**به نام خدا**

**گزارش کار**

**شبیه­سازی استفاده از فیلتر FIR در پردازش سیگنال صوتی**

در این گزارش تاثیر استفاده از فیلتر FIR در پردازش سیگنال صوتی دریافتی از میکروفون بررسی می­شود. برای تولید ضرایب فیلتر و اعمال آن به سیگنال دریافتی از نرم­افزارMATLAB استفاده شده است. بعد از وارد کردن سیگنال به پروژه متلب آن را از فیلترهای مختلف عبور داده و خروجی فیلترها باهم مقایسه خواهد شد.

برای شروع کار باید ابتدا سیگنال صوتی توسط متلب خوانده شود. فرکانس آن تعیین گردد و ضرایب آن استخراج شود. لذا از دستورات زیر استفاده می شود.

[y, Fs] = audioread('record1234.wav');

audioSignal = y(:,1);

از آنجایی که تابع audioSignal یک تابع ستونی است با دستور inputSignal=audioSignal' به یک تابع سطری تبدیل می شود تا محاسبات بعدی ساده­تر انجام شود.

فیلتر FIR با استفاده از تولباکس filterDesigner نرم­افزار متلب طراحی می­شود. در طراحی فیلتر نرخ نمونه­برداری برابر Fs=44100، پهنای باند برابر 4000 و مرتبه فیلتر برابر 10 در نظر گرفته می­شود. برای مقایسه خروجی فیلترها از پنجره های hann، Kaiser و balckman استفاده می­شود. در واقع سه فیلتر FIR با مشخصات یکسان و پنجره­های مختلف برای پردازش سیگنال استفاده شده است. توابع hannFIR، kaiserFIR و balackmanFIR با این روش تولید شده­اند. بعد از طراحی فیلترها ضرایب آنها برای استفاده در شبیه­سازی به کمک دستورات زیر استخراج می­شود.

% Design hann window

hannCoeffs = coeffs(hannFIR);

hannFilter = hannCoeffs.Numerator;

% Design kaiser window

kaiserCoeffs = coeffs(kaiserFIR);

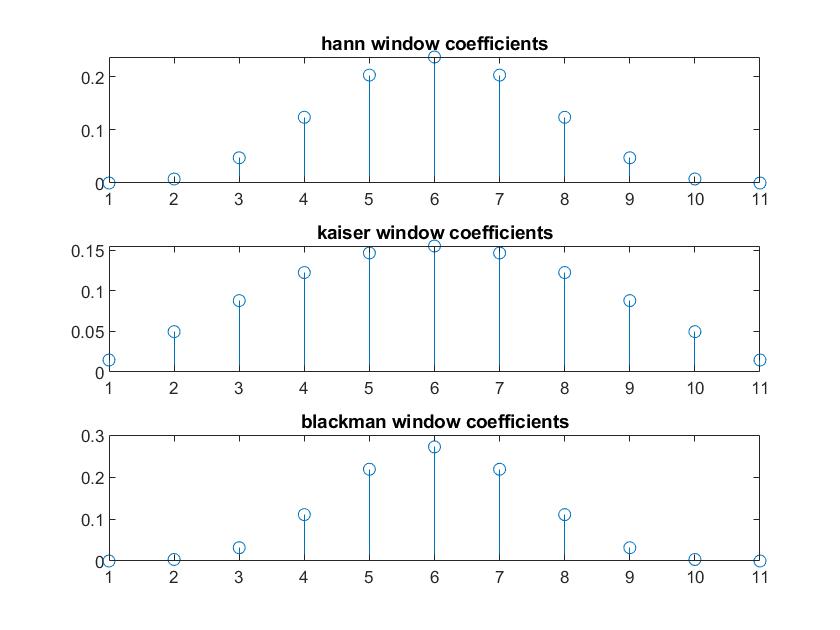
kaiserFilter = kaiserCoeffs.Numerator;

% Design blackman window

blackmanCoeffs = coeffs(blackmanFIR);

blackmanFilter = blackmanCoeffs.Numerator;

شکل 1 ضرایب فیلترهای تولید شده را نشان می­دهد.



