Contents

Chapter 1

Utilisation

1.1 Introduction

This document describes the version lite of the estimation process. The inputs are the two signals of any length which represent the signals from the system under test (SUT) and the system of reference (SREF). It performs the ratio between the auto-spectrum of SUT divided by the cross-spectrum of SREF on a given list of frequencies. The length of the list of frequencies is denoted N.

The ratio denoted R_{sup} is not corrected by the response of the SREF, nor the response of the noise reduction system.

The computation needs the description of the filter bank which is provided by a Matlab structure. That can be removed. The call function is

```
function [Rsup, freqslin, STDmoduleRlin, ...
    STDphaseRlin_rd, nboverTHlin] = ...
    estimSUTlite ...
    (signals, structfiltercharacteristics, frequencylist_Hz, ...
    Fs_Hz, MSCthreshold, trimpercent)
%=====================================================================
% Synopsis:
% [Rsup, freqslin, STDmoduleRlin, ...
%     STDphaseRlin_rd, nboverTHlin] = ...
%     estimSUTlite ...
%     (signals, structfiltercharacteristics, frequencylist_Hz, ...
%     Fs_Hz, MSCthreshold, trimpercent)
%==============
% Inputs:
%     - signals : T x 2
%     - structfiltercharacteristics (FB structure)
%         see document
%     - frequencylist_Hz: array N x 1 of the selected frequencies
%       in Hz. N can take any value under Fs_Hz/2
%     - Fs_Hz: sampling frequency in Hz
%     - MSCthreshold:
%     - trimpercent: percent of values keptfor averaging
%==============
% Outputs:
%     - Rsup: array N x 1 of the estimated ratios
%     - freqslin: array N x 1 of the selected frequencies
%       in Hz. almost the same as frequencylist_Hz, except if some
%       are outside of the FB bandwidths.
%     - STDmoduleR: array N x 1 of the STD on the module of Rsup
%     - STDphaseR_rd: array N x 1 of the STD on the phase of Rsup
%     - nboverTH: array N x 1 of the number of values over the threshold
%=====================================================================
```

The outputs are the

- list of the N values of R_{sup} averaged on the full signals
- list of the N values of STDs of the module and the phase of R_{sup} performed on the full signals,
- list of the N count number over the MSC threshold.

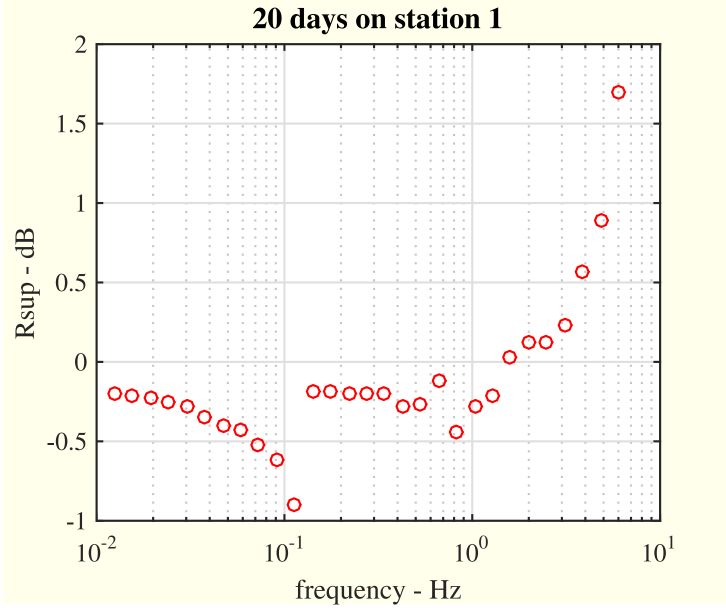
To play you can use the program `estimationwithFBlite.m`

1.2 Examples

1.2.1 Example 1

The figure corresponds to the following days on station 1:

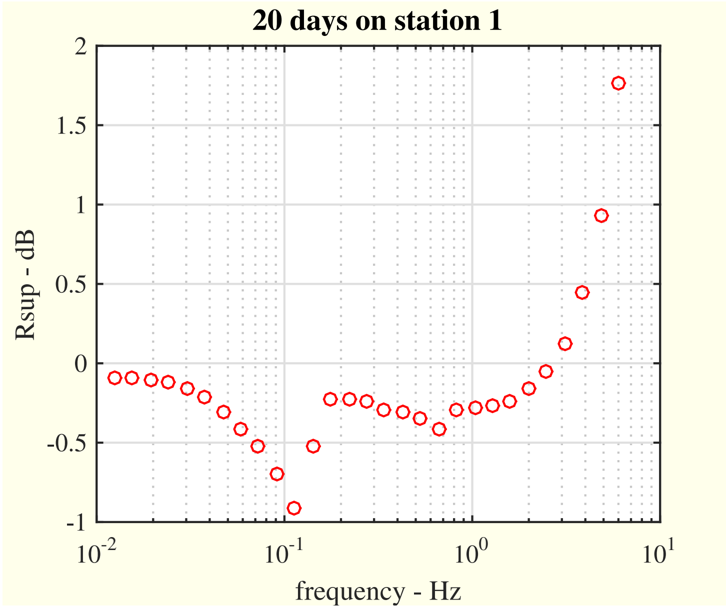
days:
`2015/06/05-06`
`2015/06/11-12`
`2015/07/01-02`
`2015/07/13-14`
`2015/08/03-04`
`2015/08/07-08`
`2015/08/09-10`
`2015/08/19-20`
`2015/09/29-30`
`2015/10/25-26`



1.2.2 Example 2

The figure corresponds to the following days on station 1:

days:
tt 2015/07/11-12
tt 2015/07/13-14
tt 2015/07/21-22
tt 2015/07/27-28
tt 2015/08/23-24
tt 2015/09/07-08
tt 2015/09/11-12
tt 2015/09/15-16
tt 2015/10/03-04
tt 2015/10/21-22



