pari(lngamma,1000);
5905.22042320918121172113
```

199. `pari(psi,`*num*`[,`*prec*`])`
:: digamma 関数
$\psi(x) = \frac{\Gamma'(x)}{\Gamma(x)}$

```
[0] pari(psi,1);
-0.57721566490153286077
[1] pari(psi,1+@i);
(0.094650320622476976191+1.0766740474685811725 09*@i)
```

200. `pari(erfc,`*num*`[,`*prec*`])`
:: 相補誤差関数 (complementary error function)
$\mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt = 1 - \mathrm{erc}(x)$

```
[0] pari(erfc,1);
0.15729920705028513139
```

201. `pari(dilog,`*num*`[,`*prec*`])`
:: dilogarithm 関数
$\mathrm{Li}_2(x) = \sum_{k=1}^{\infty} \frac{x^k}{k^2}$

```
[0] pari(dilog,1);
1.64493406684822643609
```

202. `pari(eta,`τ`[,`*prec*`])`
:: Dedekind の Eta 関数 $\eta(\tau)$ で τ は虚部が正の複素数
$\eta(\tau) = e^{\frac{\pi i \tau}{12}} \prod_{m=1}^{\infty}(1 - e^{2\pi i m \tau})$

```
[0] pari(eta,1/2*@i+1/3);
(1.022540605006162884497644506602 2-0.035807228435521009596*@i)
```

203. `pari(wp,`τ`[,`*prec*`])`
:: Weber 関数 $\mathfrak{f}(\tau)$ で τ は虚部が正の複素数
$\mathfrak{f}(\tau) = e^{\frac{-2\pi i \tau}{48}} \frac{\eta(\frac{\tau+1}{2})}{\eta(x)}$
204. `pari(wp2,`τ`[,`*prec*`])`
:: Weber 関数 $\mathfrak{f}_2(\tau)$ で τ は虚部が正の複素数
$\mathfrak{f}_2(\tau) = \sqrt{2}\frac{\eta(2\tau)}{\eta(\tau)}$

```
[0] pari(wf,1/2*@i+1/3);
(1.175500458021642396 12+0.13899974799314382921*@i)
[1] pari(wf2,1/2*@i+1/3);
(1.204439090310062473 16+0.14997728710366899256*@i)
```

205. `pari(jell,`τ`[,`*prec*`])`
:: Elliptic j-invariant $j(\tau)$ で τ は虚部が正の複素数
$j(\tau) = 1728\frac{g_2^2}{g_2^3 - 27g_3^2}$
$g_2 = 60\sum_{(m,n)\neq(0,0)}(m+n\tau)^{-4}, \quad g_3 = 140\sum_{(m,n)\neq(0,0)}(m+n\tau)^{-6}$

```
[0] pari(jell,1/3+1/2*@i);
(6087.612142594022178521 12-2773.380470131800678235 12*@i)
```

206. `pari(zeta,`s`[,`*prec*`])`
:: Riemann の ζ 関数
$\zeta(s) = \sum_{n=1}^{\infty} n^{-s}$

