pysptk Documentation

Release 0.1.11+2096bb4

Ryuichi YAMAMOTO

Jan 14, 2018

Contents

A python wrapper for Speech Signal Processing Toolkit (SPTK).

https://github.com/r9y9/pysptk

The wrapper is based on a modified version of SPTK (r9y9/SPTK)

CHAPTER 1

Full documentation

A full documentation of SPTK is available at http://sp-tk.sourceforge.net. If you are not familiar with SPTK, I recommend you to take a look at the doc first before using `pysptk`.

CHAPTER 2

Demonstration notebooks

- Introduction notebook: a brief introduction to pysptk
- Speech analysis and re-synthesis resynthesis notebook: a demonstration notebook for speech analysis and re-synthesis. Synthesized audio examples(English) are available on the notebook.

CHAPTER 3

Installation guide

3.1 Installation guide

The latest release is availabe on pypi. Assuming you have already `numpy` installed, you can install pysptk by:

```
pip install pysptk
```

If yout want the latest development version, run:

```
pip install git+https://github.com/r9y9/pysptk
```

or:

```
git clone https://github.com/r9y9/pysptk
cd pysptk
python setup.py develop # or install
```

This should resolve the package dependencies and install `pysptk` property.

Note: If you use the development version, you need to have `cython` (and C compiler) installed to compile cython module(s).

3.1.1 For Windows users

There are some binary wheels available on pypi, so you can install `pysptk` via pip **without cython and C compilier** if there exists a binary wheel that matches your environment (depends on bits of system and python version). For now, wheels are available for:

- Python 2.7 on 32 bit system
- Python 2.7 on 64 bit system

- Python 3.4 on 32 bit system

If there is no binary wheel available for your environment, you can build `pysptk` from the source distribution, which is also available on pypi. Note that in order to compile `pysptk` from source in Windows, it is highly recommended to use Anaconda , since installation of numpy, cython and other scientific packages is really easy. In fact, continuous integration in Windows on AppVeyor uses Anacona to build and test `pysptk`. See pysptk/appveyor.yml for the exact build steps.

CHAPTER 4

API documentation

4.1 API

4.1.1 Core SPTK API

All functionality in `pysptk.sptk` (the core API) is directly accesible from the top-level `pysptk.*` namespace.

For convenience, vector-to-vector functions (`pysptk.mcep`, `pysptk.mc2b`, etc) that takes an input vector as the first argment, can also accept matrix. As for matrix inputs, vector-to-vector functions are applied along with the last axis internally; e.g.

```
mc = pysptk.mcep(frames) # frames.shape == (num_frames, frame_len)
```

is equivalent to:

```
mc = np.apply_along_axis(pysptk.mcep, -1, frames)
```

Warning: The core APIs in `pysptk.sptk` package are based on the SPTK's internal APIs (e.g. code in `_mgc2sp.c`), so the functionalities are not exactly same as SPTK's CLI. If you find any inconsistency that should be addressed, please file an issue.

Note: Almost all of pysptk functions assume that the input array is **C-contiguous** and has `float64` element type. For vector-to-vector functions, the input array is automatically converted to `float64`-typed one, the function is executed on it, and then the output array is converted to have the same type with the input you provided.

Library routines

agexp(r, x, y)	Magnitude squared generalized exponential function
gexp(r, x)	Generalized exponential function
glog(r, x)	Generalized logarithmic function
mseq()	M-sequence

pysptk.sptk.agexp

pysptk.sptk.**agexp**(*r*, *x*, *y*)

Magnitude squared generalized exponential function

Parameters **r** : float

Gamma

x : float

Real part

y : float

Imaginary part

Returns Value

pysptk.sptk.gexp

pysptk.sptk.**gexp**(*r*, *x*)

Generalized exponential function

Parameters **r** : float

Gamma

x : float

Arg

Returns Value

pysptk.sptk.glog

pysptk.sptk.**glog**(*r*, *x*)

Generalized logarithmic function

Parameters **r** : float

Gamma

x : float

Arg

Returns Value

pysptk.sptk.mseq

pysptk.sptk.**mseq**()
M-sequence

Returns A sample of m-sequence

Adaptive cepstrum analysis

acep(x, c[, lambda_coef, step, tau, pd, eps])	Adaptive cepstral analysis
agcep(x, c[, stage, lambda_coef, step, tau, eps])	Adaptive generalized cepstral analysis
amcep(x, b[, alpha, lambda_coef, step, tau, ...])	Adaptive mel-cepstral analysis

pysptk.sptk.acep

pysptk.sptk.**acep**(*x*, *c*, *lambda_coef=0.98*, *step=0.1*, *tau=0.9*, *pd=4*, *eps=1e-06*)
Adaptive cepstral analysis

Parameters **x** : double

A input sample

c : array, shape(`order + 1`)

Cepstrum. The result is stored in place.

lambda_coef : float, optional

Leakage factor. Default is 0.98.

step : float, optional

Step size. Default is 0.1.

tau : float, optional

Momentum constant. Default is 0.9.

pd : int, optional

Order of pade approximation. Default is 4.

eps : float, optional

Minimum value for epsilon. Default is 1.0e-6.

Returns **prederr** : float

Prediction error

Raises **ValueError**

if invalid order of pade approximation is specified

See also:

pysptk.sptk.uels, *pysptk.sptk.gcep*, *pysptk.sptk.mcep*, *pysptk.sptk.mgcep*, *pysptk.sptk.amcep*, *pysptk.sptk.agcep*, *pysptk.sptk.lmadf*

pysptk.sptk.agcep

pysptk.sptk.**agcep**(*x*, *c*, *stage=1*, *lambda_coef=0.98*, *step=0.1*, *tau=0.9*, *eps=1e-06*)

Adaptive generalized cepstral analysis

Parameters **x** : float

A input sample

c : array, shape(`order + 1`), optional

Cepstrum. The result is stored in-place.

stage : int, optional

-1 / gamma. Default is 1.

lambda_coef : float, optional

Leakage factor. Default is 0.98.

step : float, optional

Step size. Default is 0.1.

tau : float, optional

Momentum constant. Default is 0.9.

eps : float, optional

Minimum value for epsilon. Default is 1.0e-6.

Returns **prederr** : float

Prediction error

Raises **ValueError**

if invalid number of stage is specified

See also:

pysptk.sptk.acep, *pysptk.sptk.amcep*, *pysptk.sptk.glsadf*

pysptk.sptk.amcep

pysptk.sptk.**amcep**(*x*, *b*, *alpha=0.35*, *lambda_coef=0.98*, *step=0.1*, *tau=0.9*, *pd=4*, *eps=1e-06*)

Adaptive mel-cepstral analysis

Parameters **x** : float

A input sample

b : array, shape(`order + 1`), optional

MLSA filter coefficients. The result is stored in-place.

alpha : float, optional

All-pass constant. Default is 0.35.

lambda_coef : float, optional

Leakage factor. Default is 0.98.

step : float, optional

Step size. Default is 0.1.

tau : float, optional

Momentum constant. Default is 0.9.

pd : int, optional

Order of pade approximation. Default is 4.

eps : float, optional

Minimum value for epsilon. Default is 1.0e-6.

