Adaptive Blind System Identification and Equalization Toolbox for MATLAB

Version 1.2

Emanuël A.P. Habets, Xiang (Shawn) Lin, Wancheng Zhang,
Andy Khong, Nikolay D. Gaubitch, Patrick A. Naylor

March 29, 2011

Contents

Abstract

This report provides an introduction to blind system identification and system equalization, and describes a MATLAB toolbox that contains state-of-the-art blind system identification and system equalization algorithms, performance evaluation methods and analyses tools. In this report, we focus on single-input multiple-output (SIMO) acoustic systems.

In the first part of this report, background information on blind system identification and system equalization is provided. Here we focus on blind system identification methods that jointly minimize the cross-relation error between different pairs of sensor signals. In addition, it is shown that an estimate of the (unknown) source signal can be obtained by equalizing the acoustic system. In the second part of this report, a variety of adaptive blind system identification algorithms are described that minimize the total cross-relation error, either in the time or frequency domain. In the third part of this report, varies algorithms are described that can be used to compute the equalization system given the estimated impulse responses of the acoustic system. Finally, we describe performance evaluation methods and analysis tools that are commonly used in this context.

1 Introduction

1.1 Literature Overview

The idea of blind system identification (BSI) was first introduced to the communications community by Sato with the intention of designing efficient communication systems that did not require a training phase [1]. From that point on, BSI has become an extremely active topic of research, and has been widely applied to various areas such as communications [2], geophysics, underwater communications [3], and multimedia signal processing [**?**, 4]. In contrast to non-blind system identification where a known signal is used to facilitate the identification of the acoustic channel, BSI algorithms only utilize signals received from the output of an unknown system for channel estimation. For the case of speech dereverberation, acoustic channels are first identified blindly using received signals from the microphones. These estimated channels are then used to design equalization filters in order to remove reverberation induced by the acoustic channels.

BSI methods can generally be classified into two main categories: second order statistical (SOS) and higher order statistical (HOS) methods. Comparisons between SOS and HOS methods have been presented in [5]. The HOS information of a stochastic process can be described by its kth-order cumulant or its Fourier transform; for $k > 2$ is also known as a polyspectrum [6, 7]. The HOS information of the system output signals contains the phase information of a non-Gaussian process and can be exploited by the HOS-based algorithms either directly, using the polyspectra method [8], or indirectly, using the Bussgang algorithm proposed in [9, 10]. Because HOS information cannot be accurately computed from a small number of observations, slow convergence is the critical drawback of all existing HOS methods [11]. In addition, a cost function based on HOS information is, in general, not convex. As a consequence an HOS-based algorithm can converge to a local minimum. In addition, their overall performance is degraded by corrupting noise in the observations.

Fortunately, these problems can be overcome in multichannel systems such as single-input multiple-output (SIMO) and multiple-input multiple-output (MIMO) systems where multiple sensors are employed [5]. Specifically, SOS information of the system outputs in the form of *cyclostationarity* can be exploited for blind identification, where cyclostationarity indicates the periodicity of the mean and autocorrelation function of a stochastic process [12]. By exploiting such periodicity, the preservation of phase information is now possible for processes that are wide-sense cyclostrationary. The use of cyclostationarity for recovering the amplitude and phase of the channel response was first recognized in [13], and was formally developed in the context of BSI in [14] based on the spatial diversity obtained from multiple channels. Since then, a large number of SOS-based multichannel BSI algorithms have been developed [5, 15], among which celebrated work includes the cross-relation (CR) method [16–20], the subspace method [21], the LP-based subspace algorithm [22], and the two-step maximum likelihood algorithm [23]. These methods utilize either the cyclostationarity of the received signals or spatio-temporal diversity of the multichannel systems to achieve successful BSI up to a non-zero arbitrary scale factor using only the SOS of the multiple outputs [?, 2]. In this report, only SOS-based BSI algorithms are considered.

Recent advances and innovations in speech communications have led to increased interest in acoustic signal processing aiming at extracting and interpreting information from acoustic signals in, for example, the cocktail party situation [11,24]. As a result, accurately identifying acoustic systems has become the main issue in the literature for the study of how acoustic signals are transmitted and distorted. This has motivated the adoption of BSI techniques into the acoustic signal processing community, where speech source separation and speech dereverberation are two important research topics. However, such adoption is not straightforward due to the substantial differences between communication systems and acoustic systems [25]. Traditionally, antenna arrays for wireless communications work in a fairly open space. Typical channel impulse responses are much shorter than acoustic impulse responses. For example, a typical office or living room exhibits reverberation time T_{60} in the order of 50 to 600 ms. At a sample frequency of 8 kHz, the acoustic impulse responses are around

400 to 4800 coefficients long. In addition, human hearing has an extremely wide dynamic range and is very sensitive to the audible effect caused by the tail of the acoustic impulse response. Undoubtedly, these facts have created great challenges for well-established classic BSI algorithms in terms of computational complexity and identification accuracy [11, 26, 27].

Nevertheless, promising progress has been made in the acoustic signal processing community as several batch and adaptive algorithms have been developed for blind identification of SIMO acoustic systems [28–32]. These algorithms can accurately determine the impulse responses of an identifiable multichannel system using a finite number of samples when additive noise in the system outputs is weak. Among these algorithms, batch algorithms are difficult to implement in an adaptive mode and are computationally expensive given the high order nature of acoustic impulse responses [27]. In addition, their performance relies on the existence of numerically well-defined dimensions of the signal or noise subspace. Adaptive algorithms, in contrast, are easier to implement and are suitable for real-time applications. They are also capable of tracking the dynamic nature of acoustic impulse responses. However, the overall performance for these algorithms is still unsatisfactory due to various issues, which can be summarized as follows:

- The order of the unknown system is often erroneously assumed to be available [30].
- Even with a small amount of noise, most algorithms cannot operate successfully.
- The computational burden due to long acoustic impulse responses is still significant even for adaptive algorithms.
- The presence of common zeros limits the accuracy and convergence speed of blind identification.

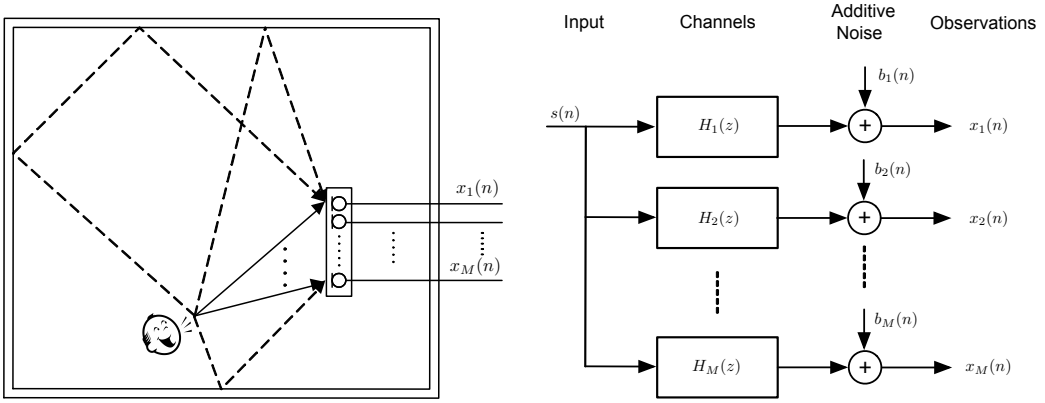
These issues have become the subjects of current research in the community [?, 5, 11, 20, 23, 33, 34], and significant contributions have been made to overcome some of them. For example, algorithms have been developed in [35–39] to improve the noise robustness of multichannel least mean squares (MCLMS) and normalized multichannel frequency domain least mean squares (NMCFLMS) algorithms [28, 29]. Additionally, system order estimation was investigated in [19, 40].

System equalization aims to recover the anechoic source signal from the sensor signals by equalizing the acoustic system. Since a single acoustic channel is generally considered as a nonminimum phase system [41], the infinite impulse response (IIR) causal inverse is unstable, which is not applicable in practice. To equalize a single acoustic channel, a finite impulse response (FIR) least squares (LS) inverse filter can be used [42]. When multiple channels are employed, it is revealed by the multiple-input/output inverse theorem (MINT) that the multichannel acoustic system can be exactly inverted by a set of FIR filters subject to some conditions [43]. Later, weighted LS inverse systems were used for room acoustics equalization [44]. More recently, channel shortening (CS) techniques [45, 46], which were firstly exploited in digital communications to mitigate inter-symbol and inter-carrier interferences, were also used for the equalization of acoustic systems [47].

1.2 Problem Formulation

The aim of BSI is to identify unknown system impulse responses excited by unknown source signals. As shown in Fig. 1, an acoustic environment containing one source and M sensors can be modelled as a linear SIMO system. Each sensor signal results from a source signals which is passed though an acoustic channel. In this report we focus on SIMO systems that can be extended to multi-input multi-output systems as shown in [11,27]. It should also be noted that although acoustic channels are inherently time-varying due to, for example, changes of source-sensor-configurations, we can use FIR filters to model acoustic channels since acoustic systems generally change slowly compared to the length of their channel impulse responses [27]. The order of the unknown system is also assumed to be known. Although algorithms such as

minimum description length (MDL) [48] or information theoretic criteria (AIC) [49] can in principle be employed for channel order estimation, they are sensitive to variations in SNR and data sample size [40].



(a) (b)

Figure 1: Diagram of (a) an M-channel SIMO acoustic system, and (b) BSI problem formulation.

For an M-channel SIMO system as shown in Fig. 1, the mth impulse response with L coefficients can be denoted as

$$\mathbf{h}_m \;=\; [h_{m,0}\; h_{m,1}\; \ldots\; h_{m,L-1}]^T, \tag{1}$$

for $m = 1, 2, \ldots, M$, and the mth microphone signal can be expressed as

$$x_m(n) \;=\; \sum_{j=0}^{L-1} h_{m,j}\, s(n-j) + b_m(n), \tag{2}$$

where $s(n)$ is the source signal and $b_m(n)$ is the additive noise. The additive noise is assumed to be zero-mean and uncorrelated with the source signal. In vector form, (2) can be written[1]

$$\mathbf{x}_m(n) \;=\; \mathbf{H}_m\, \mathbf{s}(n) + \mathbf{b}_m(n), \tag{3}$$

where $\mathbf{s}(n) = [s(n)\; s(n-1)\; \ldots\; s(n-2L+2)]^T$, $\mathbf{x}_m(n) = [x_m(n)\; x_m(n-1)\; \ldots\; x_m(n-L+1)]^T$, $\mathbf{b}_m(n) = [b_m(n)\; b_m(n-1)\; \ldots\; b_m(n-L+1)]^T$, and $\mathbf{H}_m$ is the $L \times (2L-1)$ convolution matrix for the mth channel such that

$$\mathbf{H}_m = \begin{bmatrix} h_{m,0} & \cdots & h_{m,L-1} & \cdots & \cdots & 0 \\ 0 & h_{m,0} & \cdots & h_{m,L-1} & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & h_{m,0} & \cdots & h_{m,L-1} \end{bmatrix}. \tag{4}$$

Since the impulse responses are assumed to be slowly time-varying, $\mathbf{H}_m$ is independent of n. By concatenating all M outputs of (3), a system of equations

$$\mathbf{x}(n) \;=\; \mathbf{H}\mathbf{s}(n) + \mathbf{b}(n) \tag{5}$$

can be obtained using the following quantities

$$\mathbf{x}(n) = [\mathbf{x}_1^T(n)\; \mathbf{x}_2^T(n)\; \ldots\; \mathbf{x}_M^T(n)]^T, \tag{6}$$

$$\mathbf{H} = [\mathbf{H}_1^T\; \mathbf{H}_2^T\; \ldots\; \mathbf{H}_M^T]^T, \tag{7}$$

$$\mathbf{b}(n) = [\mathbf{b}_1^T(n)\; \mathbf{b}_2^T(n)\; \ldots\; \mathbf{b}_M^T(n)]^T. \tag{8}$$

[1]Note that the elements of the signal vectors $\mathbf{x}_m(n)$, $\mathbf{s}(n)$, and $\mathbf{b}_m(n)$ are in time reversed order.

The problem of BSI is to find $\mathbf{h} = [\mathbf{h}_1^T \ \mathbf{h}_2^T \ \dots \ \mathbf{h}_M^T]^T$ using only $\mathbf{x}(n)$. This means that, with reference to Fig. 1(b), for a given output $\mathbf{x}(n)$ a unique solution to $\mathbf{h}$ should be obtained up to a non-zero scale factor across all channels. This scale factor is irrelevant in most of acoustic signal processing applications.

1.3 Channel Identifiability Conditions

Channel identifiability is concerned with the existence of a unique solution to the unknown system impulse responses with respect to a particular type of system identification algorithm [?]. According to [20], two inductive conditions are necessary and sufficient for blind identifiability of a SIMO system using BSI algorithms, which can be summarized as follows:

(C1.1) Channel diversity: The use of multisensor techniques introduces channel diversity and enables the exploration of SOS of the system outputs for blind identification of SIMO systems [15, 17]. Channel diversity in this context refers to channels being *coprime*, that is, multichannel transfer functions do not share any common zeros. If one or more common zeros exist across all channels then these channels are not coprime. As an extreme example, a SIMO system with M identical channels is of no difference to a single-channel system which exhibits no channel diversity and is thus unidentifiable using BSI algorithms. Most existing methods [20, 28–30] fail to produce a unique solution of channel estimates when common zeros exist since they cannot distinguish the common zeros due to the unknown system from ones due to the source signal.

(C1.2) Condition for the input signals: Although BSI algorithms only rely on output signals of the system, the characteristics of input signals are not negligible. An obvious requirement is that the input data should be non-zero valued. According to [20], the $L \times L$ Hankel matrix of the source signal given by

$$\mathbf{S}(n) = \begin{bmatrix} s(n) & s(n-1) & \cdots & s(n-L+1) \\ s(n-1) & s(n-2) & \cdots & s(n-L) \\ \vdots & \vdots & \ddots & \vdots \\ s(n-L+1) & s(n-L) & \cdots & s(n-2L+2) \end{bmatrix} \tag{9}$$

must be of full-rank. This can be understood by expressing,

$$\mathbf{S}(n)\mathbf{h}_m \ = \ \mathbf{x}_m(n), \qquad m = 1, 2, \dots, M \tag{10}$$

for the noiseless case, from which it can be found that if $\mathbf{S}(n)$ is rank deficient, (10) will not lead to a unique solution even if the source signal $s(n)$ is known since there are insufficient number of linear equations to solve all unknown coefficients of $\mathbf{h}_m$. In other words, a rank-deficient $\mathbf{S}(n)$ can not fully excite any SIMO system.

The channel identifiability conditions have been studied in [50–54] and extended to MIMO case in [55].

1.4 Cross-Relation Method

As one of the first proposed SOS-based multichannel BSI algorithms, the CR method [20] has served as the foundation for many subsequent algorithms. Using (2) and assuming that $b_m(n) = 0 \ \forall m$, we can derive the fact that

$$x_m(n) * h_l \ = \ s(n) * h_m * h_l \ = \ x_l(n) * h_m, \qquad m, l = 1, 2, \dots, M, \ m \neq l. \tag{11}$$

In vector form, (11) can be expressed as

$$\mathbf{x}_m^T(n)\mathbf{h}_l \ = \ \mathbf{x}_l^T(n)\mathbf{h}_m, \qquad m, l = 1, 2, \dots, M, \ m \neq l, \tag{12}$$

where h_l and h_m denote the lth and mth acoustic impulse responses, respectively. Multiplying (12) by $\mathbf{x}_m(n)$ and taking the expectation on both sides leads to

$$\mathbf{R}_{x_m x_m}\mathbf{h}_l \;=\; \mathbf{R}_{x_m x_l}\mathbf{h}_m, \qquad m,l=1,2,\ldots,M,\;\; m\neq l. \tag{13}$$

where $\mathbf{R}_{x_m x_l}$ is the cross-correlation matrix between $\mathbf{x}_m(n)$ and $\mathbf{x}_l(n)$ given by

$$\mathbf{R}_{x_m x_l} \;=\; \mathrm{E}\{\mathbf{x}_m(n)\mathbf{x}_l^T(n)\}. \tag{14}$$

The relation described by (13) results in $M(M-1)$ distinct equations. Summing (13) over $M-1$ cross-relations associated with one particular channel $\mathbf{h}_m$ results in

$$\sum_{m=1,\;m\neq l}^{M}\mathbf{R}_{x_m x_m}\mathbf{h}_l \;=\; \sum_{m=1,\;m\neq l}^{M}\mathbf{R}_{x_m x_l}\mathbf{h}_m, \quad m=1,\ldots,M. \tag{15}$$

For $m=1,\ldots,M$ we then have a total of M equations that can be expressed in matrix form as

$$\mathbf{R}\mathbf{h} \;=\; \mathbf{0}_{[ML\times 1]} \tag{16}$$

with

$$\mathbf{R} = \begin{bmatrix} \sum_{m\neq 1}\mathbf{R}_{x_m x_m} & -\mathbf{R}_{x_2 x_1} & \cdots & -\mathbf{R}_{x_M x_1} \\ -\mathbf{R}_{x_1 x_2} & \sum_{m\neq 2}\mathbf{R}_{x_m x_m} & \cdots & -\mathbf{R}_{x_M x_2} \\ \vdots & \vdots & \ddots & \vdots \\ -\mathbf{R}_{x_1 x_M} & -\mathbf{R}_{x_2 x_M} & \cdots & \sum_{m\neq M}\mathbf{R}_{x_m x_m} \end{bmatrix}_{[ML\times ML]} \tag{17}$$

$$\mathbf{h} = [\mathbf{h}_1^T,\; \mathbf{h}_2^T,\; \ldots,\; \mathbf{h}_M^T]^T, \tag{18}$$

and $\mathbf{0}_{ML\times 1}$ is a $ML\times 1$ null vector. If the channel identifiability conditions are satisfied, the rank of $\mathbf{R}$ is $ML-1$, and the solution to $\mathbf{h}$ lies within the null space of $\mathbf{R}$ [20, 56].