```
[0] pari(zeta,2);
1.64493406684822643609
```

3.3.10 描画函数

207. `ifplot(`*func* `[,`*geometry*`] [,`*xrange*`] [,`*yrange*`] [,`*id*`] [,`*name*`])`
:: 2 変数関数の実数上での零点を表示する
208. `conplot(`*func* `[,`*geometry*`] [,`*xrange*`] [,`*yrange*`] [,`*zrange*`] [,`*id*`] [,`*name*`])`
:: 2 変数関数の実数上での等高線を表示する
209. `plot(`*func* `[,`*geometry*`] [,`*xrange*`] [,`*id*`] [,`*name*`])`
:: 1 変数関数のグラフを表示する
210. `polarplo(`*func* `[,`*geometry*`] [,`*thetarange*`] [,`*id*`] [,`*name*`])`
:: 極形式 $r =$*func*(θ) で与えられた曲線を表示する
211. `plotover(`*func*`,`*id*`,`*number*`)`
:: すでに存在しているウィンドウへ描画する
   - これらは OpenXM サーバとして実現されている．UNIX 上では `'ox_plot'` が, Windows 上では `'engine'` がこれらの機能を提供しており, これらは `Asir` の標準ライブラリディレクトリにある．アクティブな `'ox_plot'` の `id` が *id* として指定された場合, そのサーバが用いられる．*id* の

指定がない場合には, 起動されているサーバのうち, ‘ox_plot’ があればそのサーバが用いられる. ‘ox_plot’ が起動されていない場合には, ox_launch_nox() が自動的に実行されて, ‘ox_plot’ が立ち上がり, それが用いられる. 引数の内, func は必須である. その他の引数はオプションである. オプションの形式およびそのデフォルト値 (カッコ内) は次の通り.

geometry ウィンドウのサイズをドット単位で [x,y] で指定する. ([300,300].)

xrange yrange 変数の範囲の指定で, [v,vmin,vmax] で指定する. (いずれの変数も [v,-2,2].) この指定がない場合, *func* に含まれる変数の内変数順序の上の変数が ‘x’, 下の変数が ‘y’ として扱われる. これを避けるためには *xrange*, *yrange* を指定する. また, *func* が 1 変数の場合, これらの指定は必須となる.

zrange conplot() の場合のみ指定できる. 形式は [v,vmin,vmax [,step]] で, step が指定された場合には, 等高線の間隔が (vmax-vmin)/step となる. ([z,-2,2,16].)

id 遠隔プロセスの番号, すなわち ox_launch() が返した番号を指定する. (一番・近に作られ, かつアクティブなプロセスに対応する番号.)

name ウィンドウの名前. (Plot.) 生成されたウィンドウのタイトルは *name*:n/m となる. これは, プロセス番号 n のプロセスの, m 番のウィンドウを意味する. この番号は, plotover() で用いられる.

- 一つのプロセス上で描画できるウィンドウの数は最大 128 個である.
- plotover() は, 指定したウィンドウ上に, 引数である 2 変数多項式の零点を上書きする.
- 描画終了後のウィンドウ上で, マウスの左ボタンを押しながらのドラッグで範囲を指定しボタンを離すと新たなウィンドウが生成され, 指定した範囲が拡大して表示される. ドラッグは左上から右下へと行う. ドラッグを始めた後キャンセルする場合は, マウスポインタを始点の上か左に持っていってボタンを離せばよい. 新しいウインドウの形は, 指定領域と相似で, 最大辺が, 元のウィンドウの最大辺と一致するように定められる. 以下で説明する precise が on の場合, 選択した領域が同一 window 上で書き直される.
- ウィンドウ内で右ボタンを押すと, その点の座標がウィンドウの下部に表示される.
- conplot() で生成したウィンドウにおいて, ウィンドウの右側のマーカを中ボタンでドラッグすると, 対応する等高線の色が変わり, 右上のウィンドウに対応するレベルが表示される.
- UNIX 版ではいくつかのボタンによりいくつかの設定変更, 操作ができる. UNIX 版では次のボタンがある.

  quit window を破壊する. 計算を中断する場合, ox_reset() を用いる.

  wide (トグル) 現在の表示部分を縦横各 10 倍した領域を表示する. 現在表示されている範囲はこの表示において中央部に長方形で示される. この表示で範囲指定を行うと, その範囲が新しいウィンドウに描画される.

  precise (トグル) 選択領域を, 整数演算により, より正確に再描画する. これは, *func* が有理数係数の 2 変数多項式の場合にのみ有効である. このモードでは Sturm 列と二分法により, 区間内の零点の個数を正確に求めていくもので, デフォルトの計算法よりも正確な描画が期待できる. ただし, 描画時間は余計にかかる場合が多い. この説明から明らかなように, この機能は有理数係数の多項式の描画に対してのみ有効である ($(x^2+y^2-1)^2$ の描画で試してみよ.)

  formula 対応する式を表示する.

  noaxis (トグル) 座標軸を消す.
- ‘ox_plot’ が起動されるマシンによっては, スタックを大量に使用するものもあるため, ‘.cshrc’ でスタックサイズを大きめ (16MB 程度) に指定しておくのが安全である. スタックサイズは limit stacksize 16m などと指定する.
- X では、ウインドウの各部分について resource により色付けや、ボタンの形を変えることができる. resource の指定の仕方は以下の通り. (デフォルトを示しておく) plot*form*shapeStyle は、rectangle, oval, ellipse, roundedRectangle が、指定できる.

```
plot*background:white
plot*form*shapeStyle:rectangle
plot*form*background:white
```

```
plot*form*quit*background:white
plot*form*wide*background:white
plot*form*precise*background:white
plot*form*formula*background:white
plot*form*noaxis*background:white
plot*form*xcoord*background:white
plot*form*ycoord*background:white
plot*form*level*background:white
plot*form*xdone*background:white
plot*form*ydone*background:white
```

212. open_canvas(*id*[,*geometry*])
:: 描画用ウィンドウ (キャンバス) を生成する
213. clear_canvas(*id*,*index*)
:: キャンバスをクリアする
214. draw_obj(*id*,*index*,*pointorsegment* [,*color*])
:: キャンバス上に点または線分を描画する
215. draw_string(*id*,*index*,[*x*,*y*],*string* [,*color*])
:: キャンバス上に文字列を描画する

- これらは OpenXM サーバ 'ox_plot' (Windows 上では 'engine') により提供される.
- open_canvas() は, 描画用のウィンドウ (キャンバス) を生成する. *geometry* によりウィンドウのサイズを pixel 単位で [*x*,*y*] で指定する. default size は [300,300]. キャンバスの識別子として, 整数値を OpenXM サーバのスタックに push する. この識別子は draw_obj の呼び出しに必要であり, ox_pop_cmo により取り出して保持する必要がある.
- clear_canvas() は, サーバ id id, キャンバス id index で指定されるキャンバスをクリアする.
- draw_obj は, サーバ id id, キャンバス id index で指定されるキャンバスに点または線分を描画する. *pointorsegment* が [*x*,*y*] の場合点の座標, [*x*,*y*,*u*,*v*] の場合 [*x*,*y*], [*u*,*v*] を結ぶ線分を表すと見なされる. キャンバスの座標は, 左上隅を原点として横方向に第一座標, 縦方向に第二座標をとる. 値は pixel 単位で指定する. color の指定がある場合, color/65536 mod 256, color/256 mod 256, color mod 256 をそれぞれ Red, Green, Blue の値 (最大 255) とみなす.
- draw_string は, サーバ id id, キャンバス id index で指定されるキャンバスに文字列を描画する. 位置は [*x*,*y*] により指定する.

