Documentation for function *tfa_car.m*

This function is a Matlab implementation of the TFA as specified in the White Paper (Journal of Cerebral Blood Flow and Metabolism, 2016). An example of use on the supplied data (*tfa_sample_data.txt*) is given in *tfa_demo.m* and results are reported at the end of this report. Please note that the function uses the built-in Matlab function *nanmean*, which is part of the statistics toolbox of Matlab, and *boxcar,* which is part of the signal processing toolbox[1].

```
function tfa_out=tfa_car(ABP,CBFV,fs,params)
```

<u>Input parameters:</u>

ABP: Arterial blood pressure (assumed to be in mmHg)

CBFV: Cerebral blood flow velocity (assumed to be in cm/s)

fs: Sampling frequency (assumed to be in Hz)

params: A series of parameters that control TFA analysis (window-length, frequency bands …). If this is not provided, default values, corresponding to those recommended in the white paper, will be used. These default values are given below for each parameter.

params.vlf=[0.02,0.07]: Limits of very low frequency band (in Hz). Note that to avoid double counting any frequency, values exactly at the upper limit will be excluded, but values exactly at the lower limit will be included.

params.lf=[0.07,0.2]: Limits of low frequency band (in Hz). Note that to avoid double counting any frequency, values exactly at the upper limit will be excluded, but values exactly at the lower limit will be included.

params.hf=[0.2,0.5]: Limits of high frequency band (in Hz). Note that to avoid double counting any frequency, values exactly at the upper limit will be excluded, but values exactly at the lower limit will be included.

params.detrend=0: linear detrending of data prior to TFA analysis (detrending is carried out as one continuous trend over the whole length of the recording, not segment-by-segment). 0: no, 1: yes. Note that the mean of the whole segment is removed for both ABP and CBFV, regardless of this setting.

params.spectral_smoothing=3: The length, in samples, of the triangular spectral smoothing function. Note that this must be an odd number, to ensure that smoothing is symmetrical around the centre frequency.

params.coherence2_thresholds=[3:15;0.51,0.40,0.34,0.29,0.25,0.22,0.20,0.18,0.17,0.15,0.14,0.13,0.12]': The critical values (alpha=5%, bottom row of the matrix) for coherence for a number of windows (given by the top row of the matrix, here from 3 to 15). These values were obtained by Monte Carlo simulation, using the default parameter settings for the TFA analysis (Hanning window, overlap of 50% and 3-point spectral smoothing was assumed). These values should be recalculated for different settings. Note that if *params.overlap_adjust=1,* the overlap will vary depending on the length of data. With an overlap of 60% (see below), the critical values increase by between 0.04 (for 3 windows) and 0.02 (for 15 windows).

[1] For the next iteration of this software, replacement functions that do not use these toolboxes will be provided.

params.apply_coherence2_threshold=1: Apply the thresholds given above to the TFA estimates. 0: no, 1: yes. All frequencies with magnitude-squared coherence below the threshold value are excluded from averaging when calculating the mean values of gain and phase across the bands. Note that low values of coherence are not excluded in the average of coherence across the bands.

params.remove_negative_phase=1: Remove (ignore) negative values of phase in averaging across bands. 0: no, 1: yes. Negative phase values are removed only for frequencies below the frequency given below, when calculating the average phase in bands.

params.remove_negative_phase_f_cutoff=0.1: The cut-off frequency below-which negative phase values are neglected (if the parameter above is set to 1).

params.normalize_ABP=0: Normalize ABP by dividing by the mean and multiplying by 100, to *express ABP change in %.* Note that mean-values are always removed from ABP prior to analysis.

params.normalize_CBFV=0: Normalize CBFV by dividing by the mean and multiplying by 100, to *express CBFV change in %.* Note that the band-average values of gain are always calculated both with and without normalization of CBFV, in accordance with the recommendations (see *tfa_out.Gain_vlf_not_norm*, *tfa_out.Gain_vlf_norm, etc.)* Note also that mean-values are always removed from CBFV prior to analysis.

params.window_type='hanning': Chose window 'hanning' or 'boxcar'.

params.window_length=102.4: Length of the data-window, in seconds.

params.overlap=59.99: Overlap of the windows, in %. If *params.overlap_adjust=1 (*see below), then this value may be automatically reduced, to ensure that windows cover the full length of data. Here 59.99% rather than 60% was chosen, so that with data corresponding to 5 windows of 100 s at an overlap of 50%, 5 windows are indeed chosen.

params.overlap_adjust=1: Ensure that the full length of data is used (i.e. the last window finishes as near as possible to the end of the recording), by adjusting the overlap up to a maximum value given by *params.overlap.* 0: no, 1: yes.

params.plot=1: Plot graphs. 0: no, 1: yes.

params.plot_f_range=[0,0.5]: Range of frequencies to show in the plots.

params.plot_title='': Title of plots.

