Table of Contents


```
%RTL_FILT
%
% Extraction of 19 KHz FM carrier using different filter structures
% Implements similar filters as in Egg,harris, "Forming Narrowband Filters
% at a Fixed Sample Rate with Polyphase Down and Up Sampling Filters"
%
% By R.W.

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

Initialization

```
% Define yourself:
%       Upperscale variables
%       Functions rw_*
%
% Setting some parameters. The p-code file containing the filter designs
% and functions you must implement yourself.
[FPASS, FPROTO, FLOW, FPROTOI, FINT, FPER, NDEC, FESR, FPOLY] = rw_ini_filt();

% Samples to read at a time
% Implementation of the polyphase filter is easier when the length of the
% input signal is a multiple of the down-sampling factor.
nSample = NDEC*1024;
% FM station, YLE Puhe
expFreq = 103.7e6;
% Your dongle's PPM here. Compensate the frequency offset as
% well as possible, because passbands are narrow
PPM = 38;

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

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

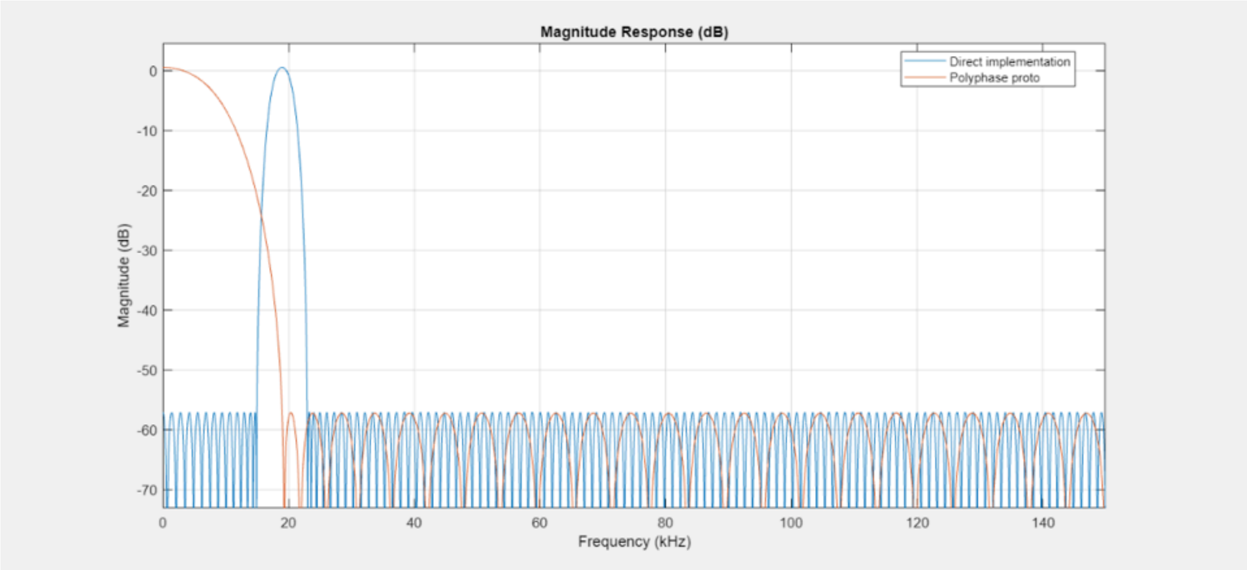
```
pv = [30 40 30 40];
% Tune YLimits according to your signal strength so that you can see the
% spectrum.
hSpectrumAnalyzer = dsp.SpectrumAnalyzer(...
    'Name',             'Received spectrum',...
    'Title',            'Received spectrum', ...
    'SpectrumType',     'Power density',...
    'FrequencySpan',    'Full', ...
    'SampleRate',       FESR, ...
    'YLimits',          [-40,0],...
    'SpectralAverages', 10, ...
    'FrequencySpan',    'Start and stop frequencies', ...
    'StartFrequency',   -30e3, ...
    'StopFrequency',    30e3,...
    'Position',         figposition(pv));
 %'FrequencyOffset', offs,...

% SpectralAverages slows down the changes so that the display is easier
% to follow
hSA2 = clone(hSpectrumAnalyzer);
set(hSA2,'Name','Filtered spectrum','Title','Filtered spectrum',...
    'SpectralAverages', 10,...
    'NumInputPorts',3,...
    'ShowLegend', true,...
    'ChannelNames', {'direct','IFIR','poly'},...
    'Position', figposition([pv(1)+30,pv(2:4)]));

% Show a single filtered signal only
hSA2s = clone(hSpectrumAnalyzer);
set(hSA2s,'Name','Filtered spectrum','Title','Filtered spectrum',...
    'SpectralAverages', 10,...
    'NumInputPorts',1,...
    'Position', figposition([pv(1)+30,pv(2:4)]));

% Down-sampled polyphase spectrum
hSA3 = clone(hSpectrumAnalyzer);
set(hSA3,'Name','Decimated signal','Title','Decimated signal',...
    'SampleRate', FESR/NDEC,...
    'YLimits',          [-50,0],...
    'StartFrequency',   -FESR/2/NDEC, ...
    'StopFrequency',    FESR/2/NDEC,...
    'Position', figposition([pv(1)+15,pv(2)-15,pv(3:4)/2]));
```

