Exercise 9

Ultrasound Signal Processing

TTK 4165 MEDICAL SIGNAL PROCESSING

NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

EVEN FLØRENÆS

3. APRIL 2016

Part 1: Theory

Even Florenæs MTEL

a)

Received fequency is the remaining of the transmitted after doppler shift:

$$f_r = f_0 - 2f_0 \frac{v\cos\varphi}{c}$$

$$= \underline{\underline{f_0\left(1 - 2\frac{v\cos\varphi}{c}\right)}}$$

b)

The dopplershift:

$$\underline{\underline{f_d = f_0 \frac{2v\cos\varphi}{c}}}$$

c)

$f_0 = 2.5\ MHz$, $c = 1540 m/s$, $v = 1 m/s$

1) $\varphi = 90°$: $f_r = f_0\left(1 - 2\frac{v\cos 90°}{c}\right)$

$$= f_0 = \underline{\underline{2.5 MHz}}$$

$$f_d = f_0 \cdot 2 \frac{v\cos 90°}{c} = \underline{\underline{0}}$$

2) $\varphi = 45^\circ$

$$f_r = f_0\left(1 - \frac{2v\cos 45^\circ}{c}\right)$$

$$= f_0\left(1 - \frac{v}{c}\, 2\, \frac{\sqrt{2}}{2}\right) = f_0\left(1 - \frac{v}{c}\sqrt{2}\right)$$

$$= \underline{\underline{2.4977\ \text{MHz}}}$$

$$f_d = f_0\, \frac{v}{c}\sqrt{2} = 2295.8\ \text{Hz}$$

$$= \underline{\underline{2.3\ \text{kHz}}}$$

d) $r = 7.7\ \text{cm}$

$\varphi = 45^\circ$

$v_{max} = 1.5\ \text{m/s}$

1) Time per measurement:

$$\Delta t_{min} = \frac{2r}{c} = \underline{10^{-4}\ \text{s}}$$

$$PRF_{max} = \frac{1}{\Delta t_{min}} = 10^4\ \text{Hz} = \underline{\underline{10\ \text{kHz}}}$$

2)

$$PRF_{max} = \frac{4 f_0}{c}\, v_{max} \cos 45^\circ$$

$$f_0 = \frac{c \cdot PRF_{max}}{4 v_{max} \cdot \cos 45^\circ} =$$ ~~[illegible]~~

$$= \underline{\underline{3.63\ \text{MHz}}}$$

3) $\varphi = 30^\circ$ Even Florences MTEL

Need to find phase-shift for this case:

(From Ex. 8):

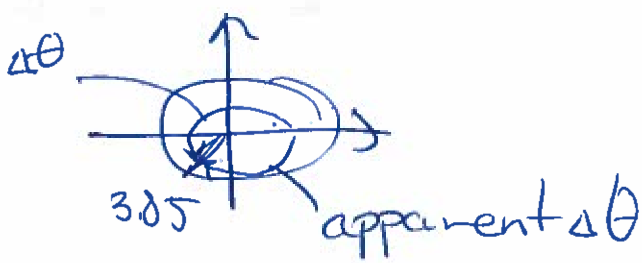
$$\Delta\theta = 4\pi \frac{v}{PRF \cdot \lambda} = 4\pi \frac{v \cdot f_0}{c \cdot PRF}$$

$$= 4\pi \frac{V_{max} \cdot f_0 \cdot \cos(\phi)}{c \cdot PRF}$$

$$= 4\pi \frac{3.63 \cdot 10^6 \frac{1}{s} \cdot 1.5 \frac{m}{s} \cdot \cos(30^\circ)}{1540 \frac{m}{s} \cdot 10^4 Hz}$$

$$= \underline{3.85} \text{ rad}$$

$\Delta\theta$ will not be the apparent $\Delta\theta$:



$\Delta\theta_{apparent}$ = Apparent $\Delta\theta$

$$\Delta\theta_{apparent} = -(2\pi - 3.85) = \underline{-2.43 \text{ rad}}$$

$$\Delta\theta_{apparent} = 4\pi \frac{V_{apparent} \cdot f_0}{c \cdot PRF}$$

$$V_{apparent} = \Delta\theta_{apparent} \frac{c \cdot PRF}{4\pi \cdot f_0} = \underline{\underline{-0.82 \text{ m/s}}}$$

$$V_{apparant} = V_{Max\,apparat} \cdot \cos(30°)$$

$$V_{max\,apparant} = \frac{V_{apparant}}{\cos 30°} = -0.95\ m/s$$

Doppler shift:

$$f_d = f_o \frac{2 V_{apparant}}{c} = -3865.7\ Hz$$

$$= -3.9\ kHz$$

e)

Even Fløren og I-MTEL

1) CW Doppler have no depth resolution

True, CW Doppler have continous transmission of waves which will give no resolution along the beam. The results of transmission will all flow along the line-of-sight add together and mix.

2) Maximum velocity with PW (Measurable) is inverseley proportional to the distance

If this is true: $V_{max} = \frac{k}{r}$ where k - const. and r - distance to sample volume

Relation between PRF_{max} and V_{max}:

$$PRF_{max} = \frac{4f_0}{c} V_{max} \cos\theta \quad (1)$$

Find PRF_{max}:

$$t_{min} = \frac{2r}{c} \Rightarrow PRF_{max} = \frac{1}{t_{min}} = \frac{c}{2r} \quad (2)$$

$(2) \rightarrow (1)$:

$$\frac{c}{2r} = \frac{4f_0}{c} V_{max} \cos\theta$$

$$V_{max} = \frac{c^2}{2 \cdot 4 f_0 \cos\theta} \cdot \frac{1}{r} \overset{\text{if } f_0 = \text{const}}{=} \underline{\underline{k \cdot \frac{1}{r}}}$$

$$\underline{\underline{k = \frac{c^2}{8 f_0 \cos\theta}}}$$

statement is $\underline{\underline{\text{true}}}$

f) Increasing transmit frequency implies a better velocity resolution in PW doppler when the PRF is kept constant.

Velocity (max) in PW Doppler is explained by:

$$V_{max} = \frac{c^2}{8 f_0 \cos\theta} \cdot \frac{1}{r} = \frac{c \cdot PRF_{max}}{4 f_0 \cos\theta}$$

If f_0 is increased (with PRF kept const.) the resolution in velocity will decrease.

Statement is not true (untrue)

g)

In many practices Color flow replaces PW and CW. Color flow can easily be used for measuring size and direction for imaged organs. Timing information is more complicated in 2D-color flow display. For measuring velocity PW and CW would be a better choice. Statement is untrue.

h)

Formula for speed resolution:

$$V = \frac{c \cdot PRF}{f_0 \cdot 4\cos\theta}$$

A shorter pulse will not effect the speed resolution as it will change the spread of frequency (bandwith), but not the PRF or f_0. It will give better lateral resolution.

Statement untrue.

