Coarse Frame Search Using the Frequency Domain Representation

Author: Michel Barbeau, Carleton University

Draft version: January 29, 2019

Intro

This MATLAB live script describes the coarse search for frames. This script reads a file with the ".c2" extension. The c2-format is a baseband representation of channel data, with the in-phase and quadrature. In the representation, the script finds frequencies that potentially contain frames. Each frame comprises 162 channel symbols, which encode 50 information bits. Convolutional Forward Error Correction (FEC) is used, with a constraint of 32 and a rate of $1/2$. Convolutional encoding of the information bits yields 162 bits. They are interleaved with 162 synchronization bits s_i $(0 = 1, \ldots, 161)$:

```
clear;
% synchronization bits
s = [1 1 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 1 0 0 1 0 1 1 1 1 0 0 0 0 0 0 0 1 0 0 1 0 1 .
      0 0 0 0 0 0 1 0 1 1 0 0 1 1 0 1 0 0 0 1 1 0 1 0 0 0 0 1 1 0 1 0 1 0 1 0 1 0 0 1 .
      0 0 1 0 1 1 0 0 0 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 1 0 0 1 0 0 1 1 1 0 1 1 0 0 1 1 .
      0 1 0 0 0 1 1 1 0 0 0 0 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 1 1 0 1 0 1 1 0 0 0 1 1 0 .
      0 0];
```

Every data bit is paired with a synchronization bit. Each pair makes a channel symbol. Modulation is four-tone Multiple Frequency-Shift Keying (MFSK) at 1.46 baud ($375/256$). The complex modulation envelope frequencies are -2.2, -0.7, 0.7 and 2.2 Hertz, corresponding to channel symbols 0, 1, 2 and 3. The signal bandwidth is therefore 4.4 Hertz. The transmission time of a frame is 111 seconds.

Reading a file with the .c2 extension

```
% name of ".c2" file
FileName = '150426_0918.c2';
fileID = fopen(FileName,'r');
```

The following three parameters (file name, type and frequency) are read, but not interpreted.

```
% read file name (14 characters)
fname=fread(fileID,[1 14],'char');
% read WSPR type (integer) 2="2 minute mode"
type=fread(fileID,[1 1],'int');
% read frequency (double)
dfreq=fread(fileID,[1 1],'double');
```

The content of the file represents 120 seconds of signal samples that have been buffered, termed the *two minute mode*. The sampling rate (f_s) is 375 samples per second (sps). Hence there are $n = 375 \cdot 120 = 45,000$ complex samples (in-phase and quadrature) read. Let $x_0, x_1, \ldots, x_{n-1}$ represent the discrete complex samples of two-minute channel data.

```
n=45000; % number of sambles (120 seconds of channel data)
```

```
fs=375; % sampling frequency
% read data (float)
buffer=fread(fileID,[2 n],'float');
% close the ".c2" file
fclose(fileID);
```

The in-phase and quadrature signals are respectively assigned to the MATLAB variables I and Q.

```
% in-phase
I=buffer(1,:);
% quadrature
Q=buffer(2,:);
```

We have that every sample x_i is equal to the complex number $\mathrm{I}[i+1]+jQ[i+1]$. Each channel symbol is represented by 256 samples. A frame consists of 256 samples per symbol times 162 channel symbols, i.e., 41,472 samples.

Ploting the data in the frequency domain

This part builds and plots the frequency domain representation. The representation decomposes the whole 120 second period into components of different frequencies. A stepped-frequency domain representation is constructed. Steps are counted in half symbol periods. At each step, a sub-window corresponding to the duration of two symbols is decomposed into component frequencies. Each symbol being represented by 256 samples, a half symbol consists of 128 samples. Therefore, the number of FFTs is

$$nffts = \left\lfloor \frac{n \text{ samples}}{128 \text{ samples per half symbol}} \right\rfloor - 3 = 348 \text{ FFTs}.$$

The subtrahend 3 is present because calculations of windowed FFTs stop before the third to last half-sample.

```
% number of samples per half symbol
hs = 128;
% number of FFTs
nffts=floor(n/hs)-3;
```

The size of each FFT (variable N) is $N = 2 \text{ symbols} \cdot 256 \text{ samples/symbol} = 512 \text{ bins}$.

```
% FFT size
N=512; % bins
```

