stftb 2.0

- Modification
    - scale — Scale a signal using the Mellin transform
    - sigmerge — Add two signals with given energy ratio in dB
- Noise Realizations
    - noisecg — Analytic complex gaussian noise
    - noisecu — Analytic complex uniform white noise
- Other
    - contwtgn — computes a continuous wavelet transform
    - contwtgnmir — Continuous wavelet transform of mirrored 1-D signals
    - correlmx — correlation matrix of a signal
    - d2statio — Distance to stationarity
    - disprog — Display progression of a loop
    - divider — Find dividers of an integer
    - dwindow — Derive a window
    - fzero — solves the scalar nonlinear equation such that F(X) == 0
    - gaussn — generates the order n derivative of the gaussian window, centered at frequency f0
    - imextrac — imextrac(Image) extract and isolate dots in a binary image
    - integ — Approximate 1D integral of a discrete signal
    - integ2d — Approximate 2-D integral
    - istfr1 — returns true is method is a time frequency representation of type 1 (frequencies >0 or <0)
    - istfr2 — returns true is method is a time frequency representation of type 2 (only frequencies > 0)
    - izak — Inverse Zak transform
    - kaytth — Kay-Tretter filter computation
    - odd — Round towards nearest odd value
    - rem — Return the remainder of the division x/y
    - rot90 — rotates the given 2-D array by 90 degrees
    - umaxbert — Determination of the maximum value of u for Bertrand distribution
    - umaxdfla — Determination of the maximum value of u for D-Flandrin distribution
    - umaxunte — Determination of the maximum value of u for Unterberger distribution
    - vecmodulo — Congruence of a vector
    - zak — Zak transform
- Post-Processing or Help to the Interpretation
    - friedman — FRIEDMAN Instantaneous frequency density
    - holder — Estimate the Holder exponent through an affine TFR
    - htl — Hough transform for detection of lines in images
    - margtfr — Marginals and energy of a time-frequency representation
    - midpoint — Mid-point construction used in the interference diagram
    - momftfr — Frequency moments of a time-frequency representation
    - momttfr — Time moments of a time-frequency representation
    - plotsid — Schematic interference diagram of FM signals
    - renyi — Measure Renyi information
    - ridges — Extraction of ridges
    - tfrideal — Ideal TFR for given instantaneous frequency laws
- Reassigned Time-Frequency Processing
    - tfrrgab — Reassigned Gabor spectrogram time-frequency distribution
    - tfrrmsc — Reassigned Morlet Scalogram time-frequency distribution

    - tfrrpmh — Reassigned pseudo Margenau-Hill time-frequency distribution
    - tfrrppag — Reassigned pseudo Page time-frequency distribution
    - tfrrpwv — Reassigned pseudo Wigner-Ville distribution
    - tfrrsp — Reassigned Spectrogram
    - tfrrspwv — Reassigned smoothed pseudo Wigner-Ville distribution
    - tfrrstan — Reassigned Stankovic distribution
- Choice of Particular Signals
    - altes — Altes signal in time domain
    - anaask — Amplitude Shift Keying (ASK) signal
    - anabpsk — Binary Phase Shift Keying (BPSK) signal
    - anafsk — Frequency Shift Keying (FSK) signal
    - anapulse — Analytic projection of unit amplitude impulse signal
    - anaqpsk — Quaternary Phase Shift Keying (QPSK) signal
    - anasing — Lipschitz singularity
    - anastep — Analytic projection of unit step signal
    - atoms — Linear combination of elementary Gaussian atoms
    - chirp — Evaluate a chirp signal at time t. A chirp signal is a frequency swept cosine wave
    - dopnoise — Complex noisy doppler signal
    - doppler — Generate complex Doppler signal
    - griffitc — Test signal example C of Griffiths' paper
    - klauder — Klauder wavelet in time domain
    - mexhat — Mexican hat wavelet in time domain
    - pulstran — pulse generation
    - sawtooth — Generates a sawtooth wave
    - tftb_window — Window generation
- Time-Domain Processing
    - ifestar2 — Instantaneous frequency estimation using AR2 modelisation
    - instfreq — Instantaneous frequency estimation
    - loctime — Time localization caracteristics
- Visualization and backup
    - plotifl — Plot normalized instantaneous frequency laws
    - tfrparam — parametric time-frequency representation
    - tfrqview — Quick visualization of time-frequency representations
    - tfrsave — Save the parameters of a time-frequency representation
    - tfrview — Visualization of time-frequency representations

Ambiguity plane function (in C)

- Caf2tfr — From ambiguity plane to time frequency plane
- Cambifunb — Ambiguity function
- Ctfrker — Time frequency representation kernel

Caf2tfr

From ambiguity plane to time frequency plane

Calling Sequence

```
TFR = Caf2tfr(AF, kernel)
```

Parameters

AF :

Square complex matrix, the ambiguity function of the signal

kernel :

Square real matrix with same sizes as AF, the kernel of the TFR

TFR :

Square real matrix, the time frequency response for the given kernel

Description

Computes the Cohen's group Time Frequency Representation (TFR) of a signal given its ambiguity function and the kernel of the TFR. Warning : AF and kernel are matrices of the same size and square!

Examples

```
//Wigner TFR of a tone embedded in noise
AF = Cambifunb(hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal')),-63:64,128);
ker = Ctfrker(128,128,'WV');
Wigner_TFR = Caf2tfr(AF,ker);
grayplot(1:128,linspace(0,0.5,128),Wigner_TFR');
```

See also

- Cambifunb — Ambiguity function
- Ctfrker — Time frequency representation kernel

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000

Cambifunb

Ambiguity function

Calling Sequence

```
[NAF,TAU,XI] = Cambifunb(X)
[NAF,TAU,XI] = Cambifunb(X, TAU)
[NAF,TAU,XI] = Cambifunb(X, TAU, N)
```

Parameters

X :

Real or complex signal of length Nx for which the AF should be computed

TAU :

Vector of lag values. It must contains regularily spaced integer values in [-Nx/2,Nx/2]. The default value is : -Nx/2:Nx/2.

N :

Number of frequency bins. It must be in the interval [1, Nx]. The default value is Nx.

NAF :

doppler-lag representation, with the doppler bins stored in the rows and the time-lags stored in the columns.

XI :

vector of doppler values.

Note:

The cross ambiguity function is not implemented yet

Examples

```
[AF,TAU,XI] = Cambifunb(hilbert(sin(2*%pi*0.25*
(1:128)))+0.5*rand(1,128,'normal'),-63:64,128);
grayplot(XI,TAU,abs(AF)')
xlabel 'Time lag'
ylabel 'Frequency lag'
```

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000

Ctfrker

Time frequency representation kernel

Calling Sequence

```
kernel = Ctfrker(N_Doppler, N_Delay, kernel_type)
kernel = Ctfrker(N_Doppler, N_Delay, kernel_type,
optional_parameters)
```

Parameters

N_doppler :

Number of doppler bins, number of rows in the kernel matrix

N_delay :

Number of delay bins, number of columns in the kernel matrix

kernel :

Computed kernel

kernel_type :

one of the following types:

MTEK :

Multiform tiltable exponential kernel 7 parameters required : [alpha beta gamma r tau_0 nu_0 lambda]

RGK :

Radially gaussian kernel : an odd number of parameters required, the Fourier descriptors of the contour function [c a1 ... ap b1 ... bp]

GMCWK :

Generalized marginals Choi-Williams kernel at least two parameters required : the kernel width and the angles of the branches in [0 pi]. [sigma angle_1 .. angle_2]

WV :

Wigner-Ville kernel : no parameters required

SPECTRO :

Spectrogram kernel The parameters are the window with EVEN length

Description

/ Computes the Cohen's group Time Frequency Representation's (TFR) kernel, in the ambiguity plane.

note: more kernels will be implemented in future versions

Examples

```
AF = Cambifunb(hilbert(sin(2*%pi*0.25*
(1:128))),-63:64,128);
ker = Ctfrker(128,128,'rgk',[1 1 1]);
TFR = Caf2tfr(AF,ker);
grayplot(1:128,linspace(0,0.5,128),TFR');
```

Bibliography

H. Costa and G.F. Boudreaux-Bartels, Design of Time-Frequency Representations Using a Multiform, Tiltable Exponential Kernel IEEE Trans. on Signal Processing, October 1995, vol. 43, no 10, pp 2283-2301

X.-G. Xia and Y. Owechko and B. H. Soffer and R. M. Matic, Generalized-Marginal Time-Frequency Distributions, TFTS 1996, pp. 509-51

See also

- Cambifunb — Ambiguity function
- Caf2tfr — From ambiguity plane to time frequency plane

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000

Miscellanaeous functions (C toolbox)

- Chtl — Hough transform for detection of lines in images
- Ctfrdist — Time frequency distance
- Ctfrreas — Time Frequency Representation reassignment
- Cwindow — Creates a window of length N with a given shape.

Chtl

Hough transform for detection of lines in images

Calling Sequence

```
[HT,RHO,THETA] = Chtl(IM)
[HT,RHO,THETA] = Chtl(IM, M)
[HT,RHO,THETA] = Chtl(IM, M, N)
```

Parameters

IM :

image to be analyzed (size Xmax x Ymax).

M :

desired number of samples along the radial axis (default : Xmax).

N :

desired number of samples along the azimutal (angle) axis (default : Ymax).

HT :

output matrix (MxN matrix).

RHO :

sequence of samples along the radial axis.

THETA :

sequence of samples along the azimutal axis.

Description

From an image IM, computes the integration of the values of the image over all the lines. The lines are parametrized using polar coordinates. The origin of the coordinates is fixed at the center of the image, and theta is the angle between the VERTICAL axis and the perpendicular (to the line) passing through the origin.

Examples

```
x = hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal'));
tfr = Ctfrwv(x);
[HT,RHO,THETA] = Chtl(tfr);
grayplot(THETA, RHO, HT');
```

```
xlabel theta
ylabel rho
```

See also

- Ctfrsp — Spectrogram Time Frequency distribution of a signal X

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000
- Manuel Davy - 2000

Ctfrdist

Time frequency distance

Calling Sequence

```
distance = Ctfrdist(TFR1, TFR2, dist_name)
distance = Ctfrdist(TFR1, TFR2, dist_name, dist_coef)
```

Parameters

TFR1 :

first TFR

TFR2 :

second TFR, the same size as previous one

dist_name :

identifier of the distance to compute

dist_coef :

optional parameter necessary for some distance measures

dist_name is one of the following :

'Lq'	Lq distance, 'dist_coef > 0' is the parameter q
'Quadratic'	Quadratic distance, no 'dist_coef' required
'Correlation'	Correlation distance, no 'dist_coef' required
'Kolmogorov'	Kolmogorov distance, no 'dist_coef' required
'Kullback'	Kullback distance, no 'dist_coef' required
'Chernoff'	Chernoff distance, one 'dist_coef in [0;1]' required
'Matusita'	Matusita distance, one 'dist_coef >= 1' required
'NLq'	Normalized Lq distance,'dist_coef > 0' is q
'LSD'	Log Spectral deviation, one 'dist_coef > 0' required
'Jensen'	Jensen distance, one 'dist_coef > 0' required

Description

Computes distances between Time Frequency Representations (TFRs)

Examples

```
AF1 = Cambifunb(hilbert(sin(2*%pi*0.25*(1:128))) +
0.5*rand(1,128,'normal'),-63:64,128);
AF2 = Cambifunb(hilbert(sin(2*%pi*0.15*(1:128))) +
0.5*rand(1,128,'normal'),-63:64,128);
ker = Ctfrker(128,128,'WV');
Wigner_TFR1 = Caf2tfr(AF1,ker);
Wigner_TFR2 = Caf2tfr(AF2,ker);
dist_kolm = Ctfrdist(Wigner_TFR1, Wigner_TFR2,
'Kolmogorov');
```

See also

- Cambifunb — Ambiguity function
- Ctfrker — Time frequency representation kernel
- Caf2tfr — From ambiguity plane to time frequency plane

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000

Ctfrreas

Time Frequency Representation reassignment

Calling Sequence

```
TFR_reas = Ctfrreas(TFR, field_time, field_freq)
```

Parameters

TFR :

Real matrix, TFR to be reassigned

field_time :

Real Complex matrix with same size as TFR. Time components of the field of reassignment vectors

field_freq :

Real matrix with same size as TFR. Frequency components of the field of reassignment vectors

TFR_reas :

Reassigned TFR

Description

Reassigns the pixels of a TFR, given a field of reassignment vectors.

Examples

```
[SP_reas,SP,field] = Ctfrrsp(hilbert(sin(2*%pi*0.25*
(1:128)))+0.5*rand(1,128,'normal'));
SP_reas2 = Ctfrreas(SP,real(field),imag(field));
grayplot(1:128,1:128,SP_reas');
scf(); grayplot(1:128,1:128,SP_reas2');
```

See also

- Ctfrrsp — Reassigned Spectrogram

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000

Cwindow

Creates a window of length N with a given shape.

Calling Sequence

```
H = Cwindow(N, NAME)
H = Cwindow(N, NAME, PARAM)
H = Cwindow(N, NAME, PARAM, PARAM2)
```

Parameters

N :

a positive integer value: the length of the window

NAME :

a character string : the name of the window shape (see below)

PARAM :

a real number: an optional parameter for 'Kaiser', "Spline" or "Nutbess" windows

PARAM2 :

a real number: a second optional parameters for "Spline" or "Nutbess" windows

H :

a real column vector of length N: The window coefficients

Description

Creates a window of length N with a given shape.

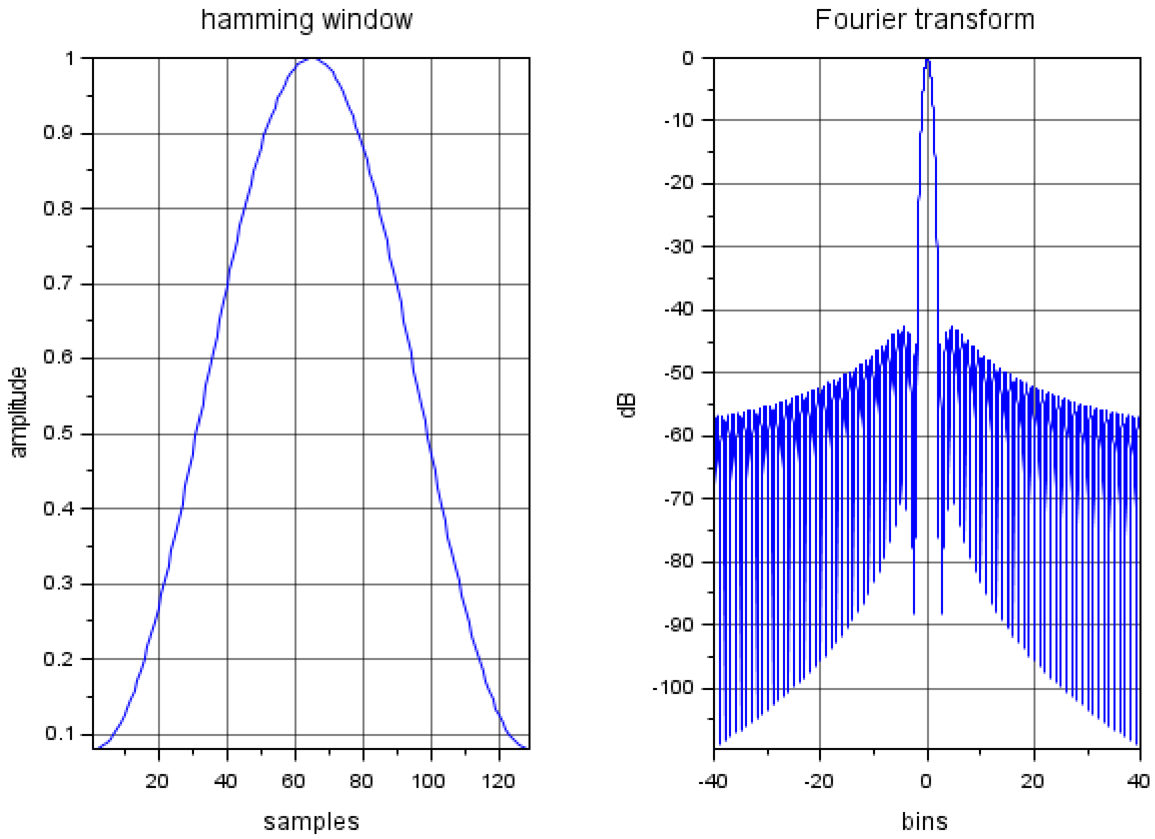
Possible names are :

'Hamming', 'Hanning', 'Kaiser', 'Nuttall', 'Papoulis', 'Sine', 'Harris', 'Rect', 'Triang', 'Bartlett', 'BartHann', 'Blackman' 'Gauss', 'Parzen', 'Powersine', 'Nutbess', 'spline', 'Flattop_ni', 'Flattop', 'Flattop_m', 'Flattop_srs'

this function is similar to tftb_window but it is coded in C so it is more efficient.

- **The Hamming window** is defined by:

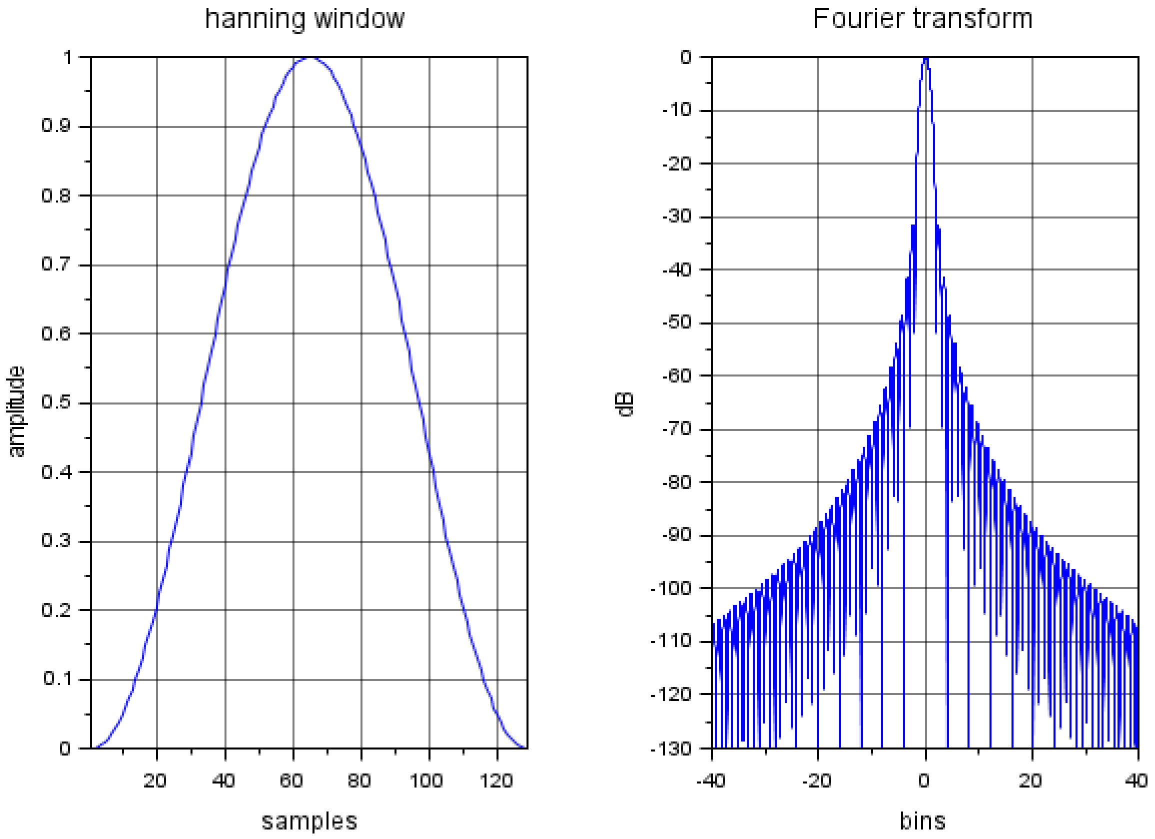
$$H(n+1) = a_0 - (1-a_0)\cos\left(\frac{2\pi n}{N-1}\right), \quad 0 \le n \le N-1 \text{ with } a_0 = 0.54$$



- **The Hann (Hanning) window** is defined by:

$$H(n+1) = a_0 - \underbrace{(1-a_0)}_{a_1} \cdot \cos\left(\frac{2\pi n}{N-1}\right), \quad 0 \le n \le N-1$$

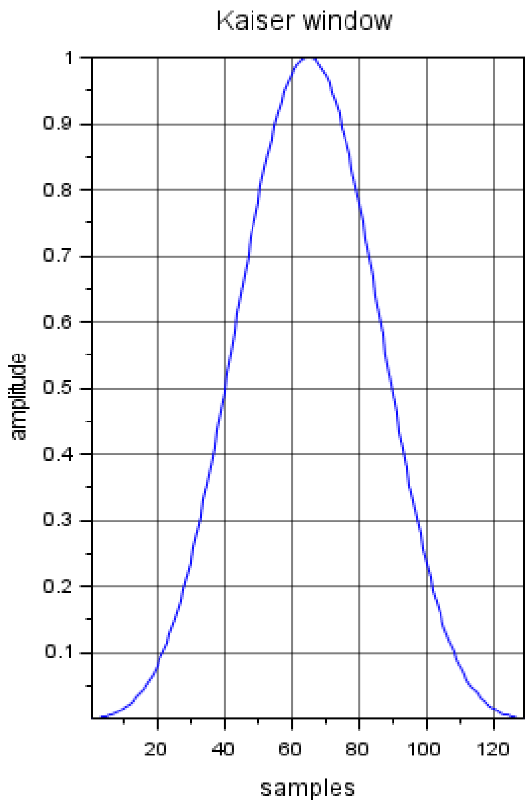
with $a_0 = 0.5$, $a_1 = 1 - a_0$

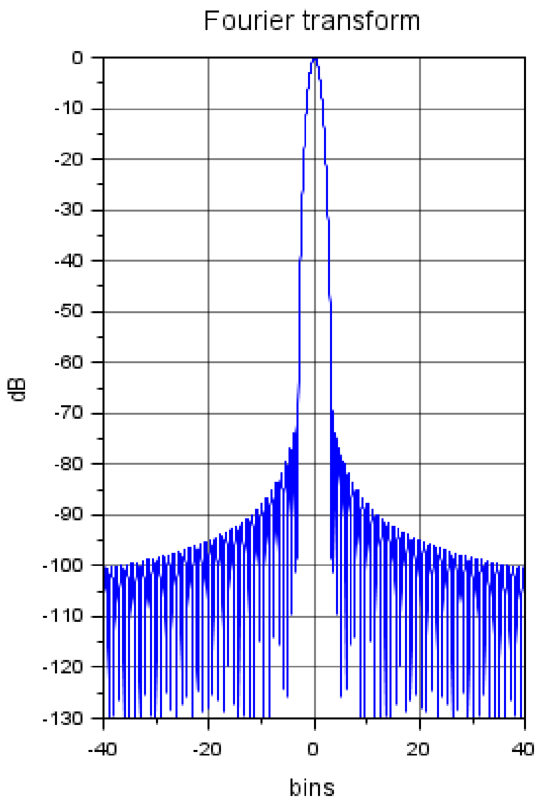


- **The Kaiser window** is defined by:

$$H(n+1) = \frac{I_0\left(\beta\sqrt{1-(\frac{2n}{N-1})^2}\right)}{I_0(\pi\alpha)} w_0(n) = \frac{I_0\left(\pi\alpha\sqrt{1-(\frac{2n}{N-1})^2}\right)}{I_0(\beta)}, \quad 0 \leq n \leq N-1$$

The optional parameter PARAM gives the beta value. Its default value is 3π.

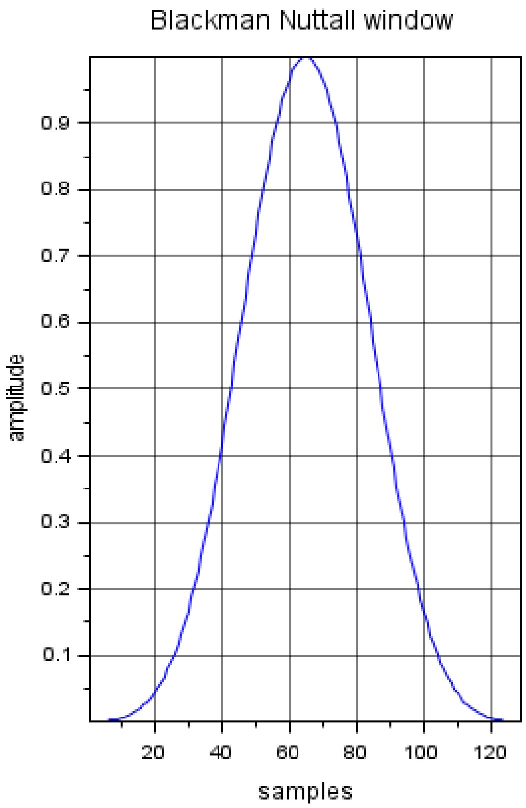


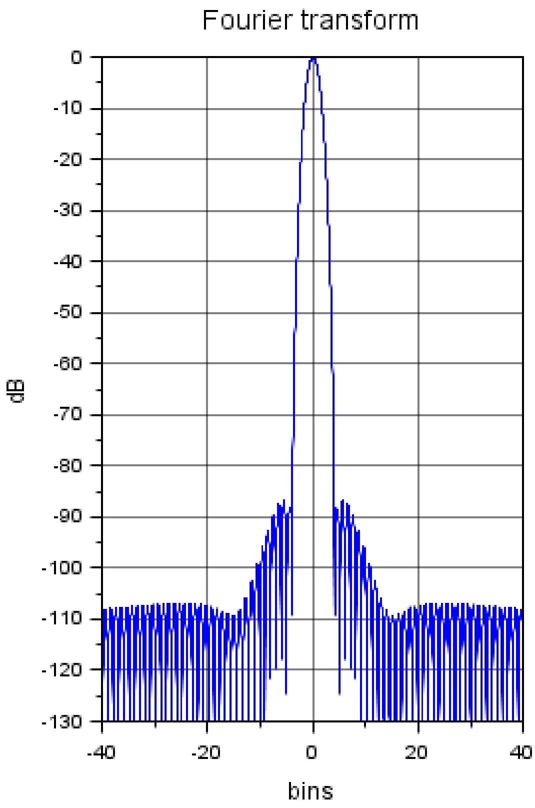


- **The Nuttall (Blackman Nuttall) window** is defined by

$$H(n+1) = a_0 - a_1\cos\left(\frac{2\pi n}{N-1}\right) + a_2\cos\left(\frac{4\pi n}{N-1}\right) - a_3\cos\left(\frac{6\pi n}{N-1}\right), \quad 0 \leq n \leq N-1$$

with a_0=0.3635819, a_1=0.4891775, a_2=0.1365995, a_3=0.0106411

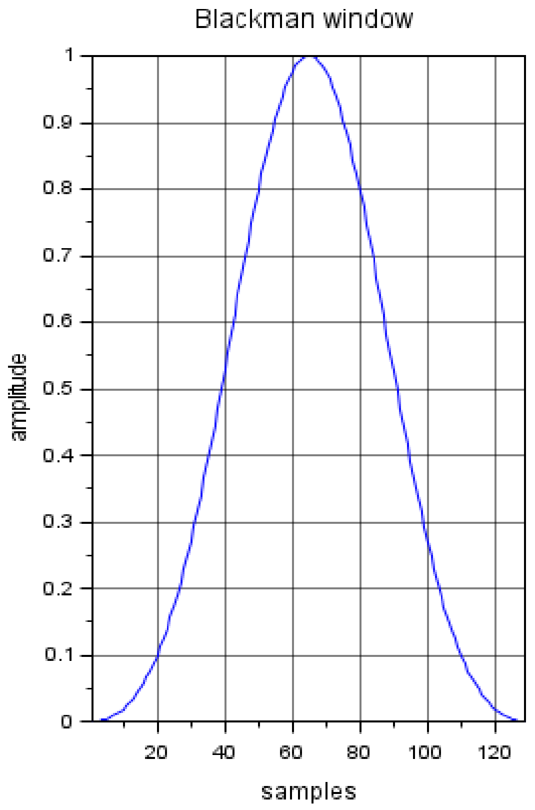


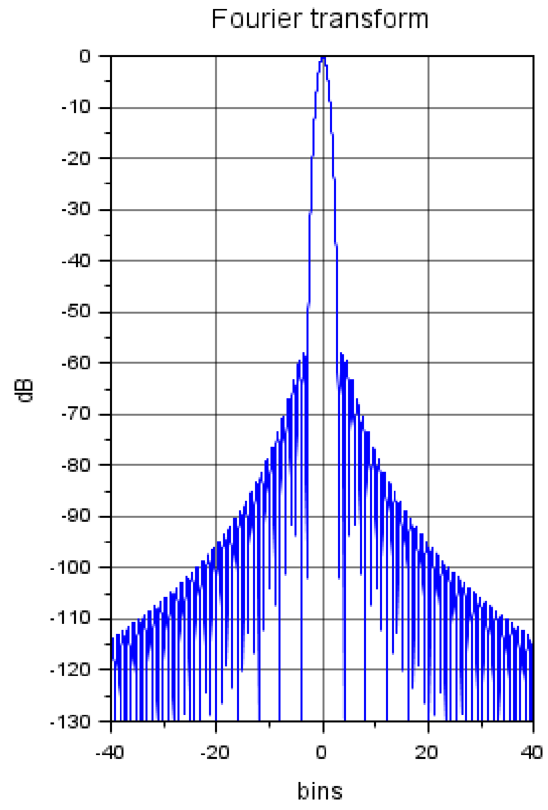


- **The Blackman window** is defined by:

$$H(n+1) = a_0 - a_1 \cos\left(\frac{2\pi n}{N-1}\right) + a_2 \cos\left(\frac{4\pi n}{N-1}\right), \quad 0 \le n \le N-1$$

with $\alpha=0.16$, $a_0=(1-\alpha)/2$, $a_1=1/2$, $a_2=\alpha/2$

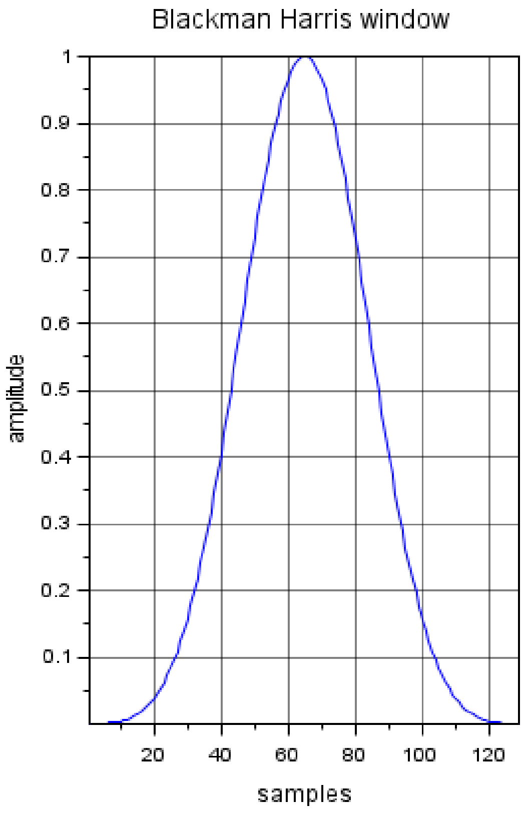


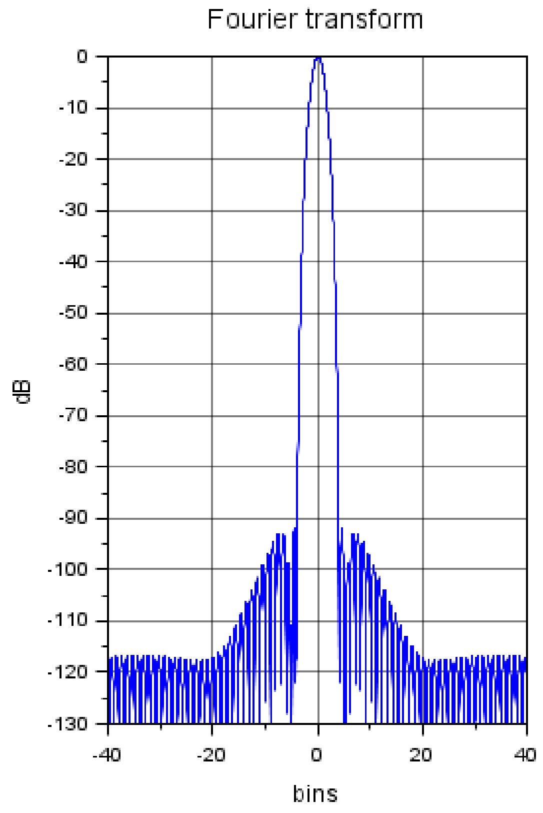


- **The Harris (Blackman Harris) window** is defined by:

$$H(n+1) = a_0 - a_1 \cos\left(\frac{2\pi n}{N-1}\right) + a_2 \cos\left(\frac{4\pi n}{N-1}\right) - a_3 \cos\left(\frac{6\pi n}{N-1}\right), \quad 0 \le n \le N-1$$

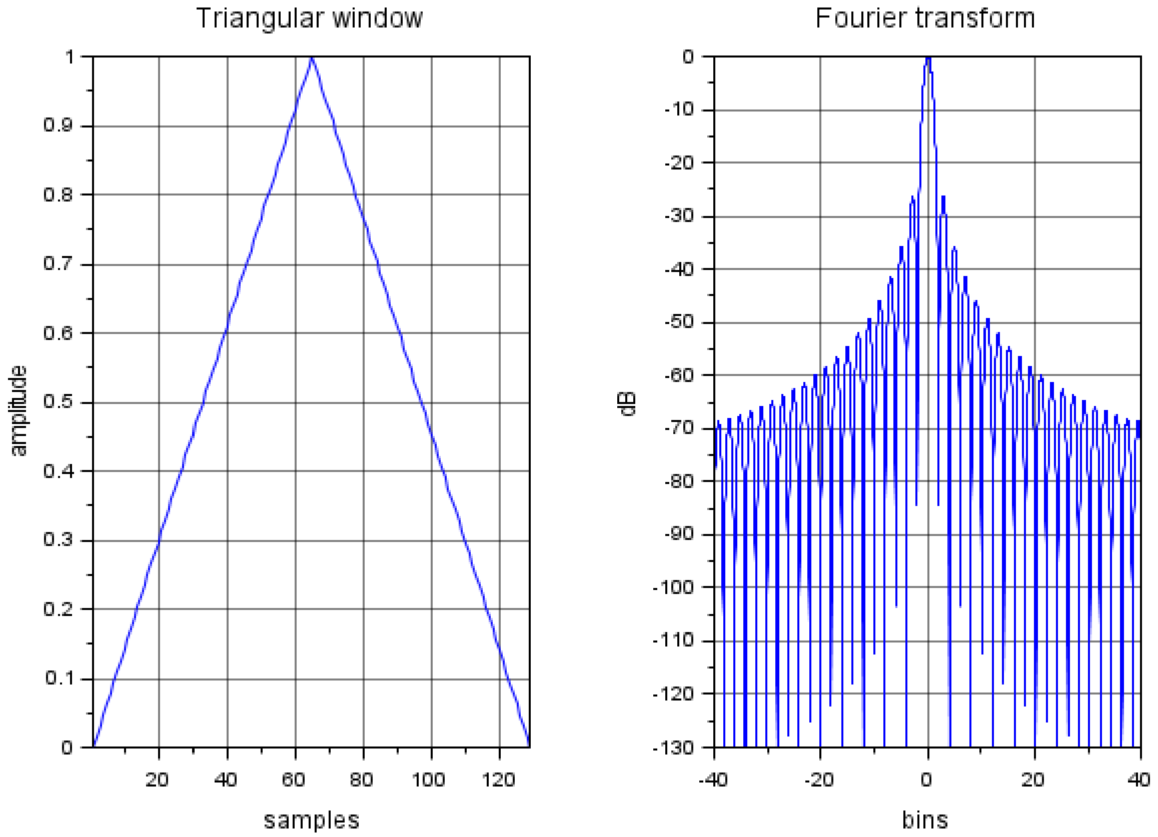
with $a_0=0.35875$, $a_1=0.48829$, $a_2=0.14129$, $a_3=0.01168$





- **The rectangular window** is defined by $H(n+1)=1,\quad 0\le n\le N-1$
- **The Triangular or Bartlett window** is defined by

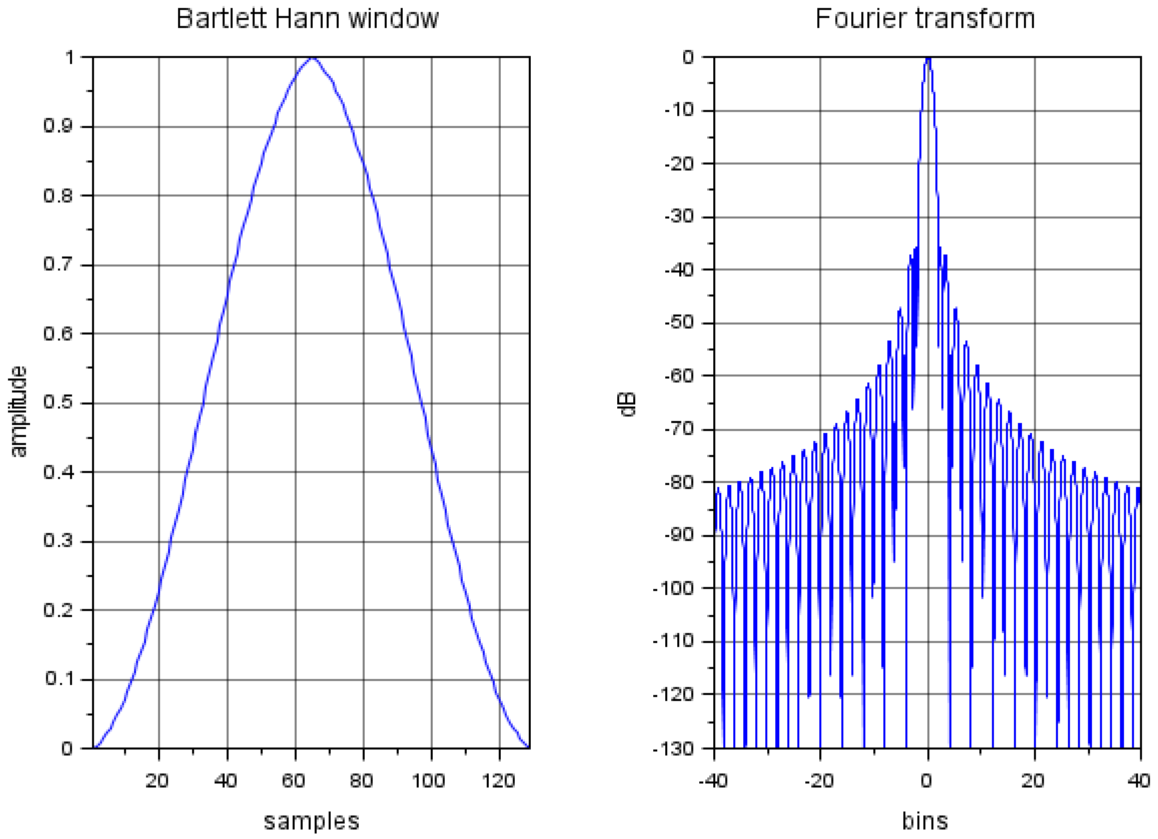
$$H(n+1)=1-\left|\frac{2n}{N-1}-1\right|,\quad 0\le n\le N-1$$



- **The Barthann (Bartlett Hann) window** is defined by

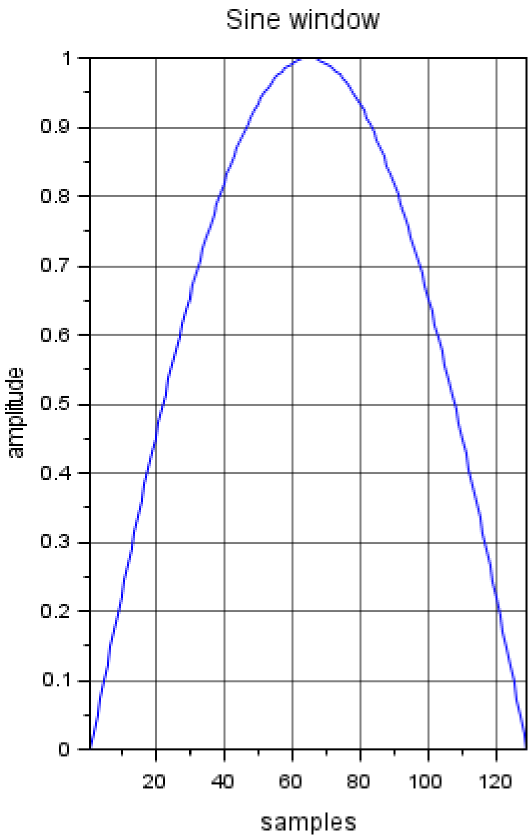
$$H(n+1)=a_0-a_1\left|\frac{n}{N-1}-\frac{1}{2}\right|-a_2\cos\left(\frac{2\pi n}{N-1}\right),\quad 0\le n\le N-1$$

with a_0=0.62, a_1=0.48, a_2=0.38

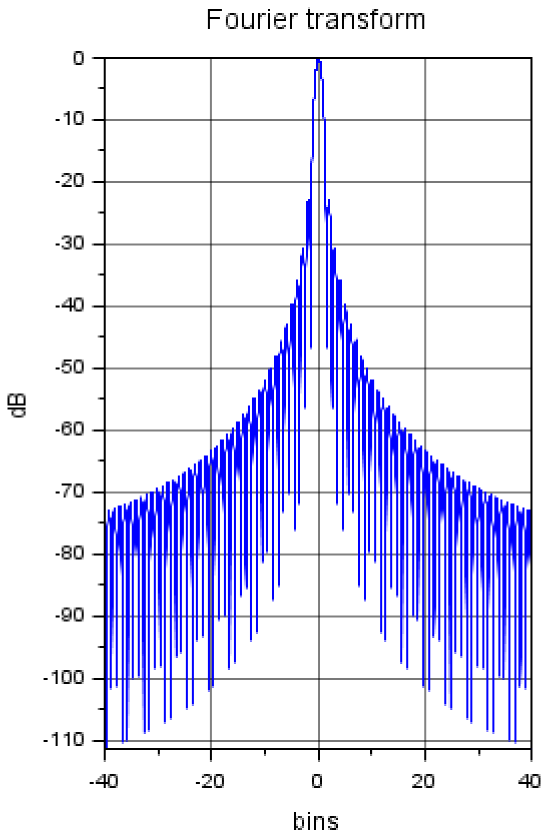


- **The Papoulis or Sine window** is defined by:

$$H(n+1)=\sin\left(\frac{n\pi}{N-1}\right);\quad 0\leq n\leq N-1$$



Sine window

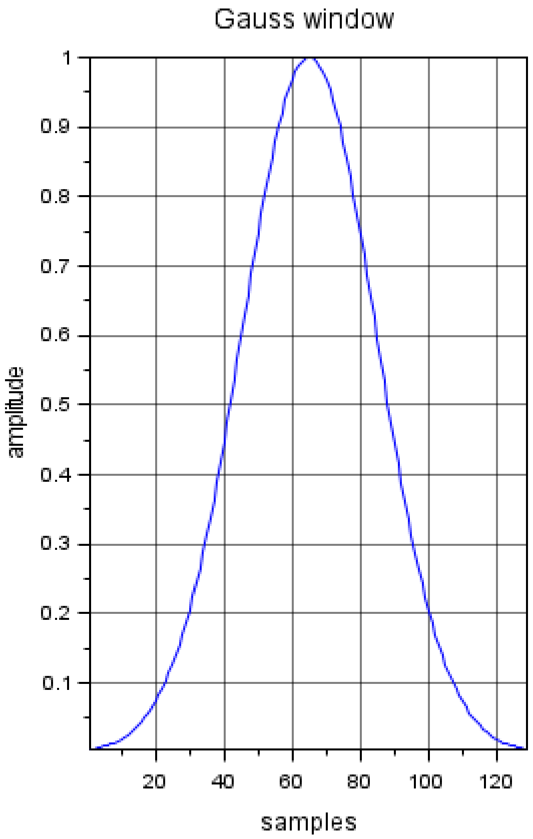


Fourier transform

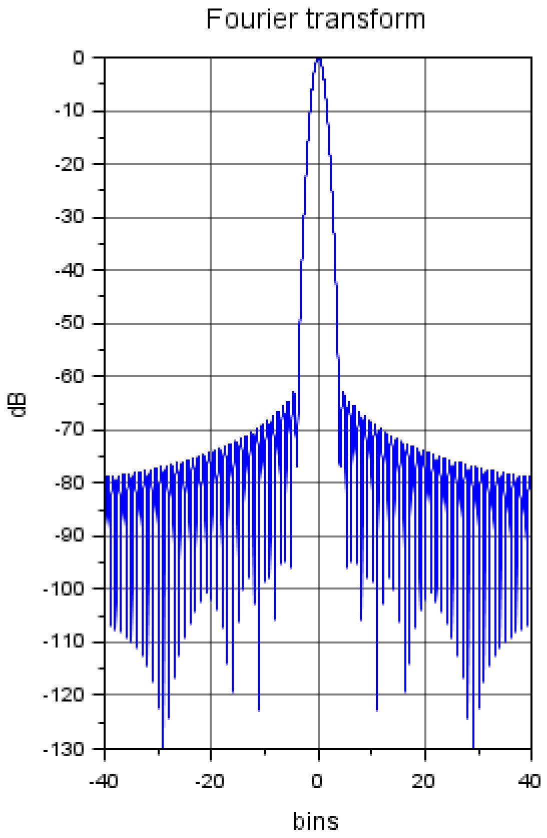
- **The Gauss window** is defined by:

$$H(n+1)=\exp\left(-1+\frac{2*n}{N-1}^2\log(K)\right);\quad 0\leq n\leq N-1$$

For the gaussian window, the optionnal parameter K sets the value at both extremities. The default value is 0.005



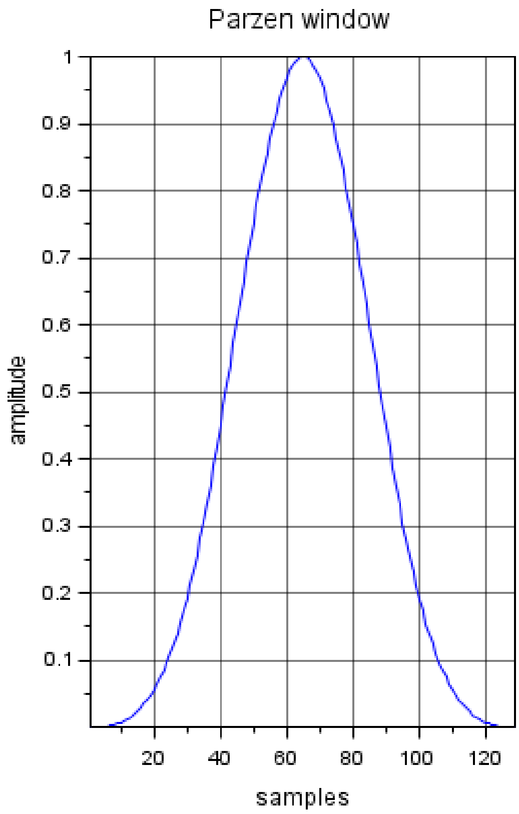
Gauss window

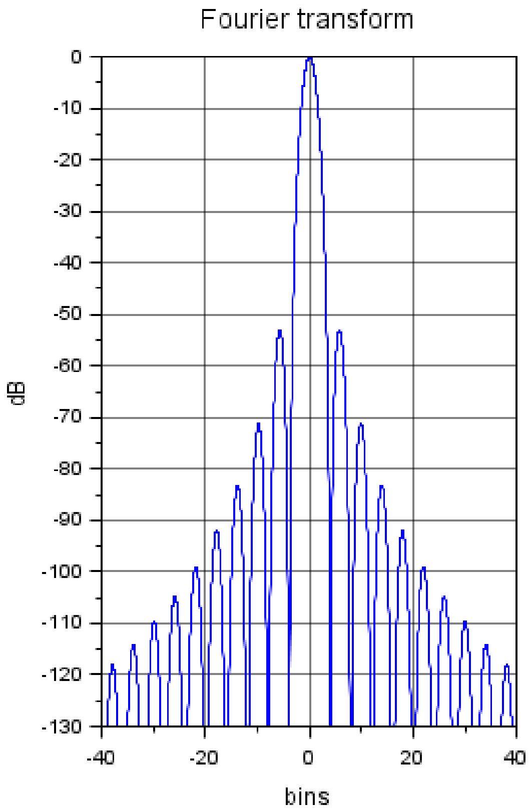


Fourier transform

- **The Parzen window** is defined by

$$H(n+1) = \begin{cases} 1-6\left(\frac{n}{N/2}\right)^2\left(1-\frac{|n|}{N/2}\right), & 0 \leqslant |n| \leqslant \frac{N}{4} \\ 2\left(1-\frac{|n|}{N/2}\right)^3, & \frac{N}{4} \leq |n| \leq \frac{N}{2} \end{cases} \quad ; \quad 0 \leq n \leq N-1$$

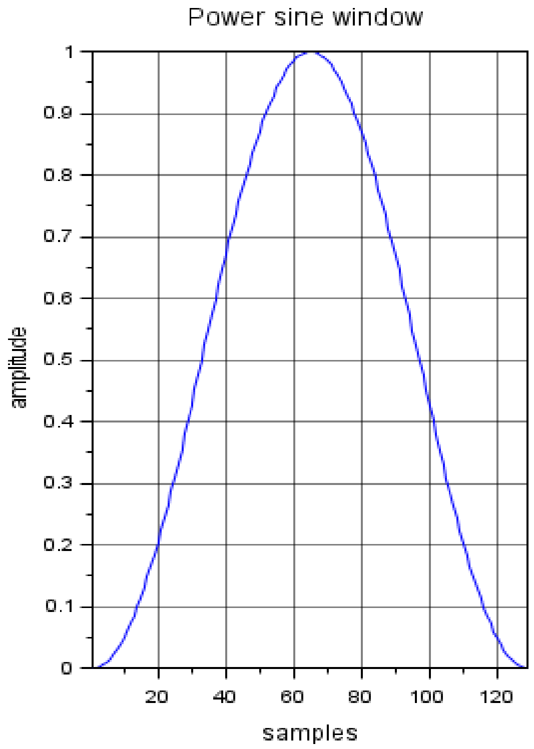


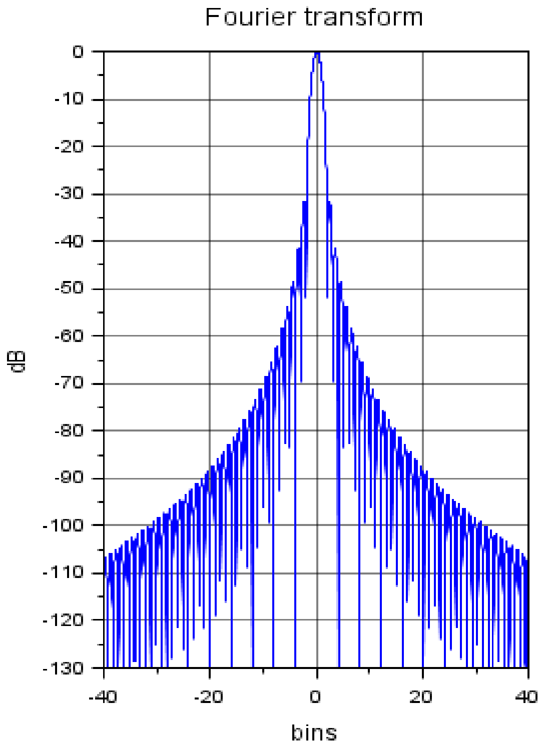


- **The Powersine (Power of sine) windows** are defined by:

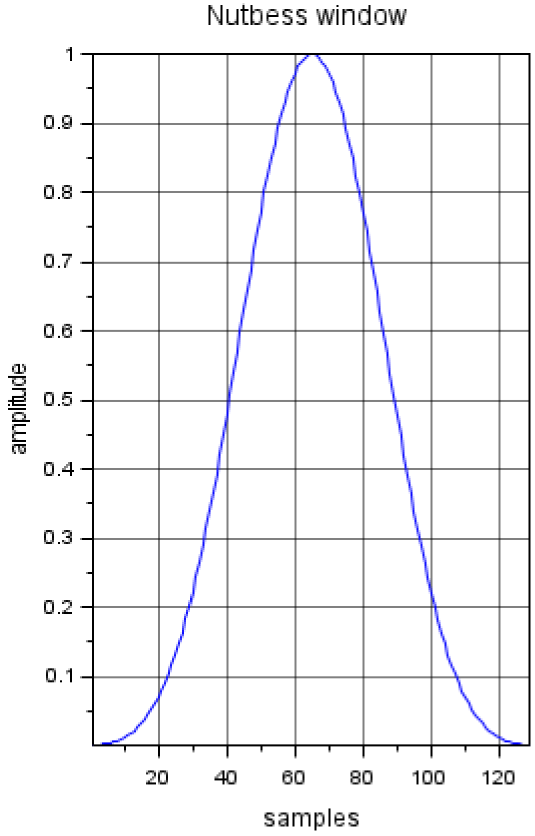
$$H(n+1) = \sin\left(\frac{n\pi}{N-1}\right)^{2L}; \quad 0 \leq n \leq N-1$$

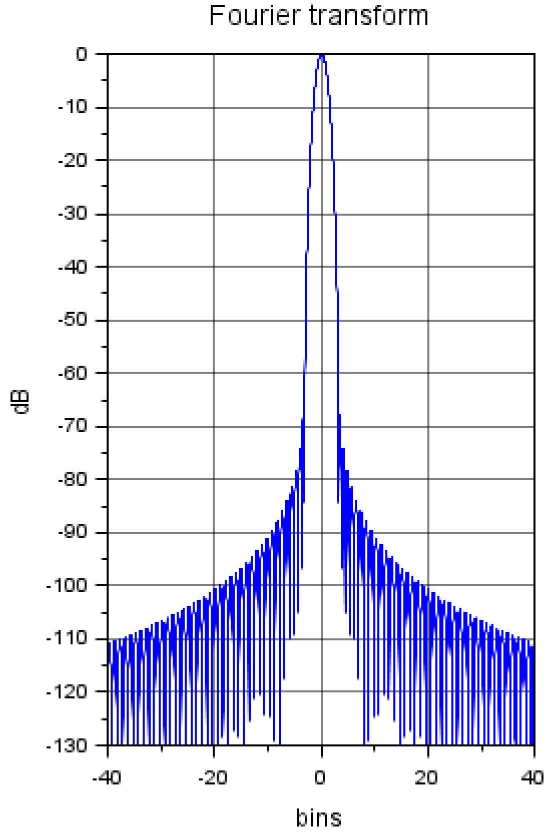
Where L is the optionnal parameter (defaut value: 1). The rectangular window (L=0), the sine window (L=1/2), and the Hann window (L=1) are particular cases.





- **The Nutbess windows** are defined by





- **The Spline window** is defined by

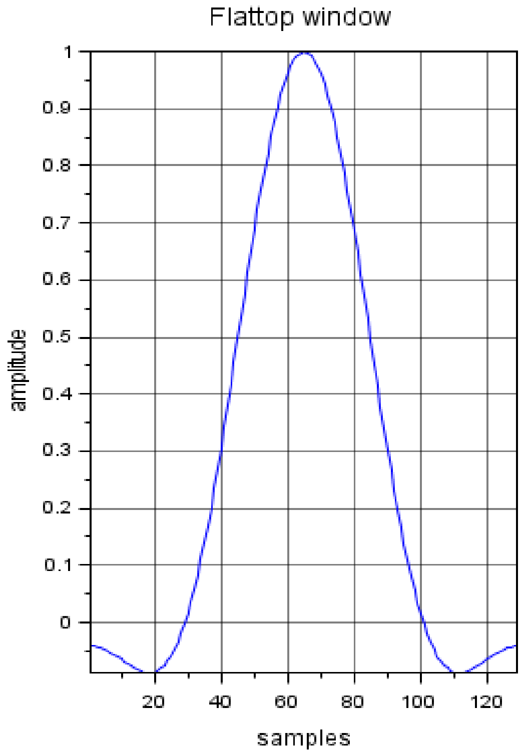
$$H(n+1) = sinc(\frac{n freq}{2p}(-fracN - 12 + n))^p;$$.

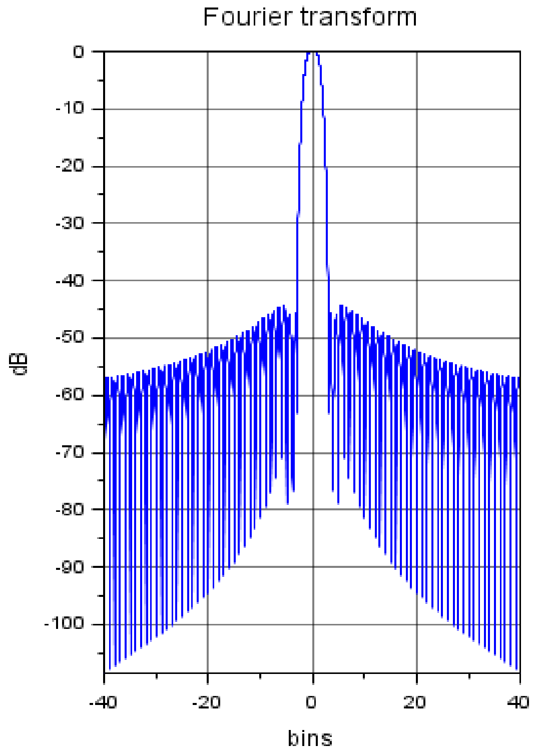
where nfreq (frequency bandwidth) is given by the arguement PARAM and the spline order p is given by the optional parameter PARAM2 (default value π.N.nfreq/10).

- **The Flattop and Flattop_ni windows** are defined by

$$H(n+1) = a_0 - a_1 \cos\left(\frac{2\pi n}{N-1}\right) + a_2 \cos\left(\frac{4\pi n}{N-1}\right), \quad 0 \leq n \leq N-1$$

with a_0 = 0.2810639, a_1 = 0.5208972, a_2 = 0.1980399

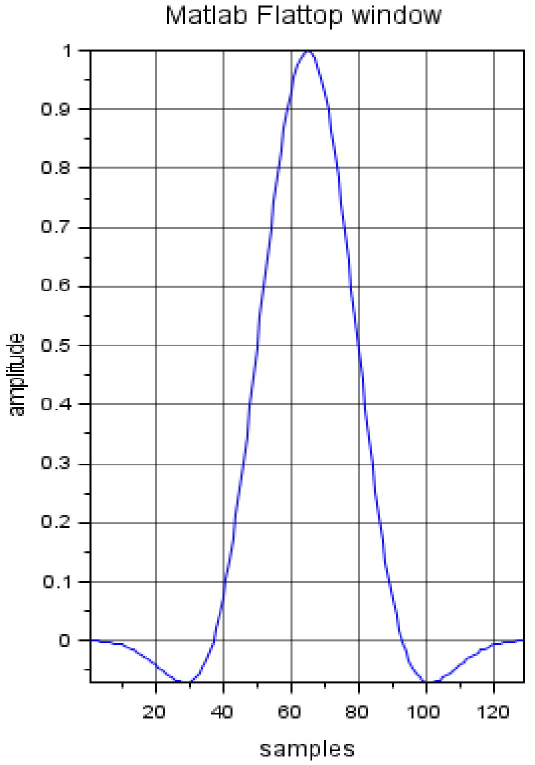


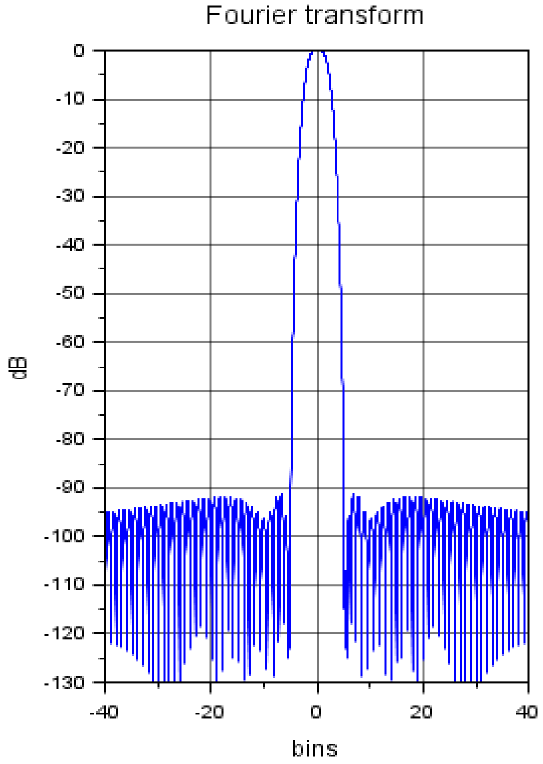


- **The Flattop_m (Matlab equivalent) window** is defined by:

$$H(n+1) = a_0 - a_1 \cos\left(\frac{2\pi n}{N-1}\right) + a_2 \cos\left(\frac{4\pi n}{N-1}\right) - a_3 \cos\left(\frac{6\pi n}{N-1}\right) + a_4 \cos\left(\frac{8\pi n}{N-1}\right); \quad 0 \le n \le N-1$$

with a_0 = .21557895, a_1 = 0.41663158, a_2 = 0.277263158, a_3 = 0.083578947, a_4 = 0.006947368

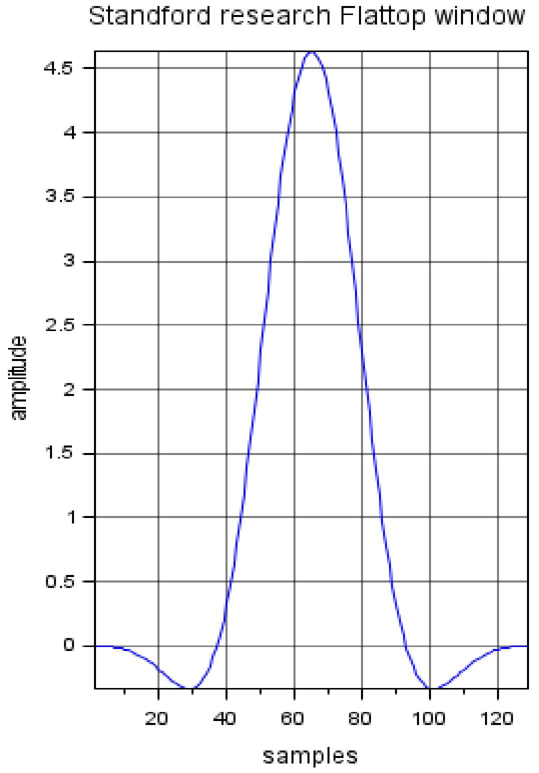


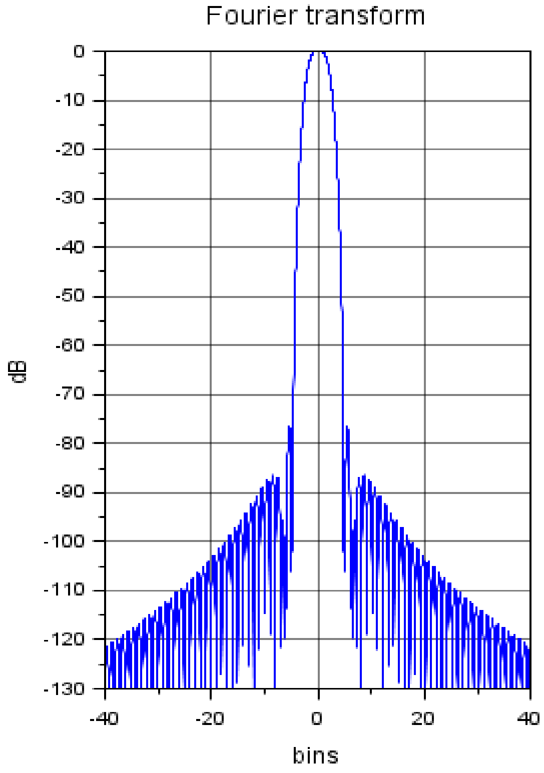


- **The Flattop_srs (Standford Research) window** is defined by:

$$H(n+1) = a_0 - a_1 \cos\left(\frac{2\pi n}{N-1}\right) + a_2 \cos\left(\frac{4\pi n}{N-1}\right) - a_3 \cos\left(\frac{6\pi n}{N-1}\right) + a_4 \cos\left(\frac{8\pi n}{N-1}\right); \quad 0 \le n \le N-1$$

with a_0 = 1, a_1 = 1.93, a_2 = 1.29, a_3 = 0.388, a_4 = 0.028





Examples

```
h = Cwindow(256,'Gauss',0.005); plot(h);
```

See Also

- tftb_window — Window generation
- window

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000
- Serge Steer - 2018

Time-Frequency representations (in C)

- Ctfrbj — Born-Jordan time-frequency distribution
- Ctfrbud — Butterworth time-frequency distribution
- Ctfrcw — Choi-Williams time-frequency distribution
- Ctfrgrd — Generalized rectangular time-frequency distribution
- Ctfrmh — Margenau-Hill time-frequency distribution
- Ctfrmhs — Margenau-Hill-Spectrogram time-frequency distribution
- Ctfrmmce — Minimum mean cross-entropy combination of spectrograms
- Ctfrpage — Page time-frequency distribution
- Ctfrpmh — Pseudo Margenau-Hill time-frequency distribution
- Ctfrppage — Pseudo Page time-frequency distribution
- Ctfrpwv — Pseudo Wigner-Ville time-frequency distribution
- Ctfrri — Rihaczek time-frequency distribution
- Ctfrridb — Reduced Interference Distribution with Bessel kernel
- Ctfrridbn — Reduced Interference Distribution with binomial kernel
- Ctfrridh — Reduced Interference Distribution with Hanning kernel
- Ctfrridt — Reduced Interference Distribution with Triangular kernel
- Ctfrrsp — Reassigned Spectrogram
- Ctfrsp — Spectrogram Time Frequency distribution of a signal X
- Ctfrspwv — Smoothed Pseudo Wigner-Ville time-frequency distribution
- Ctfrstft — Short time Fourier transform
- Ctfrwv — Wigner-Ville time-frequency distribution
- Ctfrzam — Zao-Atlas-Marks time-frequency distribution

Ctfrbj

Born-Jordan time-frequency distribution

Calling Sequence

```
[TFR,T,F] = Ctfrbj(X)
[TFR,T,F] = Ctfrbj(X, T)
[TFR,T,F] = Ctfrbj(X, T, N)
[TFR,T,F] = Ctfrbj(X, T, N, G)
[TFR,T,F] = Ctfrbj(X, T, N, G, H)
```

Parameters

X :

Analyzed signal.

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

G :

time smoothing window, (default : Hamming(N/10)).

H :

frequency smoothing window, (default : Hamming(N/4)).

TFR :

time-frequency representation.

F :

vector of normalized frequencies.

Description

computes the Born-Jordan distribution of a signal X.

