Supplementary material for Chapter 17

(Draft Version-January 2016)

The time-frequency toolbox TFSAP 7.0 provides two usage options: either to use the Graphical User Interface (GUI), or to use online commands in the user's own code. In addition to technical details and general user's guide, chapter 17 focusses on the GUI option. The objective of this supplementary material (SM) is to assist the readers in using the different features of TFSAP-7.0 toolbox from command window and to use them in their own codes. Examples are provided to illustrate the approach.

A. Background and summary: Sections 17.1 and 17.2

For detailed technical information regarding the functionalities and usage of toolbox TFSAP 7.0, please read the below mentioned first two documents: (1) TFSAP.pdf and (2) Chapter 17. These two documents describe the contents and features of the toolbox. The third document, (3) TFSAP_7.0.ppt, is a demo presentation of the toolbox in accordance with chapter 17.

In addition to the previous signals in "Demo data", more examples of real signals have been added in this supplementary material: whale2 and one speech signal. Also an example of comparison between the Multidirectional Distribution (MDD) and the CKD is provided for a simulated signal. For speech signal analysis, a *(t,f)* representation is generated using the spectrogram with direct implementation.

As mentioned above, the TFSAP 7.0 toolbox provides two key options to the user i.e.: use either the GUI model or online commands in the user's own code.

i. Type "tfsa7" in the Matlab command window followed by "enter" to start the toolbox. The Main Menu of the GUI will then appear (for details read Chapter 17).

Running TFSAP tfsa7 % in command window.

ii. If the user wants to use online commands then the codes below access the same TFSAP 7.0 functions directly in the user's code.

B. Section 17.3

1. Section 17.3.3: Signal generation

The code below shows examples for generating test signals for different types and characteristics.

```
%Example_1:

%Generating a Quadratic FM signal
output= gsig('quad', 0.1, 0.4, 128, 1); % Generating QFM signal
%f1=0.1 Hz, f2=0.4 Hz, number of samples= 128
% sig_type =1 (real signal)
figure; plot(output);

%Example_2:

%Generating a Demo signal(bat1)
output_bat1=load('bat1');
figure; plot(output_bat1);
```

2. Section 17.3.4: Instantaneous Frequency Estimation

The code below shows how to estimate the instantaneous frequency of a given time-domain signal (For details see Chapter 10 of the book).

```
%Generating a Quadratic-FM signal
input_signal= gsig('quad', 0.1, 0.4, 128, 1);

%Applying Phase Difference Algorithm of order 1
Out_IF = pde( input_signal, 1);
figure; plot(Out_IF);
```

3. Section 17.3.5: Quadratic TFD analysis

The code below provides an option to generate different TFDs for any time-domain signal. For analyzing the quadratic TFD of a signal, the user needs to perform the following tasks:

- Generate a time-domain signal as shown in (Signal Generation)
- Select a Quadratic *(t,f)* Analysis Distribution
- Select the parameters.

An example of how to generate TFDs of a given time-domain signal is shown below using first the trivial case of the WVD and then the EMBD.

```
%Example_1:
```

```
input_signal= gsig('quad', 0.1, 0.4, 128, 1); % Generating QFM signal
wl = 127;tr = 1; fftl = 128;
output = quadtfd( input_signal, wl, tr, 'wvd', fftl);% generating WVD.
figure; tfsapl(input_signal,output);

%Example_2:

param_beta=0.9;param=0.1;
output = quadtfd( input_signal, wl, tr, 'emb', param, param_beta, fftl); % generating EMBD
figure; tfsapl(input_signal,output);
```

4. Section 17.3.6: Signal Synthesis

A discrete time-domain signal $z[n]$ can be synthesized from a given TFD $\rho_z[n,k]$ under some conditions (see Sections 2.3.1.1, 3.1.1, and 11.2 in the book).

An example is shown below to illustrate how to synthesize a signal from a given *(t,f)* image.

```
input_signal= gsig('quad', 0.1, 0.4, 128, 1); % Generating QFM signal
tfd = quadtfd( input_signal, 127, 1, 'wvd', 128);% generating WVD.
figure; tfsapl(input_signal,tfd);
%lag_window_length = 127;
%time_res = 1;
%kernel= WVD;
%fft length = 128;
output = synthesize( tfd, 'wvd', 127);  %synthesizing signal
figure; plot(real(output));%plotting time domain signal
%lag_window_length = 127;
%ANALYSIS TYPE ='wvd';
```

C. Section 17.4

1. Section 17.4.1.1: Results for Quadratic FM Signal

The user needs to perform the following tasks:

- Generate a time-domain signal as shown in (Signal Generation)
- Select different Quadratic TF Analysis Distributions
- Plot and compare