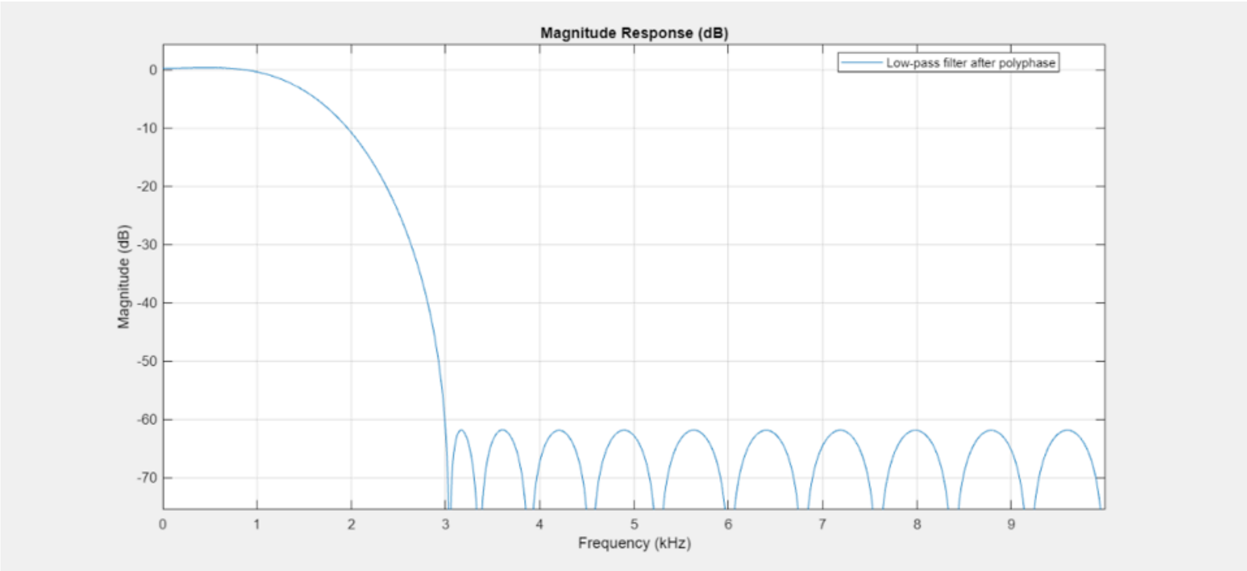
Display all filters

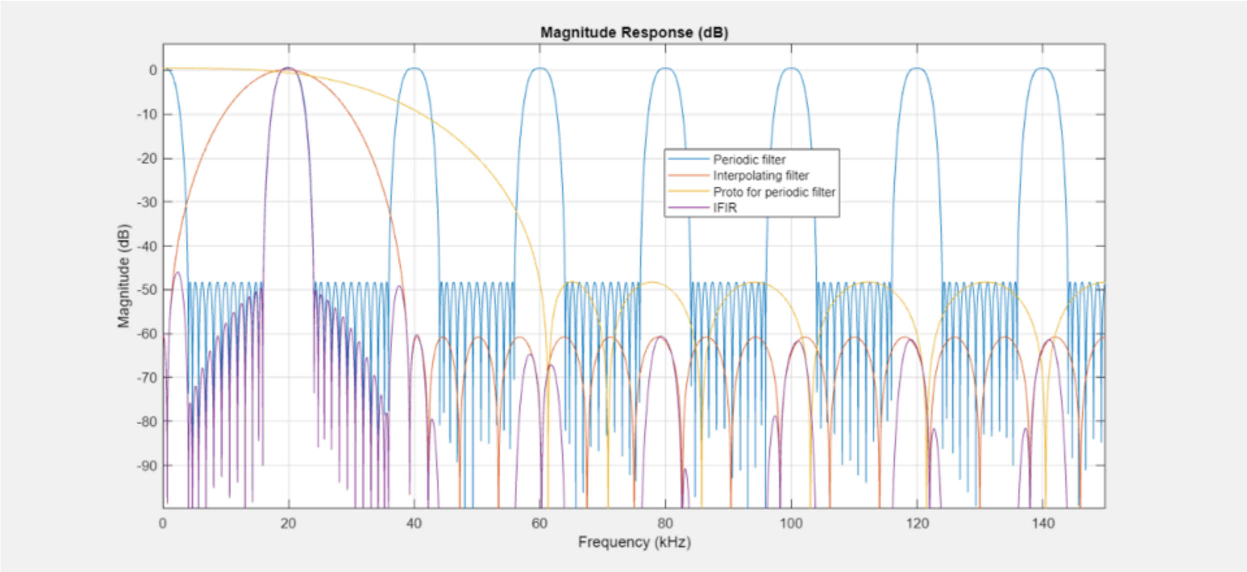
Bandpass filter FPASS = direct implementation and the prototype filter FPROTO for the polyphase filter. All filters are designed using first firpmord() and then firpm()

```
hnd1 = fvtool(FPASS,1,FPROTO,1,'Fs',FESR);
legend(hnd1,'Direct implementation','Polyphase proto')

% Low-pass filter running at down-sampled frequency to extract the carrier
% not at -1 KHz.
```

```
hnd2 = fvtool(FLOW,1,'Fs',FESR/NDEC);
legend(hnd2,'Low-pass filter after polyphase')

% IFIR filter. FPROTOI is the prototype filter, FPER is the sparse version
% with multiple compressed copies of FPROTOI's frequency response. The IFIR
% is then the convolution of the impulse responses although the filtering
% is done separataly with the two filters to get the advantage of zeros in
% FPER impulse response.
hnd3 = fvtool(FPER,1,FINT,1,FPROTOI,1,conv(FPER,FINT),1,'Fs',FESR);
legend(hnd3,'Periodic filter','Interpolating filter',...,
    'Proto for periodic filter','IFIR')
```





