「音声信号処理ツールキット」
リファレンスマニュアル

REFERENCE MANUAL for
Speech Signal Processing Toolkit Ver. 3.0

December 16, 2003

各コマンドのヘルプメッセージ (コマンドの説明，使い方，オプションの説明等) は，-h オプションを指定するにより表示されます.

例：mcep コマンドの場合 (% はシェルのプロンプト)

```
> % mcep -h
>
>  mcep - mel cepstral analysis
>
>   usage:
>        mcep [ options ] [ infile ] > stdout
>   options:
>        -a a  : all-pass constant               [0.35]
>        -m m  : order of mel cepstrum           [25]
>        -l l  : frame length                    [256]
>        -h    : print this message
>      (level 2)
>        -i i  : minimum iteration               [2]
>        -j j  : maximum iteration               [30]
>        -d d  : end condition                   [0.001]
>        -e e  : small value added to periodgram [0]
>   infile:
>        windowed sequences (float)              [stdin]
>   stdout:
>        mel-cepstrum (float)
```

本ツールキットに関する情報は,

http://kt-lab.ics.nitech.ac.jp/~tokuda/SPTK/

を御覧下さい. 現在,「音声信号処理ツールキットの使用例」(Examples of Using Speech Signal Processing Toolkig) を見ることができます.
本ツールキットに関するバグレポート，コメント，質問などは

dsp-cmnd@ip.titech.ac.jp

まで email でお願い致します. 質問などにはできる限りお答えしたいと思いますが，それをあらかじめ保証するものではないことを御了承下さい.

目 次

NAME

acep – 適応ケプストラム分析 [4545]

SYNOPSIS

acep [–**m** M] [–**l** L] [–**t** T] [–**k** K] [–**p** P] [–**s**] [–**e** E] [–**P** Pa] [***pefile***] < ***infile***

DESCRIPTION

適応ケプストラム分析を行い，ケプストラム係数 [4], [5], を標準出力に，予測残差を *pefile*（省略時は出力しません）に出力します.

データ形式は入力，出力とも float 形式です.

適応ケプストラム分析では，時刻 n から $n+1$ への係数更新アルゴリズムは

$$
\begin{aligned}
\boldsymbol{c}^{(n+1)} &= \boldsymbol{c}^{(n)} - \mu^{(n)}\hat{\nabla}\varepsilon_{\tau}^{(n)} \\
\hat{\nabla}\varepsilon_{0}^{(n)} &= -2e(n)\boldsymbol{e}^{(n)} \qquad (\tau = 0) \\
\hat{\nabla}\varepsilon_{\tau}^{(n)} &= -2(1-\tau)\sum_{i=-\infty}^{n}\tau^{n-i}e(i)\boldsymbol{e}^{(i)} \quad (0 \le \tau < 1) \\
\hat{\nabla}\varepsilon_{\tau}^{(n)} &= \tau\hat{\nabla}\varepsilon_{\tau}^{(n-1)} - 2(1-\tau)e(n)\boldsymbol{e}^{(n)} \\
\mu^{(n)} &= \frac{k}{M\varepsilon^{(n)}} \\
\varepsilon^{(n)} &= \lambda\varepsilon^{(n-1)} + (1-\lambda)e^2(n)
\end{aligned}
$$

で与えられます．ただし，$\boldsymbol{c} = [c(1), \ldots, c(M)]^T$, $\boldsymbol{e}^{(n)} = [e(n-1), \ldots, e(n-M)]^T$ です．また，ゲインを表す $c(0)$ は次式で推定されます.

$$c(0) = \frac{1}{2}\log\varepsilon^{(n)}$$

図1に適応ケプストラム分析系を示します.

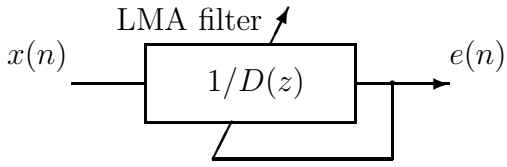


図 1: 適応ケプストラム分析系

OPTIONS

–m	M	分析次数.	[25]
–l	L	忘却係数 λ.	[0.98]
–t	T	momentum 定数 τ.	[0.9]
–k	K	ステップサイズ k.	[0.1]
–p	P	ケプストラム係数の出力周期.	[1]
–s		出力周期内で平均化したケプストラム係数を出力.	[FALSE]
–e	E	$\varepsilon^{(n)}$ の最小値	[0.0]
–P	Pa	LMA フィルタのパデ近似次数. Pa は 4 または 5 を指定できます.	[4]

EXAMPLE

float 形式のファイル *data.f* を次数 15 次で適応ケプストラム分析し，ケプストラム係数は 100 サンプル毎に *data.acep* に，予測残差は *data.er* に出力する:

```
acep -m 15 -p 100 data.er < data.f > data.acep
```

SEE ALSO

uels, gcep, mcep, mgcep, amcep, agcep, lmadf

NAME

acorr – 短時間自己相関を求める

SYNOPSIS

acorr [–**m** M] [–**l** L] [***infile***]

DESCRIPTION

ファイル *infile*（省略時は標準入力）から読み込んだ，フレーム毎の m 次の自己相関系列を標準出力に書き出します．つまり，入力データを

$$x(0), x(1), \cdots, x(L-1)$$

として，

$$r(k) = \sum_{m=0}^{L-1-k} x(m)x(m+k), \qquad k = 0, 1, \cdots, M$$

を計算し，

$$r(0), r(1), \cdots, r(M)$$

を出力します．データ形式は入力，出力ともに float 形式です．

OPTIONS

–**m**	M	出力する自己相関系列の次数.	[25]
–**l**	L	入力データのフレーム長.	[256]

EXAMPLE

float 形式のファイル *data.f* をフレーム長 256，フレーム周期 100 のフレームに分割し，各フレームにブラックマン窓を掛けたデータの 10 次の自己相関系列をファイル *data.acorr* に出力する:

```
frame -l 256 -p 100 < data.f | window | acorr -m 10 > data.acorr
```

SEE ALSO

c2acr, levdur

NAME

agcep – 適応一般化ケプストラム分析 [9]

SYNOPSIS

agcep [–**m** M] [–**g** G] [–**l** L] [–**t** T] [–**k** K] [–**p** P]
[–**s**] [–**n**] [–**e** E] [***pefile***] < ***infile***

DESCRIPTION

適応一般化ケプストラム分析を行い，正規化一般化ケプストラム係数 $c_\gamma(m)$ を標準出力に，予測残差を *pefile*（省略時は出力しません）に出力します.

データ形式は入力，出力とも float 形式です.

適応一般化ケプストラム分析では，時刻 n から $n+1$ への係数更新アルゴリズムは

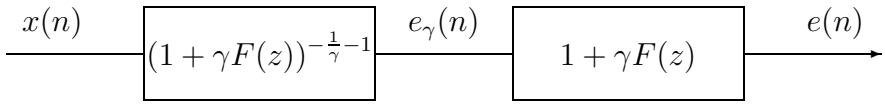
$$
\begin{aligned}
\boldsymbol{c}_\gamma^{(n+1)} &= \boldsymbol{c}_\gamma^{(n)} - \mu^{(n)}\hat{\nabla}\varepsilon_\tau^{(n)} \\
\hat{\nabla}\varepsilon_0^{(n)} &= -2e_\gamma(n)\boldsymbol{e}_\gamma^{(n)} \qquad (\tau = 0) \\
\hat{\nabla}\varepsilon_\tau^{(n)} &= -2(1-\tau)\sum_{i=-\infty}^{n}\tau^{n-i}e_\gamma(i)\boldsymbol{e}_\gamma^{(i)} \quad (0 \le \tau < 1) \\
\hat{\nabla}\varepsilon_\tau^{(n)} &= \tau\hat{\nabla}\varepsilon_\tau^{(n-1)} - 2(1-\tau)e_\gamma(n)\boldsymbol{e}_\gamma^{(n)} \\
\mu^{(n)} &= \frac{k}{M\varepsilon^{(n)}} \\
\varepsilon^{(n)} &= \lambda\varepsilon^{(n-1)} + (1-\lambda)e_\gamma^2(n)
\end{aligned}
$$

で与えられます. ただし，$\boldsymbol{c}_\gamma = [c_\gamma(1), \ldots, c_\gamma(M)]^T$, $\boldsymbol{e}_\gamma = [e_\gamma(n-1), \ldots, e_\gamma(n-M)]^T$ です. $e_\gamma(n)$ は,

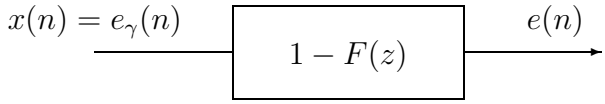
$$F(z) = \sum_{m=1}^{M} c_\gamma(m)z^{-m}$$

として，入力 $x(n)$ をフィルタ $(1+\gamma F(z))^{-\frac{1}{\gamma}-1}$ に通した出力です.

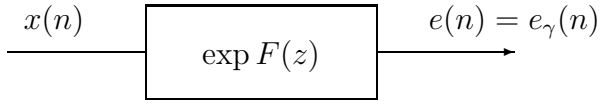
ここで，特に n を自然数として $\gamma = -1/n$ の場合の適応一般化ケプストラム分析系を図1に示します. $n = 1$ のときにはLMS linear predictorと等価であり，$n \to \infty$ とした場合には適応ケプストラム分析に等価になります.



(a) $-1 \leq \gamma \leq 0$



(b) $\gamma = -1$



(c) $\gamma = 0$

図 1: 適応一般化ケプストラム分析系

OPTIONS

–**m**	M	分析次数.	[25]
–**g**	G	一般化ケプストラムのべきパラメータ γ.	[1]
		ただし，$G > 1.0$ のときは $\gamma = -1/G$.	
–**l**	L	忘却係数 λ.	[0.98]
–**t**	T	momentum 定数 τ.	[0.9]
–**k**	K	ステップサイズ k.	[0.1]
–**p**	P	一般化ケプストラム係数の出力周期.	[1]
–**s**		出力周期内で平均化した一般化ケプストラム係数を出力.	[FALSE]
–**n**		規格化した一般化ケプストラム係数を出力.	[FALSE]
–**e**	E	$\varepsilon^{(n)}$ の最小値.	[0.0]

EXAMPLE

float 形式のファイル *data.f* を次数 15 次で適応一般化ケプストラム分析し，一般化ケプストラム係数を *data.agcep* に，予測残差は *data.er* に出力する:

```
agcep -m 15 data.er < data.f > data.agcep
```

SEE ALSO

acep, amcep, glsadf

NAME

amcep – 適応メルケプストラム分析 [11121112]

SYNOPSIS

amcep [–**m** M] [–**a** A] [–**l** L] [–**t** T] [–**k** K] [–**p** P] [–**s**] [–**e** E] [–**P** Pa] [***pefile***] < ***infile***

DESCRIPTION

適応メルケプストラム分析を行い，メルケプストラム係数 $c_\alpha(m)$ を標準出力に，予測残差を *pefile*（省略時は出力しません）に出力します.

データ形式は入力，出力とも float 形式です.

適応メルケプストラム分析では, 時刻 n から $n+1$ への係数 $b(m)$ の更新アルゴリズムは,

$$
\begin{aligned}
\boldsymbol{b}^{(n+1)} &= \boldsymbol{b}^{(n)} - \mu^{(n)}\hat{\nabla}\varepsilon_\tau^{(n)} \\
\hat{\nabla}\varepsilon_0^{(n)} &= -2e(n)\boldsymbol{e}_\Phi^{(n)} \qquad (\tau = 0) \\
\hat{\nabla}\varepsilon_\tau^{(n)} &= -2(1-\tau)\sum_{i=-\infty}^{n} \tau^{n-i} e(i)\boldsymbol{e}_\Phi^{(i)} \quad (0 \le \tau < 1) \\
\hat{\nabla}\varepsilon_\tau^{(n)} &= \tau\hat{\nabla}\varepsilon_\tau^{(n-1)} - 2(1-\tau)e(n)\boldsymbol{e}_\Phi^{(n)} \\
\mu^{(n)} &= \frac{k}{M\varepsilon^{(n)}} \\
\varepsilon^{(n)} &= \lambda\varepsilon^{(n-1)} + (1-\lambda)e^2(n)
\end{aligned}
$$

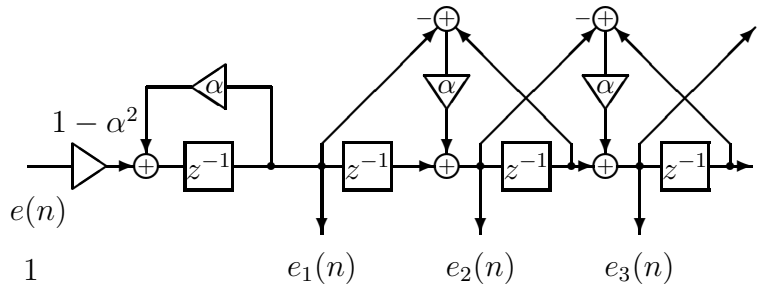


図 1: フィルタ $\Phi_m(z)$

で与えられます. ただし, $\boldsymbol{b} = [b(1), b(2), \ldots, b(M)]^T$, $\boldsymbol{e}_\Phi^{(n)} = [e_1(n), e_2(n), \ldots, e_M(n)]^T$ であり，$e_m(n)$ は逆フィルタ出力を図1に示す $\Phi_m(z)$ に通した出力です.

この係数 $b(m)$ はMLSA フィルタの係数と等価ですから，線形変換によりメルケプストラム係数 $c_\alpha(m)$ を得ることが出来ます（コマンド b2mc, mc2b 参照）.

図2に適応メルケプストラム分析系を示します.

ここで，$1/D(z)$ は MLSA フィルタにより実現されます.

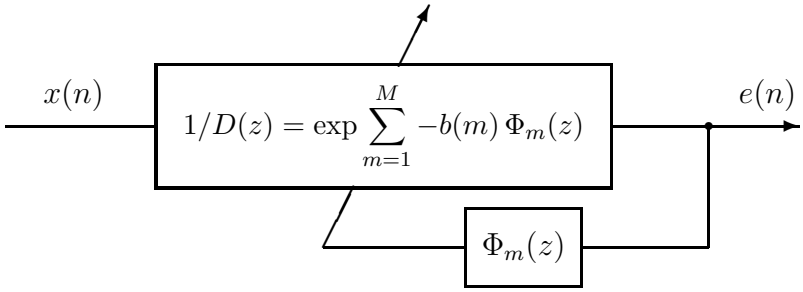


図 2: 適応メルケプストラム分析系

OPTIONS

–m	M	分析次数.	[25]
–a	A	周波数圧縮パラメータ α.	[0.35]
–l	L	忘却係数 λ.	[0.98]
–t	T	momentum 定数 τ.	[0.9]
–k	K	ステップサイズ k.	[0.1]
–p	P	メルケプストラム係数の出力周期.	[1]
–s		出力周期内で平均化したメルケプストラム係数を出力.	[FALSE]
–e	E	$\varepsilon^{(n)}$ の最小値.	[0.0]
–P	Pa	MLSA フィルタのパデ近似次数. Pa は 4 または 5 を指定できます.	[4]

EXAMPLE

float 形式のファイル *data.f* を次数 15 次で適応メルケプストラム分析し，メルケプストラム係数を 100 サンプル毎に *data.amcep* に出力する:

```
amcep -m 15 -p 100  < data.f > data.amcep
```

SEE ALSO

acep, agcep, mc2b, b2mc, mlsadf

NAME

average – ブロック毎の平均を計算する

SYNOPSIS

average [–**l** L] [–**n** N] [***infile***]

DESCRIPTION

指定されたファイルから長さ L の時系列

$$x(0), x(1), \ldots, x(L-1)$$

を読み込み, その平均

$$\frac{x(0) + x(1) + \ldots + x(L-1)}{L}$$

を出力します．ただし，$L = 0$ のときにはファイル全体の平均を出力します.

データ形式は入力，出力とも float 形式です.

OPTIONS

–**l**	L	平均をとるブロック長.	[0]
–**n**	N	平均をとるブロックの次数.	[L-1]

EXAMPLE

float 形式のファイル *data.f* の平均を計算し，*data.av* に出力する:

```
average < data.f > data.av
```

SEE ALSO

histogram, vsum

NAME

b2mc – MLSA フィルタの係数からメルケプストラム係数を求める

SYNOPSIS

b2mc [–**m** M] [–**a** A] [***infile***]

DESCRIPTION

MLSA フィルタの係数 $b(m)$ からメルケプストラム係数 $c_\alpha(m)$ を求め，標準出力に出力します.

データ形式は入力，出力とも float 形式です.

係数 $b(m)$ からメルケプストラム係数 $c_\alpha(m)$ への変換式は

$$c_\alpha(m) = \begin{cases} b(M), & m = M \\ b(m) + \alpha b(m+1), & 0 \le m < M \end{cases}$$

で与えられます．この変換式は mc2b の逆変換となります.

OPTIONS

–**m**	M	MLSA フィルタの係数の次数.	[25]
–**a**	A	周波数圧縮パラメータ α.	[0.35]

EXAMPLE

float 形式の MLSA フィルタの係数（$M = 15, \alpha = 0.35$）のファイル *data.b* をメルケプストラムに変換し，*data.mcep* に出力する:

```
b2mc -m 15 < data.b > data.mcep
```

SEE ALSO

b2mc, mcep, mlsadf

NAME

bcp – データブロックの一部または全部のコピー

SYNOPSIS

bcp [–**l** l] [–**L** L] [–**n** n] [–**N** N] [–**s** s] [–**S** S] [–**e** e] [–**f** f]
[+***type***] [***infile***]

DESCRIPTION

bcp は，指定されたファイルをオプションで指定された大きさのブロックに切り分け，そのブロックの一部または全部を標準出力に出力します.

データ型に ascii を指定した場合，入力ファイルの中の文字列をデータの一単位とし，出力ブロックを改行で区切ります.

ファイルが指定されなかった場合，データは標準入力から読み込まれます.

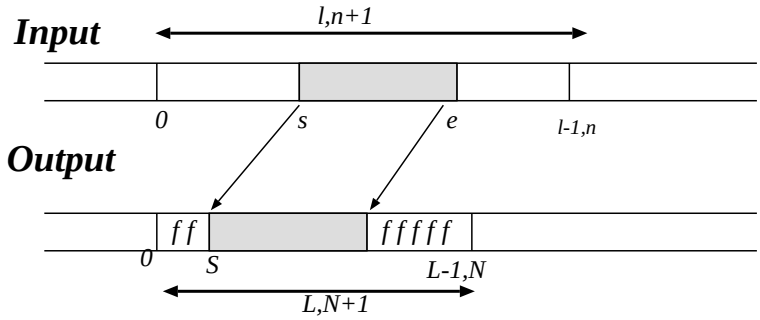


図 3: Example of the bcp command

OPTIONS

–**l**	l	ファイルを切り分けるブロックの大きさ.	[512]
		一つのブロックにいくつのデータが入っているかで指定.	
–**L**	L	出力するブロックの大きさ.	[N/A]
–**n**	n	ファイルを切り分けるブロックの次数.	[l-1]
		ファイルを切り分けるブロックの大きさは $n+1$ となる.	
–**N**	N	出力するブロックの次数.	[N/A]
		出力するブロックの大きさは $N+1$ となる.	

–**s**	*s*	切り分けたブロックの中でのコピー開始データ番号.	[0]
–**S**	*S*	出力ブロックの中でのコピー開始データ番号.	[0]
–**e**	*e*	切り分けたブロックの中でのコピー終了データ番号.	[EOF]
–**f**	*f*	出力ブロックのうち，コピーされない部分を *f* で埋める. 入力データが ascii 型でない場合に有効.	[0]
+*t*		入力データの形式.	[f]

c	char 型 (1byte)	s	short 型 (2bytes)
i	int 型 (4bytes)	l	long 型 (4bytes)
f	float 型 (4bytes)	d	double 型 (8bytes)
a	ascii 文字列		

EXAMPLE

ファイル *data.f* には float 形式で配列 a(0), a(1), a(2), ... , a(20) のデータが繰り返し書き込まれているとする.
この中から a(1), a(2), ... , a(10) をとり出してファイル *data.bcp* に出力したいならば

```
bcp data.f +f -l 21 -s 1 -e 10 > data.bcp
```

また，おなじファイル *data.f* に対して

```
bcp data.f +f -l 21 -s 3 -e 5 -S 6 -L 10 > data.bcp
```

とすれば，出力は以下で与えられるブロック

```
0, 0, 0, 0, 0, 0, a(3), a(4), a(5), 0
```

の繰り返しになる.

SEE ALSO

bcut, merge, reverse

NAME

bcut – ファイルの切り出し（ブロックを単位とする範囲指定可能）

SYNOPSIS

bcut [–**s** *S*] [–**e** *E*] [–**l** *L*] [–**n** *N*] [+***type***] [***infile***]

DESCRIPTION

bcut は，指定されたファイルから，オプションで指定された部分を切り出します．ファイルが指定されなかった場合，データは標準入力から読み込まれます．

OPTIONS

–**s**	*S*	切り出すブロックの開始点．	[0]
–**e**	*E*	切り出すブロックの終了点．	[EOF]
–**l**	*L*	データのブロック長．	[1]
–**n**	*N*	データブロックの次数．ブロック長は次数+1 となります．	[L-1]
+*t*		入力データの形式．	[f]

c	char 型 (1byte)	s	short 型 (2bytes)
i	int 型 (4bytes)	l	long 型 (4bytes)
f	float 型 (4bytes)	d	double 型 (8bytes)

EXAMPLE

float 形式のファイル *data.f* の 3 番目から 5 番目までのデータを切り出し，*data.cut* に出力する:

```
bcut -s 3 -e 5 data.f > data.cut
```

例えば，ファイル *data.f* に，float 形式の数値データ

$$1, 2, 3, 4, 5, 6, 7$$

が入っていた場合，ファイル *data.cut* の中には，

$$4, 5, 6$$

が得られます．

同じデータをブロック長を指定して切り出す:

```
bcut -l 2 data.f -s 1 -e 2 > data.cut
```

この場合,

$$3, 4, 5, 6$$

が切り出されます.

ディジタルフィルタをパルス列で励振し，定常状態に達するまでの 100 点を切り捨てる:

```
train -p 10 -l 256 | dfs -a 1 0.8 0.6 | bcut -s 100 > data.cut
```

この場合，ファイル *data.cut* の大きさは 156 点となります.

窓長 256 点の窓で次々と窓掛けされたデータが

```
sin -p 30 -l 2000 | window > data.f
```

のようにつくられたとき，3 番目 (先頭を 0 番目として) の窓掛けデータだけを取り出す:

```
bcut -l 256 -s 3 -e 3 < data.f > data.cut
```

SEE ALSO

bcp,merge,reverse

NAME

bell – ベルを鳴らす

SYNOPSIS

bell [*num*]

DESCRIPTION

*bell*は，オプションで指定した回数だけベルを鳴らします.

OPTIONS

num ベルを鳴らす回数. [1]

EXAMPLE

ベルを 10 回鳴らす :

```
bell 10
```

NAME

cat2 – ファイルを標準エラー出力へ出力

SYNOPSIS

cat2 [**–n**] [**file**]

DESCRIPTION

cat2 は，引数で指定されたフィアルを標準エラー出力に出力します．ファイルの指定が無い場合，標準入力から読み込まれます．

OPTIONS

–n 行番号を出力． [FALSE]

EXAMPLE

standard error に，*file* を出力する

```
cat2 file
```

NAME

c2acr – ケプストラムから自己相関を求める

SYNOPSIS

c2acr [–**m** M_1] [–**M** M_2] [–**l** L] [***infile***]

DESCRIPTION

標準入力から入力される M_1 次のケプストラム

$$c(0), c(1), \ldots, c(M_1)$$

から，M_2 次の自己相関

$$r(0), r(1), \ldots, r(M_2)$$

を求めます.

データ形式は入力，出力とも float 形式です.

M_1 次のケプストラムをフーリエ変換して，対数スペクトルを求め，パワースペクトルを計算します．このパワースペクトルを逆フーリエ変換し，自己相関を求めます.

OPTIONS

–**m**	M_1	ケプストラムの次数.	[25]
–**M**	M_2	自己相関の次数.	[25]
–**l**	L	FFT 長.	[256]

EXAMPLE

float 形式の 30 次のケプストラムファイル *data.cep* から自己相関を介して 15 次の線形予測係数を求め，*data.lpc* に出力する:

```
c2acr -m 30 -M 15 < data.cep | lev_dur -n 15 > data.lpc
```

SEE ALSO

uels, c2sp, c2ir, lpc2c

NAME

c2ir – ケプストラムからインパルス応答を求める

SYNOPSIS

c2ir [**–l** L] [**–m** M_1] [**–M** M_2] [**–i**] [***infile***]

DESCRIPTION

ファイル *infile*（省略時は標準入力）から最小位相ケプストラムを読み込み，最小位相インパルス応答を出力します．つまり, 入力数列を

$$c(0), c(1), c(2), \cdots, c(M_1)$$

として,

$$h(n) = \begin{cases} h(0) = \exp(c(0)) \\ h(n) = \displaystyle\sum_{k=1}^{M_1} \frac{k}{n} c(k) h(n-k) \quad n \geq 1 \end{cases}$$

を計算し,

$$h(0), h(1), h(2), \cdots, h(L-1)$$

を出力します．データ形式は，入出力ともに float 形式です．

OPTIONS

–l	L	インパルス応答の持続長.	[256]
–m	M_1	ケプストラム係数の次数.	[25]
–M	M_2	インパルス応答の持続次数.	[L-1]
–i		最小位相数列を読み込んでケプストラムを出力.	[FALSE]

ケプストラム次数 M_1 の指定なしで，かつ入力ケプストラムの個数が L より短い場合，実際に読み込んだ次数を M_1 の値とする．

EXAMPLE

float 形式の 30 次のケプストラムファイル *data.cep* を読み込んで，インパルス応答を $n = 0 \sim 99$ の範囲で求め，*data.ir* に出力する:

```
c2ir -l 100 -m 30 data.cep > data.ir
```

SEE ALSO

c2sp, c2acr

NAME

c2sp – ケプストラムからスペクトルを求める

SYNOPSIS

c2sp [–**m** M] [–**l** L] [–**p**] [–**o** O] [***infile***]

DESCRIPTION

c2sp は，最小位相のケプストラムからスペクトルを求めます．データ形式は入力，出力ともに float 形式です．

OPTIONS

–**m**	M	ケプストラムの次数.	[25]
–**l**	L	フレーム長.	[256]
–**p**		スペクトルの位相を出力.	[FALSE]
–**o**	O	–p オプションが指定されていない場合，出力するスペクトルのスケールを指定.	[0]

$O = 0$ $20 \times \log |H(z)|$
$O = 1$ $\ln |H(z)|$
$O = 2$ $|H(z)|$

–p オプションが指定されている場合，出力する位相の単位を指定.

$O = 0$ $\arg |H(z)| \div \pi$ $[\pi\ rad.]$
$O = 1$ $\arg |H(z)|$ $[rad.]$
$O = 2$ $\arg |H(z)| \times 180 \div \pi$ $[deg.]$

EXAMPLE

float 形式の 15 次のケプストラムファイル *data.cep* からランニングスペクトルを求めて画面上に表示する:

```
c2sp -m 15 data.cep | grlogsp | xgr
```

SEE ALSO

uels, mgc2sp

NAME

ccp – データの切り出しと変換

SYNOPSIS

ccp [–**S** S_1] [–**s** S_2] [–**E** E_1] [–**e** E_2] [+t_1t_2] [+$type_1$] [+$type_2$] [***infile***]

DESCRIPTION

ccp は，指定されたファイルから，オプションで指定された部分を切り出し，指定されたデータ形式に変換して, 標準出力に出力します．ファイルが指定されなかった場合，データは標準入力から読み込まれます．

–**S**, –**E** オプション，及び–**s**, –**e** オプションでデータの切り出し範囲を指定します．

入力データ，及び出力データの形式は，+t_1t_2 (*iodata_type*) で指定します．

OPTIONS

–**S** S_1	切り出すデータの開始バイトアドレス.	[0]
–**s** S_2	切り出すデータの開始データ番号.	[0]
–**E** E_1	切り出すデータの終了バイトアドレス.	[EOF]
–**e** E_2	切り出すデータの終了データ番号.	[0]
+t_1t_2	入出力データの型指定．t_1，t_2 はそれぞれ入力データ，出力データのデータ型を表す文字.	[ff]

c	char 型 (1byte)	s	short 型 (2bytes)
i	int 型 (4bytes)	l	long 型 (4bytes)
f	float 型 (4bytes)	d	double 型 (8bytes)
a	ascii 文字列		

データを t_1 型から t_2 型へ型変換して出力する． t_2 が省略された場合は $t_2 = t_1$ とみなし型変換は行なわない.

+$type_1$	入力データの形式.	[f]
+$type_2$	出力データの形式.	[f]

データアドレス及びデータ番号の指定は，10 進数または数字の前に 0x を付けた 16 進数で指定します． ascii 文字列は入力の場合，空白，タブ，改行で区切られた数字の列で，出力の場合は “%g¥n” 変換による文字列です.

EXAMPLE

ファイル *data.s* の 256 バイト目から short 型で入っている整数の 10〜19 番目を float 型に変換して *data.f* に出力する:

```
ccp -S 0x100 -s 10 -e 19 +sf data.s > data.f
```

数字を入力して *data.f* に float 形式で出力する:

```
ccp +af > data
```

SEE ALSO

bcp, bcut, merge, reverse, x2x

NAME

cdist – ケプストラム距離の計算

SYNOPSIS

cdist [–**m** M] [–**o** O] [–**f**] ***cfile*** [***infile***]

DESCRIPTION

ファイル *infile*（省略時は標準入力）とファイル *cfile* とのケプストラム距離を計算します．*infile* のフレーム t のケプストラムを

$$c_{1,t}(0), c_{1,t}(1), c_{1,t}(2), \cdots, c_{1,t}(M)$$

cfile のフレーム t のケプストラムを

$$c_{2,t}(0), c_{2,t}(1), c_{2,t}(2), \cdots, c_{2,t}(M)$$

とした時，フレームごとの自乗ケプストラム距離は

$$d(t) = \sum_{k=1}^{M} (c_{1,t}(k) - c_{2,t}(k))^2$$

となり，ファイル全体のケプストラム距離は

$$d = \frac{1}{T} \sum_{t=0}^{T} d(t)$$

となります．

但し、T は *infile*，*cfile* のうち少ない方のフレーム数となります．

OPTIONS

–**m** M 最小位相ケプストラムの次数． [25]

–**o** O ケプストラム距離のフォーマット． [0]

$O = 0$ $\frac{10}{\ln 10}\sqrt{2d(t)}$ [db]

$O = 1$ $d(t)$

$O = 2$ $\sqrt{d(t)}$

–**f** フレームごとのケプストラム距離を出力 [FALSE]

EXAMPLE

float 形式の 15 次のケプストラムファイル *data1.cep*，*data2.cep* との自乗ケプストラム距離を求め，表示します.

```
cdist -m 15 data1.cep data2.cep | dmp
```

SEE ALSO

acep, agcep, amcep, mcep

NAME

clip – データの値の範囲を制限 (クリッピング) する

SYNOPSIS

clip [−**y** y_{min} y_{max}] [−**ymin** y_{min}] [−**ymax** y_{max}] [***infile***]

DESCRIPTION

clip は，float 型で与えられるデータ列を入力ファイル *infile*(省略時は標準入力) から読み込み，指定されたオプションで指定される最小値から最大値の範囲外の値であれば，最小値または最大値に値を制限して標準出力に出力します.

OPTIONS

−**y**	y_{min} y_{max}	データの値の範囲の最小値と最大値.	[−1.0 1.0]
−**ymin**	y_{min}	データの値の範囲の最小値. 上限は制限しない.	[N/A]
−**ymax**	y_{max}	データの値の範囲の最大値. 下限は制限しない.	[N/A]

EXAMPLE

float 形式のファイル *data.f* には以下のデータが入っているとする.

1.0, 2.0, 3.0, 4.0, 5.0, 6.0

ここで

```
clip -y 2.5 5.5 < data.f > data.clip
```

とするとファイル *data.clip* には以下のデータが入る.

