Command Reference

SHELL, MATHEMATIC LIBRARY

SL-UTIL, SYSTEM, ISPP, GPM

BPS, NCS, NPE, DCM

Version 2.0, 24th Apr 2000

BPEL-TUT

目 次

SHELL (シェル) **1**
ALIAS 2
BREAK 3
CONST 4
CONTINUE 5
DEFINE 6
DO 7
EVAL 8
EXIT 9
EXTERNAL 10
FOR 11
FUNC 12
GARCO 13
GOODBYE 14
HISTORY 15
IF 16
INDEX 17
INLINE 18
ISDEF 19
LENGTH 20
MODULE 21
PRINT 22
PRINTF 23
PROC 24
READ 25
RETURN 26
SCALAR 27
SERIES 28
SNAPSHOT 29
STRING 30
STRLEN 31
SYMBOLS 32
TYPEOF 33
UNDEF 34
UNIX 35

WELCOME · · · · · 36
WHILE · · · · · 37
WRITE_TYPE · · · · · 38

Mathematic Library (数学ライブラリ) 39
ABS · · · · · 40
ACOS · · · · · 41
ASIN · · · · · 42
ATAN · · · · · 43
ATAN2 · · · · · 44
COS · · · · · 45
EXP · · · · · 46
EXP2 · · · · · 47
INT · · · · · 48
LOG · · · · · 49
LOG2 · · · · · 50
LOG10 · · · · · 51
MOD · · · · · 52
POW · · · · · 53
SGN · · · · · 54
SIN · · · · · 55
SQRT · · · · · 56
TAN · · · · · 57

SL_UTIL (ユーティリティ) 58
EXIST · · · · · 59
DUMP · · · · · 60
EXTRACT · · · · · 61
FILEINDEX · · · · · 62
HGET · · · · · 63
INDEXSIZE · · · · · 64
PUTS · · · · · 65
REVTIME · · · · · 66
Scalar · · · · · 67
Series · · · · · 68
Snapshot · · · · · 69
SPRINTF · · · · · 70
Sprintf · · · · · 71
String · · · · · 72
TEXT2STRING · · · · · 73
UNITMAT · · · · · 74
TOLOWER · · · · · 75
TOUPPER · · · · · 76

SYSTEM (基本コマンド) **77**
BM · · · · 78
CUT · · · · 79
FILL · · · · 80
FIND · · · · 81
GET · · · · 82
HEADER · · · · 83
HELP · · · · 84
MABI · · · · 85
MAX · · · · 87
MAXPOS · · · · 88
MERGE · · · · 89
MIN · · · · 90
MINPOS · · · · 91
PUT · · · · 92
REFORM · · · · 93
REVERSE · · · · 94
ROTATE · · · · 95
SAM · · · · 96
WAIT · · · · 97
ZERO · · · · 98

ISPP (ディジタル信号処理) **99**
AKIMA · · · · 100
ARAND · · · · 101
ARGEN · · · · 102
AVERAGE · · · · 104
BPBTW · · · · 105
BURG · · · · 106
CEP · · · · 108
DCCUT · · · · 109
DET · · · · 110
EIGEN · · · · 111
FFTC · · · · 112
FFTN · · · · 113
FIR · · · · 114
FIRMAKE · · · · 115
GAUSS2 · · · · 116
HIL · · · · 117
HPBTW · · · · 118
ICEP · · · · 119
IIR · · · · 120
IIRCOEF · · · · 121

INTEG · · · 122
INTERP · · · 123
INV · · · 124
LEVIN · · · 125
LPBTW · · · 126
MNRAND · · · 127
MUL · · · 128
NMEQ · · · 129
NORM · · · 130
NRAND · · · 131
PHASE · · · 132
POLE · · · 133
POWER · · · 134
RANK · · · 135
SHIFT · · · 136
SPCF · · · 137
SPLINE · · · 138
TRANS · · · 139
URAND · · · 140
WINDOW · · · 141

GPM (グラフィックシステム) 142
AXIS · · · 143
CHWIN · · · 145
COLOR · · · 146
CONT · · · 147
DRAW · · · 148
EGRAPH · · · 149
FACTOR · · · 151
FONT · · · 152
FRAME · · · 153
GINIT · · · 154
GPM2PS · · · 155
GRAPH · · · 156
GSOLM · · · 158
GSTAT · · · 160
LABEL · · · 161
LINE · · · 162
LTYPE · · · 163
LWIDTH · · · 164
MAP · · · 165
NEWPAGE · · · 167
ORIGIN · · · 168

PREV · · · · · 169
SCALE · · · · · 170
SIZE · · · · · 172
TITLE · · · · · 173
WCLOSE · · · · · 174
WE · · · · · 175
WOPEN · · · · · 176

BPS (ニューラルネットシミュレータ) 177
ACTLOAD · · · · · 178
BPLOAD · · · · · 179
BPSAVE · · · · · 180
COR · · · · · 181
DISP · · · · · 182
ERRFUNC · · · · · 183
ERRLOAD · · · · · 184
ERROR · · · · · 185
FUNCTION · · · · · 186
LALGO · · · · · 187
LAYER · · · · · 189
LEARN · · · · · 190
LEND · · · · · 191
PDISP · · · · · 192
PINIT · · · · · 193
REC · · · · · 195
RVMAP · · · · · 196
SETREC · · · · · 197
SIGMOID · · · · · 198
TEACH · · · · · 199
WALGO · · · · · 200
WEIGHT · · · · · 201
WGTLOAD · · · · · 202
WGTSET · · · · · 204
WINIT · · · · · 205

NCS (神経回路シミュレータ) 206
NASSIGN · · · · · 207
NPP · · · · · 208
NLINK · · · · · 209
NMAKE · · · · · 210
NTIME · · · · · 211
NSTIM · · · · · 212
NOUT · · · · · 213

NPARA · · · · · 214
NINTEG · · · · · 215
NDELAY · · · · · 216
NLIST · · · · · 217
NSCLIST · · · · · 218
NERASE · · · · · 219
NGETP · · · · · 220
NCAL · · · · · 221
NE · · · · · 222
NCHGBUFF · · · · · 223

NPE (非線形パラメータ推定システム) 224
CDATA · · · · · 225
CDEL · · · · · 226
CDISP · · · · · 227
CHISTORY · · · · · 228
CINIT · · · · · 229
CINTEG · · · · · 230
CLIST · · · · · 231
CLOAD · · · · · 232
CLOGIC · · · · · 233
CLSEARCH · · · · · 234
CMETHOD · · · · · 235
CMODEL · · · · · 236
CNORM · · · · · 237
CNUMBER · · · · · 238
CPENALTY · · · · · 239
CPOINT · · · · · 240
CRESULT · · · · · 241
CSCALE · · · · · 242
CSTORE · · · · · 243
CTERM · · · · · 244
CWEIGHT · · · · · 245
NPE · · · · · 246
NPEINIT · · · · · 247

DCM (データ変換モジュール) 248
ABF2SATELLITE · · · · · 249
ATF2SATELLITE · · · · · 250
BUF2AVS · · · · · 251
BUFFER2AVS · · · · · 252
BUF2TXT · · · · · 253
BUFFER2TEXT · · · · · 254

BUF2MAT 255
BUFFER2MATLAB 256
BUF2MATH 257
BUFFER2MATHEMATICA 258
GEN2BUF 259
GENESIS2BUFFER 260
MAT2SAT 261
MATLAB2SATELLITE 262
NRN2SAT 263
NEURON2SATELLITE 264
TXT2BUF 265
TEXT2BUFFER 266

コマンド索引 **267**

シェル

SHELL

ALIAS

機能) **コマンドなどの別名を表示・定義・変更・削除する**.

形式) **alias([“another_name”, “command_string”])**

パラメータ)
1. another_name : **別名文字列**
2. command_string : **実行コマンド**

解説)
1. **別名が実行コマンドと同じ名前である時には**, **最初の１回だけエイリアス展開する**.
2. **何もパラメータを設定しない場合には**, **別名定義の一覧を表示する**.
3. **別名を削除したい場合には**, command_string **にヌル文字** (“”) **を設定すればよい**.

使用例)
1. **別名定義の一覧表示**

```
% alias()
        ls      \ls -lCF
        h       history()
        close   exit
```

2. **別名の定義**

```
% alias( "quit", "exit" )
```

3. **別名の削除**

```
% alias( "quit", "" )
```

ソースファイル **組み込み関数**

BREAK

機能) **ループを強制的に抜ける**.

形式) break

パラメータ) **なし**

解説) **使用例を示す**.

```
% for( i=0; i<100; i++ ) {
%     .....処理1.......
%     if ( data>1e15 ) {
%         break; /* NEXTに処理が移る */
%     }
%     .....処理2......
% }
% .......NEXT.......
```

関連項目) CONTINUE

ソースファイル **組み込み関数**

CONST

機能) **定数を定義する**

形式) **const name = value**

パラメータ) 1. name : **定数名**

2. value : **定数** (**数値**, **文字列**)

解説) **定数として定義した値の変更は**, undef **で定義を取り消すか** const **で再定義し直すこと以外は許可されない**.

ソースファイル

組み込み関数

CONTINUE

機能) **ループの先頭に処理を移す**.

形式) continue

パラメータ) **なし**

解説) **使用例を示す**.

```
% for( i=0; i<100; i++ ) {
%     ......処理1......
%     if ( data != 0.0 ) {
%         continue; /* ループ先頭(処理1)に処理が移る */
%     }
%     ......処理2......
% }
```

関連項目) BREAK

ソースファイル

組み込み関数

DEFINE

機能) **コマンドモジュールを定義する**.

形式) **define mod_name { stmt }**

パラメータ) 1. mod_name : **モジュール名**

2. stmt : **パラメータ列**

解説) **モジュールの定義例を示す**.

```
define SYSTEM {
  set Command_Path "/usr/local/bin/sl-bin/SYSTEM/"
  set Command_File "/usr/local/bin/sl-bin/SYSTEM/COMMAND.SYSTEM"
  set Message_File "/usr/local/bin/sl-bin/SYSTEM/MESSAGE.SYSTEM"
  set Error_File   "/usr/local/bin/sl-bin/SYSTEM/ERROR.SYSTEM"
  set Setup_File   "/usr/local/bin/sl-bin/SYSTEM/SETUP.SYSTEM"
  set Clean_File   "/usr/local/bin/sl-bin/SYSTEM/CLEAN.SYSTEM"
}
```

関連項目) MODULE

DO

機能) DO–WHILE **型のループ制御をする**.

形式) **do { stmt } while (expr);**

パラメータ)
1. stmt : **処理** ({} **で囲まれた複数のコマンドによる処理の記述ができる**)
2. expr : **条件判定**

解説)
1. expr **が真** (0 **以外**) **である間**, stmt **を繰り返し実行する**.
2. **コマンド末尾にセミコロン** (;) **が必要なので注意すること**.
3. expr **において**, **論理演算子には** AND **演算子** "&&" **や** "||", **及び全ての関係演算子が使用できる**. **論理演算の結果が** 0 **ならば偽**, **それ以外は真となる**.

使用例) **処理を** 100 **回繰り返して実行する場合の記述例**

```
% i=0
% do {
%     ......処理......
%     i++
% } while ( i<100 );
```

ソースファイル

組み込み関数

EVAL

機能) **文字列を** *SATELLITE* **のコマンドとして評価する**.

形式) **eval(string)**

パラメータ) string : **文字列**

使用例) **例えば** inline **コマンドでは通常**, **引数に直接** String **オブジェクトをとることができないが**, eval **コマンドを使用することで**, **これを記述することができる**.

```
% string rdfile
% rdfile = "sample.sl"
%
% /* 次の記述ではエラーが発生 */
% inline( rdfile )
%
% /* 対処方法 */
% eval( "inline( \" "+rdfile+" \" )" )
```

ソースファイル

組み込み関数

EXIT

機能) *SATELLITE*シェルを終了する.

形式)

```
exit
```

パラメータ) なし

ソースファイル

組み込み関数

EXTERNAL

機能) **オブジェクトの外部参照を宣言する**.

形式) external x, y, z, ...

パラメータ) x, y, z, ... **外部参照とするオブジェクト名**

解説) **宣言した変数は**, **トップレベルで使用している変数が参照される**. **関数および手続き内での宣言のみ有効であり**, **関数および手続きの定義中の先頭で行なう必要がある**.

使用例) **トップレベルで宣言した** Series **オブジェクト** x **を関数内で参照する場合の記述例**

```
% n = 20
%
% series x
% x = 0~n/n
%
% func sinwave( a,f,th ) {
%    external x
%    series y
%
%    y = a*sin( 2*PI*f*x+th )
%
%    return y
% }
```

関連項目) FUNC, PROC

FOR

機能) for **ループ制御をする**

形式) **for (stmt1 ; expr ; stmt2) stmt3**

パラメータ)

1. stmt1 : **カウンタの初期化**
2. expr : **条件判定**
3. stmt2 : **カウンタの再設定**
4. stmt3 : **処理** ({} **で囲まれた複数のコマンドによる処理の記述ができる**)

解説)

1. stmt1, stmt2, expr **に複文を記述することはできない**.
2. for**ループでは**, **まず** stim1 **が実行され**, **次に** expr **が真である間**, stim3 **を実行し**, stim2 **が実行される**.
3. expr **において**, **論理演算子には** AND **演算子** "&&" **や** "||", **及び全ての関係演算子が使用できる**. **論理演算の結果が** 0 **ならば偽**, **それ以外は真となる**.

使用例) **処理を** 100 **回繰り返して実行する場合の記述例**

```
% for( i=0; i<100; i++ ) {
%     ......処理......
% }
```

ソースファイル

組み込み関数

FUNC

機能) **関数を定義する**.

形式) **func function([arg1,...,argn]) stmt**

パラメータ)

1. function : **関数名**
2. arg1,...,argn : **引数名**
3. stmt : **処理** ({} **で囲まれた複数のコマンドによる処理の記述ができる**)

解説)

1. **引数**, **また返り値として**, Scalar, Series, Snapshot, String **型のオブジェクトを扱うことができる**.
2. return **コマンドにより**, **呼び出し元にデータを返すことができる**.
3. **関数内で引数の内容の書き換えた場合**, **この変化は呼び出し元の対応する変数に影響する**.

使用例) **アスキーテキストファイルの数値データを** Series **オブジェクトに変換する関数の記述例**

```
% func asciitoseries( filename ) {
%     return 0 + unix( "cat " + filename )
% }
```

関連項目) EXTERNAL, RETURN

ソースファイル

組み込み関数

機能) **ガーベージ・コレクションを強制的に行なう**.

形式) GarCo()

パラメータ) **なし**

解説) **未定義シンボルなどが削除される**.

ソースファイル **組み込み関数**

GOODBYE

機能) **終了時のメッセージを表示する**.

形式) goodbye()

パラメータ) **なし**

解説) .clean.sl **に記述しておけば**, *SATELLITE***終了時に次のようなメッセージが表示される**.

```
Exit The SATELLITE World      ... Thank you.
```

ソースファイル

組み込み関数

HISTORY

機能) **ヒストリ (コマンド実行履歴) の大きさの設定, ヒストリの一覧を表示する.**

形式) history([hsiz])

パラメータ) hsiz : **ヒストリのサイズ**

解説)

1. **ヒストリサイズはデフォルトで** 256 **に設定されている.**
2. **ヒストリサイズを指定しない場合には, ヒストリの一覧が表示される.**
3. **ヒストリサイズに負値を与えると, ヒストリサイズが出力される.**

ソースファイル

組み込み関数

IF

機能) **条件分岐をする**.

形式) **if (expr) stmt1 [else { stmt2 }]**

パラメータ)

1. expr : **条件判定**
2. stmt1 : **処理**
3. stmt2 : **処理**

解説)

1. else **文を記述する場合**, stmt1 **が**, **単文であっても** { } **で囲む必要がある**. **また**, `} else {` **のように**, } **の直後に** else **を続けて記述する**. **改行を入れてはいけない**.
2. expr **において**, **論理演算子には** AND **演算子** "&&" **や** "||", **及び全ての関係演算子が使用できる**. **論理演算の結果が** 0 **ならば偽**, **それ以外は真となる**.

ソースファイル

組み込み関数

INDEX

機能) **オブジェクトのインデックスを得る**.

形式)

```
idx = index( x )
```

パラメータ) 1. x : **オブジェクト** (Series, Snapshot, String)

2. idx : **要素数出力** (Series, Scalar)

解説) **データのインデックスを出力する**. **一次元データであれば** Scalar **値が**, **多次元データであれば** Series **値で出力される**. **例えば**, **インデックスが** [10][50] **のデータであれば**, [0]:% 10 50 **が得られる**.

ソースファイル

組み込み関数

INLINE

機能) *SATELLITE* **言語のスクリプトを実行する**.

形式) **inline("script")**

パラメータ) script : **スクリプトファイル名**

解説)

1. inline **コマンドで使用されるオブジェクトは**, **スクリプトの実行終了後も有効であり**, func, proc **とは異なる**.
2. **戻り値はない**.
3. inline **を入れ子にした場合には，まずすべての** inline **で指定されたスクリプトがスタックコードに展開される．これを抑制し，実行途中でスクリプトを読み込ませるには，**eval **コマンドを併用し，**`eval("inline \"script\")` **のようにする．**

使用例) sample.sl **を実行する**

```
% inline( "sample.sl" )
```

ソースファイル

組み込み関数

ISDEF

機能) **オブジェクト名が使用されているかどうかを調べる．**

形式) isdef(name)

パラメータ) name : **オブジェクト名**

解説)

1. Scalar, Series, Snapshot, String **型のオブジェクト，もしくは定数として定義されている場合には** 1 **を返す．**
2. **オブジェクト名が定義されていない場合** 0 **を返す．**
3. **モジュール名，**func, proc **などの名前で使われている場合には，**-1 **を返す．**

使用例) sample.sl **を実行する**

```
% isdef( a );
```

LENGTH

機能) **オブジェクトの要素数を得る**.

形式) `num = length( x )`

パラメータ) 1. x : **オブジェクト** (Series, Snapshot, String)

2. num : **要素数出力** (Scalar)

解説) **多次元データの時系列方向の要素数を出力する**. **例えば**, **インデックスが** [10][50] **のデータであれば**, **出力は** 10 **である**.

ソースファイル

組み込み関数

MODULE

機能) **コマンドモジュールを登録する**.

形式) **module(mod_name)**

パラメータ) mod_name : **モジュール名**

解説)

1. **モジュール名は**, **モジュールの諸パラメータの設定されたモジュールとして宣言されているものでなければならない**.
2. **同じコマンド名を含むモジュールを登録した場合**, **新しく登録したモジュールのコマンドの方が実行されるようになる**.
3. **モジュールの定義については**, DEFINE **を参照**.
4. **標準モジュールとして**, **現在**SYSTEM, ISPP, GPM, BPS, NCS, NPE **が用意されており**, **モジュール名は定義済みである**.

関連項目) DEFINE

ソースファイル

組み込み関数

PRINT

機能) **オブジェクトの内容を表示する**.

形式)

```
print x
```

パラメータ) x : **オブジェクト** (Scalar, Series, Snapshot, String)

解説) **すべてのデータを表示する**.

ソースファイル

組み込み関数

PRINTF

機能) **オブジェクトの内容を指定した形式で表示する**.

形式) **printf(format, x [, y, ...])**

パラメータ) 1. format : **フォーマット文字列**

2. x, y ... : **オブジェクト** (Scalar, Series, Snapshot, String)

解説) **フォーマット文字列の書式は**, C **言語と同様である**.

型	フォーマット文字列
整数	%d %nd
実数	%f %nf %n.mf %g %ng %n.mg %e %ne %n.me
文字 **文字列**	%c %nd %s %ns
改行 **タブ**	%n %r %t

使用例) **整数**i, **実数**a **を表示する**.

```
% printf("%4d : %8.3f\n",i,a);
```

PROC

機能) **手続きを定義する**.

形式) **proc method([arg1,...,argn]) stmt**

パラメータ)
1. method : **手続き名**
2. arg1,...,argn : **引数名**
3. stmt : **処理** ({ } **で囲まれた複数のコマンドによる処理の記述ができる**)

解説) **手続き内で引数を書き換えた場合**, **呼出元の対応する変数に影響する**.

使用例) **変数** object **の内容を**, unix **コマンド** more **を使って表示する手続きの記述例**

```
% proc dump( object ) {
%     unix( "more " ) << object;
% }
```

関連項目) EXTERNAL

ソースファイル

組み込み関数

READ

機能) **指定した型のデータをキーボードから読み込む**.

形式) `x = read( type )`

パラメータ) 1. x : **出力オブジェクト**

2. type : **データ型を表す文字列** ("scalar", "string" **など**)

解説) x **には**, **キーボードから入力した値を指定した型に変換した値が入る**.

ソースファイル

組み込み関数

RETURN

機能) **指定されたデータを関数の呼出元に渡す**.

形式) `return stmt`

パラメータ) 1. stmt : **評価式** (**オブジェクトのみも含む**)

解説) **関数の処理結果を呼出元に渡すコマンドである**. **複数のオブジェクトは指定できない**.

関連項目) FUNC

ソースファイル

組み込み関数

SCALAR

機能) Scalar **オブジェクトを宣言する**

形式)

```
scalar x, y, z, ...
```

パラメータ) x, y, z,... : **オブジェクト名**

解説)

1. Scalar **型のオブジェクトは**, **１個の実数値が格納される変数である**.
2. **宣言したオブジェクトの大きさおよび型は**, **コマンドの出力値が格納されることにより**, **変更される可能性がある**.

ソースファイル

組み込み関数

SERIES

機能) Series **オブジェクトを宣言する**.

形式)

```
series x, y, z, ...
```

パラメータ) x, y, z,... : **オブジェクト名**

解説)

1. Series **型のオブジェクトは**, **可変長の数列を格納するものであり**, **宣言時に下位次元の大きさ** (**サブインデックス**) **を示さなければならない**. **例えば**, series x[64][32], y[100] **などのように設定する**. **前者では** $n \times 64 \times 32$ **の３次元配列**, **後者では** $n \times 100$ **の２次元配列となる**. **何も大きさを示さなかった場合には**, **１次元配列となる**. **下位次元についてはその大きさは固定である**. **最上位次元の** n **については可変であり**, **宣言直後は１となっている**.
2. **宣言したオブジェクトの大きさおよび型は**, **コマンドの出力値が格納されることにより**, **変更される可能性がある**.
3. reform **コマンドによってオブジェクトの次元数・大きさを変更することができる**.
4. **サブバッファの参照値** (x:[3] **など**) **のオブジェクトのクラスは** Snapshot **となる**.

ソースファイル

組み込み関数

SNAPSHOT

機能) Snapshot **オブジェクトを宣言する**.

形式)

```
snapshot x, y, z, ...
```

パラメータ) x, y, z,... : **オブジェクト名**

解説)

1. Snapshot **型のオブジェクトは**, **大きさは固定であり**, **外部コマンドの出力を代入しない限り**, **宣言時に示した大きさから変わることはない**.
2. reform **コマンドによってオブジェクトの次元数・大きさを変更することができる**.

ソースファイル

組み込み関数

STRING

機能) String **オブジェクトを宣言する**.

形式)

```
string x, y, z, ....
```

パラメータ) x, y, z,... : **オブジェクト名**

解説)

1. String **型のオブジェクトは**, **複数の文字列を格納する配列である**.
2. **大きさの設定方法は**, Snapshot **型の場合と同じであり**, **それぞれの要素に文字列が格納される**.
3. **宣言したオブジェクトの大きさおよび型は**, **コマンドの出力値が格納されることにより**, **変更される可能性がある**.

ソースファイル

組み込み関数

STRLEN

機能)　**文字列の長さを得る**

形式)

```
str = strlen( x )
```

パラメータ)　1. x : String **オブジェクト**

2. str : **文字列の長さ出力** (Scalar)

ソースファイル

組み込み関数

SYMBOLS

機能) **シンボルテーブルを表示する**.

形式)

```
Symbols()
```

パラメータ) **なし**

解説) **定義されているシンボル** (**オブジェクト名**, **定数名**, **モジュール名**) **を表示する**.

TYPEOF

機能) **オブジェクトのクラスを得る**

形式)

```
str = typeof( x )
```

パラメータ)

1. x : **オブジェクト** (Scalar, Series, Snapshot, String)
2. str : **オブジェクトのクラス出力** (String)

解説) **データのオブジェクトのタイプが出力される**. **出力値は**String **型であり**, **それぞれのオブジェクトクラスによって次の値が出力される**.

Scalar **型**	–	scalar
Series **型**	–	series
Snapshot **型**	–	snapshot
String **型**	–	string

ソースファイル

組み込み関数

UNDEF

機能) **指定したオブジェクトの登録を削除する**.

形式)

```
undef x [, y ... ]
```

パラメータ) x, y : **オブジェクト**

解説) *SATELLITE*シェルのシンボルテーブルから指定したオブジェクトを削除する. オブジェクトには，func, proc で定義した関数，手続きも含まれる．現在定義されているシンボルの一覧を得るコマンドは，Symbols() である．

UNIX

機能) UNIX **コマンドを実行する**.

形式) **unix(command_statements)**

パラメータ) command_statements : UNIX **コマンド文字列** (string **オブジェクト**)

解説) UNIX **コマンドの出力は**, *SATELLITE* **には**, String **オブジェクトとして渡される**.

使用例)

1. **数値データのテキストファイルを**, Series **オブジェクトとして読み込む**

```
% x = 0 + unix( "cat data.txt" )
```

2. *SATELLITE* **データを** UNIX **コマンドに渡す**

```
% unix( "more" ) << x
```

ソースファイル

組み込み関数

WELCOME

機能) *SATELLITE*起動時のメッセージを表示する.

形式) **welcome([“string”])**

パラメータ) string : メッセージの上・下側に表示するラインの文字を指定する場合には, 文字列を記述する.

解説) .setup.sl に記述しておけば, *SATELLITE*起動時に次のようなメッセージが表示される.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Welcome to The SATELLITE World
Copyright (C) 1992 - 1999, BPEL, Toyohashi University of Technology
Version 2.92
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

ソースファイル

組み込み関数

WHILE

機能) WHILE**型のループ制御をする**.

形式)

```
while( expr ) stmt1
```

パラメータ)

1. expr : **条件判定**
2. stmt1 : **処理** ({}**で囲まれた複数のコマンドによる処理の記述ができる**)

ソースファイル

組み込み関数

WRITE_TYPE

機能) **データファイル書き込みフォーマットを設定する**.