For the noisy cases, i.e. $b_m(n)\neq 0,\;\forall m$, (16) must be rewritten [28]

$$\mathbf{R}\mathbf{h} \;=\; \mathbf{e}. \tag{19}$$

The latter expression can be used to define a cost function

$$J = \|\mathbf{e}\|_2^2 \;=\; \mathbf{e}^T\mathbf{e}. \tag{20}$$

Correspondingly, $\widehat{\mathbf{h}}$ can be obtained by minimizing (20) in the LS sense, that is,

$$\widehat{\mathbf{h}} \;=\; \arg\min_{\mathbf{h}}\mathbf{h}^T\mathbf{R}^T\mathbf{R}\mathbf{h}, \tag{21}$$

where $\widehat{\mathbf{h}}$ can be found from eigenvectors of $\mathbf{R}$ associated with the smallest eigenvalue. It is worthwhile noting that (20) does not necessarily lead to a noise-robust solution to $\mathbf{h}$ as indicated in [36].

1.5 Hierarchy of the Toolbox

As illustrated in Fig. 2, the toolbox functions are divided into four categories:

1. Blind System Identification
   (a) Initialization
   (b) Algorithms
2. System Equalization
3. Performance Evaluation and Analyses
4. Auxiliary

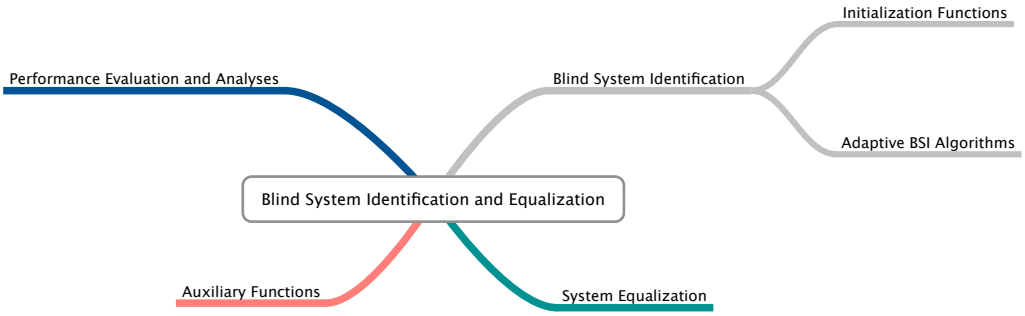


Figure 2: Structure of the BSIE toolbox for MATLAB

2 Blind System Identification

In this Section various blind system identification algorithms are described. All algorithms are based on minimizing the so-called cross-relation error [28]. Both time- and frequency domain algorithms are described including an example script. Currently, only adaptive algorithms are incorporated in the toolbox.

2.1 Multichannel LMS algorithm: mclms()

Purpose Time-domain multichannel LMS adaptive filtering algorithm for blind system identification.

Syntax

```
[h_hat, xin] = init_mclms(L, N, M, h_hat0, xin0)
```

The input and output parameters of `init_mclms()` are summarized below.

```
Input Parameters [size]:
       L       : filter length
       M       : number of channels
       xin0    : initial input matrix [L x M]
       h_hat0  : initial filter coef. matrix
                 (unit-norm constrained) [L x M]

Output parameters [size]:
       xin     : initialized signal matrix [L x M]
       h_hat   : initialized filter coef. matrix [L x M]
```

```
[h_hat, J] = mclms(xin, h_hat, mu, ss_cntr)
```

The input and output parameters of `mclms()` for an FIR adaptive filter of L coefficients are summarized below.

```
Input Parameters [size]:
       xin     : input vector [N x M]
       h_hat   : current filter coef. matrix [L x M]
       mu      : step-size
       ss_cntr : type of step-size control
                 (optional: default 'normalized'):
                 'fixed' - fixed with unit-norm-constrained;
                 'vss' - variable step-size
```

```
Output parameters [size]:
       h_hat    : updated filter coef. matrix [L x M]
       J        : value of cost function
```

Description `mclms()` is a classic time-domain multichannel adaptive algorithm (with unit-norm constraint) for blind system identification based on the cross-relation (CR) [28]. `mclms()` takes the observed microphone signals $\mathbf{x}(n)$, the length of the room impulse responses L, the step size μ, the true room impulse responses $\mathbf{h}$, and returns the estimated room impulse responses $\widehat{\mathbf{h}}(n)$ and the value of cost function at discrete time n which is given by

$$J(n) = \frac{E(n)}{\|\widehat{\mathbf{h}}(n-1)\|_2^2} \tag{22}$$

where

$$E(n) = \sum_{m=1}^{M-1} \sum_{l=m+1}^{M} e_{ml}^2(n) \tag{23}$$

is the total *a priori* error with

$$e_{ml}(n) = \mathbf{x}_m^T(n)\widehat{\mathbf{h}}_l(n-1) - \mathbf{x}_l^T(n)\widehat{\mathbf{h}}_m(n-1), \quad m, l = 1, \ldots, M, \tag{24}$$

the *a priori* error signal associated with channel m and l. The update equation for $\widehat{\mathbf{h}}$ is given by [28]:

$$\begin{aligned}\widehat{\mathbf{h}}(n) &= \widehat{\mathbf{h}}(n-1) - \mu\nabla J(n) \\ &= \widehat{\mathbf{h}}(n-1) - \frac{2\mu}{\|\widehat{\mathbf{h}}(n-1)\|_2^2}\left[\widetilde{\mathbf{R}}(n)\widehat{\mathbf{h}}(n-1) - J(n)\widehat{\mathbf{h}}(n-1)\right],\end{aligned} \tag{25}$$

where $\|\cdot\|_2$ denotes l_2-norm and $\widetilde{\mathbf{R}}(n)$ is the instantaneous estimate of (17). If the channel estimate is always normalized after each updating iteration, (25) can be written as

$$\widehat{\mathbf{h}}(n) = \frac{\widehat{\mathbf{h}}(n-1) - 2\mu[\widetilde{\mathbf{R}}(n)\widehat{\mathbf{h}}(n-1) - E(n)\widehat{\mathbf{h}}(n-1)]}{\left\|\widehat{\mathbf{h}}(n-1) - 2\mu[\widetilde{\mathbf{R}}(n)\widehat{\mathbf{h}}(n-1) - E(n)\widehat{\mathbf{h}}(n-1)]\right\|_2}. \tag{26}$$

To avoid trivial solutions such as $\widehat{\mathbf{h}}(n) = \mathbf{0}_{ML\times 1}$, where $\mathbf{0}_{ML\times 1}$ is a null vector of length ML, the unit-norm constraint is imposed and $\widehat{\mathbf{h}}(n)$ is initialized as $\widehat{\mathbf{h}}_m(0) = [1/\sqrt{M}\ 0\ \ldots\ 0]^T$, $m = 1, 2, \ldots, M$ such that $\|\widehat{\mathbf{h}}(0)\|_2^2 = 1$.

Recently, various variable step sizes were derived for the MCLMS algorithm. In in [32] the following variable step size was derived:

$$\mu_{\text{vss}}(n) = \frac{\widehat{\mathbf{h}}^T(n-1)\nabla J(n)}{\|\nabla J(n)\|_2^2}. \tag{27}$$

Example The following is an example script (`test_mclms.m`) of running `mclms()` in the context of multichannel blind system identification.

```
% Run adaptive BSI algorithm: MCLMS
% This script runs the MCLMS algorithm for blind
% system identification
%
% References:
```

```
%   [1] J. Allen and D. Berkley, "Image method for
%       efficiently simulating small room acoustics",
%       JASA, 1979.
%   [2] Y. Huang and J. Benesty, "Adaptive multi-
%       channel mean square and Newton algorithms
%       for blind channel identification",
%       Signal Process., vol. 83, no. 8,
%       pp. 1127-1138, Aug 2002.
%
% Authors: S. Lin, E.A.P. Habets
%
% History: 2009-06-25 Initial version by SL
%          2009-07-10 Major changes by EH
%
% Copyright (C) Imperial College London 2009-2010
% Version: $Id: test_mclms.m,v 1.5 2010/11/29 11:52:53↩
    mrt102 Exp $

clc;
clear;
close all;

%% Initialization
M = 5;       % number of channels
fs = 8e3;    % sampling frequency
L = 128;     % channel length
SNR = 30;    % signal to noise ratio (dB)
N = 10*fs;   % data length (samples)
s_seed = 1; % seed for generating source signal
v_seed = 50;% seed for generating noise signal
mu = .8;     % step-size

% Generate sensor signals and AIRs
air.c = 342;  % speed of sound
air.T60 = 0.3;  % reverberation time
air.room_dim = [5; 6; 3];  % room dimension
air.mic_spacing = 0.2;  % microphone spacing (m)
air.src_pos = [100*pi/180 2*pi/180 2];  % source ↩
    location
air.cen_pos = [2.5; 2; 1.6];  % centre pos. of the ↩
    array
[h, x] = generate_data(M, L, fs, air, N, SNR, s_seed, ↩
    v_seed);

% Initiailize MCLMS
[xin, h_hat] = init_mclms(L, M);
npm_dB = zeros(N,1);
J = zeros(N,1);

%% Processing Loop: run MCLMS [2]
wbar = waitbar(0,'MCLMS');
ss_cntr = {'fixed','vss'};
h_hat = repmat(h_hat,[1 1 length(ss_cntr)]);
for nn = 1 : N
    waitbar(nn/N);
    xin = [x(nn, :); xin(1 : L-1, :)];
    for tt = 1 : length(ss_cntr)
        [h_hat(:,:,tt), J(nn,tt)] = mclms(xin, h_hat↩
            (:,:,tt), mu, ss_cntr{tt});
```

```
            npm_dB(nn,tt) = 20*log10(npm(h, h_hat(:,:,tt))↩
                );
        end
    end
    close(wbar);

    %% Plot results
    figure(1);
    phandle = plot_npm(npm_dB, fs,1);
    lhandle = addmarkers(phandle,20);
    legend(lhandle,['MCLMS' strcat(ss_cntr(2:end),'-MCLMS'↩
        )]);
    title(['L= ',num2str(L), ', \mu= ',num2str(mu), ...
        ', SNR= ',num2str(SNR), ', M= ', num2str(M)]);

    figure(2); plot_J(J, fs);  % cost function
    legend(['MCLMS' strcat(ss_cntr(2:end),'-MCLMS')]);
    title(['L= ',num2str(L), ', \mu= ',num2str(mu), ...
        ', SNR= ',num2str(SNR), ', M= ', num2str(M)]);

    figure(3); plot_filter(h, h_hat(:,:,2)) % filter coeff↩
        .
```

Running the above script (`test_mclms.m`) will produce Fig. 3. Fig. 3(a) shows the rate of convergence of MCLMS algorithm in NPM against time. Fig. 3(b) then plots the time-domain samples of $\mathbf{h}_1(n)$ and $\widehat{\mathbf{h}}_1(n)$, where the blue lines indicate the amplitude of each sample of $\mathbf{h}_1(n)$ and the red lines indicate those of $\widehat{\mathbf{h}}_1(n)$. In order to show the estimation performance, the arbitrary scale factor resultant from using MCLMS algorithm is removed by normalizing the impulse responses in the time domain. Finally, in Fig. 3(c) the cost function $J(n)$ is shown, where we observe that the value of the cost function is, on average, decreasing along time.

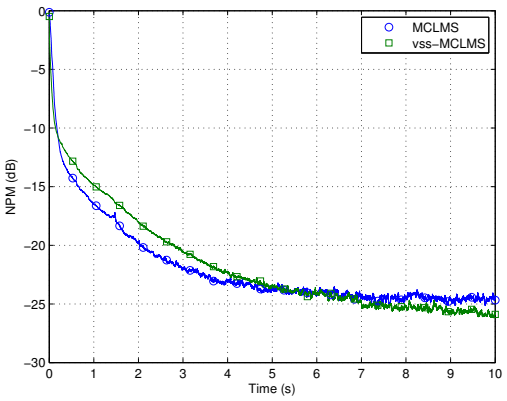
Algorithm `mclms()` performs the following operations:

1. Computes the CR error at discrete time n using (24) and (23).
2. Computes the cost function at discrete time n using (22).
3. Computes the instantaneous estimate of the matrix $\mathbf{R}$ given by (17).
4. Updates the adaptive filter coefficients using (26).

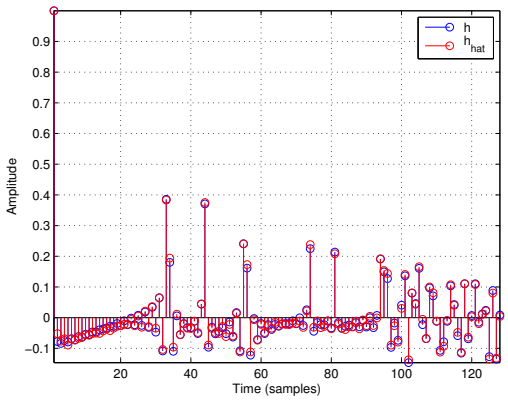
Remarks The MCLMS algorithm is one of the first adaptive blind system identification algorithms proposed in the context of acoustic signal processing. However, it has following limitations:

- Slow convergence due to large number of adaptive filter coefficients and the potential presence of common zeros [57].
- Very sensitive to additive noise [36, 38].
- Does not support complex data and impulse responses.
- Smaller step size is preferable for speech input data, but at the cost of slower rate of convergence.

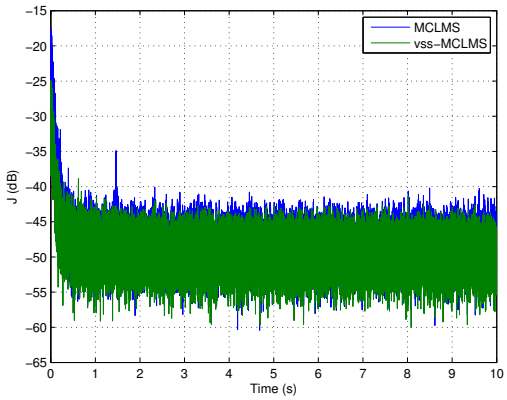
See also `mcn()`



(a) NPM as a function of time.



(b) Time-domain coefficients of the true and estimated impulse responses of channel 1 obtained using constrained-MCLMS.



(c) Cost function $J(n)$.

Figure 3: MCLMS with and without variable step sizes using a WGN source signal (SNR=30 dB, $L = 128$, $M = 5$).

2.2 Multichannel Newton algorithm: mcn()

Purpose Time-domain multichannel Newton adaptive filtering algorithm for blind system identification.

Syntax `[xin, h_hat, Rhat] = init_mcn(L, N, M, xin0, h_hat0)`

The input and output parameters of `init_mcn()` are summarized below.

```
Input Parameters [size]:
       L      : filter length
       M      : number of channels
       xin0   : initial input matrix [L x M]
       h_hat0 : initial filter coef. matrix (unit-norm
                constrained) [L x M]

   Output parameters [size]:
       xin    : initialized input matrix [L x M]
       h_hat  : initialized filter coef. matrix [L x M]
       Rhat   : covariance matrix [M L x M L]
```

```
[h_hat, R_hat, J] = mcn(xin, h_hat, R_hat, rho, lambda)
```

The input and output parameters of `mcn()` for an FIR adaptive filter of L coefficients are summarized below.

```
Input parameters:
    xin   : input matrix [L x M]
       h_hat : current filter coef. matrix [L x M]
       R_hat : covariance matrix [M L x M L]
       rho   : step size (optional: default rho=0.95)
       lamda : exponential forgetting factor (0 < lamda < 1)
               (optional: default lamda=0.99)

Outputs parameters:
       h_hat : updated filter coef. matrix [L x M]
       R_hat : covariance matrix [M L x M L]
       J     : value of cost function
```

Description While the MCLMS algorithm with unit-norm constraint can converge in the mean to the desired channel impulse responses the main difficulty is the selection of the step size μ. As pointed out in many studies, there is a trade-off between the amount of excess mean-squared error, the rate of convergence, and the ability of the algorithm to track changes in the system. In order to obtain a good balance of these competing design objectives, the unit-norm-constrained multichannel Newton algorithm with a variable step size during adaptation was derived in [28]. The update of the estimated impulse responses is given by:

$$\widehat{\mathbf{h}}(n) = \widehat{\mathbf{h}}(n-1) - \mathrm{E}^{-1}\{\nabla^2 J(n)\}\,\nabla J(n), \tag{28}$$

where $\nabla^2 J(n)$ is the Hessian matrix of $J(n)$ with respect to $\widehat{\mathbf{h}}(n-1)$. With some approximations the unit-norm-constrained update equation can be written as

$$\widehat{\mathbf{h}}(n) = \frac{\widehat{\mathbf{h}}(n-1) - 2\rho\mathbf{V}^{-1}(n)[\widehat{\mathbf{R}}(n)\widehat{\mathbf{h}}(n-1) - E(n)\widehat{\mathbf{h}}(n-1)]}{\left\|\widehat{\mathbf{h}}(n-1) - 2\rho\mathbf{V}^{-1}(n)[\widehat{\mathbf{R}}(n)\widehat{\mathbf{h}}(n-1) - E(n)\widehat{\mathbf{h}}(n-1)]\right\|_2}, \tag{29}$$

where

$$\mathbf{V}(n) = 2\widehat{\mathbf{R}}(n) - 4\widehat{\mathbf{h}}(n-1)\widehat{\mathbf{h}}^T(n-1)\widehat{\mathbf{R}}(n) - 4\widehat{\mathbf{R}}(n)\widehat{\mathbf{h}}(n-1)\widehat{\mathbf{h}}^T(n-1) \quad (30)$$

approximates $\mathrm{E}\{\nabla^2 J(n)\}$, ρ $(0 < \rho < 1)$ denotes the step size that is commonly chosen close to 1, and $\widehat{\mathbf{R}}(n)$ is an estimate of (17) computed using

$$\widehat{\mathbf{R}}(n) = \lambda\,\widehat{\mathbf{R}}(n-1) + (1-\lambda)\,\widetilde{\mathbf{R}}(n), \quad (31)$$

where λ $(0 < \lambda < 1)$ is an exponential forgetting factor.

