BSS Oracle Toolbox Version 2.1

User Guide

Emmanuel Vincent

Rémi Gribonval

Mark D. Plumbley

October 12, 2007

Contents

Chapter 1

Getting started

1.1 Download and install

Two versions of the BSS Oracle toolbox are available.

The basic version of the toolbox, which includes the main Matlab®[1] programs and this user guide, can be downloaded at
`http://bass-db.gforge.inria.fr/bss_oracle/bss_oracle_basic.zip`
After unzipping this file, you should get a directory called `bss_oracle_2.1`. To install, simply add the full path to this directory to your Matlab® path using the command `pathtool`.

The full version of the toolbox, which includes additional example data and Matlab® programs, can be downloaded at
`http://bass-db.gforge.inria.fr/bss_oracle/bss_oracle_full.zip`
After unzipping this file, you should get in addition to the main directory called `bss_oracle_2.1` two sub-directories called `examples` and `data`. To install, simply add the full paths to the main directory and to the sub-directories to your Matlab® path.

1.2 Software dependencies

BSS Oracle consists in a set of Matlab®functions, and as such needs Matlab® to run.

Some functions of BSS Oracle (involving MDCT, CP or WP transforms) also depend on the Wavelab toolbox version 802 by D. Donoho, M.R. Duncan, X. Huo and O. Levi, available at
`http://www-stat.stanford.edu/~wavelab/`
Follow the provided documentation for install instructions. Note that Wavelab is copyrighted and cannot be redistributed together with BSS Oracle.

1.3 Getting help

Within Matlab®, you can get basic help about the toolbox by typing `help bss_oracle_2.1`

[1] Matlab® is a registered trademark of The MathWorks, Inc.

1.4 Citation

If you use the BSS Oracle toolbox in a work that you wish to publish, please cite it as:

E. Vincent, R. Gribonval and M.D. Plumbley. BSS Oracle Toolbox Version 2.1.
`http://bass-db.gforge.inria.fr/bss_oracle/`

1.5 Licenses

The files contained in the BSS Oracle toolbox are distributed under different licenses. Therefore it is crucial that you understand which license applies to each file before attempting to redistribute or modify some files.

The files contained in the main directory `bss_oracle_2.1` and in the subdirectory `examples` are distributed under the terms of the GNU General Public License (GPL) version 2. A copy of the GPL is distributed along with the toolbox in the file `LICENSE.txt`.

The music sound files contained in the subdirectory `data` are distributed under specific Creative Commons licenses. For more details about the license applying to each file, see the file `data/LICENSES.txt`.

All other files of the subdirectory `data` are license free.

Chapter 2

Content

The purpose of the BSS Oracle toolbox is to compute the best performance achievable by a class of source separation algorithms in an evaluation framework where target signals are known. It does not provide any blind source separation method.

2.1 Reference publications

The mathematical details underlying the toolbox are described in [3, 1, 2].

2.2 Principle

Let us suppose that we observe a mixture signal $\mathbf{x}(t)$ from which we want to extract a set of source signals $\mathbf{y}(t)$. Within a given class of source separation algorithms, the estimated signal $\widehat{\mathbf{y}}(t)$ can always be expressed under the form

$$\widehat{\mathbf{y}} = f(\mathbf{x}, \theta) \quad \text{with } \theta \in \Theta, \tag{2.1}$$

where f is a fixed parametric function, θ a vector of separation parameters and Θ a set of acceptable parameters. Different algorithms correspond to different ways of estimating θ.