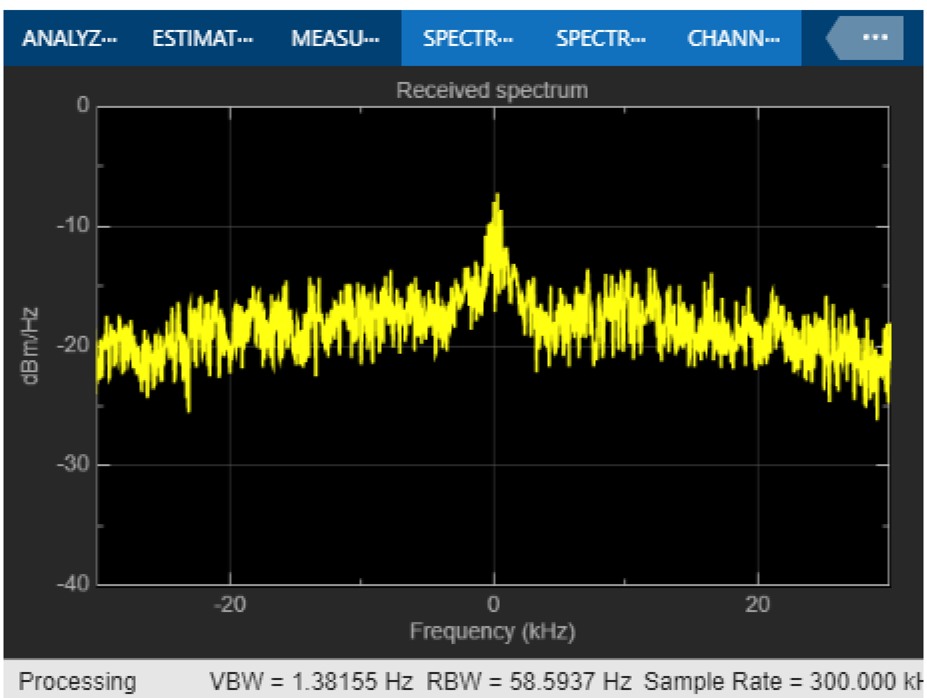


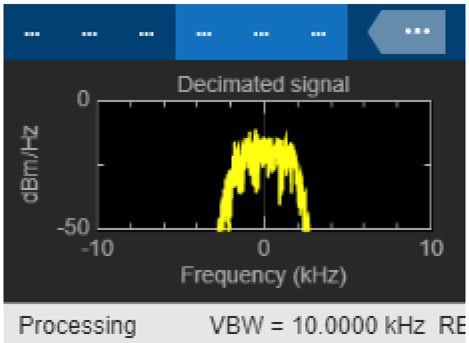
Stream Processing

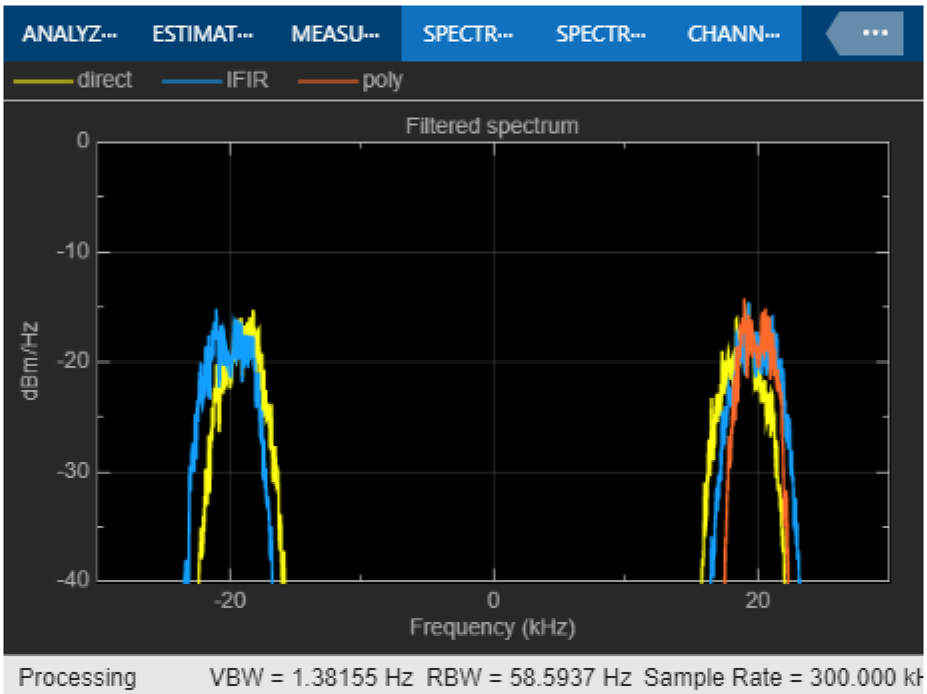
```
nFrame = 1e3;

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

fprintf('Simulation time %f [s]    \n', nSample/FESR*nFrame)
% The loop should be as simple as possible to keep the operation fast
% Timing the loop
tic;
for kk = 1:nFrame
    [rxSig, ~] = step(hSDRrRx);
    rxSig = rxSig - mean(rxSig);  % Remove DC component

 % Display the received signal
    hSpectrumAnalyzer(rxSig);

    % Direct implementation of the bandpass filter
    % 19 KHz center frequency, 2 KHz two-sided bandwidth, 2 KHz transfer band
    % Real coefficients
    fltDir = filter(FPASS,1,rxSig);

    % Interpolated FIR filter. The order of the filters does not matter
    % because sampling rate is not changed
    fltIF = filter(FINT, 1, filter(FPER,1,rxSig));


    % Polyphase implementation, i.e. the first filter in Fig. 7 in Egg&harris'
    % paper.
    % FPOLY is a matrix whose rows contain the