Example The following is an example script (`test_mcn.m`) of running `mcn()` in the context of multi-channel blind system identication.

```
% Run adaptive BSI algorithm: MCN
% This script runs the MCN algorithm for blind
% system identification
%
% References:
%   [1] J. Allen and D. Berkley, "Image method for
%       efficiently simulating small room acoustics",
%       JASA, 1979.
%   [2] Y. Huang and J. Benesty, "Adaptive multi-
%       channel mean square and Newton algorithms
%       for blind channel identification",
%       Signal Process., vol. 83, no. 8,
%       pp. 1127-1138, Aug 2002.
%
% Authors: E.A.P. Habets
%
% History: 2009-07-10 Initial version
%
% Copyright (C) Imperial College London 2009-2010
% Version: $Id: test_mcn.m,v 1.4 2010/11/29 11:52:53 ↩
    mrt102 Exp $

clc;
clear;
close all;

%% Initialization
M = 5;          % number of channels
fs = 8e3;       % sampling frequency
L = 16;         % channel length
SNR = 40;       % signal to noise ratio (in dB)
N = 0.2*fs;     % data length (samples)
s_seed = 1; % seed for generating source signal
v_seed = 50;% seed for generating noise signal
rho = 0.98;     % step-size
lambda = 0.99; % forgetting-factor

% Generate sensor signals and AIRs
air.c = 342;  % speed of sound
air.T60 = 0.3;  % reverberation time
air.room_dim = [5; 6; 3];  % room dimension
air.mic_spacing = 0.2;  % microphone spacing (m)
air.src_pos = [100*pi/180 2*pi/180 2];  % source ↩
    location
air.cen_pos = [2.5; 2; 1.6];  % centre pos. of the ↩
    array
```

```
[h, x] = generate_data(M, L, fs, air, N, SNR, s_seed, ↩
    v_seed);

% Initiailize MCLMS
[xin, h_hat, R_hat] = init_mcn(L, M, [x(1,:) ; ...
    zeros(L-1,M)]);
npm_dB = zeros(N, 1);
N = size(x, 1);
J = zeros(N, 1);

%% Processing Loop: run MCN [2]
wbar = waitbar(0,'MCN');
for nn = 1 : N
    waitbar(nn/N);
    xin = [x(nn, :) ; xin(1 : L-1, :)];
    [h_hat, R_hat, J(nn)] = mcn(xin, h_hat,...
        R_hat, rho, lambda);
    npm_dB(nn) = 20*log10(npm(h, h_hat));
end
close(wbar);

%% Plot results
figure(1); plot_npm(npm_dB, fs);  % NPM
legend('MCN');
title(['L= ',num2str(L), ', \rho= ',num2str(rho), ...
    ', \lambda= ',num2str(lambda), ', SNR= ', ...
    num2str(SNR), ', M= ', num2str(M)]);

figure(2); plot_J(J, fs);  % cost function
title(['L= ',num2str(L), ', \rho= ',num2str(rho), ...
    ', \lambda= ',num2str(lambda), ', SNR= ', ...
    num2str(SNR), ', M= ', num2str(M)]);

figure(3); plot_filter(h,h_hat);  % filter coeff.
```

Running the above script (`test_mcn.m`) will produce Fig. 4. Fig. 4(a) shows the rate of convergence of MCN algorithm in NPM against time. Fig. 4(b) then plots the time-domain samples of $\mathbf{h}_1(n)$ and $\widehat{\mathbf{h}}_1(n)$, where the blue lines indicate the amplitude of each sample of $\mathbf{h}_1(n)$ and the red lines indicate those of $\widehat{\mathbf{h}}_1(n)$. In order to show the estimation performance, the arbitrary scale factor resultant from using MCN algorithm is removed by normalizing the impulse responses in the time domain. Compared to the MNCLMS algorithm the MCN algorithm converges much faster.

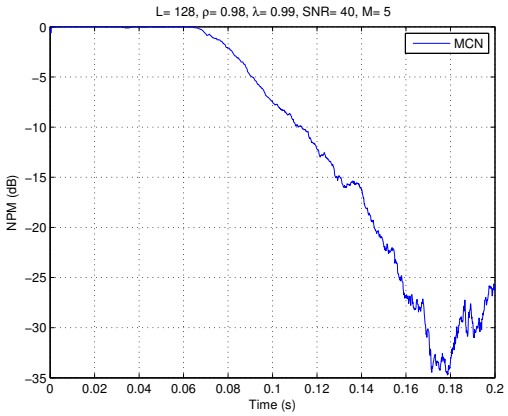
Algorithm `mcn()` performs the following operations:

1. Computes the CR error at discrete time n using (24) and (23).
2. Computes the cost function at discrete time n using (22).
3. Estimates the matrix $\mathbf{R}$ given by (17) using (31).
4. Updates the adaptive filter coefficients using (30) and (29).

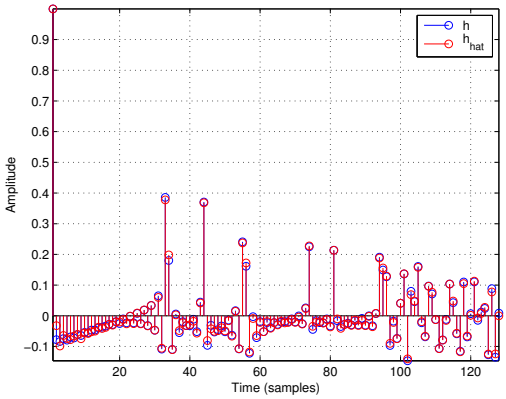
See also `mclms()`

2.3 Multichannel frequency-domain LMS algorithm: mcflms()

Purpose Frequency-domain multichannel LMS adaptive filtering algorithm for blind system identification.



(a) NPM as a function of time.



(b) Time-domain coefficients of the true and estimated impulse responses of channel 1.

Figure 4: MCN with WGN source signal (SNR=40 dB, $L = 128$, $M = 5$).

Syntax

```
[h_hat] = init_mcflms(L, M, h_hat0)
```

The input and output parameters of `init_mcflms()` for an FIR adaptive filter of L coefficients are summarized below.

```
Input Parameters [size]:
       L       : filter length
       M       : number of channels
       h_hat0  : initial filter coef. vector
                 (unit-norm constrained) [L x M]

Output parameters [size]:
       h_hat   : initialized filter coef. matrix [L x M]
```

```
[h_hat, J] = mcflms(xm, h_hat, mu, ss_cntr)
```

The input and output parameters of `mcflms()` for an FIR adaptive filter of L coefficients are summarized below.

```
Input Parameters [size]:
       xm      : input matrix [F x M]
       h_hat   : current filter coef. matrix [L x M]
       mu      : step size (optional: default mu=0.9)
       ss_cntr : type of step-size control
                 (optional: default 'fixed'):
                 'fixed' - fixed with unit-norm-constrained;
                 'vss' - variable step-size
                 'rvss' - robust variable step-size

Output Parameters:
       h_hat   : updated filter coef. matrix [L x M]
       J       : value of the cost function
```

Description The MCFLMS algorithm is developed based on the derivation of cross-relation (CR) in the frequency domain. The linear convolution between

the mth channel output $\mathbf{x}_m(n)$ and the lth channel estimate $\widehat{\mathbf{h}}_l(n)$ can be implemented using a vector of length $F = 2L$ from the circular convolution

$$\widetilde{\mathbf{y}}_{ml}(k) \;=\; \mathbf{C}_{x_m}(k)\widehat{\mathbf{h}}_l^{10}(k), \tag{32}$$

where k is the frame index, $\widehat{\mathbf{h}}_l^{10}(k) = [\widehat{\mathbf{h}}_l^T(k)\ \mathbf{0}_{L\times 1}^T]^T$ is the lth channel estimate with zero padding, and

$$\mathbf{C}_{x_m}(k) = \begin{bmatrix} x_m(kL-L) & x_m(kL+L-1) & \cdots & x_m(kL-L+1) \\ x_m(kL-L+1) & x_m(kL-L) & \cdots & x_m(kL-L+2) \\ \vdots & \vdots & \ddots & \vdots \\ x_m(kL+L-1) & x_m(kL+L-2) & \cdots & x_m(kL-L) \end{bmatrix} \tag{33}$$

is the $2L \times 2L$ circulant matrix constructed from

$$\boldsymbol{\chi}_m(k) \;=\; [x_m(kL-L)\ x_m(kL-L+1)\ \ldots\ x_m(kL+L-1)]^T. \tag{34}$$

It should be noted that $\mathbf{C}_{x_m}(k)$ is only determined by $\boldsymbol{\chi}_m(k)$, e.g., the second column of $\mathbf{C}_{x_m}(k)$ is the shifted version of $\boldsymbol{\chi}_m(k)$. A 50% overlap-save[2] is employed so that L previous samples of $\mathbf{x}_m(n)$ are included in $\boldsymbol{\chi}_m(k)$. For each $\widetilde{\mathbf{y}}_{ml}(k)$ of length $2L$, the last L samples are retained since they correspond to the linear convolution given by $\mathbf{x}_m^T(n)\widehat{\mathbf{h}}_l(n)$. As a result, by defining two selecting matrices

$$\mathbf{W}_{L\times 2L}^{01} \;=\; [\mathbf{0}_{L\times L}\ \mathbf{I}_{L\times L}], \qquad \mathbf{W}_{2L\times L}^{10} \;=\; [\mathbf{I}_{L\times L}\ \mathbf{0}_{L\times L}]^T \tag{35}$$

the desired result $\mathbf{y}_{ml}(k)$ can be obtained

$$\begin{aligned} \mathbf{y}_{ml}(k) = \mathbf{W}_{L\times 2L}^{01}\widetilde{\mathbf{y}}_{ml}(k) \;&=\; \mathbf{W}_{L\times 2L}^{01}\mathbf{C}_{x_m}(k)\widehat{\mathbf{h}}_l^{10}(k), \\ &= \mathbf{W}_{L\times 2L}^{01}\mathbf{C}_{x_m}(k)\mathbf{W}_{2L\times L}^{10}\widehat{\mathbf{h}}_l(k). \end{aligned} \tag{36}$$

To employ FFT techniques for efficient implementation of circular convolution, it is important to note that the circulant matrix $\mathbf{C}_{x_m}(k)$ can be decomposed as

$$\mathbf{C}_{x_m}(k) \;=\; \mathbf{F}_{2L}^{-1}\underline{\boldsymbol{\mathcal{D}}}_m(k)\mathbf{F}_{2L}, \tag{37}$$

where $\mathbf{F}_{2L}$ is a $2L \times 2L$ DFT matrix with $\mathbf{F}_{2L}^{-1} = \mathbf{F}_{2L}^H/(2L)$, and $\underline{\boldsymbol{\mathcal{D}}}_m(k)$ is a diagonal matrix with diagonal elements being given by the DFT of $\boldsymbol{\chi}_m(k)$. Now, the frequency-domain CR error function can be written

$$\begin{aligned} \underline{\boldsymbol{e}}_{ml}(k) &= \mathbf{F}_L\left[\mathbf{y}_{ml}(k) - \mathbf{y}_{lm}(k)\right], \\ &= \mathbf{F}_L\mathbf{W}_{L\times 2L}^{01}\left[\mathbf{C}_{x_m}(k)\mathbf{W}_{2L\times L}^{10}\widehat{\mathbf{h}}_l(k) - \mathbf{C}_{x_l}(k)\mathbf{W}_{2L\times L}^{10}\widehat{\mathbf{h}}_m(k)\right], \\ &= \boldsymbol{\mathcal{W}}_{L\times 2L}^{01}\left[\underline{\boldsymbol{\mathcal{D}}}_m(k)\boldsymbol{\mathcal{W}}_{2L\times L}^{10}\underline{\widehat{\mathbf{h}}}_l(k) - \underline{\boldsymbol{\mathcal{D}}}_l(k)\boldsymbol{\mathcal{W}}_{2L\times L}^{10}\underline{\widehat{\mathbf{h}}}_m(k)\right], \end{aligned} \tag{38}$$

for $m, l = 1, 2, \ldots, M,\ m \neq l$, where

$$\boldsymbol{\mathcal{W}}_{L\times 2L}^{01} \;=\; \mathbf{F}_L\mathbf{W}_{L\times 2L}^{01}\mathbf{F}_{2L}^{-1}, \;\; \boldsymbol{\mathcal{W}}_{2L\times L}^{10} \;=\; \mathbf{F}_{2L}\mathbf{W}_{2L\times L}^{10}\mathbf{F}_L^{-1}, \tag{39}$$

$$\underline{\widehat{\mathbf{h}}}_m(k) \;=\; \mathbf{F}_L\widehat{\mathbf{h}}_m(k). \tag{40}$$

[2]It should be noted that the implementation allows the use of any overlap smaller or equal than 50%. However, the most computational efficient implementation is obtained with 50% overlap.

The MCFLMS algorithm is therefore given by [29], for $m = 1, 2, \ldots, M$,

$$\underline{\boldsymbol{e}}_{ml}^{01}(k) = \boldsymbol{\mathcal{W}}_{2L\times L}^{01}\underline{\boldsymbol{e}}_{ml}(k) \;=\; \mathbf{F}_{2L}\begin{bmatrix}\mathbf{0}_{L\times 1}\\ \mathbf{F}_L^{-1}\underline{\boldsymbol{e}}_{ml}(k)\end{bmatrix}, \tag{41}$$

$$\widehat{\underline{\mathbf{h}}}_m^{10}(k-1) = \mathbf{F}_{2L}\widehat{\mathbf{h}}_m^{10}(k-1) \;=\; \mathbf{F}_{2L}\mathbf{W}_{2L\times L}^{10}\widehat{\mathbf{h}}_m(k-1), \tag{42}$$

$$\widehat{\underline{\mathbf{h}}}_m^{10}(k) = \widehat{\underline{\mathbf{h}}}_m^{10}(k-1) - \mu\sum_{l=1}^{M}\underline{\boldsymbol{\mathcal{D}}}_l^*(k)\underline{\boldsymbol{e}}_{ml}^{01}(k) \tag{43}$$

where μ is the step size, and

$$\boldsymbol{\mathcal{W}}_{2L\times L}^{01} \;=\; \mathbf{F}_{2L}\mathbf{W}_{2L\times L}^{01}\mathbf{F}_L^{-1} \;=\; 2\left(\boldsymbol{\mathcal{W}}_{L\times 2L}^{01}\right)^H \tag{44}$$

with $\mathbf{W}_{2L\times L}^{01} \;=\; \left(\mathbf{W}_{L\times 2L}^{01}\right)^T$. To satisfy the unit-norm constraint [29], the frequency-domain coefficients of the adaptive filter are initialized as $\widehat{\underline{\mathbf{h}}}_m^{10}(0) = (\mathbf{1}_{2L\times 1})/\sqrt{M}$.

The performance of the MCFLMS algorithm is highly dependent upon the step size. The standard MCFLMS uses a fixed step size. Two alternative variable step sizes are available, viz., `vss` and `rvss`, that were derived in [58].

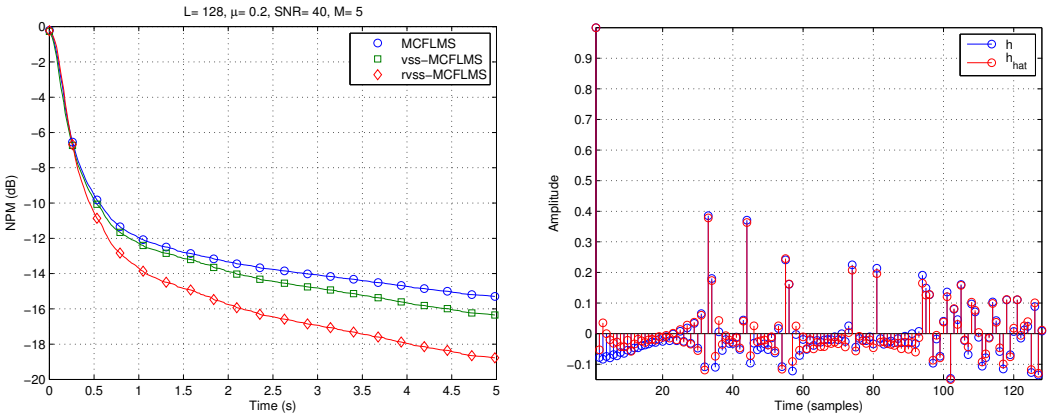
Example The following is an example script (`test_mcflms.m`) of running `mcflms()` in the context of multichannel blind system identification.

```
% Run adaptive BSI algorithm: MCFLMS
% This script runs the MCFLMS algorithm for
% blind system identification
%
% References:
%   [1] J. Allen and D. Berkley, "Image method for
%       efficiently simulating small room acoustics",
%       JASA, 1979.
%   [2] Y. Huang and J. Benesty, "Frequency-Domain
%       adaptive approaches to blind multi-channel
%       identification," IEEE Trans. Sig. Process.,
%       Vol. 51, No. 1, Jan 2003.
%
% Authors: E.A.P. Habets
%
% History: 2009-07-10 Initial version
%
% Copyright (C) Imperial College London 2009-2010
% Version: $Id: test_mcflms.m,v 1.4 2010/11/29 ↩
    11:52:53 mrt102 Exp $

clc;
clear;
close all;

%% Initialization
M = 5;      % number of channels
fs = 8e3;   % sampling frequency
L = 128;    % channel length
F = 2*L;    % frame lenght
SNR = 40;   % signal to noise ratio (in dB)
N = 5*fs;   % data length (samples)
```

```
s_seed = 1; % seed for generating source signal
v_seed = 50;% seed for generating noise signal
mu = 0.2;  % step-size

% Generate sensor signals and AIRs
air.c = 342;  % speed of sound
air.T60 = 0.3;  % reverberation time
air.room_dim = [5; 6; 3];  % room dimension
air.mic_spacing = 0.2;  % microphone spacing (m)
air.src_pos = [100*pi/180 2*pi/180 2];  % source location
air.cen_pos = [2.5; 2; 1.6];  % centre pos. of the array
[h, x] = generate_data(M, L, fs, air, N, SNR, s_seed, v_seed);

% Initiailize MCLMS
[h_hat] = init_mcflms(L, M);
ns = F-L+1;  % number of new samples per iteration
B = fix(N/ns);  % number of input blocks of length L
npm_dB = zeros(B,1);

%% Processing Loop: run MCFLMS [2]
wbar = waitbar(0,'MCFLMS');
ss_cntr = {'fixed','vss','rvss'};
h_hat = repmat(h_hat,[1 1 length(ss_cntr)]);
for bb = 1 : B
    waitbar(bb/B);
    if bb == 1
        xm = [zeros(F-ns,M); x(1:ns,:)];
    else
        xm = [xm(ns+1:end,:); x((bb-1)*ns+1:bb*ns,:)];
    end
    for tt = 1 : length(ss_cntr)
        [h_hat(:,:,tt)] = mcflms(xm, h_hat(:,:,tt), mu, ss_cntr{tt});
        npm_dB(bb,tt) = 20*log10(npm(h, h_hat(:,:,tt)));
    end
end
close(wbar);

%% Plot results
% NPM
figure(1);
phandle = plot_npm(npm_dB, fs, ns);
lhandle = addmarkers(phandle,20);
legend(lhandle,['MCFLMS' strcat(ss_cntr(2:end),'-MCFLMS')]);
title(['L= ',num2str(L), ', \mu= ',num2str(mu), ...
    ', SNR= ',num2str(SNR), ', M= ', num2str(M)]);

% Time-domain coefficients
figure(2); plot_filter(h, h_hat(:,:,1));
```

Running the above script (`test_mcflms.m`) will produce Fig. 5. Fig. 5(a) showing the rate of convergence of MCFLMS algorithm in NPM against time. Fig. 5(b) then plots the time-domain samples of $\mathbf{h}_1(n)$ and $\widehat{\mathbf{h}}_1(n)$,

where the blue lines indicate the amplitude of each sample of $\mathbf{h}_1(n)$ and the red lines indicate those of $\widehat{\mathbf{h}}_1(n)$. In order to show the estimation performance, the arbitrary scale factor resultant from using MCFLMS algorithm is removed by normalizing the impulse responses in the time domain.