```
[182] Id=ox_launch_nox(0,"ox_plot");
0
[183] open_canvas(Id);
0
[184] Ind=ox_pop_cmo(Id);
0
[185] draw_obj(Id,Ind,[100,100]);
0
[186] draw_obj(Id,Ind,[200,200],0xffff);
0
[187] draw_obj(Id,Ind,[10,10,50,50],0xff00ff);
0
[187] draw_string(Id,Ind,[100,50],"hello",0xffff00);
0
[189] clear_canvas(Id,Ind);
```

0

3.3.11 有限体に関する演算

216. `setmod_ff([p|defpoly2])` `setmod_ff([defpolyp,p])` `setmod_ff([p,n])`
:: 有限体の設定, 設定されている有限体の法, 定義多項式の表示
p 素数
defpoly2 $GF(2)$ 上既約な 1 変数多項式
defpolyp $GF(p)$ 上既約な 1 変数多項式
n 拡大次数
- 引数が正整数 p の時, $GF(p)$ を基礎体として設定する.
- 引数が多項式 *defpoly2* の時, $GF(2^{deg}(defpoly2 \mod 2)) = GF(2)[t]/(defpoly2(t) \mod 2)$ を基礎体として設定する.
- 引数が *defpolyp* と p の時, $GF(p^{deg}(defpolyp))$ を基礎体として設定する.
- 引数が p と n の時, $GF(p^n)$ を基礎体として設定する. p^n は 2^{29} 未満でなければならない. また, p が 2^{14} 以上のとき, n は 1 でなければならない.
- 無引数の時, 設定されている基礎体が $GF(p)$ の場合 p, $GF(2^n)$ の場合定義多項式を返す. 基礎体が `setmod_ff(defpoly,p)` で定義された $GF(p^n)$ の場合, `[defpoly,p]` を返す. 基礎体が `setmod_ff(p,n)` で定義された $GF(p^n)$ の場合, `[p,defpoly,prim_elem]` を返す. ここで, *defpoly* は, n 次拡大の定義多項式, *prim_elem* は, $GF(p^n)$ の乗法群の生成元を意味する.
- $GF(2^n)$ の定義多項式は, $GF(2)$ 上 n 次既約ならなんでも良いが, 効率に影響するため, `defpoly_mod2()` で生成するのがよい.

```
[0] defpoly_mod2(100);
x^100+x^15+1
[1] setmod_ff(@@);
x^100+x^15+1
[2] setmod_ff();
x^100+x^15+1
[3] setmod_ff(x^4+x+1,547);
[1*x^4+1*x+1,547]
[4] setmod_ff(2,5);
[2,x^5+x^2+1,x]
```

217. `field_type_ff([p|defpoly2])`
:: 設定されている基礎体の種類
- 設定されている基礎体の種類を返す.
- 設定なしなら 0, $GF(p)$ なら 1, $GF(2^n)$ なら 2 を返す.

```
[0] field_type_ff();
0
[1] setmod_ff(3);
3
[2] field_type_ff();
1
[3] setmod_ff(x^2+x+1);
x^2+x+1
[4] field_type_ff();
2
```

218. field_order_ff()
:: 設定されている基礎体の位数
- 設定されている基礎体の位数 (元の個数) を返す.
- 設定されている体が GF(q) ならば q を返す.

```
[0] field_order_ff();
field_order_ff : current_ff is not set
return to toplevel
[1] setmod_ff(3);
3
[2] field_order_ff();
3
[3] setmod_ff(x^2+x+1);
x^2+x+1
[4] field_order_ff();
4
```

219. characteristic_ff()
:: 設定されている体の標数
$GF(p)$ の場合 p, $GF(2^n)$ の場合 2 を返す.

```
[0] characteristic_ff();
characteristic_ff : current_ff is not set
return to toplevel
[1] setmod_ff(3);
3
[2] characteristic_ff();
3
[3] setmod_ff(x^2+x+1);
x^2+x+1
[4] characteristic_ff();
2
```

220. extdeg_ff()
:: 設定されている基礎体の, 素体に対する拡大次数
$GF(p)$ の場合 1, $GF(2^n)$ の場合 n を返す.

```
[0] extdeg_ff();
extdeg_ff : current_ff is not set
return to toplevel
[1] setmod_ff(3);
3
[2] extdeg_ff();
1
[3] setmod_ff(x^2+x+1);
x^2+x+1
[4] extdeg_ff();
```

```
2
```

221. simp_ff(*obj*)
:: 数, あるいは多項式の係数を有限体の元に変換
- 数, あるいは多項式の係数を有限体の元に変換する.
- 整数, あるいは整数係数多項式を, 有限体, あるいは有限体係数に変換するために用いる.
- 有限体の元に対し, 法あるいは定義多項式による reduction を行う場合にも用いる.
- 小標数有限体の元に変換する場合, 一旦素体上に射影してから, 拡大体の元に変換される. 拡大体の元に直接変換するには ptosfp() を用いる.

```
[0] simp_ff((x+1)^10);
x^10+10*x^9+45*x^8+120*x^7+210*x^6+252*x^5+210*x^4+120*x^3+45*x^2+10*x+1
[1] setmod_ff(3);
3
[2] simp_ff((x+1)^10);
1*x^10+1*x^9+1*x+1
[3] ntype(coef(@@,10));
6
[4] setmod_ff(2,3);
[2,x^3+x+1,x]
[5] simp_ff(1);
@_0
[6] simp_ff(2);
0
[7] ptosfp(2);
@_1
```

222. random_ff()
:: 有限体の元の乱数生成
- 有限体の元を乱数生成する.
- random(), lrandom() と同じ 32bit 乱数発生器を使用している.