Assuming that the target signal $\mathbf{y}(t)$ is known, the separation performance of a given algorithm can be evaluated using the Euclidean distortion measure

$$d(\widehat{\mathbf{y}}, \mathbf{y}) = \|\widehat{\mathbf{y}} - \mathbf{y}\|^2. \tag{2.2}$$

The oracle estimator of the target signal is then defined by

$$\widetilde{\mathbf{y}}(\mathbf{y}, \mathbf{x}, \Theta) = f(\mathbf{x}, \widetilde{\theta}(\mathbf{y}, \mathbf{x}, \Theta)), \tag{2.3}$$

where $\widetilde{\theta}(\mathbf{y}, \mathbf{x}, \Theta)$ is the set of parameters resulting in the smallest distortion among the set of acceptable parameters Θ:

$$\widetilde{\theta}(\mathbf{y}, \mathbf{x}, \Theta) = \arg\min_{\theta \in \Theta} d(f(\mathbf{x}, \theta), \mathbf{y}). \tag{2.4}$$

2.3 Summary of the content

The basic version of the toolbox implements oracle source estimators for four classes of algorithms: multichannel time-invariant filtering, single-channel time-frequency masking, multichannel time-frequency masking and best basis masking. In some cases, the exact oracle estimators

cannot be computed due to high memory and/or computational time requirements. Thus near-optimal source estimators are implemented instead.

The full version of the toolbox also contains example data and routines that were used to create the figures of the reference publications.

Chapter 3

Reference manual

3.1 General notations

T	length of the signals in samples
I	number of channels of the mixture signal
J	number of source or target signals
L	length of the demixing filters, or MDCT/STFT length
M	stepsize between successive STFT windows
K	number of mixture signals (for generic oracle bases)

3.2 Oracle source estimators

`bss_oracle_multifilt`

Oracle estimator for source separation by multichannel time-invariant filtering in the time domain.

Syntax:

```
[Se,W,SDR]=bss_oracle_multifilt(X,S,L)
```

Inputs:

`X`	$I \times T$ matrix containing the multichannel mixture signal
`S`	$J \times T$ matrix containing the target signals (e.g. sources or source images)
`L`	length of the demixing filters in samples

Outputs:

`Se`	$J \times T$ matrix containing the oracle estimates of the target signals (truncated to the same time range as the original signals)
`W`	$J \times I \times L$ table containing the coefficients of the oracle demixing filters (delays from $-\frac{L}{2}+1$ to $\frac{L}{2}$)
`SDR`	achieved SDR in deciBels (before truncation of the target estimates)

Reference:

See [3] Section 4.2.

`bss_oracle_monomask`

Oracle estimator for single-channel source separation by time-frequency masking using MDCT with sine window.

Syntax:

```
[Se,W,SDR]=bss_oracle_monomask(x,S,L)
[Se,W,SDR]=bss_oracle_monomask(x,S,L,mreal)
[Se,W,SDR]=bss_oracle_monomask(x,S,L,mreal,mconst)
```

Inputs:

`x`	$1 \times T$ vector containing the single-channel mixture signal
`S`	$J \times T$ matrix containing the target signals (e.g. source images)
`L`	length of the MDCT window in samples (must be a multiple of 4)
`mreal`	`true` for real-valued masking (default), `false` for binary masking
`mconst`	`true` when the masks are subject to a unitary sum constraint (default), `false` otherwise

Outputs:

`Se`	$J \times T$ matrix containing the oracle estimates of the target signals (truncated to the same time range as the original signals)
`W`	$\frac{L}{2} \times N \times J$ table containing the oracle masks with $N = \text{ceil}\left(\frac{2T}{L}\right)$
`SDR`	achieved SDR in deciBels (before truncation of the target estimates)

Reference:

See [3] Section 5.2.

bss_oracle_binmask

Oracle estimator for multichannel source separation by constrained binary time-frequency masking using MDCT with sine window.

Syntax:

```
[Se,W,SDR]=bss_oracle_binmask(X,S,L)
```

Inputs:

`X`	$I \times T$ matrix containing the multichannel mixture signal
`S`	$I \times T \times J$ table containing the target signals (source images)
`L`	length of the MDCT window in samples (must be a multiple of 4)

Outputs:

`Se`	$I \times T \times J$ table containing the oracle estimates of the target signals (truncated to the same time range as the original signals)
`W`	$\frac{L}{2} \times N \times J$ table containing the oracle masks with $N = \text{ceil}\left(\frac{2T}{L}\right)$
`SDR`	achieved SDR in deciBels (before truncation of the target estimates)

Reference:

See [3] Section 6.2.

