UDC 654.1:621.396.7
DOI https://doi.org/10.32782/2663-5941/2024.3.2/07

Ikhsanov Sh.M.
Admiral Makarov National University of Shipbuilding

Diakonov O.S.
Admiral Makarov National University of Shipbuilding

IMPROVING NONCOHERENT DISCRIMINANT FM SIGNAL PROCESSING ALGORITHM FOR SOFTWARE-DEFINED RADIO AND ITS MATLAB IMPLEMENTATION

The paper analyzes the drawbacks of an non coherent discriminant algorithm used to detect the mono signal from broadcasting FM radio stations. It has been shown that the MATLAB implementation of this algorithm for the RTL-SDR receiver suffers from low selectivity, leading to the phenomenon of overlapping sounds from neighboring broadcasting FM radio stations, which results in interference. To eliminate this disadvantage an improved algorithm was developed. This algorithm uses a low-frequency finite impulse response (FIR) filter in the FM signal band before the decimator. The research involved three types of FIR filters: Hamming, Kaiser, and Parks-McClellan. Kaiser and Parks-McClellan FIR filters have a high order and provide an almost vertical cutoff, with slightly different levels of side lobes. Kaiser filters start at −40 dB and drop to −70 dB gradually, while Parks-McClellan filters immediately reach a constant −58 dB level. Hamming filter has a lower order, with similar levels of side lobes, but a flatter main lobe (losses at cutoff frequency are 6 dB and side lobes begin at ~220 kHz). After the FIR decimator, based on the Parks-McClellan algorithm, starting at 20 kHz, attenuation of about 40 dB is provided. To better understand the influence of various factors in the proposed processing method, a model was created using two broadcasting FM signals with a frequency difference of 700 kHz, simulating the situation where signal reception must take place from one radio station in the presence of another powerful radio station located nearby in frequency. In the experiments, the excess of the interference signal over the desired signal is assumed to be 20 dB. The inverse of the standard deviation of the original and filtered signals is used as a criterion to compare the FIR filters. It has been shown that over the entire range of signal-to-noise ratios (0–20 dB), the Parks-McClellan FIR filter performs best. The Hamming filter, with a relatively low order, is inferior by 5 % and the Kaiser filter is inferior by 7 %. Implementation of the proposed algorithms is provided in the MATLAB programming language.

Key words: *FM broadcast, software-defined radio (SDR), digital signal processing, FM signal, FIR filter.*

Formulation of the problem. The development of the concept of software-defined radio (SDR) in recent years has allowed a paradigm shift from a traditional hardware or digital radio system towards a radio system with software signal processing [1]. This allows you to change the characteristics of the radio system programmatically without making significant changes to its hardware. An example of an SDR is the common RTL-SDR receiver. The quality of the SDR is determined by the selected processing algorithms. It should be noted that optimization of algorithms based on the simulated signal does not guarantee their optimality for real signals.

Analysis of recent research and publications. Among the algorithms for FM signal decoding, the FM_Demod block, implemented in the GNU Radio project, can be distinguished. This is also an example of an FM broadcast receiver provided by MathWorks in the form of a MATLAB script. Testing these algorithms when decoding real FM radio stations using an RTL-SDR receiver revealed a significant drawback: the lack of selectivity when there are several radio stations operating in the same frequency band. This manifests itself as interference from a more powerful station superimposed on the sound of the weaker stations. Therefore, the goal is to improve these algorithms to overcome this limitation.

Task statement. The aim of this work is to develop and implement an improved discriminant algorithm for digital signal processing using FIR filters in SDR technology. The algorithm will be implemented in MATLAB to achieve this goal.

Outline of the main material of the study. Let's start processing signals from FM radio stations with a fairly simple non-coherent discriminant algorithm for detecting a mono signal. The MATLAB program is given in the book's file library [2, p. 348] (Listing 1).

In analytical form, such an FM signal is written as follows [5]:

$$s_{fm}(t) = A_0 \exp\left(j(2\pi f_c t + 2\pi K_{fm} \int_{-\infty}^{t} s_i(\tau) d\tau) \right), \quad (1)$$

where f_c is the carrier radio frequency, K_{fm} is the frequency deviation coefficient, $s_i(t)$ is the information (sound) signal. The carrier frequency is modulated not by the audio signal itself, but by an integral of it.

