FIR Implementation Using Matlab

Executed by:Kevin Anum
Mailbox:108
DSP Lab
Natural Science Laboratory, Jacobs University Bremen

April 25, 2017

Introduction

Objective

In this lab we designed and tested an FIR filter block in MATLAB using buffer-oriented signal processing.

Background

What is a FIR filter ?

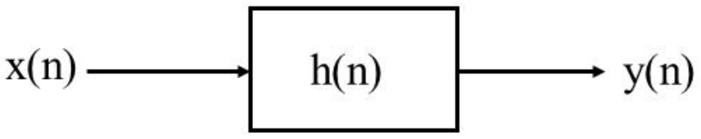


Figure 1: From the figure an example of an FIR filter will be when $y(n) = \frac{1}{2}^{n} u(n)$where n is from 0 to 10.If n to not limited then $y(n)$ becomes an IIR (Infinite Impulse Response)filter.

An FIR (Finite Impulse Response)filter is a filter that has a finite impulse response (finite output) to it settles to zero in finite time.

An important property of an FIR filter is that linear phase can be designed easily by making the coefficient sequence symmetric.(therefor delaying the input signal without distorting its phase can be achieved easily.There are many was to design an FIR filter this includes:

- Window design method
- Frequency Sampling method
- Weighted least squares design

In this lab we will be using the Window design method.

Window design method

Window design method is when you cut an IIR to make an FIR by limiting the response with a window.

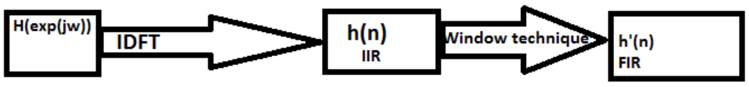


Figure 2: The figure shows how you can convert the IIR into an FIR using the window method.In the diagram IDFT stands for Inverse Discrete Fourier transform.

There are three window methods.There are:

- Rectangular Window $W_r(n) = 1$ where $0 \leqslant n \leqslant m-1$ where m is the filter order.
- Hamming Window $W_h(n) = 0.54 + 0.46\cos(\frac{2\pi n}{m-1})$ where $0 \leqslant n \leqslant m-1$ where m is the filter order.
- Hanning Window $W_{hn}(n) = 0.5 + 0.5\cos(\frac{2\pi n}{m-1})$ where $0 \leqslant n \leqslant m-1$where m is the filter order.

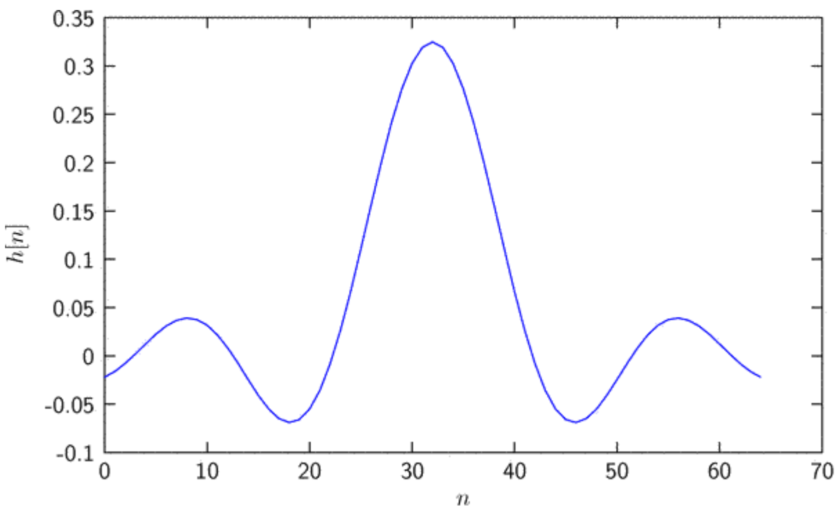


Figure 3: The figure shows the effect of using the rectangular window.

In this experiment we do not use a rectangular window,because the rectangular window tends to have high side lobes, meaning frequencies in the stop band may not be attenuated as much as we want.

