Table of Contents


```
%RTL_CFO
%
% Estimation of carrier frequency offset based on autocorrelation.
% Uses GSM frequency synchronization burst, FCCH
%
% By R.W.

% Minimize the risk of using old variables
clear all, close all
```

Initialization

```
% Define yourself:
%      Upperscale variables
%      Functions rw_*
%rw_ini_cfo
MAXLAG = 5;
% GSM parameters
% A TDMA frame consists of 8 time slots and is 4.615 ms long. A control
% channel multi-frame is composed of 51 TDMA frames (numbered 0-50) and is
% 235.4 ms long. A base station transmits a frequency correction burst in
% regular positions.
% The frequency correction channel repeats every 51 frames and the burst
% occurs at the TS0 time-slot in frames 0,10,20,30 and 40 in the control
% channel multi-frame
% The FCCH burst is a burst of a pure frequency tone
% at 1/4th the bitrate of GSM or (1625e3/6)/4 = 67.7083KHz offset from the
% center frequency.
% GSM frame structure: www.sharetechnote.com/html/FrameStructure_GSM.html
% GSM radio interface: https://www.3g4g.co.uk/GsmGprsEdge/gsm003.pdf
%
% Get GSM downlink center frequency close to your whereabouts from
% cellmapper.net
%expFreq = 949.6e6;
%expFreq = 951.4e6;
% Elisa base station close to Maarintie
expFreq = 949e6;

% Number of symbols in GSM; slots and frames
nSym_per_slot = 625/4;
nSlot_per_frame = 8;
nSym_per_frame = nSym_per_slot*nSlot_per_frame;
```

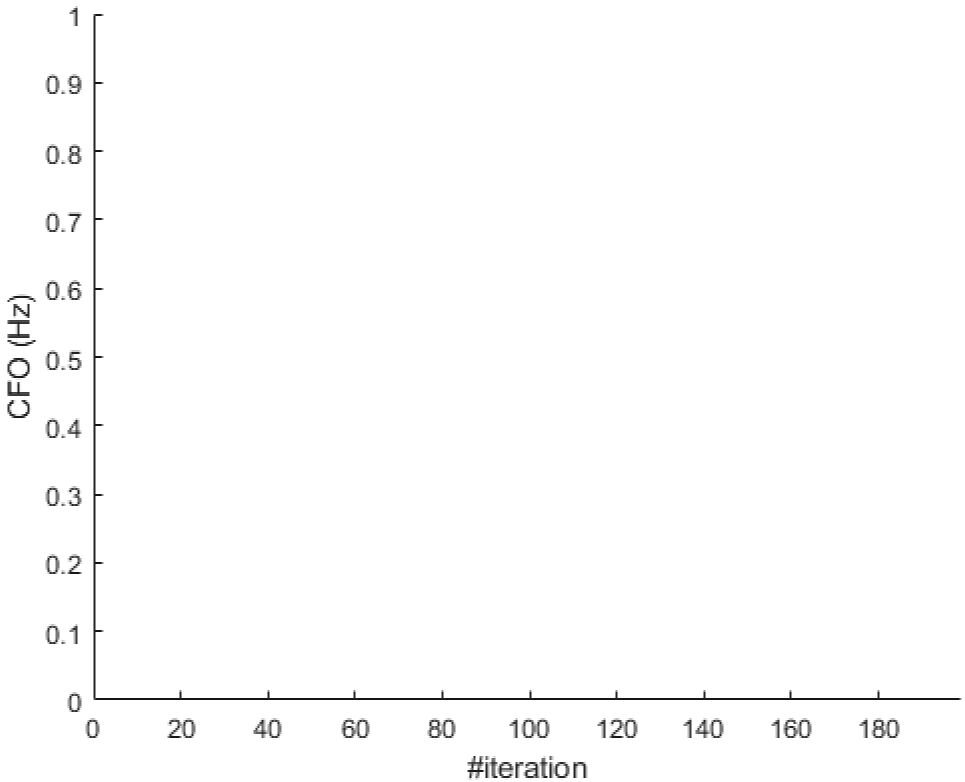
```
% Symbols in FCCH burst
nSym_FCCH = 142;
% Symbol rate
symRate = 1625e3/6;
% The length of FCCH in time
t_FCCH = nSym_FCCH/symRate;

% Oversampling factor w.r.t. the symbol rate. At the same time it is the
% down-sampling factor back to the symbol rate
NDEC = 6;

% Front-end sampling rate: symbol rate x oversampling factor
% The comm.SDRRTLReceiver accepts only integer values for sampling rate
FESR = symRate*NDEC;

% #GSM frames to read, 51 frames is one multiframe but comm.SDRRTLReceiver is
 not
% able to read that much.
nFrame = 20;
% #Samples to read becomes
nSample = nSym_per_frame* nFrame *NDEC;

% Number of multiframes to read. Change at will
nMulti = 2e2;

% Your dongle's PPM here. Use PPM that matches to the passband of the 2nd
% narrowband filter. When the second filter is a low-pass filter, like in this
% example, the PPM is tuned intentionally such that the 67.7 KHz FCCH falls
% close to zero frequency. On the other hand, the frequency should not be
% too close to zero as the performance is difficult to verify then.
% You can consider a passband filter as the second filter as well.
PPM = 5;

% Get the filters. First one operates at the symbol rate, the second one at
% down-sampled rate. The second filter
[FLOW1, FLOW2] = rw_cfoFilters(FESR,NDEC);
```