Exercise 9 TTK4165 Medical Signal Processing

Even Florenes Spring 2016

Contents

- Documentation
- Part 2 Pulsed Wave Doppler w/ analytic velocity
- Part 3 - Doppler shift and aliasing
- Part 4 - Doppler sound
- Part 5 - Clutter and clutter filter
- Part 6 - Blood flow measurement using Doppler

Documentation

Purpose:

Script answering tasks given in exercise 9 in the course TTK4165 Medical Signal Processing

Related files:

```
imagelog.m: Image a matrix of ultrasound power in log scale
```

Made by:

```
Even Florenes NTNU 2016
```

Last changes:

```
   2016-03-25 EF: First attempt on part 2,3 and 4

   2016-03-30 EF: First attempt on part 5,6

   2016-03-31 EF: Finished all parts, commented, updated documentation
                  and published
```

Status:

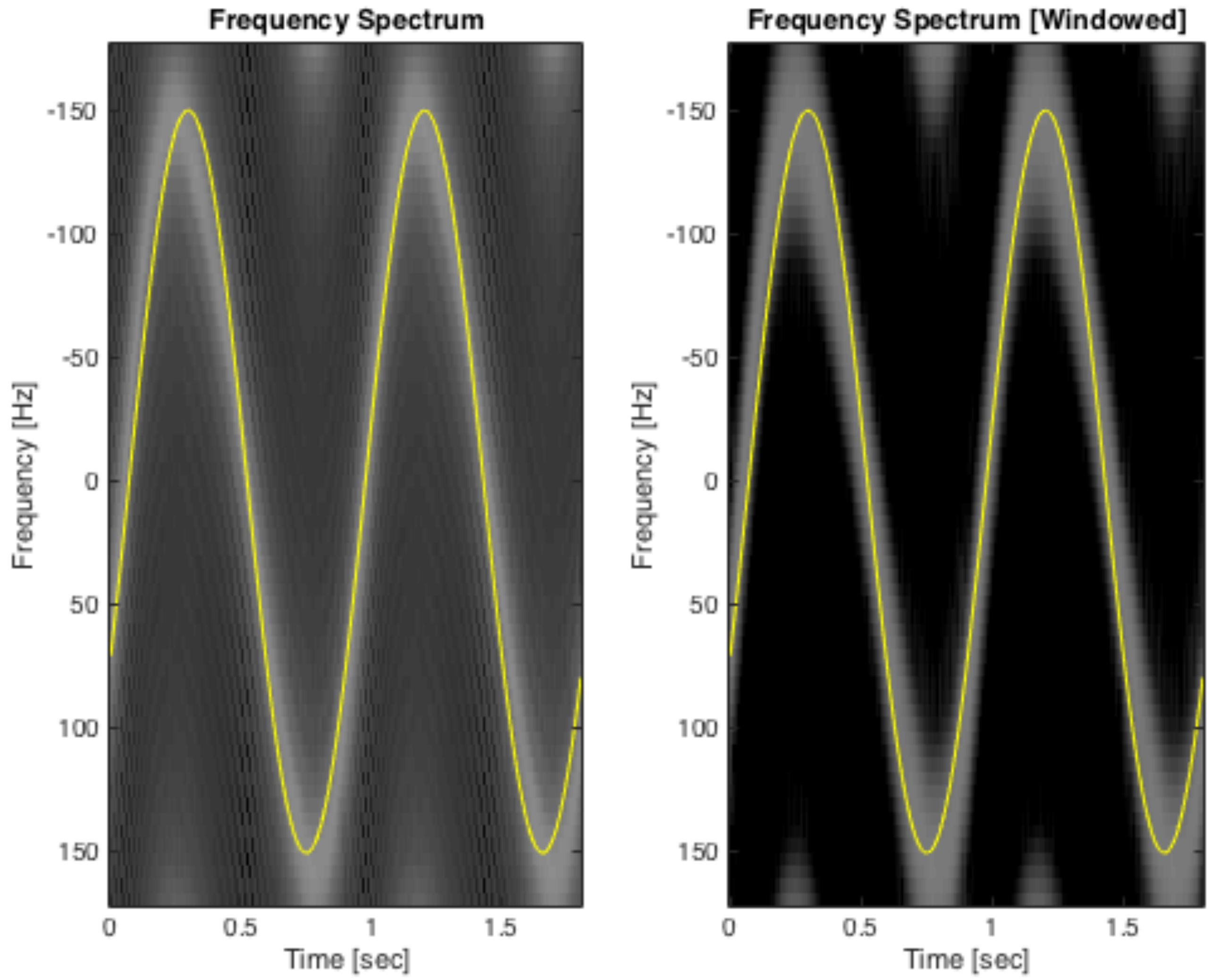
```
Works
```

Part 2 Pulsed Wave Doppler w/ analytic velocity

```
load slowmotion

% Find middle beam
middleBeamIq = squeeze(iq(:,4,:));

frameRate = s.Framerate_fps; % nFrames/seconds

nFrames = size(middleBeamIq,2); %nFrames

nSamples = size(middleBeamIq,1);
```

```
nSeconds = nFrames/frameRate;   %seconds = (nFrames/(nFrames/seconds))

time = 0:nSeconds/(nFrames-1):nSeconds;

distanceLength = s.iq.DepthIncrementIQ_m;

distance = 0:distanceLength/(nSamples-1):distanceLength;

% Find analytic velocity
rotationPeriod=0.908;
t0=0.0708;
excenterDistance=0.67/100;

pistonAngularFrequency = (2*pi*(time-t0))/rotationPeriod;
pistonVelocityAmplitude = -(2*pi*excenterDistance)/rotationPeriod;

pistonVelocity = pistonVelocityAmplitude*sin(pistonAngularFrequency);
pointVelocity = -pistonVelocity;

centerFrequency = s.fProbe_Hz;
speedSound = 1540;

dopplerShift = -centerFrequency*(2*pointVelocity)/speedSound;

% Make Pulsed Wave Doppler Spectrum
Nfft=64; %Zeropadding to length 64
crop=16;
depthMiddle = round(size(middleBeamIq,1)/2);
depthindex = depthMiddle-9:depthMiddle+10;
PHamming=zeros(Nfft, nFrames-crop+1);
P=zeros(Nfft, nFrames-crop+1);
for n=1:nFrames-crop+1,
    middleBeamIqFrames=middleBeamIq(depthindex,n+[0:crop-1])';
    P(:,n) = mean(abs(fftshift(fft(middleBeamIqFrames,Nfft))),2);
    middleBeamIqFrames=middleBeamIqFrames.*(hamming(crop)*ones(1,length(depthindex)));
    PHamming(:,n)=mean(abs(fftshift(fft(middleBeamIqFrames,Nfft))),2);
end;
%Frequency axis
frequencyAxis=(([0:Nfft-1]/Nfft)-0.5)*frameRate;

%Greyscale image of frequency specter in dB
gain = -25;
dynamicRange = 40;

timeAxis = 0:nSeconds/(size(PHamming,2)-1):nSeconds;
PHamming=imagelog(PHamming,gain,dynamicRange);
P = imagelog(P,gain,dynamicRange);
figure(1);
% Plot image without windowing
subplot(1,2,1),image(timeAxis,frequencyAxis,P),colormap(gray(64));
hold on
subplot(1,2,1),plot(time,dopplerShift,'y'),title('Frequency Spectrum'),xlabel('Time [sec]')
,...
    ylabel('Frequency [Hz]');
% Plot image with hamming windowing
subplot(1,2,2),image(timeAxis,frequencyAxis,PHamming),colormap(gray(64));
hold on
subplot(1,2,2),plot(time,dopplerShift,'y'),title('Frequency Spectrum [Windowed]'),xlabel('T
ime [sec]'),...
    ylabel('Frequency [Hz]');
hold off
```

```
%
% Comments on image:
% Looking at the image you can see that the doppler frequency is
% higher then the maximum allowed by Nyquist criterion. Which leads to
% aliasing. The aliasing is shown in the image by the amplitude of the
% doppler frequency exceeding the window, and the remaining of the amplitude is
% flipped to the opposite side of the window.
%
% Using hamming window leads to less noisy image.
%
```