`bss_oracle_pinvmask`

Oracle estimator for multichannel source separation of instantaneous mixtures by time-frequency masking and mixing matrix pseudo-inversion using MDCT with sine window.

Syntax:

```
[Se,W,SDR]=bss_oracle_pinvmask(X,S,L,A)
[Se,W,SDR]=bss_oracle_pinvmask(X,S,L,A,Ja)
```

Inputs:

`X`	$I \times T$ matrix containing the multichannel mixture signal
`S`	$I \times T \times J$ table containing the target signals (source images)
`L`	length of the MDCT window in samples (must be a multiple of 4)
`A`	$I \times J$ real-valued mixing matrix (may be different from the one actually used to generate S)
`Ja`	number of active sources per time-frequency point (by default or if $Ja = 0$, the best number is estimated for each time-frequency point)

Outputs:

`Se`	$I \times T \times J$ table containing the oracle estimates of the target signals (truncated to the same time range as the original signals)
`W`	$\frac{L}{2} \times N \times J$ table of binary coefficients indicating the oracle source activity patterns with $N = \text{ceil}\left(\frac{2T}{L}\right)$
`SDR`	achieved SDR in deciBels (before truncation of the target estimates)

Reference:

See [3] Section 6.2.

`bss_oracle_bbasis_monomask`

Oracle estimator for single-channel source separation by time-frequency masking using the best CP/WP basis.

Syntax:

```
[Se,btree,W,SDR,stree]=bss_oracle_bbasis_monomask(x,S,Dmin,Dmax)
[Se,btree,W,SDR,stree]=bss_oracle_bbasis_monomask(x,S,Dmin,Dmax,pcos)
[Se,btree,W,SDR,stree]=bss_oracle_bbasis_monomask(x,S,Dmin,Dmax,pcos,mreal)
[Se,btree,W,SDR,stree]=bss_oracle_bbasis_monomask(x,S,Dmin,Dmax,pcos,mreal,mconst)
```

Inputs:

`x`	$1 \times T$ vector containing the single-channel mixture signal
`S`	$J \times T$ matrix containing the target signals (e.g. source images)
`Dmin`	minimal packet depth
`Dmax`	maximal packet depth
`pcos`	`true` for CP basis with sine window (default), `false` for WP basis with symmlet-8
`mreal`	`true` for real-valued masking (default), `false` for binary masking
`mconst`	`true` when the masks are subject to a unitary sum constraint (default), `false` otherwise

Outputs:

`Se`	$J \times T$ matrix containing the oracle estimates of the target signals (truncated to the same time range as the original signals)
`btree`	$1 \times (2^{D_{\max}+1} - 1)$ vector of binary values representing the tree structure corresponding to the oracle best basis
`W`	$2^N \times (D_{\max} - D_{\min} + 1) \times J$ table containing the oracle masking coefficients for each scale with $N = \text{nextpow2}(T)$
`SDR`	achieved SDR in deciBels (before truncation of the target estimates)
`stree`	$1 \times (2^{D_{\max}+1} - 1)$ vector containing the oracle distortion for all basis elements (infinite for disallowed scales)

Reference:

See [2] Section 7.2.

`bss_oracle_bbasis_binmask`

Oracle estimator for multichannel source separation by constrained binary time-frequency masking using the best CP/WP basis.