Examples

```
x = hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal'));
g = Cwindow(9,'Hamming');
h = Cwindow(27,'Hamming');
```

```
t = 1:128;
[tfr,T,F] = Ctfrbj(x,t,128,g,h);
grayplot(T,F,tfr');
xlabel time
ylabel frequency
```

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000

Ctfrbud

Butterworth time-frequency distribution

Calling Sequence

```
[TFR,T,F] = Ctfrbud(X)
[TFR,T,F] = Ctfrbud(X, T)
[TFR,T,F] = Ctfrbud(X, T, N)
[TFR,T,F] = Ctfrbud(X, T, N, G)
[TFR,T,F] = Ctfrbud(X, T, N, G, H)
[TFR,T,F] = Ctfrbud(X, T, N, G, H, SIGMA)
```

Parameters

X :

Analyzed signal

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

G :

time smoothing window (default : Hamming(N/4)).

H :

frequency smoothing window(default : Hamming(N/4)).

SIGMA :

kernel width (default : 1).

TFR :

time-frequency representation

F :

vector of normalized frequencies.

Description

computes the Butterworth distribution of a signal X.

Examples

```
x = hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal'));
g = Cwindow(9,'Hamming');
h = Cwindow(27,'Hamming');
t = 1:128;
[tfr,T,F] = Ctfrbud(x,t,128,g,h,3.6);
grayplot(T,F,tfr');
xlabel time
ylabel frequency
```

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000
- Manuel Davy - 2000

Ctfrcw

Choi-Williams time-frequency distribution

Calling Sequence

```
[TFR,T,F] = Ctfrcw(X)
[TFR,T,F] = Ctfrcw(X, T)
[TFR,T,F] = Ctfrcw(X, T, N)
[TFR,T,F] = Ctfrcw(X, T, N, G)
[TFR,T,F] = Ctfrcw(X, T, N, G, H)
[TFR,T,F] = Ctfrcw(X, T, N, G, H, SIGMA)
```

Parameters

X :

Analyzed signal.

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

G :

time smoothing window, (default : Hamming(N/10)).

H :

frequency smoothing window, in the time-domain, (default : Hamming(N/4)).

SIGMA :

kernel width

TFR :

time-frequency representation.

F :

vector of normalized frequencies.

Description

computes the Choi-Williams distribution of a signal X.

Examples

```
x = hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal'));
g = Cwindow(9,'Hamming');
h = Cwindow(27,'Hamming');
t = 1:128;
[tfr,T,F] = Ctfrcw(x,t,128,g,h,1);
grayplot(T,F,tfr');
xlabel time
ylabel frequency
```

See also

- Ctfrwv — Wigner-Ville time-frequency distribution
- Ctfrsp — Spectrogram Time Frequency distribution of a signal X

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000
- Manuel Davy - 2000

Ctfrgrd

Generalized rectangular time-frequency distribution

Calling Sequence

```
[TFR,T,F] = Ctfrgrd(X)
[TFR,T,F] = Ctfrgrd(X, T)
[TFR,T,F] = Ctfrgrd(X, T, N)
[TFR,T,F] = Ctfrgrd(X, T, N, G)
[TFR,T,F] = Ctfrgrd(X, T, N, G, H)
[TFR,T,F] = Ctfrgrd(X, T, N, G, H, RS)
[TFR,T,F] = Ctfrgrd(X, T, N, G, H, RS, MOVERN)
```

Parameters

X :

Analyzedd signal

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X))

G :

time smoothing window (default : Hamming(N/10)).

H :

frequency smoothing window (default : Hamming(N/4)).

RS :

kernel width (default : 1).

MOVERN :

dissymmetry ratio (default : 1).

TFR :

time-frequency representation

F :

vector of normalized frequencies.

Description

computes the Generalized Rectangular distribution of a signal X.

Examples

```
x = hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal'));
g = Cwindow(9,'hamming');
h = Cwindow(27,'hamming');
t = 1:128;
[tfr,T,F] = Ctfrgrd(x,t,128,g,h,36,1/5);
grayplot(T,F,tfr');
xlabel time
ylabel frequency
```

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000

Ctfrmh

Margenau-Hill time-frequency distribution

Calling Sequence

```
[TFR,T,F] = Ctfrmh(X)
[TFR,T,F] = Ctfrmh(X, T)
[TFR,T,F] = Ctfrmh(X, T, N)
```

Parameters

X :

Analyzed signal.

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

TFR :

time-frequency representation.

F :

vector of normalized frequencies.

Description

computes the Margenau-Hill distribution of a signal X.

Examples

```
x = hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal'));
t = 1:128;
[tfr,T,F] = Ctfrmh(x,t,128);
grayplot(T,F,tfr');
xlabel time
ylabel frequency
```

See also

- Ctfrpmh — Pseudo Margenau-Hill time-frequency distribution
- Ctfrwv — Wigner-Ville time-frequency distribution

- Ctfrri — Rihaczek time-frequency distribution

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000

Ctfrmhs

Margenau-Hill-Spectrogram time-frequency distribution

Calling Sequence

```
[TFR,T,F] = Ctfrmhs(X)
[TFR,T,F] = Ctfrmhs(X, T)
[TFR,T,F] = Ctfrmhs(X, T, N)
[TFR,T,F] = Ctfrmhs(X, T, N, G)
[TFR,T,F] = Ctfrmhs(X, T, N, G, H)
```

Parameters

X :

Analyzed signal.

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

G :

time smoothing window, (default : Hamming(N/10)).

H :

frequency smoothing window, in the time-domain, (default : Hamming(N/4)).

TFR :

time-frequency representation.

F :

vector of normalized frequencies.

Description

computes the Margenau-Hill-Spectrogram distribution of a signal X.

Examples

```
x = hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal'));
g = Cwindow(21,'Hamming');
h = Cwindow(63,'Hamming');
t = 1:128;
```

```
[tfr,T,F] = Ctfrmhs(x,t,128,g,h);
grayplot(T,F,tfr');
xlabel time
ylabel frequency
```

See also

- Ctfrmh — Margenau-Hill time-frequency distribution
- Ctfrpmh — Pseudo Margenau-Hill time-frequency distribution
- Ctfrsp — Spectrogram Time Frequency distribution of a signal X

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000

stftb > Time-Frequency representations (in C) > Ctfrmmce

Ctfrmmce

Minimum mean cross-entropy combination of spectrograms

Calling Sequence

```
[TFR,T,F] = Ctfrmmce(X)
[TFR,T,F] = Ctfrmmce(X, H)
[TFR,T,F] = Ctfrmmce(X, H, T)
[TFR,T,F] = Ctfrmmce(X, H, T, N)
```

Parameters

X :

Analyzed signal.

H :

frequency smoothing windows, stored in a matrix. Each row is a window

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X))

TFR :

time-frequency representation.

F :

vector of normalized frequencies.

Description

computes the minimum mean cross-entropy combination of spectrograms using aswindows the columns of the matrix H

Examples

```
x = hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal'));
h(10+(-5:5),1) = Cwindow(11,'gauss');
h(10+(-7:7),2) = Cwindow(15,'gauss');
h(10+(-9:9),3) = Cwindow(19,'gauss');
[tfr,T,F] = Ctfrmmce(x,h);
grayplot(T,F,tfr');
```

```
xlabel time
ylabel frequency
```

See also

- Ctfrsp — Spectrogram Time Frequency distribution of a signal X

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000

Ctfrpage

Page time-frequency distribution

Calling Sequence

```
[TFR,T,F] = Ctfrpage(X)
[TFR,T,F] = Ctfrpage(X, T)
[TFR,T,F] = Ctfrpage(X, T, N)
```

Parameters

X :

Analyzed signal.

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

TFR :

time-frequency representation.

F :

vector of normalized frequencies.

Examples

```
x = hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal'));
t = 1:128;
[tfr,T,F] = Ctfrpage(x,t,128);
grayplot(T,F,tfr');
xlabel time
ylabel frequency
```

Description

Computes the Page distribution for a signal X.

See also

- Ctfrri — Rihaczek time-frequency distribution
- Ctfrppage — Pseudo Page time-frequency distribution

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000
- Manuel Davy - 2000

Ctfrpmh

Pseudo Margenau-Hill time-frequency distribution

Calling Sequence

```
[TFR,T,F] = Ctfrpmh(X)
[TFR,T,F] = Ctfrpmh(X, T)
[TFR,T,F] = Ctfrpmh(X, T, N)
[TFR,T,F] = Ctfrpmh(X, T, N, H)
```

Parameters

X :

Analyzed signal.

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

H :

frequency smoothing window (default : Hamming(N/4)).

TFR :

time-frequency representation.

F :

vector of normalized frequencies.

Description

computes the Pseudo Margenau-Hill distribution of a signal X.

Ctfrppage

Pseudo Page time-frequency distribution

Calling Sequence

```
[TFR,T,F] = Ctfrppage(X)
[TFR,T,F] = Ctfrppage(X, T)
[TFR,T,F] = Ctfrppage(X, T, N)
```

Parameters

T = time instant(s) (default :

1:length(X)).

N = number of frequency bins (default :

length(X)).

H = frequency smoothing window (default :

Hamming(N/4)).

Description

Computes the pseudo Page distribution for a signal X.

Examples

```
x = hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal'));
t = 1:128;
[tfr,T,F] = Ctfrppage(x,t,128);
grayplot(T,F,tfr');
xlabel time
ylabel frequency
```

See also

- Ctfrri — Rihaczek time-frequency distribution
- Ctfrpage — Page time-frequency distribution

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000
- Manuel Davy - 2000

Ctfrpwv

Pseudo Wigner-Ville time-frequency distribution

Calling Sequence

```
[TFR,T,F] = Ctfrpwv(X)
[TFR,T,F] = Ctfrpwv(X, T)
[TFR,T,F] = Ctfrpwv(X, T, N)
[TFR,T,F] = Ctfrpwv(X, T, N, H)
```

Parameters

X :

Analyzed signal.

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

H :

frequency smoothing window, in the time-domain, (default : Hamming(N/4)).

TFR :

time-frequency representation.

F :

vector of normalized frequencies.

Description

computes the Pseudo Wigner-Ville distribution of a signal X.

Examples

```
x = hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal'));
t = 1:128;
[tfr,T,F] = Ctfrpwv(x,t,128);
grayplot(T,F,tfr');
xlabel time
ylabel frequency
```

See also

- Ctfrwv — Wigner-Ville time-frequency distribution
- Ctfrspwv — Smoothed Pseudo Wigner-Ville time-frequency distribution

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000

Ctfrri

Rihaczek time-frequency distribution

Calling Sequence

```
[TFR,T,F] = Ctfrri(X)
[TFR,T,F] = Ctfrri(X, T)
[TFR,T,F] = Ctfrri(X, T, N)
```

Parameters

X :

Analyzed signal.

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

TFR :

time-frequency representation.

F :

vector of normalized frequencies.

Description

Computes the Rihacez distribution for a signal X.

Examples

```
x = hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal'));
t = 1:128;
[tfr,T,F] = Ctfrri(x,t,128);
grayplot(T,F,abs(tfr)');
xlabel time
ylabel frequency
```

See also

- Ctfrridb — Reduced Interference Distribution with Bessel kernel
- Ctfrridh — Reduced Interference Distribution with Hanning kernel

- Ctfrridt — Reduced Interference Distribution with Triangular kernel

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000
- Manuel Davy - 2000

Ctfrridb

Reduced Interference Distribution with Bessel kernel

Calling Sequence

```
[TFR,T,F] = Ctfrridb(X)
[TFR,T,F] = Ctfrridb(X, T)
[TFR,T,F] = Ctfrridb(X, T, N)
[TFR,T,F] = Ctfrridb(X, T, N, G)
[TFR,T,F] = Ctfrridb(X, T, N, G, H)
```

Parameters

X :

Analyzed signal.

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

G :

time smoothing window, (default : Hamming(N/10)).

H :

frequency smoothing window (default : Hamming(N/4)).

TFR :

time-frequency representation.

F :

vector of normalized frequencies.

Description

Computes the Reduced Interference Distribution with a kernel based on the Bessel function of the first kind, for a signal X.

Examples

```
x = hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal'));
g = Cwindow(31,'rect');
h = Cwindow(63,'rect');
```

```
t = 1:128;
[tfr,T,F] = Ctfrridb(x,t,128,g,h);
grayplot(T,F,tfr');
xlabel time
ylabel frequency
```

See also

- Ctfrridh — Reduced Interference Distribution with Hanning kernel
- Ctfrridt — Reduced Interference Distribution with Triangular kernel
- Ctfrridbn — Reduced Interference Distribution with binomial kernel

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000
- Manuel Davy - 2000

Ctfrridbn

Reduced Interference Distribution with binomial kernel

Calling Sequence

```
[TFR,T,F] = Ctfrridbn(X)
[TFR,T,F] = Ctfrridbn(X, T)
[TFR,T,F] = Ctfrridbn(X, T, N)
[TFR,T,F] = Ctfrridbn(X, T, N, G)
// [TFR,T,F] = Ctfrridbn(X, T, N, G, H)
```

Parameters

X :

Analyzed signal.

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

G :

time smoothing window, (default : Hamming(N/10)).

H :

frequency smoothing window (default : Hamming(N/4)).

TFR :

time-frequency representation.

F :

vector of normalized frequencies.

Description

Computes the Reduced Interference Distribution with a kernel based on the binomial coefficients, for a signal X.

Examples

```
x = hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal'));
g = Cwindow(31,'rect');
h = Cwindow(63,'rect');
```

```
t = 1:128;
[tfr,T,F] = Ctfrridbn(x,t,128,g,h);
grayplot(T,F,tfr');
xlabel time
ylabel frequency
```

See also

- Ctfrridb — Reduced Interference Distribution with Bessel kernel
- Ctfrridh — Reduced Interference Distribution with Hanning kernel
- Ctfrridt — Reduced Interference Distribution with Triangular kernel

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000
- Manuel Davy - 2000

Ctfrridh

Reduced Interference Distribution with Hanning kernel

Calling Sequence

```
[TFR,T,F] = Ctfrridh(X)
[TFR,T,F] = Ctfrridh(X, T)
[TFR,T,F] = Ctfrridh(X, T, N)
[TFR,T,F] = Ctfrridh(X, T, N, G
[TFR,T,F] = Ctfrridh(X, T, N, G, H)
```

Parameters

X :

Analyzed signal.

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

G :

time smoothing window, (default : Hamming(N/10)).

H :

frequency smoothing window (default : Hamming(N/4)).

TFR :

time-frequency representation.

F :

vector of normalized frequencies.

Description

Computes the Reduced Interference Distribution with a kernel based on the Hanning window, for a signal X.

Examples

```
x = hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal'));
g = Cwindow(31,'rect');
h = Cwindow(63,'rect');
```

```
t = 1:128;
[tfr,T,F] = Ctfrridh(x,t,128,g,h);
grayplot(T,F,tfr');
xlabel time
ylabel frequency
```

See also

- Ctfrridb — Reduced Interference Distribution with Bessel kernel
- Ctfrridt — Reduced Interference Distribution with Triangular kernel
- Ctfrridbn — Reduced Interference Distribution with binomial kernel

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000
- Manuel Davy - 2000

Ctfrridt

Reduced Interference Distribution with Triangular kernel

Calling Sequence

```
[TFR,T,F] = Ctfrridt(X)
[TFR,T,F] = Ctfrridt(X, T)
[TFR,T,F] = Ctfrridt(X, T, N)
[TFR,T,F] = Ctfrridt(X, T, N, G)
[TFR,T,F] = Ctfrridt(X, T, N, G, H)
```

Parameters

X :

Analyzed signal.

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

G :

time smoothing window, (default : Hamming(N/10)).

H :

frequency smoothing window (default : Hamming(N/4)).

TFR :

time-frequency representation.

F :

vector of normalized frequencies.

Description

Computes the Reduced Interference Distribution with a kernel based on the Triangular window, for a signal X.

Examples

```
x = hilbert(sin(2*%pi*0.25*(1:128)) +
0.5*rand(1,128,'normal'));
g = Cwindow(31,'rect');
h = Cwindow(63,'rect');
```

```
t = 1:128;
[tfr,T,F] = Ctfrridt(x,t,128,g,h);
grayplot(T,F,tfr');
xlabel time
ylabel frequency
```

See also

- Ctfrridb — Reduced Interference Distribution with Bessel kernel
- Ctfrridh — Reduced Interference Distribution with Hanning kernel
- Ctfrridbn — Reduced Interference Distribution with binomial kernel

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000
- Manuel Davy - 2000

Ctfrrsp

Reassigned Spectrogram

Calling Sequence

```
[SP,SP_reas,FIELD] = Ctfrrsp(X)
[SP,SP_reas,FIELD] = Ctfrrsp(X, T)
[SP,SP_reas,FIELD] = Ctfrrsp(X, T, N)
[SP,SP_reas,FIELD] = Ctfrrsp(X, T, N, H)
```

Parameters

X :

Analyzed signal

T :

the time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

H :

frequency smoothing window, (default : Hamming(N/4)).

SP_reas :

Reassigned spectrogram

SP :

Spectrogram (not reassigned)

FIELD :

Field of reassignment vectors

Description

Computes the reassigned spectrogram, the spectrogram and the field of reassignment vectors

Examples

```
[SP,SP_reas] = Ctfrrsp(hilbert(sin(2*%pi*0.25*
(1:128)))+0.5*rand(1,128,'normal'));
grayplot(1:128,1:128,SP_reas');
```

See also

- Ctfrreas — Time Frequency Representation reassignment
- Cwindow — Creates a window of length N with a given shape.

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000
- Manuel Davy - 2000

Ctfrsp

Spectrogram Time Frequency distribution of a signal X

Calling Sequence

```
[SP,T,F,norm] = Ctfrsp(X)
[SP,T,F,norm] = Ctfrsp(X, T)
[SP,T,F,norm] = Ctfrsp(X, T, N)
[SP,T,F,norm] = Ctfrsp(X, T, N, H)
```

Parameters

X :

Analyzed signal

T :

the time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

H :

frequency smoothing window, (default : Hamming(N/4)).

SP :

Spectrogram

F :

Vector of frequency bins

NORM :

Vector of normalization applied at each time instant

Description

Computes the spectrogram of a signal X

Examples

```
x = hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal'));
[sp,T,F] = Ctfrsp(x,1:128,128);
grayplot(T,F,sp');
```

See also

- Ctfrrsp — Reassigned Spectrogram
- Ctfrstft — Short time Fourier transform
- Ctfrwv — Wigner-Ville time-frequency distribution
- Cwindow — Creates a window of length N with a given shape.

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000

Ctfrspwv

Smoothed Pseudo Wigner-Ville time-frequency distribution

Calling Sequence

```
[TFR,T,F] = Ctfrspwv(X)
[TFR,T,F] = Ctfrspwv(X, T)
[TFR,T,F] = Ctfrspwv(X, T, N)
[TFR,T,F] = Ctfrspwv(X, T, N, G)
[TFR,T,F] = Ctfrspwv(X, T, N, G, H)
```

Parameters

X :

Analyzed signal.

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

G :

time smoothing window, (default : Hamming(N/10)).

H :

frequency smoothing window, in the time-domain, (default : Hamming(N/4)).

TFR :

time-frequency representation.

F :

vector of normalized frequencies.

Description

computes the Smoothed Pseudo Wigner-Ville distribution of a signal X.

Examples

```
x = hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal'));
t = 1:128;
[tfr,T,F] = Ctfrspwv(x,t,128);
grayplot(T,F,tfr'); xlabel('time'); ylabel('frequency')
```

See also

- Ctfrwv — Wigner-Ville time-frequency distribution
- Ctfrpwv — Pseudo Wigner-Ville time-frequency distribution

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000
- Manuel Davy - 2000

stftb > Time-Frequency representations (in C) > Ctfrstft

Ctfrstft

Short time Fourier transform

Calling Sequence

```
[STFT,T,F] = Ctfrstft(X)
[STFT,T,F] = Ctfrstft(X, T)
[STFT,T,F] = Ctfrstft(X, T, N)
[STFT,T,F] = Ctfrstft(X, T, N,H)
```

Parameters

X :

signal for which the STFT is computed

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

H :

frequency smoothing window (default : Hamming(N/4)).

STFT :

Computed Short time Fourier Transform

F :

Vector of frequency bins

Example :

x = hilbert(sin(2*%pi*0.25*(1:

128))+0.5*rand(1,128,'normal'));

[stft,T,F] = Ctfrstft(x,1:

128,128);

See also

- Ctfrsp — Spectrogram Time Frequency distribution of a signal X
- Ctfrrsp — Reassigned Spectrogram
- Cwindow — Creates a window of length N with a given shape.

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000

Ctfrwv

Wigner-Ville time-frequency distribution

Calling Sequence

```
[TFR,T,F] = Ctfrwv(X)
[TFR,T,F] = Ctfrwv(X, T)
[TFR,T,F] = Ctfrwv(X, T, N)
```

Parameters

X :

Analyzed signal.

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

TFR :

time-frequency representation.

F :

vector of normalized frequencies.

Description

computes the Wigner-Ville distribution of a signal X.

Examples

```
x = hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal'));
t = 1:128;
[tfr,T,F] = Ctfrwv(x,t,128);
grayplot(T,F,tfr');
xlabel time
ylabel frequency
```

See also

- Ctfrpwv — Pseudo Wigner-Ville time-frequency distribution
- Ctfrspwv — Smoothed Pseudo Wigner-Ville time-frequency distribution

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000

Ctfrzam

Zao-Atlas-Marks time-frequency distribution

Calling Sequence

```
[TFR,T,F] = Ctfrzam(X)
[TFR,T,F] = Ctfrzam(X, T)
[TFR,T,F] = Ctfrzam(X, T, N)
[TFR,T,F] = Ctfrzam(X, T, N, G)
[TFR,T,F] = Ctfrzam(X, T, N, G, H)
```

Parameters

X :

Analyzed signal.

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

G :

time smoothing window, (default : Hamming(N/10)).

H :

frequency smoothing window (default : Hamming(N/4)).

TFR :

time-frequency representation.

F :

vector of normalized frequencies.

Description

computes the Zao-Atlas-Marks distribution of asignal X.

Examples

```
x = hilbert(sin(2*%pi*0.25*
(1:128))+0.5*rand(1,128,'normal'));
g = Cwindow(9,'Hamming');
h = Cwindow(27,'Hamming');
```

```
t = 1:128;
[tfr,T,F] = Ctfrzam(x,t,128,g,h);
grayplot(T,F,tfr');
xlabel time
ylabel frequency
```

Authors

- H. Nahrstaedt - Sep 2010
- Emmanuel Roy - 2000
- Manuel Davy - 2000

Ambiguity Functions

- ambifunb — Narrow-band ambiguity function
- ambifuwb — Wide-band ambiguity function

ambifunb

Narrow-band ambiguity function

Calling Sequence

```
[NAF,TAU,XI] = ambifunb(X)
[NAF,TAU,XI] = ambifunb(X, TAU)
[NAF,TAU,XI] = ambifunb(X, TAU, N)
[NAF,TAU,XI] = ambifunb(X, TAU, N, TRACE)
[NAF,TAU,XI] = ambifunb(...,'plot')
```

Parameters

X:

a real or complex Nx elements real (auto-Untenberger) or a Nx by 2 array signal (cross-Untenberger).

TAU :

real vector of lag values (default is -Nx/2:Nx/2).

N :

a positive integer: the number of frequency bins (default is Nx).

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if input contains the string 'plot', the output values will be plotted

NAF :

doppler-lag representation, with the doppler bins stored in the rows and the time-lags stored in the columns. When called without output arguments, ambifunb displays the squared modulus of the ambiguity function by means of contour.

XI:

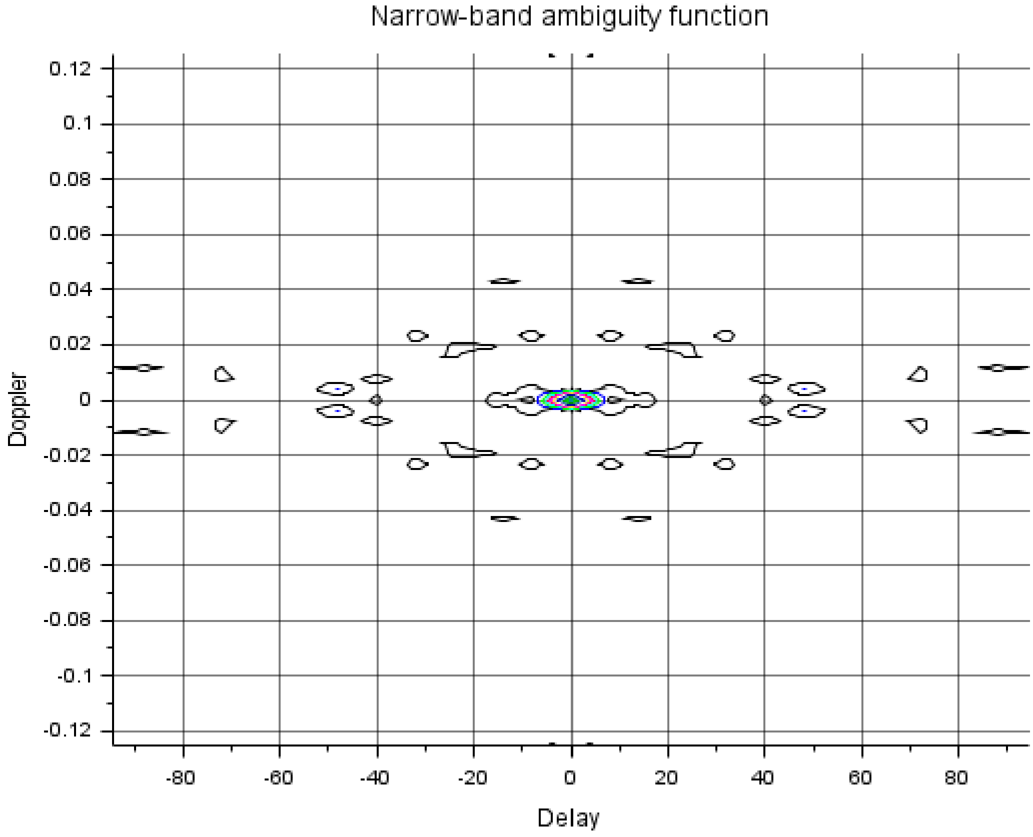
row vector of doppler values.

Description

ambifunb computes the narrow-band ambiguity function of a signal X, or the cross-ambiguity function between two signals.

Examples

```
sig = anabpsk(256,8);
ambifunb(sig,'plot');
```



See also

- ambifuwb — Wide-band ambiguity function

Authors

- H. Nahrstaedt - Aug 2010
- O. Lemoine
- F. Auger - August 1995.

ambifuwb

Wide-band ambiguity function

Calling Sequence

```
[WAF,TAU,THETA] = ambifuwb(X)
[WAF,TAU,THETA] = ambifuwb(X, FMIN)
[WAF,TAU,THETA] = ambifuwb(X, FMIN, FMAX)
[WAF,TAU,THETA] = ambifuwb(X, FMIN, FMAX, N)
[WAF,TAU,THETA] = ambifuwb(X, FMIN, FMAX, N, TRACE)
[WAF,TAU,THETA] = ambifuwb(...,'plot')
```

Parameters

X :

real vector of length Nx : the signal (in time) to be analyzed.

FMIN:

a positive scalar in]0 0.5], the normalized lower frequency bound in (Hz) of the analyzed signal. When unspecified, you have to enter it at the command line from the plot of the spectrum.

FMAX :

a positive scalar in]0 0.5], the normalized upper frequency bound (in Hz) of the analyzed signal. When unspecified, you have to enter it at the command line from the plot of the spectrum.

N :

a positive integer: number of analyzed voices. When unspecified, you have to enter it at the command line from the plot of the spectrum.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

AF :

matrix containing the coefficients of the ambiguity function. X-coordinate corresponds to the dual variable of scale parameter ; Y-coordinate corresponds to time delay, dual variable of frequency.

'plot' :

When called with the additional argument 'plot', ambifuwb displays the squared modulus of the ambiguity function by means of contour.

TAU :

X-coordinate corresponding to time delay

THETA :

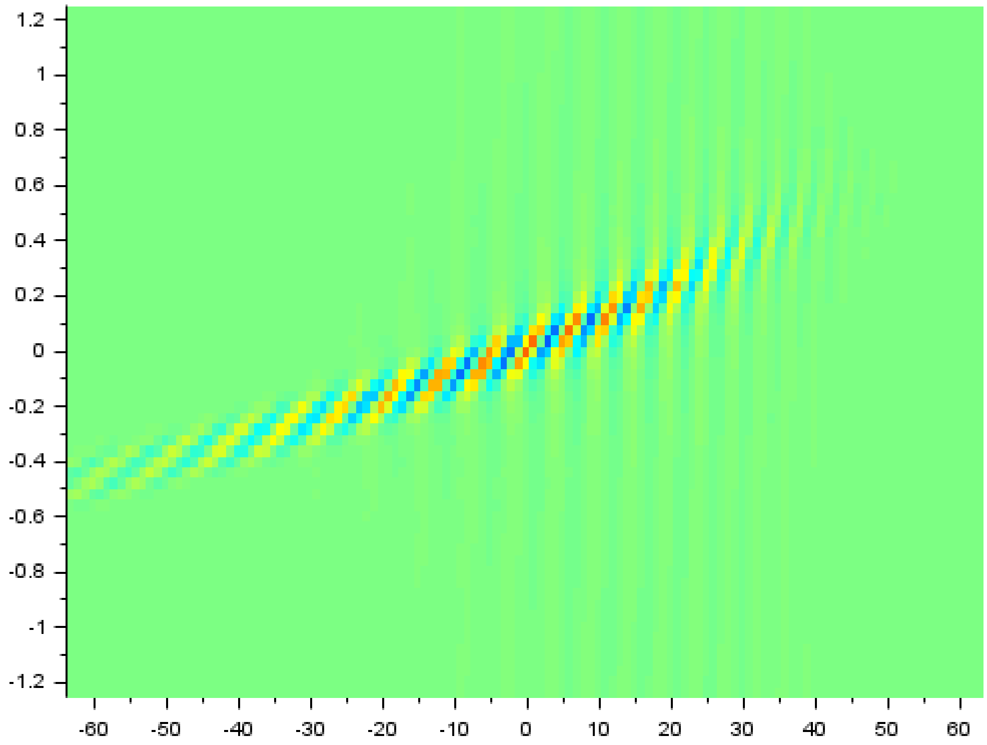
Y-coordinate corresponding to the log(scale) variable

Description

ambifuwb calculates the asymetric wide-band ambiguity function.

Examples

```
sig = altes(128,0.1,0.45);
[TFR,T,F] = ambifuwb(sig,0.1,0.35,64);
clf; gcf().color_map = jetcolormap(128);
grayplot(T,F,TFR');
```



See also

- ambifunb — Narrow-band ambiguity function

Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalvez - December 1995, O. Lemoine - August 1996.

Choice of the Instantaneous Amplitude

- amexpo1s — Generate one-sided exponential amplitude modulation
- amexpo2s — Generate bilateral exponential amplitude modulation
- amgauss — Generate gaussian amplitude modulation
- amrect — Generate rectangular amplitude modulation
- amtriang — Generate triangular amplitude modulation

amexpo1s

Generate one-sided exponential amplitude modulation

Calling Sequence

```
Y = amexpo1s(N)
Y = amexpo1s(N, T0)
Y = amexpo1s(N, T0, T)
```

Parameters

N :

a positive integer value: the number of points.

T0 :

a positive scalar: the arrival time of the exponential (default : N/2).

T :

a positive scalar: the time spreading (default : 2*sqrt(N)).

Y :

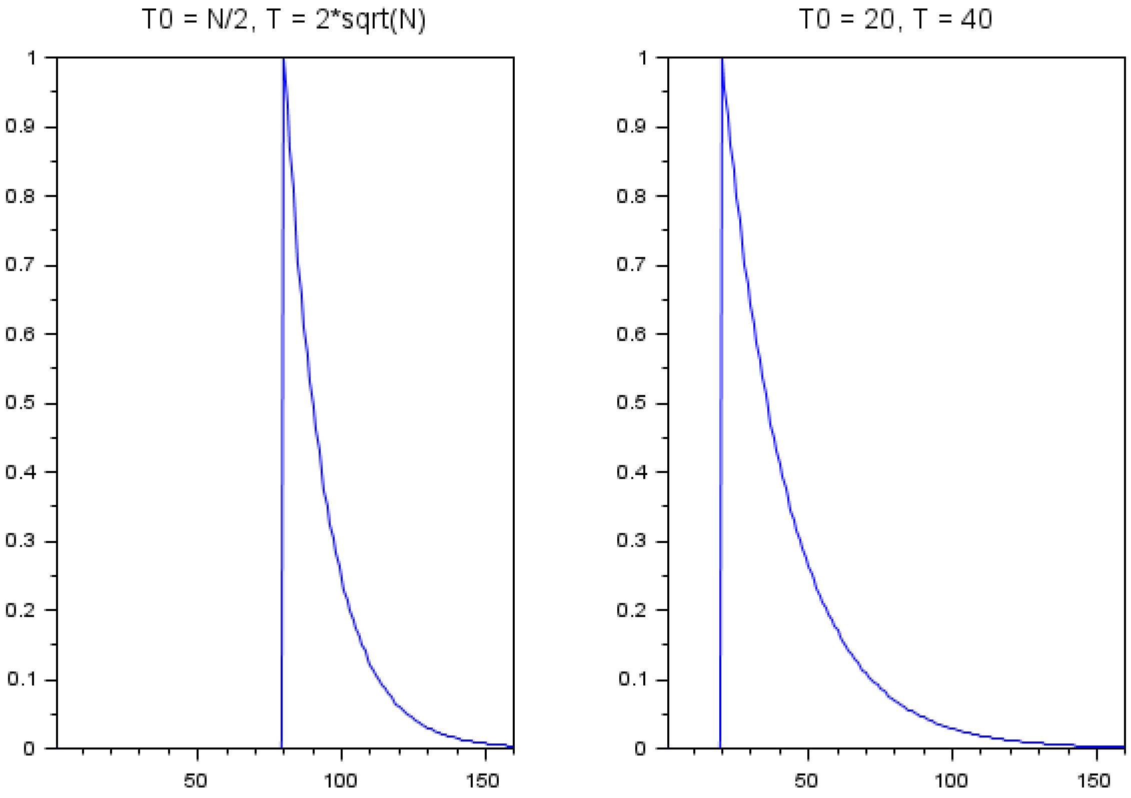
a real row vector: the signal.

Description

amexpo1s generates a one-sided exponential amplitude modulation centered on a time T0, and with a spread proportional to T. This modulation is scaled such that Y(T0)=1.

Examples

```
z = amexpo1s(160);
subplot(121);plot(z); xtitle "T0 = N/2, T = 2*sqrt(N)"
z = amexpo1s(160,20,40);
subplot(122); plot(z); xtitle "T0 = 20, T = 40"
```



See also

- amexpo2s — Generate bilateral exponential amplitude modulation
- amgauss — Generate gaussian amplitude modulation
- amrect — Generate rectangular amplitude modulation
- amtriang — Generate triangular amplitude modulation

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, July 1995.
- Copyright (c) 1996 by CNRS (France).

amexpo2s

Generate bilateral exponential amplitude modulation

Calling Sequence

```
Y = amexpo2s(N)
Y = amexpo2s(N, T0)
Y = amexpo2s(N, T0, T)
```

Parameters

N :

a positive integer value: the number of points.

T0 :

a positive scalar: the arrival time of the exponential (default : N/2).

T :

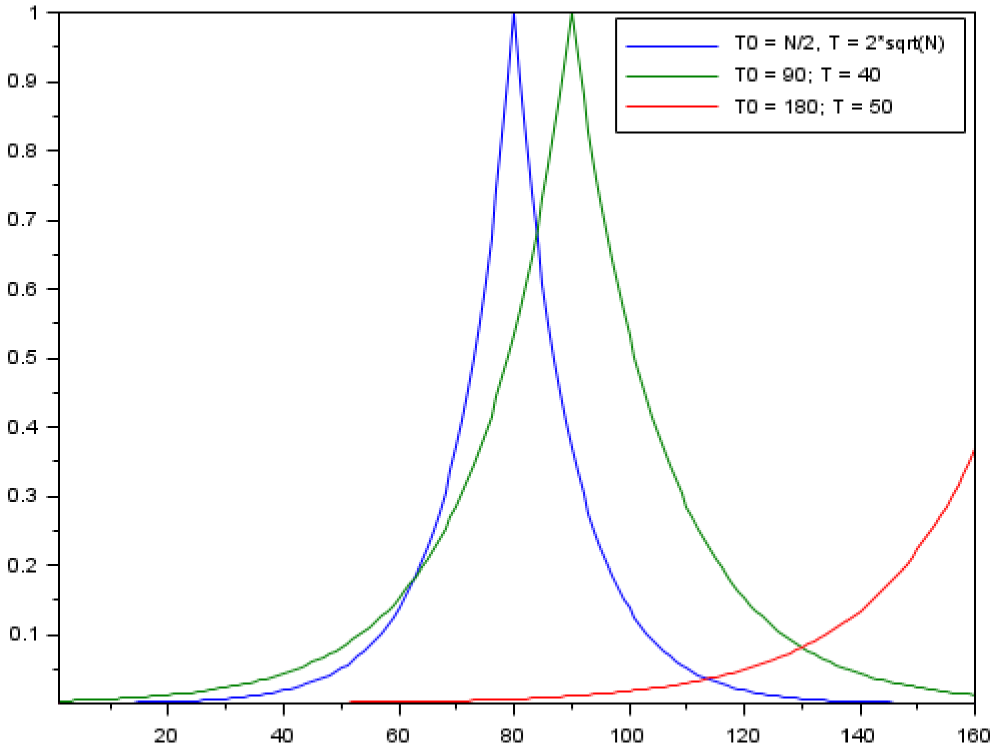
a positive scalar: the time spreading (default : 2*sqrt(N)).

Y :

a real row vector: the signal.

Examples

```
z1 = amexpo2s(160);
z2 = amexpo2s(160,90,40);
z3 = amexpo2s(160,180,50);
clf; plot([z1 z2 z3])
legend(["T0 = N/2, T = 2*sqrt(N)";"T0 = 90; T = 40";"T0 =
180; T = 50"]);
```



See also

- amexpo1s — Generate one-sided exponential amplitude modulation
- amgauss — Generate gaussian amplitude modulation
- amrect — Generate rectangular amplitude modulation
- amtriang — Generate triangular amplitude modulation

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, July 1995.
- Copyright (c) 1996 by CNRS (France).

amgauss

Generate gaussian amplitude modulation

Calling Sequence

```
Y = amgauss(N)
Y = amgauss(N, T0)
Y = amgauss(N, T0, T)
```

Parameters

N :

a positive integer value: the number of points.

T0 :

a positive scalar: time center (default : N/2).

T :

a positive scalar: the time spreading (default : 2*sqrt(N)).

Y :

a real row vector: the signal.

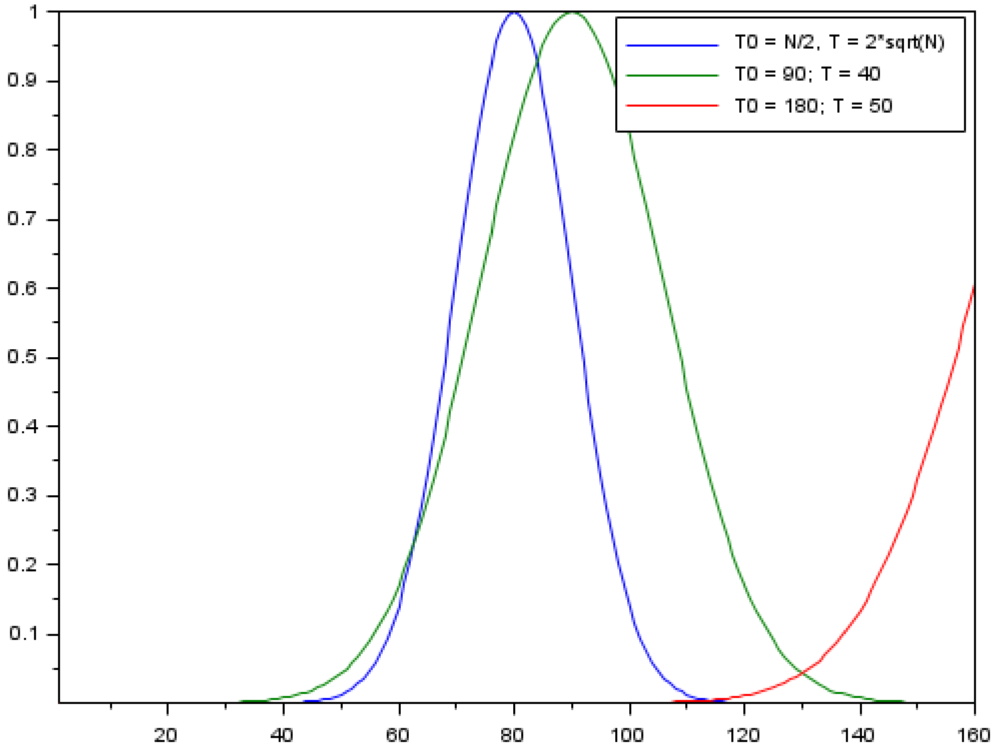
Description

amgauss generates a gaussian amplitude modulation centered on a time T0, and with a spread proportional to T. This modulation is scaled such that Y(T0)=1 and Y(T0+T/2) and Y(T0-T/2) are approximately equal to 0.5 .

Examples

With default parameters

```
z1 = amgauss(160);
z2 = amgauss(160,90,40);
z3 = amgauss(160,180,50);
clf; plot([z1 z2 z3])
legend(["T0 = N/2, T = 2*sqrt(N)";"T0 = 90; T = 40";"T0 =
180; T = 50"]);
```



See also

- amexpo1s — Generate one-sided exponential amplitude modulation
- amexpo2s — Generate bilateral exponential amplitude modulation
- amrect — Generate rectangular amplitude modulation
- amtriang — Generate triangular amplitude modulation

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, July 1995.
- Copyright (c) 1996 by CNRS (France).

amrect

Generate rectangular amplitude modulation

Calling Sequence

```
Y = amrect(N)
Y = amrect(N, T0)
Y = amrect(N, T0, T)
```

Parameters

N :

a positive integer value: the number of points.

T0 :

a positive scalar: time center (default : N/2).

T :

a positive scalar: the time spreading (default : 2*sqrt(N)).

Y :

a real row vector: the signal.

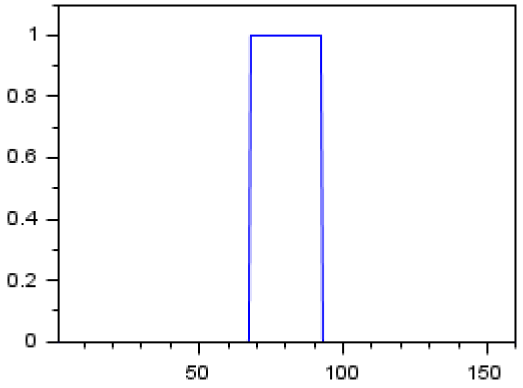
Description

amrect generates a rectangular amplitude modulation centered on a time T0, and with a spread proportional to T. This modulation is scaled such that Y(T0)=1.

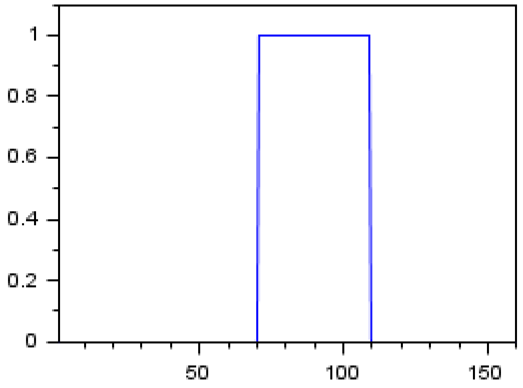
Examples

```
clf
z = amrect(160);
subplot(221); plot(z); gca().data_bounds(2,2)=1.1;
xtitle("T0 = N/2, T = 2*sqrt(N)");
z = amrect(160,90,40);
subplot(222); plot(z); gca().data_bounds(2,2)=1.1;
xtitle "T0 = 90; T = 40"
z = amrect(160,180,70);
subplot(223); plot(z); gca().data_bounds(2,1:2)=[180
1.1];
xtitle "T0 = 180; T = 50"
```

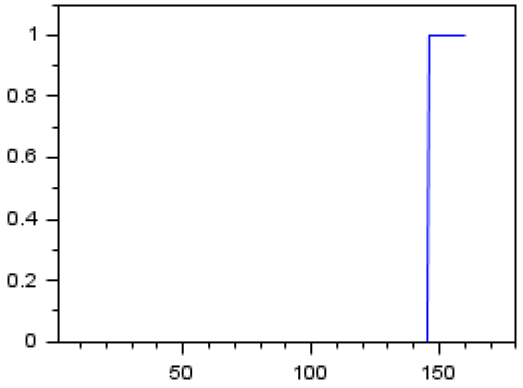
T0 = N/2, T = 2*sqrt(N)



T0 = 90; T = 40



T0 = 180; T = 50



See also

- amexpo1s — Generate one-sided exponential amplitude modulation
- amexpo2s — Generate bilateral exponential amplitude modulation
- amgauss — Generate gaussian amplitude modulation
- amtriang — Generate triangular amplitude modulation

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, July 1995.
- Copyright (c) 1996 by CNRS (France).

amtriang

Generate triangular amplitude modulation

Calling Sequence

```
Y = amtriang(N)
Y = amtriang(N, T0)
Y = amtriang(N, T0, T)
```

Parameters

N :

a positive integer value: the number of points.

T0 :

a positive scalar: time center (default : N/2).

T :

a positive scalar: the time spreading (default : 2*sqrt(N)).

Y :

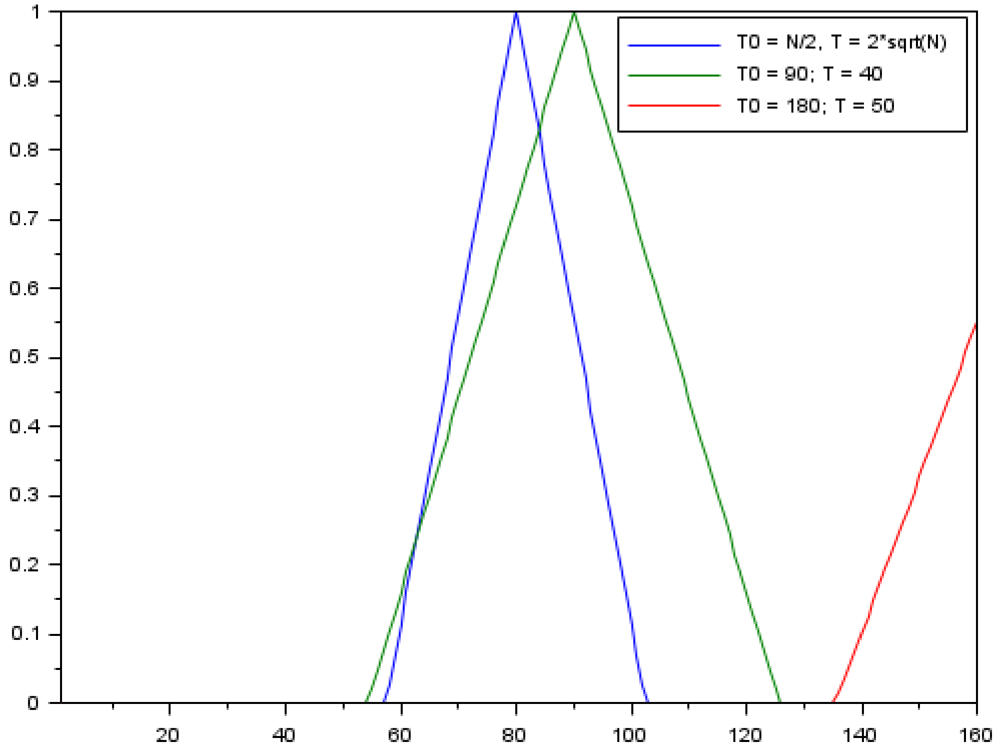
a real row vector: the signal.

Description

amtriang generates a triangular amplitude modulation centered on a time T0, and with a spread proportional to T. This modulation is scaled such that Y(T0)=1.

Examples

```
z1 = amtriang(160);
z2 = amtriang(160,90,40);
z3 = amtriang(160,180,50);
clf; plot([z1 z2 z3])
legend(["T0 = N/2, T = 2*sqrt(N)";"T0 = 90; T = 40";"T0 =
180; T = 50"]);
```



See also

- amexpo1s — Generate one-sided exponential amplitude modulation
- amexpo2s — Generate bilateral exponential amplitude modulation
- amgauss — Generate gaussian amplitude modulation
- amrect — Generate rectangular amplitude modulation

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, July 1995.
- Copyright (c) 1996 by CNRS (France).

Bilinear Time-Frequency Processing in the Affine Class

- istfraff — returns true is method is the name of an affine time frequency representation.
- lambdak — Evaluate lambda function for Affine Wigner distribution.
- tfrbert — Unitary Bertrand time-frequency distribution.
- tfrdfla — D-Flandrin time-frequency distribution.
- tfrscalo — Scalogram, for Morlet or Mexican hat wavelet.
- tfrspaw — Smoothed Pseudo Affine Wigner time-frequency distributions.
- tfrunter — Unterberger time-frequency distribution.

istfraff

returns true is method is the name of an affine time frequency representation.

Calling Sequence

```
flag = istfr2(method)
```

Parameters

method :

A character string.

flg :

An integer with 0 and 1 as possible values.

Description

This function returns 1 if method has one of the following values (The case does not matter). 'TFRASPW', 'TFRSCALO', 'TFRDFLA', 'TFRSPAW', 'TFRUNTER', 'TFRBERT', 'TFRSPBK'.

See also

- istfr1 — returns true is method is a time frequency representation of type 1 (frequencies >0 or <0)
- istfr2 — returns true is method is a time frequency representation of type 2 (only frequencies > 0)

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, may 98

lambdak

Evaluate lambda function for Affine Wigner distribution.

Calling Sequence

```
Y = lambdak(U,K)
```

Parameters

U :

real vector

K :

real scalar

Y :

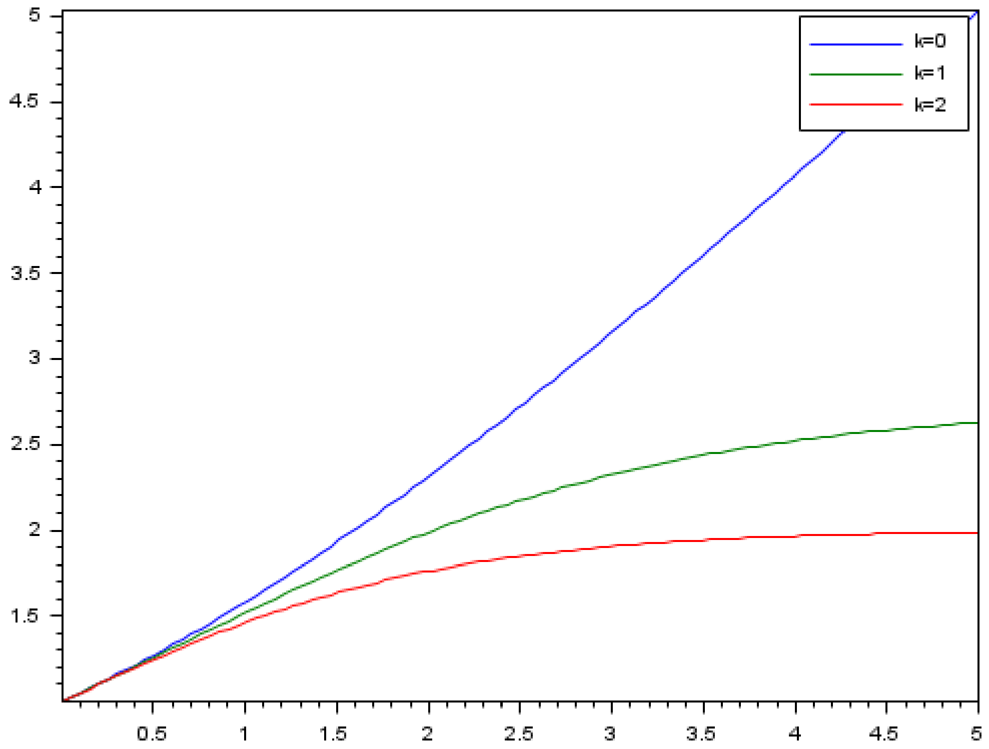
value of LAMBDAD at point(s) U

Description

lambdak evaluates the parametrization lambda function involved in the affine smoothed pseudo Bertrand distribution. `lambdak(U,0) = -U/(exp(-U)-1) for K = 0 lambdak(U,1) = exp(1+U exp(-U)/(exp(-U)-1)) for K = 1 lambdak(U,K) = (K (exp(-U)-1)/(exp(-KU)-1))^(1/(K-1)) otherwise`

Examples

```
u=linspace(0.001,5,100)';
clf;plot(u,[lambdak(u,0),lambdak(u,1),lambdak(u,2)])
legend("k="+["0","1","2"]);
```



Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves, October 95

tfrbert

Unitary Bertrand time-frequency distribution.

Calling Sequence

```
[TFR,T,F] = tfrbert(X)
[TFR,T,F] = tfrbert(X, T)
[TFR,T,F] = tfrbert(X, T, FMIN,FMAX)
[TFR,T,F] = tfrbert(X, T, FMIN,FMAX, N)
[TFR,T,F] = tfrbert(X, T, FMIN,FMAX, N, TRACE)
[TFR,T,F] = tfrbert(...,'plot')
```

Parameters

X :

It can be:

a vector of size Nx: the signal (in time)to be analyzed .

or a 2 by Nx matrix for the cross-unitary Bertrand distribution)

T :

a real vector: the time instant(s) on which the TFR is evaluated (default : 1:Nx).

FMIN,FMAX :

respectively lower and upper frequency bounds of the analyzed signal. These parameters fix the equivalent frequency bandwidth (expressed in Hz). When unspecified, you have to enter them at the command line from the plot of the spectrum. FMIN and FMAX must be >0 and <=0.5.

N :

number of analyzed voices (default : automatically determined).

TRACE :

if nonzero, the progression of the algorithm is shown (default : 0).

'plot':

if one input parameter is 'plot', tfrbert runs tfrqview. and TFR will be plotted

TFR :

time-frequency matrix containing the coefficients of the distribution (x-coordinate corresponds to uniformly sampled time, and y-coordinate corresponds to a geometrically sampled frequency). First row of TFR corresponds to the lowest frequency.

F :

vector of normalized frequencies (geometrically sampled from FMIN to FMAX).

Description

tfrbert generates the auto- or cross- unitary Bertrand distribution.

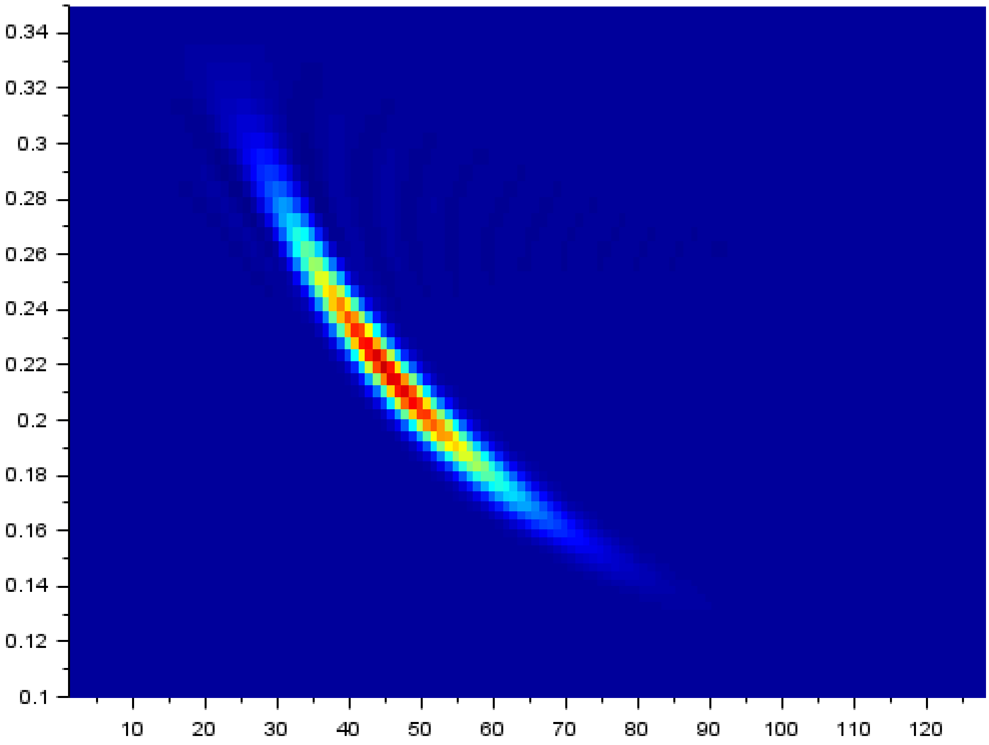
Examples

Interactive use

```
sig = altes(64,0.1,0.45); tfrbert(sig,'plot');
```

Non interactive use

```
N = 128;
sig = altes(N,0.1,0.45);
T = 1:N;
[tfr,t,f] = tfrbert(sig,T,0.1,0.35,64);
clf; gcf().color_map = jetcolormap(128);
grayplot(t,f,tfr');
```



See also

- tfrscalo — Scalogram, for Morlet or Mexican hat wavelet.
- tfrunter — Unterberger time-frequency distribution.
- tfrdfla — D-Flandrin time-frequency distribution.
- tfrspaw — Smoothed Pseudo Affine Wigner time-frequency distributions.

Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves, October 95 - O. Lemoine, June 1996.

tfrdfla

D-Flandrin time-frequency distribution.

Calling Sequence

```
[TFR,T,F] = tfrdfla(X)
[TFR,T,F] = tfrdfla(X, T)
[TFR,T,F] = tfrdfla(X, T, FMIN,FMAX)
[TFR,T,F] = tfrdfla(X, T, FMIN,FMAX, N)
[TFR,T,F] = tfrdfla(X, T, FMIN,FMAX, N, TRACE)
[TFR,T,F] = tfrdfla(...,'plot')
```

Parameters

X :

It can be:

a vector of size Nx: the signal (in time)to be analyzed .

or a 2 by Nx matrix for the cross-unitary Bertrand distribution)

T :

a real vector: the time instant(s) on which the TFR is evaluated (default : 1:Nx).

FMIN,FMAX :

respectively lower and upper frequency bounds of the analyzed signal. These parameters fix the equivalent frequency bandwidth (expressed in Hz). When unspecified, you have to enter them at the command line from the plot of the spectrum. FMIN and FMAX must be >0 and <=0.5.

N :

number of analyzed voices (default : automatically determined).

TRACE :

if nonzero, the progression of the algorithm is shown (default : 0).

'plot':

if one input parameter is 'plot', tfrbert runs tfrqview. and TFR will be plotted

TFR :

time-frequency matrix containing the coefficients of the distribution (x-coordinate corresponds to uniformly sampled time, and y-coordinate corresponds to a geometrically sampled frequency). First row of TFR corresponds to the lowest frequency.

F :

vector of normalized frequencies (geometrically sampled from FMIN to FMAX).

Description

tfrdfla generates the auto- or cross- D-Flandrin distribution.

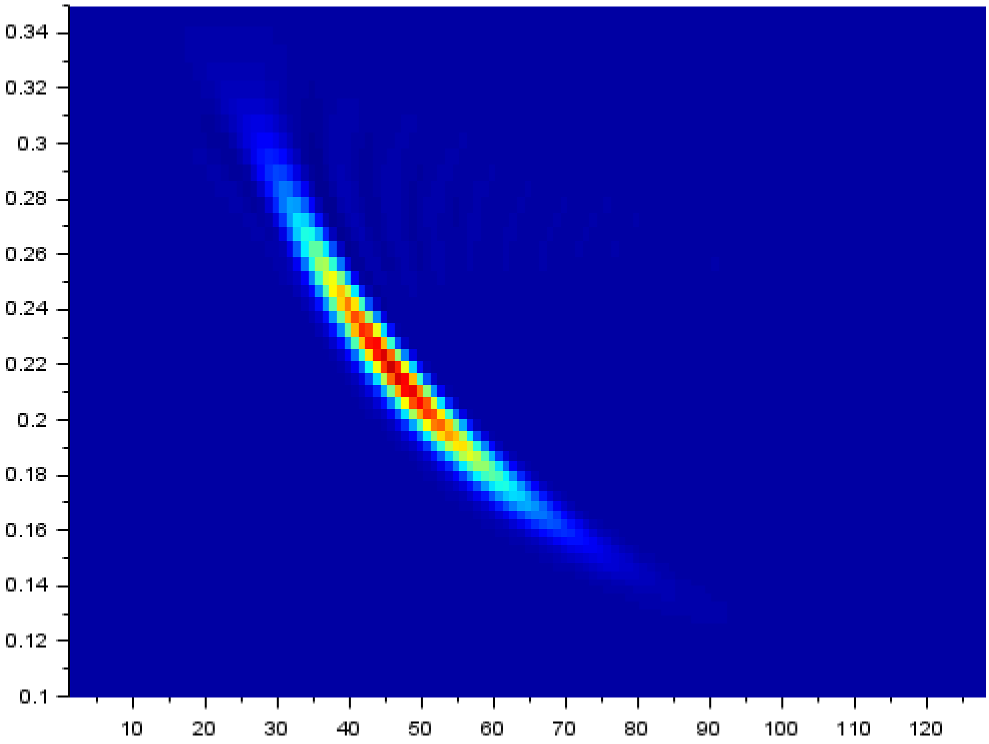
Examples

Interactive use

```
sig = altes(64,0.1,0.45);
tfrdfla(sig,'plot');
```

Non interactive use

```
N = 128;
sig = altes(N,0.1,0.45);
T = 1:N;
[tfr,t,f] = tfrdfla(sig,1:N,0.1,0.35);
clf;gcf().color_map = jetcolormap(128);
grayplot(t,f,tfr');
```



Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves, October 95 - O. Lemoine, June 1996.

tfrscalo

Scalogram, for Morlet or Mexican hat wavelet.

Calling Sequence

```
[TFR,T,F,WT] = tfrscalo(X)
[TFR,T,F,WT] = tfrscalo(X, T)
[TFR,T,F,WT] = tfrscalo(X, T, WAVE, FMIN,FMAX)
[TFR,T,F,WT] = tfrscalo(X, T, WAVE, FMIN,FMAX, N)
[TFR,T,F,WT] = tfrscalo(X, T, WAVE, FMIN,FMAX, N, TRACE)
[TFR,T,F,WT] = tfrscalo(...,'plot')
```

Description

tfrscalo computes the scalogram (squared magnitude of a continuous wavelet transform).

Parameters

X :

signal (in time) to be analyzed (Nx=length(X)). Its analytic version is used (z=hilbert(real(X))).

T :

a real Nt vector with elements in [1 Nx] : time instant(s) on wich the tfr is computed. (default: 1:NX).

WAVE :

half length of the Morlet analyzing wavelet at coarsest scale. If WAVE = 0, the Mexican hat is used. WAVE can also be a vector containing the time samples of any bandpass function, at any scale. (default : sqrt(Nx)).

FMIN:

a positive scalar in]0 0.5], the normalized lower frequency bound in (Hz) of the analyzed signal. When unspecified, you have to enter it at the command line from the plot of the spectrum.

FMAX :

a positive scalar in]0 0.5], the normalized upper frequency bound (in Hz) of the analyzed signal. When unspecified, you have to enter it at the command line from the plot of the spectrum.

N :

positive integer: number of analyzed voices. When unspecified, you have to enter it at the command line from the plot of the spectrum.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

N by Nt real array: the time-frequency matrix containing the coefficients of the decomposition (abscissa correspond to uniformly sampled time, and ordinates correspond to a geometrically sampled frequency). First row of TFR corresponds to the lowest frequency.

F :

vector of normalized frequencies (geometrically sampled from FMIN to FMAX).

WT :

N by Nt complex array containing the corresponding wavelet transform. The scalogram TFR is the square modulus of WT multiplied by a constant factor.

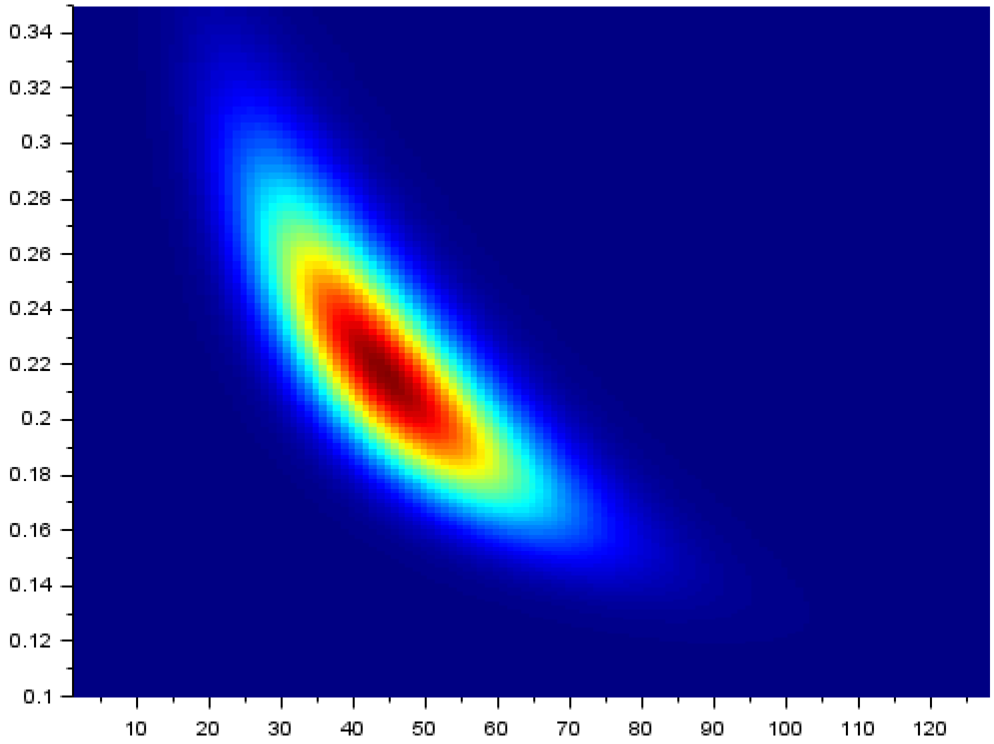
Examples

Interactive use

```
N = 64;
sig = altes(N,0.1,0.45);
tfrscalo(sig,'plot');
```

Non-interactive use

```
N = 128;
sig = altes(N,0.1,0.45);
T = 1:N;
wave = sqrt(N);
[fmin, fmax] = (0.1, 0.35);
[tfr,t,f,wt] = tfrscalo(sig,T,wave,fmin,fmax,128);
clf; gcf().color_map = jetcolormap(128);
grayplot(t,f,tfr');
```



Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves, October 1995 - O. Lemoine, June 1996.
- Copyright (c) 1995 Rice University - CNRS 1996.

tfrspaw

Smoothed Pseudo Affine Wigner time-frequency distributions.