(a) NPM as a function of time.

(b) Time-domain coefficients of the true and estimated impulse responses of channel 1 obtained using constrained-MCFLMS.

Figure 5: MCFLMS with different step size mechanisms and a WGN source signal (SNR=40 dB, $L = 128$, $M = 5$).

Algorithm `mcflms()` performs the following operations:

1. Computes the CR error of time frame k using (38) and (41).
2. Computes the cost function for time frame k.
3. Updates the adaptive filter coefficients using (42) and (43).

See also `mclms()`, `mcn()`, `nmcflms()`

2.4 Normalized multichannel frequency-domain LMS algorithm: nmcflms()

Purpose Normalized Frequency-domain multichannel LMS adaptive filtering algorithm for blind system identification.

Syntax `[h_hat, P_k] = init_nmcflms(L, F, M, xm0, h_hat0)`

The input and output parameters of `init_nmcflms()` are summarized below.

```
Input Parameters [size]:
       L      : filter length
       F      : frame length
       M      : number of channels
       xm0    : initial input block used
                to calculate P_k [F x M]
       h_hat0 : initial filter coef. matrix
                (unit-norm constrained) [L x M]
```

```
Output parameters [size]:
       h_hat  : initialized filter coef. matrix [L x M]
       P_k    : initialized PSDs [F x M]

[h_hat, P_k, J] = nmcflms(xm, h_hat, P_k, ro, lambda, delta)
```

The input and output parameters of `nmcflms()` for an FIR adaptive filter of L coefficients are summarized below.

```
Input Parameters [size]:
       xm      : input matrix [F x M]
       h_hat   : current filter coef. matrix [L x M]
       P_k_avg : estimated PSD matrix [F x M]
       rho     : step size (optional: default rho=0.8)
       lambda  : exponential forgetting factor
                 (0 < lambda < 1) (optional)
       delta   : regularization (optional)

Output Parameters:
       h_hat   : updated filter coef. matrix [L x M]
       P_k_avg : updated PSD matrix [F x M]
       J       : cost function values [F-L+1 x 1]
```

Description The NMCFLMS algorithm is given by [29], for $m = 1, 2, \ldots, M$,

$$\underline{\boldsymbol{\mathcal{P}}}_m(k) = \lambda \underline{\boldsymbol{\mathcal{P}}}_m(k-1) + (1-\lambda) \sum_{l=1, l\neq m}^{M} \underline{\boldsymbol{\mathcal{D}}}_l^*(k) \underline{\boldsymbol{\mathcal{D}}}_l(k), \tag{45}$$

$$\underline{\widehat{\mathbf{h}}}_m^{10}(k) = \underline{\widehat{\mathbf{h}}}_m^{10}(k-1) - \mu [\underline{\boldsymbol{\mathcal{P}}}_m(k) + \delta \mathbf{I}_{2L\times 2L}]^{-1} \times \sum_{l=1}^{M} \underline{\boldsymbol{\mathcal{D}}}_l^*(k) \underline{\boldsymbol{e}}_{ml}^{01}(k), \tag{46}$$

where $\lambda = [1 - 1/(3L)]^L$ is the forgetting factor, μ is the step size, δ is the regularization parameter. To satisfy the unit-norm constraint [29], the frequency-domain coefficients of the adaptive filter are initialized as $\underline{\widehat{\mathbf{h}}}_m^{10}(0) = (\mathbf{1}_{2L\times 1})/\sqrt{M}$.

In addition to the standard NMCFLMS algorithm the toolbox also facilitates the robust NMCFLMS (RNMCFLMS) algorithm (`rnmcflms()`) derived in [58], for details regarding this algorithm we refer the reader to that publication.

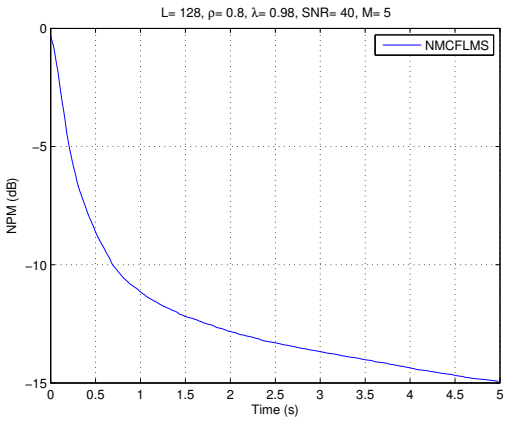
Example Here we present two example scripts that use `nmcflms()` and `rnmcflms()` in the context of multi-channel blind system identication. The first example script (`test_nmcflms.m`) uses a WGN source signal:

```
% Run adaptive BSI algorithm: NMCFLMS
% This script runs the NMCFLMS algorithm for
% blind system identification
%
% References:
%   [1] J. Allen and D. Berkley, "Image method for
%       efficiently simulating small room acoustics",
%       JASA, 1979.
%   [2] Y. Huang and J. Benesty, "Frequency-Domain
%       adaptive approaches to blind multi-channel
%       identification," IEEE Trans. Sig. Process.,
```

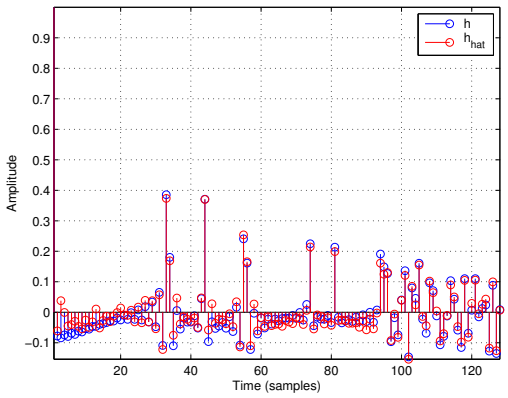
```
%       Vol. 51, No. 1, Jan 2003.
%
% Authors: E.A.P. Habets
%
% History: 2009-07-10 Initial version
%
% Copyright (C) Imperial College London 2009-2010
% Version: $Id: test_nmcflms.m,v 1.7 2011/03/01 ↩
    14:55:11 ehabets Exp $

clc;
clear;
%close all;

%% Initialization
M = 5;           % number of channels
fs = 8e3;        % sampling frequency
L = 128;         % channel length
F = 2*L;         % frame lenght
SNR = 40;        % signal to noise ratio (in dB)
s_seed = 1;      % seed for generating source signal
v_seed = 50;     % seed for generating noise signal
N = 5*fs;        % data length (samples)
rho = 0.8;       % step-size
lambda = 0.98;   % forgetting-factor

% Generate sensor signals and AIRs
air.c = 342;  % speed of sound
air.T60 = 0.3;  % reverberation time
air.room_dim = [5; 6; 3];  % room dimension
air.mic_spacing = 0.2;  % microphone spacing (m)
air.src_pos = [100*pi/180 2*pi/180 2];  % source ↩
    location
air.cen_pos = [2.5; 2; 1.6];  % centre pos. of the ↩
    array
[h, x] = generate_data(M, L, fs, air, N, SNR, s_seed, ↩
    v_seed);

% Initiailize NMCFLMS
[h_hat, P_k_avg] = init_nmcflms(L, F, M, x(1:F,:));
ns = F-L+1;
B = fix(N/ns);  % number of input blocks of length L
J = zeros(N,1);
npm_dB = zeros(B,1);

Xm = fft(x(1:F,:),F);
P_x = conj(Xm).*Xm;
delta = (M-1)*mean(mean(P_x));

%% Processing Loop: run NMCFLMS [2]
wbar = waitbar(0,'NMCFLMS');
for bb = 1 : B
    waitbar(bb/B);
    if bb == 1
        xm = [zeros(F-ns,M); x(1:ns,:)];
    else
        xm = [xm(ns+1:end,:); x((bb-1)*ns+1:bb*ns,:)];
    end
    [h_hat, P_k_avg, J_tmp] = nmcflms(xm, h_hat,...
```

```
        P_k_avg, rho, lambda, delta);
    J((bb-1)*ns+1:(bb)*ns) = J_tmp(F-L-ns+2:end);
    npm_dB(bb) = 20*log10(npm(h, h_hat));
end
close(wbar);

%% Plot results
figure(1); plot_npm(npm_dB, fs, ns);  % NPM
legend('NMCFLMS');
title(['L= ',num2str(L), ', \rho= ',num2str(rho), ...
    ', \lambda= ',num2str(lambda), ', SNR= ', ...
    num2str(SNR), ', M= ', num2str(M)]);

figure(2); plot_J(J, fs);  % cost function
legend('NMCFLMS');
title(['L= ',num2str(L), ', \rho= ',num2str(rho), ...
    ', \lambda= ',num2str(lambda), ', SNR= ', ...
    num2str(SNR), ', M= ', num2str(M)]);

figure(3); plot_filter(h,h_hat);  % filter coeff.
```

Running the above script (`test_nmcflms.m`) will produce Fig. 6. Fig. 6(a) shows the rate of convergence of NMCFLMS algorithm in NPM against time. Fig. 6(b) then plots the time-domain samples of $\mathbf{h}_1(n)$ and $\widehat{\mathbf{h}}_1(n)$, where the blue lines indicate the amplitude of each sample of $\mathbf{h}_1(n)$ and the red lines indicate those of $\widehat{\mathbf{h}}_1(n)$. In order to show the estimation performance, the arbitrary scale factor resultant from using NMCFLMS algorithm is removed by normalizing the impulse responses in the time domain.



(a) NPM as a function of time.



(b) Time-domain coefficients of the true and estimated impulse responses of channel 1.

Figure 6: NMCFLMS with WGN source signal (SNR=40 dB, $L = 128$, $M = 5$).

The second example script (`test_rnmcflms_speech.m`) uses a male speech source signal:

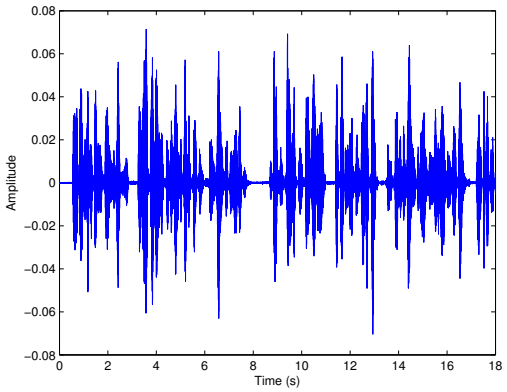
```
% Run adaptive BSI algorithm: RNMCFLMS
% This script runs the RNMCFLMS algorithm for blind
```

```
% system identification, the acoustic source is
% a male speaker
%
% References:
%   [1] J. Allen and D. Berkley, "Image method for
%       efficiently simulating small room acoustics",
%       JASA, 1979.
%   [2] Y. Huang and J. Benesty, "Frequency-Domain
%       adaptive approaches to blind multi-channel
%       identification," IEEE Trans. Sig. Process.,
%       Vol. 51, No. 1, Jan 2003.
%   [3] M. Haque and M. Hasan, "Noise robust multi-
%       channel frequency-domain LMS algorithms
%       for blind channel identification," IEEE Signal
%       Process. Lett., vol. 15, pp. 305-308, 2008.
%
% Authors: E.A.P. Habets
%
% History: 2009-07-10 Initial version
%
% Copyright (C) Imperial College London 2009-2010
% Version: $Id: test_rnmcflms_speech.m,v 1.6 ↩
    2010/10/21 13:29:00 ehabets Exp $

clc;
clear;
close all;

%% Initialization
M = 4;           % number of channels
fs = 8e3;        % sampling frequency
L = 256;         % channel length
F = 2*L;         % frame-length
SNR = 50;        % signal to noise ratio (in dB)
rho = 0.2;       % step-size
lambda = 0.98;   % forgetting-factor

% Generate AIRs
air.c = 342;  % speed of sound
air.T60 = 0.3;  % reverberation time
air.room_dim = [5; 6; 3];  % room dimension
air.mic_spacing = 0.2;  % microphone spacing (m)
air.src_pos = [100*pi/180 2*pi/180 2];  % source ↩
    location
air.cen_pos = [2.5; 2; 1.6];  % centre pos. of the ↩
    array
h = generate_data(M, L, fs, air);

% Load source signal
N = 18*fs;  % Simulation time in seconds
[s, fs_wav] = wavread([datapath 'male_english_8k.wav'↩
    ]);
if fs_wav ~= fs
    error('Incorrect sample frequency.');
end
s = s(1:N);
% Add small amount of noise to source signal to ↩
    improve stability
RandStream.setDefaultStream(RandStream('mt19937ar','↩
```

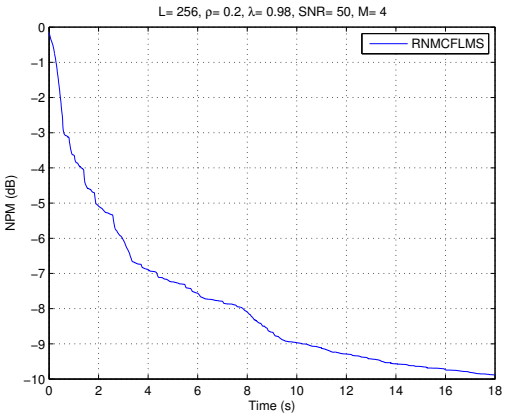
```
    Seed',1));
va = randn(size(s,1), 1);
va = sqrt(var(s)/(10^(40/10)*mean(var(va)))) .* va;
s = s + va;
s = 0.9 * s' / max(abs(s));

% Calculate noisy sensor signals with correct SNR
z = fftfilt(h, s);
RandStream.setDefaultStream(RandStream('mt19937ar','↩
    Seed',50));
v = randn(N, M);  % Generate additive noise
v = sqrt(var(s)*norm(h(:),2).^2/(10^(SNR/10)*M*mean(↩
    var(v)))) .* v;
x = z + v;

% Initiailize RNMCFLMS
[h_hat, P_k_avg] = init_nmcflms(L, F, M, x(1:F,:));
ns = L/8;  % number of new samples per iteration
B = fix(N/ns);  % number of input blocks of length L
npm_dB = zeros(B,1);

%% Processing Loop: run RNMCFLMS [2,3]
wbar = waitbar(0,'RNMCFLMS');
for bb = 1 : B
    waitbar(bb/B);
    if bb == 1
        xm = [zeros(F-ns,M); x(1:ns,:)];
    else
        xm = [xm(ns+1:end,:); x((bb-1)*ns+1:bb*ns,:)];
    end

    Xm = fft(xm,F);
    P_x = conj(Xm).*Xm;
    delta = (M-1)*mean(mean(P_x));

    [h_hat, P_k_avg] = rnmcflms(xm, h_hat, P_k_avg,...
        rho, lambda, delta);
    npm_dB(bb) = 20*log10(npm(h, h_hat));
end
close(wbar);

%% Plot results
% NPM
figure(1); plot_npm(npm_dB, fs, ns);
legend('RNMCFLMS');
title(['L= ',num2str(L), ', \rho= ',num2str(rho), ...
    ', \lambda= ',num2str(lambda), ', SNR= ', ...
    num2str(SNR), ', M= ', num2str(M)]);

% Filter coeff.
figure(2); plot_filter(h,h_hat);

% Magnitude response
figure(3);
freqs = (0:(fs/2)/(L/2):(fs/2))/1000;
H = fft(h(:,1)./norm(h(:,1)),L);
H_hat = fft(h_hat(:,1)./norm(h_hat(:,1)),L);
plot(freqs,20*log10(abs(H(1:end/2+1)))); hold on;
plot(freqs,20*log10(abs(H_hat(1:end/2+1))), 'r'); grid↩
```

```
    on; hold off;
xlabel('Frequency (kHz)');
ylabel('Magnitude (dB)');
legend('h','h_{hat}');
axis tight;

% First microphone signal
figure(4);
plot_time(x(:,1), fs);
```

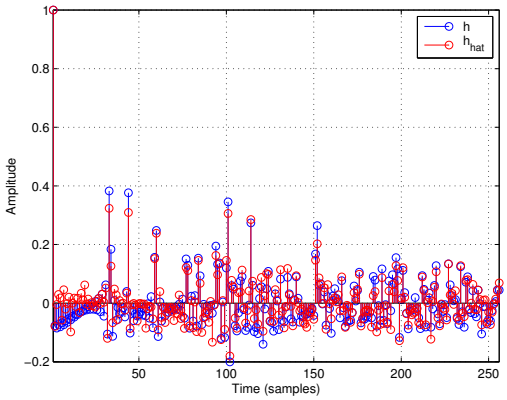
Running the above script (`test_rnmcflms_speech.m`) will produce four figures as shown in Fig. 7. Fig. 7(a) shows the signal received by the first microphone. Fig. 7(b) shows the rate of convergence of RNMCFLMS algorithm in NPM against time. Fig. 7(c) then plots the time-domain samples of $\mathbf{h}_1(n)$ and $\widehat{\mathbf{h}}_1(n)$, where the blue lines indicate the amplitude of each sample of $\mathbf{h}_1(n)$ and the red lines indicate those of $\widehat{\mathbf{h}}_1(n)$. Fig. 7(d) shows the magnitude response of the true and estimate impulse responses of the first acoustic channel.



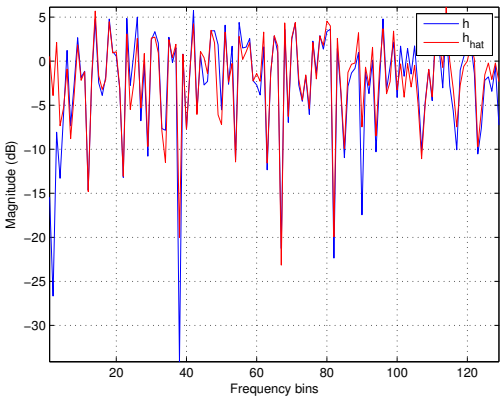
(a) Reverberant signal received by the first microphone.



