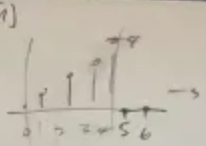
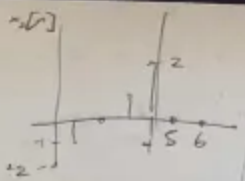
(a)





n	x_1	x_2
0	0	-2
1	1	-1
2	2	0
3	3	1
4	4	2
5	0	0
6	0	0

n	$x_3[n]$
0	$x_1[0]x_2[0] + x_1[0]x_2[6] + x_1[2]x_2[5] + x_1[3]x_2[4] + x_1[4]x_2[3] + x_1[5]x_2[2]$ $= 0+0+0+6+4 = \boxed{10}$
1	$x_1[0]x_2[1] + x_1[1]x_2[0] + x_1[2]x_2[6] + x_1[3]x_2[5] + x_1[4]x_2[4]$ $= 0+-2+0+0+8 = \boxed{6}$
2	$0+1(0)+2(-2)+0+0 = \boxed{-5}$
3	$0+0+2(-1)+3(-2)+0+0+0 = \boxed{-8}$
4	$0+1(1)+0+3(-1)+4(-2)+0+0 = \boxed{-10}$
5	$0+1(2)+2(1)+0+4(-1)+0+0 = \boxed{0}$
6	$0+0+4+3+0+0 = \boxed{7}$

$\Rightarrow x_3[n] = [10, 6, -5, -8, -10, 0, 7]$

(b)

n	x_1	x_2
0	0	-2
1	1	-1
2	2	0
3	3	1
4	4	2
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0

$x_1[n] \circledast x_2[n] = y[n]$

n	y[n]
6	7
7	0 + 0 + 0 + 6 + 4 + 0 + 0 + ... = 10
8	0 + 0 + 0 + 0 + 8 + 0 + 0 ... = 8
9	0
10	0
⋮	0

$N = 9$

Contents


```
%1a

x1 = [0 1 2 3 4];
x2 = [-2 -1 0 1 2];
x3 = ifft( fft(x1,7) .* fft(x2,7));

fprintf('1a, 7-point circular convolution \n');
for i=1:length(x3)
    fprintf('%.2f\n',x3(i));

end
```

```
1a, 7-point circular convolution
10.00
6.00
-5.00
-8.00
-10.00
-0.00
7.00
```

1b

```
y = conv(x1,x2);

fprintf('1b, linear convolution \n');
for i=1:length(y)
    fprintf('%.2f\n',y(i));
end

x_test = ifft( fft(x1,9) .* fft(x2,9));

fprintf('1b, N = 9-point circular convolution \n');

for i=1:length(x_test)
    fprintf('%.2f\n',x_test(i));
end
```

```
1b, linear convolution
0.00
-2.00
-5.00
-8.00
-10.00
0.00
7.00
10.00
```

```
8.00
1b, N = 9-point circular convolution
0.00
-2.00
-5.00
-8.00
-10.00
0.00
7.00
10.00
8.00
```

1c

```
%testing out higher N > 9 point circular convolutions


x_test2 = ifft( fft(x1,10) .* fft(x2,10));

fprintf('1c, N = 10-point circular convolution \n');

for i=1:10
    fprintf('%.2f\n',x_test2(i));
end


x_test3 = ifft( fft(x1,11) .* fft(x2,11));

fprintf('1c, N = 11-point circular convolution \n');

for i=1:10
    fprintf('%.2f\n',x_test3(i));
end


x_testN = ifft( fft(x1,100) .* fft(x2,100));

fprintf('1c, N = 100-point circular convolution \n');

for i=1:10
    fprintf('%.2f\n',x_testN(i));
end
```

