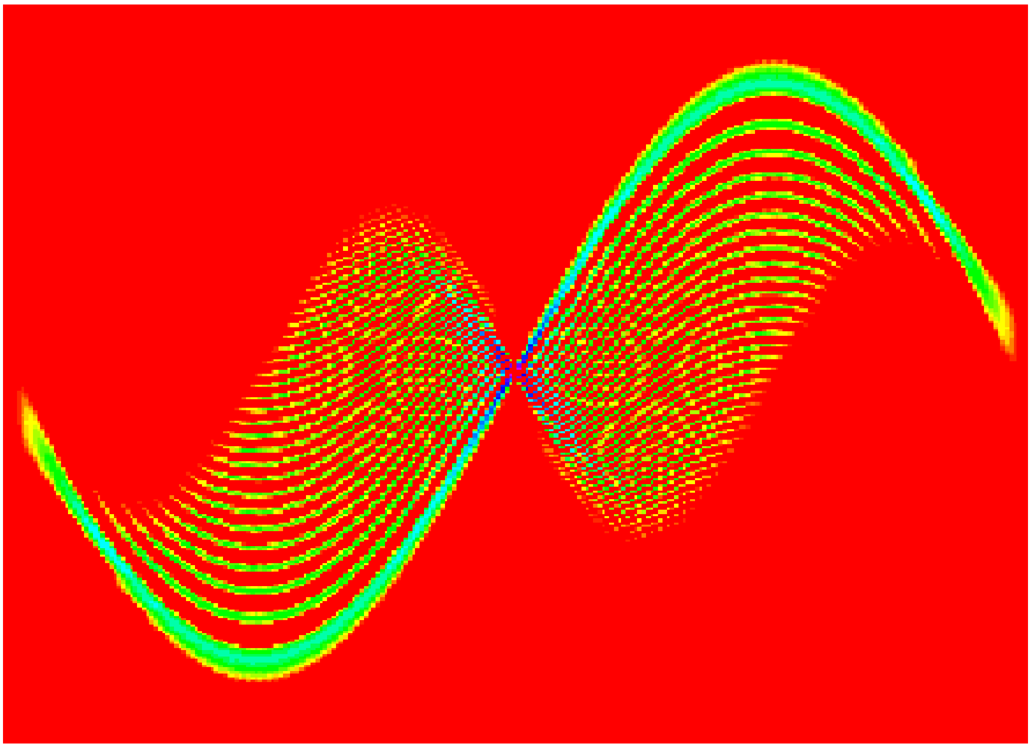
Time-Frequency Toolbox

For Use with MATLAB



Reference Guide

*François Auger **

*Patrick Flandrin **

Paulo Gonçalvès °

*Olivier Lemoine **

** CNRS (France)*

° Rice University (USA)

1995-1996

Copyright (C) 1996 CNRS (France) and Rice University (USA).
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled ”GNU Free Documentation License”.

The Time-Frequency Toolbox has been mainly developed under the auspices of the French CNRS (*Centre National de la Recherche Scientifique*). It results from a research effort conducted within its Groupements de Recherche ”Traitement du Signal et Images” (O. Macchi) and ”Information, Signal et Images” (J.-M. Chassery). Parts of the Toolbox have also been developed at Rice University, when one of the authors (PG) was visiting the Department of Electrical and Computer Engineering, supported by NSF. Supporting institutions are gratefully acknowledged, as well as M. Guglielmi, M. Najim, R. Settineri, R.G. Baraniuk, M. Chausse, D. Roche, E. Chassande-Mottin, O. Michel and P. Abry for their help at different phases of the development.

Glossary and summary

This section contains detailed descriptions of all the Time-Frequency Toolbox functions. It begins with a glossary and a list of functions grouped by subject area and continues with the reference entries in alphabetical order. Information is also available through the online help facility.

AF	Ambiguity function
AR	Auto-regressive (filter or model)
ASK	Amplitude shift keyed signal
BJD	Born-Jordan distribution
BPSK	Binary phase shift keyed signal
BUD	Butterworth distribution
CWD	Choi-Williams distribution
FM	Frequency modulation
FSK	Frequency shift keyed signal
GRD	Generalized rectangular distribution
HT	Hough transform
MHD	Margenau-Hill distribution
MHSD	Margenau-Hill-Spectrogram distribution
MMCE	Minimum mean cross-entropy
NAF	Narrow-band ambiguity function
PMHD	Pseudo Margenau-Hill distribution
PWVD	Pseudo Wigner-Ville distribution
QPSK	Quaternary phase shift keyed signal
RID	Reduced interference distribution
STFT	Short-time Fourier transform
TFR	Time-frequency representation
WAF	Wide-band ambiguity function
WVD	Wigner-Ville distribution
ZAM	Zhao-Atlas-Marks distribution

- **Signal generation files**

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

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
`dopnoise`	Complex Doppler random signal
`doppler`	Complex Doppler signal
`klauder`	Klauder wavelet in time domain
`mexhat`	Mexican hat wavelet in time domain
`window`	Window generation

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 files**

Time-Domain Processing	
ifestar2	Instantaneous frequency estimation using AR2 modelisation.
instfreq	Instantaneous frequency estimation
loctime	Time localization characteristics

Frequency-Domain Processing	
fmt	Fast Mellin transform
ifmt	Inverse fast Mellin transform
locfreq	Frequency localization characteristics
sgrpdlay	Group delay estimation

Linear Time-Frequency Processing	
tfrgabor	Gabor representation
tfrstft	Short time Fourier transform

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

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

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

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 Hlder 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
modulo	Congruence of a vector
movcw4at	Four atoms rotating, analyzed by the Choi-Williams distribution
movpwdph	Influence of a phase-shift on the interferences of the PWVD
movpwjph	Influence of a jump of phase on the interferences of the PWVD
movsc2wv	Movie illustrating the passage from the scalogram to the WVD
movsp2wv	Movie illustrating the passage from the spectrogram to the WVD
movwv2at	Oscillating structure of the interferences of the WVD
odd	Round towards nearest odd value
zak	Zak transform

Contents

altes

Purpose

Altes signal in time domain.

Synopsis

```
x = altes(N)
x = altes(N,fmin)
x = altes(N,fmin,fmax)
x = altes(N,fmin,fmax,alpha)
```

Description

`altes` generates the Altes signal in the time domain.

Name	Description	Default value
`N`	number of points in time	
`fmin`	lower frequency bound (value of the hyperbolic instantaneous frequency law at the sample `N`), in normalized frequency	`0.05`
`fmax`	upper frequency bound (value of the hyperbolic instantaneous frequency law at the first sample), in normalized frequency	`0.5`
`alpha`	attenuation factor of the envelope	`300`
`x`	time row vector containing the Altes signal samples	

Example

```
x=altes(128,0.1,0.45); plot(x);
```

plots an Altes signal of 128 points whose normalized frequency goes from 0.45 down to 0.1.

See Also

`klauder, anasing, anapulse, anastep, doppler.`

ambifunb

Purpose

Narrow-band ambiguity function.

Synopsis

```
[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)
```

Description

`ambifunb` computes the narrow-band ambiguity function of a signal, or the cross-ambiguity function between two signals. Its definition is given by

$$A_x(\xi,\tau) = \int_{-\infty}^{+\infty} x(s+\tau/2)\, x^*(s-\tau/2)\, e^{-j2\pi\xi s}\, ds.$$

Name	Description	Default value
`x`	signal if auto-AF, or `[x1,x2]` if cross-AF (`length(x)=Nx`)	
`tau`	vector of lag values	`(-Nx/2:Nx/2)`
`N`	number of frequency bins	`Nx`
`trace`	if non-zero, the progression of the algorithm is shown	`0`
`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	

This representation is computed such as its 2D Fourier transform equals the Wigner-Ville distribution. When called without output arguments, `ambifunb` displays the squared modulus of the ambiguity function by means of `contour`.
The ambiguity function is a measure of the time-frequency correlation of a signal x, i.e. the degree of similarity between x and its translated versions in the time-frequency plane.

Examples

Consider a BPSK signal (see `anabpsk`) of 256 points, with a keying period of 8 points, and analyze it with the narrow-band ambiguity function :

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

The resulting function presents a high thin peak at the origin of the ambiguity plane, with small sidelobes around. This means that the inter-correlation between this signal and a time/frequency-shifted version of it is nearly zero (the ambiguity in the estimation of its arrival time and mean-frequency is very small).

Here is an other example that checks the correspondance between the WVD and the narrow-band ambiguity function by means of a 2D Fourier transform :

```
N=128; sig=fmlin(N); amb=ambifunb(sig);
amb=amb([N/2+1:N 1:N/2],:);
ambi=ifft(amb).';
tdr=zeros(N);  % Time-delay representation
tdr(1:N/2,:)=ambi(N/2:N-1,:);
tdr(N:-1:N/2+2,:)=ambi(N/2-1:-1:1,:);
wvd1=real(fft(tdr));

wvd2=tfrwv(sig);
diff=max(max(abs(wvd1-wvd2)))
diff =
       1.5632e-13
```

See Also

`ambifuwb`.

ambifuwb

Purpose

Wide-band ambiguity function.

Synopsis

```
[waf,tau,theta] = ambifuwb(x)
[waf,tau,theta] = ambifuwb(x,fmin,fmax)
[waf,tau,theta] = ambifuwb(x,fmin,fmax,N)
[waf,tau,theta] = ambifuwb(x,fmin,fmax,N,trace)
```

Description

`ambifuwb` calculates the asymetric wide-band ambiguity function, defined as

$$\Xi_x(a,\tau) = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} x(t)\, x^*(t/a-\tau)\, dt = \sqrt{a} \int_{-\infty}^{+\infty} X(\nu)\, X^*(a\nu)\, e^{j2\pi a\tau\nu}\, d\nu.$$

Name	Description	Default value
`x`	signal (in time) to be analyzed (the analytic associated signal is considered), of length `Nx`	
`fmin`, `fmax`	respectively lower and upper frequency bounds of the analyzed signal. When specified, these parameters fix the equivalent frequency bandwidth (both are expressed in Hz)	`0, 0.5`
`N`	number of Mellin points. This number is needed when `fmin` and `fmax` are forced	`Nx`
`trace`	if non-zero, the progression of the algorithm is shown	0
`waf`	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.	
`tau`	X-coordinate corresponding to time delay	
`theta`	Y-coordinate corresponding to the $\log(a)$ variable, where a is the scale	

When called without output arguments, `ambifuwb` displays the squared modulus of the ambiguity function by means of `contour`.

Example

Consider a BPSK signal (see `anabpsk`) of 256 points, with a keying period of 8 points, and analyze it with the wide-band ambiguity function :

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

The result, to be compared with the one obtained with the narrow-band ambiguity function, presents a thin high peak at the origin of the ambiguity plane, but with more important sidelobes than with the narrow-band ambiguity function. It means that the narrow-band assumption is not very well adapted to this signal, and that the ambiguity in the estimation of its arrival time and mean frequency is not so small.

See Also

`ambifunb`.

amexpo1s

Purpose

One-sided exponential amplitude modulation.

Synopsis

```
y = amexpo1s(N)
y = amexpo1s(N,t0)
y = amexpo1s(N,t0,T)
```

Description

`amexpo1s` generates a one-sided exponential amplitude modulation starting at time `t0`, and with a spread proportional to `T`.
This modulation is scaled such that `y(t0)=1`.

Name	Description	Default value
`N`	number of points	
`t0`	arrival time of the exponential	`N/2`
`T`	time spreading	`2*sqrt(N)`
`y`	signal	

Examples

```
z=amexpo1s(160);       plot(z);
z=amexpo1s(160,20,40); plot(z);
```

See Also

`amexpo2s, amgauss, amrect, amtriang.`

amexpo2s

Purpose

Bilateral exponential amplitude modulation.

Synopsis

```
y = amexpo2s(N)
y = amexpo2s(N,t0)
y = amexpo2s(N,t0,T)
```

Description

`amexpo2s` generates a bilateral exponential amplitude modulation centered on a time `t0`, and with a spread proportional to `T`.
This modulation is scaled such that `y(t0)=1`.

Name	Description	Default value
`N`	number of points	
`t0`	time center	`N/2`
`T`	time spreading	`2*sqrt(N)`
`y`	signal	

Examples

```
z=amexpo2s(160);         plot(z);
z=amexpo2s(160,90,40);   plot(z);
z=amexpo2s(160,180,50);  plot(z);
```

See Also

`amexpo1s`, `amgauss`, `amrect`, `amtriang`.

amgauss

Purpose

Gaussian amplitude modulation.

Synopsis

```
y = amgauss(N)
y = amgauss(N,t0)
y = amgauss(N,t0,T)
```

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 :

$$y(t) = e^{-\pi\left(\frac{t-t_0}{T}\right)^2}$$

Name	Description	Default value
`N`	number of points	
`t0`	time center	`N/2`
`T`	time spreading	`2*sqrt(N)`
`y`	signal	

Examples

```
z=amgauss(160);        plot(z);
z=amgauss(160,90,40);  plot(z);
z=amgauss(160,180,50); plot(z);
```

See Also

`amexpo1s, amexpo2s, amrect, amtriang.`

amrect

Purpose

Rectangular amplitude modulation.

Synopsis

```
y = amrect(N)
y = amrect(N,t0)
y = amrect(N,t0,T)
```

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`.

Name	Description	Default value
`N`	number of points	
`t0`	time center	`N/2`
`T`	time spreading	`2*sqrt(N)`
`y`	signal	

Examples

```
z=amrect(160);        plot(z);
z=amrect(160,90,40);  plot(z);
z=amrect(160,180,70); plot(z);
```

See Also

`amexpo1s, amexpo2s, amgauss, amtriang.`

amtriang

Purpose

Triangular amplitude modulation.

Synopsis

```
y = amtriang(N)
y = amtriang(N,t0)
y = amtriang(N,t0,T)
```

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`.

Name	Description	Default value
`N`	number of points	
`t0`	time center	`N/2`
`T`	time spreading	`2*sqrt(N)`
`y`	signal	

Examples

```
z=amtriang(160);        plot(z);
z=amtriang(160,90,40);  plot(z);
z=amtriang(160,180,50); plot(z);
```

See Also

`amexpo1s, amexpo2s, amgauss, amrect.`

anaask

Purpose

Amplitude Shift Keyed (ASK) signal.

Synopsis

```
[y,am] = anaask(N)
[y,am] = anaask(N,ncomp)
[y,am] = anaask(N,ncomp,f0)
```

Description

`anaask` returns a complex amplitude modulated signal of normalized frequency `f0`, with a uniformly distributed random amplitude. Such signal is only 'quasi'-analytic.

Name	Description	Default value
`N`	number of points	
`ncomp`	number of points of each component	`N/5`
`f0`	normalized frequency	`0.25`
`y`	signal	
`am`	resulting amplitude modulation	

Example

```
[signal,am]=anaask(512,64,0.05);
subplot(211); plot(real(signal));
subplot(212); plot(am);
```

See Also

`anafsk, anabpsk, anaqpsk.`

Reference

[1] W. Gardner *Statistical Spectral Analysis - A Nonprobabilistic Theory* Englewood Cliffs, N.J. Prentice Hall, 1987.

anabpsk

Purpose

Binary Phase Shift Keyed (BPSK) signal.

Synopsis

```
[y,am] = anabpsk(N)
[y,am] = anabpsk(N,ncomp)
[y,am] = anabpsk(N,ncomp,f0)
```

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.

Name	Description	Default value
`N`	number of points	
`ncomp`	number of points of each component	`N/5`
`f0`	normalized frequency	`0.25`
`y`	signal	
`am`	resulting amplitude modulation	

Example

```
[signal,am]=anabpsk(300,30,0.1);
subplot(211); plot(real(signal));
subplot(212); plot(am);
```

See Also

`anafsk, anaqpsk, anaask.`

Reference

[1] W. Gardner *Introduction to Random Processes, with Applications to Signals and Systems*, 2nd Edition, McGraw-Hill, New-York, p. 360 ,1990.

anafsk

Purpose

Frequency Shift Keyed (FSK) signal.

Synopsis

```
[y,iflaw] = anafsk(N)
[y,iflaw] = anafsk(N,ncomp)
[y,iflaw] = anafsk(N,ncomp,nbf)
```

Description

`anafsk` simulates a phase coherent Frequency Shift Keyed (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.

Name	Description	Default value
`N`	number of points	
`ncomp`	number of points of each component	`N/5`
`nbf`	number of distinct frequencies	`4`
`y`	signal	
`iflaw`	instantaneous frequency law	

Example

```
[signal,ifl]=anafsk(512,64,5);
subplot(211); plot(real(signal));
subplot(212); plot(ifl);
```

See Also

`anabpsk`, `anaqpsk`, `anaask`.

Reference

[1] W. Gardner *Introduction to Random Processes, with Applications to Signals and Systems*, 2nd Edition, McGraw-Hill, New-York, p. 357 ,1990.

anapulse

Purpose

Analytic projection of unit amplitude impulse signal.

Synopsis

```
y = anapulse(N)
y = anapulse(N,ti)
```

Description

`anapulse` returns an analytic N-dimensional signal whose real part is a Dirac impulse at `t=ti`.

Name	Description	Default value
`N`	number of points	
`ti`	time position of the impulse	`round(N/2)`
`y`	output signal	

Example

```
signal=2.5*anapulse(512,301);
plot(real(signal));
```

See Also

`anastep`, `anasing`, `anabpsk`, `anafsk`.

anaqpsk

Purpose

Quaternary Phase Shift Keyed (QPSK) signal.