Returns **prederr** : float

Prediction error

Raises **ValueError**

if invalid order of pade approximation is specified

See also:

pysptk.sptk.acep, *pysptk.sptk.agcep*, *pysptk.sptk.mc2b*, *pysptk.sptk.b2mc*, *pysptk.sptk.mlsadf*

Mel-generalized cepstrum analysis

mcep(windowed[, order, alpha, miniter, ...])	Mel-cepstrum analysis
gcep(windowed[, order, gamma, miniter, ...])	Generalized-cepstrum analysis
mgcep(windowed[, order, alpha, gamma, ...])	Mel-generalized cepstrum analysis
uels(windowed[, order, miniter, maxiter, ...])	Unbiased estimation of log spectrum
fftcep(logsp[, order, num_iter, ...])	FFT-based cepstrum analysis
lpc(windowed[, order, min_det])	Linear prediction analysis

pysptk.sptk.mcep

pysptk.sptk.**mcep**(*windowed*, *order=25*, *alpha=0.35*, *miniter=2*, *maxiter=30*, *threshold=0.001*, *etype=0*, *eps=0.0*, *min_det=1e-06*, *itype=0*)

Mel-cepstrum analysis

Parameters **windowed** : array, shape (`frame_len`)

A windowed frame

order : int, optional

Order of mel-cepstrum. Default is 25.

alpha : float, optional

All pass constant. Default is 0.35.

miniter : int, optional

Minimum number of iteration. Default is 2.

maxiter : int, optional

Maximum number of iteration. Default is 30.

threshold : float, optional

Threshold in theq. Default is 0.001.

etype : int, optional

Type of parameter `eps`

0. not used
1. initial value of log-periodogram
2. floor of periodogram in db

Default is 0.

eps : float, optional

Initial value for log-periodogram or floor of periodogram in db. Default is 0.0.

min_det : float, optional

Mimimum value of the determinant of normal matrix. Default is 1.0e-6

itype : float, optional

Input data type

0. windowed signal
1. log amplitude in db
2. log amplitude
3. amplitude
4. periodogram

Default is 0.

Returns **mc** : array, shape (`order + 1`)

Mel-cepstrum

Raises **ValueError**

- if invalid `itype` is specified
- if invalid `etype` is specified
- if nonzero `eps` is specified when etype = 0
- if negative `eps` is specified
- if negative `min_det` is specified

RuntimeError

- if zero(s) are found in periodogram
- if error happened in theq

See also:

pysptk.sptk.uels, *pysptk.sptk.gcep*, *pysptk.sptk.mgcep*, *pysptk.sptk.mlsadf*

pysptk.sptk.gcep

pysptk.sptk.**gcep**(*windowed*, *order=25*, *gamma=0.0*, *miniter=2*, *maxiter=30*, *threshold=0.001*, *etype=0*, *eps=0.0*, *min_det=1e-06*, *itype=0*, *norm=False*)

Generalized-cepstrum analysis

Parameters **windowed** : array, shape (`frame_len`)

A windowed frame

order : int, optional

Order of generalized-cepstrum. Default is 25.

gamma : float, optional

Parameter of generalized log function. Default is 0.0.

miniter : int, optional

Minimum number of iteration. Default is 2.

maxiter : int, optional

Maximum number of iteration. Default is 30.

threshold : float, optional

Threshold in theq. Default is 0.001

etype : int, optional

Type of parameter `eps`

0. not used
1. initial value of log-periodogram
2. floor of periodogram in db

Default is 0.

eps : float, optional

Initial value for log-periodogram or floor of periodogram in db. Default is 0.0.

min_det : float, optional

Mimimum value of the determinant of normal matrix. Default is 1.0e-6.

itype : float, optional

Input data type

0. windowed signal
1. log amplitude in db
2. log amplitude
3. amplitude
4. periodogram

Default is 0.

Returns **gc** : array, shape (`order + 1`)

Generalized cepstrum

Raises **ValueError**

- if invalid `itype` is specified
- if invalid `etype` is specified
- if nonzero `eps` is specified when etype = 0
- if negative `eps` is specified
- if negative `min_det` is specified

RuntimeError

- if error happened in theq

See also:

pysptk.sptk.uels, *pysptk.sptk.mcep*, *pysptk.sptk.mgcep*, *pysptk.sptk.glsadf*

pysptk.sptk.mgcep

pysptk.sptk.**mgcep**(*windowed*, *order=25*, *alpha=0.35*, *gamma=0.0*, *num_recursions=None*, *miniter=2*, *maxiter=30*, *threshold=0.001*, *etype=0*, *eps=0.0*, *min_det=1e-06*, *itype=0*, *otype=0*)

Mel-generalized cepstrum analysis

Parameters **windowed** : array, shape (`frame_len`)

A windowed frame

order : int, optional

Order of mel-generalized cepstrum. Default is 25.

alpha : float, optional

All pass constant. Default is 0.35.

gamma : float, optional

Parameter of generalized log function. Default is 0.0.

num_recursions : int, optional

Number of recursions. Default is `len(windowed) - 1`.

miniter : int, optional

Minimum number of iteration. Default is 2.

maxiter : int, optional

Maximum number of iteration. Default is 30.

threshold : float, optional

Threshold. Default is 0.001.

etype : int, optional

Type of paramter `e`

0. not used
1. initial value of log-periodogram
2. floor of periodogram in db

Default is 0.

eps : float, optional

Initial value for log-periodogram or floor of periodogram in db. Default is 0.0.

min_det : float, optional

Mimimum value of the determinant of normal matrix. Default is 1.0e-6.

itype : float, optional

Input data type

0. windowed signal
1. log amplitude in db
2. log amplitude
3. amplitude
4. periodogram

Default is 0.

otype : int, optional

Output data type

0. mel generalized cepstrum: (c~0...c~m)
1. MGLSA filter coefficients: b0...bm
2. K~,c~‘1...c~’m
3. K,b‘1...b’m
4. K~,g*c~‘1...g*c~’m
5. K,g*b‘1...g*b’m

Default is 0.

Returns **mgc** : array, shape (`order + 1`)

mel-generalized cepstrum

Raises **ValueError**

- if invalid `itype` is specified
- if invalid `etype` is specified
- if nonzero `eps` is specified when etype = 0
- if negative `eps` is specified
- if negative `min_det` is specified
- if invalid `otype` is specified

RuntimeError

- if error happened in theq

See also:

pysptk.sptk.uels, *pysptk.sptk.gcep*, *pysptk.sptk.mcep*, *pysptk.sptk.freqt*, *pysptk.sptk.gc2gc*, *pysptk.sptk.mgc2mgc*, *pysptk.sptk.gnorm*, *pysptk.sptk.mglsadf*

pysptk.sptk.uels

pysptk.sptk.**uels**(*windowed*, *order=25*, *miniter=2*, *maxiter=30*, *threshold=0.001*, *etype=0*, *eps=0.0*, *itype=0*)

Unbiased estimation of log spectrum

Parameters **windowed** : array, shape (`frame_len`)

A windowed frame

order : int, optional

Order of cepstrum. Default is 25.

miniter : int, optional

Minimum number of iteration. Default is 2.

maxiter : int, optional

Maximum number of iteration. Default is 30.

threshold : float, optional

Threshold in theq. Default is 0.001

etype : int, optional

Type of parameter `eps`

0. not used
1. initial value of log-periodogram
2. floor of periodogram in db

Default is 0.

eps : float, optional

Initial value for log-periodogram or floor of periodogram in db. Default is 0.0.

itype : float, optional

Input data type

0. windowed signal
1. log amplitude in db
2. log amplitude
3. amplitude
4. periodogram

Default is 0.