```
[0] random_ff();
random_ff : current_ff is not set
return to toplevel
[1] setmod_ff(pari(nextprime,2^40));
1099511627791
[2] random_ff();
561856154357
[3] random_ff();
45141628299
```

223. lmptop(*obj*)
:: $GF(p)$ 係数多項式の係数を整数に変換
obj $GF(p)$ 係数多項式
- $GF(p)$ 係数多項式の係数を整数に変換する.
- $GF(p)$ の元は, 0 以上 p 未満の整数で表現されている. 多項式の各係数は, その値を整数オブジェクト (数識別子 0) としたものに変換される.

```
[0] setmod_ff(pari(nextprime,2^40));
1099511627791
[1] F=simp_ff((x-1)^10);
1*x^10+1099511627781*x^9+45*x^8+1099511627671*x^7+210*x^6
+1099511627539*x^5+210*x^4+1099511627671*x^3+45*x^2+1099511627781*x+1
[2] setmod_ff(547);
547
[3] F=simp_ff((x-1)^10);
1*x^10+537*x^9+45*x^8+427*x^7+210*x^6+295*x^5+210*x^4+427*x^3
+45*x^2+537*x+1
[4] lmptop(F);
x^10+537*x^9+45*x^8+427*x^7+210*x^6+295*x^5+210*x^4+427*x^3
+45*x^2+537*x+1
[5] lmptop(coef(F,1));
537
[6] ntype(@@);
0
```

224. ntogf2n(m)
:: 自然数を $GF(2^n)$ の元に変換
m　　非負整数
- 自然数 m の 2 進表現 $m = m_0 + m_1 2 + \cdots + m_k 2^k$ に対し, $GF(2^n) = GF(2)[t]/(g(t))$ の元 $m_0 + m_1 t + \cdots + m_k t^k \mod g(t)$ を返す.
- 定義多項式による剰余は自動的には計算されないため, simp_ff() を適用する必要がある.

```
[0] setmod_ff(x^30+x+1);
x^30+x+1
[1] N=ntogf2n(2^100);
(@^100)
[2] simp_ff(N);
(@^13+@^12+@^11+@^10)
```

225. gf2nton(m)
:: $GF(2^n)$ の元を自然数に変換
m　　$GF(2^n)$ の元
- gf2nton の逆変換である.

```
[1] setmod_ff(x^30+x+1);
x^30+x+1
[2] N=gf2nton(2^100);
(@^100)
[3] simp_ff(N);
(@^13+@^12+@^11+@^10)
[4] gf2nton(N);
1267650600228229401496703205376
[5] gf2nton(simp_ff(N));
15360
```

226. ptogf2n(*poly*)
:: 一変数多項式を $GF(2^n)$ の元に変換
poly　　一変数多項式
- *poly* の表す $GF(2^n)$ の元を生成する. 係数は, 2 で割った余りに変換される. *poly* の変数に @ を代入した結果と等しい.

```
[0] setmod_ff(x^30+x+1);
x^30+x+1
[1] ptogf2n(x^100);
(@^100)
```

227. gf2ntop(m,[v])
:: $GF(2^n)$ の元を多項式に変換
m　$GF(2^n)$ の元
v　不定元
- m を表す多項式を, 整数係数の多項式オブジェクトとして返す.
- v の指定がない場合, 直前の ptogf2n() 呼び出しにおける引数の変数 (デフォルトは x), 指定がある場合には指定された不定元を変数とする多項式を返す.

```
[0] setmod_ff(x^30+x+1);
x^30+x+1
[1] N=simp_ff(gf2ntop(2^100));
(@^13+@^12+@^11+@^10)
[2] gf2ntop(N);
x^13+x^12+x^11+x^10
[3] gf2ntop(N,t);
t^13+t^12+t^11+t^10
```

228. ptosfp(p)
:: 小標数有限体への変換
229. sfptop(p)
:: 小標数有限体からの変換
p　　多項式
- ptosfp() は, 多項式の係数を, 現在設定されている小標数有限体 $GF(p^n)$ の元に直接変換する. 係数が既に有限体の元の場合は変化しない. 正整数の場合, まず位数で剰余を計算したあと, 標数 p により p 進展開し, p を x に置き換えた多項式を, 原始元表現に変換する. 例えば, $GF(3^5)$ は $GF(3)[x]/(x^5+2x+1)$ として表現され, その各元は原始元 x に関するべき指数 k により @_k として表示される. このとき, 例えば $23=2\cdot 3^2+3+2$ は, $2x^2+x+2$ と表現され, これは結局 x^{17} と法 x^5+2x+1 で等しいので, @_17 と変換される.
- sfptop() は ptosfp() の逆変換である.