Synopsis

```
[y,pm0] = anaqpsk(N)
[y,pm0] = anaqpsk(N,ncomp)
[y,pm0] = anaqpsk(N,ncomp,f0)
```

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.

Name	Description	Default value
`N`	number of points	
`ncomp`	number of points of each component	`N/5`
`f0`	normalized frequency	`0.25`
`y`	signal	
`pm0`	initial phase of each component	

Example

```
[signal,pm0]=anaqpsk(512,64,0.05);
subplot(211); plot(real(signal));
subplot(212); plot(pm0);
```

See Also

`anafsk, anabpsk, anaask.`

Reference

[1] W. Gardner *Introduction to Random Processes, with Applications to Signals and Systems*, 2nd Edition, McGraw-Hill, New-York, p. 362 ,1990.

anasing

Purpose

Lipschitz singularity.

Synopsis

```
x = anasing(N)
x = anasing(N,t0)
x = anasing(N,t0,H)
```

Description

`anasing` generates the N-points Lipschitz singularity centered around `t0` : $x(t) = |t - t0|^H$.

Name	Description	Default value
`N`	number of points in time	
`t0`	time localization of the singularity	`N/2`
`H`	strength of the Lipschitz singularity (positive or negative)	`0`
`x`	the time row vector containing the signal samples	

Example

```
x=anasing(128); plot(real(x));
```

See Also

`anastep, anapulse, anabpsk, doppler, holder.`

Reference

[1] S. Mallat and W.L. Hwang "Singularity Detection and Processing with Wavelets" IEEE Trans. on Information Theory, Vol 38, No 2, March 1992, pp. 617-643.

anastep

Purpose

Analytic projection of unit step signal.

Synopsis

```
y = anastep(N)
y = anastep(N,ti)
```

Description

`anastep` generates the analytic projection of a unit step signal : $y(t) = 0$ for $t < t_i$, and $y(t) = 1$ for $t \geq t_i$.

Name	Description	Default value
`N`	number of points	
`ti`	starting position of the unit step	`N/2`
`y`	output signal	

Examples

```
signal=anastep(256,128);      plot(real(signal));
signal=-2.5*anastep(512,301); plot(real(signal));
```

See Also

`anasing`, `anafsk`, `anabpsk`, `anaqpsk`, `anaask`.

atoms

Purpose

Linear combination of elementary Gaussian atoms.

Synopsis

```
[sig,locatoms] = atoms(N)
[sig,locatoms] = atoms(N,coord)
```

Description

`atoms` generates a signal consisting in a linear combination of elementary gaussian atoms. 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`), with the help of a menu.

Name	Description	Default value
`N`	number of points of the signal	
`coord`	matrix of time-frequency centers, of the form `[t1,f1,T1,A1;...;tM,fM,TM,AM]`. `(ti,fi)` are the time-frequency coordinates of atom `i`, `Ti` is its time duration and `Ai` its amplitude. Frequencies `f1..fM` should be between 0 and 0.5. If `nargin==1`, the location of the atoms will be defined by clicking with the mouse	`Ti=N/4, Ai=1`.
`sig`	output signal	
`locatoms`	matrix of time-frequency coordinates and durations of the atoms	

When the selection of the atoms is finished (after clicking on the 'Stop' buttom, or after having specified the coordinates at the command line with the input parameter `coord`), the signal in time together with a schematic representation of the atoms in the time-frequency plane are displayed on the current figure.

Examples

```
sig=atoms(128);
sig=atoms(128,[32,0.3,32,1;56,0.15,48,1.22;102,0.41,20,0.7]);
```

See Also

`amgauss, fmconst`.

disprog

Purpose

Display progression of a loop.

Synopsis

```
disprog(k,N,steps)
```

Description

`disprog` displays the progression of a loop. This function is intended to see the progression of slow algorithms.

Name	Description	Default value
`k`	loop variable	
`N`	final value of `k`	
`steps`	number of displayed steps	

Example

```
N=16; for k=1:N, disprog(k,N,5); end;

 20 40 60 80 100 % complete in 0.0333333 seconds.
```

divider

Purpose

Find dividers of an integer, closest from the square root of the integer.

Synopsis

```
[N,M] = divider(N1)
```

Description

`divider` find two integers `N` and `M` such that `M*N=N1`, with `M` and `N` as close as possible from `sqrt(N1)`.

Examples

```
N1=256; [N,M]=divider(N1); [N,M]
ans =
      16    16
N1=258; [N,M]=divider(N1); [N,M]
ans =
      6     43
```

dopnoise

Purpose

Complex doppler random signal.

Synopsis

```
[y,iflaw] = dopnoise(N,fs,f0,d,v)
[y,iflaw] = dopnoise(N,fs,f0,d,v,t0)
[y,iflaw] = dopnoise(N,fs,f0,d,v,t0,c)
```

Description

`dopnoise` generates a complex noisy doppler signal, normalized so as to be of unit energy.

Name	Description	Default value
`N`	number of points	
`fs`	sampling frequency (in Hz)	
`f0`	target frequency (in Hz)	
`d`	distance from the line to the observer (in meters)	
`v`	target velocity (in m/s)	
`t0`	time center	`N/2`
`c`	wave velocity (in m/s)	`340`
`y`	output signal	
`iflaw`	model used as instantaneous frequency law	

`[y,iflaw] = dopnoise(N,fs,f0,d,v,t0,c)` returns the signal received by a fixed observer from a moving target emitting a random broad-band white gaussian signal whose central frequency is `f0`. The target is moving along a straight line, which gets closer to the observer up to a distance `d`, and then moves away. `t0` is the time center (i.e. the time at which the target is at the closest distance from the observer), and `c` is the wave velocity in the medium.

Example

Consider such a noisy doppler signal and estimate its instantaneous frequency (see `instfreq`) :

```
[z,iflaw]=dopnoise(500,200,60,10,70,128);
subplot(211);    plot(real(z));
subplot(212);    plot(iflaw);   hold;
ifl=instfreq(z); plot(ifl,'g'); hold;
sum(abs(z).^2)
ans =
      1.0000
```

The frequency evolution is hardly visible from the time representation, whereas the instantaneous frequency estimation shows it with success. We check that the energy is equal to 1.

See Also

`doppler`, `noisecg`.

doppler

Purpose

Complex Doppler signal.

Synopsis

```
[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)
```

Description

`doppler` 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`.

Name	Description	Default value
`N`	number of points	
`fs`	sampling frequency (in Hz)	
`f0`	target frequency (in Hz)	
`d`	distance from the line to the observer (in meters)	
`v`	target velocity (in m/s)	
`t0`	time center	`N/2`
`c`	wave velocity (in m/s)	`340`
`fm`	output frequency modulation	
`am`	output amplitude modulation	
`iflaw`	output instantaneous frequency law	

The doppler effect characterizes the fact that a signal returned from a moving target is scaled and delayed compared to the transmitted signal. For narrow-band signals, this scaling effect can be considered as a frequency shift.

`[fm,am,iflaw] = doppler(N,fs,f0,d,v,t0,c)` returns the signal received by a fixed observer from a moving target emitting a pure frequency `f0`. The target is moving along a straight line, which gets closer to the observer up to a distance `d`, and then moves away. `t0` is the time center (i.e. the time at which the target is at the closest distance from the observer), and `c` is the wave velocity in the medium.

Example

Plot the signal and its instantaneous frequency law received by an observer from a car moving at the speed $v = 50m/s$, passing at 10 meters from the observer (the radar). The rotating frequency of the engine is $f_0 = 65Hz$, and the sampling frequency is $f_s = 200Hz$:

```
N=512; [fm,am,iflaw]=doppler(N,200,65,10,50);
subplot(211); plot(real(am.*fm));
subplot(212); plot(iflaw);
[ifhat,t]=instfreq(sigmerge(am.*fm,noisecg(N),15),11:502,10);
hold on; plot(t,ifhat,'g'); hold off;
```

See Also

`dopnoise`.

dwindow

Purpose

Derive a window.

Synopsis

```
dh = dwindow(h)
```

Description

`dwindow` derives the window `h`.

Example

```
h=window(200,'hanning');
subplot(211); plot(h);
subplot(212); plot(dwindow(h));
```

See Also

`window`.

fmconst

Purpose

Signal with constant frequency modulation.

Synopsis

```
[y,iflaw] = fmconst(N)
[y,iflaw] = fmconst(N,fnorm)
[y,iflaw] = fmconst(N,fnorm,t0)
```

Description

`fmconst` generates a frequency modulation with a constant frequency `fnorm` and unit amplitude. The phase of this modulation, determined by `t0`, is such that `y(t0)=1`. The signal is analytic.

Name	Description	Default value
`N`	number of points	
`fnorm`	normalised frequency	`0.25`
`t0`	time center	`N/2`
`y`	signal	
`iflaw`	instantaneous frequency law	

Example

```
z=amgauss(128,50,30).*fmconst(128,0.05,50);
plot(real(z));
```

represents the real part of a complex sinusoid of normalized frequency `0.05`, centered at `t0=50`, and with a gaussian amplitude modulation maximum at `t=t0`.

See Also

`fmlin, fmsin, fmodany, fmhyp, fmpar, fmpower.`

fmhyp

Purpose

Signal with hyperbolic frequency modulation or group delay law.

Synopsis

```
[x,iflaw] = fmhyp(N,P1)
[x,iflaw] = fmhyp(N,P1,P2)
```

Description

`fmhyp` generates a signal with a hyperbolic frequency modulation

$$x(t) = \exp\left(i2\pi\left(f_0 t + \frac{c}{log|t|}\right)\right).$$

Name	Description	Default value
`N`	number of points in time	
`P1`	if `nargin==2, P1` is a vector containing the two coefficients `[f0 c]`. If `nargin==3, P1` (as `P2`) is a time-frequency point of the form `[ti fi]`. `ti` is in seconds and `fi` is a normalized frequency (between 0 and 0.5). The coefficients `f0` and `c` are then deduced such that the frequency modulation law fits the points `P1` and `P2`	
`P2`	same as `P1` if `nargin==3`	optional
`x`	time row vector containing the modulated signal samples	
`iflaw`	instantaneous frequency law	

Examples

```
[X,iflaw]=fmhyp(100,[1 .5],[32 0.1]);
subplot(211); plot(real(X));
subplot(212); plot(iflaw);
```

See Also

`fmlin, fmsin, fmpar, fmconst, fmodany, fmpower.`

fmlin

Purpose

Signal with linear frequency modulation.

Synopsis

```
[y,iflaw] = fmlin(N)
[y,iflaw] = fmlin(N,fnormi)
[y,iflaw] = fmlin(N,fnormi,fnormf)
[y,iflaw] = fmlin(N,fnormi,fnormf,t0)
```

Description

`fmlin` generates a linear frequency modulation, going from `fnormi` to `fnormf`. The phase of this modulation is such that `y(t0)=1`.

Name	Description	Default value
`N`	number of points	
`fnormi`	initial normalized frequency	`0.0`
`fnormf`	final normalized frequency	`0.5`
`t0`	time reference for the phase	`N/2`
`y`	signal	
`iflaw`	instantaneous frequency law	

Example

```
z=amgauss(128,50,40).*fmlin(128,0.05,0.3,50);
plot(real(z));
```

See Also

`fmconst, fmsin, fmodany, fmhyp, fmpar, fmpower.`

fmodany

Purpose

Signal with arbitrary frequency modulation.

Synopsis

```
[y,iflaw] = fmodany(iflaw)
[y,iflaw] = fmodany(iflaw,t0)
```

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`.

Name	Description	Default value
`iflaw`	vector of the instantaneous frequency law samples	
`t0`	time reference	`1`
`y`	output signal	

Example

```
[y1,ifl1]=fmlin(100); [y2,ifl2]=fmsin(100);
iflaw=[ifl1;ifl2]; sig=fmodany(iflaw);
subplot(211); plot(real(sig))
subplot(212); plot(iflaw);
```

This example shows a signal composed of two successive frequency modulations : a linear FM followed by a sinusoidal FM.

See Also

`fmconst, fmlin, fmsin, fmpar, fmhyp, fmpower.`

fmpar

Purpose

Signal with parabolic frequency modulation.

Synopsis

```
[x,iflaw] = fmpar(N,P1)
[x,iflaw] = fmpar(N,P1,P2,P3)
```

Description

`fmpar` generates a signal with parabolic frequency modulation law :

$$x(t) = \exp(j2\pi(a_0 t + \frac{a_1}{2}t^2 + \frac{a_2}{3}t^3)).$$

Name	Description	Default value
`N`	number of points in time	
`P1`	if `nargin=2`, `P1` is a vector containing the three coefficients `(a0 a1 a2)` of the polynomial instantaneous phase. If `nargin=4`, P1 (as `P2` and `P3`) is a time-frequency point of the form `(ti fi)`. The coefficients `(a0,a1,a2)` are then deduced such that the frequency modulation law fits these three points	
`P2, P3`	same as `P1` if `nargin=4`.	optional
`x`	time row vector containing the modulated signal samples	
`iflaw`	instantaneous frequency law	

Examples

```
[x,iflaw]=fmpar(200,[1 0.4],[100 0.05],[200 0.4]);
subplot(211);plot(real(x));subplot(212);plot(iflaw);
[x,iflaw]=fmpar(100,[0.4 -0.0112 8.6806e-05]);
subplot(211);plot(real(x));subplot(212);plot(iflaw);
```

See Also

`fmconst, fmhyp, fmlin, fmsin, fmodany, fmpower.`

fmpower

Purpose

Signal with power-law frequency modulation.

Synopsis

```
[x,iflaw] = fmpower(N,k,P1)
[x,iflaw] = fmpower(N,k,P1,P2)
```

Description

`fmpower` generates a signal with a power-law frequency modulation :

$$x(t) = \exp(j2\pi(f_0 t + \frac{c}{1-k}|t|^{1-k})).$$

Name	Description	Default value
`N`	number of points in time	
`k`	degree of the power-law (`k`$\neq$1)	
`P1`	if `nargin==3, P1` is a vector containing the two coefficients `(f0 c)` for a power-law instantaneous frequency (sampling frequency is set to 1). If `nargin=4, P1` (as `P2`) is a time-frequency point of the form `(ti fi)`. `ti` is in seconds and `fi` is a normalized frequency (between 0 and 0.5). The coefficients `f0` and `c` are then deduced such that the frequency modulation law fits the points `P1` and `P2`	
`P2`	same as `P1` if `nargin=4`	optional
`x`	time row vector containing the modulated signal samples	
`iflaw`	instantaneous frequency law	

Example

```
[x,iflaw]=fmpower(200,0.5,[1 0.5],[180 0.1]);
subplot(211); plot(real(x));
subplot(212); plot(iflaw);
```

See Also

`gdpower, fmconst, fmlin, fmhyp, fmpar, fmodany, fmsin.`

fmsin

Purpose

Signal with sinusoidal frequency modulation.

Synopsis

```
[y,iflaw] = fmsin(N)
[y,iflaw] = fmsin(N,fmin)
[y,iflaw] = fmsin(N,fmin,fmax)
[y,iflaw] = fmsin(N,fmin,fmax,period)
[y,iflaw] = fmsin(N,fmin,fmax,period,t0)
[y,iflaw] = fmsin(N,fmin,fmax,period,t0,f0)
[y,iflaw] = fmsin(N,fmin,fmax,period,t0,f0,pm1)
```

Description

`fmsin` generates a sinusoidal frequency modulation, whose minimum frequency value is `fmin` and maximum is `fmax`. This sinusoidal modulation is designed such that the instantaneous frequency at time `t0` is equal to `f0`, and the ambiguity between increasing or decreasing frequency is solved by `pm1`.

Name	Description	Default value
`N`	number of points	
`fmin`	smallest normalized frequency	`0.05`
`fmax`	highest normalized frequency	`0.45`
`period`	period of the sinusoidal frequency modulation	`N`
`t0`	time reference for the phase	`N/2`
`f0`	normalized frequency at time `t0`	`0.25`
`pm1`	frequency direction at `t0` (-1 or +1)	`+1`
`y`	signal	
`iflaw`	instantaneous frequency law	

Example

```
z=fmsin(140,0.05,0.45,100,20,0.3,-1.0);
plot(real(z));
```

See Also

`fmconst, fmlin, fmodany, fmhyp, fmpar, fmpower.`

fmt

Purpose

Fast Mellin Transform.

Synopsis

```
[mellin,beta] = fmt(x)
[mellin,beta] = fmt(x,fmin,fmax)
[mellin,beta] = fmt(x,fmin,fmax,N)
```

Description

`fmt` computes the Fast Mellin Transform of signal `x`.

Name	Description	Default value
`x`	signal in time	
`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 between 0 and 0.5	
`N`	number of analyzed voices. `N` must be even	auto[a]
`mellin`	the `N`-points Mellin transform of signal `x`	
`beta`	the `N`-points Mellin variable	

The Mellin transform is invariant in modulus to dilations, and decomposes the signal on a basis of hyperbolic signals. This transform can be defined as :

$$M_x(\beta) = \int_0^{+\infty} x(\nu)\ \nu^{j2\pi\beta-1}\ d\nu$$

where $x(\nu)$ is the Fourier transform of the analytic signal corresponding to $x(t)$. The β-parameter can be interpreted as a *hyperbolic modulation rate*, and has no dimension ; it is called the *Mellin's scale*.

In the discrete case, the Mellin transform can be calculated rapidly using a fast Fourier transform (`fft`). The fast Mellin transform is used, for example, in the computation of the affine time-frequency distributions.