Returns **c** : array, shape (`order + 1`)

cepstrum estimated by uels

Raises **ValueError**

- if invalid `itype` is specified
- if invalid `etype` is specified
- if nonzero `eps` is specified when etype = 0

- if negative `eps` is specified

RuntimeError

- if zero(s) are found in periodogram

See also:

pysptk.sptk.gcep, *pysptk.sptk.mcep*, *pysptk.sptk.mgcep*, *pysptk.sptk.lmadf*

pysptk.sptk.fftcep

`pysptk.sptk.`**`fftcep`**(*logsp*, *order=25*, *num_iter=0*, *acceleration_factor=0.0*)

FFT-based cepstrum analysis

Parameters **logsp** : array, shape (`frame_len`)

Log power spectrum

order : int, optional

Order of cepstrum. Default is 25.

num_iter : int, optional

Number of iteration. Default is 0.

acceleration_factor : float, optional

Acceleration factor. Default is 0.0.

Returns **c** : array, shape (`order + 1`)

Cepstrum

See also:

pysptk.sptk.uels

pysptk.sptk.lpc

`pysptk.sptk.`**`lpc`**(*windowed*, *order=25*, *min_det=1e-06*)

Linear prediction analysis

Parameters **windowed** : array, shape (`frame_len`)

A windowed frame

order : int, optional

Order of LPC. Default is 25.

min_det : float, optional

Mimimum value of the determinant of normal matrix. Default is 1.0e-6.

Returns **a** : array, shape (`order + 1`)

LPC

Raises **ValueError**

- if negative `min_det` is specified

RuntimeError

- if error happened in levdur

See also:

pysptk.sptk.lpc2par, *pysptk.sptk.par2lpc*, *pysptk.sptk.lpc2c*, *pysptk.sptk.lpc2lsp*, *pysptk.sptk.ltcdf*, *pysptk.sptk.lspdf*

MFCC

mfcc(x[, order, fs, alpha, eps, window_len, ...])	MFCC

pysptk.sptk.mfcc

pysptk.sptk.**mfcc**(*x*, *order=14*, *fs=16000*, *alpha=0.97*, *eps=1.0*, *window_len=None*, *frame_len=None*, *num_filterbanks=20*, *cepslift=22*, *use_dft=False*, *use_hamming=False*, *czero=False*, *power=False*)

MFCC

Parameters **x** : array

A input signal

order : int, optional

Order of MFCC. Default is 14.

fs : int, optional

Sampling frequency. Default is 160000.

alpha : float, optional

Pre-emphasis coefficient. Default is 0.97.

eps : float, optional

Flooring value for calculating `log(x)` in filterbank analysis. Default is 1.0.

window_len : int, optional

Window lenght. Default is `len(x)`.

frame_len : int, optional

Frame length. Default is `len(x)`.

num_filterbanks : int, optional

Number of mel-filter banks. Default is 20.

cepslift : int, optional

Liftering coefficient. Default is 22.

use_dft : bool, optional

Use DFT (not FFT) or not. Default is False.

use_hamming : bool, optional

Use hamming window or not. Default is False.

czero : bool, optional

If True, `mfcc` returns 0-th coefficient as well. Default is False.

power : bool, optional

If True, `mfcc` returns power coefficient as well. Default is False.

Returns **cc** : array

MFCC vector, which is ordered as:

mfcc[0], mfcc[1], mfcc[2], ... mfcc[order-1], c0, Power.

Note that c0 and Power are optional.

Shape of `cc` is:

- `order` by default.
- `orde + 1` if `czero` or `power` is set to True.
- `order + 2` if both `czero` and `power` is set to True.

Raises **ValueError**

if `num_filterbanks` is less than or equal to `order`

See also:

pysptk.sptk.gcep, *pysptk.sptk.mcep*, *pysptk.sptk.mgcep*

LPC, LSP and PARCOR conversions

lpc2c(lpc[, order])	LPC to cepstrum
lpc2lsp(lpc[, numsp, maxiter, eps, loggain, ...])	LPC to LSP
lpc2par(lpc)	LPC to PARCOR
par2lpc(par)	PARCOR to LPC
lsp2sp(lsp[, fftlen])	LSP to spectrum

pysptk.sptk.lpc2c

pysptk.sptk.**lpc2c**(*lpc*, *order=None*)

LPC to cepstrum

Parameters **lpc** : array

LPC

order : int, optional

Order of cepstrum. Default is `len(lpc) - 1`.

Returns **ceps** : array, shape (`order + 1`)

cepstrum

See also:

pysptk.sptk.lpc, *pysptk.sptk.lspdf*

pysptk.sptk.lpc2lsp

pysptk.sptk.**lpc2lsp**(*lpc*, *numsp=512*, *maxiter=4*, *eps=1e-06*, *loggain=False*, *otype=0*, *fs=None*)

LPC to LSP

Parameters **lpc** : array

LPC

numsp : int, optional

Number of unit circle. Default is 512.

maxiter : int, optional

Maximum number of iteration. Default is 4.

eps : float, optional

End condition for iteration. Default is 1.0e-6.

loggain : bool, optional

whether the converted lsp should have loggain or not. Default is False.

fs : int, optional

Sampling frequency. Default is None and unused.

otype : int, optional

Output format LSP

0. normalized frequency (0 ~ pi)
1. normalized frequency (0 ~ 0.5)
2. frequency (kHz)
3. frequency (Hz)

Default is 0.

Returns **lsp** : array, shape (`order + 1`)

LSP

Raises **ValueError**

if `fs` is not specified when otype = 2 or 3.

See also:

pysptk.sptk.lpc, *pysptk.sptk.lspdf*

pysptk.sptk.lpc2par

pysptk.sptk.**lpc2par**(*lpc*)

LPC to PARCOR

Parameters **lpc** : array

LPC

Returns **par** : array, shape (same as `lpc`)

PARCOR

See also:

pysptk.sptk.lpc, *pysptk.sptk.par2lpc*, *pysptk.sptk.ltcdf*

pysptk.sptk.par2lpc

pysptk.sptk.**par2lpc**(*par*)

PARCOR to LPC

Parameters **par** : array

PARCOR

Returns **lpc** : array, shape (same as `par`)

LPC

See also:

pysptk.sptk.lpc, *pysptk.sptk.lpc2par*

pysptk.sptk.lsp2sp

pysptk.sptk.**lsp2sp**(*lsp*, *fftlen=256*)

LSP to spectrum

Parameters **lsp** : array

LSP

fftlen : int, optional

FFT length

TODO: consider ‘‘otype‘‘ optional argument

Returns **sp** : array, shape

Spectrum. ln|H(z)|.

See also:

pysptk.sptk.lpc2par

Notes

It is asuumed that `lsp` has loggain at `lsp[0]`.