Calling Sequence

```
[TFR,T,F] = tfrspaw(X)
[TFR,T,F] = tfrspaw(X, T)
[TFR,T,F] = tfrspaw(X, T, K)
[TFR,T,F] = tfrspaw(X, T, K, NH0)
[TFR,T,F] = tfrspaw(X, T, K, NH0, NG0)
[TFR,T,F] = tfrspaw(X, T, K, NH0, NG0, FMIN,FMAX)
[TFR,T,F] = tfrspaw(X, T, K, NH0, NG0, FMIN,FMAX, N)
[TFR,T,F] = tfrspaw(X, T, K, NH0, NG0, FMIN,FMAX, N,
TRACE)
[TFR,T,F] = tfrspaw(...,'plot')
```

Parameters

X :

A Nx elements vector (signal) or a Nx by 2 array signal (cross-Smoothed Pseudo Affine Wigner distribution.).

T:

a real Nt vector with elements in [1 Nx] : time instant(s) on which the TFR is evaluated (default: 1:NX).

K :

label of the K-Bertrand distribution. The distribution with parameterization function `lambdak(u,K) = (K(exp(-u)-1)/(exp(-Ku)-1))^(1/(K-1))` is computed (default is 0).

- K=-1: Smoothed pseudo (active) Unterberger distribution
- K=0: Smoothed pseudo Bertrand distribution
- K=1/2: Smoothed pseudo D-Flandrin distribution
- K=2: Affine smoothed pseudo Wigner-Ville distribution.

NH0 :

half length of the analyzing wavelet at coarsest scale. A Morlet wavelet is used. NH0 controles the frequency smoothing of the smoothed pseudo Affine Wigner distribution. (default is sqrt(Nx)).

NG0 :

half length of the time smoothing window. If NG0 is set to zero the time smoothing window is a rectangular one and the length is automatically determined.

FMIN:

a positive scalar in]0 0.5], the normalized lower frequency bound in (Hz) of the analyzed signal. When unspecified, you have to enter it at the command line from the plot of the spectrum.

FMAX :

a positive scalar in]0 0.5], the normalized upper frequency bound (in Hz) of the analyzed signal. When unspecified, you have to enter it at the command line from the plot of the spectrum.

N :

positive integer: number of analyzed voices. When unspecified, you have to enter it at the command line from the plot of the spectrum.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

N by Nt real array: the time-frequency matrix containing the coefficients of the decomposition (abscissa correspond to uniformly sampled time, and ordinates correspond to a geometrically sampled frequency). First row of TFR corresponds to the lowest frequency.

F :

vector of normalized frequencies (geometrically sampled from FMIN to FMAX).

Description

tfrspaw generates the auto- or cross- Smoothed Pseudo Affine Wigner distributions.

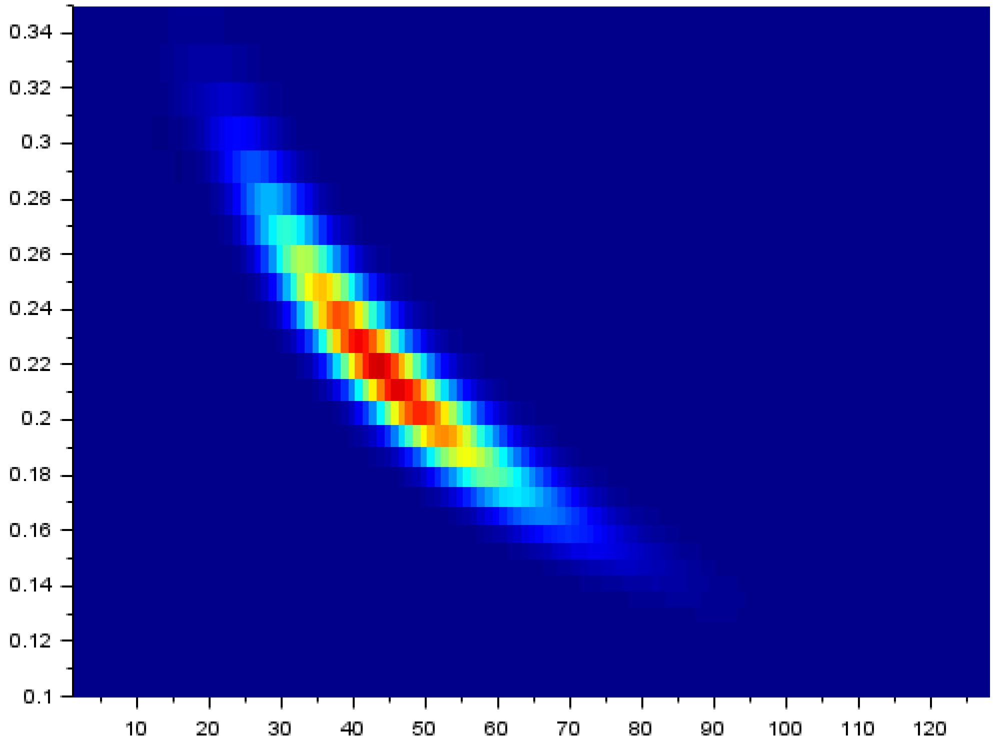
Examples

Interactive use

```
N = 128;
sig = altes(N,0.1,0.45);
tfrspaw(sig,"plot");
```

Non interactive use

```
N = 128;
sig = altes(N,0.1,0.45);
T = 1:N;
K = 0;

[tfr,t,f] = tfrspaw(sig,1:N,0,2*sqrt(N),0,0.1,0.35,32);
clf;gcf().color_map =  jetcolormap(128);
grayplot(t,f,tfr');
```



Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves, October 95
- O. Lemoine, June 1996.
- Copyright (c) 1995 Rice University - CNRS (France) 1996.

tfrunter

Unterberger time-frequency distribution.

Calling Sequence

```
[TFR,T,F] = tfrunter(X,)
[TFR,T,F] = tfrunter(X, T)
[TFR,T,F] = tfrunter(X, T, FORM)
[TFR,T,F] = tfrunter(X, T, FORM, FMIN,FMAX)
[TFR,T,F] = tfrunter(X, T, FORM, FMIN,FMAX, N)
[TFR,T,F] = tfrunter(X, T, FORM, FMIN,FMAX, N, TRACE)
[TFR,T,F] = tfrunter(...,'plot')
```

Parameters

X :

a real or complex Nx elements real (auto-Untenberger) or a Nx by 2 array signal (cross-Untenberger).

T:

a real Nt vector with elements in [1 Nx] : time instant(s) on which the TFR is evaluated (default: 1:NX).

FORM :

a character with possible values 'A' for active, 'P' for passive Unterberger distribution.(default : 'A'). Cas do not matter.

FMIN:

a positive scalar in]0 0.5], the normalized lower frequency bound in (Hz) of the analyzed signal. When unspecified, you have to enter it at the command line from the plot of the spectrum.

FMAX :

a positive scalar in]0 0.5], the normalized upper frequency bound (in Hz) of the analyzed signal. When unspecified, you have to enter it at the command line from the plot of the spectrum.

N :

a positive integer: number of analyzed voices. When unspecified, you have to enter it at the command line.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

F :

A N vector of normalized frequencies.

Description

tfrunter generates the auto- or cross-Unterberger distribution (active or passive form).

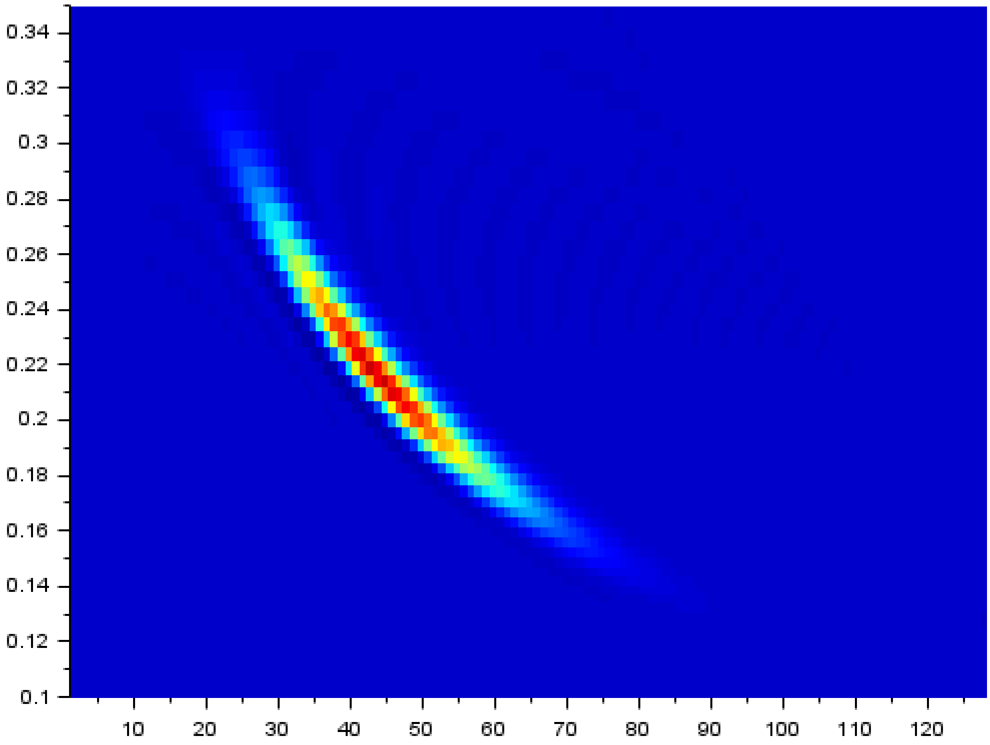
Examples

Interactive use

```
N = 128;
sig = altes(N,0.1,0.45);
tfrunter(sig);
```

Non-interactive use

```
N = 128;
sig = altes(N,0.1,0.45);
[tfr,t,f] = tfrunter(sig,1:N,"A",0.1,0.35,56);
gcf().color_map = jetcolormap(128);
grayplot(t,f,tfr');
```



Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves, November 95 - O. Lemoine, June 1996.
- Copyright (c) 1995 Rice University - CNRS (France) 1996.

Bilinear Time-Frequency Processing in the Cohen's Class

- tfrbj — Born-Jordan time-frequency distribution.
- tfrbud — Butterworth time-frequency distribution
- tfrcw — Choi-Williams time-frequency distribution
- tfrgrd — Generalized rectangular time-frequency distribution
- tfrmh — Margenau-Hill time-frequency distribution
- tfrmhs — Margenau-Hill-Spectrogram time-frequency distribution
- tfrmmce — Minimum mean cross-entropy combination of spectrograms
- tfrpage — Page time-frequency distribution
- tfrpmh — Pseudo Margenau-Hill time-frequency distribution
- tfrppage — Pseudo Page time-frequency distribution
- tfrpwv — Pseudo Wigner-Ville time-frequency distribution
- tfrri — Rihaczek time-frequency distribution.
- tfrridb — Reduced Interference Distribution with Bessel kernel.
- tfrridbn — Reduced Interference Distribution with a binomial kernel.
- tfrridh — Reduced Interference Distribution with Hanning kernel.
- tfrridt — Reduced Interference Distribution with triangular kernel.
- tfrsp — Spectrogram time-frequency distribution.
- tfrspbk — Smoothed Pseudo K-Bertrand time-frequency distribution.
- tfrspwv — Smoothed Pseudo Wigner-Ville time-frequency distribution.
- tfrwv — Wigner-Ville time-frequency distribution
- tfrzam — Zao-Atlas-Marks time-frequency distribution

tfrbj

Born-Jordan time-frequency distribution.

Calling Sequence

```
[TFR,T,F] = tfrbj(X)
[TFR,T,F] = tfrbj(X, T)
[TFR,T,F] = tfrbj(X, T, N)
[TFR,T,F] = tfrbj(X, T, N, G)
[TFR,T,F] = tfrbj(X, T, N, G, H)
[TFR,T,F] = tfrbj(X, T, N, G, H, TRACE)
[TFR,T,F] = tfrbj(...,'plot')
```

Parameters

X :

A Nx elements vector (auto-BJ) or a Nx by 2 array signal (cross-BJ).

T:

a real Nt vector with elements in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:NX). For faster computation N should be a power of 2.

G:

a real vector with odd length: the time smoothing window, (default :Hamming(N/10)).

It will be normalized such as the middle point equals 1 to preserve signal energy .

H :

real vector with odd length: the frequency smoothing window,(default: Hamming(N/4)).

It will be normalized such as the middle point equals 1 to preserve signal energy.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

F :

A N vector of normalized frequencies.

Description

tfrbj computes the Born-Jordan distribution of a discrete-time signal X, or the cross Born-Jordan representation between two signals.

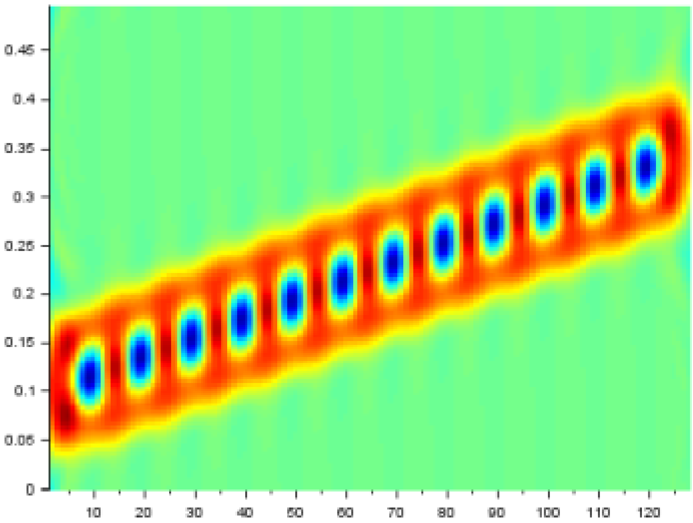
Examples

Interactive use

```
N = 128;
sig = fmlin(N,0.05,0.3) + fmlin(N,0.15,0.4);
g = window("kr",9,3*%pi);
h = window("kr",27,3*%pi);
t = 1:N;
tfrbj(sig,t,N,g',h',1,'plot');
```

Non interactive use

```
N = 128;
sig = fmlin(N,0.05,0.3) + fmlin(N,0.15,0.4);
g = window("kr",9,3*%pi);
h = window("kr",27,3*%pi);
t = 1:N;
[TFR,T,F] = tfrbj(sig,t,N,g,h);
clf; gcf().color_map = jetcolormap(128);
grayplot(T,F,TFR');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-August 1994, July 1995.

tfrbud

Butterworth time-frequency distribution

Calling Sequence

```
[TFR,T,F] = tfrbud(X)
[TFR,T,F] = tfrbud(X, T)
[TFR,T,F] = tfrbud(X, T, N)
[TFR,T,F] = tfrbud(X, T, N, G)
[TFR,T,F] = tfrbud(X, T, N, G, H)
[TFR,T,F] = tfrbud(X, T, N, G, H, SIGMA)
[TFR,T,F] = tfrbud(X, T, N, G, H, SIGMA, TRACE)
[TFR,T,F] = tfrbud(...,'plot')
```

Parameters

X :

A Nx elements vector (auto-BUD) or a Nx by 2 array signal (cross-BUD).

T:

a real Nt vector with elements in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:NX). For faster computation N should be a power of 2.

G:

a real vector with odd length: the time smoothing window, (default :Hamming(N/10)).

It will be normalized such as the middle point equals 1 to preserve signal energy.

H :

real vector with odd length: the frequency smoothing window,(default: Hamming(N/4)).

It will be normalized such as the middle point equals 1 to preserve signal energy.

SIGMA :

a positive scalar: the kernel width (default : 1).

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

F :

A N vector of normalized frequencies.

Description

tfrbud computes the Butterworth distribution of a discrete-time signal X, or the cross Butterworth representation between two signals.

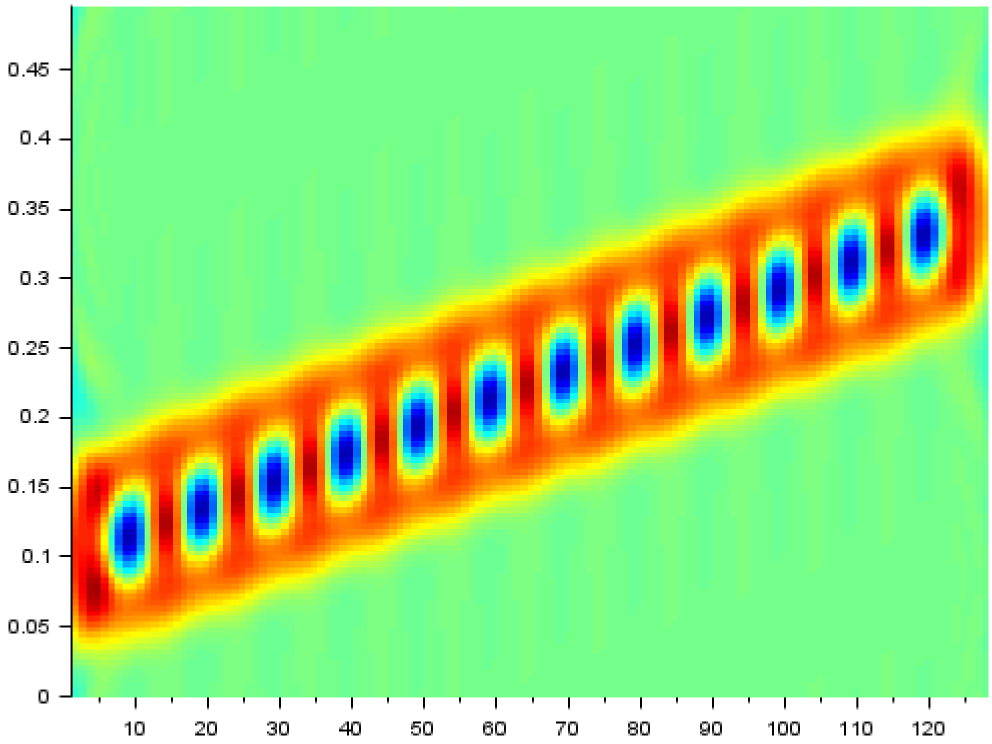
Examples

Interactive use

```
N = 128;
sig = fmlin(N,0.05,0.3) + fmlin(N,0.15,0.4);
g = window("kr",9,3*%pi);
h = window("kr",27,3*%pi);
t = 1:N;
tfrbud(sig,t,N,g,h,3.6,1,'plot');
```

Non interactive use

```
N = 128;
sig = fmlin(N,0.05,0.3) + fmlin(N,0.15,0.4);
g = window("kr",9,3*%pi);
h = window("kr",27,3*%pi);
t = 1:N;
[TFR,T,F] = tfrbud(sig,t,N,g,h,3.6);
clf; gcf().color_map = jetcolormap(128);
grayplot(T,F,TFR');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-August 1994, July 1995.
- Copyright (c) 1996 by CNRS (France).

tfrcw

Choi-Williams time-frequency distribution

Calling Sequence

```
[TFR,T,F] = tfrcw(X
[TFR,T,F] = tfrcw(X, T)
[TFR,T,F] = tfrcw(X, T, N)
[TFR,T,F] = tfrcw(X, T, N, G)
[TFR,T,F] = tfrcw(X, T, N, G, H)
[TFR,T,F] = tfrcw(X, T, N, G, H, SIGMA)
[TFR,T,F] = tfrcw(X, T, N, G, H, SIGMA)
[TFR,T,F] = tfrcw(...,'plot')
```

Parameters

X :

A Nx elements vector (auto-CW) or a Nx by 2 array signal (cross-CW).

T:

a real Nt vector with elements in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:NX). For faster computation N should be a power of 2.

G:

a real vector with odd length: the time smoothing window, (default :Hamming(N/10)).

It will be normalized such as the middle point equals 1 to preserve signal energy.

H :

real vector with odd length: the frequency smoothing window,(default: Hamming(N/4)).

It will be normalized such as the middle point equals 1 to preserve signal energy.

SIGMA :

a positive scalar: the kernel width (default : 1).

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

F :

A N vector of normalized frequencies.

Description

tfrcw computes the Choi-Williams distribution of a discrete-time signal X, or the cross Choi-Williams representation between two signals.

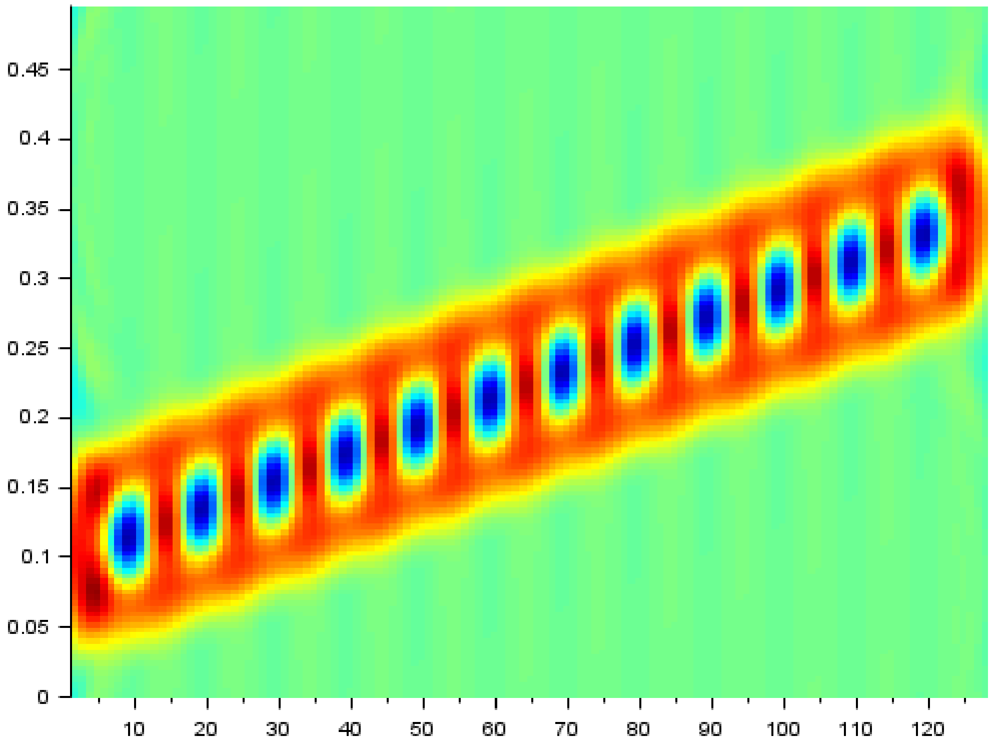
Examples

Interactive use

```
N=128;
sig=fmlin(N,0.05,0.3)+fmlin(N,0.15,0.4);
g=window("kr",9,3*%pi); h=window("kr",27,3*%pi);
t=1:N; tfrcw(sig,t,N,g,h,3.6,1,'plot');
```

Non interactive use

```
N = 128;
sig = fmlin(N,0.05,0.3) + fmlin(N,0.15,0.4);
g = window("kr",9,3*%pi);
h = window("kr",27,3*%pi);
t = 1:N;
[TFR,T,F] = tfrcw(sig,t,N,g,h,3.6);
clf; gcf().color_map = jetcolormap(128);
grayplot(T,F,TFR');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-August 1994, July 1995.
-

tfrgrd

Generalized rectangular time-frequency distribution

Calling Sequence

```
[TFR,T,F] = tfrgrd(X)
[TFR,T,F] = tfrgrd(X, T)
[TFR,T,F] = tfrgrd(X, T, N)
[TFR,T,F] = tfrgrd(X, T, N, G)
[TFR,T,F] = tfrgrd(X, T, N, G, H)
[TFR,T,F] = tfrgrd(X, T, N, G, H, RS)
[TFR,T,F] = tfrgrd(X, T, N, G, H, RS, MOVERN)
[TFR,T,F] = tfrgrd(X, T, N, G, H, RS, MOVERN, TRACE)
[TFR,T,F] = tfrgrd(...,'plot')
```

Parameters

X :

A Nx elements vector (auto-GRD) or a Nx by 2 array signal (cross-GRD).

T:

a real Nt vector : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:NX).

G:

a real vector with odd length: the time smoothing window, (default :Hamming(N/10)).

It will be normalized such as the middle point equals 1 to preserve signal energy.

H :

real vector with odd length: the frequency smoothing window,(default: Hamming(N/4)).

It will be normalized such as the middle point equals 1 to preserve signal energy.

RS :

a positive scalar: the kernel width (default : 1).

MOVERN :

a positive scalar: the dissymmetry ratio (default : 1).

TRACE :

if nonzero,the progression of the algorithm is shown (default : 0).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

F :

A N vector of normalized frequencies.

Parameters

X :

signal if auto-GRD, or [X1,X2] if cross-GRD.

T :

time instant(s) (default : 1:length(X)).

N :

number of frequency bins (default : length(X)).

G :

time smoothing window, G(0) being forced to 1. (default : Hamming(N/10)).

H :

frequency smoothing window, H(0) being forced to 1. (default : Hamming(N/4)).

RS :

kernel width (default : 1).

MOVERN :

dissymmetry ratio (default : 1).

TRACE :

if nonzero, the progression of the algorithm is shown (default : 0).

'plot':

if one input parameter is 'plot', tfrgrd runs tfrqview. and TFR will be plotted

TFR :

time-frequency representation.

F :

vector of normalized frequencies.

Description

tfrgrd computes the Generalized Rectangular distribution of a discrete-time signal X, or the cross GRD representation between two signals.

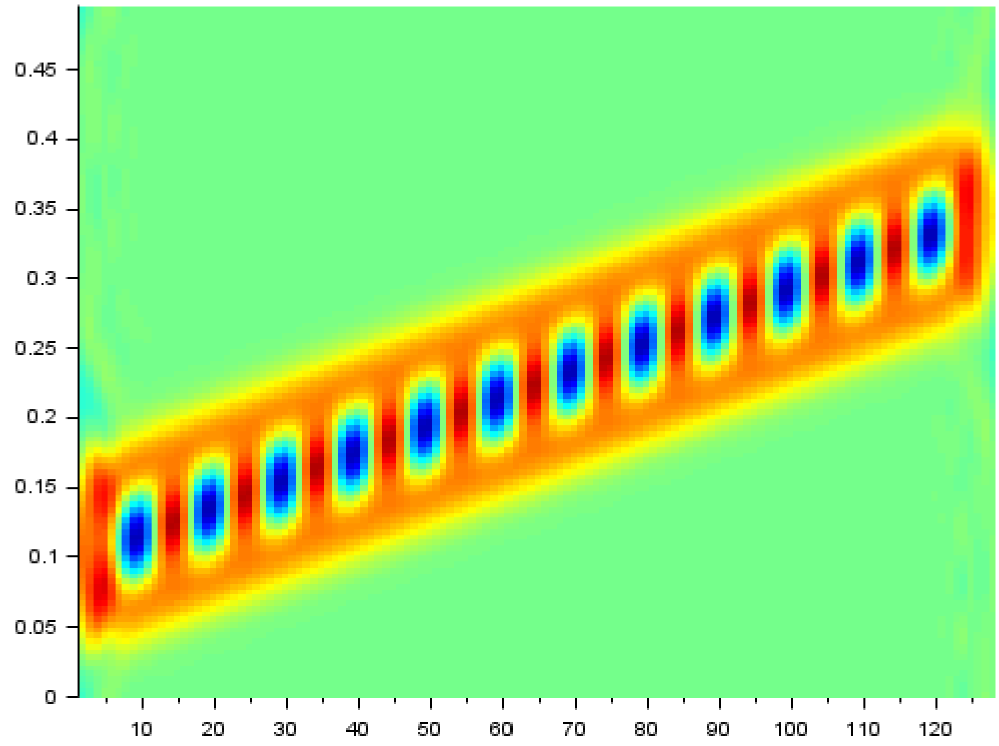
Examples

Interactive use

```
N = 128
sig = fmlin(N,0.05,0.3) + fmlin(N,0.15,0.4);
g = window("kr",9,3*%pi);
h = window("kr",27,3*%pi);
t = 1:N;
tfrgrd(sig,t,N,g,h,36,1/5,1,'plot');
```

Non interactive use

```
N = 128;
sig = fmlin(N,0.05,0.3) + fmlin(N,0.15,0.4);
g = window("kr",9,3*%pi);
h = window("kr",27,3*%pi);
t = 1:N;
[TFR,T,F] = tfrgrd(sig,t,N,g,h,36,1/5);
clf; gcf().color_map = jetcolormap(128);
grayplot(T,F,TFR');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-August 1994, July 1995.
- Copyright (c) 1996 by CNRS (France).

tfrmh

Margenau-Hill time-frequency distribution

Calling Sequence

```
[TFR,T,F] = tfrmh(X)
[TFR,T,F] = tfrmh(X, T)
[TFR,T,F] = tfrmh(X, T, N)
[TFR,T,F] = tfrmh(X, T, N, TRACE)
[TFR,T,F] = tfrmh(...,'plot')
```

Parameters

X :

A Nx elements vector (auto-MH) or a Nx by 2 array signal (cross-MH).

T:

a real Nt vector with elements in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:NX). For faster computation N should be a power of 2.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

F :

A N vector of normalized frequencies.

Description

tfrmh computes the Margenau-Hill distribution of a discrete-time signal X, or the cross Margenau-Hill representation between two signals.

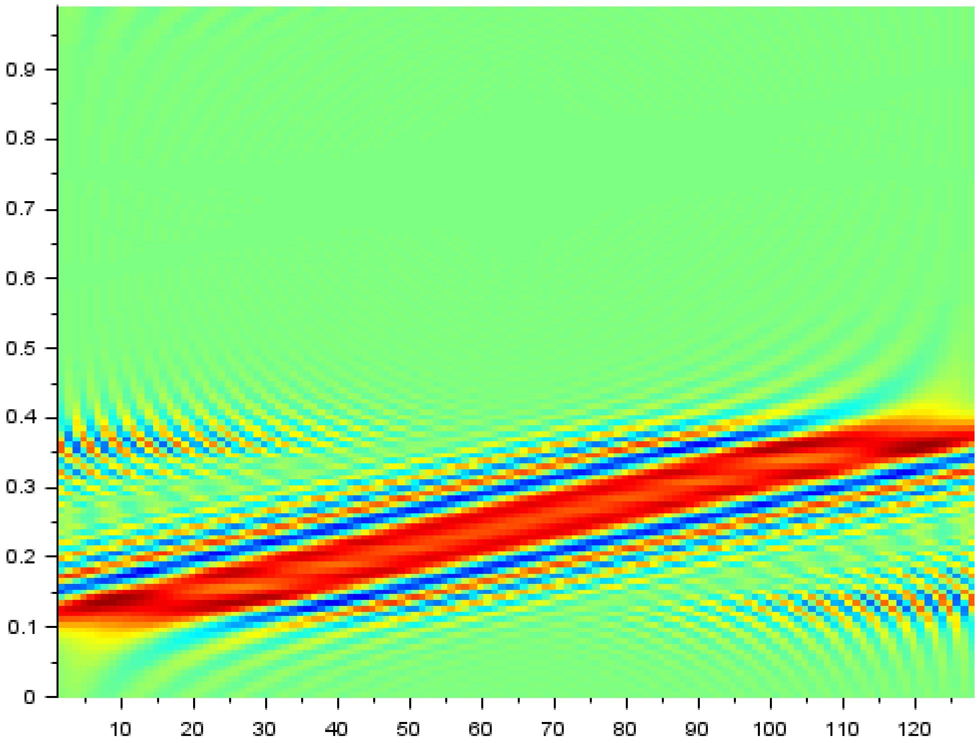
Examples

Interactive use

```
N = 128;
sig = fmlin(N,0.1,0.4);
tfrmh(sig,1:N,N,'plot');
```

Non interactive use

```
N = 128;
sig = fmlin(N,0.1,0.4);
[TFR,T,F] = tfrmh(sig,1:N,N);
clf; gcf().color_map = jetcolormap(128);
grayplot(T,F,TFR');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-August 1994, July 1995.

tfrmhs

Margenau-Hill-Spectrogram time-frequency distribution

Calling Sequence

```
[TFR,T,F] = tfrmhs(X)
[TFR,T,F] = tfrmhs(X, T)
[TFR,T,F] = tfrmhs(X, T, N)
[TFR,T,F] = tfrmhs(X, T, N, G)
[TFR,T,F] = tfrmhs(X, T, N, G, H)
[TFR,T,F] = tfrmhs(X, T, N, G, H, TRACE)
[TFR,T,F] = tfrmhs(...,'plot')
```

Parameters

X :

A Nx elements vector (auto-MHS) or a Nx by 2 array signal (cross-MHS).

T:

a real Nt vector with elements in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:NX). For faster computation N should be a power of 2.

G:

a real vector with odd length: the time smoothing window, (default :Hamming(N/10)).

H :

real vector with odd length: the frequency smoothing window,(default: Hamming(N/4)).

It will be normalized such as the middle point equals 1 to preserve signal energy.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

F :

A N vector of normalized frequencies.

Description

tfrmhs computes the Margenau-Hill-Spectrogram distribution of a discrete-time signal X, or the cross Margenau-Hill-Spectrogram representation between two signals.

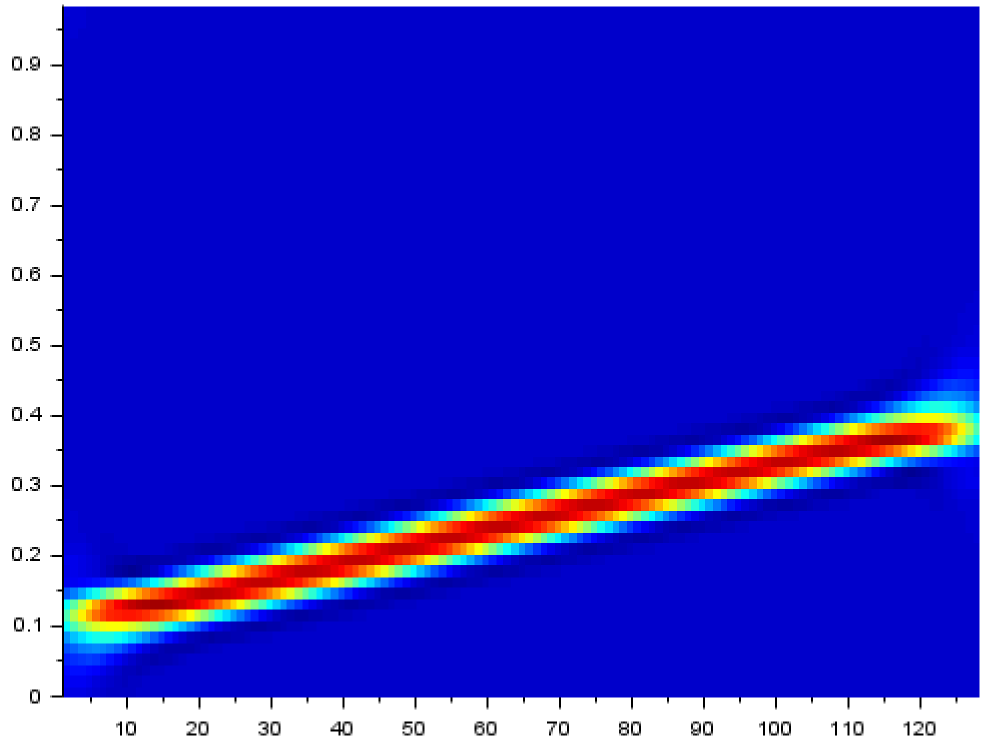
Examples

Interactive use

```
N = 128;
sig = fmlin(N,0.1,0.4);
g = window("kr",21,3*%pi);
h = window("kr",63,3*%pi);
tfrmhs(sig,1:N,64,g,h,'plot');
```

Non interactive use

```
N = 128;
sig = fmlin(N,0.1,0.4);
g = window("kr",21,3*%pi);
h = window("kr",63,3*%pi);
[TFR,T,F] = tfrmhs(sig,1:N,64,g,h);
clf; gcf().color_map = jetcolormap(128);
grayplot(T,F,TFR');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-August 1994, July 1995.

tfrmmce

Minimum mean cross-entropy combination of spectrograms

Calling Sequence

```
[TFR,T,F] = tfrmmce(X, H)
[TFR,T,F] = tfrmmce(X, H, T)
[TFR,T,F] = tfrmmce(X, H, T, N)
[TFR,T,F] = tfrmmce(X, H, T, N, TRACE)
[TFR,T,F] = tfrmmce(...,'plot')
```

Parameters

X :

A Nx elements vector.

H :

A real array with at least two columns and an odd number of rows .

T:

a real Nt vector with elements in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:NX). For faster computation N should be a power of 2.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

F :

A N vector of normalized frequencies.

Description

tfrmmce computes the minimum mean cross-entropy combination of spectrograms using as windows the columns of the matrix H.

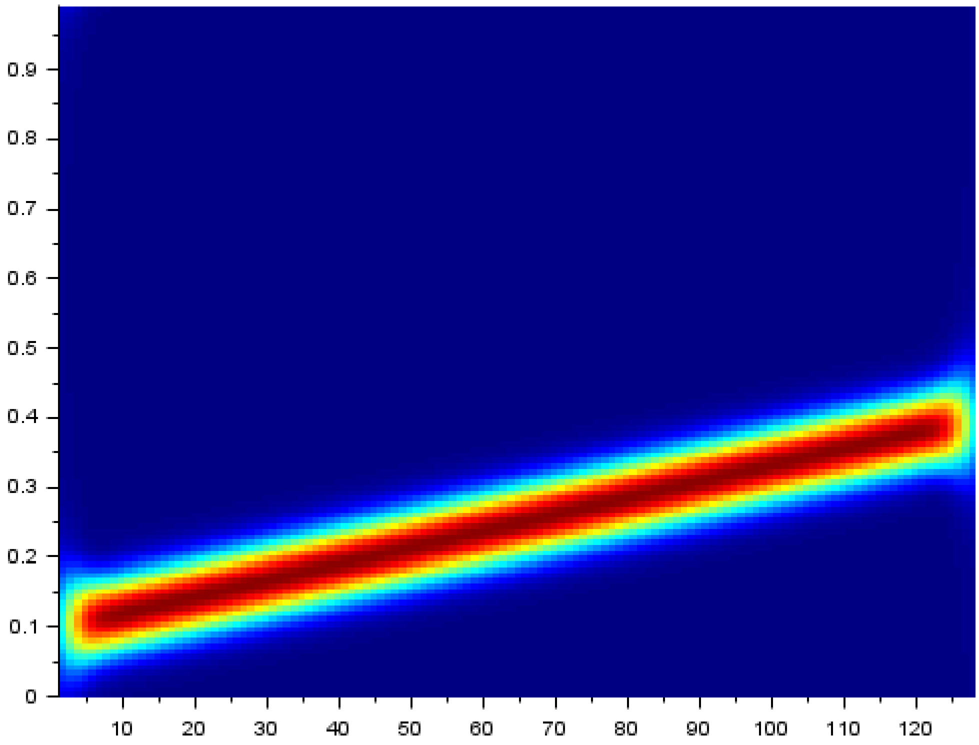
Examples

Interactive use

```
N = 128;
sig = fmlin(N,0.1,0.4);
h = zeros(19,3);
h(10+(-5:5),1) = window("hm",11);
h(10+(-7:7),2) = window("hm",15);
h(10+(-9:9),3) = window("hm",19);
tfrmmce(sig,h,'plot');
```

Non interactive use

```
N = 128;
sig = fmlin(N,0.1,0.4);
h = zeros(19,3);
h(10+(-5:5),1) = window("hm",11);
h(10+(-7:7),2) = window("hm",15);
h(10+(-9:9),3) = window("hm",19);
[TFR,T,F] = tfrmmce(sig,h);
clf; gcf().color_map = jetcolormap(128);
grayplot(T,F,TFR');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, August 1995.

tfrpage

Page time-frequency distribution

Calling Sequence

```
[TFR,T,F] = tfrpage(X)
[TFR,T,F] = tfrpage(X, T)
[TFR,T,F] = tfrpage(X, T, N)
[TFR,T,F] = tfrpage(X, T, N, TRACE)
[TFR,T,F] = tfrpage(...,'plot')
```

Parameters

X :

A Nx elements vector (auto-PAGE) or a Nx by 2 array signal (cross-PAGE).

T:

a real Nt vector with elements in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:NX). For faster computation N should be a power of 2.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

F :

A N vector of normalized frequencies.

Description

tfrpage computes the Page distribution of a discrete-time signal X, or the cross Page representation between two signals.

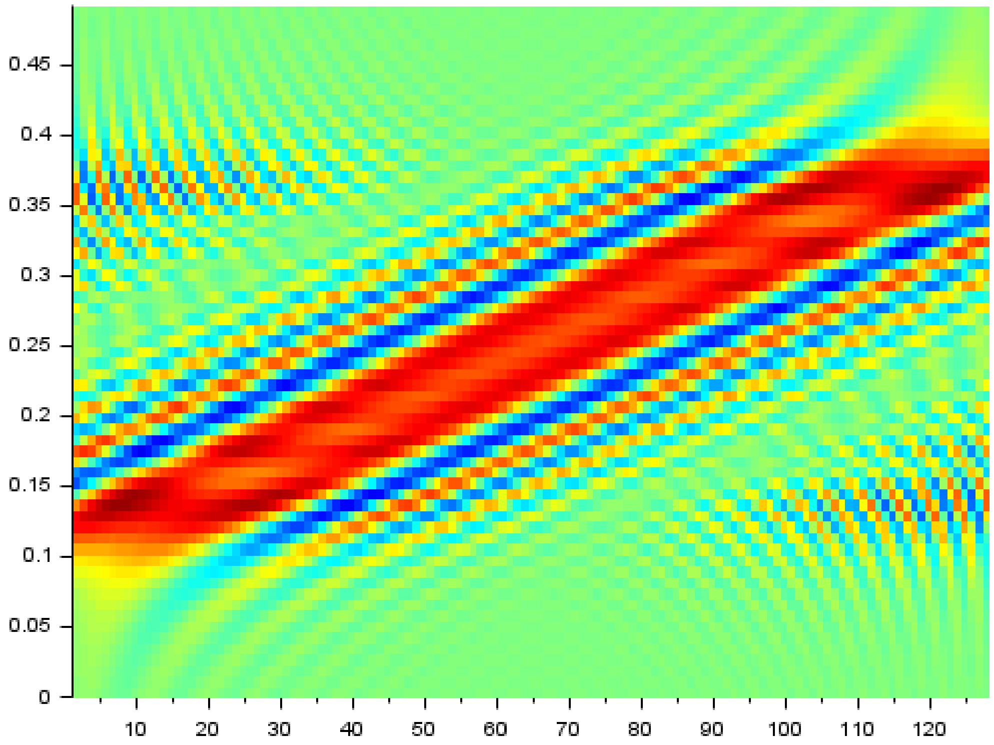
Examples

Interactive use

```
N = 128;
sig = fmlin(N,0.1,0.4);
tfrpage(sig,'plot');
```

Non interactive use

```
N = 128;
sig = fmlin(N,0.1,0.4);
[TFR,T,F] = tfrpage(sig);
clf; gcf().color_map = jetcolormap(128);
grayplot(T,F(1:N/2),TFR(1:N/2,:)');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-August 1994, July 1995.

tfrpmh

Pseudo Margenau-Hill time-frequency distribution

Calling Sequence

```
[TFR,T,F] = tfrpmh(X)
[TFR,T,F] = tfrpmh(X, T)
[TFR,T,F] = tfrpmh(X, T, N)
[TFR,T,F] = tfrpmh(X, T, N, H)
[TFR,T,F] = tfrpmh(X, T, N, H, TRACE)
[TFR,T,F] = tfrpmh(...,'plot')
```

Parameters

X :

A Nx elements vector (auto-PMH) or a Nx by 2 array signal (cross-PMH).

T:

a real Nt vector with elements in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:Nx). For faster computation N should be a power of 2.

H :

real vector with odd length: the frequency smoothing window,(default: Hamming(N/4)).

It will be normalized such as the middle point equals 1 to preserve signal energy.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

F :

A N vector of normalized frequencies.

Description

tfrcw computes the Choi-Williams distribution of a discrete-time signal X, or the cross Choi-Williams representation between two signals.

Description

tfrpmh computes the Pseudo Margenau-Hill distribution of a discrete-time signal X, or the cross Pseudo Margenau-Hill representation between two signals.

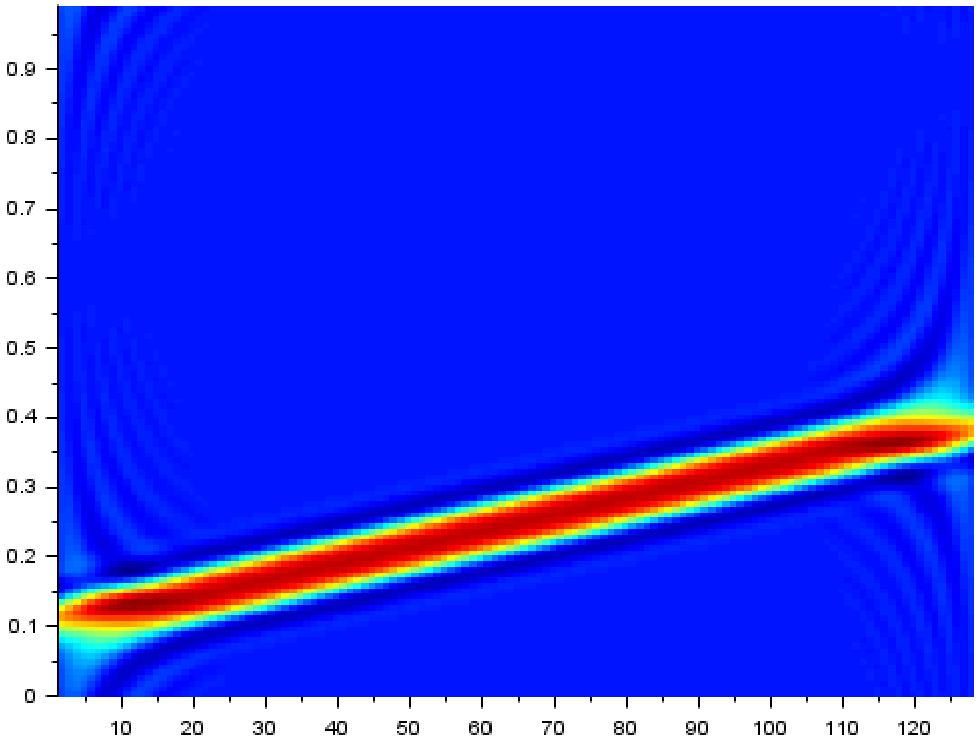
Examples

Interactive use

```
N = 128;
sig = fmlin(N,0.1,0.4);
h = window("kr",N/2-1,3*%pi);
tfrpmh(sig,1:N,N,h,'plot');
```

Non interactive use

```
N = 128;
sig = fmlin(N,0.1,0.4);
h = window("kr",N/2-1,3*%pi);
[TFR,T,F] = tfrpmh(sig,1:N,N,h);
clf; gcf().color_map = jetcolormap(128);
grayplot(T,F,TFR');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-August 1994, July 1995.
- Copyright (c) 1996 by CNRS (France).

tfrppage

Pseudo Page time-frequency distribution

Calling Sequence

```
[TFR,T,F] = tfrppage(X)
[TFR,T,F] = tfrppage(X, T)
[TFR,T,F] = tfrppage(X, T, N)
[TFR,T,F] = tfrppage(X, T, N, H)
[TFR,T,F] = tfrppage(X, T, N, H, TRACE)
[TFR,T,F] = tfrppage(...,'plot')
```

Parameters

X :

A Nx elements vector (auto-PPAGE) or a Nx by 2 array signal (cross-PPAGE).

T:

a real Nt vector with elements in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:Nx). For faster computation N should be a power of 2.

H :

real vector with odd length: the frequency smoothing window,(default: Hamming(N/4)).

It will be normalized such as the middle point equals 1 to preserve signal energy.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

F :

A N vector of normalized frequencies.

Description

tfrppage computes the Pseudo Page distribution of a discrete-time signal X, or the cross Pseudo Page representation between two signals.

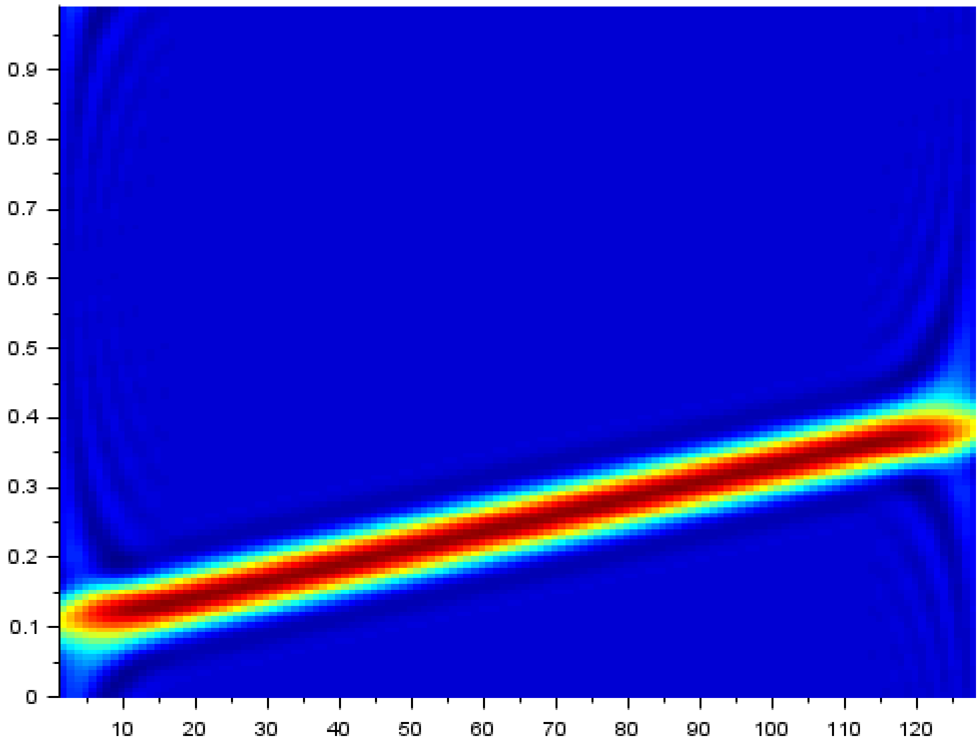
Examples

Interactive use

```
n = 128;
sig = fmlin(N,0.1,0.4);
tfrppage(sig,'plot');
```

Non interactive use

```
N = 128;
sig = fmlin(N,0.1,0.4);
[TFR,T,F] = tfrppage(sig,1:N,N,window("hm",N/4+1));
clf; gcf().color_map = jetcolormap(128);
grayplot(T,F,TFR');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-August 1994, July 1995.

tfrpwv

Pseudo Wigner-Ville time-frequency distribution

Calling Sequence

```
[TFR,T,F] = tfrpwv(X)
[TFR,T,F] = tfrpwv(X, T)
[TFR,T,F] = tfrpwv(X, T, N)
[TFR,T,F] = tfrpwv(X, T, N, H)
[TFR,T,F] = tfrpwv(X, T, N, H, TRACE)
[TFR,T,F] = tfrpwv(...,'plot')
```

Parameters

X :

A Nx elements vector (auto-PWV) or a Nx by 2 array signal (cross-PWV).

T:

a real Nt vector with elements in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:Nx). For faster computation N should be a power of 2.

H :

real vector with odd length: the frequency smoothing window,(default: window("hm",N/4)).

It will be normalized such as the middle point equals 1 to preserve signal energy.

TRACE :

if nonzero,the progression of the algorithm is shown (default : 0).

'plot':

if the last input parameter value is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

F :

A N vector of normalized frequencies.

Description

tfrpwv computes the Pseudo Wigner-Ville distribution of a discrete-time signal X, or the cross Pseudo Wigner-Ville representation between two signals.

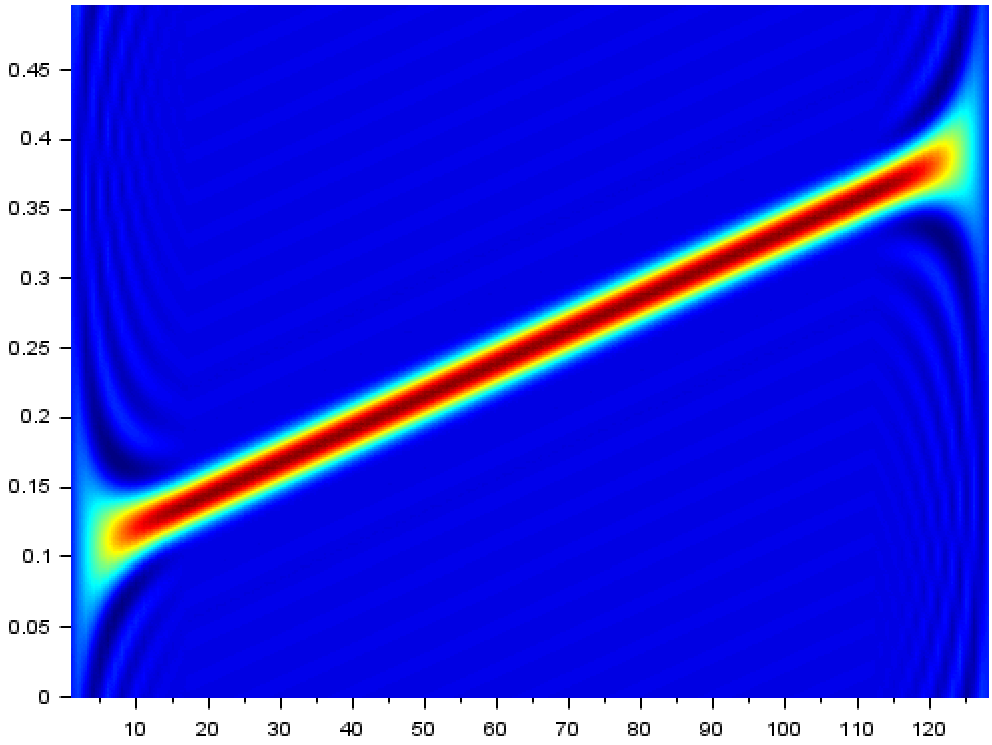
Examples

Interactive use

```
N = 128;
sig = fmlin(N,0.1,0.4);
tfrpwv(sig,'plot');
```

Non interactive use

```
N = 128;
sig = fmlin(N,0.1,0.4);
[TFR,T,F] = tfrpwv(sig,1:N,N,window("hm",N/4+1));
clf; gcf().color_map = jetcolormap(128);
Sgrayplot(T,F,TFR');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-August 1994, July 1995.

tfrri

Rihaczek time-frequency distribution.

Calling Sequence

```
[TFR,T,F] = tfrri(X)
[TFR,T,F] = tfrri(X, T)
[TFR,T,F] = tfrri(X, T, N)
[TFR,T,F] = tfrri(X, T, N, TRACE)
[TFR,T,F] = tfrri(...,'plot')
```

Parameters

X :

A Nx elements vector (auto-RI) or a Nx by 2 array signal (cross-RI).

T:

a real Nt vector with elements in [1 Nx]: time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:NX). For faster computation N should be a power of 2.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A complex N by Nt array: the time-frequency representation.

F :

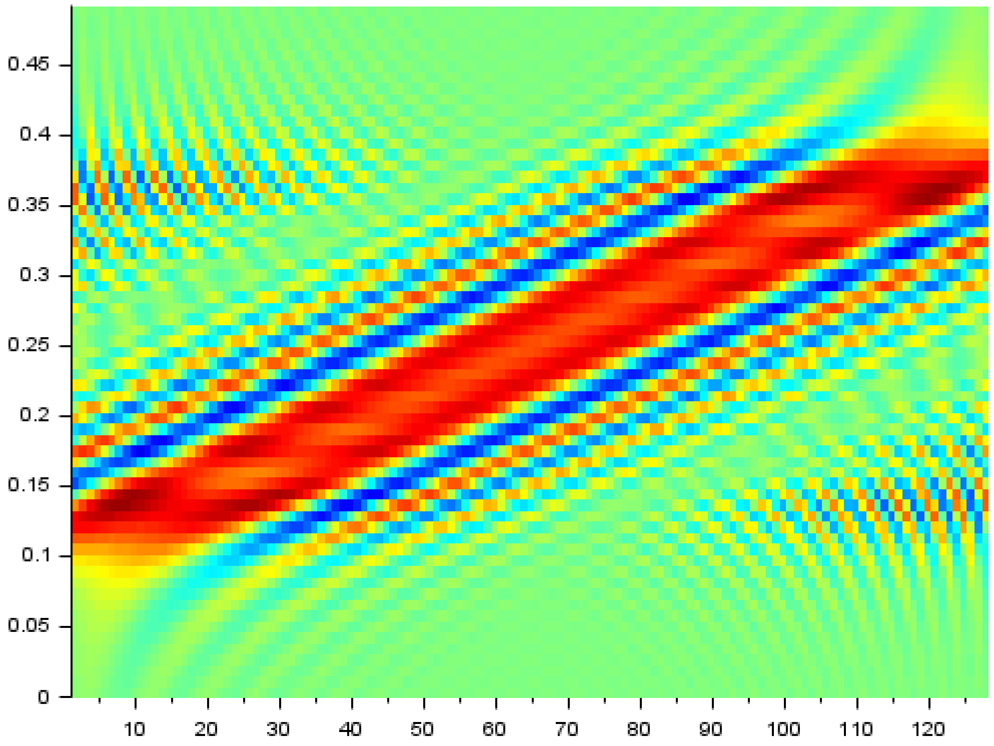
A N vector of normalized frequencies.

Description

tfrri computes the Rihaczek distribution of a discrete-time signal X, or the cross Rihaczek representation between two signals.

Examples

```
N = 128;
sig = fmlin(N,0.1,0.4);
[TFR,T,F] = tfrri(sig);
clf;gcf().color_map = jetcolormap(128);
grayplot(T,F(1:N/2),real(TFR(1:N/2,:))');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-August 1994, July 1995.
- Copyright (c) 1996 by CNRS (France).

tfrridb

Reduced Interference Distribution with Bessel kernel.

Calling Sequence

```
[TFR,T,F] = tfrridb(X)
[TFR,T,F] = tfrridb(X, T)
[TFR,T,F] = tfrridb(X, T, N)
[TFR,T,F] = tfrridb(X, T, N, G)
[TFR,T,F] = tfrridb(X, T, N, G, H)
[TFR,T,F] = tfrridb(X, T, N, G, H, TRACE)
[TFR,T,F] = tfrridb(...,'plot')
```

Parameters

X :

A Nx elements vector (auto-RIDB) or a Nx by 2 array signal (cross-RIDB).

T:

a real Nt vector with elements in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:NX). For faster computation N should be a power of 2.

G:

a real vector with odd length: the time smoothing window, (default :Hamming(N/10)).

It will be normalized such as the middle point equals 1 to preserve signal energy .

H :

real vector with odd length: the frequency smoothing window,(default: Hamming(N/4)).

It will be normalized such as the middle point equals 1 to preserve signal energy.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A N by Nt array: the time-frequency representation.

F :

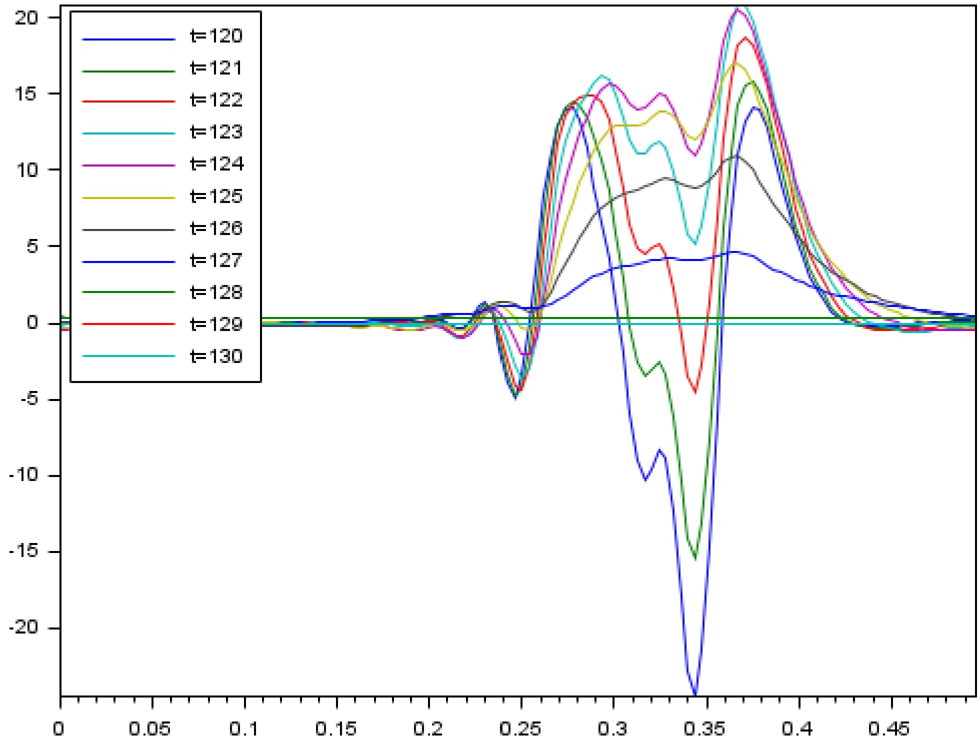
A N vector of normalized frequencies.

Description

tfrridb Reduced Interference Distribution with a kernel based on the Bessel function of the first kind. tfrridb computes either the distribution of a discrete-time signal X, or the cross representation between two signals.

Examples

```
N = 128;
sig = [fmlin(N,0.05,0.3)+fmlin(N,0.15,0.4) ;
zeros(N/2,1)];
g = window("re",31);
h = window("re",63);
t = 120:130;
[TFR,T,F] = tfrridb(sig,t,128,g,h,0);

clf; plot(F,TFR);
legend("t="+string(120:130),"in_upper_left");
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, August 1995.
- Copyright (c) 1996 by CNRS (France).

tfrridbn

Reduced Interference Distribution with a binomial kernel.

Calling Sequence

```
[TFR,T,F] = tfrridbn(X)
[TFR,T,F] = tfrridbn(X, T)
[TFR,T,F] = tfrridbn(X, T, N)
[TFR,T,F] = tfrridbn(X, T, N, G)
[TFR,T,F] = tfrridbn(X, T, N, G, H)
[TFR,T,F] = tfrridbn(X, T, N, G, H, TRACE)
[TFR,T,F] = tfrridbn(...,'plot')
```

Parameters

X :

A Nx elements vector (auto-RIDBN) or a Nx by 2 array signal (cross-RIDBN).

T:

a real Nt vector with elements in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:Nx). For faster computation N should be a power of 2.

G:

a real vector with odd length: the time smoothing window, (default :Hamming(N/10)).

It will be normalized such as the middle point equals 1 to preserve signal energy .

H :

real vector with odd length: the frequency smoothing window,(default: Hamming(N/4)).

It will be normalized such as the middle point equals 1 to preserve signal energy.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A N by Nt array: the time-frequency representation.

F :

A N vector of normalized frequencies.

Description

tfrridbn Reduced Interference Distribution with a kernel based on the binomial coefficients. tfrridbn computes either the distribution of a discrete-time signal X, or the cross representation between two signals.

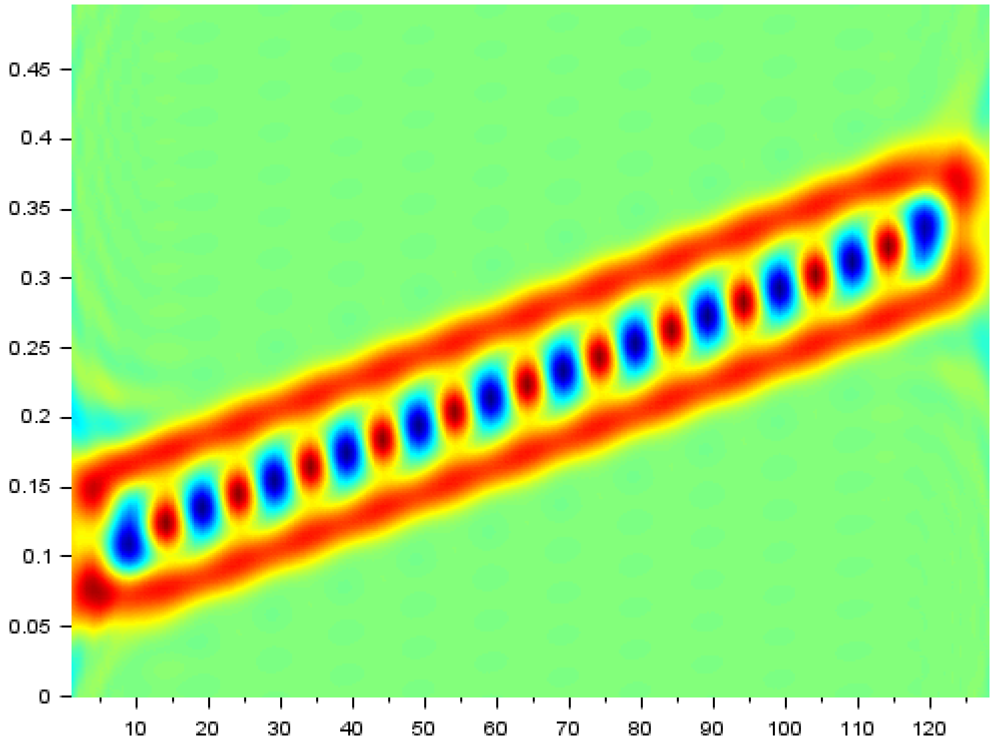
Examples

Interactive use

```
sig = fmlin(128,.05,.3) + fmlin(128,.15,.4);
tfrridbn(sig,1:N,N,window("hm",13),window("hm",33),'plot');
```

Non interactive use

```
N = 128;
sig = fmlin(N,.05,.3) + fmlin(N,.15,.4);
[TFR,T,F] =
tfrridbn(sig,1:N,N,window("hm",13),window("hm",33));
clf; gcf().color_map= jetcolormap(128);
Sgrayplot(T,F,TFR');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, June 1996.
- Copyright (c) 1996 by CNRS (France).

tfrridh

Reduced Interference Distribution with Hanning kernel.

Calling Sequence

```
[TFR,T,F] = tfrridh(X)
[TFR,T,F] = tfrridh(X, T)
[TFR,T,F] = tfrridh(X, T, N)
[TFR,T,F] = tfrridh(X, T, N, G)
[TFR,T,F] = tfrridh(X, T, N, G, H)
[TFR,T,F] = tfrridh(X, T, N, G, H, TRACE)
[TFR,T,F] = tfrridh(...,'plot')
```

Parameters

X :

A Nx elements vector (auto-RIDH) or a Nx by 2 array signal (cross-RIDH).

T:

a real Nt vector with elements in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:NX). For faster computation N should be a power of 2.

G:

a real vector with odd length: the time smoothing window, (default :Hamming(N/10)).

It will be normalized such as the middle point equals 1 to preserve signal energy .

H :

real vector with odd length: the frequency smoothing window,(default: Hamming(N/4)).

