Flexible Audio Source Separation Toolbox (FASST)  
Version 1.0  
User Guide  

Alexey Ozerov 1, Emmanuel Vincent 1 and Frédéric Bimbot 2  

1INRIA, Centre de Rennes - Bretagne Atlantique  
2 IRISA, CNRS - UMR 6074  
Campus de Beaulieu, 35042 Rennes cedex, France  
{alexey.ozerov, emmanuel.vincent}@inria.fr, frederic.bimbot@irisa.fr  

April 7, 2011  

1 Introduction  

This user guide describes how to use FASST, an implementation of the general flexible source separation framework presented in [1]. Before reading the user guide you are strongly encouraged to read [1], at least the two first sections.  

This guide is organized as follows. Some notations and abbreviations used throughout this document are listed in section 2. Section 3 gives a detailed specification of the mixture structure (a Matlab structure), used to define the available prior information. The main functions the user should know about are listed in section 4 and an example of usage is given in section 5.  

2 Some abbreviations and notations  

2.1 Abbreviations  

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>GMM</td>
<td>Gaussian mixture model</td>
<td></td>
</tr>
<tr>
<td>GSMM</td>
<td>Gaussian Scaled Mixture Model</td>
<td></td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Markov Model</td>
<td></td>
</tr>
<tr>
<td>NMF</td>
<td>Nonnegative matrix factorization</td>
<td></td>
</tr>
<tr>
<td>PSD</td>
<td>Power Spectral Density</td>
<td></td>
</tr>
<tr>
<td>QERB</td>
<td>Quadratic Equivalent Rectangular Bandwidth transform</td>
<td></td>
</tr>
<tr>
<td>S-HMM</td>
<td>Scaled Hidden Markov Model</td>
<td></td>
</tr>
<tr>
<td>STFT</td>
<td>Short-Time Fourier Transform</td>
<td></td>
</tr>
</tbody>
</table>
### 2.2 Notations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F$</td>
<td>Number of frequency bins in the corresponding time-frequency representation</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of time frames in the corresponding time-frequency representation</td>
</tr>
<tr>
<td>$I$</td>
<td>Number of channels (this version is only implemented for $I = 1$ or $I = 2$)</td>
</tr>
<tr>
<td>$J_{\text{spat}}$</td>
<td>Number of spatial components (see Section 3)</td>
</tr>
<tr>
<td>$J_{\text{spec}}$</td>
<td>Number of spectral components (see Section 3)</td>
</tr>
<tr>
<td>$R_j$</td>
<td>Rank of the covariance matrix of the $j$-th spatial component</td>
</tr>
<tr>
<td>$C_j$</td>
<td>Number of factors in the $j$-th spectral component</td>
</tr>
<tr>
<td>$C_j$</td>
<td>Number of factors in the $j$-th spectral component (see [1])</td>
</tr>
<tr>
<td>$L_{ex}^j$</td>
<td>Number of narrowband excitation spectral patterns (see [1]) in the $j$-th spec. comp.</td>
</tr>
<tr>
<td>$K_{ex}^j$</td>
<td>Number of characteristic excitation spectral patterns (see [1]) in the $j$-th spec. comp.</td>
</tr>
<tr>
<td>$M_{ex}^j$</td>
<td>Number of time-localized excitation patterns (see [1]) in the $j$-th spec. comp.</td>
</tr>
<tr>
<td>$L_{ft}^j$</td>
<td>Number of narrowband filter spectral patterns (see [1]) in the $j$-th spec. comp.</td>
</tr>
<tr>
<td>$K_{ft}^j$</td>
<td>Number of characteristic filter spectral patterns (see [1]) in the $j$-th spec. comp.</td>
</tr>
<tr>
<td>$M_{ft}^j$</td>
<td>Number of time-localized filter patterns (see [1]) in the $j$-th spec. comp.</td>
</tr>
<tr>
<td>$A_j$</td>
<td>Mixing parameters ($\in C^I \times R_j \times F \times N$) in the $j$-th spatial comp. (see [1]),</td>
</tr>
<tr>
<td>$W_{ex}^j$</td>
<td>Narrowband excitation spectral patterns ($\in \mathbb{R}^{F \times L_{ex}^j}$) in the $j$-th spec. comp. (see [1]),</td>
</tr>
<tr>
<td>$U_{ex}^j$</td>
<td>Excitation spectral pattern weights ($\in \mathbb{R}^{L_{ex}^j \times K_{ex}^j}$) in the $j$-th spec. comp. (see [1]),</td>
</tr>
<tr>
<td>$G_{ex}^j$</td>
<td>Excitation time pattern weights ($\in \mathbb{R}^{K_{ex}^j \times M_{ex}^j}$) in the $j$-th spec. comp. (see [1]),</td>
</tr>
<tr>
<td>$H_{ex}^j$</td>
<td>Time-localized excitation patterns ($\in \mathbb{R}^{M_{ex}^j \times N}$) in the $j$-th spec. comp. (see [1]),</td>
</tr>
<tr>
<td>$W_{ft}^j$</td>
<td>Narrowband filter spectral patterns ($\in \mathbb{R}^{F \times L_{ft}^j}$) in the $j$-th spec. comp. (see [1]),</td>
</tr>
<tr>
<td>$U_{ft}^j$</td>
<td>Filter spectral pattern weights ($\in \mathbb{R}^{L_{ft}^j \times K_{ft}^j}$) in the $j$-th spec. comp. (see [1]),</td>
</tr>
<tr>
<td>$G_{ft}^j$</td>
<td>Filter time pattern weights ($\in \mathbb{R}^{K_{ft}^j \times M_{ft}^j}$) in the $j$-th spec. comp. (see [1]),</td>
</tr>
<tr>
<td>$H_{ft}^j$</td>
<td>Time-localized filter patterns ($\in \mathbb{R}^{M_{ft}^j \times N}$) in the $j$-th spec. comp. (see [1]),</td>
</tr>
<tr>
<td>$\mathbb{R}$</td>
<td>Set of real numbers</td>
</tr>
<tr>
<td>$\mathbb{R}_+$</td>
<td>Set of nonnegative real numbers</td>
</tr>
<tr>
<td>$\mathbb{C}$</td>
<td>Set of complex numbers</td>
</tr>
</tbody>
</table>