Mel-generalized cepstrum conversions

mc2b(mc[, alpha])	Mel-cepsrum to MLSA filter coefficients
b2mc(b[, alpha])	MLSA filter coefficients to mel-cesptrum
c2acr(c[, order, fftlen])	Cepstrum to autocorrelation
c2ir(c[, length])	Cepstrum to impulse response
ic2ir(h[, order])	Impulse response to cepstrum
c2ndps(c[, fftlen])	Cepstrum to Negative Derivative of Phase Spectrum (NDPS)
ndps2c(ndps[, order])	Cepstrum to Negative Derivative of Phase Spectrum (NDPS)
gc2gc(src_ceps[, src_gamma, dst_order, . . .])	Generalized cepstrum transform

Continued on next page

Table 4.6 – continued from previous page

gnorm(ceps[, gamma])	Gain normalization
ignorm(ceps[, gamma])	Inverse gain normalization
freqt(ceps[, order, alpha])	Frequency transform
mgc2mgc(src_ceps[, src_alpha, src_gamma, ...])	Mel-generalized cepstrum transform
mgc2sp(ceps[, alpha, gamma, fftlen])	Mel-generalized cepstrum transform
mgclsp2sp(lsp[, alpha, gamma, fftlen, gain])	MGC-LSP to spectrum

pysptk.sptk.mc2b

pysptk.sptk.**mc2b**(*mc*, *alpha=0.35*)

Mel-cepsrum to MLSA filter coefficients

Parameters **mc** : array, shape

Mel-cepstrum.

alpha : float, optional

All-pass constant. Default is 0.35.

Returns **b** : array, shape(same as `mc`)

MLSA filter coefficients

See also:

pysptk.sptk.mlsadf, *pysptk.sptk.mglsadf*, *pysptk.sptk.b2mc*, *pysptk.sptk.mcep*, *pysptk.sptk.mgcep*, *pysptk.sptk.amcep*

pysptk.sptk.b2mc

pysptk.sptk.**b2mc**(*b*, *alpha=0.35*)

MLSA filter coefficients to mel-cesptrum

Parameters **b** : array, shape

MLSA filter coefficients

alpha : float, optional

All-pass constant. Default is 0.35.

Returns **mc** : array, shape (same as `b`)

Mel-cepstrum.

See also:

pysptk.sptk.mc2b, *pysptk.sptk.mcep*, *pysptk.sptk.mlsadf*

pysptk.sptk.c2acr

pysptk.sptk.**c2acr**(*c*, *order=None*, *fftlen=256*)

Cepstrum to autocorrelation

Parameters **c** : array

Cepstrum

order : int, optional

Order of cepstrum. Default is `len(c) - 1`.

fftlen : int, optional

FFT length. Default is 256.

Returns **r** : array, shape (`order + 1`)

Autocorrelation

Raises **ValueError**

if non power of 2 `fftlen` is specified

See also:

pysptk.sptk.uels, *pysptk.sptk.c2ir*, *pysptk.sptk.lpc2c*

pysptk.sptk.c2ir

pysptk.sptk.**c2ir**(*c*, *length=256*)

Cepstrum to impulse response

Parameters **c** : array

Cepstrum

length : int, optional

Length of impulse response. Default is 256.

Returns **h** : array, shape (`length`)

impulse response

See also:

pysptk.sptk.c2acr

pysptk.sptk.ic2ir

pysptk.sptk.**ic2ir**(*h*, *order=25*)

Impulse response to cepstrum

Parameters **h** : array

Impulse response

order : int, optional

Order of cepstrum. Default is 25.

Returns **c** : array, shape (`order + 1`)

Cepstrum

See also:

pysptk.sptk.c2ir

pysptk.sptk.c2ndps

pysptk.sptk.**c2ndps**(*c*, *fftlen=256*)

Cepstrum to Negative Derivative of Phase Spectrum (NDPS)

Parameters **c** : array

Cepstrum

fftlen : int, optional

FFT length. Default is 256.

Returns **ndps** : array, shape (`fftlen // 2 + 1`)

NDPS

Raises **ValueError**

if non power of 2 `fftlen` is specified

See also:

pysptk.sptk.mgcep, *pysptk.sptk.ndps2c*

pysptk.sptk.ndps2c

pysptk.sptk.**ndps2c**(*ndps*, *order=25*)

Cepstrum to Negative Derivative of Phase Spectrum (NDPS)

Parameters **ndps** : array, shape (`fftlen // 2 + 1`)

NDPS

order : int, optional

Order of cepstrum. Default is 25.

Returns **c** : array, shape (`order + 1`)

Cepstrum

Raises **ValueError**

if non power of 2 `fftlen` is detected

See also:

pysptk.sptk.mgc2sp, *pysptk.sptk.c2ndps*

pysptk.sptk.gc2gc

pysptk.sptk.**gc2gc**(*src_ceps*, *src_gamma=0.0*, *dst_order=None*, *dst_gamma=0.0*)

Generalized cepstrum transform

Parameters **src_ceps** : array

Generalized cepstrum.

src_gamma : float, optional

Gamma of source cepstrum. Default is 0.0.

dst_order : int, optional

Order of destination cepstrum. Default is `len(src_ceps) - 1`.

dst_gamma : float, optional

Gamma of destination cepstrum. Default is 0.0.

Returns **dst_ceps** : array, shape (`dst_order + 1`)

Converted generalized cepstrum

Raises **ValueError**

- if invalid `src_gamma` is specified
- if invalid `dst_gamma` is specified

See also:

pysptk.sptk.gcep, *pysptk.sptk.mgcep*, *pysptk.sptk.freqt*, *pysptk.sptk.mgc2mgc*, *pysptk.sptk.lpc2c*

pysptk.sptk.gnorm

pysptk.sptk.**gnorm**(*ceps*, *gamma=0.0*)

Gain normalization

Parameters **ceps** : array

Generalized cepstrum.

gamma : float, optional

Gamma. Default is 0.0.

Returns **dst_ceps** : array, shape(same as `ceps`)

Normalized generalized cepstrum

Raises **ValueError**

if invalid `gamma` is specified

See also:

pysptk.sptk.ignorm, *pysptk.sptk.gcep*, *pysptk.sptk.mgcep*, *pysptk.sptk.gc2gc*, *pysptk.sptk.mgc2mgc*, *pysptk.sptk.freqt*

pysptk.sptk.ignorm

pysptk.sptk.**ignorm**(*ceps*, *gamma=0.0*)

Inverse gain normalization

Parameters **c** : array

Normalized generalized cepstrum

gamma : float, optional

Gamma. Default is 0.0.

Returns **dst_ceps** : array, shape (same as `ceps`)

Generalized cepstrum

Raises **ValueError**

if invalid `gamma` is specified

See also:

pysptk.sptk.gnorm, *pysptk.sptk.gcep*, *pysptk.sptk.mgcep*, *pysptk.sptk.gc2gc*, *pysptk.sptk.mgc2mgc*, *pysptk.sptk.freqt*

pysptk.sptk.freqt

`pysptk.sptk.`**`freqt`**(*ceps*, *order=25*, *alpha=0.0*)

Frequency transform

Parameters **ceps** : array

Cepstrum.

order : int, optional

Desired order of transformed cepstrum. Default is 25.

alpha : float, optional

All-pass constant. Default is 0.0.