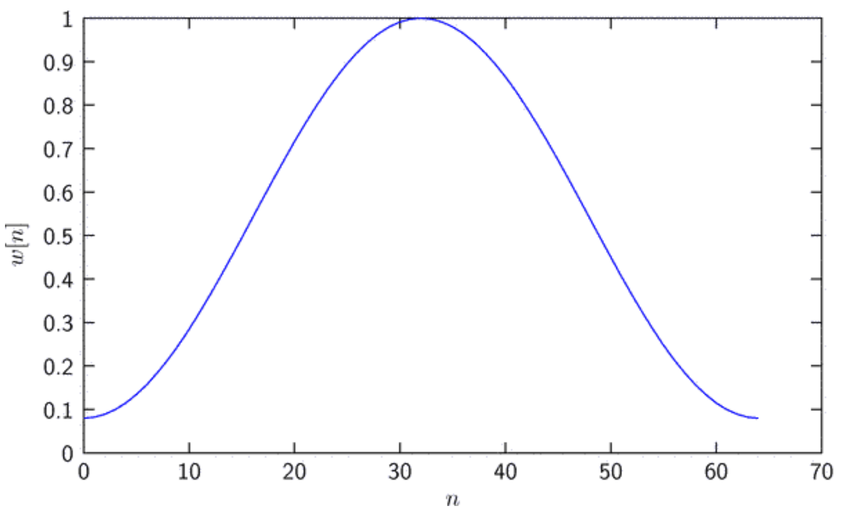


Figure 4: The figure shows the effect of using the hamming window.

In this experiment we use the hamming window because the side slopes are not to high.

Implementing an FIR in MATLAB

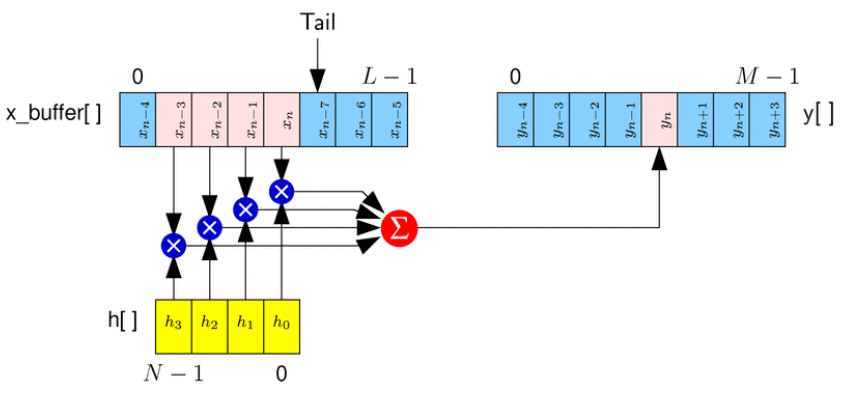


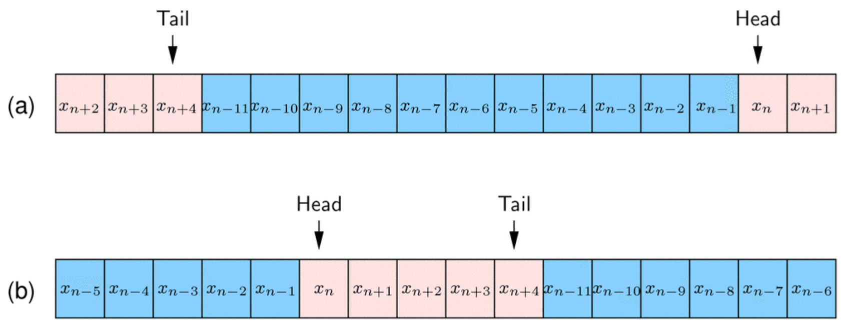
Figure 5: The figure shows how the convolution is performed.

In MATLAB we go through the same process as Figure(2).In finding y(n) we perform a convolution with h(n) and x(n).

Coding this as a buffer-oriented code the following takes place:

1. A buffer of N samples is received.
2. Samples are moved into the circular buffer
3. The samples are filtered to generate N output samples
4. The N samples are copied into the output buffer

Circular Buffer



Tail
Head
(a)
x_{n+2} x_{n+3} x_{n+4} x_{n-11} x_{n-10} x_{n-9} x_{n-8} x_{n-7} x_{n-6} x_{n-5} x_{n-4} x_{n-3} x_{n-2} x_{n-1} x_n x_{n+1}
Head
Tail
(b)
x_{n-5} x_{n-4} x_{n-3} x_{n-2} x_{n-1} x_n x_{n+1} x_{n+2} x_{n+3} x_{n+4} x_{n-11} x_{n-10} x_{n-9} x_{n-8} x_{n-7} x_{n-6}

Figure 6: The figure shows the representation of a circular buffer in memory.In order to make it circular we wrap the ends of the buffer.This is the same buffer we use in the delay.