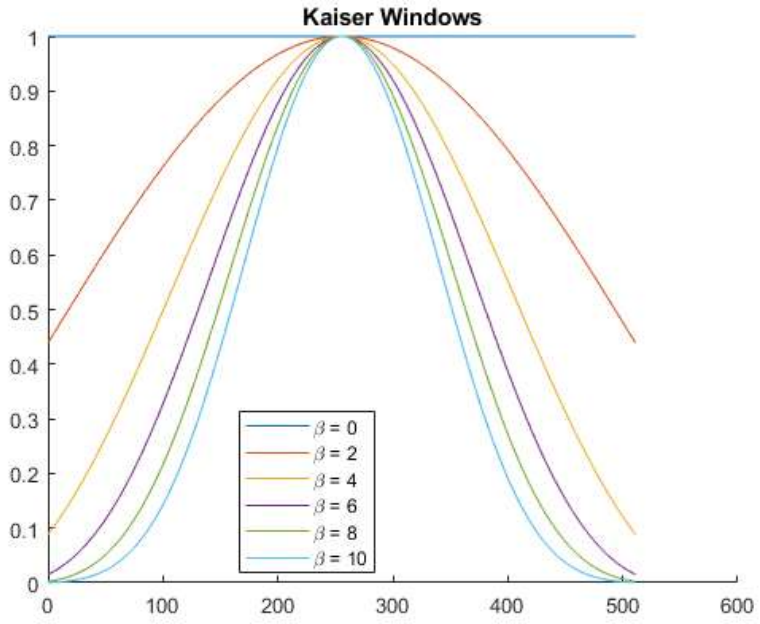
```
1c, N = 10-point circular convolution
0.00
-2.00
-5.00
-8.00
-10.00
0.00
7.00
10.00
8.00
0.00
1c, N = 11-point circular convolution
```

```
  0.00
 -2.00
 -5.00
 -8.00
-10.00
 -0.00
  7.00
 10.00
  8.00
  0.00
1c, N = 100-point circular convolution
  0.00
 -2.00
 -5.00
 -8.00
-10.00
  0.00
  7.00
 10.00
  8.00
  0.00
```

Published with MATLAB® R2024b

Contents


```
% 2a
samp = 512; %window length
beta = [0 2 4 6 8 10];%kaiser parameters
figure;
hold on;
for i = 1:length(beta) %for all beta values

    w = kaiser(N,beta(i));

    plot(0:N-1,w,'DisplayName', sprintf('\\beta = %g', beta(i)));
end

legend('show','Location','best');
title('Kaiser Windows')
hold off;
```