Listing 1. Implementation of the non-coherent discriminant algorithm for detecting a mono signal

```
% RTL-SDR(rx) FM Mono Non Coherent Discriminator
%  Demodulator
function rtlsdr_fm_discrim_demod_matlab
%% PRINT FILE INFORMATION HYPERLINK TO COMMAND WINDOW
disp(['View file information for <a href="matlab:
mfileinfo(''',mfilename,''')">',mfilename,'</a>']);
%% PARAMETERS (edit)
offline = 0; % 0 = use RTL-SDR, 1 = import data
% path to signal
offline_filepath = 'rec_data\wfm_mono.mat';
rtlsdr_id = '0'; % stick ID
rtlsdr_fc  = 102.8e6; % tuner centre frequency in Hz
rtlsdr_gain = 50; % tuner gain in dB
rtlsdr_fs = 2.4e6; % tuner sampling rate
rtlsdr_ppm = 0; % tuner parts per million correction
rtlsdr_frmlen = 256*25;  % output data frame size
rtlsdr_datatype  = 'single';% output data type
deemph_region  = 'eu'; % set to either eu or us
audio_fs = 48e3; % audio output sampling rate
sim_time = 3600; % simulation time in seconds
%% CALCULATIONS (do not edit)
% calculate time for 1 frame of data:
rtlsdr_frmtime = rtlsdr_frmlen/rtlsdr_fs;
if deemph_region == 'eu' % find de-emphasis filter coeff
   [num,den] = butter(1,3183.1/(audio_fs/2));
elseif deemph_region == 'us'
   [num,den] = butter(1,2122.1/(audio_fs/2));
else
   error('Invalid region for de-emphasis filter – …
        must be either "eu" or "us"');
end
%% SYSTEM OBJECTS (do not edit)
if offline == 1   % check if running offline
   % link to an rtl-sdr data file
   obj_rtlsdr = import_rtlsdr_data(...
      'filepath', offline_filepath,...
      'frm_size', rtlsdr_frmlen,...
      'data_type',rtlsdr_datatype);
   rtlsdr_fs = 240e3; % reduce sampling rate
   % fir decimator - fs = 240kHz downto 48kHz
   obj_decmtr = dsp.FIRDecimator('DecimationFactor', 5,...
      'Numerator',…
firpm(100,[0,15e3,20e3,(240e3/2)]/(240e3/2),...
      [1 1 0 0], [1 1], 20));
else
   % link to a physical rtl-sdr
   obj_rtlsdr = comm.SDRRTLReceiver(rtlsdr_id,...
      'CenterFrequency', rtlsdr_fc,...
      'EnableTunerAGC', false,...
      'TunerGain', rtlsdr_gain,...
      'SampleRate', rtlsdr_fs, ...
      'SamplesPerFrame', rtlsdr_frmlen,...
      'OutputDataType', rtlsdr_datatype,...
      'FrequencyCorrection', rtlsdr_ppm);
```

```
    % fir decimator - fs = 2.4MHz downto 48kHz
    obj_decmtr = dsp.FIRDecimator('DecimationFactor',… 50, 'Numerator',…
firpm(350,[0,15e3,48e3,(2.4e6/2)]/(2.4e6/2),...
        [1 1 0 0], [1 1], 20));
end;
% iir de-emphasis filter
obj_deemph = dsp.IIRFilter('Numerator', num,...
    'Denominator', den);
obj_delay = dsp.Delay; % delay
obj_audio = dsp.AudioPlayer(audio_fs); % audio output
% spectrum analyzers
obj_spectrummod   = dsp.SpectrumAnalyzer(...
    'Name', 'Spectrum Analyzer Modulated',...
    'Title', 'Spectrum Analyzer Modulated',...
    'SpectrumType', 'Power density',...
    'FrequencySpan', 'Full',...
    'SampleRate', rtlsdr_fs);
obj_spectrumdemod = dsp.SpectrumAnalyzer(...
    'Name', 'Spectrum Analyzer Demodulated',...
    'Title', 'Spectrum Analyzer Demodulated',...
    'SpectrumType', 'Power density',...
    'FrequencySpan', 'Full',...
    'SampleRate', audio_fs);
%% SIMULATION
% if using RTL-SDR, check first if RTL-SDR is active
if offline == 0
    if ~isempty(sdrinfo(obj_rtlsdr.RadioAddress))
    else
      error(['RTL-SDR failure. Please check connection to ',...
          'MATLAB using the "sdrinfo" command.']);
    end
end
run_time = 0; % reset run_time to 0 (secs)
% loop while run_time is less than sim_time
while run_time < sim_time
    % fetch a frame from obj_rtlsdr (live or offline)
    rtlsdr_data = step(obj_rtlsdr);
% upd. modulated spectrum analyzer window with new data
    step(obj_spectrummod, rtlsdr_data);
    % implement frequency discriminator
    discrim_delay = step(obj_delay,rtlsdr_data);
    discrim_conj  = conj(rtlsdr_data);
    discrim_pd = discrim_delay.*discrim_conj;
    discrim_arg = angle(discrim_pd);
    % decimate + de-emphasis filter data
    data_dec = step(obj_decmtr,discrim_arg);
    data_deemph = step(obj_deemph,data_dec);
%upd. 'demodulated' spec. analyzer window with new data
    step(obj_spectrumdemod, data_deemph);
    % output demodulated signal to speakers
    step(obj_audio,data_deemph);
   % update run_time after processing another frame
    run_time = run_time + rtlsdr_frmtime;
end
end
```