形式) **write_type(type, endian_type, datum_size)**

パラメータ)

1. type : **データファイルのフォーマット**
   "new" – *SATELLITE* **言語フォーマット**
   "old" – SATELITE MS-DOS **フォーマット**
2. endian_type : **エンディアンの種類**
   "BigEndian" – SPARC, Mips **などのプロセッサ**
   "LittleEndian" – Alpha, Intel **プロセッサなど**
3. datum_size : **データ１要素のサイズ** [byte] (2, 4, 8)

解説) old **フォーマットでは**, **データサイズは** 4 [byte] **固定である**.

使用例) **データサイズを** double(8[byte]) **で出力する場合の記述例**

```
% write_type( "new", "BigEndian", 8 )
```

ソースファイル

組み込み関数

数学ライブラリ

Mathematic Library

機能) **オブジェクトの内容の絶対値を求める**.

形式) **abs(x)**

パラメータ) 1. x : **入力オブジェクト** (Scalar, Series, Snapshot, File)

解説) 1. **処理**

$y_i = |x_i|$

ソースファイル

組み込み関数

ACOS

機能) **逆余弦を求める**.

形式) **acos(x)**

パラメータ) 1. x : **入力オブジェクト** (Scalar, Series, Snapshot, File)

解説) 1. **処理**

$y_i = \cos^{-1} x_i$

ソースファイル

組み込み関数

ASIN

機能) **逆正弦を求める**.

形式) **asin(x)**

パラメータ) 1. x : **入力オブジェクト** (Scalar, Series, Snapshot, File)

解説) 1. **処理**

$y_i = \sin^{-1} x_i$

ソースファイル

組み込み関数

ATAN

機能) **逆正接を求める**.

形式) **atan(x)**

パラメータ) 1. x : **入力オブジェクト** (Scalar, Series, Snapshot, File)

解説) 1. **処理**

$y_i = \tan^{-1} x_i$

ソースファイル

組み込み関数

ATAN2

機能) **逆正接を求める**.

形式) **atan2(y, x)**

パラメータ) 1. y : **入力オブジェクト** (Scalar, Series, Snapshot, File)

2. x : **入力オブジェクト** (Scalar, Series, Snapshot, File)

解説) 1. **処理**

$z_i = \tan^{-1} \frac{y_i}{x_i}$

ソースファイル

組み込み関数

COS

機能) **余弦を求める**.

形式) cos(x)

パラメータ) 1. x : **入力オブジェクト** (Scalar, Series, Snapshot, File)

解説) 1. **処理**

$y_i = \cos x_i$

ソースファイル

組み込み関数

EXP

機能) **データの指数を計算する**.

形式) exp(x)

パラメータ) x : **入力オブジェクト**

解説) 1. **処理**

$y_i = e^{x_i}$

ソースファイル

組み込み関数

EXP2

Mathematic Library

機能) **データの指数を計算する**.

形式) exp2(x)

パラメータ) x : **入力オブジェクト** (Scalar, Series, Snapshot)

解説) 1. **処理**

$y_i = 2^{x_i}$

ソースファイル

組み込み関数

INT

機能) **オブジェクトの内容の小数以下を切捨てた値を求める**.

形式) **int(x)**

パラメータ) 1. x : **入力オブジェクト** (Scalar, Series, Snapshot, File)

解説) 1. **処理**

$y_i = \lceil x_i \rceil$

ソースファイル

組み込み関数

LOG

Mathematic Library

機能) **データの自然対数を計算する**.

形式) log(x)

パラメータ) x : **入力オブジェクト** (Scalar, Series, Snapshot)

解説) $y_i = \log_e x_i$

ソースファイル

組み込み関数

LOG2

Mathematic Library

機能) **データの**, **底が** 2 **の対数を計算する**.

形式) **log2(x)**

パラメータ) x : **入力オブジェクト** (Scalar, Series, Snapshot)

解説) $y_i = \log_2 x_i$

ソースファイル

組み込み関数

LOG10

機能) **データの常用対数を計算する**.

形式) **log10(x)**

パラメータ) x : **入力オブジェクト** (Scalar, Series, Snapshot)

解説) $y_i = \log_{10} x_i$

ソースファイル

組み込み関数

機能) **剰余を求める**.

形式1) `mod( x, y )`

形式2) `x % y`

パラメータ) x , y : **入力オブジェクト** (Scalar, Series, Snapshot)

解説) $z_i = x_i \% y_i$

ソースファイル

組み込み関数

POW

機能)　　**累乗を求める**

形式1)　　**pow(x, y)**

形式2)　　x^y

パラメータ) x , y : **入力オブジェクト** (Scalar, Series, Snapshot)

解説)　　$z_i = x_i^{y_i}$

ソースファイル

組み込み関数

SGN

機能) **符号を得る**.

形式) **sgn(x)**

パラメータ) 1. x : **入力オブジェクト** (Scalar, Series, Snapshot, File)

解説) 1. **処理**

$y_i = \mathrm{sgn}(x_i) = x_i/|x_i|$

-1, 0, 1 **のいずれかを返す**. **実際の処理では符号を判定している**.

ソースファイル

組み込み関数

SIN

機能) **正弦を求める**.

形式) **sin(x)**

パラメータ) 1. x : **入力オブジェクト** (Scalar, Series, Snapshot, File)

解説) 1. **処理**

$y_i = \sin x_i$

ソースファイル

組み込み関数

SQRT

機能) **平方根を求める**.

形式) `sqrt( x )`

パラメータ) 1. x : **入力オブジェクト** (Scalar, Series, Snapshot)

解説) $y_i = \sqrt{x_i}$

ソースファイル

組み込み関数

TAN

機能) **正接を求める**

形式) **tan(x)**

パラメータ) x : **入力オブジェクト** (Scalar, Series, Snapshot)

解説) $y_i = \tan x_i$

ソースファイル

組み込み関数

ユーティリティ

SL_UTIL

EXIST

機能)　　**ファイルがあるか調べる関数**.

形式)　　flag = exist(filename)

パラメータ)　1. filename : **対象ファイル名** (string)

2. flag : **結果**. (flag = 1 : **存在**, flag = 0 : **存在しない**)

ソースファイル　　slutil.sl

DUMP

機能) **変数の内容を表示する**.

形式) dump(x)

パラメータ) x : **入力オブジェクト**

解説) **表示には** UNIX **コマンド** less **を用いている**.

ソースファイル slutil.sl

EXTRACT

機能) **変数のインデックスを変更し**, **元の値をそのままはめ込む**.

形式) **y = extract(x,index)**

パラメータ)
1. x : **入力オブジェクト**
2. index : **変更後のインデックス**
3. y : **出力オブジェクト**

解説) reform **では**, **大きさは変更できるが**, **値は先頭から順番に詰め込まれる**. **本関数は元のデータの並びを保ち**, **且つインデックスを変更したい場合に用いるとよい**.

参照) REFORM

ソースファイル func_lib.sl

FILEINDEX

機能)　　**データファイルのインデックスを得る**.

形式)　　**index = fileindex(filename)**

パラメータ)　1. filenmae : **対象ファイル名** (string)

2. index : **対象ファイル名のインデックス**

ソースファイル　　func_lib.sl

HGET

機能) get **の機能を内部コマンドのみで実現した関数**.

形式) **y = hget(x, pos)**

パラメータ)
1. x : **入力オブジェクト**
2. pos : **データの座標**
3. y : **指定した位置のデータ** (scalar)

参照) GET

ソースファイル func_lib.sl

INDEXSIZE

機能) **変数の全体の要素数を求める**

形式) **IndexSize(x)**

パラメータ) x : **入力オブジェクト**

解説) length(x) **では**, **多次元の場合の全体の要素数が分からないが**, **この関数を使えば**, **全体の要素数がわかる**.

ソースファイル slutil.sl

PUTS

機能) C **言語の** puts **と同機能**.

形式) **puts(x)**

パラメータ) x : **オブジェクト**

ソースファイル slutil.sl

REVTIME

SL_UTIL

機能) **時間軸方向を逆転する**.

形式) **y = revTime(x)**

パラメータ) 1. x : **入力オブジェクト** (series)
2. y : **出力オブジェクト** (series)

解説) **多次元データでは**, reverse **の結果と異なる**.

参照) REVERSE

ソースファイル slutil.sl

Scalar

SL_UTIL

機能) **指定した変数を**, scalar **型の値にして返す**.

形式) **y = Scalar(x)**

パラメータ) 1. x : **入力オブジェクト**

2. y : **出力** (scalar)

解説) 1 **つの実数値が返される**. series **などは**, **先頭の値が変換される**.

参照) Snapshot, Series, String

ソースファイル slutil.sl

Series

SL_UTIL

機能) **指定した変数を** series **型にした値を返す**.

形式) **y = Series(x)**

パラメータ) 1. x : **入力**

2. y : **出力** (series)

参照) Scalar, Snapshot, String

ソースファイル slutil.sl

Snapshot

SL_UTIL

機能) **指定した変数を** snapshot **型にした値を返す**.

形式) **y = Snapshot(x)**

パラメータ) 1. x : **入力**

2. y : **出力** (snapshot)

参照) Scalar, Series, String

ソースファイル slutil.sl

SPRINTF

機能) **数値を指定した形式で, 文字列に変換する.**

形式)

```
str = sprintf( format, x )
```

パラメータ)

1. format : **出力フォーマット** (string)
2. x : **データ** (scalar)
3. str : **出力文字列** (string)

解説)

1. format **は**, printf **と同じでよい**. **ただし**, **小数点以下の桁数指定のみ有効**. **スペースはいくつあっても** 1 **つになる**.
2. **指定できる変数は** 1 **つだけである**.

参照) Sprintf

ソースファイル slutil.sl

Sprintf

SL_UTIL

機能) **多引数版** sprintf **関数**

形式) **Sprintf(format, x1, [x2, ···, x15])**

パラメータ) 1. format : **出力フォーマット** (string)

2. x1 : **出力したいオブジェクト**

解説) format **は**, printf **と同じ記述方法**. **引数が多数指定できる** (15 **個まで**). **フォーマット文字列に含まれるスペースは**, **その数だけ出力される**.

ただし, **引数に指定できるのは**, scalar **値のみであり**, series, snapshot **値が指定された場合は**, **その先頭の値のみを処理する**.

参照) sprintf

ソースファイル Sprintf.c

String

SL_UTIL

機能) **指定した変数を** String **型にした値を返す**.

形式) **y = String(x)**

パラメータ) 1. x : **入力**

2. y : **出力** (string)

参照) Scalar, Snapshot, Series

ソースファイル slutil.sl

TEXT2STRING

SL_UTIL

機能) **テキストファイルの内容を** string **データに変換する**.

形式 | **str = text2string(filename)**

パラメータ) 1. filename : **テキストファイル名** (string)

2. str : **出力** (string)

解説) **データの区切り文字は**, **スペース**, **タブ**, **改行 である**.

参照) TEXT2BUFFER, BUFFER2TEXT

ソースファイル slutil.sl

UNITMAT

機能) **単位行列を生成する**.

形式) **Imat = unitMat(dim)**

パラメータ) 1. dim : **次元数** (scalar)

2. Imat : **出力** (snapshot)

ソースファイル slutil.sl

TOLOWER

機能) **小文字に変換した文字列を返す**.

形式) **toLower(string)**

パラメータ) string : **入力文字列** (string)

使用例)
```
% toLower("toLower");
tolower
```

参照) TOUPPER

ソースファイル toLower.c

TOUPPER

機能) **大文字に変換した文字列を返す**.

形式) **toUpper(string)**

パラメータ) string : **入力文字列** (string)

使用例)
```
% toUpper("toUpper");
TOUPPER
```

参照) TOLOWER

ソースファイル toUpper.c

基本コマンド

SYSTEM

BM

機能) **バッファモニター**

形式) bm(x)

パラメータ) x : **オブジェクト**

解説)

1. **オブジェクト** x **の変化を逐次グラフ表示する**.
2. **バッファモニターを実行中に**, **モニタリング対象のオブジェクトを再宣言すると**, **バッファモニターは機能しなくなるため**, **注意が必要**.

ソースファイル bm/*.c

CUT

SYSTEM

機能) **オブジェクトの一部を取り出す**.

形式) **y = cut(x, start, end)**

パラメータ)
1. x : **入力オブジェクト** (Series , Snapshot)
2. y : **出力オブジェクト** (Series)
3. start : **切り取すデータの始点座標**
4. end : **切り出すデータの終点座標** (>start)

解説)
1. x **の** start **から，** end **の位置までのデータを取り出す**.
2. start, end **には**, x **が多次元データならば** index (Series) **を**, 1 **次元データならば** data point (Scalar) **を設定する**.

使用例) 2 **次元の場合**

```
[0]:[0]%  1   2   3   4   5
[1]:[0]%  6   7   8   9  10
```

という x **に対し**,
`cut(x,(0,1),(0,3))` **を実行すると**,

```
[0]:[0]%  2   3   4
```

ソースファイル cut.c

FILL

SYSTEM

機能) **オブジェクトの指定した範囲を指定した値で埋める**.

形式) **y = fill(x, start, end, value)**

パラメータ)

1. x : **入力オブジェクト** (Series , Snapshot)
2. y : **出力オブジェクト** (Series)
3. start : **始点座標**
4. end : **終点座標** (>start)
5. value : **埋める数値** (Scalar)

解説)

1. x **の** start **から，** end **の位置までの範囲を** value **の値で満たす** (value = 0 **の場合は**, "zero" **コマンドと等価**).
2. start, end **には**, x **が多次元データならば** index (Series) **を**, 1 **次元データならば** data point (Scalar) **を設定する**.

使用例) 2**次元の場合**

```
[0]:[0]%  1   2   3   4   5
[1]:[0]%  6   7   8   9  10
```

という x **に対し**,

`fill(x,(0,1),(0,3),30)` **を実行すると**,

```
[0]:[0]%  1  30  30  30   5
[1]:[0]%  6   7   8   9  10
```

ソースファイル fill.c

FIND

SYSTEM

機能) **入力オブジェクトから指定した値に最も近いデータを見つけ**, **値とその位置を表示し**, **位置を** Series **オブジェクトとして返す.**

形式) **ip = find(x, val, num)**

パラメータ)

1. x : **入力オブジェクト** (Series, Snapshot)
2. ip : **データの位置** (Series)
3. val : **捜し出す値**
4. num : **捜し出すデータの個数**

解説)

1. **入力オブジェクト内から** val **に最も近い値を持つインデックスの系列を取得する**.
2. **インデックスの系列は**, **次元数分の要素を持ち**, **時間方向に** num **個得られる**.
3. **取得するデータ数が** 1 **個で**, **かつ** 1 **次元データの位置を示す場合には戻り値は**, Scalar **オブジェクトである**.

使用例)

1 **次元の場合**

```
[0]:%  11   12   13   14   15
```

という x **に対し**,

`find(x,14,1)` **を実行すると**,

```
3
```

`find(x,14,3)` **を実行すると**,

```
[0]:[0]%  3
[1]:[0]%  4
[2]:[0]%  2
```

2 **次元の場合**

```
[0]:[0]%  11   12   13   14   15
[1]:[0]%  16   17   18   19   20
```

という x **に対し**,

`find(x,14,3)` **を実行すると**,

```
[0]:[0]%  0   3
[1]:[0]%  0   4
[2]:[0]%  0   2
```

ソースファイル find.c

GET

SYSTEM

機能) **オブジェクトの指定した位置の値を得る**.

形式) **y = get(x, position)**

パラメータ)
1. x : **入力オブジェクト** (Series , Snapshot)
2. y : **指定した位置のデータ値** (Scalar)
3. position: **データの座標**

解説)
1. x **の** position **の位置のデータ値を返す**.
2. position **には**, x**が多次元データならば** index (Series) **を**, 1**次元データならば** data point (Scalar) **を設定する**.

使用例) 2**次元の場合**

```
[0]:[0]%  1   2   3   4   5
[1]:[0]%  6   7   8   9  10
```

という x **に対し**,
`get(x,(0,3))` **を実行すると**,

```
[0]:[0]%  4
```

ソースファイル get.c

HEADER

SYSTEM

機能) **データファイルの情報を表示, 変更する**.

形式) **header(file_name, mode)**

パラメータ)

1. file_name : **データファイル名**
2. mode : **モード**
   (a) 0 : **画面に表示する**.
   (b) 1 : **変更する**.

解説)

1. **データファイル** file_name **の情報を**, mode **に従って処理する**.
2. **データファイル名は**, “ ” **で囲む必要がある**.
3. **表示・変更される情報には**, Byte Order, Data size, Owner, Date, Comment, Sampling frequency, Dimension, Index **などがある**.

ソースファイル header.c

HELP

SYSTEM

機能) **コマンドマニュアルを表示する**.

形式) help(com_name)

パラメータ) com_name : **コマンド名**

解説)

1. **コマンド** com_name **の機能**, **形式**, **パラメータなどを表示する**.
2. **コマンド名は**, “ ” **で囲む必要がある**.

ソースファイル help.c

機能) **データを指定したステップ数で間引く**.

形式) **y = mabi(x, step)**

パラメータ)
1. x : **入力時系列** (Series)
2. y : **出力時系列** (Series)
3. step : **間引きの間隔** (> 1)

解説)
1. **入力時系列の** step-1 **点分のデータを間引いて出力する**.
2. step=0 **および** 1 **の場合はデータは変化しない**.
3. step **には**, x **が多次元データならば** Series **を**, 1 **次元データならば** Scaler **を設定する**.
4. **多次元オブジェクトの入力に対しては**, **次のように間引かれる**.
   5×10 **の** 2 **次元オブジェクトの場合**

   y = mabi(x, (2, 3));

   y **には**,

$$\begin{array}{llllll} y:[0] & x_{00} & x_{03} & x_{06} & x_{09} \\ y:[1] & x_{20} & x_{23} & x_{26} & x_{29} \\ y:[2] & x_{40} & x_{43} & x_{46} & x_{49} \end{array}$$

   が, **出力される**.

使用例) 1 **次元の場合**

```
[0]:%  1   2   3   4   5   6   7   8   9   10
```

という x **に対し**,
mabi(x,2) **を実行すると**,

```
[0]:%  1   3   5   7   9
```

2 **次元の場合**

```
[0]:[0]%  11   12   13   14   15
[1]:[0]%  16   17   18   19   20
[2]:[0]%  21   22   23   24   25
[3]:[0]%  26   27   28   29   30
[4]:[0]%  31   32   33   34   35
```

という x **に対し**,

mabi(x,(2,3)) **を実行すると**,

```
[0]:[0]%  11   14
[1]:[0]%  21   24
[2]:[0]%  31   34
```

ソースファイル mabi.c

MAX

SYSTEM

機能) **データの最大値を得る**.

形式) y = max(x)

パラメータ) 1. x : **入力オブジェクト** (Series, Snapshot)

2. y : **最大値** (Scalar)

ソースファイル max.c

MAXPOS

SYSTEM

機能) **データの最大値のデータ位置を得る**.

形式) **y = maxpos(x, num)**

パラメータ)
1. x : **入力オブジェクト** (Series, Snapshot)
2. y : **位置格納オブジェクト** (Series)
3. num : **取得するデータ位置の数** (≥ 1)

解説)
1. **入力オブジェクト内で最も大きな値を持つインデックスの系列を取得する**.
2. **インデックスの系列は**, **次元数分の要素を持ち**, **時間方向に** num **個得られる**.
3. **取得するデータ数が１個で**, **かつ１次元データの位置を示す場合には戻り値は**, Scalar **オブジェクトである**.

使用例) 1**次元の場合**

```
[0]:%  13   12   15   11   14
```
という x **に対し**,
`maxpos(x,1)` **を実行すると**,
2
`maxpos(x,3)` **を実行すると**,
```
[0]:[0]%  2
[1]:[0]%  4
[2]:[0]%  0
```

2**次元の場合**

```
[0]:[0]%  13   17   15   11   18
[1]:[0]%  20   12   16   19   14
```
という x **に対し**,
`maxpos(x,3)` **を実行すると**,
```
[0]:[0]%  1   0
[1]:[0]%  1   3
[2]:[0]%  0   4
```

ソースファイル maxpos.c

MERGE

SYSTEM

機能) 2 **つのオブジェクトのデータを連結する**.

形式) **z = merge(x, y)**

パラメータ)
1. x : **入力オブジェクト** 1 (Series , Snapshot)
2. y : **入力オブジェクト** 2 (Series , Snapshot)
3. z : **出力オブジェクト** (Series)

解説)
1. z **には**, x **の終端に** y **を連結したデータが出力される**.
2. **多次元データの場合には**, **サブインデックスが一致しなければ連結しない**.

使用例) 2 **次元の場合**

x :
```
[0]:[0]%  1   2
[1]:[0]%  3   4
[2]:[0]%  5   6
[3]:[0]%  7   8
```

y :
```
[0]:[0]%  30   31
[1]:[0]%  32   33
[2]:[0]%  34   35
```

に対し, merge(x,y) **を実行すると**,
```
[0]:[0]%  1   2
[1]:[0]%  3   4
[2]:[0]%  5   6
[3]:[0]%  7   8
[4]:[0]%  30   31
[5]:[0]%  32   33
[6]:[0]%  34   35
```

ソースファイル merge.c

MIN

SYSTEM

機能) **データの最小値を得る**.

形式) $\mathbf{y = min(\ x\ )}$

パラメータ) 1. x : **入力オブジェクト** (Series , Snapshot)

2. y : **最小値** (Scalar)

ソースファイル min.c

MINPOS

SYSTEM

機能) **データの最小値のデータ位置を得る**.

形式) **y = minpos(x, num)**

パラメータ)
1. x : **入力オブジェクト** (Series , Snapshot)
2. y : **位置格納オブジェクト** (Series)
3. num : **取得するデータ位置の数** (≥ 1)

解説)
1. **入力オブジェクト内で最も小さな値を持つインデックスの系列を取得する**.
2. **インデックスの系列は**, **次元数分の要素を持ち**, **時間方向に** num **個得られる**.
3. **取得するデータ数が１個で**, **かつ１次元データの位置を示す場合には戻り値は**, Scalar **オブジェクトである**.

使用例) 1**次元の場合**

```
[0]:%  13   12   15   11   14
```
という x **に対し**,
`minpos(x,1)` **を実行すると**,
3
`minpos(x,3)` **を実行すると**,
```
[0]:[0]%  3
[1]:[0]%  1
[2]:[0]%  0
```

2**次元の場合**

```
[0]:[0]%  13   17   15   11   18
[1]:[0]%  20   12   16   19   14
```
という x **に対し**,
`minpos(x,3)` **を実行すると**,
```
[0]:[0]%  0   3
[1]:[0]%  1   1
[2]:[0]%  0   0
```

ソースファイル minpos.c

PUT

SYSTEM

機能) **オブジェクトの一部に**, **別のデータを入れる**.

形式) **z = put(x, y, index)**

パラメータ)

1. x : **入力オブジェクト** (Series, Snapshot)
2. y : **挿入オブジェクト** (Series, Snapshot)
3. z : **出力オブジェクト** (Series)
4. index : y **を書き込む座標**

解説)

1. x **の** index **の位置から，** y **の内容を上書きしたデータを出力する**. **ただし** z **は**, x **の時間軸方向の長さに依存する**.
2. x **が多次元ならば** index **は** Series **値で**, 1 **次元ならば** Scalar **値で設定する**.
3. **多次元データの場合には**, **サブインデックスが一致しなければ上書きしない**.

使用例) 2 **次元の場合**

x :

```
[0]:[0]%  1   2
[1]:[0]%  3   4
[2]:[0]%  5   6
[3]:[0]%  7   8
[4]:[0]%  9  10
```

y :

```
[0]:[0]%  30   31
[1]:[0]%  32   33
[2]:[0]%  34   35
```

に対し, `put(x,y,(1,0))` **を実行すると**,

```
[0]:[0]%  1    2
[1]:[0]% 30   31
[2]:[0]% 32   33
[3]:[0]% 34   35
[4]:[0]%  9   10
```

が得られる.

ソースファイル put.c

REFORM SYSTEM

機能) **オブジェクトのフォーマットを変更する**.

形式) **y = reform(x, index)**

パラメータ)
1. x : **入力オブジェクト** (Series , Snapshot)
2. y : **出力オブジェクト** (Series)
3. index : **データサイズ** (Scalar , Series)

解説)
1. **指定したサイズが入力オブジェクトより大きい場合には**, **データの後部に** 0 **詰めを行ない**, **入力オブジェクトより小さい場合には**, **データの削除を行なって**, **オブジェクトのフォーマットを変更する**.
2. x **が多次元ならば** index **は** Series **値で**, 1 **次元ならば** Scalar **値で設定する**.

使用例)

`x = (1,2,3,4,5,6,7,8,9,10)` **に対し**,
`reform(x,(2,5))` **を実行すると**,
```
[0]:[0]%  1   2   3   4   5
[1]:[0]%  6   7   8   9  10
```
`reform(x,(2,7))` **を実行すると**,
```
[0]:[0]%  1   2   3   4   5   6   7
[1]:[0]%  8   9  10   0   0   0   0
```
`reform(x,(2,3))` **を実行すると**,
```
[0]:[0]%  1   2   3
[1]:[0]%  4   5   6
```

ソースファイル reform.c

REVERSE

SYSTEM

機能) **入力データの配列を逆にしたデータを得る**.

形式) **y = reverse(x)**

パラメータ) 1. x : **入力オブジェクト** (Series , Snapshot)

2. y : **出力オブジェクト** (Series)

使用例) **多次元データに使用した場合**, **次のような結果になる**.

2**次元の場合**

```
[0]:[0]%  1   2   3   4   5      [0]:[0]% 10   9   8   7   6
[1]:[0]%  6   7   8   9  10      [1]:[0]%  5   4   3   2   1
```

ソースファイル reverse.c

ROTATE

SYSTEM

機能) **オブジェクトの内容を**, **指定した位置が先頭となるように移動する**.

形式) **y = rotate(x, index)**

パラメータ)
1. x : **入力オブジェクト** (Series, Snapshot)
2. y : **出力オブジェクト** (Series)
3. index : **先頭にする座標**

解説)
1. **データの先頭から** index **までのデータは**, **オブジェクトの最後に移動される**.
2. index **が負であった場合**, |index| **の位置に入力の先頭が移動される**.
3. x **が多次元ならば** index **は** Series **値で**, 1 **次元ならば** Scaler **値で設定する**.