Used together with the FFT, the windowing function is

$$w(t) = \sin\left(\frac{\pi}{512} \cdot t\right), \text{ t=0,...,N-1}.$$

It is a half-sine wave cycle, in N steps.

```
% windowing function
w=sin([0:(N-1)].*pi/N);
```

Every FFT coefficient is defined as

$$X_{m,k} = \sum_{t=0}^{N-1} x_{128m+t} \cdot w(t) \cdot e^{-j2\pi kt/N}.$$

The FFT window index m is in the range $0, \ldots, \text{nffts} - 1$. It represents the relative amplitude, and phase, of frequency

$$\frac{k \cdot f_s \text{ sps}}{N \text{ samples}} \text{ Hertz}.$$

From frequency-bin-to-frequency-bin, there is an offset Δf of $375/512 = 0.73$ Hertz. At 375 sps and according to the Nyquist criterion, the frequency range of each FFT is ± 187 Hertz. Because there are 512 bins, the coefficient index k is in the range -256,...-1,0,...255. In the frequency domain, every two-minute time interval is represented by the following matrix:

$$X = \begin{pmatrix} X_{0,-256} & \ldots & X_{0,0} & \ldots & X_{0,255} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ X_{nffts-1,-256} & \ldots & X_{nffts-1,0} & \ldots & X_{nffts-1,255} \end{pmatrix}$$

In MATLAB:

```
% coefficient array
X=[];
for k=1:nffts
    % FFT over a two-symbol interval
    k=(k-1)*hs+1;
    % windowing function
    Iw=I(k:k+N-1).*w;
    Qw=Q(k:k+N-1).*w;
    Y=fft(complex(Iw,Qw),N)/N;
    Z=[Y(1,N/2+1:N),Y(1,1:N/2)];
    X=[X ; Z];
end;
```

Note that the array items represented by variable Y(1,N/2+1:N) correspond to the negative frequencies

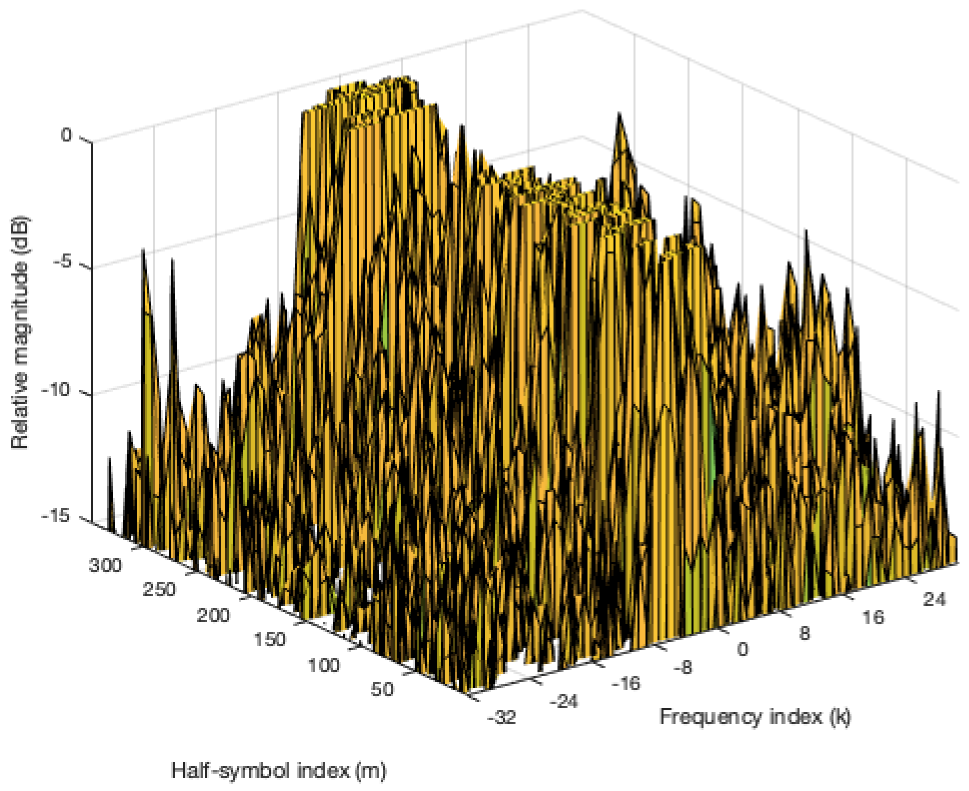
$$X_{m,-N/2}, \ldots, X_{m,-1}$$

while the array items represented by variable Y(1,1:N/2) correspond to the positive frequencies