Before analyzing the processing algorithm in detail, let's take a look at how successfully the program decodes the signals from FM radio stations. With the set parameters, even the strongest station is poorly heard, while the rest of the stations are not heard at all or are barely audible against the background of strong interference. Let's now consider the processing algorithm itself. The complex IQ-samples received from the receiver are delayed by one clock cycle (at the specified parameters 38400 IQ-complex samples are received per cycle with an interval of $\Delta t = 1/(2.4\text{ MHz}) = 0.417$ microseconds). To do this, a DSP object is created beforehand. During processing, the created object is called with the command: discrim_delay = step(obj_delay, rtlsdr_data). The Delay() function allows you to set the delay time, and by default, a shift of one clock cycle is made. Next, a complex-conjugate array is created using the count() function, and a complex element-by-element multiplication of two arrays is performed.

Let's consider what happens to the signal given by formula (1):

$$s_p(t_k) = A_0 \exp\left(j(2\pi f_c t_k + 2\pi K_{fm} \int_{-\infty}^{t_k} s_i(\tau)d\tau) \right) \times$$
$$A_0 \exp\left(-j(2\pi f_c t_{k-1} + 2\pi K_{fm} \int_{-\infty}^{t_{k-1}} s_i(\tau)d\tau) \right) = \qquad (2)$$
$$A_0^2 \exp\left(j(2\pi f_c \Delta t + 2\pi K_{fm} \int_{t_{k-1}}^{t_k} s_i(\tau)d\tau) \right).$$

Note that the carrier frequency f_s is only included in expression (2) as the value for the frequency mismatch between the transmitter and receiver. After the signal is received at the frequency of the radio station, it is shifted to zero frequency. The first term under the exponential in the final part of formula (2) is a constant and is very small, so it can be ignored. Since the amplitude of the sound within the quantization interval Δt changes only slightly, the integral of the second term $\int_{t_{k-1}}^{t_k} s_i(\tau)d\tau = s_i(t_k)\Delta t$ becomes insignificant, and at this stage of processing, we have a signal that is:

$$s_p(t_k) \approx A_0^2 \exp\left(j2\pi K_{fm} s_i(t_k)\Delta t \right).$$

The phase of this signal represents the original sound, with a certain level of amplification and a very high sampling frequency. Therefore, in the next step, the angle() function in the program will extract the desired phrase. Most commonly, a sampling frequency of 48 kHz is used to convert analog sound into digital data. With these selected parameters, it is necessary to reduce the sampling frequency by 50 times (from 2.4 MHz to 48 kHz). This frequency reduction process is performed using a decimator, which is often combined with signal filtering.

Digital signal model of FM radio stations. We will simulate the output signal from an RTL-SDR receiver at a frequency of 2.4 MHz. To do this, we will use an audio file that has been recorded with a standard sampling rate of 48 kHz. This means that we will need to interpolate the audio data to match the frequency of the receiver. Since the audio samples in the file change slightly at the sampling rate of 48 kHz, we can use a simple linear approximation to convert the data. We will then insert the audio into the main loop for receiving information from the RTL-SDR receiver (see Listing 2).

Listing 2. Reading data from an audio file and linearly interpolating to the receiver frequency

```
model=1;
noise=1;
snr=-4; % SNR, dB
ks=0;
Fc = 0e3; % Receiver and transmitter carrier frequency
        % deviation (Hz)
dev = 250; % Frequency deviation with FM (Hz)
t_rtl = [0:rtlsdr_frmlen-1]'/rtlsdr_fs; % RTL-SDR time array
Krtl_s = rtlsdr_fs/audio_fs;
n_s=rtlsdr_frmlen/Krtl_s+1;  % number of audio samples
% per  interval rtlsdr_frmlen
dt_rtl=1/rtlsdr_fs; % Sample time for RTL-SDR
dt_s=1/audio_fs;  % Sample time for audio
t=0:dt_rtl:dt_s;
OMc=2*pi*Fc;
FIs=2*pi*dev;
Ki= 1525;
N1=rtlsdr_frmlen;
sigma=10^(-snr/20)/sqrt(2);
if  model == 0
  % Receive data from RTL-SDR:
    rtlsdr_data = step(obj_rtlsdr);
else
  % Read data from audio file:
    samples = [ks*n_s+1,(ks+1)*n_s];
    x = audioread('music3_mono48kHz.wav',samples);
    ks=ks+1;
  % Converting sound to quantization frequency rtlsdr_fs by
..% linear approximation of amplitudes
    for i=1:n_s-1
       dx=(x(i+1)-x(i))/dt_s;
       for j=1:Krtl_s
          x_rtl((i-1)*Krtl_s+j)=x(i)+dx*t(j);
       end
    end
end
```

As indicated in formula (1), in FM modulation, not the original sound signal is used, but rather its integral. In digital modeling, this integral is replaced with a cumulative sum, as shown in Listing 3.