[a] This value, determined from `fmin` and `fmax`, is the next-power-of-two of the minimum value checking the non-overlapping condition in the fast Mellin transform.

Example

```
sig=altes(128,0.05,0.45);
[mellin,beta]=fmt(sig,0.05,0.5,128);
```

```
plot(beta,real(mellin));
```

See Also

`ifmt`, `fft`, `ifft`.

References

[1] J. Bertrand, P. Bertrand, J-P. Ovarlez “Discrete Mellin Transform for Signal Analysis” Proc IEEE-ICASSP, Albuquerque, NM USA, 1990.

[2] J-P. Ovarlez, J. Bertrand, P. Bertrand “Computation of Affine Time-Frequency Representations Using the Fast Mellin Transform” Proc IEEE-ICASSP, San Fransisco, CA USA, 1992.

friedman

Purpose

Instantaneous frequency density.

Synopsis

```
tifd = friedman(tfr,hat)
tifd = friedman(tfr,hat,t)
tifd = friedman(tfr,hat,t,method)
tifd = friedman(tfr,hat,t,method,trace)
```

Description

`friedman` computes the time-instantaneous frequency density (defined by Friedman [1]) of a reassigned time-frequency representation.

Name	Description	Default value
`tfr`	time-frequency representation, `(N,M)` matrix	
`hat`	complex matrix of the reassignment vectors	
`t`	time instant(s)	`(1:M)`
`method`	chosen representation	`'tfrrsp'`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tifd`	time instantaneous-frequency density. When called without output arguments, `friedman` runs `tfrqview`	

Warning : `tifd` is not an energy distribution, but an estimated probability distribution.

Example

Here is an example of such an estimated probability distribution operated on the reassigned pseudo-Wigner-Ville distribution of a linear frequency modulation :

```
sig=fmlin(128,0.1,0.4);
[tfr,rtfr,hat]=tfrrpwv(sig);
friedman(tfr,hat,1:128,'tfrrpwv',1);
```

The result is almost perfectly concentrated on a line in the time-frequency plane.

See Also

`ridges`.

Reference

[1] D.H. Friedman, ”Instantaneous Frequency vs Time : An Interpretation of the Phase Structure of Speech”, Proc. IEEE ICASSP, pp. 29.10.1-4, Tampa, 1985.

gdpower

Purpose

Signal with a power-law group delay.

Synopsis

```
[x,gpd,f] = gdpower(N)
[x,gpd,f] = gdpower(N,k)
[x,gpd,f] = gdpower(N,k,c)
```

Description

`gdpower` generates a signal with a power-law group delay of the form

$$t_x(f) = t_0 + c\, f^{k-1}.$$

The output signal is of unit energy.

Name	Description	Default value
`N`	number of points in time (must be even)	
`k`	degree of the power-law	`0`
`c`	rate-coefficient of the power-law group delay. `c` must be non-zero.	`1`
`x`	time row vector containing the signal samples	
`gpd`	output vector containing the group delay samples, of length `round(N/2)`	
`f`	frequency bins	

Examples

Consider a hyperbolic group-delay law, and compute the Bertrand distribution of it :

```
sig=gdpower(128);
tfrbert(sig,1:128,0.01,0.3,128,1);
```

We note that the perfect localization property of the Bertrand distribution on hyperbolic group-delay signals is checked in that case.

Plot the instantaneous frequency law on which the D-Flandrin distribution is perfectly concentrated :

```
[sig,gpd,f]=gdpower(128,1/2);
plot(gpd,f);
tfrdfla(sig,1:128,.01,.3,218,1);
```

See Also

`fmpower`.

holder

Purpose

Hlder exponent estimation through an affine TFR.

Synopsis

```
h = holder(tfr,f)
h = holder(tfr,f,n1)
h = holder(tfr,f,n1,n2)
h = holder(tfr,f,n1,n2,t)
```

Description

`holder` estimates the Hlder exponent of a signal through an affine time-frequency representation of it.

Name	Description	Default value
`tfr`	affine time-frequency representation	
`f`	frequency values of the spectral analysis	
`n1`	indice of the minimum frequency for the linear regression	`1`
`n2`	indice of the maximum frequency for the linear regression	`length(f)`
`t`	time vector. If `t` is omitted, the function returns the global estimate of the Hlder exponent. Otherwise, it returns the local estimates `h(t)` at the instants specified in `t`	
`h`	output value (if `t` omitted) or vector (otherwise) containing the Hlder estimate(s)	

Example

For instance, we consider a 64-points Lipschitz singularity (see `anasing`) of strength `h=0`, centered at `t0=32`, analyze it with the scalogram (Morlet wavelet with half-length = 4), and estimate its Hlder exponent,

```
sig=anasing(64);
[tfr,t,f]=tfrscalo(sig,1:64,4,0.01,0.5,256,1);
h=holder(tfr,f,1,256,1:64);
```

the value obtained at time `t0` is a good estimation of `h` (we obtain `h(t0)=-0.0381`).

See Also

`anastep, anapulse, anabpsk, doppler.`

Reference

[1] S. Jaffard "Exposants de Hlder en des points donns et coefficients d'ondelettes" C.R. de l'Acadmie des Sciences, Paris, t. 308, Srie I, p. 79-81, 1989.

[2] P. Gonalvs, P. Flandrin "Scaling Exponents Estimation From Time-Scale Energy Distributions" IEEE ICASSP-92, pp. V.157 - V.160, San Fransisco 1992.

htl

Purpose

Hough transform for detection of lines in images.

Synopsis

```
[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).
```

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. Only the values of `IM` exceeding 5 % of the maximum are taken into account (to speed up the algorithm).

Name	Description	Default value
`IM`	image to be analyzed (size `(Xmax,Ymax)`)	
`M`	desired number of samples along the radial axis	`Xmax`
`N`	desired number of samples along the azimutal (angle) axis	`Ymax`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`HT`	output matrix (`MxN` matrix)	
`rho`	sequence of samples along the radial axis	
`theta`	sequence of samples along the azimutal axis	

When called without output arguments, `htl` displays `HT` using `mesh`.

Example

The Wigner-Ville distribution of a linear frequency modulation is almost perfectly concentrated (in the discrete case) on a straight line in the time-frequency plane. Thus, applying the Hough transform on this image will produce a representation with a peak, whose coordinates give estimates of the linear frequency modulation parameters (initial frequency and sweep rate) :

```
N=64; t=(1:N); y=fmlin(N,0.1,0.3);
IM=tfrwv(y,t,N); imagesc(IM); pause(1);
htl(IM,N,N,1);
```

Reference

[1] H. Matre “Un Panorama de la Transformation de Hough”, Traitement du Signal, Vol 2, No 4, pp. 305-317, 1985.

ifestar2

Purpose

Instantaneous frequency estimation using AR2 modelisation.

Synopsis

```
[fnorm,t2,ratio] = ifestar2(x)
[fnorm,t2,ratio] = ifestar2(x,t)
```

Description

`ifestar2` computes an estimation of the instantaneous frequency of the real signal `x` at time instant(s) `t` using an auto-regressive model of order 2. 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.

Name	Description	Default value
`x`	real signal to be analyzed	
`t`	time instants (must be greater than 4)	`(4:length(x))`
`fnorm`	output (normalized) instantaneous frequency	
`t2`	time instants coresponding to `fnorm`. Since the algorithm do not systematically give a value, `t2` is different from `t` in general	
`ratio`	proportion of instants where the algorithm yields an estimation	

This estimator is the causal version of the estimator called "4 points Prony estimator" in article [1].

Example

Here is a comparison between the instantaneous frequency estimated by `ifestar2` and the exact instantaneous frequency law, obtained on a sinusoidal frequency modulation :

```
[x,if]=fmsin(100,0.1,0.4); x=real(x);
[if2,t]=ifestar2(x);
plot(t,if(t),t,if2);
```

The estimation follows quite correctly the right law, but with a small bias and with some weak oscillations.

See Also

`instfreq`, `kaytth`, `sgrpdlay`.

Reference

[1] Prony ”Instantaneous frequency estimation using linear prediction with comparisons to the dESAs”, IEEE Signal Processing Letters, Vol 3, No 2, p 54-56, February 1996.

ifmt

Purpose

Inverse fast Mellin transform.

Synopsis

```
x = ifmt(mellin,beta)
x = ifmt(mellin,beta,M)
```

Description

`ifmt` 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.

Name	Description	Default value
`mellin`	Mellin transform to be inverted. `mellin` must have been obtained from `fmt` with frequency running from `fmin` to 0.5 Hz	
`beta`	Mellin variable issued from `fmt`	
`M`	number of points of the inverse Mellin transform	`length(mellin)`
`x`	inverse Mellin transform with `M` points in time	

Example

To check the perfect reconstruction property of the inverse Mellin transform, we consider an analytic signal, compute its fast Mellin transform with an upper frequency bound of 0.5, and apply on the output vector the `ifmt` algorithm :

```
sig=atoms(128,[64,0.25,32,1]); clf;
[mellin,beta]=fmt(sig,0.08,0.5,128);
x=ifmt(mellin,beta,128); plot(abs(x-sig));
```

We can observe the almost perfect equality between `x` and `sig`.

See Also

`fmt, fft, ifft.`

instfreq

Purpose

Instantaneous frequency estimation.

Synopsis

```
[fnormhat,t] = instfreq(x)
[fnormhat,t] = instfreq(x,t)
[fnormhat,t] = instfreq(x,t,l)
[fnormhat,t] = instfreq(x,t,l,trace)
```

Description

`instfreq` computes the estimation of 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.

Name	Description	Default value
`x`	analytic signal to be analyzed	
`t`	time instants	`(2:length(x)-1)`
`l`	if `l=1`, computes the estimation of the (normalized) instantaneous frequency of `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. `l` must be an integer	`1`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`fnormhat`	output (normalized) instantaneous frequency	

Examples

Consider a linear frequency modulation and estimate its instantaneous frequency law with `instfreq`:

```
[x,ifl]=fmlin(70,0.05,0.35,25);
[instf,t]=instfreq(x);
plotifl(t,[ifl(t) instf]);
```

Now consider a noisy sinusoidal frequency modulation with a signal to noise ratio of 10 dB :

```
N=64; SNR=10.0; L=4; t=L+1:N-L;
x=fmsin(N,0.05,0.35,40);
sig=sigmerge(x,hilbert(randn(N,1)),SNR);
plotifl(t,[instfreq(sig,t,L),instfreq(x,t)]);
```

See Also

`ifestar2, kaytth, sgrpdlay.`

Reference

[1] I. Vincent, F. Auger, C. Doncarli “A Comparative Study Between Two Instantaneous Frequency Estimators”, Proc Eusipco-94, Vol. 3, pp. 1429-1432, 1994.

[2] P. Djuric, S. Kay “Parameter Estimation of Chirp Signals” IEEE Trans. on Acoust. Speech and Sig. Proc., Vol. 38, No. 12, 1990.

[3] S.M. Tretter “A Fast and Accurate Frequency Estimator”, IEEE Trans. on ASSP, Vol. 37, No. 12, pp. 1987-1990, 1989.

integ

Purpose

Approximate integral.

Synopsis

```
som = integ(y)
som = integ(y,x)
```

Description

`integ` approximates the integral of vector `y` according to `x`.

Name	Description	Default value
`y`	`N`-row-vector (or `(M,N)`-matrix) to be integrated (along each row).	
`x`	`N`-row-vector containing the integration path of `y`	`(1:N)`
`som`	value (or `(M,1)` vector) of the integral	

Example

```
y = altes(256,0.1,0.45,10000)';
x = (0:255); som = integ(y,x)
som =
      2.0086e-05
```

See Also

`integ2d`.

integ2d

Purpose

Approximate 2-D integral.

Synopsis

```
som = integ2d(MAT)
som = integ2d(MAT,x)
som = integ2d(MAT,x,y)
```

Description

`integ2d` approximates the 2-D integral of matrix `MAT` according to abscissa `x` and ordinate `y`.

Name	Description	Default value
`MAT`	`(M,N)` matrix to be integrated	
`x`	`N`-row-vector indicating the abscissa integration path	`(1:N)`
`y`	`M`-column-vector indicating the ordinate integration path	`(1:M)`
`som`	result of integration	

Example

Consider the scalogram of a sinusoidal frequency modulation of 128 points, and compute the integral over the time-scale plane of the scalogram :

```
S = fmsin(128,0.2,0.3);
[TFR,t,f] = tfrscalo(S,1:128,8,0.1,0.4,128,1);
Etfr = integ2d(TFR,t,f)
Etfr =
       128.0000
```

We find for `Etfr` the value of the signal energy, which is the expected value since the scalogram preserves energy.

See Also

`integ`.

izak

Purpose

Inverse Zak transform.

Synopsis

```
sig = izak(DZT)
```

Description

`izak` computes the inverse Zak transform of matrix `DZT`.

Name	Description	Default value
`DZT`	`(N,M)` matrix of Zak samples (obtained with `zak`)	
`sig`	output signal `(M*N,1)` containing the inverse Zak transform	

Example

If we compute the discrete Zak transform of a signal and apply on the output matrix the inverse Zak transform, we should obtain again the original signal :

```
sig=fmlin(250); DZT=zak(sig); sigr=izak(DZT);
plot(real(sigr-sig));
```

See Also

`zak, tfrgabor.`

kaytth

Purpose

Kay-Tretter filter computation.

Synopsis

```
h = kaytth(N);
```

Description

`kaytth` computes the Kay-Tretter filter.

Name	Description	Default value
`N`	length of the filter	
`h`	impulse response of the filter	

This filter is used in the computation of `instfreq`.

See Also

`instfreq`.

Reference

[1] P. Djuric and S. Kay ”Parameter Estimation of Chirp Signals” IEEE Trans. on Acoust. Speech and Sig. Proc., Vol 38, No 12, 1990.

[2] S.M. Tretter “A Fast and Accurate Frequency Estimator”, IEEE Trans. on ASSP, Vol. 37, No. 12, pp. 1987-1990, 1989.

klauder

Purpose

Klauder wavelet in time domain.

Synopsis

```
x = klauder(N)
x = klauder(N,lambda)
x = klauder(N,lambda,f0)
```

Description

`klauder` generates the Klauder wavelet in the time domain :

$$K(f) = e^{-2\pi\lambda f} f^{2\pi\lambda f_0 - 1/2}.$$

Name	Description	Default value
`N`	number of points in time	
`lambda`	attenuation factor or the envelope	`10`
`f0`	central frequency of the wavelet	`0.2`
`x`	time row vector containing the klauder samples	

Example

```
x=klauder(150,50,0.1);
plot(x);
```

See Also

`altes, anasing, doppler, anafsk, anastep.`

locfreq

Purpose

Frequency localization characteristics.

Synopsis

```
[fm,B] = locfreq(x)
```

Description

`locfreq` computes the frequency localization characteristics of signal `x`. The definition used for the averaged frequency and the frequency spreading are the following :

$$
\begin{aligned}
f_m &= \frac{1}{E_x} \int_{-\infty}^{+\infty} \nu \, |X(\nu)|^2 \, d\nu \\
B &= 2 \sqrt{\frac{\pi}{E_x} \int_{-\infty}^{+\infty} (\nu - f_m)^2 \, |X(\nu)|^2 \, d\nu}
\end{aligned}
$$

where E_x is the energy of the signal and $X(\nu)$ the Fourier transform of $x(t)$. With this definition (and the one used in `loctime`), the Heisenberg-Gabor inequality writes $B\,T \geq 1$.

Name	Description	Default value
`x`	signal	
`fm`	averaged normalized frequency center	
`B`	frequency spreading	

Example

```
z=amgauss(160,80,50).*fmconst(160,0.2);
[fm,B]=locfreq(z); [fm,B]
ans =
      0.2000    0.0200
```

See Also

`loctime`.

loctime

Purpose

Time localization characteristics.

Synopsis

```
[tm,T] = loctime(x)
```

Description

`loctime` computes the time localization characteristics of signal `x`. The definition used for the averaged time and the time spreading are the following :

$$
\begin{aligned}
t_m &= \frac{1}{E_x} \int_{-\infty}^{+\infty} t\, |x(t)|^2\, dt \\
T &= 2 \sqrt{\frac{\pi}{E_x} \int_{-\infty}^{+\infty} (t - t_m)^2\, |x(t)|^2\, dt}
\end{aligned}
$$

where E_x is the energy of the signal. With this definition (and the one used in `locfreq`), the Heisenberg-Gabor inequality writes $B\,T \geq 1$.

Name	Description	Default value
`x`	signal	
`tm`	averaged time center	
`T`	time spreading	

Examples

Here is an example of signal which corresponds to the lower bound of the Heisenberg-Gabor inequality.

```
z=amgauss(160,80,50);
[tm,T]=loctime(z);
[fm,B]=locfreq(z);
[tm,T,fm,B,T*B]
ans =
      80.0000   50.0000   0.0000   0.0200   1
```

See Also

`locfreq`.

margtfr

Purpose

Marginals and energy of a time-frequency representation.