2.5, 2.5, 3.0, 4.0, 5.0, 5.5

NAME

da – オーディオデータの再生

SYNOPSIS

da [–**s** *S*] [–**c** *C*] [–**g** *G*] [–**a** *A*] [–**o** *O*] [–**w**] [–**H** *H*]
[–**v**] [+*type*] [***infile1***] [***infile2***] **...**

DESCRIPTION

da は，指定されたファイルのデータをスピーカーまたはヘッドフォンから再生します．ファイルが指定されなかった場合，データは標準入力から読み込まれます．オーディオデバイスの対応していない標本化周波数の場合，適切な周波数にアップサンプリングして再生します．

16bit オーディオデバイスの使用可能な Linux (i386) と FreeBSD (i386 newpcm driver)，SunOS 4.1.x，SunOS 5.x (SPARC) に対応しています．

オプションは環境変数で設定することもできます．

DA_GAIN	ゲイン
DA_AMPGAIN	振幅ゲイン
DA_PORT	出力ポート
DA_HDRSIZE	ヘッダサイズ
DA_FLOAT	入力データ型を float 型に指定

OPTIONS

–**s**	*S*	標本化周波数 (kHz)．指定可能な値は 8，10，11.025，12，16，22.05，32，44.1，48 (但し，11.025，22.05，44.1 はそれぞれ 11，22，44 に省略可能) です．	[16]
–**g**	*G*	ゲイン．入力データを 2^G 倍して出力します．	[0]
–**a**	*A*	オーディオデバイスの振幅ゲイン (0..100)．指定しない場合はそれまでの設定が適用されます．	[N/A]
–**o**	*O*	出力ポート (s : スピーカ，h : ヘッドフォン)．SPARC 版のみ有効．	[s]
–**w**		入力データのバイトスワップを行う．	[FALSE]
–**H**	*H*	ヘッダサイズ (バイト数)．	[0]
–**v**		データファイル名を表示．	[FALSE]
+*type*		入力データ型 (s : short 型，f : float 型)．	[s]

EXAMPLE

標本化周波数 8kHz の short 型の音声データファイル *data.s* をヘッドフォンから再生する:

```
da -s 8 -o h data.s
```

BUGS

linux 版では，出力ポートの指定ができません.

NAME

dawrite – オーディオデバイスへの書き出し

SYNOPSIS

dawrite [–**s** *S*] [–**c** *C*] [–**g** *G*] [–**a** *A*] [–**o** *O*] [–**w**] [–**H** *H*] [–**v**] [+*type*]

[***infile1***] [***infile2***] **...**

DESCRIPTION

dawrite は，指定されたファイルのデータをスピーカーまたはヘッドフォンから再生します．ファイルが指定されなかった場合，データは標準入力から読み込まれます．ゲインを G とすると入力データを 2^G 倍して出力し，振幅ゲインはオーディオデバイスのゲインを 0〜100 の間で設定します．指定できる標本化周波数は，オーディオデバイスドライバの対応している値のみとなります．なお，標本化周波数の指定には 11.025，22.05，44.1kHz をそれぞれ 11，22，44 に省略することができます．

16bit オーディオデバイスの使用可能な Linux(i386) と FreeBSD(i386 newpcm driver), SunOS 4.1.x，SunOS 5.x(SPARC) に対応しています．

オプションは環境変数で設定することもできます．

DA_SAMPFREQ	標本化周波数
DA_GAIN	ゲイン
DA_AMPGAIN	振幅ゲイン
DA_PORT	出力ポート指定
DA_HDRSIZE	ヘッダサイズ
DA_FLOAT	入力データ型を float 型にする

OPTIONS

–**s**	*S*	標本化周波数.(8，11.025，16，22.05，32，44.1，48 kHz，但し，linux では 44.1kHz 系統のみ．また，11.025，22.05，44.1 は，それぞれ 11，22，44 に省略可能)	[16]
–**g**	*G*	ゲイン.(..，-2，-1，0，-1，2，..)	[0]
–**a**	*A*	振幅ゲイン.(0..100) 指定しない場合はそれまでの設定が適用されます．	[N/A]
–**o**	*O*	出力ポート指定.(s : スピーカ，h : ヘッドフォン．SPARC 版のみ有効)	[s]
–**w**		入力データのバイトスワップを行う	[FALSE]

–H *H*	ヘッダサイズ（バイト単位）.	[0]
–v	データファイル名を表示.	[FALSE]
+type	入力データ型.（s : short 型，f : float 型）	[s]

EXAMPLE

標本化周波数 8kHz の short 型の音声データファイル *data.s* をヘッドフォンから再生する:

```
dawrite -s 8 -g -1 -o h data.s
```

BUGS

linux 版では，出力ポートの指定ができません.

SEE ALSO

da, us

NAME

decimate – デシメーション（データ列の間引き）

SYNOPSIS

decimate [–**p** P] [–**s** S] [***infile***]

DESCRIPTION

標準入力から入力される時系列

$$x(0), x(1), x(2), \ldots,$$

から周期 P，先頭 S として，

$$x(S), x(S+P), x(S+2P), x(S+3P), \ldots,$$

のように decimation します.

データ形式は入力，出力とも float 形式です.

OPTIONS

–**p**	P	decimation の周期.	[10]
–**s**	S	decimation を始める先頭からのサンプル数.	[0]

EXAMPLE

float 形式のデータ *data.f* を周期 2 で decimation し，さらに 0 を周期 2 で interpolation して *data.di* に出力する:

```
decimate -p 2  < data.f | interpolate -p 2 > data.di
```

SEE ALSO

interpolate

NAME

delay – 遅延

SYNOPSIS

delay [–**s** S] [–**f**] [***infile***]

DESCRIPTION

delay は，入力された信号を遅延させます．つまり，

$$x(0), x(1), \ldots, x(T)$$

を，

$$\underbrace{0, \ldots, 0}_{S}, x(0), x(1), \ldots, x(T)$$

とします．-f オプションを指定すると，

$$\underbrace{0, \ldots, 0}_{S}, x(0), x(1), \ldots, x(T-S)$$

となります．

データ形式は入力，出力ともに float 形式です．

OPTIONS

–**s**	S	開始サンプル．	[0]
–**f**		入力ファイルのサイズを保つ．	[False]

EXAMPLE

float 形式のファイル *data.f* には以下のデータが入っているとする．

$$1.0, 2.0, 3.0, 4.0, 5.0, 6.0$$

ここで

```
delay -s 3 < data.f > data.delay
```

とするとファイル *data.delay* には以下のデータが入る.

0.0, 0.0, 0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0

```
delay -s 3 -f < data.f > data.delay
```

とするとファイル *data.delay* には以下のデータが入る.

0.0, 0.0, 0.0, 1.0, 2.0, 3.0

NAME

df2 – 2 次のディジタルフィルタ

SYNOPSIS

df2 [–**f** f_0] [–**p** f_1 b_1] [–**z** f_2 b_2] [***infile***]

DESCRIPTION

入力ファイル *infile* (省略時には標準入力) から読み込まれたデータを，オプションで指定された中心周波数と帯域幅で与えられる 2 次のディジタルフィルタで, フィルタリングし，標準出力に出力します．フィルタの伝達関数は,

$$H(z) = \frac{1 - 2\exp(-\pi b_2/f_0)\cos(2\pi f_2/f_0)z^{-1} + \exp(-2\pi b_2/f_0)z^{-2}}{1 - 2\exp(-\pi b_1/f_0)\cos(2\pi f_1/f_0)z^{-1} + \exp(-2\pi b_1/f_0)z^{-2}}$$

で与えられます.
–p および –z を複数回指定することにより任意のフィルタが構成できます.
データ形式は，入出力ともに float 形式です.

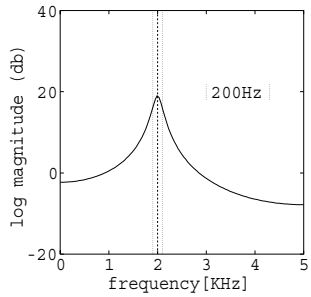
OPTIONS

–**f**	f_0	標本化周波数 f_0[Hz].	[10000]
–**p**	f_1 b_1	極の中心周波数 f_1[Hz] と帯域幅 b_1[Hz].	[N/A]
–**z**	f_2 b_2	零の中心周波数 f_2[Hz] と帯域幅 b_2[Hz].	[N/A]

EXAPMLE

極の中心が 2000Hz 帯域幅 200Hz のフィルタのインパルス応答を出力する:

```
impulse | df2 -p 2000 200
```



NAME

dfs – 直接形のディジタルフィルタ

SYNOPSIS

dfs [**–a** K $a(1)$ $\cdots$ $a(M)$] [**–b** $b(0)$ $b(1)$ $\cdots$ $b(N)$] [**–p** ***pfile***] [**–z** ***zfile***] [***infile***]

DESCRIPTION

dfs は，ファイル *infile* (省略時には標準入力) から読み込まれたデータを，オプションで指定された係数をもつ標準形のディジタルフィルタでフィルタリングし，標準出力に出力します．フィルタの伝達関数は，

$$H(z) = K \frac{\sum_{n=0}^{N} b(n) z^{-n}}{1 + \sum_{m=1}^{M} a(m) z^{-m}}$$

で与えられます．

データの形式は，入出力ともに float 形式です．

OPTIONS

–a	K $a(1) \cdots a(M)$	伝達関数の分母多項式の係数．但し，K は伝達関数のゲインを表す．	[N/A]
–b	$b(0)$ $b(1) \cdots b(N)$	伝達関数の分子多項式の係数．	[N/A]
–p	$pfile$	伝達関数の分母多項式の係数が書かれたファイル．ファイルの内容は float 形式で次のように与える． $K, a(1), \ldots, a(M)$	[NULL]
–z	$zfile$	伝達関数の分子多項式の係数が書かれたファイル．ファイルの内容は float 形式で次のように与える． $b(0), b(1), \ldots, b(N)$	[NULL]

–a と**–p** オプションがともに省略されると，分母多項式，K ともに 1 として扱われます．また，**–b** と**–z** オプションがともに省略されると，分子多項式は 1 として扱われます．

EXAMPLE

伝達関数

$$H(z) = \frac{1 + 2z^{-1} + z^{-2}}{1 + 0.9z^{-1}}$$

をもつフィルタのインパルス応答を出力させる:

```
impulse | dfs -a 1 0.9 -b 1 2 1 | dmp
```

float 形式のファイル *data.p, data.z* で指定された係数をもつディジタルフィルタの周波数応答を画面にプロットさせる:

```
impulse | dfs -p data.p -z data.z | spec | fdrw | xgr
```

data.p, data.z は，コマンド *x2x* などでつくることができます.

NAME

dmp – バイナリファイルのダンプ

SYNOPSIS

dmp [–**n** N] [–**l** L] [+***type***] [%***form***] [***infile***]

DESCRIPTION

dmp は，指定されたファイルから読み込まれたデータに番号をふってダンプします．ファイルが指定されなかった場合，データは標準入力から読み込まれます．

OPTIONS

–**n**	N	データの次数．ふられる番号は，$0 \sim N$ となる．指定しない場合は，$N = \infty$ と等価.	[EOD]
–**l**	L	データのブロック長．ふられる番号は，$1 \sim L$ となる．指定しない場合は，$L = \infty$ と等価.	[EOD]
$+t$		入力データの形式.	[f]
		c char 型 (1byte) s short 型 (2bytes) i int 型 (4bytes) l long 型 (4bytes) f float 型 (4bytes) d double 型 (8bytes)	
%form		出力のフォーマット指定. C 言語の printf 関数の引数の形で出力のフォーマットを指定する．入力データの型に合わせる必要がある.	[N/A]

EXAMPLE

float 形式のファイル *data.f* にあるデータをダンプする:

```
dmp +f data.f
```

例えば，ファイル *data.f* に，float 形式の数値データ

$$1, 2, 3, 4, 5, 6, 7$$

が入っていた場合,

```
0       1
1       2
```

```
2       3
3       4
4       5
5       6
6       7
```

と画面（標準出力）に出力されます.

同じデータをブロック次数を指定してダンプする:

```
dmp -n 2 +f data.f
```

この場合,

```
0       1
1       2
2       3
0       4
1       5
2       6
0       7
```

と表示されます.

ディジタルフィルタのユニットパルス応答を画面にダンプする:

```
impulse | dfs -a 1 0.9 | dmp
```

printf の%e フォーマットで，正弦波をダンプする:

```
sin -p 30 | dmp %e
```

小数点以下を 3 桁にする:

```
sin -p 30 | dmp %.3e
```

SEE ALSO

x2x, fd

NAME

ds – サンプリングレート変換（ダウンサンプリング）

SYNOPSIS

ds [–**s** S] [***infile***]

DESCRIPTION

ダウンサンプリングを行ないます.

データ形式は入力，出力とも float 形式です．フィルタ係数は次のものが用いられます.

$S = 21$	\$SPTK/lib/lpfcoef.2to1
$S = 43$	\$SPTK/lib/lpfcoef.4to3
$S = 52, s = 54$	\$SPTK/lib/lpfcoef.5to2up
	\$SPTK/lib/lpfcoef.5to2dn
	(\$SPTK はインストールしたディレクトリ)

なお，ファイルタ係数のファイルは ASCII 形式となっています.

OPTIONS

–**s** S 変換タイプ. [21]

$S = 21$ 比率 2 : 1 でダウンサンプリング
$S = 43$ 比率 4 : 3 でダウンサンプリング
$S = 52$ 比率 5 : 2 でダウンサンプリング
$S = 54$ 比率 5 : 4 でダウンサンプリング

EXAMPLE

float 形式の標本化周波数 16kHz の音声データ *data.16* を標本化周波数 8kHz にダウンサンプリングし， *data.8* に出力する:

```
ds data.16 > data.8
```

NAME

echo2 – 標準エラーへの出力

SYNOPSIS

echo2 [**–n**] [**argument**]

DESCRIPTION

echo2 は，引数を標準エラーに出力します.

OPTIONS

–n 出力後に改行しません.

EXAMPLE

standard error に改行せずに，”error!” と出力する.

```
echo2 -n "error!"
```

NAME

excite – 音声合成のための音源生成

SYNOPSIS

excite [–**p** P] [–**i** I] [–**n**] [–**s** S] [***infile***]

DESCRIPTION

ピッチ周期ファイルから，有声音に対してはパルス列，無声音に対しては M 系列またはガウス雑音を生成し，標準出力に出力します.

データ形式は入力，出力とも float 形式です.

OPTIONS

–**p**	P	フレーム周期.	[100]
–**i**	I	補間周期.	[1]
–**n**		無声音をガウス雑音で励振させる.	[FALSE]
		指定しない場合，M 系列で励振させます.	
–**s**	S	ガウス雑音の場合，ランダムの初期化の seed	[1]

EXAMPLE

float 形式のピッチデータ *data.p* から励振源を作成し，あらかじめ線形予測分析によって求めておいた線形予測係数 *data.lpc* により合成音 *data.syn* を出力する:

```
excite < data.p | poledf data.lpc > data.syn
```

無声音をガウス雑音で励振させる場合

```
excite -n < data.p | poledf data.lpc > data.syn
```

SEE ALSO

poledf

NAME

extract – あるインデックスに分類された入力ベクトルの抽出

SYNOPSIS

extract [–**l** *L*] [–**i** *I*] ***indexfile*** [***infile***]

DESCRIPTION

extract は infile から index *I* に分類された入力ベクトルを標準出力に出力します.

OPTIONS

–**l**	*L*	ベクトルの次数.	[10]
–**i**	*I*	コードブックインデックス.	[0]

EXAMPLE

float 形式の 10 次のベクトルのファイル *data.v* をベクトル量子化した結果である *data.idx* 中で index が 0 となるベクトルを抽出し，*data.ex* に出力する:

```
extract -i 0 data.idx data.v > data.ex
```

SEE ALSO

ivq, vq

NAME

fd – バイナリデータのダンプ

SYNOPSIS

fd [**−a** *A*] [**−n** *N*] [**−m** *M*] [***−ent***] [***+type***] [***%form***] [***infile***]

DESCRIPTION

指定されたファイルの全データを，指定されたフォーマットでダンプします．ファイルが指定されなかった場合，データは標準入力から読み込まれます．

OPTIONS

−a	*A*	アドレスつきで出力． *A* は最初のアドレス．	[0]
−n	*N*	番号付きで出力． *N* は最初の番号．	[0]
−m	*M*	番号付けを行なう場合に *M* による modulo をとる．	[EOF]
−ent		*ent* は数字で，1 行に出力するデータの個数．	[0]
+t		入力データの形式．	[c]

c	char 型 (1byte)	s	short 型 (2bytes)
i	int 型 (4bytes)	l	long 型 (4bytes)
f	float 型 (4bytes)	d	double 型 (8bytes)

%form	出力のフォーマット指定． C 言語の printf 関数の引数の形で出力のフォーマットを指定する．ただし，入力データの形式が f,d の時のみ有効．	[N/A]

EXAMPLE

音声データ sample.wav をアドレス付きでダンプする．

```
fd -a sample.wav
```

結果

```
000000  52 49 46 46 9a 15 00 00 57 41 56 45 66 6d 74 20 |RIFF....WAVEfmt |
000010  10 00 00 00 01 00 01 00 40 1f 00 00 40 1f 00 00 |........@...@...|
000020  01 00 08 00 64 61 74 61 76 15 00 00 8a 8a 8f 99 |....datav.......|
```

⋮

SEE ALSO

dmp

NAME

fdrw – 入力データのグラフを描く

SYNOPSIS

fdrw [–**F** *F*] [–**R** *R*] [–**W** *W*] [–**H** *H*] [–**o** *xo yo*] [–**g** *G*] [–**m** *M*] [–**l** *L*] [–**p** *P*] [–**n** *N*] [–**t** *T*] [–**y** *ymin ymax*] [–**z** *Z*] [–**b**] [***infile***]

DESCRIPTION

ファイル *infile* (省略時には標準入力) から読み込まれた float 形式データ列を y 座標として直線で結びます．x 座標は等間隔になります．

出力はプロッタ（グラフテック社 FP5301）のコマンド列です．

OPTIONS

–**F**	*F*	グラフを描く倍率.	[1]
–**R**	*R*	グラフを描く角度.	[0]
–**W**	*W*	グラフの幅 (×100mm).	[1]
–**H**	*H*	グラフの高さ (×100mm).	[1]
–**o**	*xo yo*	グラフを描く位置． 描画域の左下を原点として，グラフの左下の位置を mm 単位で指定.	[20 25]
–**g**	*G*	グラフの枠の形式 (0 ∼ 2)．＜fig 参照＞.	[1]
–**m**	*M*	線の種類 (1 ∼ 5). 1: 実線 2: 点線 3: 一点鎖線 4: 破線 5: 長破線	[0]
–**l**	*L*	$ltype > 1$ の場合の線のピッチを指定．L は mm 単位.	[0]
–**p**	*P*	ペン番号 (1 ∼ 10).	[1]
–**n**	*N*	1 回に描くデータ数．入力データ数が N より多い場合は，N データ毎に繰り返し線を描く.	[0]
–**t**	*T*	座標軸を回転．$T = -1$ では，グラフの左上を原点とし，下向きに x 軸，左向きに y 軸をとる．また，$T = 1$ では，グラフの右下を原点とし，左向きに x 軸，上向きに y 軸をとる.	[0]
–**y**	*ymin ymax*	y 軸のスケーリング.	[-1 1]
–**z**	*Z*	N データ毎に繰り返し線を描く場合，y 座標を Z(mm) ずらしながら描く.	[0]

–b 棒グラフ状に描くことを指定. [FALSE]

x 座標のスケーリングは最初のデータが図の左端，N 番目のデータが右端となります．**–n** オプションを省略して N を与えていない場合，入力データ数が 5000 以下ならば最終データが右端となるようにスケーリングされます．5000 を越える場合は $N = 5000$ として描きます．
–y オプションを省略した場合，入力データの最小値が $ymin$，最大値が $ymax$ となります．

EXAMPLE

ディジタルフィルタのインパルス応答を X ウィンドウ上に描く:

```
impulse | dfs -a 1 0.8 0.5 | fdrw -H 0.3 | xgr
```

グラフの幅は 10cm，高さは 3cm になります．

ディジタルフィルタの対数振幅応答を X ウィンドウ上に描く:

```
impulse | dfs -a 1 0.8 0.5 | spec | fdrw -y -60 40 | xgr
```

y 軸は -60dB から 40dB となります．

連続するフレームのスペクトルを X ウィンドウ上に描く:

```
fig -g 0 -w 0.4 << EOF
    x 0 5
    xscale 0 1 2 3 4 5
    xname "FREQUENCY (kHz)"
EOF
spec < data |\
fdrw -w 0.4 -h 0.2 -g 0 -n 129 -y -30 30 -z 3 |\
xgr
```

画面と同じものをレーザプリンタに出力するコマンドとして *psgr* があります．*fdrw* の出力はプロッタ（グラフテック社 FP5301）のコマンド列ですので，プロッタに直接出力を送って，図を描かせることもできます．

SEE ALSO

fig, xgr, psgr

NAME

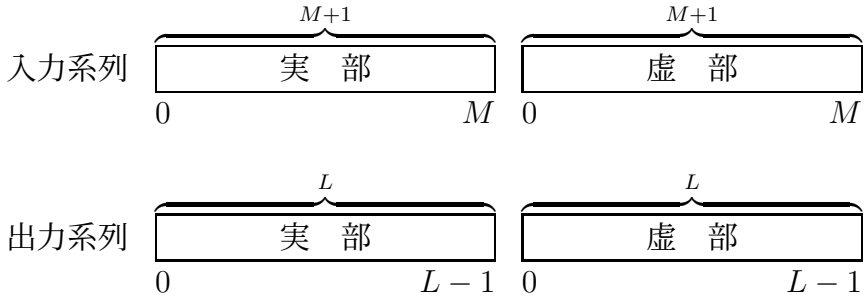
fft – 複素数列の高速フーリエ変換

SYNOPSIS

fft [–**l** *L*] [–**m** *M*] [–**{ A** | **R** | **I** | **P } **] [***infile***]

DESCRIPTION

複素数列を *infile* から読み込み，FFT アルゴリズムにより DFT を実行します．*infile* の指定がないときには，標準入力からデータが読み込まれます．データ形式は入出力とも float 形で，入出力データの順序は次の通りです．



OPTIONS

–**l**	*L*	フーリエ変換のサイズ．2 のべき乗で指定.	[256]
–**m**	*M*	複素数列の次数.	[L-1]
–**A**		振幅スペクトルを出力.	[FALSE]
–**R**		実部のみを出力.	[FALSE]
–**I**		虚部のみを出力.	[FALSE]
–**P**		パワースペクトルを出力.	[FALSE]

EXAMPLE

float 形式のファイル *data.f* にある複素数列（実部 256 点，虚部 256 点）の DFT の振幅を求め，*data.dft* に出力する:

```
fft data.f -l 256 -A > data.dft
```

SEE ALSO

fftr, spec, phase

NAME

fftcep – FFT ケプストラム

SYNOPSIS

fftcep [–**m** M] [–**l** L] [–**j** J] [–**k** K] [–**e** E] [***infile***]

DESCRIPTION

FFT ケプストラム係数 $c(m)$ を標準出力に出力します．入力は窓掛けされた長さ L の時系列

$$x(0), x(1), \ldots, x(L-1)$$

です．

データ形式は入力，出力とも float 形式です．

繰り返し回数 J および、加速係数 K を指定した場合，改良ケプストラム法によりケプストラム係数を計算します．

OPTIONS

–**m**	M	分析次数.	[25]
–**l**	L	フレーム長.	[256]
–**j**	J	繰り返し回数.	[0]
–**k**	K	加速係数.	[0.0]
–**e**	E	振幅スペクトルの対数をとる際に足し込む小さな値.	[0.0]

EXAMPLE

float 形式の音声データ *data.f* を分析し，*data.cep* にケプストラム係数を得る:

```
frame < data.f | window | fftcep > data.cep
```

SEE ALSO

uels, ceps

NAME

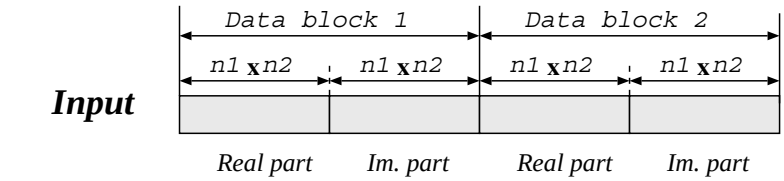
fft2 – 複素数列の 2 次元高速フーリエ変換

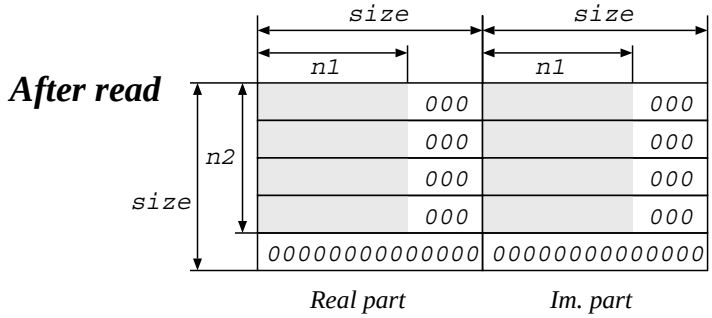
SYNOPSIS

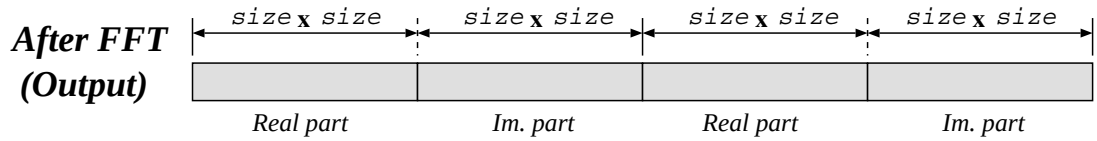
fft2 [–**l** *L*] [–**m** M_1 M_2] [–**t**] [–**c**] [–**q**] [–**{ A** | **R** | **I** | **P }]**
[***infile***]

DESCRIPTION

fft2 は，複素数で与えられるデータ列を *infile* から読み込んで 2 次元フーリエ変換を行ない，結果を標準出力に出力します．入出力のデータフォーマットは以下の通りです．





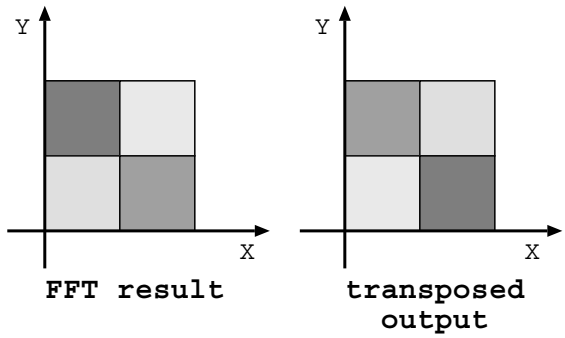


OPTIONS

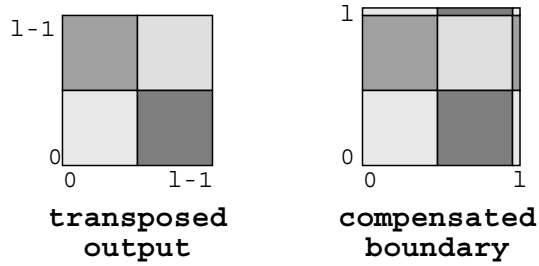
–l L フーリエ変換のサイズ． 2 のべき乗で指定. [64]

–m M_1 M_2 データの次数を $M_1 \times M_2$ で指定．ファイルサイズ k が $64^2 \times 2$ よりも小さい場合，$\sqrt{k \div 2}$ が整数ならば $M_1 = M_2 = \sqrt{k \div 2}$ とし，整数でない場合は標準エラー出力にエラーメッセージを出力してから処理を終了. [64, M_1]

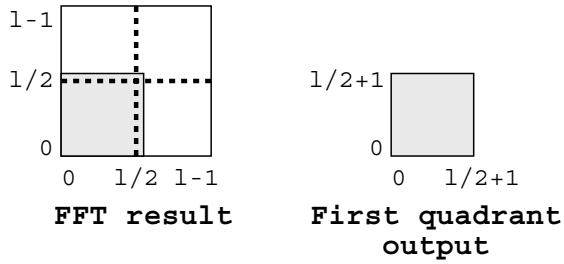
–t FFT の結果を transpose して出力. [FALSE]



–c transpose する際，境界の 1 データを反対側から持ってきて，$(L+1) \times (L+1)$ 個のデータを出力. [FALSE]



–q FFT の結果の最初の 1/4 のデータのみを出力．この際 c オプションと同様に境界の補償を行ない，$(\frac{L}{2}+1) \times (\frac{L}{2}+1)$ 個のデータを出力． [FALSE]



–A 振幅スペクトルを出力． [FALSE]
–R 実部のみを出力． [FALSE]
–I 虚部のみを出力． [FALSE]
–P パワースペクトルを出力． [FALSE]

EXAMPLE

float 形式のファイル *data.f* にある 2 次元複素数列の DFT の振幅を求め，*data.dft* に出力する:

```
fft2 -A data.f > data.dft
```

SEE ALSO

fft, fftr2, ifft

NAME

fftr – 実数列の高速フーリエ変換

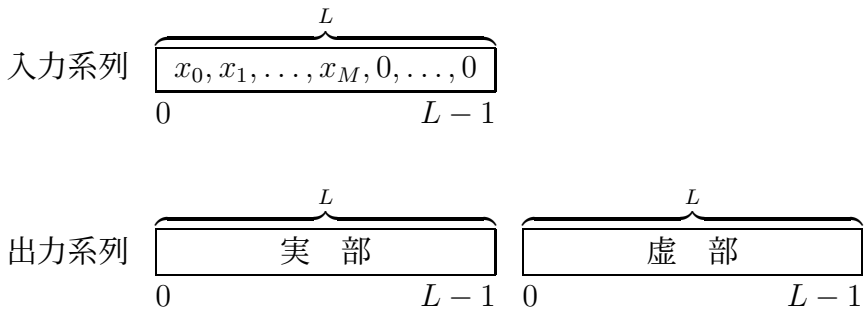
SYNOPSIS

fftr [–**l** L] [–**m** M] [–**{ A** | **R** | **I** | **P }]** [–**H**] [***infile***]

DESCRIPTION

実数列を *infile* から読み込み，FFT アルゴリズムにより DFT を実行します．*infile* の指定がないときには，標準入力からデータが読み込まれます．データ形式は入出力とも float 形です．

–**m** オプションで M の指定がなく，かつ入力系列長が FFT サイズより短い場合，入力系列に 0 を付加して FFT を実行します．



OPTIONS

–**l**	L	フーリエ変換のサイズ．2 のべき乗で指定．	[256]
–**m**	M	実系列の次数．	[L-1]
–**A**		振幅スペクトルを出力．	[FALSE]
–**R**		実部のみを出力．	[FALSE]
–**I**		虚部のみを出力．	[FALSE]
–**P**		パワースペクトルを出力．	[FALSE]
–**H**		出力系列を $(L/2+1)$ 個とする．	[FALSE]

EXAMPLE

正弦波にブラックマン窓をかけ，DFT 行った結果の振幅をプロットする:

```
sin -t 30 | window | fftr -A | fdrw | xgr
```

SEE ALSO

fft, spec, phase

NAME

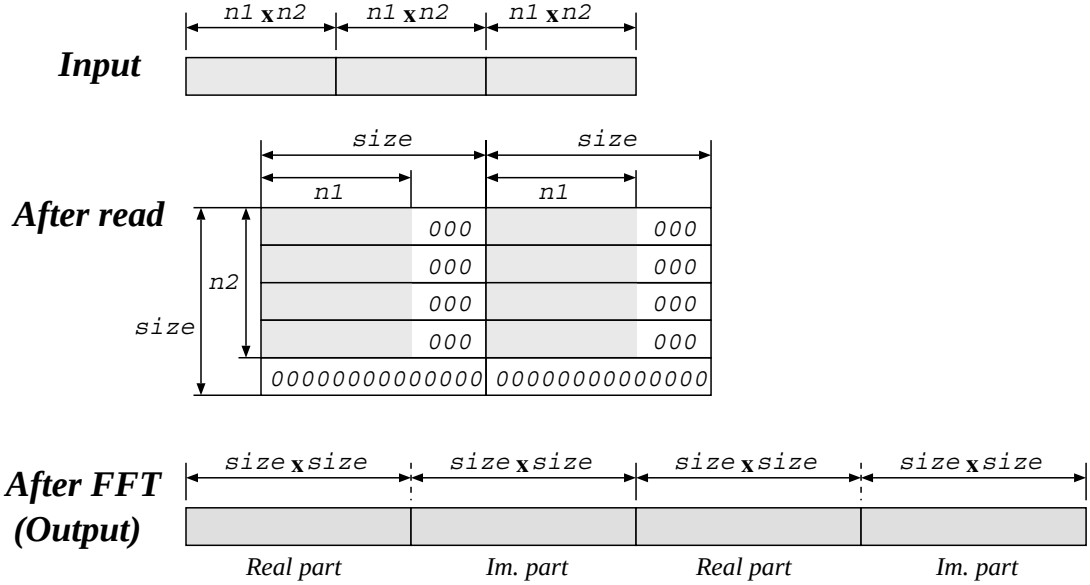
fftr2 – 実数列の 2 次元高速フーリエ変換

SYNOPSIS

fftr2 [**–l** L] [**–m** M_1 M_2] [**–t**] [**–c**] [**–q**] [**–{ A** | **R** | **I** | **P }]** [***infile***]

DESCRIPTION

fftr2 は，float 型で与えられる実数のデータ列を *infile* から読み込んで 2 次元フーリエ変換を行ない，結果を標準出力に出力します．*infile* が指定されない時は標準入力から読み込みます．入出力のデータフォーマットは以下の通りです．



OPTIONS

–l	L	フーリエ変換のサイズ． 2 のべき乗で指定．	[64]
–m	M_1 M_2	データの次数を $M_1 \times M_2$ で指定．ファイルサイズ k が 64^2 よりも小さい場合，$\sqrt{k}$ が整数ならば $M_1 = M_2 = \sqrt{k}$ とし，整数でない場合は標準エラー出力にエラーメッセージを出力してから処理を終了．	[64, M_1]
–t		FFT の結果を transpose して出力．＜fft2 参照＞．	[FALSE]
–c		transpose する際，境界の 1 データを反対側から持ってきて，$(L+1) \times (L+1)$ 個のデータを出力．＜fft2 参照＞．	[FALSE]

–q	FFT の結果の最初の 1/4 のデータのみを出力．この際 c オプションと同様に境界の補償を行ない，$(\frac{L}{2}+1)\times(\frac{L}{2}+1)$ 個のデータを出力．＜fft2 参照＞.	[FALSE]
–A	振幅スペクトルを出力.	[FALSE]
–R	実部のみを出力.	[FALSE]
–I	虚部のみを出力.	[FALSE]
–P	パワースペクトルを出力.	[FALSE]

EXAMPLE

float 形式のファイル *data.f* にある 2 次元実数列の DFT の振幅を求め，*data.dft* に出力する:

```
fftr2 -A data.f > data.dft
```

SEE ALSO

fft, fft2, ifft

NAME

fig – 図を作成する

SYNOPSIS

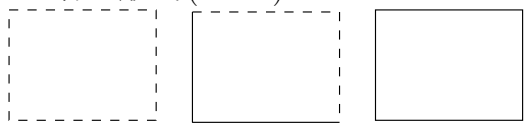
fig [–**F** *F*] [–**R** *R*] [–**W** *W*] [–**H** *H*] [–**o** *xo yo*] [–**g** *G*] [–**p** *P*] [–**s** *S*] [–**f** *file*] [–**t**] [***infile***]

DESCRIPTION

ファイル *infile* (省略時は標準入力) から与えられるコマンドによりグラフを描きます．UNIX 標準コマンドの graph と同様な機能がある他，グラフの座標軸，目盛り，名前を入れる機能があります．出力はプロッタのコマンド列です．そのため，X 端末に出力する場合はコマンド xgr，ポストスクリプトに出力する場合はコマンド psgr を使用して下さい．

OPTIONS

–**F**	*F*	グラフを描く倍率.	[1]
–**R**	*R*	グラフを描く角度.	[0]
–**W**	*W*	グラフの幅 (×100mm).	[1]
–**H**	*H*	グラフの高さ (×100mm).	[1]
–**o**	*xo yo*	グラフを描く位置． 描画域の左下を原点として，グラフの左下の位置を mm で指定.	[20 20]
–**g**	*G*	グラフの枠の形式 (0 ∼ 2).	[2]
–**p**	*P*	ペン番号 (1 ∼ 10).	[1]
–**s**	*S*	文字サイズ (1 ∼ 4).	[1]
–**f**	*file*	コマンド実行時，最初に読み込むファイル．(*infile* より優先).	[NULL]
–**t**		*x* 軸と *y* 軸を入れ換える.	[FALSE]



EXAMPLE

data.fig のグラフを X 端末に出力する.

```
fig data.fig |xgr
```

data.fig のグラフをポストスクリプトにしたデータを ghostview で見る.

```
fig data.fig | psgr | ghostview -
```

USAGE

コマンド

入力データはコマンド行とデータ行にわけることができます. コマンド行は, グラフの目盛り, 名前, スケーリング等を指定します. データ行は $(x\ y)$ 座標ペアを指定します. コマンド行によって指定された数値は, 次のコマンド行による設定がなされるまで有効です.