Syntax:

```
[Se,btree,W,SDR,stree]=bss_oracle_bbasis_binmask(X,S,Dmin,Dmax)
[Se,btree,W,SDR,stree]=bss_oracle_bbasis_binmask(X,S,Dmin,Dmax,pcos)
```

Inputs:

`X`	$I \times T$ matrix containing the multichannel mixture signal
`S`	$J \times T$ matrix containing the target signals (source images)
`Dmin`	minimal packet depth
`Dmax`	maximal packet depth
`pcos`	`true` for CP basis with sine window (default), `false` for WP basis with symmlet-8

Outputs:

`Se`	$J \times T$ matrix containing the oracle estimates of the target signals (truncated to the same time range as the original signals)
`btree`	$1 \times (2^{D_{\max}+1} - 1)$ vector of binary values representing the tree structure corresponding to the oracle best basis
`W`	$2^N \times (D_{\max} - D_{\min} + 1) \times J$ table containing the oracle masking coefficients for each scale with $N = \text{nextpow2}(T)$
`SDR`	achieved SDR in deciBels (before truncation of the target estimates)
`stree`	$1 \times (2^{D_{\max}+1} - 1)$ vector containing the oracle distortion for all basis elements (infinite for disallowed scales)

Reference:

See [1].

`bss_oracle_bbasis_pinvmask`

Oracle estimator for multichannel source separation of instantaneous mixtures by time-frequency masking and mixing matrix pseudo-inversion using the best CP/WP basis.

Syntax:

```
[Se,btree,W,SDR,stree]=bss_oracle_bbasis_pinvmask(X,S,A,Dmin,Dmax)
[Se,btree,W,SDR,stree]=bss_oracle_bbasis_pinvmask(X,S,A,Dmin,Dmax,Ja)
[Se,btree,W,SDR,stree]=bss_oracle_bbasis_pinvmask(X,S,A,Dmin,Dmax,Ja,pcos)
```

Inputs:

`X`	$I \times T$ matrix containing the multichannel mixture signal
`S`	$J \times T$ matrix containing the target signals (source images)
`A`	$I \times J$ real-valued mixing matrix (may be different from the one actually used to generate S)
`Dmin`	minimal packet depth
`Dmax`	maximal packet depth
`Ja`	number of active sources per time-frequency point (by default or if $Ja = 0$, the best number is estimated for each time-frequency point)
`pcos`	`true` for CP basis with sine window (default), `false` for WP basis with symmlet-8

Outputs:

`Se`	$J \times T$ matrix containing the oracle estimates of the target signals (truncated to the same time range as the original signals)
`btree`	$1 \times (2^{D_{\max}+1} - 1)$ vector of binary values representing the tree structure corresponding to the oracle best basis
`W`	$2^N \times (D_{\max} - D_{\min} + 1) \times J$ table containing the oracle masking coefficients for each scale with $N = \text{nextpow2}(T)$
`SDR`	achieved SDR in deciBels (before truncation of the target estimates)
`stree`	$1 \times (2^{D_{\max}+1} - 1)$ vector containing the oracle distortion for all basis elements (infinite for disallowed scales)

Reference:

See [1] Section 3.2.

`bss_oracle_gbasis_monomask`

Oracle estimator for single-channel source separation of several mixtures by time-frequency masking using the best generic CP/WP basis.

Syntax:

```
[Se,btree,SDR,gSDR,stree]=bss_oracle_gbasis_monomask(x,S,Dmin,Dmax)
[Se,btree,SDR,gSDR,stree]=bss_oracle_gbasis_monomask(x,S,Dmin,Dmax,pcos)
[Se,btree,SDR,gSDR,stree]=bss_oracle_gbasis_monomask(x,S,Dmin,Dmax,pcos,mreal)
[Se,btree,SDR,gSDR,stree]=bss_oracle_gbasis_monomask(x,S,Dmin,Dmax,pcos,mreal,mconst)
```

Warnings:

Despite the default setting, this function is mostly relevant for WP bases.
Due to huge memory requirements, the optimal masks are not output.
The function stores temporary data in a temporary directory defined by the variable `tmpdir` (default is `/tmp/`).