    % polyphase components. The third argument specifies the frequency zone
```

```
    % between [0,NDEC-1]
    decPol = rw_polydecmod(FPOLY,rxSig,1);
    % Low-pass filter operating at the lower sampling rate, 2 KHz two-sided
 BW,
    % 2 KHz transfer band, the 2nd filter in Fig. 7
    dec2Pol = filter(FLOW,1,decPol);
    % Sound check. Display the output of the polyphase
    hSA3(dec2Pol)

    % Upsample back to the front-end sampling rate
    % Compensate the gain by multiplying with NDEC if not done in the
 polyphase filter
    % Upsample the baseband signal. Together with the modulation, the 3rd
    % filter in Fig. 7.
    fltPol = NDEC*rw_polyint(FPOLY,dec2Pol);

    % Modulate the baseband signal to the exp(j*2*pi/
NDEC*[0:nSample-1].')first Nyquist zone.
    fltPol = fltPol .* exp(j*2*pi/NDEC*[0:nSample-1].');


    % Display the spectra after filtering with the three filters
    hSA2(fltDir,fltIF,fltPol)

    % One channel only
    %hSA2s(fltDir);
end
fprintf('Clock receive time %f [s]\n', toc)
```

Simulation time 51.200000 [s]
Clock receive time 53.454940 [s]