コマンド行

x [mel α] $xmin$ $ymax$ [xa]
y [mel α] $xmin$ $ymax$ [ya]

x 軸, y 軸のスケーリングを設定します.
xa, ya は目盛りを打つ軸の位置を指定します. 省略された場合, $xa = xmin$, $ya = ymin$ となります.
mel α (α は数字で, 例えば mel 0.35) を指定すると, 1 次オールパスフィルタの位相特性により周波数変換した目盛りとみなします.

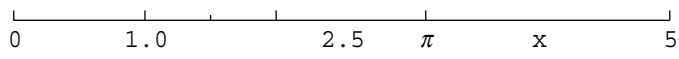
xscale $x_1\ x_2\ x_3\ \cdots$
yscale $y_1\ y_2\ y_3\ \cdots$

x 軸, y 軸をそれぞれ描き, 指定された点 $x_1, x_2, x_3, \cdots$ および $y_1, y_2, y_3, \cdots$ に目盛り, 数値を書きます. 但し, 非数字+数値 (例えば '2,*3.14 等) には, 次の機能があります.

s	半分の高さの目盛りを描きます.
\	数値だけを書きます.
@	なにも書きません (目盛りの位置のみ指定).
その他	目盛りだけを書きます.

" " で囲まれた文字列が出現すると, 直前の目盛りの位置にその文字列を書きます. 文字列中の特種文字の取り扱いについては, コマンド行 x/yname 項を参照して下さい.
(例)

```
x 0 5
xscale 0 1.0 s1.5 '2 \2.5 '3.14 "\pi" @4 "x" 5
```

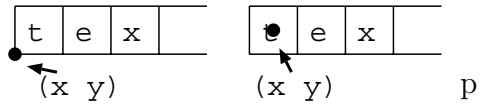


xname "*text*"
yname "*text*"

x 軸, y 軸に名前 (文字列 *text*) を入れます. *text* は必ず" " で囲んで下さい. *text* 中には"\" に続く, TEX の数式環境中の特種文字を使用することができます. また, TEX と同様にして上つき文字, 下つき文字を使用することができます.

print x y "*text*" [*th*]
printc x y "*text*" [*th*]

(x y) の位置に文字列 *text* を書きます. *th* は文字列の角度 (deg) を指定します.
文字列を書く位置:



print printc

title x y "*text*" [*th*]
titlec x y "*text*" [*th*]

print(c) と同じです. 但し, (x y) は mm 単位で表した絶対座標です. 原点は左下隅です.

csize h [w]

x/yscale, x/yname, print/c, title/c 等で書く文字の高さ h と幅 w を mm 単位で指定します. w を省略した場合 w = h となります. デフォルトは –**s** オプションの値に従って,

–**s**	w	h
1	2.5	2.2
2	5	2.6
3	2.5	4.4
4	5	4.4

です.

pen *penno*

使用するペン *penno* を選びます. $1 \leqq penno \leqq 10$. 付録参照.

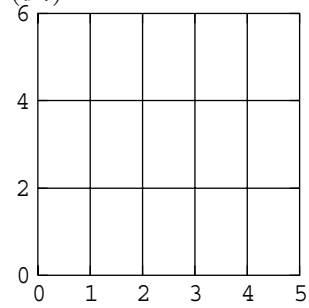
line *ltype* [*lpt*]

座標データ間を結ぶ線の種類 *ltype* とピッチ *lpt* を指定します. $0 \leqq ltype \leqq 5$. *lpt* は mm 単位です. *ltype*=0:線を引かない. 1:実線. 2:点線. 3:1 点鎖線. 4:破線. 5:長破線.
付録参照.

xgrid x_1 x_2 $\cdots$
ygrid y_1 y_2 $\cdots$

x_1 x_2 $\cdots$, y_1 y_2 $\cdots$ の位置に，それぞれ縦と横のグリッドを引きます．

(例)




```
x 0 5
y 0 6
xscale 0 1 2 3 4 5
yscale 0 2 4 6

xgrid 1 2 3 4
ygrid 2 4
```

mark *label* [*th*]

データ行で与えられる座標に文字列 *label* を書きます．*th* は文字列を書く角度 (deg) です．*label* に \0 を指定すると解除します．マーク，特殊文字の書き方はデータ行の *label* 項を参照して下さい．

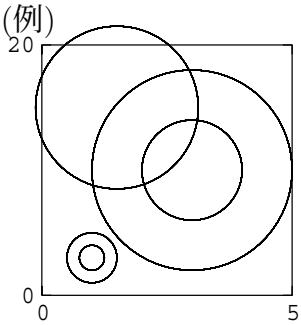
hight h [w]
italic th

データ行で *label* を指定した場合に書く文字の大きさ h(mm)，間隔 w(mm) とイタリック指定 th(deg) をします．

circle x y r_1 r_2 $\cdots$
xcircle x y r_1 r_2 $\cdots$
ycircle x y r_1 r_2 $\cdots$

(x y) を中心とし半径が r_1 r_2 $\cdots$ の円を描きます．但し，r_x の単位は circle は mm，xcircle はグラフの x スケール，ycircle はグラフの y スケールです．

(例)



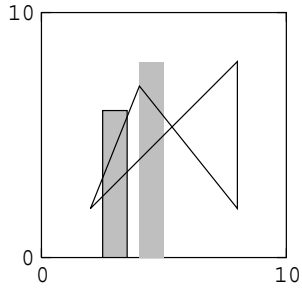

```
x 0 5
y 0 20
xscale 0 5
yscale 0 20

xcircle 3 10 1 2
ycircle 1 3 1 2
circle 1.5 15 13
```

box $x_0\ y_0\ x_1\ \ y_1\ [\ x_2\ y_2\ \cdots\ ]$
paint *type*

$(x_0\ y_0)$, $(x_1\ y_1)$ を結ぶ直線を対角線とする長方形を paint で指定された *type* で描きます．但し， $x_2\ y_2\ \cdots$ が指定された場合， $(x_0\ y_0)$,$(x_1\ y_1)$, $(x_2\ y_2)$,$\cdots$ を結ぶ多角形を描きます (paint は *type* 1 以外指定しないで下さい)．paint のデフォルトは 1 です．

(例)



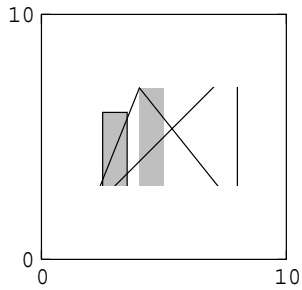

```
x 0 10
y 0 10
xscale 0 10
yscale 0 10

paint 18
box 2.5 0 3.5 6
paint -18
box 4 0 5 8
paint 1
box 2 2 8 8 8 2 4 7
```

clip $x_0\ y_0\ x_1\ \ y_1$

$(x_0\ y_0)$, $(x_1\ y_1)$ を結ぶ直線を対角線とする長方形の範囲の中だけ描きます．$(x_0\ y_0)$, $(x_1\ y_1)$ が省略された時，clip を解除します．

(例)




```
x 0 10
y 0 10
xscale 0 10
yscale 0 10

clip 2 3 9 7
paint 18
box 2.5 0 3.5 6
paint -18
box 4 0 5 8
paint 1
box 2 2 8 8 8 2 4 7
```

any comment

コメント行です．動作に影響を及ぼしません．

データ行

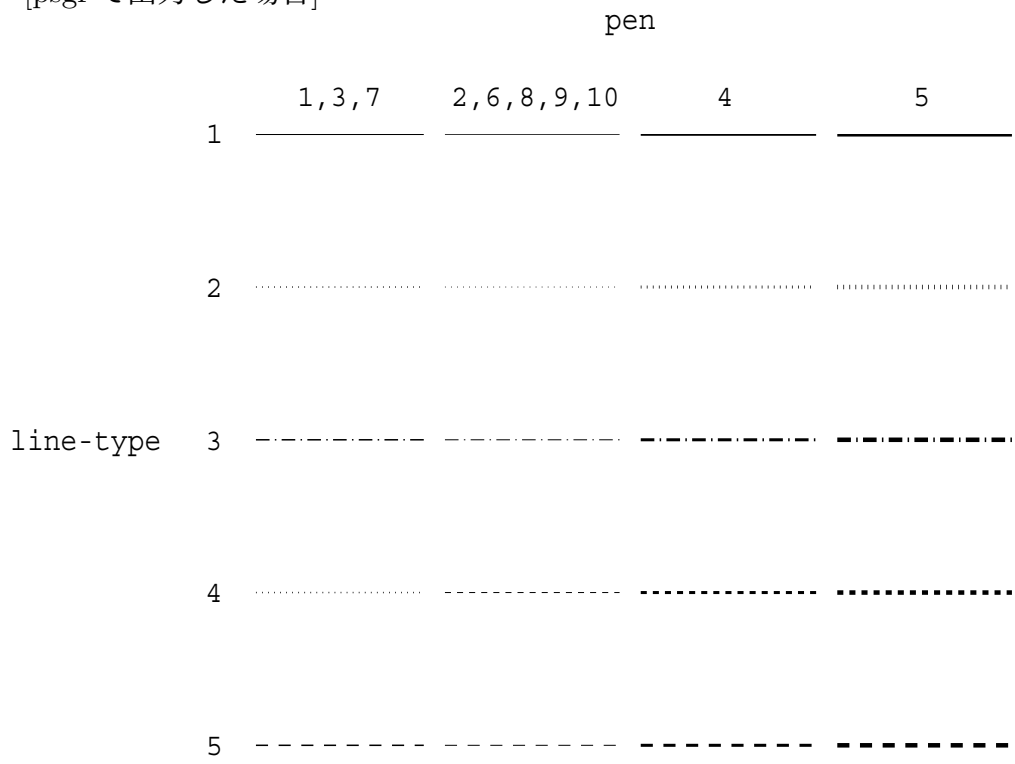
x y [*label* [*th*]]	(x y) 座標は，コマンド行 x, y で指定された数値でスケーリングされます．*label* の位置に文字列を書くと，その文字列が (x y) の位置に書かれます．*label* は空白文字を含んではいけません．コマンド行 mark で *label* が指定されている場合は，この座標に限り指定の *label* に置き換わります．*th* は角度を与えます． *label* 文字列に \n 0 ≦ n ≦ 15 を指定すると，対応する番号のマークを描きます (マークの種類は付録を参照して下さい)．マーク番号に– (マイナス) を付けると，マークの中心を線で結びます．通常は，マークと線が重ならないように描きます． $n = 16$(\16) で コマンド行 hight で指定された h を直径とする小円を描きます．また，$n > 32$ で指定コード番号の ASCII 文字または特殊文字を描きます．
eod EOD	データの区切りです．この前後の座標間には線を引きません．

付録

- mark 又はデータ行で *label* が指定された場合に出力されるマーク:

0	1	2	3	4	5	6	7
	•	×	□	△	○	◊	×
8	9	10	11	12	13	14	15
+	⊗	⊕	■	▲	●	♦	*

- pen と line のタイプ:
  [psgr で出力した場合]

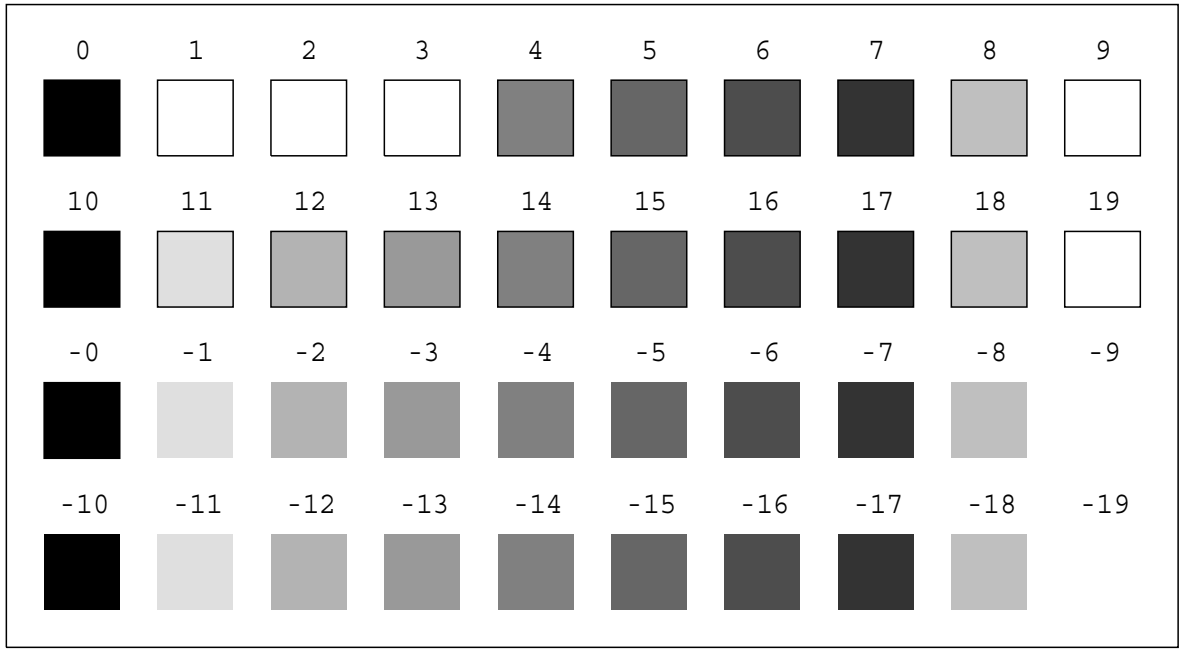


(注) pen のタイプは出力するプリンタに依存します. (このページを出力してみて下さい)

[xgr で出力した場合]
下表に示す色で表示されます.

pen タイプ	1	2	3	4	5	6	7	8	9	10
色	黒	青	赤	緑	ピンク	オレンジ	エメラルド	灰	茶	紺

- paint のタイプ:



(注) $1 \sim 3$ は枠のみ, $-9, -19$ は枠無しの白です.

NAME

frame – フレームの切り出し

SYNOPSIS

frame [–**l** L] [–**n**] [–**p** P] [+***type***] [***infile***]

DESCRIPTION

frame は、指定されたファイルから、入力データをフレーム周期 P，フレーム長 L で切り出します．入力データを $x(0), x(1), \ldots, x(T)$ とした時，出力は

$$
\begin{array}{ccccccccc}
0 & , & 0 & , & \ldots & , & x(0) & , \ldots , & x(L/2) \\
x(P-L/2) & , & x(P-L/2+1) & , & \ldots & , & x(P) & , \ldots , & x(P+L/2) \\
x(2P-L/2) & , & x(2P-L/2+1) & , & \ldots & , & x(2P) & , \ldots , & x(2P+L/2) \\
 & & & & & & \vdots & &
\end{array}
$$

となります．

OPTIONS

–**l**	L	切り出すフレーム長．	[256]
–**p**	P	フレーム周期．	[100]
–**n**		開始点をフレームの中心にしない．	[FALSE]
$+t$		入出力データの形式．	[f]

c	char 型 (1byte)	s	short 型 (2bytes)
i	int 型 (4bytes)	l	long 型 (4bytes)
f	float 型 (4bytes)	d	double 型 (8bytes)

EXAMPLE

float 形式の ファイル *data.f* からフレーム周期 80 でデータを切りだし，線形予測分析を行い，予測係数を *data.lpc* に出力する:

```
frame -p 80 < data.f | window | lpc > data.lpc
```

SEE ALSO

bcp, x2x, bcut, window

NAME

freqt – 周波数変換

SYNOPSIS

freqt [**–m** M_1] [**–M** M_2] [**–a** A_1] [**–A** A_2] [***infile***]

DESCRIPTION

ファイル中の M_1 次の最小位相数列

$$c_{\alpha_1}(0), c_{\alpha_1}(1), \ldots, c_{\alpha_1}(M_1)$$

に対して周波数変換

$$\alpha = (\alpha_1 - \alpha_2)/(1 - \alpha_1\alpha_2)$$

$$c^{(i)}_{\alpha_2}(m) = \left\{ \begin{array}{ll} c_{\alpha_1}(-i) + \alpha\, c^{(i-1)}_{\alpha_2}(0), & m = 0 \\ (1-\alpha^2)\, c^{(i-1)}_{\alpha_2}(0) + \alpha\, c^{(i-1)}_{\alpha_2}(1), & m = 1 \\ c^{(i-1)}_{\alpha_2}(m-1) + \alpha \left(c^{(i-1)}_{\alpha_2}(m) - c^{(i)}_{\alpha_2}(m-1) \right), & m = 2, \ldots, M_2 \end{array} \right\}, \quad i = -M_1, \ldots, -1, 0$$

を行い，M_2 次の周波数変換した数列

$$c^{(0)}_{\alpha_2}(0), c^{(0)}_{\alpha_2}(1), \ldots, c^{(0)}_{\alpha_2}(M_2)$$

を出力します.

データ形式は入力，出力とも float 形式です.

OPTIONS

–m	M_1	入力される最小位相数列の次数.	[25]
–M	M_2	出力される周波数変換された数列の次数.	[25]
–a	A_1	入力数列の周波数圧縮パラメータ α_1.	[0]
–A	A_2	出力数列の周波数圧縮パラメータ α_2.	[0.35]

EXAMPLE

float 形式の線形予測係数のファイル *data.lpc* から 30 次の LPC メルケプストラムを *data.lpcmc* に出力する:

```
lpc2cep < data.lpc | freqt -m 30 > data.lpcmc
```

SEE ALSO

mgc2mgc

NAME

gc2gc – 一般化対数変換

SYNOPSIS

gc2gc [–**m** M_1] [–**g** G_1] [–**n**] [–**u**]
[–**M** M_2] [–**G** G_2] [–**N**] [–**U**] [***infile***]

DESCRIPTION

べきパラメータ γ_1 の一般化ケプストラムからべきパラメータ γ_2 の一般化ケプストラムを一般化対数変換の再帰式により求め，標準出力に出力します.

データ形式は入力，出力とも float 形式です.

一般化対数変換は,

$$\begin{aligned} c_{\gamma_2}(m) &= c_{\gamma_1}(m) + \sum_{k=1}^{m-1} \frac{k}{m}(\gamma_2 c_{\gamma_1}(k) c_{\gamma_2}(m-k) \\ &\qquad -\gamma_1 c_{\gamma_2}(k) c_{\gamma_1}(m-k)), \quad m > 0 \end{aligned}$$

の再帰式で与えられます．この変換式は，$\gamma_1 = -1, \gamma_2 = 0$ とすれば線形予測係数から LPC ケプストラムの再帰式，$\gamma_1 = 0, \gamma_2 = 1$ とすればケプストラムから最小位相インパルス応答への再帰式などと等価になります.

γ を掛けた形で入出力する場合は，$c_\gamma(m)$ が正規化されていない場合は,

$$1 + \gamma c_\gamma(0), \gamma c_\gamma(1), \ldots, \gamma c_\gamma(M)$$

とし，正規化されている場合は

$$K_\alpha, \gamma c'_\gamma(1), \ldots, \gamma c'_\gamma(M)$$

とします.

OPTIONS

–**m**	M_1	入力される一般化ケプストラムの次数.	[25]
–**g**	G_1	入力される一般化ケプストラムのべきパラメータ γ_1. ただし，$G_1 > 1.0$ のときは $\gamma_1 = -1/G_1$.	[0]
–**n**		入力を正規化されたケプストラムと見なす.	[FALSE]
–**u**		入力を γ_1 を掛けた形とみなす.	[FALSE]

–**M**	M_2	出力される一般化ケプストラムの次数.	[25]
–**G**	G_2	出力される一般化ケプストラムのべきパラメータ γ_2. ただし，$G_2 > 1.0$ のときは $\gamma_2 = -1/G_2$.	[1]
–**N**		出力を正規化された一般化ケプストラムと見なす.	[FALSE]
–**U**		出力を γ_2 を掛けた形とみなす.	[FALSE]

EXAMPLE

float 形式の一般化ケプストラムファイル *data.gcep* ($M = 10, \gamma_1 = -0.5$) を 30 次のケプストラムに変換し *data.cep* に出力する:

```
gc2gc -m 10 -g 2 -M 30 -G 0 < data.gcep > data.cep
```

SEE ALSO

gcep, mgcep, freqt, mgc2mgc, lpc2c

NAME

gcep – 一般化ケプストラム分析 [678678678]

SYNOPSIS

gcep [–**m** M] [–**g** G] [–**l** L] [–**n**] [–**i** I] [–**j** J] [–**d** D] [–**e** E] [***infile***]

DESCRIPTION

一般化ケプストラム分析を行い，正規化一般化ケプストラム $c'_\gamma(m)$ を標準出力に出力します．入力は窓掛けされた長さ L の時系列

$$x(0), x(1), \ldots, x(L-1)$$

です．

データ形式は入力，出力とも float 形式です．

一般化ケプストラム分析では、音声スペクトルを M 次の一般化ケプストラム $c_\gamma(m)$ または正規化一般化ケプストラム $c'_\gamma(m)$ により

$$\begin{aligned} H(z) &= s_\gamma^{-1}\left(\sum_{m=0}^{M} c_\gamma(m) z^{-m}\right) \\ &= K \cdot s_\gamma^{-1}\left(\sum_{m=1}^{M} c'_\gamma(m) z^{-m}\right) \\ &= \begin{cases} K \cdot \left(1 + \gamma \sum_{m=1}^{M} c'_\gamma(m) z^{-m}\right)^{1/\gamma}, & -1 \le \gamma < 0 \\ K \cdot \exp \sum_{m=1}^{M} c'_\gamma(m) z^{-m}, & \gamma = 0 \end{cases} \end{aligned}$$

とモデル化し，対数スペクトルの不偏推定法における評価関数を適用します．評価関数の最小化は、 $\gamma = -1$ のときには線形予測法での線形方程式を解くことと等価になりますが，$\gamma = -1$ 以外では非線形方程式を解くことになりますので，ここでは Newton–Raphson 法を利用して解を求めています．

OPTIONS

–**m**	M	分析次数.	[25]
–**g**	G	一般化ケプストラムのべきパラメータ γ. ただし，$G > 1.0$ のときは $\gamma = -1/G$.	[0]

–l	L	入力データのフレーム長.	[256]
–n		正規化したケプストラムを出力.	[FALSE]

通常，以下のオプションの指定は必要ありません.

–i	I	分析の最小反復回数.	[2]
–j	J	分析の最大反復回数.	[30]
–d	D	Newton-Raphson 法の終了条件．デフォルトは $D = 0.001$ で，このときは $\varepsilon^{(i)}$ の繰り返しによる変化率が 0.001 つまり 0.1% 以内になったときに終了.	[0.001]
–e	E	ピリオドグラムに足し込む小さな値.	[0]

EXAMPLE

float 形式のファイル *data.f* を次数 15 次で一般化ケプストラム分析し，一般化ケプストラムを *data.gcep* に出力する:

```
frame < data.f | window | gcep -m 15 > data.gcep
```

SEE ALSO

icep, uels, mcep, mgcep, glsadf

NAME

glogsp – 対数スペクトルのプロット

SYNOPSIS

glogsp [–**O** *O*] [–**x** *X*] [–**y** *ymin ymax*] [–**ys** *YS*] [–**p** *P*] [–**ln** *LN*]
[–**s** *S*] [–**l** *L*] [–**c** *comment*] [***infile***]

DESCRIPTION

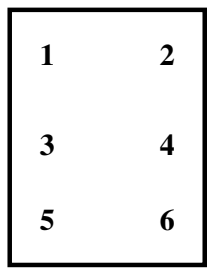
glogsp は，float 型のデータを標準入力から読み込んで，それをグラフ表示するプロッタのコマンド列を生成します．従って，*xgr* コマンド等と併用して使います．

実体はシェルスクリプトで，内部では *fig* と *fdrw* を用いています．

OPTIONS

–**O** *O* グラフの開始点. [1]

1 (40,205) [mm]
2 (125,205) [mm]
3 (40,120) [mm]
4 (125,120) [mm]
5 (40, 35) [mm]
6 (125, 35) [mm]



–**x** *X* x 軸のスケール. [1]

1 正規化された周波数 $(0 \sim 0.5)$
2 正規化された周波数 $(0 \sim \pi)$
4 周波数 (0 ∼ 4kHz)
5 周波数 (0 ∼ 5kHz)
8 周波数 (0 ∼ 8kHz)
10 周波数 (0 ∼ 10kHz)

–**y** *ymin ymax* y 軸のスケール [dB]. [0 100]

–**ys** *YS* y 軸のスケーリング率. [20]

–p	P	使用するペン番号 (1 ∼ 10).	[1]
–ln	LN	描画する際の線の種類 (0 ∼ 5). ＜ fig 参照＞.	[1]
–s	S	入力するデータの何フレーム目からグラフに書くかを指定.	[0]
–l	L	フレーム長. 実際には $\frac{L}{2}$ 個のデータをグラフに書く.	[256]
–c	comment	グラフのコメント.	[N/A]

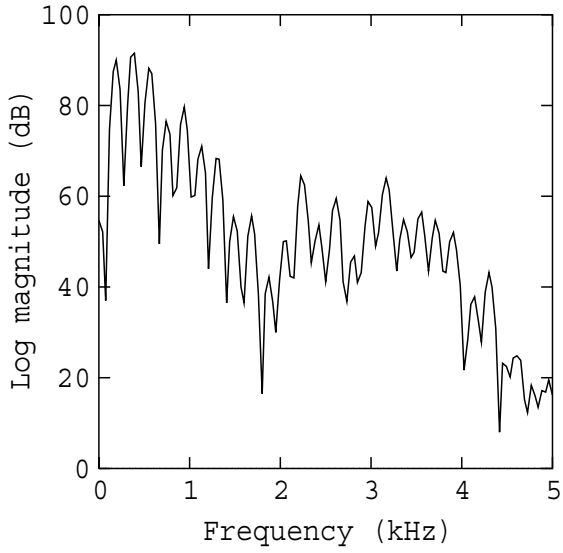
通常, 以下のオプションの指定の必要はありません.

–W	W	グラフの幅 (mm).	[0.6]
–H	H	グラフの高さ (mm).	[0.6]
–v		–v オプションが指定されていないと 1 フレームだけグラフに書くが, 指定されると –s で指定されたフレーム以降を全てグラフに上書き出力. ＜ fdrw の –n オプション参照＞	[FALSE]
–o	xo yo	グラフの開始点を mm 単位で指定. –o オプションが指定されると –O オプションは無効.	[40 205]
–g	G	グラフの枠の形式 (0 ∼ 2). ＜ fig 参照＞.	[2]
–f	$file$	fig コマンド用の追加データファイル.	[NULL]
–help		詳細なヘルプを表示.	

EXAMPLE

short 形式の音声データ (標本化周波数 10kHz) のファイル *data.s* の対数振幅スペクトルを求め, そのグラフを画面に描画する:

```
x2x +sf data.s | bcut -s 4000 -e 4255 | window -n 2| spec |\
glogsp -x 5 | xgr
```



SEE ALSO

fig, fdrw, xgr, psgr, grlogsp, gwave

NAME

glsadf – 音声合成のための GLSA フィルタ [17]

SYNOPSIS

glsadf [**–m** M] [**–g** G] [**–p** P] [**–i** I] [**–n**] [**–k**] ***gcfile*** [***infile***]

DESCRIPTION

入力データを *gcfile* の正規化一般化ケプストラム係数 $K, c'_\gamma(1), \ldots, c'_\gamma(M)$ をもつ GLSA フィルタによりフィルタリングし，標準出力に出力します.

データ形式は入力，出力とも float 形式です.

M 次の正規化一般化ケプストラム $c'_\gamma(m)$ による合成フィルタの伝達関数 $H(z)$ は

$$
\begin{aligned}
H(z) &= K \cdot D(z) \\
&= \begin{cases} K \cdot \left(1 + \gamma \sum_{m=1}^{M} c'_\gamma(m) z^{-m}\right)^{1/\gamma}, & 0 < \gamma \leq -1 \\ K \cdot \exp \sum_{m=1}^{M} c'_\gamma(m) z^{-m}, & \gamma = 0 \end{cases}
\end{aligned}
$$

となります．ここでは，べきパラメータが $\gamma = -1/G$ (G:自然数) のときのみを考えます．この場合，フィルタ $D(z)$ は図1のように,

$$
\frac{1}{C(z)} = \frac{1}{1 + \gamma \sum_{m=1}^{M} c'_\gamma(m) z^{-m}}
$$

の G 段縦続構成で実現することができます.

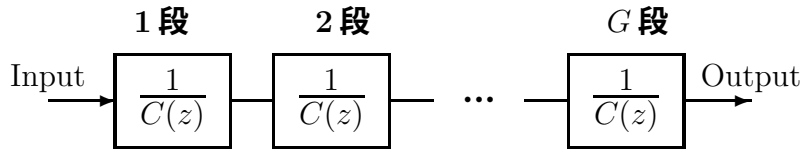


図 1: フィルタ $D(z)$ の構成

OPTIONS

–m	M	一般化ケプストラムの次数.	[25]
–g	G	一般化ケプストラムのべきパラメータ $\gamma = -1/G$.	[1]
–p	P	フレーム周期.	[100]
–i	I	補間周期.	[1]
–n		入力を規格化されたケプストラムとみなす.	[FALSE]
–k		ゲインを除いたシステム関数でフィルタリングする	[FALSE]

EXAMPLE

float 形式のピッチデータ *data.pitch* から励振源を作成し，一般化ケプストラムファイル *data.gcep* により GLSA フィルタを駆動し，合成音声を *data.syn* に出力する:

```
excite < data.pitch | glsadf data.gcep > data.syn
```

BUGS

n を自然数として，$\gamma = -1/n$ の場合にしか対応していない.