### 3 Mixture structure

The mixture structure is a Matlab structure that is used to incorporate prior information into the framework. The structure has a hierarchical organization that can be seen from the example in figure 1. Global parameters (e.g., signal representation) are defined on the first level of the hierarchy. The second level consists of $J_{\text{spat}}$ spatial components and $J_{\text{spec}}$ spectral components. Each source is typically modeled by one spectral component, although some sources (e.g., drums) might be modeled by several spectral components (e.g., bass drum, snare, etc.). Furthermore, each spectral component must be associated with one spatial component, and each spatial component must have at least one spectral component associated to it. $^1$ Compared to the description of the framework in [1], this implementation is more general in the sense that the number of spectral components is

$^1$This extension makes it possible to model the fact that several sources have the same direction, which is very often the case for professionally produced music recordings. It is implemented by simply adding the power spectrograms of the spectral components corresponding to the same spatial component.
not necessarily equal to that of spatial components, and more precisely $J_{\text{spec}} \geq J_{\text{spat}}$. The third level of the hierarchy consists in factorizing each spectral component into one or more factors representing for instance excitation and filter structures (see [1])\(^2\). Finally, on the fourth level of the hierarchy, each factor is represented as the product of three or four matrices (see Table 4), which are not represented in Figure 1. For instance, the factor representing excitation structure is either represented as the product of four matrices $W_{\text{ex}}^j U_{\text{ex}}^j G_{\text{ex}}^j H_{\text{ex}}^j$ representing, respectively, narrowband spectral patterns, spectral pattern weights, time pattern weights and time-localized patterns (see [1]) or as the product of three matrices $W_{\text{ex}}^j U_{\text{ex}}^j G_{\text{ex}}^j$ when $H_{\text{ex}}^j$ is marked by the empty matrix $\emptyset$. Almost all the fields of the mixture structure must be filled as specified in Tables 1, 2, 3 and 4, except those marked by the empty matrix $\emptyset$.