It will be normalized such as the middle point equals 1 to preserve signal energy.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

F :

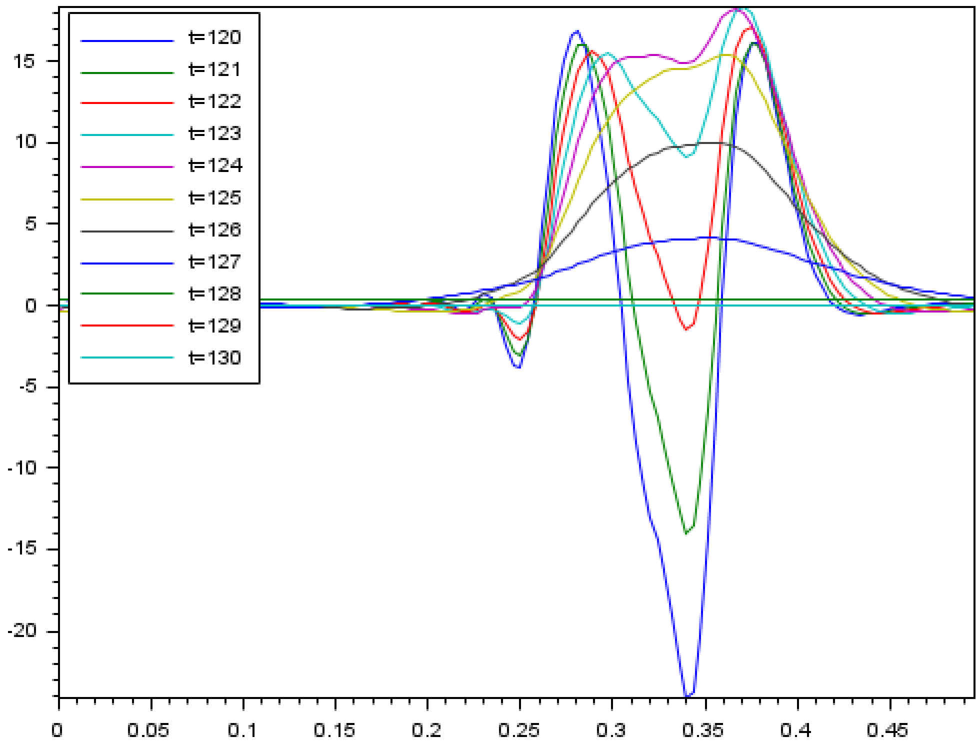
A N vector of normalized frequencies.

Description

tfrridh Reduced Interference Distribution with a kernel based on the Hanning window. tfrridh computes either the distribution of a discrete-time signal X, or the cross representation between two signals.

Examples

```
sig = [fmlin(128,0.05,0.3) + fmlin(128,0.15,0.4) ;
zeros(64,1)];
g = window("re",31); h=window("re",63);
t = 120:130;
[TFR,T,F] = tfrridh(sig,t,128,g,h,0);
clf; plot(F,TFR);
legend("t="+string(120:130),"in_upper_left");
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-August 1994, August 1995.
- Copyright (c) 1996 by CNRS (France).

tfrridt

Reduced Interference Distribution with triangular kernel.

Calling Sequence

```
[TFR,T,F] = tfrridt(X)
[TFR,T,F] = tfrridt(X, T)
[TFR,T,F] = tfrridt(X, T, N)
[TFR,T,F] = tfrridt(X, T, N, G)
[TFR,T,F] = tfrridt(X, T, N, G, H)
[TFR,T,F] = tfrridt(X, T, N, G, H, TRACE)
[TFR,T,F] = tfrridt(...,'plot')
```

Parameters

X :

A Nx elements vector (auto-RIDT) or a Nx by 2 array signal (cross-RIDT).

T:

a real Nt vector with elements in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:NX). For faster computation N should be a power of 2.

G:

a real vector with odd length: the time smoothing window, (default :Hamming(N/10)).

It will be normalized such as the middle point equals 1 to preserve signal energy .

H :

real vector with odd length: the frequency smoothing window,(default: Hamming(N/4)).

It will be normalized such as the middle point equals 1 to preserve signal energy.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

F :

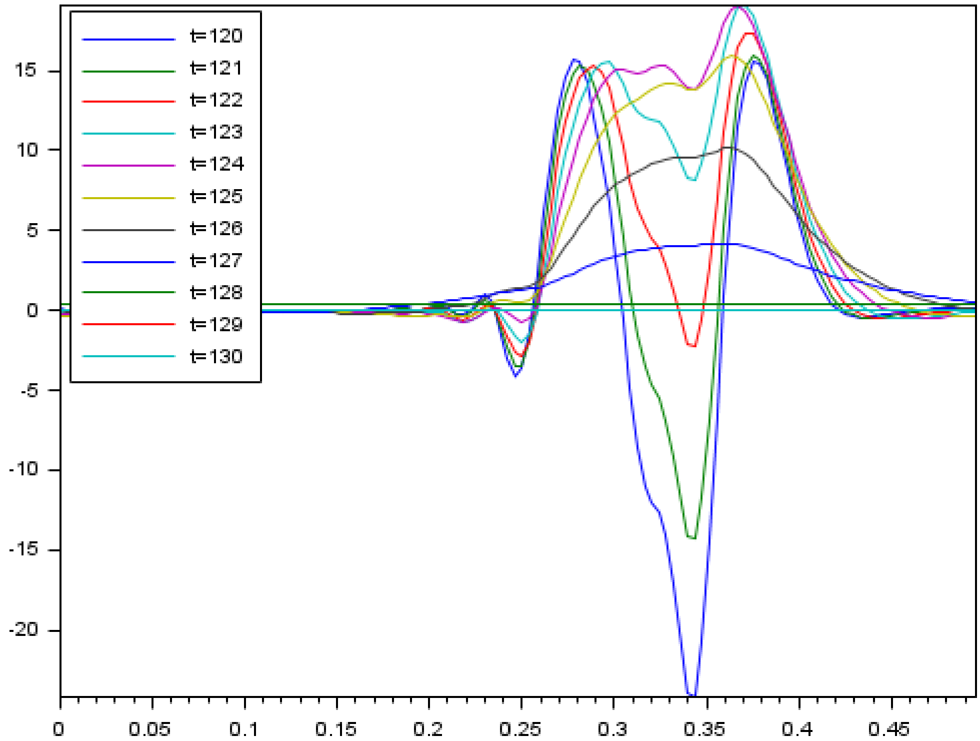
A N vector of normalized frequencies.

Description

tfrridt Reduced Interference Distribution with a kernel based on the triangular (or Bartlett) window. tfrridt computes either the distribution of a discrete-time signal X, or the cross representation between two signals.

Examples

```
sig = [fmlin(128,0.05,0.3)+fmlin(128,0.15,0.4) ;
zeros(64,1)];
g = window("re",31);
h = window("re",63);
[TFR,T,F] = tfrridt(sig,120:130,128,g,h,0);
clf; plot(F,TFR);
legend("t="+string(120:130),"in_upper_left");
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-August 1994, August 1995.
- Copyright (c) 1996 by CNRS (France).

tfrsp

Spectrogram time-frequency distribution.

Calling Sequence

```
[TFR,T,F] = tfrsp(X)
[TFR,T,F] = tfrsp(X, T)
[TFR,T,F] = tfrsp(X, T, N)
[TFR,T,F] = tfrsp(X, T, N, H)
[TFR,T,F] = tfrsp(X, T, N, H, TRACE)
[TFR,T,F] = tfrsp(...,'plot')
```

Description

tfrsp computes the Spectrogram distribution of a discrete-time signal X.

Parameters

X :

A Nx elements vector (auto-SP) or a Nx by 2 array signal (cross-SP).

T:

a real Nt vector with elements in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:NX). For faster computation N should be a power of 2.

H:

a real vector with odd length: the analysis window, H being normalized so as to be of unit energy. (default : Hamming(N/4)).

TRACE :

if nonzero,the progression of the algorithm is shown (default : 0).

'plot':

if the last input parameter value is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

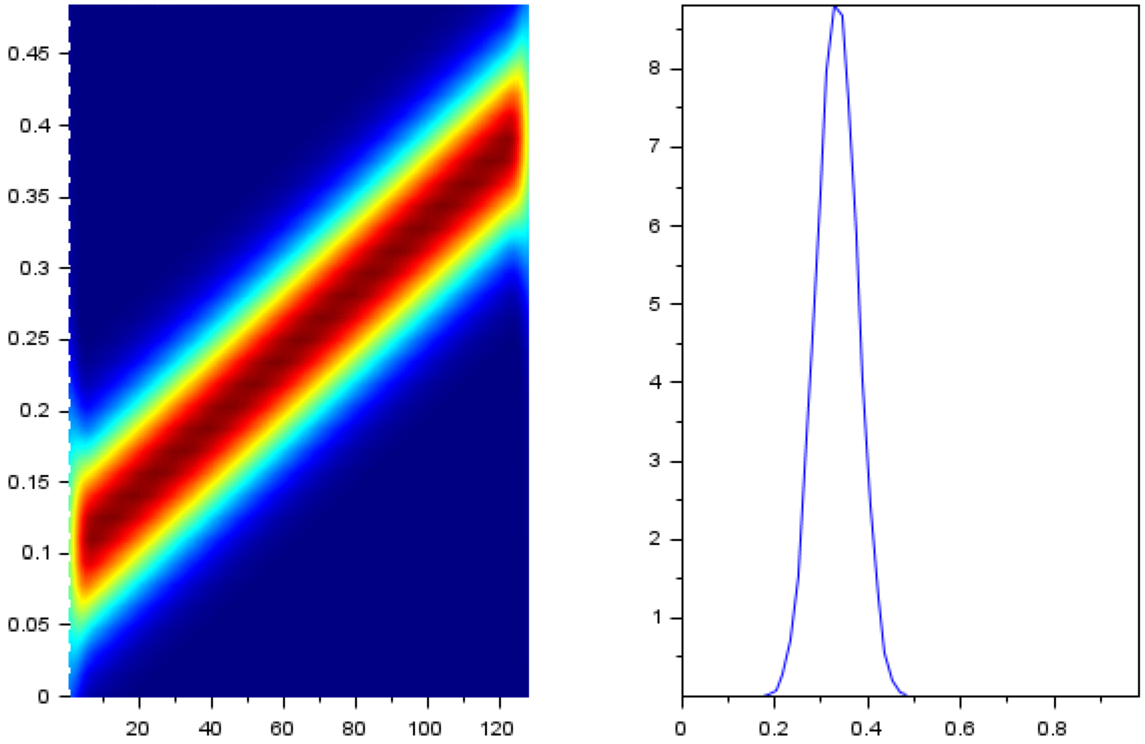
A real N by Nt array: the time-frequency representation.

F :

A N vector of normalized frequencies.

Examples

```
N = 128;
sig = fmlin(N,0.1,0.4);
h = window("kr",17,3*%pi);

[TFR,T,F] = tfrsp(sig,1:N,N/2,h);
clf; gcf().color_map = jetcolormap(128);
subplot(121); Sgrayplot(T,F(1:$/2),TFR(1:$/2,:)');
subplot(122); plot(fftshift(F'),fftshift(TFR(:,100)))
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-August 1994, July 1995.
- Copyright (c) 1996 by CNRS (France).

tfrspbk

Smoothed Pseudo K-Bertrand time-frequency distribution.

Calling Sequence

```
[TFR,T,F] = tfrspbk(X)
[TFR,T,F] = tfrspbk(X, T)
[TFR,T,F] = tfrspbk(X, T, K)
[TFR,T,F] = tfrspbk(X, T, K, NH0)
[TFR,T,F] = tfrspbk(X, T, K, NH0,NG0)
[TFR,T,F] = tfrspbk(X, T, K, NH0,NG0, FMIN,FMAX)
[TFR,T,F] = tfrspbk(X, T, K, NH0,NG0, FMIN,FMAX, N)
[TFR,T,F] = tfrspbk(X, T, K, NH0,NG0, FMIN,FMAX, N,
TRACE)
[TFR,T,F] = tfrspbk(...,'plot')
```

Parameters

X :

A Nx elements vector (auto-SPBK) or a Nx by 2 array signal (cross-SPBK).

T:

a real Nt vector : time instant(s) (default: 1:NX).

T must be a uniformly sampled vector whose elements are between 1 and Nx

K :

label of the K-Bertrand distribution. The distribution with parametrization function lambdak(u,K) = (K (exp(-u)-1)/(exp(-Ku)-1))^(1/(K-1)) is computed (default : 0).

K=-1 : Smoothed pseudo (active) Unterberger distribution

K=0 : Smoothed pseudo Bertrand distribution

K=0.5 : Smoothed pseudo D-Flandrin distribution

K=2 : Affine smoothed pseudo Wigner-Ville distribution

NH0 :

half length of the analyzing wavelet at coarsest scale. A Morlet wavelet is used. NH0 controles the frequency smoothing of the smoothed pseudo K-Bertrand distribution. (default : sqrt(Nx)).

NG0 :

half length of the time smoothing window. NG0 = 0 corresponds to the Pseudo K-Bertrand distribution. (default : 0).

FMIN:

a positive scalar in]0 0.5], the normalized lower frequency bound in (Hz) of the analyzed signal. When unspecified, you have to enter it at the command line from the plot of the spectrum.

FMAX :

a positive scalar in]0 0.5], the normalized upper frequency bound (in Hz) of the analyzed signal. When unspecified, you have to enter it at the command line from the plot of the spectrum.

N :

positive integer: the number of analyzed voices. When unspecified, you have to enter it at the command line from the plot of the spectrum.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A complex N by Nt array: the time-frequency representation.

abscissa correspond to uniformly sampled time, and ordinates correspond to a geometrically sampled frequency). First row of TFR corresponds to the lowest frequency.

F :

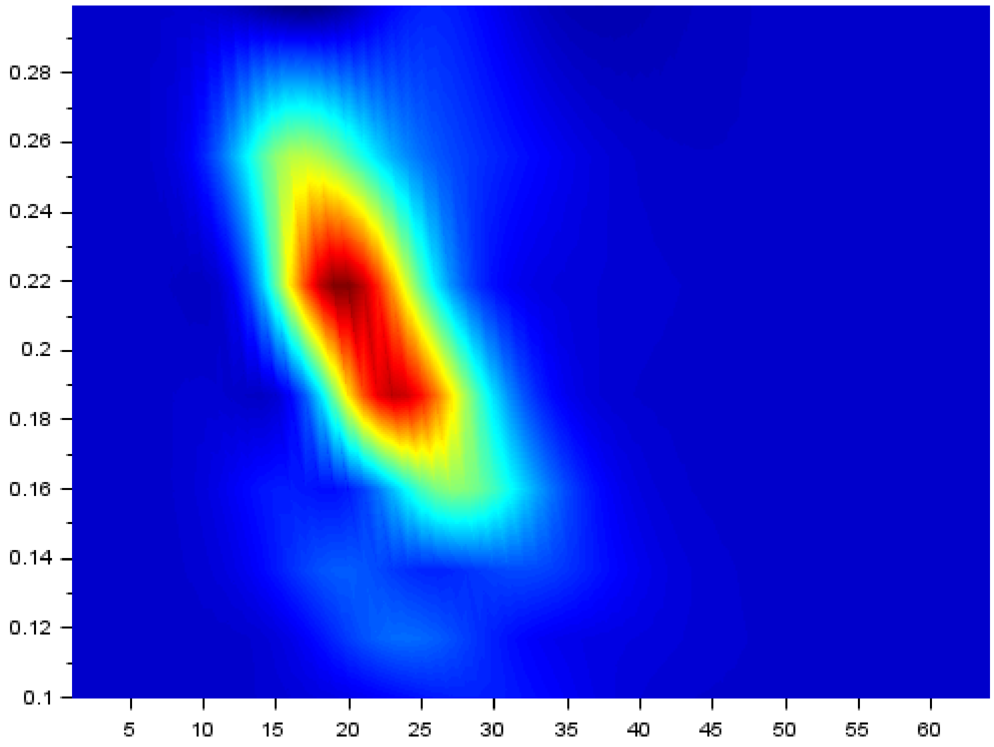
A N vector of normalized frequencies (geometrically sampled from FMIN to FMAX).

Description

tfrspbk generates the auto or cross Smoothed Pseudo K-Bertrand distribution.

Examples

```
N = 64,
sig = altes(N,0.1,0.45);
[TFR,T,F] = tfrspbk(sig,1:N,0,sqrt(N),0,0.1,0.35,8);
clf; gcf().color_map = jetcolormap(128);
Sgrayplot(T,F,TFR');
```



Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves, October 95 - O. Lemoine, June 1996.
- Copyright (c) 1995 Rice University

tfrspwv

Smoothed Pseudo Wigner-Ville time-frequency distribution.

Calling Sequence

```
[TFR,T,F] = tfrspwv(X)
[TFR,T,F] = tfrspwv(X, T)
[TFR,T,F] = tfrspwv(X, T, N)
[TFR,T,F] = tfrspwv(X, T, N, G
[TFR,T,F] = tfrspwv(X, T, N, G, H)
[TFR,T,F] = tfrspwv(X, T, N, G, H, TRACE)
[TFR,T,F] = tfrspwv(...,'plot')
```

Parameters

X :

A Nx elements vector (auto-SPWV) or a Nx by 2 array signal (cross-SPWV).

T:

a real Nt vector with elements in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:NX). For faster computation N should be a power of 2.

G:

a real vector with odd length: the time smoothing window, (default :Hamming(N/10)).

It will be normalized such as the middle point equals 1 to preserve signal energy .

H :

real vector with odd length: the frequency smoothing window,(default: Hamming(N/4)).

It will be normalized such as the middle point equals 1 to preserve signal energy.

TRACE :

if nonzero,the progression of the algorithm is shown (default : 0).

'plot':

if the last input parameter value is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

F :

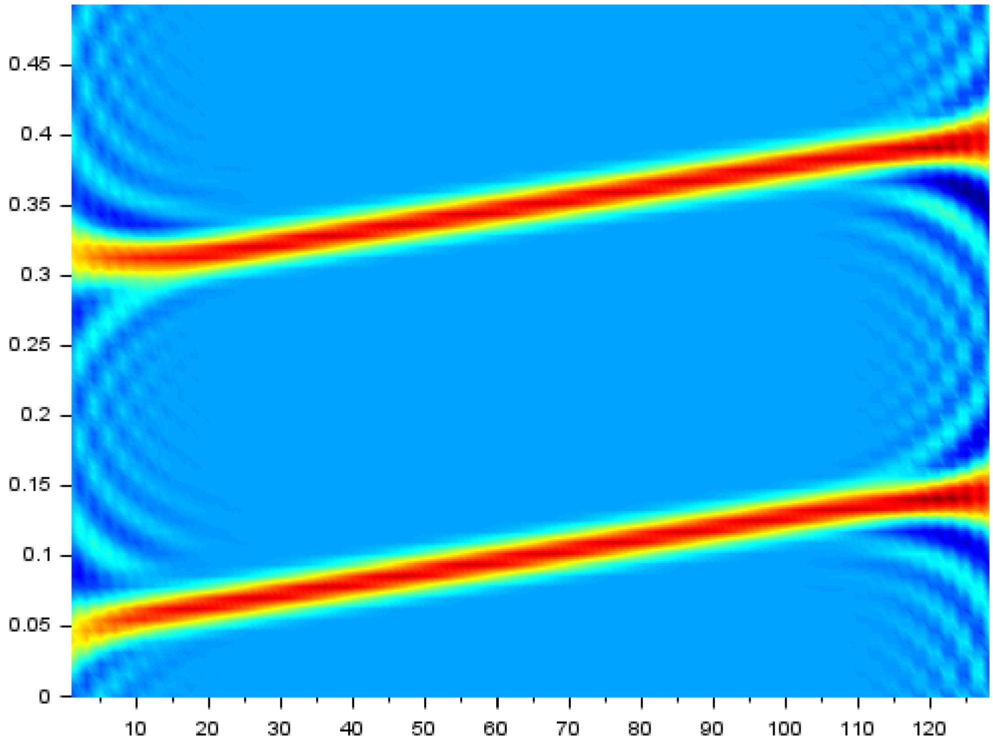
A N vector of normalized frequencies.

Description

computes the Smoothed Pseudo Wigner-Ville distribution of a discrete-time signal X, or the cross Smoothed Pseudo Wigner-Ville representation between two signals.

Examples

```
N = 128;
sig = fmlin(N,0.05,0.15) + fmlin(N,0.3,0.4);
g = window("kr",15,3*%pi);
h = window("kr",63,3*%pi);
[TFR,T,F] = tfrspwv(sig,1:N,N/2,g,h);
clf; gcf().color_map = jetcolormap(128);
Sgrayplot(T,F,TFR');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-August 1994, July 1995.
- Copyright (c) 1996 by CNRS (France).

tfrwv

Wigner-Ville time-frequency distribution

Calling Sequence

```
[TFR,T,F] = tfrwv(X)
[TFR,T,F] = tfrwv(X, T)
[TFR,T,F] = tfrwv(X, T, N)
[TFR,T,F] = tfrwv(X, T, N, TRACE)
[TFR,T,F] = tfrwv(...,'plot')
```

Parameters

X :

A Nx elements vector (auto-WV) or a Nx by 2 array signal (cross-WV).

T:

a real Nt vector with elements in [1 Nx] : time instant(s) on which the TFR is evaluated (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:NX).

TRACE :

if nonzero,the progression of the algorithm is shown (default : 0).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

F :

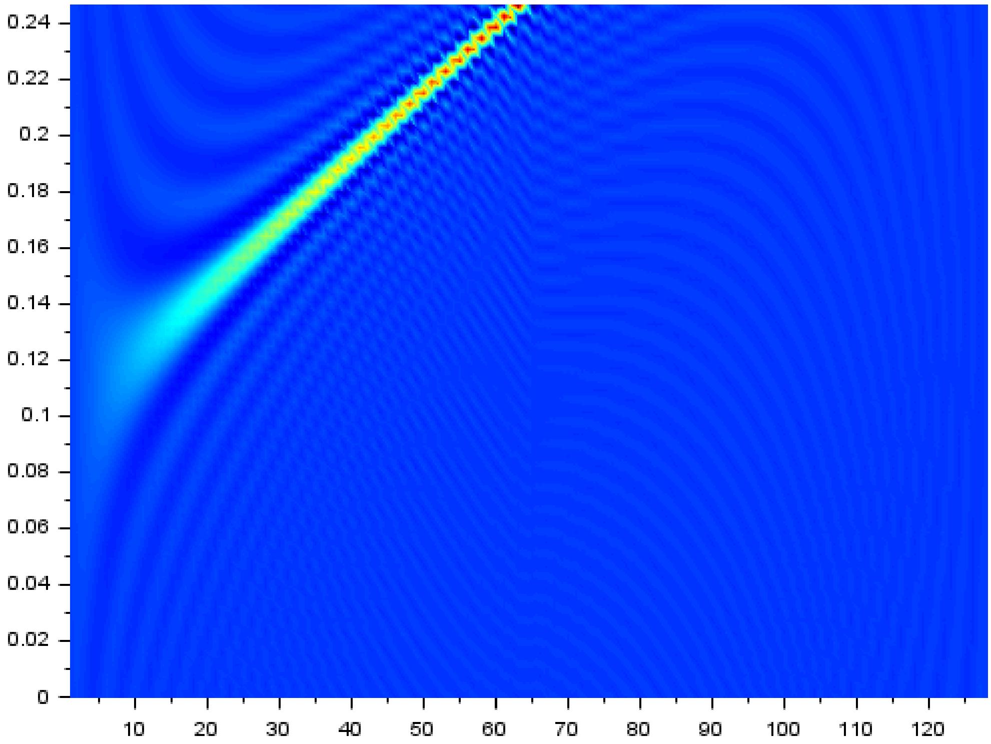
A N vector of normalized frequencies.

Description

tfrwv computes the Wigner-Ville distribution of a discrete-time signal or the cross Wigner-Ville representation between two signals.

Examples

```
sig = fmlin(128,0.1,0.4);
[TFR,T,F] = tfrwv(sig);
clf; gcf().color_map = jetcolormap(128);
Sgrayplot(T,F(1:$/2),TFR(1:$/2,:)');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-August 1994, July 1995.
- Copyright (c) 1996 by CNRS (France).

tfrzam

Zao-Atlas-Marks time-frequency distribution

Calling Sequence

```
[TFR,T,F] = tfrzam(X)
[TFR,T,F] = tfrzam(X, T)
[TFR,T,F] = tfrzam(X, T, N)
[TFR,T,F] = tfrzam(X, T, N, G)
[TFR,T,F] = tfrzam(X, T, N, G, H)
[TFR,T,F] = tfrzam(X, T, N, G, H, TRACE)
[TFR,T,F] = tfrzam(...,'plot')
```

Parameters

X :

A Nx elements vector (auto-ZAM) or a Nx by 2 array signal (cross-ZAM).

T:

a real Nt vector with elements in [1 Nx] : time instant(s) on which the TFR is evaluated (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:NX).

G:

a real vector with odd length: the time smoothing window, (default :Hamming(N/10)).

It will be normalized such as the middle point equals 1.

H :

real vector with odd length: the frequency smoothing window,(default: Hamming(N/4)).

It will be normalized such as the middle point equals 1.

TRACE :

if nonzero,the progression of the algorithm is shown (default : 0).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

F :

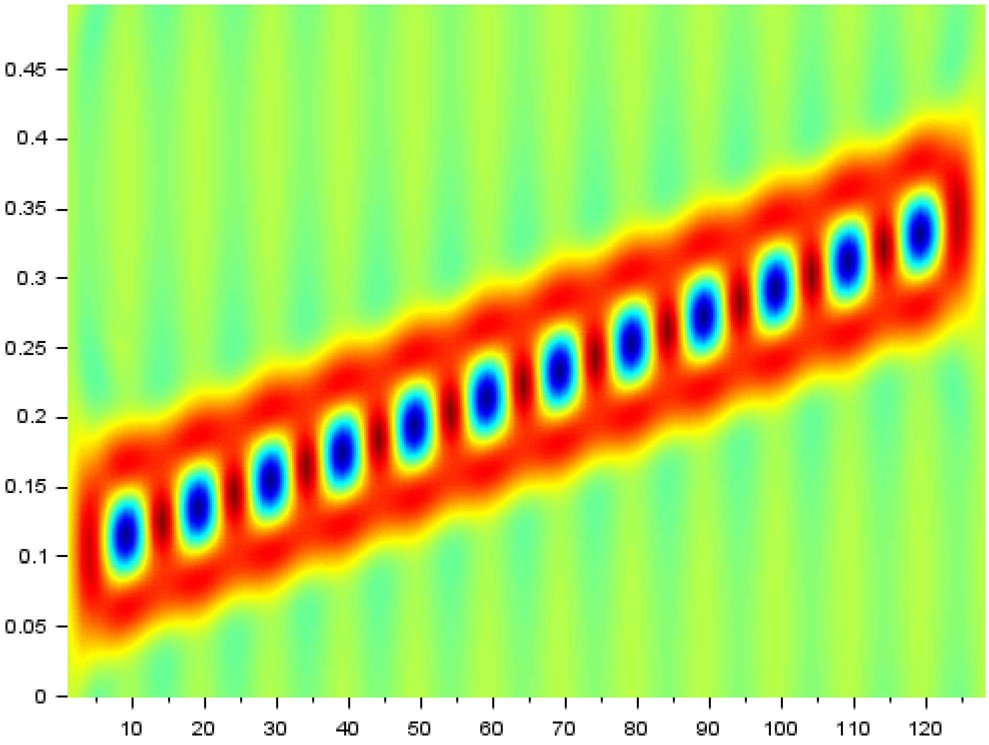
A N vector of normalized frequencies.

Description

tfrzam computes the Zao-Atlas-Marks distribution of a discrete-time signal X, or the cross Zao-Atlas-Marks representation between two signals.

Examples

```
sig = fmlin(128,0.05,0.3) + fmlin(128,0.15,0.4);
g = window("kr",9,3*%pi);
h = window("kr",27,3*%pi);
[TFR,T,F] = tfrzam(sig,1:128,128,g,h);
clf; gcf().color_map = jetcolormap(128);
Sgrayplot(T,F,TFR');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-August 1994, July 1995.
- Copyright (c) 1996 by CNRS (France).

Frequency-Domain Processing

- ffmt — Fast Fourier Mellin Transform
- frpowerspec — Computes the energy spectrum of the signal
- frspec — Computes the spectrum of the signal
- iffmt — Inverse fast Mellin transform
- locfreq — Frequency localization caracteristics
- parafrep — parametric frequency representation of a signal
- sgrpdlay — Group delay estimation of a signal
- tftb_fft — matlab compatible fft
- tftb_ifft — matlab compatible ifft

ffmt

Fast Fourier Mellin Transform

Calling Sequence

```
[MELLIN,BETA] = ffmt(X)
[MELLIN,BETA] = ffmt(X, FMIN, FMAX)
[MELLIN,BETA] = ffmt(X, FMIN, FMAX, N)
```

Parameters

X :

a real vector of size Nx: the signal in time.

FMIN:

a positive scalar in]0 0.5], the normalized lower frequency bound in (Hz) of the analyzed signal. When unspecified, you have to enter it at the command line from the plot of the spectrum.

FMAX :

a positive scalar in]0 0.5], the normalized upper frequency bound (in Hz) of the analyzed signal. When unspecified, you have to enter it at the command line from the plot of the spectrum.

N :

a positive integer: number of analyzed voices. When unspecified, you have to enter it at the command line.

MELLIN :

a complex row vector of size N: the N-points Mellin transform of signal X.

BETA :

a real row vector of size N: the N-points Mellin variable.

The increment between two consecutive elements is constant.

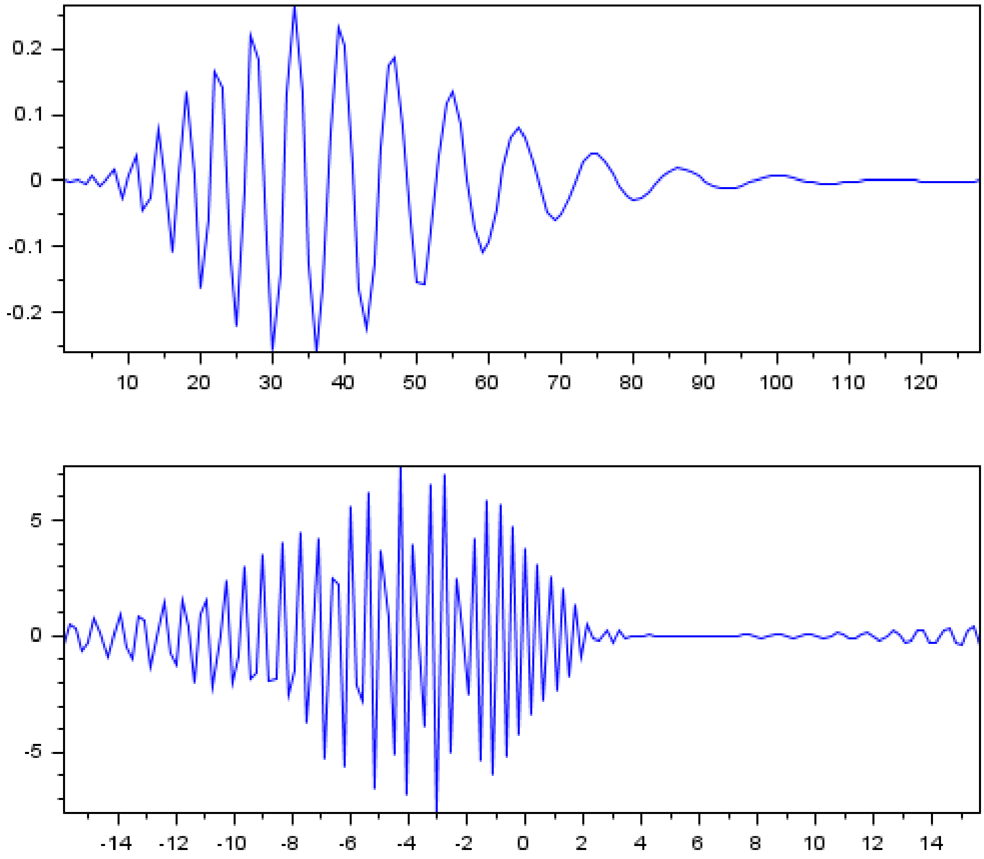
Description

ffmt computes the Fast Mellin Transform of signal X.

Examples

```
sig = altes(128,0.05,0.45);
[MELLIN,BETA] = ffmt(sig,0.05,0.5,148);
clf
```

```
subplot(211); plot(sig)
subplot(212); plot(BETA,real(MELLIN));
```



See also

- iffmt — Inverse fast Mellin transform

Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves 9-95 - O. Lemoine, June 1996.

frpowerspec

Computes the energy spectrum of the signal

Calling Sequence

```
[SP,F] = frpowerspec(X)
[SP,F] = frpowerspec(X, N)
[SP,F] = frpowerspec(X, N, H)
[SP,F] = frpowerspec(...,'plot')
```

Parameters

X :

A Nx elements real vector : the signal in time to be analysed.

N:

a positive integer: the number of frequency bins (default:Nx). For faster computation N should be a power of 2.

H :

real Nx elements vector: the analysis window,(default: window("re",Nx)).

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

SP :

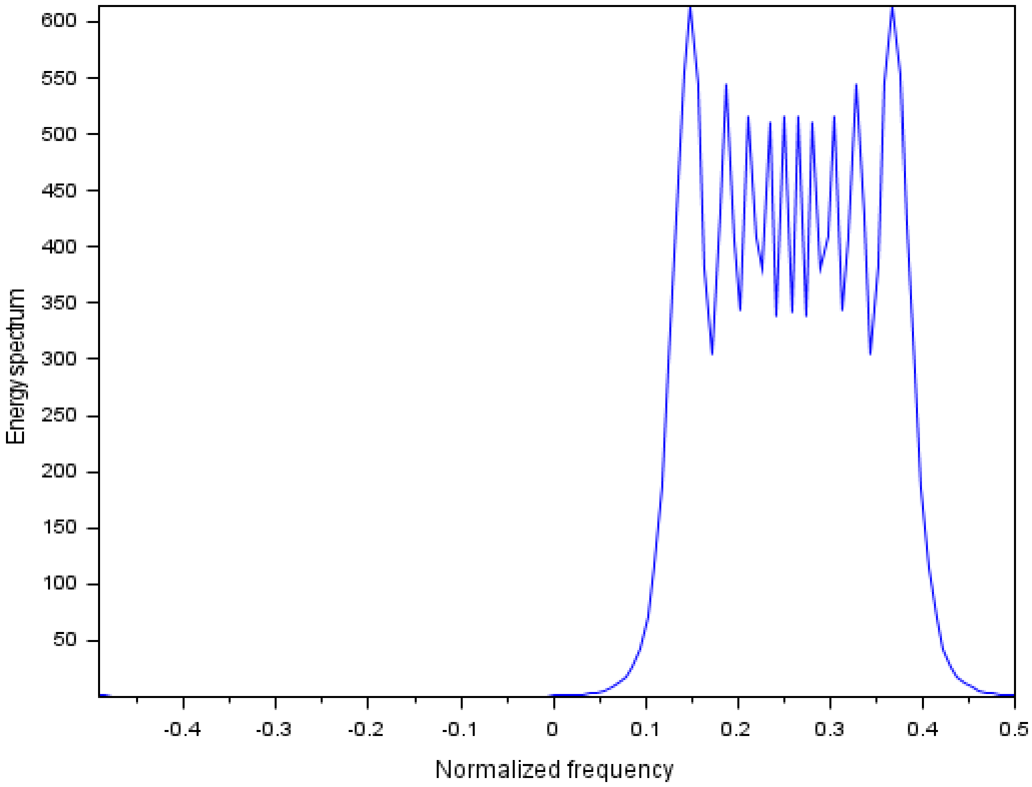
A real column vector of size N: the energy spectrum of the signal.

F :

A real column vector of size N: the normalized frequencies.

Examples

```
sig = fmlin(128,0.1,0.4);
[SP,F] = frpowerspec(sig);
clf; plot(F,SP)
xlabel(_("Normalized frequency"));
ylabel(_("Energy spectrum"));
```



See also

- cspect
- pspect

Authors

- H. Nahrstaedt - Aug 2010

frspec

Computes the spectrum of the signal

Calling Sequence

```
[SP,F] = frspec(X)
[SP,F] = frspec(X, T)
[SP,F] = frspec(X, T, H)
[SP,F] = frspec(...,'plot')
```

Parameters

X :

A Nx elements real vector : the signal in time to be analysed.

N:

a positive integer: the number of frequency bins (default:Nx). For faster computation N should be a power of 2.

H :

real Nx elements vector: the analysis window,(default: window("re",Nx)).

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

SP :

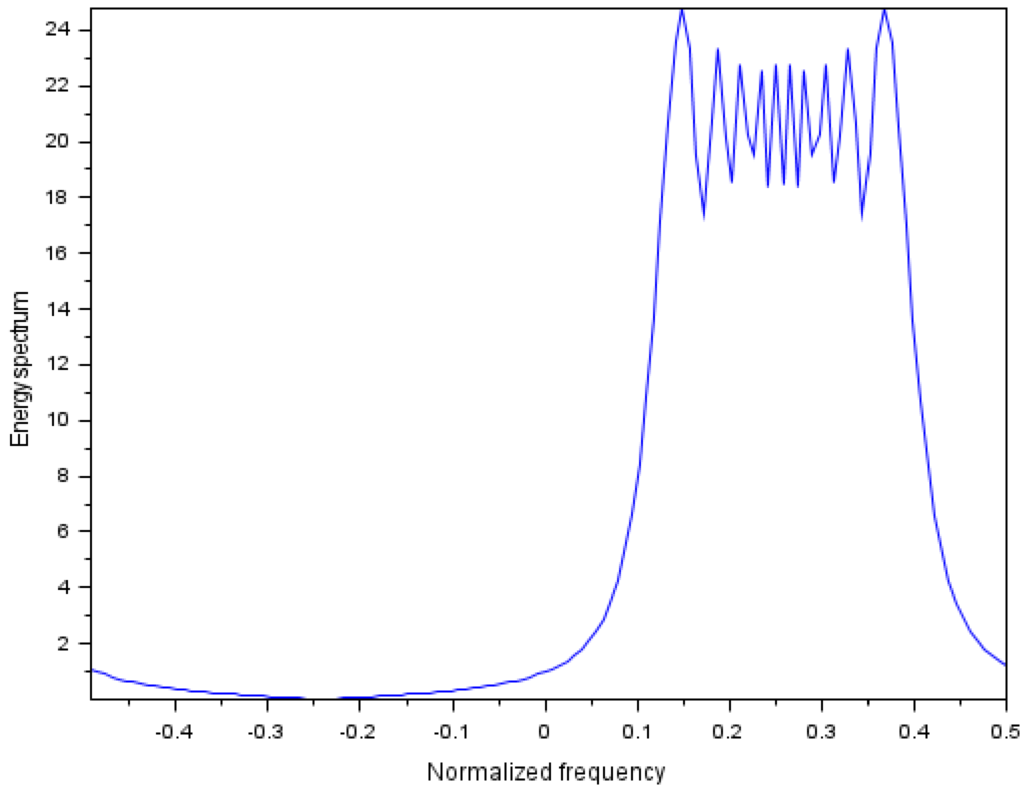
A real column vector of size N: the spectrum of the signal.

F :

A real column vector of size N: the normalized frequencies.

Examples

```
sig = fmlin(128,0.1,0.4);
[SP,F] = frspec(sig);
clf; plot(F,SP)
xlabel(_("Normalized frequency"));
ylabel(_("Energy spectrum"));
```



Authors

- H. Nahrstaedt - Aug 2010

iffmt

Inverse fast Mellin transform

Calling Sequence

```
X = iffmt(MELLIN, BETA)
X = iffmt(MELLIN, BETA, M)
```

Parameters

MELLIN :

a complex vectorof size N: the Mellin transform to be inverted.

Mellin must have been obtained from FMT with frequency running from FMIN to 0.5 Hz.

BETA :

a real vector: the Mellin variable issued from FMT (only the two first elements are used).

M :

a positive integer: the number of points of the inverse Mellin transform. (default : N).

X :

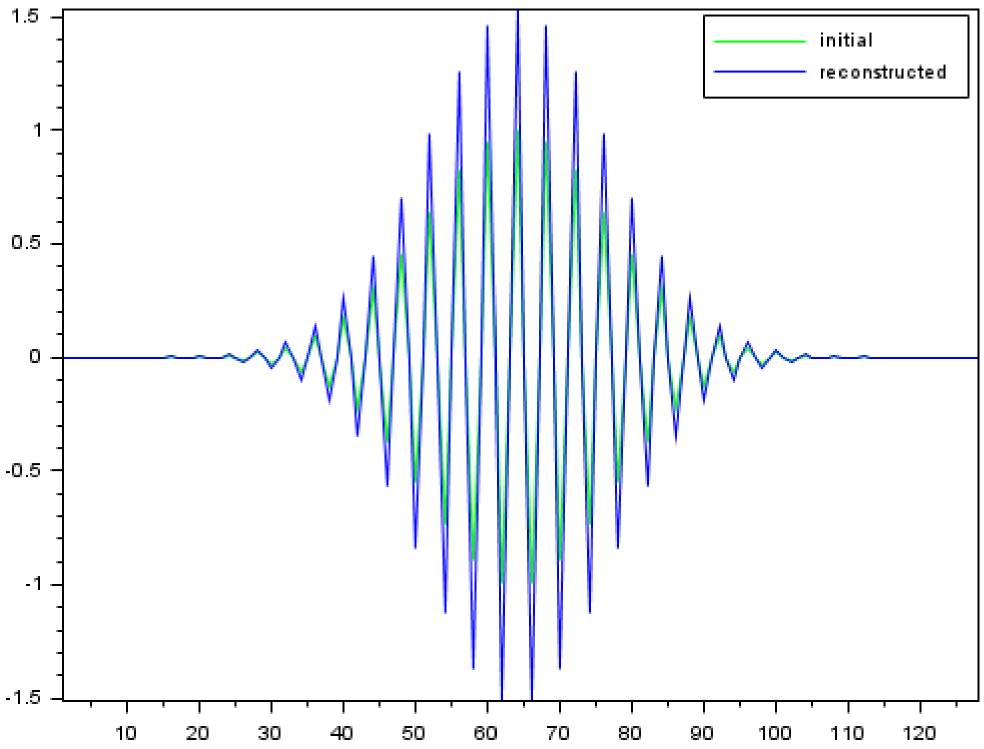
a complex column vector of size M: the inverse Mellin transform.

Description

iffmt computes the inverse fast Mellin transform of MELLIN. WARNING : the inverse of the Mellin transform is correct only if the Mellin transform has been computed from FMIN to 0.5 Hz, and if the original signal is analytic.

Examples

```
sig = atoms(128,[64,0.25,32,1]);
[MELLIN,BETA] = ffmt(real(sig),0.005,0.5,300);

X = iffmt(MELLIN,BETA,128);
clf; plot(real(sig),'g'); plot(real(X));
legend(_("initial"),_("reconstructed"))
```



See also

- ffmt — Fast Fourier Mellin Transform

Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves 9-95 - O. Lemoine, June 1996.

locfreq

Frequency localization caracteristics

Calling Sequence

```
[FM,B] = locfreq(SIG)
```

Parameters

SIG:

a complex vector

FM :

a real scalar: the central frequency

B :

a real scalar: the frequency spreading.

Description

locfreq computes the frequency localization caracteristics of signal SIG.

Examples

```
z = amgauss(160,80,50);
[tm,T] = loctime(z),
[fm,B] = locfreq(z),B*T
```

See also

- loctime — Time localization caracteristics

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, July 1995.
- Copyright (c) 1996 by CNRS (France).

parafrep

parametric frequency representation of a signal

Calling Sequence

```
[spec,freqs] = parafrep(Rx)
[spec,freqs] = parafrep(Rx, N)
[spec,freqs] = parafrep(Rx, N, method)
[spec,freqs] = parafrep(...,'plot')
```

Parameters

Rx :

A (p+1) by (p+1) array of double: the correlation matrix.

N :

a positive integer: the number of frequency bins.

method :

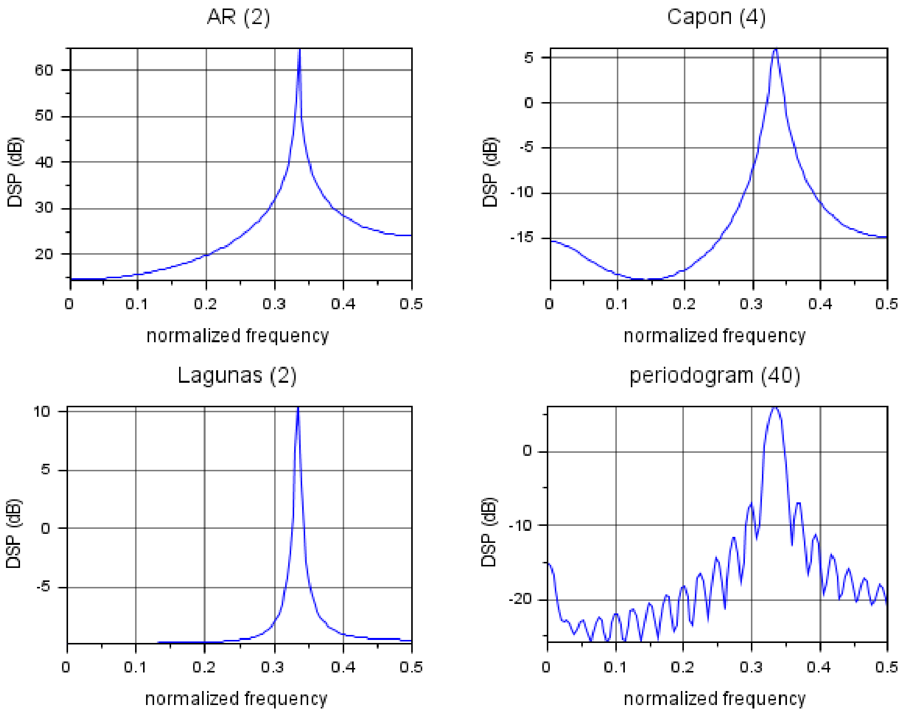
a character string with possible values: 'AR', 'PERIODOGRAM', 'CAPON', 'CAPNORM', 'LAGUNAS', or 'GENLAG'.

'plot':

if one input parameter is 'plot', the frequency representation will be plotted.

Examples

```
noise = rand(1000,1);
signal = filter([1 0 0],[1 1 1],noise);
clf
subplot(221); parafrep(correlmx(signal,2,'hermitian'),128,'AR',"plot");title('AR
(2)');
subplot(222);
parafrep(correlmx(signal,4,'hermitian'),128,'Capon',"plot");title('Capon (4)');
subplot(223);
parafrep(correlmx(signal,2,'hermitian'),128,'lagunas',"plot");title('Lagunas (2)');
subplot(224);
parafrep(correlmx(signal,40,'hermitian'),128,'periodogram',"plot");title('periodogram
(40)');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, july 1998, april 99.

sgrpdlay

Group delay estimation of a signal

Calling Sequence

```
[GD,FNORM] = sgrpdlay(X)
[GD,FNORM] = sgrpdlay(X, FNORM)
```

Parameters

X :

signal in the time-domain.

FNORM :

a real vector with elements in [-0.5 0.5]: the normalized frequencies. By default, FNORM is a linearly spaced vector between -0.5 and 0.5 with length(X) elements.

GD :

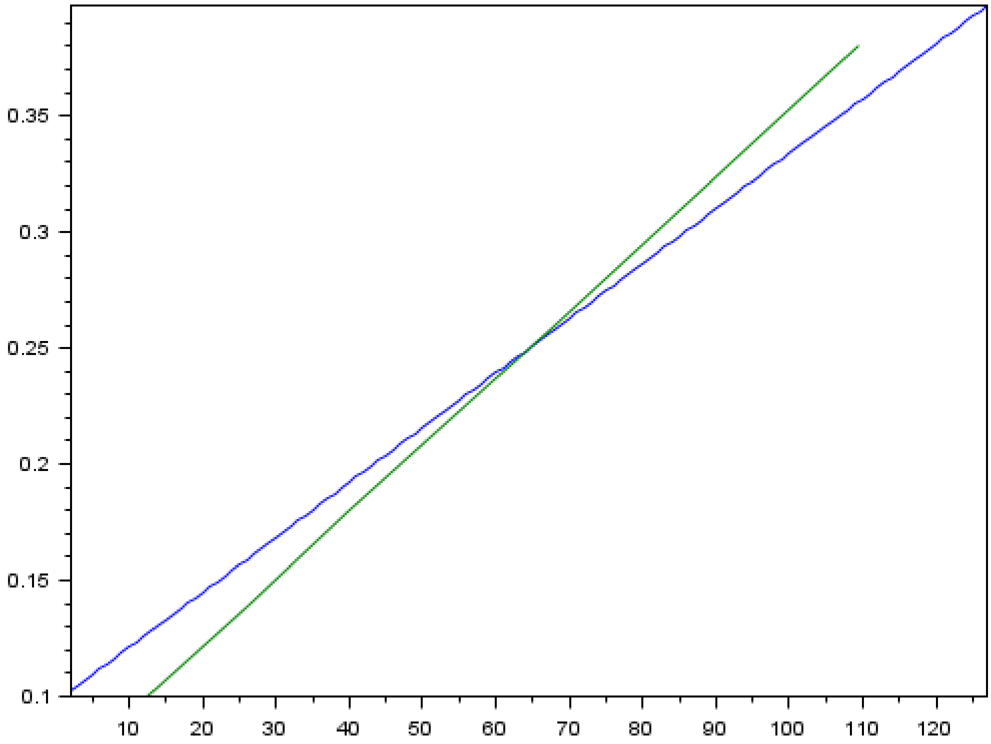
computed group delay. When GD equals zero, it means that the estimation of the group delay for this frequency was outside the interval [1 xrow], and therefore meaningless.

Description

sgrpdlay estimates the group delay of signal X at the normalized frequency(ies) FNORM.

Examples

```
N = 128;
x = amgauss(N,64,30).*fmlin(N,0.1,0.4);
fnorm = 0.1:0.04:0.38;
gd = sgrpdlay(x,fnorm);
t = 2:N-1;
instf = instfreq(x,t);
clf; plot(t,instf',gd,fnorm');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, March 1994, July 1995.
- Copyright (c) 1996 by CNRS (France).

tftb_fft

matlab compatible fft

Calling Sequence

```
y = tftb_fft(x, n)
```

Description

This function is much (100x) faster then mtlb_fft

Compute the FFT of A using subroutines from FFTW. The FFT is calculated along the first non-singleton dimension of the array. Thus if A is a matrix, `fft (A)' computes the FFT for each column of A.

If called with two arguments, N is expected to be an integer specifying the number of elements of A to use, or an empty matrix to specify that its value should be ignored. If N is larger than the dimension along which the FFT is calculated, then A is resized and padded with zeros. Otherwise, if N is smaller than the dimension along which the FFT is calculated, then A is truncated.

See also

- tftb_ifft — matlab compatible ifft

Authors

- H. Nahrstaedt - Aug 2010

tftb_ifft

matlab compatible ifft

Calling Sequence

```
y = tftb_ifft(x, n)
```

Description

This funcion is much (100x) faster then mtlb_ifft

Compute the IFFT of A using subroutines from FFTW. The IFFT is calculated along the first non-singleton dimension of the array. Thus if A is a matrix, `ifft (A)' computes the IFFT for each column of A.

If called with two arguments, N is expected to be an integer specifying the number of elements of A to use, or an empty matrix to specify that its value should be ignored. If N is larger than the dimension along which the FFT is calculated, then A is resized and padded with zeros. Otherwise, if N is smaller than the dimension along which the IFFT is calculated, then A is truncated.

See also

- tftb_fft — matlab compatible fft

Authors

- H. Nahrstaedt - Aug 2010

Choice of the Instantaneous Frequency

- fmconst — Signal with constant frequency modulation
- fmhyp — Signal with hyperbolic frequency modulation
- fmlin — Signal with linear frequency modulation
- fmodany — Signal with arbitrary frequency modulation
- fmpar — Parabolic frequency modulated signal
- fmpower — Signal with power-law frequency modulation
- fmsin — Signal with sinusoidal frequency modulation
- gdpower — Signal with power-law group delay
- if2phase — Generate the phase from the instantaneous frequency

fmconst

Signal with constant frequency modulation

Calling Sequence

```
[Y,IFLAW] = fmconst(N, FNORM, T0)
```

Parameters

N :

a positive integer: the number of points.

FNORM :

a real scalar in [-0.5 0.5]: the normalized frequency (default: 0.25)

T0 :

an integer in [1 N]: the time center (default: round(N/2)).

Y :

a complex column vector of size N: the signal.

IFLAW :

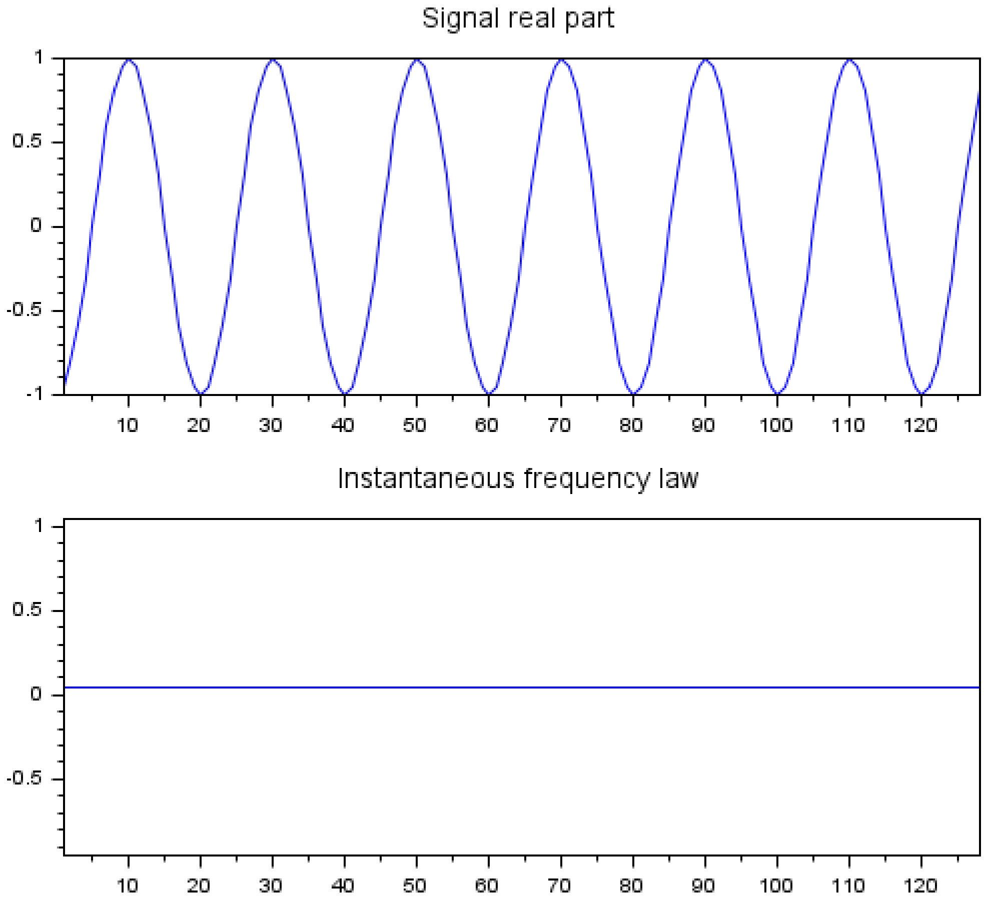
a real column vector of size N: the instantaneous frequency law (optional).

Description

fmconst generates a frequency modulation with a constant frequency fnorm. The phase of this modulation is such that y(t0)=1.

Examples

```
[X,IFLAW] = fmconst(128,0.05,50);
clf
subplot(211); plot(real(X)); xtitle(_("Signal real
part"))
subplot(212); plot(IFLAW); xtitle(_("Instantaneous
frequency law"))
```



See also

- fmlin — Signal with linear frequency modulation
- fmsin — Signal with sinusoidal frequency modulation
- fmodany — Signal with arbitrary frequency modulation
- fmhyp — Signal with hyperbolic frequency modulation
- fmpar — Parabolic frequency modulated signal
- fmpower — Signal with power-law frequency modulation

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, July 1995.

fmhyp

Signal with hyperbolic frequency modulation

Calling Sequence

```
[X,IFLAW] = fmhyp(N, P1, P2)
```

Parameters

N :

a positive integer: number of points in time

P1 :

a real 2 elements vector:

if the number of input arguments is 2, P1 is a vector containing the two coefficients [F0 C] for an hyperbolic instantaneous frequency .

if the number of input arguments is 3, P1 is a time-frequency point of the form [t,f]. where t is in seconds and f is a normalized frequency (between 0 and 0.5).

P2 :

a real 2 elements vector:

P2 is a time-frequency point of the form [t,f]. where t is in seconds and f is a normalized frequency (between 0 and 0.5).

X :

real column vector: the modulated signal time samples. Sampling frequency is set to 1.

IFLAW :

real column vector: instantaneous frequency law

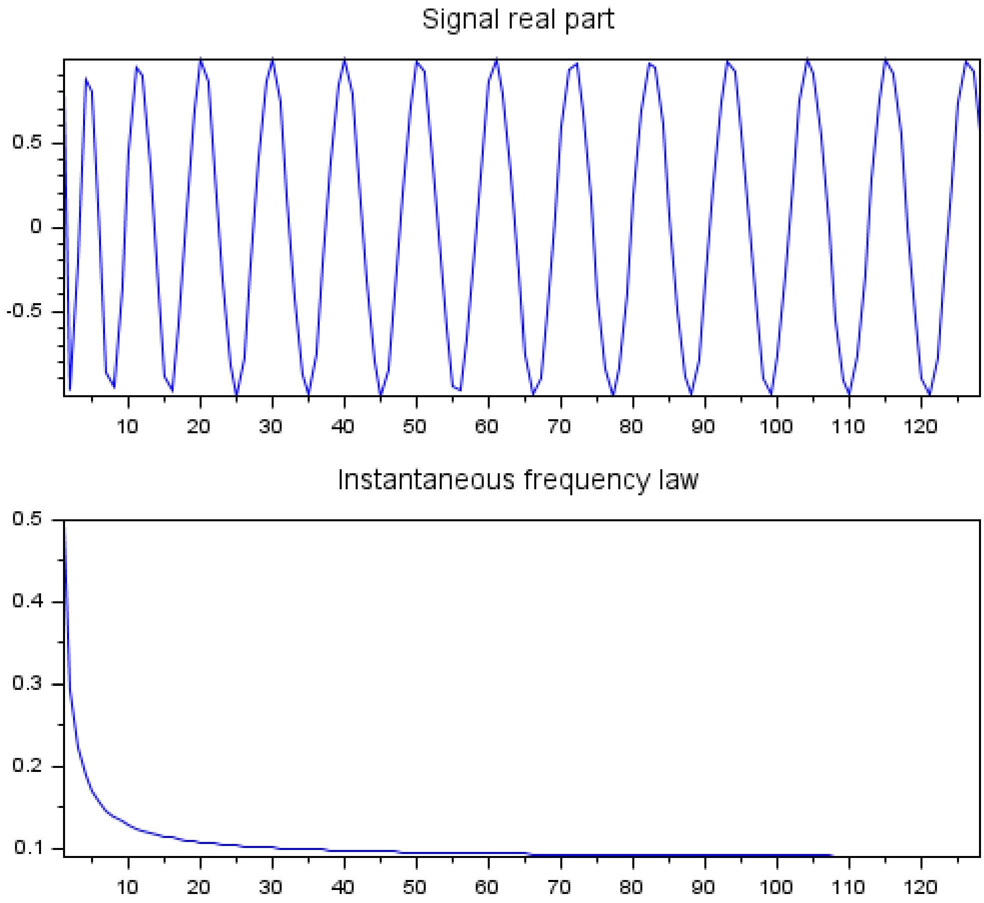
Description

fmhyp generates a signal with hyperbolic frequency modulation : X(t) = exp(i.2.pi(F0.t + C/log|t|)). When called with 3 input arguments F0 and C are derived from P1 and P2 such that the frequency modulation law fits the points P1 and P2.

Examples

```
[X,IFLAW] = fmhyp(128,[1 .5],[32 0.1]);
clf
subplot(211); plot(real(X)); xtitle(_("Signal real
part"))
```

```
subplot(212); plot(IFLAW); xtitle(_("Instantaneous
frequency law"))
```



See also

- fmlin — Signal with linear frequency modulation
- fmsin — Signal with sinusoidal frequency modulation
- fmpar — Parabolic frequency modulated signal
- fmconst — Signal with constant frequency modulation
- fmodany — Signal with arbitrary frequency modulation
- fmpower — Signal with power-law frequency modulation

Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves - October 1995, O. Lemoine - November 1995

fmlin

Signal with linear frequency modulation

Calling Sequence

```
[Y,IFLAW] = fmlin(N)
[Y,IFLAW] = fmlin(N, FNORMI)
[Y,IFLAW] = fmlin(N, FNORMI, FNORMF)
[Y,IFLAW] = fmlin(N, FNORMI, FNORMF, T0)
```

Parameters

N :

a positive integer: the number of points.

FNORMI :

a real scalar in [-0.5 0.5]: the initial normalized frequency (default: 0)

FNORMF :

a real scalar in [-0.5 0.5]: the final normalized frequency (default: 0.5)

T0 :

an integer in [1 N]: the time center (default: round(N/2)).

Y :

a complex column vector of size N: the signal.

IFLAW :

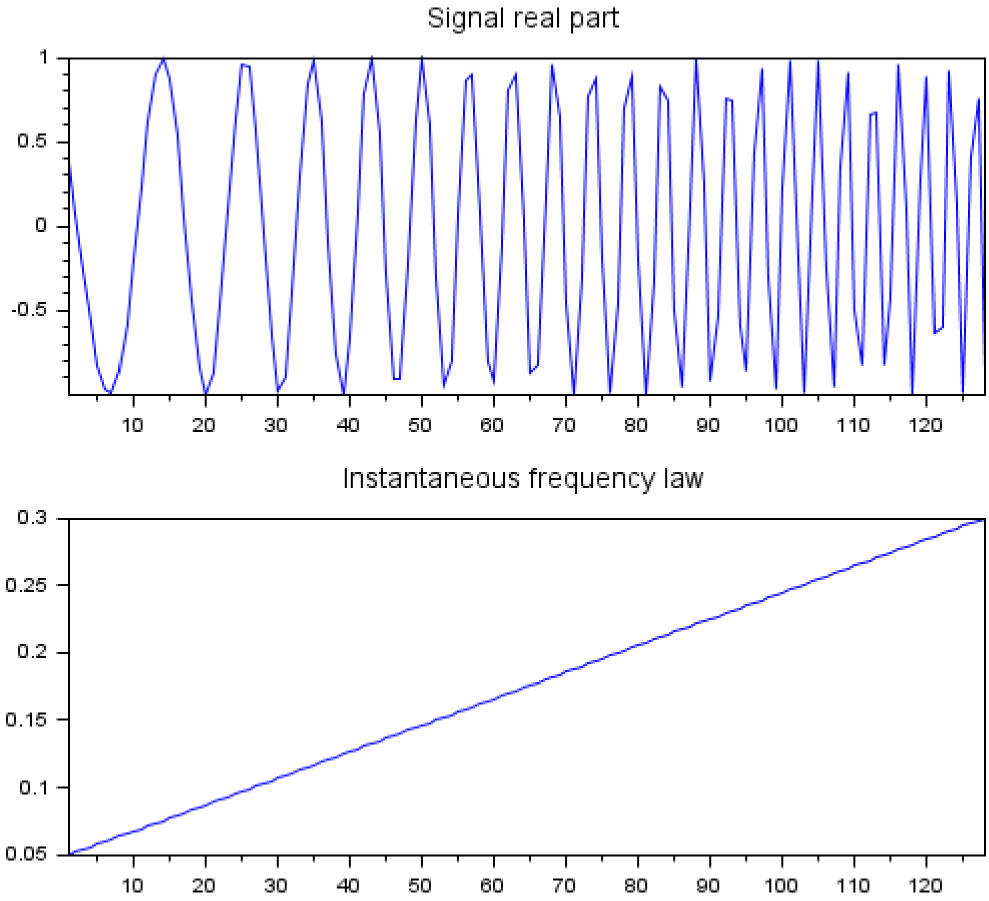
a real column vector of size N: the instantaneous frequency law (optional).

Description

fmlin generates a linear frequency modulation. The phase of this modulation is such that Y(T0)=1.

Examples

```
[X,IFLAW] = fmlin(128,0.05,0.3,50);
clf
subplot(211); plot(real(X)); xtitle(_("Signal real
part"))
subplot(212); plot(IFLAW); xtitle(_("Instantaneous
frequency law"))
```



See also

- fmhyp — Signal with hyperbolic frequency modulation
- fmsin — Signal with sinusoidal frequency modulation
- fmpar — Parabolic frequency modulated signal
- fmconst — Signal with constant frequency modulation
- fmodany — Signal with arbitrary frequency modulation
- fmpower — Signal with power-law frequency modulation

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, July 1995.

fmodany

Signal with arbitrary frequency modulation

Calling Sequence

```
[Y,IFLAW] = fmodany(IFLAW)
[Y,IFLAW] = fmodany(IFLAW, T0)
```

Parameters

IFLAW :

real vector of length N: the instantaneous frequency law samples.

T0 :

an integer in [1 N]: the time reference (default: 1).

Y :

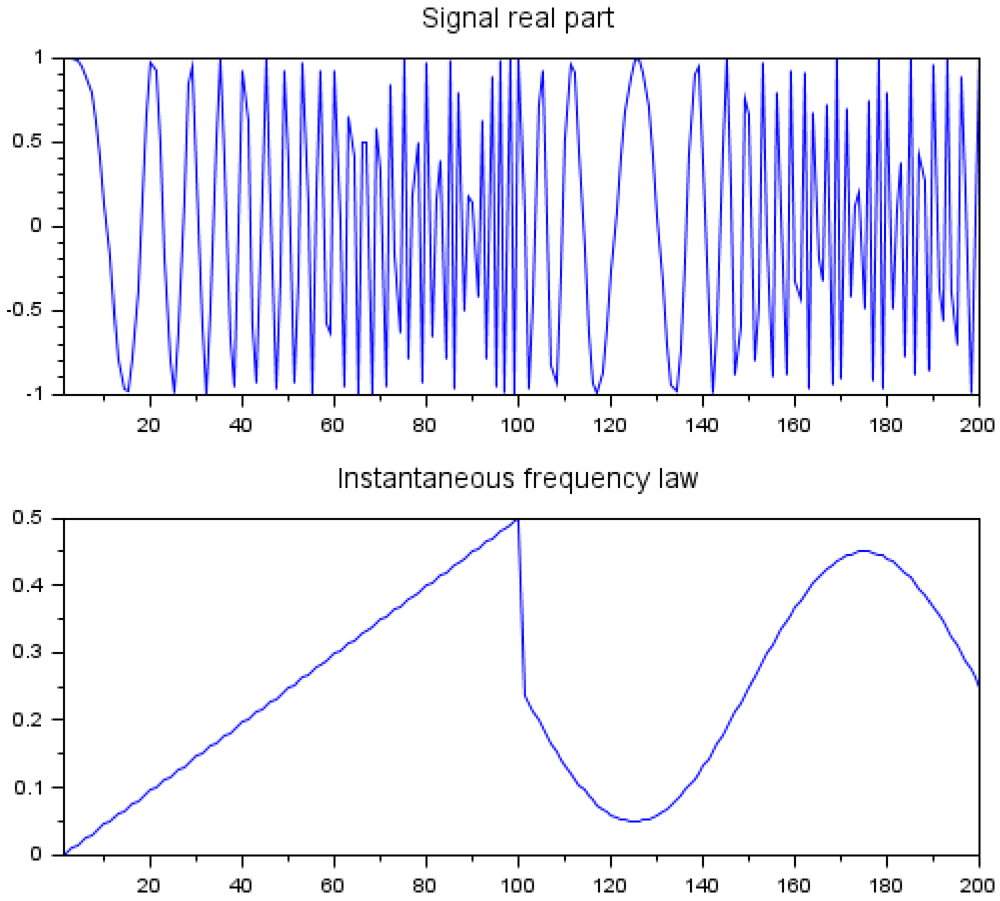
a complex column vector of size N: the output signal

Description

fmodany generates a frequency modulated signal whose instantaneous frequency law is approximately given by the vector IFLAW (the integral is approximated by CUMSUM). The phase of this modulation is such that y(t0)=1.