SEE ALSO

ltcdf, lmadf, lspdf, mlsadf, mglsadf

NAME

gnorm – 一般化ケプストラムのゲインの正規化

SYNOPSIS

gnorm [–**m** M] [–**g** G] [***infile***]

DESCRIPTION

正規化されていない一般化ケプストラム $c_\gamma(m)$ を正規化し，標準出力に出力します．
データ形式は入力，出力とも float 形式です．
正規化した一般化ケプストラム $c'_\gamma(m)$ は

$$c'_\gamma(m) = \frac{c_\gamma(m)}{1 + \gamma c_\gamma(0)}, \quad m > 0$$

で求められます．ただし，ゲイン項 $K = c'_\gamma(0)$ は

$$K = \begin{cases} \left(\dfrac{1}{1 + \gamma c_\gamma(0)}\right)^{1/\gamma}, & 0 < |\gamma| \leq 1 \\ \exp c_\gamma(0), & \gamma = 0 \end{cases}$$

となります．

OPTIONS

–**m**	M	一般化ケプストラムの次数．	[25]
–**g**	G	一般化ケプストラムのべきパラメータ γ ． ただし，$G > 1.0$ のときは $\gamma = -1/G$．	[0]

EXAMPLE

float 形式の正規化されていない一般化ケプストラムファイル *data.gcep* ($M = 15, \gamma = -0.5$) を正規化し，正規化一般化ケプストラムを *data.ngcep* に出力する:

```
gnorm -m 15 -g 2 < data.gcep > data.ngcep
```

SEE ALSO

ignorm, gcep, mgcep, gc2gc, mgc2mgc, freqt

NAME

grlogsp – ランニング対数スペクトルのプロット

SYNOPSIS

grlogsp [**–t**] [**–O** *O*] [**–x** *X*] [**–y** *ymin*] [**–yy** *YY*] [**–yo** *YO*] [**–p** *P*]
[**–ln** *LN*] [**–s** *S*] [**–e** *E*] [**–n** *N*] [**–l** *L*]
[**–c** *comment*1] [**–c2** *comment*2] [**–c3** *comment*3] [***infile***]

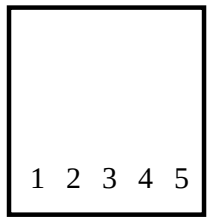
DESCRIPTION

grlogsp は、float 型のデータを標準入力から読み込んで，そのランニングスペクトルをグラフ表示するプロッタのコマンド列を生成します．従って，*xgr* コマンド等と併用して使います．実体はシェルスクリプトで，内部では *fig* と *fdrw* を用いています．

OPTIONS

–t		x 軸と y 軸を入れ換える	[FALSE]
–O	*O*	グラフの開始点.	[1]

1 (25,YO) [mm]
2 (60,YO) [mm]
3 (95,YO) [mm]
4 (130,YO) [mm]
5 (165,YO) [mm]



-t オプションが指定されている場合は、$(YO + 100, X)$ [mm] の位置にグラフを書きます.

–x	*X*	x 軸のスケール. 1 正規化された周波数 (0 ∼ 0.5) 2 正規化された周波数 (0 ∼ π) 4 周波数 (0 ∼ 4kHz) 5 周波数 (0 ∼ 5kHz) 8 周波数 (0 ∼ 8kHz) 10 周波数 (0 ∼ 10kHz)	[1]
–y	*ymin*	y 軸のスケーリングの最小値.	[-100]
–yy	*YY*	y 軸のスケーリング [dB/10mm] の範囲.	[100]
–yo	*YO*	y 軸のオフセット.	[30]
–p	*p*	使用するペン番号 (1 ∼ 10).	[2]
–ln	*LN*	描画する際の線の種類 (0 ∼ 5). ＜fig 参照＞.	[1]
–s	*S*	入力するデータの何フレーム目からグラフに書くかを指定.	[0]
–e	*E*	入力するデータの何フレーム目までグラフに書くかを指定.	[EOF]
–n	*N*	ランニングプロットするデータのフレーム数.	[EOF]
–l	*L*	フレーム長. 実際には $\frac{L}{2}$ 個のデータをグラフに書く.	[256]
–c, c2, c3	*comment*1 ∼ 3	グラフのコメント.	[N/A]

通常, 以下のオプションの指定の必要はありません.

–W	*W*	グラフの幅 (×100mm).	[0.25]
–H	*H*	グラフの高さ (×100mm).	[1.5]
–z	*Z*	ランニングスペクトルを描画する際、各フレームを y 軸方向にずらす距離 (mm).	[1]
–o	*xo yo*	グラフの開始点を mm 単位で指定. –o オプションが指定されると –O オプションは無効.	[95 30]
–g	*G*	グラフの枠の形式 (0 ∼ 2). ＜fig 参照＞.	[2]
–cy	*cy*	1 番目のコメントの位置.	[-8]
–cy2	*cy*2	2 番目のコメントの位置.	[-14]
–cy3	*cy*3	3 番目のコメントの位置.	[-20]
–cs	*cs*	コメントのサイズ.	[1]
–f	*f*	fig コマンド用の追加データファイル.	[NULL]

EXAMPLE

float 形式のファイル *data.f* のデータの対数振幅スペクトルを求め，ランニングプロットしたグラフを PostScript 形式で data.ps に書き込む:

```
frame < data.f | window |\
uels -m 15 | c2sp -m 15 |\
grlogsp | psgr > data.ps
```

SEE ALSO

fig, fdrw, xgr, psgr, glogsp, gwave

NAME

grpdelay – 郡遅延を求める

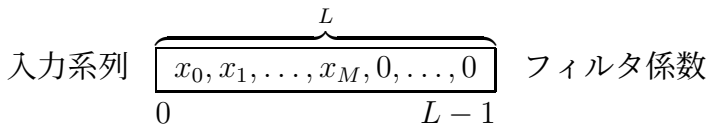
SYNOPSIS

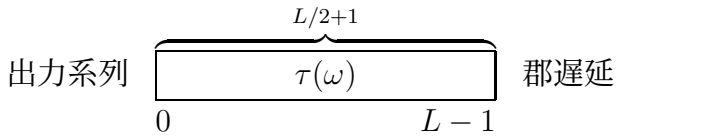
grpdelay [–**l** L] [–**m** M] [–**a**] [***infile***]

DESCRIPTION

ファイル *infile*(省略時は標準入力) からフィルタ係数 (FIR フィルタ) を読み込み, 郡遅延を出力します. データ形式は入出力は共に float 形式です.

–**m** オプションで M の指定がなく, かつ入力系列長が FFT サイズより短い場合, 入力系列に 0 を付加して FFT を実行します. –**a** オプションが指定された場合, 入力の零次の項はゲイン項として取り扱います.





OPTIONS

–**l**	L	フーリエ変換のサイズ. 2 のべき乗で指定.	[256]
–**m**	M	フィルタの次数.	[L-1]
–**a**		入力系列を AR フィルタの係数とみなす.	[FALSE]

EXAMPLE

伝達関数

$$H(z) = \frac{1}{1 + 0.9z^{-1}}$$

をもつフィルタのインパルス応答の群遅延を求め, 画面にプロットさせる:

```
impluse | dfs -a 1 0.9 | grpdelay | fdrw | xgr
```

SEE ALSO

delay, phase

NAME

gwave – 音声波形のプロット

SYNOPSIS

gwave [–**s** *S*] [–**e** *E*] [–**n** *N*] [–**i** *I*] [–**y** *ymax*] [–**y2** *ymin*] [–**p** *P*]
[+***type***] [***infile***]

DESCRIPTION

gwave は、音声波形データを標準入力から読み込んで表示する，プロッタのコマンド列を生成します．従って，*xgr* コマンド等と併用して使います．実体はシェルスクリプトで，内部では *fig* と *fdrw* を用いています.

OPTIONS

–**s**	*S*	表示開始ポイント	[0]
–**e**	*E*	表示終了ポイント	[EOF]
–**n**	*N*	グラフ 1 行に表示するデータ数	[N/A]
		指定しない場合は 1 画面に全てのデータを表示します.	
–**i**	*I*	1 画面のグラフの行数	[5]
–**y**	*ymax*	*y* 軸の最大値.	[N/A]
		指定しない場合は入力データの最大絶対値となります.	
–**y2**	*ymin*	*y* 軸の最小値.	[-YMAX]
–**p**	*P*	使用するペン番号 (1 ∼ 10).	[1]
+*t*		入力データの形式.	[f]

s short 型 (2bytes) i int 型 (4bytes)
f float 型 (4bytes) d double 型 (8bytes)

EXAMPLE

float 形式の音声波形ファイル *data.f* を PostScript 形式で *data.ps* に書き込む.

```
gwave < data.f | psgr > data.ps
```

SEE ALSO

fig, fdrw, xgr, psgr, glogsp, grlogsp

NAME

histogram – ヒストグラムの計算

SYNOPSIS

histogram [–**l** L] [–**i** I] [–**j** J] [–**s** S] [–**n**] [***infile***]

DESCRIPTION

指定されたファイルからヒストグラムを計算し，標準出力に出力します.

データ形式は入力，出力とも float 形式です．グラフを表示する場合は，fdrw に入力してください.

範囲外のデータが含まれている場合，ヒストグラムは出力されますが，返り値が 0 以外の値となります.

OPTIONS

–**l**	L	フレームのサイズ.	[0]
		$L > 0$ フレームごとにヒストグラムを計算.	
		$L = 0$ ファイル全体のヒストグラムを出力.	
–**i**	I	ヒストグラムの最小値.	[0.0]
–**j**	J	ヒストグラムの最大値.	[1.0]
–**s**	S	ヒストグラムのステップサイズ.	[0.1]
–**n**		出力をサンプル数で割る.	[FALSE]

EXAMPLE

float 形式の音声波形ファイル *data.f* のヒストグラムを表示する:

```
histogram -i -16000 -j 16000 -s 100 data.f | fdrw | xgr
```

SEE ALSO

average

NAME

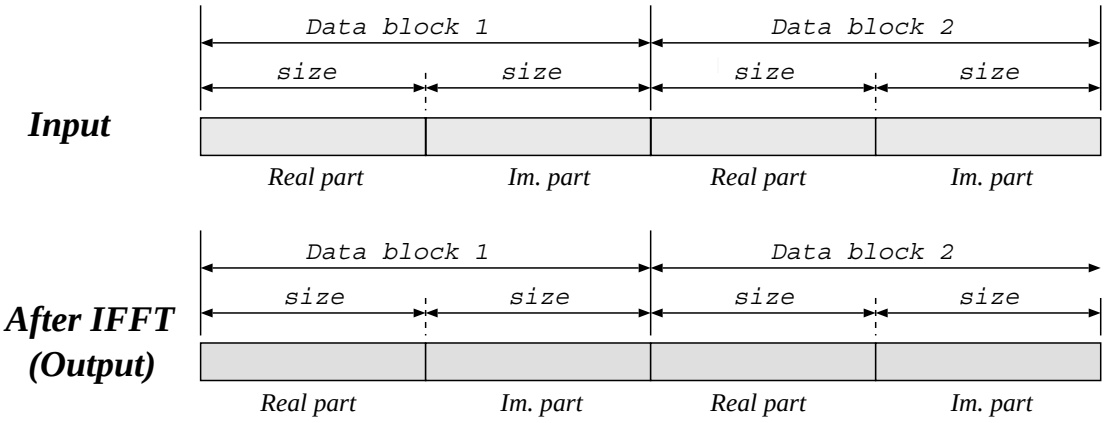
ifft – 複素数列の高速逆フーリエ変換

SYNOPSIS

ifft [–l *L*] [–{ **R** | **I** }] [***infile***]

DESCRIPTION

ifft は，複素数で与えられるデータ列を *infile* から読み込んで逆フーリエ変換を行ない，結果を標準出力に出力します．入力は float 形式で，実部を L 個並べた後に虚部を L 個並べた形になります．



OPTIONS

–l	*L*	逆フーリエ変換のサイズ．2 のべき乗で指定.	[256]
–R		逆変換の結果の実部のみを出力.	[FALSE]
–I		逆変換の結果の虚部のみを出力.	[FALSE]

EXAMPLE

float 形式のファイル *data.f* にあるデータ列（実部 256 点，虚部 256 点）の 逆 DFT を求め，data.ifft に出力する:

```
ifft data.f -l 256 > data.ifft
```

SEE ALSO

fft, fft2, fftr, fftr2, ifft2

NAME

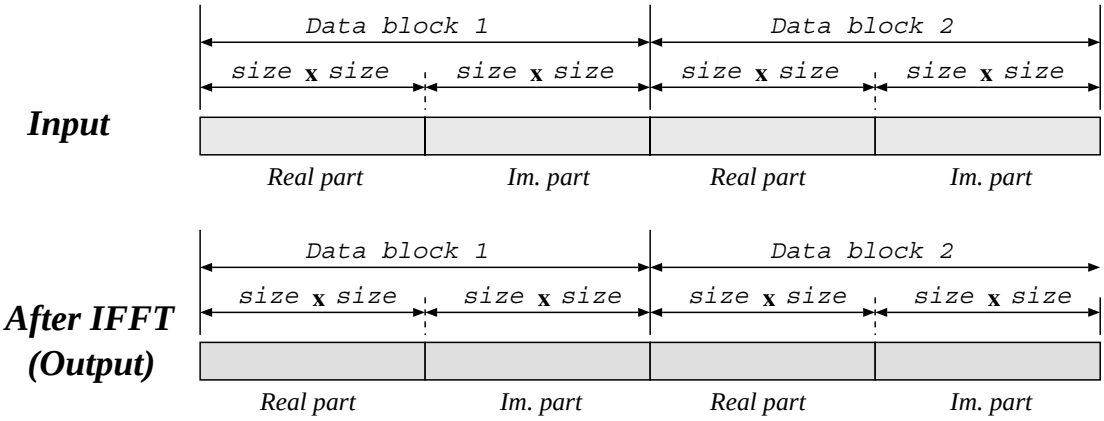
ifft2 – 複素数列の 2 次元高速逆フーリエ変換

SYNOPSIS

ifft2 [–**l** *L*] [+**r**] [–**t**] [–**c**] [–**q**] [–**{ R** | **I } **] [***infile***]

DESCRIPTION

ifft2 は，複素数系列で与えられるデータ列を *infile* から読み込んで 2 次元逆フーリエ変換を行ない，結果を標準出力に出力します．*infile* が指定されない時は標準入力から読み込みます．データのフォーマットは以下の通りです．



OPTIONS

–**l**	*L*	フーリエ変換のサイズ．2 のべき乗で指定.	[64]
+**r**		入力を実数型とする.	[FALSE]
–**t**		逆 FFT の結果を transpose して出力．＜fft2 参照＞.	[FALSE]
–**c**		transpose する際，境界の 1 データを反対側から持ってきて，$(L+1)\times(L+1)$ 個のデータを出力．＜fft2 参照＞.	[FALSE]
–**q**		逆 FFT の結果の最初の 1/4 のデータのみを出力．この際 c オプションと同様に，境界の補償を行ない，$(\frac{L}{2}+1)\times(\frac{L}{2}+1)$ 個のデータを出力．＜fft2 参照＞.	[FALSE]
–**R**		実部のみを出力.	[FALSE]
–**I**		虚部のみを出力.	[FALSE]

EXAMPLE

float 形式のファイル *data.f* にある 2 次元複素数列の 逆 DFT を求め，*data.ifft2* に出力する:

```
ifft2 -A < data.f > data.ifft2
```

SEE ALSO

fft, fft2, ifft

NAME

iglsadf – 逆 GLSA ディジタルフィルタ [17]

SYNOPSIS

iglsadf [**–m** M] [**–g** G] [**–p** P] [**–i** I] [**–n**] [**–k**] ***gcfile*** [***infile***]

DESCRIPTION

iglsadf は，逆 GLSA フィルタです.

データ形式は入力，出力とも float 形式です.

OPTIONS

–m	M	一般化ケプストラムの次数.	[25]
–g	G	一般化ケプストラムのべきパラメータ $\gamma = -1/G$.	[1]
–p	P	フレーム周期.	[100]
–i	I	補間周期.	[1]
–n		入力を規格化されたケプストラムとみなす.	[FALSE]
–k		ゲインを除いたシステム関数でフィルタリングする	[FALSE]

EXAMPLE

float 形式の音声ファイル *data.f* を一般化ケプストラム分析 (15 次，$\gamma = -1/2$) の結果に基づいて逆フィルタ出力し，*data.e* を求める:

```
frame < data.f | window | gcep -m 15 -g 2 > data.gc
iglsadf -m 15 -g 2 data.gc < data.f > data.e
```

BUGS

n を自然数として，$\gamma = -1/n$ の場合にしか対応していない.

SEE ALSO

glsadf, gcep

NAME

ignorm – 一般化ケプストラムの逆正規化

SYNOPSIS

ignorm [–**m** M] [–**g** G] [***infile***]

DESCRIPTION

正規化された一般化ケプストラム $c'_\gamma(m)$ を正規化されていないものに変換し，標準出力に出力します.

データ形式は入力，出力とも float 形式です.

正規化した一般化ケプストラム $c'_\gamma(m)$ から正規化されていない一般化ケプストラム $c_\gamma(m)$ は

$$c_\gamma(m) = \left(c'_\gamma(0)\right)^\gamma c'_\gamma(m), \quad m > 0$$

で求められます．ただし，ゲイン項 $c_\gamma(0)$ は

$$c_\gamma(0) = \begin{cases} \dfrac{\left(c'_\gamma(0)\right)^\gamma - 1.0}{\gamma}, & 0 < |\gamma| \leq 1 \\ \log c'_\gamma(0), & \gamma = 0 \end{cases}$$

となります.

OPTIONS

–**m**	M	一般化ケプストラムの次数.	[25]
–**g**	G	一般化ケプストラムのべきパラメータ γ .	[0]
		ただし，$G > 1.0$ のときは $\gamma = -1/G$.	

EXAMPLE

float 形式の正規化されている一般化ケプストラムファイル *data.ngcep* ($M = 15, \gamma = -0.5$) を逆正規化し，正規化されていない一般化ケプストラムを data.gcep に出力する:

```
ignorm -m 15 -g -0.5 < data.ngcep > data.gcep
```

SEE ALSO

gcep, mgcep, gc2gc, mgc2mgc, freqt

NAME

imglsadf – 音声合成のための逆 MGLSA フィルタ [19]

SYNOPSIS

imglsadf [**–m** M] [**–a** A] [**–g** G] [**–p** P] [**–i** I] [**–t**] [**–k**]
mgcfile [***infile***]

DESCRIPTION

mgcfile のメル一般化ケプストラム係数 $c_{\alpha,\gamma}(m)$ で与えられる伝達関数の逆関数をもつ MGLSA フィルタにより，入力データをフィルタリングして標準出力に出力します.

データ形式は入力，出力とも float 形式です.

OPTIONS

–m	M	メル一般化ケプストラムの次数.	[25]
–a	A	周波数圧縮パラメータ α.	[0.35]
–g	G	一般化ケプストラムのべきパラメータ γ.	[1]
		ただし，$G > 1.0$ のときは $\gamma = -1/G$.	
–p	P	係数の更新周期.	[100]
–i	I	係数の補間周期.	[1]
–t		転置型フィルタ.	[FALSE]
–k		ゲインを除いたシステム関数でフィルタリングする	[FALSE]

EXAMPLE

float 形式の音声データファイル *data.f* をメル一般化ケプストラム分析 (15 次，$\alpha = 0.35$，$\gamma = -1/2$) の結果に基づいて，逆フィルタを出力し，*data.e* を求める:

```
frame < data.f | window | \
    mgcep -m 15 -a 0.35 -g -0.5 > data.mgc
imglsadf -m 15 -a 0.35 -g -0.5 data.mgc < data.f \
    > data.e
```

BUGS

n を自然数として，$\gamma = -1/n$ の場合にしか対応していない.

SEE ALSO

mgcep, amgcep, ltcdf, lmadf, mlsadf, glsadf, lspdf

NAME

impulse – ユニットパルス列 (インパルス列) を発生する

SYNOPSIS

impulse [–**l** L] [–**n** N]

DESCRIPTION

長さ L のユニットパルス列を標準出力に出力します．つまり，

$$\underbrace{1, 0, 0, \ldots, 0}_{L}$$

出力データは，float 形式です．–l と–n オプションを同時に指定した場合，後に指定したものに従います．

OPTIONS

–**l**	L	ユニットパルス列の長さ． $L < 0$ とすると無限長の数列を生成．	[256]
–**n**	N	ユニットパルス列の次数． N+1 の系列を生成．	[255]

EXAMPLE

標準形のディジタルフィルタにユニットパルス列を通し，値を表示する:

```
impulse | dfs -a 1 0.9 -b 1 2 1 | dmp
```

SEE ALSO

step, train, ramp, sin, nrand

NAME

imsvq – マルチステージベクトル量子化器のデコーダ

SYNOPSIS

imsvq [–**l** *L*] [–**n** *N*] [–**s** *S* ***cbfile***] [***infile***]

DESCRIPTION

imsvq は，指定されたファイルからインデックスを読み込み，符号帳ベクトルを出力します．-s オプションはステージの段数回繰り返して指定します．

データ形式は入力は int 形式，出力は float 形式です．

OPTIONS

–**l**	*L*	ベクトルのサイズ．	[26]
–**n**	*N*	ベクトルの次数．ベクトルのサイズは N+1 になります．	[L-1]
–**s**	*S cbfile*	コードブックの指定．	[N/A N/A]
		S コードブックサイズ	
		cbfile コードブックファイル	

EXAMPLE

コードブックサイズ 256 のコードブック *cbfile1* と *cbfile2* で 2 段 VQ を行った結果であるファイル *data.vq* に対応する符号帳ベクトルを *data.ivq* に出力します:

```
imsvq -s 256 cbfile1 -s 256 cbfile2 < data.vq > data.ivq
```

SEE ALSO

msvq, ivq, vq

NAME

interpolate – データ列の補間

SYNOPSIS

interpolate [–**p** P] [–**s** S] [***infile***]

DESCRIPTION

標準入力から入力される時系列

$$x(0), x(1), x(2), \ldots,$$

から周期 P で

$$\underbrace{0, 0, \ldots, 0}_{S-1}, \underbrace{x(0), 0, 0, \ldots, 0}_{P}, \underbrace{x(1), 0, 0, \ldots, 0}_{P}, x(2), \ldots,$$

のように 0 を補間します.

データ形式は入力，出力とも float 形式です.

OPTIONS

–**p**	P	補間周期.	[10]
–**s**	S	開始サンプル.	[0]

EXAMPLE

float 形式のデータ *data.f* を周期 2 で decimation し，さらに 0 を周期 2 で interpolation して *data.di* に出力する:

```
decimate -p 2  < data.f | interpolate -p 2 > data.di
```

SEE ALSO

decimate

NAME

ivq – ベクトル量子化のデコーダ

SYNOPSIS

ivq [–**l** L] [–**n** N] ***cbfile*** [***infile***]

DESCRIPTION

ivq は，指定されたファイルからインデックス i を読み込み，コードブックファイル *cbfile* に従って，符号帳ベクトル

$$c_i(0), c_i(1), \ldots, c_i(L-1)$$

を出力します.

データ形式は入力は int 形式，出力は float 形式です.

OPTIONS

–**l**	L	ベクトルのサイズ.	[26]
–**n**	N	ベクトルの次数．ベクトルのサイズは N+1 になります.	[L-1]

EXAMPLE

25 次のベクトルに対し，コードブック *cbfile* で VQ を行った結果である *data.vq* に対応する符号帳ベクトルを *data.ivq* に出力します.

```
ivq cbfile data.vq > data.ivq
```

SEE ALSO

vq, imsvq, msvq

NAME

lbg – ベクトル量子化器設計のための LBG アルゴリズム

SYNOPSIS

lbg [–**l** L] [–**n** N] [–**t** T] [–**s** S] [–**e** E] [–**f** F] [–**d** D] [–**r** R] [***indexfile***] < ***infile***

DESCRIPTION

lbg は，lbg アルゴリズムによりコードブックの学習を行ないます．データ形式は入力，出力ともに float 型です．

lbg アルゴリズムでは，標準入力から与えるベクトルサイズ L，のトレーニングベクトル系列

$$\mathrm{x}(0), \mathrm{x}(1), \ldots, \mathrm{x}(T-1)$$

からコードブック

$$\mathrm{C}_E = \{\mathrm{c}_E(0), \mathrm{c}_E(1), \ldots, \mathrm{c}_E(E-1)\}$$

を求め，標準出力に出力します．次のアルゴリズムにより，コードブックを作成します．

step.0 初期コードブックが指定された場合は，それを C_S とし，そうでない場合は，すべてのトレーニングデータのセントロイド

$$\mathrm{c}_1(0) = \frac{1}{T}\sum_{n=0}^{T-1}\mathrm{x}(n)$$

を求め，$\mathrm{C}_1 = \{\mathrm{c}_1(0)\}, S = 1$ とします．

step.1 コードブック C_S を C_{2S} にします．この時長さ L の正規乱数 rnd および，splitting factor R を用いて，

$$\mathrm{c}_{2S}(n) = \begin{cases} \mathrm{c}_S(n) + R\cdot\mathrm{rnd} & (0 \le n \le S-1) \\ \mathrm{c}_S(n) - R\cdot\mathrm{rnd} & (S \le n \le 2S-1) \end{cases}$$

とします．$D_0 = \infty, k = 1$ とします．

step.2 現在のコードブック C_{2S} に対して，トレーニングベクトルを量子化します．その後，トレーニングベクトルと，コードベクトルとの平均ユークリッド距離 D_k を求め，終了条件 D を用いて，

$$\left|\frac{D_{k-1} - D_k}{D_k}\right| < D$$

の場合 **step.4** に進み，それ以外の場合は **step.3** に進みます.

step.3 **step.2** で得られた結果からセントロイドを求め，コードブック C_{2S} を更新します．ただし，量子化した結果，インデックス i にトレーニングベクトルが一つも割り当てられなかった場合は，最も割り当てられた数が多いコードベクトル $\mathrm{c}_{2S}(j)$ を用いて

$$\mathrm{c}_{2S}(i) = \mathrm{c}_{2S}(j) + R \cdot \mathrm{rnd}$$

として求めます．その後 $k = k + 1$ として，**step.2** に戻ります.

step.4 $2S = E$ なら終了，そうでなければ **step.1** に戻ります.

OPTIONS

–l	L	ベクトルのサイズ.	[26]
–n	N	ベクトルの次数．ベクトルのサイズは N+1 になります.	[L-1]
–t	T	トレーニングベクトル数.	[N/A]
		入力がパイプでない場合は省略できます.	
–s	S	初期コードブックサイズ.	[1]
–e	E	最終コードブックサイズ．2 のべき乗で指定.	[256]
–f	F	初期コードブックファイルネーム.	[NULL]

通常，以下のオプションの指定は必要ありません.

–d	D	終了条件.	[0.0001]
–r	R	splitting factor	[0.0001]

EXAMPLE

float 形式の 25 次のトレーニングベクトルファイル *data.f* からコードブックサイズ 256 のコードブックを作成し，*cbfile* に出力します:

```
lbg < data.f > cbfile
```

SEE ALSO

vq, ivq, msvq

NAME

levdur – レビンソン・ダービンの算法

SYNOPSIS

levdur [–**m** M] [***infile***]

DESCRIPTION

自己相関行列から線形方程式を解き，線形予測係数を求めます．指定されたファイルから M 次の自己相関数列

$$r(0), r(1), \ldots, r(M)$$

を読み込み，レビンソン・ダービンの算法により線形方程式を解きます．

データ形式は入力，出力とも float 形式です．

線形予測係数は全極型ディジタルフィルタ

$$H(z) = \frac{K}{1 + \sum_{i=1}^{M} a(k) z^{-i}}$$

の係数 $K, a(1), \ldots, a(M)$ です。線形予測分析の自己相関法では，次の線形方程式

$$\begin{pmatrix} r(0) & r(1) & \cdots & r(M-1) \\ r(1) & r(0) & & \vdots \\ \vdots & & \ddots & \\ r(M-1) & & \cdots & r(0) \end{pmatrix} \begin{pmatrix} a(1) \\ a(2) \\ \vdots \\ a(M) \end{pmatrix} = - \begin{pmatrix} r(1) \\ r(2) \\ \vdots \\ r(M) \end{pmatrix}$$

を解くことで線形予測係数を求めます．この際，係数行列の Toeplitz 性を利用することにより，次のようなダービンの再帰法で効率的に計算することができます．

$$\begin{aligned} E^{(0)} &= r(0) && (1) \\ k(i) &= \frac{-r(i) - \sum_{j=1}^{i} a^{(i-1)}(j) r(i-j)}{E^{(i-1)}} && (2) \\ a^{(i)}(i) &= k(i) && (3) \\ a^{(i)}(j) &= a^{(i-1)}(j) + k(i) a^{(i-1)}(i-j), \quad 1 \le j \le i-1 && (4) \\ E^{(i)} &= (1 - k^2(i)) E^{(i-1)} && (5) \end{aligned}$$

ただし，式 (2) から (5) を $i = 1, 2, \ldots, M$ の順に繰り返します．また，ゲイン K は

$$K = \sqrt{E^{(M)}}$$

で求めます．

OPTIONS

−m M 分析次数. [25]

EXAMPLE

float 形式のファイル *data.f* を線形予測分析し，線形予測係数を *data.lpc* に出力する:

```
frame < data.f | window | acorr -m 25 | levdur > data.lpc
```

SEE ALSO

acorr, lpc

NAME

linear_intpl – データ列の直線補間

SYNOPSIS

linear_intpl [–**l** L] [–**m** M] [–**x** x_{min} x_{max}] [***infile***]

DESCRIPTION

標準入力から入力される 2 次元データ列

$$x_0, y_0$$
$$x_1, y_1$$
$$\vdots$$
$$x_K, y_K$$

に対して，x 軸を $L-1$ 点で等間隔に補間したときの y 軸の値

$$y_0, y_1, \ldots, y_{L-1}$$

を標準出力に出力します.

データ形式は入力，出力とも float 形式です.

このコマンドは，ディジタルフィルタの特性を与えるときなど，x 軸が等間隔ではないデータ列に対して補間を行うことができます.

OPTIONS

–**l**	L	出力の長さ.	[256]
–**m**	M	補間の点数.	[L-1]
–**x**	x_{min} x_{max}	入力データの x 座標の範囲の最小値と最大値.	[0.0 0.0]

EXAMPLE

float 形式のファイル *data.f* には以下のデータが入っているとする.

$$0, 2$$
$$2, 2$$

3, 0
5, 1

ここで

```
linear_intpl -m 10 -x 0 5 < data.f > data.intpl
```

とすると，ファイル *data.intpl* には以下のデータが入る．

2, 2, 2, 2, 2, 1, 0, 0.25, 0.5, 0.75, 1

NAME

lmadf – 音声合成のための LMA フィルタ [516516]

SYNOPSIS

lmadf [–**m** M] [–**p** P] [–**i** I] [–**P** Pa] [–**k**] ***cfile*** [***infile***]

DESCRIPTION

指定されたファイルから読み込まれたデータを *cfile* のケプストラム係数 $c(0)$,$c(1)$,...,$c(M)$ をもつ LMA フィルタによりフィルタリングし，標準出力に出力します.

データ形式は入力，出力とも float 形式です.

LMA フィルタは，M 次のケプストラム $c(m)$ で表現された指数形伝達関数

$$H(z) = \exp \sum_{m=0}^{M} c(m) z^{-m}$$

を高精度に近似します．伝達関数 $H(z)$ からゲイン $K = \exp c(0)$ を除いた伝達関数

$$D(z) = \exp \sum_{m=1}^{M} c(m) z^{-m}$$

は，次式の FIR フィルタ

$$F(z) = \sum_{m=1}^{M} c(m) z^{-m}$$

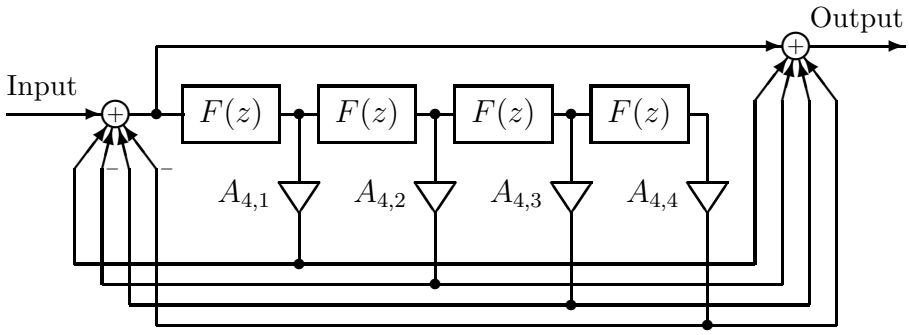
を基礎フィルタとして図1(a) のように実現することができます．ここで，図1(b) のように基礎フィルタ $F(z)$ を

$$F(z) = F_1(z) + F_2(z)$$

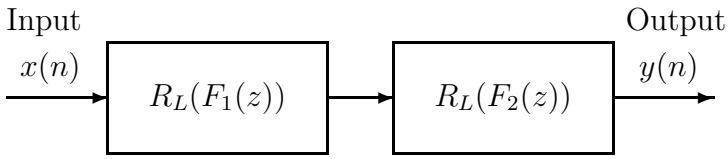
ただし，

$$\begin{aligned} F_1(z) &= c(1) z^{-1} \\ F_2(z) &= \sum_{m=2}^{M} c(m) z^{-m} \end{aligned}$$

と分割することにより，近似精度を上げています．また，図1(a) の係数 $A_{4,l}$ の値を表1に示します.