Below is the example of comparing different TFDs on a Quadratic FM signal.

```
wl=15; tr=1; fftl=128; swl=15; param_beta=0.9; param=0.1; win win='hamm';
```

```
%Step1
input1= gsig('quad', 0.1, 0.4, 128, 1); % Generating QFM signal


%Step2
tfd_wvd = quadtfd( input1, wl, tr, 'wvd', fftl); %Generating WVD  Distribution
tfd_cw = quadtfd( input1, wl, tr, 'cw', param, fftl);  %Generating Expoential Distribution
tfd_spec = quadtfd( input1, wl, tr, 'specx', swl, win, fftl);   %(Spectrogram)
tfd_B = quadtfd( input1, wl, tr, 'b', param, fftl); %Generating B_Distribution
tfd_mb = quadtfd( input1, wl, tr, 'mb', param, fftl);%Generating MB_Distribution
tfd_emb = quadtfd( input1, wl, tr, 'emb', param, param_beta, fftl); %Generating EMB_Distribution

%Step3
figure; tfsapl(input1,tfd_wvd,'Timeplot','on','Freqplot','on','Grayscale','on','Title','WVD of QFM signal' );
figure; tfsapl(input1,tfd_spec,'Timeplot','on','Freqplot','on','Grayscale','on','Title', 'Spectrogram' );
figure;tfsapl(input1,tfd_B,'Timeplot','on','Freqplot','on','Grayscale','on','Title', 'BD of QFM signal');
figure;tfsapl(input1,tfd_mb,'Timeplot','on','Freqplot','on','Grayscale','on','Title', 'MBD of QFM signal');
figure; tfsapl (input1,tfd_emb,'Timeplot','on','Freqplot','on','Grayscale','on', 'Title', 'EMBD of QFM
signal');
```

2. Section 17.4.1.2: Results for IF Estimation of Quadratic FM Signals

The Matlab code below illustrates the IF estimation of a Quadratic FM signal using three different methods; i.e. Zero-crossing, Phase Differentiation and Peaks of WVD.

The user needs to perform the following tasks:

- Generate a time-domain signal as shown in (Signal Generation)
- Generate IF estimated signal
- Plot and compare

```
wl=127;  % lag window length
zw=64;   % zero crossing
order=2; % order for finite phase difference estimation algorithm
tr=1;    % time resolution
fft1=128; % fft length

%Step1:
input_signal= gsig('quad', 0.1, 0.4, 128, 1); % Generating QFM signal

%Step 2:
% Zero Crossing method
out1 = zce( input_signal,wl );
% Finite Phase Difference method
out2 = pde( input_signal, order);
% Peak of WVD
out3 = wvpe( input_signal, wl, tr, fftl );
```

```
%Step 3:
figure;plot(1:128,out1,1:128,out2,1:128,out3,'linewidth',1.5);xlabel('Samples'); ylabel('Frequency');
legend('Zero Crossing method','Finite Phase Difference method','Peak of WVD')
```

3. Section 17.4.2.1: Results for a Bat Signal

The user needs to perform the following tasks:

- Generate the Bat signal present in the list of demo signals.
- Generate the WVD and MBD of the input bat signal
- Plot the outputs with 'tfsapl' plot and compare the results

Below is the Matlab code

```
wl=63;   % lag window length
tr=3;    % time resolution
fs=142000;  % sampling frequency of bat signal
fftl=128;   % fft length
alpha=0.1;   % value of alpha
l=400; % length of bat signal

%Step1:
input1=load('bat1');

%Step 2:
tfd_wvd = quadtfd(input1, wl, tr, 'wvd', fftl); %Generating WVD
tfd_mb = quadtfd(input1, wl, tr, 'mb', param, fftl);%Generating MBD

%Step 3:
figure;tfsapl(input1,tfd_wvd,'SampleFreq',fs,'Res',tr,'Timeplot','on', 'Freqplot','on','Grayscale','on', 'Title',
'WVD of a Bat signal');
figure;tfsapl(input1,tfd_mb,'SampleFreq',fs,'Res',tr,'Timeplot','on','Freqplot','on','Grayscale','on', 'Title',
'MBD of a Bat signal');
```

4. Section 17.4.2.2: Result for EEG Signal

The user needs to perform the following tasks:

- Generate EEG signal present in the list of demo signals.
- Generate EMBD of the input EEG signal
- Plot the outputs with tfsapl plot option

Below is the Matlab code.

```
wl=127;   % lag window length
tr=1;    % time resolution
fs=50;  % sampling frequency of bat signal
fftl=128;   % fft length
alpha=0.05;   % value of alpha
param_beta=0.5;
l= 1024; %length of eeg signal

%Step1:
input1=load('eeg1');

%Step 2:
tfd_emb= quadtfd(input1,wl,tr,'emb',alpha,param_beta,fftl);
%Generating EMB_Distribution

%Step 3:
figure;tfsapl(input1,tfd_emb,'SampleFreq',fs,'Res',tr,'Timeplot','on', 'Freqplot','on','Grayscale','on', 'Title',
'EMBD of EEG signal');
```