Inputs:

`x`	$1 \times T \times K$ table containing K single-channel mixture signals
`S`	$J \times T \times K$ table containing K sets of target signals (e.g. source images)
`Dmin`	minimal packet depth
`Dmax`	maximal packet depth
`pcos`	`true` for CP basis with sine window (default), `false` for WP basis with symmlet-8
`mreal`	`true` for real-valued masking (default), `false` for binary masking
`mconst`	`true` when the masks are subject to a unitary sum constraint (default), `false` otherwise

Outputs:

`Se`	$J \times T \times K$ table containing the oracle estimates of the target signals (truncated to the same time range as the original signals)
`btree`	$1 \times (2^{D_{\max}+1} - 1)$ vector of binary values representing the tree structure corresponding to the oracle best generic basis
`SDR`	$K \times 1$ vector containing the achieved SDR in deciBels for each mixture (before truncation of the target estimates)
`gSDR`	total SDR for all mixtures
`stree`	$1 \times (2^{D_{\max}+1} - 1)$ vector containing the oracle distortion for all basis elements (infinite for disallowed scales)

Reference:

See [2] Section 7.3.2.

3.3 Near-optimal source estimators

`bss_nearopt_multifilt`

Near-optimal demixing matrices for source separation by frequency-domain multichannel time-invariant filtering using STFT with sine window (coefficients derived for each frequency bin separately).

Syntax:

```
[Se,W,SDR]=bss_nearopt_multifilt(X,S,L)
[Se,W,SDR]=bss_nearopt_multifilt(X,S,L,M)
```

Inputs:

`X`	$I \times T$ matrix containing the multichannel mixture signal
`S`	$J \times T$ matrix containing the target signals (e.g. sources or source images)
`L`	length of the STFT window in samples (must be a multiple of 4)
`M`	step between successive windows in samples (must be a multiple of 2, a divider of L and smaller than $\frac{L}{2}$) (default: $\frac{L}{2}$)

Outputs:

`Se`	$J \times T$ matrix containing the near-optimal estimates of the target signals (truncated to the same time range as the original signals)
`W`	$J \times I \times \left(\frac{L}{2}+1\right)$ table containing near-optimal demixing matrices for positive frequencies
`SDR`	achieved SDR in deciBels (before truncation of the target estimates)

Reference:

See [3] Section 4.4.

`bss_nearopt_monomask`

Near-optimal time-frequency masks for single-channel source separation using STFT with sine window (coefficients derived for each time-frequency point separately).

Syntax:

```
[Se,W,SDR]=bss_nearopt_monomask(x,S,L)
[Se,W,SDR]=bss_nearopt_monomask(x,S,L,M)
[Se,W,SDR]=bss_nearopt_monomask(x,S,L,M,mreal)
[Se,W,SDR]=bss_nearopt_monomask(x,S,L,M,mreal,mconst)
```

Inputs:

`x`	$1 \times T$ vector containing the single-channel mixture signal
`S`	$J \times T$ matrix containing the target signals (e.g. source images)
`L`	length of the STFT window in samples (must be a multiple of 4)
`M`	step between successive windows in samples (must be a multiple of 2, a divider of L and smaller than $\frac{L}{2}$) (default: $\frac{L}{2}$)
`mreal`	`true` for real-valued masking (default), `false` for binary masking
`mconst`	`true` when the masks are subject to a unitary sum constraint (default), `false` otherwise

Outputs:

`Se`	$J \times T$ matrix containing the near-optimal estimates of the target signals (truncated to the same time range as the original signals)
`W`	$\left(\frac{L}{2}+1\right) \times N \times J$ table containing near-optimal masks with $N = \text{ceil}\left(\frac{T}{M}\right)$
`SDR`	achieved SDR in deciBels (before truncation of the target estimates)

Reference:

See [3] Section 5.4.

`bss_nearopt_binmask`

Near-optimal constrained binary time-frequency masks for multichannel source separation using STFT with sine window (coefficients derived for each time-frequency point separately).