Synopsis

```
[margt,margf,E] = margtfr(tfr)
[margt,margf,E] = margtfr(tfr,t)
[margt,margf,E] = margtfr(tfr,t,f)
```

Description

`margtfr` calculates the time and frequency marginals and the energy of a time-frequency representation. The definitions used for the computation are the following :

$$m_f(t) = \int_{-\infty}^{+\infty} \mathrm{tfr}(t,f)\; df \qquad \textit{time marginal}$$

$$m_t(f) = \int_{-\infty}^{+\infty} \mathrm{tfr}(t,f)\; dt \qquad \textit{frequency marginal}$$

$$E = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \mathrm{tfr}(t,f)\; df\; dt \qquad \textit{energy}$$

Name	Description	Default value
`tfr`	time-frequency representation `(M,N)`	
`t`	vector containing the time samples in sec.	`(1:N)`
`f`	vector containing the frequency samples in Hz, not necessary uniformly sampled	`(1:M)`
`margt`	time marginal	
`margf`	frequency marginal	
`E`	energy of `tfr`	

Example

```
S=amgauss(128).*fmlin(128);
[tfr,t,f]=tfrscalo(S,1:128,8,.05,.45,128,1);
[margt,margf,E] = margtfr(tfr);
subplot(211); plot(t,margt);
subplot(212); plot(f,margf);
```

See Also

`momttfr, momftfr.`

mexhat

Purpose

Mexican hat wavelet in time domain.

Synopsis

```
h = mexhat
h = mexhat(nu)
```

Description

`mexhat` returns the mexican hat wavelet, with central frequency `nu` (`nu` is a normalized frequency). Its expression writes

$$h(t) = \nu\,\frac{\sqrt{\pi}}{2}\,(1-2(\pi\nu t)^2)\,\exp[-(\pi\nu\,t)^2].$$

Name	Description	Default value
`nu`	any real between 0 and 0.5	`0.05`
`h`	time vector containing the mexhat samples `length(h)=2*ceil(1.5/nu)+1`	

Example

```
plot(mexhat);
```

See Also

`klauder`.

midscomp

Purpose

Mid-point construction used in the interference diagram.

Synopsis

```
[ti,fi] = midpoint(t1,f1,t2,f2,K)
```

Description

`midscomp` 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 $t_x(\nu) = t_0 + c\ \nu^{K-1}$. This function is mainly called by `plotsid`.

Name	Description
`t1`	time-coordinate of the first point
`f1`	frequency-coordinate of the first point (> 0)
`t2`	time-coordinate of the second point
`f2`	frequency-coordinate of the second point (> 0)
`K`	power of the group-delay law. Example of distributions satisfying this interference construction : `K = 2` : Wigner-Ville distribution `K = 1/2` : D-Flandrin distribution `K = 0` : Bertrand (unitary) distribution `K = -1` : Unterberger (active) distribution `K = Inf` : Margenau-Hill-Rihaczek distribution
`ti`	time-coordinate (abscissa) of the interference-point
`fi`	frequency-coordinate (ordinate) of the interference-point

Example

Here is the locus of the interference terms between two points, for `K` going from -15 to 15 :

```
t1=10; f1=0.45; t2=90; f2=0.05; hold on
for K=-15:15,
 [ti(2*K+31),fi(2*K+31)]=midscomp(t1,f1,t2,f2,K);
end
```

See Also

`plotsid`.

modulo

Purpose

Congruence of a vector.

Synopsis

```
y = modulo(x,N)
```

Description

`modulo` gives the congruence of each element of the vector `x` modulo `N`. These values are strictly positive and lower equal than `N`.

Name	Description	Default value
`x`	vector of real values, positive or negative	
`N`	congruence number (not necessarily an integer)	
`y`	output vector of real values, >0 and $\leq$`N`	

Example

```
x=[1.3 -2.13 9.2 0 -13 2];
modulo(x,2)
ans =
      1.3000    1.8700    1.2000    2.0000    1.0000    2.0000
```

See Also

`rem`.

momftfr

Purpose

Frequency moments (order 1 and 2) of a time-frequency representation.

Synopsis

```
[tm,T2] = momftfr(tfr)
[tm,T2] = momftfr(tfr,tmin)
[tm,T2] = momftfr(tfr,tmin,tmax)
[tm,T2] = momftfr(tfr,tmin,tmax,time)
```

Description

`momftfr` computes the frequeny moments of order 1 and 2 of a time-frequency representation :

$$t_m(f) = \frac{1}{E}\int_{-\infty}^{+\infty} t\ \mathrm{tfr}(t,f)\ dt\ \ ;\ \ T^2(f) = \frac{1}{E}\int_{-\infty}^{+\infty} t^2\ \mathrm{tfr}(t,f)\ dt - t_m(f)^2.$$

Name	Description	Default value
`tfr`	time-frequency representation (size `(N,M)`).	
`tmin`	smallest column element of `tfr` taken into account	`1`
`tmax`	highest column element of `tfr` taken into account	`M`
`time`	true time instants	`(1:M)`
`tm`	averaged time (first order moment)	
`T2`	squared time duration (second order moment)	

Example

```
sig=fmlin(200,0.1,0.4); [tfr,t,f]=tfrwv(sig);
[tm,T2]=momftfr(tfr);
subplot(211); plot(f,tm); subplot(212); plot(f,T2);
```

The first order moment represents an estimation of the group delay, and the second order moment the variance of this estimator. We can see that the estimation is better around the time center position than at the edges of the observation interval.

See Also

`momttfr, margtfr.`

momttfr

Purpose

Time moments of a time-frequency representation.

Synopsis

```
[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)
```

Description

`momttfr` computes the time moments of order 1 and 2 of a time-frequency representation :

$$f_m(t) = \frac{1}{E} \int_{-\infty}^{+\infty} f \ \mathrm{tfr}(t,f)\ df \ \ ; \ \ B^2(t) = \frac{1}{E} \int_{-\infty}^{+\infty} f^2 \ \mathrm{tfr}(t,f)\ df - f_m(t)^2.$$

Name	Description	Default value
`tfr`	time-frequency representation (size `(N,M)`)	
`method`	chosen representation (name of the corresponding M-file).	
`fbmin`	smallest frequency bin	`1`
`fbmax`	highest frequency bin	`M`
`freqs`	true frequency of each frequency bin. `freqs` must be of length `fbmax-fbmin+1`	auto[a]
`fm`	averaged frequency (first order moment)	
`B2`	squared frequency bandwidth (second order moment)	

[a] `freqs` goes from 0 to 0.5 or from -0.5 to 0.5 depending on `method`.

Examples

```
sig=fmlin(200,0.1,0.4); tfr=tfrwv(sig);
[fm,B2]=momttfr(tfr,'tfrwv');
subplot(211); plot(fm); subplot(212); plot(B2);
freqs=linspace(0,99/200,100); tfr=tfrsp(sig);
[fm,B2]=momttfr(tfr,'tfrsp',1,100,freqs);
subplot(211); plot(fm); subplot(212); plot(B2);
```

The first order moment represents an estimation of the instantaneous frequency, and the second order moment the variance of this estimator. We can see that the estimation is better around the time center position than at the edges of the observation interval. Besides, the second estimator (using the spectrogram) has a lower variance than the first one (using the Wigner-Ville distribution), but presents an important bias.

See Also

`momftfr`, `margtfr`.

movcw4at

Purpose

Four atoms rotating, analyzed by the Choi-Williams distribution.

Synopsis

```
M = movcw4at(N)
M = movcw4at(N,Np)
```

Description

`movcw4at` generates the movie frames illustrating the influence of an overlapping in time and/or frequency of different components of a signal on the interferences of the Choi-Williams distribution between these components.

Name	Description	Default value
`N`	number of points of the analyzed signal	
`Np`	number of snapshots	`7`
`M`	matrix of movie frames	

Example

```
M=movcw4at(128,15);
movie(M,10);
```

See Also

`movpwjph, movpwdph, movsc2wv, movsp2wv, movwv2at.`

movpwdph

Purpose

Influence of a phase-shift on the interferences of the PWVD.

Synopsis

```
M = movpwdph(N)
M = movpwdph(N,Np)
M = movpwdph(N,Np,typesig)
```

Description

`movpwdph` generates the movie frames illustrating the influence of a phase-shift between two signals on the interference terms of the pseudo Wigner-Ville distribution.

Name	Description	Default value
`N`	number of points for the signal	
`Np`	number of snapshots	`8`
`typesig`	type of signal	`’C’`
	`’C’` : constant frequency modulation	
	`’L’` : linear frequency modulation	
	`’S’` : sinusoidal frequency modulation	
`M`	matrix of movie frames	

Example

```
M=movpwdph(128,8,’S’);
movie(M,10);
```

See Also

`movpwjph, movcw4at, movsc2wv, movsp2wv, movwv2at.`

movpwjph

Purpose

Influence of a jump of phase on the interferences of the PWVD.

Synopsis

```
M = movpwjph(N)
M = movpwjph(N,Np)
M = movpwjph(N,Np,typesig)
```

Description

`movpwjph` generates the movie frames illustrating the influence of a jump of phase in different frequency modulations on the interference terms of the pseudo Wigner-Ville distribution.

Name	Description	Default value
`N`	number of points for the signal	
`Np`	number of snapshots	`8`
`typesig`	type of signal	`'C'`
	`'C'` : constant frequency modulation	
	`'L'` : linear frequency modulation	
	`'S'` : sinusoidal frequency modulation	
`M`	matrix of movie frames	

Example

```
M=movpwjph(128,8,'S');
movie(M,10);
```

See Also

`movcw4at, movpwdph, movsc2wv, movsp2wv, movwv2at.`

movsc2wv

Purpose

Movie illustrating the passage from the scalogram to the WVD.

Synopsis

```
M = movsc2wv(N)
M = movsc2wv(N,Np)
```

Description

`movsc2wv` generates the movie frames illustrating the passage from the scalogram to the WVD using the affine smoothed pseudo-WVD with different smoothing gaussian windows.

Name	Description	Default value
`N`	number of points of the analyzed signal	
`Np`	number of snapshots	`8`
`M`	matrix of movie frames	

Example

```
M=movsc2wv(64,8);
movie(M,10);
```

See Also

`movpwjph`, `movpwdph`, `movcw4at`, `movsp2wv`, `movwv2at`.

movsp2wv

Purpose

Movie illustrating the passage from the spectrogram to the WVD.

Synopsis

```
M = movsp2wv(N)
M = movsp2wv(N,Np)
```

Description

`movsp2wv` generates the movie frames illustrating the passage from the spectrogram to the WVD using the smoothed pseudo-WVD with different smoothing gaussian windows.

Name	Description	Default value
`N`	number of points of the analyzed signal	
`Np`	number of snapshots	8
`M`	matrix of movie frames	

Example

```
M=movsp2wv(128,15);
movie(M,10);
```

See Also

`movpwjph`, `movpwdph`, `movsc2wv`, `movcw4at`, `movwv2at`.

movwv2at

Purpose

Oscillating structure of the interferences of the WVD.

Synopsis

```
M = movwv2at(N)
M = movwv2at(N,Np)
```

Description

`movwv2at` generates the movie frames illustrating the influence of the distance between two components on the oscillating structure of the interferences of the WVD.

Name	Description	Default value
`N`	number of points of the analyzed signal	
`Np`	number of snapshots	9
`M`	matrix of movie frames	

Example

```
M=movwv2at(128,15);
movie(M,10);
```

See Also

`movpwjph`, `movpwdph`, `movsc2wv`, `movsp2wv`, `movcw4at`.

noisecg

Purpose

Analytic complex gaussian noise (white or colored).

Synopsis

```
noise = noisecg(N)
noise = noisecg(N,a1)
noise = noisecg(N,a1,a2)
```

Description

`noisecg` computes an analytic complex gaussian noise of length `N` with mean 0 and variance 1.0.

Name	Description	Default value
`N`	length of the output vector	
`a1`	first coefficient of the auto-regressive filter used to color the noise	`0`
`a2`	second coefficient of the auto-regressive filter used to color the noise	`0`
`noise`	output vector containing the noise samples	

`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 first order filter whose impulse response is

$$H(z) \ = \ \frac{\sqrt{1-a_1^2}}{1-a_1\ z^{-1}}.$$

`noise=noisecg(N,a1,a2)` yields a complex colored gaussian noise obtained by filtering a white gaussian noise through a second order filter whose impulse response is

$$H(z) \ = \ \frac{\sqrt{1-a_1^2-a_2^2}}{1-a_1\ z^{-1}-a_2\ z^{-2}}.$$

Example

```
N=500; noise=noisecg(N);
[abs(mean(noise)),std(noise).^2]
ans =
      0.0152    0.9680

subplot(211); plot(real(noise)); axis([1 N -3 3]);
subplot(212); f=linspace(-0.5,0.5,N);
plot(f,abs(fftshift(fft(noise))).^2);
```

See Also

rand, randn, noisecu.

noisecu

Purpose

Analytic complex uniform white noise.

Synopsis

```
noise = noisecu(N)
```

Description

`noisecu` computes an analytic complex white uniform noise of length `N` with mean 0 and variance 1.0.

Example

```
N=512; noise=noisecu(N);
[abs(mean(noise)),std(noise).^2]
ans =
      0.0099    1.0000

subplot(211); plot(real(noise)); axis([1 N -1.5 1.5]);
subplot(212); f=linspace(-0.5,0.5,N);
plot(f,abs(fftshift(fft(noise))).^2);
```

See Also

`rand, randn, noisecg.`

odd

Purpose

Round towards nearest odd value.

Synopsis

```
y = odd(x)
```

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. `x` can be a scalar, a vector or a matrix.

Name	Description	Default value
`x`	scalar, vector or matrix to be rounded	
`y`	output scalar, vector or matrix containing only odd values	

Example

```
x=[1.3 2.08 -3.4 90.43];
y=odd(x)
ans =
      1     3    -3    91
```

See Also

`round`, `ceil`, `fix`, `floor`.

plotifl

Purpose

Plot normalized instantaneous frequency laws.

Synopsis

```
plotifl(t,iflaws)
```

Description

`plotifl` plot the normalized instantaneous frequency laws of each signal component.

Name	Description	Default value
`t`	time instants (size `(M,1)`)	
`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 instants. The values of `iflaws` must be between -0.5 and 0.5.	

Example

```
N=140; t=0:N-1; [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];
plotifl(t,[if1 if2]);
```

See Also

`plotsid, tfrqview, tfrview.`

plotsid

Purpose

Schematic interference diagram of FM signals.

Synopsis

```
plotsid(t,iflaws)
plotsid(t,iflaws,K)
```

Description

`plotsid` plots the schematic interference diagram of any distribution in the affine class which is perfectly localized for signals with a power-law group-delay of the form $t_x(\nu) = t_0 + c\,\nu^{K-1}$. This diagram is computed for any (analytic) FM signal.

Name	Description	Default value
`t`	time instants	
`iflaws`	matrix of instantaneous frequencies, with as many columns as signal components	
`K`	distribution parameter `K = 2` : Wigner-Ville distribution `K = 1/2` : D-Flandrin distribution `K = 0` : Bertrand (unitary) distribution `K = -1` : Unterberger (active) distribution `K = inf` : Margenhau-Hill-Rihaczek dist.	2

Example

Here is the interference diagram corresponding to the Bertrand distribution, for a signal composed of two components : a linear and a constant frequency modulation :

```
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, midpoint, tfrqview, tfrview.`

renyi

Purpose

Measure Renyi information.

Synopsis

```
R = renyi(tfr)
R = renyi(tfr,t)
R = renyi(tfr,t,f)
R = renyi(tfr,t,f,alpha)
```

Description

`renyi` measures the Renyi information relative to a 2-D density function `tfr` (which can be eventually a time-frequency representation). Renyi information of order α is defined as :

$$R_x^\alpha = \frac{1}{1-\alpha}\, log_2 \left\{ \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \mathrm{tfr}_x^\alpha(t,\nu)\, dt\, d\nu \right\}$$

The result produced by this measure is expressed in *bits* : if one elementary signal yields zero bit of information (2^0), then two well separated elementary signals will yield one bit of information (2^1), four well separated elementary signals will yield two bits of information (2^2), and so on.

Name	Description	Default value
`tfr`	`(M,N)` 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`	abscissa vector parametrizing the `tfr` matrix. `t` can be a non-uniform sampled vector (eventually a time vector)	`(1:N)`
`f`	ordinate vector parametrizing the `tfr` matrix. `f` can be a non-uniform sampled vector (eventually a frequency vector)	`(1:M)`
`alpha`	rank of the Renyi measure	`3`
`R`	the alpha-rank Renyi measure (in bits if `tfr` is a time-frequency matrix).	

Examples

```
s=atoms(64,[32,.25,16,1]); [tfr,t,f]=tfrsp(s);
R1=renyi(tfr,t,f,3)
ans =
      0.9861

s=atoms(64,[16,.2,16,1;48,.3,16,1]); [tfr,t,f]=tfrsp(s);
R2=renyi(tfr,t,f,3)
ans =
      1.9890
```

We can see that if `R` is set to 0 for one elementary atom by subtracting `R1`, we obtain a result close to 1 bit for two atoms (`R2-R1`=1.0029).

Reference

[1] W. Williams, M. Brown, A. Hero III, "Uncertainty, information and time-frequency distributions", SPIE Advanced Signal Processing Algorithms, Architectures and Implementations II, Vol. 1566, pp. 144-156, 1991.

ridges

Purpose

Extraction of ridges from a reassigned TF representation.

Synopsis

```
[ptt,ptf] = ridges(tfr,hat,t,method)
[ptt,ptf] = ridges(tfr,hat,t,method,trace)
```

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.

Name	Description	Default value
`tfr`	time-frequency representation	
`hat`	complex matrix of the reassignment vectors	
`t`	the time instant(s)	
`method`	the chosen representation	
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`ptt`, `ptf`	two vectors for the time and frequency coordinates of the stationary points of the reassignment. Therefore, `plot(ptt,ptf,'.')` shows the squeleton of the representation	

When called without output arguments, `ridges` runs `plot(ptt,ptf,'.')`.

Example

Consider the ridges of the smoothed-pseudo WVD of a linear chirp signal :