Figure 1: Visualization of a mixture structure example.

### 4 Main functions

The user should know about three main functions `comp_transf_Cx`, `estim_param_a_post_model` and `separate_spec_comps`, allowing, respectively, to compute the input time-frequency transform, estimate the model parameters and separate the spectral components. The headers of these functions are listed in Figures 2, 3 and 4.

\(^2\)Note that in [1] the usage of two factors (excitation and filter) is described. The implementation presented here is more flexible, since one can use any number of factors $C_j$, and it reduces to [1] when $C_j = 2$. This is done for convenience of usage. For example if one needs to implement an excitation model only or a filter model only (**direct model**), one simply needs to choose $C_j = 1$ without bothering to specify and to process an additional dummy factor.

\(^3\)In [1] only the case of four matrices is considered, and the case of three matrices $W_{\text{ex}}^j U_{\text{ex}}^j G_{\text{ex}}^j$ is just equivalent to fixing $H_{\text{ex}}^j$ to the $N \times N$ identity matrix. Since $N$ may be quite big, we fix $H_{\text{ex}}^j$ to $\emptyset$ by convention in the latter case in order to avoid storing a big identity matrix in memory.
5 Examples of usage

The user should also know how to fill and browse the mixture structure and how to use the above-mentioned three functions. An example of mixture structure filling and browsing is given in Figures 5 and 6. An example script for the separation of an instantaneous mixture of music signals is given in Figure 7.

Function EXAMPLE_prof_rec_sep_drums_bass_melody.m contains a more sophisticated example allowing the separation of the following four sources:

- drums,
- bass,
- melody (singing voice or leading melodic instrument),
- remaining sounds,

from a stereo music recording. Due to memory limits in Matlab this function cannot process sound excerpts longer than 30 seconds. For full length music recording the function EXAMPLE_prof_rec_sep_drums_bass_melody_FULL.m should be used. This function simply cuts the full recording into small parts, and applies EXAMPLE_prof_rec_sep_drums_bass_melody.m to each of them.

References


Table 1: Specification of the mixture structure (mix_str).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cx</td>
<td>$F \times N \times I \times I$ complex-valued tensor of local mixture covariances</td>
<td>$\in \mathbb{C}^{F \times N \times I \times I}$</td>
</tr>
<tr>
<td>transf</td>
<td>Input time-frequency transform</td>
<td>'stft' for STFT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'qerb' for QERB</td>
</tr>
<tr>
<td>fs</td>
<td>Sampling frequency in Hz</td>
<td>$\in {16000, 44100, \ldots}$</td>
</tr>
<tr>
<td>wlen</td>
<td>Analysis window length (used to compute STFT or QERB) in samples</td>
<td>$\in {512, 1024, \ldots}$</td>
</tr>
<tr>
<td>Noise_PSD</td>
<td>$F \times 1$ real-valued nonnegative vector of additive noise PSD, e.g., for annealing</td>
<td>$\in \mathbb{R}^{1 \times F}$ or $[\ ]$</td>
</tr>
<tr>
<td>spat_comps</td>
<td>$1 \times J_{spat}$ cell array of spatial component structures</td>
<td>see Table 2</td>
</tr>
<tr>
<td>spec_comps</td>
<td>$1 \times J_{spec}$ cell array of spectral component structures</td>
<td>see Table 3</td>
</tr>
</tbody>
</table>