Syntax:

```
[Se,W,SDR]=bss_nearopt_binmask(X,S,L)
[Se,W,SDR]=bss_nearopt_binmask(X,S,L,M)
```

Inputs:

`X`	$I \times T$ matrix containing the multichannel mixture signal
`S`	$I \times T \times J$ table containing the target signals (source images)
`L`	length of the STFT window in samples (must be a multiple of 4)
`M`	step between successive windows in samples (must be a multiple of 2, a divider of L and smaller than $\frac{L}{2}$) (default: $\frac{L}{2}$)

Outputs:

`Se`	$I \times T \times J$ table containing the near-optimal estimates of the target signals (truncated to the same time range as the original signals)
`W`	$\left(\frac{L}{2}+1\right) \times N \times J$ table containing near-optimal masks with $N = \text{ceil}\left(\frac{T}{M}\right)$
`SDR`	achieved SDR in deciBels (before truncation of the target estimates)

Reference:

See [3] Section 6.4.

`bss_nearopt_pinvmask`

Near-optimal estimator for multichannel source separation of possibly convolutive mixtures by time-frequency masking and mixing matrix pseudo-inversion using STFT with sine window (activity patterns derived for each time-frequency point separately).

Syntax:

```
[Se,W,SDR]=bss_nearopt_pinvmask(X,S,A)
[Se,W,SDR]=bss_nearopt_pinvmask(X,S,A,Ja)
[Se,W,SDR]=bss_nearopt_pinvmask(X,S,A,Ja,M)
```

Inputs:

`X`	$I \times T$ matrix containing the multichannel mixture signal
`S`	$I \times T \times J$ table containing the target signals (source images)
`A`	$I \times J \times \left(\frac{L}{2}+1\right)$ table containing complex mixing matrices for positive frequencies (may be different from the ones actually used to generate S), with L being the length of the STFT window in samples
`Ja`	number of active sources per time-frequency point (by default or if $Ja = 0$, the best number is estimated for each time-frequency point)
`M`	step between successive windows in samples (must be a multiple of 2, a divider of L and smaller than $\frac{L}{2}$) (default: $\frac{L}{2}$)

Outputs:

`Se`	$I \times T \times J$ table containing the near-optimal estimates of the target signals (truncated to the same time range as the original signals)
`W`	$\left(\frac{L}{2}+1\right) \times N \times J$ table of binary coefficients indicating the oracle source activity patterns with $N = \text{ceil}\left(\frac{T}{M}\right)$
`SDR`	achieved SDR in deciBels (before truncation of the target estimates)

Reference:

See [3] Section 6.4.

3.4 Time-frequency transforms

`mdct`

Modified Discrete Cosine Transform using a sine window.

Syntax:

```
X=mdct(x,L)
```

Inputs:

`x`	$1 \times T$ vector containing a single-channel signal
`L`	length of the MDCT window in samples (must be a multiple of 4)

Output:

`X`	$\frac{L}{2} \times N$ matrix containing the MDCT coefficients with $N = \text{ceil}\left(\frac{2T}{L}\right)$

`imdct`

Inverse Modified Discrete Cosine Transform using a sine window.

Syntax:

`x=imdct(X)`

Input:

`X`	$\frac{L}{2} \times N$ matrix containing a set of MDCT coefficients

Outputs:

`x`	$1 \times \frac{NL}{2}$ vector containing the inverse MDCT signal

If `x` is a signal of length `T`, `X=mdct(x,L)` and `y=imdct(X)`, then `x=y(1:T)`.

stft

Short-Term Fourier Transform using a sine window.

Syntax:

```
X=stft(x,L)
X=stft(x,L,M)
```

Inputs:

`x`	$1 \times T$ vector containing a single-channel signal
`L`	length of the STFT window in samples (must be a multiple of 4)
`M`	step between successive windows in samples (must be a multiple of 2, a divider of L and smaller than $\frac{L}{2}$) (default: $\frac{L}{2}$)

Output:

`X`	$\left(\frac{L}{2}+1\right) \times N$ matrix containing the STFT coefficients for positive frequencies with $N = \text{ceil}\left(\frac{T}{M}\right)$

istft

Inverse Short-Term Fourier Transform using a sine window.