```
sig=fmlin(128,0.1,0.4); t=1:2:127;
[tfr,rtfr,hat]=tfrrspwv(sig,t,128);
ridges(tfr,hat,t,'tfrrspwv',1);
```

The points obtained are almost perfectly localized on the instantaneous frequency law of the signal.

See Also

`friedman`.

scale

Purpose

Scale a signal using the Mellin transform.

Synopsis

```
S = scale(x,a,fmin,fmax,N)
```

Description

`scale` computes the `a`-scaled version of signal `x` : $x_a(t) \;=\; a^{-\frac{1}{2}}\; x(\frac{t}{a})$ using the Mellin transform.

Name	Description	Default value
`x`	signal in time to be scaled (`Nx=length(x)`)	
`a`	scale factor. `a` < 1 corresponds to a compression in the time domain and `a` > 1 to a dilation. `a` can be a vector.	`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 $\leq$`0.5`	
`N`	number of analyzed voices	auto[a]
`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`.	

[a]This value, determined from `fmin` and `fmax`, is the next-power-of-two of the minimum value checking the non-overlapping condition in the fast Mellin transform.

Example

Dilate a Klauder-wavelet by a factor of 2 :

```
sig=klauder(100); S=scale(sig,2,.05,.45,100);
subplot(211); plot(sig);
subplot(212); plot(real(S(51:150)));
```

See Also

`fmt`.

sgrpdlay

Purpose

Group delay estimation of a signal.

Synopsis

```
[gd,fnorm] = sgrpdlay(x)
[gd,fnorm] = sgrpdlay(x,fnorm)
```

Description

`sgrpdlay` estimates the group delay of a signal `x` at the normalized frequency(ies) `fnorm`.

Name	Description	Default value
`x`	signal in the time-domain (`N=length(x)`)	
`fnorm`	normalized frequency	`linspace(-.5,.5,N)`
`gd`	output vector containing the group delay samples. 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.	

Example

Let us compare the estimated group-delay and instantaneous frequency of a linear chirp signal :

```
N=128; x=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);
plot(t,instf,gd,fnorm); axis([1 N 0 0.5]);
```

The two curves are almost superposed, which is normal for a large time-bandwidth product signal.

See Also

`instfreq`.

sigmerge

Purpose

Add two signals with a given energy ratio in dB.

Synopsis

```
x = xmerge(x1,x2)
x = sigmerge(x1,x2,ratio)
```

Description

`sigmerge` adds two signals so that a given energy ratio expressed in deciBels is satisfied :

```
x=x1+h*x2,
```

such that

```
20*log(norm(x1)/norm(h*x2))=ratio.
```

Name	Description	Default value
`x1, x2`	input signals	
`ratio`	energy ratio in deciBels	`0 dB`
`x`	output signal	

Example

```
x=fmlin(64,0.01,0.05,1); noise=hilbert(randn(64,1));
SNR=15; xn=sigmerge(x,noise,SNR);
Ex=mean(abs(x).^2); Enoise=mean(abs(xn-x).^2);
10*log10(Ex/Enoise)
ans =
      15.0000
```

See Also

`noisecg`.

tfrbert

Purpose

Unitary Bertrand time-frequency distribution.

Synopsis

```
[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)
```

Description

`tfrbert` generates the auto- or cross- unitary Bertrand distribution, defined as

$$B_x(t,\nu) = \nu \int_{-\infty}^{+\infty} \frac{u/2}{\sinh\left(\frac{u}{2}\right)}\; X\left(\frac{\nu\; u\; e^{-u/2}}{2\sinh\left(\frac{u}{2}\right)}\right)\; X^*\left(\frac{\nu\; u\; e^{+u/2}}{2\sinh\left(\frac{u}{2}\right)}\right)\; e^{-j2\pi\nu u t}\, du$$

where $X(\nu)$ is the Fourier transform of $x(t)$.

Name	Description	Default value
`x`	signal (in time) to be analyzed. If `x=[x1 x2]`, `tfrbert` computes the cross-unitary Bertrand distribution `(Nx=length(x))`	
`t`	time instant(s) on which the `tfr` is evaluated	`(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	auto[a]
`trace`	if nonzero, the progression of the algorithm is shown	`0`

[a] This value, determined from `fmin` and `fmax`, is the next-power-of-two of the minimum value checking the non-overlapping condition in the fast Mellin transform.

Name	Description	Default value
`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`)	

When called without output arguments, `tfrbert` runs `tfrqview`

Example

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

See Also

all the `tfr*` functions.

References

[1] J. Bertrand, P. Bertrand "Time-Frequency Representations of Broad-Band Signals" IEEE ICASSP-88, pp. 2196-2199, New-York, 1988.

[2] J. Bertrand, P. Bertrand "A Class of Affine Wigner Functions with Extended Covariance Properties", J. Math. Phys., Vol. 33, No. 7, July 1992.

tfrbj

Purpose

Born-Jordan time-frequency distribution.

Synopsis

```
[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)
```

Description

`tfrbj` computes the Born-Jordan distribution of a discrete-time signal `x`, or the cross Born-Jordan representation between two signals. This distribution has the following expression :

$$BJ_x(t,\nu) = \int_{-\infty}^{+\infty} \frac{1}{|\tau|} \int_{t-|\tau|/2}^{t+|\tau|/2} x(s+\tau/2)\, x^*(s-\tau/2)\, ds\, e^{-j2\pi\nu\tau} d\tau.$$

Name	Description	Default value
`x`	signal if auto-BJ, or `[x1,x2]` if cross-BJ. `Nx=length(x)`	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`g`	time smoothing window with odd length, `g(0)` being forced to `1`	`window(odd(N/10))`
`h`	frequency smoothing window with odd length, `h(0)` being forced to `1`	`window(odd(N/4))`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency representation	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrbj` runs `tfrqview`.

Example

```
sig=fmlin(128,0.05,0.3)+fmlin(128,0.15,0.4);
g=window(9,'Kaiser'); h=window(27,'Kaiser');
t=1:128; tfrbj(sig,t,128,g,h,1);
```

See Also

all the `tfr*` functions.

Reference

[1] L. Cohen “Generalized Phase-Space Distribution Functions”, J. Math. Phys., Vol. 7, No. 5, pp. 781-786, 1966.

tfrbud

Purpose

Butterworth time-frequency distribution.

Synopsis

```
[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)
```

Description

`tfrbud` computes the Butterworth distribution of a discrete-time signal `x`, or the cross Butterworth representation between two signals. This distribution has the following expression :

$$Bud_x(t,\nu) = \int_{-\infty}^{+\infty} \frac{\sqrt{\sigma}}{2|\tau|}\, e^{-|v|\sqrt{\sigma}/|\tau|}\, x(t+v+\frac{\tau}{2})\, x^*(t+v-\frac{\tau}{2})\, e^{-j2\pi\nu\tau}\, dv\, d\tau.$$

Name	Description	Default value
`x`	signal if auto-BUD, or `[x1,x2]` if cross-BUD. `Nx=length(x)`	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`g`	time smoothing window, `G(0)` being forced to `1`, where `G(f)` is the Fourier transform of `g(t)`.	`window(odd(N/10))`
`h`	frequency smoothing window, `h(0)` being forced to `1`.	`window(odd(N/4))`
`sigma`	kernel width	`1`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency representation	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrbud` runs `tfrqview`

Example

```
sig=fmlin(128,0.05,0.3)+fmlin(128,0.15,0.4);
g=window(9,'Kaiser'); h=window(27,'Kaiser');
t=1:128; tfrbud(sig,t,128,g,h,3.6,1);
```

See Also

all the `tfr*` functions.

Reference

[1] D. Wu, J. Morris, "Time frequency representations using a radial butterworth kernel", Proc IEEE Symp TFTSA Philadelphia PA, pp. 60-63, oct. 1994.

[2] A. Papandreou, G.F. Boudreaux-Bartels, "Generalization of the Choi-Williams and the Buitterworth Distribution for Time-Frequency Analysis", IEEE Trans SP, vol 41, pp 463-472, Jan 1993.

[3] F. Auger "Reprsentations Temps-Frquence des Signaux Non-Stationnaires : Synthse et Contributions" Ph. D. Thesis, Ecole Centrale de Nantes, France, 1991.

tfrcw

Purpose

Choi-Williams time-frequency distribution.

Synopsis

```
[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,trace)
```

Description

`tfrcw` computes the Choi-Williams distribution of a discrete-time signal `x`, or the cross Choi-Williams representation between two signals. This distribution has the following expression :

$$CW_x(t,\nu) = 2\iint_{-\infty}^{+\infty} \frac{\sqrt{\sigma}}{4\sqrt{\pi}|\tau|}\, e^{-v^2\sigma/(16\tau^2)}\, x(t+v+\frac{\tau}{2})\, x^*(t+v-\frac{\tau}{2})\, e^{-j2\pi\nu\tau}\, dv\, d\tau.$$

Name	Description	Default value
`x`	signal if auto-CW, or `[x1,x2]` if cross-CW (`Nx=length(x)`)	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`g`	time smoothing window, `G(0)` being forced to `1`, where `G(f)` is the Fourier transform of `g(t)`	`window(odd(N/10))`
`h`	frequency smoothing window, `h(0)` being forced to `1`	`window(odd(N/4))`
`sigma`	kernel width	`1`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency representation	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrcw` runs `tfrqview`.

Example

```
sig=fmlin(128,0.05,0.3)+fmlin(128,0.15,0.4);
g=window(9,'Kaiser'); h=window(27,'Kaiser');
t=1:128; tfrcw(sig,t,128,g,h,3.6,1);
```

See Also

all the `tfr*` functions.

Reference

[1] H. Choi, W. Williams "Improved Time-Frequency Representation of Multicomponent Signals Using Exponential Kernels", IEEE Trans. on Acoustics, Speech and Signal Processing, Vol. 37, No. 6, June 1989.

tfrdfla

Purpose

D-Flandrin time-frequency distribution.

Synopsis

```
[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)
```

Description

`tfrdfla` generates the auto- or cross- D-Flandrin distribution. This distribution has the following expression :

$$D_x(t,\nu) = \nu \int_{-\infty}^{+\infty} (1-(\gamma/4)^2)\; X\left(\nu(1-\gamma/4)^2\right)\; X^*\left(\nu(1+\gamma/4)^2\right)\; e^{-j2\pi\gamma t\nu}\, d\gamma.$$

Name	Description	Default value
`x`	signal (in time) to be analyzed. If `x=[x1 x2]`, `tfrdfla` computes the cross-D-Flandrin distribution (`Nx=length(X)`)	
`t`	time instant(s) on which the `tfr` is evaluated	`(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	auto[a]
`trace`	if nonzero, the progression of the algorithm is shown	`0`

[a] This value, determined from `fmin` and `fmax`, is the next-power-of-two of the minimum value checking the non-overlapping condition in the fast Mellin transform.

Name	Description	Default value
tfr	time-frequency matrix containing the coefficients of the decomposition (abscissa correspond to uniformly sampled time, and ordonates 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)	

When called without output arguments, tfrdfla runs tfrqview.

Example

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

See Also

all the tfr* functions.

Reference

[1] P. Flandrin "Temps-frquence" Trait des Nouvelles Technologies, srie Traitement du Signal, Hermès, 1993.

tfrgabor

Purpose

Gabor representation of a signal.

Synopsis

```
[tfr,dgr,gam] = tfrgabor(x)
[tfr,dgr,gam] = tfrgabor(x,N)
[tfr,dgr,gam] = tfrgabor(x,N,Q)
[tfr,dgr,gam] = tfrgabor(x,N,Q,h)
[tfr,dgr,gam] = tfrgabor(x,N,Q,h,trace)
```

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. The expression of the Gabor representation is the following :

$$G_x[n,m;h] \quad = \quad \sum_k x[k]\; h^*[k-n]\; \exp\left[-j2\pi mk\right]$$

Name	Description	Default value
`x`	signal to be analyzed (`length(x)=N1`)	
`N`	number of Gabor coefficients in time (`N1` must be a multiple of `N`)	`divider(N1)`
`Q`	degree of oversampling ; must be a divider of `N`	`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 $\geq$`N`. `h` must be of unit energy, and centered	`window(odd(N), 'gauss')`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	square modulus of the Gabor coefficients	
`dgr`	Gabor coefficients (complex values)	
`gam`	biorthogonal (dual frame) window associated to `h`	

When called without output arguments, `tfrgabor` runs `tfrqview`.

If Q=1, the time-frequency plane (TFP) is critically sampled, so there is no redundancy. If Q>1, the TFP is oversampled, allowing a greater numerical stability of the algorithm.

Example

```
sig=fmlin(128);
tfrgabor(sig,64,32);
```

See Also

all the `tfr*` functions.

References

[1] Zibulski, Zeevi ”Oversampling in the Gabor Scheme” IEEE Trans. on Signal Processing, Vol. 41, No. 8, pp. 2679-87, August 1993.

[2] Wexler, Raz ”Discrete Gabor Expansions” Signal Processing, Vol. 21, No. 3, pp. 207-221, Nov 1990.

tfrgrd

Purpose

Generalized rectangular time-frequency distribution.

Synopsis

```
[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,alpha)
[tfr,t,f] = tfrgrd(x,t,N,g,h,rs,alpha,trace)
```

Description

`tfrgrd` computes the Generalized Rectangular Distribution of a discrete-time signal `x`, or the cross GRD representation between two signals. Its expression is :

$$GRD_x(t,\nu) = \iint_{-\infty}^{+\infty} \frac{2r_s}{|\tau|^\alpha} \operatorname{sinc}\left(\frac{2\pi r_s v}{|\tau|^\alpha}\right) x(t+v+\frac{\tau}{2})\, x^*(t+v-\frac{\tau}{2})\, e^{-j2\pi\nu\tau}\, dv\, d\tau$$

where r_s is a scaling factor which determines the spread of the low-pass filter, and α is the dissymetry ratio.

Name	Description	Default value
`x`	signal if auto-GRD, or `[x1,x2]` if cross-GRD (`Nx=length(x)`)	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`g`	time smoothing window, `G(0)` being forced to `1`, where `G(f)` is the Fourier transform of `g(t)`.	`window(odd(N/10))`
`h`	frequency smoothing window, `h(0)` being forced to `1`.	`window(odd(N/4))`
`rs`	kernel width	`1`
`alpha`	dissymmetry ratio	`1`
`trace`	if nonzero, the progression of the algorithm is shown	`0`

Name	Description	Default value
`tfr`	time-frequency representation	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrgrd` runs `tfrqview`.

Example

```
sig=fmlin(128,0.05,0.3)+fmlin(128,0.15,0.4);
g=window(9,'Kaiser'); h=window(27,'Kaiser');
t=1:128; tfrgrd(sig,t,128,g,h,36,1/5,1);
```

See Also

all the `tfr*` functions.

Reference

[1] F. Auger "Some Simple Parameter Determination Rules for the Generalized Choi-Williams and Butterworth Distributions" IEEE Signal processing letters, Vol 1, No 1, pp. 9-11, Jan. 1994.

tfrideal

Purpose

Ideal TF-representation for given instantaneous frequency laws.

Synopsis

```
[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)
```

Description

`tfrideal` generates the ideal time-frequency representation corresponding to the instantaneous frequency laws of the components of a signal.

Name	Description	Default value
`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 instants. The values of `iflaws` must be between 0 and 0.5	
`t`	time instant(s)	`(1:M)`
`N`	number of frequency bins	`M`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	output time-frequency matrix, of size `(N,length(t))`	
`f`	vector of normalized frequencies	

When called without output arguments, a contour plot of `tfr` is automatically displayed on the screen.

Example

```
N=140; t=0:N-1; [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];
tfrideal([if1 if2]);
```

See Also

`plotifl`, `plotsid` and all the `tfr*` functions.

tfrmh

Purpose

Margenau-Hill time-frequency distribution.

Synopsis

```
[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)
```

Description

`tfrmh` computes the Margenau-Hill distribution of a discrete-time signal `x`, or the cross Margenau-Hill representation between two signals. This distribution has the following expression :

$$\begin{aligned} MH_x(t,\nu) &= \Re\left\{x(t)\; X^*(\nu)\; e^{-j2\pi\nu t}\right\} \\ &= \int_{-\infty}^{+\infty} \frac{1}{2}\; (x(t+\tau)\; x^*(t) + x(t)\; x^*(t-\tau))\; e^{-j2\pi\nu\tau}\; d\tau. \end{aligned}$$

It corresponds to the real part of the Rihaczek distribution (see `tfrri`).

Name	Description	Default
`x`	signal if auto-MH, or `[x1,x2]` if cross-MH. `(Nx=length(x))`	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency representation	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrmh` runs `tfrqview`.

Example

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

See Also

all the `tfr*` functions.

Reference

[1] H. Margenhau, R. Hill “Correlation between Measurements in Quantum Theory”, Prog. Theor. Phys. Vol. 26, pp. 722-738, 1961.

tfrmhs

Purpose

Margenau-Hill-Spectrogram time-frequency distribution.

Synopsis