使用例) 2 **次元の場合**

```
[0]:[0]%  1   2   3   4   5
[1]:[0]%  6   7   8   9  10
```
という x **に対し**,
`rotate(x,(0,3))` **を実行すると**,
```
[0]:[0]%  4   5   1   2   3
[1]:[0]%  9  10   6   7   8
```
`rotate(x,(0,-3))` **を実行すると**,
```
[0]:[0]%  3   4   5   1   2
[1]:[0]%  8   9  10   6   7
```

ソースファイル rotate.c

機能) **サンプリング周波数を変更する**.

形式) **sam(frequency)**

パラメータ) frequency : **サンプリング周波数** (Scalar)

解説)

1. **サンプリング周波数を**, **指定した値に変更する**. **ただし**, frequency **のデフォルトは** 1000.0 **である**.
2. **戻り値には**, **設定値を返す**.
3. **引数に** 0 **以下の数値を渡した場合**, **サンプリング周波数の変更は行なわれない**.
4. **ここで設定するサンプリング周波数は**, ISPP **モジュールのコマンドや**, GPM **モジュールの** "graph" **コマンドに参照される**.

ソースファイル sam.c

WAIT

SYSTEM

機能) **リターンキーが押されるまで待つ**.

形式)

```
wait( )
```

パラメータ) **なし**.

解説) **デモンストレーションなどで**, **画面を静止状態にしたい場合などに用いる**.

ソースファイル wait.c

ZERO

SYSTEM

機能) **データの指定した範囲に**0**を詰める**.

形式) **y = zero(x, start, end)**

パラメータ)
1. x : **入力オブジェクト** (Series, Snapshot)
2. y : **出力オブジェクト** (Series, Snapshot)
3. start : **始点データ座標**
4. end : **終点データ座標** (>start)

解説)
1. x **の** start **から** end **の範囲を**0**にした結果が** y **に出力される**.
2. start, end **には**, x **が多次元ならば** index (Series) **を**, 1**次元ならば** data point (Scalar) **を設定する**.

使用例) 2**次元の場合**

```
[0]:[0]%  1   2   3   4   5
[1]:[0]%  6   7   8   9  10
```

という x **に対し**,
`zero(x,(0,1),(0,3))` **を実行すると**,

```
[0]:[0]%  1   0   0   0   5
[1]:[0]%  6   7   8   9  10
```

ソースファイル zero.c

ディジタル信号処理

ISPP

AKIMA

機能) Akima **の補間法により，**1 **次元系列の入出力データ対** x, y **を補間した系列** xip, yip **を求める．**

形式1) **yip = akima (x, y, xip)**

形式2) **yip = akima (x, y, dpt [, xip2])**

パラメータ)

1. x : **入力系列** (1 **次元** Series)
2. y : **出力系列** (1 **次元** Series)
3. dpt : **補間後のデータ点数** (>0), **サンプリング周波数からデータ点数，補間後の入力系列を求める** (=0).
4. xip : **補間後の入力系列** (1 **次元** Series)
5. yip : **補間後の出力系列** (1 **次元** Series)
6. xip2 : **自動生成された補間後の入力系列** (1 **次元** Series)

解説)

1. x, y **は同じデータ点数でなければならない．**
2. x **および** xip **は，ともに** $x_i < x_{i+1}$ **の関係を満たしていなければならない．**

形式1 **について**:
- xip **の系列と対になる** yip **が求められる．**
- xip **の値の範囲は** x **の値の範囲を超えてはならない．**

形式2 **について**:
- dpt = 0 **の場合は，サンプリング周波数をもとに，以下の定義により，**xip **が自動生成される．**

$$xip_0 = x_0, xip_i = x_0 + step \times i, ...xip_{dpt-1} = x_{n-1}$$

ここで，$step = 1.0/sf, dpt = (x_{n-1} - x_0) \times sf + 1,$

ただし，sf **はサンプリング周波数，**n **は** x **のデータ点数．**

- dpt > 0 **の場合は，上式の** step **を次式として** xip **が生成される．**

$dpt > 1$ **のとき** $step = (x_{n-1} - x_0)/(dpt - 1)$

$dpt = 1$ **のとき** $step$ **は** 0

- **第** 4 **引数** xip2 **には，自動生成された補間後の入力系列** xip **が格納される．**

参照) SPLINE

ソースファイル akima.c

ARAND

機能) **任意の経験分布関数に従う乱数を生成する**.

形式) **x = arand(dpt, seed, f, min, max)**

パラメータ)

1. x : **出力時系列** (Series)
2. dpt : **出力データ点数**
3. seed : **乱数の初期値** (**奇数**)
4. f : **経験分布関数バッファ** (Series, Snapshot)
5. min : **経験分布関数定義域下限** (**乱数最小値**)
6. max : **経験分布関数定義域上限** (**乱数最大値**)

解説)

1. **乱数のシードは**, M **系列を用いて生成している**.
2. **乱数生成のアルゴリズムは**, **逆関数法を用いている**.
3. **経験分布関数** f **の値域は**, $0 \leqq f \leqq 1$ **の区間に入っていなければならない**.

使用例) **コーシー分布に従う乱数を発生させる**.

```
func cauchy(x) {                          #
  return 1 / (PI * (x^2 + 1));            #  コーシー分布の
}                                         #       密度関数定義

series f;
x = (0~999) / 1000 - 0.5;                 #  分布関数の定義域設定
ff = cauchy(x);                           #  密度関数の値設定
a = integ(ff,f);                          #  密度関数を積分する
f = f / a;                                #  経験分布関数の設定
data = arand(1024,1,f,min(x),max(x));     #  乱数生成
```

ソースファイル arand.c

ARGEN

機能) **線形** AR **モデルに従う信号を生成する**.

形式) **x = argen(para, dpt)**

パラメータ)

1. para : **パラメータ** (Scalar, Series, Snapshot)
   - 0 : **パラメ- タを会話的に設定し**, **オブジェクト** x **に書き込む**.
   - **オブジェクト名** : **パラメ- タを** para **から読み込んで実行する**.
2. x : **出力時系列** (Series)
3. dpt : **出力デ- タ点数** (Scalar)

解説)

1. para**が**0**の場合はパラメータを会話的に設定し**, **出力オブジェクトに返す**. para **が オブジェクト名の場合は**, **指定したパラメータバッファからパラメータを読み込んで**, AR**信号を生成する**.
2. **パラメ- タ設定は**, **画面にコメントが表示されるので**, **それに従って数値を入力すればよい**.
3. **パラメータのフォーマットは**, **次の通りである**.

   [**0**] **入力正規乱数の初期値**(**NRAND コマンドの**init **と同じく**, **奇数**)
   [**1**] Ignored Iteration (ii)
   [**2**] AR **モデルの次数** (n)
   [**3**] AR **モデルの係数** (**次数** 1)
   ⋮
   [**n+2**] AR **モデルの係数** (**次数** n)
   [**n+3**] **入力正規乱数の分散** (**入力分散**)
4. **はじめの** ii **個は出力されず**, **その後の** dpt **個のデータが** x **に出力される**.
5. **入力正規乱数は** NRAND **と同じモジュールを使用**.
6. **入力分散が設定されていない場合は** 1.0 **とする**.
7. BURG, LEVIN **コマンドで得られたパラメータオブジェクトは直接使用できない**(**以下の使用例を参照**).

使用例)

$$x_k + 0.8x_{k-1} = e_k, \quad e_k \sim N(0.0, 2.0)$$

に従う信号を生成する.

- **パラメータ バッファ** `para` **の生成**

```
% para = argen(0,0);
 Init : 3                    #  入力乱数の初期値(奇数)
 Ignored Iteration : 10      #  Ignored Iteration
 Order of AR : 1             #  AR モデルの次数
 AR Coef[ 1] : 0.8           #  AR 係数
 Input Variance : 2.0        #  入力分散

% para                       #  設定されたパラメータ
[0]:%  3  10  1  0.8  2      #  バッファの内容
```

- **信号の生成**

```
% x = argen(para,1024)       #  para の内容に従って 1024
                             #  点データを生成
```

- BURG,LEVIN **コマンドで求めたパラメータ** para **を使用する場合**(**入力分散** 1.0)

```
% tmp = (3,10,length(para))  #  Init, Order を格納
% para2 = merge(tmp,para);
% para2:[length(para2)] = 1.0 #  入力分散
```

信号生成には para2 **を用いる**.

参照) BURG, LEVIN

ソースファイル argen.c

AVERAGE

機能) **指定したオブジェクトの平均値を求める**.

形式) y = average(x)

パラメータ) 1. x : **入力オブジェクト** (Series , Snapshot)

2. y : **平均値** (Scalar)

ソースファイル average.c

BPBTW

機能) **バタワース特性のIIR型バンドパスフィルタを設計する**.

形式) **gain = bpbtw(fc1, fc2, order, zr, zi, pr, pi)**

パラメータ)

1. fc1 : **遮断周波数**(**低周波数側**) (**入力**) [Hz] (Scalar)
2. fc2 : **遮断周波数**(**高周波数側**) (**入力**) [Hz] (Scalar)
3. order : **フィルタ次数** (**入力**) (Scalar)
4. zr : **零点**(**実数部**) **出力** (Series)
5. zi : **零点**(**虚数部**) **出力** (Series)
6. pr : **極**(**実数部**) **出力** (Series)
7. pi : **極**(**虚数部**) **出力** (Series)
8. gain : **ゲイン** (Scalar)

解説) **設計値は**, iircoef **を使用すれば直接型**IIR **フィルタの係数に変換できる**.

使用例) ISPP User's Manual 3.2 **節 フィルタ処理 参照**.

参照) LPBTW, HPBTW, IIRCOEF

ソースファイル bpbtw.c

BURG

機能) **時系列に線形** AR **モデルをあてはめ**, Burg **法を用いて** AR **係数を求め**, **パワースペクトルを出力する**.

形式) **burg(x, pw, ret1, mode, odr, dpt, tola, ret2)**

パラメータ)

1. x : **入力時系列** (Series , Snapshot)
2. pw : **出力時系列**:**パワースペクトル** (Series)
3. ret1 : **出力** 1 (Series)
4. mode : **モード** (0, "A", "E", "K", "F", "C")
5. odr : **モデルの最大次数**
6. dpt : **パワースペクトルの出力データ点数**
7. tola : **次数を下げる基準** (0, 1, 2 or 3)
8. ret2 : **出力** 2 (Series)

解説)

1. **各モードの計算方式と出力の関係は以下の通り**.
   - 0 AIC_{min} **で計算** : **出力** 1 = AIC, **出力** 2 = AR **係数**
   - A AIC_{min} **で計算**. **ただし** AIC_{min} **の値と** tola **以内であれば**, **次数の少ない方を選択する**. **出力は** "0" **と同じ**.
   - E odr **の次数で計算** : **出力** 1 = **誤差分散**, **出力** 2 = AR **係数**
   - K odr **の次数で計算** : **出力** 1 = **反射係数**, **出力** 2 = AR **係数**
   - F odr **の次数で計算** : **出力** 1 = **反射係数**, **出力** 2 = **誤差系列**
   - C odr **の次数で計算** : **出力** 1 = **出力** 2 = AR **係数**
2. **入力データは**, **DCCUT コマンドにより直流成分を除去しておく**.

使用例)

$$x_k + 0.8x_{k-1} = e_k, \quad e_k \sim N(0.0, 1.0)$$

に従う信号のパワースペクトルと AR **係数を求める** (odr:**固定**).

```
% x = argen(para,1024);                 #  信号の生成.

% series pw,sigma,coef;                 #  Series 変数の宣言.
% burg(x,pw,sigma,"E",1,64,1,coef);     #  推定.
odr= 1
ndpt= 64
tola= 1.000000
```

```
p[ 1 ] = 0.956714
  AR COEFFICIENT(  1)=      0.7914491

% coef
[0]:%       0.7914
```

参照)　　LEVIN

ソースファイル　　burg.c

CEP

機能) **指定したオブジェクトの複素ケプストラムを算出する**.

形式) **cep(x, y, u, v)**

パラメータ)
1. x : **入力時系列**:**実部** (Series , Snapshot)
2. y : **入力時系列**:**虚部** (Series , Snapshot)
3. u : **出力時系列**:**実部** (Series)
4. v : **出力時系列**:**虚部** (Series)

解説)
1. y **に** 0 **を入力した場合は**, **虚部のデータに全て** 0 **が入力されたとして処理する**.
2. **出力** u,v **は予め宣言しておく必要がある**.

参照) ICEP

ソースファイル cep.c

DCCUT

機能) **指定したオブジェクトの内容の直流成分を除去する**.

形式) $\mathbf{y = dccut(\ x\ )}$

パラメータ) 1. x : **入力時系列** (Series , Snapshot)

2. y : **出力時系列** (Series)

解説) **データの平均値を引いたものが出力される**.

ソースファイル dccut.c

DET

機能) 2 **次元データを行列とみなし**, **行列式の値を計算する**.

形式)

$y = \mathbf{det}(\ x\ )$

パラメータ) 1. x : **入力オブジェクト** (Series, Snapshot)

2. y : **計算値** (Scalar)

解説) 1. **正方行列のみ計算可能**

2. **３次元以上のデータは**, **下位の２次元データを配列とみなし**, **すべての配列について行列式を計算し**, Series **を返す**.

使用例) 3 **次元以上のデータの場合**

```
% matrix                                    #  入力データ
[0]:[0][0]%             1           0
[0]:[1][0]%             0           1

[1]:[0][0]%             0           1
[1]:[1][0]%             1           0

[2]:[0][0]%             1           2
[2]:[1][0]%            -2           1

% det(matrix);
[0]:%           1          -1           5  #  計算結果
```

参照) INV

ソースファイル det.c

EIGEN

機能) 2**次元データを行列とみなし**, **固有値および固有ベクトルを数値的に求める**.

形式) **eigen(x, vec, val)**

パラメータ)
1. x : **入力オブジェクト** (Series, Snapshot)
2. vec : **出力オブジェクト：固有ベクトル** (Series)
3. val : **出力オブジェクト：固有値** (Series)

解説)
1. **実対称行列のみ計算可能である**.
2. **固有値はハウスホルダー法による直交変換により求めている**.
3. **あらかじめ**, vec **および** val **は** Series **変数として宣言しておく必要がある**.

使用例) **行列** $\begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$ **の固有値**, **固有ベクトルを求める**.

```
% matrix                          #  入力
[0][0]%            2          -1
[1][0]%           -1           2

% series vec,val                  #  出力を格納する変数宣言

% eigen(matrix,vec,val);
0

% val                             #  固有値
[0]:%            3           1

% vec                             #  固有ベクトル
[0]:[0]%      -0.7071     -0.7071

[1]:[0]%       0.7071     -0.7071
```

ソースファイル eigen.c

FFTC

機能) **複素数入力データの離散フーリエ変換値, または逆変換値を計算する**.

形式) **fftc(flg, x, y, u, v)**

パラメータ) 1. flg : **計算方式フラグ** (P or I)

P **正フーリェ変換**

I **逆フーリェ変換**

2. x : **入力時系列**:**実部** (Series , Snapshot)
3. y : **入力時系列**:**虚部** (Series , Snapshot)
4. u : **出力時系列**:**実部** (Series)
5. v : **出力時系列**:**虚部** (Series)

解説) 1. **入力** x,y **の長さは** 2 **の巾乗にしなければならない**.

2. **出力** u,v **は予め宣言しておく必要がある**.

使用例) ISPP User's Manual 3.1 **節 フーリエ変換 参照**.

参照) SPCF,FFTN

ソースファイル fftc.c

FFTN

機能) **多次元複素数入力データの離散フーリエ変換値**, **または逆変換値を計算する**.

形式) **fftn(flg, x, y, u, v)**

パラメータ) 1. flg : **計算方式フラグ** (P or I)

P **正フーリエ変換**

I **逆フーリエ変換**

2. x : **入力オブジェクト**:**実数部** (Series, Snapshot)
3. y : **入力オブジェクト**:**虚数部** (Series, Snapshot)
4. u : **出力オブジェクト**:**実数部** (Series, Snapshot)
5. v : **出力オブジェクト**:**虚数部** (Series, Snapshot)

解説) u, v **は**, **予め宣言しておく必要がある**.

使用例) ISPP User's Manual 3.3 **節** 2 **次元** series **の扱い方 参照**.

参照) FFTC

ソースファイル fftn.c

FIR

機能) **データに対して非再帰型**(FIR)**フィルタをかける**.

形式) **y = fir(prm, x[,data])**

パラメータ)

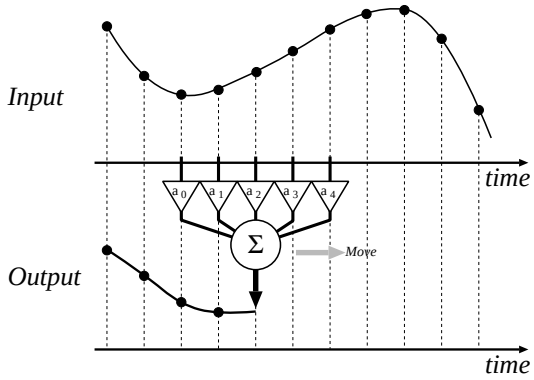
1. prm : **パラメータ系列** (Series, Snapshot)
2. x : **入力オブジェクト** (Series, Snapshot)
3. data : 2**次元**FIR**フィルタをかける場合に入力データからはみ出した部分を補完する**. **入力の平均を用いる場合は設定しなくてよい**. (Scalar)
4. y : **出力オブジェクト** (Series)

解説)

1. **入力が**2**次元のときは**, 2**次元**FIR**フィルタをかける**.
2. **フィルタリング・スパン** (**パラメータ系列の長さ**) **は**, **奇数値でなければならない**.
3. 1**次元フィルタの係数は** firmake **コマンドを用いて生成できる**.
4. 1**次元**FIR **フィルタの処理概念図を以下に示す**.

使用例) ISPP User's Manual 3.2 **節 フィルタ処理**, 3.3 **節** 2**次元**series**の扱い方 参照**.

参照) FIRMAKE, IIR



fir コマンドのフィルタ処理概念図

$a_0, a_1, \cdots, a_4$**は**, **コマンドにおけるパラメータ系列** (**添字は**, **系列の順番を示している**).

ソースファイル fir.c

FIRMAKE

機能) **非再帰型** (FIR) **フィルタを設計する**

形式) **prm = firmake(Type,Span,Cutoff,Window[,coef])**

パラメータ)

1. Type : **フィルタ特性** (1: LowPass, 2: HighPass, 3: BandPass)
2. Span : **フィルタ次数** (**奇数**)
3. Cutoff : **遮断周波数**
   LowPass, HighPass **では** Scalar **値** (**周波数** Hz)
   BandPass **特性では**, Series **で設定する**, **例えば**,
   Para = firmake(3,11,(100,200),3) のように設定する.
4. Window : **設計に使用する窓関数** (0: **矩形窓**, 1: **ハニング窓**, 2: **ハミング窓**, 3: **ブラックマン窓**, 4: **カイザー窓**)
5. coef : **カイザー窓を使用する場合は**, **カイザー窓の係数を与える** (coef $>$ 21). **他の窓では設定しなくてよい**.
6. prm : **パラメータ系列** (Series)

解説)

1. **フィルタ設計には**, **窓関数法を用いている**.
2. **２次元フィルタは設計できない**.
3. **フィルタリング・スパン** (**パラメータ系列の長さ**) **は**, **奇数値でなければならない**.

使用例) ISPP User's Manual 3.2 **節 フィルタ処理 参照**.

参照) FIR, IIR

ソースファイル firmake.c

GAUSS2

機能) 2**次元ガウス関数を生成する**.

形式) **x = gauss2(size, center, r11, r12, r22, max)**

パラメータ)

1. x : **出力オブジェクト** (Series)
2. size : **データサイズ** (**インデックス**)
3. center : **ガウス関数の中心座標** (**インデックス**)
4. r11,r12,r22 : **共分散行列の要素** (Scalar)
5. max : **ガウス関数の最大値** (Scalar)

ソースファイル gauss2.c

HIL

機能) **ヒルベルト変換処理を行う**.

形式) **hil(x, u, v)**

パラメータ)
1. x : **入力時系列** (Series , Snapshot)
2. u : **出力時系列**:**実部** (Series)
3. v : **出力時系列**:**虚部** (Series)

解説) **出力** u,v **は予め宣言しておく必要がある**.

ソースファイル hil.c

HPBTW

機能) **バタワース特性の**IIR**型ハイパスフィルタを設計する**.

形式) **gain = hpbtw(fc, order, zr, zi, pr, pi)**

パラメータ)

1. fc : **遮断周波数** (**入力**) [Hz] (Scalar)
2. order : **フィルタ次数** (**入力**) (Scalar)
3. zr : **零点**(**実数部**) **出力** (Series)
4. zi : **零点**(**虚数部**) **出力** (Series)
5. pr : **極**(**実数部**) **出力** (Series)
6. pi : **極**(**虚数部**) **出力** (Series)
7. gain : **ゲイン** (Scalar)

解説)

1. **設計値は**, iircoef **を使用すれば直接型**IIR **フィルタの係数に変換できる**.
2. **設計可能フィルタ次数は最大** 101 **次である**.

使用例) ISPP User's Manual 3.2 **節 フィルタ処理 参照**.

参照) LPBTW, BPBTW, IIRCOEF

ソースファイル hpbtw.c

ICEP

機能) 2 **つのオブジェクトのデータの逆ケプストラムを計算する**.

形式) **icep(x, y, u, v)**

パラメータ)
1. x : **入力時系列**:**実部** (Series, Snapshot)
2. y : **入力時系列**:**虚部** (Series, Snapshot)
3. u : **出力時系列**:**実部** (Series)
4. v : **出力時系列**:**虚部** (Series)

解説)
1. y **に** 0 **を入力した場合は**, **虚部のデータに全て** 0 **が入力されたとして処理する**.
2. **出力** u,v **は予め宣言しておく必要がある**.

参照) CEP

ソースファイル icep.c

IIR

機能) **データに対して再帰型** (IIR) **フィルタをかける**.

形式) **y = iir(prm, x)**

パラメータ)
1. prm : **パラメータ系列** (Series , Snapshot)
2. x : **入力時系列** (Series , Snapshot)
3. y : **出力時系列** (Series)

解説)
1. **入力時系列は** 1 **次元でなければならない**.
2. **アルゴリズム**

   入力データを n **個の離散値** $x(i)$(**ただし**, $i = 1, 2, \cdots, n$) **で表し**, N **個の離散点からなる "重み関数"** $w(j)$(**ただし**, $j = 1, \cdots, N$) **を用いる**. **出力値** $y(i)$ **は次のように求める**.

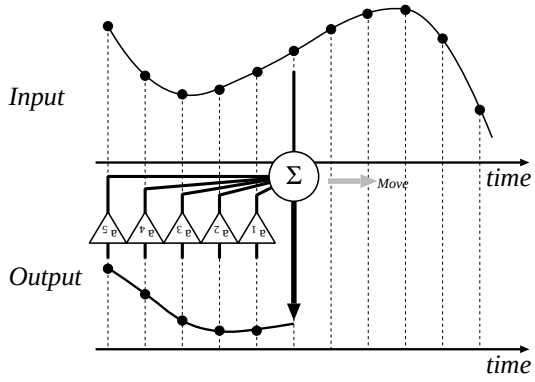
$$y(i) = \sum_{j=1}^{N} w(j)y(i-j) + x(i)$$

   iir **コマンドの処理概念図を以下に示す**.
3. **通常の方法で設計される直接型の** IIR **フィルタによる処理を行なうには**, **次のようにする**. par_a, par_b **はそれぞれ伝達関数** $H(z)$ **の分母**, **分子の各係数を** fir, iir **コマンドに直接入力できるように** iircoef **コマンドで調整したものである**.

```
% temp = fir( par_b , input_signal );
% output_signal = gain * iir( par_a , temp );
```

使用例) ISPP User's Manual 3.2 **節 フィルタ処理**

参照) FIR, IIRCOEF, BPBTW, HPBTW, LPBTW



iir コマンドのフィルタ処理概念図

$a_1, \cdots, a_5$ **は**, **コマンドにおけるパラメータ系列** (**添字は系列の順番を示す**).

ソースファイル iir.c

IIRCOEF

機能) **零点**, **および極から直接型**IIR **フィルタの係数を求める**.

形式) **iircoef(zr, zi, pr, pi, a, b)**

パラメータ)
1. zr : **零点の実数部 入力オブジェクト** (Series, Snapshot)
2. zi : **零点の虚数部 入力オブジェクト** (Series, Snapshot)
3. pr : **極の実数部 入力オブジェクト** (Series, Snapshot)
4. pi : **極の虚数部 入力オブジェクト** (Series, Snapshot)
5. a : **伝達関数の分母の係数の出力オブジェクト** (Series)
6. b : **伝達関数の分子の係数の出力オブジェクト** (Series)

解説)
1. **出力される係数は**, iir, fir **それぞれのコマンドにそのまま使用できるフォーマットで出力される**.
2. **出力** a,b **は予め宣言しておく必要がある**.

使用例) ISPP User's Manual 3.2 **節 フィルタ処理**

参照) BPBTW, HPBTW, LPBTW, FIR, IIR

ソースファイル iircoef.c, Complex.h, complex.c

INTEG

機能) Series**の内容を積算する**.

形式) **n = integ(x [,y])**

パラメータ)
1. x : **入力オブジェクト** (Series, Snapshot)
2. y : **出力オブジェクト** (Series)
3. n : **積算値** (Scalar)

解説)
1. **出力** y **には**, **積算の途中経過が格納される**.
   (**数学の積分とは異なる**. sin(t) → –a × cos(t))
2. **返り値** n **は**, **全要素についての合計**
3. **処理** $n = \sum_{j=1}^{m} x(j)$

使用例)

```
% x = (-2,-1,0,1,2);          #  入力データ
% series y;                   #  途中結果を格納する変数

% integ(x,y);
0                             #  計算結果

% y
[0]:%  -2  -3  -3  -2  0  # 途中経過
```

ソースファイル integ.c

INTERP

機能) 2**次元データを読み込み**2**次元補間を行ない**, **その結果を出力する**.

形式) **y = interp(x, s1, s2)**

パラメータ)
1. x : 2**次元入力オブジェクト** (Series, Snapshot)
2. y : 2**次元出力オブジェクト** (Series)
3. s1 : X**軸データの補間数** ($\geq$0)
4. s2 : Y**軸データの補間数** ($\geq$0)

解説)
1. **補間数は**, **元データの**2**点のデータ間を**, **何個のデータをもって補間するのかを指定する**.
2. **入力オブジェクトは**, 2**次元データでなければならない**.
3. X**軸とは**, **時間軸** (**データ長が可変のデータ方向**)

参照) CONT(NEWGPM)

ソースファイル interp.c

INV

機能) 2 **次元データを行列とみなし，逆行列を求める**.