(a)



(b)

図 1: (a) $R_L(F(z)) \simeq D(z) \quad L = 4$
(b) 2 段縦続接続構成
$R_L(F_1(z)) \cdot R_L(F_2(z)) \simeq D(z)$

表 1: 有理近似式の係数 $A_{L,l}$ の値

l	$A_{4,l}$	$A_{5,l}$
1	4.999273×10^{-1}	4.999391×10^{-1}
2	1.067005×10^{-1}	1.107098×10^{-1}
3	1.170221×10^{-2}	1.369984×10^{-2}
4	5.656279×10^{-4}	9.564853×10^{-4}
5		3.041721×10^{-5}

OPTIONS

–m	M	ケプストラムの次数.	[25]
–p	P	係数の更新周期.	[100]
–i	I	補間周期.	[1]
–P	Pa	パデ近似次数.	[4]
		Pa は 4 又は 5 を指定できます.	

–k	ゲインを除いたシステム関数でフィルタリングする	[FALSE]

EXAMPLE

float 形式のピッチデータ *data.pitch* から励振源を作成し，ケプストラムファイル *data.cep* により LMA フィルタを駆動し，合成音声を *data.syn* に出力する:

```
excite < data.pitch | lmadf data.cep > data.syn
```

SEE ALSO

icep, uels, acep, poledf, ltcdf, glsadf, mlsadf, mglsadf

NAME

lpc – 線形予測分析

SYNOPSIS

lpc [–**l** L] [–**m** M] [***infile***]

DESCRIPTION

線形予測分析を行います．指定されたファイルから窓掛けされた長さ L の時系列

$$x(0), x(1), \ldots, x(L-1)$$

を読み込み，まず自己相関を計算します（acorr 参照）．次に，レビンソン・ダービンの算法により線形予測係数

$$K, a(1), \ldots, a(M)$$

を求め（levdur 参照），標準出力に出力します．ここで，K はゲインを表しています．

データ形式は入力，出力とも float 形式です．

OPTIONS

–**l**	L	入力データのフレーム長.	[256]
–**m**	M	分析次数.	[25]

EXAMPLE

float 形式のファイル *data.f* を分析次数 20 次で線形予測分析し，線形予測係数を *data.lpc* に出力する:

```
frame < data.f | window | lpc -m 20 > data.lpc
```

SEE ALSO

acorr, levdur, lpc2par, par2lpc, lpc2c, lpc2lsp, lsp2lpc, ltcdf, lspdf

NAME

lpc2c – LPC ケプストラム

SYNOPSIS

lpc2c [–**m** M_1] [–**M** M_2] [***infile***]

DESCRIPTION

lpc2c は，指定されたファイルから LPC 係数を読み込み，LPC ケプストラムを出力します．つまり，入力数列を

$$\sigma, a(1), a(2), \cdots, a(p)$$

として，

$$c(n) = \begin{cases} ln(h), & n = 0 \\ -a(n) = -\sum_{k=1}^{n-1} \frac{k}{n} c(k)a(n-k), & 1 \leq n \leq P \\ -\sum_{k=n-P}^{n-1} \frac{k}{n} c(k)a(n-k), & n > P \end{cases}$$

但し，

$$H(z) = \frac{\sigma}{A(z)} = \frac{\sigma}{1 + \sum_{k=1}^{P} a(k)z^{-k}}$$

を計算し，

$$c(0), c(1), \cdots, c(M)$$

を出力します．データ形式は入出力とも float 形式です．

OPTIONS

–**m** M_1 LPC の次数. [25]
–**M** M_2 ケプストラムの次数. [25]

EXAMPLE

float 形式の音声データ *data.f* に窓掛けしたデータの 10 次の LPC 係数を求め，さらに 15 次の LPC ケプストラムを求め，*data.cep* に出力する:

```
frame < data.f | window | lpc -m 10 |\
lpc2c -m 10 -M 15 > data.cep
```

SEE ALSO

lpc, gc2gc, mgc2mgc, freqt

NAME

lpc2lsp – LPC から LSP に変換

SYNOPSIS

lpc2lsp [–**m** M] [–**s** S] [–**k**] [–**o** O] [–**n** N] [–**p** P] [–**q** Q] [–**d** D] [***infile***]

DESCRIPTION

lpc2lsp は，指定されたファイルから線形予測係数

$$K, a(1), \ldots, a(M)$$

を読み込み，LSP 係数を計算し，標準出力に出力します．ここで，K はゲインを表していますが，得られる LSP 係数の値には影響を及ぼしません．

M 次の線形予測多項式 $A(z)$ は

$$A_M(z) = 1 + \sum_{m=1}^{M} a(m) z^{-m}$$

で与えられ，PARCOR 係数を用いた漸化式

$$\begin{aligned} A_m(z) &= A_{m-1}(z) - k(m) B_{m-1}(z) \\ B_m(z) &= z^{-1}(B_{m-1}(z) - k(m) A_{m-1}(z)) \end{aligned}$$

を満足します．但し，初期条件は次の通りです．

$$A_0(z) = 1, \quad B_0(z) = z^{-1}$$

いま，M 次の線形予測多項式 $A_M(z)$ が与えられたとき，$k(M+1)$ を 1 および -1 としたときの $A_{M+1}(z)$ をそれぞれ $P(z)$ と $Q(z)$ で表すと

$$\begin{aligned} P(z) &= A_M(z) - B_M(z) \\ Q(z) &= A_M(z) + B_M(z) \end{aligned}$$

となります．ここで，$k(M+1)$ を ± 1 とすることは，PARCOR 係数によって定まる擬声道モデルの声門での境界条件を完全反射とすることに対応しています．また，$A_M(z)$ は $P(z)$ と $Q(z)$ により

$$A_M(z) = (P(z) + Q(z))/2$$

と表されます. $A_M(z)$ をこのように表現したとき, $A_M(z)$ が安定, つまり $A_M(z) = 0$ のすべての根が単位円内に存在するための必要十分条件は次のように与えられます.

- $P(z) = 0$ と $Q(z) = 0$ の根はすべて単位円上にある.
- $P(z) = 0$ と $Q(z) = 0$ の根は円周上で互いに他を隔離する.

すなわち, $P(z) = 0$ と $Q(z) = 0$ の根が上記の性質を満たせば, $A_M(z)$ は安定となります.

ここで, M を偶数と仮定すると, $P(z)$ と $Q(z)$ は次のように因数分解されます.

$$\begin{aligned} P(z) &= (1 - z^{-1}) \prod_{i=2,4,\cdots,M} (1 - 2z^{-1}\cos\omega_i + z^{-2}) \\ Q(z) &= (1 + z^{-1}) \prod_{i=1,3,\cdots,M-1} (1 - 2z^{-1}\cos\omega_i + z^{-2}) \end{aligned}$$

ただし, ω_i は次の関係を満たすように順序付けされています. 奇数の場合も同様に求めることができます.

$$0 < \omega_1 < \omega_2 < \cdots < \omega_{M-1} < \omega_M < \pi$$

この因数分解に現れる ω_i を LSP と呼びます.

OPTIONS

−m	M	LPC の次数.	[25]
−s	S	サンプリング周波数 (kHz).	[10]
−k		ゲインを出力する	[TRUE]
−o	O	出力形式.	[0]
		0 規格化周波数 $(0 \ldots \pi)$	
		1 規格化周波数 $(0 \ldots 0.5)$	
		2 周波数 (kHz)	
		3 周波数 (Hz)	

通常, 以下のオプションの指定は必要ありません.

−n	N	unit circle の split 数.	[128]
−p	P	$P(z)$ の補間の最大数.	[4]
−q	Q	$Q(z)$ の補間の最大数.	[15]
−d	D	補間の終了状態.	[1e-06]

EXAMPLE

float 形式の音声ファイル *data.f* から 10 次の LPC 係数を求め，さらに LSP 係数を求め *data.lsp* に出力する:

```
frame < data.f | window | lpc -m 10 |\
lpc2lsp -m 10 > data.lsp
```

SEE ALSO

lpc, lsp2lpc, lspdf

NAME

lpc2par – LPC から反射係数を求める

SYNOPSIS

lpc2par [–**m** M] [–**g** G] [–**s**] [***infile***]

DESCRIPTION

線形予測係数から反射係数 (PARCOR) を求めます．指定されたファイルから M 次の線形予測係数

$$K, a(1), \ldots, a(M)$$

を読み込み，反射係数

$$K, k(1), \ldots, k(M)$$

を標準出力に出力します．また，–s オプションを指定した場合には，安定性の判別を行い，安定の場合には 0，不安定の場合には 1 が標準出力に出力されます．

データ形式は入力，出力とも float 形式です．

線形予測係数から反射係数を求める変換は，

$$\begin{aligned} k(m) &= a^{(m)}(m) \\ a^{(m-1)}(i) &= \frac{a^{(m)}(i) + a^{(m)}(m)a^{(m)}(m-i)}{1-k^2(m)}, \qquad 1 \le i \le m-1 \end{aligned}$$

を $m = p, p-1, \ldots, 1$ の順に繰り返します．ここで，初期条件は

$$a^{(M)}(m) = a(m), \qquad 1 \le m \le M$$

です．–g オプションを指定した場合，べきパラメータを γ とする正規化一般化ケプストラムを入力として，反射係数を求めます．この場合入力

$$K, c'_\gamma(1), \ldots, c'_\gamma(M)$$

を読み込み，

$$a^{(M)}(m) = \gamma c'_\gamma(M), \qquad 1 \le m \le M$$

として，計算します．

また，安定性の判別する際には，反射係数がすべて

$$-1 < k(m) < 1$$

という条件を満たしているか調べています.

OPTIONS

–m	M	LPC の次数.	[25]
–g	G	一般化ケプストラムのべきパラメータ γ.	[1]
		ただし，$G > 1.0$ のときは $\gamma = -1/G$.	
–s		安定性を判別し，安定ならば 0，不安定ならば 1 を出力.	[FALSE]

EXAMPLE

float 形式のファイル *data.f* を線形予測分析し，線形予測係数を反射係数に変換し，*data.rc* に出力する:

```
frame < data.f | window | lpc | lpc2par > data.rc
```

SEE ALSO

acorr, levdur, lpc, par2lpc, ltcdf

NAME

lsp2lpc – LSP から LPC に変換

SYNOPSIS

lsp2lpc [**–m** M] [**–s** S] [**–k**] [**–i** I] [***infile***]

DESCRIPTION

lsp2lpc は，lsp 係数を線形予測係数に変換します．ゲイン項を入出力しない場合はゲイン項を 1 とした LPC 係数

$$K = 1, a(1), \ldots, a(M)$$

を出力します.

OPTIONS

–m	M	LPC の次数.	[25]
–s	S	サンプリング周波数 (kHz).	[10]
–k		ゲインの入出力	[TRUE]
–i	I	入力形式.	[0]

0 規格化周波数 $(0 \ldots \pi)$
1 規格化周波数 $(0 \ldots 0.5)$
2 周波数 (kHz)
3 周波数 (Hz)

EXAMPLE

float 形式の 10 次の lsp 係数ファイル *data.lsp* から線形予測係数を求め，*data.lpc* に出力する:

```
lsp2lpc -m 10 < data.lsp > data.lpc
```

SEE ALSO

lpc, lpc2lsp

NAME

lspcheck – LSP の安定性のチェックと並べ替え

SYNOPSIS

lspcheck [**–m** *M*] [**–s** *S*] [**–k**] [**–i** *I*] [**–o** *O*] [**–r**] [***infile***]

DESCRIPTION

lspcheck は，lsp 係数の安定性を調べ，安定でない場合は係数の順序を並べ変えて安定係数とします.

データ形式は入力は float 形式で，出力は rearrange LSP を出力する場合は float 形式，それ以外は，安定でないフレーム番号を ASCII 形式で出力します.

OPTIONS

–m	*M*	LPC の次数.	[25]
–s	*S*	サンプリング周波数 (kHz).	[10]
–k		ゲインの入出力	[TRUE]
–i	*I*	入力形式.	[0]
–o	*O*	出力形式.	[0]
		0 規格化周波数 $(0 \ldots \pi)$	
		1 規格化周波数 $(0 \ldots 0.5)$	
		2 周波数 (kHz)	
		3 周波数 (Hz)	
–r		rearrange LSP	[FALSE]

EXAMPLE

float 形式の 10 次の lsp 係数ファイル *data.lsp* の安定性を調べ，安定でない係数は安定係数として，*data.lspr* に出力する:

```
lspcheck -r < data.lsp > data.lspr
```

SEE ALSO

lpc, lpc2lsp, lsp2lpc

NAME

lspdf – 音声合成のための LSP ディジタルフィルタ

SYNOPSIS

lspdf [**–m** M] [**–p** P] [**–i** I] [**–s** S] [**–o** O] [**–k**] ***lspfile*** [***infile***]

DESCRIPTION

lspdf は，入力データを *lspfile* の lsp 係数 $K, f(1), \ldots, f(M)$ をもつ lsp 合成フィルタによりフィルタリングし，標準出力に出力します.

データ形式は入力，出力とも float 形式です.

OPTIONS

–m	M	lsp 係数の次数.	[25]
–p	P	フレーム周期	[100]
–i	I	補間周期.	[1]
–s	S	サンプリング周波数 (kHz).	[10]
–o	O	入力形式.	[0]
		0 規格化周波数 $(0 \ldots \pi)$	
		1 規格化周波数 $(0 \ldots 0.5)$	
		2 周波数 (kHz)	
		3 周波数 (Hz)	
–k		ゲインを除いたシステム関数でフィルタリングする	[FALSE]

EXAMPLE

float 形式のピッチデータ *data.pitch* から励振源を作成し，lsp ファイル *data.lsp* により lsp 合成フィルタを駆動し，合成音声を *data.syn* に出力する:

```
excite < data.pitch | lspdf data.lsp > data.syn
```

SEE ALSO

lps, lpc2lsp

NAME

ltcdf – 音声合成のためのラティスフィルタ

SYNOPSIS

ltcdf [–**m** M] [–**p** P] [–**i** I] [–**k**] ***rcfile*** [***infile***]

DESCRIPTION

入力データを，*rcfile* の反射係数 $K, k(1), \ldots, k(M)$ をもつラティスフィルタによりフィルタリングし，標準出力に出力します.

データ形式は入力，出力とも float 形式です.

OPTIONS

–**m**	M	反射係数の次数.	[25]
–**p**	P	係数の更新周期.	[100]
–**i**	I	補間周期.	[1]
–**k**		ゲインを除いたシステム関数でフィルタリングする	[FALSE]

EXAMPLE

float 形式のピッチデータ *data.pitch* から励振源を作成し，PAECOR 係数ファイル *data.k* によりラティスフィルタを駆動して，合成音声を *data.syn* に出力する:

```
excite < data.pitch | ltcdf data.k > data.syn
```

SEE ALSO

lpc, acorr, levdur, lpc2par, par2lpc, poledf, zerodf, lspdf

NAME

mc2b – メルケプストラム係数から MLSA フィルタの係数を求める

SYNOPSIS

mc2b [–**a** A] [–**m** M] [***infile***]

DESCRIPTION

メルケプストラム係数 $c_\alpha(m)$ から MLSA フィルタの係数 $b(m)$ を求め，標準出力に出力します.

データ形式は入力，出力とも float 形式です.

メルケプストラム係数 $c_\alpha(m)$ から係数 $b(m)$ への変換式は

$$b(m) = \begin{cases} c_\alpha(M), & m = M \\ c_\alpha(m) - \alpha b(m+1), & 0 \le m < M \end{cases}$$

で与えられ，この係数 $b(m)$ を直接 MLSA フィルタの係数として用いることが出来ます．この変換式は b2mc の逆変換となります.

OPTIONS

–**a**	A	周波数圧縮パラメータ α.	[0.35]
–**m**	M	メルケプストラムの次数.	[25]

EXAMPLE

float 形式の音声ファイル *data.f* を 12 次でメルケプストラム分析し，得られたメルケプストラムを MLSA フィルタの係数に変換し，係数 $b(m)$ を *data.b* に出力する:

```
frame < data.f | window | mcep -m 12 | mc2b -m 12 > data.b
```

SEE ALSO

mlsadf, mglsadf, b2mc, mcep, mgcep, amcep

NAME

mcep – メルケプストラム分析 [10121012]

SYNOPSIS

mcep [–**a** A] [–**m** M] [–**l** L] [–**i** I] [–**j** J] [–**d** D] [–**e** E] [***infile***]

DESCRIPTION

メルケプストラム分析を行い，メルケプストラム係数 $c_\alpha(m)$ を標準出力に出力します．入力は長さ L の時系列

$$x(0), x(1), \ldots, x(L-1)$$

です．

データ形式は入力，出力とも float 形式です．

メルケプストラム分析では，音声のスペクトルを M 次のメルケプストラム $c_\alpha(m)$ により

$$H(z) = \exp \sum_{m=0}^{M} c_\alpha(m) \tilde{z}^{-m}$$

とモデル化し，対数スペクトルの不偏推定法における評価関数を適用します．ここで，$\tilde{z}^{-1}$ は 1 次のオールパス関数

$$\tilde{z}^{-1} = \frac{z^{-1} - \alpha}{1 - \alpha z^{-1}}$$

であり，その位相特性は標本化周波数 10kHz の場合 $\alpha = 0.35$，標本化周波数 8kHz の場合 $\alpha = 0.31$ と選べば，人間の音の高さに対する聴覚特性を表すメル尺度をよく近似します．

ここでは，評価関数を最小にするメルケプストラムを求めるために，Newton–Raphson 法が用いられています．

OPTIONS

–a	A	周波数圧縮パラメータ α.	[0.35]
–m	M	分析次数.	[25]
–l	L	フレーム長.	[256]

通常，以下のオプションの指定は必要ありません.

–i	I	Newton-Raphson 法の最小反復回数.	[2]
–j	J	Newton-Raphson 法の最大反復回数.	[30]
–d	D	Newton-Raphson 法の終了条件.	[0.001]
		$\varepsilon^{(i)}$ の繰り返しによる変化率が D 以内になったときに終了.	
–e	E	ピリオドグラムに足し込む小さな値.	[0.0]

EXAMPLE

float 形式の音声データ *data.f* を分析し，*data.mcep* にメルケプストラム係数を得る:

```
frame < data.f | window | mcep > data.mcep
```

SEE ALSO

uels, gcep, mgcep, mlsadf

NAME

merge – フレームへのデータの挿入

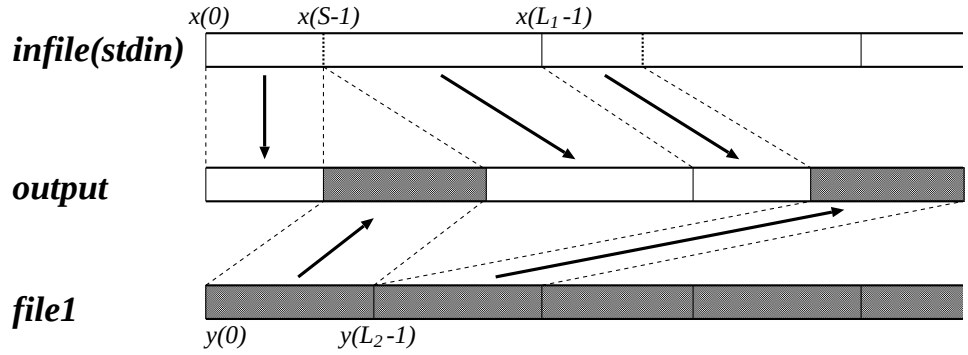
SYNOPSIS

merge [–**s** S] [–**l** L_1] [–**n** N_1] [–**L** L_2] [–**N** N_2]
[–**o**] [+***type***] ***file1*** [***infile***]

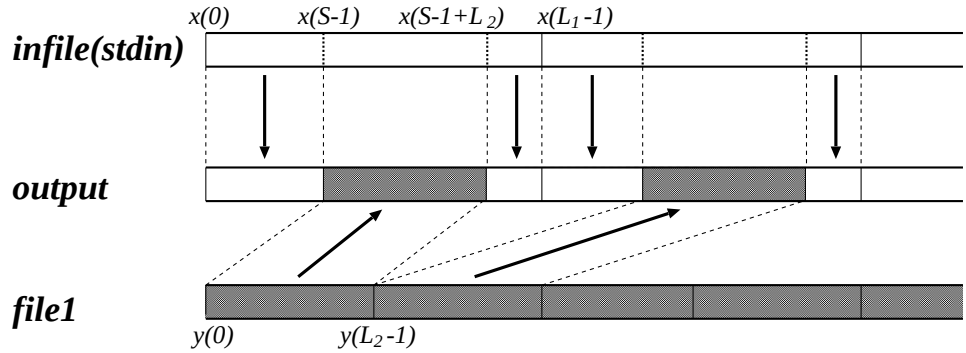
DESCRIPTION

標準入力から入力されたデータのフレーム毎に，指定されたファイルのフレームを挿入するか，そのフレームで書き換えます.

Insert mode



Overwrite mode



OPTIONS

–**s**	S	フレームの先頭からの挿入開始点，又は書き換え開始点.	[0]
–**l**	L_1	入力データのフレーム長.	[25]

–n	N_1	入力データの次数．入力データのフレーム長は N_1+1 になります.	$[L_1-1]$
–L	L_2	挿入するデータのフレーム長.	[10]
–N	N_2	挿入データの次数．挿入データのフレーム長は N_2+1 になります.	$[L_2-1]$
–o		上書き (書き換え) モード.	[FALSE]
$+t$		入力データの形式.	[f]

c	char 型 (1byte)	s	short 型 (2bytes)
i	int 型 (4bytes)	l	long 型 (4bytes)
f	float 型 (4bytes)	d	double 型 (8bytes)

EXAPMLE

short 型のファイル *data.f1* のフレーム長 3 の各フレームの 3 番目から，short 型のファイル *data.f2* のデータを 2 個づつ挿入して *data.merge* に出力する:

```
merge -s 2 -l 3 -L 2 +s data.f2 < data.f1 > data.merge
```

例えば，ファイル *data.f1* に $1,1,1,2,2,2,\cdots$，ファイル *data.f2* に $2,3,5,6,\cdots$ がそれぞれ入っていた場合，ファイル *data.merge* の中には，

$$1,1,2,3,1,\ 2,2,5,6,2,\cdots$$

が得られます。

long 型のファイル *data.f1* のフレーム長 4 の各フレームの 2 番目からを，long 型のファイル *data.f2* のデータで 2 個づつ書き換えて *data.merge* に出力する:

```
merge -s 1 -l 4 -L 2 +l -o data.f2 < data.f1 > data.merge
```

例えば，ファイル *data.f1* に $1,1,1,1,2,2,2,2,\cdots$，ファイル *data.f2* に $3,4,5,6,\cdots$ がそれぞれ入っていた場合，ファイル *data.merge* の中には

$$1,3,4,1,\ 2,5,6,2,\cdots$$

が得られます.

NAME

mgc2mgc – メル一般化対数変換

SYNOPSIS

mgc2mgc [–**m** M_1] [–**a** A_1] [–**g** G_1] [–**n**] [–**u**]
[–**M** M_2] [–**A** A_2] [–**G** G_2] [–**N**] [–**U**] [***infile***]

DESCRIPTION

入力されたメル一般化ケプストラム $c_{\alpha_1,\gamma_1}(0),\ldots,c_{\alpha_1,\gamma_1}(M_1)$ に対してメル一般化対数変換を行い，メル一般化ケプストラム $c_{\alpha_2,\gamma_2}(0),\ldots,c_{\alpha_2,\gamma_2}(M_2)$ を出力します.

データ形式は入力，出力とも float 形式です.

まず入力されたメル一般化ケプストラム $c_{\alpha_1,\gamma_1}(m)$ を周波数変換 ($\alpha_1 \rightarrow \alpha_2$) を行い，$c_{\alpha_2,\gamma_1}(m)$ を求めます.

$$\alpha = (\alpha_2 - \alpha_1)/(1 - \alpha_1\alpha_2)$$

$$c^{(i)}_{\alpha_2,\gamma_1}(m) = \left\{ \begin{array}{ll} c_{\alpha_1,\gamma_1}(-i) + \alpha\, c^{(i-1)}_{\alpha_2,\gamma_1}(0), & m = 0 \\ (1-\alpha^2)\, c^{(i-1)}_{\alpha_2,\gamma_1}(0) + \alpha\, c^{(i-1)}_{\alpha_2,\gamma_1}(1), & m = 1 \\ c^{(i-1)}_{\alpha_2,\gamma_1}(m-1) + \alpha \left(c^{(i-1)}_{\alpha_2,\gamma_1}(m) - c^{(i)}_{\alpha_2,\gamma_1}(m-1) \right), & m = 2,\ldots,M_2 \end{array} \right\}, \quad i = -M_1,\ldots,-1,0$$

ゲインの正規化をして，$c'_{\alpha_2,\gamma_1}(m)$ を求めます.

$$K_{\alpha_2} = s^{-1}_{\gamma_1}\left(c^{(0)}_{\alpha_2,\gamma_1}(0)\right),$$
$$c'_{\alpha_2,\gamma_1}(m) = c^{(0)}_{\alpha_2,\gamma_1}(m) / \left(1 + \gamma_1\, c^{(0)}_{\alpha_2,\gamma_1}(0)\right), \quad m = 1,2,\ldots,M_2$$

$c'_{\alpha_2,\gamma_1}(m)$ に一般化対数変換 ($\gamma_1 \rightarrow \gamma_2$) を行い，$c'_{\alpha_2,\gamma_2}(m)$ を求めます.

$$c'_{\alpha_2,\gamma_2}(m) = c'_{\alpha_2,\gamma_1}(m) + \sum_{k=1}^{m-1} \frac{k}{m} \Big(\gamma_2\, c_{\alpha_2,\gamma_1}(k)\, c'_{\alpha_2,\gamma_2}(m-k) - \gamma_1\, c_{\alpha_2,\gamma_2}(k)\, c'_{\alpha_2,\gamma_1}(m-k) \Big), \quad m = 1,2,\ldots,M_2$$

最後にゲインを逆正規化して，$c_{\alpha_2,\gamma_2}(m)$ を求めます.

$$c_{\alpha_2,\gamma_2}(0) = s_{\gamma_2}\left(K_{\alpha_2}\right),$$
$$c_{\alpha_2,\gamma_2}(m) = c'_{\alpha_2,\gamma_2}(m)\ \left(1 + \gamma_2\, c_{\alpha_2,\gamma_2}(0)\right), \quad m = 1,2,\ldots,M_2$$

γ を掛けた形で入出力する場合は，$c_{\alpha,\gamma}(m)$ が正規化されていない場合は，

$$1 + \gamma c_{\alpha,\gamma}(0), \gamma c_{\alpha,\gamma}(1), \ldots, \gamma c_{\alpha,\gamma}(M)$$

とし，正規化されている場合は

$$K_\alpha, \gamma c'_{\alpha,\gamma}(1), \ldots, \gamma c'_{\alpha,\gamma}(M)$$

とします.

OPTIONS

−m	M_1	入力されるメル一般化ケプストラムの次数.	[25]
−a	A_1	入力されるメル一般化ケプストラムの周波数圧縮パラメータ α_1.	[0]
−g	G_1	入力されるメル一般化ケプストラムのべきパラメータ γ_1. ただし，$G_1 > 1.0$ のときは $\gamma_1 = -1/G_1$.	[0]
−n		入力をゲインが正規化されたメル一般化ケプストラムとみなす.	[FALSE]
−u		入力を γ_1 を掛けた形とみなす.	[FALSE]
−M	M_2	出力されるメル一般化ケプストラムの次数.	[25]
−A	A_2	出力されるメル一般化ケプストラムの周波数圧縮パラメータ α_2.	[0]
−G	G_2	出力されるメル一般化ケプストラムのべきパラメータ γ_2. ただし，$G_2 > 1.0$ のときは $\gamma_2 = -1/G_2$.	[1]
−N		出力をゲインが正規化されたメル一般化ケプストラムとみなす.	[FALSE]
−U		出力を γ_2 を掛けた形とみなす.	[FALSE]

EXAMPLE

float 形式の 12 次の線形予測係数ファイル *data.lpc* から，30 次のメルケプストラムを求め，*data.mcep* に出力する:

```
mgc2mgc -m 12 -a 0 -g -1 -M 30 -A 0.31 -G 0
                        < data.lpc > data.mcep
```

SEE ALSO

uels, gcep, mcep, mgcep, gc2gc, freqt, lpc2c, c2lpc

NAME

mgc2sp – メル一般化ケプストラムから対数振幅スペクトルを求める

SYNOPSIS

mgc2sp [**–a** A] [**–g** G] [**–m** M] [**–n**] [**–u**] [**–l** L] [**–p**]
[**–o** O] [***infile***]

DESCRIPTION

入力されたメル一般化ケプストラム $c_{\alpha,\gamma}(m)$ から対数振幅スペクトルを求めます.

データ形式は入力，出力とも float 形式です.

メル一般化ケプストラム $c_{\alpha,\gamma}(m)$ をメル一般化対数変換（mgc2mgc 参照）によりケプストラムに変換し，ケプストラムから対数振幅スペクトルを求めます（spec 参照）.

ゲインで正規化されたものを入力する場合は、入力が

$$K_\alpha = s_\gamma^{-1}\left(c^{(0)}_{\alpha,\gamma}(0)\right),$$
$$c'_{\alpha,\gamma}(m) = c^{(0)}_{\alpha,\gamma}(m) / \left(1 + \gamma\, c^{(0)}_{\alpha,\gamma}(0)\right), \quad m = 1, 2, \ldots, M$$

の形であるとみなします.

また，γ を掛けた形で入力する場合は,
入力がゲインで正規化されていない場合,

$$1 + \gamma c_{\alpha,\gamma}(0), \gamma c_{\alpha,\gamma}(1), \ldots, \gamma c_{\alpha,\gamma}(M)$$

正規化されている場合,

$$K_\alpha, \gamma c'_{\alpha,\gamma}(1), \ldots, \gamma c'_{\alpha,\gamma}(M)$$

の形であるとみなします.

OPTIONS

–a	A	周波数圧縮パラメータ α.	[0]
–g	G	一般化ケプストラムのべきパラメータ γ . ただし，$G > 1.0$ のときは $\gamma = -1/G$.	[0]
–m	M	入力されるメル一般化ケプストラムの次数.	[25]

–n		入力を正規化されたメル一般化ケプストラムとみなす.	[FALSE]
–u		入力を γ を掛けた形とみなす.	[FALSE]
–l	L	FFT 長.	[256]
–p		スペクトルの位相を出力.	[FALSE]
–o	O	–p オプションが指定されていない場合, 出力するスペクトルのスケールを指定.	[0]

$O = 0$ $20 \times \log |H(z)|$
$O = 1$ $\ln |H(z)|$
$O = 2$ $|H(z)|$

–p オプションが指定されている場合, 出力する位相の単位を指定.