```
[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)
```

Description

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

$$MHS_x(t,\nu) = \Re\left\{K_{gh}^{-1}\, F_x(t,\nu;g)\, F_x^*(t,\nu;h)\right\}$$

$$\text{where } K_{gh} = \int h(u)\, g^*(u)\, du$$

and $F_x(t,\nu;g)$ is the short-time Fourier transform of x (analysis window g).

Name	Description	Default value
`x`	signal if auto-MHS, or `[x1,x2]` if cross-MHS (`Nx=length(x)`)	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`g, h`	analysis windows, normalized so that the representation preserves the signal energy	`window(odd(N/10))`, `window(odd(N/4))`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency representation	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrmhs` runs `tfrqview`.

Example

```
sig=fmlin(128,0.1,0.4);
g=window(21,'Kaiser');
h=window(63,'Kaiser');
tfrmhs(sig,1:128,64,g,h,1);
```

See Also

all the `tfr*` functions.

Reference

[1] R. Hippenstiel, P. De Oliviera “Time-Varying Spectral Estimation Using the Instantaneous Power Spectrum (IPS)”, IEEE Trans. on Acoust., Speech and Signal Proc. Vol. 38, No. 10, pp. 1752-1759, 1990.

tfrmmce

Purpose

Minimum mean cross-entropy combination of spectrograms.

Synopsis

```
[tfr,t,f] = tfrmmce(x)
[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)
```

Description

`tfrmmce` computes the minimum mean cross-entropy combination of spectrograms using as windows the columns of the matrix `h`. The expression of this distribution writes

$$\Pi_x(t,\nu) = \frac{E}{\|\Pi_{k=1}^N |F_x(t,\nu;h_k)|^{2/N}\|_1}\ \Pi_{k=1}^N |F_x(t,\nu;h_k)|^{2/N},$$

where $\|\ \|_1$ denotes the L_1 norm, E the energy of the signal :

$$E = \int_{-\infty}^{+\infty} |x(t)|^2\ dt = \iint_{-\infty}^{+\infty} \Pi_x(t,\nu)\ dt\ d\nu = \|\Pi_x(t,\nu)\|_1,$$

and $F_x(t,\nu;h_k)$ the short-time Fourier transform of x, with analysis window $h_k(t)$.

Name	Description	Default value
`x`	signal (`Nx=length(x)`)	
`h`	frequency smoothing windows, the `h(:,i)` being normalized so as to be of unit energy	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency representation	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrmmce` runs `tfrqview`.

Example

Here is a combination of three spectrograms with gaussian analysis windows of different lengths :

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

See Also

all the `tfr*` functions.

Reference

[1] P. Loughlin, J. Pitton, B. Hannaford "Approximating Time-Frequency Density Functions via Optimal Combinations of Spectrograms" IEEE Signal Processing Letters, Vol. 1, No. 12, Dec. 1994.

tfrpage

Purpose

Page time-frequency distribution.

Synopsis

```
[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)
```

Description

`tfrpage` computes the Page distribution of a discrete-time signal `x`, or the cross Page representation between two signals. The expression of the Page distribution is

$$\begin{aligned} P_x(t,\nu) &= \frac{d[|\int_{-\infty}^{t} x(u)\, e^{-j2\pi\nu u}\, du|^2]}{dt} \\ &= 2\,\Re\left\{x(t)\ \left(\int_{-\infty}^{t} x(u)\, e^{-j2\pi\nu u} du\right)^{*}\ e^{-j2\pi\nu t}\right\}. \end{aligned}$$

Name	Description	Default value
`x`	signal if auto-Page, or `[x1,x2]` if cross-Page (`Nx=length(x)`)	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency representation	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrpage` runs `tfrqview`.

Example

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

See Also

all the `tfr*` functions.

References

[1] C. Page “Instantaneous Power Spectra” J. Appl. Phys., Vol. 23, pp. 103-106, 1952.

[2] O. Grace “Instantaneous Power Spectra” J. Acoust. Soc. Am., Vol. 69, pp. 191-198, 1981.

tfrparam

Purpose

Return the paramaters needed to display (or save) a TF-representation.

Synopsis

```
tfrparam(method)
```

Description

`tfrparam` returns on the screen the meaning of the parameters `p1..p5` used in the files `tfrqview`, `tfrview` and `tfrsave`, to view or save a time-frequency representation.

Name	Description	Default value
`method`	chosen representation (name of the corresponding M-file)	

Example

```
tfrparam('tfrspwv');

 P1 : time      smoothing window (odd length, column vector)
 P2 : frequency smoothing window (odd length, column vector)
```

See Also

`tfrqview, tfrview, tfrsave.`

tfrpmh

Purpose

Pseudo Margenau-Hill time-frequency distribution.

Synopsis

```
[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)
```

Description

`tfrpmh` computes the Pseudo Margenau-Hill distribution of a discrete-time signal `x`, or the cross Pseudo Margenau-Hill representation between two signals. Its expression is

$$PMH_x(t,\nu) = \int_{-\infty}^{+\infty} \frac{h(\tau)}{2}\,(x(t+\tau)\,x^*(t) + x(t)\,x^*(t-\tau))\,e^{-j2\pi\nu\tau}\,d\tau.$$

Name	Description	Default value
`x`	signal if auto-PMH, or `[x1,x2]` if cross-PMH (`Nx=length(x)`)	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`h`	frequency smoothing window, `h(0)` being forced to `1`	`window(odd(N/4))`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency representation	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrpmh` runs `tfrqview`.

Example

```
sig=fmlin(128,0.1,0.4); t=1:128;
h=window(63,'Kaiser');
tfrpmh(sig,t,128,h,1);
```

See Also

all the `tfr*` functions.

References

[1] H. Margenhau, R. Hill "Correlation between Measurements in Quantum Theory", Prog. Theor. Phys. Vol. 26, pp. 722-738, 1961.

[2] R. Hippenstiel, P. De Oliviera "Time-Varying Spectral Estimation Using the Instantaneous Power Spectrum (IPS)" IEEE Trans. on Acoust., Speech and Signal Proc. Vol. 38, No. 10, pp. 1752-1759, 1990.

tfrppage

Purpose

Pseudo-Page time-frequency distribution.

Synopsis

```
[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)
```

Description

`tfrppage` computes the pseudo-Page distribution of a discrete-time signal `x`, or the cross pseudo-Page representation between two signals. The pseudo-Page distribution has the following expression :

$$PP_x(t,\nu) = 2\,\Re\left\{x(t)\ \left(\int_{-\infty}^{t}\ x(u)\ h^*(t-u)\ e^{-j2\pi\nu u} du\right)^*\ e^{-j2\pi\nu t}\right\}.$$

Name	Description	Default value
`x`	signal if auto-PPage, or `[x1,x2]` if cross-PPage (`Nx=length(x)`)	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`h`	frequency smoothing window, `h(0)` being forced to `1`	`window(odd(N/4))`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency representation	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrppage` runs `tfrqview`.

Example

```
sig=fmlin(128,0.1,0.4);
tfrppage(sig);
```

See Also

all the `tfr*` functions.

References

[1] C. Page “Instantaneous Power Spectra” J. Appl. Phys., Vol. 23, pp. 103-106, 1952.

[2] P. Flandrin, B. Escudier, W. Martin “Reprsentations Temps-Frquence et Causalit”, GRETSI-85, Juan-les-Pins (France), pp. 65-70, 1985.

tfrpwv

Purpose

Pseudo Wigner-Ville time-frequency distribution.

Synopsis

```
[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)
```

Description

`tfrpwv` computes the pseudo Wigner-Ville distribution of a discrete-time signal `x`, or the cross pseudo Wigner-Ville distribution between two signals. The pseudo Wigner-Ville distribution writes

$$PW_x(t,\nu) = \int_{-\infty}^{+\infty} h(\tau)\; x(t+\tau/2)\; x^*(t-\tau/2)\; e^{-j2\pi\nu\tau}\, d\tau.$$

Name	Description	Default value
`x`	signal if auto-PWV, or `[x1,x2]` if cross-PWV (`Nx=length(x)`)	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`h`	frequency smoothing window, in the time-domain, `h(0)` being forced to `1`	`window(odd(N/4))`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency representation	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrpwv` runs `tfrqview`.

Example

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

See Also

all the `tfr*` functions.

Reference

[1] T. Claasen, W. Mecklenbrauker "The Wigner Distribution - A Tool for Time-Frequency Signal Analysis" *3 parts* Philips J. Res., Vol. 35, No. 3, 4/5, 6, pp. 217-250, 276-300, 372-389, 1980.

tfrqview

Purpose

Quick visualization of a time-frequency representation.

Synopsis

```
tfrqview(tfr)
tfrqview(tfr,sig)
tfrqview(tfr,sig,t)
tfrqview(tfr,sig,t,method)
tfrqview(tfr,sig,t,method,p1)
tfrqview(tfr,sig,t,method,p1,p2)
tfrqview(tfr,sig,t,method,p1,p2,p3)
tfrqview(tfr,sig,t,method,p1,p2,p3,p4)
tfrqview(tfr,sig,t,method,p1,p2,p3,p4,p5)
```

Description

`tfrqview` allows a quick visualization of a time-frequency representation. `tfrqview` is called by any time-frequency representation of the toolbox (`tfr*` functions) when these functions are called without any output argument.

Name	Description	Default value
`tfr`	time-frequency representation `(MxN)`	
`sig`	signal in time. If unavailable, put `sig=[]` as input parameter	`[]`
`t`	time instants	`(1:N)`

`method`	name of chosen representation (see the `tfr*` files 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`
		`'type1'`

`p1..p5`	optional parameters of the representation : run the file `tfrparam(method)` to know the meaning of `p1..p5` for your method	

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.

Example

```
sig=fmsin(128);
tfr=tfrwv(sig);
tfrqview(tfr,sig,1:128,'tfrwv');
```

See Also

`tfrview, tfrsave, tfrparam.`

tfrrgab

Purpose

Reassigned Gabor spectrogram time-frequency distribution.

Synopsis

```
[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)
```

Description

`tfrrgab` computes the Gabor spectrogram and its reassigned version. The analysis window h used in this spectrogram is a gaussian window, which allows a 20 % faster algorithm than with the `tfrrsp` function (windows $\mathcal{T}_h$ and $\mathcal{D}_h$ defined above are colinear in this case). The reassigned Gabor spectrogram is given by the following expressions :

$$S_x^{(r)}(t',\nu';h) = \iint_{-\infty}^{+\infty} S_x(t,\nu;h)\ \delta(t'-\hat{t}(x;t,\nu))\ \delta(\nu'-\hat{\nu}(x;t,\nu))\ dt\ d\nu,$$

where

$$\hat{t}(x;t,\nu) = t - \Re\left\{\frac{F_x(t,\nu;\mathcal{T}_h)\ F_x^*(t,\nu;h)}{|F_x(t,\nu;h)|^2}\right\}$$
$$\hat{\nu}(x;t,\nu) = \nu + \Im\left\{\frac{F_x(t,\nu;\mathcal{D}_h)\ F_x^*(t,\nu;h)}{2\pi\ |F_x(t,\nu;h)|^2}\right\}$$

with $\mathcal{T}_h(t) = t\ h(t)$ and $\mathcal{D}_h(t) = \frac{dh}{dt}(t)$.

Name	Description	Default value
`x`	analyzed signal (`Nx=length(x)`)	
`t`	the time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`Nh`	length of the gaussian window	`N/4`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`k`	value at both extremities	`0.001`

Name	Description	Default value
`tfr`, `rtfr`	time-frequency representation and its reassigned version	
`hat`	complex matrix of the reassignment vectors	

When called without output arguments, `tfrrgab` runs `tfrqview`.

Example

```
sig=fmlin(128,0.1,0.4);
tfrrgab(sig,1:128,128,19,1);
```

See Also

all the `tfr*` functions.

Reference

[1] F. Auger, P. Flandrin “Improving the Readability of Time-Frequency and Time-Scale Representations by the Reassignment Method” IEEE Transactions on Signal Processing, Vol. 43, No. 5, pp. 1068-89, 1995.

tfrri

Purpose

Rihaczek time-frequency distribution.

Synopsis

```
[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)
```

Description

`tfrri` computes the Rihaczek distribution of a discrete-time signal `x`, or the cross Rihaczek representation between two signals. Its expression is

$$R_x(t,\nu) = x(t)\; X^*(\nu)\; e^{-j2\pi\nu t}.$$

Name	Description	Default value
`x`	signal if auto-Ri, or `[x1,x2]` if cross-Ri (`Nx=length(x)`)	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency representation	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrri` applies `tfrqview` on the real part of the distribution, which is equal to the Margenau-Hill distribution.

Example

```
sig=fmlin(128,0.1,0.4); tfrri(sig);
```

See Also

all the `tfr*` functions.

Reference

[1] A. Rihaczek "Signal Energy Distribution in Time and Frequency", IEEE Tans. on Info. Theory, Vol. 14, No. 3, pp. 369-374, 1968.

tfrridb

Purpose

Reduced Interference Distribution with Bessel kernel.

Synopsis

```
[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)
```

Description

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. This distribution writes

$$\begin{aligned} RIDB_x(t,\nu) &= \int_{-\infty}^{+\infty} h(\tau) R_x(t,\tau)\, e^{-j2\pi\nu\tau}\, d\tau \\ \text{with}\quad R_x(t,\tau) &= \int_{t-|\tau|}^{t+|\tau|} \frac{2\, g(v)}{\pi|\tau|} \sqrt{1-\left(\frac{v-t}{\tau}\right)^2}\, x(v+\frac{\tau}{2})\, x^*(v-\frac{\tau}{2})\, dv. \end{aligned}$$

Name	Description	Default value
`x`	signal if auto-RIDB, or `[x1,x2]` if cross-RIDB (`Nx=length(x)`)	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`g`	time smoothing window, `G(0)` being forced to `1`, where `G(f)` is the Fourier transform of `g(t)`	`window(odd(N/10))`
`h`	frequency smoothing window, `h(0)` being forced to `1`	`window(odd(N/4))`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency representation	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrridb` runs `tfrqview`.

Example

```
sig=[fmlin(128,0.05,0.3)+fmlin(128,0.15,0.4)];
g=window(31,'rect'); h=window(63,'rect');
tfrridb(sig,1:128,128,g,h,1);
```

See Also

all the `tfr*` functions.

Reference

[1] Z. Guo, L.G. Durand, H.C. Lee "The Time-Frequency Distributions of Nonstationary Signals Based on a Bessel Kernel" IEEE Trans. on Signal Proc., vol 42, pp. 1700-1707, july 1994.

tfrridbn

Purpose

Reduced Interference Distribution with a binomial kernel.

Synopsis

```
[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)
```

Description

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. This distribution has the following discrete-time continuous-frequency expression :

$$RIDBN_x(t,\nu) = \sum_{\tau=-\infty}^{+\infty} \sum_{v=-|\tau|}^{+|\tau|} \frac{1}{2^{2|\tau|+1}} \binom{2|\tau|+1}{|\tau|+v+1} x[t+v+\tau]\, x^*[t+v-\tau]\, e^{-j4\pi\nu\tau}.$$

Name	Description	Default value
`x`	signal if auto-RIDBN, or `[x1,x2]` if cross-RIDBN (`Nx=length(x)`)	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`g`	time smoothing window, `G(0)` being forced to `1`, where `G(f)` is the Fourier transform of `g(t)`	`window(odd(N/10))`
`h`	frequency smoothing window, `h(0)` being forced to `1`	`window(odd(N/4))`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency representation.	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrridbn` runs `tfrqview`.

Example

```
sig=[fmlin(128,.05,.3)+fmlin(128,.15,.4)];
tfrridbn(sig);
```

See Also

all the `tfr*` functions.

Reference

[1] W. Williams, J. Jeong "Reduced Interference Time-Frequency Distributions" in *Time-Frequency Analysis - Methods and Applications* Edited by B. Boashash, Longman-Cheshire, Melbourne, 1992.

tfrridh

Purpose

Reduced Interference Distribution with Hanning kernel.

Synopsis

```
[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)
```

Description

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. This distribution has the following expression :

$$\begin{aligned} RIDH_x(t,\nu) &= \int_{-\infty}^{+\infty} h(\tau)\, R_x(t,\tau)\, e^{-j2\pi\nu\tau}\, d\tau, \\ \text{with} \quad R_x(t,\tau) &= \int_{-\frac{|\tau|}{2}}^{+\frac{|\tau|}{2}} \frac{g(v)}{|\tau|} \left(1+\cos(\frac{2\pi v}{\tau})\right) x(t+v+\frac{\tau}{2})\, x^*(t+v-\frac{\tau}{2})\, dv. \end{aligned}$$

Name	Description	Default value
`x`	signal if auto-RIDH, or `[x1,x2]` if cross-RIDH (`Nx=length(x)`)	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`g`	time smoothing window, `G(0)` being forced to `1`, where `G(f)` is the Fourier transform of `g(t)`	`window(odd(N/10))`
`h`	frequency smoothing window, `h(0)` being forced to `1`	`window(odd(N/4))`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency representation	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrridh` runs `tfrqview`.

Example

```
sig=[fmlin(128,0.05,0.3)+fmlin(128,0.15,0.4)];
g=window(31,'rect'); h=window(63,'rect');
tfrridh(sig,1:128,128,g,h,0);
```

See Also

all the `tfr*` functions.

Reference

[1] J. Jeong, W. Williams “Kernel Design for Reduced Interference Distributions” IEEE Trans. on Signal Proc., Vol. 40, No. 2, pp. 402-412, Feb. 1992.

tfrridt

Purpose

Reduced Interference Distribution with triangular kernel.

Synopsis