形式) y = inv(x)

パラメータ) 1. x : **入力オブジェクト** (Series, Snapshot)

2. y : **出力オブジェクト** (Series, Snapshot)

解説) 1. **正方行列のみ計算可能**

2. **３次元以上のデータは**, **下位の２次元データを配列とみなし**, **すべての配列について逆行列を計算し**, **同じインデックスサイズの** Series **を返す**.

使用例) 3 **次元データの場合**

```
% matrix                                          #  入力データ
[0]:[0][0]%             1             0
[0]:[1][0]%             0             1

[1]:[0][0]%             0             1
[1]:[1][0]%             1             0

[2]:[0][0]%             1             2
[2]:[1][0]%            -2             1

% inv(matrix);                                    #  逆行列の計算
[0]:[0][0]%             1             0
[0]:[1][0]%             0             1

[1]:[0][0]%             0             1
[1]:[1][0]%             1             0

[2]:[0][0]%           0.2          -0.4
[2]:[1][0]%           0.4           0.2
```

ISPP User's Manual 3.4 **節 行列演算 参照**.

参照) DET

ソースファイル inv.c

LEVIN

機能) **時系列に線形** AR **モデルをあてはめ**, Levinson–Durbin **アルゴリズムを用いて** AR **係数を求め**, **パワースペクトルを出力する**

形式) **levin(x, pw, aic, method, odr, dpt, tola, err, coef)**

パラメータ)

1. x : **入力時系列** (Series, Snapshot)
2. pw : **出力時系列**:**パワースペクトル** (Series)
3. aic : **出力時系列**:AIC (Series)
4. method : **次数決定方式**(A or other)

   A AIC_{min} **で次数決定**

   数値 Fixed Order
5. odr : AR **モデルの最大次数**
6. dpt : **パワースペクトルの出力データ点数**
7. tola : AIC **に対する許容値**(BURG **コマンド参照**)
8. err : **出力時系列**:**予測誤差** (Series)
9. coef : **出力時系列**:AR **係数** (Series)

解説)

1. **入力データは** "DCCUT" **コマンドにより**, **直流分をカットしておく**
2. "POLE" **等により**, **極の特性を知ることができる**
3. **出力** pw,aic,err,coef **は予め宣言しておく必要がある**.

使用例)

$$x_k + 0.8x_{k-1} = e_k, \quad e_k \sim N(0.0, 1.0)$$

に従う信号のパワースペクトルと AR **係数を求める**(odr:**固定**).

```
% x = argen(para,1024);                  #  信号の生成.

% series pw,aic,err,coef;                #  Series 変数の宣言.
% levin(x,pw,aic,1,1,128,2,err,coef); #  推定.
Order =    1     AIC =        -41.225
 AR Coefficient(  1)=      0.7817198,      0.7817198

% coef
[0]:%       0.7917
```

参照) BURG, POLE

ソースファイル levin.c

LPBTW

機能) **バタワース特性の**IIR**型ローパスフィルタを設計する**.

形式) **gain = lpbtw(fc, order, zr, zi, pr, pi)**

パラメータ)

1. fc : **遮断周波数** (**入力**) [Hz] (Scalar)
2. order : **フィルタ次数** (**入力**) (Scalar)
3. zr : **零点**(**実数部**) **出力** (Series)
4. zi : **零点**(**虚数部**) **出力** (Series)
5. pr : **極**(**実数部**) **出力** (Series)
6. pi : **極**(**虚数部**) **出力** (Series)
7. gain : **ゲイン** (Scalar)

解説)

1. **設計値は**, iircoef **を使用すれば直接型**IIR **フィルタの係数に変換できる**.
2. **設計可能フィルタ次数は最大** 101 **次である**.

使用例) ISPP User's Manual 3.2 **節 フィルタ処理 参照**.

参照) HPBTW, BPBTW, IIRCOEF

ソースファイル lpbtw.c

MNRAND

機能) **多次元正規分布に従う乱数ベクトルを発生する**.

形式) **x = mnrand(dpt, seed, mvec, vmat)**

パラメータ)

1. x : **出力オブジェクト** (Series)
2. dpt : **出力ベクトルの数**
3. seed : **乱数の初期値** (**奇数**)
4. mvec : **入力オブジェクト：平均値ベクトル** (Series, Snapshot)
5. vmat : **入力オブジェクト：分散共分散行列** (Series, Snapshot)

解説)

1. **乱数のシードは**, M **系列を用いて生成している**.
2. **分散共分散行列に対してコレスキー分解を施すことにより**, **多次元乱数を生成している**.
3. **分散共分散行列は**, **正定値対称行列でなければならない**.

使用例) **平均** $\mu_1 = \mu_2 = 0$, **分散** $\sigma_1^2 = \sigma_2^2 = 1$, **相関係数** $\rho_{12} = 0.5$ **で規定される** 2 **次元正規分布に従う確率変数ベクトル** (y_1, y_2) **を** 1000 **点生成する**.

```
mvec = (0,0);                  # 平均ベクトルの設定
vmat = (1,0.5,0.5,1);          # 分散共分散
vmat = reform(vmat,(2,2));     #      行列の設定
y = mnrand(1000,1,mvec,vmat); # 乱数生成
y1 = y[0];                     # 1000 点の乱数が
y2 = y[1];                     #          格納される
```

参照) NRAND, URAND

ソースファイル mnrand.c

MUL

機能) 2つの2次元データを行列とみなして積を求める.

形式)

```
z = mul( x, y )
```

パラメータ) 1. x, y : **入力オブジェクト** (Series, Snapshot)

2. z : **出力オブジェクト** (Series, Snapshot)

解説) 1. **1次元データは**, **行ベクトルとみなして計算する**.

2. **3次元以上のデータについての動作は**, **次のようになる**.

- x: **2次元以下**,y: **3次元**—x **に** y **の行列をそれぞれ乗ずる**.
- x: **3次元**,y: **2次元以下**—x **の行列のそれぞれに** y **を乗ずる**.
- x: **3次元**,y: **3次元**—x **と** y **の行列が一対一に対応すればそれぞれ積を求める**, **それ以外はエラー**.

使用例) ISPP User's Manual 3.4 **節 行列演算 参照**.

ソースファイル mul.c

NMEQ

機能) **正規方程式** $\mathbf{X}^T\mathbf{X}\beta = \mathbf{X}^T\mathbf{y}$ **を解き**, **パラメータの最小**2**乗推定量** $\widehat{\beta}$ **を求める**.

形式) **para = nmeq(x, y)**

パラメータ)
1. para : **出力オブジェクト ：パラメータベクトル** (Series)
2. x : **入力オブジェクト：計画行列** (Series, Snapshot)
3. y : **入力オブジェクト：従属変数ベクトル** (Series, Snapshot)

解説)
1. **ハウスホルダー変換を用いて計算している**.

ISPP User's Manual 3.4 **節 行列演算 では行列演算コマンドを組合わせて同様の機能を実現しているが**, **精度等については本コマンドの方が優れている** (**ハウスホルダー変換を用いているため**).

使用例) **あるシステムへの入力データ** $\mathrm{x} = (-2, -1, 0, 1, 2)$ **と**, **出力データ** $\mathrm{y} = (7.25, 3.81, 1.88, 2.33, 5.12)$ **が与えられている時**, **これを**, **以下のようなモデル**,

$$y_i = \beta_1 + \beta_2 x_i + \beta_3 x_i^2 + \varepsilon_i, \quad i = 1, \cdots, 5$$

により同定する.

```
% x = (-2,-1,1e-10,1,2);
% y = (7.25,3.81,1.88,2.33,5.12); # 従属変数ベクトルの設定

% series xmat[3];
% xmat[0] = x^0;                  # 計画行列の設定
% xmat[1] = x^1;
% xmat[2] = x^2;

% beta = nmeq(xmat,y);            # パラメータ値を求める
```

パラメータの最小2**乗推定量は次のようになる**.

```
% beta
[0]:%        1.958       -0.574         1.06
```

ソースファイル nmeq.c

NORM

機能) **指定されたオブジェクトの内容を正規化する**.

形式) **y = norm(x , type)**

パラメータ)

1. x : **入力オブジェクト** (Series, Snapshot)
2. y : **出力オブジェクト** (Series, Snapshot)
3. type : **正規化方法**
   - **1** — **最大値で正規化**
   - **2** — **最小値で正規化**
   - **3** — **絶対値最大で正規化**
   - **4** — **絶対値最小で正規化**
   - **5** — **平均**0, **分散**1 **に正規化**

解説) **処理内容を以下に示す**.

$$
\begin{array}{lcl}
\text{type=1} & : & y_i = x_i / x_{max} \\
\text{type=2} & : & y_i = x_i / x_{min} \\
 & & z_i = |x_i| \\
\text{type=3} & : & y_i = x_i / z_{max} \\
\text{type=4} & : & y_i = x_i / z_{min} \\
\text{type=5} & : & y_i = (x_i - x_{mean}) / \sqrt{1/n \sum_j (x_j - x_{mean})^2}
\end{array}
$$

使用例) **データの最大値を**1, **最小値を**0 **に正規化する場合の記述例**

```
% y = norm(x-min(x), 1);
```

ソースファイル norm.c

NRAND

機能) **正規乱数デ- タを生成する**.

形式) **x = nrand(dpt, init, mean, variance)**

パラメータ)
1. x : **出力時系列** (Series)
2. dpt : **出力デ- タ点数**
3. init : **乱数の初期値** (**奇数**)
4. mean : **平均値**
5. variance : **分散**

解説)
1. **乱数のシ- ドは**, M **系列を用いて生成している**

使用例) **平均** 0.5 **分散** 1.0 **の正規分布に従う乱数を** 1024 **点生成する**.

```
% x = nrand(1024, 1, 0.5, 1.0);
```

ISPP User's Manual 3.1 **節 フーリエ変換 参照**.

参照) MNRAND, URAND

ソースファイル nrand.c

PHASE

機能) 2**つのオブジェクトの内容により位相を求める**.

形式) **ph = phase(x, y, d, u)**

パラメータ)

1. x : **入力オブジェクト**:**実部** (Series, Snapshot)
2. y : **入力オブジェクト**:**虚部** (Series, Snapshot)
3. ph : **出力オブジェクト** (Series)
4. d : **出力データ形式** (D or O)
   - D degree
   - O radian
5. u : **位相戻し操作フラグ** (U or O)
   - U **位相戻しする**
   - O **位相戻ししない**

解説)

1. **連続するデータが** π **以上変化したとき位相戻し操作が行われる**
2. **実部および虚部がともに 0 となるデータを含む場合には**, **位相戻しが正常に動作しない場合がある**.

使用例) ISPP User's Manual 3.1 **節 フーリエ変換 参照**.

参照) FFTC, POWER, SPCF

ソースファイル phase.c

POLE

機能) BURG, LEVIN **コマンドで求められた** AR **係数から極の位置を求める**.

形式) **pole(ar, xr, xi)**

パラメータ) 1. ar : AR**係数を格納してある時系列** (Series, Snapshot)

2. xr : **極の実数部の値を格納する時系列** (Series, Snapshot)

3. xi : **極の虚数部の値を格納する時系列** (Series, Snapshot)

使用例) **線形** AR **モデル**

$$x_k + 0.8x_{k-1} = e_k$$

の極の位置を求める.

```
% para                                          # AR 係数
[0]:%   0.8
% series xr,xi;                                 # 出力を格納する変数の宣言

% pole(para,xr,xi);
 AR[  0]=            0.8

 I= 0  X=           -0.8  Y=              0
 I= 1  X=    3.3952e-313  Y=    3.3952e-313
 I= 2  X=              0  Y=            0.8
 I= 3  X=    3.3952e-313  Y=    3.3952e-313

% xr                                            # 実部
[0]:%         -0.8
% xi                                            # 虚部
[0]:%            0
```

参照) BURG, LEVIN

ソースファイル pole.c

POWER

機能) 2**つのオブジェクトの内容を**2**乗して加算する**.

形式) **z = power(x, y)**

パラメータ) 1. x, y : **入力オブジェクト** (Series, Snapshot)

2. z : **出力オブジェクト** (Series)

解説) **処理** : $z_i = x_i^2 + y_i^2$

使用例) ISPP User's Manual 3.1 **節 フーリエ変換 参照**.

参照) FFTC, PHASE, SPCF

ソースファイル power.c

RANK

機能) **データの内容から**, **度数**, **分散**, **平均値を求め**, **その正規分布を算出する**.

形式) **rank(x, u, v, w, xmin, xmax, n)**

パラメータ)

1. x : **入力時系列** (Series , Snapshot)
2. u : **出力時系列**:**度数-** Y**軸** (Series, Snapshot)
3. v : **出力時系列**:**度数-** X**軸** (Series, Snapshot)
4. w : **出力時系列**:**正規分布-** Y**軸** (Series, Snapshot)
5. xmin : **階級の最小値** (Scalar)
6. xmax : **階級の最大値** (Scalar)
7. n : **階級の個数**

使用例) ISPP User's Manual 3.5 **節 データの統計処理 参照**.

ソースファイル rank.c

SHIFT

機能) **データ上の任意の点を指定したレベルと一致するように全体をシフトする**.

形式) y = shift(x, pos, level)

パラメータ)
1. x : **入力オブジェクト** (Series, Snapshot)
2. y : **出力オブジェクト** (Series, Snapshot)
3. pos : **データインデックス**
4. level : **指定レベル** (Scalar)

ソースファイル shift.c

SPCF

機能) **入力データのパワースペクトル, 位相を計算する**.

形式) **spcf(x, amp, phs)**

パラメータ)
1. x : **入力時系列** (Series, Snapshot)
2. amp : **出力時系列:パワースペクトル** (Series, Snapshot)
3. phs : **出力時系列:位相** (Series, Snapshot)

解説)
1. N **点のデータを入力したときの出力データ**

   データ点数 : $N/2+1$
   パワースペクトル : $amp_i = \frac{2}{N \cdot sf}\{X^2_{\rm real}(i) + X^2_{\rm imag}(i)\}$
   Nyquist **周波数までの片側** PSD **を出力するので** 2 **倍してパワーが同じになるようにしてある**.
   位相 : **単位は** degree **である**.
   sf **は**, **サンプリング周波数**.
2. **出力** amp, phs **は予め宣言しておく必要がある**.

離散データからの FFT **によるパワースペクトル推定について簡単に触れておく**. **入力** $x_t, t = 0, 1, \cdots, N-1$ **のパワースペクトルの計算には一般的に次式が用いられている** [1].

$$\begin{aligned} P(f) &= \frac{1}{T}\|X(f)\|^2 = \frac{sf}{N}\|X(f)\|^2 \\ X(f) &= \frac{1}{sf}\sum_{t=0}^{N-1} x_t e^{-j\cdot 2\pi f t/sf} \end{aligned}$$

ただし本コマンドは FFTC コマンドと同一のモジュールを使用しているため以下のようになる.

$$\begin{aligned} P(f) &= \frac{sf}{N}\|X(f)\|^2 \\ &= \frac{1}{N\cdot sf}\{X^2_{\rm real}(f) + X^2_{\rm imag}(f)\} \\ X(f)\cdot sf &= X_{\rm real}(f) - jX_{\rm imag}(f) \end{aligned}$$

上式以外の定義を用いる文献 [2] **があるので注意が必要である**.

使用例) ISPP User's Manual 3.1 **節 フーリエ変換 参照**.

参照) FFTC, PHASE, POWER

[1] **中溝 高好**: "**信号解析とシステム同定**", **コロナ社**, 1988
[2] **越川 常治**: "**信号解析入門**", **近代科学社**, 1992

ソースファイル spectr.c

SPLINE

機能) **自然**3**次スプライン補間により，**1**次元系列の入出力データ対**x, y **を補間した系列** xip, yip **を求める．**

形式1) **yip = spline (x, y, xip)**

形式2) **yip = spline (x, y, dpt [, xip2])**

パラメータ)

1. x : **入力時系列** (Series, Snapshot)
2. y : **出力系列** (1 **次元** Series)
3. dpt : **補間後のデータ点数** (>0), **サンプリング周波数からデータ点数**, **補間後の入力系列を求める** (=0)**．**
4. xip : **補間後の入力系列** (1 **次元** Series)
5. yip : **補間後の出力系列** (1 **次元** Series)
6. xip2 : **動生成された補間後の入力系列** (1 **次元** Series)

解説)

1. x, y **は同じデータ点数でなければならない．**
2. x **および** xip **は，ともに** $x_i < x_{i+1}$ **の関係を満たしていなければならない．**

形式1 **について**:

- xip **の系列と対になる** yip **が求められる．**
- xip **の値の範囲は** x **の値の範囲を超えてはならない．**

形式2 **について**:

- dpt = 0 **の場合は，サンプリング周波数をもとに**, **以下の定義により，**xip **が自動生成される．**

$$xip_0 = x_0, xip_i = x_0 + step \times i, ...xip_{dpt-1} = x_{n-1}$$

ここで,$step = 1.0/sf, dpt = (x_{n-1} - x_0) \times sf + 1,$

ただし,sf **はサンプリング周波数,**n **は**x **のデータ点数．**

- dpt > 0 **の場合は，上式の** step **を次式として** xip **が生成される．**

$dpt > 1$ **のとき** $step = (x_{n-1} - x_0)/(dpt - 1)$

$dpt = 1$ **のとき** $step$ **は** 0

- **第**4**引数** xip2 **には，自動生成された補間後の入力系列** xip **が格納される．**

参照) AKIMA

ソースファイル spline.c

TRANS

機能) 2**次元データを行列とみなし転置行列を求める**.

形式) **y = trans(x)**

パラメータ) 1. x : **入力オブジェクト** (Series, Snapshot)

2. y : **出力オブジェクト** (Series, Snapshot)

解説) 1. **１次元データは**, **行ベクトルとみなして処理する**.

2. **３次元以上のデータについては**, **複数の行列とみなし**, **下位の２次元についてを行列として処理する**.

使用例)

```
% x = (1,2,3);          #  入力データ（行ベクトル）

% trans(x);             #  転置（列ベクトル）
[0]:[0]%   1
[1]:[0]%   2
[2]:[0]%   3
```

ISPP User's Manual 3.4 **節 行列演算 参照**.

ソースファイル trans.c

URAND

機能) **一様乱数デ- タを生成する**.

形式) **x = urand(dpt, init, lower, upper)**

パラメータ)
1. x : **出力時系列** (Series, Snapshot)
2. dpt : **出力デ- タポイント**
3. init : **初期値** (**奇数**)
4. lower : **デ- タの最小値** (Scalar)
5. upper : **デ- タの最大値** (Scalar)

解説)
1. **乱数のシ- ドは**, M **系列を用いて生成している**.

使用例) **区間** $[0, 1]$ **の一様乱数を** 1024 **点生成する**.

```
% x = urand(1024, 1, 0, 1);
```

参照) MNRAND, NRAND

ソースファイル urand.c

WINDOW

機能) **データに対して, ウィンドウ処理(ハニング, ハミング)を施す**.

形式) **y = window(x, type, flag)**

パラメータ)

1. x : **入力オブジェクト** (Series, Snapshot)
2. y : **出力オブジェクト** (Series, Snapshot)
3. type : **ウィンドウの種類**(1∼4)
   - 1 — **ハミングウィンドウ**
   - 2 — **ハニングウィンドウ**
   - 3 — **ブラックマン**
   - 4 — **トライアングル**
4. flag : **データ補正フラグ**(0 or 1)
   - 0 — **補正しない**
   - 1 — **補正する**

解説)

1. **データ補正を行なうとウィンドウを処理前後の積分値が等しくなる**.
2. **入力が**2**次元データの場合には**, 2**次元ウィンドウ処理を行なう**.

使用例) ISPP User's Manual 3.1 **節 フーリエ変換 参照**.

ソースファイル window.c

グラフィックシステム

GPM

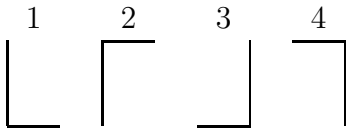
AXIS

機能) “cont”, “graph”, “map” **コマンドで表示したグラフの**X**軸**, Y**軸を**Linear**もしくは**Log**表示する**.

形式) **axis(tbuf, typ, flg1, flg2 , size, grid, form, xscl, yscl, index)**

パラメータ)

1. tbuf : **タイトルバッファ番号**(1, 2, 3)
   グラフィックコマンドでは, **軸のタイトルを登録するために**, x, y, z**軸** 1**組で**, 3**本のタイトルバッファを持っている**. **その中の何番のタイトルを表示するかの指定を行う**. **なお**, **本コマンドでは**z**軸の値は使用しない**.
2. typ : **軸の表示形式**(1, 2, 3 or 4)



3. flg1 : **表示する軸の指定**(“X”, “Y” or “XY”)
   “X” X**軸のみ表示**
   “Y” Y**軸のみ表示**
   “XY” XY**両軸を表示**
4. flg2 : **表示するスケールの指定**(“X”, “Y” or “XY”)
   “X” X**軸のスケールのみ表示**
   “Y” Y**軸のスケールのみ表示**
   “XY” XY**両軸のスケールを表示**
5. size : **表示するスケール**, **タイトルの文字サイズ**[mm]
6. grid : **グリッド表示の指定**
   0 **グリッドなし**
   1 X**軸のみを表示**
   2 Y**軸のみを表示**
   3 XY**両軸を表示**
7. form : **スケール表示の型指定**
   0 X**軸**···**実数**, Y**軸**···**実数**
   1 X**軸**···**整数**, Y**軸**···**実数**
   2 X**軸**···**実数**, Y**軸**···**整数**

3　X**軸**···**整数**, Y**軸**···**整数**

8. xscl : X**軸スケール表示間隔**
9. yscl : Y**軸スケール表示間隔**
10. index : linear **スケール値の指数表示指定** (0 or 1)

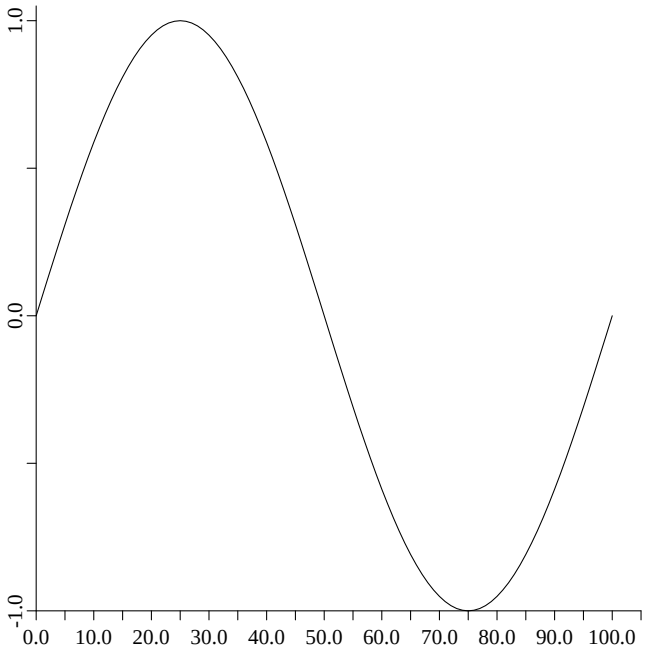
0　**必要ならば指数表示をする** (**デフォルト**)

1　**スケール値に関係なく指数表示を行なわない**

使用例)

```
t = (0~100)/100*2*PI
data = sin(t)
wopen(1,"A4",0,0)
axis(1,1,"XY","XY",3.5,0,0,10,1,0)
```

と入力すると,



となる.

ソースファイル　axis.c

CHWIN

機能) **アクティブウィンドウを切り替える**.

形式) **chwin(wnum)**

パラメータ) 1. wnum : **切り替えたいウィンドウ番号**

ソースファイル chwin.c

COLOR

機能) **グラフ**, **軸**, **フレームの色を指定する**.

形式) **color(pen1, pen2)**

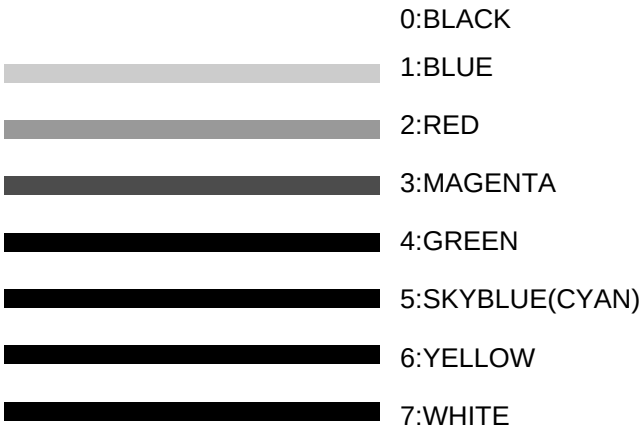
パラメータ) 1. pen1 : **グラフの色** (1≤pen1≤10, **整数** or **カラー名**)

2. pen2 : **軸**, **フレームの色** (1≤pen2≤10, **整数** or **カラー名**)

解説) 1. **色指定** (**ディスプレイ**)
色指定は, **以下の整数値以外にカラー名を指定してもよい**. (**小文字でも可**)
カラー名： "BLACK", "BLUE", "RED", "MAGENTA", "GREEN", "SKY-BLUE"("CYAN"), "YELLOW", "WHITE", "GLAY1", "GLAY2", "GLAY3"

0	**黒**	1	**青**
2	**赤**	3	**紫**
4	**緑**	5	**水色**
6	**黄色**	7	**白**
8	**グレー**1	9	**グレー**2
10	**グレー**3		

使用例) **プリンタに出力したときは次のようになる**. **ディスプレイと紙では白黒が反転するのに注意**. (GRAY1 **から** GRAY3 **のプリンタへの出力は，それぞれ** 3, 2, 1 **のカラー番号を指定したときと同じ濃度で出力される**)



ソースファイル newpen.c

CONT

機能) **入力された**2**次元データの等高線を表示する**.