Listing 3. Calculation of the cumulative sum

```
int_x = cumsum(x_rtl)/Ki;
rtlsdr_data = exp(1i*(OMc*t_rtl - FIs*int_x'));
if noise % Adding noise
  gaussnoise=(randn(N1,1)+1i*randn(N1,1))*sigma;
  rtlsdr_data = rtlsdr_data+gaussnoise;
end
```

The cumulative sum is calculated using the cumsum() function. The K_i coefficient is selected based on the similarity between the decoded and original sounds. When listening to the decoded audio, it matches the original sound closely. It is also possible to plot waveforms on a single graph (Figure 1, a). We can see that the waveform amplitudes of the original and decoded sounds are similar, indicating the correct choice of the K_i gain factor. We also notice that the decoded audio lags slightly behind the original in time. To plot waveforms it is necessary to adjust the time synchronization accordingly (Listing 4):

Listing 4. Synchronization of source and decoded sound when plotting their waveforms

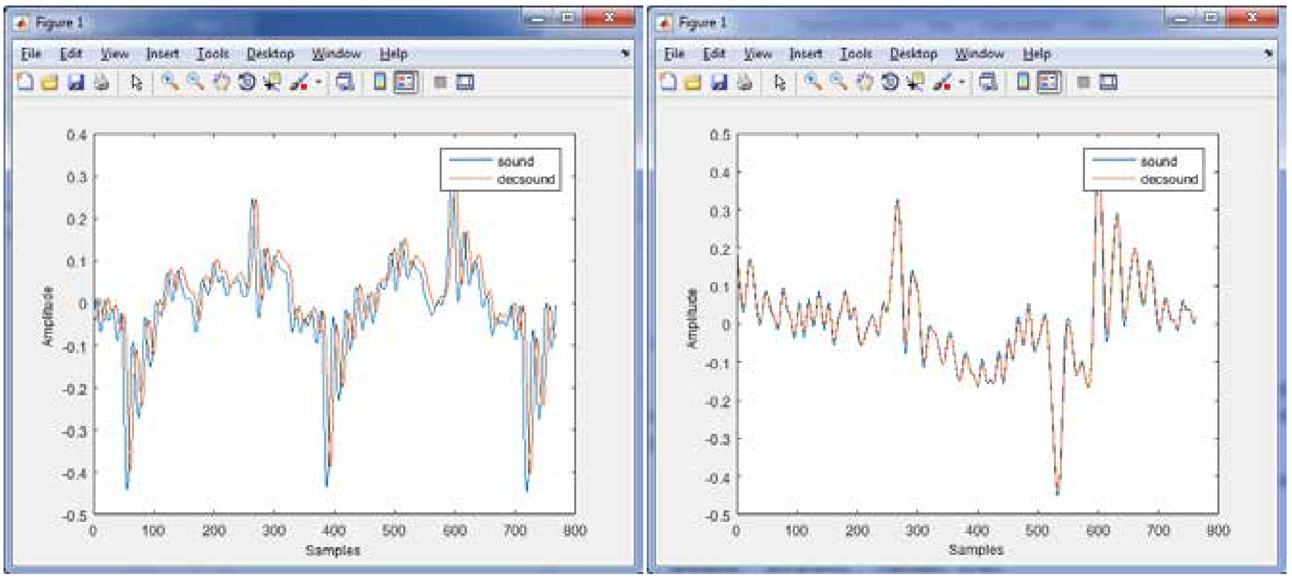
```
Nsd=5; % Shift of the demodulated sound in the model
    %relative to the original sound
if Fplot
  Nn=1; % Starting point for display
  dN=n_s-1-Nn; % number of points to display
  Nk=Nn+dN-Nsd;
  tgr=Nn:Nk;
  if model
  plot(tgr,x(Nn:Nk),tgr,data_deemph(Nn+Nsd:Nk+Nsd));
    legend('sound','decsound');
  end
  pause(1);
end
```

To accurately match the time shift, we will calculate the standard deviation of difference between the demodulated sound and the original sound (see Listing 5).

Listing 5. Calculation the standard deviation of difference between the demodulated and original sound

```
if model
  n1=length(data_deemph);
  D(m)=var(data_deemph(1+Nsd:n1)-x(1:n1-Nsd));
  m=m+1;
  if m == 500
    SKO=mean(D)^0.5;
    fprintf(Nsd=%d, Ki=%d, md=%.3f\n',Nsd,Ki,md*1e3);
    break;
  end
end
```

In each cycle of receiving a portion of information from the SDR-RTL receiver, the deviation variance is calculated. By averaging the deviation variance over a given number of portions, the standard deviation value is determined. When compensating for time shift, the decoded and original sounds are almost identical (Fig. 1, b).



a) b)

Fig. 1. Comparison of the original sound with the sound extracted from the FM signal model (a), their comparison after time synchronization (b)

Note that in reality, the time offset was first selected and then the gain Ki was selected using the RMS.