$$X_{m,0} \ldots, X_{m,N/2-1}.$$

They are re-ordered into MATLAB variable Z from negative to positive frequencies. The frequency domain representation is used for a coarse signal search. The procedure looks for candidates in the frequency domain, i.e., columns in the matrix where there is a local Signal-to-Noise Ratio (SNR) peak. We plot on a dB scale, the magnitude of the signal for frequency indices $-32, \ldots, -1, 0, \ldots 31$, i.e., ± 23 Hertz.

```
figure;
[m,k]=meshgrid(1:N,1:nffts);
surfc(m,k,db(abs(X(1:nffts,1:N))));
grid on;
xlabel('Frequency index (k)');
xlim([225 288]);
set(gca,'XTick',225:8:288)
set(gca,'XTickLabel',-32:8:64)
zlim([-15 0]);
ylabel('Half-symbol index (m)');
zlabel('Relative magnitude (dB)');
```



The vertical axis (Z) provides the relative magnitude of the signal (in dB) as a function of time (counted in half-symbol), axis X, and frequency (in units of 0.73 Hertz). The indices of the negative frequencies are N/2+1 to N. The indices of the positive frequencies are 1 to N/2. This plots reveals the presence of a data signal at zero Hertz.

Finding frequencies of candidate signals

We first store into variable W the complex magnitude at each frequency bin, a calculation corresponding to the following equation $W_{m,k} = |X_{m,k}|$. We have that the variable $W_{m,k}$ denote the magnitude of the signal at indices m and k, in linear form.

```
W=abs(X);
```

Next, we compute average magnitude of each bin:

```
M=mean(W,1);
```

Variable M represents the average signal magnitude as a function of frequency. The magnitude is smoothed taking, for each frequency, the sum of the magnitudes of signals at three adjacent frequencies, resulting into $N-(2\cdot 3)=506$ bins. This results into the relative magnitude of the signal over a bandwidth of $6\cdot 0.73=4.38$ Hertz:

```
% smoothing
SM=zeros(1,506);
for k=1:length(SM)
    for j=-3:1:3
        SM(k)=SM(k)+M(k+3+j);
    end;
end;
```

The smoothed average magnitude levels are sorted in ascending order:

```
SSM=sort(SM);
```

The following heuristic is used. The 150-th component (30% of 506) is chosen as the noise reference level:

```
noise_level=SSM(150);
```

A minimum reference Signal-to-Noise Ratio (SNR) is chosens, -7 dB, in linear form it is