```
[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)
```

Description

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 distribution between two signals. This distribution has the following expression :

$$\begin{aligned} RIDT_x(t,\nu) &= \int_{-\infty}^{+\infty} h(\tau)\, R_x(t,\tau)\, e^{-j2\pi\nu\tau}\, d\tau \\ \text{with} \quad R_x(t,\tau) &= \int_{-\frac{|\tau|}{2}}^{+\frac{|\tau|}{2}} \frac{2\, g(v)}{|\tau|}(1-\frac{2|v|}{|\tau|})\, x(t+v+\frac{\tau}{2})x^*(t+v-\frac{\tau}{2})\, dv. \end{aligned}$$

Name	Description	Default value
`x`	signal if auto-RIDT, or `[x1,x2]` if cross-RIDT (`Nx=length(x)`)	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`g`	time smoothing window, `G(0)` being forced to `1`, where `G(f)` is the Fourier transform of `g(t)`	`window(odd(N/10))`
`h`	frequency smoothing window, `h(0)` being forced to `1`	`window(odd(N/4))`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency representation	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrridt` runs `tfrqview`.

Example

```
sig=[fmlin(128,0.05,0.3)+fmlin(128,0.15,0.4)];
g=window(31,'rect'); h=window(63,'rect');
tfrridt(sig,1:128,128,g,h,0);
```

See Also

all the `tfr*` functions.

Reference

[1] J. Jeong, W. Williams “Kernel Design for Reduced Interference Distributions” IEEE Trans. on Signal Proc., Vol. 40, No. 2, pp. 402-412, Feb. 1992.

tfrrmsc

Purpose

Reassigned Morlet Scalogram time-frequency distribution.

Synopsis

```
[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)
```

Description

`tfrrmsc` computes the Morlet scalogram and its reassigned version. The reassigned Morlet scalogram has the following expression, where $h(t)$ is a gaussian window :

$$SC_x^{(r)}(t',a';h) = \iint_{-\infty}^{+\infty} a'^2 \, SC_x(t,a;h) \, \delta(t'-\hat{t}(x;t,a)) \, \delta(a'-\hat{a}(x;t,a)) \, \frac{dt \, da}{a^2},$$

where

$$\hat{t}(x;t,a) = t - \Re\left\{ a \, \frac{T_x(t,a;\mathcal{T}_h) \, T_x^*(t,a;h)}{|T_x(t,a;h)|^2} \right\}$$

$$\hat{\nu}(x;t,a) = \frac{\nu_0}{\hat{a}(x;t,a)} = \frac{\nu_0}{a} + \Im\left\{ \frac{T_x(t,a;\mathcal{D}_h) \, T_x^*(t,a;h)}{2\pi a \, |T_x(t,a;h)|^2} \right\}$$

with $\mathcal{T}_h(t) = t \; h(t)$ and $\mathcal{D}_h(t) = \frac{dh}{dt}(t)$. $SC_x(t,a;h)$ denotes the scalogram and $T_x(t,a;h)$ the wavelet transform :

$$SC_x(t,a;h) = |T_x(t,a;h)|^2 = \frac{1}{|a|} \left| \int_{-\infty}^{+\infty} x(s) \, h^*\left(\frac{s-t}{a}\right) \, ds \right|^2.$$

Name	Description	Default value
`x`	analyzed signal (`Nx=length(x)`)	
`t`	the time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`f0t`	time-bandwidth product of the mother wavelet	`2.5`
`trace`	if nonzero, the progression of the algorithm is shown	`0`

Name	Description	Default value
`tfr`, `rtfr`	time-frequency representation and its reassigned version	
`hat`	complex matrix of the reassignment vectors	

When called without output arguments, `tfrrmsc` runs `tfrqview`.

Example

```
sig=fmlin(64,0.1,0.4);
tfrrmsc(sig,1:64,64,2.1,1);
```

See Also

all the `tfr*` functions.

Reference

[1] F. Auger, P. Flandrin “Improving the Readability of Time-Frequency and Time-Scale Representations by the Reassignment Method” IEEE Transactions on Signal Processing, Vol. 43, No. 5, pp. 1068-89, 1995.

tfrrpmh

Purpose

Reassigned pseudo Margenau-Hill time-frequency distribution.

Synopsis

```
[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)
```

Description

`tfrrpmh` computes the pseudo Margenau-Hill distribution and its reassigned version. The reassigned pseudo-MHD is given by the following expression :

$$PMH_x^{(r)}(t',\nu';h) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} PMH_x(t,\nu;h)\ \delta(t'-\hat{t}(x;t,\nu))\ \delta(\nu'-\hat{\nu}(x;t,\nu))\ dt\ d\nu,$$

where

$$\hat{t}(x;t,\nu) = t \quad \text{and} \quad \hat{\nu}(x;t,\nu) = \nu + \Im\left\{\frac{F_x(t,\nu;\mathcal{D}_h)\ F_x^*(t,\nu;h)}{2\pi|F_x(t,\nu;h)|^2}\right\}.$$

$\mathcal{D}_h(t) = \frac{dh}{dt}(t)$ and $F_x(t,\nu;h)$ is the short-time Fourier transform of $x(t)$ with analysis window $h(t)$.

Name	Description	Default value
`x`	analyzed signal (`Nx=length(x)`)	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`h`	frequency smoothing window, `h(0)` being forced to `1`	`window(odd(N/4))`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr, rtfr`	time-frequency representation and its reassigned version	
`hat`	complex matrix of the reassignment vectors	

When called without output arguments, `tfrrpmh` runs `tfrqview`.

Example

```
sig=fmlin(128,0.1,0.4);
h=window(17,'Kaiser');
tfrrpmh(sig,1:128,64,h,1);
```

See Also

all the `tfr*` functions.

Reference

[1] F. Auger, P. Flandrin "Improving the Readability of Time-Frequency and Time-Scale Representations by the Reassignment Method" IEEE Transactions on Signal Processing, Vol. 43, No. 5, pp. 1068-89, 1995.

tfrrppag

Purpose

Reassigned pseudo Page time-frequency distribution.

Synopsis

```
[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)
```

Description

`tfrrppag` computes the pseudo Page distribution and its reassigned version. The reassigned pseudo Page distribution is given by the following expressions :

$$PP_x^{(r)}(t',\nu';h) = \iint_{-\infty}^{+\infty} PP_x(t,\nu;h)\ \delta(t'-\hat{t}(x;t,\nu))\ \delta(\nu'-\hat{\nu}(x;t,\nu))\ dt\ d\nu,$$

where

$$\hat{t}(x;t,\nu) = t \quad \text{and} \quad \hat{\nu}(x;t,\nu) = \nu + \Im\left\{\frac{F_x(t,\nu;\mathcal{D}_h)\ F_x^*(t,\nu;h)}{2\pi|F_x(t,\nu;h)|^2}\right\}.$$

$\mathcal{D}_h(t) = \frac{dh}{dt}(t)$ and $F_x(t,\nu;h)$ is the short-time Fourier transform of $x(t)$ with a causal analysis window $h(t)$.

Name	Description	Default value
`x`	analyzed signal (`Nx=length(x)`)	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`h`	frequency smoothing window, `h(0)` being forced to `1`	`window(odd(N/4))`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr, rtfr`	time-frequency representation and its reassigned version	
`hat`	complex matrix of the reassignment vectors	

When called without output arguments, `tfrrpmh` runs `tfrqview`.

Example

```
sig=fmlin(128,.1,.4);
h=window(65,'gauss');
tfrrppag(sig,1:128,128,h,1);
```

See Also

all the `tfr*` functions.

Reference

[1] F. Auger, P. Flandrin "Improving the Readability of Time-Frequency and Time-Scale Representations by the Reassignment Method" IEEE Transactions on Signal Processing, Vol. 43, No. 5, pp. 1068-89, 1995.

tfrrpwv

Purpose

Reassigned pseudo Wigner-Ville distribution.

Synopsis

```
[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)
```

Description

`tfrrpwv` computes the pseudo Wigner-Ville distribution and its reassigned version. These distributions are given by the following expressions :

$$\begin{aligned} PWV_x(t,\nu;h) &= \int_{-\infty}^{+\infty} h(\tau)\, x(t+\tau/2)\, x^*(t-\tau/2)\, e^{-j2\pi\nu\tau}\, d\tau \\ PWV_x^{(r)}(t',\nu';h) &= \iint_{-\infty}^{+\infty} PWV_x(t,\nu;h)\, \delta(t'-\hat{t}(x;t,\nu))\, \delta(\nu'-\hat{\nu}(x;t,\nu))\, dt\, d\nu, \end{aligned}$$

where

$$\hat{t}(x;t,\nu) = t \quad \text{and} \quad \hat{\nu}(x;t,\nu) = \nu + j\,\frac{PWV_x(t,\nu;\mathcal{D}_h)}{2\pi PWV_x(t,\nu;h)}$$

with $\mathcal{D}_h(t) = \frac{dh}{dt}(t)$.

Name	Description	Default value
`x`	analyzed signal (`Nx=length(x)`)	
`t`	the time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`h`	frequency smoothing window, `h(0)` being forced to `1`	`window(odd(N/4))`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`, `rtfr`	time-frequency representation and its reassigned version	
`hat`	complex matrix of the reassignment vectors	

When called without output arguments, `tfrrpwv` runs `tfrqview`.

Example

```
sig=fmlin(128,0.1,0.4);
h=window(17,'Kaiser');
tfrrpwv(sig,1:128,64,h,1);
```

See Also

all the `tfr*` functions.

Reference

[1] F. Auger, P. Flandrin "Improving the Readability of Time-Frequency and Time-Scale Representations by the Reassignment Method" IEEE Transactions on Signal Processing, Vol. 43, No. 5, pp. 1068-89, 1995.

tfrrsp

Purpose

Reassigned Spectrogram.

Synopsis

```
[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)
```

Description

`tfrrsp` computes the spectrogram and its reassigned version. The reassigned spectrogram is given by the following expression :

$$S_x^{(r)}(t',\nu';h) = \iint_{-\infty}^{+\infty} S_x(t,\nu;h)\;\delta(t'-\hat{t}(x;t,\nu))\;\delta(\nu'-\hat{\nu}(x;t,\nu))\;dt\;d\nu,$$

where

$$\hat{t}(x;t,\nu) = t - \Re\left\{\frac{F_x(t,\nu;\mathcal{T}_h)\;F_x^*(t,\nu;h)}{|F_x(t,\nu;h)|^2}\right\}$$
$$\hat{\nu}(x;t,\nu) = \nu + \Im\left\{\frac{F_x(t,\nu;\mathcal{D}_h)\;F_x^*(t,\nu;h)}{2\pi\;|F_x(t,\nu;h)|^2}\right\}$$

with $\mathcal{T}_h(t) = t\;h(t)$ and $\mathcal{D}_h(t) = \frac{dh}{dt}(t)$.

Name	Description	Default value
`x`	analyzed signal (`Nx=length(x)`)	
`t`	the time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`h`	frequency smoothing window, `h(0)` being forced to `1`	`window(odd(N/4))`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr, rtfr`	time-frequency representation and its reassigned version	
`hat`	complex matrix of the reassignment vectors	

When called without output arguments, `tfrrsp` runs `tfrqview`.

Example

```
sig=fmlin(128,0.1,0.4);
h=window(17,'Kaiser');
```

```
tfrrsp(sig,1:128,64,h,1);
```

See Also

all the `tfr*` functions.

Reference

[1] F. Auger, P. Flandrin "Improving the Readability of Time-Frequency and Time-Scale Representations by the Reassignment Method" IEEE Transactions on Signal Processing, Vol. 43, No. 5, pp. 1068-89, 1995.

tfrrspwv

Purpose

Reassigned smoothed pseudo Wigner-Ville distribution.

Synopsis

```
[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)
```

Description

`tfrrspwv` computes the smoothed pseudo Wigner-Ville distribution and its reassigned version. These distributions are given by the following expressions :

$$\begin{aligned} SPWV_x(t,\nu;g,h) &= \int_{-\infty}^{+\infty} h(\tau) \int_{-\infty}^{+\infty} g(s-t)\, x(s+\tau/2)\, x^*(s-\tau/2)\, ds\, e^{-j2\pi\nu\tau}\, d\tau \\ SPWV_x^{(r)}(t',\nu';g,h) &= \iint_{-\infty}^{+\infty} SPWV_x(t,\nu;g,h)\, \delta(t'-\hat{t}(x;t,\nu))\, \delta(\nu'-\hat{\nu}(x;t,\nu))\, dt\, d\nu, \end{aligned}$$

where

$$\hat{t}(x;t,\nu) = t - \frac{SPWV_x(t,\nu;\mathcal{T}_g,h)}{2\pi\, SPWV_x(t,\nu;g,h)}$$

$$\hat{\nu}(x;t,\nu) = \nu + j\frac{SPWV_x(t,\nu;g,\mathcal{D}_h)}{2\pi\, SPWV_x(t,\nu;g,h)}$$

with $\mathcal{D}_h(t) = \frac{dh}{dt}(t)$.

Name	Description	Default value
`x`	analyzed signal (`Nx=length(x)`)	
`t`	the time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`g`	time smoothing window, `G(0)` being forced to `1`, where `G(f)` is the Fourier transform of `g(t)`	`window(odd(N/10))`
`h`	frequency smoothing window, `h(0)` being forced to `1`	`window(odd(N/4))`

Name	Description	Default value
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`, `rtfr`	time-frequency representation and its reassigned version.	
`hat`	complex matrix of the reassignment vectors	

When called without output arguments, `tfrrspwv` runs `tfrqview`.

Example

```
sig=fmlin(128,0.05,0.15)+fmlin(128,0.3,0.4);
g=window(15,'Kaiser'); h=window(63,'Kaiser');
tfrrspwv(sig,1:128,64,g,h,1);
```

See Also

all the `tfr*` functions.

Reference

[1] F. Auger, P. Flandrin "Improving the Readability of Time-Frequency and Time-Scale Representations by the Reassignment Method" IEEE Transactions on Signal Processing, Vol. 43, No. 5, pp. 1068-89, 1995.

tfrsave

Purpose

Save the parameters of a time-frequency representation.

Synopsis

```
tfrsave(name,tfr,method,sig)
tfrsave(name,tfr,method,sig,t)
tfrsave(name,tfr,method,sig,t,f)
tfrsave(name,tfr,method,sig,t,f,p1)
tfrsave(name,tfr,method,sig,t,f,p1,p2)
tfrsave(name,tfr,method,sig,t,f,p1,p2,p3)
tfrsave(name,tfr,method,sig,t,f,p1,p2,p3,p4)
tfrsave(name,tfr,method,sig,t,f,p1,p2,p3,p4,p5)
```

Description

`tfrsave` saves the parameters of a time-frequency representation in the file `name.mat`. Two additional parameters are saved : `TfrQView` and `TfrView`. If you load the file `name.mat` and do `eval(TfrQView)`, you will restart the display session under `tfrqview` ; if you do `eval(TfrView)`, you will display the representation by means of `tfrview`.

Name	Description	Default value
`name`	name of the mat-file (less than 8 characters)	
`tfr`	time-frequency representation `(M,N)`	
`method`	chosen representation	
`sig`	signal from which the `tfr` was obtained	
`t`	time instant(s)	`(1:N)`
`f`	frequency bins	`0.5*(0:M-1)/M`
`p1..p5`	optional parameters : run `tfrparam(method)` to know the meaning of `p1..p5` for your method	

Example

```
sig=fmlin(64); tfr=tfrwv(sig);
tfrsave('wigner',tfr,'TFRWV',sig,1:64);
clear; load wigner; eval(TfrQView);
```

See Also

`tfrqview, tfrview, tfrparam.`

tfrscalo

Purpose

Scalogram, for Morlet or Mexican hat wavelet.

Synopsis

```
[tfr,t,f,wt] = tfrscalo(x)
[tfr,t,f,wt] = tfrscalo(x,t)
[tfr,t,f,wt] = tfrscalo(x,t,wave)
[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)
```

Description

`tfrscalo` computes the scalogram (squared magnitude of a continuous wavelet transform). Its expression is the following :

$$SC_x(t,a;h) = |T_x(t,a;h)|^2 = \frac{1}{|a|}\left|\int_{-\infty}^{+\infty} x(s)\; h^*\left(\frac{s-t}{a}\right)\; ds\right|^2 .$$

This time-scale expression has an equivalent time-frequecy expression, obtained using the formal identification $a = \frac{\nu_0}{\nu}$, where ν_0 is the central frequency of the mother wavelet $h(t)$.

Name	Description	Default value
`x`	signal to be analyzed (`Nx=length(x)`). Its analytic version is used (`z=hilbert(real(x))`)	
`t`	time instant(s) on which the `tfr` is evaluated	`(1:Nx)`
`wave`	half length of the Morlet analyzing wavelet at coarsest scale. If `wave=0`, the Mexican hat is used	`sqrt(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 $\leq$`0.5`	
`N`	number of analyzed voices	auto[a]

[a] This value, determined from `fmin` and `fmax`, is the next-power-of-two of the minimum value checking the non-overlapping condition in the fast Mellin transform.

Name	D
`trace`	if
`tfr`	ti d p s tl
`f`	v p
`wt`	C tr o

When calle

Example

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

See Also

all the `tfr*` functions.

Reference

[1] O. Rioul, P. Flandrin “Time-Scale Distributions : A General Class Extending Wavelet Transforms”, IEEE Transactions on Signal Processing, Vol. 40, No. 7, pp. 1746-57, July 1992.

tfrsp

Purpose

Spectrogram time-frequency distribution.

Synopsis