The effect of AWGN on sound demodulation. At a signal-to-noise ratio (SNR) of 6 dB, the decoded audio quality corresponds to that of the original sound, while interference becomes audible at an SNR of 4 dB. Even with an SNR of 0 dB, recorded music can be listened to, albeit with some background noise [6, p. 41].

Note that, in this case, the SNR at the receiver's output is measured over the entire 2.4 MHz bandwidth, resulting in small values. After signal processing, the SNR of the signal's peak in its 250 kHz bandwidth for real FM radio stations typically varies between 35 and 45 dB [7].

A disadvantage of the original algorithm is its lack of selectivity, which can be verified by examining the signals from real FM radio stations. For example, at the frequency of "Nashe Radio", 102.8 MHz, the stronger signal of "Kiss FM" operating at 102.1 MHz can be heard.

Improvement of the processing algorithm. Firstly, the signal from the receiver's output must be filtered within the range used for FM signals (250 kHz), and accordingly, the quantization frequency should be reduced to 240 kHz. Note that the program under consideration provides an option to read amplitude values in IQ-channels from a file that was recorded with a sampling frequency of 240 kHz. Let's consider three options for low-pass FIR filters: a filter with Hamming and Kaiser Window, and a filter synthesized using the Parks-McClellan algorithm. As a starting point, we can consider a relatively simple Hamming filter with the parameters shown in Listing 6.

Listing 6. Creating a FIR Filter with a Hamming Window

```
n = 45;     % Hamming filter order
Df=125e3;   % Half bandwidth of FM signals
W=Df/(rtlsdr_fs/2);
numIn=fir1(n,W); % Hamming window
```

The Kaiser window filter is more versatile and is defined as follows (Listing 7).

Listing 7. Creating a FIR Filter with a Kaiser Window

```
Df=125e3;   % Half bandwidth of FM signals
fcuts = [Df Df+Df/5]/(rtlsdr_fs/2);
mags = [1 0];
devs = [0.05 0.03];
[n,Wn,beta,ftype] = kaiserord(fcuts,mags,devs);
numIn = fir1(n,Wn,ftype,kaiser(n+1,beta),'noscale');
```

The filter order is not explicitly specified, but it is calculated based on the requirements for the desired filter frequency response. These requirements include the passband cutoff frequency at which attenuation must begin, as well as the stopband cutoff frequency at which maximum attenuation should be achieved. Frequencies should be normalized to the Nyquist frequency in order to ensure accurate results. The mags is band amplitude vector. In this case, the filter has been set to a specific stopband cutoff frequency of 25 kHz in order to achieve maximum attenuation. Dev is a vector that specifies the maximum allowable deviation between the frequency response of the output filter and its band amplitude, for each band. For the given parameters, the filter order is 215.

The Parks-McClellan filter is used with the following parameters (Listing 8). The filter order is 300 and stopband cutoff frequency is 25 kHz as in the Kaiser filter.

Listing 8. Creating the Parks-McClellan filter

```
numIn = firpm(300,[0,Df,Df+Df/5,rtlsdr_fs/2]/(rtlsdr_fs/2),...
        [1 1 0 0], [1 1], 20);
```

The frequency response of the proposed filters is shown in Fig. 2.

Kaiser and Parks-McClellan filters, with a large order, provide an almost vertical cutoff and differ slightly in the level of side lobes. For the Kaiser filter, the side lobes start at –40 dB and gradually drop to –70 dB, while the Parks-McClellan filter immediately reaches a constant level of –58 dB. The Hamming filter has a significantly lower order, with almost the same level of side lobes, but a much flatter main lobe (losses at the cutoff frequency are 6 dB and the side lobes begin at a frequency of ~220 kHz).

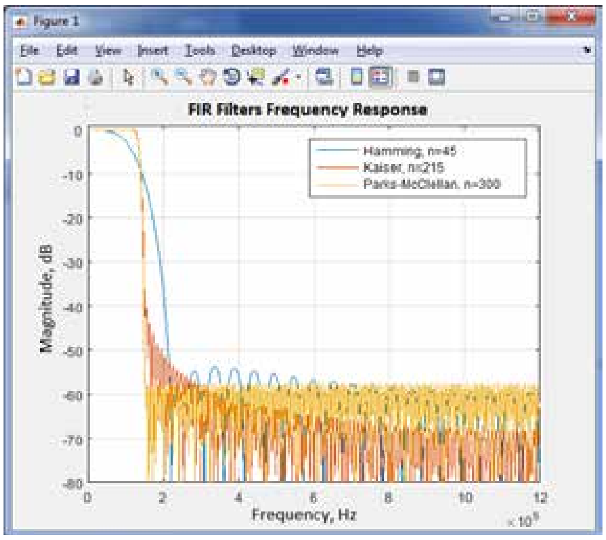
Next, we'll use an FIR decimator with the parameters presented in Listing 9.