Syntax:

```
x=istft(X)
x=istft(X,M)
```

Inputs:

`X`	$\left(\frac{L}{2}+1\right)\times N$ matrix containing a set of STFT coefficients for positive frequencies
`M`	step between successive windows in samples (must be a multiple of 2, a divider of L and smaller than $\frac{L}{2}$) (default: $\frac{L}{2}$)

Output:

`x`	$1 \times NM$ vector containing the inverse STFT signal

If `x` is a signal of length `T`, `X=stft(x,L)` and `y=istft(X)`, then `x=y(1:T)`.

3.5 Filtering and masking functions

`apply_multifilt_temp`

Apply time-domain demixing filters.

Syntax:

```
Se=apply_multifilt_temp(X,W)
```

Inputs:

`X`	$I \times T$ matrix containing the multichannel mixture signal
`W`	$J \times I \times L$ table containing the coefficients of the demixing filters (delays from $-\frac{L}{2}+1$ to $\frac{L}{2}$)

Output:

`Se`	$J \times (T+L-1)$ matrix containing the demixed signals

Reference:

See [3] Section 4.1.

`apply_multifilt_freq`

Apply frequency-domain demixing matrices using STFT with sine window.

Syntax:

```
Se=apply_multifilt_freq(X,W)
Se=apply_multifilt_freq(X,W,M)
```

Inputs:

`X`	$I \times T$ matrix containing the multichannel mixture signal
`W`	$J \times I \times \left(\frac{L}{2}+1\right)$ table containing complex demixing matrices for positive frequencies, with L being the length of the STFT window in samples
`M`	step between successive windows in samples (must be a multiple of 2, a divider of L and smaller than $\frac{L}{2}$) (default: $\frac{L}{2}$)

Output:

`Se`	$J \times NM$ matrix containing the demixed signals with $N = \text{ceil}\left(\frac{T}{M}\right)$

Reference:

See [3] Section 4.4.

`apply_pinvmask_inst`

Apply multichannel time-frequency masks with mixing matrix pseudo-inversion using MDCT with sine window.

Syntax:

```
Se=apply_pinvmask_inst(X,A,W)
```

Inputs:

`X`	$I \times T$ matrix containing the multichannel mixture signal
`A`	$I \times J$ real-valued mixing matrix
`W`	$\frac{L}{2} \times N \times J$ table of binary coefficients indicating the source activity patterns with L being the length of the MDCT window in samples and $N = \text{ceil}\left(\frac{2T}{L}\right)$

Output:

`Se`	$J \times \frac{NL}{2} \times J$ table containing the derived source images

Reference:

See [3] Section 6.1.2.

`apply_pinvmask_conv`

Apply multichannel time-frequency masks with mixing matrix pseudo-inversion using STFT with sine window.

Syntax:

```
Se=apply_pinvmask_conv(X,A,W)
Se=apply_pinvmask_conv(X,A,W,M)
```

Inputs:

`X`	$I \times T$ matrix containing the multichannel mixture signal
`A`	$I \times J \times \left(\frac{L}{2}+1\right)$ table containing complex mixing matrices for positive frequencies, with L being the length of the STFT window in samples
`W`	$\frac{L}{2} \times N \times J$ table of binary coefficients indicating the source activity patterns with $N = \text{ceil}\left(\frac{2T}{L}\right)$
`M`	step between successive windows in samples (must be a multiple of 2, a divider of L and smaller than $\frac{L}{2}$) (default: $\frac{L}{2}$)

Output:

`Se`	$J \times NM \times J$ table containing the derived source images

Reference:

See [3] Section 6.4.