```
[0] setmod_ff(3,5);
[3,x^5+2*x+1,x]
[1] A = ptosfp(23);
@_17
[2] 9*2+3+2;
23
[3] x^17-(2*x^2+x+2);
x^17-2*x^2-x-2
```

```
[4] sremm(@,x^5+2*x+1,3);
0
[5] sfptop(A);
23
```

230. defpoly_mod2(d)
:: $GF(2)$ 上既約な d 次の一変数多項式の生成
d　　正整数
- fff で定義されている.
- 与えられた次数 d に対し, $GF(2)$ 上 d 次の既約多項式を返す.
- もし 既約 3 項式が存在すれば, 第 2 項の次数がもっとも小さい 3 項式, もし既約 3 項式が存在しなければ, 既約 5 項式の中で, 第 2 項の次数がもっとも小さく, その中で第 3 項の次数がもっとも小さく, その中で第 4 項の次数がもっとも小さいものを返す.

231. sffctr(*poly*)
:: 多項式の小標数有限体上での既約分解
poly　　有限体上の多項式
- 多項式を, 現在設定されている小標数有限体上で既約分解する.
- 結果は, [[f_1,m_1],[f_2,m_2],...] なるリストである. ここで, f_i は monic な既約因子, m_i はその重複度である.

```
[0] setmod_ff(2,10);
[2,x^10+x^3+1,x]
[1] sffctr((z*y^3+z*y)*x^3+(y^5+y^3+z*y^2+z)*x^2+z^11*y*x+z^10*y^3+z^11);
[[@_0,1],[@_0*z*y*x+@_0*y^3+@_0*z,1],[(@_0*y+@_0)*x+@_0*z^5,2]]
```

232. fctr_ff(*poly*)
:: 1 変数多項式の有限体上での既約分解
poly　　有限体上の 1 変数多項式
- fff で定義されている.
- 一変数多項式を, 現在設定されている有限体上で既約分解する.
- 結果は, [[f_1,m_1],[f_2,m_2],...] なるリストである. ここで, f_i は monic な既約因子, m_i はその重複度である.
- *poly* の主係数は捨てられる.

```
[0] setmod_ff(2^64-95);
18446744073709551521
[1] fctr_ff(x^5+x+1);
[[1*x+14123390394564558010,1],[1*x+6782485570826905238,1],
[1*x+15987612182027639793,1],[1*x^2+1*x+1,1]]
```

233. irredcheck_ff(*poly*)
:: 1 変数多項式の有限体上での既約判定
- 有限体上の 1 変数多項式
- fff で定義されている.
- 有限体上の 1 変数多項式の既約判定を行い, 既約の場合 1, それ以外は 0 を返す.

```
[1] setmod_ff(2^64-95);
18446744073709551521
[2] F=x^10+random_ff();
x^10+14687973587364016969
[3] irredcheck_ff(F);
```

1

234. randpoly_ff(d,v)
:: 有限体上の乱数係数 1 変数多項式の生成
d 正整数
v 不定元
- fff で定義されている.
- d 次未満, 変数が v, 係数が現在設定されている有限体に属する 1 変数多項式を生成する. 係数は gotor:randomffrandom_ff() により生成される.

```
[0] setmod_ff(2^64-95);
18446744073709551521
[1] F=x^10+random_ff();
[2] randpoly_ff(3,x);
17135261454578964298*x^2+476682669965361 5429*x+18317369440429479651
[3] randpoly_ff(3,x);
7565988813172050604*x^2+743007576727966 5339*x+4699662986224873544
[4] randpoly_ff(3,x);
10247781277095450395*x^2+10243690944992524936*x+40638290492688454 92
```

235. ecm_add_ff(p_1,p_2,ec)
:: 楕円曲線上の点の加算
236. ecm_sub_ff(p_1,p_2,ec)
:: 楕円曲線上の点の減算
237. ecm_chsgn_ff(p_1)
:: 楕円曲線上の点の逆元
p_1 p_2 長さ 3 のベクトルまたは 0
ec 長さ 2 のベクトル
- 現在設定されている有限体上で, ec で定義される楕円曲線上の点 p_1, p_2 の和 p_1+p_2, 差 p_1-p_2, 逆元 $-p_1$ を返す.
- ec は, 設定されている有限体が奇標数素体の場合, $y^2 = x^3 + ec[0]x + ec[1]$, 標数 2 の場合 $y^2+xy = x^3 + ec[0]x^2 + ec[1]$ を表す.
- 引数, 結果ともに, 無限遠点は 0 で表される.
- p_1, p_2 が長さ 3 のベクトルの場合, 斉次座標による曲線上の点を表す. この場合, 第 3 座標は 0 であってはいけない.
- 結果が長さ 3 のベクトルの場合, 第 3 座標は 0 でないが, 1 とは限らない. アフィン座標による結果を得るためには, 第 1 座標, 第 2 座標を第 3 座標で割る必要がある.
- p_1, p_2 が楕円曲線上の点かどうかのチェックはしない.