$O = 0$ $\arg |H(z)| \div \pi$ $[\pi\ rad.]$
$O = 1$ $\arg |H(z)|$ $[rad.]$
$O = 2$ $\arg |H(z)| \times 180 \div \pi$ $[deg.]$

EXAMPLE

float 形式のメル一般化ケプストラムファイル *data.mgcep* ($M = 12, \alpha = 0.35, \gamma = -0.5$) から対数振幅スペクトルを求め，表示する:

```
mgc2sp -m 12 -a 0.35 -r -0.5 < data.mgcep | glogsp |gr
```

SEE ALSO

c2sp, mgc2mgc, gc2gc, freqt, gnorm, lpc2c, c2lpc

NAME

mgcep – メル一般化ケプストラム分析 [13141314]

SYNOPSIS

mgcep [–**a** A] [–**g** G] [–**m** M] [–**l** L] [–**o** O]
[–**i** I] [–**j** J] [–**d** D] [–**p** P] [– **e** E] [***infile***]

DESCRIPTION

メル一般化ケプストラム分析を行います．分析結果は，–**o** オプションの指定に従った形式で，標準出力に出力します．入力は窓掛けされた長さ L の時系列

$$x(0), x(1), \ldots, x(L-1)$$

です．

データ形式は入力，出力とも float 形式です．

メル一般化ケプストラム分析では，音声のスペクトルを M 次のメル一般化ケプストラム $c_{\alpha,\gamma}(m)$ により

$$\begin{aligned} H(z) &= s_\gamma^{-1}\left(\sum_{m=0}^{M} c_{\alpha,\gamma}(m) z^{-m}\right) \\ &= \begin{cases} \left(1+\gamma\sum_{m=1}^{M} c_{\alpha,\gamma}(m)\tilde{z}^{-m}\right)^{1/\gamma}, & -1 \le \gamma < 0 \\ \exp\sum_{m=1}^{M} c_{\alpha,\gamma}(m)\tilde{z}^{-m}, & \gamma = 0 \end{cases} \end{aligned}$$

とモデル化し，対数スペクトルの不偏推定法における評価関数を適用します．ここで，$\tilde{z}^{-1}$ は 1 次のオールパス関数

$$\tilde{z}^{-1} = \frac{z^{-1}-\alpha}{1-\alpha z^{-1}}$$

であり，その位相特性は標本化周波数 10kHz の場合 $\alpha = 0.35$ ，標本化周波数 8kHz の場合 $\alpha = 0.31$ と選べば，人間の音の高さに対する聴覚特性を表すメル尺度をよく近似します．

ここでは，評価関数を最小にするメル一般化ケプストラムを求めるために，Newton–Raphson 法が用いられています．

メル一般化ケプストラム分析は，α 及び γ の値により，他のいくつかの音声分析法を特別な場合として含みます．これらの関係は図 1 のように表すことができます．

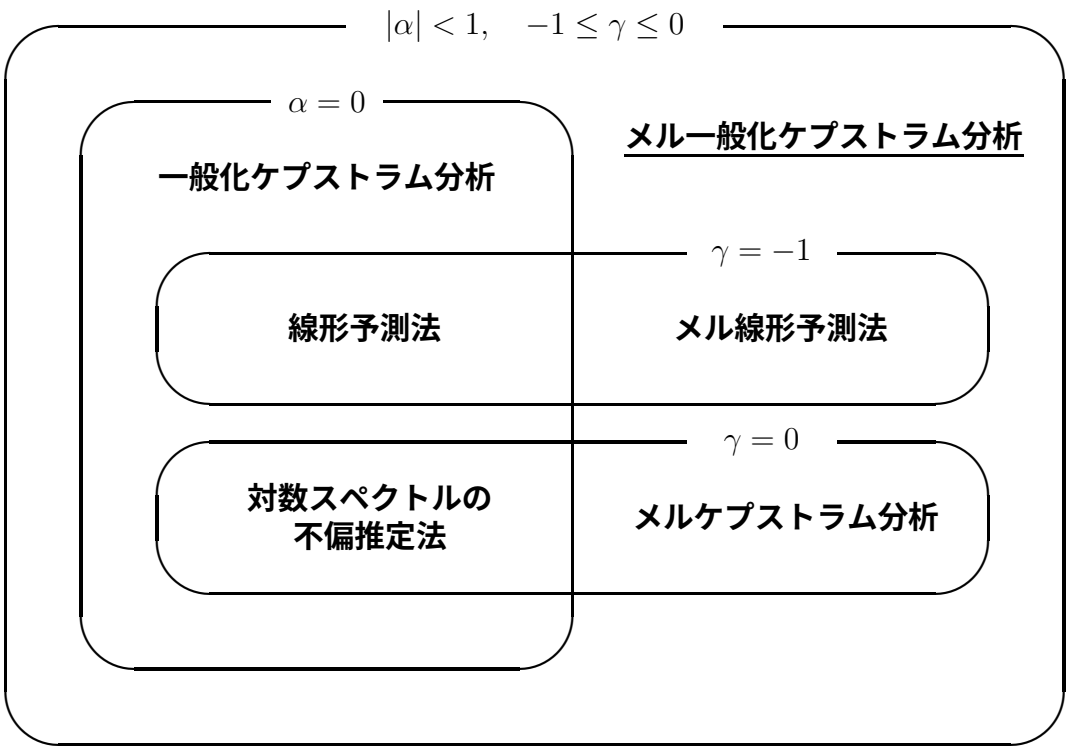


図 1: メル一般化ケプストラム分析と他手法との関係

OPTIONS

–a	A	周波数圧縮パラメータ α .	[0.35]
–g	G	一般化ケプストラムのべきパラメータ γ. ただし，$G > 1.0$ のときは $\gamma = -1/G$.	[0]
–m	M	分析次数.	[25]
–l	L	FFT の点数．2 のべき乗数で指定.	[256]
–o	O	$O = 0$ のとき	[0]

$$c_{\alpha,\gamma}(0), c_{\alpha,\gamma}(1), \ldots, c_{\alpha,\gamma}(M)$$

$O = 1$ のとき

$$b_{\gamma}(0), b_{\gamma}(1), \ldots, b_{\gamma}(M)$$

$O = 2$ のとき

$$K_{\alpha}, c'_{\alpha,\gamma}(1), \ldots, c'_{\alpha,\gamma}(M)$$

$O = 3$ のとき

$$K, b'_{\gamma}(1), \ldots, b'_{\gamma}(M)$$

$O = 4$ のとき

$$K_{\alpha}, \gamma\, c'_{\alpha,\gamma}(1), \ldots, \gamma\, c'_{\alpha,\gamma}(M)$$

$O = 5$ のとき

$$K, \gamma\, b'_{\gamma}(1), \ldots, \gamma\, b'_{\gamma}(M)$$

が標準出力に出力されます.

通常，以下のオプションの指定は必要ありません.

–i	I	Newton-Raphson 法の最小反復回数.	[2]
–j	J	Newton-Raphson 法の最大反復回数.	[30]
–d	D	Newton-Raphson 法の終了条件．デフォルトは $D = 0.001$ で，このとき $\varepsilon^{(i)}$ の繰り返しによる変化率が 0.001 つまり 0.1% 以内になったとき終了します.	[0.001]
–p	P	再帰式の打ち切り次数．特に，処理時間を短くしたい場合以外，P の指定は必要ありません.	[$L-1$]
–e	E	ピリオドグラムに足し込む小さな値	[0]

EXAMPLE

float 形式の音声データ *data.f* を $\gamma = 0$, $\alpha = 0$ (つまり対数スペクトルの不偏推定法) で分析し，*data.cep* にケプストラム係数を得る:

```
frame < data.f | window | mgcep -a 0 > data.cep
```

同様にメルケプストラム係数を *data.mcep* に得る:

```
frame < data.f | window | mgcep -a 0.35 > data.mcep
```

同様に線形予測係数を *data.lpc* に得る:

```
frame < data.f | window | mgcep -a 0 -g -1 -o 5 > data.lpc
```

この際，予測係数は,

$$K, a(1), a(2), \ldots, a(M)$$

の形式で得られます.

SEE ALSO

uels, gcep, mcep, freqt, gc2gc, mgc2mgc, gnorm, mglsadf

NAME

mgcep2 – 高速メル一般化ケプストラム分析 ($\gamma = -\frac{1}{2}$) [13141314]

SYNOPSIS

mgcep2 [–**a** A] [–**m** M] [–**l** L] [–**o** O] [–**i** I] [–**j** J] [–**d** D] [***infile***]

DESCRIPTION

メル一般化ケプストラム分析を行います．$\gamma = -\frac{1}{2}$ とすることで高速化しています．分析結果は，–**o** オプションの指定に従った形式で，標準出力に出力します．入力は窓掛けされた長さ L の時系列

$$x(0), x(1), \ldots, x(L-1)$$

です．

データ形式は入力，出力とも float 形式です．

メル一般化ケプストラム分析では，音声のスペクトルを M 次のメル一般化ケプストラム $c_{\alpha,\gamma}(m), \gamma = -\frac{1}{2}$ により

$$\begin{aligned} H(z) &= s_{-\frac{1}{2}}^{-1} \left(\sum_{m=0}^{M} c_{\alpha,-\frac{1}{2}}(m) z^{-m} \right) \\ &= \left(1 - \frac{1}{2} \sum_{m=1}^{M} c_{\alpha,-\frac{1}{2}}(m) \tilde{z}^{-m} \right)^{-2} \end{aligned}$$

とモデル化し，対数スペクトルの不偏推定法における評価関数を適用します．ここで，$\tilde{z}^{-1}$ は 1 次のオールパス関数

$$\tilde{z}^{-1} = \frac{z^{-1} - \alpha}{1 - \alpha z^{-1}}$$

であり，その位相特性は標本化周波数 10kHz の場合 $\alpha = 0.35$ ，標本化周波数 8kHz の場合 $\alpha = 0.31$ と選べば，人間の音の高さに対する聴覚特性を表すメル尺度をよく近似します．

ここでは，評価関数を最小にするメル一般化ケプストラムを求めるために，Newton–Raphson 法が用いられています．

OPTIONS

–a	A	周波数圧縮パラメータ α .	[0.35]
–m	M	分析次数.	[25]
–l	L	FFT の点数．2 のべき乗数で指定.	[256]
–o	O	$O = 0$ のとき	[0]

$$c_{\alpha,\gamma}(0), c_{\alpha,\gamma}(1), \ldots, c_{\alpha,\gamma}(M)$$

$O = 1$ のとき

$$b_{\gamma}(0), b_{\gamma}(1), \ldots, b_{\gamma}(M)$$

$O = 2$ のとき

$$K_{\alpha}, c'_{\alpha,\gamma}(1), \ldots, c'_{\alpha,\gamma}(M)$$

$O = 3$ のとき

$$K, b'_{\gamma}(1), \ldots, b'_{\gamma}(M)$$

$O = 4$ のとき

$$K_{\alpha}, \gamma\, c'_{\alpha,\gamma}(1), \ldots, \gamma\, c'_{\alpha,\gamma}(M)$$

$O = 5$ のとき

$$K, \gamma\, b'_{\gamma}(1), \ldots, \gamma\, b'_{\gamma}(M)$$

が標準出力に出力されます.

通常，以下のオプションの指定は必要ありません.

–i	I	Newton-Raphson 法の最小反復回数.	[2]
–j	J	Newton-Raphson 法の最大反復回数.	[30]
–d	D	Newton-Raphson 法の終了条件．デフォルトは $D = 0.001$ で，このとき $\varepsilon^{(i)}$ の繰り返しによる変化率が 0.001 つまり 0.1% 以内になったとき終了します.	[0.001]

EXAMPLE

float 形式の音声データ *data.f* を $\gamma = -\frac{1}{2}$, $\alpha = 0$ (つまり対数スペクトルの不偏推定法) で分析し，*data.cep* にケプストラム係数を得る:

```
frame < data.f | window | mgcep2 > data.cep
```

SEE ALSO

uels, gcep, mcep, mgcep, freqt, gc2gc, mgc2mgc, gnorm, mglsadf

NAME

mglsadf – 音声合成のための MGLSA フィルタ [19]

SYNOPSIS

mglsadf [–**m** M] [–**a** A] [–**g** G] [–**p** P] [–**i** I] [–**t**] [–**k**]
mgcfile [***infile***]

DESCRIPTION

入力データを *mgcfile* のメル一般化ケプストラム係数 $c_{\alpha,\gamma}(m)$ をもつ MGLSA フィルタによりフィルタリングし，標準出力に出力します．

データ形式は入力，出力とも float 形式です．

M 次のメル一般化ケプストラム $c_{\alpha,\gamma}(m)$ による合成フィルタの伝達関数 $H(z)$ は

$$\begin{aligned} H(z) &= s_\gamma^{-1}\left(\sum_{m=0}^{M} c_{\alpha,\gamma}(m)\tilde{z}^{-m}\right) \\ &= \begin{cases} \left(1+\gamma\sum_{m=0}^{M} c_{\alpha,\gamma}(m)\tilde{z}^{-m}\right)^{1/\gamma}, & 0<\gamma\le -1 \\ \exp\sum_{m=0}^{M} c_{\alpha,\gamma}(m)\tilde{z}^{-m}, & \gamma = 0 \end{cases} \end{aligned}$$

ただし，

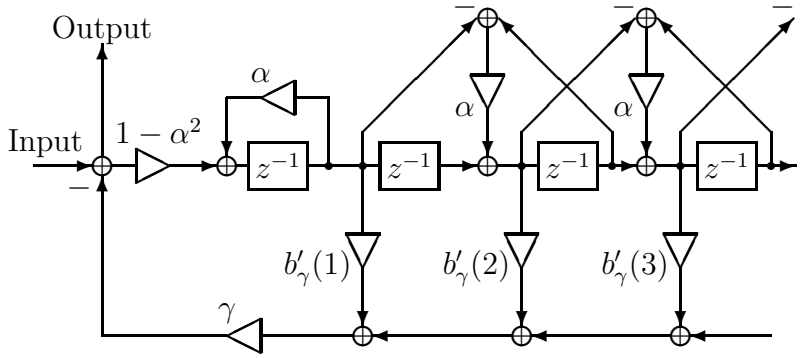
$$\tilde{z}^{-1} = \frac{z^{-1}-\alpha}{1-\alpha z^{-1}}$$

となります．ここで，$H(z)$ からゲイン K をくくり出すことにより

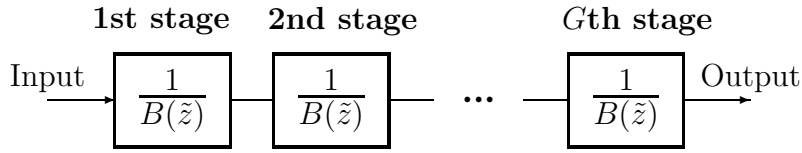
$$\begin{aligned} H(z) &= s_\gamma^{-1}\left(\sum_{m=0}^{M} b'_\gamma(m)\,\Phi_m(z)\right) \\ &= K\cdot D(z) \end{aligned}$$

ただし，

$$\Phi_m(z) = \begin{cases} 1, & m=0 \\ \dfrac{(1-\alpha^2)z^{-1}}{1-\alpha z^{-1}}\tilde{z}^{-(m-1)}, & m\ge 1 \end{cases}$$



(a) フィルタ $1/B(z)$ の構成



(b) $1/B(z)$ の G 段縦続接続構成

図 1: 合成フィルタ $D(z)$

および,

$$\begin{aligned} K &= s_\gamma^{-1}(b_\gamma(0)) \\ D(z) &= s_\gamma^{-1}\left(\sum_{m=1}^{M} b_\gamma(m)\,\Phi_m(z)\right) \end{aligned}$$

と変形します．また，係数 $b'_\gamma(m)$ は $c_{\alpha,\gamma}(m)$ を正規化し (gnorm 参照)，さらに線形変換をすることにより得ることができます (mc2b, b2mc 参照)．ここでは，べきパラメータが $\gamma = -1/G$ (G:自然数) のときのみを考えます．この場合，フィルタ $D(z)$ は図(b) のように，図(a) に示すフィルタ

$$\frac{1}{B(\tilde{z})} = \frac{1}{1+\gamma\sum_{m=1}^{M} b'_\gamma(m)\,\Phi_m(z)}$$

の G 段縦続構成で実現することができます.

OPTIONS

–m	M	メル一般化ケプストラムの次数.	[25]
–a	A	周波数圧縮パラメータ α.	[0.35]
–g	G	一般化ケプストラムのべきパラメータ $\gamma = -1/G$.	[1]
–p	P	係数の更新周期.	[100]
–i	I	係数の補間周期.	[1]
–t		転置型フィルタ.	[FALSE]
–k		ゲインを除いたシステム関数でフィルタリングする	[FALSE]

EXAMPLE

float 形式のピッチデータ *data.pitch* から励振源を作成し，メル一般化ケプストラムファイル *data.mgcep* により MGLSA フィルタを駆動し，合成音声を *data.syn* に出力する:

```
excite < data.pitch | mglsadf data.mgcep > data.syn
```

BUGS

n を自然数として，$\gamma = -1/n$ の場合にしか対応していない.

SEE ALSO

mgcep, poledf, zerodf, ltcdf, lmadf, mlsadf, glsadf

NAME

minmax – 最小値・最大値を求める

SYNOPSIS

minmax [–**l** L] [–**n** N] [–**b** B] [–**d**] [***infile***]

DESCRIPTION

ファイル *infile*（省略時は標準入力）から読み込んだ，フレーム毎の最小値と最大値を標準出力に出力します．上位 B 個の結果を出力します．

フレームの長さ L が 1 の時には，ファイル中のデータ全体の最小値と最大値を出力します．

データ形式は入力は float 形式で，出力は，データ番号を出力する場合は ascii 形式，値のみを出力する場合は float 形式となります．データ番号を出力する場合は，

$$value : position_0, position_1, \ldots$$

の形式で n 個の最小値，n 個の最大値を出力します．

OPTIONS

–**l**	L	ベクトルの長さ．	[1]
–**n**	N	ベクトルの次数．ベクトルの長さは $N+1$ になります．	[L-1]
–**b**	B	N-Best の値を出力	[1]
–**d**		最大最小のデータ番号の出力	[FALSE]

EXAMPLE

float 形式のファイル *data.f* のデータが

$$1, 1, 2, 3, 4, 5, 6, 7, 8, 9, 9, 10$$

の時，

```
minmax data.f -l 6 > data.m
```

とすると，ファイル *data.m* 中に，

$$1, 5, 6, 10$$

が得られます．また，

```
minmax -b 2 -d data.f
```

とすると,

```
1:0,1
2:2
10:11
9:9,10
```

が出力されます.

NAME

mlpg – 分布列から最尤パラメータ列を生成 [20212021]

SYNOPSIS

mlpg [–**l** L] [–**m** M] [–**d** (*fn* | d_0 [d_1 ...])] [–**r** N_R W_1 [W_2]]
[–**i** I] [–**s** S] [***infile***]

DESCRIPTION

mlpg は，指定されたファイルから (対角共分散) ガウス分布の列，つまり，平均ベクトルと共分散行列の対角成分の列

$$\ldots, \mu_t(0), \ldots, \mu_t(M), \mu_t^{(1)}(0), \ldots, \mu_t^{(1)}(M), \ldots, \mu_t^{(N)}(M),$$
$$\sigma_t^2(0), \ldots, \sigma_t^2(M), {\sigma^{(1)}}_t^2(0), \ldots, {\sigma^{(1)}}_t^2(M), \ldots, {\sigma^{(N)}}_t^2(M), \ldots$$

を読み込み，与えられた分布列に対して最も尤度の高いパラメータ列を求めて標準出力に出力します.

データ形式は入力，出力とも float 形式です.

フレーム t における音声パラメータベクトル $\boldsymbol{o}_t$ が，静的特徴ベクトル

$$\boldsymbol{c}_t = [c_t(0), c_t(1), \ldots, c_t(M)]'$$

とその動的特徴ベクトル $\Delta^{(1)}\boldsymbol{c}_t, \ldots, \Delta^{(N)}\boldsymbol{c}_t$ からなる，つまり

$$\boldsymbol{o}_t = [\boldsymbol{c}_t', \Delta^{(1)}\boldsymbol{c}_t', \ldots, \Delta^{(N)}\boldsymbol{c}_t']'$$

とします．ここで，動的特徴ベクトル $\Delta^{(n)}\boldsymbol{c}_t$ は静的特徴ベクトルと次式で関係づけられています.

$$\Delta^{(n)}\boldsymbol{c}_t = \sum_{\tau=-L^{(n)}}^{L^{(n)}} w^{(n)}(\tau)\boldsymbol{c}_{t+\tau}$$

この条件の下で，与えられた分布列 $((\boldsymbol{\mu}_1, \boldsymbol{U}_1), (\boldsymbol{\mu}_2, \boldsymbol{U}_2), \ldots, (\boldsymbol{\mu}_T, \boldsymbol{U}_T))$，ただし

$$\begin{aligned} \boldsymbol{\mu}_t &= [\boldsymbol{\mu}_t'^{(0)}, \boldsymbol{\mu}_t'^{(1)}, \ldots, \boldsymbol{\mu}_t'^{(N)}]' \\ \boldsymbol{U}_t &= \mathrm{diag}[\boldsymbol{U}_t^{(0)}, \boldsymbol{U}_t^{(1)}, \ldots, \boldsymbol{U}_t^{(1)}] \end{aligned}$$

に対して最も尤度の高いパラメータ列 $(\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_T)$ を求め，得られたパラメータベクトル $\boldsymbol{o}_t$ のうちの静的特徴ベクトル $\boldsymbol{c}_t$ の列 $(\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_T)$ を出力します．ここで，$\boldsymbol{\mu}^{(0)}, \boldsymbol{U}^{(0)}$ は静的特徴ベクトルに対する平均ベクトルと共分散行列，$\boldsymbol{\mu}^{(n)}, \boldsymbol{U}^{(n)}$ は n 次の動的特徴ベクトルに対する平均ベクトルと共分散行列です.

OPTIONS

–l L パラメータの長さ. [26]

–m M パラメータの次数. パラメータの長さは $M+1$ になります. [$L-1$]

–d (fn | d_0 [d_1 ...]) fn はデルタパラメータを計算する際の係数のファイル (float 形式) のファイル名. 係数は左右の長さが同じであると仮定しているため, 左右の長さが異なる場合は短い方に 0 を加える必要がある. 例えば, [N/A]

$$w(-1), w(0), w(1), w(2), w(3)$$

という係数を用いる場合は, 左側に 0 を加え,

$$0, 0, w(-1), w(0), w(1), w(2), w(3)$$

とする. ファイル名 fn を指定する代わりに係数 (ファイル fn の内容) を直接コマンドラインに書いても良い. 複数のデルタパラメータを用いる場合は繰り返し指定する.
–r オプションとの併用は不可.

–r N_R W_1 [W_2] デルタパラメータとして N_R 次までの回帰係数を使用する ($N_R = 1$ または 2). W_1, W_2 は一次または二次の回帰係数を求める際の (片側の) 幅を表す. 時刻 t における一次回帰係数 $\Delta \boldsymbol{c}_t$ は, [N/A]

$$\Delta \boldsymbol{c}_t = \frac{\sum_{\tau=-W_1}^{W_1} \tau \boldsymbol{c}_{t+\tau}}{\sum_{\tau=-W_1}^{W_1} \tau^2}$$

二次回帰係数 $\Delta^2 \boldsymbol{c}_t$ は, $a_2 = \sum_{\tau=-W_2}^{W_2} \tau^4$, $a_1 = \sum_{\tau=-W_2}^{W_2} \tau^2$, $a_0 = \sum_{\tau=-W_2}^{W_2} 1$ として

$$\Delta^2 \boldsymbol{c}_t = \frac{\sum_{\tau=-W_2}^{W_2} (a_0 \tau^2 - a_1) \boldsymbol{c}_{t+\tau}}{2(a_2 a_0 - a_1^2)}$$

により計算される.
–d オプションとの併用は不可.

–i I 入力のタイプを指定. [0]

$I = 0$ $(\boldsymbol{\mu}, \quad \boldsymbol{U} \quad)$
$I = 1$ $(\boldsymbol{\mu}, \quad \boldsymbol{U}^{-1})$
$I = 2$ $(\boldsymbol{\mu}\boldsymbol{U}^{-1}, \boldsymbol{U}^{-1})$

–s	S	あるフレームのパラメータが影響を及ぼすフレームの範囲.	[30]

EXAMPLE

パラメータの次数が 15，窓幅 1 の一次および二次の回帰係数を用いる場合に，分布列からパラメータ列を求める.

```
mlpg -m 15 -r 2 1 1 data.pdf > data.par
```

または,

```
echo "-0.5 0 0.5" | x2x +af > delta
echo "0.25 -0.5 0.25" | x2x +af > accel
mlpg -m 15 -d delta -d accel data.pdf > data.par
```

NAME

mlsadf – 音声合成のための MLSA フィルタ [18121812]

SYNOPSIS

mlsadf [–**m** M] [–**a** A] [–**p** P] [–**i** I] [–**b**] [–**P** Pa] [–**k**]
mcfile [***infile***]

DESCRIPTION

入力データを *mcfile* のメルケプストラム係数 $c_\alpha(0), c_\alpha(1), \ldots, c_\alpha(M)$ をもつ MLSA フィルタによりフィルタリングし，標準出力に出力します.

データ形式は入力，出力とも float 形式です.

MLSA フィルタは，M 次のメルケプストラム $c_\alpha(m)$ で表現された指数形伝達関数

$$H(z) = \exp \sum_{m=0}^{M} c_\alpha(m) \tilde{z}^{-m}$$

ただし、

$$\tilde{z}^{-1} = \frac{z^{-1} - \alpha}{1 - \alpha z^{-1}}$$

を高精度に近似します．ここで，伝達関数 $H(z)$ からゲイン K を

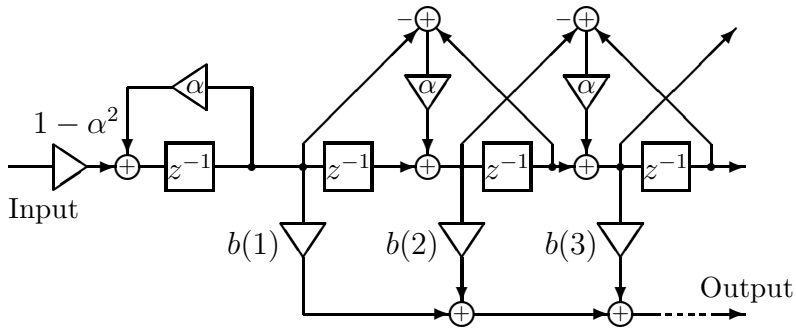
$$\begin{aligned} H(z) &= \exp \sum_{m=0}^{M} b(m) \Phi_m(z) \\ &= K \cdot D(z) \end{aligned}$$

ただし,

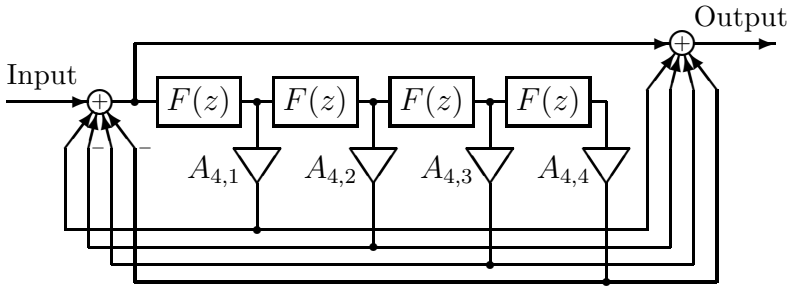
$$\Phi_m(z) = \begin{cases} 1, & m = 0 \\ \dfrac{(1-\alpha^2)z^{-1}}{1 - \alpha z^{-1}} \tilde{z}^{-(m-1)}, & m \geq 1 \end{cases}$$

および,

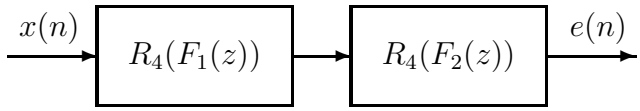
$$\begin{aligned} K &= \exp b(0) \\ D(z) &= \exp \sum_{m=1}^{M} b(m) \Phi_m(z) \end{aligned}$$



(a) Basic filter $F(z)$



(b) $R_L(F(z)) \simeq D(z) \quad L = 4$



(c) Two-stage cascade structure
$R_4(F_1(z)) \cdot R_4(F_2(z)) \simeq D(z)$

図 1: 指数形伝達関数 $1/D(z)$ の実現

のようにくくり出します．また，係数 $b(m)$ は $c_\alpha(m)$ と線形変換の関係にあります (mc2b, b2mc 参照)．

フィルタ $D(z)$ は次式の IIR フィルタ

$$F(z) = \sum_{m=1}^{M} b(m)\,\Phi_m(z)$$

を基礎フィルタとして (図1(a))，図1(b) のように実現することができます．ここで，基礎フィルタを図1(c) のように

$$F(z) = F_1(z) + F_2(z)$$

ただし，

$$F_1(z) = b(1)z^{-1}$$
$$F_2(z) = \sum_{m=2}^{M} b(m)\Phi_m(z)$$

と分割することにより，近似精度を上げています.

また，図1(b) の係数 $A_{4,l}$ はLMA フィルタと同じ値を用いています (lmadf 参照).

OPTIONS

–m	M	メルケプストラムの次数.	[25]
–a	A	周波数圧縮パラメータ α.	[0.35]
–p	P	係数の更新周期.	[100]
–i	I	係数の補間周期.	[1]
–b		係数ファイルが $b(m)$ (メルケプストラムを線形変換した係数) であることを指定.	[FALSE]
–P	Pa	パデ近似次数. Pa は 4 又は 5 を指定できます.	[4]
–k		ゲインを除いたシステム関数でフィルタリングする	[FALSE]

EXAMPLE

float形式のピッチデータ *data.pitch* から励振源を作成し，メルケプストラムファイル *data.mcep* により MLSA フィルタを駆動し，合成音声を *data.syn* に出力する:

```
excite < data.pitch | mlsadf data.mcep > data.syn
```

SEE ALSO

mcep, amcep, poledf, zerodf, ltcdf, lmadf, glsadf, mglsadf

NAME

msvq – 多段ベクトル量子化

SYNOPSIS

msvq [–**l** L] [–**n** N][–**s** S ***cbfile***] [–**q**] [***infile***]

DESCRIPTION

*msvq*は，指定されたファイルに対し，多段ベクトル量子化を行います．-s オプションはステージの段数回繰り返して指定します．

データ形式は入力は float 形式，出力は int 形式です．

OPTIONS

–**l**	L	ベクトル長.	[26]
–**n**	N	ベクトルの次数.	[$L-1$]
–**s**	S $cbfile$	コードブックの指定.	[N/A N/A]
		S コードブックサイズ	
		$cbfile$ コードブックファイル	
–**q**		量子化されたコードベクトルを出力.	[FALSE]

EXAMPLE

コードブックサイズ 256 のコードブック *cbfile1*と *cbfile2*による 2 段 VQ で，*data.f* を量子化し，*data.vq* に出力します:

```
msvq -s 256 cbfile1 -s 256 cbfile2 < data.f > data.vq
```

SEE ALSO

imsvq, vq, ivq, lbg

NAME

nan – データのチェック

SYNOPSIS

nan [–**d**] [**infile**]

DESCRIPTION

nan は，入力データ中の Nan または Infty & Indefinite Value の存在をチェックし，存在する場合はデータ番号を出力します．-d オプションを指定しない場合は，入力データ型は float 型です．

OPTIONS

–**d** 入力データを double 型にする． [False]

EXAMPLE

float 型のデータ *data.f* をチェックする:

```
nan data.f
```

NAME

norm0 – フィルタ係数を 0 次項で正規化する.

SYNOPSIS

norm0 [**−m** M] [***infile***]

DESCRIPTION

標準入力から入力されるデータ列

$$x(0), x(1), \ldots, x(M)$$

を 0 次の項で正規化

$$1, x(1)/x(0), \ldots, x(M)/x(0)$$

を行い，出力します.

データ形式は入力，出力とも float 形式です.

OPTIONS

−m M 入力データの次数. [25]

EXAMPLE

float 形式のファイル *data.f* から正規化した 15 次の自己相関係数を求め、*data.nacorr* に出力する:

```
frame < data.f | window | acorr -m 15 |\
norm0 -m 15 > data.nacorr
```

SEE ALSO

linear_intpl

NAME

nrand – 正規雑音を発生する.

SYNOPSIS

nrand [–**l** L] [–**s** S]

DESCRIPTION

nrand は，正規雑音を発生し，標準出力に出力します．内部では nrand と srnd を使っています.

出力データは，float 形式です.