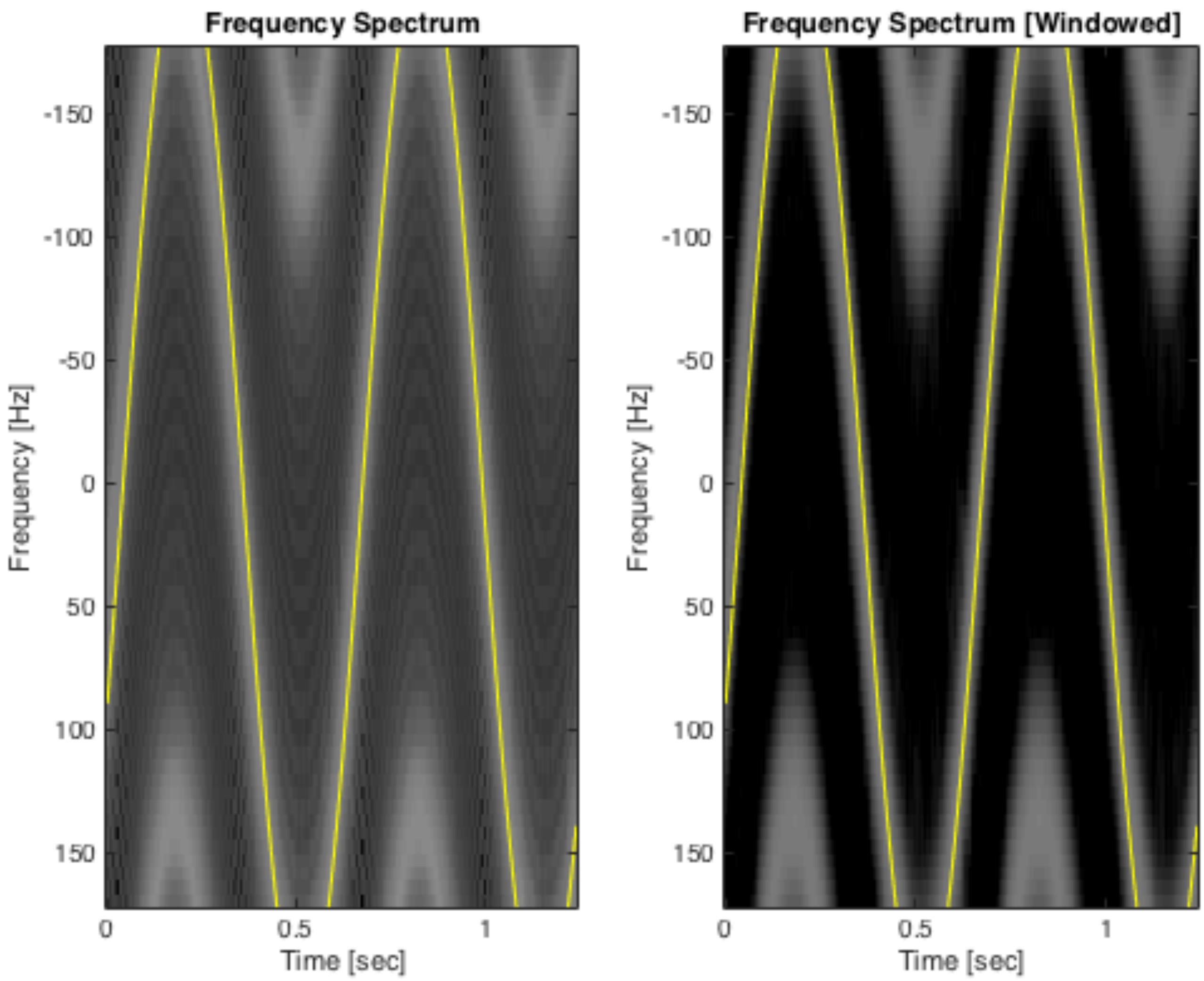


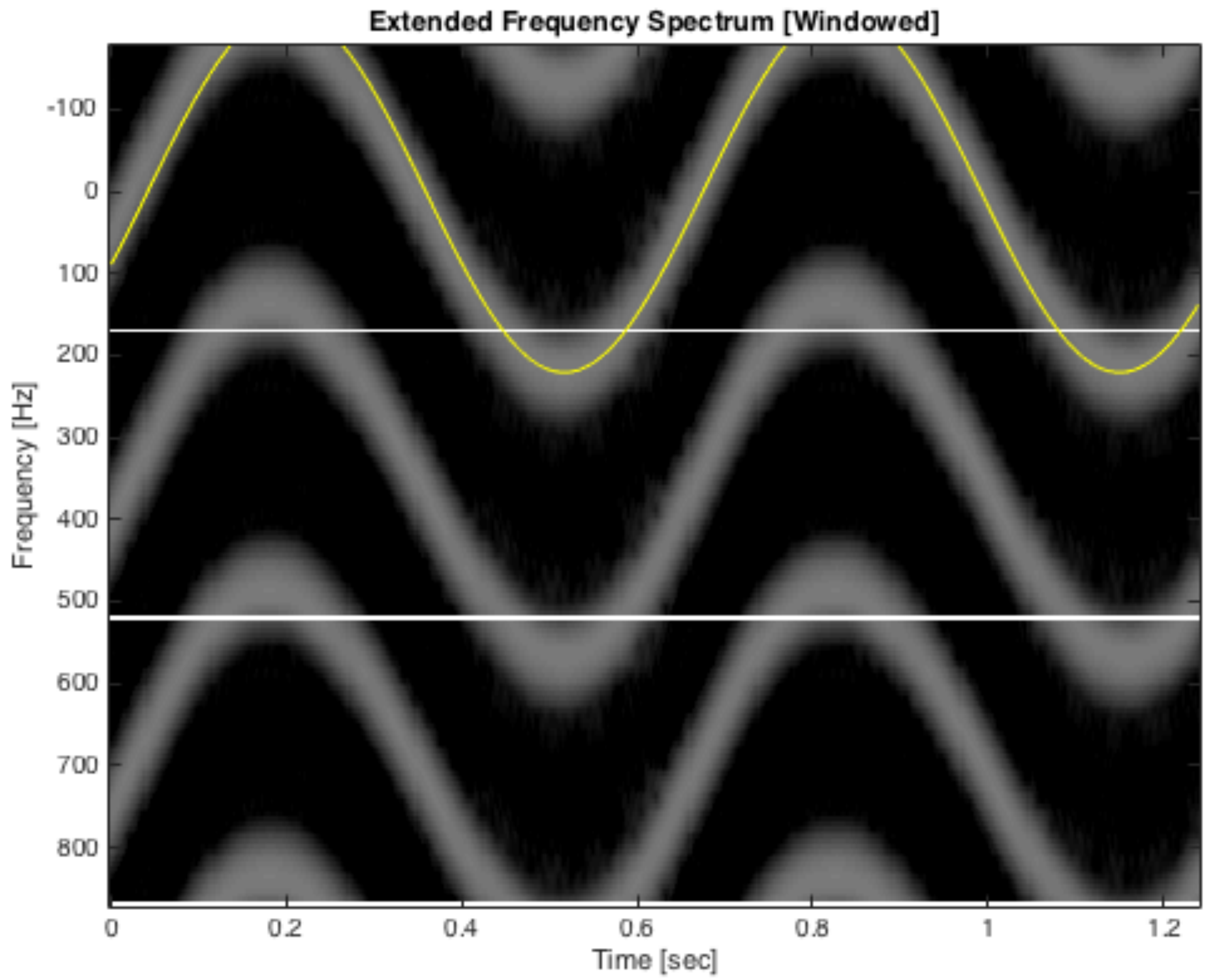
Part 3 - Doppler shift and aliasing

```
load fastmotion.mat

% Find middle beam iq
middleBeamIq = squeeze(iq(:,4,:));

frameRate = s.Framerate_fps; % nFrames/seconds

nFrames = size(middleBeamIq,2); %nFrames

nSamples = size(middleBeamIq,1);

nSeconds = nFrames/frameRate;   %seconds = (nFrames/(nFrames/seconds))

time = 0:nSeconds/(nFrames-1):nSeconds;

distanceLength = s.iq.DepthIncrementIQ_m;

distance = 0:distanceLength/(nSamples-1):distanceLength;
```

```
% From suggested solution exercise 8
x=[0.0419;0.356;0.675];y=[2.75;4.12;2.78];
excenterDistance=((y(2)-y(1))/2)/100;
rotationPeriod=x(3)-x(1);
t0=x(1);%R in cm, T and t0 in seconds

pistonAngularFrequency = (2*pi*(time-t0))/rotationPeriod;
pistonVelocityAmplitude = -(2*pi*excenterDistance)/rotationPeriod;

pistonVelocity = pistonVelocityAmplitude*sin(pistonAngularFrequency);
pointVelocity = -pistonVelocity;


centerFrequency = s.fProbe_Hz;
speedSound = 1540;

dopplerShift = -centerFrequency*(2*pointVelocity)/speedSound;

% Make Pulsed Wave Doppler Spectrum
Nfft=64; %Zeropadding to length 64
crop=16;
depthMiddle = round(size(middleBeamIq,1)/2);
depthindex = depthMiddle-9:depthMiddle+10;
PHamming=zeros(Nfft, nFrames-crop+1);
P=zeros(Nfft, nFrames-crop+1);
for n=1:nFrames-crop+1,
    middleBeamIqFrames=middleBeamIq(depthindex,n+[0:crop-1])';
    P(:,n) = mean(abs(fftshift(fft(middleBeamIqFrames,Nfft))),2);