Table 2: Specification of the spatial component structure (spat_comps$\{j\}, j = 1, \ldots, J_{spat}$).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>time_dep</td>
<td>Stationarity of mixing</td>
<td>'indep' for time-invariant mixing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'dep' for time-varying mixing</td>
</tr>
<tr>
<td>mix_type</td>
<td>Mixing type</td>
<td>'inst' for instantaneous (freq.-indep.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'conv' for convolutive (freq.-dep.)</td>
</tr>
<tr>
<td>frdm_prior</td>
<td>Degree of adaptability</td>
<td>'free' for adaptive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'fixed' for fixed</td>
</tr>
<tr>
<td>params</td>
<td>Tensor of mixing parameters</td>
<td>$\in \mathbb{R}^{I \times R_j}$ for mix_type = 'inst'</td>
</tr>
<tr>
<td></td>
<td>(corresponding to $A_j$ from [1])</td>
<td>$\in \mathbb{C}^{I \times R_j \times F}$ for mix_type = 'conv'</td>
</tr>
</tbody>
</table>

Table 3: Specification of the spectral component structure (spec_comps$\{j\}, j = 1, \ldots, J_{spec}$).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>spat_comp_ind</td>
<td>Index of the corresponding spatial component</td>
<td>$\in {1, \ldots, J_{spat}}$</td>
</tr>
<tr>
<td>factors</td>
<td>cell array of factor structures</td>
<td>$1 \times L_j$</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Value</td>
</tr>
<tr>
<td>--------------</td>
<td>----------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>FB_frdm_prior</td>
<td>Degree of adaptability for narrowband spectral patterns</td>
<td>'free' for adaptive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'fixed' for fixed</td>
</tr>
<tr>
<td>FW_frdm_prior</td>
<td>Degree of adaptability for spectral pattern weights</td>
<td>'free' for adaptive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'fixed' for fixed</td>
</tr>
<tr>
<td>TW_frdm_prior</td>
<td>Degree of adaptability for time pattern weights</td>
<td>'free' for adaptive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'fixed' for fixed</td>
</tr>
<tr>
<td>TB_frdm_prior</td>
<td>Degree of adaptability for time-localized patterns</td>
<td>'free' for adaptive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'fixed' for fixed</td>
</tr>
<tr>
<td>FB</td>
<td>Narrowband spectral patterns (Frequency Blobs) (corresponding to $W_j^x$ or $W_j^y$)</td>
<td>$\mathbb{R}^{F \times L_j^x}$ or $\mathbb{R}^{F \times L_j^y}$</td>
</tr>
<tr>
<td>FW</td>
<td>Spectral pattern weights (Frequency Weights) (corresponding to $U_j^x$ or $U_j^y$)</td>
<td>$\mathbb{R}^{L_j^x \times K_j^x}$ or $\mathbb{R}^{L_j^y \times K_j^y}$</td>
</tr>
<tr>
<td>TW</td>
<td>Time pattern weights (Time Weights) (corresponding to $G_j^x$ or $G_j^y$)</td>
<td>$\mathbb{R}^{K_j^x \times M_j^x}$ or $\mathbb{R}^{K_j^y \times M_j^y}$</td>
</tr>
<tr>
<td>TB</td>
<td>Time-localized patterns (Time Blobs) (corresponding to $H_j^x$ or $H_j^y$)</td>
<td>$\mathbb{R}^{M_j^x \times N}$ or $\mathbb{R}^{M_j^y \times N}$ or $\emptyset$</td>
</tr>
<tr>
<td>TW_constr</td>
<td>Constraint on the time pattern weights</td>
<td>'NMF' no constraint</td>
</tr>
<tr>
<td></td>
<td>(note that nontrivial constraints, i.e., different from 'NMF' are not compatible with nonempty time patterns TB)</td>
<td>'GMM' for GMM, 'HMM' for HMM, 'GSMM' for GSMM, 'S-HMM' for S-HMM</td>
</tr>
<tr>
<td>TW_DP_params</td>
<td>Discrete probability (DP) parameters for the time pattern weights (needed only when $TW\text{_constr} \neq \text{'NMF'}$)</td>
<td>$1 \times K_j^x$ ($1 \times K_j^y$) vector of Gaussian weights for GMM or GSMM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$K_j^x \times K_j^x$ ($K_j^y \times K_j^y$) matrix of transition probabilities for HMM or S-HMM</td>
</tr>
<tr>
<td>TW_DP_frdm_prior</td>
<td>Degree of adaptability for DP parameters (needed only when $TW\text{_constr} \neq \text{'NMF'}$)</td>
<td>'free' for adaptive</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'fixed' for fixed</td>
</tr>
<tr>
<td>TW_all</td>
<td>Matrix of all time weights (corresponding to $\tilde{G}_j^x$ or $\tilde{G}_j^y$ from [1]) (needed only when $TW\text{_constr} \neq \text{'NMF'}$)</td>
<td>Nonnegative real-valued matrix of the same size as $TW$</td>
</tr>
</tbody>
</table>