3.6 Auxiliary functions

`optim_coeffs`

Oracle constrained real-valued masking coefficients for a single basis element.

Syntax:

```
[wo,disto]=optim_coeffs(r)
```

Input:

`r`	$J \times 1$ vector containing ratios between MDCT, CP or WP coefficients or real parts of ratios between STFT coefficients of the targets and the mixture for a single basis element

Outputs:

`wo`	oracle masking coefficients for this basis element
`disto`	achieved distortion

Reference:

See [3] Section 5.2.

`pinv_filt`

Pseudo-inversion of a filter system.

Syntax:

```
[W,B,SIR]=pinv_filt(A,zdel,L)
```

Inputs:

`A`	$I \times J \times T$ table containing filters of length T (delays from $-zdel+1$ to $T-zdel$)
`zdel`	sample index corresponding to zero delay
`L`	length of the pseudo-inverse filters in samples

Outputs:

`W`	$J \times I \times L$ table containing pseudo-inverse filters (delays from $-\frac{L}{2}+1$ to $\frac{L}{2}$)
`B`	$J \times J \times (T+L-1)$ product of W and A (delays from $-zdel-\frac{L}{2}+2$ to $T-zdel+\frac{L}{2}$)
`SIR`	achieved SIR in deciBels

Reference:

See [2] Section 4.3.3.

sdr

Signal to Distortion Ratio.

Syntax:

```
SDR=sdr(Se,S)
```

Inputs:

`Se`	$J \times T$ matrix containing the estimated signals
`S`	$J \times T$ matrix containing the target signals

Output:

`SDR`	achieved SDR in deciBels

Reference:

See [3] Section 2.3.

Chapter 4

Example data and applications

The full version of BSS Oracle contains example sources, filters and scripts that were used to plot the figures of the reference publications.

4.1 Sources and filters

`data/mix`k`_s`j`.wav`	source j of mixture k, with $1 \leq j \leq 3$ (music for $1 \leq k \leq 10$, speech for $11 \leq k \leq 20$)
`data/ir_`t`.mat`	mixing impulse responses with reverberation time t equal to `anechoic`, `50ms`, `250ms` or `1.25s` with source 1 at -40°
`data/ir_move1_250ms.mat`	mixing impulse responses with 250 ms reverberation time with source 1 at -38°
`data/ir_move2_250ms.mat`	mixing impulse responses with 250 ms reverberation time with source 1 at -36°
`data/ir_move3_250ms.mat`	mixing impulse responses with 250 ms reverberation time with source 1 at -32°

See [3] Section 3.

4.2 Scripts

`multifilt1.m`	plots figure 2 of [3]
`multifilt2.m`	plots figure 3 of [3]
`multifilt3.m`	plots figure 4 of [2]
`multifilt4.m`	plots figure 4 of [3]
`monomask1.m`	plots figure 5 of [3]
`monomask2.m`	plots figure 7 of [2]
`monomask3.m`	plots figure 6 of [3]
`multimask1.m`	plots figure 7 of [3]
`multimask2.m`	plots figure 8 of [3]
`bbasis_monomask1.m`	plots figure 11 of [2]
`bbasis_monomask2.m`	plots figure 12 of [2]
`bbasis_monomask3.m`	plots figures 13 and 14 of [2]
`robust1.m`	plots figure 9 of [3]
`robust2.m`	plots figure 10 of [3]

Bibliography

[1] E. Vincent and R. Gribonval. Blind criterion and oracle bound for instantaneous audio source separation using adaptive time-frequency representations. In Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA), 2007.

[2] E. Vincent, R. Gribonval, and M.D. Plumbley. Oracle estimators for the benchmarking of source separation algorithms. Technical Report C4DM-TR-06-03, Queen Mary, University of London, 2006.

[3] E. Vincent, R. Gribonval, and M.D. Plumbley. Oracle estimators for the benchmarking of source separation algorithms. Signal Processing, 87(8):1933–1950, 2007.