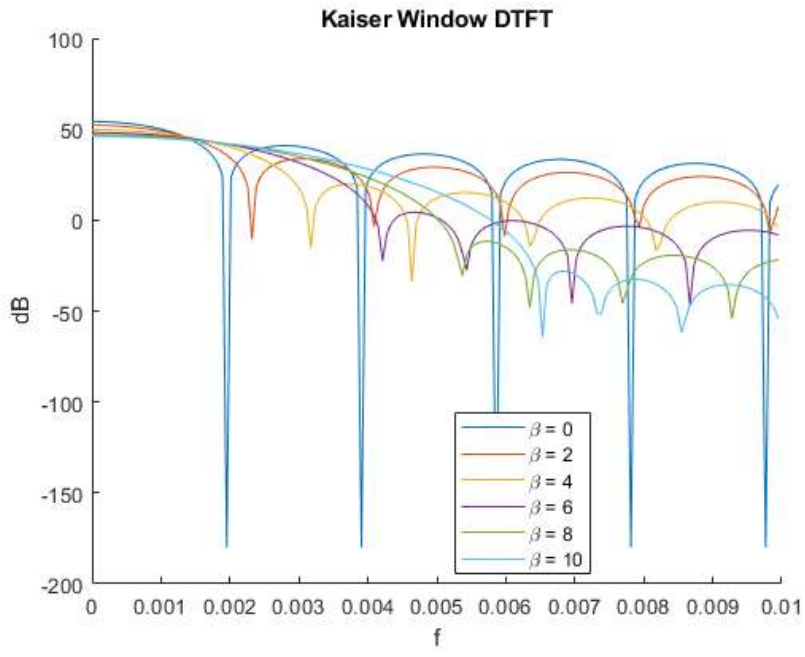


2b , 2c

```
N = 512; %window length again
beta = [0 2 4 6 8 10]; %kaiser parameters again

NFFT = 16*1024; %fft sample size


figure;
hold on;

for i = 1:length(beta)
    w = kaiser(N, beta(i));
    w_fft = fft(w,NFFT); %computing DTFT
    f = linspace(0,1,NFFT); %frequency axis
    w_fft_dB = 20 * log10(abs(w_fft)+0.000000001); %doesn't like log10(0)
    plot(f(f <= 0.01),w_fft_dB(f <= 0.01),'DisplayName', ['\beta = ', num2str(beta(i))]); %limiting axis to lower discrete time frequency

    fprintf('For beta = %.0f\n',beta(i)); %2c segment
    fprintf('W(0) = %.4f\n', abs(w_fft(1)));
    fprintf('Window function sum = %.4f\n',sum(w)); % sum of w[n]
    fprintf('\n');
end
```

```
hold off;

xlabel('f');
ylabel('dB');
title('Kaiser Window DTFT');
legend('show','Location','best');

%ylim([-100,60]);
```

```
For beta = 0
W(0) = 512.0000
Window function sum = 512.0000

For beta = 2
W(0) = 406.9431
Window function sum = 406.9431

For beta = 4
W(0) = 308.5565
Window function sum = 308.5565

For beta = 6
W(0) = 255.5273
Window function sum = 255.5273

For beta = 8
W(0) = 222.6691
Window function sum = 222.6691

For beta = 10
W(0) = 199.8700
Window function sum = 199.8700
```



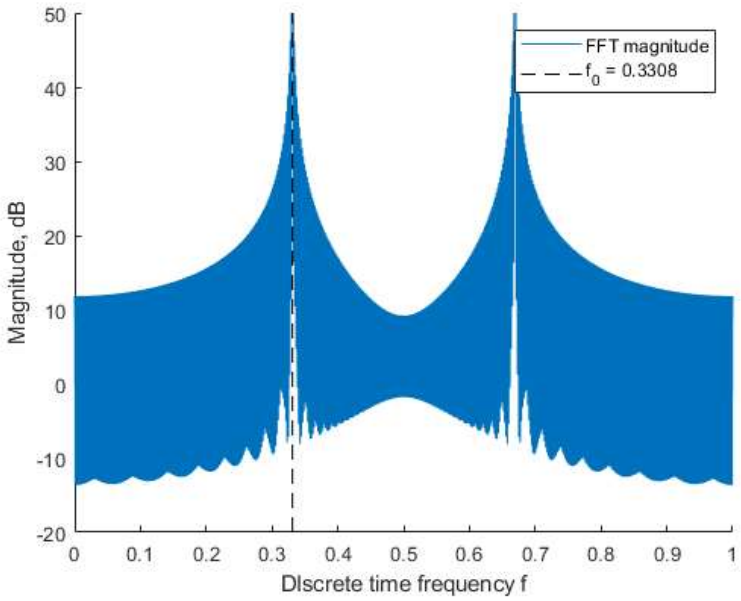
2d

```
%i - Width increases for increasing beta.

%ii - Height decreases for increasing beta.
```