<u>Output values:</u>

tfa_out provides the values of all expected measures from TFA analysis

tfa_out.Mean_abp : Mean ABP

tfa_out.Std_abp: Standard deviation of ABP

tfa_out.Mean_cbfv: Mean CBFV

tfa_out.Std_cbfv: Standard deviation of CBFV

tfa_out.overlap: overlap of windows (in %). Note that this may not be the precise value given in params (or the default value set), when overlap is changed slightly to avoid discarding data (see *params.overlap_adjust*)

tfa_out. H: The complex frequency response (transfer function) over the full frequency range

tfa_out.C: The coherence (complex, not magnitude squared)

tfa_out.f: Frequencies corresponding to each of the values of the frequency response (*tfa_out.H)*, coherence (*tfa_out.C*), and spectra (see next lines)

tfa_out.Pxx: Powerspectrum of input signals (ABP)

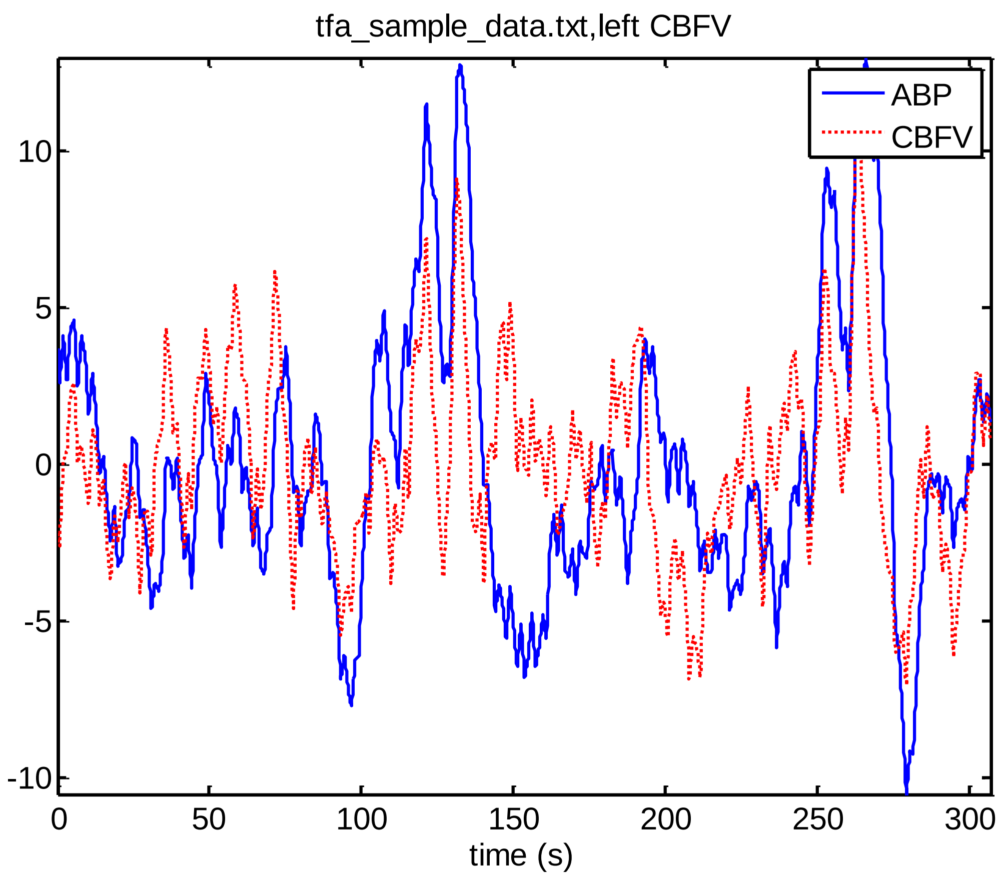
tfa_out.Pyy: Powerspectrum of output signal (CBFV)