形式) **cont(x, step, dirc, view, mode)**

パラメータ)

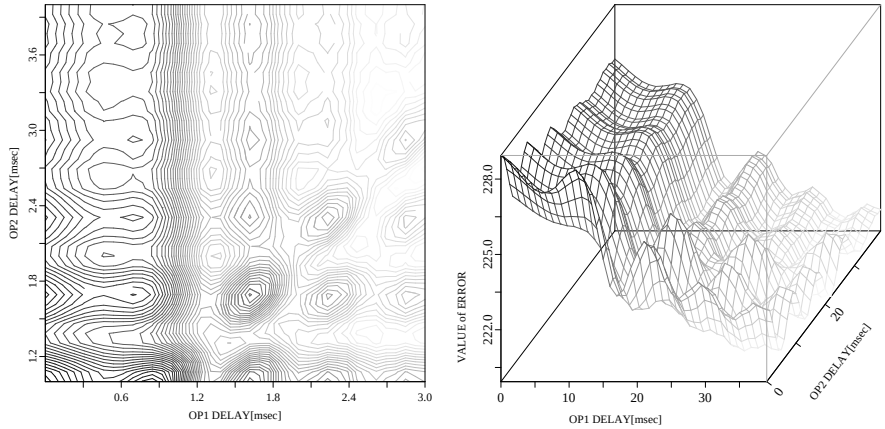
1. x : **入力オブジェクト** (Series, Snapshot)
2. step : **等高線の表示間隔**
3. dirc : **表示方向**
   - "X" **オブジェクトの上位次元が**X**軸方向になる**
   - "Y" **オブジェクトの下位次元が**Y**軸方向になる**
4. view : **表示データの原点の指定** (1 : **左下が原点**, -1 : **左上が原点**)
5. mode : **表示モードの指定** (0 : **モノクロ**, 1 : **カラー**)

解説)

1. **表示する等高線の最大**, **最小値の設定は**, "scale" **コマンドで設定する**.
2. **表示を行なう際**, **補間は行なわない**. **等高線を滑らかに表示するためには**interp **コマンドを用いる必要がある**.

参照) INTERP

使用例)



ソースファイル cont.c

DRAW

機能) “cont”, “graph”, “map” **コマンドで表示したグラフに対し**, **指定したレベルのラインを表示する**

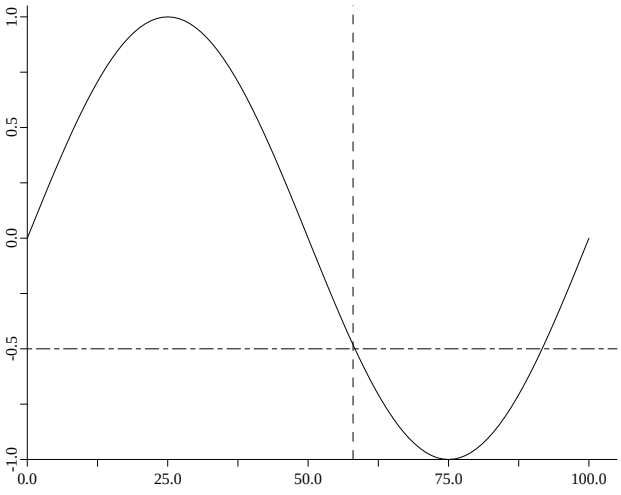
形式) **draw(axs, val)**

パラメータ)
1. axs : **軸の指定**(“X” or “Y”)
2. val : **ライン表示する値** (Scalar)

解説)
1. **第１引数が** “X” **ならば**, x = val **となるような**, “Y” **ならば**, y = val **となるような直線を作画する**.
2. **色の指定は** “color” **コマンドの第** 2 **パラメータで指定する**
3. graph **を表示しなくても**, scale **で設定したレベルに基づいてラインを引くこともできる**.

使用例)

```
   graph(data,"T",0,0,0,0,0)
ltype( 2, 2 )     # 長鎖線
draw( "X", 58 )   # X 軸に補助線を引く
ltype( 3, 3 )     # 一点鎖線
draw( "Y", -0.5 ) # Y 軸に補助線を引く
```



ソースファイル drawl.c

EGRAPH

機能) **エラーバー付きのグラフを表示する**.

形式) **egraph(y, x, axs, lne, step, sym1, sym2)**

パラメータ)

1. y : Y**軸方向のデータ** (Numeric, "T" or "F")

   object **入力オブジェクト** (Series, Snapshot)
   "T" **時間軸**
   "F" **周波数軸**
   "D" **データ点数**

2. x : X**軸方向のデータ** (Numeric, "T" or "F")

   object **入力オブジェクト** (Series, Snapshot)
   "T" **時間軸**
   "F" **周波数軸**
   "D" **データ点数**

3. axs :XY**軸の形式** (0, 1, 2 or 3)

	X**軸**	Y**軸**
0	LINEAR	LINEAR
1	LOG	LINEAR
2	LINEAR	LOG
3	LOG	LOG

4. lne : **グラフ描画の形式**

   0 **エラーバー付きのグラフを引く**
   1 **センターシンボルおよびエラーバーを表示**
   2 **エラーバーおよびグラフ, センターシンボルを表示**
   3 **棒グラフを描く**
   4 lne = 0 **および** lne = 3 **で表示する**
   5 **零レベルとグラフで囲まれた領域を塗りつぶす**
   6 **エラーバーのみ描く**

5. step : **表示間隔** (step≥0, **整数型**)
   (STEP–1) **点ごとのデータを対象に処理を行う**

6. sym1 : **センターシンボルの指定** (0≤sym1≤16, **整数型**) **シンボルの種類については**, graph **コマンド参照**

7. sym2 : **センターシンボルの大きさ** [mm](sym2≥0, **実数型**)

解説)

1. **データ形式は**, index **が** (n,2) **または**, (n,3) **である必要がある**. **例えば**, y **が表示するデータである場合**, **前者の場合には**, y[0] **がグラフ**, y[0]±y[1] **がエラーバーとして表示される**. **後者の場合**, y[0] **をグラフ**, y[1], y[2] **をエラーバーの端点として表示する**.
2. **時間軸表示**
$$buff(i) = \frac{1000.0 \times float(i-1)}{sam}$$
3. **周波数軸表示**
$$delta = \frac{sam}{float(n-1) \times 2.0}$$
$$buff(i) = delta \times float(i-1)$$
4. **センターシンボルの大きさ**

   大きさは [mm] **で入力する**. **但し**, **零を入力したときは** 0.5[mm] **に自動設定される**

ソースファイル egraph.c graph_sub.c graph_sub.h

FACTOR

機能) **グラフィック表示倍率を指定する**.

形式) factor(x)

パラメータ) 1. x : **表示倍率** (x>0)

注意) **デフォルト値は** 1.0.

ソースファイル factor.c

FONT

機能)　　**グラフィック端末の文字種類を指定する**.

形式)　　font(type)

パラメータ)　1. type : **文字種類番号** (**文字種類と番号の対応は以下のようになっている**)

	roman	italic	bold	bold-italic
times	FONT 1	*FONT 2*	**FONT 3**	***FONT 4***
helvetica	FONT 5	*FONT 6*	**FONT 7**	***FONT 8***
courier	FONT 9	*FONT 10*	**FONT 11**	***FONT 12***

ソースファイル　　font.c

FRAME

機能) **グラフの枠を描く**.

形式) frame()

パラメータ) **なし**

解説) 1. origin, size **などで指定されたパラメータに従って**, **グラフの枠を描く**.

ソースファイル frame.c

GINIT

機能) **グラフィックパラメータを初期化する**.

形式) ginit()

パラメータ) **なし**

解説) 1. **初期設定**

カラー (グラフ) 白
カラー (軸, フレーム) 白
スケール設定 自動設定
スケールファクター 1.0
原点 (X, Y) (20.0, 20.0)
サイズ (X, Y) (100.0, 100.0)

ソースファイル ginit.c

GPM2PS

機能) GPM **ファイル** (GPMDVIFILE) **をポストスクリプトファイルに変換する**

形式)

gpm2ps [**-rv**][**-cps**] [**dvifilename**]
gpm2eps [**-rv**][**-cps**] [**dvifilename**]

パラメータ)

1. -rv : map, gsolm, cont **の濃淡レベルを黒**:**低**–**白**:**高とする**. (**無指定時の濃淡レベルは**, **白**:**低**–**黒**:**高**)
2. -cps : **カラー** PS **を出力する**. (**無指定時は**, **モノクロ出力**)
3. dvifilename : GPM **ファイル名** (**省略時は**, GPMDVIFILE1 **が設定される**)

解説)

1. **ポストスクリプト出力は**, **標準出力に書き出されるので**, **リダイレクトによりファイルに格納するか**, **パイプによりプリンタに出力することができる**.
2. **本コマンドで作成したファイルは**, ps2epsi **コマンドを使用することにより** eps **ファイルに変換できる** (ps2epsi **は** ghostscript **に同梱されるツールである**).
3. gpm2eps **コマンドは**, gpm2ps **および** ps2epsi **を実行するものである**. **直接** eps **ファイルを作成できるので**, TEX **などに取り込む目的でファイルを作成する場合には**, gpm2eps **を用いるとよい**.
4. map **コマンドのカラー** PS **については，疑似カラーのみ対応で，グレイスケールには対応していない．**

ソースファイル gpm2ps.c gpm2eps.sh

GRAPH

機能) **データをグラフ表示する**.

形式) **graph(y, x, axs, lne, step, sym1, sym2)**

パラメータ) 1. y : Y**軸方向のデータ** (Numeric, "T" or "F")

object	**入力オブジェクト** (Series, Snapshot)
"T"	**時間軸**
"F"	**周波数軸**
"D"	**データ点数**

2. x : X**軸方向のデータ** (Numeric, "T" or "F")

object	**入力オブジェクト** (Series, Snapshot)
"T"	**時間軸**
"F"	**周波数軸**
"D"	**データ点数**

3. axs :XY**軸の形式** (0, 1, 2 or 3)

	X**軸**	Y**軸**
0	LINEAR	LINEAR
1	LOG	LINEAR
2	LINEAR	LOG
3	LOG	LOG

4. lne : **グラフ描画の形式**

0	**線のみを引く**
1	**センターシンボルのみを表示**
2	**線とセンターシンボルを表示**
3	**棒グラフを描く**
4	lne = 0 **および** lne = 3 **で表示する**
5	**零レベルとグラフで囲まれた領域を塗りつぶす**

5. step : **表示間隔** (step≥0, **整数型**)
(STEP–1) **点ごとのデータを対象に処理を行う**

6. sym1 : **センターシンボルの指定** (0≤sym1≤16, **整数型**)

Symbol0	□	Symbol1	○
Symbol2	△	Symbol3	＋
Symbol4	×	Symbol5	◇
Symbol6	■	Symbol7	●

Symbol8 ▲　　Symbol9 ▽
Symbol10 ▼　　Symbol11 ◆
Symbol12 □　　Symbol13 ○
Symbol14 △　　Symbol15 ◇
Symbol16 ▽

12**から**16**までは白抜きである**.

7. sym2：**センターシンボルの大きさ**[mm](sym2≥0, **実数型**)

解説)

1. 2**次元データの場合には**, **上位次元方向を時系列の方向と考え**, **下位次元の要素分の本数のグラフを作画する**.
2. **時間軸表示**

$$buff(i) = \frac{1000.0 \times float(i-1)}{sam}$$

3. **周波数軸表示**

$$delta = \frac{sam}{float(n-1) \times 2.0}$$
$$buff(i) = delta \times float(i-1)$$

4. **センターシンボルの大きさ**

   大きさは[mm] **で入力する**. **但し**, **零を入力したときは**0.5[mm] **に自動設定される**

ソースファイル　　Graph.c graph_sub.c graph_sub.h

GSOLM

機能) **直交座標系で定義される二変数関数** $Z = f(X, Y)$ **が矩形領域格子点上で与えられたとき**, **すなわち**

$$Z_{i,j} = f(X_i, Y_j) 0 \leq i \leq mx, 0 \leq j \leq my$$

が与えられたとき, **立体図を表示する**.

形式)

gsolm(x , rh1, rh2, f1, f2, f3, f4, col, div1, div2, dirc, view, mode [, size])

パラメータ)

1. x : **入力オブジェクト** (Series, Snapshot)
2. rh1, rh2 : **表示形式** (–1.0<rh1, rh2<1.0)
   (**この値によって表示される図の形状が変わる**)
3. f1 : **作画条件** 1(0 or 1)
   - 0 **隠線消去する**
   - 1 **隠線消去しない**
4. f2 : **作画条件** 2(0, 1 or 2)
   - 0 X**方向**, Y**方向ともに格子線を引く**
   - 1 X**方向のみ引く**
   - 2 Y**方向のみ引く**
5. f3 : **作画条件** 3(0, 1 or 2)
   - 0 **視点に近いへりを上側と下側ともに表示する**
   - 1 **上側のみ表示する**
   - 2 **下側のみ表示する**
6. f4 : **作画条件** 4(0~ 7)
   - 0 **図に座標軸と枠をつけない**
   - 1~ 7 **指定された色で**, 3**次元の座標軸を表示する**
7. col : **零平面表示形式** (0~ 7)
   - 0 **零平面を表示しない**
   - 1~ 7 **指定された色で零平面を表示する**
8. div1 : x**軸方向表示ステップ** (≥1, **整数型**)
9. div2 : y**軸方向表示ステップ** (≥1, **整数型**)
10. dirc : **表示方向**

“X” **オブジェクトの上位次元が** X **軸方向になる**

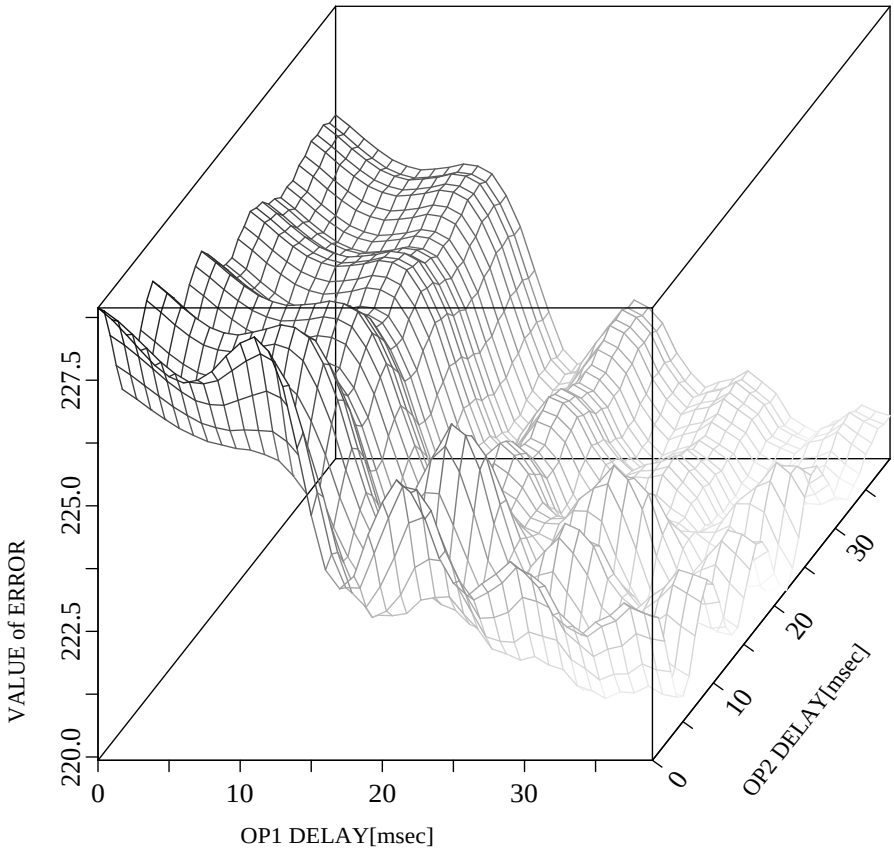
“Y” **オブジェクトの下位次元が** Y **軸方向になる**

11. view : **表示データの原点の指定** (1 : **左下が原点**, -1 : **左上が原点**)
12. mode : **表示モードの指定** (0 : **モノクロ**, 1 : **カラー**)
13. size : **座標軸につけるタイトルなどの文字の大きさ** (> 0mm)

解説)

1. **表示するデータの最大**, **最小値は**, “scale” **コマンドの** z **軸の値の設定により行なう**
2. $Z_{i,j} \leq Z_{min}$ **なら** $Z_{i,j} = Z_{min}$ **とし**, $Z_{max} \geq Z_{i,j}$ **なら** $Z_{i,j} = Z_{max}$ **とみなして表示する**.

使用例)



ソースファイル gsolm.c

GSTAT

機能) **グラフィックパラメータの現在値を表示する**.

形式) gstat()

パラメータ) **なし**

実行例)

```
[]SATELLITE[]~Demo/SIGMA:[271]% gstat()
-------------------------------------
GPM COMMON PARAMETERS
Current Window = No. 1
-------------------------------------
Paper Size: A4, Orientation: 1 (Landscape)
DVIFILEPATH = "/home/assie/SATELLITE/ComRef/Demo/SIGMA/GPMDVIFILE1"
-------------------------------------
Factor    : 1.0000
Origin    : (15.00 mm, 15.00 mm)
Size      : (100.00 mm, 100.00 mm)
Font Type : 1 (Times Roman)
Frame Line: Width 0, Type 0, Color 7
Graph Line: Width 0, Type 0, Color 5
Scale type: X=0, Y=0, Z=0 (0:Normal 1:Log)
Scale mode: X=1, Y=1, Z=0 (0:Auto 1:Fixed 2:Default)
Scale X = min:0, max:3
Scale Y = min:1, max:4
Scale Z = min:219.934, max:228.94
TITLE 1 :
  X: "OP1 DELAY[msec]"
  Y: "OP2 DELAY[msec]"
  Z: ""
TITLE 2 :
  X: ""
  Y: ""
  Z: ""
TITLE 3 :
  X: ""
  Y: ""
  Z: ""
-------------------------------------
```

ソースファイル gstat.c

LABEL

機能) **グラフィック画面に文字列を書く**.

形式) **label(ctyp, xorg, yorg, high, th [, str [, mode]])**

パラメータ)

1. ctyp : **設定形式**

   "I" **インディックス入力によって座標を設定**

   "F" **インディックス入力によって座標を設定し**, **文字列をファイルから読み込む**

2. xorg, yorg : **書き始める座標** [mm](**相対座標**)
3. high : **書く文字のサイズ** [mm]
4. th : **書く文字の角度** (0, 90, 180, 270)[deg]
5. str :

   ctyp="I" **文字列** ("" **で囲む**, 30 **文字以内**)(**省略可能**)

   ctyp="F" **ファイル名**
6. mode : **表示モード** (0 : Left, 1 : Right, 2 : Center) **指定した座標を**, **文字列の左端**, **右端**, **中央として表示する**.

解説)

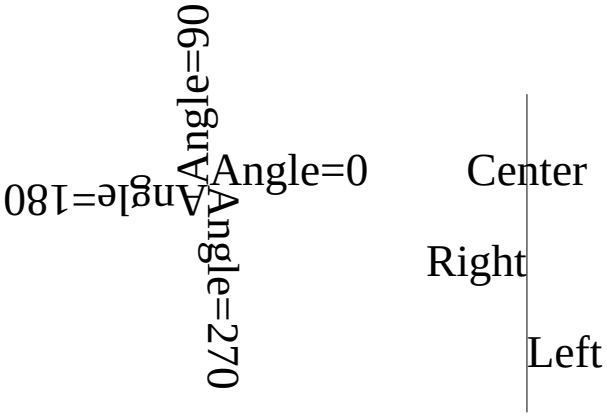
1. **処理**

   ctyp = "I" **の時** : **文字列を省略すると**

   >

   が表示されて入力待ちになるので, **表示したい文字列を入力する**. **なお**, **この入力方法は** 1 **度の入力で** 80 **文字まで表示することができる**.

使用例)



ソースファイル label.c

LINE

機能)　　**グラフィック画面にラインをひく**.

形式)　　**line(xorg, yorg, xend, yend, mode)**

パラメータ)

1. xorg, yorg : **始点座標**[mm](origin **コマンドで設定した座標からの相対座標**)
2. xend, yend : **終点座標**[mm](origin **コマンドで設定した座標からの相対座標**)
3. mode : "L"(**直線**), "B"(**矩形**), "BF"(**矩形内を塗りつぶす**)

使用例)

ソースファイル　　Line.c

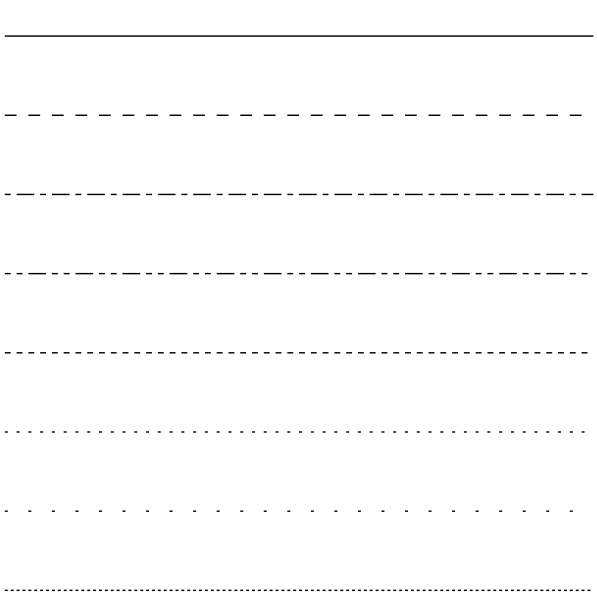
LTYPE

機能) **図形を出力する場合の線種を指定する**

形式) **ltype(ilt1, ilt2)**

パラメータ)

1. ilt1 : **グラフの線種** (1≤ilt1≤8)
2. ilt2 : **座標軸**, **フレームの線種**



ソースファイル ltype.c

LWIDTH

機能) **グラフや座標軸の線幅を指定する**.

形式) **lwidth(ilt1, ilt2)**

パラメータ) 1. ilt1 : **グラフ線幅** (1≤ilt1≤4)

2. ilt2 : **座標軸**, **フレームの線幅** (1≤ilt2≤4)

1:

2:

3:

4:

ソースファイル lwidth.c

MAP

機能) 2**次元データのカラーマップ表示を行う**.

形式) **map(x, dirc, style, psize, view [, time, start, end, step])**

パラメータ) 1. x : **入力オブジェクト** 2**次元または**3**次元** (Series, Snapshot)

2. dirc : **表示方向** ("X" or "Y")

"X" **データの上位次元方向を** X **軸とする**

"Y" **データの下位次元方向を** X **軸とする**

3. style : **表示形式** (0 ~ 7, **整数**)

0 **パターンの色が変化**

1 **パターンのサイズが変化** (**パターンの色は** COLOR **コマンドで設定する**. **データ** >0 : **第**1**パラメータ**, **データ** <0 : **第**2**パラメータ**)

2 **パターンの色とサイズが変化**

3 **データが** min **と** max **の範囲に正規化され**, **パターンのサイズが変化** (fill)

4 style=3 **において**, **パターンを** open **で表示**

5 style=3 **において**, **パターンの色が変化**

6 **パターンサイズが変化し**, **データ** >0 : open, **データ** <0 : fill

7 **パターンサイズが変化し**, **データ** >0 : fill, **データ** <0 : open

8–14 **パターンの色が**, 8:**青**, 9:**赤**, 10:**マゼンタ**, 11:**緑**, 12:**シアン**, 13:**黄色**, 14:**白の強弱で変化する**.

4. psize : **パターンサイズ** (0 or 1)

0 MAP **全体を** SIZE **で指定された大きさで表示** (**デフォルト**)

1 **各パターンを正方形で表示**

5. view : **表示データの原点の指定** (1 : **左下が原点**, -1 : **左上が原点**)

6. time : **描画する間隔** msec **単位**

7. start : **表示開始点**

8. end : **表示終了点** (0**のときはデータの最後まで表示する**)

9. step : **表示データのステップ** (1**以上**, 0**のときは** 1 **と同じ**)

解説) 1. **表示データの範囲**

"scale" **コマンドの** z **軸の設定にしたがって**, **表示データの最大**, **最小値を設定する**.

2. PostScript **プリンタへの出力**

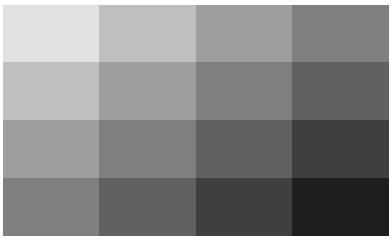
style=0, 2, 5, 8–14 では, 表示データが gray スケールの濃淡レベルに変換される (データの大 → 小 に対して, 黒 → 白に変化).
その他の style では, color コマンドの出力と同様.

3. time, start, end, step は, データが 3 次元の場合のみ有効である.

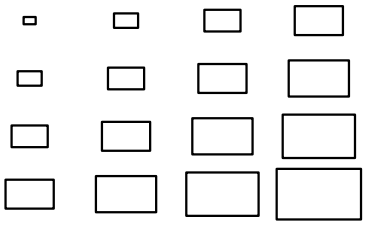
実行例) 以下のデータに対する PostScript プリンタへの出力結果を示す.
color(4,2); lwidth(2,1); scale("N","A","N","A","N","F",-3,5); を指定

$$\begin{aligned}
\mathrm{dat}:[0] &= (\ -2,\ -1,\ 0,\ 1)\\
\mathrm{dat}:[1] &= (\ -1,\ 0,\ 1,\ 2)\\
\mathrm{dat}:[2] &= (\ 0,\ 1,\ 2,\ 3)\\
\mathrm{dat}:[3] &= (\ 1,\ 2,\ 3,\ 4)
\end{aligned}$$

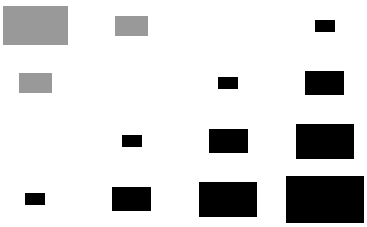
map(dat,"Y",0,0,-1)



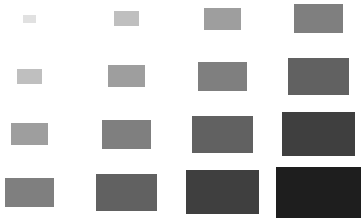
map(dat,"Y",4,0,-1)



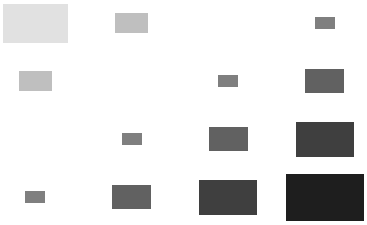
map(dat,"Y",1,0,-1)



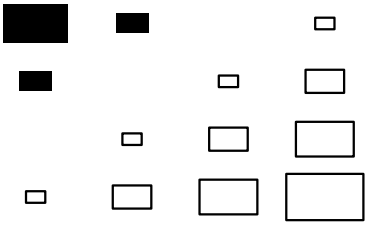
map(dat,"Y",5,0,-1)



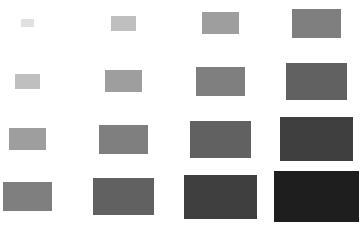
map(dat,"Y",2,0,-1)



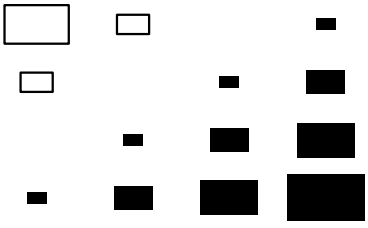
map(dat,"Y",6,0,-1)



map(dat,"Y",3,0,-1)



map(dat,"Y",7,0,-1)



ソースファイル map.c

NEWPAGE

機能) **プリンタの改ページを行う**.