Table 4: Specification of the spectral component factor structure ($\text{factors}(l), l = 1, \ldots, L_j$).
function Cx = comp_transf_Cx(x, transf, win_len, fs, qerb_nbin)

% Cx = comp_transf_Cx(x, transf, win_len, fs, qerb_nbin);
% compute spatial covariance matrices for the corresponding transform
% input
% x : [I x nsampl] matrix containing I time-domain mixture signals
% with nsampl samples
% transf : transform
%   'stft'
% % 'qerb'
% win_len : window length
% fs : (opt) sampling frequency (Hz)
% qerb_nbin : (opt) number of bins for qerb transform
% output
% Cx : [F x N x I x I] matrix containing the spatial covariance
% matrices of the input signal in all time-frequency bins

Figure 2: **comp_transf_Cx**: FASST function for the computation of the input time-frequency transform.

function [mix_str, log_like_arr] = estim_param_a_post_model(mix_str_inp, ...
 iter_num, sim_ann_opt, Ann_PSD_beg, Ann_PSD_end)

% [mix_str, log_like_arr] = estim_param_a_post_model(mix_str_inp, ...
% iter_num, sim_ann_opt, Ann_PSD_beg, Ann_PSD_end);
% estimate a posteriori mixture model parameters
% input
% ______
% mix_str_inp : input mixture structure
% iter_num : (opt) number of EM iterations (def = 100)
% sim_ann_opt : (opt) simulated annealing option (def = 'ann')
%   'no_ann' : no annealing (zero noise)
%   'ann' : annealing
%   'ann_ns_inj' : annealing with noise injection
%   'upd_ws_prm' : update noise parameters
% Ann_PSD_beg : (opt) [F x 1] beginning vector of annealing noise PSD
%   (def = X_power / 100)
% Ann_PSD_end : (opt) [F x 1] end vector of annealing noise PSD
%   (def = X_power / 10000)
% output
% ______
% mix_str : estimated output mixture structure
% log_like_arr : array of log-likelihoods

Figure 3: **estim_param_a_post_model**: FASST function for the estimation of the model parameters.
function ie = separate_spec_comps(x, mix_str, sep_cmp_inds)

% ie = separate_spec_comps(x, mix_str, sep_cmp_inds);
% separate spectral components
%
% input
%
% x : [nchan x nsampl] mixture signal
% mix_str : input mix structure
% sep_cmp_inds : (opt) array of indices for components to separate
%   (def = [1, 2, ..., K_spec])
%
% output
%
% ie : [K_sep x nsampl x nchan] estimated spectral components images,
%   where K_sep = length(sep_cmp_inds) is the number of
%   components to separate
%

Figure 4: separate_spec_comps: FASST function for the separation of the spectral component signals.
function mix_str = init_mix_struct_Mult_BMF_inst(Cx, J, K, transf, fs, wlen)

% An example of mixture structure initialization, corresponding to
% multichannel NMF model (instantaneous case)
% Most of parameters are initialized randomly

% input
% Cx : [F x N x 1 x 1] matrix containing the spatial covariance
% matrices of the input signal in all time-frequency bins
% J : number of components (here J_spats = J_spct)
% K : number of NMF components per source
% transf : transform ('stft' or 'qerb')
% fs : sampling frequency in Hz
% wlen : length of the time integration window (must be a power of 2)

% output
% mix.str : initialized mixture structure

rank = 1;