5. Section 17.4.2.3, 4, 5: Results for Whale, HRV, and Bird signals

The user needs to perform the following tasks:

- Generate Whale, HRV, and Bird signals present in the list of demo signals.
- Generate EMBD of Whale signal, MDD of HRV signal, and EMBD of a Bird signal
- Plot the outputs with TFSAPL plot option

Below is the Matlab code for each type of signal.

```
Wl_whale1=127;   % lag window length for wha1e1 signal
Wl_hrv2=63;     % lag window length for hrv2 signal
Wl_bird1=127;       % lag window length for Bird1 signal

fs_whale1=8000;  % sampling frequency of a whale signal
fs_hrv2=2;  % sampling frequency of a hrv2 signal
fs_bird1=12209; %sampling frequency of a bird2 signal

fftl=128;   % fft length
tr=1;       % time resolution
tr2=50 ;      % for Bird signal

alpha_whale1=0.05;   % value of alpha for whale1 signal
alpha_hrv2=0.02;   % value of alpha hrv2 signal
alpha_bird1=0.05;
param_beta_whale1=0.5;
param_beta_bird1=0.5;

l_whale1= 7002; %length of whale1 signal
```

```
l_hrv2= 256; %length of hrv2 signal
l_bird1=9766; % length of a Bird1 signal

%Step1:
input1=load('whale1');
input2=load('hrv1');
input3=load('bird');

%Step 2:
% EMBD of Whale1 signal
tfd_emb1=quadtfd(input1,Wl_whale1,tr,'emb',alpha_whale1,param_beta_whale1, fftl);%Generating
EMB_Distribution

%MBD of HRv2 signal
tfd_mb2= quadtfd(input2,Wl_hrv2,tr,'mb', alpha_hrv2, fftl);%Generating MB_Distribution

% EMBD of a Bird Signal
tfd_emb3=quadtfd(input3,Wl_bird1,tr2,'emb',alpha_bird1, param_beta_bird1, fftl);%Generating
EMB_Distribution

%Step 3:
% plotting Whale1 signal
figure;tfsapl(input1,tfd_emb1,'SampleFreq',
fs_whale1,'Res',tr,'Timeplot','on','Freqplot','on','Grayscale','on', 'Title', 'EMBD of Whale signal');

% Plotting HRV2 signal
figure;tfsapl(input2,tfd_mb2,'SampleFreq',fs_hrv2,'Res',tr,'Timeplot','on','Freqplot','on','Grayscale','on','Tit
le', 'MBD of a HRV signal');

% Plotting Bird signal
figure;tfsapl(input3,tfd_emb3,'SampleFreq',fs_bird1,'Res',tr2,
'Timeplot','on','Freqplot','on','Grayscale','on','Title','EMBD of a Bird signal');
```

6. Section 17.4.3.1.1: Signal reconstruction

The user needs to perform the following tasks:

- Generate a Linear FM signal.
- Generate the WVD of the LFM signal.
- Synthesize the time domain signal using WVD method.
- Generate the WVD of the synthesized signal.

Matlab code:

```
wl=127; tr=1; fft=128; f1=0.1; f2=0.4; l=128;

%Step 1
input= gsig('lin', f1, f2, l, 1); % Generating LFM signal
```

```
%Step 2
tfd_wvd = quadtfd( input, wl, tr, 'wvd', fft); %Generating WVD  Distribution

%Step 3
output = synthesize( tfd_wvd, 'wvd', wl);

%Step 4
tfd_wvds = quadtfd( output, wl, tr, 'wvd', fft); %Generating WVD
Distribution
%Plots
figure;plot(real(input));
figure;tfsapl(input,tfd_wvd);
figure;plot(real(output));
figure;tfsapl(output,tfd_wvds);
```

7. Section 17.4.3.1.2: Denoising using the WVD

The user needs to perform the following tasks:

- Generate a Linear FM noisy signal.
- Generate the WVD of the LFM signal.
- Using a mask to reduce the noise.
- Synthesize the time domain signal using WVD method.

```
input_signal= gsig('lin', 0.1, 0.4, 256, 1); % Generating LFM signal
g=awgn(input_signal,20);  %Adding Gaussian noise of 20 dB's to the LFM signal
tfd = quadtfd( g, 127, 1, 'wvd',256);
figure;tfsapl(g,tfd);

%Step 1:
if1 = wvpe( g, 127, 1, 256 ); % estimating IF

%Step 2:
binaryImage=zeros(size(tfd));
f = linspace(0,0.5,size(tfd,2));
for k=1:length(f)
[val,pos(k)]=min(abs(f-if1(k)));
end

for k=1:length(f)
binaryImage(pos(k),k)=1;
end

 %Steps 3, 4 and 5:
filt=zeros(size(tfd));
for k=1:length(f)
filt(k,:) = conv(binaryImage(k,:),gausswin(256,16),'same');
end
```

```
DenoisedTfd = filt.*tfd;
output = synthesize(DenoisedTfd, 'wvd', 127);
figure;plot(0:255,output,0:255,input_signal)
figure;tfsapl(output,DenoisedTfd);
```