(b) NPM as a function of time.



(c) Time-domain coefficients of the true and estimated impulse responses of channel 1.



(d) Magnitude response of the true and estimated impulse responses of channel 1.

Figure 7: RNMCFLMS with speech source signal (SNR = 50 dB, $L = 256$, $M = 4$).

Algorithm `nmcflms()` performs the following operations:

1. Computes the CR error of time frame k using (38) and (41).
2. Computes the cost function for time frame k.
3. Updates the adaptive filter coefficients using (46) and (42).

See also `mclms()`, `mcn()`, `mcflms()`

2.5 Normalized Multichannel Multi-Delay Filter LMS Algorithm: nmcmdflms()

Purpose Normalized Frequency-domain multichannel multi-delay filtering LMS adaptive filtering algorithm for blind system identification.

Syntax `[shMDF, H_hat] = init_nmcmlflms(L, Ms, Mch, xm0, H_hat0)`

The input and output parameters of `init_nmcmdflms()` are summarized below.

```
Input Parameters [size]:
       L      : filter length
       Ms     : number of MDF channels
       Mch    : number of channels
       xm0    : input data
       H_hat0 : initial filter coef. matrix
                (unit-norm constrained) [L x M]

Output parameters [size]:
       shMDF  : MDF data structure
       H_hat  : initialized filter coef. matrix [L x M]
```

The input and output parameters of `nmcmdflms()` for an FIR adaptive filter of L coefficients are summarized below.

```
[shMDF H_hat J] = nmcmdflms(shMDF, xm, rho, lambda, delta)

   Input Parameters [size]:
       shMDF    : MDF structure
       xm       : input matrix [F x M]
       H_hat    : current filter coef. matrix [L x M]
       rho      : step size (optional: default rho=0.2)
       lambda   : exponential forgetting factor (0 < lambda < 1) (optional)
       delta    : regularization (optional)

   Output Parameters:
       shMDF    : MDF structure
       H_hat    : updated filter coef. matrix [L x M]
       J        : cross-relation error [Ms x Mch x Mch]

   shMDF structure:
       H_hat    : MDF representation of H_hat (MC)
       Xm       : MDF representation of x - with overlapping (MC)
       Ms       : number of MDF segments (blocks).
       Mch      : number of channels
       Nm       : segment length
```

Description The NMCMDFLMS algorithm is found in [59]. The MDF structure [60] reduces the problem of delay in frequency domain algorithm implementation by partitioning the adaptive filter of length L into K blocks such that $L = KM$ where N is the block length. Let m be the frame index and we define input matrix $\mathbf{X}_i(m)$ and *a priori* error $\mathbf{e}_{ij}(m)$ for $i \neq j$:

$$\mathbf{X}_i(m) = [\mathbf{x}_i(mN) \dots \mathbf{x}_i(nM + N - 1)], \ i = 1, 2, \dots, M, \tag{47}$$

$$\mathbf{e}_{ij}(m) = \mathbf{X}_i(m)\widehat{\mathbf{h}}_j(m-1) - \mathbf{X}_j(m)\widehat{\mathbf{h}}_i(m-1) \tag{48}$$

$$= [e_{ij}(mN) \dots e_{ij}(mN + N - 1)]^T. \tag{49}$$

Defining k as the block index, the diagonal data matrix $\mathbf{D}_{x_i}(m)$ for channel i is

$$\mathbf{D}_{x_i}(m) = \text{diag}\{\mathbf{F}_{2N}\boldsymbol{\chi}_i(m)\}, \tag{50}$$

$$\boldsymbol{\chi}_i(m) = [x_i(\tau + N) \ x_i(\tau - N + 1) \dots x_i(\tau + N - 1)]^T, \tag{51}$$

where $\tau = mN - kN$. Note that the first element of the diagonal of $\mathbf{D}_{x_i}(m)$ is arbitrary, but it is normally equal to the first sample of the previous block $k - 1$ [26]. We now define the frequency domain quantities: $\underline{\widehat{\mathbf{h}}}_{i,k}(m) = \mathbf{F}_{2N}\begin{bmatrix} \widehat{\mathbf{h}}_{i,k}(m) \\ \mathbf{0}_{N\times 1} \end{bmatrix}$, $\underline{\mathbf{e}}_{i,j}(m) = \mathbf{F}_{2N}\begin{bmatrix} \mathbf{0}_{N\times 1} \\ \mathbf{e}_{i,j}(m) \end{bmatrix}$, $\mathbf{W}^{01}_{2N\times 2N} = \begin{bmatrix} \mathbf{0}_{N\times N} & \mathbf{0}_{N\times N} \\ \mathbf{0}_{N\times N} & \mathbf{I}_{N\times N} \end{bmatrix}$, $\mathbf{W}^{10}_{2N\times 2N} = \begin{bmatrix} \mathbf{I}_{N\times N} & \mathbf{0}_{N\times N} \\ \mathbf{0}_{N\times N} & \mathbf{0}_{N\times N} \end{bmatrix}$, $\mathbf{G}^{10}_{2N\times 2N} = \mathbf{F}_{2N}\mathbf{W}^{10}_{2N\times 2N}\mathbf{F}^{-1}_{2N}$ and $\mathbf{G}^{01}_{2N\times 2N} = \mathbf{F}_{2N}\mathbf{W}^{01}_{2N\times 2N}\mathbf{F}^{-1}_{2N}$, where $\underline{\widehat{\mathbf{h}}}_{i,k}(m)$ is the k^{th} subfilter of the i^{th} channel, for $k = 0, \dots, K-1$ and $i = 1, \dots, M$. The MCMDF adaptive algorithm for BCI is then given by

$$\underline{\mathbf{e}}_{ij}(m) = \mathbf{G}^{01}_{2N\times 2N}\sum_{k=0}^{K-1}\mathbf{D}_{x_i}(m-k)\underline{\widehat{\mathbf{h}}}_{j,k}(m-1) - \tag{52}$$

$$\mathbf{G}^{01}_{2N\times 2N}\sum_{k=0}^{K-1}\mathbf{D}_{x_j}(m-k)\underline{\widehat{\mathbf{h}}}_{i,k}(m-1), \tag{53}$$

$$\mathbf{S}_i(m) = \lambda\mathbf{S}_i(m-1) + \tag{54}$$

$$(1-\lambda)\sum_{j=1, j\neq i}^{M}\mathbf{D}^*_{x_j}(m)\mathbf{D}_{x_j}(m), \tag{55}$$

$$\underline{\widehat{\mathbf{h}}}_{i,k}(m) = \underline{\widehat{\mathbf{h}}}_{i,k}(m-1) - \beta\mathbf{G}^{10}_{2N\times 2N}\times \tag{56}$$

$$[\mathbf{S}_i(m) + \delta_{\text{MDF}}]^{-1}\sum_{j=1}^{M}\mathbf{D}^*_{x_j}(m-k)\underline{\mathbf{e}}_{ji}(m), \tag{57}$$

where β is the step size $0 \ll \lambda < 1$ is the forgetting factor. Defining σ_x^2 as the input signal variance, $\mathbf{S}_i(0) = \sigma^2_{x_i}/100$ is the initialization [26] and δ_{MDF} is the regularization constant which is set to one fifth the total power over all channels at the first frame [29].

Example Here we present an example script that uses `nmcmdflms()` in the context of multi-channel blind system identication. The example script (`test_nmcdfflms.m`) uses a WGN source signal:

```
% Run adaptive BSI algorithm: NMCMDFLMS
% This script runs the NMCMDFLMS algorithm [2] for
% blind system identification
```

```
%
% References:
%   [1] J. Allen and D. Berkley, "Image method for
%       efficiently sirholating small room acoustics",
%       JASA, 1979.
%   [2] R. Ahmad, A. W. H. Khong, and P. Naylor, ↩
    Proportionate frequency
%       domain adaptive algorithms for blind channel ↩
    identification,  in
%       Proc. ICASSP, Toulouse, France, May 2006, vol.↩
     5, pp. 29 32.
%
% Authors: E.A.P. Habets
%
% History: 2011-03-01 Initial version
%
% Copyright (C) Imperial College London 2011
% Version: $Id: test_nmcmdflms.m,v 1.1 2011/03/01 ↩
    14:59:08 ehabets Exp $

clc;
clear;
close all;

%% Initialization
Mch = 5;            % number of channels
fs = 8e3;           % sampling frequency
L = 128;            % channel length
F = 2*L;            % frame lenght
SNR = 40;           % signal to noise ratio (in dB)
TimeSec = 10;       % duration (seconds)
Ldata = TimeSec*fs; % data length (samples)
rho = 0.6;
Mks = 0:2;          % determines the number of ↩
    partitions to consider
lstyle = {'-','--',':'};

% Generate sensor signals and AIRs
air.c = 342;  % speed of sound
air.T60 = 0.3;  % reverberation time
air.room_dim = [5; 6; 3];  % room dimension
air.mic_spacing = 0.2;  % microphone spacing (m)
air.src_pos = [100*pi/180 2*pi/180 2];  % source ↩
    location
air.cen_pos = [2.5; 2; 1.6];  % centre pos. of the ↩
    array
[h, x] = generate_data(Mch, L, fs, air, Ldata, SNR);

%% Process
for Mk_idx = 1:length(Mks)
    Ms = 2^Mks(Mk_idx);
    Ns = 2*L/Ms;
    npm_dB_all = [];

    for rho_idx = 1:length(rho)
        % Initiailize NMCMDFLMS
        [shMDF h_hat] = init_nmcmdflms(L, Ms, Mch, x);

        ns = Ns/2; % number of new samples per ↩
```

```
            iteration
        B = fix(Ldata/ns); % number of input blocks of↩
             length L
        npm_dB = zeros(B,1);

        % Processing Loop: run NMCMDFLMS
        wbar = waitbar(0,'NMCMDFLMS');

        for bb = 1 : B
            waitbar(bb/B);

            if bb == 1
                xm = [zeros(ns,Mch); x(1:ns,:)];
            else
                xm = [xm(ns+1:end,:); x((bb-1)*ns+1:bb↩
                    *ns,:)];
            end

            [shMDF, h_hat] = nmcmdflms(shMDF, xm, rho(↩
                rho_idx));
            npm_dB(bb) = 20*log10(npm(h, h_hat));
        end
        npm_dB_all = [npm_dB_all, npm_dB];
        close(wbar)
    end

    NN = length(npm_dB);
    xscale = 1:NN;
    tsec = Ldata/fs;
    xscale = xscale/NN*tsec;

    figure(1);
    hold on;
    plot(xscale,npm_dB_all,lstyle{Mk_idx});
end

%% Markup plot
leg = [];
for Mk_idx = 1:length(Mks)
    rho_str = num2str(rho.');
    for kk = 1:size(rho_str,1)
        leg = [leg ; ['NMCMDFLMS \rho = ', rho_str(kk↩
            ,:), ', Ms = ', num2str(2^Mks(Mk_idx))]];
    end
end
grid on;
axis([0 TimeSec -25 0])
legend(leg)
hold off;
xlabel('Time (s)');
ylabel('NPM (dB)');
```

Running the above script (`test_nmcmdflms.m`) will produce Fig. 8, showing the rate of convergence of NMCMDFLMS algorithm in NPM against time.

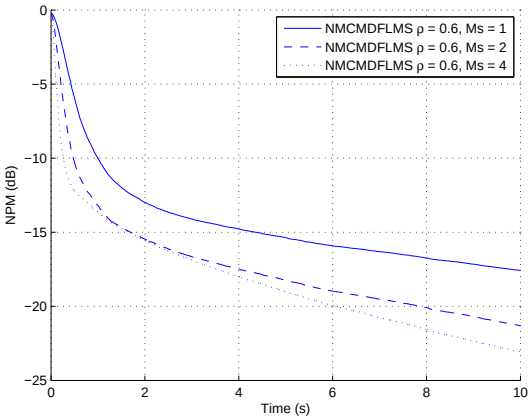


Figure 8: NMCMDFLMS with WGN source signal (SNR=40 dB, $L = 128$, $M = 5$).

3 System Equalization

In this Section various algorithms are describe that compute a set of equalization filters given the estimated impulse responses of the acoustic system.

3.1 Least squares algorithm: lsinvfilt()

Purpose Design a set of equalization filters given the estimated channel impulse responses.

Syntax `[g] = lsinvfilt(h_hat, Li, k)`

The input and output parameters of `lsinvfilt()` for an FIR system of size $L \times M$ are summarized below.

```
Input Parameters [size]:

 h_hat   :  M impulse responses of length L [L x M]
 Li      :  length of the equalization filters
 k       :  delay of the target response

Output Parameters [size]:

 g   :  equalization filters [Li x M]
```

Description A set of equalization filters, with length L_i, can be calculated by solving the following system of equations

$$\sum_{m=1}^{M} \widehat{h}_m(j) * g_m(j) = d(j) \quad \text{for } j = 0, \ldots, L + L_i - 2, \tag{58}$$

where

$$d(j) = \begin{cases} 0 & \text{if } 0 \leq j < \tau; \\ 1 & \text{if } j = \tau; \\ 0 & \text{otherwise,} \end{cases} \tag{59}$$

represents the target response with delay τ. In matrix form, (58) can be written as

$$\widehat{\mathbf{H}}\mathbf{g} = \mathbf{d}, \tag{60}$$

where $\mathbf{d} = [d(0) \cdots d(L + L_i - 2)]^T$ represents the target response vector and

$$\widehat{\mathbf{H}} = [\widehat{\mathbf{H}}_1 \ \cdots \ \widehat{\mathbf{H}}_M] \tag{61}$$

with $\widehat{\mathbf{H}}_m$ an $(L + L_i - 1) \times L_i$ convolution matrix of $\widehat{\mathbf{h}}_m$:

$$\widehat{\mathbf{H}}_m = \begin{bmatrix} \widehat{h}_m(0) & 0 & \cdots & 0 \\ \widehat{h}_m(1) & \widehat{h}_m(0) & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \widehat{h}_m(L-1) & \cdots & \vdots & \vdots \\ 0 & \widehat{h}_m(L-1) & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & \widehat{h}_m(L-1) \end{bmatrix}. \tag{62}$$

For single channel case, where $M = 1$, (60) is always an over-determined system of equations. The LS solution that minimizes the cost function

$$J = \|\widehat{\mathbf{H}}\mathbf{g} - \mathbf{d}\|_2^2, , \tag{63}$$

has been used for acoustic system equalization [42]. The LS solution is given by

$$\mathbf{g} = \widehat{\mathbf{H}}^{\dagger}\mathbf{d}, \tag{64}$$

where $\{\cdot\}^{\dagger}$ denotes Moore-Penrose pseudo-inverse [61].

When $M > 2$, exact solution(s) that satisfy (60) exist when the following two conditions are both satisfied [43]:

(C2.1) $\widehat{H}_m(z^{-1})$, the z-transforms of the multichannel RIRs $\widehat{\mathbf{h}}_m$ do not have any common zeros.

(C2.2) $L_i \geq L_c$, with $L_c = \lceil \frac{L-1}{M-1} \rceil$ and $\lceil \kappa \rceil$ denotes the smallest integer larger than or equal to κ [62].

As proven in [21], the exact solution to (60) always exists when $L_i \geq L-1$. Unfortunately, this cannot be guaranteed when $L_i \geq L_c$. However, it has been proved in [62] that an exact solution exists for almost all cases.

If both conditions are satisfied, (64) gives the minimum norm solution. If any one or both conditions are violated, (64) gives the LS solution.

Example The following is an example script (`test_ls.m`) that uses `lsinvfilt()` without weighting. System identification errors that are generated using the method described in [63] are added to simulated acoustic impulse responses. The NPM of the pertubated system is -30 dB.

```
% Run equalization algorithm: LS
% This script runs the least squares (LS)
% algorithm for equalization.
%
% Authors: E.A.P. Habets
%
% History: 2009-07-11 Initial version
%
% Copyright (C) Imperial College London 2009-2010
% Version: $Id: test_lsinvfilt.m,v 1.4 2010/03/31 ↩
    10:59:11 ehabets Exp $
```

```
clc
clear
close all

%% Initialization
M = 2;      % number of channels
fs = 8e3;   % sampling frequency
L = 128;    % channel length
Li = L-1;
k = 0;

% Generate AIRs
air.c = 342;  % speed of sound
air.T60 = 0.3;  % reverberation time
air.room_dim = [10; 10; 3];  % room dimension
air.mic_spacing = 0.2;  % microphone spacing (m)
air.src_pos = [100*pi/180 2*pi/180 3];  % source ↩
    location
air.cen_pos = [5; 4; 1.6];  % centre pos. of the array
h = generate_data(M, L, fs, air);

% Truncate and normalize room impulse response
h = h(1:L,:)./h(1,1);

% Add identification errors, desired NPM = -30 dB
ie = generate_sie(h,-30,'prop');
h_tilde = h + ie;

%% Load room impulse responses
% load ../Data/ht_3_C;
% h = -ht(91:91+L-1,1:M)./ht(91,1);
% clear ht;

%% Compute equalization system using MINT (Li=L-1)
g = lsinvfilt(h_tilde, Li, k);

%% Compute equalized response
er = zeros(L+Li-1,1);
for m = 1:M
    er = er + conv(h(:,m), g(:,m));
end

%% Plot results
figure(1);
plot([[h(:,1) ; zeros(Li-1,1)] er]);
title('Impulse Responses');
xlabel('Time (samples)');
ylabel('Amplitude');
legend('First channel','Equalized channel');
```

Running the above script (`test_lsinvfilt.m`) will produce Fig. 9 that shows the impulse response of the first acoustic channel and the equalized response.

Algorithm `lsinvfilt()` performs the following operations:

1. Constructs the matrix $\mathbf{H}$ and the vector $\mathbf{d}$.

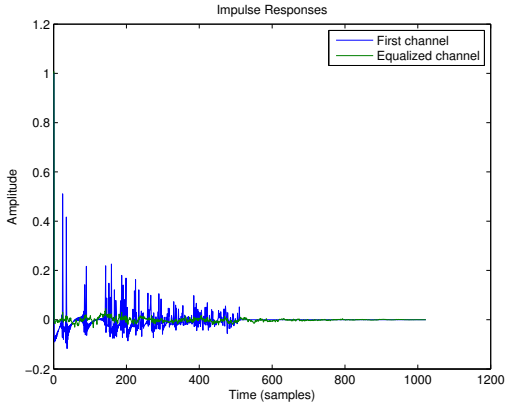


Figure 9: Impulse response of the first channel and the equalized response obtained using the LS algorithm. The NPM of the estimated system is −30 dB.