Returns **dst_ceps** : array, shape(`order + 1`)

frequency transofmed cepsttrum (typically mel-cepstrum)

See also:

pysptk.sptk.mgc2mgc

pysptk.sptk.mgc2mgc

`pysptk.sptk.`**`mgc2mgc`**(*src_ceps*, *src_alpha=0.0*, *src_gamma=0.0*, *dst_order=None*, *dst_alpha=0.0*, *dst_gamma=0.0*)

Mel-generalized cepstrum transform

Parameters **src_ceps** : array

Mel-generalized cepstrum.

src_alpha : float, optional

All-pass constant of source cesptrum. Default is 0.0.

src_gamma : float, optional

Gamma of source cepstrum. Default is 0.0.

dst_order : int, optional

Order of destination cepstrum. Default is `len(src_ceps) - 1`.

dst_alpha : float, optional

All-pass constant of destination cesptrum. Default is 0.0.

dst_gamma : float, optional

Gamma of destination cepstrum. Default is 0.0.

Returns **dst_ceps** : array, shape (`dst_order + 1`)

Converted mel-generalized cepstrum

Raises **ValueError**

- if invalid `src_gamma` is specified
- if invalid `dst_gamma` is specified

See also:

pysptk.sptk.uels, *pysptk.sptk.gcep*, *pysptk.sptk.mcep*, *pysptk.sptk.mgcep*, *pysptk.sptk.gc2gc*, *pysptk.sptk.freqt*, *pysptk.sptk.lpc2c*

pysptk.sptk.mgc2sp

`pysptk.sptk.`**`mgc2sp`**(*ceps*, *alpha=0.0*, *gamma=0.0*, *fftlen=256*)

Mel-generalized cepstrum transform

Parameters **ceps** : array

Mel-generalized cepstrum.

alpha : float, optional

All-pass constant. Default is 0.0.

gamma : float, optional

Gamma. Default is 0.0.

fftlen : int, optional

FFT length. Default is 256.

Returns **sp** : array, shape (`fftlen // 2 + 1`)

Complex spectrum

Raises **ValueError**

- if invalid `gamma` is specified
- if non power of 2 `fftlen` is specified

See also:

pysptk.sptk.mgc2mgc, *pysptk.sptk.gc2gc*, *pysptk.sptk.freqt*, *pysptk.sptk.gnorm*, *pysptk.sptk.lpc2c*

pysptk.sptk.mgclsp2sp

`pysptk.sptk.`**`mgclsp2sp`**(*lsp*, *alpha=0.0*, *gamma=0.0*, *fftlen=256*, *gain=True*)

MGC-LSP to spectrum

Parameters **lsp** : array

MGC-LSP

alpha : float, optional

All-pass constant. Default is 0.0.

gamma : float, optional

Gamma. Default is 0.0.

fftlen : int, optional

FFT length. Default is 256.

gain : bool, optional

Whether the input MGC-LSP should have loggain or not. Default is True.

Returns **sp** : array, shape (`fftlen // 2 + 1`)

Complex spectrum

Raises **ValueError**

- if invalid `gamma` is specified
- if non power of 2 `fftlen` is specified

See also:

pysptk.sptk.mgc2mgc

F0 analysis

swipe(x, fs, hopsize[, min, max, threshold, ...])	SWIPE' - A Saw-tooth Waveform Inspired Pitch Estimation
rapt(x, fs, hopsize[, min, max, voice_bias, ...])	RAPT - a robust algorithm for pitch tracking

pysptk.sptk.swipe

pysptk.sptk.**swipe**(*x*, *fs*, *hopsize*, *min=60.0*, *max=240.0*, *threshold=0.3*, *otype='f0'*)

SWIPE' - A Saw-tooth Waveform Inspired Pitch Estimation

Parameters **x** : array

A whole audio signal

fs : int

Sampling frequency.

hopsize : int

Hop size.

min : float, optional

Minimum fundamental frequency. Default is 60.0

max : float, optional

Maximum fundamental frequency. Default is 240.0

threshold : float, optional

Voice/unvoiced threshold. Default is 0.3.

otype : str or int, optional

Output format

0. pitch
1. f0
2. log(f0)

Default is f0.

Returns **f0** : array, shape(`np.ceil(float(len(x))/hopsize)`)

Estimated f0 trajectory

Raises **ValueError**

if invalid otype is specified

See also:

pysptk.sptk.rapt

Examples

pysptk.sptk.rapt

`pysptk.sptk.`**`rapt`**(*x*, *fs*, *hopsize*, *min=60*, *max=240*, *voice_bias=0.0*, *otype='f0'*)

RAPT - a robust algorithm for pitch tracking

Parameters **x** : array, dtype=np.float32

A whole audio signal

fs : int

Sampling frequency.

hopsize : int

Hop size.

min : float, optional

Minimum fundamental frequency. Default is 60.0

max : float, optional

Maximum fundamental frequency. Default is 240.0

voice_bias : float, optional

Voice/unvoiced threshold. Default is 0.0.

otype : str or int, optional

Output format

0. pitch
1. f0
2. log(f0)

Default is f0.

Returns **f0** : array, shape(`np.ceil(float(len(x))/hopsize)`)

Estimated f0 trajectory

Raises **ValueError**

- if invalid min/max frequency specified
- if invalid frame period specified (not in [1/fs, 0.1])
- if input range too small for analysis by get_f0

RuntimeError

- problem in init_dp_f0()

Please see also the RAPT code in SPTK for more detailed exception conditions.

See also:

pysptk.sptk.swipe

Notes

It is assumed that input array `x` has np.float32 dtype, while swipe assumes np.float64 dtype.

Examples

Excitation generation

excite(pitch[, hopsize, interp_period, ...])	Excitation generation

pysptk.sptk.excite

pysptk.sptk.**excite**(*pitch*, *hopsize=100*, *interp_period=1*, *gaussian=False*, *seed=1*)

Excitation generation

Parameters **pitch** : array

Pitch sequence.

Note: `excite` assumes that input is a **pitch** sequence, not **f0** sequence. Pitch sequence can be obtained by speficying `` `otype="pitch" `` to F0 estimation methods.

---

hopsize : int

Hop size (frame period in sample). Default is 100.

interp_period : int

Interpolation period. Default is 1.

gaussian : bool

If True, generate gausssian noise for unvoiced frames, otherwise generate M-sequence. Default is False.

seed : int

Seed for nrand for Gaussian noise. Default is 1.

Returns **excitation** : array

Excitation signal

See also:

pysptk.sptk.poledf, *pysptk.sptk.swipe*, *pysptk.sptk.rapt*

Window functions

blackman(n[, normalize])	Blackman window
hamming(n[, normalize])	Hamming window
hanning(n[, normalize])	Hanning window
bartlett(n[, normalize])	Bartlett window
trapezoid(n[, normalize])	Trapezoid window
rectangular(n[, normalize])	Rectangular window

pysptk.sptk.blackman

pysptk.sptk.**blackman**(*n*, *normalize=1*)

Blackman window

Parameters **n** : int

Window length

normalize : int, optional

Normalization flag

0. don't normalize
1. normalize by power
2. normalize by magnitude

Default is 1.

Returns **w** : array, shape (n,)

blackman window

pysptk.sptk.hamming

pysptk.sptk.**hamming**(*n*, *normalize=1*)

Hamming window

Parameters **n** : int

Window length

normalize : int, optional

Normalization flag

0. don't normalize
1. normalize by power
2. normalize by magnitude

Default is 1.