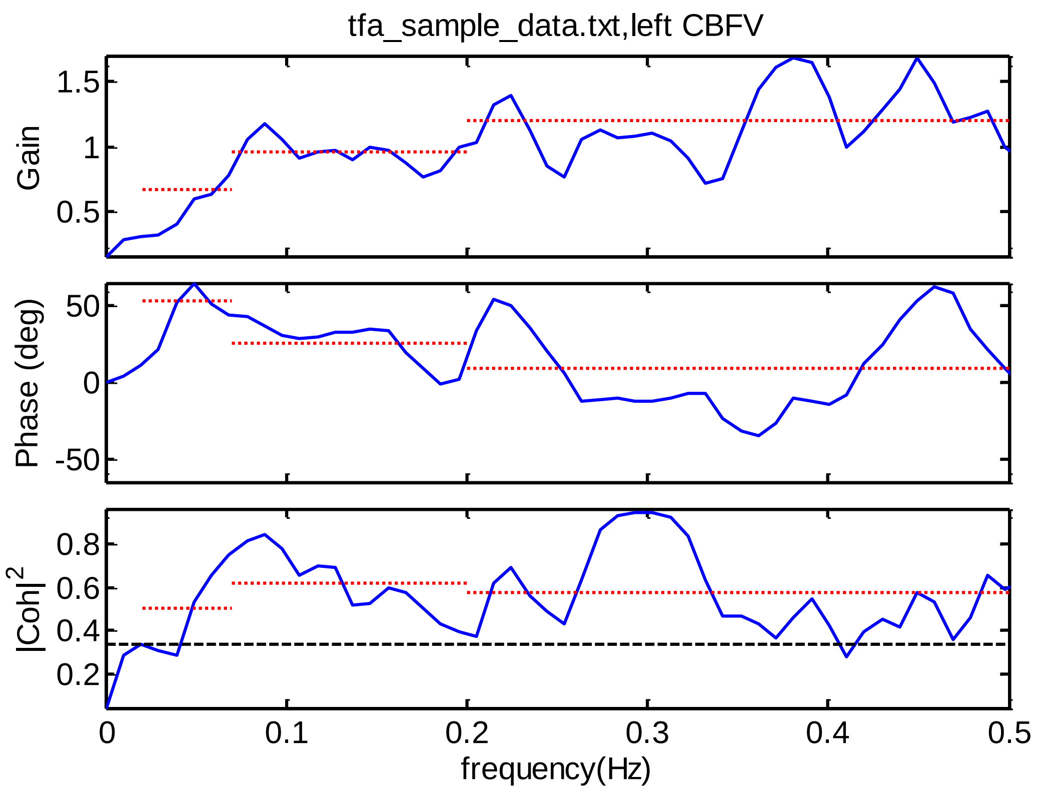
tfa_out.Pxy: Cross-powerspectral density of input and output

tfa_out.No_windows: Number of Windows used
tfa_out.Gain_vlf: average gain in very low frequency band (normalized or not, depending on the *params.normalize … setting)*
tfa_out.Phase_vlf: average phase (in degrees) in very low frequency band
tfa_out.Coh2_vlf: average magnitude-squared coherence in very low frequency band
tfa_out. P_abp_vlf: ABP power in very low frequency band
tfa_out.P_cbfv_vlf: CBFV power in very low frequency band
tfa_out.Gain_lf: average gain in low frequency band (normalized or not, depending on the *params.normalize … setting)*
tfa_out.Phase_lf: average phase (in degrees) in low frequency band
tfa_out.Coh2_lf: average magnitude-squared coherence in low frequency band
tfa_out.P_abp_lf: ABP power in very low frequency band
tfa_out.P_cbfv_lf: CBFV power in very low frequency band
tfa_out.Gain_hf: average gain in high frequency band (normalized or not, depending on the *params.normalize … setting)*
tfa_out.Phase_hf: average phase (in degrees) in high frequency band
tfa_out.Coh2_hf: average magnitude-squared coherence in high frequency band
tfa_out.P_abp_hf: ABP power in high frequency band
tfa_out.P_cbfv_hf: CBFV power in high frequency band
tfa_out.Gain_vlf_not_norm: average gain in very low frequency band (without CBFV normalization, i.e. usually cm.s^{-1}.mmHg^{-1} or cm.s^{-1}/% if *params.normalize_ABP=1.*
tfa_out.Gain_lf_not_norm: average gain in low frequency band (without CBFV normalization, i.e. usually cm.s^{-1}.mmHg^{-1} or cm.s^{-1}/% if *params.normalize_ABP=1.*
tfa_out.Gain_hf_not_norm: average gain in high frequency band (without CBFV normalization, i.e. usually cm.s^{-1}.mmHg^{-1} or cm.s^{-1}/% if *params.normalize_ABP=1.*
tfa_out.Gain_vlf_norm: average gain in very low frequency band (with CBFV normalization, i.e. usually %.mmHg^{-1} or %/% if *params.normalize_ABP=1*
tfa_out.Gain_lf_norm: average gain in low frequency band (with CBFV normalization, i.e. usually %.mmHg^{-1} or %/% if *params.normalize_ABP=1*.
tfa_out.Gain_hf_norm: average gain in high frequency band (with CBFV normalization, i.e. usually %.mmHg^{-1} or %/% if *params.normalize_ABP=1*.