2. Computes **g** using (64).

See also `wls()`, `wls_iterative()`, `channel_shortening()`

3.2 Weighted least squares algorithm: wls()

Purpose Design a set of equalization filters given the estimated channel impulse responses.

Syntax `[g] = wls(h_hat, Li, k, w)`

The input and output parameters of `wls()` for an FIR system of size $L \times M$ are summarized below.

```
Input Parameters [size]:

 h_hat  :  M impulse responses of length L [L x M]
 Li     :  length of the equalization filters
 k      :  delay of the target response
 w      :  weighting function [L+Li-1 x 1]

Output Parameters [size]:

 g  :  equalization filters [Li x M]
```

Description The weighted LS method [44] is used for the over-determined cases when the condition C2.1 is violated. The weighted LS solution is found by minimizing

$$J = \|\mathbf{W}(\widehat{\mathbf{H}}\mathbf{g} - \mathbf{d})\|_2^2, \tag{65}$$

where $\mathbf{W} = \text{diag}\{\mathbf{w}\}$ with $\mathbf{w} = [w(0) \ \cdots \ w(L + L_i - 2)]^T$, and is given by

$$\mathbf{g} = (\mathbf{W}\widehat{\mathbf{H}})^{\dagger}\mathbf{W}\mathbf{d}. \tag{66}$$

When conditions C2.1 and C2.2 are both satisfied, the solution given by (66) is same as that given by (64) as long as $w(n) \neq 0$, for $n = 0, \ \cdots, L + L_i - 2$ [64].

Example The following is an example script (`test_wls.m`) that uses `wls()`. System identification errors that are generated using the method described in [63] are added to simulated acoustic impulse responses. The NPM of the 'estimated' system is -30 dB.

```
% Run equalization algorithm: WLS
% This script runs the weighted least squares (WLS)
% algorithm for equalization.
%
% Authors: E.A.P. Habets
%
% History: 2009-07-11 Initial version
%
% Copyright (C) Imperial College London 2009-2010
% Version: $Id: test_wls.m,v 1.4 2010/03/31 10:59:11 ↩
    ehabets Exp $

clc
clear
close all

%% Initialization
M = 2;      % number of channels
fs = 8e3;   % sampling frequency
L = 512;    % channel length
Li = L-1;
k = 0;
a = 1.00145;
w = [ones(k+1,1); a.^(1:L+Li-2-k)'-1];

% Generate AIRs
air.c = 342;  % speed of sound
air.T60 = 0.3;  % reverberation time
air.room_dim = [10; 10; 3];  % room dimension
air.mic_spacing = 0.2;  % microphone spacing (m)
air.src_pos = [100*pi/180 2*pi/180 3];  % source ↩
    location
air.cen_pos = [5; 4; 1.6];  % centre pos. of the array
h = generate_data(M, L, fs, air);

% Truncate and normalize room impulse response
h = h(1:L,:)./h(1,1);

% Add identification errors, desired NPM = -30 dB
ie = generate_sie(h,-30,'prop');
h_tilde = h + ie;

% Check the NPM level for h_tilde
display(sprintf('NPM for h_tilde: %d dB\n',20*log10(↩
    npm(h,h_tilde))));

%% Load room impulse responses
% load ../Data/ht_3_C;
% h = -ht(91:91+L-1,1:M)./ht(91,1);
% clear ht;

%% Compute equalization system using MINT (Li=L-1)
g = wls(h_tilde, Li, k, w);
```

```
%% Compute equalized response
er = zeros(L+Li-1,1);
for m = 1:M
    er = er + conv(h(:,m), g(:,m));
end

%% Plot results
figure(1);
plot([[h(:,1) ; zeros(Li-1,1)] er]);
title('Impulse Responses');
xlabel('Time (samples)');
ylabel('Amplitude');
legend('First channel','Equalized channel');

figure(2)
subplot(211);
[md md_all] = magnitude_deviation(er);
plot(md_all);
axis tight;
grid on;
title('Magnitude Distortion');
xlabel('Frequency bins');
ylabel('Distrotion (dB)');

figure(2)
subplot(212);
[pd pd_all] = phase_deviation(er);
plot(pd_all);
axis tight;
grid on;
title('Phase Distortion');
xlabel('Phase (rad)');
ylabel('Distrotion (dB)');
```

Running the above script (`test_wls.m`) will produce Fig. 10 that shows the impulse response of the first acoustic channel and the equalized response.

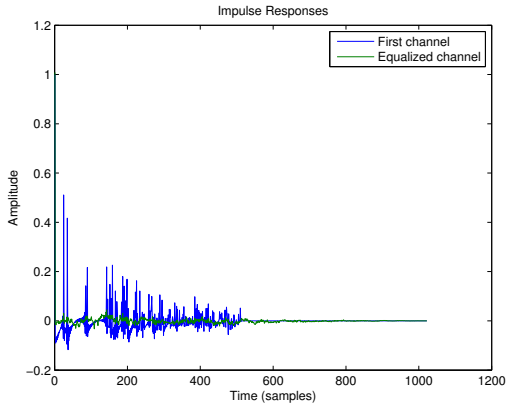


Figure 10: Impulse response of the first channel and the equalized response obtained using the WLS algorithm. The NPM of the estimated system is −30 dB.

Algorithm `wls()` performs the following operations:

1. Constructs the matrix $\widehat{\mathbf{H}}$ and $\mathbf{W}$, and the vector $\mathbf{d}$.
2. Computes $\mathbf{g}$ using (66).

See also `ls()`, `wls_iterative()`, `channel_shortening()`

3.3 Iterative weighted least squares algorithm: wls_iterative()

Purpose Design a set of equalization filters given the estimated channel impulse responses.

Syntax `[g, J] = wls_iterative(h_hat, Li, k, iter, w)`

The input and output parameters of `wls_iterative()` for an FIR system of size $L \times M$ are summarized below.

```
Input Parameters [size]:

 h_hat  :  M impulse responses of length L [L x M]
 Li     :  length of the equalization filters
 k      :  delay of the target response
 iter   :  number of iterations
 w      :  weighting function [L+Li-1 x 1]

Output Parameters [size]:

 g  :  equalization filters [Li x M]
```

Description Equation (66) can also be solved iteratively using a conjugate gradient method [65].

Example The following is an example script (`test_wls_iterative.m`) that uses `wls_iterative()`. System identification errors that are generated using the method described in [63] are added to simulated acoustic impulse responses. The NPM of the 'estimated' system is -30 dB.

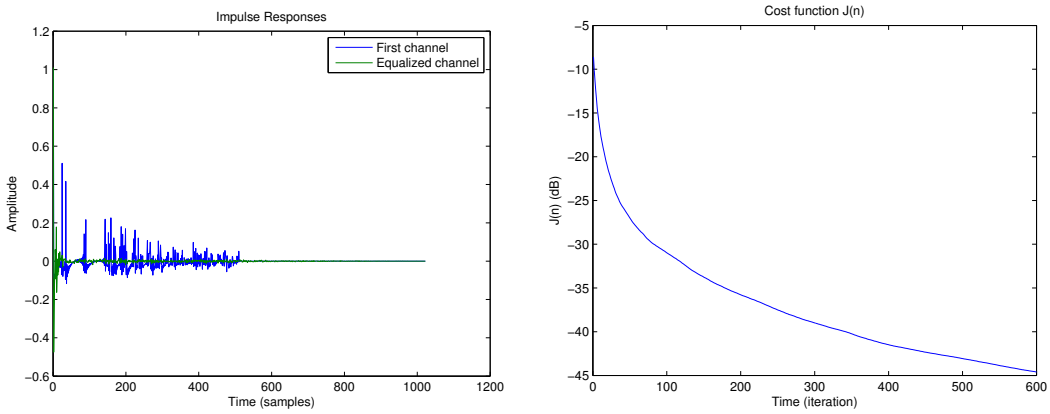
```
% Run equalization algorithm: WLS_iterative
% This script runs the iterative weighted
% least squares (LS) algorithm for equalization.
%
% Authors: E.A.P. Habets
%
% History: 2009-07-11 Initial version
%
% Copyright (C) Imperial College London 2009-2010
% Version: $Id: test_wls_iterative.m,v 1.4 2010/03/31 ↩
    10:59:11 ehabets Exp $

clc
clear
close all

%% Initialization
M = 2;          % number of channels
fs = 8e3;       % sampling frequency
L = 512;        % channel length
```

```
Li = L-1;          % equalizer length
k = 0;             % delay
iter = 600;        % number of iterations
a = 1.00145;       % weight parameter
w = [ones(k+1,1); a.^(1:L+Li-2-k)'-1];  % weighting

% Generate AIRs
air.c = 342;  % speed of sound
air.T60 = 0.3;  % reverberation time
air.room_dim = [10; 10; 3];  % room dimension
air.mic_spacing = 0.2;  % microphone spacing (m)
air.src_pos = [100*pi/180 2*pi/180 3];  % source ↩
    location
air.cen_pos = [5; 4; 1.6];  % centre pos. of the array
h = generate_data(M, L, fs, air);

% Truncate and normalize room impulse response
h = h(1:L,:)./h(1,1);

% Add identification errors, desired NPM = -30 dB
ie = generate_sie(h,-30,'prop');
h_tilde = h + ie;

%% Load room impulse responses
% load ../Data/ht_3_C;
% h = -ht(91:91+L-1,1:M)./ht(91,1);
% clear ht;

%% Compute equalization system using iterative WLS
[g, J] = wls_iterative(h_tilde, Li, k, iter, w);

%% Compute equalized response
er = zeros(L+Li-1,1);
for m = 1:M
    er = er + conv(h(:,m), g(:,m));
end

%% Plot results
figure(1);
plot([[h(:,1) ; zeros(Li-1,1)] er]);
title('Impulse Responses');
xlabel('Time (samples)');
ylabel('Amplitude');
legend('First channel','Equalized channel');

figure(2);
plot(10*log10(J));
title('Cost function J(n)');
xlabel('Time (iteration)');
ylabel('J(n) (dB)');
```

Running the above script (`test_wls_iterative.m`) will produce the figures shown in Fig. 11. Fig. 11(a) shows the impulse response of the first acoustic channel and the equalized response and Fig. 11(b) shows the value of the cost function at each iteration.

Algorithm `wls_iterative()` performs the following operations:

1. Initializes **g**.



(a) (b)

Figure 11: Result obtained using the iterative weighted LS algorithm: (a) impulse response of the first acoustic channel, (b) the value of the cost function at iteration n. The NPM of the estimated system is -30 dB.

2. Constructs the matrix $\widehat{\mathbf{H}}$ and $\mathbf{W}$, and the vector $\mathbf{d}$.
3. Performs `iter` iteration to update $\mathbf{g}$.

See also `ls()`, `wls()`, `channel_shortening()`

3.4 Channel shortening algorithm: channel_shortening()

Purpose Design a set of equalization filters given the estimated channel impulse responses.

Syntax `[g] = channel_shortening(h_hat, Li, Lw, k)`

The input and output parameters of `channel_shortening()` for an FIR system of size $L \times M$ are summarized below.

```
Input Parameters [size]:
       h_hat : M impulse responses of length L [L x M]
       Li    : length of the equalization filters
       k     : delay of the target response
       Lw    : length of the shortened impulse response

Output parameters [size]:
       g     : equalization filters [Li x M]
```

Description Channel shortening techniques [45,46], which are firstly developed for the equalization of digital communication channels have also been used for acoustic system equalization [47]. The equalization filters can be obtained by maximizing the generalized Rayleigh quotient:

$$\mathbf{g} = \arg\max_{\mathbf{g}} \frac{\mathbf{g}^T\mathbf{A}\mathbf{g}}{\mathbf{g}^T\mathbf{B}\mathbf{g}}, \tag{67}$$

where

$$\mathbf{A} = \widehat{\mathbf{H}}^T \mathrm{diag}\{\mathbf{w}_d\}^T \mathrm{diag}\{\mathbf{w}_d\}\widehat{\mathbf{H}} \tag{68}$$

$$\mathbf{B} = \widehat{\mathbf{H}}^T \mathrm{diag}\{\mathbf{w}_u\}^T \mathrm{diag}\{\mathbf{w}_u\}\widehat{\mathbf{H}} \tag{69}$$

with

$$\mathbf{w}_d = [\underbrace{0 \cdots 0}_{\tau} \ \underbrace{1 \cdots 1}_{L_w} \ 0 \cdots 0]^T_{[L+L_i-1\times 1]}$$

$$\mathbf{w}_u = \mathbf{1}_{[L+L_i-1\times 1]} - \mathbf{w}_d,$$

where L_w defines the length of the non-zero part of the the equalized impulse response.

The solution is the eigenvector corresponding to the largest eigenvalue of the generalized eigenvalue problem [47]

$$\mathbf{A}\mathbf{g} = \lambda \mathbf{B}\mathbf{g}. \tag{70}$$

When conditions C2.1 and C2.2 are both satisfied, (67) has multiple solutions $\mathbf{g}$ corresponding to $\lambda = \infty$ and the solution given by (64) is included in the solution space of (67).

Example The following is an example script (`test_channel_shortening.m`) that uses `channel_shortening()`. System identification errors that are generated using the method described in [63] are added to simulated acoustic impulse responses. The NPM of the 'estimated' system is -30 dB.

```
% Run equalization algorithm: Channel_Shortening
% This script runs the channel shortening algorithm
% for equalization
%
% Authors: E.A.P. Habets
%
% History: 2009-07-11 Initial version
%
% Copyright (C) Imperial College London 2009-2010
% Version: $Id: test_channel_shortening.m,v 1.4 ↩
    2010/03/31 10:59:11 ehabets Exp $

clc
clear
close all

%% Initialization
M = 2;      % number of channels
fs = 8e3;   % sampling frequency
L = 512;    % channel length
Li = L-1;
Lw = 128;
k = 0;
iter = 600;
a = 1.00145;
w = [ones(k+1,1); a.^(1:L+Li-2-k)'-1];

% Generate AIRs
air.c = 342;  % speed of sound
air.T60 = 0.3;  % reverberation time
air.room_dim = [10; 10; 3];  % room dimension
air.mic_spacing = 0.2;  % microphone spacing (m)
air.src_pos = [100*pi/180 2*pi/180 3];  % source ↩
    location
air.cen_pos = [5; 4; 1.6];  % centre pos. of the array
```

```
h = generate_data(M, L, fs, air);

% Truncate and normalize room impulse response
h = h(1:L,:)./h(1,1);

% Add identification errors, desired NPM = -30 dB
ie = generate_sie(h,-30,'prop');
h_tilde = h + ie;

%% Load room impulse responses
% load ../Data/ht_3_C;
% h = -ht(91:91+L-1,1:M)./ht(91,1);
% clear ht;

%% Compute equalization system using CS algorithm
g = channel_shortening(h_tilde, Li, Lw, k);

%% Compute equalized response
er = zeros(L+Li-1,1);
for m = 1:M
    er = er + conv(h(:,m), g(:,m));
end

%% Plot results
figure(1);
plot([[h(:,1) ; zeros(Li-1,1)] er]);
title('Impulse Responses');
xlabel('Time (samples)');
ylabel('Amplitude');
legend('First channel','Equalized');
```

Running the above script (`test_channel_shortening.m`) will produce Fig. 12 that shows the impulse response of the first acoustic channel and the equalized response.

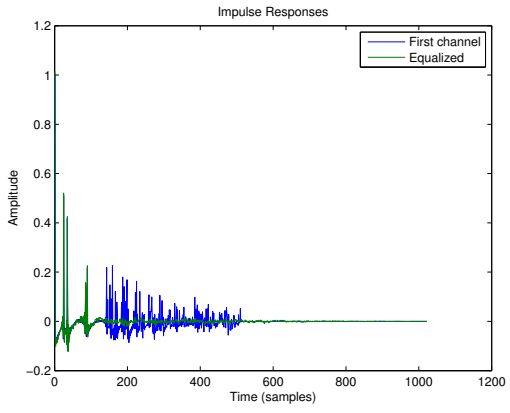


Figure 12: Impulse response of the first channel and the equalized response obtained using the CS algorithm. The NPM of the estimated system is −30 dB.

Algorithm `channel_shortening()` performs the following operations:

1. Constructs the matrix $\mathbf{A}$ and $\mathbf{B}$ defined in (69) and (68), respectively.

2. Computes the solutions of the generalized eigenvalue problem in (70).
3. Returns the eigenvector that corresponds to the largest eigenvalue λ.

See also `ls()`, `wls()`, `wls_iterative()`

4 Performance Evaluation and Analyses

4.1 Normalized Projection Misalignment: npm()

Purpose Compute the normalized project misalignment (NPM) of the true and the estimate system.

Syntax `[npm_val] = npm(h, hhat)`

The input and output parameters of `npm()` are summarized below.

```
Input Parameters [size]:
       h       : true impulse responses [L x M]
       hhat    : estimated impulse responses [L x M]

Output Parameter:
       npm_val : Normalize Projection Misalignment
```

Description The performance of BSI is measured by the similarity between true channel impulse responses and estimated ones. Although not available in practice, the true channel impulse response is available for reference in simulations. The evaluation method should be based on an error measure that is appropriate for the specific problem and independent of how the estimates are derived. Accordingly, define the estimated impulse response for channel m as

$$\widehat{\mathbf{h}}_m(n) = [\widehat{h}_{m,0}(n)\ \widehat{h}_{m,1}(n)\ \ldots\ \widehat{h}_{m,L-1}(n)]^T, \tag{71}$$

so that the normalized projection misalignment [66] given by

$$\mathrm{NPM}(n) = 20\log_{10}\left(\frac{\left\|\mathbf{h} - \kappa(n)\widehat{\mathbf{h}}(n)\right\|_2}{\|\mathbf{h}\|_2}\right)\ \mathrm{dB} \tag{72}$$

can be used as the performance measurement for BSI algorithms, where

$$\mathbf{h} = [\mathbf{h}_1^T,\ \mathbf{h}_2^T,\ \ldots,\ \mathbf{h}_M^T]^T, \tag{73}$$

$$\widehat{\mathbf{h}}(n) = [\widehat{\mathbf{h}}_1^T(n),\ \widehat{\mathbf{h}}_2^T(n),\ \ldots,\ \widehat{\mathbf{h}}_M^T(n)]^T, \tag{74}$$

are concatenated vectors of true and estimated channel responses respectively, and

$$\kappa(n) = \frac{\mathbf{h}^T\widehat{\mathbf{h}}(n)}{\widehat{\mathbf{h}}^T(n)\widehat{\mathbf{h}}(n)}. \tag{75}$$

$\mathrm{NPM}(n)$ can be interpreted as the normalized minimum squared distance from the true channels to the linear manifold of the estimated channels, which is obtained by projecting the former onto the latter [66]. Since classic BSI algorithms estimate the channel responses up to an (unknown) scale factor, the projection error ensures that only the intrinsic misalignment of the channel estimate is taken into account so that the scale factor would not affect the evaluation.