Examples

```
[y1,ifl1] = fmlin(100);
[y2,ifl2] = fmsin(100);
iflaw = [ifl1;ifl2];
sig = fmodany(iflaw);
clf
subplot(211); plot(real(sig)); xtitle(_("Signal real
part"))
subplot(212); plot(iflaw); xtitle(_("Instantaneous
frequency law"))
```



See also

- fmhyp — Signal with hyperbolic frequency modulation
- fmsin — Signal with sinusoidal frequency modulation
- fmpar — Parabolic frequency modulated signal
- fmconst — Signal with constant frequency modulation
- fmlin — Signal with linear frequency modulation
- fmpower — Signal with power-law frequency modulation

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, August 1995.

fmpar

Parabolic frequency modulated signal

Calling Sequence

```
[X,IFLAW] = fmpar(N, P1)
[X,IFLAW] = fmpar(N, P1, P2)
[X,IFLAW] = fmpar(N, P1, P2, P3)
```

Parameters

N :

a positive integer: number of points in time

P1 :

a real vector of size 2 or 3

if the number of input arguments is 2, P1 is a vector containing the three coefficients [A0 A1 A2] of the polynomial instantaneous phase.

if the number of input arguments is 3, P1 is a time-frequency point of the form [t,f]. where t is in seconds and f is a normalized frequency (between 0 and 0.5).

P2 :

a real 2 elements vector:

P2 is a time-frequency point of the form [t,f]. where t is in seconds and f is a normalized frequency (between 0 and 0.5).

P3 :

a real 2 elements vector:

P3 is a time-frequency point of the form [t,f]. where t is in seconds and f is a normalized frequency (between 0 and 0.5).

X :

real column vector: the modulated signal time samples. Sampling frequency is set to 1.

IFLAW :

real column vector: instantaneous frequency law

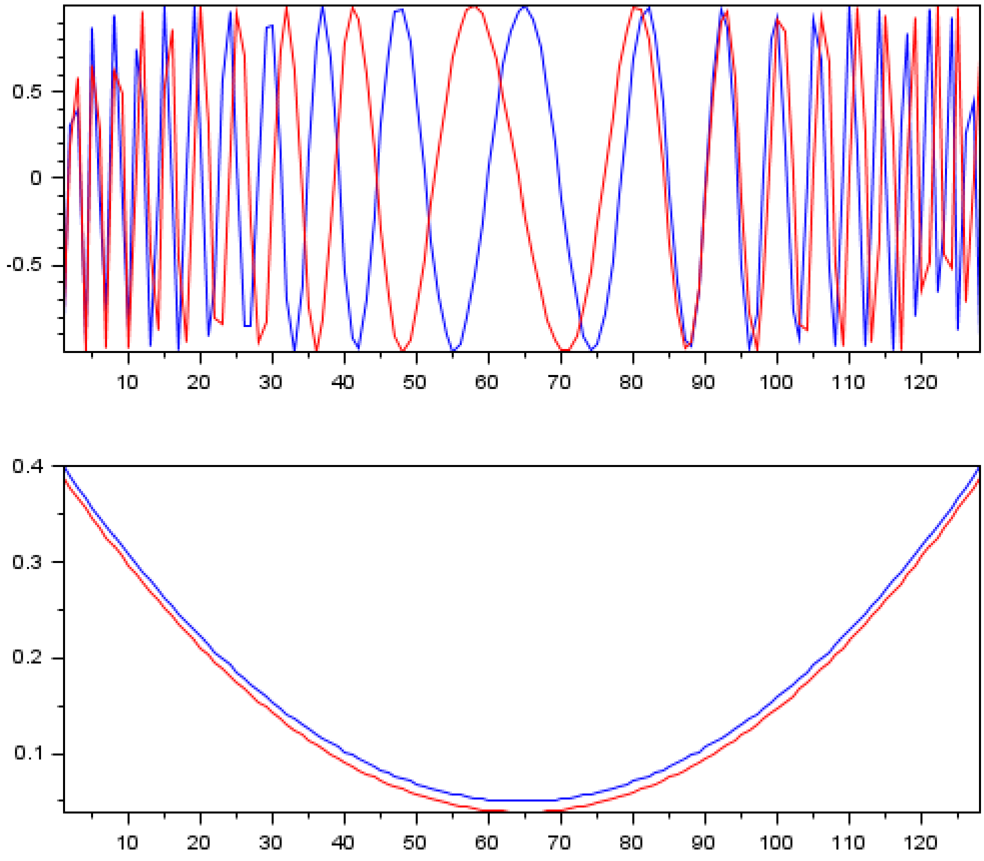
Description

fmpar generates a signal with parabolic frequency modulation law. X(T) = exp(j*2*pi(A0.T + A1/2.T^2 +A2/3.T^3)). When called with 4 input arguments F0 and C are derived from

P1, P2 and P3 such that the frequency modulation law fits the points P1, P2 and P3.

Examples

```
[X1,IFLAW1] = fmpar(128,[1 0.4],[64 0.05],[128 0.4]);
[X2,IFLAW2] = fmpar(128,[0.4 -0.0112 8.6806e-05]);
subplot(211); plot(real(X1),"b"); plot(real(X2),"r")
subplot(212); plot(IFLAW1,"b"); plot(real(IFLAW2),"r")
```



See also

- fmhyp — Signal with hyperbolic frequency modulation
- fmsin — Signal with sinusoidal frequency modulation
- fmodany — Signal with arbitrary frequency modulation
- fmconst — Signal with constant frequency modulation
- fmlin — Signal with linear frequency modulation
- fmpower — Signal with power-law frequency modulation

Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves - October 1995, O. Lemoine - November 1995

fmpower

Signal with power-law frequency modulation

Calling Sequence

```
[X,IFLAW] = fmpower(N, K, P1)
[X,IFLAW] = fmpower(N, K, P1, P2)
```

Parameters

N :

a positive integer: number of points in time

K :

a real scalar degree of the power-law (K~=1)

P1 :

a real 2 elements vector:

if the number of input arguments is 2, P1 is a vector containing the two coefficients [F0 C] for an hyperbolic instantaneous frequency .

if the number of input arguments is 3, P1 is a time-frequency point of the form [t,f]. where t is in seconds and f is a normalized frequency (between 0 and 0.5).

P2 :

a real 2 elements vector:

P2 is a time-frequency point of the form [t,f]. where t is in seconds and f is a normalized frequency (between 0 and 0.5).

X :

real column vector: the modulated signal time samples. Sampling frequency is set to 1.

IFLAW :

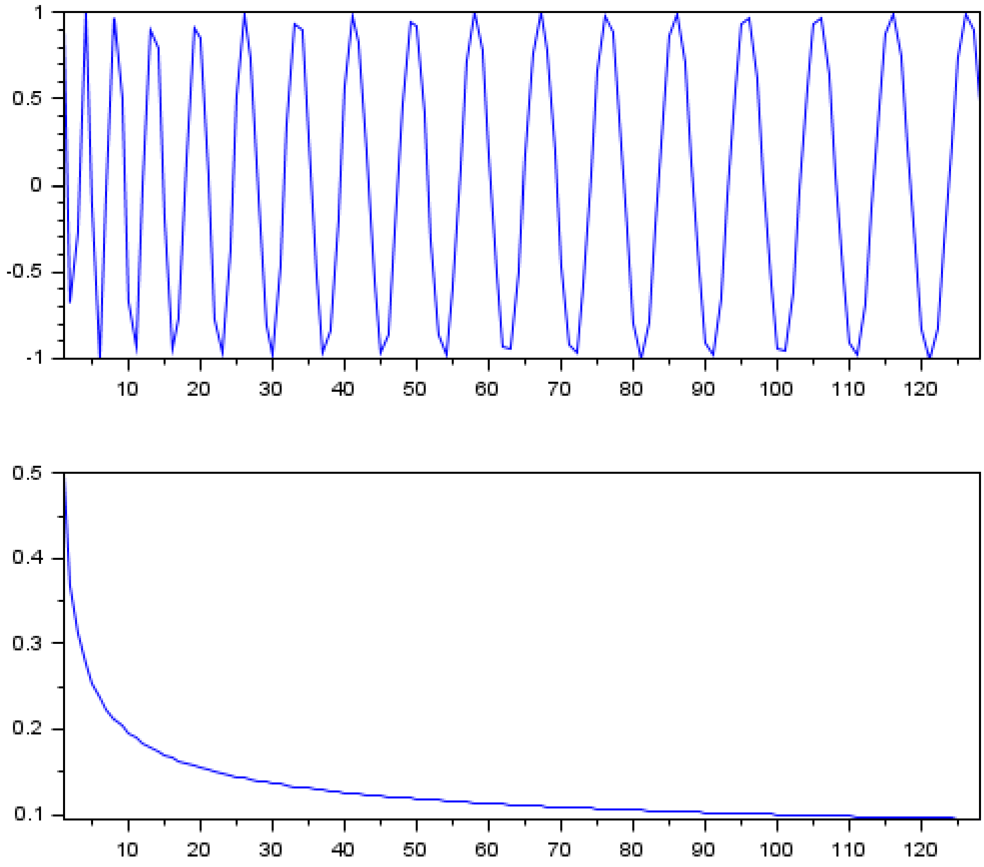
real column vector: instantaneous frequency law

Description

fmpower generates a signal with a power-law frequency modulation. X(t) = exp(j*2*pi(F0*t + C/(1-K)*abs(t).^(1-K))). When called with 4 input arguments F0 and C are derived from P1 and P2 such that the frequency modulation law fits the points P1 and P2.

Examples

```
[X,IFLAW] = fmpower(128,0.5,[1 0.5],[100 0.1]);
clf
subplot(211); plot(real(X));
subplot(212); plot(IFLAW);
```



See also

- fmhyp — Signal with hyperbolic frequency modulation
- fmsin — Signal with sinusoidal frequency modulation
- fmodany — Signal with arbitrary frequency modulation
- fmconst — Signal with constant frequency modulation
- fmlin — Signal with linear frequency modulation
- fmpar — Parabolic frequency modulated signal

Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves - October 1995, O. Lemoine - April 1996.

fmsin

Signal with sinusoidal frequency modulation

Calling Sequence

```
[Y,IFLAW] = fmsin(N)
[Y,IFLAW] = fmsin(N, FNORMIN)
[Y,IFLAW] = fmsin(N, FNORMIN, FNORMAX)
[Y,IFLAW] = fmsin(N, FNORMIN, FNORMAX, PERIOD)
[Y,IFLAW] = fmsin(N, FNORMIN, FNORMAX, PERIOD, T0)
[Y,IFLAW] = fmsin(N, FNORMIN, FNORMAX, PERIOD, T0,
FNORM0)
[Y,IFLAW] = fmsin(N, FNORMIN, FNORMAX, PERIOD, T0,
FNORM0, PM1)
```

Parameters

N :

a positive integer: number of points in time

FNORMIN :

a real scalar in [-0.5 0.5]: the smallest normalized frequency (default: 0.05)

FNORMAX :

a real scalar in [-0.5 0.5]: the highest normalized frequency (default: 0.45)

PERIOD :

a positive scalar: the period of the sinusoidal fm (default: N)

T0 :

an integer in [1 N]: the time reference for the phase(default: round(N/2)).

FNORM0 :

a real scalar in [-0.5 0.5]: the highest normalized frequency (default: (FNORMAX+FNORMIN)/2)

PM1 :

an integer value in {-1 +1}: the frequency direction at T0 (default: +1)

Y :

a complex column vector of size N: the signal.

IFLAW :

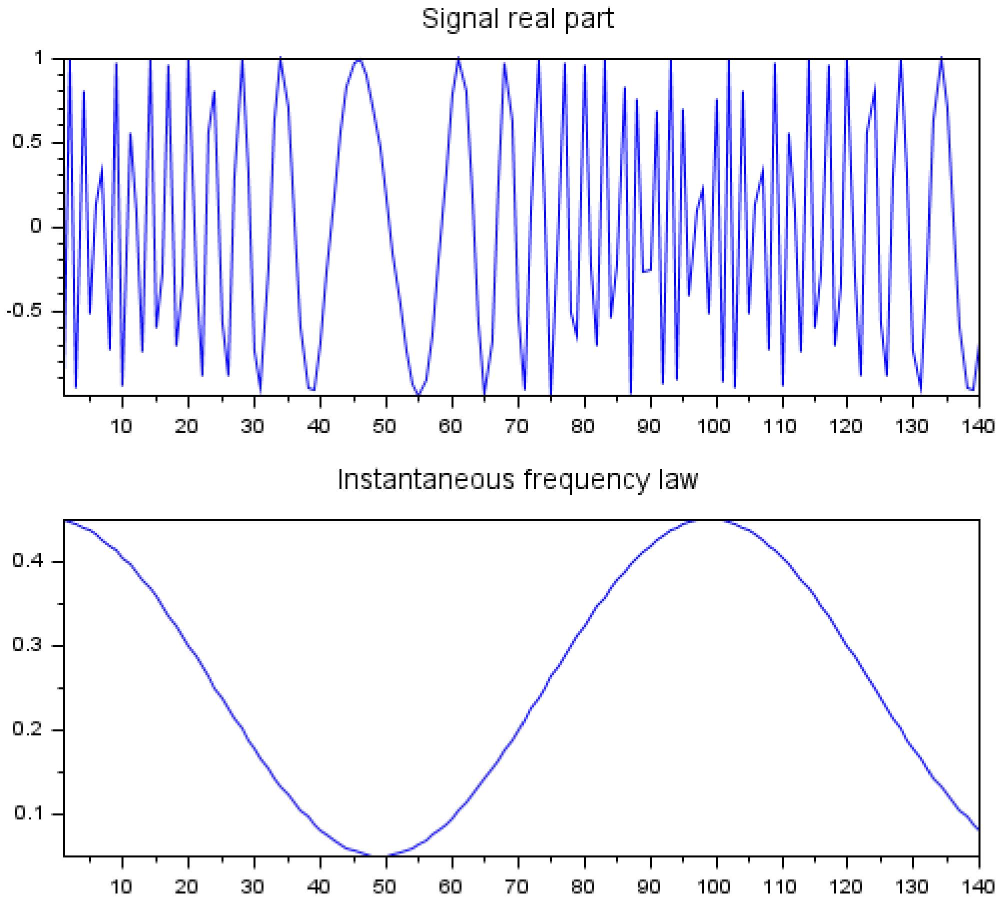
a real column vector of size N: the instantaneous frequency law (optional).

Description

fmsin generates a sinusoidal frequency modulation. This modulation is designed such that the instantaneous frequency at time T0 is equal to FNORM0, and the ambiguity between increasing or decreasing frequency is solved by PM1.

Examples

```
[X,IFLAW] = fmsin(140,0.05,0.45,100,20,0.3,-1.0);
clf
subplot(211); plot(real(X)); xtitle(_("Signal real part"))
subplot(212); plot(IFLAW); xtitle(_("Instantaneous frequency law"))
```



See also

- fmhyp — Signal with hyperbolic frequency modulation
- fmpower — Signal with power-law frequency modulation
- fmodany — Signal with arbitrary frequency modulation
- fmconst — Signal with constant frequency modulation
- fmlin — Signal with linear frequency modulation
- fmpar — Parabolic frequency modulated signal

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, July 1995.

gdpower

Signal with power-law group delay

Calling Sequence

```
[X,GPD,F] = gspower(N, K, C)
```

Parameters

N :

an even positive integer: number of points in time

K :

a real scalar: the degree of the power-law (default : 0)

C :

a real non zero scalar: the rate-coefficient of the power-law group delay. (default : 1)

X :

a real column vector: the modulated signal samples

GPD :

a real row vector (length : round(N/2)-1): the group delay (length : round(N/2)-1)

F :

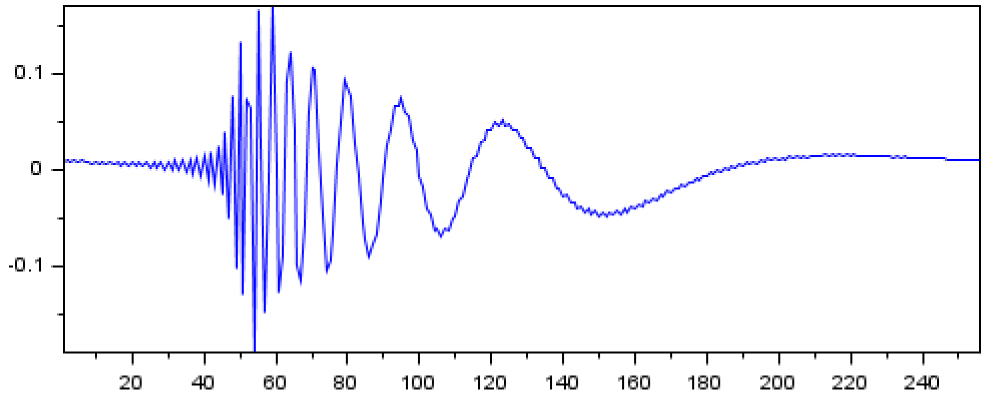
a real row vector (length : round(N/2)-1): the frequency bins

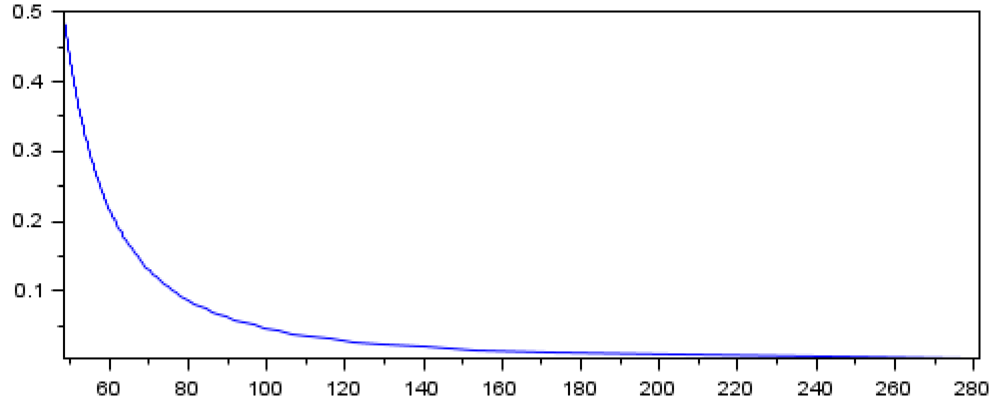
Description

generates a signal with a power-law group delay of the form tx(f) = T0 + C*f^(K-1). The output signal is of unit energy.

Examples

```
[sig,gpd,f] = gdpower(256,1/2);
clf
subplot(211); plot(sig);
subplot(212); plot(gpd,f);
```





See also

- fmpower — Signal with power-law frequency modulation

Authors

- H. Nahrstaedt - Aug 2010
- O. Lemoine - April, July 1996

if2phase

Generate the phase from the instantaneous frequency

Calling Sequence

```
phi = if2phase(iflaw)
```

Parameters

iflaw:

A real vector of size N: the instantaneous frequency array in Hz.

phi:

A real column vector of size N: the phase array in radians .

Authors

- H. Nahrstaedt - Aug 2010
- O. Lemoine - February 1996.

Linear Time-Frequency Processing

- tffilter — Time frequency filtering of a signal
- tfrgabor — Gabor representation of a signal
- tfristft — Inverse Short time Fourier transform
- tfrstft — Short time Fourier transform
- tfrsurf — extract from a time-frequency representation the biggest energy dots

tffilter

Time frequency filtering of a signal

Calling Sequence

```
Y = tffilter(TFR, X, T)
Y = tffilter(TFR, X, T, TRACE)
```

Parameters

TFR :

a M by N array: the Wigner-Ville distribution of the filter frequency axis is graduated from 0.0 to 0.5.

X :

a complex vector of size N: the input signal (must be analytic).

T :

a real vector with elements in [1 N]: the time instant(s)(default : 1:length(X)).

TRACE :

if nonzero, the progression of the algorithm is shown (default : 0).

Y:

a vector with same sizes as X: the filtered signal.

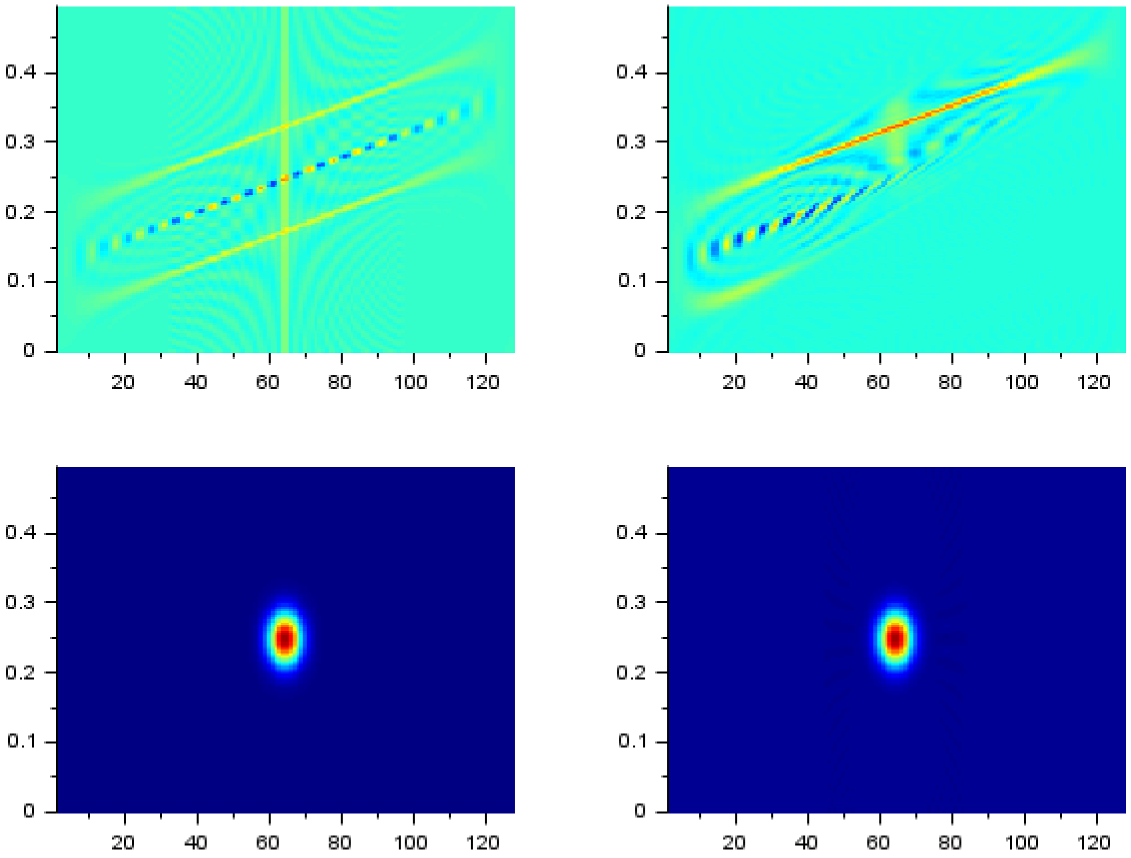
Description

tffilter filters the signal X with a non stationary filter.

Examples

```
clf; gcf().color_map = jetcolormap(128);

Nt = 128;
t = 1:Nt;
sig = fmlin(Nt,0.05,0.3) + fmlin(Nt,0.2,0.45);
sig(Nt/2) = sig(Nt/2)+8;
[TFR,T,F] = tfrwv(sig,t);
subplot(221); grayplot(T,F,TFR');

Nf = 128; freqs = 0.5*(0:Nf-1).'/Nf;
H = [];
for tloop = 1:Nt,
    rate = 0.2*(tloop-1)/(Nt-1);
    H(:,tloop) = (0+rate<freqs).*(freqs<0.1+rate);
```

```
end
y = tffilter(H,sig,t);
[TFR,T,F] = tfrwv(y,t);
subplot(222); grayplot(T,F,TFR');

Nt = 128;
t = 1:Nt;
sig = atoms(128,[64 0.25 round(sqrt(128)) 1],0);
[TFR,T,F] = tfrwv(sig,t);
subplot(223); grayplot(T,F,TFR');

Nf = 64;
H = zeros(Nf,Nt);H(Nf/4+(-15:15),Nt/2+(-15:15))=ones(31);
y = tffilter(H,sig,t);
[TFR,T,F] = tfrwv(y,t);
subplot(224); grayplot(T,F,TFR');
```



See also

- tfristft — Inverse Short time Fourier transform

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, Jan 1997.

tfrgabor

Gabor representation of a signal

Calling Sequence

```
[TFR,DGR,GAM] = tfrgabor(SIG, N, Q)
[TFR,DGR,GAM] = tfrgabor(SIG, N, Q, H)
[TFR,DGR,GAM] = tfrgabor(SIG, N, Q, H, TRACE)
[TFR,DGR,GAM] = tfrgabor(...,'plot')
```

Parameters

SIG :

signal to be analyzed (length(SIG)=N1).

N :

number of Gabor coefficients in time (N1 must be a multiple of N) (default : divider(N1)).

Q :

degree of oversampling ; must be a divider of N (default : Q=divider(N)).

H :

synthesis window, which was originally chosen as a Gaussian window by Gabor. Length(H) should be as closed as possible from N, and must be >=N (default : Gauss(N+1)). H must be of unit energy, and CENTERED.

TRACE :

if nonzero, the progression of the algorithm is shown (default : 0).

'plot':

if one input parameter is 'plot', tfrgabor runs tfrqview. and TFR will be plotted

TFR :

a N1 by N1 real matrix: the Square modulus of the Gabor coefficients.

DGR :

a N1 by N1 complex matrix: the Gabor coefficients.

GAM :

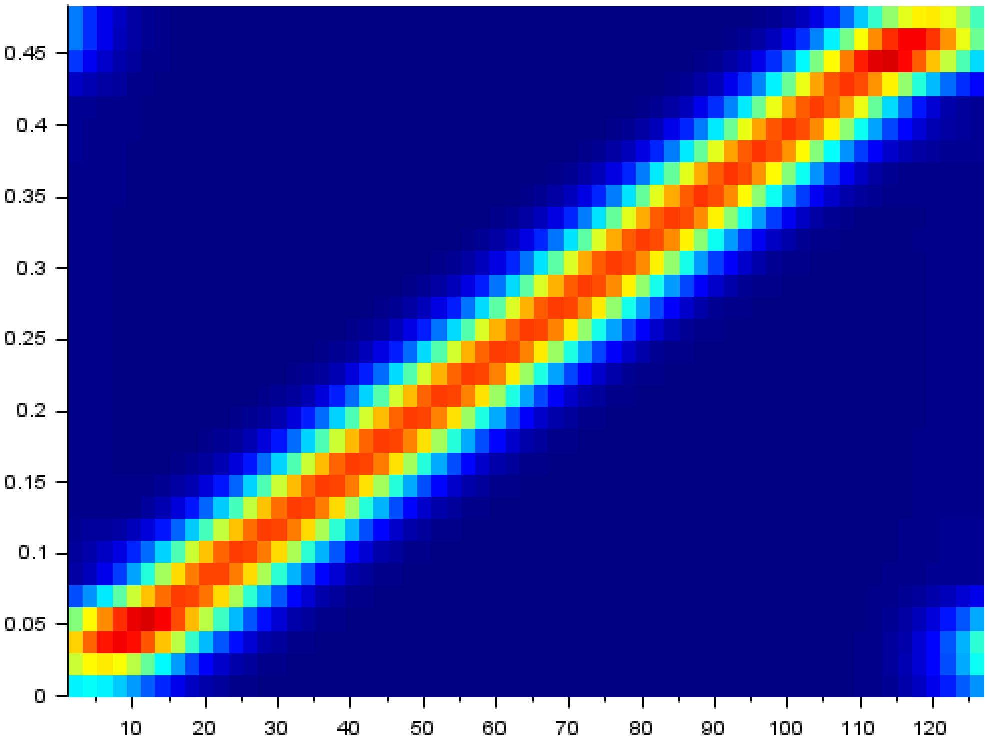
a column vector of size N: the biorthogonal (dual frame) window associated to H.

Description

tfrgabor computes the Gabor representation of signal X, for a given synthesis window H, on a rectangular grid of size (N,M) in the time-frequency plane. M and N must be such that N1 = M * N / Q where N1=length(X) and Q is an integer corresponding to the degree of oversampling. If Q=1, the time-frequency plane (TFP) is critically sampled, so there is no redundancy in the TFP. If Q>1, the TFP is oversampled, allowing a greater numerical stability of the algorithm.

Examples

```
N1 = 128;
sig = fmlin(N1);
N = N1/2; q = N/2;
[tfr,dgr,gam] = tfrgabor(sig,N,q);
t = 1:2:N1;
nf = size(tfr,1)/2;
f = (0.5*(0:nf-1)/nf);
clf; gcf().color_map = jetcolormap(128);
grayplot(t,f,tfr(1:$/2,:)');
```



Authors

- H. Nahrstaedt - Aug 2010
- O. Lemoine, October 1995 - February 1996.

tfristft

Inverse Short time Fourier transform

Calling Sequence

```
[X,T] = tfristft(tfr, T, H)
[X,T] = tfristft(tfr, T, H, TRACE)
```

Parameters

tfr :

>a N by M complex matrix: the time-frequency representation.

T:

a vector with integer values and increments between elements equal to 1: the time instant(s)(default : 1:length(X)).

H :

a real vector with odd size: the frequency smoothing window, H being normalized so as to be of unit energy.

TRACE :

if nonzero of %t, the progression of the algorithm is shown (default : %f).

X :

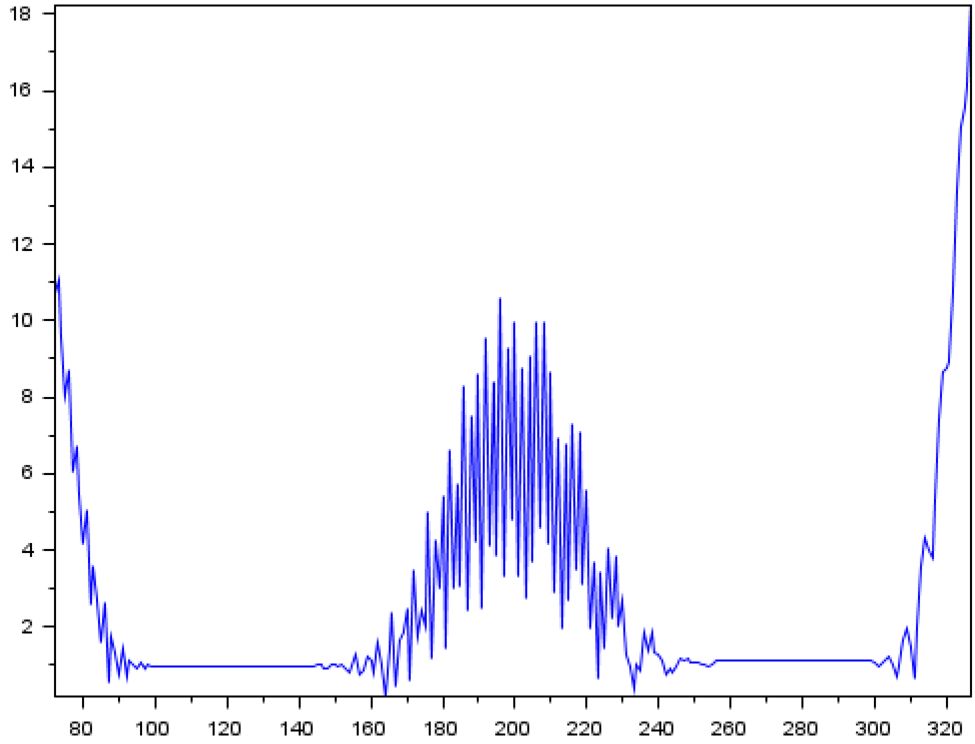
a column vector of size length(t): the signal with the specified time frequency representation.

Description

tfristft computes the inverse short-time Fourier transform of a discrete-time signal X. This function may be used for time-frequency synthesis of signals.

Examples

```
t = 200+(-128:127);
sig = [fmconst(200,0.2) ; fmconst(200,0.4)];
h = window("hm",57);
tfr = tfrstft(sig,t,256,h);
sigsyn = tfristft(tfr,t,h);
plot(t',abs(sigsyn-sig(t)))
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, November 1996.

tfrstft

Short time Fourier transform

Calling Sequence

```
[TFR,T,F] = tfrstft(X)
[TFR,T,F] = tfrstft(X, T)
[TFR,T,F] = tfrstft(X, T, N)
[TFR,T,F] = tfrstft(X, T, N, H
[TFR,T,F] = tfrstft(X, T, N, H, TRACE)
[TFR,T,F] = tfrstft(...,'plot')
```

Parameters

X :

A Nx elements vector: the signal.

T:

a real Nt vector with elements in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:NX). For faster computation N should be a power of 2.

H:

a real vector with odd length: the analysis window, H being normalized so as to be of unit energy. (default : Hamming(N/4)).

TRACE :

if nonzero,the progression of the algorithm is shown (default : 0).

'plot':

if the last input parameter value is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A complex N by Nt array: the time-frequency representation.

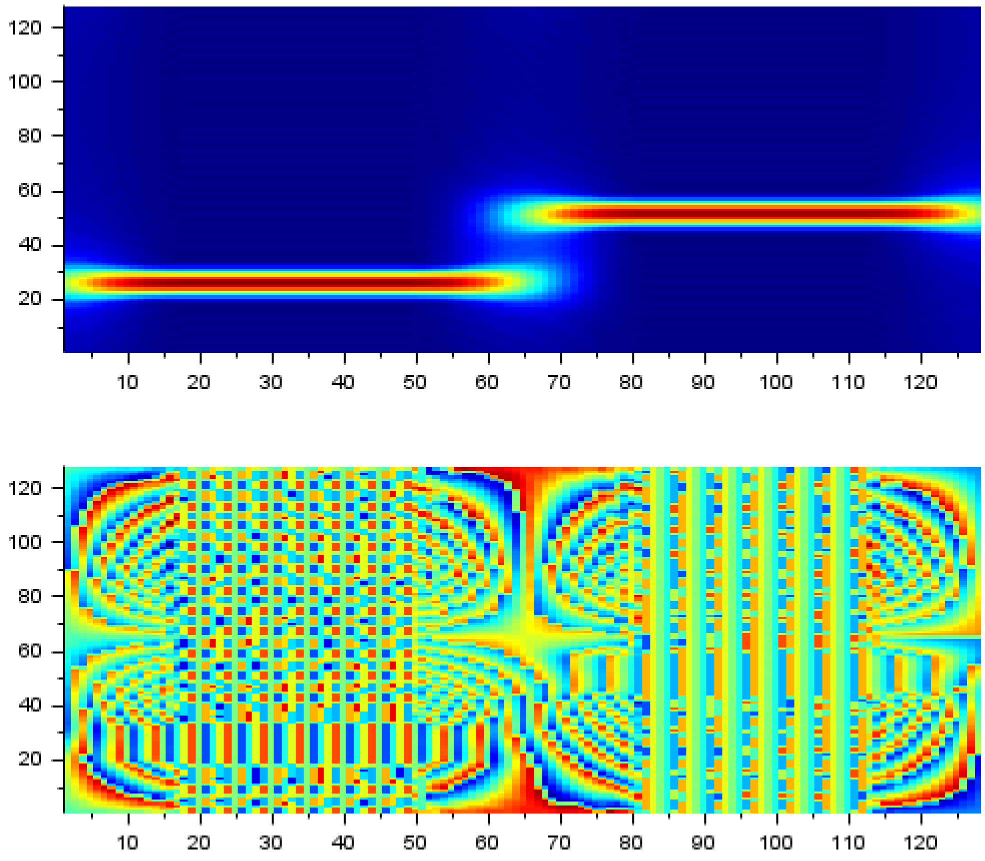
F :

A N vector of normalized frequencies.

Examples

```
sig = [fmconst(64,0.2) ; fmconst(64,0.4)];
tfr = tfrstft(sig);
```

```
clf; gcf().color_map = jetcolormap(128);
subplot(211); grayplot(1:128,1:128,abs(tfr)');
subplot(212);
grayplot(1:128,1:128,atan(imag(tfr),real(tfr))');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-August 1994, July 1995.
- Copyright (c) 1996 by CNRS (France).

tfrsurf

extract from a time-frequency representation the biggest energy dots

Calling Sequence

```
[tfr2,OrderedSurfaces] = tfrsurf(tfr)
[tfr2,OrderedSurfaces] = tfrsurf(tfr, threshold)
[tfr2,OrderedSurfaces] = tfrsurf(tfr, threshold, keep)
[tfr2,OrderedSurfaces] = tfrsurf(tfr, threshold, keep,
trace)
```

Parameters

TFR :

time-frequency representation.

THRESHOLD :

a positive scalar: the energy threshold, in %

KEEP :

a positive integer value: the number of dots to keep

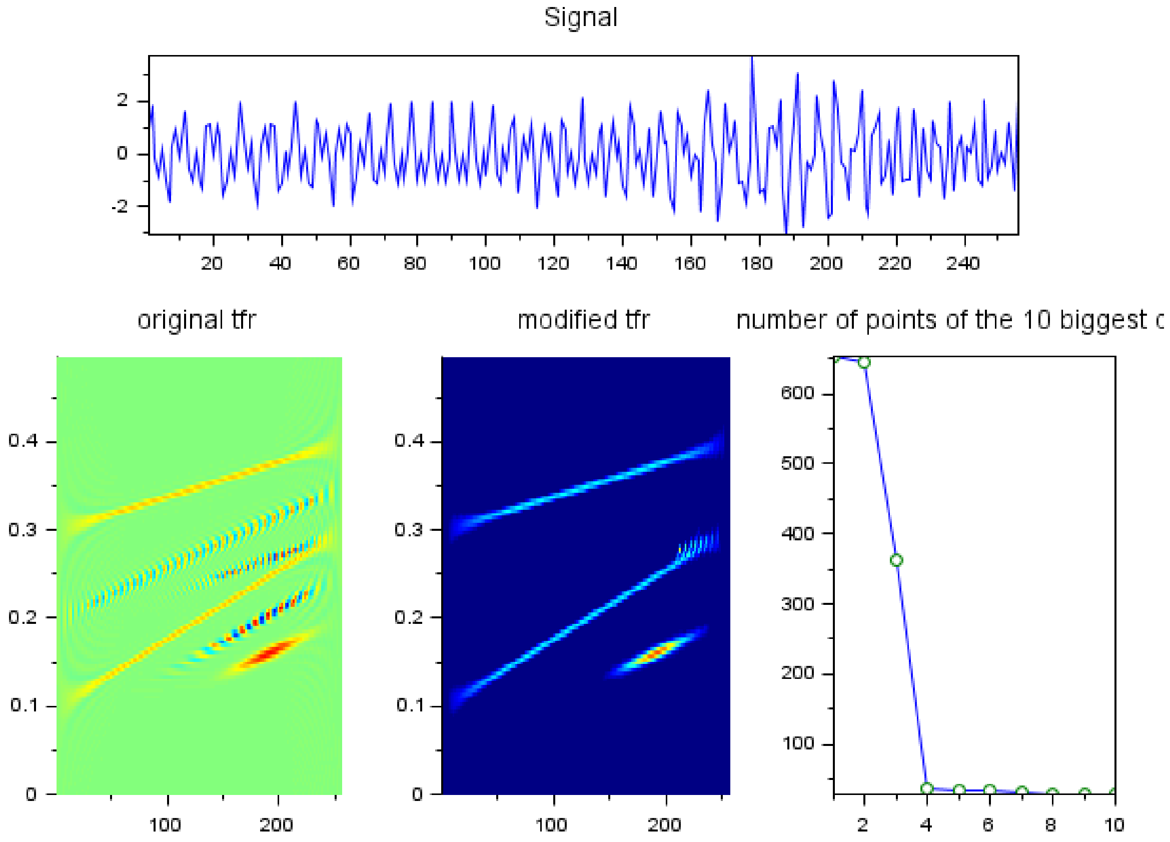
TRACE :

if nonzero, the progression of the algorithm is shown (default : 0).

Examples

```
N = 256;
sig = fmlin(N,0.1,0.3) + fmlin(N,0.3,0.4) +
2*fmlin(N,0.05,0.2).*amgauss(N,190,70);
clf; gcf().color_map = jetcolormap(128);
ax1 = gca();
ax1.axes_bounds = [0,0,1,1/3];ax1.tight_limits="on";
plot(sig);title('Signal')

[tfr,t,f] = tfrwv(sig,1:N,128);
ax2 = newaxes(); ax2.axes_bounds = [0,1/3,1/3,2/3];
ax2.tight_limits = "on";
grayplot(t,f,tfr')
title 'original tfr'

[tfr2,OrderedSurfaces] = tfrsurf(tfr,5,3);
ax3 = newaxes(); ax3.axes_bounds = [1/3,1/3,1/3,2/3];
ax3.tight_limits = "on";
grayplot(t,f,tfr2')
title 'modified tfr'

ax4 = newaxes(); ax4.axes_bounds = [2/3,1/3,1/3,2/3];
ax4.tight_limits = "on";
```

```
plot(1:10,OrderedSurfaces(1:10),'-',1:10,OrderedSurfaces(1:10),'o');
//semilogy
title 'number of points of the 10 biggest dots'
```



See also

- imextract

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, oct 1999
- Copyright (c) CNRS - France 1999.

Modification

- scale — Scale a signal using the Mellin transform
- sigmerge — Add two signals with given energy ratio in dB

scale

Scale a signal using the Mellin transform

Calling Sequence

```
S = scale(X)
S = scale(X, A)
S = scale(X, A, FMIN, FMAX)
S = scale(X, A, FMIN, FMAX, N)
S = scale(X, A, FMIN, FMAX, N, TRACE)
```

Parameters

X :

signal in time to be scaled (Nx=length(X)).

A :

scale factor. A < 1 corresponds to a compression in the time domain. A can be a vector.(default : 2)

FMIN,FMAX :

respectively lower and upper frequency bounds of the analyzed signal. These parameters fix the equivalent frequency bandwidth (expressed in Hz). When unspecified, you have to enter them at the command line from the plot of the spectrum. FMIN and FMAX must be >0 and <=0.5.

N :

number of analyzed voices (default : automatically determined).

TRACE :

if nonzero, the progression of the algorithm is shown (default : 0).

S :

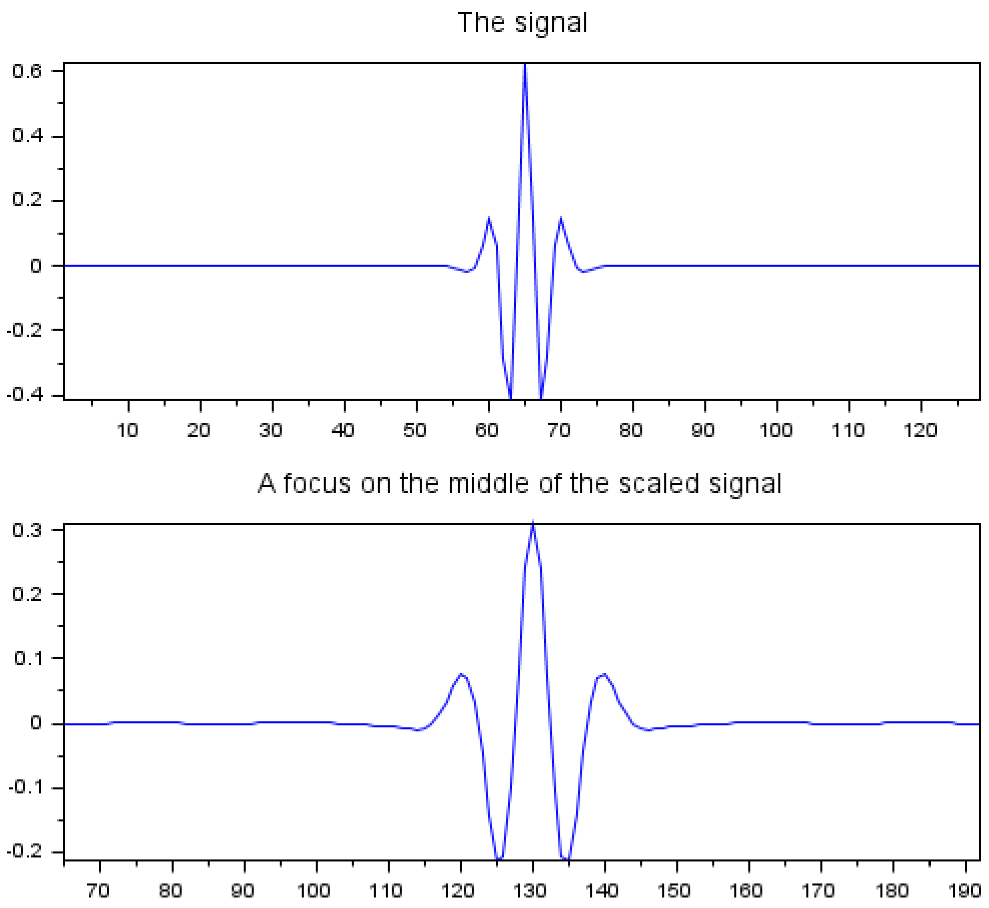
the A-scaled version of signal X. Length of S can be larger than length of X if A > 1. If A is a vector of length L, S is a matrix with L columns. S has the same energy as X.

Description

scale computes the A-scaled version of signal X : A^(-1/2) X(T/A) using its Mellin transform.

Examples

```
N = 128; a = 2;
sig = klauder(N);
S = scale(sig,a,.05,.45,128);
clf
subplot(211);
plot(sig);gca().tight_limits="on";
title("The signal")
subplot(212);
plot(real(S))
//focus on the scaled signal's middle
ax = gca();
ax.data_bounds(1,1) = 1+N/a;
ax.data_bounds(2,1) = a*N-N/a;
ax.tight_limits = "on";
title("A focus on the middle of the scaled signal")
```



Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves, October 1995 - O. Lemoine, June 1996.

sigmerge

Add two signals with given energy ratio in dB

Calling Sequence

```
SIG = sigmerge(X1, X2)
SIG = sigmerge(X1, X2, RATIO)
```

Parameters

X1, X2 :

real or complex vectors of same sizes: the input signals.

RATIO :

a real scalar: the Energy ratio in deciBels (default : 0 dB).

X :

a real or complex vector with same sizes as X1 and X2: the output signal.

Description

sigmerge adds two signals so that a given energy ratio expressed in deciBels is satisfied.

Examples

```
sig = fmlin(64,0.01,0.05,1);
noise = hilbert(rand(64,1,'normal'));
SNR = 15; // dB
x = sigmerge(sig,noise,SNR);
//Check SNR
Esig = mean(abs(sig).^2);
Enoise = mean(abs(x-sig).^2);

10*log10(Esig/Enoise)
```

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, July 1995.

Noise Realizations

- noisecg — Analytic complex gaussian noise
- noisecu — Analytic complex uniform white noise

noisecg

Analytic complex gaussian noise

Calling Sequence

```
NOISE = noisecg(N)
NOISE = noisecg(N, A1)
NOISE = noisecg(N, A1, A2)
```

Parameters

N:

a positive integer: the length of the noise signal.

A1:

a real scalar (see below).

A2:

a real scalar (see below).

NOISE:

a complex column vector of size N: the noise signal.

Description

NOISE=noisecg(N,A1,A2) computes an analytic complex gaussian noise of length N with mean 0.0 and variance 1.0.

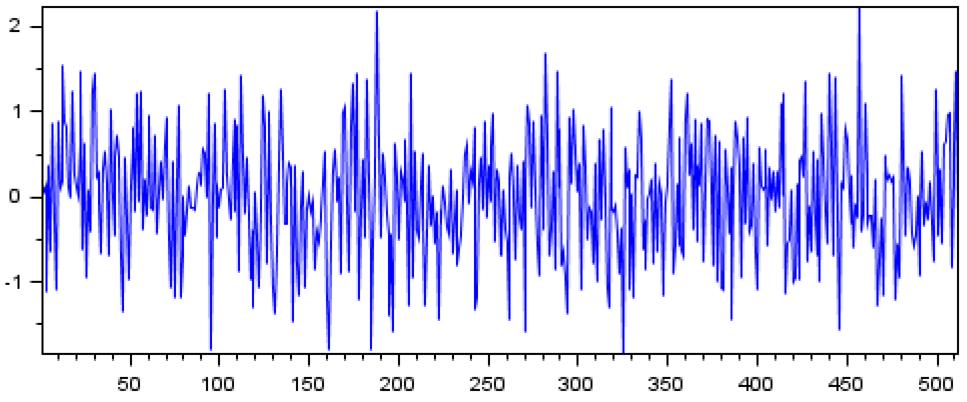
NOISE=noisecg(N) yields a complex white gaussian noise.

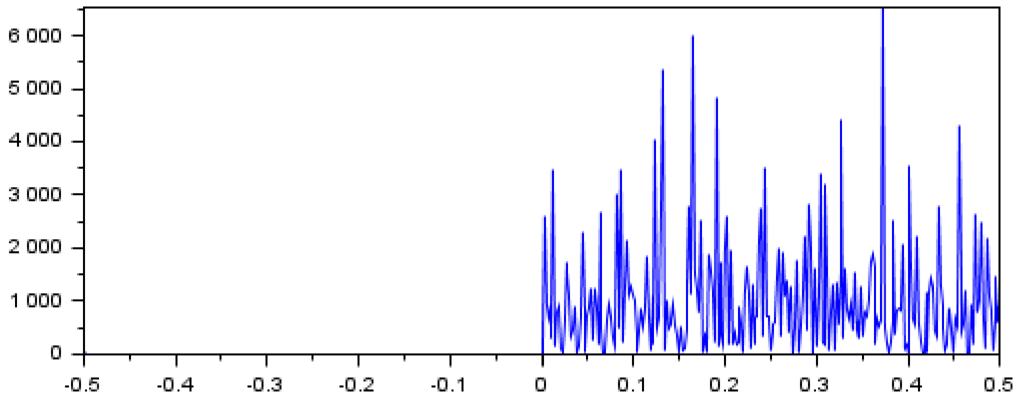
NOISE=noisecg(N,A1) yields a complex colored gaussian noise obtained by filtering a white gaussian noise through a sqrt(1-A1^2)/(1-A1*z^(-1)) first order filter.

NOISE=noisecg(N,A1,A2) yields a complex colored gaussian noise obtained by filtering a white gaussian noise through a sqrt(1-A1^2-A2^2)/(1-A1*z^(-1)-A2*z^(-2)) second order filter.

Examples

```
N = 512;
noise = noisecg(N);
m = mean(noise)
sigma2 = sum((noise-m).^2)
clf
subplot(211); plot(real(noise)); a = gca();
f = linspace(-0.5,0.5,N);
subplot(212); plot(f',abs(fftshift(fft(noise))).^2);
```





See also

- noisecu — Analytic complex uniform white noise

Authors

- H. Nahrstaedt - Aug 2010
- O. Lemoine, June 95/May 96 - F. Auger, August 95.

noisecu

Analytic complex uniform white noise

Calling Sequence

```
NOISE = noisecu(N)
```

Parameters

N:

a positive integer: the length of the noise signal.

NOISE:

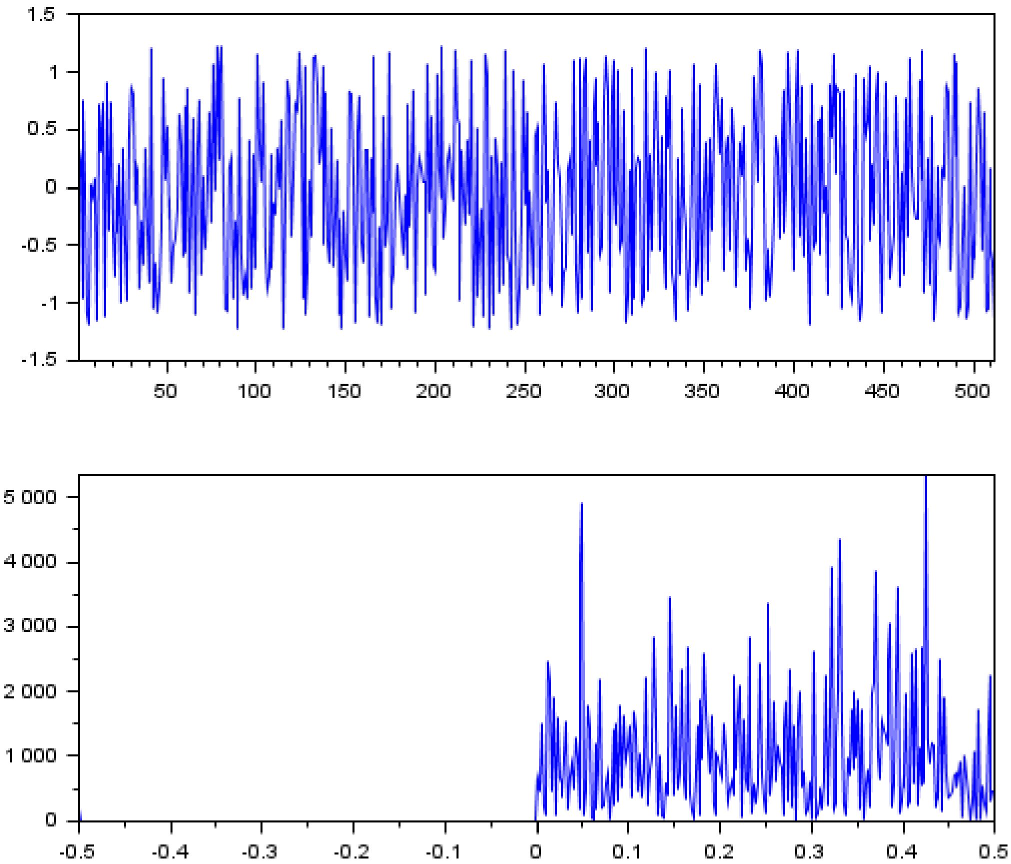
a complex column vector of size N: the noise signal.

Description

noisecu computes an analytic complex white uniform noise of length N with mean 0.0 and variance 1.0.

Examples

```
N = 512; noise = noisecu(N);
m = mean(noise)
sigma2 = sum((noise-m).^2)
subplot(211); plot(real(noise)); a=gca(); a.data_bounds =
([1 N -1.5 1.5]);
f = linspace(-0.5,0.5,N);
subplot(212); plot(f,abs(fftshift(fft(noise))).^2);
```



See also

- noisecg — Analytic complex gaussian noise

Authors

- H. Nahrstaedt - Aug 2010
- O. Lemoine, June 95/May 96 - F. Auger, August 95.

Other

- contwtgn — computes a continuous wavelet transform
- contwtgnmir — Continuous wavelet transform of mirrored 1-D signals
- correlmx — correlation matrix of a signal
- d2statio — Distance to stationarity
- disprog — Display progression of a loop
- divider — Find dividers of an integer
- dwindow — Derive a window
- fzero — solves the scalar nonlinear equation such that F(X) == 0
- gaussn — generates the order n derivative of the gaussian window, centered at frequency f0
- imextrac — imextrac(Image) extract and isolate dots in a binary image
- integ — Approximate 1D integral of a discrete signal
- integ2d — Approximate 2-D integral
- istfr1 — returns true is method is a time frequency representation of type 1 (frequencies >0 or <0)
- istfr2 — returns true is method is a time frequency representation of type 2 (only frequencies > 0)
- izak — Inverse Zak transform
- kaytth — Kay-Tretter filter computation
- odd — Round towards nearest odd value
- rem — Return the remainder of the division x/y
- rot90 — rotates the given 2-D array by 90 degrees
- umaxbert — Determination of the maximum value of u for Bertrand distribution
- umaxdfla — Determination of the maximum value of u for D-Flandrin distribution
- umaxunte — Determination of the maximum value of u for Unterberger distribution
- vecmodulo — Congruence of a vector
- zak — Zak transform

contwtgn

computes a continuous wavelet transform

Calling Sequence

```
[scalo,f,T,a,wt,wavescaled] = contwtgn(x, fmin, fmax, N,
wave)
```

Parameters

x :

a real vector of size Nx: the signal (in time) to be analyzed

fmin:

a positive scalar in]0 0.5], the normalized lower frequency bound in (in Hz) of the analyzed signal. When unspecified, you have to enter it at the command line from the plot of the spectrum.

fmax :

a positive scalar in]0 0.5], the normalized upper frequency bound (in Hz) of the analyzed signal. When unspecified, you have to enter it at the command line from the plot of the spectrum.

N :

a positive integer: number of analyzed voices. When unspecified, you have to enter it at the command line.

wave :

a positive integer: the analyzing wavelet order "wave" derivative of the Gaussian is chosen

scalo :

a real N by Nx matrix: the scalogram (squared magnitude of WT)

f :

a real column vector of size N: the frequency samples (geometrically sampled between fmax and fmin).

T :

a real row vector of size Nx: the time samples (sampling period =1)

a :

a real column vector of size N: the scale vector (geometrically sampled between 1 and fmax/fmin)

wt :

a real N by Nx matrix: the coefficients of the wavelet transform. X-axis corresponds to time (uniformly sampled), Y-axis corresponds to frequency (or scale) samples (geometrically sampled between fmin (resp. fmax/fmin and fmax (resp. 1) First row of wt corresponds to the lowest analyzed frequency.

wavescaled :

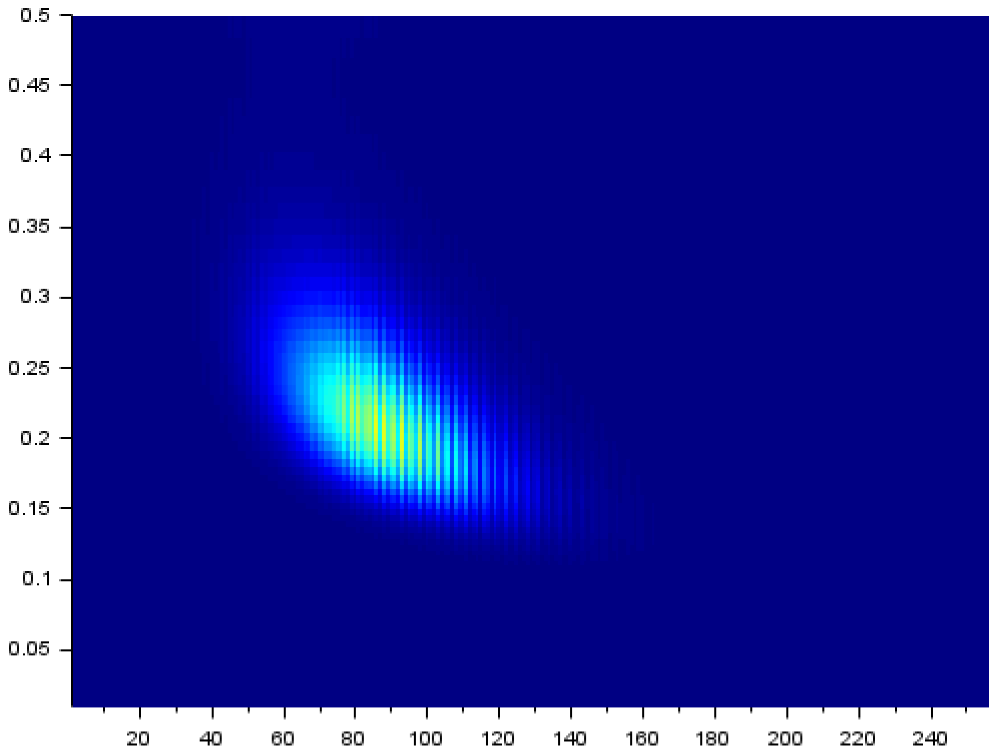
when the analyzing wavelet is Morlet or Mexican hat, wavescaled = wave.

Description

contwtgn computes a continuous wavelet transform.

Examples

```
S = altes(256,0.1,0.45,10000) ;
[scalo,f,T] = contwtgn(S,0.01,0.5,128,8) ;

clf; gcf().color_map= jetcolormap(128);
grayplot(T,f,scalo')
```



Authors

- H. Nahrstaedt - Aug 2010

contwtgnmir

Continuous wavelet transform of mirrored 1-D signals

Calling Sequence

```
[scalo,f,T,a,wt,wavescaled] = contwtgnmir(x, fmin, fmax,
N, wave)
```

Parameters

x :

a real vector of size Nx: the signal (in time) to be analyzed

fmin:

a positive scalar in]0 0.5], the normalized lower frequency bound in (in Hz) of the analyzed signal. When unspecified, you have to enter it at the command line from the plot of the spectrum.

fmax :

a positive scalar in]0 0.5], the normalized upper frequency bound (in Hz) of the analyzed signal. When unspecified, you have to enter it at the command line from the plot of the spectrum.

N :

a positive integer: number of analyzed voices. When unspecified, you have to enter it at the command line.

wave :

a positive integer: the analyzing wavelet order "wave" derivative of the Gaussian is chosen

scalo :

a real N by Nx matrix: the scalogram (squared magnitude of WT)

f :

a real column vector of size N: the frequency samples (geometrically sampled between fmax and fmin).

T :

a real row vector of size Nx: the time samples (sampling period =1)

a :

a real column vector of size N: the scale vector (geometrically sampled between 1 and fmax/fmin)

wt :

a real N by Nx matrix: the coefficients of the wavelet transform. X-axis corresponds to time (uniformly sampled), Y-axis corresponds to frequency (or scale) samples (geometrically sampled between fmin (resp. fmax/fmin and fmax (resp. 1) First row of wt corresponds to the lowest analyzed frequency.

wavescaled :

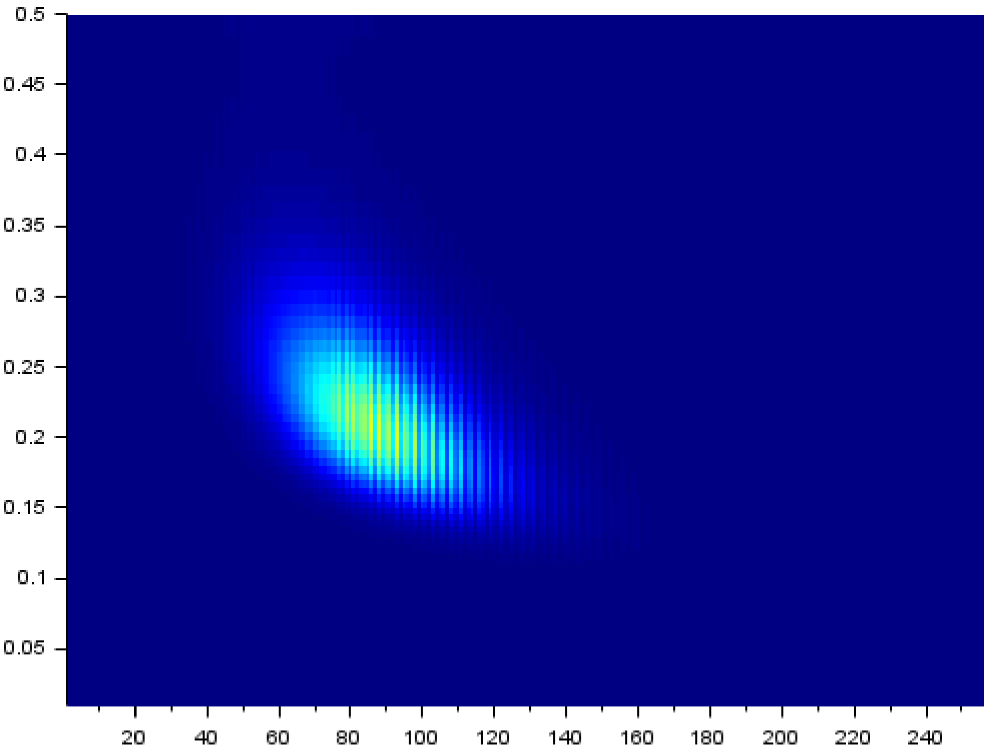
when the analyzing wavelet is Morlet or Mexican hat, wavescaled = wave.

Description

If x = [a b c e f] is the signal to analyzed, contwtmir runs contwt on the mirrored version XxX = [c b [a b c d e f] e d]. The number of mirrored samples depends on the analyzed scale and the wavelet length. USE AN ORDER "wave" DERIVATIVE OF THE GAUSSIAN

Examples

```
S = altes(256,0.1,0.45,10000) ;
[scalo,f,T] = contwtgnmir(S,0.01,0.5,128) ;
clf;gcf().color_map = jetcolormap(128);
grayplot(T,f,scalo')
```



correlmx

correlation matrix of a signal

Calling Sequence

```
Rx = correlmx(x, p)
Rx = correlmx(x, p, Rxtype)
```

Parameters

x :

a real or complex vector of size Nx: the analyzed signal.

p :

an integer in [1 Nx]: the last autocorrelation lag

Rxtype :

computation algorithm (default : 'fbhermitian') possible values : 'hermitian', 'fbhermitian', 'burg' or 'fbburg'. The default value is 'hermitian'

Rx :

a real or complex matrix of size p+1 by p+1: the correlation matrix.

Examples

```
N = 1000; f = 0.1;
sig = real(fmconst(N,f))+0.2*rand(N,1,"normal");

Rx = correlmx(sig,2,'burg');
[v,d] = spec(Rx);
acos(-0.5*v(2,1)/v(1,1))/(2*%pi) //should be near f

Rx = correlmx(sig,2,'hermitian');
[v,d] = spec(Rx);
acos(-0.5*v(2,1)/v(1,1))/(2*%pi) //should be near f
```

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, july 1998.

d2statio

Distance to stationarity

Calling Sequence

```
[D,F] = D2STATIO(SIG)
```

Parameters

SIG :

signal to be analyzed (real or complex).

D :

vector giving the distance to stationarity for each frequency.