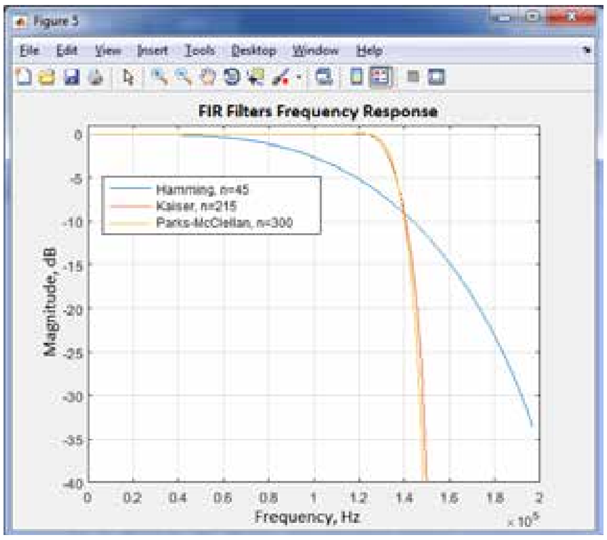
Listing 9. Implementation of an FIR decimator

```
obj_decmtr = dsp.FIRDecimator(...
    'DecimationFactor', 5,'Numerator',...
    firpm(100,[0,15e3,20e3,(240e3/2)]/(240e3/2),...
    [1 1 0 0], [1 1], 20));
```

The order of the Parks-McClellan filter has been reduced from 350 to 100. Full attenuation of high frequencies begins at 20 kHz, instead of 48 kHz, and the decimation factor is 240/48 = 5. The frequency response of this filter is shown in Figure 3.

As it can be seem from the graph, starting from 20 kHz, attenuation of about 40 dB is provided. Now, let's take a look at how our improvements have impacted the signal reception quality for real FM stations.





a) b)

Fig. 2. Frequency response of Hamming, Kaiser and Parks-McClellan filters in the 0–1.2 MHz band (a), in the 0–200 kHz band (b)

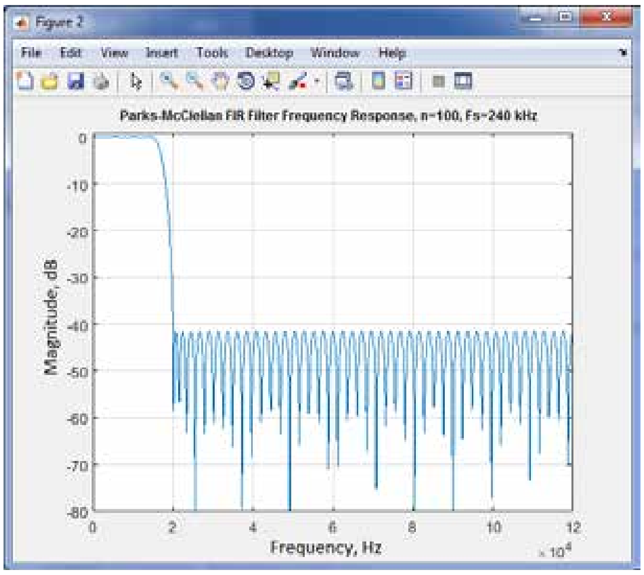


Fig. 3. Frequency response of the Parkes-McClellan filter of the 100th order in the 0–120 kHz band

Table 1

Evaluation of the sound quality of FM radio stations

FM Station No.	Frequency of FM station? MHz	Processing on Fs=2.4 MHz (initial algorithm)	FIR decimation to Fs=240 kHz (window)		
			Hamming	Kaiser	Parks–McClellan
1	99.2	3	10	10	10
2	102.1	9	10	10	10
3	102.8	0	7	7	7
4	103.3	0	1	1	1
5	104.1	0	10	10	10
6	105.1	0	9	9	9
7	106.4	0	10	10	10
8	107.8	5	10	10	10

The experimental results are presented in Table 1. We will use the sound quality provided by the SDRSharp program as a reference, which we will assign a value of 10.