Algorithm `npm()` computes (75) and subsequently (72).

4.2 Magnitude Deviation: magnitude_deviation()

Purpose Compute the difference between the magnitude response of a true and estimated impulse responses.

Syntax `[emag emag_freq] = magnitude_deviation(heq, NFFT)`

The input and output parameters of `magnitude_deviation()` are summarized below.

```
Input Parameters:
       heq       :  equalized impulse response
       NFFT      :  length discrete Fourier transform

Output Parameters:
       emag      :  magnitude deviation
       emag_freq :  magnitude deviation per frequency
```

Description Let us first define the equalized response as

$$b(i) = \sum_{m=1}^{M} h_m(i) * g_m(i) \quad \text{for } i = 0, \ldots, L + L_i - 2. \tag{76}$$

We can then define the magnitude deviation as the standard deviation of the equalized magnitude response of $b(i)$, denoted by $B(k)$ $(0 \leq k \leq K-1)$, as [67]

$$\sigma = \sqrt{\frac{1}{K} \sum_{k=0}^{K-1} \left(10 \log_{10}\left(|B(k)|\right) - \bar{B}\right)^2} \tag{77}$$

with

$$\bar{B} = \frac{1}{K} \sum_{k=0}^{K-1} 10 \log_{10}\left(|B(k)|\right). \tag{78}$$

This measure is scaling independent and equal to zero for exact equalization.

Algorithm `magnitude_deviation()` computes (78), and subsequently (77).

See also `phase_deviation()`

4.3 Phase Distortion: phase_deviation()

Purpose Compute the difference between the phase response of a true and estimated impulse responses.

Syntax `[epha epha_freq] = phase_deviation(heq, NFFT)`

The input and output parameters of `phase_deviation()` are summarized below.

```
Input Parameters:
       heq       :  equalized impulse response
       NFFT      :  length discrete Fourier transform

Output Parameters:
       epha      :  phase deviation
       epha_freq :  phase deviation per frequency
```

Description Linear phase deviation is defined as the deviation of the unwrapped phase from a linear fit to its values and is defined as

$$\Delta = \sqrt{\frac{1}{K}\sum_{k=0}^{K-1}\left(\theta(k) - \bar{\theta}\right)^2} \tag{79}$$

where $\bar{\theta}(k)$ is the least squares linear approximation to the phase at frequency bin k.

Algorithm `phase_deviation()` computes (79).

See also `magnitude_distortion()`

4.4 Generalized Multichannel Clustering: gmc_st()

Purpose The generalized multichannel clustering algorithm, proposed in [68] efficiently extracts clusters of near-common zeros within a specified pairwise distance in the z-plane. Using this algorithm it is possible to quantify the number of common zeros that exist in acoustic systems.

Syntax `[ClustMtx] = gmc_st(zr, tol)`

The input and output parameters of `gmc_st()` are summarized below.

```
Input Parameters [size]:
       zr       : zeros of the channel impulse
                  responses [L-1 x M]
       tol      : tolerence

Output Parameters:
       ClustMtx : a matrix containing number of
                  clusters-by-cluster members
```

Description For a multichannel system, clusters of near-common zeros must satisfy two conditions:

i. The number of zeros within each cluster must correspond to the number of channels for that system with each channel contributing exactly one zero.

ii. All possible pairs of zeros in a cluster must lie within a vicinity δ in terms of their Euclidean distances where $\delta \geq 0$ is defined as the tolerance.

Condition (i) results from the definition of zeros being near common across all channels. Condition (ii) defines the closeness between pairs of near-common zeros. It is worthwhile noting that the channel disparity depends on these pairwise distances [69] and any zero can be a member of more than one cluster. As explained in [68], this implies that classical clustering algorithms such as the k- and c-means algorithms [70] cannot be employed. An illustrative example of two clusters of near-common zeros in the z-plane for a three-channel system satisfying the above conditions is shown in Fig. 13. Symbols $\triangle$, $\square$ and $\circ$ represent zeros for each channel and they lie within pairwise δ vicinity from each other.

Extraction of clusters of near-common zeros in a multichannel system involves the computation of the Euclidean distances between any pair of zeros from different channels. A novel approach to compute these

distances between any two channels was proposed in [68]. It is assumed that each of these high order polynomials has been factorized using efficient factorization algorithms presented in [71]. Defining $\mathbf{h}_m = [h_{m,0} \;\; h_{m,1} \;\; \ldots \;\; h_{m,L-1}]^T$ as the mth channel impulse response of length L we can then express for $z = e^{i2\pi f}$ and $i = \sqrt{-1}$,

$$H_m(z) = h_{m,0} + \ldots + h_{m,L} z^{-L+1} = K \prod_{p=1}^{L-1} \big(z - z_m(p)\big),$$

where f is the normalized frequency and K is the gain constant. The term $z_m(p) = x_m(p) + iy_m(p)$ is the pth zero and its location in the z-plane is defined by $x_m(p)$ and $y_m(p)$ along the real and imaginary axis of the unit circle respectively. It has been shown [72] that as L increases, the radii of these zeros tend toward unity while their angles tend toward a uniform distribution.

The rate of convergence for adaptive BSI algorithms reduces with the reduction of Euclidean distances between the zeros [57]. In addition, the relationship between channel disparity and pairwise distances of the zeros has been established in [69]. To quantify these pairwise distances, we introduce an $L-1 \times L-1$ dissimilarity matrix $\mathbf{D}_{\{m,n\}}$ defined between channels m and n where the pth row and qth column element is given by

$$\begin{aligned} D_{\{m,n\}}(p,q) &= |z_m(p) - z_n(q)| \\ &= \sqrt{\big[x_m(p) - x_n(q)\big]^2 + \big[y_m(p) - y_n(q)\big]^2} \end{aligned} \tag{80}$$

for $p, q = 1, \ldots, L$ and $m \neq n$. Unless the zeros are exactly common, (i) the diagonal elements of $\mathbf{D}_{\{m,n\}}$ are non-zero and (ii) $\mathbf{D}_{\{m,n\}}$ is not symmetric. We next define two $L \times 1$ vectors

$$\mathbf{z}_m = [z_m(1) \; z_m(2) \; \ldots \; z_m(L-1)]^T, \tag{81}$$

$$\mathbf{z}_n = \big[z_n(1) \; z_n(2) \; \ldots \; z_n(L-1)\big]^T \tag{82}$$

containing L zeros of the mth and nth channel respectively. Defining $\widetilde{\mathbf{Z}}_m = \mathbf{Z}_m \odot \mathbf{Z}_m$ with $\odot$ being the Hadamard product, $\mathbf{Z}_m = \mathbf{z}_m \mathbf{1}^T$, $\mathbf{Z}_n = \mathbf{1}\mathbf{z}_n^T$ and $\mathbf{1}_{[L-1\times 1]} = [1 \;\; 1 \;\; \ldots \;\; 1]^T$, computation of (80) for all p and q can be efficient using

$$\mathbf{D}_{\{m,n\}} = \Big[\Big|\widetilde{\mathbf{Z}}_m - 2\mathbf{z}_m\mathbf{z}_n^T + \widetilde{\mathbf{Z}}_n\Big|\Big]^{\circ\frac{1}{2}}, \tag{83}$$

with $[\cdot]^{\circ\frac{1}{2}}$ and $|\cdot|$ being the Hadamard square root and elemental absolute respectively. To illustrate the validity of (83), we write

$$\begin{aligned} &\widetilde{\mathbf{Z}}_m - 2\mathbf{z}_m\mathbf{z}_n^T + \widetilde{\mathbf{Z}}_n \\ &= \begin{bmatrix} \big(z_m(1) - z_n(1)\big)^2 & \cdots & \big(z_m(1) - z_n(L-1)\big)^2 \\ \vdots & \ddots & \vdots \\ \big(z_m(L-1) - z_n(1)\big)^2 & \cdots & \big(z_m(L-1) - z_n(L-1)\big)^2 \end{bmatrix}. \end{aligned} \tag{84}$$

Let $a = x_m(p) - x_n(q)$ and $b = y_m(p) - y_n(q)$. Invoking Euler's identity $a + ib = re^{i\theta}$ where $r = \sqrt{a^2 + b^2}$ and $\theta = \tan^{-1}(b/a)$, we have $\big|(a+ib)^2\big| = \big|r^2 e^{i2\theta}\big| = r^2$, from which we obtain the important result

$$\begin{aligned} \mathbf{D}_{\{m,n\}} &= \Big[\big|\widetilde{\mathbf{Z}}_m - 2\mathbf{z}_m\mathbf{z}_n^T + \widetilde{\mathbf{Z}}_n\big|\Big]^{\circ\frac{1}{2}} \\ &= \begin{bmatrix} \sqrt{|(z_m(1) - z_n(1))^2|} & \cdots & \sqrt{|(z_m(1) - z_n(L-1))^2|} \\ \vdots & \ddots & \vdots \\ \sqrt{|(z_m(L-1) - z_n(1))^2|} & \cdots & \sqrt{|(z_m(L-1) - z_n(L-1))^2|} \end{bmatrix} \end{aligned}$$

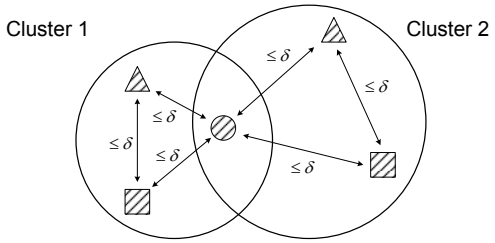


Figure 13: An example of two clusters for a three-channel system in the z-plane.

hence verifying the validity of (83). For $m \neq n$, since all pairwise distances are computed only once, no computational redundancy occurs making this computation efficient.

Employing $\mathbf{D}_{\{m,n\}}$ between any two out of M channels, consider the case where there are c_{mn} clusters of near-common zeros between channels m and n. We next define a sub-cluster group matrix $\boldsymbol{\mathcal{C}}_{\{m,n\}}$ containing these c_{mn} clusters. This $c_{mn} \times 2$ matrix can be obtained by searching within elements in $\mathbf{D}_{\{m,n\}}$ for indices p and q such that

$$\boldsymbol{\mathcal{C}}_{\{m,n\}} = \arg D_{\{m,n\}}(p,q) \leq \delta \tag{85}$$

is satisfied. We next denote c_t as the total number of clusters for the M-channel system. We define cluster matrix $\boldsymbol{\mathcal{C}}_{\{1:M\}}$ of dimension $c_t \times M$ for the whole system where each row contains a cluster with M elements each containing indices corresponding to elements in $\{\mathbf{z}_1 \;\; \mathbf{z}_2 \;\; \ldots \;\; \mathbf{z}_M\}$ for the respective channels 1 to M. We employ (85) across each pair of channels selected from M. The aim of the generalized multichannel clustering (GMC) algorithm is to obtain $\boldsymbol{\mathcal{C}}_{\{1:M\}}$ using sub-cluster groups $\boldsymbol{\mathcal{C}}_{\{m,n\}}$.

Example The following is an example script (`test_gmc.m`) of running `gmc_st()`.

```
% Runs the GMC_ST algorithms to find
% the number of near-common zeros
%
% References:
%   [1] J. Allen and D. Berkley, "Image method for
%       efficiently simulating small room acoustics",
%       JASA, 1979.
%
% Authors: E.A.P. Habets
%
% History: 2009-07-14 Initial version
%
% Copyright (C) Imperial College London 2009-2010
% Version: $Id: test_gmc.m,v 1.3 2010/02/12 11:40:12 ↩
    ehabets Exp $

clc;
clear;
close all;

%% Initialization
M = 2;          % number of channels
fs = 8e3;       % sampling frequency
```

```
L = 512;      % channel length
tol = 0.005; % tolerance

% Generate AIRs
air.c = 342;  % speed of sound
air.T60 = 0.3;  % reverberation time
air.room_dim = [5; 6; 3];  % room dimension
air.mic_spacing = 0.2;  % microphone spacing (m)
air.src_pos = [100*pi/180 2*pi/180 2];  % source ↩
    location
air.cen_pos = [2.5; 2; 1.6];  % centre pos. of the ↩
    array
h = generate_data(M, L, fs, air);

% Find zeros of the acoustic impulse responses
for ii=1:M;
    zr(:, ii) = roots(h(:, ii));
end

% Find number of near-common zeros using GMC-DC
% display(sprintf('\nRunning GMC-DC...'))
% ZeroMtx = gmc_dc(zr, tol);
% display(sprintf('GMC-DC: %d\n', size(ZeroMtx, 1)));
% clear ZeroMtx;

% Find clusters of near-common zeros using GMC-ST
display(sprintf('\nRunning GMC-ST...'))
[ClustMtx, ClustMem, ClustNum]  = gmc_st(zr, tol);
display(sprintf('GMC-ST: %d',ClustNum));
```

Algorithm The GMC Search-and-Trim (GMC-ST) algorithm first computes $\mathbf{D}_{\{m,n\}}$ for all possible pairs of channels with $m = 1, \ldots, M-1$ and $n = m+1$. This employs (83) a total of $0.5M(M-1)$ times. Invoking (85), this algorithm then aims to extract $\boldsymbol{\mathcal{C}}_{\{1:M\}}$ from these $0.5M(M-1)$ sub-cluster groups $\boldsymbol{\mathcal{C}}_{\{m,n\}}$ using an efficient search technique. This is achieved by first selecting the sub-cluster group

$$\boldsymbol{\mathcal{C}}_{\{m_s,n_s\}} = \min_{c_{mn}} \Big\{ \boldsymbol{\mathcal{C}}_{\{1,2\}} \;\; \boldsymbol{\mathcal{C}}_{\{1,3\}} \; \ldots \; \boldsymbol{\mathcal{C}}_{\{M-1,M\}} \Big\}, \tag{86}$$

having the smallest c_{mn} as a reference group. This is equivalent to finding the two channels m_s and n_s having the smallest number of sub-clusters. Since near-common zeros must satisfy condition (i), GMC-ST begins its search from $\boldsymbol{\mathcal{C}}_{\{m_s,n_s\}}$. For each row of $\boldsymbol{\mathcal{C}}_{\{m_s,n_s\}}$, GMC-ST initializes a row vector $\mathbf{R} = [r(1) \; \ldots \; r(M)]$ with only two non-empty elements $r(m_s) = p$ and $r(n_s) = q$ where p and q are two elements obtained from each row in $\boldsymbol{\mathcal{C}}_{\{m_s,n_s\}}$. The next stage is to search, for row vector $\mathbf{r}$ of $\mathbf{R}$, the remaining $M-2$ empty elements. This search space is confined within $\big\{ \boldsymbol{\mathcal{C}}_{\{1,2\}} \;\; \boldsymbol{\mathcal{C}}_{\{1,3\}} \; \ldots \; \boldsymbol{\mathcal{C}}_{\{M-1,M\}} \big\}$ excluding $\boldsymbol{\mathcal{C}}_{\{m_s,n_s\}}$ since, from (85), only zeros within these groups are all within tolerance δ. If k elements are found, then $\mathbf{R}$ is updated as

$$\widetilde{\mathbf{R}} = \mathbf{1}_{[k\times 1]}\mathbf{r}, \qquad \mathbf{R} = [\widetilde{\mathbf{R}}^T \;\; \mathbf{R}^T]^T, \tag{87}$$

where $\mathbf{1}_{[k\times 1]} = [1 \; \ldots \; 1]^T$. This implies that the zeros belong to k different clusters. The trimming process then ensures that all pair-wise elements for each row in $\mathbf{R}$ can be found within the search space

$\{\boldsymbol{\mathcal{C}}_{\{1,2\}} \ \boldsymbol{\mathcal{C}}_{\{1,3\}} \ \ldots \ \boldsymbol{\mathcal{C}}_{\{M-1,M\}}\}$ in order to satisfy condition (ii). If this condition is violated, the entire row is deleted and the search-and-trim process is repeated until every element in each row of $\mathbf{R}$ is found or all rows have been deleted.

4.5 Energy Decay Curve and Derived Measures: edc()

Purpose The energy decay curve (EDC) is a measure of the decay of an RIR or EIR, shown in [73] to be identical to the ensemble average decay of a system excited with white noise. It is inversely proportional to reverberation time [74] and is used in the derivation of a set of objective measures for an RIR.

Syntax `[decay,...] = edc(h,fs,...)`

The input and output parameters of `edc()` are summarized below. If no outputs are specified then the EDC is plotted.

```
Input Parameters [size]:
       h    : room impulse response
              [L-1 x 1]
       fs   : Sampling frequency (Hz)
       meas : [Optional] Any number of strings containing
              'Txx': Reverberation time (ms)
              'Cxx': Clarity index (early-to-late ratio) (dB)
              'Dxx': Deutlichkeit (early-to-total sound energy) (dB)
              where xx specifies the argument in dB for 'Txx' and in
              ms for 'Cxx' and 'Dxx'. Meas can also contain cell arrays.

Output Parameters [size]:
       decay: Normalized energy decay curve in dB
              [Lx1]
       meas : Measures in order of input argument
```

Description The EDC of an RIR is calculated by [73]

$$\mathrm{EDC}(i) = \frac{1}{\|\mathbf{h}\|_2^2} \sum_{n=i}^{L-1} h^2(n), \ i \in \{0, 1, \ldots, L-1\}. \tag{88}$$

The measure T_{60} the time for the energy decay curve (EDC) of an RIR to drop by 60 dB of its maximum value [74]. Similarly, the measure T_{30} is commonly used in the context of EIRs.