```
[0] setmod_ff(1125899906842679)$
[1] EC=newvect(2,[ptolmp(1),ptolmp(1)])$
[2] Pt1=newvect(3,[1,-412127497938252,1])$
[3] Pt2=newvect(3,[6,-252647084363045,1])$
[4] Pt3=ecm_add_ff(Pt1,Pt2,EC);
[ 560137044461222 184453736165476 125 ]
[5] F=y^2-(x^3+EC[0]*x+EC[1])$
[6] subst(F,x,Pt3[0]/Pt3[2],y,Pt3[1]/Pt3[2]);
0
[7] ecm_add_ff(Pt3,ecm_chsgn_ff(Pt3),EC);
```

```
0
[8] D=ecm_sub_ff(Pt3,Pt2,EC);
[ 886545905133065 119584559149586 886545905133065 ]
[9] D[0]/D[2]==Pt1[0]/Pt1[2];
1
[10] D[1]/D[2]==Pt1[1]/Pt1[2];
1
```

3.3.12 その他の函数

238. `ctrl("`*switch*`" [,`*obj*`])`

:: "*switch*" で指定した環境の設定，設定値を返す.

- Asir の実行環境の設定変更, 参照を行う.
- *switch* のみの場合, そのスイッチの現在の状態を返す.
- *obj* が与えられているとき, その値を設定する.
- スイッチは文字列として入力する. すなわちダブルクォートで囲む.
- スイッチは次の通り. 以下で, on は 1, off は 0 を意味する.

cputime　on の時 CPU time および GC time を表示, off の時 表示しない. `ctrl("cputime",onoff)` は `cputime(onoff)` と同じである.

nez　EZGCD のアルゴリズムの切替え. デフォルトで 1 であり, とくに切替える必要はない.

echo　on の時は標準入力を繰り返して出力し, off の時は標準入力を繰り返さない. output コマンドを用いる際に有効である.

bigfloat　on の時, 入力された浮動小数は bigfloat に変換され, 浮動小数演算は PARI により行われる. デフォルトの有効桁数は 9 桁である. 有効桁数を増やしたい時には `setprec()` を用いる. off の時, 入力された浮動小数は, 倍精度浮動小数に変換される.

adj　ガーベッジコレクションの頻度の変更. 1 以上の有理数が指定できる. デフォルト値は 3. 1 に近い程, ガーベッジコレクションせずにヒープを大きくとるようになる. 整数値はコマンドラインで指定できる. See section コマンドラインオプション.

verbose　on の時, 函数の再定義時にメッセージを表示する.

quiet_mode　1 のとき, 起動時に著作権表示を行わない. See section コマンドラインオプション.

prompt　0 のときプロンプトを表示しない. 1 のとき標準プロンプトを表示. C スタイルのフォマット文字列をもちいるとユーザ定義のプロンプト.
例 (`asirgui` では不可): `ctrl("prompt","\033[32m[%d]:= \033[0m")`

hex　1 のとき, 整数は 0x で始まる 16 進数として表示される. −1 のとき, 16 進数は, 間に '—' をはさんで 8 桁ごとに区切って表示される.

real_digit　倍精度浮動小数の表示の桁数を指定する.

double_output　1 のとき, 倍精度浮動小数はつねに ddd.ddd の形で表示される.

fortran_output　1 のとき, 多項式の表示が `FORTRAN` スタイルになる. すなわち冪が '^' の代わりに '**' で表される. (デフォルト値は 0.)

ox_batch　1 のとき, 送信バッファがいっぱいになった時のみ自動的に flush. 0 のとき, データ, コマンド送信毎に flush. (デフォルト値は 0.) See section 分散計算.

ox_check　1 のとき, 送信データを相手プロセスが受け取れるかどうかチェックする. 0 のときしない. (デフォルト値は 1.) See section 分散計算.

ox_exchange_mathcap　1 のとき, OX server との接続開始時に, 自動的に mathcap の交換を行う. (デフォルト値は 1.) See section 分散計算.

loadpath　`asir` のロードパスの出力または設定を行う.

```
[0] L=ctrl("loadpath");
[/home/you/OpenXM/lib/asir-contrib,/home/you/OpenXM/lib/asir,.]
[1] ctrl("loadpath", cons(getenv("HOME")+"/lib",L));
```

0

239. debug
:: デバッグモードに入る
- debug は無引数の函数であるが, '()' なしで呼び出せる.
- デバッグモードに入るとプロンプトが (debug) となり, コマンド受け付け状態となる. quit を入力するとデバッガから抜ける.
- デバッグモードについての詳細はデバッガを参照.

240. error(*message*)
:: エラーメッセージを表示してデバッグモードに入る
- 一般に, 引数の間違いなど, 続行不可能なエラーが組み込み函数において発生した時, トップレベルに戻る前に, 可能ならばそのエラーの時点でデバッグモードに入る. error() は, ユーザ函数の内部でこの動作と同様の動作を行わせるための函数である.
- 引数は, error() が呼び出される際に表示されるメッセージで, 文字列である.
- ユーザ函数において, 変数をチェックして, あり得ない値の場合に error() を呼び出すようにしておけば, その時点で自動的にデバッグモードに入れる.

```
% cat mod3
def mod3(A) {
    if ( type(A) >= 2 )
        error("invalid argument");
    else
        return A % 3;
}
end$
% asir
[0] load("mod3");
1
[1] mod3(5);
2
[2] mod3(x);
invalid argument
stopped in mod3 at line 3 in file "./mod3"
3                       error("invalid argument");
(debug) print A
A = x
(debug) quit
return to toplevel
[3]
```

241. time()
:: セッション開始から現在までの CPU 時間および GC 時間を表示する
- CPU 時間および GC 時間の表示に関するコマンドである.
- GC 時間とは, ガーベジコレクタにより消費されたと見なされる時間, CPU 時間は, 全体の CPU 時間から GC 時間を引いた残りで, 単位は秒である.
- time() は引数なしで, セッション開始から現在までの CPU 時間, GC 時間, 現在までに要求されたメモリののべ容量, およびセッション開始から現在までの経過時間の表示をする. すなわち, [CPU 時間 (秒), GC 時間 (秒), メモリ量 (ワード), 経過時間 (秒)] なるリストを返す. 1 ワードは通常 4 バイトである.

- 計算の実行開始時, 終了時の time() から, その計算に対する CPU 時間, GC 時間がわかる.
- メモリ量は多倍長数ではないため, ある値を越えると無意味な値となるためあくまでも目安として用いるべきである. ctrl() や cputime() により cputime スイッチが on になっている場合には, トップレベルの文を一つの単位として, その実行時間が表示される. しかし, プログラムの内部などで, 特定の計算に対する計算時間を知りたい時には, time() などを使う必要がある. getrusage() が使える UNIX 上では time() は信頼性のある値を返すが, Windows 95, 98 上では時刻を用いるほか方法がないため経過時間そのものが表示される. よって, 待ち状態があると, それも経過時間に加算される.