Returns **w** : array, shape (n,)

hamming window

pysptk.sptk.hanning

pysptk.sptk.**hanning**(*n*, *normalize=1*)
Hanning window

Parameters **n** : int

Window length

normalize : int, optional

Normalization flag

0. don't normalize
1. normalize by power
2. normalize by magnitude

Default is 1.

Returns **w** : array, shape (n,)

hanning window

pysptk.sptk.bartlett

pysptk.sptk.**bartlett**(*n*, *normalize=1*)
Bartlett window

Parameters **n** : int

Window length

normalize : int, optional

Normalization flag

0. don't normalize
1. normalize by power
2. normalize by magnitude

Default is 1.

Returns **w** : array, shape (n,)

bartlett window

pysptk.sptk.trapezoid

pysptk.sptk.**trapezoid**(*n*, *normalize=1*)
Trapezoid window

Parameters **n** : int

Window length

normalize : int, optional

Normalization flag

0. don't normalize

1. normalize by power
2. normalize by magnitude

Default is 1.

Returns **w** : array, shape (n,)

trapezoid window

pysptk.sptk.rectangular

pysptk.sptk.**rectangular**(*n*, *normalize=1*)

Rectangular window

Parameters **n** : int

Window length

normalize : int, optional

Normalization flag

0. don't normalize
1. normalize by power
2. normalize by magnitude

Default is 1.

Returns **w** : array, shape (n,)

rectangular window

Waveform generation filters

poledf(x, a, delay)	All-pole digital filter
lmadf(x, b, pd, delay)	LMA digital filter
lspdf(x, f, delay)	LSP synthesis digital filter
ltcdf(x, k, delay)	All-pole lattice digital filter
glsadf(x, c, stage, delay)	GLSA digital filter
mlsadf(x, b, alpha, pd, delay)	MLSA digital filter
mglsadf(x, b, alpha, stage, delay)	MGLSA digital filter

pysptk.sptk.poledf

pysptk.sptk.**poledf**(*x*, *a*, *delay*)

All-pole digital filter

Parameters **x** : float

A input sample

a : array

AR coefficients

delay : array

Delay

Returns **y** : float

A filtered sample

Raises **ValueError**

if invalid delay length is supplied

See also:

pysptk.sptk.lpc, *pysptk.sptk.ltcdf*, *pysptk.sptk.lmadf*

pysptk.sptk.lmadf

pysptk.sptk.**lmadf**(*x*, *b*, *pd*, *delay*)

LMA digital filter

Parameters **x** : float

A input sample

c : array

Cepstrum

pd : int

Order of pade approximation

delay : array

Delay

Returns **y** : float

A filtered sample

Raises **ValueError**

- if invalid order of pade approximation is specified
- if invalid delay length is supplied

See also:

pysptk.sptk.uels, *pysptk.sptk.acep*, *pysptk.sptk.poledf*, *pysptk.sptk.ltcdf*, *pysptk.sptk.glsadf*, *pysptk.sptk.mlsadf*, *pysptk.sptk.mglsadf*

pysptk.sptk.lspdf

pysptk.sptk.**lspdf**(*x*, *f*, *delay*)

LSP synthesis digital filter

Parameters **x** : float

A input sample

f : array

LSP coefficients

delay : array

Delay

Returns **y** : float

A filtered sample

Raises **ValueError**

if invalid delay length is supplied

See also:

pysptk.sptk.lpc2lsp

pysptk.sptk.ltcdf

pysptk.sptk.**ltcdf**(*x*, *k*, *delay*)

All-pole lattice digital filter

Parameters **x** : float

A input sample

k : array

PARCOR coefficients.

delay : array

Delay

Returns **y** : float

A filtered sample

Raises **ValueError**

if invalid delay length is supplied

See also:

pysptk.sptk.lpc, *pysptk.sptk.lpc2par*, *pysptk.sptk.lpc2lsp*, *pysptk.sptk.poledf*, *pysptk.sptk.lspdf*

pysptk.sptk.glsadf

pysptk.sptk.**glsadf**(*x*, *c*, *stage*, *delay*)

GLSA digital filter

Parameters **x** : float

A input sample

c : array

Geneeraized cepstrum

stage : int

-1 / gamma

delay : array

Delay

Returns **y** : float

A filtered sample

Raises **ValueError**

- if invalid number of stage is specified
- if invalid delay length is supplied

See also:

pysptk.sptk.ltcdf, *pysptk.sptk.lmadf*, *pysptk.sptk.lspdf*, *pysptk.sptk.mlsadf*, *pysptk.sptk.mglsadf*

pysptk.sptk.mlsadf

pysptk.sptk.**mlsadf**(*x*, *b*, *alpha*, *pd*, *delay*)

MLSA digital filter

Parameters **x** : float

A input sample

b : array

MLSA filter coefficients

alpha : float

All-pass constant

pd : int

Order of pade approximation

delay : array

Delay

Returns **y** : float

A filtered sample

Raises **ValueError**

- if invalid order of pade approximation is specified
- if invalid delay length is supplied

See also:

pysptk.sptk.mcep, *pysptk.sptk.amcep*, *pysptk.sptk.poledf*, *pysptk.sptk.ltcdf*, *pysptk.sptk.lmadf*, *pysptk.sptk.lspdf*, *pysptk.sptk.glsadf*, *pysptk.sptk.mglsadf*

pysptk.sptk.mglsadf

pysptk.sptk.**mglsadf**(*x*, *b*, *alpha*, *stage*, *delay*)

MGLSA digital filter

Parameters **x** : float

A input sample

b : array

MGLSA filter coefficients

alpha : float

All-pass constant

stage : int

-1 / gamma

delay : array

Delay

Returns **y** : float

A filtered sample

Raises **ValueError**

- if invalid number of stage is specified
- if invalid delay length is supplied

See also:

pysptk.sptk.mgcep, *pysptk.sptk.poledf*, *pysptk.sptk.ltcdf*, *pysptk.sptk.lmadf*, *pysptk.sptk.lspdf*, *pysptk.sptk.mlsadf*, *pysptk.sptk.glsadf*

Utilities for waveform generation filters

poledf_delay(order)	Delay for poledf
lmadf_delay(order, pd)	Delay for lmadf
lspdf_delay(order)	Delay for lspdf
ltcdf_delay(order)	Delay for ltcdf
glsadf_delay(order, stage)	Delay for glsadf
mlsadf_delay(order, pd)	Delay for mlsadf
mglsadf_delay(order, stage)	Delay for mglsadf

pysptk.sptk.poledf_delay

pysptk.sptk.**poledf_delay**(*order*)

Delay for poledf

Parameters **order** : int

Order of poledf filter coefficients

Returns **delay** : array

Delay

pysptk.sptk.lmadf_delay

pysptk.sptk.**lmadf_delay**(*order*, *pd*)

Delay for lmadf

Parameters **order** : int

Order of lmadf filter coefficients

pd : int

Order of pade approximation.