    middleBeamIqFrames=middleBeamIqFrames.*(hamming(crop)*ones(1,length(depthindex)));
    PHamming(:,n)=mean(abs(fftshift(fft(middleBeamIqFrames,Nfft))),2);
end;
%Frequency axis
frequencyAxis=(([0:Nfft-1]/Nfft)-0.5)*frameRate;

%Greyscale image of frequency specter in dB
gain = -25;
dynamicRange = 40;

timeAxis = 0:nSeconds/(size(PHamming,2)-1):nSeconds;
PHamming=imagelog(PHamming,gain,dynamicRange);
P = imagelog(P,gain,dynamicRange);
figure(2);
% Plot image without windowing
subplot(1,2,1),image(timeAxis,frequencyAxis,P),colormap(gray(64));
hold on
subplot(1,2,1),plot(time,dopplerShift,'y'),title('Frequency Spectrum'),xlabel('Time [sec]')
,...
    ylabel('Frequency [Hz]');
% Plot image with hamming windowing
subplot(1,2,2),image(timeAxis,frequencyAxis,PHamming),colormap(gray(64));
hold on
subplot(1,2,2),plot(time,dopplerShift,'y'),title('Frequency Spectrum [Windowed]'),xlabel('T
ime [sec]'),...
    ylabel('Frequency [Hz]');

PhammingExtendable = PHamming;
PhammingExtendable(end,:) = 64;
PExtended = [PhammingExtendable;PhammingExtendable;PhammingExtendable];

frequencyAxisStacked=[frequencyAxis,-2*min(frequencyAxis)+frequencyAxis+1,-4*min(frequencyA
xis)+frequencyAxis+1];
```

```
figure(3);
image(timeAxis,frequencyAxisStacked,PExtended),colormap(gray(64));
hold on
plot(time,dopplerShift,'y'),title('Extended Frequency Spectrum [Windowed]'),xlabel('Time [s
ec]'),...
    ylabel('Frequency [Hz]');

% Comments on image:
% The Nyquist limit in the extended image is shown as a white line in the
% image.
%
```