```
[0] T0=time();
[2.390885,0.484358,46560,9.157768]
[1] G=hgr(katsura(4),[u4,u3,u2,u1,u0],2)$
[2] T1=time();
[8.968048,7.705907,1514833,63.359717]
[3] ["CPU",T1[0]-T0[0],"GC",T1[1]-T0[1]];
[CPU,6.577163,GC,7.221549]
```

242. cputime(*onoff*)
:: 引数が 0 ならば cputime の表示を止める, それ以外ならば表示を行う
243. tstart()
:: CPU time 計測開始
244. tstop()
:: CPU time 計測終了および表示
- cputime() は, 引数が 0 ならば CPU time の表示を止める. それ以外ならば表示を行う.
- tsart は引数なし, '()' なしで, CPU time 計測を開始する.
- tstop は引数なし, '()' なしで, CPU time 計測を終了, および表示する.
- cputime(onoff) は ctrl("cputime",onoff) と同じである.
- tstart, tstop は, 入れ子にして使われることは想定していないため, そのような可能性がある場合には, time() による計測を行う必要がある.
- cputime() による on, off は, 単に表示の on, off であり, トップレベルの一つの文に対する計測は常に行われている. よって, 計算を始めてからでも, 計算終了前にデバッガに入って cputime(1) を実行させれば計算時間は表示される.

```
[0] tstart$
[1] fctr(x^10-y^10);
[[1,1],[x+y,1],[x^4-y*x^3+y^2*x^2-y^3*x+y^4,1],[x-y,1],
[x^4+y*x^3+y^2*x^2+y^3*x+y^4,1]]
[2] tstop$
80msec + gc : 40msec
```

245. timer(*interval*,*expr*,*val*)
:: 制限時間つきで計算を実行する
- 時間を指定して計算を実行する. 指定時間内に計算が完了した場合その値を返す. 指定時間内に計算が完了しなかった場合, 第 3 引数を返す.
- 第 3 引数の値は, 計算が完了した場合の値と区別できる必要がある.

```
[0] load("cyclic");
1
[1] timer(10,dp_gr_main(cyclic(7),[c0,c1,c2,c3,c4,c5,c6],1,1,0),0);
interval timer expired (VTALRM)
```

```
0
[2]
```

246. currenttime()
1970 年 1 月 1 日 0 時 0 分 0 秒からの経過秒数
- currenttime() は現在時刻を返す. UNIX の場合, time(3) を呼んでいるだけである.

```
[0] currenttime();
1071639228
[1]
```

247. sleep(*interval*)
:: プロセスの実行を $interval \times 10^{-3}$ 秒停止
- sleep() は, プロセスの実行を停止する. UNIX の場合, usleep を呼んでいるだけである

```
[0] sleep(1000);
1
[1]
```

248. heap()
:: 現在のヒープの大きさを返す (単位:バイト)
- 現在のヒープの大きさ (単位 : バイト) を返す. ヒープとは, Asir のさまざまな数式や, ユーザプログラムなどがおかれるメモリの領域で, ガーベジコレクタにより管理されている. プログラムの動作中は, ヒープの大きさは単調非減少であり, 実メモリの量をこえて大きくなった場合には, OS によるスワップエリアへの読み書きがほとんどの計算時間を占めることになる.
- 実メモリが少ない場合には, 起動時の -adj オプションにより, GC 主体の設定を行っておく必要がある.

```
% asir -adj 16
[0] load("fctrdata")$
0
[97] cputime(1)$
0msec
[98] heap();
524288
0msec
[99] fctr(Wang[8])$
3.190sec + gc : 3.420sec
[100] heap();
1118208
0msec
[101] quit;
% asir
[0] load("fctrdata")$
0
[97] cputime(1)$
0msec
[98] heap();
827392
```

```
0msec
[99] fctr(Wang[8])$
3.000sec + gc : 1.180sec
[100] heap();
1626112
0msec
[101] quit;
```

249. `version()`
:: `Asir` のバージョン（整数）を返す.

```
[0] version();
991214
```

250. `shell(`*command*`)`
:: *command* をシェルコマンドとして実行する
- *command* を *C* の `system()` 函数によりシェルコマンドとして実行する. シェルの終了ステータスを返す.