F :

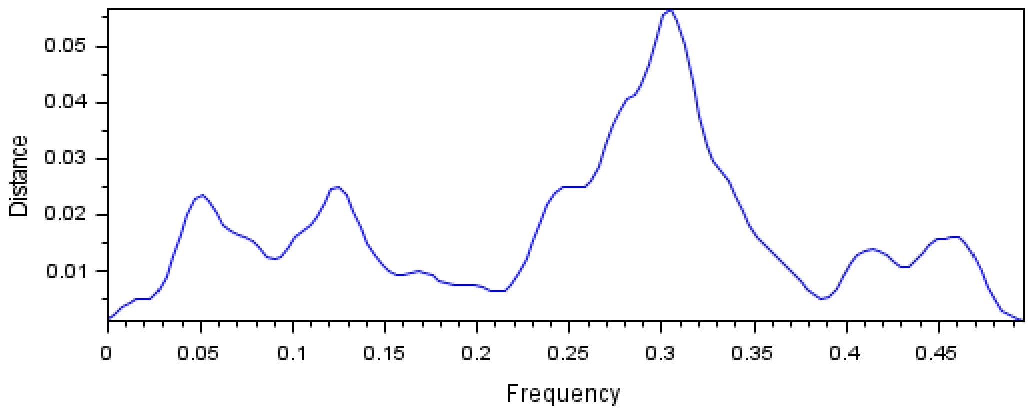
vector of frequency bins

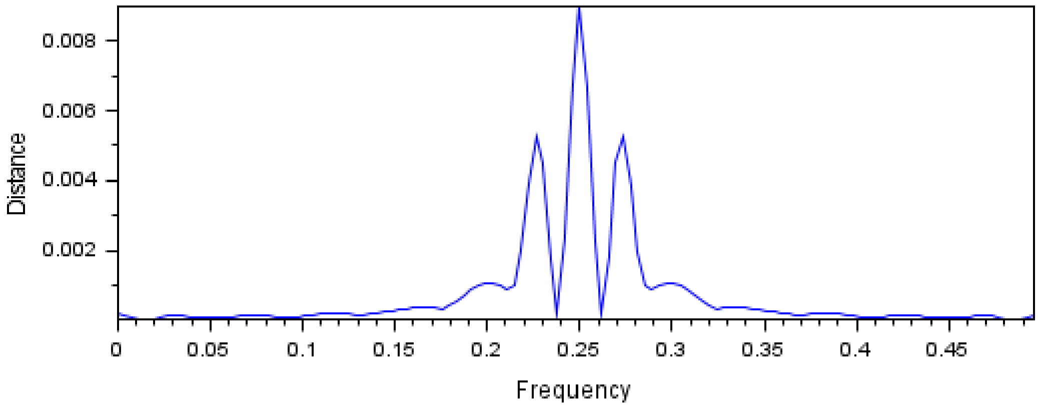
Description

d2statio evaluates the distance of the signal to stationarity, using the pseudo Wigner-Ville distribution.

Examples

```
rand("seed",0);
sig = noisecg(128);
[d,f] = d2statio(sig);
clf
subplot(211); plot(f,d);
xlabel(_('Frequency')); ylabel(_('Distance'));

sig = fmconst(128);
[d,f] = d2statio(sig);
subplot(212);plot(f,d);
xlabel(_('Frequency')); ylabel(_('Distance'));
```





Authors

- H. Nahrstaedt - Aug 2010
- O. Lemoine - May 1996.

disprog

Display progression of a loop

Calling Sequence

```
disprog(i,N,steps)
```

Parameters

I :

a real scalar in [1 N]: the loop variable

N :

a positive scalar: the final value of i

STEPS :

a positive integer: the number of displayed steps.

Description

disprog displays the progression of a loop.

Examples

```
N = 16;
for i=1:N
 disprog(i,N,5);
end
```

See also

- waitbar

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, August, December 1995.
- from an idea of R. Settineri.

divider

Find dividers of an integer

Calling Sequence

```
[N,M] = divider(N1)
```

Parameters

N1 :

an array of positive integers.

N:

an array of positive integers with same sizes as N1.

M:

an array of positive integers with same sizes as N1.

Description

find two integers N and M such that M.*N=N1 and M and N as close as possible from sqrt(N1).

Examples

```
[N,M] = divider(258)
    [N,M] = divider([6 8;4 198])
```

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger - November 1995.
- Copyright (c) 1996 by CNRS (France).

dwindow

Derive a window

Calling Sequence

```
DH = dwindow(H)
DH = dwindow(N, NAME)
DH = dwindow(N, NAME, PARAM)
DH = dwindow(N, NAME, PARAM, PARAM2)
```

Parameters

H :

a real vector: the window coefficients.

N :

a positive integer value: the length of the window

NAME :

a character string : the name of the window shape (see below)

PARAM :

a real number: an optional parameter for 'Kaiser', "Spline" or "Nutbess" windows

PARAM2 :

a real number: a second optional parameters for "Spline" or "Nutbess" windows

DH :

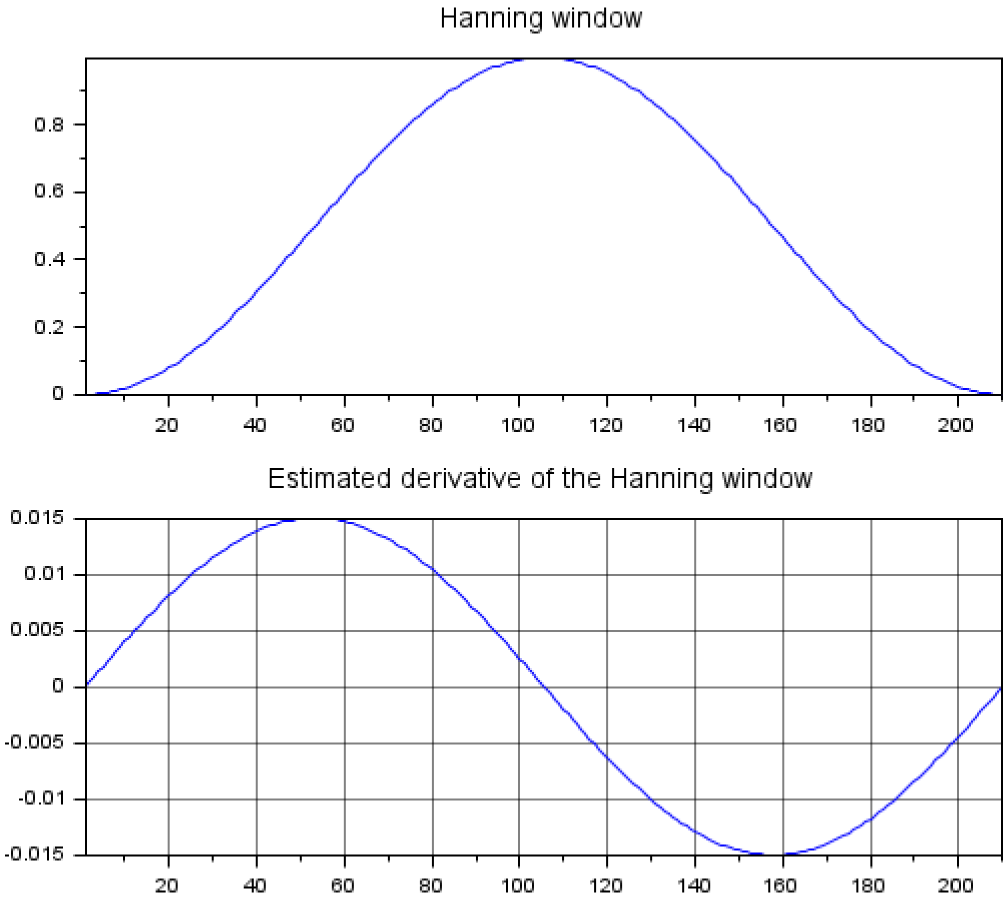
a real vector with same sizes as H: the coefficients of the derivative of H.

Description

dwindow(H) computes an estimation of the given window H. It is should not be accurate for small length windows. dwindow(N,NAME,...) compute exact derivative of the window defined by its name and its length. See tftb_window.

Examples

```
h = tftb_window(210,"hanning");
clf
subplot(211); plot(h);
subplot(212); plot(dwindow(h));xgrid()
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, August 1994, July 1995.
- S. Steer, August 2018.

fzero

solves the scalar nonlinear equation such that F(X) == 0

Calling Sequence

```
[X, FX, INFO] = fzero (FCN, APPROX, OPTIONS)
```

Parameters

FCN :

function name as string

OPTIONS is a structure, with the following fields:

Description

Given Func, the name of a function of the form `F (X)', and an initial approximation APPROX, `fzero' solves the scalar nonlinear equation such that `F(X) == 0'. Depending on APPROX, `fzero' uses different algorithms to solve the problem: either the Brent's method or the Powell's method of `fsolve'. The computed approximation to the zero of FCN is returned in X. FX is then equal to FCN(X). If the iteration converged, INFO == 1. If Brent's method is used, and the function seems discontinuous, INFO is set to -5. If fsolve is used, INFO is determined by its convergence.

Examples

```
fzero('sin',[-2 1])
[x, fx, info] = fzero('sin',-2)
options.abstol = 1e-2; fzero('sin',-2, options)
```

Authors

- H. Nahrstaedt - Aug 2010
- Copyright (C) 2004 Lukasz Bodzon

gaussn

generates the order n derivative of the gaussian window, centered at frequency f0

Calling Sequence

```
gn = gaussn(f0, n)
```

Parameters

f0:

a positive scalar in]0 0.5], the normalized center frequency in (Hz).

n :

a positive integer; the order of derivation

gn :

a real row vector the coefficients of the nth order derivative.

Description

The wavelet gn is real, but it is its analytic form that is synthetized first. The real part is then normalized by the analytical value of its energy (so, L2 normalization!)

Authors

- H. Nahrstaedt

imextrac

imextrac(Image) extract and isolate dots in a binary image

Calling Sequence

```
I = imextrac(Image)
I = imextrac(Image, trace)
```

Parameters

Image :

a boolean array: then black and white image.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

I:

A matrix with same sizes as Image with integer elements: the resulting image with cluster indexed.

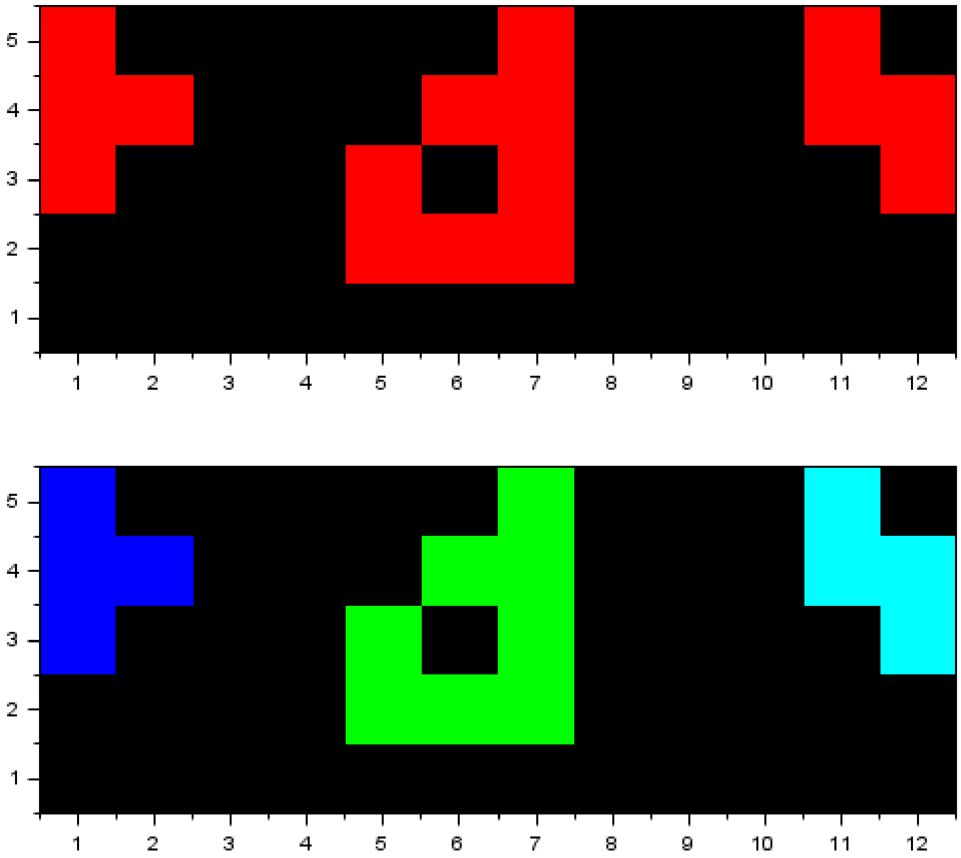
Description

imextrac extracts and isolates dots in a binary image. it finds and indexes the cluster of true value. a cluster is a set of true values image(i,j) such image(i-1,j)==%t or image(i,j-1)==%t.

Examples

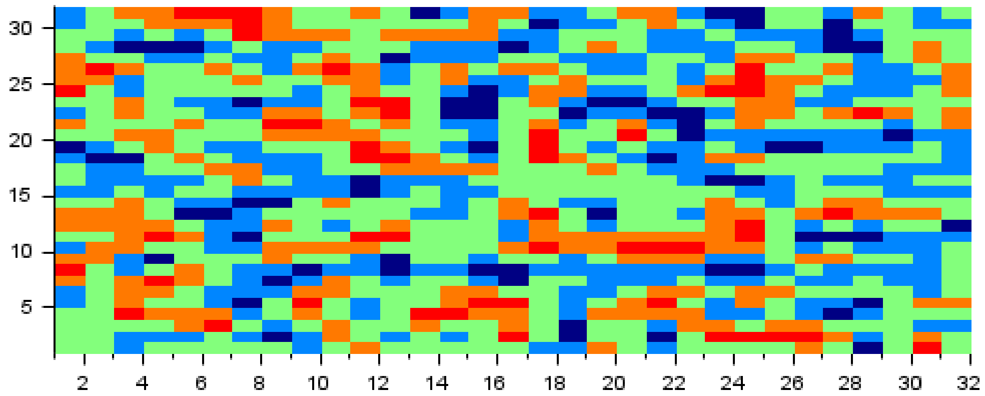
Small example

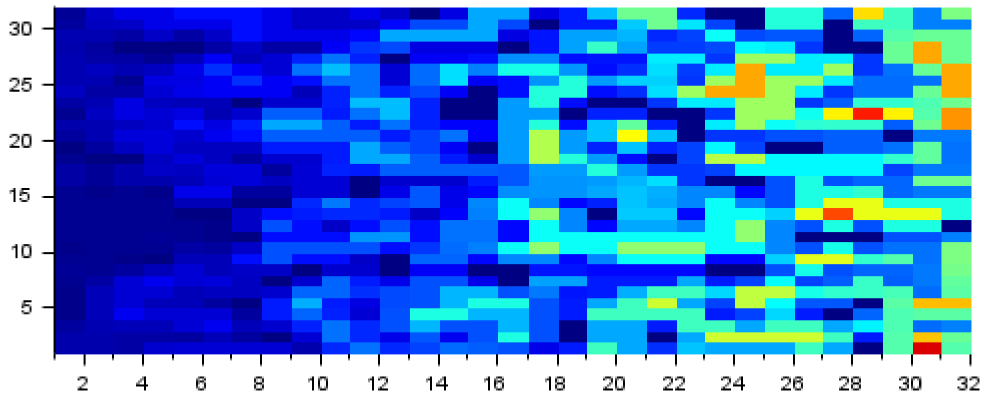
```
Image=[5 0 0 0 0 0 5 0 0 0 5 0 ;
       5 5 0 0 0 5 5 0 0 0 5 5 ;
       5 0 0 0 5 0 5 0 0 0 0 5 ;
       0 0 0 0 5 5 5 0 0 0 0 0 ;
       0 0 0 0 0 0 0 0 0 0 0 0 ];
clf
subplot(211); Matplot(Image)
I = imextrac(Image>0);
subplot(212); Matplot(I)
c_ind = unique(I(I>0)) //the cluster indices
```



Big ones

```
rand("seed",0)
N = 32;
Image = 10*rand(N,N);
Image(Image<5)=0;
clf; gcf().color_map = jetcolormap(150);
Image(Image>0) = color("red");
subplot(211); Matplot(Image)
I = imextrac(Image>0);
subplot(212); Matplot(I)
```





See also

- tfrsurf — extract from a time-frequency representation the biggest energy dots

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, oct 1999

integ

Approximate 1D integral of a discrete signal

Calling Sequence

```
SOM = integ(Y, X)
```

Parameters

Y :

N-vector (or MxN-matrix) to be integrated (along each row).

X :

N-vector containing the integration path of Y (default : 1:N)

SOM :

value (or Mx1 vector) of the integral

Description

integ approximates the integral of a discrete function given by the points (X(k),Y(k)) using trapezoidal approximation.

Examples

```
Y = altes(256,0.1,0.45,10000)'; X = (0:255);
SOM = integ(Y,X)
```

See also

- integ2d — Approximate 2-D integral

Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves, October 95

integ2d

Approximate 2-D integral

Calling Sequence

```
SOM = integ2d(MAT)
SOM = integ2d(MAT, X)
SOM = integ2d(MAT, X, Y)
```

Parameters

MAT :

MxN matrix to be integrated

X :

N-vector indicating the abscissa integration path (default : 1:N)

Y :

M-vector indicating the ordinate integration path (default : 1:M)

SOM :

result of integration

Description

integ2d approximates the 2-D integral of matrix MAT according to abscissa X and ordinate Y.

⚠ Initial version assumes X sampled with a constant step

Examples

```
S = altes(256,0.1,0.45,10000) ;
[TFR,T,F] = tfrscalo(S,21:190,8,0.1,0.35,256) ;
E = integ2d(TFR,T,F)
```

See also

- integ — Approximate 1D integral of a discrete signal

Authors

- H. Nahrstaedt - Aug 2010

- P. Goncalves, October 95

istfr1

returns true is method is a time frequency representation of type 1 (frequencies >0 or <0)

Calling Sequence

```
flag = istfr1(method)
```

Parameters

method :

a character string. Case do not matter.

flag :

a boolean rue if method is the name of a time frequency representation of type 1.

See also

- istfr2 — returns true is method is a time frequency representation of type 2 (only frequencies > 0)
- istfraff — returns true is method is the name of an affine time frequency representation.

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, may 98

istfr2

returns true is method is a time frequency representation of type 2 (only frequencies > 0)

Calling Sequence

```
flag = istfr2(method)
```

Parameters

method :

a character string. Case do not matter.

flag :

a boolean rue if method is the name of a time frequency representation of type 2.

See also

- istfr1 — returns true is method is a time frequency representation of type 1 (frequencies >0 or <0)
- istfraff — returns true is method is the name of an affine time frequency representation.

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, may 98

izak

Inverse Zak transform

Calling Sequence

```
SIG = izak(DZT)
```

Parameters

DZT :

N by M matrix of Zak samples.

SIG :

Output signal (M*N,1) containing the inverse Zak transform.

Description

izak computes the inverse Zak transform of matrix DZT.

Examples

```
sig = fmlin(256);
DZT = zak(sig);
sigr = izak(DZT);
norm(sig-sigr)
```

See also

- zak — Zak transform
- tfrgabor — Gabor representation of a signal

Authors

- H. Nahrstaedt - Aug 2010
- O. Lemoine - February 1996

kaytth

Kay-Tretter filter computation

Calling Sequence

```
H = kaytth(length)
```

Parameters

length:

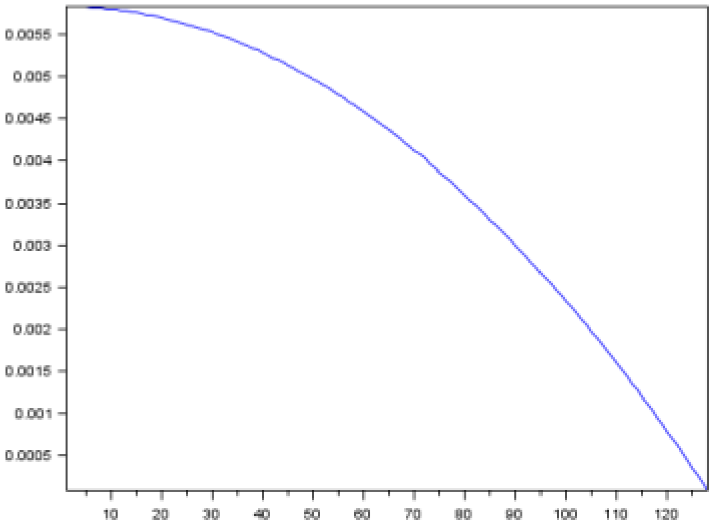
a positive integer: the filter length.

H:

a row vector: the filter coefficients.

Examples

```
clf; plot(kaytth(128))
```



See also

- instfreq — Instantaneous frequency estimation

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, March 1994.

odd

Round towards nearest odd value

Calling Sequence

```
Y = odd(X)
```

Parameters

X :

real array to be rounded

Y :

real array with same sizes as X. The nearest odd values

Description

odd rounds each element of X towards the nearest odd integer value. If an element of X is even, ODD adds +1 to this value.

Examples

```
X = [1.3 2.08 -3.4 90.43];
Y = odd(X)
```

Authors

- H. Nahrstaedt - Aug 2010
- O. Lemoine - August 1996.

rem

Return the remainder of the division x/y

Calling Sequence

```
r = rem(x, y)
```

Parameters

x :

a real or integer type array.

y :

a real o integer type array.

Description

Return the remainder of the division x/y, more precisely r = x - y .* fix (x ./ y);

x or y may be scalars or arrays. If both are arrays they must have the same sizes

x and y may have different types, the results type if the type of x.

Examples

```
rem(1:5,3)
rem(1:5,3*ones(1,5))
r = rem(uint32(1:5),2)
r = rem(uint32(1:5),2)
typeof(r)
```

See Also

- modulo
- pmodulo

rot90

rotates the given 2-D array by 90 degrees

Calling Sequence

```
y = rot90(x, k)
```

Parameters

x :

a 2D array.

k :

an integer which specifies how many 90-degree rotations are to be applied.

Description

Return a copy of x with the elements rotated counterclockwise in 90-degree increments. The second argument is optional, and specifies how many 90-degree rotations are to be applied (the default value is 1). Negative values of k rotate the matrix in a clockwise direction.

Examples

```
A = [1 2 3
     4 5 6
     7 8 9];
rot90(A)      // counterclockwise
rot90(A,-1)   // clockwise
```

umaxbert

Determination of the maximum value of u for Bertrand distribution

Calling Sequence

```
Y = umaxbert(u)
```

Parameters

U :

real vector

Y :

value of the function (H(u)+u/2)/(H(u)-u/2)-fmax/fmin.

Description

umaxbert is the function Y(u)=(H(u)+u/2)/(H(u)-u/2)-fmax/fmin. Doing UMAX = fzero('umaxbert',0); gives the maximum value for U in the computation of the Bertrand distribution. For this distribution, H(u) = (u/2)*coth(u/2).

Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves, October 95 - O. Lemoine, July 1996.
- Copyright (c) 1995 Rice University - CNRS (France).

umaxdfla

Determination of the maximum value of u for D-Flandrin distribution

Calling Sequence

```
Y = umaxdfla(u)
```

Parameters

U :

real vector

Y :

value of the function (H(u)+u/2)/(H(u)-u/2)-FMAX/FMIN.

Description

umaxdfla is the function Y(u)=(H(u)+u/2)/(H(u)-u/2)-fmax/fmin. Doing UMAX = fzero('umaxdfla',0); gives the maximum value for U in the computation of the D-Flandrin distribution. For this distribution, H(u) = 1+(u/4)^2.

Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves, October 95 - O. Lemoine, July 1996.
- Copyright (c) 1995 Rice University - CNRS (France).

umaxunte

Determination of the maximum value of u for Unterberger distribution

Calling Sequence

```
Y = umaxunte(u)
```

Parameters

U :

real vector

Y :

value of the function (H(u)+u/2)/(H(u)-u/2)-fmax/fmin.

Description

umaxunte is the function Y(u)=(H(u)+u/2)/(H(u)-u/2)-fmax/fmin. Doing UMAX = fzero('umaxunte',0); gives the maximum value for U in the computation of the Unterberger distribution. For this distribution, H(u) = sqrt(1+(u/2)^2).

Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves, October 95 - O. Lemoine, July 1996.
- Copyright (c) 1995 Rice University - CNRS (France).

vecmodulo

Congruence of a vector

Calling Sequence

```
Y = vecmodulo(X,N)
```

Description

vecmodulo gives the congruence of each element of the vector X modulo N. These values are strictly positive and lower equal than N.

⚠ vecmodulo can be replaced by modulo.

See also

- rem — Return the remainder of the division x/y
- modulo

Authors

- H. Nahrstaedt - Aug 2010
- O. Lemoine - February 1996.
- Copyright (c) 1996 by CNRS (France).

zak

Zak transform

Calling Sequence

```
DZT = zak(SIG,N,M)
```

Parameters

SIG :

Signal to be analysed (length(X)=N1).

N :

number of Zak coefficients in time (N1 must be a multiple of N) (default : closest integer towards sqrt(N1)).

M :

number of Zak coefficients in frequency (N1 must be a multiple of M) (default : M=N1/N).

DZT :

Output matrix (N,M) containing the discrete Zak transform.

Description

zak computes the Zak transform of signal SIG.

Examples

```
sig = fmlin(256);
DZT = zak(sig);
grayplot(1:16,1:16,DZT');
```

See also

- izak — Inverse Zak transform
- tfrgabor — Gabor representation of a signal

Authors

- H. Nahrstaedt - Aug 2010
- O. Lemoine - February 1996.
- Copyright (c) 1996 by CNRS (France).

Post-Processing or Help to the Interpretation

- friedman — FRIEDMAN Instantaneous frequency density
- holder — Estimate the Holder exponent through an affine TFR
- htl — Hough transform for detection of lines in images
- margtfr — Marginals and energy of a time-frequency representation
- midpoint — Mid-point construction used in the interference diagram
- momftfr — Frequency moments of a time-frequency representation
- momttfr — Time moments of a time-frequency representation
- plotsid — Schematic interference diagram of FM signals
- renyi — Measure Renyi information
- ridges — Extraction of ridges
- tfrideal — Ideal TFR for given instantaneous frequency laws

friedman

FRIEDMAN Instantaneous frequency density

Calling Sequence

```
TIFD = friedman(TFR, HAT
TIFD = friedman(TFR, HAT, T
TIFD = friedman(TFR, HAT, T, METHOD)
TIFD = friedman(TFR, HAT, T, METHOD, TRACE)
TIFD = friedman(...,'plot')
```

Parameters

TFR :

a N by M matrix: the time-frequency representation.

HAT :

a N by M matrix: then complex matrix of the reassignment vectors.

T:

a vector of positive integer values: the time instant(s) (default : (1:M)).

METHOD:

a character string: the chosen representation for plotting (default : 'tfrrsp').

'plot':

if the last input parameter value is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

TIFD :

a real N by M matrix: the time instantaneous-frequency density.

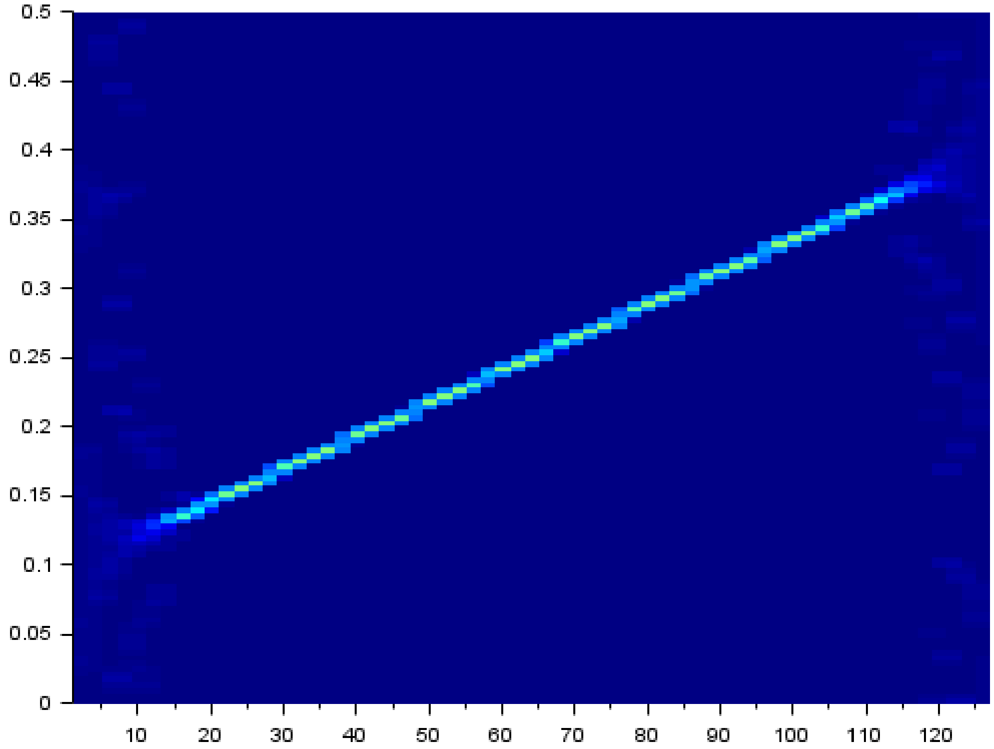
Description

computes the time-instantaneous frequency density (defined by Friedman [1]) of a reassigned time-frequency representation.

WARNING : TIFD is not an energy distribution, but an estimated probability distribution !

Examples

```
sig = fmlin(128,0.1,0.4);
h = window('kr',47,3*%pi);
t = 1:2:127;
[tfr,rtfr,hat] = tfrrpwv(sig,t,128,h);
tfid = friedman(tfr,hat,t);
clf; gcf().color_map = jetcolormap(128);
grayplot(t,linspace(0,0.5,128),tfid')
```



See also

- ridges — Extraction of ridges

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, August 1994, Decembre 1995.

holder

Estimate the Holder exponent through an affine TFR

Calling Sequence

```
H = holder(TFR, F)
H = holder(TFR, F, N1)
H = holder(TFR, F, N1, N2)
H = holder(TFR, F, N1, N2, T)
H = holder(...'plot')
```

Parameters

TFR :

a M by N array of doubles: the affine time-frequency representation.

F :

a real vector of size M with elements in]0 0.5]: the frequency values of the spectral analysis.

N1 :

an integer value in [1 M]: the indice of the minimum frequency for the linear regression. (default : 1).

N2 :

an integer value in [1 M]: the indice of the maximum frequency for the linear regression. (default : M).

T :

a vector of integer values in [1 N]: the time vector. If T is omitted, the function returns the global estimate of the Holder exponent. Otherwise, it returns the local estimates H(T) at the instants specified in T.

H :

a real scalar if T omitted or a real column vector: the Holder estimate(s).

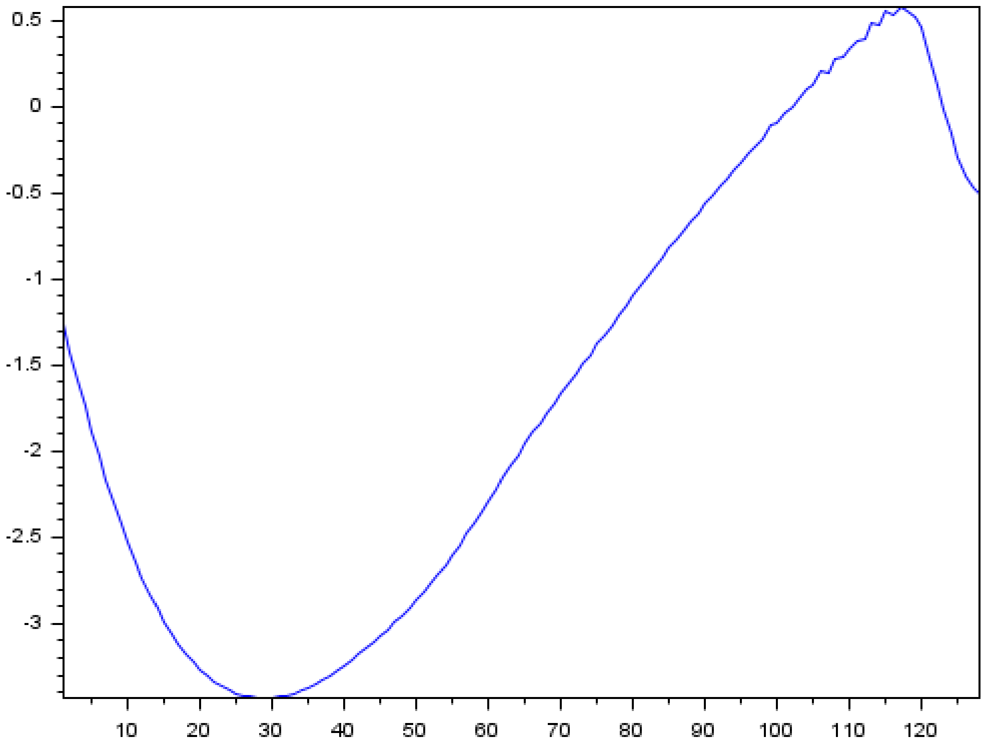
Description

holder estimates the Holder exponent of a function through an affine time-frequency representation of it, and plots the frequency marginal and the regression line.

Examples

```
S = altes(128);
[TFR,T,F] = tfrscalo(S,1:128,8,0.01,0.35,128);
```

```
H = holder(TFR,F,1,length(F),1:128);
clf; plot(H)
```



Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves, October 1995
- Copyright (c) 1995 Rice University

htl

Hough transform for detection of lines in images

Calling Sequence

```
[HT,RHO,THETA] = htl(IM)
[HT,RHO,THETA] = htl(IM, M,)
[HT,RHO,THETA] = htl(IM, M, N)
[HT,RHO,THETA] = htl(IM, M, N, TRACE)
[HT,RHO,THETA] = htl(...,'plot')
```

Parameters

IM :

A real Xmax by Ymax matrix: the image to be analyzed.

M :

A positive integer value: the desired number of samples along the radial axis (default : Xmax).

N :

A positive integer value: the desired number of samples along the azimutal (angle) axis (default : Ymax).

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

HT :

A real M by N matrix: the Hough transform result.

'plot':

when called with the additional string 'plot', htl displays HT using mesh.

RHO :

A real row vector of size M: the radial coordinates.

THETA :

A real row vector of size N: the azimutal angles (in radians).

Description

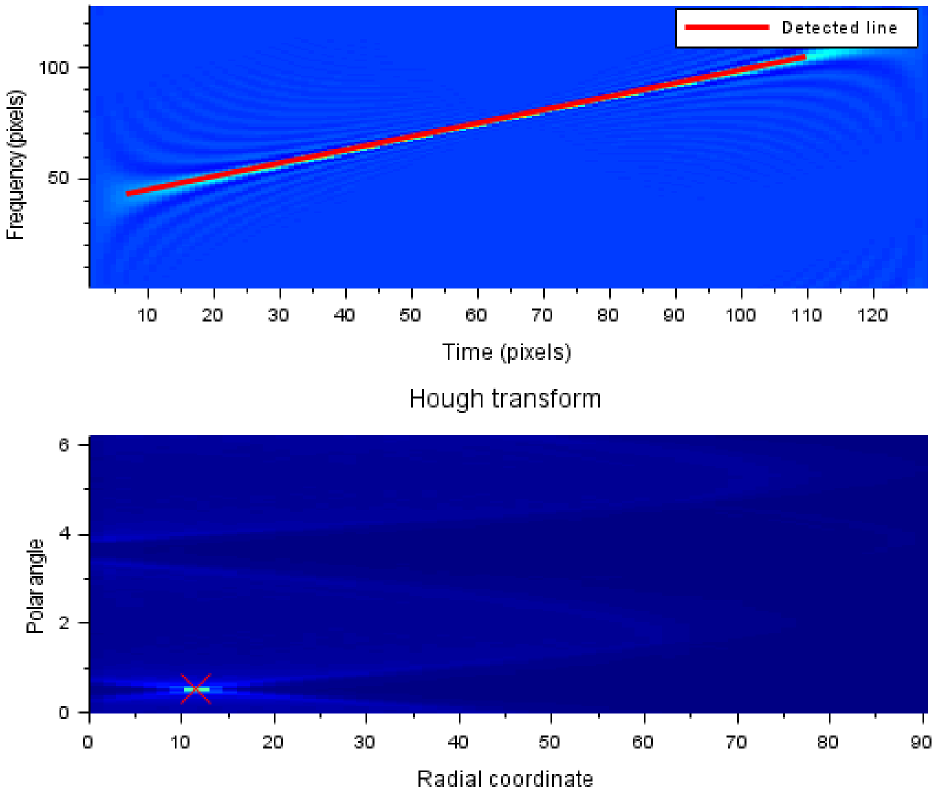
From an image IM, computes the integration of the values of the image over all the lines. The lines are parametrized using polar coordinates.

The origin of the coordinates is fixed at the center of the image (in pixel coordinates), THETA is the azimutal angle and RHO is the distance to the origin of the coordinates.

Only the values of IM exceeding 5% of the maximum are taken into account (to speed up the algorithm).

Examples

```
N = 128; t = (1:N); y = fmlin(N,0.15,0.45);
[IM,T,F] = tfrwv(y,t,N);
//Draw the image
clf; gcf().color_map = jetcolormap(128);
//Draw the image
subplot(211); grayplot(1:N,1:N,IM'),ax=gca();
xlabel(_("Time (pixels)"))
ylabel(_("Frequency (pixels)"))

//Draw the Hough transform result
[HT,RHO,THETA] = htl(IM,64);
subplot(212); grayplot(RHO,THETA,HT),
xlabel(_("Radial coordinate"));
ylabel(_("Polar angle"))
xtitle(_("Hough transform"))

//Locate the Highest HT value
[m,ind] = max(HT);
theta = THETA(ind(2));rho=RHO(ind(1));
xpoly(rho,theta,"marks");

// Draw the line
set(gce(),"mark_style",2,"mark_size",2,"mark_foreground",color("red"));
//Take care rho and theta correspond to an image coordinates in pixels
xl = N/2-rho*sin(theta)+60*cos(theta)*[-1 1];
yl = N/2+rho*cos(theta)+60*sin(theta)*[-1 1];
sca(ax); xpoly(xl,yl);
set(gce(),"foreground",color("red"),"thickness",3);
legend("Detected line");
```



See Also

- Chtl An equivalent function written in C an much more efficient

Authors

- H. Nahrstaedt - Aug 2010
- O. Lemoine - June 1995.

margtfr

Marginals and energy of a time-frequency representation

Calling Sequence

```
[MARGT,MARGF,E] = margtfr(TFR, T, F)
```

Parameters

TFR :

a M by N complex array: the time-frequency representation.

T :

a real vector of size N: the time samples in sec. (default : (1:N))

F :

a real vector of size M: the frequency samples in Hz, not necessary uniformly sampled. (default : (1:M))

MARGT :

a real column vector of size N: then time marginal

MARGF :

a real colum vectorof size M: the frequency marginal

E :

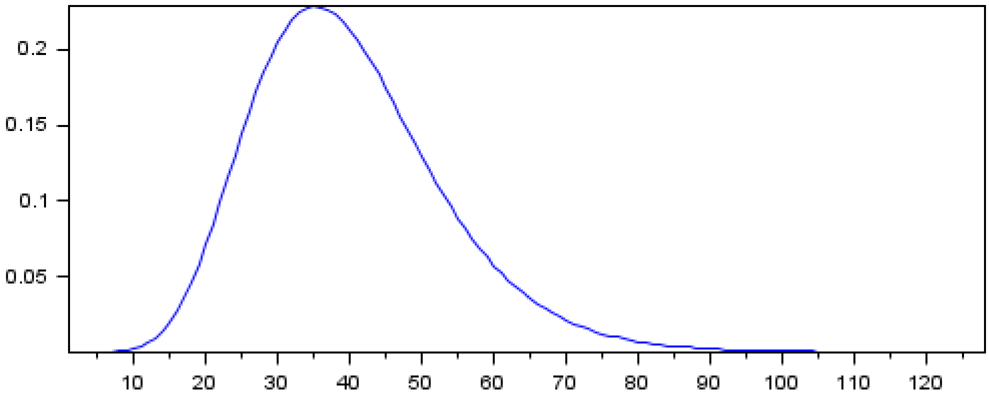
a positive scalar: the energy of TFR

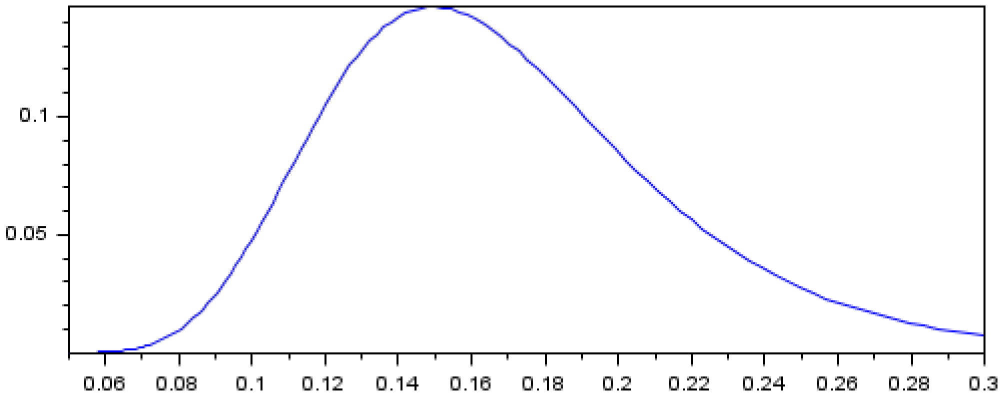
Description

margtfr calculates the time and frequency marginals and the energy of a time-frequency representation.

Examples

```
S = altes(128,0.05,0.45);
[TFR,T,F] = tfrscalo(S,1:128,8,0.05,0.3,128);
[MARGT,MARGF,E] = margtfr(TFR);
clf
subplot(211); plot(T,MARGT);
subplot(212); plot(F,MARGF);
```





See also

- momttfr — Time moments of a time-frequency representation

Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves, October 95
- Copyright (c) 1995 Rice University

midpoint

Mid-point construction used in the interference diagram

Calling Sequence

```
[TI,FI] = midpoint(T1, F1, T2, F2, K)
```

Parameters

T1 :

a real vector of size N: the time-coordinate(s) of the first point(s)

F1 :

a real vector of size N with positive elements: the frequency-coordinate(s) of the first point(s)

T2 :

a real vector of size N: the time-coordinate(s) of the second point(s)

F2 :

a real vector of size N with positive elements: the frequency-coordinate(s) of the second point(s)

K :

a real scalar: the power of the group-delay law

- K = 2 :Wigner-Ville
- K = 1/2 :D-Flandrin
- K = 0:Bertrand (unitary)
- K = -1 :Unterberger (active)
- K = %inf :Margenau-Hill-Rihaczek

TI :

a real row vector of size N: the time-coordinate(s) of the interference term(s)

FI :

a real row vector of size N: frequency-coordinate(s) of the interference term(s)

Description

midpoint gives the coordinates in the time-frequency plane of the interference-term corresponding to the points (T1,F1) and (T2,F2), for a distribution in the affine class perfectly localized on power-law group-delays of the form : tx(nu)=t0+c nu^(K-1).

See also

- plotsid — Schematic interference diagram of FM signals

Authors

- H. Nahrstaedt - Aug 2010
- P. Flandrin, September 1995 - F. Auger, April 1996.

momftfr

Frequency moments of a time-frequency representation

Calling Sequence

```
[TM,D2] = momftfr(TFR)
[TM,D2] = momftfr(TFR, TMIN)
[TM,D2] = momftfr(TFR, TMIN, TMAX)
[TM,D2] = momftfr(TFR, TMIN, TMAX, TIME)
```

Parameters

TFR :

a matrix of size Nrow by Ncol: the time-frequency representation.

TMIN :

an integer in [1 Ncol]: the smallest column element of TFR taken into account (default : 1)

TMAX :

an integer in [TMIN Ncol]: highest column element of TFR taken into account (default : Ncol)

TIME :

a real vector of size TMAX-TMIN+1: the true time instants (default : 1:Ncol)

TM :

a real column vector of size Nrow: the averaged time (first order moment)

D2 :

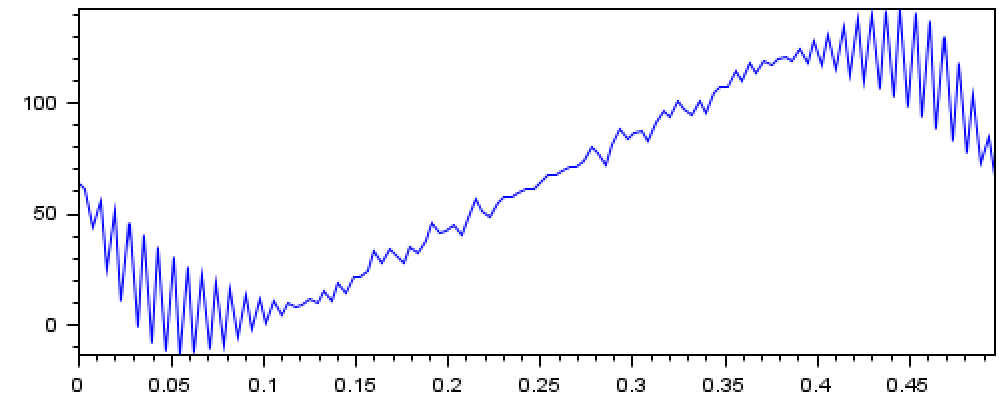
a real column vector of size Nrow: the squared time duration (second order moment)

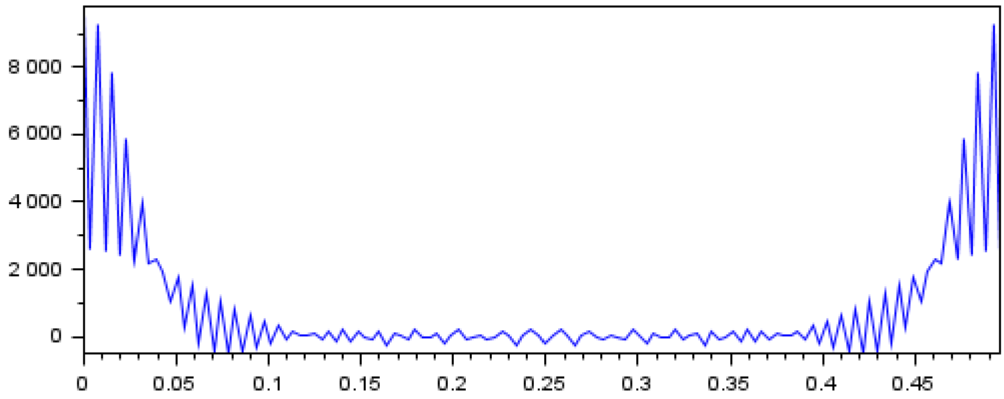
Description

momftfr computes the frequeny moments of a time-frequency representation.

Examples

```
sig = fmlin(128,0.1,0.4);
[tfr,t,f] = tfrwv(sig);
[tm,D2] = momftfr(tfr);
subplot(211); plot(f,tm);
subplot(212); plot(f,D2);
```





See also

- momttfr — Time moments of a time-frequency representation

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, August 1995.

<< momftfr plotsid >>

stftb > Post-Processing or Help to the Interpretation > momttfr

momttfr

Time moments of a time-frequency representation

Calling Sequence

```
[FM,B2] = momttfr(TFR)
[FM,B2] = momttfr(TFR, METHOD)
[FM,B2] = momttfr(TFR, METHOD, FBMIN)
[FM,B2] = momttfr(TFR, METHOD, FBMIN, FBMAX)
[FM,B2] = momttfr(TFR, METHOD, FBMIN, FBMAX, FREQS)
```

Parameters

TFR :

time-frequency representation ([Nrow,Ncol]=size(TFR)).

METHOD:

chosen representation (name of the corresponding M-file).

FBMIN :

smallest frequency bin (default : 1)

FBMAX :

highest frequency bin (default : Nrow)

FREQS :

true frequency of each frequency bin. FREQS must be of length FBMAX-FBMIN+1. (default : 0:step:(0.5-step) or -0.5:step:(0.5-step) depending on METHOD)

FM :

averaged frequency (first order moment)

B2 :

frequency band (second order moment)

Description

momttfr computes the time moments of a time-frequency representation.

Examples

```
sig = fmlin(128,0.1,0.4);
tfr = tfrwv(sig);
[fm,B2] = momttfr(tfr,'tfrwv');
subplot(211); plot(fm); subplot(212); plot(B2);
```

```
freqs = linspace(0,63/128,64);
tfr=tfrsp(sig);
[fm,B2] = momttfr(tfr,'tfrsp',1,64,freqs);
subplot(211); plot(fm); subplot(212); plot(B2);
```

See also

- momftfr — Frequency moments of a time-frequency representation

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, August 1995.
- Copyright (c) 1996 by CNRS (France).

plotsid

Schematic interference diagram of FM signals

Calling Sequence

```
plotsid(T, IFLAWS)
plotsid(T, IFLAWS, K)
```

Parameters

T :

a real vector of size M: the time instants,

IFLAWS :

an M by P real matrix with values in [-0.5 0.5]: each column corresponds to the instantaneous frequency law of a signal.

K :

a real scalar with possible values -1, 0, 1/2, 2 or %inf: the distribution (default : 2):

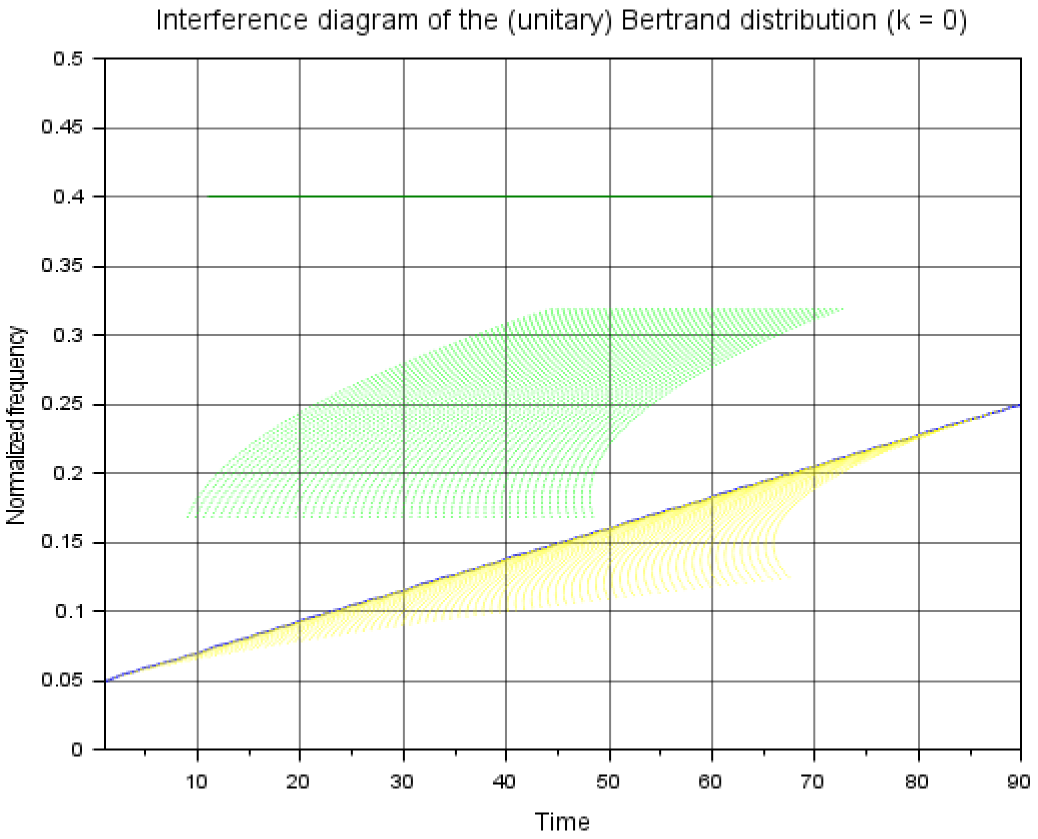
- K=2: Wigner-Ville
- K=1/2: D-Flandrin
- K=0: Bertrand (unitary)
- K=-1 Unterberger (active)
- K=%inf: Margenhau-Hill-Rihaczek

Description

plotsid plots the schematic interference diagram of (analytic) FM signals.

Examples

```
Nt = 90;
[y,iflaw] = fmlin(Nt,0.05,0.25);
[y2,iflaw2] = fmconst(50,0.4);
iflaw(:,2) = [%nan*ones(10,1); iflaw2; %nan*ones(Nt-
60,1)];
plotsid(1:Nt,iflaw,0);
```



See also

- plotifl — Plot normalized instantaneous frequency laws
- midpoint — Mid-point construction used in the interference diagram

Authors

- H. Nahrstaedt - Aug 2010
- P. Flandrin, September 1995.

renyi

Measure Renyi information

Calling Sequence

```
R = renyi(TFR)
R = renyi(TFR, T)
R = renyi(TFR, T, F)
R = renyi(TFR, T, F, ALPHA)
```

Parameters

TFR :

M by N array: the 2-D density function (or mass function). Eventually TFR can be a time-frequency representation, in which case its first row must correspond to the lower frequencies.

T :

a real vector of size N: the abscissa vector parametrizing the TFR matrix. T can be a non-uniform sampled vector (eventually a time vector)(default : (1:N)).

F :

a real vector of size M: the ordinate vector parametrizing the TFR matrix. F can be a non-uniform sampled vector (eventually a frequency vector) (default : (1:M)).

ALPHA :

a positive scalar: the rank of the Renyi measure (default : 3).

R :

the alpha-rank Renyi measure (in bits if TFR is a time-frequency matrix) : R=log2[Sum[TFR(Fi,Ti)^ALPHA dFi.dTi]/(1-ALPHA)]

Description

renyi measures the Renyi information relative to a 2-D density function TFR (which can be eventually a TF representation).

The Renyi entropy furnishes measures for estimating signal information and complexity in the time–frequency plane. When applied to a TFR from the Cohen’s or the affine classes, the Renyi entropies conform to the notion of complexity that we use when inspecting time–frequency images.

Examples

```
s = atoms(64,[32,.3,16,1]);
```

```
[TFR,T,F] = tfrsp(s);
R = renyi(TFR,T,F,3)

s = atoms(64,[16,.2,10,1;40,.4,12,1]);
[TFR,T,F] = tfrsp(s);
R = renyi(TFR,T,F,3)
```

Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves, October 95

ridges

Extraction of ridges

Calling Sequence

```
[POINTST,POINTSF] = ridges(TFR, HAT)
[POINTST,POINTSF] = ridges(TFR, HAT, T)
[POINTST,POINTSF] = ridges(TFR, HAT, T, METHOD)
[POINTST,POINTSF] = ridges(TFR, HAT, T, METHOD, TRACE)
[POINTST,POINTSF] = ridges(...,'plot')
```

Parameters

TFR :

time-frequency representation

HAT :

complex matrix of the reassignment vectors.

T :

the time instant(s).

METHOD :

the chosen representation (default: 'tfrrsp').

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

when called with the additional string 'plot', the output values will be plotted

POINTST:

a real vector: the time coordinates of the stationary points of the reassignment.

POINTSF:

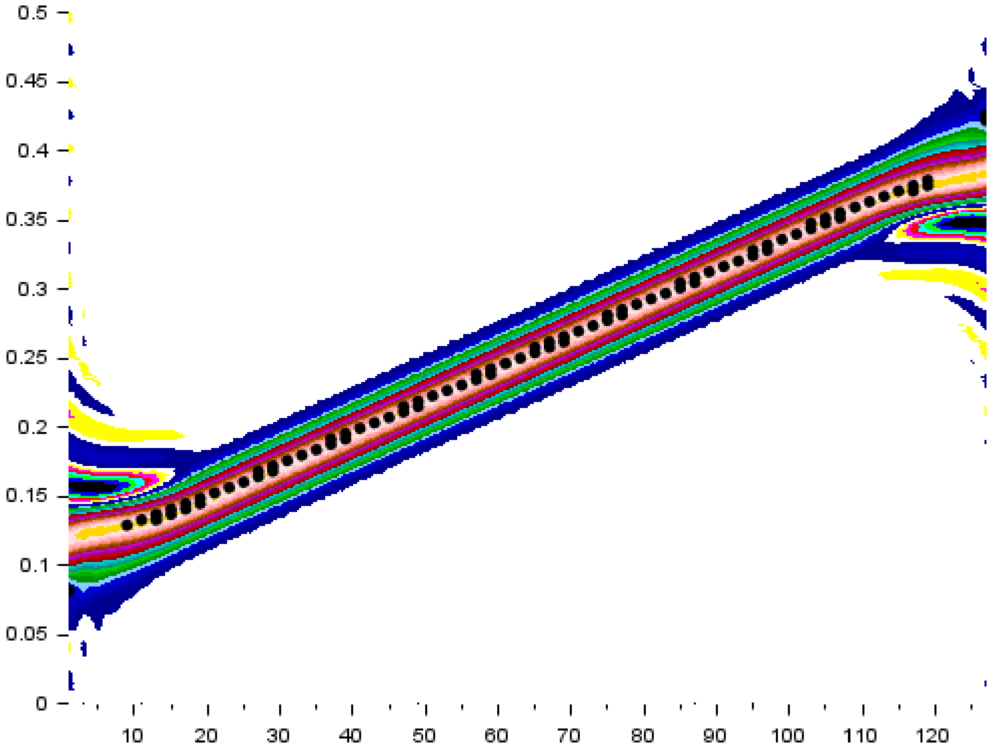
a real vector: the frequency coordinates of the stationary points of the reassignment. Therefore, plot(POINTST,POINTSF,'.') shows the squeleton of the representation.

Description

ridges extracts the ridges of a time-frequency distribution. These ridges are some particular sets of curves deduced from the stationary points of their reassignment operators.

Examples

```
sig = fmlin(128,0.1,0.4);
g = window("kr",21,3*%pi)';
h = window("kr",47,3*%pi)';
t = 1:2:127;
[tfr,rtfr,hat] = tfrrspwv(sig,t,128,g,h);
[pt,pf] = ridges(tfr,hat,t,'tfrrspwv');
clf
Sgrayplot(t,linspace(0,0.5,128),tfr')
plot(pt,pf,".k")
```



See also

- friedman — FRIEDMAN Instantaneous frequency density

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, August 1994, December 1995.

stftb > Post-Processing or Help to the Interpretation > tfrideal

tfrideal

Ideal TFR for given instantaneous frequency laws

Calling Sequence

```
[TFR,T,F] = tfrideal(IFLAWS)
[TFR,T,F] = tfrideal(IFLAWS, T,)
[TFR,T,F] = tfrideal(IFLAWS, T, N)
[TFR,T,F] = tfrideal(IFLAWS, T, N, TRACE)
[TFR,T,F] = tfrideal(...,'plot')
```

Parameters

IFLAWS :

(M,P)-matrix where each column corresponds to the instantaneous frequency law of an (M,1)-signal, These P signals do not need to be present at the same time (missing values are represented by NaN's. The values of IFLAWS must be between 0 and 0.5.

T :

a vector of integer: the time instant(s) (default : 1:M).

N :

a positive integer: the number of frequency bins (default : M).

TRACE :

if non zero or %t, the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrideal runs tfrqview. and TFR will be plotted

TFR :

a real N by length(T) matrix with values in {0 1 %nan}: the output time-frequency matrix.

F :

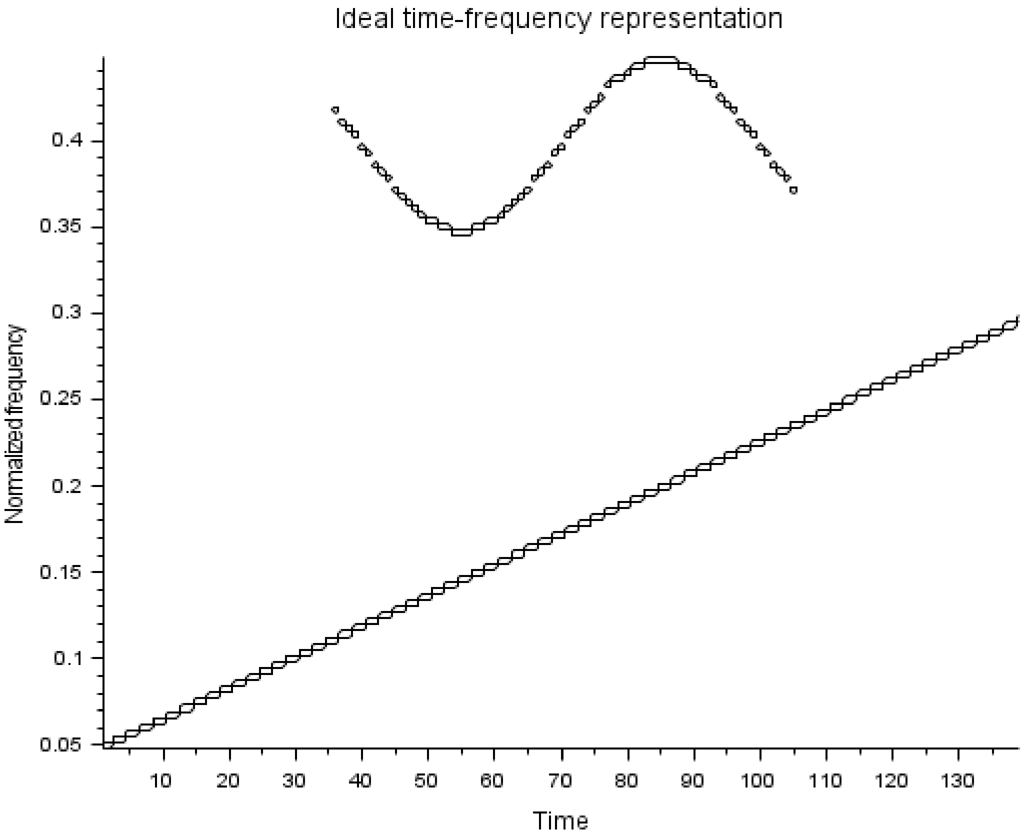
a real vector: the normalized frequencies.

Description

tfrideal generates the ideal time-frequency representation corresponding to the instantaneous frequency laws of the components of a signal.

Examples

```
N = 140;
[x1,if1] = fmlin(N,0.05,0.3);
[x2,if2] = fmsin(70,0.35,0.45,60);
if2 = [zeros(35,1)*%nan ; if2 ; zeros(35,1)*%nan];
[tfr,t,f] = tfrideal([if1 if2],1:N-1);
clf
contour(t,f,tfr',1);
xlabel(_('Time')); ylabel(_('Normalized frequency'));
title(_('Ideal time-frequency representation'));
```



Authors

- H. Nahrstaedt - Aug 2010
- O. Lemoine, F. Auger - March, April 1996.

Reassigned Time-Frequency Processing

- tfrrgab — Reassigned Gabor spectrogram time-frequency distribution
- tfrrmsc — Reassigned Morlet Scalogram time-frequency distribution
- tfrrpmh — Reassigned pseudo Margenau-Hill time-frequency distribution
- tfrrppag — Reassigned pseudo Page time-frequency distribution
- tfrrpwv — Reassigned pseudo Wigner-Ville distribution
- tfrrsp — Reassigned Spectrogram
- tfrrspwv — Reassigned smoothed pseudo Wigner-Ville distribution
- tfrrstan — Reassigned Stankovic distribution

tfrrgab

Reassigned Gabor spectrogram time-frequency distribution

Calling Sequence

```
[TFR,RTFR,HAT] = tfrrgab(X)
[TFR,RTFR,HAT] = tfrrgab(X, T)
[TFR,RTFR,HAT] = tfrrgab(X, T, N)
[TFR,RTFR,HAT] = tfrrgab(X, T, N, NH)
[TFR,RTFR,HAT] = tfrrgab(X, T, N, NH, TRACE)
[TFR,RTFR,HAT] = tfrrgab(X, T, N, NH, TRACE, K)
[TFR,RTFR,HAT] = tfrrgab(...,'plot')
```

Parameters

X :

A Nx elements vector: the signal.

T:

a real Nt vector with elements in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:NX). For faster computation N should be a power of 2.

NH :

an odd positive integer: the length of the gaussian window (default : the first odd number greater than N/4))

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

K :

a real sclar: the value at both extremities (default 0.001)

'plot':

if last input parameter value is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

RTFR :

A real N by Nt array: the reassigned time-frequency representation.

HAT :

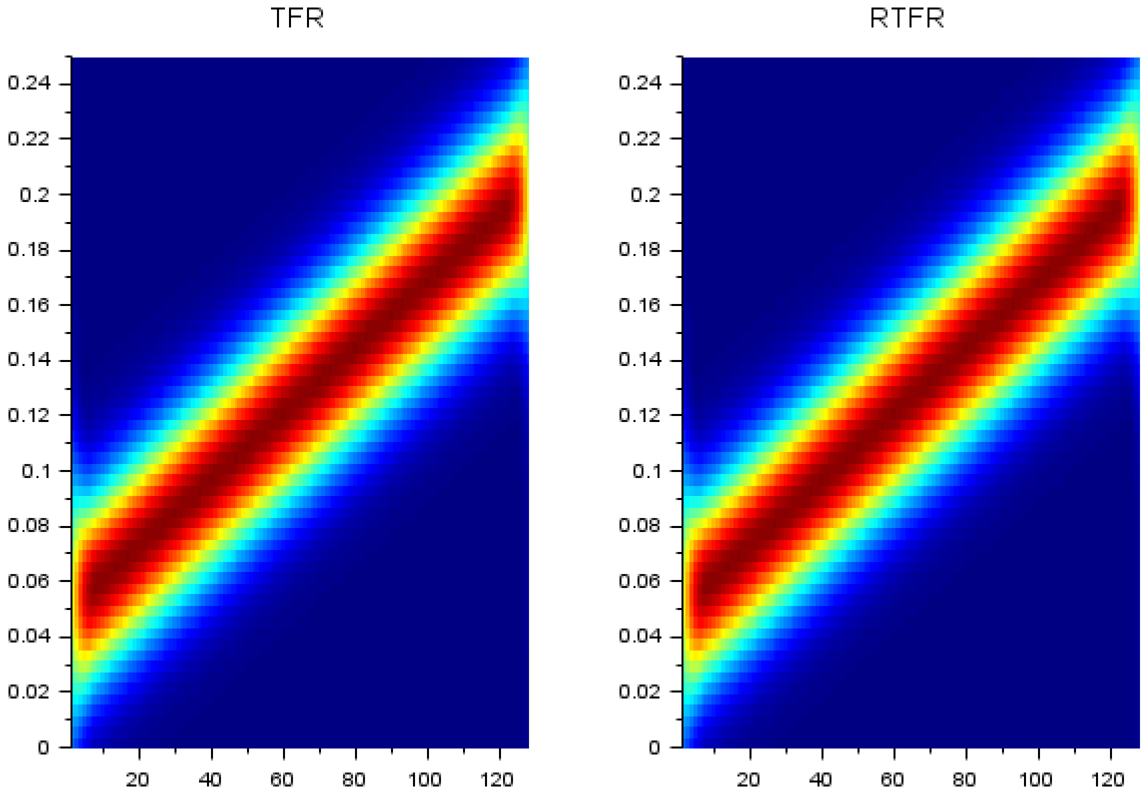
>A complex N by Nt array: the reassignment vectors.

Description

tfrrgab computes the Gabor spectrogram and its reassigned version. This particular window (a Gaussian window) allows a 20 % faster algorithm than the tfrrsp function.

Examples

```
N = 128;
sig = fmlin(N,0.1,0.4);
t = 1:128;
Nh = 19;
[tfr,rtfr,hat] = tfrrgab(sig,t,N,Nh);
clf; gcf().color_map = jetcolormap(128);
subplot(121)
grayplot(t,linspace(0,0.25,N/2),tfr(1:N/2,:)')
title TFR
subplot(122)
grayplot(t,linspace(0,0.25,N/2),tfr(1:N/2,:)')
title RTFR
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-July 1994, July 1995.

tfrrmsc

Reassigned Morlet Scalogram time-frequency distribution

Calling Sequence

```
[TFR,RTFR,HAT] = tfrrmsc(X)
[TFR,RTFR,HAT] = tfrrmsc(X, T)
[TFR,RTFR,HAT] = tfrrmsc(X, T, N)
[TFR,RTFR,HAT] = tfrrmsc(X, T, N, F0T)
[TFR,RTFR,HAT] = tfrrmsc(X, T, N, F0T, TRACE)
[TFR,RTFR,HAT] = tfrrmsc(...,'plot')
```

Parameters

X :

A Nx elements vector or a Nx by 2 array signal.