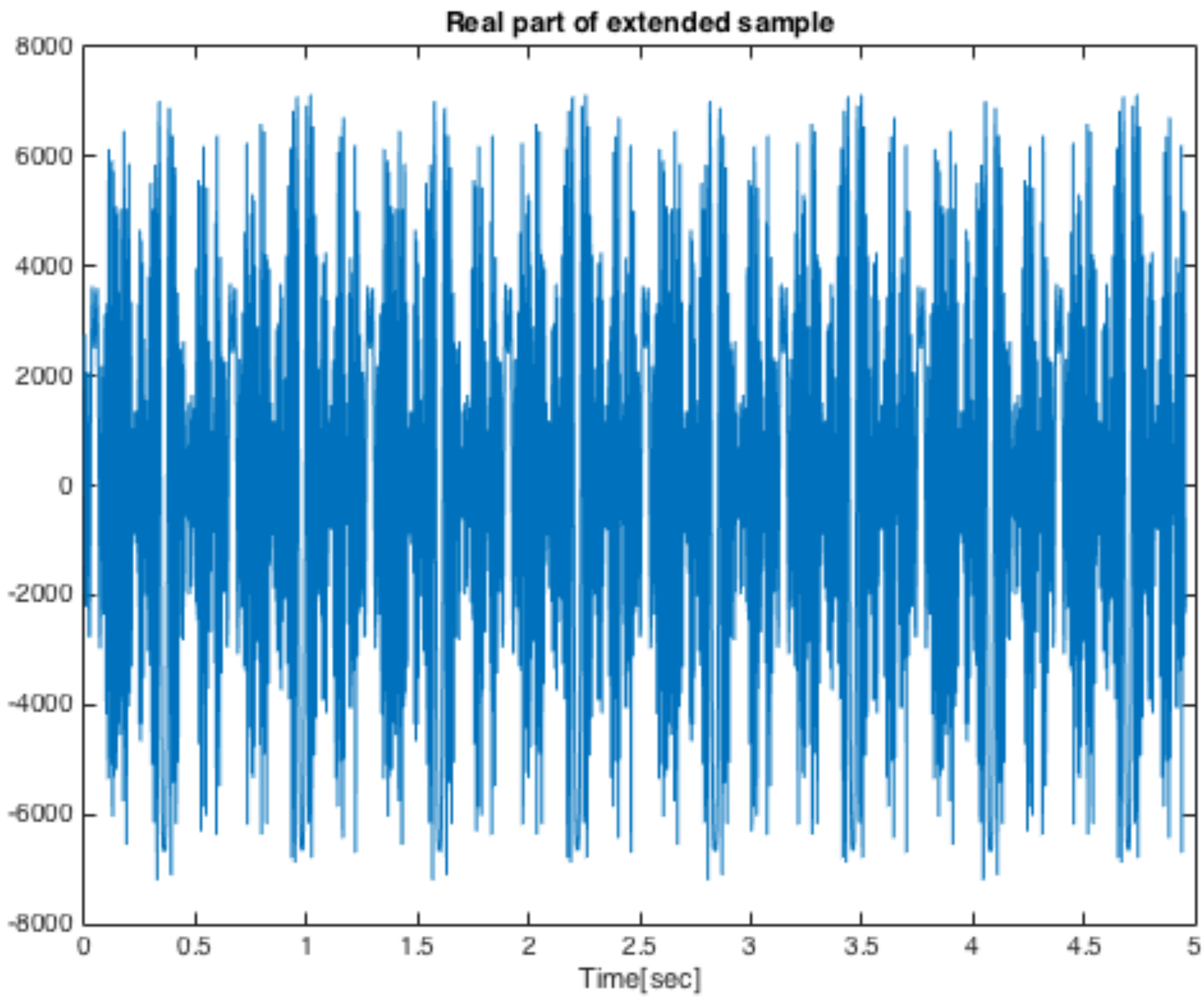


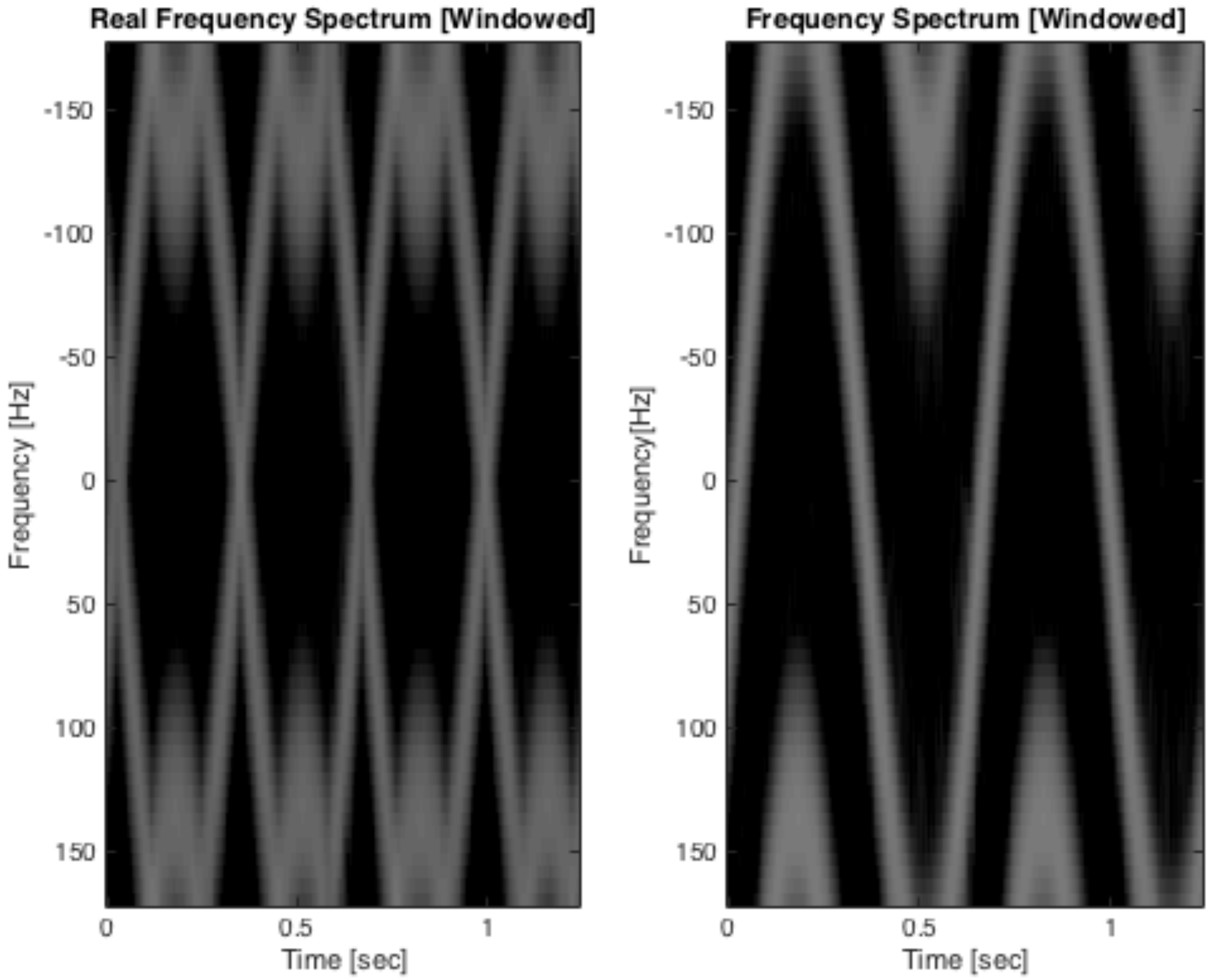


Part 4 - Doppler sound

```
% Find middle sample
iqSample = middleBeamIq(round(size(middleBeamIq,1)/2),:);

% Extend sample, find real part, resample and rescale
iqSampleExtended = [iqSample,iqSample,iqSample,iqSample];
realSample = real(iqSampleExtended);
realSampleResampled = resample(realSample,8192,round(frameRate));
realSampleScaled = realSampleResampled/max(abs(realSampleResampled));

% Play hearable doppler frequency
soundsc(realSampleScaled,8192,8);

% Plot changes in time domain
timeExtended = 0:time(end)*4/(size(realSample,2)-1):time(end)*4;
figure(4);
plot(timeExtended,realSample),xlabel('Time[sec]'),...
    title('Real part of extended sample');

% Image FFT of real sample
PHammingReal=zeros(Nfft, nFrames-crop+1);
for n=1:nFrames-crop+1,
    middleBeamIqFrames=middleBeamIq(depthindex,n+[0:crop-1])';
    middleBeamIqFrames=middleBeamIqFrames.*(hamming(crop)*ones(1,length(depthindex)));
    PHammingReal(:,n)=mean(abs(fftshift(fft(real(middleBeamIqFrames),Nfft))),2);
end
timeAxis = 0:nSeconds/(size(PHamming,2)-1):nSeconds;
PHammingReal=imagelog(PHammingReal,gain,dynamicRange);
figure(5),subplot(1,2,1),image(timeAxis,frequencyAxis,PHammingReal),colormap(gray(64));
title('Real Frequency Spectrum [Windowed]'),xlabel('Time [sec]'),...
    ylabel('Frequency [Hz]');
subplot(1,2,2),image(timeAxis,frequencyAxis,PHamming),colormap(gray(64)),...
```

```
title('Frequency Spectrum [Windowed]'),xlabel('Time [sec]'),...
    ylabel('Frequency[Hz]');

% Comments on time plot:
% The time plot show that the extended sample looks many sinc-functions
% with varying amplitude and frequency.
%
%
%
% Comments on image:
%
% FFT of the real part of the signal produces an image where the doppler
% shift varies simultaneous as a negative and positive frequency. The two
% variations are perfectly matched.
%
%
```