OPTIONS

–**l**	L	発生する雑音系列の長さ.	[256]
		ただし，$L \leq 0$ の時は無限に生成します.	
–**s**	S	乱数初期化の seed.	[1]

EXAMPLE

長さ 100 の正規雑音系列を発生し *data.rnd* に出力する:

```
nrand -l 100 -s 3 > data.rnd
```

NAME

par2lpc – 反射係数から LPC に変換

SYNOPSIS

par2lpc [**–m** M] [***infile***]

DESCRIPTION

反射係数（PARCOR 係数）から線形予測係数を求めます．指定されたファイルから M 次の反射係数

$$K, k(1), \ldots, k(M)$$

を読み込み，線形予測係数

$$K, a(1), \ldots, a(M)$$

を標準出力に出力します．

データ形式は入力，出力とも float 形式です．

反射係数から線形予測係数を求める変換は，ダービンの解法の一部である式

$$\begin{aligned} a^{(m)}(m) &= k(m) \\ a^{(m)}(i) &= a^{(m-1)}(i) + k(m)a^{(m-1)}(m-i), \qquad 1 \le i \le m \end{aligned}$$

を使って， $m = 1, 2, \ldots, p$ の順に繰り返します．ここで、初期条件は

$$a^{(M)}(m) = a(m), \qquad 1 \le m \le M$$

です．

OPTIONS

–m M LPC の次数. [25]

EXAMPLE

float 形式の反射係数ファイル *data.rc* を線形予測係数に変換し、*data.lpc* に出力する:

```
par2lpc < data.rc > data.lpc
```

SEE ALSO

acorr, levdur, lpc, lpc2par

NAME

phase – 位相特性を求める

SYNOPSIS

phase [–**l** L] [–**p** ***pfile***] [–**z** ***zfile***] [–**m** M] [–**n** N] [***infile***]

DESCRIPTION

標準入力から入力された実数列のスペクトルの位相を計算します．つまり，入力数列を

$$x(0), x(1), \ldots, x(L-1)$$

として，

$$\begin{aligned} X_k &= X(e^{j\omega})\Big|_{\omega = \frac{2\pi k}{L}} \\ &= \sum_{m=0}^{L-1} x(m) e^{-j\omega m}\Big|_{\omega = \frac{2\pi\, k}{L}} \quad , \qquad k = 0, 1, \ldots, L-1 \end{aligned}$$

を FFT により計算し，

$$Y_k = \arg X_k, \qquad k = 0, 1, \ldots, L/2$$

を出力します．この際，位相はアンラップされます．出力されるデータは，角周波数 $0 \sim \pi$ に対応します．データ形式は入力，出力とも float 形式です．

–p, –z オプションが指定されたときには，指定されたファイルを係数としてもつディジタルフィルタの位相特性を，同様に出力します[1]．

OPTIONS

–l	L	FFT の点数．2 のべき乗数を指定．	[256]
–p	$pfile$	伝達関数の分母多項式の係数が書かれたファイル．ファイル内容は float 形式で次のように与えます． $K, a(1), \ldots, a(M)$	[NULL]

[1] この際，フィルタのインパルス応答から位相を求めるのではなく，分子係数，分母係数から別々に位相を計算し，その差によって全体の位相特性を求めています．

–z	*zfile*	伝達関数の分子多項式の係数が書かれたファイル．フ ァイルの内容は float 形式で次のように与えます。 $b(0), b(1), \ldots, b(N)$ *pfile*, *zfile* の内容は，*dfs* の場合と同じです．**–p**、**–z** オプションの内，**–p** オプションだけが指定された場合，分子多項式は 1 として扱われ，**–z** オプションだけが指定された場合には，分母多項式，ゲイン K とも 1 として扱われます．**–p**，**–z** オプションいずれも省略された場合には，データは標準入力から読まれます．	[NULL]
–m	M	伝達関数の分子多項式の次数．実際に入力されたデータ数が，$M+1$ より少なかった場合には，$M =$ (実際の入力データ数) -1 となるので，長さ $M+1$ のデータを何組か続けて入力する場合以外は，M の指定は必要ありません．	$[L-1]$
–n	N	伝達関数の分母多項式の次数．**–m** オプションと同様，実際に入力されたデータ数が，$N+1$ より少なかった場合には，$N =$ (実際の入力データ数) -1 となるので，長さ $N+1$ のデータを何組か続けて入力する場合以外は，N の指定は必要ありません．	$[L-1]$
–u		アンラップ．	[TRUE]

EXAMPLE

float 形式のファイル *data.p, data.z* で指定された係数をもつディジタルフィルタの位相特性を表示する:

```
phase -p data.p -z data.z | fdrw | xgr
```

data.p, *data.z* の表すフィルタが安定な場合には,

```
impulse | dfs -p data.p -z data.z | phase | fdrw | xgr
```

としてもほぼ同様の結果が得られます.

SEE ALSO

spec, fft, fftr, dfs

BUGS

サンプル間隔が粗い場合 (–l オプションの指定値が小さい)，位相特性が急峻な場合 (極，零点が z 平面の単位円付近にある場合など) には，位相のアンラップがうまく行われないことがあります.

NAME

pitch – ピッチ抽出

SYNOPSIS

pitch [–**s** S] [–**l** L] [–**t** T] [–**L** Lo] [–**H** Hi] [–**e** E]
[–**i** I] [–**j** J] [–**d** D] [***infile***]

DESCRIPTION

ケプストラム法により抽出したピッチ $p(t)$ を標準出力に出力します．入力は窓掛けされた長さ L の時系列

$$x(0), x(1), \ldots, x(L-1)$$

です．

データ形式は入力，出力とも float 形式です．

有声音と無声音の識別は，不偏推定法により求めた $S/10 \times 25$ 次のケプストラムから得られる対数スペクトル包絡 $\hat{g}_i(\Omega_k)$ の基本周波数帯域部分の平均値 v_i を利用します．

$$v_i = \frac{1}{14n} \sum_{k=4n}^{17n} \hat{g}_i(\Omega_k), \qquad (\Omega_k = \frac{2\pi k}{N}, n = N/256)$$

ここで，FFT サイズ N は L より大きい 2 のべき乗とします．

有声音 $(v_i > T)$ の場合は，FFT ケプストラム $c(m)$ を $c(m) \times m$ と変換し、周波数 Lo (Hz) から Hi (Hz) の間のピークを求めます．無声音 $(vi < T)$ の場合は 0 を出力します．

OPTIONS

–**s**	S	サンプリング周波数 (kHz).	[10]
–**l**	L	入力データのフレーム長.	[400]
–**t**	T	有声音/無声音の閾値.	[6.0]
–**L**	Lo	基本周波数の最小値 (Hz).	[60]
–**H**	Hi	基本周波数の最大値 (Hz).	[240]
–**e**	E	パワースペクトルの対数をとる際に足し込む小さな値.	[0.0]
–**i**	I	不偏推定の最小反復回数.	[2]
–**j**	J	不偏推定の最大反復回数.	[30]
–**d**	D	不偏推定の終了条件.	[0.1]

EXAMPLE

float 形式の音声データ (サンプリング周波数 10kHz) *data.f* をからピッチを抽出し, *data.pitch* に出力する:

```
frame -l 400 < data.f | window -l 400 | pitch -l 400 > data.pitch
```

SEE ALSO

excite

NAME

poledf – 音声合成のための全極標準型フィルタ

SYNOPSIS

poledf [**–m** M] [**–p** P] [**–i** I] [**–t**] [**–k**] ***afile*** [***infile***]

DESCRIPTION

入力データを *afile* の線形予測係数 $K, a(1), \ldots, a(M)$ をもつ全極標準形フィルタによりフィルタリングし，標準出力に出力します.

データ形式は入力，出力とも float 形式です.

全極標準形フィルタの伝達関数 $H(z)$ は,

$$H(z) = \frac{K}{1 + \sum_{m=1}^{M} a(m) z^{-m}}$$

です.

OPTIONS

–m	M	フィルタの次数.	[25]
–p	P	係数の更新周期.	[100]
–i	I	係数の補間周期.	[1]
–t		転置型フィルタ.	[FALSE]
–k		ゲインを除いたシステム関数でフィルタリングする	[FALSE]

EXAMPLE

float 形式のピッチデータ *data.pitch* から励振源を作成し，線形予測係数ファイル *data.lpc* により標準形合成フィルタを駆動し，合成音声を *data.syn* に出力する:

```
excite < data.pitch | poledf data.lpc > data.syn
```

SEE ALSO

lpc, acorr, ltcdf, lmadf, zerodf

NAME

psgr – プロッタコマンド列を PostScript コードに変換

SYNOPSIS

psgr [–**t** ***title***] [–**s** *S*] [–**c** *C*] [–**x** *X*] [–**y** *Y*] [–**p** **P**] [–**r** *R*] [–**b**]
[–**T** *T*] [–**B** *B*] [–**L** *L*] [–**R** *R*] [–**P**] [***infile***]

DESCRIPTION

標準入力からプロッタコマンド列を読み込んで，グラフ出力を PostScript (EPSF or PS) コードに変換し標準出力に出力します.

OPTIONS

–**t**	*title*	タイトル.	[NULL]
–**s**	*S*	描画サイズ縮小比率.	[1.0]
–**c**	*C*	コピー部数.	[1]
–**x**	*X*	x 軸のシフト. (mm)	[0]
–**y**	*Y*	y 軸のシフト. (mm)	[0]
–**p**	*P*	紙のサイズ. (Letter, A3, A4, A5, B4, B5)	[A4]
–**l**		ランドスケープ.	[FALSE]
–**r**	*R*	解像度. (dpi)	[600]
–**b**		ボールドフォント.	[FALSE]
–**T**	*T*	上部マージン. (mm)	[0]
–**B**	*B*	下部マージン. (mm)	[0]
–**L**	*L*	左側マージン. (mm)	[0]
–**R**	*R*	右側マージン. (mm)	[0]
–**P**		PostScript code を出力.	[FALSE]

EXAMPLE

fig で *data.fig* に含まれる図を描き、プリンターに出力する:

```
fig data.fig | psgr | lpr
```

BUGS

- Y 軸名などの周辺部が欠ける可能性がありますユーザー側での対処法として，マージンによる修正を行なって下さい．
- シュリンクによって描画サイズを変更した場合，TEX への取り込みの際に正しく反映されない場合があるかも知れません．そのような場合には，TEX 側の取り込み時のオプションで処理して下さい．

SEE ALSO

fig, fdrw, xgr

NAME

ramp – ランプ列を発生する

SYNOPSIS

ramp [–**l** L] [–**n** N] [–**s** S] [–**e** E] [–**t** T]

DESCRIPTION

ランプ列を標準出力に出力します．つまり，

$$\underbrace{S, S+T, S+2T, \ldots, S+(L-1)T}_{L}$$

を出力します．

出力データは，float 形式です．最終値を指定した場合は，

$$\underbrace{S, S+T, S+2T, \ldots, E}_{(E-S)/T}$$

となります．-l オプションと-e オプションと-n オプションを 2 つ以上同時に指定した場合は，後で指定したものに従います．

OPTIONS

–**l**	L	ランプ列の長さ． ただし，$L \leq 0$ の時は無限に生成する．	[256]
–**n**	N	ランプ列の次数．	[L-1]
–**s**	S	初期値．	[0]
–**e**	E	最終値．	[N/A]
–**t**	T	ステップサイズ．	[1]

EXAMPLE

数列

$$y(n) = \exp(-n)$$

を出力する:

```
ramp | sopr -m -1 -E | dmp
```

SEE ALSO

impulse, step, train, sin

NAME

reverse – データの順番を反対に並び変える

SYNOPSIS

reverse [–**l** *L*] [–**n** *N*] [***infile***]

DESCRIPTION

reverse は，float 型で与えられるデータ列を標準入力から読み込み，オプションで指定されるブロック長に区切ってデータ列を反転させ，標準出力に出力します．ブロック長を指定しなかった場合はデフォルトでファイル全体が指定されます．ファイル全体のサイズがブロック長で丁度割り切れない場合は余りのブロックは無視され，出力もされません．

OPTIONS

–**l**	*L*	ブロック長.	[ファイル全体]
–**n**	*N*	ブロックの次数.	[ファイル全体-1]

EXAMPLE

float 形式のファイル *data.in* には以下のデータが入っているとする．

$$\underbrace{0.0,\ 1.0,\ 2.0},\ \underbrace{3.0,\ 4.0,\ 5.0},\ \underbrace{6.0,\ 7.0,\ 8.0},\ 9.0$$

ここで

```
reverse -l 3 data.in > data.out
```

とするとファイル *data.out* には以下のデータが入る．

$$\underbrace{2.0,\ 1.0,\ 0.0},\ \underbrace{5.0,\ 4.0,\ 3.0},\ \underbrace{8.0,\ 7.0,\ 6.0}$$

NAME

rmse – RMSE を計算する

SYNOPSIS

rmse [–l L] ***file1*** [***infile***]

DESCRIPTION

指定されたファイルから長さ L の 2 つの時系列

$$\underbrace{x_1(0), x_1(1), \ldots, x_1(L-1)}, \underbrace{x_2(0), x_2(1), \ldots}$$

と

$$\underbrace{y_1(0), y_1(1), \ldots, y_1(L-1)}, \underbrace{y_2(0), y_2(1), \ldots}$$

を読み込み，その RMSE (Root Mean Square Error)

$$\mathrm{RMSE}_j = \sqrt{\sum_{m=0}^{L-1} (x_j(m) - y_j(m))^2 / L}$$

を出力します．データ形式は入力，出力とも float 形式です．

OPTIONS

–l L RMSE をとるデータ長. [0]
$L = 0$ なら，ファイル全体の RMSE を出力.

EXAMPLE

float 形式のファイル *data.f1* と *data.f2* の RMSE を計算し、その最大値と最小値を出力する:

```
rmse -l 26 data.f1 data.f2 | minmax | dmp
```

SEE ALSO

histogram, minmax

NAME

rndpg – 分布列に従ってランダムにパラメータを生成

SYNOPSIS

rndpg [–**l** L] [–**m** M] [–**d** (*fn* | d_0 [d_1 ...])] [–**r** N_R W_1 [W_2]]
[–**i** I] [–**s** S] [–**R**] [***infile***]

DESCRIPTION

rndpg は，指定されたファイルから (対角共分散) ガウス分布の列，つまり，平均ベクトルと共分散行列の対角成分の列

$$\ldots, \mu_t(0), \ldots, \mu_t(M), \mu_t^{(1)}(0), \ldots, \mu_t^{(1)}(M), \ldots, \mu_t^{(N)}(M),$$
$$\sigma_t^2(0), \ldots, \sigma_t^2(M), {\sigma^{(1)}}_t^2(0), \ldots, {\sigma^{(1)}}_t^2(M), \ldots, {\sigma^{(N)}}_t^2(M), \ldots$$

を読み込み，与えられた分布列に従ってランダムにパラメータ列を生成し標準出力に出力します.

データ形式は入力，出力とも float 形式です.

OPTIONS

–**l**	L	パラメータの長さ.	[26]
–**m**	M	パラメータの次数. パラメータの長さは $M+1$ になります.	[$L-1$]

–d $(fn \mid d_0\ [d_1\ \ldots])$ fn はデルタパラメータを計算する際の係数のファイル (float 形式) のファイル名．係数は左右の長さが同じであると仮定しているため，左右の長さが異なる場合は短い方に 0 を加える必要がある．例えば， [N/A]

$$w(-1), w(0), w(1), w(2), w(3)$$

という係数を用いる場合は，左側に 0 を加え，

$$0, 0, w(-1), w(0), w(1), w(2), w(3)$$

とする．ファイル名 fn を指定する代わりに係数 (ファイル fn の内容) を直接コマンドラインに書いても良い．複数のデルタパラメータを用いる場合は繰り返し指定する．
–r オプションとの併用は不可．

–r $N_R\ W_1\ [W_2]$ デルタパラメータとして N_R 次までの回帰係数を使用する ($N_R = 1$ または 2)．W_1，W_2 は一次または二次の回帰係数を求める際の (片側の) 幅を表す．時刻 t における一次回帰係数 $\Delta \boldsymbol{c}_t$ は， [N/A]

$$\Delta \boldsymbol{c}_t = \frac{\sum_{\tau=-W_1}^{W_1} \tau \boldsymbol{c}_{t+\tau}}{\sum_{\tau=-W_1}^{W_1} \tau^2}$$

二次回帰係数 $\Delta^2 \boldsymbol{c}_t$ は，$a_2 = \sum_{\tau=-W_2}^{W_2} \tau^4$，$a_1 = \sum_{\tau=-W_2}^{W_2} \tau^2$，$a_0 = \sum_{\tau=-W_2}^{W_2} 1$ として

$$\Delta^2 \boldsymbol{c}_t = \frac{\sum_{\tau=-W_2}^{W_2} (a_0 \tau^2 - a_1) \boldsymbol{c}_{t+\tau}}{2(a_2 a_0 - a_1^2)}$$

により計算される．
–d オプションとの併用は不可．

–i I 入力のタイプを指定． [0]
$I = 0$ $(\boldsymbol{\mu},\ \ \boldsymbol{U}\ \ )$
$I = 1$ $(\boldsymbol{\mu},\ \ \boldsymbol{U}^{-1})$
$I = 2$ $(\boldsymbol{\mu}\boldsymbol{U}^{-1}, \boldsymbol{U}^{-1})$

–s S あるフレームのパラメータが影響を及ぼすフレームの範囲． [30]

–R ランダムなパラメータ列を生成する．FALSE の場合は分布列に対して最尤のパラメータ列を生成する． [TRUE]

EXAMPLE

パラメータの次数が 15，窓幅 1 の一次および二次の回帰係数を用いる場合に，分布列からパラメータ列を求める．

```
rndpg -m 15 -r 2 1 1 data.pdf > data.par
```

または，

```
echo "-0.5 0 0.5" | x2x +af > delta
echo "0.25 -0.5 0.25" | x2x +af > accel
rndpg -m 15 -d delta -d accel data.pdf > data.par
```

NAME

root_pol – 多項式の根を求める

SYNOPSIS

root_pol [−**m** M] [−**n** N] [−**e** E] [−**i**] [−**s**] [−**r**] [***infile***]

DESCRIPTION

多項式の根を求めます．指定されたファイルから n 次の多項式

$$P(x) = a_0 x^n + a_1 x^{n-1} + \ldots + a_{n-1} x + a_n$$

の係数

$$a_0, a_1, \ldots, a_n$$

を読み込み，解を Durand-Kerner-Aberth 法により計算します．
結果は，$P(x)$ の根を z_i として複素形式

$$\begin{aligned} &\mathrm{Re}[z_0], \quad \mathrm{Im}[z_0] \\ &\mathrm{Re}[z_1], \quad \mathrm{Im}[z_1] \\ &\quad \vdots \\ &\mathrm{Re}[z_{n-1}], \quad \mathrm{Im}[z_{n-1}] \end{aligned}$$

あるいは，極形式

$$\begin{aligned} &|z_0|, \quad \arg[z_0] \\ &|z_1|, \quad \arg[z_1] \\ &\quad \vdots \\ &|z_{n-1}|, \quad \arg[z_{n-1}] \end{aligned}$$

の形式で標準出力に出力します．
データ形式は入力，出力とも float 形式です．

OPTIONS

−m	M	多項式の次数.	[32]
−n	N	根を探索する際の最大反復回数.	[1000]
−e	E	根の許容誤差 ε.	[10^{-14}]
−i		$a_0 = 1$ にする.	[FALSE]
−s		係数を逆順にする．つまり，根の位置 z_0 を $1/z_0^*$ と置き換えることを意味する.	[FALSE]
−r		出力する根を極形式（絶対値, 偏角）にする.	[（実部, 虚部）]

EXAMPLE

float 形式の多項式ファイル *data.z* の根を極座標表示で求め，表示する:

```
root -r < data.z | x2x +a 2
```

NAME

sin – 正弦列を発生する

SYNOPSIS

sin [**–l** L] [**–p** P] [**–m** M]

DESCRIPTION

周期 P で，長さ L ，高さ M の離散正弦列

$$x(n) = M \sin(\frac{2\pi}{P} n)$$

を標準出力に出力します.

出力データは，float 形式です.

OPTIONS

–l	L	正弦波の長さ.	[256]
		ただし，$L \leq 0$ なら無限に生成します.	
–p	P	正弦波の周期.	[10.0]
–m	M	正弦波の振幅.	[1.0]

EXAMPLE

正弦波にブラックマン窓をかけた波形を画面に表示する:

```
sin -p 12.3 | window | fdrw | xgr
```

SEE ALSO

impulse, step, train, ramp

NAME

smcep – 2 次オールパス関数を用いたメルケプストラム分析 [15]

SYNOPSIS

smcep [–**a** A] [–**t** T] [–**m** M] [–**l** L]
[–**i** I] [–**j** J] [–**d** D] [–**e** E] [***infile***]

DESCRIPTION

2 次オールパス関数を 1/2 乗したシステム関数 $A(z)$

$$A(z) = \left(\frac{z^{-2} - 2\alpha \cos\theta z^{-1} + \alpha^2}{1 - 2\alpha \cos\theta z^{-1} + \alpha^2 z^{-2}} \right)^{\frac{1}{2}} \tilde{z}^{-1} = \frac{z^{-1} - \alpha}{1 - \alpha z^{-1}}$$

を用いたメルケプストラム分析を行い，メルケプストラム係数 $c(m)$ を標準出力に出力します．入力は長さ L の時系列

$$x(0), x(1), \ldots, x(L-1)$$

です．

データ形式は入力，出力とも float 形式です．

2 次オールパス関数を用いたメルケプストラム分析では，音声のスペクトルを M 次のメルケプストラム $c(m)$ により以下のようにモデル化します．

$$H(z) = \exp \sum_{m=0}^{M} c(m)\, B_m(e^{j\omega})$$

ただし，

$$\mathrm{Re}\left[B_m(e^{j\omega}) \right] = \left\{ A^m(e^{j\omega}) + A^m(e^{-j\omega}) \right\} / 2$$

です．

ここでは，評価関数を最小にするメルケプストラムを求めるために，Newton–Raphson 法が用いられています．

OPTIONS

–a	A	周波数圧縮パラメータ α.	[0.35]
–t	T	強調する周波数位置 $\theta * \pi$(rad).	[0]
–m	M	分析次数.	[25]
–l	L_1	フレーム長.	[256]
–L	L_2	IFFT サイズ.	[256]

通常，以下のオプションの指定は必要ありません.

–i	I	Newton-Raphson 法の最小反復回数.	[2]
–j	J	Newton-Raphson 法の最大反復回数.	[30]
–d	D	Newton-Raphson 法の終了条件.	[0.001]
		$\varepsilon^{(i)}$ の繰り返しによる変化率が D 以内になったときに終了.	
–e	E	ピリオドグラムに足し込む小さな値.	[0.0]

EXAMPLE

float 形式の音声データ *data.f* を分析し，*data.mcep* にメルケプストラム係数を得る:

```
frame < data.f | window | smcep > data.mcep
```

SEE ALSO

uels, gcep, mcep, mgcep, mlsadf

NAME

snr – SNR とセグメンタル SNR を求める

SYNOPSIS

snr [–**l** L] [$+t_1t_2$] [–**o** O] ***file1*** [***infile***]

DESCRIPTION

指定された 2 つのファイルからの入力

$$x_1(0), x_1(1), \ldots, x_1(L-1)$$

と

$$x_2(0), x_2(1), \ldots, x_2(L-1)$$

から SNR(Signal Noise Ratio) と $\mathrm{SNR}_{\mathrm{seg}}$(セグメンタル SNR) を計算します.

SNR, $\mathrm{SNR}_{\mathrm{seg}}$ は次式で計算することができます.

$$\mathrm{SNR} = 10 \log \frac{\sum_n (x(n))^2}{\sum_n (e(n))^2} \quad [\mathrm{dB}]$$

$$\mathrm{SNR}_{\mathrm{seg}} = \frac{1}{N_i} \sum_{i=1}^{N_i} \mathrm{SNR}_i \quad [\mathrm{dB}]$$

ただし,

$$e(n) = x_1(n) - x_2(n)$$

ここで, N_i はフレーム数を表しています. セグメンタル SNR は子音区間のような比較的小振幅区間の SNR も反映されるため, 主観値との対応が比較的良いという特徴があります.

OPTIONS

–l *L* セグメンタルSNRを計算するときのフレーム長 [256]

+t_1t_2 入力データの形式．t_1，t_2 はそれぞれ *file1*，*infile* のデータ型を表す． [sf]

s short型 (2bytes) f float型 (4bytes)

–o *O* 出力形式. [0]

0 SNR and SNRseg
1 SNR and SNRseg in detail
2 SNR
3 SNRseg

0,1 はASCII形式で出力し 2,3 はfloat型で出力します

EXAMPLE

short形式のファイル *data.s* と float形式のファイル *data.f* のSNRとセグメンタルSNRを出力する:

```
snr +sf data.s data.f
```

SEE ALSO

histogram, average, rmse

NAME

sopr – スカラ演算を行う

SYNOPSIS

sopr [–**a** A] [–**s** S] [–**m** M] [–**d** D] [–**ABS**] [–**INV**]
[–**P**] [–**R**] [–**SQRT**] [–**LN**] [–**LOG10**] [–**EXP**] [–**POW10**]
[–**FIX**] [–**UNIT**] [–**CLIP**] [–**SIN**] [–**COS**] [–**TAN**]
[–**ATAN**] [–**r m**n] [–**w m**n] [***infile***]

DESCRIPTION

指定されたファイルから入力 x を読み込み，指定された演算の実行結果を標準出力に出力します．*infile* の指定がないときには，標準入力からデータが読み込まれます．

データ形式は入力，出力とも float 形式です．

演算は与えられた順番に実行します．

OPTIONS

–**a**	A	加算．$y = x + A$.	[FALSE]
–**s**	S	減算．$y = x - S$.	[FALSE]
–**m**	M	乗算．$y = x * M$.	[FALSE]
–**d**	D	除算．$y = x/D$.	[FALSE]

演算オプションの引数として “*dB*” を指定すると $20/\log_e 10$ の値が設定されます．また，同様に “*pi*” を指定すると π の値が設定され，“*ln2*”，“*exp10*”，“*sqrt30*” というように，特定の関数の引数に任意の数字を書いても，それを計算して定数とします．

–**ABS**	絶対値．$y = \|x\|$.	[FALSE]
–**INV**	逆数．$y = 1/x$.	[FALSE]
–**P**	二乗．$y = x^2$.	[FALSE]
–**R**	平方根．$y = \sqrt{x}$.	[FALSE]
–**SQRT**	平方根．$y = \sqrt{x}$.	[FALSE]
–**LN**	自然対数．$y = \log x$.	[FALSE]
–**LOG10**	常用対数．$y = \log_{10} x$.	[FALSE]
–**EXP**	指数．$y = \exp x$.	[FALSE]

–POW10		10 のべき乗．$y = 10^x$.	[FALSE]
–FIX		小数点以下の切捨て．$(int)x$.	[FALSE]
–UNIT		ユニットステップ．$u(x)$	[FALSE]
–CLIP		クリッピング．$x * u(x)$	[FALSE]
–SIN		正弦．$y = \sin(x)$	[FALSE]
–COS		余弦．$y = \cos(x)$	[FALSE]
–TAN		正接．$y = \tan(x)$	[FALSE]
–ATAN		逆正接．$y = \mathrm{atan}(x)$	[FALSE]
–r	mn	メモリレジスタ mn からの読み込み ($n = 0..9$)	
–w	mn	メモリレジスタ mn への書き込み ($n = 0..9$)	

EXAMPLE

ランプ関数 $(0, 1, 2, \ldots)$ を 2 倍 $(0, 2, 4, \ldots)$ してから，1 を足して $(1, 3, 5, \ldots)$ 出力する:

```
ramp | sopr -m 2 -a 1 | dmp
```

foat 形式のファイル *data.f1* のデータと float 形式のファイル *data.f2* のデータの平均をファイル *data.avrg* に出力する:

```
vopr -a data.f1 data.f2 | sopr -d 2 > data.avrg
```

float 形式のファイル *data.f* からデータを読み込み，dB 単位に変換した結果を出力する:

```
sopr data.f -LN -m dB | dmp
```

この例は,

```
sopr data.f -LOG10 -m 20 | dmp
```

としても結果は同じです.

メモリレジスタを使い

$$y = (1 + 3x + 4x^2)/(1 + 2x + 5x^2)$$

を計算する：

```
sopr data.f -w m0 -m 5 -a 2 -m m0 -a 1 -w m1 \
       -r m0 -m 4 -a 3 -m m0 -a 1 -d m1 | dmp
```

ここで m0, m1 はメモリレジスタで，m0 から m9 まで使えます．-w はメモリレジスタへの書き出し，-r はメモリレジスタから現在の演算レジスタへの読み込みを行います.

SEE ALSO

vopr, vsum

NAME

spec – 対数スペクトルを求める

SYNOPSIS

spec [–**l** L] [–**m** M] [–**n** N] [–**z** ***zfile***] [–**p** ***pfile***]
[–**y** Y] [***infile***]

DESCRIPTION

標準入力から入力された実数列の対数振幅スペクトルを計算します．つまり，入力数列を

$$x(0), x(1), \ldots, x(L-1)$$

として，

$$\begin{aligned} X_k &= X(e^{j\omega})\Big|_{\omega = \frac{2\pi k}{L}} \\ &= \sum_{m=0}^{L-1} x(m) e^{-j\omega m}\Big|_{\omega = \frac{2\pi\, k}{L}} \quad , \qquad k = 0, 1, \ldots, L-1 \end{aligned}$$

を FFT により計算し，–**y** オプションの指定に従って，

$$Y_k = 20 \log_{10} |X_k|, \qquad k = 0, 1, \ldots, L/2$$

などを出力します．出力されるデータは，角周波数 $0 \sim \pi$ に対応します．データ形式は入力，出力とも float 形式です．

–**p**, –**z** オプションが指定されたときには，指定されたファイルを係数としてもつディジタルフィルタの周波数特性を，同様に出力します[2]．

OPTIONS

–**l**	L	FFT の点数．2 のべき乗数を指定．	[256]
–**m**	M	伝達関数の分子多項式の次数．実際に入力されたデータ数が，$M+1$ より少なかった場合には，M = (実際の入力データ数) -1 となるので，長さ $M+1$ のデータを何組か続けて入力する場合以外は，M の指定は必要ありません．	[L-1]

[2] この際、フィルタのインパルス応答から振幅特性を求めるのではなく，分子係数，分母係数から別々に振幅特性を求め，その商によって，全体の特性を求めています．

−n N 伝達関数の分母多項式の次数．**−m** オプションと同様，実際に入力されたデータ数が，$N+1$ より少なかった場合には，$N=$(実際の入力データ数)-1 となるので，長さ $N+1$ のデータを何組か続けて入力する場合以外は，N の指定は必要ありません． [L-1]

−z $zfile$ 伝達関数の分子多項式の係数が書かれたファイル．ファイルの内容は float 形式で次のように与えます． [NULL]

$$b(0), b(1), \ldots, b(N)$$

−p $pfile$ 伝達関数の分母多項式の係数が書かれたファイル．ファイル内容は float 形式で次のように与えます． [NULL]

$$K, a(1), \ldots, a(M)$$

−y Y 出力するスペクトルのスケールを指定． [0]

$Y=0$	$20\times\log\lvert X_k\rvert$	$k=0,1,\ldots,L/2$
$Y=1$	$\ln\lvert X_k\rvert$	$k=0,1,\ldots,L/2$
$Y=2$	$\lvert X_k\rvert$	$k=0,1,\ldots,L/2$

pfile, zfile の内容は，*dfs* の場合と同じです．
−p，**−z** オプションの内，**−p** オプションだけが指定された場合，分子多項式は 1 として扱われ、**−z** オプションだけが指定された場合には，分母多項式，ゲイン K とも 1 として扱われます．**−p**，**−z** オプションいずれも省略された場合には，データは標準入力から読まれます．

EXAMPLE

パルス列で励振したディジタルフィルタの出力にブラックマン窓をかけ，その対数振幅スペクトルを表示する:

```
train -t 50 | dfs -a 1 0.9 | window | spec | fdrw | xgr
```

float 形式のファイル *data.p, data.z* で指定された係数をもつディジタルフィルタの周波数特性を表示する:

```
spec -p data.p -z data.z | fdrw | xgr
```

data.p, data.z の表すフィルタが安定な場合には，

```
impulse | dfs -p data.p -z data.z | spec | fdrw | xgr
```

としてもほぼ同様の結果が得られます．

SEE ALSO

phase, fft, fftr, dfs

NAME

srcnv – サンプリングレート変換（アップサンプリング）

SYNOPSIS

srcnv [**–s** S] [**–c** C] [**–u** U] [**–d** D] [***infile***]

DESCRIPTION

srcnv アップサンプリングを行います.

データ形式は入力，出力とも float 形式です．フィルタ係数は次のものが用いられます.

$S = 23F$	$SPTK/lib/lpfcoef.2to3f
$S = 23S$	$SPTK/lib/lpfcoef.2to3s
$S = 34$	$SPTK/lib/lpfcoef.3to4
$S = 45$	$SPTK/lib/lpfcoef.4to5
$S = 57$	$SPTK/lib/lpfcoef.5to7
$S = 58$	$SPTK/lib/lpfcoef.5to8
	($SPTK はインストールしたディレクトリ)

–u，–d オプションで，アップサンプリングとダウンサンプリングの比率を変更することができます．フィルタ係数のファイルを指定する場合は，**–u，–d** オプションも同時に指定します.

なお，ファイルタ係数のファイルは ASCII 形式となっています.