The early-to-late reverberation reverberation ratio (ELR), also known as the Clarity Index, is highly-correlated with the intelligibility of reverberant speech and is defined as [74]

$$\mathrm{C}_{n_e} = \frac{\sum_{n=0}^{n_e} h^2(n)}{\sum_{n=n_e+1}^{\infty} h^2(n)} \tag{89}$$

$$= \frac{\mathrm{EDC}(0) - \mathrm{EDC}(n_e+1)}{\mathrm{EDC}(n_e+1)}. \tag{90}$$

Deutlichkeit, the early-to-total sound energy, is defined as [74]

$$\mathrm{D}_{n_e} = \frac{\sum_{n=0}^{n_e} h^2(n)}{\sum_{n=0}^{\infty} h^2(n)} \tag{91}$$

$$= \frac{\mathrm{EDC}(0) - \mathrm{EDC}(n_e+1)}{\mathrm{EDC}(0)}, \tag{92}$$

The constant n_e is often chosen to be $n_e = 50\ \text{ms} \cdot f_s$. EDC, C_{n_e} and D_{n_e} outputs are expressed on a log scale.

5 Auxiliary Functions

5.1 Generate Data: generate_data()

Purpose This function computes the room impulse responses and generates WGN data with additive noise.

Syntax `[h, x, tau] = generate_data(M, L, fs, air, N, SNR)`

The input and output parameters of `generate_data()` are summarized below.

```
Input Parameters:
       M      : number of channels
       L      : filter length
       fs     : sample frequency
       air    : struct with acoustic impulse response
                parameters
       N      : data length
       SNR    : average signal to noise ratio across
                sensor signals (optional: default 30 dB)

Output Parameters:
       h      : acoustic impulse responses
       x      : noisy and reverberant sensor signals
```

5.2 Generate System Identification Errors: generate_sie()

Purpose This function generates system identification errors.

Syntax `[ie]  = generate_sie(h, npm, mode, T60, fs)`

The input and output parameters of `generate_sie()` are summarized below.

```
Input Parameters:
       h      : system impulse responses
       npm    : desired NPM level in dB
       mode   : the shape of errors. Three options are
                'wgn'  white Gaussian noise like errors;
                'prop' WGN but proportional to h;
                'damp' damping errors, damping factor same as h
       T60    : the reverberation time of h, only needed for
                generating damping errors
       fs     : sampling frequency, only needed for generating
                damping errors (optional: default 8000)

Output Parameters:
       ie     : the representation of identification errors
```

5.3 Remove Direct-Path Propagation: rdp()

Purpose This function removes the smallest direct-path propagation of room impulse responses generated by image method.

Syntax `[h, dl] = rdp(h_unc, tau, stat)`

The input and output parameters of `rdp()` are summarized below.

```
Input Parameters:
       h_unc   : original room impulse responses
       tau     : TDOAs with respect to first sensor in
                 (fractional) samples
       seed    : seed for generating padding random taps

Output Parameters:
       h       : updated room impulse responses
```

5.4 Sensor Positions of a Uniform Linear Array: ula_pos()

Purpose This function positions a ULA such that its centroid is in the middle of the room.

Syntax `micPos = ula_pos(micSpc,micNum,CenPos)`

The input and output parameters of `ula_pos()` are summarized below.

```
Input Parameters:
       micSpc  : microphone spacing
       micNum  : number of microphones
       CenPos  : center position of the array

Output Parameters:
       micPos  : a 3-by-micNum vector containing mic positions
```

6 Conclusions

This BSIE toolbox contains various state-of-the-art algorithms for blind system identification and system equalization. In addition, it contains performance evaluation and analyses functions.

Acknowledgment

The authors would like to thank Mr. Mark Thomas for proofreading the manuscript.

References

[1] Y. Sato, "A method of self-recovering equalization for multilevel amplitude-modulation," *IEEE Trans. Commun.*, vol. 6, pp. 679–682, Jun. 1975.

[2] G. B. Giannakis, Y. Hua, P. Stoica, and L. Tong, Eds., *Signal Processing Advances in Wireless and Mobile Communications, Volume I: Trends in Channel Estimation and Equalization.* Upper Saddle River, N.J.: Prentice Hall PTR, 2000.

[3] O. Lee, Y. Son, and K. Kim, "Underwater digital communication using acoustic channel estimation," in *Proc. Oceans*, vol. 4, Oct. 2002, pp. 2453–2456.

[4] H. Luo and Y. Li, "The application of blind channel identification techniques to prestack seismic deconvolution," *Proc. IEEE*, vol. 86, no. 10, pp. 2082–2089, Oct. 1998.

[5] L. Tong and S. Perreau, "Multichannel blind identification: from subspace to maximum likelihood methods," *Proc. IEEE*, vol. 86, no. 10, pp. 1951–1968, 1998.

[6] S. Haykin, *Blind Deconvolution*, 4th ed., ser. Prentice-Hall Information and System Sciences. Prentice-Hall, 1994.

[7] J. K. Tugnait, "Channel estimation and equalization using higher-order statistics," in *Signal Processing Advances in Wireless and Mobile Communications, Volume I: Trends in Channel Estimation and Equalization*, G. B. Giannakis, Y. Hua, P. Stoica, and L. Tong, Eds. Upper Saddle River, N.J.: Prentice Hall PTR, 2000, ch. 1, pp. 1–40.

[8] R. Pan and C. L. Nikias, "The complex cepstrum of higher order cumulants and non-minimum phase system identification," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 36, no. 2, pp. 186–205, Feb. 1988.

[9] R. Godfrey and F. Rocca, "Zero memory non-linear deconvolution," *Geophysical Prospecting*, vol. 29, no. 2, pp. 189–228, 1981.

[10] S. Bellini, "Bussgang techniques for blind deconvolution and equalization," in *Blind deconvolution*, S. Haykin, Ed. Englewood Cliffs, N.J.: Prentice Hall, Inc., 1994, ch. 2, pp. 8 – 59.

[11] Y. Huang, J. Benesty, and J. Chen, "Identification of acoustic MIMO systems: Challenges and opportunities," *Signal Processing*, vol. 6, no. 86, pp. 1278–1295, 2006.

[12] S. Haykin, *Adaptive Filter Theory*, 4th ed. Prentice-Hall, 2002.

[13] W. A. Gardner, "A new method of channel identification," *IEEE Trans. Commun.*, vol. 39, no. 6, pp. 813–817, Jun. 1991.

[14] L. Tong, G. Xu, and T. Kailath, "A new approach to blind identification and equalization of multipath channels," in *Proc. Asilomar Conf. on Signals, Systems and Computers*, vol. 2, Pacific Grove, CA, USA, Nov. 1991, pp. 856–860.

[15] P. Loubaton, E. Moulines, and P. Regalia, "Subspace method for blind identification and deconvolution," in *Signal Processing Advances in Wireless and Mobile Communications, Volume I: Trends in Channel Estimation and Equalization*, G. B. Giannakis, Y. Hua, P. Stoica, and L. Tong, Eds. Upper Saddle River, N.J.: Prentice Hall PTR, 2000, ch. 3, pp. 63–112.

[16] H. Liu, G. Xu, and L. Tong, "A deterministic approach to blind equalization," in *Proc. Asilomar Conf. on Signals, Systems and Computers*, vol. 1, Nov. 1993, pp. 751–755.

[17] L. Tong, G. Xu, and T. Kailath, "Blind identification and equalization based on second-order statistics: a time domain approach," *IEEE Trans. Inf. Theory*, vol. 40, no. 2, pp. 340–349, Mar. 1994.

[18] L. Tong, G. Xu, B. Hassibi, and T. Kailath, "Blind channel identification based on second-order statistics: a frequency-domain approach," *IEEE Trans. Inf. Theory*, vol. 41, no. 1, pp. 329–334, Jan. 1995.

[19] M. I. Gürelli and C. L. Nikias, "EVAM: An eigenvector-based algorithm for multichannel blind deconvolution of input colored signals," *IEEE Trans. Signal Process.*, vol. 43, no. 1, pp. 134–149, Jan. 1995.

[20] G. Xu, H. Liu, L. Tong, and T. Kailath, "A least-squares approach to blind channel identification," *IEEE Trans. Signal Process.*, vol. 43, no. 12, pp. 2982–2993, Dec. 1995.

[21] E. Moulines and J. Laroche, "Non-parametric techniques for pitch-scale and time-scale modification of speech," *Speech Communication*, vol. 16, pp. 175–205, Feb. 1995.

[22] D. Slock, "Blind fractionally-spaced equalization, perfect reconstruction filterbanks, and multilinear prediction," in *Proc. IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 4, 1994, pp. 585–588.

[23] Y. Hua, "Fast maximum likelihood for blind identification of multiple FIR channels," *IEEE Trans. Signal Process.*, vol. 44, no. 3, pp. 661–672, Mar. 1996.

[24] Y. Huang, J. Benesty, and J. Chen, "A blind channel identification-based two-stage approach to separation and dereverberation of speech signals in a reverberant environment," *IEEE Trans. Speech Audio Process.*, vol. 13, no. 5, pp. 882–895, Sep. 2005.

[25] J. Benesty, J. Chen, and Y. Huang, "On the importance of the Pearson correlation coefficient in noise reduction," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 16, pp. 757–765, May 2008.

[26] J. Benesty, T. Gänsler, D. R. Morgan, M. M. Sondhi, and S. L. Gay, *Advances in Network and Acoustic Echo Cancellation*. Springer-Verlag, 2001.

[27] Y. Huang, J. Benesty, and J. Chen, *Acoustic MIMO Signal Processing*. Springer-Verlag Berlin Heidelberg, 2006.

[28] Y. Huang and J. Benesty, "Adaptive multi-channel least mean square and Newton algorithms for blind channel identification," *Signal Processing*, vol. 82, pp. 1127–1138, Aug. 2002.

[29] ——, "A class of frequency-domain adaptive approaches to blind multichannel identification," *IEEE Trans. Signal Process.*, vol. 51, no. 1, pp. 11–24, Jan. 2003.

[30] S. Gannot and M. Moonen, "Subspace methods for multimicrophone speech dereverberation," *EURASIP Journal on Applied Signal Processing*, vol. 2003, no. 11, pp. 1074–1090, 2003.

[31] J. Benesty, Y. Huang, and J. Chen, "An exponentiated gradient adaptive algorithm for blind identification of sparse SIMO systems," in *Proc. IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 2, 2004, pp. 829–832.

[32] Y. Huang, J. Benesty, and J. Chen, "A blind channel identification-based two-stage approach to separation and dereverberation of speech signals in a reverberant environment," *IEEE Trans. Speech Audio Process.*, vol. 13, no. 5, pp. 882–895, Sep. 2005.

[33] Q. Liu, B. Champagne, and P. Kabal, "A microphone array processing technique for speech enhancement in a reverberant space," *Speech Communication*, vol. 18, no. 4, pp. 317–334, 1996.

[34] N. D. Gaubitch, K. Hasan, and P. A. Naylor, "Noise robust adaptive blind channel identification using spectral constraints," in *Proc. IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 5, Toulouse, France, May 2006.

[35] M. K. Hasan, N. M. Hossain, and P. A. Naylor, "Autocorrelation model-based identification method for ARMA systems in noise," *IEE Proc. Vision Image Signal Processing*, vol. 152, no. 5, pp. 520 – 526, 2005.

[36] M. K. Hasan and P. A. Naylor, "Analyzing effect of noise on LMS-type approaches to blind estimation of SIMO channels: robustness issue," in *Proc. European Signal Processing Conf. (EUSIPCO)*, Florence, Italy, Sep. 2006.

[37] R. Ahmad, A. W. H. Khong, M. K. Hasan, and P. A. Naylor, "An extended normalized multichannel FLMS algorithm for blind channel identification," in *Proc. European Signal Processing Conf. (EUSIPCO)*, Florence, Italy, Sep. 2006.

[38] N. D. Gaubitch, "Blind identification of acoustic systems and enhancement of reverberant speech," Ph.D. dissertation, Imperial College London, 2006.

[39] R. Ahmad, A. W. H. Khong, and P. A. Naylor, "A practical adaptive blind multichannel estimation algorithm with application to acoustic impulse response," in *Proc. IEEE Intl. Conf. Digital Signal Processing (DSP)*, Cardiff, Wales, 2007, pp. 31–34.

[40] A. P. Liavas, P. A. Regalia, and J.-P. Delmas, "Blind channel approximation: Effective length determination," *IEEE Trans. Signal Process.*, vol. 47, no. 12, pp. 3336–3344, Dec. 1999.

[41] S. T. Neely and J. B. Allen, "Invertibility of a room impulse response," *J. Acoust. Soc. Am.*, vol. 66, no. 1, pp. 165–169, Jul. 1979.

[42] J. Mourjopoulos, P. Clarkson, and J. Hammond, "A comparative study of least-squares and homomorphic techniques for the inversion of mixed phase signals," in *Proc. IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 7, May 1982, pp. 1858–1861.

[43] M. Miyoshi and Y. Kaneda, "Inverse filtering of room acoustics," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 36, no. 2, pp. 145–152, Feb. 1988.

[44] M. Hofbauer, "Optimal linear separation and deconvolution of acoustical convolutive mixtures," Ph.D. dissertation, Swiss Federal Institute of Technology, 2005.

[45] P. J. W. Melsa, R. C. Younce, and C. E. Rohrs, "Impulse response shortening for discrete multitone transceivers," *IEEE Trans. Commun.*, vol. 44, pp. 1662–1672, 1996.

[46] R. K. Martin, K. Vanbleu, M. Ding, G. Ysebaert, M. Milosevic, B. L. Evans, M. Moonen, and C. R. Johnson Jr., "Unification and evaluation of equalization structures and design algorithms for discrete multitone modulation systems," *IEEE Trans. Signal Process.*, vol. 53, pp. 3880–3894, 2005.

[47] M. Kallinger and A. Mertins, "Multi-channel room impulse response shaping - a study," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 2006.

[48] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[49] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Autom. Control*, vol. AC-19, no. 6, pp. 716–723, Dec. 1974.

[50] J. Tugnait, "On blind identifiability of multipath channels using fractional sampling and second-order cyclostationary statistics," *IEEE Trans. Inf. Theory*, vol. 41, no. 1, pp. 308–311, 1995.

[51] W. Ding and H. Kasuya, "A novel approach to the estimation of voice source and vocal tract parameters from speech signals," in *Proc. Intl. Conf. on Spoken Lang. Processing (ICSLP)*, vol. 2, 1996, pp. 1257–1260.

[52] Y. Hua and M. Wax, "Strict identifiability of multiple FIR channels driven by an unknown arbitrary sequence," *IEEE Trans. Signal Process.*, vol. 44, no. 3, pp. 756–759, Mar. 1996.

[53] V. U. Reddy, C. B. Papadias, and A. J. Paulraj, "Blind identifiability of certain classes of multipath channels from second-order-statistics using antenna arrays," *IEEE Signal Process. Lett.*, vol. 4, no. 5, pp. 138–141, May 1997.

[54] E. Serpedin and G. Giannakis, "A simple proof of a known blind channel identifiability result," *IEEE Trans. Signal Process.*, vol. 47, pp. 591–593, Feb. 1999.

[55] Y. Hua and J. K. Tugnait, "Blind identifiability of FIR-MIMO systems with colored input using second order statistics," *IEEE Signal Process. Lett.*, vol. 7, no. 12, pp. 348–350, Dec. 2000.

[56] C. Avendano, J. Benesty, and D. R. Morgan, "A least squares component normalization approach to blind channel identification," in *Proc. IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 4, Mar. 1999, pp. 1797–1800.

[57] X. Lin, N. D. Gaubitch, and P. A. Naylor, "Two-stage blind identification of SIMO systems with common zeros," in *Proc. European Signal Processing Conf. (EUSIPCO)*, Florence, Italy, Sep. 2006.

[58] M. Haque and M. Hasan, "Noise robust multichannel frequency-domain LMS algorithms for blind channel identification," *IEEE Signal Process. Lett.*, vol. 15, pp. 305–308, 2008.

[59] R. Ahmad, A. W. H. Khong, and P. A. Naylor, "Proportionate frequency domain adaptive algorithms for blind channel identification," in *Proc. IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 5, Toulouse, France, 2006, pp. 2932–2935.

[60] J. S. Soo and K. K. Pang, "Multidelay block frequency domain adaptive filter," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 38, pp. 373–376, Feb. 1990.

[61] R. Rado, "Note on generalized inverse of matrices," *Proc. Cambridge Philos. Soc.*, vol. 52, pp. 600–601, 1956.

[62] G. Harikumar and Y. Bresler, "FIR perfect signal reconstruction from multiple convolutions: minimum deconvolver orders," *IEEE Trans. Signal Process.*, vol. 46, pp. 215–218, 1998.

[63] W. Zhang and P. A. Naylor, "An algorithm to generate representations of system identiï??cation errors," *Research letters in signal processing*, vol. 2008, 2008.

[64] T. N. E. Greville, "Note on the generalized inverse of a matrix product," *SIAM Review*, vol. 8, pp. 518–521, 1966.

[65] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. MD: John Hopkins University Press, Balimore, 1996.

[66] D. R. Morgan, J. Benesty, and M. Sondhi, "On the evaluation of estimated impulse responses," *IEEE Signal Process. Lett.*, vol. 5, no. 7, pp. 174–176, Jul. 1998.

[67] B. D. Radlović and R. A. Kennedy, "Nonminimum-phase equalization and its subjective importance in room acoustics," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 6, pp. 728–737, Nov. 2000.

[68] A. W. H. Khong, X. Lin, and P. A. Naylor, "Algorithms for identifying clusters of near-common zeros in multichannel blind system identification and equalization," in *Proc. IEEE Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Las Vegas, USA, Apr. 2008, pp. 389–392.

[69] I. Fijalkow, "Multichannel equalization lower bound: a function of channel noise and disparity," in *Proc. IEEE Signal Process. Workshop on Statistical Signal and Array Process. (SSAP)*, Jun. 1996, pp. 344–347.

[70] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[71] G. Sitton, C. Burrus, J. Fox, and S. Treitel, "Factoring very-high-degree polynomials," *IEEE Signal Process. Mag.*, vol. 20, no. 6, pp. 27 – 42, Nov. 2003.

[72] C. P. Hughes and A. Nikeghbali, "The zeros of random polynomials cluster uniformly near the unit circle," *ArXiv Mathematics e-prints*, Jun. 2004.

[73] M. R. Schroeder, "New method of measuring reverberation time," *J. Acoust. Soc. Am.*, vol. 37, pp. 409–412, 1965.

[74] H. Kuttruff, *Room Acoustics*, 4th ed. Taylor & Frances, 2000.