Visualization objects

Max #samples the driver can read read = 375e3 Sampling frequencies 225e3 < R <= 300e3 or 900e3 < R <= 3200e3

```
hSDRrRx = comm.SDRRTLReceiver(...
 'RadioAddress', '0',...
    'CenterFrequency',     expFreq, ...
    'EnableTunerAGC',      true, ...
    'SampleRate',          FESR, ...
    'SamplesPerFrame',     nSample, ...
    'FrequencyCorrection', PPM, ...
    'OutputDataType',      'double');


% Align figure positions. Left corner + height and width
```

```
pv = [30 40 30 40];
hSpectrumAnalyzer = dsp.SpectrumAnalyzer(...
    'Name',             'Received spectrum',...
    'Title',            'Received spectrum', ...
    'SpectrumType',     'Power density',...
    'FrequencySpan',    'Full', ...
    'SampleRate',       FESR, ...
    'YLimits',          [-40,0],...
    'SpectralAverages', 1, ...
    'FrequencySpan',    'Start and stop frequencies', ...
    'StartFrequency',   -60e3, ...
    'StopFrequency',    60e3,...
    'ViewType',         'spectrum and spectrogram',...
    'Position',         figposition(pv));
 %'FrequencyOffset', offs,...

% Spectral averaging slows down the changes so that the display is easier
% to follow. Does not work with spectrogram option
hSA2 = clone(hSpectrumAnalyzer);
set(hSA2,'Name','CFO-corrected signal','Title','CFO-corrected signal',...
    'SpectralAverages', 1,...
    'SampleRate',       FESR/NDEC, ...
    'Position', figposition([pv(1)+30,pv(2:4)]));

% Display time-domaing signal. Alternative to timescope
hTimeArray = dsp.ArrayPlot(...
    'Name', 'Channel outputs',...
    'NumInputPorts', 1,...,
    'YLimits', [-1,1],...,
 'Position', figposition([pv(1),pv(2)-40,pv(3:4)]),...
    'PlotType',  'Line');

% Time scope. Parameters can be tuned while running
hTimeScope = timescope(...
 'Position', figposition([pv(1),pv(2)-40,pv(3:4)]));

% Figure to display the frequency offset estimates in the loop.
fc = figure('Name', 'CFO');
hc = animatedline('Color','b','Marker','.');
axis([0 nMulti-1 -inf inf]); ylabel('CFO (Hz)'), xlabel('#iteration')
```