Published with MATLAB® R2024b

Contents


```
A = 3.7; %amplitude in V
f0 = 0.3308;
N = 512; %block size
NFFT = 32768; %FFT points

n = linspace(0,N-1,N); %time indices
x_w = A * cos(2*pi()*f0*n); %sample signal as function of time indices

fft_x_w = fft(x_w,NFFT); %dtft calculation from FFT
ffw_x_w_dB = 20 * log10(abs(fft_x_w) + 0.0000000000001); %magnitude of FFT, small positive value in log10() argument to avoid log10(0)

f = linspace(0,1,NFFT); %frequency from 0 to 1, NFFT indices

figure;
hold on;
plot(f,ffw_x_w_dB,'DisplayName','FFT magnitude'); %plots discrete time frequency 'f' vs magnitude
ylim([-20 50]);
plot([f0 f0], ylim,'--k', 'DisplayName','f_0 = 0.3308'); %shows f0
xlabel('DIscrete time frequency f');
ylabel('Magnitude, dB');
legend('show');

hold off;
```

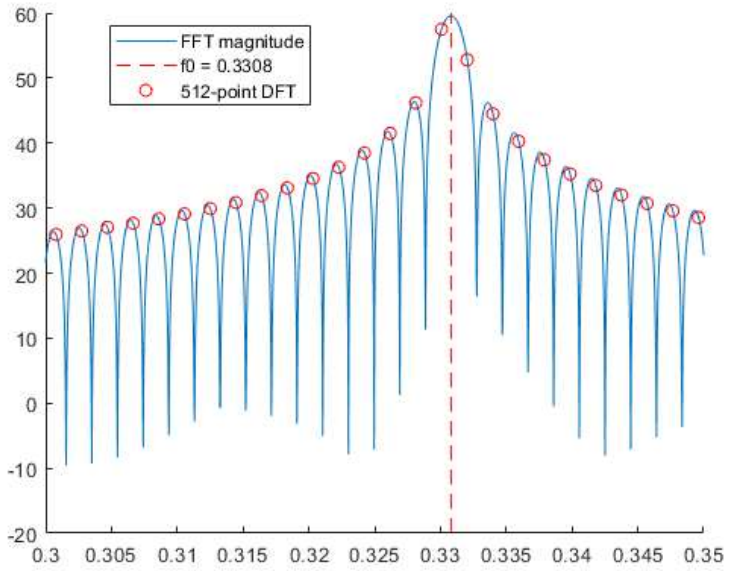


3b, copied 3a code

```
fprintf('NEW INSTANCE \n');

A = 3.7;
f0 = 0.3308;
N = 512;
NFFT = 32768;
n = linspace(0,N-1,N);
x_w = A * cos(2*pi()*f0*n);
fft_x_w = fft(x_w,NFFT);
ffw_x_w_dB = 20 * log10(abs(fft_x_w) + 0.0000000000001);
f = linspace(0,1,NFFT);
figure;
hold on;
plot(f,ffw_x_w_dB,'DisplayName','FFT magnitude');
ylim([-20 60]);
xlim([0.30 0.35]);
```

```
plot([f0 f0], ylim,'--r','DisplayName','f0 = 0.3308'); %shows f0
%no hold-off here, is instead below to include DFT on single plot

%end of 3a code

ffw_x_w_DFT = fft(x_w); %512-point DFT calculation
fft_x_w_DFT_dB = 20 * log10(abs(ffw_x_w_DFT) + 0.00000000000000000001); %compute 32768 point FFT


f_DFT = linspace(0,1,N+1);%aligns 512 DFT points to 32768 point DFT curve


plot(f_DFT(1:N),fft_x_w_DFT_dB,'ro','MarkerSize',6,'DisplayName','512-point DFT');

legend('show', 'Location','best');




ylim([-20 60]); %dB only has so much of a range
xlim([0.30 0.35]);%discrete time frequency range specified in problem






hold off;
```