T :

the time instant(s) with elements in [1 Nx] (default : 1:length(X))

N :

number of frequency bins (default : length(X)). For faster computation N should be a power of 2.

F0T :

a positive scalar: the time-bandwidth product of the mother wavelet (default : 2.5))

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

a N by Nt array: the time-frequency representation.

RTFR :

A real N by Nt array: the reassigned time-frequency representation.

HAT :

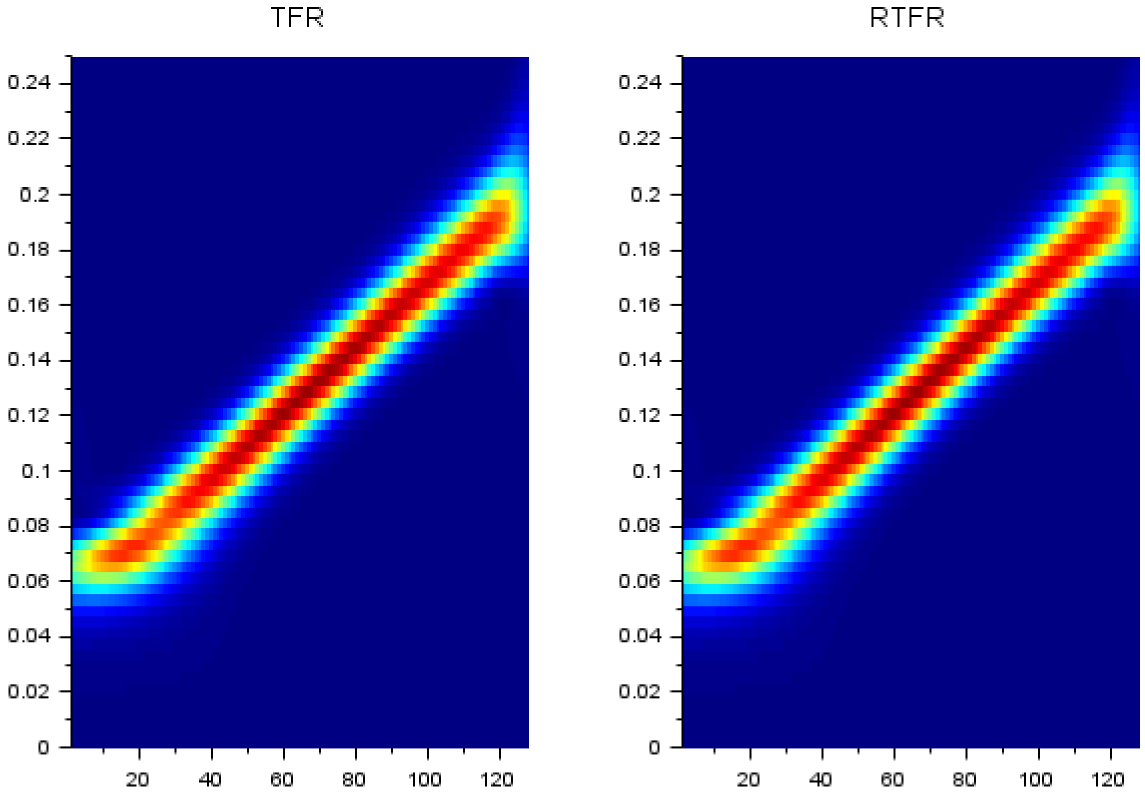
a N by Nt complex matrix: the reassignment vectors.

Description

tfrrmsc computes the Morlet scalogram and its reassigned version.

Examples

```
N = 128;
sig = fmlin(N,0.1,0.4);
t = 1:128;

[tfr,rtfr,hat] = tfrrmsc(sig,t,N,2.1);
clf; gcf().color_map = jetcolormap(128);
subplot(121);
grayplot(t,linspace(0,0.25,N/2),tfr(1:N/2,:)')
title TFR
subplot(122);
grayplot(t,linspace(0,0.25,N/2),rtfr(1:N/2,:)')
title RTFR
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, January, April 1996.
- Copyright (c) 1996 by CNRS (France).

tfrrpmh

Reassigned pseudo Margenau-Hill time-frequency distribution

Calling Sequence

```
[TFR,RTFR,HAT] = tfrrpmh(X)
[TFR,RTFR,HAT] = tfrrpmh(X, T)
[TFR,RTFR,HAT] = tfrrpmh(X, T, N)
[TFR,RTFR,HAT] = tfrrpmh(X, T, N, H)
[TFR,RTFR,HAT] = tfrrpmh(X, T, N, H, TRACE)
[TFR,RTFR,HAT] = tfrrpmh(...,'plot')
```

Parameters

X :

A Nx elements vector or a Nx by 2 array signal.

T :

the time instant(s) with elements in [1 Nx] (default : 1:length(X))

N :

number of frequency bins (default : length(X)). For faster computation N should be a power of 2.

H :

a real vector of odd length: the frequency smoothing window, (default : Hamming(N/4)).

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

a N by Nt array: the time-frequency representation.

RTFR :

A real N by Nt array: the reassigned time-frequency representation.

HAT :

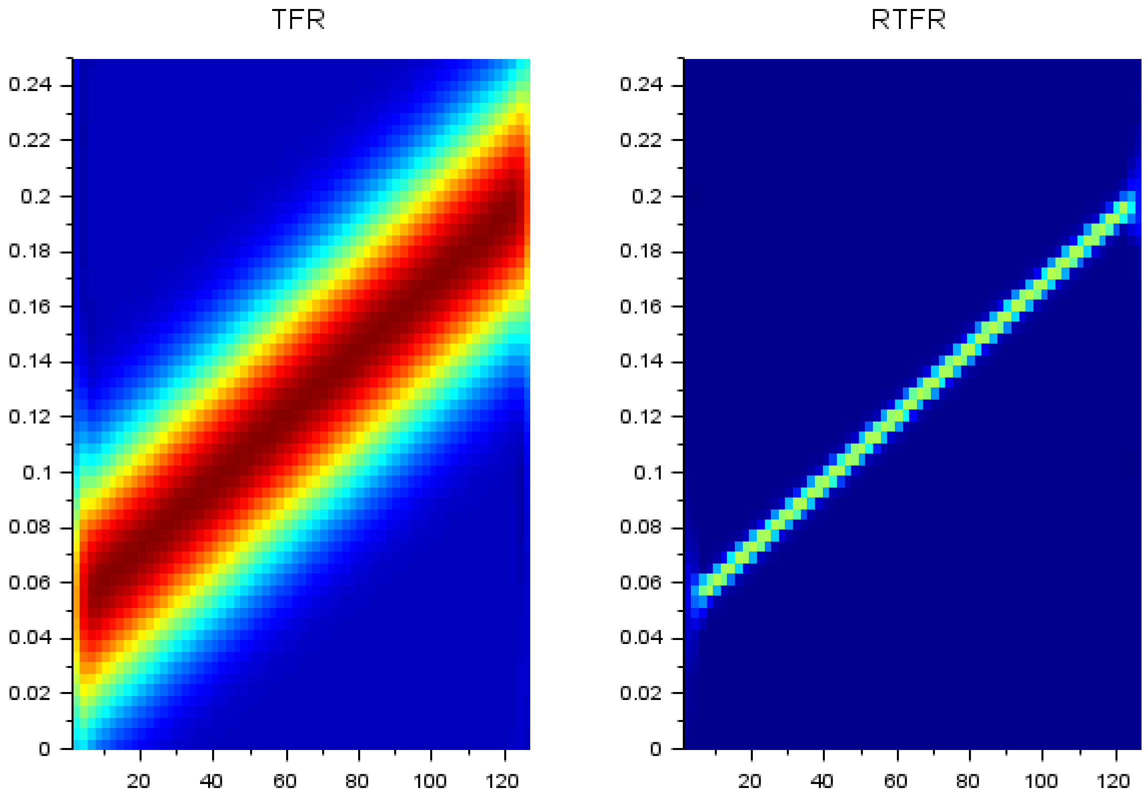
a N by Nt complex matrix: the reassignment vectors.

Description

tfrrpmh computes the pseudo Margenau-Hill distribution and its reassigned version.

Examples

```
N = 128
sig = fmlin(N,0.1,0.4);
t = 1:2:N;
h = window("kr",17,3*%pi);
[tfr,rtfr,hat] = tfrrpmh(sig,t,N,h);
clf; gcf().color_map = jetcolormap(128);
subplot(121)
grayplot(t,linspace(0,0.25,N/2),tfr(1:N/2,:)')
title TFR
subplot(122)
grayplot(t,linspace(0,0.25,N/2),rtfr(1:N/2,:)')
title RTFR
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-July 1994, July 1995.

tfrrppag

Reassigned pseudo Page time-frequency distribution

Calling Sequence

```
[TFR,RTFR,HAT] = tfrrppag(X)
[TFR,RTFR,HAT] = tfrrppag(X, T)
[TFR,RTFR,HAT] = tfrrppag(X, T, N)
[TFR,RTFR,HAT] = tfrrppag(X, T, N, H)
[TFR,RTFR,HAT] = tfrrppag(X, T, N, H, TRACE)
[TFR,RTFR,HAT] = tfrrppag(...,'plot')
```

Parameters

X :

A Nx elements vector or a Nx by 2 array signal.

T :

the time instant(s) with elements in [1 Nx] (default : 1:length(X))

N :

number of frequency bins (default : length(X)). For faster computation N should be a power of 2.

H :

a real vector of odd length: the frequency smoothing window, (default : Hamming(N/4)).

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

a N by Nt array: the time-frequency representation.

RTFR :

A real N by Nt array: the reassigned time-frequency representation.

HAT :

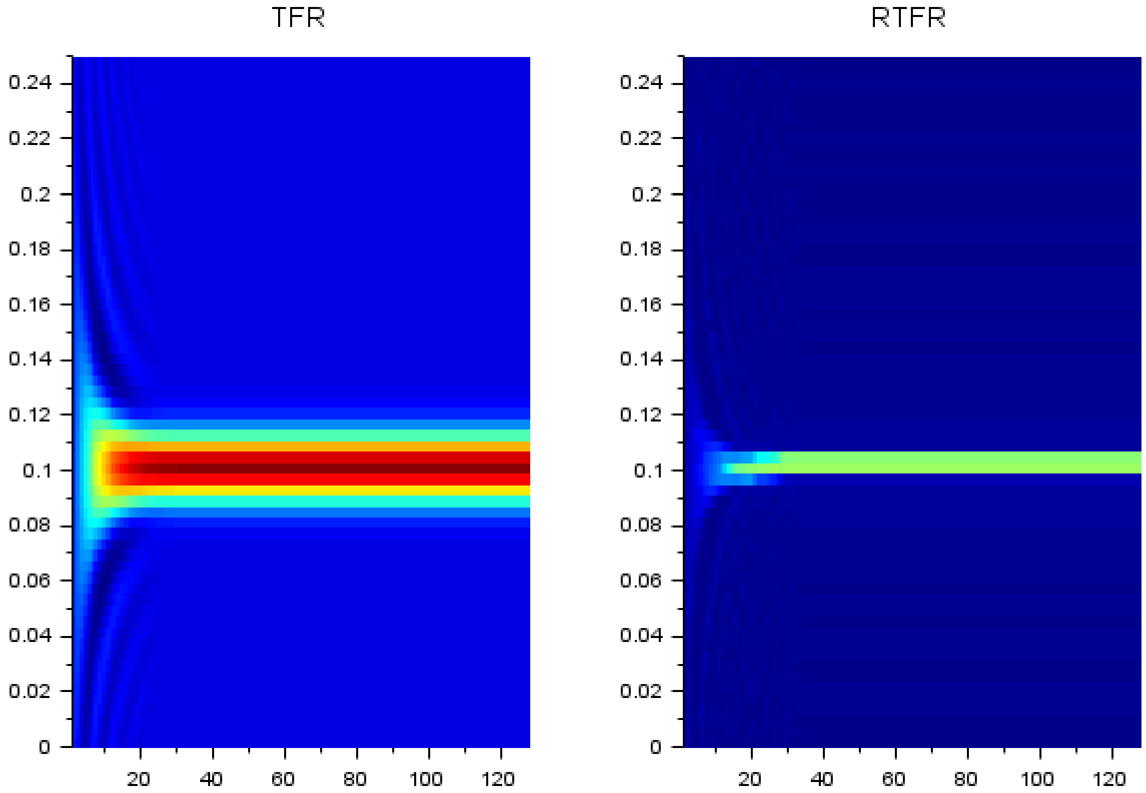
a N by Nt complex matrix: the reassignment vectors.

Description

tfrrppag computes the pseudo Page distribution and its reassigned version.

Examples

```
N = 128;
sig = fmconst(N,0.2);
t = 1:N;
h = tftb_window(65,'gauss');
[tfr,rtfr,hat] = tfrrppag(sig,t,N,h);
clf; gcf().color_map = jetcolormap(128);
 subplot(121)
 grayplot(t,linspace(0,0.25,N/2),tfr(1:N/2,:)')
 title TFR
 subplot(122)
 grayplot(t,linspace(0,0.25,N/2),rtfr(1:N/2,:)')
 title RTFR
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-July 1994, July 1995.

tfrrpwv

Reassigned pseudo Wigner-Ville distribution

Calling Sequence

```
[TFR,RTFR,HAT] = tfrrpwv(X)
[TFR,RTFR,HAT] = tfrrpwv(X, T)
[TFR,RTFR,HAT] = tfrrpwv(X, T, N)
[TFR,RTFR,HAT] = tfrrpwv(X, T, N, H)
[TFR,RTFR,HAT] = tfrrpwv(X, T, N, H, TRACE)
[TFR,RTFR,HAT] = tfrrpwv(...,'plot')
```

Parameters

X :

A Nx elements vector: the analyzed signal.

T:

a real Nt vector with elements regularly spaced in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default: Nx). For faster computation N should be a power of 2.

H :

real vector with odd length: the frequency smoothing window,(default: window("hm",N/4)).

It will be normalized such as the middle point equals 1 to preserve signal energy.

TRACE :

if nonzero,the progression of the algorithm is shown (default : 0).

'plot':

if the last input parameter value is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

RTFR :

A real N by Nt array: the reassigned time-frequency representation.

HAT :

AN by Nt complex matrix of the reassignment vectors.

Description

tfrrpwv computes the pseudo Wigner-Ville distribution and its reassigned version.

Examples

```
sig = fmlin(128,0.1,0.4);
t = 1:2:128;
h = tftb_window("kr",17,3*%pi);
tfrrpwv(sig,t,64,h,1,'plot');
```

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-July 1994, July 1995.
- Copyright (c) 1996 by CNRS (France).

tfrrsp

Reassigned Spectrogram

Calling Sequence

```
[TFR,RTFR,HAT] = TFRRSP(X)
[TFR,RTFR,HAT] = TFRRSP(X, T)
[TFR,RTFR,HAT] = TFRRSP(X, T, N)
[TFR,RTFR,HAT] = TFRRSP(X, T, N, H)
[TFR,RTFR,HAT] = TFRRSP(X, T, N, H, TRACE)
[TFR,RTFR,HAT] = TFRRSP(...,'plot')
```

Parameters

X :

A Nx elements vector or a Nx by 2 array signal.

T :

the time instant(s) with elements in [1 Nx] (default : 1:length(X))

N :

number of frequency bins (default : length(X)). For faster computation N should be a power of 2.

H :

a real vector of odd length: the frequency smoothing window, (default : Hamming(N/4)).

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

a real N by Nt array: the time-frequency representation.

RTFR :

A real N by Nt array: the reassigned time-frequency representation.

HAT :

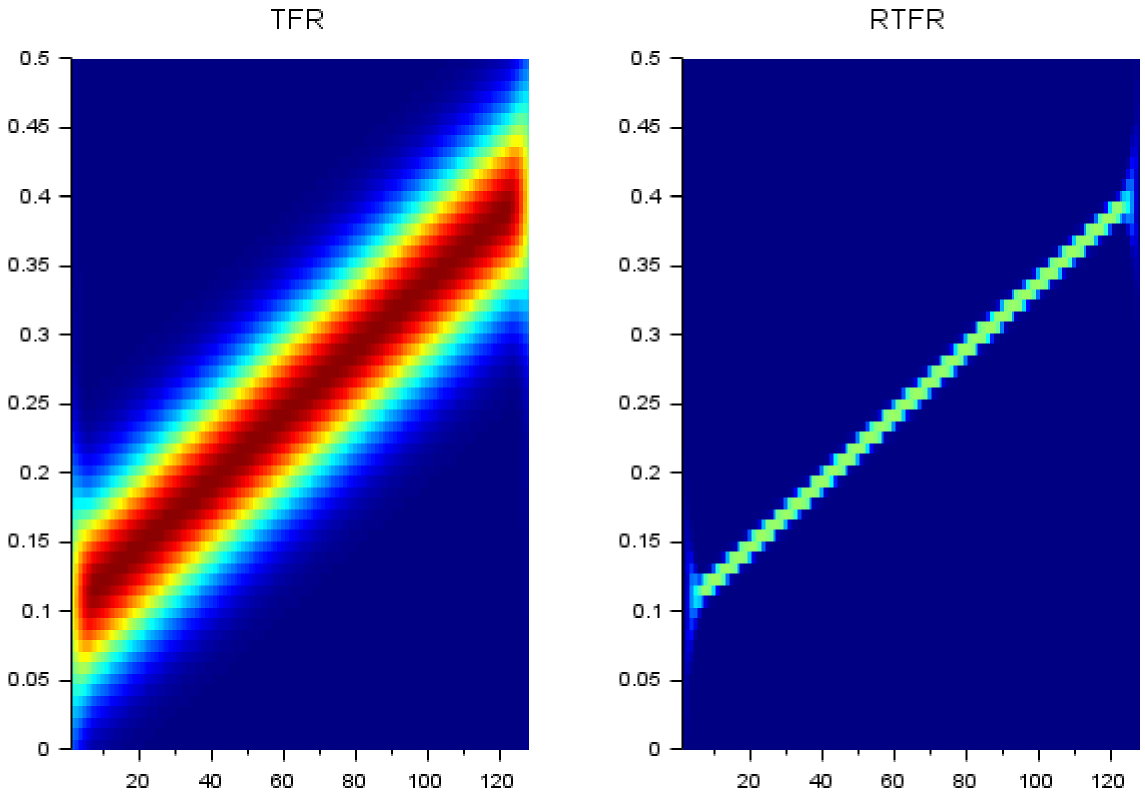
a N by Nt complex matrix: the reassignment vectors.

Description

tfrrsp computes the spectrogram and its reassigned version.

Examples

```
N = 128;
sig = fmlin(N,0.1,0.4);
t = 1:N;
h = window("kr",17,3*%pi);
[tfr,rtfr,hat] = tfrrsp(sig,t,N,h);
clf; gcf().color_map = jetcolormap(128);
subplot(121)
grayplot(t,linspace(0,0.5,N/2),tfr(1:N/2,:)')
title TFR
subplot(122)
grayplot(t,linspace(0,0.5,N/2),rtfr(1:N/2,:)')
title RTFR
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-July 1994, July 1995.

tfrrspwv

Reassigned smoothed pseudo Wigner-Ville distribution

Calling Sequence

```
[TFR,RTFR,HAT] = tfrrspwv(X)
[TFR,RTFR,HAT] = tfrrspwv(X, T)
[TFR,RTFR,HAT] = tfrrspwv(X, T, N)
[TFR,RTFR,HAT] = tfrrspwv(X, T, N, G)
[TFR,RTFR,HAT] = tfrrspwv(X, T, N, G, H)
[TFR,RTFR,HAT] = tfrrspwv(X, T, N, G, H, TRACE)
[TFR,RTFR,HAT] = tfrrspwv(...,'plot')
```

Parameters

X :

A Nx elements vector or a Nx by 2 array signal.

T :

the time instant(s) with elements in [1 Nx] (default : 1:length(X))

N :

number of frequency bins (default : length(X)). For faster computation N should be a power of 2.

G :

a real vector of odd length: the time smoothing window (default : Hamming(N/10)).

H :

a real vector of odd length: the frequency smoothing window, (default : Hamming(N/4)).

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

a real N by Nt array: the time-frequency representation.

RTFR :

A real N by Nt array: the reassigned time-frequency representation.

HAT :

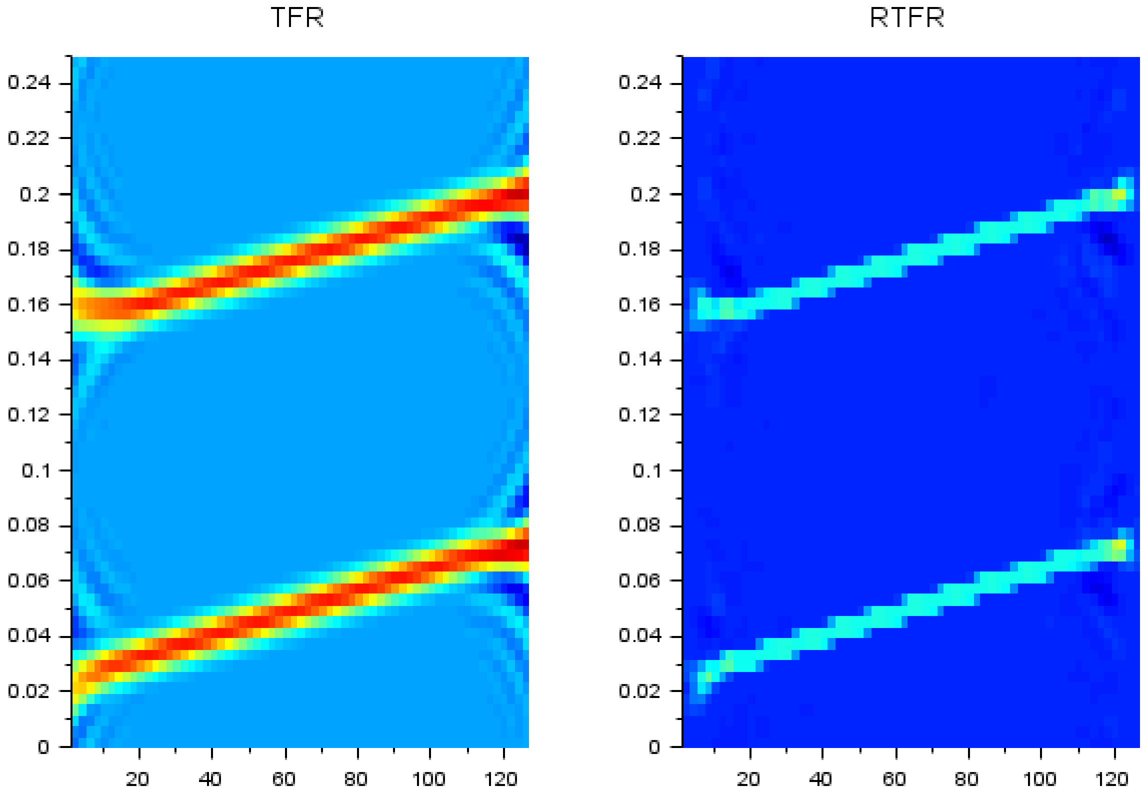
a N by Nt complex matrix: the reassignment vectors.

Description

tfrrspwv computes the smoothed pseudo Wigner-Ville distribution and its reassigned version.

Examples

```
N = 128;
sig = fmlin(N,0.05,0.15) + fmlin(N,0.3,0.4);
t = 1:2:N;
g = window("kr",15,3*%pi);
h = window("kr",63,3*%pi);
[tfr,rtfr,hat] = tfrrspwv(sig,t,64,g,h);
clf; gcf().color_map = jetcolormap(128);
subplot(121)
grayplot(t,linspace(0,0.25,N/2),tfr(1:N/2,:)')
title TFR
subplot(122)
grayplot(t,linspace(0,0.25,N/2),rtfr(1:N/2,:)')
title RTFR
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, May-July 1994, July 1995.

tfrrstan

Reassigned Stankovic distribution

Calling Sequence

```
[TFR,RTFR,HAT] = tfrrstan(X)
[TFR,RTFR,HAT] = tfrrstan(X, T)
[TFR,RTFR,HAT] = tfrrstan(X, T, N)
[TFR,RTFR,HAT] = tfrrstan(X, T, N, G)

[TFR,RTFR,HAT] = tfrrstan(X, T, N, G, H)
[TFR,RTFR,HAT] = tfrrstan(X, T, N, G, H, TRACE)
[TFR,RTFR,HAT] = tfrrstan(...,'plot')
```

Parameters

X :

A Nx elements vector: the analyzed signal .

T:

a real Nt vector with elements regularly spaced in [1 Nx] : time instant(s) (default: 1:NX).

N:

a positive integer: the number of frequency bins (default:NX). For faster computation N should be a power of 2.

G:

a real vector with odd length: the frequency averaging window (default :[0.25 0.5 0.25]).

H :

real vector with odd length: the stft smoothing window,(default: Hamming(N/4)).

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

TFR :

A real N by Nt array: the time-frequency representation.

RTFR :

A real N by Nt array: the reassigned time-frequency representation.

HAT :

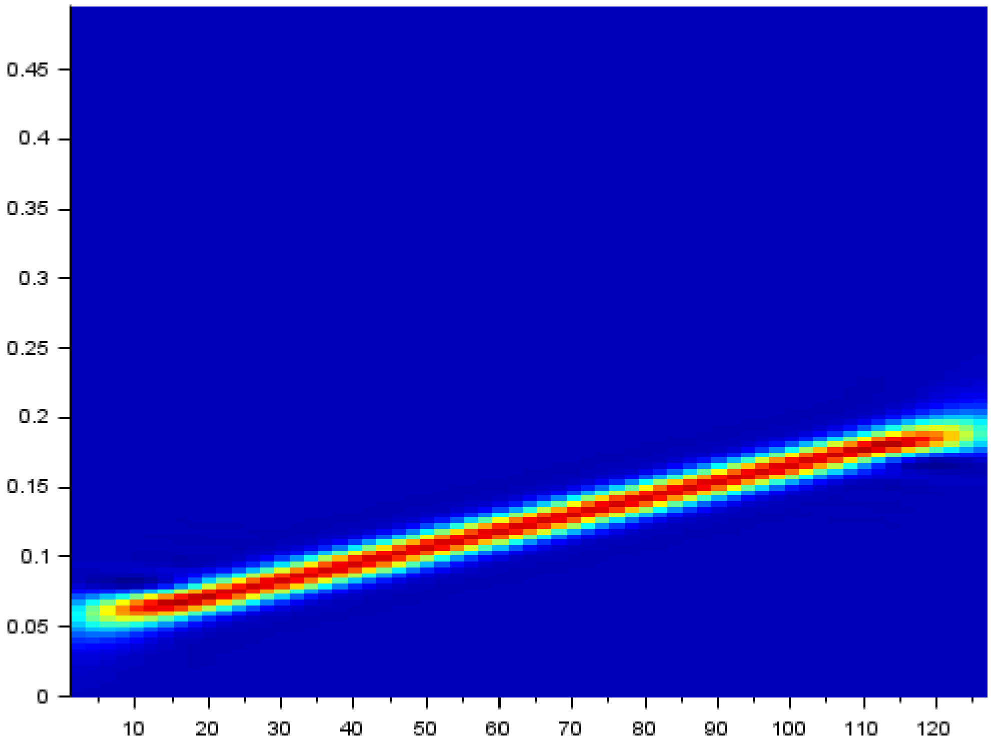
A complex N by Nt array: the reassignment vectors.

Description

tfrrstan computes the Stankovic distribution and its reassigned version.

Examples

```
N = 128;
sig = fmlin(N,0.1,0.4);
g = window('hn',9);
h = window("hn",61);
t = 1:2:128;
tfr = tfrrstan(sig,t,N,g,h);
f = (0.5*(0:N-1)/N)';
clf; gcf().color_map = jetcolormap(128);
grayplot(t,f,tfr');
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, August, September 1997.

Choice of Particular Signals

- altes — Altes signal in time domain
- anaask — Amplitude Shift Keying (ASK) signal
- anabpsk — Binary Phase Shift Keying (BPSK) signal
- anafsk — Frequency Shift Keying (FSK) signal
- anapulse — Analytic projection of unit amplitude impulse signal
- anaqpsk — Quaternary Phase Shift Keying (QPSK) signal
- anasing — Lipschitz singularity
- anastep — Analytic projection of unit step signal
- atoms — Linear combination of elementary Gaussian atoms
- chirp — Evaluate a chirp signal at time t. A chirp signal is a frequency swept cosine wave
- dopnoise — Complex noisy doppler signal
- doppler — Generate complex Doppler signal
- griffitc — Test signal example C of Griffiths' paper
- klauder — Klauder wavelet in time domain
- mexhat — Mexican hat wavelet in time domain
- pulstran — pulse generation
- sawtooth — Generates a sawtooth wave
- tftb_window — Window generation

altes

Altes signal in time domain

Calling Sequence

```
x = altes(N)
x = altes(N, fmin)
x = altes(N, fmin, fmax)
x = altes(N, fmin, fmax, alpha)
```

Parameters

N :

A scalar with a positive integer value: the number of points in time

fmin :

a positive scalar: the lower frequency bound (value of the hyperbolic instantaneous frequency law at the sample N), in normalized frequency (default is 0.05).

fmax :

a positive scalar: the upper frequency bound (value of the hyperbolic instantaneous frequency law at the first sample), in normalized frequency (default is 0.5).

⚠ fmax must be greater than fmin.

alpha :

a real scalar >1: the attenuation factor of the envelope (default is 300).

x :

real row vector containing the Altes signal samples. the associated time step is 1/

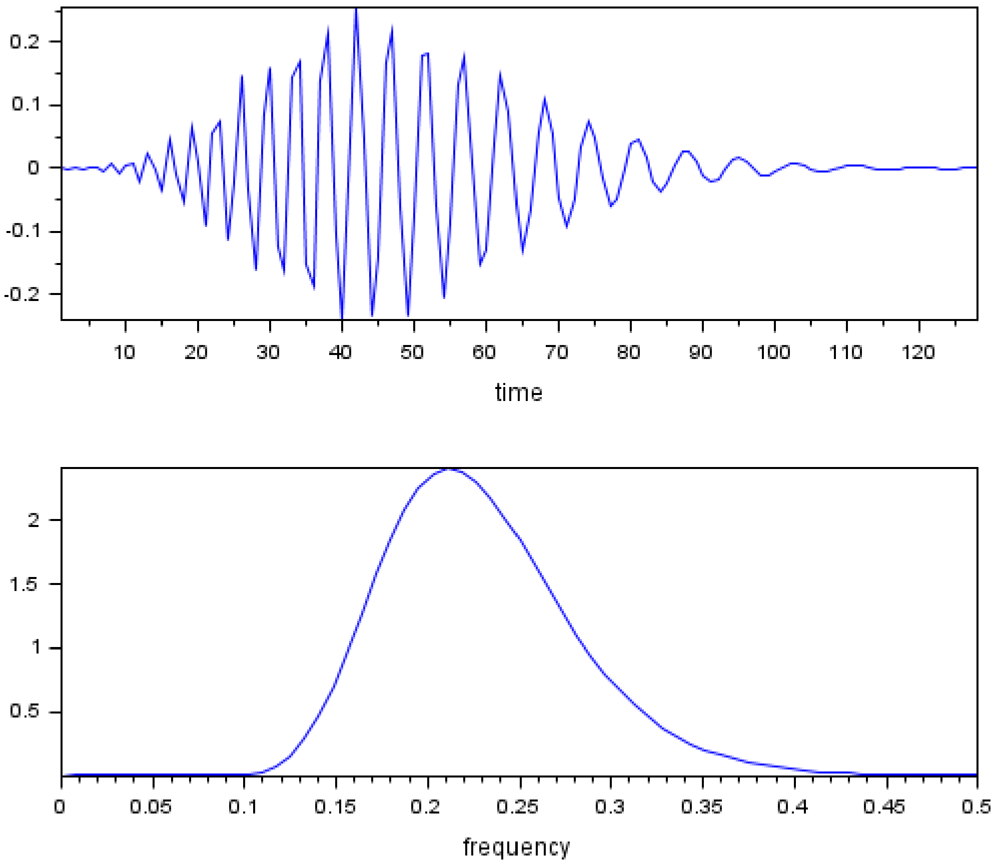
Description

altes generates the Altes signal in the time domain.

Examples

```
N = 128;
x = altes(N,0.1,0.45);
clf
subplot(211); plot(1:N,x); xlabel(_("time"))
y = fft(x);
f = (0:(N/2))/N; Nf=size(f,"*");
subplot(212); plot(f',abs(y(1:Nf)));
xlabel(_("frequency"))
```

```
// plots an Altes signal of 128 points whose normalized
frequency goes from 0.45 down to 0.1.
```



See also

- klauder — Klauder wavelet in time domain
- anasing — Lipschitz singularity
- anapulse — Analytic projection of unit amplitude impulse signal
- anastep — Analytic projection of unit step signal
- doppler — Generate complex Doppler signal

Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves - September 1995.

anaask

Amplitude Shift Keying (ASK) signal

Calling Sequence

```
[Y,AM] = anaask(N)
[Y,AM] = anaask(N, NCOMP)
[Y,AM] = anaask(N, NCOMP, F0)
```

Parameters

N:

a positive integer: the signal length.

NCOMP :

number of points of each component (default: N/5)

F0:

a real scalar in [0 0.5]: then normalized frequency (default: 0.25)

Y:

a complex column vector of length N: the signal

AM:

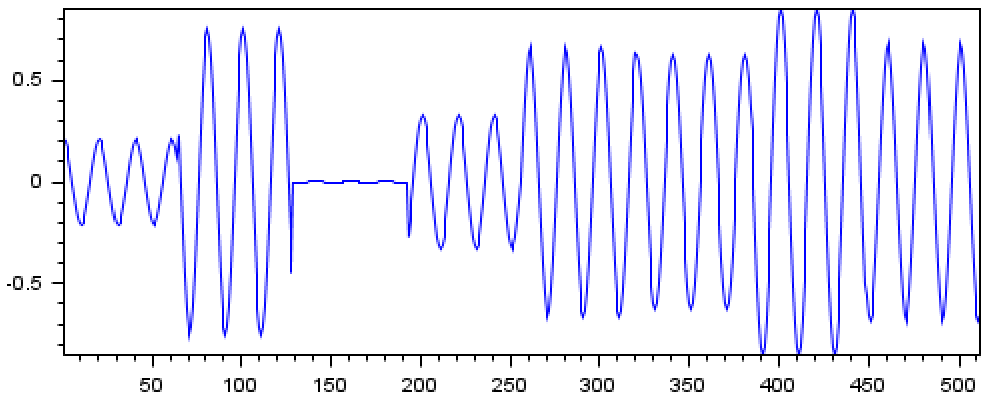
a real column vector of length N: the resulting amplitude modulation.

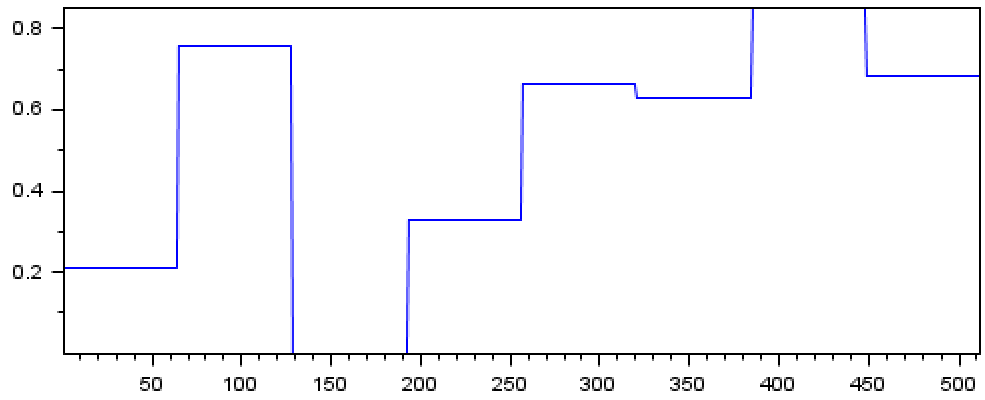
Description

anaask returns a complex amplitude modulated signal of normalized frequency F0, with a uniformly distributed random amplitude. Such signal is only 'quasi'-analytic.

Examples

```
[signal,am] = anaask(512,64,0.05);
clf
subplot(211); plot(real(signal));
subplot(212); plot(am);
```





See also

- anafsk — Frequency Shift Keying (FSK) signal
- anabpsk — Binary Phase Shift Keying (BPSK) signal
- anaqpsk — Quaternary Phase Shift Keying (QPSK) signal

Authors

- H. Nahrstaedt - Aug 2010
- O. Lemoine - October 1995
- Copyright (c) 1996 by CNRS (France).

anabpsk

Binary Phase Shift Keying (BPSK) signal

Calling Sequence

```
[Y,AM] = anabpsk(N)
[Y,AM] = anabpsk(N, NCOMP)
[Y,AM] = anabpsk(N, NCOMP, F0)
```

Parameters

N:

a positive integer: the signal length.

NCOMP :

number of points of each component (default: N/5)

F0:

a real scalar in [0 0.5]: then normalized frequency (default: 0.25)

Y:

a complex column vector of length N: the signal

AM:

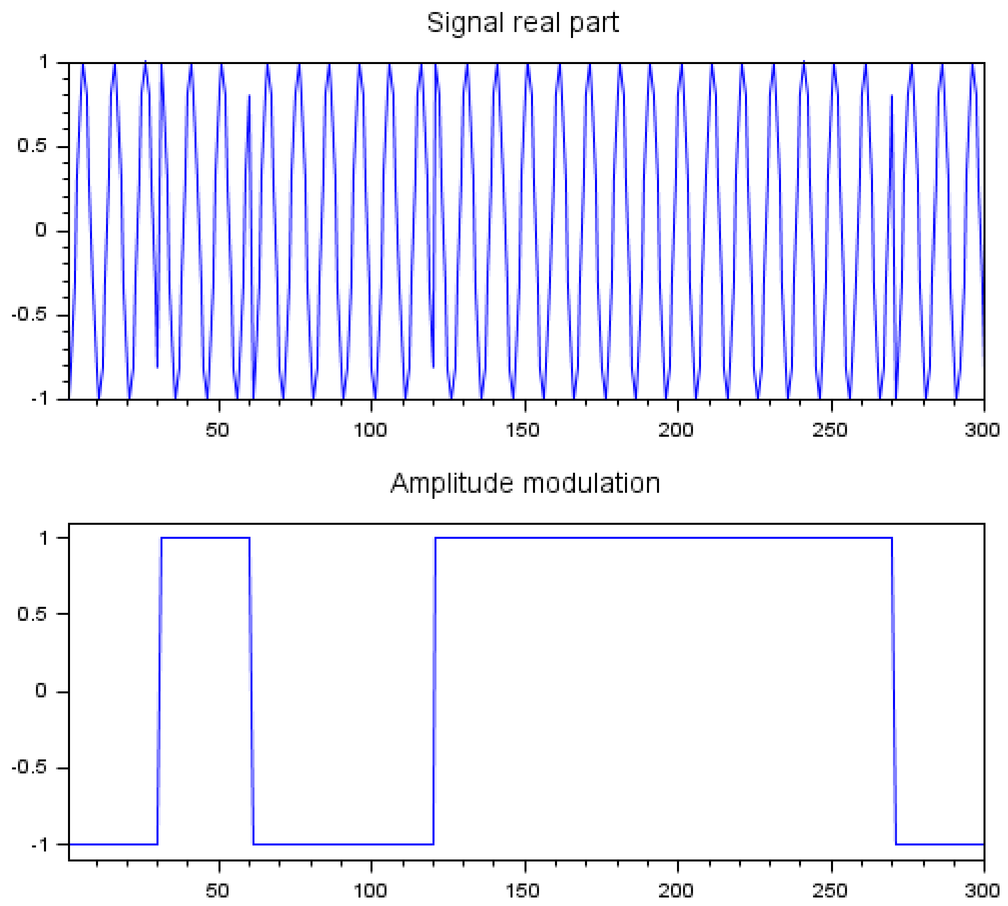
a real column vector of length N: the resulting amplitude modulation.

Description

anabpsk returns a succession of complex sinusoids of NCOMP points each, with a normalized frequency F0 and an amplitude equal to -1 or +1, according to a discrete uniform law. Such signal is only 'quasi'-analytic.

Examples

```
rand("seed",0);
[signal,am] = anabpsk(300,30,0.1);
clf
subplot(211); plot(real(signal));xtitle(_("Signal real
part"));
subplot(212); plot(am); gca().data_bounds(:,2)=[-1.1
1.1];
xtitle(_("Amplitude modulation"));
```



See also

- anafsk — Frequency Shift Keying (FSK) signal
- anaqpsk — Quaternary Phase Shift Keying (QPSK) signal
- anaask — Amplitude Shift Keying (ASK) signal

Authors

- O. Lemoine - June 1995, F. Auger - August 1995.
- Copyright (c) 1996 by CNRS (France).

anafsk

Frequency Shift Keying (FSK) signal

Calling Sequence

```
[Y,IFLAW] = ANAFSK(N, NCOMP, NBF)
```

Parameters

N:

a positive integer: the signal length.

NCOMP :

number of points of each component (default: N/5)

NBF:

a positive integer: the number of distinct frequencies (default: 4).

Y:

a complex column vector of length N: the signal

IFLAW:

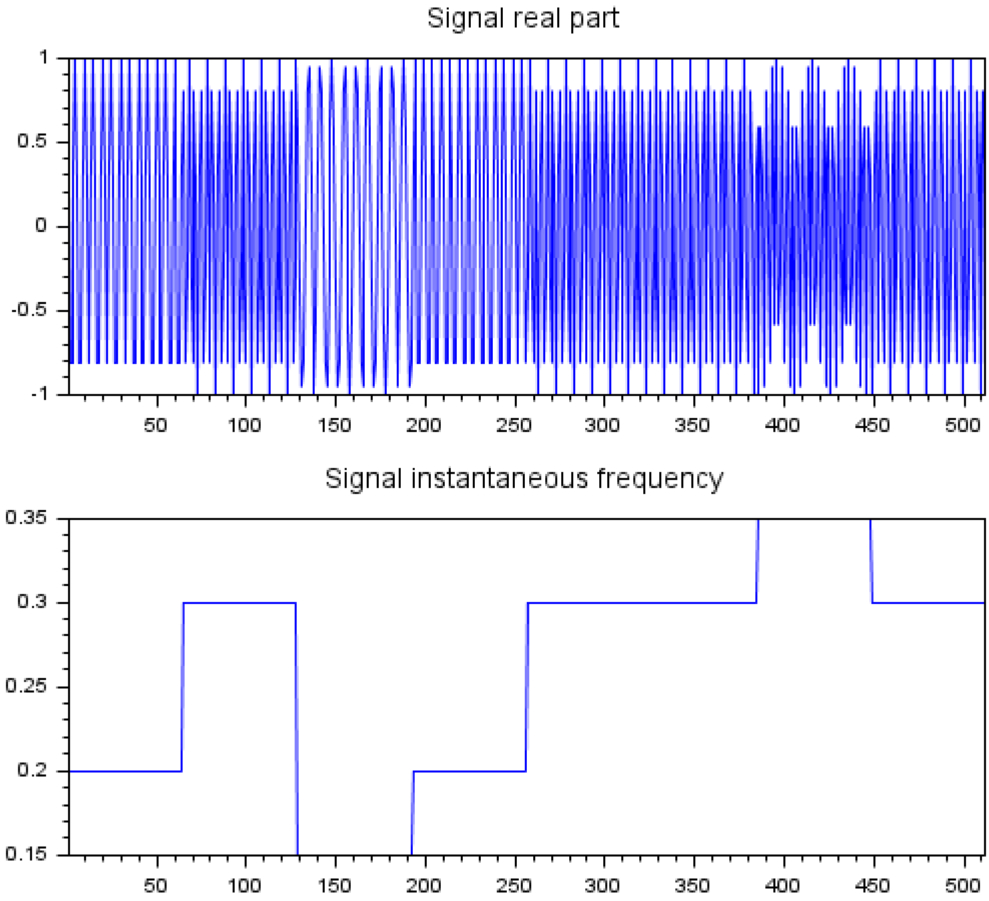
a real column vector of length N: the resulting instantaneous frequency.

Description

simulates a phase coherent Frequency Shift Keying (FSK) signal. This signal is a succession of complex sinusoids of NCOMP points each and with a normalized frequency uniformly chosen between NBF distinct values between 0.0 and 0.5. Such signal is only 'quasi'-analytic.

Examples

```
[signal,ifl] = anafsk(512,64,5);
clf
subplot(211); plot(real(signal));
subplot(212); plot(ifl);
```



See also

- anabpsk — Binary Phase Shift Keying (BPSK) signal
- anaqpsk — Quaternary Phase Shift Keying (QPSK) signal
- anaask — Amplitude Shift Keying (ASK) signal

Authors

- O. Lemoine - June 1995, F. Auger - August 1995.
- Copyright (c) 1996 by CNRS (France).

anapulse

Analytic projection of unit amplitude impulse signal

Calling Sequence

```
Y = anapulse(N)
Y = anapulse(N, TI)
```

Parameters

N :

a positive integer value: the signal length.

TI :

a positive integer value: the time position (index) of the impulse (default : round(N/2)).

Y:

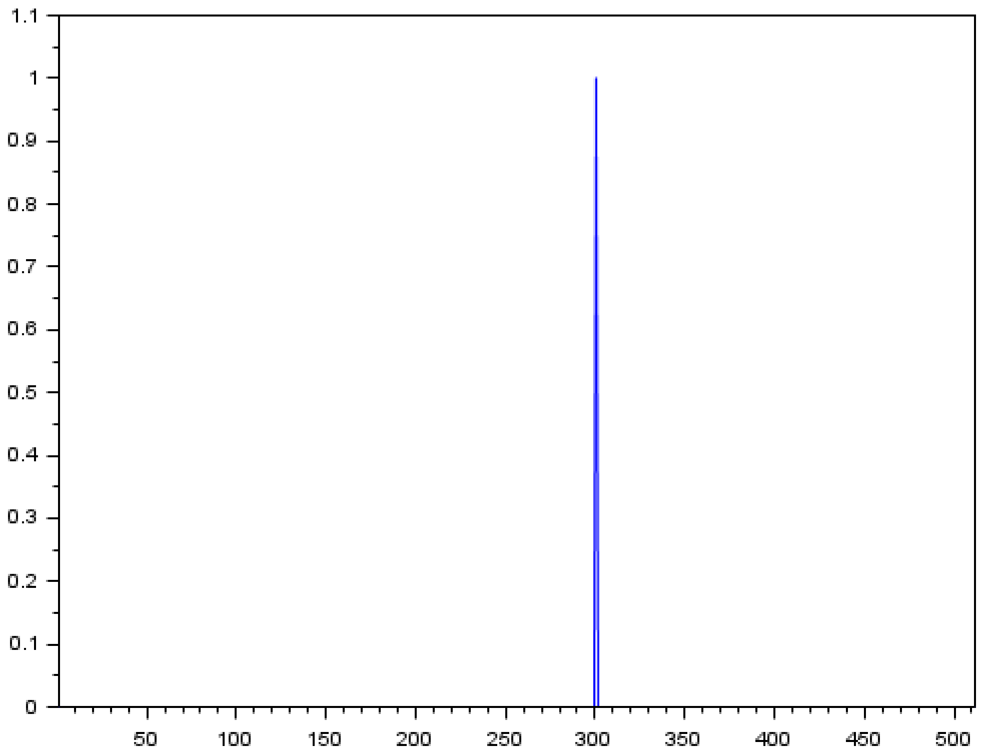
a complex column vector of length N: the signal samples.

Description

anapulse returns an analytic N-dimensional signal whose real part is a Dirac impulse at index t=TI. The Dirac heigth is 1.

Examples

```
signal = anapulse(512,301);
clf; plot(real(signal)); gca().data_bounds(2,2)=1.1;
```



See also

- anastep — Analytic projection of unit step signal
- anasing — Lipschitz singularity
- anabpsk — Binary Phase Shift Keying (BPSK) signal
- anafsk — Frequency Shift Keying (FSK) signal

Authors

- H. Nahrstaedt - Aug 2010
- O. Lemoine - June 1995, F. Auger, August 1995.
- Copyright (c) 1996 by CNRS (France).

anaqpsk

Quaternary Phase Shift Keying (QPSK) signal

Calling Sequence

```
[Y,PM] = anaqpsk(N)
[Y,PM] = anaqpsk(N, NCOMP)
[Y,PM] = anaqpsk(N, NCOMP, F0)
```

Parameters

N:

a positive integer: the signal length.

NCOMP :

number of points of each component (default: N/5)

F0:

a real scalar in [0 0.5]: then normalized frequency (default: 0.25)

Y:

a complex column vector of length N: the signal

PM0 :

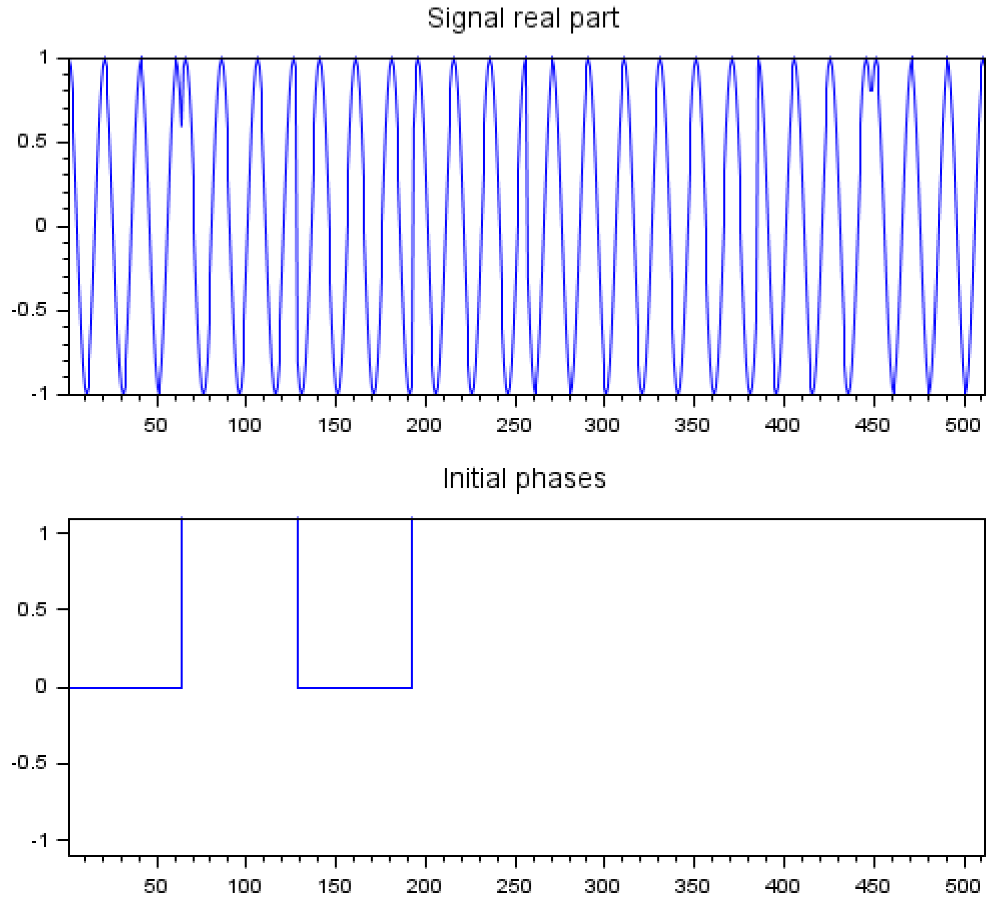
initial phase of each component (optional).

Description

anaqpsk returns a complex phase modulated signal of normalized frequency F0, whose phase changes every NCOMP point according to a discrete uniform law, between the values (0, %pi/2, pi, 3*%pi/2). Such signal is only 'quasi'-analytic.

Examples

```
rand("seed",0);
[signal,pm0] = anaqpsk(512,64,0.05);
clf
subplot(211); plot(real(signal));
subplot(212); plot(pm0); gca().data_bounds(1,2)=-0.1;
```



See also

- anafsk — Frequency Shift Keying (FSK) signal
- anabpsk — Binary Phase Shift Keying (BPSK) signal
- anaask — Amplitude Shift Keying (ASK) signal

Authors

- H. Nahrstaedt - Aug 2010
- O. Lemoine - October 1995
- Copyright (c) 1996 by CNRS (France).

anasing

Lipschitz singularity

Calling Sequence

```
X = anasing(N)
X = anasing(N, T0)
X = anasing(N, T0, H)
```

Parameters

N :

a positive integer: the number of points in time

T0 :

an integer value in]0 N] : the index of time localization of the singularity (default : round(N/2)).

H :

a real scalar: the strenght of the Lipschitz singularity (positive or negative) (default : 0.0)

X :

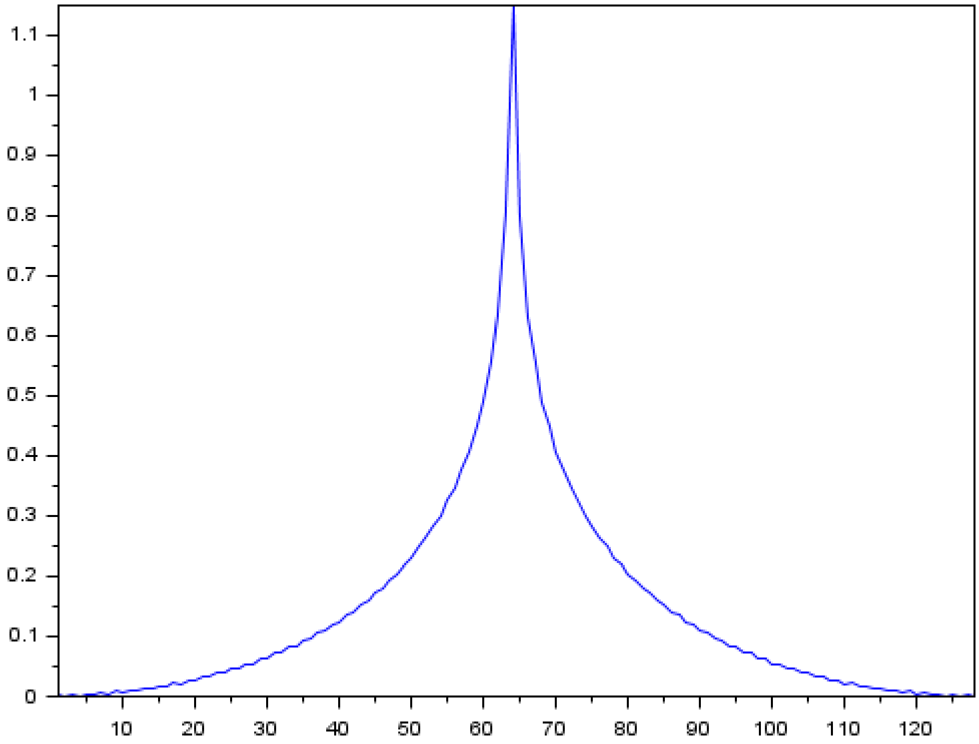
a complex row vector: the signal samples

Description

anasing generates the N-points Lipschitz singularity centered around T=T0 : X(T) = |T-T0|^H.

Examples

```
x = anasing(128);
clf(); plot(real(x));
```



See also

- anastep — Analytic projection of unit step signal
- anapulse — Analytic projection of unit amplitude impulse signal
- anabpsk — Binary Phase Shift Keying (BPSK) signal
- doppler — Generate complex Doppler signal

Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves - September 1995
- Copyright (c) 1995 Rice University

anastep

Analytic projection of unit step signal

Calling Sequence

```
Y = anastep(N)
Y = anastep(N, TI)
```

Parameters

N :

a positive integer value: the signal length.

TI :

a positive integer value: the time position (index) of the impulse (default : round(N/2)).

Y :

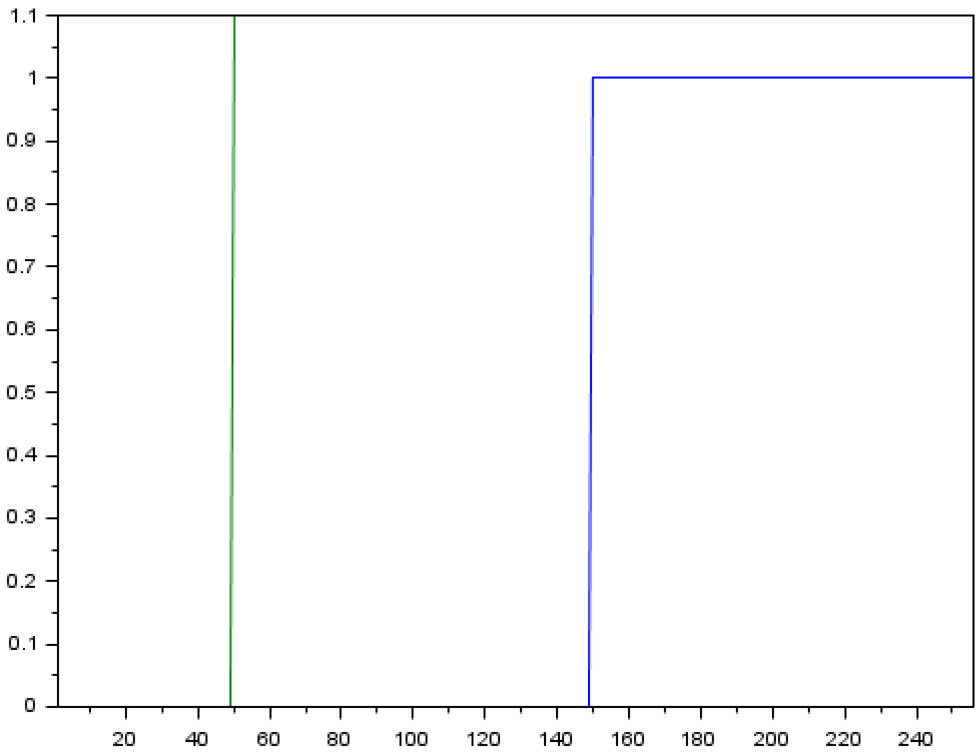
a complex column vector of length N: the signal samples.

Description

anastep generates the analytic projection of a unit step signal. The step heigth is 1.

Examples

```
s1 = anastep(256,150);
s2 = 1.1*anastep(256,50);
clf; plot(real([s1 s2]));
```



See also

- anasing — Lipschitz singularity
- anafsk — Frequency Shift Keying (FSK) signal
- anabpsk — Binary Phase Shift Keying (BPSK) signal
- anaqpsk — Quaternary Phase Shift Keying (QPSK) signal
- anaask — Amplitude Shift Keying (ASK) signal

Authors

- H. Nahrstaedt - Aug 2010
- O. Lemoine - June 1995, F. Auger, August 1995.
- Copyright (c) 1996 by CNRS (France).

atoms

Linear combination of elementary Gaussian atoms

Calling Sequence

```
[SIG,LOCATOMS] = atoms(N, COORD, DISPLAY)
```

Parameters

N :

number of points of the signal

COORD :

a real matrix with four columns: the time-frequency centers, the ith row has the form [ti,fi,Ti,Ai].

(ti,fi) are the time-frequency coordinates of atom i.

Ti is its time duration and Ai its amplitude.

Frequencies f1..fM should be normalized (between 0 and 0.5).

If nargin==1, the location of the atoms will be defined by clicking with the mouse, with the help of a menu. The default value for Ti is N/4.

SIG:

a column complex vector of length N: the output signal.

LOCATOMS :

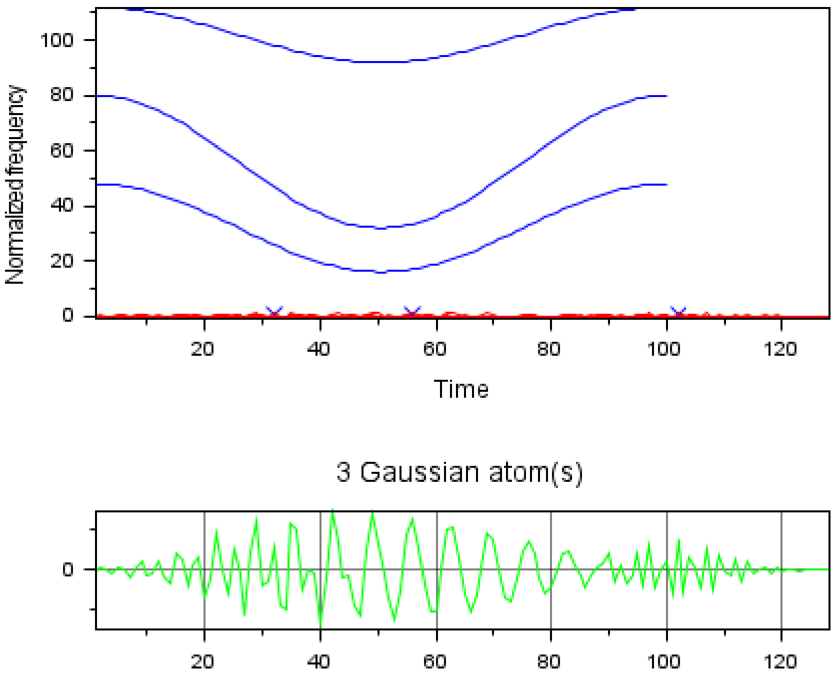
a real matrix with four columns: matrix of time-frequency coordinates and durations of the atoms (see COORD).

Description

atoms generates a signal consisting in a linear combination of elementary gaussian wave packets. The locations of the time-frequency centers of the different atoms are either fixed by the input parameter COORD or successively defined by clicking with the mouse (if NARGIN==1).

Examples

```
clf
coord=[32, 0.3,  32, 1    ;
       56, 0.15, 48, 1.22;
       102,0.41, 20, 0.7 ];
sig = atoms(128,coord,1);
```



Authors

- H. Nahrstaedt - Aug 2010
- P. Flandrin, May 1995 - O. Lemoine, February 1996.
- F. Auger - O. Lemoine, June 1996.
- E. Chassande-Mottin, F. Auger, May 1998.

chirp

Evaluate a chirp signal (frequency swept cosine wave) at time t.

Calling Sequence

```
y = chirp(t)
y = chirp(t, f0)
y = chirp(t, f0, t1)
y = chirp(t, f0, t1, f1)
y = chirp(t, f0, t1, f1, form)
y = chirp(t, f0, t1, f1, form, phase)
```

Parameters

t:

real vector of times (in seconds) to evaluate the chirp signal

f0:

a scalar ≥ 0: the frequency (in Hz) at time t=0 (default: 0). If set to [] the default value is used.

t1:

a scalar ≥ 0: the time (in seconds) t1 (default: 1). If set to [] the default value is used.

f1:

a scalar ≥ 0: frequency (in Hz) at time t=t1 (default: 100). If set to [] the default value is used.

form:

a character string: the shape of frequency sweep;

- 'linear' : f(t) = (f1-f0)*(t/t1) + f0,
- 'quadratic': f(t) = (f1-f0)*(t/t1)^2 + f0,
- 'logarithmic': f(t) = (f1-f0)^(t/t1) + f0

The default value is "linear". If set to [] the default value is used.

phase:

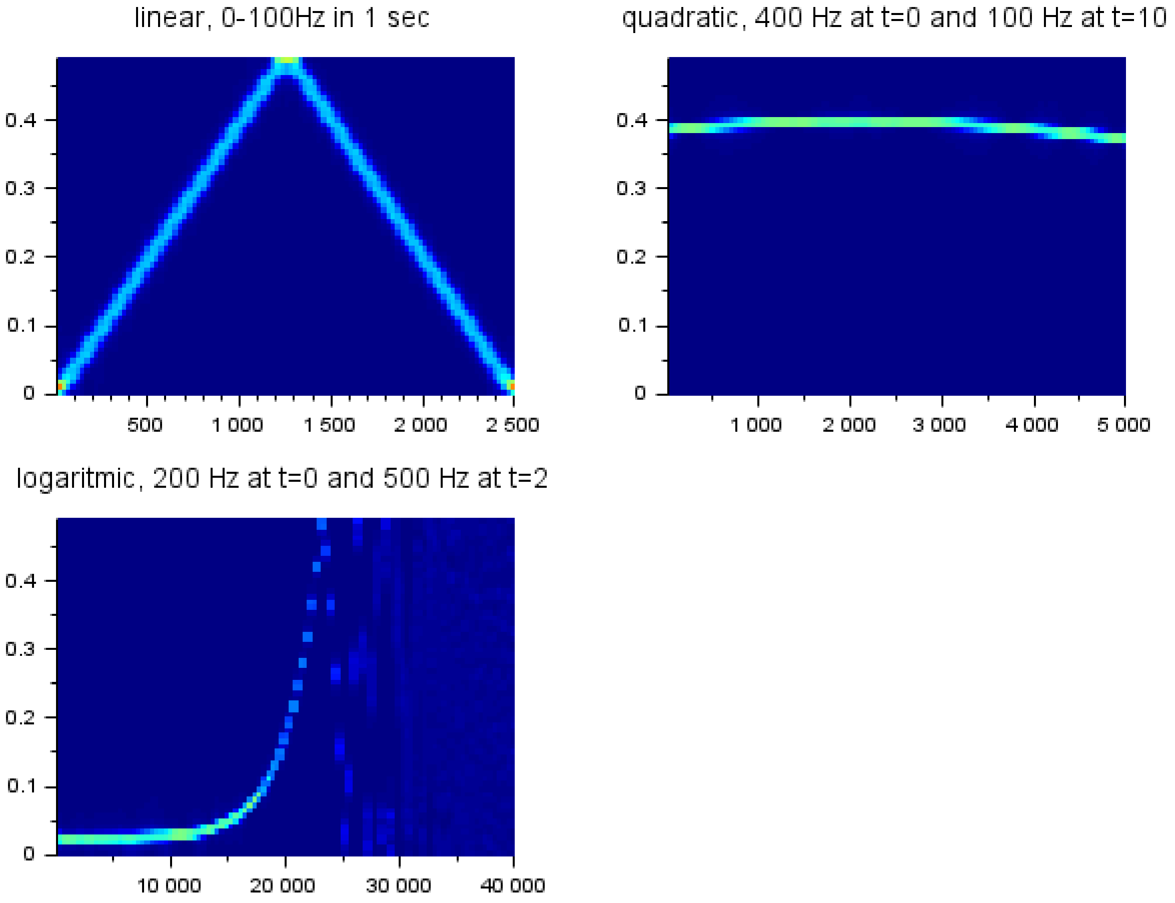
a real scalar: the phase shift (in degree) at t=0 (default: 0). If set to [] the default value is used.