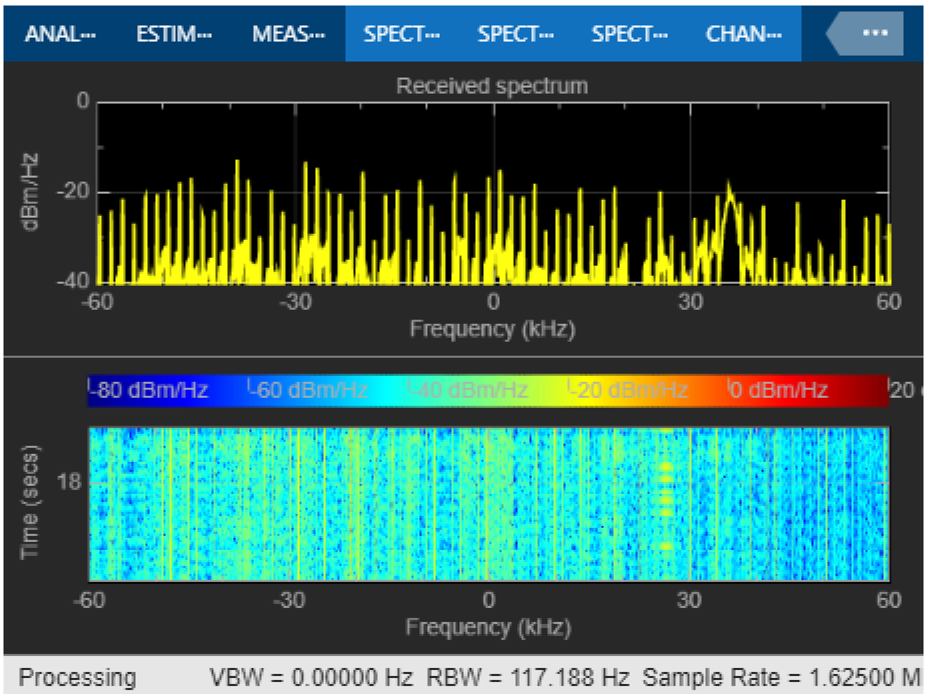
Stream Processing

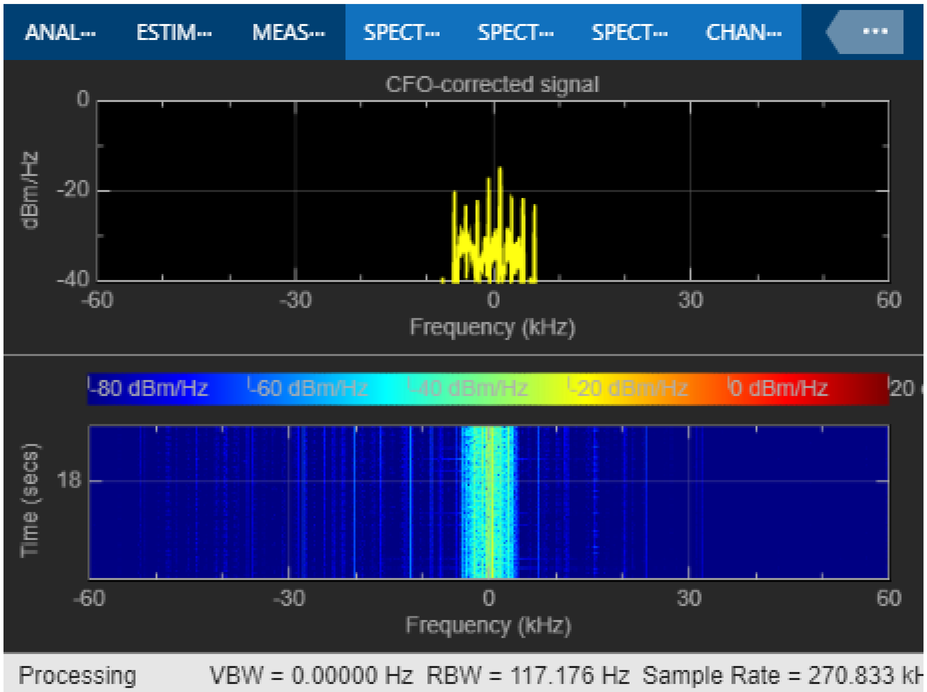
```
if isempty(sdrinfo(hSDRrRx.RadioAddress))
    warning(message('SDR:sysobjdemos:MainLoop'))
    return
end

fprintf('Simulation time %f s  \n', nSample/FESR*nMulti)
% The loop should be as simple as possible to keep the operation fast

% Store the evolution of the estimated frequency offset
offsvec = zeros(1,nMulti);
% Error flags
flg = zeros(1,nMulti);

% Count the wall clock time
tic;
for ii = 1:nMulti
    [rxSig, ~] = step(hSDRrRx);
    rxSig = rxSig - mean(rxSig);  % Remove DC component

 % Display the received signal. You should see 200 KHz GSM signal
    hSpectrumAnalyzer(rxSig);

    % Two-stage decimation. First to the symbol rate 1625e3/6 and then the
```