Release system objects

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

calculate the number of multiplications per second the filters have

Direct implementation of the bandpass filter: FPASS

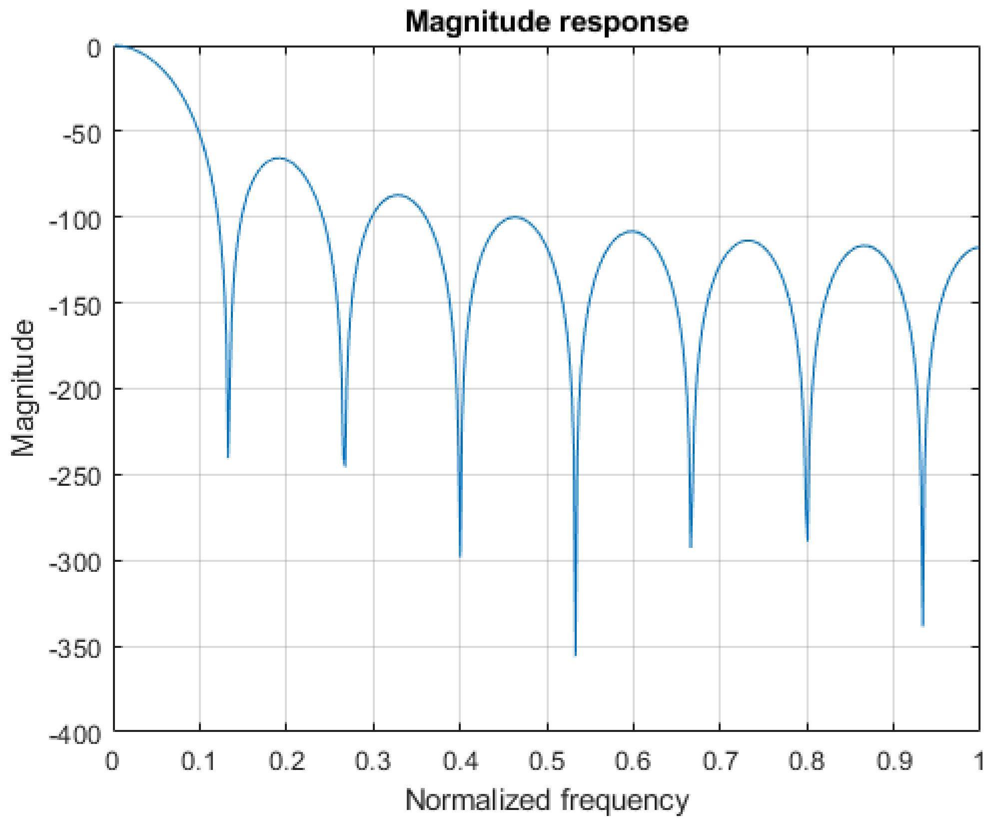
```
num_multiplications_FPASS = length(FPASS) * FESR;

% Interpolated FIR filter: FPER and FINT
num_multiplications_IFIR = (length(FINT) + length(FPROTOI)) * FESR;

% Polyphase implementation: FPLOY
[num_row, num_col] = size(FPOLY);
```

```
num_multiplications_FPOLY = (num_row * (num_col+1) * FESR) * 2 + length(FLOW)
 * FESR / NDEC;

fprintf("The number of multiplications per second of Direct implementation of
 the bandpass filte is: %d\n",num_multiplications_FPASS)
fprintf("The number of multiplications per second of Interpolated FIR filter
 is: %d\n",num_multiplications_IFIR)
fprintf("The number of multiplications per second of Polyphase implementation
 is %d\n",num_multiplications_FPOLY)
```

```
The number of multiplications per second of Direct implementation of the
 bandpass filte is: 70800000
The number of multiplications per second of Interpolated FIR filter is:
 15600000
The number of multiplications per second of Polyphase implementation is
 45480000
```

Implementing CIC filter

```
den = [1 -1];
num = zeros(1,NDEC+1);
num(1) = 1;
num(NDEC+1) = -1;
N = 5; % number of stage of CIC filter

%generate the CIC filter
for i = 1 : N -1
    if i == 1
        cicFilter_num = conv(num,num);
        cicFilter_den = conv(den,den);
    else
        cicFilter_num = conv(cicFilter_num,num);
        cicFilter_den = conv(cicFilter_den,den);
    end
end
[f,w] = freqz(cicFilter_num,cicFilter_den);
f = f/max(f); % normalize the magnitute
plot(w/pi,20*log10(abs(f)))
xlabel('Normalized frequency'), ylabel("Magnitude")
title('Magnitude response'), grid on
% When the number of CIC stages equals to 5, the attenuation is 65dB.
% When the number of CIC stages equals to 4, the attenuation is 52dB.
% Therefore the 5 stages is needed.
```



Published with MATLAB® R2023a