8. Section 17.4.3.2: denoising using the STFT and Spectrogram

The user needs to perform the following tasks:

- Generate a signal.
- Generate the Short time Fourier Transform of the signal.
- Using a mask to reduce the noise.
- Synthesize the time domain signal using STFT reconstruction methods.

```
load speechSignal;
load G;
noisySignal=awgn(speechSignal,40);
tfd = stft(noisySignal,fs,512,'hann',384,512,1);
figure;imagesc(20*log(abs(tfd)));axis xy%plot of the spectrogram in dB
denoisedtfd = tfd.*mask;
figure;imagesc(20*log(abs(denoisedtfd)));axis xy %plot of the  denoised
spectrogram in dB
```

The user can find attached a file 'G.mat' to run this script. Please see Section 11.6 where different methods to compute the gain function (mask in the code above) are presented and see Section 11.2 where an efficient method to reconstruct the signal from the denoised STFT is described.

9. Section 17.4.3.3: TF Synthesis of a Signal from an Arbitrary *(t,f)* Image

Below is an example of signal synthesis from an arbitrary image describing some desired *(t,f)* specifications. The procedure involves the following steps:

- Form the desired *(t,f)* image. Import the image in the MATLAB environment, i.e.,
- Transform the image in gray level (note that this step is necessary only for RGB images):
- Transpose the image:
- Choose the modified Spectrogram synthesis method
- Generate Spectrogram of the synthesized signal

```
wl=127; % lag window length
tol=0.25;  % tolerance factor
tr=3;  %time resolution
swl=127;   % Smoothing Window Length
fftl=512;
tfd = double(imread('signal1.jpg','jpg'));
%tfd = rgb2gray(tfd); only for RGB images.
```

```
tfd = tfd';
output = synthesize( tfd, 'mspec', wl, 'hamm', tol );
tfd_syn = quadtfd( output, wl, tr, 'specx', swl, 'hamm', fftl);
figure;tfsapl(output,tfd);
figure;tfsapl(output,tfd_syn);
```

D. Section 17.4.5 and new examples of functionalities in TFSAP-7.0

1. Generating the CKD and MDD of a simulated signal

The user needs to perform the following tasks:

- Generate a simulated signal composed with three monocomponent signals.
- Take the CKD and MDD of the time domain signal.
- Plot the result

The code below generates the MDD and CKD of a simulated signal.

```
s1= gsig('lin', 0.05, 0.3, 256, 1);
s2= gsig('lin', 0.1, 0.35, 256, 1);
s3= gsig('lin', 0.4,0.36, 256, 1);
sig = s1+s2+s3;
C=[0.1 0.1]; D=[0.1 0.1]; E=[0.23 0.5]; theta=[96 64]; tr=2;
tfdMDD= mdd(sig, C, D, E,theta, tr);
figure;tfsapl(sig,tfdMDD);
fftl= 256; C= 0.1; D= 0.2; E= 0.1;
tfd_CKD = cmpt( sig, 'ckd', C, D, E, fftl);
figure;tfsapl(sig,tfd_CKD);
```

2. Generating the spectrogram of a speech signal

The code below shows an analysis of a speech signal using the short time Fourier transform and how we can use toolbox functions with Matlab code.

The procedure involves the following steps:

- Generate a Speech signal from the Demo data.
- Take Spectrogram of the time domain signal.
- Plot the result

```
input1 = load('speechSignal');

Fs = 8000;
fftl =2048;
param_overlap = 1000;
win = 'hann';
wl= 1024;
```

```
[output,t,f] = stft (input1, Fs, wl, win, param_overlap, fftl,1);
output = 10*log10(abs(output.^2));
figure; imagesc(t,f,output); axis xy ; colorbar; ylabel('Frequency(Hz)'); xlabel('Time(sec)')
```

3. New TFSAP-7.0 functions

The following are functions that were not available in the previous version 6.4 and earlier versions.

Function	Description
wd	Wigner Distribution
stft	Short Time Fourier Transform
specSM	Enhanced spectrogram using S-method
mdd	Multi-Directional distribution

E. Future updates

This manuscript will be further reviewed and expanded as part of the task of completing the Supplementary Material for the Elsevier Academic Press TFSAP book (2nd Ed.). The updated file will be available on the book companion website. The release date of the next update is dependent on the feedback received for this version. The current estimate is that it is expected around March/April 2016.