Chapter 2

Codes

2.1 Main program

```
%===================== estimationwithFBlite.m ==========================
% Program estimates SUT response from the signals
% located in the directory directorysignals.
% The signals correspond to the pair of sensors
% SUT/SREF during a given duration T, typically 48 hours.
% Here we concatene NBCONCAT randomly chosen files.
%
% The evaluated parameters consist of the ratios, the STDs
% They are obtained by averaging on the period T.
% Results are plotted in figure 1
%============
% Here we use the structure FILTERCHARACT and the list of frequencies.
% The only useful processing is the call to the function
%             ESTIMSUTLITE.M
%============  IMPORTANT: list of inputs to the developper =============
% - filter.num and .den
% - allfreqsinfilter_Hz
% - Fs_Hz
% - the signals (signals_centered variable in the following)
% - ref sensor response
%=======================================================================
clear

% the following lines can be changed by the user:
MSCthreshold     = 0.98;
FLAGsavesmall    = 0;
Fs_Hz            = 20;
ihc              = 1;
trimpercent      = 0.7;
nbrandomconcat   = 10;
frequencylist_Hz = logspace(-2,log10(6),30);
%===
% directories
directorysignals    = '../../../AAdataI26calib/';
directoryFB         = '../fullprocess/filtercharacteristics/';
%=======================================================================
%=======================================================================
%====== load the filter bank characteristics
%  the useful structure is FILTERCHARACT
filtercharactfilename = 'filtercharacteristics1.m';
cmdloadfilter         = sprintf('run(''%s%s'')',directoryFB,...
    filtercharactfilename);
eval(cmdloadfilter);
%==================== read data ========================
fileswithdotmat        = dir(sprintf('%ss%i/s%iy*.mat',...
    directorysignals,ihc,ihc));
nbmats                 = length(fileswithdotmat);
signals                = [];
indperm                = randperm(nbmats);
alldates               = cell(nbrandomconcat,1);
selectedlist           = indperm(1:nbrandomconcat);
for indfile = 1:nbrandomconcat
    ifile              = selectedlist(indfile);
    fullfilename_i     = fileswithdotmat(ifile).name;
```

```
    dotlocation        = strfind(fullfilename_i,'.');
    underscorelocation = strfind(fullfilename_i,'_');
    filenameonly       = fullfilename_i(...
        setdiff(1:dotlocation-1,...
        underscorelocation));
    commandload        = sprintf('load %ss%i/%s',...
        directorysignals,ihc,fullfilename_i);
    eval(commandload)
    aux = str2double(fullfilename_i(21:22));
    if aux<9
        straux = ['0' num2str(aux+1)];
    else
        straux = num2str(aux+1);
    end
    date_i             = ...
        sprintf('%s/%s/%s-%s',fullfilename_i(7:10),...
        fullfilename_i(16:17),fullfilename_i(21:22),...
        straux);
    alldates{indfile} = date_i;
    signals           = [signals;signals_centered];
end
%%
sortalldates = sort(alldates);
txt = [];
disp('**********************************************')
display(sprintf('Station %i:',ihc));
for ii=1:nbrandomconcat
   aux = sprintf('%stt %s %s','{',[sortalldates{ii}],'}\\');
   txt = [txt;aux];
end
display(sort(alldates))
disp('************* start process *******************')
%%
%=============================================================
%=============================================================
%============== processing function call =====================
%=============================================================
[Rsup, freqslin, STDmoduleR, STDphaseR, nboverTH] = estimSUTlite ...
    (signals, filtercharact, frequencylist_Hz, ...
    Fs_Hz, MSCthreshold, trimpercent);

%%
%=============================================================
%=============================================================
%=============================================================
figure(1)
semilogx(freqslin, 20*log10(abs(Rsup)),'or')
set(gca,'fontname','times','fontsize',12)
grid on
hold off
title(sprintf('%i days on station %i',2*nbrandomconcat,ihc),...
    'fontname','times','fontsize',14)
xlabel('frequency - Hz')
ylabel('Rsup - dB')
%=============================================================
HorizontalSize = 12;
VerticalSize   = 10;
set(gcf,'units','centimeters');
set(gcf,'paperunits','centimeters');
set(gcf,'PaperType','a3');
set(gcf,'position',[0 5 HorizontalSize VerticalSize]);
set(gcf,'paperposition',[0 0 HorizontalSize VerticalSize]);
set(gcf,'color', [1,1,0.92]);
set(gcf, 'InvertHardCopy', 'off');

printdirectory  = ' ../calibtexte2lite/';
fileprint = sprintf('%sexample2onstation%i.eps',printdirectory,ihc);
figure(1)
fileprintepscmd = sprintf('print -depsc -loose %s',fileprint);
fileeps2pdfcmd  = sprintf('!epstopdf %s',fileprint);
filermcmd       = sprintf('!rm %s',fileprint);


%     eval(fileprintepscmd)
%     eval(fileeps2pdfcmd)
%     eval(filermcmd)
```