Returns **delay** : array

Delay

pysptk.sptk.lspdf_delay

pysptk.sptk.**lspdf_delay**(*order*)
Delay for lspdf

Parameters **order** : int

Order of lspdf filter coefficients

Returns **delay** : array

Delay

pysptk.sptk.ltcdf_delay

pysptk.sptk.**ltcdf_delay**(*order*)
Delay for ltcdf

Parameters **order** : int

Order of ltcdf filter coefficients

Returns **delay** : array

Delay

pysptk.sptk.glsadf_delay

pysptk.sptk.**glsadf_delay**(*order*, *stage*)
Delay for glsadf

Parameters **order** : int

Order of glsadf filter coefficients

stage : int

-1 / gamma

Returns **delay** : array

Delay

pysptk.sptk.mlsadf_delay

pysptk.sptk.**mlsadf_delay**(*order*, *pd*)
Delay for mlsadf

Parameters **order** : int

Order of mlsadf filter coefficients

pd : int

Order of pade approximation.

Returns **delay** : array

Delay

pysptk.sptk.mglsadf_delay

pysptk.sptk.**mglsadf_delay**(*order*, *stage*)

Delay for mglsadf

Parameters **order** : int

Order of mglsadf filter coefficients

stage : int

-1 / gamma

Returns **delay** : array

Delay

Metrics

cdist(c1, c2[, otype, frame])	Calculation of cepstral distance

pysptk.sptk.cdist

pysptk.sptk.**cdist**(*c1*, *c2*, *otype=0*, *frame=False*)

Calculation of cepstral distance

Parameters **c1** : array

Minimum-phase cepstrum

c2 : array

Minimum-phase cepstrum

otype : int

Output data type

0. [db]
1. squared error
2. root squared error

Default is 0.

frame : bool

If True, returns frame-wise distance, otherwise returns mean distance. Default is False.

Returns distance

4.1.2 Other conversions

Not exist in SPTK itself, but can be used with the core API. Functions in the `pysptk.conversion` module can also be directly accesible by `pysptk.*`.

mgc2b(mgc[, alpha, gamma])	Mel-generalized cepstrum to MGLSA filter coefficients
sp2mc(powerspec, order, alpha)	Convert spectrum envelope to mel-cepstrum
mc2sp(mc, alpha, fftlen)	Convert mel-cepstrum back to power spectrum
mc2e(mc[, alpha, irlen])	Compute energy from mel-cepstrum

pysptk.conversion.mgc2b

`pysptk.conversion.`**`mgc2b`**(*mgc*, *alpha=0.35*, *gamma=0.0*)

Mel-generalized cepstrum to MGLSA filter coefficients

Parameters **mgc** : array, shape

Mel-generalized cepstrum

alpha : float

All-pass constant. Default is 0.35.

gamma : float

Parameter of generalized log function. Default is 0.0.

Returns **b** : array, shape(same as `mgc`)

MGLSA filter coefficients

See also:

pysptk.sptk.mlsadf, *pysptk.sptk.mglsadf*, *pysptk.sptk.mc2b*, *pysptk.sptk.b2mc*, *pysptk.sptk.mcep*, *pysptk.sptk.mgcep*

pysptk.conversion.sp2mc

`pysptk.conversion.`**`sp2mc`**(*powerspec*, *order*, *alpha*)

Convert spectrum envelope to mel-cepstrum

This is a simplified implementation of `mcep` for input type is 4.

Parameters **powerspec** : array

Power spectrum

order : int

Order of mel-cepstrum

alpha : float

All-pass constant.

Returns **mc** : array, shape(`order+1`)

mel-cepstrum

See also:

pysptk.sptk.mcep, *pysptk.conversion.mc2sp*

pysptk.conversion.mc2sp

pysptk.conversion.**mc2sp**(*mc*, *alpha*, *fftlen*)
Convert mel-cepstrum back to power spectrum

Parameters **mc** : array

Mel-spectrum

alpha : float

All-pass constant.

fftlen : int

FFT length

Returns **powerspec** : array, shape(`fftlen//2 +1`)

Power spectrum

See also:

pysptk.sptk.mcep, *pysptk.conversion.sp2mc*

pysptk.conversion.mc2e

pysptk.conversion.**mc2e**(*mc*, *alpha=0.35*, *irlen=256*)
Compute energy from mel-cepstrum

Inspired from hts_engine

Parameters **mc** : array

Mel-spectrum

alpha : float

All-pass constant.

irlen : int

IIR filter length

Returns **energy** : floating point, scalar

frame energy

4.1.3 High-level interface for waveform synthesis

Module `pysptk.synthesis` provides high-leve interface that wraps low-level SPTK waveform synthesis functions (e.g. `mlsadf`),

Synthesizer

class pysptk.synthesis.**Synthesizer**(*filt*, *hopsize*)
Speech waveform synthesizer

Attributes

filt	(SynthesisFilter) A speech synthesis filter
hopsize	(int) Hop size

synthesis (*source*, *b*)
Synthesize a waveform given a source excitation and sequence of filter coefficients (e.g. cepstrum).

Parameters **source** : array

Source excitation

b : array

Filter coefficients

Returns **y** : array, shape (same as `source`)

Synthesized waveform

synthesis_one_frame (*source*, *prev_b*, *curr_b*)
Synthesize one frame waveform

Parameters **source** : array

Source excitation

prev_b : array

Filter coefficients of previous frame

curr_b : array

Filter coefficients of current frame

Returns **y** : array

Synthesized waveform

SynthesisFilters

LMADF

class pysptk.synthesis.**LMADF** (*order=25*, *pd=4*)
LMA digital filter that wraps `lmadf`

Attributes

pd	(int) Order of pade approximation. Default is 4.
delay	(array) Delay

filt (*x*, *coef*)
Filter one sample using using `lmadf`

Parameters **x** : float

A input sample

coef: array

LMA filter coefficients (i.e. Cepstrum)

Returns **y** : float

A filtered sample

See also:

pysptk.sptk.lmadf

MLSADF

class pysptk.synthesis.**MLSADF**(*order=25*, *alpha=0.35*, *pd=4*)

MLSA digital filter that wraps `mlsadf`

Attributes

alpha	(float) All-pass constant
pd	(int) Order of pade approximation. Default is 4.
delay	(array) Delay

filt(*x*, *coef*)

Filter one sample using `mlsadf`

Parameters **x** : float

A input sample

coef: array

MLSA filter coefficients

Returns **y** : float

A filtered sample

See also:

pysptk.sptk.mlsadf, *pysptk.sptk.mc2b*

MGLSADF

class pysptk.synthesis.**MGLSADF**(*order=25*, *alpha=0.35*, *stage=1*)

MGLSA digital filter that wraps `mglsadf`

Attributes

alpha	(float) All-pass constant
stage	(int) -1/gamma
delay	(array) Delay

filt(*x*, *coef*)

Filter one sample using `mglsadf`

Parameters **x** : float

A input sample

coef: array

MGLSA filter coefficients

Returns **y** : float

A filtered sample

See also:

pysptk.sptk.mglsadf

AllPoleDF

class pysptk.synthesis.**AllPoleDF**(*order=25*)

All-pole digital filter that wraps `poledf`

Attributes

delay	(array) Delay

filt(*x*, *coef*)

Filter one sample using using `poledf`

Parameters **x** : float

A input sample

coef: array

LPC (with loggain)

Returns **y** : float

A filtered sample

See also:

pysptk.sptk.poledf

AllPoleLatticeDF

class pysptk.synthesis.**AllPoleLatticeDF**(*order=25*)

All-pole lttice digital filter that wraps `ltcdf`

Attributes

delay	(array) Delay

filt(*x*, *coef*)

Filter one sample using using `ltcdf`

Parameters **x** : float

A input sample

coef: array

PARCOR coefficients (with loggain)

Returns **y** : float

A filtered sample

See also:

pysptk.sptk.ltcdf

Synthesis filter interface

class pysptk.synthesis.**SynthesisFilter**

Synthesis filter interface

All synthesis filters must implement this interface

filt(*x*, *coef*)

Filter one sample

Parameters **x** : float

A input sample

coef : array

Filter coefficients

Returns **y** : float

A filtered sample

4.1.4 Utilities

Audio files

example_audio_file()	Get the path to an included audio example file.

pysptk.util.example_audio_file

pysptk.util.**example_audio_file**()

Get the path to an included audio example file.