Results from using tfa_car.m on the sample dataset: tfa_sample.txt. The script *tfa_demo.m* shows how to use the function and results are given below, including the output plots.

Left channel:


tfa_sample_data.txt,left CBFV
ABP
CBFV
10
5
0
-5
-10
0
50
100
150
200
250
300
time (s)


tfa_sample_data.txt,left CBFV
Gain
1.5
1
0.5
Phase (deg)
50
0
-50
$|Coh|^2$
0.8
0.6
0.4
0.2
0
0.1
0.2
0.3
0.4
0.5
frequency(Hz)

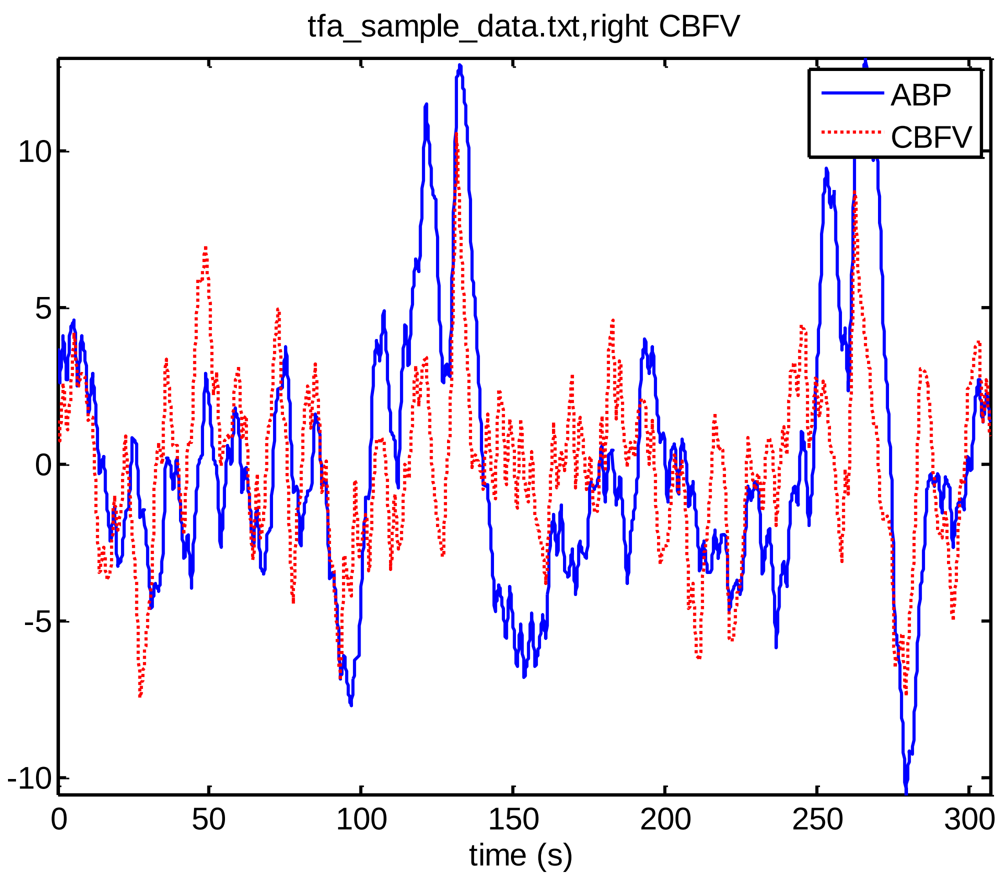
The red dotted lines show the mean values calculated over the respective frequency ranges. The black dashed line shows the critical value of coherence for the 5 windows used here. Note that mean gain and phase is only calculated where coherence is above the dashed line. As no normalization is applied (default setting), the signals are in cm/s and mmHg respectively (but mean has been removed). The gain is thus in cm.s^{-1}.mmHg^{-1}.