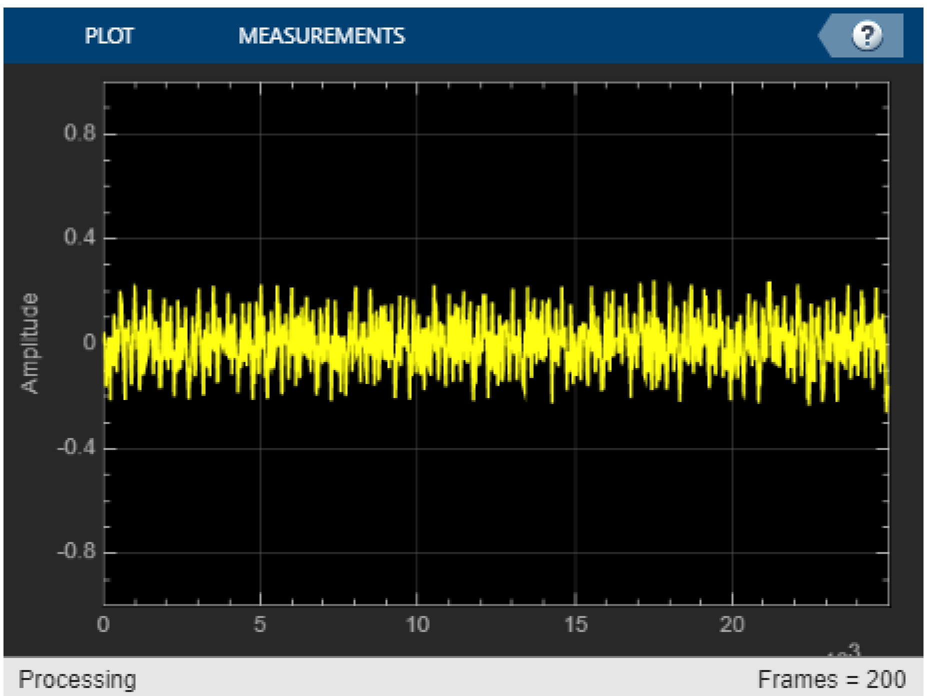
```
    % narrow-band filter to reduce noise
    rxFlt = filter(FLOW1,1,rxSig);
    % This is now in symbol rate
    rxFlt2 = filter(FLOW2,1,rxFlt(1:NDEC:end));

    % Display the spectrum after decimation at symbol rate
    % Verify that the FCCH falls into the passband
    hSA2(rxFlt2);

    % Display the time-domain signal to verify bursts are there
    % Only the real part to make the display more clear
    hTimeArray(real(rxFlt2));
    %hTimeScope(real(rxFlt2));

    % Locate the FCCH burst and optionally return more than one burst.
    % Note that finding the maximum is of complexity O(n) operation and
 sorting at best
    % O(n*log(n)) operation, where n is the number of samples.
    % At most five bursts are returned so sorting the whole vector would be an
 overkill.
    % No additional input arguments like burst length etc., because they are
    % fixed numbers.
    % Optional error flag indicates an error in the position estimate. It
    % estimates two bursts and if the difference of the burst indices does
    % not match to multiple of 10 frames =- window of few samples, the error
 flag is
    % raised
    [burst,flg(ii)] = rw_getBurst(rxFlt2);

    % Estimate frequency (not angular frequency) offset from the
    % compensated signal using Fitz and Luise&Regiannini.
    % Requires sampling frequency and MAXLAG, the maximum number of lags used

    fitzoffs = rw_fitz(burst, FESR/NDEC, MAXLAG); % MAXLAG == 5
    lroffs = rw_lr(burst, FESR/NDEC, MAXLAG);

    % Update the offset. fitzoffs & lroffs contain estimates for 1:MAXLAG,
    % and only the last one is needed. Estimates using a different number
    % of lags can be used to study the stability of the estimate
    offsvec(ii) = fitzoffs(end);

    % Display only the values that were not flagged erroneous.
    if flg(ii), offsvec(ii) = NaN; end

    % Display the estimates one-by-one instead of waiting till the end of
    % the loop
    addpoints(hc, ii, offsvec(ii)), drawnow limitrate
end
fprintf('Elapsed time %f s\n', toc)
%plot(offsvec), ylabel CFO, xlabel #iteration

% 1) Calculate the frequency offset from the estimates
% 2) Calculate the PPM corresponding to the frequency offset taking into
% account:
```

