Project 3 - Signal Denoising

ECE505, Spring 2020
EECS, University of Tennessee
(Due 04/07)

1 Objective

The objective of this project is to apply various types of filters in different design structures to solve a problem in smart grid, where voltage signals collected from the wall outlet need to be enhanced (or cleaned up) in order to yield a better (more smooth) estimate of the frequency, which is essential to the subsequent stability analysis of the grid.

2 Background: Frequency Estimation in Power Grid

2.1 FDR and FNET

To maintain power system operation in a balanced and stable condition, the frequency deviation and the rate of frequency change information are highly desired. How to obtain frequency information more accurately and efficiently has been an active research topic for decades. During the 1980s, work first began on closely synchronized measurements that would allow direct measurement of voltage phase angle at the *transmission level*. As a result, PMUs (Phasor Measurement Units) were installed in substations and measure phasors at high voltage levels. At present, there are 105 PMUs installed in the Eastern Interconnection and 56 in the Western Interconnection. With DOE support, about 800 such units will be installed in the US by 2015. This slow development is in part due to the high installation cost and in part due to the time delays in getting approval, which have limited their applications for wide area control and stability analysis of power system.

As a member of the PMU family, the Frequency Disturbance Recorder (FDR) was developed at Virginia Tech in 2003. The effort was led by Dr. Yilu Liu. At the distribution level, FDR collects instantaneous phasor and frequency taken at any low voltage level such as a 120 V wall outlet and transmit measured frequency data remotely via the Ethernet. The uniqueness of FDR is that it can *measure essential transmission level information at distribution level* using low-cost sensors. Based on these low-cost FDRs, a US-wide Frequency Network (FNET) has been implemented and some power system monitoring applications are being developed by taking full advantage of the FDRs. Thus far there are about 110 FDRs installed in the United States and another 20 installed worldwide. Currently, the system is operated by the University of Tennessee and Virginia Tech as a joint effort. Based on the success of the FNET project, a full-scale monitoring infrastructure, called GridEye, will be built at DOE's Oak Ridge National Laboratory in the near future.

The fact that the FDRs are installed at the 120V distribution level reduces their manufacturing and installation costs significantly. For example, at the Tennessee Valley Authority (TVA), the installed cost of one PMU was more than \$80,000. Installed cost of one frequency and angle sensor using the FDR with GPS and network capability can be made below \$1,000 each. *With the sensing hardware in place, the challenge associated with the distribution-level approach versus the transmission-level approach lies in signal processing*.

2.2 Frequency Estimation

Frequency disturbances is one of the most important indicators that reflects the state of an electrical power system. Making accurate frequency measurement from low voltage distribution systems through the wide deployment of FDRs has been the major innovation for the FNET. Currently, the frequency calculation algorithm based on the phasor angles (FPA) of the measured voltage signal in the FDR, has achieved high accuracy. However, this algorithm is very sensitive to noise which is inevitable in the signal sampling process. As a result, the frequency estimation accuracy will be degraded if the measured signal is not clean enough [PESGM:13].

2.3 Characteristics of the Voltage Signal

The raw voltage signal collected by FDR from the power grid is very noisy. From the frequency spectrum, we observe that harmonics around the 60Hz signal and Gaussian white noise are dominant and non-negligible. Besides the peak frequency around 60 Hz, there also exist impulses around 120Hz and 180Hz along with the irregular fluctuations and ripples in the background,

The mathematical expression for the signal in frequency domain can be denoted as:

$$x_f[n] = s_f[n] + \sum_i h_i[n] + n_f[n] \tag{1}$$

where $s_f[n]$ is the sinusoidal signal with frequency around 60Hz, $h_i[n]$ is the harmonics impulses, and $n_f[n]$ is the Gaussian white noise with zero mean and variance σ_n^2.

2.4 Frequency estimation by Phase Angle (FPA)

Given a voltage signal, denoted as S, a moving window ζ with fixed length is applied on S. By moving the window ζ with a fixed step ω, whose value is less than the length of the window, we can crop a sequence of short period signals, denoted as $s_1, s_2, s_3, \cdots$. With each signal fragment s_i, the FPA algorithm is utilized to estimate the value of frequency at the current time period i. The FPA algorithm is a recursive algorithm that consists of two major steps: first, a rough frequency estimation in the signal fragment s_i is computed using a second order least square approximation on the phasor angles; then, a resampling based on the rough frequency estimation is carried out, followed by another second order least square approximation to obtain the final estimation.

2.5 Data and Code

The following dataset and code can be downloaded from Canvas. Please do not distribute outside of the class.

- Two benchmark voltage signals: signal $S1$ (`SimuCase1.mat`) is a sinusoidal wave of stable frequency as 59.98Hz with white noise added, the other signal $S2$ (`SimuCase2.mat`) is a sinusoidal wave of unstable frequency varying around 58.98hz with also white noise added. The SNR of both noisy signals $S1$ and $S2$ are 20dB. In all the tests using simulated data, we set the length of each signal fragment s_i as about 8 cycles and set the window's moving step ω as 1 for FPA. The code that generates the simulated data is provided for your reference (`generatenoisysignal.m`).
- Real data collected from FDR (`RealData.mat`).
- The frequency estimate code from Dr. Liu's group (`movWinDft.m`)
- `mylms.m` and `myrls.m` for your reference. You can directly use the adaptive filtering toolbox provided by MATLAB [MATLAB:adaptfilt]

3 Tasks

You need to design ten different filters for the denoising operation in both options, including five IIR filter designs (Butterworth with impulse response, Butterworth with bilinear transformation, Chebyshev I, Chebyshev II, and elliptic with bilinear transformation), three FIR filter designs (Rectangular window, Kaiser window, and optimal filter using the Parks-McClellan algorithm), and the adaptive filter implementation with FIR. Pick the best performing IIR lowpass filter and convert it to a bandpass filter. Pick the best performaning filter and try different design structures. Submit the matlab code.