形式) newpage()

パラメータ) **なし**

解説) **ポストスクリプトファイルに複数のページを出力できる**

ソースファイル roll.c

ORIGIN

機能) **グラフィック相対座標の原点を指定する**.

形式) **origin(xorg, yorg)**

パラメータ) 1. xorg : X**軸の原点**[mm](xorg≥0.0, **実数型**)

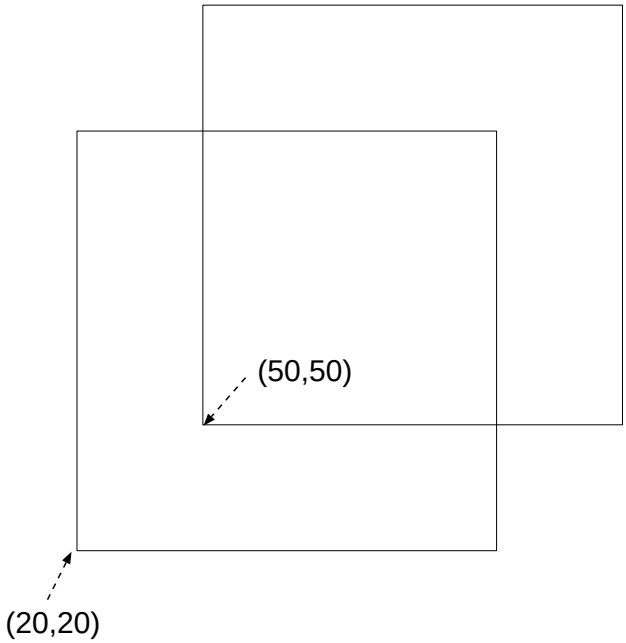
2. yorg : Y**軸の原点**[mm](yorg≥0.0, **実数型**)

解説) 1. **原点の指定は**, **絶対座標**[mm] **で行う**

使用例)

```
origin(20,20); frame();
origin(50,50); frame();
```

で,



と出力される. **ウィンドウ左上に表示されるマウスカーソルの座標を目安にするとよい**. **また**, **ウィンドウ上の座標値は**Copy/Paste **することができる**.

ソースファイル origin.c

PREV

機能) **中間ファイル** (GPMDVIFILE) **の内容を** *SATELLITE* **ウィンドウに表示する**.

形式) **prev([filename])**

パラメータ) filename : GPMDVIFILE **の名前** (String)

解説) **省略した場合は**, "GPMDVIFILE1" **のデータを表示する**.

ソースファイル prev.c

SCALE

機能) **グラフィック表示のスケールの指定**.

形式)

```
scale( xtyp, xaxs, ytyp, yaxs [, scl1, scl2 ][, scl3, scl4] )
scale( xtyp, xaxs, ytyp, yaxs, ztyp, zaxs
                                  [, scl1, scl2 ][, scl3, scl4][, scl5, scl6] )
```

パラメータ)

1. xtyp : X**軸のスケール形式** (“N” or “L”)
   “N” LINEAR**軸**
   “L” LOG**軸**
2. xaxs : X**軸のスケール設定方式** (“A”, “F” or “D”)
   “A” **自動設定**
   “F” **マニュアル設定**
   “D” **デフォルト値設定** (**前回処理したときの値**)
3. ytyp : Y**軸のスケール形式** (“N” or “L”)
   “N” LINEAR**軸**
   “L” LOG**軸**
4. yaxs : Y**軸のスケール設定方式** (“A”, “F” or “D”)
   “A” **自動設定**
   “F” **マニュアル設定**
   “D” **デフォルト値設定** (**前回処理したときの値**)
5. ztyp : Z**軸のスケール形式** (“N” or “L”)
   “N” LINEAR**軸**
   “L” LOG**軸**
6. zaxs : Z**軸のスケール設定方式** (“A”, “F” or “D”)
   “A” **自動設定**
   “F” **マニュアル設定**
   “D” **デフォルト値設定** (**前回処理したときの値**)
7. scl1 x**軸の最小スケール値**
8. scl2 x**軸の最大スケール値**
9. scl3 y**軸の最小スケール値**
10. scl4 y**軸の最大スケール値**
11. scl5 z**軸の最小スケール値**

12. scl6 z **軸の最大スケール値**

解説) 1. **入力を省略した場合**, **現在のスケール値を表示して入力待ちとなる**.

2. *SATELLITE***起動時には全軸自動設定になっている**.

ソースファイル scale.c

SIZE

機能) **描画領域のサイズを指定する**.

形式) **size(len1, len2)**

パラメータ) 1. len1 : X**軸方向の長さ** [mm]

2. len2 : Y**軸方向の長さ** [mm]

解説) 1. *SATELLITE***起動時には** len1=100[mm], len2=100[mm] **に設定されている**.

ソースファイル size.c

TITLE

機能) **軸を表示する際のタイトルを指定する**.

形式) **title(tbuf [, "x-title", "y-title" [, "z-title"]])**

パラメータ)

1. tbuf : **タイトル番号** (1, 2, 3)
2. "x-title" : X **軸のタイトル** (30 **文字以内**)
3. "y-title" : Y **軸のタイトル** (30 **文字以内**)
4. "z-title" : Z **軸のタイトル** (30 **文字以内**)

解説)

1. **タイトル設定法**

   x-title, y-title **が省略された場合**, **現在設定されているタイトルを表示して入力待ちになる**.

2. z **軸のタイトル設定について**

   z-title **は**, x-title, y-title **とも省略した場合**, **現在設定されているタイトルを表示して入力待ちになる**.

ソースファイル title.c

WCLOSE

機能) *SATELLITE*グラフィックウィンドウをクローズする.

形式) **wclose(wnum)**

パラメータ) 1. wnum : グラフィックウィンドウ番号 (Scalar) なお, "ALL" (String) を入力した場合は, 開いている全ウィンドウをクローズする.

ソースファイル wclose.c

WE

機能) *SATELLITE*グラフィックウィンドウの画面を消去する.

形式) we()

パラメータ) なし

ソースファイル werase.c

WOPEN

機能) *SATELLITE*グラフィックウィンドウをオープンする.

形式) **wopen(wnum, size, orientation, device [, X, Y])**

パラメータ)

1. wnum : グラフィックウィンドウ番号
2. size : "A4", "B4", "Free" (Free 指定時は X,Y を指定する)
3. orientation: 0–Portrait, 1–Landscape
4. device: 0–Display Only, 1–Display and File, 2–File Only
5. X : オープンするウィンドウのX方向サイズ [pixels]
   device を 1 に指定した場合は, GPMDVIFILE のファイル名 (string)．ファイル名を省略した場合は, GPMDVIFILEn で n は, ウィンドウ番号 (wnum) が使われる.
6. Y : オープンするウィンドウのY方向サイズ [pixels]

解説)

- マウスをウィンドウ内に移動した時には, ウィンドウ左上にポインタの座標が表示される (size=A4 or B4 のとき単位は [mm]，Free の場合は [pixels]).
- ウィンドウ内でマウスをクリックすると, X Window のカットバッファに x,y 座標値がコピーされる. マウスの中ボタンをクリックすれば, エディタ上に座標値をペーストできる.
- size を Free とした場合, device はディスプレイのみに設定される.

ソースファイル wopen.c

ニューラルネットシミュレータ

BPS

ACTLOAD

機能) **テスト**(**認識**)**結果ファイルから**snapshot**へ活性値をロードする**.

形式) **buf = actload(pnum, unum, fname)**

パラメータ)

1. pnum : **パターンの軸の指定**, **またはパターン番号の指定**(X, Y or num)
   - X **パターンを**X**軸にとる**
   - Y **パターンを**Y**軸にとる**
   - num **その番号の活性値を対象とする**
2. unum : **ユニットの軸の指定**, **またはユニット番号の指定**(X, Y or num)
   - X **ユニットを**X**軸にとる**
   - Y **ユニットを**Y**軸にとる**
   - num **その番号の活性値を対象とする**
3. fname : **テスト**(**認識**)**結果ファイル名**

戻り値) buf : **活性値を出力するオブジェクト** (Snapshot)

ソースファイル getparam.c loadlib.c actload.c

BPLOAD

機能) **シミュレーションに必要なパラメータをパラメータファイルから読み込む**.

形式) **bpload(fname)**

パラメータ) 1. fname : **パラメータファイル名**

解説) 1. **パラメータファイルの識別子には自動的に** ".prm" **が付けられる**.

参照) BPSAVE

ソースファイル bpload.c

BPSAVE

機能) **現在設定されているシミュレーションに必要なパラメータをパラメータファイルに保存する**

形式) **bpsave(fname)**

パラメータ) 1. fname : **パラメータファイル名**

解説) 1. **パラメータファイルの識別子には自動的に** ".prm" **が付けられる**.

2. **パラメータはコマンドリファレンス** "PINIT" **もしくは** User's Manual **表** 6.3 **を参照して下さい.**

参照) BPLOAD

ソースファイル bpsave.c

COR

機能) **バッファのデータに対し, 相関を計算し, データを 2 次元バッファに相関マトリクスとして格納する.**

形式) **bufout = cor(buf, mode)**

パラメータ)
1. buf : **データが格納されている** Snapshot
2. mode : **相関モード** (C or D)
   C:**相関**
   D:**ユークリッド距離**

戻り値) bufout : **ストア先の** Snapshot

ソースファイル correlat.c

DISP

機能) **学習回数**, **誤差値などの表示間隔を設定する**.

形式) **disp(dint, str)**

パラメータ) 1. dint : **学習回数**, **誤差**, **コメント等の表示間隔**

2. str : **学習時に表示するコメント**

ソースファイル disp.c

ERRFUNC

機能) **評価関数を演算し**, **データを** 2 **次元バッファへ格納する**.

形式)

```
buf1 = errfunc(his1, l1, us1, ud1, min1, max1, d1, l2, us2, ud2,
min2, max2, d2 )
```

パラメータ)

1. his1 : **ウェイトヒストリ** (**ブロック**) **番号**
2. l1 : **結合元層番号**
3. us1 : **結合元ユニット番号**
4. ud1 : **結合先ユニット番号**
5. min1 : **結合重み最小値**
6. max1 : **結合重み最大値**
7. d1 : **データポイント数**
8. his2 : **ウェイトヒストリ** (**ブロック**) **番号**
9. l2 : **結合元層番号**
10. us2 : **結合元ユニット番号**
11. ud2 : **結合先ユニット番号**
12. min2 : **結合重み最小値**
13. max2 : **結合重み最大値**
14. d2 : **データポイント数**

戻り値) buf1 : **格納先の** Snapshot

解説)

1. **指定した結合重み** (1 and 2) **を変化させ**, **評価関数を計算する**. **このとき**, **指定した結合重み以外は** SETREC **で指定したウェイトヒストリ番号の結合重みに固定してある**.

ソースファイル

charfunc.c filehndl.c getparam.c nethndl.c
learnlib.c errfunc.c

ERRLOAD

機能) **エラーヒストリファイルから誤差値を** Snapshot **へロードする**.

形式) **buf = errload(it, unum, fname)**

パラメータ)

1. it : **学習回数の軸の指定**, **またはヒストリ番号の指定**(X, Y or num)
   - X **学習回数を**X**軸にとる**
   - Y **学習回数を**Y**軸にとる**
   - num **その番号の誤差を対象とする**
2. unum : **ユニットの軸の指定**, **またはユニット番号の指定**(X, Y, S or num)
   - X **ユニットを**X**軸にとる**
   - Y **ユニットを**Y**軸にとる**
   - S **総和誤差を対象とする**
   - num **その番号のユニット誤差を対象とする**
3. fname : **エラーヒストリファイル名**

戻り値) buf : **誤差値を格納するオブジェクト** (Snapshot)

ソースファイル getparam.c loadlib.c errload.c

ERROR

機能) **エラーヒストリファイルに関する設定を行う**.

形式) **error(fname, sint, sdir [, smod])**

パラメータ)

1. fname : **エラーヒストリファイル名**
2. sint : **エラーヒストリストア間隔**
3. sdir : **エラーヒストリストア方向** (R or D)
   - R **レコード方向**
   - D **データポイント方向**
4. smod : **エラーヒストリーストアモード** (A or O)
   - A **追加**
   - O **上書き**

解説)

1. **エラーヒストリファイルの識別子には自動的に** ".bhe" **が付けられる**.
2. smod(**エラーヒストリストアモード**) **は**, sdir **がレコード方向の時のみ設定する**.

ソースファイル error.c

FUNCTION

機能) **各層のユニット特性とバイアスユニットの有無を設定する**.

形式) **function([p1, p2, p3, ···, p10])**

パラメータ)

1. p1 : **入力層のユニットの特性**, **バイアスユニットの有無の設定** (LN,LA,SN or SA)

	LN	LA	SN	SA
ユニット特性	Linear	Linear	sigmoid	sigmoid
バイアスユニット	**無**	**有**	**無**	**有**

2. p2 : **第** 2 **層目のユニットの特性**, **バイアスユニットの有無の設定** (LN,LA,SN or SA)
3. p_i : **第** i **層目のユニットの特性**, **バイアスユニットの有無の設定** (LN,LA,SN or SA)

解説)

1. LAYER **で設定した層数分のパラメータを入力する**. **入力を略した場合**, **現在のユニット特性**, **バイアスの有無を表示して入力待ちとなる**.
2. LAYER **で層数を設定した後に**, **本コマンドを実行すること**.

参照) LAYER

ソースファイル function.c

機能) **学習アルゴリズムと学習パラメータを設定する**.

形式) **lalgo(mode, alg, p1 [, p2, p3, p4, p5, p6])**

パラメータ)

1. mode : **一括学習**, **逐次学習の選択**
   - S **一括学習**
   - P **逐次学習**
2. alg : **学習アルゴリズム** (1∼5)
   - 1 steep **法**
   - 2 momentum **法**
   - 3 Vogl **法**
   - 4 Jacobs **法**
   - 5 momentum Vogl **係数法**
   - 6 **落合法**
3. p1 : **学習率**
4. p2 : **慣性率**
5. p3 : Vogl&**落合アルゴリズム用学習率増加係数**
6. p4 : Vogl&**落合アルゴリズム用学習率減少係数**
7. p5 : Vogl **アルゴリズム用閾値**
8. p6 : **落合アルゴリズム用平滑化微係数**
9. str : **構造学習法** (0∼3)
   - 0 **構造学習無し**
   - 1 Weight Decay **法**
   - 2 **忘却付学習法**
   - 3 **側抑制学習法**
10. ps : **構造学習用効果率係数**

解説)

1. **学習パラメータ** (p1∼p5,str,ps) **は**,alg **により次のように設定する**.
   入力を略した場合, **現在の学習パラメータを表示して入力待ちとなる**.
   構造学習法str **を指定し、**sp **の入力を略した場合**, **現在の学習パラメータを表示して入力待ちとなる**. str **以前を略した場合**, **構造学習は行われないものと判断され、**str,sp **は入力待ちにならない**.

alg	p1	p2	p3	p4	p5	p6	str	ps
1	○	—	—	—	—	—	—	—
2	○	○	—	—	—	—	○	○
3	○	○	○	○	○	—	—	—
4	○	○	○	○	—	—	—	—
5	○	○	—	—	—	—	—	—
6	○	○	○	○	—	○	○	○

ソースファイル lalgo.c

LAYER

機能) **ネットワークの層数, 各層のユニット数を設定する.**

形式) **layer(n [, lnum1, lnum2, lnum3, ···, lnum10])**

パラメータ)

1. n : **層数** (3≤N≤10)
2. lnum1 : **入力層のユニット数**
3. lnum2 : **第2層目のユニット数**
4. lnum_i : **第i層目のユニット数**

解説)

1. **指定した層数分だけユニット数を入力する. ユニット数の入力を略した場合, 現在のユニット数を表示して入力待ちとなる.**

ソースファイル layer.c

LEARN

機能) **学習を行う**.

形式) **learn(buf)**

パラメータ) 1. buf : **バッファ名** (0 or name)

0 **総和誤差をバッファに出力しない**

name **総和誤差を出力するバッファ名**

解説) 1. **あらかじめ設定コマンド群を実行し**, **学習に必要なパラメータを設定するか**, BPLOAD **を実行し**, **パラメータファイルからパラメータを読み込んでおく必要がある**.

2. *SATELLITE* **のバッファモニタ機能を利用して**, **学習誤差のリアルタイム表示が可能である**. **利用時には**, **あらかじめ** buf **と対応するモニタ機能を起動しておく必要がある**.

参照) LAYER, FUNCTION, WEIGHT, ERROR, TEACH, LALGO, LEND, DISP, BM

ソースファイル

charfunc.c filehndl.c getparam.c nethndl.c
wgtrenew.c learnlib.c learn.c

LEND

機能) **学習の終了条件を設定する**.

形式) **lend(err, n)**

パラメータ) 1. err : **学習終了誤差**

2. n : **学習回数**

解説) 1. **総和誤差が**err **以下か**, **学習回数が**n **回になれば学習を終了する**.

ソースファイル lend.c

PDISP

機能) BPS **パラメータの現在の設定値を表示する**.

形式) **pdisp()**

パラメータ) **なし**

参照) BPLOAD,BPSAVE

ソースファイル pdisp.c

PINIT

機能) **シミュレーションに必要な全てのパラメータの初期設定を行う**.

形式) **pinit()**

パラメータ) **なし**

解説) **各パラメータの初期設定状態は**, **以下のようになっている**.

1. **層数**：0
2. **ユニット数**：0
3. **ユニットの特性**：EOS
4. **結合重みの初期値生成アルゴリズム**：ramdom
5. **乱数の種**：1
6. **結合重み初期値 最大値**：1.0
7. **結合重み初期値 最小値**：-1.0
8. **イニッシャライズウェイトファイル名**：EOS
9. **ウェイトヒストリファイル名**：EOS
10. **ウェイトストアインターバル**：1
11. **ウェイトストアモード**：append
12. **エラーヒストリファイル名**：EOS
13. **エラーストアインターバル**：1
14. **エラーストアディレクトション**：record
15. **エラーストアモード**：append
16. **入力データファイル名**：EOS
17. **教師データファイル名**：EOS
18. **入力開始パターン番号**：0
19. **入力最終パターン番号**：0
20. **学習モード**：**一括学習**
21. **学習アルゴリズム**：momentum
22. **学習率**：0.0
23. **慣性率**：0.0
24. vogl**アルゴリズム用増加係数**：0.0

25. vogl**アルゴリズム用減少係数**：0.0
26. vogl**アルゴリズム用閾値**：0.0
27. **落合アルゴリズム用ファクタ**：0.0
28. **学習終了誤差**：0.0
29. **学習回数**：0
30. **表示間隔**：0
31. **コメント**：EOS
32. **テスト用ウェイトヒストリファイル名**：EOS
33. **ウェイヒストリ番号**：0
34. **テストデータファイル名**：EOS
35. **テストデータ入力開始パターン番号**：0
36. **テストデータ入力最終パターン番号**：0
37. **入力層番号**：0
38. **出力層番号**：0
39. **テスト結果ファイル名**：EOS

ソースファイル pinit.c

REC

機能) **ネットワークのテスト (認識) 及びネットワーク構造表示を行う**.

形式) **rec(stor, disp [, mode, min, max, fname1, fname2])**

パラメータ) 1. stor : SETREC **により設定したテスト (認識) 結果ファイルへのストア**

0 **ストアしない**

1 **ストアする**

2. disp : **ネットワーク構造表示**

0 **表示しない**

1 **表示する**

3. mode : **表示モード**

0 **活性値を色の変化で表示する**

1 **活性値を正方形の大きさで表示する**

2 **認識を行わず, 構造表示のみを行う (特性関数のマークを書く)**

3 **認識を行わず, 構造表示のみを行う (特性関数のマークは書かない)**

4. min : **スケール最小値**

5. max : **スケール最大値**

6. fname1 : **教師データファイル名 (教師データも共に表示する)**

7. fname2 : **エラーファイル名 (教師データとの誤差をこのファイルにストアする)**

解説) 1. **あらかじめ** BPLOAD **を実行してパラメータを読み込んでおくか, 設定コマンドを実行して必要なパラメータを設定しておく必要がある**.

2. **教師データファイル名, エラーファイル名は省略しても入力待ちにはならない**.

参照) LAYER, FUNCTION, SETREC

ソースファイル

charfunc.c	filehndl.c	getparam.c	nethndl.c
trace.c	testlib.c	test.c	

RVMAP

機能) **結合加重の逆投影演算を行い**, 2 **次元バッファ格納する**.

形式) **buf = rvmap(lay1, lay2, his)**

パラメータ)
1. lay1 : **出力層番号**
2. lay2 : **入力層番号**
3. his : **ウェイトヒストリ番号**

戻り値) buf : **格納先**Snapshot

ソースファイル charfunc.c filehndl.c getparam.c nethndl.c rvmap.c

SETREC

機能) **ネットワークのテスト (認識) に関するパラメータを設定する**.

形式) **setrec(fname1, hisno, fname2, ps, pe, ilay, olay, fname3)**

パラメータ)

1. fname1 : **ウェイトファイル名**
2. hisno : **ウェイトヒストリ番号**
3. fname2 : **入力データファイル名**
4. ps : **入力開始パターン番号**
5. pe : **入力最終パターン番号**
6. ilay : **入力層番号**
7. olay : **出力層番号**
8. fname3 : **テスト (認識) 結果出力ファイル名**

解説)

1. **ウェイトファイル** (fname1) **の識別子には自動的に** ".bhw" **が付けられる**
2. **ウェイトヒストリ番号** (hisno) **は**, **ブロック番号に対応する**.
3. **入力データファイル** (fname2) **の識別子には自動的に** ".dat" **が付けられる**
4. **出力ファイル** (fname3) **の識別子には自動的に** ".brc" **が付けられる**
5. **入力層を第** 0 **層とする**

ソースファイル setrec.c

SIGMOID

機能) SIGMOID **曲線上に活性値をプロットする**. **または**, **入力総和値のヒストグラムを描く**.

形式) **sigmoid(lay, unit, mode, n)**

パラメータ)

1. lay : **対象ユニットの層番号**
2. unit : **対象ユニット番号**
3. mode : **モード** (P or H)
   P:**シグモイド曲線上に活性値をプロットする**.
   H:**入力総和のヒストグラムを描く**
4. n : mode **が** P **のときには**, **シンボル番号** (GRAPH **コマンド参照**), H **のときには**, **分割数である**.

ソースファイル

charfunc.c filehndl.c getparam.c nethndl.c testlib.c sigmoid.c

TEACH

機能) **入力データ, 教師データに関する設定を行う.**

形式) **teach(fname1, fname2, ps, pe)**

パラメータ)
1. fname1 : **入力データファイル名**
2. fname2 : **教師データファイル名**
3. ps : **入力開始パターン番号**
4. pe : **入力最終パターン番号**

解説)
1. **データファイルの識別子には自動的に** ".dat" **が付けられる**

ソースファイル teach.c

WALGO

機能) **ウェイトの初期値生成に必要なパラメータを設定する**.

形式) walgo(alg, fname, seed, rmax, rmin)

パラメータ)

1. alg : **初期値生成アルゴリズム** (J or R)
   - J JIA **のアルゴリズム**
   - R **一様乱数**
2. fname : **出力する初期値ファイル名**
3. seed : **初期値の種**
4. rmax : **生成する初期値の範囲** (**最大値**)
5. rmin : 〃 (**最小値**)

解説)

1. **パラメータファイルの識別子には自動的に** ".bhw" **が付けられる**

ソースファイル walgo.c

WEIGHT

機能) **ウェイトヒストリファイルに関する設定を行う**.

形式) **weight(fname1, fname2, sint, smod)**

パラメータ)

1. fname1 : **ネットワークを構成するためのウェイトが格納されているファイル名**
2. fname2 : **ウェイトヒストリーファイル名**
3. sint : **ウェイトヒストリーストア間隔**
4. smod : **ウェイトヒストリーストアモード** (A or O)
   - A **追加** (**レコード方向にアペンド**)
   - O **上書き** (**オーバーライト**)

解説)

1. **パラメータファイルの識別子には自動的に** ".bhw" **が付けられる**

ソースファイル weight.c

WGTLOAD

機能) **ウェイトヒストリファイルから** Snapshot **へ結合重み値をロードする**.