Examples

Mel-cepstrum analysis

mcepalpha(fs[, start, stop, step, num_points])	Compute appropriate frequency warping parameter given a sampling frequency

pysptk.util.mcepalpha

pysptk.util.**mcepalpha**(*fs*, *start=0.0*, *stop=1.0*, *step=0.001*, *num_points=1000*)

Compute appropriate frequency warping parameter given a sampling frequency

It would be useful to determine alpha parameter in mel-cepstrum analysis.

The code is traslated from https://bitbucket.org/happyalu/mcep_alpha_calc.

Parameters **fs** : int

Sampling frequency

start : float

start value that will be passed to numpy.arange. Default is 0.0.

stop : float

stop value that will be passed to numpy.arange. Default is 1.0.

step : float

step value that will be passed to numpy.arange. Default is 0.001.

num_points : int

Number of points used in approximating mel-scale vectors in fixed- length.

Returns **alpha** : float

frequency warping paramter (offen denoted by alpha)

See also:

pysptk.sptk.mcep, *pysptk.sptk.mgcep*

CHAPTER 5

Developer Documentation

5.1 Developer Documentation

5.1.1 Design principle

pysptk is a thin python wrapper of SPTK. It is designed to be API consistent with the original SPTK as possible, but give better interface. There are a few design principles to wrap C interface:

1. Avoid really short names for variables (e.g. a, b, c, aa, bb, dd)

   Variable names should be informative. If the C functions have such short names, use self-descriptive names instead for python interfaces, unless they have clear meanings in their context.

2. Avoid too many function arguments

   Less is better. If the C functions have too many function arguments, use keyword arguments with proper default values for optional ones in python.

3. Handle errors in python

   Since C functions might *exit* (unfortunately) inside their functions for unexpected inputs, it should be check if the inputs are supported or not in python.

To wrap C interface, Cython is totally used.

5.1.2 How to build pysptk

You have to install `numpy` and `cython` first, and then:

```
git clone https://github.com/r9y9/pysptk
cd pysptk
git submodule update --init
python setup.py develop
```

should work.

Note: Dependency to the SPTK is added as a submodule. You have to checkout the supported SPTK as `git sudmobule update --init` before running setup.py.

5.1.3 How to build docs

pysptk docs are managed by the python sphinx. Docs-related dependencies can be resolved by:

```
pip install .[docs]
```

at the top of pysptk directory.

To build docs, go to the *docs* directory and then:

```
make html
```

You will see the generated docs in *_build* directory as follows (might different depends on sphinx version):

```
% tree _build/ -d
_build/
├── doctrees
│   └── generated
├── html
│   ├── _images
│   ├── _modules
│   │   └── pysptk
│   ├── _sources
│   │   └── generated
│   ├── _static
│   │   ├── css
│   │   ├── fonts
│   │   └── js
│   └── generated
└── plot_directive
    └── generated
```

See *_build/html/index.html* for the top page of the generated docs.

5.1.4 How to add a new function

There are a lot of functions unexposed from SPTK. To add a new function to pysptk, there are a few typical steps:

1. Add function signature to `_sptk.pxd`
2. Add cython implementation to `_sptk.pyx`
3. Add python interface (with docstrings) to `sptk.py` (or some proper module)

As you can see in setup.py, `_sptk.pyx` and SPTK sources are compiled into a single extension module.

Note: You might wonder why cython implementation and python interface should be separated because cython module can be directly accessed by python. The reasons are 1) to avoid rebuilding cython module when docs strings

are changed in the source 2) to make doc looks great, since sphinx seems unable to collect function argments correctly from cython module for now. Relevant issue: pysptk/#33

An example

In _sptk.pyd:

```
cdef extern from "SPTK.h":
    double _agexp "agexp"(double r, double x, double y)
```

In _sptk.pyx:

```
def agexp(r, x, y):
    return _agexp(r, x, y)
```

In sptk.pyx:

```
def agexp(r, x, y):
    """Magnitude squared generalized exponential function

    Parameters
    ----------
    r : float
        Gamma
    x : float
        Real part
    y : float
        Imaginary part

    Returns
    -------
    Value

    """
    return _sptk.agexp(r, x, y)
```

5.2 Change log

5.2.1 v0.1.11 <2018-xx-xx>

- #55: Add numpy implementation of cdist

5.2.2 v0.1.10 <2018-01-02>

- #54: Changes from SPTK v3.11 release. 6 and 7 pade approximatino in lmadf and mlsadf is now supported,

5.2.3 v0.1.9 <2018-01-01>

- BUG fix: example_audio_data is now included in the release tar.gz

5.2.4 v0.1.8 <2017-12-25>

- c2acr: Fix segfaults for small fftsize

5.2.5 v0.1.7 <2017-06-28>

- Extend vec2vec functions to mat2mat #49
- Support automatic type conversions #48

5.2.6 v0.1.6 <2017-05-18>

- Add `mcepalpha`. #43
- Add `mc2e`. #42
- Add `sp2mc` and `mc2sp`. #41

5.2.7 v0.1.5 <2017-04-22>

- Fix mcep eps check and input length #39

5.2.8 v0.1.4 <2015-11-23>

- Add developer documentation (#34)
- Separate cython implementation and interface (#35)
- Add RAPT (#32)
- Add excite function (#31) @jfsantos
- Fix inconsistent docs about normalization flag for window functions
- Fix test failure in c2dps / ndps2c (#29)

5.2.9 v0.1.3 <2015-10-02>

- Building binary wheels for Windows using Appveyor (#28)
- Add Installation guide on windows (#25)
- Start Windows continuous integration on AppVeyor (#24). As part of the issue, binary dependency was updated so that SPTK library can be compiled on linux, osx and Windows as well.
- Remove unnecesarry array initialization (#23)

5.2.10 v0.1.2 <2015-09-12>

- Add `pysptk.synthesis` package that provides high level interfaces for speech waveform synthesis (#14)
- Add cross-link to the docs
- Add `pysptk.conversion.mgc2b`
- Add speech analysis and re-synthesis demonstration notebook (#13)

- Add `pysptk.util.example_audio_file`
- Add `fftcep` (#18)
- Add `mfcc` (#21)
- Cython is now only required to build development versioni of pysptk. (#8)

5.2.11 v0.1.1 <2015-09-05>

- Include *.c to pypi distribution

5.2.12 v0.1.0 <2015-09-05>

- Initial release

CHAPTER 6

Indices and tables

- genindex
- search

Python Module Index

p

Index

N

P

R

S

T

U