The output values from the function (note that these are the results in the White Paper: Transfer function analysis of dynamic cerebral autoregulation: a white paper from the International Cerebral Autoregulation Research Network (CARNet), Journal of Cerebral Blood Flow and Metabolism, 2016):

```
tfa_out =

            Mean_abp: 70.0036
             Std_abp: 4.3092
           Mean_cbfv: 64.9327
            Std_cbfv: 2.9676
             overlap: 50
                   H: [1024x1 double]
                   C: [1024x1 double]
                   f: [1024x1 double]
                 Pxx: [1024x1 double]
                 Pyy: [1024x1 double]
                 Pxy: [1024x1 double]
          No_windows: 5
            Gain_vlf: 0.6760
           Phase_vlf: 52.9658
            Coh2_vlf: 0.5054
           P_abp_vlf: 6.2455
          P_cbfv_vlf: 3.2171
             Gain_lf: 0.9579
            Phase_lf: 25.4391
             Coh2_lf: 0.6171
            P_abp_lf: 1.5583
           P_cbfv_lf: 2.2532
             Gain_hf: 1.1988
            Phase_hf: 9.3763
             Coh2_hf: 0.5730
            P_abp_hf: 0.2131
           P_cbfv_hf: 0.3039
   Gain_vlf_not_norm: 0.6760
    Gain_lf_not_norm: 0.9579
    Gain_hf_not_norm: 1.1988
       Gain_vlf_norm: 1.0410
        Gain_lf_norm: 1.4752
        Gain_hf_norm: 1.8462
```