A circular buffer is a memory allocation scheme where memory is reused (reclaimed) when an index, incremented modulo the buffer size, writes over a previously used location. A circular buffer makes a bounded queue when separate indices are used for inserting and removing data.("Wikipedia",2017).

Why MATLAB ?

MATLAB serve as a valuable first step for prototyping and testing an algorithm before coding it for the DSP. Many simple errors and problems can be flushed out using the user-friendly MATLAB environment in much less time than would be required on the DSP.**(Fu,2015)**

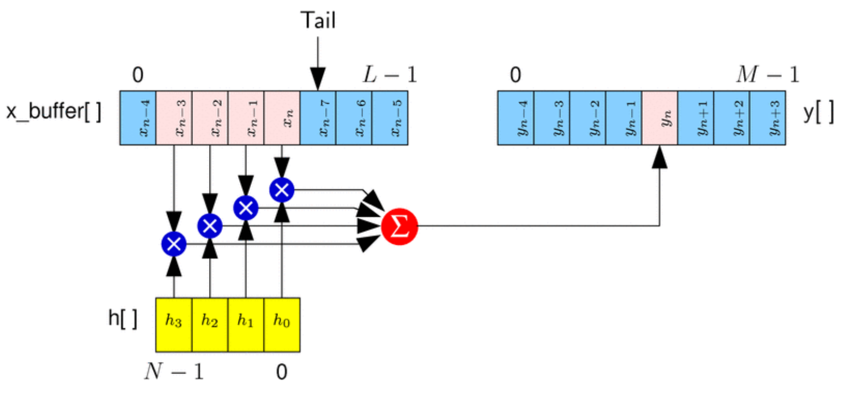


Figure 7: The figure shows how your FIR filter uses a circular buffer to implement the FIR equation.

LAB Write up

Problem 2

The reason we use a circular buffer is because we can simulate a continuous stream of samples with finite-sized buffers .

Problem 3

fir_ int .m

```
 function [state] = fir_init(h, Ns);

% [state] = fir_init(h, Ns);
%
% Creates a new FIR filter.
%
% Inputs:
% h Filter taps
% Ns Number of samples processed per block
% Outputs:
% state Initial state

%% 1. Save parameters
state.h = h;
state.Ns = Ns;

%% 2. Create state variables
```

```
state.M = 2^(ceil(log2(state.Ns+1)));
state.Mmask = state.M - 1;
% Temporary storage for circular buffer
state.buff = zeros(state.M , 1);
% Set initial tail pointer and temp pointer (see pseudocode)
state.n_t = 0;
state.ptr = 0;
```

fir.m

```
 function [state_out, y] = fir(state_in, x)

% [state_out, y] = fir(state_in, x);
%
% Executes the FIR block.
%
% Inputs:
% state_in Input state
% x Samples to process
% Outputs:
% state_out Output state
% y Processed samples

% Get state
s = state_in;
% Move samples into tail of buffer

for ii = 0: s.Ns - 1
  % Store a sample
  s.buff(s.n_t+1) = x(ii+1);
  % Increment head index (circular)
  s.n_t = bitand(s.n_t+1, s.Mmask);
  s.ptr = bitand(s.n_t+s.Mmask, s.Mmask);
  sum = 0.0;
  for j = 0:length(h)-1
    sum = sum + s.buff(s.ptr+1)*s.h(j+1);
    s.ptr = bitand(s.ptr+s.Mmask, s.Mmask);
  end
   y(ii+1) = sum;
```

```
end
% Filter samples and move into output

% Return updated state
state_out = s;
```