Description

If you want a different sweep shape f(t), use the following: y = cos(2*%pi*integral(f(t)) + 2*%pi*f0*t + phase);

Examples

```
clf; gcf().color_map = jetcolormap(128);
// Linear
subplot(221);
[TFR,T,F] = tfrsp(chirp([0:0.002:5])',1:2:2501,128);
subTFR = linspace(1, size(TFR, 'c'), 100);
grayplot(T(subTFR),F(1:$/2),TFR(1:$/2,subTFR)');
xtitle "linear, 0-100Hz in 1 sec"

//Quadratic
subplot(222);
[TFR,T,F] = tfrsp(chirp([-2:0.001:15], 400, 10, 100,
'quadratic')',1:2:5001,128);
subTFR = linspace(1, size(TFR, 'c'), 100);
grayplot(T(subTFR),F(1:$/2),TFR(1:$/2,subTFR)');
xtitle "quadratic, 400 Hz at t=0 and 100 Hz at t=10"

//Logarithmic
subplot(223);
[TFR,T,F] = tfrsp(chirp([0:1/8000:5], 200, 2, 500,
"logarithmic")',1:20:40001,128);
subTFR = linspace(1, size(TFR, 'c'), 100);
grayplot(T(subTFR),F(1:$/2),TFR(1:$/2,subTFR)');
xtitle "logaritmic, 200 Hz at t=0 and 500 Hz at t=2"
```



Authors

- 2001-08-31 Paul Kienzle pkienzle@users.sf.net

dopnoise

Complex noisy doppler signal

Calling Sequence

```
[Y,IFLAW] = dopnoise(N, FS, F0, D, V, T0, C)
```

Parameters

N :

a positive integer: the number of points.

FS :

a positive scalar : the sampling frequency (in Hertz).

F0 :

a scalar in]0 FS/2]: the target frequency (in Hertz).

D :

a positive scalar: the distance from the line to the observer (in meters).

V :

a positive scalar: the target velocity (in m/s).

T0 :

a positive integer in [1 N]: the time center index (default : round(N/2)).

C :

a positive scalar: the wave velocity (in m/s) (default : 340).

Y :

Output signal.

IFLAW :

Model used as instantaneous frequency law.

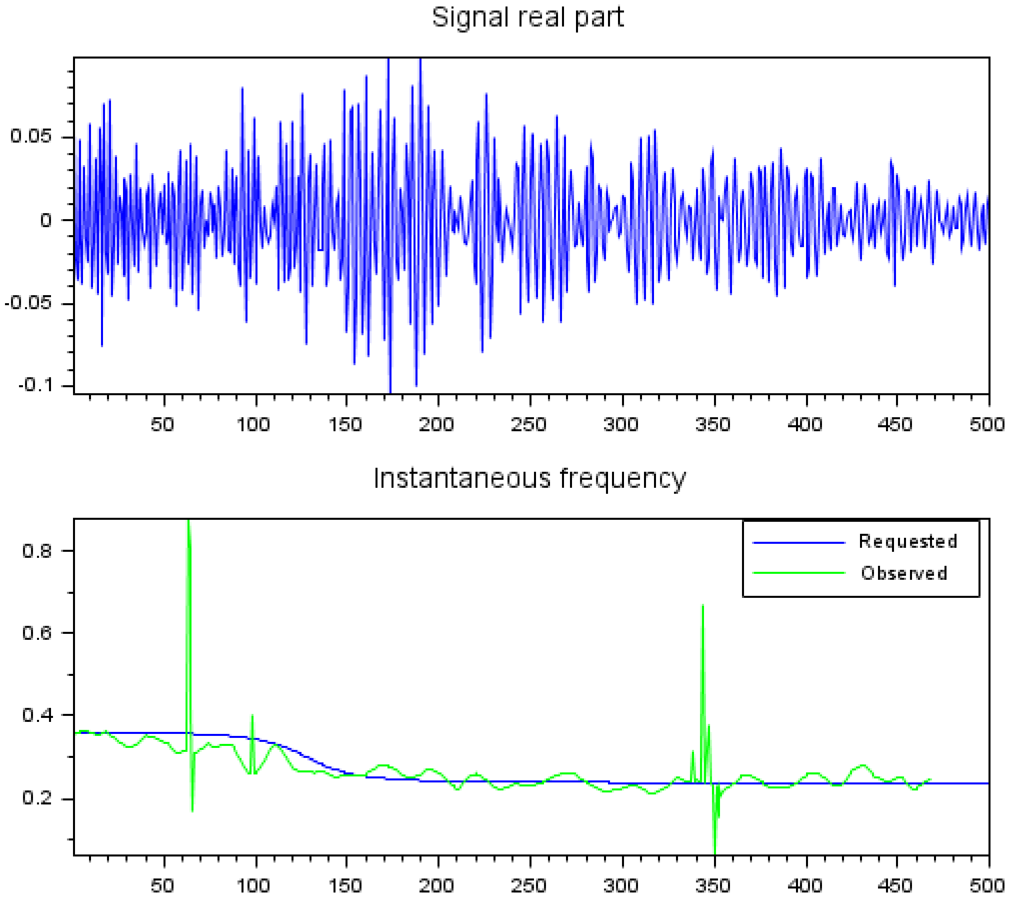
Description

dopnoise generates a complex noisy doppler signal, normalized so as to be of unit energy.

Examples

```
rand("seed",0)
```

```
[z,iflaw] = dopnoise(500,200,60,10,70,128);
clf
subplot(211); plot(real(z)); xtitle(_("Signal real
part"))
subplot(212); plot(iflaw);
ifl = instfreq(z,11:478,10);
plot(ifl,'g');

xtitle(_("Instantaneous frequency"))
legend([_("Requested"),_("Obtained")]);

sum(abs(z).^2) // check energy
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, July 94, August 95.

doppler

Generate complex Doppler signal

Calling Sequence

```
[FM,AM,IFLAW] = doppler(N, FS, F0, D, V)
[FM,AM,IFLAW] = doppler(N, FS, F0, D, V, T0)
[FM,AM,IFLAW] = doppler(N, FS, F0, D, V, T0, C)
```

Parameters

N :

a positive integer: the number of points.

FS :

a positive scalar : the sampling frequency (in Hertz).

F0 :

a scalar in]0 FS/2]: the target frequency (in Hertz).

D :

a positive scalar: the distance from the line to the observer (in meters).

V :

a positive scalar: the target velocity (in m/s).

T0 :

a positive integer in [1 N]: the time center index (default : round(N/2)).

C :

a positive scalar: the wave velocity (in m/s) (default : 340).

FM :

a real column vector of size N: the output frequency modulation.

AM :

a real column vector of size N: the output amplitude modulation.

IFLAW :

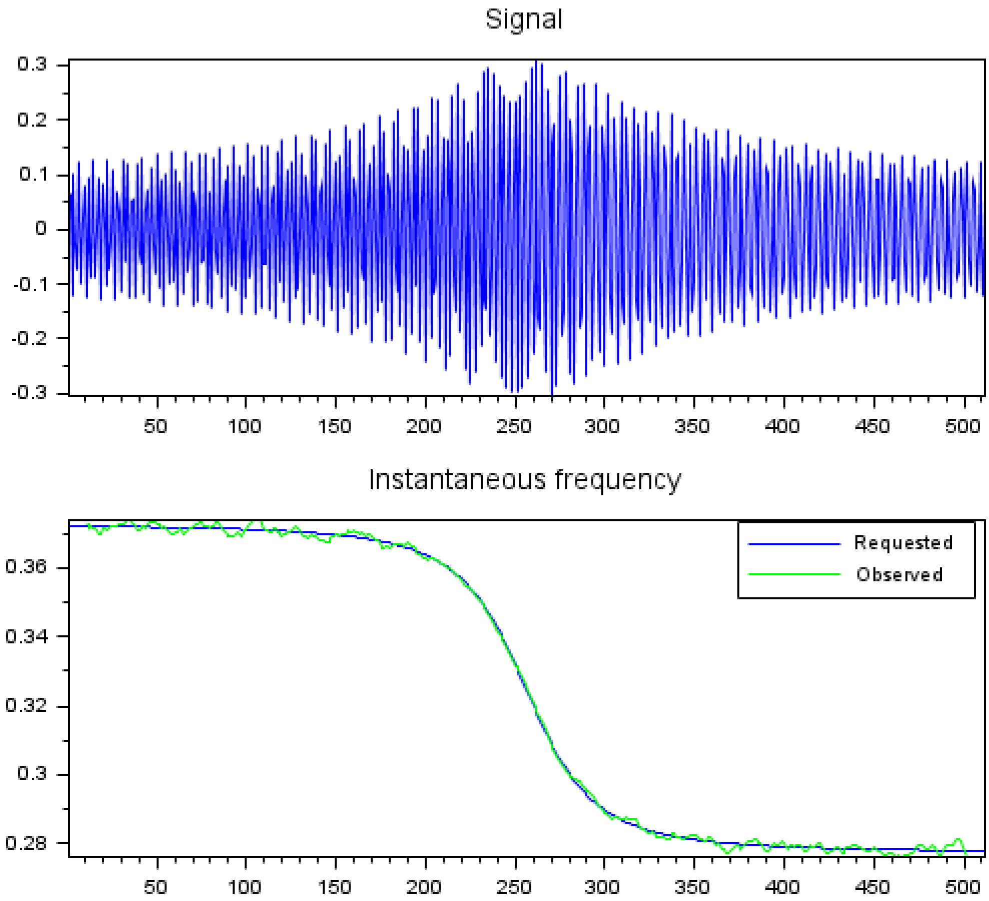
a real column vector of size N: the output instantaneous frequency law.

Description

Returns the frequency modulation (FM), the amplitude modulation (AM) and the instantaneous frequency law (IFLAW) of the signal received by a fixed observer from a moving target emitting a pure frequency f0.

Examples

```
rand("seed",0)
N = 512;
[fm,am,iflaw] = doppler(N,200,65,10,50);
clf
subplot(211); plot(real(am.*fm));xtitle(_("Signal "))
subplot(212); plot(iflaw);
[ifhat,t] =
instfreq(sigmerge(am.*fm,noisecg(N),15),11:502,10);
plot(t,ifhat,'g');
xtitle(_("Instantaneous frequency"))
legend([_("Requested"),_("Observed")]);
```



See also

- dopnoise — Complex noisy doppler signal

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, July 94, August 95 - O. Lemoine, October 95.

griffitc

Test signal example C of Griffiths' paper

Calling Sequence

```
[SIG,IFLAWS] = griffitc(N)
[SIG,IFLAWS] = griffitc(N, SNR)
```

Parameters

N :

a positive integer: number of points in time of the signal (default: 200)

SNR :

a positive scalar: the signal to noise ratio (default: 25 dB).

SIG :

a complex column vector of size N: the signal.

IFLAWS :

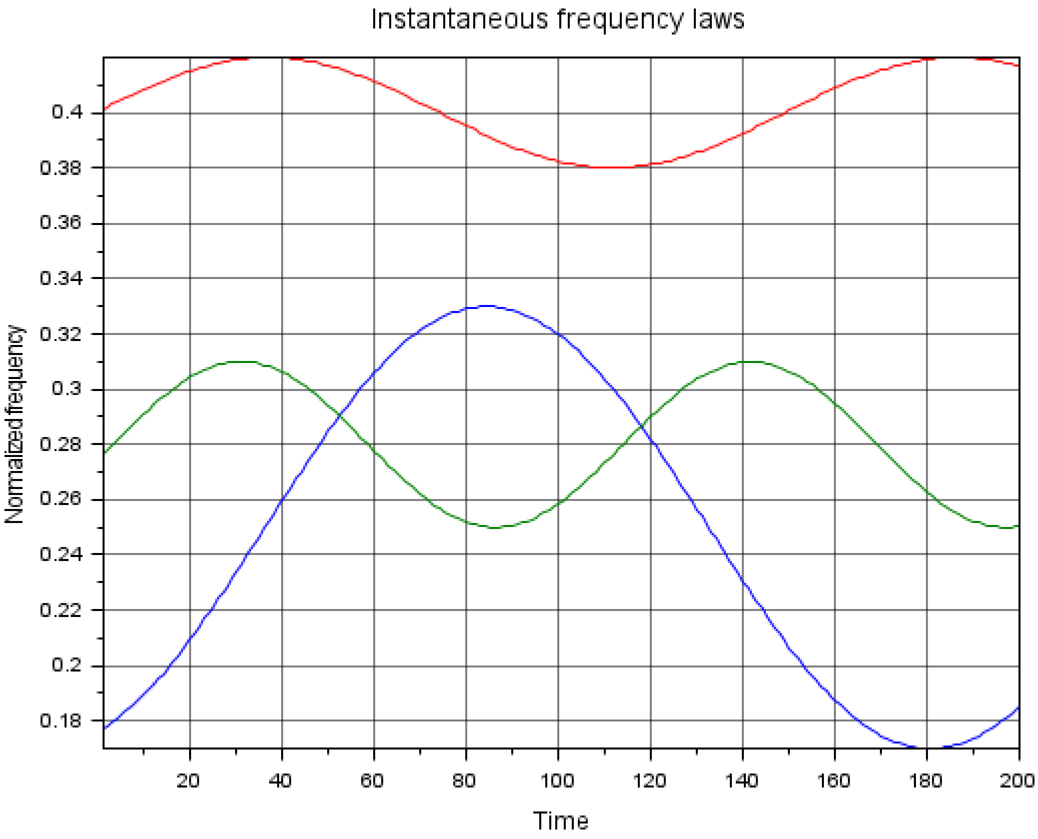
a N by 3 real array : the instantaneous frequencies of the 3 components .

Description

griffitc generates the test signal of the example C of the paper of Griffiths.

Examples

```
[sig,iflaws] = griffitc();
plot(iflaws); xgrid;
xlabel(_("Time"));
ylabel(_("Normalized frequency"))
xtitle(_("Instantaneous frequency laws"))
```



References

L.J. Griffiths, "Rapid measurement of digital instantaneous frequency", IEEE Trans on ASSP, Vol 23, No 2, pp 207-222, 1975.

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, July 1995.

klauder

Klauder wavelet in time domain

Calling Sequence

```
X = klauder(N)
X = klauder(N, LAMBDA)
X = klauder(N, LAMBDA, F0)
```

Parameters

N :

a positive scalar with integer value: the number of points in time

LAMBDA :

a positive scalar: the attenuation factor or the envelope (default : 10)

F0 :

a scalar in [0 0.5]: the central frequency of the wavelet (default : 0.2)

X :

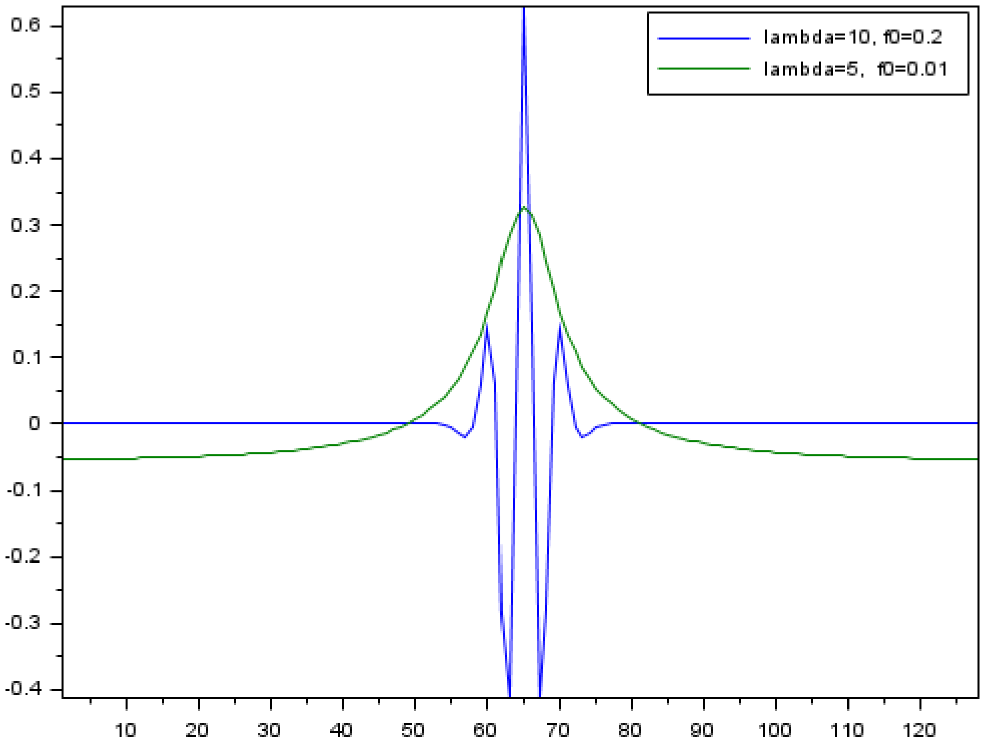
a real column vector of size N: the klauder time samples.

Description

klauder generates the KLAUDER wavelet in the time domain K(f) = e^{-2.pi.LAMBA.f} f^{2.pi.LAMBDA.F0-1/2}

Examples

```
x1 = klauder(128);
x2 = klauder(128,5,0.01);
clf; plot([x1 x2]);
legend("lambda=10, f0=0.2","lambda=5,  f0=0.01");
```



See also

- altes — Altes signal in time domain
- anasing — Lipschitz singularity
- doppler — Generate complex Doppler signal
- anafsk — Frequency Shift Keying (FSK) signal
- anastep — Analytic projection of unit step signal

Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves 9-95

mexhat

Mexican hat wavelet in time domain

Calling Sequence

```
[H,T] = mexhat()
[H,T] = mexhat(NU)
```

Parameters

NU :

a scalar in [0 0.5] (default : 0.05).

H :

a real row vector of length 2*ceil(1.5/NU)+1: the the mexhat time samples.

T :

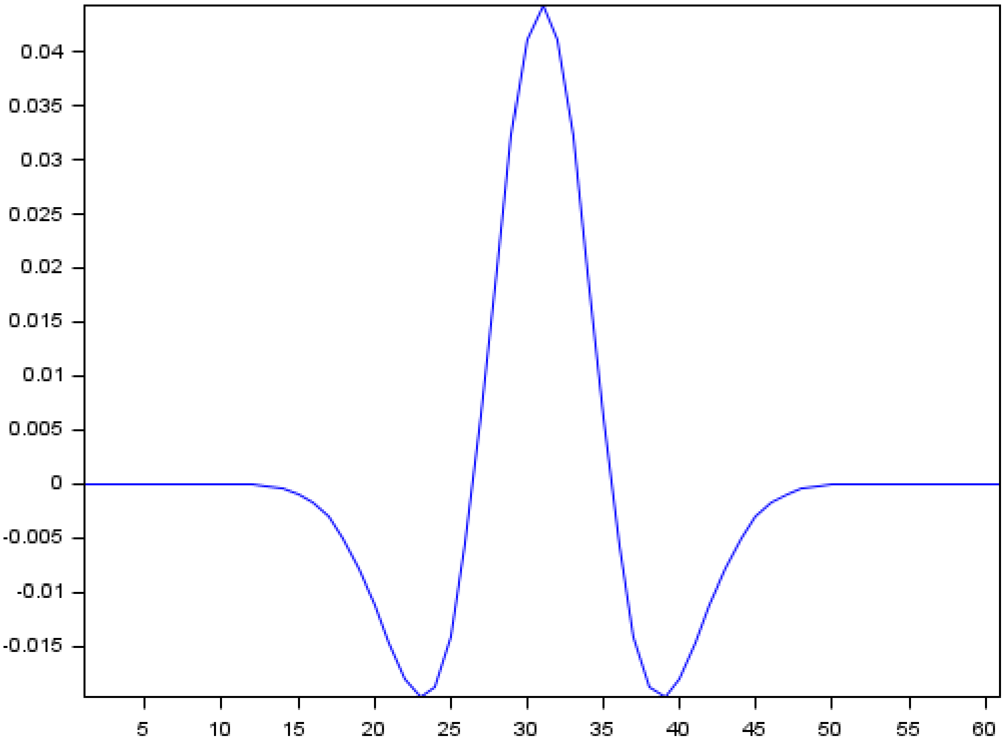
a real row vector of length 2*ceil(1.5/NU)+1: the mexican hat time support.

Description

mexhat returns the mexican hat wavelet with central frequency NU (NU is a normalized frequency in Hz).

Examples

```
clf
plot(mexhat(0.05));
```



See also

- klauder — Klauder wavelet in time domain

Authors

- H. Nahrstaedt - Aug 2010
- P. Goncalves, October 95

pulstran

pulse generation

Calling Sequence

```
y = pulstran(t, d, func,...)
y = pulstran(t, d, p, Fs, 'interp')
```

Parameters

'rectpuls' :

rectangular pulse, parameter are (w) (default w=1)

'tripuls' :

triangular pulse, parameter are (w, skew) (default w=1,skew=0)

'gmonopuls':

gaussian monopulse, parameter are (fc) (default fc=1e3)

'gauspuls':

Gaussian modulated sinusoidal pulse, parameter are (fc, bw) (default fc=1e3, bw=0.5)

Description

Generate the signal y=sum(func(t+d,...)) for each d. If d is a matrix of two columns, the first column is the delay d and the second column is the amplitude a, and y=sum(a*func(t+d)) for each d,a. Clearly, func must be a function which accepts a vector of times. Any extra arguments needed for the function must be tagged on the end.

If instead of a function name you supply a pulse shape sampled at frequency Fs (default 1 Hz), an interpolated version of the pulse is added at each delay d. The interpolation stays within the the time range of the delayed pulse. The interpolation method defaults to linear, but it can be any interpolation method accepted by the function interp1.

Examples

```
fs = 11025;  // arbitrary sample rate
f0 = 100;    // pulse train sample rate
w = 0.001;   // pulse width of 1 millisecond
plot(pulstran(0:1/fs:0.1, 0:1/f0:0.1, 'rectpuls', w));

fs = 11025;  // arbitrary sample rate
f0 = 100;    // pulse train sample rate
w = ones(10,1);  // pulse width of 1 millisecond at 10
```

```
kHz
plot(pulstran(0:1/fs:0.1, 0:1/f0:0.1, w, 10000));
```

Authors

- Copyright (C) 2000 Paul Kienzle

sawtooth

Generates a sawtooth wave

Calling Sequence

```
y = sawtooth(t)
y = sawtooth(t, width)
```

Parameters

t :

a real vector: the time vector.

width :

a scalar in [0 1].

Description

Generates a sawtooth wave of period 2 * pi with limits +1/-1 for the elements of t.

width is a real number between 0 and 1 which specifies the point between 0and 2 * pi where the maximum is. The function increases linearly from -1 to 1 in [0, 2 * pi * width] interval, and decreases linearly from 1 to -1 in the interval[2 * pi * width, 2 * pi].

If width is 0.5, the function generates a standard triangular wave.

If width is not specified, it takes a value of 1, which is a standard sawtooth function.

Examples

```
t = linspace(0,20,3000);
clf
plot(t, sawtooth(t,0.2),"b")
plot(t, sawtooth(t,0.5),"r")
legend("width="+string([0.2 0.5]));
```

Authors

- Juan Aguado 2007

tftb_window

Window generation

Calling Sequence

```
H = tftb_window(N)
H = tftb_window(N, NAME)
H = tftb_window(N, NAME, PARAM)
H = tftb_window(N, NAME, PARAM, PARAM2)
H = tftb_window(...,'plot')
```

Parameters

N :

length of the window

NAME :

name of the window shape (default : Hamming)

PARAM :

optional parameter

PARAM2 :

second optional parameters

'plot':

if one input parameter is 'plot', the window will be plotted

Description

yields a window of length N with a given shape.

Possible names are :

'Hamming', 'Hanning', 'Nuttall', 'Papoulis', 'Harris',

'Rect', 'Triang', 'Bartlett', 'BartHann', 'Blackman'

'Gauss', 'Parzen', 'Kaiser', 'Dolph', 'Hanna'.

'Nutbess', 'spline', 'Flattop'

For the gaussian window, an optionnal parameter K sets the value at both extremities. The default value is 0.005

For the Kaiser-Bessel window, an optionnal parameter sets the scale. The default value is 3*pi.

For the Spline windows, h=tftb_window(N,'spline',nfreq,p) yields a spline weighting function of order p and frequency bandwidth proportional to nfreq.

Examples

```
h = tftb_window(256,'Gauss',0.005);
plot(0:255, h);
a = gca(); a.data_bounds=([0,255,-0.1,1.1]); xgrid
```

See also

- dwindow — Derive a window

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, June 1994 - November 1995.
- Copyright (c) 1996 by CNRS (France).

Time-Domain Processing

- ifestar2 — Instantaneous frequency estimation using AR2 modelisation
- instfreq — Instantaneous frequency estimation
- loctime — Time localization caracteristics

ifestar2

Instantaneous frequency estimation using AR2 modelisation

Calling Sequence

```
[FNORM,T2,RATIO] = ifestar2(X, T)
[FNORM,T2,RATIO] = ifestar2(X)
```

Parameters

X :

A real vector of size N: the signal to be analyzed.

T :

A vector with integer elements >= 4: Time instants (default : 4:N).

FNORM :

a real column vector: the (normalized) instantaneous frequency.

T2 :

a real column vector: the time instants coresponding to FNORM. Since the algorithm can not always give a value, T2 may be different of T.

RATIO :

a scalar in [0 1]: the proportion of instants where the algorithm yields an estimation

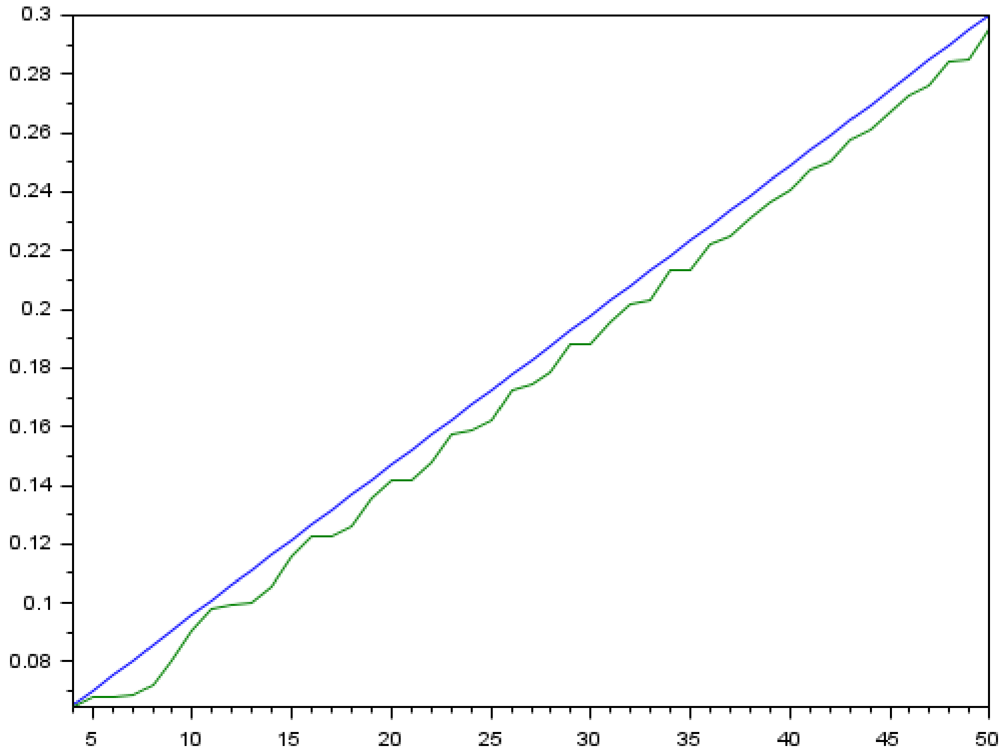
Description

ifestar2 computes an estimate of the instantaneous frequency of the real signal X at time instant(s) T. The result FNORM lies between 0.0 and 0.5. This estimate is based only on the 4 last signal points, and has therefore an approximate delay of 2.5 points.

Examples

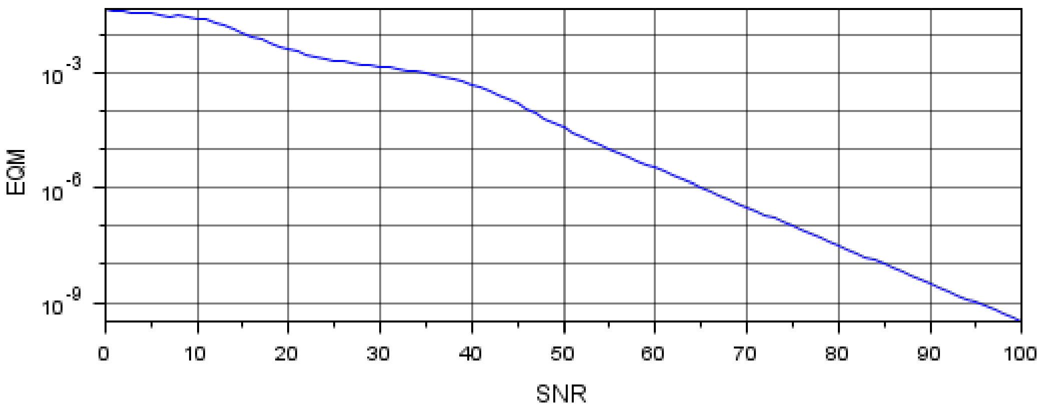
First example

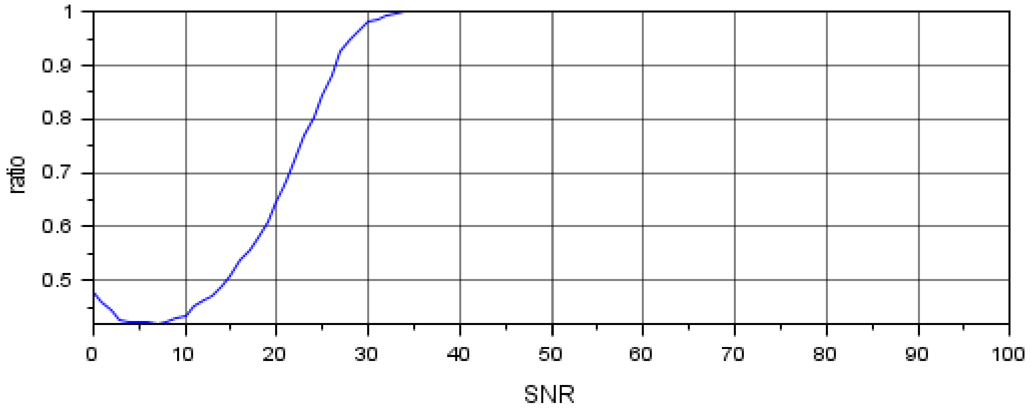
```
[x,if1] = fmlin(50,0.05,0.3,5);
[if2,t] = ifestar2(real(x));
clf; plot(t,if1(t),t,if2);
```



Second example

```
N=1100;
[deter,if1]=fmconst(N,0.05);
deter=real(deter);
noise=rand(N,1,'normal'); NbSNR=101;
SNR=linspace(0,100,NbSNR)';
for iSNR=1:NbSNR,
sig=sigmerge(deter,noise,SNR(iSNR));
[if2,t,ratio(iSNR)]=ifestar2(sig);
EQM(iSNR,1)=norm(if1(t)-if2)^2 / length(t)  ;
end;

clf();
subplot(211); plot(SNR,EQM); gca().log_flags="nl";xgrid;
xlabel('SNR'); ylabel('EQM');
subplot(212); plot(SNR,ratio); xgrid;
xlabel('SNR'); ylabel('ratio');
```





See also

- instfreq — Instantaneous frequency estimation
- kaytth — Kay-Tretter filter computation
- sgrpdlay — Group delay estimation of a signal

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, April 1996.

instfreq

Instantaneous frequency estimation

Calling Sequence

```
[FNORMHAT,T] = instfreq(X)
[FNORMHAT,T] = instfreq(X, T)
[FNORMHAT,T] = instfreq(X, T, L)
[FNORMHAT,T] = instfreq(X, T, L, TRACE)
```

Parameters

X :

a complex vector of size N: the analytic signal to be analyzed.

T :

a vector with integer elements in [L, N-L]: the Time instants (default : 2:length(X)-1).

L :

An integer >=1:

If L=1 (the default value) computes the (normalized) instantaneous frequency of the signal X defined as angle(X(T+1)*conj(X(T-1)) ;

if L>1, computes a Maximum Likelihood estimation of the instantaneous frequency of the deterministic part of the signal blurried in a white gaussian noise.

TRACE :

A boolean (or a real scalar) if true (or nonzero),the progression of the algorithm is shown (default : %f).

FNORMHAT :

a column vector with same size as T: the (normalized) instantaneous frequency.

T :

a row vector: the time instants.

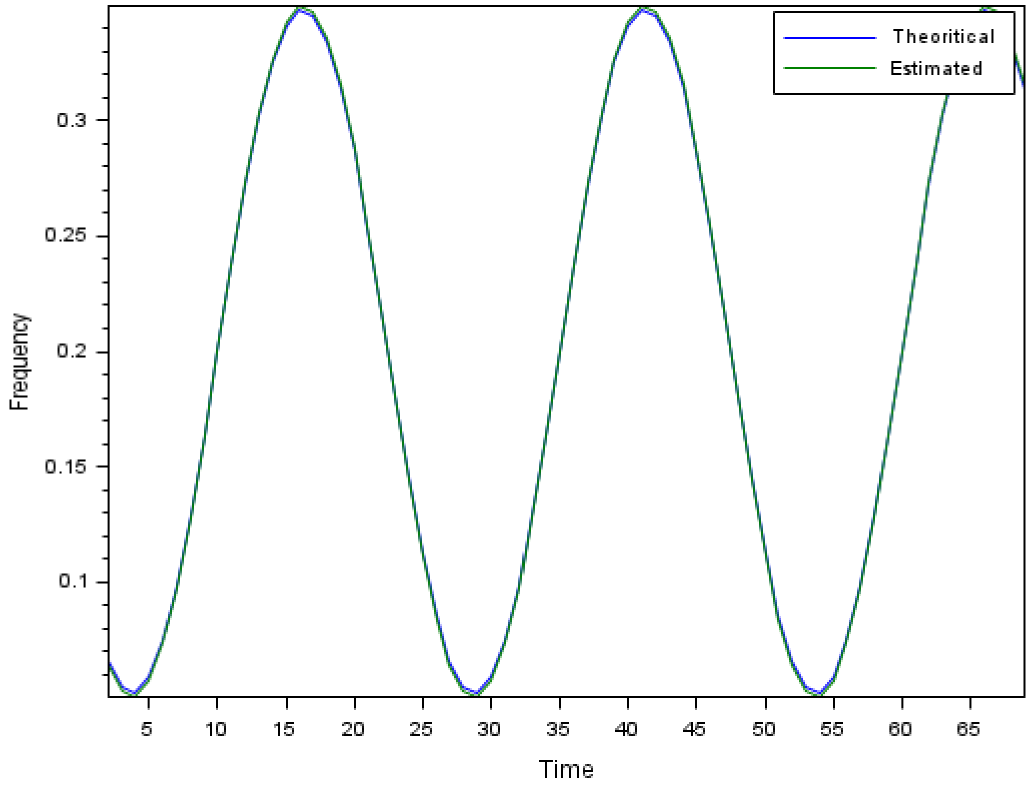
Description

instfreq computes the instantaneous frequency of the analytic signal X at time instant(s) T, using the trapezoidal integration rule. The result FNORMHAT lies between 0.0 and 0.5.

Examples

```
[x,iflaw] = fmsin(70,0.05,0.35,25);
```

```
[instf,t] = instfreq(x);
clf; plot(t',instf,t',iflaw(2:$-1))
xlabel(_("Time"))
ylabel(_("Frequency"))
legend(_("Theoritical"),_("Estimated"))
```



See also

- kaytth — Kay-Tretter filter computation
- sgrpdlay — Group delay estimation of a signal

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, March 1994, July 1995.

loctime

Time localization caracteristics

Calling Sequence

```
[TM,T] = loctime(SIG)
```

Parameters

SIG:

a real vector

TM :

a real scalar: the averaged time center

T :

a real scalar: the time spreading.

Description

loctime computes the time localization caracteristics of signal SIG.

Examples

```
z = amgauss(160,80,50);
[tm,T] = loctime(z)
```

See also

- locfreq — Frequency localization caracteristics

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, July 1995.

<< Time-Domain Processing

Visualization and backup

- plotifl — Plot normalized instantaneous frequency laws
- tfrparam — parametric time-frequency representation
- tfrqview — Quick visualization of time-frequency representations
- tfrsave — Save the parameters of a time-frequency representation
- tfrview — Visualization of time-frequency representations

plotifl

Plot normalized instantaneous frequency laws

Calling Sequence

```
plotifl(T, IFLAWS)
plotifl(T, IFLAWS, SIGNAL)
```

Parameters

T :

a real vector of size M: the time instants,

IFLAWS :

an M by P real matrix with values in [-0.5 0.5]: each column corresponds to the instantaneous frequency law of a signal.

SIGNAL :

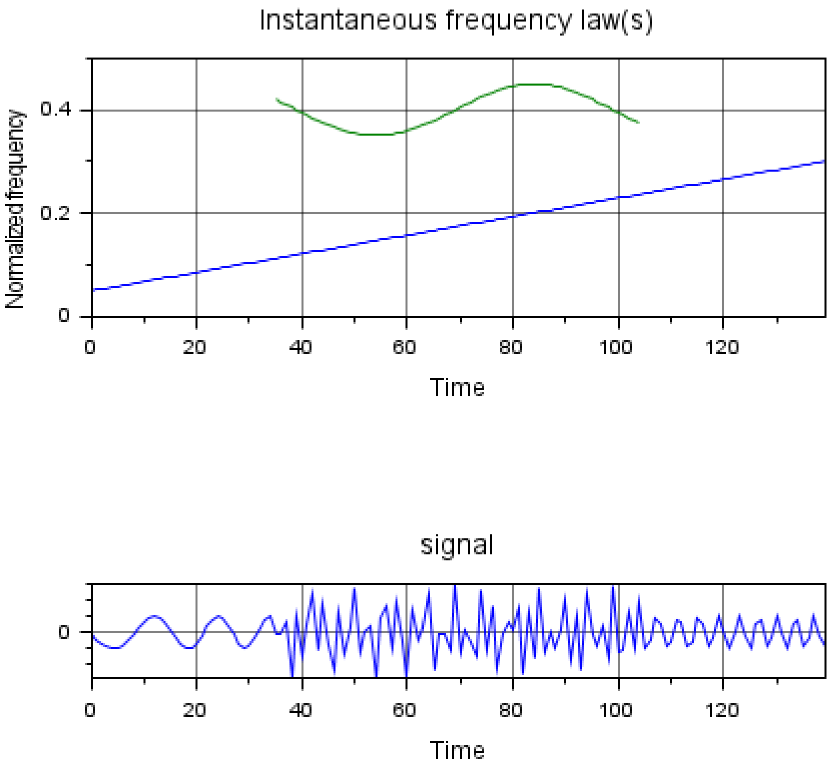
a real or complex vector of size M: if given a subwindow is drawn with plot(T,real(SIGNAL))

Description

plotifl plot the normalized instantaneous frequency laws of each signal component.

Examples

```
N = 140; t = (0:N-1)';
[x1,if1] = fmlin(N,0.05,0.3);
[x2,if2] = fmsin(70,0.35,0.45,60);
x1(35+(1:70)) = x1(35+(1:70))+2*x2;
if2 = [zeros(35,1)*%nan ; if2 ; zeros(N-70-35,1)*%nan];
plotifl(t,[if1 if2],x1);
```



See also

- tfrideal — Ideal TFR for given instantaneous frequency laws
- plotsid — Schematic interference diagram of FM signals

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, August 94, August 95, May 1998.

tfrparam

parametric time-frequency representation

Calling Sequence

```
[tfr,t,f] = tfrparam(x)
[tfr,t,f] = tfrparam(x, t)
[tfr,t,f] = tfrparam(x, t, N)
[tfr,t,f] = tfrparam(x, t, N, p)
[tfr,t,f] = tfrparam(x, t, N, p, L)
[tfr,t,f] = tfrparam(x, t, N, p, L, ptrace)
[tfr,t,f] = tfrparam(x, t, N, p, L, ptrace, Rxtype)
[tfr,t,f] = tfrparam(x, t, N, p, L, ptrace, Rxtype,
method)
[tfr,t,f] = tfrparam(x, t, N, p, L, ptrace, Rxtype,
method, q)
[tfr,t,f] = tfrparam(...,"plot")
```

Parameters

X :

a vector of complex numbers of size Nx: the signal

T :

a vector with integer values in [1 Nx]: time instant(s) (default : 1:Nx).

N:

a positive integer: the number of frequency bins (default : max(256,Nx)).

p :

a positive integer: the last autocorrelation lag (default : 2).

L:

an odd positive integer: the length of the window around the analyzed time sample (default : max(51,round(Nx/10))).

Rxtype :

character string with possible values 'hermitian', 'fbhermitian', 'burg' or 'fbburg': the choice of the correlation matrix algorithm (default : 'fbhermitian')

Case does not matter.

method :

a character string: can be either 'ar', 'periodogram', 'capon', 'capnorm', 'lagunas', or 'genlag'.

Case does not matter.

q :

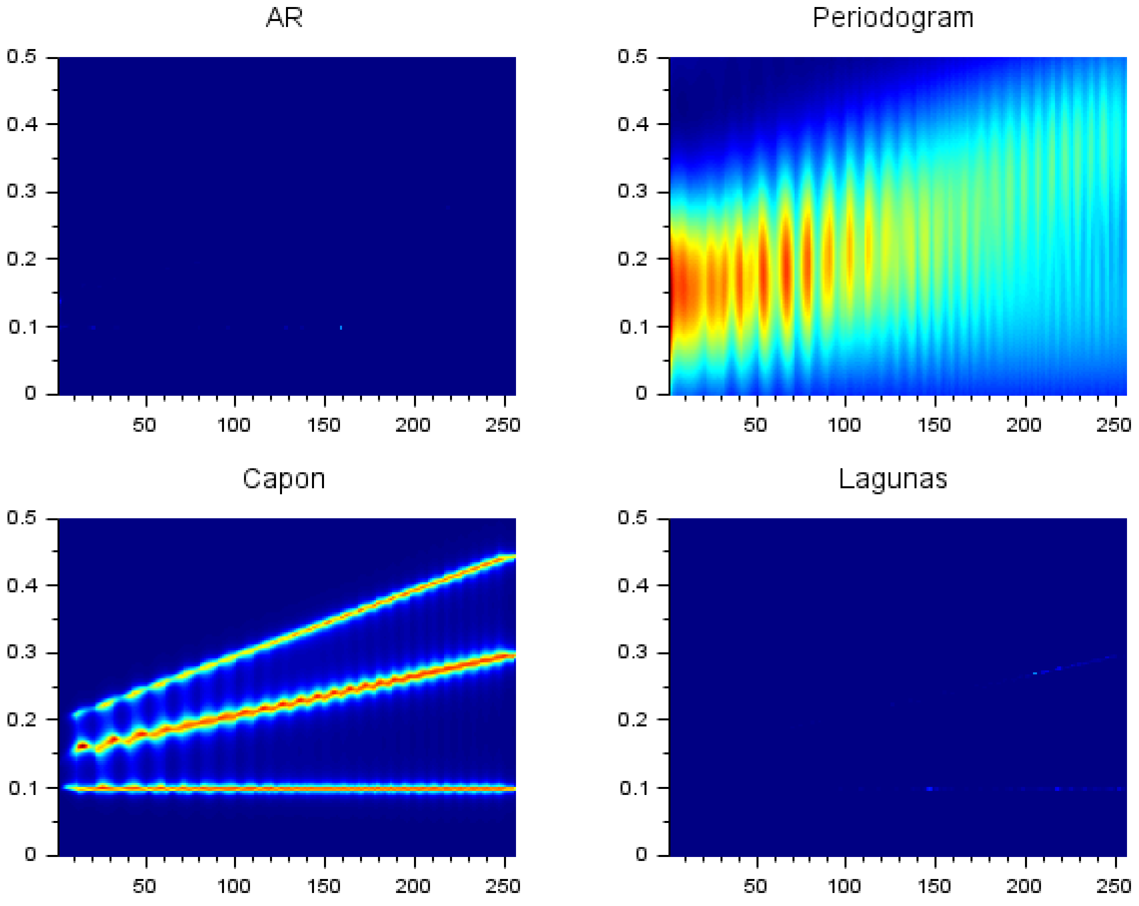
parameter for the generalized Lagunas method.

'plot':

if one input parameter is 'plot', tfrqview is called and the time-frequency representation will be plotted.

Examples

```
x = fmconst(256,0.1) + fmlin(256,0.15,0.30) +
fmlin(256,0.2,0.45);
clf; gcf().color_map = jetcolormap(128);
subplot(221)
[tfr,t,f] =
tfrparam(x,1:256,256,3,21,%f,'fbhermitian','ar');
grayplot(t,f,tfr'); title("AR")
subplot(222)
[tfr,t,f] =
tfrparam(x,1:256,256,3,21,%f,'fbhermitian','periodogram');
grayplot(t,f,tfr'); title("Periodogram")
subplot(223)
[tfr,t,f] =
tfrparam(x,1:256,256,3,21,%f,'fbhermitian','capon');
grayplot(t,f,tfr'); title("Capon")
subplot(224)
[tfr,t,f] =
tfrparam(x,1:256,256,3,21,%f,'fbhermitian','lagunas');
grayplot(t,f,tfr'); title("Lagunas")
```



Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, july, november 1998, april 99.

tfrqview

Quick visualization of time-frequency representations

Calling Sequence

```
tfrqview(TFR, SIG, T, METHOD, P1, P2, P3, P4, P5)
```

Parameters

TFR :

time-frequency representation (MxN).

SIG :

signal in time. If unavailable, put sig=[] as input parameter. (default : []).

T :

time instants (default : 1:N).

METHOD :

name of chosen representation (default : 'TYPE1').

tfr* :

See the help for authorized names.

TYPE1 :

the representation TFR goes in normalized frequency from -0.5 to 0.5 ;

TYPE2 :

the representation TFR goes in normalized frequency from 0 to 0.5.

P1...P5 :

optional parameters of the representation : run the file TFRPARAM(METHOD) to know the meaning of P1..P5 for your method.

Description

tfrqview allows a quick visualization of a time-frequency representation. When you use the 'save' option in the main menu, you save all your variables as well as two strings, TfrQView and TfrView, in a mat file. If you load this file and do eval(TfrQView), you will restart the display session under tfrqview ; if you do eval(TfrView), you will obtain the exact layout of the screen you had when clicking on the 'save' button.

Examples

```
sig = fmsin(128);
tfr = tfrwv(sig);
tfrqview(tfr,sig,1:128,'tfrwv');
```

See also

- tfrview — Visualization of time-frequency representations
- tfrsave — Save the parameters of a time-frequency representation
- tfrparam — parametric time-frequency representation

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, September 1994, July 1995
- O. Lemoine, Oct 1995, May-July 1996.
- F. Auger, May 1998.
- Copyright (c) 1996 by CNRS (France).

tfrsave

Save the parameters of a time-frequency representation

Calling Sequence

```
tfrsave(NAME, TFR, METHOD, SIG, T, F, P1, P2, P3, P4, P5)
```

Parameters

NAME :

name of the mat-file (less than 8 characters).

TFR :

time-frequency representation (MxN).

METHOD :

chosen representation.

SIG :

signal from which the TFR was obtained

T :

time instant(s) (default : (1:N)).

F :

frequency bins (default : linspace(0,0.5,M)).

P1..P5 :

optional parameters : run the file tfrparam(METHOD) to know the meaning of P1..P5 for your method.

Description

tfrsave saves the parameters of a time-frequency representation in the file NAME.dat. Two additional parameters are saved : TfrQView and TfrView. If you load the file 'name.dat' and do execstr(TfrQView), you will restart the display session under tfrqview ; if you do execstr(TfrView), you will display the representation by means of tfrview.

Examples

```
sig = fmlin(64);
tfr = tfrwv(sig);
tfrsave('wigner',tfr,'TFRWV',sig,1:64);
```

```
clear sig tfr
load 'wigner.dat'; execstr(TfrQView);
```

See also

- tfrqview — Quick visualization of time-frequency representations
- tfrview — Visualization of time-frequency representations
- tfrparam — parametric time-frequency representation

Authors

- H. Nahrstaedt - Aug 2010
- F. Auger, August 1994 - O. Lemoine, June 1996.
- Copyright (c) 1996 by CNRS (France).

tfrview

Visualization of time-frequency representations

Calling Sequence

```
tfrview(TFR, SIG, T, METHOD, PARAM)
tfrview(TFR, SIG, T, METHOD, PARAM, P1)
tfrview(TFR, SIG, T, METHOD, PARAM, P1, P2)
tfrview(TFR, SIG, T, METHOD, PARAM, P1, P2, P3)
tfrview(TFR, SIG, T, METHOD, PARAM, P1, P2, P3, P4)
tfrview(TFR, SIG, T, METHOD, PARAM, P1, P2, P3, P4, P5)
```

Parameters

TFR :

time-frequency representation.

SIG :

signal in the time-domain.

T :

time instants.

METHOD :

chosen representation (name of the corresponding sci-file)

PARAM :

visualization parameter vector : PARAM = [DISPLAY LINLOG THRESHOLD LEVNUMB NF2 LAYOUT FS ISGRID fmin fmax] where

DISPLAY :

1..5 for contour, imagesc, pcolor, surf or mesh

LINLOG :

0/1 for linearly/logarithmically spaced levels

THRESHOLD :

is the visualization threshold, in %

LEVELNUMB :

is the number of levels used with contour

NF2 :

is the number of frequency bins displayed

LAYOUT :

determines the layout of the figure : TFR alone (1), TFR and SIG (2), TFR and spectrum (3), TFR and SIG and spectrum (4), add 4 if you want a colorbar

FS :

is the sampling frequency (may be set to 1.0)

ISGRID :

depends on the grids' presence : isgrid=isgridsig+2*isgridspec+4*isgridtfr where isgridsig=1 if a grid is present on the signal and =0 if not, and so on

fmin :

smallest normalized frequency

fmax :

highest normalized frequency

P1..P5:

parameters of the representation. Run the file tfrparam(METHOD) to know the meaning of P1..P5.

Description

tfrview allows to visualize a time-frequency representation. tfrview is called through tfrqview from any tfr* function.

See also

- tfrqview — Quick visualization of time-frequency representations
- tfrparam — parametric time-frequency representation

Authors

- H. Nahrstaedt, Aug 2010
- F. Auger, July 1994, July 1995 -
- O. Lemoine, October-November 1995, May-June 1996.
- F. Auger, May 1998.
- Copyright (c) CNRS - France 1996.

stftb

  - tfrbert — Unitary Bertrand time-frequency distribution.
  - tfrdfla — D-Flandrin time-frequency distribution.
  - tfrscalo — Scalogram, for Morlet or Mexican hat wavelet.
  - tfrspaw — Smoothed Pseudo Affine Wigner time-frequency distributions.
  - tfrunter — Unterberger time-frequency distribution.
- Bilinear Time-Frequency Processing in the Cohen's Class
  - tfrbj — Born-Jordan time-frequency distribution.
  - tfrbud — Butterworth time-frequency distribution
  - tfrcw — Choi-Williams time-frequency distribution
  - tfrgrd — Generalized rectangular time-frequency distribution
  - tfrmh — Margenau-Hill time-frequency distribution
  - tfrmhs — Margenau-Hill-Spectrogram time-frequency distribution
  - tfrmmce — Minimum mean cross-entropy combination of spectrograms
  - tfrpage — Page time-frequency distribution
  - tfrpmh — Pseudo Margenau-Hill time-frequency distribution
  - tfrppage — Pseudo Page time-frequency distribution
  - tfrpwv — Pseudo Wigner-Ville time-frequency distribution
  - tfrri — Rihaczek time-frequency distribution.
  - tfrridb — Reduced Interference Distribution with Bessel kernel.
  - tfrridbn — Reduced Interference Distribution with a binomial kernel.
  - tfrridh — Reduced Interference Distribution with Hanning kernel.
  - tfrridt — Reduced Interference Distribution with triangular kernel.
  - tfrsp — Spectrogram time-frequency distribution.
  - tfrspbk — Smoothed Pseudo K-Bertrand time-frequency distribution.
  - tfrspwv — Smoothed Pseudo Wigner-Ville time-frequency distribution.
  - tfrwv — Wigner-Ville time-frequency distribution
  - tfrzam — Zao-Atlas-Marks time-frequency distribution
- Frequency-Domain Processing
  - ffmt — Fast Fourier Mellin Transform
  - frpowerspec — Computes the energy spectrum of the signal
  - frspec — Computes the spectrum of the signal
  - iffmt — Inverse fast Mellin transform
  - locfreq — Frequency localization caracteristics
  - parafrep — parametric frequency representation of a signal
  - sgrpdlay — Group delay estimation of a signal
  - tftb_fft — matlab compatible fft
  - tftb_ifft — matlab compatible ifft
- Choice of the Instantaneous Frequency
  - fmconst — Signal with constant frequency modulation
  - fmhyp — Signal with hyperbolic frequency modulation
  - fmlin — Signal with linear frequency modulation
  - fmodany — Signal with arbitrary frequency modulation
  - fmpar — Parabolic frequency modulated signal
  - fmpower — Signal with power-law frequency modulation
  - fmsin — Signal with sinusoidal frequency modulation
  - gdpower — Signal with power-law group delay
  - if2phase — Generate the phase from the instantaneous frequency
- Linear Time-Frequency Processing
  - tffilter — Time frequency filtering of a signal
  - tfrgabor — Gabor representation of a signal
  - tfristft — Inverse Short time Fourier transform
  - tfrstft — Short time Fourier transform

    - tfrsurf — extract from a time-frequency representation the biggest energy dots
- Modification
    - scale — Scale a signal using the Mellin transform
    - sigmerge — Add two signals with given energy ratio in dB
- Noise Realizations
    - noisecg — Analytic complex gaussian noise
    - noisecu — Analytic complex uniform white noise
- Other
    - contwtgn — computes a continuous wavelet transform
    - contwtgnmir — Continuous wavelet transform of mirrored 1-D signals
    - correlmx — correlation matrix of a signal
    - d2statio — Distance to stationarity
    - disprog — Display progression of a loop
    - divider — Find dividers of an integer
    - dwindow — Derive a window
    - fzero — solves the scalar nonlinear equation such that F(X) == 0
    - gaussn — generates the order n derivative of the gaussian window, centered at frequency f0
    - imextrac — imextrac(Image) extract and isolate dots in a binary image
    - integ — Approximate 1D integral of a discrete signal
    - integ2d — Approximate 2-D integral
    - istfr1 — returns true is method is a time frequency representation of type 1 (frequencies >0 or <0)
    - istfr2 — returns true is method is a time frequency representation of type 2 (only frequencies > 0)
    - izak — Inverse Zak transform
    - kaytth — Kay-Tretter filter computation
    - odd — Round towards nearest odd value
    - rem — Return the remainder of the division x/y
    - rot90 — rotates the given 2-D array by 90 degrees
    - umaxbert — Determination of the maximum value of u for Bertrand distribution
    - umaxdfla — Determination of the maximum value of u for D-Flandrin distribution
    - umaxunte — Determination of the maximum value of u for Unterberger distribution
    - vecmodulo — Congruence of a vector
    - zak — Zak transform
- Post-Processing or Help to the Interpretation
    - friedman — FRIEDMAN Instantaneous frequency density
    - holder — Estimate the Holder exponent through an affine TFR
    - htl — Hough transform for detection of lines in images
    - margtfr — Marginals and energy of a time-frequency representation
    - midpoint — Mid-point construction used in the interference diagram
    - momftfr — Frequency moments of a time-frequency representation
    - momttfr — Time moments of a time-frequency representation
    - plotsid — Schematic interference diagram of FM signals
    - renyi — Measure Renyi information
    - ridges — Extraction of ridges
    - tfrideal — Ideal TFR for given instantaneous frequency laws
- Reassigned Time-Frequency Processing
    - tfrrgab — Reassigned Gabor spectrogram time-frequency distribution
    - tfrrmsc — Reassigned Morlet Scalogram time-frequency distribution
    - tfrrpmh — Reassigned pseudo Margenau-Hill time-frequency distribution

    - tfrrppag — Reassigned pseudo Page time-frequency distribution
    - tfrrpwv — Reassigned pseudo Wigner-Ville distribution
    - tfrrsp — Reassigned Spectrogram
    - tfrrspwv — Reassigned smoothed pseudo Wigner-Ville distribution
    - tfrrstan — Reassigned Stankovic distribution
- Choice of Particular Signals
    - altes — Altes signal in time domain
    - anaask — Amplitude Shift Keying (ASK) signal
    - anabpsk — Binary Phase Shift Keying (BPSK) signal
    - anafsk — Frequency Shift Keying (FSK) signal
    - anapulse — Analytic projection of unit amplitude impulse signal
    - anaqpsk — Quaternary Phase Shift Keying (QPSK) signal
    - anasing — Lipschitz singularity
    - anastep — Analytic projection of unit step signal
    - atoms — Linear combination of elementary Gaussian atoms
    - chirp — Evaluate a chirp signal at time t. A chirp signal is a frequency swept cosine wave
    - dopnoise — Complex noisy doppler signal
    - doppler — Generate complex Doppler signal
    - griffitc — Test signal example C of Griffiths' paper
    - klauder — Klauder wavelet in time domain
    - mexhat — Mexican hat wavelet in time domain
    - pulstran — pulse generation
    - sawtooth — Generates a sawtooth wave
    - tftb_window — Window generation
- Time-Domain Processing
    - ifestar2 — Instantaneous frequency estimation using AR2 modelisation
    - instfreq — Instantaneous frequency estimation
    - loctime — Time localization caracteristics
- Visualization and backup
    - plotifl — Plot normalized instantaneous frequency laws
    - tfrparam — parametric time-frequency representation
    - tfrqview — Quick visualization of time-frequency representations
    - tfrsave — Save the parameters of a time-frequency representation
    - tfrview — Visualization of time-frequency representations

<< stftb

Ambiguity plane function (in C) >>

Overview of STFTb features

Scilab Time Frequency Toolbox

Signal generation functions:

Choice of the Instantaneous Amplitude :

amexpo1s One-sided exponential amplitude modulation

amexpo2s Bilateral exponential amplitude modulation

amgauss Gaussian amplitude modulation

amrect Rectangular amplitude modulation

amtriang Triangular amplitude modulation

Choice of the Instantaneous Frequency :

fmconst Signal with constant frequency modulation

fmhyp Signal with hyperbolic frequency modulation

fmlin Signal with linear frequency modulation

fmodany Signal with arbitrary frequency modulation

fmpar Signal with parabolic frequency modulation

fmpower Signal with power-law frequency modulation

fmsin Signal with sinusoidal frequency modulation

gdpower Signal with a power-law group delay

if2phase Generate the phase from the instantaneous frequency

Choice of Particular Signals

altes Altes signal in time domain

anaask Amplitude Shift Keyed (ASK) signal

anabpsk Binary Phase Shift Keyed (BPSK) signal

anafsk Frequency Shift Keyed (FSK) signal

anapulse Analytic projection of unit amplitude impulse signal

anaqpsk Quaternary Phase Shift Keyed (QPSK) signal

anasing Lipschitz singularity

anastep Analytic projection of unit step signal

atoms Linear combination of elementary Gaussian atoms chirp Evaluate a chirp signal at time t. A chirp signal is a frequency swept cosine wave.

dopnoise Complex Doppler random signal

doppler Complex Doppler signal

klauder Klauder wavelet in time domain

mexhat Mexican hat wavelet in time domain

pulstran Puls generation

sawtooth Generates a sawtooth wave

tftb_window Window generation

griffitc Test signal example C of Griffiths' paper.

Noise Realizations

noisecg Analytic complex gaussian noise

noisecu Analytic complex uniform white noise

Modification

scale Scale a signal using the Mellin transform

sigmerge Add two signals with a given energy ratio in dB

Processing functions

Time-Domain Processing

ifestar2 Instantaneous frequency estimation using AR2 modelisation.

instfreq Instantaneous frequency estimation

loctime Time localization characteristics

Frequency-Domain Processing

ffmt Fast Mellin transform

iffmt Inverse fast Mellin transform

locfreq Frequency localization characteristics

sgrpdlay Group delay estimation

tftb_fft Fast Fourier transform

tftb_ifft Inverse fast Fourier transform

frspec Spectrum

frpowerspec Power spectrum

parafrep parametric frequency representation of a signal.

Linear Time-Frequency Processing

tfrgabor Gabor representation

tfrstft Short time Fourier transform

tfrsurf extract from a time-frequency representation the biggest energy dots

tfristft Inverse Short time Fourier transform.

tffilter Time frequency filtering of a signal.

Bilinear Time-Frequency Processing in the Cohen's Class

tfrbj Born-Jordan distribution

tfrbud Butterworth distribution

tfrcw Choi-Williams distribution

tfrgrd Generalized rectangular distribution

tfrmh Margenau-Hill distribution

tfrmhs Margenau-Hill-Spectrogram distribution

tfrmmce Minimum mean cross-entropy combination of spectrograms

tfrpage Page distribution

tfrpmh Pseudo Margenau-Hill distribution

tfrppage Pseudo Page distribution

tfrpwv Pseudo Wigner-Ville distribution

tfrri Rihaczek distribution

tfrridb Reduced interference distribution (Bessel window)

tfrridbn Reduced interference distribution (binomial window)

tfrridh Reduced interference distribution (Hanning window)

tfrridt Reduced interference distribution (triangular window)

tfrsp Spectrogram distribution

tfrspwv Smoothed Pseudo Wigner-Ville distribution

tfrwv Wigner-Ville distribution

tfrzam Zhao-Atlas-Marks distribution

tfrspbk Smoothed Pseudo K-Bertrand time-frequency distribution.

Bilinear Time-Frequency Processing in the Affine Class

tfrbert Unitary Bertrand distribution

tfrdfla D-Flandrin distribution

tfrscalo Scalogram, for Morlet or Mexican hat wavelet

tfrspaw Smoothed Pseudo Affine Wigner distributions

tfrunter Unterberger distribution, active or passive form

istfraff returns true is method is an affine time frequency representation.

lambdak Evaluate lambda function for Affine Wigner distribution.

Reassigned Time-Frequency Processing

tfrrgab Reassigned Gabor spectrogram

tfrrmsc Reassigned Morlet Scalogram time-frequency distribution

tfrrpmh Reassigned Pseudo Margenau-Hill distribution

tfrrppag Reassigned Pseudo Page distribution

tfrrpwv Reassigned Pseudo Wigner-Ville distribution

tfrrsp Reassigned Spectrogram

tfrrspwv Reassigned Smoothed Pseudo WV distribution

tfrrstan Reassigned Stankovic distribution.

Ambiguity Functions

ambifunb Narrow-band ambiguity function

ambifuwb Wide-band ambiguity function

Post-Processing or Help to the Interpretation

friedman Instantaneous frequency density

holder Estimation of the Holder exponent through an affine TFR

htl Hough transform for detection of lines in images

margtfr Marginals and energy of a time-frequency representation

midpoint Mid-point construction used in the interference diagram

momftfr Frequency moments of a time-frequency representation

momttfr Time moments of a time-frequency representation

plotsid Schematic interference diagram of FM signals

renyi Measure Renyi information

ridges Extraction of ridges from a reassigned TFR

tfrideal Ideal TFR for given frequency laws

Visualization and backup

plotifl Plot normalized instantaneous frequency laws

tfrparam Return the paramaters needed to display (or save) a TFR

tfrqview Quick visualization of a time-frequency representation

tfrsave Save the parameters of a time-frequency representation

tfrview Visualization of time-frequency representations

Other

disprog Display the progression of a loop

divider Find dividers of an integer, closest from the square root of the integer

dwindow Derive a window

integ Approximate an integral

integ2d Approximate a 2-D integral

izak Inverse Zak transform

kaytth Computation of the Kay-Tretter filter

vecmodulo Congruence of a vector

odd Round towards nearest odd value

zak Zak transform

rem Return the remainder of the division x/y

rot90 rotates the given matrix clockwise by 90 degrees

fzero solves the scalar nonlinear equation such that F(X) == 0

d2statio Distance to stationarity

correlmx correlation matrix of a signal

gaussn generates the order n derivative of the gaussian window, centered at frequency f0

imextrac imextrac(Image) extract and isolate dots in a binary image

umaxbert Determination of the maximum value of u for Bertrand distribution.

umaxdfla Determination of the maximum value of u for D-Flandrin distribution.

umaxunte Determination of the maximum value of u for Unterberger distribution.

istfr1 returns true is method is a time frequency representation of type 1 (positive and negative frequencies).

istfr2 returns true is method is a time frequency representation of type 2 (only positive frequencies).

contwtgn computes a continuous wavelet transform.

contwtgnmir Continuous wavelet transform of mirrored 1-D signals

C - Toolbox functions

Time-Frequency representations

Ctfrbj Born-Jordan time-frequency distribution.

Ctfrbud Butterworth time-frequency distribution.

Ctfrcw Choi-Williams time-frequency distribution.

Ctfrgrd Generalized rectangular time-frequency distribution.

Ctfrmh Margenau-Hill time-frequency distribution.

Ctfrmhs Margenau-Hill-Spectrogram time-frequency distribution.

Ctfrmmce Minimum mean cross-entropy combination of spectrograms.

Ctfrpage Page time-frequency distribution.

Ctfrpmh Pseudo Margenau-Hill time-frequency distribution.

Ctfrppage Pseudo Page time-frequency distribution.

Ctfrpwv Pseudo Wigner-Ville time-frequency distribution.

Ctfrri Rihaczek time-frequency distribution.

Ctfrridb Reduced Interference Distribution with Bessel kernel.

Ctfrridbn Reduced Interference Distribution with binomial kernel.

Ctfrridh Reduced Interference Distribution with Hanning kernel.

Ctfrridt Reduced Interference Distribution with Triangular kernel.

Ctfrrsp Reassigned Spectrogram

Ctfrsp Spectrogram Time Frequency distribution

Ctfrspwv Smoothed Pseudo Wigner-Ville time-frequency distribution.

Ctfrstft Short time Fourier transform

Ctfrwv Wigner-Ville time-frequency distribution.

Ctfrzam Zao-Atlas-Marks time-frequency distribution.

Ambiguity plane function

Cambifunb Ambiguity function

Ctfrker Time frequency representation kernel.

Caf2tfr From ambiguity plane to time frequency plane

Miscellanaeous functions

Chtl Hough transform for detection of lines in images..

Ctfrdist Time frequency distance measures

Ctfrreas Time Frequency Representation reassignment

Cwindow Window generation.

Authors

- F. Auger, oct 1999.
- M. Davy, 2000.
- E. Roy, 2000.
- Holger Nahrstaedt - Aug 2010