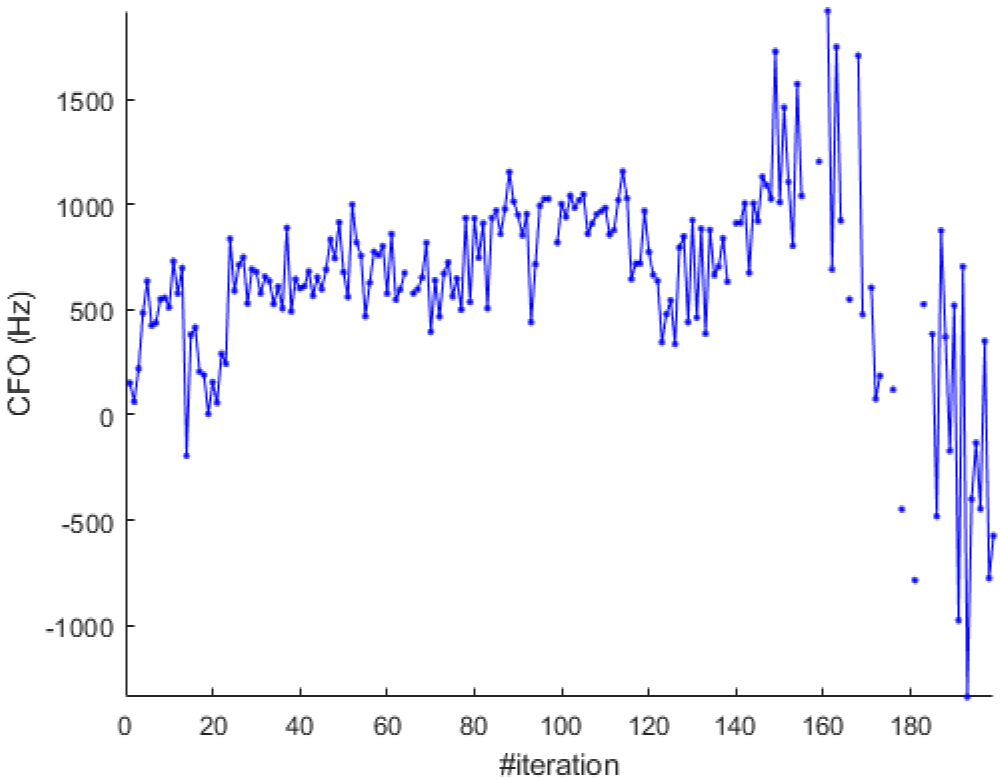
```
% - The PPM you set in the function call
% - FCCH is 67.7 KHz with respect to the center frequency

% Convert the frequecy offset to PPM
referenceFrequency = 67.7 * 10^3;
ppmValues_1 = (offsvec(1:nMulti-1) / referenceFrequency) * 1e6;
ppmValues_2 = (offsvec(2:nMulti) / referenceFrequency) * 1e6;
integerPPM = round(abs(ppmValues_1 - ppmValues_2));
figure(2)
plot(integerPPM)
title("Integer PPM Values")
xlabel("iteration")
```