```
[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)
```

Description

`tfrsp` computes the spectrogram distribution of a discrete-time signal `x`. It corresponds to the squared modulus of the short-time Fourier transform. Its expression writes

$$S_x(t,\nu) = \left| \int_{-\infty}^{+\infty} x(u)\; h^*(u-t)\; e^{-j2\pi\nu u}\; du \right|^2.$$

Name	Description	Default value
`x`	analyzed signal (`Nx=length(x)`)	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`h`	smoothing window, `h` being normalized so as to be of unit energy.	`window(odd(N/4))`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency representation	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrsp` runs `tfrqview`.

Example

```
sig=fmlin(128,0.1,0.4);
h=window(17,'Kaiser');
tfrsp(sig,1:128,64,h,1);
```

See Also

all the `tfr*` functions.

References

[1] W. Koenig, H. Dunn, L. Lacy "The sound spectrograph", J. Acoust. Soc. Am., Vol. 18, No. 1, pp. 19-49, 1946.

[2] J. Allen, L. Rabiner "A Unified Approach to Short-Time Fourier Analysis and Synthesis" Proc. IEEE, Vol. 65, No. 11, pp. 1558-64, 1977.

tfrspaw

Purpose

Smoothed pseudo affine Wigner time-frequency distributions.

Synopsis

```
[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)
```

Description

`tfrspaw` generates the auto- or cross- smoothed pseudo affine Wigner distributions. Its general expression writes

$$\tilde{P}_x^k(t,\nu) = \int_{-\infty}^{+\infty} \frac{\mu_k(u)}{\sqrt{\lambda_k(u)\lambda_k(-u)}}\, T_x(t,\lambda_k(u)\nu;\psi)\, T_x^*(t,\lambda_k(-u)\nu;\psi)\, du,$$

where $T_x(t,\nu;\psi)$ is the continuous wavelet transform,

$$\psi(t) = (\pi t_0^2)^{-1/4} \exp\left[-\frac{1}{2}(t/t_0)^2 + j2\pi\nu_0 t\right]$$

is the Morlet wavelet, and $\lambda_k(u,k) = \left(\frac{k(e^{-u}-1)}{e^{-ku}-1}\right)^{\frac{1}{k-1}}$.

Name	Description	Default
`x`	signal (in time) to be analyzed. If `x=[x1 x2]`, `tfrspaw` computes the cross-smoothed pseudo affine Wigner distribution. `(Nx=length(X))`	
`t`	time instant(s) on which the `tfr` is evaluated	`(1:Nx)`
`k`	label of the distribution `k=-1 :` smoothed pseudo active Unterberger `k=0 :` smoothed pseudo Bertrand `k=1/2 :` smoothed pseudo D-Flandrin `k=2 :` affine smoothed pseudo Wigner-Ville	0

Name	D
`nh0`	h A s ti
`ng0`	h s
`fmin,` `fmax`	r a q y o
`N`	n
`trace`	i
`tfr`	ti d ti f f
`f`	v f

When calle

[a]This value

the non-overla

Example

```
sig=altes(64,0.1,0.45);
tfrspaw(sig);
```

See Also

all the `tfr*` functions.

Reference

[1] P. Gonalvs, R. Baraniuk “Pseudo Affine Wigner Distributions and Kernel Formulation” Submitted to IEEE Transactions on Signal Processing, 1996.

tfrspwv

Purpose

Smoothed pseudo Wigner-Ville time-frequency distribution.

Synopsis

```
[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)
```

Description

`tfrspwv` computes the smoothed pseudo Wigner-Ville distribution of a discrete-time signal `x`, or the cross smoothed pseudo Wigner-Ville distribution between two signals. Its expression writes

$$SPW_x(t,\nu) = \int_{-\infty}^{+\infty} h(\tau)\ \int_{-\infty}^{+\infty} g(s-t)\ x(s+\tau/2)\ x^*(s-\tau/2)\ ds\ e^{-j2\pi\nu\tau}\ d\tau.$$

Name	Description	Default value
`x`	signal if auto-SPWV, or `[x1,x2]` if cross-SPWV (`Nx=length(x)`)	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`g`	time smoothing window, `G(0)` being forced to `1`, where `G(f)` is the Fourier transform of `g(t)`	`window(odd(N/10))`
`h`	frequency smoothing window in the time-domain, `h(0)` being forced to `1`	`window(odd(N/4))`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency representation	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrspwv` runs `tfrqview`.

Example

```
sig=fmlin(128,0.05,0.15)+fmlin(128,0.3,0.4);
g=window(15,'Kaiser'); h=window(63,'Kaiser');
tfrspwv(sig,1:128,64,g,h,1);
```

See Also

all the `tfr*` functions.

References

[1] P. Flandrin "Some Features of Time-Frequency Representations of Multi-Component Signals" IEEE Int. Conf. on Acoust. Speech and Signal Proc., pp. 41.B.4.1-41.B.4.4, San Diego (CA), 1984.

[2] T. Claasen, W. Mecklenbrauker "The Wigner Distribution - A Tool for Time-Frequency Signal Analysis" *3 parts* Philips J. Res., Vol. 35, No. 3, 4/5, 6, pp. 217-250, 276-300, 372-389, 1980.

tfrstft

Purpose

Short time Fourier transform.

Synopsis

```
[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)
```

Description

`tfrstft` computes the short-time Fourier transform of a discrete-time signal `x`. Its continuous expression writes

$$F_x(t,\nu;h) = \int_{-\infty}^{+\infty} x(u)\, h^*(u-t)\, e^{-j2\pi\nu u}\, du$$

where $h(t)$ is a *short time analysis window* localized around $t = 0$ and $\nu = 0$.

Name	Description	Default value
`x`	signal (`Nx=length(x)`)	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`h`	smoothing window, `h` being normalized so as to be of unit energy.	`window(odd(N/4))`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency decomposition (complex values). The frequency axis is graduated from `-0.5` to `0.5`	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrstft` runs `tfrqview`, which displays the squared modulus of the short-time Fourier transform.

Example

```
sig=[fmlin(128,0.05,.45);fmlin(128,0.35,.15)];
tfr=tfrstft(sig);
subplot(211); imagesc(abs(tfr(1:128,:))); axis('xy')
subplot(212); imagesc(angle(tfr(1:128,:))); axis('xy')
```

See Also

all the `tfr*` functions.

References

[1] J. Allen, L. Rabiner "A Unified Approach to Short-Time Fourier Analysis and Synthesis" Proc. of the IEEE, Vol. 65, No. 11, pp. 1558-64, Nov. 1977.

[2] S. Nawab, T. Quatieri "Short-Time Fourier Transform", chapter in *Advanced Topics in Signal Processing* J. Lim and A. Oppenheim eds. Prentice Hall, Englewood Cliffs, NJ, 1988.

tfrunter

Purpose

Unterberger time-frequency distribution, active or passive form.

Synopsis

```
[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)
```

Description

`tfrunter` generates the auto- or cross-Unterberger distribution (active or passive form). The expression of the active Unterberger distribution writes

$$U_x^{(a)}(t,a) = \frac{1}{|a|} \int_0^{+\infty} (1+\frac{1}{\alpha^2})\, X\left(\frac{\alpha}{a}\right)\; X^*\left(\frac{1}{\alpha a}\right)\; e^{j2\pi(\alpha-1/\alpha)\frac{t}{a}}\, d\alpha,$$

whereas the passive Unterberger distribution writes

$$U_x^{(p)}(t,a) = \frac{1}{|a|} \int_0^{+\infty} \frac{2}{\alpha}\, X\left(\frac{\alpha}{a}\right)\; X^*\left(\frac{1}{\alpha a}\right)\; e^{j2\pi(\alpha-\frac{1}{\alpha})\frac{t}{a}}\, d\alpha.$$

Name	Description	Default value
`x`	signal (in time) to be analyzed. If `x=[x1 x2]`, `tfrunter` computes the cross-Unterberger distribution `(Nx=length(x))`	
`t`	time instant(s) on which the `tfr` is evaluated	`(1:Nx)`
`form`	`'A'` for active, `'P'` for passive Unterberger distribution	`'A'`
`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 $\leq$`0.5`	
`N`	number of analyzed voices	auto[a]

[a] This value, determined from `fmin` and `fmax`, is the next-power-of-two of the minimum value checking the non-overlapping condition in the fast Mellin transform.

Name	D
`trace`	if
`tfr`	ti d p s th
`f`	v p

When calle

Example

```
sig=altes(64,0.1,0.45);
tfrunter(sig);
```

See Also

all the `tfr*` functions.

References

[1] A. Unterberger "The Calculus of Pseudo-Differential Operators of Fuchs Type" Comm. in Part. Diff. Eq., Vol. 9, pp. 1179-1236, 1984.

[2] P. Flandrin, P. Gonalvs "Geometry of Affine Time-Frequency Distributions" Applied and Computational Harmonic Analysis, Vol. 3, pp. 10-39, January 1996.

tfrview

Purpose

Visualization of time-frequency representations.

Synopsis

```
tfrview(tfr,sig,t,method,param,map)
tfrview(tfr,sig,t,method,param,map,p1)
tfrview(tfr,sig,t,method,param,map,p1,p2)
tfrview(tfr,sig,t,method,param,map,p1,p2,p3)
tfrview(tfr,sig,t,method,param,map,p1,p2,p3,p4)
tfrview(tfr,sig,t,method,param,map,p1,p2,p3,p4,p5)
```

Description

Name	D
	-
	s
	n
	-
	-
	i
	w
	a
`map`	s
`p1..p5`	p
	t
	p

`tfrview` visualizes a time-frequency representation. It is called through `tfrqview` from any `tfr*` function when this function is called without output argument. **Use `tfrqview` preferably**.

Name	Description
`tfr`	time-frequency representation
`sig`	signal in the time-domain
`t`	time instants
`method`	chosen representation (name of the corresponding M-file)
`param`	visualization parameter vector : `param = [display linlog threshold levnumb nf2` ... `layout access state fs isgrid]` where - `display=1..5` for `contour`, `imagesc`, `pcolor`, `surf` or `mesh` - `linlog=0/1` for linearly/logarithmically spaced levels for the amplitude of `tfr` - `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/remove the colorbar (5) - `access` depends on the way you access to `tfrview`: from the command line (0) ; from `tfrqview`, except after a change in the sampling frequency or in the layout (1) ; from `tfrqview`, after a change in the layout (2) ; from `tfrqview`, after a change in the sampling frequency (3)

See Also

`tfrqview`, `tfrparam`, `tfrsave`.

tfrwv

Purpose

Wigner-Ville time-frequency distribution.

Synopsis

```
[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)
```

Description

`tfrwv` computes the Wigner-Ville distribution of a discrete-time signal `x`, or the cross Wigner-Ville representation between two signals. The continuous expression of the Wigner-Ville distribution writes

$$W_x(t,\nu) = \int_{-\infty}^{+\infty} x(t+\tau/2)\; x^*(t-\tau/2)\; e^{-j2\pi\nu\tau}\; d\tau,$$

Name	Description	Default value
`x`	signal if auto-WV, or `[x1,x2]` if cross-WV (`Nx=length(x)`)	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency representation.	
`f`	vector of normalized frequencies	

When called without output arguments, `tfrwv` runs `tfrqview`.

Example

The Wigner-Ville distribution is perfectly localized on linear chirp signals. Here is what we obtain in the discrete case :

```
sig=fmlin(128,0.1,0.4);
tfrwv(sig);
```

See Also

all the `tfr*` functions.

References

[1] E. Wigner “On the Quantum Correction for Thermodynamic Equilibrium” Phys. Res., Vol. 40, pp. 749-759, 1932.

[2] J. Ville “Thorie et Application de la Notion de Signal Analytique” Cbles et Transmission, 2eme A. , No. 1, pp. 61-74, 1948.

[3] T. Claasen, W. Mecklenbrauker “The Wigner Distribution - A Tool for Time-Frequency Signal Analysis” *3 parts* Philips J. Res., Vol. 35, No. 3, 4/5, 6, pp. 217-250, 276-300, 372-389, 1980.

tfrzam

Purpose

Zhao-Atlas-Marks time-frequency distribution.

Synopsis

```
[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)
```

Description

`tfrzam` computes the Zhao-Atlas-Marks distribution of a discrete-time signal `x`, or the cross Zhao-Atlas-Marks representation between two signals. This distribution writes

$$ZAM_x(t,\nu) = \int_{-\infty}^{+\infty} \left[h(\tau) \int_{t-|\tau|/2}^{t+|\tau|/2} x(s+\tau/2)\, x^*(s-\tau/2)\, ds \right] e^{-j2\pi\nu\tau}\, d\tau.$$

It is also known as the *Cone-Shaped Kernel distribution*.

Name	Description	Default value
`x`	signal if auto-ZAM, or `[x1,x2]` if cross-ZAM (`Nx=length(x)`)	
`t`	time instant(s)	`(1:Nx)`
`N`	number of frequency bins	`Nx`
`g`	time smoothing window, `G(0)` being forced to `1`, where `G(f)` is the Fourier transform of `g(t)`	`window(odd(N/10))`
`h`	frequency smoothing window, `h(0)` being forced to `1`	`window(odd(N/4))`
`trace`	if nonzero, the progression of the algorithm is shown	`0`
`tfr`	time-frequency representation	
`f`	vector of normalized frequencies	

When called without outpout arguments, `tfrzam` runs `tfrqview`.

Example

```
sig=fmlin(128,0.05,0.3)+fmlin(128,0.15,0.4);
g=window(9,'Kaiser'); h=window(27,'Kaiser');
tfrzam(sig,1:128,128,g,h,1);
```

See Also

all the `tfr*` functions.

Reference

[1] Y. Zhao, L. Atlas, R. Marks "The Use of the Cone-Shaped Kernels for Generalized Time-Frequency Representations of Nonstationary Signals" IEEE Trans. on Acoust., Speech and Signal Proc., Vol. 38, No. 7, pp. 1084-91, 1990.

tftb_window

Purpose

Window generation.

Synopsis

```
h = tftb\_window(N)
h = tftb\_window(N,name)
h = tftb\_window(N,name,param)
h = tftb\_window(N,name,param,param2)
```

Description

`tftb_window` yields a window of length `N` with a given shape.

Name	Description	Default value
`N`	length of the window	
`name`	name of the window shape	`'Hamming'`
`param`	optional parameter	
`param2`	second optional parameter	
`h`	output window	

Possible names are :
`'Hamming'`, `'Hanning'`, `'Nuttall'`, `'Papoulis'`, `'Harris'`, `'Rect'`, `'Triang'`, `'Bartlett'`, `'BartHann'`, `'Blackman'`, `'Gauss'`, `'Parzen'`, `'Kaiser'`, `'Dolph'`, `'Hanna'`, `'Nutbess'`, `'spline'`

For the gaussian window, an optional parameter `k` sets the value at both extremities. The default value is `0.005`.

For the Kaiser-Bessel window, an optional 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`.

Example

```
h=tftb\_window(256,'Gauss',0.005);
plot(h);
```

See Also

`dwindow.`

Reference

[1] F. Harris “On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform”, Proceedings of the IEEE, Vol. 66, pp. 51-83, 1978.

[2] A.H. Nuttal, ”A Two-Parameter Class of Bessel Weighting Functions for Spectral Analysis or Array Processing”, IEEE Trans on ASSP, Vol 31, pp 1309-1311, Oct 1983.

[3] Y. Ho Ha, J.A. Pearce, ”A New Window and Comparison to Standard Windows”, Trans IEEE ASSP, Vol 37, No 2, pp 298-300, February 1989.

[4] C.S. Burrus, “Multiband Least Squares FIR Filter Design”, Trans IEEE SP, Vol 43, No 2, pp 412-421, February 1995.

zak

Purpose

Zak transform.

Synopsis

```
dzt = zak(sig)
dzt = zak(sig,N)
dzt = zak(sig,N,M)
```

Description

`zak` computes the Zak transform of a signal. Its definition is given by

$$Z_{sig}(t,\nu) = \sum_{n=-\infty}^{+\infty} sig(t+n)\, e^{-j2\pi n\nu}.$$

Name	Description	Default value
`sig`	Signal to be analyzed `(length(sig)=N1)`	
`N`	number of Zak coefficients in time (`N1` must be a multiple of `N`)	`divider(N1)`
`M`	number of Zak coefficients in frequency (`N1` must be a multiple of `M`)	`N1/N`
`dzt`	Output matrix `(N,M)` containing the discrete Zak transform	

Example

```
sig=fmlin(256);
DZT=zak(sig);
imagesc(DZT);
```

See Also

`izak, tfrgabor.`

Reference

[1] L. Auslander, I. Gertner, R. Tolimieri, “The Discrete Zak Transform Application to Time-Frequency Analysis and Synthesis of Nonstationary Signals” IEEE Trans. on Signal Proc., Vol. 39, No. 4, pp. 825-835, April 1991.

GNU Free Documentation License

Version 1.2, November 2002
Copyright ©2000,2001,2002 Free Software Foundation, Inc.

51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document ”free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of ”copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **”Document”**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **”you”**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **”Modified Version”** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **”Secondary Section”** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **”Invariant Sections”** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled ”History” in the various original documents, forming one section Entitled ”History”; likewise combine any sections Entitled ”Acknowledgements”, and any sections Entitled ”Dedications”. You must delete all sections Entitled ”Endorsements”.

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an ”aggregate” if the copyright resulting from the compilation is not used to limit the legal rights of the compilation’s users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document’s Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled ”Acknowledgements”, ”Dedications”, or ”History”, the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License ”or any later version” applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled ”GNU Free Documentation License”.

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the ”with...Texts.” line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.