$$min_{SNR}=10^{-0.7}.$$

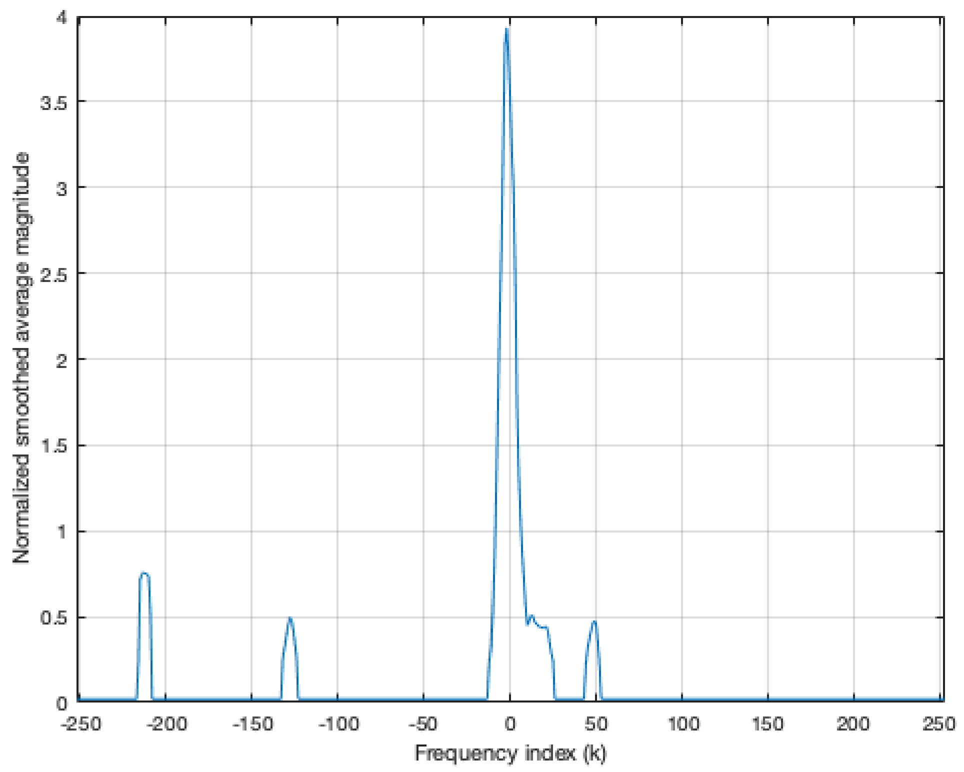
```
min_snr = 10.0^(-7.0/10.0);
```

The smoothed average magnitude levels are normalized with respect to the reference noise level.

```
for j=1:length(SM)
    SM(j)=SM(j)/noise_level - 1.0;
    if (SM(j) < min_snr)
        SM(j)=0.1*min_snr;
    end
end;
```

Everything below the $\min_{SNR}$ is scaled downnn to value $0.1\cdot\min_{SNR}$. The following plots the normalized smoothed average power versus noise reference level:

```
figure;
plot([-(length(SM)/2-1):1:-1 0 1:length(SM)/2],SM);
grid on;
xlim([-(length(SM)/2-1) length(SM)/2]);
xlabel('Frequency index (k)');
ylabel('Normalized smoothed average magnitude');
```



This result into the following plot. It reveals the presence of frequencies with signal peaks with potential frames. This will be confirmed in the sequel of the search.

After that manipulation, indices with normalized smoothed average magnitudes above $\min_{\mathrm{SNR}}$ become indices of candidate frequencies.

```
candidate=[];
for k=1:length(SM)
    if(SM(k)>min_snr)
        candidate = [candidate struct('index',k-(length(SM)/2-1)-1,'SNR',SM(k))];
    end;
end;
fprintf('The number of candidates is %d\n', length(candidate));
```

```
The number of candidates is 64
```

```
fprintf('[index 6-Hz-SNR(dB)]\n');
```

```
[index 6-Hz-SNR(dB)]
```

```
for k=1:length(candidate)
    fprintf('[%d %2.2f] ', candidate(k).index, db(candidate(k).SNR));
end
```

```
[-216 -12.50] [-215 -2.91] [-214 -2.53] [-213 -2.39] [-212 -2.47] [-211 -2.51] [-210 -2.70] [-209 -5.74]
```

```
fprintf('\n');
```

Every candidate is further examined for the presence of a frame.

The receiver has the capability to search for frames with possibly linearly or nonlinearly drifting carriers. Using the corresponding candidate frequencies as carriers, refined signal paths are searched. A path is defined by a signal tracking function

$$f_\theta : \{0 \ldots nffts - 1\} \mapsto \{-256 \ldots 0 \ldots 255\}$$

where θ is a parameter of function$f$. Let Φ denote the set of all instances of such signal tracking functions (assume it is finite size, i.e., there is a finite number of functions and the domains of their parameters are finite). There are three cases. Firstly, the tracking of a signal not subject to the Doppler effect, i.e., its frequency is not drifting, is represented by a constant function $f(i) = c$, where i is the half-symbol index and c is the carrier frequency. Secondly, the tracking of a signal subject to a Doppler effect such that the carrier frequency is drifting linearly, is represented by function $f_\delta(i) = c + \delta i$. The parameter δ represents the quantity of frequency drift per half symbol. Thirdly, tracking of a signal subject to a Doppler effect such that the carrier frequency is drifting nonlinearly, is represented by an arbitrary function.