形式) **buf = wgtload(it, lnum, unum1, unum2, fname [, bias])**

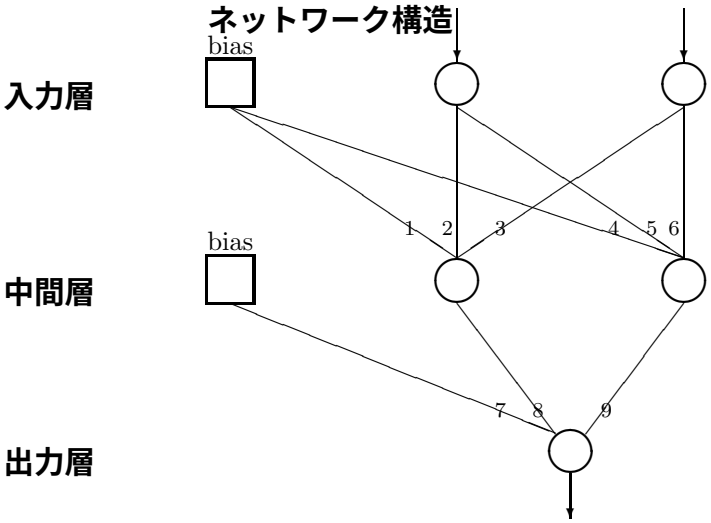
パラメータ)

1. it : **学習回数の軸の指定**, **及びヒストリ番号の指定** (X, Y or num)
   - X **学習回数を** X **軸にとる**
   - Y **学習回数を** Y **軸にとる**
   - num **その番号のウェイトを対象とする**
2. lnum : **ウェイト結合元の層番号** ($\geq$0)
3. unum1 : **結合元ユニットの軸の指定**, **及びユニット番号の指定** (X, Y or num)
   - X **結合元ユニットを** X **軸にとる**
   - Y **結合元ユニットを** Y **軸にとる**
   - num **その番号のユニットを結合元とするウェイトを対象とする**
4. unum2 : **結合先ユニットの軸の指定**, **及びユニット番号の指定** (X, Y or num)
   - X **結合先ユニットを** X **軸にとる**
   - Y **結合先ユニットを** Y **軸にとる**
   - num **その番号のユニットを結合先とするウェイトを対象とする**
5. fname : **ウェイトヒストリファイル名**
6. bias : **結合元のユニットとしてバイアスも対象に加える場合**, 1 **を入力する**

戻り値) buf : **結合重み値を格納するオブジェクト** (Snapshot)

解説)

1. 2 **次元バッファの** X **軸は**, **学習**, **結合元ユニット**, **結合先ユニットのうち**, 1 **つのみ指定できる**. **例えば**, **学習回数と結合元ユニットの両方を** X **軸に指定することはできない** (Y **軸についても同様**).
2. **使用例**



入力例

buf = wgtload(1, 0, X, Y, filename)

- **学習番号**： 1
- **結合元の層番号**： 0(**入力層**)
- **結合元のユニット**：X**軸**
- **結合先のユニット**：Y**軸**

2**次元**snapshot

X**方向** (**入力層ユニット**No.)

Y**方向** (**中間層ユニット**No.)

重み値2	**重み値**3
重み値5	**重み値**6

ソースファイル

charfunc.c filehndl.c getparam.c nethndl.c loadlib.c wgtload.c

WGTSET

機能) **ウェイトファイルに任意のリンクの重みを設定する**.

形式) wgtset(lay, us, ud, data)

パラメータ)

1. lay : **層番号**
2. us : **リンク結合元のユニット番号**
3. ud : **リンク結合先のユニット番号**
4. data : **重み値**

ソースファイル wgtset.c

WINIT

機能) **ウェイトの初期値を生成し, 設定したファイルに格納する**.

形式) **winit()**

パラメータ) **なし**

解説) 1. **あらかじめ**BPLOAD**を実行してパラメータを読み込んでおくか, 設定コマンドを実行して必要なパラメータを設定しておく必要がある**.

参照) LAYER, FUNCTION, WALGO

ソースファイル

charfunc.c filehndl.c getparam.c nethndl.c winit.c

神経回路シミュレータ

NCS

NASSIGN

機能) NCS**で使用するモデルファイル名を定義する**.

形式) **nassign(filename {, header} {, library})**

パラメータ)

1. filename : **ファイル名**
2. header : **ユーザ関数用ヘッダーファイル名**
3. library : **ユーザ関数用ライブラリ名**

解説)

1. **モデルファイルの識別子は自動的に** ".mdl" **が付けられる**.
2. assign **されたモデルファイル名は**, SATELITE **のワークエリア内のファイル** SCFN.NCS **に格納される**.
3. **パラメーター** 2, 3 **の** header, library **は**, **拡張子が** ".h" **の場合ヘッダーファイル名と判断され**, **それ以外はライブラリ名と判断される**.
4. **ユーザ関数用ヘッダーファイル名はフルパスまたは**, **相対パスで指定する**.
5. **ユーザ関数用ライブラリ名はフルパスおよびフルネームで指定する**.
6. **ファイル名に**"?" **を指定した場合**, **現在** assign **されているファイル名**, **ユーザ関数用ヘッダーファイル名**, **ユーザ関数用ライブラリ名が表示される**.

ソースファイル nasgn.c

NPP

機能) NCS 言語を C 言語に変換する.

形式) **npp({model } {, header})**

パラメータ)

1. model : **モデルファイル名**
2. header : **ユーザ関数用ヘッダーファイル名**

解説)

1. **モデルファイルの識別子は自動的に** ".mdl" **が付けられる**.
2. **モデルファイル名を指定して** NPP **を実行すると自動的に指定したモデルファイルが** assign **される**.
3. **モデルファイル名を省略した時は**, assign **されているモデルファイルが対象となる**.
4. **プリプロセッサを通すと** C **言語のソースファイル**, **ヘッダファイル**, **シミュレーション条件ファイル群**, assign **ファイルが** SATELITE **のワークエリア内に生成される**.
5. **ユーザ関数用ヘッダーファイル名を指定して** NPP **を実行すると自動的に指定しヘッダーファイルが** assign **される**.
6. **ユーザ関数用ヘッダーファイル名はフルパスまたは**, **相対パスで指定する**.
7. **ユーザ関数用ヘッダーファイル名を省略した時は**, assign **されていれば**, **そのユーザ関数用ヘッダーファイルが対象となる**.

ソースファイル nnpp.l nnpp.y list.c ncs_func_lib.c

NLINK

機能) NPP で変換された C 言語のソースファイルをコンパイルし, NCS ライブラリとのリンクを行い, 実行形式を作成する.

形式) **nlink({opt} {, flag} {, library })**

パラメータ)

1. opt : **コンパイル時の最適化レベルの指定** (**コンパイラに依存**)
   - –g **デバッガを使用するための指定**
   - –O～–O4 **最適化レベル** 1～4
2. flag : **エラー** or **オブジェクトファイル用フラグ**
   - E **エラーファイルを出力する**.
   - O **オブジェクトファイルを出力する** (NPE **のため**).
   - N **エラーファイル**, **オブジェクトファイルとも出力する**.
   - **その他 どちらも出力しない**.
3. library : **ユーザ関数用ライブラリ名**

解説)

1. NLINK **は**, shell **スクリプトも用いている**.
2. **コンパイルとリンクには** “cc” **または** “gcc” **を使用する**. “cc” **か** “gcc” **かは**, NCS **を** make **した時にどちらが使用されたかによって決定されている**.
3. **コンパイル時には**, **ヘッダファイルのサーチパスに**
   “/usr/local/lib/sl-include”
   ライブラリのサーチパスに
   “/usr/local/lib/sl-lib”
   をそれぞれ加えている.
4. flag **省略時はエラーファイル** (**拡張子** “.err”), **オブジェクトファイル** (**拡張子** “.o”) **とも出力されない**.
5. **最適化レベルなど**, **コンパイル**, **リンクオプションの詳細は**, UNIX **のマニュアルなどを参照のこと** (cc, gcc **のバージョンや** OS **等に依存している**).
6. **ユーザ関数用ライブラリ名を指定して** NLINK **を実行すると自動的に指定しライブラリ名が** assign **される**.
7. **ユーザ関数用ライブラリ名はフルパスおよびフルネームで指定する**.
8. **ユーザ関数用ライブラリ名を省略した時は**, assign **されていれば**, **そのユーザ関数用ライブラリが対象となり**, **そうでない場合はライブラリを使用しない**.

ソースファイル nlink.c

機能) **プリプロ**&**コンパイル**&**リンクの自動実行をする**.

形式) nmake()

パラメータ) 1. **なし**

解説) 1. **作成したモデルソースファイルから実行形式のファイルを作成する**.
2. nassign **コマンドで** assign **されているモデルファイルが対象となる**.

ソースファイル nmake.c

NTIME

NCS

機能) **時間情報を設定する**.

形式) **ntime(last, cal, str {, interval})**

パラメータ)

1. last : **計算時間**
2. cal : **計算刻み**
3. str : **データをストアする間隔**
4. interval : **データをファイルに書き込む間隔**

解説)

1. **一度のデータファイル書き込みで**, str **で設定した値によって決まる回数分のデータが書き込まれる**.
2. cal≤str<last.
3. **書込み刻みとは計算結果を出力バッファに書込む刻み時間である**. itv > last **の時** last **時に書込まれる**.
4. **書込み刻みを小さく取りすぎると**, **シミュレーション時間が長く必要になるので適切な値を選ぶこと**.
5. **このコマンド実行時には**, **npp コマンドまたは** nmake **コマンドの実行によって**, **シミュレーション条件ファイル群が生成されている必要がある**.

ソースファイル ntime.c

NSTIM

NCS

機能) **外部入力(刺激)を設定する.**

形式)

```
nstim( mdl, com, type, p1 { , p2, p3, p4, p5 } )
```

パラメータ)

1. mdl : **モジュール名**
2. com : **コンポーネント番号**
3. type : **入力波形の指定**
   - P **パルス関数**
   - R **ランプ関数**
   - B **バッファ入力**
4. p1∼p5 : **パラメータ列**

入力波形	p1	p2	p3	p4	p5
パルス関数(P)	入力開始時刻	入力初期値	高さ	時間幅	周期
ランプ関数(R)	入力開始時刻	入力初期値	傾き	—	—
バッファ入力(B)	バッファ名	{配列番号}	{ 時間情報}	—	—

解説)

1. **ファイル及びバッファ入力においては**, Euler **法**, **自動刻み幅積分法**, RKG **法を選択した場合いずれにおいても**, **データにない点は補間して用いられる**. **なお**, **補間には** 4 **次のラグランジュ補間を使用している**.
2. NTIME **で指定する** last/cal **は**, **バッファ入力のデータ点数を越えてはならない**.
3. **バッファ入力の「配列番号」とは**, **入力バッファに** series **型の** 1 **次元配列を用いた場合**, **何番目の** series **データを使用するのかを指定する**. **デフォルトは** 0.
4. **バッファ入力の「時間情報」には**, **全部の入力系列の時刻を格納したバッファ名を指定する**. ntime **コマンドで指定された計算刻みで計算されるが**, **ここで指定されたバッファを調べ**, **計算時刻のデータがあるのであれば**, **補間せず計算を行う** (**通常は** Euler **法だと** $h/2$ **の点が**, RKG **法だと** $h/4, h/2, 3h/4$ **の点が補間されている** [h:ntime **で指定した計算刻み**]).
5. **このコマンド実行時には**, **npp コマンドまたは nmake コマンドの実行によって**, **シミュレーション条件ファイル群が生成されている必要がある**.

ソースファイル nstim.c

NOUT

機能) **出力情報を設定する**.

形式) **nout(buff, { , num } mdl, com, type { , val })**

パラメータ)

1. buff : **出力値を格納するバッファ番号**
2. num : **出力値を格納する配列番号**
3. mdl : **モジュール名**
4. com : **コンポーネント番号**
5. type : **出力情報**
   - 1 **出力値**
   - 2 **入力値**
   - 3 **内部変数値**
6. val : type=3 **を指定したとき**, **内部変数名を指定する**.

解説)

1. **引数に指定するバッファ名は**, **事前に** series **宣言されたものを使用する必要がある**.
2. num **を指定した場合**, series **変数の配列の** num **番目に出力される**. **例えば**, buff **に** "out", num **に** "3" **を指定した場合**, series **変数** "out[3]" **に結果が出力される**.
3. **このコマンド実行時には**, **npp コマンドまたは nmake コマンドの実行によって**, **シミュレーション条件ファイル群が生成されている必要がある**.

ソースファイル nout.c

NPARA

NCS

機能) **パラメータを変更する**.

形式) **npara(mdl, var, num)**

パラメータ)

1. mdl : **モジュール名**
2. var : **パラメータ名**
3. num : **パラメータの設定値**

解説)

1. NCS **言語記述においてパラメーター宣言された内部変数のみに有効である**.
2. **このコマンド実行時には**, **npp コマンドまたは nmake コマンドの実行によって**, **シミュレーション条件ファイル群が生成されている必要がある**.

ソースファイル npara.c

NINTEG

機能) **積分方式の指定および最大セル数を変更する**.

形式) **ninteg(type [, mcell, relerr])**

パラメータ) 1. type : **積分方式の指定**

F **自動刻み積分法**

R **ルンゲ-クッタ-ギル法**

E **オイラー法**

2. mcell : **最大セル数**

3. relerr : **数値積分の相対許容誤差**

解説) 1. **最大セル数とは**, cell, synapse, gap **文で定義した配列の最大値**.

2. **このコマンドを指定しなかった場合**, **自動刻み積分法で計算される**.

3. **このコマンド実行時には**, **npp コマンドまたは nmake コマンドの実行によって**, **シミュレーション条件ファイル群が生成されている必要がある**.

ソースファイル ninteg.c

NDELAY

機能) **ディレイ情報を設定する**.

形式) **ndelay(mdl, var, dt [, init])**

パラメータ)

1. mdl : **モジュール名**
2. var : **内部変数名**
3. dt : **ディレイ時間**
4. init : **出力の初期値** (**省略時は**, **モデル記述で指定した値になる**.)

   AUTO **時刻** 0 **の出力値を初期値とする**

解説)

1. **ディレイはモジュール入力のみ設定可能**.
2. **ディレイは** QUEUE **を使用することによって実現している**. **時刻** 0 **から** dt **までの間は**, init **の値が出力される**.
3. NETWORK() **中で**, 2 **度以上呼び出しのあるモジュールでの動作は補償しない**.
4. **このコマンド実行時には**, **npp コマンドまたは nmake コマンドの実行によって**, **シミュレーション条件ファイル群が生成されている必要がある**.

ソースファイル ndelay.c

NLIST

機能) **パラメータリストを表示する**.

形式) **nlist(mdl)**

パラメータ)

1. mdl : **モジュール名**
2. ALL : **全てのパラメータを表示する**

解説)

1. mdl **に** ALL **を指定すると**, **全てのパラメータを表示する**.
2. **端末の表示行数の限界を越えた場合**, **環境変数** PAGER **で指定されたページャを起動して表示する**. PAGER **が設定されていない場合**, "more" **が使用される**.
3. **このコマンド実行時には**, **npp コマンドまたは nmake コマンドの実行によって**, **シミュレーション条件ファイル群が生成されている必要がある**.

ソースファイル nlist.c

NSCLIST

機能) **シミュレーション条件を表示する**.

形式) **nsclist(type)**

パラメータ) 1. type : **表示内容**

- S **刺激条件の表示**
- O **出力条件の表示**
- T **時間情報の表示**
- D **ディレイ情報の表示**

解説) 1. **端末の表示行数の限界を越えた場合**, **環境変数**PAGER **で指定されたページャを起動して表示する**. PAGER **が設定されていない場合**, "more" **が使用される**.

2. **このコマンド実行時には**, **npp コマンドまたは nmake コマンドの実行によって**, **シミュレーション条件ファイル群が生成されている必要がある**.

ソースファイル nsclst.c

NERASE

NCS

機能) **シミュレーション条件を削除する**.

形式) **nerase(type, var { , dim})**

パラメータ)

1. type : **削除内容**
   - S **刺激情報**
   - O **出力情報**
   - D **ディレイ情報**
2. var : **変数名**
3. dim : **配列番号**

解説)

1. no **に** ALL **を指定すれば**, **全ての条件を削除する**.
2. **出力に配列を使用している場合**, dim **に配列番号も指定する**.
3. **このコマンド実行時には**, **npp コマンドまたは** nmake **コマンドの実行によって**, **シミュレーション条件ファイル群が生成されている必要がある**.

ソースファイル nerase.c

NGETP

機能) **パラメータを取得する**.

形式) **ngetp(mdl, var)**

パラメータ)
1. mdl : **モジュール名**
2. var : **パラメータ名**

解説)
1. **パラメーター値を返す** (Scalar).
2. NCS **言語記述においてパラメーター宣言された内部変数のみに有効である**.
3. **このコマンド実行時には**, **npp コマンドまたは nmake コマンドの実行によって**, **シミュレーション条件ファイル群が生成されている必要がある**.

ソースファイル ngetp.c

機能) **計算を実行する**.

形式) ncal()

パラメータ) **なし**

解説) 1. **このコマンド実行時には**, npp **コマンドまたは** nmake **コマンドの実行によって**, **シミュレーション条件ファイル群が生成されている必要がある**.

ソースファイル ncal.c

機能) **エディタを起動する**.

形式) ne()

パラメータ) **なし**

解説)

1. assign **されたモデルファイルを編集の対象として**, **エディタを起動する**.
2. **環境変数** EDITOR **が設定されている場合それを起動し**, **設定されていない場合は** vi **エディタが起動される**.

ソースファイル ne.c

NCHGBUFF

NCS

機能) nout **で指定できる変数の最大個数の変更**

形式) **nchgbuff(number)**

パラメータ) 1. number : **変数の個数の最大数**

解説) 1. **モデル記述内で** exinput, output, observable **文によって指定した変数を**, nout **コマンドで指定できる個数はある値** (**デフォルトは** 255) **に制限されている**. **大規模モデルなどで**, **多くの変数を取り出すようにしたいときなどに使用する**.

2. **このコマンド実行時には**, **npp コマンドまたは** nmake **コマンドの実行によって**, **シミュレーション条件ファイル群が生成されている必要がある**.

ソースファイル nchgbuff.c

非線形パラメータ推定システム

NPE

CDATA

機能) **データファイルを指定する．**

形式) cdata(file_name)

パラメータ) 1. file_name : **データファイル名**

解説)

- **モデル出力の数だけ，連続したレコードに実験データが格納されていなければならない．**
- **データファイルの格納形式は，モデルの出力** 1 **個に対して** 1 **レコードを対応させる．**

使用例) 1-**出力モデルのデータとしてサイン波，**1000 **点を与える．**

```
% series x,y[1];
% x = (0~(1000-1))/(1000-1);
% y[0] = sin( 2*PI*x );
% $"sin1.dat" = trans(y);
% index($"sin1.dat");
[0]:%            1        1000
```

参照) CWEIGHT

ソースファイル setdata.c common1.c common2.c

CDEL

機能) **設定されている推定条件を削除する.**

形式) cdel(type)

パラメータ) 1. type : **削除する推定条件の種類**

METHOD LSEARCH MODEL INIT SCALE TERM NUMBER POINT
DATA WEIGHT RESULT HISTORY INTEG SCOPE NORM DISP

ソースファイル delblock.c common1.c common2.c

CDISP

機能) **表示する評価関数値の計算方法を指定する．**

形式) cdisp(type)

パラメータ) 1. type : **表示する評価関数値の計算方法**
- **0 重みなし**
- **1 重みつき**

解説)

- **与えるデータを** $t_{i,j}$**，モデルからの出力を** $o_{i,j}$**，重みを** $w_{i,j}$ **とすると，評価関数値** $Value$**(2-ノルム) は**

$$Value = \sum_{i,j} w_{i,j}(t_{i,j} - o_{i,j})^2$$

である．評価関数値を表示する際に $Value$ **を使うか，重みを全て** 1 **として計算した** $Value'$ **を用いるかを指定する．**

- **表示する評価関数値の計算方法であって，停止基準には関係しない．ただし，評価関数値の履歴に格納される値は，このコマンドで指定した方法で計算される．**
- **省略した場合は** 1**(重みつき) が設定される．**

参照) CHISTORY CNORM

ソースファイル setdisp.c common1.c common2.c

CHISTORY

機能) **パラメータおよび評価関数値の履歴を格納する方法を指定する．**

形式) **chistory(file_name, interval)**

パラメータ)

1. file_name : **格納するファイル名**
2. interval : **格納間隔**(interval = 0 **は** interval = 1 **と同じ**)

解説)

1. **推定対象のパラメータ値，評価関数値の履歴を，指定したファイルに指定した格納間隔でアペンドする．ただし，停止条件を満たした場合は，格納間隔によらずアペンドする．**
2. **評価関数値の履歴は指定したファイルの最後のレコードに格納される．**
3. **格納される評価関数値は**CDISP **コマンドで指定した方法で計算される．また，制約項の値は加えられない．**

参照) CDISP

ソースファイル sethistory.c common1.c common2.c

CINIT

機能) パラメータの初期値等の情報を指定する．

形式) cinit(ninit, inum, val, flag, name, span)

パラメータ)

1. ninit : **推定するパラメータの数**
2. inum : **パラメータ番号**
3. val : **パラメータの初期値**
4. flag : **パラメータの固定/可変の指定**
   FIX　**パラメータ値を初期値で固定** (**推定しない**)
   VAR　**パラメータ値を可変** (**推定する**)
5. name : **パラメータの名前**
6. span : **差分間隔**

解説)

1. **パラメータ数** ninit **の値を変更する場合は，それ以前に設定されていたパラメータの情報は全て失われる．**
2. ninit **個あるパラメータのうち，**1 **から数えて** inum **番目のパラメータの情報を指定する．**
3. **パラメータの名前は，使用するモデルによって次のように指定する内容が異なる．**
   ・C 言語で記述したモデルの場合

   推定実行時に表示される名前を指定する．(**モデルファイルと対応させる必要はない**)

   ・NCS モデルの場合

   各種情報を設定したいパラメータを，NCS **モデル記述内のモジュール名とパラメータ名を**@**で接続した文字列で指定する．**
   (**例**) BHL **モジュールの** Tau_L **というパラメータを推定する場合**
   name = Tau_L@BHL

   また，パラメータの名前は最大で 25 **文字である．**
4. **各パラメータに対する偏微分は中心差分で行うため** span **は十分に小さい値でなければならない．**

ソースファイル　setinit.c　common1.c　common2.c

CINTEG

機能) NCS **モデルを用いる場合の積分方式を指定する.**

形式) **cinteg(integmethod)**

パラメータ) 1. integmethod : **積分方式**

- R **ルンゲクッタ法**(4 **次**)
- E **オイラー法**
- A ADAMS**法**
- F ADAPTIVE**法**

ソースファイル setxinteg.c common1.c common2.c

CLIST

機能) **設定されている推定条件を表示する．**

形式) **clist([elmnttype])**

パラメータ) 1. elmnttype : **表示する推定条件の種類**

METHOD LSEARCH MODEL INIT SCALE TERM NUMBER POINT
DATA WEIGHT RESULT HISTORY INTEG SCOPE DISPLAY NORM

解説)

- **表示する推定条件の指定がなければ全て表示する．**
- **推定条件の指定には大文字，小文字どちらでも構わない．また，頭の**3**文字だけでも良い．**

使用例) LSEARCH **の内容を表示させる**.

```
% clist("lsearch");

lsearch : golden          0.0001

% clist("lse");

lsearch : golden          0.0001
```

ソースファイル npelist.c common1.c common2.c

CLOAD

機能) **設定ファイルの内容(推定条件)をコモン領域へ読み込む.**

形式) **cload(file_name)**

パラメータ) 1. file_name : **設定ファイル名**

解説)

1. **追加,変更されたコモン領域の内容を,設定ファイルに格納**(CSTORE)**せずに新たに読み込もうとすると,警告を発して確認を求めるフェイルセーフ機能を有している.**
2. **各種コマンドの組合せのみで必要な条件を設定することが可能であるが,毎回変更しない条件を設定ファイルに書き込んでおくと便利である.**
3. **設定ファイルをはじめて作る時は各コマンドで条件を設定し,**CSTORE**コマンドを利用するとよい.**

参照) CSTORE NPEINIT

ソースファイル NPEload.c common1.c common2.c

CLOGIC

機能) **停止基準の論理式を指定する．**

形式) clogig(logic)

パラメータ) 1. logic : **停止基準の論理式**

解説)
- CTERM **コマンドで設定された停止基準を論理和** (|)**，論理積** (&) **および括弧を用いて組み合わせる．**
- **CTERM コマンドよりも先に実行する必要がある．**

使用例)

```
% clogic("0|1" );
% cterm(0, 1000 );
% cterm(1, 1.0e-04);
% clist("term");
term    : 0|1
          TERM 0 = 1000
          TERM 1 = 0.0001
```

参照 CTERM

ソースファイル setlogic.c common1.c common2.c

CLSEARCH

機能) **直線探索法を指定する．**

形式) **clsearch(linear_method, bracket_init)**

パラメータ) 1. linear_method : **直線探索法**

2. bracket_init : **囲い込みの幅の初期値**

解説) 1. **利用可能な直線探索法**
golden **黄金分割法**
cubic **三次補間法**

2. **ほとんどの最適化手法は下位のルーチンとして直線上で評価関数値を最小にするパラメータを求めるルーチンを必要とする．黄金分割法はパラメータに関する偏導関数を計算する必要がないが，三次補間法は計算しなければならない．前者は滑らかでない関数でも計算でき，後者は滑らかならば少ないステップ数で計算が終了するという特徴を持つ．**

3. **囲い込みの幅(最大探索幅)は自動的に拡大する．**

使用例)
```
% clsearch("golden",1.0e-03);
% clist("lsearch");

lsearch : golden          0.001
```

参照) CMETHOD

ソースファイル setlsearch.c common1.c common2.c

CMETHOD

機能) **最適化アルゴリズムを指定する．**

形式) **cmethod(method)**

パラメータ) 1. method : **最適化法**

解説)

- **利用可能な最適化法**

SIMPLEX	Simplex **法**
BFGS	BFGS(Broyden-Fletcher-Goldfarb-Shanno) **法**
DFP	DFP(Davideon-Fletcher-Powell) **法**
SSVM	SSVM(Self-Scaling Variables Metric) **法**
CONJFR	**共役勾配法**(Fletcher-Reeves **の更新式**)
CONJPRP	**共役勾配法**(Polak-Ribière-Polyak **の更新式**)

- Simplex**法以外では直線探索法も指定しなければならない．**
- **最適化法に一般的にベストだと言える方法は発見されていないため，いくつかの方法を試して見ることが常識となっている．経験的には**BFGS**法が優れていると言われている．**

使用例)

```
% cmethod("bfgs");
% clist("method");

method  : bfgs
```

参照) CLSEARCH

ソースファイル setmethod.c common1.c common2.c

CMODEL

機能) **モデルを指定する．**

形式) **cmodel(mdl_type, file_name)**

パラメータ) 1. mdl_type : **モデルの形式**
USR C **言語で記述したモデル**
NCS NCS **言語で記述したモデル**
2. file_name : **モデルのオブジェクトファイル名**

解説) • **モデルの記述方法については，ユーザーズマニュアルを参照．**

ソースファイル setmodel.c common1.c common2.c

CNORM

機能) **評価関数のノルムの計算方法を指定する．**

形式) **cnorm(norm)**

パラメータ) 1. norm : **ノルムのタイプ**
- **0** **無限大ノルム**
- **1** 1-**ノルム**
- **2** 2-**ノルム**