Simulation time 18.461538 s
Elapsed time 26.837784 s

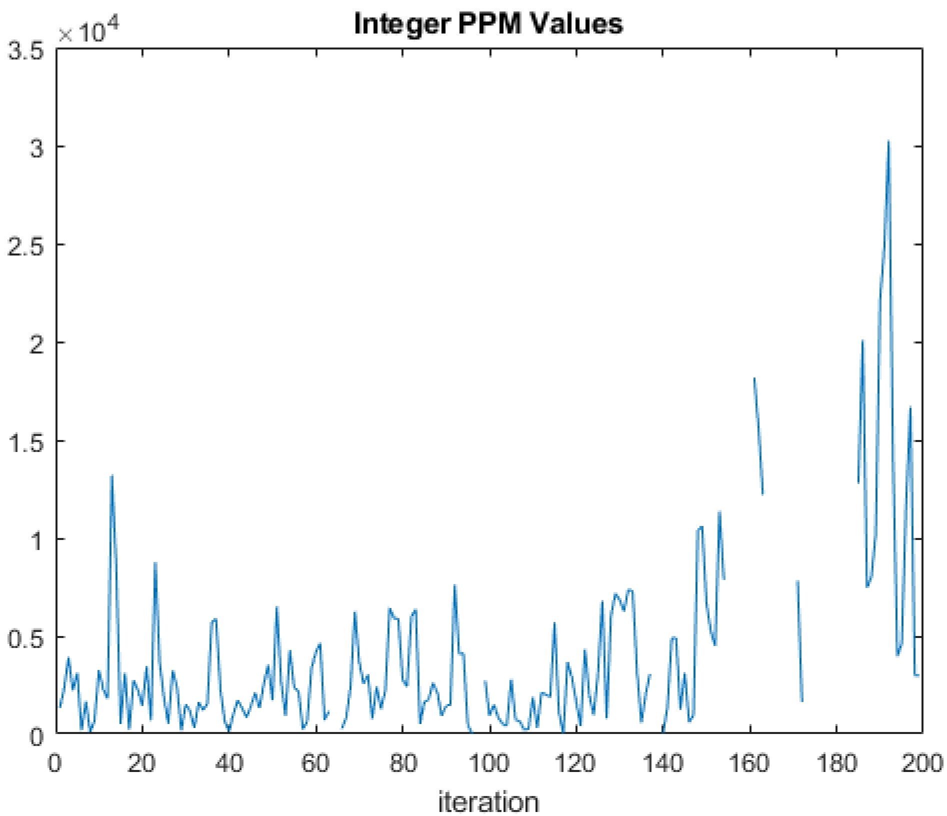



ANAL···
ESTIM···
MEAS···
SPECT···
SPECT···
SPECT···
CHAN···
CFO-corrected signal
dBm/Hz
0
-20
-40
-60
-30
0
30
60
Frequency (kHz)
-80 dBm/Hz
-60 dBm/Hz
-40 dBm/Hz
-20 dBm/Hz
0 dBm/Hz
Time (secs)
18
Processing
VBW = 0.00000 Hz RBW = 117.176 Hz Sample Rate = 270.833 kH


PLOT
MEASUREMENTS
Amplitude
0.8
0.4
0
-0.4
-0.8
0
5
10
15
20
Processing
Frames = 200



CFO (Hz)
1500
1000
500
0
-500
-1000
0
20
40
60
80
100
120
140
160
180
#iteration



Integer PPM Values
$\times 10^4$
3.5
3
2.5
2
1.5
1
0.5
0
0
20
40
60
80
100
120
140
160
180
200
iteration

Release all System objects

```
release(hSDRrRx);
clear hSDRrRx
```

Answer the questions

```
% Q: Which algorithm works the best, Fitz or L&R? Can you notice any
 difference?
% A: The Fitz works the best, because the result data of Fitz are more stable
 than the L&R.
% Fitz algorithm is based on the principle of maximum likelihood estimation
% and therefore it can theoretically provide near-optimal performance.
% However, the computational complexity of Fitz is relatively high,
% requiring more mathematical operations

% Q: Does the estimate look more reliable with large M?
% A: Yes, the result look more close to the real frequency offset(the
 frequency offset spectrum).

% Q: Does the CFO stabilize over time when the dongle is used?
% A: No, the result is unstable, the result fluctuated greatly at some
 moments.

% Q: Convert the CFO estimate to PPM. Does the (integer) PPM stay the same
% even when the CFO estimate is changing
% A: As the figire 2 shows, it's not stay the same.
```

Published with MATLAB® R2023a