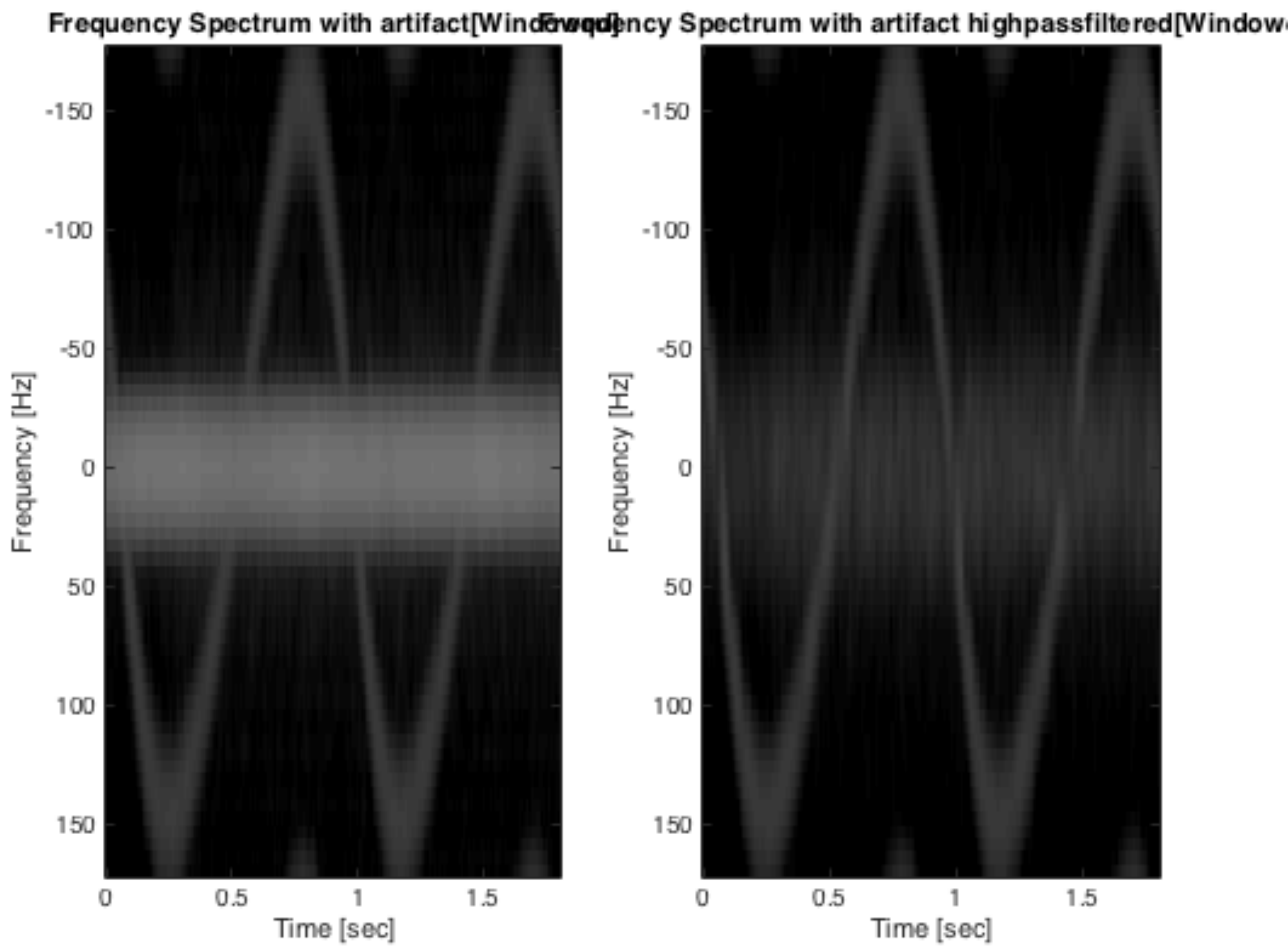


Part 5 - Clutter and clutter filter

```
load slowmotion_clutter

% Find middle beam
middleBeamIq = squeeze(iq(:,4,:));

frameRate = s.Framerate_fps; % nFrames/seconds

nFrames = size(middleBeamIq,2); %nFrames

nSamples = size(middleBeamIq,1);

nSeconds = nFrames/frameRate;   %seconds = (nFrames/(nFrames/seconds))

time = 0:nSeconds/(nFrames-1):nSeconds;

distanceLength = s.iq.DepthIncrementIQ_m;

distance = 0:distanceLength/(nSamples-1):distanceLength;

% Make Pulsed Wave Doppler Spectrum
Nfft=64; %Zeropadding to length 64
crop=16;
depthMiddle = round(size(middleBeamIq,1)/2);
depthindex = depthMiddle-9:depthMiddle+10;
PHamming=zeros(Nfft, nFrames-crop+1);
for n=1:nFrames-crop+1,
    middleBeamIqFrames=middleBeamIq(depthindex,n+[0:crop-1])';
    middleBeamIqFrames=middleBeamIqFrames.*(hamming(crop)*ones(1,length(depthindex)));
    PHamming(:,n)=mean(abs(fftshift(fft(middleBeamIqFrames,Nfft))),2);
end;
%Frequency axis
```

```
frequencyAxis=(([0:Nfft-1]/Nfft)-0.5)*frameRate;

%Greyscale image of frequency specter in dB
gain = -25;
dynamicRange = 40;

timeAxis = 0:nSeconds/(size(PHamming,2)-1):nSeconds;
PHamming=imagelog(PHamming,gain,dynamicRange);
figure(6);
% Plot image without windowing
subplot(1,2,1),image(timeAxis,frequencyAxis,PHamming),colormap(gray(64));
title('Frequency Spectrum with artifact[Windowed]'),xlabel('Time [sec]'),...
    ylabel('Frequency [Hz]');

% Low pass filter
nFilterCoefficients = 8;
filterCoefficents=ones(1,nFilterCoefficients); %=boxcar(N). May also use hamming(N), hannin
g(N), ....
filterCoefficents=filterCoefficents/sum(filterCoefficents); %Normalization of filter coeffi
cients
iqLowPassFiltered=filter(filterCoefficents,1,middleBeamIq,[],2); %Filter along rows
iqHighPassFiltered=middleBeamIq-iqLowPassFiltered; %Subtract low pass component:

% Make Pulsed Wave Doppler Spectrum
Nfft=64; %Zeropadding to length 64
crop=16;
depthMiddle = round(size(middleBeamIq,1)/2);
depthindex = depthMiddle-9:depthMiddle+10;
PHamming=zeros(Nfft, nFrames-crop+1);
for n=1:nFrames-crop+1,