2.2 Main function

```
function [Rsup, freqslin, STDmoduleRlin, ...
```

```
    STDphaseRlin_rd, nboverTHlin] = ...
    estimSUTlite ...
    (signals, structfiltercharacteristics, frequencylist_Hz, ...
    Fs_Hz, MSCthreshold, trimpercent)
%=========================================================================
% Synopsis:
% [Rsup, freqslin, STDmoduleRlin, ...
%     STDphaseRlin_rd, nboverTHlin] = ...
%     estimSUTlite ...
%     (signals, structfiltercharacteristics, frequencylist_Hz, ...
%     Fs_Hz, MSCthreshold, trimpercent)
%==============
% Inputs:
%     - signals : T x 2
%     - structfiltercharacteristics (FB structure)
%           see document
%     - frequencylist_Hz: array N x 1 of the selected frequencies
%       in Hz. N can take any value under Fs_Hz/2
%     - Fs_Hz: sampling frequency in Hz
%     - MSCthreshold:
%     - trimpercent: percent of values keptfor averaging
%==============
% Outputs:
%     - Rsup: array N x 1 of the estimated ratios
%     - freqslin: array N x 1 of the selected frequencies
%       in Hz. almost the same as frequencylist_Hz, except if some
%       are outside of the FB bandwidths.
%     - STDmoduleR: array N x 1 of the STD on the module of Rsup
%     - STDphaseR_rd: array N x 1 of the STD on the phase of Rsup
%     - nboverTH: array N x 1 of the number of values over the threshold
%=========================================================================

nbfrequencies          = length(frequencylist_Hz);
Pfilter                = length(structfiltercharacteristics);
frequenciesinfilter_Hz = cell(Pfilter,1);
nbfreqsbyfilter        = NaN(Pfilter,1);

%=== determine the frequencies inside the bank filters
%  in such a way that all frequencies are only in
%  ONE filter band
frequencylist_Hz_ii = frequencylist_Hz;
nbfrequencies_ii    = nbfrequencies;
for idfilter=1:Pfilter
    fqlow_Hz    = structfiltercharacteristics(idfilter).Wlow_Hz;
    fqhigh_Hz   = structfiltercharacteristics(idfilter).Whigh_Hz;
    cp=0;
    for idf=1:nbfrequencies_ii
        if and(frequencylist_Hz_ii(idf)>fqlow_Hz, ...
                frequencylist_Hz_ii(idf)<=fqhigh_Hz)
            cp=cp+1;
            frequenciesinfilter_Hz{idfilter}(cp) = ...
                frequencylist_Hz_ii(idf);
        end
    end
    nbfreqsbyfilter(idfilter) = cp;
    frequencylist_Hz_ii = ...
        setdiff(frequencylist_Hz_ii,frequenciesinfilter_Hz{idfilter});
    nbfrequencies_ii = length(frequencylist_Hz_ii);
end
nbofallfrequencies = sum(nbfreqsbyfilter);
%========== we perform the filter coeffiecient from the structure
%           denoted structfiltercharacteristics
% using the Matlab functions as BUTTER.M
filterbankcoeff = cell(Pfilter,1);
for ifilter = 1:Pfilter
    fname   = structfiltercharacteristics(ifilter).designname;
    forder  = structfiltercharacteristics(ifilter).Norder;
    fqlow   = structfiltercharacteristics(idfilter).Wlow_Hz/Fs_Hz;
    fqhigh  = structfiltercharacteristics(idfilter).Whigh_Hz/Fs_Hz;
    switch fname
        case 'fir1'
            fdesign = sprintf('filnum = %s(%i,[%5.8f,%5.8f]);',...
                fname,forder,2*fqlow,2*fqhigh);
            filden = 1;
        case 'butter'
            fdesign = sprintf('[filnum,filden] = %s(%i,[%5.8f %5.8f]);',...
                fname,forder,2*fqlow,2*fqhigh);
        case 'cheby1'
            fdesign = sprintf('[filnum,filden] = %s(%i,%i,[%5.8f %5.8f]);',...