```
[0] shell("ls");
alg             da              katsura         ralg            suit
algt            defs.h          kimura          ratint          test
alpi            edet            kimura3         robot           texput.log
asir.o          fee             mfee            sasa            wang
asir_symtab     gr              mksym           shira           wang_data
base            gr.h            mp              snf1            wt
bgk             help            msubst          solve
chou            hom             p               sp
const           ifplot          proot           strum
cyclic          is              r               sugar
0
```

251. `map(`*function*`,`*arg0*`,`*arg1*`,...)`
:: リスト, 配列の各要素に函数を適用する
- *arg0* の各要素を最初の引数, *arg1* 以下の残りの引数として函数 *function* を呼び出し, *arg0* の対応する要素の位置に函数呼び出しの結果が入った同じ型のオブジェクトを生成して返す.
- *function* は, ダブルクォートのない函数名を用いる.
- *function* にプログラム変数は使えない.
- *arg0* がリスト, ベクトル, 行列以外の場合, 単に *arg0*, *arg1*, ... を引数として *function* を呼び出しその結果を返す.
- `map` の引数 *function* で与えられる函数は, 内部的にも函数として実装されていなければならない. そうでなければ parse error になる. 例えば `map` 自身や `car`, `cdr` などは内部的には函数ではなく, `Asir` の文法におけるキーワードとして実装されている. したがって `map` の引数に `map` をとることはできない
- オプションやネスティングが可能な函数 `mtransbys()` がある.

```
[0] def afo(X) { return X^3; }
[1] map(afo,[1,2,3]);
[1,8,27]
```

252. flist()
:: 現在定義されている組み込み函数, ユーザ定義函数の函数名を文字列リストとして返す
- システム函数の後にユーザ定義函数が続く.

```
[0] flist();
[defpoly,newalg,mainalg,algtorat,rattoalg,getalg,alg,algv,...]
```

253. delete_history([*index*])
:: ヒストリを消去する
- 引数がないとき, これまで計算したヒストリを全て消去する.
- 引数があるとき, その番号の結果のみ消去する.
- ここでヒストリとは, 番号つきのプロンプトに対しての入力を評価して得られた式で, この式は @*number* により取り出すことができる. このことは, ガーベッジコレクションの際にもこの式が生き残ることを意味する.
- 大きな式がヒストリとして残った場合, 以降のメモリ管理に支障を来す場合が多いため, bsave() などでファイルにセーブして, delete_history() によりヒストリを消去しておくのが有効である.

```
[0] (x+y+z)^100$
[1] @0;
...
[2] delete_history(0);
[3] @0;
0
```

254. get_rootdir()
:: Asir のルートディレクトリ名を取り出す
- UNIX 版の場合, 環境変数 ASIR_LIBDIR が定義されている場合にはその値, されていない場合には ‘/usr/local/lib/asir’ を返す.
- Windows 版の場合, ‘asirgui.exe’ のあるディレクトリ (‘bin’ という名前のはずである) の親ディレクトリが返される.
- この関数が返すディレクトリ名を基準とした相対パス名を指定することにより, インストールされた場所によらないファイル読み込みプログラムを書くことができる.

255. getopt([*key*])
:: オプションの値を返す
- ユーザ定義関数は, 固定個数引数でしか宣言できない. ユーザ定義関数で可変個引数を実現する方法の一つとして, オプションによる引数の指定がある (see オプション指定). 指定されたオプションを関数内で受け取るためにこの関数を用いる.
- 無引数で呼び出された場合, getopt() は [[*key1*,*value1*],[*key2*,*value2*],...] なるリストを返す. ここで, *key* は関数呼び出し時に指定されたオプション, *value* はその値である.
- 関数呼び出しの際に key がオプションとして指定されている場合には, その値を返す. もし指定がない場合には, VOID 型オブジェクト (型識別子 −1) を返す. getopt() が返した値の型を type() で調べることで, そのオプションが指定されたかどうか調べることができる.
- 関数呼び出しにおけるオプションの指定は, 正規の引数ならびの後ろに
  xxx(A,B,C,D|x=X,y=Y,z=Z)
  という風に, ‘|’ に続く, *key*=value の ‘,’ で区切られた並びを置くことで行う.

256. getenv(*name*)
:: 環境変数 *name* の値を返す

```
[0] getenv("HOME");
/home/pcrf/noro
```

参考文献

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

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

[O1] T. Oshima, 特殊関数と代数的線型常微分方程式, 東京大学数理科学レクチャーノート **11**, 2011, http://www.ms.u-tokyo.ac.jp/publication/documents/spfct3.pdf.

[O2] T. Oshima, *Fractional calculus of Weyl algebra and Fuchsian differential equations*, MSJ Memoirs **28**, Mathematical Society of Japan, Tokyo, 2012.

[O3] T. Oshima, Transformation of KZ type equations, to appear in RIMS Kôkyûroku Bessatsu, 2016.

[O4] 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.

[O5] T. Oshima, `okubo`, a computer program for Katz/Yokoyama/Oshima algorithms for spectral types, 2007-8, ftp://akagi.ms.u-tokyo.ac.jp/pub/math/okubo/

[O6] T. Oshima, `os_muldif.rr`, a library of the calculation of differential operators for computer algebra Risa/Asir, 2007–2016, ftp://akagi.ms.u-tokyo.ac.jp/pub/math/muldif/

[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.

[Shoup] V. Shoup, *A new polynomial factorization algorithm and its implementation*, J. Symb. Comp. **20**(1995), 364–397.

[Weber] K. Weber, *The accelerated Integer GCD Algorithm*, ACM TOMS, **21, 1**(1995), 111–122.

索引