    middleBeamIqFrames=iqHighPassFiltered(depthindex,n+[0:crop-1])';
    middleBeamIqFrames=middleBeamIqFrames.*(hamming(crop)*ones(1,length(depthindex)));
    PHamming(:,n)=mean(abs(fftshift(fft(middleBeamIqFrames,Nfft))),2);
end;
%Frequency axis
frequencyAxis=(([0:Nfft-1]/Nfft)-0.5)*frameRate;

%Greyscale image of frequency specter in dB
gain = -25;
dynamicRange = 40;

timeAxis = 0:nSeconds/(size(PHamming,2)-1):nSeconds;
PHamming=imagelog(PHamming,gain,dynamicRange);
% Plot image without windowing
figure(6);
subplot(1,2,2),image(timeAxis,frequencyAxis,PHamming),colormap(gray(64));
title('Frequency Spectrum with artifact highpassfiltered[Windowed]'),xlabel('Time [sec]'),.
..
    ylabel('Frequency [Hz]');

% Make sound of filtered and unfiltered

% Find middle sample
iqSample = middleBeamIq(round(size(middleBeamIq,1)/2),:);
iqSampleFiltered = iqHighPassFiltered(round(size(iqHighPassFiltered,1)/2),:);
% Extend sample, find real part, resample and rescale
iqSampleExtended = [iqSample,iqSample,iqSample,iqSample];
iqSampleExtendedFiltered = [iqSampleFiltered,iqSampleFiltered,iqSampleFiltered ,...
    iqSampleFiltered];
iqSamples = [iqSampleExtended;iqSampleExtendedFiltered];
for i = 1:size(iqSamples,1)
```

```
    realSample = real(iqSamples(i,:));
    realSampleResampled = resample(realSample,8192,round(frameRate));
    realSampleScaled = realSampleResampled/max(abs(realSampleResampled));

    % Play hearable doppler frequency
    soundsc(realSampleScaled,8192,8);
    pause(10);
end % for i

% Comments on clutter image:
% The slowmotion_clutter.mat gives a more noisy image. The image shows that
% there are lot of small movements in the imaged area, which leads to the
% quality of the large movement in the image is reduced, compared with
% slowmotion.mat.

% Comments on sound test:
% In the unfiltered sample the clutter movements leads to the sound of the
% blood flow being contaminated.
%
% The filtering removes some of the noise due to clutter movements, which
% makes the sound of the blood flow more hearable
%
```



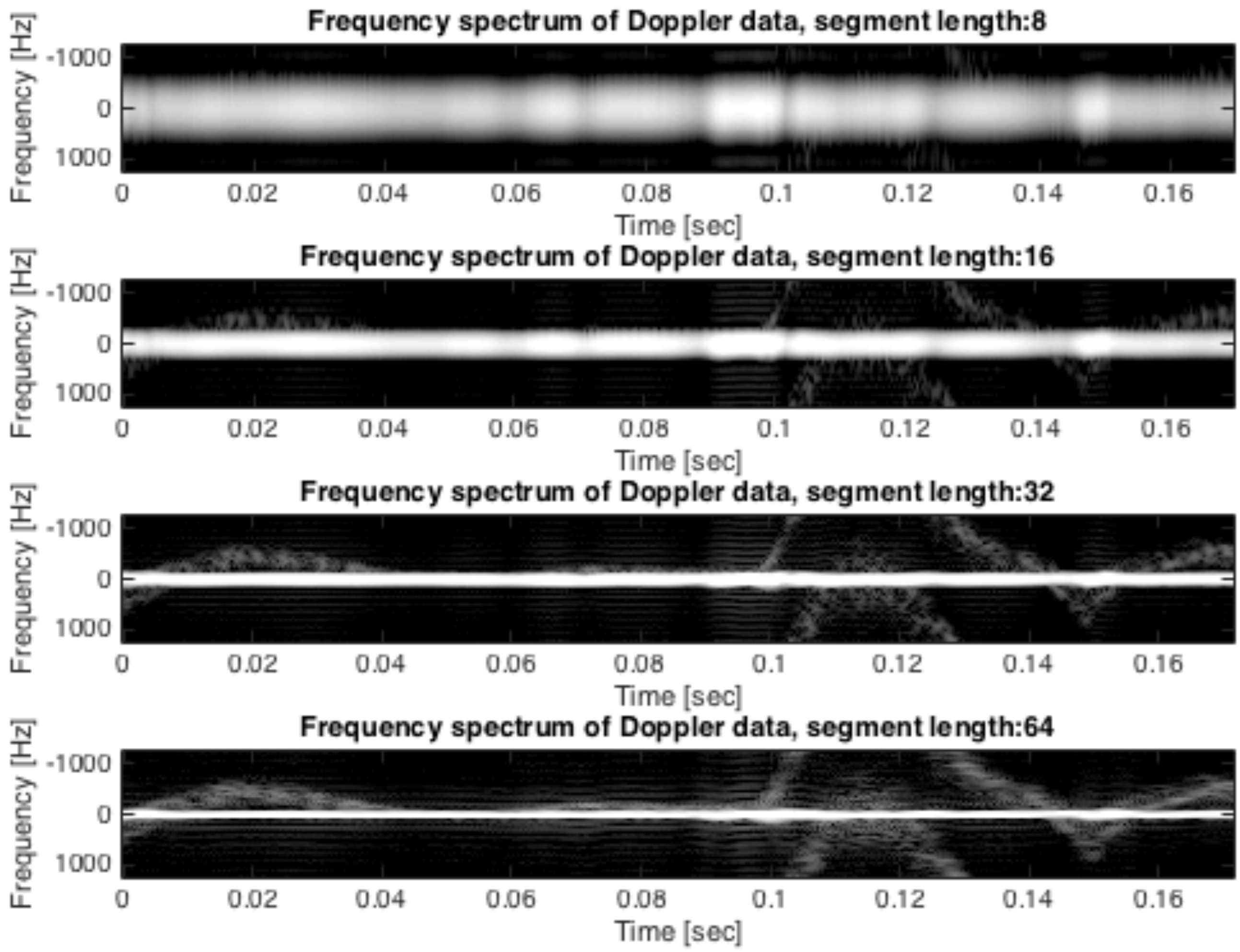
Part 6 - Blood flow measurement using Doppler