                fname,forder,0.02,2*fqlow,2*fqhigh);
    end
    eval(fdesign)
    filterbankcoeff{ifilter}.num = filnum;
    filterbankcoeff{ifilter}.den = filden;
```

```
end
%========== we perform the shape window from the structure
%           denoted structfiltercharacteristics
% using the Matlab functions as HANN.M
windshape = cell(Pfilter,1);
for ifilter = 1:Pfilter
    windowshapename = structfiltercharacteristics(ifilter).windowshape;
    SCPperiod_sec   = structfiltercharacteristics(ifilter).SCPperiod_sec;
    ratioDFT2SCP    = structfiltercharacteristics(ifilter).ratioDFT2SCP;
    lengthDFT       = fix(SCPperiod_sec*Fs_Hz/ratioDFT2SCP);
    switch windowshapename
        case 'hann'
            windshape{ifilter} = hann(lengthDFT,'periodic');
            windshape{ifilter} = windshape{ifilter} / ...
                sqrt(sum(windshape{ifilter} .^2));
    end
end
%==== pre-computation of the exponentials used by
% the direct DFTs
EXPV                = cell(Pfilter,1);
for ifilter = 1:Pfilter
    SCPperiod_sec   = structfiltercharacteristics(ifilter).SCPperiod_sec;
    ratioDFT2SCP    = structfiltercharacteristics(ifilter).ratioDFT2SCP;
    lengthDFT       = fix(SCPperiod_sec*Fs_Hz/ratioDFT2SCP);
    DFTindex        = (0:lengthDFT-1)'/Fs_Hz;
    EXPV{ifilter}   = exp(-2j*pi*DFTindex*frequenciesinfilter_Hz{ifilter});
end
%============================================
%============================================
Nsignals   = size(signals,1);
R          = cell(Pfilter,1);
STDmoduleR = cell(Pfilter,1);
STDphaseR  = cell(Pfilter,1);
nboverTH   = cell(Pfilter,1);
for ifilter = 1:Pfilter
    filnum = filterbankcoeff{ifilter}.num;
    filden = filterbankcoeff{ifilter}.den;
    filteredsignals = filter(filnum,filden,signals);

    SCPperiod_sec   = structfiltercharacteristics(ifilter).SCPperiod_sec;
    ratioDFT2SCP    = structfiltercharacteristics(ifilter).ratioDFT2SCP;
    overlapDFT      = structfiltercharacteristics(ifilter).overlapDFT;
    % Computation
    lengthDFT       = fix(SCPperiod_sec*Fs_Hz/ratioDFT2SCP);
    lengthSCP       = fix(SCPperiod_sec*Fs_Hz);
    DFTshift        = fix((1-overlapDFT)*lengthDFT);
    NSCPwindows     = fix(Nsignals/Fs_Hz/SCPperiod_sec);
    sigauxW         = zeros(lengthDFT,2);

    SCP_ifreq11     = zeros(nbfreqsbyfilter(ifilter),NSCPwindows-1);
    SCP_ifreq22     = zeros(nbfreqsbyfilter(ifilter),NSCPwindows-1);
    SCP_ifreq12     = zeros(nbfreqsbyfilter(ifilter),NSCPwindows-1);

    for iwindowSCP  = 1:NSCPwindows-1
        id0   = (iwindowSCP-1)*lengthSCP;
        id1   = 0;
        cpDFT = 0;
        while id1<id0+lengthSCP-lengthDFT
            cpDFT  = cpDFT+1;
            id1    = id0 + (cpDFT-1)*DFTshift+1;
            id2    = id1+lengthDFT-1;
            sigaux = filteredsignals(id1:id2,:);
            sigauxW(:,1) = sigaux(:,1) .* windshape{ifilter};
            sigauxW(:,2) = sigaux(:,2) .* windshape{ifilter};
            for ifreq = 1:nbfreqsbyfilter(ifilter)
                X_ifreq1 = sum(sigauxW(:,1) .* EXPV{ifilter}(:,ifreq));
                X_ifreq2 = sum(sigauxW(:,2) .* EXPV{ifilter}(:,ifreq));
                SCP_ifreq11(ifreq,iwindowSCP) = SCP_ifreq11(ifreq,iwindowSCP) + ...
                    X_ifreq1 .* conj(X_ifreq1);
                SCP_ifreq22(ifreq,iwindowSCP) = SCP_ifreq22(ifreq,iwindowSCP) + ...
                    X_ifreq2 .* conj(X_ifreq2);