Right CBFV

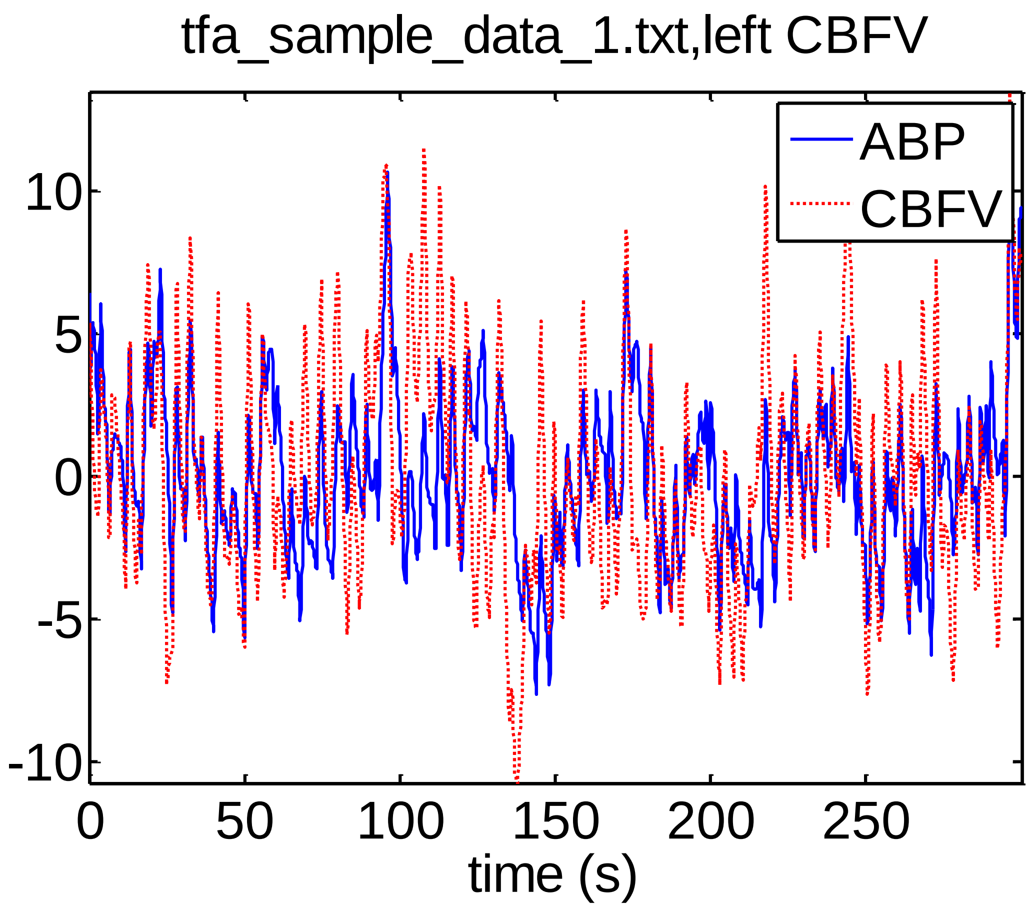


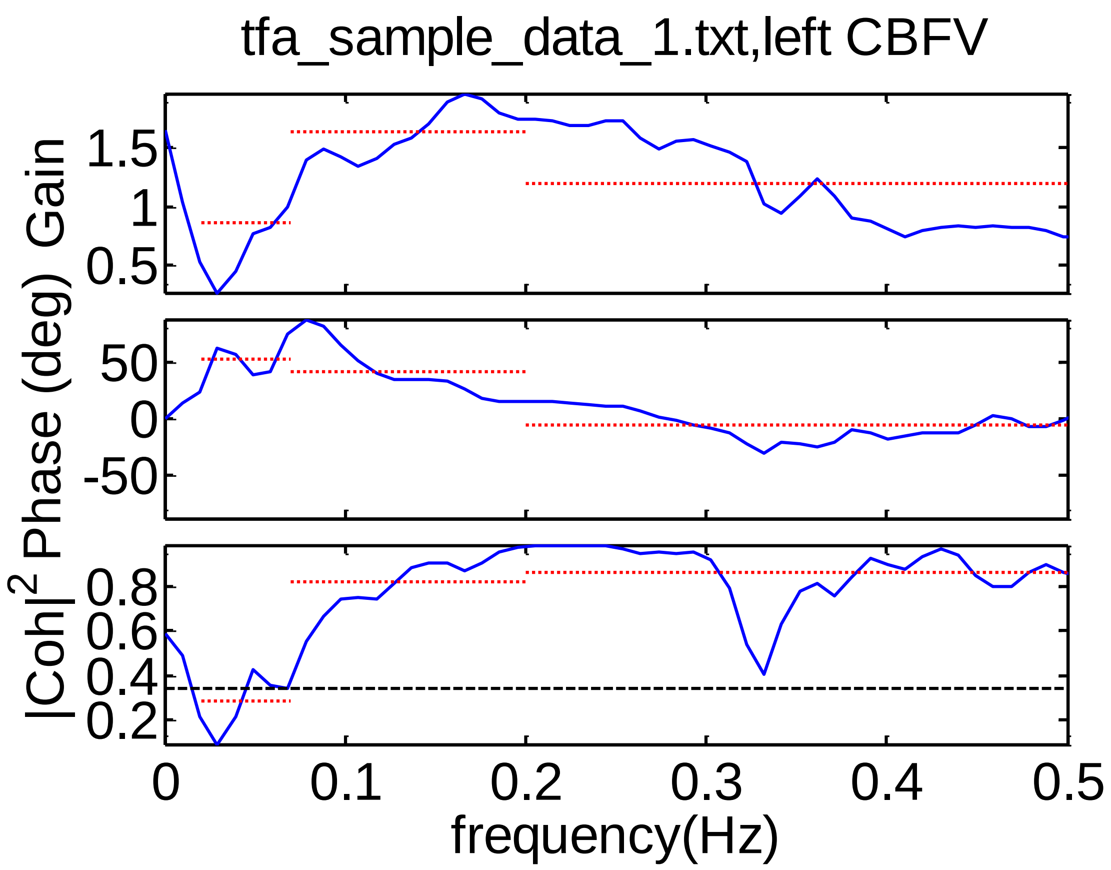
```
tfa_out =

        Mean_abp: 70.0036
         Std_abp: 4.3092
       Mean_cbfv: 61.5967
        Std_cbfv: 2.7737
         overlap: 50
               H: [1024x1 double]
               C: [1024x1 double]
               f: [1024x1 double]
             Pxx: [1024x1 double]
             Pyy: [1024x1 double]
             Pxy: [1024x1 double]
      No_windows: 5
        Gain_vlf: 0.5115
       Phase_vlf: 35.6390
        Coh2_vlf: 0.4941
       P_abp_vlf: 6.2455
      P_cbfv_vlf: 2.6237
         Gain_lf: 0.8792
        Phase_lf: 31.8909
         Coh2_lf: 0.4565
        P_abp_lf: 1.5583
```

P_cbfv_lf:	1.9993
Gain_hf:	1.1033
Phase_hf:	3.0658
Coh2_hf:	0.4774
P_abp_hf:	0.2131
P_cbfv_hf:	0.3292
Gain_vlf_not_norm:	0.5115
Gain_lf_not_norm:	0.8792
Gain_hf_not_norm:	1.1033
Gain_vlf_norm:	0.8304
Gain_lf_norm:	1.4273
Gain_hf_norm:	1.7911