- Task 1: Write a matlab function and use the signal processing toolbox to implement the above filters.
- Task 2: Compare the orders of the filters
- Task 3: Show the pole-zero plots and frequency response of the filters

To make it less complicated for you, following is a sample code that implements an upsampler-by-four system (Problem 7.10 in our textbook). The ideal filter for 1:4 interpolation is an ideal lowpass filter with gain of 4 and cutoff frequency $\pi/4$. To approximate this filter, the specifications of the digital filter are given as follows:

```
    passband edge frequency $\omega_p = 0.22\pi$
    stopband edge frequency $\omega_s = 0.29\pi$
    maximum passband gain = 0 dB
    minimum passband gain = -1 dB
    maximum stopband gain = -40 dB
```

IIR filter design - Impulse invariance - Butterworth

```
T = 1;              % Mapping the specs using impulse invariance
Wp = wp/T; Ws = ws/T;

[N, Wn] = buttord(Wp, Ws, rp, rs, 's'); % analog Butterworth filter
[B, A] = butter(N, Wn, 's');            % finding the coefficient
                                        % of the analog filter
[Bz, Az] = impinvar(B, A);              % map to digital filter

[Hz, Wz] = freqz(Bz, Az);               % the digital filter
```

IIR filter design - Bilinear transformation - Butterworth

```
Wp = 2*tan(wp/2); Ws = 2*tan(ws/2);
[N, Wn] = buttord(Wp, Ws, rp, rs, 's');  % analog Butterworth filter
[B, A] = butter(N, Wn, 's');
[Bz, Az] = bilinear(B, A, 1);
[Hz, Wz] = freqz(Bz, Az);     % the digital filter
```

FIR filter design - Window method - Rectangular. Note that the rectangular window cannot be used for FIR filter design since the peak approximation error is higher than what's required. So we choose the Hamming window. Pay attention to how the approximation error is calculated.

```
wp = 0.22*pi;
ws = 0.29*pi;
rp = 1;              % minimum passband gain -20*log10(1-delta_p)
rs = 40;             % maximum stopband gain -20*log10(delta_s)

delta_s = 20*log10(10^(-rs/20));     % maximum approximation error
                                     % in stopband in dB
delta_p = 20*log10(1-10^(-rp/20));   % maximum approximation error
                                     % in passband in dB
delta_peak = min(delta_s, delta_p); % peak approximation error
                                     % should be less than this.
dw = ws - wp;             % transition bandwidth
Wn = (ws+wp)/2;
N = ceil(8*pi/dw);        % according to Table 7.2, column 3
B = fir1(N, Wn/pi, hamming(N+1));
[Hz, Wz] = freqz(B, 1);
```

FIR filter design - Window method - Kaiser

```
wp = 0.22*pi;
ws = 0.29*pi;
rp = 1;              % minimum passband gain -20*log10(1-delta_p)
rs = 40;             % maximum stopband gain -20*log10(delta_s)

delta_s = 10^(-rs/20);     % maximum approximation error in stopband
delta_p = 1-10^(-rp/20);   % maximum approximation error in passband

[N, Wn, beta, ftype] = kaiserord([wp/pi ws/pi], [1 0], [delta_p delta_s]);
                          % since the first argument in kaiserord()
                          % needs to be in Hz, and omega =
                          % 2*pi*f/f_s, also that the default
                          % sampling rate is f_s=2, we have
                          % f=omega*f_s/(2*pi) = omega/pi

% alternatively, you can manually calculate the order and beta
% based on Eq. 7.75 and 7.76
```

```
delta = min(delta_s, delta_p); % take the minimum of these two as
                               % for windows method, the
                               % approximation error in passband
                               % and stopband need to be the same,
                               % we then take the minimum
A = -20*log10(delta);
N = ceil((A-8)/(2.285*dw));
if A>50
  beta = 0.1102*(A-8.7);
elseif A < 21
  beta = 0;
else
  beta = 0.5842*(A-21)^0.4 + 0.07886*(A-21);
end

B = fir1(N, Wn, ftype, kaiser(N+1, beta), 'noscale');
[Hz, Wz] = freqz(B, 1);
```

FIR filter design - Optimal method - Parks-McClellan

```
wp = 0.22*pi;
ws = 0.29*pi;
rp = 1;               % minimum passband gain -20*log10(1-delta_p)
rs = 40;              % maximum stopband gain -20*log10(delta_s)

delta_s = 10^(-rs/20);     % maximum approximation error in stopband
delta_p = 1-10^(-rp/20);   % maximum approximation error in passband

[N, Fo, Ao, W] = firpmord([wp/pi ws/pi], [1 0], [delta_p delta_s]);

% alternatively, you can manually calculate the order based on
% Eq. 7.117. However, my calculation shows N=33 while the order
% calculated from firpmord is 35. Not sure why.
dw = ws - wp;
%N = ceil((-10*log10(delta_p*delta_s)-13)/(2.324*dw));

B = firpm(N, Fo, Ao, W);
[Hz, Wz] = freqz(B, 1);
```

4 Reference

1.

$$PESGM : 13$$

Wei Wang, Liu Liu, Li He, Lingwei Zhan, Hairong Qi, Yilu Liu, "Highly accurate frequency estimation for FNET," IEEE Power & Energy Society General Meeting (PESGM), Vancouver, Canada, July 21-25, 2013.

2.

$$MATLAB : adaptfilt$$

Adaptive Filters: `http://www.mathworks.com/help/dsp/adaptive-filters.html`