                SCP_ifreq12(ifreq,iwindowSCP) = SCP_ifreq12(ifreq,iwindowSCP) + ...
                    (X_ifreq1) .* conj(X_ifreq2);
            end
        end
    end
    tabMSC_ifilter   = (abs(SCP_ifreq12) .^2) ./ ...
        (SCP_ifreq11 .* SCP_ifreq22);
    ind_ifilter_cst = (tabMSC_ifilter>MSCthreshold);
    tabMSC_ifilter_cst = NaN(size(tabMSC_ifilter));
    tabMSC_ifilter_cst(ind_ifilter_cst) = ...
        tabMSC_ifilter(ind_ifilter_cst);

    tabRsup_ifilter  = SCP_ifreq11 ./ conj(SCP_ifreq12);
```

```
    tabRsup_ifilter_cst = ...
        NaN(size(tabRsup_ifilter))+1j*NaN(size(tabRsup_ifilter));
    tabRsup_ifilter_cst(ind_ifilter_cst) = ...
        tabRsup_ifilter(ind_ifilter_cst);

    tabRsup_ifilter_cst_trim = ...
        trimmeancomplex(tabRsup_ifilter_cst,trimpercent);

    SCP_ifreq11_cst   = NaN(size(SCP_ifreq11));
    SCP_ifreq11_cst(ind_ifilter_cst) = SCP_ifreq11(ind_ifilter_cst);
    SCP_ifreq22_cst   = NaN(size(SCP_ifreq22));
    SCP_ifreq22_cst(ind_ifilter_cst) = SCP_ifreq22(ind_ifilter_cst);
    tabR1122_cst      = SCP_ifreq11_cst ./ SCP_ifreq22_cst;

    weightMSCsupeta  = (tabMSC_ifilter_cst .^2) ./  ...
        (1-tabMSC_ifilter_cst) .* tabR1122_cst;

    R_filter         = ...
        nansum(tabRsup_ifilter_cst_trim .* weightMSCsupeta,2) ...
        ./ nansum(weightMSCsupeta,2);

    R{ifilter}       = R_filter;
    nboverTH_ii      = nansum(ind_ifilter_cst,2);
    %===== perform STD on module and phase
    STDmoduleR{ifilter}   = nanstd(abs(tabRsup_ifilter_cst),[],2) ./...
        sqrt(nboverTH_ii);
    STDphaseR{ifilter}    = nanstd(angle(tabRsup_ifilter_cst),[],2) ./...
        sqrt(nboverTH_ii);
    nboverTH{ifilter}     = nboverTH_ii;
end
freqslin      = zeros(nbofallfrequencies,1);
Rsup          = zeros(nbofallfrequencies,1);
id2           = 0;
STDmoduleRlin = zeros(nbofallfrequencies,1);
STDphaseRlin_rd  = zeros(nbofallfrequencies,1);
nboverTHlin   = zeros(nbofallfrequencies,1);
for ifilter=1:Pfilter
    id1 = id2+1;
    id2 = id1+nbfreqsbyfilter(ifilter)-1;
    Rsup(id1:id2)     = R{ifilter};
    freqslin(id1:id2) = frequenciesinfilter_Hz{ifilter}';
    STDmoduleRlin(id1:id2) = STDmoduleR{ifilter};
    STDphaseRlin_rd(id1:id2) = STDphaseR{ifilter};
    nboverTHlin(id1:id2) =  nboverTH{ifilter};
end

%====================================================
%====================================================
%====================================================
function trimmedz = trimmeancomplex(z,trimpercent)

[ra,co] = size(z);
trimmedz = nan(ra,co);

for ira=1:ra
    indout   = quadform(z(ira,:),trimpercent);
    trimmedz(ira,indout==1) = z(ira,indout==1);
end
%====================================================
function indout = quadform(z, apercent)
%====================================================
c     = -2*log(1-apercent);
z     = z(:);
N     = length(z);
meanz = nanmean(z);
zc    = z-ones(N,1)*meanz;
HH    = [real(z) imag(z)];
R     = nancov(HH);
rizc  = [real(zc), imag(zc)];
if or(sum(any(isnan(R)))>0,sum(any(isinf(R)))>0)
    indout = zeros(N,1);
elseif rank(R)==2
    Fm1   = sqrtm(R);
    % valp   = eig(R);
    % area   = sqrt(prod(valp))*c*pi;
    indout = zeros(N,1);
    for ii=1:N
        indout(ii) = rizc(ii,:) * Fm1 *rizc(ii,:)'<c;
    end
else
    indout = zeros(N,1);
end
%=====================================================
```