For sample data tfa_sample_data_1.txt – left channel





```
tfa_out = 

       Mean_abp: 84.0305
        Std_abp: 2.8100
      Mean_cbfv: 68.6305
       Std_cbfv: 3.8692
        overlap: 51.7578
              H: [1024x1 double]
              C: [1024x1 double]
              f: [1024x1 double]
            Pxx: [1024x1 double]
            Pyy: [1024x1 double]
            Pxy: [1024x1 double]
     No_windows: 5
       Gain_vlf: 0.8604
      Phase_vlf: 52.4607
       Coh2_vlf: 0.2862
      P_abp_vlf: 2.6053
     P_cbfv_vlf: 3.3860
        Gain_lf: 1.6352
       Phase_lf: 41.9833
        Coh2_lf: 0.8243
```

P_abp_lf: 1.3000

P_cbfv_lf: 4.1607

Gain_hf: 1.1894

Phase_hf: -6.2410

Coh2_hf: 0.8667

P_abp_hf: 1.5022

P_cbfv_hf: 3.7727

Gain_vlf_not_norm: 0.8604

Gain_lf_not_norm: 1.6352

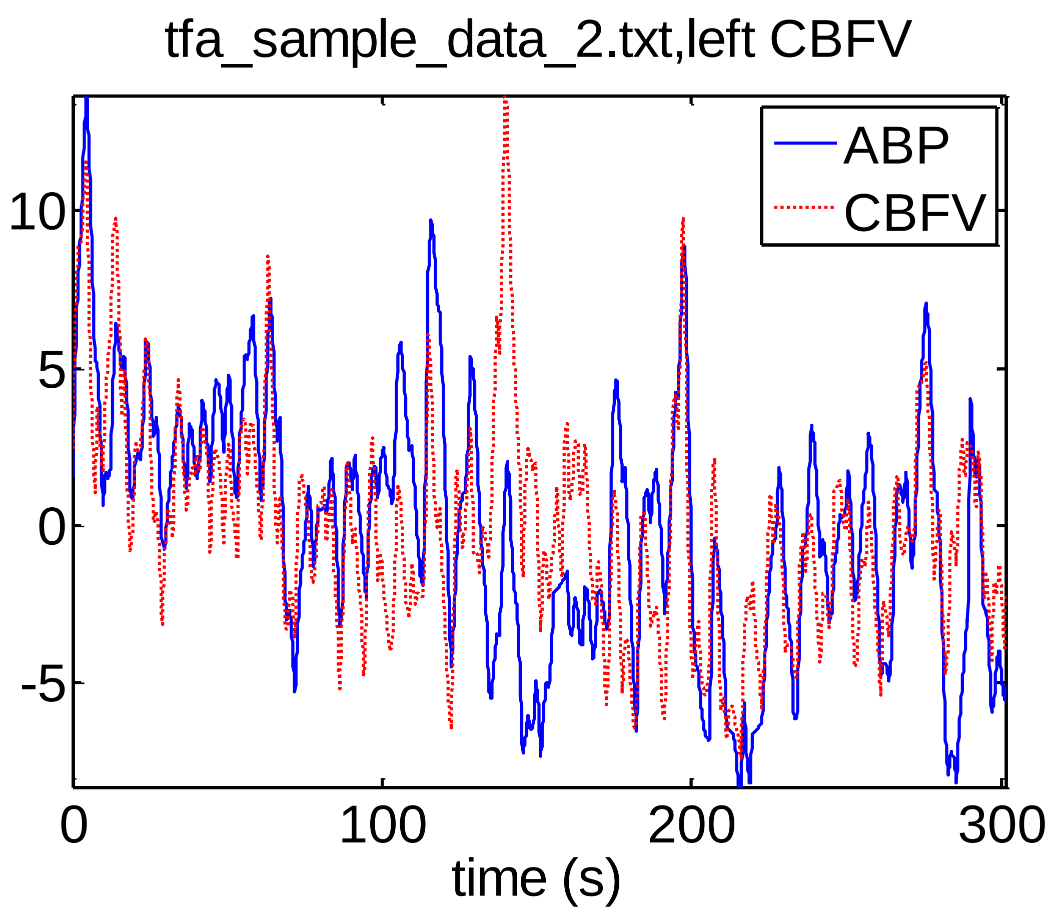
Gain_hf_not_norm: 1.1894

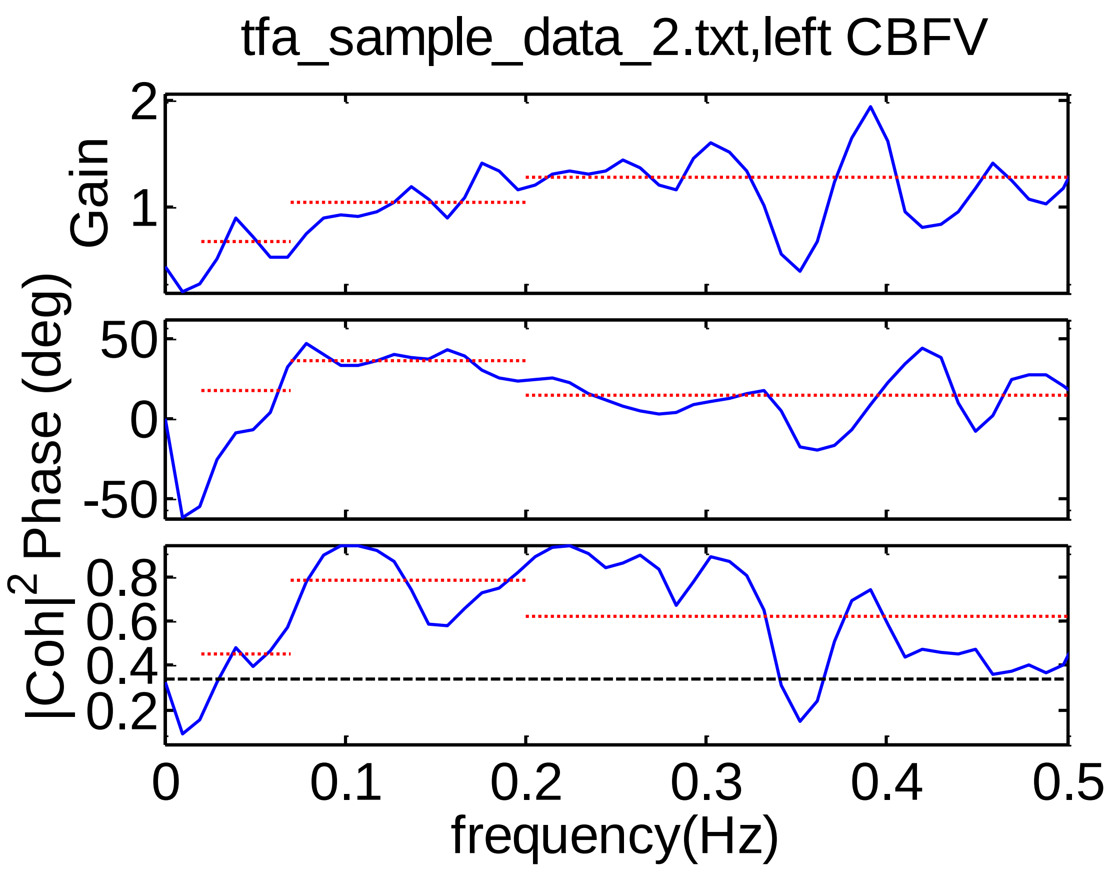
Gain_vlf_norm: 1.2537