[F, N, I, I] = size(Cx);
mix_str.Cx = Cx;
mix_str.transf = transf;
mix_str.fs = fs;
mix_str.wlen = wlen;
mix_str.spat_comps = cell(1,J);
mix_str.spec_comps = cell(1,J);

for j = 1:J
    % initialize spatial component
    mix_str.spat_comps{j}.time_dep = 'indep';
mix_str.spat_comps{j}.type = 'inst';
mix_str.spat_comps{j}.frdms_prior = 'free';
mix_str.spat_comps{j}.params = randn(1, rank);
    % initialize single factor spectral component
    mix_str.spec_comps{j}.spat_comp_ind = j;
mix_str.spec_comps{j}.factors = cell(1, 1);
    factor1.FB = 0.75 * abs(randn(F, K)) + 0.25 * ones(F, K);
    factor1.FW = diag(ones(1, K));
    factor1.TW = 0.75 * abs(randn(K, N)) + 0.25 * ones(K, N);
    factor1.TB = [];
    factor1.FB_frdm_prior = 'free';
    factor1.FW_frdm_prior = 'fixed';
    factor1.TW_frdm_prior = 'free';
    factor1.TB_frdm_prior = [];
    factor1.TW_tree = 'NMF';
mix_str.spec_comps{j}.factors{1} = factor1;
end

Figure 5: Example of filling of the mixture structure corresponding to the multichannel NMF method [2] (instantaneous case).
>> mix_str

mix_str =

Cx: [4-D double]
transf: 'stft'
fs: 16000
wlen: 1024
spat_comps: {{1x1 struct} [1x1 struct] [1x1 struct]}
spec_comps: {{1x1 struct} [1x1 struct] [1x1 struct]}
Noise_PSD: [5x1 double]

>> mix_str.spat_comps{2}

ans =

time_dep: 'indep'
mix_type: 'inst'
frdm_prior: 'free'
params: [2x1 double]

>> mix_str.spec_comps{3}

ans =

spat_comp_ind: 3
factors: {{1x1 struct}}

>> mix_str.spec_comps{3}.factors{1}

ans =

FB: [513x4 double]
FW: [4x4 double]
TW: [4x98 double]
TB: []
FB_frdm_prior: 'free'
FW_frdm_prior: 'fixed'
TW_frdm_prior: 'free'
TB_frdm_prior: []
TW_constr: 'NMF'

Figure 6: Browsing in Matlab of the example mixture structure in Table 5.
function EXAMPLE_ssep_Mult_NMF_inst()

data_dir = 'example_data/';
result_dir = 'example_data/';
file_prefix = 'Shannon_Hurley_Sunrise_inst.';
transf = 'stft';
wlen = 1024;
nsrc = 3;  \% number of sources
NMF_ncomp = 4;  \% number of NMF components
iter_num = 200;

% load mixture
[x, fs, nbins] = wavread(fullfile(data_dir, file_prefix, 'mix.wav'));
x = x(:, 1);
mix_nsamp = size(x, 2);
% compute time-frequency representation
Cx = comp_transf_Cx(x, transf, wlen, fs);
% fill in mixture structure
mix_str = init_mix_struct_Mult_NMF_inst(Cx, nsrc, NMF_ncomp, transf, fs, wlen);
% reinitialize mixing parameters
A = [sin(pi/8), sin(pi/4), sin(3*pi/8); cos(pi/8), cos(pi/4), cos(3*pi/8)];
for j = 1:nsrc
    mix_str.spat_comps{j}.params = A(:, j);
end;
% run parameters estimation (with simulated annealing)
mix_str = estin_param_a_post_model(mix_str, iter_num, 'ann');
% source separation
ie_EM = separate_spat_comps(x, mix_str);
% Computation of the spatial source images
fprintf('Computation of the spatial source images\n');
for j = 1:nsrc,
    wavwrite(reshape(ie_EM(j, :, :), mix_nsamp, 2), fs, nbins, ...
        [result_dir, file_prefix, '_sim_' int2str(j) '.wav']);
end

Figure 7: Example of usage involving all three main functions (runs the multichannel NMF method