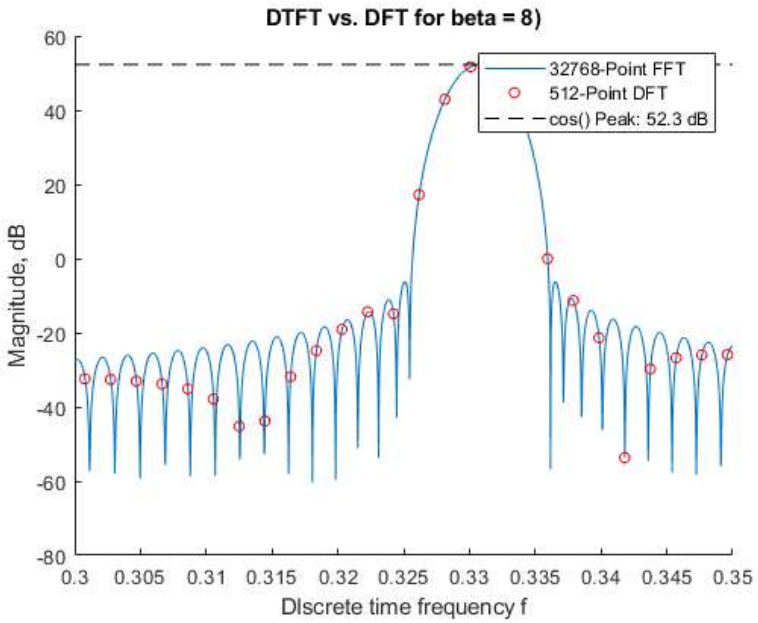
NEW INSTANCE



3c

```
w = kaiser(N,8); %N = 512, 8 = beta

x_w = x .* w'; %applies kaiser window to x[n]

%compute DTFT
DTFT_x_w = fft(x_w, NFFT);
DTFT_x_w_dB = 20 * log10(abs(DTFT_x_w) + 0.000000000000001);

%compute 512-point DFT
DFT_x_w = fft(x_w);
DFT_x_w_dB = 20 * log10(abs(DFT_x_w) + 0.000000000000001);

%defining frequency axes
DTFT_f = linspace(0,1,NFFT);
DFT_f = linspace(0,(N-1)/N,N);
```

```
%calculating cosine peak
peak = A * (sum(w)/N) *N/2; %amplitude of cos() times average value of window function times N/2 (since positive and negative frequencies split)
peak_dB = 20 * log10(peak); %'peak' is nonzero so no correction needed

figure;

hold on;

plot(DTFT_f,DTFT_x_w_dB,'DisplayName','32768-Point FFT'); %DTFT on plot
plot(DFT_f,DFT_x_w_dB,'ro','MarkerSize',5,'DisplayName','512-Point DFT'); %DFT on plot
plot([0.30 0.35], [peak_dB peak_dB], '--k','DisplayName',sprintf('cos() Peak: %.1f dB', peak_dB)); %cos() amplitude in dB on plot

xlim([0.3 0.35]);
xlabel('DIscrete time frequency f');
ylabel('Magnitude, dB');
title('DTFT vs. DFT for beta = 8)');
legend('show');


hold off;
```