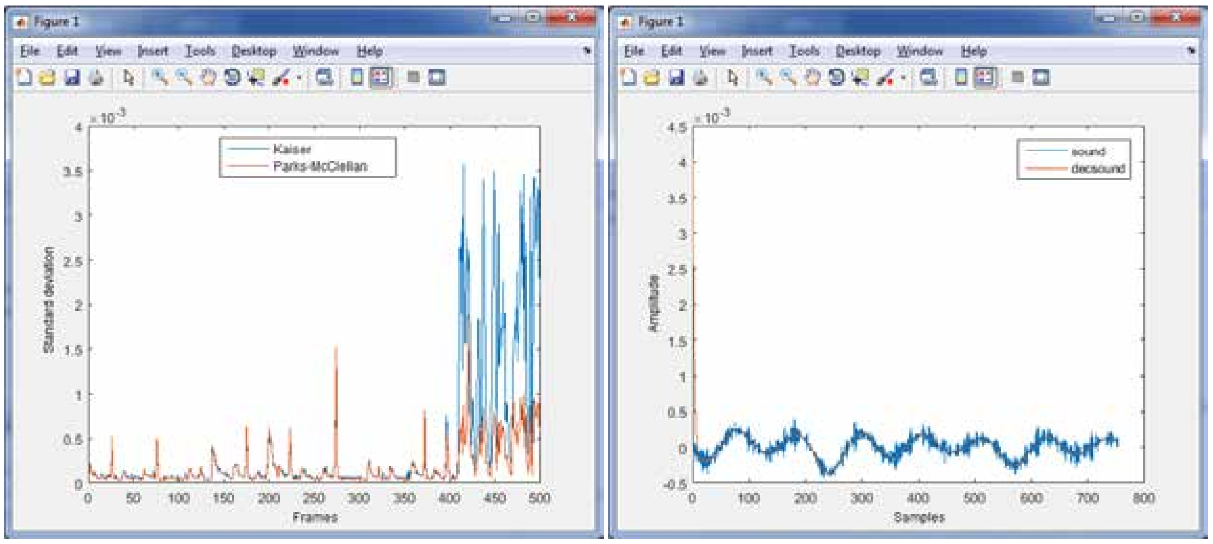
The new algorithm has significantly improved the listening experience of FM radio stations. In the original version, one radio station could be comfortably listened to, while two others were difficult to hear. However, after pre-filtering and reducing the quantization frequency from 2.4 MHz to 240 kHz, good audibility was achieved on six radio stations, and one was listened to with satisfaction, although one was still not practically heard. It is clear that it is not possible to hear the difference between the various FIR filters by ear.

For a more detailed study of the influence of various processing elements, consider the signal model with two FM signals separated in frequency by 700 kHz. This model simulates the situation with “Kiss FM” and “Nashe Radio”, when signal reception needs to be provided from one radio station under the background of another powerful radio station located nearby in frequency band. We will additively mix both signals with each other and with the AWGN. Let’s set the amplitude of the first signal equal to unity, and set the amplitude of the second signal with parameter A1. The resulting SNR over the entire reception band for the second signal will be calculated as follows: SNR1 = SNR+20*lg(A1). For analysis, we will use melodies of various genres, which provide a wider use of the sound band compared to speech.

If we take the inverse standard deviation as a criterion for filter quality and take the Parks-McClellan filter as 100 %, we get a more convenient table for analysis (Table 2).

Thus, the Parks-McClellan filter proved to be the most effective in terms of SNR, which ensures comfortable listening to musical content (0–20 dB). The Hamming filter, with a relatively low order, is inferior to it by 5 %, and the Kaiser filter is inferior by 7 %. For weaker signals, the Hamming filter proves to be the most efficient. Repeating the study with a different musical composition resulted in an even stronger emphasis on the benefits of the Parks-McClellan filter. To visually represent the results, we present standard deviation plot for the first 500 processed frames of the received data (Fig. 4a).

In conclusion, we present a waveform of the original sound, in which oscillation above 48 kHz are



a) **b)**

Fig. 4. Standard deviation of the difference of the demodulated sound and the original one for the Kaiser and Parks-McClellan filters (a), elimination of high-frequency noise with improved processing (b)

Table 2

Numerical analysis of sound quality of FM radio stations

SNR, dB	SNR1, dB	Recording 1			Recording 2		
		Filter Type			Filter Type		
		Hamming	Kaiser	Parks–McClellan	Hamming	Kaiser	Parks–McClellan
20	40	95 %	93 %	100 %	68 %	65 %	100 %
6	26	95 %	93 %	100 %	69 %	65 %	100 %
0	20	95 %	93 %	100 %	71 %	68 %	100 %
–4	16	111 %	97 %	100 %	104 %	88 %	100 %

observed, which are absolutely inaccessible to human hearing (Fig. 4, b).

Perhaps such ultrasounds are produced by some musical instruments, perhaps these are just failures. The graph (Fig. 4, b) clearly shows how the proposed algorithm for processing FM signals completely eliminates them.