Gain_lf_norm: 2.3825

Gain_hf_norm: 1.7330

For tfa_sample_data_2.txt, left channel





```
tfa_out =

     Mean_abp: 77.1532
      Std_abp: 3.9936
    Mean_cbfv: 65.3554
     Std_cbfv: 3.4283
      overlap: 51.4648
            H: [1024x1 double]
            C: [1024x1 double]
            f: [1024x1 double]
          Pxx: [1024x1 double]
          Pyy: [1024x1 double]
          Pxy: [1024x1 double]
   No_windows: 5
     Gain_vlf: 0.6667
    Phase_vlf: 18.1278
     Coh2_vlf: 0.4490
    P_abp_vlf: 2.9248
   P_cbfv_vlf: 2.6534
      Gain_lf: 1.0451
     Phase_lf: 36.0840
      Coh2_lf: 0.7834
     P_abp_lf: 3.5365
```

```
P_cbfv_lf: 3.3673
Gain_hf: 1.2715
Phase_hf: 14.7200
Coh2_hf: 0.6188
P_abp_hf: 0.4585
P_cbfv_hf: 0.9203
Gain_vlf_not_norm: 0.6667
Gain_lf_not_norm: 1.0451
Gain_hf_not_norm: 1.2715
Gain_vlf_norm: 1.0201
Gain_lf_norm: 1.5991
Gain_hf_norm: 1.9455
```

David Simpson, 12/12/2015

Ds@isvr.soton.ac.uk