Published with MATLAB® R2024b

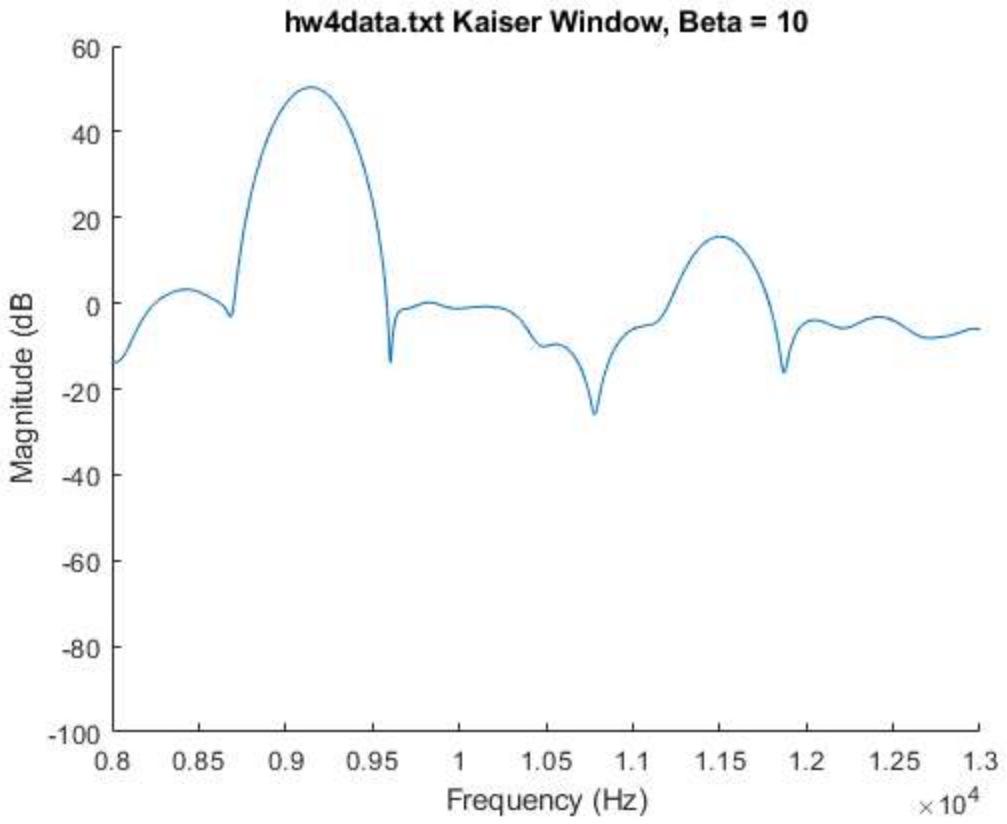
Contents


```
fprintf('NEW INSTANCE \n');

%getting data from text file
directory = 'C:\Users\vifro\OneDrive\Documents\MATLAB';
file_name = 'hw4data.txt';
file_path = fullfile(directory, file_name);
x = load(file_path);
x = x(:);
```

```
NEW INSTANCE
```

```
Fs = 75 * 1000; %75 ksps = sampling frequency
N = length(x); %is 512

NFFT = 32768;

f = linspace(0, Fs, NFFT);

beta = 10;

figure;
hold on;


    win = kaiser(N,beta);
    x_win = x .* win;

    fft_x_win = fft(x_win,NFFT);
    fft_x_win_dB = 20 * log10( abs(fft_x_win) + 0.0000000000001);


    plot(f, fft_x_win_dB);




xlim([8000 13000]);
ylim([-100 60]);

hold off;

xlabel('Frequency (Hz)');
ylabel('Magnitude (dB');
title('hw4data.txt Kaiser Window, Beta = 10');

%%calculate where gain is maximum, getting frequency

ind_large = find(f >9000 & f < 10000);
```

```
[M_large, I_large] = max(fft_x_win_dB(ind_large));
f_large = f(ind_large(I_large));



ind_small = find(f > 11000 & f < 12000);
[M_small, I_small] = max(fft_x_win_dB(ind_small));
f_small = f(ind_small(I_small));
```



calculate amplitude in V

```
fprintf('NEW INSTANCE \n');

%multiply by 2 because symmetric
%divide by window size to account for tapering by window

A_large = 2 * abs(fft_x_win(ind_large(I_large))) / sum(win);

A_small = 2 * abs(fft_x_win(ind_small(I_small))) / sum(win);
```

```
NEW INSTANCE
```

```
fprintf('Large Component Frequency, Hz = %.4f\n',f_large);
fprintf('Large Component Amplitude, V = %.4f\n', A_large);

fprintf('Small Component Frequency, Hz = %.4f\n',f_small);
fprintf('Small Component Amplitude, V = %.4f\n', A_small);
```