OPTIONS

–s S 変換タイプ. [58]

$S = 23F$	比率 2 : 3 でアップサンプリング
$S = 23S$	比率 2 : 3 でアップサンプリング
$S = 34$	比率 3 : 4 でアップサンプリング
$S = 45$	比率 4 : 5 でアップサンプリング
$S = 57$	比率 5 : 7 でアップサンプリング
$S = 58$	比率 5 : 8 でアップサンプリング

–c *file* ローパスフィルタ係数ファイル名. [Default]

–u U アップサンプリングの比率 [N/A]

–d D ダウンサンプリングの比率 [N/A]

EXAMPLE

float形式の標本化周波数10kHzの音声データファイル *data.10* を標本化周波数16kHzにアップサンプリングし，*data.16* に出力する:

```
srcnv -s 58 data.10 > data.16
```

NAME

step – ユニットステップ列を発生する

SYNOPSIS

step [–**l** L] [–**n** N]

DESCRIPTION

長さ L または次数 N のユニットステップを標準出力に出力します．つまり，

$$\underbrace{1, 1, 1, \ldots, 1}_{L}$$

出力データは，float 形式です．

OPTIONS

–**l** L ステップ列の長さ． $L \leq 0$ の場合，無限にステップ列を生成． [256]
–**n** N ステップ列の次数． [255]

EXAMPLE

標準形のディジタルフィルタにユニットステップ列を加える:

```
step | dfs -a 1 -0.8 | dmp
```

SEE ALSO

impulse, train, ramp, sin

NAME

swab – バイト単位でのスワップ

SYNOPSIS

swab [–**S** S_1] [–**s** S_2] [–**E** E_1] [–**e** E_2] [+*type*] [***infile***]

DESCRIPTION

swab は，リトルエンディアン (Intel,DEC 等) およびビッグエンディアン (Sun,HP 等) のバイトオーダーを変換 (バイトスワップ) し，標準出力に出力します．ファイルが指定されなかった場合，データは標準入力から読み込まれます．

–**S**, –**E** オプション，及び –**s**, –**e** オプションでスワップするデータ範囲を指定します．

入力データ，及び出力データの形式は，+*type* で指定されたものとなります．

OPTIONS

–**S**	S_1	スワップするデータの開始バイトアドレス．	[0]
–**s**	S_2	スワップするデータの開始データ番号．	[0]
–**E**	E_1	スワップするデータの終了バイトアドレス．	[EOF]
–**e**	E_2	スワップするデータの終了データ番号．	[0]
+*type*		入出力データの型指定．	[s]

s short 型 (2bytes) l long 型 (4bytes)
f float 型 (4bytes) d double 型 (8bytes)

EXAMPLE

float 形式のファイル *data.f* のバイトオーダーを変換してファイル *data.swab* に出力する:

```
swab +f data.f > data.swab
```

NAME

train – パルス列あるいはM系列を発生する

SYNOPSIS

train [–**l** L] [–**p** P]

DESCRIPTION

長さ L，周期 P のパルス列，あるいはM系列を標準出力に出力します．出力データは，float 形式です．

M系列は ± 1 の値をもちます．

OPTIONS

–**l**	L	出力データの長さ．	[256]
–**p**	P	パルス列の周期．$P = 0$ のときM系列を出力．	[0]
–**n**	N	パルス列の正規化．	[1]

パルス列を $x(n)$ としたとき

0 正規化しない．

1 $\sum_{n=0}^{L-1} x^2(n) = 1$ となるように正規化．

2 $\sum_{n=0}^{L-1} x(n) = 1$ となるように正規化．

EXAMPLE

ディジタルフィルタをM系列で励振した出力のスペクトルを見る:

```
train | dfs -b 1 0.9 | window | spec | fdrw | xgr
```

SEE ALSO

impulse, sin, step, ramp

NAME

uels – 対数スペクトルの不偏推定法 [2323]

SYNOPSIS

uels [–**m** M] [–**l** L] [–**i** I] [–**j** J] [–**d** D] [–**e** E] [***infile***]

DESCRIPTION

対数スペクトルの不偏推定法によって得られたケプストラム係数 $c(m)$ を標準出力に出力します．入力は窓掛けされた長さ L の時系列

$$x(0), x(1), \ldots, x(L-1)$$

です．

データ形式は入力，出力とも float 形式です．

従来の対数スペクトルの推定法あるいは平滑化法の主な問題点は，対数スペクトルに対する平滑化の意味が明確ではないことと推定値のバイアスが十分小さいという保証がないことでした．

対数スペクトルの不偏推定値を得るための計算手順は，改良ケプストラム法とほとんど同じですが，バイアスが生じないような非線形平滑化を行うようにしている点が根本的に異なっています．

OPTIONS

–**m**	M	分析次数.	[25]
–**l**	L	入力データのフレーム長.	[256]

通常，以下のオプションの指定は必要ありません．

–**i**	I	分析の最小反復回数.	[2]
–**j**	J	分析の最大反復回数.	[30]
–**d**	D	分析の終了条件.	[0.001]
–**e**	E	ピリオドグラムに足し込む小さな値	[0.0]

EXAMPLE

float 形式のファイル *data.f* を次数 15 次で対数スペクトルの不偏推定法で分析し，ケプストラムを *data.cep* に出力する:

```
frame < data.f | window | uels -m 15 > data.cep
```

SEE ALSO

icep, gcep, mcep, mgcep, lmadf

NAME

ulaw – μ-law 圧縮・展開

SYNOPSIS

ulaw [–**v** V] [–**u** U] [–**c**] [–**d**] [***infile***]

DESCRIPTION

ulaw によりデータを圧縮・展開します．入力を $x(n)$，出力を $y(n)$ とし，入力の最大値を V，圧縮係数を U とすると圧縮は

$$y(n) = sgn(x(n))V\frac{\log(1+U\frac{|x(n)|}{V})}{\log(1+U)}$$

となり，展開は

$$y(n) = sgn(x(n))V\frac{(1+u)^{|x(n)|/V}-1}{U}$$

となります．

OPTIONS

–**v**	V	入力の最大値.	[32768]
–**u**	U	圧縮係数.	[256]
–**c**		圧縮モード.	[TRUE]
–**d**		展開モード.	[FALSE]

EXAMPLE

16 bit のデータ *data.s* を 8 bit に ulaw 圧縮し，*data.ulaw* に出力する:

```
x2x +sf data.s | ulaw | sopr -d 256 | x2x +fc -r > data.ulaw
```

NAME

us – サンプリングレート変換（アップサンプリング）

SYNOPSIS

us [–**s** S] [–**c** ***file***] [–**u** U] [–**d** D] [***infile***]

DESCRIPTION

us アップサンプリングを行います.

データ形式は入力，出力とも float 形式です．フィルタ係数は次のものが用いられます.

$S = 23F$	$SPTK/lib/lpfcoef.2to3f
$S = 23S$	$SPTK/lib/lpfcoef.2to3s
$S = 34$	$SPTK/lib/lpfcoef.3to4
$S = 45$	$SPTK/lib/lpfcoef.4to5
$S = 57$	$SPTK/lib/lpfcoef.5to7
$S = 58$	$SPTK/lib/lpfcoef.5to8
	($SPTK はインストールしたディレクトリ)

–u，–d オプションで，アップサンプリングとダウンサンプリングの比率を変更することができます．フィルタ係数のファイルを指定する場合は，**–u，–d** オプションも同時に指定します.

なお，ファイルタ係数のファイルは ASCII 形式となっています.

OPTIONS

–s	S	変換タイプ.	[58]
		$S = 23F$ 比率 2 : 3 でアップサンプリング	
		$S = 23S$ 比率 2 : 3 でアップサンプリング	
		$S = 34$ 比率 3 : 4 でアップサンプリング	
		$S = 45$ 比率 4 : 5 でアップサンプリング	
		$S = 57$ 比率 5 : 7 でアップサンプリング	
		$S = 58$ 比率 5 : 8 でアップサンプリング	
–c	*file*	ローパスフィルタ係数ファイル名.	[Default]
–u	U	アップサンプリングの比率	[N/A]
–d	D	ダウンサンプリングの比率	[N/A]

EXAMPLE

float 形式の標本化周波数 16kHz の音声データファイル *data.16* を標本化周波数 44.1kHz にアップサンプリングし，*data.44* に出力する:

```
us -s 23F data.16 | us -s 23S | us -s 57 | \
us -c /usr/local/SPTK/lib/lpfcoef.5to7 -u 7 -d 8 > data.44
```

注: $\frac{44100}{16000} = \frac{3 \times 3 \times 7 \times 7 \times 100}{2 \times 2 \times 5 \times 8 \times 100}$

NAME

us16 – サンプリングレート変換（16kHz へのアップサンプリング）

SYNOPSIS

us16 [–**s** *S*] [+***type***] [***infile***] [***outfile***]

us16 [–**s** *S*] [+***type***] ***infile1*** ··· [***infileN***] ***outdir***

DESCRIPTION

us16 は，10，12kHz から 16kHz へのアップサンプリングを行います．入力ファイル，出力ファイルの指定が無い場合は，それぞれ標準入力から読み込み，標準出力へ書き込まれます．また，複数のファイルを同時に指定することも可能です．この場合，入力ファイル名のサフィックスが変換後の標本化周波数に置き換えられ，*outdir* 以下に出力されます．

OPTIONS

–**s**	*S*	入力データの標本化周波数.（10，12 kHz）	[10]
+*t*		入力データの形式.	[s]
		s short 型 (2bytes)	
		f float 型 (4bytes)	

EXAMPLE

short 形式の標本化周波数 10kHz の音声データファイル *data.10* を標本化周波数 16kHz にアップサンプリングし，*data.16* に出力する:

```
us16 -s 10 < data.10 > data.16
```

NAME

uscd – サンプリングレート変換（44.1kHz 系へのアップサンプリング）

SYNOPSIS

uscd [–**s** *S*] [–**S** *S*] [+***type***] [***infile***] [***outfile***]

uscd [–**s** *S*] [–**S** *S*] [+***type***] ***infile1*** ··· [***infileN***] ***outdir***

DESCRIPTION

*uscd*は，8，10，12，16kHz から 11.025，22.05，44.1kHz へのアップサンプリングを行います．入力ファイル，出力ファイルの指定が無い場合は，それぞれ標準入力から読み込み，標準出力へ書き込まれます．また，複数のファイルを同時に指定することも可能です．この場合，入力ファイル名のサフィックスが変換後の標本化周波数に置き換えられ，*outdir* 以下に出力されます．

なお，オプションで指定する 11.025，22.05，44.1kHz は，それぞれ 11，22，44 のように省略可能です．

OPTIONS

–**s**	*S*	入力データの標本化周波数.（8，10，11，12 kHz）	[10]
–**S**	*S*	出力データの標本化周波数.（11.025，22.05，44.1 kHz） 但し，11.025，22.05，44.1kHz は，それぞれ 11，22，44 に省略可能.	[11.025]
+*t*		入力データの形式.	[s]
		s short 型 (2bytes)	
		f float 型 (4bytes)	

EXAMPLE

short 形式の標本化周波数 16kHz の音声データファイル *data.16* を標本化周波数 22.05kHz にアップサンプリングし，*data.22* に出力する:

```
uscd -s 16 -S 22.05 < data.16 > data.22
```

NAME

vopr – ベクトル演算を行う

SYNOPSIS

vopr [–**l** L] [–**n** N] [–**i**] [–**a**] [–**s**] [–**m**] [–**d**]
[–**ATAN2**][***file1***] [***infile***]

DESCRIPTION

入力データに対しベクトル演算を行います．つまり，

file1 被演算ベクトルファイル (省略時は stdin)

infile 演算ベクトルファイル (省略時は stdin)

として演算ベクトル **a**、被演算ベクトル **b** を読み込み，オプションで指定された演算を行い結果を標準出力に出力します．

データ形式は入力，出力とも float 形式です．

指定ファイル数とベクトルの長さの組み合わせにより動作が異なります．

ファイルを 2 個指定した場合（*file1* として stdin を想定した場合，つまり–**i** オプションなしで *infile* だけ指定した場合も含みます）には，ベクトルの長さの値により次のような動作をします．

$L = 1$ のとき

file1 (stdin)	a_1	a_2	...	a_i	...
infile	b_1	b_2	...	b_i	...
Output (stdout)	y_1	y_2	...	y_i	...

各ファイルのデータが 1 対 1 に対応します．

$L \geq 2$ のとき

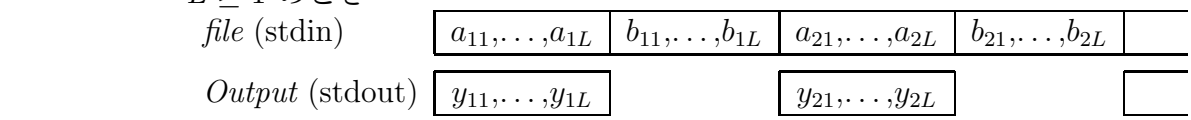
file1 (stdin)	$a_{11},\ldots,a_{1L}$	$a_{21},\ldots,a_{2L}$	$a_{31},\ldots,a_{3L}$	$a_{41},\ldots$
infile	$b_1,\ldots,b_L$			
Output (stdout)	$y_{11},\ldots,y_{1L}$	$y_{21},\ldots,y_{2L}$	$y_{31},\ldots,y_{3L}$	$y_{41},\ldots$

演算ベクトルは *infile* から 1 回だけ読み込まれ，繰り返し被演算ベクトルに対して作用します．

ファイルを 1 個だけ指定した場合（つまり，ファイル名をひとつだけ記述し，–**i** オプションにより *infile=file1* と指定した場合，またファイルをひとつも記述せず

file1=infile=stdin と指定した場合）には，ベクトルの長さによらず次のような動作をします.

$L \geq 1$ のとき



演算ベクトル，被演算ベクトルとも同じファイルから読み込まれます.

OPTIONS

–l	L	ベクトルの長さ.	[1]
–n	N	ベクトルの次数.	[L-1]
–i		ファイルを 1 個しか指定しなかった場合，そのファイルを被演算ベクトルと演算ベクトルの両方を含んだファイルとみなす.	[FALSE]
–a		加算. $y_i = a_i + b_i$.	[FALSE]
–s		減算. $y_i = a_i - b_i$.	[FALSE]
–m		乗算. $y_i = a_i * b_i$.	[FALSE]
–d		除算. $y_i = a_i / b_i$.	[FALSE]
–ATAN2		逆正接. $y_i = \mathrm{atan}\,2(b_i, a_i)$.	[FALSE]

EXAMPLE

float 形式のファイル *data.a* とファイル *data.b* の和をファイル *data.c* に出力する:

```
vopr -a data.a data.b > data.c
```

ファイル *data.w* に入っている適当な窓 (窓長 256) を用いて，正弦波に窓かけを行う.

```
sin -t 30 -l 1000 | vopr data.w -l 256 -m | fdrw | xgr
```

これは，ファイル *data.w* にブラックマン窓が入っていたとすれば,

```
sin -t 30 -l 1000 | window | fdrw | xgr
```

に等価です.

SEE ALSO

sopr, vsum

NAME

vq – ベクトル量子化

SYNOPSIS

vq [–**l** L] [–**n** N] [–**q**] ***cbfile*** [***infile***]

DESCRIPTION

指定されたファイルから長さ L の時系列

$$x(0), x(1), \ldots, x(L-1)$$

を読み込み，コードブックファイル *cbfile* 中の各コードベクトルとのユークリッド距離 d_i

$$d_i = \frac{1}{L} \sum_{m=0}^{L-1} (x(m) - c_i(m))^2$$

が最小になるインデックス i を出力します．-q オプションが指定された場合には，コードベクトル $[c_i(0), c_i(1), \cdots, c_i(L-1)]$ を出力します．

データ形式は入力は float 形式，出力は int 形式です．ただし, -q オプションが指定された場合には, 出力は float 形式になります．

OPTIONS

–**l**	L	ベクトルのサイズ.	[26]
–**n**	N	ベクトルの次数.	[25]
–**q**		量子化されたコードベクトルを出力.	[FALSE]

EXAMPLE

float 形式の 25 次のベクトル列のファイル *data.f* に対し，コードブック *cbfile* で VQ を行い *data.vq* に出力する:

```
vq -q cbfile < data.f > data.vq
```

SEE ALSO

ivq, msvq, imsvq, lbg

NAME

vstat – ベクトルの平均，共分散を求める

SYNOPSIS

vstat [–**l** L] [–**n** N] [–**t** T] [–**d**] [–**o** O] [***infile***]

DESCRIPTION

指定されたファイルから読み込まれた L 次元ベクトルの平均ベクトルおよび共分散行列を T 個毎に求め，標準出力に出力します．つまり，入力データ

$$\overbrace{\overbrace{x_1(1),\ldots,x_1(L)}^{L},\overbrace{x_2(1),\ldots,x_2(L)}^{L},\ldots,\overbrace{x_N(1),\ldots,x_N(L)}^{L},\ldots}^{T\times L}$$

から，

$$\overbrace{m(1),\ldots,m(L)}^{L},\overbrace{\overbrace{U(11),\ldots,U(1L)}^{L},\ldots\overbrace{U(L1),\ldots,U(LL)}^{L},\ldots}^{L\times L}$$

が出力されます．$\boldsymbol{m}$，$\boldsymbol{U}$ はそれぞれ，

$$\boldsymbol{m} = \frac{1}{N}\sum_{k=1}^{N}\boldsymbol{x}$$

$$\boldsymbol{U} = \frac{1}{N}\sum_{k=1}^{N}\boldsymbol{x}\boldsymbol{x}^T - \boldsymbol{m}\boldsymbol{m}^T$$

で計算されます．ただし，対角共分散の場合は非対角成分は出力されません．

ファイルが指定されなかった場合，データは標準入力から読み込まれます．

データ形式は入力、出力とも float 形式です。

OPTIONS

–**l**	L	ベクトルの長さ.	[1]
–**n**	N	ベクトルの次数.	[L-1]
–**t**	T	和をとるベクトルの個数.	[N/A]
–**d**		対角共分散のみ出力	[FALSE]

–o *O* 出力のフォーマット. [0]

$O = 0$ 平均ベクトルおよび共分散行列
$O = 1$ 平均ベクトルのみ
$O = 2$ 共分散行列のみ

–d 分散の代わりに相関を出力する [FALSE]

EXAMPLE

float 形式のファイル *data.f* 全体の平均および分散を求め，*data.stat* に出力する:

```
vstat data.f > data.stat
```

次元 15 次の係数ベクトルのとなり合う 3 フレームの平均を求め *data.av* に出力する:

```
vstat -l 15 -t 3 -o 1 data.f > data.av
```

SEE ALSO

average, vsum

NAME

vsum – ベクトルの総和をとる

SYNOPSIS

vsum [–**l** L] [–**n** N] [***infile***]

DESCRIPTION

指定されたファイルから読み込まれた L 次元ベクトルの各要素毎に総和をとり，標準出力に出力します．つまり，入力データ

$$\overbrace{\overbrace{a_1(1),\ldots,a_1(L)}^{L},\overbrace{a_2(1),\ldots,a_2(L)}^{L},\ldots,\overbrace{a_N(1),\ldots,a_N(L)}^{L},\ldots}^{N\cdot L}$$

から，

$$\overbrace{s(1),\ldots,s(L)}^{L},\ldots$$

が出力され，$s(n)$ は，

$$s(n)=\sum_{k=1}^{N}a_k(n)$$

で計算されます．

ファイルが指定されなかった場合，データは標準入力から読み込まれます．

データ形式は入力、出力とも float 形式です。

OPTIONS

–**l**	L	ベクトルの次元.	[1]
–**n**	N	和をとるベクトルの個数.	[EOD]

EXAMPLE

float 形式のファイル *data.f* 全体の和を求め *data.sum* に出力する:

```
vsum data.f > data.sum
```

次元 10 次のベクトルのノルムを求め *data.n* に出力する:

```
sopr data.f -P | vsum -n 10 | sopr -R > data.n
```

次元 15 次の係数ベクトルのとなり合う 3 フレームの平均を求め *data.av* に出力する:

```
vsum -l 15 -n 3 data.f | sopr -d 3 > data.av
```

SEE ALSO

sopr

NAME

window – 窓をかける

SYNOPSIS

window [–**l** L_1] [–**L** L_2] [–**n** N] [–**w** W] [***infile***]

DESCRIPTION

標準入力から入力される実数列にオプションで指定された窓をかけます．つまり，入力数列を

$$x(0), x(1), \ldots, x(L_1 - 1)$$

窓を

$$w(0), w(1), \ldots, w(L_1 - 1)$$

として，

$$x(0) \cdot w(0),\ x(1) \cdot w(1),\ \ldots,\ x(L_1 - 1) \cdot w(L_1 - 1)$$

が出力されます．L_2 が L_1 より大きい場合は 0 が付加され，

$$\underbrace{x(0) \cdot w(0),\ x(1) \cdot w(1),\ \ldots,\ x(L_1 - 1) \cdot w(L_1 - 1), 0, \ldots, 0}_{L_2}$$

が出力されます．

データ形式は入力，出力とも float 形式です．

OPTIONS

–**l** L_1 窓の長さ．$(L \leq 2048)$ [256]

–**L** L_2 出力の長さ． [L_1]

–**n** N 正規化． [1]

0 正規化しない．

1 窓の振幅を $\sum_{n=0}^{L-1} w^2(n) = 1$ となるように正規化．

2 窓の振幅を $\sum_{n=0}^{L-1} w(n) = 1$ となるように正規化．

−**w** *W* 窓の形. [0]

0 blackman
1 hamming
2 hanning
3 bartlett
4 rectangular

EXAMPLE

周期 20 の正弦波に，ブラックマン窓をかけた波形を表示する:

```
sin -p 20 | window | fdrw | xgr
```

パルス列で励振したディジタルフィルタの出力に窓長 50 点のブラックマン窓をかけ，その対数振幅スペクトルを 512 点の FFT によって計算し表示する:

```
train -p 50 | dfs -a 1 0.9 | window -l 50 -L 512 |\
spec -l 512 | fdrw | xgr
```

SEE ALSO

fftr, spec

NAME

x2x – データ形式の変換

SYNOPSIS

x2x [+*type*1] [+*type*2] [%*format*] [+*a* **A**] [−**r**]

DESCRIPTION

x2x は，標準入力から読み込まれたデータを指定されたデータ形式に変換し，標準出力に出力します.

入力データの形式は，+*type*1 オプションで，出力データの形式は，+*type*2 オプションで，指定します.

OPTIONS

+*type*1 入力データの形式. [f]

+*type*2 出力データの形式. [type1]

オプション *type*1, *type*2 ともに，以下のいずれかで指定します.

c	char 型 (1byte)	C	unsigned char 型 (1byte)
s	short 型 (2bytes)	S	unsigned short 型 (2bytes)
i	int 型 (4bytes)	I	unsigned int 型 (4bytes)
l	long 型 (4bytes)	L	unsigned long 型 (4bytes)
f	float 型 (4bytes)	d	double 型 (8bytes)
a	ascii 文字列		

データを t_1 型から t_2 型へ型変換して出力する． t_2 が省略された場合は $t_2 = t_1$ とみなし型変換は行なわない.

+**a** *A* カラム数．データを *A* 個出力する毎に改行する． [1]

−**r** 浮動小数点データ（float, double, ascii 文字型）を整数データ（char, short, int, long 型）に変換する際，小数点以下を四捨五入. [FALSE]

%*format* *type*2 が ascii である時，C 言語の printf() 関数のような出力フォーマットで出力. [%*g*]

EXAMPLE

ascii 形式でかかれたデータファイル *data.asc* を float 形式のファイル *data.f* に変換する:

```
x2x +af < data.asc > data.f
```

float 形式のファイル *data.f* にあるデータを ascii 形式に変換し，画面に出力する:

```
x2x +fa < data.f
```

例えば，ファイル *data.f* に，float 形式の数値データ

$1, 2, 3, 4, 5, 6, 7$

が入っていた場合,

```
1
2
3
4
5
6
7
```

と画面（標準出力）に出力されます.

同じデータをカラム数を指定して出力する:

```
x2x +fa 3 < data.f
```

この場合,

```
1       2       3
4       5       6
7
```

と表示されます.

出力を printf の%e フォーマットで行う:

```
x2x +fa %9.4e < data.f
```

この例では，幅 11 桁 ，小数点以下 4 桁を指定しており,

```
1.0000e+000
2.0000e+000
    ⋮
7.0000e+000
```

と表示されます.

SEE ALSO

dmp

NAME

xgr – プロッタコマンド列を X-Window 上に表示

SYNOPSIS

xgr [–**s** ***S***] [–**l**] [–**rv**] [–**m**] [–**bg** ***BG***] [–**hl** ***HL***] [–**bd** ***BD***]
[–**ms** ***MS***] [–**g** ***G***] [–**d** ***D***] [–**t** ***T***] [***infile***]

DESCRIPTION

標準入力から プロッタコマンド列を読み込み，グラフ出力を X-Window 上に表示します.

- ウィンドウマネージャの機能によりウィンドウサイズは変わりますが，仮想画面上に書かれているグラフ図形の表示はズーミングされません.
- ウィンドウサイズが仮想画面よりも小さい場合には，仮想画面をスクロールすることができます.（vi 風キーバインド）

  h: 左スクロール
  j: 下スクロール
  k: 上スクロール
  l: 右スクロール

- 終了は，”q”，”Ctrl-c”，”Ctrl-d” のいずれかを入力して下さい.

OPTIONS

–**s**	*S*	ウィンドウ縮小比率.	[3.38667]
–**l**		ランドスケープモード.	[FALSE]
–**rv**		リバースモード.	[FALSE]
–**m**		白黒ディスプレイモード.	[FALSE]
–**bg**	*BG*	バックグラウンドカラー.	[white]
–**hl**	*HL*	ハイライトカラー.	[blue]
–**bd**	*BD*	ボーダーカラー.	[blue]
–**ms**	*MS*	マウスカラー.	[red]
–**g**	*G*	ウィンドウのサイズ及び表示位置.	[NULL]
–**d**	*D*	ディスプレイサーバ.	[NULL]
–**t**	*T*	ウィンドウのタイトル.	[xgr]

EXAMPLE

fdrw を用いてファイル *data.f* に含まれるデータのグラフを描き，X-window 上に出力:

```
fdrw < data.f | xgr
```

BUGS

- ディスプレイサーバが，バッキングストア機能を持っていない場合には隠れ面が露出した際に画面の再描画が行なわれません．
- 表示の待ち時間を減らすため，まず生成したウィンドウ上に図形を描き，仮想画面にコピーしてイメージを保持して再描画等を行ないますので，”-g”オプションでウィンドウサイズを小さくし過ぎた場合や描画中にウィンドウに隠れ面ができると表示できない部分が生じます．最初からウィンドウサイズを小さくしたい場合には，”-s”オプションを併用して描画サイズも小さくしておくことを薦めます．

SEE ALSO

fig, fdrw

NAME

zcross – 零交差数を求める

SYNOPSIS

zcross [–**l** *L*] [–**n**] [***infile***]

DESCRIPTION

標準入力から入力される時系列から零交差数を求めます.

データ形式は入力，出力とも float 形式です.

OPTIONS

–**l**	*L*	フレーム長（$L \leq 0$ の時は何も出力しない）.	[256]
–**n**		零交差数をフレーム長で正規化.	[FALSE]

EXAMPLE

float 形式のデータ *data.f* の零交差数を求め， *data.zc* に出力する:

```
zcross < data.f > data.zc
```

SEE ALSO

frame, spec

NAME

zerodf – 音声合成のための全零フィルタ

SYNOPSIS

zerodf [**–m** M] [**–p** P] [**–i** I] [**–t**] [**–k**] ***bfile*** [***infile***]

DESCRIPTION

指定されたファイルから読み込まれたデータを *bfile* の係数，$b(0), b(1), \ldots, b(M)$ をもつ全零標準形フィルタによりフィルタリングし，標準出力に出力します.

データ形式は，入力，出力とも float 形式です.

全零標準形フィルタの伝達関数 $H(z)$ は,

$$H(z) = \sum_{m=0}^{M} b(m) z^{-m}$$

と表されます.

OPTIONS

–m	M	フィルタの次数.	[25]
–p	P	フレーム周期.	[100]
–i	I	補間周期.	[1]
–t		転置型フィルタ.	[FALSE]
–k		ゲインを除いたシステム関数でフィルタリングする	[FALSE]

EXAMPLE

float 形式のピッチデータ *data.pitch* から励振源を作成し，零型フィルタの係数ファイル *data.b* により全零フィルタを駆動し，合成音声を *data.syn* に出力する:

```
excite < data.pitch  | zerodf data.b > data.syn
```

SEE ALSO

poledf, lmadf

参考文献

[1] 今井 聖, 阿部 芳春, “改良ケプストラム法によるスペクトル包絡の抽出,” 信学論 (A), Vol.J62-A, No.4, pp.217–223, Apr. 1979.

[2] 今井 聖, 古市 千枝子, “対数スペクトルの不偏推定,” 信学論 (A), Vol.J70-A, No.3, pp.471–480, Mar. 1987.

[3] S. Imai and C. Furuichi, “Unbiased estimator of log spectrum and its application to speech signal processing,” Signal Processing IV: Theory and Applications, Vol.1, pp.203–206, Elsevire, North-Holland, 1988.

[4] 徳田 恵一, 小林 隆夫, 塩本 祥司, 今井 聖, “適応ケプストラム分析 — ケプストラムを係数とする適応フィルタ —,” 信学論 (A), Vol.J73-A, No.7, pp.1207–1215, July 1990.

[5] K. Tokuda, T. Kobayashi and S. Imai, “Adaptive cepstral analysis of speech,” IEEE Trans. Speech and Audio Process., Vol.3, No.6, pp.481–488, Nov. 1995.

[6] 徳田 恵一, 小林 隆夫, 山本 竜太郎, 今井 聖, “一般化ケプストラムをパラメータとする音声のスペクトル推定,” 信学論 (A), Vol.J72-A, No.3, pp.457–465, Mar. 1989.

[7] T. Kobayashi and S. Imai, “Spectral analysis using generalized cepstrum,” IEEE Trans. Acoust., Speech, Signal Process., Vol.ASSP-32, No.5, pp.1087–1089, Oct. 1984.

[8] K. Tokuda, T. Kobayashi and S. Imai, “Generalized cepstral analysis of speech — a unified approach to LPC and cepstral method,” Proc. ICSLP-90, pp.37–40, Nov. 1990.

[9] 深田 俊明, 徳田 恵一, 小林 隆夫, 今井 聖, “適応一般化ケプストラム分析の検討,” 1990 信学春季全大, A-150, pp.1-150, Mar. 1990.

[10] 徳田 恵一, 小林 隆夫, 深田 俊明, 斎藤 博徳, 今井 聖, “メルケプストラムをパラメータとする音声のスペクトル推定,” 信学論 (A), Vol.J74-A, No.8, pp.1240–1248, Aug. 1991.

[11] 徳田 恵一, 小林 隆夫, 深田 俊明, 今井 聖, “音声の適応メルケプストラム分析,” 信学論 (A), Vol.J74-A, No.8, pp.1249–1256, Aug. 1991.

[12] T. Fukada, K. Tokuda, T. Kobayashi and S. Imai, “An adaptive algorithm for mel-cepstral analysis of speech,” Proc. ICASSP-92, pp.137–140, Mar. 1992.

[13] 徳田 恵一, 小林 隆夫, 千葉 建司, 今井 聖, “メル一般化ケプストラム分析による音声のスペクトル推定,” 信学論 (A), Vol.J75-A, No.7, pp.1124–1134, July 1992.

[14] K. Tokuda, T. Kobayashi, T. Masuko and S. Imai, “Mel-generalized cepstral analysis — a unified approach to speech spectral estimation,” Proc. ICSLP-94, pp.1043–1046, Sep. 1994.

[15] 若子 武士, 徳田 恵一, 益子 貴史, 小林 隆夫, 北村 正, “対数スペクトルの任意基底関数による展開に基づく音声のスペクトル推定,” 信学論 (A), Vol.J82-D-II, No.12, pp.2203–2211, Dec. 1999.

[16] 今井 聖, “対数振幅近似 (LMA) フィルタ,” 信学論 (A), vol.J63-A, no.12, pp.886–893, Dec. 1980.

[17] 千葉 健司, 徳田 恵一, 小林 隆夫, 今井 聖, “一般化ケプストラムをパラメータとする音声合成,” 昭 63 電子情報通信学会春季全国大会, A-34, pp.1-34, Mar. 1988.

[18] 今井 聖, 住田 一男, 古市 千枝子, “音声合成のためのメル対数スペクトル近似 (MLSA) フィルタ,” 信学論 (A), vol.J66-A, no.2, pp.122–129, Feb. 1983.

[19] 小林 隆夫, 今井 聖, 福田 豊, “メル一般化対数スペクトル近似 (MGLSA) フィルタ,” 信学論 (A), Vol.J68-A, No.6, pp.610–611, June 1985.

[20] 徳田 恵一, 益子 貴史, 小林 隆夫, 今井 聖, “動的特徴を用いた HMM からの音声パラメータ生成アルゴリズム,” 日本音響学会誌, vol.53, no.3, pp.192–200, Mar. 1997.

[21] K. Tokuda, T. Masuko, T. Yamada, T. Kobayashi and S. Imai, “An algorithm for speech parameter generation from continuous mixture HMMs with dynamic features,” Proc. EUROSPEECH-95, pp.757–760, Sep. 1995.