شکل 1- ضرایب فیلترهای تولید شده

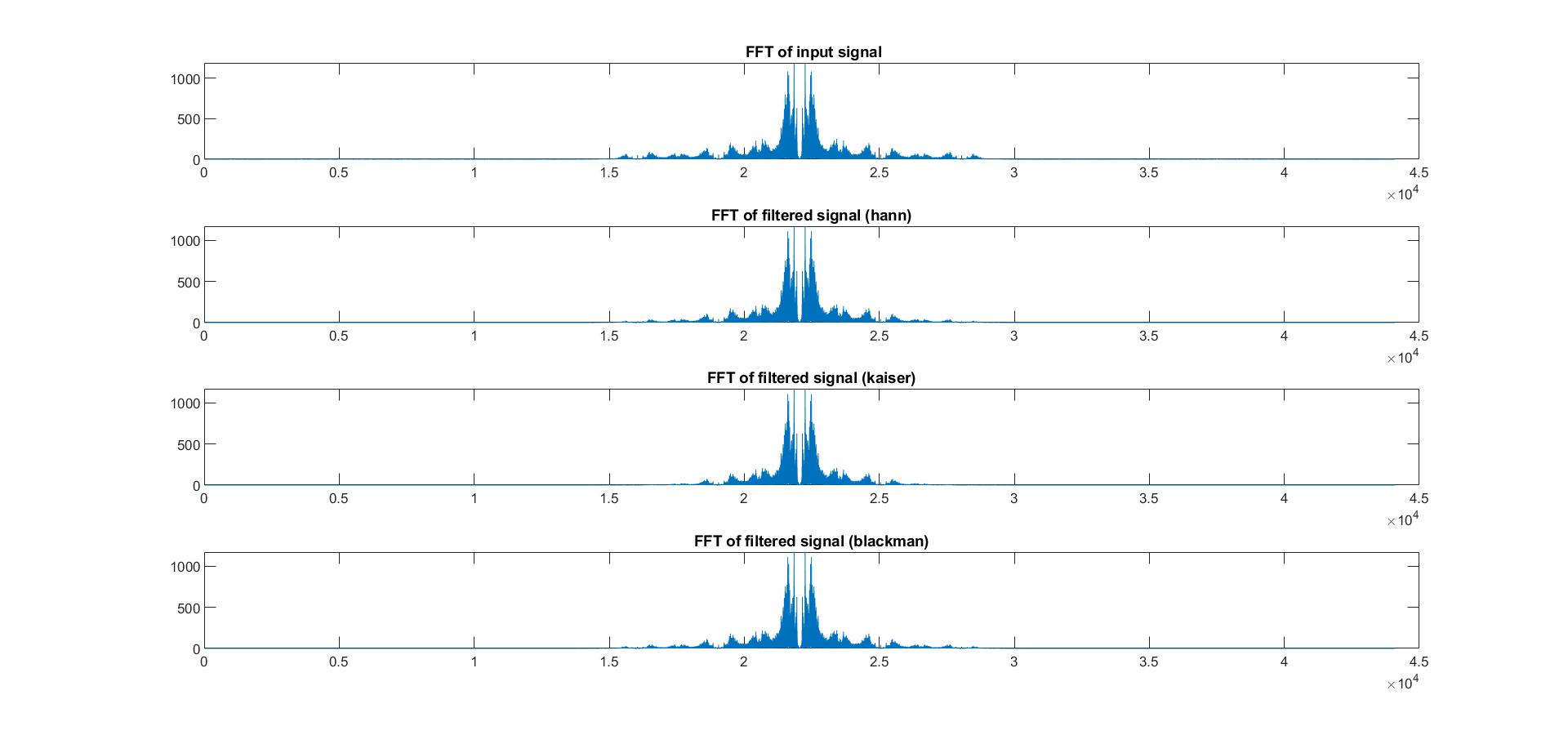
برای فیلتر کردن سیگنال ورودی جهت حذف فرکانسهای بالا، کانولوشن سیگنال inputSignal با توابع فیلتر تولید شده­ی hannFilter، kaiserFilter و blackmanFilter با استفاده از تابع myConv1 محاسبه شد. دلیل این کار این است که خروجی فیلتر، کانولوشن سیگنال ورودی با پاسخ ضربه فیلتر است. تابع myConv1 عمل کانولوشن را انجام می­دهد.

hannOutput = myConv1(inputSignal,hannFilter);

kaiserOutput = myConv1(inputSignal,kaiserFilter);

blackmanOutput = myConv1(inputSignal,blackmanFilter);

شکل 2 طیف پاسخ فرکانسی خروجی­های به دست آمده را به همراه نمایش سیگنال ورودی در حوزه فرکانس را نشان می­دهد. محور افقی، فرکانس و محور عمودی اندازه تابع در حوزه فرکانس را نشان می­دهد.