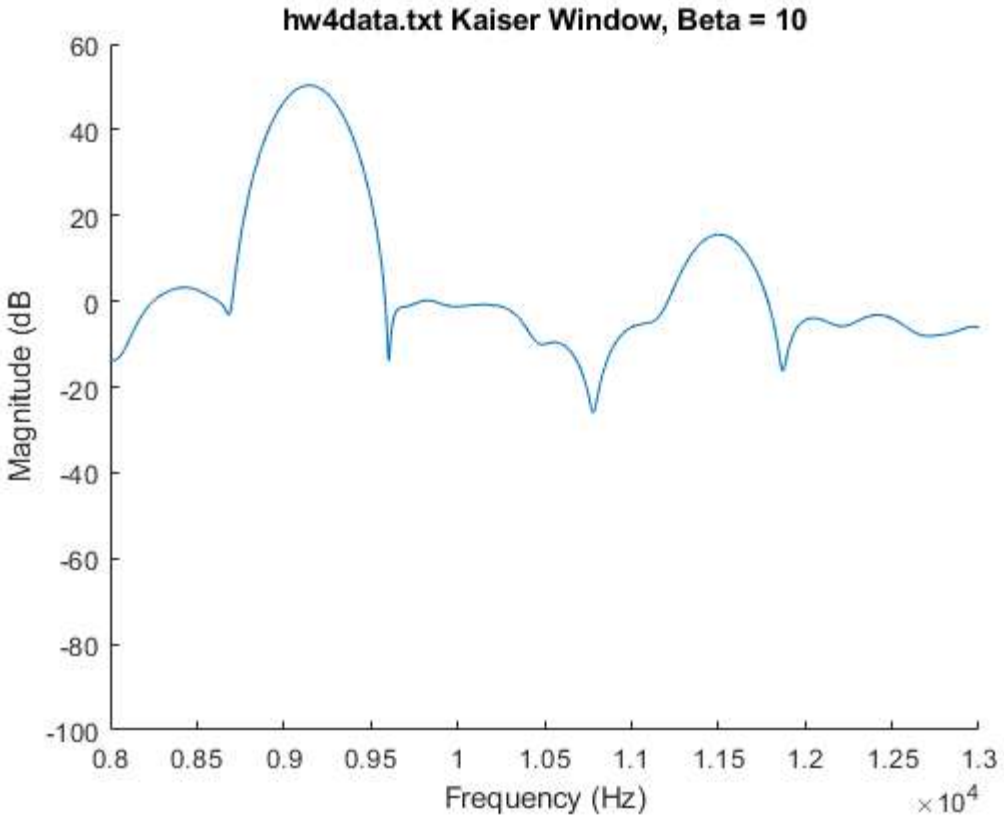
```
Large Component Frequency, Hz = 9146.3973
Large Component Amplitude, V = 3.2987
Small Component Frequency, Hz = 11503.9521
Small Component Amplitude, V = 0.0598
```