Testing correlation with synchronization bits and resolving timing offset

In a window of two minutes, A complete frame can start anywhere from the beginning to a time delay corresponding to nine seconds (26 half symbols) into the interval.

```
max_offset=2*9*375/256-1; % half symbols ("-2" because there are 348 FFTs)
```

For the signal tracking function f_θ, the timing offset τ is the value in the range $0, \ldots, 25$ that maximizes the sum:

$$\sum_{i=0}^{161} (2s_i - 1) \left[\frac{(W_{2i+\tau, f_\theta(2i+\tau)-4} + W_{2i+\tau, f_\theta(2i+\tau)+1}) - (W_{2i+\tau, f_\theta(2i+\tau)-1} + W_{2i+\tau, f_\theta(2i+\tau)+4})}{\sum_{k=-4,-1,1,4} |W_{2i+\tau, f_\theta(i)+k}|} \right]$$

The summation measures the correlation of the spectrum magnitude around the frequency returned by the function f_θ with the synchronization bit-string s. The multiplicand $2s_i - 1$maps the synchronization bit s_i which is 0 or 1, to value -1 or 1. The term

$$W_{2i+\tau, f_\theta(2i+\tau)-4} + W_{2i+\tau, f_\theta(2i+\tau)+1}$$

is the sum of the magnitudes at the frequencies of synchronization bit value 1, while the term

$$W_{2i+\tau, f_\theta(2i+\tau)-1} + W_{2i+\tau, f_\theta(2i+\tau)+4}$$

is the sum of the magnitudes at the frequencies of synchronization bit value 0. The denominator represents the sum of all magnitudes around frequency $f_\theta(i)$.

```
% for every candidate signal, determine the timing offset
timing = [];
for k=1:length(candidate)
```

```
    % frequency index (-256..0..255 mapped to 1..512, a MATLAB array index)
    g = candidate(k).index + 257; % no drift assumed
    offset = 0; max = 0;
    % find correlation peak vs timing offset
    for tau=0:max_offset
        % compute correlation
        corr = 0;
        for i=1:162
            corr=corr + (...
                (2*s(i)-1)*...
                ( (W((2*i-1)+tau,g-4)+W((2*i-1)+tau,g+1))-(W((2*i-1)+tau,g-1)+W((2*i-1
                (W((2*i-1)+tau,g-4)+W((2*i-1)+tau,g-1)+W((2*i-1)+tau,g+1)+W((2*i-1)+tau
        end
        % is it a peak?
        if corr>max
            max = corr;
            offset = tau;
        end
    end
    timing = [timing...
        struct('index',candidate(k).index,'SNR',candidate(k).SNR,'offset',offset)];
end
fprintf('[index 6-Hz-SNR(dB) offset(half-symbols)]\n');
```

```
[index 6-Hz-SNR(dB) offset(half-symbols)]
```

```
for k=1:length(timing)
   fprintf('[%d %2.2f %d] ', timing(k).index, db(timing(k).SNR), timing(k).offset);
end
```

```
[-216 -12.50 5] [-215 -2.91 0] [-214 -2.53 0] [-213 -2.39 22] [-212 -2.47 19] [-211 -2.51 20] [-210 -2.70
```

```
fprintf('\n');
```

Calculating the soft symbols (to be revised)

The power at synchronization bits is made relative to all the power at the candidate frequency.

For each signal tracking function f_θ , over a symbol interval of length T, the signal samples correlated with the waveforms are added to obtain the total magnitude at each symbol frequency $(f = -2.2, -0.7, 0.7, 2.2)$:

$$P_{i,f} = \left| \sum_{t=iT+\tau}^{(i+1)T+\tau} x_t \cdot e^{-j2\pi[f_\theta(t)\cdot 0.73+f]t} \right|$$

with $i = 0, \ldots, 161$.

```
P=zeros(length(candidate),162,4);
freqs=[-2.2 -0.7 0.7 2.2];
for k=1:length(candidate)
    % frequency index (-256..0..255 mapped to 1..512, a MATLAB array index)
    g = candidate(k).index + 257; % no drift assumed
    for i=1:162
        for f=1:4
            for j=1:255
                t=(i-1)*256 + timing(k).offset + j;
                P(k,i,f) = P(k,i,f) + complex(I(t),Q(t))*exp(-1i*2*pi*(g*0.73+freqs(f)
            end
        end
    end
end
```

The list of four magnitudes $P_{i,f}$ are used to calculate soft symbols. A soft symbol represents a value and its quality. Receive quality metrics are associated with the symbols. This information is used in the decoding process. The most likely symbols are selected first. A de-interleaving procedure reorders the 162 data soft symbols. The resulting 162 soft symbols are passed to a FEC decoder.