شکل 2- طیف فرکانسی سیگنالهای ورودی و خروجی

برای تولید سیگنال خروجی به صورت یک فایل صوتی از دستور audiowrite استفاده شد. از آنجایی که از سه فیلتر در شبیه­سازی استفاده شده در نتیجه از دستور audiowrite سه بار استفاده شده است.

filename = 'output\_hann.wav';

audiowrite(filename,hannOutput,Fs);

filename = 'output\_kaiser.wav';

audiowrite(filename,kaiserOutput,Fs);

filename = 'output\_blackman.wav';

audiowrite(filename,blackmanOutput,Fs);

برای اینکه مشخص شود کدام یک از خروجی­های به دست آمده عملکرد بهتری در کاهش نویز ورودی داشته است انرژی سیگنال­های خروجی باهم مقایسه شده است. هر کدام از خروجی­ها انرژی کمتری داشته باشد حذف نویز بیشتری در آن صورت گرفته است. برای به دست آوردن این مقدار برای هر خروجی از ضرب داخلی سیگنال در خودش استفاده شده است. بنابراین:

eng\_in = inputSignal\*inputSignal';

Eng\_out= hannOutput\*hannOutput';

Eng\_kaiserOut = kaiserOutput\*kaiserOutput';

Eng\_blackmanOut = blackmanOutput\*blackmanOutput';

نتایج به دست آمده به قرار زیر است:

sampling frequency =44100

cutoff frequency = 4000

order = 10

Energy of input signal = 1919.0228

Energy of output signal by hannFilter = 1841.7349

Energy of output signal by kaiser window= 1787.0603

Energy of output signal by blackman window= 1787.0603

برای پخش فایل صوتی از دستور sound استفاده می شود.

sound(inputSignal,Fs)

pause(27)

sound(kaiserOutput,Fs)

**تابع myConv1**

تابع myConv1 عملیات کانولوشن را روی دو سیگنال انجام می­دهد. این تابع، خروجی مربوطه را بدین صورت به دست می­آورد. ابتدا سیگنال­های تعریف شده وارد تابع می­شوند. طول هر سیگنال با دستور length محاسبه می­شود. طول خروجی برابر طول سیگنالهای ورودی منهای یک خواهد بود. یعنی:

Np1 = length(x1);

Mp2 = length(x2);

Tconv = Np1+Mp2-1;

یکی از سیگنالها به عنوان سیگنالی که قرار است از بی­نهایت شروع به حرکت کرده و تا بی­نهایت به حرکت ادامه دهد انتخاب می­شود. در این تابع سیگنال اول به عنوان سیگنال متحرک در نظر گرفته می­شود. با صفرگذاری در انتهای سیگنال اول به اندازه Tconv-Np1 طول آن افزایش می­یابد. این کار تاثیری در ماهیت سیگنال ایجاد نمی­کند ولی برای انجام عملیات محاسباتی لازم است. در نتیجه:

p01 = [x1,zeros(1,Tconv-Np1)];

سپس ضریب اولِ سیگنال دوم در تمام ضرایب سیگنال اول صفرگذاری شده ضرب می­شود. خروجی به دست آمده تحت عنوان تابع z ذخیره می­شود. یک واحد به شمارنده اضافه می­شود. سیگنال اول صفرگذاری شده یک واحد به سمت راست شیفت داده می­شود و یک صفر به ابتدای آن اضافه می­گردد. در واقع یک صفر از انتهای سیگنال برداشته و به ابتدای آن اضافه می­شود.

ضریب دوم سیگنال دوم در تمام ضرایب سیگنال شیفت یافته ضرب می­شود و نتیجه به دست آمده با z جمع می­شود و تحت عنوان z ذخیره می­گردد. سیگنال اول شیفت می­یابد و با ضریب سوم سیگنال دوم ضرب می­شود و نتیجه به دست آمده با نتایج قبلی جمع می­گردد و یک واحد دیگر به شمارنده اضافه می­شود.

این عملیات تا زمانی انجام می­شود که همه ضرایب سیگنال دوم به تمام ضرایب سیگنال ضرب شده و نتایج به دست آمده باهم جمع شود. به عبارت دیگر تا زمانی که شمارنده کمتر از طول سیگنال دوم است این عملیات ادامه می­یابد.

در دستورات زیر counter برابر شمارنده، k11 برابر سیگنال شیفت یافته در مرحله قبلی، w11 سیگنال شیفت یافته در مرحله فعلی، z1 حاصلضرب ضریب سیگنال دوم در تابع شیفت یافته سیگنال اول و z جمعِ حاصلضربها می باشد.

counter = 0;

z = zeros(1,Tconv);

k11 = p01;

counter = 0;

while (counter<Mp2)

z1 = x2(counter+1)\*k11;

w11 = [0,k11(1:end-1)];

k11 = w11;

z = z+z1;

counter = counter+1;

end

convolution = z;

خروجی نهایی (convolution) کانولوشن سیگنال اول و دوم خواهد بود.