Conclusions. As a result of the research, it was found that in order to increase the selectivity of the incoherent discriminant algorithm for detecting the mono signal of FM radio stations, it is necessary to use an additional low-frequency FIR filter before the FIR decimator. When comparing three types of FIR filters (Hamming, Kaiser and Parks-McClellan), the Parks-McLellan filter showed the best result in SNR range (0–20 dB). Studies of the improved algorithm were carried out for both model and real FM signals.

References:

1. S.O. Ugwuanyi, M.A. Ahaneku. Radio frequency and channel investigation using software defined radio in MATLAB And simulink environment. *IJOTECH.* Vol. 37. № 4 (2018). P. 1049–1057. DOI: 10.4314/njt.v37i4.26.

2. Robert W. Stewart, Kenneth W. Barlee, Dale S.W. Atkinson, Louise H. Crockett. Software Defined Radio using MATLAB & Simulink and the RTL-SDR / University of Strathclyde, Glasgow, Scotland, UK, 2015, 674 p.

3. Example of FM Broadcast Receiver using MATLAB. URL: https://www.mathworks.com/help/comm/ug/fm-broadcast-receiver.html (дата звернення: 04.05.24).

4. Generalized FM demodulation block with deemphasis and audio filtering. URL: https://wiki.gnuradio.org/index.php/FM_Demod (дата звернення: 04.05.24)

5. Der, Lawrence. "Frequency Modulation (FM) Tutorial." Silicon Laboratories Inc. URL: https://cdn.weka-fachmedien.de/whitepaper/files/005_fmtutorial.pdf (дата звернення: 04.05.24).

6. Іхсанов Ш. М., Рябенький В. М., Дьяконов О.С. Дослідження сигналів реальних інформаційних систем з використанням приймачів RTL-SDR : навч. посіб. Миколаїв, 2019. 184 с.

Іхсанов Ш.М., Дьяконов О.С. УДОСКОНАЛЕННЯ НЕКОГЕРЕНТНОГО ДИСКРИМІНАНТНОГО АЛГОРИТМУ ОБРОБКИ FM-СИГНАЛІВ ДЛЯ ПРОГРАМНО-ВИЗНАЧЕНОЇ РАДІОСИСТЕМИ ТА ЙОГО РЕАЛІЗАЦІЯ У MATLAB

У роботі аналізуються недоліки некогерентного дискримінантного алгоритму виділення моно сигналу FM-радіостанцій. Показано, що MATLAB-реалізація існуючого алгоритму для приймача RTL-SDR має низьку вибірковість, що призводить до ефекту накладення звуку сусідніх FM-радіостанцій у вигляді перешкоди.

Для усунення даного недоліку був розроблений вдосконалений алгоритм, що полягає у використанні низькочастотного фільтра зі скінченною імпульсною характеристикою (СІХ) в смузі FM-сигналів 250 кГц перед СІХ-дециматором. Дослідження проводилися для СІХ-фільтрів трьох типів: Хеммінга, Кайзера і Паркс-Макклеллана. Фільтри Кайзера, Паркс-Макклеллана з великим порядком забезпечують практично вертикальний зріз і трохи відрізняються рівнем бічних пелюсток. У фільтра Кайзера вони починаються на рівні −40 дБ і поступово опускаються до −70 дБ, в той час як у фільтра Паркс-Макклеллана вони відразу виходять на постійний рівень −58 дБ. У фільтра Хеммінга з істотно меншим порядком, рівень бічних пелюсток практично такий же, але значно більш пологий основний пелюстка (втрати на частоті зрізу досягають 6 дБ, а бічні пелюстки починаються з частоти ~220 кГц). Після КІХ-дециматора на основі алгоритму Паркс-Макклеллана починаючи 20 кГц забезпечується придушення близько 40 дБ.

Для більш детального вивчення впливу різних елементів запропонованої обробки використовувалася модель двох FM-сигналів з різницею в частотах в 700 кГц, що імітує ситуацію, коли прийом сигналу потрібно забезпечити з однієї радіостанції на тлі іншої потужної радіостанції, розташованої поруч по частоті. Перевищення сигналу, що заважає, над сигналом, що прослуховується, в експериментах прийнято рівним 20 дБ. Як критерій для порівняння СІХ-фільтрів використовувалася величина зворотна середньоквадратичному відхиленню вихідного інформаційного і відфільтрованого сигналів. Показано, що у всьому діапазоні відношення сигнал/перешкода (0–20 дБ), фільтр Паркс-Макклеллана виявився кращим. Фільтр Хеммінга з відносно невеликим порядком поступається йому на 5 %, а фільтр Кайзера – на 7 %. Наведено реалізацію запропонованих алгоритмів мовою MATLAB.

Key words: *FM-мовлення, програмно-визначена радіосистема (SDR), цифрова обробка сигналу, FM-сигнал, СІХ-фільтр.*