解説)
- **与えるデータを** $t_{i,j}$**，モデル出力を** $o_{i,j}$**，重みデータを** $w_{i,j}$ **とすると，評価関数は各ノルム毎に以下のように定義される．**

  0-ノルム

$$Value = \max_{i,j}\Big\{w_{i,j}|t_{i,j} - o_{i,j}|\Big\}$$

  1-ノルム

$$Value = \sum_{i,j} w_{i,j}|t_{i,j} - o_{i,j}|$$

  2-ノルム

$$Value = \sum_{i,j} w_{i,j}(t_{i,j} - o_{i,j})^2$$

- **タイプの指定がなければ，2-ノルムが設定される．**

参照) CDISP

ソースファイル setnorm.c common1.c common2.c

CNUMBER

機能) **モデル出力の数を指定する.**

形式) **cnumber(numwave)**

パラメータ) 1. numwave : **モデル出力の数**

解説)

- **実験データや重みデータはこのコマンドで指定しただけのレコード数を持たなければならない.**

使用例) 1-**出力モデルの例.**

```
% cnumber(1);
% clist("number");

number  :    1
```

データファイルから設定する.

```
% cdata("sin1.dat" );
% cnumber(length( $"sin1.dat" ) );
```

参照) CDATA CWEIGHT

ソースファイル setnumber.c common1.c common2.c

CPENALTY

機能) **ペナルティ関数を指定する．**

形式) **cpenalty(file_name)**

パラメータ) 1. file_name : **制約を記述したオブジェクトファイル名**

解説) 1. NPE **で利用可能な制約の課し方はペナルティ法である．ペナルティ法はその名の通り，パラメータが制約条件を満たしていなければ評価関数値に大きな値 (ペナルティ) を加える．これによって，制約条件を満たすように推定されることが期待される．**

2. **制約条件はオブジェクトファイルの形式になっていなければならない．**

ソースファイル setpenalty.c

CPOINT

機能) **データ点数を指定する.**

形式) **cpoint(datapoint)**

パラメータ) 1. datapoint : **データ点数**

解説)

- **実験データや重みデータはこのコマンドで指定しただけのデータ点数を持たなければならない.**

使用例) **ファイルに格納されているデータを指定する.**

```
% cdata("sin1.dat" );
% cpoint( length( $"sin1.dat":[0] ) );
```

参照) CNUMBER CDATA CWEIGHT

ソースファイル setpoint.c common1.c common2.c

CRESULT

機能) **モデル出力の履歴を格納するファイル名を指定する．**

形式) **cresult(file_name)**

パラメータ) 1. file_name : **格納するファイル名**

解説) 1. **推定が完了したパラメータを用いてモデル出力を計算し，指定されたファイルに格納する．**

ソースファイル setresult.c common1.c common2.c

CSCALE

機能) **スケーリング法を指定する．**

形式) `cscale( scalingmethod )`

パラメータ) 1. scalingmethod : **スケーリング法**

- **0** **スケーリングなし**
- **1** **桁スケーリング有り** (**初回のみ**)
- **2** **桁スケーリング有り** (**毎ステップ**)
- **3** re-scaling

解説)

- **スケーリングを指定することで，推定が難しい状況** (ill-condition) **が改善されることが期待される．**
- **指定されなかった場合は，0**(**スケーリングなし**) **が設定される．**

参照) **落合 慶広，榊原 学，臼井 支朗**: "**非線形最適化法に関する研究**"，
豊橋技術科学大学大学院 修士論文 (1989)**．**

ソースファイル setscale.c common1.c common2.c

CSTORE

機能)　**設定されている推定条件を指定したファイルに格納する.**

形式)　**cstore(file_name)**

パラメータ)　1. file_name : **推定条件を格納するファイル名**

解説)　• **設定ファイルをはじめて作成する時に利用すると便利である.**

参照)　CLOAD　NPEINIT

ソースファイル　NPEstore.c　common1.c　common2.c

CTERM

機能) **停止基準を設定する.**

形式) **cterm(termnum, termvalue)**

パラメータ) 1. termnum : **停止基準の番号**
   0 : **最大反復回数** ≧ termvalue
   1 : **評価関数値** ≦ termvalue
   9 : simplex**の評価関数値の標準偏差** ≦ termvalue
2. termvalue : **停止基準の値**

解説)
- **このコマンドで指定した停止基準値は,** CLOGIC **コマンドで組み合わされ, 停止基準として用いられる.**
- **必ず** CLOGIC **コマンドの後に実行する.**

参照) CLOGIC

ソースファイル setterm.c common1.c common2.c

CWEIGHT

機能) **評価関数に重みづけをする重みデータファイルを指定する.**

形式) **cweight(file_name)**

パラメータ) 1. file_name : **重みデータファイル名**

解説) 1. **モデル出力の数だけ, 連続したレコードに重みデータが格納されていなければならない.**

2. **モデルの出力**1**個に対して**1**レコードを対応させる.**

使用例) 1-**出力モデルの重みデータ**(1000 **点**, **すべて** 1)**を生成する**.

```
% series wgt[1];
% wgt[0] = (1~1000)*0 + 1;
% $"sin1.wgt" = trans(wgt);
% index($"sin1.wgt");
[0]:%            1         1000
```

参照) CDATA

ソースファイル setweight.c common1.c common2.c

NPE

機能) **パラメータ推定を実行する．**

形式) **npe(res, hist)**

パラメータ)

1. res : **推定したパラメータを用いて計算したモデルの出力**
2. hist : **パラメータ・評価関数値の履歴**

解説)

- **設定された推定条件に従って，パラメータ推定を行なう．**
- res **に格納される内容は** cresult **コマンドで設定したファイルに格納される内容と同じ．**
- hist **に格納される内容は** chistory **コマンドで設定したファイルに格納される内容と同じ．**

ソースファイル main.c common1.c common2.c

NPEINIT

機能) 推定条件を保持する領域を初期化する.

形式) `npeinit()`

パラメータ) なし

解説) 必ずNPEを利用する前に実行しなければならない.

ソースファイル NPEinit.c

データ変換モジュール

DCM

ABF2SATELLITE DCM

機能) AXON pClamp の ABF 形式のファイルを *SATELLITE* の ファイルオブジェクトに変換する.

書式) abf2satellite(filename.abf)

パラメータ) 1. filename : 変換する abf ファイル名

解説) ABF 形式のファイルを *SATELLITE* のファイルオブジェクト (filename.dat) と、ABF ファイルに含まれるさまざまな情報を含んだファイル (filename.cnf) に変換する.

使用例) abf ファイル 2000-07-01-00000.abf を *SATELLITE* のファイルオブジェクトに変換する. タ形式に変換する.

```
% abf2satellite("2000-07-01-00000.abf")
% ls
2000-07-01-00000.abf 2000-07-01-00000.cnf 2000-07-01-00000.dat
```

バグ) いくつか実装していない形式のファイルが存在します.

ソースファイル abf2satellite.c abf2satellite.h

ATF2SATELLITE

機能) AXON pClamp の ATF 形式のファイルを *SATELLITE* の ファイルオブジェクトに変換する.

書式) atf2satellite(filename.atf)

パラメータ) 1. filename : 変換する atf ファイル名

解説) ATF 形式のファイルを *SATELLITE* のファイルオブジェクト (filename.dat) に変換する.

使用例) atf ファイル 2000-07-01-00000.atf を *SATELLITE* のファイルオブジェクトに変換する. タ形式に変換する.

```
% atf2satellite("2000-07-01-00000.atf")
```

バグ) いくつか実装していない形式のファイルが存在します.

ソースファイル atf2satellite.c

BUF2AVS

DCM

機能) *SATELLITE*の series 変数 を AVS のフィールドデータ形式のファイルに変換する.

書式) buf2avs(filename , x)

パラメータ) 1. filename : 拡張子を省いた出力したいファイル名 (string)
2. x : オブジェクト (series)

解説) 出力されるファイルは 2 つ存在するため, 出力するファイル名には, 拡張子を含ませないでおく.

使用例) series オブジェクト x を AVS のフィールドデータ形式に変換する.

```
% buf2avs("FieldData",x)
```

ソースファイル buffer2avs.c

BUFFER2AVS

DCM

機能) *SATELLITE* **の** series **変数 を AVS のフィールドデータ形式のファイルに変換する**.

書式) buffer2avs(filename , x)

パラメータ) 1. filename : **拡張子を省いた出力したいファイル名** (string)
2. x : **オブジェクト** (series)

解説) **出力されるファイルは** 2 **つ存在するため**, **出力するファイル名には**, **拡張子を含ませないでおく**.

使用例) series **オブジェクト** x **を AVS のフィールドデータ形式に変換する**.

```
% buffer2avs("FieldData",x)
```

ソースファイル buffer2avs.c

BUF2TXT

DCM

機能) *SATELLITE*の series 変数 をテキスト形式のファイルに変換する.

書式) buf2txt(filename , format , x,y. . .)

パラメータ) 1. filename : テキストファイル名 (string)
2. format : フォーマット文字列
3. x,y,. . . : オブジェクト (series)

解説) フォーマット文字列の書式は, 整数及び実数を扱う部分のみ C 言語と同様である.

型	フォーマット文字列		
整数	%d	%nd	
実数	%f	%nf	%nmf
	%g	%ng	%nmg
	%e	%ne	%nme

使用例) series オブジェクト x , y をテキストファイル test.txt に出力する.

```
% buf2txt("test.txt","%f %e\n",x,y)
```

ソースファイル buffer2text.c

BUFFER2TEXT

DCM

機能) *SATELLITE* **の** series **変数 をテキスト形式のファイルに変換する**.

書式) buffer2text(filename , format , x,y...)

パラメータ)
1. filename : **テキストファイル名** (string)
2. format : **フォーマット文字列**
3. x,y,... : **オブジェクト** (series)

解説) **フォーマット文字列の書式は**, **整数及び実数を扱う部分のみ** C **言語と同様である**.

型	**フォーマット文字列**
整数	%d %nd
実数	%f %nf %nmf %g %ng %nmg %e %ne %nme

使用例) series **オブジェクト** x , y **をテキストファイル** test.txt **に出力する**.

```
% buffer2text("test.txt","%f %e\n",x,y)
```

ソースファイル buffer2text.c

BUF2MAT

DCM

機能) *SATELLITE*の series 変数 を MATLAB の Version 4 の MAT-ファイル形式に変換する.

書式) buf2mat(filename , variable , x)

パラメータ)
1. filename : 出力したいファイル名 (string)
2. variable : MATLAB において実際に用いられる変数名 (string)
3. x : オブジェクト (series)

解説) オブジェクトが 2 次元の場合, 値を行列とみなしてファイルを変換する.

使用例) series オブジェクト x を MATLAB の MAT-ファイル形式に変換する.

```
% buf2mat("Matlab.mat","Satellite",x)
```

ソースファイル buffer2matlab.c

BUFFER2MATLAB DCM

機能) *SATELLITE* **の** series **変数 を** MATLAB **の** Version 4 **の** MAT-**ファイル形式に変換する**.

書式) buffer2matlab(filename , variable , x)

パラメータ)
1. filename : **出力したいファイル名** (string)
2. variable : **MATLAB において実際に用いられる変数名** (string)
3. x : **オブジェクト** (series)

解説) **オブジェクトが** 2 **次元の場合**, **値を行列とみなしてファイルを変換する**.

使用例) series **オブジェクト** x **を** MATLAB **の** MAT-**ファイル形式に変換する**.

```
% buffer2matlab("Matlab.mat","Satellite",x)
```

ソースファイル buffer2matlab.c

BUF2MATH

DCM

機能) *SATELLITE*の series 変数 を Mathematica で使用可能なリスト形式に変換する.

書式) buf2math(filename , x)

パラメータ) 1. filename : 出力したいファイル名 (string)
2. x : オブジェクト (series)

解説) オブジェクトが2次元の場合, 値を行列とみなしてファイルを変換する.

使用例) series オブジェクト x を Mathematica のリスト形式に変換する.

```
% buf2math("Mathematica.dat",x)
```

ソースファイル buffer2mathematica.c

BUFFER2MATHEMATICA DCM

機能) *SATELLITE*の series 変数 を Mathematica で使用可能なリスト形式に変換する.

書式) buffer2mathematica(filename , x)

パラメータ) 1. filename : 出力したいファイル名 (string)
2. x : オブジェクト (series)

解説) オブジェクトが2次元の場合, 値を行列とみなしてファイルを変換する.

使用例) series オブジェクト x を Mathematica のリスト形式に変換する.

```
% buffer2mathematica("Mathematica.dat",x)
```

ソースファイル buffer2mathematica.c

GEN2BUF

DCM

機能) **Genesis の** disk_out **オブジェクト よって出力されたファイルを** *SATELLITE***の** series **変数 に変換する**.

書式) x = gen2buf(filename [, start , step])

パラメータ)

1. x : **出力オブジェクト** (series)
2. filename : **入力したいファイル名** (string)
3. start : **Genesis でシミュレーションした際の開始時間** (series)
4. step : **Genesis でシミュレーションした際の時間刻み幅** (series)

解説) start **および** step **は**, **任意に付加できる引数となっているが**, **この引数を使用する際は**, **あらかじめ**, series **で宣言しておかなければならない**.

使用例) **Genesis によって出力されたファイル** Genesis.dat **を** *SATELLITE***の** series **変数** x **に変換する**.

```
% series start,step
% x = gen2buf("Genesis.dat",start,step)
```

ソースファイル genesis2buffer.c

GENESIS2BUFFER

機能) **Genesis の** disk_out **オブジェクト よって出力されたファイルを** *SATELLITE***の** series **変数 に変換する**.

書式) x = genesis2buffer(filename [, start , step])

パラメータ)

1. x : **出力オブジェクト** (series)
2. filename : **入力したいファイル名** (string)
3. start : **Genesis でシミュレーションした際の開始時間** (series)
4. step : **Genesis でシミュレーションした際の時間刻み幅** (series)

解説) start **および** step **は**, **任意に付加できる引数となっているが**, **この引数を使用する際は**, **あらかじめ**, series **で宣言しておかなければならない**.

使用例) **Genesis によって出力されたファイル** Genesis.dat **を** *SATELLITE***の** series **変数** x **に変換する**.

```
% series start,step
% x = genesis2buffer("Genesis.dat",start,step)
```

ソースファイル genesis2buffer.c

MAT2SAT

DCM

機能) **MATLAB によって出力された** Version 4 **の** MAT-**ファイル形式のファイルを** *SATELLITE***の ファイルオブジェクトに変換する**.

書式) mat2sat(filename)

パラメータ) 1. filename : **入力したいファイル名** (string)

解説) **このコマンドにより出力されるファイルは**, **実際に**MATLAB **で使用されていた 変数名** ?? **を含む** mat_??.dat **という名前になる**. **また**, MAT-**ファイル が複素数のデータである場合**, **出力されるファイル名は**, **実数部が** mat_??_r.dat , **虚数部が** mat_??_i.dat **となる**.

使用例) **MATLAB で出力されたファイル** Matlab.mat **を** *SATELLITE***のファイルオブジェクト形式に変換する**.

```
% mat2sat("Matlab.mat")
```

ソースファイル matlab2satellite.c

MATLAB2SATELLITE DCM

機能) **MATLAB によって出力された** Version 4 **の** MAT-**ファイル形式のファイルを** *SATELLITE***の ファイルオブジェクトに変換する**.

書式) matlab2satellite(filename)

パラメータ) 1. filename : **入力したいファイル名** (string)

解説) **このコマンドにより出力されるファイルは**, **実際に**MATLAB **で使用されていた変数名** ?? **を含む** mat_??.dat **という名前になる**. **また**, MAT-**ファイル が複素数のデータである場合**, **出力されるファイル名は**, **実数部が** mat_??_r.dat , **虚数部が** mat_??_i.dat **となる**.

使用例) **MATLAB で出力されたファイル** Matlab.mat **を** *SATELLITE***のファイルオブジェクト形式に変換する**.

```
% matlab2satellite("Matlab.mat")
```

ソースファイル matlab2satellite.c

NRN2SAT

機能) **Neuron によって出力されたファイルを *SATELLITE*の ファイルオブジェクトに変換する**.

書式) nrn2sat(filename)

パラメータ) 1. filename : **入力したいファイル名** (string)

解説) **このコマンドにより出力されるファイルは**, **実際に Neuron で使用されていたラベルを用いたファイル名になる**.

使用例) **Neuron で出力されたファイル** Neuron.dat **を *SATELLITE*のファイルオブジェクト形式に変換する**.

```
% nrn2sat("Neuron.dat")
```

ソースファイル neuron2satllite.c

NEURON2SATELLITE DCM

機能) **Neuron によって出力されたファイルを *SATELLITE*の ファイルオブジェクトに変換する**.

書式) neuron2satellite(filename)

パラメータ) 1. filename : **入力したいファイル名** (string)

解説) **このコマンドにより出力されるファイルは**, **実際に Neuron で使用されていたラベルを用いたファイル名になる**.

使用例) **Neuron で出力されたファイル** Neuron.dat **を *SATELLITE*のファイルオブジェクト形式に変換する**.

```
% neuron2satllite("Neuron.dat")
```

ソースファイル neuron2satellite.c

TXT2BUF

DCM

機能) **テキスト形式のファイルを** *SATELLITE* **の** series **型変数に変換する**.

書式) x = txt2buf(filename)

パラメータ) 1. x : **出力オブジェクト** (series)

2. filename : **入力したいファイル名** (string)

解説) **このコマンドにより出力されたオブジェクトはテキストファイルが** n × m **の場合**, **時系列を** 2 **次元の行列にみたてた形で表現される**.

使用例) **テキストファイル** text.txt **を** *SATELLITE* **の** series **オブジェクト** x **に変換する**.

```
% x = txt2buf("text.txt")
```

ソースファイル text2buffer.c

TEXT2BUFFER

DCM

機能) **テキスト形式のファイルを** *SATELLITE* **の** series **型変数に変換する**.

書式) x = text2buffer(filename)

パラメータ) 1. x : **出力オブジェクト** (series)

2. filename : **入力したいファイル名** (string)

解説) **このコマンドにより出力されたオブジェクトはテキストファイルが** n × m **の場合**, **時系列を** 2 **次元の行列にみたてた形で表現される**.

使用例) **テキストファイル** text.txt **を** *SATELLITE* **の** series **オブジェクト** x **に変換する**.

```
% x = text2buffer("text.txt")
```

ソースファイル text2buffer.c

コマンド索引

ABF2SATELLITE (DCM) 249
ABS (Mathematic Library) 40
ACOS (Mathematic Library) 41
ACTLOAD (BPS) 178
AKIMA (ISPP) 100
ALIAS (SHELL) 2
ARAND (ISPP) 101
ARGEN (ISPP) 102
ASIN (Mathematic Library) 42
ATAN (Mathematic Library) 43
ATAN2 (Mathematic Library) 44
ATF2SATELLITE (DCM) 250
AVERAGE (ISPP) 104
AXIS (GPM) 143
BM (SYSTEM) 78
BPBTW (ISPP) 105
BPLOAD (BPS) 179
BPSAVE (BPS) 180
BREAK (SHELL) 3
BUF2AVS (DCM) 251
BUF2MAT (DCM) 255
BUF2MATH (DCM) 257
BUF2TXT (DCM) 253
BUFFER2AVS (DCM) 252
BUFFER2MATHEMATICA (DCM) 258
BUFFER2MATLAB (DCM) 256
BUFFER2TEXT (DCM) 254
BURG (ISPP) 106
CDATA (NPE) 225
CDEL (NPE) 226
CDISP (NPE) 227
CEP (ISPP) 108
CHISTORY (NPE) 228
CHWIN (GPM) 145
CINIT (NPE) 229
CINTEG (NPE) 230
CLIST (NPE) 231
CLOAD (NPE) 232
CLOGIC (NPE) 233
CLSEARCH (NPE) 234
CMETHOD (NPE) 235
CMODEL (NPE) 236
CNORM (NPE) 237
CNUMBER (NPE) 238
COLOR (GPM) 146
CONST (SHELL) 4
CONT (GPM) 147
CONTINUE (SHELL) 5
COR (BPS) 181
COS (Mathematic Library) 45
CPENALTY (NPE) 239
CPOINT (NPE) 240
CRESULT (NPE) 241
CSCALE (NPE) 242
CSTORE (NPE) 243
CTERM (NPE) 244
CUT (SYSTEM) 79
CWEIGHT (NPE) 245
DCCUT (ISPP) 109
DEFINE (SHELL) 6
DET (ISPP) 110
DISP (BPS) 182
DO (SHELL) 7
DRAW (GPM) 148
DUMP (SL_UTIL) 60
EGRAPH (GPM) 149
EIGEN (ISPP) 111
ERRFUNC (BPS) 183
ERRLOAD (BPS) 184
ERROR (BPS) 185
EVAL (SHELL) 8
EXIST (SL_UTIL) 59
EXIT (SHELL) 9
EXP (Mathematic Library) 46
EXP2 (Mathematic Library) 47
EXTERNAL (SHELL) 10
EXTRACT (SL_UTIL) 61
FACTOR (GPM) 151
FFTC (ISPP) 112
FFTN (ISPP) 113
FILEINDEX (SL_UTIL) 62
FILL (SYSTEM) 80
FIND (SYSTEM) 81
FIR (ISPP) 114
FIRMAKE (ISPP) 115
FONT (GPM) 152
FOR (SHELL) 11
FRAME (GPM) 153
FUNC (SHELL) 12
FUNCTION (BPS) 186
GARCO (SHELL) 13
GAUSS2 (ISPP) 116
GEN2BUF (DCM) 259
GENESIS2BUFFER (DCM) 260
GET (SYSTEM) 82
GINIT (GPM) 154
GOODBYE (SHELL) 14
GPM2PS (GPM) 155

GRAPH (GPM) 156
GSOLM (GPM) 158
GSTAT (GPM) 160
HEADER (SYSTEM) 83
HELP (SYSTEM) 84
HGET (SL_UTIL) 63
HIL (ISPP) 117
HISTORY (SHELL) 15
HPBTW (ISPP) 118
ICEP (ISPP) 119
IF (SHELL) 16
IIR (ISPP) 120
IIRCOEF (ISPP) 121
INDEX (SHELL) 17
INDEXSIZE (SL_UTIL) 64
INLINE (SHELL) 18
INT (Mathematic Library) 48
INTEG (ISPP) 122
INTERP (ISPP) 123
INV (ISPP) 124
ISDEF (SHELL) 19
LABEL (GPM) 161
LALGO (BPS) 187
LAYER (BPS) 189
LEARN (BPS) 190
LEND (BPS) 191
LENGTH (SHELL) 20
LEVIN (ISPP) 125
LINE (GPM) 162
LOG (Mathematic Library) 49
LOG10 (Mathematic Library) 51
LOG2 (Mathematic Library) 50
LPBTW (ISPP) 126
LTYPE (GPM) 163
LWIDTH (GPM) 164
MABI (SYSTEM) 85
MAP (GPM) 165
MAT2SAT (DCM) 261
MATLAB2SATELLITE (DCM) 262
MAX (SYSTEM) 87
MAXPOS (SYSTEM) 88
MERGE (SYSTEM) 89
MIN (SYSTEM) 90
MINPOS (SYSTEM) 91
MNRAND (ISPP) 127
MOD (Mathematic Library) 52
MODULE (SHELL) 21
MUL (ISPP) 128
NASSIGN (NCS) 207
NCAL (NCS) 221
NCHGBUFF (NCS) 223
NDELAY (NCS) 216
NE (NCS) 222
NERASE (NCS) 219
NEURON2SATELLITE (DCM) 264
NEWPAGE (GPM) 167
NGETP (NCS) 220
NINTEG (NCS) 215
NLINK (NCS) 209
NLIST (NCS) 217
NMAKE (NCS) 210
NMEQ (ISPP) 129
NORM (ISPP) 130
NOUT (NCS) 213
NPARA (NCS) 214
NPE (NPE) 246
NPEINIT (NPE) 247
NPP (NCS) 208
NRAND (ISPP) 131
NRN2SAT (DCM) 263
NSCLIST (NCS) 218
NSTIM (NCS) 212
NTIME (NCS) 211
ORIGIN (GPM) 168
PDISP (BPS) 192
PHASE (ISPP) 132
PINIT (BPS) 193
POLE (ISPP) 133
POW (Mathematic Library) 53
POWER (ISPP) 134
PREV (GPM) 169
PRINT (SHELL) 22
PRINTF (SHELL) 23
PROC (SHELL) 24
PUT (SYSTEM) 92
PUTS (SL_UTIL) 65
RANK (ISPP) 135
READ (SHELL) 25
REC (BPS) 195
REFORM (SYSTEM) 93
RETURN (SHELL) 26
REVERSE (SYSTEM) 94
REVTIME (SL_UTIL) 66
ROTATE (SYSTEM) 95
RVMAP (BPS) 196
SAM (SYSTEM) 96
SCALAR (SHELL) 27
SCALE (GPM) 170
SERIES (SHELL) 28
SETREC (BPS) 197
SGN (Mathematic Library) 54
SHIFT (ISPP) 136
SIGMOID (BPS) 198
SIN (Mathematic Library) 55
SIZE (GPM) 172
SNAPSHOT (SHELL) 29
SPCF (ISPP) 137
SPLINE (ISPP) 138
SPRINTF (SL_UTIL) 70
SQRT (Mathematic Library) 56
STRING (SHELL) 30
STRLEN (SHELL) 31
SYMBOLS (SHELL) 32
Scalar (SL_UTIL) 67

Series (SL_UTIL) 68
Snapshot (SL_UTIL) 69
Sprintf (SL_UTIL) 71
String (SL_UTIL) 72
TAN (Mathematic Library) 57
TEACH (BPS) 199
TEXT2BUFFER (DCM) 266
TEXT2STRING (SL_UTIL) 73
TITLE (GPM) 173
TOLOWER (SL_UTIL) 75
TOUPPER (SL_UTIL) 76
TRANS (ISPP) 139
TXT2BUF (DCM) 265
TYPEOF (SHELL) 33
UNDEF (SHELL) 34
UNITMAT (SL_UTIL) 74
UNIX (SHELL) 35
URAND (ISPP) 140
WAIT (SYSTEM) 97
WALGO (BPS) 200
WCLOSE (GPM) 174
WE (GPM) 175
WEIGHT (BPS) 201
WELCOME (SHELL) 36
WGTLOAD (BPS) 202
WGTSET (BPS) 204
WHILE (SHELL) 37
WINDOW (ISPP) 141
WINIT (BPS) 205
WOPEN (GPM) 176
WRITE_TYPE (SHELL) 38
ZERO (SYSTEM) 98