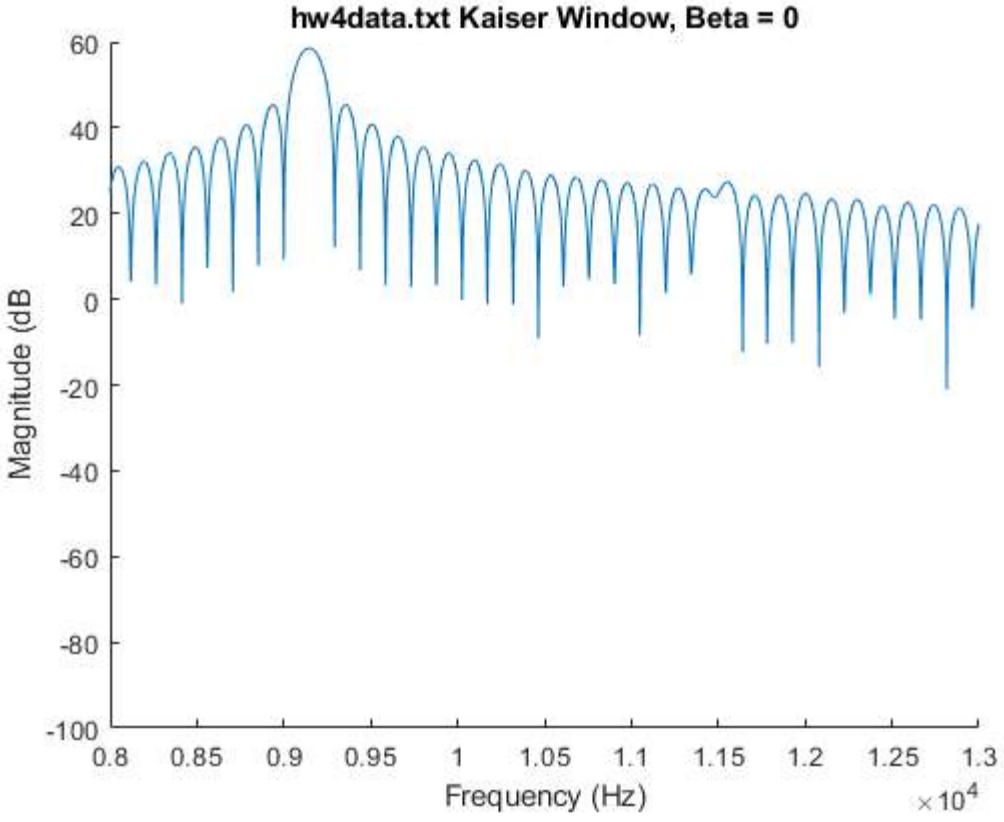
Testing out different beta = 0, see if same result (it's the same, mostly)

```
%getting data from text file
directory = 'C:\Users\vifro\OneDrive\Documents\MATLAB';
file_name = 'hw4data.txt';
file_path = fullfile(directory, file_name);
x = load(file_path);
x = x(:);




Fs = 75 * 1000; %75 ksps = sampling frequency
N = length(x); %is 512

NFFT = 32768;

f = linspace(0, Fs, NFFT);

beta = 0;

figure;
hold on;


    win = kaiser(N,beta);
    x_win = x .* win;

    fft_x_win = fft(x_win,NFFT);
    fft_x_win_dB = 20 * log10( abs(fft_x_win) + 0.0000000000001);


    plot(f, fft_x_win_dB);




xlim([8000 13000]);
ylim([-100 60]);

hold off;

xlabel('Frequency (Hz)');
ylabel('Magnitude (dB');
title('hw4data.txt Kaiser Window, Beta = 0');

%calculate where gain is maximum, getting frequency

ind_large = find(f >9000 & f < 10000);
[M_large, I_large] = max(fft_x_win_dB(ind_large));
f_large = f(ind_large(I_large));
```

```
ind_small = find(f > 11000 & f < 12000);
[M_small, I_small] = max(fft_x_win_dB(ind_small));
f_small = f(ind_small(I_small));

% calculate amplitude in V

fprintf('NEW INSTANCE \n');

%multiply by 2 because symmetric
%divide by window size to account for tapering by window

A_large = 2 * abs(fft_x_win(ind_large(I_large))) / sum(win);

A_small = 2 * abs(fft_x_win(ind_small(I_small))) / sum(win);



fprintf('Beta = 0, Large Component Frequency, Hz = %.4f\n',f_large);
fprintf('Beta = 0, Large Component Amplitude, V = %.4f\n', A_large);

fprintf('Beta = 0, Small Component Frequency, Hz = %.4f\n',f_small);
fprintf('Beta = 0, Small Component Amplitude, V = %.4f\n', A_small);
```

```
NEW INSTANCE
Beta = 0, Large Component Frequency, Hz = 9146.3973
Beta = 0, Large Component Amplitude, V = 3.3016
Beta = 0, Small Component Frequency, Hz = 11552.0188
Beta = 0, Small Component Amplitude, V = 0.0907
```





Published with MATLAB® R2024b