testfir.m

```
 % test_fir.m
%
% Script to test the FIR filter.

% Global parameters
Nb = 100; % Number of buffers
Ns = 128; % Samples in each buffer

% Generate filter coefficients
p.beta = 0.5;
p.fs = 0.1;
p.root = 0; % 0=rc 1=root rc
M = 512;
[h f H Hi] = win_method('rc_filt', p, 0.2, 1, M, 0);

% Generate some random samples.
x = randn(Ns*Nb, 1);

% Type of simulation
%stype = 0; % Do simple convolution
stype = 1; % DSP-like filter

if stype==0,
  y = conv(x, h);
elseif stype==1,
  % Simulate realistic DSP filter
  state_fir1 = fir_init(h,Ns);
  % Reshape into buffers
  xb = reshape(x, Ns, Nb);
  % Output samples
  yb = zeros(Ns, Nb);
  % Process each buffer
  for bi=1:Nb
    [state_fir1 yb(:,bi)] = fir(state_fir1, xb(:,bi));
  end
  % Convert individual buffers back into a contiguous signal.
  y = reshape(yb, Ns*Nb, 1);
else
  error('Invalid simulation type.');
end

% Compute approximate transfer function using PSD
Npsd = 200; % Blocksize (# of freq) for PSD
[Y1 f1] = psd1(y, Npsd);
[X1 f1] = psd1(x, Npsd);
grid on;
plot(f1, abs(sqrt(Y1./X1)), f, abs(H));
legend('Simulated Response', 'Ideal Response');
xlim([0 0.2]);
hold on;
% Global parameters
Nb = 100; % Number of buffers
Ns = 128; % Samples in each buffer

% Generate filter coefficients
p.beta = 0.5;
p.fs = 0.1;
p.root = 0; % 0=rc 1=root rc
M = 256;
[h1 f H Hi] = win_method('rc_filt', p, 0.2, 1, M, 0);

% Generate some random samples.
x1 = randn(Ns*Nb, 1);

% Type of simulation
%stype = 0; % Do simple convolution
stype = 1; % DSP-like filter

if stype==0,
  y1 = conv(x, h);
elseif stype==1,
  % Simulate realistic DSP filter
  state_fir1 = fir_init(h,Ns);
  % Reshape into buffers
  xb = reshape(x, Ns, Nb);
  % Output samples
  yb = zeros(Ns, Nb);
  % Process each buffer
  for bi=1:Nb
    [state_fir1 yb1(:,bi)] = fir(state_fir1, xb1(:,bi));
  end
  % Convert individual buffers back into a contiguous signal.
  y1 = reshape(yb, Ns*Nb, 1);
else
  error('Invalid simulation type.');
end

% Compute approximate transfer function using PSD
Npsd = 200; % Blocksize (# of freq) for PSD
[Y1 f2] = psd1(y1, Npsd);
[X1 f2] = psd1(x1, Npsd);
grid on;
plot(f2, abs(sqrt(Y2./X2)));
legend('Simulated Response', 'Ideal Response');
xlim([0 0.2]);
```

Problem 4

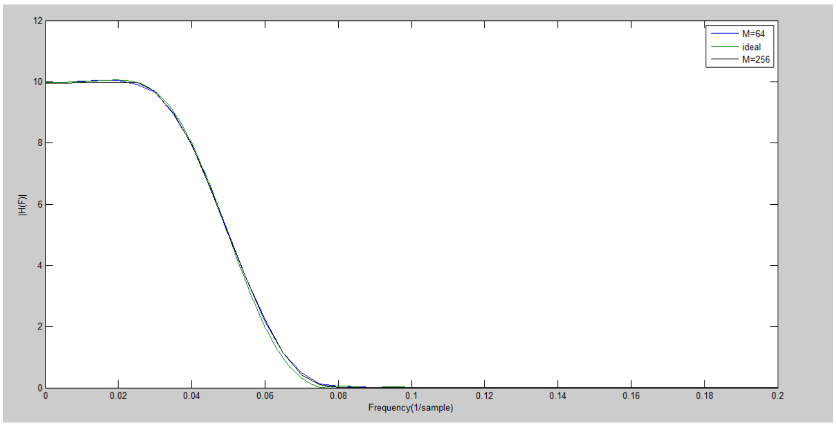


Figure 8: The figure shows the ideal response of your filter compared to the simulated response of the filter.

The large the taps the more the they are closer to the ideal response. If we make M small, the DSP code will run faster, but unfortunately the deviation from the specified filter response will be higher; the frequency domain response is “ smeared out ”.

Problem 5

It was challenging knowing how to initialize the to pointers but with the help of the TA’s I was able to figure it out.

Conclusion

In this lab we got to see that FIR implementation in MATLAB is a suitable tool to test an FIR filter code.We could see that as we increased the tabs the simulated FIR was close to the ideal curve.

Reference

```
 "Circular Buffer".
 En.wikipedia.org. N.p., 2017. Web. 16 May 2017.

 Basics, FIR.
 "FIR Filter Basics - Dspguru". dspGuru. N.p., 2017. Web. 16 May 2017.

 "Design FIR Filter Using Window Method - Complete Basics".
 YouTube. N.p., 2017. Web. 16 May 2017.

"Delay And FIR (Matlab Part)  | DSP And Communication Lab".
Dsp-fhu.user.jacobs-university.de. N.p., 2017. Web. 16 May 2017.
```