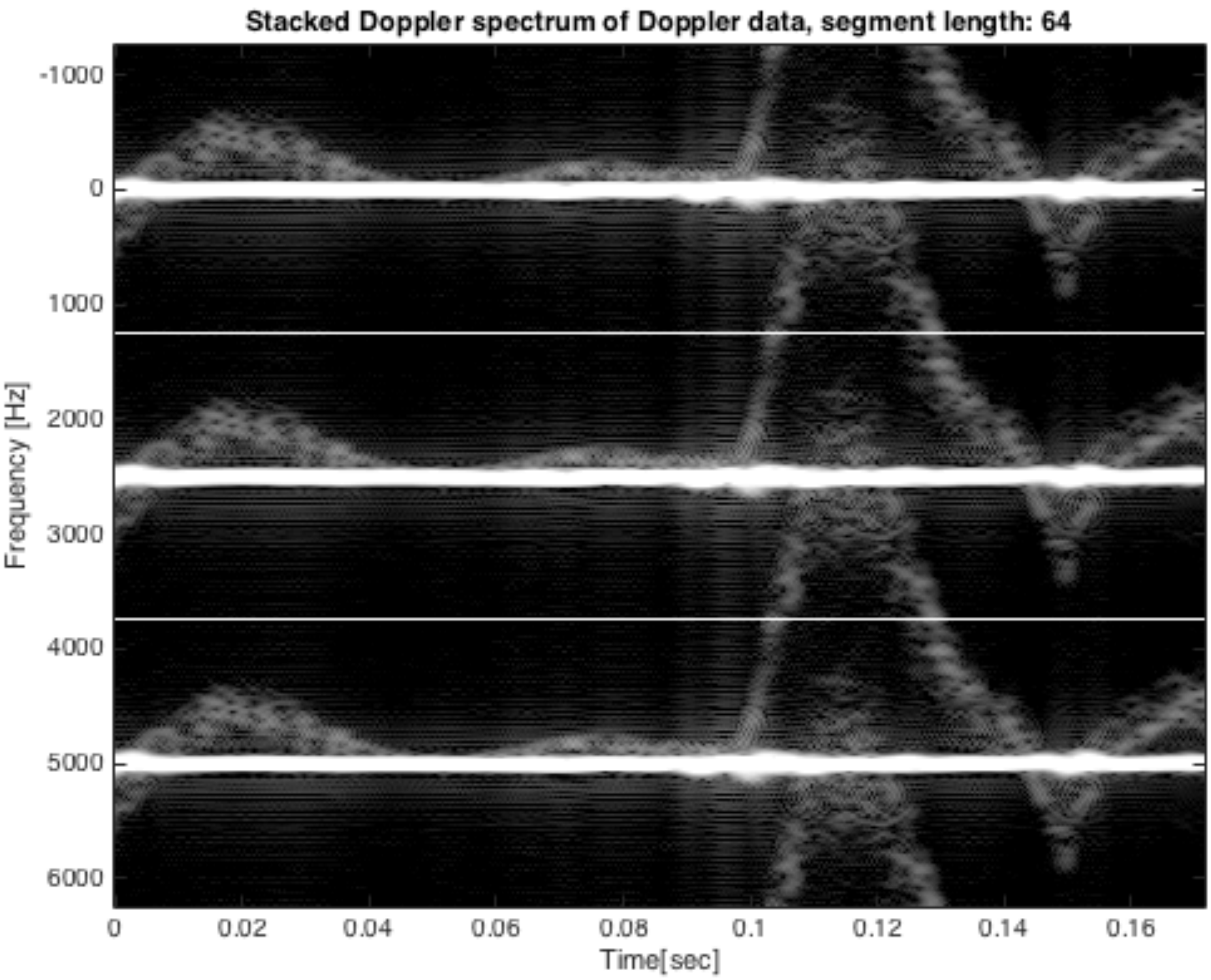
```
load Dopplerdata
frameRate = s.Framerate_fps; % nFrames/seconds
% Find middle beam
middleBeamIq = iq;
nFrames = size(middleBeamIq,2); %nFrames

nSamples = size(middleBeamIq,1);
```

```
nSeconds = nFrames/frameRate;   %seconds = (nFrames/(nFrames/seconds))




% Make Pulsed Wave Doppler Spectrum
Nfft=256; %Zeropadding to length 64
crops=[8,16,32,64];
for i = 1:length(crops)
    crop = crops(i);
    depthindex = [70:80];
    PHamming=zeros(Nfft, nFrames-crop+1);
    for n=1:nFrames-crop+1,
        middleBeamIqFrames=middleBeamIq(depthindex,n+[0:crop-1])';
        middleBeamIqFrames=middleBeamIqFrames.*(hamming(crop)*ones(1,length(depthindex)));
        PHamming(:,n)=mean(abs(fftshift(fft(middleBeamIqFrames,Nfft))),2);
    end;
    %Frequency axis
    frequencyAxis=(([0:Nfft-1]/Nfft)-0.5)*prf;

    %Greyscale image of frequency specter in dB
    gain = -20;
    dynamicRange = 20;

    timeAxis = 0:nSeconds/(size(PHamming,2)-1):1/nSeconds;
    PHamming=imagelog(PHamming,gain,dynamicRange);
    figure(7);
    plotTitle = ['Frequency spectrum of Doppler data, segment length:' num2str(crop)];
    % Plot image without windowing
    subplot(length(crops),1,i),image(timeAxis,frequencyAxis,PHamming),colormap(gray(64)),..
.
        title(plotTitle),xlabel('Time [sec]'),...
        ylabel('Frequency [Hz]');
end
PHammingStakable = PHamming;
PHammingStakable(1,:) = 64;
PHammingStakable(end,:) = 64;
PHammingStacked = [PHammingStakable;PHammingStakable;PHammingStakable];
frequencyAxisStacked=[frequencyAxis,-2*min(frequencyAxis)+frequencyAxis+1,-4*min(frequencyA
xis)+frequencyAxis+1];
figure(8),image(timeAxis,frequencyAxisStacked,PHammingStacked),colormap(gray(64)),...
    title('Stacked Doppler spectrum of Doppler data, segment length: 64'),xlabel('Time[sec]
'),...
    ylabel('Frequency [Hz]');
hold on
%[x,y] = ginput(2);

centerFrequency = f0;
speedSound = 1540; %cm/s
pulseRepetition = prf;
nyquistSpeed = (speedSound*pulseRepetition)/(4*centerFrequency);
dopplerShift = centerFrequency*(2*nyquistSpeed)/speedSound;
% Measured Nyquist limit:

%Measured using ginput:maxDopplerShift = y(2)-y(1);
maxDopplerShift = 2864.81;
maxVelocity = (speedSound*maxDopplerShift)/(2*centerFrequency);
pulseRepetitionForMaxVelocity = (4*centerFrequency*maxVelocity)/speedSound;
fprintf('Nyquist speed with given PRF: %g m/s\n',nyquistSpeed);
fprintf('Nyquist doppler shift: %g Hz\n',dopplerShift);
```

```
fprintf('Measured maximum doppler shift: %g Hz \n',maxDopplerShift);
fprintf('Maximum velocity: %g m/s\n',maxVelocity);
fprintf('PRF needed to avoid aliasing: %.0f Hz\n',pulseRepetitionForMaxVelocity);
```

```
Nyquist speed with given PRF: 0.385 m/s
Nyquist doppler shift: 1250 Hz
Measured maximum doppler shift: 2864.81 Hz
Maximum velocity: 0.882361 m/s
PRF needed to avoid aliasing: 5730 Hz
```





Published with MATLAB® R2015a